**Documentation**

**to the**

**Database Project**

**‘Music Store database system’**

Created By team 17

GURMINDER SINGH

SAMPOORNANAND CHAUDARY

**Introduction**

The goal of this report is to design and develop a relational database for online music stores. The information of customers and artists consisting of their albums must be stored in this database. The ER diagram structured database explains the tables and relationships among customers , albums, artists, orders etc. which makes the task of managing music store data efficient and convenient.

First of all, The ER diagram is designed as per following requirements.

1. Customers table is a collection of customer’s information

2. One customer can have many orders

3. Orders can have many items

4. Artists table contain the information of artists

5. A artist can have many albums

6. Many albums can have single genre

7. Single album can have many orders

8. Order\_items stores the quantity of an ordered product and its price as well.

Furthermore, explicit requirements have been put into the database each table comprises of one primary key, NOT NULL, Auto-increment function and foriegn keys, for example, in customers table, id customers is made primary key with constraint like not null and auto-increment, in orders table, id Orders is an primary key and so on. Moreover, this ER model is in 3rd normalization form.

.According to requirements, model should consists following relationships:

1. one-to-one

2. one-to-many

3. many to many

To address the problems of user in given test scenarios, database objects procedure and function have been used in MySQL queries.

**General scenario**

It has 7 questions that have been addressed with MySQL queries using following database functions and objects -

1. query1 named Procedure created, IF ELSE and validations are used to display the list of clients that spent more than the average spent by client in the past month.

2. Max() and Min() are used along with join query to get the result of top sold and least told product over the week

3. Max() function is used to get the result -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.

4. COUNT() function and Group By and Order By clauses are used to display the number of customers system has by location (Country, Province, and City)

5. Function named product\_sales created and Month() function to retrieve the month part from specified date, which is passed as an argument to function is used to list the sold products for a particular month.

6. COUNT() along with DISTINCT keyword and GROUP BY clause is used to get the information about - list how many distinct albums each singer has.

7. GROUP BY and ORDER BY clauses and SUM() function are used to answer the question - List how many number of songs are available for a particular albums.( we have asked you mam to changed this question, as we have missed number of copies column in our tables. )

**Specific scenario**

 This testing scenario consists of two questions to be addressed –

1.Filter albums according to instruments used in the albums

Inner join has been used here to retrieve data by joining two tables albums and instruments.

2. Less popular instrument used in the albums

Subquery from album table has been used to answer this question

**Documentation**

The structure of the database using ER diagram demonstrates the formation and the communication in the database. The following are the entities with constraints of the music store database.

|  |  |
| --- | --- |
|  |  |

**Table:albums**

**Columns:**

|  |  |
| --- | --- |
| **idalbums** | int AI PK |
| Album\_name  Release\_Date  Num\_of\_songs | varchar(45)  datetime  int |
| Album\_price | decimal(10,2) |
|  |  |
| **genres\_idgenres** | int |
|  |  |
| **Artist\_idArtist** | **int** |

**Table:customers**

**Columns:**

|  |  |
| --- | --- |
| **idCustomers** | int AI PK |
| First\_Name | varchar(45) |
| Last\_Name | varchar(45) |
| Email | varchar(45) |
| Gender | varchar(45) |
| City | varchar(45) |
| Country | varchar(45) |
| Province | varchar(45) |

**Table: orders**

**Columns:**

|  |  |
| --- | --- |
| **idOrders** | int AI PK |
| Order\_Date | date |
| Ship\_Amount | decimal(10,2) |
| Ship\_Date | date |
| **Customers\_idCustomers** | int |

**Table:order\_items**

**Columns:**

|  |  |
| --- | --- |
| **idOrder\_Items** | int AI PK |
| Product\_Price | decimal(10,2) |
| Product\_quantity | int |
| **Orders\_idOrders** | int |
| **Albums\_idAlbums** | int |

**Table:Artist**

**Columns:**

|  |  |
| --- | --- |
| **idArtist** | int AI PK |
| Artist\_First\_Name | varchar(45) |
| Artist\_Last\_Nmae | varchar(45) |
| Artist\_Gender | varchar(45) |
| Artist\_age | int |

**Table:genres**

**Columns:**

|  |  |
| --- | --- |
| **idGenre** | int AI PK |
| Genre | varchar(45) |
| **Table:Instruments**  **Columns:**   |  |  | | --- | --- | | **idInstruments** | int AI PK | | Instrument\_Name | varchar(45) | | Instrument\_Type | varchar(45) | |  |

Mapping cardinalities in ERD are illustrated below:

1. A customer can have one to many with order

2. An order can be of many products

3. Album entity has one-to-many relationship with order-items, which further has many-to-one relationship with order, which means each album can have multiple orders

4. genre has one-to-many with albums

5. Artist can have one to many albums.

6. Instruments has many-to-many with albums

 Next step after modelling the normalized model according to determined requirements is to generate the script.

Now that database is designed and developed, populated the database to process the testing whether it is apt to address the problems of user by using MySQL queries consisting of specific database functions and objects. Lastly, dump file is created for backup of database.

**Additional Scenario**

**These are additional queries system might have asked:-**

1. List Most and least popular Artist.
2. Customer orders data should be deleted after few months and create a backup of deleted data.

**Challenges**

Challenges faced by (Sampoornanand Chaudhary):

1. Had little confusion at the starting , that how many tables needs to be created but after reading the project instruction , it was cleared.

2. Was confusion in , which data type to be used, while making the columns but it was clear later by doing the research.

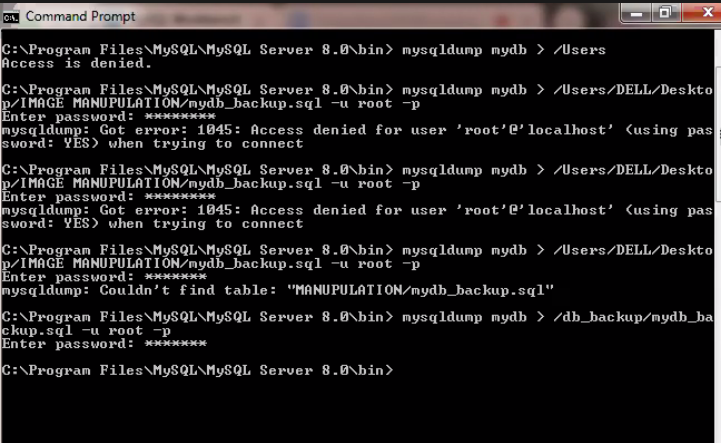
3. Relationships was made easily but had struggled during the data entry.

Challenges faced by (Gurminder singh):

4. As I was unaware that just after entering the data inside the table if we refresh the DB, data use to get disappeared

5. Again we have to add the data to the table.

6.There was little problem during taking the DB backup, it was due to putting the wrong path of the SQL Workbench in the command of the command prompt. And later was also solved



1. To accomplish result In 5th question, for particular month, Month() function is used to extract the month part from order date.
2. For 7th question we missed Number of copies column in albums table but later professor replaced it with another question
3. Now when started the query writing for fetching the result, it took time as it was not so simple and not so difficult. But after putting so much of hard work we achieved it.

**conclusion**

 The online music store database is outlined by considering all the requirements and the shortcomings of the database users. The database is developed with basic functionalities and constraints have been applied. Evidently, it makes the way to include additional features by bringing in more entities, attributes and testing every possibilities.

The goal of this report is to design and develop a relational database for online music stores. The information of customers and artists consisting of their albums must be stored in this database. The ER diagram structured database explains the tables and relationships among customers , albums, artists, orders etc. which makes the task of managing music store data efficient and convenient.

## Appendix:

Tables

**Table:orders**  
  
**Columns:**

|  |  |
| --- | --- |
| **idOrders** | int AI PK |
| Order\_Date | date |
| Ship\_Amount | decimal(10,2) |
| Ship\_Date | date |
| **Customers\_idCustomers** | int |

**Orders table** will keep track of customers’ orders. it addresses the question- List how many products the store has sold for a particular month. For month order date is required.

**idorder** – it is primary key attribute with auto-increment.

**order\_date** – it will store the date on which order has been placed so datatype is DATE.

**customers\_idcustomers** – it is foreign key which refers to customers table.

Ship\_amount – it store the amount of shipping charges , its datatype is decimal(10,2)

**Ship\_Date** – its data type is Date because it stores shipping date of order.

**Table:order\_items**  
  
**Columns:**

|  |  |
| --- | --- |
| **idOrder\_Items** | int AI PK |
| Product\_Price | decimal(10,2) |
| Product\_quantity | int |
| **Orders\_idOrders** | int |
| **Albums\_idAlbums** | int |

**order\_items** is designed to store the quantity and price of each ordered product from specific album. It addresses the question - List how many products the store has sold for a particular month. Attribute quantity is needed to calculate the number of products sold.

**idorder\_items** – it stores the integer values with constraint primary key and auto-increment.

**order\_idorder** – it is foreign key refers to orders table.

**Product\_quantity** – it stores the integer values.

**Product\_price** – data type decimal justifies the price value of product

**albums\_idalbums** – it is foreign key refers to albums table.