ASSIGNMENT 2

Task 1: Database Design

Create database named “SISDB”

```
mysql> create database SISDB;
Query OK, 1 row affected (0.01 sec)

mysql> use SISDB;
Database changed
mysql>
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sisdb              |
| sys                |
| techshop           |
| world              |
+--------------------+
8 rows in set (0.00 sec)
```

Create Tables:

- Students
- Courses
- Enrollments
- Teacher
- Payments

```
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 (0.04 sec)

mysql> desc table Students;
+----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+-------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra |
+----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+-------+
|  1 | SIMPLE      | Students | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    1 |   100.00 | NULL  |
+----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+-------+
1 row in set, 1 warning (0.01 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> CREATE TABLE Teacher (
    ->     teacher_id INT PRIMARY KEY,
    ->     first_name VARCHAR(50),
    ->     last_name VARCHAR(50),
    ->     email VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.04 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> 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 (0.04 sec)

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 (0.00 sec)
```

```
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 (0.06 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> 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 (0.06 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)
```

Insert at least 10 values in each of the table

```
mysql> INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number)
    -> VALUES
    -> (1, 'John', 'Doe', '1990-01-15', 'john.doe@example.com', '123-456-7890'),
    -> (2, 'Jane', 'Smith', '1992-05-22', 'jane.smith@example.com', '987-654-3210'),
    -> (3, 'Michael', 'Johnson', '1993-08-10', 'michael.johnson@example.com', '555-123-4567'),
    -> (4, 'Emily', 'Williams', '1995-03-27', 'emily.williams@example.com', '789-012-3456'),
    -> (5, 'David', 'Brown', '1994-12-03', 'david.brown@example.com', '321-654-0987'),
    -> (6, 'Sophia', 'Miller', '1991-07-18', 'sophia.miller@example.com', '111-222-3333'),
    -> (7, 'Daniel', 'Jones', '1996-02-09', 'daniel.jones@example.com', '444-777-8888'),
    -> (8, 'Olivia', 'Davis', '1997-11-05', 'olivia.davis@example.com', '666-999-0000'),
    -> (9, 'Liam', 'Taylor', '1998-09-14', 'liam.taylor@example.com', '222-333-4444'),
    -> (10, 'Ava', 'Moore', '1999-04-30', 'ava.moore@example.com', '888-555-6666');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Students;
+------------+------------+-----------+---------------+---------------------------+--------------+
| student_id | first_name | last_name | date_of_birth | email                     | phone_number |
+------------+------------+-----------+---------------+---------------------------+--------------+
|          1 | John       | Doe       | 1990-01-15    | john.doe@example.com      | 123-456-7890 |
|          2 | Jane       | Smith     | 1992-05-22    | jane.smith@example.com    | 987-654-3210 |
|          3 | Michael    | Johnson   | 1993-08-10    | michael.johnson@example.com | 555-123-4567 |
|          4 | Emily      | Williams  | 1995-03-27    | emily.williams@example.com | 789-012-3456 |
|          5 | David      | Brown     | 1994-12-03    | david.brown@example.com   | 321-654-0987 |
|          6 | Sophia     | Miller    | 1991-07-18    | sophia.miller@example.com | 111-222-3333 |
|          7 | Daniel     | Jones     | 1996-02-09    | daniel.jones@example.com  | 444-777-8888 |
|          8 | Olivia     | Davis     | 1997-11-05    | olivia.davis@example.com  | 666-999-0000 |
|          9 | Liam       | Taylor    | 1998-09-14    | liam.taylor@example.com   | 222-333-4444 |
|         10 | Ava        | Moore     | 1999-04-30    | ava.moore@example.com     | 888-555-6666 |
+------------+------------+-----------+---------------+---------------------------+--------------+
10 rows in set (0.00 sec)
```

```
mysql> INSERT INTO Teacher (teacher_id, first_name, last_name, email)
    -> VALUES
    -> (1, 'Professor', 'Johnson', 'prof.johnson@example.com'),
    -> (2, 'Professor', 'Williams', 'prof.williams@example.com'),
    -> (3, 'Dr.', 'Anderson', 'dr.anderson@example.com'),
    -> (4, 'Mrs.', 'Smith', 'mrs.smith@example.com'),
    -> (5, 'Mr.', 'Brown', 'mr.brown@example.com'),
    -> (6, 'Dr.', 'Davis', 'dr.davis@example.com'),
    -> (7, 'Ms.', 'Jones', 'ms.jones@example.com'),
    -> (8, 'Professor', 'Miller', 'prof.miller@example.com'),
    -> (9, 'Dr.', 'Moore', 'dr.moore@example.com'),
    -> (10, 'Mrs.', 'Taylor', 'mrs.taylor@example.com');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Teacher;
+------------+------------+-----------+---------------------------+
| teacher_id | first_name | last_name | email                     |
+------------+------------+-----------+---------------------------+
|          1 | Professor  | Johnson   | prof.johnson@example.com  |
|          2 | Professor  | Williams  | prof.williams@example.com |
|          3 | Dr.        | Anderson  | dr.anderson@example.com   |
|          4 | Mrs.       | Smith     | mrs.smith@example.com     |
|          5 | Mr.        | Brown     | mr.brown@example.com      |
|          6 | Dr.        | Davis     | dr.davis@example.com      |
|          7 | Ms.        | Jones     | ms.jones@example.com      |
|          8 | Professor  | Miller    | prof.miller@example.com   |
|          9 | Dr.        | Moore     | dr.moore@example.com      |
|         10 | Mrs.       | Taylor    | mrs.taylor@example.com    |
+------------+------------+-----------+---------------------------+
10 rows in set (0.00 sec)
```

```
mysql> INSERT INTO Courses (course_id, course_name, credits, teacher_id)
    -> VALUES
    -> (101, 'Mathematics', 3, 1),
    -> (102, 'History', 4, 2),
    -> (103, 'Computer Science', 3, 3),
    -> (104, 'Physics', 4, 4),
    -> (105, 'English Literature', 3, 5),
    -> (106, 'Chemistry', 4, 6),
    -> (107, 'Art History', 3, 7),
    -> (108, 'Economics', 4, 8),
    -> (109, 'Biology', 3, 9),
    -> (110, 'Psychology', 4, 10);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Courses;
+-----------+--------------------+---------+------------+
| course_id | course_name        | credits | teacher_id |
+-----------+--------------------+---------+------------+
|       101 | Mathematics        |       3 |          1 |
|       102 | History            |       4 |          2 |
|       103 | Computer Science   |       3 |          3 |
|       104 | Physics            |       4 |          4 |
|       105 | English Literature |       3 |          5 |
|       106 | Chemistry          |       4 |          6 |
|       107 | Art History        |       3 |          7 |
|       108 | Economics          |       4 |          8 |
|       109 | Biology            |       3 |          9 |
|       110 | Psychology         |       4 |         10 |
+-----------+--------------------+---------+------------+
10 rows in set (0.00 sec)
```

```
mysql> INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
    -> VALUES
    -> (301, 1, 101, '2023-01-15'),
    -> (302, 2, 102, '2023-02-22'),
    -> (303, 3, 103, '2023-03-10'),
    -> (304, 4, 104, '2023-04-27'),
    -> (305, 5, 105, '2023-05-03'),
    -> (306, 6, 106, '2023-06-18'),
    -> (307, 7, 107, '2023-07-09'),
    -> (308, 8, 108, '2023-08-05'),
    -> (309, 9, 109, '2023-09-14'),
    -> (310, 10, 110, '2023-10-30');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           301 |          1 |       101 | 2023-01-15      |
|           302 |          2 |       102 | 2023-02-22      |
|           303 |          3 |       103 | 2023-03-10      |
|           304 |          4 |       104 | 2023-04-27      |
|           305 |          5 |       105 | 2023-05-03      |
|           306 |          6 |       106 | 2023-06-18      |
|           307 |          7 |       107 | 2023-07-09      |
|           308 |          8 |       108 | 2023-08-05      |
|           309 |          9 |       109 | 2023-09-14      |
|           310 |         10 |       110 | 2023-10-30      |
+---------------+------------+-----------+-----------------+
10 rows in set (0.00 sec)
```

```
mysql> INSERT INTO Payments (payment_id, student_id, amount, payment_date)
    -> VALUES
    -> (501, 1, 5000.00, '2023-01-01'),(502, 2, 7500.50, '2023-02-15'),
    -> (503, 3, 6000.00, '2023-03-10'),(504, 4, 8000.75, '2023-04-27'),
    -> (505, 5, 3500.25, '2023-05-03'),(506, 6, 1900.50, '2023-06-18'),
    -> (507, 7, 4450.00, '2023-07-09'),(508, 8, 2700.25, '2023-08-05'),
    -> (509, 9, 1550.00, '2023-09-14'),(510, 10, 1000.00, '2023-10-30');
Query OK, 10 rows affected (0.08 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Payments;
+------------+------------+---------+--------------+
| payment_id | student_id | amount  | payment_date |
+------------+------------+---------+--------------+
|        501 |          1 | 5000.00 | 2023-01-01   |
|        502 |          2 | 7500.50 | 2023-02-15   |
|        503 |          3 | 6000.00 | 2023-03-10   |
|        504 |          4 | 8000.75 | 2023-04-27   |
|        505 |          5 | 3500.25 | 2023-05-03   |
|        506 |          6 | 1900.50 | 2023-06-18   |
|        507 |          7 | 4450.00 | 2023-07-09   |
|        508 |          8 | 2700.25 | 2023-08-05   |
|        509 |          9 | 1550.00 | 2023-09-14   |
|        510 |         10 | 1000.00 | 2023-10-30   |
+------------+------------+---------+--------------+
10 rows in set (0.00 sec)
```

Task 2: Select, Where, Between, AND, LIKE:

Write SQL query to insert new student into Students table with given details.

```
mysql> INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number)
    -> VALUES
    -> (11,'Johny','Day','1995-08-15','johny.day@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 | John       | Doe       | 1990-01-15    | john.doe@example.com      | 123-456-7890 |
|          2 | Jane       | Smith     | 1992-05-22    | jane.smith@example.com    | 987-654-3210 |
|          3 | Michael    | Johnson   | 1993-08-10    | michael.johnson@example.com | 555-123-4567 |
|          4 | Emily      | Williams  | 1995-03-27    | emily.williams@example.com | 789-012-3456 |
|          5 | David      | Brown     | 1994-12-03    | david.brown@example.com   | 321-654-0987 |
|          6 | Sophia     | Miller    | 1991-07-18    | sophia.miller@example.com | 111-222-3333 |
|          7 | Daniel     | Jones     | 1996-02-09    | daniel.jones@example.com  | 444-777-8888 |
|          8 | Olivia     | Davis     | 1997-11-05    | olivia.davis@example.com  | 666-999-0000 |
|          9 | Liam       | Taylor    | 1998-09-14    | liam.taylor@example.com   | 222-333-4444 |
|         10 | Ava        | Moore     | 1999-04-30    | ava.moore@example.com     | 888-555-6666 |
|         11 | Johny      | Day       | 1995-08-15    | johny.day@example.com     | 1234567890   |
+------------+------------+-----------+---------------+---------------------------+--------------+
11 rows in set (0.00 sec)
```

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

```
mysql> INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
    -> VALUES
    -> (11, 3, 103, '2023-11-20');
Query OK, 1 row affected (0.01 sec)

mysql> select * from Enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|            11 |          3 |       103 | 2023-11-20      |
|           301 |          1 |       101 | 2023-01-15      |
|           302 |          2 |       102 | 2023-02-22      |
|           303 |          3 |       103 | 2023-03-10      |
|           304 |          4 |       104 | 2023-04-27      |
|           305 |          5 |       105 | 2023-05-03      |
|           306 |          6 |       106 | 2023-06-18      |
|           307 |          7 |       107 | 2023-07-09      |
|           308 |          8 |       108 | 2023-08-05      |
|           309 |          9 |       109 | 2023-09-14      |
|           310 |         10 |       110 | 2023-10-30      |
+---------------+------------+-----------+-----------------+
11 rows in set (0.00 sec)
```

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

```
mysql> UPDATE Teacher
    -> SET email = 'new.email@example.com'
    -> WHERE teacher_id = 3;
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 | Professor  | Johnson   | prof.johnson@example.com |
|          2 | Professor  | Williams  | prof.williams@example.com |
|          3 | Dr.        | Anderson  | new.email@example.com    |
|          4 | Mrs.       | Smith     | mrs.smith@example.com    |
|          5 | Mr.        | Brown     | mr.brown@example.com     |
|          6 | Dr.        | Davis     | dr.davis@example.com     |
|          7 | Ms.        | Jones     | ms.jones@example.com     |
|          8 | Professor  | Miller    | prof.miller@example.com  |
|          9 | Dr.        | Moore     | dr.moore@example.com     |
|         10 | Mrs.       | Taylor    | mrs.taylor@example.com   |
+------------+------------+-----------+--------------------------+
10 rows in set (0.00 sec)
```

Write SQL query to delete specific enrollment record from Enrollments table. Select enrollment record based on student and course.

```
mysql> DELETE FROM Enrollments
    -> WHERE student_id = 3 AND course_id = 103;
Query OK, 2 rows affected (0.01 sec)

mysql> select * from Enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           301 |          1 |       101 | 2023-01-15      |
|           302 |          2 |       102 | 2023-02-22      |
|           304 |          4 |       104 | 2023-04-27      |
|           305 |          5 |       105 | 2023-05-03      |
|           306 |          6 |       106 | 2023-06-18      |
|           307 |          7 |       107 | 2023-07-09      |
|           308 |          8 |       108 | 2023-08-05      |
|           309 |          9 |       109 | 2023-09-14      |
|           310 |         10 |       110 | 2023-10-30      |
+---------------+------------+-----------+-----------------+
9 rows in set (0.00 sec)
```

Update Courses table to assign specific teacher to course. Choose any course and teacher from respective tables.

```
mysql> UPDATE Courses
    -> SET teacher_id = 2
    -> WHERE course_id = 101;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Courses;
+-----------+--------------------+---------+------------+
| course_id | course_name        | credits | teacher_id |
+-----------+--------------------+---------+------------+
|       101 | Mathematics        |       3 |          2 |
|       102 | History            |       4 |          2 |
|       103 | Computer Science   |       3 |          3 |
|       104 | Physics            |       4 |          4 |
|       105 | English Literature |       3 |          5 |
|       106 | Chemistry          |       4 |          6 |
|       107 | Art History        |       3 |          7 |
|       108 | Economics          |       4 |          8 |
|       109 | Biology            |       3 |          9 |
|       110 | Psychology         |       4 |         10 |
+-----------+--------------------+---------+------------+
10 rows in set (0.00 sec)
```

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

```
mysql> DELETE FROM Enrollments
    -> WHERE student_id = 3;
Query OK, 0 rows affected (0.00 sec)

mysql> DELETE FROM Students
    -> WHERE student_id = 3;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails
EY (`student_id`) REFERENCES `students` (`student_id`))
mysql> select * from Enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           301 |          1 |       101 | 2023-01-15      |
|           302 |          2 |       102 | 2023-02-22      |
|           304 |          4 |       104 | 2023-04-27      |
|           305 |          5 |       105 | 2023-05-03      |
|           306 |          6 |       106 | 2023-06-18      |
|           307 |          7 |       107 | 2023-07-09      |
|           308 |          8 |       108 | 2023-08-05      |
|           309 |          9 |       109 | 2023-09-14      |
|           310 |         10 |       110 | 2023-10-30      |
+---------------+------------+-----------+-----------------+
9 rows in set (0.00 sec)
```

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

```
mysql> UPDATE Payments
    -> SET amount = 900.00
    -> WHERE payment_id =507;
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 |
+------------+------------+---------+--------------+
|        501 |          1 | 5000.00 | 2023-01-01   |
|        502 |          2 | 7500.50 | 2023-02-15   |
|        503 |          3 | 6000.00 | 2023-03-10   |
|        504 |          4 | 8000.75 | 2023-04-27   |
|        505 |          5 | 3500.25 | 2023-05-03   |
|        506 |          6 | 1900.50 | 2023-06-18   |
|        507 |          7 |  900.00 | 2023-07-09   |
|        508 |          8 | 2700.25 | 2023-08-05   |
|        509 |          9 | 1550.00 | 2023-09-14   |
|        510 |         10 | 1000.00 | 2023-10-30   |
+------------+------------+---------+--------------+
10 rows in set (0.00 sec)
```

Task 3: Aggregate function, Having, OrderBy, GroupBy and Joins:

Write SQL query to calculate total payments made by specific student. You will need to join Payments table with Students table based on students id.

```
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 = 3;
+------------+------------+-----------+----------------+
| student_id | first_name | last_name | total_payments |
+------------+------------+-----------+----------------+
|          3 | Michael    | Johnson   |        6000.00 |
+------------+------------+-----------+----------------+
1 row in set (0.00 sec)
```

Write SQL query to retrieve list of courses along with count of students enrolled in each course. Use join operation between Courses table and Enrollments table.

```
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
    -> ORDER BY c.course_id;
+-----------+--------------------+-------------------------+
| course_id | course_name        | enrolled_students_count |
+-----------+--------------------+-------------------------+
|       101 | Mathematics        |                       1 |
|       102 | History            |                       1 |
|       103 | Computer Science   |                       0 |
|       104 | Physics            |                       1 |
|       105 | English Literature |                       1 |
|       106 | Chemistry          |                       1 |
|       107 | Art History        |                       1 |
|       108 | Economics          |                       1 |
|       109 | Biology            |                       1 |
|       110 | Psychology         |                       1 |
+-----------+--------------------+-------------------------+
10 rows in set (0.00 sec)
```

Write SQL query to find names of students who have not enrolled in any course. Use left join between students table and Enrollments table to identify students without enrollments.

```
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 |
+------------+------------+-----------+
|          3 | Michael    | Johnson   |
|         11 | Johny      | Day       |
+------------+------------+-----------+
2 rows in set (0.00 sec)
```

Write SQL query to retrieve first name, last name of students and names of courses they are enrolled in. Use join operations between Students table and Enrollments and courses tables.

```
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        |
+------------+-----------+--------------------+
| John       | Doe       | Mathematics        |
| Jane       | Smith     | History            |
| Emily      | Williams  | Physics            |
| David      | Brown     | English Literature |
| Sophia     | Miller    | Chemistry          |
| Daniel     | Jones     | Art History        |
| Olivia     | Davis     | Economics          |
| Liam       | Taylor    | Biology            |
| Ava        | Moore     | Psychology         |
+------------+-----------+--------------------+
9 rows in set (0.00 sec)
```

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

```
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        |
+--------------------+-------------------+--------------------+
| Professor          | Williams          | Mathematics        |
| Professor          | Williams          | History            |
| Dr.                | Anderson          | Computer Science   |
| Mrs.               | Smith             | Physics            |
| Mr.                | Brown             | English Literature |
| Dr.                | Davis             | Chemistry          |
| Ms.                | Jones             | Art History        |
| Professor          | Miller            | Economics          |
| Dr.                | Moore             | Biology            |
| Mrs.               | Taylor            | Psychology         |
+--------------------+-------------------+--------------------+
10 rows in set (0.00 sec)
```

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
    ->     s.first_name,
    ->     s.last_name,
    ->     e.enrollment_date
    -> 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 ='History';
+------------+-----------+-----------------+
| first_name | last_name | enrollment_date |
+------------+-----------+-----------------+
| Jane       | Smith     | 2023-02-22      |
+------------+-----------+-----------------+
1 row in set (0.00 sec)
```

Find the names of students who have not made any payments. Use left join between the Students table and the Payments table and filter for students with null payment records.

```
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;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| Johny      | Day       |
+------------+-----------+
1 row in set (0.00 sec)
```

Write query to identify courses that have no enrollments. You'll need to use left join between Courses table and Enrollments table and filter for courses with null enrollment records.

```
mysql> SELECT
    ->     c.course_id,
    ->     c.course_name
    -> FROM
    ->     Courses c
    -> LEFT JOIN
    ->     Enrollments e ON c.course_id = e.course_id
    -> WHERE
    ->     e.enrollment_id IS NULL;
+-----------+------------------+
| course_id | course_name      |
+-----------+------------------+
|       103 | Computer Science |
+-----------+------------------+
1 row in set (0.00 sec)
```

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

Find teachers who are not assigned to any courses. Use left join between Teacher table and courses table and filter for teachers with null course assignments.

```
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 |
+------------+------------+-----------+
|          1 | Professor  | Johnson   |
+------------+------------+-----------+
1 row in set (0.00 sec)
```

Task 4. Subquery and its type:

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

```
mysql> SELECT
    ->     course_id,
    ->     course_name,
    ->     AVG(num_students_enrolled) AS avg_students_enrolled
    -> FROM (
    ->     SELECT
    ->         c.course_id,
    ->         c.course_name,
    ->         COUNT(e.student_id) AS num_students_enrolled
    ->     FROM
    ->         Courses c
    ->     LEFT JOIN
    ->         Enrollments e ON c.course_id = e.course_id
    ->     GROUP BY
    ->         c.course_id, c.course_name
    -> ) AS course_enrollments
    -> GROUP BY
    ->     course_id, course_name;
+-----------+--------------------+----------------------+
| course_id | course_name        | avg_students_enrolled |
+-----------+--------------------+----------------------+
|       101 | Mathematics        |               1.0000 |
|       102 | History            |               1.0000 |
|       103 | Computer Science   |               0.0000 |
|       104 | Physics            |               1.0000 |
|       105 | English Literature |               1.0000 |
|       106 | Chemistry          |               1.0000 |
|       107 | Art History        |               1.0000 |
|       108 | Economics          |               1.0000 |
|       109 | Biology            |               1.0000 |
|       110 | Psychology         |               1.0000 |
+-----------+--------------------+----------------------+
10 rows in set (0.00 sec)
```

Identify students who made the highest payment.Use subquery to find maximum payment amount and then retrieve the students associated with that amount.

```
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 |
+------------+------------+-----------+-----------------------+
|          4 | Emily      | Williams  |                8000.75 |
+------------+------------+-----------+-----------------------+
1 row in set (0.00 sec)
```

Retrieve list of courses with highest number of enrollments. Use subqueries to find the courses with maximum enrollment count.

```
mysql> SELECT
    ->     c.course_id,
    ->     c.course_name,
    ->     COUNT(e.student_id) AS enrollment_count
    -> FROM
    ->     Courses c
    -> LEFT JOIN
    ->     Enrollments e ON c.course_id = e.course_id
    -> GROUP BY
    ->     c.course_id, c.course_name
    -> 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_enrollment_subquery
    ->     );
+-----------+--------------------+------------------+
| course_id | course_name        | enrollment_count |
+-----------+--------------------+------------------+
|       101 | Mathematics        |                1 |
|       102 | History            |                1 |
|       104 | Physics            |                1 |
|       105 | English Literature |                1 |
|       106 | Chemistry          |                1 |
|       107 | Art History        |                1 |
|       108 | Economics          |                1 |
|       109 | Biology            |                1 |
|       110 | Psychology         |                1 |
+-----------+--------------------+------------------+
9 rows in set (0.00 sec)
```

Calculate total payments made to courses taught by each teacher. Use subqueries to sum payments for each teachers courses.

```
mysql> SELECT
    ->     t.teacher_id,
    ->     t.first_name AS teacher_first_name,
    ->     t.last_name AS teacher_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 | teacher_first_name | teacher_last_name | total_payments |
+------------+---------------------+--------------------+----------------+
|          2 | Professor          | Williams           |       12500.50 |
|          3 | Dr.                | Anderson           |           NULL |
|          4 | Mrs.               | Smith              |        8000.75 |
|          5 | Mr.                | Brown              |        3500.25 |
|          6 | Dr.                | Davis              |        1900.50 |
|          7 | Ms.                | Jones              |         900.00 |
|          8 | Professor          | Miller             |        2700.25 |
|          9 | Dr.                | Moore              |        1550.00 |
|         10 | Mrs.               | Taylor             |        1000.00 |
+------------+---------------------+--------------------+----------------+
9 rows in set (0.00 sec)
```

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
    ->     (
    ->         SELECT
    ->             COUNT(DISTINCT e.course_id)
    ->         FROM
    ->             Courses c
    ->         LEFT JOIN
    ->             Enrollments e ON c.course_id = e.course_id
    ->     ) = (
    ->         SELECT
    ->             COUNT(DISTINCT e.course_id)
    ->         FROM
    ->             Enrollments e
    ->         WHERE
    ->             e.student_id = s.student_id
    ->     );
Empty set (0.00 sec)
```

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

```
mysql> SELECT
    ->     t.teacher_id,
    ->     t.first_name,
    ->     t.last_name
    -> FROM
    ->     Teacher t
    -> WHERE
    ->     t.teacher_id NOT IN (
    ->         SELECT DISTINCT
    ->             c.teacher_id
    ->         FROM
    ->             Courses c
    ->     );
+------------+------------+-----------+
| teacher_id | first_name | last_name |
+------------+------------+-----------+
|          1 | Professor  | Johnson   |
+------------+------------+-----------+
1 row in set (0.00 sec)
```

Calculate 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
    ->         TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age
    ->     FROM
    ->         Students
    -> ) AS student_ages;
+-------------+
| average_age |
+-------------+
|     28.4545 |
+-------------+
1 row in set (0.00 sec)
```

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

```
mysql> SELECT
    ->     c.course_id,
    ->     c.course_name
    -> FROM
    ->     Courses c
    -> WHERE
    ->     NOT EXISTS (
    ->         SELECT 1
    ->         FROM
    ->             Enrollments e
    ->         WHERE
    ->             e.course_id = c.course_id
    ->     );
+-----------+------------------+
| course_id | course_name      |
+-----------+------------------+
|       103 | Computer Science |
+-----------+------------------+
1 row in set (0.00 sec)
```

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
    -> JOIN
    ->     Payments p ON s.student_id = p.student_id
    -> GROUP BY
    ->     s.student_id, s.first_name, s.last_name
    -> HAVING
    ->     COUNT(p.payment_id) > 1;
Empty set (0.00 sec)
```

Write SQL query to calculate total payments made by each student. Join Students table with 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,
    ->     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 |
+------------+------------+-----------+----------------+
|          1 | John       | Doe       |        5000.00 |
|          2 | Jane       | Smith     |        7500.50 |
|          3 | Michael    | Johnson   |        6000.00 |
|          4 | Emily      | Williams  |        8000.75 |
|          5 | David      | Brown     |        3500.25 |
|          6 | Sophia     | Miller    |        1900.50 |
|          7 | Daniel     | Jones     |         900.00 |
|          8 | Olivia     | Davis     |        2700.25 |
|          9 | Liam       | Taylor    |        1550.00 |
|         10 | Ava        | Moore     |        1000.00 |
|         11 | Johny      | Day       |           NULL |
+------------+------------+-----------+----------------+
11 rows in set (0.00 sec)
```

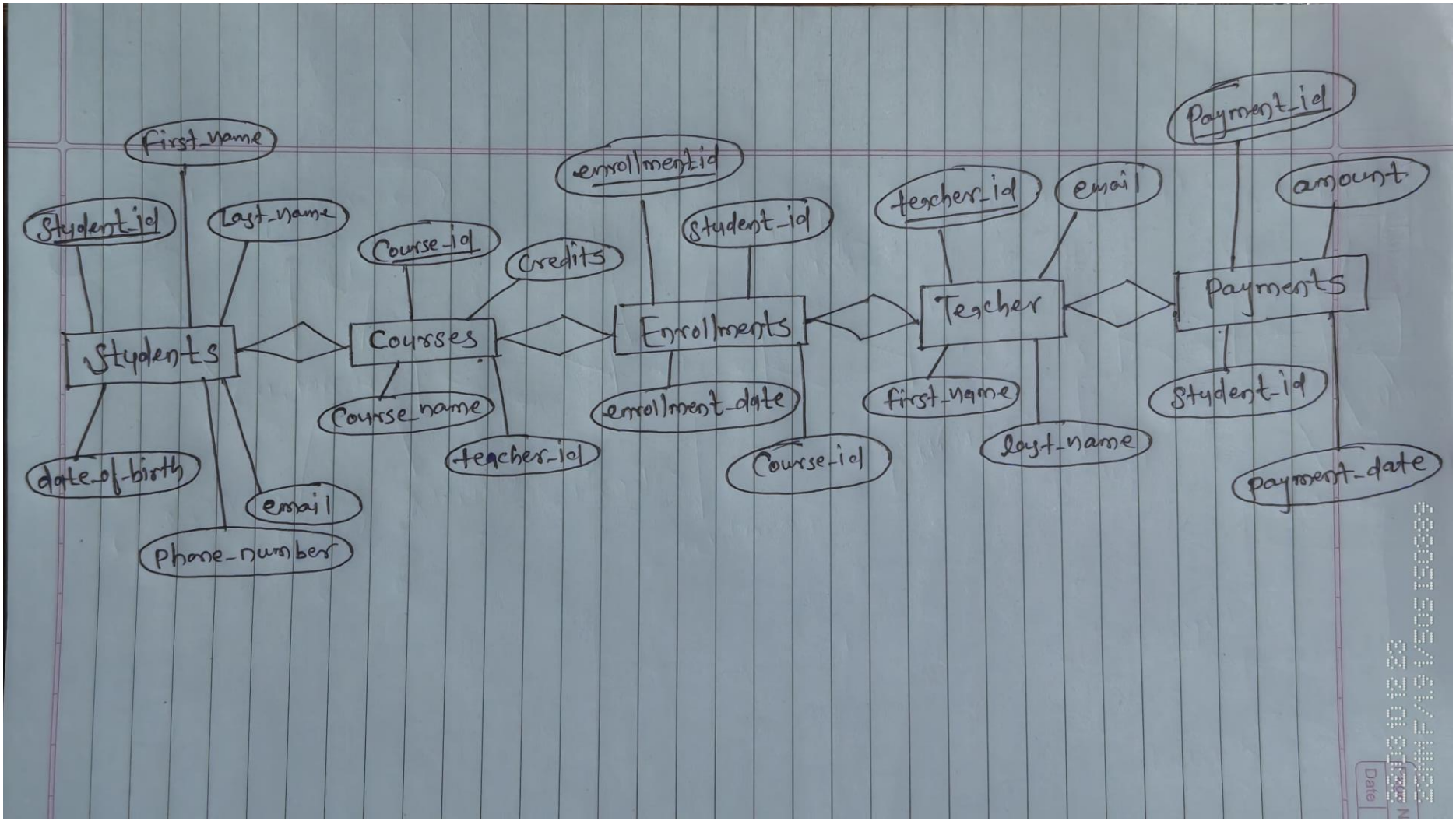
Retrieve a list of course names along with count of students enrolled in each course. Use join operations between Courses table and Enrollments table and group by to count enrollments.

```
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_name;
+--------------------+--------------------------+
| course_name        | enrolled_students_count |
+--------------------+--------------------------+
| Mathematics        |                        1 |
| History            |                        1 |
| Computer Science   |                        0 |
| Physics            |                        1 |
| English Literature |                        1 |
| Chemistry          |                        1 |
| Art History        |                        1 |
| Economics          |                        1 |
| Biology            |                        1 |
| Psychology         |                        1 |
+--------------------+--------------------------+
10 rows in set (0.00 sec)
```

Calculate average payment amount made by students. Use join operations between Students table and Payments table and group by to calculate average.

```
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 |
+------------+------------+-----------+------------------------+
|          1 | John       | Doe       |            5000.000000 |
|          2 | Jane       | Smith     |            7500.500000 |
|          3 | Michael    | Johnson   |            6000.000000 |
|          4 | Emily      | Williams  |            8000.750000 |
|          5 | David      | Brown     |            3500.250000 |
|          6 | Sophia     | Miller    |            1900.500000 |
|          7 | Daniel     | Jones     |             900.000000 |
|          8 | Olivia     | Davis     |            2700.250000 |
|          9 | Liam       | Taylor    |            1550.000000 |
|         10 | Ava        | Moore     |            1000.000000 |
|         11 | Johny      | Day       |                    NULL |
+------------+------------+-----------+------------------------+
11 rows in set (0.00 sec)
```

Entity Relationship Diagram


Students
Student_id
First_name
Last_name
date_of_birth
email
Phone_number
Courses
Course_id
Course_name
Credits
Teacher_id
Enrollments
enrollment_id
Student_id
enrollment_date
Course_id
Teacher
teacher_id
email
first_name
last_name
Payments
Payment_id
amount
Student_id
payment_date