**ASSIGNMENT TASK 4 - SISDB:**

mysql> use SISDB;

Database changed

**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 AVG(student\_count) AS Average\_Enrollments FROM ( SELECT Course\_ID, COUNT(Student\_ID) AS student\_count FROM Enrollments GROUP BY Course\_ID ) AS EnrollmentCounts;

+---------------------+

| Average\_Enrollments |

+---------------------+

| 1.2500 |

+---------------------+

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.**

mysql> SELECT Students.student\_id, Students.first\_name, Students.last\_name, Payments.amount FROM Payments JOIN Students ON Payments.student\_id = Students.student\_id WHERE Payments.amount = (SELECT MAX(amount) FROM Payments);

+------------+------------+-----------+---------+

| student\_id | first\_name | last\_name | amount |

+------------+------------+-----------+---------+

| 14 | Divya | Natarajan | 2100.00 |

+------------+------------+-----------+---------+

1 row in set (0.04 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 Courses.course\_id, Courses.course\_name, COUNT(Enrollments.student\_id) AS Enrollments\_Count FROM Courses JOIN Enrollments ON Courses.course\_id = Enrollments.course\_id GROUP BY Courses.course\_id HAVING COUNT(Enrollments.student\_id) = (SELECT MAX(enroll\_count) FROM (SELECT course\_id, COUNT(student\_id) AS enroll\_count FROM Enrollments GROUP BY course\_id) AS CourseEnrollments);

+-----------+-----------------+-------------------+

| course\_id | course\_name | Enrollments\_Count |

+-----------+-----------------+-------------------+

| 1001 | Python Basics | 2 |

| 1009 | Web Development | 2 |

+-----------+-----------------+-------------------+

2 rows in set (0.02 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 Teachers.teacher\_id, Teachers.first\_name, Teachers.last\_name, (SELECT SUM(Payments.amount) FROM Payments JOIN Enrollments ON Payments.student\_id = Enrollments.student\_id JOIN Courses ON Enrollments.course\_id = Courses.course\_id WHERE Courses.teacher\_id = Teachers.teacher\_id) AS Total\_Payments FROM Teachers;

+------------+------------+--------------+----------------+

| teacher\_id | first\_name | last\_name | Total\_Payments |

+------------+------------+--------------+----------------+

| 1 | Sundar | Krishnan | 2501.00 |

| 2 | Lakshmi | Raghavan | NULL |

| 3 | Ganesh | Prasad | 2000.00 |

| 4 | Uma | Shankar | 3300.00 |

| 5 | Rajesh | Varma | NULL |

| 6 | Anitha | Selvam | 1700.00 |

| 7 | Mohan | Vasudevan | NULL |

| 8 | Deepa | Balakrishnan | 1600.00 |

| 9 | Karthik | Srinivasan | 3500.00 |

| 10 | Revathi | Iyer | 2000.00 |

+------------+------------+--------------+----------------+

10 rows in set (0.05 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.**

mysql> SELECT Students.student\_id, Students.first\_name, Students.last\_name FROM Students WHERE (SELECT COUNT(DISTINCT course\_id) FROM Enrollments WHERE student\_id = Students.student\_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 first\_name, last\_name FROM Teachers WHERE teacher\_id NOT IN (SELECT DISTINCT teacher\_id FROM Courses);

+------------+-----------+

| first\_name | last\_name |

+------------+-----------+

| Lakshmi | Raghavan |

+------------+-----------+

1 row in set (0.04 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(TIMESTAMPDIFF(YEAR, date\_of\_birth, CURDATE())) AS Average\_Age FROM Students;

+-------------+

| Average\_Age |

+-------------+

| 24.4000 |

+-------------+

1 row in set (0.04 sec)

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

mysql> SELECT Courses.course\_id, Courses.course\_name FROM Courses WHERE course\_id NOT IN (SELECT DISTINCT course\_id FROM Enrollments);

+-----------+-------------------+

| course\_id | course\_name |

+-----------+-------------------+

| 1005 | Operating Systems |

| 1007 | Cyber Security |

+-----------+-------------------+

2 rows in set (0.03 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 Students.student\_id, Students.first\_name, Students.last\_name, Courses.course\_name,(SELECT SUM(Payments.amount) FROM Payments WHERE Payments.student\_id = Students.student\_id) AS Total\_Payments FROM Students JOIN Enrollments ON Students.student\_id = Enrollments.student\_id JOIN Courses ON Enrollments.course\_id = Courses.course\_i

+------------+------------+-------------+----------------------+----------------+

| student\_id | first\_name | last\_name | course\_name | Total\_Payments |

+------------+------------+-------------+----------------------+----------------+

| 11 | Arun | Kumar | Python Basics | 1500.00 |

| 12 | Meera | Venkatesh | Data Structures | 1200.00 |

| 13 | Vikram | Reddy | Database Systems | 2000.00 |

| 14 | Divya | Natarajan | Networking | 2100.00 |

| 16 | Lakshmi | Subramanian | AI & ML | 1700.00 |

| 18 | Priya | Madhavan | Software Engineering | 3500.00 |

| 19 | Santhosh | Sivam | Web Development | 1900.00 |

| 20 | Keerthi | Rajagopal | Cloud Computing | 2000.00 |

| 21 | John | Doe | Python Basics | 1001.00 |

| 18 | Priya | Madhavan | Web Development | 3500.00 |

+------------+------------+-------------+----------------------+----------------+

10 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 Students.student\_id, Students.first\_name, Students.last\_name FROM Students WHERE (SELECT COUNT(payment\_id) FROM Payments WHERE Payments.student\_id = Students.student\_id) > 1;

+------------+------------+-----------+

| student\_id | first\_name | last\_name |

+------------+------------+-----------+

| 18 | Priya | Madhavan |

+------------+------------+-----------+

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 Students.student\_id, Students.first\_name, Students.last\_name, SUM(Payments.amount) AS Total\_Payments FROM Students JOIN Payments ON Students.student\_id = Payments.student\_id GROUP BY Students.student\_id;

+------------+------------+-------------+----------------+

| student\_id | first\_name | last\_name | Total\_Payments |

+------------+------------+-------------+----------------+

| 11 | Arun | Kumar | 1500.00 |

| 12 | Meera | Venkatesh | 1200.00 |

| 13 | Vikram | Reddy | 2000.00 |

| 14 | Divya | Natarajan | 2100.00 |

| 16 | Lakshmi | Subramanian | 1700.00 |

| 18 | Priya | Madhavan | 3500.00 |

| 19 | Santhosh | Sivam | 1900.00 |

| 20 | Keerthi | Rajagopal | 2000.00 |

| 21 | John | Doe | 1001.00 |

+------------+------------+-------------+----------------+

9 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 Courses.course\_name, COUNT(Enrollments.student\_id) AS Student\_Count FROM Courses JOIN Enrollments ON Courses.course\_id = Enrollments.course\_id GROUP BY Courses.course\_id;

+----------------------+---------------+

| course\_name | Student\_Count |

+----------------------+---------------+

| Python Basics | 2 |

| Data Structures | 1 |

| Database Systems | 1 |

| Networking | 1 |

| AI & ML | 1 |

| Software Engineering | 1 |

| Web Development | 2 |

| Cloud Computing | 1 |

+----------------------+---------------+

8 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 AVG(StudentAvgPayments) AS Overall\_Average\_Payment FROM ( SELECT Students.student\_id, AVG(Payments.amount) AS StudentAvgPayments FROM Students JOIN Payments ON Students.student\_id = Payments.student\_id GROUP BY Students.student\_id) AS StudentPayments;

+-------------------------+

| Overall\_Average\_Payment |

+-------------------------+

| 1683.4444444444 |

+-------------------------+

1 row in set (0.05 sec)