

## Problem\_statement:

## "Jaffle Shop," a fictional fast-food chain, wants to improve its business insights by transforming raw data from its operations into a clean, organized, and reliable dataset for analysis.

## Specifically, Jaffle Shop's data team faces the following challenges:

## Data Silos: Customer and order data are stored in separate tables (jaffle\_shop\_customers and jaffle\_shop\_orders), making it difficult to analyze customer order patterns.

## Data Inconsistency: The raw data may contain inconsistencies, such as missing customer information or duplicate order entries, which can lead to inaccurate reporting.=

## Lack of Historical Tracking: Jaffle Shop needs to track how order statuses change over time to understand order fulfillment and identify potential bottlenecks.

## Inefficient Data Transformation: The current process of transforming raw data for analysis is manual and time-consuming, requiring significant SQL coding and maintenance.

## Solution Objectives

## Using dbt, Perform the following:

## Create staging models (stg\_customers, stg\_orders) to extract and clean data from source tables.

## Build a core model (customer\_orders) that joins customer and order data to provide a unified view of customer order information.

## Implement data quality checks (tests) to ensure the accuracy and completeness of the transformed data.

## Use snapshots to track historical changes in order data.

## Develop a reusable macro to improve efficiency.

## Prerequisites

- dbt Cloud Account: Sign up at https://cloud.getdbt.com/signup  
- Databricks Account: Sign up at https://databricks.com/try-databricks  
- Cloud Provider Account: AWS, GCP, or Azure with permissions to create resources

## Step 1: Create a Databricks Workspace

1. **Sign Up**: Use your existing account or sign up for a Databricks account. Complete the form with your user information and click **Continue**.
2. **Select Cloud Provider**: On the next screen, select your cloud provider. This tutorial uses AWS as the cloud provider, but if you use Azure or GCP internally, please select your platform. The setup process will be similar. *Do not select the "Get started with Community Edition" option*, as this will not provide the required compute for this guide.
3. **Email Verification**: Check your email and complete the verification process.
4. **Choose Plan**: After completing the verification processes, you will be brought to the first setup screen. Databricks defaults to the Premium plan, and you can change the trial to Enterprise on this page.
5. **Create Workspace**: Now, it's time to create your first workspace. A Databricks workspace is an environment for accessing all of your Databricks assets. The workspace organizes objects like notebooks, SQL warehouses, clusters, and more into one place. Provide the name of your workspace, choose the appropriate AWS region, and click **Start Quickstart**. You might get the checkbox of *I have data in S3 that I want to query with Databricks*. You do not need to check this off for this tutorial.
6. **AWS Login**: By clicking on **Start Quickstart**, you will be redirected to AWS and asked to log in if you haven’t already. After logging in, you should see a page similar to this.

*Tip*: If you get a session error and don’t get redirected to this page, you can go back to the Databricks UI and create a workspace from the interface. All you have to do is click **Create Workspaces**, choose the quickstart, fill out the form, and click **Start Quickstart**.

1. **Create Stack**: There is no need to change any of the pre-filled out fields in the Parameters. Just add in your Databricks password under Databricks Account Credentials. Check off the Acknowledgement and click **Create stack**.
2. **Workspace Ready**: Go back to the Databricks tab. You should see that your workspace is ready to use.
3. **Access Workspace**: Now let’s jump into the workspace. Click **Open** and log into the workspace using the same login as you used to log into the account.

## Step 2: Load Sample Data

1. **Download Sample Data**: Download these CSV files (the Jaffle Shop sample data) that you will need for this guide:
   * [jaffle\_shop\_customers.csv](https://dbt-tutorial-public.s3-us-west-2.amazonaws.com/jaffle_shop_customers.csv)
   * [jaffle\_shop\_orders.csv](https://dbt-tutorial-public.s3-us-west-2.amazonaws.com/jaffle_shop_orders.csv)
   * [stripe\_payments.csv](https://dbt-tutorial-public.s3-us-west-2.amazonaws.com/stripe_payments.csv)
2. **Access SQL Space**: First, we need a SQL warehouse. Find the drop-down menu and toggle into the SQL space.
3. **Set Up SQL Warehouse**: We will be setting up a SQL warehouse now. Select **SQL Warehouses** from the left-hand side console. You will see that a default SQL Warehouse exists.
4. **Upload Data**: Use the Databricks UI to upload the CSV files into tables named jaffle\_shop\_customers, jaffle\_shop\_orders, and stripe\_payments within the default schema.
5. **Verify Data**: Once that's done, make sure you can query the training data. Navigate to the SQL Editor through the left-hand menu. This will bring you to a query editor.

Run the following queries to ensure the data is loaded:



1. **Grant Permissions**: To ensure any users who might be working on your dbt project have access to your objects, run this command:



## Step 3: Connect dbt Cloud to Databricks

There are two ways to connect dbt Cloud to Databricks:

### Option 1: Partner Connect (Recommended for Quick Setup)

1. **Access Partner Connect**: In the sidebar of your Databricks account, click **Partner Connect**.
2. **Select dbt**: Click the dbt tile.
3. **Configure Connection**:
   * Select a catalog from the drop-down list, and then click **Next**. The drop-down list displays catalogs you have read and write access to. If your workspace isn't <UC>-enabled, the legacy Hive metastore (hive\_metastore) is used.
   * If there are SQL warehouses in your workspace, select a SQL warehouse from the drop-down list. If your SQL warehouse is stopped, click **Start**.
   * If there are no SQL warehouses in your workspace:
     + Click **Create warehouse**. A new tab opens in your browser that displays the New SQL Warehouse page in the Databricks SQL UI.
     + Follow the steps in Create a SQL warehouse in the Databricks docs.
     + Return to the Partner Connect tab in your browser, and then close the dbt tile.
     + Re-open the dbt tile.
     + Select the SQL warehouse you just created from the drop-down list.
   * Select a schema from the drop-down list, and then click **Add**. The drop-down list displays schemas you have read and write access to. You can repeat this step to add multiple schemas.
4. **Connect to dbt Cloud**: Click **Connect to dbt Cloud**. A new tab opens in your web browser, which displays the getdbt.com website.
5. **Complete Setup**: Complete the on-screen instructions on the getdbt.com website to create your trial dbt Cloud account.

### Option 2: Manual Connection

If you prefer to set up the connection manually, refer to the [Databricks setup guide](https://docs.getdbt.com/docs/core/connect-data-platform/databricks-setup) for detailed instructions on configuring your profiles.yml file and connecting dbt Cloud to Databricks.

## Step 4: Set Up Your dbt Project

1. **Create a New Project**: In dbt Cloud, create a new project named jaffle\_shop.
2. **Configure Repository**: You can either set up a dbt Cloud-hosted managed repository or connect to a supported Git provider (e.g., GitHub). For beginners, using a managed repository is recommended.
3. **Set Up Environment**: Configure your deployment environment, ensuring it points to the Databricks connection established earlier.
4. **Initialize Project**: Use the dbt Cloud IDE to initialize your project structure.

## Step 5: Define dbt\_project.yml

Create a dbt\_project.yml file in your project root with the following content:



## Step 6: Create Model Files

### models/staging/stg\_customers.sql



### models/staging/stg\_orders.sql

### models/marts/customer\_orders.sql (Incremental Model)



## Step 7: Define schema.yml

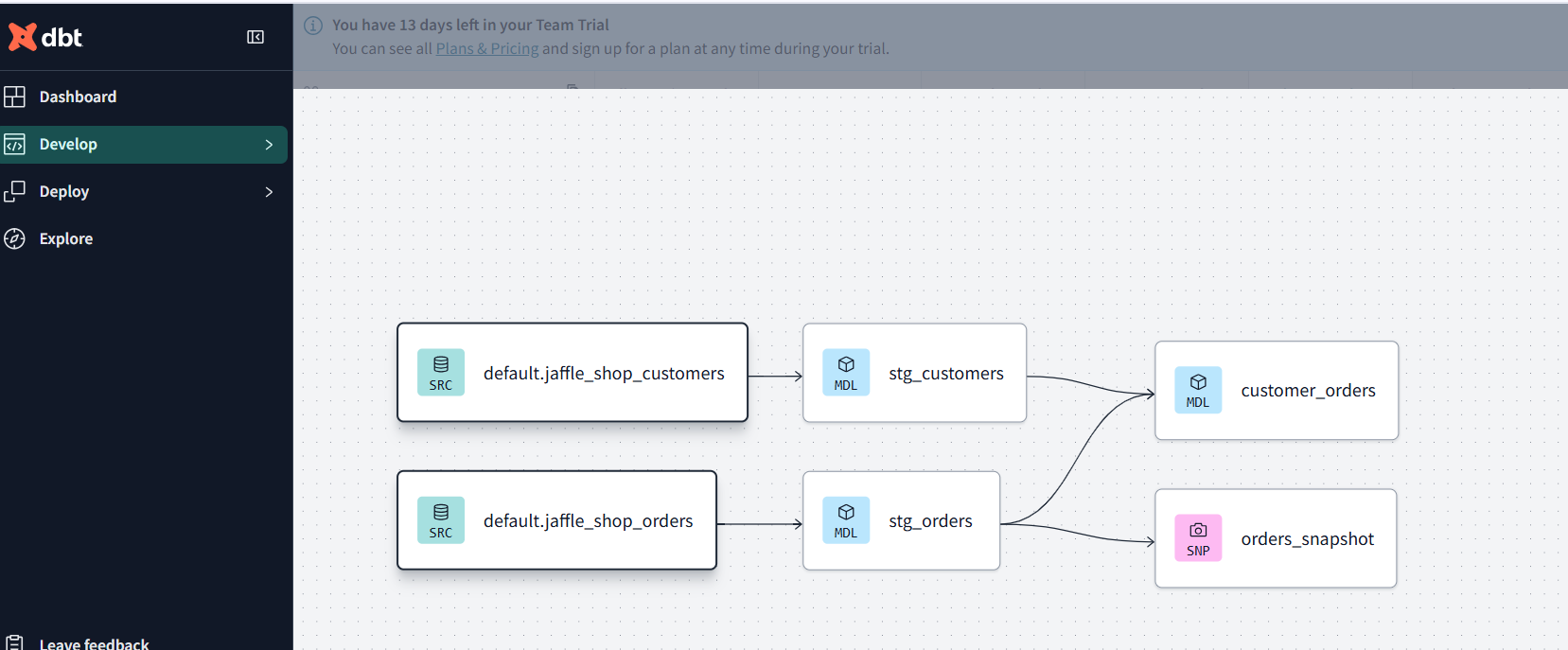
Create a schema.yml file in your models directory:



## Step 8: Create Snapshot

Create a snapshot file at snapshots/orders\_snapshot.sql:

Snapshots let you track changes over time in your source data — this is useful for slowly changing dimensions (SCDs).



## Step 9: Add a Custom Jinja Macro

Create a macro file at macros/common\_macros.sql:

You can use this macro in models or in dbt run-operation commands like:



## Step 10: Add dbt Tests

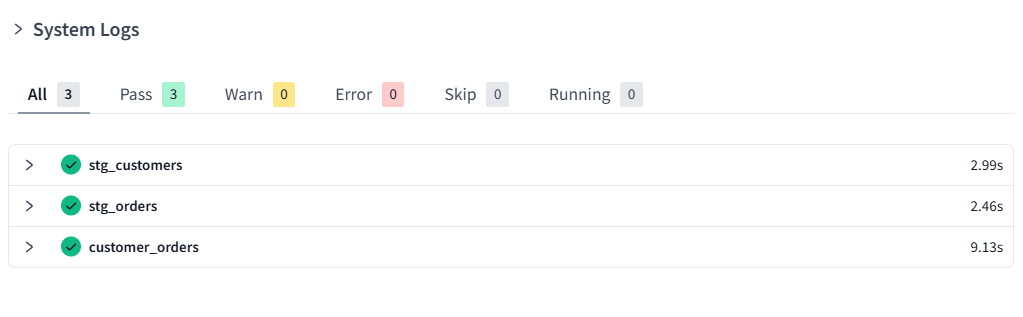
These were already included in the schema.yml, but it’s worth highlighting:

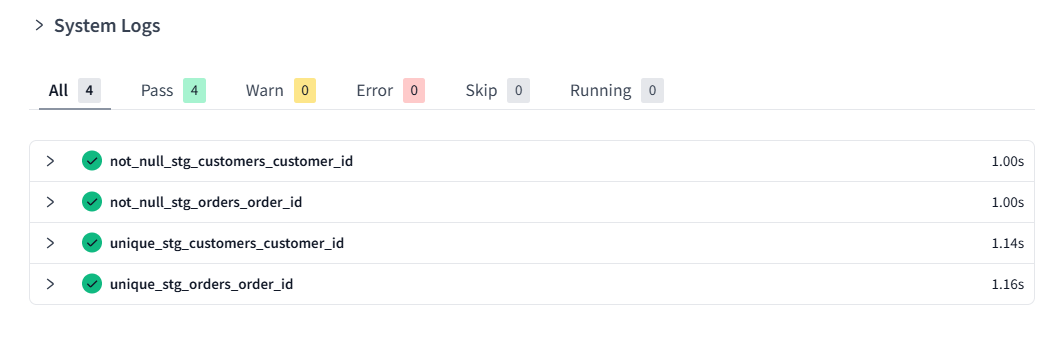


## Step 11: Run Everything in dbt Cloud

### From the dbt Cloud IDE:

Run each of these in order using the sidebar buttons or command bar.





Check the snapshot table in databricks –

GRANT SELECT ON TABLE workspace.snapshots.orders\_snapshot TO `<user>`;

SELECT \* FROM workspace.snapshots.orders\_snapshot;

## Step 12: Explore the Lineage Graph

In dbt Cloud, go to the **"Documentation"** tab and click **"Generate Docs"**.

Explore the graph of models and sources to visually demonstrate dependencies.

## Optional: Semantic Layer (Advanced)

If you're on a dbt Cloud plan that supports semantic models, create a file like semantic\_models/customer\_orders.yaml:



## Bonus: Testing Incremental Models

Update a row in your raw orders table (in Databricks), then re-run dbt run to observe that only new data is processed:



This shows how dbt handles data incrementally for large datasets.