Middle East Technical University | Department of Computer Engineering

CENG 352

Database Management Systems

Spring 2023-2024
Project 1

Due date: March 24, 2024, Sunday, 23:59

1 Objectives

In this project you are asked to write several SQL queries on a relational database using Olist e-commerce dataset. Your SQL server will be PostgreSQL server and you will work on your local environment. You can see tutorials about how to setup working environment for PostgreSQL. This project consists of 3 parts:

- Creating the database using the given 'csv' files.
- Writing proper SQL queries for certain problems.
- Creating triggers and views.

2 Database Schema

2.1 Tables

Here are the database tables and their columns:

```
product_category
    category_id
    name

products
    product_id
    name
    category_id
    weight
    price
```

```
customers
    customer_id
    name
    surname
    address
    state
    gender

orders
    order_id
    customer_id
    order_time
    shipping_time
    status

shopping_carts
    order_id
    product_id
    amount

refunds
    order_id
    reason
```

3 Tasks

3.1 Task 1 - Creating the Database (13 pts)

Using the given csv files and considering the database schema above, you should create a file named **task1.sql** that contains SQL statements that you've used to create the tables. You need to use **PostgreSQL**.

3.2 Task 2 - SQL Queries (72 pts)

For this task you should be able to write SQL queries for given problems. See given query results to understand the structure. Create an individual SQL file for each subtask. Name the SQL files in this format: **task2_1.sql, task2_2.sql etc.**. Each task is worth 8 points.

1. Find customers who returned their order after the order is completed. For each such customer, list customer id, gender and the total amount of money that must be refunded, in ascending order of customer ids. **You must solve this using "partition by".**

2. Find the customers who have the most expensive order for each month. Return customer_ids, year, month, cart total in ascending order by month.

3. List the first 25% product categories that were in the refunded orders, because of a fraud suspicion. Return category name and fraud count in descending order by fraud count. **You must solve this using "ntile" function.**

4. Find the top 3 profitable days of the week, with the highest average revenue for all weeks. consider non-cancelled, non-refunded orders. e.g, calculate revenue for each Monday, take the average. if it is greater than the other days, then the answer includes Monday. Return the day of the week and average daily revenue in descending order by avg daily revenue.

5. Find which product category pair is ordered together most. List 10 pairs, and the number of orders. Consider non-cancelled, non-refunded orders. Consider 2 different categories in pairs. Return category 1, category 2, and the order count in descending order by count.

6. For each category find the product id that has been ordered most. Return the category, product id, amount and the difference of this amount compared to the previous category in decreasing order of amount. **You must solve this using "lag" function.**

7. Write a single SQL query that computes the statistics present in a cross tabulation over genders and months. The aggregate value reported should be total money spent for each gender-month pair. Each row should contain gender, month name, total money spent. **You must solve this using "cube" function.** Also write a crosstab function with this query.

8. Your aim is to find out which products under which category are most fragile products. List the product category, product names, product rank when ranked by number of refunded orders because of damaged delivery for each product and number of refunded orders because of damaged delivery.

9. **Users & permissions:**

   - Create a user and login to the database with that user.
   - Try to query all cancelled orders.
   - Now give permission to query orders table to the new user.
   - Try to query all cancelled orders again.
   - Write down all your observations and queries you executed during this process. Basic .txt file will be enough for this report. Please be brief for your explanations.

3.3 Task 3 - Triggers and Views (15 pts)

Name the SQL files in this format: **task3_1.sql, task3_2.sql, task3_2.sql**.

1. **Trigger #1:** We want to keep average product prices per product category.

   - Write query to create *category_avg_prices* table, by taking the average price of products in a category.
   - This table should contain *category_id* and *avg_price*.
   - Then write queries to trigger an update on *category_avg_prices* table, when a new product is added to the database.

- You can assume that the product category will exist in the database, while adding the product.

2. **Trigger #2:** Write a trigger to throw an error when a new entry is added to *shopping_carts* with amount less than or equal to 0.

3. **Views:** Create a customer view that allows customers to see only their shopping cart, and orders. They should not have access to refunds, orders, details of other customers (like in a realistic e-commerce site) Which type of view you would choose, standard or materialized? What are the reasons? Explain briefly in a few sentences and put your SQL query in the .sql file.

4 Regulations

1. **Submission:** Submission will be done via ODTUClass. Please remember that **late submission is allowed up to 5 days for the entire semester.** You can distribute these 5 days to any project your want.

   You should put the answer query to each question in a separate .sql file and zip those .sql files with following name:

   ```
   e1234567_project1.zip
       -> task1.sql
       -> task2_1.sql
       ...
       -> task2_9.sql
       -> task3_1.sql
       -> task3_2.sql
       -> task3_3.sql
   ```

   Where you should replace "1234567" with your own student number.

2. **SQL Style:** You should write your queries in readable manner. Write each clause on a separate line, add indentation for clear look. For example:

   This is easier to read

   ```
   select
     *
   from
     orders o
   where
     o.order_id = 'test'
   ```

   than this

   ```
   select * from orders o where o.order_id = 'test'
   ```

You can use online-formatters/beautifiers for better indentation if you want.

3. **Newsgroup:** You must follow the ODTUClass for discussions and possible updates on a daily basis.

5 Tutorial & Guides

- You can download PostgreSQL server from **here**.
- For visualization, **DBeaver** is a nice tool which works for all OS. You can also use it for many other database servers.
- Once you start database server, open DBeaver and click 'New Database Connection' on top left corner. (Figure 1)
- Choose PostgreSQL. (Figure 2)
- Enter credentials and connect to default 'postgres' database. You will create your own database for this project later. (Figure 3)
- Open a new script and execute **"create database <database_name>"** by selecting script and hitting (CTRL + ENTER). (Figure 4)
- Now that you have created another database, connect to newly created database just like before. (Figure 5)
- Open 'Tables' to see the tables. (Figure 6) Since there are no tables you need to create tables using your script. Run queries by selecting parts of script and hitting CTRL + ENTER. (Figure 7)
- You are almost there. Now, you need to import the data to tables.
- To import data, right click on the table name and click 'Import data' (Figure 8)
- Choose CSV type and .csv file that you want to import for selected table. (Figure 9, 10)
- Choose **comma(,)** as delimiter for given csv files. You can observe that in individual csv files.
- That's it. Now you can write queries on tables.
- Note: Don't mind that the figures are old, the import behaviour is exactly the same.

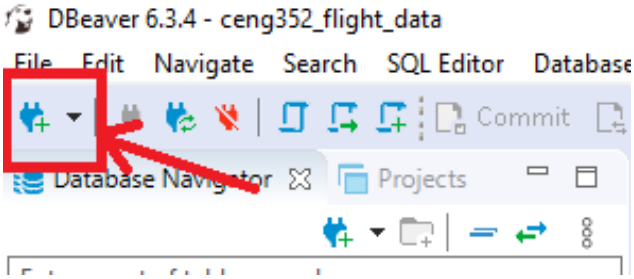

Figure 1: New database connection

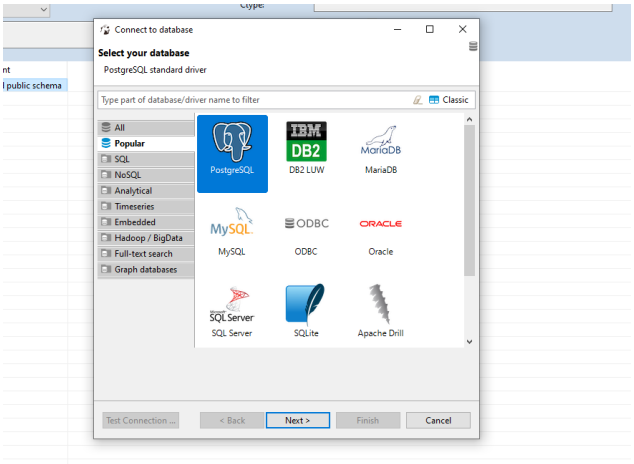

Figure 2: Database options

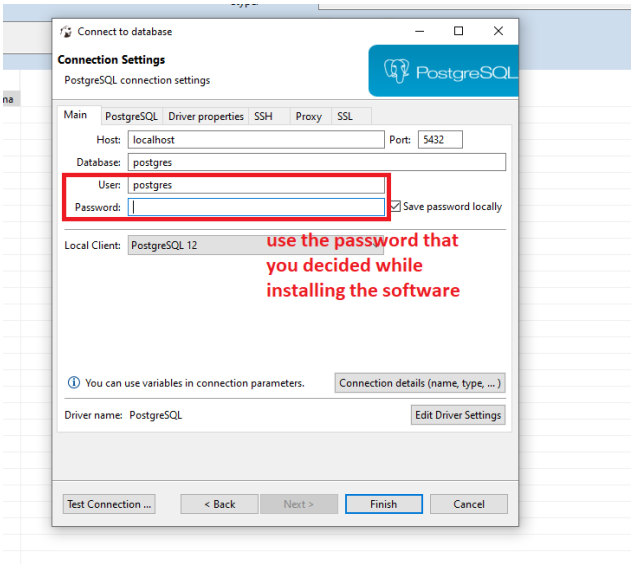

Figure 3: Enter credentials and connect

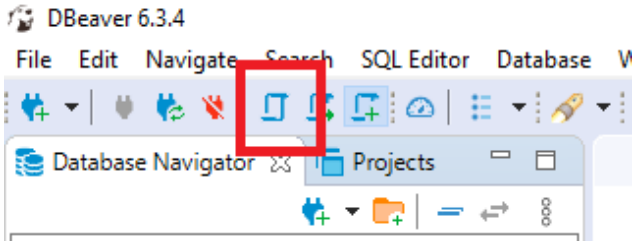

Figure 4: Open new script

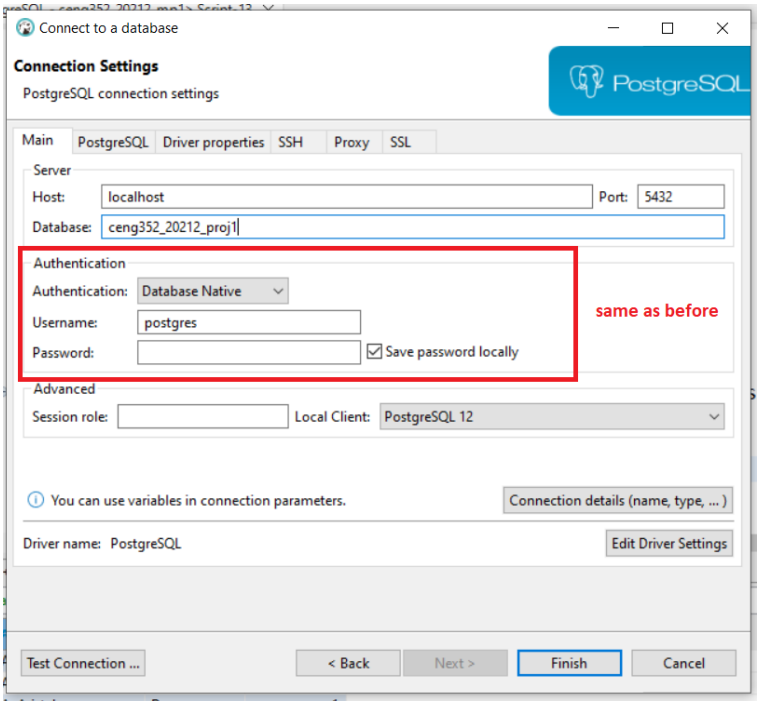

Figure 5: Connect to new database

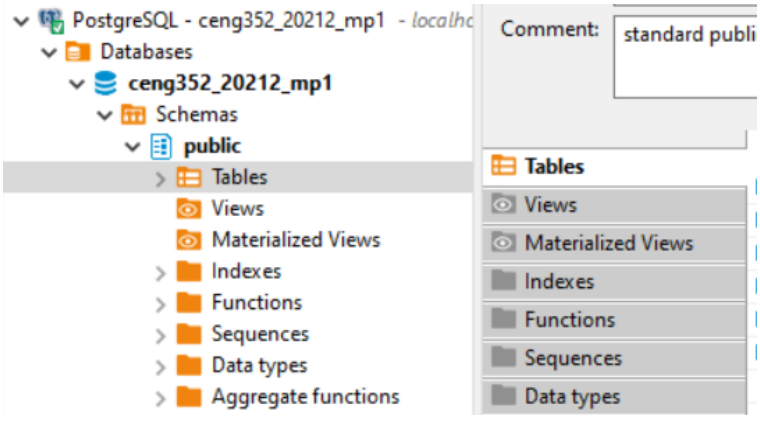

Figure 6: Open tables view

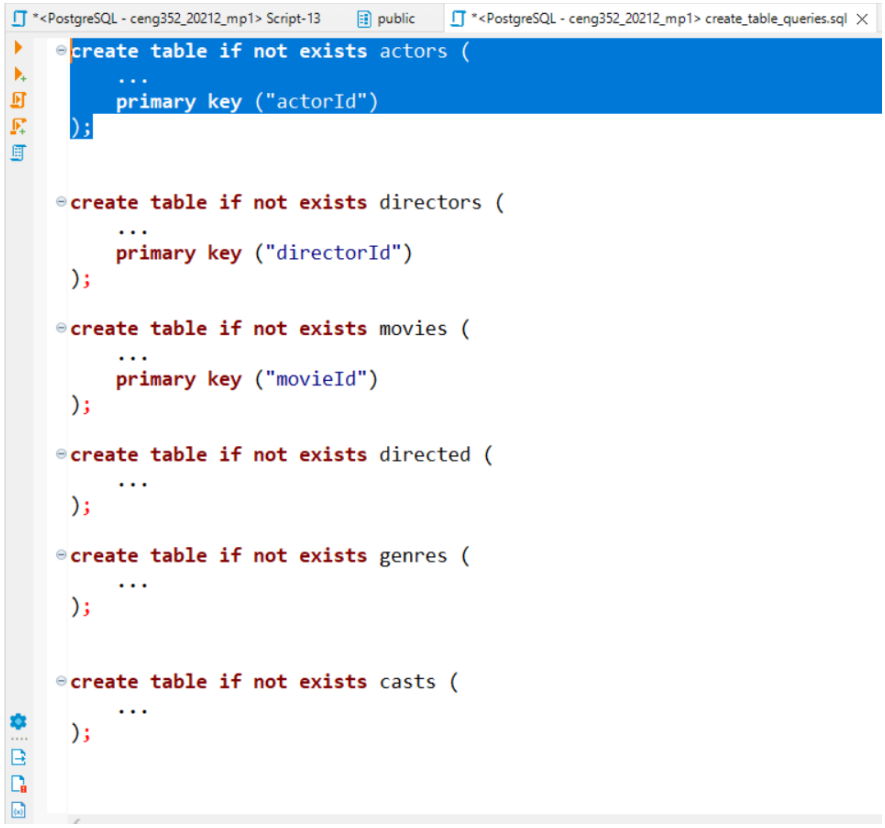

Figure 7: Run create scripts

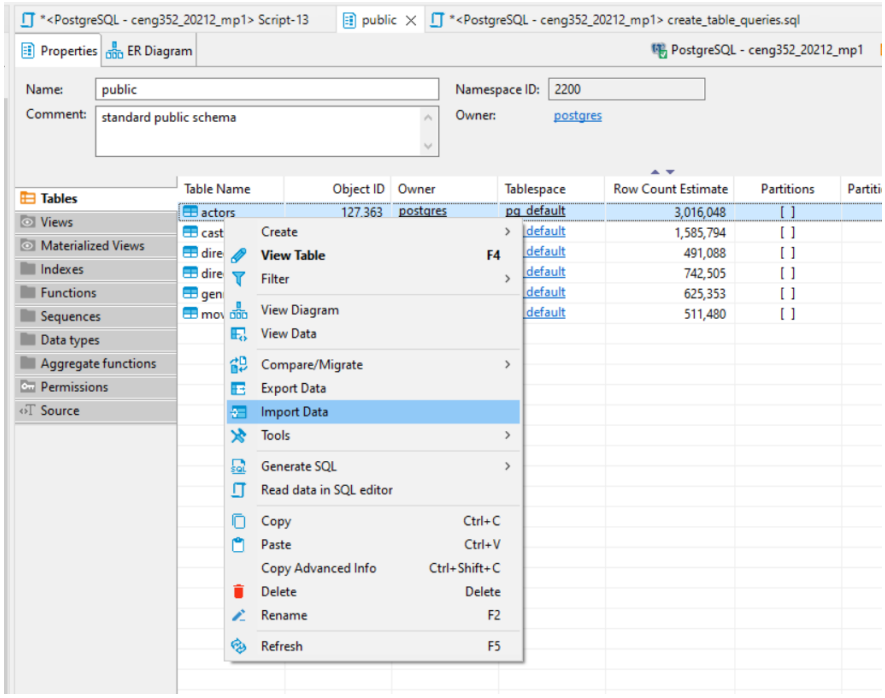

Figure 8: Import data

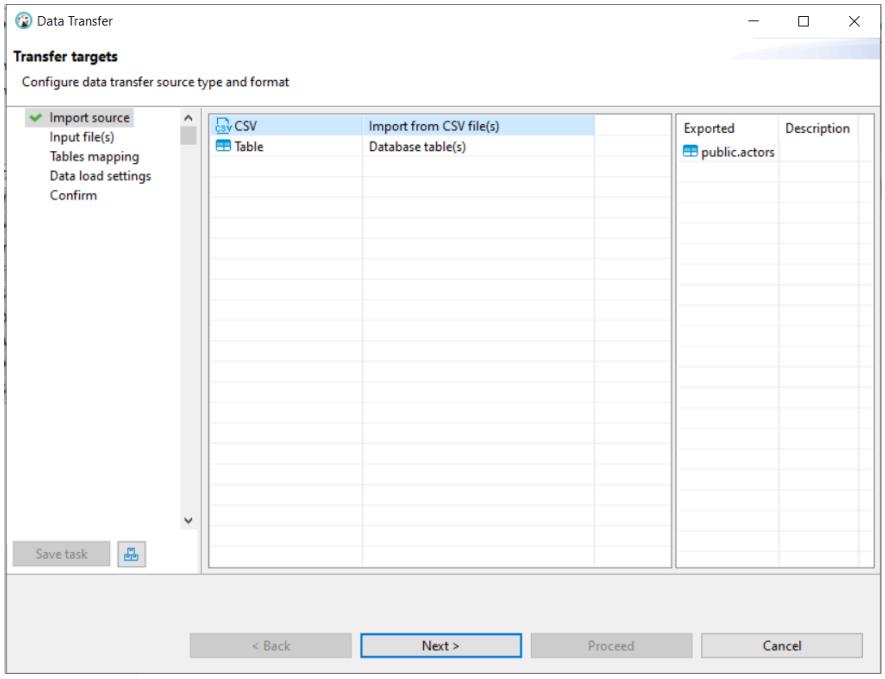

Figure 9: Choose CSV

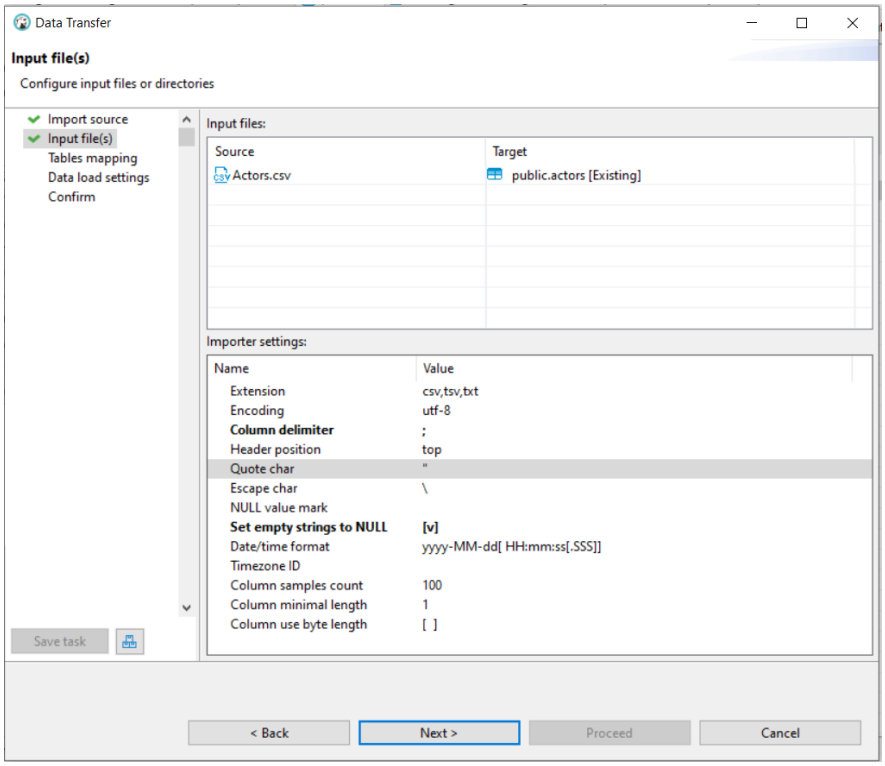

Figure 10: Set delimiter