**Lab Exercise 9- Simple Flow using MySQL Database in Metaflow**

In this lab exercise, you will create a simple data flow in Metaflow that interacts with a MySQL database. You will perform the following tasks:

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

**Prerequisites**

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

To install the MySQL connector, run:

pip install mysql-connector-python

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

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

CREATE DATABASE metaflow\_db;

USE metaflow\_db;

CREATE TABLE employees (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(50),

position VARCHAR(50),

salary DECIMAL(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 this database, retrieves the data, and processes it.

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

from metaflow import FlowSpec, step

import mysql.connector

class MySQLFlow(FlowSpec):

@step

def start(self):

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

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

# MySQL connection configuration

self.connection\_config = {

'host': 'your\_mysql\_host', # Replace with your MySQL host

'user': 'your\_mysql\_user', # Replace with your MySQL user

'password': 'your\_mysql\_password', # Replace with your MySQL 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 = mysql.connector.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\_\_":

MySQLFlow()

**Step 3: Run the Flow**

To run the flow, execute the following command:

python mysql\_flow.py run

**Step 4: Explanation of Steps**

1. **start step**: This step establishes the configuration required to connect to the MySQL database. It defines the connection parameters such as the host, user, password, and database name.
2. **query\_db step**: This step connects to the MySQL 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**: In this step, the flow outputs the final processed result (i.e., the average salary).

**Step 5: Modify Database Configuration**

Make sure to replace the placeholder values in the self.connection\_config dictionary (i.e., your\_mysql\_host, your\_mysql\_user, your\_mysql\_password) with the actual details of your MySQL instance.

**Step 6: Extend the Exercise**

To extend the exercise, you can:

* Add more complex queries (e.g., filtering data, joining tables).
* Insert new data into the database using Metaflow.
* Handle error scenarios (e.g., connection issues) using Metaflow's @retry decorator.

**Conclusion**

This lab exercise demonstrated how to build a Metaflow flow that interacts with a MySQL database. You learned how to connect to the database, fetch data, process it, and use Metaflow's step-based flow structure to manage each phase of the workflow. You can expand this flow to suit more complex data processing or database interactions.