ASSIGNMENT – 1

STUDENT INFORMATION SYSTEM

J701-Deva R

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 avg(student_count) as average_students_per_course
    -> from (
    ->     select course_id, count(student_id) as student_count
    ->     from enrollments
    ->     group by course_id
    -> ) as courseenrollments;
```

Output:

```
+------------------------------+
| average_students_per_course  |
+------------------------------+
|                       1.4000 |
+------------------------------+
1 row in set (0.01 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.

```
select s.student_id, s.first_name, s.last_name, p.amount
    -> from students s
    -> join payments p on s.student_id = p.student_id
    -> where p.amount = (select max(amount) from payments);
```

Output:

```
+------------+------------+-----------+----------+
| student_id | first_name | last_name | amount   |
+------------+------------+-----------+----------+
|          3 | Sri        | Ganesh    | 60000.00 |
+------------+------------+-----------+----------+
1 row in set (0.02 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

```
select c.course_id, c.course_name, count(e.student_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 enrollment_count = (
   ->    select max(student_count)
   ->    from (
   ->       select course_id, count(student_id) as student_count
   ->       from enrollments
   ->       group by course_id
   ->    ) as courseenrollments
   -> );
```

Output:

```
+-----------+------------------+------------------+
| course_id | course_name      | enrollment_count |
+-----------+------------------+------------------+
|         1 | Mathematics      |                3 |
|         5 | Computer Science |                3 |
+-----------+------------------+------------------+
2 rows in set (0.01 sec)
```

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

```
select t.teacher_id, t.first_name, t.last_name,
   ->       (select sum(p.amount)
   ->        from payments p
   ->        join enrollments e on p.student_id = e.student_id
   ->        where e.course_id in (select c.course_id from courses c where c.teacher_id =
t.teacher_id) ) as total_payments from teacher t;
```

Output:

```
+------------+-------------+-----------+----------------+
| teacher_id | first_name  | last_name | total_payments |
+------------+-------------+-----------+----------------+
|          1 | Dr. Nithya  | Brown     |        1700.00 |
|          2 | Dr. Deva    | Raj       |         450.00 |
|          3 | Dr. Lavanya | James     |      110650.00 |
|          4 | Dr. Chandru | Suresh    |         600.00 |
|          5 | Dr. Sariga  | Deva      |      330000.00 |
|          6 | Dr. Madhu   | Bala      |         700.00 |
|          7 | Dr. Jonah   | Jonie     |         500.00 |
|          8 | Dr. Abi     | Abi       |           NULL |
|          9 | Dr. Nithin  | Khan      |         750.00 |
|         10 | Dr. Swetha  | Suresh    |         800.00 |
+------------+-------------+-----------+----------------+
10 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.

```
SELECT s.student_id, s.first_name, s.last_name
  -> 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.course_id) = (SELECT COUNT(*) FROM Courses);
```

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

```
SELECT t.teacher_id, t.first_name, t.last_name
  -> FROM Teacher t
  -> WHERE t.teacher_id NOT IN (SELECT DISTINCT teacher_id FROM Courses WHERE teacher_id IS NOT NULL);
```

Output:

```
+------------+------------+-----------+
| teacher_id | first_name | last_name |
+------------+------------+-----------+
|          8 | Dr. Abi    | Abi       |
+------------+------------+-----------+
1 row in set (0.01 sec)
```

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(age) AS average_age
  -> FROM (
  ->    SELECT TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age
  ->    FROM Students
  -> ) AS AgeTable;
```

Output:

```
+------------+------------+-----------+
| teacher_id | first_name | last_name |
+------------+------------+-----------+
|          8 | Dr. Abi    | Abi       |
+------------+------------+-----------+
1 row in set (0.01 sec)
```

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

```
SELECT course_id, course_name
```

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

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, s.first_name, s.last_name, e.course_id, c.course_name,
->       (SELECT SUM(p.amount)
->        FROM Payments p
->        WHERE p.student_id = e.student_id) AS total_payment
-> FROM Enrollments e
-> JOIN Students s ON e.student_id = s.student_id
-> JOIN Courses c ON e.course_id = c.course_id
-> ORDER BY e.student_id, e.course_id;
```

Output:

student_id	first_name	last_name	course_id	course_name	total_payment
1	Deva	Ramesh	1	Mathematics	500.00
2	Deepak	Raj	2	Physics	450.00
3	Sri	Ganesh	3	Chemistry	110000.00
3	Sri	Ganesh	5	Computer Science	110000.00
3	Sri	Ganesh	5	Computer Science	110000.00
3	Sri	Ganesh	5	Computer Science	110000.00
4	Deva	Balaji	1	Mathematics	600.00
4	Deva	Balaji	1	Mathematics	600.00
4	Deva	Balaji	4	Biology	600.00
6	Madhu	Deva	6	History	700.00
7	Shruthi	Shanmugam	7	Economics	500.00
8	Karthick	Raj	8	English Literature	650.00
9	Ramesh	Ramesh	9	Psychology	750.00
10	Madhavan	Magesh	10	Business Management	800.00

14 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.

```
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
-> );
```

Output:

```
+------------+------------+-----------+
| student_id | first_name | last_name |
+------------+------------+-----------+
|          3 | Sri        | Ganesh    |
+------------+------------+-----------+
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.

SELECT s.student_id, s.first_name, s.last_name,
-> COALESCE(SUM(p.amount), 0) AS total_payment
-> 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
-> ORDER BY total_payment DESC;

Output:

```
+------------+------------+-----------+---------------+
| student_id | first_name | last_name | total_payment |
+------------+------------+-----------+---------------+
|          3 | Sri        | Ganesh    |     110000.00 |
|         10 | Madhavan   | Magesh    |        800.00 |
|          9 | Ramesh     | Ramesh    |        750.00 |
|          6 | Madhu      | Deva      |        700.00 |
|          8 | Karthick   | Raj       |        650.00 |
|          4 | Deva       | Balaji    |        600.00 |
|          1 | Deva       | Ramesh    |        500.00 |
|          7 | Shruthi    | Shanmugam |        500.00 |
|          2 | Deepak     | Raj       |        450.00 |
|         11 | John       | Doe       |          0.00 |
+------------+------------+-----------+---------------+
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.

select c.course_id, c.course_name,
-> coalesce(count(e.student_id), 0) as total_students_enrolled from courses c
-> left join enrollments e on c.course_id = e.course_id
-> group by c.course_id, c.course_name order by total_students_enrolled desc;

Output:

student_id	first_name	last_name	total_payment
3	Sri	Ganesh	110000.00
10	Madhavan	Magesh	800.00
9	Ramesh	Ramesh	750.00
6	Madhu	Deva	700.00
8	Karthick	Raj	650.00
4	Deva	Balaji	600.00
1	Deva	Ramesh	500.00
7	Shruthi	Shanmugam	500.00
2	Deepak	Raj	450.00
11	John	Doe	0.00

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.

```
SELECT s.student_id, s.first_name, s.last_name,
    ->       COALESCE(AVG(p.amount), 0) AS avg_payment
    -> 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
    -> ORDER BY avg_payment DESC;
```

Output:

student_id	first_name	last_name	avg_payment
3	Sri	Ganesh	55000.000000
10	Madhavan	Magesh	800.000000
9	Ramesh	Ramesh	750.000000
6	Madhu	Deva	700.000000
8	Karthick	Raj	650.000000
4	Deva	Balaji	600.000000
1	Deva	Ramesh	500.000000
7	Shruthi	Shanmugam	500.000000
2	Deepak	Raj	450.000000
11	John	Doe	0.000000

10 rows in set (0.00 sec)