SPECIAL OPERATORS :

1. IN
2. NOT IN
3. BETWEEN
4. NOT BETWEEN
5. IS
6. IS NOT
7. LIKE
8. NOT LIKE

1. **IN :** *It is a multi-valued operator which can accept multiple values At the RHS* **.**

Syntax: Column_Name / Exp **IN** (v1 , v2 , . . Vn)

Example :

- WAQTD name and deptno of the employees working in dept 10 or 30 .

```
SELECT ENAME , DEPTNO
FROM EMP
WHERE DEPTNO = 10 OR DEPTNO = 30 ;
```

```
SELECT ENAME , DEPTNO
FROM EMP
WHERE DEPTNO IN ( 10 , 30 ) ;
```

- WAQTD name and job of the employee working as a clerk or manager Or salesman .

```
SELECT ENAME , JOB
FROM EMP
WHERE JOB IN ('CLERK' , 'MANAGER' ,
'SALESMAN' ) ;
```

2. **NOT IN :** *It is a multi-valued operator which can accept multiple values At the RHS . It is similar to IN op instead of selecting it Rejects the values .*

Syntax: Column_Name / Exp **NOT IN** (v1 , v2 , . . vn)

Example :

- WAQTD name and deptno of all the employees except the emp Working in dept 10 or 40 .

```
SELECT ENAME , DEPTNO
FROM EMP
WHERE DEPTNO NOT IN ( 10 , 40 ) ;
```

- WAQTD name , deptno and job of the employee working in dept 20 but not as a clerk or manager .


```
SELECT ENAME , DEPTNO
FROM EMP
WHERE DEPTNO = 20 AND
JOB NOT IN ( 'CLERK' ,'MANAGER' ) ;
```

ANSWERS :

1.WAQTD DETAILS OF THE EMPLOYEES WORKING AS CLERK AND EARNING LESS THAN1500

```
SELECT *
FROM EMP
WHERE JOB ='CLERK' AND SAL< 1500 ;
```

2.WAQTD NAME AND HIREDATE OF THE EMPLOYEES WORKING AS MANAGER IN DEPT 30

```
SELECT ENAME , HIREDATE
FROM EMP
WHERE JOB ='MANAGER' AND DEPTNO=30 ;
```

3. **BETWEEN :** *"It is used whenever we have range of values "*
[Start value and Stop Value] .

Syntax:

```
Column_Name BETWEEN Lower_Range AND Higher_Range ;
```

- *Between Op works including the range .*

Example :

- WAQTD name and salary of the employees if the emp is earning Salary in the range 1000 to 3000 .

```
SELECT ENAME , SAL
FROM EMP
WHERE SAL BETWEEN 1000 AND 3000 ;
```

- WAQTD name and deptno of the employees working in dept 10 And hired during 2019 (the entire year of 2019) .

```
SELECT ENAME , DEPTNO
FROM EMP
WHERE DEPTNO = 10 AND HIREDATE BETWEEN '01-
JAN-2019' AND '31-DEC-2019' ;
```

- WAQTD name , sal and hiredate of the employees hired during 2017 into dept 20 with a salary greater that 2000 .

```
SELECT ENAME , SAL , HIREDATE
FROM EMP
WHERE DEPTNO = 20 AND SAL> 2000 AND HIREDATE
BETWEEN '01-JAN2017' AND 31-DEC-2017' ;
```

4. **NOT BETWEEN :** It is Opposite of Between .

Syntax:

```
Column_Name NOT BETWEEN Lower_Range AND Higher_Range ;
```

4. **NOT BETWEEN :** It is Opposite of Between .

Syntax:

```
Column_Name NOT BETWEEN Lower_Range AND Higher_Range ;
```

Example :

- WAQTD name and salary of the employees if the emp is not earning Salary in the range 1000 to 3000 .

```
SELECT ENAME , SAL
FROM EMP
WHERE SAL NOT BETWEEN 1000 AND 3000 ;
```


- WAQTD name and deptno of the employees working in dept 10 And not hired during 2019 .

```
SELECT ENAME , DEPTNO
FROM EMP
WHERE DEPTNO = 10 AND HIREDATE NOT BETWEEN '01-
JAN-2019' AND '31-DEC-2019' ;
```

- WAQTD name , sal and hiredate of the employees who were not hired during 2017 into dept 20 with a salary greater that 2000 .

```
SELECT ENAME , SAL , HIREDATE
FROM EMP
WHERE DEPTNO = 20 AND SAL> 2000 AND HIREDATE NOT
BETWEEN '01-JAN2017' AND 31-DEC-2017' ;
```

5. **IS :** *"It is used to compare only NULL "*

```
Syntax:  Column_Name IS NULL ;
```

5. **IS :** *"It is used to compare only NULL "*

Syntax: Column_Name **IS** NULL ;

Example :

EID	ENAME	SAL	COMM
1	A	1000	100
2	B	null	null
3	C	null	200
4	D	2000	null

- WAQTD name of the employee who is not getting salary .

```
SELECT ENAME
FROM EMP
WHERE SAL IS NULL ;
```

- WAQTD name of the emp who doesn't get commission .

```
SELECT ENAME
FROM EMP
WHERE COMM IS NULL ;
```

- WAQTD name , sal and comm of the emp if the emp doesn't ear both .

```
SELECT ENAME , SAL , COMM
FROM EMP
WHERE COMM IS NULL AND SAL IS NULL ;
```

6. **IS NOT :** *"It is used to compare the values with NOT NULL ".*

Syntax: Column_Name **IS NOT** NULL ;

6. **IS NOT :** *"It is used to compare the values with NOT NULL ".*

Syntax: Column_Name **IS NOT** NULL ;

Example :

- WAQTD name of the employee who is getting salary .

```
SELECT ENAME
FROM EMP
WHERE SAL IS NOT NULL ;
```

- WAQTD name of the emp who gets commission .

```
SELECT ENAME
FROM EMP
WHERE COMM IS NOT NULL ;
```

- WAQTD name , sal and comm of the emp if the emp doesn't ear commission but gets salary .

```
SELECT ENAME , SAL , COMM
FROM EMP
WHERE COMM IS NULL AND SAL IS NOT NULL ;
```

7. **LIKE :** *"It is used for Pattern Matching ".*

To achieve pattern matching we use special characters .

- Percentile (%)
- Underscore (_)

Syntax: Column_Name LIKE 'pattern' ;

Example :

- WAQTD details of an employee whose name is SMITH .

```
SELECT *
FROM EMP
WHERE ENAME ='SMITH' ;
```

- WAQTD details of the employee who's name starts with 'S' .

```
SELECT *
FROM EMP
WHERE ENAME LIKE 'S%' ;
```

- WAQTD details of the employee who's name ends with 'S' .

```
SELECT *
FROM EMP
WHERE ENAME LIKE '%S' ;
```

- WAQTD names of the employees who have character 'S' in their names .

```
SELECT *
FROM EMP
```



```
WHERE ENAME LIKE '%S%' ;
```

- WAQTD names that starts with 'J' and ends with 'S' .

```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE 'J%S' ;
```

- WAQTD names of the employee if the emp has char 'A' as his second character .

```
SELECT ENAME
FROM EMP
```

- WAQTD names of the employee if the emp has char 'A' as his second character .

```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE '_A%' ;
```

- WAQTD names of the employee if the emp has char 'A' as his character .

```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE '__A%' ;
```

- WAQTD names of the employee if the emp has char 'A' as his second character and 'S' is last character .

```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE '_A%S' ;
```

- WAQTD names of the employee if the emp has char 'A' presen least 2 times .

```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE '%A%A%' ;
```

- WAQTD names of the employee if the emp name starts with 'A ends with 'A' .

```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE 'A%A' ;
```

- WAQTD names of the employee if the emp's salary's last 2 dig 50 rupees .

```
SELECT ENAME
FROM EMP
WHERE SAL LIKE '%50' ;
```

- WAQTD names of the employee if the emp has char 'A' as his second character and 'S' is last character .

```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE '_A%S' ;
```

- WAQTD names of the employee if the emp has char 'A' present at least 2 times .

```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE '%A%A%' ;
```

- WAQTD names of the employee if the emp name starts with 'A' an ends with 'A' .

```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE 'A%A' ;
```

- WAQTD names of the employee if the emp's salary's last 2 digit is 50 rupees .

```
SELECT ENAME
FROM EMP
WHERE SAL LIKE '%50' ;
```

- WAQTD names of the employees hired in November .

```
SELECT ENAME
FROM EMP
```



```
WHERE HIREDATE LIKE '%NOV%' ;
```

8. **NOT LIKE :**Opposite of Like .

Syntax: Column_Name **NOT LIKE** 'pattern' ;

FUNCTIONS

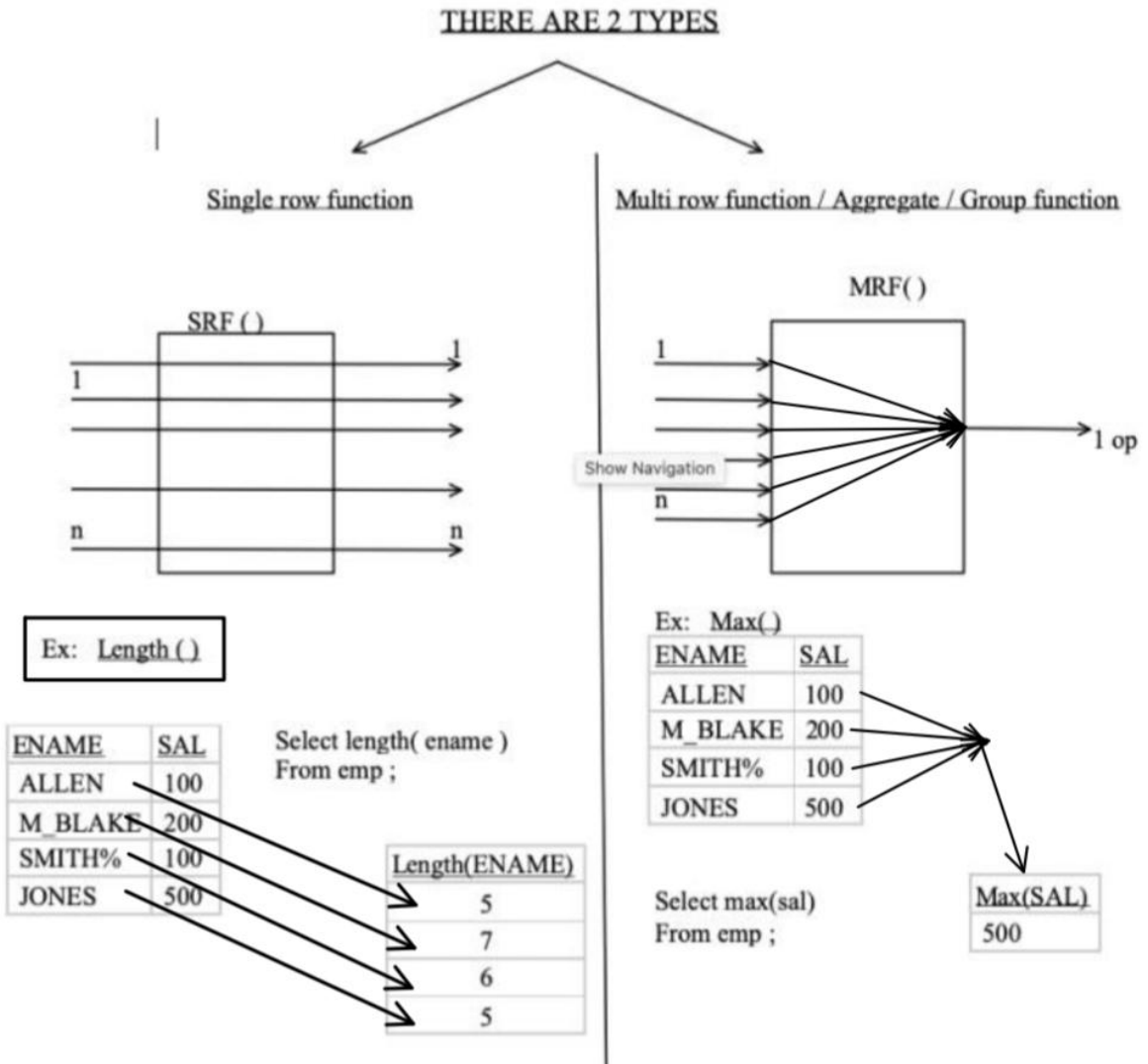
Are a block of code or list of instructions which are used to perform a specific task .

There are 3 main components of a function

1. Function_Name
2. Number_of_arguments (no of inputs)
3. Return type

Types of Functions in SQL :

1. *SINGLE ROW FUNCTIONS*
2. *MUTLI ROW FUNCTIONS* / AGGREGATE / GROUP FUNCTIONS.



Multi Row Functions;

It takes all the inputs at one shot and then executes and provides A single output .

- If we pass 'n' number of inputs to a MRF( ) it returns '1' Output .

List of MRF ( )

1. MAX( ) : it is used to obtain the maximum value present in the column
2. MIN ( ) : it is used to obtain the minimum value present in the

column
3. SUM ( ) : it is used to obtain the summation of values present in the column
4. AVG( ) : it is used to obtain the average of values present in the column
5. COUNT( ) : it is used to obtain the number of values present in the column

NOTE :

- Multi row functions can accept only one argument , i.e a Column_Name or an Expression

MRF (Column_Name / Exp)

- Along with a MRF( ) we are not supposed to use any other Column_Name in the select clause .
- MRF( ) ignore the Null .
- We cannot use a MRF( ) in where clause .
- COUNT( ) is the only MRF which can accept * as an Argument .

GROUP & FILTERING

GROUPING : GROUP BY Clause

Group by clause is used to *group the records* .

SYNTAX:

```
SELECT  group_by_expression / group_function
FROM table_name
[WHERE <filter_condition>]
GROUP BY column_name/expression ;
```

ORDER OF EXECUTION:

1-FROM
2-WHERE(if used) [ROW-BY-ROW]
3-GROUP BY [ROW-BY-ROW]
4-SELECT [GROUP-BY-GROUP]

EMP

EID	ENAME	SAL	DEPTNO
1	A	100	20
2	B	200	10
3	C	300	30
4	D	100	10
5	E	200	10
6	A	400	30
7	C	500	20
8	F	200	30

Example :

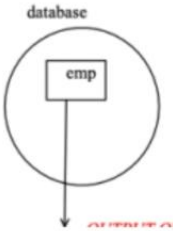
- WAQTD number of employees working in each dept .

```
SELECT  COUNT(*)
FROM EMP
GROUP BY DEPTNO ;
```

Example :

- WAQTD number of employees working in each dept .

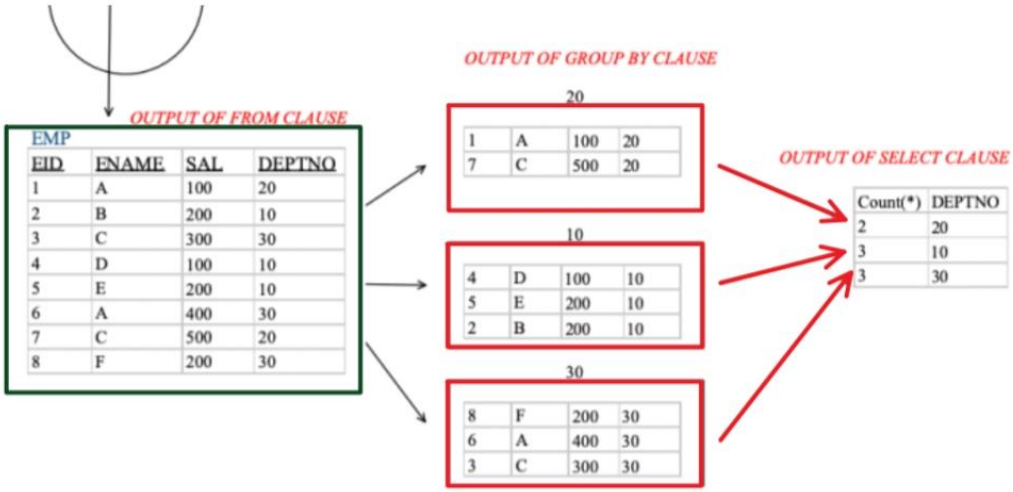
```
SELECT  COUNT(*)
FROM EMP
GROUP BY DEPTNO ;
```




```
Select count(*) , DEPTNO
From emp
Group by deptno;
```

OUTPUT OF GROUP BY CLAUSE

20



NOTE :

- Group By clause is used to group the records .
- Group By clause executes row by row .
- After the execution of Group By clause we get Groups .
- Therefore any clause that executes after group by must execute Group By Group .
- The Column_Name or expression used for grouping can be used In select clause .
- Group By clause can be used without using Where clause .

- Therefore any clause that executes after group by must execute Group By Group .
- The Column_Name or expression used for grouping can be used In select clause .
- Group By clause can be used without using Where clause .

Questions :

1. WAQTD number of employees working in each dept except the Employee working as analyst .

```
SELECT DEPTNO , COUNT(*)
FROM EMP
WHERE JOB NOT IN 'ANALYST'
GROUP BY DEPTNO ;
```

2. WAQTD maximum salary given to each job .

```
SELECT JOB , MAX( SAL )
FROM EMP
GROUP BY JOB ;
```

3. WAQTD number of employees working in each job if the employees Have character 'A' in their names .

```
SELECT JOB , COUNT(*)
FROM EMP
WHERE ENAME LIKE '%A%'
GROUP BY JOB ;
```

4. WAQTD number of employees getting commission in each dept .

```
SELECT DEPTNO , COUNT( COMM )
```



```
FROM EMP
GROUP BY DEPTNO ;
```

FILTERING : HAVING Clause

" Having Clause is used to Filter the Group "

SYNTAX:

```
SELECT  group_by_expression / group_function
FROM table_name
[WHERE <filter_condition>]
GROUP BY column_name/expression
HAVING  <group_filter_condition>
```

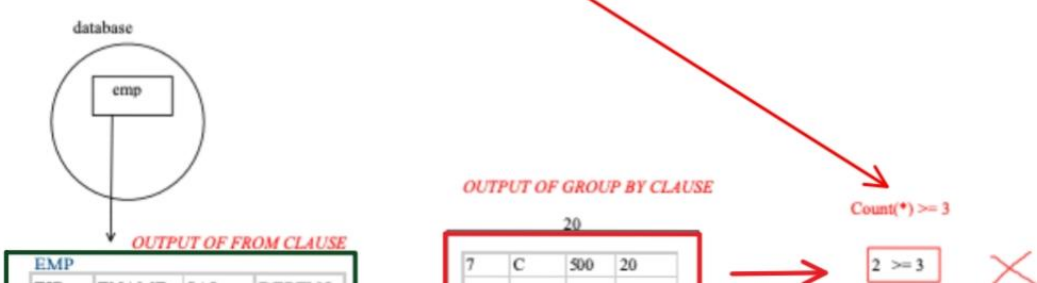
ORDER OF EXECUTION:

1-FROM
2-WHERE(if used) *[ROW-BY-ROW]*
3-GROUP BY(if used) *[ROW-BY-ROW]*
4-HAVING (if used) *[GROUP-BY-GROUP]*
5-SELECT *[GROUP-BY-GROUP]*

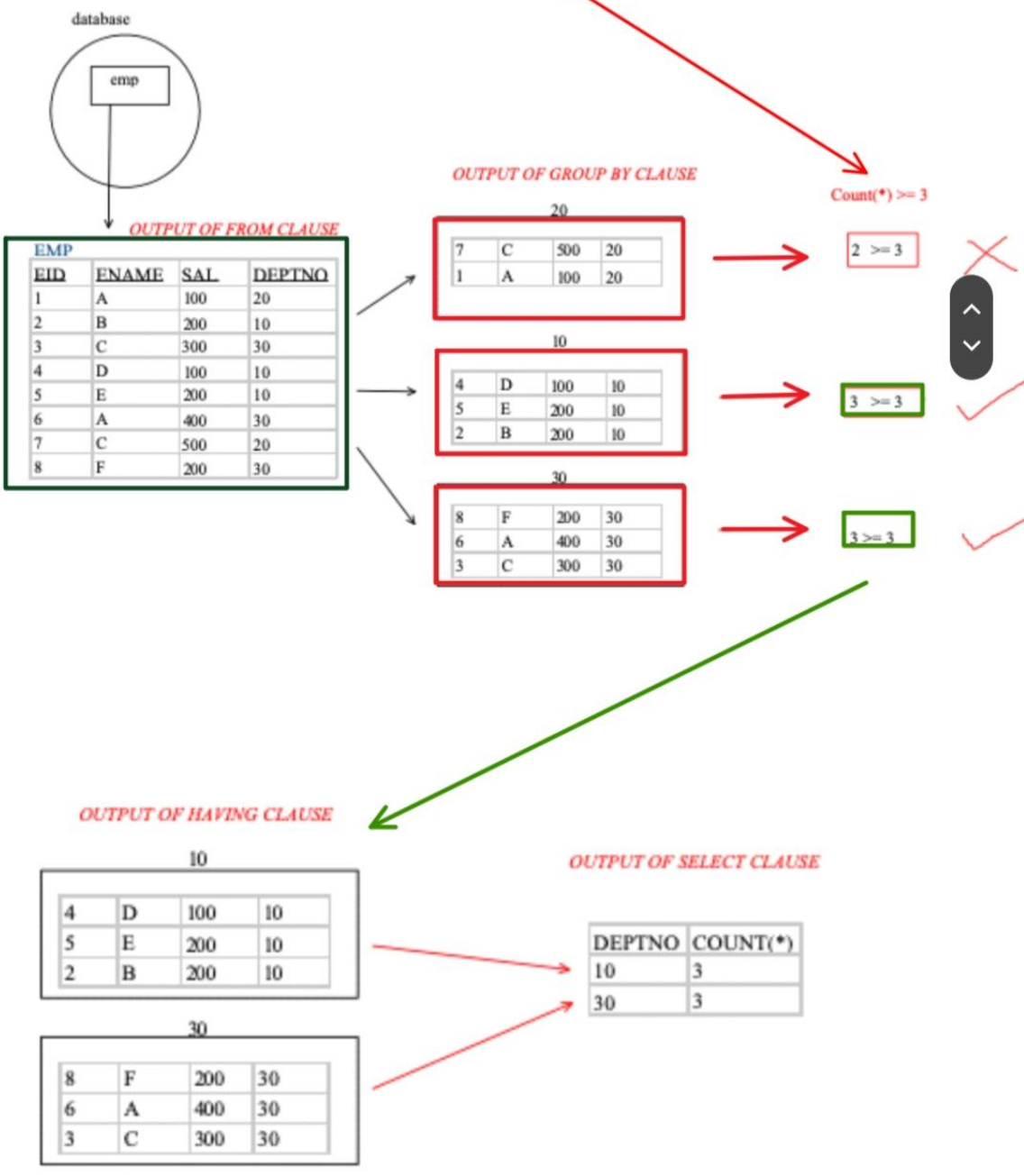
Example :

- WAQTD to find number of employees working in each Dept if there are at least 3 employees in each dept .


```
SELECT DEPTNO , COUNT(*)
FROM EMP
GROUP BY DEPTNO
HAVING COUNT(*)>=3 ;
```



```
SELECT DEPTNO , COUNT(*)
FROM EMP
GROUP BY DEPTNO
HAVING COUNT(*)>=3 ;
```



Questions :

1. WAQTD the designations in which there are at lest 2 employe
   Present .

30

8	F	200	30
6	A	400	30
3	C	300	30

Questions :

1. WAQTD the designations in which there are at lest 2 employees Present .

```
SELECT JOB , COUNT(*)
FROM EMP
GROUP BY JOB
HAVING COUNT(*)>=2 ;
```

2. WAQTD the names that are repeated .

```
SELECT ENAME , COUNT(*)
FROM EMP
GROUP BY ENAME
HAVING COUNT(*) > 1 ;
```

3. WAQTD names that are repeated exactly twice .


```
SELECT ENAME , COUNT(*)
FROM EMP
GROUP BY ENAME
HAVING COUNT(*)  = 2  ;
```

4. WAQTD the salary that is repeated .

```
SELECT SAL, COUNT(*)
FROM EMP
GROUP BY SAL
HAVING COUNT(*) > 1 ;
```

NOTE :

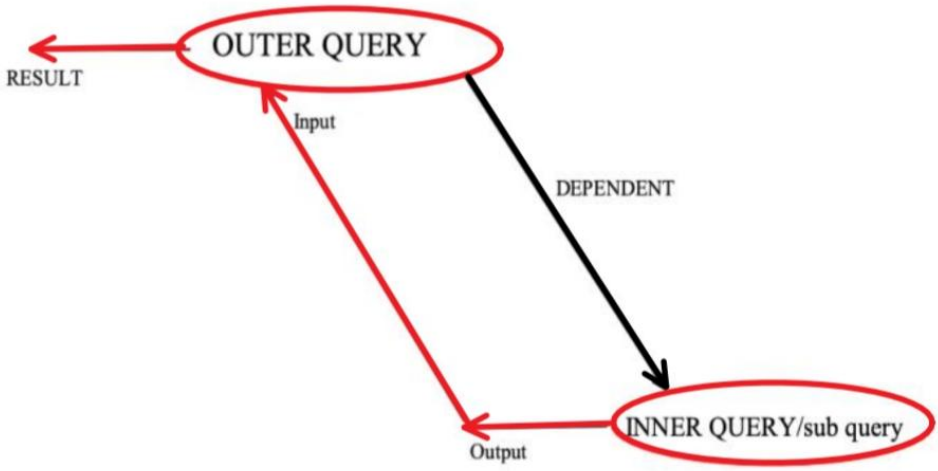
Differentiate between Where and Having .

WHERE	HAVING
➢ Where clause is used to Filter the records	➢ Having clause is used to Filter the groups .
➢ Where clause executes row By row .	➢ Having clause executes Group by group
➢ In Where Clause we cannot Use MRF( )	➢ Can use MRF( ).
➢ Where clause executes before Group by clause .	➢ Having clause executes After group by clause .

SUB-QUERY

" A QUERY WRITTEN INSIDE ANOTHER QUERY IS KNOWN AS SUB QUERY "

Working Principle :



Let us consider two queries Outer Query and Inner Query .

- Inner Query executes first and produces an Output .
- The Output of Inner Query is given / fed as an Input to Outer Query .
- The Outer Query generates the Result.
- Therefore we can state that 'the Outer Query is dependent on Inner Query' and this is the Execution Principle of Sub Query .

Why / When Do we use SUB QUERY :

Case 1 : Whenever we have Unknowns present in the Question We use sub query to find the Unknown .

Example :

EMP

EID	ENAME	SAL	DEPTNO
1	ALLEN	1000	20
2	BLAKE	2000	10
3	CLARK	3000	30
4	MILLER	1500	10
5	SMITH	2500	10

1. WAQTD names of the employees earning more than 2500 .

```
SELECT ENAME
FROM EMP
```

CASE-2 : Whenever the data to be selected and the condition to be executed are present in different tables we use Sub Query .

Example :

Emp

EID	ENAME	SAL	DEPTNO
1	ALLEN	1000	20
2	BLAKE	2000	10
3	CLARK	3000	30
4	MILLER	1500	10
5	ADAMS	2500	20

DEPT

DEPTNO	DNAME	LOC
10	D1	L1
20	D2	L2
30	D3	L3

1. WAQTD deptno of the employee whose name is Miller .

```
SELECT DEPTNO
FROM EMP
WHERE ENAME ='MILLER' ;
```

2. WAQTD **dname** of the employee whose name is **Miller** .

```
SELECT DNAME
FROM DEPT
WHERE DEPTNO = ( SELECT DEPTNO
                 FROM EMP
                 WHERE ENAME ='MILLER' ) ;
```

3. WAQTD Location of ADAMS

```
SELECT LOC
FROM DEPT
WHERE DEPTNO = ( SELECT DEPTNO
FROM EMP
WHERE ENAME ='ADAMS' ) ;
```

4. WAQTD names of the employees working in Location L2.

```
SELECT ENAME
FROM EMP
WHERE DEPTNO = ( SELECT DEPTNO
FROM DEPT
WHERE LOC ='L2' ) ;
```

5. WAQTD number of employees working in dept D3 .

```
SELECT COUNT(*)
FROM EMP
```