

# Amazon QuickSight RoadShow Deep Dive③

データセットパラメータでクエリーを最適化し、CloudWatch で測定

坂下 和香奈

QuickSight Solutions Architect  
アマゾンウェブサービスジャパン合同会社

# Agenda

- パラメータとは
- データセットパラメータ機能
- ユースケース
- デモ
  - 1) データセットパラメータを使用してデータセット・分析を作成
  - 2) CloudWatchによるダッシュボードロード時間の確認
- 制限事項

# パラメータとは

- アクションまたはオブジェクトにより使用される値を転送できる名前付きの変数
- インタクティブなダッシュボードの作成を可能に
- パラメータを使用するには、以下で参照設定可
  - コントロール、フィルタ、アクション
  - 計算フィールド、ナラティブ、タイトル/サブタイトル内
- ダッシュボード間の連携にも利用可

パラメータが追加されました

パラメータを接続:

パラメータ、新しいコントロール、フィルタの組み合わせを使用してフィルタを作成します。

フィルタまたは計算済みフィールドに向けた新しいコントロールを作成します。

計算済みフィールドでパラメータを使用します。

パラメータで URL アクションを作成します

閉じる

フィルター

コントロール

計算フィールド

カスタムアクション

オーダー日付

次の間 - StartDate および EndDate

フィルタータイプ

日付と時刻の範囲

条件

次の間

パラメータを使用

開始日のパラメータ

StartDate

開始日を含める

終了日のパラメータ

EndDate

計算フィールドを追加

名前を追加

フィールド

パラメータ

パラメータを検索

EndDate

GrowthThreshold

MarginThreshold

Period

Region

SalesThreshold

StartDate

# データセットパラメータ機能とは

ダイレクトクエリーで大規模なデータを扱う場合、動的にスライスされたデータのみを取得するようにクエリーを最適化し、**データソース側のスキャン量を制御**したい

- ➡ 特に多くのテーブルをJoinしたクエリー等において、クエリーを最適化でき、フィルター制御のパフォーマンス向上が期待できる
- ➡ データセットの所有者がデータセット内にパラメータを作成。それを分析のフィルターコントロールで利用できる

# データセットパラメータ使用方法

データセットレベルで、データセット所有者によって、データ準備画面で作成

## パラメータの使用場所：

1) ダイレクトクエリーモードでの**カスタムSQL**内

**単一値パラメータ** : SELECT \* FROM all\_flights  
WHERE origin\_state = <<**\$\$State**>>

**複数值パラメータ** : SELECT \* FROM all\_flights  
WHERE origin\_state in (<<**\$\$States**>>)

2) ダイレクトクエリー / SPICEデータセットの

**計算フィールド**内

The screenshot shows the QuickSight interface with a custom SQL query. The query is as follows:

```
1 SELECT t0.*
2 FROM (SELECT carrier_name, RANK() OVER (ORDER BY SUM(distance) DESC) AS rank_num
3 FROM "AwsDataCatalog"."qshandson"."flightdata2"
4 GROUP BY "carrier_name") AS t
5 INNER JOIN (
6 SELECT *
7 FROM "AwsDataCatalog"."qshandson"."flightdata2"
8 ) AS t0 ON t.carrier_name = t0.carrier_name
9 where t.rank_num <<$top>>
```

The results table below the query shows the following data:

| flight_date   | flight_num... | origin_city     | origin_state | destinatio...   | destinatio... | departure... | arrival_delay | cancelled | diverted | air_tim |
|---------------|---------------|-----------------|--------------|-----------------|---------------|--------------|---------------|-----------|----------|---------|
| 2013-03-31... | 1239          | Houston, TX     | Texas        | Los Angeles,... | California    | 5.0          | -18.0         | 0.0       | 0.0      | 188.0   |
| 2013-03-31... | 1281          | Los Angeles,... | California   | Houston, TX     | Texas         | 3.0          | -19.0         | 0.0       | 0.0      | 159.0   |
| 2013-03-31... | 1287          | Chicago, IL     | Illinois     | Newark, NJ      | New Jersey    | 1.0          | -8.0          | 0.0       | 0.0      | 99.0    |
| 2013-03-31... | 745           | Denver, CO      | Colorado     | Portland, OR    | Oregon        | 9.0          | -28.0         | 0.0       | 0.0      | 126.0   |
| 2013-03-31... | 1697          | Las Vegas, NV   | Nevada       | San Francisc... | California    | 97.0         | 108.0         | 0.0       | 0.0      | 68.0    |

分析のパラメータ値をデータセットのパラメータにマッピングすることにより、可視化時にクエリー内のパラメータを置き換えることができる。

# データセットパラメータ使用例①

## <ユーザ選択によるベーシックフィルター>

```
SELECT *  
FROM transactions  
WHERE region = <<$RegionName>>
```

データセット所有者は、Where句をカスタムSQLに加える。それにより、Readerが選択した地域（リージョン）によりデータはフィルターされ、スキャンデータ量は大きく軽減される

パラメータは単一文字列として作成されているが、複数值も対応可。

上記のようなシンプルなカスタムSQLをダイレクトクエリー実施する場合は、カスタムSQL+データセットパラメータではなく、テーブル指定のデータセットを作成し、分析上で地域（リージョン）にてフィルターした方が望ましい（=QuickSightが同じクエリーを内部で生成しており、効果はあまり期待できない<詳細は後述>）

# データセットパラメータ使用例②

## < case when または ifelse で使用 >

```
SELECT Region, Country, date, sum(sales)
FROM transactions
WHERE region=
  (Case
  WHEN <<$UserFIRSTNAME>> In
    (select firstname from user where region='region1')
    and <<$UserLASTNAME>> In
    (select lastname from user where region='region1')
  THEN 'region1'
  WHEN <<$UserFIRSTNAME>> In
    (select firstname from user where region='region2')
    and <<$UserLASTNAME>> In
    (select lastname from user where region='region2')
  THEN 'region2'
  ELSE 'region3'
  END)
```

データセット所有者は、2つのパラメータ（名前と苗字）を作成。Readerがダッシュボード上で選択した2つの文字列が、casewhen句で2回参照される

# データセットパラメータ使用例③

## < Select句で新しい列を作成 >

```
SELECT Region, Country, date,  
       (case  
        WHEN <<$RegionName>>='EU'  
        THEN sum(sales) * 0.93      --convert US dollar to euro  
        WHEN <<$RegionName>>='CAN'  
        THEN sum(sales) * 0.78     --convert US dollar to Canadian Dollar  
        ELSE sum(sales) -- US dollar  
        END  
       ) as "Sales"  
FROM transactions  
WHERE region = <<$RegionName>>
```

ダッシュボード上で選択された地域（リージョン）名に従い、Salesという列が新たに作成される。  
地域により、CASE WHEN句を使用して現地の通貨で売上を再計算



# ユースケース

## 1) ダイレクトクエリーのカスタムクエリーを最適化

クエリー作成の自由度が上がるため、データソース側の機能を引き出しやすくなり、より柔軟な処理の実現や、パフォーマンス改善につながります。

## 2) 分析で利用するデータセットを汎用化

分析で利用しているパラメータを使用した計算フィールドを、データセットレベルに移すことができようになり、データセットを汎用化することができ、データセットの管理も軽減されます。

## 3) パラメータの繰り返し利用によるデータセットのメンテナンスの簡素化

パラメータをカスタムクエリーだけでなく、計算フィールドでも利用することにより、メンテナンスを簡素化することができます。

# デモ

# デモ

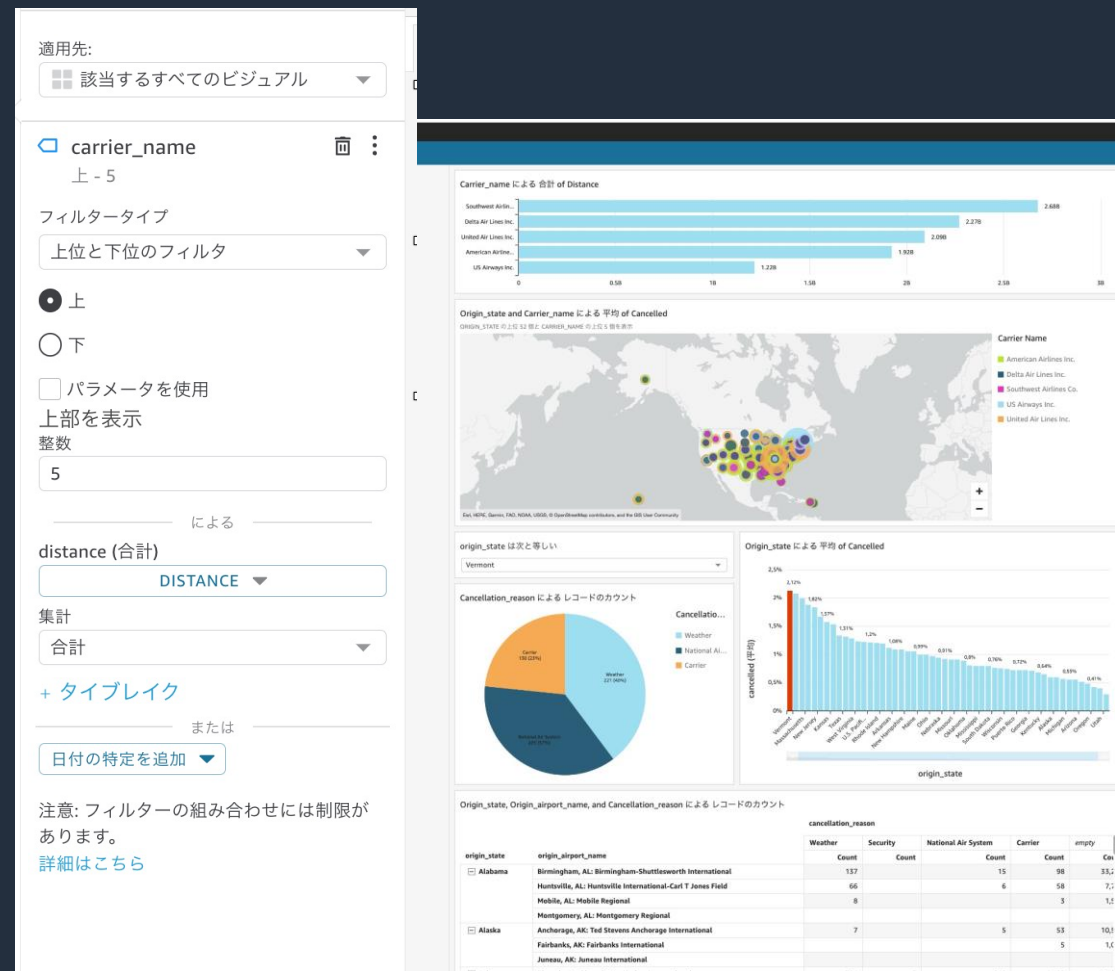
- 米国交通統計局 ( <https://www.bts.gov/> ) が公開している米国国内線の発着データ(2010-2014)、約2,000万件
- S3 via Athena

## シナリオ:

- 航続距離上位5位の航空会社でフィルター
- ダッシュボードを作成済



上位数をパラメータ定義し、カスタムSQLにて上位フィルターしたデータセットを作成



# デモ① データセットパラメータを使用した分析を作成

1. 上位数のデータセットパラメータ（整数値）を作成（\$Top）
2. データセットパラメータを使用したカスタムSQL作成

```
SELECT t0.*
FROM (
  SELECT carrier_name, RANK() OVER (ORDER BY SUM(distance) DESC) AS rank_num
  FROM "AwsDataCatalog"."default"."flightdata2"
  GROUP BY "carrier_name" ) AS t
INNER JOIN (
  SELECT *
  FROM "AwsDataCatalog"."default"."flightdata2"
) AS t0 ON t.carrier_name = t0.carrier_name
where t.rank_num <= <<$top>>
```

距離数合計で航空会社をランク付け

JOINにて、必要なデータ項目を取得

Where句で、選択された上位ランクのみを取得

# デモ① データセットパラメータを使用して分析を作成

## 3. 分析画面で、作成したデータセットパラメータを、分析のパラメータにマッピング

- データセットパラメータがunmappedパラメータとして表示される
- マッピングする新しいパラメータを作成し、スライダーコントロールを作成

 スライダー上で選択した値が、カスタムSQLに渡される



# データセットパラメータ+カスタムSQLで、航続距離数上位5位の航空会社でフィルターしたダッシュボード

パラメータ

パラメータを検索

# top

カスタム SQL 名

TopN-Query

カスタム SQL

```
1 SELECT t0.*
2 FROM (SELECT carrier_name, RANK() OVER (ORDER BY SUM(distance) DESC) AS rank_num
3 FROM "AwsDataCatalog"."qshandson"."flightdata2"
4 GROUP BY "carrier_name" ) AS t
5 INNER JOIN (
6 SELECT *
7 FROM "AwsDataCatalog"."qshandson"."flightdata2"
8 ) AS t0 ON t.carrier_name = t0.carrier_name
9 where t.rank_num <= <<$top>>
```

データセット

分析

キャンセル状況分析

フィールドウェル Y軸 carrier\_name 値 # distance (合計) グループ/色

シート 1

コントロール

上位航空会社

5

Carrier\_name による 合計 of Distance

| Carrier_name          | 合計 of Distance |
|-----------------------|----------------|
| Southwest Airline...  | 2.68B          |
| Delta Air Lines Inc.  | 2.27B          |
| United Air Lines Inc. | 2.09B          |
| American Airline...   | 1.92B          |
| US Airways Inc.       | 1.22B          |

Origin\_state and Carrier\_name による 平均 of Cancelled

Origin\_state の上位 52 個と CARRIER\_NAME の上位 5 個を表示

Carrier Name

- American Airlines Inc.
- Delta Air Lines Inc.
- Southwest Airlines Co.
- US Airways Inc.
- United Air Lines Inc.

出発地の州選択

Vermont

Origin\_state による 平均 of Cancelled

2.5%



# データソース(Athena) へのクエリを確認

```
    COUNT(*) AS "count", DENSE_RANK() OVER (ORDER BY "origin_state" NULLS FIRST,
    "origin_airport_name" NULLS FIRST) AS "$RANK_1", DENSE_RANK() OVER (ORDER BY
    "cancellation_reason" DESC) AS "$RANK_2"
8 FROM (SELECT t0.*
9 FROM (SELECT carrier_name, RANK() OVER (ORDER BY SUM(distance) DESC) AS rank_num
10 FROM "AwsDataCatalog"."default"."flightdata2"
11 GROUP BY "carrier_name" ) AS t
12 INNER JOIN (
13 SELECT *
14 FROM "AwsDataCatalog"."default"."flightdata2"
15 ) AS t0 ON t.carrier_name = t0.carrier_name
16 where t.rank_num <= 5) AS "TopN-Query"
17 GROUP BY "origin_state", "origin airport_name", "cancellation_reason"
18 ) AS "t"
19 WHERE "$RANK_1" <= 500 AND
```

作成したカスタムクエリー(TopN-Query)が、サブクエリー化され、ビジュアルのクエリーのFROMで利用

上位パラメータの値がデフォルト値に置き換わっている



# データソース(Athena) へのクエリを確認

## 分析フィルターを使用した場合

```
3 SELECT "carrier_name" AS "$f0e87223-03a8-4e64-93c1-ed6191df4011.carrier_name_join_1", RANK() OVER
   (ORDER BY SUM("distance") DESC) AS "$RANK_3"
4 FROM "AwsDataCatalog"."default"."flightdata2"
5 GROUP BY "carrier_name"
6 ) AS "t"
7 INNER JOIN (
8 SELECT "origin_state" AS "f0e87223-03a8-4e64-93c1-ed6191df4011.origin_state", "cancelled" AS "f0e87223
   -03a8-4e64-93c1-ed6191df4011.cancelled", "carrier_name" AS "f0e87223-03a8-4e64-93c1-ed6191df4011
   .carrier_name"
9 FROM "AwsDataCatalog"."default"."flightdata2"
10 ) AS "t0" ON "t"."$f0e87223-03a8-4e64-93c1-ed6191df4011.carrier_name_join_1" = "t0"."f0e87223-03a8
   -4e64-93c1-ed6191df4011.carrier_name" OR "t"."$f0e87223-03a8-4e64-93c1-ed6191df4011
   .carrier name join 1" IS NULL AND "t0"."f0e87223-03a8-4e64-93c1-ed6191df4011.carrier_name" IS NULL
11 WHERE "t"."$RANK_3" <= 5
12 GROUP BY "t0"."f0e87223-03a8-4e64-93c1-ed6191df4011.origin_state", "t0"."f0e87223-03a8-4e64-93c1
```

サブクエリー化されていないが、基本同じクエリーが生成されている

## デモ② CloudWatchでダッシュボードのロード時間を確認する

- CloudWatchにて、QuickSightのメトリックス（SPICEへの取り込み状況、ダッシュボードやビジュアルのパフォーマンス）を参照可能
- 特定のダッシュボードやビジュアル単位にメトリックスを指定し、CloudWatch上でアラーム設定をし監視することができる。
  1. 公開した2種類のダッシュボードから、**ダッシュボードID**を取得
  2. CloudWatch上で、取得したダッシュボードIDの**LoadDashboardTime**のメトリックスを選択し、グラフを追加



# データセットパラメータの制限事項

- SPICEデータセットでは利用できない
- 動的デフォルト値設定は、分析のパラメータでのみ利用可能
- データセットパラメータをマッピングした分析のパラメータで 複数値選択のコントロールを使用可能だが、「すべて選択」オプションは利用できない
- データセットパラメータを使用したカスタマイズコントロールはサポート外
- データセットパラメータは分析のフィルターでは使用できず、データセット上のフィルターで使用可（ダイレクトクエリーのみ）
- Eメールでの定期配信をスケジュールするとき、コントロールの選択値がデータセットパラメータの値（フィルター値）としてレポートには反映されない。パラメータのデフォルト値がレポートでは使用される

# Call to Action

<https://catalog.us-east-1.prod.workshops.aws/workshops/aa601d0b-84c9-4f77-b9a7-5954d8574cd5/ja-JP/7-connecting-to-data-via-direct-query/use-dataset-paramter-feature>

The screenshot shows the AWS Workshop Studio interface. On the left is a navigation sidebar with a tree view. The main content area displays a document page. The document title is 'データセットパラメータを使用してクエリーを最適化する'. The sidebar has a yellow highlight around the 'データセットパラメータを使用してクエリーを最適化する' item. The document content includes an introduction, a list of use cases, and a note about the workshop's focus.

aws workshop studio

Amazon QuickSight - Visualization Basics (Japanese) > データソースに接続しダッシュボードを作成する > データセットパラメータを使用してクエリーを最適化する

## データセットパラメータを使用してクエリーを最適化する

ここでは、データセットパラメータを使用することで、データソースで実行されるクエリーを最適化し、用途やニーズに応じ柔軟に大規模データを取り扱えることを確認します。

パラメータは、「高度なダッシュボードを作成する」でも説明したように、アクションまたはオブジェクトにより使用される値を転送できる名前付きの変数で、インタクティブなダッシュボードの作成を可能にします。

**データセットパラメータ**は、データセットの所有者がデータセットの準備画面で、データセットレベルで作成することができます。作成したパラメータは、以下で使用することができます。

- ダイレクトクエリーのカスタムクエリー内
- ダイレクトクエリー、またはSPICEデータセットの計算フィールド内

また、分析で作成したパラメータを、このデータセットパラメータとマッピングすることにより、ダッシュボードをロードする際に実行されるクエリー内のパラメータを動的に上書きすることができます。

主なユースケースは

- ダイレクトクエリーのカスタムクエリーを最適化  
クエリー作成の自由度が上がるため、データソース側の機能を引き出しやすくなり、より柔軟な処理の実現や、パフォーマンス改善につながります。
- 分析で利用するデータセットを汎用化  
分析で利用しているパラメータを使用した計算フィールドを、データセットレベルに移すことができるようになり、データセットを汎用化することができ、データセットの管理も軽減されます。
- パラメータの繰り返し利用によるデータセットのメンテナンスの簡素化  
パラメータをカスタムクエリーだけでなく、計算フィールドでも利用することにより、メンテナンスを簡素化することができます。

当ハンズオンでは、前々項「大規模データをAthena経由で可視化をする」で作成したダッシュボードの見た目を変更せずに、データセットパラメータを使用して、Athenaへのデータ取得方法を変更していくことで、どのようにデータセットパラメータを利用できるかを体験いただける内容になっています。

© 2008 - 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy policy Terms of use

# Thank you!