Database Design and Maintenance Final Group Project

Music Store Database

Group Members:

Ashwin Kumar Davley Suresh Rao,

Ishwarya Vijaykrishnan

Sanjana Surendran

Table of Contents

[TABLE OF FIGURES: 3](#_Toc88637543)

[OBJECTIVE: 4](#_Toc88637544)

[EER MODEL: 5](#_Toc88637545)

[TABLE BRAKEDOWN: 6](#_Toc88637546)

[SCENARIOS: 13](#_Toc88637547)

[TECHNICAL DIFFICULTIES 23](#_Toc88637548)

# **TABLE OF FIGURES:**

[Figure 1Music Store DB - EER model 5](#_Toc88675019)

[Figure 2 - City 6](#_Toc88675020)

[Figure 3 - City 6](#_Toc88675021)

[Figure 4 – Customer 7](#_Toc88675022)

[Figure 5 - Discount 7](#_Toc88675023)

[Figure 6 - Gender 8](#_Toc88675024)

[Figure 7 – Gender 8](#_Toc88675025)

[Figure 8 - Language 9](#_Toc88675026)

[Figure 9 - Order 9](#_Toc88675027)

[Figure 10 - Product 10](#_Toc88675028)

[Figure 11 - Province 10](#_Toc88675029)

[Figure 12 - Singer 11](#_Toc88675030)

[Figure 13 – product Audit Table 12](#_Toc88675031)

[Figure 14 – Query 1 13](#_Toc88675032)

[Figure 15 – Query 2 14](#_Toc88675033)

[Figure 16 – Query 3 14](#_Toc88675034)

[Figure 17 – Query 4 15](#_Toc88675035)

[Figure 18 – Query 5 16](#_Toc88675036)

[Figure 19 – Query 6 17](#_Toc88675037)

[Figure 20 – Query 7 18](#_Toc88675038)

[Figure 21 – Trigger to Update the Available Columns automatically 19](#_Toc88675039)

[Figure 22 - Automatic update of the Product Audit table 19](#_Toc88675040)

[Figure 23 - Scheduled Weekly Backup of the Database 21](#_Toc88675041)

# **OBJECTIVE:**

Create a Music store Database to accommodate the stores inventory, sales, customer records and maintain seamless backup strategies.

Few of our teams goals is to satisfy the below requirements:

* Maintain Customer Records and purchase details
* Update inventory as per sales
* Easy one stop function to fetch available records, discount reports.
* Maintain weekly database backup.
* Track inventory update using an audit table.

We have tried to incorporate as much as learnings we have gathered throughout the course.

* The database must be normalized to 3rd normal form.
* Make sure to include primary/foreign keys where appropriate.
* Appropriate usage of Triggers, events, Stored procedure, Functions for easy data retrieval and updates(including dynamic updates).
* Determine the tables and relationships and generate EER diagram
* Generate SQL script from the EER model to create the database

# EER MODEL:

The below EER diagram provides you with all the elements and its attribute or relationship inheritances for the Music Store Database.

Diagram

Description automatically generated

Figure 1Music Store DB - EER model

# **TABLE BRAKEDOWN:**

**1. Table Name : City**

**Columns:**

City id - ID

City Name. - Name of City

Graphical user interface, text, application

Description automatically generated

Figure 2 - City

**2.Table Name :Country**

**Columns:**

Country Id - ID

Country Name - Name of Country

Graphical user interface, text, application

Description automatically generated

Figure 3 - City

**3.Table Name :Customer**

**Columns:**

Customerid - ID

Name - Name of Customer

mobile number - Customer mobile number

email address - Customer Email Address

city - Customer Resident city (FK)

province - Customer Resident Province (FK)

gender - Customer Gender (FK)

Graphical user interface, text, application

Description automatically generated

Figure 4 – Customer

**4.Table Name :Discount**

**Columns:**

Discountid - ID

Product - Album Product ID (FK)

Promocode - Discount code

Isactive - Is the discount active now

Startdate - Start date for the discount

Enddate - End date for the discount

Discountamount – Amount

Graphical user interface, text, application, email

Description automatically generated

Figure 5 - Discount

**5.Table Name :gender**

**columns:**

Genderid - ID

Gendername - Name of Gender

Graphical user interface, text, application, email

Description automatically generated

Figure 6 - Gender

**6.Table Name :genre**

**columns:**

Genreid - ID

Genrename - Name of Genre

Graphical user interface, text, application

Description automatically generated

Figure 7 – Gender

**7.Table Name :Language**

**Columns:**

Languageid - ID

LanguageName - Language name

Graphical user interface, text, application

Description automatically generated

Figure 8 - Language

**8.Table Name :Order**

**Columns:**

Orderid - ID

Customerid - Customer ID (FK)

Product - Product ID (FK)

Purchaseamount - Purchase Amount

Discount - Discount Amount

Purchasedate - Purchase Date

Graphical user interface, text

Description automatically generated

Figure 9 - Order

**9.Table Name :products**

**columns:**

Productid - ID

AlbumName - Name of the Album

Genre - Genre ID (FK)

Price - Price of the Album

Singer - Name ID (FK)

Language - Language ID (FK)

Available copies -Copies available in the store

Isactive - Is the product available for sales

Createdby - created by the user

Createddate - Created date

modified date - Modified date

modified by - Modified by user

Graphical user interface, table

Description automatically generated

Figure 10 - Product

**10.Table Name :Province**

**Columns:**

Provinceid - ID

Name - Name of the province

Country - Country under which the Province is present

Graphical user interface, text, application

Description automatically generated

Figure 11 - Province

**11.Table Name :Singer**

**Columns:**

Singerid - ID

Firstname - Singers First name

Lastname - Singers Last name

Gender -Gender of the Singer (FK)

Graphical user interface, text, application

Description automatically generated

Figure 12 - Singer

**12:Table Name :Productaudit**

**Columns:**

Productid\_audit - ID

Product - Album id

Albumname - Name of the Album

Genre - Album Genre

Price - Price of the album

Singer - Singer ID

Language -Language ID

Availablecopies -Available copies in the store

Modifiedby -Modified By

Modifieddate -Modified date

Graphical user interface

Description automatically generated

Figure 13 – product Audit Table

TABLE BREAKDOWN INDIVIDUALLY:

TABLES:

* Customer
* City
* Province
* Gender

Customer:

Columns:

* Customer id :

ID of the customer

Primary key

Auto Increment

Integer value

* Name :

Name of Customer

Varchar

* mobile Number:

Customer mobile number

Varchar

* email Address:

Customer Email Address

Varchar

* city :

Customer Resident City (FK)

Integer

In relation with base table city, instead of changing in main table, could add the values in base table and add that integer province id to it.

* Province:

Customer Resident Province (FK)

Integer

In relation with base table province, instead of changing in main table, could add the values in base table and add that integer province id to it.

* Gender:

Customer Gender (FK)

Integer

In relation with base table Gender, instead of changing in main table, could add the values in base table and add that integer gender id to it.

In this Music store database project we just ask the details of the user and store it for future reference and send them the mails and texts regarding the billing and discount and offers.

* City

Columns:

* City id :

ID of the city

Primary Key

Integer

* City Name:

Name of City

Varchar

* Country

Columns:

* Country Id

ID of the country

Primary Key

Integer

* Country Name

Name of Country

Varchar

* Gender

Columns:

* Genderid

ID of the gender table

Primary Key

Integer

* Gendername

Name of Gender

Varchar

# **SCENARIOS:**

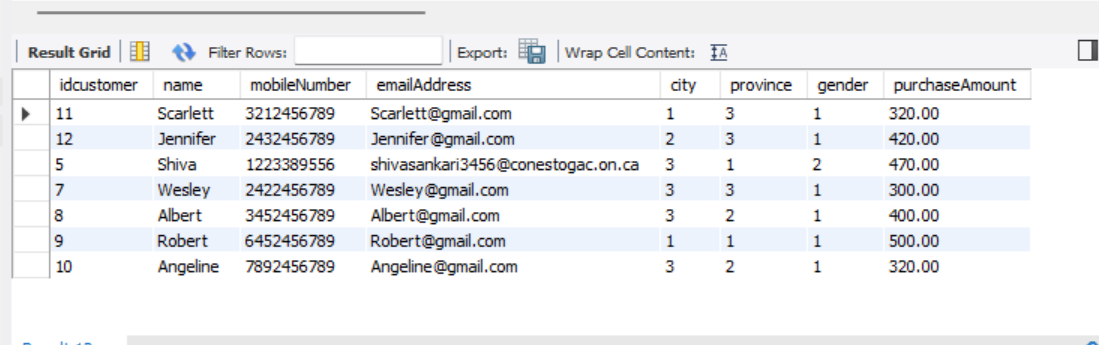
The below are few scenarios that helps to capture the given queries and additional automatic update.

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

QUERY:

|  |
| --- |
| select c.\*,O.purchaseAmount  from `order` O  inner join customer C on C.idcustomer = O.customerid  where O.purchaseAmount > (select P.AverageAmount from (select AVG(purchaseAmount) AS AverageAmount,  DATE\_FORMAT(purchaseDate, "%M-%Y") AS PurchaseMonth  from `order`  where MONTH(purchaseDate) = MONTH(CURRENT\_DATE - INTERVAL 1 MONTH)  group by DATE\_FORMAT(purchaseDate, "%M-%Y")) AS P); |

Figure 14 – Query 1

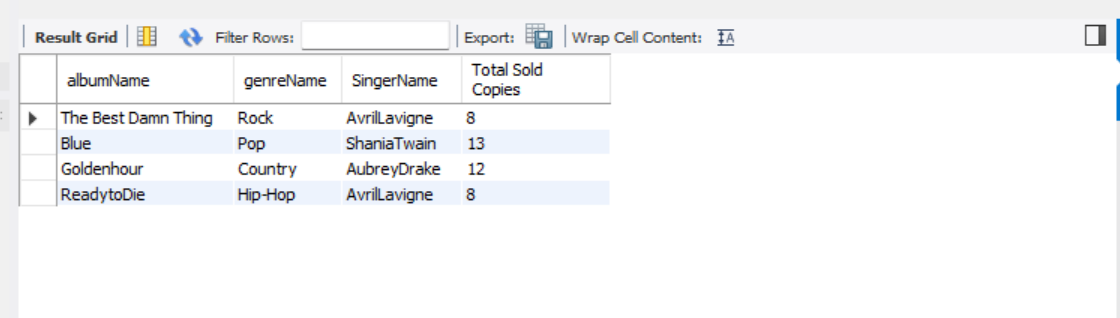


* The top sold products and least sold products over a week.

QUERY:

|  |
| --- |
| SELECT P.albumName,G.genreName,CONCAT(S.firstname, S.lastname) AS SingerName  ,SUM(OO.Count) as `Total Sold Copies`  FROM products P  INNER JOIN (SELECT \* FROM (SELECT COUNT(O.product) AS Count, O.product  FROM `ORDER` O  WHERE DAY(O.purchaseDate) BETWEEN (DAY(O.purchaseDate) - 7) AND DAY(O.purchaseDate)  GROUP BY O.product) AS O) OO on OO.product = P.idproducts  INNER JOIN genre G ON G.idgenre = P.genre  INNER JOIN singer S ON S.idsinger = P.singer  GROUP BY P.albumName,G.genreName,CONCAT(S.firstname, S.lastname); |

Figure 15 – Query 2

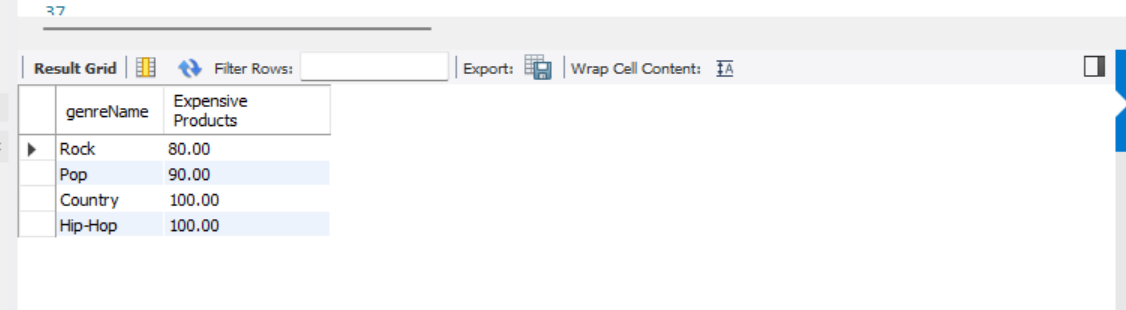


* The maximum price of products in the same genre (for example, rock, pop, country, hip-hop). Use GROUP BY to list all the genres and their maximum price.

QUERY:

|  |
| --- |
| SELECT G.genreName,MAX(P.price) AS `Expensive Products`  FROM products P  INNER JOIN genre G ON G.idgenre = P.genre  GROUP BY G.genreName |

Figure 16 – Query 3



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

QUERY:

|  |
| --- |
| SELECT COUNT(IDCUSTOMER) AS "CUSTOMERS PER LOCATION",  C.NAME AS CITY, PR.NAME AS PROVINCE, COU.NAME AS COUNTRY FROM CUSTOMER CU JOIN CITY C ON C.IDCITY = CU.CITY JOIN PROVINCE PR ON PR.IDPROVINCE = CU.PROVINCE JOIN COUNTRY COU ON COU.IDCOUNTRY = PR.COUNTRY GROUP BY CITY ORDER BY CITY, PROVINCE,COUNTRY |

Graphical user interface, application

Description automatically generated

Figure 17 – Query 4

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

QUERY:

|  |
| --- |
| SELECT PRODUCT, COUNT(\*) AS TOTALNOOFPRODUCTS, EXTRACT(MONTH FROM PURCHASEDATE) AS MONTH, monthname(PURCHASEDATE) AS MONTHNAME FROM MUSIC\_STORE.ORDER ORD JOIN MUSIC\_STORE.PRODUCTS PR ON PR.IDPRODUCTS=ORD.PRODUCT GROUP BY MONTH(PURCHASEDATE) |

Graphical user interface, application, email

Description automatically generated

Figure 18 – Query 5

* List how many distinct albums each singer has.

FUNCTION:

|  |
| --- |
| SELECT CONCAT(S.FIRSTNAME, ' ', S.LASTNAME) AS SINGERNAME, COUNT(DISTINCT ALBUMNAME) AS total\_no\_albums FROM PRODUCTS P JOIN SINGER S ON P.SINGER=S.IDSINGER GROUP BY IDSINGER ORDER BY SINGERNAME |

Graphical user interface, text, application

Description automatically generated

Figure 19 – Query 6

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

QUERY:

|  |
| --- |
| DELIMITER //  DROP FUNCTION IF EXISTS available\_copies; CREATE FUNCTION available\_copies ( singer\_name VARCHAR(100) ) RETURNS INT NOT DETERMINISTIC READS SQL DATA BEGIN DECLARE no\_of\_copies INT; SELECT SUM(avaliableCopies) INTO no\_of\_copies FROM music\_store.products WHERE music\_store.products.singer = (SELECT idsinger FROM singer WHERE CONCAT(music\_store.singer.firstName, ' ', music\_store.singer.lastName) = singer\_name);  RETURN no\_of\_copies; END//  SELECT P.albumName AS ALBUMNAME  ,SUM(available\_copies(CONCAT(S.firstName, ' ', S.lastName))) as AVAILABLECOPIES  FROM singer S  INNER JOIN products P ON P.singer = S.idsinger  WHERE idsinger = 1  GROUP BY p.albumname |

Graphical user interface, text, application

Description automatically generated

Figure 20 – Query 7

* Automatic Update of the Available copies with respect to the Order update.

TRIGGER:

|  |
| --- |
| DELIMITER // DROP TRIGGER IF EXISTS update\_pending\_copies;  CREATE TRIGGER update\_pending\_copies AFTER INSERT ON music\_store.order FOR EACH ROW BEGIN DECLARE product\_id INT; DECLARE available\_copies INT;  SET product\_id = NEW.product;  SELECT avaliableCopies INTO available\_copies FROM music\_store.products WHERE idproducts = product\_id;  UPDATE music\_store.products SET music\_store.products.avaliableCopies = available\_copies -1 WHERE idproducts = product\_id; END// |

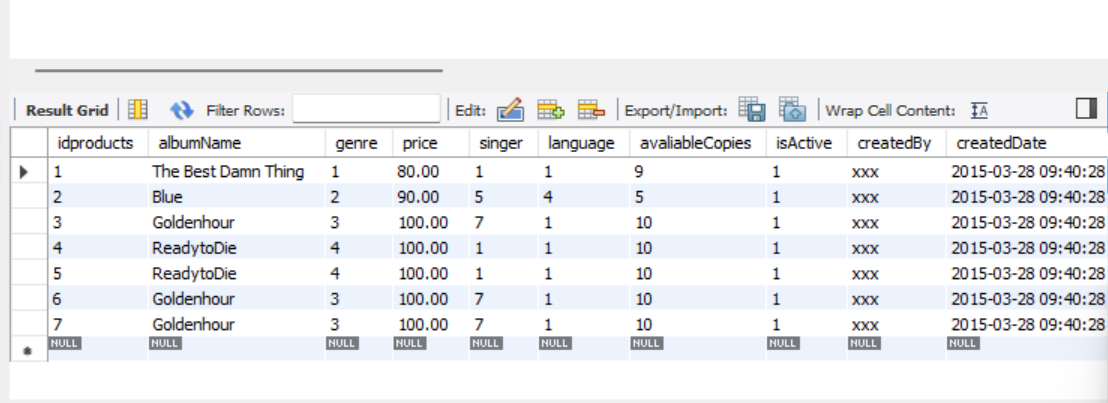
Figure 21 – Trigger to Update the Available Columns automatically

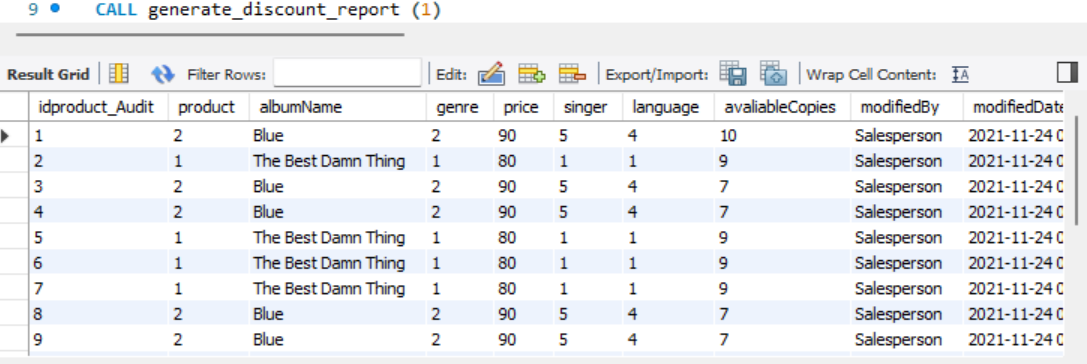
* Automatic update of the Product Audit table for every update occurring in the product table.

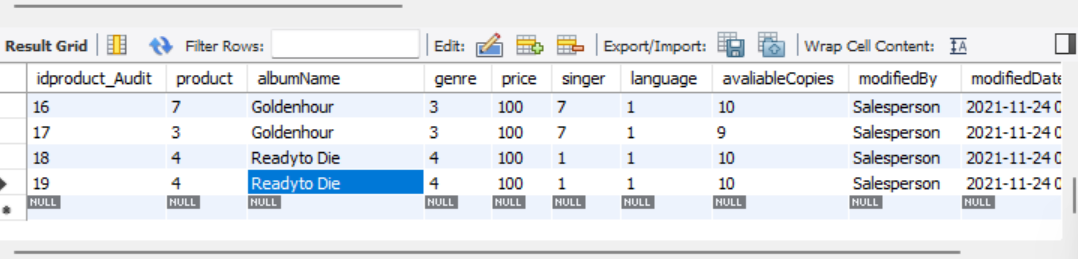
TRIGGER:

|  |
| --- |
| DELIMITER //  DROP TRIGGER IF EXISTS update\_product\_audit\_table;  CREATE TRIGGER update\_product\_audit\_table  AFTER UPDATE ON products  FOR EACH ROW  BEGIN  INSERT INTO product\_Audit  (product, albumName, genre, price, singer, language, avaliableCopies, modifiedBy, modifiedDate)VALUES  (OLD.idproducts, OLD.albumName, OLD.genre, OLD.price, OLD.singer, OLD.language,  OLD.avaliableCopies, OLD.modifiedBy, NOW());  END// |

Figure 22 - Automatic update of the Product Audit table







/\*---SCENARIO 8:--\*/

INSERT into music\_store.order ( idorder, customerID, product, purchaseAmount, discount, purchaseDate)

values ('18', '1', '3', '150', '1', '2021-1-11');

/\*---SCENARIO 9:--\*/

CALL generate\_discount\_report (1);

/\*---SCENARIO 10:--\*/

SELECT \* FROM PRODUCTS;

SELECT \* FROM PRODUCT\_AUDIT;

UPDATE PRODUCTS SET ALBUMNAME='ReadytoDie' WHERE IDPRODUCTS=4

* Create views for discount report – Customer wise

|  |
| --- |
| CREATE VIEW CustomerDiscountReport AS SELECT P.name,SUM(O.purchaseAmount) AS TotalSaleAmount,SUM(C.price) AS TotalProductPrice ,SUM(D.discountAmount) AS DiscountAmount FROM `order` O INNER JOIN customer P ON P.idcustomer = O.customerID INNER JOIN discount D ON D.iddiscount = O.discount INNER JOIN products C ON C.idproducts = O.product WHERE O.discount IS NOT NULL GROUP BY P.name; |

Graphical user interface, table

Description automatically generated

Figure 23 - Scheduled Weekly Backup of the Database

* Create views for discount report – Month wise

|  |
| --- |
| CREATE VIEW MonthWiseDiscountReport  AS  SELECT DATE\_FORMAT(purchaseDate, "%M-%Y") as `Month`,SUM(O.purchaseAmount) AS TotalSaleAmount,SUM(C.price) AS TotalProductPrice  ,SUM(D.discountAmount) AS DiscountAmount  FROM `order` O  INNER JOIN customer P ON P.idcustomer = O.customerID  INNER JOIN discount D ON D.iddiscount = O.discount  INNER JOIN products C ON C.idproducts = O.product  WHERE O.discount IS NOT NULL  GROUP BY DATE\_FORMAT(purchaseDate, "%M-%Y") |

# TECHNICAL DIFFICULTIES

* Had issues while updating the available copies column in the Products table with respect to the dynamic order entry but overcame it with the usage of trigger ,that will be triggered every time the customer places the order.
* During the SQL dump had issues with the permissions regarding the files but when we use the command along with ‘SUDO SU’, we were able to execute the MYSQL dump and create the database backup successfully.
* Initially had issues with the constraint but overcame it by mapping the relationships in ER diagram.