Task 1. Database Design:

1. Create the database named "SISDB"

```
mysql> CREATE DATABASE SISDB;
Query OK, 1 row affected (1.32 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sisdb              |
| student            |
| sys                |
| techshop           |
| world              |
+--------------------+
9 rows in set (0.96 sec)
```

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

```
mysql> CREATE TABLE Students (
    ->     student_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     first_name VARCHAR(255),
    ->     last_name VARCHAR(255),
    ->     date_of_birth DATE,
    ->     email VARCHAR(255),
    ->     phone_number VARCHAR(15)
    -> );
Query OK, 0 rows affected (9.08 sec)

mysql> desc students;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| student_id    | int          | NO   | PRI | NULL    | auto_increment |
| first_name    | varchar(255) | YES  |     | NULL    |                |
| last_name     | varchar(255) | YES  |     | NULL    |                |
| date_of_birth | date         | YES  |     | NULL    |                |
| email         | varchar(255) | YES  |     | NULL    |                |
| phone_number  | varchar(15)  | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
6 rows in set (3.64 sec)
```

b. Courses

```
mysql> CREATE TABLE Courses (
    ->     course_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     course_name VARCHAR(255),
    ->     credits INT,
    ->     teacher_id INT,
    ->     FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
    -> );
Query OK, 0 rows affected (2.79 sec)

mysql> desc courses;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| course_id   | int          | NO   | PRI | NULL    | auto_increment |
| course_name | varchar(255) | YES  |     | NULL    |                |
| credits     | int          | YES  |     | NULL    |                |
| teacher_id  | int          | YES  | MUL | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.10 sec)
```

c. Enrollments

```
mysql> CREATE TABLE Enrollments (
    ->     enrollment_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     student_id INT,
    ->     course_id INT,
    ->     enrollment_date DATE,
    ->     FOREIGN KEY (student_id) REFERENCES Students(student_id),
    ->     FOREIGN KEY (course_id) REFERENCES Courses(course_id)
    -> );
Query OK, 0 rows affected (5.22 sec)

mysql> desc enrollments;
+-----------------+------+------+-----+---------+----------------+
| Field           | Type | Null | Key | Default | Extra          |
+-----------------+------+------+-----+---------+----------------+
| enrollment_id   | int  | NO   | PRI | NULL    | auto_increment |
| student_id      | int  | YES  | MUL | NULL    |                |
| course_id       | int  | YES  | MUL | NULL    |                |
| enrollment_date | date | YES  |     | NULL    |                |
+-----------------+------+------+-----+---------+----------------+
4 rows in set (0.12 sec)
```

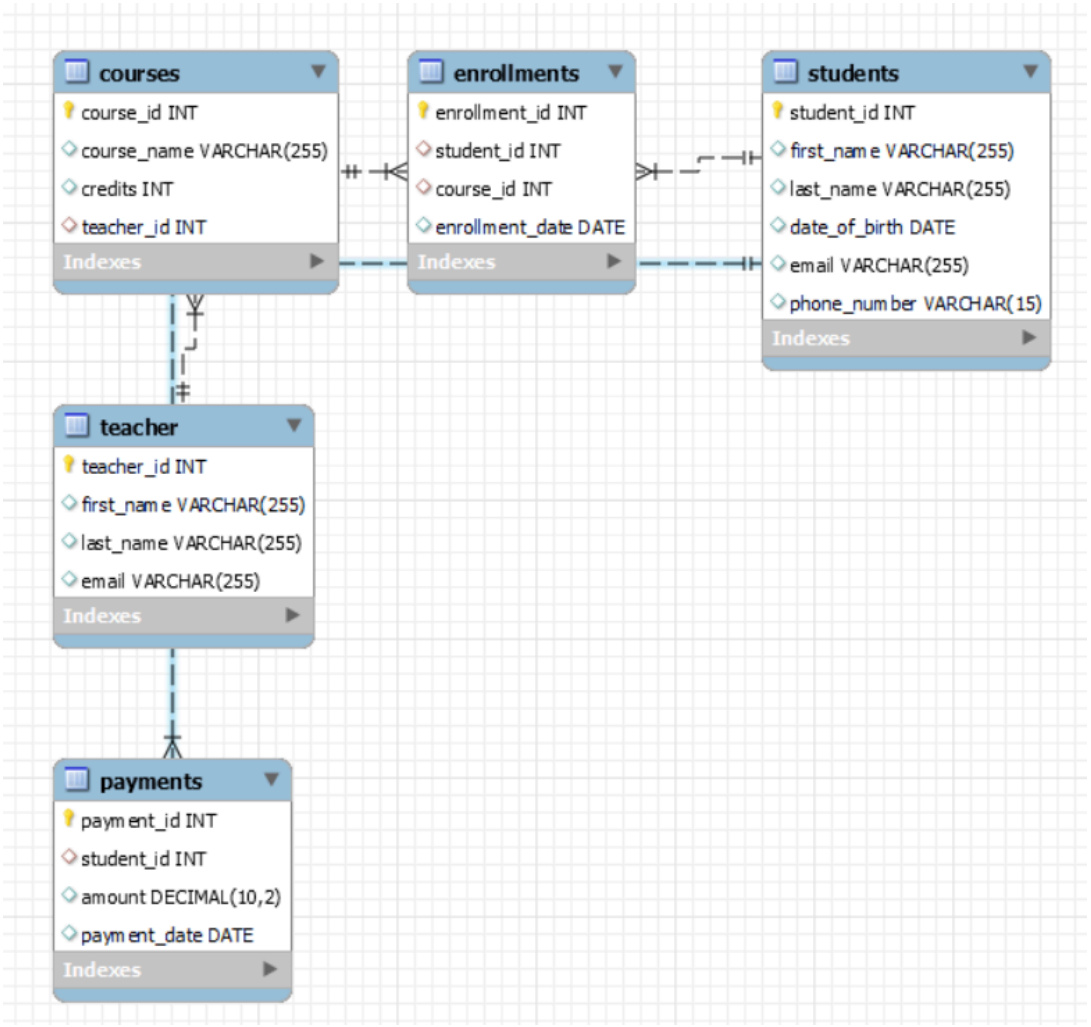
d. Teacher

```
mysql> CREATE TABLE Teacher (
    ->     teacher_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     first_name VARCHAR(255),
    ->     last_name VARCHAR(255),
    ->     email VARCHAR(255)
    -> );
Query OK, 0 rows affected (1.39 sec)

mysql> desc teacher;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| teacher_id | int          | NO   | PRI | NULL    | auto_increment |
| first_name | varchar(255) | YES  |     | NULL    |                |
| last_name  | varchar(255) | YES  |     | NULL    |                |
| email      | varchar(255) | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
4 rows in set (0.03 sec)
```

e. Payments

```
mysql> CREATE TABLE Payments (
    ->     payment_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     student_id INT,
    ->     amount DECIMAL(10, 2),
    ->     payment_date DATE,
    ->     FOREIGN KEY (student_id) REFERENCES Students(student_id)
    -> );
Query OK, 0 rows affected (7.31 sec)

mysql> desc payments;
+--------------+---------------+------+-----+---------+----------------+
| Field        | Type          | Null | Key | Default | Extra          |
+--------------+---------------+------+-----+---------+----------------+
| payment_id   | int           | NO   | PRI | NULL    | auto_increment |
| student_id   | int           | YES  | MUL | NULL    |                |
| amount       | decimal(10,2) | YES  |     | NULL    |                |
| payment_date | date          | YES  |     | NULL    |                |
+--------------+---------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

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



4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.

```
mysql> CREATE TABLE Students (
    ->     student_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     first_name VARCHAR(255),
    ->     last_name VARCHAR(255),
    ->     date_of_birth DATE,
    ->     email VARCHAR(255),
    ->     phone_number VARCHAR(15)
    -> );
Query OK, 0 rows affected (9.08 sec)

mysql> desc students;
+---------------+-------------+------+-----+---------+----------------+
| Field         | Type        | Null | Key | Default | Extra          |
+---------------+-------------+------+-----+---------+----------------+
| student_id    | int         | NO   | PRI | NULL    | auto_increment |
| first_name    | varchar(255) | YES  |     | NULL    |                |
| last_name     | varchar(255) | YES  |     | NULL    |                |
| date_of_birth | date        | YES  |     | NULL    |                |
| email         | varchar(255) | YES  |     | NULL    |                |
| phone_number  | varchar(15)  | YES  |     | NULL    |                |
+---------------+-------------+------+-----+---------+----------------+
6 rows in set (3.64 sec)
```

```
mysql> CREATE TABLE Teacher (
    ->     teacher_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     first_name VARCHAR(255),
    ->     last_name VARCHAR(255),
    ->     email VARCHAR(255)
    -> );
Query OK, 0 rows affected (1.39 sec)

mysql> desc teacher;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| teacher_id | int         | NO   | PRI | NULL    | auto_increment |
| first_name | varchar(255) | YES  |     | NULL    |                |
| last_name  | varchar(255) | YES  |     | NULL    |                |
| email      | varchar(255) | YES  |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
4 rows in set (0.03 sec)

mysql> CREATE TABLE Courses (
    ->     course_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     course_name VARCHAR(255),
    ->     credits INT,
    ->     teacher_id INT,
    ->     FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
    -> );
Query OK, 0 rows affected (2.79 sec)

mysql> desc courses;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| course_id   | int         | NO   | PRI | NULL    | auto_increment |
| course_name | varchar(255) | YES  |     | NULL    |                |
| credits     | int         | YES  |     | NULL    |                |
| teacher_id  | int         | YES  | MUL | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
4 rows in set (0.10 sec)
```

```
mysql> CREATE TABLE Enrollments (
    ->     enrollment_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     student_id INT,
    ->     course_id INT,
    ->     enrollment_date DATE,
    ->     FOREIGN KEY (student_id) REFERENCES Students(student_id),
    ->     FOREIGN KEY (course_id) REFERENCES Courses(course_id)
    -> );
Query OK, 0 rows affected (5.22 sec)

mysql> desc enrollments;
+-----------------+------+------+-----+---------+----------------+
| Field           | Type | Null | Key | Default | Extra          |
+-----------------+------+------+-----+---------+----------------+
| enrollment_id   | int  | NO   | PRI | NULL    | auto_increment |
| student_id      | int  | YES  | MUL | NULL    |                |
| course_id       | int  | YES  | MUL | NULL    |                |
| enrollment_date | date | YES  |     | NULL    |                |
+-----------------+------+------+-----+---------+----------------+
4 rows in set (0.12 sec)

mysql> CREATE TABLE Payments (
    ->     payment_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     student_id INT,
    ->     amount DECIMAL(10, 2),
    ->     payment_date DATE,
    ->     FOREIGN KEY (student_id) REFERENCES Students(student_id)
    -> );
Query OK, 0 rows affected (7.31 sec)

mysql> desc payments;
+--------------+---------------+------+-----+---------+----------------+
| Field        | Type          | Null | Key | Default | Extra          |
+--------------+---------------+------+-----+---------+----------------+
| payment_id   | int           | NO   | PRI | NULL    | auto_increment |
| student_id   | int           | YES  | MUL | NULL    |                |
| amount       | decimal(10,2) | YES  |     | NULL    |                |
| payment_date | date          | YES  |     | NULL    |                |
+--------------+---------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

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

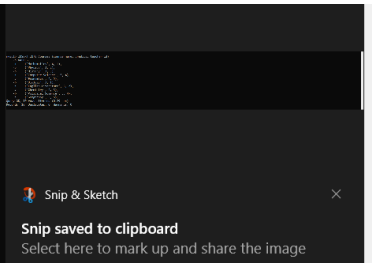
i. Students

```
mysql> INSERT INTO Students (first_name, last_name, date_of_birth, email, phone_number)
    -> VALUES
    ->     ('Rahul', 'Kumar', '1995-03-15', 'rahul.kumar@example.com', '+91-9876543210'),
    ->     ('Priya', 'Sharma', '1996-06-25', 'priya.sharma@example.com', '+91-8765432109'),
    ->     ('Amit', 'Patel', '1994-12-10', 'amit.patel@example.com', '+91-7654321098'),
    ->     ('Neha', 'Verma', '1997-02-18', 'neha.verma@example.com', '+91-6543210987'),
    ->     ('Vikram', 'Singh', '1993-08-05', 'vikram.singh@example.com', '+91-5432109876'),
    ->     ('Sneha', 'Gupta', '1998-04-30', 'sneha.gupta@example.com', '+91-4321098765'),
    ->     ('Rajesh', 'Yadav', '1992-11-22', 'rajesh.yadav@example.com', '+91-3210987654'),
    ->     ('Kavita', 'Rajput', '1996-09-14', 'kavita.rajput@example.com', '+91-2109876543'),
    ->     ('Arjun', 'Malhotra', '1991-07-08', 'arjun.malhotra@example.com', '+91-1098765432'),
    ->     ('Anjali', 'Mishra', '1999-01-03', 'anjali.mishra@example.com', '+91-9876543210');
Query OK, 10 rows affected (1.33 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Students;
+------------+------------+-----------+---------------+---------------------------+---------------+
| student_id | first_name | last_name | date_of_birth | email                     | phone_number  |
+------------+------------+-----------+---------------+---------------------------+---------------+
|          1 | Rahul      | Kumar     | 1995-03-15    | rahul.kumar@example.com   | +91-9876543210 |
|          2 | Priya      | Sharma    | 1996-06-25    | priya.sharma@example.com  | +91-8765432109 |
|          3 | Amit       | Patel     | 1994-12-10    | amit.patel@example.com    | +91-7654321098 |
|          4 | Neha       | Verma     | 1997-02-18    | neha.verma@example.com    | +91-6543210987 |
|          5 | Vikram     | Singh     | 1993-08-05    | vikram.singh@example.com  | +91-5432109876 |
|          6 | Sneha      | Gupta     | 1998-04-30    | sneha.gupta@example.com   | +91-4321098765 |
|          7 | Rajesh     | Yadav     | 1992-11-22    | rajesh.yadav@example.com  | +91-3210987654 |
|          8 | Kavita     | Rajput    | 1996-09-14    | kavita.rajput@example.com | +91-2109876543 |
|          9 | Arjun      | Malhotra  | 1991-07-08    | arjun.malhotra@example.com | +91-1098765432 |
|         10 | Anjali     | Mishra    | 1999-01-03    | anjali.mishra@example.com | +91-9876543210 |
+------------+------------+-----------+---------------+---------------------------+---------------+
10 rows in set (0.06 sec)
```

ii. Courses

```
mysql> INSERT INTO Courses (course_name, credits, teacher_id)
    -> VALUES
    ->     ('Mathematics', 4, 1),
    ->     ('Physics', 3, 2),
    ->     ('History', 3, 3),
    ->     ('Computer Science', 5, 4),
    ->     ('Economics', 4, 5),
    ->     ('Biology', 3, 1),
    ->     ('English Literature', 3, 2),
    ->     ('Chemistry', 4, 3),
    ->     ('Political Science', 3, 4),
    ->     ('Geography', 3, 5);
Query OK, 10 rows affected (0.29 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
mysql> select * from courses;
+-----------+--------------------+---------+------------+
| course_id | course_name        | credits | teacher_id |
+-----------+--------------------+---------+------------+
|        11 | Mathematics        |       4 |          1 |
|        12 | Physics            |       3 |          2 |
|        13 | History            |       3 |          3 |
|        14 | Computer Science   |       5 |          4 |
|        15 | Economics          |       4 |          5 |
|        16 | Biology            |       3 |          1 |
|        17 | English Literature |       3 |          2 |
|        18 | Chemistry          |       4 |          3 |
|        19 | Political Science  |       3 |          4 |
|        20 | Geography          |       3 |          5 |
+-----------+--------------------+---------+------------+
10 rows in set (0.02 sec)
```



iii. Enrollments

```
mysql> INSERT INTO Enrollments (student_id, course_id, enrollment_date)
    -> VALUES
    ->     (1, 11, '2023-01-25'),
    ->     (2, 13, '2023-01-26'),
    ->     (3, 15, '2023-01-27'),
    ->     (4, 12, '2023-01-28'),
    ->     (5, 14, '2023-01-29'),
    ->     (6, 16, '2023-01-30'),
    ->     (7, 18, '2023-01-31'),
    ->     (8, 19, '2023-02-01'),
    ->     (9, 17, '2023-02-02'),
    ->     (10, 20, '2023-02-03');
Query OK, 10 rows affected (0.21 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|            21 |          1 |        11 | 2023-01-25      |
|            22 |          2 |        13 | 2023-01-26      |
|            23 |          3 |        15 | 2023-01-27      |
|            24 |          4 |        12 | 2023-01-28      |
|            25 |          5 |        14 | 2023-01-29      |
|            26 |          6 |        16 | 2023-01-30      |
|            27 |          7 |        18 | 2023-01-31      |
|            28 |          8 |        19 | 2023-02-01      |
|            29 |          9 |        17 | 2023-02-02      |
|            30 |         10 |        20 | 2023-02-03      |
+---------------+------------+-----------+-----------------+
10 rows in set (0.00 sec)
```

iv. Teacher

```
mysql> INSERT INTO Teacher (first_name, last_name, email)
    -> VALUES
    ->     ('Dr. Manisha', 'Rathi', 'manisha.rathi@example.com'),
    ->     ('Prof. Prakash', 'Khanna', 'prakash.khanna@example.com'),
    ->     ('Mr. Rakesh', 'Chopra', 'rakesh.chopra@example.com'),
    ->     ('Dr. Nisha', 'Saxena', 'nisha.saxena@example.com'),
    ->     ('Prof. Sanjay', 'Bhatia', 'sanjay.bhatia@example.com'),
    ->     ('Mrs. Anjali', 'Shukla', 'anjali.shukla@example.com'),
    ->     ('Dr. Rajat', 'Jain', 'rajat.jain@example.com'),
    ->     ('Prof. Preeti', 'Mishra', 'preeti.mishra@example.com'),
    ->     ('Mr. Akash', 'Srivastava', 'akash.srivastava@example.com'),
    ->     ('Mrs. Meena', 'Gupta', 'meena.gupta@example.com');
Query OK, 10 rows affected (0.17 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Teacher;
+------------+--------------+------------+-----------------------------+
| teacher_id | first_name   | last_name  | email                       |
+------------+--------------+------------+-----------------------------+
|          1 | Dr. Manisha  | Rathi      | manisha.rathi@example.com   |
|          2 | Prof. Prakash| Khanna     | prakash.khanna@example.com  |
|          3 | Mr. Rakesh   | Chopra     | rakesh.chopra@example.com   |
|          4 | Dr. Nisha    | Saxena     | nisha.saxena@example.com    |
|          5 | Prof. Sanjay | Bhatia     | sanjay.bhatia@example.com   |
|          6 | Mrs. Anjali  | Shukla     | anjali.shukla@example.com   |
|          7 | Dr. Rajat    | Jain       | rajat.jain@example.com      |
|          8 | Prof. Preeti | Mishra     | preeti.mishra@example.com   |
|          9 | Mr. Akash    | Srivastava | akash.srivastava@example.com|
|         10 | Mrs. Meena   | Gupta      | meena.gupta@example.com     |
+------------+--------------+------------+-----------------------------+
10 rows in set (0.02 sec)
```

v. Payments

```
mysql> INSERT INTO Payments (student_id, amount, payment_date)
    -> VALUES
    ->     (1, 5000.00, '2023-01-15'),
    ->     (2, 4500.00, '2023-01-16'),
    ->     (3, 5500.00, '2023-01-17'),
    ->     (4, 4000.00, '2023-01-18'),
    ->     (5, 6000.00, '2023-01-19'),
    ->     (6, 3500.00, '2023-01-20'),
    ->     (7, 4800.00, '2023-01-21'),
    ->     (8, 5200.00, '2023-01-22'),
    ->     (9, 4200.00, '2023-01-23'),
    ->     (10, 5800.00, '2023-01-24');
Query OK, 10 rows affected (0.21 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from payments;
+------------+------------+---------+--------------+
| payment_id | student_id | amount  | payment_date |
+------------+------------+---------+--------------+
|          1 |          1 | 5000.00 | 2023-01-15   |
|          2 |          2 | 4500.00 | 2023-01-16   |
|          3 |          3 | 5500.00 | 2023-01-17   |
|          4 |          4 | 4000.00 | 2023-01-18   |
|          5 |          5 | 6000.00 | 2023-01-19   |
|          6 |          6 | 3500.00 | 2023-01-20   |
|          7 |          7 | 4800.00 | 2023-01-21   |
|          8 |          8 | 5200.00 | 2023-01-22   |
|          9 |          9 | 4200.00 | 2023-01-23   |
|         10 |         10 | 5800.00 | 2023-01-24   |
+------------+------------+---------+--------------+
10 rows in set (0.00 sec)
```

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

```
mysql> INSERT INTO Students(first_name, last_name, date_of_birth, email, phone_number)
    -> VALUES
    ->     ('John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
Query OK, 1 row affected (0.18 sec)

mysql> select * from Students;
+------------+------------+-----------+---------------+-----------------------------+---------------+
| student_id | first_name | last_name | date_of_birth | email                       | phone_number  |
+------------+------------+-----------+---------------+-----------------------------+---------------+
|          1 | Rahul      | Kumar     | 1995-03-15    | rahul.kumar@example.com     | +91-9876543210 |
|          2 | Priya      | Sharma    | 1996-06-25    | priya.sharma@example.com    | +91-8765432109 |
|          3 | Amit       | Patel     | 1994-12-10    | amit.patel@example.com      | +91-7654321098 |
|          4 | Neha       | Verma     | 1997-02-18    | neha.verma@example.com      | +91-6543210987 |
|          5 | Vikram     | Singh     | 1993-08-05    | vikram.singh@example.com    | +91-5432109876 |
|          6 | Sneha      | Gupta     | 1998-04-30    | sneha.gupta@example.com     | +91-4321098765 |
|          7 | Rajesh     | Yadav     | 1992-11-22    | rajesh.yadav@example.com    | +91-3210987654 |
|          8 | Kavita     | Rajput    | 1996-09-14    | kavita.rajput@example.com   | +91-2109876543 |
|          9 | Arjun      | Malhotra  | 1991-07-08    | arjun.malhotra@example.com  | +91-1098765432 |
|         10 | Anjali     | Mishra    | 1999-01-03    | anjali.mishra@example.com   | +91-9876543210 |
|         11 | John       | Doe       | 1995-08-15    | john.doe@example.com        | 1234567890     |
+------------+------------+-----------+---------------+-----------------------------+---------------+
11 rows in set (0.00 sec)
```

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.

```
mysql> INSERT INTO Enrollments (student_id, course_id, enrollment_date)
    -> VALUES (1, 11, CURDATE());
Query OK, 1 row affected (0.21 sec)

mysql> select * from enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|            21 |          1 |        11 | 2023-01-25      |
|            22 |          2 |        13 | 2023-01-26      |
|            23 |          3 |        15 | 2023-01-27      |
|            24 |          4 |        12 | 2023-01-28      |
|            25 |          5 |        14 | 2023-01-29      |
|            26 |          6 |        16 | 2023-01-30      |
|            27 |          7 |        18 | 2023-01-31      |
|            28 |          8 |        19 | 2023-02-01      |
|            29 |          9 |        17 | 2023-02-02      |
|            30 |         10 |        20 | 2023-02-03      |
|            31 |          1 |        11 | 2024-01-16      |
+---------------+------------+-----------+-----------------+
11 rows in set (0.00 sec)
```

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

```
mysql> UPDATE Teacher
    -> SET email = 'sanjay2@gmail.com'
    -> WHERE teacher_id = 5;
Query OK, 1 row affected (0.11 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> SELECT * FROM Teachers
    -> WHERE teacher_id = 5;
+------------+-------------+-----------+-------------------+
| teacher_id | first_name  | last_name | email             |
+------------+-------------+-----------+-------------------+
|          5 | Prof. Sanjay | Bhatia    | sanjay2@gmail.com |
+------------+-------------+-----------+-------------------+
1 row in set (0.25 sec)
```

Renaming table Teacher -> Teachers

```
mysql> ALTER TABLE Teacher
    -> RENAME TO Teachers;
Query OK, 0 rows affected (4.36 sec)
```

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.

```
mysql> DELETE FROM Enrollments
    -> WHERE student_id = 1 AND course_id = 11;
Query OK, 2 rows affected (0.12 sec)

mysql> select * from Enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|            22 |          2 |        13 | 2023-01-26      |
|            23 |          3 |        15 | 2023-01-27      |
|            24 |          4 |        12 | 2023-01-28      |
|            25 |          5 |        14 | 2023-01-29      |
|            26 |          6 |        16 | 2023-01-30      |
|            27 |          7 |        18 | 2023-01-31      |
|            28 |          8 |        19 | 2023-02-01      |
|            29 |          9 |        17 | 2023-02-02      |
|            30 |         10 |        20 | 2023-02-03      |
+---------------+------------+-----------+-----------------+
9 rows in set (0.00 sec)
```

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

```
mysql> UPDATE Courses
    -> SET teacher_id = 2
    -> WHERE course_id = 14;
Query OK, 1 row affected (0.22 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from courses;
+-----------+--------------------+---------+------------+
| course_id | course_name        | credits | teacher_id |
+-----------+--------------------+---------+------------+
|        11 | Mathematics        |       4 |          1 |
|        12 | Physics            |       3 |          2 |
|        13 | History            |       3 |          3 |
|        14 | Computer Science   |       5 |          2 |
|        15 | Economics          |       4 |          5 |
|        16 | Biology            |       3 |          1 |
|        17 | English Literature |       3 |          2 |
|        18 | Chemistry          |       4 |          3 |
|        19 | Political Science  |       3 |          4 |
|        20 | Geography          |       3 |          5 |
+-----------+--------------------+---------+------------+
10 rows in set (0.00 sec)
```

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.

```
mysql> DELETE FROM Enrollments
    -> WHERE student_id = 1;
Query OK, 0 rows affected (0.04 sec)

mysql>
mysql> DELETE FROM Students
    -> WHERE student_id = 1;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (`sisdb`.`payments`, CONSTRAINT `payments_ibfk_1` FOREIGN KEY (`student_id`) REFERENCES `students` (`student_id`))
mysql> select * from Students;
+------------+------------+-----------+---------------+---------------------------+----------------+
| student_id | first_name | last_name | date_of_birth | email                     | phone_number   |
+------------+------------+-----------+---------------+---------------------------+----------------+
|          1 | Rahul      | Kumar     | 1995-03-15    | rahul.kumar@example.com   | +91-9876543210 |
|          2 | Priya      | Sharma    | 1996-06-25    | priya.sharma@example.com  | +91-8765432109 |
|          3 | Amit       | Patel     | 1994-12-10    | amit.patel@example.com    | +91-7654321098 |
|          4 | Neha       | Verma     | 1997-02-18    | neha.verma@example.com    | +91-6543210987 |
|          5 | Vikram     | Singh     | 1993-08-05    | vikram.singh@example.com  | +91-5432109876 |
|          6 | Sneha      | Gupta     | 1998-04-30    | sneha.gupta@example.com   | +91-4321098765 |
|          7 | Rajesh     | Yadav     | 1992-11-22    | rajesh.yadav@example.com  | +91-3210987654 |
|          8 | Kavita     | Rajput    | 1996-09-14    | kavita.rajput@example.com | +91-2109876543 |
|          9 | Arjun      | Malhotra  | 1991-07-08    | arjun.malhotra@example.com| +91-1098765432 |
|         10 | Anjali     | Mishra    | 1999-01-03    | anjali.mishra@example.com | +91-9876543210 |
|         11 | John       | Doe       | 1995-08-15    | john.doe@example.com      | 1234567890     |
+------------+------------+-----------+---------------+---------------------------+----------------+
11 rows in set (0.00 sec)

mysql> DELETE FROM Payments
    -> WHERE student_id = 1;
Query OK, 1 row affected (0.11 sec)

mysql> DELETE FROM Students
    -> WHERE student_id = 1;
Query OK, 1 row affected (0.12 sec)
```

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

```
mysql> UPDATE Payments
    -> SET amount = 6500.00
    -> WHERE payment_id = 3;
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * form Payments;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'form Payments' at line 1
mysql> select * fr0m Payments;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'fr0m Payments' at line 1
mysql> select * from Payments;
+------------+------------+---------+--------------+
| payment_id | student_id | amount  | payment_date |
+------------+------------+---------+--------------+
|          2 |          2 | 4500.00 | 2023-01-16   |
|          3 |          3 | 6500.00 | 2023-01-17   |
|          4 |          4 | 4000.00 | 2023-01-18   |
|          5 |          5 | 6000.00 | 2023-01-19   |
|          6 |          6 | 3500.00 | 2023-01-20   |
|          7 |          7 | 4800.00 | 2023-01-21   |
|          8 |          8 | 5200.00 | 2023-01-22   |
|          9 |          9 | 4200.00 | 2023-01-23   |
|         10 |         10 | 5800.00 | 2023-01-24   |
+------------+------------+---------+--------------+
9 rows in set (0.00 sec)
```

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.

```
mysql> SELECT Students.first_name, Students.last_name, SUM(Payments.amount) AS total_payments
    -> FROM Students
    -> JOIN Payments ON Students.student_id = Payments.student_id
    -> WHERE Students.student_id = 5;
+------------+-----------+---------------+
| first_name | last_name | total_payments |
+------------+-----------+---------------+
| Vikram     | Singh     |       6000.00 |
+------------+-----------+---------------+
1 row in set (0.00 sec)
```

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.

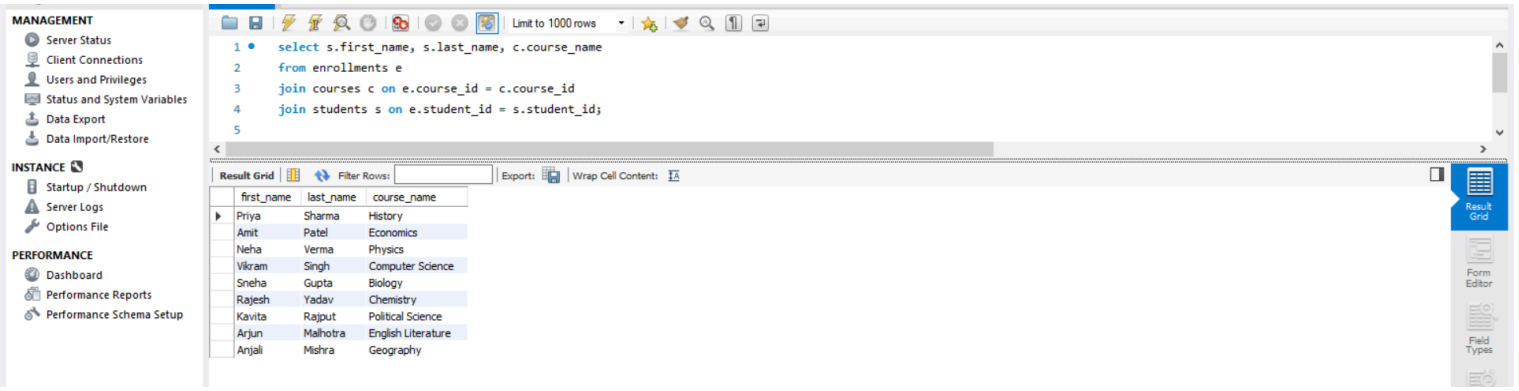
```
mysql> SELECT Courses.course_id, Courses.course_name, COUNT(Enrollments.student_id) AS enrolled_students
    -> FROM Courses
    -> LEFT JOIN Enrollments ON Courses.course_id = Enrollments.course_id
    -> GROUP BY Courses.course_id, Courses.course_name;
+-----------+--------------------+-------------------+
| course_id | course_name        | enrolled_students |
+-----------+--------------------+-------------------+
|        11 | Mathematics        |                 0 |
|        12 | Physics            |                 1 |
|        13 | History            |                 1 |
|        14 | Computer Science   |                 1 |
|        15 | Economics          |                 1 |
|        16 | Biology            |                 1 |
|        17 | English Literature |                 1 |
|        18 | Chemistry          |                 1 |
|        19 | Political Science  |                 1 |
|        20 | Geography          |                 1 |
+-----------+--------------------+-------------------+
10 rows in set (1.62 sec)
```

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.

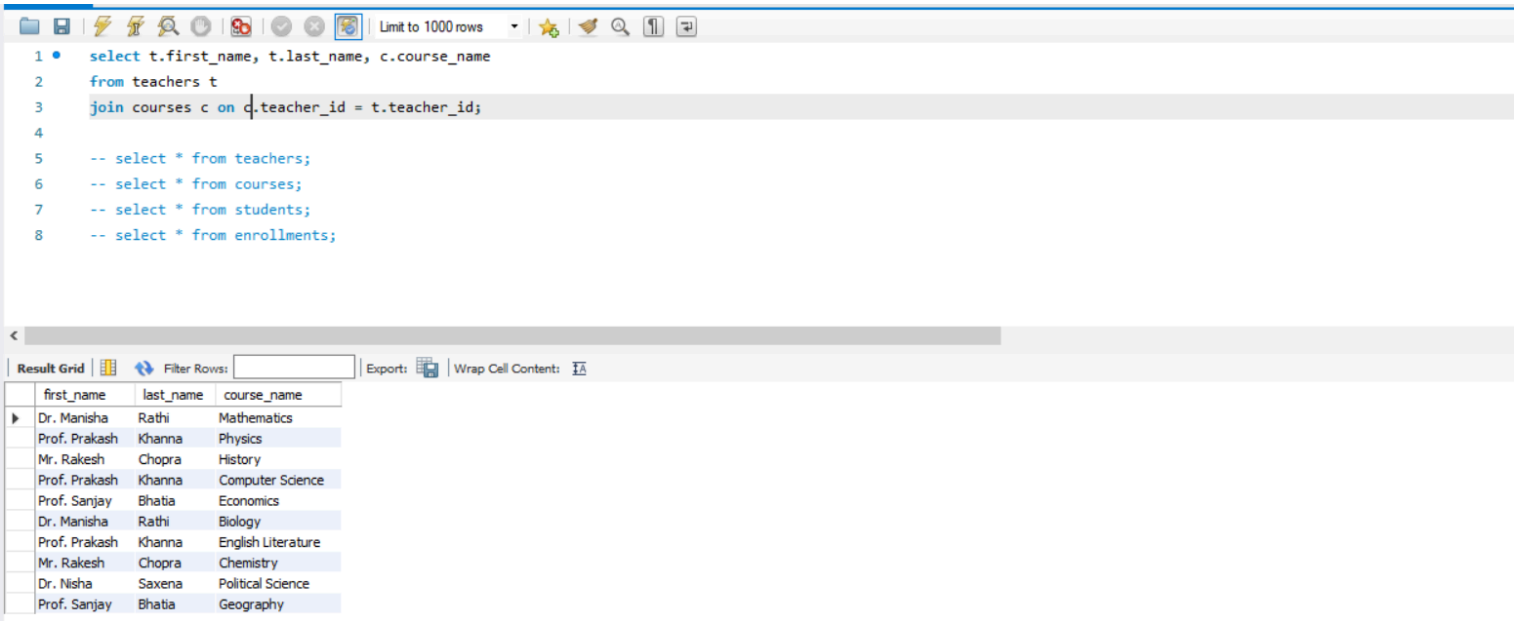
```sql
select s.first_name, s.last_name
from students s
left join enrollments e on s.student_id = e.student_id
where e.enrollment_id is null;
```

first_name	last_name
John	Doe

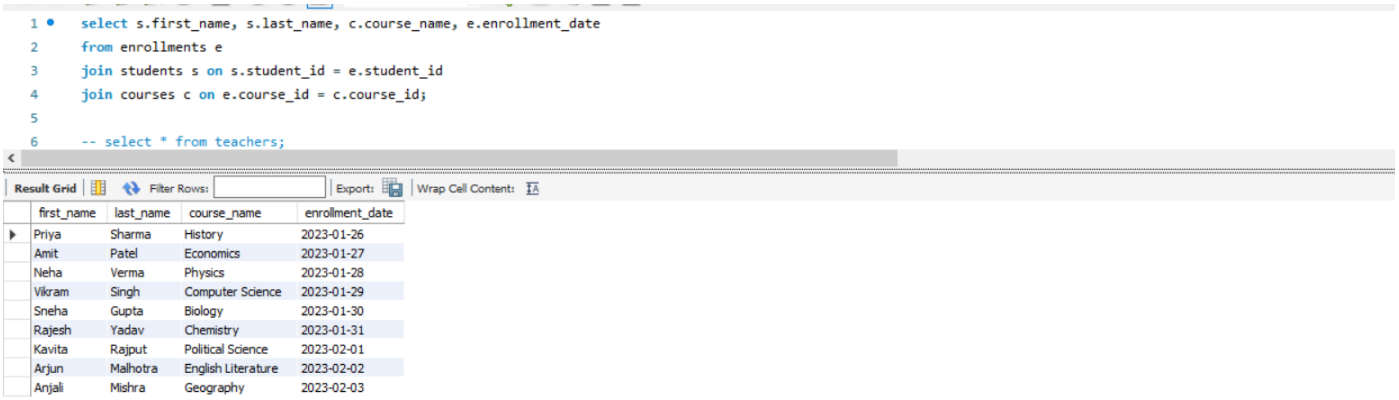
4. Write an SQL query to retrieve the first name, last name of students, and the names of the courses they are enrolled in. Use JOIN operations between the "Students" table and the "Enrollments" and "Courses" tables.



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.



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.



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.

```sql
select s.first_name, s.last_name, p.amount
from payments p
left join students s on s.student_id = p.student_id
where p.amount = 0 or p.amount is null;

-- update payments
-- set amount = 0
-- where student_id = 4;
```

first_name	last_name	amount
Neha	Verma	0.00

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.

```sql
select c.course_name
from courses c
left join enrollments e on e.course_id = c.course_id
where e.student_id is null;

-- select * from teachers;
-- select * from courses;
-- select * from students;
```

course_name
Mathematics

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.

```sql
INSERT INTO enrollments(student_id, course_id, enrollment_date)
VALUES
    (2, 15, current_date());

SELECT E1.student_id, S.first_name, S.last_name
FROM Enrollments E1
JOIN Enrollments E2 ON E1.student_id = E2.student_id AND E1.enrollment_id <> E2.enrollment_id
JOIN Students S ON E1.student_id = S.student_id
GROUP BY E1.student_id, S.first_name, S.last_name
HAVING COUNT(DISTINCT E1.course_id) > 1;
```

student_id	first_name	last_name
2	Priya	Sharma

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.

```sql
select t.teacher_id, t.first_name, t.last_name
from teachers t
left join courses c on c.teacher_id = t.teacher_id
where c.course_id is null;
```

teacher_id	first_name	last_name
6	Mrs. Anjali	Shukla
7	Dr. Rajat	Jain
8	Prof. Preeti	Mishra
9	Mr. Akash	Srivastava
10	Mrs. Meena	Gupta

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.

```sql
SELECT AVG(student_count) AS average_students_per_course
FROM (
    SELECT COUNT(DISTINCT student_id) AS student_count
    FROM Enrollments
    GROUP BY course_id
) AS course_student_counts;
```

average_students_per_course
1.1111

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.

```sql
select s.student_id, s.first_name, s.last_name, p.amount
from students s
join payments p on p.student_id = s.student_id
where p.amount = (
    select MAX(p2.amount)
    from payments p2
);
```

student_id	first_name	last_name	amount
3	Amit	Patel	6500.00

3. Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the course(s) with the maximum enrollment count.

```sql
select c.course_id, c.course_name, COUNT(distinct e.enrollment_id) as enrollment_count
from courses c
join enrollments e on c.course_id = e.course_id
group by e.course_id
having COUNT(distinct e.enrollment_id) = (
    select MAX(enr_counts)
    from (
        select COUNT(distinct e1.enrollment_id) as enr_counts
        from enrollments e1
        group by e1.course_id
    ) as maqx_enr_count
);
```

course_id	course_name	enrollment_count
15	Economics	3

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

```sql
SELECT
    T.teacher_id,
    T.first_name AS teacher_first_name,
    T.last_name AS teacher_last_name,
    (
        SELECT COALESCE(SUM(P.amount), 0)
        FROM Payments P
        JOIN Enrollments E ON P.student_id = E.student_id
        JOIN Courses C ON E.course_id = C.course_id
        WHERE C.teacher_id = T.teacher_id
    ) AS total_payments
FROM
    Teachers T;
```

teacher_id	teacher_first_name	teacher_last_name	total_payments
1	Dr. Manisha	Rathi	3500.00
2	Prof. Prakash	Khanna	10200.00
3	Mr. Rakesh	Chopra	9300.00
4	Dr. Nisha	Saxena	5200.00
5	Prof. Sanjay	Bhatia	21300.00
6	Mrs. Anjali	Shukla	0.00
7	Dr. Rajat	Jain	0.00
8	Prof. Preeti	Mishra	0.00
9	Mr. Akash	Srivastava	0.00
10	Mrs. Meena	Gupta	0.00

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

```
select s.student_id, s.first_name, s.last_name, count(distinct e.enrollment_id) as enrollment_count
FROM students s
join enrollments e on s.student_id = e.student_id
group by s.student_id
having count(distinct e.enrollment_id) = (
    select count(distinct c.course_id) as total_courses
    from courses c
);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	first_name	last_name	enrollment_count

6. Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to find teachers with no course assignments.

```
select t.teacher_id, t.first_name, t.last_name
from courses c
right join teachers t on t.teacher_id = c.teacher_id
group by t.teacher_id
having count(distinct c.course_name) = 0 or count(distinct c.course_name) = null
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

teacher_id	first_name	last_name
6	Mrs. Anjali	Shukla
7	Dr. Rajat	Jain
8	Prof. Preeti	Mishra
9	Mr. Akash	Srivastava
10	Mrs. Meena	Gupta

7. Calculate the average age of all students. Use subqueries to calculate the age of each student based on their date of birth.

```
select s.student_id, s.first_name, s.last_name, TIMESTAMPDIFF(YEAR, s.date_of_birth, CURRENT_DATE()) as age
from students s
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	first_name	last_name	age
2	Priya	Sharma	27
3	Amit	Patel	29
4	Neha	Verma	26
5	Vikram	Singh	30
6	Sneha	Gupta	25
7	Rajesh	Yadav	31
8	Kavita	Rajput	27
9	Arjun	Malhotra	32
10	Anjali	Mishra	25
11	John	Doe	28

8. Identify courses with no enrollments. Use subqueries to find courses without enrollment Records.

```
SELECT course_id, course_name
FROM Courses
WHERE course_id NOT IN (
    SELECT DISTINCT course_id
    FROM Enrollments
);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

course_id	course_name
11	Mathematics

9. Calculate the total payments made by each student for each course they are enrolled in. Use subqueries and aggregate functions to sum payments.

```sql
SELECT
    s.student_id,
    s.first_name,
    s.last_name,
    c.course_id,
    c.course_name,
    COALESCE((SELECT SUM(amount) FROM Payments p WHERE p.student_id = s.student_id), 0) AS total_payments
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	first_name	last_name	course_id	course_name	total_payments
2	Priya	Sharma	13	History	4500.00
2	Priya	Sharma	15	Economics	4500.00
3	Amit	Patel	15	Economics	6500.00
4	Neha	Verma	12	Physics	0.00
5	Vikram	Singh	14	Computer Science	6000.00
6	Sneha	Gupta	16	Biology	3500.00
7	Rajesh	Yadav	18	Chemistry	4800.00
8	Kavita	Rajput	19	Political Science	5200.00
9	Arjun	Malhotra	17	English Literature	4200.00
10	Anjali	Mishra	20	Geography	5800.00

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.

```sql
-- insert into payments(student_id, amount, payment_date)
-- values
--  (2, 5000, curdate())

SELECT student_id, first_name, last_name
FROM Students
WHERE student_id IN (
    SELECT student_id
    FROM Payments
    GROUP BY student_id
    HAVING COUNT(payment_id) > 1
);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

student_id	first_name	last_name
2	Priya	Sharma
NULL	NULL	NULL

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.

```sql
SELECT
    s.student_id,
    s.first_name,
    s.last_name,
    COALESCE((SELECT SUM(amount) FROM Payments p WHERE p.student_id = s.student_id), 0) AS total_payments
FROM Students s;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	first_name	last_name	total_payments
2	Priya	Sharma	14500.00
3	Amit	Patel	6500.00
4	Neha	Verma	0.00
5	Vikram	Singh	6000.00
6	Sneha	Gupta	3500.00
7	Rajesh	Yadav	4800.00
8	Kavita	Rajput	5200.00
9	Arjun	Malhotra	4200.00
10	Anjali	Mishra	5800.00
11	John	Doe	0.00

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.

```sql
SELECT
    c.course_id,
    c.course_name,
    COALESCE((SELECT COUNT(DISTINCT e.student_id) FROM Enrollments e WHERE e.course_id = c.course_id), 0) AS enrolled_students
FROM Courses c;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

course_id	course_name	enrolled_students
11	Mathematics	0
12	Physics	1
13	History	1
14	Computer Science	1
15	Economics	2
16	Biology	1
17	English Literature	1
18	Chemistry	1
19	Political Science	1
20	Geography	1

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.

```sql
SELECT
    s.student_id,
    s.first_name,
    s.last_name,
    COALESCE(AVG(p.amount), 0) AS average_payment_amount
FROM Students s
LEFT JOIN Payments p ON s.student_id = p.student_id
GROUP BY s.student_id, s.first_name, s.last_name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	first_name	last_name	average_payment_amount
2	Priya	Sharma	4833.333333
3	Amit	Patel	6500.000000
4	Neha	Verma	0.000000
5	Vikram	Singh	6000.000000
6	Sneha	Gupta	3500.000000
7	Rajesh	Yadav	4800.000000
8	Kavita	Rajput	5200.000000
9	Arjun	Malhotra	4200.000000
10	Anjali	Mishra	5800.000000
11	John	Doe	0.000000