22/03/24.

SQL: Structure Query Language.

1) It is used to communicate with relational database.

2) In the year of 1970, SQL was Invented by IBM.

3) Later in the year 1976, purchased by Oracle corporation and they released SQL-Version 2.

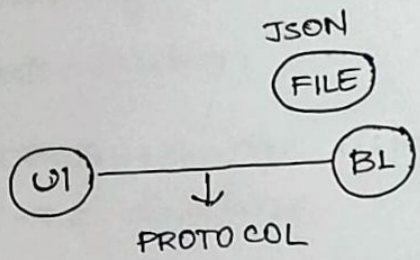
4) By using SQL, we can perform 4 operations:

(i) Create the data.

(ii) Read the data.

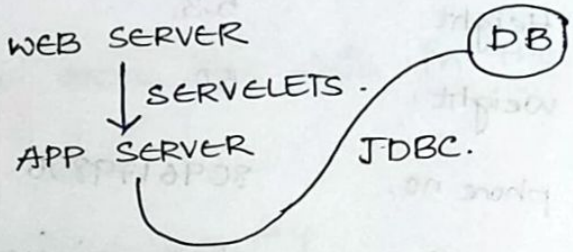
(iii) Update the data.

(iv) Delete the data.



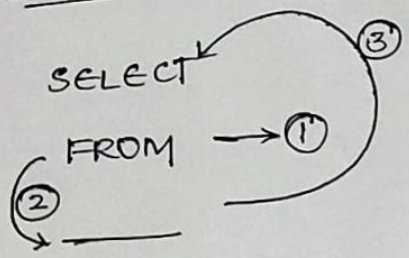
HTTPS

FTP.



⇒ EXECUTION

→ Based up on CLAUSES.



23/08/24

DATA:

Data is a Raw Fact of an Object - and it will describe the properties / attributes of an object.

→ properties is also known as "Attributes".

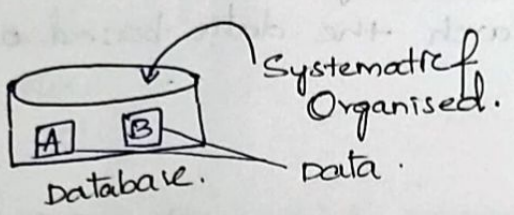
→ Object is also known as "Entity".

Eg:

Properties/Attributes:	Values:
Name:	Srinivas
Age:	23
Gender:	Male
Qualification:	B.Tech
DOB	20.12.2000
Height:	5.3.
Weight:	50
phone no.	8096179856
G-mail.	chennasrinivas44@gmail.com
Blood group:	O+
Address:	Jagital, Karimnagar.
Father Name:	Shyam
Mother Name:	Subchana.

→ Database:

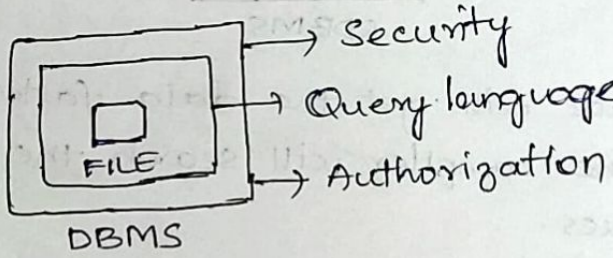
Database is a place where we can store the data in systematic and organised manner.



- There are two kinds of database are available.
  1. Relational Database.
  2. Non-Relational Database.

→ DBMS:

- Database Management System is a software used to maintain and manage the database.
- In DBMS, we are having two features:
  (i) Security.
  (ii) Authorization.
- In DBMS, we are using Query Language to communicate with the database.
- In DBMS we are going to store the data in the form of "FILE-FORMAT"



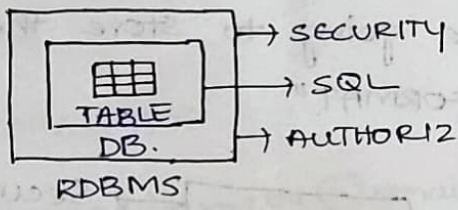
25/08/24

→ In DBMS, we are having two disadvantages:

1. Memory wastage.
2. It will take more time to display the data.
   Because the compiler will search the data based on Binary values.

RDBMS:

- It stands for Relational Database management system and it is used to maintain and manage database
- RDBMS providing two features:
  1.
  2.
- In RDBMS, we are storing the data in the form of Table Format.
- In RDBMS, we are using structured Query language to communicate with database



- In RDBMS, we will get the data faster than the DBMS.
  Because the compiler will search the data based on ASCII values.
- ASCII – American Standard Code for Information Interchange.
- ANSI – American National Standard Institute.

TABLE:

The combination of rows and columns is known as Table.

The table consists of:

1. Column
2. Row.
3. Cell

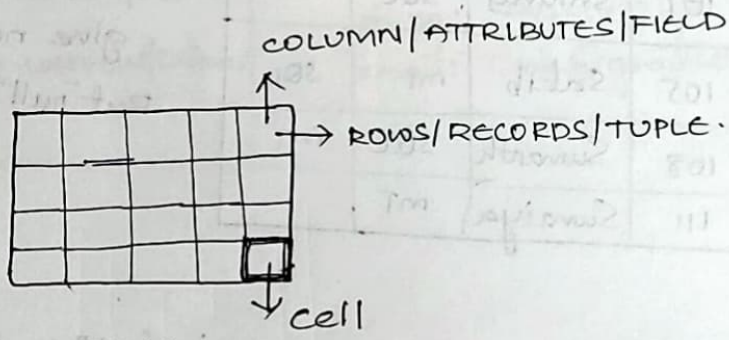
Row: The horizontal portion of the table is known as Row.

Row is also known as "RECORD", "TUPLE".

Column: The vertical portion of the table is known as column.

column is also known as "ATTRIBUTES", "FIELDS", and "PROPERTIES".

Cell: The intersection of a row and column is known as cell. cell is the smallest part of the table.



TABLE

26/03/24.

→ EDGER FRANK.

Rules of E.F. CODD: → Invented Relational Model.

Rule 1: Whenever we are inserting the data into the cell. It must be a single value.

SID	SNAME	SUBJECT
101	Srinivas	SQL, Java
105	Sakib	M.T, SQL
108	Sumanth	SQL

→ Rule no.1 is not satisfied.

SQL → MT

SQL, Java

UPDATING DATA -

MT → cell (SQL, JAVA)

↓

new value

(MT)

∴ Reason!

while updation data loss will be there.

SID	SNAME	SUB1	SUB2
101	Srinivas	SQL	JAVA
105	Sakib	MT	SQL
108	Sumanth	SQL	MT
111	Sumaiya	MT	

→ Rule 1 is satisfied.

→ while storing we should give null value.

But "null" is not displayed.

Rule 2: In RDBMS we are storing the data in multiple tables, if it is required we can establish the relation b/w the table by using key attributes.

KEY ATTRIBUTES

STUDENT

SID	SNAME	SUB ID
101	Srinivas	S1
102	Sumanth	S1
105	Arjun	S3
420	Sumaiya	S2
450	Naitik	S4

SUBJECT

SUB ID	SUB NAME
S1	Java
S2	SQL
S3	M.T.
S4	Python

Rule 3: In RDBMS, we are storing the data along with 'META DATA'.

META-DATA: The data displaying about the data is known as META-DATA.

SID	SNAME	LINKEDIN-URL
101	SHARUK	www.LINKEDIN/sharuk.
102	SALMAN	www.LINKEDIN/salman

LINKEDIN

Name: Shaauk khan
ph.no: 9876543210
email: ssk@gmail.com

META DATA.

RULE 4:

Whenever we are inserting the data into the table, the data we have to validate and verify.

To validate and verify the data we are having 2 ways.

(i) By Assigning datatypes
(ii) By Assigning constraints.

⇒ DATATYPES:

It determines or specify which type of data we are going to store in the particular memory location. We are having 5 types of datatypes.

(i). CHAR.
(ii) VARCHAR
(iii) DATE.
(iv) NUMBER.
(v) LARGE OBJECT.

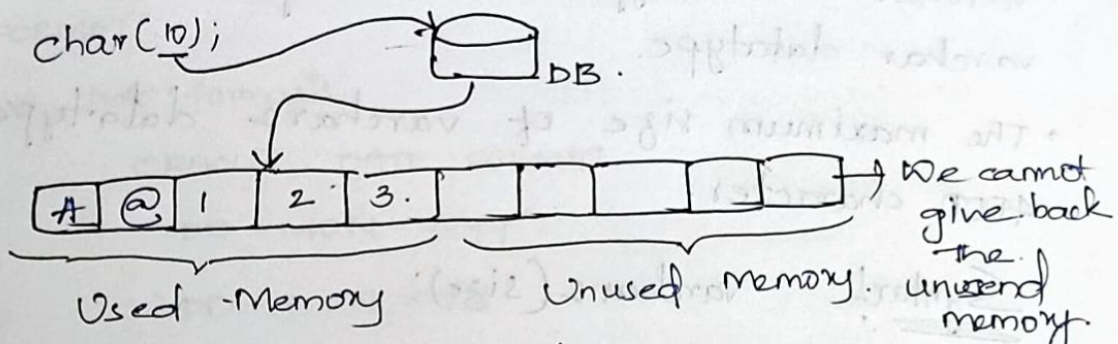
(i) CHAR:

- It can accept alphabets (A-Z, a-z), number (0-9). special characters (₹, *, @) and Alpha numeric
- whenever we are using char. datatype we have to enclosed the character within the single quotes.
- whenever we are using the char datatype, we have to mention the size.
- If we didn't mentioned the size by defaultly it will consider as 1.
- The maximum size of char. datatype, is 2000 characters.

SYNTAX:

CHAR(SIZE);

Ex: char(10);



- Char. datatype will focus fixed length memory allocation

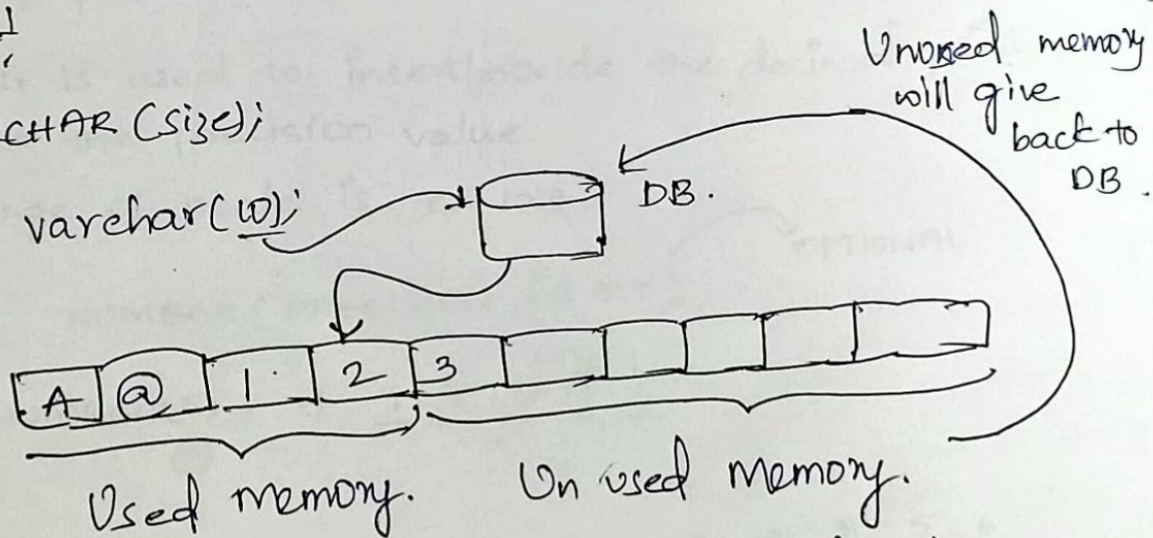
VARCHAR

- It can accepts alphabets (a-3, A-Z) number (0-9), special characters ($, @, #) and alpha numberic.
- whenever we are using the varchar datatype it should be enclosed within the single quotes.
- whenever we are using the varchar datatypes the size is mandatory.
- The maxium size of the varchar datatype is 2000 character.

Syntax:

VARCHAR(size);

Ex: varchar(10);



- Varchar data type will follows varrable length memory allocation

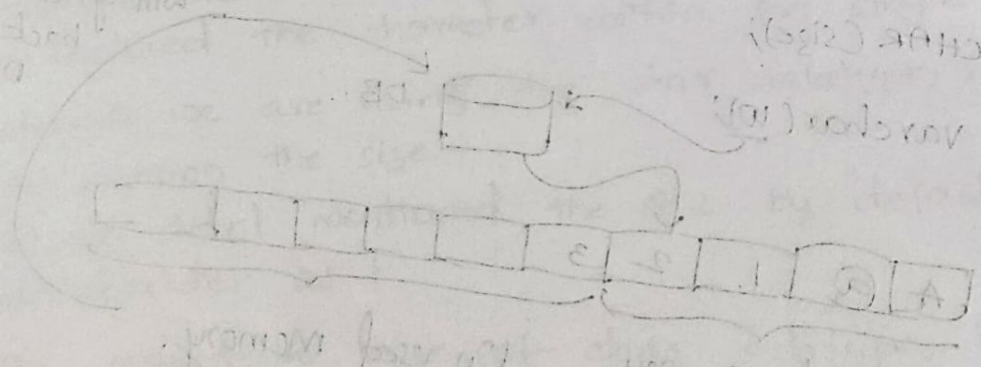
VARCHAR 2:

- varchar2 datatype is the updated version of varchar datatype.
- The maximum size of varchar2 datatype is 4000 character.

Syntax: varchar2(size);

NOTE:

Wherever we are using varchar datatype by default it will consider as varchar2 datatypes.



29/03/24

DATE: It is used to store only the date in "ORACLE-FORMAT".

There are two formats:

ORACLE DATE FORMAT

'DD-MON-YYYY'.

'DD-MON-YY'

Syntax: DATE;

Ex: '14-FEB-2025'.

'01-APR-20'.

NUMBER: It is used to store numeric values.

'NUMBER' data type is having two arguments.

1) Precision and 2) Scale.

PRECISION: It is used to insert the number.

Range of precision is 1-38.

SCALE: It is used to insert/provide the decimal point to the precision value

The range of scale is 0-127.

Syntax: NUMBER(PRECISION, [SCALE]); (SCALE: OPTIONAL.)

Ex: NUMBER(5); ⇒ 1 7 0 9 1 (P)

NUMBER(10, 6); ⇒ 8 0 9 6 . 1 7 9 8 5 6 (P > S; start counting...)

NUMBER(4,8) : ±. _ _ _ _ 1 7 2 0

(P) < (S)

NUMBER(5,5) : ±. 1 2 3 4 5

(P) = (S)

Steps:

1). Identify. precision, scale.

2) Identify the greatest value.

3). Blocks will be created.

4) Precision will be inserted.

5). Decimal point will be inserted

LARGE-OBJECT

It is used to store huge amount of data.

In Large Object we are having two types:

1. Character large object.
2. Binary. large object.

- CHARACTER LARGE OBJECT:

It is used to store huge amount of characters upto 4GB.

Syntax: CLOB;

Ex: SOCIAL MEDIA APPLICATIONS

- BINARY LARGE OBJECT:

It is used to store huge amount of binary values/characters. upto 4GB.

Syntax: BLOB;

Ex: PHOTOS, VIDEOS, MP3, AUDIO, _____

CONSTRAINTS: These are the rules used to verify the data we are having. 5 types of constraints.

1. Unique.
2. Not Null.
3. Check.
4. Primary key.
5. Foreign key.

- UNIQUE:
It is a constraint cannot accept repeated or duplicate values.

- NOT NULL:
It is a constraint cannot accept null value.

- CHECK:
It is a extra validation assigned to the column. along with the condition. If the condition is satisfied. It can accept the value. otherwise. rejects the value.

- PRIMARY-KEY:
It is a constraint used to identify the record uniquely from the table.

⇒ characteristics of primary key:

1) It cannot accept repeated or duplicated values.
2) It cannot accept null values.
3) It is the combination of: unique and not null constraints.
4) One table can accept only one column as primary key
5) Primary key is optional, but highly recommended

o FOREIGN-KEY

It is a constraint used to establish the relation b/w the tables.

⇒ CHARACTERISTICS:

1) It can accept repeated or duplicate values.
2) It can accept null values.
3) It is not a combination of unique and not null constraints.
4) One table can accept more than one column as foreign key.
5) If column wants to become foreign key, it must be primary key in its own table.
6) Foreign key is present in child table, but it belongs to parent table.
7) "F.K" is also known as referential integrity constraint "R.I.C".

check(AGE<=18) check(Length(mobile)=10)

(PK) NN UN | NN | NN | NN UN | (FK)

SID	SNAME	AGE	MOBILE	SUBID
char(3)	varchar(15)	Number(2)	Number(10)	varchar(5)
101	Sumanth	18	9876543210	S1
102	Srinivas	22	9754 63210	S2
103	Sumanth	24	9854 32010	S2

child

(PK) NN UN | NN UN

SUBID	SUBNAME
varchar(5)	varchar(10)
S1	SQL
S2	MT

Parent ←

STATEMENTS IN SQL:

We are having 5 types of statements in SQL.

1. Data Definition Language (DDL) - CREATE.
2. Data manipulation Language. (DML) - INSERT
3. Transcation control Language (TCL) -
4. Data control language (DCL) -
5. Data Query Language (DQL) - READ.

• DATA QUERY LANGUAGE:

It is used to read the data from database or table.

In DQL, we are having four statements.

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

→ SELECT:

It is used to select the data and display.

→ SELECTION:

It is used to display/select the data by selecting column as well as row.

→ PROJECTION:

It is used to read the data by selecting only the colu-mns.

→ JOINS

It is used to read the data from multiple tables simultaneously..

*PROJECTION:

It is used to read the data by selecting only the column

Syntax: SELECT */[DISTINCT] COL_NAME/EXPRESSION [ALIAS]
FROM TABLE NAME;

1. FROM CLASS:
2. SELECT CLASS:

1) FROM CLASS will execute first, it will moves to the data-base and search for the table. If the table is present, it will select the table and put under execution

2) Now, SELECT CLASS will execute. It will moves to the table, and search for the column or expression, if it is present it will select and display.

NOTE:

(1). SQL Queries will execute based on clauses.

(2). If in any one of the clause contains an error then next clause will not execute.

(3). ";" - To end the Query.
if we use ";" alone-ly, then it will execute previous query once again.

To Give the Comments:

1) "--" is used to give single line comment.

2) "/*
COMMENTS - is used for multi-line comments.
*/"

(*) "ed" - To edit the Query, we have to use "ed" key-word.

We can use "ed" (or) "edit"

→ NOTE:

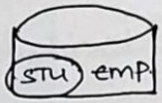
After editing the query in the notepad by using ed; command to execute the query. we should use "/" (forward slash).

→ NOTE:

SQL is not a case sensitive.

→ Ex: write a query to display name of the student from student table.

(2) - SELECT SNAME
(1) - FROM STU; ——



D.B.

STEP 1:

STU:

SID	SNAME	AGE
220	Shiva	20
320	Shyam	25
120	Srinu	29

STEP 2:

RESULT

SNAME
Shiva
Shyam
Srinivas

Q: WAQTD age of the students:

```
SELECT AGE
FROM STU;
```

Q: WAQTD ID of the students.

```
SELECT SID
FROM STU.
```

01. Write a query to display all the details from the employee table.

```
SELECT *
FROM EMP;
```

02. WAQTD names of all the employees.

```
SELECT ENAME
FROM EMP;
```

03. WAQTD name and salary given to all the employees.

```
SELECT ENAME, SAL
FROM EMP;
```

04. WAQTD name and commission given to all the employees.

```
SELECT ENAME, COMM
FROM EMP;
```

05. WAQTD employee ID and department number of all the employees in emp table

```
SELECT EMPNO, DEPTNO
FROM EMP;
```

06. WAQTD ename and hiredate of all the employee

```
SELECT ENAME, HIREDATE
FROM EMP;
```

07 WAQTD name and designation of all the employees

```
SELECT ENAME, JOB
FROM EMP;
```

08. WAQTD name, job and salary given to all the employees.

```
SELECT ENAME, JOB, SAL
FROM EMP;
```

09. WAQTD DNAMES PRESENT IN DEPARTMENT TABLE.

```
SELECT DNAME
FROM DEPT;
```

10. WAQTD DNAME AND LOCATION PRESENT IN DEPT TABLE.

```
SELECT DNAME, LOC
  FROM DEPT;
```

Distinct:

It is used to remove the duplicated/repeated values from the result set.

- Whenever we are using distinct we have to use it as an first argument.
- One select clause will accept only one distinct.
- Distinct will be used in select clause only.

Ex: WAQTD DISTINCT NAMES OF THE STUDENT.

```
SELECT DISTINCT SNAME
FROM STU;
```

Step 1:

STU

SID	SNAME	AGE
101	PAVAN	25
102	SRINU	22
103	SHIVA	20
104	PAVAN	24

Step 2:

SNAME

✓ PAVAN
✓ SRINU
✓ SHIVA
✗ PAVAN

DISTINCT →

RESULT:

SNAME

PAVAN
SRINU
SHIVA.

Q: WAQTD DISTINCT JOB OF EMPLOYEES.

```
SELECT DISTINCT JOB
  FROM EMP;
```

Q: WAQTD SALARY OF EMPLOYEES WITHOUT HAVING REPEATED VALUES

```
SELECT DISTINCT SAL
  FROM EMP;
```

Q: WAQTD DISTINCT NAME, AGE OF THE STUDENT

```
SELECT DISTINCT SNAME, AGE
FROM STU;
```

STEP 2:

SNAME	AGE
PAVAN	23
PAVAN	22
POOJITHA	22
SARIKA	42

DISTINCT →

RESULT:

SNAME	AGE.
PAVAN	23
PAVAN	22
POOJITHA	22
SARIKA.	42

Example:

CUSTOMER

CID.	CNAME.	MOBILE NO.	PRODUCT ID	PRICE
CUS101	AKHIL	9876543210	PR102	21000/-
CUS102	PAVAN	9876543210	PR101	19000/-
CUS102	PAVAN	9876543210	PR101	19000/-
CUS103	AKHIL	9632587410	PR102	21000/-

Q: WAQTD ALL THE DETAILS OF THE CUSTOMER WITHOUT REPEATED VALUES.

```
SELECT DISTINCT *
FROM CUSTOMER;
```

STEP 1:

CUSTOMER:

CID.	CNAME.	MOBILE_NO	PRODUCT_ID	PRICE.
CUS101	AKHIL	9876543210	PR102	21000/-
CUS102	PAVAN	9876543210	PR101	19000/-
CUS102	PAVAN	9876543210	PR101	19000/-
CUS103	AKHIL	9632587410	PR102	21000/-

↓ DISTINCT.

RESULT:

CID	CNAME	MOBILE_NO	PRODUCT_ID	PRICE.
CUS101	AKHIL	9876543210	PR102	21000/-
CUS102	PAVAN	9876543210	PR101	19000/-
CUS103	AKHIL	9632587410	PR102	21000/-

(1). We are going to execute the table which is given in the FROM line.

i.e., FROM (CUSTOMER);

↓

and put it under execution.

(2) We are going to select all the columns for that we'll use. '*'. and for DISTINCT elements we'll use "DISTINCT".

i.e., SELECT DISTINCT *.

(3) And the output will be displayed as RESULT-TABLE.

Q: WAQTD UNIQUE CNAME, PRODUCT_ID OF THE CUSTOMERS.

```
SELECT DISTINCT CNAME, PRODUCT_ID FROM CUSTOMER;
```

CNAME	PRODUCT_ID
~~AKHIL~~	~~PR102~~
AKHIL	PR102
PAVAN	PR101

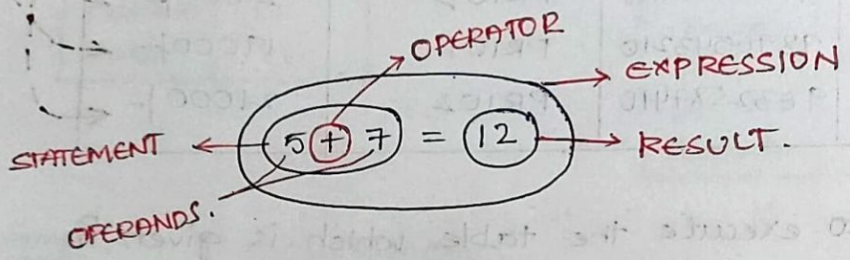
Q) WAQTD DISTINCT NAME OF CUSTOMERS.

```
SELECT DISTINCT CNAME FROM CUSTOMER;
```

CNAME
PAVAN
AKHIL.

EXPRESSION:

A statement which returns a result is known as expression.



- The combination of operator and operand is known as statement
- Operand: These are the values we have to pass.
- Operator: Operators are used to perform a specific task.

Q) WAQTD ANNUAL SALARY OF THE EMPLOYEES.

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

O/P:

SAL*12
9600
19200

Q) WAQTD HALF-TERM SALARY OF THE EMPLOYEES

```
SELECT SAL*6
FROM EMP;
```

Q: WAQTD QUATERLY SALARY OF THE EMPLOYEES

```
SELECT SAL*3
FROM EMP;
```

Q: WAQTD SALARY, SALARY WITH 10% HIKE.

```
SELECT SAL, SAL*110/100
FROM EMP;
```

ASSIGNMENT:

1) WAQTD name of the employee along with their annual salary.

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

2) WAQTD ename and job for all employee with their half term salary.

```
SELECT ENAME, JOB, SAL*6
FROM EMP;
```

3) WAQTD all the details of the employee along with an annual bonus of 2000.

```
SELECT EMP*, SAL*12+2000
FROM EMP;
```

4) WAQTD name, salary and salary with hike of 10%.

```
SELECT ENAME, SAL, SAL*110/100
FROM EMP;
```

5) WAQTD name and salary with deduction of 25%

```
SELECT ENAME, SAL-SAL/100*25
FROM EMP;
```

(OR)

SAL*0.75.

(6) WAQTD name and salary with monthly hike of 50

```
SELECT ENAME, SAL+50;
    FROM EMP;
```

(7) WAQTD name and annual salary with deduction of 10%.

```
SELECT ENAME, SAL*0.9*12.
    FROM EMP;
```

(8) WAQTD total salary given to each employee (SAL + comm)

```
SELECT SAL+COMM
    FROM EMP;
```

(9) WAQTD details of all the employee along with annual salary.

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

(10) WAQTD name and designation along with 100 penalty in salary.

```
SELECT ENAME, JOB, SAL-100
    FROM EMP;
```

TABLE NAME:

SAL

JOB.ID	JOB NAME	SAL	BONUS.
J101	Manager	15000	200
J102	Clerk	13000	120
J103	Manager	15000	-
J104	Analyst	12000	50

(1) WAQTD all the details of sal table along with annual-sal.

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

(2) WAQTD distinct job-name, sal along with 15% of hike

```
SELECT DISTINCT JOB.NAME, SAL+SAL/100*15.
    FROM SAL;
```

(3) WAQTD total income of the emp from sal table (SAL+BONUS)

```
SELECT SAL+BONUS
    FROM SAL;
```

(4) WAQTD SAL of the emp along with 20% penalty

```
SELECT SAL*0.8
    FROM SAL;
```

⇒ ALIAS:

- It is used to provide the alternate name to the column, expression and tables.
- whenever we want to provide alias name to the column (or) expression with or without using as keyword.

We can provide:

- Alias we can provide in 3 ways:
  (i) without using -: Annualsal
  (ii) with using -: Annual_sal
  (iii) with using space: "Annual sal".

Eg. 1:

```
SELECT SAL*12 AS ANNUALSAL
    FROM EMP;
```

Eg. 2:

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

Eg 3:

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

Eg. 4:

```
SELECT SAL*12 "Annual Sal".
    FROM EMP;
```

(1) WAQTD all the details of the employees along with annual salary, annual salary with penalty of 20% of sal and remove repeated values.

```
SELECT DISTINCT EMP.*, SAL*12, SAL*12*0.08
    FROM EMP;
```

(2) WAQTD ename, annual salary with deduction of 2 months of salary.

```
SELECT ENAME, SAL*12 - SAL*2
    FROM EMP;
```

(3) WAQTD ename, half term salary with bonus of. 100% of annual salary and penalty of 100% of annual salary.

```
SELECT ENAME, SAL*6 + 100/100 + SAL*12 - SAL*12 + 100/100
    FROM EMP;
```

⇒ Selection:

It is used to select the data from both row as well as column.

Syntax:

```
SELECT * Distinct col_name/expression Alias
    From Tablename
where < Filter_condition >;
          ↓
    (col.name = 'value')
```

⇒ Order of Execution:

(i) From
(ii) where
(iii) Select.

- After execution of from clause where clause will be executed.
- where clauses is used to filter the records
- where clause will execute row by row.
- After execution of where clause select clause will execute and it will display the data.

SELECTION:

Q. WAQTD SNAME OF STUDENTS. IF THEY ARE HAVING MORE THAN 80% PERCENTAGE.

STEP 1:

STU.

SID	SNAME	PERCENTAGE
101	SRINU	89 ✓
102	SHIVA	59 X
103	SHYAM	70 X
104	SUMANTH.	48 X

STEP 2:

SID	SNAME	PERCENTAGE
101	SRINU	89

STEP 3:

SNAME.
SRINU

Syntax:

SELECT * / DISTINCT COL_NAME / EXPRESSION ALIAS

FROM TABLE_NAME.

WHERE. < FILTER_CONDITIONS>;

↓

(COL_NAME = 'VALUE)

ANS:

```
SELECT SNAME.
  FROM STU
WHERE. PERCENTAGE > 80;
```

ORDER OF EXECUTION:

1. FROM
2. WHERE
3. SELECT.

Q. WAQTD the annual salary of the employee whose name is smith.

~~SELECT ENAME~~

```
SELECT SAL*12
  FROM EMP
WHERE ENAME = 'SMITH';
```

upper case.

→ Data is case sensitive

Q. WAQTD name of the employees working as clerk.

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

Q. WAQTD salary of the employee who are working as salesman.

```
SELECT SAL
  FROM EMP
WHERE JOB = 'SALESMAN';
```

Q. WAQTD details of the emp who earns more than 2000.

```
SELECT EMP.*
  FROM EMP
WHERE SAL > '2000';
```

NOTE: FOR NUMBER '' - not required.

Q. WAQTD details of the emp whose name is Jones.

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

Q. WAQTD details of the emp who has hired after 01-JAN-81.

```
SELECT *
  FROM EMP
WHERE HIREDATE > '01-JAN-81';
```

Q. WAQTD name and sal along with his annual income if the annual salary is more than 12000.

```
SELECT ENAME, SAL, SAL*12 Annual_sal
  FROM EMP
WHERE SAL*12 > '12000';
```

Q: WAQTD empno of the employees who are working in Dept 30.

```
SELECT EMPNO
  FROM EMP
WHERE DEPTNO = '30';
```

→ NO ROWS PRESENT IN EMP FOR DEPT 30.

OPERATORS:

- Operators are used to perform a specific task.
- We are having 7 types of operators.

1) Arithematic Operators.
2) Concatination Operators.
3) Comparision Operators. — '=' - Equals to; != (or) <> - Not Equals
4) Relational Operators
5) Logical Operators. [AND, OR, NOT]
6) Special Operators [IN, NOT IN, IS, IS NOT BETWEEN, NOT BETWEEN, LIKE, NOT LIKE]
7) Sub Query Operators: [All, Any, Exists, Not-Exists]

⇒ Arithematic Operators:

We are having four types.

1) Addition
2) Subtraction
3) Multiplication
4) Division.

Examples:

(1)
```
SELECT 11+20
  FROM DUAL;
```

O/P:
```
11+20
-----
   31
```

DUAL-TABLE:

- Dual is a dummy table, it contains only one row and one column.
- whenever we are passing the values in our select class at that time we have to use dual as a table.

Example:

(2) Subtraction:

```
SELECT 100-20
  FROM DUAL;
```

(3) Multiplication.

```
SELECT 10*20
  FROM DUAL;
```

(4) Division

```
SELECT 100/50
  FROM DUAL;
```

*. For giving inputs while execution:

```
SELECT '&Enter_1st value' + '&Enter_2nd-value'
    FROM DUAL;
```

o/p: Enter value for Enter_1st_value :
enter value for Enter_2nd_value :

old: 1: SELECT '&Enter_1st_value' + '&Enter_2nd_Value'

new: 1: SELECT '10+'30'.

'10'+'30'
40.

Ex:

```
SELECT '&Enter_1st_value' + &Enter_2st_value',
       '&Enter_1st_value' - '&Enter_2nd_value',
       '&Enter_1st_value' * '&Enter_2nd-value',
       '&Enter_1st_value' / '&Enter_2nd-value',
  FROM DUAL;
```

o/p:

* Concatination Operator: (||)

- It is used to combine two or more strings.

Syntax: SELECT 'STRING 1' || 'STRING2' || ------
FROM TABLE-NAME.

Example: SELECT 'SRINIVAS' || 'CHENNA'
FROM DUAL;

o/p:

'SRINIVAS'||'CH.
SRINIVASCHENNA

Q: WAQT combine ename, job if the employee is working as a salesman. ← ' '||

```
SELECT ENAME || JOB
  FROM EMP
WHERE JOB = 'SALESMAN';
```

ASSIGNMENT:

Q: WAQTD ename and hiredate if they are hired before 1981.

```
SELECT ENAME, HIREDATE.
  FROM EMP
WHERE HIREDATE > '01-JAN-1981';
```

Q: WAQTD details of the employees working as a manager.

```
SELECT emp.*
  FROM EMP
WHERE JOB = 'MANAGER';
```

Q' WAQTD name and salary given to an employee if employee earns a commission of rupees 1400.

```
SELECT ENAME, SAL
FROM EMP
WHERE COMM = 1400;
```

Q: WAQTD details of employees having commission more than salary.

```
SELECT ENAME, SAL
 FROM EMP
WHERE COMM > SAL;
```

Q: WAQTD details of employees having commission more than salary.

Q: WAQTD empno of employees hired before the year 87.

```
SELECT EMPNO.
 FROM EMP
WHERE HIREDATE < '01-JAN-87';
```

Q: WAQTD details of employees working as an 'N' ana-lyst.

```
SELECT EMP.*
 FROM EMP
WHERE JOB = 'ANALYST";
```

Q' WAQTD details of emps earning more than 2000 rupees per month.

```
SELECT EMP.*
 FROM EMP
WHERE SAL >'2000';
```

09-APR-2024.

Q: WAQT joing. MR. to the name of the employees, if the employee working as manager.

```
SELECT 'MR' || ENAME
  FROM EMP
WHERE JOB = 'MANAGER';
```

Q: WAQTD. employee name with his salary so that the output. is SMITH IS GETTING 800.

```
SELECT ENAME || ' IS GETTING ' || SAL
  FROM EMP;
```

Comparision Operators:

It is used to compare the values.
There are two operators =, != or <>

Q: WAQTD. name of the employees who is working as a manager.

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

Q: WAQTD. ename, deptno of the employees if the emp is working in dept 10.

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

Q: WAQTD. all the details of the employees except manager.

```
SELECT EMP.*
  FROM EMP
WHERE JOB != 'MANAGER'; / WHERE JOB <> 'MANAGER';
```

Q: WAQTD all the details of the employees along with halfterm sal., annual sal, if the employee is not working as saleman.

```
SELECT EMP.*, SAL*6 'HALF-TERM', SAL*12 ANNUAL-SAL
 FROM EMP
WHERE JOB <> 'SALESMAN';
```

RELATIONAL-OPERATORS:

It is used to compare values, not exactly equal ones, but greater than, or less than of the given value.

Q: WAQTD. name of the employees along with their salary. if the employee is getting salary more than 2000.

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

Q: WAQTD ename, job., sal of the employees who is getting salary of min 2000.

```
SELECT ENAME, SAL, JOB
 FROM EMP
WHERE SAL >= 3000;
```

Q: WAQTD. all the details of the employees if the employee was hired before 1984.

```
SELECT EMP.*
 FROM EMP
WHERE HIREDATE < '01-JAN-1984';
```

LOGICAL-OPERATORS:

Logical AND
Logical OR
Logical NOT
} NO SYMBOLS

→ AND Operator.

- It is used to satisfy both the conditions.

Syntax:

```
WHERE <CONDITION-1> AND <CONDITION-2> AND -----
```

Q: WAQTD name of the employee. If the emp. is getting sal morethan 2000 and present in dept 10. — (c2)

(c1)

STEP 1:

EMP:

ENAME	SAL	DEPTNO
SRINU	1000	10
SHYAM	2500	10
SHIVA	5000	20
PRANAY	3000	30

C1	AND	C2	=	R
✗	AND	✗		✗
✓	AND	✓		✓
✓	AND	✗		✗
✓	AND	✗		✗

③ — SELECT ENAME.

① — FROM EMP

② — WHERE {SAL > 2000}(C1) AND {DEPTNO = 10}(C2);

STEP 2:

ENAME	SAL	DEPT NO
SHYAM	2500	10

STEP 3:

ENAME
SHYAM } RESULT.

Q: WAQTD ename, sal, hiredate of the employees. If the employee is getting sal less than 4000. and they are hired after 1982.

```
SELECT ENAME, SAL, HIREDATE.
FROM EMP.
WHERE SAL < 4000 AND HIREDATE > '31-DEC-1982';
```

NOTE: We can pass date in lowercase format also i.e., 31-dec-1982.

Q: WAQTD name, job of the employee. If the employee is working as a manager in dept 20.

```
SELECT ENAME, JOB.
FROM EMP
WHERE JOB = 'MANAGER' AND DEPTNO = 20;
```

Q: WAQTD all the details of the employees. if the empl-oyee getting annual sal more than 12000. and they are hired in the year of 1981.

```
SELECT EMP.*
FROM EMP.
WHERE SAL*12 > 12000 AND HIREDATE >= '01-JAN-1981' AND
                     HIREDATE <= '31-Dec-1981';
```

→ OR-Operator:

It is used to satisfy any one of the condition

Syntax: WHERE <CONDITION 1> OR <CONDITION 2> OR....

Q: WAQTD name of the emp. If the emp is getting sal more than 2000. or present in dept 10.

```
SELECT ENAME
FROM EMP
WHERE SAL > 2000 OR DEPTNO = 10;
```

Explanation: OR-operator.

Step 01:

EMP:

ENAME	SAL	DEPT.NO
PAVAN	1000	10
SARIKA	2500	10
SRINIVAS	5000	20
SUMI	3000	30

C1	OR	C2	=	R
×	OR	✓	=	✓
✓	OR	✓	=	✓
✓	OR	×	=	✓
✓	OR	×	=	✓

Step 02:

ENAME	SAL	DEPT.NO
SARIKA	2500	10
PAVAN	1000	10
SRINIVAS	5000	20
SUMI	3000	30

Step 03:

ENAME
PAVAN
SARIKA
SRINIVAS
SUMI

} Result:

Q: WAQTD name of the employee who is working in deptno 10, 20.

```
SELECT ENAME
FROM EMP
WHERE DEPTNO = 10 OR DEPTNO = 20;
```

Q: WAQTD name, job of the employees if the employee is working as a manager, analyst.

```
SELECT ENAME, JOB FROM emp
WHERE JOB = 'MANAGER' OR JOB = 'ANALYST';
```

10-Apr-24

→ NOT-OPERATOR:

oIt is used to reject the records

Syntax: WHERE NOT <CONDITION1> ;

AND NOT <CONDITION 2> .... ;

NOTE:

If we use OR operator in place of AND operator while running the query.

It will check condition by condition. So, while checking condition 1, condition 2 will be true and vice versa.

So, we will use AND operator only.

Q: WAToD ename, sal, job of the emp. Except who are working as salesman.

```
SELECT ENAME, SAL, JOB
FROM emp
WHERE NOT JOB = 'SALESMAN';
```

STEP 01:

EMP:

ENAME	JOB	SAL
KING	MANAGER	5000
QUEEN	PRESIDENT	6000
MINISTER	SALESMAN	1000
SOLDIER	FIRE	500

✓ ✓ × ✓

③ SELECT ENAME, SAL, JOB

① FROM EMP

② WHERE NOT JOB = 'SALES-MAN';

ENAME	JOB	SAL
KING	MANAGER	5000
QUEEN	PRESIDENT	6000
SOLDIER	FIRE	500

ENAME	JOB	SAL.
KING	MANAGER	5000
QUEEN	PRESIDENT	6000
SOLDIER	FIRE	500

1) WAQTD all the details of the employees who are not working as a salesman, manager.

STEP 01:

EMP:

EMPNO	ENAME	JOB
101	A	MANAGER
102	B	SALESMAN
103	C	ANALYST
104	D	CLERK

③ SELECT *.
① FROM EMP
② WHERE NOT JOB = 'MANAGER'
AND NOT JOB = 'SALESMAN';

C1	AND	C2	=	R
✗		✓		✗
✓		✗		✗
✓		✓		✓
✓		✓		✓

STEP 2:

EMPNO	ENAME	JOB.
103	C	ANALYST
104	D	CLERK.

STEP 3:

EMPNO	ENAME	JOB
103	C	ANALYST
104	D	CLERK.

2) WAQTD. name of the employees who is not getting salary more than 3000.

```
SELECT ENAME.
FROM EMP
WHERE NOT SAL >= 3000;
```

3) WAQTD ename, job, sal, deptno and hiredate of the employees. if the employee is working in deptno 10, 20, as a manager and they are not working as a salesman, analyst or clerk.

```
SELECT ENAME, JOB, SAL, DEPTNO, HIREDATE
FROM EMP
WHERE DEPTNO = 10 OR DEPTNO = 20. AND JOB = 'MANAGER'
OR DEPTNO = 20 AND JOB = 'MANAGER' AND
NOT JOB = 'SALESMAN' AND NOT JOB = 'ANALYST' AND
NOT JOB = 'CLERK';
```

4) WAQTD all the details of the employees along with the annual salary. If the employee is not getting annual salary more than 19200. and they are not getting annual salary less than 13000. and they are working as a manager, analyst in the deptno 10, 20.

```
SELECT emp.*, SAL*12  ANNUAL-SAL
FROM emp.
WHERE NOT SAL*12 > 19200 AND NOT SAL*12 < 13000. AND.
(JOB = 'MANAGER' OR JOB = 'ANALYST') AND (DEPTNO=10
 OR DEPTNO=20) ;
```

SPECIAL-OPERATORS:

→ We are having 8 special-operators.

(i) IN
(ii) NOT IN
(iii) IS
(iv) IS NOT
(v) BETWEEN
(vi) NOT BETWEEN
(vii) LIKE
(viii) NOT LIKE.

→ IN-operator:

- Instead of '=' (equals to), we are using 'IN' operator.
- 'IN' operator will accept one value at L.H.S. and one or multiple values at R.H.S.
- whenever we want to pass set of values we will be using. IN operator.

IN:

Syntax: WHERE COL_NAME IN ('V1', 'V2', ...);

Q: WAQTD NAME, DEPTNO OF THE EMP. IF THE EMP IS WORKING IN DEPT 10, 20 ?

STEP 01:

EMP.

EMPNO	ENAME	DEPTNO
111	ARJUN	10
222	ANITHA	20
121	SUMI	30
212	PRASHANTH	10

③ SELECT ENAME, DEPTNO
① FROM EMP
② WHERE DEPTNO IN (10, 20);

STEP 02:

EMPNO	ENAME	DEPTNO
111	ARJUN	10
222	ANITHA	20
212	PRASHANTH	10

STEP 03:

ENAME	DEPTNO
ARJUN	111
ANITHA	222
PRASHANTH	212

} RESULT

Q: WAQTD. ename, job, hiredate, sal of. emp. If the emp. is working as manager, salesman in deptno 30.

```
SELECT ENAME, JOB, SAL, HIREDATE
FROM emp
WHERE JOB IN ('MANAGER', 'SALESMAN') AND DEPTNO = 30;
```

Q: WAQTD all the details of the emp. If. the emp is getting 3000, 5000 as sal.

```
SELECT *
FROM emp
WHERE SAL IN (3000, 5000);
```

Q: WAQTD emp.no, job, sal, deptno of the emp. If. the employees are having empno. is 7499, 7566, 7900. and they are working as salesman, clerk.

```
SELECT EMPNO, JOB, SAL, DEPTNO
FROM emp.
WHERE EMPNO IN (7499, 7566, 7900) AND JOB IN ('SAL
                                    ESMAN', 'CLERK');
```

⇒ NOT IN - OPERATOR:

- It is similar to IN - Operator but instead of selecting the values ~~and~~ it will reject the values and remaning records will be displayed.

NOT-IN:

Syntax: WHERE COL_NAME NOT IN ('V1', 'V2', ----);

Q: WAQTD NAME, DEPTNO OF THE EMP. IF THE EMP IS NOT WORKING IN DEPTNO 10, 30

STEP 01

EMP.

ENAME	DEPTNO
ARJUN	10
ANITHA	10
SUMI	20
SRINU	30

STEP 02:

ENAME	DEPTNO
SUMI	20

STEP 03:

ENAME	DEPTNO
SUMI	20

RESULT

③ SELECT ENAME, DEPTNO

① FROM EMP

② WHERE DEPTNO NOT IN (10,30);

Q: WAQTD all the details of employees except manager, clerk

```
SELECT *
FROM EMP
WHERE JOB NOT IN ('MANAGER','CLERK');
```

Q: WAQTD ename, job, sal and deptno of the employee if. employee is not working in deptno30 as a man-ager.

```
SELECT ENAME, JOB, SAL, DEPTNO
FROM emp
WHERE. DEPTNO NOT IN (30) AND JOB IN 'MANAGER';
```

Q: WAQTD all the details of the emp. who are not getting sal is 1250, 3000, 5000, 16000, 2850 and 2975. and they are not working as analyst, president, saksman.

```
SELECT *
FROM emp
WHERE SAL NOT IN (1250, 3000, 5000, 1600, 2850, 2975)
                AND JOB NOT IN ('ANALYST', 'PRESIDENT', 'SALES
                                                    MAN');
```

⇒ IS-Operator:

- It is used to compare with "null" values.

Syntax: WHERE. COLNAME IS NULL;

Q: WAQTD name of the student who is not having mobile number.

STEP01:

STU:

SID	SNAME	MOBILE
101	A	9876543210
102	B	
103	C	9611453219
104	D	

```
SELECT SNAME
FROM STU
WHERE MOBILE IS NULL;
```

STEP02:

SID	SNAME	MOBILE.
102	B	
104	D	

STEP03:

SNAME
B
D.

RESULT

Q: WAQTD ename, comm of the employees who is not getting the comm.

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

Q: WAQTD all the details of the employees if the emp are not having manager and they are not getting comm.

```
SELECT EMP.*
FROM EMP
WHERE MGR IS NULL AND COMM IS NULL;
```

Q: WAQTD ename, sal, comm of the emp. If the emp is not getting sal or comm.

```
SELECT ENAME, SAL, COMM
FROM emp
WHERE SAL IS NULL OR COMM IS NULL;
```

⇒ IS NOT - operator:

- It is similar to IS - operator. But, instead of selecting the values, it will reject the value.

Syntax: `WHERE COL_NAME IS NOT NULL;`

Q: WAQTD name of the student who is having mobileno.

STEP 01: STU

SNAME	MOBILE
A	98765432
B	
C	

```
SELECT SNAME
FROM STU
WHERE MOBILE IS NOT NULL;
```

STEP 02:

SNAME	MOBILE
A	98765432

STEP 03:

SNAME
A

} RESULT

Q: WAQTD name of the emp who is getting comm

```
SELECT ENAME
FROM emp
WHERE comm IS NOT NULL;
```

Q: WAQTD ename, sal, mgr of the emp. if the emp is having mgr.

```
SELECT ENAME, SAL, MGR
FROM emp
WHERE MGR IS NOT NULL;
```

Q: WAQTD name of the emp who is getting sal and comm.

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

⇒ BETWEEN - Operator.

- It is used to select the range of values.

Syntax: WHERE COL_NAME BETWEEN 'STARTING RANGE VALUE' AND 'ENDING RANGE VALUE';

- For names also we can use., it will be executed upon characters.

Syntax: WHERE COL_NAME OF 'STARTING VALUE +1' AND 'ENDING VALUE -1';

Q: WAQTD name of the movie If the movie release date is in the range of 2022 - 2025.

SELECT MNAME FROM MOVIE WHERE RELEASE_DATE BETWEEN '01-JAN-22' and '31-DEC-24'

STEP 01: MOVIE:

MNAME	RELEASE_DATE
PUSHPA	15-AUG-2024
LEO	31-DEC-2023
EEGA	01-JAN-2018
WAR	02-FEB-2026

STEP 02:

MNAME	RELEASE_DATE
PUSHPA	15-AUG-2024
LEO	31-DEC-2023

STEP 03:

MNAME
PUSHPA
LEO
} RESULT

Q' WAQTD name of the emp along with their sal. If emp is getting sal in the range b/w 1100 to 3000.

```
SELECT ENAME, SAL
FROM EMP
WHERE SAL BETWEEN 1101 and 2999;
```

Q' WAQTD all the details of the employees. If the emp are hired in the range of 1982 to 1984.

```
SELECT EMP.*
FROM EMP
WHERE HIREDATE BETWEEN '01-JAN-1982' and '31-DEC-1984';
```

Q' WAQTD ename, sal, hiredate of the employees. If the emp. is getting sal in the range of 1000 to 3000. or they are hired in the range b/w 1980 to 1985.

```
SELECT ENAME, SAL, HIREDATE
FROM EMP
WHERE (SAL BETWEEN 1000 and 3000) OR (HIREDATE
      BETWEEN '01-JAN-1981' and '31-DEC-1984');
```

⇒ NOT BETWEEN :- Operator:

- It is similar to between operator. But instead of selecting the value it will reject the values and select the remaining values.

Syntax:

```
WHERE COLNAME NOT BETWEEN 'STARTING VALUE' AND 'ENDING VALUE'
```

Q' WAQTD name of the movie if the movie release release date is not between of 2024-2026

```
SELECT MNAME
FROM MOVIE
WHERE RELEASE_DATE NOT BETWEEN '01-JAN-2024' and
      '31-DEC-2026';
```

STEP 01:

MNAME	RELEASE-DATE
PUSHPA	15-AUG-2024
LEO	31-DEC-2023
EEGA	01-JAN-2018
WAR	02-FEB-2026

STEP 02:

MNAME	RELEASE-DATE
~~PUSHPA~~ EEGA	01-JAN-2018
~~WAR~~ LEO	~~31~~ 31-DEC-2023

STEP 03:

~~MNAME~~
EEGA
LEO.

Q! WAQTD name of the emp along with their sal. If emp is not getting sal in the range b/w 1100 to 3000.

```
SELECT ENAME, SAL
FROM EMP
WHERE SAL NOT BETWEEN 1101 and 2999;
```

Q! WAQTD all the details of the emp. If the emp are not hired in the range of 1982 to 1984.

```
SELECT ~~ENAME~~ EMP.*
FROM EMP
WHERE HIREDATE NOT BETWEEN '01-JAN-1982' and
                           '31-DEC-1984';
```

Q! WAQTD ename, sal, hiredate of the emp. If the emp is not getting sal in the range of 1000 to 3000. or they are not hired in the range btwn 1980 to 1985.

```
SELECT ENAME, SAL, HIREDATE
FROM EMP
WHERE (SAL NOT BETWEEN 1000 and 3000) OR
      (HIREDATE NOT BETWEEN '01-JAN-1981' and
                            '31-DEC-1984');
```

15-APR-2024:

⇒ LIKE-OPERATOR:

- It is used to pattern to match.

Syntax: WHERE COLNAME LIKE 'PATTERN_TO-MATCH';

- We are having two arguments
  '%' - when we don't know the length of string
  '_' (underscore) - when we know the length of string.

Eg: WAQTD add of the cus who's and starts with 'N'.

STEP 01:

CUS:

CID	CNAME	ADD
101	SUMI	HUB
102	SAKIB	BLR
104	SASUKE	NZB
111	NARUTO	JG

③ SELECT ADD
① FROM CUS
② WHERE ADD LIKE 'N%';

STEP 02:

CID	CNAME	ADD
104	SASUKE	NZB

STEP 03:

ADD
NZB
} RESULT

Q: WAQTD CNAME OF CUS IF COS NAME ENDS WITH RAO.

```
SELECT CNAME
FROM CUS
WHERE CNAME LIKE '%RAO';
```

Q: WAQTD all the details of emp who's name contains two A's.

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

Q: WAQTD names of the emp. whose name contains 'R' is the 3rd character.

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

Q: WAQTD names of the emp. whose name ends with character 'S'.

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

Q: WAQTD name of the emp. If their name contains T. is the last but one character.

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

Q: WAQTD ename, job of emp. If their job contains 'A' is the 2nd character from both the ends. and M is the last but 2nd character.

```
SELECT ENAME, JOB
FROM EMP.
WHERE JOB LIKE '_A%MA__';
```

Q: WAQTD ename, hiredate of the employee who are hired in the month of april.

```
SELECT ENAME, HIREDATE
FROM EMP
WHERE HIREDATE LIKE '%APR%';
```

Q: WAQTD ename, hiredate of employee who are hiredate in the year of 81.

```
SELECT ENAME, HIREDATE
FROM emp
WHERE HIREDATE LIKE '%81';
```

Q: WAQTD all the details of the emp. If the emp name starts with A or S or M and job contains A is the third character and they are getting 4 digit salary or salary ends with 50.

```
SELECT emp.*
FROM emp
WHERE (ENAME LIKE 'A%' OR ENAME LIKE 'S%'
      OR ENAME LIKE 'M%') and job LIKE '__A%'
      AND (SAL LIKE '____' OR SAL LIKE '%50');
```

⇒ NOT LIKE - OPERATOR:

- It is used to pattern not to match.

Syntax: WHERE COL_NAME NOT LIKE 'PATTERN_TO_MATCH';

- We are having two arguments
  (i) '%' and (ii) '_'

Q: WAQTD ENAME OF THE CUSTOMER WHOS NAME DOSENT START WITH 'S'.

STEP 01:

CUS:

CID	CNAME	ADD
101	SRINU	JGL
102	SHIVA	MTPL
103	SHYAM	KMNR
104	NAJU	NZB

③ — SELECT ENAME.
① — FROM CUS
② — WHERE. CNAME NOT LIKE 'S%';

STEP 02:

CID	CNAME	ADD
104	NAJU	NZB

STEP 03

ENAME.
NAJU

} RESULT.

Q: WAQTD all the details of emp whos name contains two A's. ← dosen't

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

Q: WAQTD names of the emp. whose name dosent contains 'R' as the 3rd character.

```
SELECT ENAME
FROM EMP
WHERE ENAME NOT LIKE '__R%';
```

Q: WAQTD name of the emp. If their name dosent contains 'T' is the last but one character.

```
SELECT ENAME.
FROM EMP
WHERE ENAME NOT LIKE '%T_';
```

Q: WAQTD ename, job of emp. If their job1 contains 'A' is the 2nd character from both the ends, and 'M' is the last but 2nd character.

```
SELECT ENAME, JOB
FROM emp.
WHERE JOB NOT LIKE '_A%MA_';
```

Q! WAQTD ename, hiredate of the emp who are not hired in the month of april.

```
SELECT ENAME, HIREDATE.
FROM EMP
WHERE HIREDATE NOT LIKE '%APR%';
```

Q! WAQTD ename, hiredate of emp who are hired not in the year of 81.

```
SELECT ENAME, HIREDATE.
FROM EMP
WHERE HIREDATE NOT LIKE '%81';
```

16-APR-2024

⇒ Functions:

- Function is a set of instructions or block of code used to perform a specific task.
- We are having two types of functions
  (i) User Defined Function
  (ii) Pre-Defined Function.

⇒ Pre-Defined Function:

- These are also known as Built-In Functions.
- We are having two types of pre-defined functions
  (i) Single Row function
  (ii) Multi Row function

(i) Single Row Function:

whenever we are giving 1 value as an input. It will execute and it will provide only 1 output.

- If we pass 'N' no.of. values as an input. It will execute and it will provide 'N' no.of. output.
  → It will consider each i/p individually.

NOTE:

- Single row functions will execute. Row by. Row.
- SRF() we can use in SELECT, WHERE clauses.

→ 1 I/p = 1 o/p
N I/p = N o/p.

(ii) Multi-Row Function:

- whenever we are giving 1 input it will be considered as a single group. and it will provide only 1 output.
- whenever we are giving 'n' no.of. input it will also be considered as a single group and it will provide only 1 output.

NOTE:

1) MRF() will execute group by. group.
2) MRF() we can use in SELECT clause.
3) 1 I/p = 1 o/p.
   N i/p = 1 o/p.

→ In MRF() we are having 5 types

1. MAX();
2. MIN();
3. AVG();
4. SUM();
5. COUNT();

→ NOTE!

1) It will accept only. 1 argument within the function

Ex! Only 1 Arg i.e-, COL_NAME/EXP.

MRF(COL_NAME/ EXPRESSION);

2) We are not suppose to write any one of the other arguments like col_name or expression along with the multi row function in the select clause.

Ex! SELECT MRF(C_N), C_N —×

3) MRF() are not possible to use in the 'WHERE' clause

Ex! WHERE MRF(C_N) —(×)

4) MRF() ignores null values.

Ex! COUNT(COMM); → o/p (4) — ignores null values.

5) Only the count() will accept '*' as an argument, remaining functions cannot accept '*' as an args.

Ex! COUNT(*); → o/p (14) — accepts '*' value as argument.

→ COUNT() → TABLE, COLUMN

MAX() }
MIN() }
AVG() } COLUMN
SUM() }

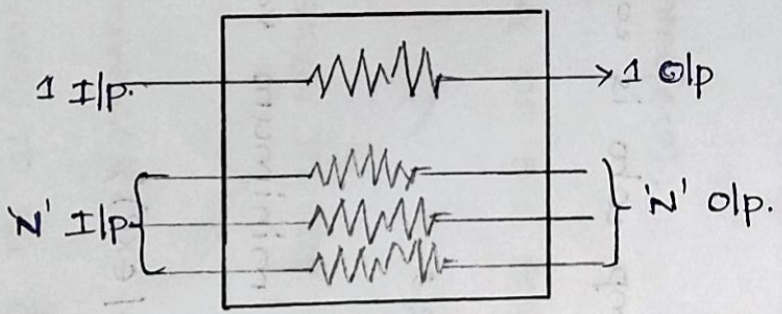
→ DEFINED. (or) BUILT-IN

MRF

FUNCTIONS.

FUNCTIONS.

- USER DEFINED FUNCTIONS.
- PRE DEFINED. (or) BUILT-IN FUNCTION.
  - SINGLE ROW FUNCTION
  - MULTI ROW FUNCTION (or) GROUP / AGGREGATE FUNCTIONS.



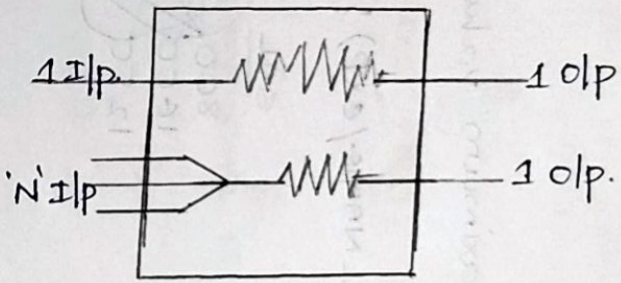
SRF()

→ ROW BY ROW

→ WHERE, SELECT.

NOTE: 1 I/p = 1 O/p.

N I/p = N O/p



MRF()

→ Group by. Group.

→ SELECT

Types!

(i) MAX();
(ii) MIN();
(iii) AVG();
(iv) SUM();
(v) COUNT();

NOTE!

* Only 1 Argument
* SELECT
→ SELECT MRF (C-N), C-N ✗
* WHERE. MRF (C-N) ✗
* Ignores null
* count(*).

MAX():

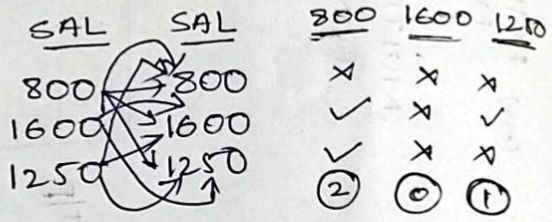
- It is used to get the maximum value present in the column.

Syntax: MRF MAX(COL_NAME/EXP);

Ex: SELECT MAX(SAL)
FROM EMP;

o/p:

MAX(SAL)
1600



Q: WAQTD max sal present in dept 10.

```
SELECT MAX(SAL)
  FROM EMP
 WHERE DEPTNO IN 10;
```

Q: WAQTD max sal of emp. who is working as a manager.

```
SELECT MAX(SAL)
 FROM EMP
WHERE JOB IN 'MANAGER';
```

MIN():

- It is used to get the minimum value present in the column.

Syntax: MIN(COL_NAME/EXP);

Ex: SELECT MIN(SAL)
FROM EMP;

o/p:

MIN(SAL)
800.

Q: WAQTD MIN SAL OF EMP.

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

Q: WAQTD MIN SAL PRESENT IN DEPT 30.

```
SELECT MIN(SAL)
FROM EMP
WHERE DEPNO IN 30;
```

Q: WAQTD WHO ARE HIRED FIRST.

```
SELECT MIN (HIREDATE)
FROM EMP;
```

Q: WAQTD MIN SAL OF EMP WHO IS WORKING AS A ANALYST.

```
SELECT MIN(SAL)
FROM EMP
WHERE JOB IN 'ANALYST';
```

AVG():

- It is used to get the average value present in the column.

Syntax: AVG(COL-NAME/EXP);

Ex:

Q: WAQTD AVERAGE SAL OF ALL EMP.

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

Q: WAQTD AVG SAL GIVEN TO THE MANAGER'S.

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

Q: WAQTD AVG SAL GIVEN TO ALL EMP EXCEPT MANAGER.

```
SELECT AVG(SAL)
FROM EMP
WHERE JOB NOT IN 'MANAGER';
```

NOTE: AVG - CANNOT BE USED FOR CHARACTERS..

i.e., ONLY DIGITS SHOULD BE PRESENT.

SUM():

- It is used to get summation of values present in the column.

Syntax: SUM(COL-NAME/EXP);

Ex:

Q: WAQTD SUM OF SAL OF ALL EMP: / TOTAL SAL OF EMP:

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

Q: WAQTD TOTAL SAL OF EMP. IF EMP NAME CONTAINS CHARACTER 'E' LAST BUT '1' CHARACTER. AND THEIR NAMES CONTAINS 'A' AND THEY ARE HIRED IN MONTH OF FEB.

```
SELECT SUM(SAL)
FROM EMP
WHERE (ENAME LIKE '%E_') AND (ENAME LIKES '%A%')
        AND HIREDATE LIKE '__-FEB-%';
```

Q. WAQTD TOTAL COMM OF EMP. WHO IS GETTING COMM

```
SELECT SUM(COMM)
FROM EMP
WHERE COMM IS NOT NULL;
```

Ex:

⇒ COUNT():

It is used to get no. of values present in a column or table.

Syntax: COUNT(*/COL-NAME/EXP);

Eg: WAQTD NO. OF VALUES PRESENT IN COMM COL

```
SELECT COUNT(COMM)
  FROM EMP;
```

Q: WAQTD NO. OF emp WORKING AS A MANAGER.

```
SELECT COUNT(JOB)
FROM EMP
WHERE JOB IN 'MANAGER';
```

Q: WAQTD TOTAL RECORDS PRESENT IN EMP TABLE.

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

→ GROUP BY CLASS:

- It is used to group the records.

Syntax:

```
SELECT GROUP_FUNCTION/COL_NAME/EXPRESSION
FROM TABLE_NAME
[WHERE FILTER_CONDITION] - optional.
GROUP BY COL_NAME/EXPRESSION;
```

SAL ... SAL

Same col_name should be present to execute.

If they are not same. It'll not execute.

ORDER OF EXECUTION:

1. FROM
2. WHERE. [Row by Row].
3. GROUP BY [Row by Row].
4. SELECT [col by col].

NOTE:

1) After execution of where clause group by will execute.
2) It is used to group the records and it will execute row by row.
3) After execution of group by we will get the groups
4) After execution of group by. The select clause will execute group by group.

5) After execution of group by, remaining all the clauses will execute group by group.

6) Whatever the arguments we are using in the group by, the same arguments we have to use in our select clause. [If it is required].

7) Grup function is not possible to use in group by.

18|APR|2024

Q. WAQTD NO.OF STUDENTS PRESENT IN EACH BRANCH.?

① STU

SID	SNAME	BRANCH
101	PAVAN	CIVIL
102	SARIKA	BCA
103	SRINU	EEE
104	SUMI	IT
105	RITHYA	BCA
106	SHIVA	IT
107	SHYAM	EEE
108	PRANAY	CIVIL

⇒ SELECT COUNT(*), BRANCH
FROM STU
GROUP BY BRANCH;

②

CIVIL

SID	SNAME	BRANCH
101	PAVAN	CIVIL
108	PRANAY	CIVIL

BCA

SID	SNAME	BRANCH
102	SARIKA	BCA
105	RITHYA	BCA

EEE

SID	SNAME	BRANCH
103	SRINU	EEE
107	SHYAM	EEE

IT

SID	SNAME	BRANCH
104	SUMI	IT
106	SHIVA	IT

Q: WAQTD max sal present in each dept.

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

Q: WAQTD max sal, min sal of each job and deptno of emp.

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

Q: WAQTD mgr and no.of.times the value is present.

```
SELECT MGR, COUNT(*)
FROM EMP
WHERE MGR IS NOT NULL
GROUP BY MGR;
```

19|APR|24

⇒ HAVING()

- It is used to filter the groups.

Syntax:

```
SELECT GROUP_FUNCTION/COL_NAME/EXP
FROM EMP
[WHERE FILTER_CONDITION]
GROUP BY COL_NAME/EXP
HAVING GROUP_FILTER_CONDITION
```

1) After execution of group by. having class will execute
2) It will filter the groups and it will execute group by group.
3) In having class we have to write group filter condition by using multi row functions.

Q: WRITE BRANCH. NO. OF STUDENTS PRESENT IN EACH BRANCH ? IF THE BRANCH CONTAINS MORE THAN 2 STUDENTS.

STEP01:

STU:

SID	SNAME	BRANCH
101	PAVAN	ECE
102	SAM	ECE
103	HARI	ECE
104	SUMA	MCA.
105	SUMI	MCA.
106	CNU	MCA
107	SHIVA	BSC
108	KANNA	BBA.

STEP02:

ECE

SID	SNAME	BRANCH
101	PAVAN	ECE
102	SAM	ECE
103	HARI	ECE

MCA.

SID	SNAME	BRANCH
104	SUMA	MCA
105	SUMI	MCA
106	CNU	MCA.

BSC

SID	SNAME	BRANCH
107	SHIVA.	BSC

BBA.

SID	SNAME	BRANCH
108	KANNA	BBA

STEP 03.

COUNT(*) ✓
3

Count(*) ✓
3

COUNT(*) ✗
1

COUNT(*) ✗
1

COUNT(*) >2

STEP 04:

COUNT(*)	BRANCH
3	ECE
3	MCA.

ANS:

④ SELECT COUNT(*), BRANCH.
① FROM STU
② GROUP BY BRANCH.
③ HAVING COUNT(*) >2 ;

ORDER

① FROM
② WHERE - RBR
③ GROUP BY - RBR
④ HAVING - GBG
⑤ SELECT - GBG

1) WAQTD name of the emp. which are repeated.

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

(2) WAQTD JOB of emp. if the job contains more than 3 emp. who is having a in their names.

```
SELECT JOB
FROM EMP
WHERE ENAME LIKE '%A%'  GROUP BY JOB
HAVING COUNT(JOB) > 3.
```

(3) WAQTD max, avg sal of each dept if the max sal greater than avg sal.

```
SELECT MAX(SAL), AVG(SAL), DEPTNO
FROM emp
WHERE MAX(SAL)
GROUP BY DEPTNO
HAVING MAX(SAL) > AVG(SAL)
```

20/APR/24

⇒) ORDER BY :

- It is used to sort the data based on ascending or, descending order
- If you want to sort the data in ascending order, we have to use "ASC" keyword.
- If we want to display in descending order we have to use "DESC" keyword.
- If we are not using "ASC", "DESC". Then by default, it will display the data in ascending order.

ORDER BY : SORT
↓
ASC, DESC
↓
BY DEFAULT ASC

Q. WAQTD NAME OF THE STU IN ASCENDING ORDER:

STEP 01: STU

SID	SNAME	PER.
101	A	90
102	B	85
103	A	90
104	B	82
105	C	93

STEP 02:

SNAME
A
B
A
B
C

STEP 03:

SNAME
A
A
B
B
C

→ ANS:

```
SELECT SNAME
FROM STU
ORDER BY SNAME ASC;
```

Syntax:

```
SELECT GROUP_FUNCTION/COL_NAME/ EXP.
FROM TABLE_NAME
[WHERE <FILTER_CONDITION>]
[GROUP BY COL_NAME/exp]
[HAVING <GROUP_FILTER_CONDITIONS>]
  ORDER BY COL_NAME DESE;
```

ORDER

1. FROM
2. WHERE - RBR
3. GROUP BY - RBR
4. HAVING - GBG
5. SELECT - GBG
6. ORDER BY - GBG

→ Order by will execute after execution of select clause
→ Order by is only the clause will execute after execution of select clause.
→ Order by will execute group by group if group by clause is present. Other wise it will execute "Row by Row".

22/04/24

Q: WAQTD along with their annual salary and display the data in ascending order by using annual salary.

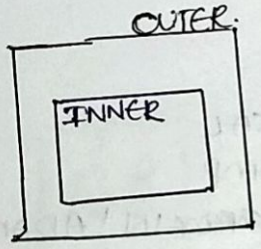
```
SELECT emp.*, SAL*12
FROM emp
ORDER BY SAL*12;
```

Q: WAQTD ename, job, mgr of the emp. and sort the data in descending order. by using job., ename in asc.
(job → index)

```
SELECT ENAME, JOB, mgr
FROM emp.
ORDER BY JOB DESC, ENAME;
```

⇒ SUB-QUERY:

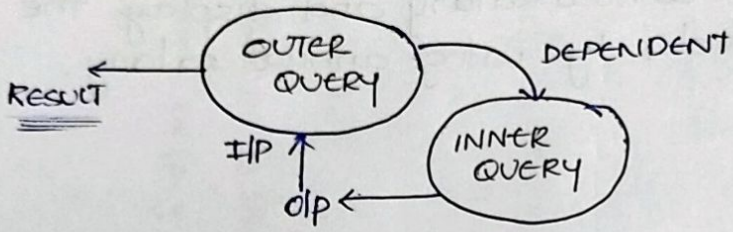
A query written inside a query it known as subquery.



- Subquery contains two queries
  (i) Outer Query.
  (ii) Inner Query.

order of execution:

→ Inner query will execute first and it will produce an o/p
→ The o/p of an inner query will be given as input for the outer query.
→ Now the outer query will execute and it will produce the final result.
→ Outer queries depending on inner query.



NOTE:

- Whenever the question contains indirect condition at that time we will use sub queries.

Case(i):

whatever the data we want to display, whatever the condition we want to satisfy. both are belongs to same table.

Q' WAQTD name of the emp who is getting sal more than 'ADAMS'.

```
SELECT ENAME
FROM EMP
WHERE SAL > (SELECT SAL
             FROM EMP
             WHERE ENAME IN 'ADAMS');
```

Case (ii):

Whatever the data we want to display, whatever the condition we have to satisfy both are present in two different tables.

Q' WAQTD name of the subject.

Q' WAQTD subject name of SUM1?

STEP 01:

STU

SID	SNAME	SUBID
101	SUM1	S1
102	SRINU	S1
103	PRASHU	S2
104	ANI	S3

STEP 02:

SID	SNAME	SUBID
101	SUM1	S1

STEP 04:

SUB

SUBID	SUBNAME
S1	SQL
S2	MT
S3	JAVA
S4	AT

STEP 05:

SUBID	SUBNAME
S1	SQL

STEP 06:

SUBNAME
SQL } RESULT

SUBID
S1. } o/p of Inner Query.

ANS:
```
SELECT SUBNAME
FROM SUB
WHERE SUBID IN (SELECT SUBID
                FROM STU
                WHERE SNAME IN 'SUM1');
```

Q. WAQTD name of the emp. who is working in new york location.

```
SELECT ENAME
FROM EMP
WHERE DEPTNO IN (SELECT DEPTNO
                 FROM DEPT
                 WHERE LOC IN 'NEW.YORK');
```

Q. WAQTD loc of the emp. who is working in dept 30.

```
SELECT LOC
FROM DEPT
WHERE DEPTNO IN (SELECT DEPTNO
                 FROM EMP
                 WHERE DEPTNO IN 30);
```

Q. WAQTD name of the emp. If the emp is getting sal more than miller. and he is working in chicago location.

```
SELECT ENAME
FROM EMP
WHERE SAL > (SELECT SAL
             FROM EMP
             WHERE ENAME IN 'MILLER') AND DEPTNO IN
                                      (SELECT DEPTNO
                                       FROM DEPT
                                       WHERE LOC IN 'CHICAGO');
```

Q. WAQTD name of the dept. If the dept is having more than 3 emp.

~~SELECT ENAME~~
~~FROM DEPT~~
~~WHERE COUNT(*) > 3 AND DEPTNO IN~~

```
SELECT DNAME, COUNT(*)
FROM DEPT
WHERE DEPTNO IN (SELECT DEPTNO. FROM EMP
                 GROUP BY DEPTNO
                 HAVING COUNT(DEPTNO) > 3);
```

~~GROUP BY DEPTNO~~

Q) WAQTD name of the dept. If the dept is having more than 3 emp.

```
SELECT DNAME.
FROM DEPT
WHERE DEPTNO IN (SELECT DEPTNO, COUNT(DEPTNO)
                 FROM emp
                 GROUP BY DEPTNO
                 HAVING COUNT (DEPTNO) > 3 );
```

25-APR-24

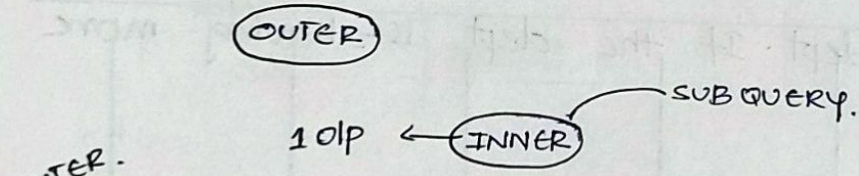
⇒ Types of Sub-Queries:

We are having two types of sub queries.

1) Single row sub query.
2) Multi row sub query.

o Single row sub query:

If the sub query returns exactly one value as an o/p, is known as single row sub query.



⇒ Example

```
OUTER.
SELECT ENAME.
FROM emp
WHERE SAL (>) ( SELECT SAL
                FROM emp
                WHERE ENAME = 'SMITH' );
```

3000

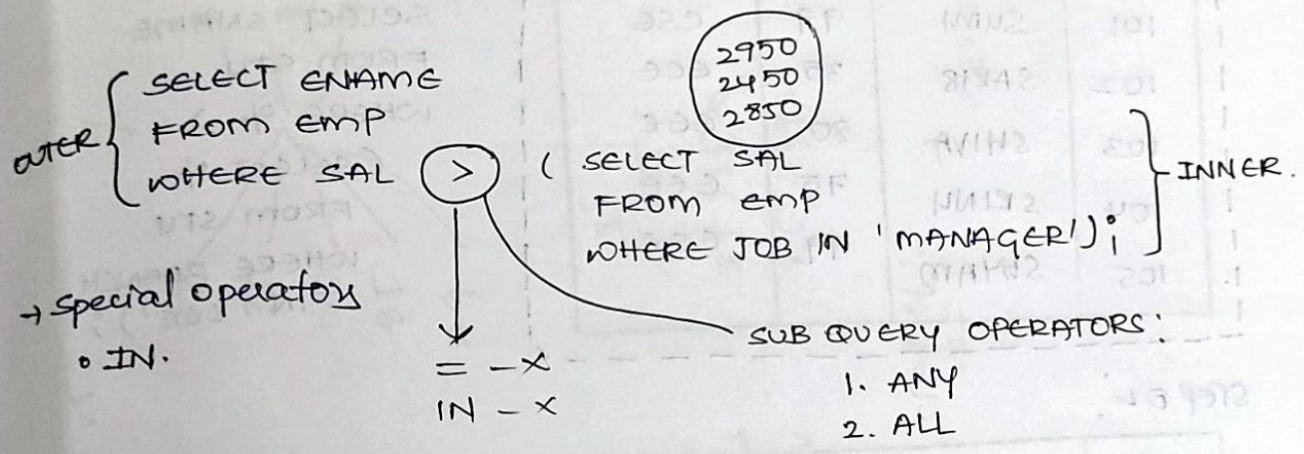
INNER.

1-o/p.

→ Comparision-operator.
→ Relational-operator.
→ Special-operator.

• whenever we are using single row sub query. To compare the values we can use normal operators as well as special operators.
• The above sub query is written in exactly one value as an output. So it will be considered as a S.R.S.Q.

→ Multi row Sub Query:
If the sub query. returns more than one value. as an output is known as multi-row sub query.
• whenever we are using multirow sub query. To comp -are the values we have to use special operators.



1) Here, directly. it's not possible to use relational operators.
2) Here, In operator not possible to use, we have to use relational operator.
3) whenever we want to use relational operators in M.R. sub query., we have to use. sub query operator. along with relational operator.

⇒ SUB QUERY OPERATOR:
• We are having 4 types of operators:
1) ALL
2) ANY
3) EXIST
4) NOT EXIST

ALL - Operator:

'ALL' is a sub query operator used along with relat-ional operators to compare the values.

'ALL' operator returns true if all the conditions has been satisfied.

Ex: WAGTD. name of the student if he is having '%' more than all EEE students.

STEP 01:

STU:

SID	SNAME	%	BRANCH
101	SUMI	99	CSE
102	SAKIB	35	ECE
103	SHIVA	80	EEE
104	SRINU	75	EEE
105	SHYAM	99	EEE

ANS:

```
SELECT SNAME
FROM STU
WHERE % >
   (SELECT %
    FROM STU
    WHERE BRANCH
      IN 'EEE');
```

STEP 02:

SID	SNAME	%	BRANCH
103	SHIVA	80	EEE
104	SRINU	75	EEE
105	SHYAM	99	EEE

Step 03:

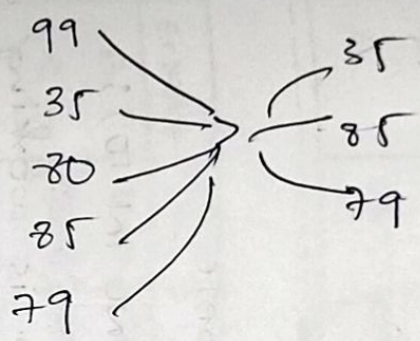
%
35
85
97

} o/p of sub query

step 04:

SID	SNAME	%	BRANCH
101	SUMI	99	CSE
102	SARIB	35	ECE
103	SHIVA	80	EEE
104	SRINU	75	EEE
105	SHYAM	99	EEE

STEP 05 :-



99

99 > 35 ✓

99 > 85 ✓

99 > 79 ✓

35

35 > 35 ✗

35 > 85 ✗

35 > 79 ✗

80

80 > 35 ✓

80 > 85 ✗

80 > 79 ✓

85

85 > 35 ✓

85 > 85 ✗

85 > 79 ✓

79

79 > 35 ✓

79 > 85 ✗

79 > 79 ✗

STEP 06 :-

SID	SNAME	%	BRANCH
101	SUMI	99	CSE

SNAME
SUMI
} RESULT

26/APR/24

Q: WAQTD ename, sal of the emp. If the emp are getting salary more than all the manager.

```
SELECT ENAME, SAL
FROM EMP
WHERE SAL > ALL ( SELECT SAL
                  FROM EMP
                  WHERE JOB IN 'MANAGER');
```

Q: WAQTD all the details of the emp. If the emp is hired before the emp working in dept 10.

```
SELECT EMP.*
FROM EMP
WHERE HIREDATE < ALL( SELECT HIREDATE
                      FROM EMP
                      WHERE DEPTNO IN 10);
```

Q: WAQTD LOC of the emp. if the emp is working as a salesman, analyst.

```
SELECT ~~LOC~~          SELECT LOC
FROM ~~DEPT~~           FROM DEPT
CO                      WHERE DEPTNO =ALL(SELECT DEPT.NO
                                          FROM EMP
                                          WHERE JOB IN ('SALESMAN',
                                                        'ANALYST'));
```

wrong → Any op.

⇒ Any-Operator:

~~If the~~

It is a sub query operator used along with the relational operator to compare the values.

- Any operator returns true if any one of the condition has been satisfied.

Q: WAQTD name of the emp. If the emp. is getting sal morethan any one of the salesman.

① emp

EMPNO	ENAME	SAL	JOB
101	A	5000	MANAGER
102	B	3000	SALESMAN
103	C	2500	SALESMAN
104	D	3200	ANALYST
105	E	2800	ANALYST

ANS:

```
SELECT ENAME
FROM EMP
WHERE SAL > ANY (SELECT SAL
                 FROM EMP
                 WHERE JOB
                 IN 'SALESMAN');
```

②

EMPNO	ENAME	SAL	JOB
102	B	3000	SALESMAN
103	C	2500	SALESMAN

③

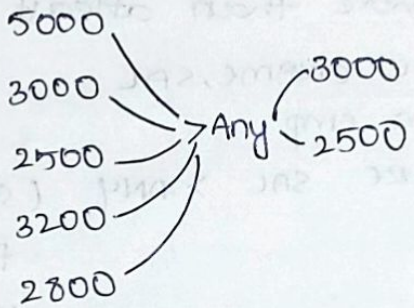
SAL
3000
2500

} Output of Inner Query.

④ emp

EMPNO	ENAME	SAL	JOB
101	A	5000	MANAGER
102	B	3000	SALESMAN
103	C	2500	SALESMAN
104	D	3200	ANALYST
105	E	2800	ANALYST

⑤



5000
5000 > 3000 ✓
5000 > 2500 ✓

3000
3000 > 3000 ×
3000 > 2500 ✓

2500
2500 > 3000 ×
2500 > 2500 ×

3200
3200 > 3000 ✓
3200 > 2500 ✓

2800
2800 > 3000 ×
2800 > 2500 ✓

⇒ TABLE

P.T.O

Q: WAQTD all the details of the emp. if the emp. getting comm more than all the emp. and they are hired before the any. one of the president

```
SELECT *
FROM emp
WHERE HIREDATE < ANY (SELECT HIREDATE
                      FROM emp
                      WHERE JOB IN 'PRESIDENT')
      AND COMM > ALL (SELECT COMM FROM EMP);
```

29/APR/24

Q: WAQTD name of the emp who's getting max salary.

```
SELECT ENAME
FROM EMP
WHERE SAL IN (SELECT MAX(SAL)
              FROM emp);
```

Q: WAQTD name of the emp. who's getting min salary.

```
SELECT ENAME
FROM EMP
WHERE SAL IN (SELECT MIN(SAL)
              FROM EMP);
```

⇒ MAX & MIN:

Q: WAQTD 2nd max salary of the emp.

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

STEP 01:

EMP.

ID	SAL
1	1000
2	1000
3	1500
4	3000
5	3000
6	4000

STEP 02:

MAX(SAL)
4000

STEP 03:

EMP.

ID	SAL
1	1000
2	1000
3	1500
4	3000
5	3000
6	4000

STEP 04:

SAL < 4000

ID	SAL
1	1000
2	1000
3	1500
4	3000
5	3000

STEP 05:

MAX(SAL)
3000

} RESULT

Q: WAQTD 2nd min sal of emp table

```
SELECT MIN(SAL)
FROM emp
WHERE SAL > (SELECT MIN(SAL)
             FROM emp);
```

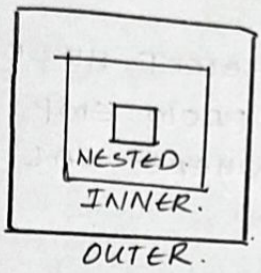
Q: WAQTD 2nd max hiredate.

```
SELECT MAX(HIREDATE)
FROM emp
WHERE HIREDATE < (SELECT MAX(HIREDATE)
                  FROM emp);
```

→ NESTED SUB QUERIES:

A query ~~returns~~ written inside a sub query (inner query) is known as nested sub query..

→ We can nest upto 255 queries.



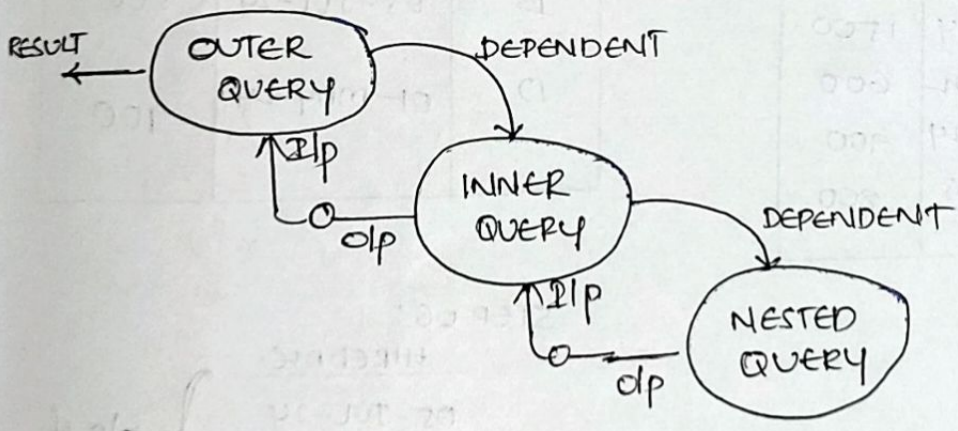
→ Order of Execution!

Nested query will execute first and it will produce an output. The output of nested query is given as input to the inner query..

→ Now inner query will execute and it will produce an output. The output of an inner query will be given as input to the outer query.

→ Now outer query will execute and produce the result.

→ Outer query is depending on inner query.
Inner query is depending on nested query.



Example:

Q: WAQTD. names of the emp. If the emp are hired who are getting sal more than smith.

```
SELECT ENAME
FROM emp
WHERE HIREDATE >ALL ( SELECT HIREDATE
                      FROM EMP
                      WHERE SAL > (SELECT SAL
                                   FROM emp
                                   WHERE ENAME
                                   IN 'SMITH'));
```

STEP 01:

emp:

ENAME	HIREDATE	SAL
A	05-DEC-24	200 ← ✗
B	05-JUL-24	1500 ✗
C	24-JUN-22	600 ✗
D	01-MAY-24	900 ✗
SMITH	19-DEC-23	800 ✓

STEP 02:

ENAME	HIREDATE	SAL
SMITH	19-DEC-23	800

STEP 03:

SAL
800

} o/p of Nested loop.

STEP 04:

emp:

ENAME	HIREDATE	SAL
A	05-DEC-24	200
B	05-JUL-24	1500
C	24-JUN-22	600
D	01-MAY-24	900
SMITH	19-DEC-23	800

STEP 05:

ENAME	HIREDATE	SAL
B	05-JUL-24	500
D	01-MAY-24	900

STEP 06:

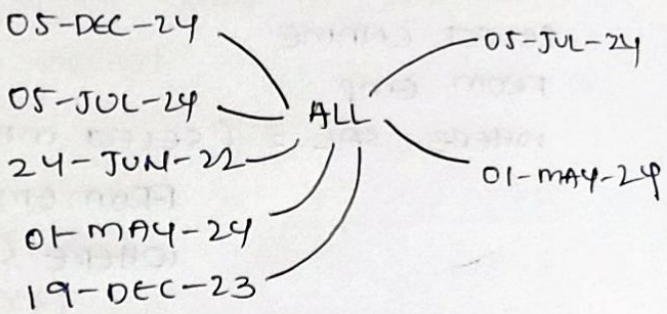
HIREDATE
05-JUL-24
01-MAY-24

} o/p of Inner & query

STEP 07:

ENAME	HIREDATE	SAL
A	05-DEC-24	200
B	05-JUL-24	1500
C	24-JUN-22	600
D	01-MAY-24	900
SMITH	19-DEC-23	800

STEP 8.



STEP 08:

ENAME	HIREDATE	SAL
A	05-DEC-24	200

STEP 09:

ENAME
A

} RESULT

30|APR|24

Q: WAQTD 3rd max sal of the emp.

```
SELECT MAX(SAL) ③
FROM EMP
WHERE SAL < (SELECT MAX(SAL)   ②
             FROM EMP
             WHERE SAL < (SELECT MAX(SAL)   ①
                          FROM EMP));
```

3rd maximum

2nd maximum

1st maximum

Q: WAQTD name of the emp who are hired second.

```
SELECT ENAME
FROM EMP
WHERE HIREDATE = (SELECT MIN(HIREDATE)
                  FROM EMP
                  WHERE HIREDATE > (SELECT MIN(HD)
                                    FROM EMP));
```

Q: WAQTD name of the emp who's getting 2nd max sal and it must be 2nd min sal.

```
SELECT ENAME
FROM EMP
WHERE SAL = [(SELECT MAX(SAL)
              FROM EMP
              WHERE SAL < (SELECT MAX(SAL)
                           FROM EMP))] AND
~~WHERE~~ SAL = (SELECT MIN(SAL)
                 FROM EMP
                 WHERE > (SELECT MIN(SAL)
                          FROM EMP));
```

Q: WAQTD name of the emp who's getting 2nd max and 4th max sal.

```
SELECT ENAME
FROM EMP
WHERE SAL = (SELECT MAX(SAL)
             FROM EMP
             WHERE SAL < (SELECT MAX(SAL)
                          FROM EMP
                          WHERE SAL < (SELECT MAX(SAL)
                                       FROM EMP
                                       WHERE
                                       SAL
                                       (SELECT MAX(SAL)
                                        FROM EMP)))
OR SAL = (SELECT MAX(SAL)
          FROM EMP
          WHERE SAL < (SELECT MAX(SAL)
                       FROM EMP));
```

01/05/24

(1) WAQTD name of the employee who is working in the same location of ward.

```
SELECT ENAME
FROM EMP
WHERE DEPTNOIN ( SELECT DEPTNO
                 FROM DEPT
                 WHERE LOC IN ( SELECT LOC
                                FROM DEPT
                                WHERE ENAME
                                IN 'WARD'));
```

⇒ EMP manager Relationship.

- It is used to identify employee details and managers details.
- we are having 2 cases:
  - (i) Case 1
  - (ii) Case 2.

Case 1:

we are having employee details we are going to identify the manager detail.

Ex:

(1) WAQTD manager name of the SMITH.

```
SELECT ENAME MNAME
FROM EMP
WHERE ._____
```

WHERE EMPNO IN (SELECT MGR
FROM emp
WHERE ENAME IN 'SMITH');

(2)

STEP 01:

emp

EMPNO	ENAME	MGR
101	SMITH	102
102	ALLEN	103
103	SCOTT	104
104	KING	

STEP 02

EMPNO	ENAME	MGR
101	SMITH	102

(3)

STEP 03:

MGR
102

} o/p of innerquery.

STEP 04:

emp:

EMPNO	ENAME	MGR
101	SMITH	102
102	ALLEN	103
103	SCOTT	104
104	KING	

(4)

STEP 05:

EMPNO	ENAME	MGR
102	ALLEN	103

STEP 06:

MNAME
ALLEN

} o/p of final result.

(2) WAQTD manager name of the employee who is working as a clerk.

```
SELECT ENAME MNAME
 FROM emp
WHERE EMPNO IN (SELECT MGR
                  FROM EMP
                 WHERE JOB IN 'CLERK');
```

(3) WAQTD manager name of the emp. if the emp are getting salary more than 2000.

```
SELECT ENAME MNAME
  FROM emp
WHERE EMPNO IN (SELECT MGR
                  FROM EMP
                 WHERE SAL > 2000);
```

(4) WAQTD manager name of the emp. if the emp are getting more than smith.

```
SELECT ENAME, MNAME
  FROM emp
 WHERE EMPNO IN (SELECT MGR
                   FROM EMP
                  WHERE SAL > (SELECT SAL
                                 FROM EMP
                                WHERE ENAME
                                 IN 'SMITH'));
```

Q: WRITE manager's manager name of SCOTT.

```
SELECT ENAME MNAME
FROM EMP
WHERE EMPNO IN (SELECT MGR
                FROM EMP
                WHERE EMPNO IN
                      (SELECT MGR
                       FROM EMP
                       WHERE ENAME IN
                             'SCOTT'));
```

Case 2:
We are having manager detail. and we are going to identify the employee details.

(1) WRITE name of the employee who is reporting to ALLEN.

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

STEP 01:

EMP

EMPNO	ENAME	MGR
101	SMITH	102
102	ALLEN	103
103	SCOTT	104
104	KING	

STEP 02:

EMPNO	ENAME	MGR
102	ALLEN	103

STEP 03:

EMPNO

102

} o/p of inner query.

STEP 04:

EMP

EMPNO	ENAME	MGR.
101	SMITH	102
102	ALLEN	103
103	SCOTT	104
104	KING	

STEP 05:

EMPNO	ENAME	MGR
101	SMITH	102

STEP 06:

ENAME

SMITH.

(2) WRITE NAME OF THE EMP WHO ARE REPORTING TO KING

```
SELECT ENAME
 FROM emp
WHERE MGR IN (SELECT EMPNO
                FROM EMP
               WHERE ENAME IN 'KING');
```

02/MAY/24

⇒ JOINS:

- It is used to read the data from multiple tables simultaneously.
- We are having 5 types of joins. They are:
  1) CARTESIAN JOIN / CROSS JOIN
  2) INNER JOIN / EQUI JOIN
  3) OUTER JOIN / -
     (i) LEFT OUTER JOIN
     (ii) RIGHT OUTER JOIN
     (iii) FULL OUTER JOIN
  4) NATURAL JOIN
  5) SELF JOIN

⇒ CARTESIAN JOIN:

- whenever there is no relation b/w the tables at that time we are going to use cartesian join.
- whenever we are using cross join each and every record of table one (T1) will be merged all the records of table two (T2). ↑ with

Syntax:

ORACLE: SELECT */COL_NAME
FROM T1, T2;

ANSI: SELECT */COL_NAME
FROM T1 CROSS JOIN T2;

- To know the no.of.records, no.of.columns present in the cartesian join table:

TOTAL COLUMNS PRESENT CARTESIAN TABLE:

NO.OF.COL = NO.OF.COL PRESENT IN T1 +
NO OF. COL PRESENT IN T2.

TOTAL RECORDS PRESENT IN CARTESIAN TABLE

TOTAL NO. OF. RECORDS = NO. OF. RECORDS PRESENT IN T1 x
NO. OF. RECORDS PRESENT IN T2.

NOTE:

1) To know the total columns we have to do the summation of Table 1 columns, Table 2 columns.

2) To know the total records we have to do multiplication of Table 1 records, Table 2 records.

Eg: WAQTD ENAME, LOC OF THE EMP.

STEP 01:

emp

EMPNO	ENAME	SAL
101	A	1500
102	B	1100
103	C	800

DEPT

DEPTNO	LOC
10	D1
20	D2

CARTESIAN JOIN TABLE:

EMPNO	ENAME	SAL	DEPTNO	LOC
101	A	1500	10	D1
101	A	1500	10	D1
102	B	1100	20	D2
102	B	1100	20	D2
103	C	800	30	D3
103	C	800	30	D3

STEP 02:

ENAME	LOC
A	D1
A	D1
B	D2
B	D2
C	D3
C	D3

} RESULT

Q: WAQTD ENAME, SAL, DEPTNO ALONG WITH DNAME OF EMP.

```
SELECT ENAME, SAL, DEPTNO, DNAME
FROM EMP, DEPT;
```

NOTE: DEPTNO col present in emp., dept table it will not execute, it will display one error. The error is 'COLUMN AMBIGUOUSLY DEFINE".

To overcome this error. we have to write table name along with col. name.

i.e.,
```
SELECT ENAME, SAL, EMP.DEPTNO, DNAME
FROM EMP, DEPT;
```

Q: WAQTD LOC, ENAME, DEPTNO OF THE EMP:

```
SELECT LOC, ENAME, EMP.DEPTNO
FROM DEPT, EMP;
```

Q: WAQTD ENAME, LOC, DNAME, SAL OF THE EMP.

```
SELECT ENAME, LOC, DNAME, SAL
FROM DEPT, EMP;
```

INNER-JOIN → Equi Join (Both sides must be same values) 03/MAY/24

- whenever there is a relation b/w the table we'll use INNER JOIN.
- wherever we are using INNER JOIN it will select only matched data.

Syntax:

ORACLE:
```
SELECT */COL_NAME.
FROM T_N1, T_N2
WHERE <JOINING CONDITION>;
```
↳ CT-N1.Col name = T_N2.Col name)

ANSI: SELECT */COL-NAME.
FROM T-N1 INNER JOIN T-N2.
ON ~~WHERE~~ < JOINING CONDITION > ;

Q: WAQTD ENAME, LOC OF THE EMP.

STEP 01:

EMP

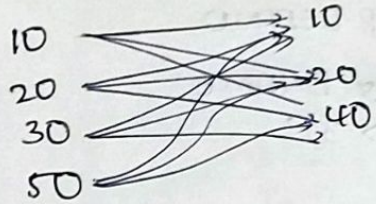
EMPNO	ENAME	DEPTNO
101	A	10
102	B	20
103	C	30
104	D	50

DEPT

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

STEP 2:

EMP.DEPTNO DEPT.DEPTNO



10	20	30	50
✓	✗	✗	✗
✗	✓	✗	✗
✗	✗	✗	✗
1	1	0	0

INNER JOIN TABLE:

EMPNO	ENAME	DEPTNO	DEPTNO	DNAME	LOC
101	A	10	10	D1	L1
102	B	20	20	D2	L2

STEP 03:

ENAME	LOC
A	L1
B	L2

} RESULT

Q: WAQTD ENAME, DNAME OF THE EMP.

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

Q: WAQTD all the details of the emp along with dept details. If they are working as a manager in the new york location.

```
SELECT EMP.*, DEPT.*
FROM EMP, DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO AND JOB IN
                   'MANAGER' AND LOC IN
                   'NEW YORK';
```

Q: WAQTD DNAME OF THE EMP, IF THE DNAME CONTAINS MORE THAN 3 EMP:

```
SELECT DNAME.
FROM DEPT, EMP.
WHERE DEPT.DEPTNO = EMP.DEPTNO
GROUP BY ~~ENAME~~. ~~DEPT~~ DNAME
HAVING (COUNT(ENAME) > 3);
```

→ OUTER JOIN):

- It is used to get unmatched data from table.
- In OUTER JOIN WE ARE HAVING 3 TYPES.
  - (i) LEFT OUTER JOIN
  - (ii) RIGHT OUTER JOIN
  - (iii) FULL OUTER JOIN.

→ LEFT OUTER JOIN:

- It is used to retrive unmatched data from the left side table along with matched data.

Syntax:

```
ORACLE: SELECT */COL_NAME
        FROM T_N1, T_N2 {
        WHERE <JOINING_CONDITION>;
```

ANSI:

```
SELECT */COL_NAME
FROM T_N1 LEFT [OUTER] JOIN T_N2
ON <JOINING_CONDITION>;
```

5) WAQTD ENAME, ~~JOIN~~ LOC OF THE EMP BY USING LEFT JOIN.

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

STEP 01:

EMP

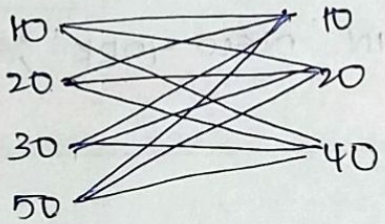
EMPNO	ENAME	DEPTNO
101	A	10
102	B	20
103	C	30
104	D	50

DEPT

DEPTNO	DNAME	LOC.
10	D1	L1
20	D2	L2
40	D3	L3

STEP 02:

EMP.DEPTNO = DEPT.DEPTNO



10	20	30	50
✓	✗	✗	✗
✗	✓	✗	✗
✗	✗	✗	✗
1	1	0	0
MATCHED		ON MATCHED	

LEFT JOIN TABLE

EMPNO	ENAME	DEPTNO	DEPTNO	DNAME	LOC
101	A	10	10	D1	L1
102	B	20	20	D2	L2
103	C	30			
104	D	50			

STEP 03!

ENAME	LOC	
A	L1	} RESULT
B	L2	
C		
D		

04|MAY|24

1) WAQTD ENAME, DNAME, HIREDATE OF THE OF THE EMP BY USING LEFT OUTER JOIN.

```
SELECT ENAME, DNAME, HIREDATE
FROM EMP, DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO(+);
```

2) WAQTD UNMATCHED DATA OF EMP TABLE IF THE EMP IS WORKING IN NEW YORK LOCATION:

```
SELECT *
FROM EMP, DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO(+) AND
                   LOC IN 'NEW YORK';
```

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

(ii) RIGHT OUTER JOIN:

- It is used to retrive the unmatched data from the right side table along with the unmatched data.

Syntax:

```
SELECT */COL_NAME
FROM T_N1, T_N2
WHERE <JOINING CONDITION>;
```

ORACLE

```
SELECT */COL_NAME
FROM T_N1 RIGHT [OUTER] JOIN
T_N2
ON <JOINING CONDITION>
```

ANSI

Q: WAQTD ename, loc of the emp by using Right Outer Join.

STEP 01:

EMP

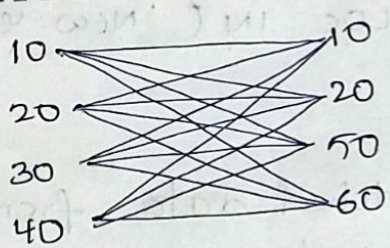
EMPNO	ENAME	DEPTNO
101	A	10
102	B	20
103	C	30
104	D	40

DEPT

DEPTNO	LOC
10	L1
20	L2
50	L3
60	L4

STEP 02:

EMP.DEPTNO = DEPT.DEPTNO



10	20	50	60
✓	×	×	×
×	✓	×	×
×	×	×	×
×	×	×	×
1	1	0	0

MATCHED — UNMATCHED.

STEP 03:

EMPNO	ENAME	DEPTNO	DEPTNO	LOC
101	A	10	10	L1
102	B	20	20	L2
			50	L3
			60	L4

STEP 03:

ENAME	LOC	
A	L1	RESULT
B	L2	
	L3	
	L4	

Q: WAQTD UNMATCHED DATA OF DEPARTMENT TABLE. IF THE EMP ARE NOT WORKING AS A MANAGER.

```
SELECT *.
FROM EMP, DEPT
WHERE EMP.DEPTNO (*) = DEPT.DEPTNO AND
                JOB NOT IN 'MANAGER';
```

Q: WAQTD ENAME, DNAME, HIREDATE, LOC OF THE EMP. WHO ARE WORKING IN THE NEW YORK, CHICAGO LOCATIONS BY USING RIGHT OUTER JOIN

```
SELECT ENAME, DNAME, HIREDATE, LOC.
  FROM EMP, DEPT
WHERE EMP.DEPTNO(+) = DEPT.DEPTNO AND
                LOC IN ('NEW YORK', 'CHICAGO');
```

(iii) FULL OUTER JOIN:

→ It is used to get unmatched data from both the tables along with matched data.

Syntax:

```
SELECT */COL_NAME
  FROM T_N1 FULL [OUTER] JOIN T_N2
    ON <JOINING_CONDITION>;
```

Q: WRITE UNMATCHED DATA OF EMP, DEPT TABLE.

Q: WRITE ENAME, LOC OF THE ~~EMPTY~~ EMP BY USING FULL OUTER JOIN.

```
SELECT ENAME, LOC
FROM EMP, DEPT
~~WHERE~~ ON EMP.DEPTNO = DEPT.DEPTNO;
```

STEP 01:

EMP:

EMPNO	ENAME	DEPTNO
101	A	10
102	B	20
103	C	30
104	D	40

DEPT:

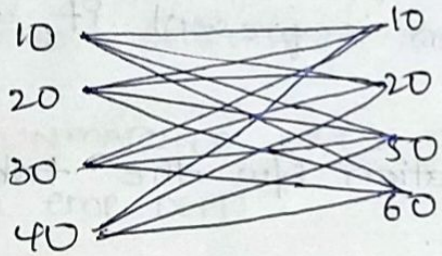
DEPTNO	LOC
10	L1
20	L2
50	L3
60	L4

STEP 02:

LEFT

EMP.DEPTNO = DEPT.DEPTNO

RIGHT



40	30
×	×
×	×
×	×
×	×
0	0

UNMATCHED.

10	20	50	60
✓	×	×	×
×	✓	×	×
×	×	×	×
×	×	×	×
1	1	0	0

MATCHED UNMATCHED

FULL - JOIN!

EMPNO	ENAME	DEPTNO	DEPTNO	LOC
101	A	10	10	L1
102	B	20	20	L2
103	C	30	-	-
104	D	40	-	L3
			50	L4
			60	

STEP 03!.

ENAME	LOC
A	L1
B	L2
C	-
D	-
	L3
	L4

} RESULT

(iv) NATURAL - JOIN!

- whenever we dont have the structure of the table at that time, we are going to use natural join.
- whenever we are using natural join, Pn between the table if the relation is present. It will acts like inner join.
- If there is no relation b/w the table. It will acts like. cartesian join.

Syntax! SELECT * / COL_NAME.
FROM T_N1 NATURAL JOIN T_N2 ;

Ex: WAQTD. DETAILS OF STUDENT, EMP TABLES.

```
SELECT *.
FROM STU, NATURAL JOIN EMP;
```

NOTE!
The above query will looks like easterian join. Because, ther's no relation present b/w the emp and student table.

Q: WAQTD DETAILS OF THE EMP. ALONG WITH THEIR DEPT DETAILS.

```
SELECT *.
FROM EMP NATURAL JOIN DEPT;
```

NOTE!
The above query will act like INNER JOIN. Because, ther's relation present b/w the employee and department tables.

06|MAY|24

Q: WAQTD ONLY UNMATCHED DATA FROM LEFT SIDE TABLE:

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

Q: WAQTD ONLY UNMATCHED DATA FROM RIGHT SIDE TABLE:

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

Q: WAQTD ONLY UNMATCHED DATA FROM BOTH THE TABLES:

```
SELECT * FROM EMP, DEPT FULL [OUTER] JOIN DEPT
ON EMP.DEPTNO = DEPT.DEPTNO
WHERE EMP.DEPTNO IS NULL OR DEPT.DEPTNO IS NULL;
```

⇒ SELF-JOIN:

- It is used to join a table by itself.

Syntax:

ORACLE:
```
SELECT *
FROM T_N1, T_N2
WHERE <JOINING_CONDITION>
      (T_N1.COL_NAME = T_N2.COL_NAME)
```

ANSI:
```
SELECT */COL_NAME.
FROM T_N1 JOIN T_N2
ON <JOINING_CONDITION>
[WHERE <FILTER_CONDITION>]
```

Q: WAQTD ~~ENAME~~ EMP. NAME, MANAGER NAME OF THE EMP.

STEP 01:

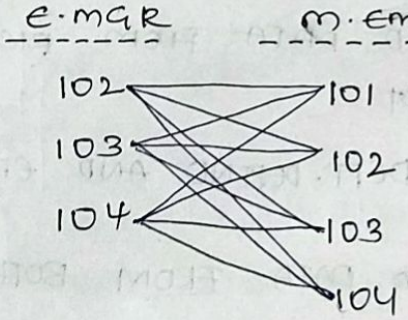
(E) EMP:

EMPNO	ENAME	MGR
101	A	102
102	B	103
103	C	104
104	D	

(M) EMP:

EMPNO	ENAME	MGR
101	A	102
102	B	103
103	C	104
104	D	

STEP 02:



102	103	104
X	X	X
✓	X	X
X	✓	X
X	X	✓
1	1	1

SELF JOIN TABLE:

EMPNO	ENAME	MGR	EMPNO	ENAME	MGR
101	A	102	102	B	103
102	B	103	103	C	104
103	C	104	104	D	

STEP 03:

ENAME	MNAME
A	B
B	C
C	D

} RESULT.

ANSWER!

```
SELECT E.ENAME, M.ENAME MNAME.
  FROM EMP E , EMP M
  WHERE E.MGR =M.EMPNO°,
```

Q. WAOTD ENAME, SAL, MGR NAME, MGR SAL OF THE EMP

```
SELECT E.ENAME, E.SAL , M.ENAME, M.SAL
   FROM EMP E, EMP M
  WHERE E.MGR = M.EMPNO°;
```

Q. WAOTD ENAME , HIREDATE OF EMP. ALONG WITH THEIR MANAGER NAME IF THE EMP IS HIRED BEFORE THEIR MANAGER

```
SELECT E.ENAME , E.HIREDATE , M.ENAME
  FROM EMP E, EMP M
  WHERE E.MGR =M. EMPNO AND
                E.HIREDATE < M.HIREDATE;
```

07/MAY/24

*HOW TO JOIN MULTIPLE TABLES:

⇒ TWO TABLES:

① EMP, DEPT — Emp.DEPTNO = DEPT.DEPTNO.

② EMP E, EMP M — E.MGR = M.EMPNO.

⇒ THREE TABLES:

① EMP E, DEPT, EMP M — E.DEPTNO = DEPT.DEPTNO; E.MGR = M.EMPNO

② EMP E, EMP M, EMP MM — E.MGR = M.EMPNO; M.MGR = MM.EMPNO.

⇒ FOUR TABLES:

EMP E, DEPT D1, EMP M, DEPT D2 — E.DEPTNO = D1.DEPTNO; E.MGR = M.EMPNO; M.DEPTNO = D2.DEPTNO

~~M.MGR = MM.EMPNO~~

ANS: SELECT E.ENAME, M.ENAME, MM.ENAME FROM EMP E, EMP M, EMP MM

ANS: SELECT E.ENAME, M.ENAME, MM.ENAME FROM EMP E, EMP M, EMP MM
WHERE E.MGR = M.EMPNO AND M.MGR = MM.DEPTNO;

STEP 01:

EMP

EMPNO	ENAME	MGR
7369	SMITH	7902
7902	FORD	7566
7566	JONES	7839
7839	KING	

C1

EMP

EMPNO	ENAME	MGR
7369	SMITH	7902
7902	FORD	7566
7566	JONES	7839
7839	KING	

C2

EMP

EMPNO	ENAME	MGR
7369	SMITH	7902
7902	FORD	7566
7566	JONES	7839
7839	KING	

(C1) (C2)

STEP 02:

E.MGR = M.EMPNO

7902 → 7369, 7902
7566 → 7566
7839 → 7839

M.MGR = MM.EMPNO

7566 → 7369
7839 → 7902, 7566, 7839

EMPNO	ENAME	MGR	EMPNO	ENAME	MGR	EMPNO	ENAME	MGR
7369	SMITH	7902	7902	FORD	7566	7566	JONES	7839
7902	FORD	7566	7566	JONES	7839	7839	KING	

(1) LOOTED EMP NAME, MANAGER NAME, MANAGERS MANAGER NAME OF THE EMPLOYEE.

→ FIVE TABLES:

EMP E.
DEPT D1
EMP M
DEPT D2
EMP MM

E.DEPTNO = D1.DEPTNO
E.MGR = M.EMPNO
M.DEPTNO = D2.DEPTNO
M.MGR = MM.EMPNO

STEP 3:

ENAME	MNAME	MMNAME
SMITH	FORD	JONES
FORD	JONES	KING

} RESULT

⑤ WAQTD EMP NAME, MANAGER NAME IF THE EMP. AND MANAGER'S MANAGER BOTH ARE WORKING IN THE SAME LOCATION.

```
SELECT E.ENAME , M.ENAME MNAME, MM.ENAME,
FROM EMP E, DEPT D , EMP M , EMP MM
WHERE E.MGR = M.EMPNO AND M.MGR = MM.EMPNO
                AND E.DEPTNO = MM.DEPTNO;
~~AND~~ E. ~~AND~~ E ~~AND~~ MM.
```