Assignment 2

BY SHREYASI REJA

Student Information System (SIS)

Task 1. Database Design:

1. Create the database named "SISDB”
   QUERY:- CREATE DATABASE SISDB;

```
mysql> CREATE DATABASE SISDB;
Query OK, 1 row affected (0.26 sec)

mysql> USE SISDB;
Database changed
mysql>
```

2.Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

a. Students b. Courses c. Enrollments d. Teacher e. Payments

QUERY:- a. Students

CREATE TABLE Students (

-> StudentID INT AUTO_INCREMENT PRIMARY KEY,

-> FirstName VARCHAR(50),

-> LastName VARCHAR(50),

-> DateofBirth DATE,

-> Email VARCHAR(100),

-> PhoneNumber VARCHAR(100)

->);

```
mysql> CREATE TABLE Students (
    -> StudentID INT AUTO_INCREMENT PRIMARY KEY,
    -> FirstName VARCHAR(50),
    -> LastName VARCHAR(50),
    -> DateofBirth DATE,
    -> Email VARCHAR(100),
    -> PhoneNumber VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.07 sec)

mysql> DESC Students;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| StudentID   | int         | NO   | PRI | NULL    | auto_increment |
| FirstName   | varchar(50) | YES  |     | NULL    |                |
| LastName    | varchar(50) | YES  |     | NULL    |                |
| DateofBirth | date        | YES  |     | NULL    |                |
| Email       | varchar(100)| YES  |     | NULL    |                |
| PhoneNumber | varchar(100)| YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)

mysql>
```

b.Courses:-

CREATE TABLE Courses (

-> CourseID INT AUTO_INCREMENT PRIMARY KEY,

-> CourseName VARCHAR(100),

-> Credits INT,

-> TeacherID INT,

-> FOREIGN KEY (TeacherID) REFERENCES Teacher(TeacherID)

->);

```
mysql> CREATE TABLE Courses (
    -> CourseID INT AUTO_INCREMENT PRIMARY KEY,
    -> CourseName VARCHAR(100),
    -> Credits INT,
    -> TeacherID INT,
    -> FOREIGN KEY (TeacherID) REFERENCES Teacher(TeacherID)
    -> );
Query OK, 0 rows affected (0.14 sec)

mysql> desc Courses;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| CourseID   | int          | NO   | PRI | NULL    | auto_increment |
| CourseName | varchar(100) | YES  |     | NULL    |                |
| Credits    | int          | YES  |     | NULL    |                |
| TeacherID  | int          | YES  | MUL | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql>
```

c.Enrollments:-

```
CREATE TABLE Enrollments (
   ->  EnrollmentID INT AUTO_INCREMENT PRIMARY KEY,
   -> StudentID INT,
   -> CourseID INT,
   -> EnrollmentDate DATE,
   -> FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
   -> FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
   -> );
```

```
mysql> CREATE TABLE Enrollments (
    -> EnrollmentID INT AUTO_INCREMENT PRIMARY KEY,
    -> StudentID INT,
    -> CourseID INT,
    -> EnrollmentDate DATE,
    -> FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    -> FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
    -> );
Query OK, 0 rows affected (0.11 sec)

mysql> DESC Enrollments;
+----------------+------+------+-----+---------+----------------+
| Field          | Type | Null | Key | Default | Extra          |
+----------------+------+------+-----+---------+----------------+
| EnrollmentID   | int  | NO   | PRI | NULL    | auto_increment |
| StudentID      | int  | YES  | MUL | NULL    |                |
| CourseID       | int  | YES  | MUL | NULL    |                |
| EnrollmentDate | date | YES  |     | NULL    |                |
+----------------+------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql>
```

d.Teacher:-

```
CREATE TABLE Teacher (
   -> TeacherID INT AUTO_INCREMENT  PRIMARY KEY,
   -> FirstName VARCHAR(50),
   -> LastName VARCHAR(50),
   -> Email VARCHAR(100)
 -> );
```

```
mysql> CREATE TABLE Teacher (
    -> TeacherID INT  AUTO_INCREMENT PRIMARY KEY,
    -> FirstName VARCHAR(50),
    -> LastName VARCHAR(50),
    -> Email VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.07 sec)

mysql> desc Teacher;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| TeacherID  | int          | NO   | PRI | NULL    | auto_increment |
| FirstName  | varchar(50)  | YES  |     | NULL    |                |
| LastName   | varchar(50)  | YES  |     | NULL    |                |
| Email      | varchar(100) | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql>
```

e.Payments:-

CREATE TABLE Payments (

-> PaymentID INT AUTO_INCREMENT PRIMARY KEY,

-> StudentID INT,

-> Amount DECIMAL(10, 2),

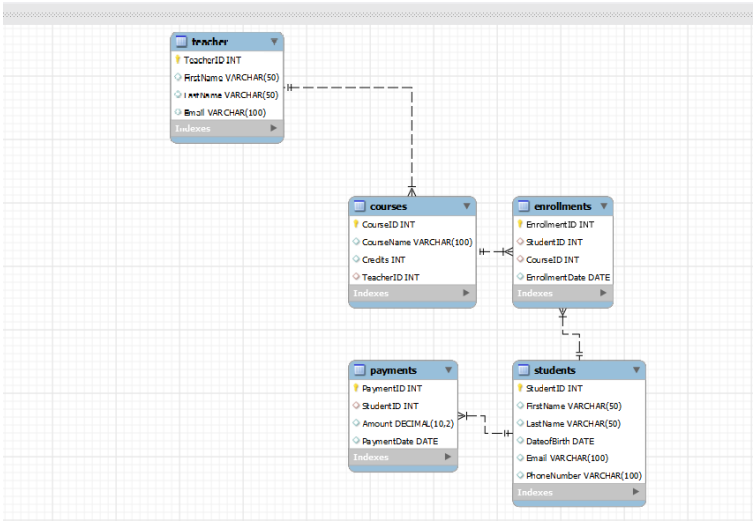
-> PaymentDate DATE,

-> FOREIGN KEY (StudentID) REFERENCES Students(StudentID)

->);

```
mysql> CREATE TABLE Payments (
    -> PaymentID INT  AUTO_INCREMENT PRIMARY KEY,
    -> StudentID INT,
    -> Amount DECIMAL(10, 2),
    -> PaymentDate DATE,
    -> FOREIGN KEY (StudentID) REFERENCES Students(StudentID)
    -> );
Query OK, 0 rows affected (0.10 sec)

mysql> DESC Payments;
+-------------+---------------+------+-----+---------+----------------+
| Field       | Type          | Null | Key | Default | Extra          |
+-------------+---------------+------+-----+---------+----------------+
| PaymentID   | int           | NO   | PRI | NULL    | auto_increment |
| StudentID   | int           | YES  | MUL | NULL    |                |
| Amount      | decimal(10,2) | YES  |     | NULL    |                |
| PaymentDate | date          | YES  |     | NULL    |                |
+-------------+---------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql>
```

3.Create an ERD (Entity Relationship Diagram) for the database.



5. Insert at least 10 sample records into each of the following tables.

i. Students ii. Courses iii. Enrollments iv. Teacher v. Payments

QUERY:-

i.Students:-

INSERT INTO Students VALUES

(1, 'Raj', 'Roy', '1999-01-15', 'rajroy2023@email.com', '9786457445'),

(2, 'Rohit', 'Ray', '1999-01-16', 'rohitray2023@email.com', '9786457446'),

(3, 'Rahul', 'Roy', '1999-02-17', 'rahulroy2023@email.com', '9786457447'),

(4, 'Akash', 'Dey', '1999-02-18', 'akashdey2023@email.com', '9786457448'),

(5, 'Amit', 'Mishra', '1999-03-19', 'amitmishra2023@email.com', '9786457449'),

(6, 'Arti', 'Mishra', '1999-04-20', 'artimishra2023@email.com', '9786457441'),

(7, 'Piu', 'Das', '1999-04-09', 'piudas2023@email.com', '9786457442'),

(8, 'Shreya', 'Sen', '1999-04-13', 'shreyassen2023@email.com', '9786457443'),

(9, 'Anik', 'Singh', '1999-06-11', 'aniksingh2023@email.com', '9786457444'),

(10, 'Ayesha', 'Singh', '1999-09-12', 'ayeshasingh2023@email.com', '9786457440');

```
mysql> INSERT INTO Students VALUES
    -> (1, 'Raj', 'Roy', '1999-01-15', 'rajroy2023@email.com', '9786457445'),
    -> (2, 'Rohit', 'Ray', '1999-01-16', 'rohitray2023@email.com', '9786457446'),
    -> (3, 'Rahul', 'Roy', '1999-02-17', 'rahulroy2023@email.com', '9786457447'),
    -> (4, 'Akash', 'Dey', '1999-02-18', 'akashdey2023@email.com', '9786457448'),
    -> (5, 'Amit', 'Mishra', '1999-03-19', 'amitmishra2023@email.com', '9786457449'),
    -> (6, 'Arti', 'Mishra', '1999-04-20', 'artimishra2023@email.com', '9786457441'),
    -> (7, 'Piu', 'Das', '1999-04-09', 'piudas2023@email.com', '9786457442'),
    -> (8, 'Shreya', 'Sen', '1999-04-13', 'shreyassen2023@email.com', '9786457443'),
    -> (9, 'Anik', 'Singh', '1999-06-11', 'aniksingh2023@email.com', '9786457444'),
    -> (10, 'Ayesha', 'Singh', '1999-09-12', 'ayeshasingh2023@email.com', '9786457440');
Query OK, 10 rows affected (0.03 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT *FROM Students;
+-----------+-----------+----------+-------------+--------------------------+-------------+
| StudentID | FirstName | LastName | DateofBirth | Email                    | PhoneNumber |
+-----------+-----------+----------+-------------+--------------------------+-------------+
|         1 | Raj       | Roy      | 1999-01-15  | rajroy2023@email.com     | 9786457445  |
|         2 | Rohit     | Ray      | 1999-01-16  | rohitray2023@email.com   | 9786457446  |
|         3 | Rahul     | Roy      | 1999-02-17  | rahulroy2023@email.com   | 9786457447  |
|         4 | Akash     | Dey      | 1999-02-18  | akashdey2023@email.com   | 9786457448  |
|         5 | Amit      | Mishra   | 1999-03-19  | amitmishra2023@email.com | 9786457449  |
|         6 | Arti      | Mishra   | 1999-04-20  | artimishra2023@email.com | 9786457441  |
|         7 | Piu       | Das      | 1999-04-09  | piudas2023@email.com     | 9786457442  |
|         8 | Shreya    | Sen      | 1999-04-13  | shreyassen2023@email.com | 9786457443  |
|         9 | Anik      | Singh    | 1999-06-11  | aniksingh2023@email.com  | 9786457444  |
|        10 | Ayesha    | Singh    | 1999-09-12  | ayeshasingh2023@email.com | 9786457440 |
+-----------+-----------+----------+-------------+--------------------------+-------------+
10 rows in set (0.00 sec)

mysql>
```

ii. Courses:-

INSERT INTO Courses VALUES

-> (101, 'Mathematics', 3, 1),

-> (102, 'History', 4, 2),

-> (103, 'Computer Science', 5, 3),

-> (104, 'English Literature', 3, 5),

-> (105, 'Physics', 4, 4),

-> (106, 'Chemistry', 4, 6),

-> (107, 'Biology', 3, 7),

-> (108, 'Art', 2, 2),

-> (109, 'Music', 3, 5),

-> (110, 'Economics', 4, 1);

```
mysql> INSERT INTO Courses VALUES
    -> (101, 'Mathematics', 3, 1),
    -> (102, 'History', 4, 2),
    ->  (103, 'Computer Science', 5, 3),
    -> (104, 'English Literature', 3, 5),
    -> (105, 'Physics', 4, 4),
    ->  (106, 'Chemistry', 4, 6),
    ->  (107, 'Biology', 3, 7),
    -> (108, 'Art', 2, 2),
    -> (109, 'Music', 3, 5),
    -> (110, 'Economics', 4, 1);
Query OK, 10 rows affected (0.03 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Courses;
+----------+--------------------+---------+-----------+
| CourseID | CourseName         | Credits | TeacherID |
+----------+--------------------+---------+-----------+
|      101 | Mathematics        |       3 |         1 |
|      102 | History            |       4 |         2 |
|      103 | Computer Science   |       5 |         3 |
|      104 | English Literature |       3 |         5 |
|      105 | Physics            |       4 |         4 |
|      106 | Chemistry          |       4 |         6 |
|      107 | Biology            |       3 |         7 |
|      108 | Art                |       2 |         2 |
|      109 | Music              |       3 |         5 |
|      110 | Economics          |       4 |         1 |
+----------+--------------------+---------+-----------+
10 rows in set (0.00 sec)

mysql>
```

iii.Enrollments:-

INSERT INTO Enrollments VALUES

(1, 1, 101, '2024-01-15'),

(2, 2, 102, '2024-01-16'),

(3, 3, 103, '2024-01-17'),

(4, 4, 104, '2024-01-18'),

(5, 5, 105, '2024-01-19'),

(6, 6, 106, '2024-01-20'),

(7, 7, 107, '2024-01-21'),

(8, 8, 108, '2024-01-22'),

(9, 9, 109, '2024-01-23'),

(10, 10, 110, '2024-01-24');

```
mysql> INSERT INTO Enrollments VALUES
    ->  (1, 1, 101, '2024-01-15'),
    ->     (2, 2, 102, '2024-01-16'),
    ->     (3, 3, 103, '2024-01-17'),
    ->     (4, 4, 104, '2024-01-18'),
    ->     (5, 5, 105, '2024-01-19'),
    ->     (6, 6, 106, '2024-01-20'),
    ->     (7, 7, 107, '2024-01-21'),
    ->     (8, 8, 108, '2024-01-22'),
    ->     (9, 9, 109, '2024-01-23'),
    ->     (10, 10, 110, '2024-01-24');
Query OK, 10 rows affected (0.05 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Enrollments;
+--------------+-----------+----------+----------------+
| EnrollmentID | StudentID | CourseID | EnrollmentDate |
+--------------+-----------+----------+----------------+
|            1 |         1 |      101 | 2024-01-15     |
|            2 |         2 |      102 | 2024-01-16     |
|            3 |         3 |      103 | 2024-01-17     |
|            4 |         4 |      104 | 2024-01-18     |
|            5 |         5 |      105 | 2024-01-19     |
|            6 |         6 |      106 | 2024-01-20     |
|            7 |         7 |      107 | 2024-01-21     |
|            8 |         8 |      108 | 2024-01-22     |
|            9 |         9 |      109 | 2024-01-23     |
|           10 |        10 |      110 | 2024-01-24     |
+--------------+-----------+----------+----------------+
10 rows in set (0.00 sec)

mysql>
```

iv. Teacher:-

INSERT INTO Teacher VALUES

-> (1, 'Aditya', 'Roy', 'aditya.roy@email.com'),

-> (2, 'Aman', 'Sen', 'aman.sen@email.com'),

-> (3, 'Rajesh', 'Ray', 'rajesh.ray@email.com'),

-> (4, 'Raman', 'Das', 'raman.das@email.com'),

-> (5, 'Priya', 'Meheta', 'priya.meheta@email.com'),

-> (6, 'Shriya', 'Iyer', 'shriya.iyer@email.com'),

-> (7, 'Daya', 'Sen', 'daya.sen@email.com'),

-> (8, 'Surya', 'Sen', 'surya.sen@email.com'),

-> (9, 'Anik', 'Dey', 'anik.dey@email.com'),

-> (10, 'Poulami', 'Pal', 'poulami.pal@email.com');

```
mysql> INSERT INTO Teacher VALUES
    -> (1, 'Aditya', 'Roy', 'aditya.roy@email.com'),
    -> (2, 'Aman', 'Sen', 'aman.sen@email.com'),
    -> (3, 'Rajesh', 'Ray', 'rajesh.ray@email.com'),
    -> (4, 'Raman', 'Das', 'raman.das@email.com'),
    -> (5, 'Priya', 'Meheta', 'priya.meheta@email.com'),
    -> (6, 'Shriya', 'Iyer', 'shriya.iyer@email.com'),
    -> (7, 'Daya', 'Sen', 'daya.sen@email.com'),
    -> (8, 'Surya', 'Sen', 'surya.sen@email.com'),
    -> (9, 'Anik', 'Dey', 'anik.dey@email.com'),
    -> (10, 'Poulami', 'Pal', 'poulami.pal@email.com');
Query OK, 10 rows affected (0.07 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Teacher;
+-----------+-----------+----------+------------------------+
| TeacherID | FirstName | LastName | Email                  |
+-----------+-----------+----------+------------------------+
|         1 | Aditya    | Roy      | aditya.roy@email.com   |
|         2 | Aman      | Sen      | aman.sen@email.com     |
|         3 | Rajesh    | Ray      | rajesh.ray@email.com   |
|         4 | Raman     | Das      | raman.das@email.com    |
|         5 | Priya     | Meheta   | priya.meheta@email.com |
|         6 | Shriya    | Iyer     | shriya.iyer@email.com  |
|         7 | Daya      | Sen      | daya.sen@email.com     |
|         8 | Surya     | Sen      | surya.sen@email.com    |
|         9 | Anik      | Dey      | anik.dey@email.com     |
|        10 | Poulami   | Pal      | poulami.pal@email.com  |
+-----------+-----------+----------+------------------------+
10 rows in set (0.00 sec)

mysql>
```

V. Payments:-

INSERT INTO Payments VALUES--(The Payment Amount is in Dollar)

(1, 1, 100.50, '2024-01-15'),

(2, 2, 75.20, '2024-01-16'),

(3, 3, 120.75, '2024-01-17'),

(4, 4, 90.00, '2024-01-18'),

(5, 5, 110.80, '2024-01-19'),

(6, 6, 200.25, '2024-01-20'),

(7, 7, 85.30, '2024-01-21'),

(8, 8, 150.60, '2024-01-22'),

(9, 9, 80.45, '2024-01-23'),

(10, 10, 95.75, '2024-01-24');

```
mysql> INSERT INTO Payments VALUES
    -> (1, 1, 100.50, '2024-01-15'),
    ->     (2, 2, 75.20, '2024-01-16'),
    ->     (3, 3, 120.75, '2024-01-17'),
    ->     (4, 4, 90.00, '2024-01-18'),
    ->     (5, 5, 110.80, '2024-01-19'),
    ->     (6, 6, 200.25, '2024-01-20'),
    ->     (7, 7, 85.30, '2024-01-21'),
    ->     (8, 8, 150.60, '2024-01-22'),
    ->     (9, 9, 80.45, '2024-01-23'),
    ->     (10, 10, 95.75, '2024-01-24');
Query OK, 10 rows affected (0.07 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SLECT * FROM Payments;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the righ
t syntax to use near 'SLECT * FROM Payments' at line 1
mysql> SELECT * FROM Payments;
+-----------+-----------+--------+-------------+
| PaymentID | StudentID | Amount | PaymentDate |
+-----------+-----------+--------+-------------+
|         1 |         1 | 100.50 | 2024-01-15  |
|         2 |         2 |  75.20 | 2024-01-16  |
|         3 |         3 | 120.75 | 2024-01-17  |
|         4 |         4 |  90.00 | 2024-01-18  |
|         5 |         5 | 110.80 | 2024-01-19  |
|         6 |         6 | 200.25 | 2024-01-20  |
|         7 |         7 |  85.30 | 2024-01-21  |
|         8 |         8 | 150.60 | 2024-01-22  |
|         9 |         9 |  80.45 | 2024-01-23  |
|        10 |        10 |  95.75 | 2024-01-24  |
+-----------+-----------+--------+-------------+
```

Tasks 2: Select, Where, Between, AND, LIKE:-

1. Write an SQL query to insert a new student into the "Students" table with the following details:

a. First Name: John

b. Last Name: Doe

c. Date of Birth: 1995-08-15

d. Email: john.doe@example.com

e. Phone Number: 1234567890

QUERY:-

INSERT INTO Students (FirstName, LastName, DateofBirth, Email, PhoneNumber)

-> VALUES ('John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');

```
mysql> INSERT INTO Students (FirstName, LastName, DateofBirth, Email, PhoneNumber)
    -> VALUES ('John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
Query OK, 1 row affected (0.06 sec)

mysql> select * from Students;
+-----------+-----------+----------+-------------+--------------------------+-------------+
| StudentID | FirstName | LastName | DateofBirth | Email                    | PhoneNumber |
+-----------+-----------+----------+-------------+--------------------------+-------------+
|         1 | Raj       | Roy      | 1999-01-15  | rajroy2023@email.com     | 9786457445  |
|         2 | Rohit     | Ray      | 1999-01-16  | rohitray2023@email.com   | 9786457446  |
|         3 | Rahul     | Roy      | 1999-02-17  | rahulroy2023@email.com   | 9786457447  |
|         4 | Akash     | Dey      | 1999-02-18  | akashdey2023@email.com   | 9786457448  |
|         5 | Amit      | Mishra   | 1999-03-19  | amitmishra2023@email.com | 9786457449  |
|         6 | Arti      | Mishra   | 1999-04-20  | artimishra2023@email.com | 9786457441  |
|         7 | Piu       | Das      | 1999-04-09  | piudas2023@email.com     | 9786457442  |
|         8 | Shreya    | Sen      | 1999-04-13  | shreyassen2023@email.com | 9786457443  |
|         9 | Anik      | Singh    | 1999-06-11  | aniksingh2023@email.com  | 9786457444  |
|        10 | Ayesha    | Singh    | 1999-09-12  | ayeshasingh2023@email.com | 9786457440 |
|        11 | John      | Doe      | 1995-08-15  | john.doe@example.com     | 1234567890  |
+-----------+-----------+----------+-------------+--------------------------+-------------+
11 rows in set (0.00 sec)

mysql>
```

2. Write an SQL query to enroll a student in a course. Choose an existing student and course and insert a record into the "Enrollments" table with the enrollment date

   QUERY:-

   INSERT INTO Enrollments (StudentID, CourseID, EnrollmentDate)

   -> VALUES (

   -> (SELECT StudentID FROM Students WHERE FirstName = 'John' AND LastName = 'Doe'),

   -> (SELECT CourseID FROM Courses WHERE CourseName = 'Mathematics'),

   -> '2024-01-18'

   ->);

```
mysql> INSERT INTO Enrollments (StudentID, CourseID, EnrollmentDate)
    -> VALUES (
    ->     (SELECT StudentID FROM Students WHERE FirstName = 'John' AND LastName = 'Doe'),
    ->     (SELECT CourseID FROM Courses WHERE CourseName = 'Mathematics'),
    -> '2024-01-18'
    -> );
Query OK, 1 row affected (0.06 sec)

mysql> SELECT * FROM Enrollments;
+--------------+-----------+----------+----------------+
| EnrollmentID | StudentID | CourseID | EnrollmentDate |
+--------------+-----------+----------+----------------+
|            1 |         1 |      101 | 2024-01-15     |
|            2 |         2 |      102 | 2024-01-16     |
|            3 |         3 |      103 | 2024-01-17     |
|            4 |         4 |      104 | 2024-01-18     |
|            5 |         5 |      105 | 2024-01-19     |
|            6 |         6 |      106 | 2024-01-20     |
|            7 |         7 |      107 | 2024-01-21     |
|            8 |         8 |      108 | 2024-01-22     |
|            9 |         9 |      109 | 2024-01-23     |
|           10 |        10 |      110 | 2024-01-24     |
|           11 |        11 |      101 | 2024-01-18     |
+--------------+-----------+----------+----------------+
11 rows in set (0.00 sec)

mysql>
```

3.Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.

QUERY:-

UPDATE Teacher
-> SET email = 'adityaroy2024@email.com'
-> WHERE FirstName = 'Aditya' AND LastName = 'Roy';

```
mysql> UPDATE Teacher
    -> SET email = 'adityaroy2024@email.com'
    -> WHERE FirstName = 'Aditya' AND LastName = 'Roy';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Teacher;
+-----------+-----------+----------+-------------------------+
| TeacherID | FirstName | LastName | Email                   |
+-----------+-----------+----------+-------------------------+
|         1 | Aditya    | Roy      | adityaroy2024@email.com |
|         2 | Aman      | Sen      | aman.sen@email.com      |
|         3 | Rajesh    | Ray      | rajesh.ray@email.com    |
|         4 | Raman     | Das      | raman.das@email.com     |
|         5 | Priya     | Meheta   | priya.meheta@email.com  |
|         6 | Shriya    | Iyer     | shriya.iyer@email.com   |
|         7 | Daya      | Sen      | daya.sen@email.com      |
|         8 | Surya     | Sen      | surya.sen@email.com     |
|         9 | Anik      | Dey      | anik.dey@email.com      |
|        10 | Poulami   | Pal      | poulami.pal@email.com   |
+-----------+-----------+----------+-------------------------+
10 rows in set (0.00 sec)

mysql>
```

4.Write an SQL query to delete a specific enrollment record from the "Enrollments" table. Select an enrollment record based on the student and course.

QUERY:-
DELETE FROM Enrollments
WHERE StudentID = (SELECT StudentID FROM Students WHERE FirstName = 'Raj' AND LastName = 'Roy')
AND CourseID = (SELECT CourseID FROM Courses WHERE courseName = 'Mathematics');

```
mysql> /*
   /*> DELETE FROM Enrollments
   /*> WHERE StudentID = (SELECT StudentID FROM Students WHERE FirstName = 'Raj' AND LastName = 'Roy')
   /*> AND CourseID = (SELECT CourseID FROM Courses WHERE courseName = 'Mathematics');
   /*> */
mysql>
```

5.Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables

QUERY:-
UPDATE Courses
-> SET TeacherID = (SELECT TeacherID FROM Teacher WHERE FirstName = 'Aditya' AND LastName = 'Roy')
-> WHERE CourseName = 'Computer Science';

```
mysql> UPDATE Courses
    -> SET TeacherID = (SELECT TeacherID FROM Teacher WHERE FirstName = 'Aditya' AND LastName = 'Roy')
    -> WHERE CourseName = 'Computer Science';
Query OK, 1 row affected (0.11 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM Courses;
+----------+--------------------+---------+-----------+
| CourseID | CourseName         | Credits | TeacherID |
+----------+--------------------+---------+-----------+
|      101 | Mathematics        |       3 |         1 |
|      102 | History            |       4 |         2 |
|      103 | Computer Science   |       5 |         1 |
|      104 | English Literature |       3 |         5 |
|      105 | Physics            |       4 |         4 |
|      106 | Chemistry          |       4 |         6 |
|      107 | Biology            |       3 |         7 |
|      108 | Art                |       2 |         2 |
|      109 | Music              |       3 |         5 |
|      110 | Economics          |       4 |         1 |
+----------+--------------------+---------+-----------+
10 rows in set (0.00 sec)

mysql>
```

6. Delete a specific student from the "Students" table and remove all their enrollment records from the "Enrollments" table. Be sure to maintain referential integrity.

QUERY:-

```
DELETE FROM Enrollments
  WHERE StudentID = (SELECT StudentID FROM Students
WHERE FirstName = 'John' AND LastName = 'Doe');
   DELETE FROM Students
   WHERE FirstName = 'John' AND LastName = 'Doe';
```

```
mysql> /*
   /*> DELETE FROM Enrollments
   /*> WHERE StudentID = (SELECT StudentID FROM Students WHERE FirstName = 'John' AND LastName = 'Doe');
   /*> DELETE FROM Students
   /*> WHERE FirstName = 'John' AND LastName = 'Doe';
   /*> */
mysql>
```

7. Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount

QUERY:-

```
UPDATE Payments
   -> SET Amount = 180
   -> WHERE PaymentID = 1;
```

```
mysql> UPDATE Payments
    -> SET Amount = 180
    -> WHERE PaymentID = 1;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM Payment;
ERROR 1146 (42S02): Table 'sisdb.payment' doesn't exist
mysql> SELECT * FROM Payments;
+-----------+-----------+--------+-------------+
| PaymentID | StudentID | Amount | PaymentDate |
+-----------+-----------+--------+-------------+
|         1 |         1 | 180.00 | 2024-01-15  |
|         2 |         2 |  75.20 | 2024-01-16  |
|         3 |         3 | 120.75 | 2024-01-17  |
|         4 |         4 |  90.00 | 2024-01-18  |
|         5 |         5 | 110.80 | 2024-01-19  |
|         6 |         6 | 200.25 | 2024-01-20  |
|         7 |         7 |  85.30 | 2024-01-21  |
|         8 |         8 | 150.60 | 2024-01-22  |
|         9 |         9 |  80.45 | 2024-01-23  |
|        10 |        10 |  95.75 | 2024-01-24  |
+-----------+-----------+--------+-------------+
10 rows in set (0.00 sec)

mysql>
```

Task 3

Aggregate functions, Having, Order By, GroupBy and Joins:

1. Write an SQL query to calculate the total payments made by a specific student. You will need to join the "Payments" table with the "Students" table based on the student's ID.

   QUERY:-

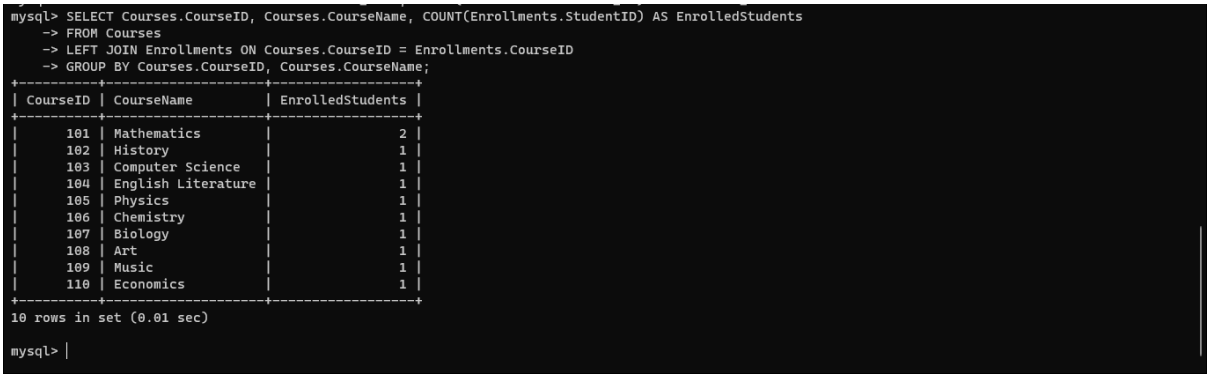
   SELECT Students.StudentID, Students.FirstName, SUM(Payments.Amount) AS TotalPayments
   -> FROM Students
   -> JOIN Payments ON Students.StudentID = Payments.StudentID
   -> WHERE Students.StudentID = 6
   -> GROUP BY Students.StudentID, Students.FirstName;

```
mysql> SELECT Students.StudentID, Students.FirstName, SUM(Payments.Amount) AS TotalPayments
    -> FROM Students
    -> JOIN Payments ON Students.StudentID = Payments.StudentID
    -> WHERE Students.StudentID = 6
    -> GROUP BY Students.StudentID, Students.FirstName;
+-----------+-----------+---------------+
| StudentID | FirstName | TotalPayments |
+-----------+-----------+---------------+
|         6 | Arti      |        200.25 |
+-----------+-----------+---------------+
1 row in set (0.01 sec)

mysql>
```

2. Write an SQL query to retrieve a list of courses along with the count of students enrolled in each course. Use a JOIN operation between the "Courses" table and the "Enrollments" table.

QUERY:-

```
SELECT Courses.CourseID, Courses.CourseName,
COUNT(Enrollments.StudentID) AS EnrolledStudents
   -> FROM Courses
   -> LEFT JOIN Enrollments ON Courses.CourseID =
Enrollments.CourseID
   -> GROUP BY Courses.CourseID, Courses.CourseName;
```

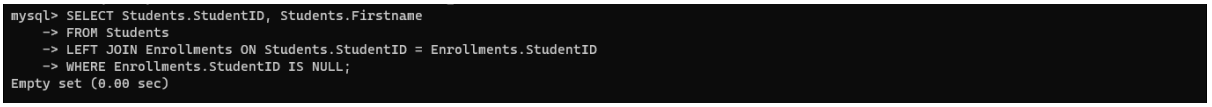


```
mysql> SELECT Courses.CourseID, Courses.CourseName, COUNT(Enrollments.StudentID) AS EnrolledStudents
    -> FROM Courses
    -> LEFT JOIN Enrollments ON Courses.CourseID = Enrollments.CourseID
    -> GROUP BY Courses.CourseID, Courses.CourseName;
+----------+--------------------+------------------+
| CourseID | CourseName         | EnrolledStudents |
+----------+--------------------+------------------+
|      101 | Mathematics        |                2 |
|      102 | History            |                1 |
|      103 | Computer Science   |                1 |
|      104 | English Literature |                1 |
|      105 | Physics            |                1 |
|      106 | Chemistry          |                1 |
|      107 | Biology            |                1 |
|      108 | Art                |                1 |
|      109 | Music              |                1 |
|      110 | Economics          |                1 |
+----------+--------------------+------------------+
10 rows in set (0.01 sec)

mysql>
```

3.Write an SQL query to find the names of students who have not enrolled in any course. Use a LEFT JOIN between the "Students" table and the "Enrollments" table to identify students without enrollments.

QUERY:-

```
SELECT Students.StudentID, Students.Firstname
   -> FROM Students
   -> LEFT JOIN Enrollments ON Students.StudentID =
Enrollments.StudentID
   -> WHERE Enrollments.StudentID IS NULL;
```



```
mysql> SELECT Students.StudentID, Students.Firstname
    -> FROM Students
    -> LEFT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID
    -> WHERE Enrollments.StudentID IS NULL;
Empty set (0.00 sec)
```

4.Write an SQL query to retrieve the first name, last name of students, and the names of the courses they are enrolled in.

Use JOIN operations between the "Students" table and the "Enrollments" and "Courses" tables.

QUERY:-

```
SELECT
   ->  Students.FirstName,
   -> Students.LastName,
   -> Courses.CourseName
   -> FROM Students
   -> JOIN Enrollments ON Students.StudentID =
Enrollments.StudentID
   -> JOIN Courses ON Enrollments.CourseID =
Courses.CourseID;
```

```
mysql> SELECT
    ->  Students.FirstName,
    -> Students.LastName,
    -> Courses.CourseName
    -> FROM Students
    -> JOIN Enrollments ON Students.StudentID = Enrollments.StudentID
    -> JOIN Courses ON Enrollments.CourseID = Courses.CourseID;
+-----------+----------+--------------------+
| FirstName | LastName | CourseName         |
+-----------+----------+--------------------+
| Raj       | Roy      | Mathematics        |
| John      | Doe      | Mathematics        |
| Rohit     | Ray      | History            |
| Rahul     | Roy      | Computer Science   |
| Akash     | Dey      | English Literature |
| Amit      | Mishra   | Physics            |
| Arti      | Mishra   | Chemistry          |
| Piu       | Das      | Biology            |
| Shreya    | Sen      | Art                |
| Anik      | Singh    | Music              |
| Ayesha    | Singh    | Economics          |
+-----------+----------+--------------------+
11 rows in set (0.00 sec)

mysql>
```

5.Create a query to list the names of teachers and the courses they are assigned to. Join the "Teacher" table with the "Courses" table.

QUERY:-

```
SELECT
   -> Teacher.TeacherID,
   -> Teacher.FirstName,
   -> Courses.CourseName
   -> FROM Teacher
   -> JOIN Courses ON Teacher.TeacherID =
Courses.TeacherID;
```

```
mysql> SELECT
    -> Teacher.TeacherID,
    -> Teacher.FirstName,
    -> Courses.CourseName
    -> FROM Teacher
    -> JOIN Courses ON Teacher.TeacherID = Courses.TeacherID;
+-----------+-----------+--------------------+
| TeacherID | FirstName | CourseName         |
+-----------+-----------+--------------------+
|         1 | Aditya    | Mathematics        |
|         2 | Aman      | History            |
|         1 | Aditya    | Computer Science   |
|         5 | Priya     | English Literature |
|         4 | Raman     | Physics            |
|         6 | Shriya    | Chemistry          |
|         7 | Daya      | Biology            |
|         2 | Aman      | Art                |
|         5 | Priya     | Music              |
|         1 | Aditya    | Economics          |
+-----------+-----------+--------------------+
10 rows in set (0.00 sec)
```

6.Retrieve a list of students and their enrollment dates for a specific course. You'll need to join the "Students" table with the "Enrollments" and "Courses" tables.

QUERY:-

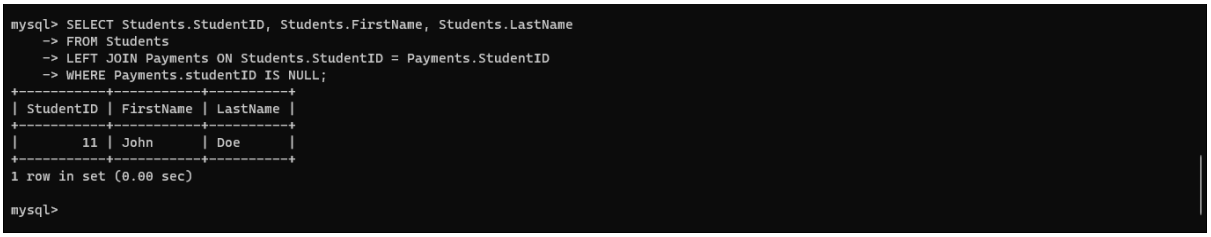
```
SELECT
   -> Students.StudentID,
   -> Students.FirstName,
   -> Students.LastName,
   -> Enrollments.EnrollmentDate
   -> FROM Students
   -> JOIN Enrollments ON Students.StudentID = Enrollments.StudentID
   -> JOIN Courses ON Enrollments.CourseID = Courses.CourseID
   -> WHERE Courses.CourseID = 104;
```

```
mysql> SELECT
    -> Students.StudentID,
    -> Students.FirstName,
    -> Students.LastName,
    -> Enrollments.EnrollmentDate
    -> FROM Students
    -> JOIN Enrollments ON Students.StudentID = Enrollments.StudentID
    -> JOIN Courses ON Enrollments.CourseID = Courses.CourseID
    -> WHERE Courses.CourseID = 104;
+-----------+-----------+----------+----------------+
| StudentID | FirstName | LastName | EnrollmentDate |
+-----------+-----------+----------+----------------+
|         4 | Akash     | Dey      | 2024-01-18     |
+-----------+-----------+----------+----------------+
1 row in set (0.00 sec)

mysql>
```

7. Find the names of students who have not made any payments. Use a LEFT JOIN between the "Students" table and the "Payments" table and filter for students with NULL payment records

QUERY:-

```
SELECT Students.StudentID, Students.FirstName,
Students.LastName
   -> FROM Students
   -> LEFT JOIN Payments ON Students.StudentID =
Payments.StudentID
   -> WHERE Payments.studentID IS NULL;
```

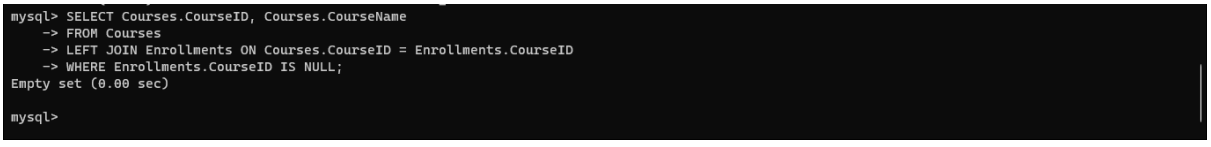


```
mysql> SELECT Students.StudentID, Students.FirstName, Students.LastName
    -> FROM Students
    -> LEFT JOIN Payments ON Students.StudentID = Payments.StudentID
    -> WHERE Payments.studentID IS NULL;
+-----------+-----------+----------+
| StudentID | FirstName | LastName |
+-----------+-----------+----------+
|        11 | John      | Doe      |
+-----------+-----------+----------+
1 row in set (0.00 sec)

mysql>
```

8. Write a query to identify courses that have no enrollments. You'll need to use a LEFT JOIN between the "Courses" table and the "Enrollments" table and filter for courses with NULL enrollment records.

QUERY:-

```
SELECT Courses.CourseID, Courses.CourseName
   -> FROM Courses
   -> LEFT JOIN Enrollments ON Courses.CourseID =
Enrollments.CourseID
   -> WHERE Enrollments.CourseID IS NULL;
```



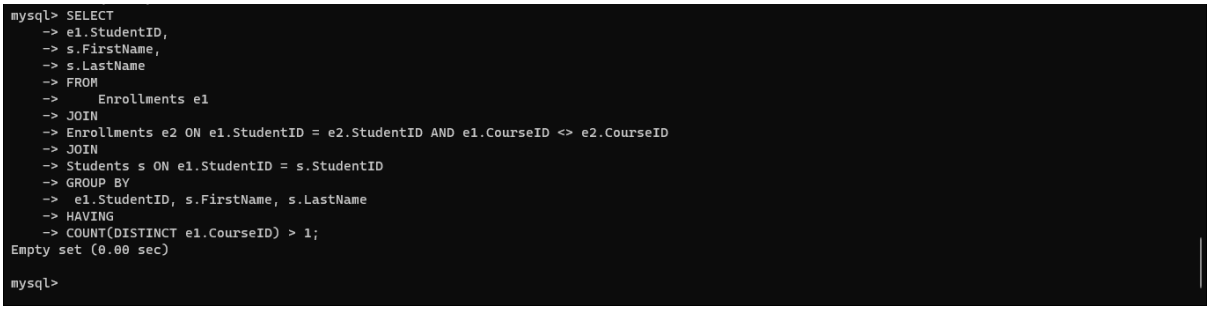
```
mysql> SELECT Courses.CourseID, Courses.CourseName
    -> FROM Courses
    -> LEFT JOIN Enrollments ON Courses.CourseID = Enrollments.CourseID
    -> WHERE Enrollments.CourseID IS NULL;
Empty set (0.00 sec)

mysql>
```

9. Identify students who are enrolled in more than one course. Use a self-join on the "Enrollments" table to find students with multiple enrollment records.

QUERY:-

```
SELECT
   -> e1.StudentID,
   -> s.FirstName,
   -> s.LastName
   -> FROM
```

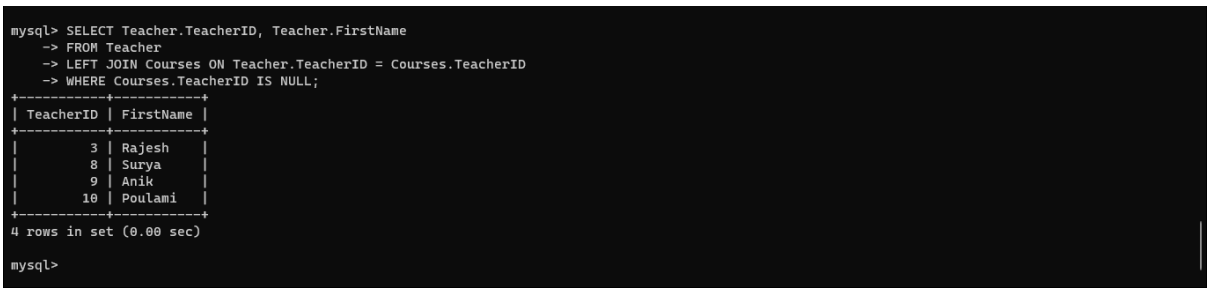
```
    ->     Enrollments e1
    -> JOIN
    -> Enrollments e2 ON e1.StudentID = e2.StudentID AND
e1.CourseID <> e2.CourseID
    -> JOIN
    -> Students s ON e1.StudentID = s.StudentID
    -> GROUP BY
    ->  e1.StudentID, s.FirstName, s.LastName
    -> HAVING
    -> COUNT(DISTINCT e1.CourseID) > 1;
```



10. Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher" table and the "Courses" table and filter for teachers with NULL course assignments.

QUERY:-

```
SELECT Teacher.TeacherID, Teacher.FirstName
    -> FROM Teacher
    -> LEFT JOIN Courses ON Teacher.TeacherID =
Courses.TeacherID
    -> WHERE Courses.TeacherID IS NULL;
```



Task 4. Subquery and its type:

1. Write an SQL query to calculate the average number of students enrolled in each course. Use aggregate functions and subqueries to achieve this

   QUERY:-

   SELECT

   -> Courses.CourseID,

   -> Courses.CourseName,

   -> AVG(num_students) AS AverageStudents

   -> FROM Courses

   -> LEFT JOIN (

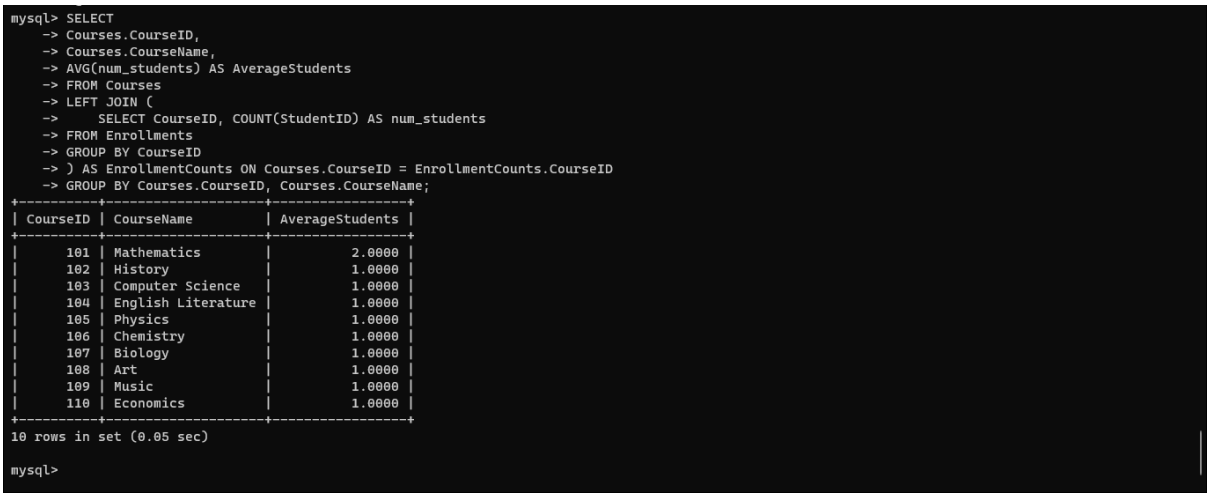
   -> SELECT CourseID, COUNT(StudentID) AS num_students

   -> FROM Enrollments

   -> GROUP BY CourseID

   ->) AS EnrollmentCounts ON Courses.CourseID = EnrollmentCounts.CourseID

   -> GROUP BY Courses.CourseID, Courses.CourseName;



```
mysql> SELECT
    -> Courses.CourseID,
    -> Courses.CourseName,
    -> AVG(num_students) AS AverageStudents
    -> FROM Courses
    -> LEFT JOIN (
    ->     SELECT CourseID, COUNT(StudentID) AS num_students
    -> FROM Enrollments
    -> GROUP BY CourseID
    -> ) AS EnrollmentCounts ON Courses.CourseID = EnrollmentCounts.CourseID
    -> GROUP BY Courses.CourseID, Courses.CourseName;
+----------+--------------------+-----------------+
| CourseID | CourseName         | AverageStudents |
+----------+--------------------+-----------------+
|      101 | Mathematics        |          2.0000 |
|      102 | History            |          1.0000 |
|      103 | Computer Science   |          1.0000 |
|      104 | English Literature |          1.0000 |
|      105 | Physics            |          1.0000 |
|      106 | Chemistry          |          1.0000 |
|      107 | Biology            |          1.0000 |
|      108 | Art                |          1.0000 |
|      109 | Music              |          1.0000 |
|      110 | Economics          |          1.0000 |
+----------+--------------------+-----------------+
10 rows in set (0.05 sec)

mysql>
```

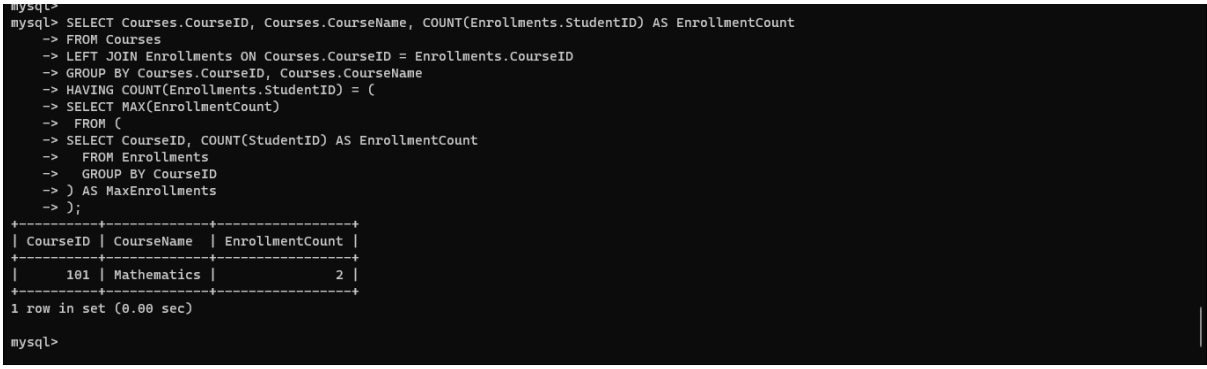
2. Identify the student(s) who made the highest payment. Use a subquery to find the maximum payment amount and then retrieve the student(s) associated with that amount.

   QUERY:-

SELECT Students.studentID, Students.FirstName, Students.LastName, Payments.Amount AS HighestPayment
-> FROM Students
-> JOIN Payments ON Students.StudentID = Payments.StudentID
-> WHERE Payments.Amount = (
-> SELECT MAX(Amount)
-> FROM Payments
->);

3. Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the course(s) with the maximum enrollment count.
QUERY:-
SELECT Courses.CourseID, Courses.CourseName, COUNT(Enrollments.StudentID) AS EnrollmentCount
-> FROM Courses
-> LEFT JOIN Enrollments ON Courses.CourseID = Enrollments.CourseID
-> GROUP BY Courses.CourseID, Courses.CourseName
-> HAVING COUNT(Enrollments.StudentID) = (
-> SELECT MAX(EnrollmentCount)
-> FROM (
-> SELECT CourseID, COUNT(StudentID) AS EnrollmentCount
-> FROM Enrollments

-> GROUP BY CourseID
->) AS MaxEnrollments
->);



```
mysql>
mysql> SELECT Courses.CourseID, Courses.CourseName, COUNT(Enrollments.StudentID) AS EnrollmentCount
    -> FROM Courses
    -> LEFT JOIN Enrollments ON Courses.CourseID = Enrollments.CourseID
    -> GROUP BY Courses.CourseID, Courses.CourseName
    -> HAVING COUNT(Enrollments.StudentID) = (
    -> SELECT MAX(EnrollmentCount)
    ->  FROM (
    -> SELECT CourseID, COUNT(StudentID) AS EnrollmentCount
    ->   FROM Enrollments
    ->   GROUP BY CourseID
    -> ) AS MaxEnrollments
    -> );
+----------+-------------+-----------------+
| CourseID | CourseName  | EnrollmentCount |
+----------+-------------+-----------------+
|      101 | Mathematics |               2 |
+----------+-------------+-----------------+
1 row in set (0.00 sec)

mysql>
```

4. Calculate the total payments made to courses taught by each teacher. Use subqueries to sum payments for each teacher's courses.

   QUERY:-

   SELECT
   -> Teacher.TeacherID,
   -> Teacher.FirstName,
   -> SUM(Payments.Amount) AS TotalPayments
   -> FROM
   -> Teacher
   -> LEFT JOIN
   -> Courses ON Teacher.TeacherID = Courses.TeacherID
   -> LEFT JOIN
   -> Enrollments ON Courses.CourseID = Enrollments.CourseID
   -> LEFT JOIN
   -> Payments ON Enrollments.StudentID = Payments.StudentID
   -> GROUP BY
   -> Teacher.TeacherID, Teacher.FirstName;

```
mysql> SELECT
    -> Teacher.TeacherID,
    -> Teacher.FirstName,
    ->  SUM(Payments.Amount) AS TotalPayments
    -> FROM
    -> Teacher
    -> LEFT JOIN
    ->  Courses ON Teacher.TeacherID = Courses.TeacherID
    -> LEFT JOIN
    -> Enrollments ON Courses.CourseID = Enrollments.CourseID
    -> LEFT JOIN
    -> Payments ON Enrollments.StudentID = Payments.StudentID
    -> GROUP BY
    -> Teacher.TeacherID, Teacher.FirstName;
+-----------+-----------+---------------+
| TeacherID | FirstName | TotalPayments |
+-----------+-----------+---------------+
|         1 | Aditya    |        396.50 |
|         2 | Aman      |        225.80 |
|         3 | Rajesh    |          NULL |
|         4 | Raman     |        110.80 |
|         5 | Priya     |        170.45 |
|         6 | Shriya    |        200.25 |
|         7 | Daya      |         85.30 |
|         8 | Surya     |          NULL |
|         9 | Anik      |          NULL |
|        10 | Poulami   |          NULL |
+-----------+-----------+---------------+
10 rows in set (0.00 sec)

mysql>
```

5. Identify students who are enrolled in all available courses. Use subqueries to compare a student's enrollments with the total number of courses.

   QUERY:-

   SELECT

   -> Students.StudentID,

   -> Students.FirstName,

   -> Students.LastName

   -> FROM

   -> Students

   -> WHERE

   -> (SELECT COUNT(DISTINCT CourseID) FROM Courses) =

   -> (SELECT COUNT(DISTINCT CourseID) FROM Enrollments WHERE Enrollments.StudentID = Students.StudentID);

```
mysql> SELECT
    -> Students.StudentID,
    ->  Students.FirstName,
    ->  Students.LastName
    -> FROM
    -> Students
    -> WHERE
    ->  (SELECT COUNT(DISTINCT CourseID) FROM Courses) =
    -> (SELECT COUNT(DISTINCT CourseID) FROM Enrollments WHERE Enrollments.StudentID = Students.StudentID);
Empty set (0.00 sec)

mysql>
```

6. Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to find teachers with no course assignments.
   QUERY:-
   SELECT
   -> TeacherID,
   -> FirstName
   -> FROM
   -> Teacher
   -> WHERE
   -> TeacherID NOT IN (
   -> SELECT DISTINCT TeacherID
   -> FROM Courses
   ->);

```
mysql> SELECT
    -> TeacherID,
    ->  FirstName
    -> FROM
    -> Teacher
    -> WHERE
    -> TeacherID NOT IN (
    -> SELECT DISTINCT TeacherID
    -> FROM Courses
    ->     );
+-----------+-----------+
| TeacherID | FirstName |
+-----------+-----------+
|         3 | Rajesh    |
|         8 | Surya     |
|         9 | Anik      |
|        10 | Poulami   |
+-----------+-----------+
4 rows in set (0.01 sec)

mysql>
```

7. Calculate the average age of all students. Use subqueries to calculate the age of each student based on their date of birth.
   QUERY:-
   SELECT
   -> AVG(TIMESTAMPDIFF(YEAR, Students.DateOfBirth, CURDATE())) AS AverageAge
   -> FROM
   -> Students;

```
mysql> SELECT
    ->  AVG(TIMESTAMPDIFF(YEAR, Students.DateOfBirth, CURDATE())) AS AverageAge
    -> FROM
    ->     Students;
+------------+
| AverageAge |
+------------+
|    24.5455 |
+------------+
1 row in set (0.00 sec)

mysql>
```

8. Identify courses with no enrollments. Use subqueries to find courses without enrollment records.
   QUERY:-
   SELECT
   -> CourseID,
   -> CourseName
   -> FROM
   -> Courses
   -> WHERE
   -> CourseID NOT IN (
   -> SELECT DISTINCT CourseID
   -> FROM Enrollments
   ->);

```
mysql> SELECT
    -> CourseID,
    -> CourseName
    -> FROM
    -> Courses
    -> WHERE
    -> CourseID NOT IN (
    -> SELECT DISTINCT CourseID
    -> FROM Enrollments
    ->  );
Empty set (0.00 sec)

mysql>
```

9. Calculate the total payments made by each student for each course they are enrolled in. Use subqueries and aggregate functions to sum payments.
   QUERY:-
   SELECT
   -> S.StudentID,
   -> S.FirstName,
   -> COALESCE(SUM(P.Amount), 0) AS TotalPayments
   -> FROM
   -> Students S

-> LEFT JOIN
-> Payments P ON S.StudentID = P.StudentID
-> GROUP BY
-> S.StudentID,S.FirstName;

```
mysql> SELECT
    ->     S.StudentID,
    -> S.FirstName,
    -> COALESCE(SUM(P.Amount), 0) AS TotalPayments
    -> FROM
    ->     Students S
    -> LEFT JOIN
    ->     Payments P ON S.StudentID = P.StudentID
    -> GROUP BY
    ->     S.StudentID,S.FirstName;
+-----------+-----------+---------------+
| StudentID | FirstName | TotalPayments |
+-----------+-----------+---------------+
|         1 | Raj       |        180.00 |
|         2 | Rohit     |         75.20 |
|         3 | Rahul     |        120.75 |
|         4 | Akash     |         90.00 |
|         5 | Amit      |        110.80 |
|         6 | Arti      |        200.25 |
|         7 | Piu       |         85.30 |
|         8 | Shreya    |        150.60 |
|         9 | Anik      |         80.45 |
|        10 | Ayesha    |         95.75 |
|        11 | John      |          0.00 |
+-----------+-----------+---------------+
11 rows in set (0.01 sec)

mysql>
```

10. Identify students who have made more than one payment. Use subqueries and aggregate functions to count payments per student and filter for those with counts greater than one.

SELECT
-> StudentID,
-> FirstName,
-> PaymentCount
-> FROM
-> (
-> SELECT
-> S.StudentID,
-> S.FirstName,
-> COUNT(P.PaymentID) AS PaymentCount
-> FROM
-> Students S
-> LEFT JOIN
-> Payments P ON S.StudentID = P.StudentID
-> GROUP BY

```
    ->             S.StudentID, S.FirstName
    ->  ) AS PaymentCounts
    -> WHERE
    ->     PaymentCount > 1;
```

```
mysql> SELECT
    ->     StudentID,
    -> FirstName,
    ->  PaymentCount
    -> FROM
    ->     (
    ->         SELECT
    -> S.StudentID,
    -> S.FirstName,
    -> COUNT(P.PaymentID) AS PaymentCount
    -> FROM
    ->             Students S
    -> LEFT JOIN
    ->             Payments P ON S.StudentID = P.StudentID
    ->  GROUP BY
    ->             S.StudentID, S.FirstName
    ->  ) AS PaymentCounts
    -> WHERE
    ->     PaymentCount > 1;
Empty set (0.00 sec)

mysql>
```

11. Write an SQL query to calculate the total payments made by each student. Join the "Students" table with the "Payments" table and use GROUP BY to calculate the sum of payments for each student.

```
SELECT
    -> S.StudentID,
    -> S.FirstName,
    -> SUM(P.Amount) AS TotalPayments
    -> FROM
    ->     Students S
    -> LEFT JOIN
    ->     Payments P ON S.StudentID = P.StudentID
    -> GROUP BY
    ->     S.StudentID, S.FirstName
    -> ;
```

```
mysql> SELECT
    -> S.StudentID,
    -> S.FirstName,
    -> SUM(P.Amount) AS TotalPayments
    -> FROM
    ->     Students S
    -> LEFT JOIN
    ->     Payments P ON S.StudentID = P.StudentID
    -> GROUP BY
    ->     S.StudentID, S.FirstName
    -> ;
+-----------+-----------+---------------+
| StudentID | FirstName | TotalPayments |
+-----------+-----------+---------------+
|         1 | Raj       |        180.00 |
|         2 | Rohit     |         75.20 |
|         3 | Rahul     |        120.75 |
|         4 | Akash     |         90.00 |
|         5 | Amit      |        110.80 |
|         6 | Arti      |        200.25 |
|         7 | Piu       |         85.30 |
|         8 | Shreya    |        150.60 |
|         9 | Anik      |         80.45 |
|        10 | Ayesha    |         95.75 |
|        11 | John      |          NULL |
+-----------+-----------+---------------+
11 rows in set (0.01 sec)

mysql>
```

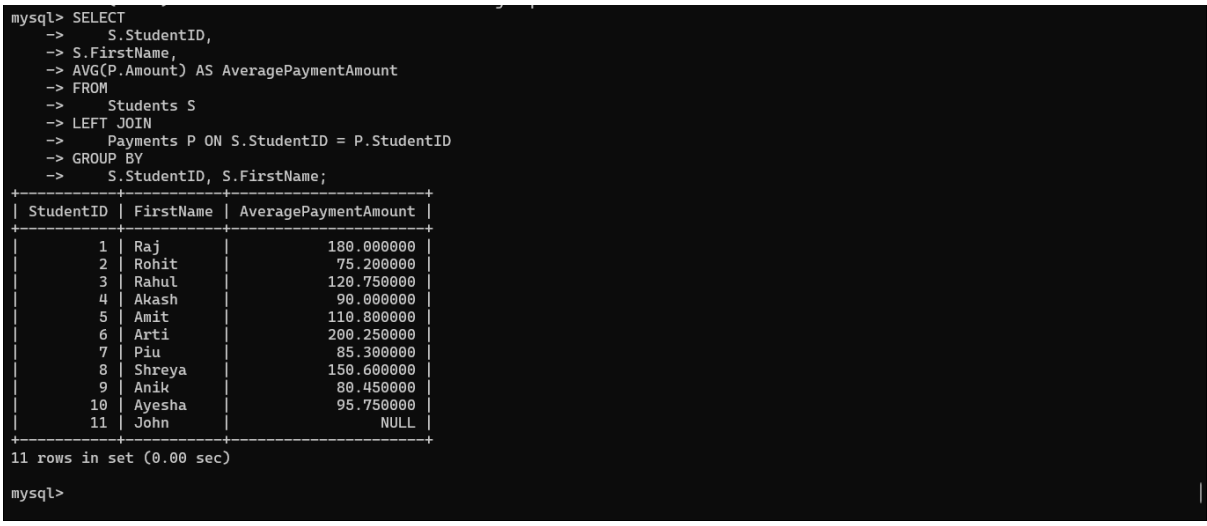
12. Retrieve a list of course names along with the count of students enrolled in each course. Use JOIN operations between the "Courses" table and the "Enrollments" table and GROUP BY to count enrollments.

```
SELECT
    ->     C.CourseID,
    -> C.CourseName,
    -> COUNT(E.StudentID) AS EnrolledStudentsCount
    -> FROM
    ->     Courses C
    -> LEFT JOIN
    ->     Enrollments E ON C.CourseID = E.CourseID
    -> GROUP BY
    ->     C.CourseID, C.CourseName;
```

```
mysql> SELECT
    ->     C.CourseID,
    -> C.CourseName,
    -> COUNT(E.StudentID) AS EnrolledStudentsCount
    -> FROM
    ->     Courses C
    -> LEFT JOIN
    ->     Enrollments E ON C.CourseID = E.CourseID
    -> GROUP BY
    ->     C.CourseID, C.CourseName;
+----------+--------------------+-----------------------+
| CourseID | CourseName         | EnrolledStudentsCount |
+----------+--------------------+-----------------------+
|      101 | Mathematics        |                     2 |
|      102 | History            |                     1 |
|      103 | Computer Science   |                     1 |
|      104 | English Literature |                     1 |
|      105 | Physics            |                     1 |
|      106 | Chemistry          |                     1 |
|      107 | Biology            |                     1 |
|      108 | Art                |                     1 |
|      109 | Music              |                     1 |
|      110 | Economics          |                     1 |
+----------+--------------------+-----------------------+
10 rows in set (0.02 sec)

mysql>
```

13. Calculate the average payment amount made by students. Use JOIN operations between the "Students" table and the "Payments" table and GROUP BY to calculate the average.

```
SELECT
    ->     S.StudentID,
    -> S.FirstName,
    -> AVG(P.Amount) AS AveragePaymentAmount
    -> FROM
    ->     Students S
    -> LEFT JOIN
    ->     Payments P ON S.StudentID = P.StudentID
    -> GROUP BY
    ->     S.StudentID, S.FirstName;
```



```
mysql> SELECT
    ->     S.StudentID,
    -> S.FirstName,
    -> AVG(P.Amount) AS AveragePaymentAmount
    -> FROM
    ->     Students S
    -> LEFT JOIN
    ->     Payments P ON S.StudentID = P.StudentID
    -> GROUP BY
    ->     S.StudentID, S.FirstName;
+-----------+-----------+----------------------+
| StudentID | FirstName | AveragePaymentAmount |
+-----------+-----------+----------------------+
|         1 | Raj       |           180.000000 |
|         2 | Rohit     |            75.200000 |
|         3 | Rahul     |           120.750000 |
|         4 | Akash     |            90.000000 |
|         5 | Amit      |           110.800000 |
|         6 | Arti      |           200.250000 |
|         7 | Piu       |            85.300000 |
|         8 | Shreya    |           150.600000 |
|         9 | Anik      |            80.450000 |
|        10 | Ayesha    |            95.750000 |
|        11 | John      |                 NULL |
+-----------+-----------+----------------------+
11 rows in set (0.00 sec)

mysql>
```