SCHOOL OF COMPUTING (SOC)

CA2 Specification

DIPLOMA IN INFORMATION TECHNOLOGY

ST0525 Database Systems (DBS)

2024/2025 Semester 1

Objective of Assignment

The objective of this assignment is to apply fundamental database concepts in the development of an application. The student is expected to apply the theoretical knowledge acquired in class to design the database and develop practical features for web application.

Instructions and Guidelines

1. This is an **INDIVIDUAL** assignment and will account for **30%** of the **module grade**.

2. The deadline of this assignment is on **12 July 2024 Friday (6:30 pm)**.

3. You are REQUIRED to do the following for your submission:

   - Deliverable #001 : Code Commits on Github Classroom repository. It will be reviewed for marking.
   - Deliverable #002 : Completed CA2 Individual Report.
   - Deliverable #003 : Zip file of your source code.
   - Deliverable #004 : Zip file of a backup of the final state of your database.
   - Declaration form : A word document for declaration of academic Integrity

   The final submission on BrightSpace is a single zip file of deliverables #002, #003, #004 and the declaration form with the following naming convention:

   ModuleClass-YourStudentID-YourName_CA2.zip
   DIT2B01-2012345-DavidTan_CA2.zip

4. Submission requirements for **deliverable #001: Commits on Github Classroom repository**

   - All code submissions must be made through GitHub Classroom. Follow the provided repository link to create your personal repository and commit your code into it.
   - **Export** the user defined functions and stored procedures in your database as a **sql file**, named as "**functions_&_stored procedures**" and **commit it** as a file in your final commit. Refer to the appendix on how to do this.
   - Submission README: Include a detailed README file in your repository. The README should contain clear instructions on how to run your application, any prerequisites or dependencies, and any additional details that will assist the reviewers in evaluating your project.
   - You are expected to practice version control by committing your code regularly (minimally once a week) as you make progress. Regular and clear commits demonstrate a well-organized development process, help track changes, and make it easier to revert to previous states if needed.
   - Submissions that do not adhere to the version control and submission guidelines would be subjected to a **penalty**.

5. Submission requirements for **deliverable #002: CA2 Individual Report**
   - The template is available in BrightSpace under Assignments > CA2 > CA2 Individual Report.

- The report should include clear documentation of evidence to support each criterion listed in the template.
- It should be submitted as a PDF document using the following file-naming convention
  YourModuleClass-YourStudentID-YourName_CA2Report.pdf
  2012345-DavidTan_CA2Report.pdf

- Submissions that do not adhere to the stipulated naming convention would be subjected to a **penalty**.

6. Submission requirements for **deliverable #003: Source codes of modified application**
   - Should be submitted as a zip document using the following file-naming convention
     YourStudentID-YourName_Code.zip
     2012345-DavidTan_Code.zip

   - The zip file should contain source codes of your application (**without the node_modules folders**)
   - Submissions that do not remove the node_modules folders or do not adhere to the stipulated naming convention would be subjected to a **penalty**.

7. Submission requirements for **deliverable #004: Backup of database**
   - Should be submitted as a zip document using the following file-naming convention
     YourStudentID-YourName_Db.zip
     2012345-DavidTan_Db.zip

   - The zip file should contain the backup SQL file of your database. The instructions on how to backup can be found on BrightSpace > Resources > Guides

8. Warning: **No marks** will be awarded, if the work is copied or you have allowed others to copy your work. **Plagiarism** means passing off as one's own the ideas, works, writings, etc., which belong to another person. In accordance with this definition, you are committing plagiarism if you copy the work of another person and turning it in as your own, even if you would have the permission of that person. Plagiarism is a serious offence, and if you are found to have committed, aided, and/or abetted the offence of plagiarism, disciplinary action will be taken against you. If you are guilty of plagiarism, you may **fail** all modules in the semester, or even be liable for **expulsion**.

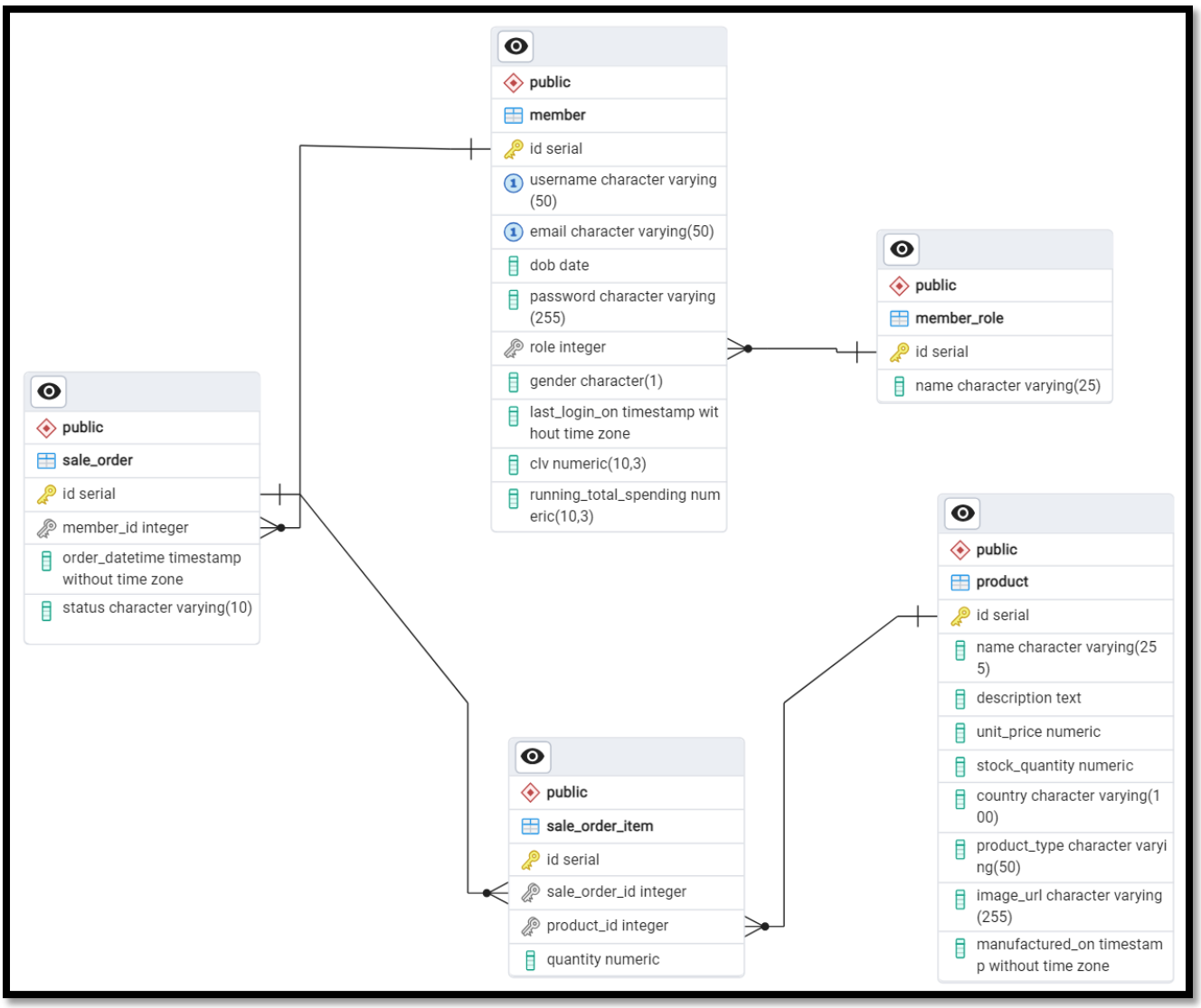
9. Late penalty: **50%** of the marks will be deducted for assignments that are received within **ONE (1)** calendar day after the submission deadline. No marks will be given thereafter. Exceptions to this policy will be given to students with **valid LOA on medical or compassionate grounds**. Students in such cases will need to inform the lecturer as soon as reasonably possible. Students are **not to assume** on their own that their deadline has been extended.

Specifications

In this assignment, you will be implementing the features of an ecommerce website. This includes developing both the application and database functionalities of the features. You will be given the initial setup and the feature requirements.

A. Initial Setup

Refer to the Setup Guide to set up the database and the code base.
The initial setup database comprises 5 tables. The ERD is as shown below.
The tables are already seeded with data.



a) Mandatory Requirements

You are part of a team that has been tasked to develop an ecommerce web application. The application is in its early phase of development, and you have been tasked to add new features and functionalities to the application.

At the current state, the main entities are:

- Member – A registered member of the application.
- Member Role – A member is either an administrator ('ADMIN') or customer ('USER')
- Product – The entity that a member can purchase on the application.
- Sale Order – An order that belongs to a member and comprises the product items that the member had purchased. A sale order status can be one of 'COMPLETED', 'CANCELLED' or 'PACKING'.
- Sale Order Item – The per item detail of an order

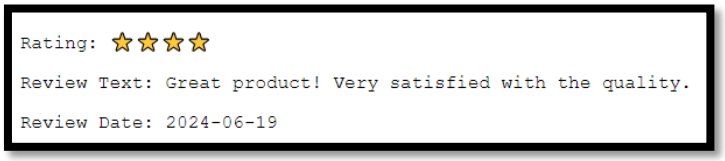
As a mandatory requirement, you are required to implement the feature for members to **manage their reviews** of products.

Take note of the business rules:

- A member can add a review for a product, only for orders that the member has previously for the product.
- A member can order a product many times in different orders.
- A review can only be added for completed (eg: status "COMPLETED") orders. For every completed order, the member can add exactly one review.
- A member can only modify or remove his/her own reviews.

Your tasks:

1. Design and implement the relevant database table(s), including the attributes, constraints and relationships. In your design consideration, you are to ensure a review will be displayed with details as shown.



2. Develop the CRUD capabilities for a member to manage his/her reviews.

   1) Specifically, implement **an appropriate stored procedure or database user defined function for each** of the following so that a member can
      i. Create a new review. Name it as create_review
      ii. Update an existing review of the member. Name it as update_review

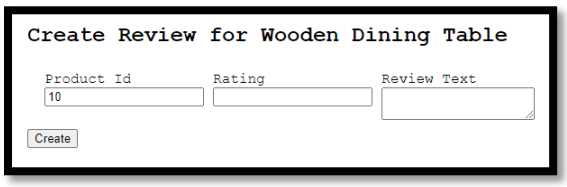
iii. Delete an existing review of the member. Name it as delete_review
iv. Retrieve a single review of the member. Name it as get_review
v. Retrieve all reviews of the member. Name it as get_all_reviews

The above stored procedures or user defined functions should include any relevant error handling. For example, there should be an error flagged if a member attempts to add a review for a product on which he/she has no completed order.

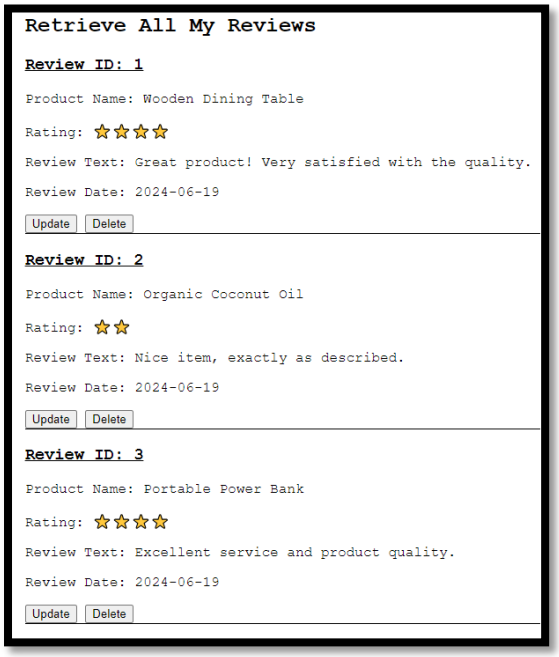
2) Implement the model functions, controller functions and express routes to complete the CRUD end-to-end.

As completed end-to-end features, the application should be able to:

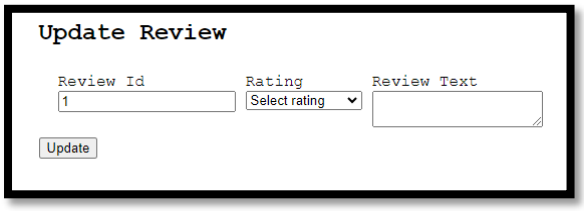
Create a new review on the /review/create/ page



Retrieve all the member's reviews on the review/retrieve/all/ page



From all reviews, click the "Update" button to update a review on the review/update/ page



Similarly, from all reviews, click the "Delete" button to update a review on the review/delete/ page

Any error during the above CRUD should have a visual feedback.

3.

1) Implement a user-defined database **function**, named **compute_running_total_spending** that computes the total spending of every recently active member.

   - A member is considered as **recently active** if his/her last login is **within the last 6 months**
   - The total spending of a member is contributed by the spending of all his/her orders that were **completed**.
   - When the function is invoked, it **computes** and **update** the running_total_spending column of the member table for **every member** that is active based on the above definition.
   - If a member is not active, the running_total_spending value should be set to null.
   - Refer to the appendix for an example of the computation.

2) Implement a **stored procedure**, named **compute_customer_lifetime_value** to compute the Customer Lifetime Value, CLV

   Customer Lifetime Value (CLV) is a metric that estimates the total revenue a business can expect from a single customer over the duration of their relationship.

$$Customer\ Lifetime\ Value = Average\ Purchase\ Value \times Purchase\ Frequency \times Retention\ Period$$

where

$$Average\ Purchase\ Value = \frac{Total\ Amount\ Spent}{Total\ Number\ of\ Orders}$$

$$Purchase\ Frequency = \frac{Total\ Number\ of\ Orders}{Customer\ Lifetime}$$

   - Customer lifetime is defined as the **number of years** between a customer's first order and a customer's most recent order.

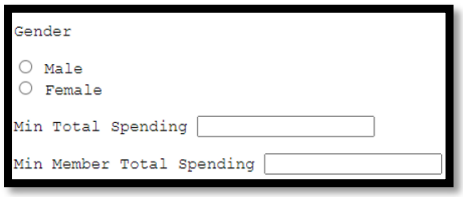
- **Retention period** is defined as how long a customer remains active or retained as a paying customer. In this computation, you can assume it to be **2 years**.
- When the stored procedure is invoked, it should **compute** and **update** the CLV column of the member table for all members.
- If a customer has no orders or only one order, the CLV should be set to null.
- Refer to the appendix for an example of the computation

4.

1) Implement a database user-defined **function**, named **get_ age_group_spending** that retrieves data that provides summarized data of the different age groups' spending as shown.

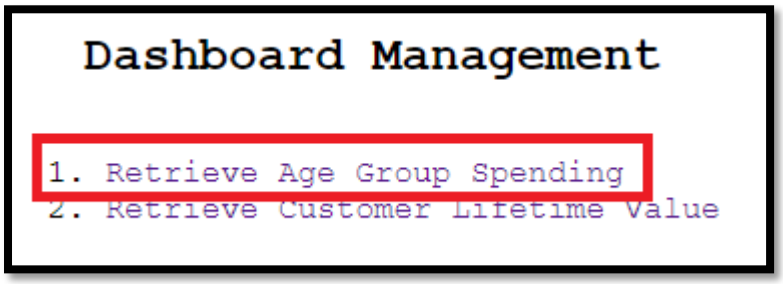
Age Group	Total Spending	No. of Members
18-29	14610.5	2
30-39	11390.5	3
40-49	4626.5	3
50-59	6967	2

It must be able to filter on the following:

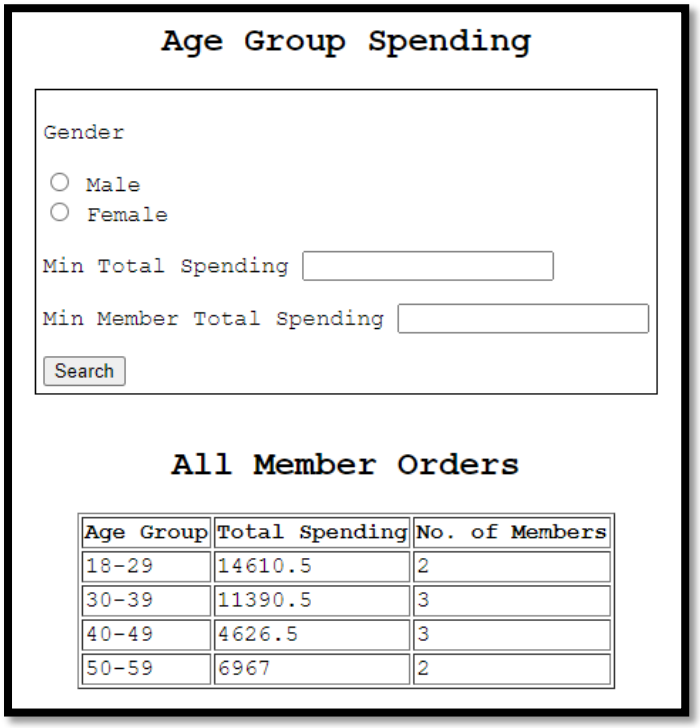


Note that this is an administrator's feature, so you will need to login as an admin to access the admin interface on the application.

As a completed end-to-end feature, it would appear as below under the admin/dashboard/ageGroupSpending/ page.



When no filters are applied, the function should display the summary data of all members' sales orders in the age groups as shown.



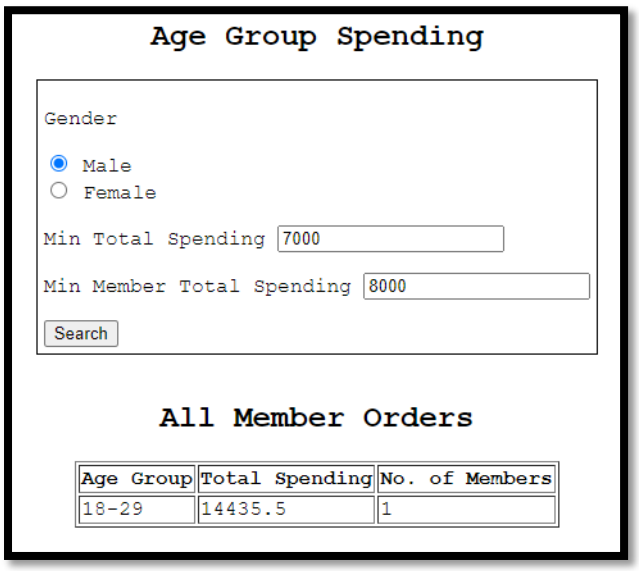
The function should be able to filter on

- gender
- minimum total spending
- minimum member total spending (the total amount of all his/her ordered items)

The filtering should work as an AND combination instead of OR combination:
For example, if

- the gender is set to 'Male'
- the minimum total spending is set to 7000
- The minimum member total spending is set to 8000

The result retrieved should include only members who are male **and** the minimum average spending is at least 7000 **and** include only the members whose total spending is at least 8000.



2) Implement the model functions, controller functions and express routes to make it a complete end-to-end feature.

b) Optional Requirement

Implement the feature for a member to **manage their favorite products** on the web application. Typically, a member would be able to add, update and remove products from their favorites. A member will also be able to view and manage their favorite products, as a single list or as multiple lists. There will also be some insightful information, such as which products are popularly favorited. On the admin side, it is useful for the administrator to have a page to visualize filtered data or summarized data about favorites to make better business decisions.

You have the discretion to:

- Decide the end-to-end functionalities of this feature
  The functionalities should serve to demonstrate your proficiency in the evaluation criteria.
- Enhance the database design
  - Model new relations, add additional table(s) needed for this feature and apply normalization to ensure data redundancy
- Develop the database functionalities.
  - Create relevant functions and/or stored procedures to support the operations required. The bulk of your logic implementation should be here instead of at the models and controller functions.
- Develop the back-end based on the MVC pattern
  - Implement the necessary routes, controller functions and models.
- Develop the front-end pages
  - Implement the html and CSS and make changes and add pages to front-end

You are expected to stay within this feature scope. You have flexibility to enhance this feature but do not implement other features not related to favorite.

B. Evaluation

Your submission will be evaluated based on the below evaluation criteria:

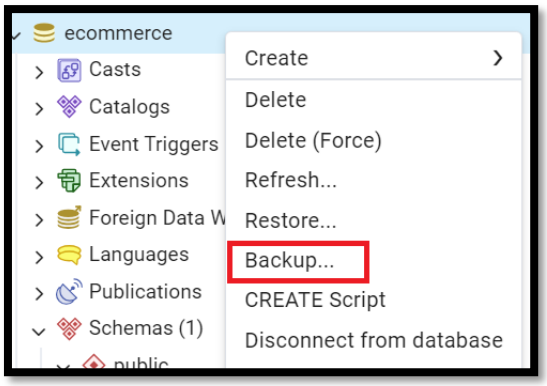
Criterion	Weight(%)	Evaluation
Database Design	10	Scope of design, normalization of tables to reduce data redundancy, as well as logical and correct representation of entities, attributes, constraints & relationships
Create Entity	10	Implementation & error-handling of database functionality to add new entities, and implementation as an end-to-end feature to fulfil requirements
Update Entity	10	Implementation & error-handling of database functionality to update entities, and implementation as an end-to-end feature to fulfil requirements
Delete Entity	10	Implementation & error-handling of database functionality to delete entities, and implementation as an end-to-end feature to fulfil requirements
Retrieve Entity	10	Implementation & error-handling of database functionality to retrieve entities, and implementation as an end-to-end feature to fulfil requirements
Query Quality	10	Proficient use of SQL operations such as joins, filtering, sorting, grouping and aggregating to enable end-to-end features
User defined Function Program Logic	10	Proficient use of control structures and program logic to handle program complexity
Stored procedures Program Logic	10	Proficient use of control structures and program logic to handle program complexity
Report Quality	10	Detailed documentation with good supporting evidence to substantiate each criterion
Demonstration & Interview	10	Demonstrate database functionalities and seamless end-to-end features. Proficiently address interview questions.

-- End of Assignment Specifications --

C. Appendix

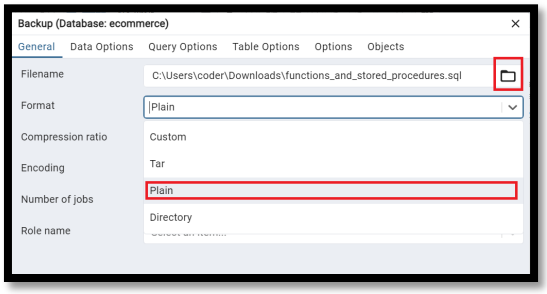
How to Export User-defined functions and stored procedures to SQL file:

1.



Choose the option "Backup..."

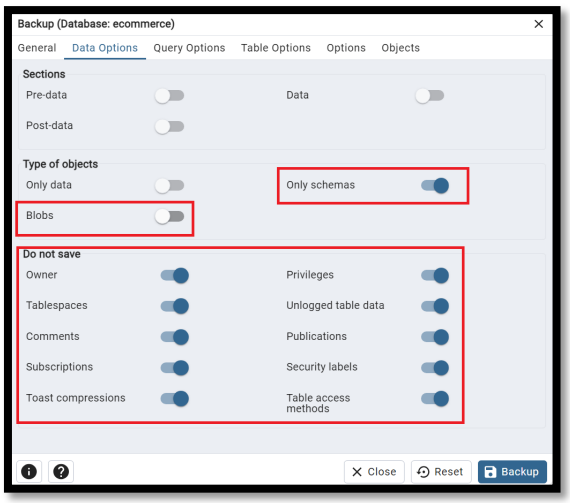
2.



Under the General Tab:

- Browse and select where to store your file
- Select the format as "Plain"

3.

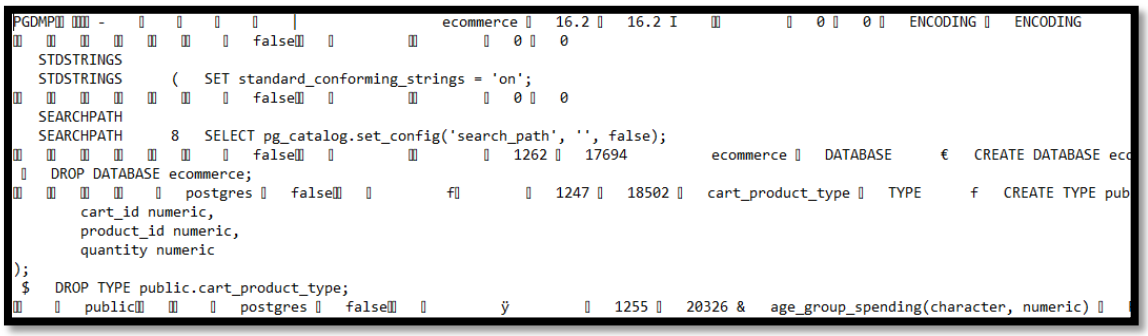


Under the Data Options Tab:

Make sure

- "Blobs" is toggled **OFF**
- "Only schemas" is toggled **ON**
- Entire "Do not save" section is toggled **OFF**

4.



- Check that if the file is opened with a text editor, it contains readable text.
- If it opens to show something similar to the above, you have exported in the wrong format.

5. Add it as a file to your repository and commit it like any other file.

Example to compute running total spending for a single member:

Member ID: 1

Last Login Date: 2024-06-24 (within last 6 months from today's date)

Orders:

Order 1:

Order ID: 101
Order Date: 2023-05-15
Status: COMPLETED
Items:
Product A: Quantity 2, Unit Price $10
Product B: Quantity 1, Unit Price $20

Order 2:

Order ID: 102
Order Date: 2023-06-10
Status: COMPLETED
Items:
Product A: Quantity 3, Unit Price $10
Product C: Quantity 1, Unit Price $30

Order 3:

Order ID: 103

Order Date: 2023-06-20
Status: CANCELLED (wont be included in computation)

Calculate Order 1:

Product A: 2 × $10 = $20
Product B: 1 × $20 = $20
Total for Order 1: $20 + $20 = $40

Calculate Order 2:

Product A: 3 × $10 = $30
Product C: 1 × $30 = $30
Total for Order 2: $30 + $30 = $60

Running Total:

For Member ID 1, considering only Order 1 and Order 2 (since Order 3 is cancelled and not included):
Running Total Amount: $40 + $60 = $100

Example to compute customer lifetime value for a single member:

Member ID: 1

Orders:

Order 1:
Amount: $100 (This is the total of unit price * quantity for all order items in this order)
Date: 2023-01-01
Order 2:
Amount: $150
Date: 2023-04-15
Order 3:
Amount: $120
Date: 2023-08-20

Calculate Total Spent:
Total Spent = $100 + $150 + $120 = $370

Total Orders:
Total Orders = 3

First order and most recent order dates:

First Order Date: 2023-01-01
Most Recent Order Date: 2023-08-20

Calculate Customer Lifetime: (computed in number of years)

Customer Lifetime = (2023-08-20 - 2023-01-01) / 365 ≈ 0.65 years

Calculate Average Purchase:

Average Purchase Value = Total Spent / Total Orders = $370 / 3 ≈ $123.33

Calculate Purchase Frequency:

Purchase Frequency = Total Orders / Customer Lifetime = 3 / 0.65 ≈ 4.62 orders per year

Calculate Customer Lifetime Value:

Assuming Retention Period = 2 years
CLV = Average Purchase Value × Purchase Frequency × Retention Period
= $123.33 × 4.62 × 2 ≈ $1141.72