**Lab Exercise 12- Complex Flow for E-commerce Data Transformation in PostgreSQL in Metaflow**

**Prerequisites**

1. **PostgreSQL Installed**: Ensure PostgreSQL is installed and running.
2. **Metaflow Installed**: Make sure Metaflow is installed in your environment.
3. **Python Libraries**: Install the necessary libraries:

pip install psycopg2-binary

**Step 1: Create the PostgreSQL Database and Tables**

1. **Connect to PostgreSQL**: Open your terminal and switch to the postgres user:

sudo -i -u postgres

1. **Create a Database**: Launch the PostgreSQL shell:

psql

Create a new database for the e-commerce app:

CREATE DATABASE ecommerce\_db;

1. **Create Tables**: Switch to the new database and create the necessary tables:

\c ecommerce\_db

CREATE TABLE products (

product\_id SERIAL PRIMARY KEY,

product\_name VARCHAR(255),

price DECIMAL

);

CREATE TABLE orders (

order\_id SERIAL PRIMARY KEY,

customer\_id INT,

order\_date DATE

);

CREATE TABLE order\_details (

order\_detail\_id SERIAL PRIMARY KEY,

order\_id INT REFERENCES orders(order\_id),

product\_id INT REFERENCES products(product\_id),

quantity INT

);

-- Insert sample data into products

INSERT INTO products (product\_name, price) VALUES

('Laptop', 1000.00),

('Smartphone', 500.00),

('Tablet', 300.00);

-- Insert sample data into orders

INSERT INTO orders (customer\_id, order\_date) VALUES

(1, '2024-09-22'),

(2, '2024-09-23');

-- Insert sample data into order\_details

INSERT INTO order\_details (order\_id, product\_id, quantity) VALUES

(1, 1, 1), -- 1 Laptop

(1, 2, 2), -- 2 Smartphones

(2, 3, 3); -- 3 Tablets

1. **Exit PostgreSQL**:

\q

**Step 2: Create the Metaflow Script**

Create a file named ecommerce\_data\_transformation\_flow.py with the following code:

from metaflow import FlowSpec, step

import psycopg2

class EcommerceDataTransformationFlow(FlowSpec):

@step

def start(self):

print("Starting the e-commerce data transformation flow.")

self.next(self.fetch\_data)

@step

def fetch\_data(self):

"""Fetch data from PostgreSQL."""

self.connection\_config = {

'host': 'localhost', # Replace with your PostgreSQL host

'user': 'postgres', # PostgreSQL user

'password': '', # PostgreSQL password

'database': 'ecommerce\_db'

}

# Connect to PostgreSQL

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

self.cursor = self.conn.cursor()

# Fetch products and order details

self.cursor.execute("""

SELECT od.order\_id, p.product\_name, od.quantity, p.price

FROM order\_details od

JOIN products p ON od.product\_id = p.product\_id;

""")

self.raw\_data = self.cursor.fetchall()

print("Fetched data:", self.raw\_data)

self.next(self.transform\_data)

@step

def transform\_data(self):

"""Transform the fetched data."""

self.transformed\_data = []

for order\_id, product\_name, quantity, price in self.raw\_data:

total\_price = quantity \* price

self.transformed\_data.append({

'order\_id': order\_id,

'product\_name': product\_name,

'quantity': quantity,

'total\_price': total\_price

})

print("Transformed data:", self.transformed\_data)

self.next(self.store\_data)

@step

def store\_data(self):

"""Store transformed data back to PostgreSQL."""

self.cursor.execute("""

CREATE TABLE IF NOT EXISTS transformed\_orders (

order\_id INT,

product\_name VARCHAR(255),

quantity INT,

total\_price DECIMAL

);

""")

for item in self.transformed\_data:

self.cursor.execute("""

INSERT INTO transformed\_orders (order\_id, product\_name, quantity, total\_price)

VALUES (%s, %s, %s, %s);

""", (item['order\_id'], item['product\_name'], item['quantity'], item['total\_price']))

self.conn.commit()

print("Stored transformed data in PostgreSQL.")

# Clean up

self.cursor.close()

self.conn.close()

self.next(self.end)

@step

def end(self):

print("E-commerce data transformation flow completed.")

if \_\_name\_\_ == '\_\_main\_\_':

EcommerceDataTransformationFlow()

**Step 3: Run the Metaflow**

In your terminal, run the flow:

python ecommerce\_data\_transformation\_flow.py run

**Step 4: Verify Results in PostgreSQL**

1. **Connect to PostgreSQL**:

sudo -i -u postgres

psql

1. **Check the Transformed Data**: Switch to the ecommerce\_db database and check the transformed\_orders table:

\c ecommerce\_db

SELECT \* FROM transformed\_orders;

1. **Exit PostgreSQL**:

\q