ETL ON SALES DATA USING PYTHON

DATA ENGINEERING

JAHANZEB AHMED
BYTEWISE LTD

TASK 7

Problem Statement:

You'll be given a csv dataset file and you'll be extracting data from it through pandas or pyspark. Then there will be the transformation phase and after performing that you'll try to load the final data to a database.

There are some problems with the data, and you need to remove them during the transformation phase to make data useful for everyone:

- An order id should always exist as an integer
- A product id cannot be 0
- We never had a product priced more than 1500 Rs. so any item with amount greater than 1500 Rs is an anomaly and it should be treated as 1500 Rs
- A status of an item can never be null or None, if it is then its an anomaly and item rows to be considered as fake orders and should not be kept in final data
- There must be duplication in final data

The solution is broken down into steps along with code:

Step 1:

Importing the required libraries:

```
import pandas as pd
import psycopg2
from psycopg2 import sql
```

Step 2:

Connecting the database to PostgreSQL:

```
# Define your PostgreSQL connection details
connection_details = {
    'dbname': 'customer_db',
    'user': 'postgres',
    'password': [redacted]
    'host': 'localhost',
    'port': '5432'
}

schema_name = 'myPublic'
```

```
# Connect to PostgreSQL database
conn = psycopg2.connect(**connection_details)
cursor = conn.cursor()

cursor.execute(f"SET search_path TO {schema_name}")
```

Step 3:

Creating table in the existing Database:

```
# Create a table with the predefined schema
create_table_query = """
CREATE TABLE IF NOT EXISTS SALES (
    order_id INTEGER,
    product_id INTEGER,
    amount FLOAT,
    status TEXT
)
"""
cursor.execute(create_table_query)
conn.commit()
```

Step 4 (Extract):

Reading the CSV file

```
# Step 1: Read the CSV file
file_path = 'C:\\Users\\DELL\\OneDrive\\Desktop\\New Database\\dataset.csv'
df = pd.read_csv(file_path)
```

Step 5 (Transformation):

1. Ensuring the order_id is an integer:

```
# Step 2: Clean and transform the data
# Ensure 'order_id' is an integer
df['order_id'] = df['order_id'].astype(int)
```

2. Filtering rows where the product_id is 0

```
# Filter out rows where 'product_id' is 0
df = df[df['product_id'] != 0]
```

3. Limit amount at 1500 Rs:

```
# Cap 'amount' at 1500 Rs.
df['amount'] = df['amount'].apply(lambda x: min(x, 1500))
```

4. Removing anomalies/where status is null:

```
# Remove rows where 'status' is null or None
df = df[df['status'].notnull()]
```

\

5. Removing the duplicate rows:

```
# Remove duplicate rows
df = df.drop_duplicates()
```

Step 6 (Load):

Loading the cleaned data into the PostgreSQL Database:

```
# Step 4: Insert the cleaned data into PostgreSQL
for index, row in df.iterrows():
        try:
            insert_query = sql.SQL("""
            INSERT INTO sales (order_id, product_id, amount, status)
            VALUES (%s, %s, %s, %s)
            """)
            cursor.execute(insert_query, (row['order_id'], row['product_id'], row['amount'], row['status']))
        except Exception as e:
            print(f"Error inserting row {index}: {e}")
            print(f"Row data: {row.to_dict()}")
            conn.rollback()
            continue

    # Commit the transaction if all inserts were successful
conn.commit()

# Commit and close the connection
conn.commit()
cursor.close()
conn.close()

print("Data has been successfully inserted into the PostgreSQL database.")
```

Conclusion:

In this task we have performed Extract, Transform and Load using python, and focused on data extraction, transformation to remove anomalies, data correction and finally loading the dataset into the PostgreSQL database.