SQL MODULE

LAB - 1

Bandi Amrutha

AF0366376

Questions

Create a database with the name StudentManagementSystem.

Create a table with named Student with attributes:

- StudentID (Primary Key)
- FirstName
- LastName
- DateOfBirth
- Gender
- Email
- Phone

Create a table with name Course with attributes:

- CourseID (Primary Key)
- CourseTitle
- Credits

Create a table with named Instructor with attributes:

- InstructorID (Primary Key)
- FirstName
- LastName
- Email

Create a table with named Enrollment with attributes:

- EnrollmentID (Primary Key)
- EnrollmentDate
- StudentID(Foreign key)
- CourseID(Foreign Key)
- InstructorID(Foreign key)

Create a table with named Score with attributes:

- ScoreID (Primary Key)
- CourseID (Foreign key)
- StudentID (Foreign Key)
- DateOfExam
- CreditObtained

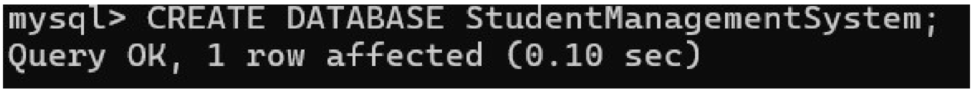
Create a table with named Feedback with attributes:

- FeedbackID (Primary Key)
- StudentID (Foreign key)

- Date
- InstructorName
- Feedback

Create a database with the name StudentManagementSystem.

Code :

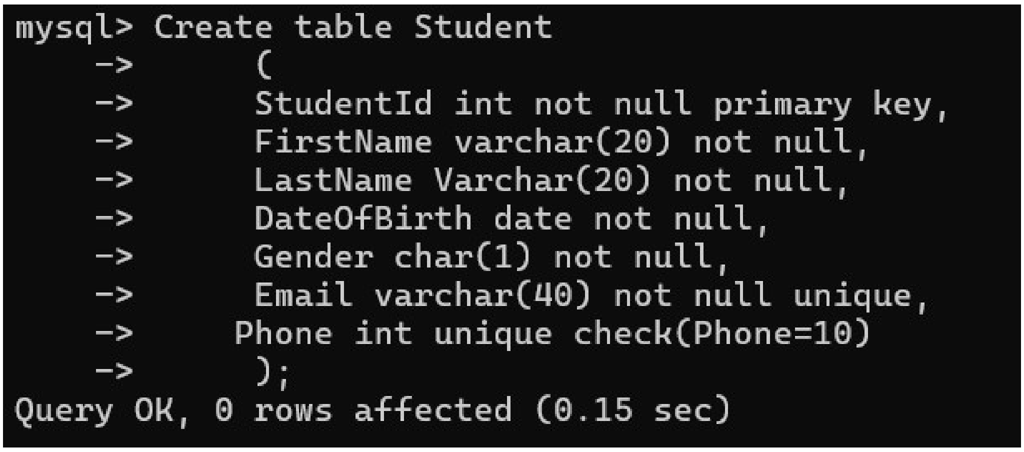


```
mysql> CREATE DATABASE StudentManagementSystem;
Query OK, 1 row affected (0.10 sec)
```

Create a table with named Student with attributes:

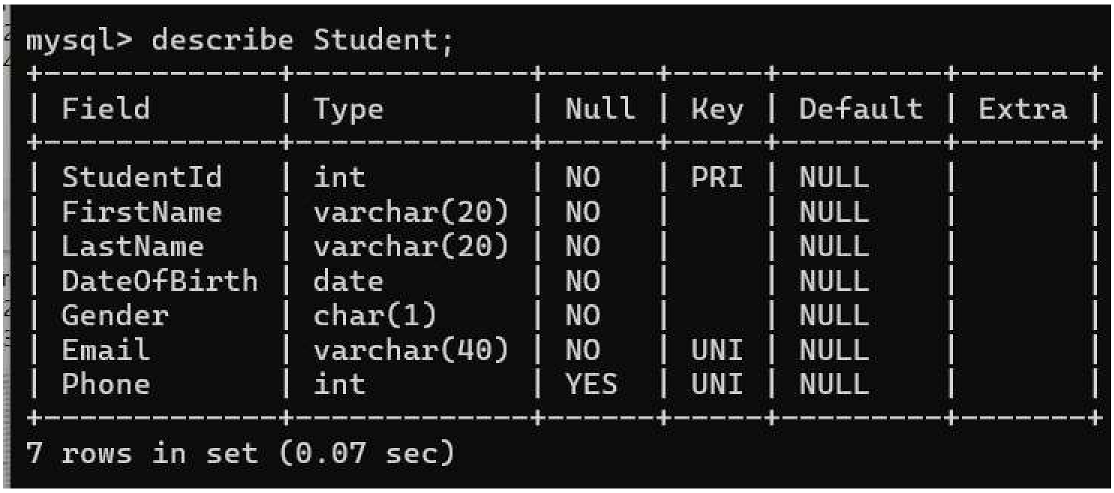
- StudentID (Primary Key)
- FirstName
- LastName
- DateOfBirth
- Gender
- Email
- Phone

Code :



```
mysql> Create table Student
    ->      (
    ->      StudentId int not null primary key,
    ->      FirstName varchar(20) not null,
    ->      LastName Varchar(20) not null,
    ->      DateOfBirth date not null,
    ->      Gender char(1) not null,
    ->      Email varchar(40) not null unique,
    ->     Phone int unique check(Phone=10)
    ->      );
Query OK, 0 rows affected (0.15 sec)
```

Output :

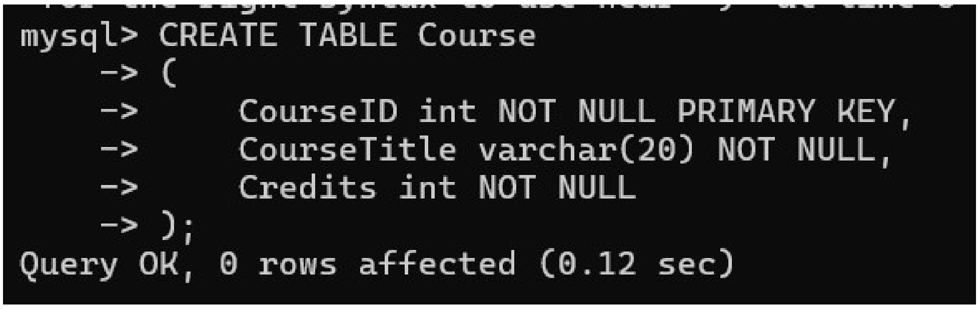


```
mysql> describe Student;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| StudentId   | int         | NO   | PRI | NULL    |       |
| FirstName   | varchar(20) | NO   |     | NULL    |       |
| LastName    | varchar(20) | NO   |     | NULL    |       |
| DateOfBirth | date        | NO   |     | NULL    |       |
| Gender      | char(1)     | NO   |     | NULL    |       |
| Email       | varchar(40) | NO   | UNI | NULL    |       |
| Phone       | int         | YES  | UNI | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
7 rows in set (0.07 sec)
```

Create a table with name Course with attributes:

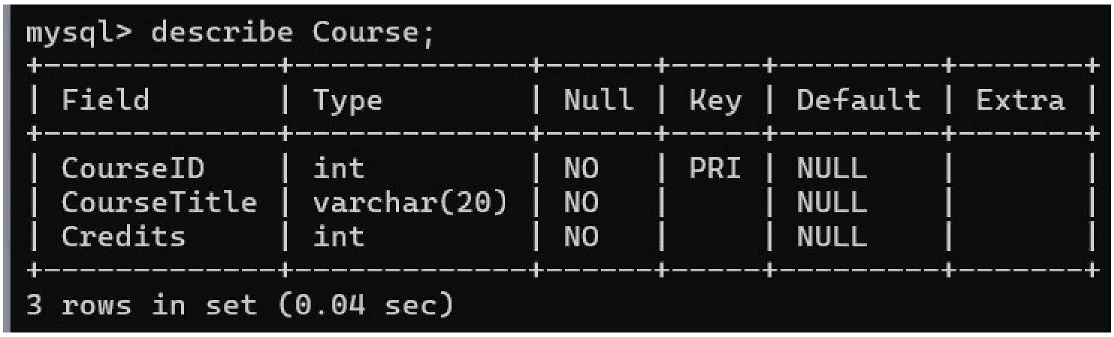
- CourseID (Primary Key)
- CourseTitle
- Credits

Code :



```
mysql> CREATE TABLE Course
    -> (
    ->     CourseID int NOT NULL PRIMARY KEY,
    ->     CourseTitle varchar(20) NOT NULL,
    ->     Credits int NOT NULL
    -> );
Query OK, 0 rows affected (0.12 sec)
```

Output :

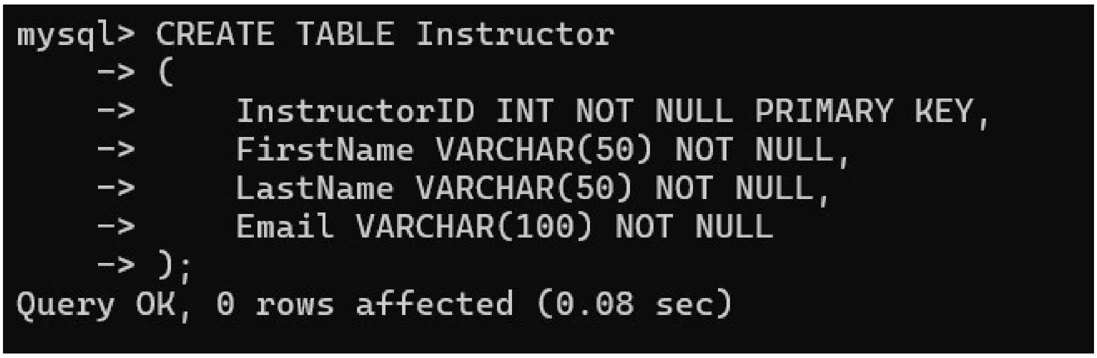


```
mysql> describe Course;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| CourseID    | int         | NO   | PRI | NULL    |       |
| CourseTitle | varchar(20) | NO   |     | NULL    |       |
| Credits     | int         | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.04 sec)
```

Create a table with named Instructor with attributes:

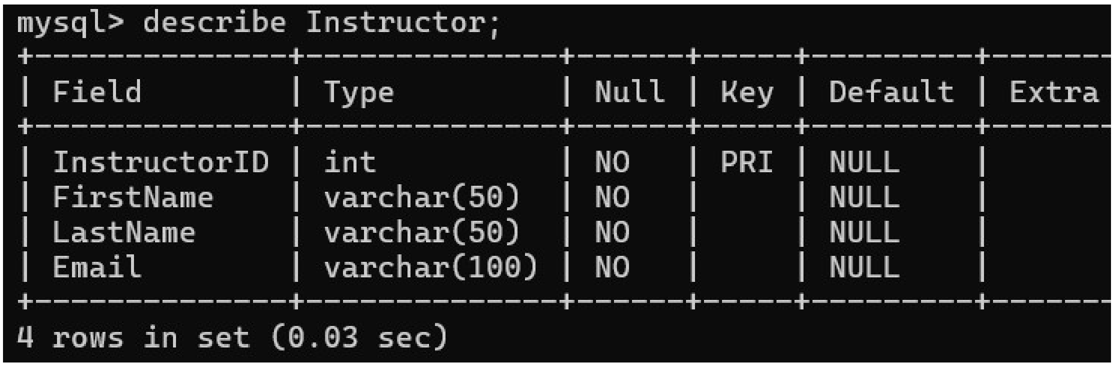
- InstructorID (Primary Key)
- FirstName
- LastName
- Email

Code :



```
mysql> CREATE TABLE Instructor
    -> (
    ->     InstructorID INT NOT NULL PRIMARY KEY,
    ->     FirstName VARCHAR(50) NOT NULL,
    ->     LastName VARCHAR(50) NOT NULL,
    ->     Email VARCHAR(100) NOT NULL
    -> );
Query OK, 0 rows affected (0.08 sec)
```

Output :

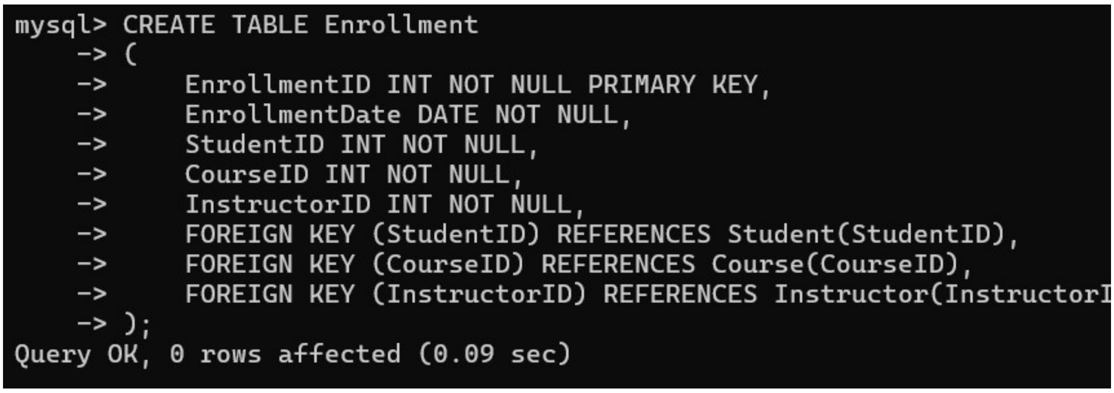


```
mysql> describe Instructor;
+--------------+--------------+------+-----+---------+-------
| Field        | Type         | Null | Key | Default | Extra
+--------------+--------------+------+-----+---------+-------
| InstructorID | int          | NO   | PRI | NULL    |
| FirstName    | varchar(50)  | NO   |     | NULL    |
| LastName     | varchar(50)  | NO   |     | NULL    |
| Email        | varchar(100) | NO   |     | NULL    |
+--------------+--------------+------+-----+---------+-------
4 rows in set (0.03 sec)
```

Create a table with named Enrollment with attributes:

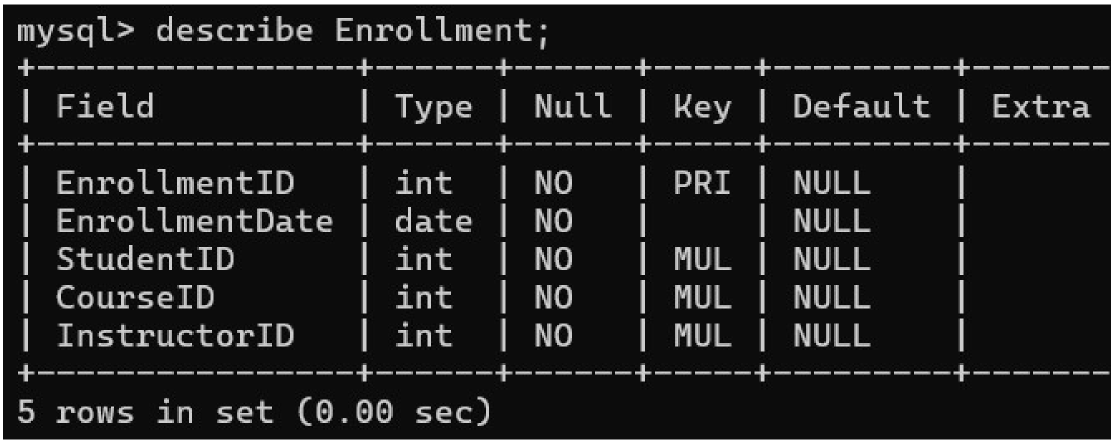
- EnrollmentID (Primary Key)
- EnrollmentDate
- StudentID(Foreign key)
- CourseID(Foreign Key)
- InstructorID(Foreign key)

Code :



```
mysql> CREATE TABLE Enrollment
    -> (
    ->     EnrollmentID INT NOT NULL PRIMARY KEY,
    ->     EnrollmentDate DATE NOT NULL,
    ->     StudentID INT NOT NULL,
    ->     CourseID INT NOT NULL,
    ->     InstructorID INT NOT NULL,
    ->     FOREIGN KEY (StudentID) REFERENCES Student(StudentID),
    ->     FOREIGN KEY (CourseID) REFERENCES Course(CourseID),
    ->     FOREIGN KEY (InstructorID) REFERENCES Instructor(InstructorI
    -> );
Query OK, 0 rows affected (0.09 sec)
```

Output :

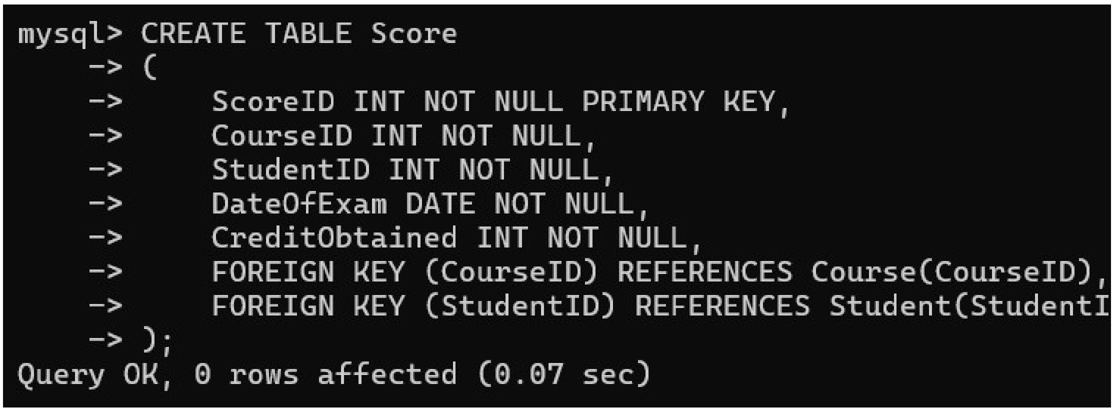


```
mysql> describe Enrollment;
+----------------+------+------+-----+---------+-------
| Field          | Type | Null | Key | Default | Extra
+----------------+------+------+-----+---------+-------
| EnrollmentID   | int  | NO   | PRI | NULL    |
| EnrollmentDate | date | NO   |     | NULL    |
| StudentID      | int  | NO   | MUL | NULL    |
| CourseID       | int  | NO   | MUL | NULL    |
| InstructorID   | int  | NO   | MUL | NULL    |
+----------------+------+------+-----+---------+-------
5 rows in set (0.00 sec)
```

Create a table with named Score with attributes:

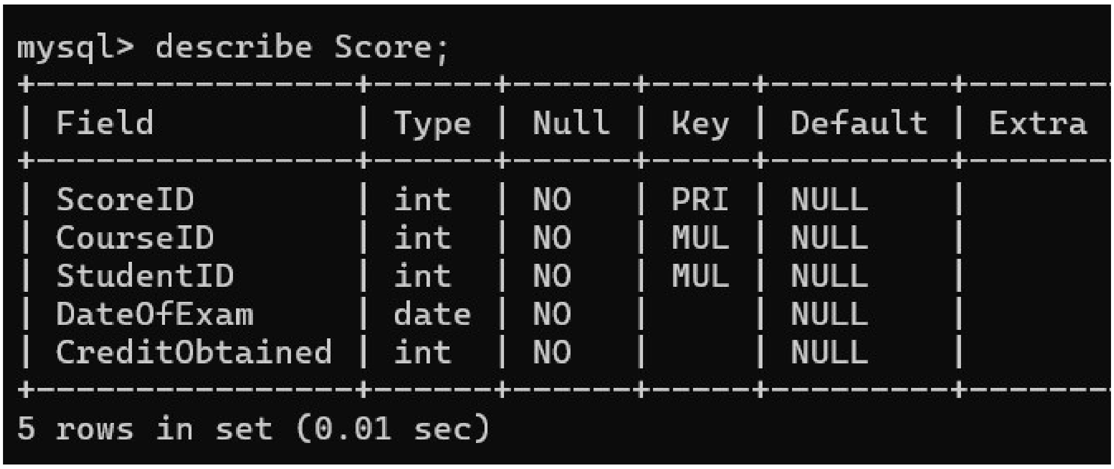
- ScoreID (Primary Key)
- CourseID (Foreign key)
- StudentID (Foreign Key)
- DateOfExam
- CreditObtained

Code :



```
mysql> CREATE TABLE Score
    -> (
    ->     ScoreID INT NOT NULL PRIMARY KEY,
    ->     CourseID INT NOT NULL,
    ->     StudentID INT NOT NULL,
    ->     DateOfExam DATE NOT NULL,
    ->     CreditObtained INT NOT NULL,
    ->     FOREIGN KEY (CourseID) REFERENCES Course(CourseID),
    ->     FOREIGN KEY (StudentID) REFERENCES Student(StudentI
    -> );
Query OK, 0 rows affected (0.07 sec)
```

Output :

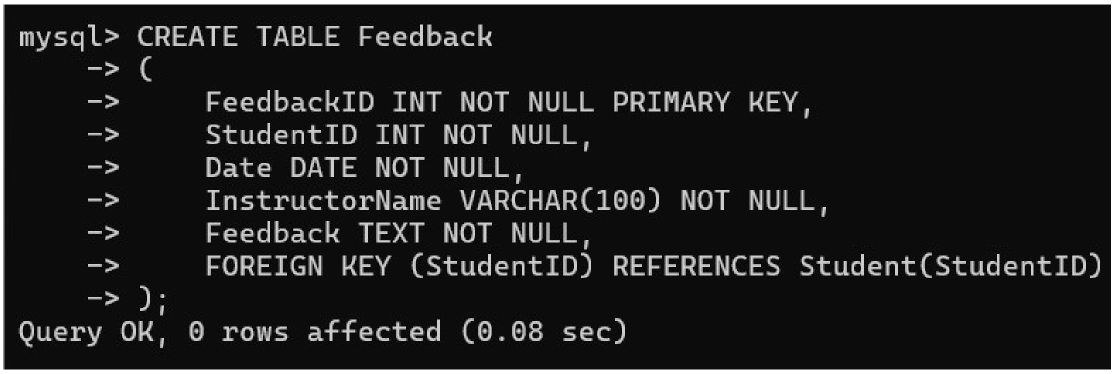


```
mysql> describe Score;
+----------------+------+------+-----+---------+-------
| Field          | Type | Null | Key | Default | Extra
+----------------+------+------+-----+---------+-------
| ScoreID        | int  | NO   | PRI | NULL    |
| CourseID       | int  | NO   | MUL | NULL    |
| StudentID      | int  | NO   | MUL | NULL    |
| DateOfExam     | date | NO   |     | NULL    |
| CreditObtained | int  | NO   |     | NULL    |
+----------------+------+------+-----+---------+-------
5 rows in set (0.01 sec)
```

Create a table with named Feedback with attributes:

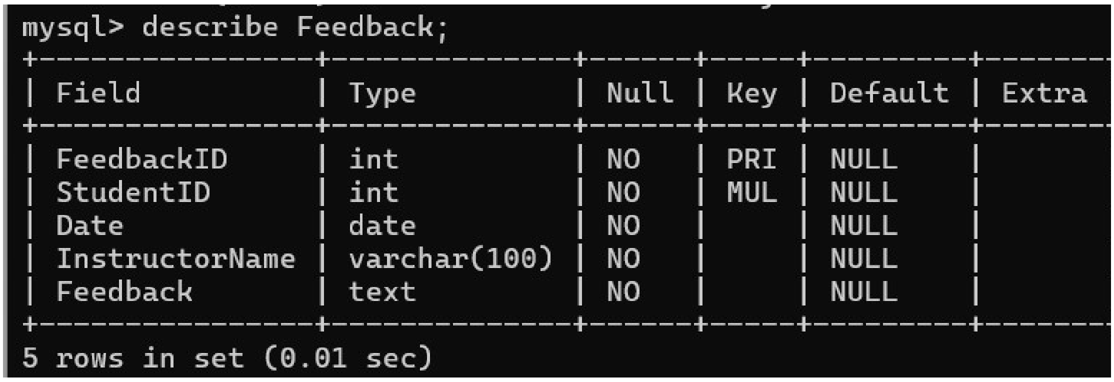
- FeedbackID (Primary Key)
- StudentID (Foreign key)
- Date
- InstructorName
- Feedback

Code :



```
mysql> CREATE TABLE Feedback
    -> (
    ->     FeedbackID INT NOT NULL PRIMARY KEY,
    ->     StudentID INT NOT NULL,
    ->     Date DATE NOT NULL,
    ->     InstructorName VARCHAR(100) NOT NULL,
    ->     Feedback TEXT NOT NULL,
    ->     FOREIGN KEY (StudentID) REFERENCES Student(StudentID)
    -> );
Query OK, 0 rows affected (0.08 sec)
```

Output :



```
mysql> describe Feedback;
+----------------+--------------+------+-----+---------+-------
| Field          | Type         | Null | Key | Default | Extra
+----------------+--------------+------+-----+---------+-------
| FeedbackID     | int          | NO   | PRI | NULL    |
| StudentID      | int          | NO   | MUL | NULL    |
| Date           | date         | NO   |     | NULL    |
| InstructorName | varchar(100) | NO   |     | NULL    |
| Feedback       | text         | NO   |     | NULL    |
+----------------+--------------+------+-----+---------+-------
5 rows in set (0.01 sec)
```

ChatGPT Exercise

Using ChatGPT generate the Database design

Scenario: Implementing Database Design

The database should store emergency contact information for each employee. This information is crucial for situations where immediate contact with family or emergency contacts is necessary. The design should consider privacy and security measures for sensitive contact details.
Use the chatGPT prompt to formulate the database design for the described scenario.

Sure, here is a detailed database design for storing emergency contact information for each employee. The design includes necessary considerations for privacy and security.

Database Design

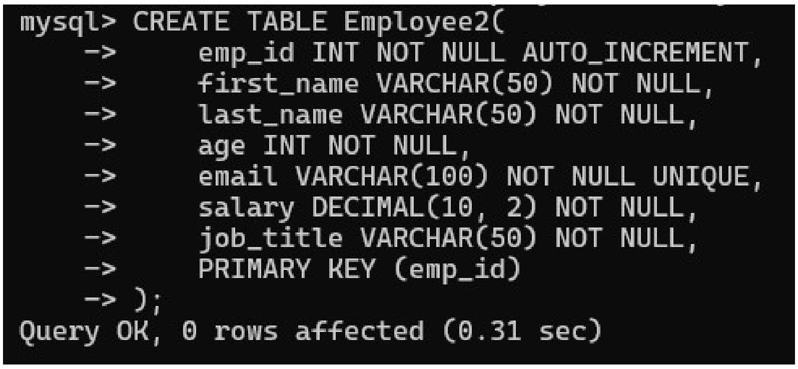
Tables:

Employee

EmergencyContact

Table: Employee

This table will store basic employee information.



```
mysql> CREATE TABLE Employee2(
    ->     emp_id INT NOT NULL AUTO_INCREMENT,
    ->     first_name VARCHAR(50) NOT NULL,
    ->     last_name VARCHAR(50) NOT NULL,
    ->     age INT NOT NULL,
    ->     email VARCHAR(100) NOT NULL UNIQUE,
    ->     salary DECIMAL(10, 2) NOT NULL,
    ->     job_title VARCHAR(50) NOT NULL,
    ->     PRIMARY KEY (emp_id)
    -> );
Query OK, 0 rows affected (0.31 sec)
```

Table: EmergencyContact

This table will store emergency contact information for each employee.

Privacy and Security Considerations

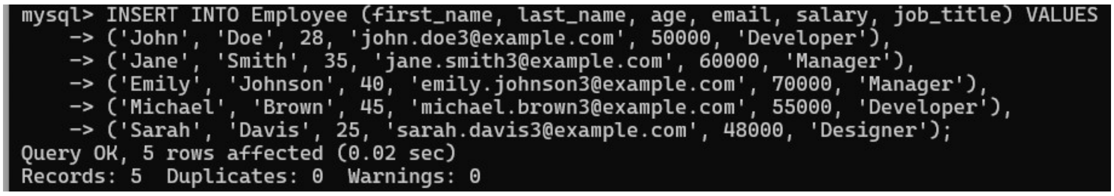
Access Control: Ensure that only authorized personnel can access or modify emergency contact information.

Encryption: Use encryption to store sensitive information like phone numbers and addresses.

Audit Logs: Maintain logs of any changes to the emergency contact information for auditing purposes.

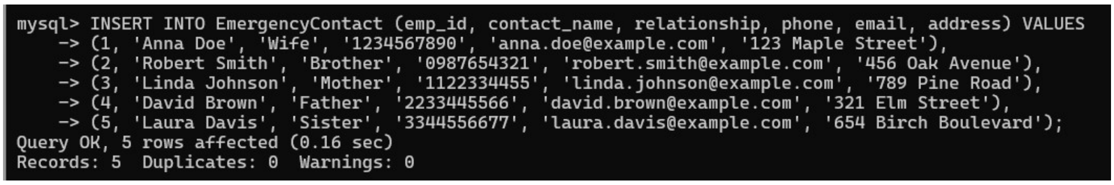
Sample Data Insertion

Insert Sample Employee Data



```
mysql> INSERT INTO Employee (first_name, last_name, age, email, salary, job_title) VALUES
    -> ('John', 'Doe', 28, 'john.doe3@example.com', 50000, 'Developer'),
    -> ('Jane', 'Smith', 35, 'jane.smith3@example.com', 60000, 'Manager'),
    -> ('Emily', 'Johnson', 40, 'emily.johnson3@example.com', 70000, 'Manager'),
    -> ('Michael', 'Brown', 45, 'michael.brown3@example.com', 55000, 'Developer'),
    -> ('Sarah', 'Davis', 25, 'sarah.davis3@example.com', 48000, 'Designer');
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Insert Sample Emergency Contact Data

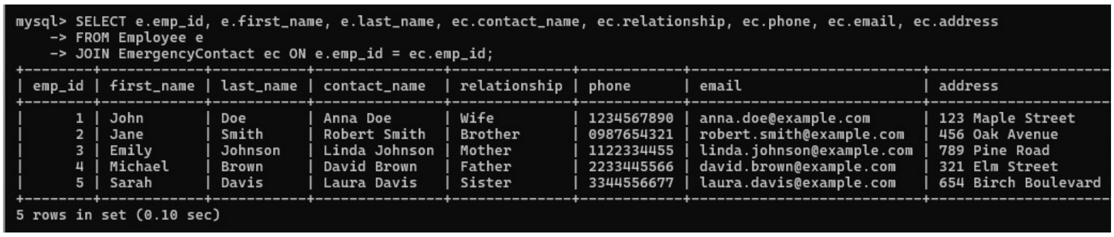


```
mysql> INSERT INTO EmergencyContact (emp_id, contact_name, relationship, phone, email, address) VALUES
    -> (1, 'Anna Doe', 'Wife', '1234567890', 'anna.doe@example.com', '123 Maple Street'),
    -> (2, 'Robert Smith', 'Brother', '0987654321', 'robert.smith@example.com', '456 Oak Avenue'),
    -> (3, 'Linda Johnson', 'Mother', '1122334455', 'linda.johnson@example.com', '789 Pine Road'),
    -> (4, 'David Brown', 'Father', '2233445566', 'david.brown@example.com', '321 Elm Street'),
    -> (5, 'Laura Davis', 'Sister', '3344556677', 'laura.davis@example.com', '654 Birch Boulevard');
Query OK, 5 rows affected (0.16 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Queries

Retrieve Employee and Their Emergency Contacts

To retrieve an employee and their emergency contacts, you can use a JOIN query:



```
mysql> SELECT e.emp_id, e.first_name, e.last_name, ec.contact_name, ec.relationship, ec.phone, ec.email, ec.address
    -> FROM Employee e
    -> JOIN EmergencyContact ec ON e.emp_id = ec.emp_id;
+--------+------------+-----------+---------------+--------------+------------+---------------------------+--------------------
| emp_id | first_name | last_name | contact_name  | relationship | phone      | email                     | address
+--------+------------+-----------+---------------+--------------+------------+---------------------------+--------------------
|      1 | John       | Doe       | Anna Doe      | Wife         | 1234567890 | anna.doe@example.com      | 123 Maple Street
|      2 | Jane       | Smith     | Robert Smith  | Brother      | 0987654321 | robert.smith@example.com  | 456 Oak Avenue
|      3 | Emily      | Johnson   | Linda Johnson | Mother       | 1122334455 | linda.johnson@example.com | 789 Pine Road
|      4 | Michael    | Brown     | David Brown   | Father       | 2233445566 | david.brown@example.com   | 321 Elm Street
|      5 | Sarah      | Davis     | Laura Davis   | Sister       | 3344556677 | laura.davis@example.com   | 654 Birch Boulevard
+--------+------------+-----------+---------------+--------------+------------+---------------------------+--------------------
5 rows in set (0.10 sec)
```

Additional Considerations

Indexes: Add indexes on columns that are frequently searched, such as emp_id in the EmergencyContact table.

Backups: Regularly backup the database to prevent data loss.

Data Validation: Implement data validation checks to ensure the integrity of the data.

This design ensures that emergency contact information is properly associated with each employee and considers necessary privacy and security measures.