aws

Unable to locate subtitle

AWS Glue DataBrewデベロッパーガイド

AWS Glue DataBrewデベロッパーガイド: ***Unable to locate subtitle***

Table of Contents

とはAWS Glue DataBrew

AWS Glue DataBrewは、ユーザーがコードを記述せずにデータをクリーンアップおよび正規化できるようにするビジュアルデータ準備ツールです。DataBrew を使用すると、カスタム開発のデータ準備と比較して、分析と機械学習 (ML) のためのデータ準備にかかる時間を最大 80% 短縮できます。250 を超える既製の変換から選択して、異常のフィルタリング、データの標準形式への変換、無効な値の修正など、データ準備タスクを自動化できます。

DataBrew を使用すると、ビジネスアナリスト、データサイエンティスト、データエンジニアは、より簡単にコラボレーションして raw データからインサイトを得ることができます。DataBrew はサーバーレスであるため、技術的なレベルに関係なく、クラスターの作成やインフラストラクチャの管理を必要とせずに、テラバイトの未加工データを探索して変換できます。

直感的な DataBrew インターフェイスを使用すると、raw データをインタラクティブに検出、視覚化、クリーンアップ、変換できます。DataBrew は、検出が難しく、修正に時間がかかるデータ品質の問題を特定するのに役立つ賢明な提案を行います。DataBrew がデータを準備することで、時間を使って結果に対応し、より迅速に反復処理できます。変換をレシピのステップとして保存し、後で他のデータセットで更新または再利用して、継続的にデプロイできます。

次の図は、DataBrew が高レベルでどのように機能するかを示しています。

DataBrew を使用するには、プロジェクトを作成し、データに接続します。プロジェクトワークスペースには、グリッドのようなビジュアルインターフェイスにデータが表示されます。ここでは、データを調べ、値の分布とグラフを表示して、そのプロファイルを理解できます。

データを準備するには、250 を超えるpoint-and-click変換から選択できます。これには、null の削除、欠損値の置換、スキーマの不整合の修正、関数に基づく列の作成などが含まれます。変換を使用して自然言語処理 (NLP) 手法を適用し、文をフレーズに分割することもできます。即時プレビューでは、変換前と変換後のデータの一部が表示されるため、データセット全体に適用する前にレシピを変更できます。

DataBrew がデータセットでレシピを実行すると、出力は Amazon Simple Storage Service (Amazon S3) に保存されます。準備済みデータセットが Amazon S3 にクリーンアップされると、別のデータストレージまたはデータ管理システムがそのデータセットを取り込むことができます。

の主要な概念と用語AWS Glue DataBrew

以下に、 の主要な概念と用語の概要を示しますAWS Glue DataBrew。このセクションを読んだら、の開始方法AWS Glue DataBrew「」を参照してください。プロジェクトの作成、データセットの接続、ジョブの実行のプロセスについて説明します。

トピック

- プロジェクト
- データセット
- レシピ
- ジョブ
- データリネージュ
- データプロフィール

プロジェクト

DataBrew のインタラクティブデータ準備ワークスペースはプロジェクトと呼ばれます。データプロジェクトを使用して、データ、変換、スケジュールされたプロセスなどの関連項目のコレクションを管理します。プロジェクトの作成の一環として、作業するデータセットを選択または作成します。次に、DataBrew が実行する一連の手順であるレシピを作成します。これらのアクションは、raw データをデータパイプラインで使用できる形式に変換します。

データセット

データセットとは、列またはフィールドに分割された行またはレコードのデータセットを意味します。DataBrew プロジェクトを作成するときは、変換または準備するデータに接続またはアップロードします。DataBrew は、フォーマットされたファイルからインポートされた任意のソースからのデータを操作でき、増え続けるデータストアのリストに直接接続します。

DataBrew の場合、データセットはデータへの読み取り専用接続です。DataBrew は、データを参照するために一連の説明メタデータを収集します。DataBrew が実際のデータを変更または保存することはできません。わかりやすくするために、データセットを使用して実際のデータセットと DataBrew が使用するメタデータの両方を参照します。

レシピ

DataBrew では、レシピは DataBrew が実行するデータの一連の指示またはステップです。レシピには多くのステップを含めることができ、各ステップには多くのアクションを含めることができます。ツールバーの変換ツールを使用して、データに加えるすべての変更を設定します。その後、レシピの完成した製品を表示する準備ができたら、このジョブを DataBrew に割り当ててスケジュールします。DataBrew はデータ変換に関する指示を保存しますが、実際のデータは保存しません。レシピは他のプロジェクトでダウンロードして再利用できます。レシピの複数のバージョンを発行することもできます。

ジョブ

DataBrew は、レシピの作成時に設定した手順を実行して、データを変換するジョブを実行します。これらの手順を実行するプロセスはジョブと呼ばれます。ジョブは、プリセットスケジュールに従ってデータレシピを実行に移すことができます。ただし、スケジュールに限定されません。オンデマンドでジョブを実行することもできます。一部のデータをプロファイリングする場合は、レシピは必要ありません。その場合は、プロファイルジョブを設定してデータプロファイルを作成することができます。

データリネージュ

DataBrew は、ビジュアルインターフェイスでデータを追跡し、データリネージュと呼ばれるオリジンを決定します。このビューは、データが元の場所とは異なるエンティティをどのように流れるかを示しています。そのオリジン、影響を受けた他のエンティティ、時間の経過とともに何が起こったか、保存場所を確認できます。

データプロフィール

データをプロファイルすると、DataBrew はデータプロファイルと呼ばれるレポートを作成します。この概要では、コンテンツのコンテキスト、データの構造、その関係など、データの既存の形状について説明します。データプロファイルジョブを実行することで、任意のデータセットのデータプロファイルを作成できます。

製品およびサービスの統合

このセクションを使用して、DataBrew と統合する製品やサービスを確認します。

DataBrew は、ネットワーク、管理、ガバナンスのために以下のAWSサービスと連携します。

- Amazon CloudFront
- AWS CloudFormation
- AWS CloudTrail
- Amazon CloudWatch
- AWS Step Functions

DataBrew は、次のAWSデータレイクとデータストアで動作します。

- AWS Lake Formation
- Amazon S3

DataBrew は、データをアップロードするための次のファイル形式と拡張子をサポートしています。

Format	ファイル拡張子 (オプション)	圧縮ファイルの拡張子 (必須)

カンマ区切り値	`.csv`	`.gz`
		`.snappy`
		`.lz4`
		`.bz2`
		`.deflate`

Format	ファイル拡張子 (オプション)	圧縮ファイルの拡張子 (必須)
Microsoft Excel ワークブック	`.xlsx`	圧縮サポートなし

JSON (JSON ドキュメントと JSON 行)	`.json, .jsonl`	`.gz`
		`.snappy`
		`.lz4`
		`.bz2`
		`.deflate`

Apache ORC	`.orc`	`.zlib` `.snappy`

Apache Parquet	`.parquet`	`.gz`
		`.snappy`
		`.lz4`

DataBrew は出力ファイルを Amazon S3 に書き込み、次のファイル形式と拡張子をサポートします。

Format	ファイル拡張子 (非圧縮)	ファイル拡張子 (圧縮)
カンマ区切り値	`.csv`	`.csv.snappy` , `.csv.gz`, `.csv.lz4`, `csv.bz2`, `.csv.deflate` , `csv.br`
タブ区切り値	`.csv`	`.tsv.snappy` , `.tsv.gz`, `.tsv.lz4`, `tsv.bz2`, `.tsv.deflate` , `tsv.br`
Apache Parquet	`.parquet`	`.parquet.snappy` , `.parquet.gz` , `.parquet.`

Format	ファイル拡張子 (非圧縮)	ファイル拡張子 (圧縮)
		`lz4` , `.parquet.lzo` , `.parquet.br`
AWS Glue Parquet	サポートされていません	`.glue.parquet.snappy`
Apache Avro	`.avro`	`.avro.snappy` , `.avro.gz`, `.avro.lz4` , `.avro.bz2` , `.avro.deflate` , `.avro.br`
Apache ORC	`.orc`	`.orc.snappy` , `.orc.lzo`, `.orc.zlib`
XML	`.xml`	`.xml.snappy` , `.xml.gz`, `.xml.lz4`, `.xml.bz2`, `.xml.deflate` , `.xml.br`
JSON (JSON Lines 形式のみ)	`.json`	`.json.snappy` , `.json.gz`, `.json.lz4` , `json.bz2`, `.json.deflate` , `.json.br`
Tableau Hyper	サポートされていません	該当しない

セットアップAWS Glue DataBrew

の使用を開始する前にAWS Glue DataBrew、いくつかのアクセス許可、ユーザー、ロールを設定する必要があります。まず、次の手順を実行します。

1. 必要に応じてAWSアカウントにサインアップし、ユーザーが DataBrew を実行できるようにする AWS Identity and Access Management(IAM) ポリシーを作成します。
   - 新しいAWSアカウントにサインアップし、ユーザーを追加します。詳細については、「新しいAWSアカウントのセットアップ」を参照してください。
   - コンソールユーザーの IAM ポリシーの追加。 これらのアクセス許可を持つユーザーは、 で DataBrew にアクセスできますAWS マネジメントコンソール。
   - IAM ロールのデータリソースに対するアクセス許可の追加。 これらのアクセス許可を持つ IAM ロールは、ユーザーに代わってデータにアクセスできます。

   ユーザー、ロール、ポリシーを作成するには、IAM 管理者である必要があります。
2. DataBrew のユーザーまたはグループを追加します。正しいアクセス許可がアタッチされたユーザーまたはグループは、 コンソールで DataBrew にアクセスできます。
3. DataBrew のデータにアクセスするためのアクセス許可を持つロールを追加します。正しいアクセス許可を持つロールは、ユーザーに代わってデータにアクセスできます。

新しいAWSアカウントのセットアップ

AWSアカウントがない場合は、AWSアカウントにサインアップし、IAM 管理者ユーザーを作成します。

がない場合はAWS アカウント、次の手順を実行して作成します。

にサインアップするにはAWS アカウント

1. https://portal.aws.amazon.com/billing/signup を開きます。
2. オンラインの手順に従います。

   サインアップ手順の一環として、電話またはテキストメッセージを受け取り、電話キーパッドで検証コードを入力します。

   にサインアップするとAWS アカウント、 AWS アカウントのルートユーザー が作成されます。ルートユーザーには、アカウントのすべてのAWS のサービスとリソースへのアクセス権があり

ます。セキュリティベストプラクティスとして、ユーザーに管理アクセス権を割り当て、ルートユーザーアクセスが必要なタスクの実行にはルートユーザーのみを使用するようにしてください。

管理者ユーザーを作成するには、以下のいずれかのオプションを選択します。

管理者を管理する方法を 1 つ選択します	目的	方法	以下の操作も可能
IAM アイデンティティセンター内 (推奨)	短期の認証情報を使用してAWSにアクセスします。 これはセキュリティのベストプラクティスと一致しています。ベストプラクティスの詳細については、「IAM ユーザーガイド」の「IAM でのセキュリティのベストプラクティス」を参照してください。	AWS IAM アイデンティティセンターユーザーガイドの「開始方法」の手順に従います。	AWS Command Line Interface ユーザーガイドの を使用する AWS CLIように を設定AWS IAM アイデンティティセンターして、プログラムによるアクセスを設定します。
IAM 内 (非推奨)	長期認証情報を使用してAWSにアクセスします。	IAM ユーザーガイドの「緊急アクセス用の IAM ユーザーを作成する」の手順に従います。	IAM ユーザーガイドの「IAM ユーザーのアクセスキーを管理する」の手順に従って、プログラムによるアクセスを設定します。

詳細については、IAM ユーザーガイドにある下記のトピックを参照してください。

- IAM とは

- IAM のセットアップ
- 管理ユーザーとグループの作成 (コンソール)

のセットアップAWS CLI

JupyterLab または DataBrew API を使用する場合は、必ず () をインストールしてくださいAWS Command Line InterfaceAWS CLI。DataBrew コンソールを使用したり、「開始方法」の演習のステップを実行したりする必要はありません。

をセットアップするにはAWS CLI

1. 以下のステップを使用してAWS CLI、 をダウンロードして設定します。
   - AWS CLIのインストール
   - 設定の基本
2. コマンドプロンプトで次の DataBrew コマンドを入力して、セットアップを確認します。

   ```
   aws databrew help
   ```

   このステートメントがエラーaws: error: argument command: Invalid choice「」を返し、その後にサービスの長いリストが返された場合は、 をアンインストールAWS CLIしてから再インストールします。このアクションは、既存の設定を上書きしません。

AWS CLIコマンドは、パラメータまたはプロファイルで設定しない限り、設定からデフォルトのAWSリージョンを使用します。各コマンドに --regionパラメータを追加できます。

必要に応じて、 ~/.aws/configまたは %UserProfile%/.aws/config (Microsoft Windows の場合) に名前付きプロファイルを追加できます。次の例に示すように、名前付きプロファイルは他の設定を保持することもできます。

```
[profile databrew]
aws_access_key_id = ACCESS-KEY-ID-OF-IAM-USER
aws_secret_access_key = SECRET-ACCESS-KEY-ID-OF-IAM-USER
region = us-east-1
output = text
```

AWS Identity and Access Management(IAM) アクセス許可の設定

開始する前に、IAM でいくつかの設定を行う必要があります。管理者であるか、管理者からのサポートが必要です。ただし、管理者権限を持つアカウントがある場合は、これらのタスクを自分で実行できます。このセクションでは、各タスクの簡単な手順を確認できます。

以下は、実行する必要があることの概要です。

- このプロセスの一環として、ユーザーを追加します。新しいユーザーを追加する必要はありません。既存のユーザーを使用できます。DataBrew アクセス許可をアタッチして、ユーザーが DataBrew コンソールを開くことができるようにします。
- IAM ロールを作成します。ロールは、特定のアクションを許可し、使用時に制限内でアクセス許可を付与します。たとえば、AWSアカウントのユーザーに対してのみ機能します。制限は後で追加できます。
- 必要な IAM ポリシーを作成します。ポリシーは、ユーザーが実行できる操作のリストです。ポリシーを作成するには、別のコンソールページを開き、ダウンロードしたファイルのテキストを貼り付けます。

> **Note**
>
> ここで提供するのは、基本的なセットアップ情報です。時間をかけてアクセス許可をカスタマイズし、セキュリティとコンプライアンスのニーズを満たすことをお勧めします。サポートが必要な場合は、管理者またはAWSサポートにお問い合わせください。

必要なアクセス許可を追加するには

1. 以下を実行して、ユーザーが DataBrew を実行できるようにする IAM ポリシーを作成します。

   - コンソールユーザーのカスタム IAM ポリシーを追加します。カスタムポリシーが必要ない場合は、代わりにAWS管理ポリシーを選択できます。ステップ 2 でユーザーに追加するだけです。これらのアクセス許可を持つユーザーは、DataBrew サービスコンソールにアクセスできます。
   - データリソースのアクセス許可を追加します。これらのアクセス許可を持つ IAM ロールは、ユーザーに代わってデータにアクセスできます。

   ユーザー、ロール、ポリシーを作成するには、管理者である必要があります。

2. DataBrew のユーザーまたはグループを追加します。正しいアクセス許可がアタッチされたユーザーまたはグループは、DataBrew コンソールにアクセスできます。
3. DataBrew のデータにアクセスするためのアクセス許可を持つロールを追加します。正しいアクセス許可を持つロールは、ユーザーに代わってデータにアクセスできます。

DataBrew の IAM ポリシーの設定

IAM ポリシーを使用してアクセス許可を管理します。ポリシーを使用すると、関連するアクセス許可を一度に 1 つずつ追加するのではなく、一度にすべて簡単に追加できます。

指定したのと同じ名前を使用してポリシーを作成することをお勧めします。ドキュメント全体で、これらのポリシーには以下に示す名前が使用されます。これらの名前を使用すると、AWSサポートに連絡する必要がある場合も簡単になります。ただし、ポリシー名とそのコンテンツの両方を変更できます。IAM ポリシーの詳細については、IAM ユーザーガイドの「カスタマー管理ポリシーの作成」を参照してください。

DataBrew を使用するために必要なポリシーを作成したら、ユーザーとロールにアタッチします。これを行う方法については、このセクションの後半で説明します。

トピック

- コンソールユーザーの IAM ポリシーの追加
- IAM ロールのデータリソースに対するアクセス許可の追加
- DataBrew の IAM ポリシーの設定

コンソールユーザーの IAM ポリシーの追加

のユーザーのアクセス許可の設定AWS マネジメントコンソールはオプションですが、コンソールへのアクセスが必要な場合は、まずこのステップを実行します。

コンソールで DataBrew にアクセスするアクセス許可を設定するには、次のいずれかを選択します。

- によって管理されるポリシーを使用しますAWS。 `AwsGlueDataBrewFullAccessPolicy`このオプションを選択した場合は、次のポリシー に進みますIAM ロールのデータリソースに対するアクセス許可の追加。
- このセクション「」で説明されているポリシーを作成しますAwsGlueDataBrewCustomUserPolicy。このオプションを使用すると、追加のカスタムセキュリティ要件を使用してポリシーをカスタマイズできます。

次のポリシーは、DataBrew コンソールの実行に必要なアクセス許可を付与します。これらのアクセス許可は、IAM を使用して付与します。

DataBrew の AwsGlueDataBrewCustomUserPolicy IAM ポリシーを定義するには (コンソール)

1. IAM ポリシーの JSON AwsGlueDataBrewCustomUserPolicy をダウンロードします。
2. にサインインAWS マネジメントコンソールし、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
3. ナビゲーションペインで、ポリシー を選択してください。
4. ポリシーごとに、ポリシーの作成を選択します。
5. ポリシーの作成画面で、JSON タブに移動します。
6. ダウンロードしたポリシー JSON ステートメントをコピーします。エディタのサンプルステートメントに貼り付けます。
7. ポリシーがアカウント、セキュリティ要件、および必要なAWSリソースにカスタマイズされていることを確認します。変更を加える必要がある場合は、エディタで変更を加えることができます。
8. [ポリシーの確認] を選択します。

DataBrew の AwsGlueDataBrewCustomUserPolicy IAM ポリシーを定義するには (AWS CLI)

1. IAM ポリシーの JSON AwsGlueDataBrewCustomUserPolicy をダウンロードします。
2. 前の手順の最初のステップで説明したように、ポリシーをカスタマイズします。
3. 次のコマンドを実行してポリシーを作成します。

```
aws iam create-policy --policy-name AwsGlueDataBrewCustomUserPolicy --policy-
document file://iam-policy-AwsGlueDataBrewCustomUserPolicy.json
```

IAM ロールのデータリソースに対するアクセス許可の追加

データに接続するには、 ユーザーに代わって渡すことができる IAM ロールAWS Glue DataBrewが必要です。以下は、後で IAM ロールにアタッチするポリシーを作成する方法を示しています。

このAwsGlueDataBrewDataResourcePolicyポリシーは、DataBrew を使用してデータに接続するために必要なアクセス許可を付与します。Amazon S3 のオブジェクトにアクセスするなど、別の

AWSリソースのデータにアクセスするオペレーションの場合、DataBrew にはユーザーに代わってリソースにアクセスするためのアクセス許可が必要です。 Amazon S3

DataBrew の AwsGlueDataBrewDataResourcePolicy IAM ポリシーを定義するには (コンソール)

1. の JSON をダウンロードしますAwsGlueDataBrewDataResourcePolicy。
2. にサインインAWS マネジメントコンソールし、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
3. ナビゲーションペインで、ポリシー を選択してください。
4. ポリシーごとに、ポリシーの作成を選択します。
5. ポリシーの作成画面で、JSON タブに移動します。
6. ダウンロードしたポリシー JSON ステートメントをコピーします。エディタのサンプルステートメントに貼り付けます。
7. ポリシーがアカウント、セキュリティ要件、および必要なAWSリソースにカスタマイズされていることを確認します。変更を加える必要がある場合は、エディタで変更を加えることができます。
8. [ポリシーの確認] を選択します。

DataBrew の AwsGlueDataBrewDataResourcePolicy IAM ポリシーを定義するには (AWS CLI)

1. の JSON をダウンロードしますAwsGlueDataBrewDataResourcePolicy。
2. 前の手順の最初のステップで説明したように、ポリシーをカスタマイズします。
3. 次のコマンドを実行してポリシーを作成します。

```
aws iam create-policy --policy-name AwsGlueDataBrewDataResourcePolicy --policy-
document file://iam-policy-AwsGlueDataBrewDataResourcePolicy.json
```

DataBrew の IAM ポリシーの設定

DataBrew で使用できる IAM ポリシーの詳細と例を以下に示します。基本ポリシーの詳細については、こちらを参照してください。さらに、DataBrew を使用する必要がない例が他にもあります。これらは、特定の状況で使用する追加の設定です。

トピック

- AwsGlueDataBrewCustomUserPolicy

- AwsGlueDataBrewDataResourcePolicy
- DataBrew で Amazon S3 オブジェクトを使用するための IAM ポリシー
- DataBrew で暗号化を使用する IAM ポリシー

AwsGlueDataBrewCustomUserPolicy

このAwsGlueDataBrewCustomUserPolicyポリシーは、DataBrew コンソールを使用するために必要なほとんどのアクセス許可を付与します。このポリシーで指定されているリソースの一部は、DataBrew で使用されるサービスを参照します。これにはAWS Glue Data Catalog、、Amazon S3 バケット、Amazon CloudWatch Logs、リソースAWS KMSの名前が含まれます。これは、 という名前のAWS管理ポリシーに似ていますAwsGlueDataBrewFullAccessPolicy。

次の表は、このポリシーによって付与されたアクセス権限を示しています。

アクション	[リソース]	説明
"databrew:*"	"*"	すべての DataBrew API オペレーションを実行するアクセス許可を付与します。
"glue:GetDatabases" "glue:GetPartitions" "glue:GetTable" "glue:GetTables" "glue:GetDataCatalogEncryptionSettings"	"*"	AWS Glueデータベースとテーブルのリストを許可します。
"dataexchange:ListDataSets" "dataexchange:ListDataSetRevisions" "dataexchange:ListRevisionAssets"	"*"	データセット内のAWS Data Exchange リソースのリストを許可します。

アクション	[リソース]	説明
"dataexchange:CreateJob" "dataexchange:StartJob" "dataexchange:GetJob"		
"kms:DescribeKey" "kms:ListKeys" "kms:ListAliases"	"*"	ジョブ出力の暗号化に使用するAWS KMSキーのリストを許可します。
"kms:GenerateDataKey"	"arn:aws:kms:::key/key_ids"	ジョブ出力の暗号化を許可します。
"s3:ListAllMyBuckets" "s3:GetBucketCORS" "s3:GetBucketLocation" "s3:GetEncryptionConfiguration"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	プロジェクト、データセット、ジョブの Amazon S3 バケットのリストを許可します。S3 への出力ファイルの送信を許可します。
"sts:GetCallerIdentity"	"*"	現在の発信者に関する情報を取得します。
"cloudtrail:LookupEvents",	"*"	データセット (データ系統) のAWS CloudTrailイベントの一覧表示を許可します。
"iam:ListRoles" "iam:GetRole"	"*"	プロジェクトとジョブに使用する IAM ロールの一覧表示を許可します。

AwsGlueDataBrewDataResourcePolicy

このAwsGlueDataBrewDataResourcePolicyポリシーは、データに接続し、DataBrew を設定するために必要なアクセス許可を付与します。

次の表は、このポリシーによって付与されたアクセス権限を示しています。

アクション	[リソース]	説明
"s3:GetObject"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	ファイルをプレビューできます。
"s3:PutObject" "s3:PutBucketCORS"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	S3 への出力ファイルの送信を許可します。
"s3:DeleteObject"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	DataBrew によって作成されたオブジェクトの削除を許可します。
"s3:ListBucket"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	プロジェクト、データセット、ジョブからの Amazon S3 バケットのリストを許可します。
"kms:Decrypt"	"arn:aws:kms:::key/key_ids"	暗号化されたデータセットの復号を許可します。
"kms:GenerateDataKey"	"arn:aws:kms:::key/key_ids"	ジョブ出力の暗号化を許可します。
"ec2:DescribeVpcEndpoints"	"*"	ジョブやプロジェクトを実行するときに、Virtual Private Cloud (VPCs) などの Amazon EC2 ネットワー

アクション	[リソース]	説明
"ec2:DescribeRouteTables" "ec2:DeleteNetworkInterface" "ec2:DescribeNetworkInterfaces" "ec2:DescribeSecurityGroups" "ec2:DescribeSubnets" "ec2:DescribeVpcAttribute" "ec2:CreateNetworkInterface"		ク項目のセットアップを許可します。
"ec2:DeleteNetworkInterface"	"*"	VPC 内のネットワークインターフェイスの削除を許可します。

アクション	[リソース]	説明

`"ec2:CreateTags"`
 	
`"ec2:DeleteTags"`
	`"arn:aws:ec2:::network-interface/*", "arn:aws:ec2:::security-group/*"`	タグの作成と削除を許可します。
		
		VPC が有効になっているAWS Glueデータカタログを使用する場合は、これらのアクセス許可が必要です。DataBrew はデータを に渡AWS Glueしてジョブとプロジェクトを実行します。これらのアクセス許可により、開発エンドポイント用に作成された Amazon EC2 リソースのタグ付けが可能になります。 はAmazon EC2 ネットワークインターフェイス、セキュリティグループ、インスタンスをでAWS Glueタグ付けしますaws-glue-service-resource 。

`"logs:CreateLogGroup"`
 	
`"logs:CreateLogStream"`
 	
`"logs:PutLogEvents"`
	`"arn:aws:logs:::log-group:/aws-glue-databrew/*"`	Amazon CloudWatch Logs へのログの書き込みを許可する
		
		DataBrew は、名前が で始まるロググループにログを書き込みますaws-glue-databrew 。

アクション	[リソース]	説明
"lakeformation:GetDataAccess"	"*"	へのアクセスを許可します。ただしAWS Lake Formation、も許可"Glue":"GetTable" されます。 Lake Formation を使用するには、Lake Formation コンソールでさらに設定する必要があります。

DataBrew で Amazon S3 オブジェクトを使用するための IAM ポリシー

このAwsGlueDataBrewSpecificS3BucketPolicyポリシーは、管理者以外のユーザーに代わって S3 にアクセスするために必要なアクセス許可を付与します。

ポリシーを次のようにカスタマイズします。

1. 使用するパスを指すように、ポリシー内の Amazon S3 パスを置き換えます。サンプルテキストでは、 *BUCKET-NAME-1/SPECIFIC-OBJECT-NAME*は特定のオブジェクトまたはファイルを表します。 は、パス名が で始まるすべてのオブジェクト (*) *BUCKET-NAME-2/*を表しますBUCKET-NAME-2/。これらを更新して、使用しているバケットに名前を付けます。
2. (オプション) Amazon S3 パスでワイルドカードを使用して、アクセス許可をさらに制限します。詳細については、IAM ユーザーガイドのIAM ポリシーエレメント: 変数およびタグを参照してください。

   セキュリティのベストプラクティス: 他のAWSアカウントの同様の名前の Amazon S3 バケットへの不正アクセスを防ぐには、ポリシーに aws:ResourceAccount条件キーを含めます。これにより、DataBrew はワイルドカードリソース ARNs を使用している場合でも、自分のAWSアカウント内のバケットにのみアクセスできます。ポリシーステートメントに次の条件を追加します。

```
"Condition": {
"StringEquals": {
"aws:ResourceAccount": "123456789012"
}
}
```

を実際のAWSアカウント ID 123456789012に置き換えます。

これの一環として、アクション s3:PutObjectおよび のアクセス許可を制限する場合がありますs3:PutBucketCORS。これらのアクションはDataBrew プロジェクトを作成するユーザーに対してのみ必要です。これらのユーザーは S3 に出力ファイルを送信できる必要があるためです。

詳細と Amazon S3 の IAM ポリシーに追加できる内容の例については、Amazon S3 デベロッパーガイドの「バケットポリシーの例」を参照してください。 Amazon S3

次の表は、このポリシーによって付与されたアクセス権限を示しています。

アクション	[リソース]	説明
"s3:GetObject"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	ファイルをプレビューできます。
"s3:PutObject" "s3:PutBucketCORS"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	S3 への出力ファイルの送信を許可します。
"s3:DeleteObject"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	オブジェクトの削除を許可します。

DataBrew の AwsGlueDataBrewSpecificS3BucketPolicy IAM ポリシーを定義するには (コンソール)

1. IAM ポリシーの JSON AwsGlueDataBrewSpecificS3BucketPolicy をダウンロードします。
2. にサインインAWS マネジメントコンソールし、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
3. ナビゲーションペインで、ポリシー を選択してください。

4. ポリシーごとに、ポリシーの作成を選択します。
5. ポリシーの作成画面で、JSON タブに移動します。
6. ポリシー JSON ステートメントをエディタのサンプルステートメントに貼り付けます。
7. ポリシーがアカウント、セキュリティ要件、および必要なAWSリソースにカスタマイズされていることを確認します。変更を加える必要がある場合は、エディタで変更を加えることができます。
8. [ポリシーの確認] を選択します。

DataBrew の AwsGlueDataBrewSpecificS3BucketPolicy IAM ポリシーを定義するには (AWS CLI)

1. の JSON をダウンロードしますAwsGlueDataBrewSpecificS3BucketPolicy。
2. 前の手順の最初のステップで説明したように、ポリシーをカスタマイズします。
3. 次のコマンドを実行して、ポリシーを作成します。

```
aws iam create-policy --policy-name AwsGlueDataBrewSpecificS3BucketPolicy --policy-
document file://iam-policy-AwsGlueDataBrewSpecificS3BucketPolicy.json
```

DataBrew で暗号化を使用する IAM ポリシー

このAwsGlueDataBrewS3EncryptedPolicyポリシーは、管理者以外のユーザーに代わってAWS Key Management Service(AWS KMS) で暗号化された S3 オブジェクトにアクセスするために必要なアクセス許可を付与します。

ポリシーを次のようにカスタマイズします。

1. 使用するパスを指すように、ポリシー内の Amazon S3 パスを置き換えます。サンプルテキストでは、 *BUCKET-NAME-1/SPECIFIC-OBJECT-NAME*は特定のオブジェクトまたはファイルを表します。 は、パス名が で始まるすべてのオブジェクト (*) *BUCKET-NAME-2/*を表しますBUCKET-NAME-2/。これらを更新して、使用しているバケットに名前を付けます。
2. (オプション) Amazon S3 パスでワイルドカードを使用して、アクセス許可をさらに制限します。詳細については、「IAM ポリシーの要素: 変数とタグ」を参照してください。

   これの一環として、アクション s3:PutObjectおよび のアクセス許可を制限する場合がありますs3:PutBucketCORS。これらのアクションはDataBrew プロジェクトを作成するユーザーに対してのみ必要です。これらのユーザーは S3 に出力ファイルを送信できる必要があるためです。

詳細と Amazon S3 の IAM ポリシーに追加できる内容の例については、「バケットポリシーの例」を参照してください。

3. ファイルで次のリソース ARNs を見つけますToUseKms。

```
  "arn:aws:kms:AWS-REGION-NAME:AWS-ACCOUNT-ID-WITHOUT-DASHES:key/KEY-IDS",
"arn:aws:kms:AWS-REGION-NAME:AWS-ACCOUNT-ID-WITHOUT-DASHES:key/KEY-IDS"
```

4. サンプルAWSアカウントをAWSアカウント番号 (ハイフンなし) に変更します。
5. サンプルリストを変更して、代わりに使用する IAM ロールを一覧表示します。IAM ポリシーの範囲は、可能な限り最小限のアクセス許可セットにすることをお勧めします。ただし、サンプルデータで個人学習アカウントを使用している場合などは、ユーザーにすべての IAM ロールへのアクセスを許可できます。リストがすべての IAM ロールにアクセスできるようにするには、サンプルリストを 1 つのエントリ に変更します"arn:aws:iam::*111122223333*:role/*"。

次の表は、このポリシーによって付与されたアクセス権限を示しています。

アクション	[リソース]	説明
"s3:GetObject"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	ファイルをプレビューできます。
"s3:ListBucket"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	プロジェクト、データセット、ジョブからの Amazon S3 バケットのリストを許可します。
"s3:PutObject"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	S3 への出力ファイルの送信を許可します。
"s3:DeleteObject"	"arn:aws:s3:::bucket_name/*",	DataBrew によって作成されたオブジェクトの削除を許可します。

アクション	[リソース]	説明
	"arn:aws:s3:::bucket_name"	
"kms:Decrypt"	"arn:aws:kms:::key/key_ids"	暗号化されたデータセットの復号を許可します。
"kms:GenerateDataKey*"	"arn:aws:kms:::key/key_ids"	ジョブ出力の暗号化を許可します。

DataBrew の AwsGlueDataBrewS3EncryptedPolicy IAM ポリシーを定義するには (コンソール)

1. IAM ポリシーの JSON AwsGlueDataBrewS3EncryptedPolicy をダウンロードします。
2. にサインインAWS マネジメントコンソールし、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
3. ナビゲーションペインで、ポリシー を選択してください。
4. ポリシーごとに、ポリシーの作成を選択します。
5. ポリシーの作成画面で、JSON タブに移動します。
6. ポリシー JSON ステートメントをエディタのサンプルステートメントに貼り付けます。
7. ポリシーがアカウント、セキュリティ要件、および必要なAWSリソースにカスタマイズされていることを確認します。変更を加える必要がある場合は、エディタで変更を加えることができます。
8. [ポリシーの確認] を選択します。

DataBrew の AwsGlueDataBrewS3EncryptedPolicy IAM ポリシーを定義するには (AWS CLI)

1. の JSON をダウンロードしますAwsGlueDataBrewS3EncryptedPolicy。
2. 前の手順の最初のステップで説明したように、ポリシーをカスタマイズします。
3. 次のコマンドを実行して、ポリシーを作成します。

```
aws iam create-policy --policy-name AwsGlueDataBrewS3EncryptedPolicy --policy-
document file://iam-policy-AwsGlueDataBrewS3EncryptedPolicy.json
```

DataBrew アクセス許可を持つユーザーまたはグループの追加

アクセス許可を管理するために、ロールにポリシーを割り当て、ユーザーとグループにロールを割り当てます。詳細については、IAM ユーザーガイドの「IAM ID (ユーザー、グループ、ロール)」を参照してください。

開始する前に、アクセス許可を割り当てるユーザーが少なくとも 1 人必要です。

次の手順を使用して、DataBrew コンソールで作業したり、 CLI で DataBrew コマンドを実行したりする必要があるユーザーの DataBrew アクセス許可を設定します。

DataBrew アクセス許可を設定するには

1. AWS CLI DataBrew 用の やその他の開発ツールを使用するために、ユーザーがアクセスキーを作成します。
2. AWS マネジメントコンソールアクセスを有効にして、ユーザーがAWSコンソールを使用できるようにします。
3. DataBrew ユーザーまたはグループのロールを作成します。
4. 使用しているポリシーを選択します。次のいずれかを行います。
   - を作成した場合は`AwsGlueDataBrewCustomUserPolicy`、リストから選択します。
   - AWS 管理ポリシーを使用するには、リストから `AwsGlueDataBrewFullAccessPolicy` を選択します。
5. そのポリシーをロールに割り当てます。
6. ユーザーまたはグループが関連するロールを引き受けられるように、ロールの信頼関係を設定します。
   - グループを使用していない場合は、 ロールでユーザーを信頼します。
   - グループを使用している場合は、 ロールでグループを信頼し、ユーザーをグループに追加します。

データリソースのアクセス許可を持つ IAM ロールの追加

IAM ロールを使用して、一緒に割り当てられたポリシーを管理します。IAM ロールは、DataBrew ユーザーや DataBrew 自体など、特定のロールで動作するユーザーが使用できます。詳細については、IAM ユーザーガイドの IAM ロールを参照してください。

DataBrew プロジェクトがデータにアクセスするために必要な IAM ロールを作成するには、次の手順に従います。

必要な IAM ポリシーを DataBrew の新しい IAM ロールにアタッチするには

1. ナビゲーションペインで [Roles (ロール)]、[Create Role (ロールの作成)] の順に選択します。
2. 信頼されたエンティティのタイプを選択する で、カードラベル付きAWSサービスを選択します。
3. リストから DataBrew を選択し、次へ: アクセス許可を選択します。
4. **AwsGlueDataBrewDataResourcePolicy** 検索ボックスに (前のステップで作成した IAM ポリシー) を入力します。ポリシーを選択し、次へ: タグを選択します。
5. [次へ: レビュー] を選択します。
6. [Role name (ロール名)] に **AwsGlueDataBrewDataAccessRole** と入力し、[Create role (ロールの作成)] を選択します。

のセットアップAWS IAM アイデンティティセンター(IAM アイデンティティセンター)

AWS IAM アイデンティティセンター(IAM Identity Center) を使用すると、ユーザーはシンプルな URL で DataBrew にサインインできます。 にサインインしなくても、AWS マネジメントコンソールAWSアカウントは必要ありません。

IAM Identity Center をセットアップするには

1. AWS Organizationsコンソールを開き、まだ組織がない場合は組織を作成します。この組織では、すべての機能がデフォルトで有効になっています。

   詳細については、AWS IAM アイデンティティセンター「前提条件」と「組織の作成と管理」を参照してください。
2. AWS IAM アイデンティティセンターコンソールを開きます。
3. ID ソースを選択します。

   デフォルトでは、迅速かつ簡単なユーザー管理のために IAM アイデンティティセンターストアを取得します。必要に応じて、代わりに外部 ID プロバイダーを接続するか、AWS Managed Microsoft ADディレクトリをオンプレミスの Active Directory に接続できます。このガイドでは、デフォルトの IAM Identity Center ストアを使用します。

詳細については、AWS IAM アイデンティティセンター「 ユーザーガイド」の「ID ソースの選択」を参照してください。

4. DataBrew アクセスのアクセス許可セットを作成します。

   a. IAM Identity Center ナビゲーションペインで、AWSアカウントを選択し、アクセス許可セットを選択します。

   b. アクセス許可セットの作成ページで、カスタムアクセス許可セットの作成を選択します。

   c. リレー状態には、 と入力しますhttps://console.aws.amazon.com/databrew/home?region=us-east-1#landing。

      これを入力すると、ユーザーは DataBrew に直接移動できます。

   d. AWS管理ポリシーをアタッチ、DataBrew を検索、AwsGlueDataBrewFullAccessPolicy を選択します。これを選択すると、DataBrew に必要なすべてのアクセス許可がユーザーに付与されます。詳細については、「」を参照してくださいコンソールユーザーの IAM ポリシーの追加。

   e. (オプション) カスタムアクセス許可ポリシーを作成し、ユーザーのアクセス許可をカスタマイズします。

5. IAM アイデンティティセンターのナビゲーションペインで、[グループ] を選択し、[グループを作成] を選択します。グループ名を入力し、[作成] を選択します。

6. IAM Identity Center ストアにユーザーを追加します。

   a. IAM アイデンティティセンターのナビゲーションペインで、[ユーザー] を選択します。

   b. [ユーザーの追加] 画面で、必要な情報を入力し、[パスワード設定手順の案内メールをユーザーに送信] を選択します。次の設定手順に関するメールがユーザーに送られます。

   c. [次へ: グループ] を選択し、目的のグループを選択して、[ユーザーを追加] を選択します。

      SSO の使用を案内する招待メールがユーザーに送られます。この E メールでは、招待を受け入れるを選択し、パスワードを設定する必要があります。また、E メールでポータル URL を見つけることもできます。この URL を使用して DataBrew にアクセスできます。

7. 各ユーザーをアカウントに割り当てます。

   a. IAM Identity Center コンソールを開き、ナビゲーションペインでAWSアカウントを選択します。

   b. AWS組織を選択し、AWSアカウントを選択します。

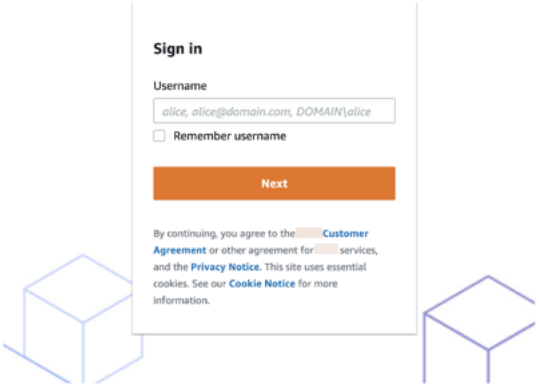
   c. ユーザーの割り当て画面で、グループタブを選択し、目的のグループを選択します。

d. [Next: Permissions sets] (次へ: アクセス許可セット) を選択します。

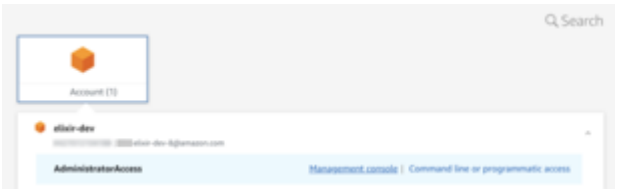
e. DataBrew のアクセス許可セットを選択し、完了を選択します。

IAM Identity Center 対応ユーザーのログイン手順

1. IAM Identity Center が有効なアカウントAWSを使用して にサインインします。



2. AWSアカウント ID をクリックします。



3. マネジメントコンソールをクリックして、DataBrew コンソールにワンクリックでリダイレクトします。

JupyterLab で DataBrew を拡張機能として使用する

> **Warning**
>
> AWS Glue DataBrew JupyterLab 3 のサポートが終了するため、JupyterLab 拡張機能のサポートは 2024 年 12 月 31 日に終了します。詳細については、JupyterLab 3 メンテナンスの終了」を参照してください。

Jupyter Notebook 環境でデータを準備する場合は、JupyterLab AWS Glue DataBrewで のすべての機能を使用できます。

JupyterLab は、Jupyter Notebook 用のウェブベースのインタラクティブな開発環境です。ローカル JupyterLab ウェブページでは、ターミナル、SQL セッション、Python などのセクションを追加できます。AWS Glue DataBrew拡張機能をインストールしたら、DataBrew コンソールのセクションを追加できます。JupyterLab 環境から直接、既存のノートブックやその他の拡張機能で実行されます。

トピック

前提条件

開始する前に、以下の項目を設定します。

- AWSアカウント – アカウントをまだお持ちでない場合は、 から始めます新しいAWSアカウントのセットアップ。
- DataBrew に必要なアクセス許可にアクセスできるAWS Identity and Access Management(IAM) ユーザー – 詳細については、「」を参照してくださいDataBrew アクセス許可を持つユーザーまたはグループの追加。
- DataBrew オペレーションで使用する IAM ロール – `AwsGlueDataBrewDataAccessRole`が設定されている場合、デフォルトを使用できます。追加の IAM ロールをセットアップするには、「」を参照してくださいデータリソースのアクセス許可を持つ IAM ロールの追加。
- JupyterLab のインストール (バージョン 2.2.6 以降) – 詳細については、JupyterLab ドキュメントの以下のトピックを参照してください。
  - JupyterLab の前提条件
  - JupyterLab のインストール – を使用することをお勧めします`pip install jupyterlab`。
- Node.js のインストール (バージョン 12.0 以降) 。
- AWS Command Line Interface(AWS CLI) のインストール – 詳細については、「」を参照してくださいのセットアップAWS CLI。
- AWS Jupyter プロキシのインストール (`pip install aws-jupyter-proxy`) – この拡張機能は、認証情報を安全に渡すAWSためにAWSサービスエンドポイントとともに使用されます。詳細については、GitHub のaws-jupyter-proxy」を参照してください。

前提条件がインストールされていることを確認するには、次の例に示すように、コマンドラインで次のようなテストを実行できます。

```
echo "
AWS CLI:"
which aws
aws --version
aws configure list
aws sts get-caller-identity

echo "
Python (current environment):"
which python
python --version

echo "
Node.JS:"
which node
node --version

echo "
Jupyter:"
where jupyter
jupyter --version
jupyter serverextension list
pip3 freeze | grep jupyter
```

出力は次のようになります。ディレクトリはオペレーティングシステムと設定によって異なります。

```
AWS CLI:
/usr/local/bin/aws
aws-cli/2.1.2 Python/3.7.4 Darwin/19.6.0 exe/x86_64
      Name                    Value             Type    Location
      ----                    -----             ----    --------
   profile                <not set>             None    None
access_key     ****************VXW4 shared-credentials-file
secret_key     ****************MRJN shared-credentials-file
    region                us-east-1      config-file    ~/.aws/config
{
    "UserId": "",
    "Account": "111122223333",
    "Arn": "arn:aws:iam::111122223333:user/user2"
}
```

```
Python (current environment):
/usr/local/opt/python /libexec/bin/python
Python 3.8.5

Node.JS:
/usr/local/bin/node
v15.0.1

Jupyter:
/usr/local/bin/jupyter
jupyter core     : 4.6.3
jupyter-notebook : 6.0.3
qtconsole        : 4.7.5
ipython          : 7.16.1
ipykernel        : 5.3.2
jupyter client   : 6.1.6
jupyter lab      : 2.2.9
nbconvert        : 5.6.1
ipywidgets       : 7.5.1
nbformat         : 5.0.7
traitlets        : 4.3.3

config dir: /usr/local/etc/jupyter
    aws_jupyter_proxy  enabled
    - Validating...
      aws_jupyter_proxy  OK
    jupyterlab  enabled
    - Validating...
      jupyterlab 2.2.9 OK

aws-jupyter-proxy==0.1.0
jupyter-client==6.1.7
jupyter-core==4.7.0
jupyterlab==2.2.9
jupyterlab-pygments==0.1.2
jupyterlab-server==1.2.0
```

拡張機能を使用するように JupyterLab を設定する

JupyterLab をインストールしたら、データアクセスを保護し、サーバー拡張機能を有効にするように設定する必要があります。

パスワードと暗号化を設定するには

1. 拡張機能に追加する予定のデータを保護するためのパスワードを設定します。Jupyter にはパスワードユーティリティが用意されています。次のコマンドを実行して、プロンプトに任意のパスワードを入力します。

   ```
   jupyter notebook password
   ```

   出力は次のようになります。

   ```
   Enter password:
   Verify password:
   [NotebookPasswordApp] Wrote hashed password to /home/ubuntu/.jupyter/
   jupyter_notebook_config.json
   ```

2. Jupyter サーバーで暗号化を有効にします。Jupyter をローカルマシンにインストールし、誰もネットワーク経由でアクセスできない場合は、このステップをスキップできます。

   Transport Layer Security (TLS) による暗号化を設定するには、環境に合わせてカスタマイズされた証明書を作成します。詳細については、Jupyter ドキュメントの「サーバーの保護での Let's Encrypt の使用」を参照してください。 https://jupyter-notebook.readthedocs.io/en/stable/public_server.html#securing-a-notebook-server

3. JupyterLab を起動するには、コマンドプロンプトで次のコマンドを実行します。

   ```
   jupyter lab
   ```

   詳細については、JupyterLab ドキュメントの「JupyterLab の開始」を参照してください。 JupyterLab

4. JupyterLab の実行中に、次のような URL でアクセスできます。 http://localhost:8888/lab暗号化を設定する場合は、 `https` の代わりに を使用しますhttp。ポートをカスタマイズした場合は、 の代わりにポート番号を置き換えます8888。

以下の手順を使用して、サードパーティー拡張機能を有効にします。

JupyterLab でサードパーティー拡張機能を有効にするには

1. JupyterLab ウェブページで、左側のメニューで Extension Manager アイコンを選択します。

2. サードパーティー拡張機能を実行するリスクに関する警告をお読みください。信頼できるデベロッパーからのみ拡張機能をインストールします。
3. JupyterLab でサードパーティー拡張機能を有効にするには、有効化を選択します。
4. プロンプトに従って JupyterLab を再構築して再ロードします。

JupyterLab の DataBrew 拡張機能の有効化

拡張機能を有効にして JupyterLab を安全にインストールしたら、DataBrew 拡張機能をインストールして、ノートブックで DataBrew を実行できるようにします。

DataBrew の拡張機能をインストールするには (コンソール)

1. JupyterLab を起動するには、コマンドプロンプトで次のコマンドを実行します。

```
jupyter lab
```

2. JupyterLab ウェブページで、左側のメニューで Extension Manager アイコンを選択します。
3. 左上の検索に**brew**「」と入力して、DataBrew 拡張機能を検索します。
4. リストから aws_glue_databrew_jupyter を見つけますが、クリックしないでください。強調表示された拡張機能の名前をクリックすると、新しいブラウザウィンドウが開き、GitHub の aws_glue_databrew_jupyter ページが表示されます。
5. DataBrew 拡張機能をインストールするには、次のいずれかを選択します。

   - コマンドラインで、 を実行しますjupyter labextension install aws_glue_databrew_jupyter。
   - 拡張カードの下部にある「aws_glue_databrew_jupyter」の下の灰色の文字でインストールを選択します。

   DataBrew 拡張機能は JupyterLab バージョン 1.2 および 2.x と互換性があります。
6. インストールされていることを確認するには、 を実行しますjupyter labextension list。出力は次のようになります。

```
JupyterLab v2.2.9
Known labextensions:
   app dir: /usr/local/share/jupyter/lab  # varies by OS
        aws_glue_databrew_jupyter v1.0.1  enabled  OK
```

7. 次のいずれかを使用して JupyterLab を再構築します。

   - コマンドプロンプトで、 を実行しますjupyter lab build。
   - ウェブページで、左上の再構築を選択します。

8. ビルドが完了したら、次のいずれかを実行します。

   - コマンドプロンプトで、 を実行しますjupyter lab。
   - ウェブページで、ビルド完了メッセージで再ロードを選択します。

9. JupyterLab ウェブページで、左側のメニューのアイコンを選択して Extension Manager を閉じます。

   拡張機能を開くには、ランAWS Glue DataBrewチャータブのその他セクションから起動を選択します。拡張機能は、アクセスキーとAWSリージョンAWS CLI設定に現在の設定を使用します。

セットアップが完了したら、 AWS Glue DataBrewタブを使用して JupyterLab 内から DataBrew とやり取りできます。

の開始方法AWS Glue DataBrew

次のチュートリアルを使用して、最初の DataBrew プロジェクトの作成をガイドできます。サンプルデータセットをロードし、そのデータセットで変換を実行し、それらの変換をキャプチャするレシピを構築し、変換されたデータを Amazon S3 に書き込むジョブを実行します。

トピック

前提条件

続行する前に、「」の該当する手順に従ってくださいセットアップAWS Glue DataBrew。次に、 に進みますステップ 1: プロジェクトを作成する。

ステップ 1: プロジェクトを作成する

このステップでは、DataBrew コンソールを使用してサンプルプロジェクトをすばやく開始します。

プロジェクトを作成するには

1. にサインインAWS マネジメントコンソールし、https://console.aws.amazon.com/databrew/ で DataBrew コンソールを開きます。
2. DataBrew コンソールの右上でAWSリージョンが選択されていることを確認します。DataBrew でサポートされているAWSリージョンのリストについては、のDataBrew エンドポイントとクォータ」を参照してくださいAWS 全般のリファレンス。
3. ナビゲーションペインで、プロジェクトを選択し、プロジェクトの作成を選択します。

4. プロジェクトの詳細ペインで、次の操作を行います。

   - プロジェクト名に と入力しますchess-project。
   - アタッチされたレシピで、新しいレシピを作成します。レシピの推奨名が指定されています (chess-project-recipe)。
5. データセットの選択ペインで、サンプルファイルを選択します。
6. サンプルファイルペインで、有名なチェスゲームの移動を選択します。このデータセットには、20,000 を超えるチェスゲームに関する詳細情報が含まれています。

   データセット名には、データセットの推奨名が指定されます (chess-games)。
7. アクセス許可ペインで、 を選択しますAwsGlueDataBrewDataAccessRole。これは、DataBrew がユーザーに代わって Amazon S3 バケットにアクセスできるようにするサービスにリンクされたロールです。
8. プロジェクトの作成を選択し、DataBrew がプロジェクトの準備を完了するまで待ちます。ウィンドウは次のようになります。

   表示されるデータは、chess-gamesデータセットのサンプルを表します。デフォルトでは、サンプルはデータセットの最初の 500 行で構成されます。このプロジェクト設定は後で変更できます。

   ツールバーは、データに適用できる数百のデータ変換へのアクセスを提供します。

   DataBrew コンソールの右側のレシピペインは、これまでに適用した変換を追跡します。

ステップ 2: データを要約する

このステップでは、DataBrew レシピを構築します。これは、このデータセットなどに適用できる変換のセットです。レシピが完了したら、使用できるように公開します。

チェスのゲームでは、他のプレイヤーに対するパフォーマンスに基づいてプレイヤーを評価できます。詳細については、「https://en.wikipedia.org/wiki/Chess_rating_system」を参照してください。このチュートリアルでは、両方のプレイヤーがクラス A であったゲームのみに焦点を当てます。つまり、評価は 1800 以上です。

データを要約するには

1. 変換ツールバーで、フィルター、条件別、以上を選択します。
2. これらのオプションを次のように設定します。

- ソース列 - white_rating
- フィルター条件 – 1800 以上

変換の仕組みを確認するには、変更のプレビューを選択します。次に、[適用]をクリックします。

3. 前のステップを繰り返しますが、今回はソース列を に設定しますblack_rating。変更を適用すると、サンプルデータには、各側のプレイヤー (白黒) がクラス A 以上であったゲームのみが含まれます。
4. データを要約して、各側で獲得されたゲームの数を決定します。これを行うには、変換ツールバーでグループを選択します。
5. グループプロパティの場合は、次の操作を行います。

   a. 最初の行で、列名winnerに を選択します。Aggregate を Group by に設定したままにします。

   b. 2 行目で、列名victory_statusに を選択します。Aggregate を Group by に設定します。

   c. 別の列を追加 を選択します。

   d. 3 行目で、列名winnerに を選択します。Aggregate を Count に設定します。

   e. グループタイプで、新しいテーブルとしてグループを選択します。プレビューペインには、結果がどのように表示されるかが表示されます。

   f. [Finish] を選択してください。
6. Publish を選択して、レシピペインの右側に作業を保存します。
7. バージョンの説明 に、レシピの最初のバージョンを入力します。次に、発行を選択します。

ステップ 3: 変換を追加する

このステップでは、レシピにさらに変換を追加し、別のバージョンのレシピを公開します。この例を絞り込むために、すべてのチェスゲームが明確な勝者になるわけではないという情報を使用します。一部のゲームはドローで再生されます。

レシピ変換を追加して再発行するには

1. 変換ツールバーから、フィルター、条件別、描画に再生されたゲームを削除しないを選択します。

2. これらのオプションを次のように設定します。

   - ソース列 - victory_status
   - フィルター条件 – そうではありません draw

   この変換をレシピに追加するには、適用を選択します。

3. よりわかりやすいvictory_statusように、 のデータを変更します。これを行うには、変換ツールバーから、値またはパターンをクリーンアップ、置換、置換を選択します。
4. これらのオプションを次のように設定します。

   - ソース列 - victory_status
   - 置き換える値を指定する – 値またはパターン
   - 置き換える値 - mate
   - を値に置き換えます - checkmate

   この変換をレシピに追加するには、適用を選択します。

5. 前のステップを繰り返しますが、 resign を に変更しますother player resigned。
6. 前のステップを繰り返しますが、 outoftime を に変更しますtime ran out。
7. Publish を選択して、レシピペインの右側に作業を保存します。

ステップ 4: DataBrew リソースを確認する

サンプルプロジェクトを使用したので、これまでに作成した DataBrew リソースを確認します。

DataBrew リソースを確認するには

1. ナビゲーションペインで、データセットを選択します。

   サンプルプロジェクトを作成すると、DataBrew がデータセット () を作成しましたchess-games。ソースデータファイルは Amazon S3 に保存され、Microsoft Excel 形式 () ですchess-games.xlsx。ファイルには、20,000 を超えるチェスゲームのメタデータが含まれています。chess-games データセットは、DataBrew がそのファイル内のデータを読み取るために必要な情報を提供します。

2. ナビゲーションペインで、プロジェクトを選択します。

前のステップ () で操作したプロジェクトが表示されますchess-project。すべてのプロジェクトにはデータセットが必要です。この場合は ですchess-games。すべてのプロジェクトにはレシピも必要です。これにより、データ変換手順を追加することができます。このサンプルプロジェクトを作成すると、DataBrew によって新しい (空の) レシピが作成され、プロジェクトにアタッチされます。

3. ナビゲーションペインでレシピを選択し、レシピ名列で chess-project-recipe を選択します。これは、DataBrew がプロジェクト用に作成したレシピと、変換ステップを追加して改良したレシピを示しています。
4. 左側で、公開されたレシピバージョンを表示します。これらのいずれかを選択して、レシピステップタブを表示します。タブには、そのバージョンのレシピの詳細とステップが表示されます。
5. データリネージタブを表示します。このタブには、データの出所と使用方法が表示されます。詳細については、図のアイコンのいずれかを選択してください。

ステップ 5: データプロファイルを作成する

プロジェクトで を使用する場合、DataBrew はサンプル内の行数や各列の一意の値の分布などの統計を表示します。これらの統計などは、サンプルのプロファイルを表します。

データプロファイルをリクエストするには、プロファイルジョブを作成して実行します。

データセットをプロファイリングするには

1. ナビゲーションペインで、ジョブを選択します。
2. プロファイルジョブタブで、ジョブの作成を選択します。
3. ジョブ名には、 と入力しますchess-data-profile。
4. ジョブタイプで、プロファイルジョブの作成を選択します。
5. ジョブ入力ペインで、次の操作を行います。
   - Run on で、データセットを選択します。
   - データセットの選択を選択して使用可能なデータセットのリストを表示し、 を選択しますchess-games。
6. ジョブ出力設定ペインで、次の操作を行います。
   - ファイルタイプで、JSON (JavaScript Object Notation) を選択します。

- S3 の場所を選択して使用可能な Amazon S3 バケットのリストを表示し、使用するバケットを選択します。次に [Browse] (参照) を選択します。フォルダのリストで、 を選択しdatabrew-output、選択 を選択します。

7. アクセス許可ペインで、 を選択しますAwsGlueDataBrewDataAccessRole。これは、DataBrew がユーザーに代わって Amazon S3 バケットにアクセスできるようにするサービスにリンクされたロールです。
8. ジョブの作成と実行を選択します。DataBrew は、設定を使用してジョブを作成し、それを実行します。
9. ジョブ実行履歴ペインで、ジョブのステータスが から Runningに変わるのを待ちますSucceeded。
10. プロファイルを表示するには、VIEW PROFILE を選択します。

DATASETS ウィンドウが表示されます。少し時間を取って、次のタブを確認してください。

- データセットのプレビュー
- プロファイルの概要
- 列統計
- データ系統統計

ステップ 6: データセットを変換する

これまでは、データセットのサンプルのみでレシピをテストしていました。次に、DataBrew レシピジョブを作成してデータセット全体を変換します。

ジョブが実行されると、DataBrew はデータセット内のすべてのデータにレシピを適用し、変換されたデータを Amazon S3 バケットに書き込みます。変換されたデータは元のデータセットとは別のものです。DataBrew はソースデータを変更しません。

続行する前に、書き込むことができる Amazon S3 バケットがアカウントにあることを確認してください。そのバケットで、DataBrew からのジョブ出力をキャプチャするフォルダを作成します。これらの手順を実行するには、次の手順を使用します。

ジョブ出力をキャプチャする S3 バケットとフォルダを作成するには

1. にサインインAWS マネジメントコンソールし、https://console.aws.amazon.com/databrew/ で Amazon S3 コンソールを開きます。

   使用可能な Amazon S3 バケットがあり、そのバケットに対する書き込みアクセス許可がある場合は、次のステップをスキップします。
2. Amazon S3 バケットがない場合は、バケットの作成を選択します。バケット名に、新しいバケットの一意の名前を入力します。[バケットを作成] を選択します。
3. バケットのリストから、使用するバケットを選択します。
4. Create folder (フォルダの作成) を選択します。
5. フォルダ名に と入力しdatabrew-output、フォルダの作成を選択します。

ジョブを含む Amazon S3 バケットとフォルダを作成したら、次の手順を使用してジョブを実行します。

レシピジョブを作成して実行するには

1. ナビゲーションペインで、ジョブを選択します。
2. レシピジョブタブで、ジョブの作成を選択します。
3. ジョブ名には、 と入力しますchess-winner-summary。
4. ジョブタイプで、レシピジョブの作成を選択します。
5. ジョブ入力ペインで、次の操作を行います。
   - Run on で、データセットを選択します。
   - データセットの選択を選択して使用可能なデータセットのリストを表示し、 を選択しますchess-games。
   - Select a recipe を選択して使用可能なレシピのリストを表示し、 を選択しますchess-project-recipe。
6. ジョブ出力設定ペインで、次の操作を行います。
   - ファイルタイプ – CSV (カンマ区切り値) を選択します。
   - S3 の場所 - 使用可能な Amazon S3 バケットのリストを表示するにはこのフィールドを選択し、使用するバケットを選択します。次に [Browse] (参照) を選択します。フォルダのリストで、 を選択しdatabrew-output、選択 を選択します。

7. アクセス許可ペインで、 を選択しますAwsGlueDataBrewDataAccessRole。このサービスにリンクされたロールにより、DataBrew はユーザーに代わって Amazon S3 バケットにアクセスできます。
8. ジョブの作成と実行を選択します。DataBrew は、設定を使用してジョブを作成し、それを実行します。
9. ジョブ実行履歴ペインで、ジョブのステータスが から Runningに変わるのを待ちますSucceeded。
10. 出力 を選択して Amazon S3 コンソールにアクセスします。S3 バケットを選択し、ジョブ出力にアクセスするフォルダを選択します databrew-output。
11. (オプション) ダウンロードを選択してファイルをダウンロードし、その内容を表示します。

ステップ 7: (オプション) クリーンアップする

チュートリアルはこれで完了です。作成した DataBrew および Amazon S3 リソースを引き続き使用するか、削除することができます。

リソースをクリーンアップするには

1. https://console.aws.amazon.com/databrew/ で DataBrew コンソールを開き、ナビゲーションペインでプロジェクトを選択します。
2. プロジェクト (サンプルプロジェクト) を選択します。[アクション] で、[削除] を選択します。
3. Delete Sample project ペインで、Delete attached recipe を選択します。その後、[削除] をクリックします。プロジェクトとそのレシピとジョブは削除されます。
4. ナビゲーションペインで、データセットを選択します。
5. データセット (chess-games) を選択し、アクションで削除を選択します。
6. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。databrew-output フォルダとそのコンテンツを削除します。

   (オプション) Amazon S3 バケットが不要になった場合は、削除できます。

を使用したデータへの接続AWS Glue DataBrew

ではAWS Glue DataBrew、データセットはファイルからアップロードされるか、他の場所に保存されるデータを表します。例えば、データは Amazon S3、サポートされている JDBC データソース、またはAWS Glueデータカタログに保存できます。DataBrew に直接ファイルをアップロードしない場合、データセットには DataBrew がデータに接続する方法の詳細も含まれています。

データセット (などinventory-dataset) を作成するときは、接続の詳細を 1 回だけ入力します。その時点から、DataBrew は基盤となるデータにアクセスできます。このアプローチを使用すると、接続の詳細やファイル形式を気にすることなく、プロジェクトを作成し、データの変換を開発できます。

トピック

データソースでサポートされているファイルタイプ

次のファイル要件は、Amazon S3 に保存されているファイルと、ローカルドライブからアップロードするファイルに適用されます。DataBrew は、カンマ区切り値 (CSV)、Microsoft Excel、JSON、ORC、Parquet のファイル形式をサポートしています。ファイルがサポートされているタイプの 1 つである場合は、非標準の拡張子を持つファイルを使用することも、拡張子を持たないファイルを使用することもできます。

DataBrew がファイルタイプを推測できない場合は、必ず正しいファイルタイプ (CSV、Excel、JSON、ORC、または Parquet) を選択してください。圧縮された CSV、JSON、ORC、および Parquet ファイルがサポートされていますが、CSV および JSON ファイルにはファイル拡張子として圧縮コーデックを含める必要があります。フォルダをインポートする場合、フォルダ内のすべてのファイルは同じファイルタイプである必要があります。

ファイル形式とサポートされている圧縮アルゴリズムを次の表に示します。

> **Note**
>
> CSV、Excel、JSON ファイルは Unicode (UTF-8) でエンコードする必要があります。

Format	ファイル拡張子 (オプション)	圧縮ファイルの拡張子 (必須)

カンマ区切り値	`.csv`	`.gz`
		`.snappy`
		`.lz4`
		`.bz2`
		`.deflate`

Microsoft Excel ワークブック	`.xlsx`	圧縮サポートなし

JSON (JSON ドキュメントと JSON 行)	`.json, .jsonl`	`.gz`
		`.snappy`
		`.lz4`
		`.bz2`
		`.deflate`

Apache ORC	`.orc`	`.zlib` `.snappy`

Apache Parquet	`.parquet`	`.gz`
		`.snappy`
		`.lz4`

データソースと出力でサポートされている接続

DataBrew レシピジョブでは、次のデータソースに接続できます。これには、DataBrew に直接アップロードするファイルではないデータソースが含まれます。使用しているデータソースは、データベース、データウェアハウス、またはその他のものと呼ばれる場合があります。すべてのデータプロバイダーをデータソースまたは接続と呼びます。

データソースとして次のいずれかを使用してデータセットを作成できます。

DataBrew レシピジョブの出力にはAWS Glue Data Catalog、Amazon RDS でサポートされている Amazon S3、または JDBC データベースを使用することもできます。Amazon AppFlow とAWS Data Exchangeは、DataBrew レシピジョブの出力でデータストアをサポートしていません。

- Amazon S3

  S3 を使用して、任意の量のデータを保存および保護できます。データセットを作成するには、DataBrew がデータファイルにアクセスできる S3 URL を指定します。次に例を示します。`s3://your-bucket-name/inventory-data.csv`

  DataBrew は S3 フォルダ内のすべてのファイルを読み取ることもできます。つまり、複数のファイルにまたがるデータセットを作成できます。これを行うには、次の形式で S3 URL を指定します: `s3://your-bucket-name/your-folder-name/`。

  DataBrew は次の Amazon S3 ストレージクラスのみをサポートします。Standard、Reduced Redundancy、Standard-IA、S3 One Zone-IA。DataBrew は、他のストレージクラスのファイルを無視します。DataBrew は空のファイル (0 バイトを含むファイル) も無視します。Amazon S3 ストレージクラスの詳細については、Amazon S3 コンソールユーザーガイド」の「Amazon S3 ストレージクラスの使用」を参照してください。 Amazon S3

- AWS Glue Data Catalog

  データカタログを使用して、AWSクラウドに保存されているデータへの参照を定義できます。データカタログを使用すると、次のサービスで個々のテーブルへの接続を構築できます。

  - データカタログ Amazon S3
  - データカタログ Amazon Redshift
  - データカタログ Amazon RDS
  - AWS Glue

DataBrew は Amazon S3 フォルダ内のすべてのファイルを読み取ることもできます。つまり、複数のファイルにまたがるデータセットを作成できます。これを行うには、次の形式で Amazon S3 URL を指定します。 `s3://your-bucket-name/your-folder-name/`

DataBrew で使用するには、 で定義されている Amazon S3 テーブルにAWS Glue Data Catalog、データの形式を `classification`、、`csvjson`または として識別する というテーブルプロパティを追加する必要がありますparquettypeOfDatafile。テーブルの作成時にテーブルプロパティが追加されなかった場合は、AWS Glueコンソールを使用して追加できます。

DataBrew は、Amazon S3 ストレージクラス Standard、Reduced Redundancy、Standard-IA、および S3 One Zone-IA のみをサポートしています。DataBrew は、他のストレージクラスのファイルを無視します。DataBrew は空のファイル (0 バイトを含むファイル) も無視します。Amazon S3 ストレージクラスの詳細については、Amazon S3 コンソールユーザーガイド」の「Amazon S3 ストレージクラスの使用」を参照してください。 Amazon S3

適切なリソースポリシーが作成されている場合、DataBrew は他のアカウントのAWS Glue Data Catalog S3 テーブルにアクセスすることもできます。コンソールのAWS GlueData Catalog の設定タブでポリシーを作成できます。以下は、単一の 専用のポリシーの例ですAWS リージョン。

> **Warning**
>
> これは、 のデータカタログへの*$ACCOUNT_TO*無制限のアクセスを許可する、非常に寛容なリソースポリシーです*$ACCOUNT_FROM*。ほとんどの場合、リソースポリシーを特定のカタログまたはテーブルにロックダウンすることをお勧めします。詳細については、「 AWS Glueデベロッパーガイド」のAWS Glue「アクセスコントロールのリソースポリシー」を参照してください。

場合によっては、AWS Glue DataBrewにある S3の場所*$ACCOUNT_FROM*を指すAWS Glue Data Catalog S3 テーブル*$ACCOUNT_TO*を使用して、 でプロジェクトを作成したり、ジョブを実行したりできます*$ACCOUNT_FROM*。 S3 このような場合、 でプロジェクトとジョブを作成するときに使用する IAM ロールには、 からその S3 ロケーションのオブジェクトを一覧表示して取得するアクセス許可*$ACCOUNT_TO*が必要です*$ACCOUNT_FROM*。詳細については、「 AWS Glueデベロッパーガイド」の「クロスアカウントアクセスの付与」を参照してください。

- JDBC ドライバーを使用して接続されたデータ

データセットを作成するには、サポートされている JDBC ドライバーを使用してデータに接続します。詳細については、「でのドライバーの使用AWS Glue DataBrew」を参照してください。

DataBrew は、Java Database Connectivity (JDBC) を使用して以下のデータソースを正式にサポートしています。

- Microsoft SQL Server
- MySQL
- Oracle
- [PostgreSQL]
- Amazon Redshift
- Snowflake Connector for Spark

データソースは、DataBrew から接続できる任意の場所に配置できます。このリストには、テスト済みの JDBC 接続のみが含まれているため、サポートできます。

Amazon Redshift と Snowflake Connector for Spark データソースは、次のいずれかの方法で接続できます。

- テーブル名。
- 複数のテーブルとオペレーションにまたがる SQL クエリ。

SQL クエリは、プロジェクトまたはジョブの実行を開始するときに実行されます。

リストにない JDBC ドライバーを必要とするデータに接続するには、ドライバーが JDK 8 と互換性があることを確認してください。ドライバーを使用するには、DataBrew の IAM ロールを使用してアクセスできるバケットの S3 にドライバーを保存します。次に、データセットをドライバーファイルにポイントします。詳細については、「でのドライバーの使用AWS Glue DataBrew」を参照してください。

SQL ベースのデータセットのクエリ例:

```
SELECT
    *
FROM
    public.customer as c
JOIN
    public.customer_address as ca on c.current_address=ca.current_address
WHERE
    ca.address_id>0 AND ca.address_id<10001 ORDER BY ca.address_id
```

カスタム SQL の制限

JDBC 接続を使用して DataBrew データセットのデータにアクセスする場合は、次の点に注意してください。

- AWS Glue DataBrewは、データセットの作成の一部として指定したカスタム SQL を検証しません。SQL クエリは、プロジェクトまたはジョブの実行を開始するときに実行されます。DataBrew は、指定したクエリを受け取り、デフォルトまたは指定された JDBC ドライバーを使用してデータベースエンジンに渡します。
- 無効なクエリで作成されたデータセットは、プロジェクトまたはジョブで使用されると失敗します。データセットを作成する前にクエリを検証します。
- SQL の検証機能は、Amazon Redshift ベースのデータソースでのみ使用できます。
- プロジェクトでデータセットを使用する場合は、プロジェクトのロード中のタイムアウトを避けるため、SQL クエリランタイムを 3 分未満に制限します。プロジェクトを作成する前に、クエリランタイムを確認します。

- Amazon AppFlow

  Amazon AppFlow を使用すると、Salesforce、Zendesk、Slack、ServiceNow などのサードパーティーの Software-as-a-Service (SaaS) アプリケーションから Amazon S3 にデータを転送できます。その後、データを使用して DataBrew データセットを作成できます。

  Amazon AppFlow では、サードパーティーアプリケーションと送信先アプリケーションの間でデータを転送するための接続とフローを作成します。DataBrew で Amazon AppFlow を使用する場合は、Amazon AppFlow 送信先アプリケーションが Amazon S3 であることを確認します。Amazon S3 以外の Amazon AppFlow 送信先アプリケーションは DataBrew コンソールに表示されません。 Amazon S3 サードパーティーアプリケーションからのデータ転送と Amazon AppFlow 接続とフローの作成の詳細については、Amazon AppFlow ドキュメントを参照してください。

  DataBrew のデータセットタブで新しいデータセットに接続を選択し、Amazon AppFlow をクリックすると、Amazon S3 を送信先アプリケーションとして設定されている Amazon AppFlow 内のすべてのフローが表示されます。データセットにフローのデータを使用するには、そのフローを選択します。

  DataBrew コンソールでフローの作成、フローの管理、Amazon AppFlow の詳細の表示を選択すると、Amazon AppFlow コンソールが開き、これらのタスクを実行できます。

Amazon AppFlow からデータセットを作成したら、フローを実行し、データセットの詳細またはジョブの詳細を表示するときに最新のフロー実行の詳細を表示できます。DataBrew でフローを実行すると、データセットは S3 で更新され、DataBrew で使用できるようになります。

DataBrew コンソールで Amazon AppFlow フローを選択してデータセットを作成すると、次の状況が発生する可能性があります。

- データが集約されていない - フロートリガーがオンデマンドで実行されるか、フルデータ転送でスケジュールどおりに実行される場合は、DataBrew データセットの作成に使用する前に、フローのデータを集約してください。フローを集約すると、フロー内のすべてのレコードが 1 つのファイルにまとめられます。トリガータイプが Run on schedule with incremental data transfer または Run on event のフローは集約を必要としません。Amazon AppFlow でデータを集約するには、フロー設定の編集 > 送信先の詳細 > 追加設定 > データ転送設定を選択します。
- フローが実行されていない - フローの実行ステータスが空の場合、次のいずれかを意味します。
  - フローを実行するトリガーがオンデマンドで実行の場合、フローはまだ実行されていません。
  - フローを実行するトリガーが Run on イベントである場合、トリガーイベントはまだ発生していません。
  - フローを実行するトリガーがスケジュールどおりに実行される場合、スケジュールされた実行はまだ発生していません。

  フローを使用してデータセットを作成する前に、そのフローの実行フローを選択します。

  詳細については、「Amazon AppFlow ユーザーガイド」の「Amazon AppFlow フロー」を参照してください。 AppFlow
- AWS Data Exchange

  数百のサードパーティーデータソースから選択できますAWS Data Exchange。これらのデータソースにサブスクライブすることで、データのup-to-dateを取得できます。

  データセットを作成するには、サブスクライブしているAWS Data Exchangeデータ製品の名前と、使用する権限を指定します。

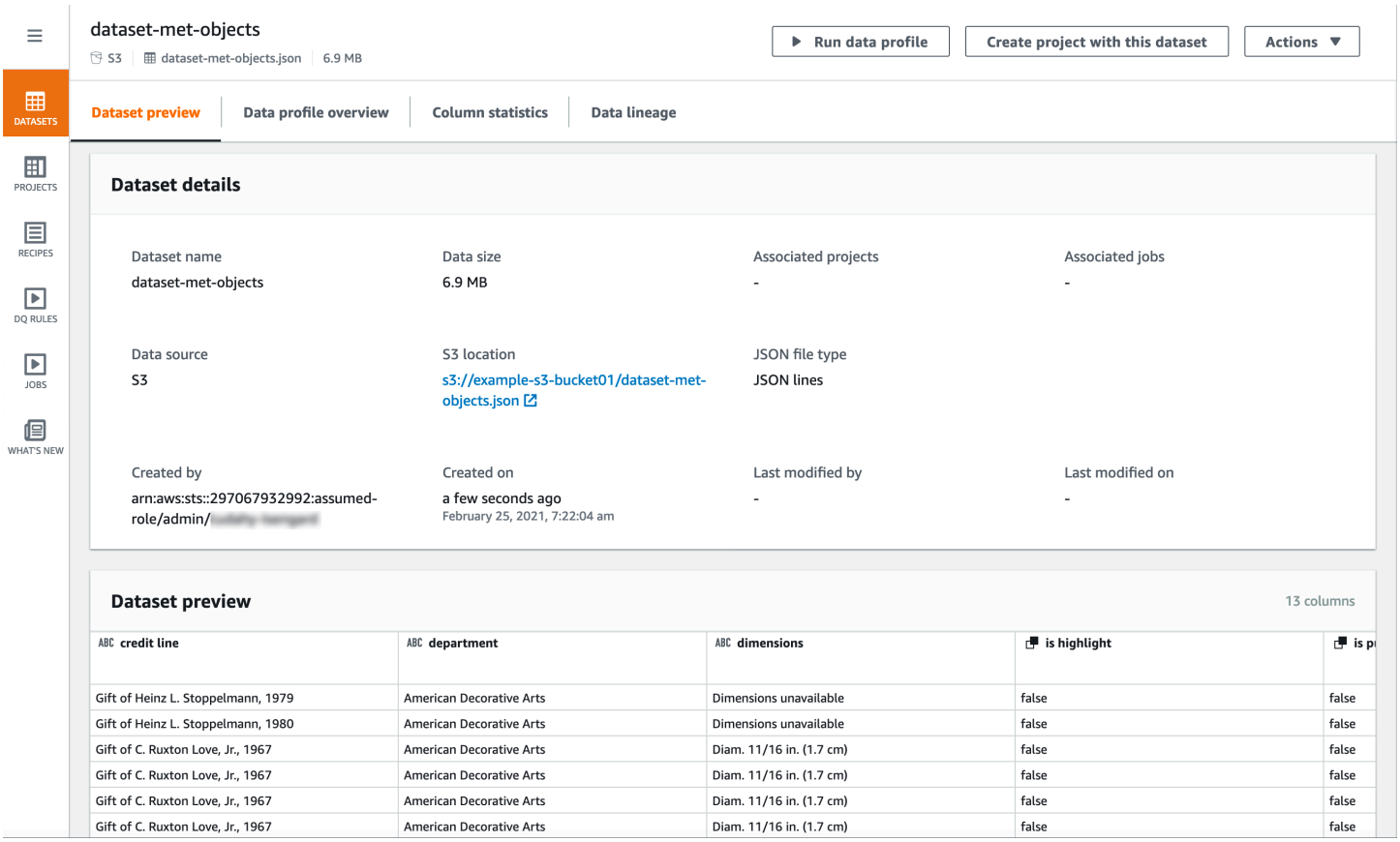
でのデータセットの使用AWS Glue DataBrew

DataBrew コンソールでデータセットのリストを表示するには、左側の DATASET を選択します。データセットページで、各データセットの詳細を表示するには、名前をクリックするか、コンテキストメニューからアクション、編集を選択します。

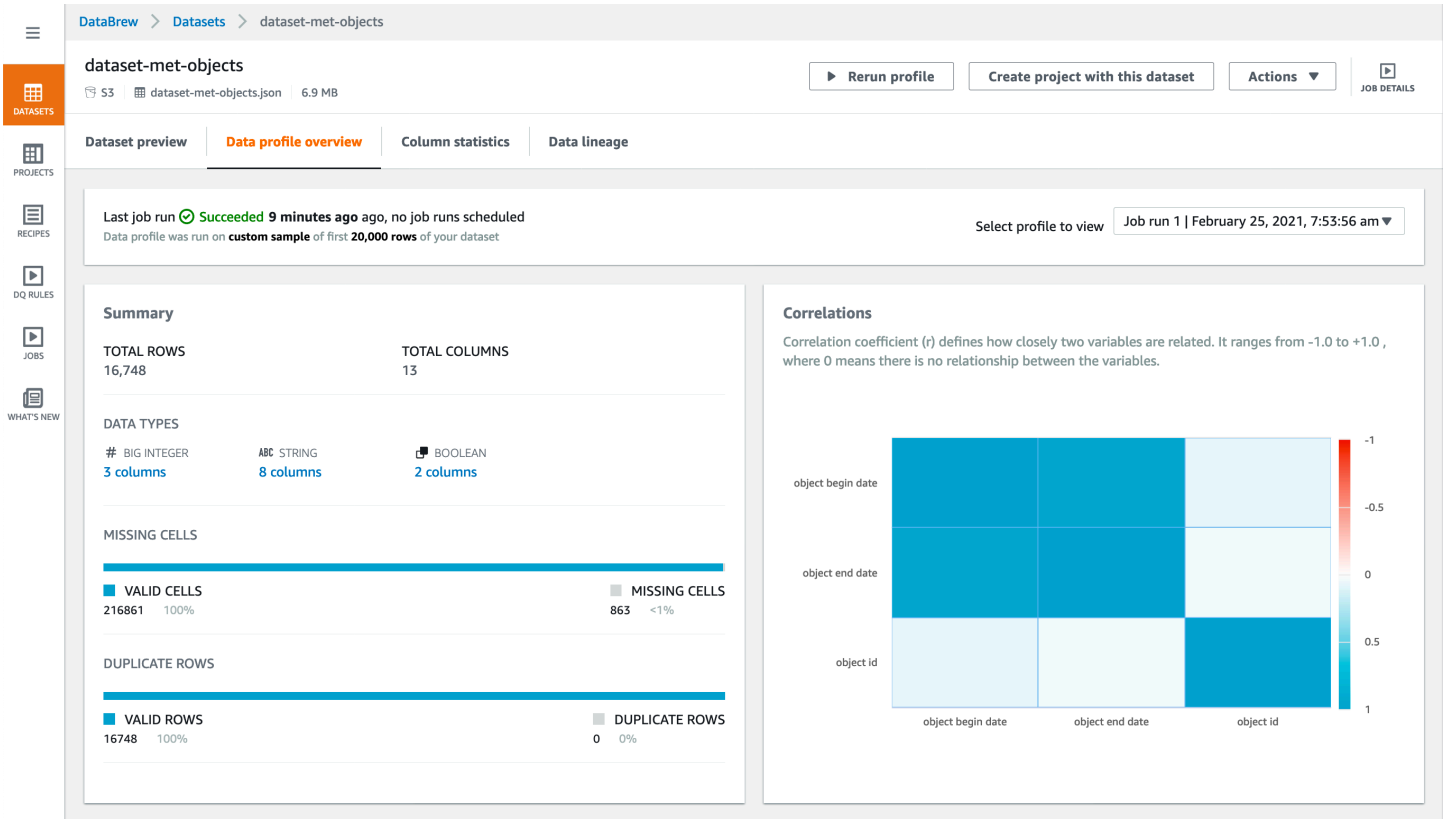
新しいデータセットを作成するには、DATASET, Connect new dataset を選択します。データソースごとに接続パラメータが異なるため、DataBrew が接続できるようにこれらを入力します。接続を保存してデータセットの作成を選択すると、DataBrew はデータに接続し、データのロードを開始します。詳細については、「データへの接続」を参照してください。

データセットページには、データの探索に役立つ以下の要素があります。

データセットのプレビュー – このタブには、次に示すように、データセットの接続情報とデータセットの全体的な構造の概要が表示されます。

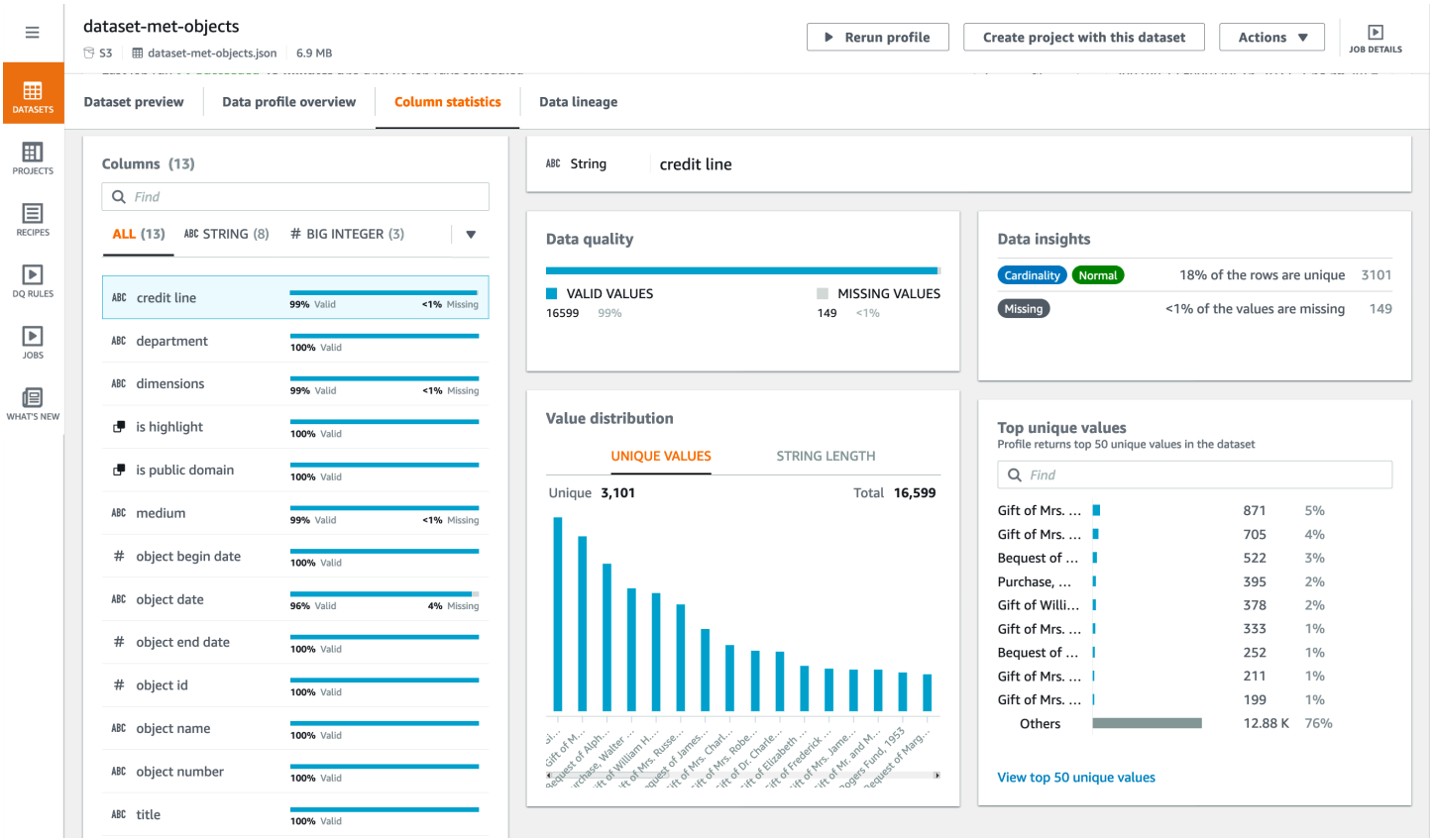


データプロファイルの概要 – このタブには、次に示すように、データセットの統計とボリューメトリックのグラフィカルデータプロファイルが表示されます。

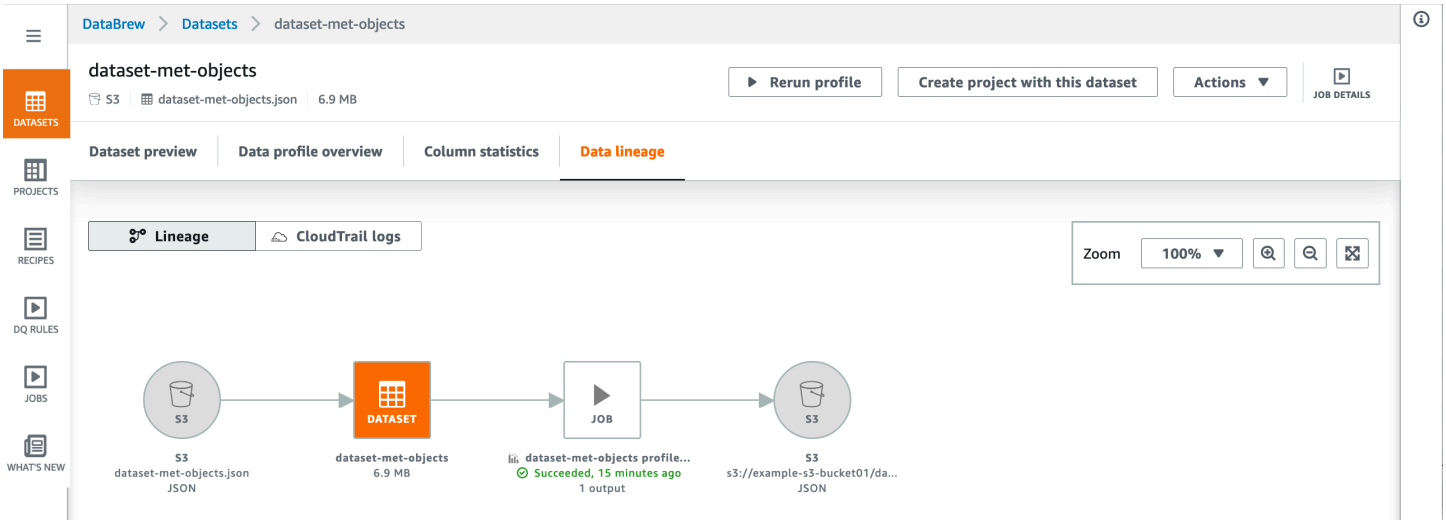


> **Note**
>
> データプロファイルを作成するには、データセットで DataBrew プロファイルジョブを実行します。これを行う方法については、「ステップ 5: データプロファイルを作成する」を参照してください。

列統計 – このタブでは、以下に示すように、データセット内の各列に関する詳細な統計を確認できます。



データリネージュ – このタブには、データセットの作成方法と DataBrew でのデータセットの使用方法が次のようにグラフィカルに表示されます。



トピック

- データセットの削除

データセットの削除

データセットが不要になった場合は、削除できます。データセットを削除しても、基盤となるデータソースには影響しません。DataBrew がデータソースへのアクセスに使用した情報は、単に削除されます。

他の DataBrew リソースがデータセットに依存している場合、データセットを削除することはできません。たとえば、データセットを使用する DataBrew プロジェクトが現在ある場合は、データセットを削除する前にまずプロジェクトを削除します。

データセットを削除するには、ナビゲーションペインからデータセットを選択します。削除するデータセットを選択し、アクションで削除を選択します。

データへの接続

次のデータソースへの接続の詳細については、該当するセクションを選択してください。

- AWS Glue Data Catalog – データカタログを使用して、AWSクラウドに保存されているデータオブジェクトへの参照を定義できます。これには、以下のサービスが含まれます。
  - Amazon Redshift
  - Aurora MySQL
  - Aurora PostgreSQL
  - Amazon RDS for MySQL
  - Amazon RDS for PostgreSQL

  DataBrew は、Data Catalog リソースに適用されたすべての Lake Formation アクセス許可を認識するため、DataBrew ユーザーは、承認された場合にのみこれらのリソースにアクセスできます。

  データセットを作成するには、Data Catalog データベース名とテーブル名を指定します。DataBrew は他の接続の詳細を処理します。
- AWS Data Exchange –AWS Data Exchange で利用可能な数百のサードパーティーデータソースから選択できます。これらのデータソースにサブスクライブすることで、常にup-to-dateのデータが得られます。

  データセットを作成するには、サブスクライブしている、または使用する権限を持つ Data Exchange データ製品の名前を指定します。
- JDBC ドライバー接続 – DataBrew を JDBC 互換データソースに接続することでデータセットを作成できます。DataBrew は、JDBC を介した次のソースへの接続をサポートしています。

- Amazon Redshift
- Microsoft SQL Server
- MySQL
- Oracle
- [PostgreSQL]
- Snowflake

トピック

でのドライバーの使用AWS Glue DataBrew

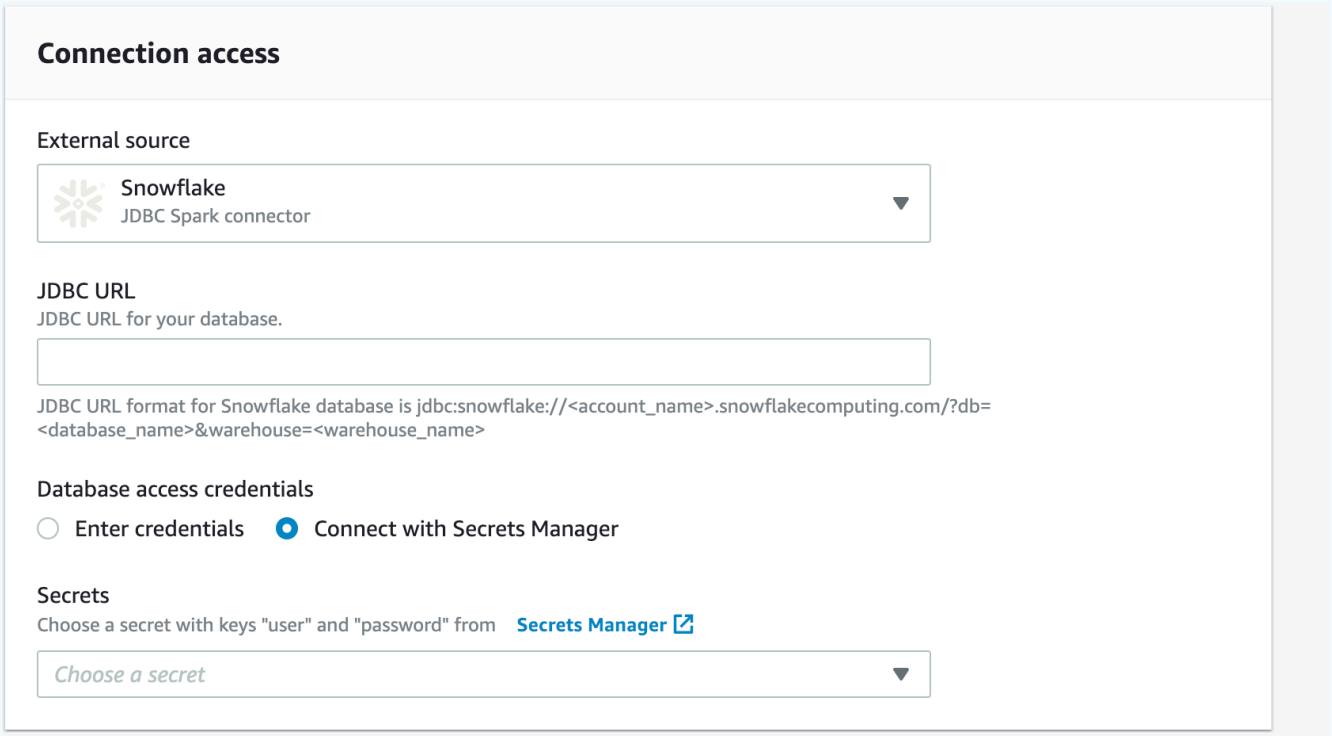
データベースドライバーは、Java Database Connectivity (JDBC) などのデータベース接続プロトコルを実装するファイルまたは URL です。ドライバーは、特定のデータベース管理システム (DBMS) と別のシステム間のアダプターまたはトランスレーターとして機能します。

この場合、AWS Glue DataBrewはデータに接続できます。その後、サポートされているデータソースからテーブルやビューなどのデータベースオブジェクトにアクセスできます。使用しているデータソースは、データベース、データウェアハウス、またはその他のものと呼ばれる場合があります。ただし、このドキュメントでは、すべてのデータプロバイダーをデータソースまたは接続と呼びます。

JDBC ドライバーまたは jar ファイルを使用するには、必要なファイルをダウンロードして S3 バケットに配置します。データへのアクセスに使用する IAM ロールには、両方のドライバーファイルに対する読み取りアクセス許可が必要です。

> **Note**
>
> With AWS Glue 4.0、データソースとしての Snowflake への接続はネイティブでサポートされています。カスタムjarファイルを指定する必要はありません。でAWS Glue DataBrew、外部ソース接続として Snowflake を選択し、Snowflake インスタンスの URL を指定します。URL では、`https://account_identifier.snowflakecomputing.com` という形式のホスト名を使用します。
> データアクセス認証情報、Snowflake データベース名、Snowflake スキーマ名を指定します。さらに、Snowflake ユーザーにデフォルトのウェアハウスセットがない場合は、ウェアハウス名を指定する必要があります。

Snowflake 接続は、AWS Secrets Managerシークレットを使用して認証情報を提供します。のプロジェクトロールとジョブロールには、このシークレットを読み取るアクセス許可が必要です。



DataBrew でドライバーを使用するには

1. 製品が提供する方法を使用して、使用しているデータソースのバージョンを確認します。
2. 必要なコネクタとドライバーの最新バージョンを見つけます。この情報は、データプロバイダーのウェブサイトで確認できます。
3. 必要なバージョンの JDBC ファイルをダウンロードします。これらは通常、Java ARchives (.JAR) ファイルとして保存されます。
4. コンソールから S3 バケットにドライバーをアップロードするか、.JAR ファイルへの S3 パスを指定します。
5. クラス、インスタンスなど、基本的な接続の詳細を入力します。
6. Virtual Private Cloud (VPC) 情報など、データソースに必要な追加の設定情報を入力します。

サポートされている JDBC ドライバー

製品	サポートされる のバージョン	ドライバーの手順とダウンロード	サポートされている SQL クエリ
Microsoft SQL Server	v6.x 以降	SQL Server 用 Microsoft JDBC ドライバー	サポートされていません
MySQL	v5.1 以降	MySQL コネクタ	サポートされていません
Oracle	v11.2 以降	Oracle JDBC ダウンロード	サポートされていません
[PostgreSQL]	v4.2.x 以降	PostgreSQL JDBC ドライバー	サポートされていません
Amazon Redshift	v4.1 以降	JDBC を使用した Amazon Redshift への接続	サポート
Snowflake	Snowflake のバージョンを確認するには、Snowflake ド	Snowflake に接続するには、次の両方が必要です。 • Snowflake JDBC ドライバー • Snowflake Connector for Spark	サポート

製品	サポートされる のバージョン	ドライバーの手順とダウンロード	サポートされている SQL クエリ
	キュメントの説明に従って CURRENT ERSION を使用します。		

DataBrew がネイティブにサポートしているものとは異なるバージョンのドライバーを必要とするデータベースまたはデータウェアハウスに接続するには、任意の JDBC ドライバーを指定できます。ドライバーは JDK 8 または Java 8 と互換性がある必要があります。データベースの最新のドライバーバージョンを検索する方法については、「」を参照してくださいでのドライバーの使用AWS Glue DataBrew。

DataBrew を使用したテキストファイル内のデータへの接続

DataBrew がサポートする入力ファイルには、次の形式オプションを設定できます。

- カンマ区切り値 (CSV) ファイル
  - 区切り文字

    デフォルトの区切り文字は、.csv ファイルのカンマです。ファイルが別の区切り文字を使用している場合は、データセットの作成時に追加設定セクションで CSV 区切り文字の区切り文字を選択します。.csv ファイルでは、次の区切り文字がサポートされています。
    - カンマ (,)
    - コロン (:)
    - セミコロン (;)
    - パイプ (|)

- タブ (\t)
- キャレット (^)
- バックスラッシュ (\)
- Space

- 列ヘッダー値

  CSV ファイルには、ファイルの最初の行としてヘッダー行を含めることができます。そうでない場合、DataBrew はヘッダー行を作成します。

  - CSV ファイルにヘッダー行が含まれている場合は、最初の行をヘッダーとして扱うを選択します。その場合、CSV ファイルの最初の行は列ヘッダー値を含むものとして扱われます。
  - CSV ファイルにヘッダー行が含まれていない場合は、デフォルトのヘッダーを追加を選択します。その場合、DataBrew はファイルのヘッダー行を作成し、データの最初の行をヘッダー値を含むものとして扱いません。DataBrew が作成するヘッダーは、、`Column_1`、`Column_2` `Column_3`などの形式で、ファイル内の各列のアンダースコアと数値で構成されます。

- JSON ファイル

  DataBrew は、JSON ファイルの JSON Lines と JSON ドキュメントの 2 つの形式をサポートしています。JSON Lines ファイルには、行ごとに 1 行が含まれます。JSON ドキュメントファイルでは、すべての行が単一の JSON 構造または配列に含まれます。JSON ファイルタイプは、JSON データセットを作成するときに追加設定セクションで指定できます。デフォルトの形式は JSON 行です。

- Excel ファイル

  DataBrew の Excel シートには、以下が適用されます。

  - Excel シートのロード

    デフォルトでは、DataBrew は Excel ファイルに最初のシートをロードします。ただし、Excel データセットを作成するときに、追加設定セクションで別のシート番号またはシート名を指定できます。

  - 列ヘッダー値

    Excel シートにはファイルの最初の行としてヘッダー行を含めることができますが、含めない場合、DataBrew によってヘッダー行が作成されます。

    - Excel シートにヘッダー行が含まれている場合は、最初の行をヘッダーとして扱うを選択します。その場合、Excel シートの最初の行は列ヘッダー値を含むものとして扱われます。

- Excel ファイルにヘッダー行が含まれていない場合は、デフォルトのヘッダーを追加を選択します。これにより、DataBrew がファイルのヘッダー行を作成し、データの最初の行をヘッダー値を含むものとして処理しないように指定します。DataBrew が作成するヘッダーは、、、Column_1、 Column_2 Column_3などの形式で、ファイル内の各列のアンダースコアと数値で構成されます。

Amazon S3 の複数のファイルでデータを接続する

DataBrew コンソールを使用すると、Amazon S3 バケットとフォルダに移動し、データセットのファイルを選択できます。ただし、データセットを 1 つのファイルに制限する必要はありません。

という名前のフォルダmy-databrew-bucketを含む という名前の S3 バケットがあるとしますdatabrew-input。そのフォルダに、同じファイル形式と.json拡張子を持つ多数の JSON ファイルがあるとします。コンソールで、 のソース URL を指定できますs3://my-databrew-bucket/databrew-input/。DataBrew コンソールで、このフォルダを選択できます。データセットは、そのフォルダ内のすべての JSON ファイルで構成されます。

DataBrew は S3 フォルダ内のすべてのファイルを処理できますが、次の条件が満たされる場合に限ります。

- フォルダ内のすべてのファイルは同じ形式です。
- フォルダ内のすべてのファイルには、同じファイル拡張子があります。

サポートされているファイル形式と拡張子の詳細については、「」を参照してくださいDataBrew input formats。

データセットとして複数のファイルを使用する場合のスキーマ

DataBrew データセットとして複数のファイルを使用する場合、スキーマはすべてのファイルで同じである必要があります。それ以外の場合、プロジェクトワークスペースは複数のファイルからスキーマの 1 つを自動的に選択し、残りのデータセットファイルをそのスキーマに準拠しようとします。この動作により、プロジェクトワークスペース中に表示されるビューが不規則になり、その結果、ジョブ出力も不規則になります。

ファイルに異なるスキーマが必要な場合は、複数のデータセットを作成して個別にプロファイリングする必要があります。

Amazon S3 のパラメータ化されたパスの使用

場合によっては、特定の命名規則に従うファイルを含むデータセット、または複数の Amazon S3 フォルダにまたがることができるデータセットを作成することがあります。または、同じデータセットを、特定のパラメータに依存するパスを持つ S3 ロケーションで定期的に生成される同一の構造化データに再利用することもできます。例としては、データ生成日に という名前のパスがあります。

DataBrew は、パラメータ化された S3 パスでこのアプローチをサポートしています。パラメータ化されたパスは、正規表現またはカスタムパスパラメータ、またはその両方を含む Amazon S3 URL です。

正規表現を使用して S3 パスを持つデータセットを定義する

パスの正規表現は、1 つ以上のフォルダの複数のファイルを照合し、同時にそれらのフォルダ内の無関係なファイルをフィルタリングするのに便利です。

以下にいくつかの例を示します。

- 名前が で始まるフォルダのすべての JSON ファイルを含むデータセットを定義しますinvoice。
- 名前に を含むフォルダ内のすべてのファイルを含むデータセットを定義します2020。

このタイプのアプローチは、データセット S3 パスで正規表現を使用して実装できます。これらの正規表現は、S3 URL の キーの任意の部分文字列を置き換えることができます (バケット名は置き換えません)。

S3 URL のキーの例については、以下を参照してください。ここで、 my-bucketはバケット名、米国東部 (オハイオ) はAWSリージョン、 puppy.pngはキー名です。

```
https://my-bucket.s3.us-west-2.amazonaws.com/puppy.png
```

パラメータ化された S3 パスでは、2 つの角括弧 (< と >) の間の文字は正規表現として扱われます。2 つの例を次に示します。

- s3://my-databrew-bucket/databrew-input/invoice<.*>/data.json はdata.json、名前databrew-inputが で始まる のすべてのサブフォルダ内の という名前のすべてのファイルと一致しますinvoice。
- s3://my-databrew-bucket/databrew-input/<.*>2020<.*>/ は、フォルダ内のすべてのファイルを名前2020に と照合します。

これらの例では、 は 0 文字以上.*に一致します。

> **Note**
>
> 正規表現は、S3 パスのキー部分、つまりバケット名の後に続く部分でのみ使用できます。したがって、 s3://my-databrew-bucket/<.*>-input/は有効ですが、 s3://my-<.*>-bucket/<.*>-input/は有効ではありません。

正規表現をテストして、必要な S3 URLs のみに一致し、不要な URL と一致しないことを確認することをお勧めします。

正規表現のその他の例を以下に示します。

- <\d{2}> は、 07や など、2 桁の連続した文字列と一致しますが03、 は一致しません1a2。
- <[a-z]+.*> は、1 つ以上の小文字のラテン文字で始まり、その後に 0 文字以上の他の文字を持つ文字列と一致します。例としては、a3、abc/def、または がありますがa-z、 ではありませんA2。
- <[^/]+> は、スラッシュ (/) を除く文字を含む文字列と一致します。S3 URL では、パス内のフォルダを分離するためにスラッシュが使用されます。
- <.*=.*> は、等号 (=) を含む文字列と一致します。たとえばmonth=02、、abc/day=2、または は一致しますが=10、 は一致しませんtest。
- <\d.*\d> は、数字で始まる文字列と終わる文字列に一致し、数字の間に 1abc2、、01-02-03など、他の文字を含めることができますが2020/Jul/21、 は使用できません123a。

カスタムパラメータを使用して S3 パスを持つデータセットを定義する

カスタムパラメータを使用してパラメータ化されたデータセットを定義すると、S3 ロケーションにパラメータを指定したい場合に正規表現を使用するよりも利点があります。

- 正規表現と同じ結果を得ることができ、正規表現の構文を知る必要はありません。パラメータは、「starts with」や「contains」などの使い慣れた用語を使用して定義できます。
- パスでパラメータを使用して動的データセットを定義する場合、「過去 1 か月」や「過去 24 時間」などの時間範囲を定義に含めることができます。これにより、データセット定義は後で新しい受信データで使用されます。

動的データセットを使用する場合の例を次に示します。

- 最終更新日またはその他の意味のある属性でパーティション分割された複数のファイルを 1 つのデータセットに接続するには。その後、これらのパーティション属性をデータセット内の追加の列としてキャプチャできます。
- データセット内のファイルを、特定の条件を満たす S3 の場所に制限するには。たとえば、S3 パスに のような日付ベースのフォルダが含まれているとしますfolder/2021/04/01/。この場合、日付をパラメータ化し、「2021 年 3 月 1 日から 2021 年 4 月 1 日の間」または「先週」などの特定の範囲に制限できます。

パラメータを使用してパスを定義するには、パラメータを定義し、次の形式を使用してパスに追加します。

```
s3://my-databrew-bucket/some-folder/{parameter1}/file-{parameter2}.json
```

> **Note**
>
> S3 パスの正規表現と同様に、パスのキー部分、つまりバケット名の後に続く部分でのみパラメータを使用できます。

パラメータ定義には、名前とタイプという 2 つのフィールドが必要です。タイプは、文字列、数値、日付のいずれかです。Date タイプのパラメータには、DataBrew が日付値を正しく解釈して比較できるように、日付形式の定義が必要です。必要に応じて、パラメータの一致条件を定義できます。DataBrew ジョブまたはインタラクティブセッションによってロードされているときに、パラメータの一致する値を列としてデータセットに追加することもできます。

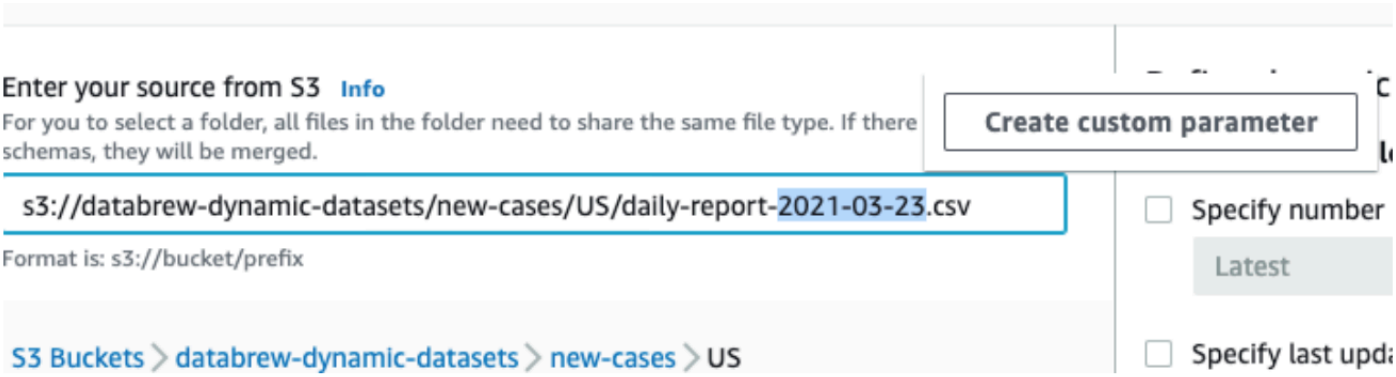
例

DataBrew コンソールでパラメータを使用して動的データセットを定義する例を考えてみましょう。この例では、入力データが次のような場所を使用して S3 バケットに定期的に書き込まれると仮定します。

- `s3://databrew-dynamic-datasets/new-cases/UR/daily-report-2021-03-30.csv`
- `s3://databrew-dynamic-datasets/new-cases/UR/daily-report-2021-03-31.csv`
- `s3://databrew-dynamic-datasets/new-cases/US/daily-report-2021-03-30.csv`
- `s3://databrew-dynamic-datasets/new-cases/US/daily-report-2021-03-31.csv`

ここには、米国のような国コードと、2021-03-30 のようなファイル名の日付の 2 つの動的部分があります。ここでは、すべてのファイルに同じクリーンアップレシピを適用できます。クリーンアップジョブを毎日実行するとします。このシナリオのパラメータ化されたパスを定義する方法は次のとおりです。

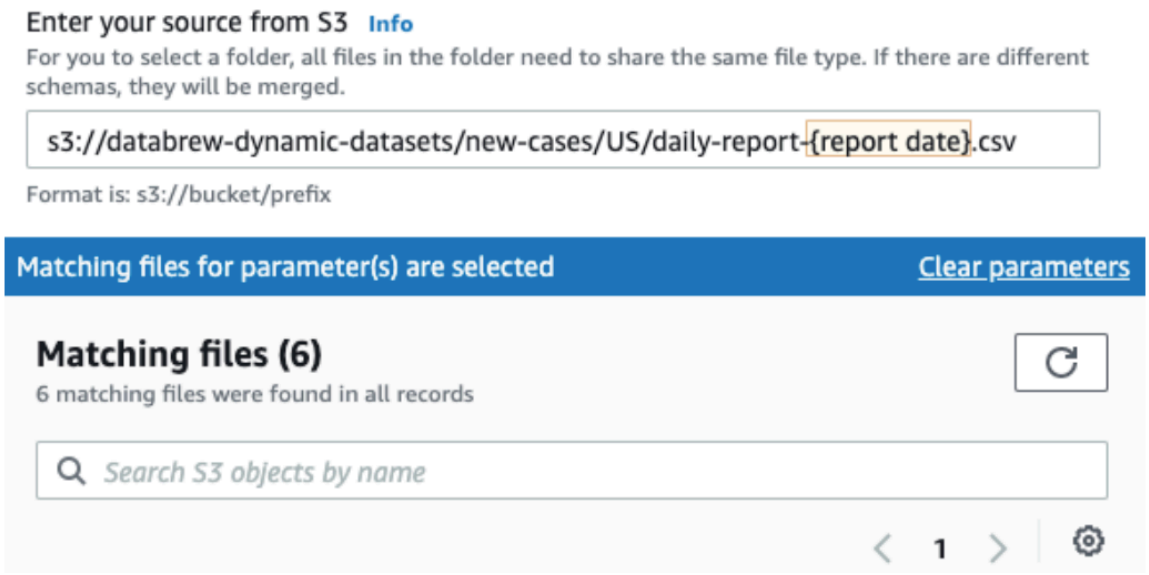
1. 特定のファイルに移動します。
2. 次に、日付などのさまざまな部分を選択し、パラメータに置き換えます。この場合、日付を置き換えます。



3. カスタムパラメータの作成のコンテキスト (右クリック) メニューを開き、そのプロパティを設定します。
   - 名前: レポート日
   - タイプ: 日付
   - 日付形式: yyyy-MM-dd (事前定義された形式から選択)
   - 条件 (時間範囲): 過去 24 時間
   - 列として追加: true (チェック済み)

   他のフィールドはデフォルト値のままにします。
4. [作成] を選択します。

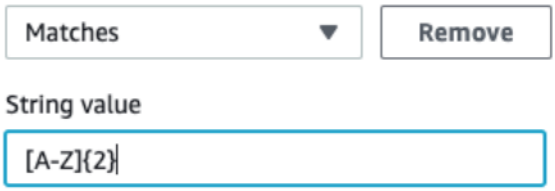
これを行うと、次のスクリーンショットのように更新されたパスが表示されます。



これで、国コードに対して同じことを行い、次のようにパラメータ化できます。

- 名前: 国コード
- タイプ: 文字列
- 列として追加: true (チェック済み)

すべての値が関連している場合は、条件を指定する必要はありません。たとえば、 new-casesフォルダには国コードを持つサブフォルダのみがあるため、条件は必要ありません。除外するフォルダが他にもある場合は、次の条件を使用できます。



このアプローチでは、新しいケースのサブフォルダに 2 つの大文字のラテン文字を含めるように制限します。

このパラメータ化の後、データセットに一致するファイルのみがあり、データセットの作成を選択できます。

> **Note**
>
> 条件で相対時間範囲を使用すると、データセットがロードされたときに時間範囲が評価されます。これは、「過去 24 時間」などの事前定義された時間範囲でも、「5 日前」などのカス

タム時間範囲でも当てはまります。この評価アプローチは、インタラクティブセッションの初期化中またはジョブの開始中にデータセットをロードするかどうかに適用されます。

データセットの作成を選択すると、動的データセットを使用する準備が整います。たとえば、最初にこれを使用してプロジェクトを作成し、インタラクティブな DataBrew セッションを使用してクリーンアップレシピを定義できます。その後、毎日実行するようにスケジュールされたジョブを作成できます。このジョブは、ジョブの開始時にパラメータの条件を満たすデータセットファイルにクリーンアップレシピを適用する場合があります。

動的データセットでサポートされている条件

条件を使用して、パラメータまたは最後に変更された日付属性を使用して、一致する S3 ファイルをフィルタリングできます。

以下に、各パラメータタイプでサポートされている条件のリストを示します。

文字列パラメータで使用される条件

DataBrew SDK の名前	SDK シノニム	DataBrew コンソールの名前	説明
is	eq、==	は正確に	パラメータの値は、条件で指定された値と同じです。
は ではありません	not eq、!=	Is not	パラメータの値は、条件で指定された値と同じではありません。
contains		Contains	パラメータの文字列値には、 条件で指定された値が含まれます。
が含まれていない		は含まれません	パラメータの文字列値に、 条件で指定された値が含まれていません。

DataBrew SDK の名前	SDK シノニム	DataBrew コンソールの名前	説明
starts_with		Starts with (で始まる)	パラメータの文字列値は、 条件で指定された値で始まります。
starts_with ではない		で始まらない	パラメータの文字列値は、 条件で指定された値で始まりません。
ends_with		Ends with	パラメータの文字列値は、 条件で指定された値で終わります。
ends_with ではない		で終わることはありません	パラメータの文字列値は、 条件で指定された値で終わりません。
matches		マッチ	パラメータの値は、条件で指定された正規表現と一致します。
一致しない		一致しない	パラメータの値が、条件で指定された正規表現と一致しません。

> **Note**
>
> 文字列パラメータのすべての条件では、大文字と小文字を区別する比較が使用されます。S3パスで使用されるケースが不明な場合は、「一致」条件を で始まる正規表現値で使用できます(?i)。これを行うと、大文字と小文字を区別しない比較になります。
> たとえば、文字列パラメータを で始めるがabc、 Abcまたは ABCも可能であるとします。この場合、条件値(?i)^abcとして で「一致」条件を使用できます。

数値パラメータで使用される条件

DataBrew SDK の名前	SDK シノニム	DataBrew コンソールの名前	説明
is	eq、==	は正確に	パラメータの値は、条件で指定された値と同じです。
は ではありません	not eq、!=	Is not	パラメータの値は、条件で指定された値と同じではありません。
less_than	lt、<	Less than	パラメータの数値が、 条件で指定された値より小さい。
less_than_equal	lte、<=	以下	パラメータの数値は、 条件で指定された値以下です。
greater_than	gt、>	Greater than	パラメータの数値が、 条件で指定された値より大きい。
greater_than_equal	gte、>=	以上	パラメータの数値は、 条件で指定された値以上です。

日付パラメータで使用される条件

DataBrew SDK の名前	DataBrew コンソールの名前	条件値の形式 (SDK)	説明
後に	Start	`2021-03-30T01:00:00Z` または のような ISO 8601 日付形式 `2021-03-30T01:00-07:00`	date パラメータの値は、 条件で指定された日付より後です。
前に	修了	`2021-03-30T01:00:00Z` または のような ISO 8601 日付形式 `2021-03-30T01:00-07:00`	date パラメータの値は、 条件で指定された日付より前です。
relative_after	開始 (相対)	`-48h` や など、正または負の時間単位の数`+7d`。	date パラメータの値は、 条件で指定された相対日付より後です。 相対日付は、データセットがロードされたとき、インタラクティブセッションが初期化されたとき、または関連するジョブが開始されたときに評価されます。これは、例で「現在」と呼ばれる瞬間です。

DataBrew SDK の名前	DataBrew コンソールの名前	条件値の形式 (SDK)	説明
relative_before	終了 (相対)	-48h や など、正または負の時間単位の数+7d。	date パラメータの値は、 条件で指定された相対日付より前です。 相対日付は、データセットがロードされたとき、インタラクティブセッションが初期化されたとき、または関連するジョブが開始されたときに評価されます。これは、例で「現在」と呼ばれる瞬間です。

SDK を使用する場合は、相対日付を の形式で指定します`±{number_of_time_units}{time_unit}`。次の時間単位を使用できます。

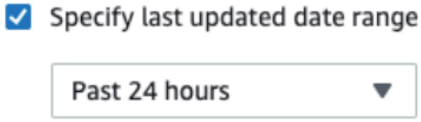
- -1h (1 時間前)
- +2d (2 日後)
- -120m (120 分前)
- 5,000 秒 (現在から 5,000 秒後)
- -3w (3 週間前)
- +400 万 (今から 4 か月後)
- -1 年 (1 年前)

相対日付は、データセットがロードされたとき、インタラクティブセッションが初期化されたとき、または関連するジョブが開始されたときに評価されます。これは、前の例で「現在」と呼ばれた瞬間です。

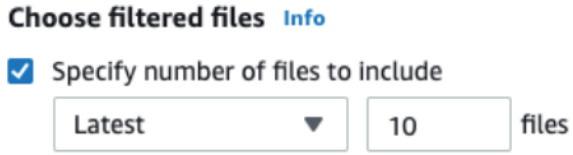
動的データセットの設定を構成する

パラメータ化された S3 パスを提供するだけでなく、複数のファイルを持つデータセットの他の設定も設定できます。これらの設定では、S3 ファイルを最終変更日でフィルタリングし、ファイル数を制限しています。

パスに日付パラメータを設定するのと同様に、一致するファイルが更新されたときの時間範囲を定義し、それらのファイルのみをデータセットに含めることができます。これらの範囲は、「2021 年 3 月 30 日」などの絶対日付または「過去 24 時間」などの相対範囲を使用して定義できます。



一致するファイルの数を制限するには、0 より大きいファイルの数と、一致するファイルが最新か最も古いかを選択します。



データ型

データセットの各列のデータは、次のいずれかのデータ型に変換されます。

- byte – 1 バイトの符号付き整数。数値の範囲は -128 ~ 127 です。
- short – 2 バイトの符号付き整数。数値の範囲は -32768 ~ 32767 です。
- integer – 4 バイトの符号付き整数。数値の範囲は -2147483648 ~ 2147483647 です。
- long – 8 バイトの符号付き整数。数値の範囲は -9223372036854775808 ~ 9223372036854775807 です。
- float – 4 バイトの単精度浮動小数点数。
- double – 8 バイトの倍精度浮動小数点数。
- decimal – 合計 38 桁、小数点以下 18 桁の符号付き小数。
- string – 文字列値。
- ブール値 – ブール型には、「true」と「false」または「yes」と「no」の 2 つの値のいずれかがあります。
- timestamp – 年、月、日、時、分、秒のフィールドで構成される値。

- date – 年、月、日フィールドで構成される値。

高度なデータ型

高度なデータ型は、DataBrew がプロジェクトの文字列列内で検出するデータ型であるため、データセットの一部ではありません。高度なデータ型の詳細については、「高度なデータ型」を参照してください。

高度なデータ型

高度なデータ型は、DataBrew がパターンマッチングによってプロジェクトの文字列列内で検出するデータ型です。文字列列をクリックすると、列内の値の 50% 以上がそのデータ型の基準を満たしている場合、列には対応するアドバンストデータ型としてフラグが付けられます。

DataBrew が検出できるデータ型は次のとおりです。

- 日付/タイムスタンプ
- SSN
- Phone number (電話番号)
- E メール
- クレジットカード
- 性別
- IP アドレス
- [URL]
- 郵便番号
- Country
- 通貨
- State
- 市

次の変換を使用して、高度なデータ型を操作できます。

- GET_ADVANCED_DATATYPE: 文字列列を指定すると、列の高度なデータ型があれば識別します。

- EXTRACT_ADVANCED_DATATYPE_DETAILS: 高度なデータ型の詳細を抽出します。
- ADVANCED_DATATYPE_FILTER: 高度なデータ型検出に基づいて現在のソース列をフィルタリングします。
- ADVANCED_DATATYPE_FLAG: 現在のソース列の値に基づいて新しいフラグ列を作成します。

でのデータ品質の検証AWS Glue DataBrew

データセットの品質を確保するために、ルールセットでデータ品質ルールのリストを定義できます。ルールセットは、さまざまなデータメトリクスを期待値と比較する一連のルールです。ルールの基準のいずれかが満たされない場合、ルールセット全体は検証に失敗します。その後、ルールごとに個別の結果を検査できます。検証に失敗するルールについては、必要な修正と再検証を行うことができます。

ルールの例は次のとおりです。

- 列の値は 0 ~ 100 "APY"です
- 列の欠損値が 5% group_nameを超えない

個々の列に各ルールを定義することも、選択した複数の列に個別に適用することもできます。次に例を示します。

- 列 、"rate"、 の最大値は 100 "pay"を超えることはありません"increase"。

ルールは、複数のシンプルなチェックで構成できます。これらをすべて true にするか、いずれかにするかを定義できます。次に例を示します。

- 列の値は、"asin-"列の値の長さ"ProductId"が 32 で始まる"ProductId"必要があります。

比較する値が 1 つnumber of duplicate valuesしかない max、min、 などの集計値、または列の各行の非集計値に対してルールを検証できます。後者の場合は、 などの「合格」しきい値を定義することもできますvalue in columnA > value in columnB for at least 95% of rows。

プロファイル情報と同様に、列レベルのデータ品質ルールは、文字列や数値などの単純な型の列に対してのみ定義できます。配列や構造など、複雑なタイプの列のデータ品質ルールを定義することはできません。プロファイル情報の操作の詳細については、「」を参照してくださいAWS Glue DataBrewプロファイルジョブの作成と操作。

データ品質ルールの検証

ルールセットを定義したら、検証のためにプロファイルジョブに追加できます。データセットには複数のルールセットを定義できます。

たとえば、1 つのルールセットに、許容できる条件が最小限であるルールが含まれている場合があります。そのルールセットの検証に失敗すると、データがそれ以上使用できない可能性があります。たとえば、機械学習トレーニングに使用されるデータセットのキー列に値が欠落しています。より厳密なルールで 2 番目のルールセットを使用して、データセットの品質が十分で、クリーンアップを必要としないかどうかを確認できます。

プロファイルジョブ設定で、特定のデータセットに定義された 1 つ以上のルールセットを適用できます。プロファイルジョブを実行すると、データプロファイルに加えて検証レポートが生成されます。検証レポートは、プロファイルデータと同じ場所で使用できます。プロファイル情報と同様に、DataBrew コンソールで結果を調べることができます。データセットの詳細ビューで、データ品質タブを選択して結果を表示します。プロファイル情報の操作の詳細については、「」を参照してくださいAWS Glue DataBrewプロファイルジョブの作成と操作。

検証結果に基づくアクション

DataBrew プロファイルジョブが完了すると、DataBrew はそのジョブ実行の詳細を含む Amazon CloudWatch イベントを送信します。また、データ品質ルールを検証するようにジョブを設定した場合、DataBrew は検証されたルールセットごとにイベントを送信します。イベントには、結果 (SUCCEEDED、FAILED、または ERROR) と、詳細なデータ品質検証レポートへのリンクが含まれます。その後、検証のステータスに応じて次のアクションを呼び出すことで、さらなるアクションを自動化できます。Amazon SNS 通知、AWS Lambda関数呼び出しなど、ターゲットアクションにイベントを接続する方法の詳細については、「Amazon EventBridge の開始方法」を参照してください。

DataBrew 検証結果イベントの例を次に示します。

```
{
  "version": "0",
  "id": "fb27348b-112d-e7c2-560d-85e7c2c09964",
  "detail-type": "DataBrew Ruleset Validation Result",
  "source": "aws.databrew",
  "account": "123456789012",
  "time": "2021-11-18T13:15:46Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "datasetName": "MyDataset",
    "jobName": "MyProfileJob",
    "jobRunId": "db_f07954d20d083de0c1fc1eee11498d8635ee5be4ca416af27d33933e91ff4e6e",
    "rulesetName": "MyRuleset",
    "validationState": "FAILED",
```

```
    "validationReportLocation": "s3://MyBucket/MyKey/
MyDataset_f07954d20d083de0c1fc1eee11498d8635ee5be4ca416af27d33933e91ff4e6e_dq-
validation-report.json"
  }
}
```

などのイベントの属性、sourceおよび 属性のネストされたプロパティを使用してdetail-type、Amazon Eventbridge でイベントパターンdetailを作成できます。 https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-event-patterns.html#eb-create-patternたとえば、DataBrew ジョブから失敗したすべての検証に一致するイベントパターンは次のようになります。

```
{
  "source": ["aws.databrew"],
  "detail-type": ["DataBrew Ruleset Validation Result"],
  "detail": {
   "validationState": ["FAILED"]
  }
}
```

ルールセットを作成し、そのルールを検証する例については、「」を参照してくださいデータ品質ルールを使用したルールセットの作成。DataBrew での CloudWatch イベントの操作の詳細については、「」を参照してください。 CloudWatch Events による DataBrew の自動化

データ品質ルールを使用したルールセットの作成

次の手順では、ルールセットを作成し、データセットに適用する例を示します。ルールセットは、さまざまなデータメトリクスを期待値と比較する一連のルールです。その後、プロファイルジョブでこのルールセットを使用して、含まれるデータ品質ルールを検証できます。

データ品質ルールを使用してルールセットの例を作成するには

1. にサインインAWS マネジメントコンソールし、https://console.aws.amazon.com/databrew/ で DataBrew コンソールを開きます。
2. ナビゲーションペインから DQ RULES を選択し、データ品質ルールセットの作成を選択します。
3. ルールセットの名前を入力します。必要に応じて、ルールセットの説明を入力します。
4. 関連付けられたデータセットで、ルールセットに関連付けるデータセットを選択します。

データセットを選択すると、右側にデータセットプレビューペインが表示されます。

5. データセットプレビューペインのプレビューを使用して、作成するデータ品質ルールを決定する際にデータセットの値とスキーマを調べます。プレビューでは、データに関する潜在的な問題に関するインサイトを得ることができます。

   データベースなどの一部のデータソースは、データプレビューをサポートしていません。その場合、まずデータ品質ルールを検証せずにプロファイルジョブを実行できます。その後、データプロファイルを使用して、データスキーマと値の分布に関する情報を取得できます。

6. レコメンデーションタブを確認します。このタブには、ルールセットの作成時に使用できるルールの提案が一覧表示されます。すべてのレコメンデーション、一部のレコメンデーション、またはレコメンデーションなしを選択できます。

   関連するレコメンデーションを選択したら、ルールセットに追加を選択します。

   これにより、ルールセットにルールが追加されます。必要に応じてパラメータを検査し、変更します。データ品質ルールで使用できるのは、文字列、数値、ブール値などの単純な型の列のみです。

7. 別のルールを追加 を選択して、レコメンデーションの対象ではないルールを追加します。ルール名を変更して、後で検証結果を解釈しやすくなります。

8. データ品質チェックスコープを使用して、このルールの各チェックごとに個々の列を選択するか、選択した列のグループに適用するかを選択します。たとえば、データセットに 0 ~ 100 の値を持つ数値列が複数ある場合、ルールを 1 回定義し、これらの列をすべて選択して、このルールでチェックできます。

9. ルールに複数のチェックがある場合、ルールの成功基準ドロップダウンで、すべてのチェックを満たすか、基準を満たすかを選択します。

10. データ品質チェックドロップダウンでこのルールを検証するために実行するチェックを選択します。利用可能なチェックの詳細については、「」を参照してください利用可能なチェック。

11. データ品質チェックスコープの各列に個別チェックを選択した場合は、列を選択します。 このチェックの列名を選択または入力します。

12. チェックに応じてパラメータを選択します。一部の条件は指定されたカスタム値のみを受け入れ、一部の条件は別の列への参照もサポートします。

13. 文字列値のインクルード条件などの列値をチェックする場合は、「合格」しきい値を指定できます。 たとえば、値の 95% 以上が条件を満たすようにするには、しきい値の条件としてより大きい を選択し、しきい値セクションの次のドロップダウンに「 (パーセント) 行」のままにする必要があります。または、値が欠落している条件が true の行を 10 行以下にする場合は、条件とし

て小なりを選択し、しきい値に 10 を入力し、次のドロップダウンで行を選択します。検証中に異なるサイズのサンプルを使用している場合は、異なる結果が得られる可能性があることに注意してください。

14. 必要に応じてルールを追加します。
15. ルールセットの作成 を選択します。

ルールセットを使用したプロファイルジョブの作成

前述のようにルールセットを作成すると、データ品質ルールページに移動し、アカウント内のすべてのルールセットが表示されます。

ルールセットを含むプロファイルジョブを作成するには

1. 以前に作成したルールセットの名前を選択して、その詳細を表示します。
2. ルールセットを使用してプロファイルジョブを作成するを選択します。

   ジョブ名は自動的に入力されますが、必要に応じて変更できます。
3. ジョブ実行サンプルでは、データセット全体または限られた数の行を実行するように選択できます。

   制限されたサンプルサイズを実行する場合は、特定のルールでは、データセット全体と結果が異なる場合があります。
4. ジョブ出力設定で、ジョブ出力の S3 の場所を選択します。アクセス可能な名前付き Amazon S3 バケット内の任意のフォルダを選択します。このバケットが存在しないフォルダ名を入力すると、このフォルダが作成されます。

   プロファイルジョブが正常に完了すると、このフォルダには JSON 形式のデータおよびデータ品質ルール検証レポートのプロファイルが含まれます。
5. 「データ品質ルール」では、ルールセットが「データ品質ルールセット名」にリストされていることに注意してください。
6. アクセス許可で、ロールを選択または作成して、DataBrew に入力 Amazon S3 の場所から読み取り、ジョブの出力場所に書き込むためのアクセス許可を付与します。ロールの準備が整っていない場合は、新しい IAM ロールの作成 を選択します。
7. 必要に応じてAWS Glue DataBrewプロファイルジョブの作成と操作、「」の説明に従って、その他のオプション設定を変更します。
8. ジョブの作成と実行を選択します。

データ品質ルールの検証結果の検査と更新

プロファイルジョブが完了したら、データ品質ルールの検証結果を表示し、必要に応じてルールを更新できます。

データ品質ルールの検証データを表示するには

1. DataBrew コンソールで、データプロファイルの表示を選択します。これにより、データセットのデータプロファイルの概要タブが表示されます。
2. データ品質ルールタブを選択します。このタブでは、すべてのデータ品質ルールの結果を表示できます。
3. 個々のルールを選択すると、そのルールの詳細が表示されます。

検証に失敗したルールについては、必要な修正を行うことができます。

データ品質ルールを更新するには

1. ナビゲーションペインで、DQ RULES を選択します。
2. データ品質ルールセット名で、編集する予定のルールを含むデータセットを選択します。
3. 変更するルールを選択し、編集を選択します。
4. 必要な修正を行い、ルールセットの更新を選択します。
5. ジョブを再実行します。すべての検証に合格するまで、このプロセスを繰り返します。

利用可能なチェック

次の表に、ルールで使用できるすべての条件のリファレンスを示します。集約された条件は、同じルールの非集約条件と組み合わせることはできません。

> **Note**
>
> SDK ユーザーの場合、同じルールを複数の列に適用するには、ルールの ColumnSelectors 属性を使用し、名前または正規表現を使用して検証済み列を指定します。この場合、暗黙的な CheckExpression を使用する必要があります。たとえば、選択した各列の値を指定された値と比較“> :val”します。DataBrew は、動的データセットで FilterExpression を定義するために暗黙的な構文を使用します。各チェックに列 (複数可) を個別に指定する場合

は、ColumnSelectors 属性を設定しないでください。代わりに、明示的な式を指定します。例えば、ルールの CheckExpression “`:col > :val`”として。

条件の種類	データの品質チェック	追加パラメータ	比較タイプ	SDK 構文の例
データセット条件を集約する	行数		カスタム値との数値比較	`"CheckExpression": "AGG(ROWS_COUNT) > :val", "SubstitutionMap": {":val", "10000"}`
	列の数		カスタム値との数値比較	`"CheckExpression": "AGG(COLUMNS_COUNT) == :val", "SubstitutionMap": {":val", "20"}`
	重複行		カスタム値との数値比較	`"CheckExpression": "AGG(DUPLICATE_ROWS_COUNT) < :val", "SubstitutionMap": {":val", "100"}`

条件の種類	データの品質チェック	追加パラメータ	比較タイプ	SDK 構文の例
				または "CheckExpression": "AGG(DUPLICATE_ROWS_PERCENTAGE) < :val", "SubstitutionMap": {":val", "5"}

条件の種類	データの品質チェック	追加パラメータ	比較タイプ	SDK 構文の例
列統計条件を集計する	欠落した値		カスタム値との数値比較	"CheckExpression": "AGG(MISSING_VALUES_COUNT) < :val", "SubstitutionMap": {":val", "100"} または "CheckExpression": "AGG(MISSING_VALUES_PERCENTAGE) < :val", "SubstitutionMap": {":val", "5"}

条件の種類	データの品質チェック	追加パラメータ	比較タイプ	SDK 構文の例

	重複する値		カスタム値との数値比較	`"CheckExpression": "AGG(DUPLICATE_VALUES_COUNT) < :val", "SubstitutionMap": {":val", "100"}`
				
				または
				
				`"CheckExpression": "AGG(DUPLICATE_VALUES_PERCENTAGE) < :val", "SubstitutionMap": {":val", "5"}`

条件の種類	データの品質チェック	追加パラメータ	比較タイプ	SDK 構文の例
	有効値		カスタム値との数値比較	"CheckExpression": "AGG(VALID_VALUES_COUNT) > :val", "SubstitutionMap": {":val", "10000"} または "CheckExpression": "AGG(VALID_VALUES_PERCENTAGE) > :val", "SubstitutionMap": {":val", "95"}

条件の種類	データの品質チェック	追加パラメータ	比較タイプ	SDK 構文の例

	個別の値		カスタム値との数値比較	`"CheckExpression": "AGG(DISTINCT_VALUES_COUNT) > :val", "SubstitutionMap": {":val", "1000"}`
				
				または
				
				`"CheckExpression": "AGG(DISTINCT_VALUES_PERCENTAGE) >= :val", "SubstitutionMap": {":val", "50"}`

条件の種類	データの品質チェック	追加パラメータ	比較タイプ	SDK 構文の例
	一意の値		カスタム値との数値比較	"CheckExpression": "AGG(UNIQUE_VALUES_COUNT) > :val", "SubstitutionMap": {":val", "100"} または "CheckExpression": "AGG(UNIQUE_VALUES_PERCENTAGE) > :val", "SubstitutionMap": {":val", "20"}

条件の種類	データの品質チェック	追加パラメータ	比較タイプ	SDK 構文の例
	外れ値	Z スコアのしきい値	カスタム値との数値比較	"CheckExpression": "AGG(Z_SCORE_OUTLIERS_COUNT, :zscore_dev) < :val", "SubstitutionMap": {":zscore_dev": "4", ":val", "100"} または "CheckExpression": "AGG(Z_SCORE_OUTLIERS_PERCENTAGE) < :val", "SubstitutionMap": {":val", "5"}

条件の種類	データの品質チェック	追加パラメータ	比較タイプ	SDK 構文の例

	値分散統計	統計名 (次の表を参照)	カスタム値との数値比較	`"CheckExpression": "AGG(<STAT_NAME>) < :val", "SubstitutionMap": {":val", "100"}`
				
				または
				
				`"CheckExpression": "AGG(<STAT_NAME>, :param) < :val", "SubstitutionMap": {":param": "0.25", :val", "5"}`
				
				Note
				可能なSTAT_NAME 値については、次の表を参照してください。

条件の種類	データの品質チェック	追加パラメータ	比較タイプ	SDK 構文の例

	数値統計	統計名 (次の表を参照)	カスタム値との数値比較	`"CheckExpression": "AGG(<STAT_NAME>) < :val", "SubstitutionMap": {":val", "100"}`
				
				または
				
				`"CheckExpression": "AGG(<STAT_NAME>, :param) < :val", "SubstitutionMap": {":param": "0.25", :val", "5"}`
				
				Note
				可能なSTAT_NAME 値については、次の表を参照してください。

条件の種類	データの品質チェック	追加パラメータ	比較タイプ	SDK 構文の例
非集計 (しきい値を受け入れる)	値は正確に		値のリストとの正確な比較	`"CheckExpression": ":col IN :list", "SubstitutionMap": {":col": "`size`", ":list": "[\"S\",\"M\",\"L\",\"XL\"]"}`
	値が正確にない		値は、リストからの値と完全に一致してはいけません	`"CheckExpression": ":col NOT IN :list", "SubstitutionMap": {":col": "`domain`", ":list": "[\"GOV\",\"ORG\"]"}`

条件の種類	データの品質チェック	追加パラメータ	比較タイプ	SDK 構文の例
	文字列値		カスタム値または他の文字列列との文字列比較	"CheckExpression": ":col STARTS_WITH :val", "SubstitutionMap": {":col": "`url`", ":val": "http"} または "CheckExpression": ":col1 contains :col2", "SubstitutionMap": {":col1": "`url`", ":col2": "`company_name`"}

条件の種類	データの品質チェック	追加パラメータ	比較タイプ	SDK 構文の例
	数値		カスタム値または他の数値列との数値比較	"CheckExpression": ":col IS_BETWEEN :val1 and :val2", "SubstitutionMap": {":col": "`APY`", ":val1": "0", ":val2": "10"} または "CheckExpression": ":col1 <= :col2", "SubstitutionMap": {":col1": "`bank_rate`", ":col2": "`fed_rate`"}

条件の種類	データの品質チェック	追加パラメータ	比較タイプ	SDK 構文の例
	値文字列の長さ		カスタム値または他の数値列との数値比較	"CheckExpression": "length(:col) IS_BETWEEN :val1 and :val2", "SubstitutionMap": {":col": "`identifier`", ":val1": "8", ":val2": "12"} または "CheckExpression": "length(:col1) <= :col2", "SubstitutionMap": {":col1": "`name`", ":col2": "`max_name_len`"}

数値比較

DataBrew は、数値比較のために次のオペレーションをサポートしています。Is equals (==)、Is not equals (!=)、Less than (<)、Less than equals (<=)、Greater than (>)、Greater than equals (>=)、Is between (is_between :val1 and :val2)。

文字列比較

次の文字列比較がサポートされています。「 で始まる」、「 で始まらない」、「 で終わる」、「 で終わる」、「 を含む」、「 を含まない」、「 等しい」、「 等しくない」、「 一致」、「 一致しない」。

次の表は、値分布統計と数値統計に使用できる統計を示しています。

データの品質チェック	統計名	追加パラメータ	SDK 構文
値分散統計	最小		`"CheckExpression": "AGG(MAX) < :val", "SubstitutionMap": {":val", "100"}`
	最大		`"CheckExpression": "AGG(MIN) > :val", "SubstitutionMap": {":val", "0"}`
	中央値		`"CheckExpression": "AGG(MEDIAN) >= :val", "SubstitutionMap": {":val", "50"}`

データの品質チェック	統計名	追加パラメータ	SDK 構文
	平均値		`"CheckExpression": "AGG(MEAN) <= :val", "SubstitutionMap": {":val", "10"}`
	モード		`"CheckExpression": "AGG(MODE) > :val", "SubstitutionMap": {":val", "0"}`
	標準偏差		`"CheckExpression": "AGG(STANDARD_DEVIATION) > :val", "SubstitutionMap": {":val", "0"}`
	エントロピー		`"CheckExpression": "AGG(ENTROPY) > :val", "SubstitutionMap": {":val", "0"}`

データの品質チェック	統計名	追加パラメータ	SDK 構文
数値統計	合計		`"CheckExpression": "AGG(SUM) > :val", "SubstitutionMap": {":val", "0"}`
	尖度		`"CheckExpression": "AGG(KURTOSIS) > :val", "SubstitutionMap": {":val", "0"}`
	歪み		`"CheckExpression": "AGG(SKEWNESS) > :val", "SubstitutionMap": {":val", "0"}`
	Variance		`"CheckExpression": "AGG(VARIANCE) > :val", "SubstitutionMap": {":val", "0"}`

データの品質チェック	統計名	追加パラメータ	SDK 構文
	絶対偏差		`"CheckExpression": "AGG(MEDIAN_ABSOLUTE_DEVIATION) > :val", "SubstitutionMap": {":val", "0"}`
	分位数	分位数: '0.25'、'0.5'、'0.75' のいずれか	`"CheckExpression": "AGG(QUANTILE, :pct) > :val", "SubstitutionMap": {":pct": "0.25", ":val", "0"}`

AWS Glue DataBrewプロジェクトの作成と使用

ではAWS Glue DataBrew、プロジェクトがデータ分析と変換の取り組みの中心です。

プロジェクトを作成するときは、次の 2 つの基本コンポーネントをまとめます。

- ソースデータへの読み取り専用アクセスを提供するデータセット。詳細については、「を使用したデータへの接続AWS Glue DataBrew」を参照してください。
- DataBrew データ変換をデータセットに適用するレシピ。詳細については、「AWS Glue DataBrew レシピの作成と使用」を参照してください。

DataBrew コンソールは、非常にインタラクティブで直感的なユーザーインターフェイスでプロジェクトを表示します。数百のデータ変換を試すことをお勧めします。これにより、データがどのように機能し、データにどのような影響があるかを学習できます。

プロジェクトビューに表示されるデータは、データセットのサンプルです。データセットは非常に大きく、数千行または数百万行もあるため、サンプルを使用すると、さまざまな方法でサンプルデータを変換しながら DataBrew コンソールが応答し続けることができます。デフォルトでは、サンプルはデータセットの最初の 500 行のデータで構成されます。サンプルサイズと選択する行には、さまざまな設定を選択できます。

サンプルデータを変換するとき、DataBrew は、これまで適用した変換のstep-by-stepのシリーズであるプロジェクトレシピの構築と改良に役立ちます。work-in-progressレシピは自動的に保存されるため、いつでもプロジェクトビューを離れ、後で戻り、中断した場所をピックアップできます。

レシピを使用する準備ができたら、公開できます。レシピを発行すると、DataBrew ジョブサブシステムで使用できるようになり、データセット全体にレシピを適用したり、データの構造、コンテンツ、統計特性を理解できる広範なデータプロファイルを作成したりできます。

トピック

- プロジェクトの作成
- DataBrew プロジェクトセッションの概要
- プロジェクトの削除

プロジェクトの作成

プロジェクトを作成するには、次の手順に従います。

プロジェクトを作成するには

1. にサインインAWS マネジメントコンソールし、DataBrew コンソール を開きます。
2. ナビゲーションペインで、PROJECTS を選択します。次に、プロジェクトの作成を選択します。
3. プロジェクトの名前を入力します。次に、プロジェクトにアタッチするレシピを選択します。

   - 最初から開始する場合は、新しいレシピを作成する を選択します。これにより、新しい空のレシピが作成され、プロジェクトにアタッチされます。
   - このプロジェクトで使用するレシピが以前に公開されている場合は、既存のレシピの編集を選択します。レシピが現在別のプロジェクトにアタッチされているか、そのプロジェクトにジョブが定義されている場合、新しいプロジェクトで使用することはできません。レシピを参照を選択して、使用可能なレシピを確認します。
   - 以前に公開された既存のレシピがあり、そのステップをインポートする場合は、レシピからステップをインポートを選択し、次の手順を実行します。

     1. レシピを参照を選択して、使用可能なレシピを確認します。
     2. 使用するレシピの公開バージョンを選択します。レシピには、プロジェクトビューでの作業中に公開した頻度に応じて、複数のバージョンを含めることができます。
     3. レシピの表示ステップを選択して、レシピのデータ変換を調べます。
4. レシピを作成したら、データセットの選択ペインで操作するデータセットを選択します。

   - データセット – 以前に作成したデータセットを選択します。詳細については、「プロジェクトの作成」を参照してください。
   - サンプルファイル – によって維持されるサンプルデータに基づいて新しいデータセットを作成しますAWS。このサンプルデータは、独自のデータを提供することなく、DataBrew ができることを調べるための優れた方法です。データセットの名前を必ず入力してください。
   - 新しいデータセット – 新しいデータセットを作成します。詳細については、「プロジェクトの作成」を参照してください。
5. アクセス許可には、DataBrew が Amazon S3 入力場所から読み取ることを許可するAWS Identity and Access Management(IAM) ロールを選択します。 Amazon S3 AWSアカウントが所有する S3 ロケーションでは、`AwsGlueDataBrewDataAccessRole`サービスマネージドロールを選択できます。これにより、DataBrew は所有している S3 リソースにアクセスできます。
6. サンプリングペインには、データセットからデータのサンプルを構築するための DataBrew のオプションがあります。

Type で、DataBrew がデータセットから行を取得する方法を選択します。

- 最初の n 行を使用して、データセットの最初の行に基づいてサンプルを作成します。
- ランダム行を使用して、データセット内の行のランダムな選択に基づいてサンプルを作成します。
- サンプルに表示する行数を選択します。最大 5,000 行まで、500、1,000、2,500、またはカスタムサンプルサイズを選択します。サンプルサイズを小さくすると、DataBrew は変換をより迅速に実行できるため、レシピの開発にかかる時間を節約できます。サンプルサイズを大きくすると、基盤となるソースデータの構成がより正確に反映されます。ただし、プロジェクトセッションの初期化とインタラクティブ変換は遅くなります。

7. (オプション) タグを選択して、データセットにタグをアタッチします。

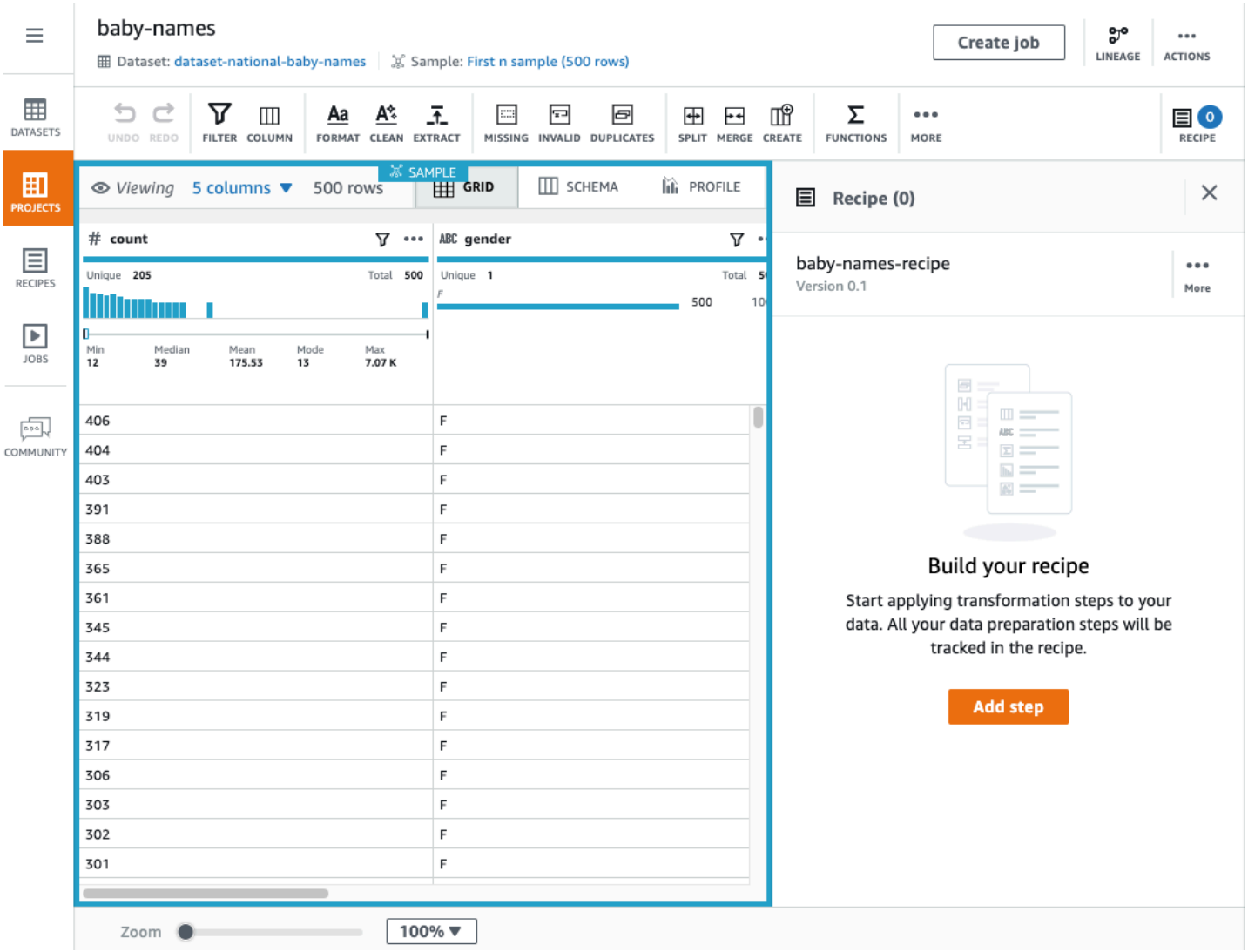
   タグは、ユーザー定義のキーとオプションの値で構成されるシンプルなラベルです。これにより、DataBrew プロジェクトを目的、所有者、環境、またはその他の基準で管理、検索、フィルタリングしやすくなります。

8. 設定が目的どおりになったら、ジョブの作成を選択します。

DataBrew は、必要に応じて新しいデータセットを作成し、必要に応じて新しいレシピを作成し、データサンプルを構築し、インタラクティブなプロジェクトセッションを作成します。このプロセスが完了するまでに数分かかる場合があります。プロジェクトを使用する準備ができたら、データサンプルの使用を開始できます。

DataBrew プロジェクトセッションの概要

DataBrew プロジェクトセッションでは、インタラクティブワークスペース内で作業します。



左側のペインには、データの現在のビューが表示されます。右側のペインには、現在空のプロジェクトの変換レシピが表示されます。

データグリッドの右上隅には、GRID、、 SCHEMAの 3 つのタブがありますPROFILE。これらのタブのいずれかを選択すると、ワークスペースに対応するビューが表示されます。次にそれらのビューについて説明します。

グリッドビュー

グリッドビューはデフォルトのビューで、サンプルは表形式で表示されます。グリッドビューの簡単なウォークスルーには、次の手順を使用します。

グリッドビューのウォークスルーを行うには

1. まず、スペース全体を表示します。

a. 左右にスクロールして、すべての列を表示します。

b. 上下にスクロールして、すべてのデータ値を表示します。

c. ワークスペースの下部にあるズームコントロールを使用して、グリッドの拡大レベルを調整します。

2. 右上には、表示されるサンプルの列の数と、サンプル内の現在の行数が表示されます。

   表示する列を変更するには、N 列リンクを選択します (N は現在表示されている列の数です)。目的の列を選択し、選択した列を表示を選択します。

3. これで、DataBrew 変換の実験を開始できます。次の操作を試してください：

   a. 変換ツールバーから、形式の選択、大文字への変更を選択します。

   b. ソース列で、文字データを含む列を選択します。

   c. その他の設定はデフォルト値のままにしておきます。

   d. 変換されたデータがどのように表示されるかを確認するには、変更のプレビューを選択します。次に、この変換をレシピに追加するには、適用を選択します。

   データ変換を適用するたびに、DataBrew はそれをレシピの作業コピーに追加します。これはワークスペースの右側に表示されます。

4. 次の操作を試してください：

   a. 変換ツールバーから、関数に基づいて作成を選択します。

   b. 関数を選択する で、 を選択しますSQUARE ROOT。

   c. ソース列で、数値データを含む列を選択します。

   d. 他の設定はデフォルトのままにしておきます。

   e. 変更のプレビューを選択して、変換されたデータがどのように表示されるかを確認します。次に、この変換をレシピに追加するには、適用を選択します。

5. RECIPE を選択して、右上のレシピペインを折りたたみます。レシピペインを展開するには、もう一度 RECIPE を選択します。

レシピの新しいバージョンの公開

変換の適用を続けると、レシピのステップ数が増加します。レシピの新しいバージョンはいつでも公開できます。レシピを発行すると、DataBrew の他の場所で使用できるようになります。これによ

り、レシピジョブを実行して、プロジェクトデータサンプルのみを変換するのではなく、データセット全体を変換できます。

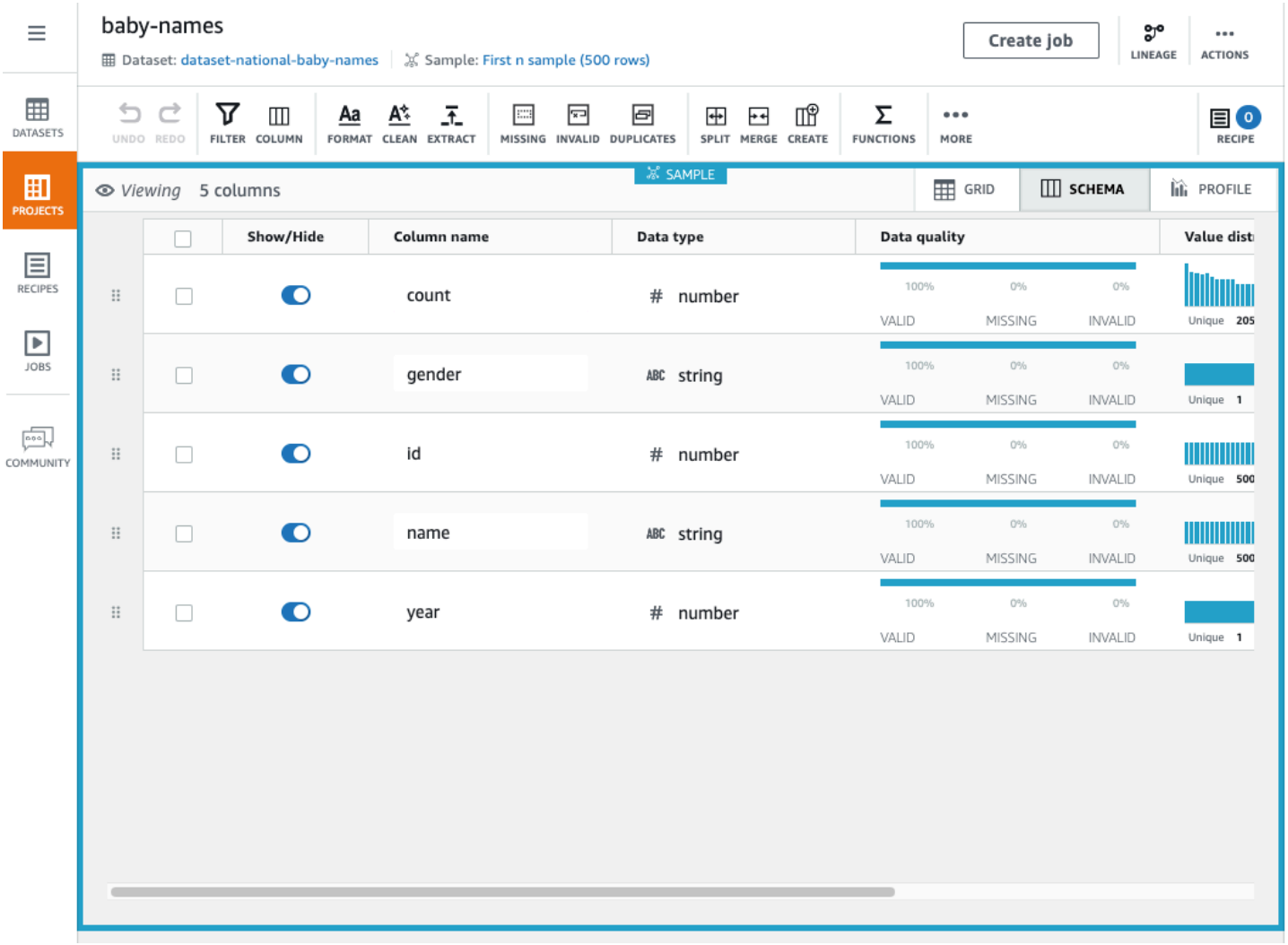
レシピを公開すると、レシピ開発に対する段階的な反復的なアプローチも推奨されます。レシピの新しいバージョンを公開できるため、必要に応じて「最後の既知の正常な」レシピバージョンにフォールバックできます。

レシピの新しいバージョンを発行するには

- レシピペインで、発行を選択します。このバージョンのレシピの説明を入力し、発行を選択します。

スキーマビュー

SCHEMA タブを選択すると、次のスクリーンショットに示すようにビューが変更されます。



スキーマビューでは、各列のデータ値に関する統計を表示できます。

左端の列で、表示/非表示の横にあるデータ列のいずれかを選択します。列の詳細ペインが右側に表示されます。このペインには、列値の統計の概要が表示されます。

列名を変更するには、列名に新しい名前を入力します。

列をドラッグアンドドロップすることで、列の順序を変更できます。

プロファイルビュー

PROFILE タブを選択すると、プロジェクトに関する詳細なボリューメトリック情報が表示されます。これを行う前に、DataBrew ジョブを実行してプロファイルを作成します。

プロファイルビューのウォークスルーを行うには

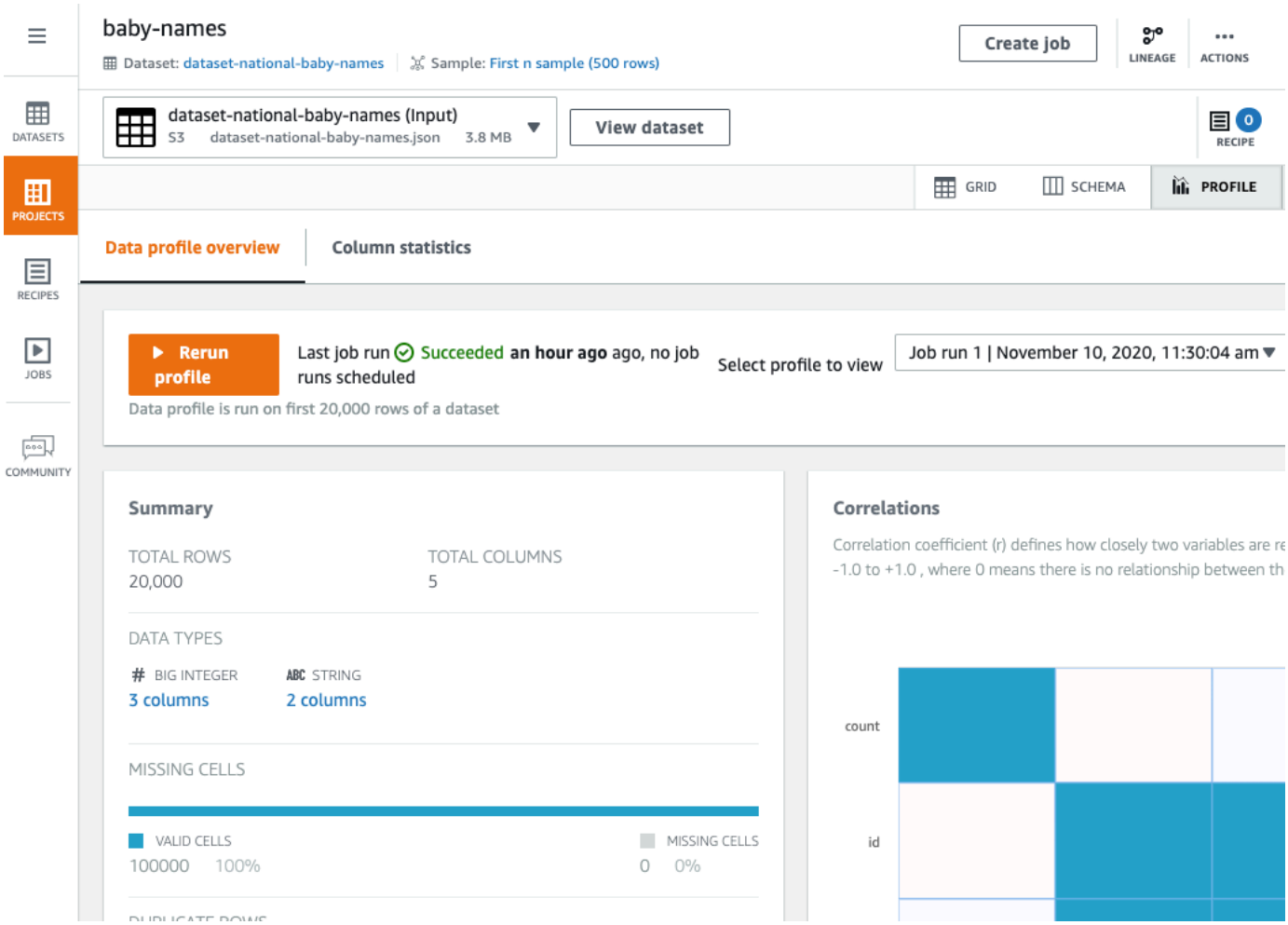
1. ジョブの作成 を選択し、ジョブの名前を入力します。
2. ジョブ出力で、ファイルタイプの CSV を選択します。
3. DataBrew からのジョブ出力を書き込む Amazon S3 バケットとフォルダをAWSアカウントで検索または作成します。

   - この Amazon S3 バケットとフォルダが既にある場合は、参照を選択してそれらを見つけます。両方に対する書き込みアクセス許可があることを確認してください。
   - この Amazon S3 バケットとフォルダがない場合は、作成します。
     1. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
     2. Amazon S3 バケットがない場合は、バケットの作成を選択します。バケット名に、新しいバケットの一意の名前を入力します。[バケットを作成] を選択します。
     3. バケットのリストから、使用するバケットを選択します。
     4. Create folder (フォルダの作成) を選択します。フォルダ名に と入力し`databrew-output`、フォルダの作成を選択します。
4. アクセス許可には、DataBrew が Amazon S3 出力場所に書き込むことを許可する IAM ロールを選択します。

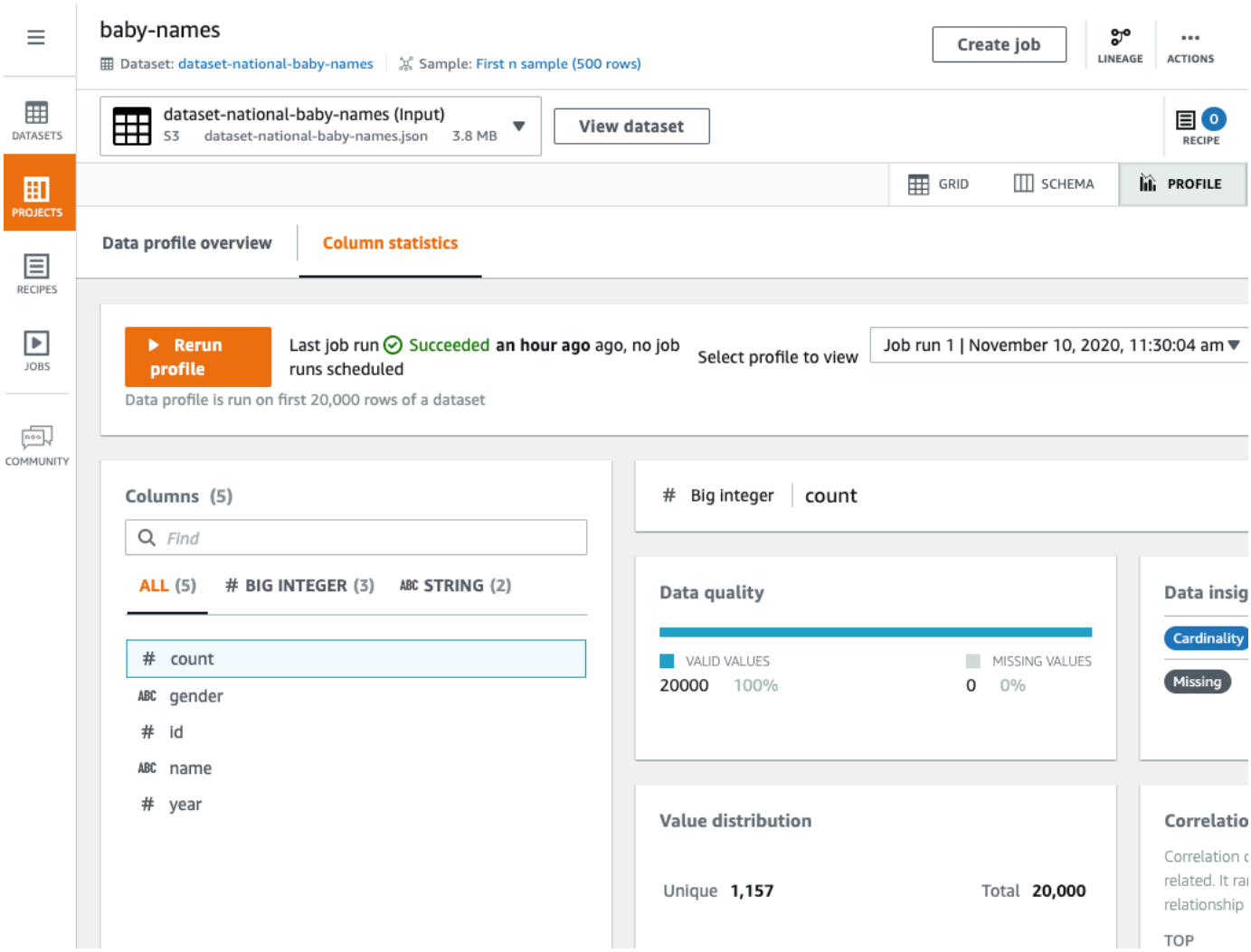
   AWSアカウントが所有する S3 ロケーションでは、`AwsGlueDataBrewDataAccessRole`サービスマネージドロールを選択できます。これにより、DataBrew は所有している S3 リソースにアクセスできます。
5. 他の設定はデフォルトのままにし、ジョブの作成と実行を選択します。

6. ジョブが実行されて完了すると、ワークスペースにはデータプロファイルのグラフィカルな概要が表示されます。

   データプロファイルの概要タブには、次のスクリーンショットに示すように、データの特性の概要が表示されます。



   列統計タブには、データ値のcolumn-by-column内訳が表示されます。



プロジェクトの削除

プロジェクトが不要になった場合は、削除できます。

プロジェクトを削除するには

1. ナビゲーションペインで、PROJECTS を選択します。
2. 削除するプロジェクトを選択し、アクションで削除を選択します。

AWS Glue DataBrewレシピの作成と使用

DataBrew では、レシピは一連のデータ変換ステップです。これらのステップをデータのサンプルに適用するか、同じレシピをデータセットに適用できます。

レシピを開発する最も簡単な方法は、DataBrew プロジェクトを作成することです。このプロジェクトでは、データのサンプルをインタラクティブに操作できます。詳細については、「」を参照してくださいAWS Glue DataBrewプロジェクトの作成と使用。プロジェクト作成ワークフローの一環として、新しい (空の) レシピが作成され、プロジェクトにアタッチされます。その後、データ変換を追加してレシピの構築を開始できます。

> **Note**
>
> 1 つの DataBrew レシピに最大 100 個のデータ変換を含めることができます。

レシピの開発を進めると、レシピを公開することで作業を保存できます。DataBrew は、レシピの公開バージョンのリストを保持します。レシピジョブで任意の公開バージョンを使用して、レシピを (レシピジョブで) 実行してデータセットを変換できます。レシピステップのコピーをダウンロードして、レシピを他のプロジェクトや他のデータセット変換で再利用することもできます。

DataBrew レシピは、AWS Command Line Interface(AWS CLI) またはいずれかのAWS SDKs を使用してプログラムで開発することもできます。DataBrew API では、変換はレシピアクションと呼ばれます。

> **Note**
>
> インタラクティブな DataBrew プロジェクトセッションでは、適用する各データ変換によって DataBrew API が呼び出されます。これらの API コールは、behind-the-scenesの詳細を知ることなく、自動的に行われます。

プログラマーでなくても、レシピの構造と DataBrew がレシピアクションを整理する方法を理解すると便利です。

トピック

- 新しいレシピバージョンの発行
- レシピ構造の定義

新しいレシピバージョンの発行

インタラクティブな DataBrew プロジェクトセッションでレシピの新しいバージョンを発行します。

新しいレシピバージョンを発行するには

1. レシピペインで、発行を選択します。
2. このバージョンのレシピの説明を入力し、発行を選択します。

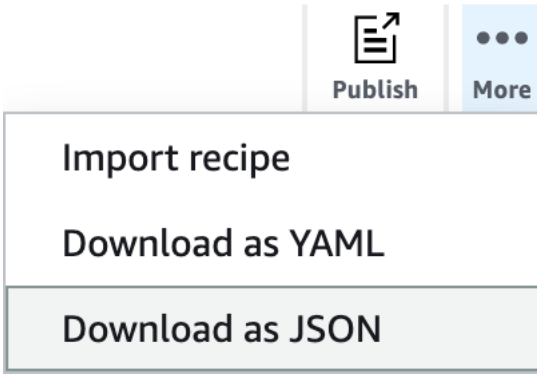
ナビゲーションペインから PROJECTS を選択すると、公開されたすべてのレシピとそのバージョンを表示できます。

レシピ構造の定義

DataBrew コンソールを使用してプロジェクトを初めて作成するときは、そのプロジェクトに関連付けるレシピを定義します。既存のレシピがない場合は、コンソールによってレシピが作成されます。

コンソールでプロジェクトを操作するときは、変換ツールバーを使用してデータセットのサンプルデータにアクションを適用します。コンソールには、レシピの作成を続行すると、レシピステップとそれらのステップの順序が表示されます。ステップに満足するまでレシピを繰り返して絞り込むことができます。

では[の開始方法AWS Glue DataBrew](#)、有名なチェスゲームのデータセットを変換するレシピを作成します。レシピステップのコピーをダウンロードするには、次のスクリーンショットに示すように、JSON としてダウンロードまたは YAML としてダウンロードを選択します。



ダウンロードした JSON ファイルには、レシピに追加した変換に対応するレシピアクションが含まれています。

新しいレシピにはステップがありません。次に示すように、新しいレシピを空の JSON リストとして表すことができます。

```
[ ]
```

のこのようなファイルの例を次に示しますchess-project-recipe。JSON リストには、レシピステップを記述する複数のオブジェクトが含まれています。JSON リスト内の各オブジェクトは中括弧 () で囲まれます{ }。JSON 行はカンマで区切られます。

```
[
    {
        "Action": {
            "Operation": "REMOVE_VALUES",
            "Parameters": {
                "sourceColumn": "black_rating"
            }
        },
        "ConditionExpressions": [
            {
                "Condition": "LESS_THAN",
                "Value": "1800",
                "TargetColumn": "black_rating"
            }
        ]
    },
    {
        "Action": {
            "Operation": "REMOVE_VALUES",
            "Parameters": {
                "sourceColumn": "white_rating"
            }
        },
        "ConditionExpressions": [
            {
                "Condition": "LESS_THAN",
                "Value": "1800",
                "TargetColumn": "white_rating"
            }
        ]
    },
    {
        "Action": {
```

```
            "Operation": "GROUP_BY",
            "Parameters": {
                "groupByAggFunctionOptions": "[{\"sourceColumnName\":\"winner\",
\"targetColumnName\":\"winner_count\",\"targetColumnDataType\":\"int\",\"functionName
\":\"COUNT\"}]",
                "sourceColumns": "[\"winner\",\"victory_status\"]",
                "useNewDataFrame": "true"
            }
        }
    },
    {
        "Action": {
            "Operation": "REMOVE_VALUES",
            "Parameters": {
                "sourceColumn": "winner"
            }
        },
        "ConditionExpressions": [
            {
                "Condition": "IS",
                "Value": "[\"draw\"]",
                "TargetColumn": "winner"
            }
        ]
    },
    {
        "Action": {
            "Operation": "REPLACE_TEXT",
            "Parameters": {
                "pattern": "mate",
                "sourceColumn": "victory_status",
                "value": "checkmate"
            }
        }
    },
    {
        "Action": {
            "Operation": "REPLACE_TEXT",
            "Parameters": {
                "pattern": "resign",
                "sourceColumn": "victory_status",
                "value": "other player resigned"
            }
        }
```

```
    },
    {
        "Action": {
            "Operation": "REPLACE_TEXT",
            "Parameters": {
                "pattern": "outoftime",
                "sourceColumn": "victory_status",
                "value": "ran out of time"
            }
        }
    }
]
```

次に示すように、新しいアクションに新しい行を追加するだけで、各アクションが個々の行であることがわかりやすくなります。

```
[
 { "Action": { "Operation": "REMOVE_VALUES", "Parameters": { "sourceColumn":
 "black_rating" } }, "ConditionExpressions": [ { "Condition": "LESS_THAN", "Value":
 "1800", "TargetColumn": "black_rating" } ] },
 { "Action": { "Operation": "REMOVE_VALUES", "Parameters": { "sourceColumn":
 "white_rating" } }, "ConditionExpressions": [ { "Condition": "LESS_THAN", "Value":
 "1800", "TargetColumn": "white_rating" } ] },
 { "Action": { "Operation": "GROUP_BY", "Parameters": { "groupByAggFunctionOptions":
 "[{\"sourceColumnName\":\"winner\",\"targetColumnName\":\"winner_count\",
\"targetColumnDataType\":\"int\",\"functionName\":\"COUNT\"}]", "sourceColumns":
 "[\"winner\",\"victory_status\"]", "useNewDataFrame": "true" } } },
 { "Action": { "Operation": "REMOVE_VALUES", "Parameters": { "sourceColumn":
 "winner" } }, "ConditionExpressions": [ { "Condition": "IS", "Value": "[\"draw\"]",
 "TargetColumn": "winner" } ] },
 { "Action": { "Operation": "REPLACE_TEXT", "Parameters": { "pattern": "mate",
 "sourceColumn": "victory_status", "value": "checkmate" } } },
 { "Action": { "Operation": "REPLACE_TEXT", "Parameters": { "pattern": "resign",
 "sourceColumn": "victory_status", "value": "other player resigned" } } },
 { "Action": { "Operation": "REPLACE_TEXT", "Parameters": { "pattern": "outoftime",
 "sourceColumn": "victory_status", "value": "ran out of time" } } }
]
```

アクションは、 ファイルと同じ順序で順番に実行されます。

- REMOVE_VALUES – プレイヤーの評価が 1,800 未満のすべてのゲームを除外するには、クラス A チェスプレイヤーに必要な最小評価です。このアクションには 2 つの出現があります。1 つは少な

くともクラス A のプレイヤーではない黒側のプレイヤーを削除し、もう 1 つはこのレベルでない白側のプレイヤーを削除します。

- GROUP_BY – データを要約します。この場合、GROUP_BY は winner (black および) の値に基づいて行をグループにソートしますwhite。これらの各グループはさらに分類され、victory_status (mate、、resign、outoftime) の値に基づいて行がサブグループにソートされますdraw。最後に、各サブグループの出現回数がカウントされます。結果の概要は、元のデータサンプルを置き換えます。
- REMOVE_VALUES – で終了したゲームの結果を削除しますdraw。
- REPLACE_TEXT – の値を変更しますvictory_status。このアクションには、、mate、 にそれぞれ 1 つずつresign、3 つの出現がありますoufoftime。

インタラクティブな DataBrew プロジェクトセッションでは、それぞれがデータサンプルに適用するデータ変換RecipeActionに対応します。

DataBrew は 200 を超えるレシピアクションを提供します。詳細については、「レシピステップと関数リファレンス」を参照してください。

条件の使用

条件を使用して、レシピアクションの範囲を絞り込むことができます。条件は、特定の列値に基づいて不要な行を削除するなど、データをフィルタリングする変換に使用されます。

のレシピアクションを詳しく見てみましょうchess-project-recipe。

```
{
  "Action": {
    "Operation": "REMOVE_VALUES",
    "Parameters": {
      "sourceColumn": "black_rating"
    }
  },
  "ConditionExpressions": [
    {
      "Condition": "LESS_THAN",
      "Value": "1800",
      "TargetColumn": "black_rating"
    }
  ]
}
```

この変換は、 black_rating列の値を読み取ります。ConditionExpressions リストはフィルタリング条件を決定します。black_rating値が 1,800 未満の行はデータセットから削除されます。

レシピのフォローアップ変換は、 に対して同じことを行いますwhite_rating。このようにして、データは各プレイヤー (黒または白) がクラス A 以上と評価されるゲームに限定されます。

文字データの列に適用される条件の別の例を次に示します。

```
{
  "Action": {
    "Operation": "REMOVE_VALUES",
    "Parameters": {
      "sourceColumn": "winner"
    }
  },
  "ConditionExpressions": [
    {
      "Condition": "IS",
      "Value": "[\"draw\"]",
      "TargetColumn": "winner"
    }
  ]
}
```

この変換は列内の値を読み取りwinner、値を探drawしてそれらの行を削除します。このように、データは、明確な勝者がいたゲームのみに制限されます。

DataBrew は次の条件をサポートしています。

- IS – 列の値は、 条件で指定された値と同じです。
- IS_NOT – 列の値は、 条件で指定された値と同じではありません。
- IS_BETWEEN – 列の値は GREATER_THAN_EQUALパラメータと LESS_THAN_EQUALパラメータの間です。
- CONTAINS – 列の文字列値には、 条件で指定された値が含まれます。
- NOT_CONTAINS – 列の値に、 条件で指定された文字列が含まれていません。
- STARTS_WITH – 列の値は、 条件で指定された文字列で始まります。
- NOT_STARTS_WITH – 列の値は、 条件で指定された文字列で始まるわけではありません。
- ENDS_WITH – 列の値は、 条件で指定された文字列で終わります。
- NOT_ENDS_WITH – 列の値は、 条件で指定された文字列で終わりません。

- LESS_THAN – 列の値は、 条件で指定された値より小さくなります。
- LESS_THAN_EQUAL – 列の値は、 条件で指定された値以下です。
- GREATER_THAN – 列の値が、 条件で指定された値より大きい。
- GREATER_THAN_EQUAL – 列の値は、 条件で指定された値以上です。
- IS_INVALID – 列の値のデータ型が正しくありません。
- IS_MISSING – 列に値はありません。

AWS Glue DataBrewジョブの作成、実行、スケジューリング

AWS Glue DataBrewには、次の 2 つの目的を果たすジョブサブシステムがあります。

1. DataBrew データセットにデータ変換レシピを適用します。これは DataBrew レシピジョブで行います。
2. データセットを分析して、データの包括的なプロファイルを作成します。これは DataBrew プロファイルジョブで行います。

トピック

AWS Glue DataBrewレシピジョブの作成と操作

DataBrew レシピジョブを使用して、DataBrew データセット内のデータをクリーンアップおよび正規化し、選択した出力場所に結果を書き込みます。レシピジョブを実行しても、データセットや基盤となるソースデータには影響しません。ジョブを実行すると、読み取り専用の方法でソースデータに接続します。ジョブ出力は、Amazon S3、、AWS Glue Data Catalogまたはサポートされている JDBC データベースで定義した出力場所に書き込まれます。

DataBrew レシピジョブを作成するには、次の手順に従います。

レシピジョブを作成するには

1. にサインインAWS マネジメントコンソールし、https://console.aws.amazon.com/databrew/ で DataBrew コンソールを開きます。
2. ナビゲーションペインから JOBS を選択し、レシピジョブタブを選択し、ジョブの作成を選択します。
3. ジョブの名前を入力し、レシピジョブの作成を選択します。
4. ジョブ入力には、作成するジョブの詳細を入力します。処理するデータセットの名前と使用するレシピです。

レシピジョブは DataBrew レシピを使用してデータセットを変換します。レシピを使用するには、必ず最初に公開してください。

5. ジョブの出力設定を構成します。

   ジョブ出力の送信先を指定します。出力先用に DataBrew 接続を設定していない場合は、「」の説明に従って、最初に DATASETS タブで設定しますデータソースと出力でサポートされている接続。次のいずれかの出力先を選択します。

   - AWS Glue Data Catalogサポートの有無にかかわらず Amazon S3
   - Amazon Redshift、AWS Glue Data Catalogサポートの有無にかかわらず
   - JDBC
   - Snowflake テーブル
   - AWS Glue Data Catalogサポートされる Amazon RDS データベーステーブル。Amazon RDS データベーステーブルは、次のデータベースエンジンをサポートしています。
     - Amazon Aurora
     - MySQL
     - Oracle
     - [PostgreSQL]
     - Microsoft SQL Server
   - AWS Glue Data Catalogサポート付きの Amazon S3。

   に基づくAWS Glue Data Catalog出力の場合AWS Lake Formation、DataBrew は既存のファイルの置き換えのみをサポートします。この方法では、ファイルは置き換えられ、データアクセスロールの既存の Lake Formation アクセス許可がそのまま維持されます。また、DataBrew はテーブルから Amazon S3 の場所を優先しますAWS Glue Data Catalog。したがって、レシピジョブの作成時に Amazon S3 の場所を上書きすることはできません。

   ジョブ出力の Amazon S3 の場所は、Data Catalog テーブルの Amazon S3 の場所とは異なる場合があります。このような場合、DataBrew はカタログテーブルから Amazon S3 の場所を使用してジョブ定義を自動的に更新します。これは、既存のジョブを更新または開始するときに行われます。

6. Amazon S3 出力先のみ、次の選択肢があります。

a. Amazon S3 で使用可能なデータ出力形式、オプションの圧縮形式、オプションのカスタム区切り文字のいずれかを選択します。出力ファイルでサポートされている区切り文字は、カンマ、コロン、セミコロン、パイプ、タブ、キャレット、バックスラッシュ、スペースの入力と同じです。フォーマットの詳細については、次の表を参照してください。

Format	ファイル拡張子 (非圧縮)	ファイル拡張子 (圧縮)
カンマ区切り値	`.csv`	`.csv.snappy` , `.csv.gz`, `.csv.lz4`, `csv.bz2`, `.csv.deflate` , `csv.br`
タブ区切り値	`.csv`	`.tsv.snappy` , `.tsv.gz`, `.tsv.lz4`, `tsv.bz2`, `.tsv.deflate` , `tsv.br`
Apache Parquet	`.parquet`	`.parquet.snappy` , `.parquet.gz` , `.parquet.lz4` , `.parquet.lzo` , `.parquet.br`
AWS Glue Parquet	サポートされていません	`.glue.parquet.snappy`
Apache Avro	`.avro`	`.avro.snappy` , `.avro.gz`, `.avro.lz4` , `.avro.bz2` , `.avro.deflate` , `.avro.br`
Apache ORC	`.orc`	`.orc.snappy` , `.orc.lzo`, `.orc.zlib`
XML	`.xml`	`.xml.snappy` , `.xml.gz`, `.xml.lz4`, `.xml.bz2`, `.xml.deflate` , `.xml.br`

Format	ファイル拡張子 (非圧縮)	ファイル拡張子 (圧縮)
JSON (JSON Lines 形式のみ)	`.json`	`.json.snappy` , `.json.gz`, `.json.lz4` , `json.bz2`, `.json.deflate` , `.json.br`
Tableau Hyper	サポートされていません	該当しない

b.
1 つのファイルを出力するか、複数のファイルを出力するかを選択します。Amazon S3 でのファイル出力には 3 つのオプションがあります。

- ファイルの自動生成 (推奨) – DataBrew が出力ファイルの最適な数を決定します。
- 単一ファイルの出力 – 単一の出力ファイルが生成されます。このオプションでは、後処理が必要なため、ジョブの実行時間が長くなる場合があります。
- 複数ファイル出力 – ジョブ出力のファイル数を指定しますか。有効な値は 2 ~ 999 です。列パーティショニングが使用されている場合、または出力の行数が指定したファイル数より少ない場合、指定した数よりも少ないファイルが出力される可能性があります。

c.
(オプション) レシピジョブ出力の列パーティショニングを選択します。

列パーティショニングは、レシピジョブの出力を複数のファイルに分割する別の方法を提供します。列パーティショニングは、新規または既存の Amazon S3 出力、または新しい Data Catalog Amazon S3 出力で使用できます。既存の Data Catalog Amazon S3 テーブルでは使用できません。出力ファイルは、指定した列名の値に基づいています。指定した列名が一意である場合、結果の Amazon S3 フォルダパスは列名の順序に基づきます。

列パーティショニングの例については、列パーティショニングの例次の「」を参照してください。

7. (オプション) ジョブ出力の暗号化を有効にするを選択して、DataBrew が出力場所に書き込むジョブ出力を暗号化し、暗号化方法を選択します。

- SSE-S3 暗号化を使用する – 出力は、Amazon S3 が管理する暗号化キーによるサーバー側の暗号化を使用して暗号化されます。
- Use AWS Key Management Service(AWS KMS) – 出力は を使用して暗号化されますAWS KMS。このオプションを使用するには、使用するAWS KMSキーの Amazon リソースネーム

(ARN) を選択します。AWS KMSキーがない場合は、キーの作成 を選択してAWS KMS作成できます。

8. アクセス許可には、DataBrew が出力場所に書き込むことを許可するAWS Identity and Access Management(IAM) ロールを選択します。アカウントAWSが所有する口ケーションでは、`AwsGlueDataBrewDataAccessRole`サービスマネージドロールを選択できます。これにより、DataBrew は所有しているAWSリソースにアクセスできます。
9. 高度なジョブ設定ペインで、ジョブの実行方法のその他のオプションを選択できます。
   - 最大ユニット数 – DataBrew は複数のコンピューティングノードを使用してジョブを処理し、並行して実行されます。ノードのデフォルト数は 5 です。ノードの最大数は 149 です。
   - ジョブのタイムアウト – ジョブの実行にここで設定した分数を超えると、タイムアウトエラーで失敗します。デフォルト値は 2,880 分、つまり 48 時間です。
   - 再試行回数 – 実行中にジョブが失敗した場合、DataBrew は再度実行を試みることができます。デフォルトでは、ジョブは再試行されません。
   - ジョブの Amazon CloudWatch Logs を有効にする – DataBrew が CloudWatch Logs に診断情報を公開できるようにします。これらのログは、トラブルシューティングの目的や、ジョブの処理方法の詳細に役立ちます。
10. スケジュールジョブの場合、DataBrew ジョブスケジュールを適用して、ジョブが特定の時間または定期的に実行されるようにできます。詳細については、「スケジュールによるジョブ実行の自動化」を参照してください。
11. 設定が目的どおりになったら、ジョブの作成を選択します。または、ジョブをすぐに実行する場合は、ジョブの作成と実行を選択します。

ジョブの実行中にジョブのステータスを確認することで、ジョブの進行状況をモニタリングできます。ジョブの実行が完了すると、ステータスは成功に変わります。ジョブ出力が、選択した出力場所で利用可能になりました。

DataBrew はジョブ定義を保存するため、後で同じジョブを実行できます。ジョブを再実行するには、ナビゲーションペインからジョブを選択します。使用するジョブを選択し、ジョブの実行を選択します。

列パーティショニングの例

列のパーティショニングの例として、3 つの列を指定し、各行に 2 つの可能な値のいずれかが含まれているとします。`Dept` 列には `Admin`または の値を指定できますEng。`Staff-type` 列には `Part-`

timeまたは の値を指定できますFull-time。Location 列には Office1または の値を指定できますOffice2。ジョブ出力の Amazon S3 バケットは次のようになります。

```
s3://bucket/output-folder/Dept=Admin/Staff-type=Part-time/Area=Office1/
jobId_timestamp_part0001.csv
s3://bucket/output-folder/Dept=Admin/Staff-type=Part-time/Location=Office2/
jobId_timestamp_part0002.csv
s3://bucket/output-folder/Dept=Admin/Staff-type=Full-time/Location=Office1/
jobId_timestamp_part0003.csv
s3://bucket/output-folder/Dept=Admin/Staff-type=Full-time/Location=Office2/
jobId_timestamp_part0004.csv
s3://bucket/output-folder/Dept=Eng/Staff-type=Part-time/Location=Office1/
jobId_timestamp_part0005.csv
s3://bucket/output-folder/Dept=Eng/Staff-type=Part-time/Location=Office2/
jobId_timestamp_part0006.csv
s3://bucket/output-folder/Dept=Eng/Staff-type=Full-time/Location=Office1/
jobId_timestamp_part0007.csv
s3://bucket/output-folder/Dept=Eng/Staff-type=Full-time/Location=Office2/
jobId_timestamp_part0008.csv
```

スケジュールによるジョブ実行の自動化

DataBrew ジョブはいつでも再実行でき、スケジュールに従って DataBrew ジョブの実行を自動化することもできます。

DataBrew ジョブを再実行するには

1. にサインインAWS マネジメントコンソールし、https://console.aws.amazon.com/databrew/ で DataBrew コンソールを開きます。
2. ナビゲーションペインで、ジョブを選択します。実行するジョブを選択し、ジョブの実行を選択します。

DataBrew ジョブを特定の時間、または定期的に実行するには、DataBrew ジョブスケジュールを作成します。その後、スケジュールに従って を実行するようにジョブを設定できます。

DataBrew ジョブスケジュールを作成するには

1. DataBrew コンソールのナビゲーションペインで、ジョブを選択します。スケジュールタブを選択し、スケジュールの追加を選択します。
2. スケジュールの名前を入力し、実行頻度の値を選択します。

- 繰り返し – ジョブを実行する頻度を選択します (12 時間ごとなど) 。次に、ジョブを実行する日を選択します。必要に応じて、ジョブが実行される時刻を入力できます。
- 特定の時刻 – ジョブを実行する時刻を入力します。次に、ジョブを実行する日を選択します。
- Enter CRON – 有効な cron 式を入力してジョブスケジュールを定義します。詳細については、「レシピジョブの cron 式の使用」を参照してください。

3. すべての設定が正しいことを確認したら、[保存] を選択します。

ジョブをスケジュールに関連付けるには

1. ナビゲーションペインで、ジョブを選択します。
2. 操作するジョブを選択し、アクションで編集を選択します。
3. スケジュールジョブペインで、スケジュールの関連付けを選択します。使用するスケジュールの名前を選択します。
4. すべての設定が正しいことを確認したら、[保存] を選択します。

レシピジョブの cron 式の使用

cron 式には 6 つの必須フィールドがあり、それらは空白で区切られます。構文は次のとおりです。

```
Minutes Hours Day-of-month Month Day-of-week Year
```

前述の構文では、指定されたフィールドに次の値とワイルドカードが使用されます。

フィールド	値	ワイルドカード
分	0 ~ 59	, - * /
時間	0 ~ 23	, - * /
日	1 ~ 31	, - * ? / L W
月	1 ~ 12 または JAN ~ DEC	, - * /
曜日	1 ~ 7 または SUN ~ SAT	, - * ? / L

フィールド	値	ワイルドカード
年	1970 ~ 2199	, - * /

これらのワイルドカードは次のように使用します。

- , (カンマ) のワイルドカードには、追加の値が含まれます。Month フィールドには、1 月、2 月、3 月JAN,FEB,MARが含まれます。
- - (ダッシュ) ワイルドカードは範囲を指定します。Day フィールドの 1 ~ 15 には、指定した月の 1 ~ 15 日が含まれます。
- [*] (アスタリスク) のワイルドカードには、フィールドのすべての値が含まれます。Hours フィールドでは、* には 1 時間ごとが含まれます。
- [/] (スラッシュ) ワイルドカードで増分を指定します。Minutes フィールドで、 と入力**1/10**して、1 時間の最初の 1 分 (11 分、21 分、31 分など) から 10 分ごとに指定できます。
- [?] (疑問符) のワイルドカードは、任意を意味します。たとえば、 Day-of-monthフィールドに 7 と入力するとします。7 日が何曜日か気にならなかった場合は、「」と入力できます。 Day-of-week フィールドの 。
- Day-of-month または Day-of-weekフィールドの L ワイルドカードは、月または週の最終日を指定します。
- Day-of-month フィールドの、ワイルドカード W は、平日を指定します。Day-of-month フィールドで、3W は月の 3 番目の平日に最も近い日を指定します。

これらのフィールドと値には、次の制限があります。

- Cron 式の Day-of-month フィールドと Day-of-week フィールドを同時に指定することはできません。一方のフィールドに値を指定する場合、もう一方のフィールドで [?] (疑問符) を使用する必要があります。
- 5 分を超える速度につながる Cron 式はサポートされていません。

スケジュールを作成するときは、以下のサンプルの cron 文字列を使用できます。

分	時間	日	月	曜日	年	意味
0	10	*	*	?	*	毎日午前 10:00 (UTC) に実行
15	12	*	*	?	*	毎日午後 12:15 (UTC) に実行
0	18	?	*	MON-FRI	*	毎週月曜日から金曜日まで午後 6:00 (UTC) に実行
0	8	1	*	?	*	毎月 1 日午前 8 時 (UTC) に実行
0/15	*	*	*	?	*	15 分ごとに実行
0/10	*	?	*	MON-FRI	*	月曜日から金曜日まで 10 分ごとに実行
0/5	8 ~ 17	?	*	MON-FRI	*	月曜日から金曜日まで午前 8:00 から午後 5:55 (UTC) まで 5 分ごとに実行

たとえば、次の cron 式を使用して、毎日 12:15 UTC にジョブを実行できます。

```
15 12 * * ? *
```

ジョブとジョブスケジュールの削除

ジョブまたはジョブスケジュールが不要になった場合は、削除できます。

ジョブを削除するには

1. ナビゲーションペインで、ジョブを選択します。
2. 削除するジョブを選択し、アクションで削除を選択します。

ジョブスケジュールを削除するには

1. ナビゲーションペインで、ジョブを選択し、スケジュールタブを選択します。
2. 削除するスケジュールを選択し、アクションで削除を選択します。

AWS Glue DataBrewプロファイルジョブの作成と操作

プロファイルジョブはデータセットに対して一連の評価を実行し、結果を Amazon S3 に出力します。データプロファイリングが収集する情報は、データセットを理解し、レシピジョブで実行するデータ準備ステップの種類を決定するのに役立ちます。

プロファイルジョブを実行する最も簡単な方法は、デフォルトの DataBrew 設定を使用することです。プロファイルジョブを実行する前に、必要な情報のみが返されるように設定できます。

DataBrew プロファイルジョブを作成するには、次の手順に従います。

プロファイルジョブを作成するには

1. にサインインAWS マネジメントコンソールし、https://console.aws.amazon.com/databrew/ で DataBrew コンソールを開きます。
2. ナビゲーションペインから JOBS を選択し、プロファイルジョブタブを選択し、ジョブの作成を選択します。
3. ジョブの名前を入力し、プロファイルジョブの作成を選択します。
4. ジョブ入力には、プロファイリングするデータセットの名前を指定します。

5. (オプション) データプロファイル設定ペインで以下を設定します。

   - データセットレベルの設定 – データセット内のすべての列のプロファイルジョブの詳細を設定します。

     必要に応じて、データセット内の重複行を検出してカウントする機能をオンにできます。相関行列を有効にし、列を選択して、複数の列の値の関連性を確認することもできます。データセットレベルで設定できる統計の詳細については、「」を参照してくださいデータセットレベルで設定可能な統計。DataBrew コンソール、または DataBrew API またはAWS SDKs を使用して統計を設定できます。

   - 列レベルの設定 – デフォルトのプロファイル設定を使用して、プロファイルジョブに含める列を選択できます。設定の上書きの追加を使用して、収集される統計の数を制限する列を選択するか、特定の統計のデフォルト設定を上書きします。列レベルで設定できる統計の詳細については、「」を参照してください列レベルで設定可能な統計。DataBrew コンソール、または DataBrew API またはAWS SDKs を使用して統計を設定できます。

     指定した設定オーバーライドが、プロファイルジョブに含めた列に適用されることを確認してください。列に対して設定した異なるオーバーライド間に競合がある場合、最後の競合するオーバーライドが優先されます。

6. (オプション) データ品質ルールを作成し、このデータセットに関連付けられた追加のルールセットを適用するか、既に適用されているルールセットを削除できます。データ品質検証の詳細については、「」を参照してくださいでのデータ品質の検証AWS Glue DataBrew。

7. 高度なジョブ設定ペインで、ジョブの実行方法のその他のオプションを選択できます。

   - 最大ユニット数 – DataBrew は複数のコンピューティングノードを使用してジョブを処理し、並行して実行されます。ノードのデフォルト数は 5 です。ノードの最大数は 149 です。
   - ジョブのタイムアウト – ジョブの実行にここで設定した分数を超えると、タイムアウトエラーで失敗します。デフォルト値は 2,880 分、つまり 48 時間です。
   - 再試行回数 – 実行中にジョブが失敗した場合、DataBrew は再度実行を試みることができます。デフォルトでは、ジョブは再試行されません。
   - ジョブの Amazon CloudWatch Logs を有効にする – DataBrew が CloudWatch Logs に診断情報を公開できるようにします。これらのログは、トラブルシューティングの目的や、ジョブの処理方法の詳細に役立ちます。

8. 関連付けられたスケジュールでは、DataBrew ジョブスケジュールを適用して、ジョブを特定の時間または定期的に実行できます。詳細については、「スケジュールによるジョブ実行の自動化」を参照してください。

9. 設定が目的どおりになったら、ジョブの作成を選択します。または、ジョブをすぐに実行する場合は、ジョブの作成と実行を選択します。

でプロファイルジョブ設定をプログラムで構築するAWS Glue DataBrew

このセクションでは、プログラムで使用できるプロファイルジョブのステップと関数について説明します。これらは、AWS Command Line Interface(AWS CLI) から、またはいずれかのAWS SDKs を使用して使用できます。

プロファイルジョブでは、DataBrew がデータセットを評価する方法を制御するように設定をカスタマイズできます。設定をデータセットに適用するか、特定の列に適用できます。プロファイルジョブの作成時に設定を構築し、いつでも更新できます。

プロファイル設定構造には 4 つの部分があります。

- ProfileColumns セクション
- DatasetStatisticsConfiguration セクション
- ColumnStatisticsConfigurations セクション
- PII を設定するための EntityDetectorConfiguration セクション

次に例を示します。

```
{
    "ProfileColumns": [
        {
            "Name": "example"
        },
        {
            "Regex": "example.*"
        }
    ],
    "DatasetStatisticsConfiguration": {
        "IncludedStatistics": [
            "CORRELATION"
        ],
        "Overrides": [
            {
                "Statistic": "CORRELATION",
                "Parameters": {
```

```
                    "columnSelectors": "[{\"name\":\"example\"}, {\"regex\":\"example.*
\"}]"
                }
            }
        ]
    },
    "ColumnStatisticsConfigurations": [
        {
            "Selectors": [
                {
                    "Name": "example"
                }
            ],
            "Statistics": {
                "IncludedStatistics": [
                    "CORRELATION",
                    "DUPLICATE_ROWS_COUNT"
                ],
                "Overrides": [
                    {
                        "Statistic": "VALUE_DISTRIBUTION",
                        "Parameters": {
                            "binNumber": "10"
                        }
                    }
                ]
            }
        }
    ]
}
```

ProfileColumns セクション

構造の ProfileColumnsセクションで、プロファイルジョブで評価するデータセットの列を設定します。 ProfileColumnsは列セレクタ (Selectors) のリストです。列セレクターでは、列名または正規表現を指定できます。以下に例を示します。

```
"ProfileColumns": [{"Name": "example"}, {"Regex": "example.*"}]
```

ProfileColumns を指定すると、 の名前または正規表現に一致する名前の列のみがプロファイルジョブProfileColumnsに含まれます。プロファイルジョブが選択した列のデータ型をサポートしていない場合、DataBrew はジョブの実行中に選択した列をスキップします。

ProfileColumns が未定義の場合、プロファイルジョブはサポートされているすべての列を評価します。サポートされている列は、サポートされているデータ型のデータを含む列です: ByteType、ShortType、、IntegerType、LongType、String、、または FloatType DoubleTypeBoolean。

DatasetStatisticsConfiguration セクション

構造の DatasetStatisticsConfigurationセクションで、列間評価の設定を構築できます。設定には IncludedStatisticsと が含まれますOverrides。以下に例を示します。

```
"DatasetStatisticsConfiguration": {
    "IncludedStatistics": ["CORRELATION"],
    "Overrides": [
        {
            "Statistic": "CORRELATION",
            "Parameters": {
                "columnSelectors": "[{\"name\":\"example\"}, {\"regex\":\"example.*
\"}]"
            }
        }
    ]
}
```

に評価名を追加することで、必要な評価を選択できますIncludedStatistics。以下に例を示します。

```
"IncludedStatistics": ["CORRELATION", "DUPLICATE_ROWS_COUNT"]
```

を指定するとIncludedStatistics、リスト内の評価のみがプロファイルジョブに含まれます。IncludedStatistics が未定義の場合、プロファイルジョブはサポートされているすべての評価をデフォルト設定で実行します。に NONE を追加することで、すべての評価を除外できますIncludedStatistics。以下に例を示します。

```
"IncludedStatistics": ["NONE"]
```

データセットレベルで設定可能な統計

構造の DatasetStatisticsConfigurationセクションで、プロファイルジョブは、次の表に示す評価をサポートします。

統計名	説明	サポートされているデータ型	デフォルトのステータス	プロファイル結果の属性	プロファイル結果のタイプ
DUPLICATE_ROWS_COUNT	データセット内の重複行の数	すべて	有効化	duplicateRowsCoun	Int
相関関係	2 つの列間のピアソン相関係数	数値	有効化	相関関係(選択した各列)	オブジェクト

ではIncludedStatistics、オーバーライドを追加することで、各評価のデフォルト設定を上書きできます。各オーバーライドには、特定の評価の名前とパラメータマップが含まれます。

ではDatasetStatisticsConfiguration、プロファイルジョブはCORRELATIONオーバーライドをサポートします。このオーバーライドは、選択した列のリストから 2 つの列間のピアソン相関係数を計算します。デフォルト設定では、最初の 10 個の数値列が選択されています。列の数または列セレクタのリストのいずれかを指定して、デフォルト設定を上書きできます。

CORRELATION は、次のパラメータを受け取ります。

- columnNumber – 数値列の数。プロファイルジョブは、データセットから最初の n 列を選択します。この値は 1 より大きくする必要があります。を使用して"ALL"、すべての数値列を選択します。
- columnSelectors: – 列セレクタのリスト。各セレクタには、列名または正規表現を含めることができます。

以下に例を示します。

```
{
```

```
    "Statistic": "CORRELATION",
    "Parameters": {
        "columnSelectors": "[{\"name\":\"example\"}, {\"regex\":\"example.*\"}]"
    }
}
```

ColumnStatisticsConfigurations セクション

構造の ColumnStatisticsConfigurationsセクションでは、特定の列の設定を構築できます。ColumnStatisticsConfigurationsはColumnStatisticsConfiguration設定のリストです。にはColumnStatisticsConfiguration、Selectors、列セレクタのリスト、および統計の設定Statistics用の があります。以下に例を示します。

```
{
    "Selectors": [{"Name": "example"}
    ],
    "Statistics": {
       "IncludedStatistics": ["CORRELATION", "DUPLICATE_ROWS_COUNT"]
        "Overrides": [
            {
                "Statistic": "VALUE_DISTRIBUTION",
                "Parameters": {
                    "binNumber": "10"
                }
            }
        ]
    }
}
```

Selectors は列セレクタのリストです。と同様にProfileColumns、各列セレクタで列名または正規表現を指定できます。を指定するとSelectors、 の任意の列セレクタに一致する列に列設定が適用されますSelectors。それ以外の場合、設定はサポートされているすべての列に適用されます。

ではStatistics、選択した列の設定を上書きできます。と同様にDatasetStatisticsConfiguration、 Statisticsには IncludedStatisticsと がありますOverrides。

必要な評価を選択するには、 に評価名を追加しますIncludedStatistics。

```
"IncludedStatistics": ["CORRELATION", "DUPLICATE_ROWS_COUNT"]
```

を指定するとIncludedStatistics、リスト内の評価のみがプロファイルジョブに含まれます。それ以外の場合、プロファイルジョブはサポートされているすべての評価をデフォルト設定で実行します。

を NONEに追加することで、すべての評価を除外できますIncludedStatistics。

```
"IncludedStatistics": ["NONE"]
```

場合によっては、同じ列に適用IncludedStatisticsできる異なる設定ColumnStatisticsConfigurationsが に複数存在することがあります。このような場合、プロファイルジョブは で最後の設定を選択しColumnStatisticsConfigurations、選択した列IncludedStatisticsに適用します。新しい設定は、古い設定を上書きします。

列レベルで設定可能な統計

ではColumnStatisticsConfigurations、プロファイルジョブは、次の表に示す評価をサポートします。

このテーブルnumberでサポートされている のデータ型は、属性のデータ型がByteType、ShortType、、、IntegerTypeLongTypeFloatType、または のいずれかであることを意味しますDoubleType。

統計名	説明	サポートされているデータ型	デフォルトのステータス	プロファイル結果の属性	プロファイル結果のタイプ
–	列の名前。	すべて	–	name	string
–	列のデータ型。	すべて	–	型	string
DISTINCT_VALUES_COUNT	個別の値の数。個別の値は、少なくとも 1 回表示される値です。	number/boolean/string	有効	distinctValuesCount	Int
エントロピー	Entropy (情報理論) 。	number/boolean/string	有効	エントロピー	Double

統計名	説明	サポートされているデータ型	デフォルトのステータス	プロファイル結果の属性	プロファイル結果のタイプ
INTER_QUARTILE_RANGE	数値の 25 パーセントから 75 パーセントの範囲。	数値	有効	interquartileRange	Double
尖度	列の尖度。	数値	有効	尖度	Double
MAX	列の最大値。	number/string の長さ	有効	max	Int/Double
MAXIMUM_VALUES	列内の最大値とその数のリスト。	数値	有効	maximumVlues	リスト
MEAN	列の値の平均値。	number/string の長さ	有効	mean	Double
MEDIAN	列の値の中央値。	number/string の長さ	有効	median	Double
MEDIAN_ABSOLUTE_DEVIATION	各データポイント間の絶対差の中央値と数値列の中央値。	数値	有効	medianAbsoluteDeviation	Double
MIN	列の最小値。	number/string の長さ	有効	min	Int/Double
最小値	列内の最小値とその数のリスト。	数値	有効	minimumVlues	リスト
MISSING_VALUES_COUNT	列内の欠損値の数。Null および空の文字列は欠落していると見なされます。	すべて	有効	missingValuesCount	Int

統計名	説明	サポートされているデータ型	デフォルトのステータス	プロファイル結果の属性	プロファイル結果のタイプ
MODE	列で最も頻繁に発生する値。複数の値が頻繁に表示される場合、モードはそれらの値の 1 つです。	number/string の長さ	有効	モード	Int/Double
MOST_COMMON_VALUES	列内の最も一般的な値のリスト。	number/boolean/string	有効	mostCommnValues	リスト
OUTLIER_DETECTION	Z_score アルゴリズムによって列内の外れ値を検出します。外れ値の数をカウントし、検出された外れ値からサンプルのリストを抽出します。	数値/文字列の長さ	有効	zScoreOutliersCount、zScoreOutliersSample	内部/リスト
パーセンタイル	数値列のパーセンタイル値 (5%、25%、75%、95%)。	数値	有効	percentile5、percertile25、percentile75、percentile95	Double
RANGE	列の値の範囲。	数値	有効	範囲	Int/Double
歪度	列の値の歪み。	数値	有効	歪度	Double
STANDARD_DEVIATION	列の値のバイアスのないサンプル標準偏差。	数値/文字列の長さ	有効	standardDeviation	Double
SUM	列の値の合計。	数値	有効	sum	Int/Double

統計名	説明	サポートされているデータ型	デフォルトのステータス	プロファイル結果の属性	プロファイル結果のタイプ
UNIQUE_VALUES_COUNT	一意の値の数。一意の値は、値が 1 回だけ表示されることを意味します。	number/boolean/string	有効	uniqueValuesCount	Int
VALUE_DISTRIBUTION	範囲による列の値の分布の測定値。	数値/文字列の長さ	有効	valueDistribution	リスト
分散	列の値の分散。	数値	有効	分散	Double
Z_SCORE_DISTRIBUTION	データポイントの z スコア値の分布を範囲別に測定します。	数値	有効	zScoreDistribution	リスト
ZEROS_COUNT	列内のゼロ (0) の数。	数値	有効	zerosCount	Int

では`IncludedStatistics`、オーバーライドを追加することで、各評価のデフォルトパラメータをオーバーライドできます。各オーバーライドには、特定の評価の名前とパラメータマップが含まれます。

ColumnStatisticsConfigurations 列のパラメータ

では`ColumnStatisticsConfigurations`、プロファイルジョブは次のパラメータをサポートします。

場合によっては、同じ列に適用`IncludedStatistics`できる異なる設定`ColumnStatisticsConfigurations`が に複数存在することがあります。このような場合、プロファイルジョブは で最後の設定を選択し`ColumnStatisticsConfigurations`、選択した列`IncludedStatistics`に適用します。新しい設定は、古い設定を上書きします。

MAXIMUM_VALUES

数値列の最大値とその数を一覧表示します。デフォルトのリストサイズは 5 です。に値を指定することで、リストサイズを上書きできます`sampleSize`。

設定

sampleSize – 数値列の値の最大数と数を含むリストのサイズ。この値は 0 より大きくする必要があります。を使用してすべての値を一覧表示"ALL"します。

例

```
{
    "Statistic": "MAXIMUM_VALUES",
    "Parameters": {
        "sampleSize": "5"
    }
}
```

最小値

数値列の最小値とその数を一覧表示します。デフォルトのリストサイズは 5 です。に値を指定することで、リストサイズを上書きできますsampleSize。

設定

sampleSize – 数値列の値の最大数と数を含むリストのサイズ。この値は 0 より大きくする必要があります。を使用してすべての値を一覧表示"ALL"します。

例

```
{
    "Statistic": "MINIMUM_VALUES",
    "Parameters": {
        "sampleSize": "5"
    }
}
```

MOST_COMMON_VALUES

列内の最も一般的な値とその数を一覧表示します。デフォルトのリストサイズは 50 です。に値を指定することで、リストサイズを上書きできますsampleSize。

設定

sampleSize – 数値列の値の最大数と数を含むリストのサイズ。この値は 0 より大きくする必要があります。を使用してすべての値を一覧表示"ALL"します。

例

```
{
    "Statistic": "MOST_COMMON_VALUES",
    "Parameters": {
        "sampleSize": "50"
    }
}
```

OUTLIER_DETECTION

Z_score アルゴリズムによって数値列または文字列列 (文字列の長さに基づく) の外れ値を検出します。

プロファイルジョブは外れ値をカウントし、外れ値とその z スコアのサンプルリストを生成します。サンプルリストは、z スコアの絶対値によって順序付けられます。デフォルトのリストサイズは 50 です。

Z_Score アルゴリズムは、平均値から標準偏差のしきい値を超える外れ値として値を識別します。デフォルトの外れ値のしきい値は 3 です。

詳細情報を取得するには、しきい値であるしきい値をもう 1 つ指定できます。しきい値はしきい値より小さくする必要があります。この機能はデフォルトでオフになっています。しきい値を指定すると、プロファイルジョブはさらに 1 つのカウント を返しますzScoreMildOutliersCount。また、この場合、 にはしきい値の異常値のサンプルを含めるzScoreOutliersSampleことができます。

設定

- threshold – 外れ値を検出するときに使用するしきい値。この値は 0 以上である必要があります。
- mildThreshold – 外れ値を検出するときに使用するしきい値。この値は 0 以上、 未満である必要がありますthreshold。
- sampleSize – 列に外れ値を含むリストのサイズ。を使用してすべての値を一覧表示"ALL"します。

例

```
{
    "Statistic": "OUTLIER_DETECTION",
    "Parameters": {
        "threshold": "5",
        "mildThreshold": "3.5",
        "sampleSize": "20"
    }
}
```

VALUE_DISTRIBUTION

値の範囲によって列の値の分布を測定します。プロファイルジョブは、数値列または文字列列 (文字列の長さに基づく) の値を数値範囲のビンにグループ化し、ビンのリストを生成します。ビンは連続しており、バケットの上限は次のバケットの上限です。

設定

binNumber – ビンの数。この値は 0 より大きくする必要があります。

例

```
{
    "Statistic": "VALUE_DISTRIBUTION",
    "Parameters": {
        "binNumber": "5"
    }
}
```

Z_SCORE_DISTRIBUTION

数値列における値の z スコアの分布を測定します。プロファイルジョブは、値の z スコアを数値範囲でビンにグループ化し、ビンのリストを生成します。ビンは連続しており、バケットの上限は次のバケットの上限です。

設定

binNumber – ビンの数。この値は 0 より大きくする必要があります。

例

```
{
    "Statistic": "Z_SCORE_DISTRIBUTION",
    "Parameters": {
        "binNumber": "5"
    }
}
```

PII を設定するための EntityDetectorConfiguration セクション

構造の EntityDetectorConfigurationセクションで、DataBrew がプロファイルジョブの個人を特定できる情報 (PII) として検出するデータセット内のエンティティタイプを設定できます。

EntityTypes

DataBrew がプロファイルジョブの PII として検出するエンティティタイプを設定します。EntityDetectorConfiguration が未定義の場合、エンティティ検出は無効になります。データセットでは、次のエンティティタイプを検出できます。

- USA_SSN
- EMAIL
- USA_ITIN
- USA_PASSPORT_NUMBER
- PHONE_NUMBER
- USA_DRIVING_LICENSE
- BANK_ACCOUNT
- CREDIT_CARD
- IP_ADDRESS
- MAC_ADDRESS
- USA_DEA_NUMBER
- USA_HCPCS_CODE
- USA_NATIONAL_PROVIDER_IDENTIFIER

- USA_NATIONAL_DRUG_CODE
- USA_HEALTH_INSURANCE_CLAIM_NUMBER
- USA_MEDICARE_BENEFICIARY_IDENTIFIER
- USA_CPT_CODE
- PERSON_NAME
- DATE

エンティティタイプグループUSA_ALLもサポートされており、 PERSON_NAMEと を除く上記のすべてのエンティティタイプが含まれていますDATE。

のタイプEntityTypesは文字列の配列です。

AllowedStatistics

検出されたエンティティを含む列で実行できる統計を設定します。AllowedStatistics が未定義の場合、検出されたエンティティを含む列では統計は計算されません。AllowedStatistics パラメータの有効な値のリスト列レベルで設定可能な統計については、「」を参照してください。

のタイプは オブジェクトの配列AllowedStatisticsですAllowedStatistics。

のセキュリティAWS Glue DataBrew

のクラウドセキュリティが最優先事項AWSです。お客様はAWS、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、AWSとお客様の間の責任共有です。責任共有モデルではこれをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ – クラウドでAWSAWSサービスを実行するインフラストラクチャを保護するAWS責任があります。AWSまた、 では、安全に使用できるサービスも提供しています。サードパーティーの監査者は、AWSコンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。が適用されるコンプライアンスプログラムの詳細についてはAWS Glue DataBrew、コンプライアンスプログラムAWSによる対象範囲内のサービスコンプライアンスプログラムを参照してください。
- クラウドのセキュリティ – お客様の責任は、使用するAWSサービスによって決まります。また、ユーザーは、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、 を使用する際の責任共有モデルの適用方法を理解するのに役立ちますAWS Glue DataBrew。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するように DataBrew を設定する方法について説明します。また、DataBrew リソースのモニタリングや保護に役立つ他のAWSサービスの使用方法についても説明します。

トピック

- でのデータ保護AWS Glue DataBrew
- の ID とアクセスの管理AWS Glue DataBrew
- DataBrew でのログ記録とモニタリング
- のコンプライアンス検証AWS Glue DataBrew
- の耐障害性AWS Glue DataBrew
- のインフラストラクチャセキュリティAWS Glue DataBrew
- での設定と脆弱性の分析AWS Glue DataBrew

でのデータ保護AWS Glue DataBrew

DataBrew には、データの保護に役立つように設計されたいくつかの機能があります。

トピック

- 保管中の暗号化
- 送信中の暗号化
- キーの管理
- 個人を特定できる情報 (PII) の特定と処理
- 他のAWSサービスへの DataBrew の依存関係

AWS責任共有モデル は、AWS Glue DataBrewでのデータ保護に適用されます。このモデルで説明されているように、AWSはすべての を実行するグローバルインフラストラクチャを保護する責任がありますAWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「AWS のサービス」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、「Data Privacy FAQChina」を参照してください。 欧州におけるデータ保護に関する情報については、General Data Protection Regulation (GDPR) Center を参照してください。

データ保護の目的で、認証情報を保護しAWS アカウント、AWS IAM アイデンティティセンターまたはAWS Identity and Access Management(IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用してAWSリソースと通信します。TLS 1.2 は必須ですが、TLS 1.3 を推奨します。
- で API とユーザーアクティビティの口グ記録を設定しますAWS CloudTrail。CloudTrail 証跡を使用してAWSアクティビティをキャプチャする方法については、「 AWS CloudTrailユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS暗号化ソリューションと、その中のすべてのデフォルトのセキュリティコントロールを使用しますAWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWSを介して にアクセスするときに FIPS 140-3 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール、API、または SDK を使用して DataBrew AWS CLIまたは他のAWS のサービスを操作する場合も同様です。AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

保管中の暗号化

DataBrew はDataBrew プロジェクトとジョブの保管時のデータ暗号化をサポートしています。プロジェクトとジョブは暗号化されたデータを読み取り、ジョブは AWS Key Management Service(AWS KMS) を呼び出して暗号化されたデータを書き込み、キーを生成してデータを復号できます。KMS キーを使用して、DataBrew ジョブによって生成されたジョブログを暗号化することもできます。DataBrew コンソールまたは DataBrew API を使用して暗号化キーを指定できます。

> **Important**
>
> AWS Glue DataBrewは対称 KMS AWSキーのみをサポートします。詳細については、「AWS Key Management Serviceデベロッパーガイド」のAWS「KMS キー」を参照してください。

暗号化を有効にして DataBrew でジョブを作成する場合、DataBrew コンソールを使用して S3-managedサーバー側の暗号化キー (SSE-S3) または に保存AWS KMSされている KMS キー (SSE-KMS) を指定し、保管中のデータを暗号化できます。

> **Important**
>
> Amazon Redshift データセットを使用すると、提供された一時ディレクトリにアンロードされたオブジェクトは SSE-S3 で暗号化されます。

DataBrew ジョブによって書き込まれたデータの暗号化

DataBrew ジョブは、暗号化された Amazon S3 ターゲットと暗号化された Amazon CloudWatch Logs に書き込むことができます。

トピック

- 暗号化を使用するように DataBrew を設定する
- VPC ジョブAWS KMSの へのルートの作成
- KMS AWSキーによる暗号化の設定

暗号化を使用するように DataBrew を設定する

暗号化を使用するように DataBrew 環境を設定するには、次の手順に従います。

暗号化を使用するように DataBrew 環境を設定するには

1. KMS AWSキーを作成または更新して、DataBrew ジョブに渡されるAWS Identity and Access Management(IAM) ロールにアクセスAWS KMS許可を付与します。これらの IAM ロールは、CloudWatch Logs と Amazon S3 ターゲットを暗号化するために使用されます。詳しくは、Amazon CloudWatch Logs ユーザーガイドの「AWS KMSを使用して CloudWatch Logs のログデータを暗号化する」を参照してください。

   次の例では、*"role1"*、、 *"role2""role3"*は DataBrew ジョブに渡される IAM ロールです。このポリシーステートメントでは、この KMS キーを使用して暗号化および復号するためのアクセス許可をリストされた IAM ロールに付与する KMS キーポリシーについて説明します。

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "logs.region.amazonaws.com",
        "AWS": [
            "role1",
            "role2",
            "role3"
        ]
    },
    "Action": [
        "kms:Encrypt*",
        "kms:Decrypt*",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:Describe*"
    ],
    "Resource": "*"
}
```

キーを使用して CloudWatch Logs を暗号化する場合、"Service": "logs.*region*.amazonaws.com" と表示される Service ステートメントが必要です。

2. キーを使用するENABLED前に、AWS KMSキーが に設定されていることを確認します。

AWS KMSキーポリシーを使用したアクセス許可の指定の詳細については、「 でのキーポリシーの使用AWS KMS」を参照してください。

VPC ジョブAWS KMSの へのルートの作成

インターネットを介さずに、仮想プライベートクラウド (VPC) のプライベートエンドポイントからAWS KMSに直接接続できます。VPC エンドポイントを使用する場合、VPC と 間の通信AWS KMSはAWSネットワーク内で完全に行われます。

AWS KMS VPC エンドポイントは VPC 内に作成できます。このステップを行わないと、DataBrew ジョブが で失敗する可能性がありますkms timeout。詳細な手順については、 AWS Key Management Serviceデベロッパーガイドの「VPC エンドポイントAWS KMSを介した への接続」を参照してください。

これらの指示に従うときは、VPC コンソールで、次の操作を行ってください。

- プライベート DNS 名を有効にするを選択します。
- セキュリティグループで、Java Database Connectivity (JDBC) にアクセスする DataBrew ジョブに使用するセキュリティグループ (自己参照ルールを含む) を選択します。

JDBC データストアにアクセスする DataBrew ジョブを実行する場合、DataBrew にはAWS KMSエンドポイントへのルートが必要です。ルートには、ネットワークアドレス変換 (NAT) ゲートウェイまたはAWS KMS VPC エンドポイントを指定できます。NAT ゲートウェイを作成するには、Amazon VPC ユーザーガイドの「NAT ゲートウェイ」を参照してください。

KMS AWSキーによる暗号化の設定

ジョブで暗号化を有効にすると、Amazon S3 と CloudWatch の両方に適用されます。 CloudWatch 渡される IAM ロールには、次のAWS KMSアクセス許可が必要です。

詳細については、Amazon Simple Storage Service コンソールユーザーガイドの以下のトピックを参照してください。

- SSE-S3 の詳細については、「Amazon S3 で管理された暗号化キーによるサーバー側の暗号化 (SSE-S3) を使用したデータの保護」を参照してください。
- の詳細についてはSSE-KMS、AWS「KMS マネージドキーによるサーバー側の暗号化を使用したデータの保護 (SSE-KMS)」を参照してください。

送信中の暗号化

AWSは、飛行中のデータに対して Secure Sockets Layer (SSL) 暗号化を提供します。

DataBrew による JDBC データソースのサポートが利用できますAWS Glue。JDBC データソースに接続する場合、DataBrew は SSL AWS Glue接続を要求するオプションなど、接続の設定を使用します。 詳細については、 AWS Glueデベロッパーガイドの AWS Glue「接続プロパティ -AWS Glue」を参照してください。

AWS KMSは、DataBrew 抽出、変換、ロード (ETL) 処理と に対して、「独自のキーを持ち込む」暗号化とサーバー側の暗号化の両方を提供しますAWS Glue Data Catalog。

キーの管理

DataBrew で IAM を使用して、アクセス、拒否などに関するユーザー、AWSリソース、グループ、ロール、きめ細かなポリシーを定義できます。

メタデータへのアクセスは、組織のニーズに応じて、リソースベースのポリシーとアイデンティティベースのポリシーの両方を使用して定義できます。リソースベースのポリシーでは、リソースへのアクセスを許可または拒否するプリンシパルを一覧表示し、クロスアカウントアクセスなどのポリシーを設定できます。アイデンティティポリシーは、IAM 内の特定のユーザー、グループ、およびロールにアタッチします。

DataBrew は、独自のAWS KMS key「独自のキーを持ち込む」暗号化の作成をサポートしています。DataBrew は、DataBrew ジョブAWS KMSの の KMS キーを使用したサーバー側の暗号化も提供します。

個人を特定できる情報 (PII) の特定と処理

分析関数または機械学習モデルを構築する場合、個人を特定できる情報 (PII) データが公開されないようにするための保護が必要です。PII は、住所、銀行口座番号、電話番号など、個人を特定するために使用できる個人データです。たとえば、データアナリストやデータサイエンティストがデータセットを使用して一般的な人口統計情報を検出する場合、特定の個人の PII にアクセスすることはできません。

DataBrew は、データ準備プロセス中に PII データを難読化するデータマスキングメカニズムを提供します。組織のニーズに応じて、さまざまな PII データ秘匿化メカニズムを使用できます。PII データを難読化して、ユーザーが元に戻せないようにしたり、難読化を元に戻したりすることができます。

DataBrew で PII データを特定してマスキングするには、顧客が PII データの編集に使用できる一連の変換を構築する必要があります。このプロセスの一環として、DataBrew コンソールのデータプロファイルの概要ダッシュボードに PII データ検出と統計情報を提供します。

次のデータマスキング手法を使用できます。

- 置換 – PII データを他の真正な値に置き換えます。
- シャッフル — 同じ列の値を異なる行にシャッフルします。
- 確定的暗号化 – 確定的暗号化アルゴリズムを列値に適用します。確定的暗号化では、値に対して常に同じ暗号文が生成されます。
- 確率的暗号化 – 確率的暗号化アルゴリズムを列値に適用します。確率的暗号化は、適用されるたびに異なる暗号文が生成されます。
- 復号 — 暗号化キーに基づいて列を復号します。
- Nulling out or delete – 特定のフィールドを null 値に置き換えるか、列を削除します。
- マスキング — 文字スクランブルを使用するか、列内の特定の部分をマスクします。
- ハッシュ — ハッシュ関数を列値に適用します。

変換の使用の詳細については、「個人を特定できる情報 (PII) レシピの手順」を参照してください。検出可能なエンティティタイプのリストなど、プロファイルジョブを使用して PII を検出する方法の詳細については、「プロファイルジョブ設定をプログラムで構築する」の「PII を設定するための EntityDetectorConfiguration」セクションを参照してください。

他のAWSサービスへの DataBrew の依存関係

DataBrew コンソールを使用するには、AWSアカウントの DataBrew リソースを操作するための最小限のアクセス許可のセットが必要です。これらの DataBrew アクセス許可に加えて、コンソールには以下のサービスからのアクセス許可が必要です。

- CloudWatch Logs ログを表示するためのアクセス許可。
- ロールを一覧表示して渡す IAM アクセス許可。

- VPCs、サブネット、セキュリティグループ、インスタンス、およびその他のオブジェクトを一覧表示するための Amazon EC2 アクセス許可。DataBrew はこれらのアクセス許可を使用して、DataBrew ジョブの実行時に VPCsなどの Amazon EC2 項目を設定します。
- バケットとオブジェクトを一覧表示するための Amazon S3 アクセス許可。
- AWS Glueデータベース、パーティション、テーブル、接続などのAWS Glueスキーマオブジェクトを読み取るための アクセス許可。
- AWS Lake Formation Lake Formation データレイクを操作するための アクセス許可。

の ID とアクセスの管理AWS Glue DataBrew

AWS Identity and Access Management(IAM) は、管理者がAWSリソースへのアクセスを安全に制御AWS のサービスするのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に DataBrew リソースの使用を許可する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できるAWS のサービスです。

トピック

- アイデンティティを使用した認証
- ポリシーを使用したアクセスの管理
- AWS Glue DataBrewおよびAWS Lake Formation
- が IAM とAWS Glue DataBrew連携する方法
- のアイデンティティベースのポリシーの例AWS Glue DataBrew
- AWSの 管理ポリシーAWS Glue DataBrew
- での ID とアクセスのトラブルシューティングAWS Glue DataBrew

アイデンティティを使用した認証

認証とは、ID 認証情報AWSを使用して にサインインする方法です。、IAM ユーザーAWS アカウントのルートユーザー、または IAM ロールを引き受けることで認証される必要があります。

AWS IAM アイデンティティセンター(IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーティッド ID としてサインインできます。サインインの詳細については、「AWS サインインユーザーガイド」の「AWS アカウントにサインインする方法」を参照してください。

プログラムによるアクセスの場合、 は SDK と CLI AWSを提供してリクエストを暗号化して署名します。詳細については、「IAM ユーザーガイド」の「API リクエストに対するAWS署名バージョン 4」を参照してください。

AWS アカウントルートユーザー

を作成するときはAWS アカウント、すべてのAWS のサービスおよび リソースへの完全なアクセス権を持つAWS アカウントroot ユーザーと呼ばれる 1 つのサインインアイデンティティから始めます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザー認証情報を必要とするタスクについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

ユーザーとグループ

IAM ユーザーは、1 人のユーザーまたは 1 つのアプリケーションに対して特定のアクセス許可を持つ ID です。長期認証情報を持つ IAM ユーザーの代わりに一時的な認証情報を使用することをお勧めします。詳細については、IAM ユーザーガイドの「ID プロバイダーとのフェデレーションを使用してにアクセスする必要があるAWS」を参照してください。

IAM グループは、IAM ユーザーの集合を指定し、大量のユーザーに対するアクセス許可の管理を容易にします。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可を持つアイデンティであり、一時的な認証情報を提供します。ユーザーから IAM ロール (コンソール) に切り替えるか、 または API オペレーションを呼び出すことで、ロールを引き受けることができます。AWS CLIAWS詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールは、フェデレーションユーザーアクセス、一時的な IAM ユーザーのアクセス許可、クロスアカウントアクセス、クロスサービスアクセス、および Amazon EC2 で実行するアプリケーションに役立ちます。詳細については、IAM ユーザーガイド の IAM でのクロスアカウントリソースアクセス を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御するAWSには、ポリシーを作成し、ID AWSまたはリソースにアタッチします。ポリシーは、アイデンティティまたはリソースに関連付けられたときにアクセス許可を定義します。は、プリンシパルがリクエストを行うときに、これらのポリシーAWSを評価します。ほとんどのポ

リシーは JSON ドキュメントAWSとして に保存されます。JSON ポリシードキュメントの詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は、ポリシーを使用して、どのプリンシパルがどのリソースに対して、どのような条件でアクションを実行できるかを定義することで、誰が何にアクセスできるかを指定します。

デフォルトでは、ユーザーやロールにアクセス許可はありません。IAM 管理者は IAM ポリシーを作成してロールに追加し、このロールをユーザーが引き受けられるようにします。IAM ポリシーは、オペレーションの実行方法を問わず、アクセス許可を定義します。

アイデンティティベースのポリシー

アイデンティティベースのポリシーは、アイデンティティ (ユーザー、グループ、またはロール) にアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、アイデンティティがどのリソースに対してどのような条件下でどのようなアクションを実行できるかを制御します。アイデンティティベースポリシーの作成方法については、IAM ユーザーガイド の カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する を参照してください。

アイデンティティベースのポリシーは、インラインポリシー (単一の ID に直接埋め込む) または管理ポリシー (複数の ID にアタッチされたスタンドアロンポリシー) にすることができます。管理ポリシーとインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「管理ポリシーとインラインポリシーのいずれかを選択する」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。例としては、IAM ロール信頼ポリシーや Amazon S3 バケットポリシーなどがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM のAWSマネージドポリシーを使用できません。

DataBrew はリソースベースのポリシーをサポートしていません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするためのアクセス許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、および Amazon VPC はAWS WAF、ACLs。ACL の詳細については、Amazon Simple Storage Service デベロッパーガイド の アクセスコントロールリスト (ACL) の概要 を参照してください。

DataBrew は ACLs をサポートしていません。

その他のポリシータイプ

AWSは、より一般的なポリシータイプによって付与されるアクセス許可の最大数を設定できる追加のポリシータイプをサポートしています。

- アクセス許可の境界 – アイデンティティベースのポリシーで IAM エンティティに付与することのできるアクセス許可の数の上限を設定します。詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可境界」を参照してください。
- サービスコントロールポリシー (SCP) -AWS Organizations内の組織または組織単位の最大のアクセス許可を指定します。詳細については、「AWS Organizationsユーザーガイド」の「サービスコントロールポリシー」を参照してください。
- リソースコントロールポリシー (RCP) – は、アカウント内のリソースで利用できる最大数のアクセス許可を定義します。詳細については、「AWS Organizationsユーザーガイド」の「リソースコントロールポリシー (RCP)」を参照してください。
- セッションポリシー – ロールまたはフェデレーションユーザーの一時セッションを作成する際にパラメータとして渡される高度なポリシーです。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成されるアクセス許可を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうかAWSを決定する方法については、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

AWS Glue DataBrewおよびAWS Lake Formation

AWS Glue DataBrewはAWS Glue Data CatalogテーブルのAWS Lake Formationアクセス許可をサポートします。データセットが Lake Formation に登録されているAWS Glue Data Catalogテーブルを使用する場合、プロジェクトまたはジョブに提供される IAM ロールには、テーブルに対する DESCRIBE および SELECT Lake Formation アクセス許可が必要です。

AWS Glue DataBrewでは、 に基づくAWS Glue Data Catalogテーブルへの書き込みがサポートされていますAWS Lake Formation。DataBrew ジョブが Lake Formation に登録されているデータカタログを使用する場合、ジョブに提供される IAM ロールには、関連するテーブルに対する Lake Formation からの INSERT、ALTER、および DELETE アクセス許可が必要です。IAM ロールには、`glue:UpdateTable` アクセス許可と、Data Catalog テーブルに関連付けられたデータロケーションへのアクセス許可が必要です。

が IAM とAWS Glue DataBrew連携する方法

IAM を使用して DataBrew へのアクセスを管理する前に、DataBrew で使用できる IAM 機能を理解しておく必要があります。DataBrew およびその他のAWSのサービスが IAM と連携する方法の概要については、IAM ユーザーガイドのAWS「IAM と連携する のサービス」を参照してください。

トピック

- DataBrew アイデンティティベースのポリシー
- DataBrew のリソースベースのポリシー
- DataBrew IAM ロール

DataBrew アイデンティティベースのポリシー

IAM のアイデンティティベースのポリシーを使用すると、許可または拒否されるアクションとリソース、さらにアクションが許可または拒否される条件を指定できます。DataBrew は、特定のアクション、リソース、および条件キーをサポートしています。JSON ポリシーで使用するすべての要素については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素のリファレンス」を参照してください。

アクション

管理者は JSON AWSポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、JSON AWSポリシーは、どのプリンシパルがどのリソースに対してどの条件でアクションを実行できるかを指定できます。

JSON ポリシーのアクション要素は、ポリシーでアクセスを許可または拒否できるアクションを記述します。ポリシーアクションの名前は通常、関連するAWS API オペレーションと同じです。一致する API オペレーションのない 許可のみのアクション など、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは 依存アクション と呼ばれます。

このアクションは関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

DataBrew のポリシーアクションは、アクションの前にプレフィックス を使用しますdatabrew:。たとえば、 Amazon EC2 RunInstances API オペレーションで Amazon EC2 インスタンスを実行するためのアクセス許可をユーザーに付与するには、ポリシーに ec2:RunInstances アクションを含めます。ポリシーステートメントには、Action または NotAction エレメントを含める必要があります。DataBrew は、ユーザーが実行できるタスクを記述する独自のアクションのセットを定義します。

単一のステートメントに複数の アクションを指定するには、次のようにコンマで区切ります。

```
"Action": [
      "databrew:CreateRecipeJob",
      "databrew:UpdateSchedule"
```

ワイルドカード *を使用して複数のアクションを指定することができます。例えば、Describe という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": "databrew:Describe*"
```

DataBrew アクションのリストを確認するには、IAM ユーザーガイドの「 で定義されるアクション AWS Glue DataBrew」を参照してください。

リソース

管理者は JSON AWSポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Resource JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。リソースレベルのアクセス許可をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

以下は、リソースレベルのアクセス許可をサポートしていない DataBrew APIsです。

- ListDatasets

- ListJobs
- ListProjects
- ListRecipes
- ListRulesets
- ListSchedules

DataBrew データセットリソースには、次の Amazon リソースネーム (ARN) があります。

```
arn:${Partition}:databrew:${Region}:${Account}:dataset/${Name}
```

ARNs、「Amazon リソースネーム (ARNsとAWSサービス名前空間」を参照してください。

例えば、ステートメントで i-1234567890abcdef0 インスタンスを指定するには、以下の ARN を使用します。

```
"Resource": "arn:aws:databrew:us-east-1:123456789012:dataset/my-chess-dataset"
```

特定のアカウントに属するすべてのインスタンスを指定するには、ワイルドカード *を使用します。

```
"Resource": "arn:aws:databrew:us-east-1:123456789012:dataset/*"
```

特定のリソースに対して、リソースを作成するためのアクションなど、一部の DataBrew アクションを実行することはできません。このような場合はワイルドカード *を使用する必要があります。

```
"Resource": "*"
```

DataBrew リソースタイプとその ARNs」を参照してください。 AWS Glue DataBrew どのアクションで各リソースの ARN を指定できるかについては、AWS Glue DataBrewで定義されるアクションを参照してください。

条件キー

DataBrew はサービス固有の条件キーを提供しませんが、一部のグローバル条件キーの使用をサポートしています。すべてのAWSグローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS「グローバル条件コンテキストキー」を参照してください。

例

DataBrew アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいのアイデンティティベースのポリシーの例AWS Glue DataBrew。

DataBrew のリソースベースのポリシー

DataBrew はリソースベースのポリシーをサポートしていません。

DataBrew IAM ロール

IAM ロールは、特定のアクセス許可を持つAWSアカウント内のエンティティです。

DataBrew での一時的な認証情報の使用

一時的な認証情報を使用して、フェデレーションでサインインする、IAM 役割を引き受ける、またはクロスアカウント役割を引き受けることができます。一時的なセキュリティ認証情報を取得するには、AssumeRole や GetFederationToken などのAWS STS API オペレーションを呼び出します。

DataBrew は一時的な認証情報の使用をサポートしています。

サービスリンクロール

サービスにリンクされたロールを使用すると、AWSサービスは他の サービスのリソースにアクセスして、ユーザーに代わってアクションを実行できます。サービスリンクロールは IAM アカウント内に表示され、サービスによって所有されます。 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

DataBrew での IAM ロールの選択

DataBrew でデータセットリソースを作成するときは、ユーザーに代わって DataBrew アクセスを許可する IAM ロールを選択します。以前にサービスロールまたはサービスにリンクされたロールを作成している場合、DataBrew は選択するロールのリストを提供します。必要に応じて、Amazon S3 バケットまたはAWS Glue Data Catalogリソースへの読み取りアクセスを許可するロールを選択してください。

のアイデンティティベースのポリシーの例AWS Glue DataBrew

デフォルトでは、ユーザーとロールには DataBrew リソースを作成または変更するアクセス許可はありません。また、AWS マネジメントコンソール、AWS CLI、またはAWS APIs を使用してタスクを実行することはできません。IAM 管理者は、指定されたリソースで特定の API 操作を実行するための許可をユーザーとロールに付与する IAM ポリシーを作成する必要があります。続いて、管理者はそれらのアクセス許可が必要なユーザーまたはグループにそのポリシーをアタッチします。

JSON ポリシードキュメントのこれらの例を使用して、IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「JSON タブでのポリシーの作成」を参照してください。

トピック

- ポリシーに関するベストプラクティス
- DataBrew コンソールの使用
- ユーザー自身のアクセス許可を表示することをユーザーに許可する
- タグに基づく DataBrew リソースの管理

ポリシーに関するベストプラクティス

ID ベースのポリシーは、アカウント内で DataBrew リソースを作成、アクセス、または削除できるかどうかを決定します。これらのアクションでは、AWS アカウントに費用が発生する場合があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与するAWS管理ポリシーを使用します。これらは で使用できますAWS アカウント。ユースケースに固有のAWSカスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、IAM ユーザーガイド の AWSマネージドポリシー または ジョブ機能のAWSマネージドポリシー を参照してください。
- 最小特権を適用する – IAM ポリシーでアクセス許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、IAM ユーザーガイド の IAM でのポリシーとアクセス許可 を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。たとえば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、サービスアクションが などの特定の を通じて使用されている場合にAWS のサービス、サービスアクションへのアクセスを許可することもできますCloudFormation。詳細については、IAM ユーザーガイド の IAM JSON ポリシー要素:条件 を参照してください。
- IAM アクセスアナライザー を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM アクセスアナライザー は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー

言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、IAM ユーザーガイド の IAM Access Analyzer でポリシーを検証する を参照してください。

- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合はAWS アカウント、MFA をオンにしてセキュリティを強化します。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、IAM ユーザーガイド の MFA を使用した安全な API アクセス を参照してください。

IAM でのベストプラクティスの詳細については、IAM ユーザーガイド の IAM でのセキュリティのベストプラクティス を参照してください。

DataBrew コンソールの使用

AWS Glue DataBrewコンソールにアクセスするには、最小限のアクセス許可のセットが必要です。これらのアクセス許可により、AWSアカウントの DataBrew リソースの詳細を一覧表示および表示できます。最低限必要なアクセス許可よりも制限の厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つユーザーまたはロールに対してコンソールは意図したとおりに機能しません。

ユーザーとロールが DataBrew コンソールを使用できるようにするには、エンティティに次のAWS管理ポリシーもアタッチします。詳細については、「IAM ユーザーガイド」の「ユーザーへのアクセス許可の追加」を参照してください。

```
AWSDataBrewConsoleAccess
```

AWS CLIまたは DataBrew API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

ユーザー自身のアクセス許可を表示することをユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、またはAWS CLIまたはAWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

タグに基づく DataBrew リソースの管理

ID ベースのポリシーの条件を使用して、タグに基づいて DataBrew リソースを管理できます。たとえば、リソースを削除、更新、または記述できます。次の例は、プロジェクトの削除を拒否するポリシーを示しています。ただし、プロジェクトタグ Owner の値が admin の場合にのみ、削除は拒否されます。このポリシーは、コンソールでこのアクションを拒否するために必要なアクセス許可も付与します。

JSON

```
{
```

```
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "DeleteResourceInConsole",
         "Effect": "Allow",
         "Action": "databrew:DeleteProject",
         "Resource": "*"
       },
       {
         "Sid": "DenyDeleteProjectIfAdminTag",
         "Effect": "Deny",
         "Action": "databrew:DeleteProject",
         "Resource": "arn:aws:databrew:*:*:project/*",
         "Condition": {
            "StringEquals": {"aws:ResourceTag/Owner": "admin"}
         }
      }
   ]
}
```

このポリシーをアカウントの ユーザーにアタッチできます。richard-roe という名前のユーザーが DataBrew プロジェクトを削除しようとする場合、リソースに Owner=admin または owner=admin というタグを付けることはできません。それ以外の場合、ユーザーはプロジェクトを削除するアクセス許可を拒否されます。条件タグキー Owner は、条件キー名で大文字と小文字が区別されないため、所有者と所有者の両方に一致します。詳細については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素: 条件」を参照してください。

> **Note**
>
> ListDatasets、ListJobs、ListProjects、ListRecipes、ListRulesets、および ListSchedules は、タグベースのアクセスコントロールをサポートしていません。

AWSの 管理ポリシーAWS Glue DataBrew

ユーザー、グループ、ロールにアクセス許可を追加するには、自分でポリシーを記述するよりもAWS管理ポリシーを使用する方が簡単です。チームに必要な権限のみを提供する IAM カスタマーマネージドポリシーを作成する には時間と専門知識が必要です。すぐに開始するには、AWSマネージドポリシーを使用できます。これらのポリシーは一般的なユースケースを対象としており、AWSア

カウントで利用できます。AWS管理ポリシーの詳細については、IAM ユーザーガイドの「AWS管理ポリシー」を参照してください。

AWSサービスは、AWS管理ポリシーを維持および更新します。AWS管理ポリシーのアクセス許可は変更できません。サービスは、新機能をサポートするために、AWS管理ポリシーに追加のアクセス許可を追加することがあります。この種類の更新はポリシーがアタッチされている、すべてのアイデンティティ (ユーザー、グループおよびロール) に影響を与えます。サービスは、新機能が起動されたとき、または新しいオペレーションが利用可能になったときに、AWSマネージドポリシーを更新する可能性が最も高いです。サービスはAWSマネージドポリシーからアクセス許可を削除しないため、ポリシーの更新によって既存のアクセス許可が損なわれることはありません。

さらに、 は、複数のサービスにまたがるジョブ関数の マネージドポリシーAWSをサポートしています。例えば、ReadOnlyAccessAWS管理ポリシーは、すべてのAWSサービスとリソースへの読み取り専用アクセスを提供します。サービスが新機能を起動すると、 は新しいオペレーションとリソースの読み取り専用アクセス許可AWSを追加します。職務機能ポリシーのリストと説明については、IAM ユーザーガイドのAWS職務機能の管理ポリシーを参照してください。

AWSマネージドポリシーへの DataBrew 更新

このサービスがこれらの変更の追跡を開始してからの DataBrew のAWSマネージドポリシーの更新に関する詳細を表示します。このページの変更に関する自動アラートについては、DataBrew ドキュメント履歴ページの RSS フィードにサブスクライブしてください。管理ポリシーは、 のAWS IAM コンソールにありますAwsGlueDataBrewFullAccessPolicy。

変更	説明	日付
AWSGlueDataBrewServiceRole – の読み取りアクセス許可が追加されAWS Glueました。	この更新では、 が追加されますglue:GetCustomEntityType 。このアクセス許可は、PII ID を有効にしてAWS Glue DataBrewプロファイルジョブを実行するために必要です。	2024 年 3 月 20 日
AWSGlueDataBrewServiceRole - の読み取りアクセス許可が追加されAWS Glueました。	この更新では、 が追加されますglue:BatchGetCustomEntityTypes 。このアクセス許可は、PII ID を有効に	2022 年 5 月 9 日

変更	説明	日付
	してAWS Glue DataBrewプロファイルジョブを実行するために必要です。	
AwsGlueDataBrewFullAccessPolicy - Amazon Redshift-Data DescribeStatements と Amazon S3 GetLifecycleConfiguration の読み取りアクセス許可が追加されました。	この更新では`redshift-data:DescribeStatement` 、Amazon Redshift ベースのデータセットの作成時に SQL の検証をサポートする が追加されました。また、 を追加して`s3:GetLifecycleConfiguration` 、一時ディレクトリとして提供する Amazon S3 バケットプレフィックスにライフサイクルが設定されているかどうかを評価します。さらに、この変更により、「databrew:*」アクセス許可が、すべての DataBrew APIs。	2022 年 2 月 4 日

変更	説明	日付
AwsGlueDataBrewFullAccessPolicy -AWS Secrets Manager の読み取り/書き込みアクセス許可が追加されました。	この更新では、DataBrew 変換で使用するデフォルトのシークレットdatabrew!default である という名前のシークレットsecretsmanager:GetSecretValue に secretsmanager:CreateSecret と が追加されます。さらに、DataBrew コンソールからシークレットを作成AwsGlueDataBrew- するためのプレフィックスが付けられたシークレットの CreateSecret にアクセス許可を追加します。AWS Key Management Service API リファレンスで説明されている GenerateRandom は、暗号的に安全なランダムなバイト文字列を生成するために使用されます。	2021 年 11 月 18 日
AWSGlueDataBrewServiceRole -AWS Secrets Manager の読み取り/書き込みアクセス許可が追加されました。	この更新secretsmanager:GetSecretValue では、DataBrew 変換で使用するデフォルトのシークレットdatabrew!default である という名前のシークレットに が追加されました。	2021 年 11 月 18 日

変更	説明	日付
AwsGlueDataBrewFullAccessPolicy -AWS Secrets Manager の読み取り/書き込みアクセス許可が追加されました。	この更新では、DataBrew 変換で使用するデフォルトのシークレットdatabrew! default である という名前のシークレットsecretsmanager:GetSecretValue に secretsmanager:CreateSecret と が追加されます。さらに、DataBrew コンソールからシークレットを作成AwsGlueDataBrew- するためのプレフィックスが付いたシークレットの CreateSecret にアクセス許可を追加します。 kms:GenerateRandom (https://docs.aws.amazon.com/kms/latest/APIReference/API_GenerateRandom.html) は、暗号的に安全なランダムなバイト文字列を生成するために使用されます。	2021 年 11 月 18 日
AWSGlueDataBrewServiceRole -AWS Secrets Manager の読み取り/書き込みアクセス許可が追加されました。	この更新secretsmanager:GetSecretValue では、DataBrew 変換で使用するデフォルトのシークレットdatabrew! default である という名前のシークレットに が追加されました。	2021 年 11 月 18 日

変更	説明	日付
AwsGlueDataBrewFullAccessPolicy -AWS Glueカタログデータベースの読み取りアクセス許可とAWS Glueカタログテーブルの作成アクセス許可が追加されました。	この更新により、DataBrewジョブへの出力の設定の一環として、AWS Glueカタログデータベースを一覧表示し、既存のデータベースの下に新しいカタログテーブルを作成するアクセス許可が追加されます。	2021 年 6 月 30 日
AwsGlueDataBrewFullAccessPolicy - Amazon AppFlow データセット機能の読み取り/書き込みアクセス許可が追加されました。	この更新では、既存のAmazon AppFlow フローとフロー実行を読み取り、フロー実行を作成するアクセス許可が追加されます。	2021 年 4 月 28 日
AwsGlueDataBrewFullAccessPolicy - データベースデータセットの読み取りアクセス許可が追加されました。	この更新では、既存のAWS Glue接続を読み取り、DataBrew で使用する新しいAWS Glue接続を作成するアクセス許可が追加されました。 また、コンソールで新しい接続を簡単に作成できるように、Amazon VPC リソースとAmazon Redshift クラスターのリストを作成できます。また、シークレットを一覧表示するアクセス許可を付与しますが、シーAWS Secrets Managerクレットを読み取ることはできません。	2021 年 3 月 30 日
DataBrew が変更の追跡を開始	DataBrew はAWS、管理ポリシーの変更の追跡を開始しました。	2021 年 3 月 30 日

での ID とアクセスのトラブルシューティングAWS Glue DataBrew

以下の情報は、DataBrew と IAM の使用時に発生する可能性がある一般的な問題の診断と修正に役立ちます。

トピック

DataBrew でアクションを実行する権限がありません

がアクションを実行する権限がないとAWS マネジメントコンソール通知した場合は、管理者に連絡してサポートを依頼してください。管理者とは、サインイン認証情報を提供した担当者です。

次の例のエラーは、mateojackson ユーザーがコンソールを使用してプロジェクトの詳細を表示する際に databrew:DescribeProject アクセス許可を持っていない場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 databrew:DescribeProject on resource: my-example-project
```

この場合、Mateo は、databrew:*GetProject* アクションを使用して *my-example-project* リソースにアクセスできるように、管理者にポリシーの更新を依頼します。

iam:PassRole を実行する権限がない

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して DataBrew にロールを渡すことができるようにする必要があります。

一部のAWS のサービスでは、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、既存のロールをそのサービスに渡すことができます。そのためには、サービスにロールを渡すアクセス許可が必要です。

次の例のエラーは、 という IAM ユーザーがコンソールを使用して DataBrew marymajor でアクションを実行しようとすると発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与されたアクセス許可が必要です。Mary には、ロールをサービスに渡すアクセス許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、AWS管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

自分のAWSアカウント以外のユーザーに DataBrew リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- DataBrew がこれらの機能をサポートしているかどうかを確認するには、「」を参照してくださいが IAM とAWS Glue DataBrew連携する方法。
- 所有AWS アカウントしている のリソースへのアクセスを提供する方法については、「IAM ユーザーガイド」の「所有AWS アカウントしている別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法についてはAWS アカウント、IAM ユーザーガイドの「サードパーティーAWS アカウントが所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイド の 外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可 を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、IAM ユーザーガイド の IAM でのクロスアカウントのリソースへのアクセス を参照してください。

DataBrew でのログ記録とモニタリング

モニタリングは、DataBrew とAWSソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。マルチポイント障害が発生した場合は、その障害をより簡単にデバッグできる

ように、AWSソリューションのすべての部分からモニタリングデータを収集する必要があります。には、DataBrew リソースをモニタリングし、潜在的なインシデントに対応するための複数のツールAWSが用意されています。

Amazon CloudWatch アラーム

Amazon CloudWatch アラームを使用して、指定した期間中、単一のメトリクスをモニタリングします。メトリクスが特定のしきい値を超えると、Amazon SNS トピックまたはAWS Auto Scalingポリシーに通知が送信されます。CloudWatch アラームは、特定の状態にあるという理由ではアクションを呼び出しません。その代わり、状態が変更され、指定期間にわたって維持される必要があります。

AWS CloudTrailログ

CloudTrail は、DataBrew のユーザー、ロール、またはAWSのサービスによって実行されたアクションの記録を提供します。CloudTrail によって収集された情報を使用して、DataBrew に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

のコンプライアンス検証AWS Glue DataBrew

サードパーティーの監査者は、複数のコンプライアンスプログラムの一環としてAWS Glue DataBrewのセキュリティとAWSコンプライアンスを評価します。これらのプログラムには、SOC、PCI、FedRAMP、HIPAA などがあります。

AWS のサービスが特定のコンプライアンスプログラムの対象であるかどうかを確認するには、「コンプライアンスAWS のサービスプログラムによる対象範囲内のコンプライアンス」を参照し、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできますAWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

を使用する際のお客様のコンプライアンス責任AWS のサービスは、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。を使用する際のコンプライアンス責任の詳細についてはAWS のサービス、AWS「 セキュリティドキュメント」を参照してください。

の耐障害性AWS Glue DataBrew

AWSグローバルインフラストラクチャは、AWSリージョンとアベイラビリティーゾーンを中心に構築されています。AWSリージョンは、低レイテンシー、高スループット、高度に冗長なネットワークで接続された、物理的に分離された複数のアベイラビリティーゾーンを提供します。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェールオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、フォールトトレランス、および拡張性が優れています。

ではAWS Glue DataBrew、1 つ以上の再試行を使用するようにジョブを設定することをお勧めします。ジョブの再試行回数は、DataBrew コンソールの高度なジョブ設定で設定されます。

AWSリージョンとアベイラビリティーゾーンの詳細については、AWS「グローバルインフラストラクチャ」を参照してください。

のインフラストラクチャセキュリティAWS Glue DataBrew

マネージドサービスの一環として、AWS Glue DataBrewは、ホワイトペーパー「Amazon Web Services: セキュリティプロセスの概要」に記載されているAWSグローバルネットワークセキュリティ手順で保護されています。

AWS公開された API コールを使用して、ネットワーク経由で DataBrew にアクセスします。クライアントで Transport Layer Security (TLS) 1.0 以降がサポートされている必要があります。TLS 1.2 以降が推奨されています。また、Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートもクライアントでサポートされている必要があります。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストにはアクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

トピック

- VPC AWS Glue DataBrewでの の使用
- VPC エンドポイントAWS Glue DataBrewでの の使用

VPC AWS Glue DataBrewでの の使用

Amazon VPC を使用してAWSリソースをホストする場合、Amazon VPC サービスに基づいて Virtual Private Cloud (VPC) を介してトラフィックをルーティングAWS Glue DataBrewするように を設定できます。DataBrew は、最初に指定したサブネットに Elastic Network Interface をプロビジョニングすることでこれを行います。次に、DataBrew はそのネットワークインターフェイスに指定したセキュリティグループをアタッチしてアクセスを制御します。指定されたセキュリティグループには、すべてのトラフィックに対して自己参照のインバウンドルールとアウトバウンドルールが必要です。また、VPC で DNS ホスト名と解決が有効になっている必要があります。詳細については、AWS Glueデベロッパーガイドの「JDBC データストアに接続するための VPC のセットアップ」を参照してください。

AWS Glue Data Catalogデータセットの場合、VPC 情報は Data Catalog で接続を作成するAWS Glueときに設定されます。この接続用の Data Catalog テーブルを作成するには、AWS Glueコンソールからクローラを実行します。詳細については、 AWS Glueデベロッパーガイドの「AWS Glue Data Catalogの入力」を参照してください。

データベースデータセットの場合は、DataBrew コンソールから接続を作成するときに VPC 情報を指定します。

NAT を使用しない VPC サブネットAWS Glue DataBrewで を使用するには、Amazon S3 へのゲートウェイ VPC エンドポイントとAWS Glueインターフェイスの VPC エンドポイントが必要です。詳細については、Amazon VPC ドキュメントの「ゲートウェイエンドポイントとインターフェイス VPC エンドポイントの作成 ()」を参照してください。 AWS PrivateLinkDataBrew によってプロビジョニングされた Elastic Interface にはパブリック IPv4 アドレスがないため、VPC インターネットゲートウェイの使用をサポートしていません。

Amazon S3 インターフェイスエンドポイントは現在サポートされていません。AWS Secrets Managerを使用してシークレットを保存する場合は、Secrets Manager へのルートが必要です。暗号化を使用している場合は、AWS Key Management Service() へのルートが必要ですAWS KMS。

VPC エンドポイントAWS Glue DataBrewでの の使用

Amazon VPC を使用してAWSリソースをホストする場合は、VPC エンドポイントをプロビジョニングすることで、VPC と DataBrew の間にプライベート接続を確立できます。この VPC エンドポイントを使用すると、DataBrew API コールを実行できます。

DataBrew VPC エンドポイントは、VPC で DataBrew を使用する必要はありません。詳細については、「VPC AWS Glue DataBrewでの の使用」を参照してください。

と VPC エンドポイントの両方をサポートするすべてのAWSリージョンでAWS Glue、VPC エンドポイントAWS Glueで を使用できます。

詳細については、Amazon VPC ユーザーガイドの次のトピックを参照してください。

- Amazon VPC とは?
- インターフェイスエンドポイントの作成

での設定と脆弱性の分析AWS Glue DataBrew

設定と IT コントロールは、AWSお客様と当社のお客様との間の責任共有です。詳細については、AWS「 責任共有モデル」を参照してください。

モニタリングAWS Glue DataBrew

モニタリングは、 およびその他のAWS Glue DataBrewAWSソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。 には、DataBrew を監視し、問題が発生したときに報告し、必要に応じて自動アクションを実行するための以下のモニタリングツールAWSが用意されています。

- Amazon CloudWatch は、AWSリソースと で実行されるアプリケーションをAWSリアルタイムでモニタリングします。メトリクスの収集と追跡、カスタマイズしたダッシュボードの作成、および指定したメトリクスが指定したしきい値に達したときに通知またはアクションを実行するアラームの設定を行うことができます。例えば、CloudWatch で Amazon EC2 インスタンスの CPU 使用率などのメトリクスを追跡し、必要に応じて新しいインスタンスを自動的に起動できます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。
- Amazon CloudWatch Events では、DataBrew で特定のイベントの自動通知を設定できます。DataBrew からのイベントは、ほぼリアルタイムで CloudWatch Events に配信されます。イベントをモニタリングし、リソース共有の変更を示すイベントに応答してターゲットを呼び出すように CloudWatch Events を設定できます。リソース共有への変更は、リソース共有の所有者およびリソース共有へのアクセスを許可されたプリンシパルの両方についてイベントをトリガーします。詳細については、「Amazon CloudWatch Events ユーザーガイド」を参照してください。
- Amazon CloudWatch Logs では、Amazon EC2 インスタンス、CloudTrail、およびその他のソースからのログファイルをモニタリング、保存、およびアクセスできます。CloudWatch Logs は、ログファイル内の情報をモニタリングし、特定のしきい値が満たされたときに通知します。高い耐久性を備えたストレージにログデータをアーカイブすることも可能です。詳細については、「Amazon CloudWatch Logs ユーザーガイド」を参照してください。
- AWS CloudTrail は、AWSアカウントによって、またはアカウントに代わって行われた API コールおよび関連イベントをキャプチャします。次に、指定した Amazon S3 バケットにログファイルが渡されます。呼び出し元のユーザーとアカウントAWS、呼び出し元のソース IP アドレス、呼び出しの発生日時を特定できます。詳細については、「AWS CloudTrailユーザーガイド」を参照してください。

トピック

- Amazon CloudWatch による DataBrew のモニタリング
- CloudWatch Events による DataBrew の自動化
- CloudWatch Logs による DataBrew のモニタリング

- を使用した DataBrew API コールのログ記録AWS CloudTrail
- AWS Glue Databrew でのAWSユーザー通知の使用

Amazon CloudWatch による DataBrew のモニタリング

CloudWatch を使用して DataBrew CloudWatch は raw データを収集し、読み取り可能なほぼリアルタイムのメトリクスに加工します。これらの統計は 15 か月間保持されるため、履歴情報にアクセスし、ウェブアプリケーションまたはサービスの動作をより的確に把握できます。また、特定のしきい値を監視するアラームを設定し、これらのしきい値に達したときに通知を送信したりアクションを実行したりできます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

AWS Glue DataBrewは、 `AWS/DataBrew`名前空間に次のメトリクスを報告します。

メトリクス	説明
`SessionCount`	お客様のアカウント全体の DataBrew セッションの合計数 有効なディメンション: LogGroupName 有効な統計: Sum 単位: カウント

CloudWatch Events による DataBrew の自動化

Amazon CloudWatch Events を使用すると、AWSサービスを自動化し、アプリケーションの可用性の問題やリソースの変更などのシステムイベントに自動的に対応できます。AWSサービスからのイベントは、ほぼリアルタイムで CloudWatch Events に配信されます。簡単なルールを記述して、注目するイベントと、イベントがルールに一致した場合に自動的に実行するアクションを指定できます。自動的にトリガーできるオペレーションには、以下が含まれます。

- Amazon EC2 Run Command の呼び出し
- Amazon Kinesis Data Streams へのイベントの中継
- AWS Step Functionsステートマシンのアクティブ化
- Amazon SNS トピックまたは Amazon SQS キューの通知

DataBrew は、AWSアカウント内のリソースの状態が変化するたびに CloudWatch Events にイベントを報告します。イベントは、ベストエフォートベースで出力されます。

DataBrew ジョブのさまざまな状態を示すいくつかのイベントの例を次に示します: SUCCEEDED、FAILED、TIMEOUT、および STOPPED。

```
{
  "version": "0",
  "id": "abcdef00-1234-5678-9abc-def012345678",
  "detail-type": "DataBrew Job State Change",
  "source": "aws.databrew",
  "account": "123456789012",
  "time": "2017-09-07T18:57:21Z",
  "region": "us-west-2",
  "resources": [],
  "detail": {
    "jobName": "MyJob",
    "severity": "INFO",
    "state": "SUCCEEDED",
    "jobRunId": "db_abcdef0123456789abcdef0123456789abcdef0123456789abcdef0123456789",
    "message": "Job run succeeded"
  }
}

{
  "version": "0",
  "id": "abcdef01-1234-5678-9abc-def012345678",
  "detail-type": "DataBrew Job State Change",
  "source": "aws.databrew",
  "account": "123456789012",
  "time": "2017-09-07T06:02:03Z",
  "region": "us-west-2",
  "resources": [],
  "detail": {
    "jobName": "MyJob",
    "severity": "ERROR",
    "state": "FAILED",
    "jobRunId": "db_0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef",
    "message": "AnalysisException: 'Path does not exist: s3://MyBucket/MyFile;'"
  }
}
```

```
{
  "version": "0",
  "id": "abcdef00-1234-5678-9abc-def012345678",
  "detail-type": "DataBrew Job State Change",
  "source": "aws.databrew",
  "account": "123456789012",
  "time": "2017-11-20T20:22:06Z",
  "region": "us-east-2",
  "resources": [],
  "detail": {
    "jobName": "MyJob",
    "severity": "WARN",
    "state": "TIMEOUT",
    "jobRunId": "db_abc0123456789abcdef0123456789abcdef0123456789abcdef0123456789def",
    "message": "Job run timed out"
  }
}

{
  "version": "0",
  "id": "abcdef00-1234-5678-9abc-def012345678",
  "detail-type": "DataBrew Job State Change",
  "source": "aws.databrew",
  "account": "123456789012",
  "time": "2017-11-20T20:22:06Z",
  "region": "us-east-2",
  "resources": [],
  "detail": {
    "jobName": "MyJob",
    "severity": "INFO",
    "state": "STOPPED",
    "jobRunId": "db_abc0123456789abcdef0123456789abcdef0123456789abcdef0123456789def",
    "message": "Job run stopped"
  }
}
```

詳細については、「Amazon CloudWatch Events ユーザーガイド」を参照してください。

CloudWatch Logs による DataBrew のモニタリング

CloudWatch Logs を使用して DataBrew ジョブをモニタリングできます。CloudWatch Logs は DataBrew ジョブサブシステムから詳細な情報を収集し、レビューに使用できます。これらのログ

は、プロファイルとレシピジョブが使用しているリソースを把握する場合や、トラブルシューティングの目的で役立ちます。詳細については、Amazon CloudWatch Logs ユーザーガイド」を参照してください。

を使用した DataBrew API コールのログ記録AWS CloudTrail

DataBrew はAWS CloudTrail、DataBrew のユーザー、ロール、または のサービスによって実行されたアクションを記録するAWSサービスである と統合されています。CloudTrail は DataBrew のすべての API コールをイベントとしてキャプチャします。キャプチャされた呼び出しには、DataBrew コンソールからの呼び出しと DataBrew API オペレーションへのコード呼び出しが含まれます。証跡を作成する場合は、DataBrew のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールの [イベント履歴] で最新のイベントを表示できます。CloudTrail によって収集された情報を使用して、DataBrew に対して行われたリクエストを判断できます。リクエストの実行元 IP アドレス、実行者、実行日時、および追加の詳細を判断することもできます。

CloudTrail の詳細については、「AWS CloudTrailユーザーガイド」を参照してください。

CloudTrail の DataBrew 情報

CloudTrail は、AWSアカウントの作成時にアカウントで有効になります。DataBrew でアクティビティが発生すると、そのアクティビティはイベント履歴の他のAWSサービスイベントとともに CloudTrail イベントに記録されます。AWSアカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、「AWS CloudTrailユーザーガイド」の「CloudTrail イベント履歴でのイベントの表示」を参照してください。

DataBrew のイベントなど、AWSアカウントのイベントの継続的な記録については、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成すると、証跡はすべてのAWSリージョンに適用されます。証跡は、AWSパーティション内のすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをさらに分析して処理するように他のAWSサービスを設定できます。詳細については、AWS CloudTrailユーザーガイドで次を参照してください。

- 証跡の作成のための概要
- CloudTrail がサポートするサービスと統合
- CloudTrail 用 Amazon SNS 通知の構成

- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

すべての DataBrew アクションは CloudTrail によってログに記録され、 API リファレンスに記載されています。例えば、`CreateDataset`、`UpdateRecipe`、`StartJobRun` の各アクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。ID 情報は次の判断に役立ちます。

- リクエストが、ルートとユーザー認証情報のどちらを使用して送信されたか。
- リクエストが、ロールとフェデレーションユーザーの一時的なセキュリティ認証情報のどちらを使用して送信されたか。
- リクエストが別のAWSサービスによって行われたかどうか。

詳細については、「CloudTrail userIdentity 要素」を参照してください。

DataBrew ログファイルエントリについて

ここでも、CloudTrail 証跡は、指定した Amazon S3 バケットにイベントをログファイルとして配信できるようにする設定です。CloudTrail のログファイルは、単一か複数のログエントリを含みます。イベントは、任意の出典からの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

`CreateProfileJob` オペレーションを示す CloudTrail ログエントリの例は、次のとおりです。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::1234567890:user/joe",
        "accountId": "1234567890",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "joe"
    },
```

```
    "eventTime": "2020-11-09T18:54:44Z",
    "eventSource": "databrew.amazonaws.com",
    "eventName": "CreateProfileJob",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "requestParameters": {
        "OutputLocation": {
            "Bucket": "bucketName",
            "Key": "keyName"
        },
        "DatasetName": "my-chess-dataset",
        "RoleArn": "arn:aws:iam::1234567890:role/custom-role",
        "Name": "my-profile-job"
    },
    "responseElements": {
        "Name": "my-profile-job"
    },
    "requestID": "993bc3b8-3980-48dd-961e-c1c8529eb248",
    "eventID": "f8128dfa-df29-458b-a2d5-34805b46eefd",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "1234567890"
}
```


AWS Glue Databrew でのAWSユーザー通知の使用

AWSユーザー通知を使用して、AWS Glue Databrew イベントに関する通知を受け取る配信チャネルを設定できます。指定したルールにイベントが一致すると、通知を受け取ります。イベントの通知は、Eメール、チャットアプリケーション内の Amazon Q Developerのチャット通知、AWS Console Mobile Applicationのプッシュ通知などの複数のチャネルで受け取ることができます。「コンソール通知センター」で通知を確認することもできます。AWSユーザー通知は集約をサポートしているため、特定のイベント中に受け取る通知の数を減らすことができます。

レシピステップと関数リファレンス

このリファレンスでは、 またはいずれかのAWS SDKs を使用してプログラムAWS CLIで使用できるレシピステップと関数について説明します。DataBrew では、レシピステップは、raw データをデータパイプラインで使用できる形式に変換するアクションです。DataBrew 関数は、パラメータに基づいて計算を実行する特殊なレシピステップです。

UI での変換のカテゴリは次のとおりです。

- 基本的な列レシピステップ
  - フィルター
  - 列
- データクリーニングレシピのステップ
  - 形式
  - Clean
  - Extract
- データ品質レシピのステップ
  - Missing (見つからない)
  - 無効
  - Duplicates
  - 外れ値
- 個人を特定できる情報 (PII) レシピの手順
  - 個人情報はマスクする
  - 個人情報を交換する
  - 個人情報を暗号化する
  - 行のシャッフル
- 列構造のレシピステップ
  - Split
  - Merge (マージ)
  - 作成
- 列フォーマットレシピのステップ
  - 10 進精度

  - 千の区切り文字
  - 省略番号
- データ構造レシピのステップ
  - ネスト-ネスト解除
  - [Pivot] (ピボット)
  - Group
  - 結合
  - Union
- データサイエンスレシピのステップ
  - テキスト
  - スケール
  - マッピング
  - エンコード
- 関数
  - 数学関数
  - 集計関数
  - テキスト関数
  - 日付および時刻関数
  - Window 関数
  - ウェブ関数
  - その他の関数

これらのレシピステップと関数がレシピでどのように使用されるか (条件式の使用を含む) の詳細については、「」を参照してくださいレシピ構造の定義。

以下のセクションでは、レシピのステップと関数を、その動作別に整理して説明します。

トピック

- 基本的な列レシピステップ
- データクリーニングレシピのステップ
- データ品質レシピのステップ
- 個人を特定できる情報 (PII) レシピの手順

- 外れ値の検出とレシピステップの処理
- 列構造のレシピステップ
- 列フォーマットレシピのステップ
- データ構造レシピのステップ
- データサイエンスレシピのステップ
- 数学関数
- 集計関数
- テキスト関数
- 日付および時刻関数
- Window 関数
- ウェブ関数
- その他の関数

基本的な列レシピステップ

これらの基本的な列レシピアクションを使用して、データに対して簡単な変換を実行します。

トピック

- CHANGE_DATA_TYPE
- DELETE
- 重複
- JSON_TO_STRUCTS
- MOVE_AFTER
- MOVE_BEFORE
- MOVE_TO_END
- MOVE_TO_INDEX
- MOVE_TO_START
- RENAME
- SORT
- TO_BOOLEAN_COLUMN
- TO_DOUBLE_COLUMN

- TO_NUMBER_COLUMN
- TO_STRING_COLUMN

CHANGE_DATA_TYPE

既存の列のデータ型を変更します。

列の値を新しい型に変換できない場合は、NULL に置き換えられます。これは、文字列列が整数列に変換された場合に発生する可能性があります。たとえば、文字列「123」は整数 123 になりますが、文字列「ABC」は数値になることができないため、NULL 値に置き換えられます。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `columnDataType` – 列の新しいタイプ。以下のデータ型 (タイプ) がサポートされています。
  - byte: 1 バイトの符号付き整数。数値の範囲は -128 ~ 127 です。
  - short: 2 バイトの符号付き整数。数値の範囲は -32768 ~ 32767 です。
  - int: 4 バイトの符号付き整数。数値の範囲は -2147483648 ~ 2147483647 です。
  - long: 8 バイトの符号付き整数。数値の範囲は -9223372036854775808 ~ 9223372036854775807 です。
  - float: 4 バイトの単精度浮動小数点数。
  - double: 8 バイトの倍精度浮動小数点数。
  - 小数: 合計 38 桁、小数点以下 18 桁の符号付き小数。
  - string: 文字列値。
  - ブール値: ブール型には、「true」と「false」または「yes」と「no」の 2 つの値のいずれかがあります。
  - timestamp: 年、月、日、時間、分、秒のフィールドで構成される値。
  - date: 年、月、日フィールドで構成される値。

Example例

```
{
    "RecipeAction": {
        "Operation": "CHANGE_DATA_TYPE",
        "Parameters": {
```

```
            "sourceColumn": "columnName",
            "columnDataType": "boolean"
        }
    }
}
```

DELETE

データセットから列を削除します。

パラメータ

- sourceColumn - 既存の列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "DELETE",
        "Parameters": {
            "sourceColumn": "extra_data"
        }
    }
}
```

重複

異なる名前で、すべての同じデータを持つ新しい列を作成します。古い列と新しい列の両方がデータセットに保持されます。

パラメータ

- sourceColumn - 既存の列の名前。
- targetColumn – 重複する列の名前。

Example例

```
{
    "RecipeAction": {
```

```
        "Operation": "DUPLICATE",
        "Parameters": {
            "sourceColumn": "last_name",
            "targetColumn": "copy_of_last_name"
        }
    }
}
```

JSON_TO_STRUCTS

JSON 文字列を静的に型指定された構造体に変換します。変換中、すべての JSON オブジェクトのスキーマを検出し、それらをマージして、JSON 文字列全体を表す最も汎用的なスキーマを取得します。unnestLevel」パラメータは、構造体に変換する JSON オブジェクトのレベルを指定します。

パラメータ

- `sourceColumns` – ソース列のリスト。
- `regexColumnSelector` – 列を選択する正規表現。
- `removeSourceColumn` – ブール値。`true` その場合はソース列を削除し、それ以外の場合は保持します。
- `unnestLevel` – ネスト解除するレベルの数。
- `conditionExpressions` – 条件式。

Example例

```
{
    "RecipeAction": {
        "Operation": "JSON_TO_STRUCTS",
        "Parameters": {
            "sourceColumns": "[\"address\"]",
            "removeSourceColumn": "true",
            "unnestLevel": "2"
       }
    }
}
```

MOVE_AFTER

列を別の列の直後の位置に移動します。

パラメータ

- sourceColumn - 既存の列の名前。
- targetColumn – 別の列の名前。で指定された列は、 で指定された列の直後に移動sourceColumnされますtargetColumn。

Example例

```
{
    "RecipeAction": {
        "Operation": "MOVE_AFTER",
        "Parameters": {
            "sourceColumn": "rating",
            "targetColumn": "height_cm"
        }
    }
}
```

MOVE_BEFORE

列を別の列の直前の位置に移動します。

パラメータ

- sourceColumn - 既存の列の名前。
- targetColumn – 別の列の名前。で指定された列は、 で指定された列の直後に移動sourceColumnされますtargetColumn。

Example例

```
{
    "RecipeAction": {
        "Operation": "MOVE_BEFORE",
        "Parameters": {
            "sourceColumn": "height_cm",
            "targetColumn": "weight_kg"
        }
    }
```

```
}
```

MOVE_TO_END

データセット内の終了位置 (最後の列) に列を移動します。

パラメータ

- sourceColumn - 既存の列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "MOVE_TO_END",
        "Parameters": {
            "sourceColumn": "height_cm"
        }
    }
}
```

MOVE_TO_INDEX

列を数値で指定された位置に移動します。

パラメータ

- sourceColumn - 既存の列の名前。
- targetIndex – 列の新しい位置。位置は 0 で始まるため、例えば、2 番目の列1を参照し、3 番目の列2を参照します。

Example例

```
{
    "RecipeAction": {
        "Operation": "MOVE_TO_INDEX",
        "Parameters": {
            "sourceColumn": "nationality",
```

```
            "targetIndex": "5"
        }
    }
}
```

MOVE_TO_START

列をデータセットの開始位置 (最初の列) に移動します。

パラメータ

- `sourceColumn` - 既存の列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "MOVE_TO_START",
        "Parameters": {
            "sourceColumn": "first_name"
        }
    }
}
```

RENAME

異なる名前で、すべての同じデータを持つ新しい列を作成します。その後、古い列がデータセットから削除されます。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `targetColumn` – 列の新しい名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "RENAME",
```

```
        "Parameters": {
            "sourceColumn": "date_of_birth",
            "targetColumn": "birth_date"
        }
    }
}
```

SORT

データセットの 1 つ以上の列のデータを昇順、降順、またはカスタム順にソートします。

パラメータ

- `expressions` – ソート式を表す 1 つ以上の JSON エンコードされた文字列を含む文字列。
  - `sourceColumn` – 既存の列の名前を含む文字列。
  - `ordering` – 順序は ASCENDING または DESCENDING のいずれかです。
  - `nullsOrdering` — Null の順序は、NULLS_TOP または NULLS_BOTTOM のいずれかで、列の先頭または末尾に null または欠損値を配置できます。
  - `customOrder` – 文字列ソートのカスタム順序を定義する文字列のリスト。デフォルトでは、文字列はアルファベット順にソートされます。
  - `isCustomOrderCaseSensitive` – ブール。デフォルト値は `false` です。

Example例

```
{
    "RecipeAction": {
        "Operation": "SORT",
        "Parameters": {
            "expressions": "[{\"sourceColumn\": \"A\", \"ordering\": \"ASCENDING\",
 \"nullsOrdering\": \"NULLS_TOP\"}]",
       }
    }
 }
```

Exampleカスタムソート順序の例

次の例では、customOrder 式文字列の形式は オブジェクトのリストです。各オブジェクトは、1 つの列のソート式を記述します。

```
[
  {
    "sourceColumn": "A",
    "ordering": "ASCENDING",
    "nullsOrdering": "NULLS_TOP",
  },
  {
    "sourceColumn": "B",
    "ordering": "DESCENDING",
    "nullsOrdering": "NULLS_BOTTOM",
    "customOrder": ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"],
    "isCustomOrderCaseSensitive": false,
  }
]
```

TO_BOOLEAN_COLUMN

既存の列のデータ型を BOOLEAN に変更します。

> **Note**
>
> TO_BOOLEAN_COLUMN ではなく、CHANGE_DATA_TYPE レシピアクションを使用することをお勧めします。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `columnDataType` – にする必要がある値`boolean`。

Example例

```
{
    "RecipeAction": {
        "Operation": "TO_BOOLEAN_COLUMN",
        "Parameters": {
            "columnDataType": "boolean",
            "sourceColumn": "is_present"
        }
```

```
    }
}
```

TO_DOUBLE_COLUMN

既存の列のデータ型を DOUBLE に変更します。

> **Note**
>
> TO_DOUBLE_COLUMN ではなく CHANGE_DATA_TYPE レシピアクションを使用することをお勧めします。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `columnDataType` – にする必要がある値`number`。

Example例

```
{
    "RecipeAction": {
        "Operation": "TO_DOUBLE_COLUMN",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "hourly_rate"
        }
    }
}
```

TO_NUMBER_COLUMN

既存の列のデータ型を NUMBER に変更します。

> **Note**
>
> TO_NUMBER_COLUMN ではなく CHANGE_DATA_TYPE レシピアクションを使用することをお勧めします。

パラメータ

- sourceColumn - 既存の列の名前。
- columnDataType – にする必要がある値number。

Example例

```
{
    "RecipeAction": {
        "Operation": "TO_NUMBER_COLUMN",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "hours_worked"
        }
    }
}
```

TO_STRING_COLUMN

既存の列のデータ型を STRING に変更します。

> **Note**
>
> TO_STRING_COLUMN ではなく、CHANGE_DATA_TYPE レシピアクションを使用することをお勧めします。

パラメータ

- sourceColumn - 既存の列の名前。
- columnDataType – にする必要がある値string。

Example例

```
{
    "RecipeAction": {
```

```
        "Operation": "TO_STRING_COLUMN",
        "Parameters": {
            "columnDataType": "string",
            "sourceColumn": "age"
        }
    }
}
```

データクリーニングレシピのステップ

これらのデータクリーニングレシピステップを使用して、既存のデータに対して簡単な変換を実行します。

トピック

- CAPITAL_CASE
- FORMAT_DATE
- LOWER_CASE
- UPPER_CASE
- SENTENCE_CASE
- ADD_DOUBLE_QUOTES
- ADD_PREFIX
- ADD_SINGLE_QUOTES
- ADD_SUFFIX
- EXTRACT_BETWEEN_DELIMITERS
- EXTRACT_BETWEEN_POSITIONS
- EXTRACT_PATTERN
- EXTRACT_VALUE
- REMOVE_COMBINED
- REPLACE_BETWEEN_DELIMITERS
- REPLACE_BETWEEN_POSITIONS
- REPLACE_TEXT

CAPITAL_CASE

列の各文字列を変更して、各単語を大文字にします。大文字の場合、各単語の最初の文字は大文字になり、残りの単語は小文字に変換されます。例として、クイックブラウンフォックスがフェンスを飛び越えました。

パラメータ

- sourceColumn - 既存の列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "CAPITAL_CASE",
        "Parameters": {
            "sourceColumn": "last_name"
        }
    }
}
```

FORMAT_DATE

日付文字列がフォーマットされた値に変換される列を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- targetDateFormat – 次のいずれかの日付形式。
  - mm/dd/yyyy
  - mm-dd-yyyy
  - dd month yyyy
  - month yyyy
  - dd month

Example例

```
{
    "RecipeAction": {
        "Operation": "FORMAT_DATE",
        "Parameters": {
            "sourceColumn": "birth_date",
            "targetDateFormat": "mm-dd-yyyy"
        }
    }
}
```

LOWER_CASE

列の各文字列を小文字に変更します。たとえば、クイックブラウンキツネがフェンスにジャンプしたなどです。

パラメータ

- `sourceColumn` - 既存の列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "LOWER_CASE",
        "Parameters": {
            "sourceColumn": "nationality"
        }
    }
}
```

UPPER_CASE

列の各文字列を大文字に変更します。例: THE QUICK BROWN Fox JUMPED OVER THE FENCE

パラメータ

- `sourceColumn` - 既存の列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "UPPER_CASE",
        "Parameters": {
            "sourceColumn": "nationality"
        }
    }
}
```

SENTENCE_CASE

列の各文字列を文の大文字と小文字に変更します。文のケースでは、各文の最初の文字が大文字になり、残りの文は小文字に変換されます。例: クイックブラウンキツネ。ジャンプオーバーしました。フェンス

パラメータ

- `sourceColumn` - 既存の列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "SENTENCE_CASE",
        "Parameters": {
            "sourceColumn": "description"
        }
    }
}
```

ADD_DOUBLE_QUOTES

列内の文字を二重引用符で囲みます。

パラメータ

- `sourceColumn` - 既存の列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "ADD_DOUBLE_QUOTES",
        "Parameters": {
            "sourceColumn": "info_url"
        }
    }
}
```

ADD_PREFIX

1 つ以上の文字を追加し、プレフィックスとして列の先頭に連結します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `pattern` – 列値の先頭に配置する文字。

Example例

```
{
    "RecipeAction": {
        "Operation": "ADD_PREFIX",
        "Parameters": {
            "pattern": "aaa",
            "sourceColumn": "info_url"
        }
    }
}
```

ADD_SINGLE_QUOTES

列内の文字を一重引用符で囲みます。

パラメータ

- `sourceColumn` - 既存の列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "ADD_SINGLE_QUOTES",
        "Parameters": {
            "sourceColumn": "info_url"
        }
    }
}
```

ADD_SUFFIX

列の末尾にサフィックスとして連結する文字をもう 1 つ追加します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `pattern` – 列の末尾に配置する文字。

Example例

```
{
    "RecipeAction": {
        "Operation": "ADD_SUFFIX",
        "Parameters": {
            "pattern": "bbb",
            "sourceColumn": "info_url"
        }
    }
}
```

EXTRACT_BETWEEN_DELIMITERS

既存の列の値から、区切り文字に基づいて新しい列を作成します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `targetColumn` – 作成される新しい列の名前。

- `startPattern` – 区切り値を開始する文字を示す正規表現。
- `endPattern` – 区切り文字、または区切り値を終了する文字を示す正規表現。

Example例

```
{
    "RecipeAction": {
        "Operation": "EXTRACT_BETWEEN_DELIMITERS",
        "Parameters": {
            "endPattern": "\\/",
            "sourceColumn": "info_url",
            "startPattern": "\\/\\/",
            "targetColumn": "raw_url"
        }
    }
}
```

EXTRACT_BETWEEN_POSITIONS

既存の列の値から、文字の位置に基づいて新しい列を作成します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `targetColumn` – 作成される新しい列の名前。
- `startPosition` – 抽出を実行する文字の位置。
- `endPosition` – 抽出を終了する文字の位置。

Example例

```
{
    "RecipeAction": {
        "Operation": "EXTRACT_BETWEEN_POSITIONS",
        "Parameters": {
            "endPosition": "9",
            "sourceColumn": "last_name",
            "startPosition": "3",
```

```
            "targetColumn": "characters_3_to_9"
        }
    }
}
```

EXTRACT_PATTERN

既存の列の値から、正規表現に基づいて新しい列を作成します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `targetColumn` – 作成される新しい列の名前。
- `pattern` – 新しい列を抽出して作成する文字を示す正規表現。

Example例

```
{
    "RecipeAction": {
        "Operation": "EXTRACT_PATTERN",
        "Parameters": {
            "pattern": "^....*...$",
            "sourceColumn": "last_name",
            "targetColumn": "first_and_last_few_characters"
        }
    }
}
```

EXTRACT_VALUE

ユーザー指定のパスから抽出された値を持つ新しい列を作成します。ソース列が Map、Array、または Struct 型の場合、パスの各フィールドはバックティック (「name」など) を使用して工スケープする必要があります。

パラメータ

- `targetColumn` – ターゲット列の名前。
- `sourceColumn` – 値が抽出されるソース列の名前。

- path – ユーザーが抽出する特定のキーへのパス。ソース列が Map、Array、または Struct 型の場合、パスの各フィールドはバックティック (「name」など) を使用してエスケープする必要があります。

  ユーザー情報の次の例を考えてみましょう。

```
user {
   name: “Ammy”
   address: {
      state: "CA",
      zipcode: 12345
   },
   phoneNumber:{"home": "123123123", "work": "456456456"}
   citizenship: ["Canada", "USA", "Mexico", "India"]
}
```

  以下は、ソース列のタイプに応じて指定するパスの例です。

  - ソース列がタイプマップの場合、ホーム電話番号を抽出するためのパスは次のとおりです。

    `` `user`.`phoneNumber`.`home` ``
  - ソース列が 型の配列の場合、2 番目の「市民権」値を抽出するためのパスは次のとおりです。

    `` `user`.`citizenship`[1] ``
  - ソース列が struct 型の場合、郵便番号を抽出するためのパスは次のとおりです。

    `` `user`.`address`.`zipcode` ``

Example例

```
{
    "RecipeAction": {
        "Operation": "EXTRACT_VALUE",
        "Parameters": {
            "sourceColumn": "age",
            "targetColumn": "columnName",
            "path": "`age`.`name`",
        }
```

```
    }
}
```

REMOVE_COMBINED

ユーザーが指定した内容に従って、列から 1 つ以上の文字を削除します。

パラメータ

- sourceColumn - 既存の列の名前。
- collapseConsecutiveWhitespace – の場合true、2 つ以上の空白文字を厳密に 1 つの空白文字に置き換えます。
- removeAllPunctuation – の場合true、 は次の文字をすべて削除します。 . ! , ?
- removeAllQuotes – の場合true、すべての一重引用符と二重引用符を削除します。
- removeAllWhitespace – の場合true、すべての空白文字を削除します。
- customCharacters – 実行できる 1 つ以上の文字。
- customValue – アクションを実行できる値。
- removeCustomCharacters – の場合true、 customCharactersパラメータで指定されたすべての文字を削除します。
- removeCustomValue – の場合true、 customValueパラメータで指定されたすべての文字を削除します。
- punctuationally – の場合true、値の開始または終了時に次の文字が発生すると、 によって削除されます。. ! , ?
- antidisestablishmentarianism – の場合true、値の先頭と末尾から一重引用符と二重引用符を削除します。
- removeLeadingAndTrailingWhitespace – の場合true、 は値の先頭と末尾からすべての空白を削除します。
- removeLetters – の場合true、すべての大文字と小文字のアルファベット文字 (A から Z、 aから) を削除しますz。
- removeNumbers – の場合true、すべての数値文字 (0 から) を削除します9。
- removeSpecialCharacters – の場合true、 は次の文字をすべて削除します。 ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ` { | } ~

Example例

```
{
    "RecipeAction": {
        "Operation": "REMOVE_COMBINED",
        "Parameters": {
            "collapseConsecutiveWhitespace": "false",
            "removeAllPunctuation": "false",
            "removeAllQuotes": "false",
            "removeAllWhitespace": "false",
            "removeCustomCharacters": "false",
            "removeCustomValue": "false",
            "removeLeadingAndTrailingPunctuation": "false",
            "removeLeadingAndTrailingQuotes": "false",
            "removeLeadingAndTrailingWhitespace": "false",
            "removeLetters": "false",
            "removeNumbers": "false",
            "removeSpecialCharacters": "true",
            "sourceColumn": "info_url"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REMOVE_COMBINED",
        "Parameters": {
            "collapseConsecutiveWhitespace": "false",
            "customCharacters": "¶",
            "removeAllPunctuation": "false",
            "removeAllQuotes": "false",
            "removeAllWhitespace": "false",
            "removeCustomCharacters": "true",
            "removeCustomValue": "false",
            "removeLeadingAndTrailingPunctuation": "false",
            "removeLeadingAndTrailingQuotes": "false",
            "removeLeadingAndTrailingWhitespace": "false",
            "removeLetters": "false",
            "removeNumbers": "false",
            "removeSpecialCharacters": "false",
            "sourceColumn": "info_url"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REMOVE_COMBINED",
        "Parameters": {
            "collapseConsecutiveWhitespace": "true",
            "customValue": "M",
            "removeAllPunctuation": "true",
            "removeAllQuotes": "false",
            "removeAllWhitespace": "false",
            "removeCustomCharacters": "false",
            "removeCustomValue": "true",
            "removeLeadingAndTrailingPunctuation": "false",
            "removeLeadingAndTrailingQuotes": "true",
            "removeLeadingAndTrailingWhitespace": "true",
            "removeLetters": "true",
            "removeNumbers": "true",
            "removeSpecialCharacters": "false",
            "sourceColumn": "info_url"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REMOVE_COMBINED",
        "Parameters": {
            "collapseConsecutiveWhitespace": "false",
            "removeAllPunctuation": "false",
            "removeAllQuotes": "false",
            "removeAllWhitespace": "false",
            "removeCustomCharacters": "false",
            "removeCustomValue": "false",
            "removeLeadingAndTrailingPunctuation": "false",
            "removeLeadingAndTrailingQuotes": "false",
            "removeLeadingAndTrailingWhitespace": "false",
            "removeLetters": "false",
            "removeNumbers": "true",
            "removeSpecialCharacters": "false",
            "sourceColumn": "first_name"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REMOVE_COMBINED",
        "Parameters": {
            "collapseConsecutiveWhitespace": "false",
            "removeAllPunctuation": "false",
            "removeAllQuotes": "false",
            "removeAllWhitespace": "false",
            "removeCustomCharacters": "false",
            "removeCustomValue": "false",
            "removeLeadingAndTrailingPunctuation": "false",
            "removeLeadingAndTrailingQuotes": "false",
            "removeLeadingAndTrailingWhitespace": "false",
            "removeLetters": "false",
            "removeNumbers": "true",
            "removeSpecialCharacters": "false",
            "sourceColumn": "first_name"
        }
    }
}
```

REPLACE_BETWEEN_DELIMITERS

2 つの区切り文字の文字をユーザーが指定したテキストに置き換えます。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `startPattern` – 置換を開始する場所を示す文字または文字、または正規表現。
- `endPattern` – 置換が終了する場所を示す文字または文字、または正規表現。
- `value` – 置き換える文字。

Example例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_BETWEEN_DELIMITERS",
        "Parameters": {
            "endPattern": ">",
```

```
            "sourceColumn": "last_name",
            "startPattern": "&lt;",
            "value": "?"
        }
    }
}
```

REPLACE_BETWEEN_POSITIONS

2 つの位置の文字をユーザーが指定したテキストに置き換えます。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `startPosition` – 置換を開始する文字列内の文字位置を示す数値。
- `endPosition` – 置換が終了する文字列内の文字位置を示す数値。
- `value` – 置き換える文字。

Example例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_BETWEEN_POSITIONS",
        "Parameters": {
            "endPosition": "20",
            "sourceColumn": "nationality",
            "startPosition": "10",
            "value": "E"
        }
    }
}
```

REPLACE_TEXT

指定された文字シーケンスを別の文字に置き換えます。

パラメータ

- `sourceColumn` - 既存の列の名前。

- pattern – ソース列で置き換える文字を示す文字または文字、または正規表現。
- value – 置き換える文字。

Example例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_TEXT",
        "Parameters": {
            "pattern": "x",
            "sourceColumn": "first_name",
            "value": "a"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REPLACE_TEXT",
        "Parameters": {
            "pattern": "[0-9]",
            "sourceColumn": "nationality",
            "value": "!"
        }
    }
}
```

データ品質レシピのステップ

これらのデータ品質レシピステップを使用して、欠損値の入力、無効なデータの削除、重複の削除を行います。

トピック

- ADVANCED_DATATYPE_FILTER
- ADVANCED_DATATYPE_FLAG
- DELETE_DUPLICATE_ROWS
- EXTRACT_ADVANCED_DATATYPE_DETAILS

- FILL_WITH_AVERAGE
- FILL_WITH_CUSTOM
- FILL_WITH_EMPTY
- FILL_WITH_LAST_VALID
- FILL_WITH_MEDIAN
- FILL_WITH_MODE
- FILL_WITH_MOST_FREQUENT
- FILL_WITH_NULL
- FILL_WITH_SUM
- FLAG_DUPLICATE_ROWS
- FLAG_DUPLICATES_IN_COLUMN
- GET_ADVANCED_DATATYPE
- REMOVE_DUPLICATES
- REMOVE_INVALID
- REMOVE_MISSING
- REPLACE_WITH_AVERAGE
- REPLACE_WITH_CUSTOM
- REPLACE_WITH_EMPTY
- REPLACE_WITH_LAST_VALID
- REPLACE_WITH_MEDIAN
- REPLACE_WITH_MODE
- REPLACE_WITH_MOST_FREQUENT
- REPLACE_WITH_NULL
- REPLACE_WITH_ROLLING_AVERAGE
- REPLACE_WITH_ROLLING_SUM
- REPLACE_WITH_SUM

ADVANCED_DATATYPE_FILTER

高度なデータ型検出に基づいて現在のソース列をフィルタリングします。例えば、DataBrew が郵便番号を含むと識別した列がある場合、この変換はタイムゾーンに基づいて列をフィルタリングできま

す。抽出できる詳細は、以下の注意事項で説明されているように、検出されたパターンによって異なります。

パラメータ

- `sourceColumn` – 文字列ソース列の名前。
- `pattern` – 抽出するパターン。
- `advancedDataType` – 電話番号、郵便番号、日時、州、クレジットカード、URL、E メール、SSN、性別のいずれかを指定できます。
- `filter values` – ユーザーが列をフィルタリングする文字列値のリスト。
- `strategy` – KEEP_ROWS または DISCARD_ROWS または CLEAR_FILTERS または CLEAR_OTHERS。
- `clearWithEmpty` – ブール値`true`または `false`。 `empty` の代わりに を使用して行をクリアします`null`。

注意事項

- advancedDataType が Phone の場合、パターンは AREA_CODE、TIME_ZONE、COUNTRY_CODE のいずれかになります。
- advancedDataType が郵便番号の場合、パターンは TIME_ZONE、COUNTRY、STATE、CITY、TYPE、REGION のいずれかになります。
- advancedDataType が日時の場合、パターンは DAY、MONTH、MONTH_NAME、WEEK、QUARTER、YEAR のいずれかになります。
- advancedDataType が State の場合、パターンは TIME_ZONE になります。
- advancedDataType がクレジットカードの場合、パターンは LENGTH または NETWORK にすることができます。
- advancedDataType が URL の場合、パターンは PROTOCOL、TLD、DOMAIN のいずれかになります。

Example例

```
{
    "RecipeAction": {
        "Operation": "ADVANCED_DATATYPE_FILTER",
        "Parameters": {
```

```
            "pattern": "AREA_CODE",
            "sourceColumn": "phoneColumn",
            "advancedDataType": "Phone",
            "filterValues": ['Ohio'],
            "strategy": "KEEP_ROWS"
        }
    }
}
```

ADVANCED_DATATYPE_FLAG

現在のソース列の値に基づいて新しいフラグ列を作成します。たとえば、郵便番号を含むソース列がある場合、この変換を使用して、特定のタイムゾーンfalseに基づいて trueまたは として値にフラグを付けることができます。抽出できる詳細は、以下の注意事項で説明されているように、検出されたパターンによって異なります。

パラメータ

- sourceColumn – 文字列ソース列の名前。
- pattern – 抽出するパターン。
- targetColumn – ターゲット列の名前。
- advancedDataType – 電話番号、郵便番号、日時、州、クレジットカード、URL、E メール、SSN、性別のいずれかを指定できます。
- filter values – ユーザーが列をフィルタリングする文字列値のリスト。
- trueString – ターゲット列trueの値。
- falseString – ターゲット列falseの値。

注意事項

- advancedDataType が Phone の場合、パターンは AREA_CODE、TIME_ZONE、COUNTRY_CODE のいずれかになります。
- advancedDataType が郵便番号の場合、パターンは TIME_ZONE、COUNTRY、STATE、CITY、TYPE、REGION のいずれかになります。
- advancedDataType が日時の場合、パターンは DAY、MONTH、MONTH_NAME、WEEK、QUARTER、YEAR のいずれかになります。
- advancedDataType が State の場合、パターンは TIME_ZONE になります。

- advancedDataType がクレジットカードの場合、パターンは LENGTH または NETWORK にすることができます。
- advancedDataType が URL の場合、パターンは PROTOCOL、TLD、DOMAIN のいずれかになります。

Example例

```
{
    "RecipeAction": {
        "Operation": "ADVANCED_DATATYPE_FLAG",
        "Parameters": {
            "pattern": "AREA_CODE",
            "sourceColumn": "phoneColumn",
            "advancedDataType": "Phone",
            "filterValues": ['Ohio'],
            "targetColumn": "targetColumnName",
            "trueString": "trueValue",
            "falseString": "falseValue"
        }
    }
}
```

DELETE_DUPLICATE_ROWS

データセット内の以前の行と完全に一致する行を削除します。最初の出現は、前の行と一致しないため、削除されません。

Example例

```
{
    "RecipeAction": {
        "Operation": "DELETE_DUPLICATE_ROWS"
    }
}
```

EXTRACT_ADVANCED_DATATYPE_DETAILS

高度なデータ型の詳細を抽出します。抽出できる詳細は、以下の注意事項で説明されているように、検出されたパターンによって異なります。

パラメータ

- sourceColumn – 文字列ソース列の名前。
- pattern – 抽出するパターン。
- targetColumn – ターゲット列の名前。
- advancedDataType – 電話番号、郵便番号、日時、州、クレジットカード、URL、E メール、SSN、性別のいずれかを指定できます。

注意事項

- advancedDataType が Phone の場合、パターンは AREA_CODE、TIME_ZONE、または COUNTRY_CODE のいずれかになります。
- advancedDataType が郵便番号の場合、パターンは TIME_ZONE、COUNTRY、STATE、CITY、TYPE、REGION のいずれかになります。
- advancedDataType が日時の場合、パターンは DAY、MONTH、MONTH_NAME、WEEK、QUARTER、YEAR のいずれかになります。
- advancedDataType が State の場合、パターンは TIME_ZONE になります。
- advancedDataType がクレジットカードの場合、パターンは LENGTH または NETWORK にすることができます。
- advancedDataType が URL の場合、パターンは PROTOCOL、TLD、DOMAIN のいずれかになります。

Example例

```
{
    "RecipeAction": {
        "Operation": "EXTRACT_ADVANCED_DATATYPE_DETAILS",
        "Parameters": {
            "pattern": "TIMEZONE"
            "sourceColumn": "zipCode",
            "targetColumn": "timeZoneFromZipCode",
            "advancedDataType": "ZipCode"
        }
    }
}
```

FILL_WITH_AVERAGE

欠落データをすべての値の平均で置き換えた列を返します。

パラメータ

- sourceColumn - 既存の列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_AVERAGE",
        "Parameters": {
            "sourceColumn": "age"
        }
    }
}
```

FILL_WITH_CUSTOM

欠落データを特定の値に置き換えた列を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- columnDataType – 列のデータ型。このタイプは、date、number、、booleanunsupported、string、または である必要がありますtimestamp。
- value – 入力するカスタム値。データ型は、 に選択した値と一致する必要がありますcolumnDataType。

Example例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_CUSTOM",
        "Parameters": {
```

```
            "columnDataType": "string",
            "sourceColumn": "last_name",
            "value": "No last name provided"
        }
    }
}
```

FILL_WITH_EMPTY

欠落データを空の文字列に置き換えた列を返します。

パラメータ

- sourceColumn - 既存の列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_EMPTY",
        "Parameters": {
            "sourceColumn": "wind_direction"
        }
    }
}
```

FILL_WITH_LAST_VALID

欠落データをその列の最新の有効な値に置き換えた列を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- columnDataType – 列のデータ型。このタイプは、date、number、、booleanunsupported、string、または である必要がありますtimestamp。

Example例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_LAST_VALID",
        "Parameters": {
            "columnDataType": "string",
            "sourceColumn": "birth_date"
        }
    }
}
```

FILL_WITH_MEDIAN

欠落データをすべての値の中央値に置き換えた列を返します。

パラメータ

- sourceColumn - 既存の列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_MEDIAN",
        "Parameters": {
            "sourceColumn": "age"
        }
    }
}
```

FILL_WITH_MODE

欠落データをすべての値のモードに置き換えた列を返します。

値の一部が同じ場合、タイブレーカーロジックを指定することもできます。例えば、次の値を考えてみましょう。

1 2 2 3 3 4

modeType の MINIMUMはFILL_WITH_MODE、モード値として 2 を返します。modeType が の場合MAXIMUM、モードは 3 です。AVERAGE の場合、モードは 2.5 です。

パラメータ

- sourceColumn - 既存の列の名前。
- modeType – データ内のタイ値を解決する方法。この値は、MINIMUM、NONE、、AVERAGEまたはである必要がありますMAXIMUM。

Example例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_MODE",
        "Parameters": {
            "modeType": "MAXIMUM",
            "sourceColumn": "age"
        }
    }
}
```

FILL_WITH_MOST_FREQUENT

欠落データを最も頻繁な値に置き換えた列を返します。

パラメータ

- sourceColumn - 既存の列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_MOST_FREQUENT",
        "Parameters": {
            "sourceColumn": "position"
        }
    }
}
```

FILL_WITH_NULL

データ値が null で置き換えられた列を返します。

パラメータ

- sourceColumn - 既存の列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_NULL",
        "Parameters": {
            "sourceColumn": "rating"
        }
    }
}
```

FILL_WITH_SUM

欠落データをすべての値の合計に置き換えた列を返します。

パラメータ

- sourceColumn - 既存の列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_SUM",
        "Parameters": {
            "sourceColumn": "age"
        }
    }
}
```

FLAG_DUPLICATE_ROWS

各行に指定された値を持つ新しい列を返します。この列は、その行がデータセット内の以前の行と完全に一致しているかどうかを示します。一致が見つかった場合、重複としてフラグが付けられます。以前の行と一致しないため、最初の出現にはフラグが付けられません。

パラメータ

- `trueString` – 行が前の行と一致する場合に挿入される値。
- `falseString` – 行が一意である場合に挿入される値。
- `targetColumn` – データセットに挿入された新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "FLAG_DUPLICATE_ROWS",
        "Parameters": {
            "trueString": "TRUE",
            "falseString": "FALSE",
            "targetColumn": "Flag"
        }
    }
}
```

FLAG_DUPLICATES_IN_COLUMN

各行に指定された値を持つ新しい列を返します。この列は、行のソース列の値がソース列の前の行の値と一致するかどうかを示します。一致が見つかった場合、重複としてフラグが付けられます。以前の行と一致しないため、最初の出現にはフラグが付けられません。

パラメータ

- `sourceColumn` – ソース列の名前。
- `targetColumn` – ターゲット列の名前。
- `trueString` – ソース列の値がその列の以前の値と重複する場合に、ターゲット列に挿入される文字列。

- `falseString` – ソース列の値がその列の以前の値と異なる場合に、ターゲット列に挿入される文字列。

Example例

```
{
    "RecipeAction": {
        "Operation": "FLAG_DUPLICATES_IN_COLUMN",
        "Parameters": {
            "sourceColumn": "Name",
            "targetColumn": "Duplicate",
            "trueString": "TRUE",
            "falseString": "FALSE"
        }
    }
}
```

GET_ADVANCED_DATATYPE

文字列列を指定した場合、 は列の高度なデータ型を識別します。

パラメータ

- `columnName` – 文字列列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "GET_ADVANCED_DATATYPE",
        "Parameters": {
            "sourceColumn": "columnName"
        }
    }
}
```

REMOVE_DUPLICATES

選択したソース列で重複した値が発生した場合は、行全体を削除します。

パラメータ

- sourceColumn - 既存の列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "REMOVE_DUPLICATES",
        "Parameters": {
            "sourceColumn": "nationality"
        }
    }
}
```

REMOVE_INVALID

その行の列で無効な値が発生した場合は、行全体を削除します。

パラメータ

- sourceColumn - 既存の列の名前。
- columnDataType – 列のデータ型。
- advancedDataType – データ型が の列で DataBrew によって検出された特殊なデータ型string。DataBrew がstring列内で検出できるタイプには、SSN、E メール、電話番号、性別、クレジットカード、URL、IP アドレス、DateTime、通貨、ZipCode、国、リージョン、州、市が含まれます。

Example例

```
{
    "RecipeAction": {
        "Operation": "REMOVE_INVALID",
        "Parameters": {
            "columnDataType": "string",
            "sourceColumn": "help_url"
        }
    }
```

```
}
```

REMOVE_MISSING

指定された列にデータが欠落していない行のみを返します。

パラメータ

- sourceColumn - 既存の列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "REMOVE_MISSING",
        "Parameters": {
            "sourceColumn": "last_name"
        }
    }
}
```

REPLACE_WITH_AVERAGE

列の各無効な値を、他のすべての値の平均に置き換えます。

パラメータ

- sourceColumn - 既存の列の名前。
- columnDataType – 列のデータ型。このタイプは である必要がありますnumber。

Example例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_AVERAGE",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "age"
        }
```

```
    }
}
```

REPLACE_WITH_CUSTOM

検出されたエンティティをカスタム値に置き換えます。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `sourceColumns` – 既存の列名のリスト。
- `columnDataType` – 列のデータ型。
- `value` – 無効な値を置き換えるために使用されるカスタム値。
- `advancedDataType` – データ型が の列で DataBrew によって検出された特殊なデータ型`string`。DataBrew が`string`列内で検出できるタイプには、SSN、E メール、電話番号、性別、クレジットカード、URL、IP アドレス、DateTime、通貨、ZipCode、国、リージョン、州、市が含まれます。

> **Note**
>
> `sourceColumn` または のいずれかを使用しますが`sourceColumns`、両方は使用しません。

Example例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_CUSTOM",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "",
            "sourceColumns": ["column1", "column2"],
            "value": 0
        }
    }
}
```

REPLACE_WITH_EMPTY

列の各無効な値を空の値に置き換えます。

パラメータ

- sourceColumn - 既存の列の名前。
- columnDataType – 列のデータ型。
- advancedDataType – データ型が の列で DataBrew によって検出された特殊なデータ型string。DataBrew がstring列内で検出できるタイプには、SSN、E メール、電話番号、性別、クレジットカード、URL、IP アドレス、DateTime、通貨、ZipCode、国、リージョン、州、市が含まれます。

Example例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_EMPTY",
        "Parameters": {
            "columnDataType": "string",
            "sourceColumn": "nationality"
        }
    }
}
```

REPLACE_WITH_LAST_VALID

列の各無効な値を最後の有効な値に置き換えます。

パラメータ

- sourceColumn - 既存の列の名前。
- columnDataType – 列のデータ型。
- advancedDataType – データ型が の列で DataBrew によって検出された特殊なデータ型string。DataBrew がstring列内で検出できるタイプには、SSN、E メール、電話番号、性別、クレジットカード、URL、IP アドレス、DateTime、通貨、ZipCode、国、リージョン、州、市が含まれます。

Example例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_LAST_VALID",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "rating"
        }
    }
}
```

REPLACE_WITH_MEDIAN

列の各無効な値を、他のすべての値の中央値に置き換えます。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `columnDataType` – 列のデータ型。このタイプは である必要がありますnumber。

Example例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_MEDIAN",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "games_won"
        }
    }
}
```

REPLACE_WITH_MODE

列の各無効な値を、他のすべての値のモードに置き換えます。

パラメータ

- `sourceColumn` - 既存の列の名前。

- columnDataType – 列のデータ型。このタイプは である必要がありますnumber。
- modeType – データ内のタイ値を解決する方法。この値は、MINIMUM、NONE、、AVERAGEまたは である必要がありますMAXIMUM。

Example例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_MODE",
        "Parameters": {
            "columnDataType": "number",
            "modeType": "MAXIMUM",
            "sourceColumn": "height_cm"
        }
    }
}
```

REPLACE_WITH_MOST_FREQUENT

列の各無効な値を最も頻繁な列値に置き換えます。

パラメータ

- sourceColumn - 既存の列の名前。
- columnDataType – 列のデータ型。
- advancedDataType – データ型が の列で DataBrew によって検出された特殊なデータ型string。DataBrew がstring列内で検出できるタイプには、SSN、E メール、電話番号、性別、クレジットカード、URL、IP アドレス、DateTime、通貨、ZipCode、国、リージョン、州、市が含まれます。

Example例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_MOST_FREQUENT",
        "Parameters": {
            "columnDataType": "string",
```

```
            "sourceColumn": "wind_direction"
        }
    }
}
```

REPLACE_WITH_NULL

列の各無効な値を null 値に置き換えます。

パラメータ

- sourceColumn - 既存の列の名前。
- columnDataType – 列のデータ型。
- advancedDataType – データ型が の列で DataBrew によって検出された特殊なデータ型string。DataBrew がstring列内で検出できるタイプには、SSN、E メール、電話番号、性別、クレジットカード、URL、IP アドレス、DateTime、通貨、ZipCode、国、リージョン、州、市が含まれます。

Example例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_NULL",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "weight_kg"
        }
    }
}
```

REPLACE_WITH_ROLLING_AVERAGE

列の各値を、前の行の「ウィンドウ」からのローリング平均に置き換えます。

パラメータ

- sourceColumn - 既存の列の名前。
- columnDataType – 列のデータ型。このタイプは である必要がありますnumber。

- period - – ウィンドウのサイズ。たとえば、 periodが 10 の場合、ローリング平均は前の 10 行を使用して計算されます。

Example例

```
{
    "RecipeStep": {
        "Action": {
            "Operation": "REPLACE_WITH_ROLLING_AVERAGE",
            "Parameters": {
                "sourceColumn": "created_at",
                "columnDataType": "number",
                "period": "2"
            }
        }
    }
}
```

REPLACE_WITH_ROLLING_SUM

列の各値を、前の「ウィンドウ」の行のローリング合計に置き換えます。

パラメータ

- sourceColumn - 既存の列の名前。
- columnDataType – 列のデータ型。このタイプは である必要がありますnumber。
- period - – ウィンドウのサイズ。たとえば、 periodが 10 の場合、ローリング合計は前の 10 行を使用して計算されます。

Example例

```
{
    "RecipeStep": {
        "Action": {
            "Operation": "REPLACE_WITH_ROLLING_SUM",
            "Parameters": {
                "sourceColumn": "created_at",
                "columnDataType": "number",
```

```
                "period": "2"
            }
        }
    }
}
```

REPLACE_WITH_SUM

列の各無効な値を、他のすべての値の合計に置き換えます。

パラメータ

- sourceColumn - 既存の列の名前。
- columnDataType – 列のデータ型。このタイプは である必要がありますnumber。

Example例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_SUM",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "games_won"
        }
    }
}
```

個人を特定できる情報 (PII) レシピの手順

これらのレシピステップを使用して、データセット内の個人を特定できる情報 (PII) の変換を実行します。

> **Note**
>
> このセクションのレシピステップに加えて、PII の処理に使用できる PII 専用に設計されていない DataBrew レシピステップがあります。例としてはDELETE、列を削除する基本的な列レシピステップである があります。

トピック

CRYPTOGRAPHIC_HASH

列のハッシュ値にアルゴリズムを適用します。

パラメータ

- `sourceColumns` – 既存の列の配列。
- `secretId` – Secrets Manager シークレットキーの ARN。ソース列をハッシュ化するためにハッシュベースのメッセージ認証コード (HMAC) プレフィックスアルゴリズムで使用されるキー、または Secrets Manager シークレットキーの値の base64 デコード出力`databrew!default`です。
- `secretVersion` - オプション。デフォルトは最新のシークレットバージョンです。
- `entityTypeFilter` – エンティティタイプのオプションの配列。検出された PII のみをフリーテキスト列で暗号化するために使用できます。
- `createSecretIfMissing` – オプションのブール値。true の場合、呼び出し元に代わってシークレットの作成を試みます。
- `algorithm` – データをハッシュするために使用されるアルゴリズム。有効な列挙値: MD5、SHA1、SHA256、SHA512、HMAC_MD5、HMAC_SHA1、HMAC_SHA256、HMAC_SHA512

  各オプションは、異なるハッシュアルゴリズムを参照します。「HMAC」プレフィックスを持つこれらのオプションは、キー付きハッシュアルゴリズムを参照し、 `secretId`パラメータを必要と

します。「HMAC」プレフィックスのないオプションの場合、 secretIdパラメータは必要ありません。

ハッシュアルゴリズムを指定しない場合、サービスはデフォルトで「HMAC_SHA256」になります。

```
{
   "sourceColumns": ["phonenumber"],
   "secretId": "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret",
   "entityTypeFilter": ["USA_ALL"]
}
```

インタラクティブエクスペリエンスで作業する場合、コンソールユーザーには、プロジェクトのロールに加えて、提供された Secrets Manager シークレットsecretsmanager:GetSecretValueに対する へのアクセス許可が必要です。

サンプルポリシー:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret"
            ]
        }
    ]
}
```

また、secretId databrew!defaultとして、パラメータを true createSecretIfMissing として渡すことで、DataBrew が作成したデフォルトのシークレットを使用することもできます。これは本番環境では推奨されません。AwsGlueDataBrewFullAccessPolicy ロールを持つユーザーは、デフォルトのシークレットを使用できます。

復号

DECRYPT 変換を使用して DataBrew 内で復号できます。Encryption SDK を使用してDataBrew AWSの外部でデータを復号することもできます。提供された KMS キー ARN が列の暗号化に使用されたものと一致しない場合、復号オペレーションは失敗します。Encryption SDK の詳細については、AWSAWS Encryption SDKデベロッパーガイドのAWS「暗号化 SDK とは」を参照してください。

パラメータ

- `sourceColumns` – 既存の列の配列。
- `kmsKeyArn` – ソース列の復号に使用するAWS Key Management Service キーのキー ARN。キー ARN の詳細については、「 AWS Key Management Serviceデベロッパーガイド」の「キー ARN」を参照してください。

```
{
   "sourceColumns": ["phonenumber"],
   "kmsKeyArn": "arn:aws:kms:us-east-1:012345678901:key/<kms-key-id>"
}
```

インタラクティブエクスペリエンスで作業する場合、プロジェクトの役割に加えて、コンソールユーザーには、提供された KMS キーに対する `kms:GenerateDataKey`および `kms:Decrypt`へのアクセス許可が必要です。

サンプルポリシー:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt"
        ],
        "Resource": [
            "arn:aws:kms:us-east-1:012345678901:key/kms-key-id"
```

```
        ]
      }
    ]
}
```

DETERMINISTIC_DECRYPT

DETERMINISTIC_ENCRYPT で暗号化されたデータを復号します。

指定されたシークレット ID とバージョンが列の暗号化に使用されたものと一致しない場合、この変換は no-op です。

パラメータ

- `sourceColumns` – 既存の列の配列。
- `secretId` – ソース列の復号に使用する Secrets Manager シークレットキーの ARN。
- `secretVersion` - オプション。デフォルトは最新のシークレットバージョンです。

例

```
{
   "sourceColumns": ["phonenumber"],
   "secretId": "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret",
   "secretVersion": "adfe-1232-7563-3123"
}
```

インタラクティブエクスペリエンスで作業する場合、コンソールユーザーには、プロジェクトのロールに加えて、提供された Secrets Manager シークレットの secretsmanager:GetSecretValue に対するアクセス許可が必要です。

サンプルポリシー:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret"
            ]
        }
    ]
}
```

DETERMINISTIC_ENCRYPT

AES-GCM-SIV と 256 ビットキーを使用して列を暗号化します。DETERMINISTIC_ENCRYPT で暗号化されたデータは、DETERMINISTIC_DECRYPT 変換を使用して DataBrew 内でのみ復号できます。この変換では、AWS KMSまたはAWS Encryption SDK は使用されず、代わりに AWS LC github ライブラリを使用します。

セルあたり最大 400KB を暗号化できます。復号時にデータ型を保持しません。

> **Note**
>
> 注: シークレットを 1 年以上使用することはお勧めしません。

パラメータ

- `sourceColumns` – 既存の列の配列。
- `secretId` – ソース列の暗号化に使用する Secrets Manager シークレットキーの ARN、または databrew!デフォルト。
- `secretVersion` - オプション。デフォルトは最新のシークレットバージョンです。
- `entityTypeFilter` – エンティティタイプのオプションの配列。検出された PII のみをフリーテキスト列で暗号化するために使用できます。
- `createSecretIfMissing` – オプションのブール値。true の場合、呼び出し元に代わってシークレットの作成を試みます。

例

```
{
   "sourceColumns": ["phonenumber"],
   "secretId": "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret",
   "secretVersion": "adfe-1232-7563-3123",
   "entityTypeFilter": ["USA_ALL"]
}
```

インタラクティブエクスペリエンスで作業する場合、コンソールユーザーには、プロジェクトのロールに加えて、提供された Secrets Manager シークレットsecretsmanager:GetSecretValueに対する へのアクセス許可が必要です。

サンプルポリシー

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret"
            ]
        }
    ]
}
```

暗号化

AWS Encryption SDK を使用してソース列の値を暗号化します。DECRYPT 変換は、DataBrew 内で復号化するために使用できます。Encryption SDK を使用して DataBrew AWSの外部でデータを復号することもできます。

ENCRYPT 変換は、セルあたり最大 128 MiB を暗号化できます。復号時にフォーマットを保持しようとします。データ型を保持するには、データ型メタデータを 1KB 未満にシリアル化する必要があります。それ以外の場合は、preserveDataType パラメータを false に設定する必要があります。

データ型メタデータは、暗号化コンテキストのプレーンテキストで保存されます。暗号化コンテキストの詳細については、「 AWS Key Management Serviceデベロッパーガイド」の「暗号化コンテキスト」を参照してください。

パラメータ

- `sourceColumns` – 既存の列の配列。
- `kmsKeyArn` – ソース列の暗号化に使用するAWS Key Management Service キーのキー ARN。キー ARN の詳細については、「 AWS Key Management Serviceデベロッパーガイド」の「キー ARN」を参照してください。
- `entityTypeFilter` – エンティティタイプのオプションの配列。検出された PII のみをフリーテキスト列で暗号化するために使用できます。
- `preserveDataType` – オプションのブール値。デフォルトは true です。false の場合、データ型は保存されません。

次の例では、 `entityTypeFilter`と `preserveDataType`はオプションです。

例

```
{
    "sourceColumns": ["phonenumber"],
    "kmsKeyArn": "arn:aws:kms:us-east-1:012345678901:key/kms-key-id",
    "entityTypeFilter": ["USA_ALL"],
    "preserveDataType": "true"
}
```

インタラクティブエクスペリエンスで作業する場合、プロジェクトの役割に加えて、コンソールユーザーには、指定されたAWS KMSキー`kms:GenerateDataKey`に対する へのアクセス許可が必要です。

サンプルポリシー:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:012345678901:key/kms-key-id"
            ]
        }
    ]
}
```

MASK_CUSTOM

指定されたカスタム値に一致する文字をマスクします。

パラメータ

- sourceColumns – 既存の列名のリスト。
- maskSymbol – 指定された文字を置き換えるために使用される記号。
- regex – true の場合、 は一致する正規表現パターンcustomValueとして扱います。
- customValue – のすべての出現 (または正規表現一致) customValueは文字列でマスクされます。
- entityTypeFilter – エンティティタイプのオプションの配列。検出された PII のみをフリーテキスト列で暗号化するために使用できます。

Example例

```
// Mask all occurrences of 'amazon' in the column
{
    "RecipeAction": {
        "Operation": "MASK_CUSTOM",
        "Parameters": {
            "sourceColumns": ["company"],
            "maskSymbol": "#",
            "customValue": "amazon"
        }
    }
}
```

MASK_DATE

ユーザーが指定したマスク記号を使用して日付のコンポーネントをマスクします。

パラメータ

- sourceColumns – 既存の列名のリスト。
- maskSymbol – 指定された文字を置き換えるために使用される記号。
- redact – マスクする日付コンポーネントの列挙型の配列。有効な列挙値: YEAR、MONTH、DAY、HOUR、MINUTE、SECOND、MILLISECOND。
- locale – オプションの IETF BCP 47 言語タグ。デフォルトは en です。日付の書式設定に使用するロケール。

Example例

```
// Mask year
{
    "RecipeAction": {
        "Operation": "MASK_DATE",
        "Parameters": {
            "sourceColumns": ["birthday"],
            "maskSymbol": "#",
            "redact": ["YEAR"]
        }
    }
}
```

MASK_DELIMITER

ユーザーが指定したマスキング記号を持つ 2 つの区切り文字の間の文字をマスクします。

パラメータ

- sourceColumns – 既存の列名のリスト。
- maskSymbol – 指定された文字を置き換えるために使用される記号。
- startDelimiter – マスキングを開始する場所を示す文字。このパラメータを省略すると、文字列の先頭からマスクが適用されます。

- `endDelimiter` – マスキングを終了する場所を示す文字。このパラメータを省略すると、startDelimiter から文字列の末尾にマスキングが適用されます。
- `preserveDelimiters` – true の場合、 は区切り文字にマスクを適用します。
- `alphabet` – マスキング中に保持する文字セットの配列。有効な列挙値: SYMBOLS、WHITESPACE。
- `entityTypeFilter` – エンティティタイプのオプションの配列。検出された PII のみをフリーテキスト列で暗号化するために使用できます。

Example例

```
// Mask string between '<' and '>', ignoring white spaces, symbols, and lowercase
 letters
{
    "RecipeAction": {
        "Operation": "MASK_DELIMITER",
        "Parameters": {
            "sourceColumns": ["name"],
            "maskSymbol": "#",
            "startDelimiter": "<",
            "endDelimiter": ">",
            "preserveDelimiters": false,
            "alphabet": ["WHITESPACE", "SYMBOLS"]
        }
    }
}
```

MASK_RANGE

ユーザーが指定したマスキング記号を使用して、2 つの位置間の文字をマスクします。

パラメータ

- `sourceColumns` – 既存の列名のリスト。
- `maskSymbol` – 指定された文字を置き換えるために使用される記号。
- `start` – マスキングを開始する文字の位置を示す数値 (0 インデックス付き、両端を含む) 。負のインデックス作成は許可されます。このパラメータを省略すると、文字列の先頭から「停止」までマスクが適用されます。

- stop – マスキングを終了する文字位置を示す数値 (0 インデックス付き、排他的) 。負のインデックス作成は許可されます。このパラメータを省略すると、「start」から文字列の末尾までマスクが適用されます。
- alphabet – マスキング中に保持する文字セットの列挙型の配列。有効な列挙値: SYMBOLS、WHITESPACE。
- entityTypeFilter – エンティティタイプのオプションの配列。検出された PII のみをフリーテキスト列で暗号化するために使用できます。

Example例

```
// Mask entire string
{
    "RecipeAction": {
        "Operation": "MASK_RANGE",
        "Parameters": {
            "sourceColumns": ["firstName", "lastName"],
            "maskSymbol": "#"
        }
    }
}
```

REPLACE_WITH_RANDOM_BETWEEN

値を乱数に置き換えます。

パラメータ

- lowerBound – 乱数範囲の下限。
- sourceColumns – 既存の列名のリスト。
- upperBound – 乱数範囲の上限。

Example例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_RANDOM_BETWEEN",
        "Parameters": {
```

```
            "lowerBound": "1",
            "sourceColumns": ["column1", "column2"],
            "upperBound": "100"
        }
    }
}
```

REPLACE_WITH_RANDOM_DATE_BETWEEN

値をランダムな日付に置き換えます。

パラメータ

- `startDate` – ランダムな日付を取得する日付の範囲の開始。
- `sourceColumns` – 既存の列名のリスト。
- `endDate` – ランダムな日付を取得する日付の範囲の終了。

Example例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_RANDOM_DATE_BETWEEN",
        "Parameters": {
            "startDate": "2020-12-12 12:12:12",
            "sourceColumns": ["column1", "column2"],
            "endDate": "2021-12-12 12:12:12"
        }
    }
}
```

SHUFFLE_ROWS

特定の列の値をシャッフルします。シャッフルは、セカンダリ列でグループ化された値で発生する可能性があります。

パラメータ

- `sourceColumns` – 既存の列の配列。
- `groupByColumns` – シャッフル中にソース列をグループ化する列の配列。

Example例

```
{
   "sourceColumns": ["age"],
   "*groupByColumns*": ["country"]
}
```

外れ値の検出とレシピステップの処理

これらのレシピステップを使用して、データ内の外れ値を操作し、高度な変換を実行します。

トピック

- FLAG_OUTLIERS
- REMOVE_OUTLIERS
- REPLACE_OUTLIERS
- RESCALE_OUTLIERS_WITH_Z_SCORE
- RESCALE_OUTLIERS_WITH_SKEW

FLAG_OUTLIERS

ソース列の値が外れ値であるかどうかを示すカスタマイズ可能な値を各行に含む新しい列を返します。

パラメータ

- `sourceColumn` – 外れ値を含む可能性のある既存の数値列の名前を指定します。
- `targetColumn` – 外れ値評価戦略の結果を挿入する新しい列の名前を指定します。
- `outlierStrategy` – 外れ値の検出に使用するアプローチを指定します。有効な値は次のとおりです。
  - `Z_SCORE` – 平均値から標準偏差のしきい値を超えて逸脱した場合、値を外れ値として識別します。
  - `MODIFIED_Z_SCORE` – 絶対偏差しきい値の中央値よりも大きく中央値から逸脱した場合、値を外れ値として識別します。
  - `IQR` – 列データの最初の四分位数と最後の四分位数を超えた値を外れ値として識別します。四分位範囲 (IQR) は、データポイントの中間 50% の場所を測定します。

- threshold – 外れ値を検出するときに使用するしきい値を指定します。で計算されたスコアがこの数outlierStrategyを超える場合、sourceColumn値は外れ値として識別されます。デフォルトは 3 です。
- trueString – 外れ値が検出された場合に使用する文字列値を指定します。デフォルトは「True」です。
- falseString – 外れ値が検出されない場合に使用する文字列値を指定します。デフォルトは「False」です。

次の例では、1 回のRecipeActionオペレーションの構文を表示します。レシピには少なくとも 1 つのRecipeStepオペレーションが含まれ、レシピステップには少なくとも 1 つのレシピアクションが含まれます。レシピアクションは、指定したデータ変換を実行します。レシピアクションのグループは、最終的なデータセットを作成するために順番に実行されます。

JSON

以下は、JSON 構文を使用して RecipeStep DataBrew レシピの例のメンバーRecipeActionとして使用する例を示しています。レシピアクションのリストを示す構文の例については、「」を参照してくださいレシピ構造の定義。

Example JSON の例

```
{
    "Action": {
        "Operation": "FLAG_OUTLIERS",
        "Parameters": {
            "sourceColumn": "name-of-existing-column",
            "targetColumn": "name-of-new-column",
            "outlierStrategy": "IQR",
            "threshold": "1.5",
            "trueString": "Yes",
            "falseString": "No"
        }
    }
}
```

API オペレーションでこのレシピアクションを使用する方法の詳細については、CreateRecipe「」または「」を参照してくださいUpdateRecipe。これらの API オペレーションやその他の API オペレーションは、独自のコードで使用できます。

YAML

以下は、YAML 構文を使用して RecipeStep DataBrew レシピの例のメンバーRecipeActionとして使用する例を示しています。レシピアクションのリストを示す構文の例については、「」を参照してくださいレシピ構造の定義。

Example YAML の例

```
- Action:
  Operation: FLAG_OUTLIERS
  Parameters:
    sourceColumn: name-of-existing-column
    targetColumn: name-of-new-column
    outlierStrategy: IQR
    trueString: Outlier
    falseString: No
    threshold: '1.5'
```

API オペレーションでこのレシピアクションを使用する方法の詳細については、CreateRecipe「」または「」を参照してくださいUpdateRecipe。これらの API オペレーションやその他の API オペレーションは、独自のコードで使用できます。

REMOVE_OUTLIERS

パラメータの設定に基づいて、外れ値として分類されるデータポイントを削除します。

パラメータ

- sourceColumn – 外れ値を含む可能性のある既存の数値列の名前を指定します。
- outlierStrategy – 外れ値の検出に使用するアプローチを指定します。有効な値は次のとおりです。
  - Z_SCORE – 値が平均値から標準偏差のしきい値を超えると、外れ値として識別されます。
  - MODIFIED_Z_SCORE – 絶対偏差しきい値の中央値よりも大きく中央値から逸脱した場合、値を外れ値として識別します。
  - IQR – 列データの最初の四分位数と最後の四分位数を超えた値を外れ値として識別します。四分位範囲 (IQR) は、データポイントの中間 50% の場所を測定します。
- threshold – 外れ値を検出するときに使用するしきい値を指定します。で計算されたスコアがこの数outlierStrategyを超える場合、sourceColumn値は外れ値として識別されます。デフォルトは 3 です。

- removeType – データを削除する方法を指定します。有効な値は、DELETE_ROWS および CLEAR です。
- trimValue – 外れ値の全部または一部を削除するかどうかを指定します。このブール値はデフォルトで になりますFALSE。
  - FALSE – すべての外れ値を削除します
  - TRUE – minValueおよび で指定されたパーセンタイルしきい値の範囲外にある外れ値を削除しますmaxValue。
- minValue – 外れ値範囲の最小パーセンタイル値を示します。有効な範囲は 0 ~ 100 です。
- maxValue – 外れ値範囲の最大パーセンタイル値を示します。有効な範囲は 0 ~ 100 です。

次の例では、1 回のRecipeActionオペレーションの構文を表示します。レシピには少なくとも 1 つのRecipeStepオペレーションが含まれ、レシピステップには少なくとも 1 つのレシピアクションが含まれます。レシピアクションは、指定したデータ変換を実行します。レシピアクションのグループは、最終的なデータセットを作成するために順番に実行されます。

JSON

以下は、JSON 構文を使用して RecipeStep DataBrew レシピの例のメンバーRecipeActionとして使用する例を示しています。レシピアクションのリストを示す構文の例については、「」を参照してくださいレシピ構造の定義。

Example JSON の例

```
{
    "Action": {
        "Operation": "REMOVE_OUTLIERS",
        "Parameters": {
            "sourceColumn": "name-of-existing-column",
            "outlierStrategy": "Z_SCORE",
            "threshold": "3",
            "removeType": "DELETE_ROWS",
            "trimValue": "TRUE",
            "minValue": "5",
            "maxValue": "95"
        }
    }
}
```

API オペレーションでこのレシピアクションを使用する方法の詳細については、CreateRecipe「」または「」を参照してくださいUpdateRecipe。これらの API オペレーションやその他の API オペレーションは、独自のコードで使用できます。

YAML

YAML 構文を使用して DataBrew レシピの例のメンバーRecipeActionとして使用する例RecipeStepを次に示します。レシピアクションのリストを示す構文の例については、「」を参照してくださいレシピ構造の定義。

Example YAML の例

```
- Action:
  Operation: REMOVE_OUTLIERS
  Parameters:
    sourceColumn: name-of-existing-column
    outlierStrategy: Z_SCORE
    threshold: '3'
    removeType: DELETE_ROWS
    trimValue: 'TRUE'
    minValue: '5'
    maxValue: '95'
```

API オペレーションでこのレシピアクションを使用する方法の詳細については、CreateRecipe「」または「」を参照してくださいUpdateRecipe。これらの API オペレーションやその他の API オペレーションは、独自のコードで使用できます。

REPLACE_OUTLIERS

パラメータの設定に基づいて、外れ値として分類されるデータポイント値を更新します。

パラメータ

- sourceColumn – 外れ値を含む可能性のある既存の数値列の名前を指定します。
- outlierStrategy – 外れ値の検出に使用するアプローチを指定します。有効な値は次のとおりです。
  - Z_SCORE – 平均値から標準偏差のしきい値を超えて逸脱した場合、値を外れ値として識別します。
  - MODIFIED_Z_SCORE – 絶対偏差しきい値の中央値よりも大きく中央値から逸脱した場合、値を外れ値として識別します。

  - IQR – 列データの最初の四分位数と最後の四分位数を超えた値を外れ値として識別します。四分位範囲 (IQR) は、データポイントの中間 50% の場所を測定します。
- threshold – 外れ値を検出するときに使用するしきい値を指定します。で計算されたスコアがこの数outlierStrategyを超える場合、sourceColumn値は外れ値として識別されます。デフォルトは 3 です。
- replaceType – 外れ値を置き換えるときに使用する方法を指定します。有効な値は次のとおりです。
  - WINSORIZE_VALUES – 最小パーセンタイルと最大パーセンタイルを使用して値を制限します。
  - REPLACE_WITH_CUSTOM
  - REPLACE_WITH_EMPTY
  - REPLACE_WITH_NULL
  - REPLACE_WITH_MODE
  - REPLACE_WITH_AVERAGE
  - REPLACE_WITH_MEDIAN
  - REPLACE_WITH_SUM
  - REPLACE_WITH_MAX
- modeType – が の場合に使用するモーダル関数のタイプを示しますreplaceTypeREPLACE_WITH_MODE。有効な値には、MIN、MAX、および が含まれますAVERAGE。
- minValue – trimValueの使用時に適用される外れ値範囲の最小パーセンタイル値を示します。有効な範囲は 0 ~ 100 です。
- maxValue – trimValueの使用時に適用される外れ値範囲の最大パーセンタイル値を示します。有効な範囲は 0 ~ 100 です。
- value – の使用時に挿入する値を指定しますREPLACE_WITH_CUSTOM。
- trimValue – 外れ値の全部または一部を削除するかどうかを指定します。このブール値は、replaceTypeが REPLACE_WITH_NULL、、または TRUEの場合REPLACE_WITH_MODEに に設定されますWINSORIZE_VALUES。他のすべての FALSE では、デフォルトで に設定されます。
  - FALSE – すべての外れ値を削除します
  - TRUE – minValueおよび で指定されたパーセンタイル上限しきい値の範囲外にある外れ値を削除しますmaxValue。

次の例では、1 回のRecipeActionオペレーションの構文を表示します。レシピには少なくとも 1 つのRecipeStepオペレーションが含まれ、レシピステップには少なくとも 1 つのレシピアクションが含まれます。レシピアクションは、指定したデータ変換を実行します。レシピアクションのグループは、最終的なデータセットを作成するために順番に実行されます。

JSON

以下は、JSON 構文を使用して RecipeStep DataBrew レシピの例のメンバーRecipeActionとして使用する例を示しています。レシピアクションのリストを示す構文の例については、「」を参照してくださいレシピ構造の定義。

Example JSON の例

```
{
    "Action": {
        "Operation": "REPLACE_OUTLIERS",
        "Parameters": {
            "maxValue": "95",
            "minValue": "5",
            "modeType": "AVERAGE",
            "outlierStrategy": "Z_SCORE",
            "replaceType": "REPLACE_WITH_MODE",
            "sourceColumn": "name-of-existing-column",
            "threshold": "3",
            "trimValue": "TRUE"
        }
    }
}
```

API オペレーションでこのレシピアクションを使用する方法の詳細については、CreateRecipe「」または「」を参照してくださいUpdateRecipe。これらの API オペレーションやその他の API オペレーションは、独自のコードで使用できます。

YAML

以下は、YAML 構文を使用して RecipeStep DataBrew レシピの例のメンバーRecipeActionとして使用する例を示しています。レシピアクションのリストを示す構文の例については、「」を参照してくださいレシピ構造の定義。

Example YAML の例

```
- Action:
```

```
Operation: REMOVE_OUTLIERS
Parameters:
  sourceColumn: name-of-existing-column
  outlierStrategy: Z_SCORE
  threshold: '3'
  replaceType: REPLACE_WITH_MODE
  modeType: AVERAGE
  minValue: '5'
  maxValue: '95'
  trimValue: 'TRUE'
```

API オペレーションでこのレシピアクションを使用する方法の詳細については、CreateRecipe「」または「」を参照してくださいUpdateRecipe。これらの API オペレーションやその他の API オペレーションは、独自のコードで使用できます。

RESCALE_OUTLIERS_WITH_Z_SCORE

パラメータの設定に基づいて、各行に再スケーリングされた外れ値を持つ新しい列を返します。このアクションでは、Z スコア正規化を適用して、平均 (μ) が 0、標準偏差 (σ) が 1 になるようにデータ値を直線的にスケールします。外れ値を処理するには、このアクションをお勧めします。

パラメータ

- sourceColumn – 外れ値を含む可能性のある既存の数値列の名前を指定します。
- targetColumn – 外れ値を含む可能性のある既存の数値列の名前を指定します。
- outlierStrategy – 外れ値の検出に使用するアプローチを指定します。有効な値は次のとおりです。
  - Z_SCORE – 平均値から標準偏差のしきい値を超えて逸脱した場合、値を外れ値として識別します。
  - MODIFIED_Z_SCORE – 絶対偏差しきい値の中央値よりも大きく中央値から逸脱した場合、値を外れ値として識別します。
  - IQR – 列データの最初の四分位数と最後の四分位数を超えた値を外れ値として識別します。四分位範囲 (IQR) は、データポイントの中間 50% の場所を測定します。
- threshold – 外れ値を検出するときに使用するしきい値。で計算されたスコアがこの数outlierStrategyを超える場合、sourceColumn値は外れ値として識別されます。デフォルトは 3 です。

次の例では、1 回のRecipeActionオペレーションの構文を表示します。レシピには少なくとも 1 つのRecipeStepオペレーションが含まれ、レシピステップには少なくとも 1 つのレシピアクションが含まれます。レシピアクションは、指定したデータ変換を実行します。レシピアクションのグループは、最終的なデータセットを作成するために順番に実行されます。

JSON

以下は、JSON 構文を使用して RecipeStep DataBrew レシピオペレーションの例のメンバーRecipeActionとして使用する例を示しています。レシピアクションのリストを示す構文の例については、「」を参照してくださいレシピ構造の定義。

Example JSON の例

```
{
    "Action": {
        "Operation": "RESCALE_OUTLIERS_WITH_Z_SCORE",
        "Parameters": {
            "sourceColumn": "name-of-existing-column",
            "targetColumn": "name-of-new-column",
            "outlierStrategy": "Z_SCORE",
            "threshold": "3"
        }
    }
}
```

API オペレーションでこのレシピアクションを使用する方法の詳細については、CreateRecipe「」または「」を参照してくださいUpdateRecipe。これらの API オペレーションやその他の API オペレーションは、独自のコードで使用できます。

YAML

以下は、YAML 構文を使用して RecipeStep DataBrew レシピオペレーションの例のメンバーRecipeActionとして使用する例を示しています。レシピアクションのリストを示す構文の例については、「」を参照してくださいレシピ構造の定義。

Example YAML の例

```
- Action:
  Operation: REMOVE_OUTLIERS
  Parameters:
    sourceColumn: name-of-existing-column
    targetColumn: name-of-new-column
```

```
    outlierStrategy: Z_SCORE
    threshold: '3'
```

API オペレーションでこのレシピアクションを使用する方法の詳細については、CreateRecipe「」または「」を参照してくださいUpdateRecipe。これらの API オペレーションやその他の API オペレーションは、独自のコードで使用できます。

RESCALE_OUTLIERS_WITH_SKEW

パラメータの設定に基づいて、各行に再スケーリングされた外れ値を持つ新しい列を返します。このアクションは、指定されたログまたはルート変換を適用することで、分散の歪みを軽減します。歪んだデータを処理するには、このアクションをお勧めします。

パラメータ

- sourceColumn – 外れ値を含む可能性のある既存の数値列の名前を指定します。
- targetColumn – 外れ値を含む可能性のある既存の数値列の名前を指定します。
- outlierStrategy – 外れ値の検出に使用するアプローチを指定します。有効な値は次のとおりです。
  - Z_SCORE – 値が平均値から標準偏差のしきい値を超えると、外れ値として識別されます。
  - MODIFIED_Z_SCORE – 絶対偏差しきい値の中央値よりも大きく中央値から逸脱した場合、値を外れ値として識別します。
  - IQR – 列データの最初の四分位数と最後の四分位数を超えた値を外れ値として識別します。四分位範囲 (IQR) は、データポイントの中間 50% の場所を測定します。
- threshold – 外れ値を検出するときに使用するしきい値を指定します。で計算されたスコアがこの数outlierStrategyを超える場合、sourceColumn値は外れ値として識別されます。デフォルトは 3 です。
- skewFunction – 外れ値を置き換えるときに使用する方法を指定します。有効な値は次のとおりです。
  - LOG – 強力な変換を適用して、正と負のスキューを減らします。これは自然対数 (2.718281828) です。
  - ROOT (value = 3 を使用) – 正と負のスキューを減らすために、かなり強力な変換を適用します。(キューブルート)
  - ROOT (value = 2 を使用) – 肯定的なスキューのみを減らすために、中程度の変換を適用します。(平方根)

  - SQUARE – 負のスキューを減らすために中程度の変換を適用します。(正方形)
  - カスタム変換 – valueパラメータで指定されたカスタム番号を使用して、指定された LOGまたは ROOT変換を適用します。
- value – カスタム変換に使用する値を指定します。skewFunction が LOG の場合、この値は口グのベースを表します。skewFunction が ROOT の場合、この値はルートのパワーを表します。

次の例では、1 回のRecipeActionオペレーションの構文を表示します。レシピには少なくとも 1 つのRecipeStepオペレーションが含まれ、レシピステップには少なくとも 1 つのレシピアクションが含まれます。レシピアクションは、指定したデータ変換を実行します。レシピアクションのグループは、最終的なデータセットを作成するために順番に実行されます。

JSON

以下は、JSON 構文を使用して RecipeStep DataBrew レシピの例のメンバーRecipeActionとして使用する例を示しています。レシピアクションのリストを示す構文の例については、「」を参照してくださいレシピ構造の定義。

Example JSON の例

```
{
    "Action": {
        "Operation": "RESCALE_OUTLIERS_WITH_SKEW",
        "Parameters": {
            "outlierStrategy": "Z_SCORE",
            "threshold": "3",
            "skewFunction": "ROOT",
            "sourceColumn": "name-of-existing-column",
            "targetColumn": "name-of-new-column",
            "value": "4"
        }
    }
}
```

API オペレーションでこのレシピアクションを使用する方法の詳細については、CreateRecipe「」または「」を参照してくださいUpdateRecipe。これらの API オペレーションやその他の API オペレーションは、独自のコードで使用できます。

YAML

YAML 構文を使用して DataBrew レシピの例のメンバーRecipeActionとして使用する例RecipeStepを次に示します。レシピアクションのリストを示す構文の例については、「」を参照してくださいレシピ構造の定義。

Example YAML の例

```
- Action:
  Operation: RESCALE_OUTLIERS_WITH_SKEW
  Parameters:
    outlierStrategy: Z_SCORE
    threshold: '3'
    skewFunction: ROOT
    sourceColumn: name-of-existing-column
    targetColumn: name-of-new-column
    value: '4'
```

API オペレーションでこのレシピアクションを使用する方法の詳細については、CreateRecipe「」または「」を参照してくださいUpdateRecipe。これらの API オペレーションやその他の API オペレーションは、独自のコードで使用できます。

列構造のレシピステップ

これらの列構造のレシピステップを使用して、データの列構造を変更します。

トピック

- BOOLEAN_OPERATION
- CASE_OPERATION
- FLAG_COLUMN_FROM_NULL
- FLAG_COLUMN_FROM_PATTERN
- MERGE
- SPLIT_COLUMN_BETWEEN_DELIMITER
- SPLIT_COLUMN_BETWEEN_POSITIONS
- SPLIT_COLUMN_FROM_END
- SPLIT_COLUMN_FROM_START
- SPLIT_COLUMN_MULTIPLE_DELIMITER

- SPLIT_COLUMN_SINGLE_DELIMITER
- SPLIT_COLUMN_WITH_INTERVALS

BOOLEAN_OPERATION

論理条件 IF の結果に基づいて、新しい列を作成します。ブール式が true の場合は true 値を返し、ブール式が false の場合は false 値を返し、カスタム値を返す。

パラメータ

- `trueValueExpression` – 条件が満たされた場合の結果。
- `falseValueExpression` – 条件が満たされない場合の結果。
- `valueExpression` – ブール条件。
- `withExpressions` – 集計結果の設定。
- `targetColumn` – 新しく作成された列の名前。

trueValueExpression、falseValueExpression、valueExpression では、定数値、列参照、集計結果を使用できます。

Example例: 定数値

数値や文など、変更されない値。

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
      "Parameters": {
        "trueValueExpression": "It is true.",
        "falseValueExpression": "It is false.",
        "valueExpression": "`column.1` < 2000",
        "targetColumn": "result.column"
      }
    }
  }
}
```

Example例: 列参照

データセット内の列である値。

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
      "Parameters": {
        "trueValueExpression": "`column.2`",
        "falseValueExpression": "`column.3`",
        "valueExpression": "`column.1` < `column.4`",
        "targetColumn": "result.column"
      }
    }
  }
}
```

Example例: 結果を集計する

集計関数によって計算される値。集計関数は列に対して計算を実行し、単一の値を返します。

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
      "Parameters": {
        "trueValueExpression": "`:mincolumn.2`",
        "falseValueExpression": "`:maxcolumn.3`",
        "valueExpression": "`column.1` < `:avgcolumn.4`",
        "withExpressions": "[{\"name\":\"mincolumn.2\",\"value\":\"min(`column.2`)\",
\"type\":\"aggregate\"},{\"name\":\"maxcolumn.3\",\"value\":\"max(`column.3`)\",\"type
\":\"aggregate\"},{\"name\":\"avgcolumn.4\",\"value\":\"avg(`column.4`)\",\"type\":
\"aggregate\"}]",
        "targetColumn": "result.column"
      }
    }
  }
}
```

ユーザーはエスケープして JSON を文字列に変換する必要があります。

trueValueExpression、falseValueExpression、および valueExpression のパラメータ名は、withExpressions の名前と一致する必要があることに注意してください。一部の列の集計結果を使用するには、それらの列のパラメータを作成し、集計関数を指定する必要があります。

Example例:

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
      "Parameters": {
        "trueValueExpression": "It is true.",
        "falseValueExpression": "It is false.",
        "valueExpression": "`column.1` < 2000",
        "targetColumn": "result.column"
      }
    }
  }
}
```

Example例: および/または

および または を使用して、複数の条件を組み合わせることができます。

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
      "Parameters": {
        "trueValueExpression": "It is true.",
        "falseValueExpression": "It is false.",
        "valueExpression": "`column.1` < 2000 and `column.2` >= `column.3",
        "targetColumn": "result.column"
      }
    }
  }
}
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
      "Parameters": {
```

```
        "trueValueExpression": "`column.4`",
        "falseValueExpression": "`column.5`",
        "valueExpression": "startsWith(`column1`, 'value1') or endsWith(`column2`,
 'value2')",
        "targetColumn": "result.column"
      }
    }
  }
}
```

有効な集計関数

以下の表は、ブール演算で使用できるすべての有効な集計関数を示しています。

[列のタイプ]	Condition	valueExpression	withExpressions	戻り値
数値	合計	`:sum.column.1`	`[ { "name": "sum.column.1", "value": "sum(`column.1`)", "type": "aggregate" } ]`	の合計を返します。 column.1
	Mean	`:mean.column.1`	`[ { "name": "mean.column.1", "value": "avg(`col`	の平均を返します。 column.1

[列のタイプ]	Condition	valueExpression	withExpressions	戻り値
			umn.1`)", "type": "aggregat e" }]	
	平均絶対偏差	`:meanabs olutedevi ation.column.1`	[{ "name": "meanabso lutedevia tion.colu mn.1", "value": "mean_abs olute_dev iation(`c olumn.1`) ", "type": "aggregat e" }]	の平均絶対偏差を返します。column.1

[列のタイプ]	Condition	valueExpression	withExpressions	戻り値
	中央値	`:median.column.1`	`[ { "name": "median.column.1", "value": "median(`column.1`)", "type": "aggregate" } ]`	の中央値を返します。 `column.1`
	製品	`:product.column.1`	`[ { "name": "product.column.1", "value": "product(`column.1`)", "type": "aggregate" } ]`	の積を返します。 `column.1`

[列のタイプ]	Condition	valueExpression	withExpressions	戻り値
	標準偏差	`:standarddeviation.column.1`	`[ { "name": "standarddeviation.column.1", "value": "stddev(`column.1`)", "type": "aggregate" } ]`	の標準偏差を返します。 `column.1`
	Variance	`:variance.column.1`	`[ { "name": "variance.column.1", "value": "variance(`column.1`)", "type": "aggregate" } ]`	の分散を返します。 `column.1`

[列のタイプ]	Condition	valueExpression	withExpressions	戻り値
	平均の標準誤差	`:standarderrorofmean.column.1`	`[ { "name": "standarderrorofmean.column.1", "value": "standard_error_of_mean(`column.1`)", "type": "aggregate" } ]`	の平均の標準誤差を返します。 column.1
	歪み	`:skewness.column.1`	`[ { "name": "skewness.column.1", "value": "skewness(`column.1`)", "type": "aggregate" } ]`	の歪度を返します。 column.1

[列のタイプ]	Condition	valueExpression	withExpressions	戻り値
	尖度	`:kurtosis.column.1`	`[ { "name": "kurtosis.column.1", "value": "kurtosis(`column.1`)", "type": "aggregate" } ]`	の尖度を返します。 `column.1`
Datetime/ Numeric/Text	カウント	`:count.column.1`	`[ { "name": "count.column.1", "value": "count(`column.1`)", "type": "aggregate" } ]`	の合計行数を返します。 `column.1`

[列のタイプ]	Condition	valueExpression	withExpressions	戻り値
	個別にカウント	`:countdistinct.column.1`	`[ { "name": "count.column.1", "value": "count(distinct `column.1`)", "type": "aggregate" } ]`	の個別の行の総数を返します。column.1
	最小	`:min.column.1`	`[ { "name": "min.column.1", "value": "min(`column.1`)", "type": "aggregate" } ]`	の最小値を返します。column.1

[列のタイプ]	Condition	valueExpression	withExpressions	戻り値
	最大	`:max.column.1`	`[ { "name": "max.column.1", "value": "max(`column.1`)", "type": "aggregate" } ]`	の最大値を返します。column.1

valueExpression の有効な条件

次の表は、サポートされている条件と使用できる値式を示しています。

[列のタイプ]	Condition	valueExpression	説明
文字列	Contains	contains(`column`, 'text')	列の値にテキストが含まれているかどうかをテストする条件
	は含まれません	!contains(`column`, 'text')	列の値にテキストが含まれていないかどうかをテストする条件
	マッチ	matches(`column`, 'pattern')	列の値がパターンと一致するかどうかをテストする条件

[列のタイプ]	Condition	valueExpression	説明
	一致しない	!matches(`column`, 'pattern')	列の値がパターンと一致しないかどうかをテストする条件
	Starts with (で始まる)	startsWith(`column`, 'text')	列の値がテキストで始まるかどうかをテストする条件
	で始まらない	!startsWith(`column`, 'text')	列の値がテキストで始まらないかどうかをテストする条件
	Ends with	endsWith(`column`, 'text')	列の値がテキストで終わるかどうかをテストする条件
	で終わることはありません	!endsWith(`column`, 'text')	列の値がテキストで終わらないかどうかをテストする条件
数値	Less than	「column」 < 数値	列の値が数値より小さいかどうかをテストする条件
	以下	`column` <= number	列の値が数値以下かどうかをテストする条件
	Greater than	`column` > number	列の値が数値より大きいかどうかをテストする条件
	以上	`column` >= number	列の値が数値以上かどうかをテストする条件

[列のタイプ]	Condition	valueExpression	説明
	の間	isBetween(`column`, minNumber, maxNumber)	列の値が minNumber と maxNumber の間にあるかどうかをテストする条件
	間ではない	!isBetween(`column`, minNumber, maxNumber)	列の値が minNumber と maxNumber の間にないかどうかをテストする条件
ブール値	正しい	`column` = TRUE	列の値がブール TRUE であるかどうかをテストする条件
	false です	`column` = FALSE	列の値がブール型 FALSE であるかどうかをテストする条件
日付/タイムスタンプ	より前	「列」 <「日付」	列の値が日付より前であるかどうかをテストする条件
	以前	`column` <= 'date'	列の値が日付以前であるかどうかをテストする条件
	より後	「列」 >「日付」	列の値が日付より後であるかどうかをテストする条件
	以降	`column` >= 'date'	列の値が日付以降であるかどうかをテストする条件
String/Numeric/Date/タイムスタンプ	は正確に	`column` = 'value'	列の値が正確に値であるかどうかをテストする条件

[列のタイプ]	Condition	valueExpression	説明
	Is not	「列」!= 「値」	列の値が値ではないかどうかをテストする条件
	が見つかりません	isMissing(`column`)	列の値が欠落しているかどうかをテストする条件
	欠落していない	!isMissing(`column`)	列の値が欠落していないかどうかをテストする条件
	有効	isValid(`column`, datatype)	列の値が有効かどうかをテストする条件 (値はデータ型であるか、データ型に変換可能)
	有効ではありません	!isValid(`column`, datatype)	列の値が有効でないかどうかをテストする条件 (値はデータ型であるか、データ型に変換可能)
ネスト済み	が見つかりません	isMissing(`column`)	列の値が欠落しているかどうかをテストする条件
	欠落していない	!isMissing(`column`)	列の値が欠落していないかどうかをテストする条件

[列のタイプ]	Condition	valueExpression	説明
	有効	isValid(`column`, datatype)	列の値が有効かどうかをテストする条件 (値はデータ型であるか、データ型に変換可能)
	有効ではありません	!isValid(`column`, datatype)	列の値が有効でないかどうかをテストする条件 (値はデータ型であるか、データ型に変換可能)

CASE_OPERATION

論理条件 CASE の結果に基づいて、新しい列を作成します。ケースオペレーションはケース条件を通過し、最初の条件が満たされると値を返します。条件が true になると、オペレーションは読み取りを停止し、結果を返します。true の条件がない場合、デフォルト値を返します。

パラメータ

- `valueExpression` – 条件。
- `withExpressions` – 集計結果の設定。
- `targetColumn` – 新しく作成された列の名前。

Example例

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "CASE_OPERATION",
      "Parameters": {
        "valueExpression": "case when `column11` < `column.2` then 'result1' when
 `column2` < 'value2' then 'result2' else 'high' end",
        "targetColumn": "result.column"
      }
```

```
    }
  }
}
```

有効な集計関数

次の表は、ケースオペレーションで使用できるすべての有効な集計関数を示しています。

[列のタイプ]	Condition	valueExpression	withExpressions	戻り値
数値	合計	`:sum.column.1`	`[ { "name": "sum.column.1", "value": "sum(`column.1`)", "type": "aggregate" } ]`	の合計を返します。 column.1
	Mean	`:mean.column.1`	`[ { "name": "mean.column.1", "value": "avg(`column.1`)", "type": "aggregate"`	の平均を返します。 column.1

[列のタイプ]	Condition	valueExpression	withExpressions	戻り値
			}]	

	平均絶対偏差	`:meanabsolutedeviation.column.1`	[
			{
			"name":
			"meanabsolutedeviation.column.1",
			"value":
			"mean_absolute_deviation(`column.1`)",
			"type":
			"aggregate"
			}
			]
				の平均絶対偏差を返します。column.1

[列のタイプ]	Condition	valueExpression	withExpressions	戻り値
	中央値	`:median.column.1`	`[ { "name": "median.column.1", "value": "median(`column.1`)", "type": "aggregate" } ]`	の中央値を返します。 `column.1`
	製品	`:product.column.1`	`[ { "name": "product.column.1", "value": "product(`column.1`)", "type": "aggregate" } ]`	の積を返します。 `column.1`

[列のタイプ]	Condition	valueExpression	withExpressions	戻り値
	標準偏差	`:standarddeviation.column.1`	`[ { "name": "standarddeviation.column.1", "value": "stddev(`column.1`)", "type": "aggregate" } ]`	の標準偏差を返します。 column.1
	Variance	`:variance.column.1`	`[ { "name": "variance.column.1", "value": "variance(`column.1`)", "type": "aggregate" } ]`	の分散を返します。 column.1

[列のタイプ]	Condition	valueExpression	withExpressions	戻り値
	平均の標準誤差	`:standarderrorofmean.column.1`	`[ { "name": "standarderrorofmean.column.1", "value": "standard_error_of_mean(`column.1`)", "type": "aggregate" } ]`	の平均の標準誤差を返します。 column.1
	歪度	`:skewness.column.1`	`[ { "name": "skewness.column.1", "value": "skewness(`column.1`)", "type": "aggregate" } ]`	の歪度を返します。 column.1

[列のタイプ]	Condition	valueExpression	withExpressions	戻り値
	尖度	`:kurtosis.column.1`	`[ { "name": "kurtosis.column.1", "value": "kurtosis(`column.1`)", "type": "aggregate" } ]`	の尖度を返します。 `column.1`
Datetime/ Numeric/Text	カウント	`:count.column.1`	`[ { "name": "count.column.1", "value": "count(`column.1`)", "type": "aggregate" } ]`	の合計行数を返します。 `column.1`

[列のタイプ]	Condition	valueExpression	withExpressions	戻り値
	個別にカウント	`:countdistinct.column.1`	`[ { "name": "count.column.1", "value": "count(distinct `column.1`)", "type": "aggregate" } ]`	の個別の行の総数を返します。column.1
	最小	`:min.column.1`	`[ { "name": "min.column.1", "value": "min(`column.1`)", "type": "aggregate" } ]`	の最小値を返します。column.1

[列のタイプ]	Condition	valueExpression	withExpressions	戻り値
	最大	`:max.column.1`	`[ { "name": "max.column.1", "value": "max(`column.1`)", "type": "aggregate" } ]`	の最大値を返します。column.1

valueExpression の有効な条件

次の表は、サポートされている条件と使用できる値式を示しています。

[列のタイプ]	Condition	valueExpression	説明
文字列	Contains	contains(`column`, 'text')	列の値にテキストが含まれているかどうかをテストする条件
	は含まれません	!contains(`column`, 'text')	列の値にテキストが含まれていないかどうかをテストする条件
	マッチ	matches(`column`, 'pattern')	列の値がパターンと一致するかどうかをテストする条件

[列のタイプ]	Condition	valueExpression	説明
	一致しない	!matches(`column`, 'pattern')	列の値がパターンと一致しないかどうかをテストする条件
	Starts with (で始まる)	startsWith(`column`, 'text')	列の値がテキストで始まるかどうかをテストする条件
	で始まらない	!startsWith(`column`, 'text')	列の値がテキストで始まらないかどうかをテストする条件
	Ends with	endsWith(`column`, 'text')	列の値がテキストで終わるかどうかをテストする条件
	で終わることはありません	!endsWith(`column`, 'text')	列の値がテキストで終わらないかどうかをテストする条件
数値	Less than	「column」 < 数値	列の値が数値より小さいかどうかをテストする条件
	以下	`column` <= number	列の値が数値以下かどうかをテストする条件
	Greater than	`column` > number	列の値が数値より大きいかどうかをテストする条件
	以上	`column` >= number	列の値が数値以上かどうかをテストする条件

[列のタイプ]	Condition	valueExpression	説明
	の間	isBetween(`column`, minNumber, maxNumber)	列の値が minNumber と maxNumber の間にあるかどうかをテストする条件
	間にはありません	!isBetween(`column`, minNumber, maxNumber)	列の値が minNumber と maxNumber の間にないかどうかをテストする条件
ブール値	正しい	`column` = TRUE	列の値がブール TRUE であるかどうかをテストする条件
	false です	`column` = FALSE	列の値がブール型 FALSE であるかどうかをテストする条件
日付/タイムスタンプ	より前	「列」 <「日付」	列の値が日付より前であるかどうかをテストする条件
	以前	`column` <= 'date'	列の値が日付以前であるかどうかをテストする条件
	より後	「列」 >「日付」	列の値が日付より後であるかどうかをテストする条件
	以降	`column` >= 'date'	列の値が日付以降であるかどうかをテストする条件
String/Numeric/Date/タイムスタンプ	は正確に	`column` = 'value'	列の値が正確に値であるかどうかをテストする条件

[列のタイプ]	Condition	valueExpression	説明
	Is not	「列」!= 「値」	列の値が値ではないかどうかをテストする条件
	が見つかりません	isMissing(`column`)	列の値が欠落しているかどうかをテストする条件
	欠落していない	!isMissing(`column`)	列の値が欠落していないかどうかをテストする条件
	有効	isValid(`column`, datatype)	列の値が有効かどうかをテストする条件 (値はデータ型であるか、データ型に変換可能)
	有効ではありません	!isValid(`column`, datatype)	列の値が有効でないかどうかをテストする条件 (値はデータ型であるか、データ型に変換可能)
ネスト済み	が見つかりません	isMissing(`column`)	列の値が欠落しているかどうかをテストする条件
	欠落していない	!isMissing(`column`)	列の値が欠落していないかどうかをテストする条件

[列のタイプ]	Condition	valueExpression	説明
	有効	isValid(`column`, datatype)	列の値が有効かどうかをテストする条件 (値はデータ型であるか、データ型に変換可能)
	有効ではありません	!isValid(`column`, datatype)	列の値が有効でないかどうかをテストする条件 (値はデータ型であるか、データ型に変換可能)

FLAG_COLUMN_FROM_NULL

既存の列に null 値が存在することに基づいて、新しい列を作成します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `targetColumn` – 作成する新しい列の名前。
- `flagType` – に設定する必要がある値`Null values`。
- `trueString` – ソースで null 値が見つかった場合の新しい列の値。値が指定されていない場合、デフォルト値は `True` です。
- `falseString` – NULL 以外の値がソースで見つかった場合の新しい列の値。値が指定されていない場合、デフォルト値は `False` です。

Example例

```
{
    "RecipeAction": {
        "Operation": "FLAG_COLUMN_FROM_NULL",
        "Parameters": {
            "flagType": "Null values",
            "sourceColumn": "weight_kg",
```

```
            "targetColumn": "is_weight_kg_missing"
        }
    }
}
```

FLAG_COLUMN_FROM_PATTERN

既存の列にユーザーが指定したパターンが存在することに基づいて、新しい列を作成します。

パラメータ

- sourceColumn - 既存の列の名前。
- targetColumn – 作成する新しい列の名前。
- flagType – に設定する必要がある値Pattern。
- pattern – 評価するパターンを示す正規表現。
- trueString – ソースで null 値が見つかった場合の新しい列の値。値が指定されていない場合、デフォルト値は True です。
- falseString – NULL 以外の値がソースで見つかった場合の新しい列の値。値が指定されていない場合、デフォルト値は False です。

Example例

```
{
    "RecipeAction": {
        "Operation": "FLAG_COLUMN_FROM_PATTERN",
        "Parameters": {
            "falseString": "No",
            "flagType": "Pattern",
            "pattern": "N.*",
            "sourceColumn": "wind_direction",
            "targetColumn": "northerly",
            "trueString": "yes"
        }
    }
}
```

MERGE

2 つ以上の列を新しい列にマージします。

パラメータ

- sourceColumns – マージする 1 つ以上の列のリストを表す JSON エンコードされた文字列。
- delimiter – ターゲット列に表示する値のオプションの区切り文字。
- targetColumn – 作成するマージされた列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "MERGE",
        "Parameters": {
            "delimiter": " ",
            "sourceColumns": "[\"first_name\",\"last_name\"]",
            "targetColumn": "Merged Column 1"
        }
    }
}
```

SPLIT_COLUMN_BETWEEN_DELIMITER

開始と終了の区切り文字に従って、列を 3 つの新しい列に分割します。

パラメータ

- sourceColumn - 既存の列の名前。
- patternOption1 – 最初の区切り文字を示す 1 つ以上の文字を表す JSON エンコードされた文字列。
- patternOption2 – 2 番目の区切り文字を示す 1 つ以上の文字を表す JSON エンコードされた文字列。
- pattern – データを分割するときに区切り文字として使用する 1 つ以上の文字。
- includeInSplit – true の場合、新しい列にパターンを含めます。それ以外の場合、パターンは破棄されます。

Example例

```
{
```

```
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_BETWEEN_DELIMITER",
        "Parameters": {
            "patternOption1": "{\"pattern\":\"H\",\"includeInSplit\":true}",
            "patternOption2": "{\"pattern\":\"M\",\"includeInSplit\":true}",
            "sourceColumn": "last_name"
        }
    }
}
```

SPLIT_COLUMN_BETWEEN_POSITIONS

指定したオフセットに従って、列を 3 つの新しい列に分割します。

パラメータ

- sourceColumn - 既存の列の名前。
- startPosition – 分割を開始する文字の位置。
- endPosition – 分割が終了する文字の位置。

Example例

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_BETWEEN_POSITIONS",
        "Parameters": {
            "endPosition": "12",
            "sourceColumn": "last_name",
            "startPosition": "2"
        }
    }
}
```

SPLIT_COLUMN_FROM_END

文字列の末尾からのオフセットで、列を 2 つの新しい列に分割します。

パラメータ

- sourceColumn - 既存の列の名前。

- position – 分割が行われる文字列の右端からの文字位置。

Example例

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_FROM_END",
        "Parameters": {
            "position": "1",
            "sourceColumn": "nationality"
        }
    }
}
```

SPLIT_COLUMN_FROM_START

文字列の先頭からのオフセットで、列を 2 つの新しい列に分割します。

パラメータ

- sourceColumn - 既存の列の名前。
- position – 分割が行われる文字列の左端からの文字位置。

Example例

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_FROM_START",
        "Parameters": {
            "position": "1",
            "sourceColumn": "first_name"
        }
    }
}
```

SPLIT_COLUMN_MULTIPLE_DELIMITER

複数の区切り文字に従って列を分割します。

パラメータ

- sourceColumn - 既存の列の名前。
- patternOptions – 分割基準を決定する 1 つ以上のパターンを表す JSON エンコードされた文字列。
- pattern – データを分割するときに区切り文字として使用する 1 つ以上の文字。
- limit – 実行する分割の数。最小は 1、最大は 20 です。
- includeInSplit – true の場合、新しい列にパターンを含めます。それ以外の場合、パターンは破棄されます。

Example例

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_MULTIPLE_DELIMITER",
        "Parameters": {
            "limit": "1",
            "patternOptions": "[{\"pattern\":\",\",\"includeInSplit\":true},{\"pattern
\":\" \",\"includeInSplit\":true}]",
            "sourceColumn": "description"
        }
    }
}
```

SPLIT_COLUMN_SINGLE_DELIMITER

特定の区切り文字に従って、列を 1 つ以上の新しい列に分割します。

パラメータ

- sourceColumn - 既存の列の名前。
- pattern – データを分割するときに区切り文字として使用する 1 つ以上の文字。
- limit – 実行する分割の数。最小は 1、最大は 20 です。
- includeInSplit – true の場合、新しい列にパターンを含めます。それ以外の場合、パターンは破棄されます。

Example例

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_SINGLE_DELIMITER",
        "Parameters": {
            "includeInSplit": "true",
            "limit": "1",
            "pattern": "/",
            "sourceColumn": "info_url"
        }
    }
}
```

SPLIT_COLUMN_WITH_INTERVALS

n 文字の間隔で列を分割し、n を指定します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `startPosition` – 分割を開始する文字の位置。
- `interval` – 次の分割前にスキップする文字数。

Example例

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_WITH_INTERVALS",
        "Parameters": {
            "interval": "4",
            "sourceColumn": "nationality",
            "startPosition": "1"
        }
    }
}
```

列フォーマットレシピのステップ

列フォーマットレシピステップを使用して、列内のデータの形式を変更します。

トピック

NUMBER_FORMAT

数値がフォーマットされた文字列に変換される列を返します。

パラメータ

- `sourceColumn` – 文字列。既存の列の名前。
- `decimalPlaces` – 整数。小数点以下の桁数の値。
- `numericDecimalSeparator` – 文字列。小数点区切り文字を示す次のいずれかの値。
  - "."
  - ","
- `numericThousandSeparator` – 文字列。千の区切り文字を示す次のいずれかの値。
  - null。千の区切り文字が有効になっていないことを示します。
  - ","
  - " "
  - "."
  - "\\"
- `numericAbbreviatedUnit` – 文字列。略語の単位を示す次のいずれかの値。
  - null。略語単位が有効になっていないことを示します。
  - 「千」
  - 「百万」
  - 「BILLION」
  - 「TRILLION」
- `numericUnitAbbreviation` – 文字列。次のいずれかの値、または単位の略語を示す任意のカスタム値。

- null。単位の略語が有効になっていないことを示します。
-

略語の単位	オプション
数千	K、k、M、千、カスタム
Million	M、m、MM、百万、カスタム
Billion	B、bn、10 億、カスタム
兆	T、tn、 trillion、カスタム

Example例

```
{
    "RecipeAction": {
        "Operation": "NUMBER_FORMAT",
        "Parameters": {
            "sourceColumn": "income",
            "decimalPlaces": "2",
            "numericDecimalSeparator": ".",
            "numericThousandSeparator": ",",
            "numericAbbreviatedUnit": "THOUSAND",
            "numericUnitAbbreviation": "K"
        }
    }
}
```

FORMAT_PHONE_NUMBER

電話番号文字列がフォーマットされた値に変換される列を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- phoneNumberFormat – 電話番号を変換するフォーマット。フォーマットが指定されていない場合、デフォルトは E.164です。これは国際的に認められている標準の電話番号フォーマットです。有効な値は次のとおりです。

  - E164 (より後の期間は省略E)
- defaultRegion – 番号自体に国コードが存在しない場合に電話番号のリージョンを指定する 2 つまたは 3 つの大文字で構成される有効なリージョンコード。最大で、defaultRegion または defaultRegionColumn のいずれかを指定できます。
- defaultRegionColumn – アドバンストデータ型 の列の名前Country。指定された列のリージョンコードは、電話番号自体に国コードがない場合に、電話番号の国コードを決定するために使用されます。最大で、defaultRegion または defaultRegionColumn のいずれかを指定できます。

注意事項

- 有効な電話番号にフォーマットできない入力は変更されません。
- デフォルトのリージョンが指定されておらず、電話番号がプラス記号 (+) と国番号で始まらない場合、電話番号はフォーマットされません。

Example

例: 固定デフォルトリージョン

```
{
    "Action": {
        "Operation": "FORMAT_PHONE_NUMBER",
        "Parameters": {
            "sourceColumn": "Phone Number",
            "defaultRegion": "US"
        }
    }
}
```

例: デフォルトのリージョン列オプション

```
{
    "Action": {
        "Operation": "FORMAT_PHONE_NUMBER",
        "Parameters": {
            "sourceColumn": "Phone Number",
            "defaultRegionColumn": "Country Code"
```

```
        }
    }
}
```

データ構造レシピのステップ

これらのレシピステップを使用して、さまざまな視点からデータを集計および要約したり、高度な関数を実行したりできます。

トピック

NEST_TO_ARRAY

ユーザーが選択した列を配列値に変換します。選択した列の順序は、結果の配列の作成中に維持されます。さまざまな列データ型は、すべての列のデータ型をサポートする共通の型に型キャストされます。

パラメータ

- `sourceColumns` — ソース列のリスト。

- targetColumn — ターゲット列の名前。
- removeSourceColumns — ユーザーが選択したソース列を削除するかどうかを示すfalse値 trueまたは が含まれます。

Example例

```
{
    "RecipeAction": {
        "Operation": "NEST_TO_ARRAY",
        "Parameters": {
            "sourceColumns": "[\"age\",\"weight_kg\",\"height_cm\"]",
            "targetColumn": "columnName",
            "removeSourceColumns": "true"
        }
    }
}
```

NEST_TO_MAP

ユーザーが選択した列をキーと値のペアに変換します。各列には列名を表すキーと行値を表す値があります。選択した列の順序は、結果マップの作成中に維持されません。さまざまな列データ型は、すべての列のデータ型をサポートする共通の型に型キャストされます。

パラメータ

- sourceColumns — ソース列のリスト。
- targetColumn — ターゲット列の名前。
- removeSourceColumns — ユーザーが選択したソース列を削除するかどうかを示すfalse値 trueまたは が含まれます。

Example例

```
{
    "RecipeAction": {
        "Operation": "NEST_TO_MAP",
        "Parameters": {
            "sourceColumns": "[\"age\",\"weight_kg\",\"height_cm\"]",
```

```
            "targetColumn": "columnName",
            "removeSourceColumns": "true"
        }
    }
}
```

NEST_TO_STRUCT

ユーザーが選択した列をキーと値のペアに変換します。各列には列名を表すキーと行値を表す値があります。選択した列の順序と各列のデータ型は、結果の構造体に保持されます。

パラメータ

- sourceColumns — ソース列のリスト。
- targetColumn — ターゲット列の名前。
- removeSourceColumns — ユーザーが選択したソース列を削除するかどうかを示すfalse値 trueまたは が含まれます。

Example例

```
{
    "RecipeAction": {
        "Operation": "NEST_TO_STRUCT",
        "Parameters": {
            "sourceColumns": "[\"age\",\"weight_kg\",\"height_cm\"]",
            "targetColumn": "columnName",
            "removeSourceColumns": "true"
        }
    }
}
```

UNNEST_ARRAY

タイプの列を新しい列arrayにネスト解除します。配列に複数の値が含まれている場合、各要素に対応する行が生成されます。この関数は、配列列の 1 つのレベルのみをネスト解除します。

パラメータ

- sourceColumn — 既存の列の名前。この列は struct型である必要があります。

- targetColumn — 生成されるターゲット列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "UNNEST_ARRAY",
        "Parameters": {
            "sourceColumn": "address",
            "targetColumn": "address"
        }
    }
}
```

UNNEST_MAP

タイプの列をネスト解除mapし、キーと値の列を生成します。キーと値のペアが複数ある場合、各キー値に対応する行が生成されます。この関数は、マップ列の 1 つのレベルのみをネスト解除します。

パラメータ

- sourceColumn — 既存の列の名前。この列は struct型である必要があります。
- removeSourceColumn — の場合true、ソース列は関数の完了後に削除されます。
- targetColumn — 指定した場合、生成された各列はプレフィックスとしてこの列で始まります。

Example例

```
{
    "RecipeAction": {
        "Operation": "UNNEST_MAP",
        "Parameters": {
            "sourceColumn": "address",
            "removeSourceColumn": "false",
            "targetColumn": "address"
        }
    }
}
```

UNNEST_STRUCT

タイプの列をネスト解除structし、構造体に存在する各キーの列を生成します。この関数は、構造体レベル 1 のみをネスト解除します。

パラメータ

- sourceColumn — 既存の列の名前。この列は struct 型である必要があります。
- removeSourceColumn — の場合true、ソース列は関数の完了後に削除されます。
- targetColumn — 指定した場合、生成された各列はプレフィックスとしてこの列で始まります。

Example例

```
{
    "RecipeAction": {
        "Operation": "UNNEST_STRUCT",
        "Parameters": {
            "sourceColumn": "address",
            "removeSourceColumn": "false"
            "targetColumn": "add"
        }
    }
}
```

UNNEST_STRUCT_N

タイプの選択した列のフィールドごとに新しい列を作成しますstruct。

たとえば、次の構造体があるとします。

```
           user {
              name: “Ammy”
              address: {
                 state: "CA",
                 zipcode: 12345
              }
           }
```

この関数は 3 つの列を作成します。

user.name	user.address.state	user.address.zipcode
アミー	CA	12345

パラメータ

- `sourceColumns` — ソース列のリスト。
- `regexColumnSelector` — ネスト解除する列を選択する正規表現。
- `removeSourceColumn` — ブール値。true の場合はソース列を削除し、それ以外の場合は保持します。
- `unnestLevel` — ネスト解除するレベルの数。
- `delimiter` — 区切り記号は、新しく作成された列名で使用され、構造体のさまざまなレベルを区切ります。例えば、区切り文字が「/」の場合、列名は「user/address/state」の形式になります。
- `conditionExpressions` — 条件式。

Example例

```
{
    "RecipeAction": {
        "Operation": "UNNEST_STRUCT_N",
        "Parameters": {
            "sourceColumns": "[\"address\"]",
            "removeSourceColumn": "true",
            "unnestLevel": "2",
            "delimiter": "/"
        }
    }
}
```

GROUP_BY

行を 1 つ以上の列でグループ化し、集計関数を各グループに適用することで、データを要約します。

パラメータ

- sourceColumns — 各グループの基礎を形成する列のリストを表す JSON エンコードされた文字列。
- groupByAggFunctions — 適用する集計関数のリストを表す JSON エンコードされた文字列。(集計しない場合は、 を指定します)UNAGGREGATED。
- useNewDataFrame — true の場合、GROUP_BY の結果がプロジェクトセッションで利用可能になり、現在の内容が置き換えられます。

Example例

```
[
  {
    "Action": {
      "Operation": "GROUP_BY",
      "Parameters": {
        "groupByAggFunctionOptions": "[{\"sourceColumnName\":\"all_votes\",
\"targetColumnName\":\"all_votes_count\",\"targetColumnDataType\":\"number\",
\"functionName\":\"COUNT\"}]",
        "sourceColumns": "[\"year\",\"state_name\"]",
        "useNewDataFrame": "true"
      }
    }
  }
]
```

JOIN

2 つのデータセットに対して結合オペレーションを実行します。

パラメータ

- joinKeys — 結合キーとして機能する各データセットの列のリストを表す JSON エンコードされた文字列。
- joinType — 実行する結合のタイプ。| INNER_JOIN | LEFT_JOIN || RIGHT_JOIN OUTER_JOIN | LEFT_EXCLUDING_JOIN | RIGHT_EXCLUDING_JOIN のいずれかである必要があります。 OUTER_EXCLUDING_JOIN

- `leftColumns` — 現在のアクティブなデータセットの列のリストを表す JSON エンコードされた文字列。
- `rightColumns` — 現在のデータセットに結合する別の (セカンダリ) データセットの列のリストを表す JSON エンコードされた文字列。
- `secondInputLocation` — セカンダリデータセットのデータファイルに解決される Amazon S3 URL。
- `secondaryDatasetName` — セカンダリデータセットの名前。

Example例

```
{
    "Action": {
        "Operation": "JOIN",
        "Parameters": {
            "joinKeys": "[{\"key\":\"assembly_session\",\"value\":\"assembly_session
\"},{\"key\":\"state_code\",\"value\":\"state_code\"}]",
            "joinType": "INNER_JOIN",
            "leftColumns": "[\"year\",\"assembly_session\",\"state_code\",\"state_name
\",\"all_votes\",\"yes_votes\",\"no_votes\",\"abstain\",\"idealpoint_estimate
\",\"affinityscore_usa\",\"affinityscore_russia\",\"affinityscore_china\",
\"affinityscore_india\",\"affinityscore_brazil\",\"affinityscore_israel\"]",
            "rightColumns": "[\"assembly_session\",\"vote_id\",\"resolution\",
\"state_code\",\"state_name\",\"member\",\"vote\"]",
            "secondInputLocation": "s3://databrew-public-datasets-us-east-1/votes.csv",
            "secondaryDatasetName": "votes"
        }
    }
}
```

PIVOT

選択した列のすべての行値を、値を持つ個々の列に変換します。

パラメータ

- sourceColumn — 既存の列の名前。列には最大 10 個の個別の値を含めることができます。
- valueColumn — 既存の列の名前。列には最大 10 個の個別の値を含めることができます。
- aggregateFunction — 集計関数の名前。集計しない場合は、キーワード を使用しますCOLLECT_LIST。

Example例

```
{
    "Action": {
        "Operation": "PIVOT",
        "Parameters": {
            "aggregateFunction": "SUM",
            "sourceColumn": "state_name",
            "valueColumn": "all_votes"
        }
    }
}
```

スケール

数値列のデータ範囲をスケールまたは正規化します。

パラメータ

- sourceColumn — 既存の列の名前。
- strategy — 列値に適用されるオペレーション:
  - MIN_MAX — 値を [0,1] の範囲に再スケーリングします。
  - SCALE_BETWEEN — 指定した 2 つの値の範囲に値を再スケーリングします。
  - MEAN_NORMALIZATION — 平均 (μ) が 0、標準偏差 (σ) が [-1, 1] の範囲内の 1 になるようにデータを再スケーリングします。
  - Z_SCORE — データ値を線形的にスケールして、平均 (μ) を 0、標準偏差 (σ) を 1 にします。外れ値の処理に最適です。
- targetColumn — 結果を含む列の名前。

Example例

```
{
    "Action": {
        "Operation": "NORMALIZATION",
        "Parameters": {
            "sourceColumn": "all_votes",
            "strategy": "MIN_MAX",
            "targetColumn": "all_votes_normalized"
        }
    }
}
```

変換

選択したすべての行を列に変換し、列を行に変換します。

Column 1	Column A	Column B	Column C
Row A	Value A	Value B	Value C
Row B	Value A1	Value B1	Value C1

New column	Row A	Row B
Column A	Value A	Value A1
Column B	Value B	Value B1
Column C	Value C	Value C1

パラメータ

- `pivotColumns` — 行が列名に変換される列のリストを表す JSON エンコードされた文字列。
- `valueColumns` — 行に変換する 1 つ以上の列のリストを表す JSON エンコードされた文字列。
- `aggregateFunction` — 集計関数の名前。集計しない場合は、キーワード を使用しますCOLLECT_LIST。
- `newColumn` — 変換された列を値として保持する列。

Example例

```
{
    "Action": {
        "Operation": "TRANSPOSE",
        "Parameters": {
            "pivotColumns": "[\"Teacher\"]",
            "valueColumns": "[\"Tom\",\"John\",\"Harry\"]",
            "aggregateFunction": "COLLECT_LIST",
            "newColumn": "Student"
        }
    }

}
```

UNION

2 つ以上のデータセットの行を 1 つの結果に結合します。

パラメータ

- datasetsColumns — データセット内のすべての列のリストを表す JSON エンコードされた文字列。
- secondaryDatasetNames — 1 つ以上のセカンダリデータセットのリストを表す JSON エンコードされた文字列。
- secondaryInputs — セカンダリデータセット (複数可) の場所を DataBrew に指示する Amazon S3 バケットとオブジェクトキー名のリストを表す JSON エンコードされた文字列。
- targetColumnNames — 結果の列名のリストを表す JSON エンコードされた文字列。

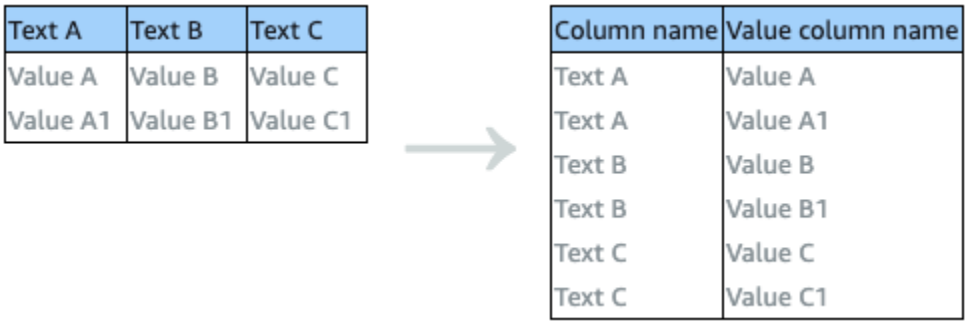
Example例

```
{
    "Action": {
        "Operation": "UNION",
        "Parameters": {
            "datasetsColumns": "[[\"assembly_session\",\"state_code\",
\"state_name\",\"year\",\"all_votes\",\"yes_votes\",\"no_votes\",\"abstain
\",\"idealpoint_estimate\",\"affinityscore_usa\",\"affinityscore_russia\",
\"affinityscore_china\",\"affinityscore_india\",\"affinityscore_brazil\",
\"affinityscore_israel\"],[\"assembly_session\",\"state_code\",\"state_name
\",null,null,null,null,null,null,null,null,null,null,null,null]]",
```

```
            "secondaryDatasetNames": "[\"votes\"]",
            "secondaryInputs": "[{\"S3InputDefinition\":{\"Bucket\":\"databrew-public-
datasets-us-east-1\",\"Key\":\"votes.csv\"}}]",
            "targetColumnNames": "[\"assembly_session\",\"state_code\",\"state_name\",
\"year\",\"all_votes\",\"yes_votes\",\"no_votes\",\"abstain\",\"idealpoint_estimate
\",\"affinityscore_usa\",\"affinityscore_russia\",\"affinityscore_china\",
\"affinityscore_india\",\"affinityscore_brazil\",\"affinityscore_israel\"]"
        }
    }
}
```

UNPIVOT

選択した行のすべての列値を、値を持つ個々の行に変換します。



Text A	Text B	Text C
Value A	Value B	Value C
Value A1	Value B1	Value C1

Column name	Value column name
Text A	Value A
Text A	Value A1
Text B	Value B
Text B	Value B1
Text C	Value C
Text C	Value C1

パラメータ

- `sourceColumns` — ピボットされない 1 つ以上の列のリストを表す JSON エンコードされた文字列。
- `unpivotColumn` — unpivot オペレーションの値列。
- `valueColumn` — ピボットされていない値を保持する列。

Example例

```
{
    "Action": {
        "Operation": "UNPIVOT",
        "Parameters": {
            "sourceColumns": "[\"idealpoint_estimate\"]",
            "unpivotColumn": "unpivoted_idealpoint_estimate",
            "valueColumn": "unpivoted_column_values"
```

```
        }
    }
}
```

データサイエンスレシピのステップ

これらのレシピステップを使用して、さまざまな視点からデータを集計および要約したり、高度な変換を実行したりできます。

トピック

- 二項化
- バケット化
- カテゴリマッピング
- ONE_HOT_ENCODING
- スケール
- 歪度
- トークン化

二項化

選択した数値ソース列のすべての値を取得し、しきい値と比較し、行ごとに 1 または 0 の新しい列を出力します。

パラメータ

- `sourceColumn` - 既存の列の名前。

  `targetColumn` – 作成される新しい列の名前。

  `threshold` – 値 0 または 1 を割り当てるためのしきい値を示す数値。

  `flip` – バイナリ割り当てを反転して、低い値が 1 に割り当てられ、高い値が 0 に割り当てられるようにするオプション。フリップパラメータが true の場合、しきい値以下の値は 1 になり、しきい値を超える値は 0 になります。

Example例

```
{
    "Action": {
        "Operation": "BINARIZATION",
        "Parameters": {
            "sourceColumn": "level",
            "targetColumn": "bin",
            "threshold": "100.0",
            "flip": "false"
        }
    }
}
```

バケット化

バケット化 (コンソールではビニングと呼ばれます) は、数値の列内の項目を受け取り、数値範囲によって定義されたビンにグループ化して、各行のビンを表示する新しい列を出力します。バケット化は、分割またはパーセンテージを使用して行うことができます。以下の最初の例では分割を使用し、2 番目の例ではパーセンテージを使用します。

パラメータ

- `sourceColumn` - 既存の列の名前。

  `targetColumn` – 作成される新しい列の名前。

  `bucketNames` – バケット名のリスト。

  `splits` – バケットレベルのリスト。バケットは連続しており、バケットの上限は次のバケットの上限になります。

  `percentage` – 各バケットはパーセンテージで記述されます。

Example分割の使用例

```
{
    "Action": {
        "Operation": "BUCKETIZATION",
        "Parameters": {
            "sourceColumn": "level",
            "targetColumn": "bin",
```

```
            "bucketNames": "[\"Bin1\",\"Bin2\",\"Bin3\"]",
            "splits": "[\"-Infinity\",\"2\",\"20\",\"Infinity\"]"
        }
    }
}
```

Exampleパーセンテージを使用した例

```
{
    "Action": {
        "Operation": "BUCKETIZATION",
        "Parameters": {
            "sourceColumn": "level",
            "targetColumn": "bin",
            "bucketNames": "[\"Bin1\",\"Bin2\"]",
            "percentage": "50"
        }
    }
}
```

カテゴリマッピング

1 つ以上のカテゴリ値を数値または他の値にマッピングします

パラメータ

- sourceColumn - 既存の列の名前。

  categoryMap – 値のカテゴリへのマップを表す JSON エンコードされた文字列。

  deleteOtherRows – の場合true、マッピングされていない行はすべてデータセットから削除されます。

  other – 指定した場合、マッピングされていない値はすべてこの値に置き換えられます。

  keepOthers – true の場合、マッピングされていない値はすべて同じままになります。

  mapType – マッピングされた列のデータ型。

  targetColumn – 結果を含む列の名前。

Example例

```
{
    "Action": {
        "Operation": "CATEGORICAL_MAPPING",
        "Parameters": {
            "categoryMap": "{\"United States of America\":\"1\",\"Canada\":\"2\",\"Cuba
\":\"3\",\"Haiti\":\"4\",\"Dominican Republic\":\"5\"}",
            "deleteOtherRows": "false",
            "keepOthers": "true",
            "mapType": "NUMERIC",
            "sourceColumn": "state_name",
            "targetColumn": "state_name_mapped"
        }
    }
}
```

ONE_HOT_ENCODING

n 個の数値列を作成します。n は選択したカテゴリ変数の一意の値の数です。

たとえば、 という名前の列を考えてみましょうshirt_size。シャツは、スモール、ミディアム、ラージ、またはエクストララージで利用できます。列データは次のようになります。

```
shirt_size
-----------
L
XL
M
S
M
M
S
XL
M
L
XL
M
```

このシナリオでは、 に 4 つの異なる値がありますshirt_size。したがって、 は 4 つの新しい列ONE_HOT_ENCODINGを生成します。各新しい列の名前は でshirt_size_*x*、 は個別のshirt_size 値*x*を表します。

shirt_size と 4 つの生成された列の結果は次のようになります。

```
shirt_size    shirt_size_S    shirt_size_M    shirt_size_L    shirt_size_XL
------------    ------------    ------------    ------------    -------------
L               0               0               1               0
XL              0               0               0               1
M               0               1               0               0
S               1               0               0               0
M               0               1               0               0
M               0               1               0               0
S               1               0               0               0
XL              0               0               0               1
M               0               1               0               0
L               0               0               1               0
XL              0               0               0               1
M               0               1               0               0
```

に指定する列には、最大 10 (10) 個の個別の値ONE_HOT_ENCODINGを指定できます。

パラメータ

- sourceColumn - 既存の列の名前。列には最大 10 個の個別の値を含めることができます。

Example例

```
{
    "RecipeAction": {
        "Operation": "ONE_HOT_ENCODING",
        "Parameters": {
            "sourceColumn": "shirt_size"
        }
    }
}
```

スケール

数値列のデータ範囲をスケールまたは正規化します。

パラメータ

- sourceColumn - 既存の列の名前。
- strategy – 列値に適用されるオペレーション。
  - MIN_MAX – 値を [0,1] の範囲に再スケーリングします。
  - SCALE_BETWEEN – 指定された 2 つの値の範囲に値を再スケーリングします。
  - MEAN_NORMALIZATION – 平均 (μ) が 0、標準偏差 (σ) が [-1, 1] の範囲内の 1 になるようにデータを再スケーリングします。
  - Z_SCORE – データ値を線形的にスケールして、平均 (μ) を 0、標準偏差 (σ) を 1 にします。外れ値の処理に最適です。
- targetColumn – 結果を含む列の名前。

Example例

```
{
    "Action": {
        "Operation": "NORMALIZATION",
        "Parameters": {
            "sourceColumn": "all_votes",
            "strategy": "MIN_MAX",
            "targetColumn": "all_votes_normalized"
        }
    }
}
```

歪度

データ値に変換を適用して、分散シェイプとそのスキューを変更します。

パラメータ

- sourceColumn - 既存の列の名前。

  targetColumn – 作成される新しい列の名前。

  skewFunction
  - ROOT – value-root を抽出します。ルートは valueパラメータで指定できます。

LOG – ログベース値。ログベースは valueパラメータで指定できます。

SQUARE – 二乗関数

value – skewFunction の引数。

Example例

```
{
    "RecipeAction": {
        "Operation": "SKEWNESS",
        "Parameters": {
            "sourceColumn": "level",
            "targetColumn": "bin",
            "skewFunction": "LOG",
            "value": "2.718281828"
        }
    }
}
```

トークン化

個々の単語や用語など、テキストを小さな単位またはトークンに分割します。

パラメータ

- sourceColumn - 既存の列の名前。
- delimiter — トークン化された単語の間に表示されるカスタム区切り文字。(デフォルトの動作では、各トークンをスペースで区切ります) 。
- expandContractions — の場合ENABLED、契約語を展開します。例えば、「いいえ」は「いいえ」になります。
- stemmingMode — 個々の小文字の単語や用語など、テキストを小さな単位またはトークンに分割します。PORTER | の 2 つのステミングモードを使用できますLANCASTER。
- stopWordRemovalMode — 、、 などの一般的な単語を削除します。
- customStopWords — の場合StopWordRemovalMode、 ではストップワードのカスタムリストを指定できます。

- `targetColumn` — 結果を含む列の名前。

Example例

```
{
    "Action": {
        "Operation": "TOKENIZATION",
        "Parameters": {
            "customStopWords": "[]",
            "delimiter": "- ",
            "expandContractions": "ENABLED",
            "sourceColumn": "dimensions",
            "stemmingMode": "PORTER",
            "stopWordRemovalMode": "DEFAULT",
            "targetColumn": "dimensions_tokenized"
        }
    }

}
```

数学関数

レシピアクションを操作する数学関数のリファレンストピックを以下に示します。

トピック

- MOD
- 乗算
- 否定
- PI
- POWER
- RADIANS
- RANDOM
- RANDOM_BETWEEN
- ROUND
- SIGN
- SQUARE_ROOT
- 減算

ABSOLUTE

新しい列の入力番号の絶対値を返します。絶対値は、正と負にかかわらず、数値がゼロからどれだけ離れているかです。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `targetColumn` – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "ABSOLUTE",
        "Parameters": {
            "sourceColumn": "freezingTemps",
            "targetColumn": "absValueOfFreezingTemps"
        }
    }
}
```

ADD

(+ sourceColumn2) またはsourceColumn1 (sourceColumn1 +) を使用して、新しい列の入力列値を合計しますvalue1。

パラメータ

- sourceColumn1 - 既存の列の名前。
- value1 – 数値。
- sourceColumn2 - 既存の列の名前。
- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "ADD",
        "Parameters": {
            "sourceColumn1": "weight_kg",
            "sourceColumn2": "height_cm",
            "targetColumn": "weight_plus_height"
        }
    }
}
```

CEILING

新しい列の入力 10 進数以上の最小整数数を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- value1 – 数値。
- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "CEILING",
        "Parameters": {
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_CEILING"
        }
    }
}
```

DEGREES

角度のラジアンを度に変換し、結果を新しい列に返します。

パラメータ

- sourceColumn - 既存の列の名前。
- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "DEGREES",
        "Parameters": {
            "sourceColumn": "height_cm",
            "targetColumn": "height_cm_DEGREES"
        }
    }
}
```

分割

1 つの入力番号を別の入力番号で割り、結果を新しい列に返します。

パラメータ

- sourceColumn1 - 既存の列の名前。
- value1 – 数値。

- sourceColumn2 - 既存の列の名前。
- value2 – 数値。
- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "DIVIDE",
        "Parameters": {
            "sourceColumn1": "height_cm",
            "targetColumn": "divide_by_2",
            "value2": "2"
        }
    }
}
```

指数

新しい列で n 度に引き上げられた Euler の数値を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "EXPONENT",
        "Parameters": {
            "sourceColumn": "age",
            "targetColumn": "age_EXPONENT"
        }
    }
}
```

FLOOR

新しい列の入力番号以上の最大の整数を返します。

パラメータ

- sourceColumn1 - 既存の列の名前。
- value – 数値。
- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "FLOOR",
        "Parameters": {
            "targetColumn": "FLOOR Column 1",
            "value": "42"
        }
    }
}
```

IS_EVEN

ソース列または値が偶数であるかどうかを示すブール値を新しい列に返します。ソース列または値が小数の場合、結果は false です。

パラメータ

- sourceColumn - 既存の列の名前。
- targetColumn – 作成される新しい列の名前。
- trueString – 値が偶数かどうかを示す文字列。
- falseString – 値が偶数ではないかどうかを示す文字列。

Example例

```
{
```

```
    "RecipeAction": {
        "Operation": "IS_EVEN",
        "Parameters": {
            "falseString": "Value is odd",
            "sourceColumn": "height_cm",
            "targetColumn": "height_cm_IS_EVEN",
            "trueString": "Value is even"
        }
    }
}
```

IS_ODD

ソース列または値が奇数であるかどうかを示すブール値を新しい列に返します。ソース列または値が小数の場合、結果は false です。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `targetColumn` – 作成される新しい列の名前。
- `trueString` – 値が奇数であるかどうかを示す文字列。
- `falseString` – 値が奇数ではないかどうかを示す文字列。

Example例

```
{
    "RecipeAction": {
        "Operation": "IS_ODD",
        "Parameters": {
            "falseString": "Value is even",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_IS_ODD",
            "trueString": "Value is odd"
        }
    }
}
```

LN

新しい列の値の自然対数 (Euler の数値) を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "LN",
        "Parameters": {
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_LN"
        }
    }
}
```

LOG

新しい列の値の対数を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- targetColumn – 作成される新しい列の名前。
- base – 対数の基礎。デフォルトは 10 です。

Example例

```
{
    "RecipeAction": {
        "Operation": "LOG",
        "Parameters": {
            "base": "10",
            "sourceColumn": "age",
            "targetColumn": "age_LOG"
        }
```

```
    }
}
```

MOD

1 つの数値が新しい列の別の数値である割合を返します。

パラメータ

- sourceColumn1 - 既存の列の名前。
- sourceColumn2 - 既存の列の名前。
- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "MOD",
        "Parameters": {
            "sourceColumn1": "start_date",
            "sourceColumn2": "end_date",
            "targetColumn": "MOD Column 1"
        }
    }
}
```

乗算

2 つの数値を乗算し、結果を新しい列に返します。

パラメータ

- sourceColumn1 - 既存の列の名前。
- value1 – 数値。
- sourceColumn2 - 既存の列の名前。
- value2 – 数値。
- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "MULTIPLY",
        "Parameters": {
            "sourceColumn1": "hourly_rate",
            "sourceColumn2": "hours",
            "targetColumn": "total_pay"
        }
    }
}
```

否定

値を否定し、結果を新しい列に返します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `targetColumn` – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "NEGATE",
        "Parameters": {
            "sourceColumn": "age",
            "targetColumn": "age_NEGATE"
        }
    }
}
```

PI

新しい列で pi (3.141592653589793) の値を返します。

パラメータ

- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "PI",
        "Parameters": {
            "targetColumn": "PI Column 1"
        }
    }
}
```

POWER

数値の値を新しい列の指数の累乗に返します。

パラメータ

- sourceColumn - 既存の列の名前。
- value – 値を引き上げる数値。
- targetColumn – 作成される新しい列の名前。
- exponent – 値が引き上げられる電力。

> **Note**
>
> sourceColumn または value を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "POWER",
```

```
        "Parameters": {
            "exponent": "3",
            "sourceColumn": "age",
            "targetColumn": "age_cubed"
        }
    }
}
```

RADIANS

度数をラジアン (180/pi で除算) に変換し、新しい列の値を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "RADIANS",
        "Parameters": {
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_RADIANS"
        }
    }
}
```

RANDOM

新しい列で 0 ~ 1 の乱数を返します。

パラメータ

- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "RANDOM",
        "Parameters": {
            "targetColumn": "RANDOM Column 1"
        }
    }
}
```

RANDOM_BETWEEN

新しい列で、 は指定された下限 (含む) と指定された上限 (含む) の間の乱数を返します。

パラメータ

- `lowerBound` – 乱数範囲の下限。
- `upperBound` – 乱数範囲の上限。
- `targetColumn` – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "RANDOM_BETWEEN",
        "Parameters": {
            "lowerBound": "1",
            "targetColumn": "RANDOM_BETWEEN Column 1",
            "upperBound": "100"
        }
    }
}
```

ROUND

数値を新しい列の最も近い整数に丸めます。小数が 0.5 以上になると切り上げられます。

パラメータ

- `sourceColumn` - 既存の列の名前。

- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "ROUND",
        "Parameters": {
            "sourceColumn": "rating",
            "targetColumn": "rating_ROUND"
        }
    }
}
```

SIGN

値が 0 より小さい場合は -1、値が 0 の場合は 0、値が 0 より大きい場合は +1 の新しい列を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "SIGN",
        "Parameters": {
            "sourceColumn": "age",
            "targetColumn": "age_SIGN"
        }
    }
}
```

SQUARE_ROOT

新しい列の値の平方根を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "SQUARE_ROOT",
        "Parameters": {
            "sourceColumn": "age",
            "targetColumn": "age_SQUARE_ROOT"
        }
    }
}
```

減算

ある数値を別の数値から減算し、結果を新しい列に返します。

パラメータ

- sourceColumn1 - 既存の列の名前。
- value1 – 数値。
- sourceColumn2 - 既存の列の名前。
- value2 – 数値。
- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "SUBTRACT",
        "Parameters": {
            "sourceColumn1": "weight_kg",
            "targetColumn": "weight_minus_10_kg",
```

```
            "value2": "10"
        }
    }
}
```

集計関数

以下に、レシピアクションを操作する集計関数のリファレンストピックを示します。

トピック

- ANY
- AVERAGE
- COUNT
- COUNT_DISTINCT
- KTH_LARGEST
- KTH_LARGEST_UNIQUE
- MAX
- MEDIAN
- MIN
- MODE
- STANDARD_DEVIATION
- SUM
- 分散

ANY

新しい列で選択したソース列の値を返します。空の値と null 値は無視されます。

パラメータ

- `sourceColumns` – 既存の列のリストを表す JSON エンコードされた文字列。
- `targetColumn` – 新しく作成された列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "ANY",
        "Parameters": {
            "sourceColumns": "[\"age\",\"last_name\"]",
            "targetColumn": "ANY Column 1"
        }
    }
}
```

AVERAGE

ソース列の値の平均を計算し、結果を新しい列で返します。数値以外のものは無視されます。

パラメータ

- `sourceColumns` – 既存の列のリストを表す JSON エンコードされた文字列。
- `targetColumn` – 新しく作成された列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "AVERAGE",
        "Parameters": {
            "sourceColumns": "[\"age\",\"weight_kg\",\"height_cm\"]",
            "targetColumn": "AVERAGE Column 1"
        }
    }
}
```

COUNT

新しい列で選択したソース列の値の数を返します。空の値と null 値は無視されます。

パラメータ

- `sourceColumns` – 既存の列のリストを表す JSON エンコードされた文字列。
- `targetColumn` – 新しく作成された列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "COUNT",
        "Parameters": {
            "sourceColumns": "[\"ANY Column 1\",\"birth_date\",\"last_name\"]",
            "targetColumn": "COUNT Column 1"
        }
    }
}
```

COUNT_DISTINCT

新しい列で選択したソース列から異なる値の総数を返します。空の値と null 値は無視されます。

パラメータ

- `sourceColumns` – 既存の列のリストを表す JSON エンコードされた文字列。
- `targetColumn` – 新しく作成された列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "COUNT_DISTINCT",
        "Parameters": {
            "sourceColumns": "[\"long_name\",\"weight_kg\"]",
            "targetColumn": "COUNT_DISTINCT Column 1"
        }
    }
}
```

KTH_LARGEST

新しい列で選択したソース列から k 番目に大きい数値を返します。

パラメータ

- `sourceColumns` – 既存の列のリストを表す JSON エンコードされた文字列。

- targetColumn – 新しく作成された列の名前。
- value – k を表す数値。

Example例

```
{
    "RecipeAction": {
        "Operation": "KTH_LARGEST",
        "Parameters": {
            "sourceColumns": "[\"height_cm\",\"weight_kg\",\"age\"]",
            "targetColumn": "KTH_LARGEST Column 1",
            "value": "2"
        }
    }
}
```

KTH_LARGEST_UNIQUE

新しい列の選択したソース列から k 番目に大きい一意の数値を返します。

パラメータ

- sourceColumns – 既存の列のリストを表す JSON エンコードされた文字列。
- targetColumn – 新しく作成された列の名前。

  value – k を表す数値。

Example例

```
{
    "RecipeAction": {
        "Operation": "KTH_LARGEST_UNIQUE",
        "Parameters": {
            "sourceColumns": "[\"age\",\"height_cm\",\"weight_kg\"]",
            "targetColumn": "KTH_LARGEST_UNIQUE Column 1",
            "value": "3"
        }
    }
```

```
}
```

MAX

新しい列で選択したソース列の最大数値を返します。数値以外のものは無視されます。

パラメータ

- `sourceColumns` – 既存の列のリストを表す JSON エンコードされた文字列。
- `targetColumn` – 新しく作成された列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "MAX",
        "Parameters": {
            "sourceColumns": "[\"age\",\"height_cm\",\"weight_kg\"]",
            "targetColumn": "MAX Column 1"
        }
    }
}
```

MEDIAN

新しい列の選択したソース列から、ソートされた数値グループの中間数の中央値を返します。数値以外のものは無視されます。

パラメータ

- `sourceColumns` – 既存の列のリストを表す JSON エンコードされた文字列。
- `targetColumn` – 新しく作成された列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "MEDIAN",
```

```
        "Parameters": {
            "sourceColumns": "[\"age\",\"years_in_service\"]",
            "targetColumn": "MEDIAN Column 1"
        }
    }
}
```

MIN

新しい列で選択したソース列から最小値を返します。数値以外のものは無視されます。

パラメータ

- `sourceColumns` – 既存の列のリストを表す JSON エンコードされた文字列。
- `targetColumn` – 新しく作成された列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "MIN",
        "Parameters": {
            "sourceColumns": "[\"age\",\"height_cm\",\"weight_kg\"]",
            "targetColumn": "MIN Column 1"
        }
    }
}
```

MODE

新しい列で選択したソース列から、最も頻繁に表示される数値である モードを返します。数値以外のものは無視されます。複数のモードでは、モードはモーダル関数を使用して計算されます。

パラメータ

- `sourceColumns` – 既存の列のリストを表す JSON エンコードされた文字列。
- `targetColumn` – 新しく作成された列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "MODE",
        "Parameters": {
            "modeType": "MINIMUM",
            "sourceColumns": "[\"years_in_service\",\"age\"]",
            "targetColumn": "MODE Column 1"
        }
    }
}
```

STANDARD_DEVIATION

新しい列で選択したソース列からの標準偏差を返します。

パラメータ

- `sourceColumns` – 既存の列のリストを表す JSON エンコードされた文字列。
- `targetColumn` – 新しく作成された列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "STANDARD_DEVIATION",
        "Parameters": {
            "sourceColumns": "[\"years_in_sservice\",\"age\"]",
            "targetColumn": "STANDARD_DEVIATION Column 1"
        }
    }
}
```

SUM

新しい列で選択したソース列の値の合計を返します。数値以外の値は 0 として扱われます。

パラメータ

- `sourceColumns` – 既存の列のリストを表す JSON エンコードされた文字列。
- `targetColumn` – 新しく作成された列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "SUM",
        "Parameters": {
            "sourceColumns": "[\"age\",\"years_in_service\"]",
            "targetColumn": "SUM Column 1"
        }
    }
}
```

分散

新しい列で選択したソース列からの分散を返します。分散は として定義されますVar(X) = [Sum ((X - mean(X))^2)]/Count(X)。

パラメータ

- sourceColumns – 既存の列のリストを表す JSON エンコードされた文字列。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "VARIANCE",
        "Parameters": {
            "sourceColumns": "[\"age\",\"years_in_service\"]",
            "targetColumn": "VARIANCE Column 1"
        }
    }
}
```

テキスト関数

レシピアクションを操作するテキスト関数のリファレンストピックを以下に示します。

トピック

CHAR

新しい列で、ソース列の各整数の Unicode 文字、またはカスタム整数値を返します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – Unicode 値を表す整数。

- targetColumn – 作成される新しい列の名前。

> **Note**
>
> sourceColumn または value を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "CHAR",
        "Parameters": {
            "sourceColumn": "age",
            "targetColumn": "age_char"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "CHAR",
        "Parameters": {
            "value": 42,
            "targetColumn": "asterisk"
        }
    }
}
```

ENDS_WITH

指定された数の右端の文字、またはカスタム文字列がパターンと一致する場合、新しい列trueで を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- value – 評価する文字列。

- pattern – 文字列の末尾と一致する必要がある正規表現。
- targetColumn – 作成される新しい列の名前。

> **Note**
>
> sourceColumn または value を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "ENDS_WITH",
        "Parameters": {
            "sourceColumn": "nationality",
            "pattern": "[Ss]",
            "targetColumn": "nationality_ends_with"
        }
    }
}
```

正確な

次のいずれかが入力された新しい列を作成します。

- True 列 (または値) 内の 1 つの文字列が、別の列 (または値) 内の別の文字列と正確に一致する場合。
- False 一致するものがない場合。

パラメータ

- sourceColumn1 - 既存の列の名前。
- sourceColumn2 - 既存の列の名前。
- value1 – 評価する文字列。
- value2 – 評価する文字列。
- targetColumn – 作成される新しい列の名前。

> **Note**
>
> 次のいずれかの組み合わせのみを指定できます。
>
> - の両方sourceColumn*N*。
> - の 1 つsourceColumn*N*と の 1 つvalue*N*。
> - の両方value*N*。

Example例

```
{
    "RecipeAction": {
        "Operation": "EXACT",
        "Parameters": {
            "sourceColumn1": "nationality",
            "value2": "Argentina",
            "targetColumn": "nationality_exact"
        }
    }
}
```

検索

左から右を検索して、ソース列またはカスタム値から指定された文字列に一致する文字列を検索し、その結果を新しい列に返します。

パラメータ

- sourceColumn - 既存の列の名前。
- pattern – 検索する正規表現。
- position – 文字列の左端から始まる文字の位置。
- ignoreCase – の場合true、大文字と小文字の違い (大文字と小文字の間) は無視します。厳密なマッチングを適用するには、false代わりに を使用します。
- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "FIND",
        "Parameters": {
            "sourceColumn": "city",
            "pattern": "[AEIOU]",
            "position": "1",
            "ignoreCase": "false",
            "targetColumn": "begins_with_a_vowel"
        }
    }
}
```

LEFT

文字数を指定すると、 はソース列またはカスタム文字列から文字列内の左端の文字数を取得し、新しい列で指定された左端の文字数を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- value – 評価する文字列。
- position – 文字列の左端から始まる文字の位置。
- targetColumn – 作成される新しい列の名前。

> **Note**
>
> sourceColumn または value を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "LEFT",
        "Parameters": {
            "position": "3",
            "sourceColumn": "city",
```

```
            "targetColumn": "city_left"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "LEFT",
        "Parameters": {
            "position": "5",
            "value": "How now brown cow",
            "targetColumn": "how_now_5_left_chars"
        }
    }
}
```

LEN

ソース列またはカスタム文字列の文字列の長さを新しい列で返します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – 評価する文字列。
- `targetColumn` – 作成される新しい列の名前。

> **Note**
>
> `sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "LEN",
        "Parameters": {
            "sourceColumn": "last_name",
```

```
            "targetColumn": "last_name_len"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "LEN",
        "Parameters": {
            "value": "Hello",
            "targetColumn": "hello_len"
        }
    }
}
```

LOWER

ソース列の文字列またはカスタム文字列のすべてのアルファベット文字を小文字に変換し、結果を新しい列で返します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – 評価する文字列。
- `targetColumn` – 作成される新しい列の名前。

> **Note**
>
> `sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "LOWER",
        "Parameters": {
            "sourceColumn": "last_name",
```

```
            "targetColumn": "last_name_lower"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "LOWER",
        "Parameters": {
            "value": "GOODBYE",
            "targetColumn": "goodbye_lower"
        }
    }
}
```

MERGE_COLUMNS_AND_VALUES

ソース列の文字列を連結し、結果を新しい列に返します。マージされた値の間に区切り文字を挿入できます。

パラメータ

- `sourceColumns` – JSON エンコード形式の 2 つ以上の既存の列の名前。
- `delimiter` - オプション。2 つのソース列の各値の間に配置する 1 つ以上の文字。
- `targetColumn` – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "MERGE_COLUMNS_AND_VALUES",
        "Parameters": {
            "sourceColumns": "[\"last_name\",\"birth_date\"]",
            "delimiter": " was born on: ",
            "targetColumn": "merged_column"
        }
    }
}
```

適切

ソース列またはカスタム値の文字列のすべてのアルファベット文字を適切な大文字に変換し、結果を新しい列で返します。

大文字とも呼ばれる適切な場合、各単語の最初の文字は大文字になり、残りの単語は小文字に変換されます。例: クイックブラウンフォックスがフェンスを飛び越えた

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – 評価する文字列。
- `targetColumn` – 作成される新しい列の名前。

> **Note**
>
> `sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "PROPER",
        "Parameters": {
            "sourceColumn": "first_name",
            "targetColumn": "first_name_proper"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "PROPER",
        "Parameters": {
            "value": "MR. H. SMITH, ESQ.",
            "targetColumn": "formal_name_proper"
        }
    }
```

```
}
```

REMOVE_SYMBOLS

ソース列またはカスタム文字列の文字列から文字、数字、アクセント付きラテン文字、空白以外の文字を削除し、その結果を新しい列で返します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – 評価する文字列。
- `targetColumn` – 作成される新しい列の名前。

> **Note**
>
> `sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "REMOVE_SYMBOLS",
        "Parameters": {
            "sourceColumn": "info_url",
            "targetColumn": "info_url_remove_symbols"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REMOVE_SYMBOLS",
        "Parameters": {
            "value": "$&#$&HEY!#@@",
            "targetColumn": "without_symbols"
        }
    }
```

```
}
```

REMOVE_WHITESPACE

ソース列またはカスタム文字列の文字列から空白を削除し、結果を新しい列で返します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – 評価する文字列。
- `targetColumn` – 作成される新しい列の名前。

> **Note**
>
> `sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "REMOVE_WHITESPACE",
        "Parameters": {
            "sourceColumn": "job_desc",
            "targetColumn": "job_desc_remove_whitespace"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REMOVE_WHITESPACE",
        "Parameters": {
            "value": "This string has spaces in it",
            "targetColumn": "string_without_spaces"
        }
    }
}
```

REPEAT_STRING

ソース列またはカスタム入力値の文字列を指定された回数繰り返し、その結果を新しい列に返します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – 評価する文字列。
- `count` – 文字列を繰り返す回数。
- `targetColumn` – 作成される新しい列の名前。

> **Note**
>
> `sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "REPEAT_STRING",
        "Parameters": {
            "count": 3,
            "sourceColumn": "last_name",
            "targetColumn": "last_name_repeat_string"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REPEAT_STRING",
        "Parameters": {
            "count": 80,
            "value": "*",
            "targetColumn": "80_stars"
        }
```

```
    }
}
```

RIGHT

文字数を指定すると、 はソース列またはカスタム文字列から文字列の右端の文字数を取得し、新しい列で指定された右端の文字数を返します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – 評価する文字列。
- `position` – 文字列の右側から始まる文字の位置。
- `targetColumn` – 作成される新しい列の名前。

> **Note**
>
> `sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "RIGHT",
        "Parameters": {
            "sourceColumn": "nationality",
            "position": "3",
            "targetColumn": "nationality_right"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "RIGHT",
        "Parameters": {
            "value": "United States of America",
```

```
            "position": "7",
            "targetColumn": "usa_right"
        }
    }
}
```

RIGHT_FIND

右から左を検索して、ソース列またはカスタム値から指定された文字列に一致する文字列を検索し、その結果を新しい列に返します。

パラメータ

- sourceColumn - 既存の列の名前。
- pattern – 検索する正規表現。
- position – 文字列の右端から始まる文字の位置。
- ignoreCase – の場合true、大文字と小文字の違い (大文字と小文字の間) は無視します。厳密なマッチングを適用するには、false代わりに を使用します。
- targetColumn – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "RIGHT_FIND",
        "Parameters": {
            "sourceColumn": "nationality",
            "pattern": "s",
            "position": "1",
            "ignoreCase": "true",
            "targetColumn": "ends_with_an_s"
        }
    }
}
```

STARTS_WITH

指定された数の左端の文字、またはカスタム文字列がパターンと一致する場合、新しい列trueで を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- value – 評価する文字列。
- pattern – 文字列の先頭と一致する必要がある正規表現。
- targetColumn – 作成される新しい列の名前。

> **Note**
>
> sourceColumn または value を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "STARTS_WITH",
        "Parameters": {
            "sourceColumn": "nationality",
            "pattern": "[AEIOU]",
            "targetColumn": "nationality_starts_with"
        }
    }
}
```

STRING_GREATER_THAN

次のいずれかが入力された新しい列を作成します。

- True 列 (または値) 内の 1 つの文字列が、別の列 (または値) 内の別の文字列より大きい場合。
- False 一致するものがない場合。

パラメータ

- sourceColumn1 - 既存の列の名前。
- sourceColumn2 - 既存の列の名前。

- value1 – 評価する文字列。
- value2 – 評価する文字列。
- targetColumn – 作成される新しい列の名前。

> **Note**
>
> 次のいずれかの組み合わせのみを指定できます。
>
> - の両方sourceColumn*N*。
> - の 1 つsourceColumn*N*と の 1 つvalue*N*。
> - の両方value*N*。

Example例

```
{
    "RecipeAction": {
        "Operation": "STRING_GREATER_THAN",
        "Parameters": {
            "sourceColumn1": "first_name",
            "sourceColumn2": "last_name",
            "targetColumn": "string_greater_than"
        }
    }
}
```

STRING_GREATER_THAN_EQUAL

次のいずれかが入力された新しい列を作成します。

- True 列 (または値) 内の 1 つの文字列が、別の列 (または値) 内の別の文字列以上である場合。
- False 一致するものがない場合。

パラメータ

- sourceColumn1 - 既存の列の名前。
- sourceColumn2 - 既存の列の名前。

- value1 – 評価する文字列。
- value2 – 評価する文字列。
- targetColumn – 作成される新しい列の名前。

> **Note**
>
> 次のいずれかの組み合わせのみを指定できます。
>
> - の両方sourceColumn*N*。
> - の 1 つsourceColumn*N*と の 1 つvalue*N*。
> - の両方value*N*。

Example例

```
{
    "RecipeAction": {
        "Operation": "STRING_GREATER_THAN_EQUAL",
        "Parameters": {
            "sourceColumn1": "nationality",
            "targetColumn": "string_greater_than_equal",
            "value2": "s"
        }
    }
}
```

STRING_LESS_THAN

次のいずれかが入力された新しい列を作成します。

- True 列 (または値) 内の 1 つの文字列が、別の列 (または値) 内の別の文字列よりも小さい場合。
- False 一致するものがない場合。

パラメータ

- sourceColumn1 - 既存の列の名前。
- sourceColumn2 - 既存の列の名前。

- value1 – 評価する文字列。
- value2 – 評価する文字列。
- targetColumn – 作成される新しい列の名前。

> **Note**
>
> 次のいずれかの組み合わせのみを指定できます。
>
> - の両方sourceColumn*N*。
> - の 1 つsourceColumn*N*と の 1 つvalue*N*。
> - の両方value*N*。

Example例

```
{
    "RecipeAction": {
        "Operation": "STRING_LESS_THAN",
        "Parameters": {
            "sourceColumn1": "first_name",
            "sourceColumn2": "last_name",
            "targetColumn": "string_less_than"
        }
    }
}
```

STRING_LESS_THAN_EQUAL

次のいずれかが入力された新しい列を作成します。

- True 列内の 1 つの文字列 (または値) が、別の列 (または値) 内の別の文字列以下である場合。
- False 一致するものがない場合。

パラメータ

- sourceColumn1 - 既存の列の名前。
- sourceColumn2 - 既存の列の名前。

- value1 – 評価する文字列。
- value2 – 評価する文字列。
- targetColumn – 作成される新しい列の名前。

> **Note**
>
> 次のいずれかの組み合わせのみを指定できます。
>
> - の両方sourceColumn*N*。
> - の 1 つsourceColumn*N*と の 1 つvalue*N*。
> - の両方value*N*。

Example例

```
{
    "RecipeAction": {
        "Operation": "STRING_LESS_THAN_EQUAL",
        "Parameters": {
            "sourceColumn1": "first_name",
            "targetColumn": "string_less_than_equal",
            "value2": "s"
        }
    }
}
```

SUBSTRING

ユーザー定義の開始インデックス値と終了インデックス値に基づいて、ソース列で指定された文字列の一部またはすべてを新しい列に返します。

パラメータ

- sourceColumn - 既存の列の名前。
- startPosition – 文字列の左端から始まる文字の位置。
- endPosition – 文字列の左端から終わる文字の位置。
- targetColumn – 作成される新しい列の名前。

> **Note**
>
> sourceColumn または value を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "SUBSTRING",
        "Parameters": {
            "sourceColumn": "last_name",
            "startPosition": "5",
            "endPosition": "8",
            "targetColumn": "chars_5_through_8"
        }
    }
}
```

TRIM

ソース列またはカスタム文字列の文字列から先頭と末尾の空白を削除し、結果を新しい列に返します。単語間のスペースは削除されません。

パラメータ

- sourceColumn - 既存の列の名前。
- value – 評価する文字列。
- targetColumn – 作成される新しい列の名前。

> **Note**
>
> sourceColumn または value を指定できます。両方を指定することはできません。

Example例

```
{
```

```
    "RecipeAction": {
        "Operation": "TRIM",
        "Parameters": {
            "sourceColumn": "nationality",
            "targetColumn": "nationality_trim"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "TRIM",
        "Parameters": {
            "value": "   This string should be trimmed       ",
            "targetColumn": "string_trimmed"
        }
    }
}
```

UNICODE

新しい列で、ソース列の文字列の最初の文字またはカスタム文字列の Unicode インデックス値を返します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – 評価する文字列。
- `targetColumn` – 作成される新しい列の名前。

> **Note**
>
> `sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
```

```
    "RecipeAction": {
        "Operation": "UNICODE",
        "Parameters": {
            "sourceColumn": "first_name",
            "targetColumn": "first_name_unicode"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "UNICODE",
        "Parameters": {
            "value": "?",
            "targetColumn": "sixty_three"
        }
    }
}
```

UPPER

ソース列の文字列またはカスタム文字列のすべてのアルファベット文字を大文字に変換し、結果を新しい列で返します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – 評価する文字列。
- `targetColumn` – 作成される新しい列の名前。

> **Note**
>
> `sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
```

```
    "RecipeAction": {
        "Operation": "UPPER",
        "Parameters": {
            "sourceColumn": "last_name",
            "targetColumn": "last_name_upper"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "UPPER",
        "Parameters": {
            "value": "a string of lowercase letters",
            "targetColumn": "string_upper"
        }
    }
}
```

日付および時刻関数

レシピアクションを操作する日付と時刻関数のリファレンストピックを以下に示します。

トピック

- CONVERT_TIMEZONE
- DATE
- DATE_ADD
- DATE_DIFF
- DATE_FORMAT
- DATE_TIME
- DAY
- HOUR
- ミリ秒
- MINUTE
- MONTH

- MONTH_NAME
- NOW
- 四半期
- SECOND
- TIME
- 本日
- UNIX_TIME
- UNIX_TIME_FORMAT
- WEEK_DAY
- WEEK_NUMBER
- YEAR

CONVERT_TIMEZONE

ソース列の時刻値を、指定されたタイムゾーンに基づいて新しい列に変換します。

パラメータ

- `sourceColumn` - 既存の列の名前。ソース列のタイプは `string`、`date`、または `timestamp` です。
- `fromTimeZone` – ソース値のタイムゾーン。何も指定しない場合、デフォルトのタイムゾーンは UTC です。
- `toTimeZone` – 変換するタイムゾーン。何も指定しない場合、デフォルトのタイムゾーンは UTC です。
- `targetColumn` – 新しく作成された列の名前。
- `dateTimeFormat` - オプション。日付の形式文字列。形式が指定されていない場合は、デフォルトの形式 が使用されます`yyyy-mm-dd HH:MM:SS`。

Example例

```
{
    "RecipeAction": {
        "Operation": "CONVERT_TIMEZONE",
```

```
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "fromTimeZone": "UTC+08:00",
            "toTimeZone": "UTC+08:00",
            "targetColumn": "DATETIME Column CONVERT_TIMEZONE",
            "dateTimeFormat": "yyyy-mm-dd HH:MM:SS"
        }
    }
}
```

DATE

ソース列または指定された値から、日付値を含む新しい列を作成します。

パラメータ

- dateTimeFormat - オプション。新しい列に表示される日付の形式文字列。この文字列が指定されていない場合、デフォルトの形式は ですyyyy-mm-dd HH:MM:SS。
- dateTimeParameters – 日付と時刻のコンポーネントを表す JSON エンコードされた文字列。
  - year
  - value
  - month
  - day
  - hour
  - second

  各コンポーネントは、次のいずれかを指定する必要があります。
  - sourceColumn - 既存の列の名前。
  - value – 評価する文字列。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "DATE",
        "Parameters": {
```

```
            "dateTimeFormat": "mm/dd/yy",
            "dateTimeParameters": "{\"year\":{\"value\":\"2019\"},\"month\":{\"value\":
\"12\"},\"day\":{\"value\":\"31\"},\"hour\":{},\"minute\":{},\"second\":{}}",
            "targetColumn": "DATE Column 1"
        }
    }
}
```

DATE_ADD

ソース列または値から日付に年、月、または日を追加し、結果を含む新しい列を作成します。

パラメータ

- sourceColumn - 既存の列の名前。
- value – 評価する文字列。
- units – 日付を調整するための測定単位。有効な値は、MONTHS、YEARS、、MILLISECONDS、QUARTERS、HOURS、MICROSECONDS、、DAYS、WEEKSSECよび ですMINUTES。
- dateAddValue – 日付unitsに追加する の数。
- dateTimeFormat - オプション。新しい列に表示される日付の形式文字列。指定されなかった場合、デフォルト値は yyyy-mm-dd HH:MM:SS です。
- targetColumn – 新しく作成された列の名前。

> **Note**
>
> sourceColumn または value を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "DATE_ADD",
        "Parameters": {
            "sourceColumn": "DATE Column 1",
            "units": "DAYS",
```

```
            "dateAddValue": "14",
            "dateTimeFormat": "mm/dd/yyyy",
            "targetColumn": "DATE Column 1_DATEADD"
        }
    }
}
```

DATE_DIFF

2 つの日付の差を含む新しい列を作成します。

パラメータ

- sourceColumn1 - 既存の列の名前。
- sourceColumn2 - 既存の列の名前。
- value1 – 評価する文字列。
- value2 – 評価する文字列。
- units – の単位は、日付の差を表します。有効な値は、MONTHS、YEARS、、MILLISECONDS、QUARTERS、HOURS、MICROSECONDS、、DAYS、WEEKSSECよび ですMINUTES。
- targetColumn – 新しく作成された列の名前。

> **Note**
>
> 次のいずれかの組み合わせのみを指定できます。
>
> - sourceColumn1 と の両方sourceColumn2。
> - sourceColumn1 または の 1 つsourceColumn2、および value1 または の 1 つvalue2。
> - value1 と の両方value2。

Example例

```
{
    "RecipeAction": {
        "Operation": "DATE_DIFF",
```

```
        "Parameters": {
            "value1": "2020-01-01",
            "value2": "2020-10-06",
            "units": "DAYS",
            "targetColumn": "DATEDIFF Column 1"
        }
    }
}
```

DATE_FORMAT

日付を表す文字列から、特定の形式の日付を含む新しい列を作成します。

パラメータ

- sourceColumn - 既存の列の名前。
- value – 評価する文字列。
- dateTimeFormat - オプション。新しい列に表示される日付の形式文字列。指定されなかった場合、デフォルト値は yyyy-mm-dd HH:MM:SS です。
- targetColumn – 新しく作成された列の名前。

> **Note**
>
> sourceColumn または value を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "DATE_FORMAT",
        "Parameters": {
            "sourceColumn": "DATE Column 1",
            "dateTimeFormat": "month*dd*yyyy",
            "targetColumn": "DATE Column 1_DATEFORMAT"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "DATE_FORMAT",
        "Parameters": {
            "value": "22:10:47",
            "dateTimeFormat": "HH:MM:SS",
            "targetColumn": "formatted_date_value"
        }
    }
}
```

DATE_TIME

ソース列または指定された値から、日付と時刻の値を含む新しい列を作成します。

パラメータ

- dateTimeFormat - オプション。新しい列に表示される日付の形式文字列。この文字列が指定されていない場合、デフォルトの形式は ですyyyy-mm-dd HH:MM:SS。
- dateTimeParameters – 日付と時刻のコンポーネントを表す JSON エンコードされた文字列。
  - year
  - value
  - month
  - day
  - hour
  - second

  各コンポーネントは、次のいずれかを指定する必要があります。
  - sourceColumn - 既存の列の名前。
  - value – 評価する文字列。

Example例

```
{
    "RecipeAction": {
        "Operation": "DATE_TIME",
        "Parameters": {
```

```
            "dateTimeFormat": "yyyy-mm-dd HH:MM:SS",
            "dateTimeParameters": "{\"year\":{\"value\":\"2010\"},\"month\":{\"value\":
\"5\"},\"day\":{\"value\":\"21\"},\"hour\":{\"value\":\"13\"},\"minute\":{\"value\":
\"34\"},\"second\":{\"value\":\"25\"}}",
            "targetColumn": "DATETIME Column 1"
        }
    }
}
```

DAY

日付を表す文字列から、日付を含む新しい列を作成します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – 評価する文字列。
- `targetColumn` – 新しく作成された列の名前。

> **Note**
>
> `sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "DAY",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_DAY"
        }
    }
}
```

HOUR

日付を表す文字列から、時間値を含む新しい列を作成します。

パラメータ

- sourceColumn - 既存の列の名前。
- value – 評価する文字列。
- targetColumn – 新しく作成された列の名前。

> **Note**
>
> sourceColumn または value を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "HOUR",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_HOUR"
        }
    }
}
```

ミリ秒

ソース列または入力値からミリ秒値を含む新しい列を作成します。

パラメータ

- sourceColumn - 既存の列の名前。ソース列のタイプは string、date、または timestamp です。
- value – 評価する文字列。
- targetColumn – 新しく作成された列の名前。

> **Note**
>
> sourceColumn または value を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "MILLISECOND",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_MILLISECOND"
        }
    }
}
```

MINUTE

日付を表す文字列から、分値を含む新しい列を作成します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – 評価する文字列。
- `targetColumn` – 新しく作成された列の名前。

> **Note**
>
> `sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "MINUTE",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_MINUTE"
        }
    }
```

```
}
```

MONTH

日付を表す文字列から、月の数を含む新しい列を作成します。

パラメータ

- sourceColumn - 既存の列の名前。
- value – 評価する文字列。
- targetColumn – 新しく作成された列の名前。

> **Note**
>
> sourceColumn または value を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "MONTH",
        "Parameters": {
            "value": "2018-05-27",
            "targetColumn": "MONTH Column 1"
        }
    }
}
```

MONTH_NAME

日付を表す文字列から、月の名前を含む新しい列を作成します。

パラメータ

- sourceColumn - 既存の列の名前。
- value – 評価する文字列。

- targetColumn – 新しく作成された列の名前。

> **Note**
>
> sourceColumn または value を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "MONTH_NAME",
        "Parameters": {
            "value": "2018-05-27",
            "targetColumn": "MONTHNAME Column 1"
        }
    }
}
```

NOW

現在の日付と時刻を 形式で含む新しい列を作成しますyyyy-mm-dd HH:MM:SS。

パラメータ

- timeZone – タイムゾーンの名前。タイムゾーンが指定されていない場合、デフォルトは協定世界時 (UTC) です。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "NOW",
        "Parameters": {
            "timeZone": "US/Pacific",
            "targetColumn": "NOW Column 1"
        }
```

```
}
```

四半期

日付を表す文字列から、日付ベースの四半期を含む新しい列を作成します。

> **Note**
>
> 四半期は、新しい列で 1、2、3、または 4 として指定されます。
>
> - 1 は 1 月、2 月、3 月です。
> - 2 は 4 月、5 月、6 月です。
> - 3 は 7 月、8 月、9 月です。
> - 4 は 10 月、11 月、12 月です。

パラメータ

- `sourceColumn` - 既存の列の名前。ソース列のタイプは `string`、`date`、または `timestamp` です。
- `value` – 評価する文字列。
- `targetColumn` – 新しく作成された列の名前。

> **Note**
>
> `sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "QUARTER",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_QUARTER"
        }
```

```
    }
}
```

SECOND

日付を表す文字列から、2 番目の値を含む新しい列を作成します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – 評価する文字列。
- `targetColumn` – 新しく作成された列の名前。

> **Note**
>
> `sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "SECOND",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_SECOND"
        }
    }
}
```

TIME

指定されたソース列または値から、時間値を含む新しい列を作成します。

パラメータ

- `dateTimeFormat` - オプション。新しい列に表示される日付の形式文字列。この文字列が指定されていない場合、デフォルトの形式は ですyyyy-mm-dd HH:MM:SS。

- dateTimeParameters – 日付と時刻のコンポーネントを表す JSON エンコードされた文字列。
  - year
  - value
  - month
  - day
  - hour
  - second

  各コンポーネントは、次のいずれかを指定する必要があります。
  - sourceColumn - 既存の列の名前。
  - value – 評価する文字列。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "TIME",
        "Parameters": {
            "dateTimeFormat": "HH:MM:SS",
            "dateTimeParameters": "{\"year\":{},\"month\":{},\"day\":{},\"hour\":
{\"sourceColumn\":\"rand_hour\"},\"minute\":{\"sourceColumn\":\"rand_minute\"},\"second
\":{\"sourceColumn\":\"rand_second\"}}",
            "targetColumn": "TIME Column 1"
        }
    }
}
```

本日

現在の日付を含む新しい列を 形式で作成しますyyyy-mm-dd。

パラメータ

- timeZone – タイムゾーンの名前。タイムゾーンが指定されていない場合、デフォルトは協定世界時 (UTC) です。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "TODAY",
        "Parameters": {
            "timeZone": "US/Pacific",
            "targetColumn": "TODAY Column 1"
        }
    }
}
```

UNIX_TIME

ソース列または入力値に基づいて、1970 年 1 月 1 日以降の秒数であるエポック時間 (Unix 時間) を表す数値を含む新しい列を作成します。タイムゾーンを推測できる場合、出力はそのタイムゾーンにあります。それ以外の場合、出力は協定世界時 (UTC) になります。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – 評価する文字列。
- `targetColumn` – 新しく作成された列の名前。

> **Note**
>
> `sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "UNIX_TIME",
        "Parameters": {
            "sourceColumn": "TIME Column 1",
            "targetColumn": "TIME Column 1_UNIXTIME"
        }
```

```
    }
}
```

UNIX_TIME_FORMAT

ソース列または入力値の Unix 時間を指定された数値日付形式に変換し、結果を新しい列で返します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – Unix エポックタイムスタンプを表す整数。
- `dateTimeFormat` - オプション。新しい列に表示される日付の形式文字列。指定されなかった場合、デフォルト値は `yyyy-mm-dd HH:MM:SS` です。
- `targetColumn` – 新しく作成された列の名前。

> **Note**
>
> `sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "UNIX_TIME_FORMAT",
        "Parameters": {
            "value": "1601936554",
            "dateTimeFormat": "yyyy-mm-dd HH:MM:SS",
            "targetColumn": "UNIXTIMEFORMAT Column 1"
        }
    }
}
```

WEEK_DAY

日付を表す文字列から、曜日を含む新しい列を作成します。

パラメータ

- sourceColumn - 既存の列の名前。
- value – 評価する文字列。
- targetColumn – 新しく作成された列の名前。

> **Note**
>
> sourceColumn または value を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "WEEK_DAY",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_WEEKDAY"
        }
    }
}
```

WEEK_NUMBER

日付を表す文字列から、週数 (1 ~ 52) を含む新しい列を作成します。

パラメータ

- sourceColumn - 既存の列の名前。
- value – 評価する文字列。
- targetColumn – 新しく作成された列の名前。

> **Note**
>
> sourceColumn または value を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "WEEK_NUMBER",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_WEEK_NUMBER"
        }
    }
}
```

YEAR

日付を表す文字列から、年を含む新しい列を作成します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – 評価する文字列。
- `targetColumn` – 新しく作成された列の名前。

> **Note**
>
> `sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "YEAR",
        "Parameters": {
            "value": "2019-06-12",
            "targetColumn": "YEAR Column 1"
        }
    }
}
```

Window 関数

レシピアクションを操作するウィンドウ関数のリファレンストピックを以下に示します。

トピック

FILL

指定されたソース列に基づいて新しい列を返します。ソース列の欠落値または null 値の場合、 は問題のソース値の前後の行のウィンドウから最新の空白以外の値FILLを選択します。選択した値が新しい列に配置されます。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `numRowsBefore` – ウィンドウの開始を表す、現在のソース行の前の行数。
- `numRowsAfter` – ウィンドウの終わりを表す、現在のソース行の後の行数。

- targetColumn – 新しく作成された列の名前。

Example例

```
{
    "Action": {
        "Operation": "FILL",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "last_name",
            "targetColumn": "last_name_FILL"
        }
    }
}
```

NEXT

新しい列を返します。各値は、ソース列の後の n 行の値を表します。

パラメータ

- sourceColumn - 既存の列の名前。
- numRows – ソース列の前の n 行を表す値。たとえば、 numRowsが 3 の場合、 NEXT は 3 番目の次のsourceColumn値を新しいtargetColumn値として使用します。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
    "Action": {
        "Operation": "NEXT",
        "Parameters": {
            "numRows": "1",
            "sourceColumn": "age",
            "targetColumn": "age_NEXT"
        }
    }
```

```
}
```

PREV

新しい列を返します。各値は、ソース列の前の n 行の値を表します。

パラメータ

- sourceColumn - 既存の列の名前。
- numRows – ソース列の前の n 行を表す値。たとえば、 numRowsが 3 の場合、 PREVは 3 番目の以前のsourceColumn値を新しいtargetColumn値として使用します。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
    "Action": {
        "Operation": "PREV",
        "Parameters": {
            "numRows": "1",
            "sourceColumn": "age",
            "targetColumn": "age_PREV"
        }
    }
}
```

ROLLING_AVERAGE

新しい列で、指定した列の現在の行より前の指定した行数から指定した行数までの値のローリング平均を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- numRowsBefore – ウィンドウの開始を表す、現在のソース行の前の行数。
- numRowsAfter – ウィンドウの終わりを表す、現在のソース行の後の行数。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
    "Action": {
        "Operation": "ROLLING_AVERAGE",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_AVERAGE"
        }
    }
}
```

ROLLING_COUNT_A

新しい列で、指定された列の現在の行より前の指定された行数から指定された行数までの NULL 以外の値のローリングカウントを返します。

パラメータ

- `sourceColumn` - 既存の列の名前。
- `numRowsBefore` – ウィンドウの開始を表す、現在のソース行の前の行数。
- `numRowsAfter` – ウィンドウの終わりを表す、現在のソース行の後の行数。
- `targetColumn` – 新しく作成された列の名前。

Example例

```
{
    "Action": {
        "Operation": "ROLLING_COUNT_A",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_COUNT_A"
        }
    }
}
```

ROLLING_KTH_LARGEST

新しい列で、指定した列の現在の行より前の指定した行数から指定した行数までの k 番目の最大値を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- numRowsBefore – ウィンドウの開始を表す、現在のソース行の前の行数。
- numRowsAfter – ウィンドウの終わりを表す、現在のソース行の後の行数。
- value – k の値。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
  "Action": {
    "Operation": "ROLLING_KTH_LARGEST",
    "Parameters": {
      "sourceColumn": "weight_kg",
      "numRowsBefore": "5",
      "numRowsAfter": "5",
      "value": "3"
      "targetColumn": "weight_kg_ROLLING_KTH_LARGEST"
    }
  }
}
```

ROLLING_KTH_LARGEST_UNIQUE

新しい列で、指定された列の現在の行より前の指定された行数から指定された行数までのローリング一意の k 番目の最大値を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- numRowsBefore – ウィンドウの開始を表す、現在のソース行の前の行数。
- numRowsAfter – ウィンドウの終わりを表す、現在のソース行の後の行数。

- value – k の値。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
  "Action": {
    "Operation": "ROLLING_KTH_LARGEST_UNIQUE",
    "Parameters": {
      "sourceColumn": "games_played",
      "numRowsBefore": "3",
      "numRowsAfter": "3",
      "value": "5",
      "targetColumn": "weight_kg_ROLLING_KTH_LARGEST_UNIQUE"
    }
  }
}
```

ROLLING_MAX

新しい列で、指定された列の現在の行より前の指定された行数から指定された行数までの値のローリング最大値を返します。

パラメータ

- sourceColumn - 既存の列の名前。

  numRowsBefore – ウィンドウの開始を表す、現在のソース行の前の行数。
- numRowsAfter – ウィンドウの終わりを表す、現在のソース行の後の行数。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
    "Action": {
        "Operation": "ROLLING_MAX",
        "Parameters": {
```

```
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_MAX"
        }
    }
}
```

ROLLING_MIN

新しい列で、指定された列の現在の行より前の指定された行数から指定された行数までの値のローリング最小値を返します。

パラメータ

- sourceColumn - 既存の列の名前。

  numRowsBefore – ウィンドウの開始を表す、現在のソース行の前の行数。
- numRowsAfter – ウィンドウの終わりを表す、現在のソース行の後の行数。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
    "Action": {
        "Operation": "ROLLING_MIN",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_MIN"
        }
    }
}
```

ROLLING_MODE

新しい列で、指定された列の現在の行より前の指定された行数から指定された行数までのローリングモード (最も一般的な値) を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- numRowsBefore – ウィンドウの開始を表す、現在のソース行の前の行数。
- numRowsAfter – ウィンドウの終わりを表す、現在のソース行の後の行数。
- modeType – ウィンドウに適用するモーダル関数。有効な値は、NONE、MINIMUM、MAXIMUM、および AVERAGE です。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
    "Action": {
        "Operation": "ROLLING_MODE",
        "Parameters": {
            "modeType": "MINIMUM",
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_MODE"
        }
    }
}
```

ROLLING_STANDARD_DEVIATION

新しい列で、指定した列の現在の行より前の指定した行数から指定した行数までの値のローリング標準偏差を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- numRowsBefore – ウィンドウの開始を表す、現在のソース行の前の行数。
- numRowsAfter – ウィンドウの終わりを表す、現在のソース行の後の行数。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
    "Action": {
        "Operation": "ROLLING_STDEV",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_STDEV"
        }
    }
}
```

ROLLING_SUM

新しい列で、指定した列の現在の行より前の指定した行数から指定した行数までの値のローリング合計を返します。

パラメータ

- sourceColumn - 既存の列の名前。

  numRowsBefore – ウィンドウの開始を表す、現在のソース行の前の行数。
- numRowsAfter – ウィンドウの終わりを表す、現在のソース行の後の行数。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
    "Action": {
        "Operation": "ROLLING_SUM",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_SUM"
        }
    }
}
```

ROLLING_VARIANCE

新しい列で、指定した列の現在の行より前の指定した行数から指定した行数までの値のローリング分散を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- numRowsBefore – ウィンドウの開始を表す、現在のソース行の前の行数。
- numRowsAfter – ウィンドウの終わりを表す、現在のソース行の後の行数。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
    "Action": {
        "Operation": "ROLLING_VAR",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_VAR"
        }
    }
}
```

ROW_NUMBER

「group by」および「order by」ステートメントの列名によって作成されたウィンドウに基づいて、新しい列にセッション識別子を返します。

パラメータ

- groupByColumns – 「group by」列を記述する JSON エンコードされた文字列。
- orderByColumns – 「order by」列を記述する JSON エンコードされた文字列。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
    "Action": {
        "Operation": "ROW_NUMBER",
        "Parameters": {
            "groupByColumns": "[\"is public domain\"]",
            "orderByColumns": "[\"dimensions\"]",
            "targetColumn": "Row number"
        }
    }
}
```

SESSION

「group by」および「order by」ステートメントの列名によって作成されたウィンドウに基づいて、新しい列にセッション識別子を返します。

パラメータ

- sourceColumn - 既存の列の名前。
- units – セッションの長さを記述するための測定単位。有効な値は、MONTHS、YEARS、、MILLISECONDS、QUARTERS、HOURS、MICROSECONDS、、DAYS、WEEKSSECよび ですMINUTES。
- value – 期間unitsを定義する の数。
- groupByColumns –「group by」列を記述する JSON エンコードされた文字列。
- orderByColumns –「order by」列を記述する JSON エンコードされた文字列。
- targetColumn – 新しく作成された列の名前。

Example例

```
{
    "Action": {
        "Operation": "SESSION",
        "Parameters": {
            "sourceColumn": "object number",
            "units": "MINUTES",
```

```
            "value": "10",
            "groupByColumns": "[\"is public domain\"]",
            "orderByColumns": "[\"dimensions\"]",
            "targetColumn": "object number_SESSION",
        }
    }
}
```

ウェブ関数

レシピアクションを操作するウェブ関数のリファレンストピックを以下に示します。

トピック

- IP_TO_INT
- INT_TO_IP
- URL_PARAMS

IP_TO_INT

ソース列の Internet Protocol バージョン 4 (IPv4) 値または他の値をターゲット列の対応する整数値に変換し、結果を新しい列に返します。この関数は IPv4 でのみ機能します。

例えば、次の IP アドレスを考えてみましょう。

```
192.168.1.1
```

この値を への入力として使用する場合IP_TO_INT、出力値は次のとおりです。

```
3232235777
```

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – 評価する文字列。
- `targetColumn` – 作成される新しい列の名前。

`sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "IP_TO_INT",
        "Parameters": {
            "sourceColumn": "my_ip_address",
            "targetColumn": "IP_TO_INT Column 1"
        }
    }
}
```

INT_TO_IP

ソース列の整数値または他の値を、次にターゲット列の対応する IPv4 値に変換し、結果を新しい列で返します。この関数は IPv4 でのみ機能します。

たとえば、次の整数を考えてみます。

```
167772410
```

この値を への入力として使用する場合INT_TO_IP、出力値は次のとおりです。

```
10.0.0.250
```

パラメータ

- sourceColumn - 既存の列の名前。
- value – 評価する文字列。
- targetColumn – 作成される新しい列の名前。

sourceColumn または value を指定できます。両方を指定することはできません。

Example例

```
[  {
    "RecipeAction": {
      "Operation": "INT_TO_IP",
```

```
      "Parameters": {
        "sourceColumn": "my_integer",
        "targetColumn": "INT_TO_IP Column 1"
      }
    }
  }
]
```

URL_PARAMS

URL 文字列からクエリパラメータを抽出し、JSON オブジェクトとしてフォーマットして、結果を新しい列に返します。

たとえば、次の URL を考えてみましょう。

```
https://example.com/?firstParam=answer&secondParam=42
```

この値を への入力として使用する場合URL_PARAMS、出力値は次のとおりです。

```
{"firstParam": ["answer"], "secondParam": ["42"]}
```

パラメータ

- `sourceColumn` - 既存の列の名前。
- `value` – 評価する文字列。
- `targetColumn` – 作成される新しい列の名前。

`sourceColumn` または `value` を指定できます。両方を指定することはできません。

Example例

```
{
    "RecipeAction": {
        "Operation": "URL_PARAMS",
        "Parameters": {
            "sourceColumn": "my_url",
            "targetColumn": "URL_PARAMS Column 1"
        }
    }
```

```
}
```

その他の関数

レシピアクションを操作する他の関数のリファレンストピックを以下に示します。

トピック

- COALESCE
- GET_ACTION_RESULT
- GET_STEP_DATAFRAME

COALESCE

列の配列で見つかった最初の NULL 以外の値を新しい列に返します。関数にリストされている列の順序によって、検索される順序が決まります。

パラメータ

- `sourceColumns` – 既存の列のリストを表す JSON エンコードされた文字列。
- `targetColumn` – 作成される新しい列の名前。

Example例

```
{
    "RecipeAction": {
        "Operation": "COALESCE",
        "Parameters": {
            "sourceColumns": "[\"nation_position\",\"joined\"]",
            "targetColumn": "COALESCE Column 1"
        }
    }
}
```

GET_ACTION_RESULT

以前に送信されたアクションの結果を取得します。インタラクティブエクスペリエンスでのみ使用します。

パラメータ

- `actionId` – 元の SendProjectSessionAction レスポンスで返された ActionId。

Example例

```
{
    "RecipeAction": {
        "Operation": "GET_ACTION_RESULT",
        "Parameters": {
            "actionId": "7",
        }
    }
}
```

GET_STEP_DATAFRAME

プロジェクトのレシピのステップからデータフレームを取得します。インタラクティブエクスペリエンスでのみ使用します。ViewFrame パラメータを使用して、大きなデータフレームをページ分割します。

パラメータ

- `stepIndex` – データフレームを取得するプロジェクトのレシピのステップのインデックス。

Example例

```
{
    "RecipeAction": {
        "Operation": "GET_STEP_DATAFRAME",
        "Parameters": {
            "stepIndex": "0"
        }
    }
}
```

のクォータAWS Glue DataBrew

DataBrew サービスクォータは、AWS Service Quotas コンソールで表示できます。調整可能なクォータについては、クォータの引き上げをリクエストすることもできます。

AWS Glue DataBrewデベロッパーガイドのドキュメント履歴

現在の API バージョン: databrew-2017-07-25

次の表に、このリリースのドキュメントを示しますAWS Glue DataBrew。AWS Glue DataBrewデベロッパーガイドの更新時に通知を受け取る場合は、RSS フィードにサブスクライブできます。

変更	説明	日付
glue:GetCustomEntityType がAWSマネージドポリシーに追加されました	このアクセス許可は、PII ID を有効にしてAWS Glue DataBrewプロファイルジョブを実行するために必要です。詳細については、「AWS Glue DataBrewAWS管理ポリシーの更新」を参照してください。	2024 年 3 月 20 日
CRYPTOGRAPHIC_HASH 変換での複数のハッシュアルゴリズムのサポート	列内の値をハッシュするときにハッシュアルゴリズムを指定できるようになりました。詳細については、「CRYPTOGRAPHIC_HASH」を参照してください。	2023 年 8 月 11 日
glue:BatchGetCustomEntityTypes がAWSマネージドポリシーに追加されました	このアクセス許可は、PII ID を有効にしてAWS Glue DataBrewプロファイルジョブを実行するために必要です。詳細については、「AWS Glue DataBrewAWS管理ポリシーの更新」を参照してください。	2022 年 5 月 9 日

Apache ORC ファイル形式のサポート	DataBrew は、DataBrew データソースと出力のファイル形式として Apache ORC をサポートするようになりました。詳細については、「データソースでサポートされているファイルタイプ」を参照してください。	2022 年 3 月 31 日
クロスアカウントAWS Glue Data Catalog Amazon S3 アクセスのサポート	AWS Glueコンソールで適切なリソースポリシーが作成されAWS アカウントている場合、他の からAWS Glue Data Catalog S3 テーブルにアクセスできるようになりました。ポリシーを作成した後、DataBrew データセットを作成するときに、関連する Data Catalog S3 テーブルを入力ソースとして選択できます。詳細については、「データソースと出力でサポートされている接続」を参照してください。	2022 年 3 月 11 日

Amazon AppFlow とのネイティブコンソール統合のサポート	DataBrew に Amazon AppFlow とのネイティブコンソール統合が追加されました。この統合により、Salesforce、Zendesk、Slack、ServiceNow、およびその他の software-as-a-service (SaaS) アプリケーションのデータに接続できます。Amazon S3 や Amazon Redshift AWS のサービスなどの のデータに接続することもできます。詳細については、「データソースと出力でサポートされている接続」を参照してください。	2021 年 11 月 18 日
データ品質ルールのサポート	DataBrew は、特定のデータのビジネス要件を定義するカスタマイズ可能な検証チェックであるデータ品質ルールの作成をサポートするようになりました。詳細については、「 でのデータ品質の検証 AWS Glue DataBrew」を参照してください。	2021 年 11 月 18 日

カスタム SQL ステートメントのサポート	DataBrew は、Amazon Redshift と Snowflake からデータを取得するためのカスタム SQL ステートメントをサポートするようになりました。このサポートにより、専用のクエリを使用して、大きなテーブルから返されるデータを選択および制限できます。詳細については、「データソースと出力でサポートされている接続」を参照してください。	2021 年 11 月 18 日
PII 検出のサポート	DataBrew は、個人を特定できる情報 (PII) の検出をサポートするようになりました。これにより、データの準備中に PII をマスキングできます。詳細については、「個人を特定できる情報 (PII) の特定と処理」を参照してください。	2021 年 11 月 18 日
追加のAWSリージョンのサポート	DataBrew は追加のAWSリージョンをサポートするようになりました。サポートされているリージョンのリストについては、AWS Glue DataBrew「 エンドポイントとクォータ」を参照してください。	2021 年 10 月 5 日

Lake Formation ベースの Amazon S3 テーブルへのデータの書き込みのサポート	DataBrew は、 に基づくAWS Glue Data Catalog S3 テーブルへのデータの書き込みをサポートするようになりましたAWS Lake Formation 。DataBrew は Tableau Hyper 形式へのデータの書き込みもサポートするようになりました。詳細については、AWS Glue DataBrew「レシピジョブの作成と操作」を参照してください。	2021 年 8 月 13 日
JDBC 送信先にデータを書き込むためのサポート	DataBrew は、JDBC がサポートするデータベースとデータウェアハウスへのデータの直接書き込みをサポートするようになりました。これには、Amazon Redshift、Snowflake、Microsoft SQL Server、MySQL、Oracle Database、PostgreSQL が含まれます。詳細については、AWS Glue DataBrew「レシピジョブの作成と操作」を参照してください。	2021 年 7 月 23 日
プロファイルジョブに対して生成されるデータ品質統計を指定するためのサポート	DataBrew は、プロファイルジョブのデータセットに対して自動生成されるデータ品質統計の指定をサポートするようになりました。詳細については、AWS Glue DataBrew「レシピジョブの作成と操作」を参照してください。	2021 年 7 月 23 日

へのデータセットの書き込みのサポートAWS Glue Data Catalog	DataBrew には、データセットを に直接書き込むためのサポートが追加されましたAWS Glue Data Catalog。データ準備レシピを実行するジョブから作成されたデータセットを、データカタログの Amazon S3、Amazon Redshift、Amazon RDS テーブルに保存することを選択できます。サポートされる RDS テーブルには、Amazon Aurora、RDS for Oracle、RDS for Microsoft SQL Server、RDS for MySQL、RDS for PostgreSQL のテーブルが含まれます。	2021 年 6 月 30 日
高度なデータ型を識別するためのサポート	DataBrew には、列の高度なデータ型を自動的に識別してマークするサポートが追加されました。これにより、特定のタイプのデータを含む列の正規化が容易になります。これらのタイプのデータには、社会保障番号、E メールアドレス、電話番号、性別、クレジットカード、URL、IP アドレス、日時、通貨、郵便番号、国、地域、州、都市が含まれます。	2021 年 6 月 30 日

Amazon AppFlow を使用して SAAS アプリケーションからデータを転送するためのサポート	DataBrew は、Amazon AppFlow を使用して、Salesforce、Zendesk、Slack、ServiceNow などのサードパーティーsoftware-as-a-service (SaaS) アプリケーションから Amazon S3 にデータを転送できるようになりました。詳細については、「データソースと出力でサポートされている接続」を参照してください。	2021 年 4 月 29 日
JDBC データベースからの入力による DataBrew データセットの作成のサポート	DataBrew は、Amazon Redshift、Snowflake、Microsoft SQL Server、MySQL、Oracle Database、PostgreSQL など、JDBC がサポートするデータベースとデータウェアハウスのデータからのデータセットの作成をサポートするようになりました。詳細については、「データソースと出力でサポートされている接続」を参照してください。	2021 年 4 月 2 日
追加の のサポートAWS リージョン	DataBrew は追加の をサポートするようになりましたAWS リージョン。サポートされているリージョンのリストについては、AWS Glue DataBrew「 エンドポイントとクォータ」を参照してください。	2021 年 1 月 28 日

重複を処理するための新しい変換	重複を処理するための 4 つの新しい変換が DataBrew コンソールと API に追加されました。詳細については、「データ品質レシピステップ」の「DELETE_DUPLICATE_ROWS」、「FLAG_DUPLICATE_ROWS」、「FLAG_DUPLICATES_IN_COLUMN」、および「REMOVE_DUPLICATES」を参照してください。	2021 年 1 月 28 日
追加の CSV 区切り文字	DataBrew は、DataBrew データセットの作成に使用されるカンマ区切り値 (CSV) ファイル内のカンマ以外の追加の区切り文字をサポートするようになりました。詳細については、AWS Glue DataBrew「データセットの作成と使用」を参照してください。	2021 年 1 月 28 日
JupyterLab の DataBrew 拡張機能	JupyterLab の拡張機能AWS Glue DataBrewとして を使用できるようになりました。詳細については、JupyterLab の「拡張機能としての DataBrew の使用 JupyterLab」を参照してください。	2020 年 11 月 20 日
新しいデータ準備ツール:AWS Glue DataBrew	これは「AWS Glue DataBrew デベロッパーガイド」の初回リリースです。	2020 年 11 月 11 日

AWS用語集

最新のAWS用語については、 AWS の用語集リファレンスのAWS用語集を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。