Task 1. Database Design:

1. Create the database named "SISDB"

```
mysql> CREATE DATABASE SISDB;
Query OK, 1 row affected (0.10 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
b. Courses
c. Enrollments
d. Teacher
e. Payments

```
mysql> CREATE TABLE Students(student_id INT NOT NULL PRIMARY KEY,
    -> first_name VARCHAR(255),
    -> last_name VARCHAR(255),
    -> date_of_birth DATE,
    -> email VARCHAR(255),
    -> phone_number INT);
Query OK, 0 rows affected (0.21 sec)
```

```
mysql> CREATE TABLE Teacher(teacher_id INT NOT NULL PRIMARY KEY,
    -> first_name VARCHAR(255),
    -> last_name VARCHAR(255),
    -> email VARCHAR(255));
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE TABLE Courses(course_id INT NOT NULL PRIMARY KEY,
    -> course_name VARCHAR(255),
    -> credits VARCHAR(255),
    -> teacher_id INT,
    -> FOREIGN KEY (teacher_id) REFERENCES Teacher (teacher_id));
Query OK, 0 rows affected (0.09 sec)

mysql> CREATE TABLE Enrollments(enrollment_id INT NOT NULL PRIMARY KEY,
    -> student_id INT,
    -> FOREIGN KEY (student_id) REFERENCES Students (student_id),
    -> course_id INT,
    -> FOREIGN KEY (course_id) REFERENCES Courses (course_id),
    -> enrollment_date DATE);
Query OK, 0 rows affected (0.08 sec)

mysql> CREATE TABLE Payments(payment_id INT NOT NULL PRIMARY KEY,
    -> student_id INT,
    -> FOREIGN KEY (student_id) REFERENCES Students (student_id),
    -> amount VARCHAR(255),
    -> payment_date DATE);
Query OK, 0 rows affected (0.06 sec)
```

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

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

```
mysql> INSERT INTO Students(student_id, first_name, last_name, date_of_birth, email, phone_number)
    -> VALUES(1, "RICHA", "GUPTA", "2001-02-01", "rd@gmail.com", 34566),
    -> (2, "RIA", "GUPTA", "2001-03-01", "rid@gmail.com", 67566),
    -> (3, "PIA", "RATHA", "2000-04-02", "pia@gmail.com", 68996),
    -> (4, "KHUSI", "DAS", "2000-04-15", "khu@gmail.com", 23496),
    -> (5, "ANCHAL", "DASH", "2000-06-15", "ana@gmail.com", 25676),
    -> (6, "AHANA", "RAY", "2001-12-16", "aha@gmail.com", 78996),
    -> (7, "PINKY", "RAY", "2001-08-23", "pinky@gmail.com", 72396),
    -> (8, "CHINKY", "KUMARI", "2001-06-30", "ck@gmail.com", 67896),
    -> (9, "PREET", "KUMARI", "2001-06-26", "pr@gmail.com", 12396),
    -> (10, "PRITI", "DAS", "2001-10-17", "prdas@gmail.com", 56736);
```

```
mysql> INSERT INTO TEACHER(teacher_id, first_name, last_name, email)
    -> VALUES(01, "RINA", "PATEL", "rina@gmail.com"),
    -> (02, "ANA", "PAL", "ana@gmail.com"),
    -> (03, "ANYA", "LAL", "anyaa@gmail.com"),
    -> (04, "DONNA", "F", "donna@gmail.com"),
    -> (05, "JAY", "K", "jk@gmail.com"),
    -> (06, "CHINMAYEE", "KHANNA", "chink@gmail.com"),
    -> (07, "ENIE", "KHAN", "enk@gmail.com"),
    -> (08, "MAMATA", "RAJ", "mraj@gmail.com"),
    -> (09, "SHREYA", "RAJ", "sreayaraj@gmail.com"),
    -> (10, "RHEYA", "TAJ", "rtaj@gmail.com");
Query OK, 10 rows affected (0.22 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO COURSES(course_id, course_name, credits, teacher_id)
    -> VALUES(201, "BIOLOGY", "4", "01"),
    -> (202, "BOTANY", "3", "02"),
    -> (203, "MATHEMATICS", "4", "03"),
    -> (204, "MACHINE", "2", "04"),
    -> (205, "ML AND AI", "3", "05"),
    -> (206, "CLOUD", "3", "06"),
    -> (207, "LAB -1", "3", "07"),
    -> (208, "LAB -2", "3", "08"),
    -> (209, "ECONOMICS", "2", "09"),
    -> (210, "MECHANICS", "3", "10");
Query OK, 10 rows affected (0.19 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO ENROLLMENTS(enrollment_id, student_id, course_id, enrollment_date)
    -> VALUES(111, 1, 201, "2024-01-22"),
    -> (121, 2, 202, "2024-01-22"),
    -> (131, 3, 203, "2024-01-22"),
    -> (141, 4, 204, "2024-01-22"),
    -> (151, 5, 205, "2024-01-22"),
    -> (161, 6, 206, "2024-01-29"),
    -> (171, 7, 207, "2024-01-29"),
    -> (181, 8, 208, "2024-01-29"),
    -> (191, 9, 209, "2024-01-29"),
    -> (1101, 10, 210, "2024-01-29");
Query OK, 10 rows affected (0.34 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO PAYMENTS(payment_id, student_id, amount, payment_date)
    -> VALUES(501, 1, 2500, "2024-01-22"),
    -> (502, 2, 2200, "2024-01-22"),
    -> (503, 3, 3200, "2024-01-22"),
    -> (504, 4, 3500, "2024-01-22"),
    -> (505, 5, 3500, "2024-01-22"),
    -> (506, 6, 3500, "2024-01-29"),
    -> (507, 7, 3000, "2024-01-29"),
    -> (508, 8, 1500, "2024-01-29"),
    -> (509, 9, 2500, "2024-01-29"),
    -> (510, 10, 2500, "2024-01-29");
Query OK, 10 rows affected (0.20 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

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(student_id, first_name, last_name, date_of_birth, email, phone_number)
    -> VALUES(11, "John", "Doe", "1995-08-15", "john.doe@gmail.com", 1234567890);
Query OK, 1 row affected (0.04 sec)
```

```
mysql> select * from students;
+------------+------------+-----------+---------------+--------------------+--------------+
| student_id | first_name | last_name | date_of_birth | email              | phone_number |
+------------+------------+-----------+---------------+--------------------+--------------+
|          1 | RICHA      | GUPTA     | 2001-02-01    | rd@gmail.com       |        34566 |
|          2 | RIA        | GUPTA     | 2001-03-01    | rid@gmail.com      |        67566 |
|          3 | PIA        | RATHA     | 2000-04-02    | pia@gmail.com      |        68996 |
|          4 | KHUSI      | DAS       | 2000-04-15    | khu@gmail.com      |        23496 |
|          5 | ANCHAL     | DASH      | 2000-06-15    | ana@gmail.com      |        25676 |
|          6 | AHANA      | RAY       | 2001-12-16    | aha@gmail.com      |        78996 |
|          7 | PINKY      | RAY       | 2001-08-23    | pinky@gmail.com    |        72396 |
|          8 | CHINKY     | KUMARI    | 2001-06-30    | ck@gmail.com       |        67896 |
|          9 | PREET      | KUMARI    | 2001-06-26    | pr@gmail.com       |        12396 |
|         10 | PRITI      | DAS       | 2001-10-17    | prdas@gmail.com    |        56736 |
|         11 | John       | Doe       | 1995-08-15    | john.doe@gmail.com |   1234567890 |
+------------+------------+-----------+---------------+--------------------+--------------+
11 rows in set (0.04 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(enrollment_id, student_id, course_id, enrollment_date)
    -> VALUES(1201, 11, 207, "2024-01-19");
Query OK, 1 row affected (0.17 sec)

mysql> select * from enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           111 |          1 |       201 | 2024-01-22      |
|           121 |          2 |       202 | 2024-01-22      |
|           131 |          3 |       203 | 2024-01-22      |
|           141 |          4 |       204 | 2024-01-22      |
|           151 |          5 |       205 | 2024-01-22      |
|           161 |          6 |       206 | 2024-01-29      |
|           171 |          7 |       207 | 2024-01-29      |
|           181 |          8 |       208 | 2024-01-29      |
|           191 |          9 |       209 | 2024-01-29      |
|          1101 |         10 |       210 | 2024-01-29      |
|          1201 |         11 |       207 | 2024-01-19      |
+---------------+------------+-----------+-----------------+
11 rows in set (0.01 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 = "hi@gmail.com"
    -> WHERE teacher_id = 09;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from teacher;
+------------+------------+-----------+-----------------+
| teacher_id | first_name | last_name | email           |
+------------+------------+-----------+-----------------+
|          1 | RINA       | PATEL     | rina@gmail.com  |
|          2 | ANA        | PAL       | ana@gmail.com   |
|          3 | ANYA       | LAL       | anyaa@gmail.com |
|          4 | DONNA      | F         | donna@gmail.com |
|          5 | JAY        | K         | jk@gmail.com    |
|          6 | CHINMAYEE  | KHANNA    | chink@gmail.com |
|          7 | ENIE       | KHAN      | enk@gmail.com   |
|          8 | MAMATA     | RAJ       | mraj@gmail.com  |
|          9 | SHREYA     | RAJ       | hi@gmail.com    |
|         10 | RHEYA      | TAJ       | rtaj@gmail.com  |
+------------+------------+-----------+-----------------+
10 rows in set (0.01 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 = 9 AND course_id = 209;
Query OK, 1 row affected (0.12 sec)

mysql> select * from ENROLLMENTS;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           111 |          1 |       201 | 2024-01-22      |
|           121 |          2 |       202 | 2024-01-22      |
|           131 |          3 |       203 | 2024-01-22      |
|           141 |          4 |       204 | 2024-01-22      |
|           151 |          5 |       205 | 2024-01-22      |
|           161 |          6 |       206 | 2024-01-29      |
|           171 |          7 |       207 | 2024-01-29      |
|           181 |          8 |       208 | 2024-01-29      |
|          1101 |         10 |       210 | 2024-01-29      |
|          1201 |         11 |       207 | 2024-01-19      |
+---------------+------------+-----------+-----------------+
10 rows in set (0.02 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 = 07
    -> WHERE course_id =208;
Query OK, 1 row affected (0.11 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from COURSES;
+-----------+-------------+---------+------------+
| course_id | course_name | credits | teacher_id |
+-----------+-------------+---------+------------+
|       201 | BIOLOGY     | 4       |          1 |
|       202 | BOTANY      | 3       |          2 |
|       203 | MATHEMATICS | 4       |          3 |
|       204 | MACHINE     | 2       |          4 |
|       205 | ML AND AI   | 3       |          5 |
|       206 | CLOUD       | 3       |          6 |
|       207 | LAB -1      | 3       |          7 |
|       208 | LAB -2      | 3       |          7 |
|       209 | ECONOMICS   | 2       |          9 |
|       210 | MECHANICS   | 3       |         10 |
+-----------+-------------+---------+------------+
10 rows in set (0.02 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 = 8;
Query OK, 1 row affected (0.03 sec)
```

```
mysql> DELETE FROM payments
    -> WHERE student_id = 8;
Query OK, 1 row affected (0.05 sec)

mysql> DELETE FROM STUDENTS
    -> where student_id = 8;
Query OK, 1 row affected (0.01 sec)

mysql> select * from students;
+------------+------------+-----------+---------------+--------------------+--------------+
| student_id | first_name | last_name | date_of_birth | email              | phone_number |
+------------+------------+-----------+---------------+--------------------+--------------+
|          1 | RICHA      | GUPTA     | 2001-02-01    | rd@gmail.com       |        34566 |
|          2 | RIA        | GUPTA     | 2001-03-01    | rid@gmail.com      |        67566 |
|          3 | PIA        | RATHA     | 2000-04-02    | pia@gmail.com      |        68996 |
|          4 | KHUSI      | DAS       | 2000-04-15    | khu@gmail.com      |        23496 |
|          5 | ANCHAL     | DASH      | 2000-06-15    | ana@gmail.com      |        25676 |
|          6 | AHANA      | RAY       | 2001-12-16    | aha@gmail.com      |        78996 |
|          7 | PINKY      | RAY       | 2001-08-23    | pinky@gmail.com    |        72396 |
|          9 | PREET      | KUMARI    | 2001-06-26    | pr@gmail.com       |        12396 |
|         10 | PRITI      | DAS       | 2001-10-17    | prdas@gmail.com    |        56736 |
|         11 | John       | Doe       | 1995-08-15    | john.doe@gmail.com |   1234567890 |
+------------+------------+-----------+---------------+--------------------+--------------+
```

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

```
mysql> select * from payments;
+------------+------------+--------+--------------+
| payment_id | student_id | amount | payment_date |
+------------+------------+--------+--------------+
|        501 |          1 | 2500   | 2024-01-22   |
|        502 |          2 | 2200   | 2024-01-22   |
|        503 |          3 | 3200   | 2024-01-22   |
|        504 |          4 | 3500   | 2024-01-22   |
|        505 |          5 | 3500   | 2024-01-22   |
|        506 |          6 | 3500   | 2024-01-29   |
|        507 |          7 | 3000   | 2024-01-29   |
|        509 |          9 | 2500   | 2024-01-29   |
|        510 |         10 | 2500   | 2024-01-29   |
+------------+------------+--------+--------------+
9 rows in set (0.00 sec)

mysql> UPDATE payments
    -> SET amount = 4000
    -> where payment_id = 503;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from payments;
+------------+------------+--------+--------------+
| payment_id | student_id | amount | payment_date |
+------------+------------+--------+--------------+
|        501 |          1 | 2500   | 2024-01-22   |
|        502 |          2 | 2200   | 2024-01-22   |
|        503 |          3 | 4000   | 2024-01-22   |
|        504 |          4 | 3500   | 2024-01-22   |
|        505 |          5 | 3500   | 2024-01-22   |
|        506 |          6 | 3500   | 2024-01-29   |
|        507 |          7 | 3000   | 2024-01-29   |
|        509 |          9 | 2500   | 2024-01-29   |
|        510 |         10 | 2500   | 2024-01-29   |
+------------+------------+--------+--------------+
9 rows in set (0.01 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.student_id,
    -> Students.first_name,
    -> SUM(Payments.amount) AS total_payments
    -> FROM Students
    -> JOIN Payments ON Students.student_id = Payments.student_id
    -> WHERE Students.student_id = 4
    -> GROUP BY Students.student_id, Students.first_name;
+------------+------------+----------------+
| student_id | first_name | total_payments |
+------------+------------+----------------+
|          4 | KHUSI      |           6000 |
+------------+------------+----------------+
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
    -> JOIN Enrollments ON Courses.course_id = Enrollments.course_id
    -> GROUP BY Courses.course_id, Courses.course_name;
+-----------+-------------+------------------+
| course_id | course_name | enrolled_students |
+-----------+-------------+------------------+
|       201 | BIOLOGY     |                1 |
|       202 | BOTANY      |                1 |
|       203 | MATHEMATICS |                1 |
|       204 | MACHINE     |                1 |
|       205 | ML AND AI   |                1 |
|       206 | CLOUD       |                1 |
|       207 | LAB -1      |                2 |
|       210 | MECHANICS   |                1 |
+-----------+-------------+------------------+
8 rows in set (0.08 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.

```
mysql> SELECT Students.student_id,
    -> Students.first_name
    -> FROM Students
    -> LEFT JOIN Enrollments ON Students.Student_id = Enrollments.student_id
    -> WHERE Enrollments.student_id is NULL;
+------------+------------+
| student_id | first_name |
+------------+------------+
|          9 | PREET      |
|         11 | John       |
+------------+------------+
2 rows in set (0.01 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.

```
mysql> SELECT Students.first_name, Students.last_name,
    -> Courses.course_id
    -> FROM Students
    -> JOIN Enrollments ON Students.student_id = Enrollments.student_id
    -> JOIN Courses ON Enrollments.course_id = Courses.course_id;
+------------+-----------+-----------+
| first_name | last_name | course_id |
+------------+-----------+-----------+
| RICHA      | GUPTA     |       201 |
| RIA        | GUPTA     |       202 |
| PIA        | RATHA     |       203 |
| KHUSI      | DAS       |       204 |
| ANCHAL     | DASH      |       205 |
| AHANA      | RAY       |       206 |
| PINKY      | RAY       |       207 |
| PRITI      | DAS       |       210 |
| PRITI      | DAS       |       207 |
+------------+-----------+-----------+
9 rows in set (0.00 sec)
```

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.

```
mysql> SELECT Teacher.teacher_id,
    -> CONCAT(Teacher.first_name, ' ', Teacher.last_name) AS teacher_name,
    -> Courses.course_name
    -> FROM Teacher
    -> JOIN Courses ON Teacher.teacher_id = Courses.teacher_id;
+------------+------------------+-------------+
| teacher_id | teacher_name     | course_name |
+------------+------------------+-------------+
|          1 | RINA PATEL       | BIOLOGY     |
|          2 | ANA PAL          | BOTANY      |
|          3 | ANYA LAL         | MATHEMATICS |
|          4 | DONNA F          | MACHINE     |
|          5 | JAY K            | ML AND AI   |
|          6 | CHINMAYEE KHANNA | CLOUD       |
|          7 | ENIE KHAN        | LAB -1      |
|          7 | ENIE KHAN        | LAB -2      |
|          9 | SHREYA RAJ       | ECONOMICS   |
|         10 | RHEYA TAJ        | MECHANICS   |
+------------+------------------+-------------+
10 rows in set (0.04 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.

```
mysql> SELECT Students.student_id,
    -> Students.first_name,
    ->     Students.last_name,
    ->     Enrollments.enrollment_date
    -> FROM Students
    -> JOIN Enrollments ON Students.student_id = Enrollments.student_id
    -> JOIN Courses ON Enrollments.course_id = Courses.course_id
    -> WHERE Courses.course_name = 'cloud';
+------------+------------+-----------+-----------------+
| student_id | first_name | last_name | enrollment_date |
+------------+------------+-----------+-----------------+
|          6 | AHANA      | RAY       | 2024-01-29      |
+------------+------------+-----------+-----------------+
1 row in set (0.01 sec)
```

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.

```
mysql> SELECT Students.student_id,
    -> Students.first_name,
    -> Students.last_name
    -> FROM Students
    -> LEFT JOIN Payments On Students.student_id = Payments.student_id
    -> WHERE Payments.student_id IS NULL;
+------------+------------+-----------+
| student_id | first_name | last_name |
+------------+------------+-----------+
|         10 | PRITI      | DAS       |
|         11 | John       | Doe       |
+------------+------------+-----------+
2 rows in set (0.02 sec)
```

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.

```
mysql> SELECT courses.course_id, courses.course_name
    -> FROM courses
    -> LEFT JOIN Enrollments ON Courses.course_id = Enrollments.course_id
    -> WHERE Enrollments.course_id IS NULL;
+♦----------+-------------+
| course_id | course_name |
+-----------+-------------+
|       208 | LAB -2      |
|       209 | ECONOMICS   |
+-----------+-------------+
2 rows in set (0.01 sec)
```

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

```
mysql> SELECT
    ->     e1.student_id,
    ->     s.first_name,
    ->     s.last_name
    -> FROM
    ->     Enrollments e1
    -> JOIN
    ->     Students s ON e1.student_id = s.student_id
    -> WHERE
    ->     e1.student_id IN (
    ->         SELECT
    ->             student_id
    ->         FROM
    ->             Enrollments
    ->         GROUP BY
    ->             student_id
    ->         HAVING
    ->             COUNT(course_id) > 1
    ->     );
+------------+------------+-----------+
| student_id | first_name | last_name |
+------------+------------+-----------+
|         10 | PRITI      | DAS       |
|         10 | PRITI      | DAS       |
+------------+------------+-----------+
2 rows in set (0.01 sec)
```

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.

```
mysql> SELECT
    -> Teacher.teacher_id,
    -> Teacher.first_name, Teacher.last_name
    -> FROM Teacher
    -> LEFT JOIN Courses ON Teacher.teacher_id = Courses.teacher_id
    -> WHERE Courses.teacher_id IS NULL;
+------------+------------+-----------+
| teacher_id | first_name | last_name |
+------------+------------+-----------+
|          8 | MAMATA     | RAJ       |
+------------+------------+-----------+
1 row in set (0.02 sec)
```

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.

```
mysql> SELECT c.course_id,
    -> c.course_name,
    -> AVG(e.student_count) AS average_students_enrolled
    -> FROM courses c
    -> JOIN ( SELECT course_id,
    -> COUNT(DISTINCT student_id) AS student_count
    -> FROM enrollments
    -> GROUP BY course_id)
    -> e ON c.course_id = e.course_id
    -> GROUP BY c.course_id, c.course_name;
+-----------+-------------+---------------------------+
| course_id | course_name | average_students_enrolled |
+-----------+-------------+---------------------------+
|       201 | BIOLOGY     |                    1.0000 |
|       202 | BOTANY      |                    1.0000 |
|       203 | MATHEMATICS |                    1.0000 |
|       204 | MACHINE     |                    1.0000 |
|       205 | ML AND AI   |                    1.0000 |
|       206 | CLOUD       |                    1.0000 |
|       207 | LAB -1      |                    2.0000 |
|       210 | MECHANICS   |                    1.0000 |
+-----------+-------------+---------------------------+
8 rows in set (0.07 sec)
```

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.

```
mysql> SELECT student_id, payment_amount
    -> FROM payments
    -> WHERE payment_amount = (SELECT MAX(payment_amount) FROM payments);
+------------+----------------+
| student_id | payment_amount |
+------------+----------------+
|          3 | 4000           |
+------------+----------------+
1 row in set (0.02 sec)
```

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

```
mysql> SELECT c.course_id, c.course_name, COUNT(e.student_id) AS enrollment_
count
    -> FROM Courses c
    -> JOIN Enrollments e ON c.course_id = e.course_id
    -> WHERE c.course_id IN (
    ->     SELECT e.course_id
    ->     FROM Enrollments e
    ->     GROUP BY e.course_id
    ->     HAVING COUNT(e.student_id) = (
    ->         SELECT MAX(enrollment_count)
    ->         FROM (
    ->             SELECT course_id, COUNT(student_id) AS enrollment_count
    ->             FROM Enrollments
    ->             GROUP BY course_id
    ->         ) AS max_enrollments
    ->     )
    -> )
    -> GROUP BY c.course_id, c.course_name;
+-----------+-------------+------------------+
| course_id | course_name | enrollment_count |
+-----------+-------------+------------------+
|       207 | LAB -1      |                2 |
+-----------+-------------+------------------+
1 row in set (0.01 sec)
```

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

```
mysql> SELECT
    ->     t.teacher_id,
    ->     t.first_name || ' ' || t.last_name AS teacher_name,
    ->     COALESCE(SUM(p.PAYMENT_amount), 0) AS total_payments
    -> FROM
    ->     Teacher t
    -> LEFT JOIN
    ->     Courses c ON t.teacher_id = c.teacher_id
    -> LEFT JOIN
    ->     Enrollments e ON c.course_id = e.course_id
    -> LEFT JOIN
    ->     Payments p ON e.student_id = p.student_id
    -> GROUP BY
    ->     t.teacher_id, t.first_name, t.last_name;
+------------+--------------+----------------+
| teacher_id | teacher_name | total_payments |
+------------+--------------+----------------+
|          1 |            0 |           2500 |
|          2 |            0 |           2200 |
|          3 |            0 |           4000 |
|          4 |            0 |           6000 |
|          5 |            0 |           3500 |
|          6 |            0 |           3500 |
|          7 |            0 |           3000 |
|          8 |            0 |              0 |
|          9 |            0 |              0 |
|         10 |            0 |              0 |
+------------+--------------+----------------+
10 rows in set, 18 warnings (0.04 sec)
```

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

```
mysql> SELECT
    ->     student_id,
    ->     first_name,
    ->     last_name
    -> FROM
    ->     students s
    -> WHERE
    ->     (
    ->         SELECT COUNT(DISTINCT course_id)
    ->         FROM enrollments e
    ->         ) = (
    ->         SELECT COUNT(DISTINCT course_id)
    ->         FROM enrollments e2
    ->         WHERE e2.student_id = s.student_id
    ->     );
Empty set (0.13 sec)
```

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

```
mysql> SELECT
    ->     teacher_id,
    ->     first_name,
    ->     last_name
    -> FROM
    ->     teacher t
    -> WHERE
    ->     NOT EXISTS (
    ->         SELECT 1
    ->         FROM courses c
    ->         WHERE c.teacher_id = t.teacher_id
    ->     );
+------------+------------+-----------+
| teacher_id | first_name | last_name |
+------------+------------+-----------+
|          8 | MAMATA     | RAJ       |
+------------+------------+-----------+
1 row in set (0.06 sec)
```

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

```
mysql> SELECT
    -> AVG(YEAR(CURDATE()) - YEAR(date_of_birth) - (RIGHT(CURDATE(), 5) < RIGHT(date_of_birth, 5))) AS average_age
    -> FROM
    -> students;
+-------------+
| average_age |
+-------------+
|     22.9000 |
+-------------+
1 row in set (0.02 sec)
```

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

```
mysql> SELECT
    ->     course_id,
    ->     course_name
    -> FROM
    ->     courses c
    -> WHERE
    ->     NOT EXISTS (
    ->         SELECT 1
    ->         FROM enrollments e
    ->         WHERE e.course_id = c.course_id
    ->     );
+-----------+-------------+
| course_id | course_name |
+-----------+-------------+
|       208 | LAB -2      |
|       209 | ECONOMICS   |
+-----------+-------------+
2 rows in set (0.01 sec)
```

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

```
mysql> SELECT
    ->     s.student_id,
    ->     s.first_name,
    ->     s.last_name,
    ->     c.course_id,
    ->     c.course_name,
    -> IFNULL(SUM(p.payment_amount), 0) AS total_amount
    -> FROM
    ->     students s
    -> JOIN
    ->     enrollments e ON s.student_id = e.student_id
    -> JOIN
    ->     courses c ON e.course_id = c.course_id
    -> LEFT JOIN
    ->     payments p ON e.student_id = p.student_id
    -> GROUP BY
    ->     s.student_id, s.first_name, s.last_name, c.course_id, c.course_name;
+------------+------------+-----------+-----------+-------------+--------------+
| student_id | first_name | last_name | course_id | course_name | total_amount |
+------------+------------+-----------+-----------+-------------+--------------+
|          1 | RICHA      | GUPTA     |       201 | BIOLOGY     |         2500 |
|          2 | RIA        | GUPTA     |       202 | BOTANY      |         2200 |
|          3 | PIA        | RATHA     |       203 | MATHEMATICS |         4000 |
|          4 | KHUSI      | DAS       |       204 | MACHINE     |         6000 |
|          5 | ANCHAL     | DASH      |       205 | ML AND AI   |         3500 |
|          6 | AHANA      | RAY       |       206 | CLOUD       |         3500 |
|          7 | PINKY      | RAY       |       207 | LAB -1      |         3000 |
|         10 | PRITI      | DAS       |       210 | MECHANICS   |            0 |
|         10 | PRITI      | DAS       |       207 | LAB -1      |            0 |
+------------+------------+-----------+-----------+-------------+--------------+
9 rows in set (0.01 sec)
```

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.

```
mysql> SELECT
    ->     student_id,
    ->     first_name,
    ->     last_name
    -> FROM
    ->     students
    -> WHERE
    ->     student_id IN (
    ->         SELECT
    ->             student_id
    ->         FROM
    ->             payments
    ->         GROUP BY
    ->             student_id
    ->         HAVING
    ->             COUNT(*) > 1
    ->     );
+------------+------------+-----------+
| student_id | first_name | last_name |
+------------+------------+-----------+
|          4 | KHUSI      | DAS       |
+------------+------------+-----------+
1 row in set (0.01 sec)
```

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.

```
mysql> SELECT
    ->     s.student_id,
    ->     s.first_name,
    ->     s.last_name,
    -> IFNULL(SUM(p.payment_amount), 0) AS total_payments
    -> 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;
+------------+------------+-----------+----------------+
| student_id | first_name | last_name | total_payments |
+------------+------------+-----------+----------------+
|          1 | RICHA      | GUPTA     |           2500 |
|          2 | RIA        | GUPTA     |           2200 |
|          3 | PIA        | RATHA     |           4000 |
|          4 | KHUSI      | DAS       |           6000 |
|          5 | ANCHAL     | DASH      |           3500 |
|          6 | AHANA      | RAY       |           3500 |
|          7 | PINKY      | RAY       |           3000 |
|          9 | PREET      | KUMARI    |           2500 |
|         10 | PRITI      | DAS       |              0 |
|         11 | John       | Doe       |              0 |
+------------+------------+-----------+----------------+
10 rows in set (0.00 sec)
```

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.

```
mysql> SELECT c.course_id, c.course_name,
    -> COUNT(e.student_id) AS student_count
    -> FROM Courses c
    -> LEFT JOIN
    ->     Enrollments e ON c.course_id = e.course_id
    -> GROUP BY
    ->     c.course_id, c.course_name;
+-----------+-------------+---------------+
| course_id | course_name | student_count |
+-----------+-------------+---------------+
|       201 | BIOLOGY     |             1 |
|       202 | BOTANY      |             1 |
|       203 | MATHEMATICS |             1 |
|       204 | MACHINE     |             1 |
|       205 | ML AND AI   |             1 |
|       206 | CLOUD       |             1 |
|       207 | LAB -1      |             2 |
|       208 | LAB -2      |             0 |
|       209 | ECONOMICS   |             0 |
|       210 | MECHANICS   |             1 |
+-----------+-------------+---------------+
10 rows in set (0.01 sec)
```

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.

```
mysql> SELECT
    ->     s.student_id,
    ->     s.first_name,
    ->     s.last_name,
    ->     AVG(p.payment_amount) AS average_payment
    -> 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;
+------------+------------+-----------+-----------------+
| student_id | first_name | last_name | average_payment |
+------------+------------+-----------+-----------------+
|          1 | RICHA      | GUPTA     |            2500 |
|          2 | RIA        | GUPTA     |            2200 |
|          3 | PIA        | RATHA     |            4000 |
|          4 | KHUSI      | DAS       |            3000 |
|          5 | ANCHAL     | DASH      |            3500 |
|          6 | AHANA      | RAY       |            3500 |
|          7 | PINKY      | RAY       |            3000 |
|          9 | PREET      | KUMARI    |            2500 |
|         10 | PRITI      | DAS       |            NULL |
|         11 | John       | Doe       |            NULL |
+------------+------------+-----------+-----------------+
10 rows in set (0.00 sec)
```