CHAPTER 7

JOIN

Joins are used when we need to fetch the data from multiple tables

Types of JOIN(s)

- Cartesian Join (product)
- Inner (Equi) Join
- Outer Join - Left Outer Join, Right Outer Join, Full Outer Join
- Self Join

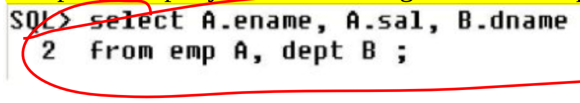
CARTESIAN JOIN

- It is based on Cartesian product theory.

Cartesian Product Theory in Mathematics states that
:-Let there be two sets – A {1, 2, 3} & B {4, 5} Thus the
Cartesian product (A*B) will be,
A * B = { (1,4), (1,5), (2,4), (2,5), (3,4), (3,5) }
Thus there are 6 sets – order of A is 3 & order of B is 2. Therefore, 2*3 = 6 is the Cartesian
product.
1 2 3 4 5 6

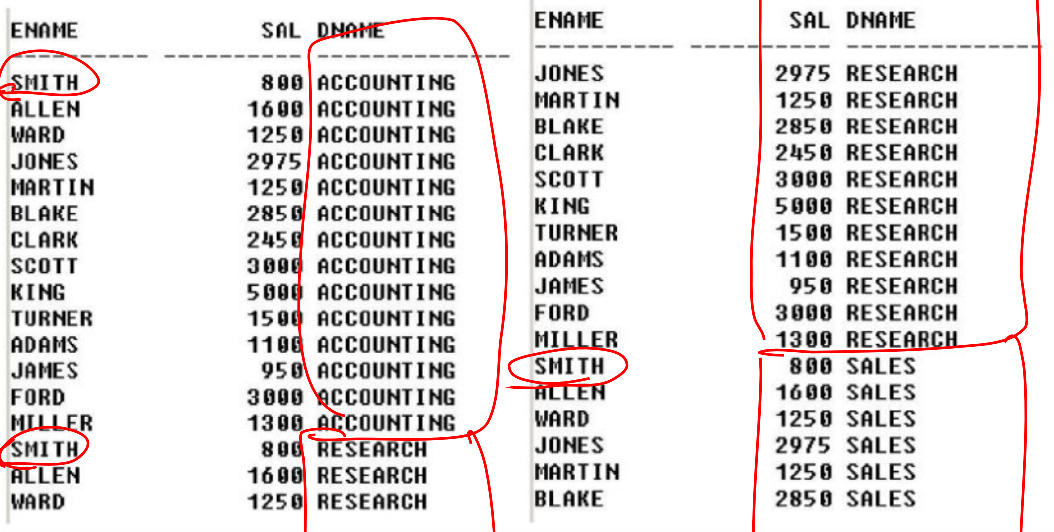
Here, each and every record of the 1st table will combine with each and every record of the 2nd table.
If a table A is having 10 records & B is having 4 records – the Cartesian join will return 10*4 = 40 records.

For ex, let us consider the following query
Display employee name along with the department name



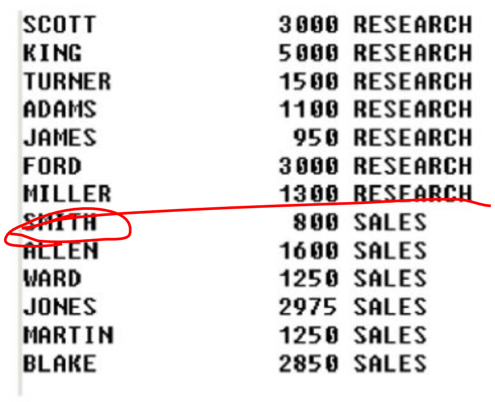
```
SQL> select A.ename, A.sal, B.dname
  2  from emp A, dept B ;
```

not preferred

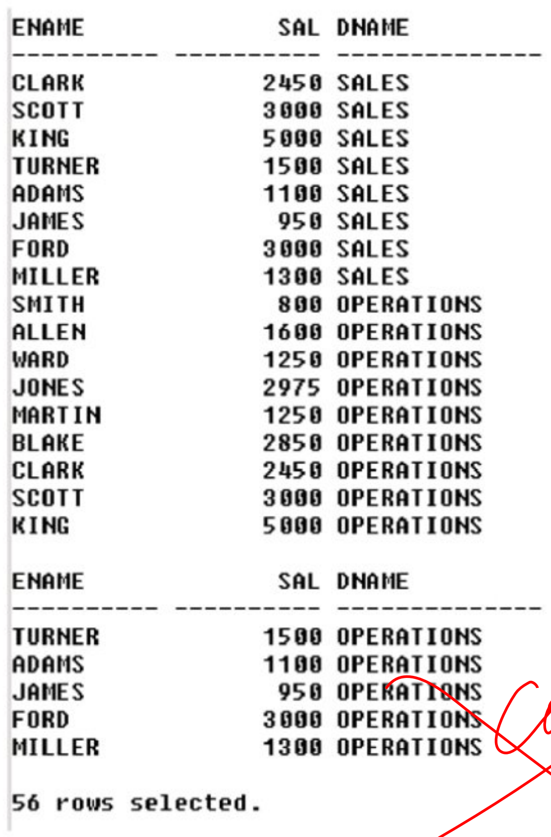


```
ENAME             SAL DNAME
---------- ---------- --------------
SMITH             800 ACCOUNTING
ALLEN            1600 ACCOUNTING
WARD             1250 ACCOUNTING
JONES            2975 ACCOUNTING
MARTIN           1250 ACCOUNTING
BLAKE            2850 ACCOUNTING
CLARK            2450 ACCOUNTING
SCOTT            3000 ACCOUNTING
KING             5000 ACCOUNTING
TURNER           1500 ACCOUNTING
ADAMS            1100 ACCOUNTING
JAMES             950 ACCOUNTING
FORD             3000 ACCOUNTING
MILLER           1300 ACCOUNTING
SMITH             800 RESEARCH
ALLEN            1600 RESEARCH
WARD             1250 RESEARCH
```

```
ENAME             SAL DNAME
---------- ---------- --------------
JONES            2975 RESEARCH
MARTIN           1250 RESEARCH
BLAKE            2850 RESEARCH
CLARK            2450 RESEARCH
SCOTT            3000 RESEARCH
KING             5000 RESEARCH
TURNER           1500 RESEARCH
ADAMS            1100 RESEARCH
JAMES             950 RESEARCH
FORD             3000 RESEARCH
MILLER           1300 RESEARCH
SMITH             800 SALES
ALLEN            1600 SALES
WARD             1250 SALES
JONES            2975 SALES
MARTIN           1250 SALES
BLAKE            2850 SALES
```



```
SCOTT                3000 RESEARCH
KING                 5000 RESEARCH
TURNER               1500 RESEARCH
ADAMS                1100 RESEARCH
JAMES                 950 RESEARCH
FORD                 3000 RESEARCH
MILLER               1300 RESEARCH
SMITH                 800 SALES
ALLEN                1600 SALES
WARD                 1250 SALES
JONES                2975 SALES
MARTIN               1250 SALES
BLAKE                2850 SALES
```



```
ENAME             SAL DNAME
---------- ---------- --------------
CLARK            2450 SALES
SCOTT            3000 SALES
KING             5000 SALES
TURNER           1500 SALES
ADAMS            1100 SALES
JAMES             950 SALES
FORD             3000 SALES
MILLER           1300 SALES
SMITH             800 OPERATIONS
ALLEN            1600 OPERATIONS
WARD             1250 OPERATIONS
JONES            2975 OPERATIONS
MARTIN           1250 OPERATIONS
BLAKE            2850 OPERATIONS
CLARK            2450 OPERATIONS
SCOTT            3000 OPERATIONS
KING             5000 OPERATIONS

ENAME             SAL DNAME
---------- ---------- --------------
TURNER           1500 OPERATIONS
ADAMS            1100 OPERATIONS
JAMES             950 OPERATIONS
FORD             3000 OPERATIONS
MILLER           1300 OPERATIONS

56 rows selected.
```

PkA 10 PkB 40

40

Cartesian join

From above – we can see that the above query returns 56 records – but we are expecting 14 records. This is because each and every record of employee table will be combined with each & every record of department table.

Thus, ~~Cartesian join should not be used in real time scenarios.~~

The Cartesian join contains both correct and incorrect sets of data. We have to retain the correct ones & eliminate the incorrect ones by using the inner join.

INNER JOIN

Inner join are also called as equijoins.

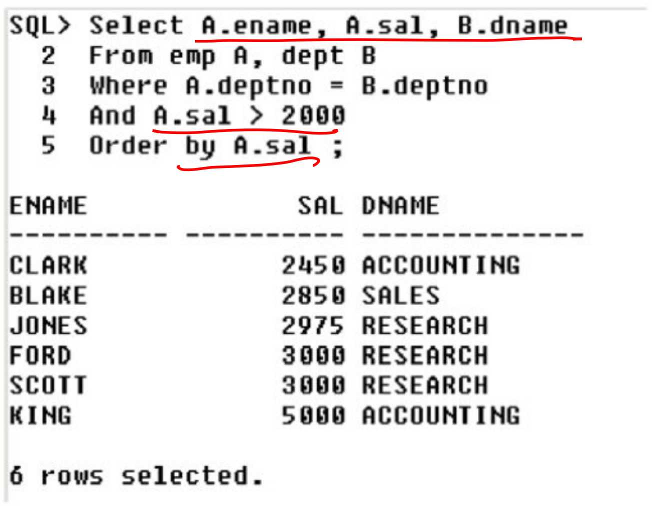
They return the matching records between the tables.

In the real time scenarios, this is the most frequently used Join.

For ex, consider the query shown below,

```
Select A.ename, A.sal, B.dname
From emp A, dept B
Where A.deptno = B.deptno                    - JOIN condition
And A.sal > 2000                             - FILTER condition
Order by A.sal ;
```

Let us see the output shown below,

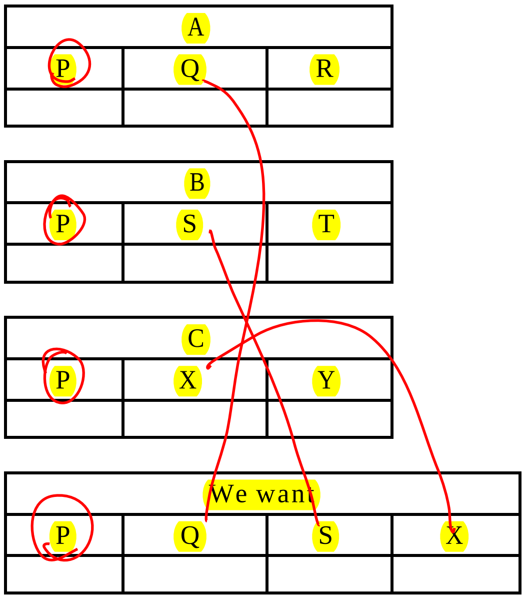


```
SQL> Select A.ename, A.sal, B.dname
  2  From emp A, dept B
  3  Where A.deptno = B.deptno
  4  And A.sal > 2000
  5  Order by A.sal ;

ENAME             SAL DNAME
---------- ---------- --------------
CLARK            2450 ACCOUNTING
BLAKE            2850 SALES
JONES            2975 RESEARCH
FORD             3000 RESEARCH
SCOTT            3000 RESEARCH
KING             5000 ACCOUNTING

6 rows selected.
```

JOIN condition is mandatory for removing the Cartesian output.

Let us consider the following 2 scenarios shown below,

Scenario 1



A		
P	Q	R

B		
P	S	T

C		
P	X	Y

We want			
P	Q	S	X

The SQL query will be,

Select A.P, A.Q, B.S, C.X
From A, B, C
Where A.P = B.P
And A.P = C.P
} Number of joins = 2

seperate 2 join statements

A 2 B

A 2 C

$$\begin{tikzcd} & B \\ A \arrow[ur, "="] \arrow[dr, "="'] & \\ & C \end{tikzcd}$$

Therefore,	Number of JOINS = Number of tables - 1

Scenario 2

A		
P	Q	R

B			
P	Q	S	T

C		
P	X	Y

We want				
P	Q	R	S	X

$$\begin{tikzcd} & A.P = B.P & \\ A \arrow[ur, bend left] \arrow[r, bend right] \arrow[dr, bend right] & A.Q = B.Q & \text{extra} \arrow[l] \\ & A.P = C.P & \end{tikzcd}$$

The SQL query is ,

Select A.P, A.Q, A.R, B.S, C.X
From A, B, C
Where A.P = B.P
And A.Q = B.Q
And A.P = C.P ;

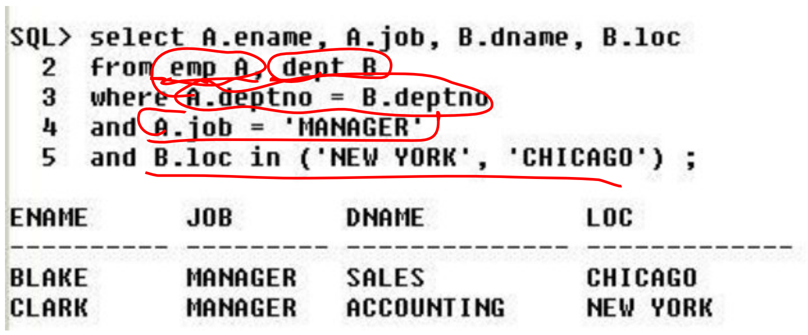
Number of Joins = 3

Therefore, **Number of JOINS = Number of common columns**

If there are no common columns, then reject it saying that the two tables can be joined.

But there are some cases – where the 2 columns will be same but having different column names. For ex – customerid & cid

Display employee name, his job, his dname and his location for all the managers living in New York or Chicago



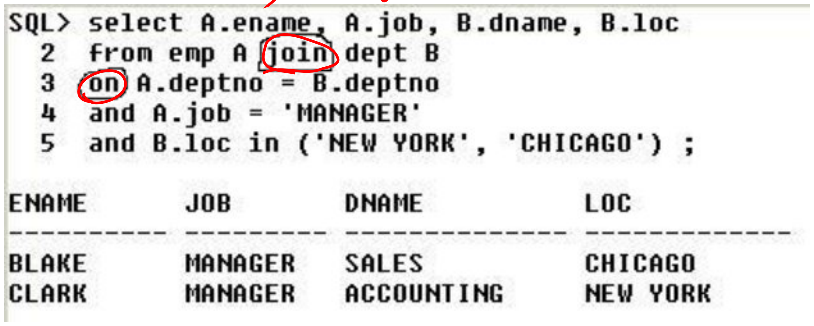
```
SQL> select A.ename, A.job, B.dname, B.loc
  2  from emp A, dept B
  3  where A.deptno = B.deptno
  4  and A.job = 'MANAGER'
  5  and B.loc in ('NEW YORK', 'CHICAGO') ;

ENAME      JOB       DNAME          LOC
---------- --------- -------------- -------------
BLAKE      MANAGER   SALES          CHICAGO
CLARK      MANAGER   ACCOUNTING     NEW YORK
```

ANSI style JOINS

This was introduced from Oracle 9i.
It is another way of writing inner joins with a few modifications.

join

where
$\downarrow$
on



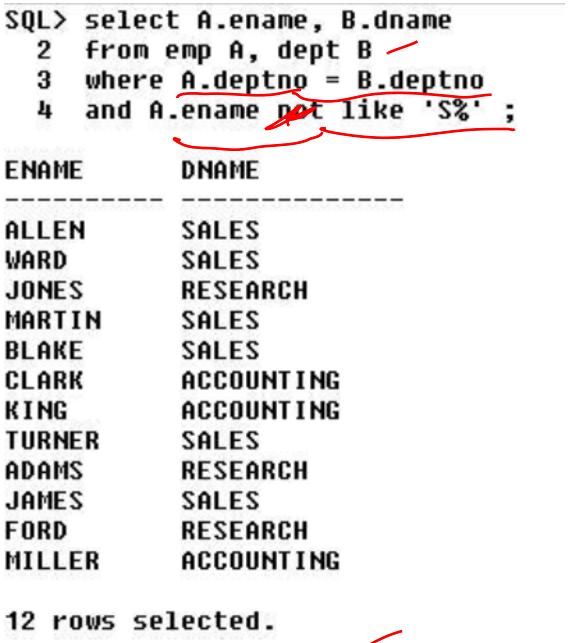
```
SQL> select A.ename, A.job, B.dname, B.loc
  2  from emp A join dept B
  3  on A.deptno = B.deptno
  4  and A.job = 'MANAGER'
  5  and B.loc in ('NEW YORK', 'CHICAGO') ;

ENAME      JOB       DNAME          LOC
---------- --------- -------------- -------------
BLAKE      MANAGER   SALES          CHICAGO
CLARK      MANAGER   ACCOUNTING     NEW YORK
```

Thus we, can see the changes ,

- In the 2nd line - ,(comma) has been replaced by the word „join"
- In the 3rd line – „where" has been replaced with „on"

Assignment

1) Display employee name and his **department name for the employees whose name starts with „S"**



```
SQL> select A.ename, B.dname
  2  from emp A, dept B
  3  where A.deptno = B.deptno
  4  and A.ename not like 'S%' ;

ENAME      DNAME
---------- --------------
ALLEN      SALES
WARD       SALES
JONES      RESEARCH
MARTIN     SALES
BLAKE      SALES
CLARK      ACCOUNTING
KING       ACCOUNTING
TURNER     SALES
ADAMS      RESEARCH
JAMES      SALES
FORD       RESEARCH
MILLER     ACCOUNTING

12 rows selected.
```

OUTER JOIN

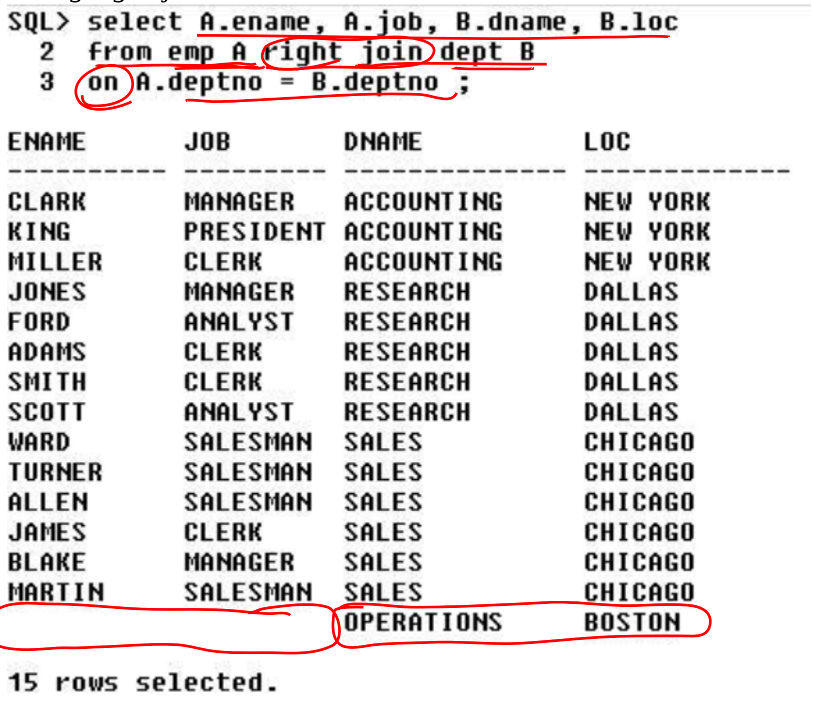
It returns both matching and non-matching records

Outer join = inner join + non-matching records

Non-matching records means data present in one table, but absent in another table w.r.to common columns.
For ex, 40 is there in deptno of dept table, but not there in deptno of emp table.

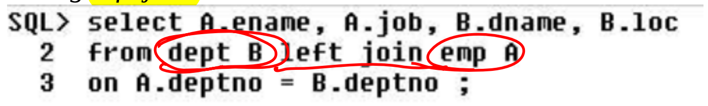
Display all the department names irrespective of any employee working in it or not. If an employee is working – display his name.

Using right join



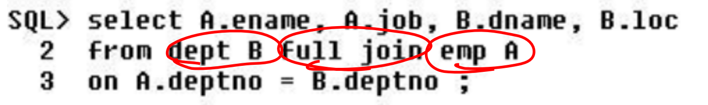
```
SQL> select A.ename, A.job, B.dname, B.loc
  2  from emp A right join dept B
  3  on A.deptno = B.deptno ;

ENAME      JOB       DNAME          LOC
---------- --------- -------------- -------------
CLARK      MANAGER   ACCOUNTING     NEW YORK
KING       PRESIDENT ACCOUNTING     NEW YORK
MILLER     CLERK     ACCOUNTING     NEW YORK
JONES      MANAGER   RESEARCH       DALLAS
FORD       ANALYST   RESEARCH       DALLAS
ADAMS      CLERK     RESEARCH       DALLAS
SMITH      CLERK     RESEARCH       DALLAS
SCOTT      ANALYST   RESEARCH       DALLAS
WARD       SALESMAN  SALES          CHICAGO
TURNER     SALESMAN  SALES          CHICAGO
ALLEN      SALESMAN  SALES          CHICAGO
JAMES      CLERK     SALES          CHICAGO
BLAKE      MANAGER   SALES          CHICAGO
MARTIN     SALESMAN  SALES          CHICAGO
                     OPERATIONS     BOSTON

15 rows selected.
```

Using left join

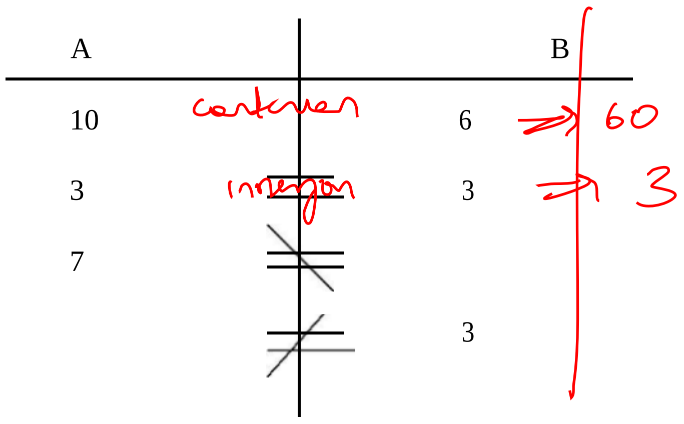


```
SQL> select A.ename, A.job, B.dname, B.loc
  2  from dept B left join emp A
  3  on A.deptno = B.deptno ;
```

Using full join



```
SQL> select A.ename, A.job, B.dname, B.loc
  2  from dept B full join emp A
  3  on A.deptno = B.deptno ;
```

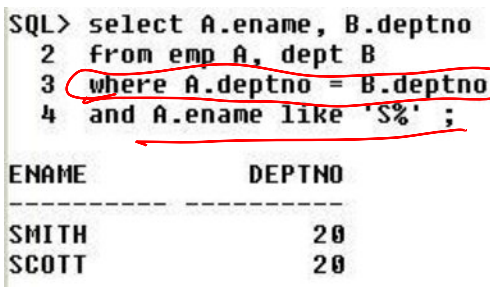


A CJ B = 60records A IJ B = 3records(3 matching)
A LJ B = 10records (3matching + 7non matching of A) ~~A~~
A RJ B = 6records (3matching + 3non matching of B)
A FJ B = 13records (3matching of A & B + 7nonmatching of A + 3nonmatching of B)

Assignment

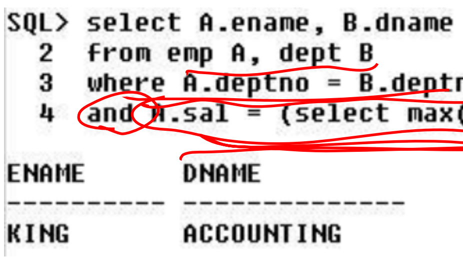
inner join

1) **Display employee name and his department name for the employees whose name starts with „S"**



```
SQL> select A.ename, B.deptno
  2  from emp A, dept B
  3  where A.deptno = B.deptno
  4  and A.ename like 'S%' ;

ENAME          DEPTNO
---------- ----------
SMITH              20
SCOTT              20
```

2) Display employee name and his department name who is earning 1st maximum salary



```
SQL> select A.ename, B.dname
  2  from emp A, dept B
  3  where A.deptno = B.deptno
  4  and A.sal = (select max(sal) from emp) ;

ENAME      DNAME
---------- --------------
KING       ACCOUNTING
```

SELF JOIN

Joining a table to itself is called self join

With a table

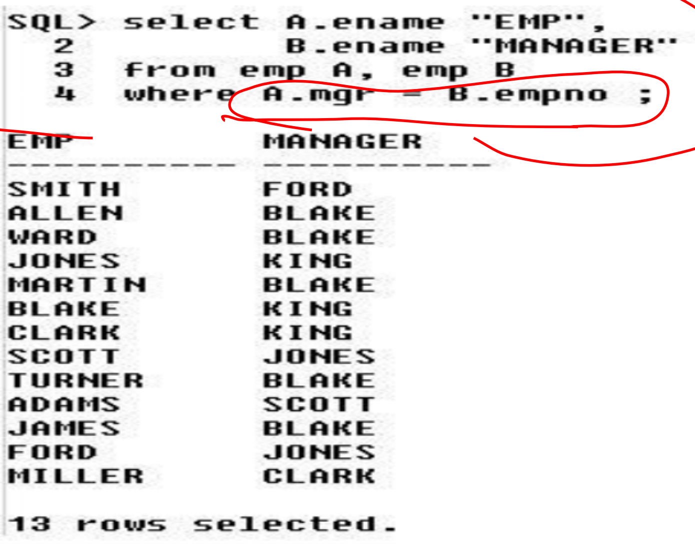
The FROM clause looks like this,

FROM emp A, emp B

Or

FROM emp A join emp B - *ANSI style*

For ex, - Display employee name along with their manager name

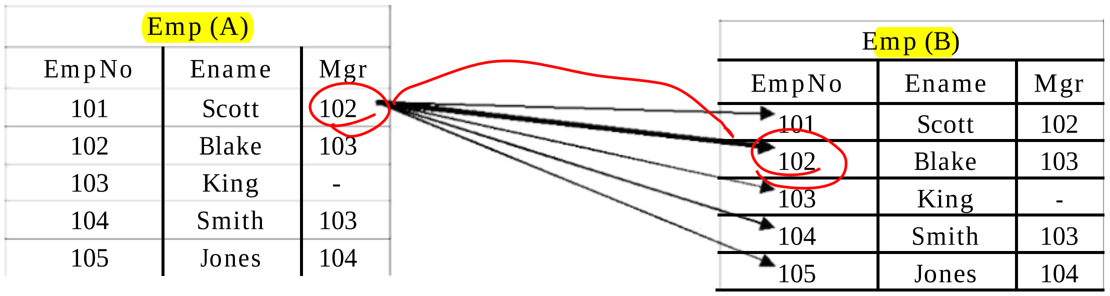


```
SQL> select A.ename "EMP",
  2         B.ename "MANAGER"
  3  from emp A, emp B
  4  where A.mgr = B.empno ;

EMP        MANAGER
---------- ----------
SMITH      FORD
ALLEN      BLAKE
WARD       BLAKE
JONES      KING
MARTIN     BLAKE
BLAKE      KING
CLARK      KING
SCOTT      JONES
TURNER     BLAKE
ADAMS      SCOTT
JAMES      BLAKE
FORD       JONES
MILLER     CLARK

13 rows selected.
```

is it Join like innerjoin?

X bcz

Now, let us see how this i.e the logic (the above query) works,



Emp (A)		
EmpNo	Ename	Mgr
101	Scott	102
102	Blake	103
103	King	-
104	Smith	103
105	Jones	104

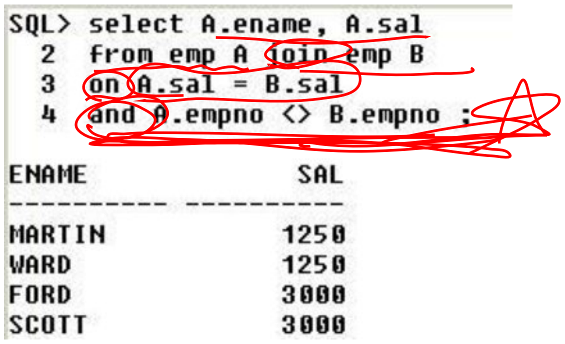
Emp (B)		
EmpNo	Ename	Mgr
101	Scott	102
102	Blake	103
103	King	-
104	Smith	103
105	Jones	104

Now, when we give the above query – in Oracle – it starts matching the „mgr" column of emp A with the „empno" of emp b – we get two tables because in self join – a duplicate of the table required is created.

Now let us consider the first employee Scott – it starts the mgrid of Scott with the empno of all the records in emp B – when two ids match, then the empno in emp B becomes the mgr of the empno in emp A. Thus, we can see that – mgr id 102 is matching with empno 102 Blake in emp B. Therefore, Blake is the manager of Scott.

Similarly we do the same for all the other records of emp A and thus find the employees and their respective managers.

<u>Display the employees who are getting the same salary</u>

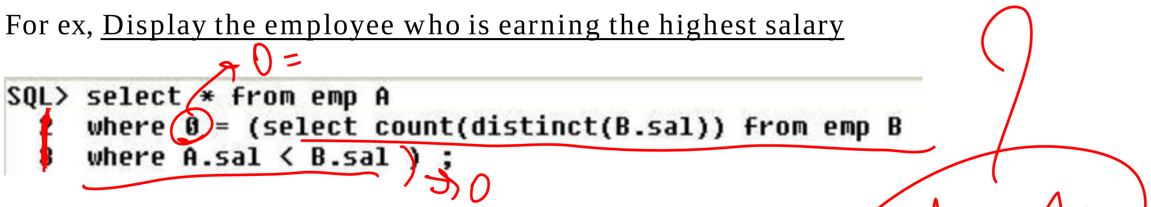


```
SQL> select A.ename, A.sal
  2  from emp A join emp B
  3  on A.sal = B.sal
  4  and A.empno <> B.empno ;

ENAME             SAL
---------- ----------
MARTIN           1250
WARD             1250
FORD             3000
SCOTT            3000
```

<u>Co – related Queries</u> :

→ They are special type of sub – queries
→ Here, both outer & inner queries are inter-dependent
→ For each & every record of outer query, the entire inner query will be executed
→ They work on the principles of both sub – queries & JOIN(s).

For ex, <u>Display the employee who is earning the highest salary</u>



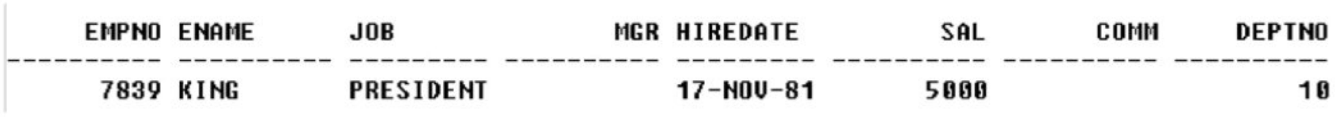
0 =

```
SQL> select * from emp A
  2  where 0 = (select count(distinct(B.sal)) from emp B
  3  where A.sal < B.sal ) ;
```

→ 0

?

doubt



```
     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
```

Thus, if an outer query column is being accessed inside the inner query, then that query is said to be co-related.

Let us see the logic i.e, how we get the 1st max salary :-

Emp (A)		
EmpNo	Ename	Sal
101	Scott	3000
102	Blake	4000
103	King	5000
104	Smith	2000
105	Jones	1000

Emp (B)		
EmpNo	Ename	Sal
101	Scott	3000
102	Blake	4000
103	King	5000
104	Smith	2000
105	Jones	1000

Since co-related queries are a combination of Joins and sub-queries.
It follows the concept of Joins and creates multiple copies of the same table.
Then it takes 1st record i.e, - Blake – sal is 3000. It starts comparing with the sal in the emp table, 3000 = 3000 - count starts from 0 – thus, 0 = 0
3000 < 4000 – thus, 0 ! = 1
3000 < 5000 – thus, 0 ! = 2
3000 > 2000 – thus , 0! = 2
3000 > 1000 – thus, 0 ! = 2 if the condition becomes false, then the count increments by 1. Here 3000 is less than 4000 & 5000, thus 0 ! = 2. Thus , Blake does not have the highest salary.

Similarly, it does for the next records,
Blake – salary of 4000 – but 4000 < 5000 – thus, 0 ! = 1. This is also false.
King – salary of 5000 – it is greater than everything – thus, 0 = 0. Thus, King has the highest salary.
But the query doesn"t stop here, it checks for Smith & Jones as well.

Similarly, if we want to find the 2nd maximum salary,
Then in the query, change „0" to „1" & here, the logic is – it compares until it gets 1 = 1.
For 3rd maximum salary – change 0 to 2 and so on – here, the logic is – it compares until it gets 2 = 2.

For any highest, always put it as „0" in the query.

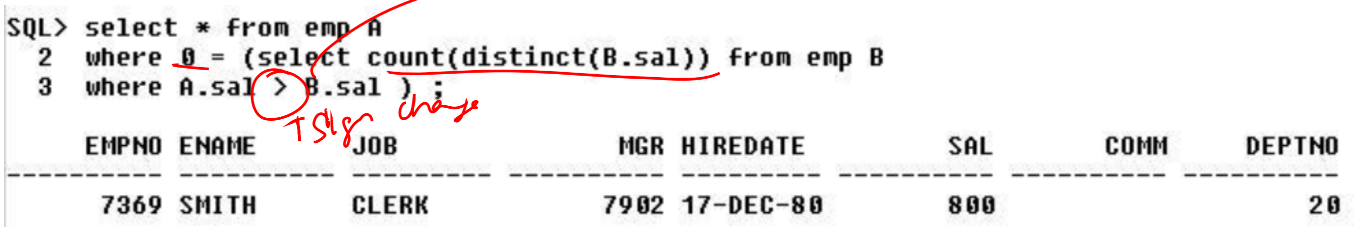
If you want n(th) salary, pass (n-1).

In interview – this is a definite question. They will ask you what is co-related queries. And then **they"ll ask you find, 1st** or max or 3rd maximum salary **–** after you write the query **–** they will ask you to explain the logic as to how it gets the same **–** draw the table and explain it to them just as shown above.

Assignment

1) Display the least salary from the employee table.

min

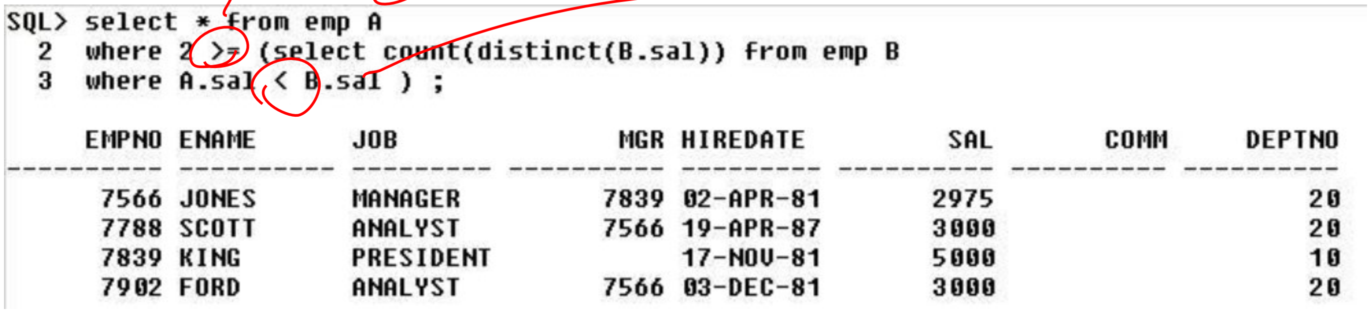


```
SQL> select * from emp A
  2  where 0 = (select count(distinct(B.sal)) from emp B
  3  where A.sal > B.sal ) ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
```

T sign change

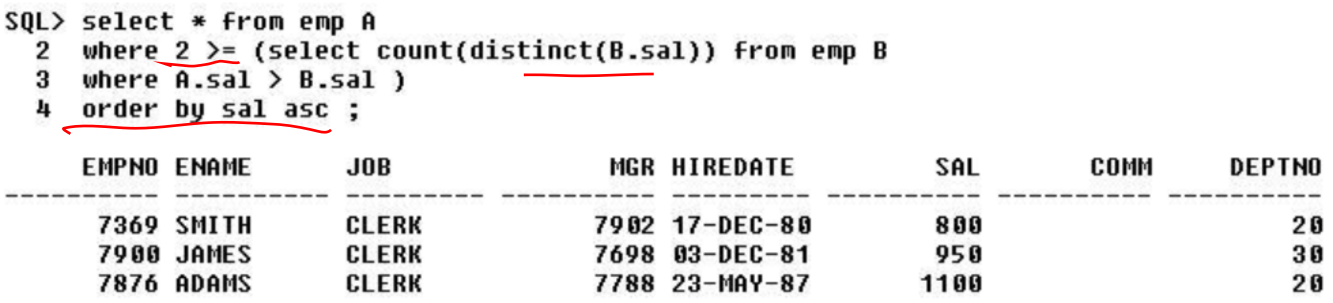
2) Display top 3 person‟s salaries from the employee table.

= exact >= max



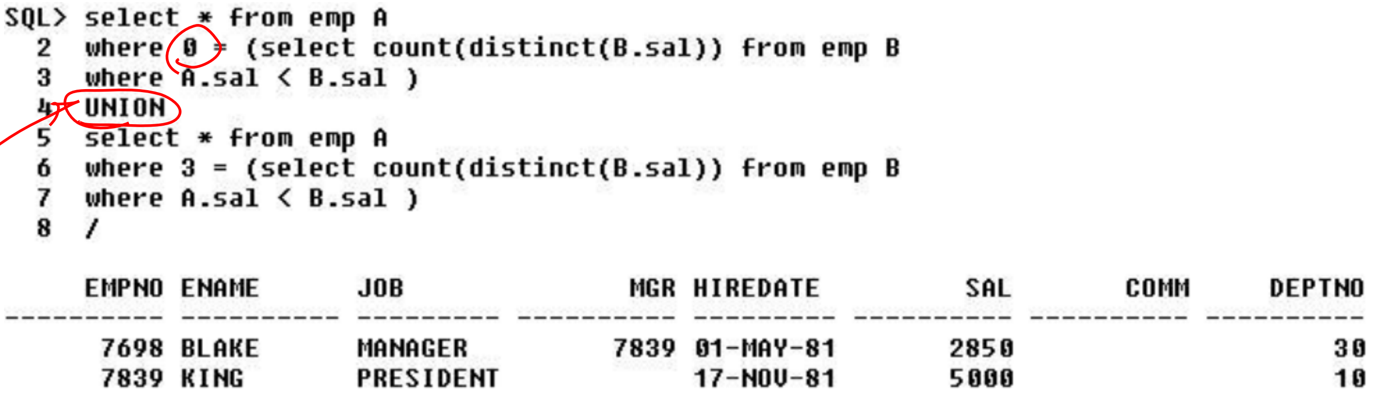
```
SQL> select * from emp A
  2  where 2 >= (select count(distinct(B.sal)) from emp B
  3  where A.sal < B.sal ) ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
```

3) Write a query to display bottom 3 salaries



```
SQL> select * from emp A
  2  where 2 >= (select count(distinct(B.sal)) from emp B
  3  where A.sal > B.sal )
  4  order by sal asc ;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20
```

4) Display 1st and 4th maximum salary

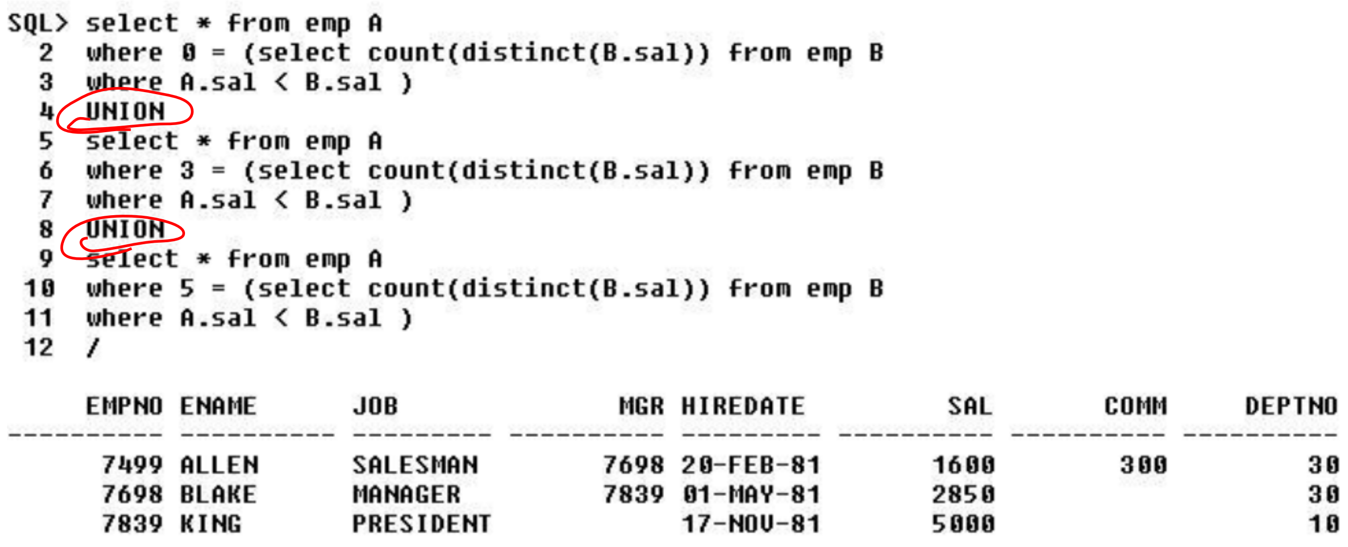


```
SQL> select * from emp A
  2  where 0 = (select count(distinct(B.sal)) from emp B
  3  where A.sal < B.sal )
  4  UNION
  5  select * from emp A
  6  where 3 = (select count(distinct(B.sal)) from emp B
  7  where A.sal < B.sal )
  8  /

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
```

2

join diff groups of diff table

5) Display 1st, 4th & 6th highest salaries in a single query



```
SQL> select * from emp A
  2  where 0 = (select count(distinct(B.sal)) from emp B
  3  where A.sal < B.sal )
  4  UNION
  5  select * from emp A
  6  where 3 = (select count(distinct(B.sal)) from emp B
  7  where A.sal < B.sal )
  8  UNION
  9  select * from emp A
 10  where 5 = (select count(distinct(B.sal)) from emp B
 11  where A.sal < B.sal )
 12  /

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
```