Music Store Database

***TEAM:* HARNISH MISTRY (8675724), KATHAN PARIKH (8687110)**

**Content**

1. Introduction
2. Group-work breakdown table
3. EER model
4. Table breakdown – Constraints, Datatype
5. Relationship among tables
6. Store Procedure & Trigger
7. Queries
8. Problems faced during the project
9. Conclusion
10. **Introduction**

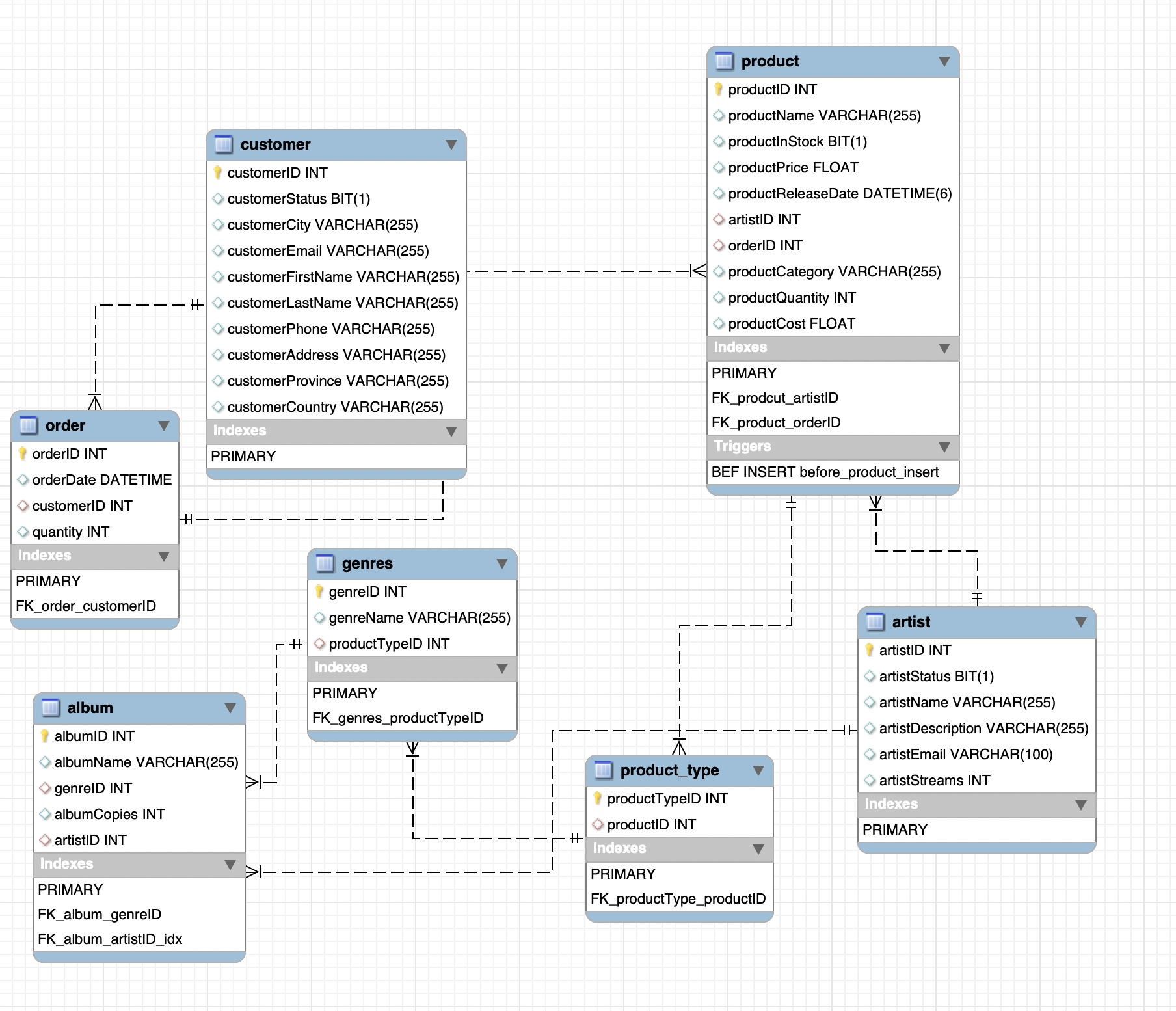
The primary objective of this project is to create a database and the appropriate views / functions / procedures / triggers based on a music store database system. Along with that, the whole database is created in such a way that it satisfies all the queries that are asked. The first step was to create an EER model which shows the overall idea about how the database would look like with all the tables and constraints. Once the EER model was completed, the tables were created, we had data to add. After adding dummy data, we started finding solutions to the queries. In the end, we created a store procedure and a trigger for the database as asked. That was the whole workflow behind creating this system.

1. **Group-work breakdown table**

It’s always a good idea to decide the individual roles and tasks when you’ll be working as a team. Here’s a table showing the tasks performed by us individually.

|  |  |
| --- | --- |
| ***Harnish Mistry*** | ***Kathan Parikh*** |
| Created tables of artist, customer, genre | Create tables of album, product, product\_type, order |
| Merged all the tables and added relationships, constraints into them | Generated SQL file for the database and added data into the tables |
| Created queries to answer the questions: 1 - 4 | Created queries to answer the questions 5 – 7 and 2 other questions |
| Created Trigger event for the database | Created store procedure for the database |
| Created the technical report for the project | Create presentation file for the project |
|  |  |

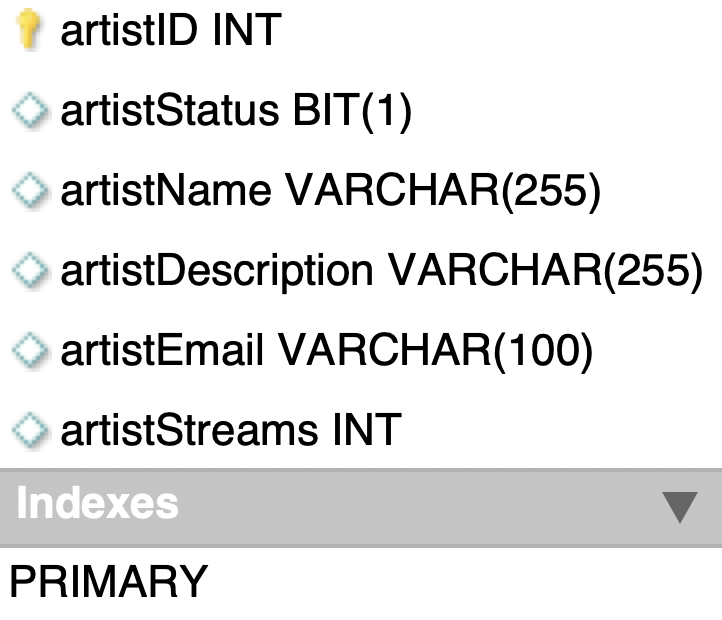
1. **EER Model**



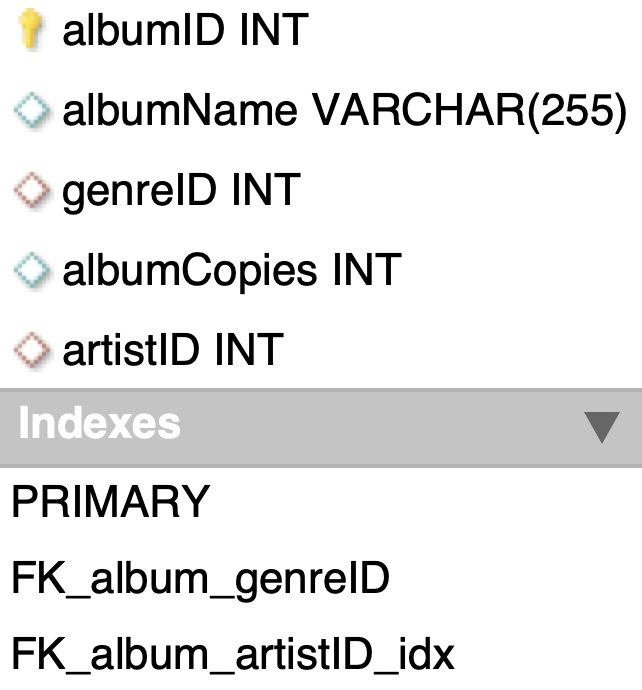
1. **Table breakdown – Constraints, Datatype**

The system consists of 7 tables named artist, album, product, product\_type, genres, order, customer. Each table is described briefly below.

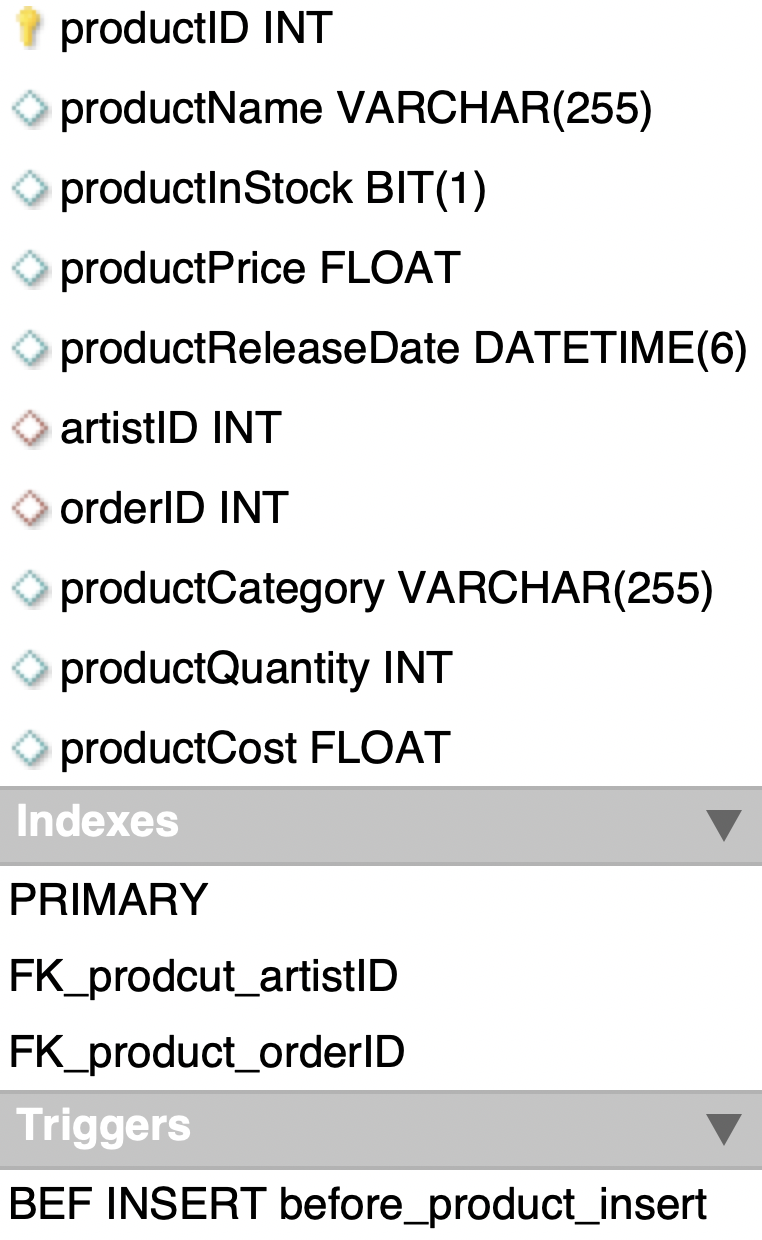
***Artist:***This table contains all the basic information about the artist himself like, ID, name, status, description, email, number of streams. The datatype for each entity is described below.



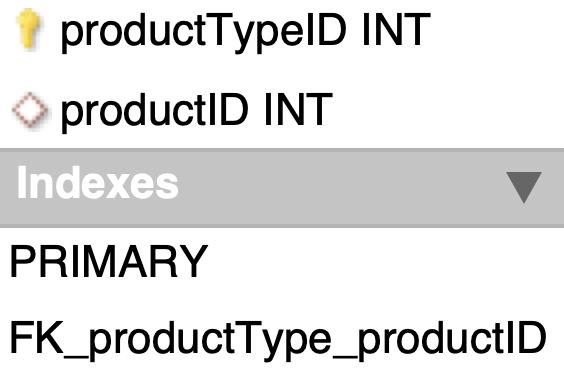
***Album:***This table contains all the details about the album like, ID, name, status, description, email, number of streams. The datatype for each entity is described below.



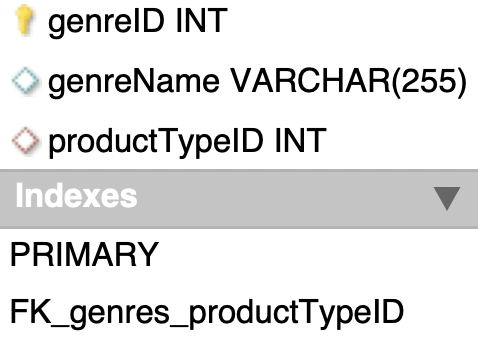
***Product:***This table contains all the details about the product like, ID, name, stock, price, release date, category, quantity, etc. Each entity is described below.



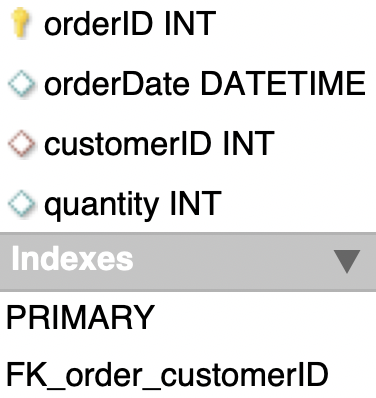
***Product\_type:***This table contains only two fields about the product type. One is the primary key itself and another is the foreign key for product ID.



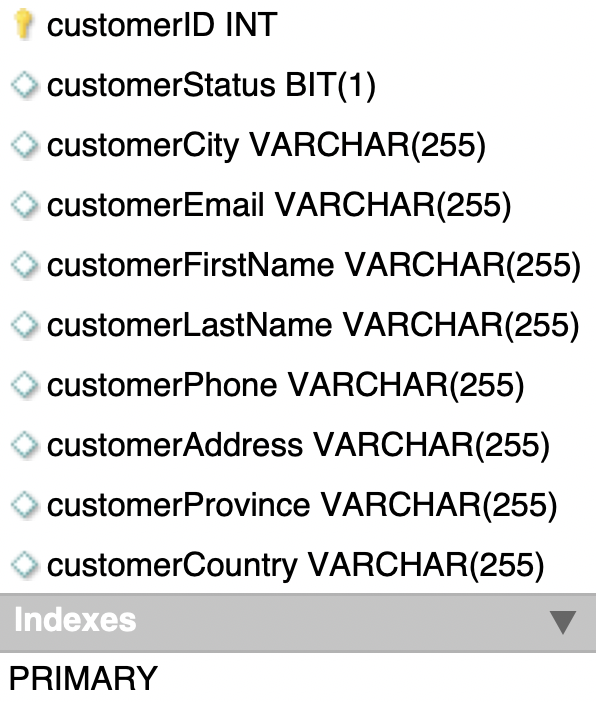
***Genres:***This table contains all the details about the genre like, ID, name, product type where product type is the foreign key.



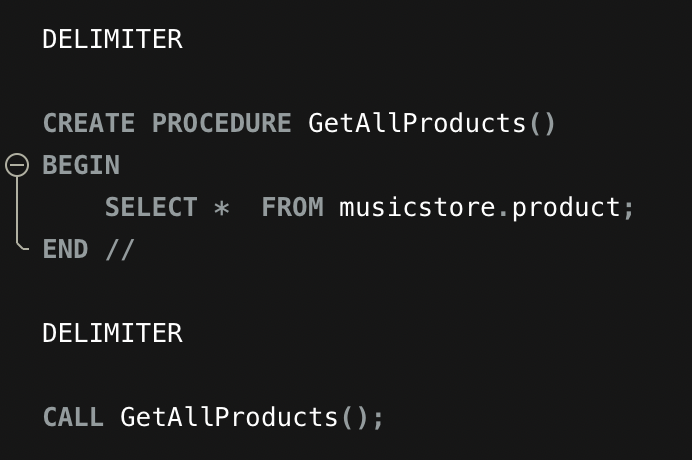
***Order:***This table contains all the details about the order like, ID, date, customer ID, quantity where customer ID is the foreign key.

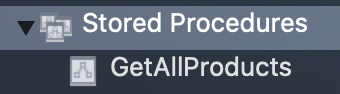


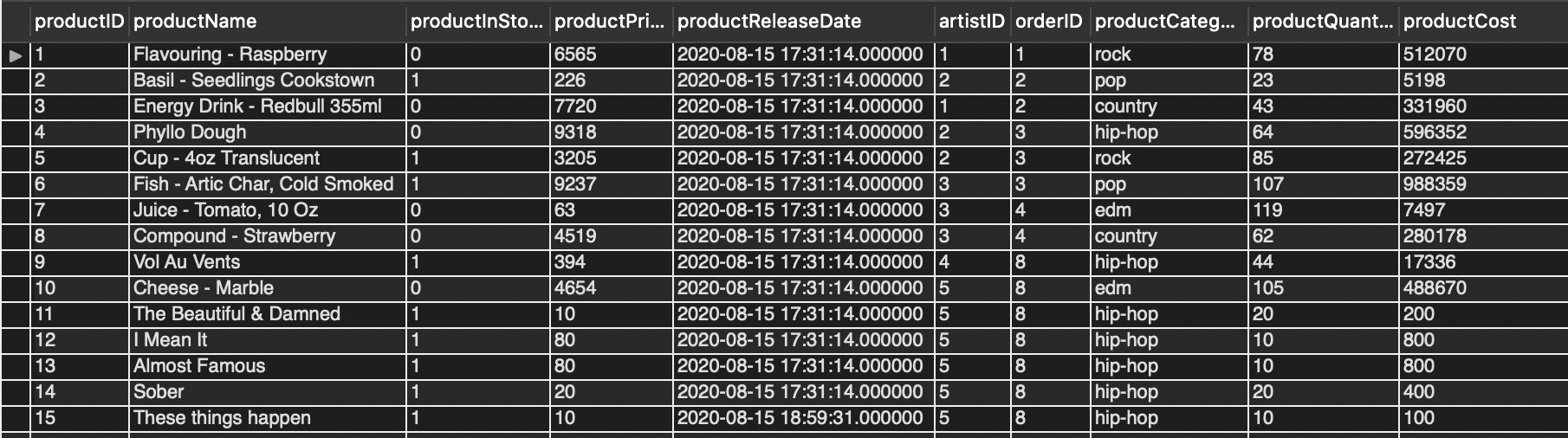
***Customer:***This table contains all the details about the customer like, ID, status, city, email, first name, last name, phone, address, province, country where customer ID is the primary key.



1. **Relationship among tables**
   1. **order-to-product:** order to product is having one to many relationships as the order can be of one of more products.
   2. **customer-to-order:** customer to order too has one to many relationships as any customer can have one of more orders.
   3. **product-to-product\_type:** product to product\_type is having one to many relationships as any product can be of one or more type.
   4. **artist-to-product:** this has also one to many relationships as one artist can have one or more products.
   5. **artist-to-album**: artist to album is having one to many relationships as a single artist can have one or more albums.
   6. **genres-to-album:** genres to album too has one to many relationships as many albums may have same genre.
2. **Stored Programs:**
   1. **MySQL Procedure for getting all the products.**

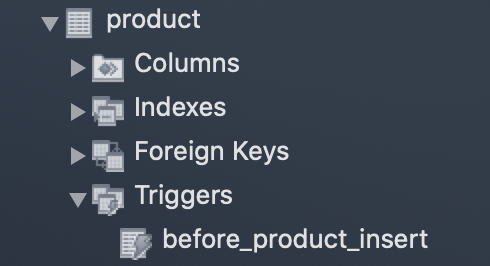


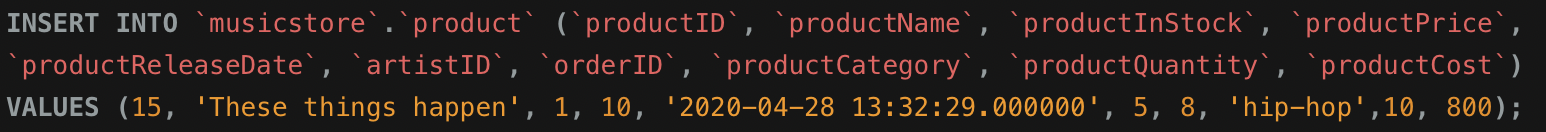




* 1. **MySQL trigger before product insertion to update the Product cost.**

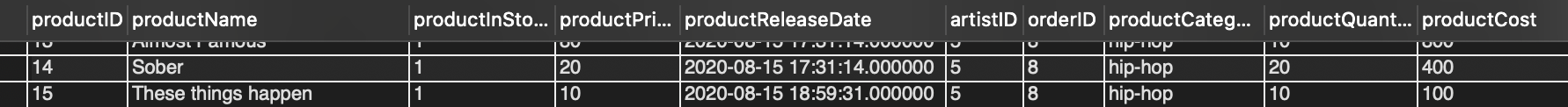




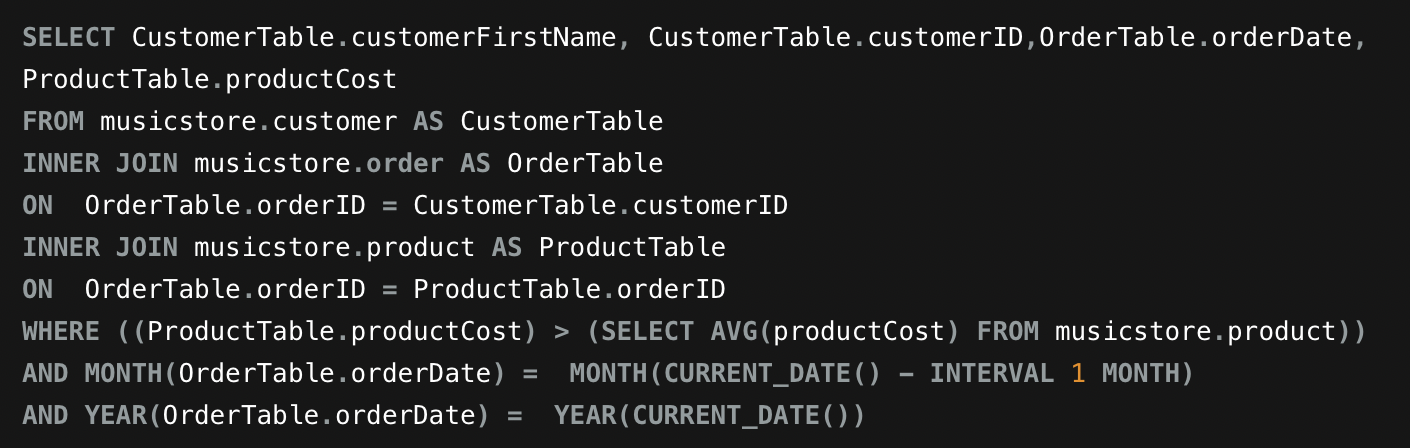


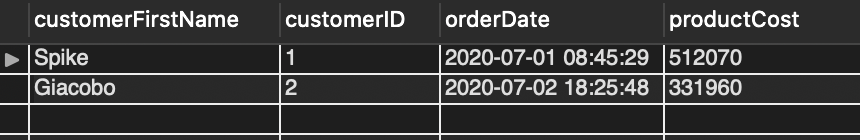
* + Here, if you look into the query, you can see that the ‘productReleaseDate’ is not the current date and the time as well. Along with that, ‘productQuantity’ and ‘productPrice’ are the same that is ‘10’ with the ‘productCost’ of ‘800’.

But, when the user will enter the same data, the trigger will be fired and the new ‘productCost’ will be 10 x 10 that is ‘100’.



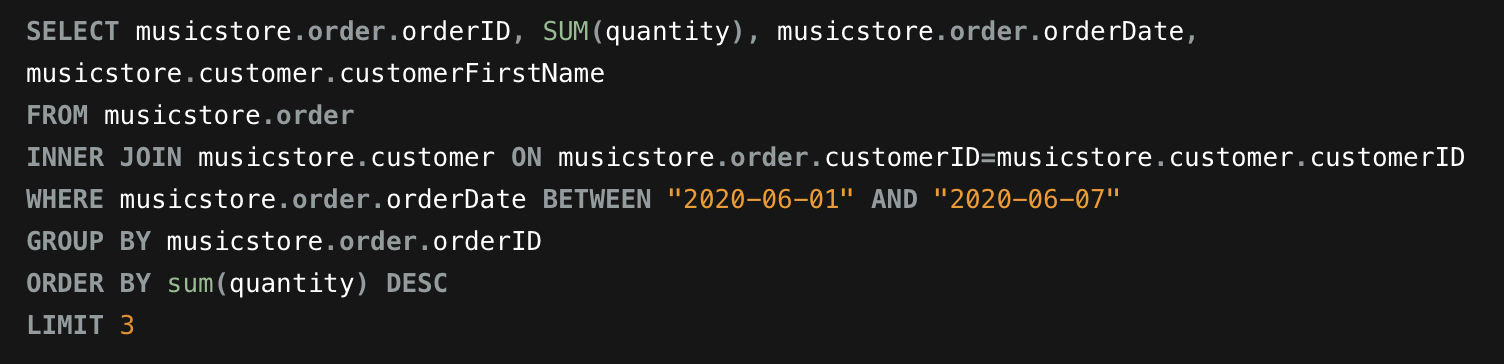
1. **Queries**
   1. Display a list of clients that spent more than the average spent by client in the past month.

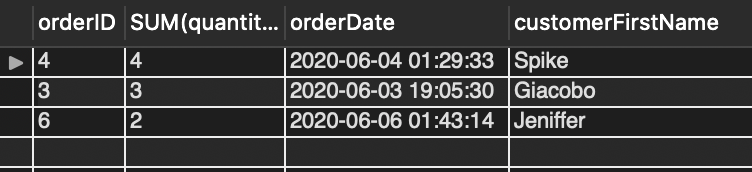




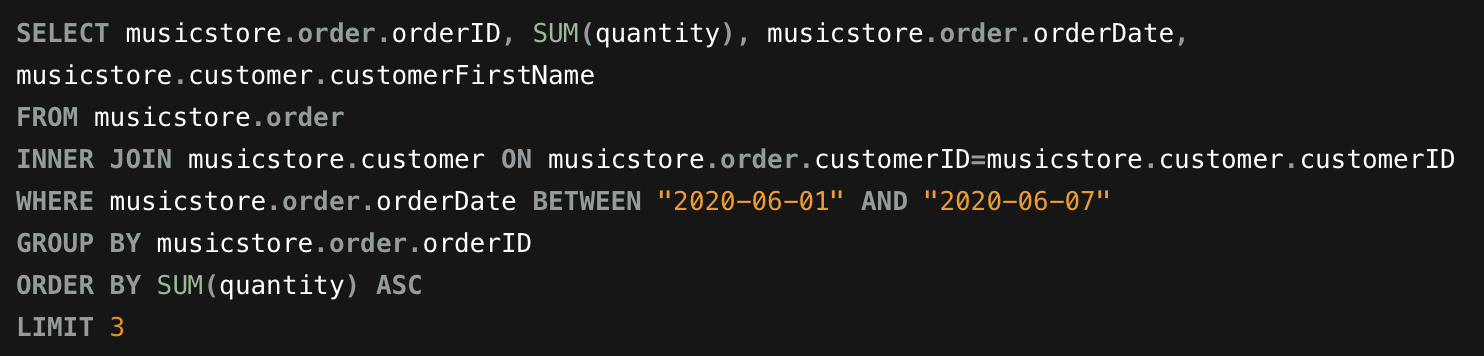
* 1. The top sold products and least sold products over a week.

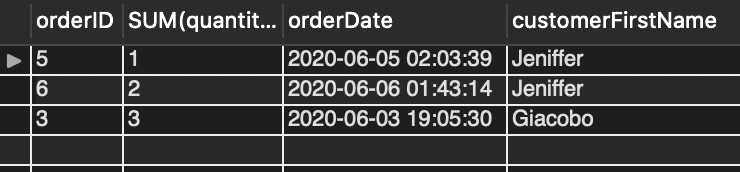
(Top sold products)



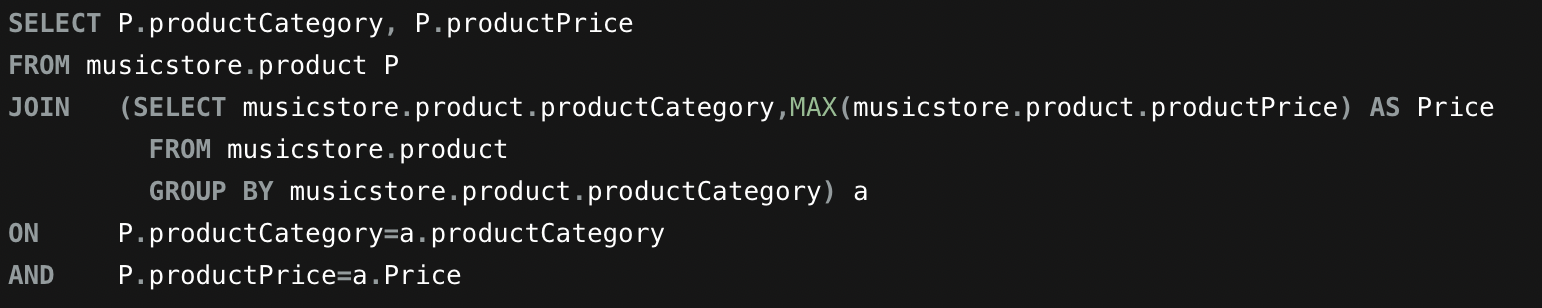


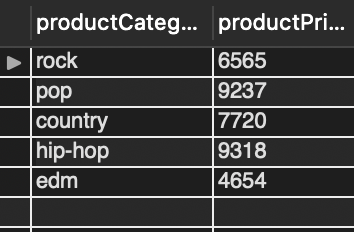
(Least sold products)



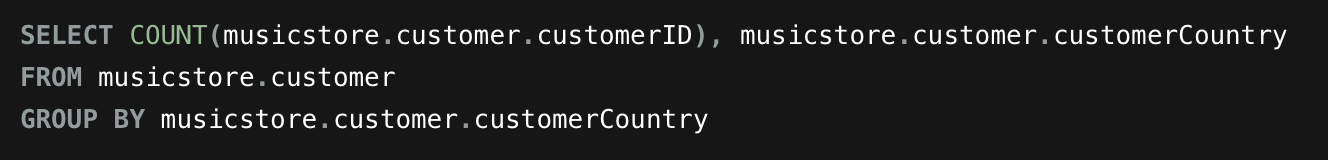


* 1. The top sold products and least sold products over a week.



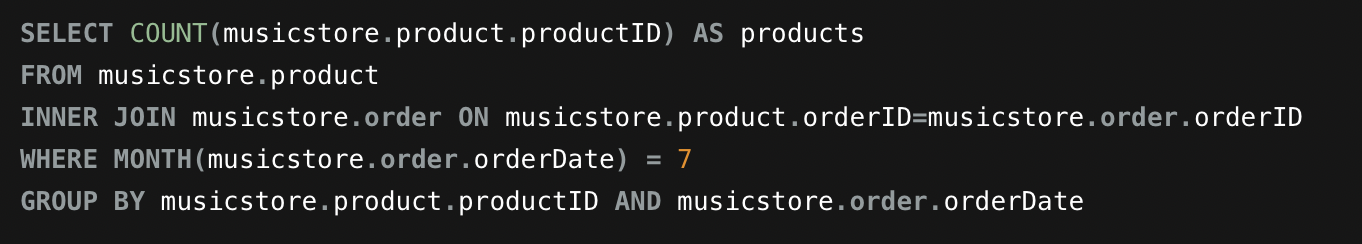


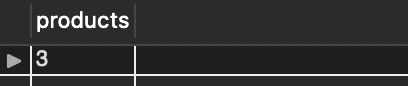
* 1. List how many customers the system has by location (Country, Province, and City), and then sort them.



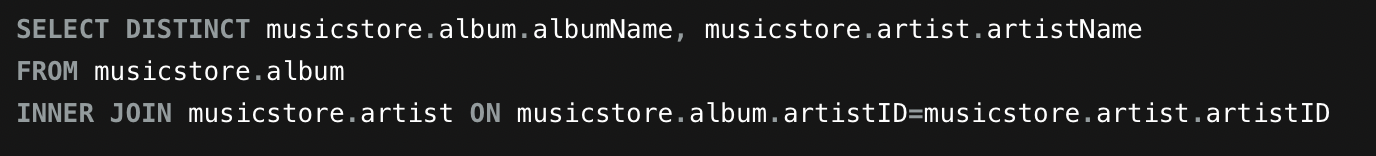


* 1. List how many products the store has sold for a particular month.



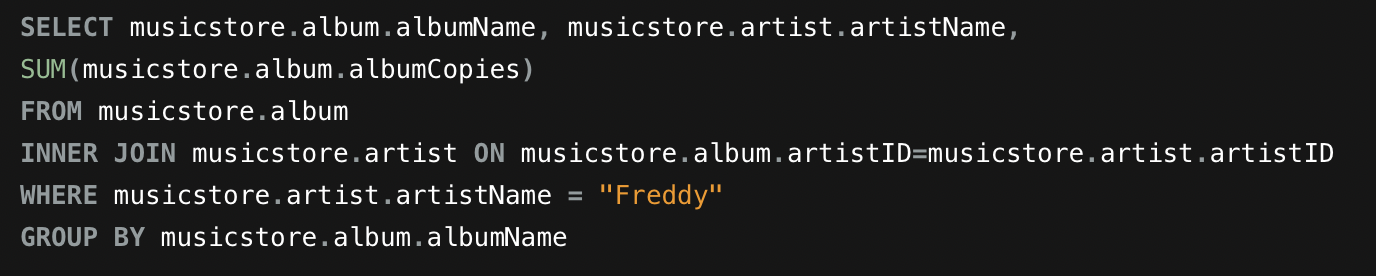


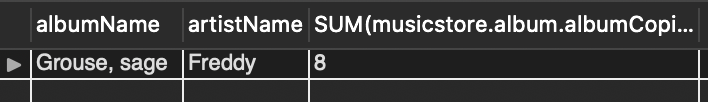
* 1. List how many distinct albums each singer has.





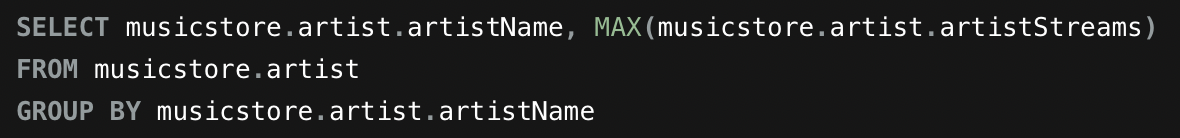
* 1. List how many copies of an album are available of a particular singer.

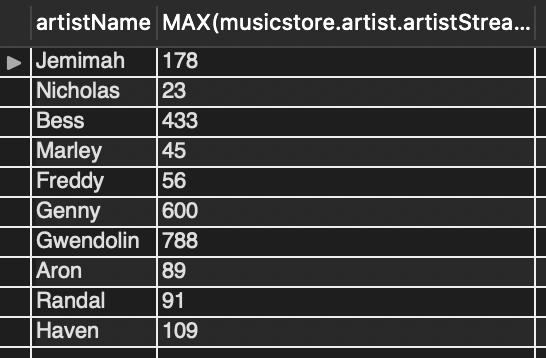




* Specific Scenario: A list of 2 other questions a user of this system might want to ask, together with a list of MySQL queries that help to answer them.

1. Find which artist has maximum number of streams.





1. Find albums for each genre.





1. **Problems faced during the project:**
   * Planning Database: Planning the database and the tables was a little bit tricky as we had to plan it in a particular way so it contains the answers to all the queries.
   * Writing Queries: We tried to create the database queries accordingly but still at some points, we had to make changes in database a little bit so we can have the results without losing the relationships between the tables.
2. **Conclusion:**

The primary goal was to create the whole music database system which also satisfies the queries along with the store procedure and a trigger. We had to face a couple of problems as we were working on it individually but as a team, still after overcoming all the difficulties, we learned a lot and tried our best to reach the goal.