DEBRE BERHAN UNIVERSITY

COLLAGE OF COMPUTING

DEPARTMENT OF SOFTWARE ENGINEERING

Fundamentals of Big Data Analytics and BI

NAME	ID No.
Natnael Geleta	DBUR/0754/13

Submitted to: Derbew Felasman
February2025
Debre Berhan, Ethiopia

1.Introduction

This document outlines the process followed to extract, clean, and load a dataset into PostgreSQL. The steps include data extraction, preprocessing (cleaning and transformation), and loading into a PostgreSQL database.

2. Data Extraction

2.1 Source of Data

- The dataset was extracted from https://www.kaggle.com/datasets/zusmani/pakistans-largest-ecommerce-dataset
- The dataset format: CSV
- Tools used: Python, Pandas, SQL

2.2 Extraction Method

- Used Pandas to read data:

```
import pandas as pd

file_path = 'Large Ecommerce Dataset.csv'
# Read the CSV file
df = pd.read_csv(file_path)
# Display the number of rows
print("Number of rows in the CSV file:", len(df))
```

Explanation

The code starts by importing the pandas library. It then defines the file path for the CSV dataset and reads the data into a Pandas DataFrame. Finally, it calculates and prints the number of rows in the dataset using the len() function on the DataFrame.

3. Data Cleaning and Transformation

```
import pandas as pd

# Load the CSV file
```

```
df = pd.read_csv("Large Ecommerce Dataset.csv",
low_memory=False)
# Remove empty rows
df = df.dropna(how="all")
# Remove duplicate rows
df = df.drop_duplicates()
# Remove unwanted columns
df = df.drop(columns=["Unnamed: 21", "Unnamed: 22",
"Unnamed: 23", "Unnamed: 24", "Unnamed:
25","sales_commission_code"])
# Save the cleaned data
df.to_csv("cleaned_data.csv", index=False)
print("Duplicates, empty rows, and unwanted columns
removed. Cleaned data saved as 'cleaned_data.csv'.")
```

Explanation

The code imports the pandas library and reads a CSV file into a DataFrame with low_memory=False to optimize memory usage. It removes empty rows using dropna(how="all") and eliminates duplicate rows with drop_duplicates(). Specific unwanted columns, including unnamed columns and sales_commission_code, are dropped using drop(). Finally, the cleaned dataset is saved as a new CSV file named cleaned_data.csv, and a message is printed confirming the successful cleaning process.

4. Loading Data into PostgreSQL

```
import pandas as pd
from sqlalchemy import create_engine

#  Database connection details
DB_NAME = "BDBIDB"
DB_USER = "postgres"
DB_PASSWORD = "postgres"
DB_HOST = "localhost"  # Change to your server's IP
if remote
DB_PORT = "5432"
#  Load cleaned dataset
file_path = "cleaned_data.csv"
df = pd.read_csv(file_path)
#  Define table name
table_name = "ecommerce_data"
```

```
#  Create a PostgreSQL connection using SQLAlchemy
engine =
create_engine(f'postgresql://{DB_USER}:{DB_PASSWORD}@
{DB_HOST}:{DB_PORT}/{DB_NAME}')
#  Ensure column names match exactly before loading
expected_columns = [
    "item_id", "status", "created_at", "sku", "price",
"qty_ordered", "grand_total",
    "increment_id", "category_name_1",
"discount_amount",
    "payment_method", "Working Date", "BI Status",
"MV", "Year", "Month",
    "Customer Since", "M-Y", "FY", "Customer ID"
]
#  Load dataset into PostgreSQL
try:
    df.to_sql(table_name, engine, if_exists="replace",
index=False)
    print(f" Cleaned data successfully loaded into
PostgreSQL table '{table_name}'.")
except Exception as e:
    print(f" Error loading data: {e}")
```

Explanation

The code starts by importing the necessary libraries, pandas for handling data and sqlalchemy for database connection. It then defines PostgreSQL database credentials, including database name, user, password, host, and port.

Next, it loads a cleaned dataset from a CSV file into a Pandas DataFrame. The table name in which the data will be stored is specified. A connection to the PostgreSQL database is then created using create_engine from SQLAlchemy.

To ensure data consistency, a list of expected column names is defined. Finally, the script attempts to insert the DataFrame into the PostgreSQL table using the to_sql method. If successful, it prints a confirmation message; otherwise, it catches and prints any errors encountered during the process.