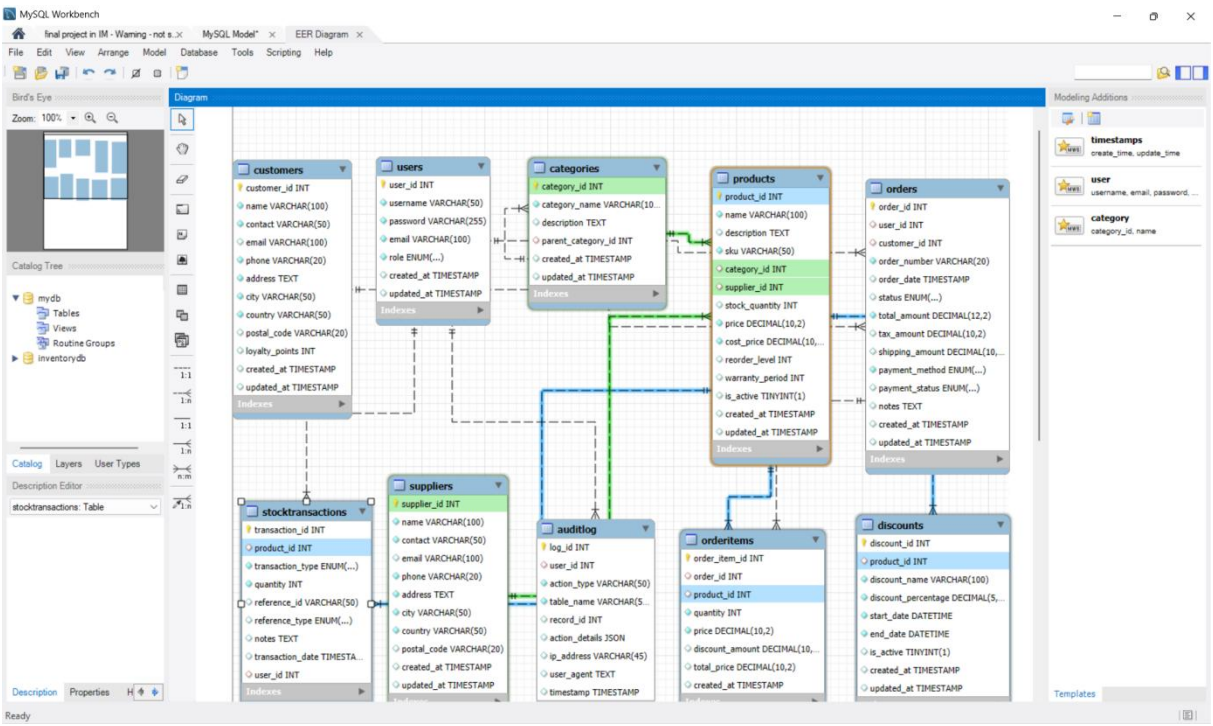
REPUBLIC OF THE PHILIPPINES
BICOL UNIVERSITY
BICOL UNIVERSITY POLANGUI

INVENTORY MANAGEMENT SYSTEM

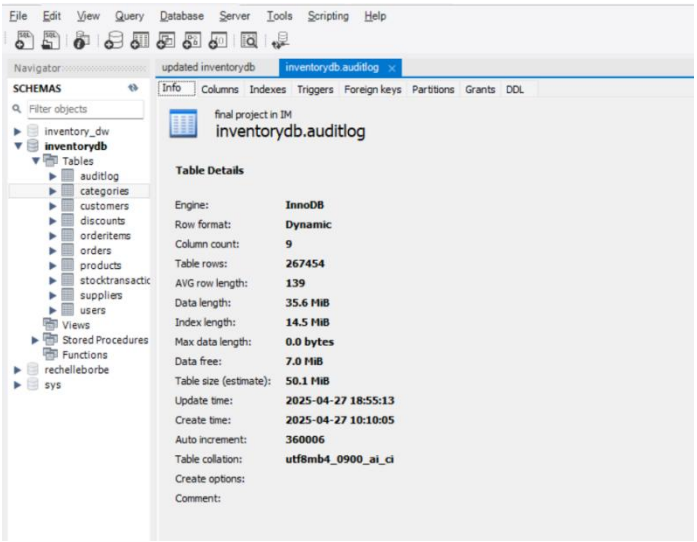
Final Documentation Compilation

➢ Final ER Diagram

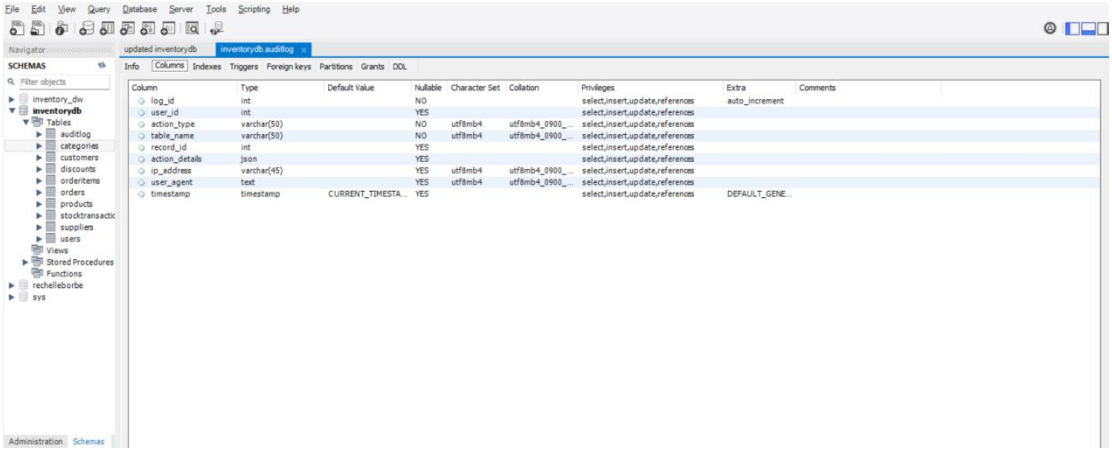


➢ Database Schema and Table Description

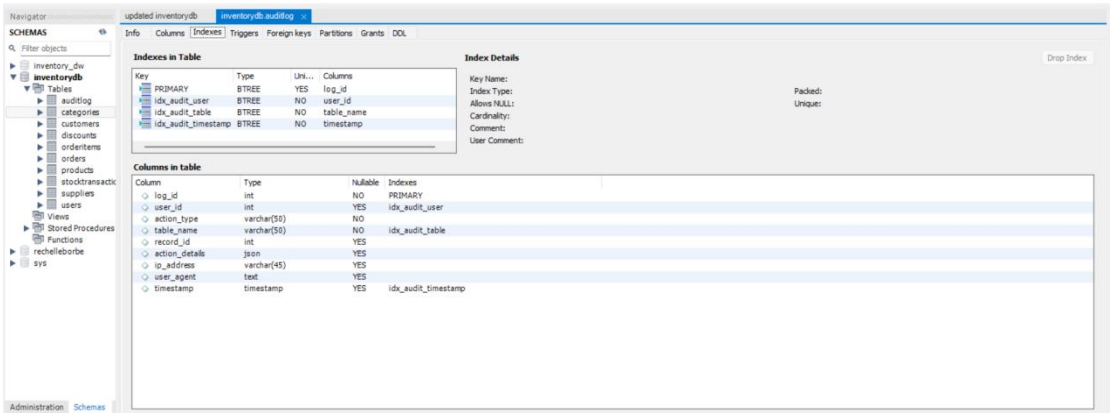
1. Auditlog



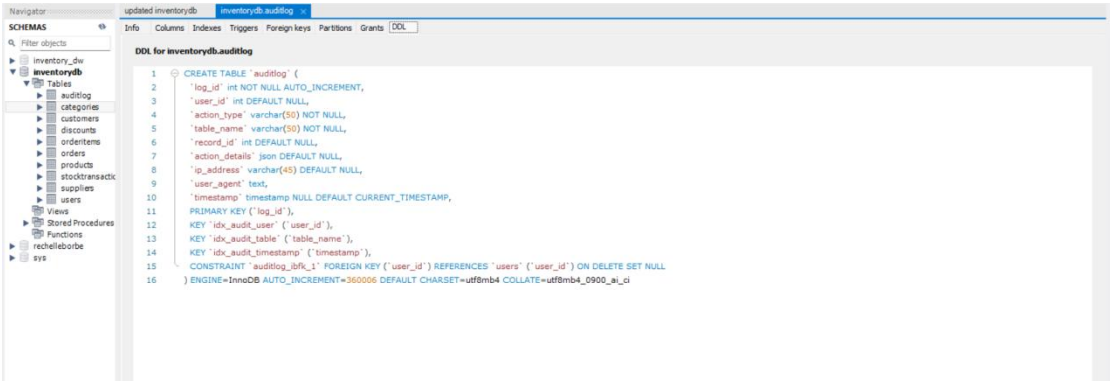
A. info



B. Columns

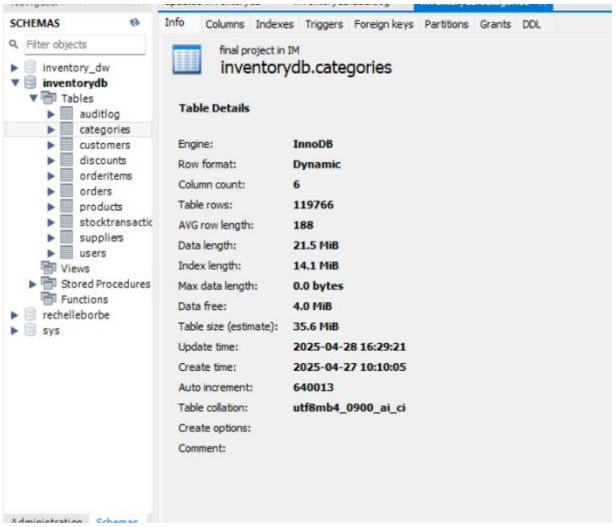


C. indexes

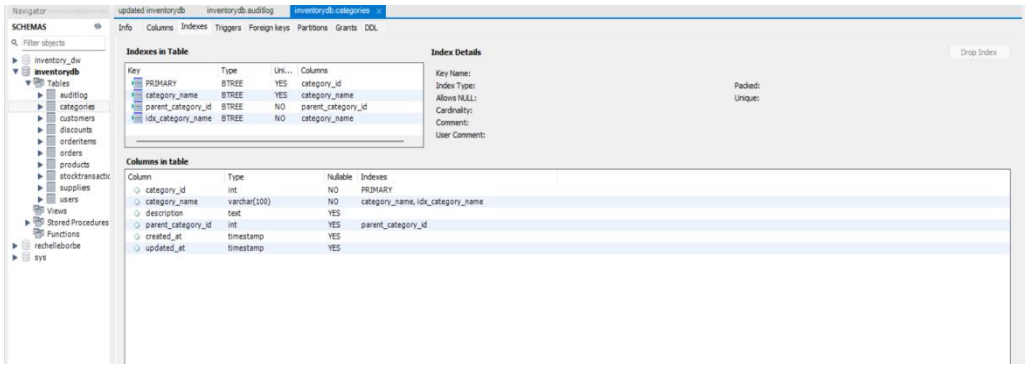


D. Data Definition Language

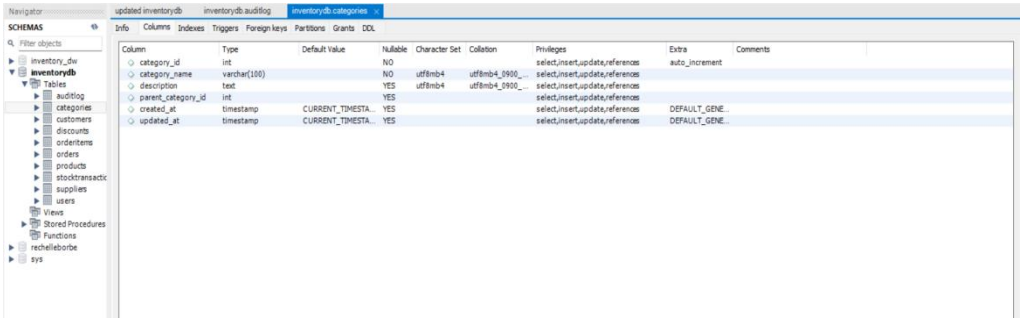
2. Categories



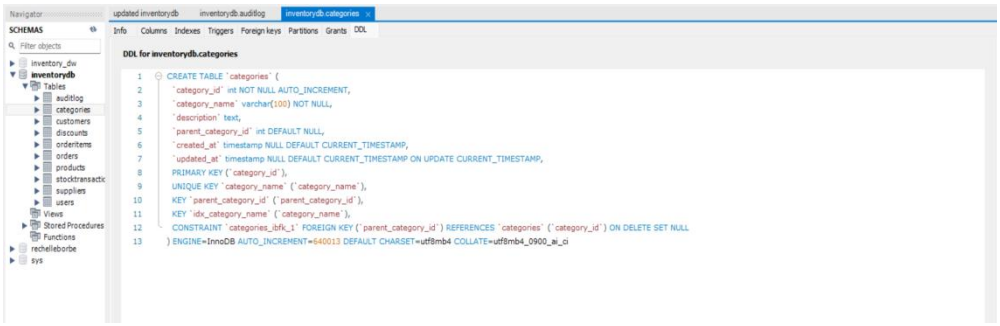
A. info



B. Columns

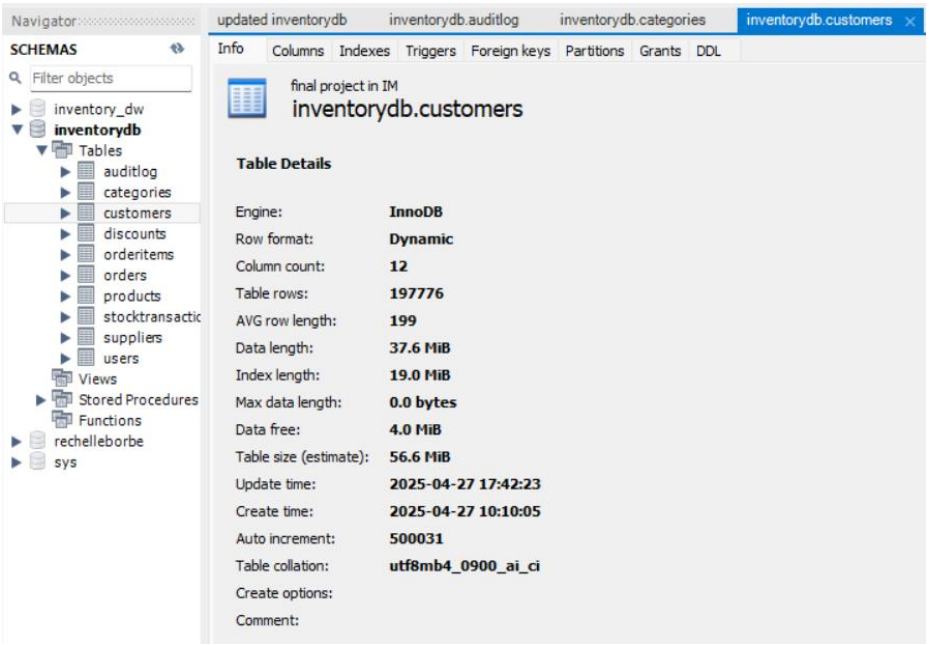


C. Indexes

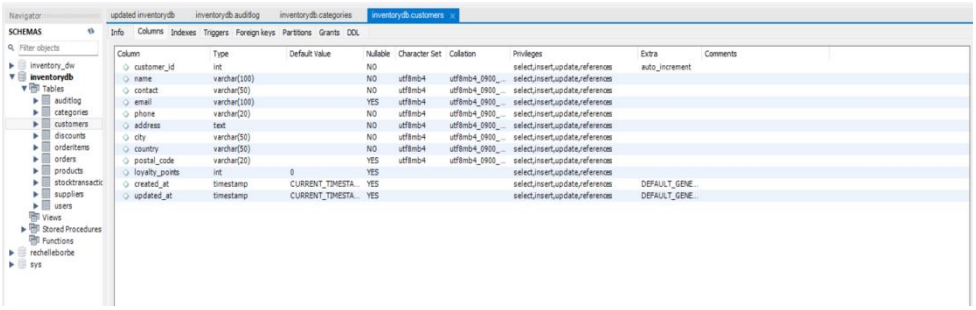


D. Data Definition Language

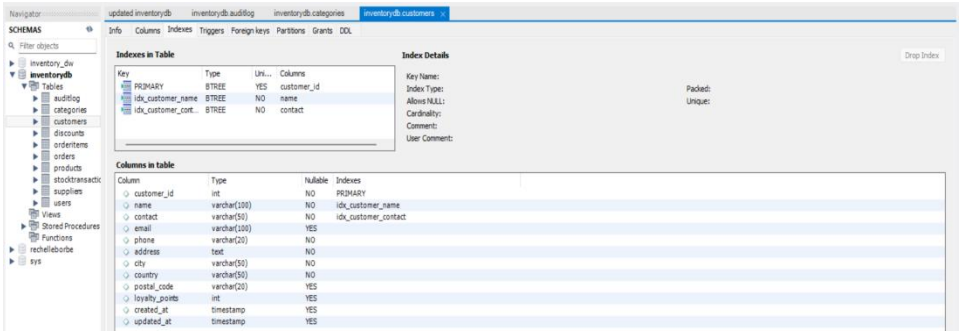
3. Costumers



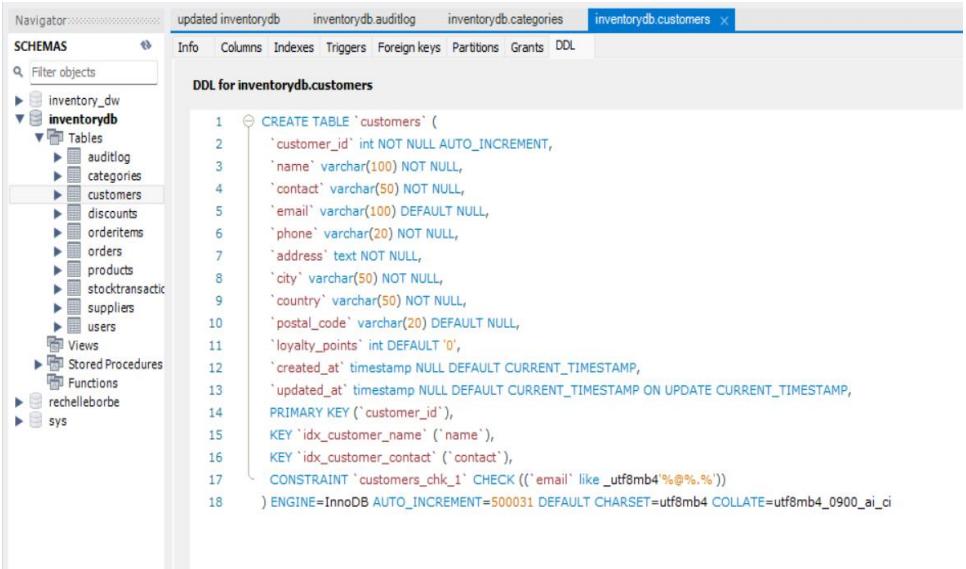
A. Info



B. Columns

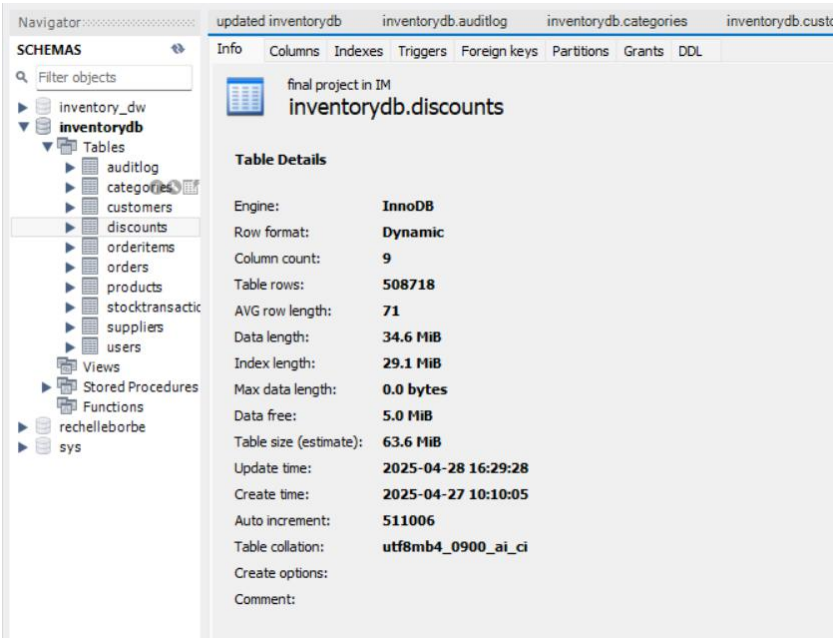


C. Indexes

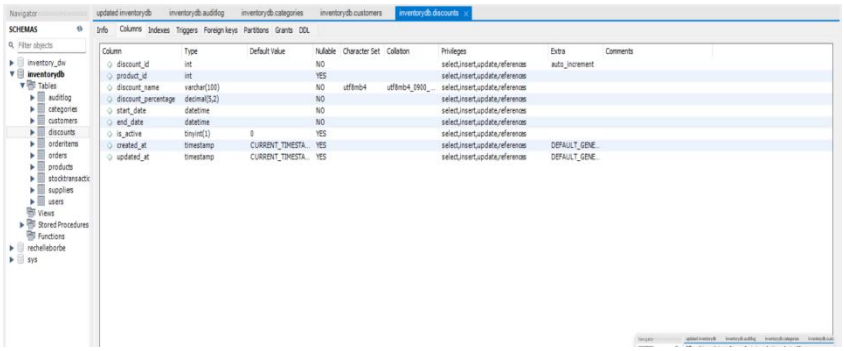


D. DAta Definition Languages

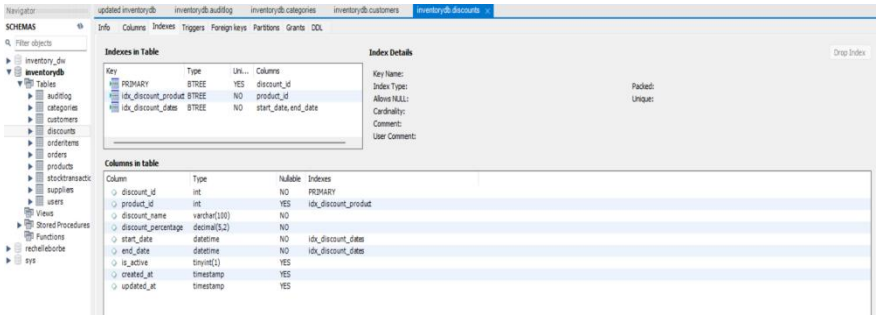
4. Discounts



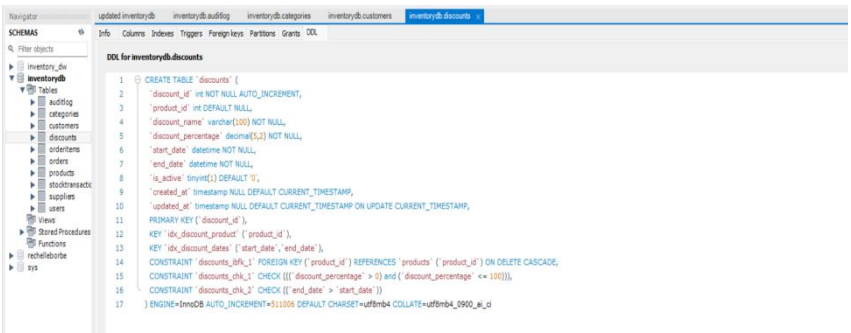
A. info



B. Columns

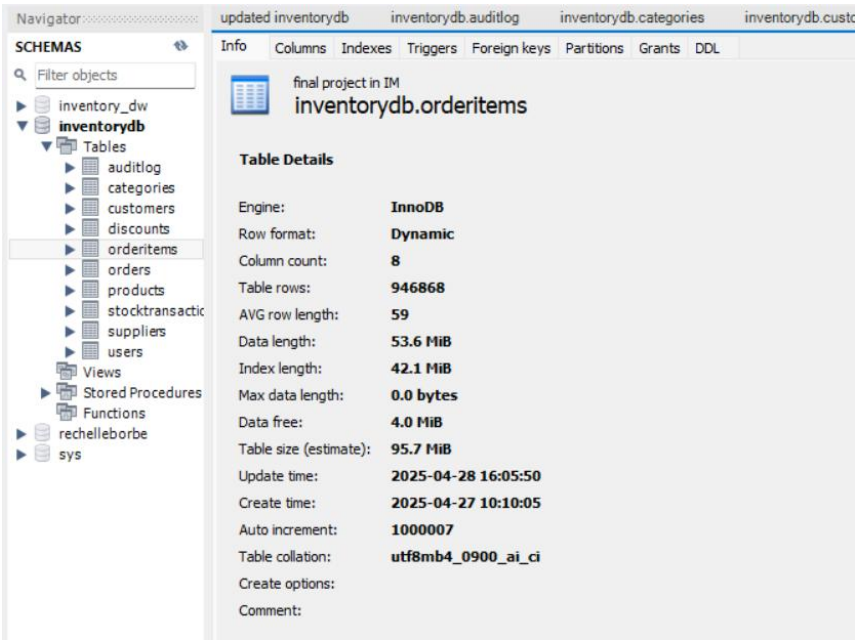


C. Indexes

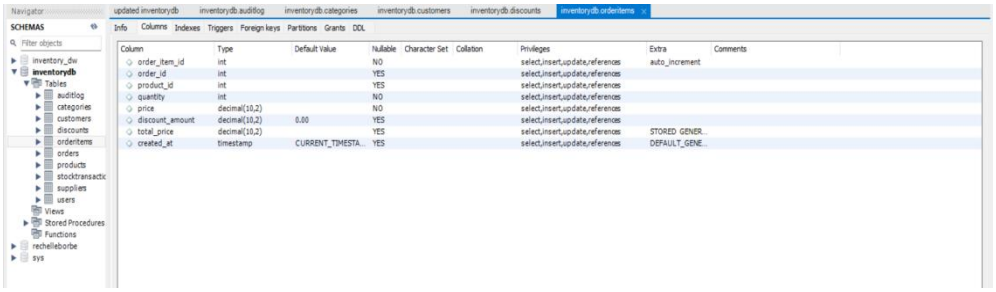


D. Data Definition Language

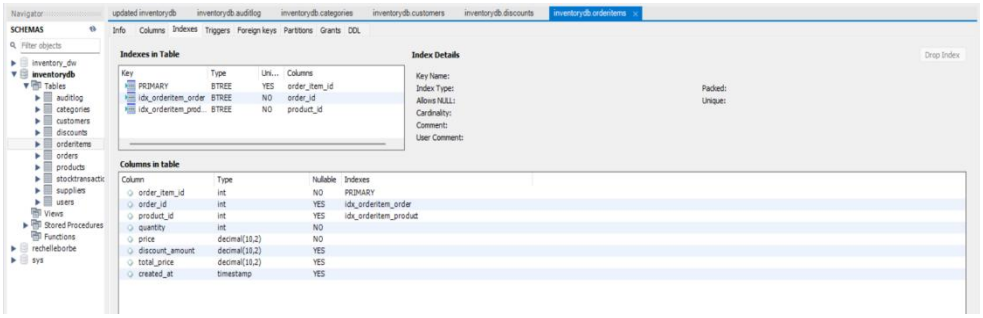
5. **OrderItems**



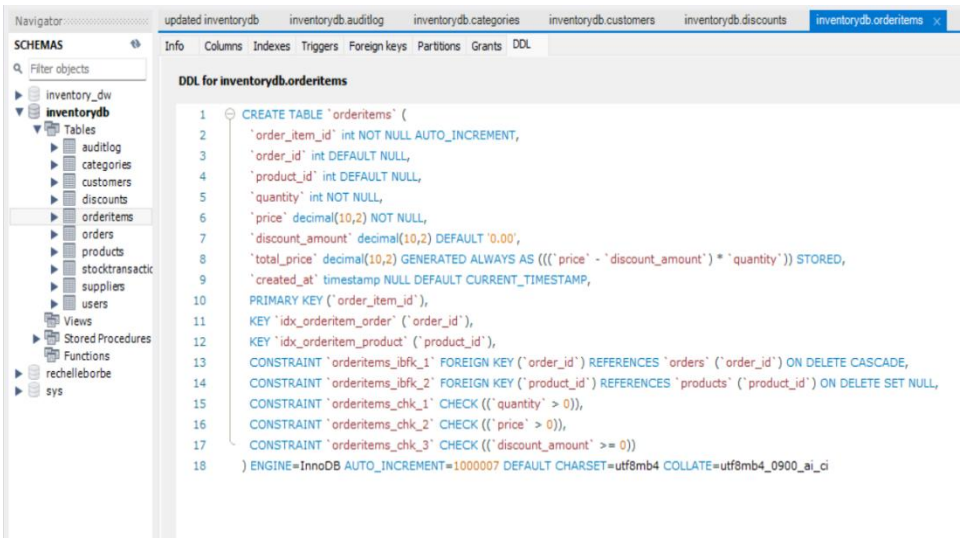
A. Info



B. Columns

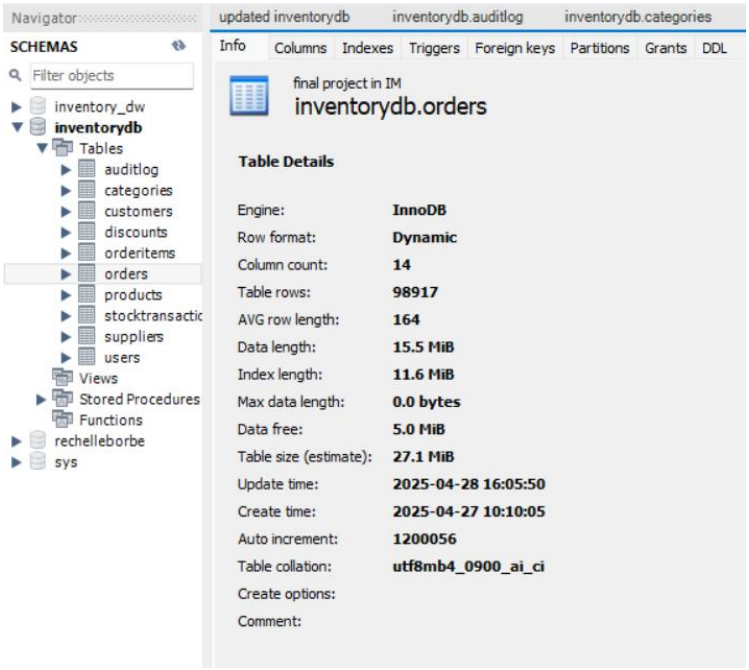


C. Indexes

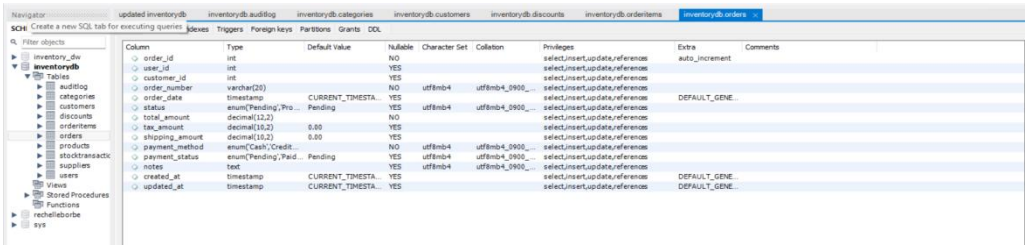


D. Data Definition Language

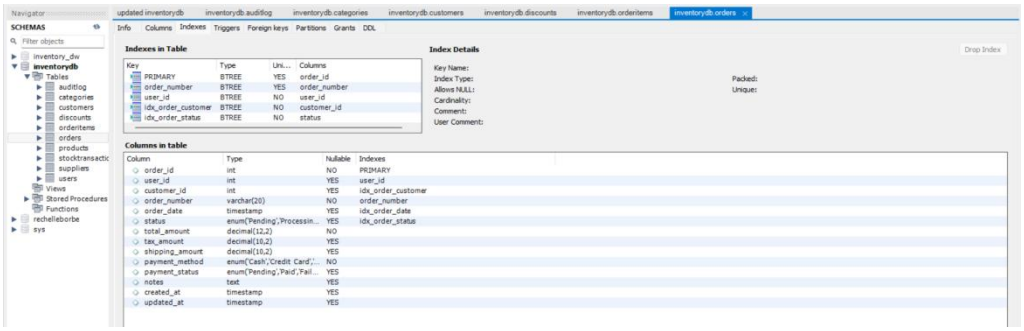
6. Orders



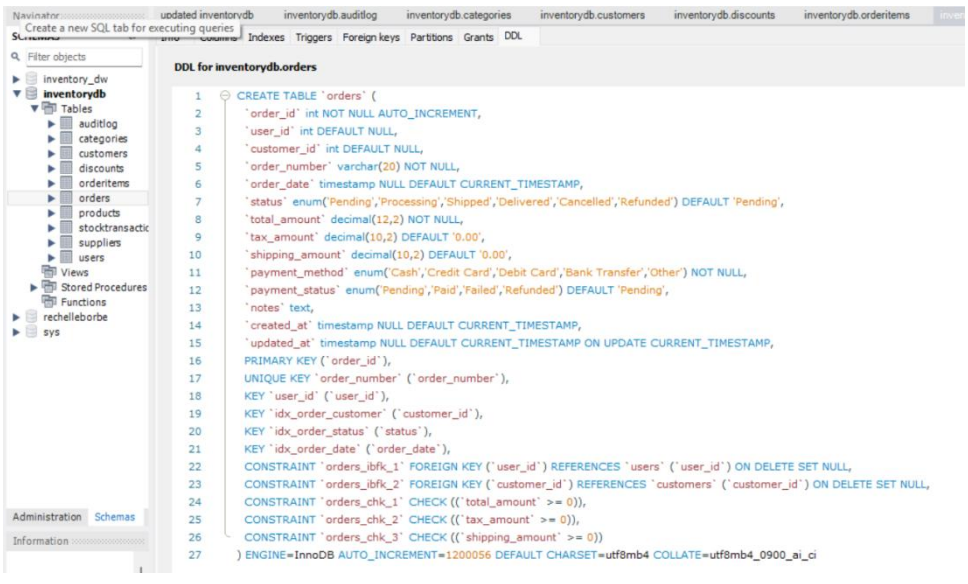
A. Info



B. Columns

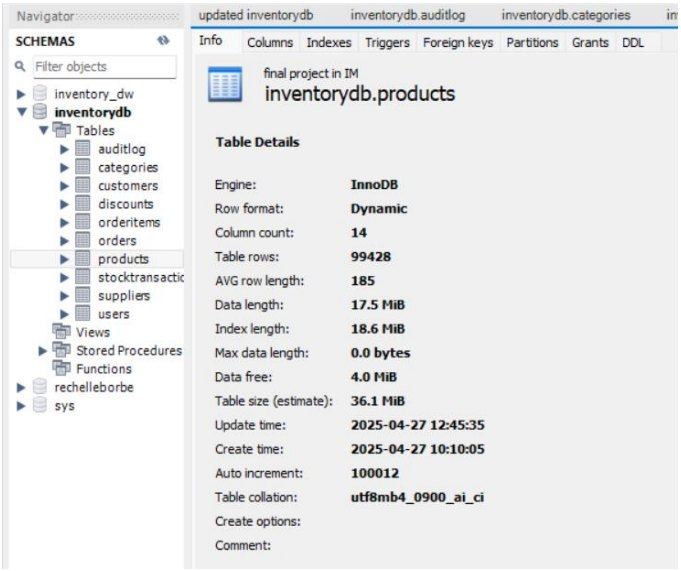


C. Indexes

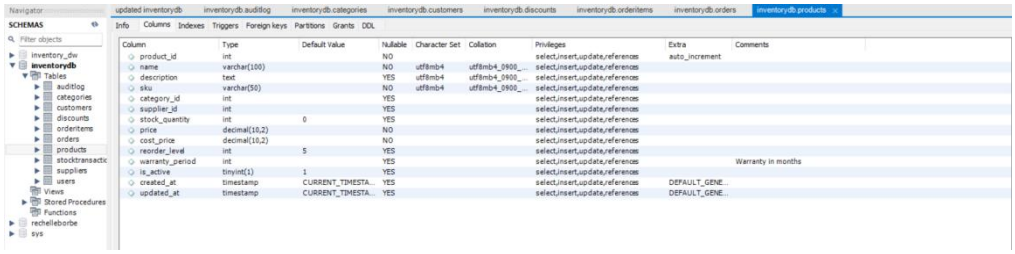


D. Data Definition Languages

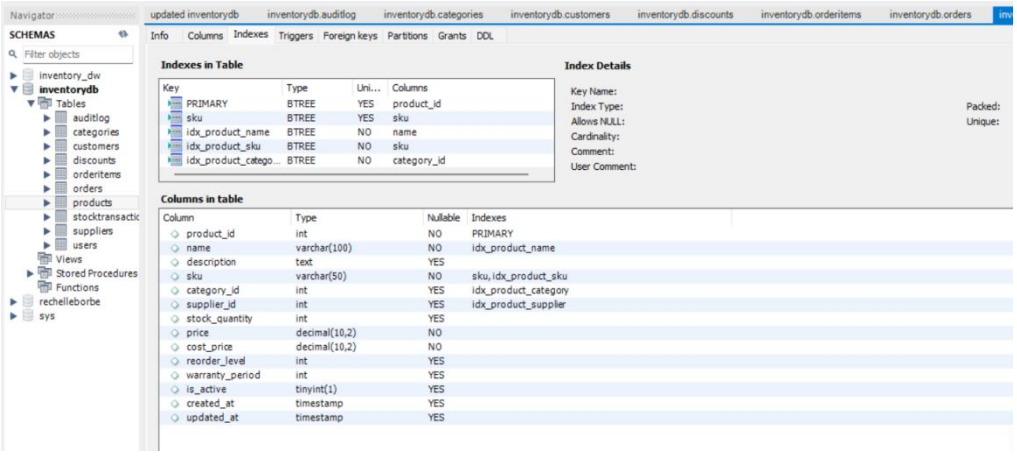
7. **Products**



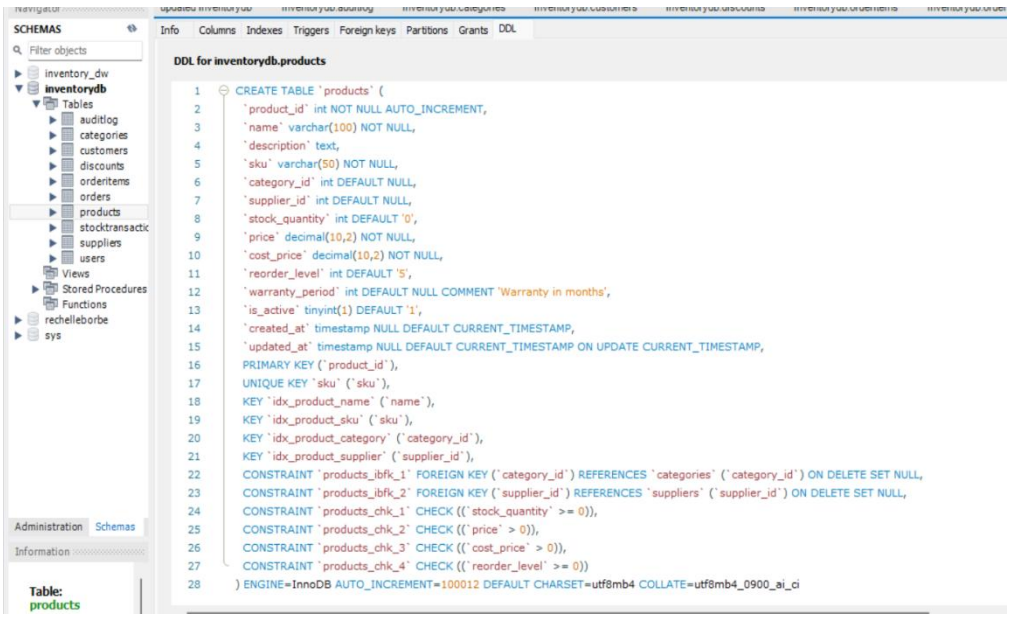
A. info



B. Columns

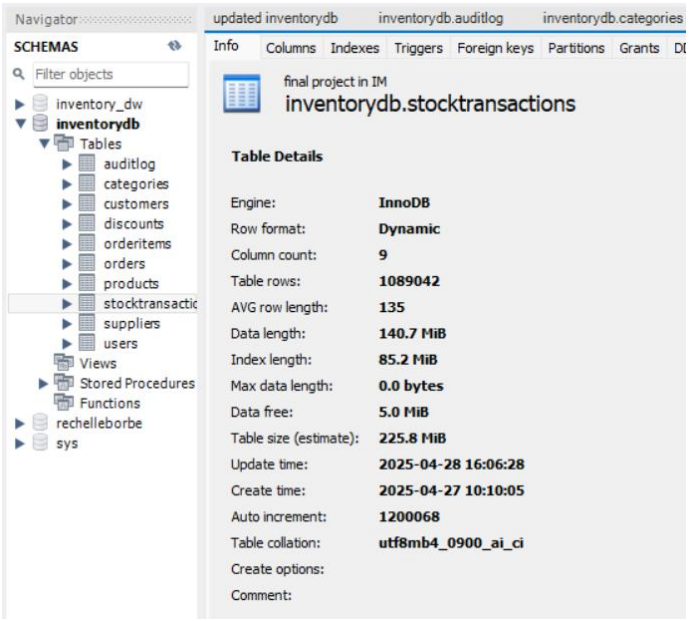


C. Indexes

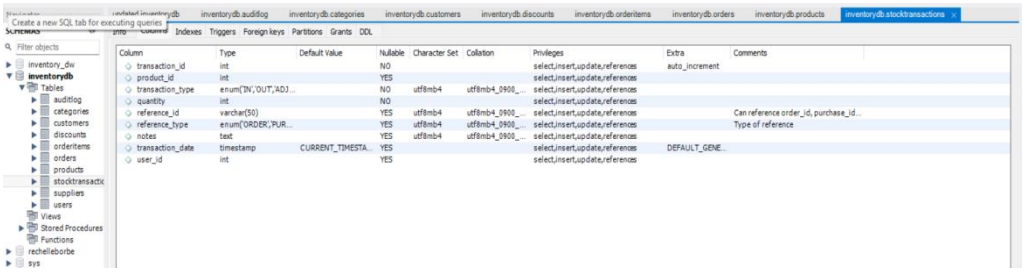


D. Data Definition Language

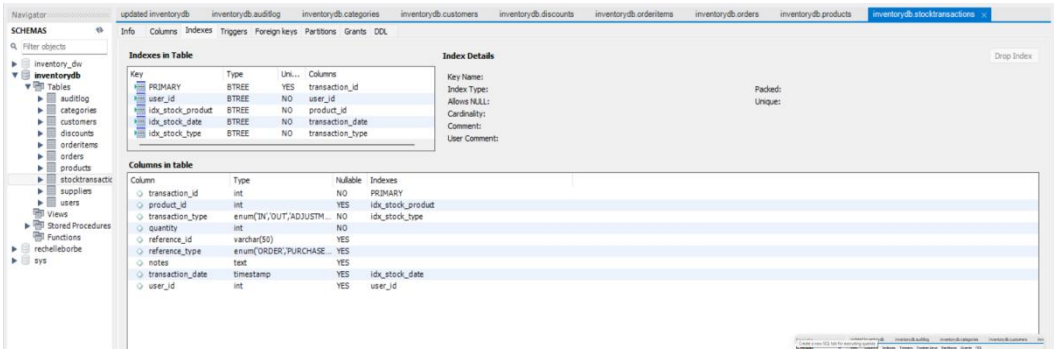
8. StockTransaction



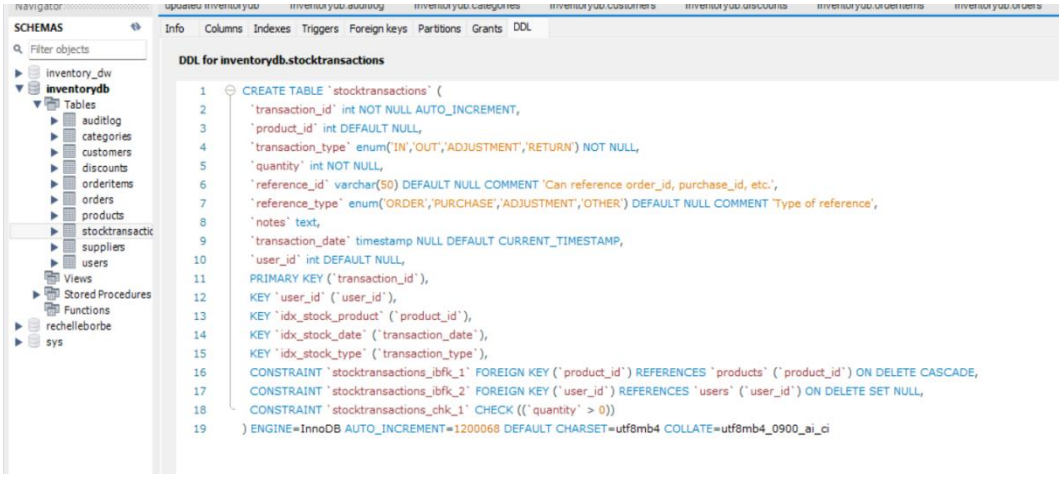
A. info



B. Columns

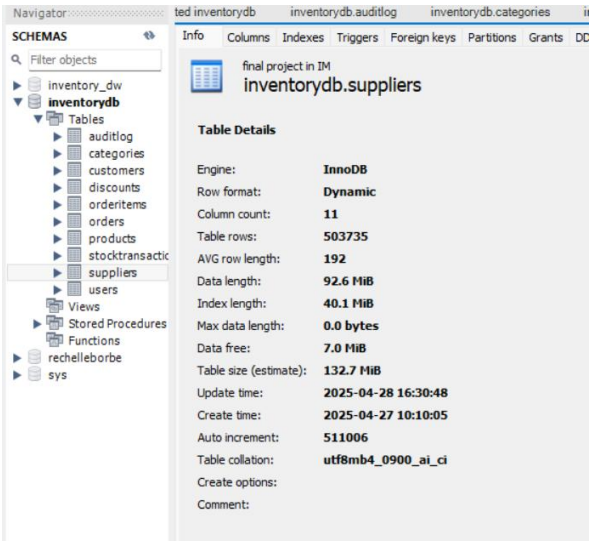


C. Indexes

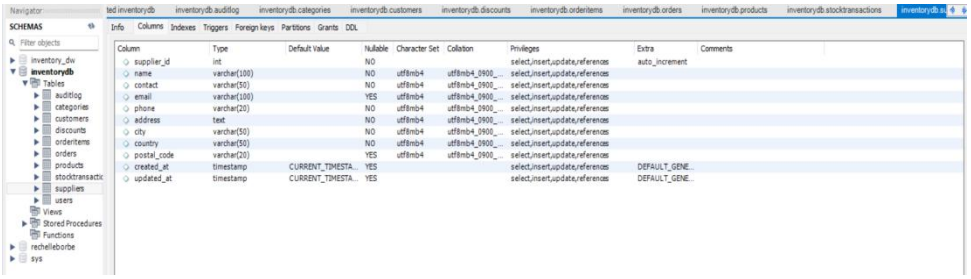


D. Data Definiton Languange

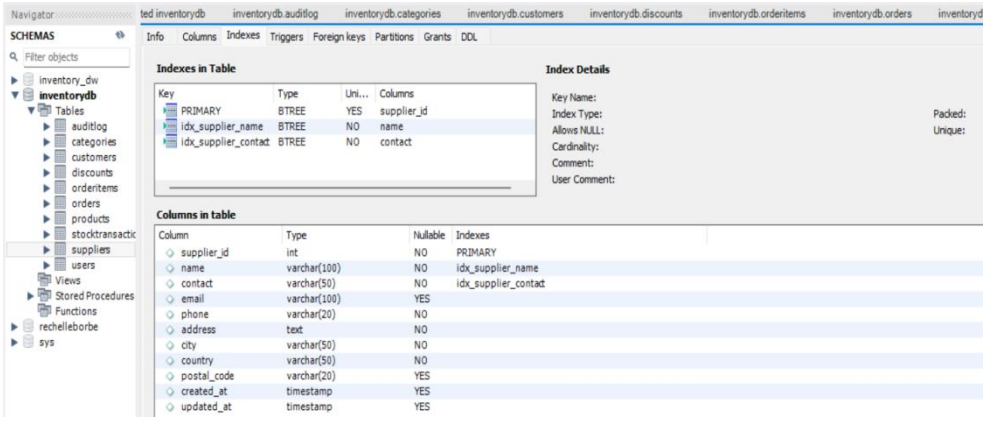
9. Suppliers



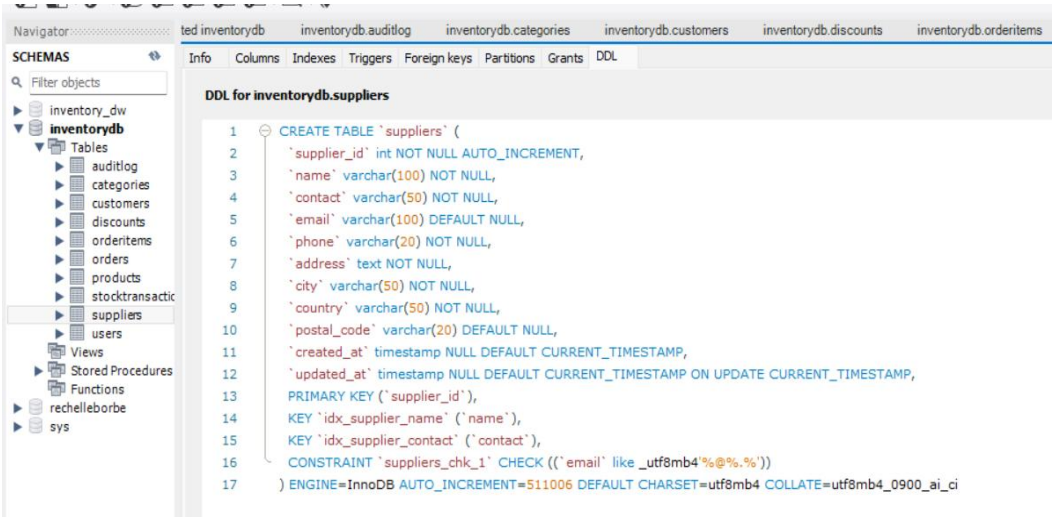
A. info



B. Columns

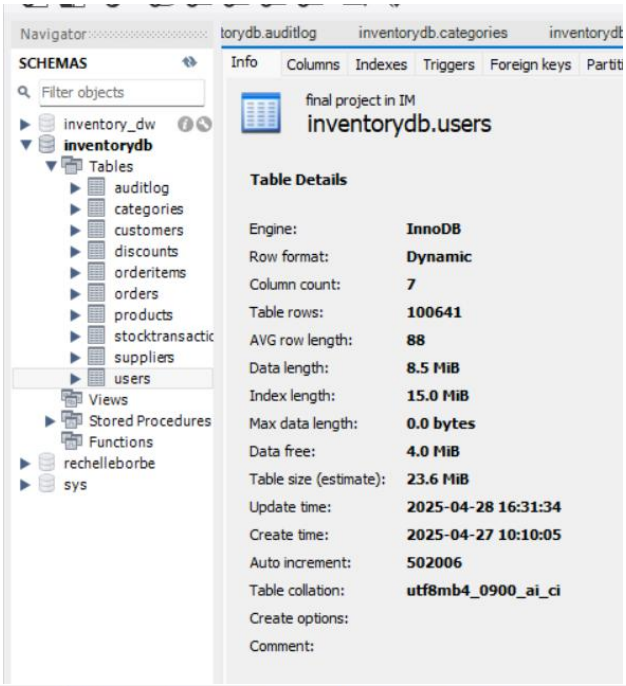


C. Indexes

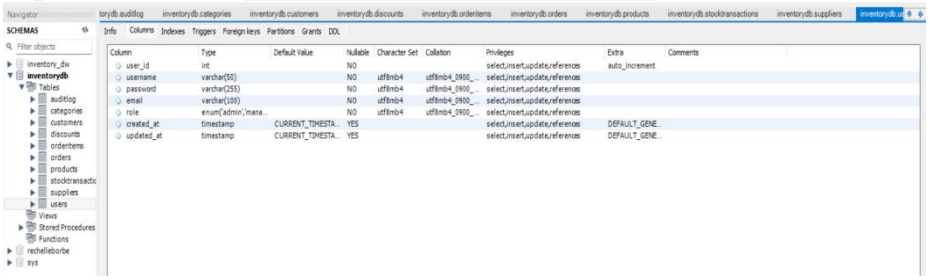


D. Data Definition Languages

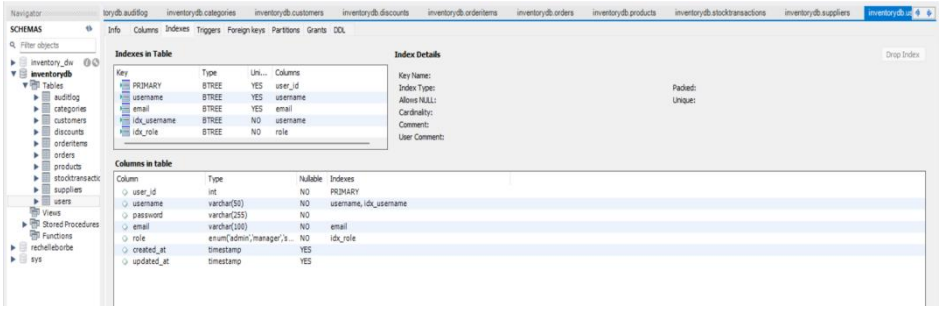
10. Users



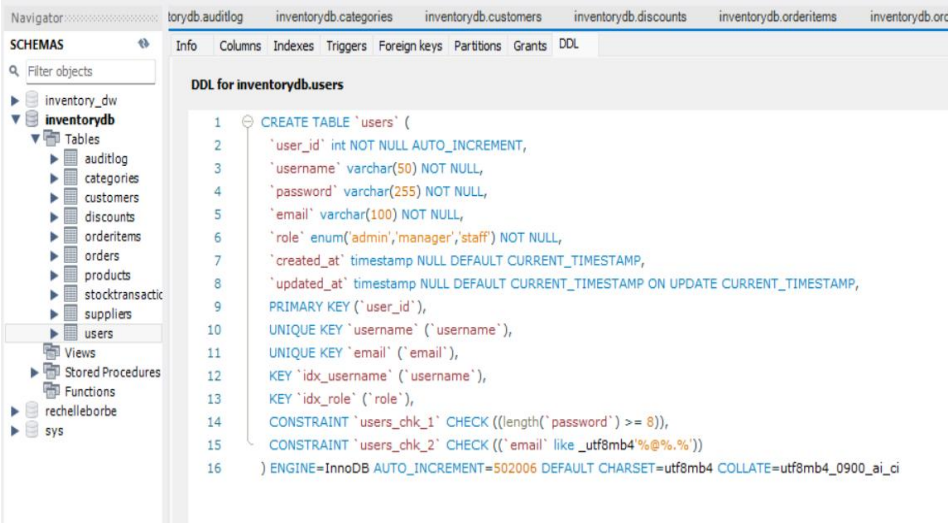
A. info



B. Columns



C. Indexes



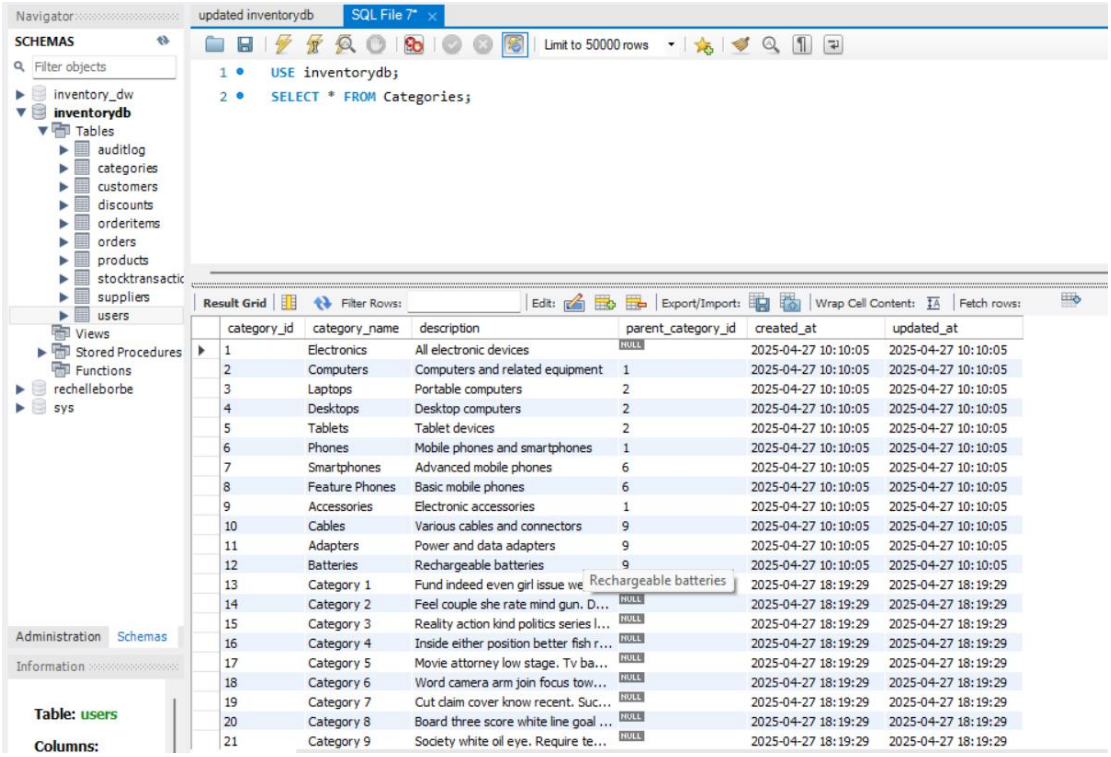
D. Data Definition Languages

➢ Sample SQL Queries(basic, BI, optimized)

1. Basic

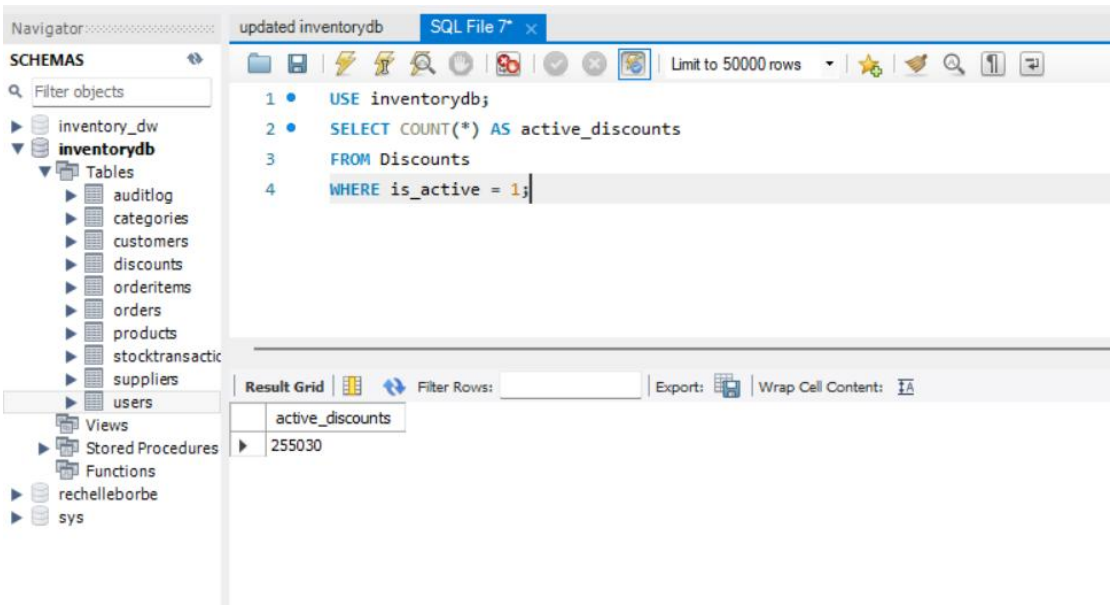
- Retrieve all categories

```
SELECT * FROM Categories;
```



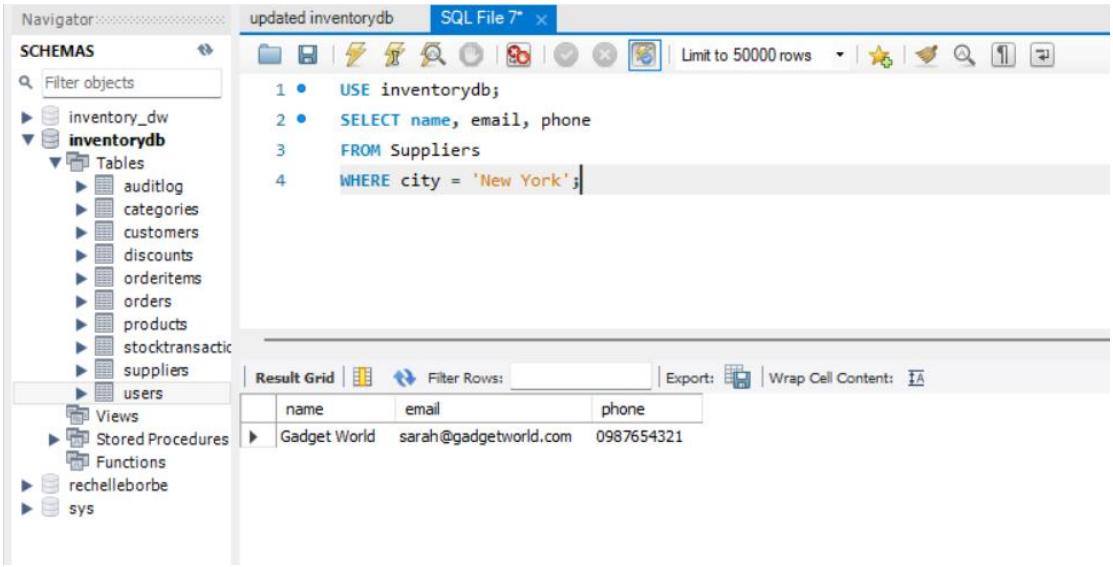
- Count the number of active discounts

```
SELECT COUNT(*) AS active_discounts
FROM Discounts
WHERE is_active = 1;
```



- List all suppliers in a specific city

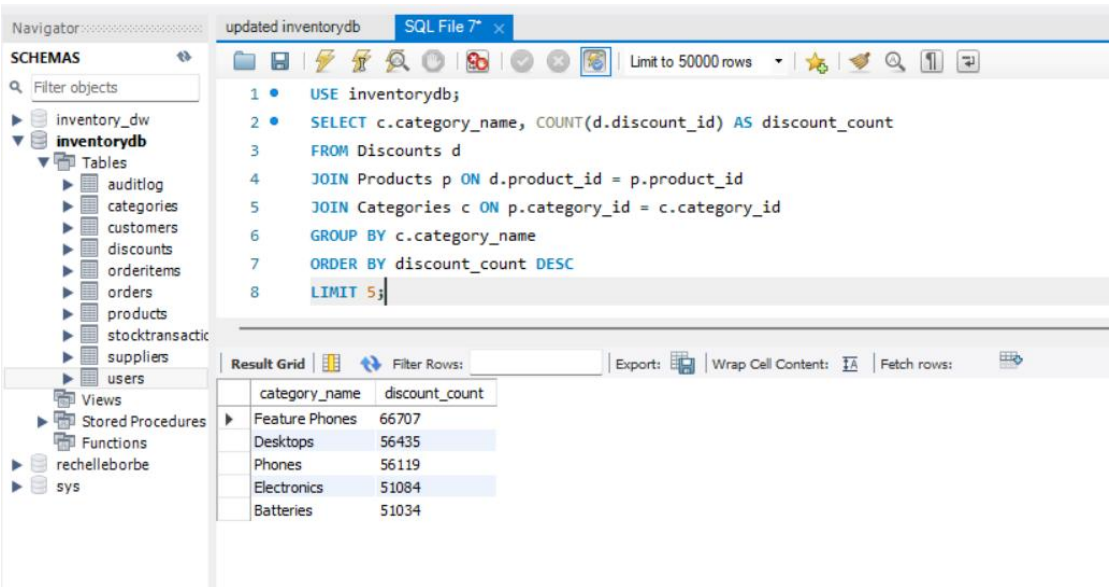
```
SELECT name, email, phone
FROM Suppliers
WHERE city = 'New York';
```



2. Business Intelligence

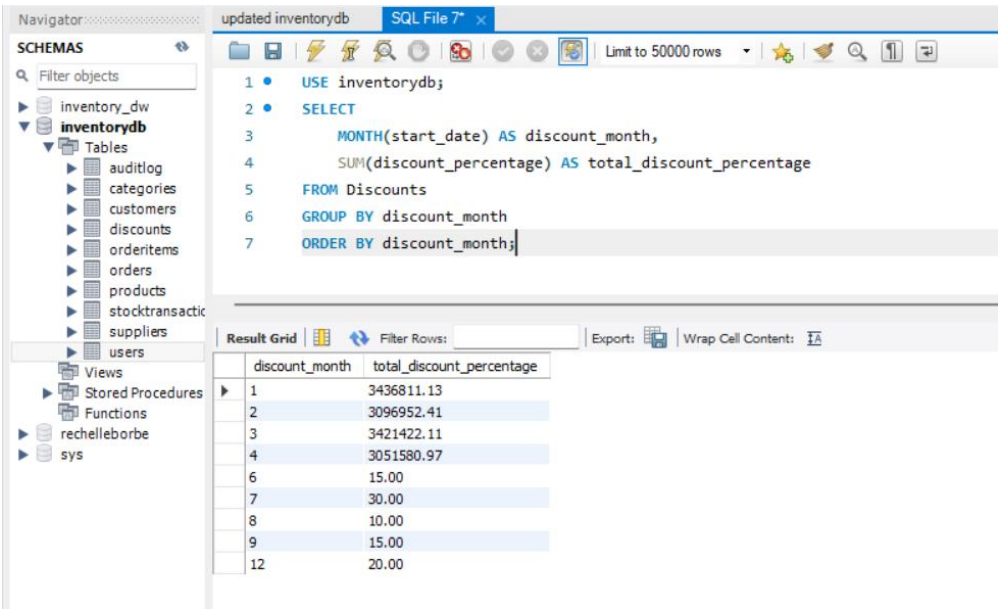
- Top 5 Categories by number of Discounts

```
SELECT c.category_name, COUNT(d.discount_id) AS discount_count
FROM Discounts d
JOIN Products p ON d.product_id = p.product_id
JOIN Categories c ON p.category_id = c.category_id
GROUP BY c.category_name
ORDER BY discount_count DESC
LIMIT 5;
```



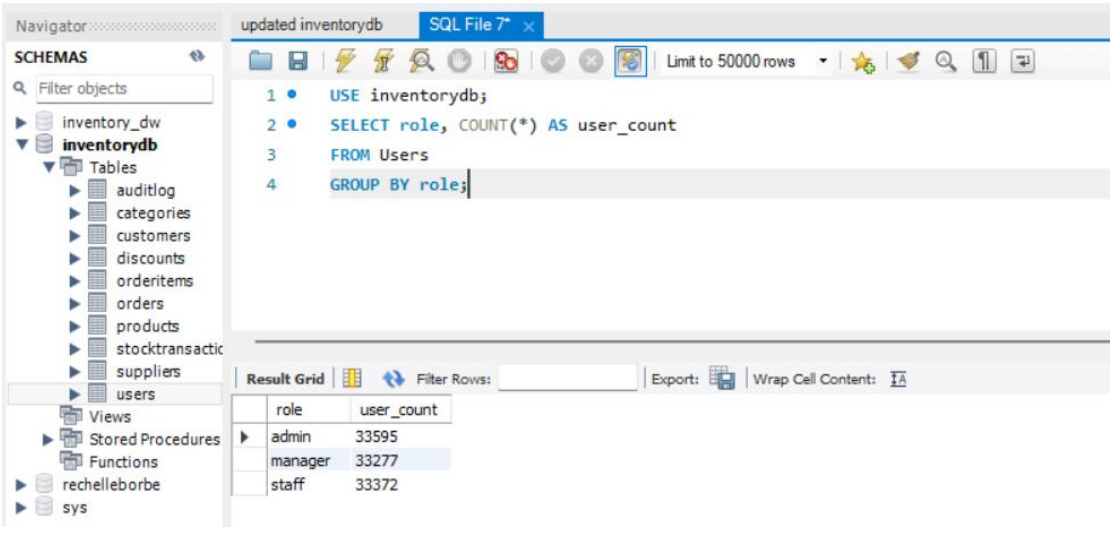
- Total Discounts By Months

```
SELECT
MONTH(start_date) AS discount_month,
SUM(discount_percentage) AS total_discount_percentage
FROM Discounts
GROUP BY discount_month
ORDER BY discount_month;
```



- Number of users by roles

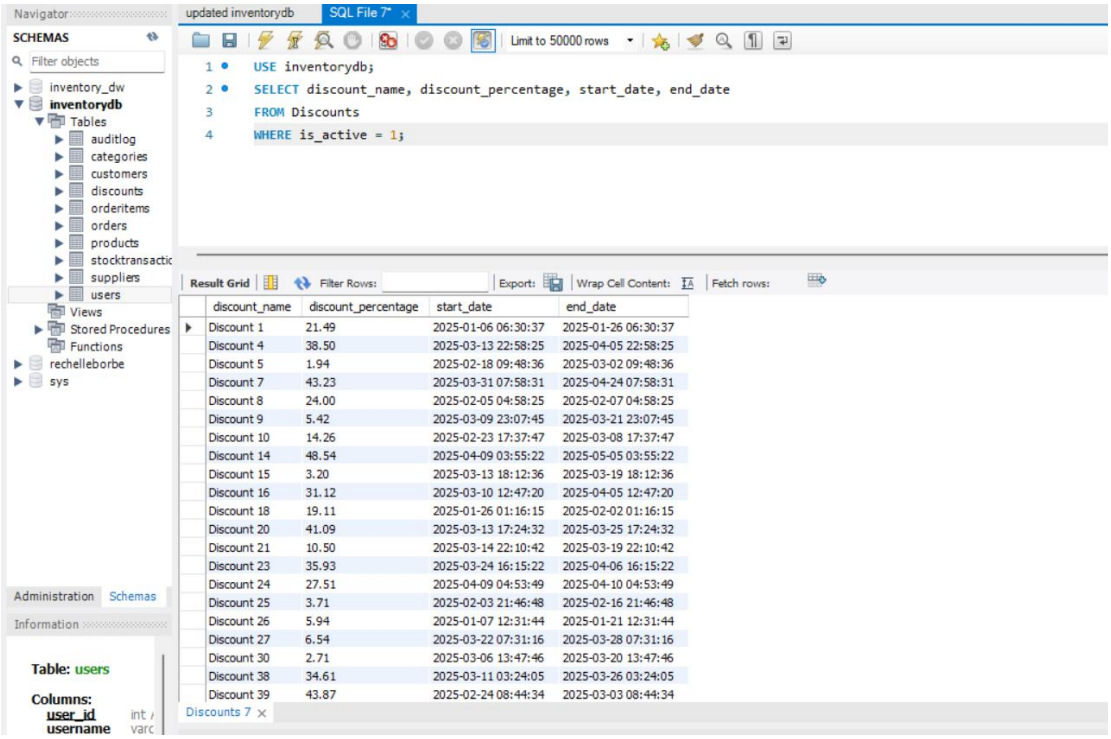
```
SELECT role, COUNT(*) AS user_count
FROM Users
GROUP BY role;
```



3. Optimized SQL queries

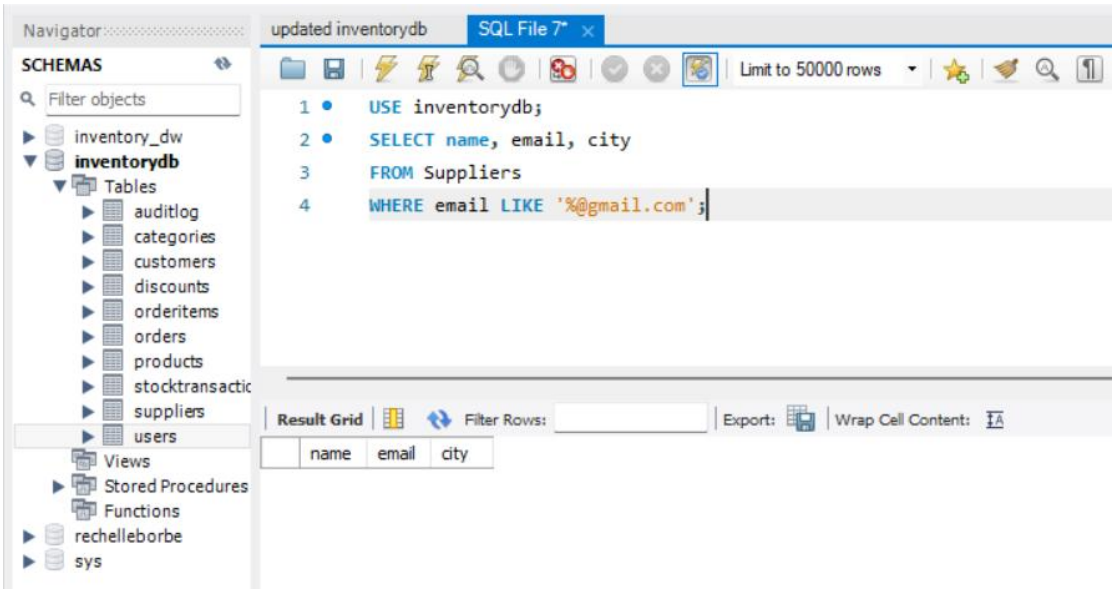
- Retrieve Active Discounts with index optimization

```
SELECT discount_name, discount_percentage, start_date, end_date
FROM Discounts
WHERE is_active = 1;
```



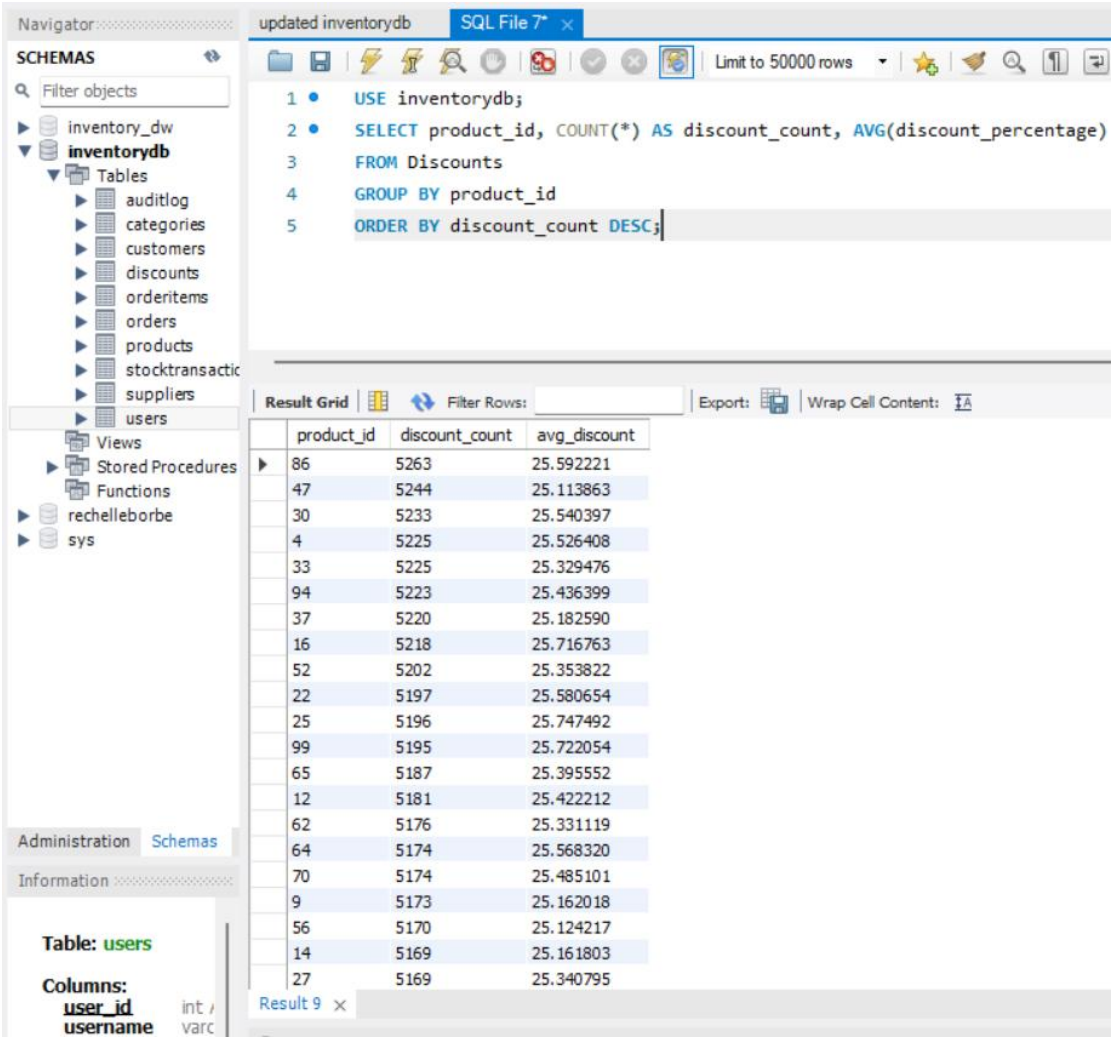
- Retrive suppliers with unique email

```
SELECT name, email, city
 FROM Suppliers
 WHERE email LIKE '%@gmail.com';
```



- Aggregate Discounts by products

```
SELECT product_id, COUNT(*) AS discount_count, AVG(discount_percentage) AS
 avg_discount
 FROM Discounts
 GROUP BY product_id
 ORDER BY discount_count DESC;
```



➢ Backup and Recovery Steps

Backup and Restore Step: Data Export and Import using MySQL Workbench

1. Open MySQL Workbench
2. Click the database connection to log in.
3. In the top menu, go to:

   Server > Data Export

1. On the left, select the database.
2. On the right:

   o Under Export Options, choose: Export to Dump Project Folder o Click Browse and select where to save the folder o May leave the rest of the settings as default.

1. Click Start Export

   This will create a folder (e.g., database_name_dump) containing .sql files — one for each table. Data Import (Restore) using MySQL Workbench

1. In MySQL Workbench, go to: Server > Data Import
2. Under Import Options, choose: Import from Dump Project Folder
3. Click Browse and select the folder exported earlier.
4. Under Default Target Schema:

   o If your original database still exists, select it. o If not, click "New" to create a new schema (database) with the same name.

1. Check the box next to schema in the list.
2. Click Start Import

   Once done, tables should be restored. To Verify the Import Go to the Schemas panel: • Right-click the schema > Refresh All • Open a table and run: SELECT COUNT(*) FROM the table; Compare with the original row counts to make sure it restored correctly.

➢ Indexing and Performance Improvements

When working with large datasets, such as the 100,000 records generated and inserted into the `inventorydb` database, performance optimization becomes critical. One of the most effective ways to improve database performance is through indexing. Indexing allows the database to retrieve data more efficiently by creating a structured reference for specific columns. In this project, we implemented indexing strategies to optimize query performance and ensure the database could handle large-scale operations effectively.

Challenges with Performance

Initially, the database faced performance bottlenecks due to the large volume of data. Queries that filtered or joined tables, such as retrieving active discounts or finding suppliers by email, were slow because the database had to perform full table scans. This was especially problematic for tables like Discounts and Users, which contained 100,000 records each. Additionally, ensuring data integrity and avoiding duplicate entries for fields like username and email required careful consideration.

Indexing Strategies

To address these challenges, we added indexes to key columns in the database:

Primary Keys- Each table had a primary key to uniquely identify records. For example, `category_name` in the Categories table and `username` in the Users table were set as primary keys. This ensured fast lookups for unique records.

Foreign Key Indexes-In the Discounts table, we added an index on `product_id` to optimize joins with the Products table. This improved the performance of queries that linked discounts to specific products.

Filtering Indexes-We created an index on the `is_active` column in the Discounts table. This significantly improved the performance of queries that filtered active discounts, as the database could use the index to locate matching rows directly.

Unique Indexes- To ensure data integrity, we added unique indexes on the `email` column in both the Users and Suppliers tables. This not only prevented duplicate entries but also sped up queries that searched for records by email.

Performance Improvements

The impact of these indexing strategies was measurable and significant. For example:

- A query to retrieve all active discounts (`is_active = 1`) initially required a full table scan, taking several seconds to execute. After adding an index on `is_active`, the query execution time was reduced to milliseconds.
- Joins between the Discounts and Products tables became faster due to the index on `product_id`, allowing the database to locate matching rows efficiently.

- Searching for suppliers or users by email became instantaneous, thanks to the unique indexes on the `email` column.

To verify these improvements, we used the EXPLAIN command in MySQL to analyze query execution plans. Before indexing, the EXPLAIN output showed full table scans (type = ALL), indicating inefficient queries. After indexing, the output showed the use of indexes (type = ref or type = index), confirming that the database was leveraging the indexes to optimize performance.

Additional Optimizations

In addition to indexing, we implemented batch processing in the Python script to handle large data inserts. Instead of inserting all 100,000 records at once, we divided the data into smaller batches of 1,000 records. This reduced the load on the database and prevented timeouts or connection issues. We also ensured that generated data, such as usernames and emails, was unique by using sets to track existing values.

Indexing played a important role in improving the performance of the `inventorydb` database. By carefully selecting columns for indexing and leveraging unique constraints, we were able to optimize query execution times and ensure data integrity. These improvements made the database more efficient and scalable, capable of handling large datasets without performance degradation. This experience reinforced the importance of indexing and query optimization in database design, and we plan to apply these strategies to future projects.

➢ **Role Assignments and contribution summary**

Project lead: Rechelle Borbe

-As the leader, I initiated the project by adding all team members as collaborators to our shared repository. I ensured that everyone had access to the project files and could contribute effectively. I also facilitated discussions to assign roles and responsibilities, ensuring that each team member understood their tasks and deadlines. I worked closely with our backend developer, who provided the initial code that served as a reference for creating the database schema. I attempted to use an SQL schema script to define the database structure and generate data. However, I later discovered that using a Python script with the Faker library was more efficient for generating large datasets. I took the lead in learning how to install Python, connect the inventorydb database to Python, and integrate Visual Studio Code with MySQL Workbench. Despite being new to this process, I successfully connected the database to Python and implemented the Faker library to generate 100,000+ records per table. As the leader, I initiated the project by adding all team members as collaborators to our shared repository. I ensured the deployment and Star Schema, I led the development of Business Intelligence (BI)-oriented queries and the design of a star schema for data analysis. I collaborated with team members to create meaningful BI queries, such as identifying top-performing categories and analyzing user roles. I ensured that the star schema was designed to facilitate efficient querying and reporting. I coordinated the implementation of MySQL backup and recovery processes and the deployment of the database to the cloud with our SQL developer and backend developer. I ensured that

backups were created and tested for reliability and worked with the team to deploy the database to a cloud platform, ensuring accessibility and scalability. I ensured that all optimizations were documented and that the database was performing at its best. As the leader, I took charge of reviewing all project requirements to ensure everything was ready for submission. I worked closely with the team to verify that all deliverables were complete, accurate, and submitted on time. This included reviewing the database schema, Python scripts, BI queries, and documentation.

Database architect: Mhelarry Valeza

-In our Inventory Management System project, I worked as the database architect, where I was responsible for designing the database structure to ensure it was well-organized and efficient. I helped build features that made it easy to import and export data in and out of the system. When errors arose, I helped troubleshoot and fix issues to ensure the system remained reliable and functional.I also took on the task of editing and updating the data within the system. This included making necessary help in data displays and visualizations, such as graphs. I worked closely with the development team to make sure the database met the project's needs and expectations. In the process, I was able to hugely help our leader by providing reliable solutions and ensuring the system ran smoothly.

Database Architecture :Curt Justin Reodique

-As a member of our team, I actively contributed in several ways to ensure the success of our group tasks. I helped input important data into our project documents and assisted in gathering and submitting screenshot for each activity deliverable. One of my key roles was writing the reflection for Week 14, which gave me a chance to summarize our progress and learning.I ensured that the star schema was designed to facilitate efficient querying and reporting. I also kept our team updated by sharing relevant news and served as a support member for our database-related work. Through this experience, I gained valuable knowledge that I wouldn't have learned on my own, especially with the guidance and support of our team leader. I also played a part in encouraging and motivating my group mates, making sure we stayed on track and worked together as a team.

Backend Developer: Susaine Rico

-In our Inventory Management System project for Information Management, my main role was as a back-end developer. I worked on handling the data by creating features that allow importing and exporting of data to and from the system. I also created the dump file to help save and restore the database easily. adding indexes to improve query performance, and implementing batch processing in the Python script to handle large data inserts efficiently The original codes used in building the system came from me, and I made sure everything in the back-end works properly so that the system runs smoothly.

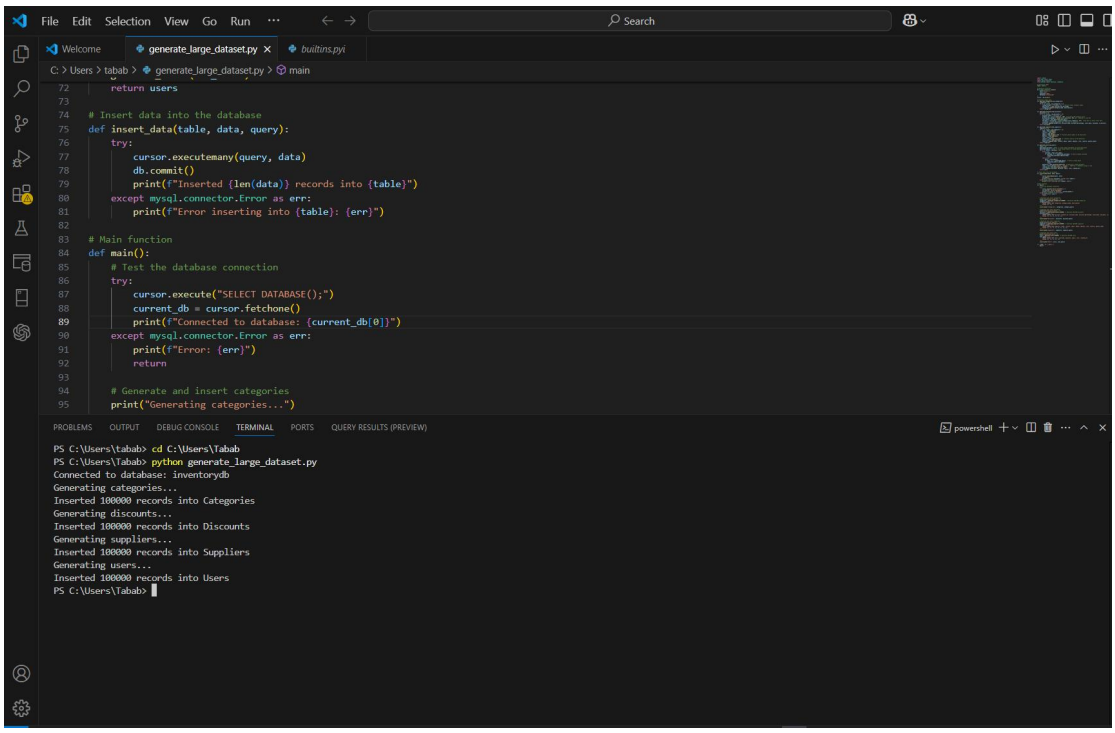
SQL Developer: Faith Sañado

-As a member of the team, I contributed by helping with the concept and structure of the database during the planning phase. I assisted in testing the tables and data to ensure everything was working correctly. I led the team i diagnosing and optimizing the database for performance. This involved analyzing query execution plans using the EXPLAIN command.I also did my best to help solve problems whenever we encountered errors, and tried my best to support them by doing research, watching tutorials, and looking for helpful information online. Aside from offering suggestions when needed, I worked hard to resolve issues related to connecting our database to Railway until we successfully established a connection. Most importantly, I remained committed to providing help to the team whenever it was needed.

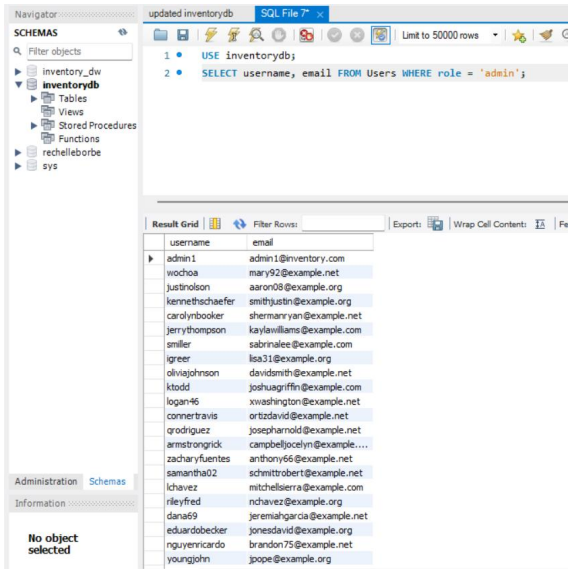
QA Tester: Shaine Sanjuan

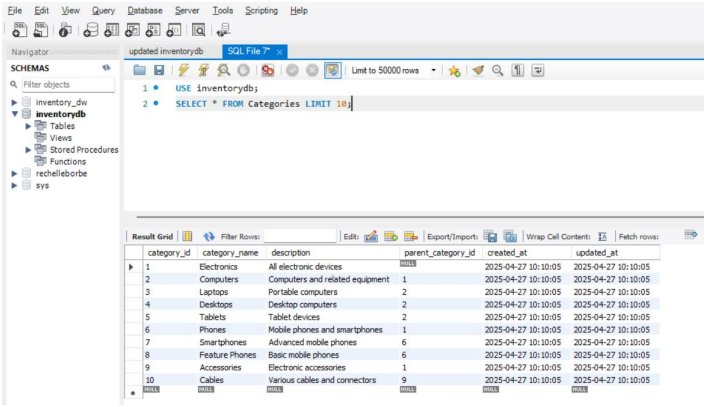
- As the QA Tester for our database project, I was responsible for thoroughly testing the SQL scripts and database queries to ensure they met the required specifications. I conducted multiple test runs to validate data integrity, enforce constraints, and confirm that transactions and queries performed as expected under various scenarios. Through trial and error, i expirience some difficulties of our code but rather than that we made a solution to fix and optimize it. Additionally, I wrote a reflection paper based on my testing experience. This paper outlines the challenges I encountered, the debugging strategies I used, and the key insights I gained about developing a reliable and effective database system. The reflection highlights how iterative testing helped improve the overall quality and performance of our database.

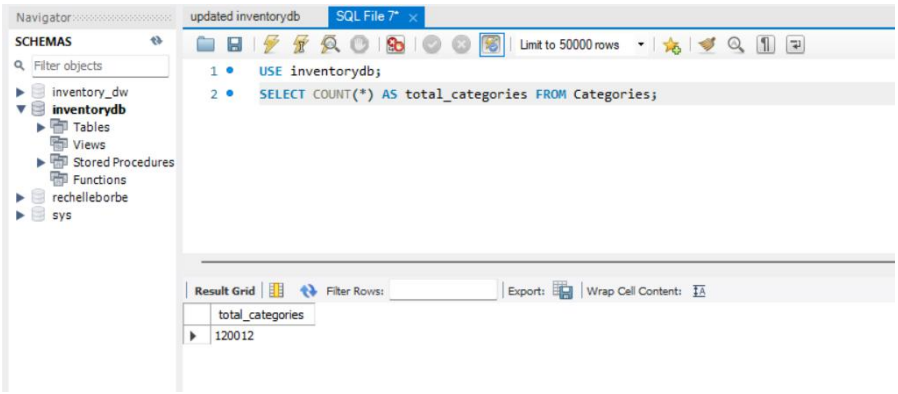
➢ Screenshot of output and Dashboard

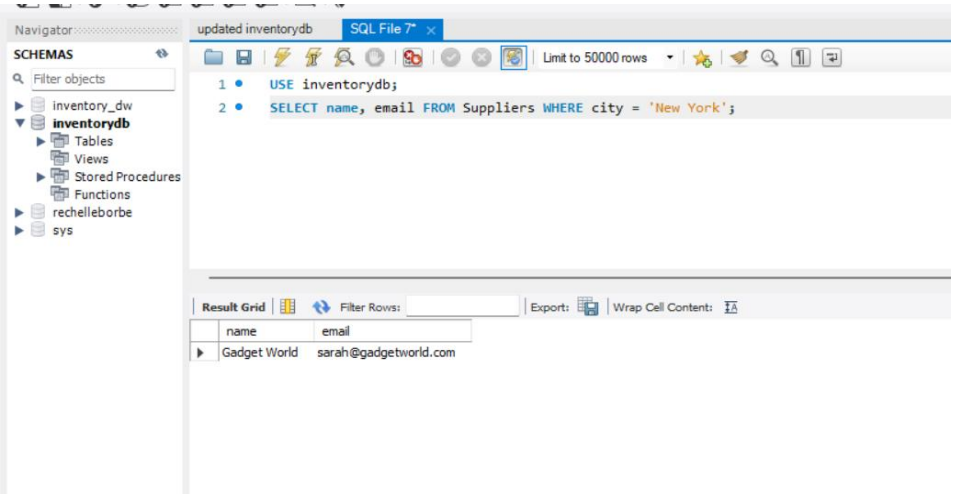


- **4 Basic Queries**: To demonstrate basic functionality.

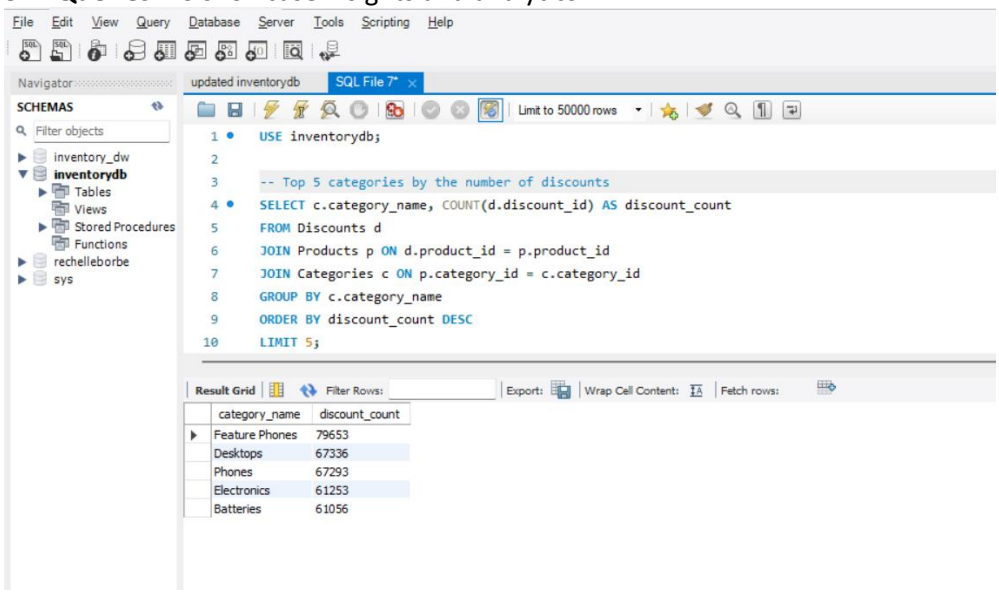


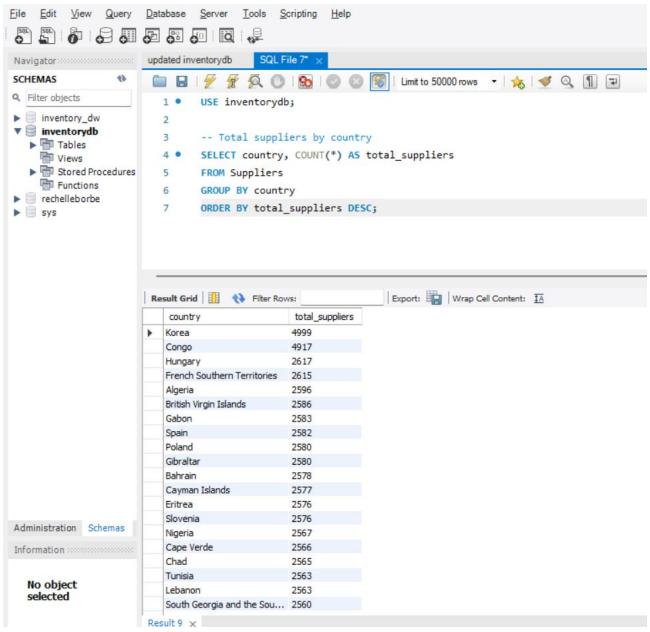


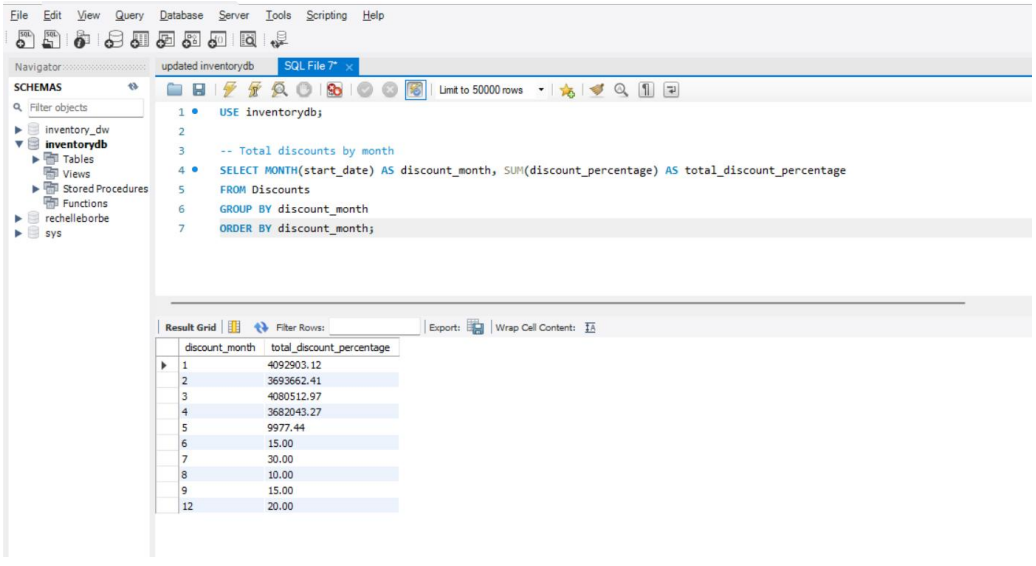




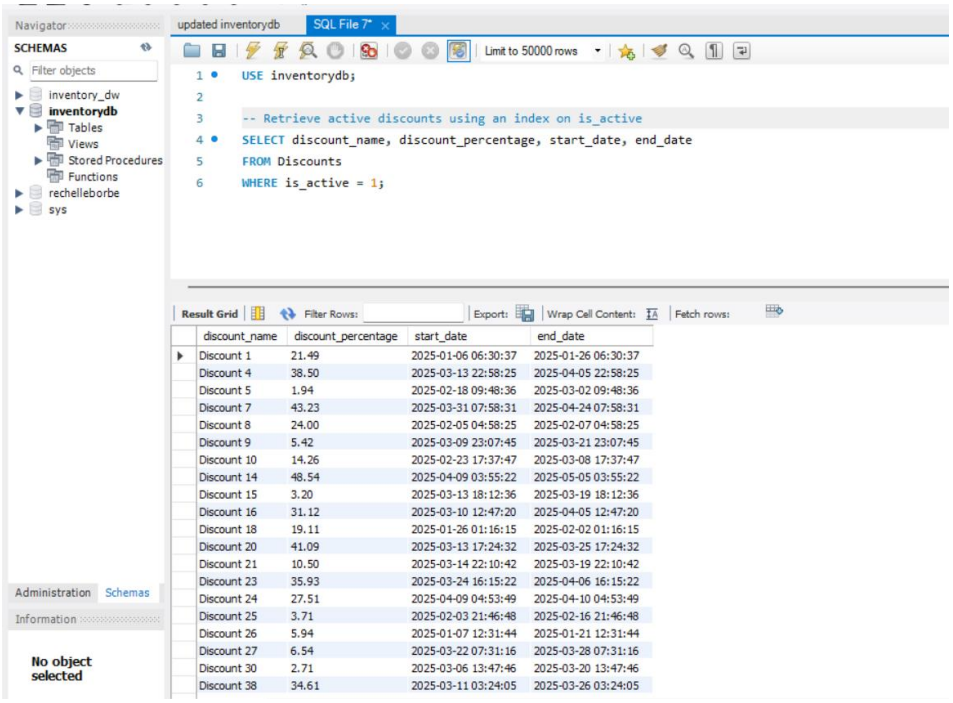
3 BI Queries: To showcase insights and analytics

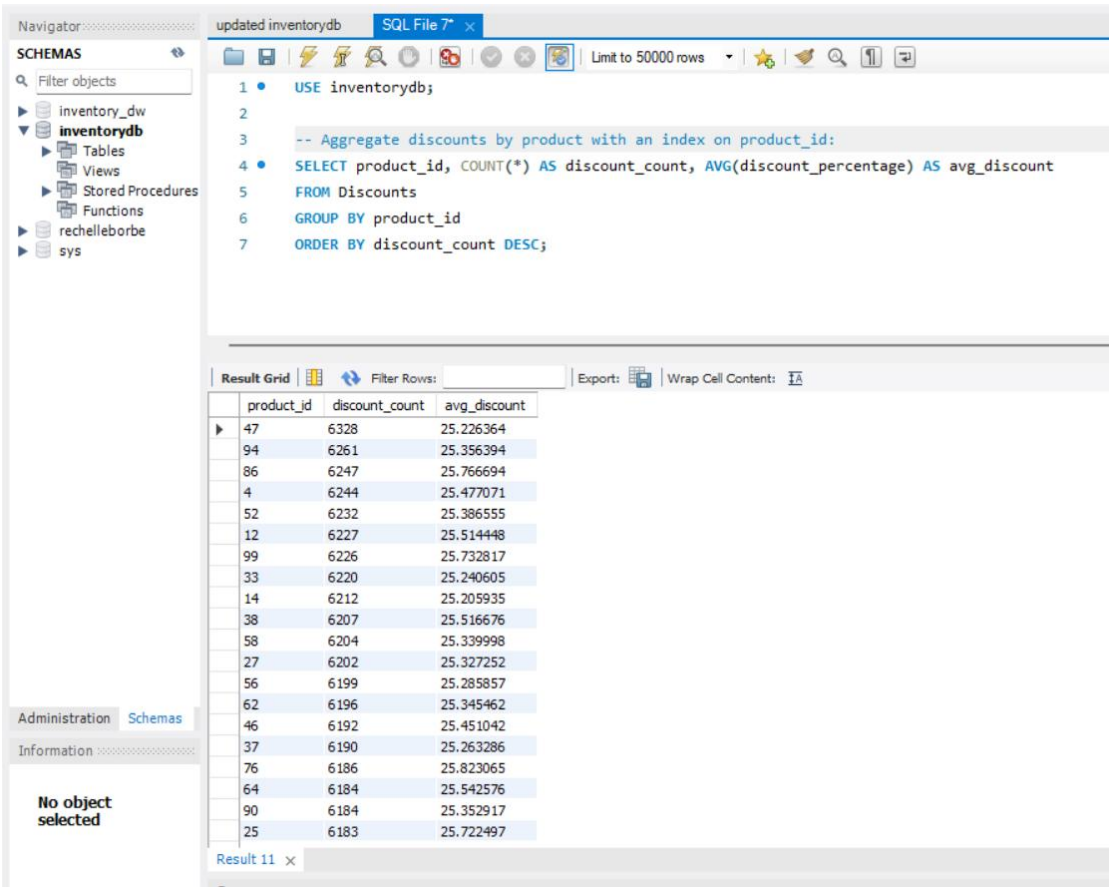




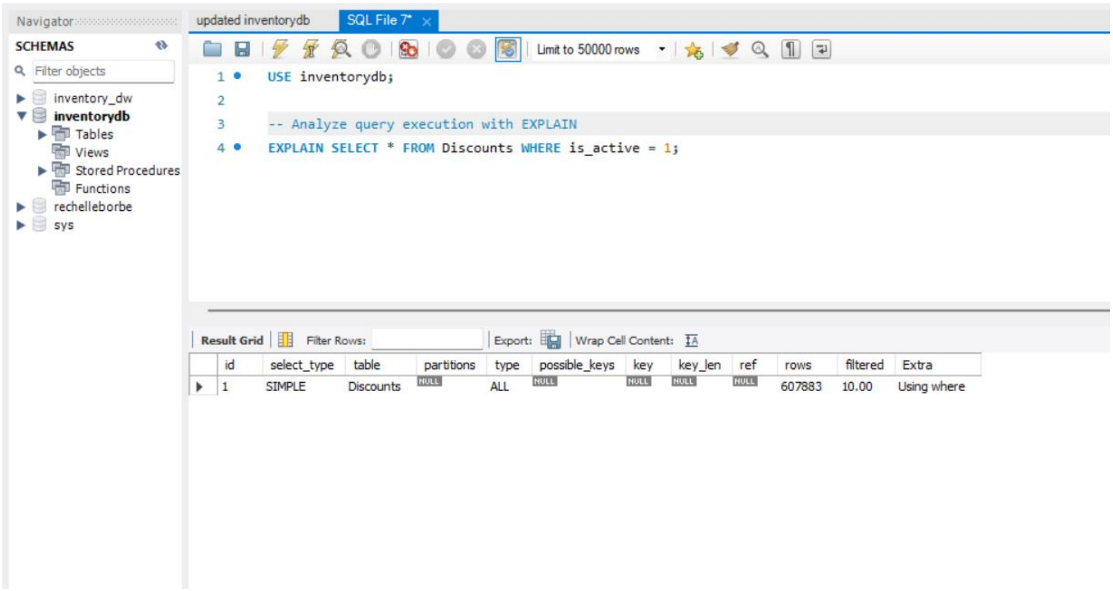


- **2 Optimized Queries**: To highlight performance improvements.





- **1 Diagnostic Queries (Optional)**: To analyze and validate performance.



➢ Reflection and Limitations

At first, we started our final project by assigning each group member a specific role. We split up the tasks so we could work faster and more efficiently. Everyone had their own part to focus on—one became the Project Lead to guide the overall progress, another was the Database Architect who designed the schema and structure, one took the role of SQL Developer to write queries and manage the database, another acted as the Back-End Developer to handle deployment and technical setups, and one worked as the QA Tester to check and verify that everything worked correctly.

As we moved forward, we faced many difficulties. At first, designing the schema and adding constraints was confusing. Importing the data took a lot of trial and error, and we often got stuck fixing small but frustrating problem, we also experience the difficulties on passing the deliverable on GitHub so we did a trial and errors so many times. One of the biggest struggles we had was with Railway during the cloud deployment. Setting up the cloud database, restoring backups, and making sure everything worked online took us a lot of time. There were issues we didn't expect, and we had to search for solutions and help each other figure things out. Backup and recovery were also difficult parts. Creating the backup, testing it, and proving that it worked was not as easy as we thought. We had to repeat the process several times before we got it right. Writing the BI queries and creating the star schema was another challenge. It was hard to fully understand how to transform the data into a format we could use for insights. But by working together and doing research, we were able to complete it. Query optimization was the last and hardest part. Learning how to use EXPLAIN and improve slow queries took a lot of effort. We had to compare the performance before and after making changes, and it took time to get it right. During the final phase of the project, we faced technical problems with our devices. Some tools and processes wouldn't work properly on our main laptops, and we couldn't continue. This was very stressful because the deadline was close.

In the end, we had to borrow or switch to another device just to execute the final steps and make sure everything worked. That extra step made things even harder, but it was the only way to complete the project. Even though we struggled through almost every part, we didn't give up. We supported each other and kept moving forward, step by step. In the end, we were able to submit all the requirements to GitHub and LMS SQL files, screenshot, documentation, and reports.

As a team,we are proud of what we've accomplished, and we know, we've done our best.

INVENTORY MANAGEMENT SYSTEM

MEMBERS

Project lead:Rechelle Borbe

Database architect: Mhelarry Valeza

Database Architecture :Curt Justin Reodique

Backend Developer: Susaine Rico

SQL Developer: Faith Sañado

QA Tester: Shaine Sanjuan