01 Dataform - KirkYagami 👨‍💻🕵️

> Read my notes first then you may watch this video to understand all the concepts in a practical manner

1. Introduction

What is Dataform?

Dataform is a data transformation tool designed to help manage and orchestrate SQL workflows in Google Cloud. It simplifies the creation, testing, and management of SQL pipelines for data warehouses like BigQuery. By integrating version control and supporting modern data engineering concepts like dependency management, it ensures efficient collaboration and development of data pipelines.

⚡

2. Key Features and Benefits

Why Dataform?

- **Simplified SQL Workflow Management**: It enables easy SQL-based transformations using declarative configurations and reduces manual steps in data workflows.
- **Version Control Integration**: Seamlessly integrates with GitHub to allow version control and collaboration, ensuring that all team members can contribute and track changes.
- **Modular SQL and Dependencies**: Supports modular workflows by allowing SQL scripts to define dependencies between datasets, ensuring an ordered and efficient transformation pipeline.
- **Collaboration**: Multiple engineers can work on the same project, improving productivity and ensuring that data workflows scale effectively.

Key Concepts

- **Dataform Project**: A collection of SQLX scripts and configurations representing a pipeline of transformations.
- **SQLX Files**: Enhanced SQL files used in Dataform that allow developers to write SQL with additional metadata and JavaScript code.
- **Tables**: Final output datasets that are created or updated via Dataform transformations.
- **Views**: Logical views built from SQL queries to provide real-time access to transformed data.

From SQL to SQLX

- **SQLX** is an extension of standard SQL, allowing you to define transformations in SQL while also embedding JavaScript to enable parameterization and logic. With SQLX, developers can manage large-scale transformations more effectively by combining the flexibility of SQL with scripting capabilities.

Version Control and Collaboration

- **Git-based Collaboration**: Dataform integrates directly with GitHub, enabling engineers to commit and track changes to their transformation scripts.
- **Development Workspaces**: Team members can create isolated workspaces, make changes, and test transformations before merging them into the production pipeline.

Dependency Management

- **Directed Acyclic Graph (DAG)**: Dataform automatically resolves dependencies between tables and views based on SQLX scripts. The execution order is determined by a DAG, ensuring the correct order of operations.
- **Referencing Dependencies**: Developers can define dependencies between SQL scripts, ensuring that one table is only updated after another has been transformed.

JavaScript in Dataform

- JavaScript is used within SQLX files to perform dynamic tasks such as parameterizing SQL queries and defining reusable variables or logic. This adds flexibility to the SQL-based pipelines by embedding business logic or conditions directly in SQL transformations.

3. Workflow Execution Scheduling Options

- **Cloud Scheduler Integration**: You can schedule Dataform workflows using Google Cloud Scheduler, automating the transformation processes to run at specific intervals (e.g., hourly, daily).
- **Manual Execution**: Dataform workflows can also be run manually from the Google Cloud Console or using the CLI, providing flexibility in execution based on operational needs.

4. Creating a Dataform Repository

Creating Necessary GitHub Assets

- Create a GitHub repository for your Dataform project to enable version control. This allows tracking, reviewing, and collaborating on SQL transformation scripts.

Adding Access Token to Secret Manager

- For secure access to GitHub from your Google Cloud environment, store your GitHub access token in **Google Secret Manager**. This token allows Dataform to authenticate with GitHub without exposing credentials in the project code.

Adding IAM Roles to Service Account

- Assign the necessary IAM roles to the Dataform service account to interact with BigQuery and other GCP services (e.g., `BigQuery Data Editor`, `Storage Admin`).

Creating Development Workspace

- Use Dataform to create isolated development environments or workspaces where individual developers can make and test changes without affecting production workflows.

5. Intro to Demo Examples

Overview of Repository

- The repository will contain SQLX scripts that define transformations for various datasets. The directory structure should organize scripts by logical data flow (e.g., raw data, staging, final views).
- Example repository might include folders such as `definitions`, `assertions`, `includes`, etc.

Running First Workflow & Adding IAM Role

- After setting up the repository, run your first Dataform workflow from the console. Ensure that the appropriate IAM roles are in place for the service account to access the required datasets in BigQuery.

6. Dependencies as DAGs

- **Directed Acyclic Graph (DAG)**: Dataform visualizes SQL workflows as DAGs, showing the relationships between different tables and transformations. This makes it easier to understand the execution flow and manage dependencies between datasets.

7. Custom Operations & Tags

Custom Operations

- Custom operations allow you to execute SQL or BigQuery commands outside the standard transformation process. For example, you might perform operations like partitioning a table or creating indexes to optimize queries.

Tags

- Dataform allows tagging datasets or operations to organize the workflow and apply certain logic conditionally, e.g., running specific tagged transformations as part of different environments or stages.

8. Assertions

- **Data Quality Assertions**: Assertions are used to validate data within your pipelines. They help to ensure data integrity by defining constraints or conditions that must be met, such as row count or data format.

Skipping Pipeline Step Execution with Assertions

- You can skip steps in a pipeline based on assertion outcomes. For example, if data fails to meet a defined condition, the transformation can be aborted, or the step can be skipped to prevent loading bad data.

9. Reuse JavaScript Variables in SQLX Files

- Variables defined in JavaScript within SQLX files can be reused across transformations, enabling developers to create dynamic queries. These variables can be anything from table names to conditions in the `WHERE` clause, making the transformations more adaptable to changes.

Implementation

Folder Structure

To implement the Dataform project, the following folder structure is recommended for storing code and configuration files:

```
dataform_project/
│
├── definitions/
│   ├── raw_data.sqlx
│   ├── transformed_data.sqlx
│   └── final_data.sqlx
│
├── assertions/
```

```
    ├── data_quality_assertion.sqlx
    └── row_count_assertion.sqlx

├── includes/
    ├── common_variables.js
    └── reusable_queries.sql

├── .dataform/
    ├── dataform.json
    └── credentials.json

├── scripts/
    └── run_workflow.sh

└── package.json
```

Folder Details

- `definitions/`: This folder contains your core SQLX files that define transformations on raw data and final data outputs.
- `assertions/`: Here, you define SQLX assertions to validate data quality.
- `includes/`: Contains reusable variables and SQL code shared across different SQLX files. For instance, you might include JavaScript variables or shared SQL snippets.
- `.dataform/`: Stores project configuration files such as `dataform.json` and authentication details like `credentials.json`.
- `scripts/`: A folder to store shell scripts to run workflows, automate processes, or perform other administrative tasks.
- `package.json`: Used for project management and dependency tracking in Node.js, which Dataform relies on.

File and Code Details

1. `definitions/raw_data.sqlx`

```
config {
  type: "table",  # Output as a table in BigQuery
  schema: "raw"  # Schema where the table will be stored
}

SELECT
  id,
  name,
  created_at
```

```
FROM
  `project.dataset.source_table`
WHERE
  created_at ⩾ CURRENT_DATE();
```

2. definitions/transformed_data.sqlx

```
config {
  type: "table",
  schema: "staging",
  dependencies: ["raw_data"]  # Depends on the output of `raw_data.sqlx`
}

SELECT
  id,
  UPPER(name) AS name,
  created_at
FROM
  ${ref("raw_data")};
```

3. definitions/final_data.sqlx

```
config {
  type: "table",
  schema: "analytics",
  dependencies: ["transformed_data"]  # Depends on the output of
`transformed_data.sqlx`
}

SELECT
  id,
  name,
  created_at,
  CURRENT_TIMESTAMP() AS processed_at
FROM
  ${ref("transformed_data")};
```

4. assertions/data_quality_assertion.sqlx

```
config {
  type: "assertion",  # Defines this as an assertion
  schema: "assertions",
  dependencies: ["raw_data"]
}
```

```
SELECT
  COUNT(*)
FROM
  ${ref("raw_data")}
WHERE
  name IS NULL;
```

5. assertions/row_count_assertion.sqlx

```
config {
  type: "assertion",
  schema: "assertions",
  dependencies: ["final_data"]
}

SELECT
  COUNT(*)
FROM
  ${ref("final_data")}
WHERE
  created_at > CURRENT_DATE();
```

6. includes/common_variables.js

```
// Define a reusable JavaScript variable for table names
const raw_table = "project.dataset.source_table";
const today = new Date().toISOString().slice(0, 10);

module.exports = { raw_table, today };
```

7. includes/reusable_queries.sql

```
-- Example of a reusable SQL query
SELECT
  id,
  LOWER(name) AS name
FROM
  ${ref("raw_data")};
```

8. .dataform/dataform.json

```
{
  "defaultSchema": "dataform",
  "warehouse": "bigquery",
```

```
  "assertionsEnabled": true,
  "projectId": "your-gcp-project-id",
  "defaultDatabase": "your-bigquery-dataset"
}
```

9. .dataform/credentials.json

This file should contain the credentials for accessing Google Cloud services, such as BigQuery. You can obtain this file by downloading the service account credentials JSON.

```
{
  "type": "service_account",
  "project_id": "your-gcp-project-id",
  "private_key_id": "xxxxxx",
  "private_key": "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----
\n",
  "client_email": "dataform@your-gcp-project-id.iam.gserviceaccount.com",
  "client_id": "xxxxxx",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/dataform%40your-gcp-
project-id.iam.gserviceaccount.com"
}
```

10. scripts/run_workflow.sh

This script runs your Dataform workflow.

```
#!/bin/bash

# Run the Dataform workflow
dataform run --config .dataform/dataform.json
```

11. package.json

This file defines the dependencies needed to run Dataform.

```
{
  "name": "dataform_project",
  "version": "1.0.0",
  "scripts": {
    "run": "bash scripts/run_workflow.sh"
```

```
  },
  "dependencies": {
    "@dataform/cli": "^1.0.0"
  }
}
```

Setup Steps for the Repository

1. **Initialize the Dataform Project**
   - Navigate to your project directory and run:

     ```
     dataform init
     ```

   - This will set up the initial structure of your Dataform project.

2. **Install Dataform CLI**
   - Ensure that you have Node.js installed, and then install the Dataform CLI globally:

     ```
     npm install -g @dataform/cli
     ```

3. **Create Your Dataform Repository**
   - Commit the folder structure and files mentioned above to a GitHub repository.
   - Add `credentials.json` to your `.gitignore` to avoid committing sensitive data.

4. **Configure IAM Roles and Access Tokens**
   - Store your GitHub access token in Google Secret Manager:

     ```
     gcloud secrets create github-token --data-file=path/to/token
     ```

   - Assign the necessary IAM roles to the service account, e.g., `BigQuery Data Editor`, `Storage Admin`.

5. **Run the Workflow**
   - Run the Dataform workflow from your terminal using the command:

     ```
     dataform run
     ```

6. **Schedule Workflow Execution**
   - Use Google Cloud Scheduler to automate the execution of this workflow by creating a scheduled job that triggers `scripts/run_workflow.sh` at desired intervals.

This folder structure and code will allow your associates to build and run a basic Dataform project that includes data transformations, assertions, and reusable components.

Made with ❤ by Nikhil Sharma