Assignment 2 SQL

By- Kunal Sudhir Mishra

Task 1. Database Design:

1. Create the database named "SISDB".

```
mysql> CREATE DATABASE SISDB;
Query OK, 1 row affected (0.01 sec)

mysql> use SISDB;
Database changed
```

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 Table

```
mysql> 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(15)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

b. Courses Table

```
mysql> 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)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

c. Enrolments Table

```
mysql> 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)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

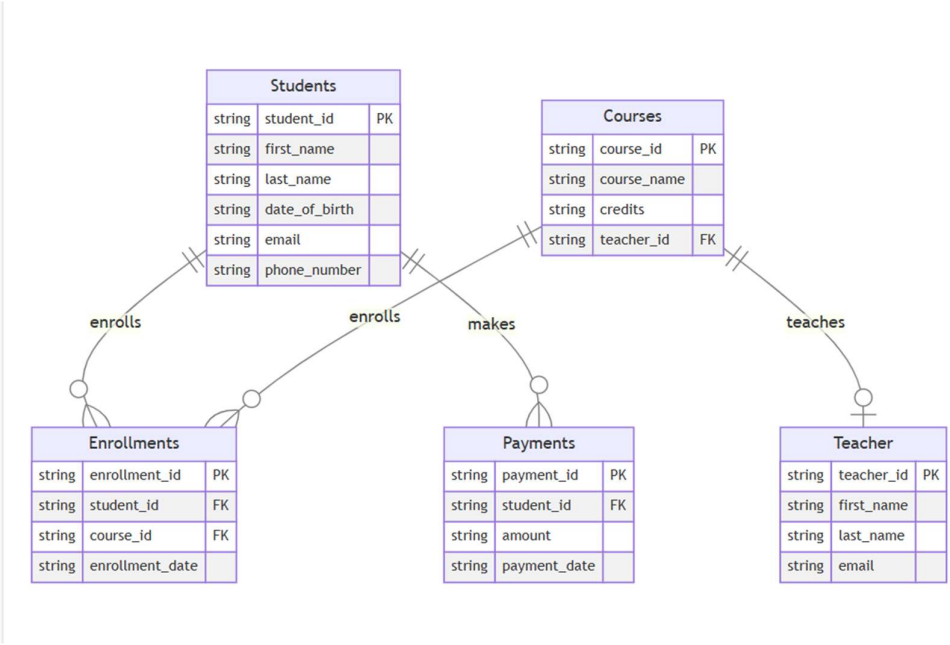
d. Teacher Table

```
mysql> CREATE TABLE Teacher (
    ->     teacher_id INT PRIMARY KEY,
    ->     first_name VARCHAR(50),
    ->     last_name VARCHAR(50),
    ->     email VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

e. Payments Table

```
mysql> 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)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

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



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

```
mysql> desc Courses;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| course_id   | int          | NO   | PRI | NULL    |       |
| course_name | varchar(100) | YES  |     | NULL    |       |
| credits     | int          | YES  |     | NULL    |       |
| teacher_id  | int          | YES  | MUL | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> desc Teacher;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| teacher_id | int          | NO   | PRI | NULL    |       |
| first_name | varchar(50)  | YES  |     | NULL    |       |
| last_name  | varchar(50)  | YES  |     | NULL    |       |
| email      | varchar(100) | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
mysql> desc Students;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| student_id    | int          | NO   | PRI | NULL    |       |
| first_name    | varchar(50)  | YES  |     | NULL    |       |
| last_name     | varchar(50)  | YES  |     | NULL    |       |
| date_of_birth | date         | YES  |     | NULL    |       |
| email         | varchar(100) | YES  |     | NULL    |       |
| phone_number  | varchar(15)  | YES  |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql> desc Enrollments;
+-----------------+------+------+-----+---------+-------+
| Field           | Type | Null | Key | Default | Extra |
+-----------------+------+------+-----+---------+-------+
| enrollment_id   | int  | NO   | PRI | NULL    |       |
| student_id      | int  | YES  | MUL | NULL    |       |
| course_id       | int  | YES  | MUL | NULL    |       |
| enrollment_date | date | YES  |     | NULL    |       |
+-----------------+------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
mysql> desc Payments;
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| payment_id   | int           | NO   | PRI | NULL    |       |
| 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 (
    ->     student_id,
    ->     first_name,
    ->     last_name,
    ->     date_of_birth,
    ->     email,
    ->     phone_number
    ->   )
    -> VALUES
    ->   (
    ->     1,
    ->     'Rahul',
    ->     'Kumar',
    ->     '1995-05-15',
    ->     'rahul@gmail.com',
    ->     '9876543210'
    ->   ),
    ->   (
    ->     2,
    ->     'Priya',
    ->     'Sharma',
    ->     '1996-08-20',
    ->     'priya@gmail.com',
    ->     '8765432109'
    ->   ),
    ->   (
    ->     3,
    ->     'Amit',
    ->     'Singh',
    ->     '1994-03-10',
    ->     'amit@gmail.com',
```

```
mysql> select * from Students;
+------------+------------+-----------+---------------+------------------+--------------+
| student_id | first_name | last_name | date_of_birth | email            | phone_number |
+------------+------------+-----------+---------------+------------------+--------------+
|          1 | Rahul      | Kumar     | 1995-05-15    | rahul@gmail.com  | 9876543210   |
|          2 | Priya      | Sharma    | 1996-08-20    | priya@gmail.com  | 8765432109   |
|          3 | Amit       | Singh     | 1994-03-10    | amit@gmail.com   | 7654321098   |
|          4 | Sneha      | Verma     | 1997-11-25    | sneha@gmail.com  | 6543210987   |
|          5 | Rajesh     | Gupta     | 1993-07-05    | rajesh@gmail.com | 5432109876   |
|          6 | Anjali     | Patel     | 1998-02-12    | anjali@gmail.com | 4321098765   |
|          7 | Vikram     | Yadav     | 1992-09-30    | vikram@gmail.com | 3210987654   |
|          8 | Neha       | Jain      | 1999-06-18    | neha@gmail.com   | 2109876543   |
|          9 | Aakash     | Shah      | 1991-01-08    | aakash@gmail.com | 1098765432   |
|         10 | Pooja      | Mishra    | 2000-04-22    | pooja@gmail.com  | 0987654321   |
+------------+------------+-----------+---------------+------------------+--------------+
10 rows in set (0.00 sec)
```

ii. Courses

```
mysql> INSERT INTO
    ->   Courses (course_id, course_name, credits, teacher_id)
    -> VALUES
    ->   (1, 'Mathematics', 3, 1),
    ->   (2, 'English Literature', 4, 2),
    ->   (3, 'Computer Science', 3, 3),
    ->   (4, 'Physics', 4, 4),
    ->   (5, 'Chemistry', 3, 5),
    ->   (6, 'Biology', 4, 6),
    ->   (7, 'History', 3, 7),
    ->   (8, 'Geography', 4, 8),
    ->   (9, 'Economics', 3, 9),
    ->   (10, 'Political Science', 4, 10);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

iii. Enrollments

```
mysql> INSERT INTO
    ->   Enrollments (
    ->     enrollment_id,
    ->     student_id,
    ->     course_id,
    ->     enrollment_date
    ->   )
    -> VALUES
    ->   (1, 1, 1, '2021-09-01'),
    ->   (2, 2, 2, '2021-09-02'),
    ->   (3, 3, 3, '2021-09-03'),
    ->   (4, 4, 4, '2021-09-04'),
    ->   (5, 5, 5, '2021-09-05'),
    ->   (6, 6, 6, '2021-09-06'),
    ->   (7, 7, 7, '2021-09-07'),
    ->   (8, 8, 8, '2021-09-08'),
    ->   (9, 9, 9, '2021-09-09'),
    ->   (10, 10, 10, '2021-09-10');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

iv. Teacher

```
mysql> INSERT INTO
    ->   Teacher (teacher_id, first_name, last_name, email)
    -> VALUES
    ->   (1, 'Rajesh', 'Kumar', 'rajesh_teacher@gmail.com'),
    ->   (2, 'Anita', 'Verma', 'anita_teacher@gmail.com'),
    ->   (3, 'Amit', 'Sharma', 'amit_teacher@gmail.com'),
    ->   (4, 'Sneha', 'Gupta', 'sneha_teacher@gmail.com'),
    ->   (5, 'Rahul', 'Patel', 'rahul_teacher@gmail.com'),
    ->   (6, 'Priya', 'Yadav', 'priya_teacher@gmail.com'),
    ->   (7, 'Vikram', 'Jain', 'vikram_teacher@gmail.com'),
    ->   (8, 'Neha', 'Shah', 'neha_teacher@gmail.com'),
    ->   (
    ->     9,
    ->     'Aakash',
    ->     'Mishra',
    ->     'aakash_teacher@gmail.com'
    ->   ),
    ->   (10, 'Pooja', 'Singh', 'pooja_teacher@gmail.com');
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

v. Payments

```
mysql> INSERT INTO
    ->   Payments (payment_id, student_id, amount, payment_date)
    -> VALUES
    ->   (1, 1, 1000, '2021-09-01'),
    ->   (2, 2, 1500, '2021-09-02'),
    ->   (3, 3, 2000, '2021-09-03'),
    ->   (4, 4, 2500, '2021-09-04'),
    ->   (5, 5, 3000, '2021-09-05'),
    ->   (6, 6, 3500, '2021-09-06'),
    ->   (7, 7, 4000, '2021-09-07'),
    ->   (8, 8, 4500, '2021-09-08'),
    ->   (9, 9, 5000, '2021-09-09'),
    ->   (10, 10, 5500, '2021-09-10');
Query OK, 10 rows affected (0.00 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@example.com',
    ->     '1234567890'
    ->   );
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 | Rahul      | Kumar     | 1995-05-15    | rahul@gmail.com      | 9876543210   |
|          2 | Priya      | Sharma    | 1996-08-20    | priya@gmail.com      | 8765432109   |
|          3 | Amit       | Singh     | 1994-03-10    | amit@gmail.com       | 7654321098   |
|          4 | Sneha      | Verma     | 1997-11-25    | sneha@gmail.com      | 6543210987   |
|          5 | Rajesh     | Gupta     | 1993-07-05    | rajesh@gmail.com     | 5432109876   |
|          6 | Anjali     | Patel     | 1998-02-12    | anjali@gmail.com     | 4321098765   |
|          7 | Vikram     | Yadav     | 1992-09-30    | vikram@gmail.com     | 3210987654   |
|          8 | Neha       | Jain      | 1999-06-18    | neha@gmail.com       | 2109876543   |
|          9 | Aakash     | Shah      | 1991-01-08    | aakash@gmail.com     | 1098765432   |
|         10 | Pooja      | Mishra    | 2000-04-22    | pooja@gmail.com      | 0987654321   |
|         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 (enrollment_id, student_id, course_id, enrollment_date)
    -> VALUES (11, 1, 2, '2024-01-30');
Query OK, 1 row affected (0.01 sec)

mysql> select * from Enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|             1 |          1 |         1 | 2021-09-01      |
|             2 |          2 |         2 | 2021-09-02      |
|             3 |          3 |         3 | 2021-09-03      |
|             4 |          4 |         4 | 2021-09-04      |
|             5 |          5 |         5 | 2021-09-05      |
|             6 |          6 |         6 | 2021-09-06      |
|             7 |          7 |         7 | 2021-09-07      |
|             8 |          8 |         8 | 2021-09-08      |
|             9 |          9 |         9 | 2021-09-09      |
|            10 |         10 |        10 | 2021-09-10      |
|            11 |          1 |         2 | 2024-01-30      |
+---------------+------------+-----------+-----------------+
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 = 'siddharth@gmail.com'
    -> WHERE  teacher_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

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 = 3
    ->   AND course_id = 2;
Query OK, 0 rows affected (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 = (
    ->     SELECT
    ->       teacher_id
    ->     FROM
    ->       Teacher
    ->     WHERE
    ->       teacher_id = 2
    ->   )
    -> WHERE
    ->   course_id = 3;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

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 Payments WHERE student_id = 3;
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM Enrollments WHERE student_id = 3;
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM Students WHERE student_id = 3;
Query OK, 1 row affected (0.01 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 = 5000
    -> WHERE
    ->   payment_id = 4;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Payments;
+------------+------------+---------+--------------+
| payment_id | student_id | amount  | payment_date |
+------------+------------+---------+--------------+
|          1 |          1 | 1000.00 | 2021-09-01   |
|          2 |          2 | 1500.00 | 2021-09-02   |
|          4 |          4 | 5000.00 | 2021-09-04   |
|          5 |          5 | 3000.00 | 2021-09-05   |
|          6 |          6 | 3500.00 | 2021-09-06   |
|          7 |          7 | 4000.00 | 2021-09-07   |
|          8 |          8 | 4500.00 | 2021-09-08   |
|          9 |          9 | 5000.00 | 2021-09-09   |
|         10 |         10 | 5500.00 | 2021-09-10   |
+------------+------------+---------+--------------+
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
    ->   SUM(amount)
    -> FROM
    ->   Payments
    -> WHERE
    ->   student_id = 5;
+-------------+
| SUM(amount) |
+-------------+
|     3000.00 |
+-------------+
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_name,
    ->   COUNT(Enrollments.student_id) AS enrolled_students
    -> FROM
    ->   Courses
    ->   JOIN Enrollments ON Courses.course_id = Enrollments.course_id
    -> GROUP BY
    ->   Courses.course_name;
+--------------------+-------------------+
| course_name        | enrolled_students |
+--------------------+-------------------+
| Mathematics        |                 1 |
| Physics            |                 1 |
| Chemistry          |                 1 |
| Biology            |                 1 |
| History            |                 1 |
| Geography          |                 1 |
| Economics          |                 1 |
| Political Science  |                 1 |
| English Literature |                 1 |
+--------------------+-------------------+
9 rows in set (0.00 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
    ->   first_name,
    ->   last_name
    -> FROM
    ->   Students
    ->   LEFT JOIN Enrollments ON Students.student_id = Enrollments.student_id
    -> WHERE
    ->   Enrollments.student_id IS NULL;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| Priya      | Sharma    |
| John       | Doe       |
+------------+-----------+
2 rows in set (0.00 sec)
```

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

```
mysql> SELECT
    ->   Students.first_name,
    ->   Students.last_name,
    ->   Courses.course_name
    -> 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_name        |
+------------+-----------+--------------------+
| Rahul      | Kumar     | Mathematics        |
| Sneha      | Verma     | Physics            |
| Rajesh     | Gupta     | Chemistry          |
| Anjali     | Patel     | Biology            |
| Vikram     | Yadav     | History            |
| Neha       | Jain      | Geography          |
| Aakash     | Shah      | Economics          |
| Pooja      | Mishra    | Political Science  |
| Rahul      | Kumar     | English Literature |
+------------+-----------+--------------------+
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
    ->   t.first_name,
    ->   t.last_name,
    ->   c.course_name
    -> FROM
    ->   Teacher t
    ->   JOIN Courses c ON t.teacher_id = c.teacher_id;
+------------+-----------+--------------------+
| first_name | last_name | course_name        |
+------------+-----------+--------------------+
| Rajesh     | Kumar     | Mathematics        |
| Anita      | Verma     | English Literature |
| Anita      | Verma     | Computer Science   |
| Sneha      | Gupta     | Physics            |
| Rahul      | Patel     | Chemistry          |
| Priya      | Yadav     | Biology            |
| Vikram     | Jain      | History            |
| Neha       | Shah      | Geography          |
| Aakash     | Mishra    | Economics          |
| Pooja      | Singh     | Political Science  |
+------------+-----------+--------------------+
10 rows in set (0.00 sec)
```

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

```
mysql> SELECT
    ->   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 = 'Physics';
+------------+-----------+-----------------+
| first_name | last_name | enrollment_date |
+------------+-----------+-----------------+
| Sneha      | Verma     | 2021-09-04      |
+------------+-----------+-----------------+
1 row in set (0.00 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
    ->   first_name,
    ->   last_name
    -> FROM
    ->   Students
    ->   LEFT JOIN Payments ON Students.student_id = Payments.student_id
    -> WHERE
    ->   Payments.payment_id IS NULL;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| John       | Doe       |
+------------+-----------+
1 row in set (0.00 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
    ->   course_name
    -> FROM
    ->   Courses
    ->   LEFT JOIN Enrollments ON Courses.course_id = Enrollments.course_id
    -> WHERE
    ->   Enrollments.enrollment_id IS NULL;
+------------------+
| course_name      |
+------------------+
| Computer Science |
+------------------+
1 row in set (0.00 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
    ->   s.student_id,
    ->   s.first_name,
    ->   s.last_name
    -> FROM
    ->   Students s
    ->   INNER JOIN Enrollments e ON s.student_id = e.student_id
    -> GROUP BY
    ->   s.student_id,
    ->   s.first_name,
    ->   s.last_name
    -> HAVING
    ->   COUNT(e.enrollment_id) > 1;
+------------+------------+-----------+
| student_id | first_name | last_name |
+------------+------------+-----------+
|          1 | Rahul      | Kumar     |
+------------+------------+-----------+
1 row in set (0.00 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
    ->   t.*
    -> 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 | email                  |
+------------+------------+-----------+------------------------+
|          3 | Amit       | Sharma    | amit_teacher@gmail.com |
+------------+------------+-----------+------------------------+
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.**

```
mysql> SELECT
    ->   course_name,
    ->   AVG(num_students) AS average_students
    -> FROM
    ->   (
    ->     SELECT
    ->       c.course_name,
    ->       COUNT(*) AS num_students
    ->     FROM
    ->       Courses c
    ->       INNER JOIN Enrollments e ON c.course_id = e.course_id
    ->     GROUP BY
    ->       c.course_name
    ->   ) AS subquery
    -> GROUP BY
    ->   course_name;
+--------------------+------------------+
| course_name        | average_students |
+--------------------+------------------+
| Mathematics        |           1.0000 |
| English Literature |           1.0000 |
| Physics            |           1.0000 |
| Chemistry          |           1.0000 |
| Biology            |           1.0000 |
| History            |           1.0000 |
| Geography          |           1.0000 |
| Economics          |           1.0000 |
| Political Science  |           1.0000 |
+--------------------+------------------+
9 rows in set (0.00 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
    -> FROM
    ->   Payments
    -> WHERE
    ->   amount = (
    ->     SELECT
    ->       MAX(amount)
    ->     FROM
    ->       Payments
    ->   );
+------------+
| student_id |
+------------+
|         10 |
+------------+
1 row in set (0.00 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
    ->   course_name
    -> FROM
    ->   Courses
    -> WHERE
    ->   course_id IN (
    ->     SELECT
    ->       course_id
    ->     FROM
    ->       Enrollments
    ->     GROUP BY
    ->       course_id
    ->     HAVING
    ->       COUNT(*) = (
    ->         SELECT
    ->           MAX(enrollment_count)
    ->         FROM
    ->           (
    ->             SELECT
    ->               course_id,
    ->               COUNT(*) as enrollment_count
    ->             FROM
    ->               Enrollments
    ->             GROUP BY
    ->               course_id
    ->           ) AS subquery
    ->       )
    ->   );
    ->       )
    ->   );
+--------------------+
| course_name        |
+--------------------+
| Mathematics        |
| English Literature |
| Physics            |
| Chemistry          |
| Biology            |
| History            |
| Geography          |
| Economics          |
| Political Science  |
+--------------------+
9 rows in set (0.00 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,
    ->   SUM(p.amount) AS total_payments
    -> FROM
    ->   Teacher t
    ->   JOIN Courses c ON t.teacher_id = c.teacher_id
    ->   JOIN Enrollments e ON c.course_id = e.course_id
    ->   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 | Rajesh     | Kumar     |        1000.00 |
|          4 | Sneha      | Gupta     |        5000.00 |
|          5 | Rahul      | Patel     |        3000.00 |
|          6 | Priya      | Yadav     |        3500.00 |
|          7 | Vikram     | Jain      |        4000.00 |
|          8 | Neha       | Shah      |        4500.00 |
|          9 | Aakash     | Mishra    |        5000.00 |
|         10 | Pooja      | Singh     |        5500.00 |
|          2 | Anita      | Verma     |        1000.00 |
+------------+------------+-----------+----------------+
9 rows in set (0.00 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
    ->   s.student_id,
    ->   s.first_name,
    ->   s.last_name
    -> FROM
    ->   Students s
    -> WHERE
    ->   NOT EXISTS (
    ->     SELECT
    ->       c.course_id
    ->     FROM
    ->       Courses c
    ->     WHERE
    ->       NOT EXISTS (
    ->         SELECT
    ->           e.course_id
    ->         FROM
    ->           Enrollments e
    ->         WHERE
    ->           e.student_id = s.student_id
    ->           AND e.course_id = c.course_id
    ->       )
    ->   )
    -> ;
Empty set (0.00 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
    ->   first_name,
    ->   last_name
    -> FROM
    ->   Teacher
    -> WHERE
    ->   teacher_id NOT IN (
    ->     SELECT
    ->       teacher_id
    ->     FROM
    ->       Courses
    ->   )
    -> ;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| Amit       | Sharma    |
+------------+-----------+
1 row in set (0.00 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(age) AS average_age
    -> FROM
    ->   (
    ->     SELECT
    ->       student_id,
    ->       DATEDIFF(CURDATE(), date_of_birth) / 365 AS age
    ->     FROM
    ->       Students
    ->   ) AS student_age
    -> ;
+-------------+
| average_age |
+-------------+
| 28.00575342 |
+-------------+
1 row in set (0.00 sec)
```

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

```
mysql> SELECT
    ->   course_id,
    ->   course_name
    -> FROM
    ->   Courses
    -> WHERE
    ->   course_id NOT IN (
    ->     SELECT
    ->       course_id
    ->     FROM
    ->       Enrollments
    ->   );
+-----------+------------------+
| course_id | course_name      |
+-----------+------------------+
|         3 | Computer Science |
+-----------+------------------+
1 row in set (0.00 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,
    ->   SUM(p.amount) 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
    ->   JOIN Payments p ON s.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_payments |
+------------+------------+-----------+-----------+--------------------+----------------+
|          1 | Rahul      | Kumar     |         1 | Mathematics        |        1000.00 |
|          4 | Sneha      | Verma     |         4 | Physics            |        5000.00 |
|          5 | Rajesh     | Gupta     |         5 | Chemistry          |        3000.00 |
|          6 | Anjali     | Patel     |         6 | Biology            |        3500.00 |
|          7 | Vikram     | Yadav     |         7 | History            |        4000.00 |
|          8 | Neha       | Jain      |         8 | Geography          |        4500.00 |
|          9 | Aakash     | Shah      |         9 | Economics          |        5000.00 |
|         10 | Pooja      | Mishra    |        10 | Political Science  |        5500.00 |
|          1 | Rahul      | Kumar     |         2 | English Literature |        1000.00 |
+------------+------------+-----------+-----------+--------------------+----------------+
9 rows in set (0.00 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
    ->   s.student_id,
    ->   s.first_name,
    ->   s.last_name
    -> FROM
    ->   Students s
    -> WHERE
    ->   s.student_id IN (
    ->     SELECT
    ->       p.student_id
    ->     FROM
    ->       Payments p
    ->     GROUP BY
    ->       p.student_id
    ->     HAVING
    ->       COUNT(p.payment_id) > 1
    ->   )
    -> ;
Empty set (0.00 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
    ->   Students.student_id,
    ->   Students.first_name,
    ->   Students.last_name,
    ->   SUM(Payments.amount) AS total_payments
    -> FROM
    ->   Students
    ->   JOIN Payments ON Students.student_id = Payments.student_id
    -> GROUP BY
    ->   Students.student_id,
    ->   Students.first_name,
    ->   Students.last_name
    -> ;
+------------+------------+-----------+----------------+
| student_id | first_name | last_name | total_payments |
+------------+------------+-----------+----------------+
|          1 | Rahul      | Kumar     |        1000.00 |
|          2 | Priya      | Sharma    |        1500.00 |
|          4 | Sneha      | Verma     |        5000.00 |
|          5 | Rajesh     | Gupta     |        3000.00 |
|          6 | Anjali     | Patel     |        3500.00 |
|          7 | Vikram     | Yadav     |        4000.00 |
|          8 | Neha       | Jain      |        4500.00 |
|          9 | Aakash     | Shah      |        5000.00 |
|         10 | Pooja      | Mishra    |        5500.00 |
+------------+------------+-----------+----------------+
9 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_name,
    ->   COUNT(e.student_id) AS enrollment_count
    -> FROM
    ->   Courses c
    ->   JOIN Enrollments e ON c.course_id = e.course_id
    -> GROUP BY
    ->   c.course_name;
+--------------------+------------------+
| course_name        | enrollment_count |
+--------------------+------------------+
| Mathematics        |                1 |
| Physics            |                1 |
| Chemistry          |                1 |
| Biology            |                1 |
| History            |                1 |
| Geography          |                1 |
| Economics          |                1 |
| Political Science  |                1 |
| English Literature |                1 |
+--------------------+------------------+
9 rows in set (0.00 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
    ->   AVG(amount)
    -> FROM
    ->   Students
    ->   JOIN Payments ON Students.student_id = Payments.student_id
    -> GROUP BY
    ->   Students.student_id;
+-------------+
| AVG(amount) |
+-------------+
| 1000.000000 |
| 1500.000000 |
| 5000.000000 |
| 3000.000000 |
| 3500.000000 |
| 4000.000000 |
| 4500.000000 |
| 5000.000000 |
| 5500.000000 |
+-------------+
9 rows in set (0.00 sec)
```