**Lab Exercise 11- Simple Flow Using PostgreSQL Database in Metaflow**

**Lab Exercise: Simple Flow Using PostgreSQL Database in Metaflow**

In this lab exercise, you will create a simple data pipeline using Metaflow that interacts with a PostgreSQL database. You will perform the following tasks:

* Connect to a PostgreSQL database.
* Perform a basic query to read data from a table.
* Process the data in the flow.
* Save the results.

**Prerequisites**

1. **PostgreSQL Database**: Ensure that you have access to a PostgreSQL database and a sample table with some data.
2. **PostgreSQL Connector**: Install the required Python package to connect to PostgreSQL.

To install the PostgreSQL connector (psycopg2), run:

pip install psycopg2

**Step 1: Create a PostgreSQL Table and Populate Data**

Before writing the Metaflow flow, ensure you have a table in your PostgreSQL database. Here's an example of creating a table and inserting some sample data.

CREATE DATABASE metaflow\_db;

\c metaflow\_db;

CREATE TABLE employees (

id SERIAL PRIMARY KEY,

name VARCHAR(50),

position VARCHAR(50),

salary NUMERIC(10, 2)

);

INSERT INTO employees (name, position, salary)

VALUES

('Alice', 'Data Scientist', 120000),

('Bob', 'Software Engineer', 100000),

('Charlie', 'DevOps Engineer', 110000);

**Step 2: Create the Metaflow Flow**

Now, let’s create a simple flow in Metaflow that connects to the PostgreSQL database, retrieves the data, and processes it.

Create a file called postgres\_flow.py with the following code:

from metaflow import FlowSpec, step

import psycopg2

class PostgresFlow(FlowSpec):

@step

def start(self):

"""Connect to PostgreSQL database and retrieve data from the employees table."""

print("Connecting to PostgreSQL database...")

# PostgreSQL connection configuration

self.connection\_config = {

'host': 'your\_postgres\_host', # Replace with your PostgreSQL host

'user': 'your\_postgres\_user', # Replace with your PostgreSQL user

'password': 'your\_postgres\_password', # Replace with your PostgreSQL password

'database': 'metaflow\_db' # Name of the database

}

self.next(self.query\_db)

@step

def query\_db(self):

"""Execute a query to fetch data from the employees table."""

print("Executing query to fetch data from employees table...")

# Establish a connection to the database

conn = psycopg2.connect(\*\*self.connection\_config)

cursor = conn.cursor()

# Execute a simple SELECT query

cursor.execute("SELECT name, position, salary FROM employees")

# Fetch all rows

self.employees = cursor.fetchall()

# Close the connection

cursor.close()

conn.close()

print(f"Retrieved {len(self.employees)} rows from the database.")

self.next(self.process\_data)

@step

def process\_data(self):

"""Process the data (e.g., calculate average salary)."""

print("Processing data...")

total\_salary = sum([emp[2] for emp in self.employees])

self.average\_salary = total\_salary / len(self.employees)

print(f"Average salary: {self.average\_salary}")

self.next(self.end)

@step

def end(self):

"""Final step: Output the processed data."""

print("Flow completed.")

print(f"The average salary of employees is: {self.average\_salary}")

if \_\_name\_\_ == "\_\_main\_\_":

PostgresFlow()

**Step 3: Run the Flow**

To run the flow, execute the following command:

python postgres\_flow.py run

**Step 4: Explanation of Steps**

1. **start step**: This step establishes the configuration required to connect to the PostgreSQL database. It defines the connection parameters such as the host, user, password, and database name.
2. **query\_db step**: This step connects to the PostgreSQL database and executes a SELECT query to fetch the employee data from the employees table.
3. **process\_data step**: The flow processes the fetched data by calculating the average salary of employees.
4. **end step**: This step outputs the final result, i.e., the average salary, and completes the flow.

**Conclusion**

This lab exercise demonstrated how to build a simple Metaflow pipeline that interacts with a PostgreSQL database. You learned how to connect to the database, fetch data, process it, and use Metaflow’s step-based structure to manage each phase of the workflow. This basic example can be expanded to build more complex data processing pipelines.