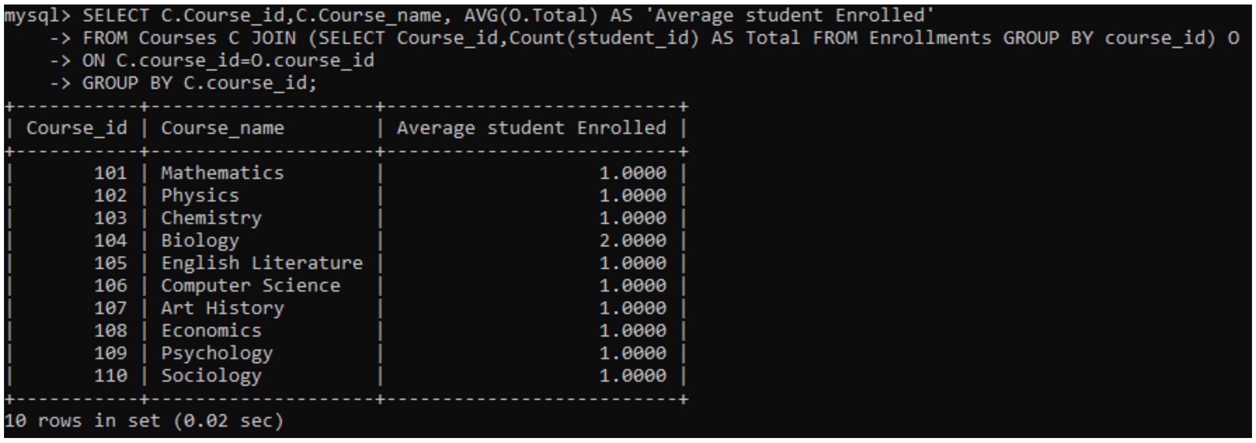
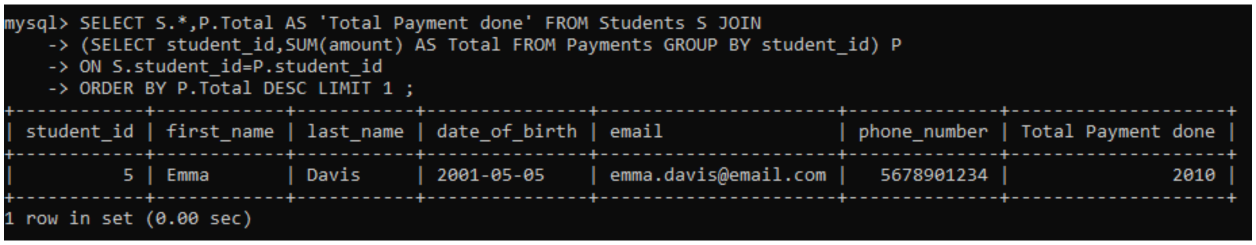
Task 4. Subquery and its type:

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



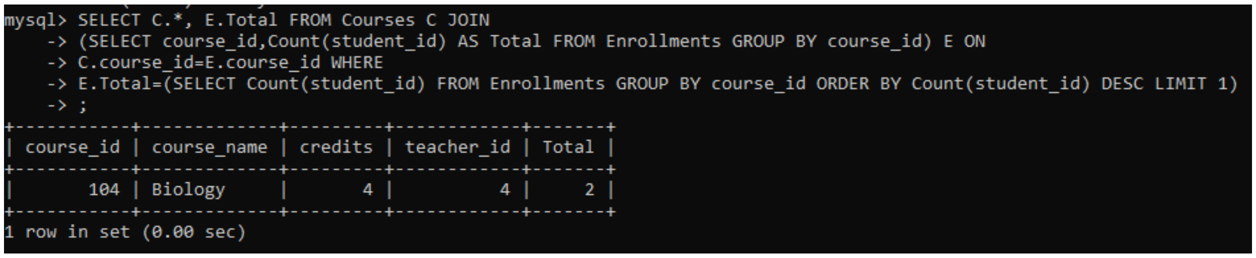
```
mysql> SELECT C.Course_id,C.Course_name, AVG(O.Total) AS 'Average student Enrolled'
    -> FROM Courses C JOIN (SELECT Course_id,Count(student_id) AS Total FROM Enrollments GROUP BY course_id) O
    -> ON C.course_id=O.course_id
    -> GROUP BY C.course_id;
+-----------+--------------------+--------------------------+
| Course_id | Course_name        | Average student Enrolled |
+-----------+--------------------+--------------------------+
|       101 | Mathematics        |                   1.0000 |
|       102 | Physics            |                   1.0000 |
|       103 | Chemistry          |                   1.0000 |
|       104 | Biology            |                   2.0000 |
|       105 | English Literature |                   1.0000 |
|       106 | Computer Science   |                   1.0000 |
|       107 | Art History        |                   1.0000 |
|       108 | Economics          |                   1.0000 |
|       109 | Psychology         |                   1.0000 |
|       110 | Sociology          |                   1.0000 |
+-----------+--------------------+--------------------------+
10 rows in set (0.02 sec)
```

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



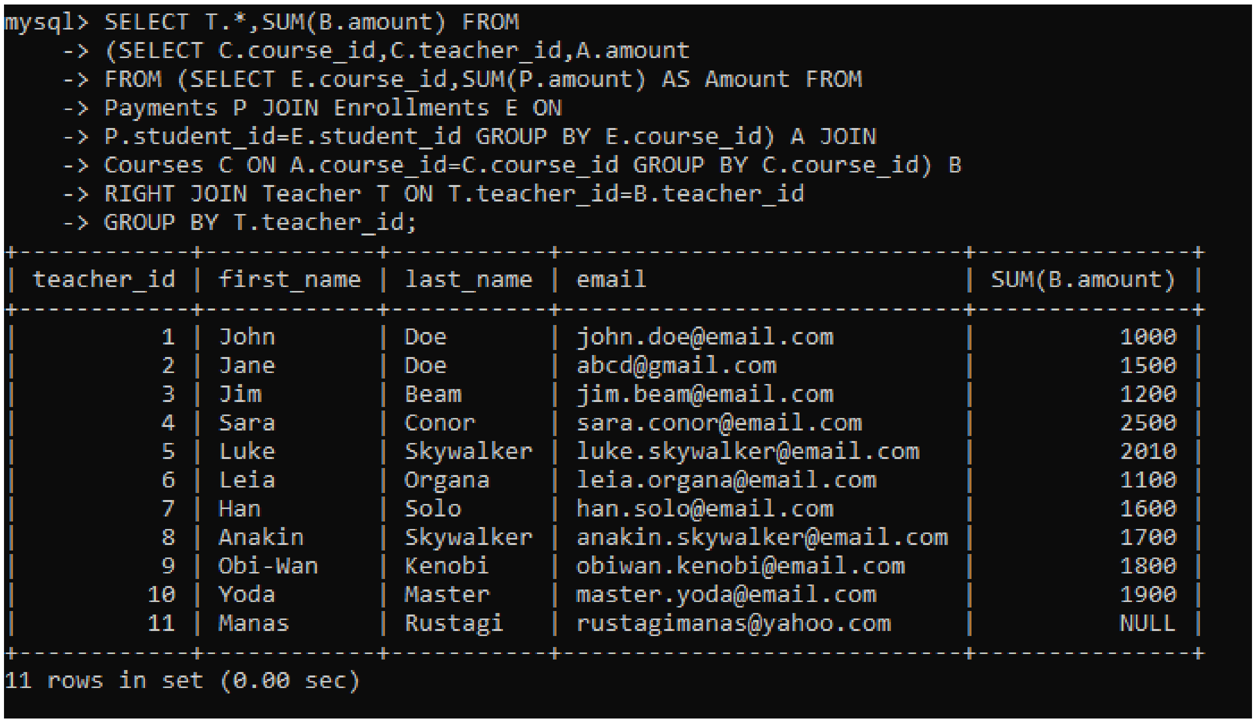
```
mysql> SELECT S.*,P.Total AS 'Total Payment done' FROM Students S JOIN
    -> (SELECT student_id,SUM(amount) AS Total FROM Payments GROUP BY student_id) P
    -> ON S.student_id=P.student_id
    -> ORDER BY P.Total DESC LIMIT 1 ;
+------------+------------+-----------+---------------+----------------------+--------------+--------------------+
| student_id | first_name | last_name | date_of_birth | email                | phone_number | Total Payment done |
+------------+------------+-----------+---------------+----------------------+--------------+--------------------+
|          5 | Emma       | Davis     | 2001-05-05    | emma.davis@email.com |   5678901234 |               2010 |
+------------+------------+-----------+---------------+----------------------+--------------+--------------------+
1 row in set (0.00 sec)
```

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



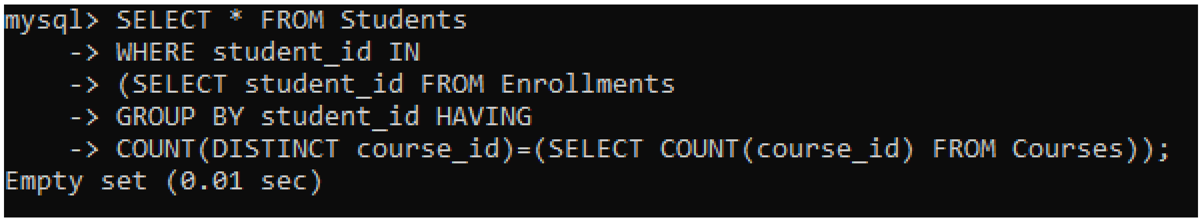
```
mysql> SELECT C.*, E.Total FROM Courses C JOIN
    -> (SELECT course_id,Count(student_id) AS Total FROM Enrollments GROUP BY course_id) E ON
    -> C.course_id=E.course_id WHERE
    -> E.Total=(SELECT Count(student_id) FROM Enrollments GROUP BY course_id ORDER BY Count(student_id) DESC LIMIT 1)
    -> ;
+-----------+-------------+---------+------------+-------+
| course_id | course_name | credits | teacher_id | Total |
+-----------+-------------+---------+------------+-------+
|       104 | Biology     |       4 |          4 |     2 |
+-----------+-------------+---------+------------+-------+
1 row in set (0.00 sec)
```

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

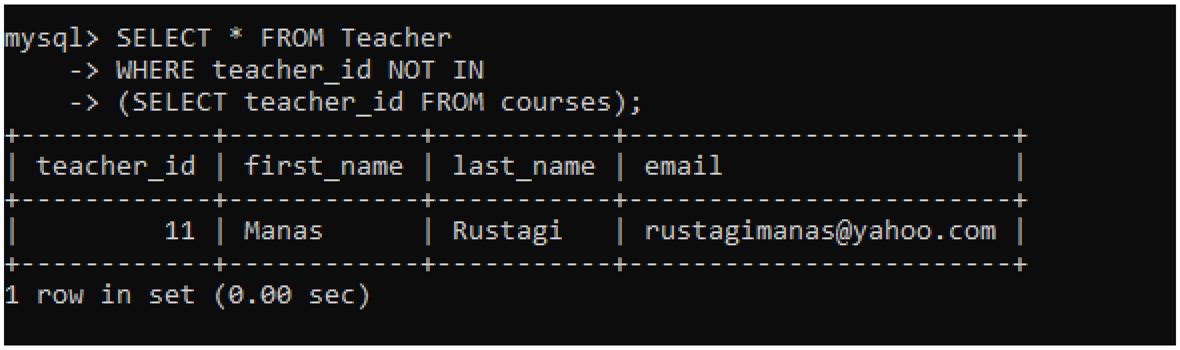

```
mysql> SELECT T.*,SUM(B.amount) FROM
    -> (SELECT C.course_id,C.teacher_id,A.amount
    -> FROM (SELECT E.course_id,SUM(P.amount) AS Amount FROM
    -> Payments P JOIN Enrollments E ON
    -> P.student_id=E.student_id GROUP BY E.course_id) A JOIN
    -> Courses C ON A.course_id=C.course_id GROUP BY C.course_id) B
    -> RIGHT JOIN Teacher T ON T.teacher_id=B.teacher_id
    -> GROUP BY T.teacher_id;
+------------+------------+-----------+----------------------------+---------------+
| teacher_id | first_name | last_name | email                      | SUM(B.amount) |
+------------+------------+-----------+----------------------------+---------------+
|          1 | John       | Doe       | john.doe@email.com         |          1000 |
|          2 | Jane       | Doe       | abcd@gmail.com             |          1500 |
|          3 | Jim        | Beam      | jim.beam@email.com         |          1200 |
|          4 | Sara       | Conor     | sara.conor@email.com       |          2500 |
|          5 | Luke       | Skywalker | luke.skywalker@email.com   |          2010 |
|          6 | Leia       | Organa    | leia.organa@email.com      |          1100 |
|          7 | Han        | Solo      | han.solo@email.com         |          1600 |
|          8 | Anakin     | Skywalker | anakin.skywalker@email.com |          1700 |
|          9 | Obi-Wan    | Kenobi    | obiwan.kenobi@email.com    |          1800 |
|         10 | Yoda       | Master    | master.yoda@email.com      |          1900 |
|         11 | Manas      | Rustagi   | rustagimanas@yahoo.com     |          NULL |
+------------+------------+-----------+----------------------------+---------------+
11 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.

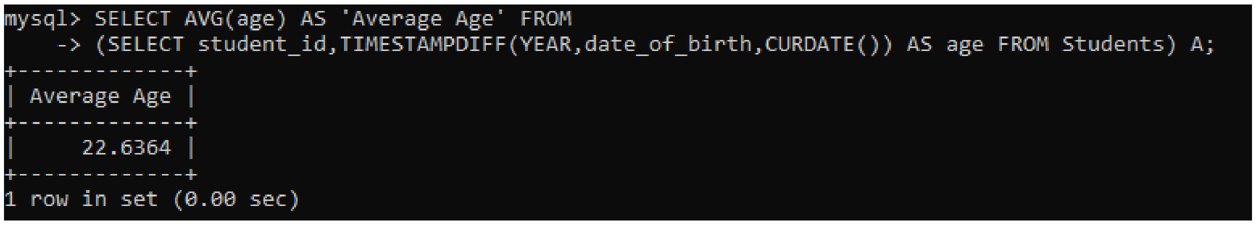
There was NO student who enrolled in all the available courses


```
mysql> SELECT * FROM Students
    -> WHERE student_id IN
    -> (SELECT student_id FROM Enrollments
    -> GROUP BY student_id HAVING
    -> COUNT(DISTINCT course_id)=(SELECT COUNT(course_id) FROM Courses));
Empty set (0.01 sec)
```

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

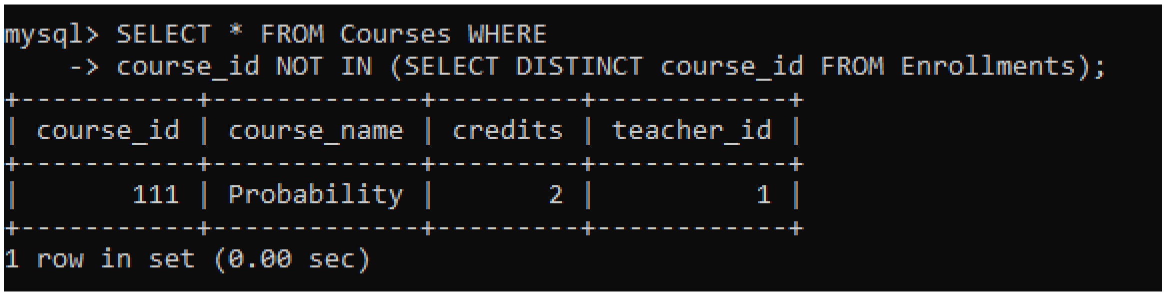

```
mysql> SELECT * FROM Teacher
    -> WHERE teacher_id NOT IN
    -> (SELECT teacher_id FROM courses);
+------------+------------+-----------+------------------------+
| teacher_id | first_name | last_name | email                  |
+------------+------------+-----------+------------------------+
|         11 | Manas      | Rustagi   | rustagimanas@yahoo.com |
+------------+------------+-----------+------------------------+
1 row in set (0.00 sec)
```

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



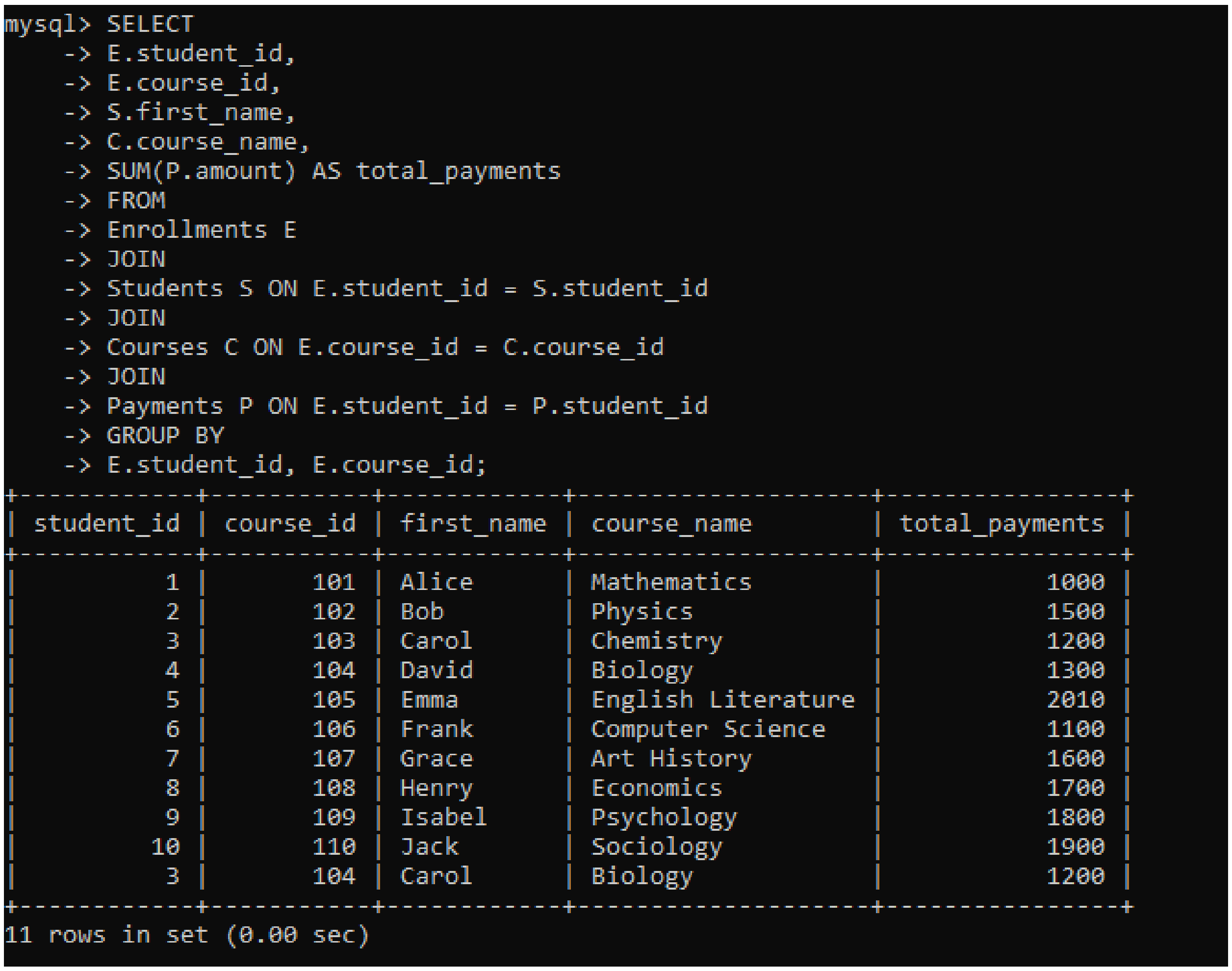
```
mysql> SELECT AVG(age) AS 'Average Age' FROM
    -> (SELECT student_id,TIMESTAMPDIFF(YEAR,date_of_birth,CURDATE()) AS age FROM Students) A;
+-------------+
| Average Age |
+-------------+
|     22.6364 |
+-------------+
1 row in set (0.00 sec)
```

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



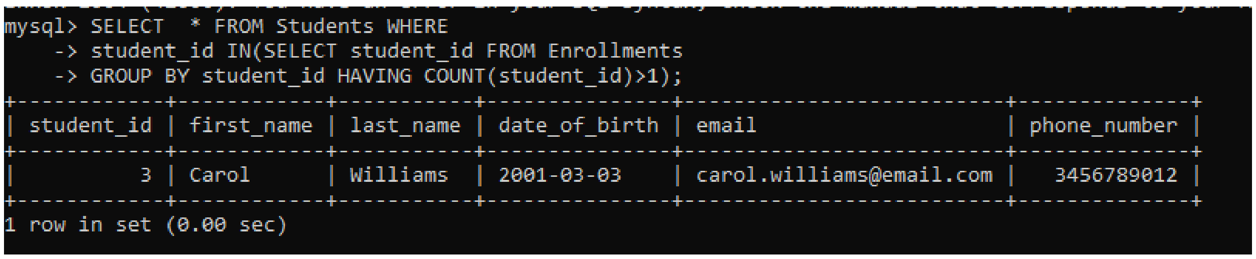
```
mysql> SELECT * FROM Courses WHERE
    -> course_id NOT IN (SELECT DISTINCT course_id FROM Enrollments);
+-----------+-------------+---------+------------+
| course_id | course_name | credits | teacher_id |
+-----------+-------------+---------+------------+
|       111 | Probability |       2 |          1 |
+-----------+-------------+---------+------------+
1 row in set (0.00 sec)
```

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



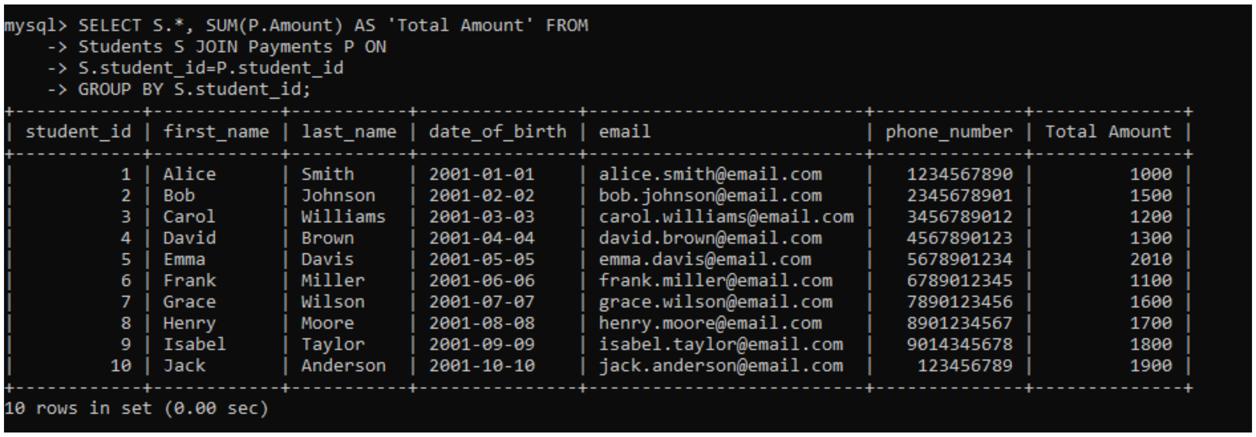
```
mysql> SELECT
    -> E.student_id,
    -> E.course_id,
    -> S.first_name,
    -> C.course_name,
    -> SUM(P.amount) AS total_payments
    -> FROM
    -> Enrollments E
    -> JOIN
    -> Students S ON E.student_id = S.student_id
    -> JOIN
    -> Courses C ON E.course_id = C.course_id
    -> JOIN
    -> Payments P ON E.student_id = P.student_id
    -> GROUP BY
    -> E.student_id, E.course_id;
+------------+-----------+------------+--------------------+----------------+
| student_id | course_id | first_name | course_name        | total_payments |
+------------+-----------+------------+--------------------+----------------+
|          1 |       101 | Alice      | Mathematics        |           1000 |
|          2 |       102 | Bob        | Physics            |           1500 |
|          3 |       103 | Carol      | Chemistry          |           1200 |
|          4 |       104 | David      | Biology            |           1300 |
|          5 |       105 | Emma       | English Literature |           2010 |
|          6 |       106 | Frank      | Computer Science   |           1100 |
|          7 |       107 | Grace      | Art History        |           1600 |
|          8 |       108 | Henry      | Economics          |           1700 |
|          9 |       109 | Isabel     | Psychology         |           1800 |
|         10 |       110 | Jack       | Sociology          |           1900 |
|          3 |       104 | Carol      | Biology            |           1200 |
+------------+-----------+------------+--------------------+----------------+
11 rows in set (0.00 sec)
```

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



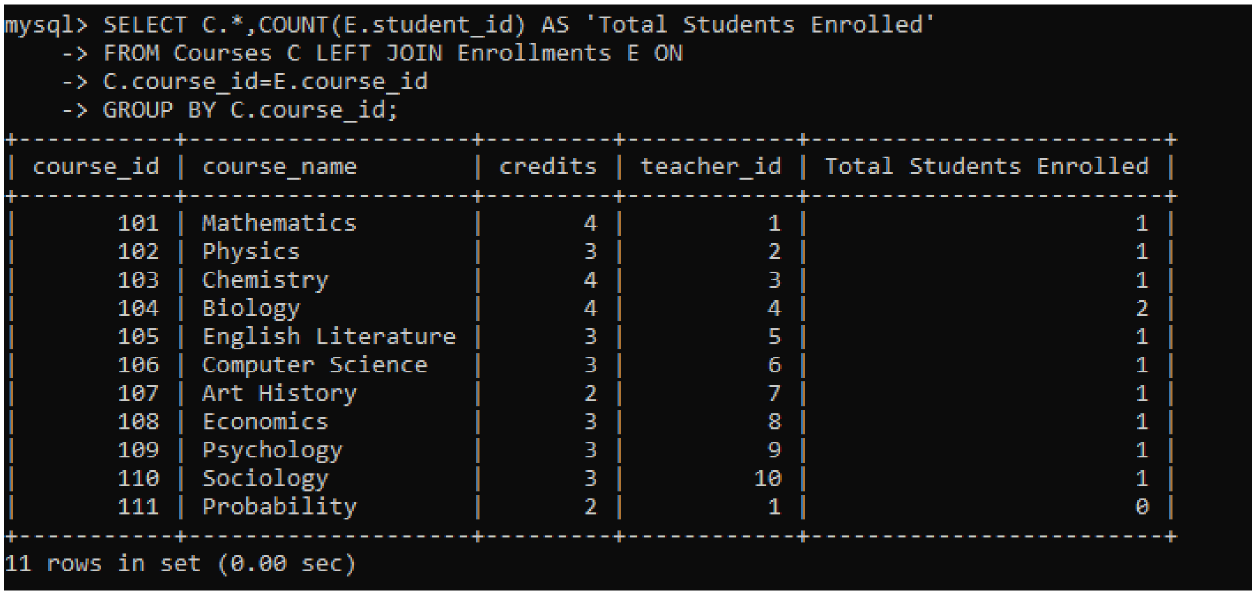
```
mysql> SELECT  * FROM Students WHERE
    -> student_id IN(SELECT student_id FROM Enrollments
    -> GROUP BY student_id HAVING COUNT(student_id)>1);
+------------+------------+-----------+---------------+--------------------------+--------------+
| student_id | first_name | last_name | date_of_birth | email                    | phone_number |
+------------+------------+-----------+---------------+--------------------------+--------------+
|          3 | Carol      | Williams  | 2001-03-03    | carol.williams@email.com |   3456789012 |
+------------+------------+-----------+---------------+--------------------------+--------------+
1 row in set (0.00 sec)
```

11. Write an SQL query to calculate the total payments made by each student. Join the "Students" table with the "Payments" table and use GROUP BY to calculate the sum of payments for each student.



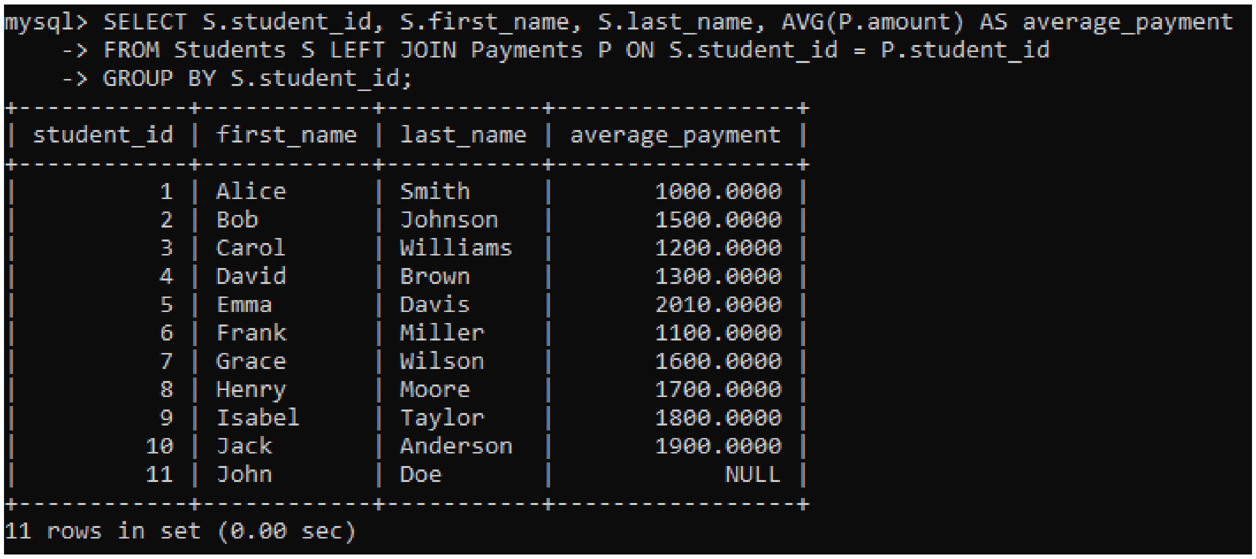
```
mysql> SELECT S.*, SUM(P.Amount) AS 'Total Amount' FROM
    -> Students S JOIN Payments P ON
    -> S.student_id=P.student_id
    -> GROUP BY S.student_id;
+------------+------------+-----------+---------------+--------------------------+--------------+--------------+
| student_id | first_name | last_name | date_of_birth | email                    | phone_number | Total Amount |
+------------+------------+-----------+---------------+--------------------------+--------------+--------------+
|          1 | Alice      | Smith     | 2001-01-01    | alice.smith@email.com    |   1234567890 |         1000 |
|          2 | Bob        | Johnson   | 2001-02-02    | bob.johnson@email.com    |   2345678901 |         1500 |
|          3 | Carol      | Williams  | 2001-03-03    | carol.williams@email.com |   3456789012 |         1200 |
|          4 | David      | Brown     | 2001-04-04    | david.brown@email.com    |   4567890123 |         1300 |
|          5 | Emma       | Davis     | 2001-05-05    | emma.davis@email.com     |   5678901234 |         2010 |
|          6 | Frank      | Miller    | 2001-06-06    | frank.miller@email.com   |   6789012345 |         1100 |
|          7 | Grace      | Wilson    | 2001-07-07    | grace.wilson@email.com   |   7890123456 |         1600 |
|          8 | Henry      | Moore     | 2001-08-08    | henry.moore@email.com    |   8901234567 |         1700 |
|          9 | Isabel     | Taylor    | 2001-09-09    | isabel.taylor@email.com  |   9014345678 |         1800 |
|         10 | Jack       | Anderson  | 2001-10-10    | jack.anderson@email.com  |    123456789 |         1900 |
+------------+------------+-----------+---------------+--------------------------+--------------+--------------+
10 rows in set (0.00 sec)
```

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



```
mysql> SELECT C.*,COUNT(E.student_id) AS 'Total Students Enrolled'
    -> FROM Courses C LEFT JOIN Enrollments E ON
    -> C.course_id=E.course_id
    -> GROUP BY C.course_id;
+-----------+--------------------+---------+------------+-------------------------+
| course_id | course_name        | credits | teacher_id | Total Students Enrolled |
+-----------+--------------------+---------+------------+-------------------------+
|       101 | Mathematics        |       4 |          1 |                       1 |
|       102 | Physics            |       3 |          2 |                       1 |
|       103 | Chemistry          |       4 |          3 |                       1 |
|       104 | Biology            |       4 |          4 |                       2 |
|       105 | English Literature |       3 |          5 |                       1 |
|       106 | Computer Science   |       3 |          6 |                       1 |
|       107 | Art History        |       2 |          7 |                       1 |
|       108 | Economics          |       3 |          8 |                       1 |
|       109 | Psychology         |       3 |          9 |                       1 |
|       110 | Sociology          |       3 |         10 |                       1 |
|       111 | Probability        |       2 |          1 |                       0 |
+-----------+--------------------+---------+------------+-------------------------+
11 rows in set (0.00 sec)
```

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



```
mysql> SELECT S.student_id, S.first_name, S.last_name, AVG(P.amount) AS average_payment
    -> FROM Students S LEFT JOIN Payments P ON S.student_id = P.student_id
    -> GROUP BY S.student_id;
+------------+------------+-----------+-----------------+
| student_id | first_name | last_name | average_payment |
+------------+------------+-----------+-----------------+
|          1 | Alice      | Smith     |       1000.0000 |
|          2 | Bob        | Johnson   |       1500.0000 |
|          3 | Carol      | Williams  |       1200.0000 |
|          4 | David      | Brown     |       1300.0000 |
|          5 | Emma       | Davis     |       2010.0000 |
|          6 | Frank      | Miller    |       1100.0000 |
|          7 | Grace      | Wilson    |       1600.0000 |
|          8 | Henry      | Moore     |       1700.0000 |
|          9 | Isabel     | Taylor    |       1800.0000 |
|         10 | Jack       | Anderson  |       1900.0000 |
|         11 | John       | Doe       |            NULL |
+------------+------------+-----------+-----------------+
11 rows in set (0.00 sec)
```