# CHAPTER 1

# INTRODUCTION

## 1.1 INTRODUCTION TO SQL

SQL which is an abbreviation for **Structured Query Language** is a language to request data from a database, to add, update, or remove data within a database, or to manipulate the metadata of the database.

Sometimes SQL is characterized as *non-procedural* because procedural languages generally require the details of the operations to be specified, such as opening and closing tables, loading and searching indexes, or flushing buffers and writing data to file systems. Therefore, SQL is designed at a higher conceptual level of operation than procedural languages.

Commonly used statements are grouped into the following categories

**Data Query Language (DQL)**

SELECT-Used to retrieve certain records from one or more tables.

**Data Manipulation Language (DML)**

INSERT - Used to create a record

UPDATE - Used to change certain records.

DELETE - Used to delete certain records.

**Data Definition Language (DDL)**

CREATE - Used to create a new table, a view of a table, or other object in database.

ALTER - Used to modify an existing database object, such as a table.

DROP - Used to delete an entire table, a view of a table or other object in the database.

**Data Control Language (DCL)**

GRANT - Used to give a privilege to someone

REVOKE - Used to take back privileges granted to someone.

## 1.2 INTRODUCTION TO FRONT END SOFTWARE

The “front end languages” live in the browser. After you type in an address in the address bar at the top and hit Enter, your browser will receive an at least an HTML file from the web server.

Each of these languages performs a separate but very important function but the work harmoniously together to determine how the web page is STRUCTURED(HTML), how it LOOKS(CSS), and how its FUNCTIONS (JavaScript).

Front end web development is NOT design (You won’t be playing around in Photoshop or anything), but a *front-end developer* does apply the work of designers to the web page by translating their well-designed layouts into real code. The front-end developer stands between the designer on one end and the back-end developer on the other, translating the design into code and plugging the data from the back-end developer into the right spots.

**PHP** is a server-side scripting language designed primarily for web development but also used as a general-purpose programming language. Originally created by Rasmus Lerdorf in 1994, the PHP reference implementation is now produced by The PHP Development Team.

PHP code may be embedded into HTML or it can be used in combination with various web template systems, web content management systems and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server software combines the result of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and it can be used to implement stand-alone graphical applications.

The standard PHP interpreter, powered by the Zend Engine, is free to use software released under the PHP License. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as Common Gateway Interface(CGI) executable. PHP has been widely ported on web servers on almost every operating system and platform, free of charge.

# CHAPTER 2

# REQUIREMENTS SPECIFICATION

## 2.1 SOFTWARE REQUIREMENTS

| | | |
|---|---|---|
| Operating System | : | 64bit WINDOWS Operating System, X64-based processor |
| Database | : | MYSQL |
| Scripting Language | : | HTML5, CSS3, PHP |
| Server | : | WAMP |

## 2.2 HARDWARE REQUIREMENTS

| | | |
|---|---|---|
| Processor | : | Intel Celeron CPU N3060 @1.60GHz or Above |
| RAM | : | 4.00 GB or Above |
| Hard Disk | : | 1 TB |
| Compact Disk | : | CD-ROM, CD-R, CD-RW |
| Input device | : | Keyboard |

# CHAPTER 3

# OBJECTIVE OF THE PROJECT

## The main objective of creating an Art Gallery database project is

- To manage the details of gallery, exhibition, artwork and artist. It manages all the sales and inventory in the gallery. The purpose of the project is to build and application program to reduce the manual work.

- To tracks all the details about the sales of the artwork, the customer that bought it, etc. It manages the information about the artwork. Provides an information and description of the artworks left, thereby increasing the efficiency of managing the gallery. The organisation can maintain a computerized record of the artwork present in the gallery.

- To helps in the utilization of the resources in an effective manner. It maintains a list of all the customers and the various artwork that they have bought and the money that have invested in each.

- To maintains the record of exhibitions and various sales made during it. The objective of developing such computerized system is to reduce the paper work and safe of time in art gallery database management, thereby increasing the efficiency and decreasing the work load.

- To develop such computerized system is to reduce the paper work and safe of time in art gallery database management, thereby increasing the efficiency and decreasing the work load.

# CHAPTER 4

# IMPLEMENTATION

## 4.1 ER DIAGRAM

1. An **entity-relationship model (ER Model)** describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types.

2. An entity may be defined as a thing capable of an independent existence that can be uniquely identified. An entity is an abstraction from the complexities of a domain.

3. Attributes are drawn as ovals and are connected with a line to exactly one entity or relationship set.

4. An entity relationship model, also called an entity-relationship (ER) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems.

5. Cardinality constraints are expressed as follows:

   a. A double line indicates a participation constraint, totality or subjectivity: all entities in the entity set must participate in at least one relationship in the relationship set.

   b. An arrow from entity set to relationship set indicates a key constraint, i.e. injectivity: each entity of the entity set can participate in at most one relationship in the relationship set.

   c. A thick line indicates both, i.e. bijectivity: each entity in the entity set is involved in exactly one relationship.

   d. An underlined name of an attribute indicates that it is a key: two different entities or relationships with this attribute always have different values for this attribute.

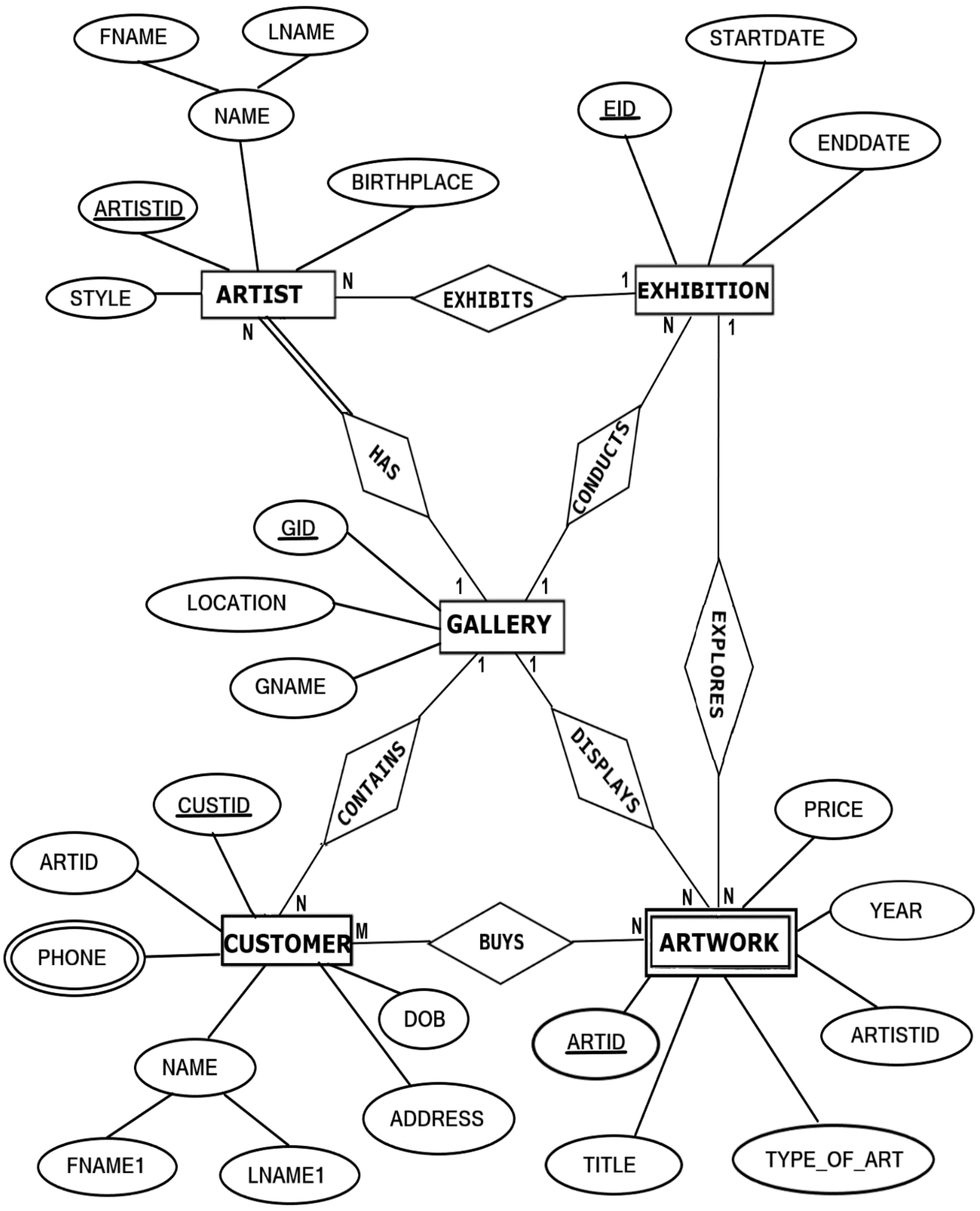


**FIGURE 4.1: ER DIAGRAM of ART GALLERY DATABASE**

## 4.2 MAPPING OF ER DIAGRAM TO RELATIONS

### STEP 1: Mapping of Regular Entities

For each regular entity type E in the ER schema, create relation R that includes all simple attributes of E.

**GALLERY**

| | | |
|---|---|---|
| **GID** | GNAME | LOCATION |

**EXHIBITION**

| | | |
|---|---|---|
| **EID** | STARTDATE | ENDDATE |

**ARTIST**

| | | | | |
|---|---|---|---|---|
| **ARTISTID** | FNAME | LNAME | BIRTHPLACE | STYLE |

**CUSTOMER**

| | | | | | | |
|---|---|---|---|---|---|---|
| **CUSTID** | **ARTID** | FNAME1 | LNAME1 | ADDRESS | PHONE | DOB |
| | **FK** | | | | | |

### STEP 2 : Mapping of Weak Entity Types

**ARTWORK**

| | | | | | |
|---|---|---|---|---|---|
| **ARTID** | **ARTISTID** | TITLE | TYPE_OF_ART | YEAR | PRICE |
| | **FK** | | | | |

## STEP 3: Mapping of 1:1 Relationship

Identify the relation S that represents the participating entity type at the 1-side of the relationship type.

Include as foreign key in S the primary key of the relations T that represents the other entity type participating in R.

For each binary 1:1 relationship type R in ER schema, identify the relations S and T that correspond to the entity types participating in R if any.

There are **no** 1:1 relationship.

## STEP 4 : Mapping of 1:N Relationship

**EXHIBITION**

| **EID** | STARTDATE | ENDDATE | **GID** |
|---|---|---|---|
| | | | **FK** |

**ARTIST**

| **ARTISTID** | FNAME | LNAME | BIRTHPLACE | STYLE | **EID** | **GID** | **CUSTID** |
|---|---|---|---|---|---|---|---|
| | | | | | **FK** | **FK** | **FK** |

**CUSTOMER**

| **CUSTID** | **ARTID** | FNAME1 | LNAME1 | ADDRESS | DOB | **GID** |
|---|---|---|---|---|---|---|
| | **FK** | | | | | **FK** |

**ARTWORK**

| **ARTID** | **ARTISTID** | TITLE | TYPE_OF_ART | YEAR | PRICE | **EID** | **GID** |
|---|---|---|---|---|---|---|---|
| | **FK** | | | | | **FK** | **FK** |

## STEP 5 : Mapping of M:N Relationship

Create a new relation S to represent R.

Include as foreign key attributes in S the primary key of the relations that represents the participating entity types their combination will form the primary key of S.

Also, include any simple attributes of the M:N relationship type as attributes of S.

## STEP 6: Mapping of Multi-Valued Attributes

For each multivalued attributes A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.

The Primary Key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

**CONTACTS**

| **CUSTID** | PHONE |
|---|---|

## STEP 7: Mapping of N-Ary Relationship Types

For each n-ary relationship type R, where n>2 create a new relationship S to represent R. λ include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.

λ also includes any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.

There are **no** n-ary relationship types.

## 4.3 SCHEMA DIAGRAM

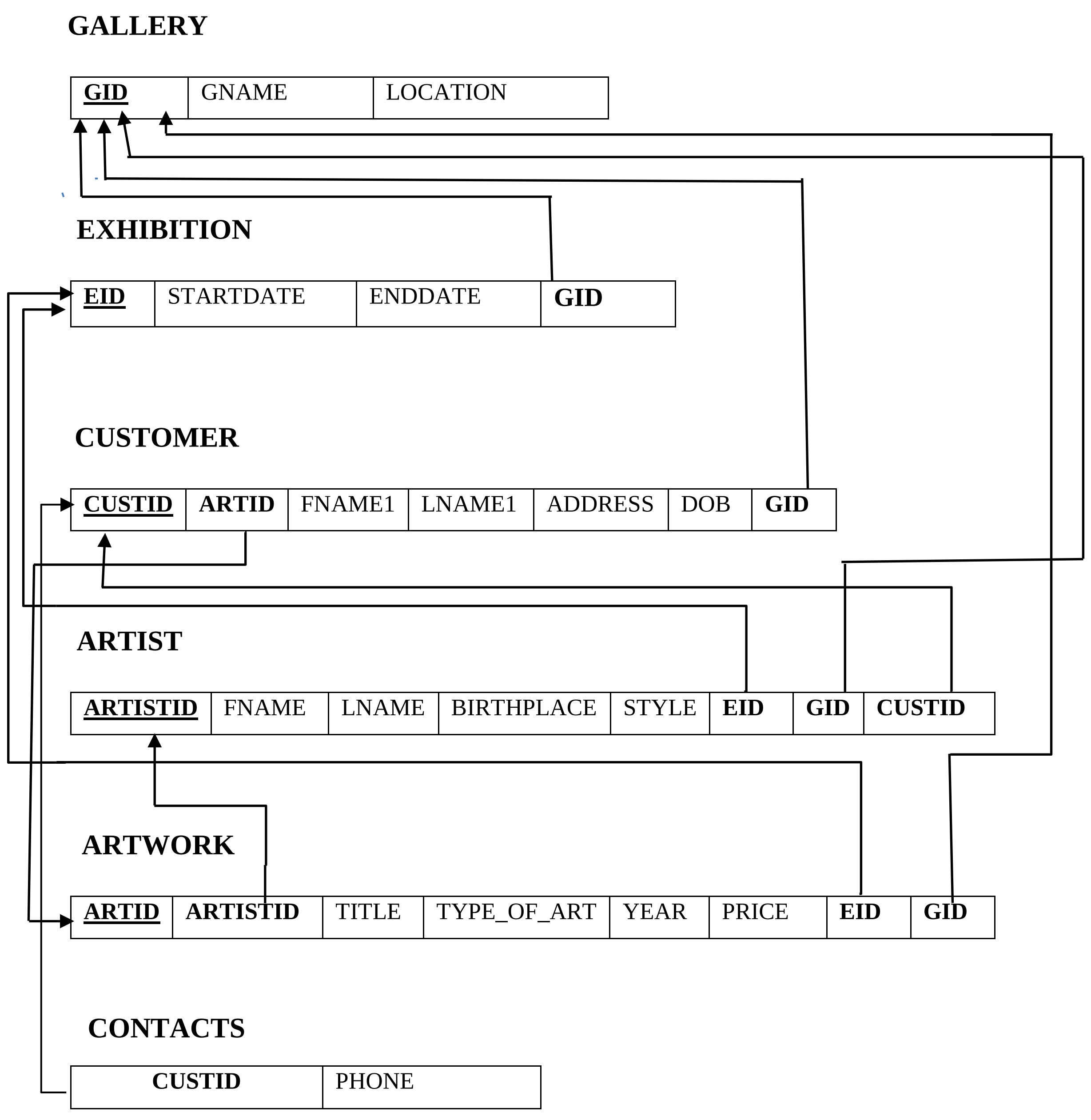


FIGURE 4.3: SCHEMA DIAGRAM

## 4.3 NORMALIZE THE RELATIONS

Database normalization, or simply normalization, is the process of organizing the columns(attributes) and tables(relations) of a relational database to reduce data redundancy and improve data integrity. Normalization involves arranging attributes in relations based on dependencies between attributes.

### 1. First Normal Form

As per First normal form, no two rows of data must contain repeating group of information. Each set of columns must have a unique value, such that multiple columns cannot be used to fetch the same row. Each table should be organized into rows, and each row should have a primary key that will distinguishes it as unique.

**Example:**

**GALLERY**

| **GID** | GNAME | LOCATION |
|---|---|---|

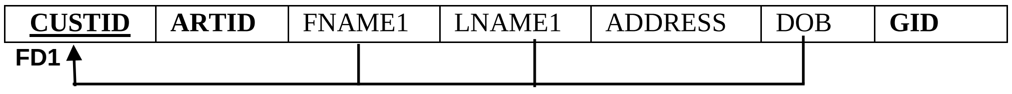
All the tables in the database are normalized to 1NF as all the attributes are atomic.

### 2. Second Normal Form (2NF)

A table is in 2NF if it is in 1NF and if all non-key attributes are fully functionally dependent on all of the key.

**Example:**

**CUSTOMER**



| **CUSTID** | **ARTID** | FNAME1 | LNAME1 | ADDRESS | DOB | **GID** |
|---|---|---|---|---|---|---|

**FD1**

| **CUSTID** | FNAME1 | LNAME1 | DOB |
|---|---|---|---|

### 3. Third Normal Form(3NF):

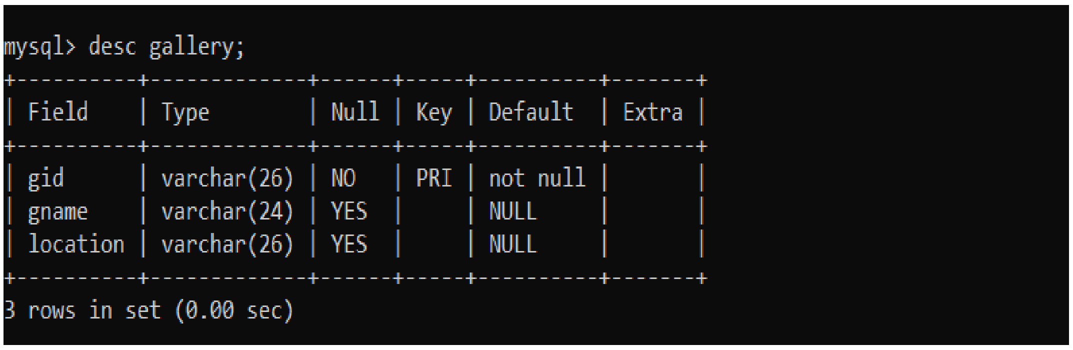
A table is in 3NF if it is in 2NF and if it has no transitive dependency. X->Y, Y->Z, X>Z

According to CODD’s definition a relation schema R is in 3NF. It satisfies 2NF and no non-prime attribute of R is transitively dependent on the primary key. All tables of database satisfies upto 3NF.

## 4.5 CREATION OF TABLES

### 1. CREATING GALLERY TABLE

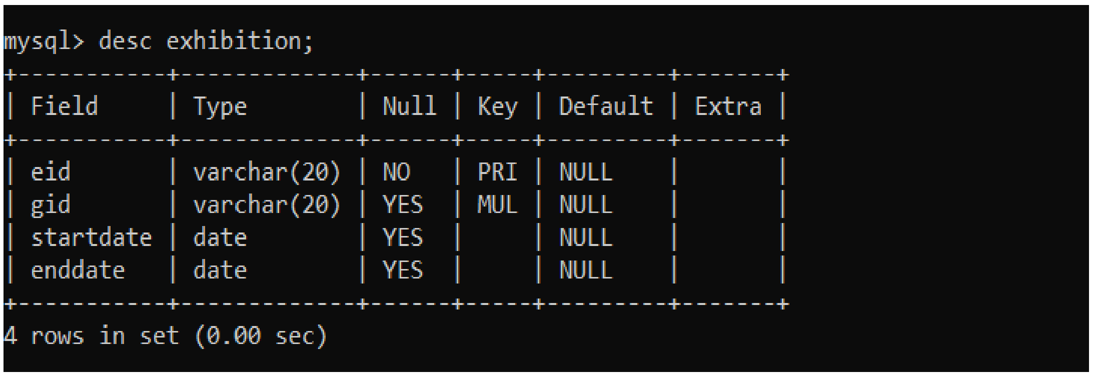
```
CREATE TABLE GALLERY
(GID VARCHAR(20) PRIMARY KEY,
GNAME CHAR(20),
LOCATION CHAR(20));
```



```
mysql> desc gallery;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| gid      | varchar(26) | NO   | PRI | not null |      |
| gname    | varchar(24) | YES  |     | NULL    |       |
| location | varchar(26) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

### 2. CREATE EXHIBITION TABLE

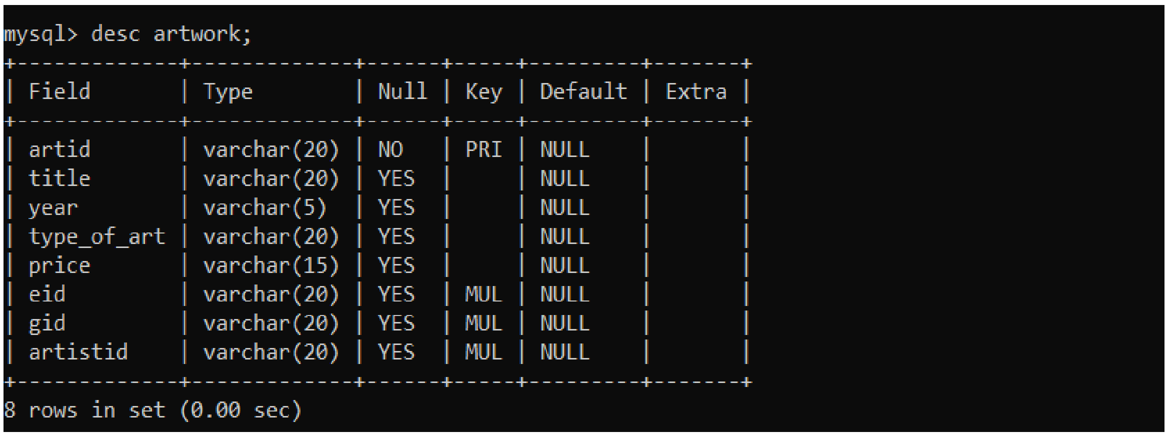
```
CREATE TABLE EXHIBITION
(EID VARCHAR(20) PRIMARY KEY,
GID VARCHAR(20),
STARTDATE DATE,
ENDDATE DATE,
FOREIGN KEY(GID) REFERENCES GALLERY(GID) ON DELETE CASCADE);
```



```
mysql> desc exhibition;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| eid       | varchar(20) | NO   | PRI | NULL    |       |
| gid       | varchar(20) | YES  | MUL | NULL    |       |
| startdate | date        | YES  |     | NULL    |       |
| enddate   | date        | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

### 3. CREATE ARTWORK TABLE

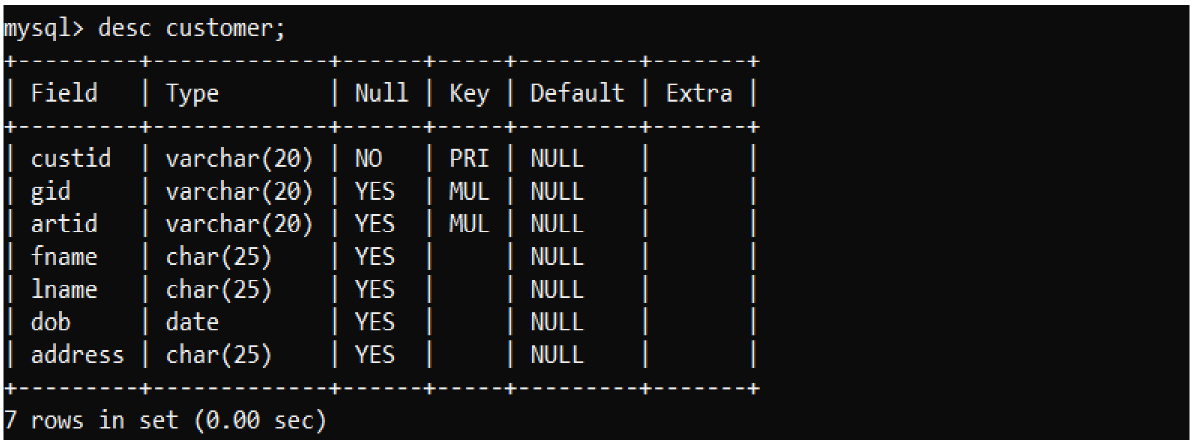
```
 CREATE TABLE ARTWORK
(ARTID VARCHAR(20) PRIMARY KEY,
TITLE VARCHAR(20),
YEAR INT,
TYPE_OF_ART VARCHAR(20),
PRICE INT,
EID VARCHAR(20), GID VARCHAR(20),
FOREIGN KEY(EID) REFERENCES EXHIBITION(EID) ON DELETE CASCADE,
FOREIGN KEY(GID) REFERENCES GALLERY(GID) ON DELETE CASCADE);
```



```
mysql> desc artwork;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| artid       | varchar(20) | NO   | PRI | NULL    |       |
| title       | varchar(20) | YES  |     | NULL    |       |
| year        | varchar(5)  | YES  |     | NULL    |       |
| type_of_art | varchar(20) | YES  |     | NULL    |       |
| price       | varchar(15) | YES  |     | NULL    |       |
| eid         | varchar(20) | YES  | MUL | NULL    |       |
| gid         | varchar(20) | YES  | MUL | NULL    |       |
| artistid    | varchar(20) | YES  | MUL | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

## 4. CREATE CUSTOMER TABLE

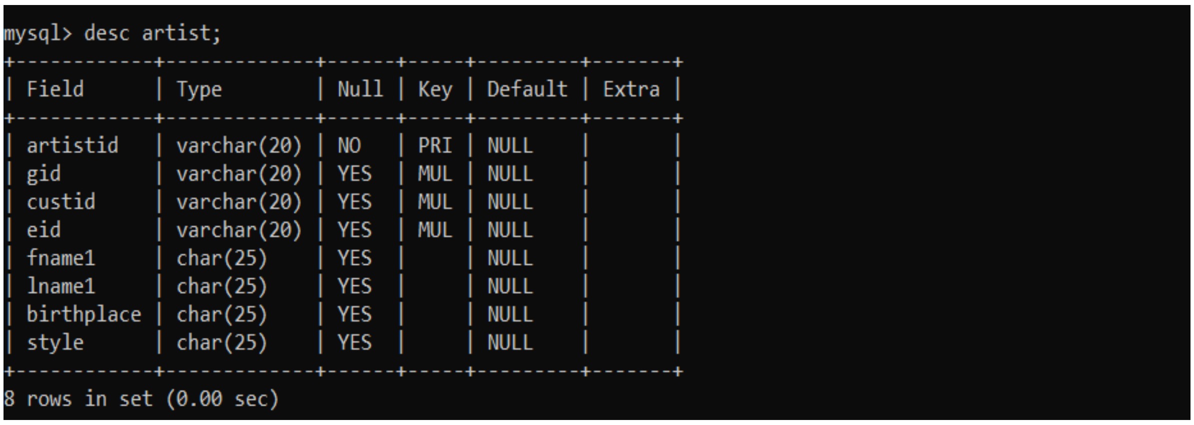
```
CREATE TABLE CUSTOMER
(CUSTID VARCHAR(20) PRIMARY KEY,
GID VARCHAR(20),
ARTID VARCHAR(20),
FNAME1 CHAR(20),
LNAME1 CHAR(20),
DOB DATE,
ADDRESS CHAR(20),
FOREIGN KEY(GID) REFERENCES GALLERY(GID) ON DELETE CASCADE,
FOREIGN KEY(ARTID) REFERENCES GALLERY(ARTID) ON DELETE CASCADE);
```



```
mysql> desc customer;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| custid  | varchar(20) | NO   | PRI | NULL    |       |
| gid     | varchar(20) | YES  | MUL | NULL    |       |
| artid   | varchar(20) | YES  | MUL | NULL    |       |
| fname   | char(25)    | YES  |     | NULL    |       |
| lname   | char(25)    | YES  |     | NULL    |       |
| dob     | date        | YES  |     | NULL    |       |
| address | char(25)    | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

## 5. CREATE ARTIST TABLE

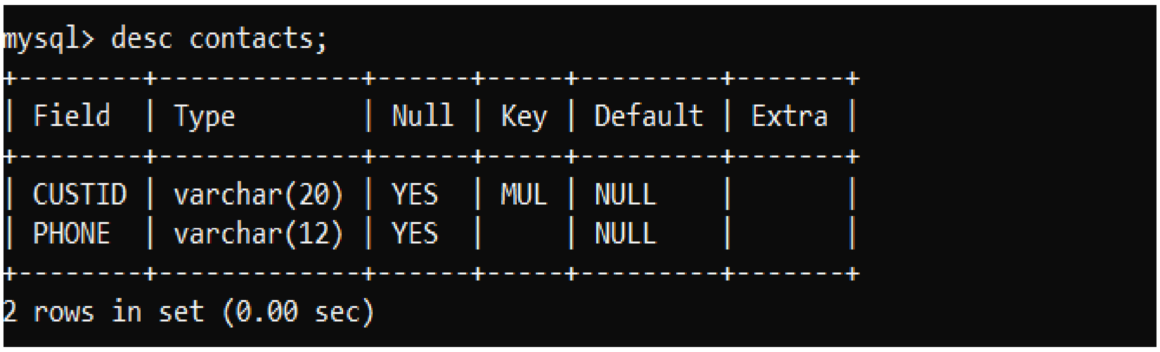
```
CREATE TABLE ARTIST
(ARTISTID VARCHAR(20) PRIMARY KEY,
GID VARCHAR(20),
CUSTID VARCHAR(20),
EID VARCHAR(20),
FNAME CHAR(20),
LNAME CHAR(20),
BIRTHPLACE CHAR(20),
STYLE CHAR(20),
FOREIGN KEY(GID) REFERENCES GALLERY(GID) ON DELETE CASCADE,
FOREIGN KEY (CUSTID) REFERENCES CUSTOMER(CUSTID) ON DELETE
CASCADE,
FOREIGN KEY(EID) REFERENCES EXHIBITION(EID) ON DELETE CASCADE);

ALTER TABLE ARTWORK ADD ARTISTID VARCHAR(20);

ALTER TABLE ARTWORK
ADD FOREIGN KEY (ARTISTID) REFERENCES ARTIST(ARTISTID) ON DELETE
CASCADE;
```



```
mysql> desc artist;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| artistid   | varchar(20) | NO   | PRI | NULL    |       |
| gid        | varchar(20) | YES  | MUL | NULL    |       |
| custid     | varchar(20) | YES  | MUL | NULL    |       |
| eid        | varchar(20) | YES  | MUL | NULL    |       |
| fname1     | char(25)    | YES  |     | NULL    |       |
| lname1     | char(25)    | YES  |     | NULL    |       |
| birthplace | char(25)    | YES  |     | NULL    |       |
| style      | char(25)    | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

## 6. CREATE CONTACTS TABLE

```
CREATE TABLE CONTACTS
(CUSTID VARCHAR(20),
PHONE VARCHAR(12),
FOREIGN KEY (CUSTID) REFERENCES CUSTOMER(CUSTID) ON DELETE
CASCADE);
```

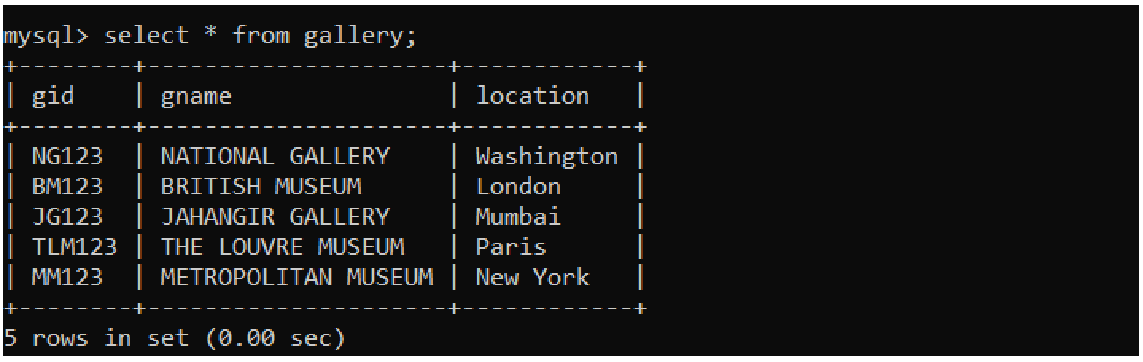


```
mysql> desc contacts;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| CUSTID | varchar(20) | YES  | MUL | NULL    |       |
| PHONE  | varchar(12) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

## 4.6 INSERTION OF TUPLES

### 1. INSERTION OF GALLERY TABLE

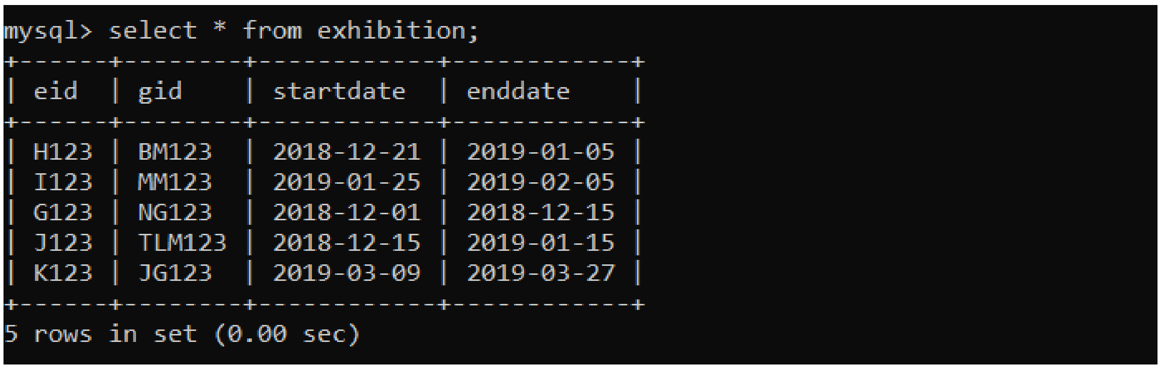
```
INSERT INTO GALLERY VALUES('NG123','National Gallery', 'Washington');
INSERT INTO GALLERY VALUES('BM123','British Museum', 'London');
INSERT INTO GALLERY VALUES('JG123','Jahangir Gallery', 'Mumbai');
INSERT INTO GALLERY VALUES('TLM123','The Louvre Museum', 'Paris');
INSERT INTO GALLERY VALUES('MM123','Metropolitan Museum', 'New York');
```



```
mysql> select * from gallery;
+--------+---------------------+------------+
| gid    | gname               | location   |
+--------+---------------------+------------+
| NG123  | NATIONAL GALLERY    | Washington |
| BM123  | BRITISH MUSEUM      | London     |
| JG123  | JAHANGIR GALLERY    | Mumbai     |
| TLM123 | THE LOUVRE MUSEUM   | Paris      |
| MM123  | METROPOLITAN MUSEUM | New York   |
+--------+---------------------+------------+
5 rows in set (0.00 sec)
```

### 2. INSERTION OF EXHIBITION TABLE

```
INSERT INTO EXHIBITION VALUES('G123','NG123','2018-12-01','2018-12-15');
INSERT INTO EXHIBITION VALUES('H123','BM123','2018-12-21','2019-01-05');
INSERT INTO EXHIBITION VALUES('I123','MM123','2019-01-25','2019-02-05');
INSERT INTO EXHIBITION VALUES('J123','TLM123','2018-12-15','2019-01-15');
INSERT INTO EXHIBITION VALUES('K123','JG123','2019-03-09','2019-03-27');
```

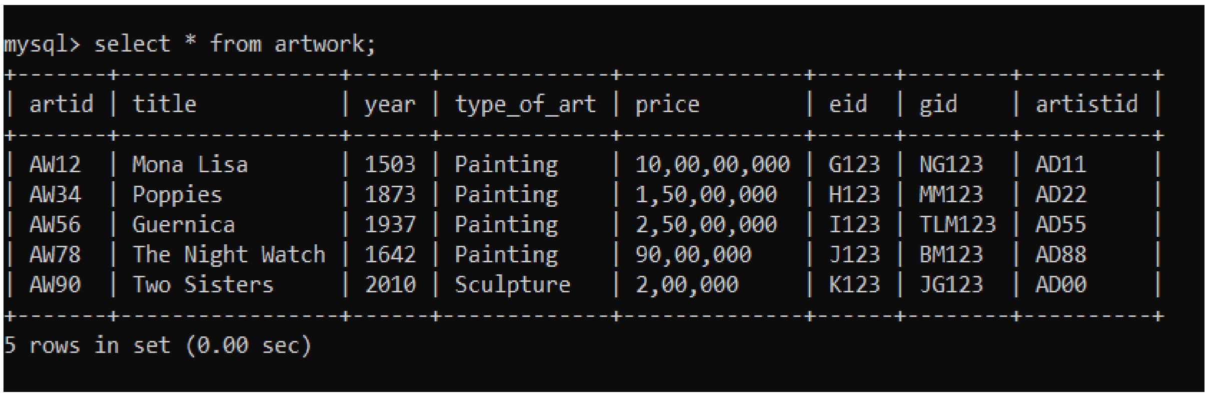


```
mysql> select * from exhibition;
+------+--------+------------+------------+
| eid  | gid    | startdate  | enddate    |
+------+--------+------------+------------+
| H123 | BM123  | 2018-12-21 | 2019-01-05 |
| I123 | MM123  | 2019-01-25 | 2019-02-05 |
| G123 | NG123  | 2018-12-01 | 2018-12-15 |
| J123 | TLM123 | 2018-12-15 | 2019-01-15 |
| K123 | JG123  | 2019-03-09 | 2019-03-27 |
+------+--------+------------+------------+
5 rows in set (0.00 sec)
```

### 3. INSERTION OF ARTWORK TABLE

```
INSERT INTO ARTWORK
VALUES('AW12','Mona Lisa','1503','Painting','10,00,00,000','G123','NG123','AD11');
INSERT INTO ARTWORK
VALUES('AW34','Poppies','1873','Painting','1,50,00,000','H123','MM123','AD22');
INSERT INTO ARTWORK
VALUES('AW56','Guernica','1937','Painting','2,50,00,000','I123','TLM123','AD55');
```

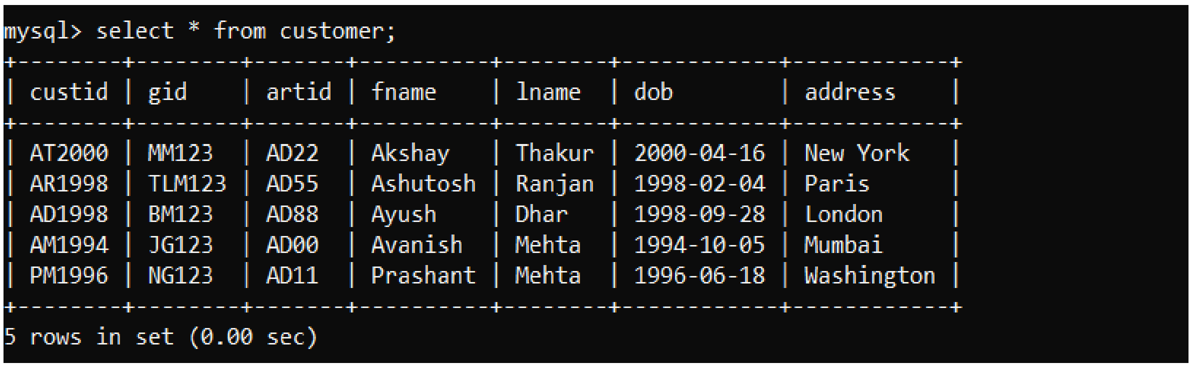
INSERT INTO ARTWORK
VALUES('AW78','The Night Watch','1642','Painting','90,00,000','J123','BM123','AD88');
INSERT INTO ARTWORK
VALUES('AW90','Two Sisters','2010','Sculpture','2,00,000','K123','JG123','AD00');



```
mysql> select * from artwork;
+-------+-----------------+------+-------------+--------------+------+--------+----------+
| artid | title           | year | type_of_art | price        | eid  | gid    | artistid |
+-------+-----------------+------+-------------+--------------+------+--------+----------+
| AW12  | Mona Lisa       | 1503 | Painting    | 10,00,00,000 | G123 | NG123  | AD11     |
| AW34  | Poppies         | 1873 | Painting    | 1,50,00,000  | H123 | MM123  | AD22     |
| AW56  | Guernica        | 1937 | Painting    | 2,50,00,000  | I123 | TLM123 | AD55     |
| AW78  | The Night Watch | 1642 | Painting    | 90,00,000    | J123 | BM123  | AD88     |
| AW90  | Two Sisters     | 2010 | Sculpture   | 2,00,000     | K123 | JG123  | AD00     |
+-------+-----------------+------+-------------+--------------+------+--------+----------+
5 rows in set (0.00 sec)
```

## 4. INSERTION OF CUSTOMER TABLE

INSERT INTO CUSTOMER VALUES
('AT2000','MM123','AD22','Akshay','Thakur','2000-04-16','New York');
INSERT INTO CUSTOMER
VALUES('AR1998','TLM123','AD55','Ashutosh','Ranjan','1998-02-04','Paris');
INSERT INTO CUSTOMER
VALUES('AD1998','BM123','AD88','Ayush','Dhar','1998-09-28','London');
INSERT INTO CUSTOMER
VALUES('AM1994','JG123','AD00','Avanish','Mehta','1994-10-05','Mumbai');
INSERT INTO CUSTOMER VALUES
('PM1996','NG123','AD11','Prashant','Mehta','1996-06-18','Washington');

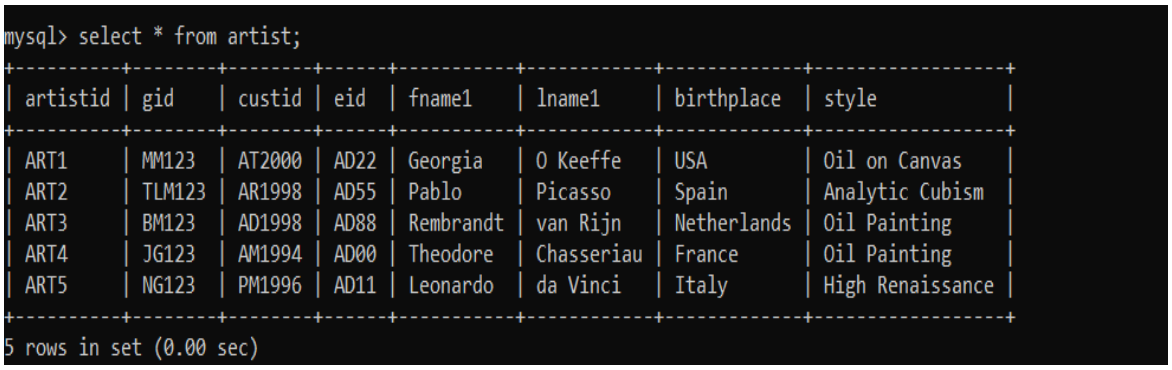


```
mysql> select * from customer;
+--------+--------+-------+----------+--------+------------+------------+
| custid | gid    | artid | fname    | lname  | dob        | address    |
+--------+--------+-------+----------+--------+------------+------------+
| AT2000 | MM123  | AD22  | Akshay   | Thakur | 2000-04-16 | New York   |
| AR1998 | TLM123 | AD55  | Ashutosh | Ranjan | 1998-02-04 | Paris      |
| AD1998 | BM123  | AD88  | Ayush    | Dhar   | 1998-09-28 | London     |
| AM1994 | JG123  | AD00  | Avanish  | Mehta  | 1994-10-05 | Mumbai     |
| PM1996 | NG123  | AD11  | Prashant | Mehta  | 1996-06-18 | Washington |
+--------+--------+-------+----------+--------+------------+------------+
5 rows in set (0.00 sec)
```

## 5. INSERTION OF ARTIST TABLE

INSERT INTO ARTIST
VALUES('ART1','MM123','AT2000','AD22','Georgia','O Keeffe','USA','Oil on Canvas');
INSERT INTO ARTIST
VALUES('ART2','TLM123','AR1998','AD55','Pablo','Picasso','Spain','Analytic Cubism');
INSERT INTO ARTIST VALUES
('ART3','BM123','AD1998','AD88','Rembrandt','van Rijn','Netherlands','Oil Painting');

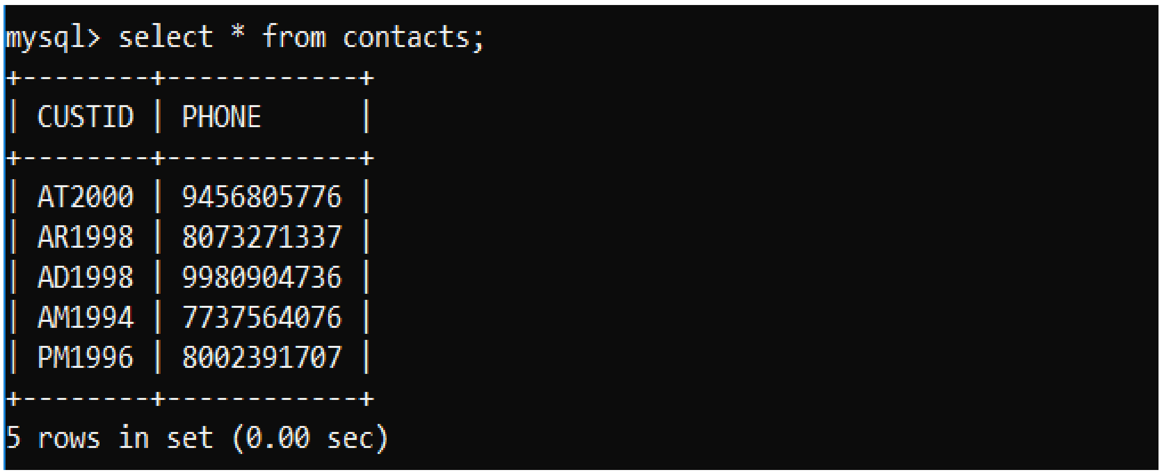
INSERT INTO ARTIST
VALUES('ART4','JG123','AM1994','AD00','Theodore','Chasseriau','France','Oil Painting');
INSERT INTO ARTIST
VALUES('ART5','NG123','PM1996','AD11','Leonardo','da Vinci','Italy','High Renaissance');



```
mysql> select * from artist;
+----------+--------+--------+------+-----------+------------+-------------+------------------+
| artistid | gid    | custid | eid  | fname1    | lname1     | birthplace  | style            |
+----------+--------+--------+------+-----------+------------+-------------+------------------+
| ART1     | MM123  | AT2000 | AD22 | Georgia   | O Keeffe   | USA         | Oil on Canvas    |
| ART2     | TLM123 | AR1998 | AD55 | Pablo     | Picasso    | Spain       | Analytic Cubism  |
| ART3     | BM123  | AD1998 | AD88 | Rembrandt | van Rijn   | Netherlands | Oil Painting     |
| ART4     | JG123  | AM1994 | AD00 | Theodore  | Chasseriau | France      | Oil Painting     |
| ART5     | NG123  | PM1996 | AD11 | Leonardo  | da Vinci   | Italy       | High Renaissance |
+----------+--------+--------+------+-----------+------------+-------------+------------------+
5 rows in set (0.00 sec)
```

## 6. INSERTION OF CONTACTS TABLE

INSERT INTO CONTACTS VALUES('AT2000', '9456805776');
INSERT INTO CONTACTS VALUES('AR1998', '8073271337');
INSERT INTO CONTACTS VALUES('AD1998', '9980904736');
INSERT INTO CONTACTS VALUES('AM1994', '7737564076');
INSERT INTO CONTACTS VALUES('PM1996', '8002391707');



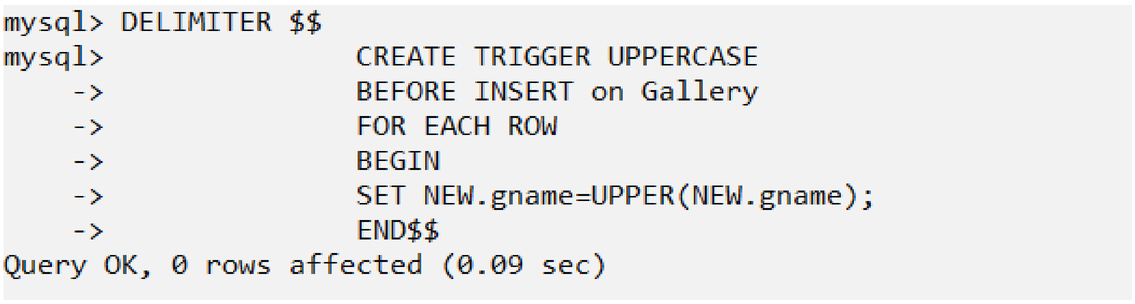
```
mysql> select * from contacts;
+--------+------------+
| CUSTID | PHONE      |
+--------+------------+
| AT2000 | 9456805776 |
| AR1998 | 8073271337 |
| AD1998 | 9980904736 |
| AM1994 | 7737564076 |
| PM1996 | 8002391707 |
+--------+------------+
5 rows in set (0.00 sec)
```

## 4.7 CREATION OF TRIGGERS

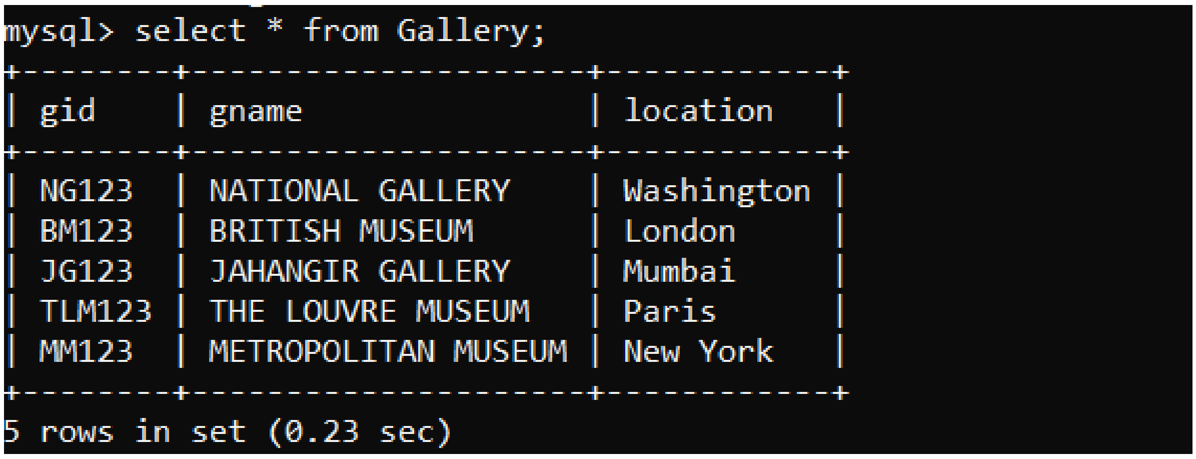
The trigger is made such that when a new record is inserted into a Gallery table, it automatically changes the lowercase name into uppercase in the backend.

**TRIGGER ON GALLERY TABLE TO CHANGING NAME TO UPPERCASE**

```
DELIMITER $$

CREATE TRIGGER UPPERCASE

BEFORE INSERT on Gallery

FOR EACH ROW

BEGIN

SET NEW.gname=UPPER(NEW.gname);

END$$
```



```
mysql> DELIMITER $$
mysql>                CREATE TRIGGER UPPERCASE
    ->                BEFORE INSERT on Gallery
    ->                FOR EACH ROW
    ->                BEGIN
    ->                SET NEW.gname=UPPER(NEW.gname);
    ->                END$$
Query OK, 0 rows affected (0.09 sec)
```



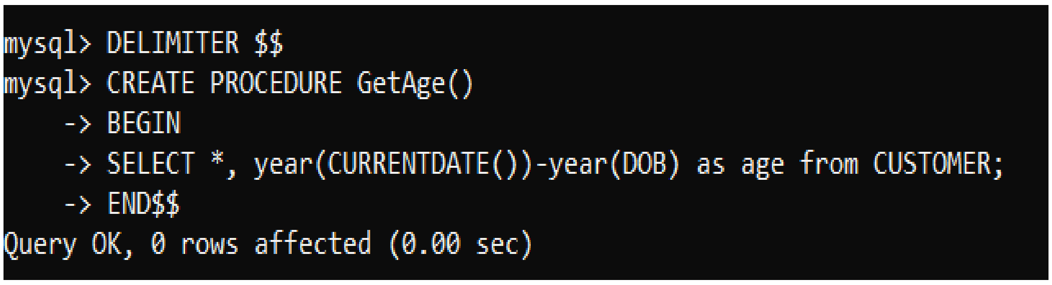
```
mysql> select * from Gallery;
+--------+----------------------+------------+
| gid    | gname                | location   |
+--------+----------------------+------------+
| NG123  | NATIONAL GALLERY     | Washington |
| BM123  | BRITISH MUSEUM       | London     |
| JG123  | JAHANGIR GALLERY     | Mumbai     |
| TLM123 | THE LOUVRE MUSEUM    | Paris      |
| MM123  | METROPOLITAN MUSEUM  | New York   |
+--------+----------------------+------------+
5 rows in set (0.23 sec)
```

## 4.8 CREATION OF STORED PROCEDURES

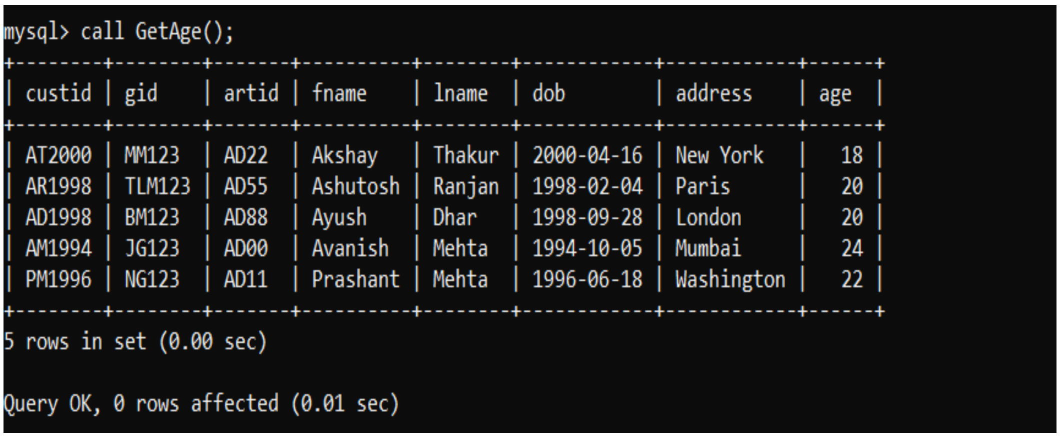
This stored procedure is used to find the age of the given customer using date of birth and current date.

### STORED PROCEDURE ON CUSTOMER TABLE TO FIND AGE

```
DELIMITER $$

CREATE PROCEDURE GetAge()

BEGIN

SELECT *, year(CURRENT_DATE())-year(DOB) as age from CUSTOMER;

END$$
```



```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE GetAge()
    -> BEGIN
    -> SELECT *, year(CURRENTDATE())-year(DOB) as age from CUSTOMER;
    -> END$$
Query OK, 0 rows affected (0.00 sec)
```



```
mysql> call GetAge();
+--------+--------+-------+----------+--------+------------+------------+------+
| custid | gid    | artid | fname    | lname  | dob        | address    | age  |
+--------+--------+-------+----------+--------+------------+------------+------+
| AT2000 | MM123  | AD22  | Akshay   | Thakur | 2000-04-16 | New York   |   18 |
| AR1998 | TLM123 | AD55  | Ashutosh | Ranjan | 1998-02-04 | Paris      |   20 |
| AD1998 | BM123  | AD88  | Ayush    | Dhar   | 1998-09-28 | London     |   20 |
| AM1994 | JG123  | AD00  | Avanish  | Mehta  | 1994-10-05 | Mumbai     |   24 |
| PM1996 | NG123  | AD11  | Prashant | Mehta  | 1996-06-18 | Washington |   22 |
+--------+--------+-------+----------+--------+------------+------------+------+
5 rows in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

# CHAPTER 5

# FRONT END DESIGN

## 5.1 SYSTEM DESIGN

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. System design could see it as the application of systems theory to product development. There is some overlap with the disciplines of system analysis, system architecture and systems engineering. If the border topic of product development "blends the perspective of marketing, design, and manufacturing into a single approach to product development," then design is the act of taking the marketing information and creating the design of the product to be manufactured. System design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

Until the 1990's systems design had a crucial and respected role in the data processing industry. In 1990's standardization of hardware and software resulted in the ability to build modular systems. The increasing importance of software running on generic platforms has enhanced the discipline of software engineering.

Object-oriented analysis and design methods are becoming the most widely used methods for computer system design. The UML has become the standard language in object-oriented analysis and design. It is widely used for modelling software systems and organizations.

System design is one of the most important phases of software development process. The purpose of the design is to plan the solution of a problem specified by the requirement documentation. In other words, the first step in the solution to the problem is the design of the project.

## 5.2 FRONT END CODE

### 5.2.1 CREATING FRONT END PAGE TO LINK ALL TABLES

```
<!DOCTYPE html>
<html>
<head>
<title>ART GALLERY DATABASE</title>
</head>
<body><h1>
<link rel="stylesheet" href='https://fonts.googleapis.com/css?family=Aclonica'>
<center>
<h1><font style="border: 13px groove powderblue;  border-radius: 7px" color=floralwhite face="Aclonica"; >ART GALLERY MANAGEMENT SYSTEM</font></h1>
</center>
<style> h1{
padding-top:  0px;
border-radius: 5px;
margin-bottom: -1px;
}
.column{}
body{background: url("https://images.pexels.com/photos/277054/pexels-photo-277054.jpeg?cs=srgb&dl=adult-architecture-art-277054.jpg&fm=jpg") no-repeat;
background-size:contain;}
ul{
margin:3pc;
padding:0.1px;
list-style:none;}
ul li{
float:left;
width:265px;
height:49px;
background-color:mediumorchid;
opacity:.8;
line-height:40px;
text-align:center;
font-size:25px;
margin-right:7px;
margin-left: 5px;
font-family: verdana;
font-weight: bold;
font-size: 35px;
}
ul li a{
text-decoration:underline;
text-align: center;
color:white; display:block;
padding-bottom: 7.5px; }
ul li a:hover{
background-color:mediumseagreen;
color: yellow;
font-family: sans-serif;}
ul li ul li{
display:none; }
ul li:hover ul li {
display:block;}
</style><br>
<ul><center>
<li><a href="gallery.html" >GALLERY</a></li>
<li><a href="exhibition.html">EXHIBITION</a></li>
<li><a href="artwork.html">ARTWORK</a></li>
<li><a href="customer.html">CUSTOMER</a></li>
<li><a href="artist.html">ARTIST</a></li>
```

```
<li><a href="contacts.html">CONTACTS</a></li>
</center>
 </ul> <br> </h1>
 <style>
 .example1 { height: 50px;
  overflow: hidden;
  position: relative;
  }
  .example1 h3 {
  font-size: 3em; color: #F7F9F9;
  position: absolute;
  width: 100%; height: 100%;
  margin-top: 0px;
  line-height: 50px;
  text-align: center;
  -moz-transform:translateX(100%);
  -webkit-transform:translateX(100%);
   transform:translateX(100%);
   -moz-animation: example1 15s linear infinite;
    -webkit-animation: example1 15s linear infinite;
    animation: example1 11s linear infinite; }
    @-moz-keyframes example1 {
    0%{ -moz-transform: translateX(100%); }
    100%{ -moz-transform: translateX(-100%); }
     }
    @-webkit-keyframes example1 {
     0%{ -webkit-transform: translateX(100%); }
     100%{ -webkit-transform: translateX(-100%); }
      }
      @keyframes example1 {
     0%{-moz-transform: translateX(100%);
  -webkit-transform: translateX(100%);
transform: translateX(100%);}
100% {-moz-transform: translateX(100%);
-webkit-transform: translateX(-100%);
transform: translateX(-100%);}
}</style>
<div class="example1">
<h3 style="font-family: Verdana">This is an ART GALLERY PROJECT!</h3>
</div>
<style>
* {box-sizing: border-box;}
.column{ float: left;
width: 33.33%;
padding: 1px; }
.row::after{ content: "";
clear: both; display: table;
}
</style></head>
</html>
```

### 5.2.2 CREATION OF NEXT PAGE AFTER SELECTION

```
<!DOCTYPE html>
<html>
<head>
        <title>GALLERY</title>
</head>
<body>
        <h1>
        <link rel="stylesheet"
href='https://fonts.googleapis.com/css?family=Aclonica'>
        <link rel="stylesheet"
href='https://fonts.googleapis.com/css?family=Lemon'>
```

```
<link rel="stylesheet" href='https://fonts.googleapis.com/css?family=Jockey
One'>

            <center>

            <h1 style="font-size: 40"><font style="border: 14px groove floralwhite;"
color=greenyellow face="Aclonica" ><u>GALLERY</font></h1><br>

</center>

       <style> body{ background: url("https://leeuwinestate.com.au/wp-
content/uploads/2015/07/Art-Gallery-2.jpg") no-repeat;

                        background-size:cover;

                        font-family:"Verdana";

                        text-color:white;

                        }

  </style>

  <b> <font style="font-family: Lemon" font color="white" size="10">

                                                SELECT ANY ONE OPTION FROM
BELOW:</font>

</b>

<style>

ul{

            margin:3pc;

            padding:0.1px;

            list-style:none;}

  ul li{

            float:initial;

            width:400px;  height:65px;

            background-color:mediumslateblue;

            opacity:.8;  line-height:30px;

            text-align:center;  font-size:35px;

            margin-bottom: 12px; }

    ul li a{

            text-decoration:underline;

            text-align: center;

            font-family: Jockey One;
            font-weight: bold;  font-size: 6;
            color:white;  display:block;
}
ul li a:hover{
             background-color: white;
             color: black;
            background-size: contain; }
ul li ul li{
            display:none; }
ul li:hover ul li{
            display:block; }
</style><br>
<ul>
            <li><a href="ginsert.html" >ADD VALUES INTO THE GALLERY
</a></li>
            <li><a href="gsearch.php">SEARCH VALUES FROM THE
                                        GALLERY</a></li>
            <li><a href="gdisplay.php">DISPLAY CONTENTS FROM THE
                                        GALLERY TABLE</a></li>
            <li><a href="gstored.php">STORED PROCEDURE OF THE
GALLERY TABLE</a></li>
            <li><a href="gdelete.php">DELETE VALUES FROM THE
                                        GALLERY</a></li> </h1>
<p style="font-family: arial"><a href="FrontEnd.html"><font
style="color:gold">GO BACK</font></a></p>
</body>
</html>
```

## 5.2.3.a INSERTION IN FRONT END HTML CODE

```
 <!DOCTYPE html>
<html>
<head>
   <title>Insertion in Gallery</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
h1{
   border: 10px solid grey;
   border-radius: 28px;
   padding: 19px;
}
body {
   font-family: Arial, Helvetica, sans-serif;
   background-color: white; }
* {
   box-sizing: border-box;}
.container {
   padding: 19px;  background-color: snow; }
input[type=text], input[type=text]
 {
   width: 50%; padding: 19px;
   margin: 5px 0 32px 0;
   display: inline-block;
   border-radius: 8px;
   border: 2px solid grey;
   background: #f1f1f0;
   font-weight: bold; font-size: 19px;
}
input[type=text]:focus, input[type=text]:focus {
   background-color: #ffffff; outline: none;}
```

```
hr {
    border: 1.5px solid #f1f1f1;
    margin-bottom: 35px;
}
.registerbtn{
    background-color: forestgreen;
    padding: 16px 10px;
    margin: 8px 20px 20px 50px;
    border-radius: 24px; cursor: pointer;
    width: 15%;  opacity: 0.7;
    align-content: center;
    font-family: "verdana";
    font-weight: bold; color: white;
    }
.registerbtn:hover {
    opacity: 1;
    background-color:forestgreen;}
 label{
     font-weight: bold;
      font-size: 20px;
      font-family: 'verdana';
         }
    </style></head>
    <body>
    <form ACTION="ginsert.php"METHOD="POST">
    <div class="container">
    <center><h1 style="color:#3366cc" size="30";><font style="border: 10px
solid grey; padding: 19px;">FILL THE FORM WITH PROPER
DETAILS</font></h1></center>
    <hr><center>
    <label for="G_ID"><b>Gallery ID</b></label><br>
    <input type="text" placeholder="Enter G_ID" name="G_ID" required><br>
    <label for="GNAME"><b>GNAME</b></label><br>
    <input type="text" placeholder="Enter Gallery Name" name="GNAME"
required><br>

    <label for="LOCATION"><b>GLOCATION</b></label><br>
    <input type="text" placeholder="Enter GLOCATION" name="LOCATION"
required><br>
    </hr>
    <button type="submit" class="registerbtn">SUBMIT</button>
    <button type="reset" class="registerbtn">RESET</button>
    </center> </div> </form> </body>
    </html>
```

### 5.2.3.b CONNECTIVITY OF THE PHP TO DATABASE TO INSERT A RECORD

```
<?php
 if(isset($_POST['G_ID']) && isset($_POST['GNAME']) &&
 isset($_POST['LOCATION'])):
 $gid = $_POST['G_ID'];
 $gname = $_POST['GNAME'];
 $location = $_POST['LOCATION'];
 $link = new mysqli('localhost','root','','art_gallery');
 if($link->connect_error)
 die('connection error: '.$link->connect_error);
$sql3 = "INSERT INTO GALLERY(gid, gname, location)
            VALUES('".$gid."', '".$gname."', '".$location."')";
 $result = $link->query($sql3);
 if($result > 0):
 echo 'Successfully Inserted into GALLERY table.';
 else:
 echo 'Unable to post';
 endif;
```

```
$link->close();
die();
endif;
?>
```

### 5.2.4 SEARCHING OF VALUES FROM FRONT END

```
<html>
<head>
   <title>Search Gallery</title>
</head>
<style> table{
      border-collapse: collapse;
      width: 60%;
      padding: 150px;
      margin-left: 280px; }
   th, td {
         text-align: center;
         padding: 8px;
         border-radius: 12px;
         }
   tr:nth-child(even) {
      background-color: #f2f2f2;
      font-family: "arial";
   font-weight: bold;  }
   th {  background-color: mediumslateblue;
   color: white; font-family:  "verdana";
   font-weight: bold;  }
input[type=text] { width: 119px;
   box-sizing: border-box;
   border: 2px solid #ccc;
   border-radius: 9px; font-size: 16px;
   background-color: white;
```

```
    background-position: 10px 10px;
    background-repeat: no-repeat;
    padding: 22px 20px 22px 10px;
    -webkit-transition: width 0.4s ease-in-out;
    transition: width 0.4s ease-in-out;
    font-weight: bold; font-size: 30px; }
input[type=text]:focus {
    width: 60%;  }
div{ font-family: "verdana";
    font-weight: bold;
    font-size: 30px;
    font-style: bold;
    margin-left:25px;
    margin-top: 35px; }
.btn{ background-color: forestgreen;
    color: white;  padding: 16px 10px;
    margin: 8px 20px 20px 50px;
    border-radius: 24px;
    cursor: pointer;
    width: 10%; opacity: 0.7;
    align-content: center;
    font-family: "verdana";
    font-weight: bold;
}
.btn:hover {
    opacity: 1;  background-color:forestgreen; }
 b{ font-family: "verdana";
    background-color: lightcyan;
    color: black;
    margin-left:80px;
    border-radius: 8px;
```

```
  text-align: center;
   font-size: 30px;
   width: 85%; }
   span{ font-family: "verdana";
   background-color: lightcyan;
   color: black; margin-top:4px;
   border-radius: 8px; text-align: center;
   font-size: 30px;  margin-left:0px;
   width: 35%;  font-weight: bold;}
</style>
<body style="background-color: lavender">
   <h1><center><font style="border:9px solid grey" face="arial">SEARCH
FROM GALLERY TABLE </font></center></h1>
<form action="gsearch.php" method="post">
            <div>Enter Gallery ID:<input type="text" name="G_ID"
placeholder="G_ID"><br></div><br><br>
            <button type="submit" value ="Find"
class="btn">SEARCH</button></form>
<?php
$host="localhost";
$user="root";
$password="";
$con= new mysqli($host,$user,$password,"art_gallery");
if ($con->connect_error) {
                        die("Connection failed: " . $con->connect_error);}
if ($_SERVER["REQUEST_METHOD"] == "POST")
{
            $n=$_POST['G_ID'];
            echo "<b><br>Entered Gallery ID is $n<br></b>";
            $sql="select * from gallery where gid='$n'";
            $result = $con->query($sql);
if ($result->num_rows > 0) {
```

```
echo "<b><br>Search Successful<br><br></b>";

echo
"<br><br><br><br><table><tr><th>G_ID</th><th>GNAME</th><th><br>LOC
ATION<br></br></th></tr>";

while($row = $result->fetch_assoc())

{

        echo "<tr><td>" . $row["gid"]. "</td><td>" .$row["gname"].
"</td><td><br>"

         . $row["location"]. "<br></br></td></tr>"; }

 echo "</table>";

                    } else {

 echo "<span><br><br>OPPS!!! Search Unsuccessful!<br><br>There is no such
Gallery    ID exists. Please Enter Correct Gallery ID and Search Again.</span>"; }

 }

  $con->close();

  ?>

 </body>

 </html>
```

## 5.2.5 DISPLAYING VALUES FROM FRONT END

```
<!DOCTYPE html>

 <html>

 <head>

  <title>Display Gallery</title>

  <style>

  table {

    border-collapse: collapse;

    width: 100%;  color: #588c7e;

    font-family: monospace;

    font-size: 25px; text-align: left;

    font-family:"Verdana";

    font-weight: bold;

    text-align: center;
  border-radius: 14px; }
 th{
  background-color: mediumpurple;
  color: white;
  border-radius: 14px; }
 h1{
   font-family: "Arial";
   font-size: 50px;
   border: 9px solid grey;
   border-radius: 17px;
    color: black; }
 td{
   padding: 12px;
   border-radius: 14px; }
 tr:nth-child(even) { background-color: #f2f2f2;
   border-radius: 14px;}
 </style>
</head>
<body style="background-color: lavender">
  <h1><center><font style="border:9px solid grey"> DISPLAY CONTENTS /\/
GALLERY TABLE </font></center></h1>
 <table>
 <tr>
  <th><br>Gallery ID<br><br></th>
  <th><br>GName<br><br></th>
  <th><br>Location<br><br></th><br><br>
 </tr>
 <?php
$con = mysqli_connect("localhost", "root", "", "art_gallery");
 if ($con->connect_error) {
   die("Connection failed: " . $con->connect_error); }
 $sql = "SELECT * from GALLERY";
 mysqli_query($con,$sql);
 if ($result = mysqli_query($con, $sql)){
  while($row = $result->fetch_assoc())
   {
   echo "<tr><td>" . $row["gid"]. "</td><td>" . $row["gname"]. "</td><td><br>".
$row["location"]. "<br></br></td></tr>";
   }
   echo "</table>";
   }else {
   echo "0 results"; }
$con->close();
?>
</table></body></html>
```

### 5.2.6 DELETION OF VALUES FROM FRONT END

```
<html>
<head>
<title>Delete from Gallery</title>
</head>
<style>b{
font-size: 30px;
font-family: 'Arial';
padding: 27px 62px;}
input[type=text] {
width: 120px;
box-sizing: border-box;
border: 2px solid #ccc;
border-radius: 9px;
font-size: 16px;
background-color: white;
background-position: 10px 10px;
background-repeat: no-repeat;
padding: 22px 20px 22px 10px;
-webkit-transition: width 0.4s ease-in-out;
transition: width 0.4s ease-in-out;
font-weight: bold;
font-size: 30px; }
input[type=text]:focus {
width: 50%; }
div{
font-family: 'Verdana';
font-size: 35px;
font-weight:bold;
margin-left:25px;
margin-top: 35px; }
.btn{ background-color: forestgreen;
color: white;
padding: 16px 10px;
margin: 8px 20px 20px 50px;
border-radius: 24px;
cursor: pointer;
width: 10%; opacity: 0.7;
align-content: center;
font-family: "verdana";
font-weight: bold; }
.btn:hover {
 opacity: 1;
 background-color:forestgreen;
 }
</style>
<body style="background-color: lavenderblush">
```

```
<h1><center><font style="border:9px solid grey" face="arial">DELETE FROM
GALLERY TABLE </font></center></h1>

<form action="gdelete.php" method="POST">

<div>Enter Gallery ID:<input type="text" name="G_ID"
placeholder="G_ID"><br></div><br><br>

<button type="submit" value ="Delete" class="btn">DELETE</button>

<button type="reset" value ="Reset" class="btn">RESET</button>

</form>

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "art_gallery";

$con = new mysqli($servername, $username, $password, $dbname);

if ($_SERVER["REQUEST_METHOD"] == "POST"){

$a=$_POST['G_ID'];

if($a!=""){

$sql1 = "SELECT * from Gallery where gid='$a'";

$result = mysqli_query($con,$sql1);

 if(mysqli_num_rows($result)>0){

$sql3="DELETE from Gallery where gid='$a'";

mysqli_query($con,$sql3);

echo "<b>Record with G_ID = $a is deleted successfully.<b>";}

else{

echo "<b>!!!Error in Deleting Record!!!<br> Record '$a' was not found in
database.<b>";

} }else{

echo "<b>G_ID Field is Empty</b>";}

$con->close();}

?>

</body>

</html>
```

# CHAPTER 6

# TESTING

This chapter gives the outline of all testing methods that are carried out to get a bug free system. Quality can be achieved by testing the product using different techniques at different phases of the project development. The purpose of testing is to discover error. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components sub-assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

## 6.1 TESTING PROCESS: -

Testing is an integral part of software development. Testing process certifies whether the product that is developed complies with the standards that I was designed to. Testing process involves building of test cases against which the product has to be used.

## 6.2 TESTING OBJECTIVE: -

The main objectives of testing process are as follows.

1. Testing is a process of executing a program with the intent of finding an error.
2. A good test case is one that has high probability of finding undiscovered error.
3. A successful test is one that uncovers the undiscovered error.

## 6.3 TEST CASE

The test cases provided here test the most important features of the project.

### Test cases for the project

| SNo | TEST INPUT | EXPECTED RESULT | OBSERVED RESULT | REMARKS |
|---|---|---|---|---|
| 1 | INSERT A RECORD | New tuple should be inserted | Query OK 1 row effected or inserted | PASS |
| 2 | SEARCH A RECORD | Display the record | Required record displayed | PASS |
| 3 | DISPLAY RECORD | Display the record | record displayed | PASS |
| 4 | DELETE A RECORD | Delete the record | Query OK 1 row affected or Row Deleted | PASS |
| 5 | CREATE TRIGGER | Trigger Created | Query OK Trigger Created | PASS |
| 6 | CREATE STORED PROCED URES | Stored procedures created | Query OK Stored Procedures Created | PASS |

# CHAPTER 7

## RESULTS

This section describes the screens of "Art Gallery Database". The snapshots are shown below for each module.

### SNAPSHOTS

- This is the main page that shows all the operations which are present in Art Gallery Database.

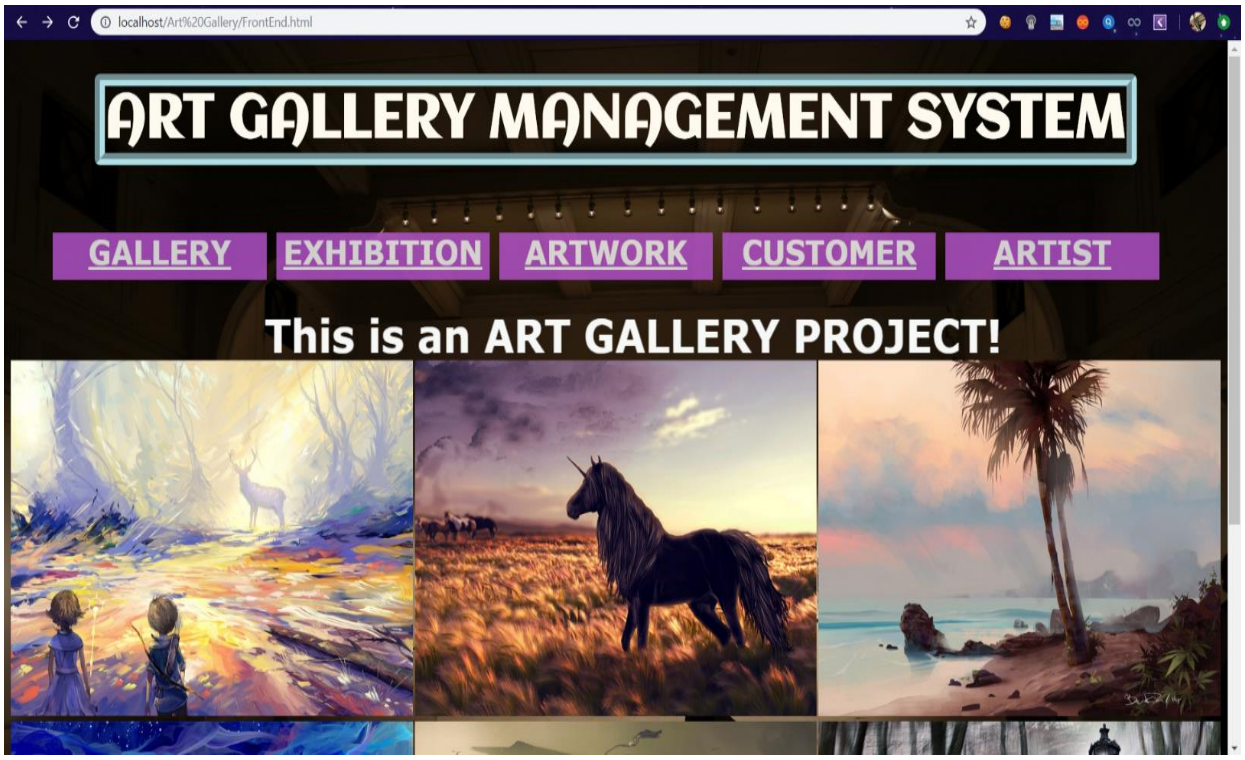
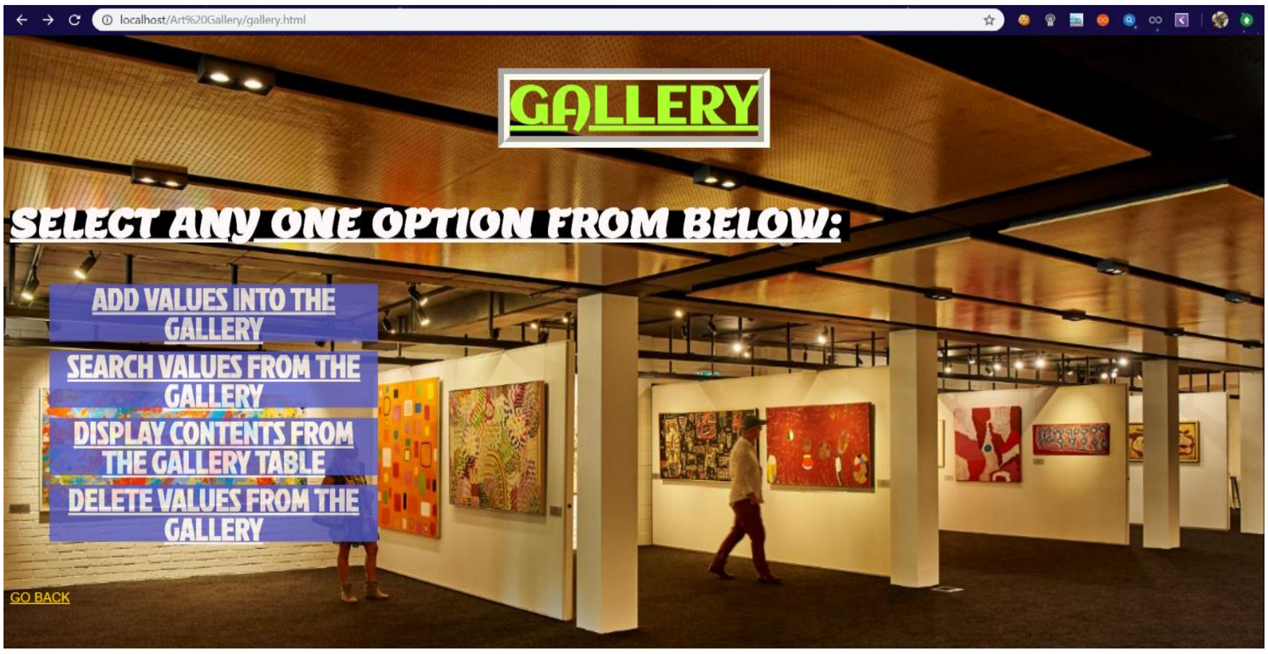

localhost/Art%20Gallery/FrontEnd.html
ART GALLERY MANAGEMENT SYSTEM
GALLERY
EXHIBITION
ARTWORK
CUSTOMER
ARTIST
This is an ART GALLERY PROJECT!

FIGURE 7.1 ART GALLERY DATABASE MAIN PAGE

- The selection page is the next displayed as soon as the table is selected in Main Page. Gallery table contains Insert, Search, Display and Delete tables where values can be inserted, deleted, etc.


localhost/Art%20Gallery/gallery.html

GALLERY

SELECT ANY ONE OPTION FROM BELOW:

ADD VALUES INTO THE GALLERY

SEARCH VALUES FROM THE GALLERY

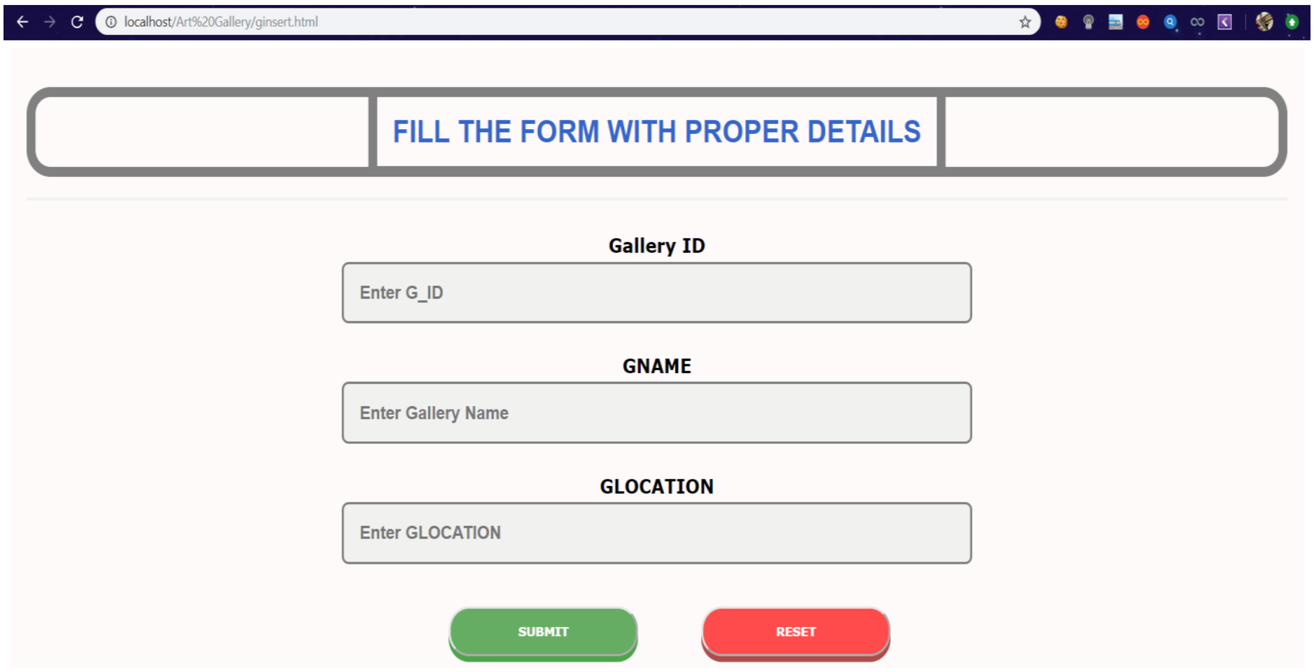
DISPLAY CONTENTS FROM THE GALLERY TABLE

DELETE VALUES FROM THE GALLERY

GO BACK

FIGURE 7.2 GALLERY TABLE SELECTION PAGE

- This snapshot shows the insertion page of Gallery Table. This front end page supports the insertion of all the attributes in Gallery table like GID, GName and Location.


localhost/Art%20Gallery/ginsert.html

FILL THE FORM WITH PROPER DETAILS

Gallery ID

Enter G_ID

GNAME

Enter Gallery Name

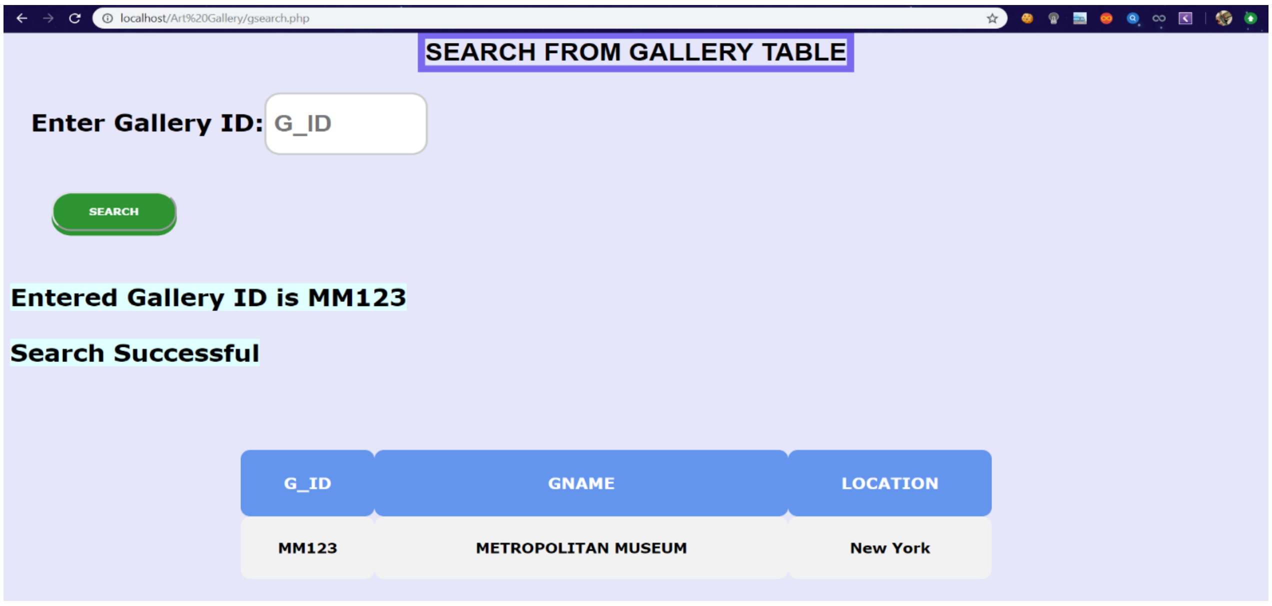
GLOCATION

Enter GLOCATION

SUBMIT RESET

FIGURE 7.3: INSERTION PAGE OF GALLERY TABLE

- This snapshot contains all the details of given GID by using Search method. Search table of Gallery contains the values like GID, GName and Location where it is searched by GID which is a primary key for this table.



localhost/Art%20Gallery/gsearch.php

SEARCH FROM GALLERY TABLE

Enter Gallery ID: G_ID

SEARCH

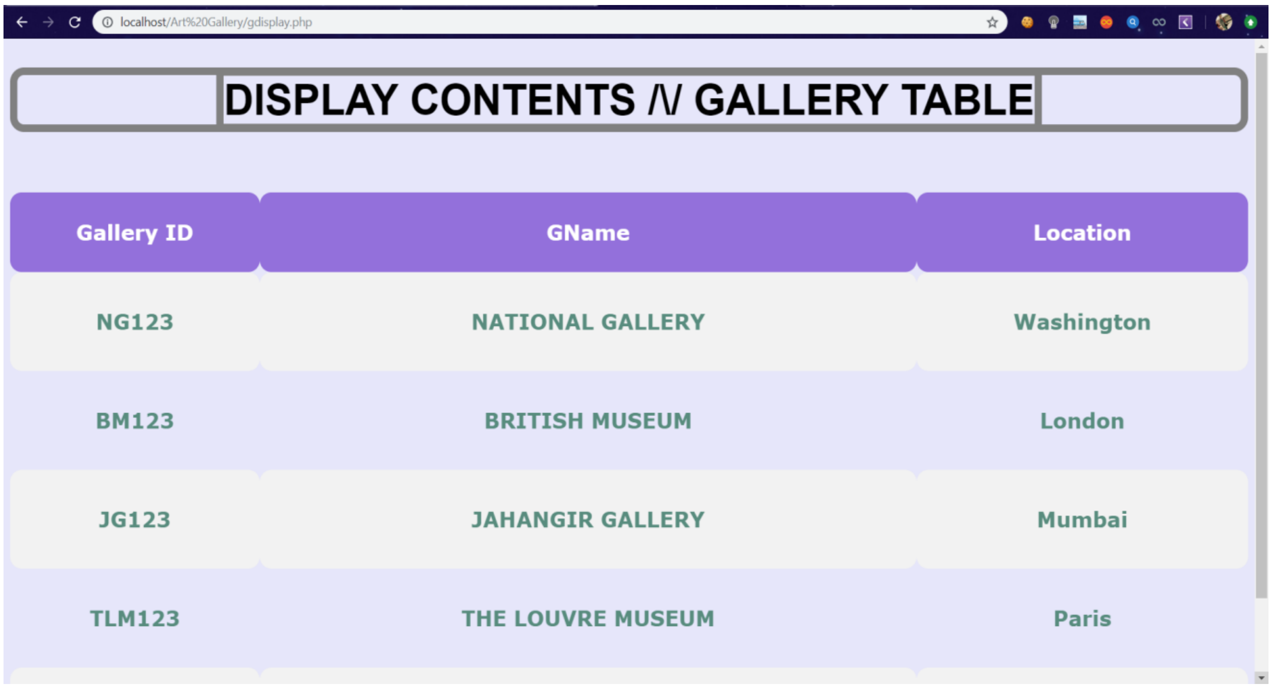
Entered Gallery ID is MM123

Search Successful

| G_ID | GNAME | LOCATION |
|---|---|---|
| MM123 | METROPOLITAN MUSEUM | New York |

FIGURE 7.4: SEARCH PAGE OF GALLERY TABLE

- This snapshots displays all the values entered in that table. Here, Gallery table is displayed on frontend page by using proper display query.



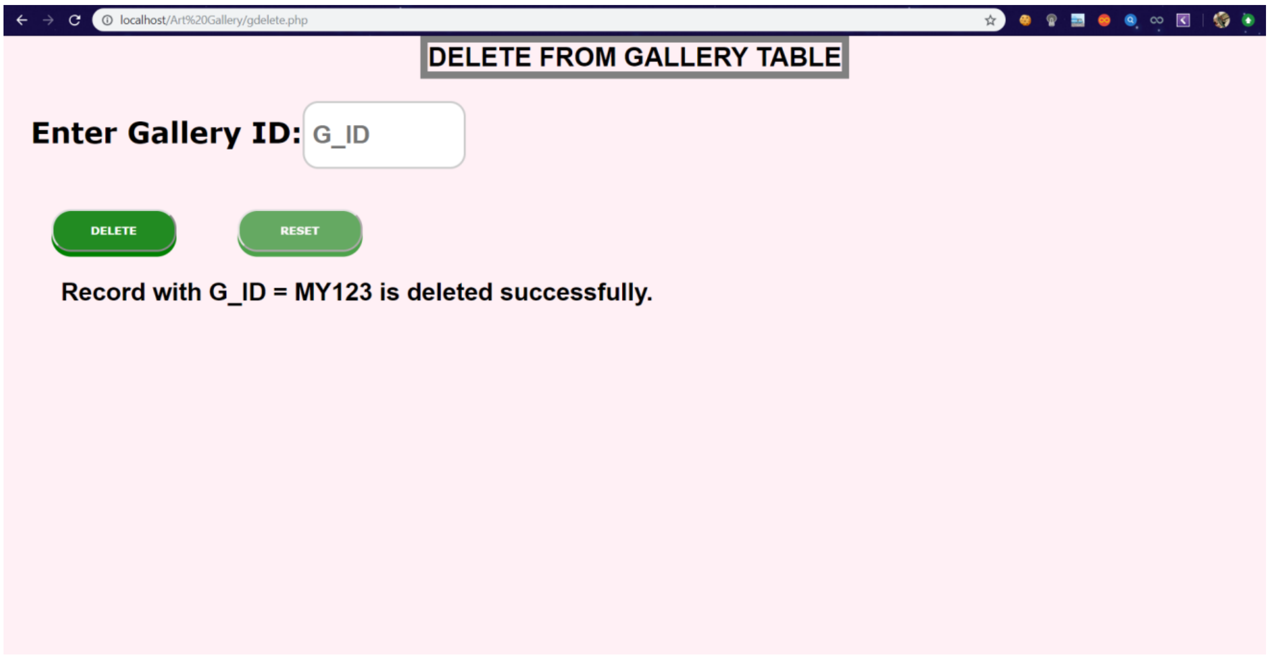
localhost/Art%20Gallery/gdisplay.php

DISPLAY CONTENTS /N GALLERY TABLE

| Gallery ID | GName | Location |
|---|---|---|
| NG123 | NATIONAL GALLERY | Washington |
| BM123 | BRITISH MUSEUM | London |
| JG123 | JAHANGIR GALLERY | Mumbai |
| TLM123 | THE LOUVRE MUSEUM | Paris |

FIGURE 7.5: DISPLAY PAGE OF GALLERY TABLE

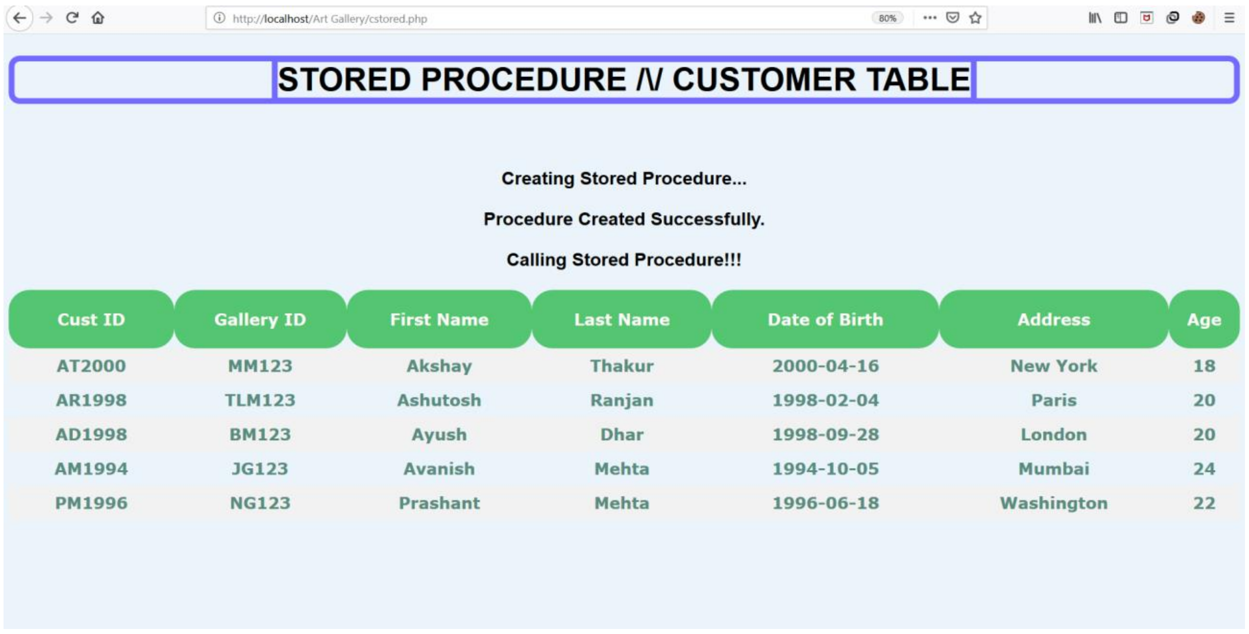
- This snapshot shows the working status of Delete Page of Gallery Table. In this page, GID is used as a parameter to delete certain record with all of their attributes in that rows. If the entered value is present in database then the value will be deleted.


localhost/Art%20Gallery/gdelete.php
DELETE FROM GALLERY TABLE
Enter Gallery ID:
G_ID
DELETE
RESET

Record with G_ID = MY123 is deleted successfully.

FIGURE 7.6: DELETION PAGE OF GALLERY TABLE

- This snapshot shows the working status of Stored Procedure of Customer table. By using stored procedure, we're calculating the Age of Customer using Date of Birth as parameter.


http://localhost/Art Gallery/cstored.php
STORED PROCEDURE /\/ CUSTOMER TABLE

Creating Stored Procedure...

Procedure Created Successfully.

Calling Stored Procedure!!!

| Cust ID | Gallery ID | First Name | Last Name | Date of Birth | Address | Age |
|---|---|---|---|---|---|---|
| AT2000 | MM123 | Akshay | Thakur | 2000-04-16 | New York | 18 |
| AR1998 | TLM123 | Ashutosh | Ranjan | 1998-02-04 | Paris | 20 |
| AD1998 | BM123 | Ayush | Dhar | 1998-09-28 | London | 20 |
| AM1994 | JG123 | Avanish | Mehta | 1994-10-05 | Mumbai | 24 |
| PM1996 | NG123 | Prashant | Mehta | 1996-06-18 | Washington | 22 |

FIGURE 7.7: STORED PROCEDURE PAGE OF CUSTOMER TABLE

# CONCLUSION

A database was created for a market that can use it for keeping track on art gallery Galleries are divided into many art galleries. Galleries have different names, locations, etc. Each gallery will have different exhibitions and each exhibition will have a start and end date. The galleries will have different artist displaying their artwork. The model can also be adapted to meet other purposes and thus be used for other projects. The database structure is quite simple, which makes it easy for also other programmers to understand it. In conclusion, a database is a far more efficient mechanism to store and organize data than spreadsheets it allows for a centralized facility that can easily be modified and quickly shared among multiple users. Having a web based front end removes the requirement of users having to understand and use a database directly, and allows users to connect from anywhere with an internet connection and a basic web browser. It also allows the possibility of queries to obtain information for various surveys. Due to the number of users reading and modifying student data in the department, it is an ideal use for such a system.

# REFERENCES

I. Fundamentals of Database System, 7th Edition

-By Elmasri Ramez and Navathe Shamkanth

II. PHP and MySQL Web Development

-By Luke Welling and Laura Thompson

III. HTML & CSS: Design and Build Web Sites

-By John Duckett

IV. For Front End Code and CSS styling

a. https://www.w3schools.com/html

b. https://www.stackoverflow.com

c. https://www.tutorialspoint.com

V. For MySQL references

a. https://www.youtube.com

b. https://www.udemy.com