Logo, company name

Description automatically generatedLogo, company name

Description automatically generated

19335244 – Ailbhe Merriman

Developing a MySQL Database

CSU34041

**Contents:**

**Section A: Description of Database Application area and ER Model**

**A.1** Application Description2

**A.2** Entity Relationship Diagram3

**A.3** Mapping to Relational Schema4

**A.4** Functional Dependency Diagram5

**Section B Explanation of data and SQL Code:**

**B.1** Explanation of one the SQL Code for Creating one of your database Tables**6**

**B.2** Explanation and SQL Code for any Altering tables6

**B.3** Explanation and SQL Code for any Trigger operations............6

**B.4** Explanation and SQL Code for and Creation of Views...............6

**B.5** Explanation and SQL Code for one of your commands to Populate a Tables……………………………………………………………….8

**B.6** Explanation and example SQL Code for retrieving information from the database ……………………………………………..............8

**B.7** Explanation and SQL Code for any Security commands……9

**A.1 Application Description**

I decided to do my project based on a fictional art museum, I’m calling it Museum Database or MDB. The database stores values based on multiple aspects of the gallery, these include Artist, who crates Art Piece and is associated with Art Style. The Art Pieces are stored in Galleries, where Visitors visit and Employees who are given Role’s work.

The following are the seven tables I have created to represent the MDB.

**Artist-** The Artist table contains an artist’s unique ID number, their name, their nationality, the year they were born and the art style id most associated with them, which is the foreign key. The primary key is the artist ID number.

**Art Style –** The Art Style table consists of the unique style ID, which is the primary key, the century it was originated and the art styles descriptor.

**Art Piece –** The Art Piece table is made up of the art piece’s unique ID, the price of the piece, the artist ID and the gallery ID which are both foreign keys and the piece’s name. The art piece’s ID is the primary key.

**Galleries –** The primary key in the Galley table is the gallery’s unique ID. Other attributes are the postcode and the area name.

**Visitor –** The Visitor table is a table that contains the attributes relating to the visitors of each of the galleries. The primary key of this table is the unique visitor ID, other attributes are the visitors full name and the gallery which they visited, which is a foreign key.

**Role –** The Role table is a table which explains the roles that each of the employees have. The primary key is the role ID. Other attributes seen in this table are name if the role, the wage per hour and the gallery ID, which is a foreign key.

**Employee –** TheEmployee table is the final table which I created. The primary key of this table is the employee ID number. It also contains the employee’s surname, their bank number, and their phone number. The table also contains the role ID which tells us which role is assigned to which employee.

**A.2 Entity Relationship Diagram**

**Diagram

Description automatically generated**

* The Primary key is underlined e.g., the employee ID attribute is underlined in the employee table.
* The arrow’s colour corresponds to the table which they stem from e.g., the employee arrow to role has a red arrow.

**A.3 Mapping to Relationship Schema**

**Diagram

Description automatically generatedhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh**

**A.4 Functional Dependency Diagram**

**Diagram

Description automatically generatedewuhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhB.1 Explanation for the SQL code for creating the Art Piece Table**

   CREATE TABLE IF NOT EXISTS ArtPiece(

     piece\_id INT NOT NULL UNIQUE,

     price TEXT(255) NOT NULL,

     artist\_id INT(2) NOT NULL,

     piece\_name TEXT(255) NOT NULL,

     gallery\_id INT(2) NOT NULL,

     PRIMARY KEY (piece\_id)

 );

1. Art Piece Table

I used the CREATE TABLE function to create my tables. For each attribute I added the primary key first and declared it as UNIQUE, as seen with the piece\_id attribute. I then decided to declare the price as TEXT rather than an INT because I wanted to add the euro symbol. As seen with all the TEXT variables I set them to 255, which is the maximum character length allowed in MySQL. The artist\_id and the gallery\_id are both set to INT of 2 characters. Each of my attributes in all my tables are declared as NOT NULL. At the end of each table, I specified the PRIMARY KEY also, which again, in this table is the piece\_id.

**B.2 Explanation and SQL code for Altering tables**

 ALTER TABLE ArtPiece

ADD CONSTRAINT FK\_ArtistPiece

FOREIGN KEY (artist\_id) REFERENCES Artist(artist\_id);

1. Alter Table

I created alter tables to show the foreign keys. For example, this is the ALTER TABLE for adding a foreign key to the Art Piece table. The foreign key in question is the artist\_id which REFERENCES the Artist table.

I did this exact same procedure for all the tables which contain foreign keys which are the Artist table, Art Piece table, Visitor Table, Role Table, and the Employee ID.

**B.3 Explanation and SQL code for any Trigger operations**

1. CREATE TRIGGER Invalid\_Phone
2. BEFORE INSERT ON Employee
3. FOR EACH ROW
4. BEGIN
5. IF new.phone\_no > 10 THEN
6. SIGNAL sqlstate '99999'
7. SET message\_text = 'Invalid domain';
8. END IF;
9. END$$

I created a trigger which will stop a user from inserting a value for the employee’s phone number, which is invalid, in this case when the integer length is more than 10. Using the CREATE TRIGGER function I named my trigger Invalid\_Phone. If someone inputs a value which s more than 10 integers long, the trigger will send an error signal with the message Invalid domain.

**B.4 Explanation and SQL code for creating Views**

CREATE VIEW ItalianPainters AS

SELECT Artist.artist\_id, Artist.name

FROM Artist

WHERE Artist.nationality = 'Italian';

CREATE VIEW Researchers AS

SELECT Employee.employee\_id, Employee.surname

FROM Employee

WHERE Employee.role\_id = 66;

I created two views using the CREATE VIEW function within my database. Firstly, I created the ItalianPainters view, which shows the artist\_id and their name when their nationality is Italian. This could be useful to users of the data base to find all the artists of the same nationality.

This is the result of the code when queried.

A screenshot of a computer

Description automatically generated

Secondly, I created a VIEW which pulls the employee’s name and their ID when they have the same role. I called this view Researchers as they would both have the role ID 66, which corresponds to the Researcher role.

This is the result when I query the code.

A screenshot of a computer

Description automatically generated

This example of output is also an example of retrieving data from the database using queries.

**B.5 Explanation and SQL code for populating the tables**

 INSERT INTO Employee VALUES(111,64,'Gaffney',13577913,0869773678);

 INSERT INTO Employee VALUES(112,62,'Vaughan',24681012,0878983678);

 INSERT INTO Employee VALUES(113,61,'Khan',34710122,0856787836);

 INSERT INTO Employee VALUES(114,66,'Doherty',76543890,0897654315);

 INSERT INTO Employee VALUES(115,63,'OSullivan',12345678,0878763312);

 INSERT INTO Employee VALUES(116,65,'Donnelly',28737892,0854678936);

 INSERT INTO Employee VALUES(117,66,'Merriman',26797892,0859638935);

1. Employee Table Values

To insert my data into each table I used the INSERT INTO function, followed by the table I wanted to populate the VALUES and then the values for each tuple in order of how I coded the tables, the primary key was always first, in this example for the Employee table, my primary key employee ID was first. The other attributes were as follows, role ID, surname, bank number and phone number. TEXT was inserted into inverted commas and the INT separated by commas.

**B.6 Explanation and SQL code for retrieving information from the database**

SELECT Artist.name, Artist.nationality, Artist.year\_born, ArtPiece.piece\_name

FROM ArtPiece

INNER JOIN Artist ON Artist.artist\_id = ArtPiece.artist\_id;

1. Join

To retrieve information from the database I used an INNER JOIN. In this example I wanted to view which art pieces were completed by which artist. I did this by using the SELECT select funtion to select the artist’s name, nationality and year born from the Artist table and the piece name from the Art Piece table where the artist ID from both the Artist table and Art Piece table match. Similarly as shown above querying can also be a way to retrieve data from this database.

The result of this join can be seen below.

Text

Description automatically generated

**B.7 Explanation and SQL code for Security commands**

 CREATE ROLE owner\_of\_gallery;

GRANT ALL ON MDB\* TO owner\_of\_gallery;

1. Security

Here using the CREATE function we are creating a role of the owner of the gallery, they have been granted using the GRANT function to grant access to all features like INSERT, UPDATE, DELETE on the Museum Database.