Assignment 2

Task 1. Database Design:

1. Create the database named "SISDB"

Soln. CREATE DATABASE SISDB;

USE 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

Soln. schema for the 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)
);
```

schema for the 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)
);
```

schema for the Enrollments table

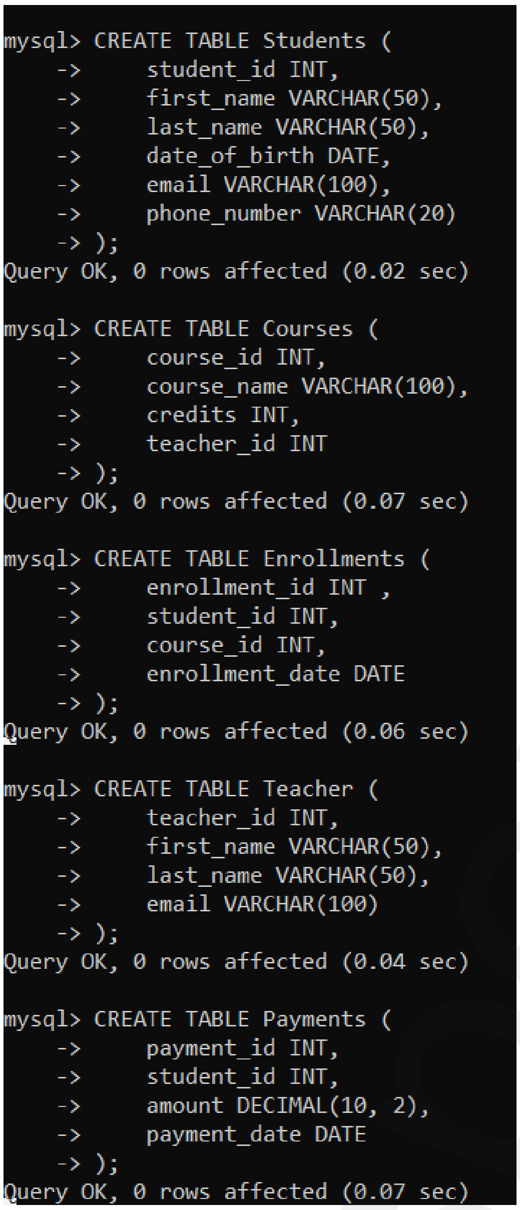
```
CREATE TABLE Enrollments (
    enrollment_id INT PRIMARY KEY,
    student_id INT,
    course_id INT,
    enrollment_date DATE,
    FOREIGN KEY (student_id) REFERENCES Students(student_id),
    FOREIGN KEY (course_id) REFERENCES Courses(course_id)
);
```

schema for the Teacher table

```
CREATE TABLE Teacher (
    teacher_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100)
);
```

schema for the Payments table

```
CREATE TABLE Payments (
    payment_id INT,
    student_id INT,
    amount DECIMAL(10, 2),
    payment_date DATE
);
```



```
mysql> CREATE TABLE Students (
    ->     student_id INT,
    ->     first_name VARCHAR(50),
    ->     last_name VARCHAR(50),
    ->     date_of_birth DATE,
    ->     email VARCHAR(100),
    ->     phone_number VARCHAR(20)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Courses (
    ->     course_id INT,
    ->     course_name VARCHAR(100),
    ->     credits INT,
    ->     teacher_id INT
    -> );
Query OK, 0 rows affected (0.07 sec)

mysql> CREATE TABLE Enrollments (
    ->     enrollment_id INT ,
    ->     student_id INT,
    ->     course_id INT,
    ->     enrollment_date DATE
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE TABLE Teacher (
    ->     teacher_id INT,
    ->     first_name VARCHAR(50),
    ->     last_name VARCHAR(50),
    ->     email VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE Payments (
    ->     payment_id INT,
    ->     student_id INT,
    ->     amount DECIMAL(10, 2),
    ->     payment_date DATE
    -> );
Query OK, 0 rows affected (0.07 sec)
```

3. **Create appropriate Primary Key and Foreign Key constraints for referential integrity.**

Soln. Primary Key constraint to Students table

ALTER TABLE Students

ADD CONSTRAINT PK_Students PRIMARY KEY (student_id);

Primary Key constraint to Courses table

ALTER TABLE Courses

ADD CONSTRAINT PK_Courses PRIMARY KEY (course_id);

Primary Key constraint to Enrollments table

ALTER TABLE Enrollments

ADD CONSTRAINT PK_Enrollments PRIMARY KEY (enrollment_id);

Foreign Key constraint to Enrollments table referencing Students table

ALTER TABLE Enrollments

ADD CONSTRAINT FK_Enrollments_Students FOREIGN KEY (student_id) REFERENCES Students(student_id);

Foreign Key constraint to Enrollments table referencing Courses table

ALTER TABLE Enrollments

ADD CONSTRAINT FK_Enrollments_Courses FOREIGN KEY (course_id) REFERENCES Courses(course_id);

Primary Key constraint to Teacher table

ALTER TABLE Teacher

ADD CONSTRAINT PK_Teacher PRIMARY KEY (teacher_id);

Foreign Key constraint to Courses table referencing Teacher table

ALTER TABLE Courses

ADD CONSTRAINT FK_Courses_Teacher FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id);

Primary Key constraint to Payments table

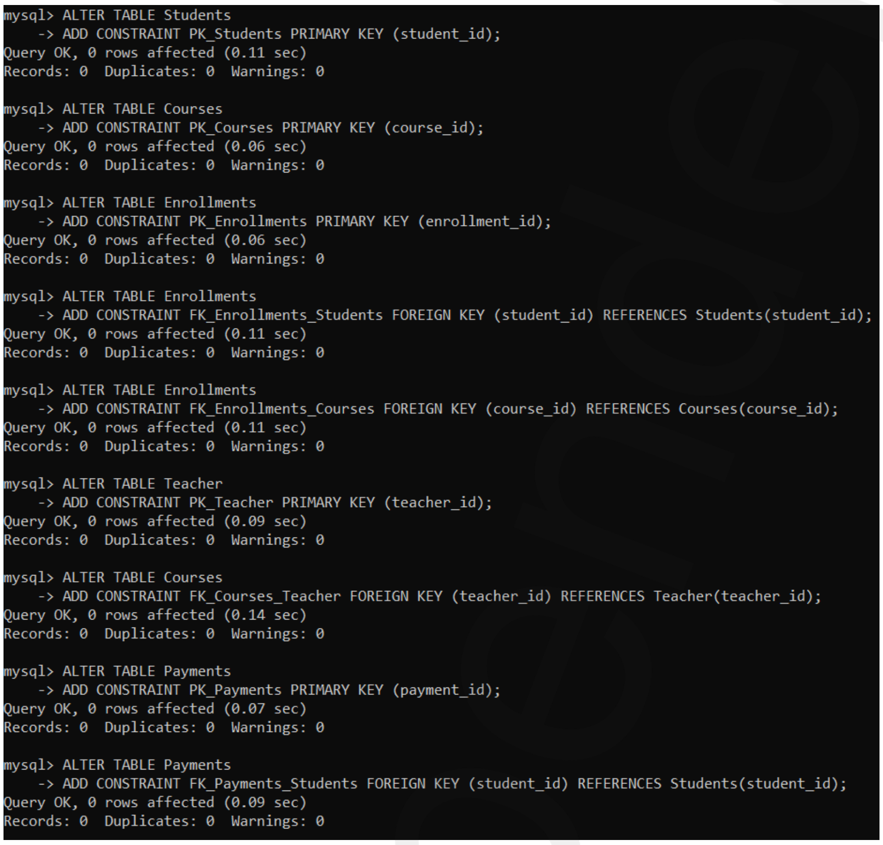
ALTER TABLE Payments

ADD CONSTRAINT PK_Payments PRIMARY KEY (payment_id);

Foreign Key constraint to Payments table referencing Students table

ALTER TABLE Payments

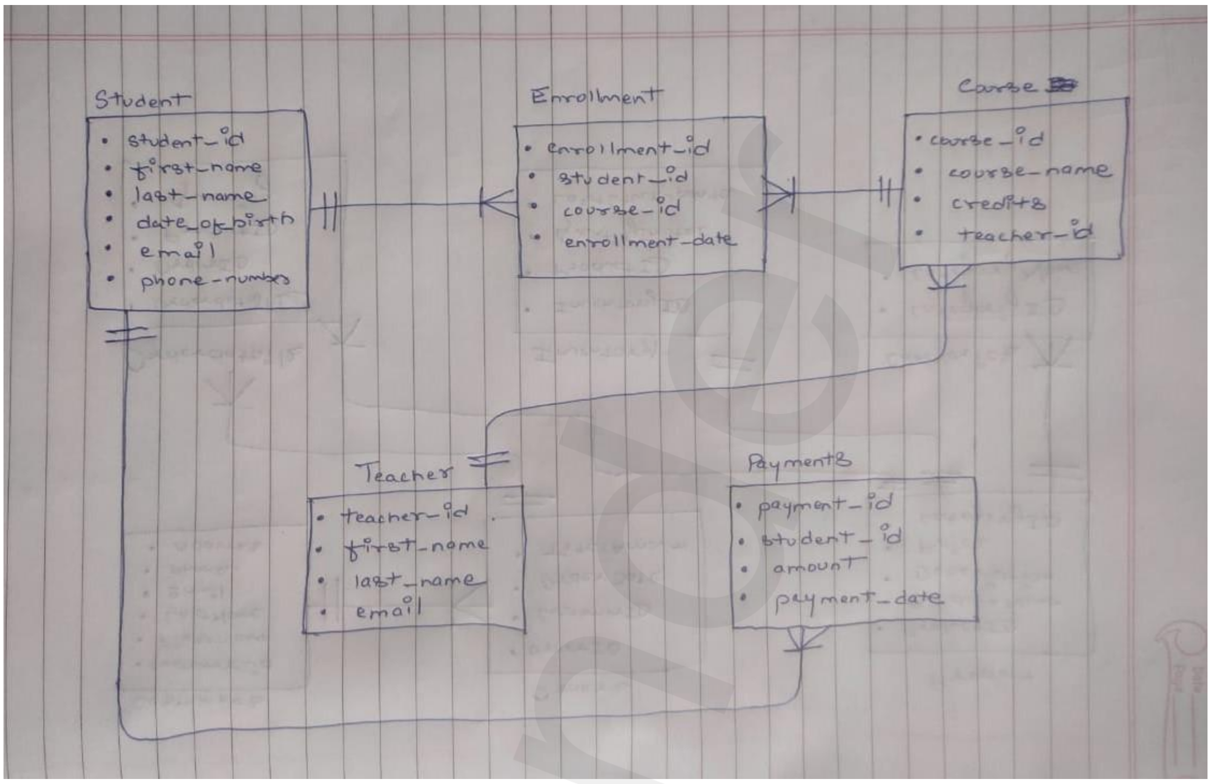
ADD CONSTRAINT FK_Payments_Students FOREIGN KEY (student_id) REFERENCES Students(student_id);



```
mysql> ALTER TABLE Students
    -> ADD CONSTRAINT PK_Students PRIMARY KEY (student_id);
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Courses
    -> ADD CONSTRAINT PK_Courses PRIMARY KEY (course_id);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Enrollments
    -> ADD CONSTRAINT PK_Enrollments PRIMARY KEY (enrollment_id);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Enrollments
    -> ADD CONSTRAINT FK_Enrollments_Students FOREIGN KEY (student_id) REFERENCES Students(student_id);
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Enrollments
    -> ADD CONSTRAINT FK_Enrollments_Courses FOREIGN KEY (course_id) REFERENCES Courses(course_id);
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Teacher
    -> ADD CONSTRAINT PK_Teacher PRIMARY KEY (teacher_id);
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Courses
    -> ADD CONSTRAINT FK_Courses_Teacher FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id);
Query OK, 0 rows affected (0.14 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Payments
    -> ADD CONSTRAINT PK_Payments PRIMARY KEY (payment_id);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Payments
    -> ADD CONSTRAINT FK_Payments_Students FOREIGN KEY (student_id) REFERENCES Students(student_id);
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

4. Create ER diagram for the above tables.

Ans.



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

Soln. Insert into Students table

INSERT INTO Students VALUES

(2, 'Jane', 'Smith', '1992-03-15', 'jane.smith@email.com', '987-654-3210'),

(3, 'Bob', 'Johnson', '1991-07-20', 'bob.johnson@email.com', '555-123-4567'),

(4, 'Emily', 'Williams', '1993-05-10', 'emily.williams@email.com', '789-321-6540'),

(5, 'Alex', 'Brown', '1994-11-28', 'alex.brown@email.com', '333-444-5555'),

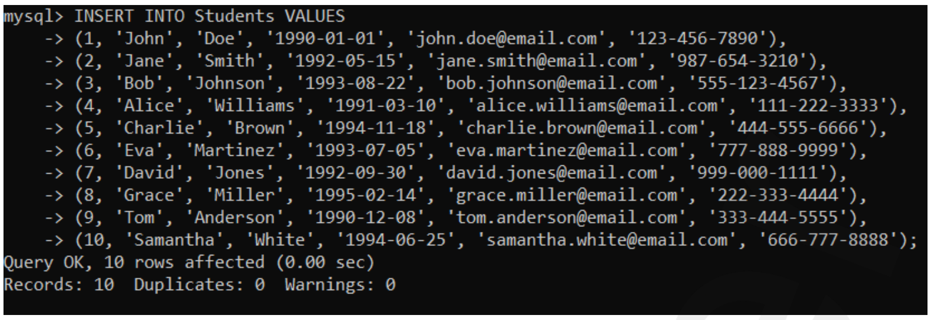
(6, 'Grace', 'Miller', '1990-09-08', 'grace.miller@email.com', '777-888-9999'),

(7, 'Daniel', 'Taylor', '1992-12-03', 'daniel.taylor@email.com', '111-222-3333'),

(8, 'Sophia', 'Anderson', '1995-02-14', 'sophia.anderson@email.com', '666-777-8888'),

(9, 'Michael', 'Clark', '1993-04-18', 'michael.clark@email.com', '444-555-6666'),

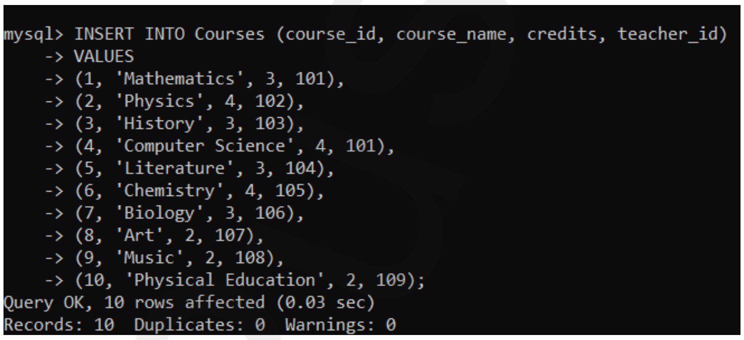
(10, 'Olivia', 'Davis', '1991-06-25', 'olivia.davis@email.com', '222-333-4444');



```
mysql> INSERT INTO Students VALUES
    -> (1, 'John', 'Doe', '1990-01-01', 'john.doe@email.com', '123-456-7890'),
    -> (2, 'Jane', 'Smith', '1992-05-15', 'jane.smith@email.com', '987-654-3210'),
    -> (3, 'Bob', 'Johnson', '1993-08-22', 'bob.johnson@email.com', '555-123-4567'),
    -> (4, 'Alice', 'Williams', '1991-03-10', 'alice.williams@email.com', '111-222-3333'),
    -> (5, 'Charlie', 'Brown', '1994-11-18', 'charlie.brown@email.com', '444-555-6666'),
    -> (6, 'Eva', 'Martinez', '1993-07-05', 'eva.martinez@email.com', '777-888-9999'),
    -> (7, 'David', 'Jones', '1992-09-30', 'david.jones@email.com', '999-000-1111'),
    -> (8, 'Grace', 'Miller', '1995-02-14', 'grace.miller@email.com', '222-333-4444'),
    -> (9, 'Tom', 'Anderson', '1990-12-08', 'tom.anderson@email.com', '333-444-5555'),
    -> (10, 'Samantha', 'White', '1994-06-25', 'samantha.white@email.com', '666-777-8888');
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

Insert 10 sample records into the "Courses" table

```
INSERT INTO Courses (course_id, course_name, credits, teacher_id)

VALUES

(1, 'Mathematics', 3, 101),

(2, 'Physics', 4, 102),

(3, 'History', 3, 103),

(4, 'Computer Science', 4, 101),

(5, 'Literature', 3, 104),

(6, 'Chemistry', 4, 105),

(7, 'Biology', 3, 106),

(8, 'Art', 2, 107),

(9, 'Music', 2, 108),

(10, 'Physical Education', 2, 109);
```



```
mysql> INSERT INTO Courses (course_id, course_name, credits, teacher_id)
    -> VALUES
    -> (1, 'Mathematics', 3, 101),
    -> (2, 'Physics', 4, 102),
    -> (3, 'History', 3, 103),
    -> (4, 'Computer Science', 4, 101),
    -> (5, 'Literature', 3, 104),
    -> (6, 'Chemistry', 4, 105),
    -> (7, 'Biology', 3, 106),
    -> (8, 'Art', 2, 107),
    -> (9, 'Music', 2, 108),
    -> (10, 'Physical Education', 2, 109);
Query OK, 10 rows affected (0.03 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

Insert at least 10 sample records into the " Enrollments" table

```
INSERT INTO Enrollments VALUES

(1, 1, 1, '2023-01-15'),
```

(2, 2, 3, '2023-02-20'),

(3, 3, 2, '2023-03-10'),

(4, 4, 1, '2023-04-05'),

(5, 5, 4, '2023-05-12'),

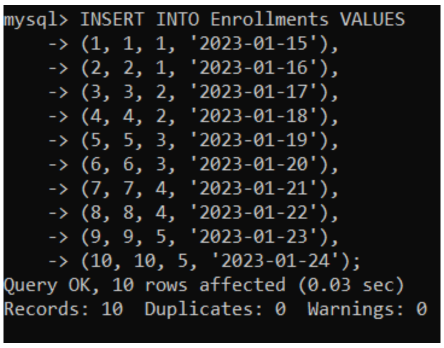
(6, 6, 3, '2023-06-18'),

(7, 7, 2, '2023-07-25'),

(8, 8, 1, '2023-08-30'),

(9, 9, 4, '2023-09-05'),

(10, 10, 3, '2023-10-10');



```
mysql> INSERT INTO Enrollments VALUES
    -> (1, 1, 1, '2023-01-15'),
    -> (2, 2, 1, '2023-01-16'),
    -> (3, 3, 2, '2023-01-17'),
    -> (4, 4, 2, '2023-01-18'),
    -> (5, 5, 3, '2023-01-19'),
    -> (6, 6, 3, '2023-01-20'),
    -> (7, 7, 4, '2023-01-21'),
    -> (8, 8, 4, '2023-01-22'),
    -> (9, 9, 5, '2023-01-23'),
    -> (10, 10, 5, '2023-01-24');
Query OK, 10 rows affected (0.03 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

Insert 10 sample records into the "Teacher" table

INSERT INTO Teacher (teacher_id, first_name, last_name, email)

VALUES

(101, 'Jane', 'Smith', 'jane.smith@email.com'),

(102, 'John', 'Doe', 'john.doe@email.com'),

(103, 'Mary', 'Johnson', 'mary.johnson@email.com'),

(104, 'Robert', 'Williams', 'robert.williams@email.com'),

(105, 'Emily', 'Davis', 'emily.davis@email.com'),

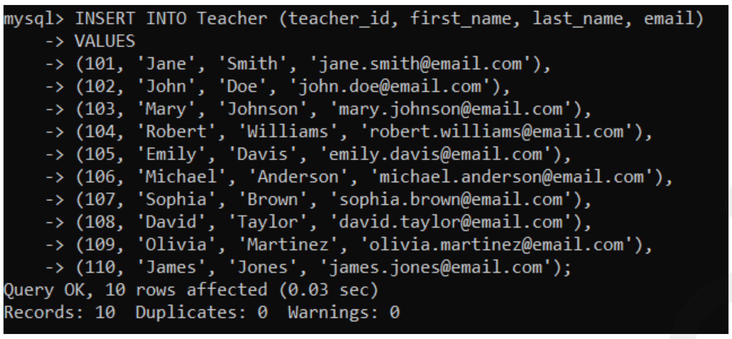
(106, 'Michael', 'Anderson', 'michael.anderson@email.com'),

(107, 'Sophia', 'Brown', 'sophia.brown@email.com'),

(108, 'David', 'Taylor', 'david.taylor@email.com'),

(109, 'Olivia', 'Martinez', 'olivia.martinez@email.com'),

(110, 'James', 'Jones', 'james.jones@email.com');


```
mysql> INSERT INTO Teacher (teacher_id, first_name, last_name, email)
    -> VALUES
    -> (101, 'Jane', 'Smith', 'jane.smith@email.com'),
    -> (102, 'John', 'Doe', 'john.doe@email.com'),
    -> (103, 'Mary', 'Johnson', 'mary.johnson@email.com'),
    -> (104, 'Robert', 'Williams', 'robert.williams@email.com'),
    -> (105, 'Emily', 'Davis', 'emily.davis@email.com'),
    -> (106, 'Michael', 'Anderson', 'michael.anderson@email.com'),
    -> (107, 'Sophia', 'Brown', 'sophia.brown@email.com'),
    -> (108, 'David', 'Taylor', 'david.taylor@email.com'),
    -> (109, 'Olivia', 'Martinez', 'olivia.martinez@email.com'),
    -> (110, 'James', 'Jones', 'james.jones@email.com');
Query OK, 10 rows affected (0.03 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

Insert at least 10 sample records into the "Payments" table

INSERT INTO Payments VALUES

(1, 1, 100.00, '2023-02-01'),

(2, 2, 150.00, '2023-02-02'),

(3, 3, 200.00, '2023-02-03'),

(4, 4, 120.00, '2023-02-04'),

(5, 5, 80.00, '2023-02-05'),

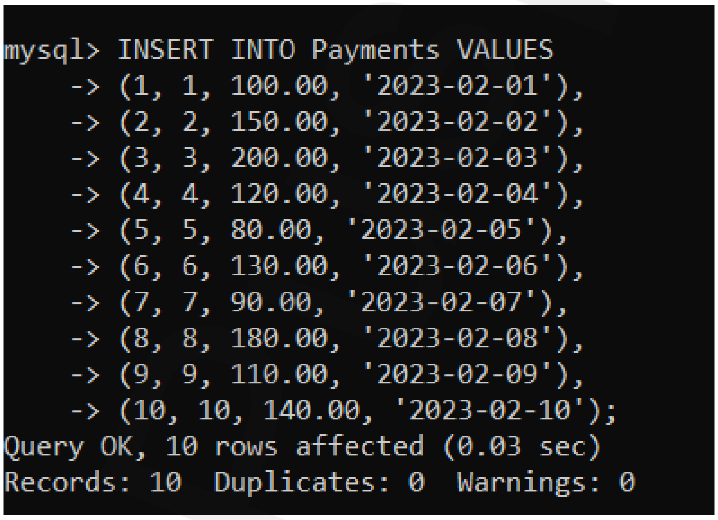
(6, 6, 130.00, '2023-02-06'),

(7, 7, 90.00, '2023-02-07'),

(8, 8, 180.00, '2023-02-08'),

(9, 9, 110.00, '2023-02-09'),

(10, 10, 140.00, '2023-02-10');


```
mysql> INSERT INTO Payments VALUES
    -> (1, 1, 100.00, '2023-02-01'),
    -> (2, 2, 150.00, '2023-02-02'),
    -> (3, 3, 200.00, '2023-02-03'),
    -> (4, 4, 120.00, '2023-02-04'),
    -> (5, 5, 80.00, '2023-02-05'),
    -> (6, 6, 130.00, '2023-02-06'),
    -> (7, 7, 90.00, '2023-02-07'),
    -> (8, 8, 180.00, '2023-02-08'),
    -> (9, 9, 110.00, '2023-02-09'),
    -> (10, 10, 140.00, '2023-02-10');
Query OK, 10 rows affected (0.03 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

Tasks 2: Select, where, Between, AND, LIKE:

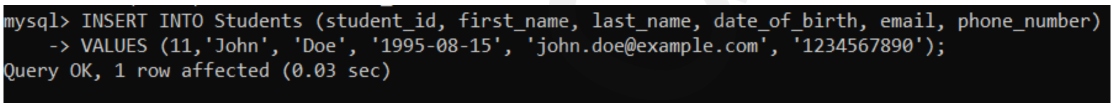
1. Write an SQL query to insert a new student into the "Students" table with the following details:

- First Name: John
- Last Name: Doe
- Date of Birth: 1995-08-15
- Email: john.doe@example.com
- Phone Number: 1234567890

Ans. Insert a new student into the "Students" table

INSERT INTO Students (first_name, last_name, date_of_birth, email, phone_number)

VALUES ('John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');



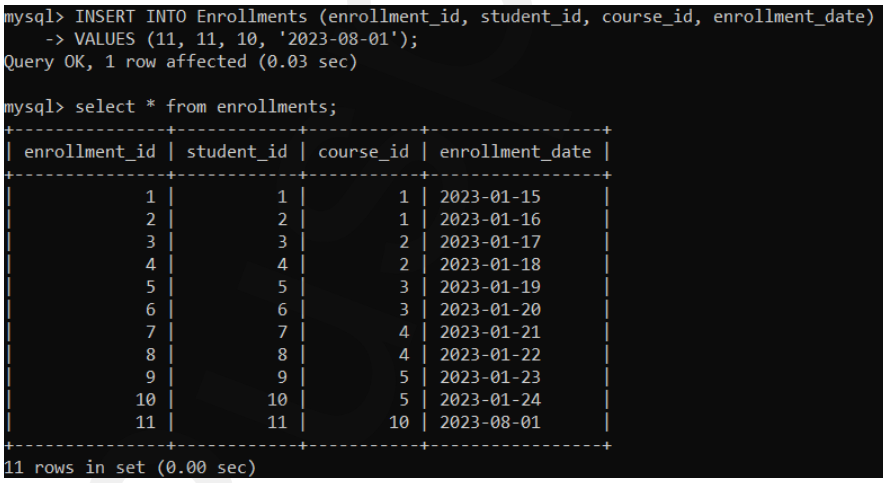
```
mysql> INSERT INTO Students (student_id, first_name, last_name, date_of_birth, email, phone_number)
    -> VALUES (11,'John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
Query OK, 1 row affected (0.03 sec)
```

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.

Ans. Enroll a student in a course

INSERT INTO Enrollments (student_id, course_id, enrollment_date)

VALUES (existing_student_id, existing_course_id, '2023-08-01');



```
mysql> INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date)
    -> VALUES (11, 11, 10, '2023-08-01');
Query OK, 1 row affected (0.03 sec)

mysql> select * from enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|             1 |          1 |         1 | 2023-01-15      |
|             2 |          2 |         1 | 2023-01-16      |
|             3 |          3 |         2 | 2023-01-17      |
|             4 |          4 |         2 | 2023-01-18      |
|             5 |          5 |         3 | 2023-01-19      |
|             6 |          6 |         3 | 2023-01-20      |
|             7 |          7 |         4 | 2023-01-21      |
|             8 |          8 |         4 | 2023-01-22      |
|             9 |          9 |         5 | 2023-01-23      |
|            10 |         10 |         5 | 2023-01-24      |
|            11 |         11 |        10 | 2023-08-01      |
+---------------+------------+-----------+-----------------+
11 rows in set (0.00 sec)
```

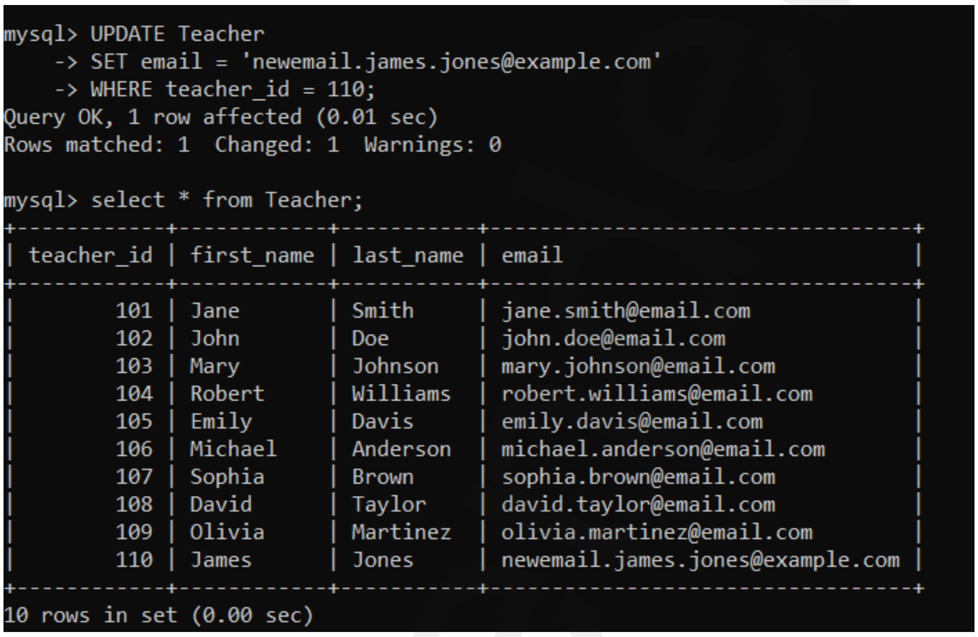
3.Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address. 4. Write an SQL query to delete a specific enrollment record from the "Enrollments" table. Select an enrollment record based on the student and course.

Ans. Update the email address of a specific teacher

UPDATE Teacher

SET email = 'newemail@example.com'

WHERE teacher_id = specific_teacher_id;



```
mysql> UPDATE Teacher
    -> SET email = 'newemail.james.jones@example.com'
    -> WHERE teacher_id = 110;
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                            |
+------------+------------+-----------+----------------------------------+
|        101 | Jane       | Smith     | jane.smith@email.com             |
|        102 | John       | Doe       | john.doe@email.com               |
|        103 | Mary       | Johnson   | mary.johnson@email.com           |
|        104 | Robert     | Williams  | robert.williams@email.com        |
|        105 | Emily      | Davis     | emily.davis@email.com            |
|        106 | Michael    | Anderson  | michael.anderson@email.com       |
|        107 | Sophia     | Brown     | sophia.brown@email.com           |
|        108 | David      | Taylor    | david.taylor@email.com           |
|        109 | Olivia     | Martinez  | olivia.martinez@email.com        |
|        110 | James      | Jones     | newemail.james.jones@example.com |
+------------+------------+-----------+----------------------------------+
10 rows in set (0.00 sec)
```

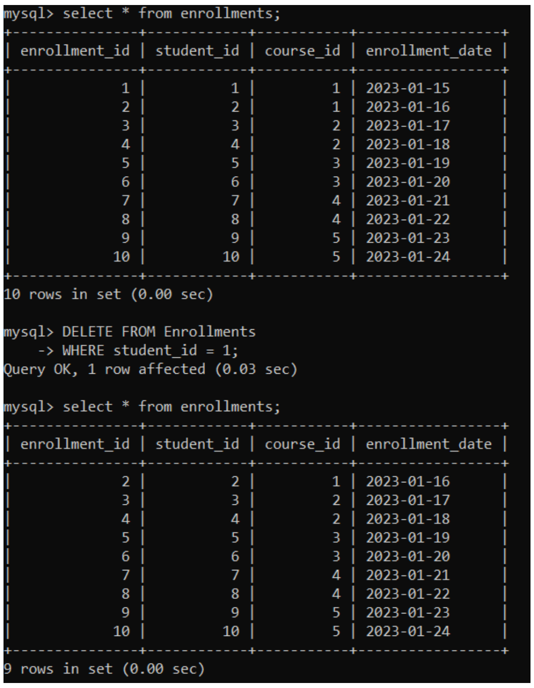
4. Write an SQL query to delete a specific enrollment record from the "Enrollments" table. Select an enrollment record based on the student and course

Ans. Delete a specific enrollment record

DELETE FROM Enrollments

WHERE student_id = specific_student_id

  AND course_id = specific_course_id;



```
mysql> select * from enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|             1 |          1 |         1 | 2023-01-15      |
|             2 |          2 |         1 | 2023-01-16      |
|             3 |          3 |         2 | 2023-01-17      |
|             4 |          4 |         2 | 2023-01-18      |
|             5 |          5 |         3 | 2023-01-19      |
|             6 |          6 |         3 | 2023-01-20      |
|             7 |          7 |         4 | 2023-01-21      |
|             8 |          8 |         4 | 2023-01-22      |
|             9 |          9 |         5 | 2023-01-23      |
|            10 |         10 |         5 | 2023-01-24      |
+---------------+------------+-----------+-----------------+
10 rows in set (0.00 sec)

mysql> DELETE FROM Enrollments
    -> WHERE student_id = 1;
Query OK, 1 row affected (0.03 sec)

mysql> select * from enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|             2 |          2 |         1 | 2023-01-16      |
|             3 |          3 |         2 | 2023-01-17      |
|             4 |          4 |         2 | 2023-01-18      |
|             5 |          5 |         3 | 2023-01-19      |
|             6 |          6 |         3 | 2023-01-20      |
|             7 |          7 |         4 | 2023-01-21      |
|             8 |          8 |         4 | 2023-01-22      |
|             9 |          9 |         5 | 2023-01-23      |
|            10 |         10 |         5 | 2023-01-24      |
+---------------+------------+-----------+-----------------+
9 rows in set (0.00 sec)
```

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

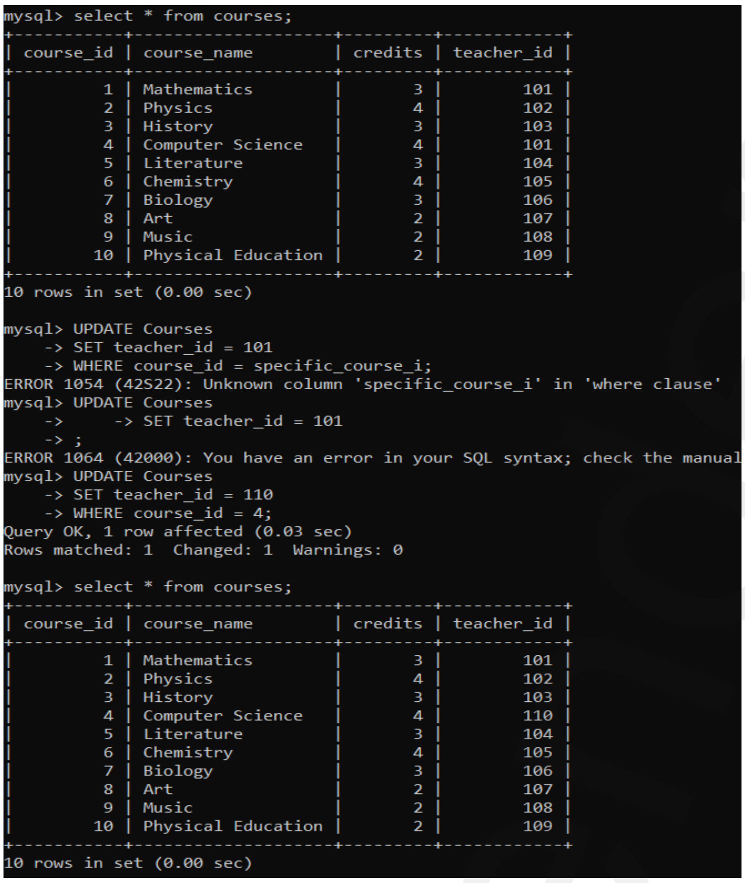
Ans. Delete a specific student and their enrollments (maintaining referential integrity)

DELETE FROM Enrollments

WHERE student_id = specific_student_id;

DELETE FROM Students

WHERE student_id = specific_student_id;



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

mysql> UPDATE Courses
    -> SET teacher_id = 101
    -> WHERE course_id = specific_course_i;
ERROR 1054 (42S22): Unknown column 'specific_course_i' in 'where clause'
mysql> UPDATE Courses
    ->     -> SET teacher_id = 101
    -> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual
mysql> UPDATE Courses
    -> SET teacher_id = 110
    -> WHERE course_id = 4;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

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

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

Ans. Delete enrollments of a specific student from the "Enrollments" table

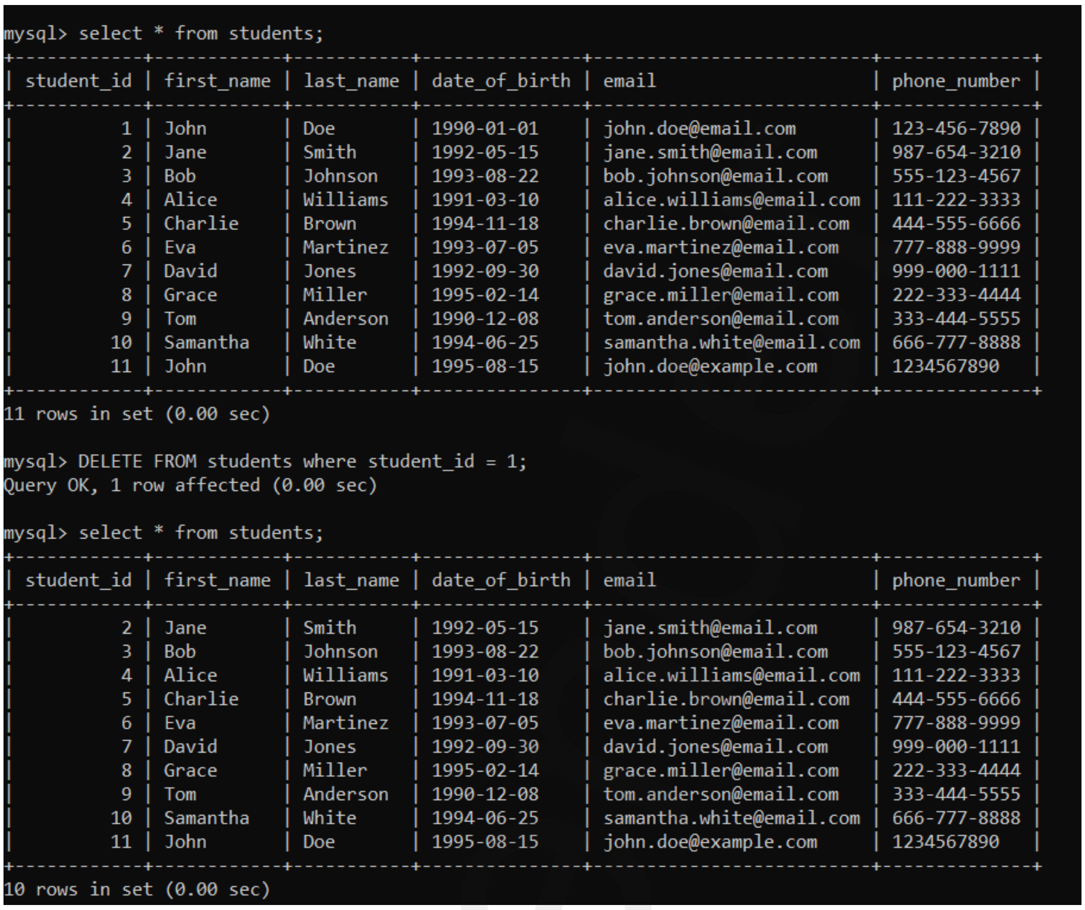
DELETE FROM Enrollments

WHERE student_id = specific_student_id;

Delete the specific student from the "Students" table

DELETE FROM Students

WHERE student_id = specific_student_id;


```
mysql> select * from students;
+------------+------------+-----------+---------------+--------------------------+--------------+
| student_id | first_name | last_name | date_of_birth | email                    | phone_number |
+------------+------------+-----------+---------------+--------------------------+--------------+
|          1 | John       | Doe       | 1990-01-01    | john.doe@email.com       | 123-456-7890 |
|          2 | Jane       | Smith     | 1992-05-15    | jane.smith@email.com     | 987-654-3210 |
|          3 | Bob        | Johnson   | 1993-08-22    | bob.johnson@email.com    | 555-123-4567 |
|          4 | Alice      | Williams  | 1991-03-10    | alice.williams@email.com | 111-222-3333 |
|          5 | Charlie    | Brown     | 1994-11-18    | charlie.brown@email.com  | 444-555-6666 |
|          6 | Eva        | Martinez  | 1993-07-05    | eva.martinez@email.com   | 777-888-9999 |
|          7 | David      | Jones     | 1992-09-30    | david.jones@email.com    | 999-000-1111 |
|          8 | Grace      | Miller    | 1995-02-14    | grace.miller@email.com   | 222-333-4444 |
|          9 | Tom        | Anderson  | 1990-12-08    | tom.anderson@email.com   | 333-444-5555 |
|         10 | Samantha   | White     | 1994-06-25    | samantha.white@email.com | 666-777-8888 |
|         11 | John       | Doe       | 1995-08-15    | john.doe@example.com     | 1234567890   |
+------------+------------+-----------+---------------+--------------------------+--------------+
11 rows in set (0.00 sec)

mysql> DELETE FROM students where student_id = 1;
Query OK, 1 row affected (0.00 sec)

mysql> select * from students;
+------------+------------+-----------+---------------+--------------------------+--------------+
| student_id | first_name | last_name | date_of_birth | email                    | phone_number |
+------------+------------+-----------+---------------+--------------------------+--------------+
|          2 | Jane       | Smith     | 1992-05-15    | jane.smith@email.com     | 987-654-3210 |
|          3 | Bob        | Johnson   | 1993-08-22    | bob.johnson@email.com    | 555-123-4567 |
|          4 | Alice      | Williams  | 1991-03-10    | alice.williams@email.com | 111-222-3333 |
|          5 | Charlie    | Brown     | 1994-11-18    | charlie.brown@email.com  | 444-555-6666 |
|          6 | Eva        | Martinez  | 1993-07-05    | eva.martinez@email.com   | 777-888-9999 |
|          7 | David      | Jones     | 1992-09-30    | david.jones@email.com    | 999-000-1111 |
|          8 | Grace      | Miller    | 1995-02-14    | grace.miller@email.com   | 222-333-4444 |
|          9 | Tom        | Anderson  | 1990-12-08    | tom.anderson@email.com   | 333-444-5555 |
|         10 | Samantha   | White     | 1994-06-25    | samantha.white@email.com | 666-777-8888 |
|         11 | John       | Doe       | 1995-08-15    | john.doe@example.com     | 1234567890   |
+------------+------------+-----------+---------------+--------------------------+--------------+
10 rows in set (0.00 sec)
```

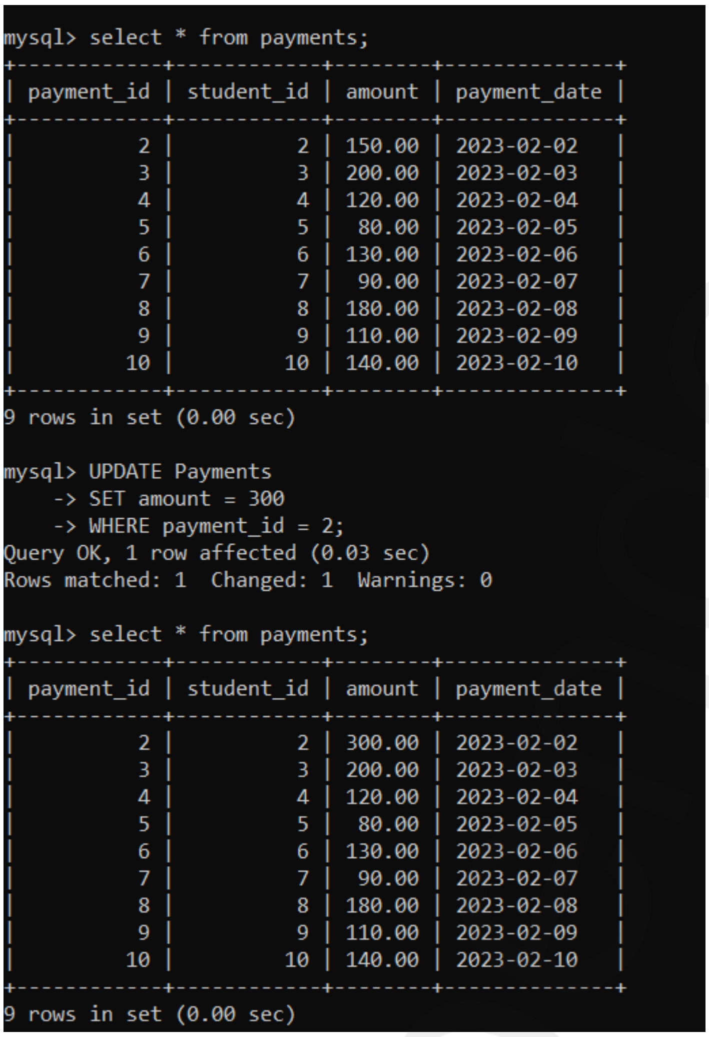
7. Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.

Ans. Update the payment amount for a specific payment record

UPDATE Payments

SET amount = new_amount

WHERE payment_id = specific_payment_id;



```
mysql> select * from payments;
+------------+------------+--------+--------------+
| payment_id | student_id | amount | payment_date |
+------------+------------+--------+--------------+
|          2 |          2 | 150.00 | 2023-02-02   |
|          3 |          3 | 200.00 | 2023-02-03   |
|          4 |          4 | 120.00 | 2023-02-04   |
|          5 |          5 |  80.00 | 2023-02-05   |
|          6 |          6 | 130.00 | 2023-02-06   |
|          7 |          7 |  90.00 | 2023-02-07   |
|          8 |          8 | 180.00 | 2023-02-08   |
|          9 |          9 | 110.00 | 2023-02-09   |
|         10 |         10 | 140.00 | 2023-02-10   |
+------------+------------+--------+--------------+
9 rows in set (0.00 sec)

mysql> UPDATE Payments
    -> SET amount = 300
    -> WHERE payment_id = 2;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from payments;
+------------+------------+--------+--------------+
| payment_id | student_id | amount | payment_date |
+------------+------------+--------+--------------+
|          2 |          2 | 300.00 | 2023-02-02   |
|          3 |          3 | 200.00 | 2023-02-03   |
|          4 |          4 | 120.00 | 2023-02-04   |
|          5 |          5 |  80.00 | 2023-02-05   |
|          6 |          6 | 130.00 | 2023-02-06   |
|          7 |          7 |  90.00 | 2023-02-07   |
|          8 |          8 | 180.00 | 2023-02-08   |
|          9 |          9 | 110.00 | 2023-02-09   |
|         10 |         10 | 140.00 | 2023-02-10   |
+------------+------------+--------+--------------+
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.

Ans. Calculate total payments made by a specific student

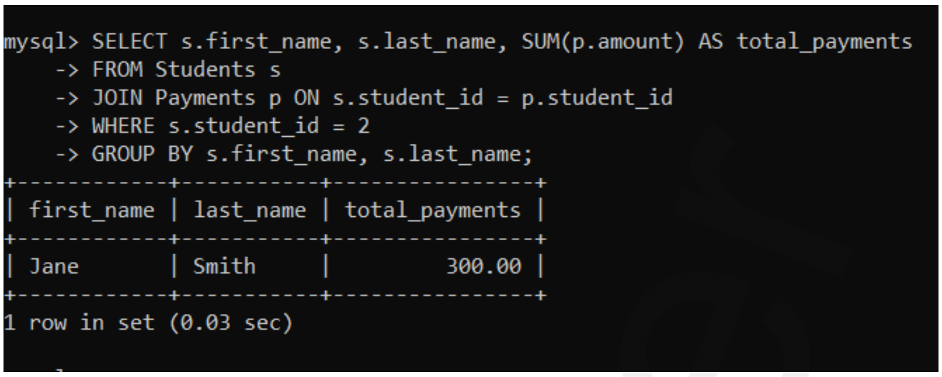
SELECT s.first_name, s.last_name, SUM(p.amount) AS total_payments

FROM Students s

JOIN Payments p ON s.student_id = p.student_id

WHERE s.student_id = 2

GROUP BY s.first_name, s.last_name;



```
mysql> SELECT 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 = 2
    -> GROUP BY s.first_name, s.last_name;
+------------+-----------+----------------+
| first_name | last_name | total_payments |
+------------+-----------+----------------+
| Jane       | Smith     |         300.00 |
+------------+-----------+----------------+
1 row in set (0.03 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.

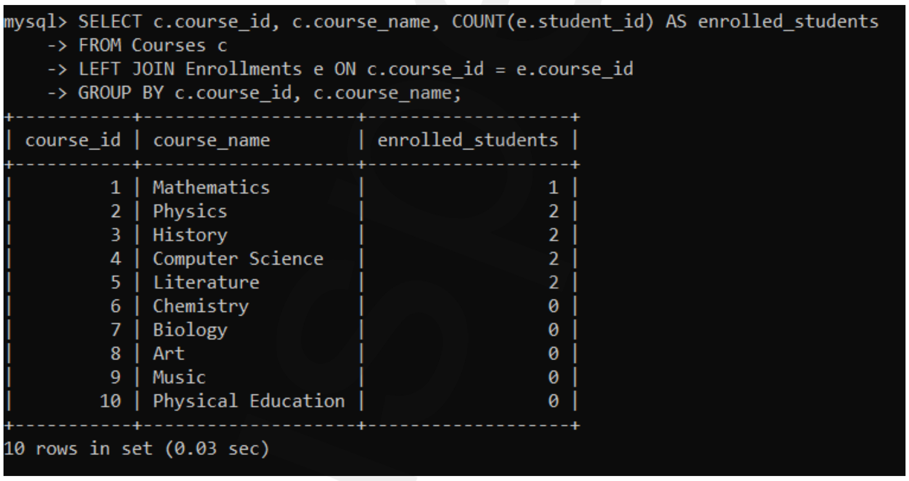
Ans. Retrieve a list of courses with the count of students enrolled in each course

SELECT c.course_id, c.course_name, COUNT(e.student_id) AS enrolled_students

FROM Courses c

LEFT JOIN Enrollments e ON c.course_id = e.course_id

GROUP BY c.course_id, c.course_name;



```
mysql> SELECT c.course_id, c.course_name, COUNT(e.student_id) AS enrolled_students
    -> 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 |
+-----------+-------------------+-------------------+
|         1 | Mathematics       |                 1 |
|         2 | Physics           |                 2 |
|         3 | History           |                 2 |
|         4 | Computer Science  |                 2 |
|         5 | Literature        |                 2 |
|         6 | Chemistry         |                 0 |
|         7 | Biology           |                 0 |
|         8 | Art               |                 0 |
|         9 | Music             |                 0 |
|        10 | Physical Education |                0 |
+-----------+-------------------+-------------------+
10 rows in set (0.03 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.

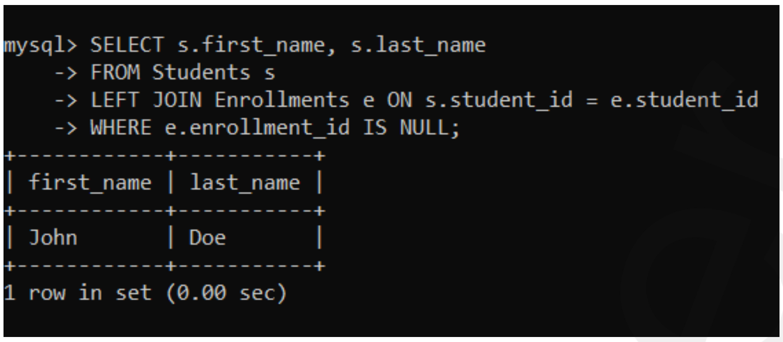
Ans. Find the names of students who have not enrolled in any course

SELECT s.first_name, s.last_name

FROM Students s

LEFT JOIN Enrollments e ON s.student_id = e.student_id

WHERE e.enrollment_id IS NULL;



```
mysql> SELECT s.first_name, s.last_name
    -> FROM Students s
    -> LEFT JOIN Enrollments e ON s.student_id = e.student_id
    -> WHERE e.enrollment_id IS NULL;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| John       | Doe       |
+------------+-----------+
1 row 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.

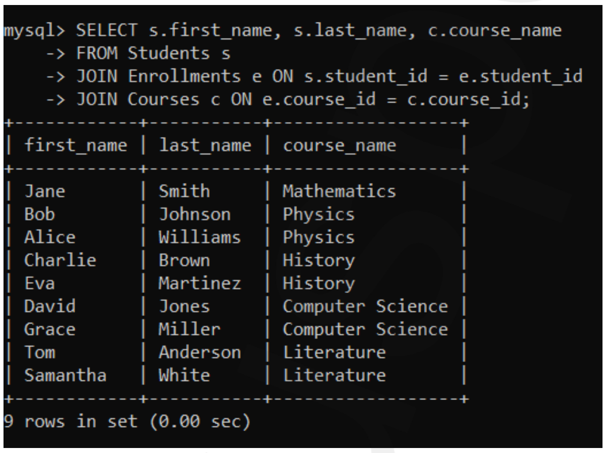
Ans. Retrieve the first name, last name of students, and the names of the courses they are enrolled in

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;



```
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      |
+------------+-----------+------------------+
| Jane       | Smith     | Mathematics      |
| Bob        | Johnson   | Physics          |
| Alice      | Williams  | Physics          |
| Charlie    | Brown     | History          |
| Eva        | Martinez  | History          |
| David      | Jones     | Computer Science |
| Grace      | Miller    | Computer Science |
| Tom        | Anderson  | Literature       |
| Samantha   | White     | Literature       |
+------------+-----------+------------------+
9 rows in set (0.00 sec)
```

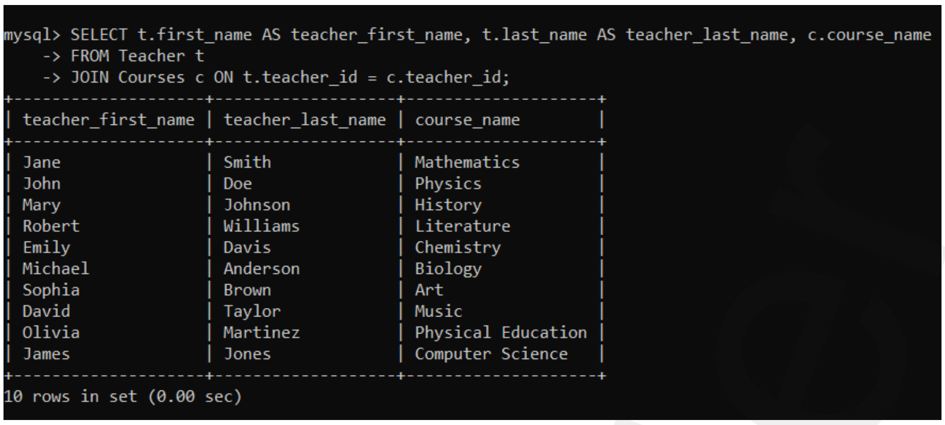
5. Create a query to list the names of teachers and the courses they are assigned to. Join the "Teacher" table with the "Courses" table.

Ans. List names of teachers and the courses they are assigned to

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;


```
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        |
| John               | Doe               | Physics            |
| Mary               | Johnson           | History            |
| Robert             | Williams          | Literature         |
| Emily              | Davis             | Chemistry          |
| Michael            | Anderson          | Biology            |
| Sophia             | Brown             | Art                |
| David              | Taylor            | Music              |
| Olivia             | Martinez          | Physical Education |
| James              | Jones             | Computer 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.

Ans. Retrieve a list of students and their enrollment dates for a specific course

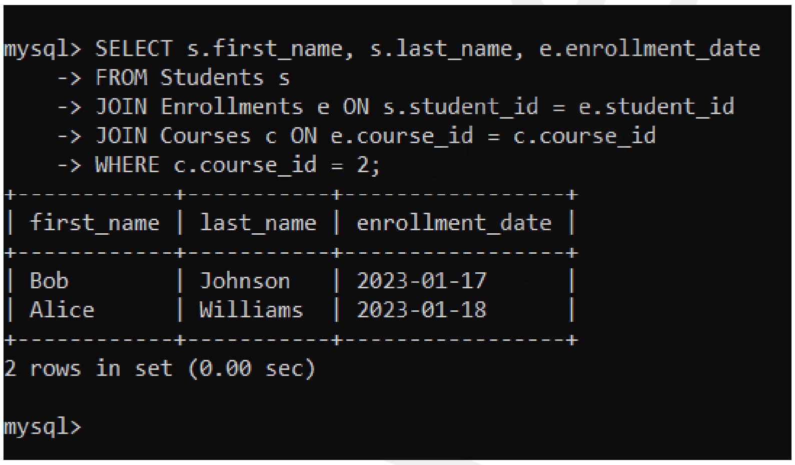
SELECT s.first_name, s.last_name, e.enrollment_date

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_id = specific_course_id;


```
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_id = 2;
+------------+-----------+-----------------+
| first_name | last_name | enrollment_date |
+------------+-----------+-----------------+
| Bob        | Johnson   | 2023-01-17      |
| Alice      | Williams  | 2023-01-18      |
+------------+-----------+-----------------+
2 rows in set (0.00 sec)

mysql>
```

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.

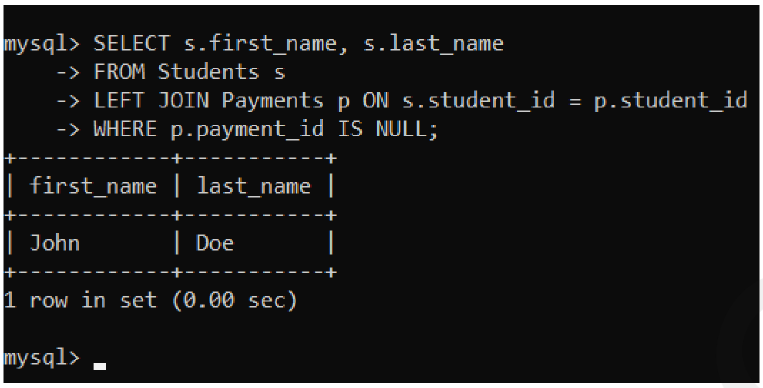
Ans. Find names of students who have not made any payments

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;



```
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 |
+------------+-----------+
| John       | Doe       |
+------------+-----------+
1 row in set (0.00 sec)

mysql> _
```

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.

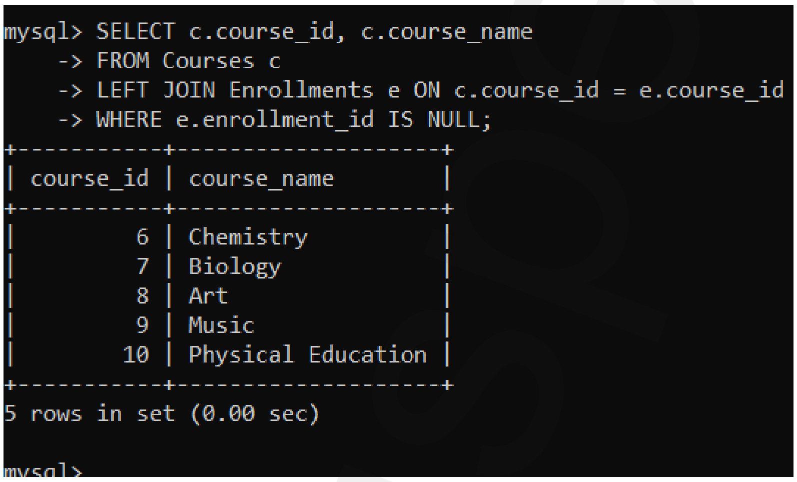
Ans. Identify courses that have no enrollments

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;



```
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        |
+-----------+--------------------+
|         6 | Chemistry          |
|         7 | Biology            |
|         8 | Art                |
|         9 | Music              |
|        10 | Physical Education |
+-----------+--------------------+
5 rows in set (0.00 sec)

mysql>
```

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.

Ans. Identify students who are enrolled in more than one course

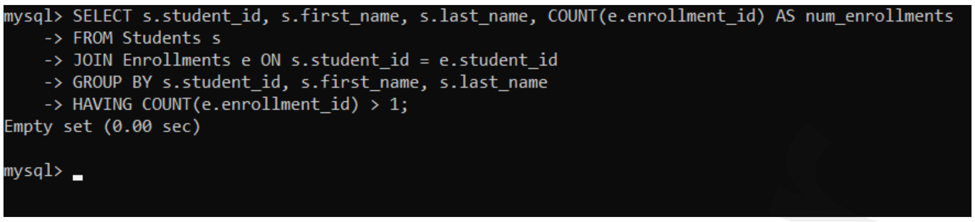
SELECT s.student_id, s.first_name, s.last_name, COUNT(e.enrollment_id) AS num_enrollments

FROM Students s

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;



```
mysql> SELECT s.student_id, s.first_name, s.last_name, COUNT(e.enrollment_id) AS num_enrollments
    -> FROM Students s
    -> 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;
Empty set (0.00 sec)

mysql> _
```

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

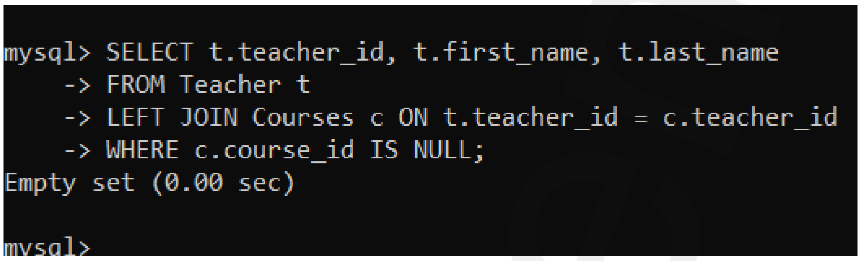
Ans. Find teachers who are not assigned to any courses

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;



```
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;
Empty set (0.00 sec)

mysql>
```

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.

Ans.

SELECT course_id, AVG(num_students) AS average_students

FROM (

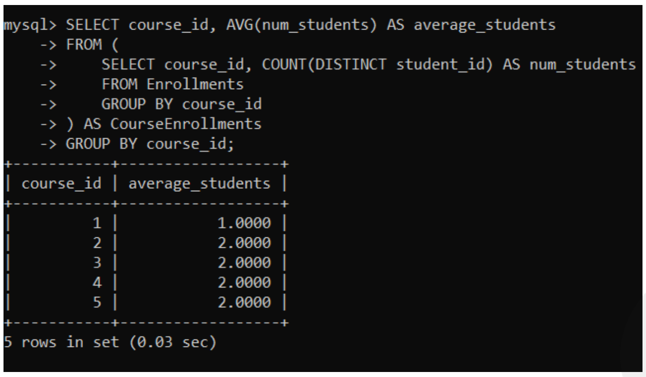
  SELECT course_id, COUNT(DISTINCT student_id) AS num_students

  FROM Enrollments

  GROUP BY course_id

) AS CourseEnrollments

GROUP BY course_id;


```
mysql> SELECT course_id, AVG(num_students) AS average_students
    -> FROM (
    ->     SELECT course_id, COUNT(DISTINCT student_id) AS num_students
    ->     FROM Enrollments
    ->     GROUP BY course_id
    -> ) AS CourseEnrollments
    -> GROUP BY course_id;
+-----------+------------------+
| course_id | average_students |
+-----------+------------------+
|         1 |           1.0000 |
|         2 |           2.0000 |
|         3 |           2.0000 |
|         4 |           2.0000 |
|         5 |           2.0000 |
+-----------+------------------+
5 rows in set (0.03 sec)
```

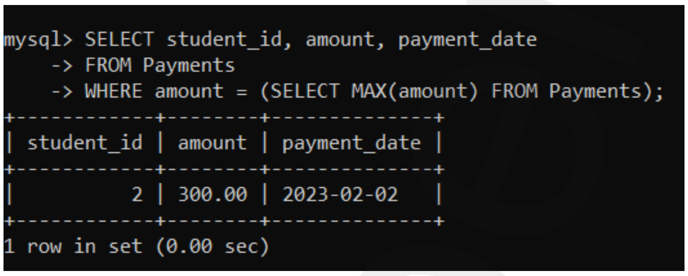
2. Identify the student(s) who made the highest payment. Use a subquery to find the maximum payment amount and then retrieve the student(s) associated with that amount.

Ans.

SELECT student_id, amount, payment_date

FROM Payments

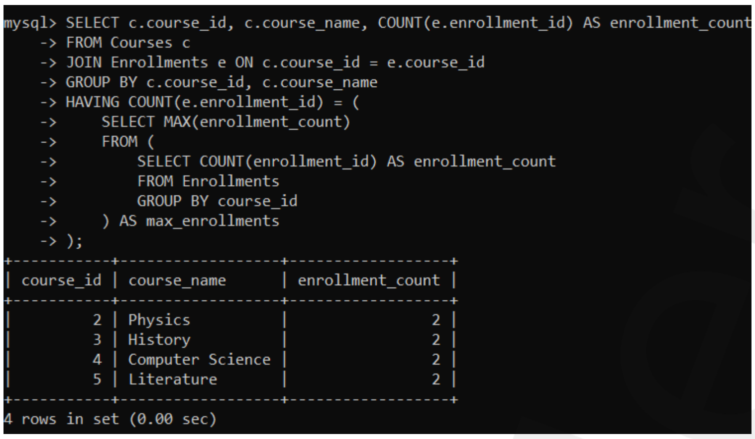
WHERE amount = (SELECT MAX(amount) FROM Payments);


```
mysql> SELECT student_id, amount, payment_date
    -> FROM Payments
    -> WHERE amount = (SELECT MAX(amount) FROM Payments);
+------------+--------+--------------+
| student_id | amount | payment_date |
+------------+--------+--------------+
|          2 | 300.00 | 2023-02-02   |
+------------+--------+--------------+
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.

Ans.

```
SELECT c.course_id, c.course_name, COUNT(e.enrollment_id) AS enrollment_count
FROM Courses c
JOIN Enrollments e ON c.course_id = e.course_id
GROUP BY c.course_id, c.course_name
HAVING COUNT(e.enrollment_id) = (
  SELECT MAX(enrollment_count)
  FROM (
    SELECT COUNT(enrollment_id) AS enrollment_count
    FROM Enrollments
    GROUP BY course_id
  ) AS max_enrollments);
```


```
mysql> SELECT c.course_id, c.course_name, COUNT(e.enrollment_id) AS enrollment_count
    -> FROM Courses c
    -> JOIN Enrollments e ON c.course_id = e.course_id
    -> GROUP BY c.course_id, c.course_name
    -> HAVING COUNT(e.enrollment_id) = (
    ->     SELECT MAX(enrollment_count)
    ->     FROM (
    ->         SELECT COUNT(enrollment_id) AS enrollment_count
    ->         FROM Enrollments
    ->         GROUP BY course_id
    ->     ) AS max_enrollments
    -> );
+-----------+------------------+------------------+
| course_id | course_name      | enrollment_count |
+-----------+------------------+------------------+
|         2 | Physics          |                2 |
|         3 | History          |                2 |
|         4 | Computer Science |                2 |
|         5 | Literature       |                2 |
+-----------+------------------+------------------+
4 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.

Ans.

SELECT t.teacher_id, t.first_name, t.last_name, SUM(p.amount) AS total_payments

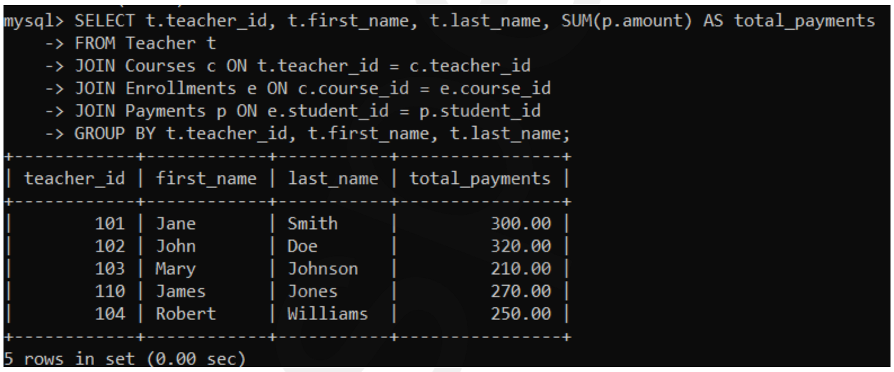
FROM Teacher t

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

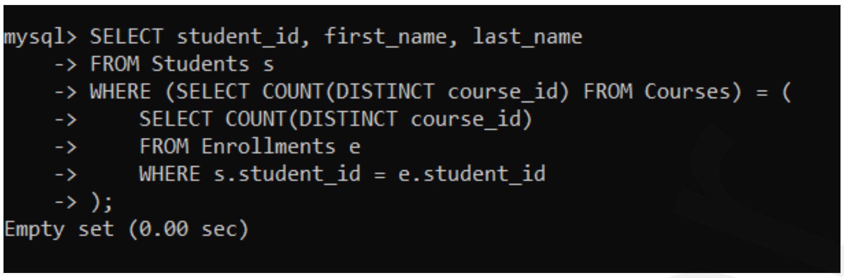
GROUP BY t.teacher_id, t.first_name, t.last_name;


```
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 |
+------------+------------+-----------+----------------+
|        101 | Jane       | Smith     |         300.00 |
|        102 | John       | Doe       |         320.00 |
|        103 | Mary       | Johnson   |         210.00 |
|        110 | James      | Jones     |         270.00 |
|        104 | Robert     | Williams  |         250.00 |
+------------+------------+-----------+----------------+
5 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.

Ans.

```
SELECT student_id, first_name, last_name
FROM Students s
WHERE (SELECT COUNT(DISTINCT course_id) FROM Courses) = (
   SELECT COUNT(DISTINCT course_id)
   FROM Enrollments e
   WHERE s.student_id = e.student_id
);
```



```
mysql> SELECT student_id, first_name, last_name
    -> FROM Students s
    -> WHERE (SELECT COUNT(DISTINCT course_id) FROM Courses) = (
    ->     SELECT COUNT(DISTINCT course_id)
    ->     FROM Enrollments e
    ->     WHERE s.student_id = e.student_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

Ans.

SELECT teacher_id, first_name, last_name

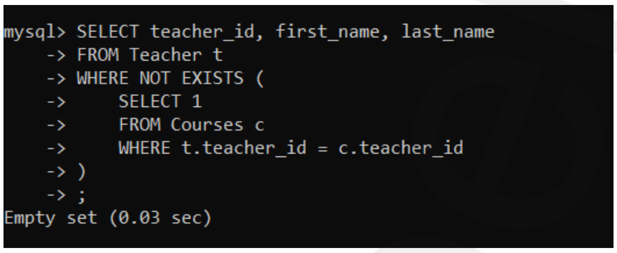
FROM Teacher t

WHERE NOT EXISTS (

SELECT 1

FROM Courses c

WHERE t.teacher_id = c.teacher_id);



```
mysql> SELECT teacher_id, first_name, last_name
    -> FROM Teacher t
    -> WHERE NOT EXISTS (
    ->     SELECT 1
    ->     FROM Courses c
    ->     WHERE t.teacher_id = c.teacher_id
    -> )
    -> ;
Empty set (0.03 sec)
```

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

Ans.

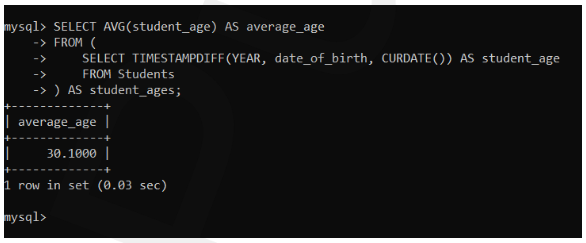
SELECT AVG(student_age) AS average_age

FROM (

SELECT TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS student_age

FROM Students

) AS student_ages;

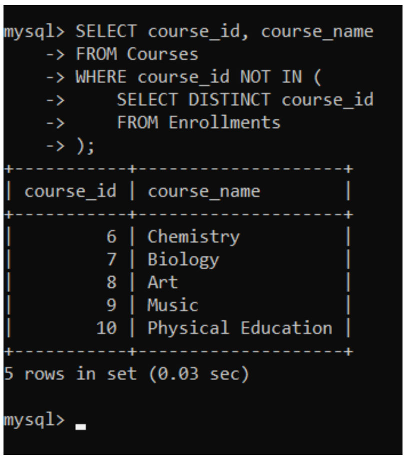


```
mysql> SELECT AVG(student_age) AS average_age
    -> FROM (
    ->     SELECT TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS student_age
    ->     FROM Students
    -> ) AS student_ages;
+-------------+
| average_age |
+-------------+
|     30.1000 |
+-------------+
1 row in set (0.03 sec)

mysql>
```

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

Ans. SELECT course_id, course_name

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

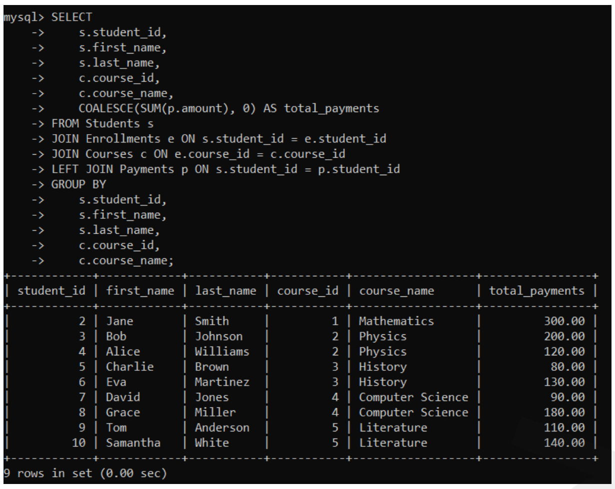


```
mysql> SELECT course_id, course_name
    -> FROM Courses
    -> WHERE course_id NOT IN (
    ->     SELECT DISTINCT course_id
    ->     FROM Enrollments
    -> );
+-----------+--------------------+
| course_id | course_name        |
+-----------+--------------------+
|         6 | Chemistry          |
|         7 | Biology            |
|         8 | Art                |
|         9 | Music              |
|        10 | Physical Education |
+-----------+--------------------+
5 rows in set (0.03 sec)

mysql>
```

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

Ans.

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

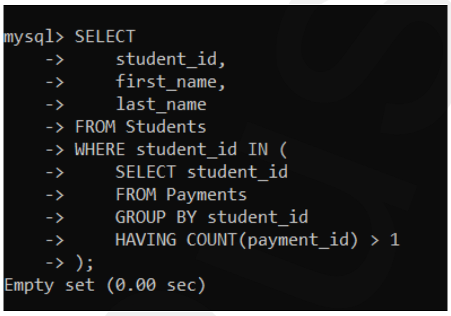


```
mysql> SELECT
    ->     s.student_id,
    ->     s.first_name,
    ->     s.last_name,
    ->     c.course_id,
    ->     c.course_name,
    ->     COALESCE(SUM(p.amount), 0) AS total_payments
    -> FROM Students s
    -> JOIN Enrollments e ON s.student_id = e.student_id
    -> JOIN Courses c ON e.course_id = c.course_id
    -> LEFT 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 |
+------------+------------+-----------+-----------+------------------+----------------+
|          2 | Jane       | Smith     |         1 | Mathematics      |         300.00 |
|          3 | Bob        | Johnson   |         2 | Physics          |         200.00 |
|          4 | Alice      | Williams  |         2 | Physics          |         120.00 |
|          5 | Charlie    | Brown     |         3 | History          |          80.00 |
|          6 | Eva        | Martinez  |         3 | History          |         130.00 |
|          7 | David      | Jones     |         4 | Computer Science |          90.00 |
|          8 | Grace      | Miller    |         4 | Computer Science |         180.00 |
|          9 | Tom        | Anderson  |         5 | Literature       |         110.00 |
|         10 | Samantha   | White     |         5 | Literature       |         140.00 |
+------------+------------+-----------+-----------+------------------+----------------+
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.

Ans.

```
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
);
```

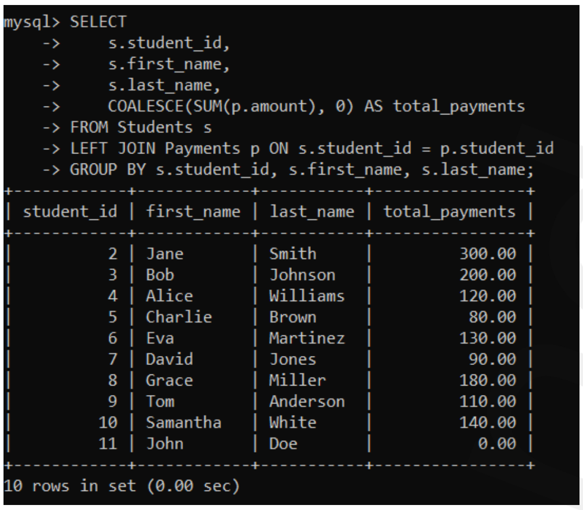


```
mysql> SELECT
    ->     student_id,
    ->     first_name,
    ->     last_name
    -> FROM Students
    -> WHERE student_id IN (
    ->     SELECT student_id
    ->     FROM Payments
    ->     GROUP BY student_id
    ->     HAVING COUNT(payment_id) > 1
    -> );
Empty set (0.00 sec)
```

11. 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.

Ans.

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

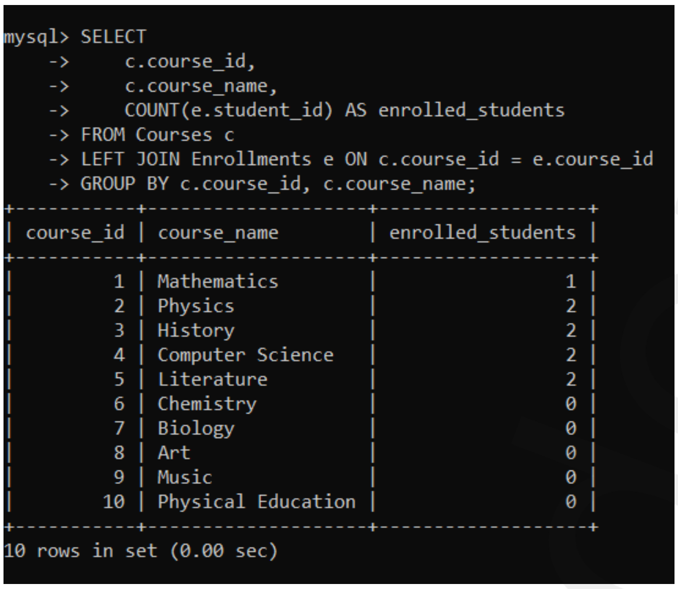


```
mysql> SELECT
    ->     s.student_id,
    ->     s.first_name,
    ->     s.last_name,
    ->     COALESCE(SUM(p.amount), 0) AS total_payments
    -> FROM Students s
    -> LEFT JOIN Payments p ON s.student_id = p.student_id
    -> GROUP BY s.student_id, s.first_name, s.last_name;
+------------+------------+----------+----------------+
| student_id | first_name | last_name | total_payments |
+------------+------------+----------+----------------+
|          2 | Jane       | Smith    |         300.00 |
|          3 | Bob        | Johnson  |         200.00 |
|          4 | Alice      | Williams |         120.00 |
|          5 | Charlie    | Brown    |          80.00 |
|          6 | Eva        | Martinez |         130.00 |
|          7 | David      | Jones    |          90.00 |
|          8 | Grace      | Miller   |         180.00 |
|          9 | Tom        | Anderson |         110.00 |
|         10 | Samantha   | White    |         140.00 |
|         11 | John       | Doe      |           0.00 |
+------------+------------+----------+----------------+
10 rows in set (0.00 sec)
```

12. 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

Ans.

```
SELECT
    c.course_id,
    c.course_name,
    COUNT(e.student_id) AS enrolled_students
FROM Courses c
LEFT JOIN Enrollments e ON c.course_id = e.course_id
GROUP BY c.course_id, c.course_name;
```



```
mysql> SELECT
    ->     c.course_id,
    ->     c.course_name,
    ->     COUNT(e.student_id) AS enrolled_students
    -> 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 |
+-----------+-------------------+-------------------+
|         1 | Mathematics       |                 1 |
|         2 | Physics           |                 2 |
|         3 | History           |                 2 |
|         4 | Computer Science  |                 2 |
|         5 | Literature        |                 2 |
|         6 | Chemistry         |                 0 |
|         7 | Biology           |                 0 |
|         8 | Art               |                 0 |
|         9 | Music             |                 0 |
|        10 | Physical Education |                0 |
+-----------+-------------------+-------------------+
10 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

Ans.

SELECT

s.student_id,

s.first_name,

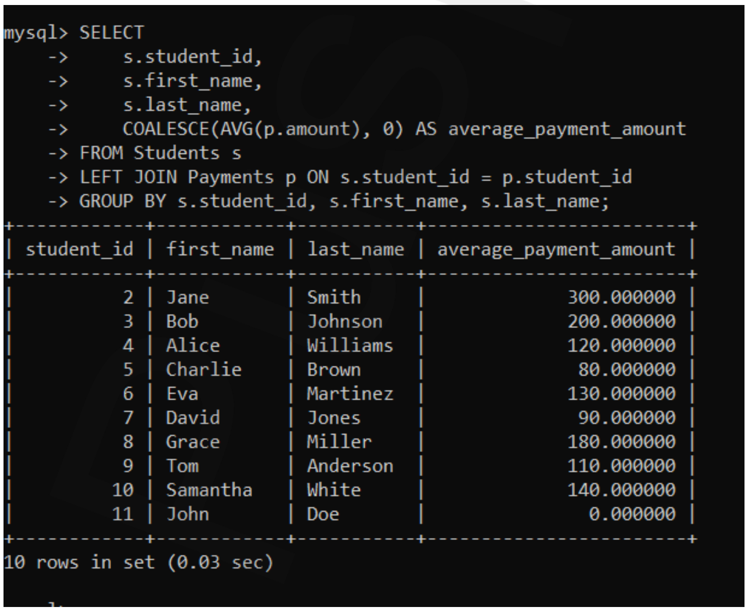
s.last_name,

COALESCE(AVG(p.amount), 0) AS average_payment_amount

FROM Students s

LEFT JOIN Payments p ON s.student_id = p.student_id

GROUP BY s.student_id, s.first_name, s.last_name;



```
mysql> SELECT
    ->     s.student_id,
    ->     s.first_name,
    ->     s.last_name,
    ->     COALESCE(AVG(p.amount), 0) AS average_payment_amount
    -> FROM Students s
    -> LEFT JOIN Payments p ON s.student_id = p.student_id
    -> GROUP BY s.student_id, s.first_name, s.last_name;
+------------+------------+-----------+------------------------+
| student_id | first_name | last_name | average_payment_amount |
+------------+------------+-----------+------------------------+
|          2 | Jane       | Smith     |             300.000000 |
|          3 | Bob        | Johnson   |             200.000000 |
|          4 | Alice      | Williams  |             120.000000 |
|          5 | Charlie    | Brown     |              80.000000 |
|          6 | Eva        | Martinez  |             130.000000 |
|          7 | David      | Jones     |              90.000000 |
|          8 | Grace      | Miller    |             180.000000 |
|          9 | Tom        | Anderson  |             110.000000 |
|         10 | Samantha   | White     |             140.000000 |
|         11 | John       | Doe       |               0.000000 |
+------------+------------+-----------+------------------------+
10 rows in set (0.03 sec)
```