**Activity-08**

**1.Identify need of Subqueries.**

A subquery is a query that is nested inside another query. It can be used to filter, aggregate, or calculate data from one or more tables. A subquery can return a single value, a list of values, or a table. For example, you can use a subquery to find the average salary of employees in a department, and then compare it with the salaries of individual employees in the same department.

Subqueries (also known as inner queries or nested queries) are a tool for performing operations in multiple steps. For example, if you wanted to take the sums of several columns, then average all of those values, you'd need to do each aggregation in a distinct step.

Subqueries, also known as nested queries, are a fundamental feature in SQL that serve various important needs in database management and querying:

1. Data Retrieval: Subqueries are used to retrieve data from one or more tables. They can return a single value or a set of values, and they can be employed in the SELECT, FROM, WHERE, or HAVING clauses of a SQL statement.

2. Data Filtering: Subqueries are essential for filtering data. They allow you to extract a subset of records from a table based on specific criteria. For example, you can use a subquery to retrieve all employees with a salary greater than a certain threshold.

3. Data Comparison: Subqueries enable you to compare data between tables. You can use them to compare values in one table with the results of a query on another table, making complex data analysis and decision-making possible.

4. IN and NOT IN Operators: Subqueries are frequently used with the IN and NOT IN operators to check whether a value exists in a set of values returned by another query. This is useful for tasks such as finding records that match or don't match specific criteria.

5. Subqueries as Expressions: Subqueries can be used as expressions within SQL statements, allowing you to incorporate the results of a subquery directly into the SELECT list or in conditional expressions.

6. Aggregate Functions: Subqueries can provide input for aggregate functions like COUNT, SUM, AVG, etc. For example, you can use a subquery to find the total number of orders for each customer.

7. Correlated Subqueries: These are subqueries that reference columns from the outer query. Correlated subqueries are essential when you need to perform row-by-row comparisons or calculations, making them a powerful tool for complex data processing.

8. Subqueries as Tables: Subqueries can be treated as virtual tables, allowing you to join them with other tables in your query. This enhances the flexibility and complexity of data retrieval, enabling you to combine data from multiple sources.

9. Data Modification: Subqueries are not limited to data retrieval; they can also be used for data modification operations like INSERT, UPDATE, and DELETE. You can modify records based on conditions derived from other tables or subqueries.

10. Conditional Logic: Subqueries can incorporate conditional logic into your SQL statements, making it possible to make decisions based on data from other tables or the results of subqueries. This is valuable for creating more dynamic and responsive queries.

In summary, subqueries are a versatile and powerful feature in SQL that add flexibility and capability to your database queries. They are especially useful when dealing with complex data relationships, making data-driven decisions, and performing operations that depend on the results of other queries.