Debre Birhan university
College of computing
Software Enginering department
Big Data analytic and business intelligence

Indivisual Assignment

Name **ID**

1. Abreham Tarekegn DBU1600396

Submit to: derbew

ETL Pipeline Code Documentation

Step 1: Data Collection (Extract)

Code Section for Extracting Data from Source (e-commerce CSV)
Python

```
import pandas as pd
# Define the path to the CSV file
file_path = "cleaned_amazon_data.csv"
# Load the dataset into a pandas DataFrametry:
    df = pd.read_csv(file_path)  # Reading CSV data into DataFrame
    print("Dataset loaded successfully!")except Exception as e:
    print(f"Error loading dataset: {e}")
```

Explanation:

Purpose: The first step of the ETL pipeline is to Extract data from the source. In this case, we are reading a CSV file containing e-commerce product information.
Process:
The data is loaded into a pandas DataFrame for easier manipulation.
Error handling is implemented to catch any issues related to file loading.

Step 2: Data Transformation (Transform)

Code Section for Data Cleaning and Feature Engineering
python

```
from scipy import stats
# Data Cleaning# Remove rows with missing critical values (e.g.,
title, price)
df = df.dropna(subset=['title', 'price'])
# Fill missing 'stars' values with the mean of the column
df['stars'] = df['stars'].fillna(df['stars'].mean())
# Remove duplicate rows based on 'asin' (Amazon Standard
Identification Number)
df = df.drop_duplicates(subset=['asin'])
# Convert 'price' column to numeric (handling errors by coercing
invalid values)
df['price'] = pd.to_numeric(df['price'], errors='coerce')
# Feature Engineering# Define price bins
bins = [0, 50, 200, 1000, float('inf')]
labels = ['Low', 'Medium', 'High', 'Premium']
df['price_category'] = pd.cut(df['price'], bins=bins, labels=labels)
# Handle Outliers (Using Z-score to identify outliers in 'price')
z_scores = stats.zscore(df['price'].dropna())
df = df[(z_scores < 3) & (z_scores > -3)]  # Remove outliers beyond 3
standard deviations
```

Explanation:

Purpose: This step transforms the raw data into a cleaner, more structured format that is ready for analysis and storage.
Process:
Handling Missing Data: We drop rows with missing critical columns like title and price. For other columns like stars, missing values are filled with the mean value.
Removing Duplicates: Duplicate rows based on asin are removed to ensure each product is unique.

Feature Engineering: A new column price_category is created to categorize products based on their price.
Outlier Removal: Outliers in the price column are detected using Z-scores and removed.

Step 3: Data Loading (Load)

Code Section for Loading Data into PostgreSQL
Python

```
import psycopg2from sqlalchemy import create_engine
# PostgreSQL connection setup
engine =
create_engine('postgresql://username:password@localhost:5432/mydataba
se')
# Load DataFrame to PostgreSQL
df.to_sql('ecommerce_products', engine, index=False,
if_exists='replace')
print("Data successfully loaded into PostgreSQL.")
```

Explanation:
Purpose: The final step in the ETL pipeline is to Load the cleaned and transformed data into a PostgreSQL database.
Process:
We establish a connection to PostgreSQL using SQLAlchemy and psycopg2.
The cleaned DataFrame df is uploaded into the PostgreSQL table ecommerce_products. If the table already exists, it is replaced.

Step 4: Data Visualization and Reporting

Code Section for Generating Visualizations
Python

```
import matplotlib.pyplot as plt
# Plotting price distribution
plt.figure(figsize=(8, 6))
plt.hist(df['price'], bins=30, color='blue', alpha=0.7)
plt.title("Price Distribution")
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.show()
```

Explanation:
Purpose: Visualizing key metrics to understand the data better and generate business insights.
Process:
A histogram is plotted to visualize the distribution of product prices, providing insights into price ranges and market segmentation.

Documentation of Findings and Design Choices

Data Schema Design

Table Name: ecommerce_products

Columns:
asin: Product identifier (Primary Key)
title: Product name
imgUrl: Image URL for the product
productURL: URL to the product page

stars: Product rating (float)
reviews: Number of reviews (integer)
price: Price of the product (float)
listPrice: Original price (float)
category_id: Category identifier (integer)
isBestSeller: Boolean value indicating whether the product is a best-seller
boughtInLastMonth: Number of times the product was bought in the last month (integer)
price_category: Categorized price range (Low, Medium, High, Premium)

Data Cleaning Processes

Missing Data Handling:

For critical columns like title and price, rows with missing values are dropped.
For less critical columns like stars, missing values are filled with the mean value.

Duplicate Removal:

Duplicate entries based on asin are removed to avoid repeated data.

Outlier Detection:

Outliers in the price column are identified using Z-scores and removed if they exceed 3 standard deviations.

Analysis Assumptions
Price Categorization: Products are categorized based on their price, which is grouped into four categories: Low, Medium, High, and Premium. This helps segment products based on their affordability and positioning.
Sentiment Analysis: While sentiment analysis was not directly included in this pipeline, the processed data can be used for future sentiment analysis related to product reviews and customer feedback.

Conclusion

This ETL pipeline cleans and transforms raw e-commerce data into a usable format, ready for analysis and storage in PostgreSQL. The cleaned data can be used to generate insights, perform trend analysis, and create interactive dashboards. The code is structured for ease of maintenance, with clear comments and explanations for each step.

Summary of Steps

Clone the repo: git clone https://github.com/abrham-tarekegn/ecommerce-data-pipeline.git

Navigate to the project folder: cd ecommerce-data-pipeline

Install dependencies: pip install -r requirements.txt (optional, if requirements.txt exists)

Run the pipeline: python etl_pipeline.py