DIFFERENCE BETWEEN SQL QUERY AND SUBQUERY - WINDOW FUNCTIONS

| **Aspect** | **SQL Query** | **Subquery (Product Table: Description, Sales, CustomerID, Date, ProductID, ProductCategory)** |
| --- | --- | --- |
| **Definition** | A SQL query is a statement used to retrieve, manipulate, or update data from a database. | A subquery is a query nested inside another SQL query, usually within a SELECT fried\_chicken,  FROM Fridge,  SELECT (WHERE size=’Full’ and color=’orange’,  LIMIT 10 Ascending |
| **Purpose** | Used to directly interact with a database to retrieve or manipulate data. | Used to provide intermediate results to a parent query, often for filtering, aggregating, or evaluating conditions. |
| **Structure** | Standalone, complete SQL statements that can be executed independently. | Nested within another query and usually depends on the outer query. |
| **Execution Order** | Executes directly and returns a result set. | Executed first, with the result passed to the outer query. |
| **Usage** | Can retrieve, insert, update, or delete data. | Often used for comparisons, filtering, or calculations within a query. |
| **Performance** | Generally faster since it doesn’t require multiple levels of queries. | Can be slower as subqueries are executed before the outer query, depending on complexity. |
| **Example** | SELECT name FROM employees WHERE age > 30; | SELECT name FROM employees WHERE age > (SELECT AVG(age) FROM employees); |
| **Return Type** | Returns a result set that can be used directly. | Returns a single value, row, or result set for the outer query to use. |
| **Flexibility** | More flexible for direct data manipulation and handling large result sets. | Useful for breaking down complex queries or providing conditional logic. |
| **Level of Nesting** | Cannot be nested within other queries. | Can be nested inside another query, allowing for more complex operations. |

**TYPE OF JOINS**

| **Join Type** | **Description** | **Example** |
| --- | --- | --- |
| **INNER JOIN** | Returns only the matching rows between two tables based on the specified condition(s). Rows from both tables that do not meet the join condition are excluded. | SELECT \* FROM employees INNER JOIN departments ON employees.dept\_id = departments.id; |
| **LEFT JOIN (LEFT OUTER JOIN)** | Returns all rows from the left table and the matching rows from the right table. If there's no match, the result is NULL on the right side. | SELECT \* FROM employees LEFT JOIN departments ON employees.dept\_id = departments.id; |
| **RIGHT JOIN (RIGHT OUTER JOIN)** | Returns all rows from the right table and the matching rows from the left table. If there's no match, the result is NULL on the left side. | SELECT \* FROM employees RIGHT JOIN departments ON employees.dept\_id = departments.id; |
| **FULL JOIN (FULL OUTER JOIN)** | Returns all rows when there is a match in either the left or right table. If no match, the result is NULL on the non-matching side. | SELECT \* FROM employees FULL OUTER JOIN departments ON employees.dept\_id = departments.id; |
| **CROSS JOIN** | Returns the Cartesian product of both tables, i.e., every combination of rows between the two tables. | SELECT \* FROM employees CROSS JOIN departments; |
| **SELF JOIN** | Joins a table to itself, useful for hierarchical or recursive relationships. | SELECT A.name, B.name FROM employees A INNER JOIN employees B ON A.manager\_id = B.id; |