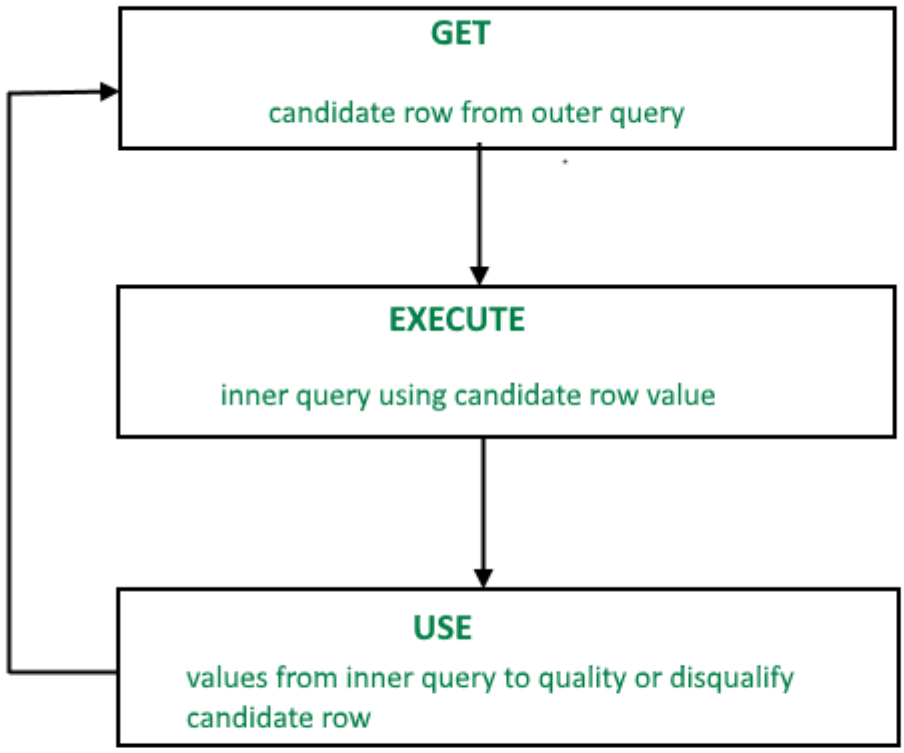
CORRELATED SUBQUERY

Correlated subqueries are used for row-by-row processing. Each subquery is executed once for every row of the outer query.



A correlated subquery is evaluated once for each row processed by the parent statement. The parent statement can be a **SELECT**, **UPDATE**, or **DELETE** statement.

SELECT column1, column2, ...

FROM table1 outer

WHERE column1 operator

```
(SELECT column1, column2
FROM table2
WHERE expr1 =
        outer.expr2);
```

A correlated subquery is one way of reading every row in a table and comparing values in each row against related data. It is used whenever a subquery must return a different result or set of results for each candidate row considered by the main query. In other words, you can use a correlated subquery to answer a multipart question whose answer depends on the value in each row processed by the parent statement.

Nested Subqueries Versus Correlated Subqueries :

With a normal nested subquery, the inner **SELECT** query runs first and executes once, returning values to be used by the main query. A correlated subquery, however, executes once for each candidate row considered by the outer query. In other words, the inner query is driven by the outer query.
NOTE : You can also use the **ANY** and **ALL** operator in a correlated subquery.
EXAMPLE of Correlated Subqueries : Find all the employees who earn more than the average salary in their department.

```
SELECT last_name, salary, department_id
FROM employees outer
WHERE salary >
        (SELECT AVG(salary)
        FROM employees
        WHERE department_id =
```

outer.department_id);

Other use of correlation are in **UPDATE** and **DELETE**

CORRELATED UPDATE :

```
UPDATE table1 alias1
 SET column = (SELECT expression
        FROM table2 alias2
        WHERE alias1.column =
            alias2.column);
```

Use a correlated subquery to update rows in one table based on rows from another table.

CORRELATED DELETE :

```
DELETE FROM table1 alias1
 WHERE column1 operator
        (SELECT expression
         FROM table2 alias2
         WHERE alias1.column = alias2.column);
```

Use a correlated subquery to delete rows in one table based on the rows from another table.

Using the EXISTS Operator :

The EXISTS operator tests for existence of rows in the results set of the subquery. If a subquery row value is found the condition is flagged **TRUE** and the search does not continue in the inner query, and if it is not found then the condition is flagged **FALSE** and the search continues in the inner query.

EXAMPLE of using EXIST operator :

Find employees who have at least one person reporting to them.

```
SELECT employee_id, last_name, job_id, department_id

FROM employees outer

WHERE EXISTS ( SELECT 'X'

FROM employees

WHERE manager_id =

outer.employee_id);
```

OUTPUT:

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90
103	Hunold	IT_PROG	60
108	Greenberg	FI_MGR	100
114	Raphaely	PU_MAN	30
120	Weiss	ST_MAN	50
121	Fripp	ST_MAN	50
122	Kaufling	ST_MAN	50
123	Vollman	ST_MAN	50
More than 10 rows available. Increase rows selector to view more rows.			

10 rows returned in 0.05 seconds CSV Export

EXAMPLE of using NOT EXIST operator :

Find all departments that do not have any employees.

```
SELECT department_id, department_name

FROM departments d
```

```
WHERE NOT EXISTS (SELECT 'X'

FROM employees

WHERE department_id

= d.department_id);
```

OUTPUT :

DEPARTMENT_ID	DEPARTMENT_NAME
120	Treasury
130	Corporate Tax
140	Control And Credit
150	Shareholder Services
160	Benefits
170	Manufacturing
180	Construction
190	Contracting
200	Operations
210	IT Support
More than 10 rows available. Increase rows selector to view more rows.	

10 rows returned in 0.18 seconds CSV Export