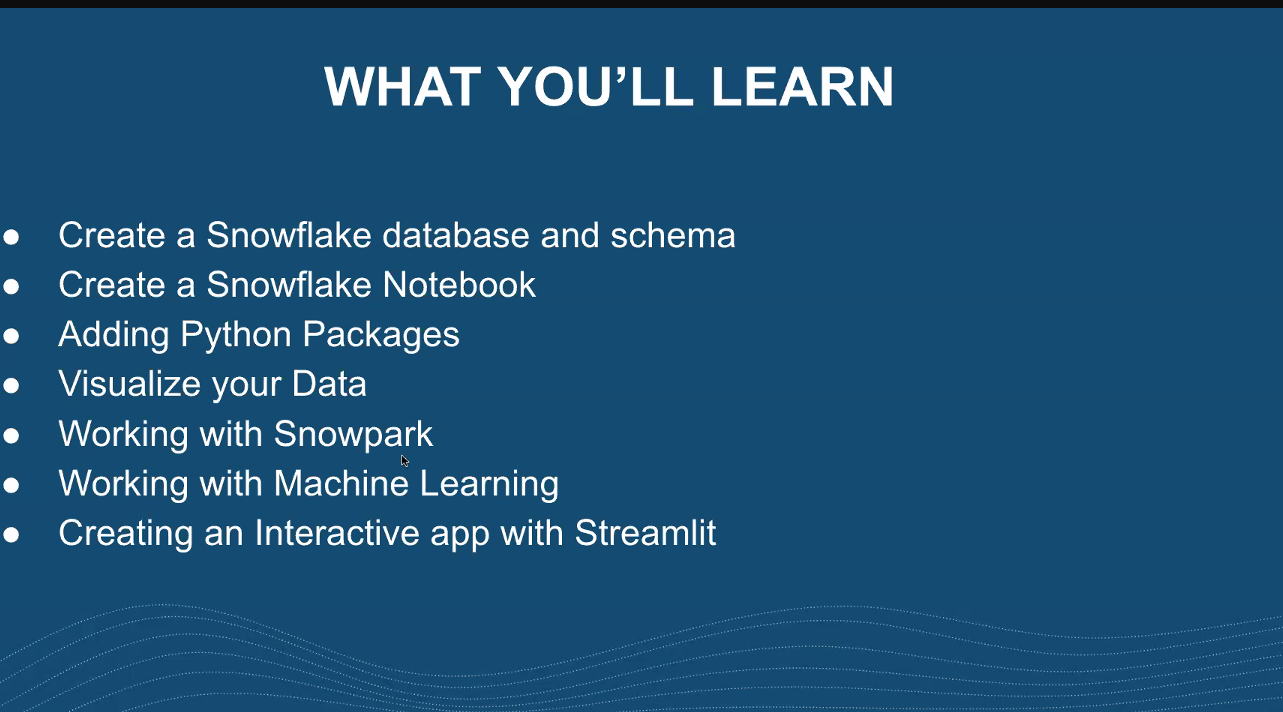
Data pipeline with DAG and ML using Snowpark for Python and Snowflake Notebooks

Project Overview

The project is a hands‐on demonstration of using Snowflake’s Snowpark for Python to handle both data engineering and machine learning tasks. It combines notebook-based workflows with database operations, showcasing how to build data pipelines and ML processes directly within the Snowflake environment.



[Getting Started with Data Engineering and ML using Snowpark for Python and Snowflake Notebooks](https://quickstarts.snowflake.com/guide/getting_started_with_dataengineering_ml_using_snowpark_python/index.html?index=..%2F..index#0)

[Snowflake Labs](https://github.com/Snowflake-Labs)

Snowflake is a casenstitive database and will uppercase everymodel name.

Copied sql raw code from github for data ingestion from aws s3 bucket as shown below -

# I Data Transformations in Snowpark

## 1. Setup Environment and Data Ingestion with SQL queries in a data warehouse.

## Create Tables and Load Data

Log into [Snowsight](https://docs.snowflake.com/en/user-guide/ui-snowsight.html#) using your credentials to create tables and load data from Amazon S3.

IMPORTANT:

* If you use different names for objects created in this section, be sure to update scripts and code in the following sections accordingly.
* For each SQL script block below, select all the statements in the block and execute them top to bottom.

In a new SQL worksheet, run the following SQL commands to create the [warehouse](https://docs.snowflake.com/en/sql-reference/sql/create-warehouse.html), [database](https://docs.snowflake.com/en/sql-reference/sql/create-database.html) and [schema](https://docs.snowflake.com/en/sql-reference/sql/create-schema.html).

USE ROLE ACCOUNTADMIN;

CREATE WAREHOUSE IF NOT EXISTS DASH\_S WAREHOUSE\_SIZE=SMALL;

CREATE DATABASE IF NOT EXISTS DASH\_DB;

CREATE SCHEMA IF NOT EXISTS DASH\_SCHEMA;

USE DASH\_DB.DASH\_SCHEMA;

USE WAREHOUSE DASH\_S;

In the same SQL worksheet, run the following SQL commands to create table CAMPAIGN\_SPEND from data hosted on publicly accessible S3 bucket.

CREATE or REPLACE file format csvformat

skip\_header = 1

type = 'CSV';

CREATE or REPLACE stage campaign\_data\_stage

file\_format = csvformat

url = 's3://sfquickstarts/ad-spend-roi-snowpark-python-scikit-learn-streamlit/campaign\_spend/';

CREATE or REPLACE TABLE CAMPAIGN\_SPEND (

CAMPAIGN VARCHAR(60),

CHANNEL VARCHAR(60),

DATE DATE,

TOTAL\_CLICKS NUMBER(38,0),

TOTAL\_COST NUMBER(38,0),

ADS\_SERVED NUMBER(38,0)

);

COPY into CAMPAIGN\_SPEND

from @campaign\_data\_stage;

In the same SQL worksheet, run the following SQL commands to create table MONTHLY\_REVENUE from data hosted on publicly accessible S3 bucket.

CREATE or REPLACE stage monthly\_revenue\_data\_stage

file\_format = csvformat

url = 's3://sfquickstarts/ad-spend-roi-snowpark-python-scikit-learn-streamlit/monthly\_revenue/';

CREATE or REPLACE TABLE MONTHLY\_REVENUE (

YEAR NUMBER(38,0),

MONTH NUMBER(38,0),

REVENUE FLOAT

);

COPY into MONTHLY\_REVENUE

from @monthly\_revenue\_data\_stage;

In the same SQL worksheet, run the following SQL commands to create table BUDGET\_ALLOCATIONS\_AND\_ROI that holds the last six months of budget allocations and ROI.

CREATE or REPLACE TABLE BUDGET\_ALLOCATIONS\_AND\_ROI (

MONTH varchar(30),

SEARCHENGINE integer,

SOCIALMEDIA integer,

VIDEO integer,

EMAIL integer,

ROI float

)

COMMENT = '{"origin":"sf\_sit-is", "name":"aiml\_notebooks\_ad\_spend\_roi", "version":{"major":1, "minor":0}, "attributes":{"is\_quickstart":1, "source":"streamlit"}}';

INSERT INTO BUDGET\_ALLOCATIONS\_AND\_ROI (MONTH, SEARCHENGINE, SOCIALMEDIA, VIDEO, EMAIL, ROI)

VALUES

('January',35,50,35,85,8.22),

('February',75,50,35,85,13.90),

('March',15,50,35,15,7.34),

('April',25,80,40,90,13.23),

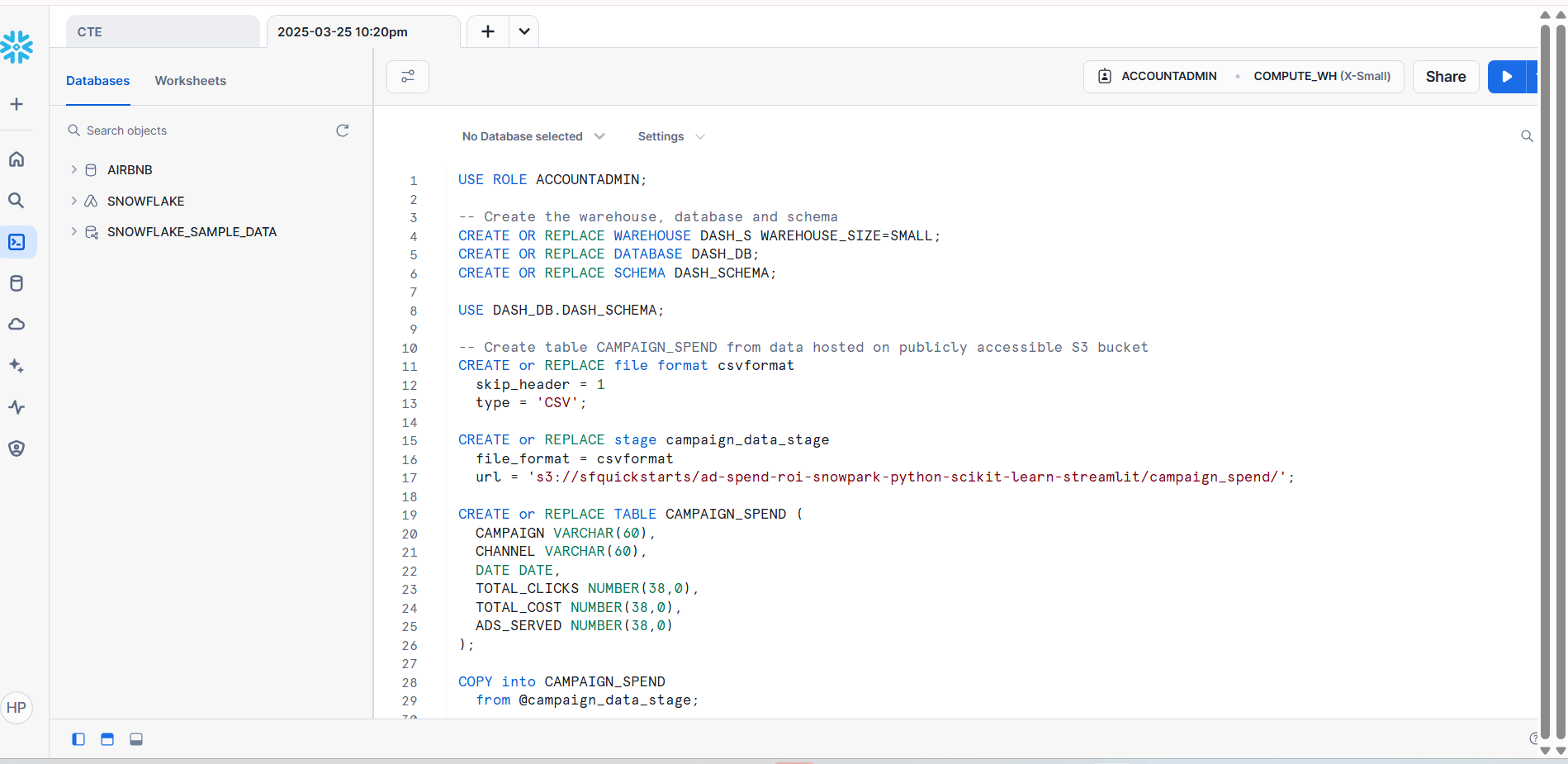
('May',95,95,10,95,6.246),

('June',35,50,35,85,8.22);

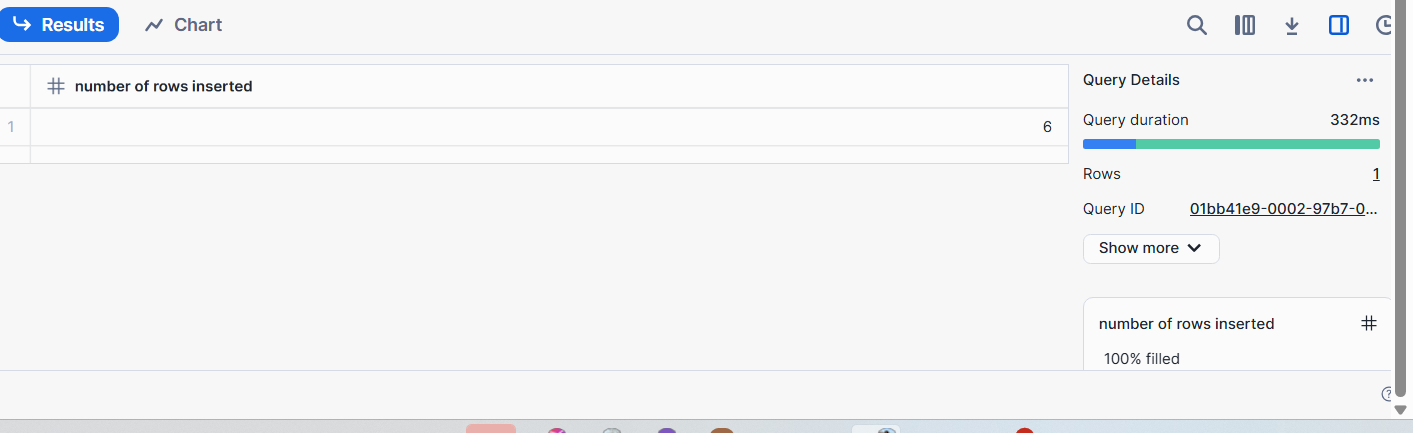
Optionally, you can also open [setup.sql](https://github.com/Snowflake-Labs/sfguide-getting-started-dataengineering-ml-snowpark-python/blob/main/setup.sql) in Snowsight and run all SQL statements to create the objects and load data from AWS S3.

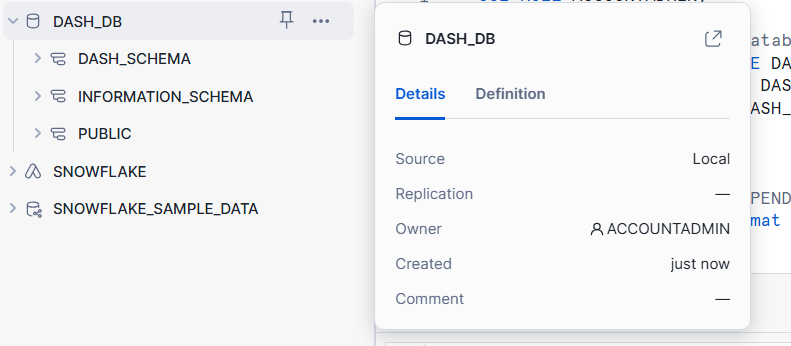
IMPORTANT: If you use different names for objects created in this section, be sure to update scripts and code in the following sections accordingly.

Executed the sql queries in snowflake notebook -

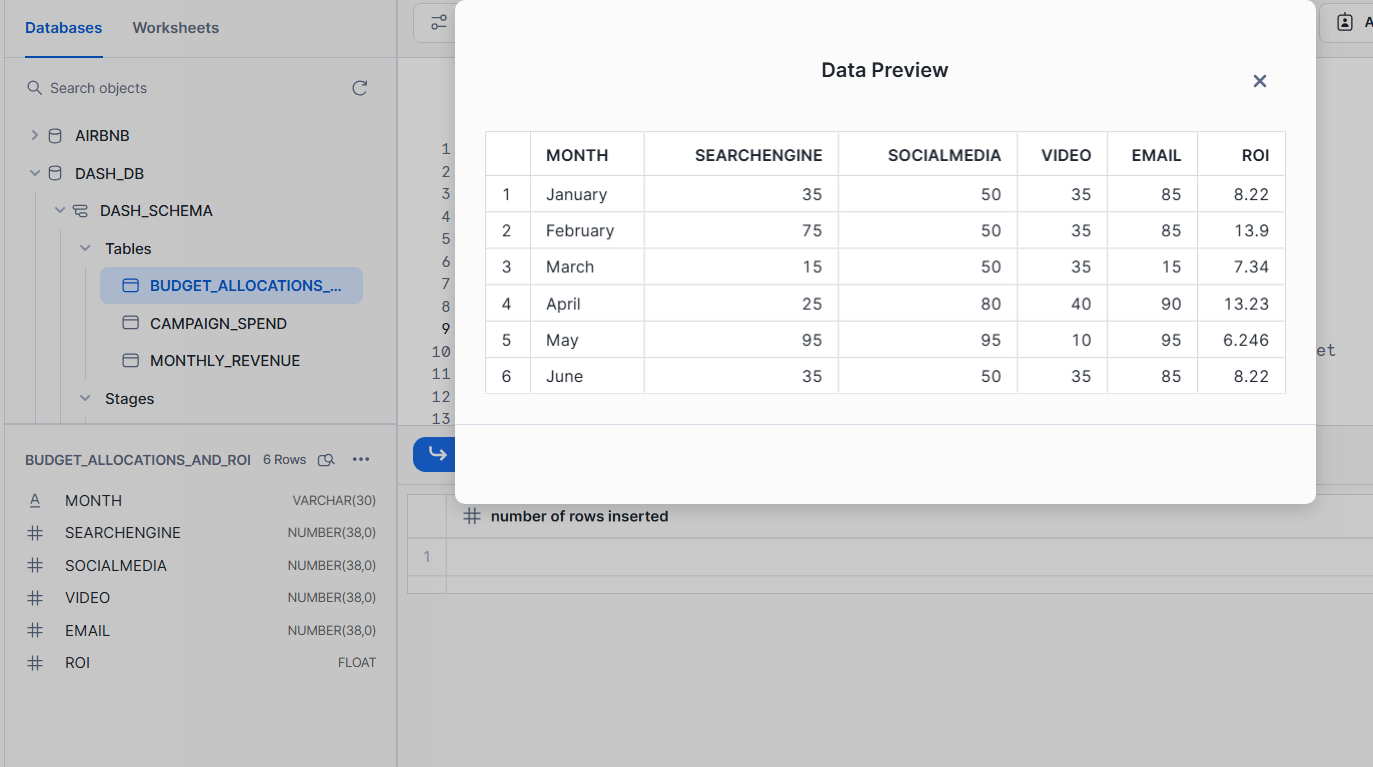


Ran successfully -



Got new db -   


Eg view for budget allocation -



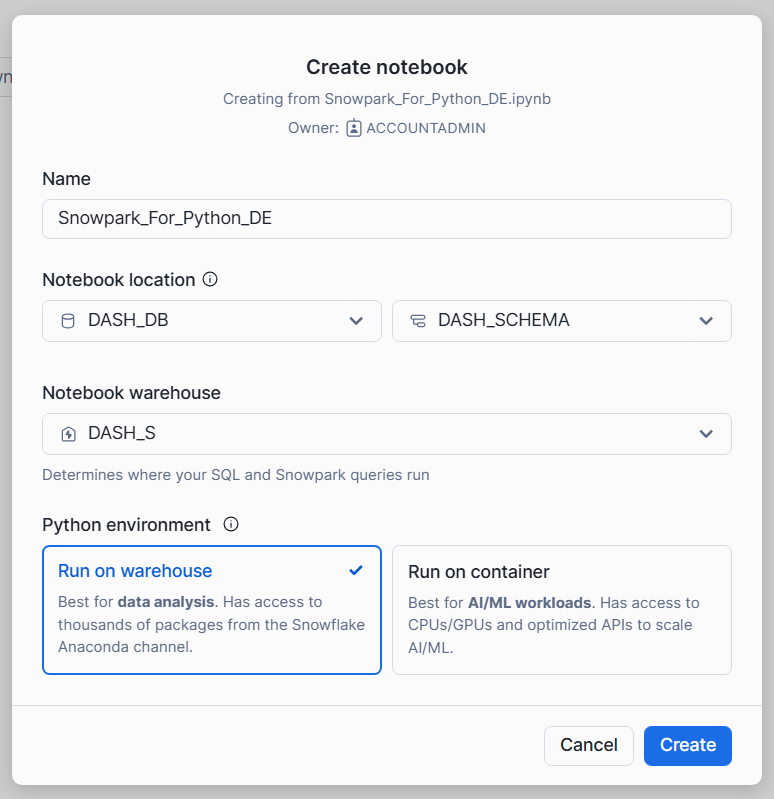
Now this was data ingestion from s3 bucket into snowflake.

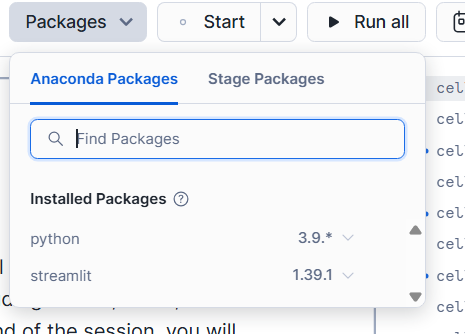
## 2. Data Engineering - Data Transformations using Snowpark (Python)

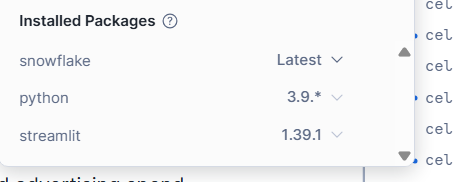
Now, data engineering starts

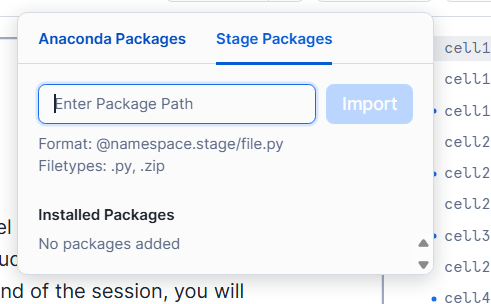
There are two types of python env on snowflake notebook -

Run on warehouse and run on container. The down side of running on warehouse is that for larger load of ml it was not great idea to explode compute resource on warehouse.   
For now i got run on warehouse option as the ml load is not that much to exploit the compute resource on warehouse.



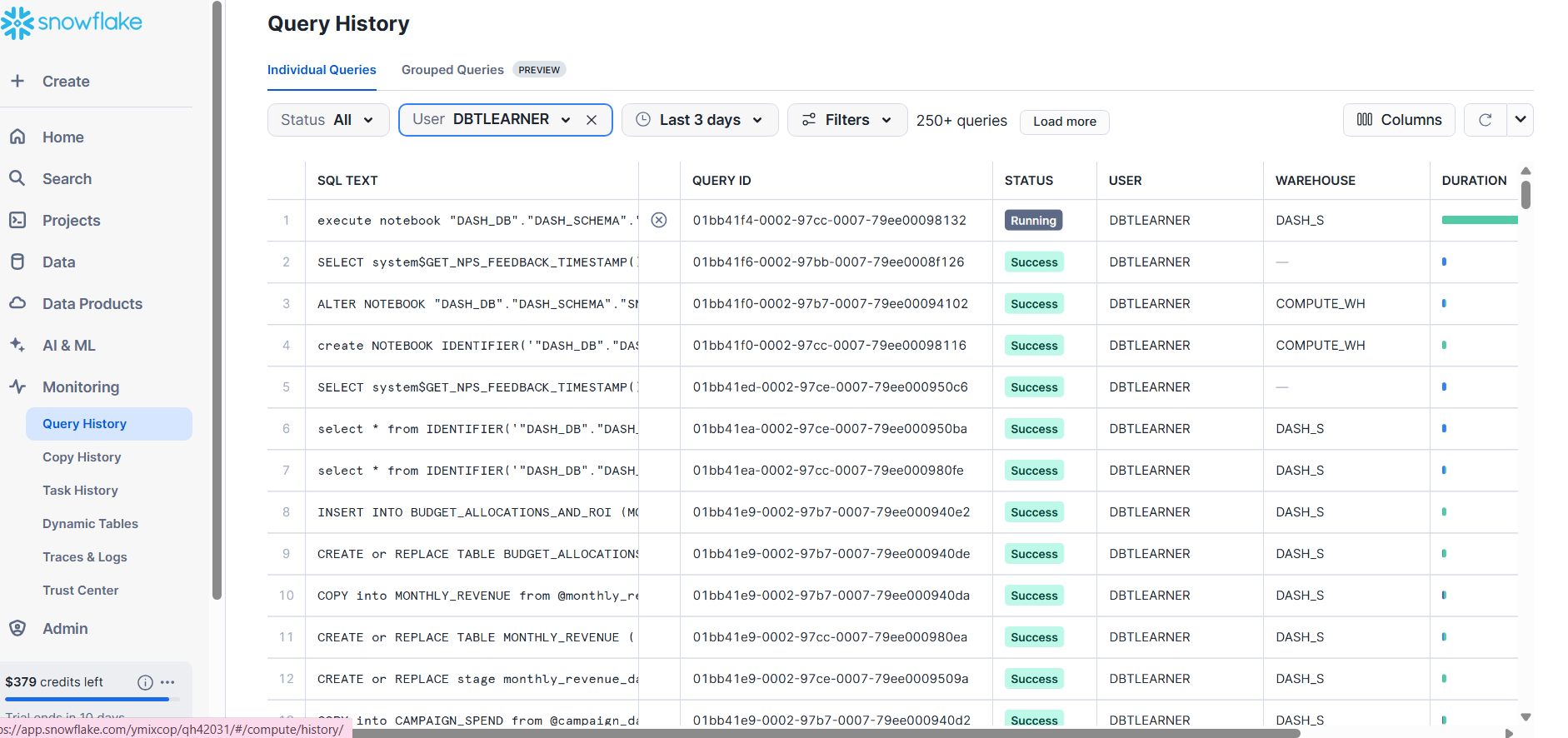


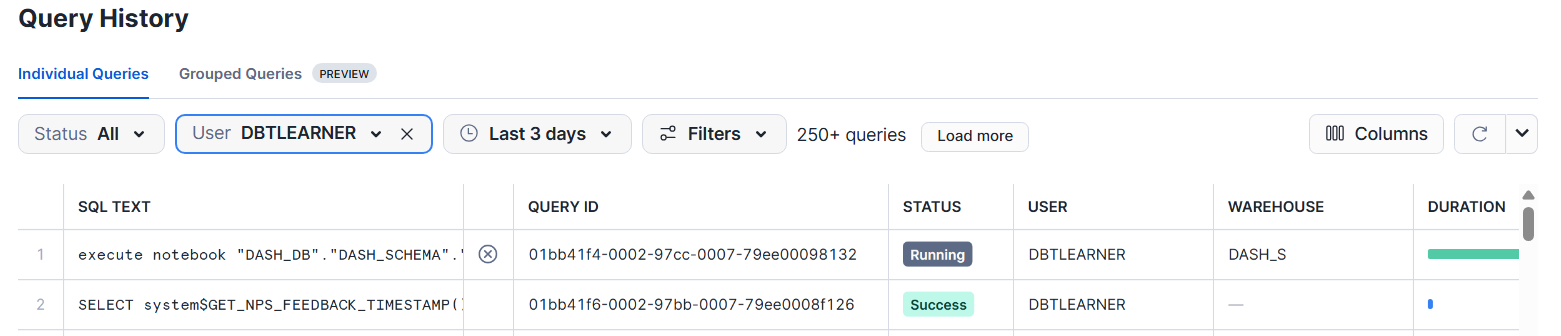




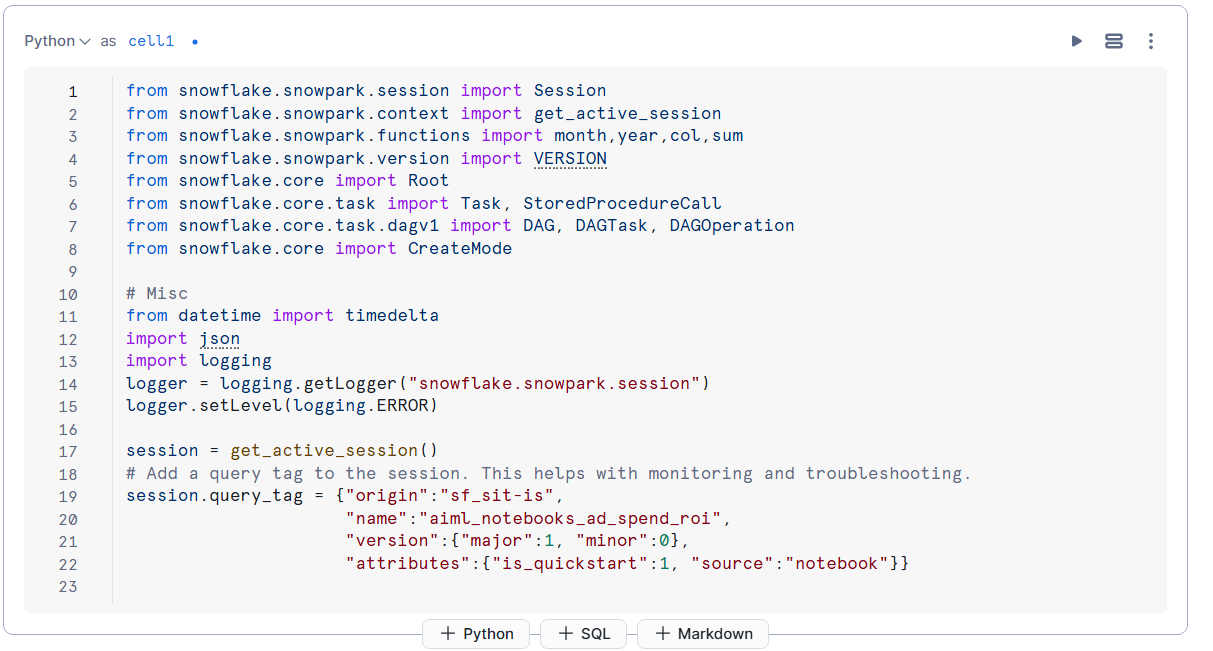
Can also import custom build libraries.

\*\*\*\*\*\* Can also view query history for this project -





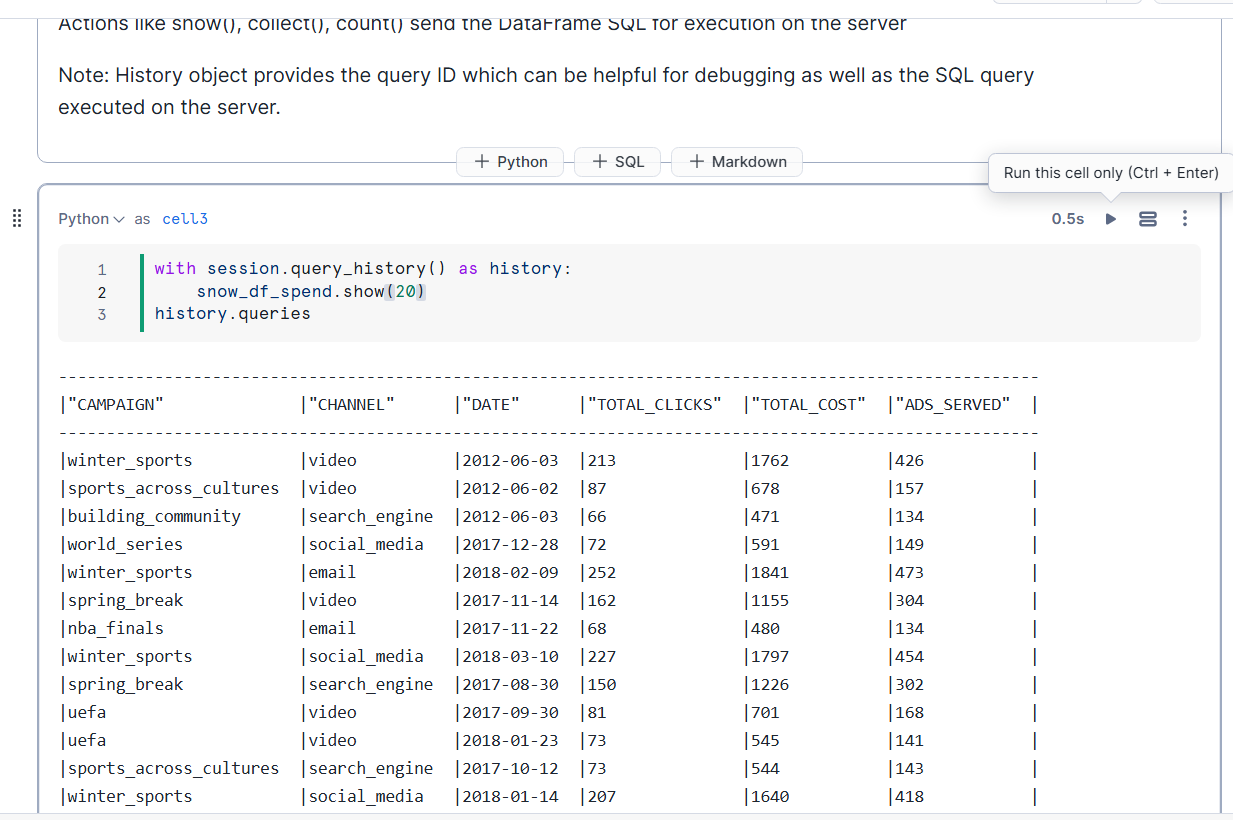
Created a session -

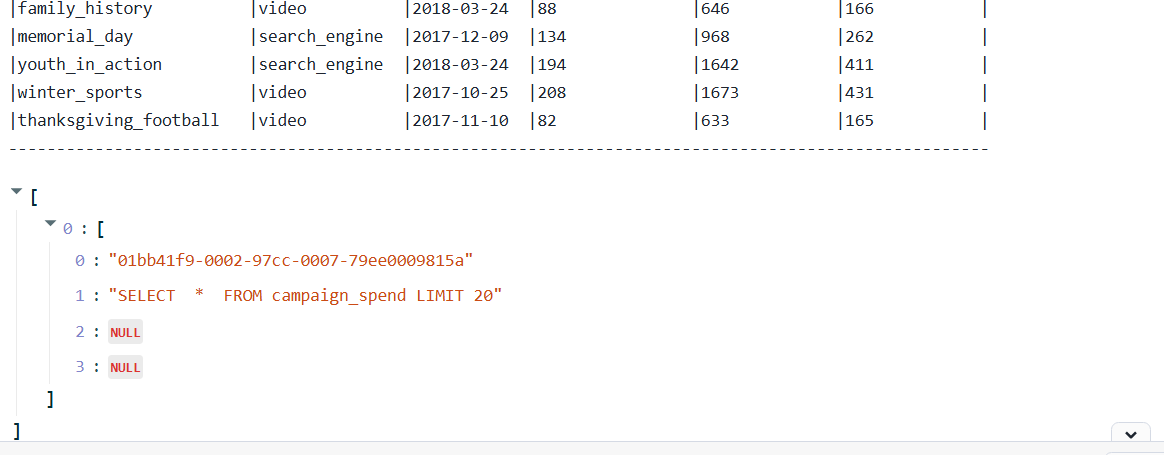


Here we used json query tag

The below ‘.queries’ shows the sql query that’s being runned under the python snowpark dataframe.-

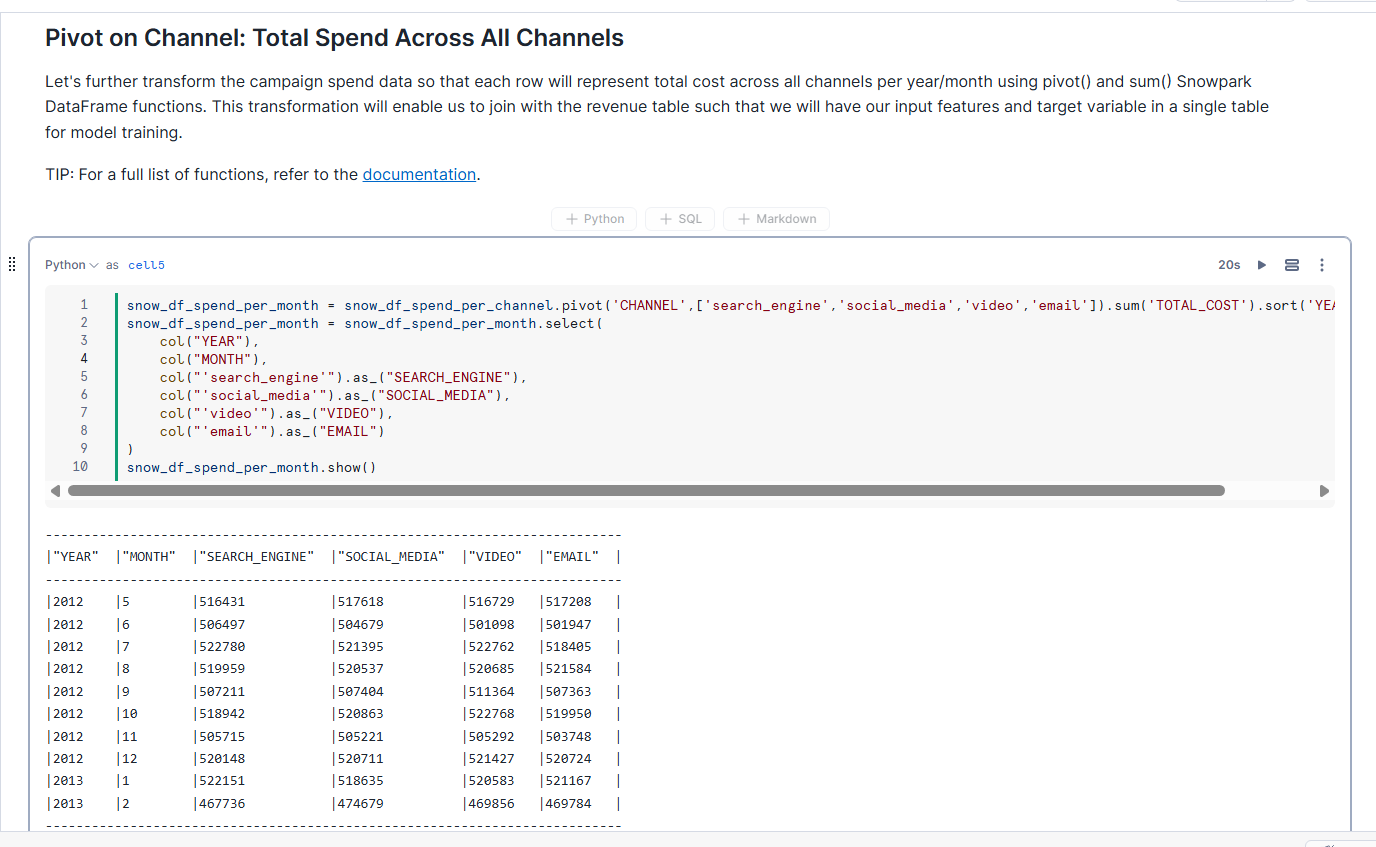






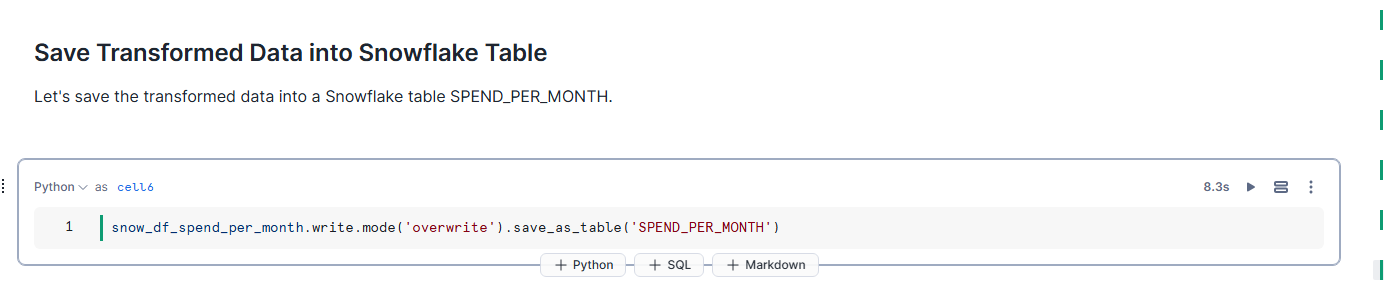
This is how the sql code complied looks like. The same aggregation logic is used behin python but writing code in python makes it much more easier for complex tasks.

Per month spend across all the channels -   

## 3. Write the transformed data to snowflake schema (Data Warehouse)

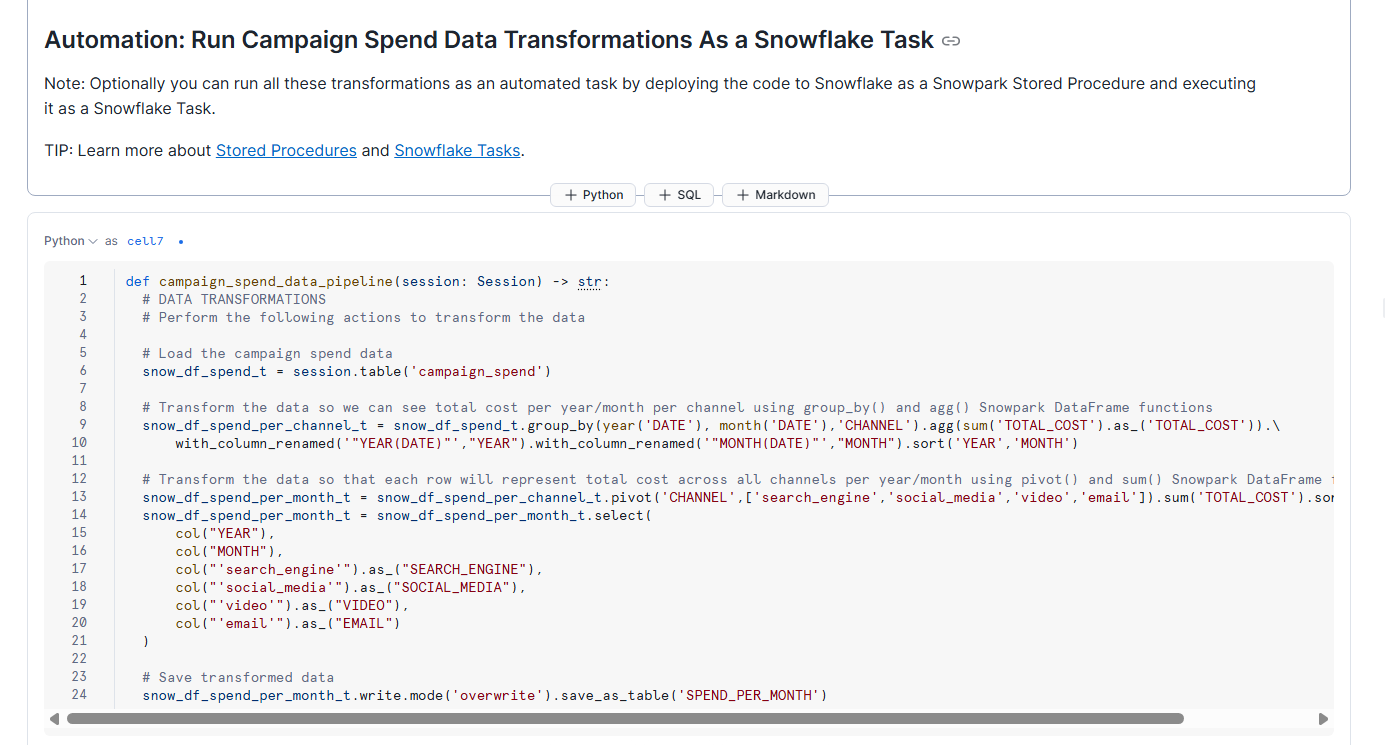
This line writes the transformed Snowpark DataFrame to a table named **SPEND\_PER\_MONTH**, overwriting any existing data. It ensures that the final summarized results are permanently stored and easily accessible within Snowflake.



## 4. Automated Data Transformations via Stored Procedure

This code block defines a pipeline that transforms raw campaign spend data into aggregated monthly results using Snowpark’s DataFrame operations. By packaging the logic as a Snowflake Stored Procedure and scheduling it as a Snowflake Task, you can automate the entire process and ensure your analytics table remains up-to-date without manual intervention.

This is pipelines - snowflake task and wrapped up as a snowflake stored procedure -

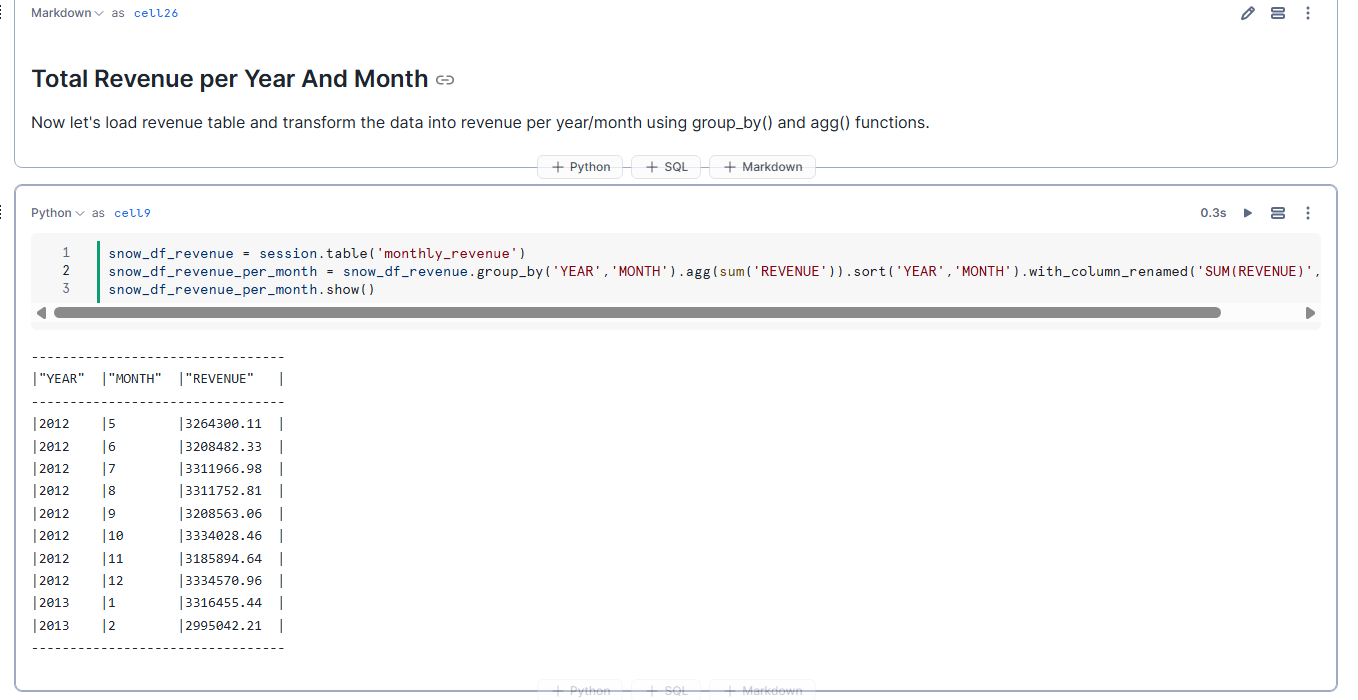


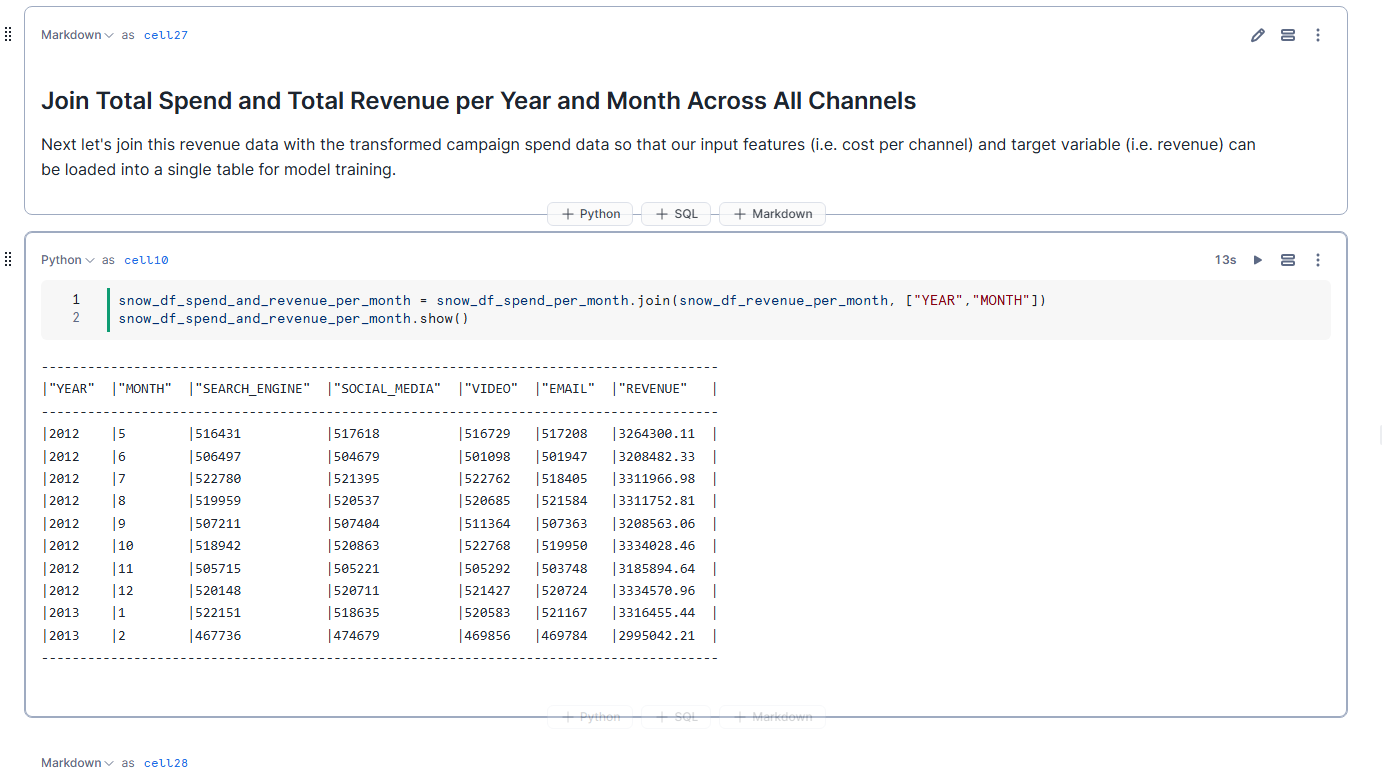
This above function (campaign\_spend\_data\_pipeline) retrieves raw campaign spend data from a Snowflake table, performs grouping and aggregation by month, year, and channel, then writes the summarized results into a new table (SPEND\_PER\_MONTH). By packaging this logic as a Snowflake Stored Procedure and running it as a Task, you can automate the entire process to keep your analytics data consistently up to date

## 5. Additional Data Transformations and Joining Spend with Revenue

1. **Group and Aggregate by Year and Month**
   * **What’s Happening:** The code uses a groupBy function on the raw revenue dataset by YEAR and MONTH, applying aggregation functions (like sum) to calculate total revenue.
   * **Why:** This step transforms granular transaction-level data into summarized metrics, making it easier to analyze revenue trends over time.
2. **Merging Spend and Revenue**
   * **What’s Happening:** After generating a DataFrame with aggregated revenue (e.g., snow\_df\_revenue\_per\_month), the code joins it with the spend DataFrame (snow\_df\_spend\_per\_month) on the matching columns YEAR and MONTH.
   * **Why:** By consolidating both spend and revenue in one place, you get a single, unified view of your performance metrics. This is crucial for downstream analytics, reporting, and model-building tasks, where you often need both cost (spend) and return (revenue) together.

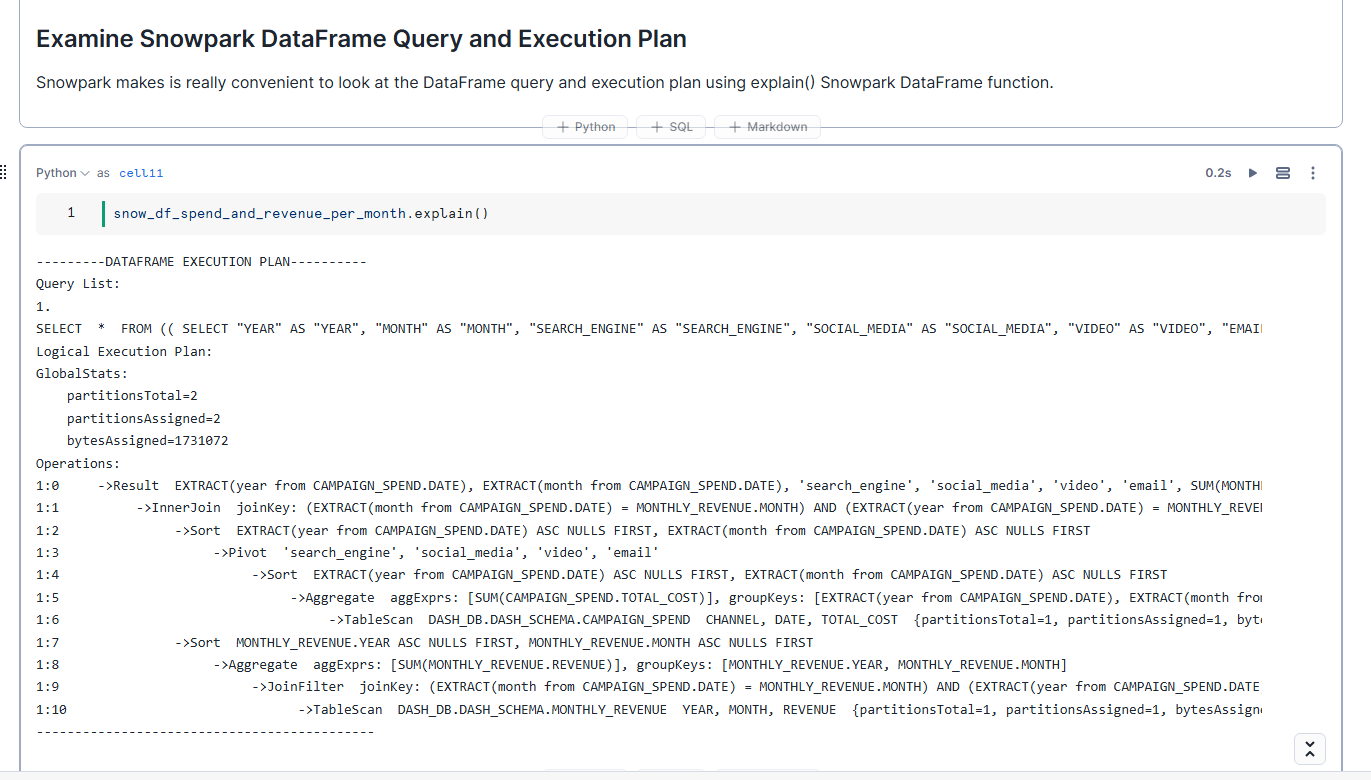






## 6. Viewing Query and Execution Plan with explain()

Calling the explain() method on a Snowpark DataFrame (snow\_df\_spend\_and\_revenue\_per\_month.explain()) displays how Snowflake plans to execute the underlying SQL operations. This includes the logical query plan and any optimizations that Snowflake applies, helping you understand and tune performance for complex transformations.

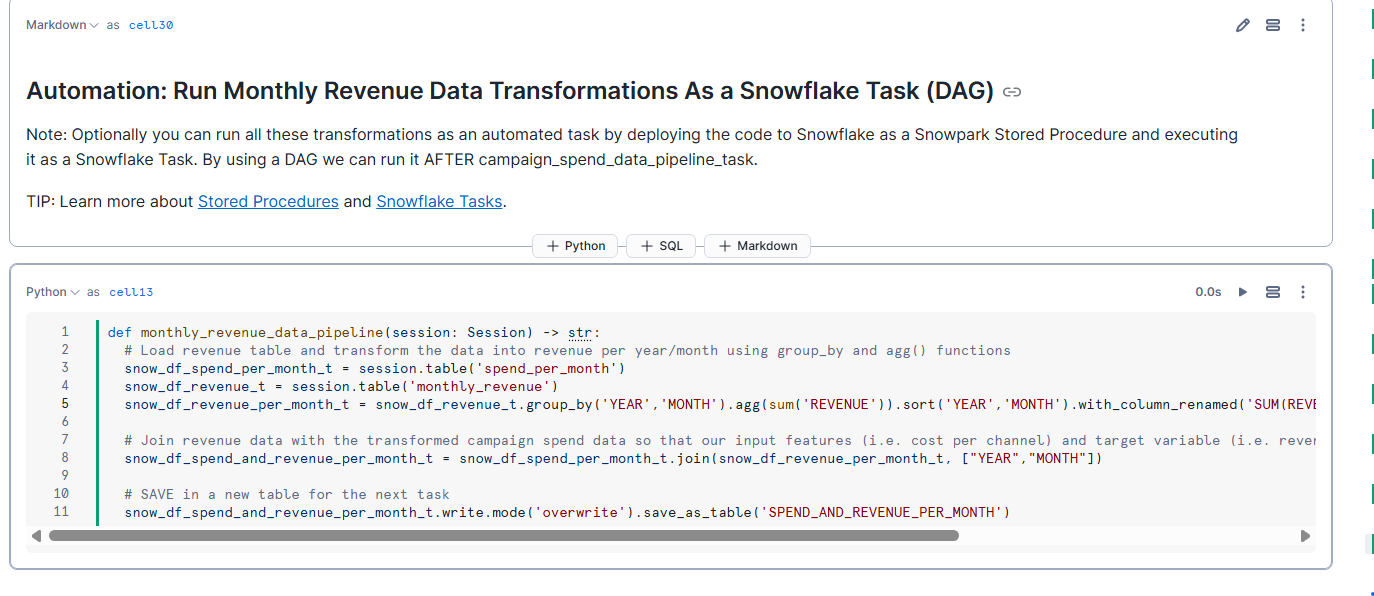


Again use the write function to save the transformed data into snowflake table

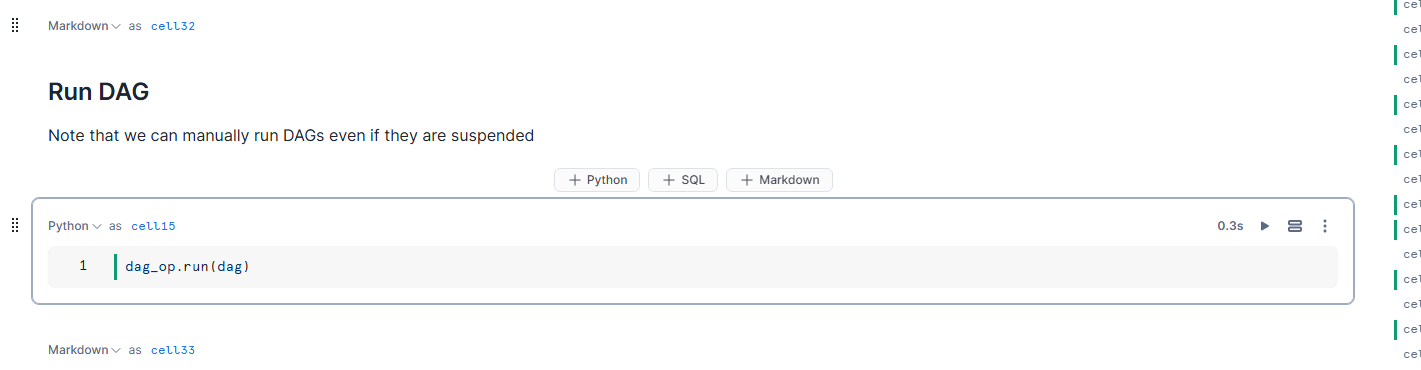


## 7. Automation with a Directed Acyclic Graph (DAG) of Snowflake Tasks

This code block demonstrates how to orchestrate multiple data transformation steps (pipelines) in a sequence using Snowflake Tasks, effectively creating a Directed Acyclic Graph (DAG). It defines a **monthly\_revenue\_data\_pipeline** function for aggregating and merging revenue data, then schedules that function as a **Snowflake Task** to run automatically. By chaining tasks together—such as running this revenue pipeline after the spend pipeline—you can build a robust, end-to-end data workflow that updates your analytics tables without manual intervention.

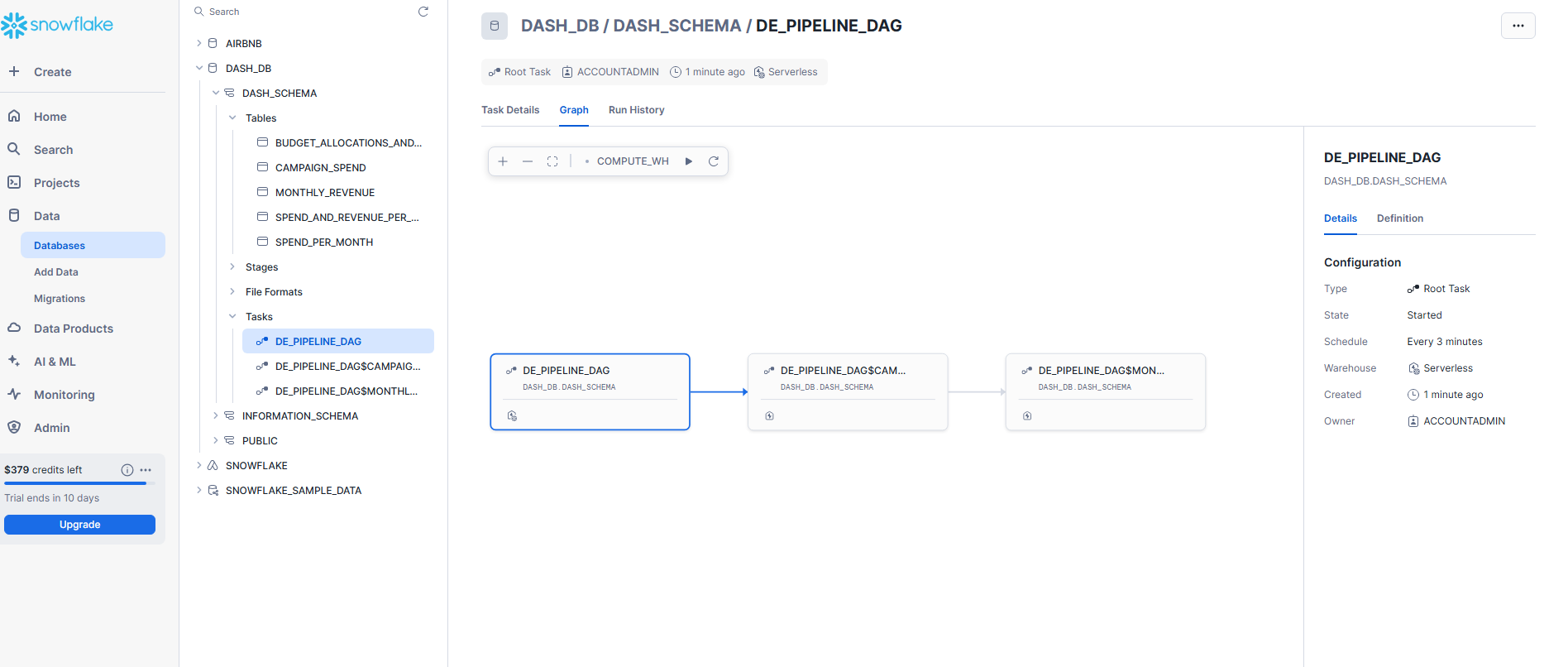






## 8. Visualizing the Task DAG in Snowflake’s UI

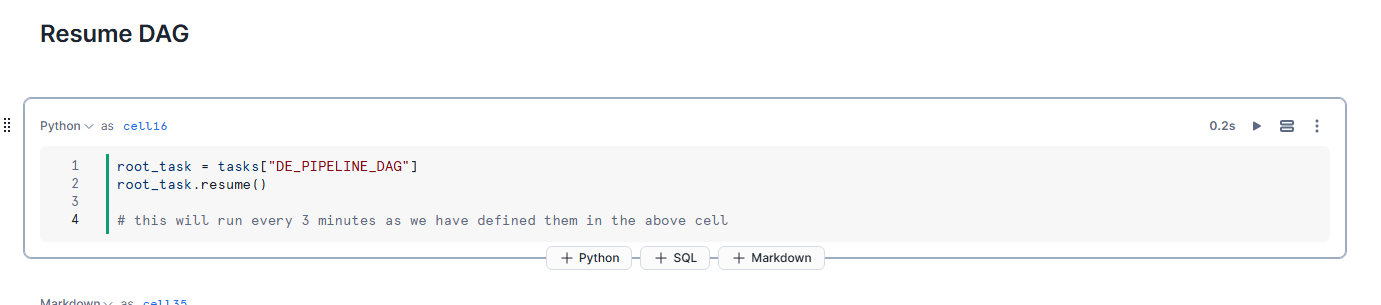
This screenshot shows the **Snowflake Tasks** user interface, where each box represents a scheduled task (e.g., a data pipeline step), and the arrows indicate the order in which they run. By laying out tasks in a Directed Acyclic Graph (DAG), you can easily see the dependencies between pipelines—such as the revenue aggregation task running after the spend aggregation task—and confirm that your automated data workflows execute in the correct sequence.

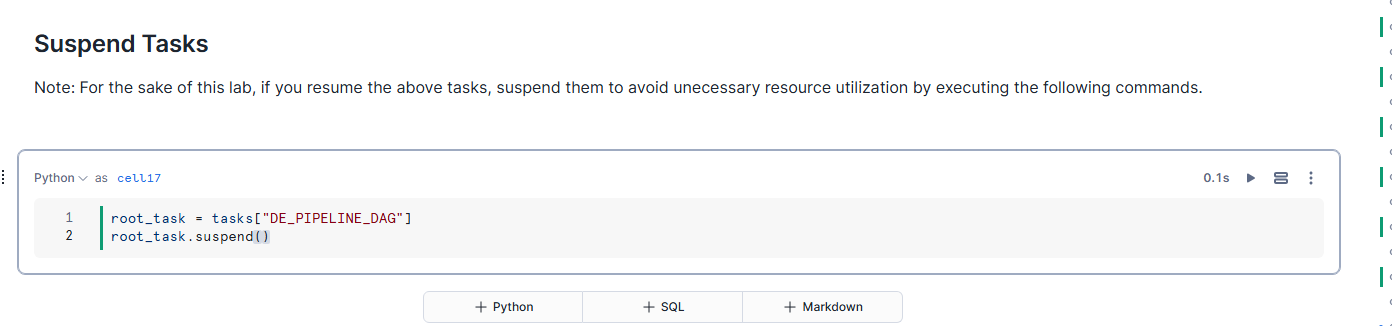


## 

## 9. Suspending Tasks to Manage Resource Usage

Suspending a Snowflake Task temporarily halts its scheduled execution, preventing any associated compute from running and incurring costs. This is useful when you no longer need a task to run (e.g., during development or testing) or if you want to pause your data pipelines while making changes. By suspending tasks, you ensure that no unnecessary processing occurs, helping optimize both resource usage and costs



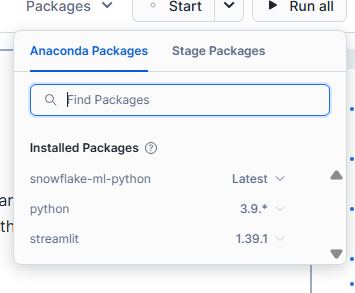


We grabbed some root tables - done pivoting- transforming - and dag

# II Machine Learning for Predictive Profit and loss using Linear Regression -

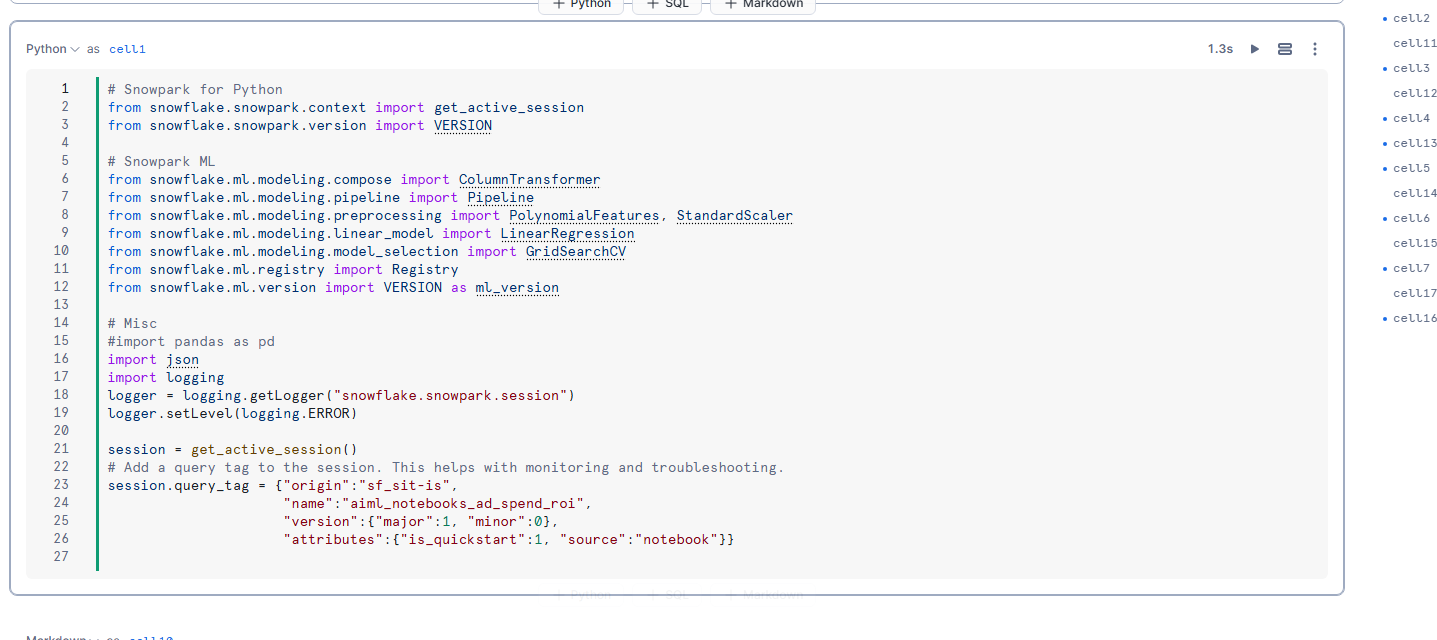
## 1. Importing ML packages in Snowflake

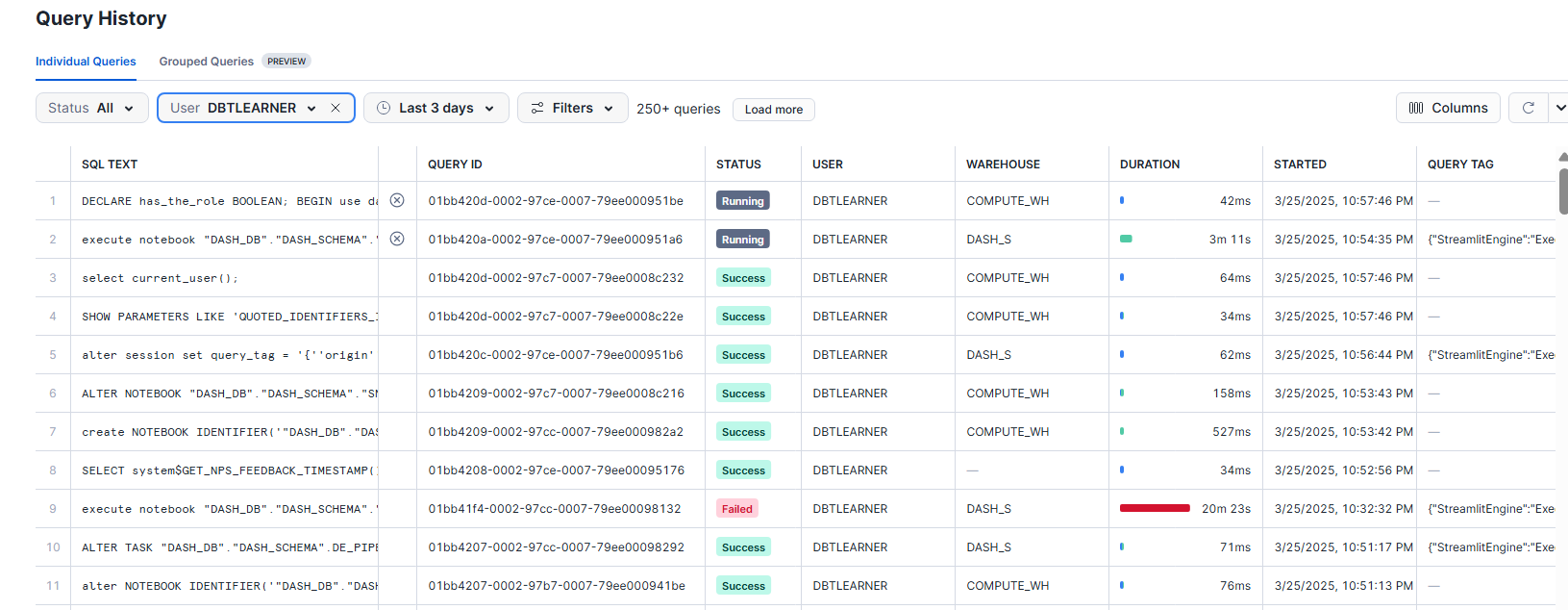
Find the packages in snowflake through the packages section.



Added the library package

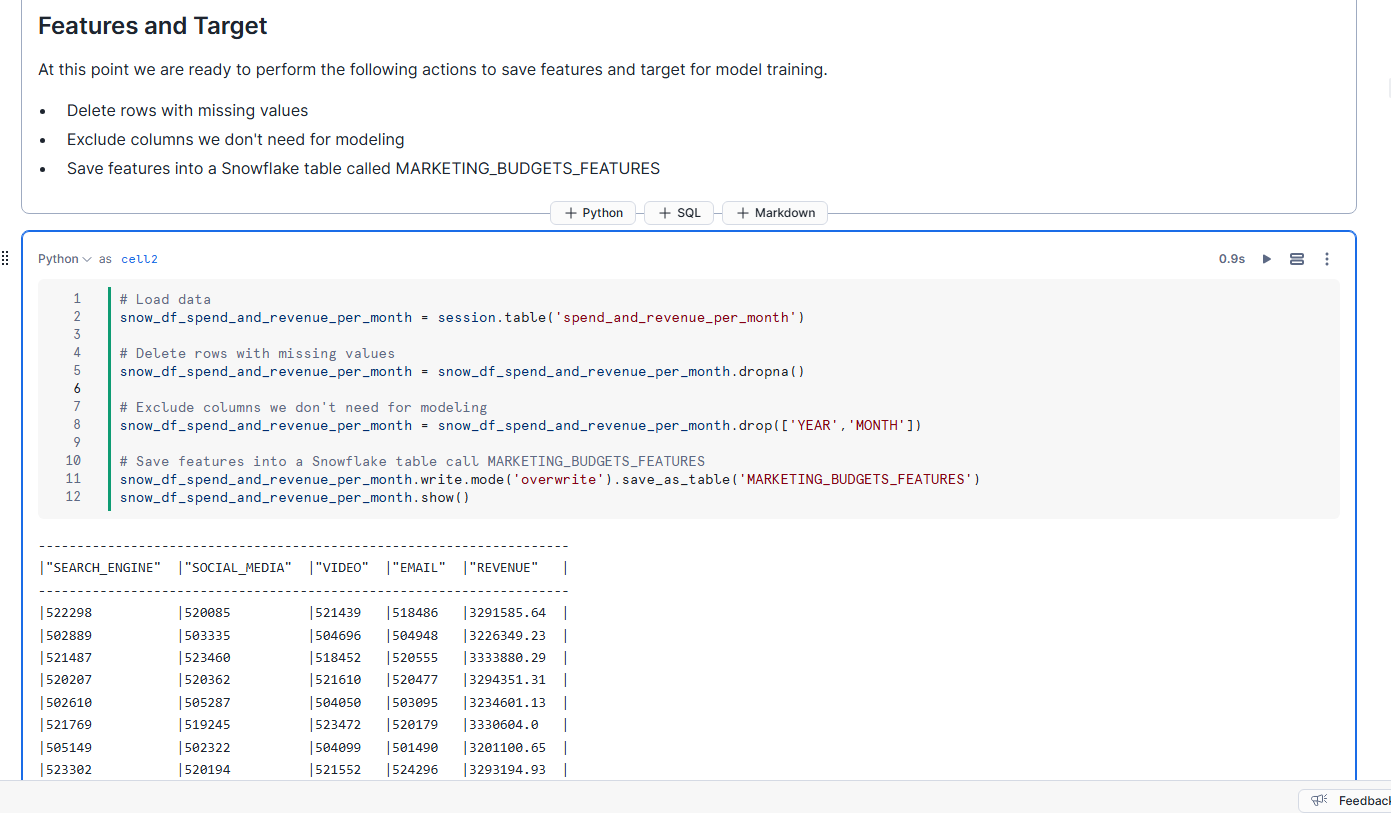
This code snippet configures a Snowpark ML environment by importing **Snowflake ML** libraries (for transformations, pipelines, and regression models) and creating a **Snowflake session**. It also sets up logging and attaches a custom **query tag**, which helps with auditing and debugging by tracking where and why queries are executed



This below screenshot shows Snowflake’s **Query History** interface, where you can see details for each executed query—such as **SQL text**, **query ID**, **status** (e.g., *success*, *failed*, *running*), **user**, **warehouse**, **duration**, and **query tag**. It’s a useful tool for monitoring performance, diagnosing issues, and auditing data operations over a specified time range.  


## 2. Preparing Features for Modeling

This code loads the **spend\_and\_revenue\_per\_month** dataset, removes any rows with missing values, and excludes columns that aren’t needed for machine learning. It then saves the refined data into a new Snowflake table called **MARKETING\_BUDGETS\_FEATURES**, providing a clean, focused dataset for subsequent model training.

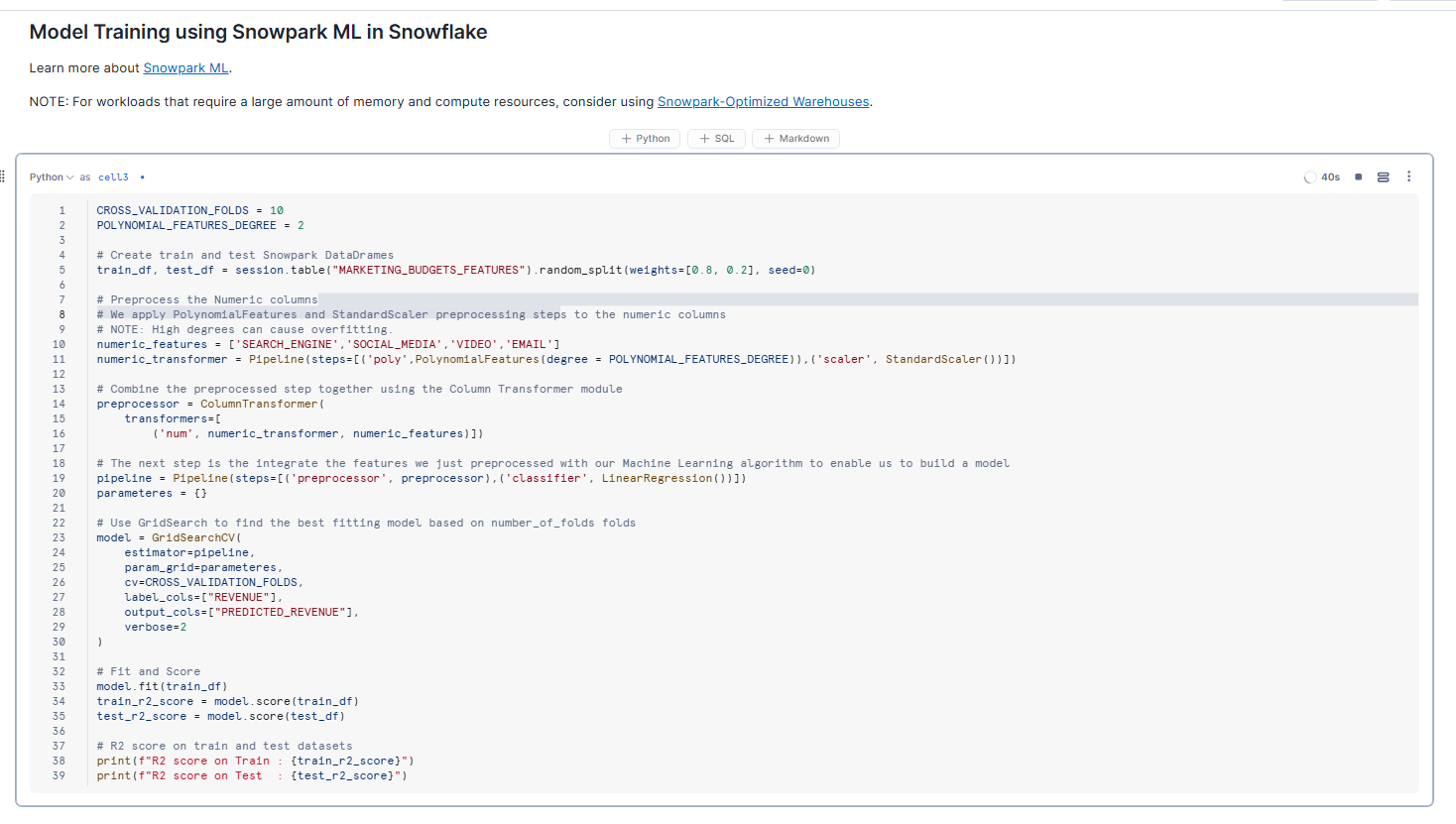


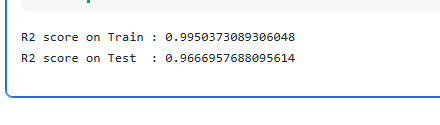
## 3. Model Training using Snowpark ML in Snowflake

This code demonstrates a typical machine learning workflow directly within Snowflake using **Snowpark ML**:

1. **Data Splitting:** The **MARKETING\_BUDGETS\_FEATURES** table is loaded and randomly split into training and testing sets.
2. **Feature Preprocessing:** Numeric columns are standardized (e.g., using a **StandardScaler**) to ensure the model handles varying data scales appropriately.
3. **Model Training and Evaluation:** A regression model (in this case, a **LinearRegression** model) is trained on the training set and then scored on both the training and test sets. The reported **R²** scores (near 0.99 on train and 0.96 on test) indicate the model is fitting the data well, although you should still validate against potential overfitting and other model performance considerations.

## 



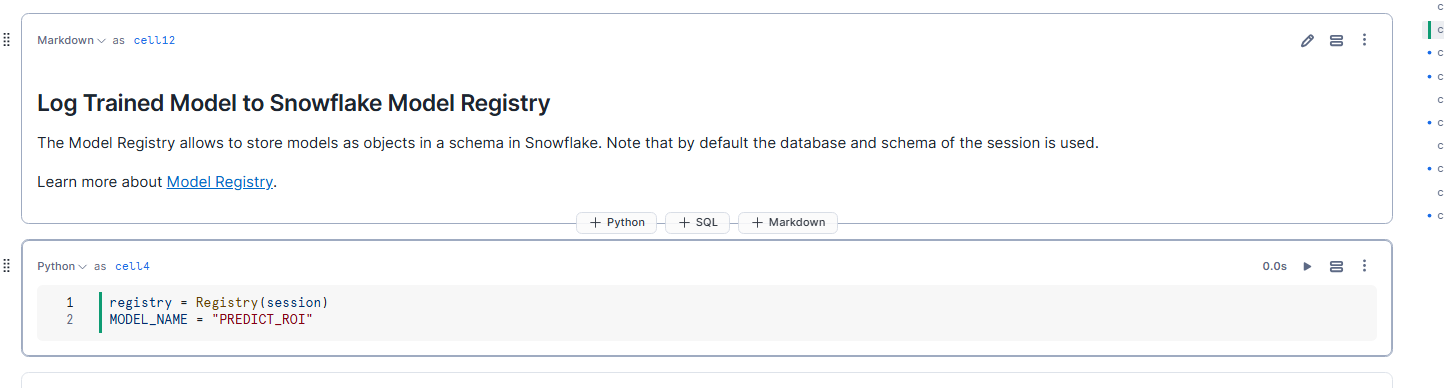


## 4. Logging and Managing Trained Models in Snowflake’s Model Registry

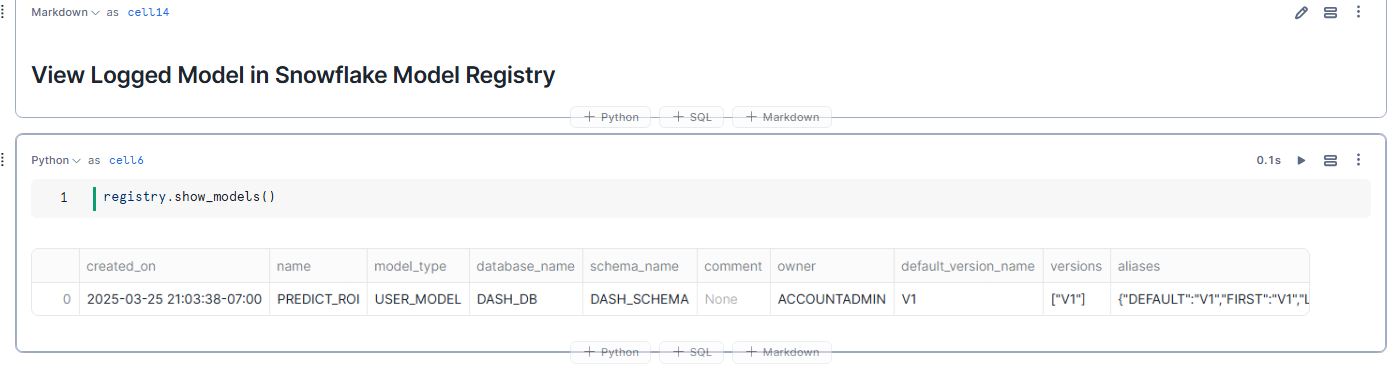
This sequence of code cells shows how to register a trained model in Snowflake’s Model Registry:

1. **Create a Registry Instance:** Instantiate a **Registry** object tied to your current Snowflake session.
2. **Log the Model:** Use **log\_model()** to store the model (along with its version, evaluation metrics like R² scores, and a descriptive comment) as an object in Snowflake.
3. **View Registered Models:** Call **registry.show\_models()** to confirm the model was successfully logged, displaying metadata such as the model name, version, and schema location.

Storing models this way centralizes their management and makes them easy to track, deploy, and maintain over time.







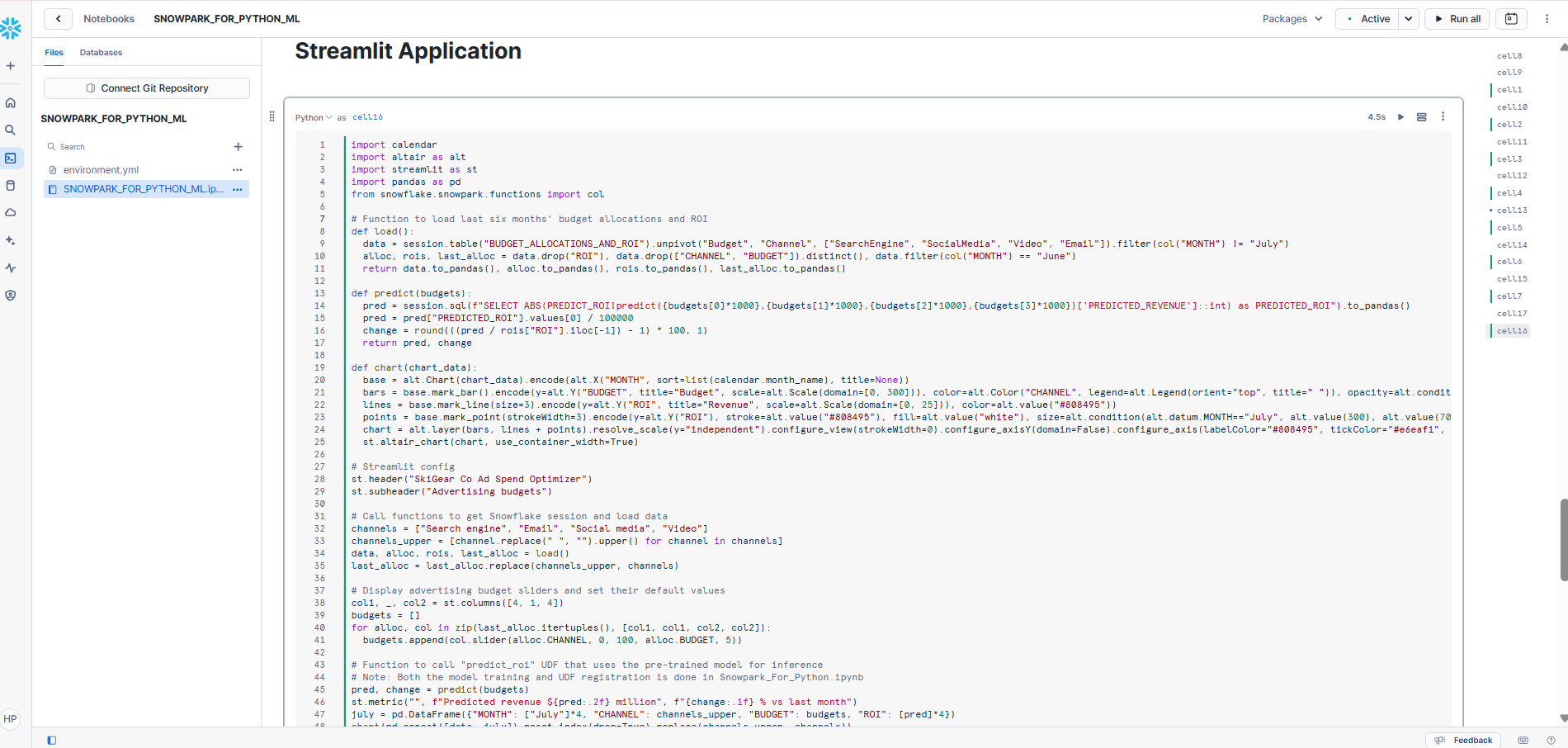
## 5. Running Inference on New Data

Here, a new Snowpark DataFrame (test\_df) is created with sample input values for each marketing channel (e.g., SEARCH\_ENGINE, SOCIAL\_MEDIA). Then, by calling **mu.run\_test\_df(...)** and specifying the logged model’s function name (e.g., predict\_bid or predict\_revenue), the code generates predictions for the unseen data directly in Snowflake. This step demonstrates how you can apply your trained and registered model to make real-time or batch predictions without leaving the Snowflake environment.



## 6. Streamlit Application for Ad Spend Optimization

## This code showcases a **Streamlit** application that takes user-input budgets across various marketing channels (e.g., Search Engine, Social Media, Video, Email) and predicts the resulting revenue using the trained Snowflake ML model. It presents the predicted revenue in real time and visualizes the spending distribution across months, offering an interactive way to explore different budget scenarios and optimize marketing strategies—all while remaining connected to Snowflake for data and model inference

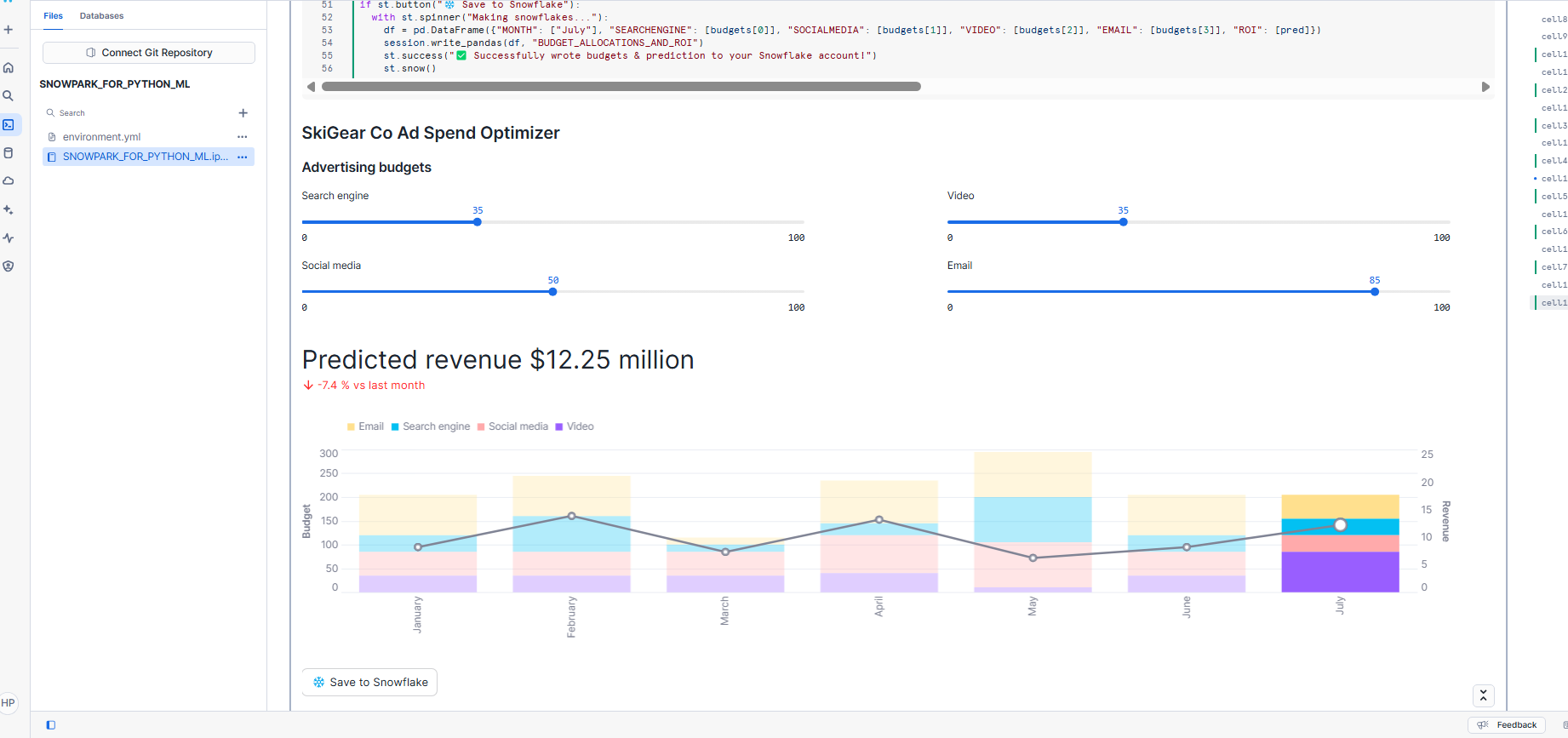


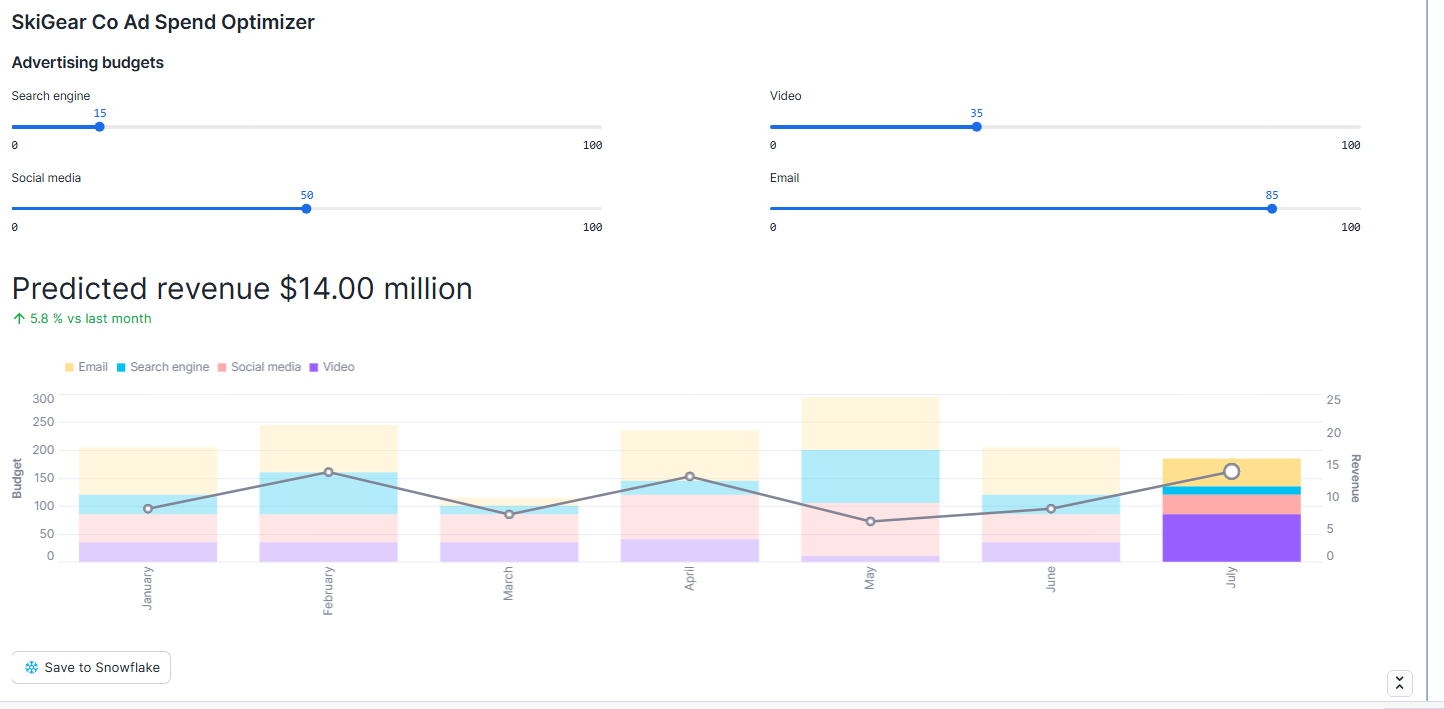
## 7. Interactive Dashboard and Insights Overview

This dashboard, built with Streamlit, ties together the earlier data transformations and machine learning steps to provide actionable marketing insights. Here's how it works:

* **Data Transformations Integration:** The dashboard leverages aggregated spend and revenue data (prepared via Snowpark transformations and stored in Snowflake tables) to build a unified view of marketing performance. These transformations—such as grouping by month/year and merging spend with revenue—ensure that the input data is clean, summarized, and ready for analysis.
* **Real-Time Predictions:** Using the trained and registered ML model, the dashboard allows users to input different budget scenarios across marketing channels. It then runs inference directly in Snowflake to predict resulting revenue, enabling users to simulate various strategies interactively.
* **Visualization and Insights:** Interactive charts and graphs display how budgets are allocated across channels and over time, while the predicted revenue helps quantify the impact of spending changes. This provides both high-level trends and detailed insights, making it easier to optimize marketing investments based on historical data patterns and real-time forecasts.

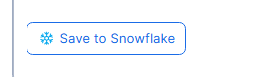
In summary, the dashboard connects the dots from raw data transformations to advanced predictive analytics, empowering users to make informed, data-driven decisions about their marketing strategies.

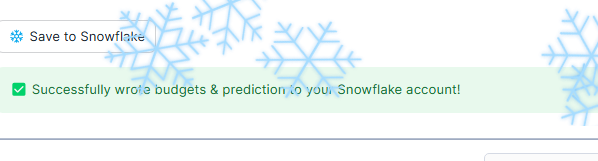




## 8. Additional Capabilities: Data Masking and Notebook Scheduling

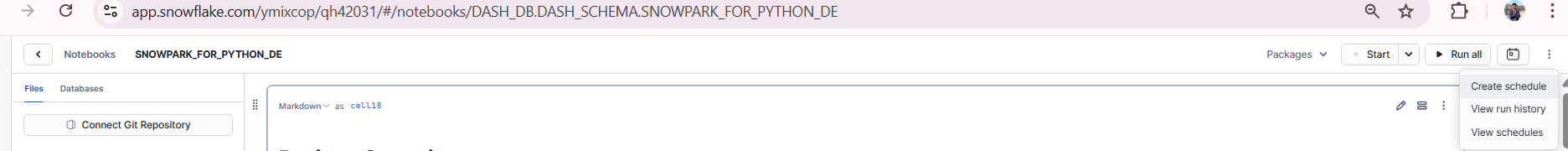
1. **Data Masking with Policies:** Snowflake allows you to define **masking policies** that automatically hide or obfuscate sensitive data (e.g., personally identifiable information) at query time. These policies can be applied to production instances and propagate seamlessly to **cloned** databases, ensuring consistent data protection across environments.
2. **Scheduling Notebook Runs:** You can schedule your Snowflake notebooks to run at regular intervals (e.g., hourly or daily), automating tasks such as data transformations, model training, or dashboard updates. This lets you maintain up-to-date insights without manual intervention, further streamlining your data workflows.

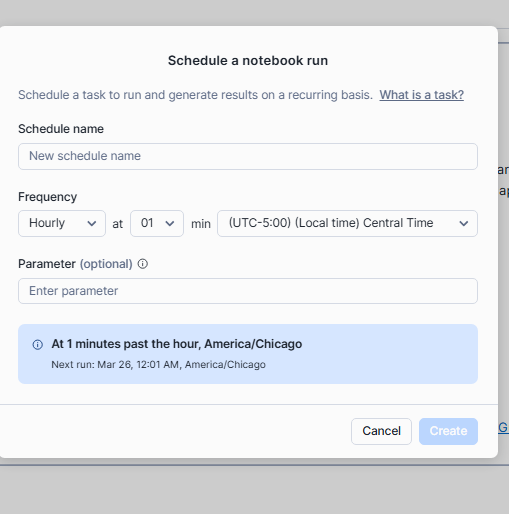




Can also perform datamasking with the help of policies. Rules can be applied to prod instance and can be drawn to the cloned as well.

Can also schedule the notebooks -





Instead of these we can also use tasks.

Summary

This project leverages Snowflake’s Snowpark and Snowflake ML capabilities to create a robust data engineering and analytics engineering solution. Here's an overview of what was accomplished:

* **Data Ingestion & Transformation:** Raw marketing campaign spend and revenue data are ingested directly from Snowflake tables and transformed using Snowpark’s DataFrame API. Aggregations (grouping by month, year, and channel) are performed to create summarized views (e.g., **SPEND\_PER\_MONTH** and **SPEND\_AND\_REVENUE\_PER\_MONTH**) which serve as the foundation for further analysis.
* **Task Orchestration & Automation:** The data transformation pipelines are encapsulated within Snowflake Stored Procedures and orchestrated as a Directed Acyclic Graph (DAG) using Snowflake Tasks. This scheduling mechanism automates the entire workflow, ensuring that data is processed and updated regularly without manual intervention.
* **Machine Learning Workflow:** A Snowpark ML environment is set up to train regression models using the aggregated marketing data. The project involves preprocessing features, splitting data into training and testing sets, and training a linear regression model. The trained model is then logged and managed through Snowflake’s Model Registry, streamlining deployment and versioning.
* **Real-Time Inference & Dashboarding:** A Streamlit application is developed to allow interactive exploration of marketing budget scenarios. Users input different spend values, triggering real-time inference against the trained model. The dashboard visualizes spending trends and predicted revenues, tying back to the earlier data transformations for seamless analytics.
* **Data Governance & Operational Efficiency:** Complementing the technical workflows, the solution implements data masking policies to secure sensitive data, and the notebook scheduling ensures consistent, automated processing—key for maintaining production-grade data pipelines.

Overall, this project illustrates how modern data and analytics engineering practices can be seamlessly integrated within the Snowflake ecosystem. By centralizing data transformations, model training, and interactive reporting, it empowers data engineers and analytics engineers to build scalable, secure, and automated end-to-end workflows for advanced marketing analytics and decision-making.