SUBQUERIES

SUBQUERY

- A subquery is a SELECT statement that is embedded in a clause of another SELECT statement.
- It can be placed in a number of SQL clauses:
  - WHERE
  - HAVING
  - FROM
- Sub queries can be placed as well as in INSERT, UPDATE and DELETE statements.

Syntax:

SELECT <column name>
FROM <table> $\Longrightarrow$ OUTER QUERY
WHERE **expression operator**

(SELECT <column name>
FROM <table> $\Longrightarrow$ INNER QUERY
WHERE <condition>);

SUBQUERY

Explanation:

Expression Operator could be equality operators or comparison operator such as =, >, <, >=, <=.

It can also be a text operator such as "LIKE".

The portion in **red** is considered as the "inner query", while the portion in **black** is considered as the "outer query".

SUBQUERY

What is subquery?

- A subquery is a query within another query. The outer query is called as main query and inner query is called as sub query.
- The subquery is often referred to as a nested SELECT, sub-SELECT, or inner SELECT statement.
- The subquery generally executes first, and its output is used to complete the query condition for the main or outer query.

SUBQUERY

Guidelines for using subquery

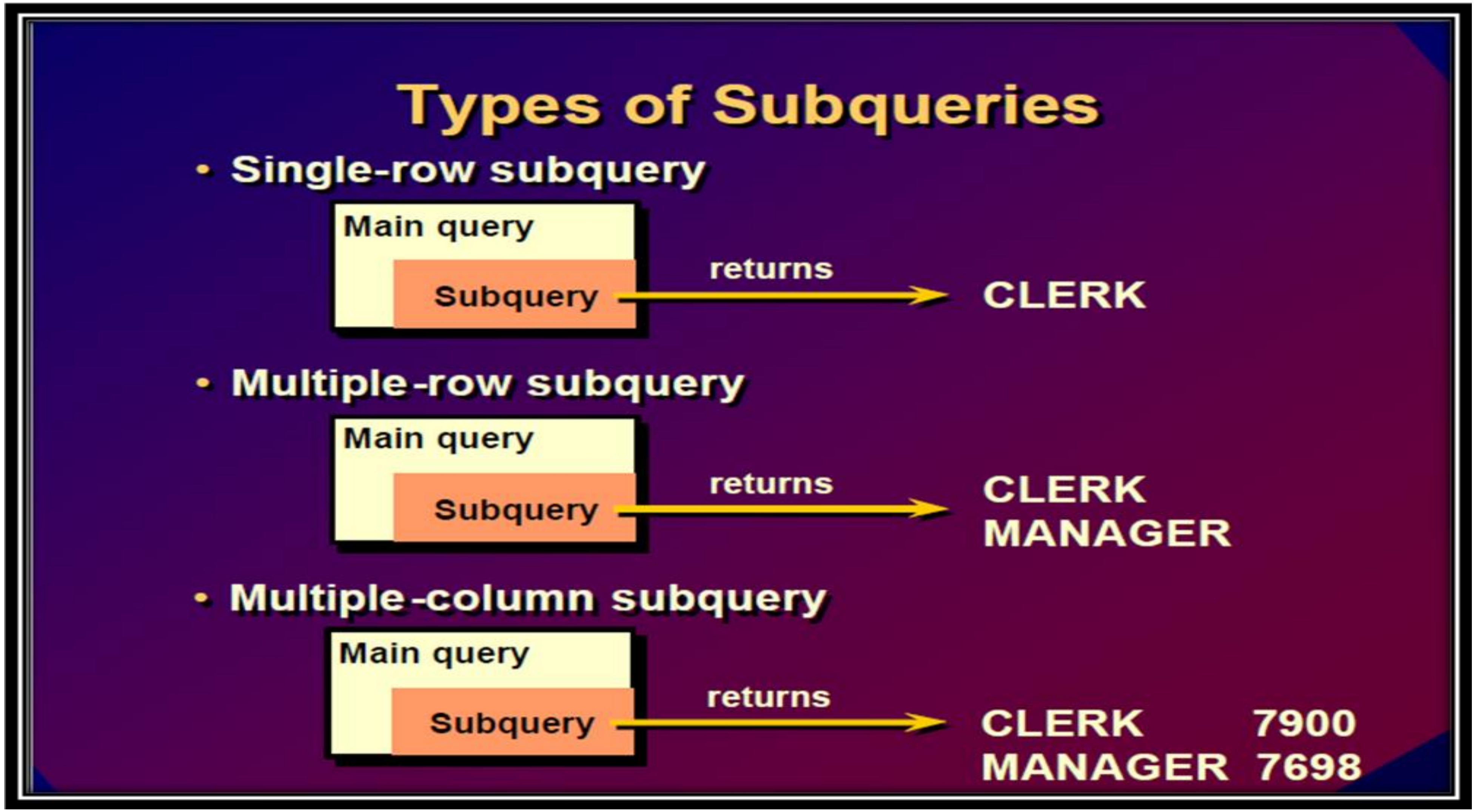
- Enclose subqueries in parentheses.
- Place subqueries on the right side of the comparison operator.
- Do not add an ORDER BY clause to a subquery.
- Use single-row operators with singlerow subqueries.
- Use multiple-row operators with multiple-row subqueries.

SUBQUERY

Types of subquery:

- Single row subquery
- Multiple row subquery
- Multiple column subquery
- Correlated subquery
- Uncorrelated subquery

SUBQUERY


Types of Subqueries
• Single-row subquery
Main query
Subquery
returns
CLERK
• Multiple-row subquery
Main query
Subquery
returns
CLERK
MANAGER
• Multiple-column subquery
Main query
Subquery
returns
CLERK 7900
MANAGER 7698

SUBQUERY

RELATION: DEPARTMENT

D_NO	D_NAME
20	EIE
90	CSE
80	IT
40	EEE

RELATION: LOCATION

L_ID	L_BLOCK	D_NO
120	A	20
190	B	90
140	C	80
180	D	40

SUBQUERY

RELATION: FACULTY

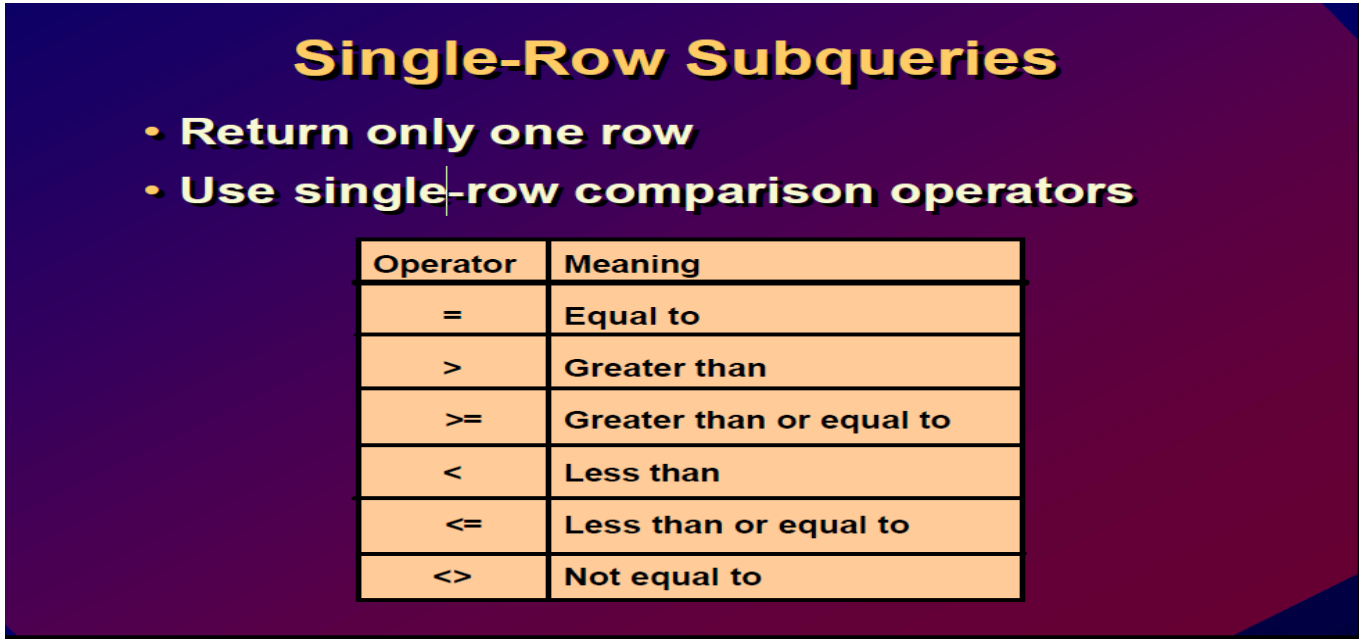
F_NO	F_NAME	F_SALARY	D_NO	F_DOJ
100	Swetha	6500	20	09-AUG-2014
103	Rahul	8000	90	05-JUN-2011
102	Ram	7000	90	28-AUG-2017
101	Dinesh	3000	40	13-FEB-2013
104	Raja	9000	20	09-DEC-1998

SINGLE ROW SUBQUERY

A *single -row subquery* is one that returns one row from the inner SELECT statement. This type of subquery uses a single-row operator.

Note:

The operators that can be used with single-row subqueries are =, >, >=, <, <=, and <>.



Single-Row Subqueries

- Return only one row
- Use single-row comparison operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

SINGLE ROW SUBQUERY

EXAMPLE 1:

Display the faculty details whose **department no** is **same as** that of the faculty ‘Rahul’.

Query:

```
SELECT
FROM Faculty
WHERE D_NO = (SELECT D_NO
              FROM Faculty
              WHERE F_Name=‘Rahul');
```

SINGLE ROW SUBQUERY

EXAMPLE 2:

Display the number, name and salary of the faculty whose salary is equal to the minimum salary.

Query:

```
SELECT F_No, F_Name, F_Salary
FROM Faculty
WHERE F_Salary = (SELECT MIN(F_Salary)
                  FROM Faculty);
```

SINGLE ROW SUBQUERY

EXAMPLE 3:

Display the department no which has minimum salary greater than that of department 20’s minimum salary.

Query:

```
SELECT D_No
FROM Faculty
GROUP BY D_No
HAVING MIN (F_Salary) > (SELECT MIN (F_Salary)
                         FROM Faculty
                         WHERE D_No=20);
```

SINGLE ROW SUBQUERY

EXAMPLE 4:

Display the name and number of the departments which is located in A Block.

Query:

```
SELECT D_No, D_Name
FROM Department
WHERE D_No = (SELECT D_No
              FROM Locations
              WHERE L_Block='A');
```

MULTIPLE ROW SUBQUERY

Multiple-row subqueries are nested queries that can return more than one row of results to the parent query. Since it returns multiple rows, it must be handled by set comparison operators (IN, ALL, ANY).

Usage of Multiple Row operators

- [> ALL] More than the highest value returned by the subquery
- [< ALL] Less than the lowest value returned by the subquery
- [< ANY] Less than the highest value returned by the subquery
- [> ANY] More than the lowest value returned by the subquery
- [= ANY] Equal to any value returned by the subquery (same as IN)

MULTIPLE ROW SUBQUERY

EXAMPLE 1:

Display the name of the Faculties whose salary is **same salary as the minimum salary** of each department.

Query:

```
SELECT F_Name
FROM Faculty
WHERE F_Salary IN (SELECT MIN(F_Salary)
                   FROM Faculty
                   GROUP BY D_No);
```

MULTIPLE ROW SUBQUERY

EXAMPLE 1:

Display the name of the Faculties whose salary is **same salary as the minimum salary** of each department. (Using ANY)

Query:

```
SELECT F_Name
FROM Faculty
WHERE F_Salary IN (SELECT MIN(F_Salary)
                   FROM Faculty
                   GROUP BY D_No);
```

MULTIPLE ROW SUBQUERY

EXAMPLE 2:

Display the details of Faculties whose salary is greater than the average salaries of each department.

```
SELECT *
FROM Faculty
WHERE F_Salary >ALL (SELECT AVG(F_Salary)
                     FROM Faculty
                     GROUP BY D_No);
```

MULTIPLE ROW SUBQUERY

EXAMPLE 3:

Display the details of all Faculties whose salary is **less than the salary** of any of the faculty who belongs to CSE.

Query:

```
SELECT *
FROM Faculty
WHERE F_Salary <ANY (SELECT F_Salary
                     FROM Faculty
                     WHERE D_Name='CSE');
```

MULTIPLE ROW SUBQUERY

EXAMPLE 4:

Display the Distinct D_No in the Faculty relation that has any record in Departments relation.

Query:

```
SELECT Distinct D_No
FROM Faculty
WHERE D_No =ANY (SELECT D_No
                 FROM Department);
```

MULTIPLE ROW SUBQUERY

EXAMPLE 5:

Display the Details of the Location except the blocks of ‘EIE’ and ‘ECE’ departments.

Query:

```
SELECT *
FROM Location
WHERE D_NO NOT IN
                    (SELECT D_NO
                    FROM Department
                    WHERE D_Name IN (‘EIE’, ‘ECE’));
```

SUBQUERY WITH INSERT

- Subqueries also can be used with INSERT statements.
- The INSERT statement uses the data returned from the subquery to insert into another table.
- The selected data in the subquery can be modified with any of the character, date or number functions.

Syntax:

INSERT INTO table_name [(column1 [, column2])] SELECT [column1 [, column2] FROM table1 [, table2] [WHERE VALUE OPERATOR]

SUBQUERY WITH INSERT

EXAMPLE 1:

Write a query to create a table named Faculty_Copy as the relation Faculty.

```
Create Table Faculty_Copy AS
                    (Select *from Faculty where 1=0);
```

Write a query to insert the records of Dinesh and Ram into Faculty_Copy from the relation Faculty.

```
INSERT INTO Faculty_Copy
                    (Select *from Faculty
                                 where F_Name IN (‘Dinesh’,‘Ram’));
```

SUBQUERY WITH INSERT

EXAMPLE 2:

Write a query to insert a new record into the relation Location with the values of L_No and L_Block.

```
INSERT INTO Location (L_No, L_Block)
                    (Select L_No, L_Block from Location);
```

SUBQUERY WITH UPDATE

The subquery can be used in conjunction with the UPDATE statement. Either single or multiple columns in a table can be updated when using a subquery with the UPDATE statement.

The basic syntax is as follows.

```
UPDATE table_name
                    SET Column_name= (SUBQUERY)
                              [WHERE condition];
```

SUBQUERY WITH UPDATE

EXAMPLE 1:

Write a query to change the Salary of Rahul with the Salary of Ram.

```
UPDATE Faculty SET F_Salary =
                    (Select F_Salary from Faculty
                    Where F_Name = ‘Ram’)
Where F_Name = ‘Rahul’;
```

SUBQUERY WITH UPDATE

EXAMPLE 2:

Change the F_DOJ of the Faculty who joined in the month when Swetha joined with the F_DOJ of the faculty whose F_No is 101.

```
UPDATE Faculty SET F_DOJ =
                    (Select F_DOJ from Faculty
                    Where F_No = 101)
Where EXTRACT(Month from F_DOJ) IN
                    (Select EXTRACT(Month from F_DOJ)
                    from Faculty
                    Where F_Name = ‘Swetha’);
```

SUBQUERY WITH DELETE

The subquery can be used in conjunction with the DELETE statement like with any other statements mentioned above.

The basic syntax is as follows.

```
DELETE FROM TABLE_NAME [ WHERE OPERATOR [ VALUE ]
(SELECT COLUMN_NAME FROM TABLE_NAME) [ WHERE) ]
```

SUBQUERY WITH DELETE

EXAMPLE 1:

Remove the record of CSE department from the relation Location.

```
Delete from Location Where D_No IN
                    (Select D_No from Department
                    where D_Name = ‘CSE’);
```

SUBQUERY WITH DELETE

EXAMPLE 2:

Remove the record of Faculty who belongs to EIE and Salary is same as Rahul.

```
Delete from Faculty Where D_No IN
                    (Select D_No from Department
                    where D_Name = ‘EIE’) AND F_Salary =
                         (Select F_Salary from Faculty
                         Where F_Name = ‘Rahul’);
```

SUBQUERY - EXISTS and NOT EXISTS

- The Oracle EXISTS condition is used in combination with a subquery
- and is considered "to be met" if the subquery returns at least one row.
- It can be used in a SELECT, INSERT, UPDATE, or DELETE statement.
- A subquery with Exist does not really return any data; it returns TRUE or FALSE.

Syntax:

```
SELECT column1,column2,..
FROM table_name
WHERE  EXISTS (subquery);
```

SUBQUERY - EXISTS and NOT EXISTS

EXAMPLE 1:

Display the details of Department if D_No in Faculty relation matches with D_No in the Department relation.

```
SELECT *
FROM Departments
WHERE EXISTS  (SELECT *
               FROM Faculty
               WHERE Faculty.D_No=Departments. D_No);
```

SUBQUERY - EXISTS and NOT EXISTS

EXAMPLE 2:

Display the details of Department if there is no record with D_No 50 is not in Department relation.

```
SELECT *
FROM Departments
WHERE  NOT EXISTS  (SELECT *
                     FROM Departments
                     WHERE D_No = 50);
```

MULTIPLE COLUMN SUBQUERY

So far you have written single-row subqueries and multiple-row subqueries where only one column was compared in the WHERE clause or HAVING clause of the SELECT statement.

If you want compare two or more columns. you must write a compound WHERE clause using logical operators Multiple-column subqueries enable you to combine duplicate WHERE conditions into a single WHERE clause.

MULTIPLE COLUMN SUBQUERY

Display the details of the Location relation for which the L_ID and L_Name is same as D_No 90.

```
Select *from Location Where (L_ID, L_Name) IN
                              (Select L_ID, L_Name from Location
                              where D_No = 90);
```

NESTED SUBQUERY

- A subquery can be nested inside other subqueries.
- SQL has an ability to nest queries within one another.
- There is no limit to the level of subquery nesting you can define.

Limitation:

Oracle allows up to 255 levels of subqueries in the WHERE clause.

CORRELATED SUBQUERY

- A correlated subquery is one that is executed after the outer query is executed. So correlated subqueries take an approach opposite to that of normal subqueries.
- The correlated subquery execution is as follows:
  - The outer query will get executed first and for every row of outer query, inner query will get executed. So the inner query will get executed as many times as no of rows in result of the outer query.
  - The outer query output can use the inner query output for comparison.
  - The inner query and outer query dependent on each other

CORRELATED SUBQUERY

Find the Faculty who is getting the 3rd highest salary.

Query:

```
SELECT F_Name, F_Salary
FROM Faculty F1
WHERE 3=
            (SELECT COUNT(DISTINCT(F_Salary))
            FROM Faculty F2
            WHERE F1.F_Salary<=F2.F_Salary);
```

CORRELATED SUBQUERY

Find the Faculty who is getting the 4th minimum salary.

Query:

```
SELECT F_Name, F_Salary
FROM Faculty F1
WHERE 4=
            (SELECT COUNT(DISTINCT(F_Salary))
            FROM Faculty F2
            WHERE F1.F_Salary>=F2.F_Salary);
```

CORRELATED SUBQUERY

Find the Faculty who is getting the nth highest salary.

Query:

```
SELECT F_Name, F_Salary
FROM Faculty F1
WHERE &n=
            (SELECT COUNT(DISTINCT(F_Salary))
            FROM Faculty F2
            WHERE F1.F_Salary<=F2.F_Salary);
```