Task 1. Database Design:

- 1. Create the database named "SISDB".
- 2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.
- a. Students
- b. Courses
- c. Enrollments
- d. Teacher
- e. Payments
- 4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.

```
1 • CREATE DATABASE SISDB;
 2 • USE SISDB;
 3 • ⊖ CREATE TABLE Students (
           student_id INT PRIMARY KEY AUTO_INCREMENT,
           first_name VARCHAR(255),
 5
 6
           last_name VARCHAR(255),
 7
           date_of_birth DATE,
 8
           email VARCHAR(255),
 9
           phone_number VARCHAR(20)
10
     );
11
12 • ⊖ CREATE TABLE Teacher (
13
          teacher_id INT PRIMARY KEY AUTO_INCREMENT,
           first_name VARCHAR(255),
14
15
           last_name VARCHAR(255),
16
           email VARCHAR(255)
     );
17
course_id INT PRIMARY KEY AUTO_INCREMENT,
20
21
         course_name VARCHAR(255),
22
         credits INT,
23
         teacher_id INT,
24
         FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_id)
25
26 • ⊖ CREATE TABLE Enrollments (
         enrollment id INT PRIMARY KEY AUTO_INCREMENT,
27
         student_id INT NOT NULL,
28
        course id INT NOT NULL,
29
         enrollment_date DATE NOT NULL,
30
31
         FOREIGN KEY (student_id) REFERENCES Students(student_id),
32
         FOREIGN KEY (course_id) REFERENCES Courses(course_id)
33
   );
```

```
35 • ○ CREATE TABLE Payments (

payment_id INT PRIMARY KEY AUTO_INCREMENT,

student_id INT NOT NULL,

amount DECIMAL(10, 2),

payment_date DATE,

FOREIGN KEY (student_id) REFERENCES Students(student_id)

);
```

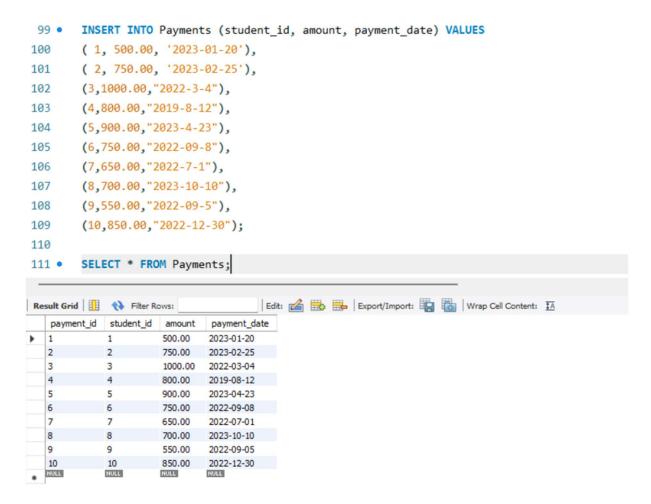
- 5. Insert at least 10 sample records into each of the following tables.
- i. Students
- ii. Courses
- ill. Enrollments
- iv. Teacher
- v. Payments

```
43 •
        INSERT INTO Students (student id, first name, last name, date of birth, email, phone number)
44
        VALUES
        (1, 'Krishna', 'Patle', '2001-08-12', 'krishnapatle@gmail.com', '9325654953'),
45
        (2, 'Kashyap', 'Punyawan', '2001-03-04', 'kashyappunyawan@gmail.com', '9325655453'),
        (3, 'Harshal', 'Meshram', '2002-05-27', 'harshalmeshram@gmail.com', '9125654953'),
47
        (4, 'Nitin', 'Turkar', '2000-12-01', 'nitinturkar@gmail.com', '8698454798'),
48
        (5, 'Vikas', 'Nagpure', '2001-04-23', 'vikasnagpure@gmail.com', '9124524953'),
49
        (6, 'Shivam', 'Kale', '2001-03-16', 'shivamkale@gmail.com', '7825654953'),
50
51
        (7, 'Ruchika', 'Chafekar', '2001-06-30', 'ruchikachafekar@gmail.com', '7447654906'),
52
        (8,'Neha','Patle','2003-09-3','nehapatle@gmail.com','7825654901'),
        (9, 'Pratiksha', 'Katre', '2000-05-12', 'pratikshakatre@gmail.com', '9125654953'),
53
54
        (10, 'Shruti', 'Kolhe', '2001-03-6', 'shrutikolhe@gmail.com', '6725654900');
55 • SELECT * FROM Students:
Edit: 🕍 🔜 Export/Import: 📳 🐻 | Wrap Cell Content: 🖽
  student_id first_name last_name date_of_birth email
                                                                phone_number
                    Patle
                              2001-08-12 krishnapatle@gmail.com
                                                                9325654953
 2
           Kashyap Punyawan 2001-03-04 kashyappunyawan@gmail.com 9325655453
  3
           Harshal
                     Meshram
                             2002-05-27
                                        harshalmeshram@gmail.com
                                                                9125654953
                  Turkar 2000-12-01 nitinturkar@gmail.com 8698454798
           Nitin
           Vikas
                    Nagpure
                              2001-04-23
                                         vikasnagpure@gmail.com
                                                                9124524953
           Shivam Kale 2001-03-16 shivamkale@gmail.com
                                                             7825654953
  6
                    Chafekar
                             2001-06-30
2003-09-03
           Ruchika
                                        ruchikachafekar@gmail.com
                                                                7447654906
  8
           Neha
                    Patle
                                        nehapatle@gmail.com
                                                               7825654901
           Pratiksha
                    Katre
                             2000-05-12
                                        pratikshakatre@gmail.com
                                                                9125654953
                                                                6725654900
                    Kolhe
                                         shrutikolhe@gmail.com
  10
                             2001-03-06
           Shruti
```

```
INSERT INTO Teacher (first_name, last_name, email) VALUES
 57 •
 58
         ("Mr. Smith", "Taylor", "smith@email.com"),
         ("Ms. Jones", "Swift", "jones@email.com"),
 59
         ("Mr. Kailas", "Shekhar", "kailash@gmail.com"),
 60
         ("Mr. Pruthvi", "Chaudhary", "pruthvi@gmail.com"),
 61
         ("Mr.Rajat", "Patidar", "rajat@gmail.com"),
 62
 63
         ("Ms. Yogita", "Lanjewar", "yogita@gmail.com"),
         ("Ms. Ragini", "Yadav", "ragini@gmail.com"),
 64
 65
         ("Mr. Saurabh", "Gedekar", "saurabh@gmail.com"),
         ("Ms. Neha", "Patle", "nehapatle@gmail.com"),
 66
         ("Mr. Harshal", "Meshram", "harshal@gmail.com");
 67
 68
 69 •
         SELECT * FROM Teacher;
                                        Edit: 🔏 🖶 Export/Import: 📳 🐻 Wrap Cell Content: 🛂
Result Grid  Filter Rows:
   teacher_id first_name
                        last_name email
   1
             Mr. Smith
                       Taylor
                                 smith@email.com
            Ms. Jones Swift
   2
                                 jones@email.com
            Mr. Kailas
                       Shekhar
                                 kailash@gmail.com
   3
   4
            Mr. Pruthvi Chaudhary pruthvi@gmail.com
             Mr.Rajat
                       Patidar
                                 rajat@gmail.com
                      Lanjewar
   6
            Ms. Yogita
                                 yogita@gmail.com
   7
                                 ragini@gmail.com
            Ms. Ragini
                       Yaday
            Mr. Saurabh Gedekar
                                 saurabh@gmail.com
   8
            Ms. Neha
                       Patle
                                 nehapatle@gmail.com
            Mr. Harshal
                     Meshram
                                 harshal@gmail.com
                       NULL
71 •
        INSERT INTO Courses (course_name, credits, teacher_id) VALUES
72
        ("Mathematics", 3, 1),
73
        ("English", 3, 2),
74
        ("History", 2, 3),
        ("Geograhy", 3, 4),
75
        ("C++",4,5),
76
77
        ("Python", 4,6),
78
        ("Economy", 2, 7),
        ("Science",4,8),
79
        ("Social Science",2,9),
80
        ("Java",4,10);
81
82
83 •
        SELECT * FROM Courses;
Edit: 🔏 📆 🖶 Export/Import: 🏭 👸 Wrap Cell Content: 🖽
  course_id course_name
                      credits
                             teacher_id
  1
           Mathematics
          English
  2
                     3
                            2
  3
          History
  4
          Geograhy
                     3
                            4
  5
          C++
                            5
  6
          Python
                            6
  7
          Economy
                            7
  8
          Science
                      4
                            8
          Social Science 2
  10
                      4
                            10
```

Krushnakumar Homraj Patle

```
85 •
       INSERT INTO Enrollments (student_id, course_id, enrollment_date) VALUES
86
        (1, 1, "2023-10-01"),
        (2, 2, "2023-10-02"),
87
       (3,5,"2022-3-4"),
88
       (4,3,"2019-8-12"),
89
90
       (5,4,"2023-4-23"),
       (6,7,"2022-09-8"),
91
       (7,8,"2022-7-1"),
92
       (8,9,"2023-10-10"),
93
       (9,10,"2022-09-5"),
94
95
       (10,6,"2022-12-30");
96
       SELECT * FROM Enrollments;
97 •
                                    Edit: 🚄 🖶 Export/Import: 📳 🐻 Wrap Cell Content: 🖽
enrollment_id
             student_id course_id enrollment_date
                              2023-10-01
                      1
  2
                              2023-10-02
  3
                              2022-03-04
             3
                      5
  4
            4
                     3
                              2019-08-12
                              2023-04-23
                     7
  6
            6
                              2022-09-08
             7
                      8
                              2022-07-01
  8
            8
                     9
                              2023-10-10
  9
            9
                              2022-09-05
                      10
  10
            10
                              2022-12-30
            HULL
                     NULL
```



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

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

a. First Name: John

b. Last Name: Doe

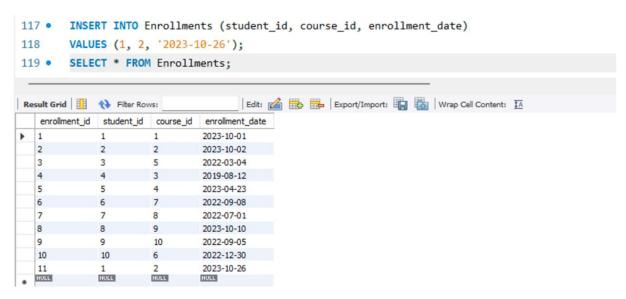
c. Date of Birth: 1995-08-15

d. Email: john.doe@example.com

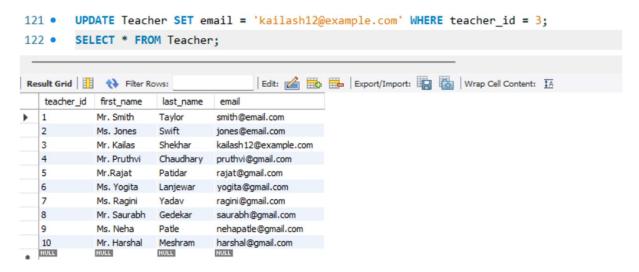
e. Phone Number: 1234567890

```
113 •
         INSERT INTO Students (first_name, last_name, date_of_birth, email, phone_number)
114
         VALUES ('John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
         SELECT * FROM Students;
115 •
| Edit: 🚄 🖶 | Export/Import: 📳 👸 | Wrap Cell Content: 🖽
   student_id first_name last_name date_of_birth email
                                                                   phone_number
                                2001-08-12
1
            Krishna
                      Patle
                                            krishnapatle@gmail.com
                                                                    9325654953
                      Punyawan 2001-03-04
                                            kashyappunyawan@gmail.com 9325655453
  2
            Kashyap
                      Meshram
                                2002-05-27
                                            harshalmeshram@gmail.com
            Harshal
                                                                    9125654953
   4
            Nitin
                      Turkar
                                2000-12-01 nitinturkar@gmail.com
                                                                   8698454798
   5
            Vikas
                                2001-04-23
                                            vikasnagpure@gmail.com
                                                                    9124524953
                      Nagpure
  6
                               2001-03-16 shivamkale@gmail.com
            Shivam
                      Kale
                                                                   7825654953
            Ruchika
  7
                      Chafekar
                                2001-06-30
                                            ruchikachafekar@gmail.com
                                                                    7447654906
  8
            Neha
                      Patle 2003-09-03 nehapatle@gmail.com
                                                                 7825654901
            Pratiksha
                      Katre
                                2000-05-12
                                            pratikshakatre@gmail.com
                                                                    9125654953
                      Kolhe
  10
                              2001-03-06 shrutikolhe@gmail.com
                                                                  6725654900
            Shruti
                      Doe
   11
            John
                                1995-08-15
                                            john.doe@example.com
                                                                    1234567890
```

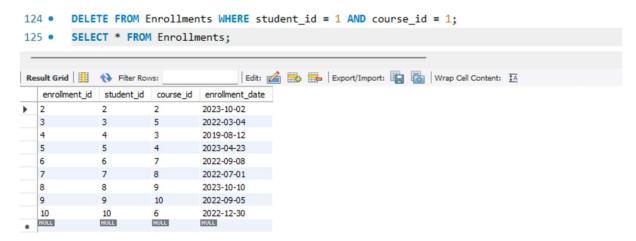
2. Write an SQL query to enroll a student in a course. Choose an existing student and course and insert a record into the "Enrollments" table with the enrollment date.



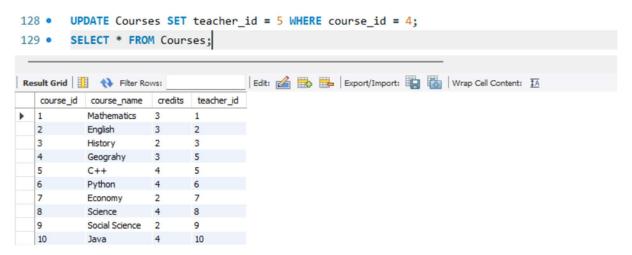
3. Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.



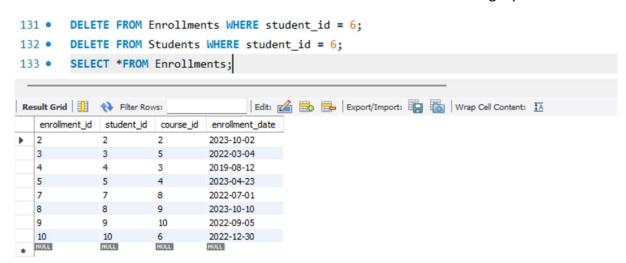
4. Write an SQL query to delete a specific enrollment record from the "Enrollments" table. Select an enrollment record based on the student and course.



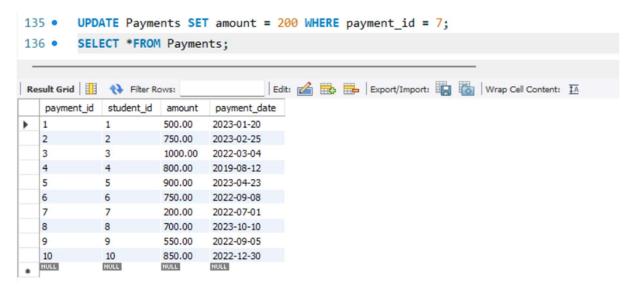
5. Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.



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

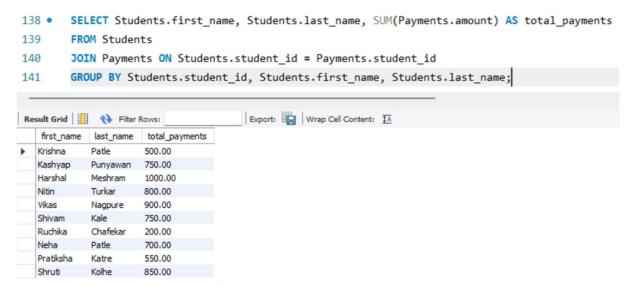


7. Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.



Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:

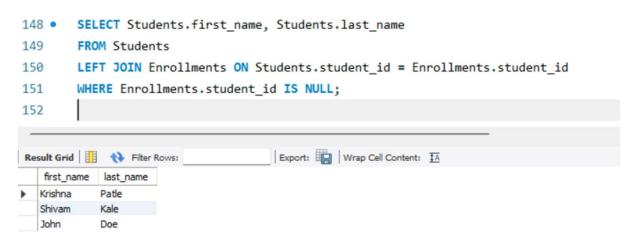
1. Write an SQL query to calculate the total payments made by a specific student. You will need to join the "Payments" table with the "Students" table based on the student's ID.



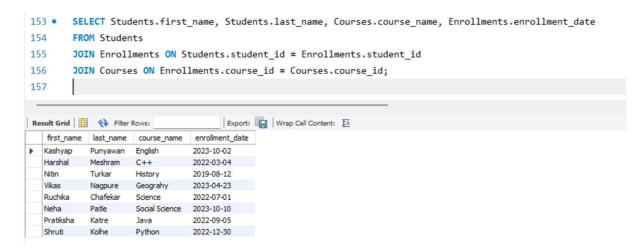
2. Write an SQL query to retrieve a list of courses along with the count of students enrolled in each course. Use a JOIN operation between the "Courses" table and the "Enrollments" table.

```
143 •
        SELECT Courses.course_name, COUNT(Enrollments.student_id) AS enrolled_students
144
        FROM Courses
145
        LEFT JOIN Enrollments ON Courses.course_id = Enrollments.course_id
146
        GROUP BY Courses.course_id, Courses.course_name;
147
                                    Export: Wrap Cell Content: IA
course_name enrolled_students
  Mathematics
             0
  English
            1
  History
  Geograhy
  C++
  Python
             1
  Economy
             0
  Science
             1
  Social Science
             1
             1
```

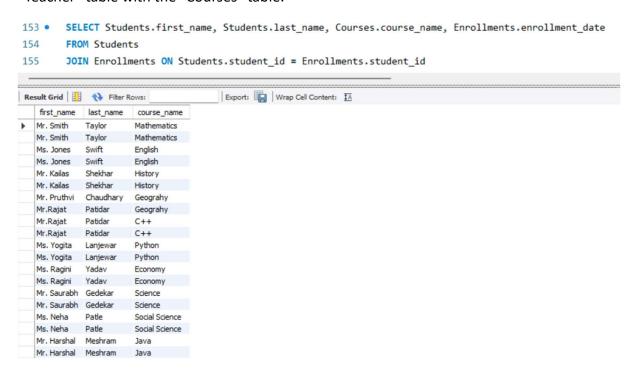
3. Write an SQL query to find the names of students who have not enrolled in any course. Use a LEFT JOIN between the "Students" table and the "Enrollments" table to identify students without enrollments.



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



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



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

```
SELECT Students.first name, Students.last name, Enrollments.enrollment date
162 •
163
       FROM Students
       JOIN Enrollments ON Students.student_id = Enrollments.student_id
164
       JOIN Courses ON Enrollments.course_id = Courses.course_id
165
       WHERE Courses.course_name = 'Science';
166
Export: Wrap Cell Content: IA
   first_name last_name enrollment_date
 Ruchika
           Chafekar
                   2022-07-01
```

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.

```
SELECT Students.first_name, Students.last_name

FROM Students

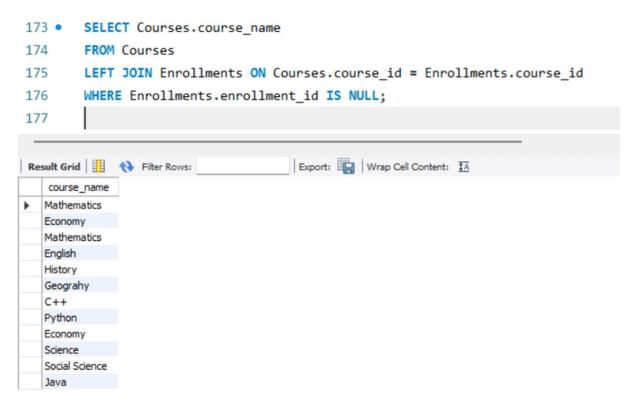
LEFT JOIN Payments ON Students.student_id = Payments.student_id

WHERE Payments.payment_id IS NULL;

Result Grid  Filter Rows:

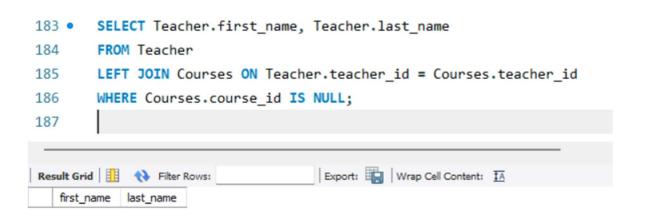
| Export: | Wrap Cell Content: | May Cell Content: | Ma
```

8. Write a query to identify courses that have no enrollments. You'll need to use a LEFT JOIN between the "Courses" table and the "Enrollments" table and filter for courses with NULL enrollment records



9. identify students who are enrolled in more than one course. Use a self-join on the "Enrollments" table to find students with multiple enrollment records.

10. Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher" table and the "Courses" table and filter for teachers with NULL course assignments



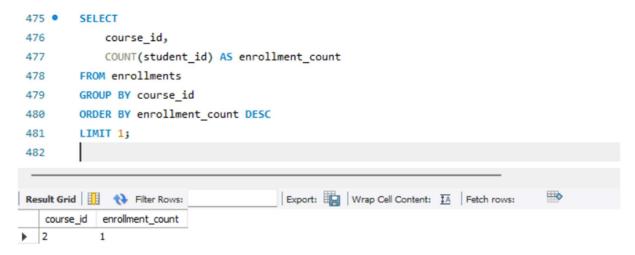
Task 4. Subquery and its type:

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

```
459 • SELECT
460
           course id,
461
           AVG(enrollment_count) AS average_students_enrolled
    ⊖ FROM (
462
463
           SELECT
464
              course_id,
              COUNT(student_id) AS enrollment_count
           FROM enrollments
           GROUP BY course id
467
468
       ) AS course_enrollments
       GROUP BY course_id;
469
Export: Wrap Cell Content: IA
  course_id average_students_enrolled
          1.0000
        1.0000
 3
          1.0000
  5
          1.0000
          1.0000
  6
  8
         1.0000
          1.0000
        1.0000
 10
```

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

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



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

```
483 •
        SELECT
484
            teacher_id,
            SUM(amount) AS total_payments
485
      ⊖ FROM (
486
            SELECT
487
                e.course_id,
488
489
                p.amount,
                c.teacher id
490
            FROM enrollments e
491
            JOIN payments p ON e.student_id = p.student_id AND e.course_id = e.course_id
492
            JOIN courses c ON e.course_id = c.course_id
493
        ) AS course_payments
494
        GROUP BY teacher_id;
495
                                        Export: Wrap Cell Content: IA
teacher_id total_payments
  2
            750.00
  5
            1900.00
  3
            800.00
  8
            200.00
  9
            700.00
  10
            550.00
  6
            850.00
```

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

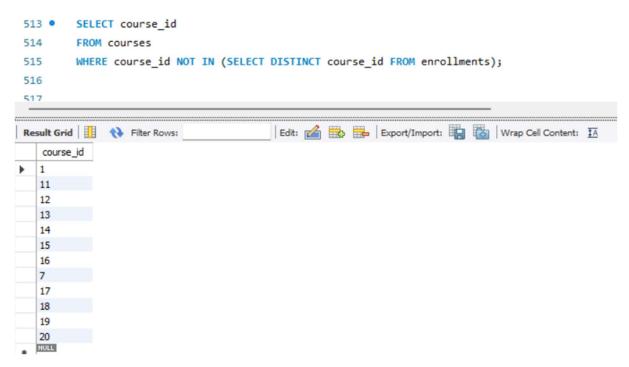
```
497 • SELECT student_id
498 FROM enrollments
499 GROUP BY student_id
500 HAVING COUNT(DISTINCT course_id) = (SELECT COUNT(*) FROM courses);
501

Result Grid  Filter Rows: Export: Wrap Cell Content: A
```

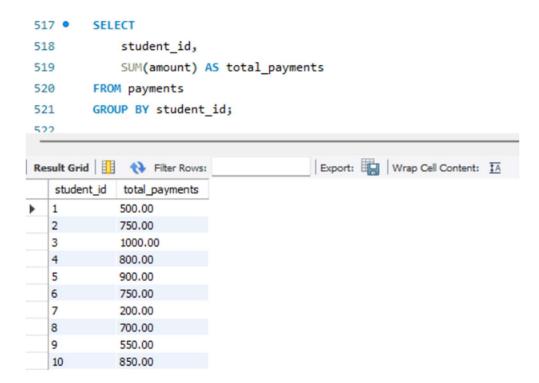
6. Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to find teachers with no course assignments.

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

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



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



10. Identify students who have made more than one payment. Use subqueries and aggregate functions to count payments per student and filter for those with counts greater than one.

```
SELECT student_id

FROM payments

GROUP BY student_id

HAVING COUNT(payment_id) > 1;

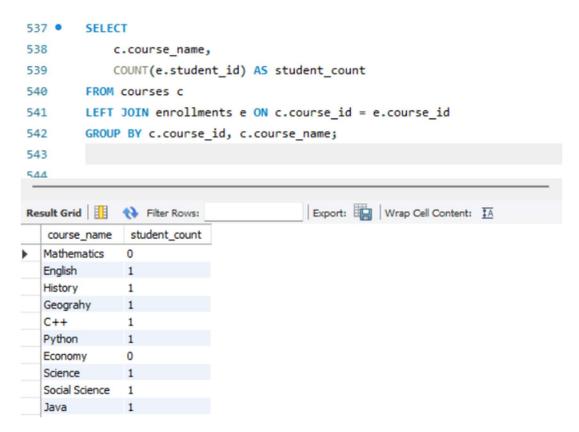
Result Grid Filter Rows:

Export: Wrap Cell Content: A student_id
```

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.

```
528 •
         SELECT
529
             s.student id,
             s.first_name,
530
             s.last_name,
531
             COALESCE(SUM(p.amount), 0) AS total_payments
532
533
         FROM students s
         LEFT JOIN payments p ON s.student_id = p.student_id
534
         GROUP BY s.student_id, s.first_name, s.last_name;
535
536
537
                                           Export: Wrap Cell Content: IA
student_id
                                   total_payments
              first_name
                        last_name
             Krishna
                                  500.00
                       Patle
  2
             Kashyap
                       Punyawan
                                  750.00
  3
             Harshal
                       Meshram
                                  1000.00
             Nitin
                       Turkar
                                  800.00
  5
             Vikas
                                  900.00
                       Nagpure
  6
             Shivam
                       Kale
                                  750.00
  7
             Ruchika
                        Chafekar
                                  200.00
  8
             Neha
                       Patle
                                  700.00
             Pratiksha
                        Katre
                                  550.00
  10
             Shruti
                       Kolhe
                                  850.00
  11
             John
                                  0.00
                       Doe
```

12. Retrieve a list of course names along with the count of students enrolled in each course. Use JOIN operations between the "Courses" table and the "Enrollments" table and GROUP BY to count enrollments.



13. Calculate the average payment amount made by students. Use JOIN operations between the "Students" table and the "Payments" table and GROUP BY to calculate the average.

