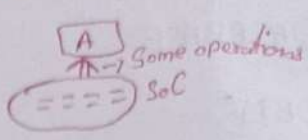
7/8/2023

SQL

STRUCTURED QUERY LAGUAGE

↓

Database



APPLICATION => It is a Set of Codes which we are using to perform Some operation

Types of Application -> We have 3 types of applications

i) Stand-Alone Application
ii) WEB Application
iii) Mobile / Client Sever Application

DINGA | DINGA

STAND-ALONE APPLICATION

=> The Application which are running through offline those applications are called as Stand-Alone Application

=> In this type of Applications the Datas will be Stored in local device or Same device

=> In S-A Application there will be No Storage of in any Database

=> Installation is mandatory in Stand-Alone Application.

WEB Application

=> The Applications which are running through online those type of Applications are called as web Application.

=> In web Application Database is Present

=> In web Application the Datas will be Stored in Global Device

=> Installation is not mandatory in this type of Application

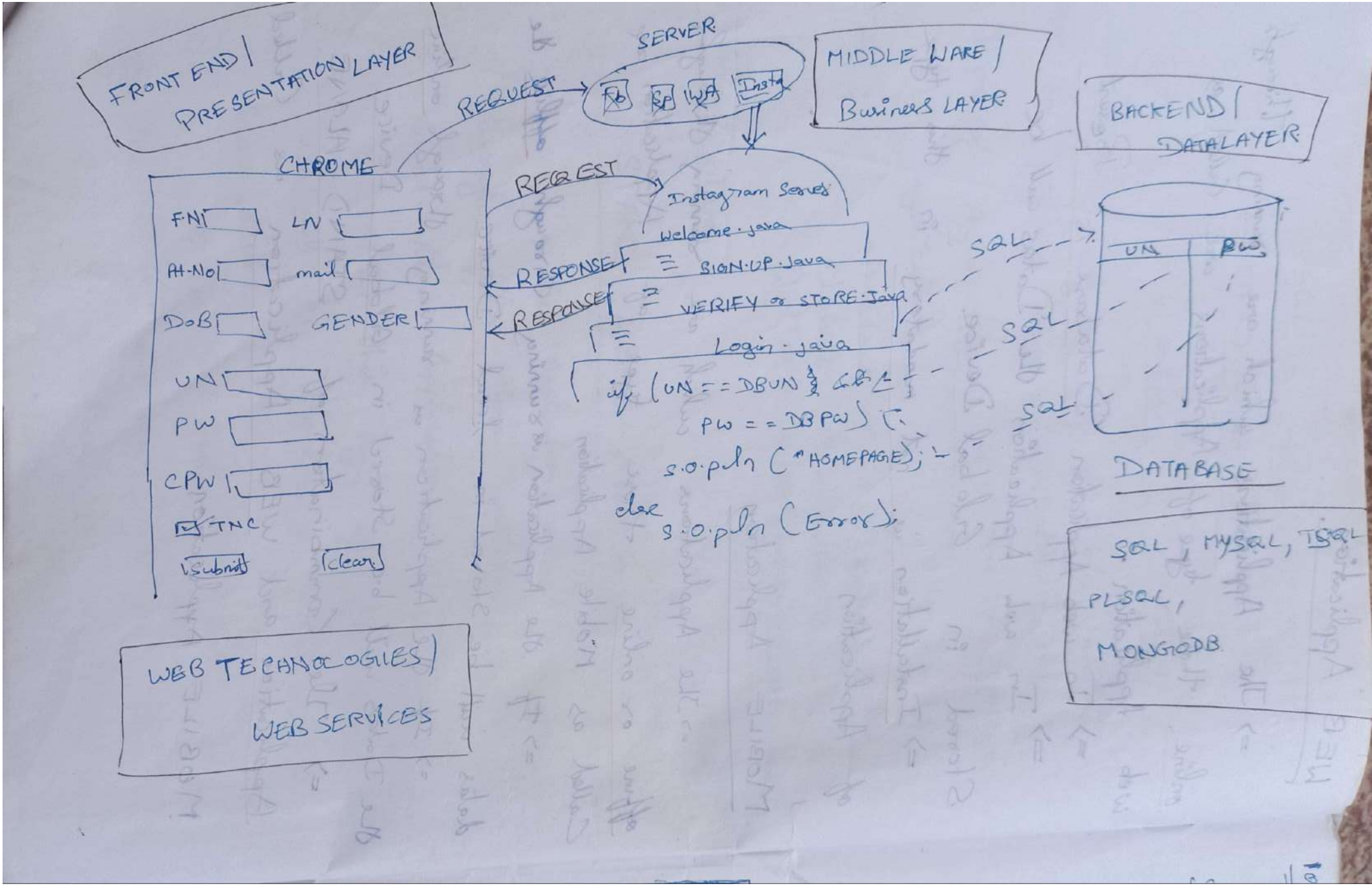
MOBILE Application

=> The Applications which are running through offline or online those type of Application are called as Mobile Application

=> If the Application is running through offline the datas will be stored in local Device

=> If the Application is running through online the Datas will be Stored in Global Device

=> The Combination of STAND-ALONE Application and WEB Application is called MOBILE Application



10/8

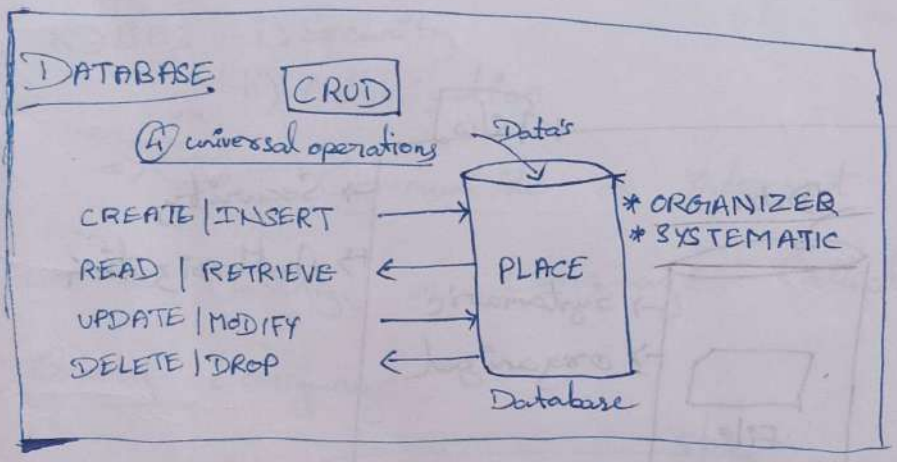
DATABASE

DATA => [Datas are the raw Facts / Facts / Values which Describes Attributes / Properties of an Entity]

eg:

	Attributes / Properties		values / Facts / Raw Facts
LAPTOP	RAM	-	4GB
	ROM	-	1TB
	Process	-	INTEL
Nonliving. object	price	-	25,000
	OS	-	Windows
Entity			

both living & nonliving can be specified



i) Database is a place where we are storing the data's in a Systematic and Organized manner.

ii) The operations perform on the Database are

1.) CREATE | INSERT
2.) READ | RETRIEVE
3.) UPDATE | MODIFY
4.) DELETE | DROP

These operations universally reffered as CRUD Operation

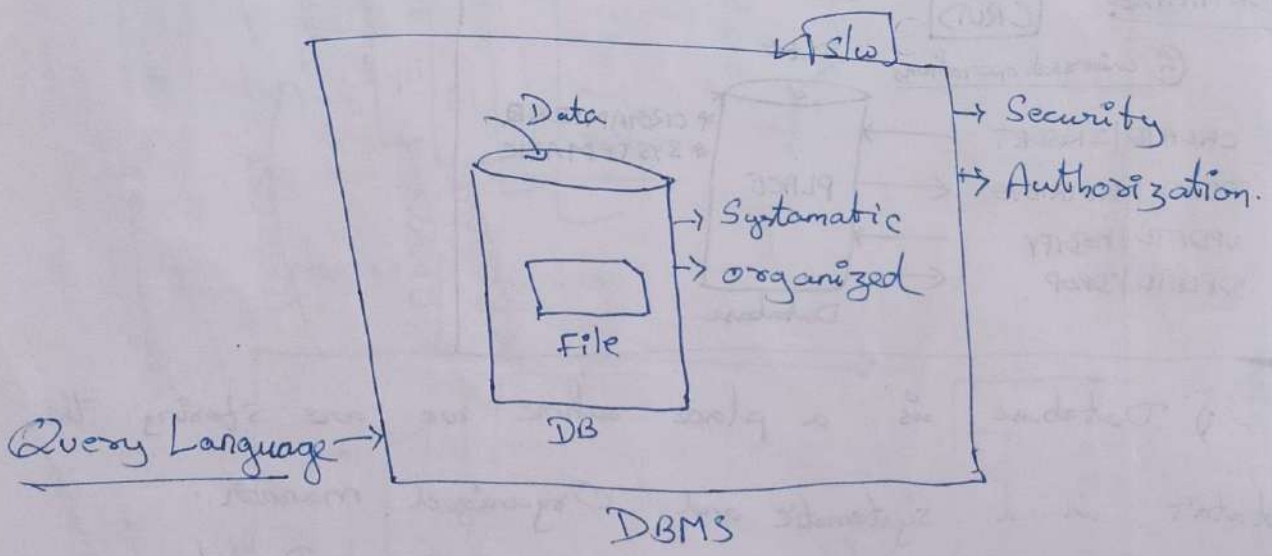
DataBase Management System (DBMS)

=> It is a Software which is used to manage the DataBase.

=> The Two important features Poovided by the DBMS are i) Security
ii) Authorization.

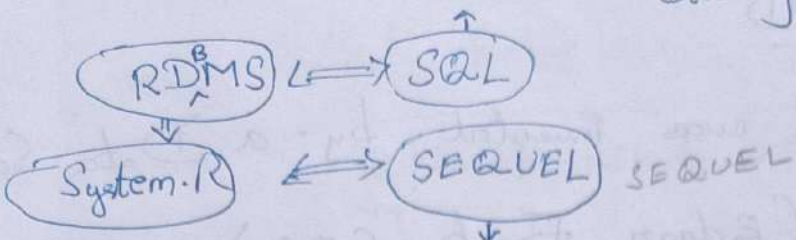
=> In DBMS we are storing the data's in file formate

=> To Communicate or interact with the DBMS we are using a Laanguage called Query Language



Relational DataBase Management System || RDBMS

Structured Query Language



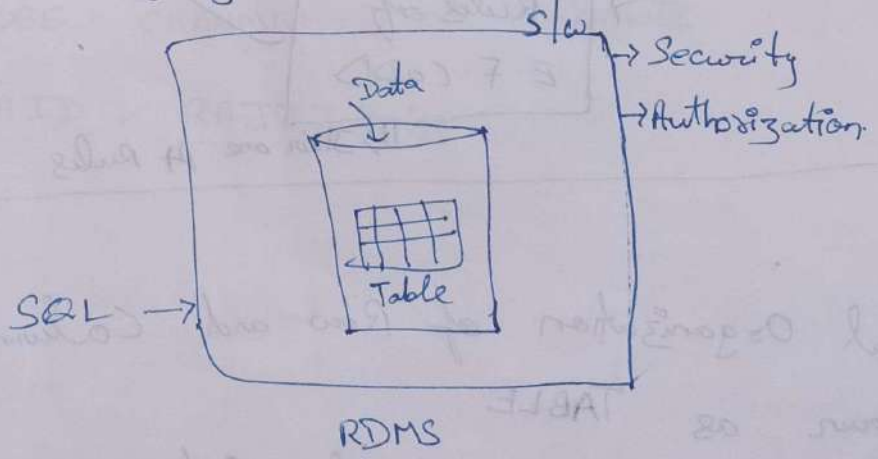
Simple English Query Language

=> It is a type of DBMS Software which is used to manage the Database

=> In RDBMS S/w we are storing the Data in Table formate

=> The two important features Provided by the RDBMS i) Security
ii) Authorization

=> To Communicate or interact with RDBMS S/w we are using a Language Called Structured Query Language.



=> In the DBMS will take lot of time to get the data for that purpose the RDBMS is Created

Relational Model

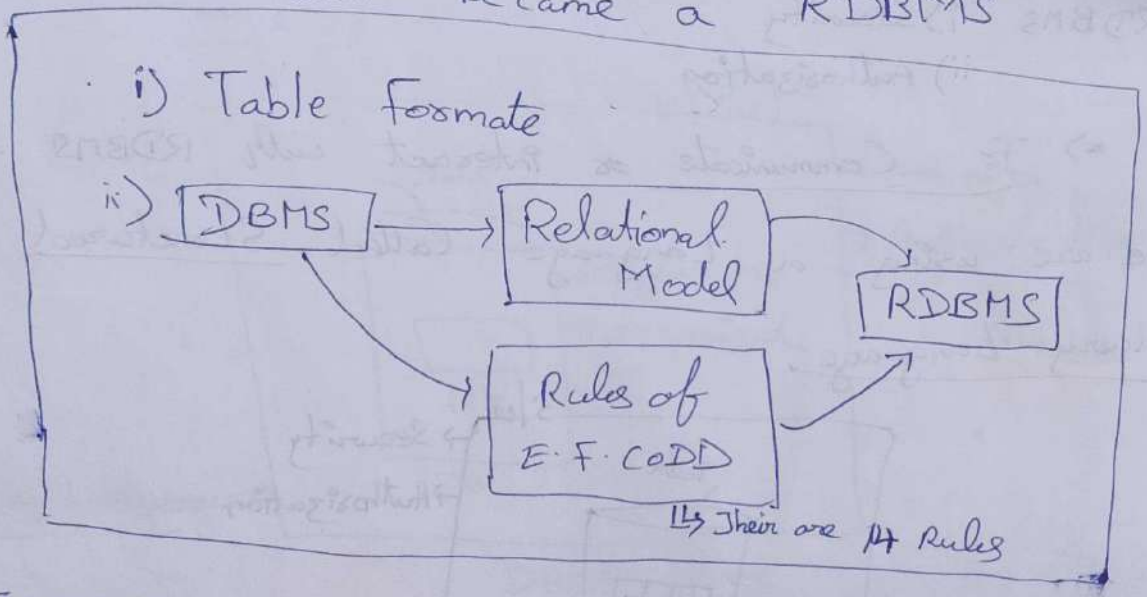
E.F. CODD

=> This model was Invented by a Data Scientis Called E.F. CODD (Edgar Frank CODD)

=> In Relational Model the data's must be Stored in Table Formate

=> Any DBMS which follows Relational Model will becames a RDBMS (or)

Any DBMS which follows Rules of E.F. CODD will became a RDBMS



TABLE

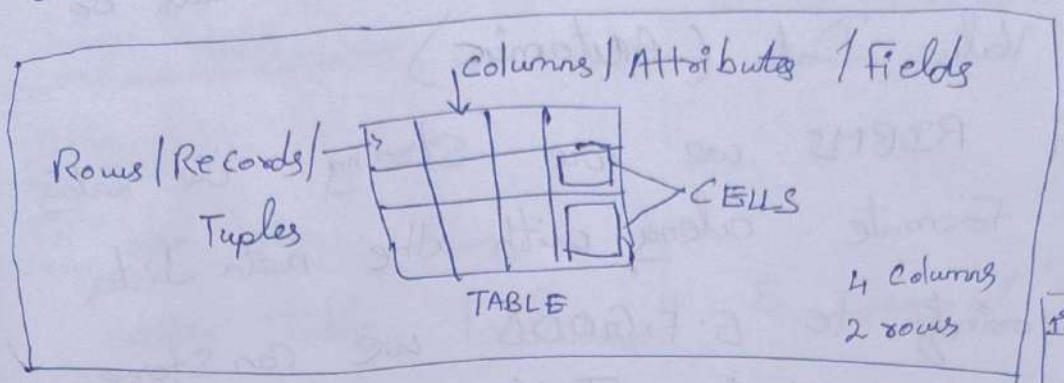
-> Logical Organization of Row and Columns is known as TABLE

-> COLUMN -> The vertical Sectors or Section in a table are Called Columns

=> Columns are also called as ATTRIBUTES / FIELDS

⇒ ROWS / RECORDS / TUPLES

⇒ The Horizontal Sectors or Section in a table are called Rows



1st row is for Column name arguments

CELL

⇒ The Smallest Unit in a Table where we will be storing the data's is Called Cell

⇒ By Intersecting Rows and Columns we Can Create Cells

SOFTWARE LINK (Desktop link Application)

bit.ly/roSoftWIN

YOUTUBE CHANNEL: ro helpmate

INSTA ID : RAJJJ3031

Rules of E.F. CODD

1.) The Data Entered in a Cell must be a Single Value Data (Atomic)

2.) In RDBMS we are storing the datas in Table formate along with the meta Data

3.) According to E.F.CODD we can store the Data's in multiple Tables with the help of key Attributes.

4.) The Data Entered in a Cell should be Validated by 2 steps

1.) by assigning Datatypes

2.) by assigning Constraints

NOTE :- Data Types are Mandatory but Constraints are optional

Meta Data => The Details About the data is known as Meta Data

Meta Table => The Table where we will be storing Meta Datas is called Meta Table.

* Meta Data and Meta Table are Automatically Generated

Datatype

→ It is used to Specify what kind of (or) what type of Data to be Stored in a memory allocation.

Types of Datatypes

→ In SQL We have 5 types of Datatypes.

1. CHAR
2. VARCHAR / VARCHAR2
3. DATE
4. NUMBER
5. LARGE OBJECT
   i) CHARACTER LARGE OBJECT
   ii) BINARY LARGE OBJECT

1-> CHAR Datatype

* Char datatype can accepts Charactes like: [A-Z, a-z, 0-9, ALPHANUMERIC, SPECIAL CHARACTERS (@, #, $, !, ...)]
* Characters Must be Enclosed within [' '] Single Quotes
* When Ever we are using Char Datatype we have to assign Size for it.
* Char Datatypes follows FIXED LENGTH MEMORY ALLOCATION

=> SIZE

-> Size Indicates No of characters it Can Store.

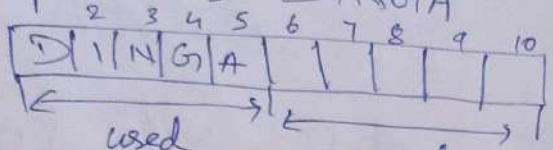
-> We Can Store upto maximum of 2000 Characters

SYNTAX

CHAR(SIZE)

eg:-

CHAR(10) 'DINGA'



Memory Wastage

2) VARCHAR

=> VARCHAR Datatype can accept characters like [A-Z, a-z, 0-9, ALPHANUMERIC, SPECIAL CHARACTERS (@, &, #, ! ....)]

=> Characters must be enclosed with in [' '] Single Quotes

=> When ever we are using VARCHAR Datatype we have to assign Size for it

-> VARCHAR Datatype follows

VARIABLE LENGTH MEMORY ALLOCATION

=> SIZE

→ Size indicates the no. of Charactors it can store

→ We can store upto 2000 charactors

SYNTAX

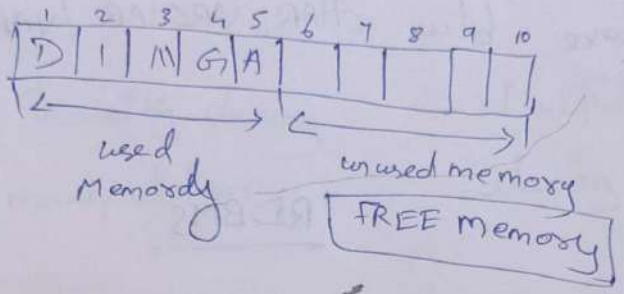
VARCHAR(SIZE)

↓

2000

Eg:-

VARCHAR (10) 'DINGA'



VARCHAR 2

=> VARCHAR 2 is the updated version of VARCHAR

=> We can store upto 4000 charactors

SYNTAX

VARCHAR2(SIZE)

↓

4000

3.) DATE DATATYPE

=> It is used to store Date Values

SYNTAX

DATE

-> To store the DATE value we need to follow the 2 formates of Data which was given by ORACLE

FORMATS

DD-MON-YY = '17-AUG-23'
DD-MON-YYYY = '17-AUG-2023'

Assignments

1.) Write the difference b/w DBMS and RDBMS
2.) List the Rules of E.F.CODD
3.) Write the difference b/w CHAR, VARCHAR / VARCHAR2

1.

	DBMS	RDBMS
Definition	Database Management System	Relational Database Management System
Data Storage	Data is Stored as file (File formate)	Data is Stored as Tables (Table Formate)
Data Access	Data Elements accessed Individually	Data Elements accessed at Same time
Relationship	There is no relationship b/w data in DBMS	Data is present in multiple tables which can be related eachother.
Normalization	Cannot be Achieved	Can be Achieved

Distributed DB	DBMS	RDBMS
	No Support	will Support
Data Quantity		
	Small Quantity of Data	Large Quantity of Data
User		
	Support Single user at a time	Support multiple user at a time
Security		
	Low Security during Data manipulation	Multilayer Security during Data Manipulation
Example		
	File Systems, XML, etc	Oracle, SQL SERVER

2.) Rules of E.F. CODD

1.) The Data Entered in a Cell must be a single Value Data

2.) In RDBMS we are storing the datas in Table formate along with the meta Data

3.) Accoading to E.F.CODD we can Store the Data's in multiple Tables with the help of key Attributes

4.) The Data Entered in a Cell Should be Validated by 2 Steps

1.) by assigning Datatypes

2.) by assigning Constraints

CHAR	VARCHAR	VARCHAR2
Can accept characters like		
A-Z, a-z, 0-9, Alphanumeric, Special characters	A-Z, a-z, 0-9 Alphanumeric, Special characters	Same as as char / Varchar
Enclosed under		
[' '] Single Quotes	[' '] Single Quotes	[' '] Single Quotes
It follows		
FIXED LENGTH MEMORY ALLOCATION	VARIABLE LENGTH MEMORY ALLOCATION	VARIABLE LENTH MEMORY ALLOCATION
SIZE of charadors		
2000	2000	4000
SYNTAX		
CHARACTER(SIZE)	VARCHAR(SIZE)	VARCHAR2(SIZE)
Unused Memory is Called as		
Memory Wastage	Free Memory	FreeMemory

NUMBER DATATYPE

=> It is used to Store Numeric Values

SYNTAX

NUMBER (PRECISION , [SCALE])

optional

PRECISION

-> It is used to Store integer Values

-> Precision ranges from 1 digit to 36 digits

-> We don't have any default Value for Precision, Since it is a mandatory argument for Number Datatype

SCALE

=> It is used to Store the Decimal Values with in the Precision

=> We Can store upto 127 digits Decimal Value

=> The Default Value for Scale is Zero

Examples

NUMBER (4) ± 9 9 9 9
NUMBER(5,2) ± 9 9 9 . 9 9
NUMBER(7,4) ± 9 9 9 . 9 9 9 9
} => PR > Sc

NUMBER(3,3) ± 9 9 9
NUMBER(6,6) ± 9 9 9 9 9 9
} => PR = Sc

NUMBER (5,7) ± 0 0 9 9 9 9 9
NUMBER (6,9) ± 0 0 0 9 9 9 9 9 9
} => PR < Sc
Sc - PR

5) LARGE OBJECT

i) CHARACTER LARGE OBJECT

=> It is used to Store characters upto 4GB of Size

SYNTAX

CLOB

ii) Binary Large Object

-> It is used to Store Binary digits like images, MP4, MP3, documents etc upto 4GB of Size

SYNTAX

BLOB

Constraints

-> It is a Extra validation which is assign for a Column

Types of Constraints => We have 5 types

i) Unique
ii) Not Null
iii) Check
iv) Primary key
v) Foreign key

UNIQUE

=> It is a Constraint which is assigned for for Column.

=> It will not accept Repeated or Duplicated Values

NOTNULL

=> It is a Constraint which is assigned for a Column

=> It will not accept NULL Values

=> It Represents the Column must Contain a Value

CHECK

=> It is a Constraint which is assigned for a Column which a Condition, If the Condition is Satisfy it will Accept the Value, Else It will Reject the Value.

PRIMARY KEY

-> It is used to Identify a record uniquly from the table.

Characteristics of Primary key

1.) It will not accept Repeated or Duplicated Values

2.) It will not accept NULL Values

iii) It is always Combination of unique and Not Null

iv) We should have only one Primary key in a table

v) It is not mandatory

FOREIGN KEY

-> It is used to Establish a Connection blw the tables

Characteristics of Foreign key

i) It can accept Repeated or duplicated Values

ii) It can accept Null Values

iii) It is not the Combination of unique and NotNull

iv) We can have many Number of Foreign keys in a table

v) It is not mandatory.

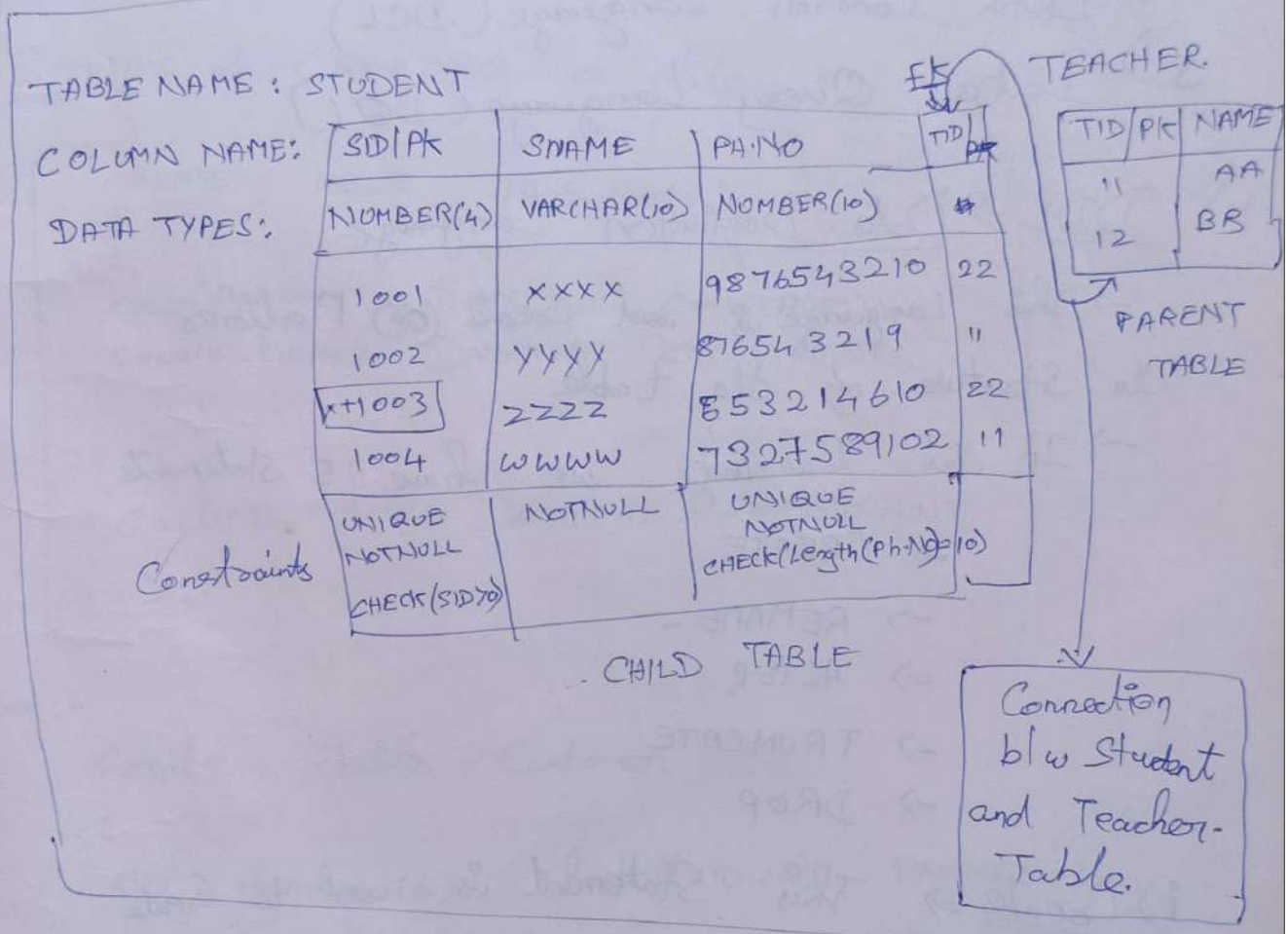
vi) For An Attribute wants to became a Foreign key it must be Primary key in its Own Table.

vii) Foreign key Present in child Table but actual belong to Parent Table.

viii) foreign key is also called refferential Integrity Constraint

Refferential Integrity Constraint

=> The Parent Table Can't be Deleted until unless we are have Connection with the child table



Overview of SQL Statements

→ We have 5 types of languages in SQL

1.) DDL
Data Definition Language

2.) DML
Data Manipulation Language

3.) Transaction Control Language (TCL)

4.) Data Control Language (DCL)

5.) Data Query Language (DQL)

1.) **DDL → Data Definition Language**

→ This Language is used Create (or) Modifies the Structure of the table

→ In this Language we have 5 statements

- => CREATE
- => RENAME -
- => ALTER
- => TRUNCATE
- => DROP

i) **Create** => This Statement is used to Create the table in database.

Flow
- => Name of the Table
- => Number of Columns
- => Name of the Columns
- => Assign Datatypes
- => Assign Constraints

TABLE-NAME - CUSTOMER.

No.OF. COLUMN - 04

DATATYPE

COLUMN_NAME	CID	CNAME	PH-No	ADDRESS
DATATYPE	NUMBER(5)	VARCHAR(20)	Number(10)	VARCHAR(30)
UNIQUE	UNIQUE		UNIQUE	
NOTNULL	NOTNULL	NOTNULL	NOTNULL	NOTNULL
CHECK	Check(CID>0)		check(Length(Ph-No)=10)	
PRIMARY KEY	PRIMARY KEY			
FOREIGN KEY				

SYNTAX of CREATING a Table

```
CREATE TABLE TABLE_NAME
(
COLUMN_NAME1 DATATYPE CONSTRAINTS,
COLUMN_NAME2 DATATYPE CONSTRAINTS,
    :
    :
COLUMN_NAMEn DATATYPE CONSTRAINT
);
```

eg:-

```
Create Table Customer
(
CID Number(5) check(CID>0) PRIMARY key,
CNAME VARCHAR(10) NOTNULL,
PH_No Number(10) UNIQUE, NOTNULL check(Length(Ph_No)=10),
Address VARCHAR(20) NOTNULL
);
```

Product

Create Table Product (

P_ID Number(4) check(P_ID>0) Primary key,
P_Name Varchar(10) Not Null,
P_Details Varchar(30) Not Null,
P_Price Number(5) check(P_Price>0)
Not Null

);

RENAME

=> This statement is used to change the existing table name to new tablename

SYNTAX

Rename Existin_Table_Name to New_Table_Name;

DESC => DESCIRPTION => Command to view the table

Ex:- 1) Rename Customer to Cust;
2) Rename Product to Prod;

ALTER

=> This Statement is used to modify the Structure of the Table

--> To Add a Column

SYNTAX

```
Alter Table Table_name
Add Column_Name Datatype Constraints;
```

eg:- Alter Table Cust
Add Blood_group Varchar(3) Not Null;

--> To Drop a Column

SYNTAX

```
Alter Table Table_name
Drop Column Column_Name;
```

eg:- Alter Table Cust
Drop Column Blood_group;

--> To Rename the Column

SYNTAX

```
Alter Table Table_name
Rename Column existing column to New_Column;
```

eg:- Alter Table Cust
Rename Column Address to Location;

--> To change the Datatype

```
Alter Table Tablename.
Modify Column_Name New_Datatype;
```

eg:- Alter Table Cust
Modify C_Name char(15);

Not Null to Null => Possible to Alter

Null to Not Null => Possible to

--> To change NULL/NOTNULL

SYNTAX

```
ALTER TABLE TABLE_NAME
MODIFY COLUMN_NAME DATATYPE NULL/NOT NULL;
```

eg:-

```
Alter Table Cust
Modify Location Varchar(10) NULL;
```

Table name (Rename)

```
Rename Product to prod;
```

To add new column to the table

```
Alter Table Prod
Add Review varchar(40) Not Null;
```

To change the datatype

```
Alter Table Prod
Modify Column Review to char(40);
```

To Rename the column name

```
Alter Table Prod
Rename column Review to Feedback;
```

To change the Not Null Constraint to Null

```
Alter Table Prod
Modify Feedback char(40) Null;
```

To drop the Table

```
Alter Table Prod
drop column Feedback;
```

TRUNCATE

→ This Statement is used to remove all the records from the table

`TRUNCATE TABLE TABLE_NAME` Syntax

eg:

Truncate table t1;

- Table Truncated.

Selet * from T1;

- no rows Selected

Desc T1;

[Column will Display]

DROP

→ This Statement is used to Remove the table from the Database

`DROP TABLE Tablename;` Syntax.

To Recover the Table from Recycle BIN:-

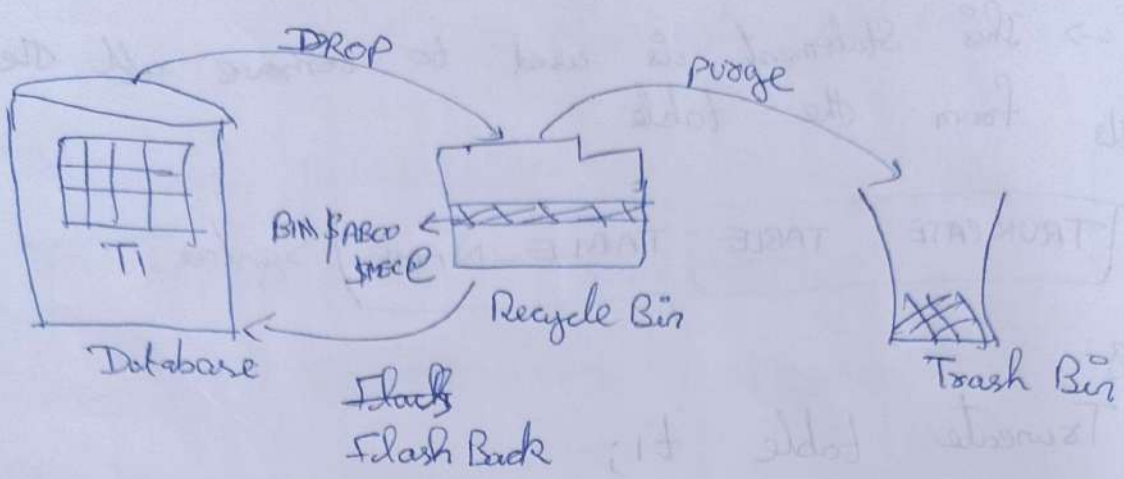
```
Flashback Table Tablename.
To before Drop;
```
Syntax

eg: Flashback Table T1
To before Drop;

To Remove table from the Recycle BIN:

`PURGE TABLE TABLENAME` Syntax

eg:- Purge Table T1;



Data Control Language (DCL)
(Grant, Revoke)

This Language is used to control the flow of Data b/w the users

In this Language we have two statements

* Grant
* Revoke

This statement is used to give permision to the user

Syntax

```
Grant SQL-Statement
On Tablename
To UserName;
```

This Statement is used to Take back the Permission from the user

Syntax

```
Revoke SQL-Statements
On Tablename
From username;
```

Grant Statement

```
Show user;
- user is "scott"
Connect;
Enter user-name: HR
Enter password: ******
                TIGER
- Connected.
Show user;
- user is "HR".
```

→ To Change the database.

To denay the access to Previous DB ⇓

Connect;
Enter user-name: ~~HR~~ Scott
Enter Password: ******
TIGER
- Connected.
Show user;
- user is "Scott"

```
Grant Select
on Dept
To HR
- Grant Succeeded.
```

Connect;
Enter username: HR
Enter Password: ******
- Connected

```
SELect *
2 From Scott.Dept
```

↓

To access Scott DB in HR DB.

To Revoke Table

Show user;
- user is "HR"
Connect;
Enter user-name: SCOTT
Enter Password: ******
- Connected.

```
Revoke Select
on Dept
from HR
- Revoked Succeeded
```

Connect;
E- U N : HR
P W : ******
- Connected

```
Select *
From Scott.Dept
```

ERROR: From Scott.Dept
*
table or view does not exist

Data Manipulation Language

This Language is used to Manipulate the table by performing Operations like
i) insert, ii) update, iii) Delete

In this Language we have 3 Statements
> insert
> update
> Delete

Insert

This Statement is used to enter the Values into the table

Syntax => Insert into Tablename Values (V1, ..., Vn);

↓

Values can be entering based on the number of Columns we Created

eg:) Insert into Tablename Customer (101, 'RAVI', 8754539214, 'HYD');

→ 1 row Created

Insert int Tablename Customer(102, 'RAVI2', 8753554924, NULL 'HYD');

→ 1 row Created

"If you mentioned Null Constraint we have to Pass Null Value to to other wise it shows no enough Values"

Update

It is used to Modify the Existing Value to new Value.

Syntax

```
Update Tablename
  Set Columnnam = Value
  [where <filter condition>];
```

eg.) Update Customer.
Set Location = Value
where CID = 1003;

→ 1 row updated

DELETE

It is used to Remove all the Record or Some Set of Records from the table.

Syntax

```
Delete From Tablename
[where <filterCondition>];
```

eg :- Delete From Customer
where CID = 102;

→ 1 row deleted.

NOTE :- Data Definition Language is a Auto Commit Statement but DML Data Manipulation Language is not a Auto Commit Statement.

TRANSACTION Control Language (TCL)

→ This Statement is used to Control the Transaction ~~Statements~~ done on the Database

→ The DML operations like insert, update, Delete are called as Transaction.

→ In this language we have 3 Statements

↔ Commit
↔ Roll back.
↔ Save Point

Data Query Language

=> This Language is used to Retrieve or Fetch the data from the Database

-> In this language we have 4 Statements

i) Select
ii) Projection
iii) Selection
iv) Joins

i) Select → It is used to retrieve the data from the table

ii) Projection → It is used to retrieve the data by Selecting only Columns

iii) Selection -> It is used to Retrieve the Data by Selecting Rows and Columns

iv) Joins => It is used to Retrieve the Data from multiple Tables ~~Table~~ Simultaneously

PROJECTION

-> It is used to Retrieve the data by Selecting only Columns.

-> In Projection by default all the records which are present in a Column will be Selected.

Syntax

```
SELECT */columnName [Distinct/Expression [ALIAS]
From Tablename;
```

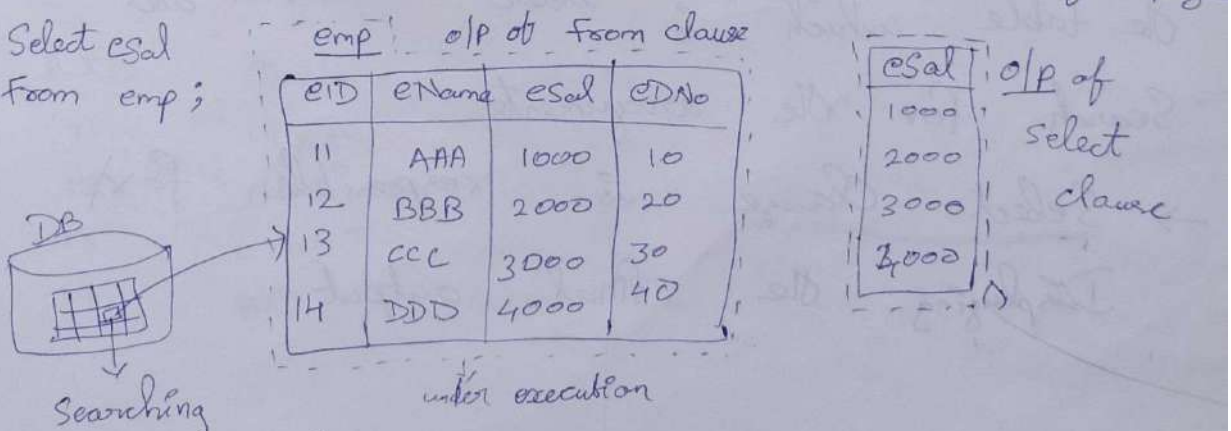
Order of Execution

Start from => From Clauses / Keywords / Statements

End with => Select Clauses / Keywords / Statements

=> Write a Query to Display Salaries of Employees

```
Select eSal
From emp;
```



Working Procedure of Projection

→ From Clause will start the Execution.
→ From Clause we can Pass Tablename as an argument
→ The Job of from clause is go to the Database and Search for the table.
→ If the table is not present From clause will Stop the execution and throws an Error message.
→ If the table is Present it will Put it under execution
→ After Completion of ~~Exec~~ from clause Execution. Select Clause will Start the execution.
→ For Select Clause we can Pass 3 arguments
  => * (Asterisk)
  => ColumnName
  => Expression
→ The Job of Select Class is go to the table which is under execution and Search for the arguments.
→ Select Clause is responsible for Displaying the final output

-> * (Astrisk)

(1) It is used to display all the detail from the table

-> Semi Colon(;)

(1) It represents End of the Query

Write a Query to display the tables which are present in Database?

```
Select *
From Tab;
```

O/P

TNAME	TABTYPE	CLUSTERID
DEPT	TABLE	
EMP	TABLE	
BONUS	TABLE	
STD	TABLE	
CUST	TABLE	

W.A.Q to Display Details of Department Table

```
Select *
From Dept;
```

D_ID	D_Name	Location
11	XYZ	NewYork
12	ABC	DALLAS
13	LMN	CHICAGO
14	QRS	BOSTON

W.A.Q to display ~~the~~ details of the Employees

Table will display with out Arrangement

```
Select *
From emp;
```

⟹

```
Set lines 100 Pages 100;
Select *
From emp;
```

} <= for Table Arrangement Number is our wish

o/p

EmpNo	EmpName	Job	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30

W.A.Q.T.D Names of the Employees

```
Select EmpName
From emp;
```

→ if any error in 1st line it shows Invalid Identifier

→ if any error in 2nd line it shows table or view does not exist.

WAQTD HIREDATE of the Employees

```
Select hiredate
From emp;
```

WAQTD NAMES, Salary, Hiredate of the Employees

```
Select EmpName, Sal, Hiredate
from emp;
```

WAQTD All the Details from the Employee Table

Ans⇒ Select *
Set lines 200 Pages 200;
Select * From emp;

WAQTD Names of All the Employees.

Select Ename
from emp;

WAQTD Names and Salary Given to all the Employees

Select Ename, Sal
From emp;

WAQTD Names and Commission Given to all Employees.

Select Ename, Comm
From Emp;

WAQTD Employee ID and Department Number of all Employees in EMP table.

Select EMPNO, DEPTNO
From Emp;

WAQTD ENAME and HIREDATE of All the employees

Select Ename, Hiredate
From emp;

WAQTD NAME and Designation of all the Employees.

```
Select Ename, Job
From EMP;
```

WAQTD NAME, JOB, & Salary of all the Employees

```
Select Ename, Job, Sal
From Emp;
```

WAQTD DNAMES Present IN Department Table.

```
Select Dnames
From Dept;
```

WAQTD DNAME and Location Present in Dept Table.

```
Select Dnames, Loc
From Dept;
```

Expression -> Any Statement which Gives Result is known as Expression

⇒ Expression is the Combination of operators and operands

Operators (+, -, *, %) : These are the Symbols which we are using to perform Some operation on the operands

Operands => It Can be Classified into 2 types

- Columnname
- Literals

Literals (Values) => These are the values which are using

=> Literals Can be Classified into 3 types

- Character Literal
- Date Literal
- Number Literal.

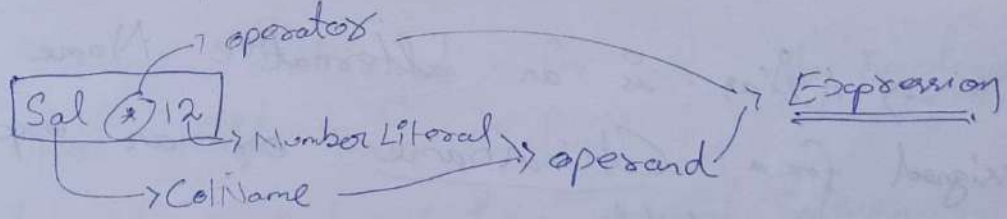
Note :- SQL is not a Case Sensitive Language but Literal are Case Sensitive.

WAQTD Details of the Employees.

```
Select * From Emp;
```

WAQTD Annual Salary of the Employees.

```
Select Sal*12
From Emp
```



HIKE IN MONTHLY SALARY

$$Sal + Sal * \frac{a}{100}$$

DEDUCTION / REDUCTION IN MONTHLY SALARY

$$Sal - Sal * \frac{a}{10}$$

HIKE IN ANUAL SALARY

$$Sal * 12 + Sal * 12 * \frac{a}{100}$$

DEDUCTION / REDUCTION IN ANUAL SALARY

$$Sal * 12 - Sal * 12 * \frac{a}{100}$$

=> Here (a) is the Percentage what the user requested

WAQTD Annual Salary with the Hiptke of 40 %.

```
Selert Sal *12 + Sal *12 * 40/100
Foom emp;
```

ALIAS

=> Alias is an alternative Name which is assigned for a Columnname or an espression in the Result table.

=> We Can assign Alias name with or without using "as" keywood.

-> Alias name must be Single String which is enclosed in [" "] or Separated by an [-]

Possibilities to have Alias in Query

eg=> Sal * 12 AS "Annual_Salary"
" AS Annual_Salary
" AS "Annual Salary"

Sal * 12 "Anual_Salary"
"
" Annual_Salary
" "Anual Salary"

Select Sal *12
from emp; o/p => Sal*12

Select Sal * 12 "Annual_Salary"
from emp; o/p => Annual_Salary

Assignment - a

1.) WAQTD NAME of the employee along with their annual Salary

```
Select ename, sal *12 as Annual_Salary
from emp;
```

2.) WAQTD Ename and job for all the Employee. with there Half Term Salary

```
select ename, job, sal * 6 "Annal Salary
from emp;
```

3.) WAQTD all the Details of the employees Along with an Annual Bonus of 2000.

Select empno, ename, job, mgr, hiredate, sal, Sal*12 +2000 Annual_Bonus, comm, deptno
from emp;

4.) WAQTD all the Details of the employees Along with an Annual bonus of 2000

4) WAQTD Name, Salary and Salary with a hike of 10%
select ename, Sal, Sal +Sal *10/100 Monthly_hike
from emp;

5.) WAQTD Name and Salary with Deduction of 25%.;
Select ename, Sal - Sal * 25/100 Monthly_Deduction
from emp;

6.) WAQTD Name and Salary with Monthly Hike of 50.
Select ename, Sal, Sal+Sal *50/100 Monthly_hike
from emp;

7.) WAQTD Name and Annual Salary with Deduction of 10%.
Select ename, sal * 12 Annual Salary, sal *12 - Sal*12 * 10/100
from emp;

8.) WAQTD Details of all the employees Along with the Annual Salary
Selet empno, ename, job mgr, hiredate, Sal, Sal*12 annual_Salary, comm, deptno
from emp;

8.) WAQTD Total Salary Given to Each Employee (Sal + comm)

Select ename, Sal, Comm, Sal+comm "Total Salary"
from emp;

10.) WAQTD Name and Designation along with 100 penality in Salary

Select ename, job, Sal, Sal - 100 "penalty"
from emp;

8.) Select ename, sal, (sal + NVL(comm, 0))
↳ function to Certain NULL
from emp

NULL VALUE ⇒ NVL

Along with the astrisk, we are not Suppose to pass any other argument in Select clause.
→ If we want to pass, we have to follow the Syntax `Tablename.*`

(3) Select emp.*, Sal*12 + 2000 "Annual bonus"
from emp;

(9) Select emp.*, Sal*12 "Annual-Salary"
from emp;

Distinct Clause

It is used to Remove the Duplicated or Repeated Values from the Result D. Table.

→ Distinct clause should be Pass as an first argument for the Select Clause

⇒ We Can Pass Multiple Columns as an argument for the distinct Class it will Remove the Columns in which the Combination of records are duplicated

STUDENT

SID	SNAME	BRANCH	PER.
1	A	ME	65
2	B	CIV	70
3	C	CSE	75
4	B	ECE	77
5	A	CIV	70

WAQTD Names of the Students

```
select sname
from student;
```

O/P

SNAME
A
B
C
B
A

WAQTD Different Names of the Students

```
Select Distinct Sname
from Student;
```

o/p

Sname
A
B
C

WAQTD Unique Branches of the Students

```
Select Distinct branch
from Student;
```

o/p

Branch
ME
CIV
CSE
ECE

WAQTD Different names and branches of the students

```
Select Distinct Sname, branch
from Student;
```

o/p

SName	Branch
A	ME
B	CIV
C	CSE
B	ECE

⇒ Selection

* It is used to Retrieve the Data by Selecting rows and Columns

Syntax

```
Select */Column-Name [Distinct]/Expression [Alias]
From Tablename
where < Filter_Condition>;
```

Order of Execution

From
where - row by row
Select - row by row

WHERE CLAUSE

* It is used to filter the Records
* For where clause we can Pass filter Condition as an argument
* It executes after the execution of from clause
* It executes Row by Row
* The filter Condition what we are Passing for the where clause it must ~~written~~ return boolean value " True or False".
* We can Pass multiple Conditions as an arguments for the where clause by using logical operators

Syntax of filter Condition.

Columnname / Expression Operator Value

WAQTD names of the Employees if they are working in Dept. No 20

```
Select Ename
from Emp
where DNo = 20
```

DNo = 20 → filter Condition

Emp ← under execution

EID	Ename	Sal	DNo
1	XYZ	1000	20
2	ABC	5000	30
3	UVW	3000	40
4	PQR	4000	10
5	EFG	2000	20

o/p of from clause

1 XYZ 1000 20 (T)
2 ABC 5000 30 X (F)
3 UVW 3000 40 X (F)
4 PQR 4000 10 X (F)
5 EFG 2000 20 ✓ (T)

1 XYZ 1000 20
5 EFG 2000 20

o/p of where clause

Ename
XYZ
EFG

o/p of Select clause

WAQTD Details of the Employee if they hired after 1981

```
Select emp.*
from emp
where hiredate > 1981;
```

inconsistent datatype

error : expected Date got Number

```
Select *
from emp
where hiredate > '31-DEC-1981';
```

WAQTD Details of the Employees if they are working as Salesman

```
Selat *
from emp
where job = 'SALESMAN';
```

```
Select *
from emp
where job = 'Salesman';
```

X no row selected

Assignment-6

1.) WAQTD the annual salary of the Employee whos name is Smith

```
Selat Sal*12 as Annual_Salary
from emp
where ename = "SMITH";
```

2.) WAQTD Name of the Employees working as Clerk

```
Selat ename
from emp
where job = 'CLERK';
```

3.) WAQTD Salary of the Employees who are working as salesman

```
Select Sal
from emp
where job = 'CLERK' 'SALESMAN';
```

4) WAQTD Details of the Emp who Earns more than 2000

```
Select *
from EMP
where Sal > 2000;
```

5.) WAQTD Details of the emp whos name is jones

```
Select *
from emp
where ename = 'Jones';
```

6.) WAQTD Details of the Emp who has Hiredate After 01-jan-81

```
Select *
from emp
where hiredate > '01-jan-81';
```

7) WAQTD name and Sal Along with his Annual Salary if the annual Salary is more that 12000

```
Select Ename, Sal, Sal*12 as Annual Salary
from emp
where sal*12 > 12000;
```

8.) WAQTD EMPNo of the Employees who are working in Dept 30.

```
Selet empno
From Emp
where deptno = 30;
```

9.) WAQTD Ename and Hiredate if they are hired before 1981

```
Select ename, hiredate
from emp
where hiredate < '01-jan-1981';
```

10) WAQTD Details of the employees working as Manager

```
Select *
from emp
where job = 'Manager';
```

11.) WAQTD name and Salary given to an Employe if employee earns a Commision of Rupees 1400

```
Select Ename, Sal
From EMP
where COMM = 1400;
```

12.) WAQTD Details of Employees Having Commission more than Salary?

```
Select *
from emp
where comm > Sal;
```

13.) WAQTD Details of Employees Hired before the year 87

```
Select *
from emp
where hiredate < '01-jan-87';
```

14) WAQTD Details of Employees working as an Analyst?

```
Select *
from emp
where job = 'Analyst';
```

15.) WAQTD Details of Employees earning more than 2000 rupees per month.

```
Select *
from emp
where Sal > 2000;
```

NULL ⇒ Is a keyword which represents empty Space

Rules of NULL =>

1.) In SQL "0" and " " will not be Considered as Null Values

2.) Any Arithmatic Operation Performed on the Null Values will Produce Null itself.

3.) We Can't Equate Null Values

TYPES OF OPERATORS (8 types)

1.) ARITHMATIC operators (+, -, *, /, %)

2.) COMPARISON operators (=, != or <>)

Angular Brackets

3.) RELATIONAL operators (>, <, >=, <=)

4.) CONCATENATION operators (||)

5.) SET operators
- * UNION
- * UNION ALL
- * INTERSET
- * MINUS

6.) LOGICAL operators
- * AND
- * OR
- * NOT

7.) SPECIAL operators
- * IN
- * Not IN
- * BETWEEN
- * NOTBETWEEN
- * IS
- * IS NOT
- * LIKE
- * NOT LIKE

8.) SUB QUERY Operators
- * ALL
- * ANY
- * EXISTS
- * NOT EXISTS

SET OPERATIONS | SET OPERATORS

=> The SQL SET operation is used to Combine the two or more SQL SELECT Statements

>> Types of Set Operators

1.) Union
2.) Union all
3.) Intersect
4.) Minus

1.) UNION

> The SQL Union operation is used to Combine the result of two or more SQL Select Queries

> In the union operation, all the number of Datatype and Columns must be Same in both the tables on which UNION operation is being applied.

> The union operation eliminate the Duplicate rows from its resultset.

Syntax

```
Select Columnname from table1
Union
Select Columnname from table2;
```

Example :- T_1

ID	N
1	A
2	B
3	C

T_2

ID	N
3	C
4	D
5	E

Union SQL Query will be;

```
Select * from T1.
Union
Select * from T2;
```

The resultset table will look like;

ID	N
1	A
2	B
3	C
4	D
5	E

2.) Union All:

> Union all operation is equal to the union operation.

> It returns the set without removing duplication and Sorting the data.

Syntax

```
Select Columnname from t1
union all
select Columnname from t2;
```

eg:- Using the above T1 and T2 table. union all Query will be like

```
Select * from T1
union all
Select * from T2;
```

The resultSet table will look like

ID	N
1	A
2	B
3	C
3	C
4	D
5	E

3.) INTERSECT:

- It is used to combine two Select statements
- The Intersect operation returns the common rows from both the select statements
- In the Intersect Operation, the number of datatype and column must be the same.
- It has no duplicates and it arranges the data in ascending order by default.

Syntax

```
Select Columnname from Table1
intersect
Select Columnnam from Table2;
```

Example.

```
Select * from T1
intersect
Select * from T2;
```

O/P

ID	N
3	C

4) MINUS

> It Combines the result of two Select Statements
> Minus Operator is used to display the rows which are present in the first Query but absent in the Second Query
> It has no duplicate and data arranged in ascending order by default.

Syntax

```
Select Columnname from Table1
MINUS
Select Columnname from Table2
```

eg:

```
Select * from T1
MINUS
Select * from T2;
```

o/p

ID	N
1	A
2	B

Logical Operators

→ The operators are used to Pass multiple Conditions for where clause

→ we have 3 types

- → AND
- → OR
- → NOT

AND operator

→ It is used to Pass Multiple Conditions for the where clause

→ It must be define in b/w the Condition

→ AND operators return True value if all the Conditions are Satisfied

→ It Performs Binary Multiplication

Syntax

Condition1 and Condition2 and Condition3

C_1	C_2	R
1	0	0
0	1	0
0	0	0
1	1	1

0 → false
1 → True

1) WAQTD Details of the Salesman

```
Select * from emp
where job = 'Salesman';
```

2.) WAQTD Details of the Employees if they are working in Dept.no 20.

```
Select * from emp
where Deptno = 20;
```

3.) WAQTD Details of the Employees if they are working as Salesman in Deptno 20

```
Select * from emp
where job = 'Salesman' and Deptno = 20;
```

O/P -> no rows selected.

4.) WAQTD Details of the Employees if they Hired after 83 and earning Salary more than 2000

```
Select *
from emp
where Hiredate > '31-12-83' and Sal > 2000;
```

OR OPERATOR

-> OR operator is used to Pass multiple Conditions for the where Clause

-> It Returns True Value if any one Condition is Satisfied

-> OR operator must be defined in b/w the Condition

-> It Performs Binary Addition

Syntax

Condition1 or Condition2 or Condition3

C_1	C_2	R
1	0	1
0	1	1
0	0	0
1	1	1

1.) WAQTD Details of the Employees if they are working as Salesmane or working in Deptno 20

```
Select *
from emp
where $job = 'Salesman' or Deptno = 20;
```

2.) WAQTD Details of the Employees if they are working in deptno 10, 20, 30.

```
Select *
from emp
where Deptno = 10 or Deptno = 20 or
                    Deptno = 30;
```

```
Doptno = 10, 20, 30;
```

O/P SQL Command not properly ended.

```
Deptno = 10 or 20 or 30
```

Assignment - 2

1.) WAQTD Details of the Employees working as clerk and Earning Salary less than 1500.

```
Select * from emp
where job = 'CLERK' and Sal<1500;
```

2.) WAQTD Name and Hiredate of the Employees working as manager in Dept 30.

```
Select ~~* from~~ ename, hiredate
from emp
where job = 'MANAGER' and deptno = 30;
```

3.) WAQTD Details of the Emp along with annual Salary if they are working in dept 30 as Salesman and their Annual Salary has to be Greater than 14000.

```
Select emp.*, Sal *12 Annual_Salary
from emp
where job = 'SALESMAN' and deptno=30 and Sal*12>14
```

4.) WAQTD all the Details of the Emp working in Dept 30 or as Analyst.

```
Select *
from emp
where job = 'ANALYST' or deptno = 30;
```

5.) WAQTD Names of the Employees who's Salary is less than 1100 and their Designation is clerk

```
Select ename
from emp
where sal<1100 and job = 'CLERK';
```

6.) WAQTD Name and SAL, Annual Sal and Deptno if Deptno is 20 earning more than 11000 and Annual Salary Exceeds 12000.

```
Select ename, Sal, Sal * 12 Annual_Salary, deptno
from emp
where Sal * 12 > 12000 and deptno = 20 and Sal > 11000;
```

7.) WAQTD EMPNO and Names of the employees working as manager in Dept 20.

```
Select empno, ename
from emp
where job='MANAGER' and deptno = 20;
```

8.) WA QTD Details of Employees working in Dept 20 or Dept 30

```
Select *
From emp
where deptno = 20 or Deptno = 30;
```

9.) WAQTD Details of Employees working as analyst in Dept 10.

```
Select *
from emp
where job = 'ANALYST' and Deptno = 10;
```

no rows selected

10.) WAQTD Details of Employees working as President with Salary of Rupees 4000.

```
Select * from emp
where job = 'PRESIDENT' and Sal = 4000;
```

no rows selected

11.) WAQTD Names and Deptno, Job of Emps working as clerk in Dept 10 or 20.

```
Select ename, deptno, job
from emp
where job='clerk' and (deptno=10 or deptno=20);
```

12.) WAQTD Details of Employees working as clerk or manager in Dept 10.

```
Select * from emp
where (job='CLERK' or job='MANAGER') and
Deptno=10;
```

13.) WAQTD Names of Employees working in Dept 10, 20, 30, 40.

no rows selected

```
Select ename
from emp
where deptno=10 ~~and~~ or deptno=20 ~~and~~ or deptno=30 ~~and~~ or
where deptno
deptno=40;
```

14.) WAQTD Details of Employees with Empno. 7902, 7839

```
Select * from emp
where empno=7902 or Empno=7839;
```

15.) WAQTD Details of Employees working as manager or Salesman or clerk.

```
Select * from emp
where job='MANAGER' or job='SALESMAN' or job='CLER
```

16.) WAQTD Names of Employees hired after 81 and before 87.

```
Select Ename
from emp
where hiredate > '31-DEC-81' and hiredate < '01-jan-87';
where hiredate between '01-JAN-81' and '31-DEC-86';
```

17.) WAQTD Details of Employees Earning more than 1250 But Less than 3000.

```
Select * from emp
where Sal > 1250 and Sal < 3000;
where Sal between 1251 and 2999;
```

18.) WAQTD Names of Employees hired After 81 into Dept 10 or 30.

```
Select ename
from emp
where hiredate > '31-DEC-81' and (deptno = 10 or
                                  deptno = 30);
```

19.) WAQTD Names of Employees along with Annual Salary for the Employees working as manager or clerk into dept 10 or 30.

```
Select ename, sal*12 Annual_Salary
from emp
where (job = 'MANAGER' OR job = 'CLERK') and
      (deptno = 10 or deptno = 30;)
```

20.) WAQTD all the Details Along with annual Salary if Sal is blw 1000 and 4000 Annual Salary More than 15000.

```
Select emp.*, Sal*12 Annual_Salary
from emp
where Sal>1000 and Sal<4000 and Sal*12>15000;
```

WAQTD Details of the Employees if they are working in Dept no 20 as a manager of Salesman.

✓
```
Select *
from emp
where deptno=20 AND (job='MANAGER' OR
                     Job='SALESMAN');
```

✗
```
Select *
from emp
where deptno=20 AND job='MANAGER' OR
                    job='SALESMAN';
```

* AND and OR

Condition1 and (Condition2 or Condition3);

Condition executions flow

first AND operator
Second OR Operator

WAQTD Details of the Employees if they are working as manager and working in Deptno 10,20,30.

```
Select * from emp
where job='MANAGER' AND (deptno=10 or
                         deptno=20 or deptno=30);
```

NOT OPERATOR → opposite Result and

WAQTD Details of the Employees if they are not working as Salesman

```
Select *
from emp
where job<>'SALESMAN';
```

```
Select * from emp
where job!='SALESMAN';
```

```
Select * from emp
where NOT job='SALESMAN';
```

NOT
i) <>
ii) !=
iii) NOT
↓
Before Condition.

[If we mentioning NOT operator to the Condition its opposite's Result to that Condition]

1.) WAQTD Deptno of Employees.

```
Select deptno from emp;
```

ORDER BY

=> It is used to arranges the Records or Result table either Ascending or Descending order

=> For Order By Clause we can pass Column name or Expression as an argument

=> Order by clause is the Only Clause which is ~~Return~~ written at the End

=> Order by clause is the Only clause which will executes at the End.

=> Order by clause executes row by row

=> By Default Order by clause arranges the Records in Ascending Order [ASC]

=> If we want to arrange the Records in Descending order we must use DESC keyword.

=> we can pass multiple arguements for the order by clause i) which will arranges the record based on the first argument

ii) will arranges the records based on the ACSII values

Low Value to High Value => ASC

High Value to Low Value => DESC

Syntax

```
SELECT COLUMNNAME/EXPRESSION
FROM TABLENAME
[WHERE <FILTER_CONDITION>]
ORDER BY COLUMNNAME/EXPRESSION [ASC]/DESC;
                                optional
```

Order of execution

```
FROM
WHERE [if used] → row-by-row
SELECT          → row-by-row
ORDER BY        → row-by row.
```

eg:- WAQTD Deptno in Ascending order.

```
select deptno
from emp
order by deptno; (or) order by deptno ASC;
```

WAQTD Deptno of Employees in Descending order.

```
select deptno
from emp
order by deptno DESC;
```

WAQTD Salaries of the Employees in ascending order.

```
select sal
from emp
order by sal;
```

WAQTD Deptno and Salaries of the employees in assending order.

```
select Deptno, Sal
from emp
Order by deptno, Sal;
```

→ will arrage results based on the first argument

WAQTD Deptno and Salaries of the employees in descending order.

```
select deptno, Sal
from emp
order by deptno, Sal DESC;
```

✗

```
select deptno, Sal
from emp
order by deptno DESC, Sal DESC;
```

✓

WAQTD Deptno and Salaries of the Employes arranged Deptno in Descending order. but Salaries in order

```
select deptno, Sal
from emp
order by deptno DESC, Sal;
```

WAQTD Deptno and Salaries of employees arrange Deptno in asending order and but Salaries in descendin order.

```
Select deptno, Sal from emp
order by deptno, Sal DESC;
```

WAQTD Names of the employees in ascendin order

```
Select ename from emp
order by ename;
```

will arrang records based on the ASCII Values, which as Least to high Values of ascending

Least to high =>

WAQTD Details of the employees details of the ~~or~~ in deptno 10, 20, 30

```
Select * from emp
where deptno = 10 or deptno = 20 or deptno = 30 or
                                    deptno = 40
```

Special Operator

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

IN (=) => It is a Special Operator for equals operator. by using in operator at a time we can Compare multiple values

Syntax => ColumnName / Expression IN ($V_1, V_2, V_3 \ldots V_n$)

=> Details of emp where deptno 10, 20, 30, 40

```
Select * from emp
where deptno IN(10,20,30,40);
```

=> WAQTD Details of employees if they are working as manager, Salesman, President

```
select * from emp
where job IN('MANAGER','SALESMAN',
'PRESIDENT');
```

NOT IN (!=)

It is similar to IN operator but instead of selecting the value it will reject the value

Syntax

Columnname/Expression Not in ($V_1, V_2, V_3 \ldots V_n$);

⇒ WAQTD Details of Employees if they are Not working as manager, Salesman, President

```
Selet * from emp
where job NOT IN ('MANAGER', 'SALESMAN',
                  'PRESIDENT');
```

Between (>= , =<)

It is used. whenever we are having range of values

Syntax

ColumnName / Expression BETWEEN LOWER RANGE AND HIGHER RANGE

NOTE/→ Between operator will includes the ranges & we are not suppose to interchange the ranges

WAQTD DOE if Salary ranges from 1000 to 5000

```
Select * from emp
where Sal BETWEEN 1000 AND 5000;
```

WAQTD NAME and Hiredate of Employes if they hireded after 1982 but before 1987

```
Select ename, hiredate from emp
where hiredate Between '01-JAN-83' AND
                       '31-DEC-86';
```

```
Select ename, hiredate from emp
where hiredate >'31-DEC-82' AND hiredate >'01-JAN-83'
```

```
Select ename, hiredate from emp
where hiredate >='01-JAN-83' AND hiredate <='31-DEC-86'
```

NOT BETWEEN

It is Similar to between operator but instead of Selecting the Value it will reject the Value

Syntax:

```
ColumnName / Expression NOT BETWEEN LOWER RANGE
                        AND HIGHER RANGE;
```

WAQTD DOF if their Salary Not Ranges from 1000 to 5000;

```
Select * from emp
where Sal Not Between 1000 to 5000;
```

WAQTD DOF if their earning Salary in b/w 1000 and 5000

```
Select * from emp.
where Sal between 1000 and 5000;
```

```
Select * from emp
where Sal between 1001 and 4999;
```

IS

=> It is used to Compare only with NULL Values

Syntax	Columnname	Expression	IS NULL;

WAQTD DOF if they are not earning Commision.

```
Select * from emp
where Comm = NULL; X no row selected.
Where Comm is NULL;
```

WAQTD DOE if they are not reporting to anyone

```
Select * from emp
where MGR IS NULL;
```

IS NOT

It is Similar to IS operator but Instead of Selecting the value it will reject the value.

Syntax=

```
Columnname/Expression IS NOT NULL;
```

WAQTD DOE if they are earning Comm

```
Select *
from emp
where comm IS NOT NULL;
```

WAQTD Names of Employees of their name Starting with Charactor 'A'

```
Select ename from emp
where ename Like 'A%';
```

LIKE

It is used to retriev the pattern matching

Syntax

Columnname / Expression LIKE 'PATTERN-TO-MATCH';

⇒ To retrieve Pattern matching we have to use two special Symbols

i) (%) Percentage Percentile

ii) (_) UnderscORE

Percentile => (Dnt known the Length)

It can accept any number of Characters any number of types and No characters also

Underscore => It can accept Only One Character for One Underscore. (knowning with the Length)

Percentile

'A%' '%A' '%A%' '%A%A%A' '%AA%'

1st char Last char Minimum one char minimum of 2 char Cons. char.

Underscore

'____' Specifying the digits / Characters

WANTED DOE if they are earning 4 digit Salary

```
Select * from emp
where Sal Like '____';
```

NOT LIKE

It is similar to like Operator but instead of selecting the Values it will reject the Value

Syntax

Columnname/Expression NOT LIKE 'PATTERN_TO_MATCH'

WAQTD Names of the employees if they does not Starts with the charactes 'A';

```
Select ename from emp
where ename NOT LIKE 'A%';
```

WAQTD Names and Salaries of employees if they are not earning 4 digit Salary

```
Select ename, sal from emp
where sal NOT LIKE '____';
```

Assignment - 1

1.) List all the Employees whose Commission is N

```
select * from emp
where comm is NULL;
```

2.) List all the Employees who don't have reporting manag

```
select * from emp
where MGR IS NULL;
```

3.) List all Salesman in dept 30

```
select job from emp
where dept IN 30, AND job='Salesman';
```

4.) List all the Salesmen in dept number 30 and having Salary Greater than 1500.

```
Select job from emp
where job='salesman' AND deptno = 30 AND SAL>
1500;
```

5.) List all the Employees whose Name Starts with 'S' or 'A'

```
Select * from emp
where ename Like 'S%' OR ename Like 'A%';
```

6.) List all the Employees except those who are working in dept 10,20.

```
Select * from emp
where deptno NOT IN(10,20);
```

7.) List the Employees whose Name does not start with 'S'.

```
Select * from emp
where ename NOT LIKE 'S%';
```

8.) List all the Employees who are having reporting managers in dept 10

```
Select * from emp
where MGR IS NOT NULL and deptno = 10;
```

9.) List all the employees who are having reporting manag whose commission is NULL and working as clerk.

```
Select * from emp
where comm is NULL and job = 'CLERK';
job IN ('CLERK');
```

10.) List all the employees who Don't have a reporting manager in deptno 10 or 30.

```
Select * from emp
where MGR is NOT NULL AND (deptno = 10 or
                           deptno = 30);
```

11.) List all the Salesman in dept 30 with Salary more than 2450.

```
Select job from emp
where job is 'SALESMAN' AND deptno = 30 AND
Sal > 2450;
```

12.) List all the Analyst in DeptNo 20 and having Salary Greater than 2500.

```
Select job from emp
where job is 'Analyst' and deptno = 20 and
                       Sal > 2500;
```

1600
1250
1250
1500
no rows selected.

13.) List all the employees whose name Starts with 'M' or 'J'

```
Select * from emp
where ename like 'M%'  (or)  ename like 'J%'
```

→ new rows selected

14. List all the employees with Annual Salary except those who are working in dept 30.

```
Select emp.*, sal*12 as annual_salary
from emp
where deptno NOT IN 30;
```

15.) List the employees whose name does not end with 'ES', 'R'

```
Select * from emp
where ename not like '%ES' and ename not like '%R';
```

16.) List all the employees who are having reporting manager in dept 10 along with 10% hike in Salary

```
Select emp.*, sal + sal * 10/1000 as monthly_hike
from emp
where MGR is NOT NULL and deptno=10;
```

17.) Display all the Employee who are 'Salesman's having 'E' at the last but one character in Ename But Salary having exactly 4 characters.

```
Select * from emp
where job in 'Salesman' and Ename like '%E_' and
Sal like '____';
```

```
where job in 'Salesman' and enam Like '%E'
and sal like '____';
```

where job in 'Salesman' and (Ename like '%E' or Ename like '%E%') and sal like '____';

18.) Display all the employee who are joined after year 81.

```
Select * from emp
where hiredate > '31-DEC-81';
```

19.) Display all the employee who are joined in Feb

```
Select * from emp
where hiredate like '%FEB%';
```

'__-FEB-__';

20) List the employees who are not working as manager and clerks in Dept 10 and 20 with a Salary in the range of 1000 to 3000.

```
Select * from emp
where job NOT IN ('manager', 'clerk') and
deptno in (10, 20) and Sal
between 1000 and 3000;
```

Functions

It is a set of codes are (or) block of instructions which is used to perform some task

=> In General functions are classified into 2 types

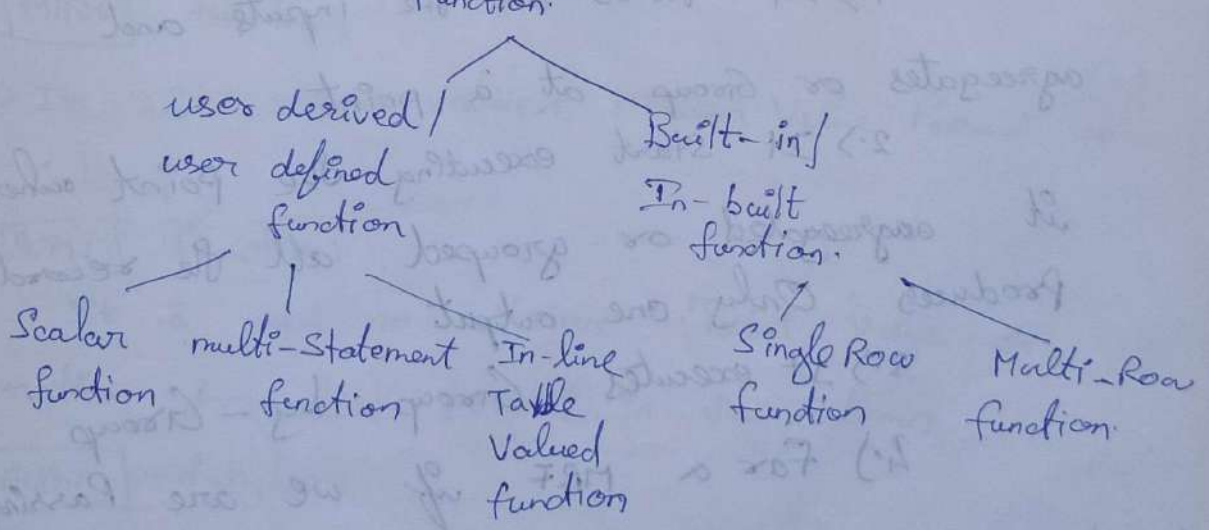
1.) USER DERIVED / USED DEFINED FUNCTION
2.) BUILT-IN / IN-BUILT FUNCTION

1.) User derived function can be classified into 3 types

1.) SCALAR FUNCTION
2.) MULTI-STATEMENT FUNCTION
3.) IN-LINE TABLE VALUED FUNCTION

2.) Built-in function can be classified into 2 type

1.) Single Row function
2.) Multi Row Function.

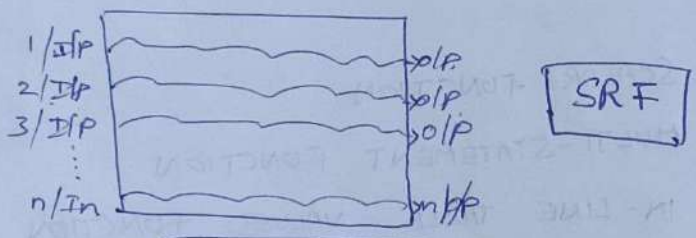


Single-Row Function (SRF)

1.) It takes first input executes and produces output, Later it will take Second input and the Process goes on till the last record.

2.) It executes Row-by-Row

3.) For a SRF if we are Passing N number of inputs we will get N number of ouputs



n(I/P) = n(O/P)

Row-by-Row

Multi-Row Function (MRF)

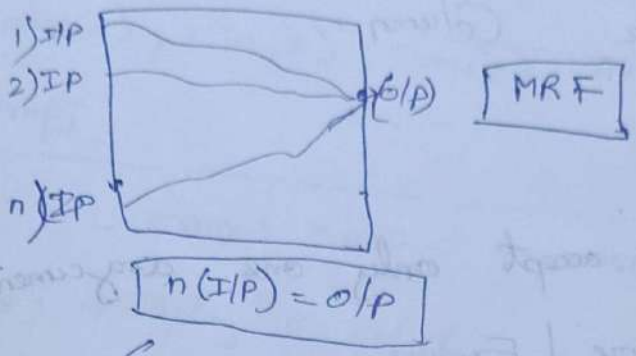
1.) It takes all the inputs and aggregates or Group at a point

2.) It start executing the point where it aggregated or grouped all the record and Produces Only one output

3.) It executes Group-by-Group

4.) For a MRF if we are Passing 'N' number of input we will get only one output

5.) MRF is also called as Group function or Aggregate function



Types of MRF (multi-Row-function)

-> we have five types of MRF

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

1.) MAX() function.

=> It is used to retrieve Maximum Value Present in the Column

2.) MIN() function

-> It is used to retrieve Minimum Value Present in the Column

3.) SUM() function

It is used to retrieve Total Value or Sumation of Values present in the Column

4.) AVG() function

It is used to retrieve AVGERAGE Value Present in the Column

5. COUNT() FUNCTION

It is used to retrieve Number of records present in the Column.

Rules of MRF

1.) MRF Can accept only one argument, that is Columnname | Expression.

Syntax => MRF(Columnname | Expression)

2.) Along with the MRF we are not Suppose to Pass any other argument in Select clause.

error: not a single-group group function.

```
Select sum(sal), Comm
from emp;
```

✗

```
Select Sum(sal), Max(sal), Min(Sal), Avg(sal), Count(sal)
from emp;
```

✓

3.) MRF Ignores NULL Values.

4.) We are not Suppose to Pass MRF in where clause.

error: group function is not allowed here.

```
Select *
from emp
where ~~Sal~~ Sum(Sal+comm);
```

5.) Count is the only MRF which can accept Astisk ~~of can~~ as an argument

```
Select count(*)
from emp;
```

✓

```
Select max(*), min(*), sum(*), Avg(*)
from emp;
```

✗

error :- missing expression

Count (*) => display the details

max (*)
min (*)
sum (*)
Avg (*)

=> [*] will Convert to multiplication. and will act as operator without operands

---

Assignment - E

1.) WAQTD Number of Employees getting Salary less than 2000 in Deptno 10.

```
Select count(sal)
from emp
where sal > 2000 and deptno=10;
```

2.) WAQTD Total Salary needed to pay Employees working as clerk.

```
Select sum(sal)
from emp
where job='CLERK';
```

3.) WAQTD Average Salary needed to pay all Employees

```
Select Avg(sal)
from emp;
```

4.) WAQTD Number of Employees having 'A' as their first Charactes

```
Select count(*)
from emp
where ename like 'A%';
```

5.) WAQTD Number of Employees working as. clerk or manager

```
Select count(*)
from emp
where job='clerk' or job='Manager';
```

6.) WAQTD Total Salary needed To pay Employees. hired in feb

```
Select sum(sal)
from emp
where hiredate like '%FEB%';  '__-FEB-__'
```

7.) WAQTD Number of Employees reporting to 7839 (mgr).

```
Select count(*)
from emp
where mgr = 7839;
```

8.) WAQTD Number of Employees getting Commision in feb deptno 30.

```
Select count(*)
from emp
where comm is not null and deptno =30;
```

9.) WAQTD Avg Sal, Total sal, Number of Emps and maximum Salary Given ufs the employees working as president

```
select avg(Sal), Sum(Sal), count(*), max(Sal)
from emp
where job = 'PRESIDENT';
```

10.) WAQTD Number of Employees having 'A' in their names

```
select count(*)
from emp
where ename like '%A%';
```

11.) WAQTD Number of Emps and Total Salary needed To pay the Employees who have 2 consicutive 2's in their names.

```
select count(*), Sum(Sal)
from emp
where ename like '%LL%';
```

12.) WAQTD NO of Departments present in Employee table

```
select count(deptno)
from emp;
```

```
select count(distinct deptno)
from emp;
```

13.) WAQTD N.of.E. having character '2' in their names.

```
select count(*)
from emp;
where ename like '%2%';
```

14.) WAQTD N. of E Having '$' in their names

```
Select count(*)
from emp
where ename like '%$%';
```

15.) WAQTD Total Salary Given to Employees working as clerk in Dept 30.

```
Select Sum(sal)
from emp
where job = 'CLERK' and deptno = 30;
```

16.) WAQTD maximum Salary given to the Employees working as Analyst

```
Select max(sal)
from emp
where job = 'ANALYST';
```

17.) WAQTD N. of Distinct Salaries present in employee table. Distinct Sal -> 12
Distinct Count(Sal)

```
Select Count(Distinct Sal);
from emp;
```

18.) WAQTD N. of Jobs present in Emp table

```
Select Count(job)
from emp;
```

```
Select count(Distinct job)
from emp;
```

19. WAQTD Avg Salary Given to the clerk.

```
select avg(sal)
from emp
where job='clerk';
```

20. WAQTD MINIMUM Salary Given to the Employees who work in dept 10 as manager or a clerk.

```
select min(sal)
from emp
where deptno=10 and job in('manager','clerk');
```

SINGLE ROW FUNCTION

1.) LENGTH()
2.) CONCAT()
3.) UPPER()
4.) LOWER()
5.) INITCAP() -> Initial Capital letter
6.) REVERSE()
7.) SUBSTR()
8.) INSTR()
9.) REPLACE()
10.) MOD()
11.) TRUNC()
12.) ROUND()
13.) TO-CHAR()
14.) MONTH_BETWEEN()
15.) ADD_MONTHS()
16.) LAST_DAY()

1) LENGTH()

used to Retrieve the No. of charaters or digits of a Given String Contains

Syntax → LENGTH('STRING')

eg:-1 Selct Length ('QSPIDERS') as "Length"
from DUAL;

o/p

Length
8

eg:-2 Select length ('BHAVITHA') as name
from Dual;

o/p

name
8

2) CONCAT()

It is used to join only two strings

Syntax → CONCAT ('STRING 1', 'STRING 2')

eg1:- select concat ('abc_', 'def') as add
from Dual;

o/p

add
abc def

eg2:- select Concat ('abc', 'def', 'ghi') as ad

error: invalid number of arguments

eg:3:- select 'QRS' || 'WXY' || 'UVW' as add
from dual;

o/p
add
QRSWXYUVW

3.) UPPER()

It will Convert the Given String into Capital letters

Syntax → UPPER('STRING')

eg:- select UPPER('dinga') as "Capital"
from dual;

o/p
Capital
DINGA

4.) LOWER()

It will Convert the Given String into Small Letters

Syntax → lower('STRING')

eg:- select lower("DINGA") as "Lower"
from dual;

o/p
Lower
dinga

5.) INITCAP()

It will Convert the Starting Character of a given String into Capital letter and the remaining characters into Small Letters

Syntax → INITCAP('STRING')

eg:- Select initcap('dinga')
from dual;

Dinga

6.) REVERSE ()

It is used to Transpose the Given String
It is used to (or) Reverse the Given String

Syntax => REVERSE ('STRING')

eg:- Select reverse ('DINGA') as reverse
from dual;

o/p

reverse
AGNID

10.) MOD()

It is used to retrieve the remainder of the given number

Syntax -> MOD(x, y)

eg: Select mod(5,2)
from dual;

12.) ROUND()

2.5 → 25

It is used to round of the Given number based on the decimal Value

If we are passing Greater than are equal to .5 as a decimal Value it will round of the Given number to the next Value

If we are passing .5 as a decimal value it will round off the given number to the same value

Syntax -> Round (NUMBER)

eg:- Select ROUND(9.5)
from dual;

O/P

Round
10

eg:- Select ROUND(9.3)
from dual;

O/P

Round
9

11.) TRUNC()

It is similar to the Round function but it will always round off the given number to the same value irrespective of decimal value

Syntax -> TRUNC (NUMBER)

eg:- Select trunc(9.5) from dual;

O/P

trunc(9.5)
9

eg:- Select trunc(9.3) from dual;

O/P

trunc(9.3)
9

(i) CURRENT_DATE
(ii) SYSDATE
(iii) SYSTIMESTAMP

← SQL DATE Commands

Current_date

→ Select current_date from dual;

o/p

13-SEP-23

SYSDATE

→ Select SYSDATE from dual;

o/p

13-SEP-23

SYSTIMESTAMP

→ Select SYSTIMESTAMP from dual;

13-SEP-23 17.14.22.572006 +05:30

13) TO_CHAR()

It will Convert the given date into String formate based on the formate models

Syntax → TO_CHAR('DATE', 'FORMAT_MODELS')

YEAR : TWENTY TWENNTY-THREE
YYYY : 2023
YY : 23
MONTH : SEPTEMBER
MON : SEP
MM : 09 (month of the year)
DAY : WEDNESDAY
DY : WED
D : 04 (Day of the week)
DD : 13

HH12 : 05
HH24 : 17
MI : 26
SS : 58

HH12 : MI : SS => 05 : 26 : 58
HH24 : MI : SS => 17 : 26 : 58

eg :- 1) Select to_char(sysdate, 'year') from dual

o/p

TWENTY TWENTY-THREE

2.) select to_char(Sysdate, 'DY') from dual;

o/p

WED

3.) select to_char(Sysdate, 'year, MON, DD, HH12) from dual;

o/p

TWENTY TWENTY-THREE, SEP, 13, 05

Assignment - 5

Group by clause

1.) WAQTD No. of Emps working in each department except president

```
Select count(*), Deptno
from emp
where job <> 'THALAKARTHIK'
Group by deptno;
```

2.) WAQTD Total Sal needed to pay all the Employees in each job

```
select job, Sum(Sal) "Total Salary"
from emp
group by job
```

3.) WAQTD No. of Emps working as manager. in each dept

```
Select count(*), deptno
from emp
where job='MANAGER'
Group by deptno;
```

4.) WAQTD Avg Sal needed to pay all the emps in each dept excluding the Emps of dept 20.

```
Select deptno, avg(sal)
from emp
where deptno <> 20
Group by deptno;
```

5.) WAQTD No. of Emps having char 'A' in their Names in Each job

```
Select job, Count(*)
from emp
where ename like '%A%'
group by job;
```

6.) WAQTD No. of Emps and Avg sal needed to pay the employees who sal in greater than 2000 in each dept.

```
Select count(*), deptno, Avg(sal)
from emp
where sal > 2000
group by deptno;
```

7.) WAQTD Total Sal needed To Pay and No. of Salesman in each dept

```
Select deptno, Count(job), Sum(sal)
from emp
where job = 'Salesman'
group by deptno;
```

8.) WAQTD No. of Emp with their MAX(Sal) in each job

```
select count(*), MAX(sal), job
from emp
group by job;
```

9.) WAQTD MAX(Sal) Given to an Emp working in each dept

```
Select max(sal), deptno
from emp
group by deptno;
```

10.) WAQTD Number of Times the Salary present in Emp table.

```
Select count(Sal), Sal
from emp
group by Sal;
```

WAQTD No. of Employees working in deptno 10

```
Select count(*) from emp
where deptno = 10;
```

WAQTD No. of Employees working in deptno 20

```
Select count(*) from emp
where deptno = 20;
```

WAQTD No. of Employees working in deptno 30

```
Select count(*) from emp
where deptno = 30;
```

WAQTD No. of Employees working in each dept group by

✗ select count(~~dept~~ *) from emp group by
where deptno ~~=10~~ in (10

✓
```
select count(*)
from emp
group by deptno;
```

GROUP BY CLAUSE

1.) It is used to Group the records

→ for group by clause we can Pass Columnname/Expression as an argument

2.) It executes Row-by-Row

3.) With or without using where clause we can use group by clause

4.) After execution of Group by Clause we will get groups.

5) if any clause wants to execute after the execution of Group by clause it must execute Group by Group.

6) The Columnname / Expression which is written in the Group-by-clause can be used in Select clause

Syntax

```
SELECT GROUP_FUNCTION / GROUP-BY-EXPRESSION
FROM TABLE-NAME
[WHERE <FILTER CONDITION>]
GROUP BY COLUMN-NAME / EXPRESSION;
```

Order of execution

FROM
WHERE (If used) (Row-by-Row)
GROUP BY (Row-by-Row)
SELECT (Group-by-Group)

```
Select count(*)
fromemp
group by dno;
```

emp

eid	ename	sal	dno
1	AAA	10	5
2	BBB	20	4
3	CCC	30	5
4	ddd	40	3

o/p from clause

5
1 AAA 10 5
3 CCC 30 5

4
1 BBB 20 5

3
4 ddd 40 3

o/p of Group_by_clause

o/p

Count(*)
2
1
1

```
Select count(*)
from emp;
Group by dno;
```

o/p

count(*)
2
1
1

```
Select dno, count(*)
from emp
group by dno;
```

o/p

dno	count(*)
5	2
4	1
3	1

```
Select sal, count(*)
from emp
group by dno;
```

error :-
not a Group by expression.

WAQTD MAX Salary and Min Salary of the Emps in each job if they hired after 1982.

```
select job, MAX(Sal), MIN(Sal)
from emp
where hiredate > '31-DEC-1982'
group by job;
```

---

o/p

(*) times
2
1
1

HAVING CLAUSE

* It is used to filter the Groups
* for HAVING CLAUSE we can Pass Group filter Condition as an argument
* Without using Group by clause we are not Suppose to use Having clause
* Having clause executes after the execution of Group by Clause
* Having clause group by group
* We Can Pass multi Row Functions (MRF) for Having clause
* we Can Pass Multiple Conditions for the Having clause by using logical operators (and, or, not)

Syntax:

```
Select group function / group by expression
from Tablename
[where <filter_Condition>]
group by Columnname / Expression
Having_ filter_ Condition>;
```

order of execution:

from
where (if used) → Row-by-Row
group by → Row-by-Row
having → Group by Group
Select → Group by Group

WAQTD No. of Emps working in each dept if there are two employees working in each dept

```
Select count(*), deptno
from emp
group by deptno
having count(*)=2;
```

Group - ~~by~~ filter - Condition

Emp table under execution.

S.No	Name	deptno
1	AAA	40
2	BBB	41
3	ccc	30
4	ddd	41
5	eee	30

o/p from clause

40

1	AAA	40

41

2	BBB	41
4	ddd	41

30

3	ccc	30
5	eee	30

o/p group clause

count(*) = 2.

2 = 1 ×
2 = 2 ✓
2 = 2 ✓

2	BBB	41
4	ddd	41

3	ccc	30
5	eee	30

o/p - Having clause

count(*)	deptno
2	41
2	30

o/p - select clause

WAQTD No. of emps, max(sal), min(sal) in each dept if the emps are working as manager and max(sal) more the 2000.

```
Select count(*), max(sal), min(sal), deptno
from emp
where job = 'MANAGER'
group by deptno
having max(sal) > 2000;
```

max(Sal) of emps in each job if their total Sal more than 3000

```
select job, max(sal)
from emp
group by job
having sum(sal) > 3000;
```

Syntax

Group-filter-Condition for Having clause.

Groupfunction operator Value

Diff between where clause / Having clause

WHERE clause	HAVING clause.
→ It is used to filter the records	→ It is used to filter the groups.
→ for where clause we can pass filter condition as an argument	→ For Having clause we can pass Group filter Condition as an argument
→ It executes after the execution of from clause	→ It executes after the execution of Group by clause.
→ It executes Row-by Row	→ It executes Group-by-Group
→ We can't Pass MRF in where clasuse	→ We Can pass MRF in Having Clause

→ | Columnname | Expression operator Value |

Syntax for filter condition in where clause

| Groupfunction operator Value |

Syntax for Group filter Condition in Having clause

Syntax

Formate of Select, from, where, Group by, Having & Group order by clauses (Combination)

```
Select COLUMNNAME | EXPRESSION
FROM TABLENAME
[WHERE <FILTER_CONDITION>]
[GROUP BY COLUMNNAME | EXPRESSION]
[HAVING <GROUP FILTER CONDITION>]
ORDER BY COLUMNNAME / EXPRESSION [ASC] | DESC;
```

order of execution.

From ——— Row-by-Row
where (if used) ——— Row-by-Row
Group by (if used) ——— Group-by Group
Having (if used) ——— Group-by-Group
Select ——— Row-by-Row
order by ———

ASSIGNMENT-6 Having clause

1) WAQTD deptno and No. of emps working in each dept in there are atleast 2 clerks in each dept.

```
Select deptno, count(*)
from emp
where job = 'clerk'
Group by deptno Having Count(*)=2
```

2) WAQTD deptno and Total Salary needed to pay all emps in each dept if there are atleast 4 emps in each dept

```
Selet deptno, Sum(sal)
from emp
group by deptno
having count(*) >=4;
```

3) WAQTD No. of emps earning sal more than 1200 in each job and the Total Sal needed to pay emp of Each job must excees 3800.

```
Select count(*)
from emp
where sal > 1200
Group by job
having Sum(sal) > 3800;
```

4) WAQTD deptno and No. of Emps working only if there are 2 emp working in each dept as manager.

```
select deptno, Count(*)
from emp
where job = 'MANAGER'
Group by deptno having Count(*) = 2;
```

5.) WAQTD job and max Sal of Emp in each job if the max Sal exceeds 2600.

```
Selet job, max(sal)
from emp
group by job
having max(sal)>2600;
```

6.) WAQTD the Salaries which are repeated in emp table

```
Select count(sal), sal
from emp
Group by sal
having count(sal)>=2;
```

7.) WAQTD the Hiredate which are duplicated in emp table

```
Select distinct Hiredate
from emp;
```

```
select count(hiredate), hiredate
from emp
group by hiredate
having count(hiredate)>=2;
```

8.) WAQTD Avg Salary of each dept if avg Sal is Less than 3000.

```
Selet Emp Avg(sal), deptno
from emp
group by deptno
having avg(sal)<3000;
```

9.) WAQTD · Deptno if there are atleast 3 emp in each dept whos ane has chas 'A' or 'S'

```
select deptno, ename
from emp
where Ename like '%A%' and ENAME LIKE '%S%'
Group by deptno
having count(*) <=3;
```

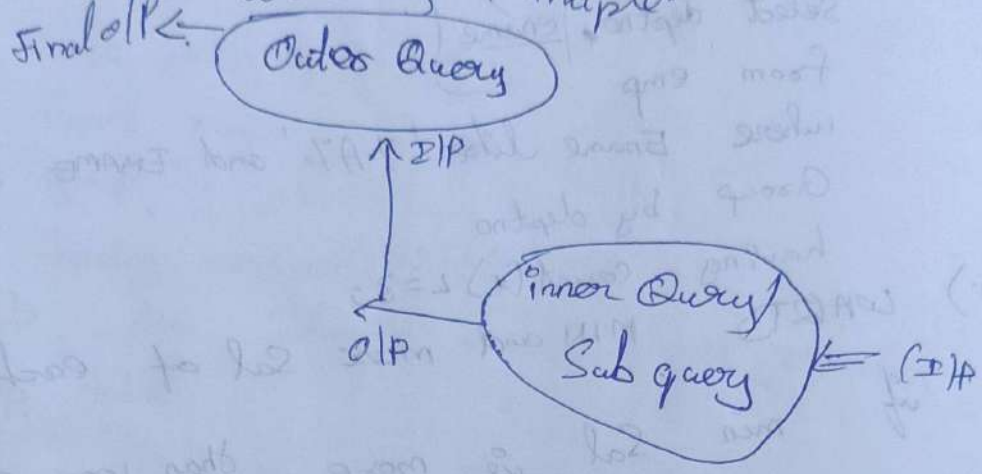
10.) WAQTD MIN and max Sal of each job if min Sal is more than 1000 and max Sal is less than 5000.

```
Select min(sal), max(sal), job
from emp
group by job
having min(sal) > 1000 and max(sal) <5000;
```

Sub Query

A Query written inside a Query is known as Sub Query

working Principle.



Let Us Consider that is Outer Query and Inner Query

=> Inner Query will Starts the Execution and Produces output

=> The Output of Inner Query is Given as an Input to the outer Query

=> After Taking the Input, the Outer Query will Starts Execution and produces final Output

=> By Seeing this we Can Say that the Outer Query is Completely Dependent on inner Query

This is the Working Priciple of Sub Query

WHERE(ex) WHY do we use Sub Query

Emp

EID	Ename	Sal	Dno
1	ALLEN	1000	10
2	ADAMS	2000	20
3	MILLER	3000	30
4	SCOTT	1500	10
5	SMITH	2500	30

WAQTD details of the Smith

```
Select *
from emp
where Ename = 'Smith';
```

WAQTD Sal of the Smith

```
Select Sal. from emp
where Ename = 'Smith';
```

WAQTD Names of the Employees if they are Earning Sal less than Smith.

wh Select ename → ALLEN, ADAMS, SCOTT
from emp
where sal < Select sal → o/p 2500
from emp
where ename = 'smith';

Case 1 :- Whenever the unknowns are Present in the Question we use Sub Query to find the unknowns.

WAQTD Details of the emps. if they are working in Same dept as ALLEN

Select * → 1 ALLEN 1100 10
4 SCOTT 1500 10
from emp
where deptno = (Select deptno → 10
from emp
where ename = 'ALLEN');

WAQTD Name and Hiredate of the emps if they hired after Miller

Select ename, hiredate
from emp
where hiredate > (Select hiredate
from emp
where ename = 'Miller');

WAQTD Name, Sal, and designation of emps if they working as manager and earning Sal more that Scott

Possible ✓

```
Select ename, Sal, job
from emp
where job='manager' and Sal > ( Selct Sal
                                from emp
                                where ename='Scott');
```

✓

```
Select ename, Sal, job
from emp
where sal > (Select Sal
             from emp
             where ename = 'Scott') and
      job = 'manager';
```

WAQTD D of Employees if they Hired after 1982 and working in same designation as Smith

```
Select *
from emp
where job = (Select job
             from emp
             where ename='Smith') and
      hiredate > ~~01-JAN-1~~
                 31-DEC-1982';
```

```
Select *
from emp
where hiredate > '31-DEC-1982' and job =
                 ( select job
                   from emp
                   where ename = 'Smith');
```

WAQTD D of Emps if they earning Sal more than Smith less than Miller.

```
Select *
from emp
where sal > (Select sal
             from emp
             where ename='Smith') and sal < (select sal
                                             from emp
                                             where ename=
                                             'Miller');
```

WAQTD D of emps if they Hired after Scott and working in same dept as Adams.

```
select *
from emp
where hiredate > (select hiredate
                  from emp
                  where ename = 'Scott') and
      deptno = (select deptno
                from emp
                where ename = 'Adams');
```

WAQTD Details of the emps if they earning Sal more than 1000 Rupees and working in same designation as 'Scott' but not as Smith.

```
select *
from emp
where sal >1000 and job =(select job
                          from emp
                          where ename='scott') ~~NOT~~ AND
      job=(select job
           from emp
           where ename='smith');
```

Diff b/w filter Condition and Group filter Condition.

Filter-Condition	Group Filter Condition
→ It is used to filter the records	→ It is used to filter the Groups
→ filter-Condition is an argument for where clause	→ Group filter Condition is an argument for Having class
Syntax: Columnname / Expression operator Value	Syntax: Group filterCondion: Group function operator Value.

WAQTD details of emp if they are in dept 10, 20, 30 if Annual Sal greated than Smith and hired after allen and Comm is not null

=> Select * from emp
where deptno in (10, 20, 30) and
Sal*12 > (Select Sal*12 from emp
where ename = 'smith') AND Hiredate >(
Select hiredate from emp
where ename = 'ALLen') and comm is NOTNULL;

Case 2:- Whenever the data to be Displayed and Condition to be executed is present in different table Subquery is used.

EMP

ED	ENAME	SAL	DNO
1	ALLEN	1000	10
2	ADAMS	2000	20
3	MILLER	3000	30
4	SCOTT	1500	10
5	SMITH	2500	20

DEPT

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

WAQTD

DeptName of Scott

Select Dname (D1)
from dept
where DNO = (Select DNO (10)
from emp
where ename = 'SCOTT');

10 = 10 ✓
20 = 10
30 = 10

WAQTD Loc of MILLER [L3]

```
Select loc
from dept
where dept = (Select dept
              from emp
              where ename = 'MILLER');
```

10 = 30
20 = 30
30 = 30 ✓

WAQTD Names of the employees if they are working in Newyork Location.

```
Select enames
from emp
where deptno = (select deptno
                from dept
                where loc = 'New YORK');
```

WAQTD Details of Emps if they are working in same designation as Miller and earning Sal same as Scott and working in Research dept

```
Select *
from emp
where job = (select job
             from emp
             where ename = 'Miller') and
      sal = (select sal
             from emp
             where enam = 'scott') and
      deptno = (select deptno
                from dept
                where dname = 'RESearch');
```

WAQTD Details of emps if they are working in operations dept and hired after ALLEN but before Adams.

```
Select *
from emp
where hiredate > (Select hiredate
                  from emp
                  where ename = 'ALLEN') and
      hiredate < (Select hiredate
                  from emp
                  where ename = 'ADAMS') and
      deptno = (Select deptno
                from dept
                where dname = 'operations');
```

Assignment – H

Sub Query Case - I

1.) WAQTD Name of employees earning more than adams.

```
Select *
from emp
where sal > (select sal
             from emp
             where ename = 'ADAMS');
```

2.) WAQTD Name and Salary of emps earning less than King

```
Select ename, Sal
from emp
where Sal < (Select Sal
             from emp
             where ename='King');
```

3.) WAQTD Name and deptno of the emps if they are working in same dept as jones.

```
select ename, deptno
from emp
where deptno = (select deptno
                from emp
                where ename='jones');
```

4.) WAQTD name and job of all the emps working in same designation as james

```
select ename, job
from emp
where job = (select job
             from emp
             where ename='james');
```

5.) WAQTD empno and ename along with annual Sal of all the employees if their annual Sal is greater than ward annual Sal.

```
select empno, ename, Sal*12 "Annual Sal"
from emp
where sal*12 > (Select sal*12
                from emp
                where ename = 'ward');
```

6.) WAQTD name and hiredate of the emplee if they are hired before Scott

```
Select ename, hiredate
from emp
where hiredate < (Select hiredate
                  from emp
                  where ename='Scott');
```

7.) WAQTD Name and hiredate of employees if they are hired after the President

```
Select ename, hiredate
from emp
where hiredate > (Select hiredate
                  from emp
                  where ename = 'President');
```

8.) WAQTD Name and Sal of emp if they are earning Sal less than the emp whos empno is 7839.

```
Selct ename and Sal
from emp
where ~~empno = 7839 and~~ Sal < (Select Sal.
                                  from emp
                                  where empno = 7839);
```

9.) WAQTD all the details of Emps if the emps hired before miller

```
Select *
from emp
where hiredate < (select hiredate
                  from emp
                  where ename='Miller');
```

10) WAQTD Ename and empno of the emps if emps are earning more than allen.

```
Select ename and empno
from emp
where sal > (Select Sal
             from emp
             where ename='ALLEN');
```

11) WAQTD ENAME and Sal of all the emps who are earning more than miller but less the allen.

```
Select ename, Sal
from emp
where sal > (Selct sal                and sal < (select sal
             from emp                            from emp
             where ename='miller')               where enam = 'allen');
```

12) WAQTD All the details of emps working in dept 20 and working in the same designation as Smith

```
Select * from emp
where deptno=20 and job = (select job
                           from emp
                           where ename = Smith)
```

13) WAQTD all the details of emps working as manager in the same dept as Turner

```
Select * from emp
where job = manager and deptno = (Select deptno
                                  from emp
                                  where enam='Turner')
```

14.) WAQTD Name and hiredate of the emps hired after 1980 and before king.

```
Select ename and hiredate
from emps
where hiredate > '31-DEC-1980' and hiredate < (Select hiredate
                                               from emp
                                               where ename='king');
```

15.) WAQTD Name and Sal along with annual Sal for all emps who Sal is less than blake and more than 3500

```
Select ename, sal, Sal*12
from emps
where sal >3500 and sal < (Select Sal
                           from emp
                           where ename='blake');
```

16.) WAQTD All the dept d.of emps who earn more that Scott but Less than king.

```
select *
from emp
where sal > (select sal                sal < (Select Sal
             from emp                         from emp
             where ename='scott') and         where ename='king');
```

17.) WAQTD Name of the emps whose Name Starts with 'A' and works in the Same dept as blake

```
Select ename
from emp
where ename like "A%." and deptno = (Select deptno
                                     from emp
                                     where ename='blake');
```

19.) Select ename, comm
from emp
where comm is NOT NULL and job = (Select job from emp
where ename = 'Smith');

19.) Select *
from emp
where job = 'clerk' and deptno = (Select deptno
from emp
where ename = 'TURNER');

20) Select ename, Sal, job
from emp
where sal * 12 > (Select Sal * 12 from emp
where ename = 'Smith') and Sal * 12 < (Select Sal * 12
from emp where ename = 'KING');

21.) Select dname from dept
where deptno = (Select deptno from emp
where ename = 'Smith');

22.) Select dname, loc from dept
where deptno = (Select deptno from emp
where ename = 'king');

23.) Select loc from dept
where deptno = (Select deptno
from emp
where empno = 7902);

24.) Select dname, loc, deptno
from dept
where deptno in (Select deptno
from emp
where ename like '%R');

25.) Select dname
from dept
where deptno = (select deptno
from emp
where job = 'president');

26.) Select ename from emp
where deptno = (select deptno
from dept
where dname = 'Accounting');

27) Select ename, sal.
from emp
where deptno = (Select deptno from dept
where loc = 'Chicago');

28.) select * from emp
where deptno = (Select deptno from dept
where dname = 'Sales');

29.) Select emp.*, sal * 12 Annual_Salary
from emp where deptno = (Select deptno
from dept
where dname = 'New York');

30) Select ename
from emp
where deptno = (Select deptno from deptno
from dept
where dname = 'Operations');

Combination of Case1 and Case 2

31.)
```
select ename
from emp
where sal > (select sal from emp
             where ename='SCOTT') and
      deptno = (select deptno
                from dept
                where dname = 'ACCOUNTING');
```

32.)
```
Select *
from emp
where job = 'Manager' and deptno = (Select deptno
                                    from dept
                                    where loc = 'Chicago');
```

33.)
```
Select ename, sal
from emp
where job = (select job
from emp
where ename = 'KING') and deptno = (select deptno
from dept
where dname = 'ACCOUNTING');
```

34.)
```
Select *
from emp
where job = 'SALESMAN' and deptno = (select deptno
from dept
where dname = 'SALES');
```

35.)
```
Select ename, sal, job, hiredate from emp
where deptno = (select deptno from dept where
dname = 'Operation') and job = (select job
                                fromemp
where ename = 'KING');
```

36.)
```
Select *
from emp
where deptno in(Select deptno
                from dept
                where dname like '%S');
```

37.)
```
Select dname
from dept
where deptno = (Select deptno
                from emp
                where ename like '%A%') and
                dname = IT;
```

38.)
```
Select dname, loc.
from dept
where deptno = (Select deptno
                from emp,
                where Sal= 800);
```

39.)
```
Select dname
from dept
where deptno = (Select deptno
                from emp
                where comm is NOT NULL);
```

40.)
```
select loc
from dept
where deptno = (Select deptno
                from emp
                where comm is NOT NULL and deptno=4);
```

WAQ Deptname of Allen, Miller, Adams

```
select dname
from dept
where deptno = (select deptno
from emp
where ename in ('Allen', 'Miller', 'Adams'));
```

Here, SubQuery is producing more than 1 output so we can't able to use normal operators we must use special operators to compare the values

ex:- Single-row SubQuery returns more than one row

Types of SubQuery

Based On No. of Output what we are getting from the Subquery we can classify SubQuery into two types

-> Single-Row SubQuery
-> Multi-Row SubQuery.

Single-Row SubQuery

-> If the SubQuery is Producing exactly one output then that type of SubQuery is known as Single-Row SubQuery

-> If the SubQuery is Producing exactly one output either normal operators or special operators to Compare the Values.

Multi-Row SubQuery

=> If the SubQuery is Producing more than one Output that that type of SubQuery is known as Multi-Row SubQuery.

=> If the SubQuery is Producing more than one output we are not Suppose to use normal operators we must use Special Operators to Compare the Values.

eg:- Select Dname
from dept
where deptno in (Select deptno
from emp
where ename in ('Allen', 'Miller', 'Adams'));

WAQTD Names of the emps if they are earning Sal less than the emps of deptno 30.

Select enames
from emp
where deptno in (Select deptno
from emp
where deptno = 30);

→ Here Subquery is producing more than one output So we can't able to use normal operators Sal. we can't able to use in and non in operators. Since those are the Special operators Since those are the Special operators to Equals and not returns more than one row

or:- Single-row Subquery

here we have to use Subquery operators along with the relational operators.

Sub-Query Operators

=> We have 4 types of Sub-Query Operators

- > ALL
- > ANY
- > EXISTS
- > NOTEXISTS

ALL

-> it is a Subquery operator which is used along with the relational operator

-> it returns true value if all the Conditions are satisfied.

```
select ename
from emp
where sal < ALL (select sal
                 from emp
                 where dno = 30);
```

Sal < All (3000, 2500)

i) 1000 < All (3000, 2500)

ii) 2000 < All (3000, 2500)

iii) 3000 < All (3000, 2500)

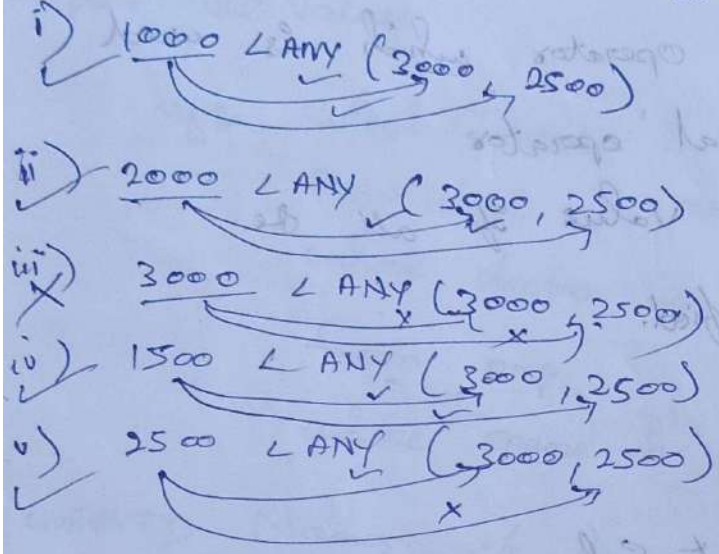
iv) 1500 < All (3000, 2500)

v) 2500 < All (3000, 2500)

ANY

It is a Subquery Operator which is used along with the relational operator. it returns true. value if any one of the condition is satisfied

eg:- Select ename.
from emp
where sal < ANY (Select sal
from emp
where deptno = 30);



Nested SubQuery

→ A SubQuery inside SubQuery is called as Nested SubQuery

→ we can nested upto 255 Subquery

Emp

Sal
1000
3000
4000
5000
2000

1) WAQTD MAX(Sal) of emp?

```
Select max(Sal)
from emp;
```

2) WAQTD Second max Sal of emp

```
Select max(sal)        [4000]
from emp
where sal < Select max(sal)    [5000]
                from emp);
```

✓ 1000 < 5000
✓ 3000 < 5000
✓ 4000 < 5000
✗ 5000 < 5000
✓ 2000 < 5000.

WAQTD 3rd Maximum Sal of emp

```
Select MAX(Sal)
from emp
where sal. < (Select Max(sal)
              from emp
              where Sal < (Select Max(Sal)
                           from emp));
```

WAQTD 4th min Sal of emps.

```
Select Min(sal.)
from. emp
where sal > (select min(Sal)
             from emp
             where Sal > (Select Min(Sal)
                          from emp
                          where Sal > (Select min(Sal)
                                       from emp; )))).
```

Employee - Manager Relation.

Emp

EID	ENAME	MGR
1	ALLEN	3
2	ADAMS	1
3	MILLER	2

Case-1 => To find Manager.

WAQTD Name of Adams Manager.

```
Select ename
from emp
where EID = (Select MGR
             from emp
             where ename = 'ADAMS');
```

1 = 1
2 = 1 x
3 = 1 x

WAQTD ALLENS Manager Name.

```
Select ename
from emp
where EID = (Select MGR
             from emp
             where ename = 'ALLENS');
```

WAQTD ID of Millers Manager.

```
Select *
from emp
where EID = (Select MGR
             from emp
             where ename = 'Millor');
```

Case-II → To find Employees

WANTED Names of the employees if they are Reporting to Miller

```
Select ename
from emp
where mgr = (Select EID
             from emp
             where ename = 'Miller');
```

3 = 3 ✓
2 = 3 ×
1 = 3 ×

WANTED No. of emps Reporting to Allen

```
Selent count(*)
from emp
where MGR = (Select EID
             from emp
             where ename = 'Allen');
```

3 = 1 ×
1 = 1 ✓
2 = 1 ×

To FIND MANAGER -> ~~Sub~~ Select MGR in Sub Query
To FIND EMPLOYEES => Select EID in Sub Query

How to Explain SubQuery

=> What is Sub Query
=> Working Procedure of Sub Query
=> When or why do we use Sub Query
=> Types of Sub Query
=> Sub Query Operators.
=> Nested Sub Query

Assignment-1 (Types of Sub Query)

1.)
```
Select ename
from emp
where sal > (select sal
             from emp
             where job = 'SALESMAN');
```

2.)
```
Select *
from emp
where hiredate > (select hiredate
                  from emp
                  where job = 'CLERK');
```

3.)
```
Select ename, sal.
from emp
where sal < (select sal
             from emp
             where job = 'MANAGER'
             Group by job
             Having count(job) = 1);
```

4.)
```
Select ename, hiredate
from emp
where hiredate <ALL (select hiredate
                     from emp
                     where job = 'MANAGER');
```

5.)
```
Select ename.
from emp
where hiredate >ALL ( select hiredate
                      from emp
                      where job = 'MANAGER') and
      sal > ALL ( select sal.
                  from emp
                  where job = 'CLERK');
```

6.)
```
select *
from emp
where job = 'CLERK' and hiredate < (select hiredate
                                    from emp
                                    where job = 'Salesman'
                                    group by job
                                    having count(job) = 1);
```

7.)
```
select *
from emp
where deptno in (select deptno
                 from ~~emp~~dept
                 where dname = 'Accounting' or dname = 'Sales');
```

8.)
```
select dname
from dept
where deptno in (select deptno
                 from emp
                 where ename = 'SMITH' and ename = 'KING' and
                                                   ename = 'Miller');
```

9.)
```
Select *
from emp
where deptno in (select deptno
                 from dept
                 where loc = 'NEW YORK' and loc = 'CHICAGO');
```

10.)
```
Select ename
from emp
where hiredate > All (select hiredate
                      from emp
                      where deptno = 10);
```

11.)
```
Select ename
from emp
Group by sal
having max(sal);
```

12.)
```
Select ename
from emp
group by sal
having min(sal);
```

13.)
```
Select ename, hiredate
from emp
where hiredate <ALL (Select hiredate
                     from emp
                     Group by hiredate
                     Having min(hiredate));
```

14.)
```
Select ename, hiredate
from emp
where hiredate = (select MAX(hiredate)
                  from emp);
```

15.)
```
Select ename, comm.
from emp
where comm. = (select min(comm)
               from emp);
```

16.)
```
Select ename., sal, comm.
from emp.
where comm.= (Select max(comm)
              from emp);
```

17.)
```
Select *
from emp
where empno = (Select max(empno)
               from emp);
```

18.)
```
select *
from emp
where hiredate = (select min(hiredate)
                  from emp);
```

19.)
```
select *
from emp
where sal*12 = (select min(sal*12)
                from emp);
```

20.)
```
select ename, sal*12.
from emp
where sal*12. > All (select sal*12
                     from emp);
```

21.)
```
select min(sal)
from emp
where sal = (select min(sal)
             from emp);
```

22.)
```
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
                                       where sal = (select
                                                    max(sal)
                                       from emp))));
```

23.)
```
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)));
```

24.) Select empno
from emp
where sal <= (select max(sal)
from emp
where sal <= (select max(sal)
from emp)

25.) Select dname
from dept
where deptno = (Select deptno from emp
where sal < (select max(sal)
from emp
where sal < (select max(sal)
from emp
where sal < (Select max(sal)
from emp)))

26.) Select * from emp
where hiredate >= (Select max(hiredate) from emp
where hiredate > (select max(hiredate)
from emp));

27.) Select ename from emp
where hire date > (select min(hiredate) from emp
where hiredate > (select min(hiredate)
from emp)));

28.) Select loc
from dept
where deptno = (select deptno
from emp
where hiredate = (select max(hiredate)
from emp));

29) Select * fromemp
where sal > (Select min(sal) from emp
where sal > (Select min(sal) from emp
where sal > (Select min(sal) from emp
where sal > (Select min(sal) from emp
))))));

30) Select dname
from dept
where deptno = (Select deptno
from emp
where sal < (select max(sal)
from emp where sal < (select max(sal)
fromemp)));

31) Select ename
from emp
where eid = (Select MGR
empno
from emp
where ename = 'SMITH');

32) Select ename
from emp
where eid = (Select MGR
empno
from emp
where ename = 'ADAMS');

33) Select dname
from dept
where deptno = (select deptno
from emp
where empno = (Select mgr
from emp
where ename = 'Jones'));

34.) Select sal
from emp
where eid = (select mgr
from emp
where ename = 'Miller');

35.) Select loc
from dept
where dno = (Select deptno
from emp
where eid = (Select mgr
from emp
where eid = (Select mgr
from emp
where ename = 'Smith')))
deptno

36.) Select ename
from emp
where mgr = (Select eid
from emp
where ename = 'BLAKE');

37.) Select count(*)
from emp
where ~~emp~~ mgr = (select eid
from emp
where ename = 'KING');

39.) Select ename
from emp
where ~~mgr~~ eid (Select eid from emp where eid = Select mg
from emp where ename = 'BLAKE');

38.) Select *
from emp where mgr = (Select eid from emp
where ename = 'JONES');

40.) Select count(*) from emp
where eid in (Select mgr from emp where ename = 'FR')

JOINS

» It is used to Retrieve the data from multiple tables Simultaniusly

When or Why do we use joins / when ever the data to be displayed Present in multiple tables we use Joins

Types of Joins

we have 5 types of Joins

1.) Cartesian Join / CROSS Join
2.) Inner Join / EQUI Join
3.) Natural Join
4.) Self Join
5.) Outer Join
    * Left Outer Join
    * Right Outer Join
    * Full Outer Join

1.) CARTESIAN JOIN

In Cartesian Join a Record from table 1 will be merge ~~will be~~ with all the records of table 2

ORACLE :-
```
SELECT *COLUMN_NAME
FROM TABLENAME1, TABLE_NAME2;
```

eg :-
```
Select *
From EMP, DEPT;
```

ANSI :-
```
SELECT COLUMN-NAME
FROM TABLENAME1 CROSS JOIN TABLENAME2;
```

American National Standard Institute

eg:- select *
from emp cross join dept

EMP

ENAME	DNO
A	30
B	10
C	20

DEPT

DNO	DNAME
10	D1
20	D2
30	D3

=>

EMP	DNO	DNO	DNAME
A	30	10	D1
A	30	20	D2
A	30	30	D3
B	10	10	D1
B	10	20	D2
B	10	30	D3
C	20	10	D1
C	20	20	D2
C	20	30	D3

Column => 2+2 = 4
Row => 3*3 = 9

NOTE 1.) The No. of Columns in the Result Table will be equalent to the Summation of Columns present in Table1 and Table2

2.) The No. of Records Present in the result table will be equalent to the Product of Records present in Table1 and Table2

=> Efficiency will be less in CROSS Join

2.)

INNER JOIN

=> It is used to retrieve only matched Records from the table

=> To Retrieve mached records we have to write join condition.

=> without writing join Condition we can't able to get match records.

Syntax of Join Condition.

TABLENAME1.COLUMN_NAME = TABLENAME2.COLUMN_NAME

ORACLE :-

```
SELECT COLUMN_NAME
FROM TABLENAME1, TABLENAM2
WHERE <JOIN_CONDITION>;
```

Eg:-
```
Select *
from emp, dept
where emp.deptno = dept.deptno;
```

ANSI :-

```
SELECT COLUMN_NAME
FROM TABLENAME1 INNER JOIN TABLENAME2
ON <JOIN_CONDITION>;
```

Eg:-
```
Select *
from emp inner join dept
on emp.deptno = dept.deptno;
```

EMP.DNO = DEPT.DNO

i) 30 = 10 ×
30 = 20 ×
30 = 30 ✓

ii) 10 = 10 ✓
10 = 20 ×
10 = 30 ×

iii) 20 = 10 ×
20 = 20 ✓
20 = 30 ×

ENAME	DNO	DNO	DNAME
A	30	30	D3
B	10	10	D1
C	20	20	D2

3.) NATURAL JOIN

=> It Performs inner join if their is a Relation between the Tables else it Performs Cartesian Join / Cross Join.

ANSI:

```
SELECT COLUMNNAME
FROM TABLENAME1 NATURAL JOIN TABLENANE2;
```

eg:

```
Select *
From emp natural join dept;
```

WAQTD employee name and dept name

```
Select ename, dname
from emp inner join dept
on emp.deptno = dept.deptno;
```

WAQTD Dname, job, loc and hiredate of emps if they are earning sal more than 1000 rupees and hired after 1982.

ANSI

```
Select Dname, job, loc, hiredate
from emp inner join dept
on emp.deptno = dept.deptno
where sal >1000 and Hiredate > '31-DEC-1982';
```

ORACLE

```
Select Dname, job, loc, hiredate
from emp, dept
where emp.deptno = dept.deptno and sal>1000 and
```

Assignment on inner join

1) Select ename, Loc
from emp, dept
where emp.deptno = dept.deptno;

2) select dname, sal
from emp inner join dept
on emp.deptno = dept.deptno
where dname = 'ACCOUNTING';

3) Select dname, sal * 12
from emp inner join dept
on emp.deptno = dept.deptno
where sal > 2340

4) Select dname, ename
from emp inner join dept
on emp.deptno = dept.deptno
where dname like '%A';

5) Select ename, dname
from emp inner join dept
on emp.deptno = dept.deptno
where job = 'SALESMAN';

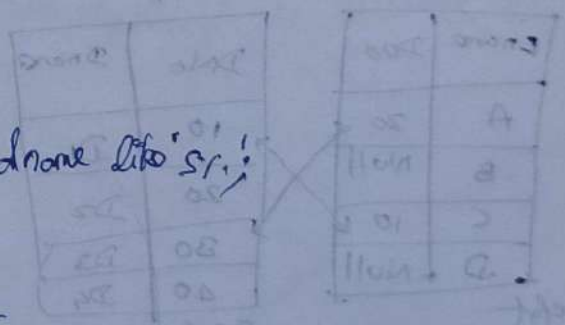
6) Select dname, job
from emp inner join dept
on emp.deptno = deptno.deptno
where ename like 'S%' and dname like 'S%';

7) Select dname, mgr
from emp natural join dept
where mgr = 7839;

8) Select dname, hiredate
from emp inner join dept
on emp.deptno = dept.deptno
where hiredate > '31-DEC-1983'
and dname in ('ACCOUNTING'
, 'RESEARCH');

9) Select ename, dname
from emp natural join
dept where comm is not
null and (deptno = 10 or
deptno = 30)

10) Select dname, eid
from emp natural join
dept where eid in
(7839, 7902) and
loc in 'NEW YORK');



(5) Outer Join

result includes unmatched rows of one of the tables or of both tables

Left Outer Join

It is used to Retrieve unmatched records from left table along with the matched records

ANSI

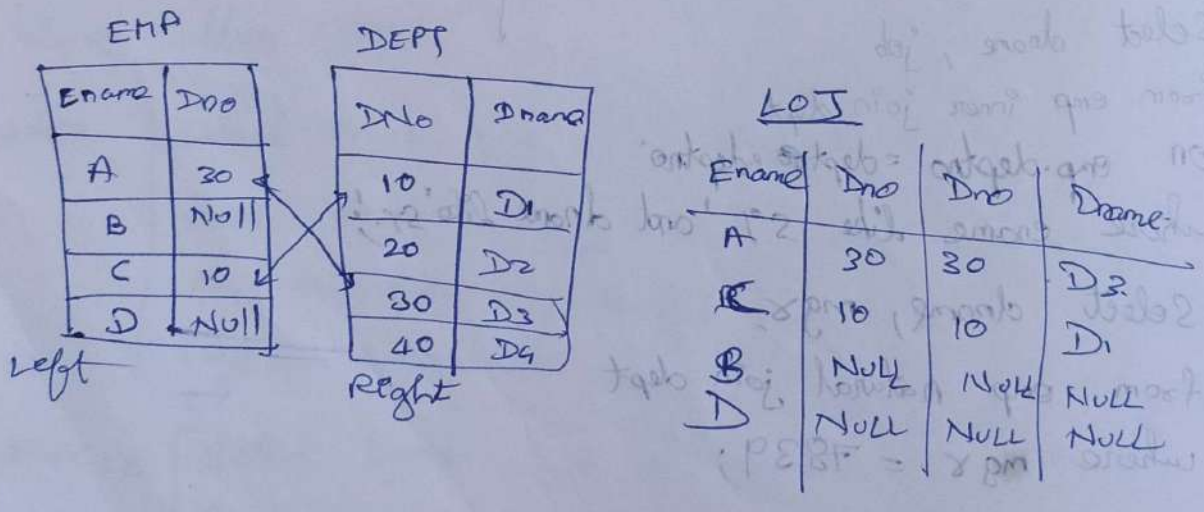
```
SELECT COLUMNNAME
FROM Tablename.1                    Tablename.2
EMP LEFT OUTER JOIN DEPT
ON EMP.DT Tablename1.Columnname = Tablename2.Column Name;
```

eg:- Select ~~Columnname~~ *
from emp left Outer join dept
on emp.~~columnname~~ deptno = dept.deptno;

ORACLE

```
SELECT COLUMNNAME
FROM Table1, Table2
where <Join Condition> (+);
```

eg.) Select *
from emp, dept
where emp.deptno = dept.deptno (+);



Right Outer Join.

It is used to Retrieve unmatched records from Right Table along with the matched records.

Syntax.

ANSI

```
SELECT COLUMNNAME
FROM TABLENAME1 RIGHT OUTER JOIN TABLENAME2
ON <JOIN_CONDITION>;
```

eg:-

```
Select *
from emp right outer join dept
on. emp.deptno = dept.deptno;
```

Oracle

```
SELECT COLUMNNAME
FROM TABLENAME1, TABLENAM2
where (+) <JOIN condition>;
```

eg:-

```
Select *
from emp, dept
where (+) emp.deptno = dept.deptno;
```

ROJ

Ename	Dno	Dno	Dname
A	30	30	D3
C	10	10	D1
NULL	NULL	20	D2
NULL	NULL	40	D4

Full Outer Join:

It is used to Retrieve unmatched Records from both Left and Right Tables along with the matched Records.

ANSI

```
SELECT COLUMNNAME
FROM TABLENAME_1 FULL OUTER JOIN TABLENAME_2.
ON TABLENAME1.COLUMNNAME= TABLENAME2.COLUMNNAME
```

eg:

```
Select *
from emp full outer join dept
on emp.deptno = dept.deptno;
```

FOJ

Ename	DNO	DNO	DName.
A	30	30	D3
C	10	10	D1
B	NULL	NULL	NULL
D	NULL	NULL	NULL
NULL	NULL	20	D2
NULL	NULL	40	D4

(4) SELF JOIN

Joining the Table by itself is known as self join. (or)

Joining the two Similar Tables (or) Same Same tables is known as Self join

when we use Self join

when ever the Data to be displayed present in same table but in different records we use self join

ANSI

```
SELECT COLUMN_NAME
FROM TABLE_NAME T1 JOIN TABLENAME T2.
on T1.COLUMN-NAME = T2.COLUMN-NAME;
```

eg:-

```
select *
from emp E1 join emp E2
on E2.empno = E1.mgr; // E1.mgr = E2.empno
```

ORACLE

```
SELECT COLUMN_NAME
FROM TABLE_NAME T1, TABLENAME_T2
where T1.Columnname = T2.Columnname;
```

eg:-

```
Select *
from emp E1, emp E2
where E1.mgr = E2.empno;
```

Emp E1

Empno	Ename	MGR
1	ALLEN	3
2	ADAMS	1
3	MILLER	2

Emp E2

Empno	Ename	MGR
1	ALLEN	3
2	ADAMS	1
3	MILLER	2

E1.MGR = E2.EmpNo

E1 => Employee Table.

E2 => Manager Table.

Empno	Ename	MGR	Empno	Ename	MGR
1	ALLEN	3	3	MILLER	2
2	ADAMS	1	1	ALLEN	3
3	MILLER	2	2	ADAMS	1

WAQTD ENAME, and Manager name for all the emp

```
Select E1.ename, E2.ename
from emp E1 join emp E2
On E1.MGR = E2.Empno;
```

WAQTD Ename on Manager name emp hiredate and manager Sal. if emp hireded after 1982 and, manager Sal ends with 50?

```
Select E1.Ename, E2.ename, E1.hiredate, E2.Sal.
from empE1
        join emp E2
On E1.mgr = E2.empno
where e1.hiredate > '31-DEC-1982' and e2.Sal
like '%50';
```

Self join Assignment

1.) Select E1.ENAME, M1.EMANE
FROM EMP E1 JOIN EMP M1
ON E1.MGR = M1.EMPNO
where E1.job = 'CLERK';

2.) Select e1.ename, m1.job
from emp e1 join Emp M1
on E1.MGR = M1.EMPNO
where M1.deptno = 10 or M1.deptno = 20;

3.) Select e1.ename, M1.Sal.
from emp e1 join Emp m1
on e1.mgr = M1.Empno
where m1.sal > 2300 and E1.Sal > 2300.

4) select E1.Ename, M1.Hiredate
from emp e1 join emp M1
on E1.mgr = M1.empno
where E1.Hiredate < '01-Jan-1982';

5.) Select e1.ename, M1.comm
from emp e1 join emp m1
on e1.mgr = M1.empno
where e1.job = 'SALESMAN' AND M1.DEPTNO = 30;

6.) Select E1.Ename, M1.Ename, E1.Sal, M1.Sal
from emp e1 join emp M1
on e1.mgr = M1.Empno
where e1.Sal > M1.Sal;

7) Select e1.ename., E1.hiredate, m1.ename, m1.hiredate
from empe1 join emp m1
on e1.mgr = m1.empno
where m1.hiredate > E1.hiredate;

8) Select e1.ename, m1.ename.
from emp e1 join emp m1
on e1.mgr = m1.empno
where e1.job = m1.job;

10.) Select e1.ename, M1.Sal*12, m1.ename, m1.Sal*12
from emp e1 join emp m1
on e1.mgr = m1.empNO
where e1.deptno in (10,20) and m1.sal > e1.sal;

9.) Select e1.ename, m1.ename
from emp e1 join emp m1
on e1.mgr = m1.empno
where m1.job = 'MANAGER';

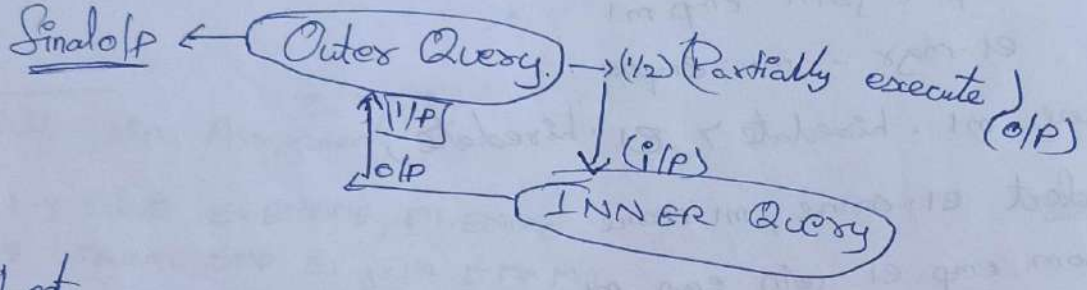
11) Select e1.*, E1.ename, M1.job
from emp e1 join emp m1
on e1.MGR = M1.empno;

12) Select e1.*, e1.ename, m1.sal.
from emp e1 join emp M1
on e1.mgr = m1.Empno
where m1.sal like '%50';

CO-RELATED SUB QUERY

A Query written inside a Query Such that Outer Query and Inner Query are interdependent.

Working Principle



Let us consider two Query that is Outer Query and Inner Query

⇒ Outer Query will Starts the execution. but Partially.

⇒ The Partially executed Outer Query will produce Partial Output for Outer Query

⇒ The Partial Output of Outer Query is given as an i/p to inner Query

⇒ After Taking the i/p inner Query will Start the execution and produces o/p

The o/p of inner Query is Given as an i/p to the outer Query

→ Now Outer Query will start the execution and produces final Output

→ By seeing this we can say the Outer Query and Inner Query are Interdependent (dependent on each other)

→ This is the working Principle of Co-related Sub Query

NOTE

:- Co-related Sub Query works with the principle of Sub Query and joins

:- in Co-related Sub Query join Condition is mandatory that to in Sub Query where clause.

DEPT

DNAME	DNO
D1	10
D2	20
D3	30
D4	40

EMP

DNO	ENAME
20	A
10	B
30	C
20	D

i) D1
ii) D2
iii) D3
iv) NULL

i) 10
ii) 20,20
iii) 30
iv) NULL

```
Select D.Dname.
from Dept D
where D.DNO in (select E.DNO
                from emp E
                where D.DNO = E.DNO);
```

i) 10 in 10 → 10=10 ✓
ii) 20 in (20,20) → 20=20 ✓, 20=20 ✓
iii) 30 in 30 → 30=30 ✓
iv) 40 in NULL → 40=NULL ✗

i) 10=20 ✗
10=10 ✓
10=30 ✗
10=20 ✗

i) 20=20 ✓
20=10 ✗
20=30 ✗
20=20 ✓

iii) 30=20 ✗
30=10 ✗
30=30 ✓
30=20 ✗

iv) 40=20 ✗
40=10 ✗
40=30 ✗
40=20 ✗

EXISTS

It is a Sub Query Operator which returns True Value if the Sub Query is returning Not NULL Values.

eg:- Select Dname
from dept
where exists (Select deptno
NOTNULL from emp
will
accept where dept.deptno = emp.deptno);

NOT EXISTS

It is a Sub Query Operator which returns True Value if the Sub Query is returning NULL Values.

eg:- select Dname
from dept
where not exists (Select deptno
NULL from emp
will
accept where dept.deptno =
emp.deptno);

WAQTD 5 max Sal of the emps.

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
where sal < (Select max(sal) from emp))));

WAQTD 10th max Sal of emp?

```
select max(sal) from emp
where Sal
```

To Find N^{th} maximum Sal by using Co-related Sub Query

Syntax

```
select e1.sal
from emp e1
where (select count(Distinct e2.sal)
       from emp e2
       where e1.sal < e2.sal) = N-1;
```

eg:- 5th max Sal

```
select e1.sal
from emp e1
where (select count(Distinct e2.sal)
from emp e2
where e1.sal < e2.sal) = 4;
```

To find N^{th} minimum Sal By using Co-related Sub Query

Syntax

```
select e1.sal
from emp e1
where (select count(Distinct e2.sal)
       from emp e2
       where e1.sal > e2.sal) = N-1;
```

Single row function

7) SUBSTR()

=> It is used to extract a part of the String from the original String.

Syntax

SUBSTR('ORIGINALSTRING', POSITION, [LENGTH]).

@SPIDERS
1 2 3 4 5 6 7 8

eg:-

i> Substr('@SPIDERS', 4, 3) => IDE
ii> substr('@SPIDERS', 2, 5) => SPIDE
iii> Substr('@SPIDERS', 3) => PIDERS

8) INSTR()

It is used to Retrieve Position Value of a Given String from the Original String.

Syntax

INSTR('ORIGINALSTRING', 'STRING', POSITION, [OCCURANCE])

eg:)

i> INSTR('BANANA', 'N', 1, 2) => 5
ii> INSTR('BANANA', 'A', 3, 2) => 6
iii> INSTR('BANANA', 'B', 1) => 1
iv) INSTR('CHENNAI', N, 3, 2) => 5

Default value for occurance is 1

9.) REPLACE()

It is used to replace a String with the new String.

Syntax

REPLACE('ORIGINALSTRING', 'STRING', ['NEW-STRING'])

eg:
i) REPLACE ('QSPIDER', 'Q', 'J') -> JSPIDER
ii) REPLACE ('DYNGA', 'A', 'I') -> DINGI
iii) REPLACE ('BHAVITHA', 'H') -> BAVITA

(14) MONTH_BETWEEN()

It is used to Retrieve No. of mnths Present between the Given. date

Syntax

MONTHS_BETWEEN ('DATE1', 'DATE2')

eg:-

1) MONTH_BETWEEN ('14-FEB-2023', '14-NOV-2023')

o/p

-9.

2) MONTH_BETWEEN ('14-NOV-2023', '14-FEB-2023')

o/p

9.

(15) ADD_MONTHS

It is used to add no. of months for a Given. daty

Syntax

ADD_MONTHS ('DATE', NUMBER.VALUE)

eg:- 1) SELECT ADD_MONTHS ('SYSDATE', 3)
from dual;

Error => a non-numeric character was found where a numeric was expected

2) SELECT ADD_MONTHS (SYSDATE, 3) from dual;

29-SEP-23

29-DEC-23

3> SELECT ADD_MONTHS ('14-JAN-22', 13)
FROM DUAL;

ADD_MONTH
14-FEB-23

16.) LAST_DAY()

It is used to Retrieve the Last Date of a Given Month.

Syntax

LAST_DAY ('DATE')

eg:- 1> Select Last_Day ('01-JAN-23') from dual;

o/p 31-JAN-23

2> Select Last_Day ('10-FEB-23') from dual;

o/p 28-FEB-23

3> select Last_Day ('09-FEB-24') from dual;

o/p 29-FEB-23 (2024 is Leap year)

TCL

COMMIT => It is used to Save the Transactions into the Database.

Syntax => COMMIT;

ROLLBACK => It is used to Retrieve only Saved Data from the Database.

Syntax => ROLLBACK;

SAVEPOINT => It is used to Save the records temporarily in the table.

Syntax => SAVEPOINT SAVEPOINT_NAME;

Application
Types of Application
PLSQL, TSQL,
DATABASE
DBMS } difference.
RDBMS }
Relational Model
Table
E.F.CODD Rules
Datatype
Types of Datatype and explain it detaily
Diff blw CHAR / VARCHAR / VARCHAR2
Date Datatype and its Syntax
Number Datatype
Constraints
Types of Constraints and explain it detaily
Diff blw Primary key / foreign key

Explain Subset (or) Languages in SQL
DDL } explain each Statements with Syntax
DML "
TCL "
DCL "
DQL "

what is Operants and it types explain it
explain Expression, ALIAS, Distinct
Explain Where clause?
Explain NULL
Types of Operators
Explain Set operations and its Statements
Explain Logical Operators and its Statements
explain ORDER BY Clause
Explain Special Operators and its Statements

Functions and its classifications
explain the types of user defined function.
& explain the types of built-in function.
explain SRF and MRF
Types of MRF and explain its 5 functions
Types of SRF and explain its functions (16)
Explain Group by Clause and its characteristics
Explain Having Clause and its characteristics
Diff B/w where clause and Having clause.
Syntax for all clauses like Select, from, where, Group by, Having and order by clausses. and its order of execution.
Explain SubQuery Operators
Diff b/w filter Condition and Group filter Condition.
Case-1 of SubQuery (See Sam examples)
Case-2 of SubQuery "
Combination of both Cases
Types of SubQuery. Explain it
is! SubQuery is different than SubQuery Operators?
Employee - manager Relation (Case I & Case II)
SubQuery Explain in a flow.
what is Joins and it types
Explain Each type of joins with Syntax and it uses.
What is CO-Related SubQuery and its working Principle.
write a Syntax for finding Nth max and min Sal.?

Expected Interview Question

1) Explain Database?
2) Diff b/w DBMS and RDBMS?
3) Diff b/w Database and DBMS?
4) Explain the Rules of E.F CODD?
5) Define Datatypes and explain the types of datatypes?
6) Explain Constraint with its types?
7) what is Projecton?
8) Explain where clause?
9) what is Set Operators?
10) Diff b/w union and union all?
11) Explain Group by clause?
12) what is Group Function (MRF)? } difference by
13) Explain the types of Aggregate Function with it Rules (MRF)?
14) Diff b/w WHERE and HAVING CLAUSE?
15) Diff b/w FILTER CONDITION and GROUP FILTER Condition.
16) Diff b/w char / Varchar / Varchar2?
17) what are the Sub Sets of SQL?
what are the Languages in SQL?
18) what is Function and list the types of Function?
19) Diff b/w SRF and MRF?