Documentation Databricks to Snowflake Conversion

To modify the connection from Databricks to Snowflake in the provided project, changes are required both in how the database connection is handled and the necessary parameters for the Snowflake connection. Below is an exploration of how this transition can be accomplished, focusing on key changes in connection logic, authentication, and query execution.

### Overview of Key Changes:

1. **Database Library**:
   * Replace the databricks.sql module with Snowflake's Python connector module, snowflake.connector. This change ensures compatibility with Snowflake's connection and query execution mechanism.
2. **Connection Parameters**:
   * Snowflake requires a different set of connection parameters compared to Databricks. The Snowflake connection includes account details, user credentials, role, warehouse, and database information, while Databricks relies on a server hostname, HTTP path, and access token.
3. **Query Execution**:
   * The structure of querying remains similar, but Snowflake uses slightly different methods for executing SQL queries and fetching results.

### Changes in the helpers\_databricks\_connect.py file

Below are the specific changes needed to modify the DatabricksConnector class to use Snowflake instead of Databricks.

#### Importing the Snowflake Connector:

Replace the databricks.sql import with snowflake.connector:

import snowflake.connector

import os

import pandas as pd

import json

#### Snowflake Connector Class:

The initialization and method for connecting to Snowflake will need to include Snowflake-specific parameters such as account, warehouse, database, schema, role, user, and password.

class SnowflakeConnector():

def \_\_init\_\_(self, input\_text="") -> None:

self.input\_text = input\_text

def get\_snowflake\_json(self) -> dict:

print("Snowflake query")

try:

# Snowflake connection parameters

connection = snowflake.connector.connect(

account=os.environ.get('SNOWFLAKE\_ACCOUNT'),

user=os.environ.get('SNOWFLAKE\_USER'),

password=os.environ.get('SNOWFLAKE\_PASSWORD'),

warehouse=os.environ.get('SNOWFLAKE\_WAREHOUSE'),

database=os.environ.get('SNOWFLAKE\_DATABASE'),

schema=os.environ.get('SNOWFLAKE\_SCHEMA'),

role=os.environ.get('SNOWFLAKE\_ROLE')

)

with connection.cursor() as cursor:

cursor.execute(self.input\_text)

column\_names = [x[0] for x in cursor.description]

result = cursor.fetchall()

result\_df = pd.DataFrame(result, columns=column\_names)

result\_json = result\_df.to\_json(orient="index")

output\_dict = {

'output': {

"data": result\_json,

"columns": column\_names

},

'object': "Not Applicable",

'id': "Not Applicable"

}

return {"status\_code": 200, "response": output\_dict}

except Exception as error:

print(error)

return {"status\_code": 599, "response": str(error)}

#### Key Connection Parameters:

Snowflake's connection uses the following environment variables that need to be configured and passed:

* SNOWFLAKE\_ACCOUNT: Snowflake account name (e.g., abc12345.us-east-1).
* SNOWFLAKE\_USER: Username for Snowflake login.
* SNOWFLAKE\_PASSWORD: Password for Snowflake login.
* SNOWFLAKE\_WAREHOUSE: Warehouse to run queries in.
* SNOWFLAKE\_DATABASE: Database to query.
* SNOWFLAKE\_SCHEMA: Schema within the database.
* SNOWFLAKE\_ROLE: Role assigned to the user in Snowflake.

### Changes in the blueprint\_databricks.py File

The connection to Snowflake instead of Databricks will require minimal changes in this file. You would rename the class instantiation from DatabricksConnector to SnowflakeConnector, and the rest of the logic (query handling, JSON formatting, etc.) remains mostly unchanged.

#### Modify the Connector Initialization:

# Call Snowflake query generator

snowflake\_instance = SnowflakeConnector(input\_text=sql\_query\_text)

response = snowflake\_instance.get\_snowflake\_json()

### Considerations

* **Environment Variables**: Ensure all the necessary environment variables for Snowflake connection (e.g., SNOWFLAKE\_ACCOUNT, SNOWFLAKE\_USER, etc.) are set up in the environment configuration or .env file.
* **Error Handling**: Snowflake’s error handling might differ slightly, so it is important to ensure that any exceptions raised are captured appropriately and returned in a structured format.
* **Query Adaptation**: The SQL queries themselves should be compatible between Databricks and Snowflake, but some query optimizations or adjustments may be needed based on the database's specific SQL dialect.

By making these changes, the project will connect to Snowflake instead of Databricks, enabling interaction with Snowflake's data warehouse while keeping the rest of the application logic mostly intact.