Middle East Technical University | Department of Computer Engineering

CENG 352

Database Management Systems

Spring 2022-2023
Project 1

Due date: April 7, 2023, Friday, 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
        product_category_name
        product_category_name_english

customers
        customer_id
        customer_unique_id
        customer_zip_code_prefix
        customer_city
        customer_state
```

```
geolocation
        geolocation_zip_code_prefix
        geolocation_lat
        geolocation_lng
        geolocation_city
        geolocation_state

order_payments
        order_id
        payment_sequential
        payment_type
        payment_installments
        payment_value

order_items
        order_id
        order_item_id
        product_id
        seller_id
        shipping_limit_date
        price
        freight_value

orders
        order_id
        customer_id
        order_status
        order_purchase_timestamp
        order_approved_at
        order_delivered_carrier_date
        order_delivered_customer_date
        order_estimated_delivery_date

sellers
        seller_id
        seller_zip_code_prefix
        seller_city
        seller_state


products
        product_id
        product_category_name
        product_name_length
        product_description_length
        product_photos_qty
```

```
        product_weight_g
        product_length_cm
        product_height_cm
        product_width_cm


order_reviews
        review_id
        order_id
        review_score
        review_comment_title
        review_comment_message
        review_creation_date
        review_answer_timestamp
```

2.2 Data details

- All identifier columns have strings with length 32
- All dates and timestamps can be considered as timestamp values
- Product category is for your information, no need to include it in the queries, because the questions in following tasks will mention both English & Portuguese translations for product categories (otherwise you would need to add one more layer to all queries)

3 Tasks

3.1 Task 1 - Creating the Database (15 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 (70 pts)

For this task you should be able to write SQL queries for given problems. Create an individual SQL file for each subtask. Name the SQL files in this format: **task2_query1.sql, task2_query2.sql etc.** Each question is 7 pts.

1. List city of customers who buy products only with voucher. (484 rows)
2. Find health product (beleza_saude) sellers from Curitiba who ships the products quickly. If a seller shippes the orders in at most 2 days on average, the seller works fast. Show results *seller_id* and *avg_duration*, with descending order for *avg_duration*. (5 rows)
3. List product categories of products that are delivered later than expected time for orders from Rio de Janiero in summer 2018 (between June 1st - September 1st). List *product_category_name*

and *late_delivery_count*, with descending order by *late_delivery_count*, ascending order for *product_category_name* (23 rows)

4. List top rated 5 sellers in Sao Paulo for auto (automotivo) products. The sellers must have at least 10 reviews. Show *seller_id*, *avg_rating* and *review_count*, with descending order for *avg_rating* (5 rows)

5. In *order_items* table, there can be multiple entries for an *order_id*, *order_item_id* pair. This indicates that these items are bought together in a single shopping cart. For each month, list the highest shopping cart totals and the customers who ordered the items in the shopping cart. List *customer_id*, *year*, *month*, *max_total_for_month*, with ascending order for *year* & *month* (25 rows)

6. Find best sellers of Q1 - 2018 (between January 1st - April 1st). Best sellers are who sold at least 50 items in each month of Q1 2018. Show *seller_id*, *total_earned* using order item prices in Q1, with descending order for *total_earned* (16 rows)

7. Show top 10 rated electronics (eletronicos) products that has been rated at least 5 times. For each product in top 10 list, show *product_id*, *review_avg*, *review_count*; with descending order for *review_avg*, descending order for *review_count* (10 rows)

8. Write a query which shows the potential product categories for new sellers in Sao Paulo. To do this analysis, find the product categories that are sold everywhere but in Sao Paulo. If there are at least 10 orders on such product category, that means it has a potential for the new sellers, because there are customers who needs such products in Sao Paulo. List *product_category_name*, *past_order_count* for such product categories, with ascending order on *product_category_name*. (1 row)

9. **Cross tabulations:**
   Write a single SQL query that computes the statistics present in a cross tabulation over sellers and months (use *shipping_limit_date*) on *order_items* table. The aggregate reported should report total freight values for each seller-month pair in 2018. In addition to seller-month pairs, the results should contain seller yearly total freight values, and also total freight value of all sellers and all months at the end.

   Each result row should contain *seller_id*, *month* and *total freight* value. (11344 rows)

10. **Window functions:**
    List the *product_id*s, *total_sales* (from *order_item*s table), *product_category_name*s and product's rank when ranked by *total_sales* for each grouping formed by *product_category_name*s. Restrict your attention to those products with a total number of sales more than 10, inclusive. (1899 rows)

3.3 Task 3 - Triggers and Views (15 pts)

Triggers. It is often useful to have the DBMS perform some actions automatically in response to operations on the database. Write necessary triggers to achieve the following functionalities:

1. We want to keep average review score of the products consistent with the user review scores:

- Create a "*user_review_scores*" table with *product_id* and *review_score* columns. Do the required calculation and insert rows to the *user_review_scores* table.
- Make sure that *review_score* column can hold a floating point value.
- Every time a new user review is inserted into *order_reviews* table, the average *review_score* in *user_review_scores* table must be updated. Write a trigger to keep average *review_score* of the products up-to-date in this manner.

2. Let's prevent adding new sellers with zip code '00000'. If there is a seller with zip code = '00000' that is being inserted to sellers table, prevent it from happening with a trigger. **If the new seller's zip code is '00000', then raise "Zip code of the seller CAN NOT be zero!" exception.**

Views. Create a view called "order_product_customer_review" that contains the order id, product id, customer id, review score (112.371 rows). Create your view as a standard view and materialized view. What is the difference between them? How the run time is changed when you switch to using a materialized view? Explain in a couple of sentences.

Write the create table & insert values queries for the *user_review_scores* table, trigger queries, view query inside a file called **task3.sql**. You can explain the difference of the views as a comment in that file.

4 Regulations

1. **Submission:** Submission will be done via ODTUClass. Please remember that **late submission is allowed 5 days for all programming proejcts.** 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_query1.sql
    ...
    -> task2_query10.sql
    -> task3.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.

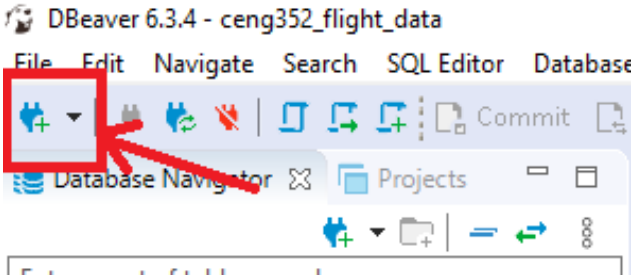

Figure 1: New database connection

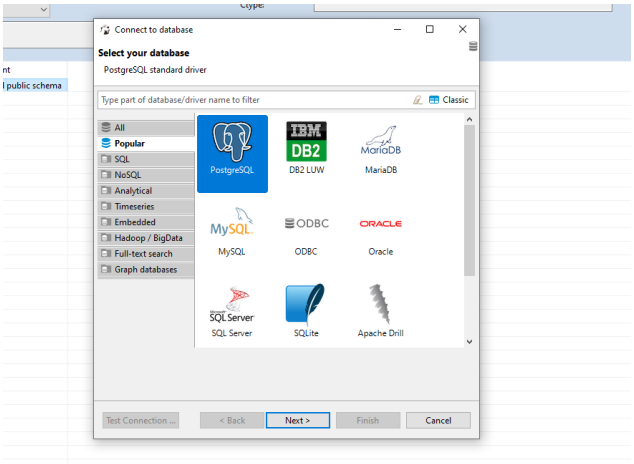

Figure 2: Database options

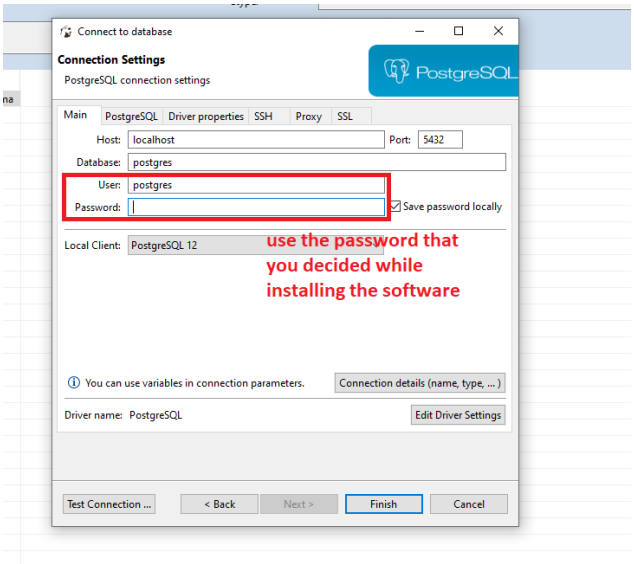

Figure 3: Enter credentials and connect

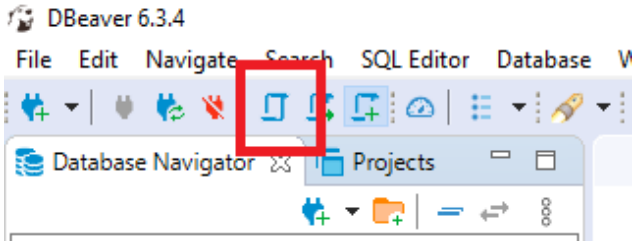

Figure 4: Open new script

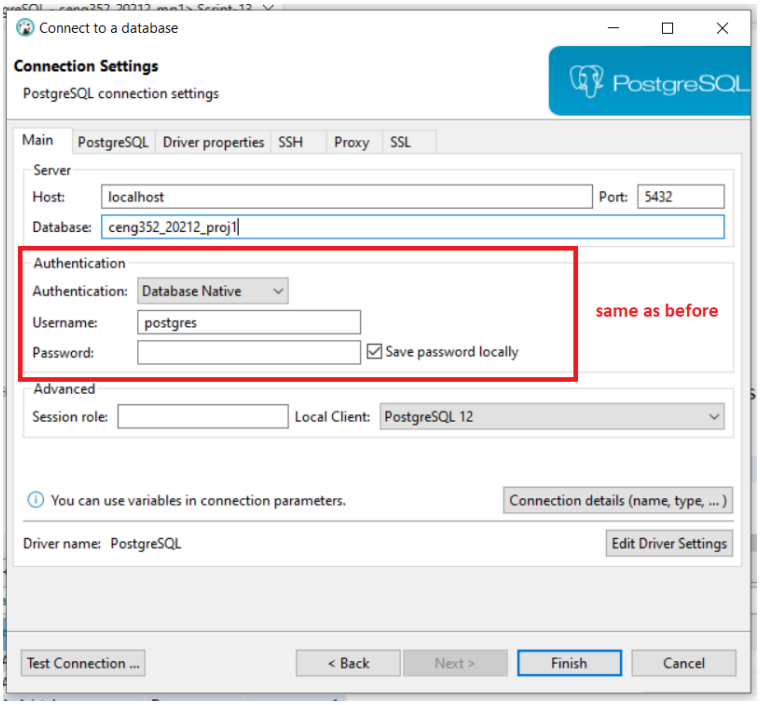

Figure 5: Connect to new database

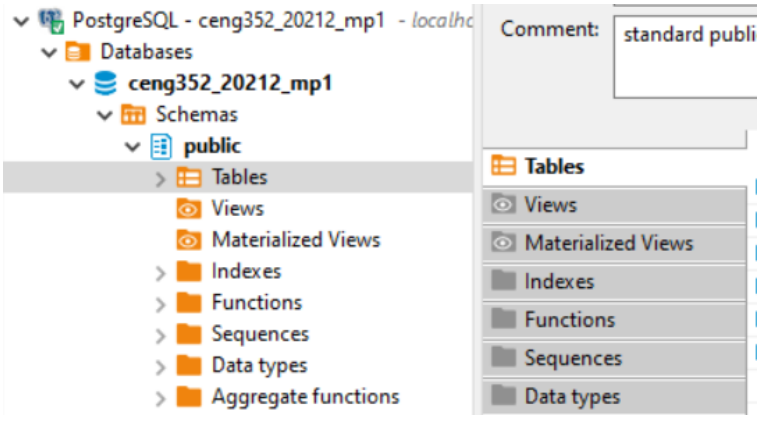

Figure 6: Open tables view

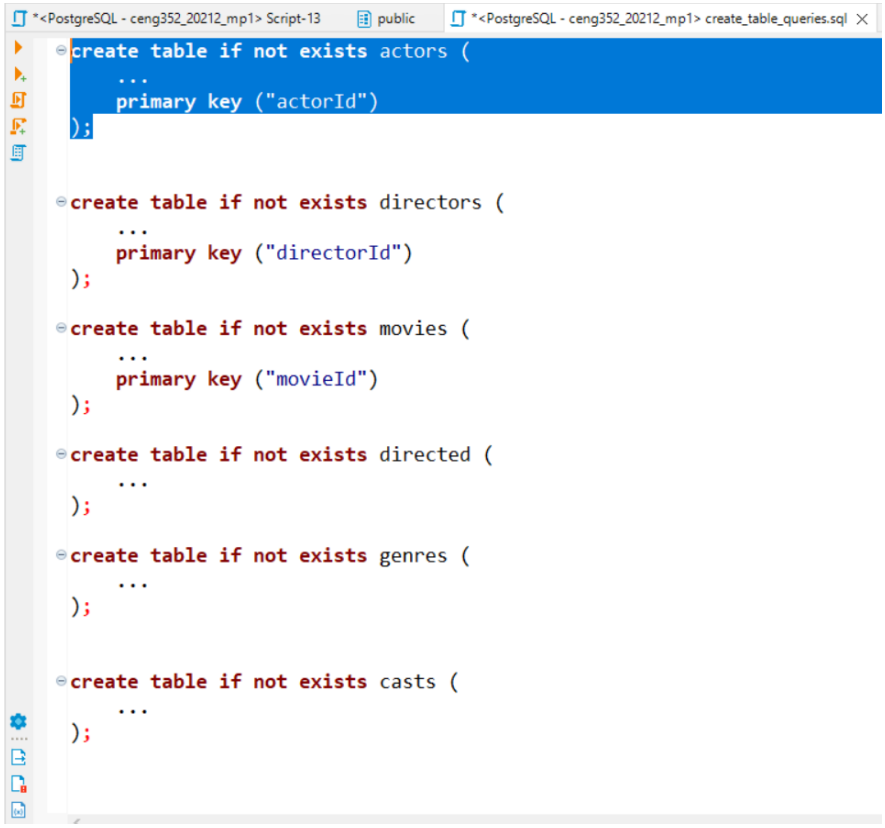

Figure 7: Run create scripts

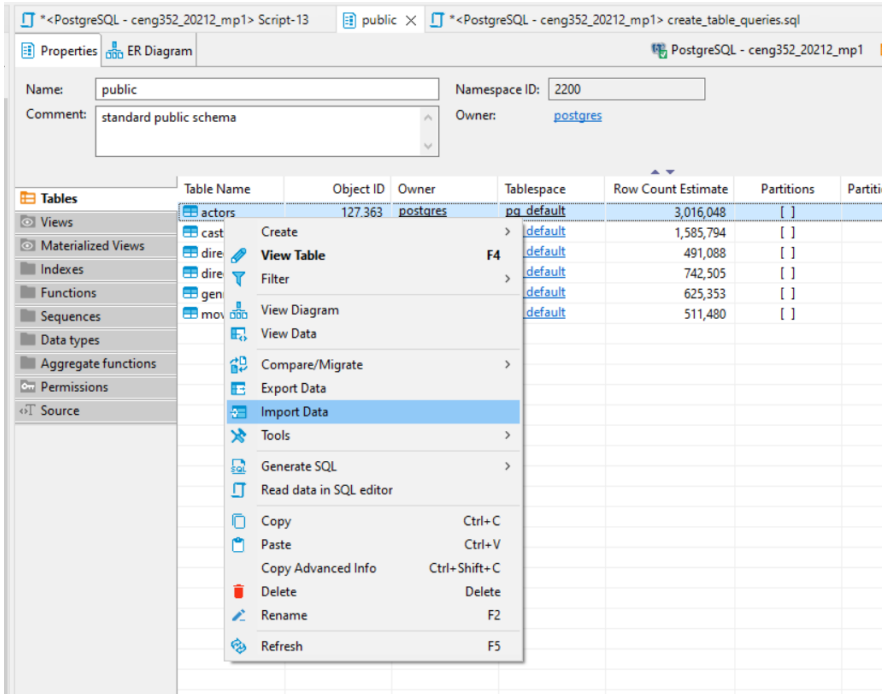

Figure 8: Import data

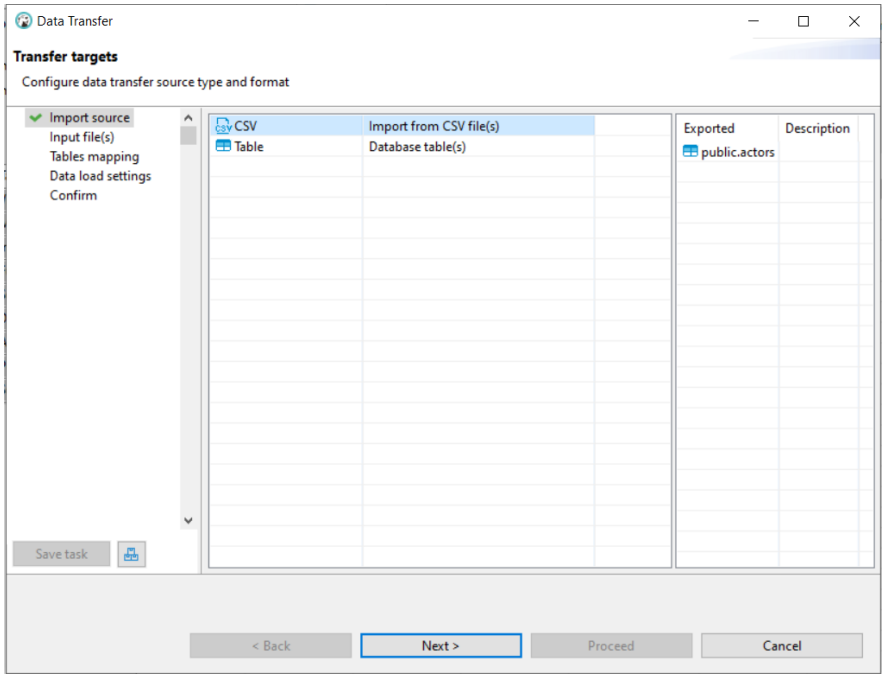

Figure 9: Choose CSV

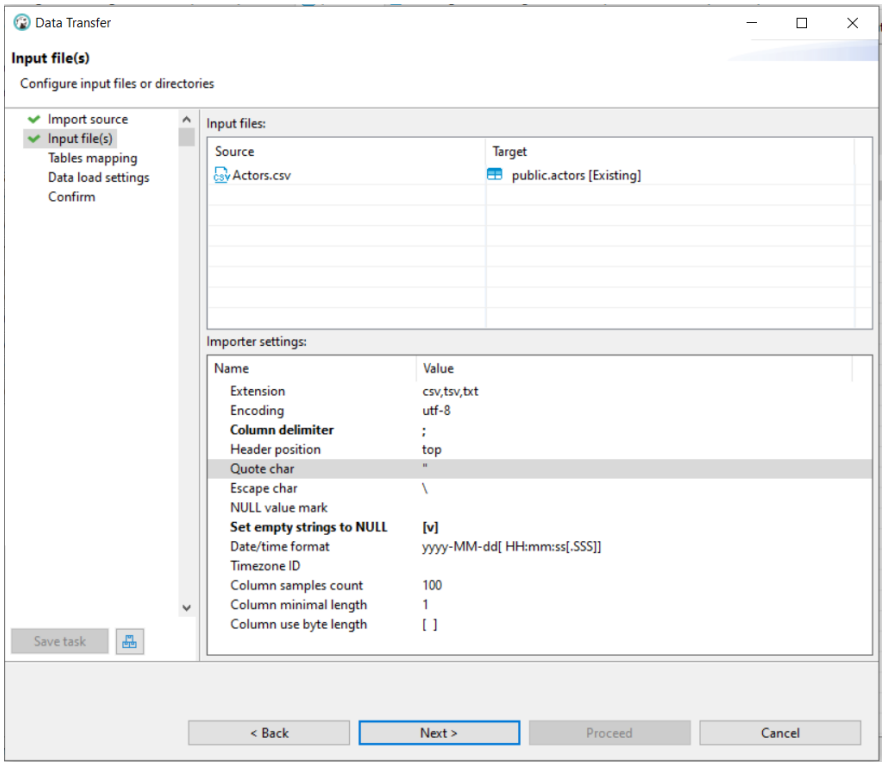

Figure 10: Set delimiter