**/\*TOPIC: Data Warehouse Design for an E-Commerce Site\*/**

**TEAM NAME: INNOVATORS**

**DATE: 17.10.2024**

STEP1: Code for ETL Process in Python

import pandas as pd

import sqlite3

# Step 1: Extract

def extract\_data():

# Sample data representing e-commerce sales

data = {

'order\_id': [1, 2, 3, 4],

'user\_id': [101, 102, 103, 101],

'product\_id': [1001, 1002, 1003, 1001],

'quantity': [1, 2, 1, 1],

'price': [500, 1500, 1200, 500],

'order\_date': ['2024-10-01', '2024-10-02', '2024-10-03', '2024-10-01']

}

df = pd.DataFrame(data)

return df

# Step 2: Transform

def transform\_data(df):

# Example of a simple transformation: Adding total amount column

df['total\_amount'] = df['quantity'] \* df['price']

return df

# Step 3: Load

def load\_data(df):

# Connect to SQLite database (can be replaced with other DBs like MySQL)

conn = sqlite3.connect('ecommerce\_warehouse.db')

# Load the data into the 'orders' table

df.to\_sql('orders', conn, if\_exists='replace', index=False)

conn.close()

print("Data loaded into the warehouse!")

# Running the ETL process

data = extract\_data()

transformed\_data = transform\_data(data)

load\_data(transformed\_data)

STEP2: Schema Creation (SQL)

-- Creating a table for storing user information

CREATE TABLE users (

user\_id INT PRIMARY KEY,

name VARCHAR(100),

age INT,

gender VARCHAR(10),

location VARCHAR(100)

);

-- Creating a table for storing product information

CREATE TABLE products (

product\_id INT PRIMARY KEY,

name VARCHAR(100),

category VARCHAR(50),

price DECIMAL(10, 2)

);

-- Creating a fact table for storing transaction data (fact table)

CREATE TABLE orders (

order\_id INT PRIMARY KEY,

user\_id INT,

product\_id INT,

quantity INT,

total\_amount DECIMAL(10, 2),

order\_date DATE,

FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (product\_id) REFERENCES products(product\_id)

);

STEP3: Inserting Sample Data (SQL):

-- Inserting data into users table

INSERT INTO users (user\_id, name, age, gender, location)

VALUES

(101, 'John Doe', 28, 'Male', 'New York'),

(102, 'Jane Smith', 22, 'Female', 'Los Angeles'),

(103, 'Sam Brown', 35, 'Male', 'Chicago');

-- Inserting data into products table

INSERT INTO products (product\_id, name, category, price)

VALUES

(1001, 'Smartphone', 'Electronics', 500),

(1002, 'Laptop', 'Electronics', 1500),

(1003, 'Headphones', 'Accessories', 1200);

-- Inserting data into orders table

INSERT INTO orders (order\_id, user\_id, product\_id, quantity, total\_amount, order\_date)

VALUES

(1, 101, 1001, 1, 500, '2024-10-01'),

(2, 102, 1002, 2, 3000, '2024-10-02'),

(3, 103, 1003, 1, 1200, '2024-10-03'),

(4, 101, 1001, 1, 500, '2024-10-01');

STEP4: Querying the Data Warehouse (SQL)

* Total Sales by Product

SELECT p.name AS product\_name, SUM(o.total\_amount) AS total\_sales

FROM orders o

JOIN products p ON o.product\_id = p.product\_id

GROUP BY p.name;

* Top Customers by Purchase Amount:

SELECT u.name, SUM(o.total\_amount) AS total\_spent

FROM orders o

JOIN users u ON o.user\_id = u.user\_id

GROUP BY u.name

ORDER BY total\_spent DESC;

* Sales Over Time:

SELECT order\_date, SUM(total\_amount) AS daily\_sales

FROM orders

GROUP BY order\_date;

STEP5: Visualization in Python (Matplotlib):

import pandas as pd

import sqlite3

import matplotlib.pyplot as plt

# Connect to the warehouse database

conn = sqlite3.connect('ecommerce\_warehouse.db')

# Query the data

query = "SELECT order\_date, SUM(total\_amount) as daily\_sales FROM orders GROUP BY order\_date"

df = pd.read\_sql(query, conn)

# Plot the data

plt.figure(figsize=(8,6))

plt.plot(df['order\_date'], df['daily\_sales'], marker='o')

plt.title('Daily Sales Over Time')

plt.xlabel('Date')

plt.ylabel('Sales ($)')

plt.grid(True)

plt.show()

# Close the connection

conn.close()