**Music Store Database**

**Table of Contents**

⦁ Introduction

⦁ Problem faced during the project

⦁ Solution for the problem

⦁ EER Diagram

⦁ Tables

⦁ Stored Procedures

⦁ Demo Queries

⦁ Appendix

**1.Introduction**

The main objective of this project is to develop a Simple relational Music Store Database. This database must store the data about Customers, Employees, Orders, Singers, Products etc.

In this database we can Insert, delete, and update the records for different bodies. This database is mainly built for Employees and Customers of a music store.

This database includes data from which we can find the following:

* List of Clients that spent more than the average amount spent by the clients in the past month.
* Top and least sold products over a week.
* The Maximum price of the products in the same genre.
* Customers based on their location.
* List of products sold in a particular month.
* List of distinct albums each singer has.
* Highest price of products in the same category.
* Available number of copies of the album of a particular singer.

This File gives idea about the progress of the database that includes structure of the database, Implementation, EER diagram, Views, Procedures, Primary Keys, Foreign Keys, and enhanced entity relation diagram.

**2.Problems Faced:**

We as a group faced a few difficulties while working on the project. To begin with, it was challenging to come up with the right user story, once we figured it out, we faced issues coming up with the right number of tables, later organising the relations between the tables. Also, faced issue while executing queries.

**3.Solution for Problems:**

Once we figured out the number of tables, content to be inserted into the tables. we worked on organising the tables. We used joins, views, procedures and developed highly efficient queries. We also watched the lecture recordings and referred e-text and came up with the relevant solution.

**4.EER Diagram:**

EER is Enhanced Entity Relationship diagram. These Diagrams are basically upgraded version of Entity-Relationship diagrams. These EER diagrams represent constraints and difficulties of complicated databases. EER diagrams are commonly used to increase the current ER diagrams to handle the difficult database in a better way.



**6.Tables:**

**i) music\_store**

* **storeId:** This column is of datatype INT and it will be the primary key of the music store table. It is defined as unique and auto incremental is enabled.
* **storeAddress:** This column is of datatype VARCHAR and used for storing the address of the music store and NOT NULL is enabled for this column so that it can't be null.
* **store\_phone\_number:** This column is of datatype VARCHAR and used to store the mobile number of the music store both unique and NOT NULL is enabled so that value it stores must be unique and not null.
* **storeZipcode:** This column is of datatype VARCHAR and used to store the postal code of the music store and NOT NULL feature is enabled so that it can't take the null value.

**ii) employees**

* **employeeId:** This column is of datatype INT and it will be the primary key of the employees table. it is used to store the id of the employee and both NOT NULL and unique is enabled so that the value it takes can't be null and will be unique values.
* **employeeName:** This column is of datatype VARCHAR and it is used for storing the name of the employee and NOT NULL feature is enabled so that it can't be null.
* **employeeAge:** This column is of datatype INT and used for storing the age of the employee and NOT NULL is enabled so that it can't accept the null values.
* **employeePhoneNumber:** This column is of datatype VARCHAR and is used for storing the mobile number of the employee and both UNIQUE and NOT NULL feature is enabled so that it will not accept the null values and must be unique.
* **music\_store\_storeId:** This column is of datatype INT and used for storing the storeId of the music store and the NOT NULL feature is enabled so that it won't accept the null values and it will be the foreign key of the employees table.
* **employeeAddress:** This column is of datatype VARCHAR and used for storing the address of the employee. The NOT NULL feature is enabled so that this column won't accept the null values.

**iii) customers**

* **customerId:** This column is of datatype INT and is used for storing the id of the customers table, it will be the primary key of the customers table, both UNIQUE and NOT NULL is enabled so that the column won't accept the null values and the entered values must be unique.
* **customerFname:** This column is of datatype VARCHAR and used for storing the first name of the customer and NOT NULL is enabled so that this column won't accept the null values.
* **customerLname:** This column is of datatype VARCHAR and used for storing the last name of the customer and NOT NULL is enabled so that this column won't accept the null values.
* **customerPhoneNumber:** This column is of datatype VARCHAR and is used for storing the mobile number of the customer, NOT NULL is enabled so that it won't accept the null values.
* **music\_store\_storeId:** This column is of datatype INT and used for storing the storeId of the music store and the NOT NULL feature is enabled so that it won't accept the null values and it will be the foreign key of the customer’s table.
* **customerAge:** This column is of datatype INT and is used for storing the age of the customers, NOT NULL is enabled so that the column won't accept the null values and the entered values must be unique.

**iv)address**

**• branchId:** This column is of datatype INT and it will be the PRIMARY KEY of the Address table. It is used to store the ID of the employee and both NOT NULL, UNIQUE and AUTO INCREMENT is enabled so that the value it takes can’t be null and always be unique values.

**• city:** This column is of datatype VARCHAR and is used for storing the city of the address table and NOT NULL is enabled so that it won’t accept the null values**.**

**• location:** This column is of datatype VARCHAR and is used for storing the location of the address table and NOT NULL is enabled so that it won’t accept the null values.

**• postal\_code**: This column is of datatype VARCHAR and is used for storing the postal code of the address and NOT NULL is enabled so that it won’t accept the null values**.**

**• music\_store\_storeId:** It is the primary key of music\_store Table and foreign key of address table.

**v)categories**

* **categoryId:** This column is of datatype INT and used for storing the categoryId of the categories table and will be the primary key of the categories table, Both NOT NULL and UNIQUE is enabled so that it won't accept the null values and values entered uniques. Auto\_incremental is enabled for this column.
* **categoryName:** This column is of datatype VARCHAR and used for storing the category name of the categories table and NOT NULL is enabled so that it won't accept the null values.
* **categoryType:** This column is of datatype VARCHAR and used for storing the type of the category and NOT NULL is enabled so that it won't accept the null values to be entered for this column.

**vi)orders**

* **orderId:** This column is of datatype INT and used for storing the orderId and it will be the primary key of the orders table and both NOT NULL and unique is enabled so that it won't accept the null values and ensures entered values to be unique.it is also auto incremental.
* **orderDate:** This column is of datatype DATETIME and used for storing the date of the order, NOT NULL is enabled so that it won't accept the null values.
* **shipAmount:** This column is of datatype INT and used for storing the ship amount of the orders table, NOT NULL is enabled so that it won't accept the null values.
* **taxAmount:** This column is of datatype INT and used for storing the tax amount of the orders table, NOT NULL is enabled so that it won't accept the null values.
* **customers\_customerId:** It will be the foreign key of orders table and primary key of customer Table.

**vii)order\_items**

**• itemId:** This column is of datatype INT and it will be the PRIMARY KEY of the order\_items table. It is used for storing the itemId of the order\_items and both AUTO INCREMENT and NOT NULL is enabled so that it won’t accept the null values.

**• itemPrice:** This column is of datatype INT and is used for storing the item price in orderItems table and NOT NULL is enabled so that it won’t accept the null values.

**• quantity**: This column is of datatype INT and is used for storing the quantity of items in orderItems table and NOT NULL is enabled so that it won’t accept the null values.

**• orders\_orderId:** It is the foreign key of order\_items table and primary key of order\_items Table.

**viii)products**

**• productId:** This column is of datatype INT and it will be the PRIMARY KEY of the products table.it is used for storing the product Id in the products table and both AUTO INCREMENT and NOT NULL is enabled so that it won’t accept the null values

**• productName:** This column is of datatype VARCHAR and is used for storing the product name in products table and NOT NULL is enabled so that it won’t accept null values.

**• productType:** This column is of datatype VARCHAR and is used for storing the type of product in products table and NOT NULL is enabled so that it won’t accept null values.

**• music\_store\_storeId:** This column is of datatype INT and it will be the primary key of Music store Table and foreign key of products table. It is used for storing store id in products table and NOT NULL is enabled so that it won’t accept null values.

**• categories\_categorieId:** This column is of datatype INT and it will be the primary key of categories Table and foreign key of products table. It is used for storing category id in products table and NOT NULL is enabled so that it won’t accept null values.

**• order\_items\_itemId**: This column is of datatype INT andit will be the foreign key of products table and primary key of order\_items Table. NOT NULL is enabled so that it won’t accept null values.

**• noOfProducts**: This column is of datatype INT and is used for storing the quantity of products in products table and NULL is enabled so that I can take default values as null.

**viii)singer**

* **singerId:** This column is of datatype INT and used for storing the singerId of the singer table and it is also the primary key of the singer table, both NOT NULL and unique is enabled so that it won't accept the null values and ensures entered values to be unique.it is also auto incremental
* **singer\_name:** This column is of datatype VARCHAR and used for storing the singer name, and NOT NULL is enabled so that it won’t accept the null values.
* **gender:** This column is of datatype VARCHAR and used for storing the gender of the singer, NOT NULL is enabled so that it won’t accept the null values.
* **singerAge:** This column is of datatype int and used for storing the age of the singer,, NOT NULL is enabled so that it won’t accept the null values.
* **music\_store\_storeId:** It is a foreign key of singer table and primary key of music\_store Table.

**ix) video**

* **videoId:** This column is of datatype INT and used for storing the id of the video.it will be the primary key of the video table, both NOT NULL and unique is enabled so that it won't accept the null values and ensures entered values to be unique.it is also auto incremental.
* **actorName:** This column is of datatype VARCHAR and used for storing the name of the actor, NOT NULL is enabled so that it won’t accept the null values.
* **genre:** This column is of datatype VARCHAR and used for storing the genre of the video,NOT NULL is enabled so that it won’t accept the null values.
* **singer\_singerId:** It will be the foreign key of the video table and primary key of singer Table.
* **videoQuality:** This column is of datatype VARCHAR and used for storing the quality of the video,NOT NULL is enabled so that it won’t accept the null values

**x) audio**

**• audioId:** This column is of datatype INT and it will be the primary key of the audio table. It is used to store audioId in the audio table and both AUTO INCREMENT and NOT NULL is enabled so that it won’t accept the null values**.**

**• singer:** This column is of datatype VARCHAR and it is used to store the singer name in audio table and NOT NULL is enabled so that it won’t allow null values**.**

**• music\_director:** This column is of datatype VARCHAR and it is used to store the music dirctor name in audio table and NOT NULL is enabled so that it won’t allow null values.

**• genre:** This column is of datatype VARCHAR and it is used to store the genre in audio table and NOT NULL is enabled so that it won’t allow null values.

**• singer\_singerId:** This column is of datatype INT and it will be the foreign key of audio table and primary key of singer Table. It is used to store the singerID in audio table and NOT NULL is enabled so that it won’t allow null values.

**• lyricist:** This column is of datatype VARCHAR and it is used to store the lyricist name in audio table and NOT NULL is enabled so that it won’t allow null values.

**7. Stored Program:**

-- Query for view --

-- creating a view for customersaveragePrice

-- drop view customersAveragePurchase;

CREATE VIEW customersAveragePurchase AS

SELECT CONCAT(customerFname,' ', customerLname) AS customerName , (orders.shipAmount + orders.taxAmount) AS totalPrice

FROM customers INNER JOIN orders ON customers.customerId = orders.customers\_customerId

having totalPrice > 100;

SELECT Avg(totalPrice) AS AversgeSale FROM customersAveragePurchase;

SELECT min(totalPrice) AS MinSale FROM customersAveragePurchase;

SELECT max(totalPrice) AS MaxSale FROM customersAveragePurchase;

select \* from customersAveragePurchase;

**-- Query for Stored Procedure 1 --**

-- a stored procedure for black friday sale

USE music\_store;

-- dropping procedure if exists already

DROP PROCEDURE IF EXISTS music\_store\_blackFridayDiscount;

DELIMITER //

-- creating procedure

CREATE PROCEDURE music\_store\_blackFridayDiscount()

BEGIN

-- declaring variables

DECLARE blackFridayDiscount INT;

DECLARE actualPrice DECIMAL(9,2);

DECLARE finalPriceAfterDiscount DECIMAL(9,2);

SET blackFridayDiscount = 2;

SELECT MAX(itemPrice) INTO actualPrice FROM order\_items;

SET finalPriceAfterDiscount = (actualPrice - (actualPrice\*(70/100)));

SELECT CONCAT('$', finalPriceAfterDiscount) AS 'BlackFridaySalePrice';

END//

DELIMITER ;

CALL music\_store\_blackFridayDiscount();

**-- Query for Stored Procedure 2 --**

-- stored procedure for the least quantity in the music\_store

USE music\_store;

DROP PROCEDURE IF EXISTS music\_store\_leastQuantity;

DELIMITER //

CREATE PROCEDURE music\_store\_leastQuantity()

BEGIN

DECLARE leastQuantityproduct INT;

SELECT MIN(quantity) INTO leastQuantityproduct FROM order\_items;

SELECT leastQuantityproduct;

END//

DELIMITER ;

CALL music\_store\_leastQuantity();

**-- Query for Stored Procedure 3 --**

-- stored procedure for calculating the final price after addition of tax

USE music\_store;

-- drops stored procedure if there exists any

DROP PROCEDURE IF EXISTS finalAmountAfterTax;

DELIMITER //

-- creation of stored procedure

CREATE PROCEDURE finalAmountAfterTax()

BEGIN

DECLARE finalPrice INT;

SELECT SUM(shipAmount + taxAmount) INTO finalPrice FROM orders;

SELECT CONCAT('$',finalPrice) as TotalAmount;

END//

DELIMITER ;

CALL finalAmountAfterTax();

-- stored procedure with transaction for updating existing data in video table

DELIMITER //

CREATE PROCEDURE videoEntryUpdates()

BEGIN

DECLARE sqlError TINYINT DEFAULT FALSE;

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION

SET sqlError = TRUE;

START TRANSACTION;

-- transaction starts here

-- setting values to new values

UPDATE Video SET actorName = "Jhonny Depp" where videoId = 1;

UPDATE Video SET genre = "Title Track" where videoId = 2;

IF sqlError = FALSE THEN

COMMIT;

SELECT("Data Updated");

ELSE

ROLLBACK;

SELECT("Three is a error in updating");

END IF;

END//

CALL videoEntryUpdates();

-- select \* from video;

**8. Demo Queries:**

**-- Query 1 --**

-- list of clients that spent more than the average spent by client in the past month

-- using joins

select CONCAT(customers.customerFname, ' ', customers.customerLname) AS "Customer's Name", customers.customerId, (shipAmount + taxAmount) as totalAmount

from customers join orders on

customers.customerId = orders.customers\_customerId

where (shipAmount + taxAmount) > (( select avg(shipAmount + taxAmount) from orders)) And (orderDate between '2021-02-25 22:59:52' and '2021-03-03 22:59:52');

**-- query 2 --**

-- The top sold products over a week

select p.productName, oi.quantity, oi.itemId

from products p join order\_items oi

on p.order\_items\_itemId=oi.itemId

-- where

order by quantity

limit 1;

-- and least sold products over a week

select p.productName, oi.quantity, oi.itemId

from products p join order\_items oi

on p.order\_items\_itemId=oi.itemId

order by quantity desc

limit 1;

**-- Query 3 --**

-- answer for the 3rd qsn

-- The maximum price of products in the same genre

-- GROUP BY to list all the genres and their maximum price.

SELECT categoryName, MAX(itemPrice) AS "Maximum Price Of a Product"

from products p join order\_items oi

on p.order\_items\_itemId=oi.itemId

join categories c

on c.categoryId=p.categories\_categoryId

group by categoryName;

**-- Query 4 --**

-- List of customers the system has by location and sorting them

select COUNT(customerFname) AS "Count Of Customers" ,city, location

from customers join music\_store

On customers.music\_store\_storeId=music\_store.storeId

join address

on music\_store.storeId=address.music\_store\_storeId

group by city

order by location asc;

**-- Query 5 --**

-- List of products the store has sold for a particular month.

select productName, orderDate, quantity

from products p join order\_items oi

on p.order\_items\_itemId=oi.itemId

join orders o

on oi.orders\_orderId=o.orderId

where orderDate between '2021-02-25 22:59:52' and '2021-03-03 22:59:52';

**-- Query 6 --**

-- List of distinct albums each singer has.

select singer\_name, a.genre AS "Album Type"

from singer s join audio a

On s.singerId=a.singer\_singerId

join video v

on s.singerId=v.singer\_singerId

UNION

select singer\_name, v.genre AS "Album Type"

from singer s join audio a

On s.singerId=a.singer\_singerId

join video v

on s.singerId=v.singer\_singerId;

**-- Query 7 --**

-- List of copies of an album are available of a particular singer.

select singer\_name, count(audioId) AS "Audio Album", count(videoId) AS "Video Album"

from singer s join audio a

On s.singerId = a.singer\_singerId

join video v

on s.singerId=v.singer\_singerId

group by singer\_name;

**-- Query 8 --**

-- list the employee details who works in Branch 4 --

select employeeName, employeePhoneNumber, employeeAge, storeId from employees

inner join music\_store on employees.music\_store\_storeId = music\_store.storeId

where music\_store.storeId=4;

**-- Query 9 --**

-- list the count of orders for each customers and displaying customerId, CustomerName, NoOfOrders and grouping by customerId

select customers.customerId, CONCAT(customerFname,' ', customerLname) AS customerName, count((orders.customers\_customerId)) as NoOfOrders

from customers inner join orders on customers.customerId = orders.customers\_customerId

group by customers.customerId;

**Appendix:**

**Table Breakdown:**

**Order\_items and Singer table:**

* Both Order\_items and Singer table are a part of the music store database.
* In the table order\_items we have a total of 4 columns, itemId, itemPrice, quantity and order\_orderid.
* For singer table we have columns like, singerId, singer\_name, gender, singerAge, and music\_store\_storeId
* To explicate more, column itemId has data type INT, Primary key, unique, and Auto Incremental is enabled. Additionally, itemPrice has data type VARCHAR and NOT NULL like column quantity.
* Last column in this table is the primary key of order\_items.
* In the other table, which is singers, the first column singerId has data types of INT, is the primary key, both NOT NULL and Unique is enabled. For Column singer\_name the data type is VARCHAR and NOT NULL is enabled.
* Moving further, gender column has data type VARCHAR and NOT NULL is enabled. For column singerAge data type is INT and NOT NULL is enabled, and the last column music\_store\_storeid is the foreign key of singer table.