DEBRE BERHAN UNIVERSITY

COLLAGE OF COMPUTING

DEPARTMENT OF SOFTWARE ENGINEERING

BIGDATA ANALYSIS AND BUSINESS INTELEGENCE

Prepared by:

Mandefro abebaw DBUR/4116/13

Submitted to Derbew Felasman(MSC)

Submission date 06/06/2017

ETL Pipeline Documentation

1. Data Extraction

Overview

The extraction step involves loading the raw e-commerce dataset from a CSV file into a Pandas DataFrame for further processing. The dataset contains transactional records, including order details, customer information, and sales data.

Code for Data Extraction

```python
# Import the pandas library for data manipulation
import pandas as pd

# Define the file path to the dataset
file_path = "Pakistan Largest Ecommerce Dataset.csv"

# Read the CSV file into a pandas DataFrame
df = pd.read_csv(file_path)

# Print the number of rows in the dataset
print(f"Number of rows: {len(df)}")
```

Explanation of the Extraction Process

1. **Library Import:**
   a. The pandas library is imported to handle data loading and manipulation.
2. **Dataset Loading:**
   a. The dataset is read from a CSV file using pd.read_csv().

   b. The file path should be updated accordingly if the dataset is stored in a different location.
3. **Row Count Display:**
   a. The len(df) function is used to check the total number of rows in the dataset.
   b. This helps ensure that the file has been loaded correctly.
4. **Previewing the Data:**
   a. df.head() is used to display the first few rows of the dataset.
   b. This helps in understanding the structure, column names, and data types before cleaning

2. Data Transformation (Cleaning)

Overview

The transformation (or cleaning) step involves preparing the raw dataset for analysis by handling missing values, removing duplicates, correcting inconsistencies, and standardizing formats. This step ensures that the data is accurate, complete, and structured correctly before loading it into a database.

Code for Data Cleaning

```
import pandas as pd  # Import the pandas library for data manipulation

# Load dataset
file_path = "Pakistan Largest Ecommerce Dataset.csv"  # Define the file path of
the dataset
df = pd.read_csv(file_path)  # Read the CSV file into a pandas DataFrame

def clean_data(df):
    """Cleans the dataset by handling missing values, duplicates, formatting, and
inconsistencies."""

    # Drop Fully Null Columns
    df.dropna(axis=1, how="all", inplace=True)  # Remove columns that have all
values as NaN (empty columns)

    # Drop a specific column "increment_id" as it is unnecessary
```

```python
    df = df.drop(columns=["increment_id"])

    # Handle Missing Values
    df.replace("", pd.NA, inplace=True)  # Convert empty string values to NaN for consistency
    df.fillna(method='ffill', inplace=True)  # Forward-fill missing values based on the previous row

    # Remove Duplicates
    df.drop_duplicates(inplace=True)  # Drop duplicate rows to avoid redundant data

    # Standardize Text Columns
    for col in df.select_dtypes(include=['object']).columns:  # Loop through all text columns
        df[col] = df[col].str.strip().str.lower()  # Remove leading/trailing spaces & convert text to lowercase

    # Fix Common Typos & Inconsistencies
    df.replace({"pakistan ": "pakistan"}, inplace=True)  # Correct typo where "pakistan " has an extra space

    # Convert Data Types
    for col in df.select_dtypes(include=['object']).columns:  # Loop through object-type columns
        try:
            df[col] = pd.to_datetime(df[col], errors='ignore')  # Convert date-like columns if applicable
        except Exception:
            pass  # Ignore errors if conversion is not possible

    for col in df.select_dtypes(include=['number']).columns:  # Loop through numeric-type columns
        df[col] = pd.to_numeric(df[col], errors='coerce')  # Convert to numeric, replacing errors with NaN

    print("\n Cleaning completed!")  # Print a message indicating successful data cleaning
    return df  # Return the cleaned DataFrame

# Apply the cleaning function
df_cleaned = clean_data(df)  # Call the function to clean the dataset
```

```python
# Save cleaned dataset
df_cleaned.to_csv("Cleaned_Ecommerce_Dataset.csv", index=False)  # Save the
cleaned dataset as a new CSV file

print("\n Final dataset saved as 'Cleaned_Ecommerce_Dataset.csv'.")  # Print
confirmation message
print("\n🔍 Final dataset info:")  # Print a separator message before showing
dataset info
print(df_cleaned.info())  # Display summary information about the cleaned dataset
```

Explanation of the Cleaning Process

1. **Dropping Fully Null Columns:**
   a. If a column has all missing values, it is removed using df.dropna(axis=1, how="all") to reduce unnecessary columns.
2. **Dropping Unnecessary Columns:**
   a. The column "increment_id" is removed if it exists since it is not needed for analysis. The errors='ignore' argument ensures no error occurs if the column is missing.
3. **Handling Missing Values:**
   a. Empty string values ("") are replaced with NaN (null values) using df.replace("", pd.NA, inplace=True).
   b. Missing values are forward-filled (ffill), meaning the previous row's value is used for missing data.
4. **Removing Duplicates:**
   a. Duplicate records are removed using df.drop_duplicates(inplace=True).
5. **Standardizing Text Columns:**
   a. All text-based columns are trimmed of extra spaces and converted to lowercase for consistency using df[col].str.strip().str.lower().
6. **Fixing Common Typos & Inconsistencies:**
   a. Specific text inconsistencies, such as "pakistan " (with an extra space), are corrected using df.replace({"pakistan ": "pakistan"}, inplace=True).

7. **Converting Data Types:**
   a. Columns that appear to be date-related are converted to datetime format using pd.to_datetime().
   b. Numeric columns are enforced using pd.to_numeric() to avoid data type inconsistencies.

3. Data Loading

Overview

The data loading step involves transferring the cleaned e-commerce dataset into a **PostgreSQL** database. This step ensures that the structured data is stored efficiently, allowing for further analysis, reporting, and visualization using BI tools like **Power BI**

Code for Data Loading

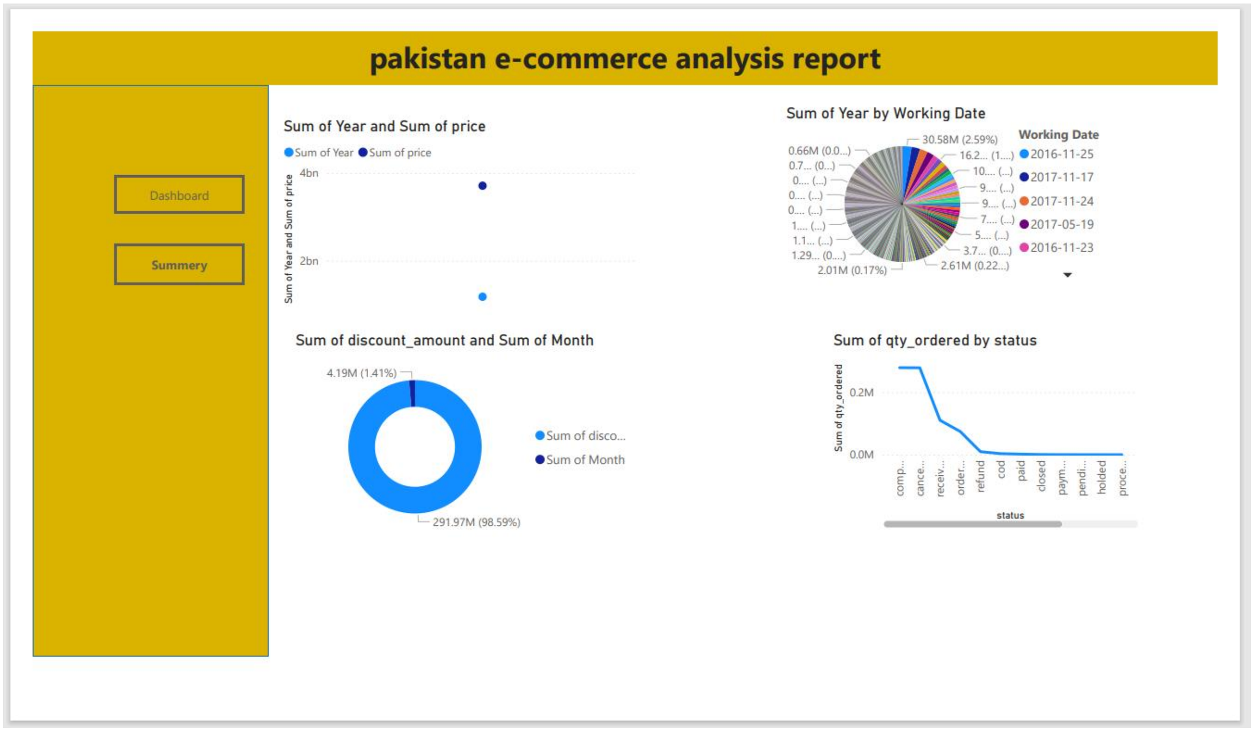
```
# Import necessary libraries
import pandas as pd
from sqlalchemy import create_engine

# Database connection details
DB_NAME = "mandefro"  # Name of the PostgreSQL database
DB_USER = "postgres"  # Database username
DB_PASSWORD = "mande123"  # Database password
DB_HOST = "localhost"  # Host where the database is running
DB_PORT = "5432"  # Default PostgreSQL port

# Load cleaned dataset from a CSV file
file_path = "Cleaned_Ecommerce_Dataset.csv"  # Path to the cleaned dataset
df = pd.read_csv(file_path)  # Read the CSV file into a pandas DataFrame

# Define the table name in PostgreSQL
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
}')
```

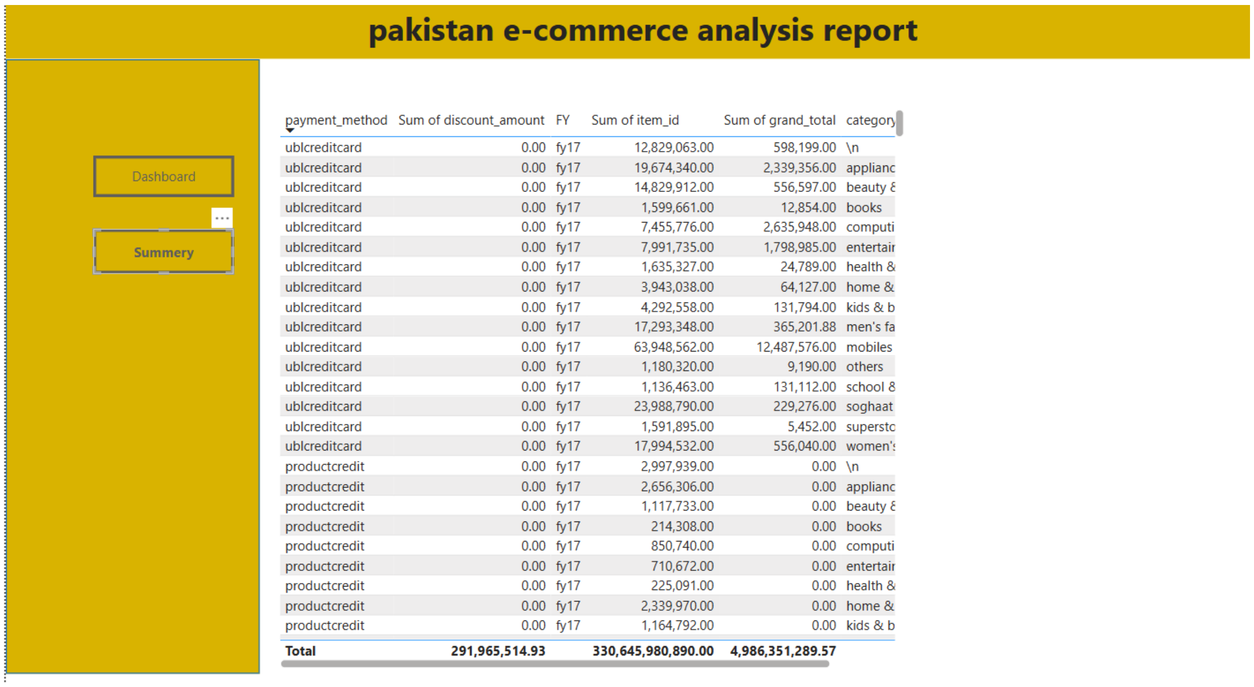
```
# Define the expected columns to ensure they match before loading into the
database
expected_columns = [
    "item_id", "status", "created_at", "sku", "price", "qty_ordered",
"grand_total",
    "category_name_1", "sales_commission_code", "discount_amount",
    "payment_method", "Working Date", "BI Status", "MV", "Year", "Month",
    "Customer Since", "M-Y", "FY", "Customer ID"
]

# Check if any expected columns are missing from the DataFrame
missing_columns = [col for col in expected_columns if col not in df.columns]

# Attempt to load the DataFrame into the PostgreSQL table
try:
    df.to_sql(table_name, engine, if_exists="replace", index=False)  # Load data
into the database, replacing existing table if needed
    print(f"Cleaned data successfully loaded into PostgreSQL table
'{table_name}'.")  # Success message
except Exception as e:
    print(f"❌ Error loading data: {e}")  # Print error message if loading fails
```

4. Data Visualization & Insights

Overview

The final step of the ETL pipeline is data visualization, where we use **Power BI** to create interactive dashboards for analyzing key e-commerce metrics





payment_method	Sum of discount_amount	FY	Sum of item_id	Sum of grand_total	category
ublcreditcard	0.00	fy17	12,829,063.00	598,199.00	\n
ublcreditcard	0.00	fy17	19,674,340.00	2,339,356.00	applianc
ublcreditcard	0.00	fy17	14,829,912.00	556,597.00	beauty &
ublcreditcard	0.00	fy17	1,599,661.00	12,854.00	books
ublcreditcard	0.00	fy17	7,455,776.00	2,635,948.00	computi
ublcreditcard	0.00	fy17	7,991,735.00	1,798,985.00	entertair
ublcreditcard	0.00	fy17	1,635,327.00	24,789.00	health &
ublcreditcard	0.00	fy17	3,943,038.00	64,127.00	home &
ublcreditcard	0.00	fy17	4,292,558.00	131,794.00	kids & b
ublcreditcard	0.00	fy17	17,293,348.00	365,201.88	men's fa
ublcreditcard	0.00	fy17	63,948,562.00	12,487,576.00	mobiles
ublcreditcard	0.00	fy17	1,180,320.00	9,190.00	others
ublcreditcard	0.00	fy17	1,136,463.00	131,112.00	school &
ublcreditcard	0.00	fy17	23,988,790.00	229,276.00	soghaat
ublcreditcard	0.00	fy17	1,591,895.00	5,452.00	supersto
ublcreditcard	0.00	fy17	17,994,532.00	556,040.00	women's
productcredit	0.00	fy17	2,997,939.00	0.00	\n
productcredit	0.00	fy17	2,656,306.00	0.00	applianc
productcredit	0.00	fy17	1,117,733.00	0.00	beauty &
productcredit	0.00	fy17	214,308.00	0.00	books
productcredit	0.00	fy17	850,740.00	0.00	computi
productcredit	0.00	fy17	710,672.00	0.00	entertair
productcredit	0.00	fy17	225,091.00	0.00	health &
productcredit	0.00	fy17	2,339,970.00	0.00	home &
productcredit	0.00	fy17	1,164,792.00	0.00	kids & b
Total	**291,965,514.93**		**330,645,980,890.00**	**4,986,351,289.57**	