ASSIGNMENT-2

NAME: SHIVAM SINGH

TASK-1:

Database Design:

1. Create the database named "SISDB"

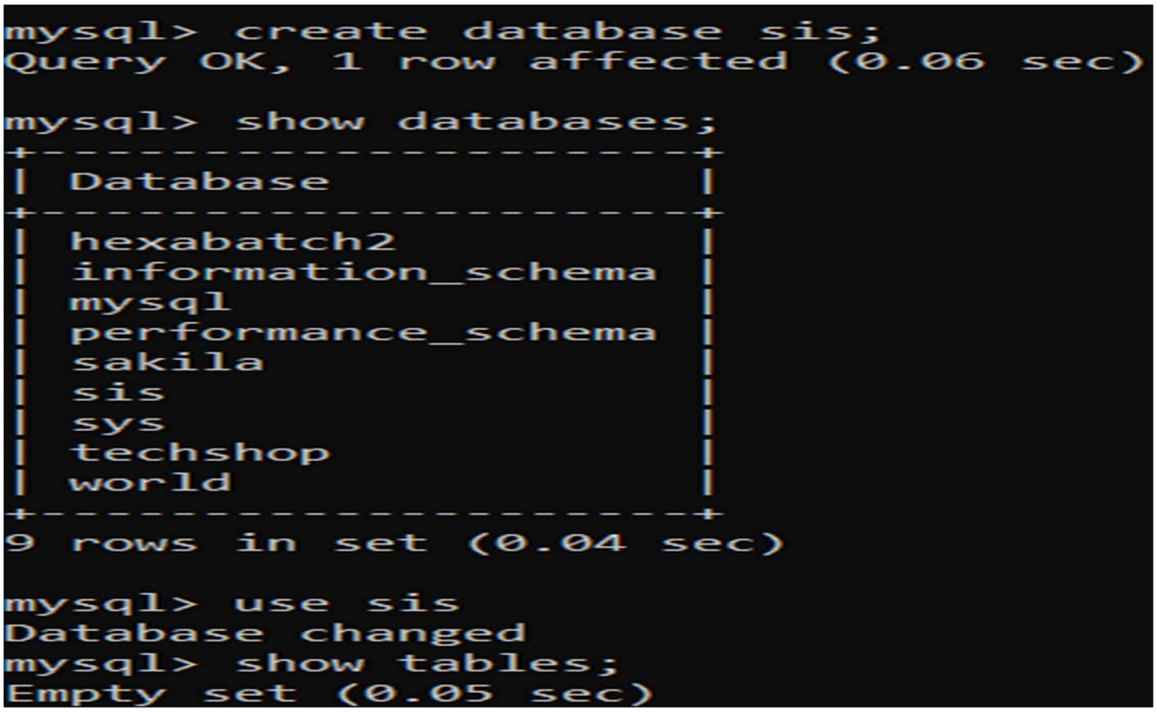
2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships. a. Students b. Courses c. Enrollments d. Teacher e. Payments

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

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

5. Insert at least 10 sample records into each of the following tables. i. Students ii. Courses iii. Enrollments iv. Teacher v. Payments

1.

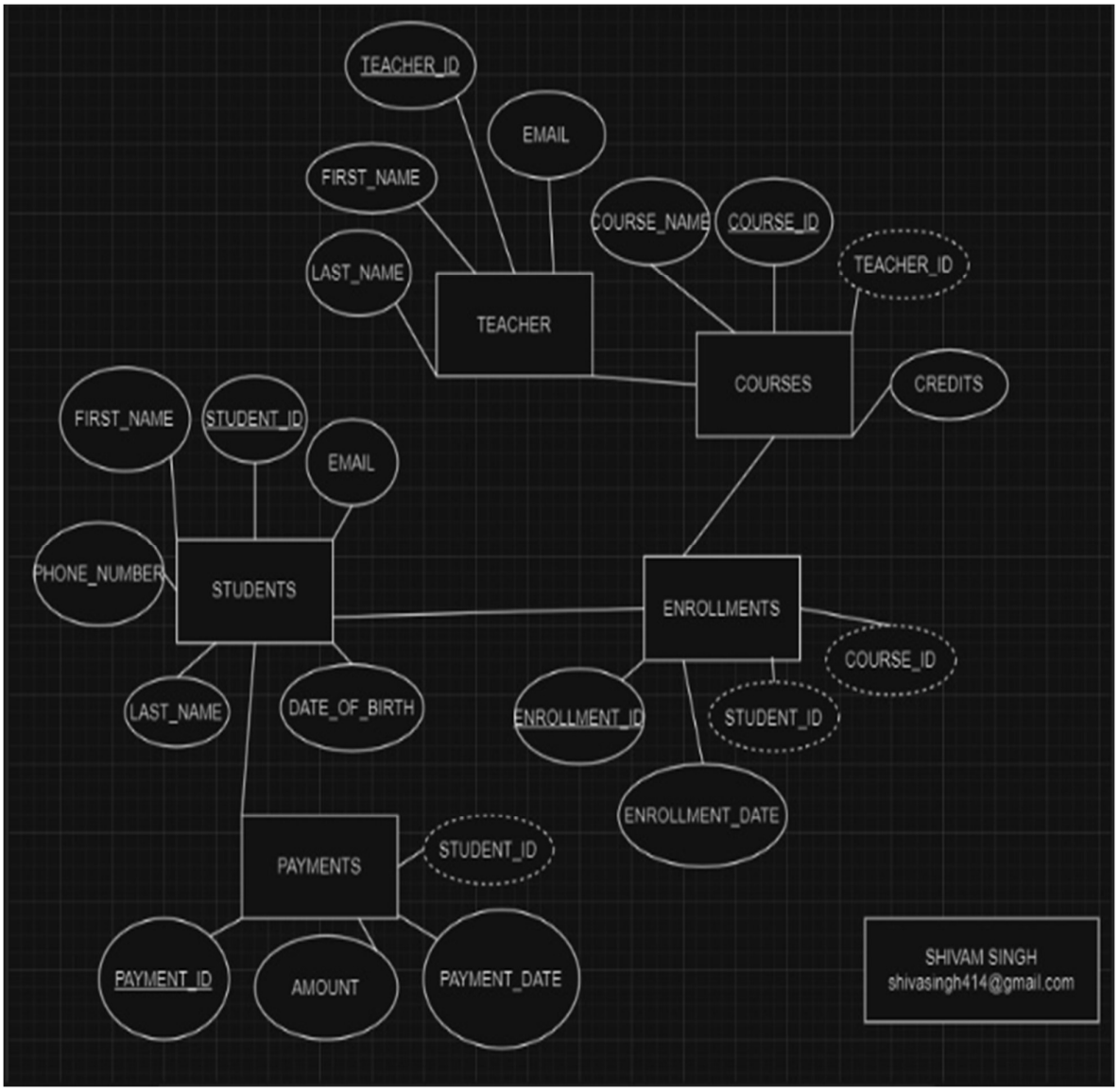


2.

```
-- Create Students Table
CREATE TABLE Students (
    student_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    date_of_birth DATE,
    email VARCHAR(100),
    phone_number VARCHAR(20)
);
-- Create Teacher Table
CREATE TABLE Teacher (
    teacher_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    phone_number VARCHAR(20)
);
-- Create Courses Table
CREATE TABLE Courses (
    course_id INT PRIMARY KEY,
    course_name VARCHAR(100),
    credits INT,
    teacher_id INT,
    FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
);
-- Create Enrollments Table
CREATE TABLE Enrollments (
    enrollment_id INT PRIMARY KEY,
    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)
);
-- Create Payments Table
CREATE TABLE Payments (
    payment_id INT PRIMARY KEY,
    student_id INT,
    amount DECIMAL(10, 2),
    payment_date DATE,
    FOREIGN KEY (student_id) REFERENCES Students(student_id)
);
```

3.



5.

Students:

```
mysql> select*from students;
+------------+------------+-----------+---------------+-------------------------+--------------+
| student_id | first_name | last_name | date_of_birth | email                   | phone_number |
+------------+------------+-----------+---------------+-------------------------+--------------+
|         10 | Shivam     | Singh     | 2000-09-24    | shivasingh414@gmail.com | 8340508631   |
|         11 | John       | Doe       | 1995-08-15    | john.doe@example.com    | 1234567890   |
|         12 | Alice      | Smith     | 1998-03-10    | alice.smith@example.com | 9876543210   |
|         13 | Bob        | Johnson   | 2002-05-05    | bob.johnson@example.com | 5678901234   |
|         14 | Emma       | Davis     | 1997-12-22    | emma.davis@example.com  | 3456789012   |
|         15 | Chris      | Brown     | 1999-11-08    | chris.brown@example.com | 8765432109   |
|         16 | Olivia     | Moore     | 2001-04-18    | olivia.moore@example.com | 6543210987   |
|         17 | Daniel     | Miller    | 1996-06-30    | daniel.miller@example.com | 2345678901   |
|         18 | Sophia     | Jones     | 2003-09-12    | sophia.jones@example.com | 8901234567   |
|         19 | William    | White     | 1994-01-07    | william.white@example.com | 4321098765   |
+------------+------------+-----------+---------------+-------------------------+--------------+
10 rows in set (0.00 sec)
```

Teacher:

```
mysql> INSERT INTO Teacher (teacher_id, first_name, last_name, email)
    -> VALUES
    ->     (1, 'Jane', 'Smith', 'jane.smith@example.com'),
    ->     (2, 'Michael', 'Johnson', 'michael.johnson@example.com'),
    ->     (3, 'Emily', 'Brown', 'emily.brown@example.com'),
    ->     (4, 'Andrew', 'Taylor', 'andrew.taylor@example.com'),
    ->     (5, 'Madison', 'Williams', 'madison.williams@example.com'),
    ->     (6, 'Aiden', 'Davis', 'aiden.davis@example.com'),
    ->     (7, 'Abigail', 'Moore', 'abigail.moore@example.com'),
    ->     (8, 'Ethan', 'Miller', 'ethan.miller@example.com'),
    ->     (9, 'Chloe', 'Jones', 'chloe.jones@example.com'),
    ->     (10, 'Mason', 'White', 'mason.white@example.com');
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select*from teacher;
+------------+------------+-----------+-------------------------------+
| teacher_id | first_name | last_name | email                         |
+------------+------------+-----------+-------------------------------+
|          1 | Jane       | Smith     | jane.smith@example.com        |
|          2 | Michael    | Johnson   | michael.johnson@example.com   |
|          3 | Emily      | Brown     | emily.brown@example.com       |
|          4 | Andrew     | Taylor    | andrew.taylor@example.com     |
|          5 | Madison    | Williams  | madison.williams@example.com  |
|          6 | Aiden      | Davis     | aiden.davis@example.com       |
|          7 | Abigail    | Moore     | abigail.moore@example.com     |
|          8 | Ethan      | Miller    | ethan.miller@example.com      |
|          9 | Chloe      | Jones     | chloe.jones@example.com       |
|         10 | Mason      | White     | mason.white@example.com       |
+------------+------------+-----------+-------------------------------+
10 rows in set (0.00 sec)
```

Payments:

```
mysql> select*from payments;
+------------+------------+--------+--------------+
| payment_id | student_id | amount | payment_date |
+------------+------------+--------+--------------+
|          1 |         10 | 100.00 | 2024-01-19   |
|          2 |         11 | 150.00 | 2024-01-20   |
|          3 |         12 | 120.00 | 2024-01-21   |
|          4 |         13 | 130.00 | 2024-01-22   |
|          5 |         14 | 110.00 | 2024-01-23   |
|          6 |         15 |  90.00 | 2024-01-24   |
|          7 |         16 |  80.00 | 2024-01-25   |
|          8 |         17 | 200.00 | 2024-01-26   |
|          9 |         18 | 170.00 | 2024-01-27   |
|         10 |         19 | 140.00 | 2024-01-28   |
+------------+------------+--------+--------------+
10 rows in set (0.00 sec)
```

Courses:

```
mysql> select*from courses;
+-----------+--------------------+---------+------------+
| course_id | course_name        | credits | teacher_id |
+-----------+--------------------+---------+------------+
|         1 | Mathematics        |       3 |          1 |
|         2 | English Literature |       4 |          2 |
|         3 | Computer Science   |       3 |          3 |
|         4 | Physics            |       4 |          4 |
|         5 | History            |       3 |          5 |
|         6 | Chemistry          |       4 |          6 |
|         7 | Biology            |       3 |          7 |
|         8 | Economics          |       3 |          8 |
|         9 | Psychology         |       4 |          9 |
|        10 | Art History        |       3 |         10 |
+-----------+--------------------+---------+------------+
10 rows in set (0.00 sec)
```

Enrollments:

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

TASK: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

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.

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

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.

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

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.

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

1.

```
mysql> select*from students;
+------------+------------+-----------+---------------+---------------------------+--------------+
| student_id | first_name | last_name | date_of_birth | email                     | phone_number |
+------------+------------+-----------+---------------+---------------------------+--------------+
|         10 | Shivam     | Singh     | 2000-09-24    | shivasingh414@gmail.com   | 8340508631   |
|         11 | John       | Doe       | 1995-08-15    | john.doe@example.com      | 1234567890   |
|         12 | Alice      | Smith     | 1998-03-10    | alice.smith@example.com   | 9876543210   |
|         13 | Bob        | Johnson   | 2002-05-05    | bob.johnson@example.com   | 5678901234   |
|         14 | Emma       | Davis     | 1997-12-22    | emma.davis@example.com    | 3456789012   |
|         15 | Chris      | Brown     | 1999-11-08    | chris.brown@example.com   | 8765432109   |
|         16 | Olivia     | Moore     | 2001-04-18    | olivia.moore@example.com  | 6543210987   |
|         17 | Daniel     | Miller    | 1996-06-30    | daniel.miller@example.com | 2345678901   |
|         18 | Sophia     | Jones     | 2003-09-12    | sophia.jones@example.com  | 8901234567   |
|         19 | William    | White     | 1994-01-07    | william.white@example.com | 4321098765   |
+------------+------------+-----------+---------------+---------------------------+--------------+
```

2.

```
    -> VALUES (11, 3, CURRENT_DATE);
ERROR 1364 (HY000): Field 'enrollment_id' doesn't have a default value
mysql> INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
    -> VALUES (11, 11, 3, CURRENT_DATE);
Query OK, 1 row affected (0.00 sec)

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

3.

```
mysql> UPDATE Teacher
    -> SET email = 'new_email@example.com'
    -> WHERE teacher_id = 2;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select*from teacher;
+------------+------------+-----------+-------------------------------+
| teacher_id | first_name | last_name | email                         |
+------------+------------+-----------+-------------------------------+
|          1 | Jane       | Smith     | jane.smith@example.com        |
|          2 | Michael    | Johnson   | new_email@example.com         |
|          3 | Emily      | Brown     | emily.brown@example.com       |
|          4 | Andrew     | Taylor    | andrew.taylor@example.com     |
|          5 | Madison    | Williams  | madison.williams@example.com  |
|          6 | Aiden      | Davis     | aiden.davis@example.com       |
|          7 | Abigail    | Moore     | abigail.moore@example.com     |
|          8 | Ethan      | Miller    | ethan.miller@example.com      |
|          9 | Chloe      | Jones     | chloe.jones@example.com       |
|         10 | Mason      | White     | mason.white@example.com       |
+------------+------------+-----------+-------------------------------+
10 rows in set (0.00 sec)
```

4.

```
mysql> DELETE FROM Enrollments
    -> WHERE student_id = 11 AND course_id = 3;
Query OK, 1 row affected (0.00 sec)

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

5.

```
mysql> UPDATE Courses
    -> SET teacher_id = 2
    -> WHERE course_id = 3;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select*from courses;
+-----------+--------------------+---------+------------+
| course_id | course_name        | credits | teacher_id |
+-----------+--------------------+---------+------------+
|         1 | Mathematics        |       3 |          1 |
|         2 | English Literature |       4 |          2 |
|         3 | Computer Science   |       3 |          2 |
|         4 | Physics            |       4 |          4 |
|         5 | History            |       3 |          5 |
|         6 | Chemistry          |       4 |          6 |
|         7 | Biology            |       3 |          7 |
|         8 | Economics          |       3 |          8 |
|         9 | Psychology         |       4 |          9 |
|        10 | Art History        |       3 |         10 |
+-----------+--------------------+---------+------------+
10 rows in set (0.00 sec)
```

6.

```
mysql> DELETE FROM Enrollments
    -> WHERE student_id = 12;
Query OK, 1 row affected (0.00 sec)

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

7.

```
mysql> UPDATE Payments
    -> SET amount = 150.00  -- Replace with the new payment amount
    -> WHERE payment_id = 3;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select*from payments;
+------------+------------+--------+--------------+
| payment_id | student_id | amount | payment_date |
+------------+------------+--------+--------------+
|          1 |         10 | 100.00 | 2024-01-19   |
|          2 |         11 | 150.00 | 2024-01-20   |
|          3 |         12 | 150.00 | 2024-01-21   |
|          4 |         13 | 130.00 | 2024-01-22   |
|          5 |         14 | 110.00 | 2024-01-23   |
|          6 |         15 |  90.00 | 2024-01-24   |
|          7 |         16 |  80.00 | 2024-01-25   |
|          8 |         17 | 200.00 | 2024-01-26   |
|          9 |         18 | 170.00 | 2024-01-27   |
|         10 |         19 | 140.00 | 2024-01-28   |
+------------+------------+--------+--------------+
10 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.

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.

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.

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.

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.

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.

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.

1.

```
mysql> SELECT
    ->     s.student_id,
    ->     s.first_name,
    ->     s.last_name,
    ->     SUM(p.amount) AS total_payments
    -> FROM
    ->     Students s
    -> JOIN
    ->     Payments p ON s.student_id = p.student_id
    -> WHERE
    ->     s.student_id = 10;
+------------+------------+-----------+----------------+
| student_id | first_name | last_name | total_payments |
+------------+------------+-----------+----------------+
|         10 | Shivam     | Singh     |         100.00 |
+------------+------------+-----------+----------------+
1 row in set (0.00 sec)
```

2.

```
mysql> SELECT
    ->     c.course_id,
    ->     c.course_name,
    ->     COUNT(e.student_id) AS enrolled_students_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        | enrolled_students_count |
+-----------+--------------------+--------------------------+
|         1 | Mathematics        |                        1 |
|         2 | English Literature |                        1 |
|         3 | Computer Science   |                        0 |
|         4 | Physics            |                        1 |
|         5 | History            |                        1 |
|         6 | Chemistry          |                        1 |
|         7 | Biology            |                        1 |
|         8 | Economics          |                        1 |
|         9 | Psychology         |                        1 |
|        10 | Art History        |                        1 |
+-----------+--------------------+--------------------------+
10 rows in set (0.01 sec)
```

3.

```
mysql> SELECT
    ->     s.student_id,
    ->     s.first_name,
    ->     s.last_name
    -> FROM
    ->     Students s
    -> LEFT JOIN
    ->     Enrollments e ON s.student_id = e.student_id
    ->
    -> WHERE
    ->     e.student_id IS NULL;
+------------+------------+-----------+
| student_id | first_name | last_name |
+------------+------------+-----------+
|         12 | Alice      | Smith     |
+------------+------------+-----------+
1 row in set (0.00 sec)
```

4.

```
mysql> SELECT
    ->     s.first_name,
    ->     s.last_name,
    ->     c.course_name
    -> FROM
    ->     Students s
    -> JOIN
    ->     Enrollments e ON s.student_id = e.student_id
    -> JOIN
    ->     Courses c ON e.course_id = c.course_id;
+------------+-----------+--------------------+
| first_name | last_name | course_name        |
+------------+-----------+--------------------+
| Shivam     | Singh     | Mathematics        |
| John       | Doe       | English Literature |
| Bob        | Johnson   | Physics            |
| Emma       | Davis     | History            |
| Chris      | Brown     | Chemistry          |
| Olivia     | Moore     | Biology            |
| Daniel     | Miller    | Economics          |
| Sophia     | Jones     | Psychology         |
| William    | White     | Art History        |
+------------+-----------+--------------------+
9 rows in set (0.00 sec)
```

5.

```
mysql> SELECT
    ->     t.first_name AS teacher_first_name,
    ->     t.last_name AS teacher_last_name,
    ->     c.course_name
    -> FROM
    ->     Teacher t
    -> JOIN
    ->     Courses c ON t.teacher_id = c.teacher_id;
+--------------------+-------------------+--------------------+
| teacher_first_name | teacher_last_name | course_name        |
+--------------------+-------------------+--------------------+
| Jane               | Smith             | Mathematics        |
| Michael            | Johnson           | English Literature |
| Michael            | Johnson           | Computer Science   |
| Andrew             | Taylor            | Physics            |
| Madison            | Williams          | History            |
| Aiden              | Davis             | Chemistry          |
| Abigail            | Moore             | Biology            |
| Ethan              | Miller            | Economics          |
| Chloe              | Jones             | Psychology         |
| Mason              | White             | Art History        |
+--------------------+-------------------+--------------------+
10 rows in set (0.00 sec)
```

6.

```
mysql> SELECT
    ->     s.first_name,
    ->     s.last_name,
    ->     e.enrollment_date,
    ->     c.course_name
    -> FROM
    ->     Students s
    -> JOIN
    ->     Enrollments e ON s.student_id = e.student_id
    -> JOIN
    ->     Courses c ON e.course_id = c.course_id
    -> WHERE
    ->     c.course_name = 'Biology';
+------------+-----------+-----------------+-------------+
| first_name | last_name | enrollment_date | course_name |
+------------+-----------+-----------------+-------------+
| Olivia     | Moore     | 2024-01-25      | Biology     |
+------------+-----------+-----------------+-------------+
1 row in set (0.00 sec)
```

7.

```
mysql> SELECT
    ->     s.first_name,
    ->     s.last_name
    -> FROM
    ->     Students s
    -> LEFT JOIN
    ->     Payments p ON s.student_id = p.student_id
    -> WHERE
    ->     p.payment_id IS NULL;
Empty set (0.00 sec)
```

8.

```
mysql> SELECT
    ->     c.course_id,
    ->     c.course_name
    -> FROM
    ->     Courses c
    -> LEFT JOIN
    ->     Enrollments e ON c.course_id = e.course_id
    -> WHERE
    ->     e.course_id IS NULL;
+-----------+------------------+
| course_id | course_name      |
+-----------+------------------+
|         3 | Computer Science |
+-----------+------------------+
1 row in set (0.00 sec)
```

9.

```
mysql> SELECT
    ->     e1.student_id,
    ->     s.first_name,
    ->     s.last_name,
    ->     COUNT(e1.course_id) AS enrolled_courses_count
    -> FROM
    ->     Enrollments e1
    -> JOIN
    ->     Students s ON e1.student_id = s.student_id
    -> GROUP BY
    ->     e1.student_id
    -> HAVING
    ->     COUNT(e1.course_id) > 1;
Empty set (0.00 sec)
```

10.

```
mysql> SELECT
    ->     t.teacher_id,
    ->     t.first_name,
    ->     t.last_name
    -> FROM
    ->     Teacher t
    -> LEFT JOIN
    ->     Courses c ON t.teacher_id = c.teacher_id
    -> WHERE
    ->     c.course_id IS NULL;
+------------+------------+-----------+
| teacher_id | first_name | last_name |
+------------+------------+-----------+
|          3 | Emily      | Brown     |
+------------+------------+-----------+
1 row in set (0.00 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.

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.

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

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

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

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

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

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

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

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.

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.

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.

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.

1.

```
mysql> SELECT
    ->     course_id,
    ->     AVG(enrollment_count) AS average_students_per_course
    -> FROM (
    ->     SELECT
    ->         course_id,
    ->         COUNT(student_id) AS enrollment_count
    ->     FROM
    ->         Enrollments
    ->     GROUP BY
    ->         course_id
    -> ) AS course_enrollment_counts
    -> GROUP BY
    ->     course_id;
+-----------+-----------------------------+
| course_id | average_students_per_course |
+-----------+-----------------------------+
|         1 |                      1.0000 |
|         2 |                      1.0000 |
|         4 |                      1.0000 |
|         5 |                      1.0000 |
|         6 |                      1.0000 |
|         7 |                      1.0000 |
|         8 |                      1.0000 |
|         9 |                      1.0000 |
|        10 |                      1.0000 |
+-----------+-----------------------------+
9 rows in set (0.00 sec)
```

2.

```
mysql> SELECT
    ->     s.student_id,
    ->     s.first_name,
    ->     s.last_name,
    ->     p.amount AS highest_payment_amount
    -> FROM
    ->     Students s
    -> JOIN
    ->     Payments p ON s.student_id = p.student_id
    -> WHERE
    ->     p.amount = (
    ->         SELECT
    ->             MAX(amount)
    ->         FROM
    ->             Payments
    ->     );
+------------+------------+-----------+-------------------------+
| student_id | first_name | last_name | highest_payment_amount |
+------------+------------+-----------+-------------------------+
|         17 | Daniel     | Miller    |                  200.00 |
+------------+------------+-----------+-------------------------+
1 row in set (0.00 sec)
```

3.

```
mysql> SELECT
    ->     c.course_id,
    ->     c.course_name,
    ->     enrollment_count
    -> FROM
    ->     Courses c
    -> JOIN (
    ->     SELECT
    ->         course_id,
    ->         COUNT(student_id) AS enrollment_count
    ->     FROM
    ->         Enrollments
    ->     GROUP BY
    ->         course_id
    -> ) AS course_enrollment_counts ON c.course_id = course_enrollment_counts.course_id
    -> WHERE
    ->     enrollment_count = (
    ->         SELECT
    ->             MAX(enrollment_count)
    ->         FROM (
    ->             SELECT
    ->                 course_id,
    ->                 COUNT(student_id) AS enrollment_count
    ->             FROM
    ->                 Enrollments
    ->             GROUP BY
    ->                 course_id
    ->         ) AS max_enrollments
    ->     );
+-----------+--------------------+-----------------+
| course_id | course_name        | enrollment_count |
+-----------+--------------------+-----------------+
|         1 | Mathematics        |               1 |
|         2 | English Literature |               1 |
|         4 | Physics            |               1 |
|         5 | History            |               1 |
|         6 | Chemistry          |               1 |
|         7 | Biology            |               1 |
|         8 | Economics          |               1 |
|         9 | Psychology         |               1 |
|        10 | Art History        |               1 |
+-----------+--------------------+-----------------+
```

4.

```
mysql> SELECT
    ->     t.teacher_id,
    ->     t.first_name,
    ->     t.last_name,
    ->     SUM(p.amount) AS total_payments
    -> FROM
    ->     Teacher t
    -> 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 | first_name | last_name | total_payments |
+------------+------------+-----------+----------------+
|          1 | Jane       | Smith     |         100.00 |
|          2 | Michael    | Johnson   |         150.00 |
|          4 | Andrew     | Taylor    |         130.00 |
|          5 | Madison    | Williams  |         110.00 |
|          6 | Aiden      | Davis     |          90.00 |
|          7 | Abigail    | Moore     |          80.00 |
|          8 | Ethan      | Miller    |         200.00 |
|          9 | Chloe      | Jones     |         170.00 |
|         10 | Mason      | White     |         140.00 |
+------------+------------+-----------+----------------+
9 rows in set (0.00 sec)
```

5.

```
mysql> SELECT
    ->     student_id,
    ->     first_name,
    ->     last_name
    -> FROM
    ->     Students
    -> WHERE
    ->     student_id IN (
    ->         SELECT
    ->             student_id
    ->         FROM
    ->             Enrollments
    ->         GROUP BY
    ->             student_id
    ->         HAVING
    ->             COUNT(DISTINCT course_id) = (
    ->                 SELECT
    ->                     COUNT(DISTINCT course_id)
    ->                 FROM
    ->                     Courses
    ->             )
    ->     );
Empty set (0.00 sec)
```

6.

```
mysql> SELECT
    ->     teacher_id,
    ->     first_name,
    ->     last_name
    -> FROM
    ->     Teacher
    -> WHERE
    ->     teacher_id NOT IN (
    ->         SELECT DISTINCT
    ->             teacher_id
    ->         FROM
    ->             Courses
    ->     );
+------------+------------+-----------+
| teacher_id | first_name | last_name |
+------------+------------+-----------+
|          3 | Emily      | Brown     |
+------------+------------+-----------+
1 row in set (0.00 sec)
```

7.

```
mysql> SELECT
    ->     AVG(age) AS average_age
    -> FROM (
    ->     SELECT
    ->         TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age
    ->     FROM
    ->         Students
    -> ) AS student_age;
+-------------+
| average_age |
+-------------+
|     24.6000 |
+-------------+
1 row in set (0.00 sec)
```

8.

```
mysql> SELECT
    ->     course_id,
    ->     course_name
    -> FROM
    ->     Courses
    -> WHERE
    ->     course_id NOT IN (
    ->         SELECT DISTINCT
    ->             course_id
    ->         FROM
    ->             Enrollments
    ->     );
+-----------+------------------+
| course_id | course_name      |
+-----------+------------------+
|         3 | Computer Science |
+-----------+------------------+
1 row in set (0.00 sec)
```

9.

```
mysql> SELECT
    ->     e.student_id,
    ->     s.first_name,
    ->     s.last_name,
    ->     e.course_id,
    ->     c.course_name,
    ->     SUM(p.amount) AS total_payments
    -> FROM
    ->     Enrollments e
    -> JOIN
    ->     Students s ON e.student_id = s.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
    ->     e.student_id, s.first_name, s.last_name, e.course_id, c.course_name;
+------------+------------+-----------+-----------+--------------------+----------------+
| student_id | first_name | last_name | course_id | course_name        | total_payments |
+------------+------------+-----------+-----------+--------------------+----------------+
|         10 | Shivam     | Singh     |         1 | Mathematics        |         100.00 |
|         11 | John       | Doe       |         2 | English Literature |         150.00 |
|         13 | Bob        | Johnson   |         4 | Physics            |         130.00 |
|         14 | Emma       | Davis     |         5 | History            |         110.00 |
|         15 | Chris      | Brown     |         6 | Chemistry          |          90.00 |
|         16 | Olivia     | Moore     |         7 | Biology            |          80.00 |
|         17 | Daniel     | Miller    |         8 | Economics          |         200.00 |
|         18 | Sophia     | Jones     |         9 | Psychology         |         170.00 |
|         19 | William    | White     |        10 | Art History        |         140.00 |
+------------+------------+-----------+-----------+--------------------+----------------+
9 rows in set (0.00 sec)
```

10.

```
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(payment_id) > 1
    ->     );
Empty set (0.00 sec)
```

11.

```
mysql> SELECT
    ->     s.student_id,
    ->     s.first_name,
    ->     s.last_name,
    ->     SUM(p.amount) 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 |
+------------+------------+-----------+----------------+
|         10 | Shivam     | Singh     |         100.00 |
|         11 | John       | Doe       |         150.00 |
|         12 | Alice      | Smith     |         150.00 |
|         13 | Bob        | Johnson   |         130.00 |
|         14 | Emma       | Davis     |         110.00 |
|         15 | Chris      | Brown     |          90.00 |
|         16 | Olivia     | Moore     |          80.00 |
|         17 | Daniel     | Miller    |         200.00 |
|         18 | Sophia     | Jones     |         170.00 |
|         19 | William    | White     |         140.00 |
+------------+------------+-----------+----------------+
10 rows in set (0.00 sec)
```

12.

```
mysql> SELECT
    ->     c.course_name,
    ->     COUNT(e.student_id) AS enrolled_students_count
    -> FROM
    ->     Courses c
    -> LEFT JOIN
    ->     Enrollments e ON c.course_id = e.course_id
    -> GROUP BY
    ->     c.course_id, c.course_name;
+--------------------+--------------------------+
| course_name        | enrolled_students_count |
+--------------------+--------------------------+
| Mathematics        |                        1 |
| English Literature |                        1 |
| Computer Science   |                        0 |
| Physics            |                        1 |
| History            |                        1 |
| Chemistry          |                        1 |
| Biology            |                        1 |
| Economics          |                        1 |
| Psychology         |                        1 |
| Art History        |                        1 |
+--------------------+--------------------------+
10 rows in set (0.00 sec)
```

13.

```
mysql> SELECT
    ->     s.student_id,
    ->     s.first_name,
    ->     s.last_name,
    ->     AVG(p.amount) 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;
+------------+------------+-----------+----------------------+
| student_id | first_name | last_name | average_payment_amount |
+------------+------------+-----------+----------------------+
|         10 | Shivam     | Singh     |           100.000000 |
|         11 | John       | Doe       |           150.000000 |
|         12 | Alice      | Smith     |           150.000000 |
|         13 | Bob        | Johnson   |           130.000000 |
|         14 | Emma       | Davis     |           110.000000 |
|         15 | Chris      | Brown     |            90.000000 |
|         16 | Olivia     | Moore     |            80.000000 |
|         17 | Daniel     | Miller    |           200.000000 |
|         18 | Sophia     | Jones     |           170.000000 |
|         19 | William    | White     |           140.000000 |
+------------+------------+-----------+----------------------+
10 rows in set (0.00 sec)
```