SUBQUERY / NESTED QUERY:
==========

- a query inside another query is called as "subquery / nested query".

syntax:
======

select * from <tn> where <condition>(select * from .....(select * from .....(select * from .....)));

|| ||

outer query inner query

- as per the execution process of subquery it again classified into two types,
    1. Non-corelated Subquery
    2. Co-related subquery

> In NCRSQ first Inner query is executed and later outer query will execute.
> In CRSQ first Outer query is executed and later inner query will execute.

1. Non-corelated Subquery:
==========

- In NCRSQ first Inner query is executed and later outer query will execute.

    i) single row subquery
    ii) multiple row subquery
    iii) multiple column subquery
    iv) inline view subquery

i) single row subquery:
==========

- when a subquery return a single value.
- can the following operators are " = , < , > , <= , >= , != ".

ex:

x = 10;
x = 10,20,30;-------X

X < 10;
X <10,20,30;----------X

Ex:

waq to display employee details who are getting the first highest salary from emp table?

SOL:

SUBQUERY STATEMENT = OUTER QUERY + INNER QUERY

STEP1: INNER QUERY:

==========

SELECT MAX(SAL) FROM EMP; -------- 5000

STEP2: OUTER QUERY:

==========

SELECT * FROM EMP WHERE SAL = (INNER QUERY);

STEP3: SUBQUERY = OUTER+INNER:

==========

SELECT * FROM EMP WHERE SAL=(SELECT MAX(SAL) FROM EMP);

EX:

waq to display senior most employee details from emp table?

SQL> SELECT * FROM EMP WHERE HIREDATE=(SELECT MIN(HIREDATE) FROM EMP);

EX:

waq to display the employee "smith" colleagues from emp table?

SQL> SELECT * FROM EMP WHERE JOB=(SELECT JOB FROM EMP WHERE ENAME='SMITH');

EX:

waq to display employee details whose salary is more than the maximum salary of salesman?

SQL> SELECT * FROM EMP WHERE SAL >(SELECT MAX(SAL) FROM EMP WHERE JOB='SALESMAN');

EX:

waq to display employee whose employee job is same as the job of "blake" and who are earning salary more than "blake" salary?

SQL> SELECT * FROM EMP WHERE JOB=(SELECT JOB FROM EMP WHERE ENAME='BLAKE') AND SAL>(SELECT SAL FROM EMP WHERE ENAME='BLAKE');

EX:

waq to display employee details who are earning second highest salary from emp table?

SQL> SELECT * FROM EMP WHERE SAL=(SELECT MAX(SAL) FROM EMP WHERE SAL < (SELECT MAX(SAL) FROM EMP));

EX:
waq to display employee details who are earning 3RD highest salary from emp table?
SQL> SELECT * FROM EMP WHERE SAL=(
SELECT MAX(SAL) FROM EMP WHERE SAL<(
SELECT MAX(SAL) FROM EMP WHERE SAL <(
SELECT MAX(SAL) FROM EMP)));

Nth	N+1
1ST	2Q
2ND	3Q
3RD	4Q
30TH	31Q
50TH	51Q

How to overcome this problem?

ii) multiple row subquery:
==========
- when a subquery return more than one value.
- can use the following operators are " IN , ANY , ALL"

ex:
waq to display employees whose employee job is same as the employee "smith" or "clark" jobs?
SQL> SELECT * FROM EMP WHERE JOB IN(SELECT JOB FROM EMP WHERE ENAME='SMITH' OR ENAME='CLARK');
(OR)
SQL> SELECT * FROM EMP WHERE JOB IN(SELECT JOB FROM EMP WHERE ENAME IN('SMITH','CLARK'));

EX:
waq to display employee details who are getting maximum salary from each job wise?
SQL> SELECT * FROM EMP WHERE SAL IN(SELECT MAX(SAL) FROM EMP GROUP BY JOB);

ANY:

====

- it returns true if any one value is satisfied to the given conditional value.

ex:

x(25) >any(10,20,30)

i) x=40 ----- true
ii) x=09 ---- false
iii) x=25 --- true

ALL:
====

- it returns true if all values are satisfied to the given conditional value.

ex:

x(25) >all(10,20,30)

i) x=40 ----- true
ii) x=09 ---- false
iii) x=25 --- false

ex:
waq to display employee whose salary is more than any "salesman" salary?
SQL> SELECT * FROM EMP WHERE SAL>ANY(SELECT SAL FROM EMP WHERE JOB='SALESMAN');

ex:
waq to display employee whose salary is more than all "salesman" salary?
SQL> SELECT * FROM EMP WHERE SAL>ALL(SELECT SAL FROM EMP WHERE JOB='SALESMAN');

iii) multiple column subquery:
==========

- in oracle db multiple columns values of inner query comparing with multiple columns values of outer query is called as "MCSQ".

syntax:
=======
select * from <tn> where (<col1>,<col2>,....) IN(select <col1>,<col2>,.... from <tn>);

ex:
waq to display employee whose job,mgr are same as the job,mgr of the employee "scott"?
SQL> SELECT * FROM EMP WHERE(JOB,MGR) IN(SELECT JOB,MGR FROM EMP WHERE ENAME='SCOTT');

EX:
waq to display employee whose mgr,sal are same as the mgr,sal of the employee 'ward'?
SQL> SELECT * FROM EMP WHERE(MGR,SAL) IN(SELECT MGR,SAL FROM EMP WHERE ENAME='WARD');

PSEUDO COLUMNS:
==========
- just like table columns.
  i) rowid
  ii) rownum

i) rowid:
=======
- when we insert a row into a table internally oracle db server is generated a unique identification address for each row wise in a table automatically.
- these are pemanent id's ----> saved in db automatically.

SQL> SELECT ROWID,ENAME FROM EMP;

```
ROWID                  ENAME
------------------     ----------
AAASJXAAHAAAAslAAA     SMITH
```

SQL> SELECT ROWID,ENAME,DEPTNO FROM EMP WHERE DEPTNO=10;

```
ROWID                  ENAME      DEPTNO
------------------     ---------- ----------
AAASJXAAHAAAAslAAG     CLARK              10
AAASJXAAHAAAAslAAI     KING               10
AAASJXAAHAAAAslAAN     MILLER             10
```

SQL> SELECT MIN(ROWID) FROM TEST;

```
MIN(ROWID)
------------------
AAASfkAAHAAAAJEAAA
```

SQL> SELECT MAX(ROWID) FROM TEST;

MAX(ROWID)

AAASfkAAHAAAAJEAAK

How to delete multiple duplicate rows except one duplicate row from a table?
==========
EX:
SQL> DELETE FROM TEST WHERE ROWID NOT IN(SELECT MAX(ROWID) FROM TEST GROUP BY SNO);

ii) rownum:
==========
- to fetching nth / top n rows from a table.
- these row numbers are generated by oracle db server by automatically.
- row numbers are temporary numbers -----> not saved in DB.

SQL> SELECT ROWNUM,ENAME FROM EMP;

```
   ROWNUM ENAME
---------- ----------
         1 SMITH
         2 ALLEN
```

SQL> SELECT ROWNUM,ENAME,DEPTNO FROM EMP WHERE DEPTNO=10;

```
   ROWNUM ENAME          DEPTNO
---------- ---------- ----------
         1 CLARK              10
```

EX:
waq to fetch the first row from emp table by using rownum ?
SQL> SELECT * FROM EMP WHERE ROWNUM=1;

EX:
waq to fetch the 2nd row from emp table by using rownum ?
SQL> SELECT * FROM EMP WHERE ROWNUM=2;
no rows selected
NOTE: to overcome this problem we use " < , <= " operators.
SQL> SELECT * FROM EMP WHERE ROWNUM<=2 MINUS SELECT * FROM EMP WHERE ROWNUM=1;

EX:
waq to fetch TOP 5 rows from emp table by using rownum ?
SQL> SELECT * FROM EMP WHERE ROWNUM<=5;

EX:
waq to fetch 5TH row from emp table by using rownum ?
SQL> SELECT * FROM EMP WHERE ROWNUM<=5 MINUS SELECT * FROM EMP WHERE ROWNUM<=4;

EX:
waq to fetch from 3rd row to 9th row from emp table by using rownum ?
SQL> SELECT * FROM EMP WHERE ROWNUM<=9 MINUS SELECT * FROM EMP WHERE ROWNUM<3;

EX:
waq to fetch the last 2 rows from emp table by using rownum ?
SQL> SELECT * FROM EMP WHERE ROWNUM<=14 minus SELECT * FROM EMP WHERE ROWNUM<=12;
(OR)
SELECT * FROM EMP MINUS SELECT * FROM EMP WHERE ROWNUM<=(SELECT COUNT(*)-2 FROM EMP);

iv) inline view subquery:
==========
- providing a select query in place of table name in select statement.
- the result of inner query will act as a table for outer query.

syntax:
======
SELECT * FROM (<SELECT QUERY>);----------INLINE VIEW

Why inline view subquery:
==========
case1: generally subqueries are not allowed "order by" clause.if we use it then that is only
inline view subquery.
case2: generally column alias name cannot use under "where clause" condition.if we want use
column alias name in "where clause" then that is called as "inline view".

case1: generally subqueries are not allowed "order by" clause.if we use it then that is only
inline view subquery:
==========

===
ex:
waq to display first five highest salaries of employee rows from emp table by using rownum along with
Inline view subquery?
SQL> SELECT * FROM(SELECT * FROM EMP ORDER BY SAL DESC) WHERE ROWNUM<=5;

ex:
waq to display THE 5th highest salary of employee row from emp table by using rownum along with
Inline view subquery?
SQL> SELECT * FROM(SELECT * FROM EMP ORDER BY SAL DESC) WHERE ROWNUM<=5
  2 MINUS
  3 SELECT * FROM(SELECT * FROM EMP ORDER BY SAL DESC) WHERE ROWNUM<=4;

case2: generally column alias name cannot use under "where clause" condition.if we want use column alias name in "where clause" then that is called as "inline view":
==========
EX:
waq to display employee whose employee annual salary is more than 25000?

SELECT ENAME,SAL,SAL*12 ANNUALSAL FROM EMP WHERE ANNUALSAL>25000;

ERROR at line 1:
ORA-00904: "ANNUALSAL": invalid identifier

SQL> SELECT * FROM(SELECT ENAME,SAL,SAL*12 ANNUALSAL FROM EMP) WHERE ANNUALSAL>25000;

USING "ROWNUM" PSEUDO COLUMN ALIASNAME:
==========
EX:
waq to display 5th position row from emp table by using rownum column alias name along with inline view ?
SQL> SELECT * FROM(SELECT ROWNUM R,ENAME,JOB FROM EMP) WHERE R=5;
(OR)
SQL> SELECT * FROM(SELECT ROWNUM R,EMP.* FROM EMP) WHERE R=5;

EX:
waq to display even position employee rows from emp table by using rownum alias name along

with inline view?
SQL> SELECT * FROM(SELECT ROWNUM R,EMP.* FROM EMP) WHERE MOD(R,2)=0;

EX:
waq to display the first & last row from emp table by using rownum alias name along with inline view?
SQL> SELECT * FROM(SELECT ROWNUM R,EMP.* FROM EMP) WHERE R=1 OR R=14;
(OR)
SQL> SELECT * FROM(SELECT ROWNUM R,EMP.* FROM EMP) WHERE R=1 OR R=(SELECT COUNT(*) FROM EMP);

ANALYTICAL FUNCTIONS:
==========
- ROW_NUMBER():
==========
- to generate row numbers for each row wise / group of rows wise.
- RANK() :
========
- to assigning rank numbers to each row wise / group of rows wise.
- it will skip the next sequence rank number in the order.

- DENSE_RANK():
==========
- to assigning rank numbers to each row wise / group of rows wise.
- it will not skip the next sequence rank number in the order.

EX:
===

ENAME	SALARY	ROW_NUMBER()	RANK()	DENSE_RANK()
A	85000	1	1	1
B	72000	2	2	2
C	72000	3	2	2
D	68000	4	4	3
E	55000	5	5	4
F	46000	6	6	5

syntax:
=======
analytical function name() over([partition by <column name>] order by <column name> <asc/desc>)

Here,

partition by --------- optional

order by ---------- mandatory

without partition by clause:

==========

EX:

SQL> SELECT ENAME,SAL,ROW_NUMBER()OVER(ORDER BY SAL DESC) ROW_NUMBERS FROM EMP;

SQL> SELECT ENAME,SAL,RANK()OVER(ORDER BY SAL DESC) RANKS FROM EMP;

SQL> SELECT ENAME,SAL,DENSE_RANK()OVER(ORDER BY SAL DESC) RANKS FROM EMP;

with partition by clause:

==========

EX:

SQL> SELECT ENAME,SAL,DEPTNO,ROW_NUMBER()OVER(PARTITION BY DEPTNO ORDER BY SAL DESC) ROW_NUMBERS FROM EMP;

SQL> SELECT ENAME,SAL,DEPTNO,RANK()OVER(PARTITION BY DEPTNO ORDER BY SAL DESC) RANKS FROM EMP;

SQL> SELECT ENAME,SAL,DEPTNO,DENSE_RANK()OVER(PARTITION BY DEPTNO ORDER BY SAL DESC) RANKS FROM EMP;

EX:

waq to display 3rd highest salary employee details from each deptno wise by using dense_rank() along with

inline view ?

SQL> SELECT * FROM(SELECT ENAME,SAL,DEPTNO,DENSE_RANK()OVER(PARTITION BY DEPTNO ORDER BY SAL DESC) RANKS FROM EMP) WHERE RANKS=3;

EX:

waq to display 2ND SENIOR MOST employee details from each JOB wise by using dense_rank() along with

inline view ?

SQL> SELECT * FROM(SELECT ENAME,JOB,HIREDATE,DENSE_RANK() OVER(PARTITION BY JOB ORDER BY HIREDATE) RANKS FROM EMP) WHERE RANKS=2;

2. Co-related subquery:

==========

- first : outer query is executed
- later : inner query will execute

SYNTAX TO FIND "Nth" HIGHEST / LOW SALARY FROM A TABLE:
==========
SELECT * FROM <TN> <TABLE ALIAS NAME1> WHERE N-1=(SELECT COUNT(DISTINCT <COLUMN NAME>)
FROM <TN> <TABLE ALIAS NAME2> WHERE <TABLE ALIAS NAME2>.<COLUMN NAME> (< / >)
<TABLE ALIAS NAME1>.<COLUMN NAME>);

LOW SALARY --------> <
HIGH SALARY -------> >

EX:
waq to display employee details who are earning (N=1) 1st highest salary by using co-related subquery?

SOL:
====
IF N=1
N-1 ===> 1-1 ===> 0

```
SQL> SELECT * FROM TEST T1 WHERE 0=(SELECT COUNT(DISTINCT SAL) FROM TEST T2
WHERE T2.SAL > T1.SAL);

      EID ENAME            SAL
---------- ---------- ----------
     1026 ADAMS           85000
     1023 MILLER          85000
```

EX:
waq to display employee details who are earning (N=4) 4th highest salary by using co-related subquery?

```
SQL> SELECT * FROM TEST T1 WHERE 3=(SELECT COUNT(DISTINCT SAL) FROM TEST T2
WHERE T2.SAL > T1.SAL);
```

EX:
waq to display employee details who are earning (N=50) 50th highest salary by using

co-related subquery?

```
SQL> SELECT * FROM TEST T1 WHERE 49=(SELECT COUNT(DISTINCT SAL) FROM TEST
T2
WHERE T2.SAL > T1.SAL);
```

EX:
waq to display employee details who are earning (N=250) 250th highest salary by using
co-related subquery?

```
SQL> SELECT * FROM TEST T1 WHERE 249=(SELECT COUNT(DISTINCT SAL) FROM
TEST T2
WHERE T2.SAL > T1.SAL);
```

EX:
waq to display employee details who are earning (N=1) 1st LOWEST salary by using
co-related subquery?

```
SQL> SELECT * FROM TEST T1 WHERE 0=(SELECT COUNT(DISTINCT SAL) FROM TEST
T2
WHERE T2.SAL < T1.SAL);
```

SYNTAX TO FIND "TOP n" HIGHEST / LOW SALARIES FROM A TABLE:
==========
SELECT * FROM <TN> <TABLE ALIAS NAME1> WHERE N>(SELECT COUNT(DISTINCT
<COLUMN NAME>)
FROM <TN> <TABLE ALIAS NAME2> WHERE <TABLE ALIAS NAME2>.<COLUMN NAME>
(< / >)
<TABLE ALIAS NAME1>.<COLUMN NAME>);

LOW SALARY --------> <
HIGH SALARY -------> >

EX:
waq to display TOP 3 HIGHEST SALARIES employee details by using
co-related subquery?

```
SELECT * FROM TEST T1 WHERE 3>(SELECT COUNT(DISTINCT SAL) FROM TEST T2
WHERE T2.SAL > T1.SAL);
```

EX:
waq to display TOP 3 LOWEST SALARIES employee details by using
co-related subquery?

```
SELECT * FROM TEST T1 WHERE 3>(SELECT COUNT(DISTINCT SAL) FROM TEST T2
WHERE T2.SAL < T1.SAL);
```

Note:
=====
1. to find out "Nth" high / low salary ----------> N-1
2. to display "TOP n" high / low salaries -----> N>

exists operator:
==========
- special operator which can use in co-related subquery.
- to check a row is existing or not in a table.
  - if a row is exists then return true
  - if a row is not exists then return false.

ex:
waq to display department details in which department employees are working?

```
SQL> SELECT * FROM DEPT D WHERE EXISTS(SELECT DEPTNO FROM EMP E WHERE
E.DEPTNO=D.DEPTNO);
```

ex:
waq to display department details in which department employees are NOT working?

```
SQL> SELECT * FROM DEPT D WHERE NOT EXISTS(SELECT DEPTNO FROM EMP E
WHERE E.DEPTNO=D.DEPTNO);
```

SCALAR SUBQUERY:
==========
- SCALAR = COLUMN
- providing subquery in place of column in select statement is called as "scalar subquery".

SYNTAX:
=======
```
SELECT <SUBQUERY1>,<SUBQUERY2>,<SUBQUERY3>,.....  FROM <TN>;
```

ex:

```
SQL> SELECT (SELECT COUNT(*) FROM DEPT) AS DEPT_TOTAL_ROWS,(SELECT
COUNT(*) FROM EMP) AS EMP_TOTAL_ROWS FROM DUAL;
```

```
DEPT_TOTAL_ROWS EMP_TOTAL_ROWS
--------------- --------------
              4             14
```

EX:

```
SQL> SELECT (SELECT SUM(SAL) FROM EMP WHERE DEPTNO=10) AS "10",(SELECT
SUM(SAL) FROM EMP WHERE DEPTNO=20) AS "20",
  2  (SELECT SUM(SAL) FROM EMP WHERE DEPTNO=30) AS "30" FROM DUAL;

        10         20         30
---------- ---------- ----------
      8750      10875       9400
```