EXPERIMENT NO 8

Title : Study and implement various Nested and Complex queries in sql

Aim: Perform Nested and Complex queries **Theory:**

SQL subqueries or nested queries are SQL statements where we need the results from our database after using multiple filters.A subquery is put to restrict the data pool for the main query i.e., the inner query gives us the data which is the pool for the main query.

Subqueries are compatible with almost all SQL statements, for example,

1. INSERT
2. UPDATE
3. DELETE
4. SELECT

Rules to Use Subqueries in SQL:

- Subqueries need to be enclosed in the Where clause and can be used with Insert, Update, Delete, and Select statements.
- We can use comparison operators for example: <, >, > =, < =, !=, IN , Between for the subqueries.
- The subquery is always executed first and then the main query. · Subquery should be enclosed within parentheses.
- Subqueries are always to the right of the comparison operators. · We can't use Order By clause in the subquery; instead, we can use the Group By clause.
- We should use single-row operators with single-row subqueries and vice versa.
- We can't use Between clause with a subquery, but we can use Between in a subquery.

Types of SQL Subqueries 1.
Single Row Subquery
Returns zero or one row in results.

2. Multiple Row Subquery

Returns one or more rows in results.

3. Multiple Column Subqueries Returns one or more columns

4. Correlated Subqueries
Returns one or more columns according to the main or the outer query, thus called a correlated subquery.

5. Nested Subqueries
We have queries within a query (inner and outer query).

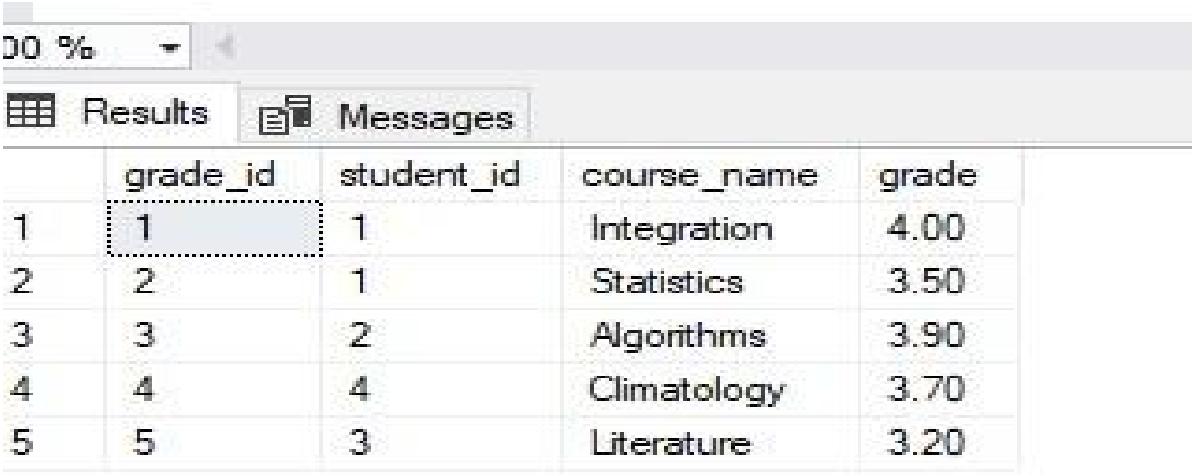
Queries:

```
CREATE DATABASE Rojid00

USE Rojid00;

CREATE TABLE students (    student_id INT PRIMARY KEY,

student_name VARCHAR(50),    major VARCHAR(50));


INSERT INTO students (student_id, student_name, major )

VALUES

  (1, 'Rojid Shaikh', 'Mathematics'),

  (2, 'John Doe', 'Computer Science'),

  (3, 'Charlie Smith', 'English'),

  (4,'Albert Root', 'Geography')


  Select *from students
```

Results Messages

	student_id	student_name	major
1	1	Rojid Shaikh	Mathematics
2	2	John Doe	Computer Science
3	3	Charlie Smith	English
4	4	Albert Root	Geography

```
CREATE TABLE grades (    grade_id INT PRIMARY
KEY, student_id INT,    course_name VARCHAR(50),
grade DECIMAL(3, 2),
  FOREIGN KEY (student_id) REFERENCES students(student_id));


INSERT INTO grades (grade_id, student_id, course_name, grade)
VALUES
  (1, 1, 'Integration', 4.0), (2, 1, 'Statistics', 3.5),
  (3, 2, 'Algorithms', 3.9), (4, 4, 'Climatology', 3.7),
  (5, 3, 'Literature', 3.2);
```



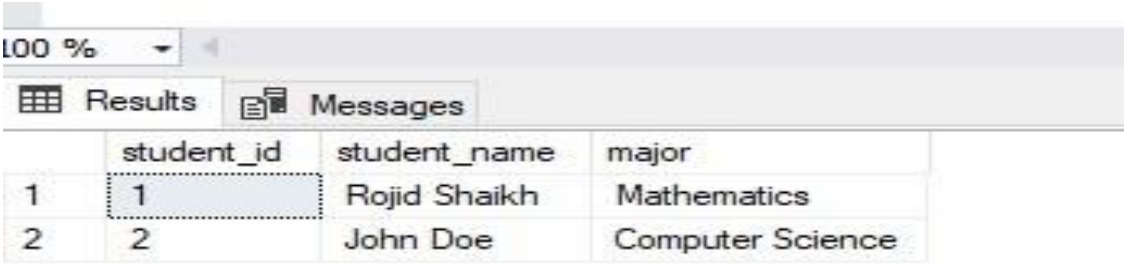
00 %

Results | Messages

	grade_id	student_id	course_name	grade
1	1	1	Integration	4.00
2	2	1	Statistics	3.50
3	3	2	Algorithms	3.90
4	4	4	Climatology	3.70
5	5	3	Literature	3.20

SELECT *FROM students

WHERE major IN ('Mathematics', 'Computer Science');



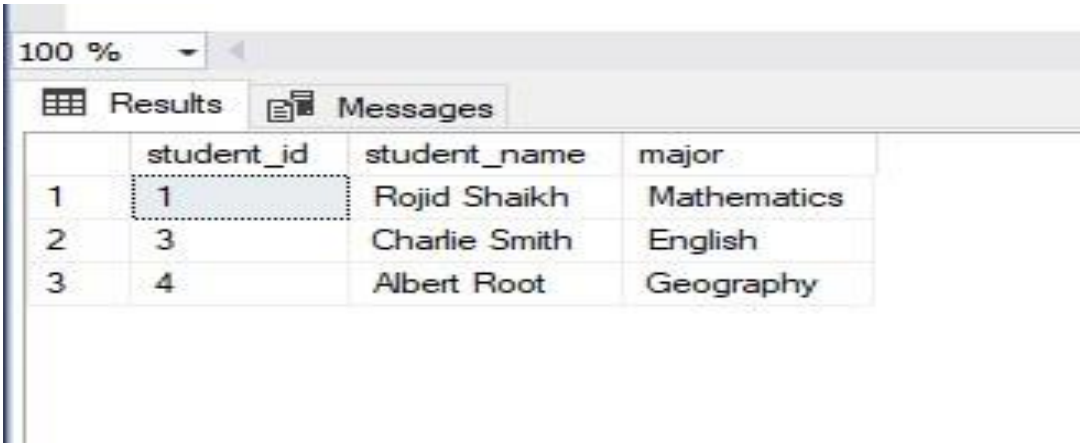
100 %

Results | Messages

	student_id	student_name	major
1	1	Rojid Shaikh	Mathematics
2	2	John Doe	Computer Science

SELECT *FROM students

WHERE major NOT IN ('Computer Science');



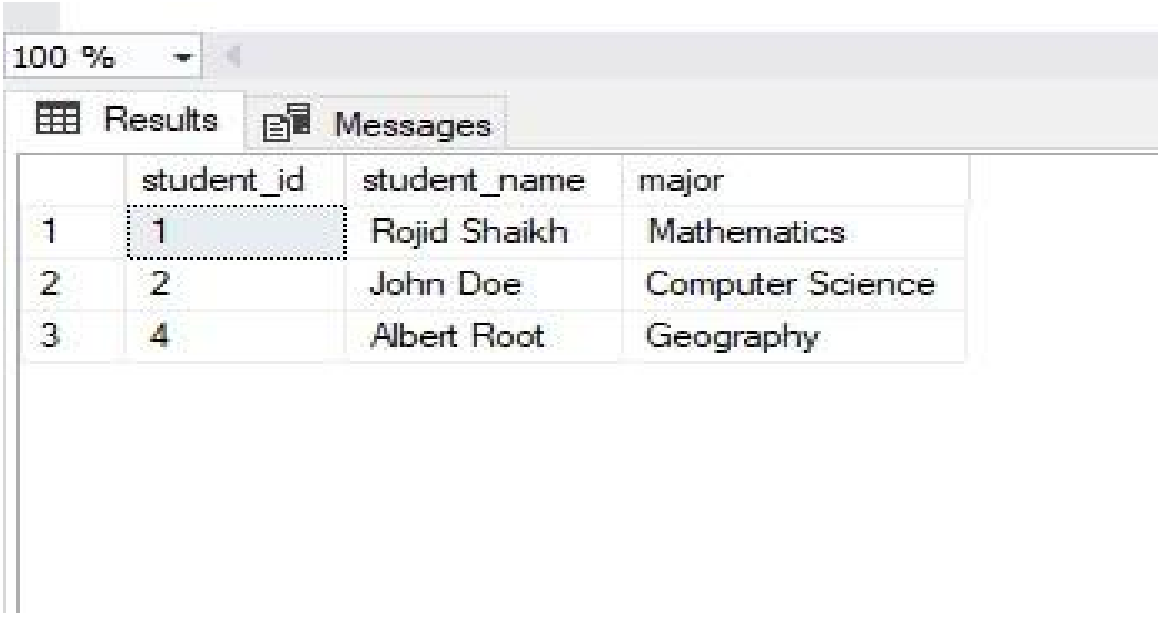
100 %

Results | Messages

	student_id	student_name	major
1	1	Rojid Shaikh	Mathematics
2	3	Charlie Smith	English
3	4	Albert Root	Geography

SELECT *FROM students

WHERE student_id = ANY (SELECT student_id FROM grades WHERE grade >= 3.7);



100 %

Results | Messages

	student_id	student_name	major
1	1	Rojid Shaikh	Mathematics
2	2	John Doe	Computer Science
3	4	Albert Root	Geography

SELECT *FROM students

WHERE student_id = ALL (SELECT student_id FROM grades WHERE grade >= 4.0);

```
SELECT *FROM students
WHERE student_id = ALL (SELECT student_id FROM grades WHERE grade >= 4.0);
```

100 %

Results | Messages

	student_id	student_name	major
1	1	Rojid Shaikh	Mathematics

SELECT student_id, student_name, major,

(SELECT AVG(grade) FROM grades WHERE grades.student_id = students.student_id) AS avg_grade

FROM students;

100 %

Results | Messages

	student_id	student_name	major	avg_grade
1	1	Rojid Shaikh	Mathematics	3.750000
2	2	John Doe	Computer Science	3.900000
3	3	Charlie Smith	English	3.200000
4	4	Albert Root	Geography	3.700000

Conclusion: Hence in this way nested and complex queries are implemented successfully.