HISTORY OF SQL

SEQUEL - SIMPLE ENGLISH QUERY LANGUAGE

RAYMOND BOYCE →1970

(ANSI) – AMERICAN NATIONAL STANDARD INSTITUTE

RENAMED – SQL – 1986

NOW THE OWNER OF SQL IS **ORACLE**

SQL

(STRUCTURED QUERY LANGUAGE)

KANNADA – KARNATAKA

MARATHI – MAHARASHTRA

TELUGU – ANDHRA PRADESH

$$\begin{tikzcd} \text{SQL} \arrow[d] \\ \text{DATABASE} \arrow[r] & \text{DATA} \end{tikzcd}$$

DATA

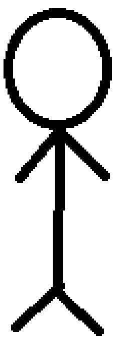
"Data is rawfact which describes attributes of an entity."

RAWFACT – unchanged

ATTRIBUTE – properties

ENTITY – object

for ex :



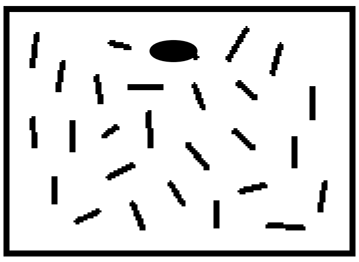
student
OBJECT / ENTITY

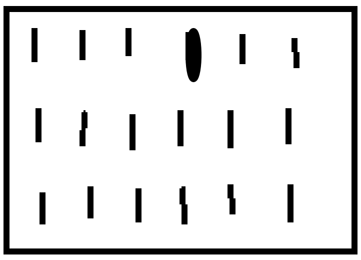
PROPERTIES

SID -	1
SNAME -	TINKU
CLASS-	10 'B'
GENDER -	MALE
PH _NO -	1234567890

DATABASE

"It is place /medium which is used to store the data in Systematic and organized manner."


parking 1


parking 2

Difficult to access

Easy to access

- **The basic operation performed on database is known as CRUD Operation**

  C – Create/Insert

  R – Read / Retrieve

  U – Update /Modify

  D – Delete/Drop

DBMS

(DATABASE MANAGEMENT SYSTEM)

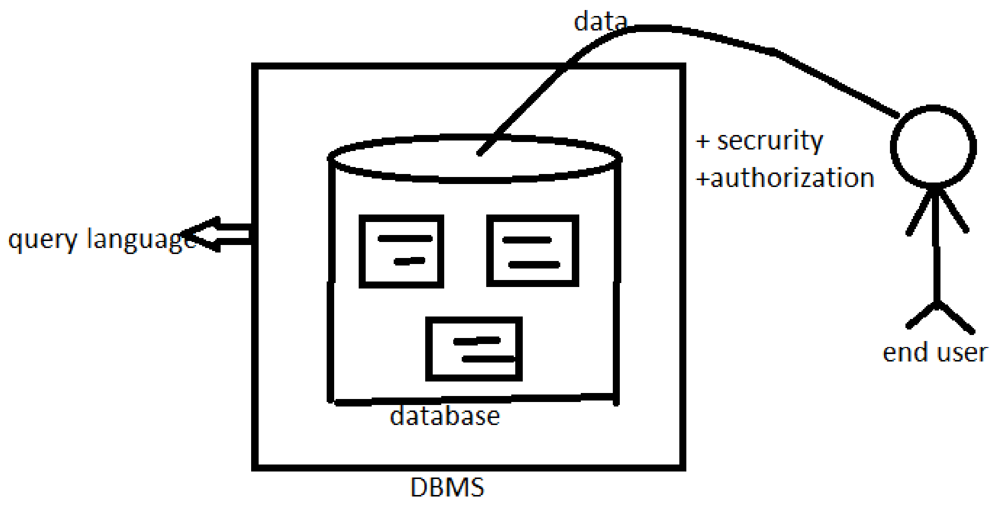
DBMS is software which used to maintain and manage the database.

DBMS provides two imp features :-

- ❖ Security – username , password
- ❖ Authorization – only one person can access – OTP

DBMS stores the data in file format.

We use query language to communicate with DBMS.



TYPES OF DBMS :-

- NETWORK DBMS
- **RDBMS**
- HIERARCHICAL DBMS
- OOPS

RDBMS

(RELATIONAL DATABASE MANAGEMENT SYSTEM)

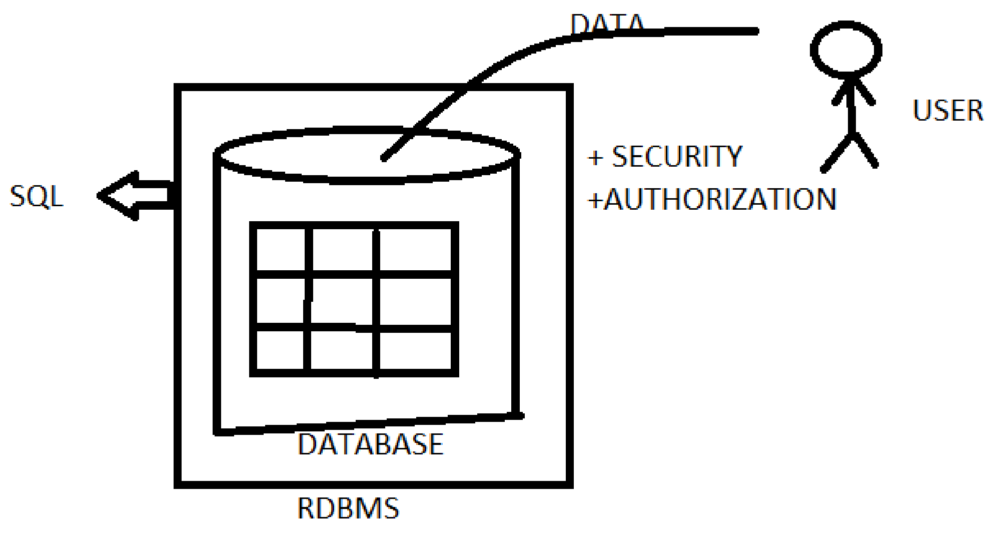
"It is a software which is used to maintain and manage the database."

RDBMS provides two imp features :-

- Security
- Authorization

RDBMS store the data in table format.

We use structured query language to communicate with RDBMS



DIFFERENCE BETWEEN DBMS AND RDBMS

DBMS	**RDBMS**
1. DBMS store the data in file format	**1. RDBMS store the data in table format**
2. We use Query language to communicate with DBMS	**2. We use structured Query language to communicate with RDBMS**
3. DBMS provide two imp features ○ **Security** ○ **Authorization**	**3. RDBMS provide two imp features** ○ **Security** ○ **Authorization**
4. Compare to RDBMS performing CRUD operation is difficult in DBMS	**4. Performing CRUD Operation is easy compare to DBMS**

RELATIONAL MODEL AND TABLE

"It is a model which is used store the data in the form of table."

- This is a model which is invented by data scientist called EDGER FRANK CODD (E.F CODD)
- Any DBMS follows Relation model it will become RDBMS.

$$\begin{tikzcd}
\text{DBMS} \arrow[r] \arrow[dr] & \text{RELATIONAL MODEL} \arrow[r] & \text{RDBMS} \\
& \text{Rules of E.F CODD} \arrow[ur] &
\end{tikzcd}$$

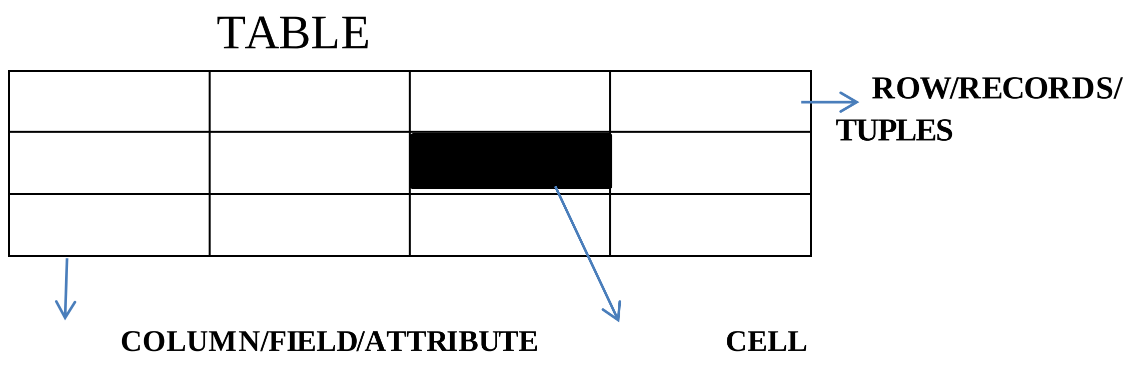
- If any DBMS following rules of E.F CODD it will become RDBMS.

TABLE

"The logical arrangement of rows and column is called table."

OR

The combination of rows and columns



CELL :- The smallest unit of a table is called cell

OR

The intersection of rows and columns are known as cells.

RULES OF E.F CODD

RULE 1. Data entered into a cell must be single valued data.

SID	SNAME	PH_NO	Alt_PH_NO
1	TINKU	9807890876	
2	DINGI	9876745476	9878987678
3	DINGA	7898789879	
4	TINKI	9878987678	9878987678

RULE 2. In RDBMS we store everything in the form table including metadata.

METADATA :- The further details about data is called metadata

EX :-

SID	SNAME	PH_NO	IMAGE
1	TINKU	9807890876	☺
2	DINGI	9898789978	☺
3	DINGA	9897898786	☺
4	TINKI	8978967898	☺

$$\begin{tikzcd}
\textbf{PROPERTIES} & \textbf{DATA} \arrow[l]
\end{tikzcd}$$

IMG NAME - IMG001

SIZE - 2MB

FORMAT - JPEG

$$\begin{tikzcd}
\text{JPEG} \arrow[d] \\
\textbf{METADATA}
\end{tikzcd}$$

METATABLE

IMG_NAME	**SIZE**	**FORMAT**
IMG001	2MB	JPEG
IMG002	3MB	PDF
IMG003	2MB	JPEG
IMG004	3MB	PDF

AUTOGENERATED

RULE 3:- According to E. F CODD we can store the data in multiple table if necessary we can establish a connection between the tables with the help of key attribute.

For ex :- Student

SID	**SNAME**	**SUBID**
1	SMITH	101
2	WARD	102

Teacher

TID	**TNAME**	**SUB**
1	VANDNA	SQL
2	DIVYA	JAVA

RULE 4 :- Data entered into a table must validate in two steps.

- By assigning DATATYPE (mandatory)
- By assigning CONSTRAINT (optional)

DATATYPE :-

"It is used to specify which type of data to be stored in particular memory location."

TYPES OF DATATYPE :-

1. CHAR DATATYPE
2. VARCHAR DATATYPE /VARCHAR 2
3. DATE DATATYPE
4. NUMBER DATATYPE
5. LARGE OBJECT
   i. CHACTER LARGE OBJECT (CLOB)
   ii. BINARY LARGE OBJECT (BLOB)

CHAR DATATYPE :-

"It is used to stored the character such as uppercase (A-Z), Lowercase (a-z),digit (0-9), alphanumeric, special characters."

- Always character enclosed within single quotes.
- Whenever we use char datatype we need to metion size

SYNTAX :- CHAR (SIZE)

- The maximum number of charater can be stored in char datatype is upto 2000.
- Char follows fixed length memory allocation.

For ex:- Char (10)

‘AJAY’

A	J	A	Y						

→ USED MEMORY ← → UNUSED MEMORY ←

WASTED MEMORY

For Which Column We Can Use Char Datatype :-

- AADHARCARD, PN_NO, IFSC ,

For Which Column We Cannot Use Char Datatype :-

- ENAME, ADDRESS, EMAIL_ID, ACCOUNT_NO

VARCHAR:- variable

“It is used to stored the character such as uppercase (A-Z), Lowercase (a-z),digit (0-9), alphanumeric, special characters”

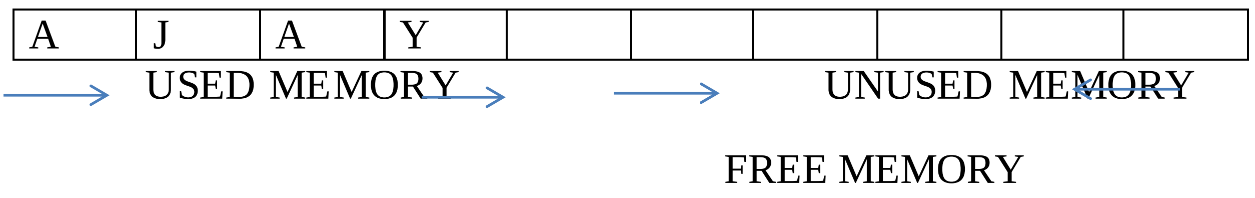
- Always character enclosed within single quotes.
- Whenever we use varchar datatype we need to metion size

SYNTAX :- VARCHAR (SIZE)

- The maximum number of charater can be stored in varchar datatype is upto 2000.
- Varchar follows variable length memory allocation.

For ex:- Varchar (10)

‘AJAY’



For Which Colummn We Can Use Varchar Datatype

- NAME, ADDRESS, AADHAR, EMAI_ID,

VARCHAR 2: -

“It is updated version of varchar.”

- We can store 4000 characters.

Syntax :- varchar2 (size)

DATE: -

“It is used to store the dates in particular format.”

- Always enclosed date in single quotes.

ORACLE SPECIFIED FORMAT :-

1. ‘DD-MON-YYYY’ $\longrightarrow$ ’01-JAN-2024’
2. ‘DD-MON-YY’ $\longrightarrow$ ’01-JUN-24’

For Which Colummn We Can Use Date Datatype

- Dob, Doj, Hiredate, Manufacture Date

NUMBER :-

“It is used store numeric values.”

SYNTAX :- NUMBER (PRECISION,[SCALE])

PRECISION :-

“It is used to store the integer values.”

- The range of precision is 1 – 38.

SCALE :-

“It is used store decimal value.”

- The range of scale of is -84 to 127
- The default values of scale is 0.

For ex :-

NUMBER (6)

$\downarrow$ P

+ - 9	9	9	9	9	9

NUMBER (7, 2)

+ - 9	9	9	9	9 .	9	9

$\leftarrow$

P>S

NUMBER (8, 4)

+ - 9	9	9	9 .	9	9	9	9

$\leftarrow$

NUMBER (39, 10) ✗

P = S

NUMBER (4,4) :-

+ - . 9	9	9	9

NUMBER (6,6)

+- .9	9	9	9	9	9

P<S

NUMBER (4,8) :- OPERATION

SCALE – PRECITION

8 – 4 = 4

+ - .0	0	0	0	9	9	9	9

NUMBER (2,7) :- 7- 2 = 5

+ - .0	0	0	0	0	9	9

For Which Column We Can Use This Scale

PRICE, SALARY, WEIGHT, PER .

LARGE OBJECT :-

"It is used to store large amount of data"

TYPES OF LARGE OBJECT :-

- **Character large object (CLOB)**
- **Binary large object (BLOB)**

2000 – VARCHAR
3000 - VARCHAR 2
4000 – VARCHAR 2
4002 - CLOB

1. Character large object (CLOB) :-

"It is used to store large amount of character upto 4GB."

For Which Column We Can Use This Clob :-

ESSAY

SUMMARY

NOVEL

2. BINARY LARGE OBJECT (BLOB) :-

" It is used to store binary value of image, video, document etc upto 4GB."

DATATYPE (MANDATORY)

CONSTRAINTS (OPTIONAL)

WHY TO LEARN CONSTRAINT :-

FOR EX :-

EMP

	EID	ENAME	SALARY	PH_NO
DATATYPE →	NUMBER(6)	VARCHAR(16)	NUMBER(6,2)	NUMBER(10)
	1	TINKU	2000.25	9878987897
	123456	TINKA	4500.00	8796787 ✖
	1234567 ✖	VIRUS	10000 ✖	9878987897 ✖
	-1234 ✖	PIKU	-5000.00 ✖	-8978897578 ✖

<u>CONTRAINT</u>: - RULE

"Constraints are rules given to a column to validate the data."

TYPES OF CONSTRAINT: -

1. UNIQUE
2. NOT NULL
3. CHECK
4. PRIMARY KEY
5. FOREIGN KEY

UNIQUE :- DIFFERENT

“It is a constraint which is used to avoid duplicate.”

Ex :-

EID
NUMBER(6) UNIQUE
1
2
3
1

→ERROR

For Which Column We Can Use This Unique Constraint :-

EX :- ID’S, PH_NO, AADHAR, ROLLNO, BANK AC/NO

NOT NULL :- EMPTY

“It is constraint which is used to avoid null into a column”

Ex :-

‘ROMA’

‘KISHORE’

KALPANA

ENAME
VARCHAR(15) NOT NULL
ROMA
KISHORE

ERROR

For which column we can use not null constraint

Name , address, (for each and every column we can use this not null)

CHECK :-

"It is a constraint which is used along with a condition if the condition is satisfied it will accept the value or it will reject the value."

Ex :-

8796787 ✖

9878987897 ✖

-8978897578 ✖

PH_NO

NUMBER(10)
Check (length(ph_no) = 10)
Unique
Check(length(ph_no >0)

9878987897

For which column we can use this check

Salary- check(sal > 0)

Id – check (id > 0)

Age - check (age >= 18)

Check constraint applied for numeric value.

PRIMARY KEY :-

" It is used to identify the records uniquely from the table."

CHARACTRISTICS OF FOREIGN KEY:-

- P.k cannot accept duplicate /repeated values.
- P.k cannot accept null.
- It is a combination of unique and not null constraint
- We can have only one primary key in a table

FOREIGN KEY :-

"It is constraint which is used to establish connection between the table."

CHARACTRISTICS OF FOREIGN KEY:-

- F. k can accept duplicate /repeated values
- F. k can accept null
- It is not a combination of unique and not null.
- We can have more than one foreign key in a table.
- F. k also known as REFERENCIAL INTEGRITY CONSTRAINT
- For attribute (column) to become a f. k It must be a P. k in its own table.

Ex :- EMP

EID	**ENAME**	**SAL**	**PH_NO**	**DNO**
NUMBER(5)	VARCHAR(15)	NUMBER(6,2)	NUMBER(10)	
UNIQUE **P. K** NN CHECK(EID >0)	- NN -	- NN CHECK(SAL>0)	UNIQUE NN CHECK(length(PHno)= 10) CHECK (PH_NO >0)	**F.K**
1	SMITH	2000.34	7898767867	10
2	ALLEN	3500.67	7649378495	30
3	WARD	4500.25	8976578997	20
4	MILLER	2500.56	9867895678	10
5	SCOTT	3400.00	7898789989	30
6	JONES	10000	7898878787	20

CHILD TABLE

DEPT

DNO P.K	**DNAME**	**LOC**
10	RESEACH	PUNE
20	SALES	BANGLORE
30	OPERATION	MUMBAI

PARENT TABLE

KEY ATTRIBUTE :- A column which is having unique records.

- Table which is having or consisting of f.k are called child table.

  OR

  Table which is having the reference of another table is called child table.
- Table which is refering a column to another table is called parent table.

OVERVIEW OF SQL STATEMENT

WE HAVE 5 TYPE OF SQL STATEMENTS

1. DATA DEFINATION LANGUAGE (DDL)
2. DATA MANIPULATION LANGUAGE (DML)
3. TRANSACTION CONTROL LANGUAGE (TCL)
4. DATA CONTROL LANGUAGE (DCL)
5. DATA QUERY LANGUAGE (DQL)

DATA DEFINATION LANGUAGE :-

It is used to create or construct an object/table/entity.

DATA MANIPULATION LANGUAGE :-

It is used manipulate the table by inserting ,updating , deleting the data.

TRANSACTION CONTROL LANGUAGE :-

It is used to control the trasaction in the table

TRANSACTION :- the operation performed on DML is known as transaction

DATA CONTROL LANGUAGE :-

It is used to control the flow of data b/w the user/person

DATA QUERY LANGUAGE :-

It is used to retrieve the data from the database.

SOFTWARE LINK FOR WINDOWS: **- bit.ly/roSoftWIN**

SOFTWARE LINK FOR MAC: **- bit.ly/roSoftMAC**

DATA QUERY LANGUAGE

"It is used to retrieve the data from the database."

We Have 4 DQL Statements

1. SELECT
2. PROJECTION
3. SELECTION
4. JOINS

SELECT: -

"It is used to retrieve the data from the table and it will display the data on output screen."

PROJECTION: -

"It is used to retrieve the data from the table by selecting only the columns"

SELECTION: -

"It is used to retrieve the data from the table by selecting rows as well as columns."

JOINS: -

"It is used to retrieve the data from the multiple table simultaneously."

PROJECTION

"It is used to retrieve the data from the table by selecting only column."

SYNTAX: -

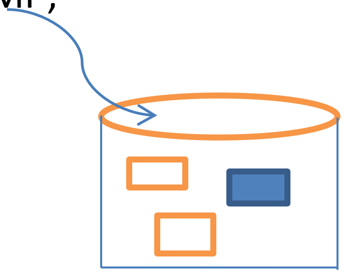
```
SELECT */ [DISTINCT] COLNAME / EXPRESSION[ALIAS]
FROM TABLENAME;
```

ORDER OF EXECUTION: -

1. FROM
2. SELECT

Q. WAQTD NAME OF THE EMPLOYEES.

```
SELECT ENAME
FROM EMP;
```



EMP

EID	ENAME	SAL	DEPTNO
1	SMITH	2000	20
2	ALLEN	2500	10
3	WARD	1800	20
4	MILLER	3000	30
5	JONES	4000	10
6	SCOTT	2000	30
7	TURNER	1000	10

ENAME
SMITH
ALLEN
WARD
MILLER
JONES
SCOTT
TURNER

O/P OF SELECT

DISTINCT

"It is used to remove duplicate/repeated values from the result table."

SYNTAX: -

```
SELECT */[DISTINCT]COLNAME/EXPRESSION[ALIAS]

FROM TABLENAME;
```

CHARACTERISTIC OF DISTINCT: -

1. Distinct clause has to be written as a first argument in select clause.
2. Column name should be written after distinct clause
3. We can pass multiple column name to distinct and it will remove the combination of column in which values are present.

Q. WAQTD different salary of an employee.

```
SELECT DISTINCT SAL

FROM EMP ;
```

EID	ENAME	SAL	DEPTNO
1	SMITH	2000	20
2	ALLEN	2500	10
3	WARD	1800	20
4	MILLER	3000	30
5	JONES	4000	10
6	SCOTT	2000	30
7	TURNER	1000	10

SAL
2000
2500
1800
3000
4000
2000
1000

SAL
2000
2500
1800
3000
4000
1000

EXPRESSION

Any statement which gives result is known as expression.

OR

Expression is combination of operator and operand

Ex :- 5 ✖

5 + ✖ operator

5 + 2

Operand

OPERATOR :- Symbols which is used to perform specific tasks is known as operators

OPERAND :- The values which is passed a user is called expression.

Q. WAQTD annual salary of an employee.

```
SELECT SAL*12
```

```
FROM EMP ;
```

Q. WAQTD deduction of 200 from the salary.

```
SELECT SAL – 200
FROM EMP ;
```

Q. WAQTD both annual salary and monthly salary.

```
SELECT  SAL , SAL * 12
FROM EMP ;
```

Q. WAQTD details of the employee along with annual salary.

```
SELECT * ,SAL *12
 FROM EMP ;


SELECT EMP.*, SAL *12
FROM EMP;
```

ALIAS

It is used to give a alternative name to a column or an expression.

- We can use ALIAS with or without using a keyword called ‘AS’
- To give a space b/w alias name
    1. ‘__’
    2. “ ”

SELECTION

It is used used to retrieve the data from the table by selecting both rows and columns.

SYNTAX :-

SELECT */[DISTINCT] COLNAME /EXPRESSION [ALIAS]

FROM TABLENAME

WHERE < FILTER_CONDITION >;

ORDER OF EXECUTION :-

1. FROM
2. WHERE
3. SELECT

WHERE

" It is used to filter the records from the table.”

- Where clause execute row by row
- Where execute after the execution of from clause
- We can pass multiple condition in where clause with the help of logical operator.

Q. WAQTD name of emp earning more than 2000.

```
SELECT ENAME
FROM EMP
WHERE SAL > 2000 ;
```

EMP

EID	ENAME	SAL	DEPTNO
1	SMITH	2000	20
2	ALLEN	2500	10
3	WARD	1800	20
4	MILLER	3000	30
5	JONES	4000	10
6	SCOTT	2000	30
7	TURNER	1000	10

SAL > 2000

2000 > 2000

2500 > 2000

1800 > 2000

3000 > 2000

4000 > 2000

2000 > 2000

1000 > 2000

O/P OF WHERE CLAUSE

ENAME
ALLEN
MILLER
JONES

O/P OF SELECT

Q. WAQTD name of employee, designation and sal of the emp who is earning less 2000.

```
SELECT ENAME
FROM EMP
WHERE SAL < 2000 ;
```

Q. WAQTD name of employee who is working as clerk.

```
SELECT ENAME
 FROM EMP
 WHERE JOB = 'CLERK' ;
```

Q. WAQTD name of employee who is working as MANAGER.

Q. WAQTD HIREDATE & COMM of employee who is NAME IS SMITH.

OPERATOR

These are the symbols which is used to perform a specific task.

TYPES OF OPERATORS :-

1. Arithmetic operator: - **(+, -, *, /)**
2. Comparison operator: **- (=, !=)**
3. Relational operator: - **(>, <, >=, <=)**
4. Concatenation operator: - **(||)**
5. Logical operator (AND, OR, NOT)
6. Special operator: -
    1. IN
    2. NOT IN
    3. BETWEEN
    4. NOT BETWEEN
    5. IS
    6. IS NOT
    7. LIKE
    8. NOT LIKE
7. SUBQUERY OPERATORS:-
    1. ALL
    2. ANY
    3. EXISTS
    4. NOT EXISTS

CONCATENATION OPERATOR: -

"It is used to join more than one strings"

SYMBOL :- **(||)** $\longrightarrow$ pipe symbol

Q. WAQTD NAMES OF EMP

```
SELECT ENAME
FROM EMP;
```

ENAME
SMITH
ALLEN
WARD
MILLER
JONES
SCOTT
TURNER

Q. WAQTD NAME OF EMP

```
SELECT 'MR.' || ENAME
FROM EMP;
```

ENAME
MR.SMITH
MR.ALLEN
MR.WARD
MR.MILLER
MR.JONES
MR.SCOTT
MR.TURNER

LOGICAL OPERATER: -

A Logical operator is a symbol or word used connect two or more expressions.

$\left.\begin{array}{l}\text{1. AND}\\ \text{2. OR}\end{array}\right\}$ binary operator

3. NOT → unary operator

AND: - It returns true when all the condition is satisfied at RHS

OR: - It returns true when any one value of the condition is satisfied.

NOT :- it is unary operator

It is used to reverse the given output.

SPECIAL OPERATOR: -

IN: - In operator is multi valued operator which is used to accept multiple values at RHS.

SYNTAX: - Colname IN (V1, V2, V3, V4, V5……. Vn);

Q. WAQTD details of the employee working in dept 10, 20, 30, 40

```
    Select  *
    From emp
  Where  deptno = 10 OR deptno = 20 OR deptno = 30 OR
deptno = 40;
```

```
  Select  *
  From emp
Where  deptno IN (10, 20, 30, 40)
```

NOT IN: -

"It is similar to IN operator instead of selecting the values at RHS it will reject the values.

SYNTAX: -

Column name NOT IN (V1,V2, V3, V4………..Vn);

Q. WAQTD details of the employee except manager as well as analyst.

```
Select *
From emp          ✗
Where job != 'MANAGER', 'ANALYST';
```

```
Select  *
from emp          ✗
where not job = 'MANAGER', 'ANALYST';
```

```
 Select  *
from emp
where job not in ('MANAGER', 'ANALYST') ;
```

BETWEEN AND NOT BETWEEN OPERATOR

BETWEEN: -

Between operator is used whenever we have range of value than we use between operator.

$$\begin{tikzcd} \text{More than} & 2000 \arrow[d, leftrightarrow] \\ \text{Less than} & 5000 \end{tikzcd}$$

Between operator includes the range

SYNTAX: -

```
Colname BETWEEN Lower range AND Higher range;
```

Q. WAQTD names of the employee who is earning more than 2000 But less than 5000

```
SELECT ENAME
FROM EMP
WHERE SAL > 2000 AND SAL < 5000;
```

```
SELECT ENAME
FROM EMP
WHERE SAL BETWEEN 2001 AND 4999;
```

Q. WAQTD names of the employee who is earning annual sal more than 10,000 but less than 40,000.

Q. WAQTD details of the employee who is hired after 1980 but hired before 1987.

```
Select *
From emp
Where hiredate between ‘01-jan-81’ and ‘31-dec-86’;
```

OR

```
Select *
From emp
Where hiredate > ’31-DEC-80’ and ’01-JAN-87’;
```

OR

```
Select *
From emp
Where hiredate >= ‘01-JAN-81’ and hiredate <= ‘31-DEC-86’;
```

Q. WAQTD between 1980 to 1987

```
Select *
From emp
Where hiredate between ‘01-jan-81’ and ‘31-dec-86’;
```

NOT BETWEEN OPERATOR: -

It is similar to between operator, instead of selecting the range it will reject the range.

SYNTAX: -

Colname NOT BETWEEN lower range AND higher range;

Q. WAQTD details of the employee who is not earning salary between 2000 to 3800.

```
Select *
From emp
Where sal not between 2001 and 3799;
```

IS OPERATOR: -

Is operator is special operator which used to compare with only null.

OR

Is operator is used to compare with empty cell.

SYNTAX: - Colname IS null;

Q. WAQTD details of the employee who is not earning any comm.

```
 Select *
From emp
Where comm is null;
```

IS NOT OPERATOR: -

Is not operator is similar to is operator, instead of select the null it will null

SYNTAX: - Colname IS NOT null;

Q. WAQTD details of employee who is earning salary.

```
Select *

From emp

Where sal is not null;
```

Q. WAQTD details of employee who is earning salary but not commission.

LIKE OPERATOR: -

It is used for pattern matching process.

SYNTAX: - Colname LIKE ‘pattern_to_match’;

Q. WAQTD name of the emp who’s names starts with character A.

```
SELECT ENAME

FROM EMP

WHERE  ENAME LIKE ‘A’ ;
```

% → Any no. character

Any no. time

Any char

_ → It will take only one character but any character.

Q. WAQTD name of the emp who's names starts with character A.

```
Select ename
From emp
Where ename like 'A%';
```

EID	ENAME	SAL	DEPTNO
1	SMITH	2000	20
2	ALLEN	2500	10
3	WARD	1800	20
4	MILLER	3000	30
5	JONES	4000	10
6	SCOTT	2000	30
7	TURNER	1000	10

ENAME
ALLEN

Q. WAQTD name of the emp who's names ends with char 'R'.

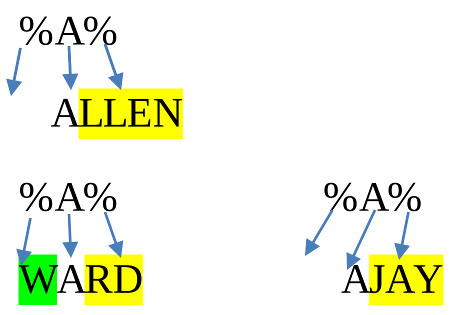
```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE '%R';
```

ENAME
MILLER
TURNER

Q. WAQTD name of the emp who's names have char A in it.

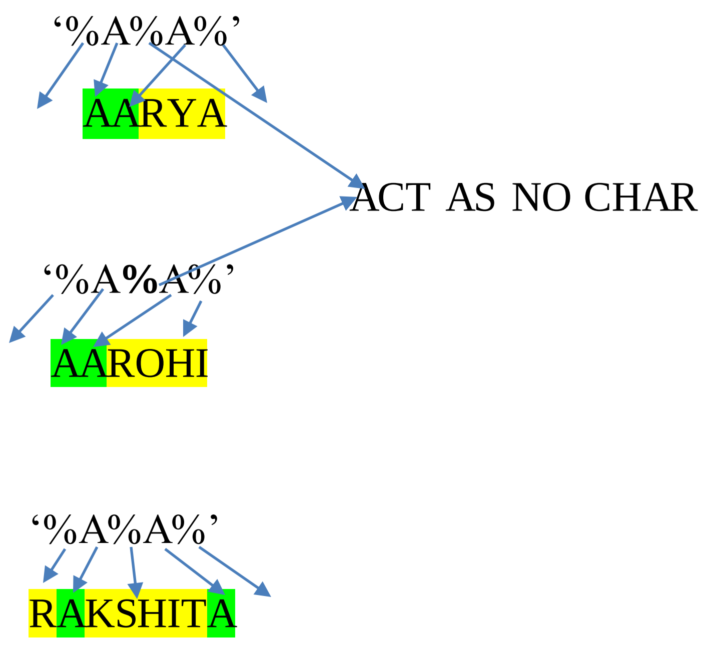
```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE '%A%';
```

ENAME
ALLEN
WARD



Q. WAQTD details of the emp who have 2 time char A in the name.

```
SELECT *                              AARYA
FROM EMP                              AAROHI
WHERE ENAME LIKE '%A%A%';             RAKSHITA
```



WHEN TO USE UNDERSCORE: -

Q. WAQTD names of the employee who having char A in 2 position.

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

Q. WAQTD names of the employee who have last 2 char is E.

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

Q. WAQTD names of the employee who have char A in 3 position.

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

NOT LIKE: -

"It is similar to like operator instead of selecting the pattern it will reject the pattern."

SYNTAX: -

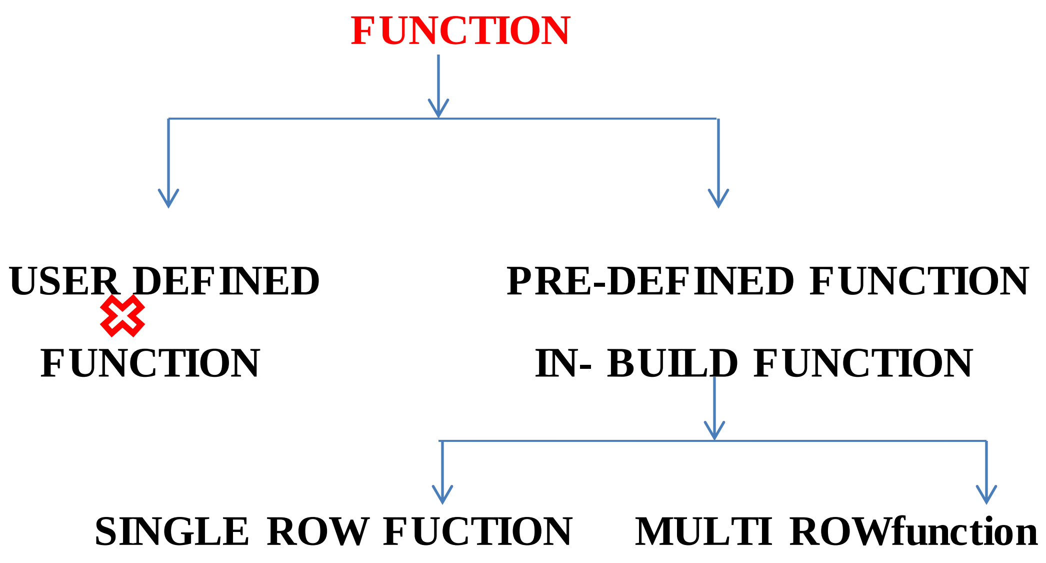
```
Colname NOT LIKE 'Pattern_to_match' ;
```

Q. WAQTD details of the emp who's name does not start with char A

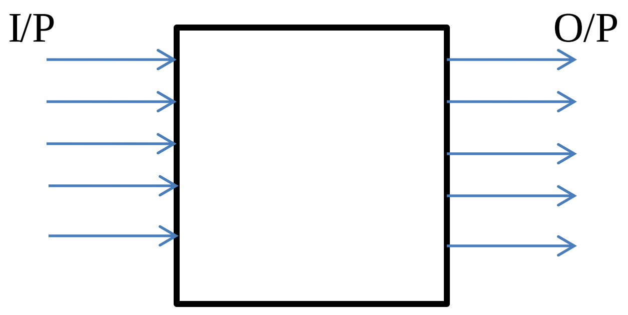
```
SELECT *
FROM EMP
WHERE ENAME  NOT LIKE 'A%' ;
```

FUNCTION : -

It is block of code/a set of instruction which is used to perform specific task.



<u>SINGLE ROW FUNCTION</u>: -



Length( ): - It is used count no. of character in a given value.

Q. WAQTD no. of character present in the employee's name.

```
SELECT LENGTH(ENAME)
FROM EMP;
```

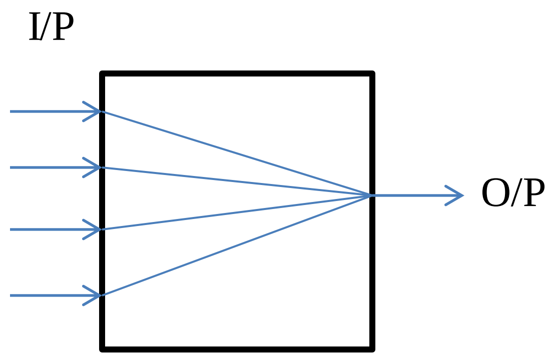
ENAME
SMITH
ALLEN
WARD
MILLER
JONES
SCOTT
TURNER

LENGTH(ENAME)
5
5
4
6
5
5
6

- SINGLE ROW FUNCTION will take N no. of input and it will give N no. of output.
- It is execute row by row.
- It will take value one by one and execute and give o/p one by one.

<u>MULTI ROW FINCTION</u>: - aggregate(), Group function

"It will take no. of input in single short and execute, them and generate a single output."



- MRF execute group by group
- If we pass N no. of input it will give a single output.

Q. WAQTD maximum salary of the emp.

```
Select MAX(SAL)
FROM EMP ;
```

OUTPUT

MAX(SAL)
4000

EMP

EID	ENAME	SAL	DEPTNO
1	SMITH	2000	20
2	ALLEN	2500	10
3	WARD	1800	20
4	MILLER	3000	30
5	JONES	4000	10
6	SCOTT	2000	30
7	TURNER	1000	10

LIST OF MRF :-

1. MAX ( ) :-
2. MIN ( ) :-
3. AVG ( ) :-
4. SUM ( ) :-
5. COUNT ( ) :- It is used to count the no. of value present in the given table.

RULES OF MRF: -

- MRF can accept only one Argument. i.e: - colname/exp

  Ex: - maximum salary of the emp.

  ```
  Select max(sal, comm)    ✗
  From emp;
  ```

- Along with MRF, we are not suppose to any other column name/any exp.

  Ex: - maximum salary,name of the emp. ✗

  ```
  Select max(sal), ename,
  From emp;
  ```

- MRF ignores null.

  Ex: -

  ```
  Select count(comm)
  From emp;
  ```

- We cannot use MRF in where clause.

  Ex: -

  ```
  Select *          ✗
  From emp
  Where max(sal) > 2000;
  ```

- Count is the only MRF, which accept asterisk(*) as an argument.

  Ex: - max(*) ✗

Min(*) ✗
Count(*)

Q. WAQTD MAX SALARY OF THE EMP WHO IS WORKING AS CLERK.

```
SELECT MAX(SAL)
FROM EMP
 WHERE JOB = 'CLERK' ;
```

Q. WAQTD NO. OF EMPLOYEE WORKING AS ANALYST.

```
SELECT  COUNT(*)
FROM   EMP
WHERE  JOB = 'ANALYST'  ;
```

Q. WAQTD AVERAGE SALARY GIVEN TO MANAGER.

```
SELECT  AVG(SAL)
FROM EMP
WHERE  JOB = 'MANAGER';
```

Q. WAQTD MIN SALARY GIVEN TO SALESMAN.

```
SELECT MIN(SAL)
 FROM EMP
  WHERE JOB = 'SALESMAN' ;
```

Q. WAQTD NO. OF THE EMP AND MIN SALARY GIVEN TO THE EMP

IF EMP NAME STARTS WITH 'S' AND THEIR HALF TERM SALARY IS MORE THAN 3000 AND EMP ARE EARNING COMMAND THEY ARE EARNING 4 DIGIT SALARY.

```
        SELECT COUNT(*), MIN(SAL)
        FROM EMP
        WHERE ENAME LIKE 'S%' AND SAL*6 > 3000 AND COMM IS
NOT NULL AND SAL LIKE '____';
```

Q. WAQTD NO. OF EMP GETTING SALARY LESS THAN 2000 IN DEPTNO 10.

```
        SELECT COUNT(*)
        FROM EMP
        WHERE SAL < 2000 AND DEPTNO = 10 ;
```

GROUP BY CLAUSE: -

"Group by clause is used group the record."

SYNTAX: -

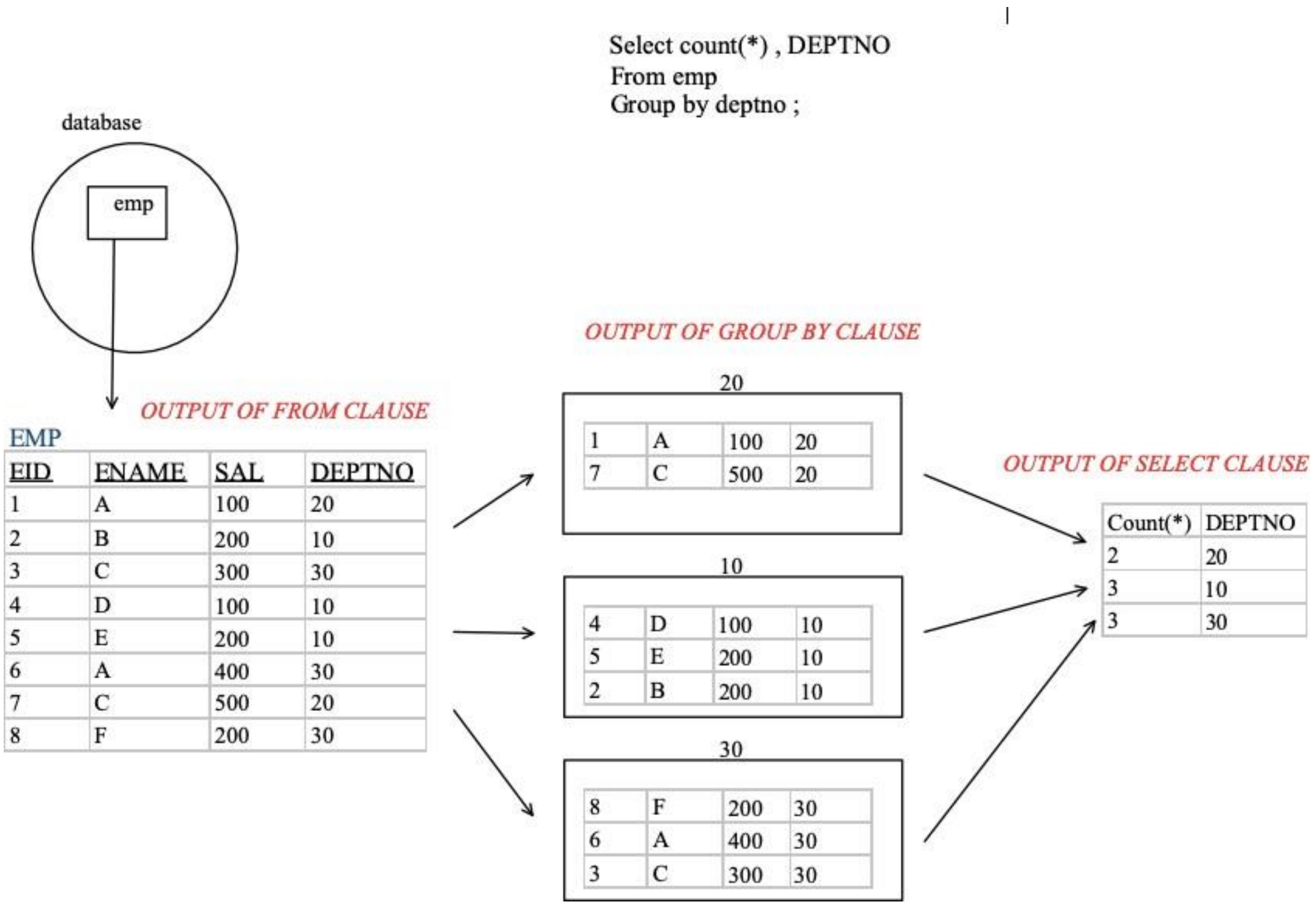
```
        SELECT GROUP FUNCTION /GROUP_BY_EXP
        FROM TABLENAME
        [WHERE <FILTER_CONDITION>]
        GROUP BY COLNAME/EXPRESSION;
```

ORDER OF EXECUTION: -

1. FROM
2. WHERE (if used) ROW BY ROW
3. GROUP ROW BY ROW
4. SELECT GROUP BY GROUP

Q. WAQTD NO. OF EMP WORKING IN EACH DEPT

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



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

1	A	100	20
7	C	500	20

4	D	100	10
5	E	200	10
2	B	200	10

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

Count(*)	DEPTNO
2	20
3	10
3	30

NOTE: -

1. It is used to group the records.
2. Group by clause executes Row by Row.
3. After the execution group by clause if any clause execute it will group by group
4. Group by clause can be used without using where clause.
5. Group by clause execute after the execution where clause (if used) or else group by clause execute after the from clause.
6. The colname/exp used in Group by clause can used in select clause also which is known as Group by expression.

Q. WAQTD NO. OF THE WORKING IN EACH DEPARTMENT EXECPT PRESIDENT.

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

Q. WAQTD NUMBER OF EMPLOYEES AND AVG SALARY NEEDED TO PAY THE EMPLOYEES WHO SALARY IN GREATER THAN 2000 IN EACH DEPT.

```
SELECT COUNT(*), AVG(SAL), DEPTNO
FROM EMP
WHERE SAL > 2000
GROUP BY DEPTNO;
```

Q. WAQTD NO. OF TIME THE SALARIES PRESENT IN EMP TABLE.

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

Q. WAQTD MAX SAL GIVEN AN EMP WORKING IN EACH DEPT.

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

HAVING CLAUSE: -

"It is used to filter the group."

SYNTAX: -

```
SELECT GROUP FUNCTION /GROUP_BY_EXP
FROM TABLENAME
[WHERE <FILTER_CONDITION>]
GROUP BY COLNAME/EXPRESSION
 HAVING < GROUP_FILTER_CONDITION>;
```

ORDER OF EXECUTION

1.FROM

2.WHERE (IF USED) → ROW BY ROW

3.GROUP BY → ROW BY ROW

4.HAVING → GROUP BY GROUP

5.SELECT → GROUP BY GROUP

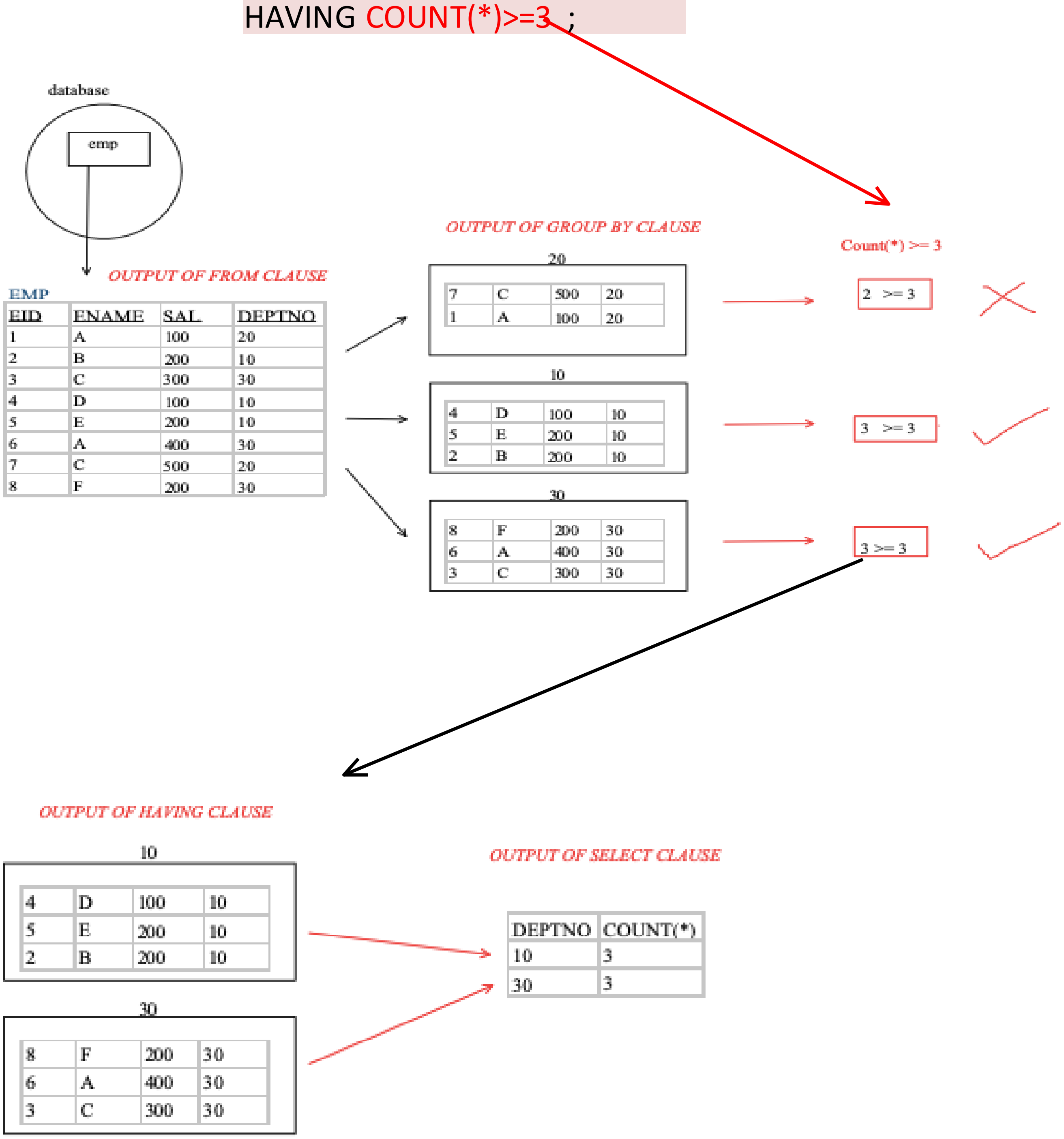
NOTES: -

- It is used to filter the group
- Having clause execute group by group
- Having clause executes after the execution group by clause.
- In having clause we can write group filter condition
- We can use MRF in having clause.

Q. WAQTD no. of employees working in each dept, if there are at least 3 emp.

```
SELECT COUNT(*), DEPTNO

FROM EMP

GROUP BY DEPTNO

HAVING COUNT(*) >= 3;
```

HAVING COUNT(*)>=3 ;



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

20

7	C	500	20
1	A	100	20

10

4	D	100	10
5	E	200	10
2	B	200	10

30

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

10

4	D	100	10
5	E	200	10
2	B	200	10

30

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

DEPTNO	COUNT(*)
10	3
30	3

Q. WAQTD max sal of the emp in each job if the emp max sal is more than 3000.

```
SELECT MAX(SAL), JOB
FROM EMP
GROUP BY JOB
HAVING MAX(SAL) > 3000;
```

Q. WAQTD no. of the emp in each dept if the emp min sal is less than max salary and their name start with and atleast 2 emp working in each dept.

```
SELECT COUNT(*), DEPTNO
FROM EMP
WHERE ENAME LIKE ‘A%’
GROUP BY DEPTN0
HAVING MIN(SAL) < MAX(SAL) AND COUNT(*) >= 2;
```

Q. WAQTD salary of the employee if they are getting same sal.

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

Q. WAQTD MAX SAL OF THE EMP IF THE EMP IS HIRED ON SAME DATE.

```
SELECT MAX(SAL)

FROM EMP

GROUP BY HIREDATE

HAVING HIREDATE > 1;
```

ORDER BY CLAUSE: -

"It is used to arrange the records either in ascending order or in descending."

SYNTAX: -

```
SELECT COLNAME/EXP

FROM TABLENAME

[WHERE <filter_condition>]

ORDER BY COLNAME/EXP   [ASC] / DESC;
```

ORDER OF EXECUTION: -

1. FROM
2. WHERE (if used)
3. SELECT
4. ORDER BY

NOTES: -

- It is used to arrange the records either in ascending order or in descending.
- By default, order by clause arranges records in ascending.
- Order by clause is only one clause which execute after the execution of select clause.
- In Order by clause we can pass multiple column as an argument.

EMP

EID	ENAME	SAL	DEPTNO
1	SMITH	2000	20
2	ALLEN	2500	10
3	WARD	1800	20
4	MILLER	3000	30
5	JONES	4000	10
6	SCOTT	2000	30
7	TURNER	1000	10

Q. WAQTD details of the employee and arrange the salary in ascending order.

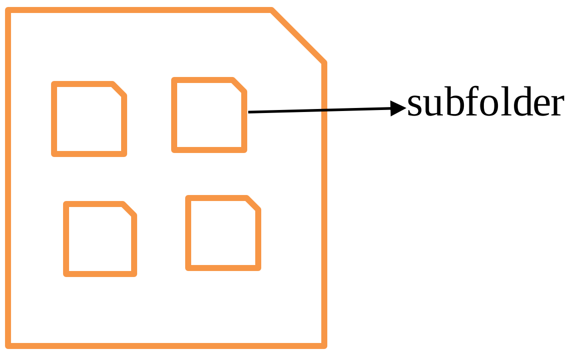
```
SELECT *

FROM EMP

ORDER BY SAL;
```

Q. WAQTD details of the employee and arrange the salary in descending order.

```
SELECT *

FROM EMP

ORDER BY SAL DESC;
```

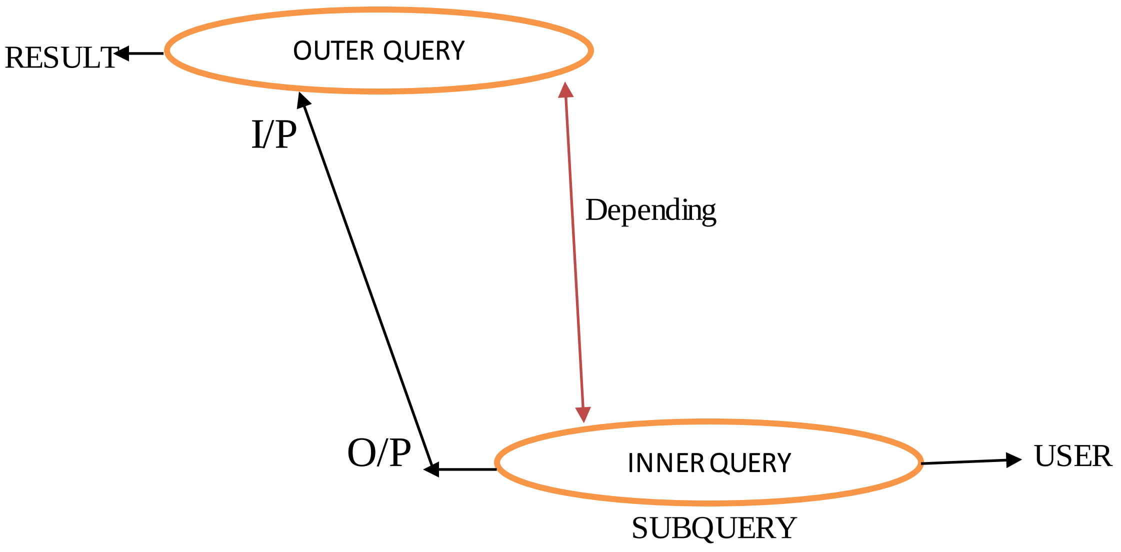
SUBQUERY: - **Inner query**

"A query written inside another query is known as subquery"



folder

WORKING PRINCIPLE OF SUBQUERY: -



NOTE: -

- Let us consider two queries outer query and inner query.
- Inner query also known as subquery.
- Always inner query will execute first and it produce output.
- The output of inner query will pass as an input outer query.
- The outer query will execute and generate the final RESULT.
- Such that we can say that outer query is dependent on inner query.

When/Why do we use subquery: -

CASE 1: -

"Whenever there is unknown present in the question we have to use subquery."

Ex: -

Q. WAQTD name of the employee earning more than 2000.

```
SELECT ENAME

FROM EMP

WHERE SAL > 2000;
```

Q. WAQTD name of the employee earning more than smith.

```
SELECT ENAME

FROM EMP

WHERE SAL > (SELECT SAL

             FROM EMP

              WHERE ENAME = 'SMITH');
```

```
SELECT ENAME → ALLEN, JONES,MILLER
FROM EMP                      2000
WHERE SAL > (SELECT SAL
             FROM EMP
             WHERE ENAME = ‘SMITH’);
```

EID	ENAME	SAL	DEPTNO
1	SMITH	2000	20
2	ALLEN	2500	10
3	WARD	1800	20
4	MILLER	3000	30
5	JONES	4000	10
6	SCOTT	2000	30
7	TURNER	1000	10

SAL > 2000

2000 > 2000

2500 > 2000

1800 > 2000

3000 > 2000

4000 > 2000

2000 > 2000

1000 > 2000

Q. WAQTD details of the emp earning less than scott.

```
SELECT *
FROM EMP              2000
WHERE SAL < (SELECT SAL
             FROM EMP
             WHERE ENAME = ‘SCOTT’);
```

Q. WAQTD details of the emp hired before allen.

```
SELECT *
FROM EMP
WHERE HIREDATE < (SELECT HIREDATE
                  FROM EMP
                  WHERE ENAME = ‘ALLEN’);
```

Q. WAQTD DETAILS OF EMP WHO IS EARNING MORE THAN SCOTT BUT LESS THAN KING.

```
SELECT *
FROM EMP
WHERE SAL > (SELECT SAL
             FROM EMP
             WHERE ENAME = ‘SCOTT’) AND
              SAL < (SELECT SAL
                     FROM EMP
                     WHERE ENAME = ‘KING’);
```

WHY / WHEN DO WE USE SUBQUERY.

CASE – II: -

"Whenever the data to be select and condition to be executed present in different tables than we can use subquery case2."

Q. WAQTD ENAME OF THE EMPLOYEE ERANING MORE THAN SMITH.

$$\begin{tikzcd} \text{ENAME} \arrow[d] & \text{ERANING MORE THAN SMITH} \arrow[d] \\ \text{EMP} & \text{EMP} \end{tikzcd}$$

EMP

EID	ENAME	SAL	DEPTNO
1	SMITH	2000	20
2	ALLEN	2500	10
3	WARD	1800	20
4	MILLER	3000	30
5	JONES	4000	10
6	SCOTT	2000	30
7	TURNER	1000	10
8	MONIKA	3000	20

$\leftrightarrow$

DEPT

DEPTNO	DNAME	LOC
10	RESEARCH	MUMBAI
20	OPERATION	BANGLORE
30	SALES	PUNE

Q. WAQTD DNAME OF THE EMPLOYEE WHOSE NAME IS SMITH.

$$\begin{tikzcd} \text{DNAME} \arrow[d] & \text{WHOSE NAME IS SMITH} \arrow[d] \\ \text{DEPT} & \text{EMP} \end{tikzcd}$$

```
SELECT DNAME → OPERATION
FROM DEPT                20
WHERE DEPTNO IN (SELECT DEPTNO
                 FROM EMP
                 WHERE ENAME = 'SMITH');
```

DEPTNO = 20

10 = 20

20 = 20

30 = 20

Q. WAQTD LOC OF THE EMP WHOSE NAME IS MONIKA.

```
SELECT LOC → BANGLORE
FROM DEPT                              20
WHERE DEPTNO IN (SELECT DEPTNO
                 FROM EMP
                 WHERE ENAME = 'MONIKA');
```

DEPTNO = 20

```
10 = 20
20 = 20
30 = 20
```

Q. WAQTD DNAME AND LOC OF THE MILLER.

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

Q. WAQTD NAMES OF THE EMPLOYEES EARNING MORE THAN SCOTT IN ACCOUNTING DEPT.

```
SELECT ENAME
FROM EMP
WHERE SAL > (SELECT SAL
             FROM EMP
             WHERE ENAME = ‘SCOTT’) AND
             DEPTNO (SELECT DEPTNO
                     FROM DEPT
                     WHERE DNAME = ‘ACCOUNTING’);
```

MAX & MIN: -

Q. WAQTD MAX SALARY OF AN EMPLOYEE.

```
SELECT MAX(SAL)
FROM EMP;
```

Q. WAQTD NAME OF THE EMPLOYEE WHO IS EARNING MAXIMUM SALARY.

```
SELECT ENAME → JONES
FROM EMP              4000
WHERE SAL IN (SELECT MAX(SAL)
              FROM EMP);
```

EMP

EID	ENAME	SAL	DEPTNO
1	SMITH	2000	20
2	ALLEN	2500	10
3	WARD	1800	20
4	MILLER	3000	30
5	JONES	4000	10
6	SCOTT	2000	30
7	TURNER	1000	10

SAL = 4000

2000 = 4000
2500 = 4000
1800 = 4000
3000 = 4000
4000 = 4000
2000 = 4000
1000 = 4000

Q. WAQTD NAME SALARY OF THE EMPLOYEE IS GETTING THE MINIMUM SALARY

```
SELECT ENAME, SAL  →TURNER,1000        SAL = 1000

FROM EMP                1000           2000 = 1000

WHERE SAL = (SELECT MIN(SAL)           2500 = 1000

                FROM EMP);             1800 = 1000

                                       3000 = 1000

                                       4000 = 1000

                                       2000 = 1000

                                       1000 = 1000
```

Q. WAQTD NAME OF THE EMPLOYEE WHO HIRED FIRST.

```
SELECT ENAME

FROM EMP

WHERE HIREDATE IN (SELECT MIN(HIREDATE)

                      FROM EMP);
```

Q. WAQTD DNAME OF THE EMPLOYEE WHO IS EARNING MAXIMUM SALARY.

```
SELECT DNAME
FROM DEPT
WHERE DEPTNO IN (SELECT DEPTNO
                 FROM EMP
                 WHERE SAL = (SELECT MAX(SAL)
                              FROM EMP );
```

TYPES OF SUBQUERY: -

1. Single row subquery
2. Multi row subquery

EMP

EID	ENAME	SAL	DEPTNO
1	SMITH	2000	20
2	ALLEN	2500	10
3	WARD	1800	20
4	MILLER	3000	30
5	JONES	4000	10
6	SCOTT	2000	30
7	TURNER	1000	10
8	MONIKA	3000	20

DEPT

DEPTNO	DNAME	LOC
10	RESEARCH	MUMBAI
20	OPERATION	BANGLORE
30	SALES	PUNE

Q. WAQTD DNAME OF ALLEN.

```
SELECT DNAME → RESEARCH
FROM DEPT                10
WHERE DEPTNO IN (SELECT DEPTNO
                    FROM EMP
                    WHERE ENAME = 'ALLEN');
```

Q. WAQTD DNAME OF ALLEN AND WARD

```
SELECT DNAME → RESEARCH
FROM DEPT              10,20
WHERE DEPTNO IN (SELECT DEPTNO
                 FROM EMP
                 WHERE ENAME IN ('ALLEN', 'WARD'));
```

1. **<u>SINGLE ROW SUBQUERY</u>: -**

   If a subquery returns exactly one value/ record we can call it as single row subquery.

   If a subquery returns single value than we can use both normal and special operator.

```
SELECT DNAME → RESEARCH
FROM DEPT              10
WHERE DEPTNO  =  (SELECT DEPTNO
                  FROM EMP
= Or IN           WHERE ENAME = 'ALLEN');
```

2. **MULTI ROW SUBQUERY: -**

If a subquery returns more than one value/ record we can call it as multi row subquery.

If a subquery returns more than one value then we cannot use and normal operator we must use in operator.

```
SELECT DNAME →  RESEARCH
FROM DEPT                       10,20
WHERE DEPTNO = (SELECT DEPTNO
                FROM EMP
        IN      WHERE ENAME IN ('ALLEN', 'WARD'));
```

EXAMPLE: -

Q. WAQTD details of employee who is working in location CHICAGO.

```
SELECT *
FROM EMP
WHERE DEPTNO IN (SELECT DEPTNO
                 FROM DEPT
                 WHERE LOC = 'CHICAGO');
```

Q. WAQTD details of employee who is working in location Chicago and NW.

```
SELECT *
FROM EMP
WHERE DEPTNO IN (SELECT DEPTNO
                FROM DEPT
                WHERE LOC IN ('CHICAGO','NEWYORK'));
```

Q. WAQTD details of the employee who is earning more than employee of dept 20.

```
           SELECT *
           FROM EMP                2000,2500,3000
           WHERE SAL > (SELECT SAL
Normal operator            FROM EMP
                           WHERE DEPTNO = 20);
```

SUBQUERY OPERATOR:-

1. ALL

2. ANY

1. **ALL OPERATOR:-** It is a special operator which is used along with relation operator (>, <, >=, <=) to compare the values present at RHS.

- All operator returns true, if all the values at RHS satisfied the condition.

EID	ENAME	SAL	DEPTNO
1	SMITH	2000	20
2	ALLEN	2500	10
3	WARD	1800	20
4	MILLER	3000	30
5	JONES	4000	10
6	SCOTT	2000	30
7	TURNER	1000	10
8	MONIKA	3000	20

Q. WAQTD details of the employee who is earning more than the employee of deptno 20.

```
    SELECT *
    FROM EMP              (2000, 1800, 3000)
    WHERE SAL >ALL (SELECT SAL
                    FROM EMP
                    WHERE DEPTNO = 20);
```

2000 >2000 ✗	2500 > 2000	1800 > 2000 ✗
2000 > 1800	2500 > 1800	1800 > 1800 ✗
2000 > 3000 ✗	2500 > 3000 ✗	1800 > 3000 ✗

3000 > 2000	4000 > 2000	1000 > 2000 ✗
3000 > 1800	4000 > 1800	1000 > 1800 ✗
3000 > 3000 ✗	4000 > 3000	1000 > 3000 ✗

EID	ENAME	SAL	DEPTNO
5	JONES	4000	20

O/P of select clause

Q. WAQTD details of the employee who is hired before manager.

```
SELECT *

FROM EMP

WHERE HIREDATE < ALL (SELECT HIREDATE

                      FROM EMP

                      WHERE JOB = 'MANAGER');
```

1. **ANY OPERATOR: - -** It is a special operator which is used along with relation operator (>, <, >=, <=) to compare the values present at RHS.

- Any operator returns true, if any one of the condition are satisfied at RHS.

Q. WAQTD details of the employee who is earning more than the at least employee of the dept 10.

```
SELECT *

FROM EMP

WHERE SAL >ANY (SELECT SAL

                FROM EMP

                WHERE DEPTNO = 10);
```

Q. WAQTD details of the employee who is earning more than one of the employee of the dept 10.

```
            SELECT *

            FROM EMP

            WHERE SAL >ANY (SELECT SAL

                                       FROM EMP

                                       WHERE DEPTNO = 10);
```

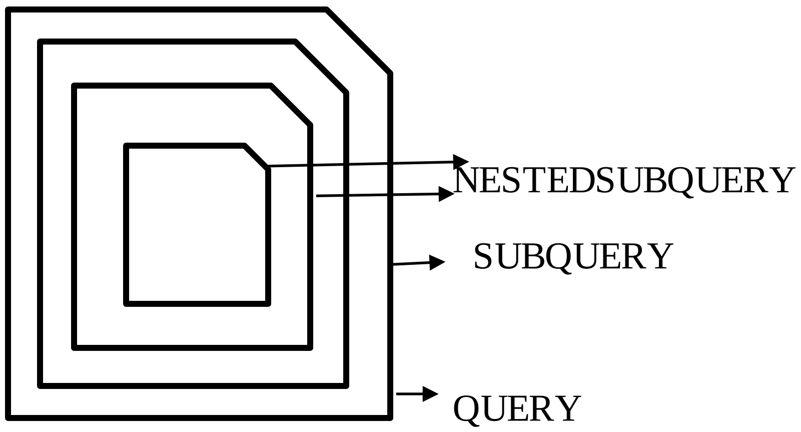
Q WAQTD name and salary for all the emp if they are earning less than one of the manager.

```
             SELECT ENAME, SAL

            FROM EMP

            WHERE SAL <ANY (SELECT SAL

                                       FROM EMP

                                       WHERE JOB = 'MANAGER');
```

NESTED SUBQUERY: -

"A subquery written inside another subquery is known as nested subquery."

Ex: -



- We can nest up to 255 subqueries

WHEN TO USE NESTED SUBQUERY: -

Q. WAQTD MAXIMUM SALARY OF THE EMP.

```
SELECT MAX(SAL) → 4000
FROM EMP;
```

SAL
2000
2500
1800
3000
4000
2000
1000
3000

Q. WAQTD 2ND MAXIMUM SALERY OF THE EMPLOYEE.

```
 SELECT MAX(SAL) → 3000
FROM EMP                 4000
WHERE SAL < (SELECT MAX(SAL)
             FROM EMP);
```

Q. WAQTD 4th MAXIMUM SALERY OF THE EMPLOYEE

```
        SELECT MAX(SAL)→2000
        FROM EMP               2500
   WHERE SAL < (SELECT MAX(SAL)
    FROM EMP                      3000
   WHERE SAL < (SELECT MAX(SAL)
                          FROM EMP                   4000
                          WHERE SAL < (SELECT MAX (SAL)
                                          FROM EMP)));
```

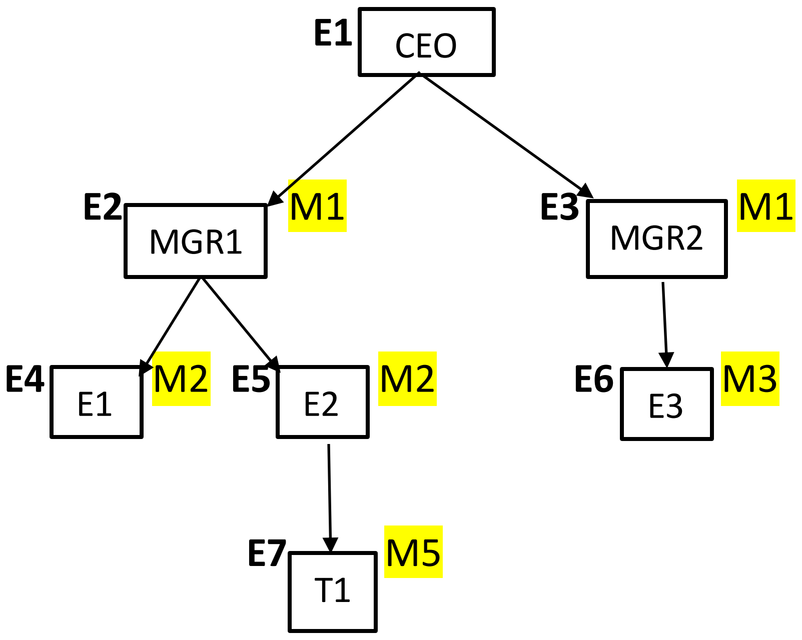
NOTE: -

- **If you are searching for max (<)**
- **If you are searching for min (>)**

<u>EMP – MGR RELATIONSHIP</u>: -

CASE 1: - To identify manager.

CASE 2: - To identify employees.



- ❖ **E = EMPLOYEE ID**
- ❖ **M = REPORTING MANAGER ID**

EMP

EID	**ENAME**	**SAL**	**MGR**
1	SMITH	2000	3
2	ALLEN	2500	4
3	WARD	1800	2
4	MILLER	3000	
5	SCOTT	1600	2

Q. WAQTD NAME OF SMITH'S MANAGER.

```
SELECT ENAME→WARD
FROM EMP                    3
WHERE EMPNO = (SELECT MGR
               FROM EMP
               WHERE ENAME = 'SMITH');
```

1 = 3 ✖

2 = 3 ✖

3 = 3

4 = 3 ✖

5 = 3 ✖

Q. WAQTD NAME OF ALLEN'S MANAGER.

```
SELECT ENAME
FROM EMP                    4
WHERE EMPNO = (SELECT MGR
               FROM EMP
                WHERE ENAME = 'ALLEN');
```

NOTE: -

Whenever identifying the manager : - mgr in subquery

Whenever identifying the employee : - empno in subquery

Q. WAQTD NAME OF JAMES MANAGER.

```
SELECT ENAME
FROM EMP
WHERE EMPNO = (SELECT MGR
               FROM EMP
               WHERE ENAME = ‘JAMES’);
```

CASE 2: - To identify the employee.

EMP

EMPNO	ENAME	SAL	MGR
1	SMITH	2000	3
2	ALLEN	2500	4
3	WARD	1800	2
4	MILLER	3000	
5	SCOTT	1600	2

Q. WAQTD details of the employee, who are reporting to ALLEN.

```
         SELECT ENAME → WARD, SCOTT

         FROM EMP                 2

         WHERE MGR IN (SELECT EMPNO

 3 = 2   2=2                    FROM EMP

4 = 2    2=2                          WHERE ENAME = 'ALLEN');
```

Q. WAQTD details of the employee who are reporting to blake.

```
        SELECT *

        FROM EMP

        WHERE MGR IN (SELECT EMPNO

                             FROM EMP

                             WHERE ENAME = 'BALKE');
```

Q. WAQTD NAME AND SAL OF THE EMPLOYEE WHO IS REPORTING TO THE KING.

```
        SELECT ENAME, SAL

        FROM EMP

        WHERE MGR IN (SELECT EMPNO

                             FROM EMP

                             WHERE ENAME = 'KING');
```

Q.WAQTD DNAME OF THE EMPLOYEE WHO ARE REPORTING TO JONES.

```
SELECT DNAME
FROM DEPT
WHERE DEPTNO IN (SELECT DEPTNO
                        FROM EMP
                   WHERE MGR IN (SELECT EMPNO
                                      FROM EMP
                                 WHERE ENAME = ‘JONES’))
```

Q. WAQTD details of the employee, who are reporting to ALLEN’S manager.

```
SELECT ENAME→ ALLEN
 FROM EMP                 4
WHERE MGR IN (SELECT EMPNO
                  FROM EMP        4
        WHERE EMPNO IN (SELECT MGR
                            FROM EMP
                          WHERE ENAME = ‘ALLEN’));
```

Q. WAQTD details of the employee, who are reporting to SMITH'S manager.

```
SELECT *
FROM EMP                    7902
WHERE MGR IN  (SELECT EMPNO
                FROM EMP              7902
              WHERE EMPNO IN  (SELECT MGR
                                FROM EMP
                                WHERE ENAME = 'SMITH'));
```

Q. WAQTD details of the employee, who are reporting to SMITH'S manager's manager.

```
SELECT *
FROM EMP
WHERE MGR IN  (SELECT EMPNO
                FROM EMP
              WHERE EMPNO IN  (SELECT MGR
                                FROM EMP
               WHERE EMPNO IN (SELECT MGR
                                FROM EMP
               WHERE ENAME = 'SMITH'));
```

JOINS: -

“The process of retrieving the data from the multiple table simultaneously is known as joins.”

TYPE OF JOIN: -

1. CARTESIAN JOIN / CROSS JOIN
2. INNER JOIN / EQUI JOIN
3. NATURAL JOIN
4. SELF JOIN
5. OUTER JOIN
   i. Left outer join
   ii. Right outer join
   iii. Full outer join

EMP

EMPNO	**ENAME**	**SAL**
-	-	-
-	-	-
-	-	-

DEPT

DEPTNO	**DNAME**	**LOC**
-	-	-
-	-	-
-	-	-

ENAME	**DNAME**
-	-
-	-
-	-

CARTESIAN JOIN/CROSS JOIN: -

"In Cartesian join a record of table 1 will merge with all the records of table2."

SNYTAX: -

ANSI

```
SELECT COLNAME/EXP
FROM Tablename1  CROSS JOIN Tablename2;
```

ORACLE

```
SELECT COLNAME/EXP
FROM Tablename1,  Tablename2;
```

EMP

EID	ENAME	DEPTNO
1	SMITH	20
2	ALLEN	10
3	WARD	30

DEPT

DEPTNO	DNAME	LOC
10	RESEACH	NW
20	OPERATION	DALLAS
30	SALES	BOSTON

Q. WAQTD details of the employee table and dept table.

```
SELECT *
FROM EMP, DEPT;
```

RESULT TABLE

EID	ENAME	DEPTNO	DEPTNO	DNAME	LOC
1	SMITH	20	10	RESEACH	NW
1	SMITH	20	20	OPERATION	DALLAS
1	SMITH	20	30	SALES	BOSTON
2	ALLEN	10	10	RESEACH	NW
2	ALLEN	10	20	OPERATION	DALLAS
2	ALLEN	10	30	SALES	BOSTON
3	WARD	30	10	RESEACH	NW
3	WARD	30	20	OPERATION	DALLAS
3	WARD	30	30	SALES	BOSTON

NOTES: -

- The no. of column present in result table will be equal to the summation of column present in both the table.
- The no. of records present in result table will be equal to the product of column present in both the table.

<u>INNER JOIN / EQUI JOIN</u>: -

"It is used to obtain only matching records."

SYNTAX: -

ANSI

```
SELECT COLNAME / EXP
FROM TABLE1 Inner join TABLE2
ON < JOIN_CONDITION >;
```

ORACLE

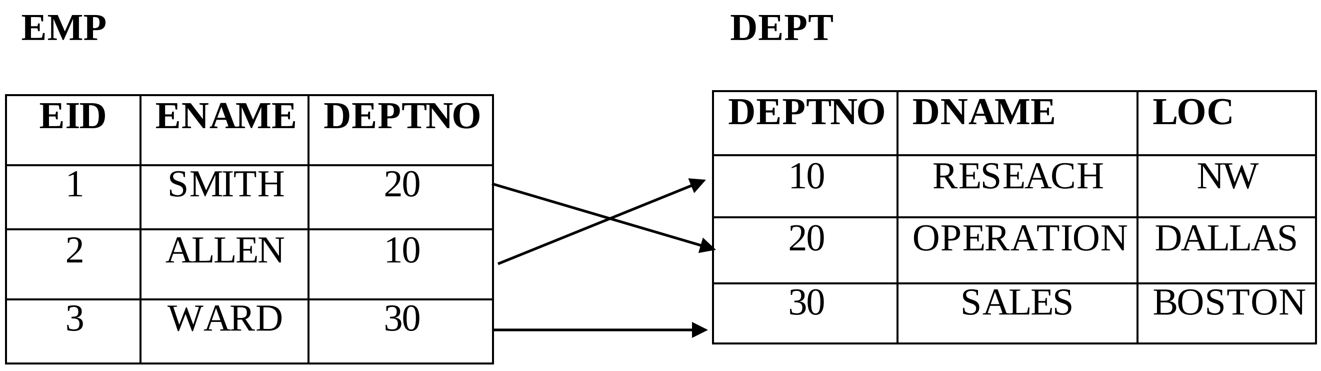
```
SELECT COLNAME / EXP
FROM TN 1, TN 2
WHERE < JOIN_CONDITION >;
```

JOIN CONDITION: -

"It is used to merge two tables."

SYNTAX: - TABLENAME1.COLNAME = TABLENAME2.COLNAME;

EX: - EMP.DEPTNO = DEPT.DEPTNO



EMP

EID	ENAME	DEPTNO
1	SMITH	20
2	ALLEN	10
3	WARD	30

DEPT

DEPTNO	DNAME	LOC
10	RESEACH	NW
20	OPERATION	DALLAS
30	SALES	BOSTON

Q. WAQTD details of the employee table and dept table.

```
SELECT *
FROM EMP, DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO;
```

20 = 10

20 = 20

20 = 30

10 = 10

10 = 20

10 = 30

30 = 10

30 = 20

30 = 30

RESULT TABLE

EID	ENAME	DEPTNO	DEPTNO	DNAME	LOC
1	SMITH	20	20	OPERATION	DALLAS
2	ALLEN	10	10	RESEARCH	NW
3	WARD	30	30	SALES	BOSTON

Q. WAQTD NAME OF THE EMPLOYEE AND THEIR DEPT NAME.

```
SELECT ENAME, DNAME
FROM EMP, DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO;
```

Q. WAQTD ENAME AND HIS SALRY ALONG WITH LOC, WHERE EMPLOYEE'S ARE EARNING MORE 2000.

```
        SELECT ENAME, SAL, LOC
        FROM EMP, DEPT
        WHERE EMP.DEPTNO = DEPT.DEPTNO AND SAL > 2000;
```

Q. WAQTD DNAME AND HIREDATED WHERE EMP ARE WORKING IN DEPT RESEACH & HIRED BEFORE 1986.

```
      SELECT DNAME, HIREDATE
      FROM EMP, DEPT
    WHERE EMP.DEPTNO = DEPT.DEPTNO AND DNAME = 'RESEARCH'
                              AND HIREDATE < '01-JAN-86';
```

Q. WAQTD ENAME & DNAME & DEPTNO WHERE EMPLOYEE ARE WORKING AS MANAGER.

```
     SELECT ENAME, DNAME, EMP.DEPTNO, DEPT.DEPTNO
     FROM EMP, DEPT
     WHERE EMP.DEPTNO = DEPT.DEPTNO AND JOB = 'MANAGER';
```

Q. WAQTD ENAME AND HIS SALARY ALONG WITH DEPT DETAILS WHERE EMPLOYEE ARE WORKING IN DEPTNO 10.

```
    SELECT ENAME, SAL, DEPT.*
    FROM EMP, DEPT
    WHERE EMP.DEPTNO = DEPT.DEPTNO AND EMP.DEPTNO = 10;
```

NATURAL JOIN: - **"**It has 2 behaviours."

1. If there is a connection between 2 tables than it will act like INNER JOIN/EQUI JOIN.
2. If there is a no connection between 2 tables than it will act like CROSS JOIN/ CARTESIAN JOIN.

SYNTAX: -

ANSI: -

```
SELECT COLNAME/ EXP
FROM TABLENAME1 Natural join TABLENAME2;
```

EID	ENAME	DEPTNO
1	SMITH	20
2	ALLEN	10
3	WARD	30

DEPTNO	DNAME	LOC
10	RESEARCH	NEWYORK
20	OPERATION	DALLAS
30	SALES	BOSTON

PID	PNAME	PRICE
P1	DELL	40000
P2	HP	30000

Q. WAQTD details of emp table and dept table.

```
SELECT *

FROM EMP NATURAL JOIN DEPT;
```

RESULT TABLE

DEPTNO	EID	ENAME	DNAME	LOC
20	1	SMITH	OPERATION	DALLAS
10	2	ALLEN	RESEARCH	NW
30	3	WARD	SALES	BOSTON

Act as a inner join

Q. WAQTD details of emp table and product table.

```
SELECT *

FROM EMP NATURAL JOIN PRODUCT;
```

RESULT TABLE

EID	ENAME	DEPTNO	PID	PNAME	PRICE
1	SMITH	20	P1	DELL	40000
1	SMITH	20	P2	HP	30000
2	ALLEN	10	P1	DELL	40000
2	ALLEN	10	P2	HP	30000
3	WARD	30	P1	DELL	40000
3	WARD	30	P2	HP	30000

Act as a Cartesian

OUTER JOIN: -

"It is used to obtain unmatched records."

Types of outer join: -

1. Left outer join
2. Right outer join
3. Full outer join

1. LEFT OUTER JOIN: -

"It is used to obtain unmatched records only from left table, along with matching records."

SYNTAX: -

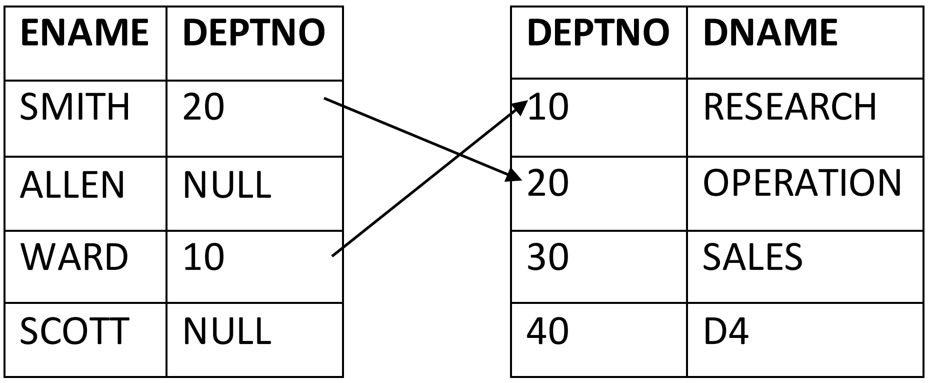
ASNI

```
SELECT COLNAME/ EXP
FROM TN1 left [outer] join TN2
ON < join_condition >;
```

ORACLE

```
SELECT COLNAME/ EXP
FROM TN1, TN2
WHERE TN1.COLNAME = TN2.COLNAME(+);
```

EX: -



ENAME	DEPTNO
SMITH	20
ALLEN	NULL
WARD	10
SCOTT	NULL

DEPTNO	DNAME
10	RESEARCH
20	OPERATION
30	SALES
40	D4

Q. WAQTD details of the emp and dept table.

```
SELECT *
FROM EMP, DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO (+);
```

RESULT TABLE

ENAME	DEPTNO	DEPTNO	DNAME
SMITH	20	20	OPERATION
WARD	10	10	RESEARCH
ALLEN	NULL		
SCOTT	NULL		

RIGHT OUTER JOIN: -

"It is used to obtain unmatched records only from Right table, along with matching records."

SYNTAX: -

ASNI

```
SELECT COLNAME/ EXP
FROM TN1 Right [outer] join TN2
ON < join_condition >;
```

ORACLE

```
SELECT COLNAME/ EXP
FROM TN1, TN2
WHERE TN1.COLNAME(+)  = TN2.COLNAME;
```

EX: -

ENAME	DEPTNO
SMITH	20
ALLEN	NULL
WARD	10
SCOTT	NULL

LEFT TABLE

DEPTNO	DNAME
10	RESEARCH
20	OPERATION
30	SALES
40	ACCOUNTING

RIGHT TABLE

Q. WAQTD details of the emp and dept table.

```
SELECT *
FROM EMP, DEPT
WHERE EMP.DEPTNO(+)  = DEPT.DEPTNO;
```

RESULT TABLE

ENAME	DEPTNO	DEPTNO	DNAME
SMITH	20	20	OPERATION
WARD	10	10	RESEARCH
		30	SALES
		40	ACCOUNTING

FULL OUTER JOIN: -

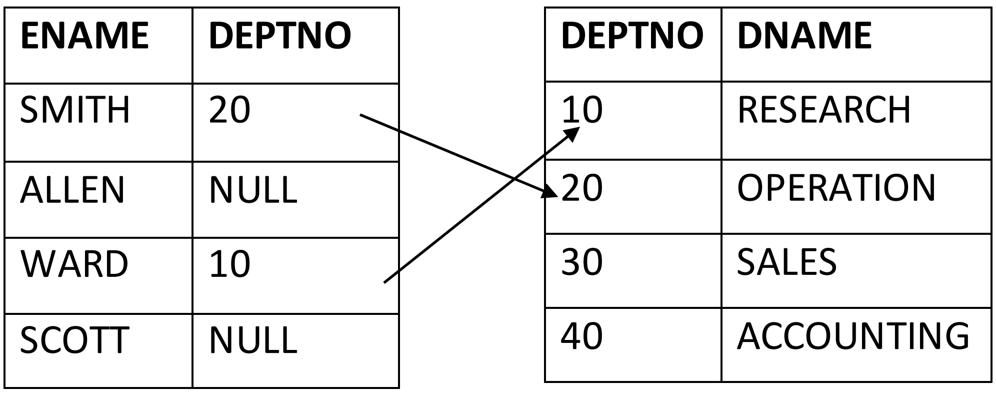
"It is used to obtain unmatched records from both the table, along with matching records."

SYNTAX: -

ASNI

```
SELECT COLNAME/ EXP
FROM TN1 Full [outer] join TN2
ON < join_condition >;
```

EX: -



ENAME	**DEPTNO**
SMITH	20
ALLEN	NULL
WARD	10
SCOTT	NULL

DEPTNO	**DNAME**
10	RESEARCH
20	OPERATION
30	SALES
40	ACCOUNTING

Q. WAQTD details of the emp and dept table.

```
SELECT *
FROM EMP full join DEPT
ON EMP.DEPTNO = DEPT.DEPTNO;
```

RESULT TABLE

ENAME	DEPTNO	DEPTNO	DNAME
SMITH	20	20	OPERATION
WARD	10	10	RESEARCH
ALLEN	NULL		
SCOTT	NULL		
		30	SALES
		40	ACCOUNTING

<u>SELF JOIN</u>: -

"Joining a table by itself."

"Joining same 2 tables is known self join."

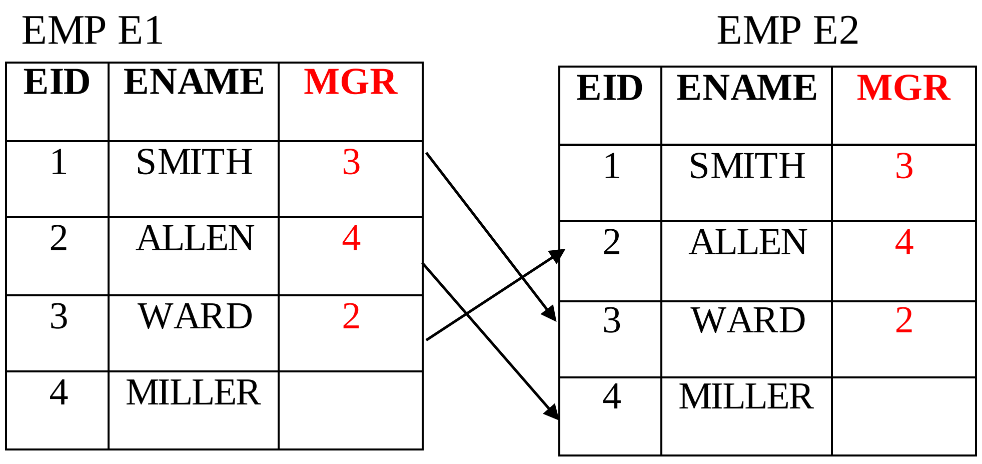
SYNTAX: -

ANSI

```
SELECT COLNAME/EXP
FROM TN1 SELF JOIN TN2
ON < JOIN_CONDITION >;
```

ORACLE

```
SELECT COLNAME/EXP
FROM TN1, TN2
WHERE < JOIN_CONDITION >;
```



EMP E1

EID	ENAME	MGR
1	SMITH	3
2	ALLEN	4
3	WARD	2
4	MILLER	

EMP E2

EID	ENAME	MGR
1	SMITH	3
2	ALLEN	4
3	WARD	2
4	MILLER	

Q. WAQTD details of the employee and their manager.

```
SELECT *
FROM EMP E1, EMP E2
WHERE E1.MGR = E2.EMPNO;
```

NOTE: -

E1: - EMPLOYEE TABLE

E2: - MANAGER TABLE

WHERE E1.MGR = E2.EMPNO

3 = 1	4 = 1	2 = 1
3= 2	4= 2	2 = 2
3= 3	4 = 3	2 = 3
3= 4	4 = 4	2 = 4

E1.EID	**E1.ENAME**	**E1.MGR**	**E2.EID**	**E2.ENAME**	**E2.MGR**
1	SMITH	3	3	WARD	2
2	ALLEN	4	4	MILLER	
3	WARD	2	2	SMITH	3

Q. WAQTD EMPLOYEE NAME AND HIS SALARY ALONG WITH MANAGER NAME AND HIS SALARY.

```
SELECT E1.ENAME, E1.SAL, E2.ENAME,E2.SAL

FROM EMP E1, EMP E2

WHERE E1.MGR = E2.EMPNO;
```

Q. WAQTD ENAME ALONG WITH MANAGER NAME AND THEIR DEPTNO WHERE EMPLOYEE IS WORKING IN DEPTNO 20.

```
SELECT E1.ENAME,E1.DEPTNO,E2.ENAME,E2.DEPTNO

FROM EMP E1, EMP E2

WHERE E1.MGR = E2.EMPNO AND E1.DEPTNO = 20;
```

Q. WAQTD NAME OF THE EMPLOYEE AND HIS MANAGER ALONG WITH THEIR SALARIES, WHERE EMPLOYEE'S ARE EARNING WHERE EARNING LESS THAN MANAGER SALARY.

```
SELECT E1.ENAME, E1.SAL, E2.ENAME, E2.SAL

FROM EMP E1, EMP E2

WHERE E1.MGR = E2.EMPNO AND E1.SAL < E2.SAL;
```

MULTIPLE CONDITION ON JOINS: -

Q. WAQTD EMPLOYEE NAME AND HIS MANAGER NAME

```
        SELECT E1.ENAME AS EMPLO, E2.ENAME AS MGR
        FROM EMP E1,EMP E2
        WHERE E1.MGR = E2.EMPNO;
```

Q. WAQTD ENAME & HIS MANAGER NAME & MANAGER'S MANAGER NAME.

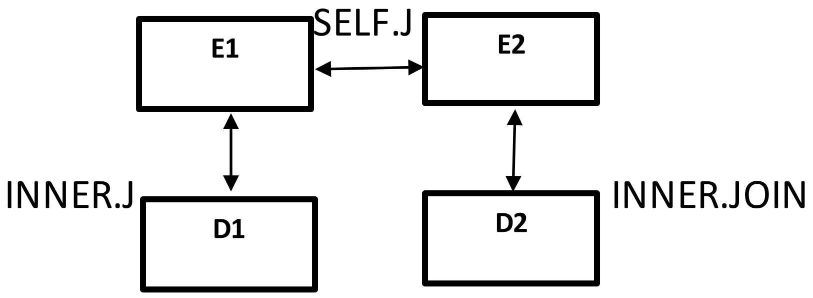
E1→EMPLOYEE

E2→MANAGER

E3→MANAGER'S MANAGER

```
SELECT E1.ENAME, E2.ENAME,E3.ENAME
FROM EMP E1, EMP E2, EMP E3
WHERE E1.MGR = E2.EMPNO AND
            E2.MGR = E3.EMPNO;
```

Q. WAQTD EMPLOYEE NAME & HIS DNAME WITH MGR'S NAME & HIS DNAME.



```
SELECT E1.ENAME, D1.DNAME,E2.ENAME,D2.DNAME
FROM EMP E1, DEPT D1, EMP E2, DEPT D2
WHERE E1.MGR = E2. EMPNO AND
        E1.DEPTNO = D1.DEPTNO AND
        E2.DEPTNO = D2.DEPTNO;
```

Q. WAQTD EMP NAME & HIS SALARY & MANAGER’S NAME AND HIS SALARY AND MANAGER’S MANAGER NAME AND HIS SALARY,

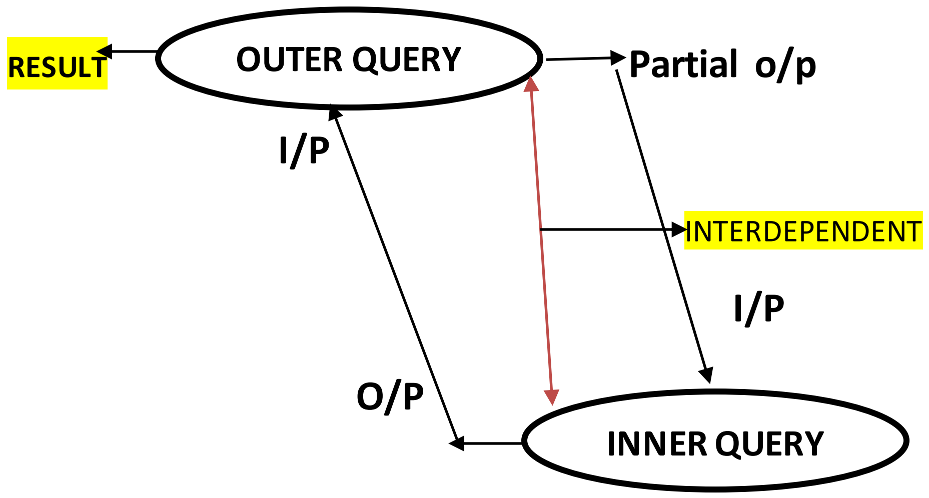
WHERE MANAGER MANAGER IS EANING MORE THAN EMPLOYEE.

```
SELECT E1.ENAME,E1.SAL,E2.ENAME,E2.SAL,E3.ENAME,E3.SAL
FROM EMP E1, EMP E2, EMP E3
WHERE E1.MGR = E2. EMPNO AND
        E2.MGR = E3.EMPNO AND
        E3.SAL > E1.SAL;
```

CO-RELATED SUBQUERY: -

"A QUERY WRITTEN INSIDE ANOTHER QUERY WHERE INNER AND OUTER QUERY BOTH ARE DEPENDING ON EACH OTHER (INTERDEPENDENT)."

WORKING PRINCIPLE: -



WORKING PRINCIPLE OF CO-RELATED SUBQUERY: -

- Let us consider 2 queries outer query and inner query
- Outer query will execute first but partially.
- The partial output of outer query will be given to inner query as an input.
- Inner query executes completely and generate an output.
- The output of inner query fed as an input to the outer query and it will produce RESULT.
- Therefore, we can state that the outer query and inner query both are dependent each other.

NOTE: -

- In co-related subquery, join condition is must and must be written in inner query.

- Co-related subquery works with the principle of both the subquery and join.

TO IDENTIFY NTH MAX SALARY: -

SYNTAX: -

```
SELECT E1.SAL
FROM EMP E1
 WHERE N-1 IN (SELECT COUNT (DISTINCT E2. SAL)
                       FROM EMP E2
                        WHERE E1.SAL < E2.SAL);
```

EMP E1

E1.SAL	
2000	$\rightarrow$ 4TH
3000	
1800	
4000	
3500	
2000	
1000	

EMP E2

E2.SAL
2000
3000
1800
4000
3500
2000
1000

Q. WAQTD 4th MAXIMUM SALARY.

```
SELECT E1.SAL  → 2000                    3
FROM EMP E1                        (3000,4000,3500)
 WHERE 3 IN (SELECT COUNT (DISTINCT E2. SAL)
                       FROM EMP E2
  3 = 3                 WHERE E1.SAL < E2.SAL);

  3 = 2

  3 = 4
```

COUNT- 3	COUNT – 2	COUNT – 5	COUNT – 0	COUNT - 1
2000 < 2000	3000 < 2000	1800 < 2000	4000 < 2000	3500 < 2000
2000 < 3000	3000 < 3000	1800 < 3000	4000 < 3000	3500 < 3000
2000 < 1800	3000 < 1800	1800 < 1800	4000 < 1800	3500 < 1800
2000 < 4000	3000 < 4000	1800 < 4000	4000 < 4000	3500 < 4000
2000 < 3500	3000 < 3500	1800 < 3500	4000 < 3500	3500 < 3500
2000 < 2000	3000 < 2000	1800 < 2000	4000 < 2000	3500 < 2000
2000 < 1000	3000 < 1000	1800 < 1000	4000 < 1000	3500 < 1000

Q. WAQTD 6th MAXIMUM SALARY.

```
SELECT SAL → 1800
FROM EMP E1
WHERE  5 IN (SELECT COUNT (DISTINCT SAL)
                   FROM EMP E2
                   WHERE E1.SAL <  E2.SAL) ;
```

DATA DEFINATION LANGUAGE: -

"DDL is used to construct an object in the database and deals with the structure of table/entity/object."

We have 5 statement: -

1. Create
2. Rename
3. Alter
4. Truncate
5. Drop

1. CREATE: -

"It is used to create our own table."

SYNTAX: -

```
        CREATE TABLE TABLENAME
         (
            COLUMNAME1 DATATYPE CONSTRAINT,
            COLUMNAME2 DATATYPE CONSTRAINT,
            COLUMNAME3 DATATYPE CONSTRAINT,
                              -
                              -
           COLUMNAME Nth DATATYPE CONSTRAINT
        );
```

Ex: -

```
        CREATE TABLE STUDENT
(
SID NUMBER (5) PRIMARY KEY,
SNAME VARCHAR (15) NOT NULL,
GENDER VARCHAR (8)   NOT NULL
);
```

RENAME: -

"It is used to change the name of the existing table."

SYNTAX: -

```
        RENAME Existing tablename to New tablename;
```

EX: -

```
 RENAME STUDENT TO STUD1;
```

ALTER: -

"It is used to modify the structure of table."

Such as: -

i. Add a column
ii. To remove the column
iii. To rename a column name
iv. To modify the datatype
v. To modify the constraint
vi. To assign a foreign key

ADD A COLUMN: -

SYNTAX: -

```
ALTER TABLE TABLENAME
ADD COLUMN NAME DATATYPE CONSTRAINT;
```

EX: -

```
ALTER TABLE STUD1
ADD EMAIL VARCHAR(15) UNIQUE NOT NULL;
```

TO REMOVE A COLUMN NAME: -

SYNTAX: -

```
ALTER TABLE TABLENAME
DROP COLUMN COLUMN NAME;
```

EX: -

```
ALTER TABLE STUD1
DROP COLUMN GENDER;
```

<u>TO RENAME A COLUMN NAME: -</u>

SYNTAX: -

```
ALTER TABLE TABLENAME
RENAME COLUMN EXISTING_COLNAME TO NEW_COLNAME;
```

EX: -

```
ALTER TABLE STUD1
RENAME COLUMN SNAME TO NAME;
```

<u>TO MODIFY THE DATATYPE: -</u>

SYNTAX: -

```
ALTER TABLE TABLENAME
MODIFY COLUMN NAME NEW_DATATYPE;
```

EX: -

```
ALTER TABLE STUD1
MODIFY GENDER CHAR (9);
```

TO MODIFY THE CONSTRAINTS: -

SYNTAX: -

```
ALTER TABLE TABLENAME
MODIFY COLUMN NAME EXISTING_DATATYPE NEW_CONSTRAINT;
```

EX: -

```
ALTER TABLE STUD1
MODIFY GENDER CHAR (9) UNIQUE NOT NULL;
```

TO ADD CONSTRAINTS: -

SYNTAX: -

```
ALTER TABLE TABLENAME
ADD CONSTRAINT CONSTRAINT_REFERENCE_NAME
CONSTRAINT_TYPE (COLUMN);
```

EX: -

```
ALTER TABLE STUD1
ADD CONSTRAINT UNI_PH UNIQUE (PH_NO);
```

TO ASSIGN A FOREIGN KEY: -

For ex: - STUD1, TEACHER

SID	SNAME	SUBID	TID
1	TINKI	100	
2	RINKI	200	
3	PINKI	300	

↔

TID	TNAME	SUB
101	VANDNA	SQL
102	LAVANYA	JAVA
103	DIVYA	MT

STEP 1: -

```
ALTER TABLE TABLENAME
ADD COLUMN DATATYPE;
```

STEP 2: -

```
ALTER TABLE TABLENAME
ADD CONTRAINT CON_REFNAME FOREIGN KEY (COLUMN)
REFERENCES PARENT_TABLE_NAME (COLUMN);
```

EXAMPLE: -

STEP 1: -

```
ALTER TABLE STUD1
ADD TID NUMBER (5);
```

STEP 2: -

```
ALTER TABLE STUD1
ADD CONSTRAINT FK_TID FOREIGN KEY(TID)
REFERENCES TEACHER (TID);
```

- **WITHOUT USING ALTER, DURING CREATION OF TABLE MAKE CONNECTION BETWEEN TABLE.**

CID	CNAME	PID
1	TINKI	
2	RINKI	
3	PINKI	

PID	PNAME	PRICE
101	VANDNA	SQL
102	LAVANYA	JAVA
103	DIVYA	MT

```
CREATE TABLE PRODUCT
(
PID NUMBER (5) PRIMARY KEY,
PNAME VARCHAR (15) NOT NULL
);
```

TABLE CREATED.

```
CREATE TABLE CUSTOMER
(
CID NUMBER (5) PRIMARY KEY,
CNAME VARCHAR (25) NOT NULL,
PID NUMBER (5),
CONSTRAINT FK_PID FOREIGN KEY (PID) REFERENCES
PRODUCT (PID)
);
```

TRUNCATE: -

"It is used to remove all the records / data from the table permanently."

SYNTAX: -

```
TRUNCATE TABLE TABLE_NAME;
```

TID	TNAME	SUB
101	VANDNA	SQL
102	LAVANYA	JAVA
103	DIVYA	MT

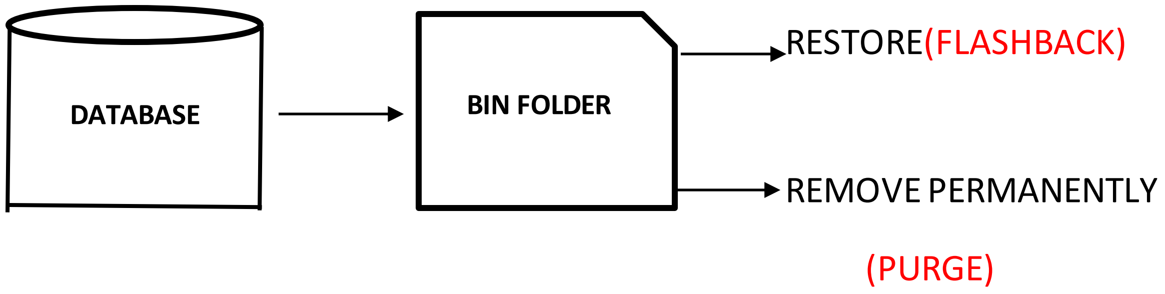
EX: - TRUNCATE TABLE PRODUCT;

DROP: -

"It is used to remove the table from the database."

SYNTAX: - DROP TABLE TABLE_NAME;

EX: - DROP TABLE PRODUCT;



FLASHBACK: - "It is used to restore the table from the recycle bin."

SYNTAX: - FLASHBACK TABLE TABLE_NAME

TO BEFORE DROP;

PURGE: -

"It is used to remove the table permanently from the recycle bin."

SYNTAX: - PURGE TABLE TABLE_NAME;