### **Project 12: Data Pipeline**

#### **Objective:**

Build a complete data pipeline that extracts data from an API or web source, processes and transforms it, loads it into a database, and visualizes the results in a dashboard. This project will utilize ETL (Extract, Transform, Load) processes and visualize the data using Matplotlib.

### **Instructions**

#### **Step 1: Set Up the Environment**

1. Create a new Python file called data\_pipeline.py.

Install the required libraries:  
Copy code  
pip install pandas sqlalchemy matplotlib requests

#### **Step 2: Import Necessary Libraries**

Import the required libraries at the top of your script:  
python  
Copy code  
import pandas as pd

import matplotlib.pyplot as plt

import requests

from sqlalchemy import create\_engine

#### **Step 3: Extract Data from an API**

Define a function called extract\_data() that sends a request to an API (e.g., OpenWeatherMap) and returns the data as a Pandas DataFrame.  
python  
Copy code  
def extract\_data(api\_url):

response = requests.get(api\_url)

if response.status\_code == 200:

data = response.json()

return pd.DataFrame(data)

else:

raise Exception("Failed to fetch data from API.")

For demonstration purposes, use a sample API URL:  
python  
Copy code  
api\_url = "https://jsonplaceholder.typicode.com/posts"

data = extract\_data(api\_url)

print(data.head())

#### **Step 4: Transform the Data**

Define a function called transform\_data(df) that processes the DataFrame and performs data cleaning, formatting, and other transformations.  
python  
Copy code  
def transform\_data(df):

# Example transformation: Select specific columns and add new ones

df = df[["userId", "id", "title"]]

df["title\_length"] = df["title"].apply(len)

return df

Apply the transformation:  
python  
Copy code  
transformed\_data = transform\_data(data)

print(transformed\_data.head())

#### **Step 5: Load Data into a Database**

Define a function called load\_data\_to\_db(df, db\_url, table\_name) that connects to a SQL database and inserts the DataFrame data into a table.  
python  
Copy code  
def load\_data\_to\_db(df, db\_url, table\_name):

engine = create\_engine(db\_url)

df.to\_sql(table\_name, con=engine, if\_exists='replace', index=False)

print(f"Data loaded successfully into the '{table\_name}' table.")

Use an SQLite database for simplicity:  
python  
Copy code  
db\_url = "sqlite:///data\_pipeline.db"

load\_data\_to\_db(transformed\_data, db\_url, "posts")

#### **Step 6: Create Visualizations**

Define a function called visualize\_data(df) that creates visualizations using Matplotlib.  
python  
Copy code  
def visualize\_data(df):

plt.figure(figsize=(10, 6))

plt.hist(df["title\_length"], bins=20, color='skyblue')

plt.title("Distribution of Title Lengths")

plt.xlabel("Title Length")

plt.ylabel("Frequency")

plt.show()

Call the function to visualize the transformed data:  
python  
Copy code  
visualize\_data(transformed\_data)

#### **Step 7: Create the Main Function**

Define a main() function that integrates all the ETL processes and runs the pipeline:  
python  
Copy code  
def main():

api\_url = "https://jsonplaceholder.typicode.com/posts"

db\_url = "sqlite:///data\_pipeline.db"

# Run the ETL pipeline

data = extract\_data(api\_url)

transformed\_data = transform\_data(data)

load\_data\_to\_db(transformed\_data, db\_url, "posts")

visualize\_data(transformed\_data)

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

main()

#### **Step 8: Run and Test the Data Pipeline**

1. Run the script to ensure it correctly extracts, transforms, loads, and visualizes the data.
2. Check the SQLite database to confirm the data has been inserted successfully.