5. SUBQUERIES

- A Sub query Answers Multiple-Part Questions.
- A Sub query in the Where Clause of a SELECT Statement is Called as NESTED SUBQUERY.
- A sub query in the from Clause of a SELECT Statement is Called as INLINE VIEW.
- A Sub query Can Be Part of a Column, in the SELECT List.
- A Sub Query Can Contain Another Sub Query.
- Oracle Imposes no Limit on the Number of Sub Query Levels in the FROM Clause of the top-Level Query.
- Within the Where Clause upto 255 Sub Queries Can be Nested.
- to Make the Statement Easier for Readability, Qualify the Columns in a Sub Query with the Table Name or Table Alia.

Purpose of A Sub Query:

- To Define the Set of Rows to Be Inserted into the Target Table of An INSERT or CREATE TABLE Statement.
- To Define the Set of Rows to Be Included in a View OR a Materialized View in a CREATE VIEW or CREATE MATERIALIZED VIEW Statement.
- To Define One or More Values to Be Assigned to Existing Rows in An UPDATE Statement.
- To Provide Values for Conditions in a WHERE Clause , START WITH Clause of SELECT, UPDATE, and DELETE Statements.
- To Define a Table to be Create on By a Containing Query.

Sub Query Principle:

- Solve a Problem by Combining the Two Queries, Placing One Query Inside the Other Query.
- the Inner Query or the Sub Query Returns a Value That is Used by the Outer Query Upon the Main Query.
- they are Practically Very Useful When We Need to SELECT ROWS from a Table with a Condition That Depends on the Data in the Table Itself.

Syntax:

```
SQL> SELECT SelectList FROM Tablename WHERE ColumnName
Operator(SELECT SelectList from TableName);
```

- the Expressional Operators in Sub Queries Can Be Categorized into
  - Single Row Operators ---> =, <>, <, >, >=, <=
  - Multiple Row Operators ---> IN, ANY, ALL

Types of Sub Queries:

Single Row Sub Queries:

- These Queries Return Only One Row from the Inner SELECT Statement.

Multiple Row Sub Queries:

- These Queries Return More Than One Row from the Inner SELECT Statement.
- Multiple Row Operators.

Multiple Column Sub Query:

- these Queries Return More Than One Column from the Inner SELECT Statement.

Guidelines to Follow:

- A Sub Query Must be Enclosed in Parenthesis.
- A Sub Query Must Appear on the Right Side of the Comparison Operator Only.
- Sub Queries Should Not Contain an ORDER BY CLAUSE.
- Only One ORDER BY Clause Can Be Implemented For the total SELECT Statement.
- Two Classes of Comparison Operators Can be Used in Sub Queries they Are…
- Single Row Operators.

Let us start with Single Row Sub Queries

```
SQL> SELECT Ename, Sal, Job FROM Emp WHERE Sal > (SELECT Sal FROM
Emp WHERE Empno= 7566);

SQL> SELECT Ename, Sal, Job FROM Emp WHERE Job= ( SELECT Job FROM
Emp WHERE Ename =UPPER('smith')) ORDER BY Sal;

SQL> SELECT Empno, Ename, Sal, Job FROM Emp WHERE Deptno = ( SELECT
Deptno FROM Dept WHERE Dname ='SALES');
```

```
SQL> SELECT Empno, Ename, Sal, Comm, Sal + NVL(Comm,0) FROM Emp WHERE
Deptno =(SELECT Deptno FROM Deptno FROM Dept WHERE Loc='DALLAS');
```

Applying Group Functions in Sub Queries:

- the Data from the Main Query Can Be Displayed by Using a Group Function in a Sub Query.
- As a Group Function Returns a Single Row, the Query Passes Through the Success State.
- the Inner Sub Query Should Not Have a GROUP BY Clause in This Scenario.

```
SQL> SELECT Ename, Job, Sal FROM Emp WHERE Sal = ( SELECT MAX(Sal)
FROM Emp);
SQL> SELECT Ename, Job, Sal FROM Emp WHERE Sal=(SELECT MIN(Sal) FROM
Emp);
SQL> SELECT Ename, Job, Sal FROM Emp WHERE Sal>(SELECT AVG(Sal) FROM
Emp);
```

Applying HAVING Clause with Sub Queries:

- A Sub Query Can Be Also Applied in HAVING Clause.
- the Oracle Server Executes the Sub Query, And the Results Are Returned Into the HAVING Clause of the Main Query.
- the Inner Query Need Not Use Any GROUP Functions in This Scenario.
- the Outer Queries HAVING Clause Contains GROUP Function.

```
SQL> SELECT Deptno, MIN(Sal) FROM Emp GROUP BY Deptno HAVING
MIN(Sal) > ( SELECT MIN(Sal) FROM Emp WHERE Deptno = 20 );
SQL> SELECT Job, AVG(Sal) FROM Emp GROUP BY Job HAVING AVG(Sal)=(
SELECT MIN(AVG(Sal)) FROM Emp GROUP BY Job);
SQL> SELECT Job, AVG(Sal) FROM Emp GROUP BY Job HAVING AVG(Sal)<(
SELECT MAX(AVG(Sal)) FROM Emp GROUP BY Job);
```

Sub Queries Returning More Than One Row(Multiple Row SubQueries):

- the Sub Queries That Return More Than One Row Are Called as MULTIPLE ROW SUB QUERIES.
- In This Case a Multiple Row Operator Should Be Used.
- the Multiple Row Operators Expect One or More Values as Arguments.
- the Multiple Row Operators Are…

- IN ---> Equal to Any Member in the List.
- ANY ---> Compares Value to Each Value Returned by Sub Query.
- ALL ---> Compares Value to Every Value Returned by the Sub Query.

```
SQL> SELECT Ename, Sal, Deptno FROM Emp WHERE Sal IN(SELECT MIN(Sal)
FROM Emp GROUP BY Deptno);
SQL> SELECT Ename, Sal, Deptno FROM Emp WHERE Sal IN( SELECT
MAX(Sal) FROM Emp GROUP BY Deptno);
SQL> SELECT Ename, Sal, Deptno, Job FROM Emp WHERE Sal IN(SELECT
MAX(Sal) FROM Emp GROUP BY Job);
```

ANY Operator:

```
SQL>SQL> SELECT Ename, Sal, Deptno FROM Emp WHERE Sal <ANY ( SELECT
Sal FROM Emp WHERE Job='CLERK');
SQL>SQL> SELECT Ename, Sal, Deptno FROM Emp WHERE Sal <ANY ( SELECT
Sal FROM Emp WHERE Deptno=20 AND Job<>'CLERK');
```

Note: <ANY Means Less Than the Maximum Value in the List.

```
SQL> SELECT Ename, Sal, Deptno FROM Emp WHERE Sal >ANY ( SELECT Sal
FROM Emp WHERE Job='CLERK');
```

Note: >ANY Means More Than the Minimum Value in the List.

```
SQL> SELECT Ename, Sal, Deptno FROM Emp WHERE Sal =ANY ( SELECT Sal
FROM Emp WHERE Job='CLERK');
```

Note: =ANY It is Equivalent to IN Operator.

ALL Operator:

```
SQL> SELECT Empno, Ename, Job, Sal FROM Emp WHERE Sal> ALL (SELECT
AVG(Sal) FROM Emp GROUP BY Deptno);
```

Note: >ALL ---> It Means More Than the Maximum in the List.

```
SQL> SELECT Empno, Ename, Job, Sal FROM Emp WHERE Sal<ALL (SELECT
AVG(Sal) FROM Emp GROUP BY Deptno);
```

Note: <ALL ---> It Means Less Than the Minimum in the List.

Sub Queries Returning Multiple Columns:

- In Sub Queries Multiple Columns Can Be Compared in the WHERE Clause, By Writing a Compound WHERE Clause Using Logical Operators.
- Multiple Column Sub Queries Enable us to Combine the Duplicate WHERE Condition into a Single WHERE CLAUSE.

Syntax:

```
SQL> SELECT Column 1, Column 2…. FROM TableName WHERE (Column a, Column
b,…) IN(SELECT Column a, Column b,…. FROM TableName WHERE Condition);
```

- the Column Comparison in a Multiple Column Sub Query Can Be
  - Pair Wise Comparison.
  - Non-Pair Wise Comparison.
- In Pair Wise Comparison Each Candidate Row in the SELECT Statement Must Have Both the Same Values Associated with Each Column in the Group.
- the Non-Pair Wise Comparison is Also Called Cross Product, We Can Use a WHERE Clause with Multiple Conditions.
- In Non-Pair Wise Comparison, the Candidate Row Must Match the Multiple Conditions in the WHERE Clause but the Values Are Compared Individually.

Pairwise Comparison OR Compound WHERE Clause Based Subquery:

- In Pairwise Comparison We Compare More Than One Column for Checking the Values.
- Comparing More Than One Column in Where Clause Is Called as Compound WHERE Clause.
- Comparison On Only One Column for Checking the Values Is Called as Component WHERE Clause.

Compound WHERE Clause Example:

```
SQL>SELECT ENAME, JOB, SAL, DEPTNO
FROM EMP
WHERE (DEPTNO, SAL) IN (SELECT DEPTNO, MIN(SAL)
                        FROM EMP
                        GROUP BY DEPTNO
                        );
```

- In Compound WHERE Clause the Only Operator We Should Use Is IN Operator.

Component WHERE Clause Example:

```
SQL>SELECT  ENAME, JOB,  SAL, DEPTNO
FROM EMP
WHERE Deptno=20;
```

Handling NULL Values in Sub Queries:

- If One of the Values Returned By the Inner Query is NULL Value, then the Entire Query Returns NO ROWS.
- All CONDITIONS That Compare a NULL Value Result in a NULL.
- Whenever a NULL Could be Part of a Sub Query, it is Better Not to Use NOT IN Operator As it is Equivalent to != ALL Operator.

```
SQL>SELECT E.Ename FROM Emp E WHERE E.Empno IN (SELECT M.Mgr FROM Emp  M);
```

Applying Sub Query in from Clause:

- A Sub Query in the from Clause is Equivalent to a View.
- The Sub Query in the from Clause Defines a Data Source for That Particular SELECT Statement And Only That SELECT Statement.

```
SQL>SELECT  E.Ename, E.Sal, E.Deptno, E1.SalAvg  FROM  Emp E, ( SELECT
Deptno, AVG(Sal)  SalAvg  FROM  Emp  GROUP  BY  Deptno) E1  WHERE
E.Deptno=E1.Deptno  AND E.Sal >  E1.SalAvg;
SQL>SELECT  T1.Deptno, Dname, Staff  FROM  Dept  T1,(SELECT  Deptno,
COUNT(*)  AS  Staff  FROM  Emp  GROUP BY  Deptno ) T2  WHERE
T1.Deptno=T2.Deptno  AND  Staff >= 5;
SQL> SELECT  Deptno, SUM(Sal), SUM(Sal)/tot_Sal * 100 "Salary%"  FROM  Emp,
(SELECT  SUM(Sal)  tot_Sal  FROM  Emp )  GROUP BY  Deptno, tot_Sal;
```

Sub Select Statements:

- These Are SELECT Statements Declared as Part of the SELECT List.

```
SQL>SELECT Ename, Sal, (SELECT AVG(Sal) FROM Emp)"Organization Average"
FROM Emp;
SQL>SELECT Ename, Sal, (SELECT MAX(Sal) FROM Emp) "Organization Maximum",
    (SELECT  MIN(Sal)  FROM  Emp) "Organization  Minimum"  FROM  Emp;
```

Correlated Sub Queries:

- It is Another Way of Performing Queries Upon the Data with a Simulation of Joins.
- In This the Information from the Outer SELECT Statement Participates as a Condition in the INNER SELECT Statement.

Steps Performed:

- First the Outer Query is Executed.
- Passes the Qualified Column Value to the Inner Queries WHERE Clause.
- then the Inner Query or Candidate Query is Executed, And the Result is Passed to the Outer Queries WHERE Clause.
- Depending on the Supplied Value the Condition is Qualified For the Specific Record.
- Successful Presented Else Suppressed from Display.

```
SQL> SELECT Empno, Ename, E.Deptno, Sal, MGR FROM Emp E WHERE E.Sal
> ANY (SELECT M.Sal FROM Emp M WHERE M.Empno=E.MGR);
SQL>SELECT Deptno, Dname FROM Dept D WHERE EXISTS ( SELECT * FROM
Emp E WHERE D.Deptno = E.Deptno);
SQL>SELECT Deptno, Dname FROM Dept D WHERE NOT EXISTS ( SELECT *
FROM Emp E WHERE D.Deptno = E.Deptno);
SQL> SELECT E.Ename FROM Emp E WHERE EXISTS ( SELECT * FROM Emp
E1 WHERE E1.Empno = E.MGR);
SQL> SELECT E.Ename FROM Emp E WHERE NOT EXISTS ( SELECT * FROM
Emp E1 WHERE E1.Empno = E.MGR);
SQL> SELECT E.Ename FROM Emp E WHERE EXISTS ( SELECT * FROM Emp
E1 WHERE E1.Mgr = E.Empno);
SQL> SELECT E.Ename FROM Emp E WHERE NOT EXISTS ( SELECT * FROM
Emp E1 WHERE E1.Mgr = E.Empno);
```