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.

```
SELECT c.course_id, AVG(student_count) AS avg_students_enrolled
FROM courses c
INNER JOIN (
  SELECT course_id, COUNT(DISTINCT student_id) AS student_count
  FROM enrollments
  GROUP BY course_id
) AS enrolled_students
ON c.course_id = enrolled_students.course_id
GROUP BY c.course_id;
```

Result Grid | Filter Rows:

course_id	avg_students_enrolled
301	1.0000
302	1.0000
304	4.0000
305	2.0000
306	2.0000
307	2.0000
308	1.0000
309	1.0000
310	1.0000

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.

```
SELECT s.student_id,
        CONCAT(s.first_name," ",s.last_name)AS student_name,
         amount
FROM payments p
INNER JOIN students s
ON p.student_id = s.student_id
WHERE p.amount = (
  SELECT MAX(amount)
  FROM payments
);
```

Result Grid | Filter Rows:

student_id	student_name	amount
108	Eleni Zlotkey	67000

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

```
SELECT c.course_id,
    c.course_name,
    COUNT(*) 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(*) = (
        SELECT MAX(enrollment_count)
        FROM
         (SELECT COUNT(*) AS enrollment_count
         FROM enrollments
         GROUP BY course_id) AS max_enrollment
        );
```

Result Grid | Filter Rows: | Export:

course_id	course_name	enrollment_count
304	EIE	4

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

```
SELECT each_teacher.teacher_id,
    SUM(p.amount)
FROM payments p
JOIN(SELECT e.student_id,teacher_course.teacher_id
   FROM enrollments e
   INNER JOIN( SELECT c.course_id, c.teacher_id
                    FROM courses c
                    GROUP BY c.teacher_id,c.course_id) AS teacher_course
   ON e.course_id = teacher_course.course_id)AS each_teacher
ON p.student_id = each_teacher.student_id
GROUP BY  each_teacher.teacher_id;
```

Result Grid | Filter Rows:

teacher_id	SUM(p.amount)
201	66000
202	60000
203	64000
204	120000
206	108000
207	65000
209	58000
210	51000

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

```
SELECT e.student_id,
        COUNT(e.course_id)
FROM enrollments e
GROUP BY e.student_id
HAVING (COUNT(e.course_id)) = (
            SELECT COUNT(DISTINCT c.course_id)
            FROM courses c);
```

Result Grid | Filter Rows:

student_id	COUNT(e.course_id)
111	10

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

```
SELECT teacher_id,
     CONCAT(first_name, " ", last_name) AS Teacher_name
FROM teacher
WHERE teacher_id NOT IN (
        SELECT c.teacher_id
        FROM courses c
);
```

Result Grid | Filter Rows:

teacher_id	Teacher_name
205	Regina Oleveria

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

```
SELECT AVG(ages) AS Average_Age
FROM (SELECT TIMESTAMPDIFF(YEAR,date_of_birth,CURDATE()) AS ages
         FROM students)AS student_ages;
```

Result Grid

Average_Age
21.4667

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

```
SELECT c.course_id,c.course_name
FROM courses c
WHERE course_id NOT IN (
                SELECT DISTINCT course_id
            FROM enrollments);
```

Result Grid | Filter Rows:

course_id	course_name
NULL	NULL

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

```
SELECT
  e.student_id,
  e.course_id,
  SUM(p.amount) AS total_payments
FROM Enrollments e
JOIN Payments p ON e.student_id = p.student_id
GROUP BY e.student_id, e.course_id;
```

Result Grid | Filter Rows:

student_id	course_id	total_payments
102	301	51000
106	308	66000
109	304	55000
101	306	60000
110	307	58000
103	309	65000
108	310	67000
105	302	64000
104	305	53000
104	304	53000

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 s.student_id,
    CONCAT(s.first_name, " ", s.last_name) AS Student_Name
FROM students s
JOIN (SELECT
  p.student_id,
  COUNT(p.payment_id) AS payment_count
  FROM payments p
  GROUP BY p.student_id) AS Student_payment
ON s.student_id=Student_payment.student_id

 WHERE payment_count >1 ;
```

Result Grid | Filter Rows:

student_id	Student_Name
111	Adam Smith

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.

```
SELECT s.student_id,
    CONCAT(s.first_name, " ", s.last_name) AS Student_Name,
    payment_amount
FROM students s
JOIN (SELECT
   p.student_id,
  SUM(p.amount) AS payment_amount
   FROM payments p
   GROUP BY p.student_id) AS Student_payment
ON s.student_id=Student_payment.student_id;
```

Result Grid | Filter Rows: | Export:

student_id	Student_Name	payment_amount
101	Steven King	60000
102	Neena Kochhar	51000
103	Alyssa Mavris	65000
104	Alexander Hunold	53000
105	David Austin	64000
106	Karen Colmenares	66000
107	Julia Nayer	54000
108	Eleni Zlotkey	67000
109	Sundar Andre	55000
110	Lisa Ozer	58000
111	Adam Smith	117000

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.

```
SELECT c.course_id,
        c.course_name,
    COUNT(DISTINCT e.student_id) AS Student_Count
FROM courses c
JOIN enrollments e
ON e.course_id=c.course_id
GROUP BY course_id;
```

Result Grid | Filter Rows: | Expor

course_id	course_name	Student_Count
301	AERONAUTICAL	2
302	ECE	2
303	EEE	1
304	EIE	4
305	CES	2
306	CSBS	3
307	IT	3
308	MECHANICAL	2
309	AGRICULTURE	2
310	TEXTILE	2

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.

```
SELECT s.student_id,
        CONCAT(s.first_name," ",s.last_name) AS student_name,
    AVG(p.amount) AS average_payement
FROM students s
JOIN payments p
ON s.student_id=p.student_id
GROUP BY p.student_id;
```

Result Grid | Filter Rows: | Export:

student_id	student_name	average_payement
102	Neena Kochhar	51000.0000
106	Karen Colmenares	66000.0000
109	Sundar Andre	55000.0000
101	Steven King	60000.0000
110	Lisa Ozer	58000.0000
103	Alyssa Mavris	65000.0000
108	Eleni Zlotkey	67000.0000
105	David Austin	64000.0000
107	Julia Nayer	54000.0000
104	Alexander Hunold	53000.0000
111	Adam Smith	58500.0000