Data Dumping from CSV Files to PostgreSQL

In [ ]:
```
import pandas as pd
from sqlalchemy import create_engine
import urllib.parse
import os
from sqlalchemy.sql import text
import time

# PostgreSQL connection details
username = 'postgres'
password = urllib.parse.quote_plus('enterpassword')
host = 'localhost'
port = '5432'
database = 'retail_db'

# connection string
connection_string = f'postgresql://{username}:{password}@{host}:{port}/{database}'
print(connection_string)

# SQLAlchemy engine
engine = create_engine(connection_string)
```

In [2]:
```
try:
    engine.connect()
    print("Connection to PostgreSQL DB successful!")
except Exception as e:
    print(f"Error: {e}")
```

```
Connection to PostgreSQL DB successful!
```

In [4]:
```
def get_sql_type(dtype):
    if pd.api.types.is_integer_dtype(dtype):
        return 'INTEGER'
    elif pd.api.types.is_float_dtype(dtype):
        return 'FLOAT'
    elif pd.api.types.is_bool_dtype(dtype):
        return 'BOOLEAN'
    elif pd.api.types.is_datetime64_any_dtype(dtype):
        return 'TIMESTAMP'
    else:
        return 'TEXT'

# List of CSV files and corresponding table names
csv_files = [
    ('customers.csv', 'customers'),
    ('orders.csv', 'orders'),
    ('sellers.csv', 'sellers'),
    ('products.csv', 'products'),
    ('geolocation.csv', 'geolocation'),
    ('payments.csv', 'payments'),
    ('order_items.csv', 'order_items')
]
```

```
folder_path = 'D:/END TO END RETAIL PROJECT'
```

```
In [7]: for csv_file, table_name in csv_files:
            start_time = time.time()

            file_path = os.path.join(folder_path, csv_file)
            print(f"\nProcessing {csv_file} for table '{table_name}'")

            chunk_size = 100000
            total_rows = 0

            for chunk_num, chunk in enumerate(pd.read_csv(file_path, chunksize=chunk_size)):

                chunk = chunk.where(pd.notnull(chunk), None)

                # Clean column names
                chunk.columns = [col.replace(' ', '_').replace('-', '_').replace('.', '_') for co

                if chunk_num == 0:

                    columns = ', '.join([f'"{col}" {get_sql_type(chunk[col].dtype)}' for col in c
                    create_table_query = f'CREATE TABLE IF NOT EXISTS "{table_name}" ({columns});

                    with engine.begin() as connection:
                        connection.execute(text(create_table_query))
                        print(f"Table '{table_name}' created or already exists.")


                chunk.to_sql(table_name, engine, if_exists='append', index=False, method='multi')

                rows_inserted = len(chunk)
                total_rows += rows_inserted
                print(f"Chunk {chunk_num + 1}: {rows_inserted} rows inserted.")

            elapsed_time = time.time() - start_time
            print(f"Total rows inserted for {table_name}: {total_rows}")
            print(f"Finished processing {csv_file} in {elapsed_time:.2f} seconds.")

        print("CSV files successfully uploaded to the PostgreSQL database!")
```

```
Processing customers.csv for table 'customers'
Table 'customers' created or already exists.
Chunk 1: 99441 rows inserted.
Total rows inserted for customers: 99441
Finished processing customers.csv in 14.01 seconds.

Processing orders.csv for table 'orders'
Table 'orders' created or already exists.
Chunk 1: 99441 rows inserted.
Total rows inserted for orders: 99441
Finished processing orders.csv in 24.12 seconds.

Processing sellers.csv for table 'sellers'
Table 'sellers' created or already exists.
Chunk 1: 3095 rows inserted.
Total rows inserted for sellers: 3095
Finished processing sellers.csv in 0.73 seconds.

Processing products.csv for table 'products'
Table 'products' created or already exists.
Chunk 1: 32951 rows inserted.
Total rows inserted for products: 32951
Finished processing products.csv in 10.04 seconds.

Processing geolocation.csv for table 'geolocation'
Table 'geolocation' created or already exists.
Chunk 1: 100000 rows inserted.
Chunk 2: 100000 rows inserted.
Chunk 3: 100000 rows inserted.
Chunk 4: 100000 rows inserted.
Chunk 5: 100000 rows inserted.
Chunk 6: 100000 rows inserted.
Chunk 7: 100000 rows inserted.
Chunk 8: 100000 rows inserted.
Chunk 9: 100000 rows inserted.
Chunk 10: 100000 rows inserted.
Chunk 11: 163 rows inserted.
Total rows inserted for geolocation: 1000163
Finished processing geolocation.csv in 126.67 seconds.

Processing payments.csv for table 'payments'
Table 'payments' created or already exists.
Chunk 1: 100000 rows inserted.
Chunk 2: 3886 rows inserted.
Total rows inserted for payments: 103886
Finished processing payments.csv in 11.87 seconds.

Processing order_items.csv for table 'order_items'
Table 'order_items' created or already exists.
Chunk 1: 100000 rows inserted.
Chunk 2: 12650 rows inserted.
Total rows inserted for order_items: 112650
Finished processing order_items.csv in 21.03 seconds.
CSV files successfully uploaded to the PostgreSQL database!
```