INTRODUCTION

24 June 2021 06:24 PM

What is an **Application?**

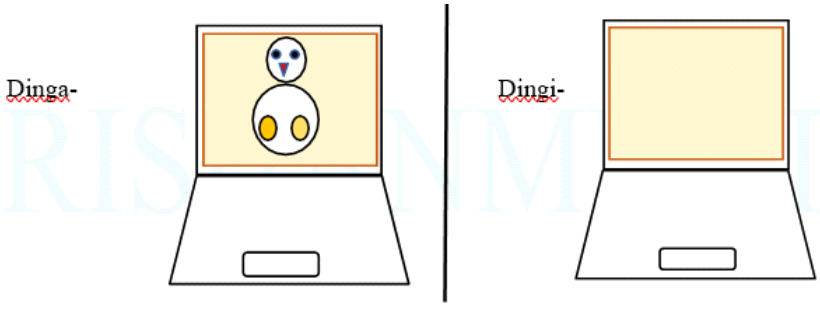
" Is a list of programs which helps us to perform some a particular task/action "

Example: WhatsApp
MS office
Web browser [CHROME, MOZILLA] Facebook
Instagram
GTA Vice city

Types of Applications:

1. Stand Alone Application.
2. Web Application.
3. Client / Server Application (Mobile Application).

1. STAND ALONE APPLICATION:



Software installed in one computer and used by **only one** person.
Examples :– Installing s/w of a Calculator, Adobe Photoshop, MS Office, AutoCAD, Paint etc.…

Advantages:

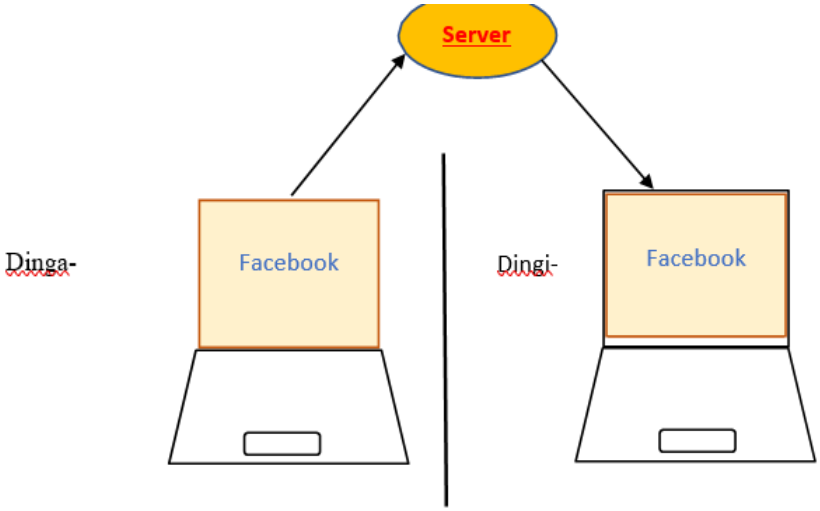
- Faster in access.
- Secured from Data hacking and virus.

Disadvantages:

- Single user access at a time.

- Installation is required

2.WEB APPLICATION:

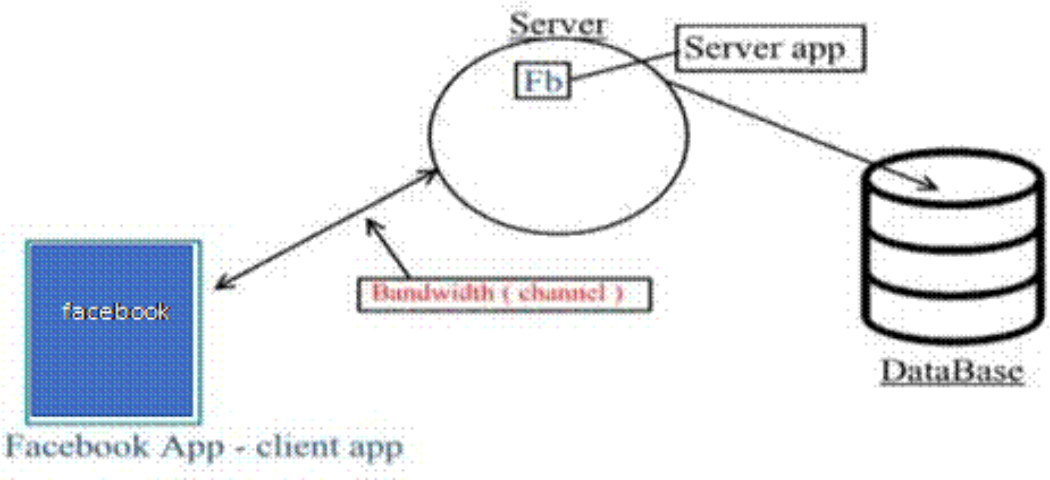


Any Application which is opened through a browser is known as WEB APPLICATION.

Examples: - Facebook , wiki , YouTube , Gmail , amazon , SkillRary etc. ...

Server: It is nothing but a *super computer* where in all the applications are installed and can be accessed by anyone. [High Configuration]

3.CLIENT / SERVER APPLICATION:



In Client Server application, unlike Standalone Application, part of application is installed on to the client system and the remaining part is installed on to the server machine.

Examples: - Facebook, WhatsApp, Instagram, YouTube, Wiki, OLX, Flipkart etc.….

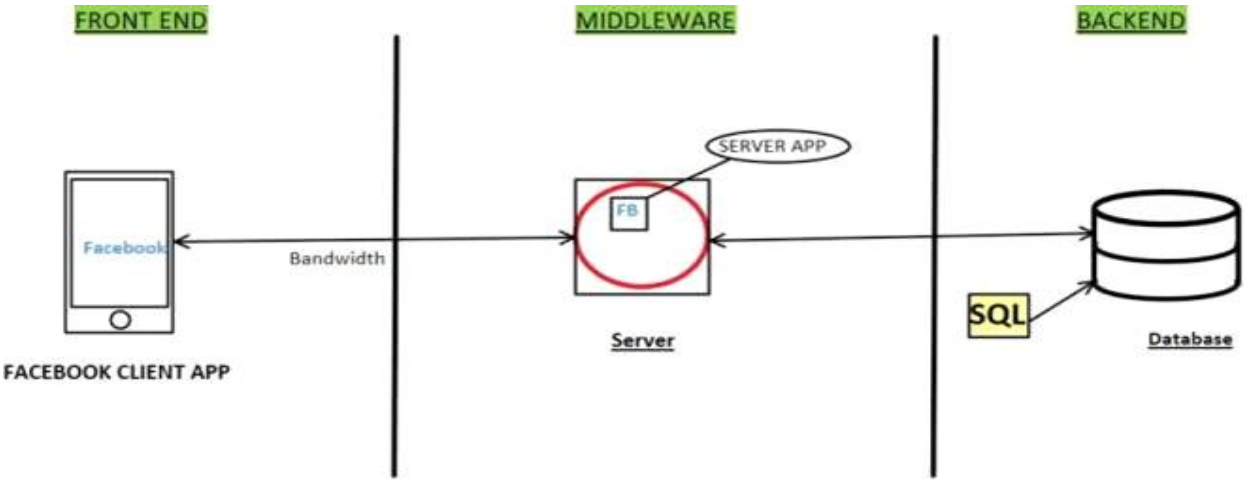
Advantages:

- Easy to access and faster in access if the bandwidth is more
- Data security from data hacking and virus
- Data sharing is possible
- Maintenance is not so tough
- Multiple users can access the application.

Disadvantages:

- Installation is still required at client's place
- If the server goes down no one can access the application

EXAMPLE:



PRESENTATION LAYER BUSINESS LAYER DATA LAYER

INTRIDUCTION OF SQL

22 July 2021 08:02 PM

SQL is standard computer language

SQL stands for ***STRUCTURED QUERY LANGUAGE***

SQL is used to communicate / interact and manipulate the DATABASE

WHAT CAN SQL DO……?

SQL can execute queries against a database
SQL can retrieve data from a database
SQL can insert records in a database
SQL can update records in a database
SQL can delete records from a database
SQL can create new tables in a database
SQL can create views in a database

HISTORY OF SQL

1) **SQL WAS DESIGNED BY 'RAYMOND BOYCE' IN THE YEAR 1970.**
2) **FATHER OF SQL IS KNOWN AS RAYMOND BOYCE.**
3) **E.F.CODD DESGINED RELATIONAL MODEL IN THE YEAR OF 1970 (SYSTEM.R).**
4) **E.F CODD IS CALLED BY CO-FATHER OF SQL.**
5) **IN OLDEN DAYS SQL IS CALLED BY SEQUEL .**
   S→SIMPLE
   E→ENGLISH
   QUE→QUERY
   L→ LANGUAGE
6) **ONCE ANSI(AMERICAN NATIONAL STANDARD INSTITUTE) TOOK OVER AND THEY WAS CHAGNED THE NAME AS SQL.**

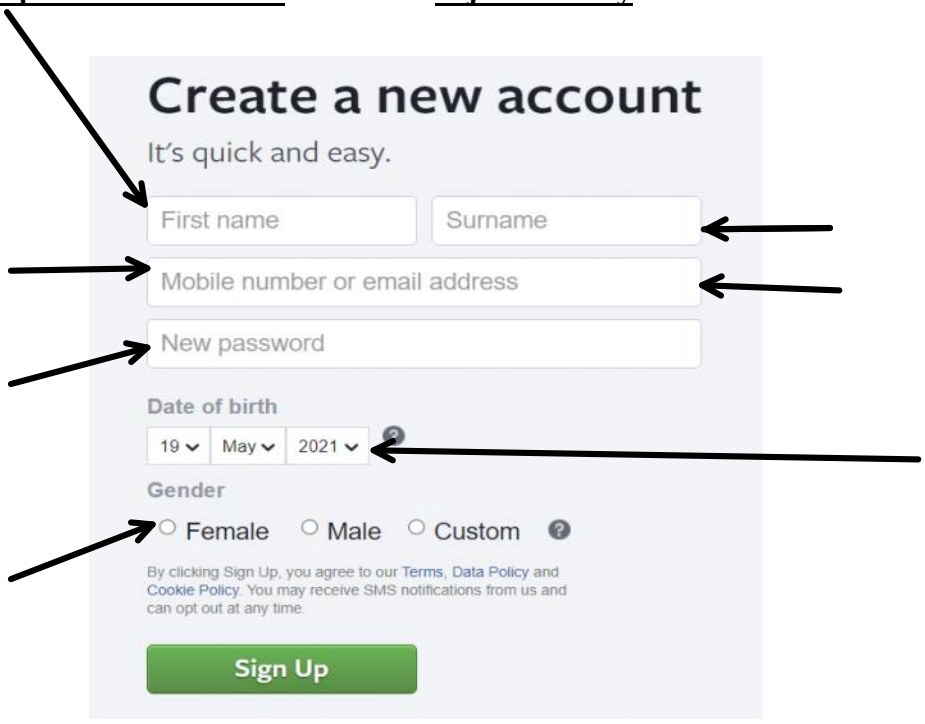
DAY1

15 May 2021 03:19 PM

DATA:

" Data is a Raw fact which describe the property of an object "

Properties or Attributes **object or entity**



I am creating my account for myself

Attributes

First Name: Krishna
Sur Name: Gowda
PH_no: 9874561230
Password: xyz@123
DOB: 12/4/19xx
Gender: Male

OBJECT

LAPTOP	LAPTOP
Brand: Dell	**Brand: Dell**
RAM: 8gb	**RAM: 8gb**
Touch: no	**Touch: yes**

Example: **WATER BOTTLE**

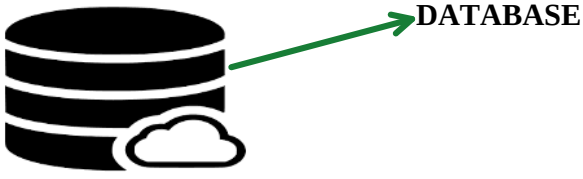
HIEGHT: 10CM

COLOUR: BULE

CAPACITY: 500ML

DATA BASE

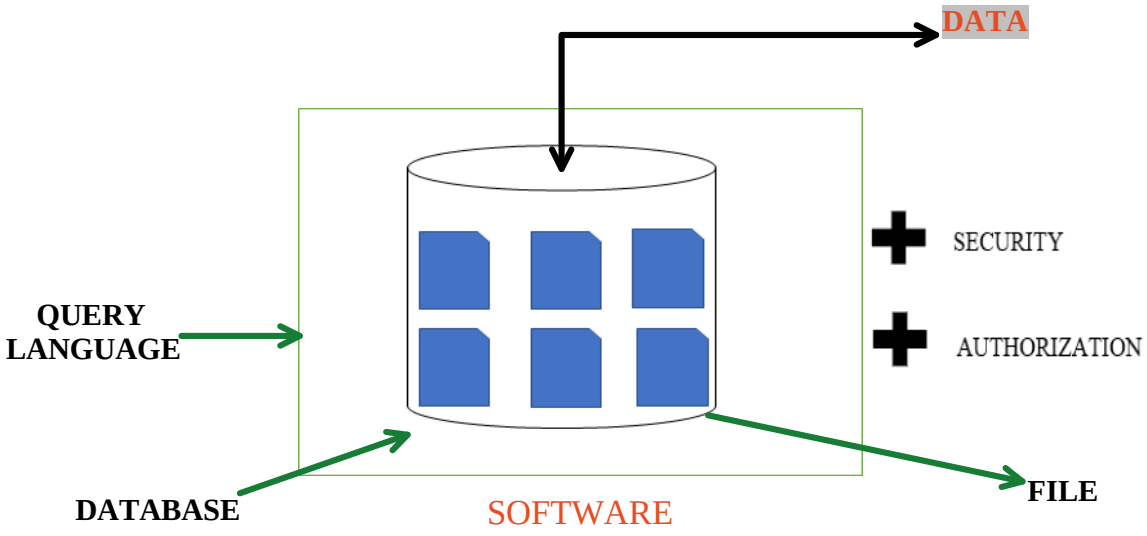
"Data base is place or media which we can store the data in systematic and* organized *manner"



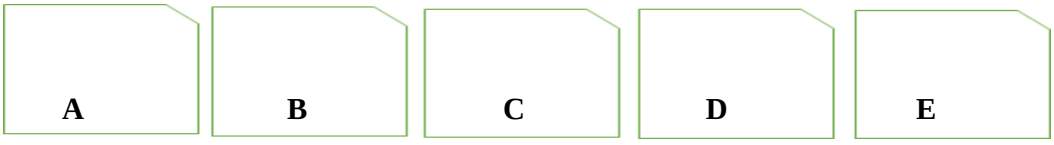
- ***The basic operations that can be performed on the database are***
  - **CREATE / INSERT**
  - **READ / RETRIVE**
  - **UPADTE / MODIFY**
  - **DROP / DELETE**

- ***"This operation are universally known as "CRUD" operation"***

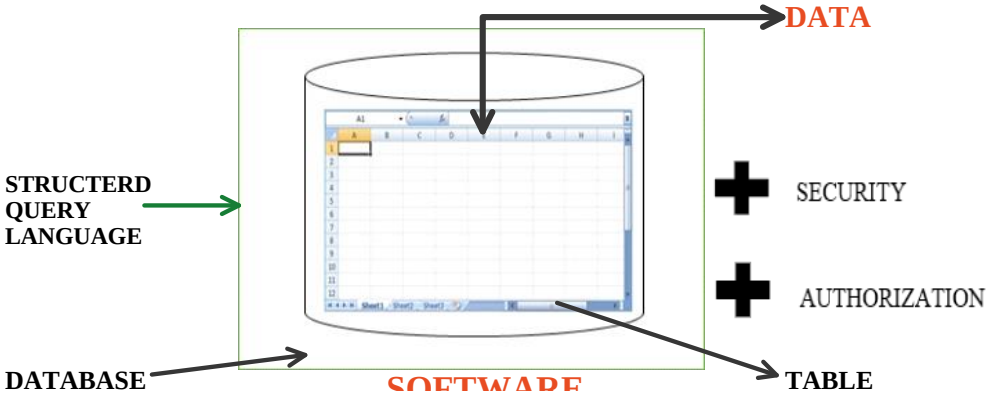
Data base Management system (DBMS)



- **DBMS is a software which is used to *maintain and manage* the database**
- ***Security and Authorization* are the two imp Key feature provided by the DBMS.**
- **We use *Query Language* to communicate or interact the DBMS**
- **DBMS Stores the data in the form of *Files***



RELETIONAL DATE BASE MANGEMENT SYSTEM (RDBMS)



- ***RDBMS*** **a type of the** ***DBMS*** **which is used to store the data in the Form of table**
- **We can use to** ***communicate or interact*** **RDBMS by using** ***SQL (Structured Query Language)***

EXAMPLE:

NAME
A
B
C
D
E

ASSINMENT:

1)DIFFERENCE BETWEEN THE DBMS AND RDBMS

krish_sql_techie

DAY2

15 May 2021 03:19 PM

RELATIONAL MODEL

- Relational Model was designed by *E.F. CODD*
- In Relational model we store the Data in the form of *Rows and Columns*

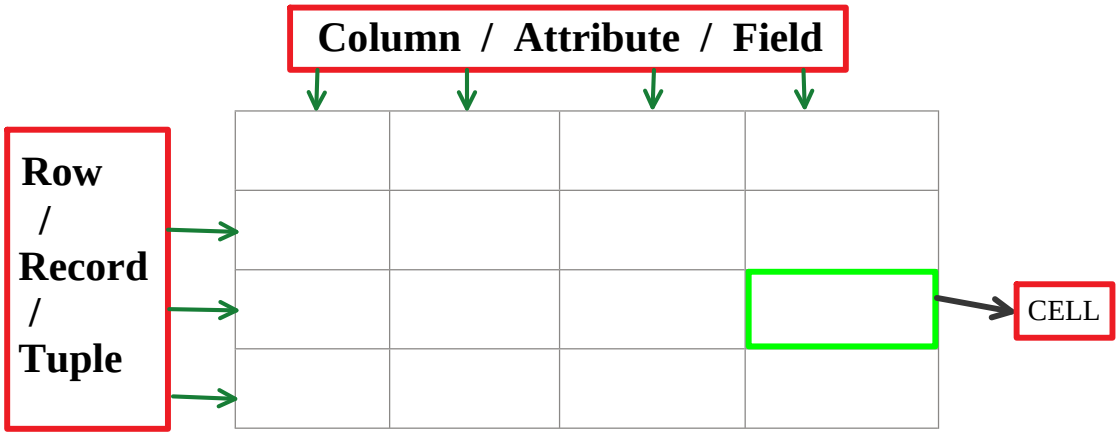
Any DBMS which follows Relational model becomes on RDBMS

OR

Any DBMS which follows E.F.CODD becomes on RDBMS

Table:

Table is logical organization of Data which consists of Rows and columns



Columns:

- **A columns is also known as *attributes OR field.***
- ***A column is used to represent property of all the entity.***

Row:

- **Row is also known as *Record or Tuples.***
- ***Row is used to represent all properties of a single entities.***

Cell:

- **Cell is the smallest unit of the table in which we store the data.**
- **The *interaction of rows and columns generate cells*.**

Example :

Emp (Entity)
- Eid
- Ename
- Salary

EMP :

EID	ENAME	SALARY
1	SMITH	1000
2	ALLEN	1500
3	CLARK	2000

RULES OF E.F.CODD

- **The data entered into a cell must be *"single valued data"* (atomic)**

  Example :

EID	ENAME	PHONE_NO
1	SMITH	101
2	ALLEN	102 , 202
3	CLARK	103

EID	ENAME	PHONE_NO	ALTERNATE NO
1	SMITH	101	null
2	ALLEN	102	202
3	CLARK	103	

- **According to E.F. CODD we can store the data in multiple table, It necessary are can established the connection between the table with the help of *"key attribute"***
- **In RDBMS are store everything in the form of table including *"META DATA"***

Example : *Metadata* : The details about a data is knows as Metadata.

EID	ENAME	PHOTO
1	SMITH	
2	ALLEN	
3	CLARK	

DATA

PHOTO

DATA

Image Name : Mypic
size : 127kb
resolution : 400 x 600
format : jpeg

MetaTable

Image name	size	Format	Resolution
Mypic	127	jpeg	400 x 600

➢ **Data enter into the table *must be validated***

classified as two way

→ **By assigning *"DATATYPE"***
→ ***By assigning "CONSTRAINTS"***

Datatype are mandatory whereas constrains are optional

DATA TYPE:

"DATATYPE is used to specify/ determine the type /kind of data that will be stored in a particular memory allocation."

DATA TYPES In SQL

1. **Char**
2. **Varchar/ Varchr2**
3. **Number**
4. **Date**
5. **Large object**

⇒ **Character large object (clob)**
⇒ **Binary large object (blob)**

Note: SQL is Not case sensitive Language

1) CHAR:

"In CHAR data type we can store characters such as"

Example: 'A-Z' , 'a-z' , '0-9' and Special character ('$', '#', '%', '&')

➢ **Whenever we used char data type we must specify the size**

Syntax: CHAR (Size)

Size: Numbers of characters

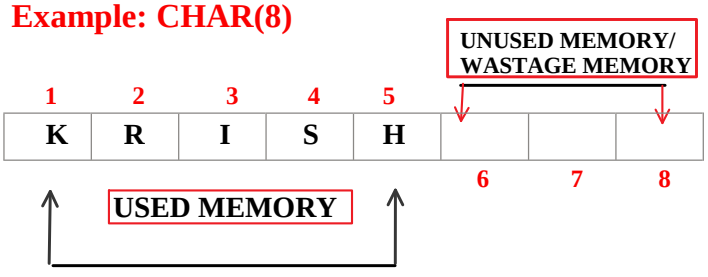
Size:

→ **It is used to specify number of characters that can be stored**
→ **we can store a max of *2000 characters***
→ **character must always be with in *single quote (' ')***

Ex: 'ABC' ,'Raju' , 'A'

➢ **Character datatype follows "*Fixed length memory allocation*"**
➢ ***Default size of char is one***

Example: CHAR(8)



2) VARCHAR/ VARCHAR2:

"In VARCHAR data type we can store characters such as"

Example: 'A-Z' , 'a-z' , '0-9' and Special character ('$', '#', '%', '&')

- **Whenever we used varchar data type we must specify the size**

Syntax: VARCHAR (Size)

Size: Numbers of characters

Size:

→ **It is used to specify number of characters that can be stored**

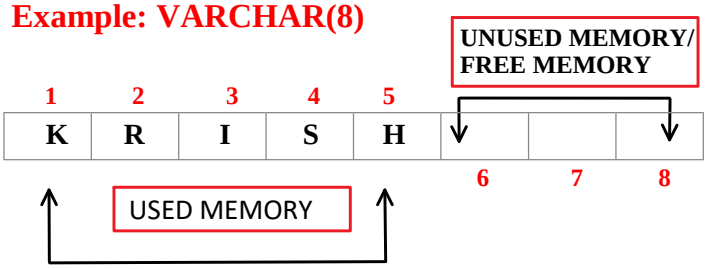
→ **we can store a max of *2000 characters***

→ **character must always be with in *single quote (' ')***

Ex: 'ABC' ,'Raju' , 'A'

- **VARChar datatype follows "*variable length memory allocation*" , *default size of varchar is zero***

Example: VARCHAR(8)



NOTE:

VARCHAR2 is nothing but update version of varchar

MAX size is ***4000*** character

ASSIGNMENT

1)DIFFRENACE BETWEEN CHAR AND VARCAHR/varchar2

DAY 3

28 June 2021 04:10 PM

3.NUMBER

' ***This number datatype is used to store numeric value'***

Number datatype can accept 2 arguments such as

- ***Precision***
- ***Scale***

SYNTAX

OPTIONAL

NUMBER(Precision, [scale])

Precision

- Precision is used to determine the number of digit used to store '***Integer value'***
- The range of the precision is ***1 to 38***

Scale

- Scale is used to determine the number of digit used to store ***Decimal*** (float) value within the precision
- The default value of scale is ***'zero'***
- The range of scale is ***-84 to 127***

Examples :

1	NUMBER(3)	+/-999
2	NUMBER(4)	+/-9999
3	NUMBER(5,2)	+/-999.99

PRECISION SCALE

5	NUMBER(7,3)	+/- 9999.999
6	NUMBER(3,6)	+/- 0.000999

7	NUMBER(2,7)	+/- 0.0000099
8	NUMBER(5,8)	+/-0.00099999
9	NUMBER(3,3)	+/- 0.999

4.DATE

The date datatype is used to store the date in specific format given by oracle

SYNTAX: ***DATE***

Format:

1) 'DD-MON-YY' '28-JUN-21'

OR

2) 'DD-MON-YYYY' '28-JUN-2021'

5.LARGE OBJECT

❖ CHARACTER LARGE OBJECT : (CLOB)

Character large object (CLOB) is used to store huge amount of the character up-to ***4GB*** of size

SYNTAX

{CLOB | CHARACTER LARGE OBJECT } [(LENGTH [{K|M|G}])]

❖ BINARY LARGE OBJECT : (BLOB)

Binary large object (BLOB) is used to store the binary values of ***"IMAGES, MP3, MP4, PDF"*** etc. up-to ***4GB*** of size

SYNTAX

{BLOB | BINARY LARGE OBJECT } [(LENGTH [{K|M|G}])]

DAY 4

15 May 2021 03:19 PM

CONSTRAINTS :

It is a rule given to a column for validation .

Types of Constraints :

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

1. **UNIQUE** : "*It is used to avoid duplicate values into the column* ".

2. **NOT NULL** : *"It is used to avoid Null ".*

3. **CHECK** : "*It is an extra validation with a condition If the condition is satisfied then the value is accepted else Rejected* ".

   EXAMPLE: CHECK(LENGTH(PH_NO))=10
   CHECK (PERCENTAGE>60)

4. **PRIMARY KEY** : "*It is a constraint which is used to identify a record Uniquely from the table*" .

Characteristics of Primary key :

- We can have only 1 PK in a table
- PK cannot accept duplicate / repeated values .
- PK cannot accept Null
- PK is always a combination of Unique and Not Null Constraint.
- PK is not mandatory but highly recommended.

5. **FOREIGN KEY** : "*It is used to establish a connection between the The tables* "

Characteristics of Foreign key :

- We can have Multiple FK in a table
- FK can accept duplicate / repeated values .
- FK can accept Null
- FK is not a combination of Unique and Not Null Constraint.
- For an Attribute (column) to become a FK ,it is mandatory that it must be a PK in its own table .
- FK are present in child table but it actually belongs to parent table
- FK is also known as "***REFRENTIAL INTEGRITY CONSTRAINT***"

Example :

EMP

PK				
		CHECK(SALARY> 0)		*CHECK(LENTH(PHONE)= 10)*
NOT NULL	*NOT NULL*	*NOT NULL*	*NOT NULL*	*NOT NULL*
UNIQUE				*UNIQUE*
EID	*NAME*	*SALARY*	*DOJ*	*PHONE*
NUMBER(4)	*VARCHAR(10)*	*NUMBER(6,2)*	*DATE*	*NUMBER(10)*
123	*RAJU*	*5000.00*	*'26-JUN-1998'*	*1234567890*
321	*PRADEEP*	*6000.00*	*'28-JAN-2001'*	*3214567890*

Example for Foreign Key :

EMP

EID	NAME	SALARY	Cid(fk)	Dno(fk)
1	A	10000	1	10
2	B	20000	1	20
3	C	35000		20
4	D	50000	2	10

Child / accepter

CUSTOMER

CID	CNAME	CNO
1	X	1001
2	Y	2002
3	Z	3003

Parent/donar

DEPT

DNO	DNAME	LOC
10	D1	L1
20	D2	L2

ASSIGNMENT :

1. Differentiate between Primary key and Foreign key .

PRIMARY KEY	FOREIGN KEY
It is used to identify a records Uniquely from the table.	It is used to establish a connection Between the tables
It cannot accept Null	It can accept Null
It cannot accept duplicate values	It can accept duplicate values

It is always a combination of Not Null and Unique constraint	It is not a combination of Not Null and Unique constraint
We can have only 1 PK in a table	We can have Multiple FK in a table

NOTE : NULL

Null Is a keyword which is used to represent Nothing / Empty Cell.

Characteristics of Null :

- Null doesn't represent 0 or Space .
- Any operations performed on a Null will result in Null itself
- Null doesn't Occupy any Memory .
- We cannot Equate Null .

Ex:

emp

eid	ename	sal	comm
1	a	100	30
2	b	200	

200*null= null
200+null= null
200-null= null
200+ null= null

200=null (not possible)

DAY 5

15 May 2021 03:19 PM

STATEMENTS

OVERVIEW OF SQL *STATEMENTS* :

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

DATA QUERY LANGUAGE (DQL) :

" ***DQL is used to retrieve/Read/fetch the data from the database*** " .

It has 4 statements :

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

SELECT :"It is used to retrieve the *data* from the table and display it.

PROJECTION :

"It is a process of retrieving the data by *selecting only the columns* is known as Projection " .

- In projection all the records / values present in a particular columns are by default selected .

SELECTION :

"It is a process of retrieving the data by *selecting both the columns and rows* is known as Selection " .

JOIN :

"It is a process of retrieving the data from *Multiple tables* simultaneously is known as Join " .

PROJECTION

- "It is a process of retrieving the data by *selecting only the columns* is known as Projection " .
- In projection all the records / values present in a particular columns are by default selected .

SYNTAX :

```
SELECT * / [DISTINCT] Column_Name / Expression [ALIAS]
FROM Table_Name ;
```

SELECT * / [DISTINCT] Column_Name / Expression [ALIAS]
FROM Table_Name ;

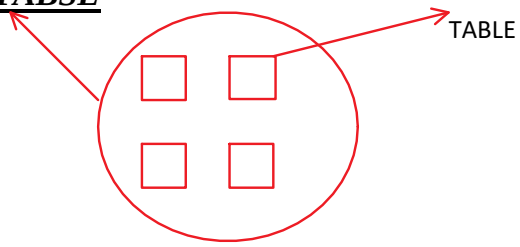
ORDER OF EXECUTION

1. FROM Clause
2. SELECT Clause

Example : .Write a query to display names of all the students

```
SELECT SNAME
FROM  STUDENT;
```

DATABSE



OUTPUT OF FROM CLASUE

STUDENT

SID	SNAME	BRANCH	PER
1	A	ECE	60
2	B	CSE	75
3	C	ME	50
4	D	ECE	80
5	C	CSE	75
6	E	CIVIL	95

OUTPUT OF SELECT CLAUSE

SNAME
A
B
C
D
E
F

FROM CLAUSE:

- FROM Clause starts the execution .
- For FROM Clause we can pass Table Name as an argument .
- The job of FROM Clause is to go to the Database and search for the table and put the table under execution .

SELECT CLAUSE

- SELECT Clause will execute after the execution of FROM Clause
- For SELECT Clause we pass 3 arguments
  i. Asterisk(*)
  ii. Column_Name
  iii. Expression
- The job of SELECT Clause is to go the table under execution and select the columns mentioned .
- SELECT Clause is responsible for preparing the result table .

NOTE

- Asterisk(*) : it means to select all the columns from the table .
- Semicolon : it means end of the query .

Example

- WAQTD student id and student names for all the students.
- WAQTD name and branch of all the students .
- WAQTD NAME , BRANCH AND PERCENTAGE FOR ALL THE STUDENTS .
- WAQTD details of all the students from students table .
- WAQTD sname , sid , per , branch of all the students .

EMP Table :

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
7369	SMITH	CLERK	17-DEC-80	7902	800		20
7499	ALLEN	SALESMAN	20-FEB-81	7698	1600	300	30
7521	WARD	SALESMAN	22-FEB-81	7698	1250	500	30
7566	JONES	MANAGER	02-APR-81	7839	2975		20
7654	MARTIN	SALESMAN	28-SEP-81	7698	1250	1400	30
7698	BLAKE	MANAGER	01-MAY-81	7839	2850		30
7782	CLARK	MANAGER	09-JUN-81	7839	2450		10
7788	SCOTT	ANALYST	19-APR-87	7566	3000		20
7839	KING	PRESIDENT	17-NOV-81		5000		10
7844	TURNER	SALESMAN	08-SEP-81	7698	1500	0	30
7876	ADAMS	CLERK	23-MAY-87	7788	1100		20
7900	JAMES	CLERK	03-DEC-81	7698	950		30
7902	FORD	ANALYST	03-DEC-81	7566	3000		20
7934	MILLER	CLERK	23-JAN-82	7782	1300		10

EXAMPLE OF EMP TABLE

- WAQTD name salary and commission given to all the employees .
- WAQTD name of the employee along with their date of joining .

DEPT :

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

- WAQTD dname and location for all the depts .

WAQTD ALL THE DETAILS OF DEPT TABEL

QUESTIONS ON EMP AND DEPT TABLE:

1. WRITE A QUERY TO DISPLAY ALL THE DETAILS FROM THE EMPLOYEE TABLE.
2. WAQTD NAMES OF ALL THE EMPLOYEES.
3. WAQTD NAME AND SALARY GIVEN TO ALL THE EMPLOYEES.
4. WAQTD NAME AND COMMISSION GIVEN TO ALL THE EMPLOYEES.
5. WAQTD EMPLOYEE ID AND DEPARTMENT NUMBER OF ALL THE EMPLOYEES IN EMP TABLE.
6. WAQTD ENAME AND HIREDATE OF ALL THE EMPLOYEES .
7. WAQTD NAME AND DESIGNATION OF ALL THE EMPLPOYEES .
8. WAQTD NAME , JOB AND SALARY GIVEN ALL THE EMPLOYEES.
9. WAQTD DNAMES PRESENT IN DEPARTMENT TABLE.
10. WAQTD DNAME AND LOCATION PRESENT IN DEPT TABLE.

DISTINCT Clause

" It is used to remove the duplicate or repeated values from the Result table ".

- Distinct clause has to be used as the first argument to select clause.
- Distinct clause must be used before column name/ expression
- We can use multiple columns as an argument to distinct clause, it will remove the combination of columns in which the records are duplicated

Example :

Student

SID	SNAME	BRANCH	PER
1	A	ECE	60
2	B	CSE	75
3	C	ME	50
4	D	ECE	80
5	C	CSE	50
6	E	CIVIL	95

SELECT DISTNICT SNAME,PER,BRANCH FROM STUDENT;

SNAME	PER	BRANCH
A	60	ECE
B	75	CSE
C	50	ME
D	80	ECE
C	50	CSE
E	95	CIV

SNAME	PER	BRANCH
A	60	ECE
B	75	CSE
C	50	ME
C	50	CSE
D	80	ECE
E	95	CIV

DAY 6

15 May 2021 03:19 PM

EXPRESSION

"Any statement which gives result is known as Expression ". Expression is a combination Operand and Operator .

Operand : These are the values that we pass .

Operator : These are the Symbols which perform some Operation on The Operand .

Example : 5 * 10

OPERAND

OPERATOR

EMP

EID	ENAME	SAL
1	A	100
2	B	200
3	C	100

- WAQTD name and salary given to the employees .
- WAQTD name and annual salary of the employees .
- WAQTD all the details of the employee along with annual salary
- WAQTD name and salary with a hike of 20%.
- WAQTD name and salary of an employee with a deduction Of 10% .

ALIAS

"It is an alternate name given to a Column or an Expression In the result table " .

- We can assign alias name with or without using 'As' keyword .
- Alias names have to be a single string which is separated by An underscore or enclosed within double quotes .

FORMAT :	ANNUAL_SALARY
	"ANNUAL SALARY"

- WAQTD name and salary with a deduction 32% .

```
SELECT ENAME,SAL,SAL-SAL*32/100 DEDUCTION
FROM EMP;
```

- WAQTD annual salary for all the employees .

  SELECT SAL*12

  FROM EMP;

ASSIGNMENT ON EXPRESSION & ALIAS

1) WAQTD NAME OF THE EMPLOYEE ALONG WITH THEIR ANNUAL SALARY.
2) WAQTD ENAME AND JOB FOR ALL THE EMPLOYEE WITH THEIR HALF TERM SALARY.
3) WAQTD ALL THE DETAILS OF THE EMPLOYEES ALONG WITH AN ANNUALBONUS OF 2000.
4) WAQTD NAME SALARY AND SALARY WITH A HIKE OF 10%.
5) WAQTD NAME AND SALARY WITH DEDUCTION OF 25%.
6) WAQTD NAME AND SALARY WITH MONTHLY HIKE OF 50%.
7) WAQTD NAME AND ANNUAL SALARY WITH DEDUCTION OF 10%.
8) WAQTD TOTAL SALARY GIVEN TO EACH EMPLOYEE (SAL+COMM).
9) WAQTD DETAILS OF ALL THE EMPLOYEES ALONG WITH ANNUAL SALARY.
10) WAQTD NAME AND DESIGNATION ALONG WITH 100 PENALTY IN SALARY.

COMMANDS ON SQL*Plus :

1. CLEAR SCREEN [**CL SCR**] : *To clear the screen*
2. SET LINES 100 PAGES 100 : *To set the dimensions of the output page* .
3. EXIT / QUIT : *To Close the Software* .
4. When account is Locked !!!
   Log in as SYSTEM
   Password Oracle1234

   ALTER USER SCOTT ACCOUNT UNLOCK ;
   ALTER USER SCOTT IDENTIFIED BY TIGER ;

5. SELECT * FROM TAB ;

- **EMP**
- **DEPT**
- SALGRADE
- BONUS

DESC TABEL_NAME------ DESCRIPTION THE TABLES

DAY 7

15 May 2021 03:19 PM

SELECTION:

"It is a process of retrieving the data by *selecting both the columns and rows* is known as Selection "

SYNTAX :

```
SELECT * / [DISTINCT] Column_Name / Expression [ALIAS]
FROM Table_Name
WHERE <Filter_Condition> ;
```

ORDER OF EXECUTION

1. FROM
2. WHERE
3. SELECT

WHERE Clause

"Where clause is used to filter the records ".

- Where clause execute row by row
- Where clause execute after the execution of from clause.
- In Where clause we can write the filter_condition.
- The return type of condition in the form of Boolean (true or false).
- We can write multiple condition in where clause with the help of logical operator.

EXAMPLE:

- WAQTD names of the employees working in dept 20 .

1. WAQTD names of the employees getting salary More than 300 .
2. WAQTD names and salary of the employees working in dept 10.
3. WAQTD all the details of the employees whose salary is Less than 1000 rupees .
4. WAQTD name and hiredate of an employee hired on '09-JUN-1981'

5. WAQTD details of the employee whose name is 'Miller'

6. WAQTD details of the employee hired after '01-JAN-1982'

7. WAQTD name sal and hiredate of the employees who were Hired before 1985 .

```
SELECT  ENAME,SAL,HIREDATE
FROM EMP
WHERE HIREDATE < '01-JAN-1985';
```

8. WAQTD name sal and hiredate of the employees who were Hired after 1985 .

9. WAQTD name of the employees who was hired on Valentine's day 2021 .

ASSIGNMENT ON WHERE Clause .

1.WAQTD THE ANNUAL SALARY OF THE EMPLOYEE WHOS NAME IS SMITH

2.WAQTD NAME OF THE EMPLOYEES WORKING AS CLERK

3.WAQTD SALARY OF THE EMPLOYEES WHO ARE WORKING AS SALESMAN

4.WAQTD DETAILS OF THE EMP WHO EARNS MORE THAN 2000

5.WAQTD DETAILS OF THE EMP WHOS NAME IS JONES

6.WAQTD DETAILS OF THE EMP WHO WAS HIRED AFTER 01-JAN-81

7.WAQTD NAME AND SAL ALONG WITH HIS ANNUAL SALARY IF THE ANNUAL SALARY IS MORE THAN 12000

8.WAQTD EMPNO OF THE EMPLOYEES WHO ARE WORKING IN DEPT 30

9.WAQTD ENAME AND HIREDATE IF THEY ARE HIRED BEFORE 1981

10.WAQTD DETAILS OF THE EMPLOYEES WORKING AS MANAGER

11.WAQTD NAME AND SALARY GIVEN TO AN EMPLOYEE IF EMPLOYEE EARNS A COMMISSION OF RUPEES 1400

12.WAQTD DETAILS OF EMPLOYEES HAVING COMMISSION MORE THAN SALARY

13.WAQTD EMPNO OF EMPLOYEES HIRED BEFORE THE YEAR 87

14.WAQTD DETAILS OF EMPLOYEES WORKING AS AN ANALYST

15.WAQTD DETAILS OF EMPS EARNING MORE THAN 2000 RUPEES PER MONTH

DAY 8

15 May 2021 03:19 PM

OPERATORS IN SQL

1) ARITHEMATIC OPERATORS :- (+ , - , * , /)
2) CONCATENATION OPERATOR :- (||)
3) COMPARISION OPERATORS :- (= , != or <>)
4) RELATIONAL OPERATOR :- (> , < , >= , <=)
5) LOGICAL OP : (AND , OR , NOT)
6) SPECIAL OPERATOR :-
    1. IN
    2. NOT IN
    3. BETWEEN
    4. NOT BETWEEN
    5. IS
    6. IS NOT
    7. LIKE
    8. NOT LIKE
7) SUBQUERY OPERATORS:-
    1. ALL
    2. ANY
    3. EXISTS
    4. NOT EXISTS

CONCATENATION Operator :

" It is used to join the strings ".

Symbol : ||

SYNTAX

```
'STRING1' || 'STRING 2'
```

NOTE :

We can join 'N' number of strings by using a concatenation operator.

Example:

WAQTD ename the emp working as manager

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

ENAME
ALLEN

JONES

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

ENAME
HI ALLEN
HI JONES

LOGICAL OPERATORS

- AND
- OR
- NOT

We use logical operators to write multiple conditions .

- WAQTD name and deptno along with job for the employee working in dept 10 .
- WAQTD name and deptno along with job for the employee working as manager in dept 10 .
- WAQTD name , deptno , salary of the employee working in dep 20 and earning less than 3000 .
- WAQTD name and salary of the employee if emp earns More than 1250 but less than 3000 .
- WAQTD name and deptno of the employees if the EMP works in dept 10 or 20 .
- WAQTD name and sal and deptno of the employees If emp gets more than 1250 but less than 4000 and works in dept 20 .
- WAQTD name , job , deptno of the employees working as a manager in dept 10 or 30 .
- WAQTD name , deptno , job of the employees working in dept 10 or 20 or 30 as a clerk .
- WAQTD name , job and deptno of the employees working as clerk or manager in dept 10 .
- WAQTD name , job , deptno , sal of the employees working as clerk or salesman in dept 10 or 30 and earning more than 1800 .
- WAQTD ENAME AND DEPTNO OF THE EMPLOYEES BY EXCLUDING THE EMPLOYEES OF DEPT 10 ,61 20 .

ASSIGNMENT ON LOGICAL OPERATORS :

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

THAN 1500

2. WAQTD NAME AND HIREDATE OF THE EMPLOYEES WORKING AS MANAGER IN DEPT 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.
4. WAQTD ALL THE DETAILS OF THE EMP WORKING IN DEPT 30 OR AS ANALYST
5. WAQTD NAMES OF THE EMPMLOYEES WHOS
6. SALARY IS LESS THAN 1100 AND THEIR DESIGNATION IS CLERK
7. WAQTD NAME AND SAL , ANNUAL SAL AND DEPTNO IF DEPTNO IS 20 EARNING MORE THAN 1100 AND ANNUAL SALARY EXCEEDS 12000
8. WAQTD EMPNO AND NAMES OF THE EMPLOYEES WORKING AS MANAGER IN DEPT 20
9. WAQTD DETAILS OF EMPLOYEES WORKING IN DEPT 20 OR 30 .
10. WAQTD DETAILS OF EMPLOYEES WORKING AS ANALYST IN DEPT 10 .
11. WAQTD DETAILS OF EMPLOYEE WORKING AS PRESIDENT WITH SALARY OF RUPEES 4000 11.WAQTD NAMES AND DEPTNO , JOB OF EMPS WORKING AS CLERK IN DEPT 10 OR 20
12. WAQTD DETAILS OF EMPLOYEES WORKING AS CLERK OR MANAGER IN DEPT 10 .
13. WAQTD NAMES OF EMPLOYEES WORKING IN DEPT 10 , 20 , 30 , 40 .
14. WAQTD DETAILS OF EMPLOYEES WITH EMPNO
15. 7902,7839.
16. WAQTD DETAILS OF EMPLOYEES WORKING AS MANAGER OR SALESMAN OR CLERK
17. WAQTD NAMES OF EMPLOYEES HIRED AFTER 81 AND BEFORE 87
18. WAQTD DETAILS OF EMPLOYEES EARNING MORE THAN 1250 BUT LESS THAN 3000
19. WAQTD NAMES OF EMPLOYEES HIRED AFTER 81 INTO DEPT 10 OR 30
20. WAQTD NAMES OF EMPLOYEES ALONG WITH ANNUAL SALARY FOR THE EMPLOYEES WORKING AS MANAGER OR CLERK INTO DEPT 10 OR 30 20.
21. WAQTD ALL THE DETAILS ALONG WITH ANNUAL SALARY IF SAL IS BETWEEN 1000 AND 4000 ANNUAL SALARY MORE THAN 15000

DAY 9

15 May 2021 03:19 PM

SPECIAL OPERATORS :

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

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

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

Example :

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

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

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

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

- WAQTD empno , ename and salary of the employees whose empno Is 7902 or 7839 and getting salary more than 2925.

```
SELECT EMPNO,ENAME,SAL
FROM EMP
WHERE EMPNO IN (7902,7839) AND SAL>2925;
```

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

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

Example :

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

```
SELECT ENAME,DEPTNO
```

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

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

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

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

Syntax:

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

→ *Between Op works including the range*
→ *We cannot interchange the range !!.*

Example :

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

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

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

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

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

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

Syntax:

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

Example :

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

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

WHERE SAL NOT BETWEEN 1000 AND 3000;

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

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

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

⇒ **IS :** *"It is used to compare only NULL "*

Syntax: Column_Name **IS** NULL ;

Example :

EMP

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

- WAQTD name of the employee who is not getting salary .
- WAQTD name of the emp who doesn't get commission .
- WAQTD name , sal and comm of the emp if the emp doesn't earn both(SAL AND COMM) .

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

Syntax: Column_Name **IS NOT** NULL ;

Example :

- WAQTD name of the employee who is getting salary .

- WAQTD name of the emp who gets commission .

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

ASSIGNMENT QUESTIONS :

⇒ WAQTD NAMES AND DEPTNO , JOB OF EMPS WORKING AS CLERK IN DEPT 10 OR 20
⇒ WAQTD DETAILS OF EMPLOYEES WORKING AS CLERK OR MANAGER IN DEPT 10 AND EARNING SAL MORE THAN 1250 .
⇒ WAQTD NAMES OF EMPLOYEES WORKING IN DEPT 10 ,20 , 30 , 40 AS A CLERK
⇒ WAQTD DETAILS OF EMPLOYEES WITH EMPNO 7902,7839 IN DEPT 10 OR 30
⇒ WAQTD DETAILS OF EMPLOYEES WORKING AS MANAGER OR SALESMAN OR CLERK IN DEPT 40
⇒ WAQTD NAMES OF EMPLOYEES HIRED AFTER 81 AND BEFORE 87 AS A PRESIDENT

⇒ WAQTD DETAILS OF EMPLOYEES EARNING MORE THAN 1250 BUT LESS THAN 3000

⇒ WAQTD NAMES OF EMPLOYEES HIRED AFTER 2020 INTO DEPT 10 OR 30

⇒ WAQTD NAMES OF EMPLOYEES WITH SALARY FOR THE EMPLOYEES WORKING AS MANAGER OR CLERK INTO DEPT 10 OR 30

⇒ WAQTD ALL THE DETAILS IDSAL IS BETWEEN 1000 AND 4000 IN DEPT 10 OR 20 AND WORKING AS A MANAGER OR ANALYST

DAY 10

18 May 2021 08:10 PM

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

To achieve pattern matching we use special characters .

- Percentile (%)
- Underscore (_)

- ***Percentile (%)*** : it is a special character which is used to match any number of character, any number of time / number of character.
- ***Underscore (_)*** : it is a special character which is used to match exactly once but any character.

Syntax: Column_Name/ exp LIKE 'pattern' ;

Example :

- WAQTD details of an employee whose name is SMITH .

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

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

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

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

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

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

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

NOTE:

'%A'	SELECT LAST CHAR
'A%'	SELECT FIRST CHAR
'%A%'	SELECT ANYWHERE CHAR

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

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

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

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

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

- WAQTD names of the employee if the emp has char 'V' present atleast 2 times .

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

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

- WAQTD names of the employee if the emp's salary's last 2 digit is 50 rupees .
  SELECT ENAME FROM EMP WHERE SAL LIKE '%50';

- WAQTD names of the employees hired in November .
  SELECT ENAME FROM EMP WHERE HIREADTE LIKE '%NOV%';

- WAQTD names and sal of employee if the emp annual salary's last 3 digit is 200 rupees
  SELECT ENAME ,SAL FROM EMP WHERE SAL*12 LIKE '%200';

NOT LIKE : Opposite of Like .

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

- WAQTD names of the employees NOT hired in November
  SELECT ENAME FROM EMP WHERE HIREDATE NOT LIKE '%NOV%';

- WAQTD names of the employee if the emp has not a char 'A' as his Third character .

ASSIGNMENT ON SEPCIAL OPERATORS :

1) LIST ALL THE EMPLOYEES WHOSE COMMISSION IS NULL
2) LIST ALL THE EMPLOYEES WHO DON'T HAVE A REPORTING MANAGER
3) LIST ALL THE SALESMEN IN DEPT 30
4) LIST ALL THE SALESMEN IN DEPT NUMBER 30 AND HAVING SALARY GREATER THAN 1500
5) LIST ALL THE EMPLOYEES WHOSE NAME STARTS WITH 'S' OR 'A'
6) LIST ALL THE EMPLOYEES EXCEPT THOSE WHO ARE WORKING IN DEPT 10 & 20.
7) LIST THE EMPLOYEES WHOSE NAME DOES NOT START WITH 'S'
8) LIST ALL THE EMPLOYEES WHO ARE HAVING REPORTING MANAGERS IN DEPT 10
9) LIST ALL THE EMPLOYEES WHOSE COMMISSION IS NULL AND WORKING AS CLERK
10) LIST ALL THE EMPLOYEES WHO DON'T HAVE A REPORTING MANAGER IN DEPTNO 10 OR 30
11) LIST ALL THE SALESMEN IN DEPT 30 WITH SAL MORE THAN 2450
12) LIST ALL THE ANALYST IN DEPT NUMBER 20 AND HAVING SALARY GREATER THAN 2500
13) LIST ALL THE EMPLOYEES WHOSE NAME STARTS WITH 'M' OR 'J'
14) LIST ALL THE EMPLOYEES WITH ANNUAL SALARY EXCEPT THOSE WHO ARE WORKING IN DEPT 30
15) LIST THE EMPLOYEES WHOSE NAME DOES NOT END WITH 'ES' OR 'R'
16) LIST ALL THE EMPLOYEES WHO ARE HAVING REPORTING MANAGERS IN DEPT 10 ALONG WITH 10% HIKE IN SALARY
17) DISPLAY ALL THE EMPLOYEE WHO ARE 'SALESMAN'S HAVING 'E' AS THE LAST BUT ONE CHARACTER IN ENAME BUT SALARY HAVING EXACTLY 4 CHARACTER
18) DISPLAY ALL THE EMPLOYEE WHO ARE JOINED AFTER YEAR 81
19) DISPLAY ALL THE EMPLOYEE WHO ARE JOINED IN FEB
20) LIST THE EMPLOYEES WHO ARE NOT WORKING AS MANAGERS AND CLERKS IN DEPT 10 AND 20 WITH A SALARY IN THE RANGE OF 1000 TO 3000.

DAY 11

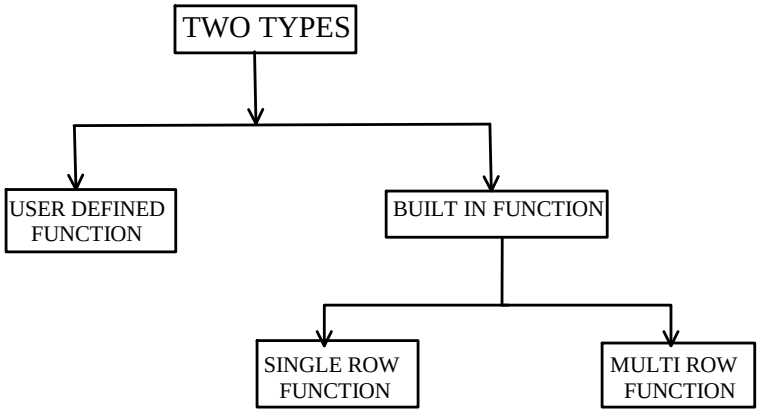
18 May 2021 08:11 PM

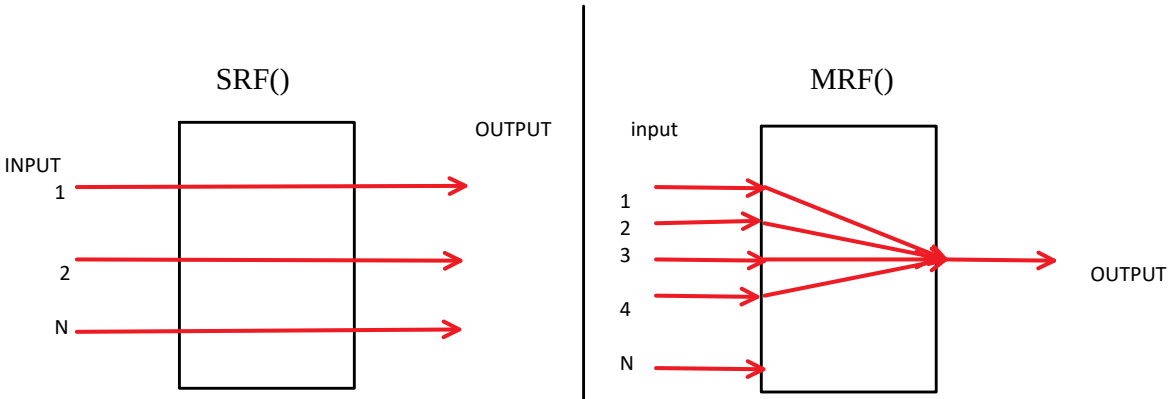
FUNCTIONS

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

There are 3 main components of a function

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





EX: LENGTH ()

ENAME	***SAL***
ALLEN	100
JONES	200
MR.ALEX	300

LENGTH(ENAME)
5
5
7

SELECT LENGTH(ENAME)
FROM EMP;

function *argument* *return type*

EX: MAX()

ENAME	SAL
ALLEN	100
JONES	200
MR.ALEX	300

SELECT MAX(SAL)
FROM EMP;

MAX(SAL)
300

- **Single row function:**
  1. SRF execute ROW BY ROW

2. SRF takes one input process and executes it to generate an output and takes next input and so on
3. IF We pass 'n' number input to the single row function, it returns 'n' number of output

❖ **Multi Row Functions;**

1. MRF execute GROUP BY GROUP
2. It takes all the inputs at one shot and then executes and provides A single output .
3. If we pass 'n' number of inputs to a MRF( ) it returns '1' Output .

List of MRF ( )

- MAX( ) : it is used to obtain the maximum value present in the column
- MIN ( ) : it is used to obtain the minimum value present in the column
- SUM ( ) : it is used to obtain the summation of values present in the column
- AVG( ) : it is used to obtain the average of values present in the column
- COUNT( ) : it is used to obtain the number of values present in the column

NOTE :

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

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

Examples :

- WAQTD maximum salary given to a manager .
- WAQTD Total salary given to dept 10
- WAQTD number of employees earning more than 1500 in dept 20
- WAQTD number of employee having 'E' in their names .
- WAQTD minimum salary given to the employees working as clerk in Dept 10 or 20 .
- WAQTD number of employees hired after 1982 and before 1985 into Dept 10 or 30 .
- WAQTD number of employees getting commission .
- WAQTD maximum salary given to employees if the emp has character 'S' in the name and works as a Manager in dept 10 with as salary of more than 1800 .
- WAQTD number of employees working in dept 10 or 30 and getting commission without the salary .
- WAQTD maximum salary given to a manager working in dept 20 and also his comm must be greater than his salary .

ASSIGNEMENT ON MRF( )

⇒ WAQTD NUMBER OF EMPLOYEES GETTING SALARY LESS THAN 2000 IN DEPTNO 10
⇒ WAQTD TOTAL SALARY NEEDED TO PAY EMPLOYEES WORKING AS CLERK
⇒ WAQTD AVERAGE SALARY NEEDED TO PAY ALL EMPLOYEES
⇒ WAQTD NUMBER OF EMPLOYEES HAVING 'A' AS THEIR FIRST CHARACTER
⇒ WAQTD NUMBER OF EMPLOYEES WORKING AS CLERK OR MANAGER
⇒ WAQTD TOTAL SALARY NEEDED TO PAY EMPLOYEES HIRED IN FEB
⇒ WAQTD NUMBER OF EMPLOYEES REPORTING TO 7839 (MGR)
⇒ WAQTD NUMBER OF EMPLOYEES GETTING COMISSION IN DEPTNO 30
⇒ WAQTD AVG SAL , TOTAL SAL , NUMBER OF EMPS AND MAXIMUM SALARY GIVEN TO EMPLOYEES WORKING AS PERSIDENT
⇒ WAQTD NUMBER OF EMPLOYEES HAVING 'A' IN THEIR NAMES
⇒ WAQTD NUMBER OF EMPS AND TOTAL SALARY NEEDED TO PAY THE EMPLOYEES WHO HAVE 2 CONSICUTIVE L's IN THEIR NAMES
⇒ WAQTD NUMBER OF DEPARTMENTS PRESENT IN EMPLOYEE TABLE
⇒ WAQTD NUMBER OF EMPLOYEES HAVING CHARACTER 'Z' IN THEIR NAMES
⇒ WAQTD NUMBER OF EMPLOYEES HAVING '$' IN THEIR NAMES .
⇒ WAQTD TOTAL SALARY GIVEN TO EMPLOYEES WORKING AS CLERK IN DEPT 30
⇒ WAQTD MAXIMUM SALARY GIVEN TO THE EMPLOYEES WORKING AS ANALYST
⇒ WAQTD NUMBER OF DISTINCT SALARIES PRESENT IN EMPLOYEE TABLE
⇒ WAQTD NUMBER OF JOBS PRESENT IN EMPLOYEE TABLE
⇒ WATQD AVG SALARY GIVEN TO THE CLERK
⇒ WAQTD MINIMUM SALARY GIVEN TO THE EMPLOYEES WHO WORK IN DEPT 10 AS MANAGER OR A CLERK

DIFFRENACE BETWEEN SRF AND MRF

Single Row Function (SRF)	Multi Row Function (MRF)
It is also called **server function**	It is also called **aggregate/group** function
Execute **row by row**	Execute **group by group**
We can pass '**n**' number of inputs, it returns '**n**' number of outputs	We can pass '**n**' number of inputs, it returns '**single**' number of outputs
used in **where clause/statement/ keyword**	Not Used in **where clause/statement/ keyword**
Along with the **SRF** can use any other column	Along with the M**RF** cannot use any other column
It not **ignores** the null	It **ignores** the null

DAY 12

18 May 2021 08:13 PM

GROUP & FILTERING

GROUPING:GROUP-BY Clause

Group by clause is used to group the records .

NOTE :

1. Group By clause executes row by row .
2. After the execution of Group By clause we get Groups .
3. Therefore any clause that executes after group by must execute Group By Group .
4. The Column_Name or expression used for grouping can be used In select clause .
5. Group By clause can be used without using Where clause .

SYNTAX:

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

ORDER OF EXECUTION:

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

EMP

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

Example :

WAQTD number of employees working in each dept .

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

OUTPUT OF GROUP BY CLAUSE

OUTPUT OF FROM CLAUSE

EID	ENAME	SAL	DEPTNO
1	A	100	20
2	B	200	10
3	C	300	30
4	D	100	10
5	E	200	10

10

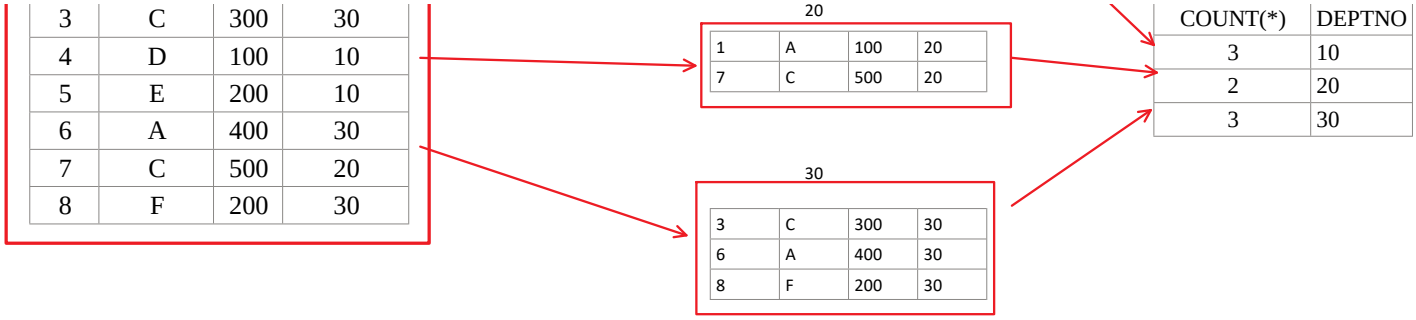
2	B	200	10
4	D	100	10
5	E	200	10

20

1	A	100	20
7	C	500	20

OUTPUT OF THE SELECT CLAUSE

COUNT(*)	DEPTNO
3	10
2	20



Questions :

- WAQTD number of employees working in each dept except the Employee working as analyst .
- WAQTD maximum salary given to each job .

- WAQTD number of employees working in each job if the employees Have character 'A' in their names .
  WAQTD number of employees getting commission in each dept .

ASSIGNMENT QUESTIONS ON GROUP BY

1. WAQTD NUMBEROF EMPLOYEES WORKING IN EACH DEPARTEMENT EXCEPT PRESIDENT.
2. WAQTD TOTAL SALARY NEEDED TO PAY ALL THE EMPLOYEES IN EACH JOB.
3. WAQTD NUMBER OF EMPLOYEEES WORKING AS MANAGER IN EACH DEPARTMENT .
4. WAQTD AVG SALARY NEEDED TO PAY ALL THE EMPLOYEES IN EACH DEPARTMENT EXCLUDING THE EMPLOYEES OF DEPTNO 20.
5. WAQTD NUMBER OF EMPLOYEES HAVING CHARACTER 'A' IN THEIR NAMES IN EACH JOB .
6. WAQTD NUMBER OF EMPLOYEES AND AVG SALARY NEEDED TO PAY THE EMPLOYEES WHO SALARY IN GREATER THAN 2000 IN EACH DEPT.
7. WAQTD TOTAL SALARY NEEDED TO PAY AND NUMBER OF SALESMANS IN EACH DEPT.
8. WAQTD NUMBER OF EMPLOYEES WITH THEIR MAXIMUM SALARIES IN EACH JOB.
9. WAQTD MAXIMUM SALARIES GIVEN TO AN EMPLOYEE WORKING IN EACH DEPT.
10. WAQTD NUMBER OF TIMES THE SALARIES PRESENT IN EMPLOYEE TABLE .

FILTERING : HAVING Clause

" Having Clause is used to Filter the Group "

NOTE:

1. Having clause execute group by group
2. Having clause always use after group by clause.
3. In Having clause we write group filter condition
4. Having clause cannot we used without the group by clause
5. In Having clause we can use multi row function(MRF) as condition.

SYNTAX:

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

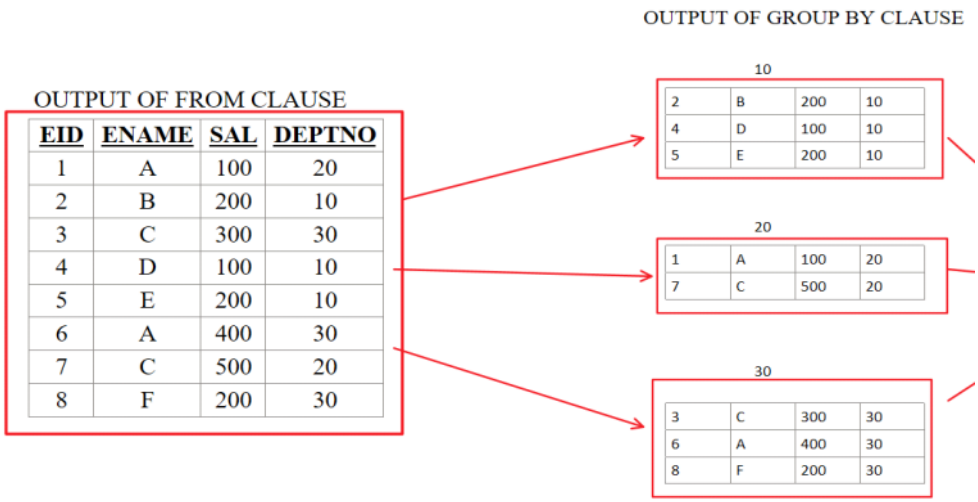
ORDER OF EXECUTION:

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

Example :

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

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



OUTPUT OF HAVING CLAUSE

DEPTNO 10

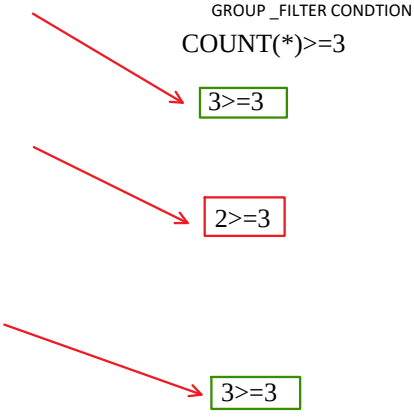
2	B	200	10
4	D	100	10
5	E	200	10

DEPTNO 30

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

OUTPUT OF SELECT CLAUSE

DEPTNO	COUNT(*)
10	3
30	3



Questions :

1. WAQTD the designations in which there are at least 2 employees Present .
2. WAQTD the names that are repeated .
3. WAQTD names that are repeated exactly twice .
4. WAQTD the salary that is repeated .
5. WAQTD number of employees working in each dept having At least 2 emp's Character 'A' or 'S' in their names .
6. WAQTD job and total salary of each job , if the total salary Of each job is greater than 3450 .
7. WAQTD job and total salary of the employees if the employees Are earning more than 1500.
8. WAQTD Job wise maximum salary if the maximum salary Of each job exceeds 2000 .

NOTE :

Differentiate between Where and Having .

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

Where clause executes before Group by clause .	Having clause executes After group by clause .

ASSIGNMENT QUESTIONS ON HAVING CLAUSE

1. WAQTD DNO AND NUMBER OF EMP WORKING IN EACH DEPT IF THERE ARE ATLEAST 2 CLERKS IN EACH DEPT
2. WAQTD DNO AND TOTAL SAALARYNEEDED TO PAY ALL EMP IN EACH DEPT IF THERE ARE ATLEAST 4 EMP IN EACH DEPT
3. WAQTD NUMBER OF EMP EARNING SAL MORE THAN 1200 IN EACH JOB AND THE TOTAL SAL NEEDED TO PAY EMP OF EACH JOB MUST EXCEES 3800
4. WAQTD DEPTNO AND NUMBER OF EMP WORKING ONLY IF THERE ARE 2 EMP WORKING IN EACH DEPT AS MANAGER .
5. WAQTD JOB AND MAX SAL OF EMP IN EACH JOB IF THE MAX SAL EXCEEDS 2600
6. WAQTD THE SALARIES WHICH ARE REPEATED IN EMP TABLE
7. WAQTD THE HIREDATE WHICH ARE DUPLICATED IN EMP TABLE
8. WAQTD AVG SALARY OF EACH DEPT IF AVG SAL IS LESS THAN 3000
9. WAQTD DEPTNO IF THERE ARE ATLEAST 3 EMP IN EACH DEPT WHOS NAME HAS CHAR 'A' OR 'S' .
10. WAQTD MIN AND MAX SALARIES OF EACH JOB IF MIN SAL IS MORE THAN 1000 AND MAX SAL IS LESS THAN 5000 .

ORDER BY CLAUSE

Order by clause is used to arrange the records either in ascending order or descending order.

1. Order by clause execute after select clause (execute at the last).
2. Order by clause should always be written at the last.
3. By default order by clause sorts the records in ascending order.

SYNTAX:

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

ORDER OF EXECUTION:

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

EXAMPLE

SAL
120
300
50

```
SELECT SAL
FROM EMP
ORDER BY SAL [ASC];
```

SAL
50
120
300

```
SELECT SAL
FROM EMP
ORDER BY SAL DESC
```

SAL
300
120
50

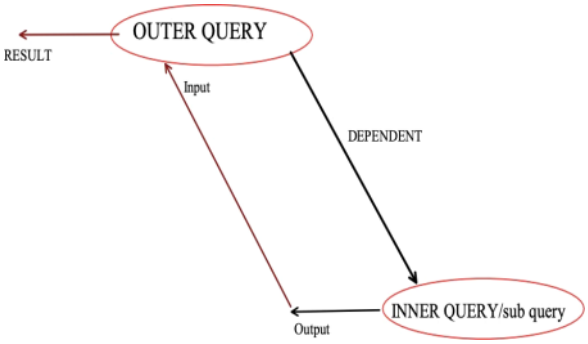
DAY 13

5 July 2021 12:24 PM

SUB-QUERY:

"A query written inside another query is known As sub query "

Working Principle :



Let us consider two queries Outer Query and Inner Query .

a. Inner Query executes first and produces an Output .
b. The Output of Inner Query is given / fed as an Input to Outer Query .
c. The Outer Query generates the Result.
d. Therefore we can state that 'the Outer Query is dependent on Inner Query' and

this is the Execution Principle of Sub Query.

Why / When Do we use SUB QUERY :

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

Example :

EMP

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

1. WAQTD names of the employees earning more than 2500 .
2. WAQTD names of the employees earning less than MILLER .
3. WAQTD name and deptno of the employees working in the same Dept as SMITH .
4. WAQTD name and hiredate of the employees if the employee Was hired after JONES .
5. WAQTD all the details of the employee working in the same Designation as KING .
6. WAQTD name , sal , deptno of the employees if the employees Earn more than 2000 and work in the same dept as JAMES .
7. WAQTD all the details of the employees working in the Same designation as MILLER and earning more than 1500.
8. WAQTD details of the employees earning more than SMITH But less than KING .
9. WAQTD name , sal and deptno of the employees if the employee Is earning commission in dept 20 and earning salary more than Scott .
   WAQTD name and hiredate of the employees whose name ends with 'S' and hired after James .
10. WAQTD names of the employees working in the same dept as JAMES and earning salary more than ADAMS
    and working in the same job role as MILLER and hired after MARTIN .
11. WAQTD all the details of the employees working as salesman in the dept 20 and earning commission more than Smith and hired after KING .
12. WAQTD number of employees earning more than SMITH and less than MARTIN .
13. WAQTD Ename and SAL for all the employees earning more than JONES .

NOTE:

- In the Inner Query / Sub Query we cannot select more than One column .
- The corresponding columns need not be same , but the datatypes of those has to be same .

ASSIGNMENT ON CASE 1

1) WAQTD NAME OF THE EMPLOYEES EARNING MORE THAN ADAMS

2) WAQTD NAME AND SALARY OF THE EMPLOYEES EARNING LESS THAN KING
3) WAQTD NAME AND DEPTNO OF THE EMPLOYEES IF THEY ARE WORKING IN THE SAME DEPT AS JONES
4) WAQTD NAME AND JOB OF ALL THE EMPLOYEES WORKING IN THE SAME DESIGNATION AS JAMES
5) WAQTD EMPNO AND ENAME ALONG WITH ANNUAL SALARY OF ALL THEEMPLOYEES IF THEIR ANNUAL
6) SALARY IS GREATER THAN WARDS ANNUAL SALARY.
7) WAQTD NAME AND HIREDATE OF THE EMPLOYEES IF THEY ARE HIRED BEFORE SCOTT
8) WAQTD NAME AND HIREDATE OF THE EMPLOYEES IF THEY ARE HIRED AFTER THE PRESIDENT
9) WAQTD NAME AND SAL OF THE EMPLOYEE IF THEY ARE EARNING SAL LESS THAN THE EMPLOYEE WHOS EMPNO IS 7839
10) WAQTD ALL THE DETAILS OF THE EMPLOYEES IF THE EMPLOYEES ARE HIRED BEFORE MILLER
11) WAQTD ENAME AND EMPNO OF THE EMPLOYEES IF EMPLOYEES ARE EARNING MORE THAN ALLEN
12) WAQTD ENAME AND SALARY OF ALL THE EMPLOYEES WHO ARE EARNING MORE THAN MILLER BUT LESS THAN ALLEN .
13) WAQTD ALL THE DETAILS OF THE EMPLOYEES WORKING IN DEPT 20 AND WORKING IN THE SAME DESIGNATION AS SMITH
14) WAQTD ALL THE DETAILS OF THE EMPLOYEES WORKING AS MANAGER IN THE SAME DEPT AS TURNER
15) WAQTD NAME AND HIREDATE OF THE EMPLOYEES HIRED AFTER 1980 AND BEFORE KING
16) WAQTD NAME AND SAL ALONG WITH ANNUAL SAL FOR ALL EMPLOYEES WHOS SAL IS LESS THAN BLAKE AND MORE THAN 3500
17) WAQTD ALL THE DETAILS OF EMPLOYEES WHO EARN MORE THAN SCOTT BUT LESS THAN KING
18) WAQTD NAME OF THE EMPLOYEES WHOS NAME STARTS WITH 'A' AND WORKS IN THE SAME DEPT AS BLAKE
19) WAQTD NAME AND COMM IF EMPLOYEES EARN COMISSION AND WORK IN THE SAME DESIGNATION AS SMITH
20) WAQTD DETAILS OF ALL THE EMPLOYEES WORKING AS CLERK IN THE SAME DEPT AS TURNER .
21) WAQTD ENAME, SAL AND DESIGNATION OF THE EMPLOYEES
22) WHOS ANNUAL SALARY IS MORE THAN SMITH AND LESS THAN KING.

DAY 14

5 July 2021 12:24 PM

SUB-QUERY

CASE-2:

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

Example :

EMP

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

DEPT

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

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

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

3. WAQTD Location of ADAMS

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

4. WAQTD names of the employees working in Location L2.
5. WAQTD number of employees working in dept D3 .
6. WAQTD Ename , sal of all the employee earning more than Scott and working in dept 20 .
7. WAQTD all the details of the employee working as a Manager In the dept Accounting .
8. WAQTD all the details of the employee working in the same designation as Miller and works in location New York .
9. WAQTD number of employees working as a clerk in the same deptno as SMITH and earning more than KING hired after MARTIN in the location BOSTON .
10. WAQTD maximum salary given to a person working in DALLAS .

EXAMPLE 2:

CUSTOMER

CID	CNAME	CNO	PID
1	SMITH	12345	101
2	JONES	12349	102
3	ALLEN	12346	103

PRODUCT

PID	PNAME	PRICE	DISCOUNT
101	Iphone 11	51000	0
102	Iphone 12	74000	1000
103	iPad	40000	100

11. WATQD the name of the product that smith purchased .

```
SELECT PNAME
FROM PRODUCT
WHERE PID =(SELECT PID
            FROM CUSTOMER
            WHERE CNAME='SMITH');
```

12. WAQTD the names of the customers who have purchased Iphone 12.

ASSIGNMENT ON CASE 2 :

1.WAQTD DNAME OF THE EMPLOYEES WHOS NAME IS SMITH
2.WAQTD DNAME AND LOC OF THE EMPLOYEE WHOS ENAME IS KING
3.WAQTD LOC OF THE EMP WHOS EMPLOYEE NUMBER IS 7902
4.WAQTD DNAME AND LOC ALONG WITH DEPTNO OF THE EMPLOYEE WHOS NAME ENDS WITH 'R' .
5.WAQTD DNAME OF THE EMPLOYEE WHOS DESIGNATION IS PRESIDENT
6.WAQTD NAMES OF THE EMPLOYEES WORKING IN ACCOUNTING DEPARTMENT
7.WAQTD ENAME AND SALARIES OF THE EMPLOYEES WHO ARE WORKING IN THE LOCATION CHICAGO
8.WAQTD DETAILS OF THE EMPLOYEES WORKING IN SALES
9.WAQTD DETAILS OF THE EMP ALONG WITH ANNUAL SALARY IF EMPLOYEES ARE WORKING IN NEW YORK
10. WAQTD NAMES OF EMPLOYEES WORKING IN OPERATIONS DEPARTMENT

ASSIGNMENT ON CASE 1 & 2

1.WAQTD NAMES OF THE EMPLOYEES EARNING MORE THAN SCOTT IN ACCOUNTING DEPT
2.WAQTD DETAILS OF THE EMPLOYEES WORKING AS MANAGER IN THE LOCATION CHICAGO
3.WAQTD NAME AND SAL OF THE EMPLOYEES EARNING MORE THAN KING IN THE DEPT ACCOUNTING
4.WAQTD DETAILS OF THE EMPLOYEES WORKING AS SALESMAN IN THE DEPARTEMENT SALES
5.WAQTD NAME , SAL , JOB , HIREDATE OF THE EMPLOYEES WORKING IN OPERATIONS DEPARTMENT AND HIRED BEFORE KING
6.DISPLAY ALL THE EMPLOYEES WHOSE DEPARTMET NAMES ENDING 'S'.
7.WAQTD DNAME OF THE EMPLOYEES WHOS NAMES HAS CHARACTER 'A' IN IT .
8.WAQTD DNAME AND LOC OF THE EMPLOYEES WHOS SALARY IS RUPEES 800 .
9.WAQTD DNAME OF THE EMPLOYEES WHO EARN COMISSION
10.WAQTD LOC OF THE EMPLOYEES IF THEY EARN COMISSION IN DEPT 40

DAY 15

5 July 2021 12:24 PM

TYPES OF SUB - QUERY :

1) SINGLE ROW SUB QUERY
2) MULTI ROW SUB QUERY

EXAMPLE:

EMP

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

DEPT

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

❖ SINGLE ROW SUB QUERY:

- If the sub query returns exactly 1 record / value we call it as Single Row Sub Query.
- *If it returns only 1 value then we can use the normal operators Or the Special Operators to compare the values.*

EXAMPLE

- WAQTD DNAME OF ALLEN

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

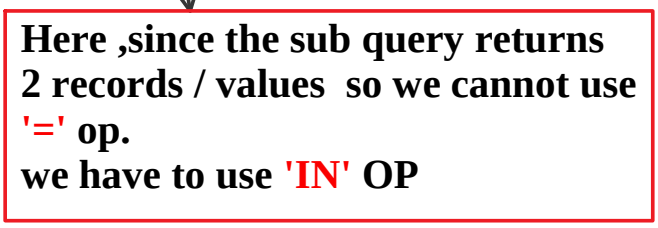
❖ MULTI ROW SUB QUERY:

- If the sub query returns more than1 record / value we call it as Multi Row Sub Query.
- *If it returns more than 1 value then we **cannot use the normal operators** We have to use only Special Operators to compare the values.*

EXAMPLE

- *WAQTD DNAMES OF ALLEN OR SMITH.*

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




```
SELECT DNAME
FROM DEPT
WHERE DEPTNO   IN  (SELECT DEPTNO
                    FROM EMP
                    WHERE ENAME IN('ALLEN' , 'SMITH');
```

Note :

It is difficult to identify whether a query Belongs Single or Multi row So , it is always recommended to use Special Operators to Compare The values.

1. WAQTD Ename and salary of the employees earning *more than* Employees of dept 10.

EMP

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

= OR IN
<> OR NOT IN

>(SINGLE VALUES)
>ALL (MULTI)

5	SMITH	2500	10

>(SINGLE VALUES)
>ALL (MULTI)

1500>1500,3000

```
SELECT ENAME ,SAL
FROM EMP
WHERE SAL > (SELECT SAL
             FROM EMP
             WHERE DEPTNO= 10);
```

2000
1500
2500

HERE we cannot use > symbol to compare multiple values

we cannot use IN or NOT IN OP as well because it is used for = and <> symbol

therefore we have to use SUB QUREY OP to comparing relational operators such as (>,<,<=,>=)

Sub Query Operators :

1. ALL
2. ANY
3. EXIST
4. NOT EXIST

❖ ***ALL :***

"It is special Op used along with a relational Op (> , < , > = , <=) to compare the values present at the RHS".

ALL Op returns true if all the values at the RHS have satisfied the condition .

EAXMPLE

1. WAQTD Ename and salary of the employees earning more than Employees of dept 10.

```
 SELECT ENAME,SAL (CLARK, 3000)
FROM EMP
WHERE SAL > ALL (SELECT SAL
                 FROM EMP
                 WHERE DEPTNO =10);
```

2000
1500
2500

1000

FROM EMP
WHERE DEPTNO =10);

2500

1000
2000
3000
1500
2500

1000 > 2000 FALSE
1000> 1500 FALSE
1000 > 2500 FALSE

REJECTED

2000 > 2000 F
2000>1500 T
2000 >2500 F
REJECTED

3000>2000 T
3000> 1500 T
3000>2500 T
SELECTED

1500 >2000 F
1500> 1500 F
1500>2500 F
REJECTED

2500>2000 T
2500>1500 T
2500> 2500 F
REJECTED

ANY:

"It is special Op used along with a relational Op (> , < , > = , <=) to compare the values present at the RHS ".

ANY Op returns true if one of the values at the RHS have satisfied the condition.

Example :

- WAQTD ENAME,SAL WHO ARE EARNING SAL MORE THAN ANY ONE OF THE EMPLOYEE IN THE DEPTNO 10

SELECT ENAME,SAL (BLAKE ,CLARK ,SMITH 2000,3000,2500)
FROM EMP
WHERE SAL >ANY (SELECT SAL
FROM EMP
WHERE DEPTNO=10);

2000
1500
2500

1000
2000
3000
1500
2500

1000>2000 F
1000>1500 F
1000 > 2500 F
REJECTED
2000>2000 F
2000>1500 T
2000>2500 F
SELECETD

3000>2000 T
3000> 1500 T
3000>2500 T
SELECTED
1500>2000 F
1500>1500 F
1500> 2500 F
REJECTED

2500>2000 T
2500>1500 T
2500>2500 F
SELECTED

1. WAQTD name of the employee if the employee earns less than ALL The employees working as salesman .

```
SELECT ENAME
 FROM EMP
 WHERE SAL <ALL(SELECT SAL
                FROM EMP
                WHERE JOB='SALESMAN');
```

2. WAQTD name of the employee if the employee earns less than At least a salesman
3. WAQTD names of the employees earning more than ADAMS .

<u>ASSIGNMENT ON TYPES OF SUB QUERY .</u>

1. WAQTD NAME OF THE EMPLOYEES EARNING SALARY MORE THAN THE SALESMAN
2. WAQTD DETAILS OF THE EMPLOYEES HIRED AFTER ALL THE CLERKS
3. WAQTD NAME AND SALARY FOR ALL THE EMPLOYEES IF THEY ARE EARNING LESS THAN ATLEST A MANAGER
4. WAQTD NAME AND HIREDATE OF EMPLOYEES HIRED BEFORE ALL THE MANAGERS
5. WAQTD NAMES OF THE EMPLOYEES HIRED AFTER ALL THE MANAGERS AND EARNING SALARY MORE THAN ALL THE CLERKS
6. WAQTD DETAILS OF THE EMPLOYEES WORKING AS CLERK AND

HIRED BEFORE ATLEST A SALESMAN

7. WAQTD DETAILS OF EMPLOYEES WORKING IN ACCOUNTING OR SALES DEPT
8. WAQTD DEPARTMENT NAMES OF THE EMPOYEES WITH NAME SMITH , KING AND MILLER
9. WAQTD DETAILS OF EMPLOYEES WORKING NEWYORK OR CHICAGO
10. WAQTD EMP NAMES IF EMPLOYEES ARE HIRED AFTER ALL THE EMPLOYEES OF DEPT 10

NESTED SUB QUERY:

"A sub query written inside a sub query is known as Nested Subquery"

- WE CAN NEST ABOUT 255 SUB QUERIES

SAL
1000
2000
4000
3000
5000

11. WAQTD MAXIMUM SALARY GIVEN TO AN EMPLOYEE .
    SELECT MAX(SAL) FROM EMP;

12. WAQTD SECOND MAXIMUM SALARY GIVEN TO AN EMPLOYEE.
    SELECT MAX(SAL) 3000
    FROM EMP
    WHERE SAL < (SELECT MAX(SAL) 5000
    FROM EMP);

WAQTD 3RD MAXIMUM SALARY.

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

FROM EMP));

WAQTD 4TH MAXIMUM SALARY.

13. WAQTD 3 MINIMUM SALARY .

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

14. WAQTD DEPT NAME OF THE EMPLOYEE GETTING 2ND MINIMUM SALARY .

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

REMEMBER :

MAXIMUM	MAX( )	<
MINIMUM	MIN( )	>

ASSIGNMENT ON NESTED SUB QUERY :

15. WAQTD 2ND MINIMUM SALARY
16. WAQTD 5TH MAXIMUM SALARY
17. WAQTD NAME OF THE EMPLOYEE EARNING 3RD MAXIMUM SALARY
18. WAQTD EMPNO OF THE EMPLOYEE EARNING 2ND MAXIMUM SALARY
19. WAQTD DEPARTMENT NAME OF AN EMPLOYEE GETTING 4TH MAX SAL
20. WAQTD DETAILS OF THE EMPLOYEE WHO WAS HIRED 2ND
21. WAQTD NAME OF THE EMPLOYEE HIRED BEFORE THE LAST EMPLOYEE
22. WAQTD LOC OF THE EMPLOYEE WHO WAS HIRED FIRST
23. WAQTD DETAILS OF THE EMPLOYEE EARNING 7TH MINIMUM SALARY
24. WAQTD DNAME OF EMPLOYEE GETTING 2ND MAXIMUM SALARY

DAY 16

5 July 2021 12:24 PM

EMPLOYEE AND MANAGER RELATION:

EMP

EMPNO	ENAME	MGR
1	ALLEN	3
2	SMITH	1
3	JAMES	2
4	KING	3

NOTE:

To find **Manager/Reporting mgr**	Select **MGR** in Sub Q
To find **Employees**	Select **EMPNO** in Sub Q

CASE 1: TO FIND MANAGER

1. WAQTD name of Allen's manager

```
SELECT ENAME (JAMES)
FROM EMP
WHERE EMPNO= (SELECT MGR  (3)
                FROM EMP
                WHERE ENAME='ALLEN');
```

2. WAQTD name of SMITH's manager .

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

3. WAQTD Names of smiths manager's manager.

```
SELECT ENAME
FROM EMP
WHERE EMPNO IN(SELECT MGR
                FROM EMP
                 WHERE EMPNO IN(SELECT MGR
```

```
                                        FROM EMP
                                        WHERE ENAME='SMITH'));
```

4. WAQTD dname of King's Manager.

```
    SELECT DNAME
    FROM DEPT
   WHERE DEPTNO IN (SELECT DEPTNO
                    FROM EMP
                     WHERE EMPNO  IN (SELECT MGR
                                      FROM EMP
                                      WHERE ENAME='KING'));
```

5. WAQTD Location of Adams's manager's manager .

```
    SELECT LOC
      FROM DEPT
      WHERE DETPNO IN (SELECT DEPTNO
      FROM EMP
WHERE EMPNO IN(SELECT MGR
FROM EMP
WHERE EMPNO IN(SELECT MGR
FROM EMP
WHERE ENAME='ADAMS')));
```

CASE 2: TO FIND EMPLOYEES

1. WAQTD Names of the employees reporting to KING.

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

2. WAQTD Name and salary given to the employees reporting To James .

```
      SELECT ENAME,SAL
       FROM EMP
       WHERE MGR IN (SELECT EMPNO
                          FROM EMP
                           WHERE ENAME='JAMES');
```

3. WAQTD dname of the employee reporting to President .

```
    SELECT DNAME
FROM DEPT
WHERE DEPTNO IN (SELECT DEPTNO
                        FROM EMP
                        WHERE MGR IN (SELECT EMPNO
                                      FROM EMP
                                      WHERE JOB='PRESIDENT'));
```

4. WAQTD Department details of the employees who are reporting to MILLER .

```
SELECT *
FROM DEPT
WHERE DEPTNO IN (SELECT DEPTNO
FROM EMP
WHERE MGR IN (SELECT EMPNO
FROM EMP
WHERE ENAME='MILLER'));
```

ASSIGNMENT ON EMP AND MANAGER RELATION

1. WAQTD SMITHS REPORTING MANAGER'S NAME
2. WAQTD ADAMS MANAGER'S MANAGER NAME
3. WAQTD DNAME OF JONES MANAGER
4. WAQTD MILLER'S MANAGER'S SALARY
5. WAQTD LOC OF SMITH'S MANAGER'S MANAGER.
6. WAQTD NAME OF THE EMPLOYEES REPORTING TO BLAKE
7. WAQTD NUMBER OF EMPLPOYEES REPORTING TO KING
8. WAQTD DETAILS OF THE EMPLOYEES REPORTING TO JONES
9. WAQTD ENAMES OF THE EMPLOYEES REPORTING TO BLAKE'S MANAGER
10. WAQTD NUMBER OF EMPLOYEES REPORTING TO FORD'S MANAGER

SUB QUERY:

What is Sub Query ?
Explain ? (draw)
Why ? When ?
Types of Sub Query

- **Single Row Sub Query**
- **Multi Row Sub Query**

Sub Query Operators
1.ALL
2.ANY
Nested Sub Query
EMP-MGR

DAY 17

5 July 2021 12:24 PM

JOINS:

"The process of retrieval of data from multiple tables simultaneously is known as JOINS ".

WHEN/ WHY do we use joins:

"whenever we have to select attribute from more than one table then we use joins"

Types of JOINS .

1. CARTESIAN JOIN / CROSS JOIN
2. INNER JOIN / EQUI JOIN
3. OUTER JOIN
    a) LEFT OUTER JOIN
    b) RIGHT OUTER JOIN
    c) FULL OUTER JOIN
4. SELF JOIN
5. NATURAL JOIN .

1. CARTESIAN JOIN / CROSS JOIN :

In Cartesian Join a record from table 1 will be merged with All the records of table 2

Number of Columns in the Result table : will be equivalent to the summations of columns present in both the tables .

Number of Col = Number of Col T1 + Number of Col T2
2 + 2
= 4 Columns

Number of Rows in the Result table : will be equivalent to the product of number of rows present in the both the tables .

Number of Rows = Number of Rows T1 x Number of Rows T2
3 x 3
= 9 Rows .

Example

EMP

ENAME	DEPTNO
A	30
B	10
C	20

DEPT

DNAME	DEPTNO
D1	10
D2	20
D3	30

Result Table :

ENAME	DEPTNO	DNAME	DEPTNO
A	30	D1	10
A	30	D2	20
A	30	D3	30
B	10	D1	10
B	10	D2	20
B	10	D3	30
C	20	D1	10

C	20	D2	20
C	20	D3	30

SYNTAX:

1. **ANSI** [American National Standard Institute]

```
SELECT Column_Name
FROM Table_Name1 CROSS JOIN Table_Name2 ;
```

2. **Oracle**

```
SELECT Column_Name
FROM Table_Name1 , Table_Name2 ;
```

Example :

WAQTD ename and dept name for all the employees .

```
SELECT ENAME,DNAME
 FROM EMP,DEPT;
      OR
 SELECT ENAME,DNAME
 FROM EMP CROSS JOIN DEPT;
```

INNER JOIN:

"It is used to Obtain only Matching Records " Or " A records which has a Pair " .

EMP

ENAME	DEPTNO
A	20
B	30
C	10

DEPT

DNAME	DEPTNO
D1	10
D2	20
D3	30

JOIN Condition : It is a condition on which the two tables Are merged .

Syntax: Table_Name1.Col_Name = Table_Name2.Col_Name

Join Condition :EMP.DEPTNO = DEPT.DEPTNO

20=10	**F**
20=20	**T**
20=30	**F**

30=10	F
30=20	F
30=30	T

10=10	T
10=20	F
10=30	F

Result Table :

ENAME	DEPTNO	DNAME	DEPTNO
A	20	D2	20
B	30	D3	30
C	10	D1	10

SYNTAX:

SYNTAX:

1. ANSI [American National Standard Institute]

```
SELECT Column_Name
FROM Table_Name1 INNER JOIN Table_Name2
ON < JOIN_CONDITION> ;

SELECT *
FROM EMP INNER JOIN DEPT
ON EMP.DEPTNO = DEPT.DEPTNO ;
```

2. Oracle

```
SELECT Column_Name
FROM Table_Name1, Table_Name2
WHERE <JOIN_CONDITION > ;

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

1. WAQTD ename and dept name for all the employees .

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

2. WAQTD ename and loc for all the employees working as Manager .

```
SEELECT ENAME,LOC
FROM EMP,DEPT
WHERE EMP.DEPTNO=DEPT.DEPTNO AND JOB='MANAGER';
```

3. WAQTD ename , sal and dname of the employee working as Clerk in dept 20 with a salary of more than 1800 .

```
SELECT ENAME,SAL,DNAME
FROM EMP, DEPT
WHERE EMP.DEPTNO =DEPT.DEPTNO AND  JOB='CLERK' AND DEPTNO=20 AND SAL>1800;
```

COLUMN IS AMBIGUOSLY DEFINED

```
SELECT ENAME,SAL,DNAME
    FROM EMP, DEPT
   WHERE EMP.DEPTNO =DEPT.DEPTNO AND  JOB='CLERK' AND DEPT.DEPTNO=20 AND SAL>1800;
```

4. WAQTD ename,deptno , dname and loc of the employee earning more than 2000 in New York .

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

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

ASSIGNMENT ON INNER JOIN :

1. NAME OF THE EMPLOYEE AND HIS LOCATION OF ALL THE EMPLOYEES .

2. WAQTD DNAME AND SALARY FOR ALL THE EMPLOYEE WORKING IN ACCOUNTING.
3. WAQTD DNAME AND ANNUAL SALARY FOR ALL EMPLOYEES WHOS SALARY IS MORE THAN 2340
4. WAQTD ENAME AND DNAME FOR EMPLOYEES HAVING CAHARACTER 'A' IN THEIR DNAME
5. WAQTD ENAME AND DNAME FOR ALL THE EMPLOYEES WORKING AS SALESMAN
6. WADTD DNAME AND JOB FOR ALL THE EMPLOYEES WHOS JOB AND DNAME STARTS WITH CHARACTER 'S'
7. WAQTD DNAME AND MGR NO FOR EMPLOYEES REPORTING TO 7839
8. WAQTD DNAME AND HIREDATE FOR EMPLOYEES HIRED AFTER 83 INTO ACCOUNTING OR RESEARCH DEPT
9. WAQTD ENAME AND DNAME OF THE EMPLOYEES WHO ARE GETTING COMM IN DEPT 10 OR 30
10. WAQTD DNAME AND EMPNO FOR ALL THE EMPLOYEES WHO'S EMPNO ARE (7839,7902) AND ARE WORKING IN LOC NEW YORK.

DAY 18

5 July 2021 12:24 PM

OUTER JOIN:

"It is used to Obtain Un-Matched Records "

Left Outer Join:

"It is used to obtain Un-Matched Records of Left Table Along with Matching Records ".

SYNTAX:

ANSI [American National Standard Institute]

```
SELECT Column_Name
FROM Table_Name1 LEFT |OUTER| JOIN Table_Name2
ON < JOIN_CONDITION> ;

SELECT *
FROM EMP LEFT JOIN DEPT
ON EMP.DEPTNO = DEPT.DEPTNO ;
```

2. Oracle

```
SELECT Column_Name
FROM Table_Name1 , Table_Name2
WHERE Table1.Col_Name = Table2.Col_Name (+) ;

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

OUTER JOIN OPERATOR

Example :

EMP

ENAME	DEPTNO
A	20
B	30
C	10
D	60

Left

DEPT

DNAME	DEPTNO
D1	10
D2	20
D3	30
D4	40

Right

Result Table :

ENAME	DEPTNO	DNAME	DEPTNO
A	20	D2	20
C	10	D1	10
B	30	D3	30
D	60	NULL	NULL

WAQTD names and dname of all the employees even though the employees Don't work in any dept .

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

ENAME	DNAME
A	D2
C	D1
B	NULL
D	NULL

Right Outer Join:

"It is used to obtain Un-Matched Records of Right Table Along with Matching Records ".

SYNTAX:

ANSI [American National Standard Institute]

```
SELECT Column_Name
FROM Table_Name1 RIGHT[OUTER] JOIN Table_Name2
ON < JOIN_CONDITION> ;

SELECT *
FROM EMP RIGHT JOIN DEPT
ON EMP.DEPTNO = DEPT.DEPTNO  ;
```

Oracle

```
SELECT Column_Name
FROM Table_Name1 , Table_Name2
WHERE Table1.Col_Name (+) = Table2.Col_Name  ;

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

Example :

DEPT

DNAME	DEPTNO
D1	10
D2	20
D3	30
D4	40

Left

EMP

ENAME	DEPTNO
A	20
B	30
C	10
D	Null

Right

Result Table :

DNAME	DEPTNO	ENAME	DEPTNO
D1	10	C	10
D2	20	A	20
D3	30	B	30
NULL	NULL	D	NULL

WAQTD names and dname of all the employees even though the there are no employees in a dept .

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

Full Outer Join :

"It is used to obtain Un-Matched Records of both Left & Right Table Along with Matching Records ".

SYNTAX

1. ANSI [American National Standard Institute]

```
SELECT Column_Name
FROM Table_Name1 FULL [OUTER] JOIN Table_Name2
ON < JOIN_CONDITION> ;

SELECT *
FROM EMP FULL JOIN DEPT
ON EMP.DEPTNO = DEPT.DEPTNO ;
```

Example :

EMP

ENAME	DEPTNO
A	20

DEPT

DNAME	DEPTNO
D1	10

ENAME	DEPTNO
A	20
B	Null
C	10
D	Null

DNAME	DEPTNO
D1	10
D2	20
D3	30
D4	40

Result Table :

ENAME	DEPTNO	DNAME	DEPTNO
A	20	D2	20
C	10	D1	10
B	NULL	NULL	NULL
D	NULL	NULL	NULL
NULL	NULL	D3	30
NULL	NULL	D4	40

MATCHING RECORDS OF BOTH TABLE

UNMATCHED RECORDS OF LEFT TABLE

UNMATCHED RECORDS OF RIGHT TABLE

WAQTD names and dname of all the employees and depts even though the employees Don't work in any dept and a dept having no employees.

```
SELECT ENAME,DNAME
FROM EMP FULL JOIN DEPT
ON EMP.DEPTNO = DEPT.DEPTNO;
```

SELF JOIN :

"Joining a table by itself is known as Self Join "

Why ? / When ?

"Whenever the data to select is in the same table but present In different records we use self-join ".

SYNTAX:

ANSI [American National Standard Institute]

```
SELECT Column_Name
FROM Table_Name1 JOIN Table_Name2
ON < JOIN_CONDITION> ;

SELECT *
FROM EMP E1 JOIN EMP E2
ON E1.MGR = E2.EMPNO ;
```

Oracle

```
SELECT Column_Name
FROM Table_Name1T1 , Table_Name2 T2
WHERE < Join_Condition > ;

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

EXAMPLE

EMP E1

EMPNO	ENAME	E1.MGR
1	ALLEN	3
2	SMITH	1
3	MILLER	2

JOIN CONDTION E1.MGR = E2.EMPNO

RESULT TABLE;

E1.EMPNO	E1.ENAME	E1.MGR	E2.EMPNO	E2.ENAME	E2.MRG
1	ALLEN	3	3	MILLER	2
2	SMITH	1	1	ALLEN	3
3	MILLER	2	2	SMITH	1

EMP DETAILS - E1 MANAGERS DETAILS -E2

1. WAQTD Ename and Manager's name for all the employees .

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

2. WAQTD Ename , sal along with manager's name and manager's salary for all the employees .

```
SELECT E1.ENAME,E1.SAL, E2.ENAME MGR_NAME ,E2.SAL MGR_SAL
FROM EMP E1,EMP E2
WHERE E1.MGR=E2.EMPNO ;
```

3. WAQTD Ename , manager's name along with their deptno If employee is working as clerk .

```
SELECT E1.ENAME, E2.ENAME MGR_NAME, E1.DEPTNO ,E2.DEPTNO
MGR_DEPT
FROM EMP E1,EMP E2
WHERE E1.MGR=E2.EMPNO AND E1.JOB IN ' CLERK';
```

4. WAQTD Ename , manager's job if manager works as Analyst .
5. WAQTD Ename and manager's name along with their job if emp and manager are working

for same designation .

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

6. WAQTD Ename emp salary manager's name manager's salary If manager earns more than employee .
7. WAQTD Ename and manager's name along with manager's commission if manager earns commission .

NOTE :

TO join 'N' number of tables we need to write 'N-1' number of join conditions

ASSIGNMENT ON SELF JOIN :

- WAQTD NAME OF THE EMPLOYEE AND HIS MANAGER'S NAME IF EMPLOYEE IS WORKING AS CLERK
- WAQTD NAME OF THE EMPLOYEE AND MANAGER'S DESIGNATION IF MANAGER WORKS IN DEPT 10 OR 20
- WAQTD NAME OF THE EMP AND MANAGERS SALARY IF EMPLOYEE AND MANAGER BOTH EARN MORE THAN 2300
- WAQTD EMP NAME AND MANAGER'S HIREDATE IF EMPLOYEE WAS HIRED BEFORE1982
- WAQTD EMP NAME AND MANAGER'S COMM IF EMPLOYEE WORKS AS SALESMAN AND MANAGER WORKS IN DEPT 30
- WAQTD EMP NAME AND MANAGER NAME AND THEIR SALARIES IF EMPLOYEE EARNS MORE THAN MANAGER
- WAQTD EMP NAME AND HIREDATE , MANAGER NAME AND HIREDATE IF MANAGER WAS HIRED BEFORE EMPLOYEE
- WAQTD EMP NAME AND MANAGER NAME IF BOTH ARE WORKING IN SAME JOB
- WAQTD EMP NAME AND MANAGER NAME IF MANAGER IS WORKING AS ACTUAL MANAGER
- WAQTD EMP NAME AND MANAGER NAME ALONG WITH THEIR ANNUAL SALARIES IF EMPLOYEE WORKS IN DEPT 10 , 20 AND MANAGER'S SAL IS GREATER THAN EMPLOYEES SALARY .
- WAQTD EMPLOYEE'S NAME AND MANAGER'S DESIGNATION FOR ALL THE EMPLOYEES
- WAQTD EMPLOYEE'S NAME AND MANAGER'S SALARY FOR ALL THE EMPLOYEES IF MANAGER'S SALARY ENDS WITH 50

5. NATURAL JOIN:

"It behaves as **INNER JOIN** if there is a relation between the given two tables , else it behaves as **CROSS JOIN" .**

SYNTAX:

ANSI :

```
SELECT Col_Name
FROM Table_Name1 NATURAL JOIN Table_Name2;
```

EXAMPLE

EMP

ENAME	DEPTNO
A	20
B	30
C	10

DEPT

DNAME	DEPTNO
D1	10
D2	20
D3	30

RESULT TABLE:

ENAME	DEPTNO	DNAME
A	20	D2
B	30	D3
C	10	D1

EMP

ENAME	DEPTNO
A	20
B	30
C	10

CUSTOMER

CNAME	CID
X	101
Y	102
Z	103

RESULT TABLE

ENAME	DEPTNO	CNAME	CID
A	20	X	101
A	20	Y	102
A	20	Z	103
B	30	X	101

B	30	Y	102
B	30	Z	103
C	10	X	101
C	10	Y	102
C	10	Z	103

EXAMPLE (NOTE)/ (BLOCK DIAGRAM)

WAQTD ENAME AND DNAME FOR ALL THE EMP.

EMP DEPT

EMP.DEPTNO =DEPT. DEPTNO

WAQTD ENAME,DNAME AND MANGER'S NAME

E1.MGR=E2.EMPNO

EMP DEPT EMP E2 3-1=2

EMP.DEPTNO=DEPT.DEPTNO

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

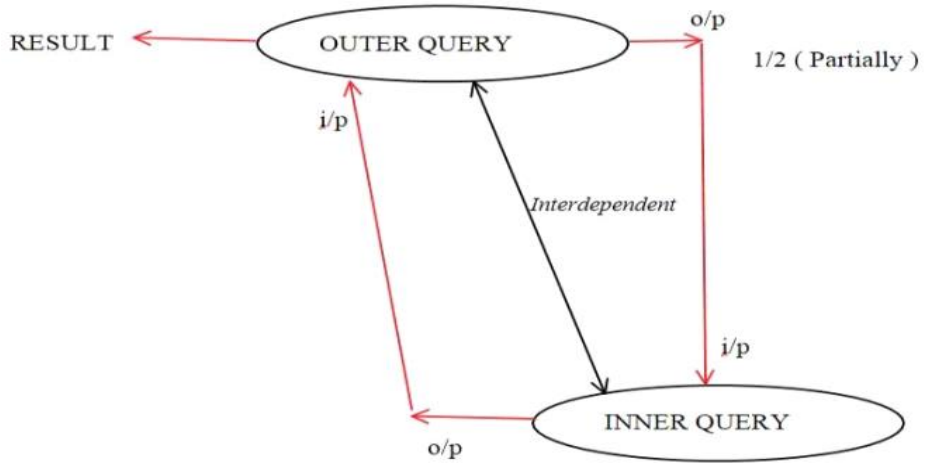
DAY 19

5 July 2021 12:24 PM

CO - RELATED SUB QUERY

" *A query written inside another query such that the outer query and the inner query are Dependent on each other , this is known as Co-Related Sub-Query* ".

WORKING PRINCIPLE :



Let us consider two queries inner and outer query respectively ,

1. Outer query executes first but partially (50 PER)
2. The partially executed output is given as an input to the inner Query
3. The inner query executes completely and generates an output
4. The output of inner query is fed as an input to the Outer query and Outer Query produces the result .
5. Therefore, we can state that the outer query and the inner query both are INTERDEPENDENT (dependent on each other) .

NOTE :

i. In co-related sub query a Join condition is a must , And must be written only in the Inner Query .
ii. Co-Related sub query works with the principles of both SUB QUERY & JOINS .

DIFFERENCE BETWEEN SUB QUERY AND CO RELATED SUB QUERY .

SUB QUERY	CO-RELATED SUB QUERY
Inner query executes first	Outer query executes first
Outer query is dependent on inner query	Both are interdependent
Join condition not mandatory	Join condition is mandatory and must be written in inner query
Outer query executes Once	Outer query executes Twice .

Example :

DEPT

DNAME	DNO
D1	10
D2	20
D3	30
D4	40

EMP

ENAME	DNO
A	20
B	10
C	20
D	30

D1 D2 D3 NULL

```
SELECT DNAME
FROM DEPT
WHERE DNO IN (SELECT DNO
              FROM EMP
              WHERE EMP.DEPTNO=DEPT.DEPTNO);
```

10
(20,20)
30
NULL

10
20
30
40

10=10	TRUE
10=20	F
10=20	F
10=30	F
20=10	F
20=20	T
20=20	T
20=30	F

30=10	F
30=20	F
30=20	F
30=30	T
40=10	F
40=20	F
40=20	F
40=30	F

20
10
20
30

10
20
30
40

EX:

1. WAQTD dname in which there are employees working .

```
SELECT DNAME,DEPTNO
FROM DEPT
WHERE DEPTNO IN (SELECT DEPTNO
                 FROM EMP
                 WHERE EMP.DEPTNO=DEPT.DEPTNO);
```

2. WAQTD dname in which there are no employees working .

```
SELECT DNAME,DEPTNO
FROM DEPT
WHERE DEPTNO  NOT IN (SELECT DEPTNO
                      FROM EMP
                      WHERE EMP.DEPTNO=DEPT.DEPTNO);
```

EXISTS & NOT EXISTS OPERATORS :

1. **EXISTS :** " Exists Op is a Unary Op (One Operand) which can accept one operand towards RHS and that operand has to be co related sub -query
   - *Exists Op returns true if the Sub Query returns Any value other than Null.*

example:

d1,d2,d3

SELECT DNAME
FROM DEPT
WHERE ***exists*** *(SELECT DNO*
FROM EMP
WHERE EMP.DEPTNO=DEPT.DEPTNO);

10
(20,20)
30
NULL

10=10	TRUE
10=20	F
10=20	F
10=30	F
20=10	F
20=20	T
20=20	T
20=30	F

30=10	F
30=20	F
30=20	F
30=30	T
40=10	F
40=20	F
40=20	F
40=30	F

2. **NOT EXISTS :**

"NOT EXISTS operator is a unary op(one operand) which can accept one operand towards RHS and that operand has to a co related sub query"

- *Not Exists Op returns true if the Sub Query returns NULL .*

To Find MAX & MIN salary :

To find MAXIMUM salary :

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

4TH MAX SAL , 8TH, 10TH

E1.SAL
1000
3000
2000

```
SELECT SAL
FROM EMP E1
WHERE (SELECT COUNT(DISTINCT SAL)
       FROM EMP E2
       WHERE E1.SAL < E2.SAL) IN (3,7,9);
```

3000
2000
4000
5000

To find MINIMUM salary :

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

ASSIGNMENT QUESTIONS:

1. WAQTD 5TH MAX SALARY
2. WAQTD 3RD MIN SALARY
3. WAQTD 2ND 5TH 8TH MAX SAL
4. WAQTD 7TH MAX SAL
5. WAQTD 5TH MIN SAL

SAL
1000
6000
2000
3000
4000
5000
8000
7000
6000
9000

DAY 20

10 July 2021 02:58 PM

SINGLE ROW FUNCTIONS:

1. LENGTH( )
2. CONCAT( )
3. UPPER( )
4. LOWER( )
5. INITCAP( )
6. REVERSE( )
7. SUBSTR()
8. INSTR( )
9. REPLACE( )
10. MOD( )
11. TRUNC( )
12. ROUND( )
13. MONTHS_BETWEEN()
14. LAST_DAY()
15. TO_CHAR( )
16. NVL( )
17. ASCII()
18. POWER()
19. SQRT()
20. SIN()
21. TAN()

LENGTH: "It is used to count the number of characters present In the given string ".

SYNTAX: **LENGTH ('string')**

Example :

WAQTd count number of characters present in 'SMITH' .

```
SELECT LENGTH (ENAME)
FROM EMP
WHERE ENAME='SMITH'
```

LENGTH(ENAME)
5

```
SELECT LENGTH('SMITH')
FROM DUAL;
```

```
SELECT LENGTH('KRISHNA')
FROM  DUAL;
```

NOTE :

DUAL TABLE

It is a DUMMY table which has 1 col and 1 row .Which is used to output the result .

- DESC DUAL ;
- SELECT * FROM DUAL ;

CONCAT( ) : "It is used to join the given two strings '

SYNTAX : CONCAT ('string1' , 'String2')

Example :

```
SELECT CONCAT('MR' , ENAME)
FROM EMP
WHERE ENAME='SMITH';
```

MR SMITH

UPPER( ) : "It is used to convert a given string to upper case "

SYNTAX: UPPER ('string')

LOWER( ) :"It is used to convert a given string to lower case "

SYNTAX: LOWER('string')

INITCAP( ):"It is used to convert a given string to initial capital letter case ".

SYNTAX: INITCAP('string')

REVERSE( ): "It is used to reverse a given string ".

SYNTAX: REVERSE('string')

SUBSTR : "It is used to extract a part of string from the given Original string " .

SYNTAX: **SUBSTR** ('Original String' , Position, [Length])

***NOTE**: Length is not mandatory , If length is not mentioned then Consider the complete string .*

Example :

-ve

-7	-6	-5	-4	-3	-2	-1
Q	**S**	**P**	**I**	**D**	**E**	**R**
1	2	3	4	5	6	7

+ve

Example :	SUBSTR('QSPIDER' , 2 , 3)	SPI
Example :	SUBSTR('QSPIDER' , 3 , 3)	PID
Example :	SUBSTR('QSPIDER' , 2)	SPIDER
Example :	SUBSTR('QSPIDER' , 1 , 6)	QSPIDE
Example :	SUBSTR('QSPIDER' , 4, 1)	I
Example :	SUBSTR('QSPIDER' , 1 , 1)	Q
Example :	SUBSTR('QSPIDER' , 7 , 1)	R
Example :	SUBSTR('QSPIDER' , 6)	ER
Example :	SUBSTR('QSPIDER' , 1 , 3)	QSP
Example :	SUBSTR('QSPIDER' , 6 , 6)	ER
Example :	SUBSTR('QSPIDER' , -2 , 1)	E
Example :	SUBSTR('QSPIDER' , -5 , 3)	PID
Example :	SUBSTR('QSPIDER' , -7 , 2)	QS
Example :	SUBSTR('QSPIDER' , -1)	R

WAQTD extract first 3 characters of the emp names .

```
SELECT SUBSTR(ENAME,1,3)
FROM EMP;
```

WAQT extract last 3 characters of the employee names .

```
SELECT SUBSTR(ENAME,-3)
FROM EMP;
```

WAQTD to display **first half of employee names .(50% OF NAME)**

ENAME	OUTPUT
SMITH	SM
MILLER	MIL
JONES	JO
WARD	WA

SELECT SUBSTR(ENAME , 1 , LENGTH(ENAME) / 2) FROM EMP ;

	SUBSTR('SMITH' , 1 , LENGTH ('SMITH') / 2)
	SUBSTR('SMITH' , 1 , 5 / 2) 2
	SUBSTR('SMITH' , 1 , 2)
	SM

WARD	SUBSTR(ENAME , 1 , LENGTH(ENAME) / 2)
	SUBSTR('WARD' , 1 , LENGTH ('WARD') / 2)
	SUBSTR('WARD' , 1 , 4 / 2)
	SUBSTR('WARD' , 1 , 2)
	WA

SUBSTR(ENAME , 1 , **LENGTH(ENAME) / 2**)

SMITH

WAQT to display **second half of employee names .(50-100 OF ENAME)**

ENAME	OUTPUT
SMITH	ITH
MILLER	LER
JONES	NES
WARD	RD

SELECT SUBSTR(ENAME , LENGTH(ENAME) / 2 + 1) FROM EMP ;

SMITH	SUBSTR(ENAME , LENGTH(ENAME) / 2 +1)
	SUBSTR('SMITH' , LENGTH ('SMITH') / 2 +1)
	SUBSTR('SMITH' , 5 / 2 +1)
	SUBSTR('SMITH' , 3)
	ITH

WARD	SUBSTR(ENAME , LENGTH(ENAME) / 2+1)
	SUBSTR('WARD' , LENGTH ('WARD') / 2+1)
	SUBSTR('WARD' , 4 / 2 +1)
	SUBSTR('WARD' , 3)
	RD

REPLACE ( ) : "It is used to replace a string with another string in The original string.

SYNTAX:**REPLACE** ('Original_String' , 'string' , ['new_String'])

Example :	REPLACE ('**B**A**N**A**N**A' , '**A**' , '**C**')	BCNCNC
Example :	REPLACE ('BANANA' , 'N' , 'ABC')	BAABCAABCA
Example :	REPLACE ('OPPO' , 'OPPO' , 'SAMSUNG')	SAMSUNG
Example :	REPLACE ('BANANA' , 'A')	BNN
Example :	REPLACE ('ENGINEERING' , 'E')	NGINRING
Example :	REPLACE ('ENGINEERING' , 'E' , '123')	123NGIN123123RIN G

NOTE : if the third argument is not mentioned the default Value of it is Null .

WAQTD the number of times char 'A' is present in BANANA !!!
SELECT LENGTH('BANANA') - LENGTH(REAPLCE('BANANA','A'))
FROM DUAL; LENGTH('BNN')

6-3 =3 TIMES PRESENT

WAQTD to count number of time 'A' is present in 'MALAYALAM'

DAY 21

10 July 2021 02:58 PM

9. INSTR ( ):"*it is used to obtain the **position** in which the string is present in the Original string* ".

It is used to search for a string in the Original string if present it returns the POSITION Else it returns **0** ".

SYNTAX

INSTR('ORIGINAL_STRING', 'STRING' , POSITION [, OCCURANCE]

Note : *if occurrence is not Mentioned then , the default value of Occurrence is 1* .

B	**A**	**N**	**A**	**N**	**A**
1	2	3	4	5	6

Example : INSTR('BANANA' , 'A' , 1 , 1)	Pos:2
Example : INSTR('BANANA' , 'A' , 2 , 1)	Pos:2
Example : INSTR('BANANA' , 'A' , 1 , 2)	Pos: 4
Example : INSTR('BANANA' , 'A' , 1 , 3)	Pos:6
Example : INSTR('BANANA' , 'A' , 1 , 4)	Pos:0
Example : INSTR('BANANA' , 'A' , 4 , 2)	Pos:6
Example : INSTR('BANANA' , 'A' , 2)	Pos:2
Example : INSTR('BANANA' , 'N' , 2 , 1)	Pos:3
Example : INSTR('BANANA' , 'O' , 1 ,1)	Pos:0
Example : INSTR('BANANA' , 'NA' , 2 , 2)	POS:5
Example : INSTR('BANANA' , 'A' , 3 , 3)	POS:0
Example : INSTR('BANANA' , 'ANA' , 1 , 2)	POS:4

MOD( ) : "It is used to obtain modulus/remainder of the given number "

Syntax: MOD (m , n)

Example : SELECT MOD(5 , 2)
FROM DUAL ;

1 . WAQTD ENAMES OF THE EMPLOYEES WHO EARN SALARY IN MULTIPLES OF 3

```
SELECT ENAME
FROM EMP
WHERE MOD(SAL,3)=0
```

2. WAQTD DETAILS OF THE EMPLOYEE WHO HAVE ODD EID

```
SELECT *
FROM EMP
WHERE MOD(EMPNO,2)=1
```

ROUND( ) : this function is used to round of the given number to its nearest value

Syntax: ROUND (Number , [Scale])

The default value of scale is 0

Example : ROUND (5.6)	6
Example : ROUND (5.5)	6
Example : ROUND (5.4)	5
Example : ROUND (9.9)	10
Example : ROUND (9.4)	9
Example : round(253.5 , 1)	254

Round (253.5248 ,-1)= 250
Round (253.52 , -2)=300
Round (233.52 ,-2)=200

TRUNC( ): "It is similar to ROUND() but it always rounds-off the given number to the lower value "

SYNATX:TRUNC (NUMBER)

Example : TRUNC (5.6)	5
Example : TRUNC (5.5)	5
Example : TRUNC (5.4)	5
Example : TRUNC (9.9)	9
Example : TRUNC (9.4)	9
Example : TRUNC (8.6)	8

NOTE :

DATE COMMANDS :

i. **SYSDATE** : " it is used to obtain Todays Date "
ii. **CURRENT_DATE** : " it is also used to obtain todays date "
iii. **SYSTIMESTAMP** : "It is used to obtain date , time and time zone "

```
SQL> SELECT SYSDATE
     FROM DUAL ;

   SYSDATE
  ----------
'17-JUL-21'

SQL> SELECT CURRENT_DATE
     FROM DUAL ;

CURRENT_D
---------
'17-JUL-21'

SQL> SELECT SYSTIMESTAMP
     FROM DUAL ;

SYSTIMESTAMP
```

MONTHS_BETWEEN( ): **"It is used to Obtain the number of months present between the Given two dates "**

SYNTAX : MONTHS_BETWEEN(DATE1 ,DATE2)

```
SELECT TRUNC( MONTHS_BETWEEN( SYSDATE , HIREDATE ) ) || ' Months'
FROM EMP;
```

LAST_DAY( ): **" it is used to Obtain the last day in the particular of the given date" .**

SYNATX: LAST_DAY(DATE)

```
SELECT LAST_DAY( SYSDATE )
 FROM DUAL ;
```

TO_CHAR( ) :"It is used to convert the given date into String format based on the Model given "

SYNTAX : TO_CHAR(DATE,'FROMAT_MODELS')

Format Models :

i. YEAR : TWENTY TWENTY
ii. YYYY : 2020
iii. YY : 20
iv. MONTH : JULY
v. MON : JUL
vi. MM : 07
vii. DAY : WEDNESDAY
viii. DY : WED
ix. DD : 08
x. D : 4 (day of the week)
xi. HH24 : 17 hours
xii. HH12 : 5 hours
xiii. MI : 22 minutes
xiv. SS : 53 seconds
xv. 'HH12:MI:SS' : 5 : 22 : 53
xvi. 'DD-MM-YY' : 17 - 05 - 20
xvii. 'MM-DD-YYYY' : 05 - 17 - 2020

1. WAQTD DETAILS OF THE EMPLOYEE WHO WAS HIRED ON A SUNDAY .

```
SELECT *
FROM EMP
WHERE  TO_CHAR(HIRADTE, 'DAY')='SUNDAY';
```

2. WAUQTD DETAILS OF AN EMPLOYEE HIRED ON MONDAY AT 3PM

```
SELECT *
FROM EMP
WHERE TO _CHAR(HIREDATE,'DAY')='MONDAY' AND
TO_CHAR(HIREDATE,'HH12')=3;
```

NVL( ) : [NULL VALUE LOGIC] " It is used to eliminate the side effects of using null in arithmetic operations " .

ENAME	SAL	COMM
A	500	100
B	1000	NULL
C	2000	200
D	2000	NULL

ENAME	SAL+COMM
A	600
B	NULL
C	2200
D	NULL

BY Using Null value logic :

SYNTAX: NVL(ARGUMENT1,ARGUMENT 2)

Argument 1 : Here write any column / exp which can result In null .

Argument 2 : Here we write a numeric value which will be substituted if argument 1 results in Null , If argument 1 is NOT NULL then the same value will be considered .

```
SELECT ENAME,SAL+NVL(COMM,0)
FROM EMP;
```

ENAME	SAL	COMM
A	500	100
B	1000	NULL
C	2000	200
D	2000	NULL

ENAME	SAL+NVL(COMM,0)
A	(500+NVL(100,0)= 500+100=600
B	(1000+NVL(NULL,0)= 1000+0=1000
C	(2000+NVL(200,0))= 2000+200=2200
D	(2000+NVL(NULL,0))=2000+0=2000

RESULT
600
1000

2200
2000

ASCII(): IT IS USED TO DETERMINE THE ASCII VALUES FOR THE SPECIFIC CHARACTER

SYNTAX:

ASCII(STRING)

POWER(): IT IS USED TO GET THE VALUES OF A NUMBER RAISED TO THE POWER OF ANOTHER NUMBER

SYNTAX:

POWER(A , B)

A: BASE VALUE
B: EXPONENT

SQRT():IT IS USED TO GET THE SQUARE ROOT OF A NUMBER

SYNTAX

SQRT(NUMBER)

TAN(): IT IS USED TO GET THE TANGENT OF A NUMBER

SYNTAX:

TAN(NUMBER)

SIN(): IT IS USED TO GET THE SINE OF A NUMBER

SYNATX:

SIN(NUMBER)

ASSIGNEMT

1. List employees whose name having 4 characters
2. List employees whose job is having 7 characters
3. Find out how many times letter 'S' occurs in 'qspiders'
4. List the employees whose job is having last 3 characters as 'man'
5. List employees whose job is having first 3 characters as 'man'.
6. Display all the names whose name is having exactly 1 'L'
7. Display dept names which are having letter 'O'

1. Calculate number of L in string 'HELLLLL'
2. Display all the employees whose job has a string 'MAN'
3. Display all the employees whose job starts with string 'MAN'
4. Display all the employees whose job ends with string 'MAN'
5. Display first 3 characters of Ename in lower case and rest everything in upper case. If Ename is 'QSPIDERS' then display this as ‘qspIDERS’
6. Display the result from emp table as below. SMITH is a CLERK and gets salary 2000 Here SMITH is ename column, CLERK is JOB and 2000 is SAL column and rest everything is literal strings.
7. list the employees hired on a Wednesday
8. list the employees hired on a leap year
9. list the employees hired on a Sunday in the month of may

DAY 22

10 July 2021 03:01 PM

STATEMENTS ARE CLASSIFIED INTO 5 DIFFERENT TYPES

- DATA DEFINITION LANGUAGE (DDL)
- DATA MANIPULATION LANGUAGE (DML)
- TRANSACTION CONTROL LANGUAGE (TCL)
- DATA CONTROL LANGUAGE (DCL)
- DATA QUERY LANGUAGE (DQL)

DATA DEFINITION LANGUAGE (DDL):

" *DDL is used to construct an object in the database and deals with the Structure of the Object* "

It has 5 statements :

1. CREATE
2. RENAME
3. ALTER
4. TRUNCATE
5. DROP

1. CREATE : " IT IS USED TO BUILD / CONSTRUCT AN OBJECT "

HOW TO CREATE/BUILD A TABLE :

1. Name of the table (OBJECT)
   Tables cannot have same names .
2. Number of Columns .
3. Assign datatypes for the Columns. {MANDATORY }
4. Assign Constraints [NOT MANDATORY] .

EXAMPLE 1:

TABLE NAME: PRODUCT (OBJECT)
N0_OF COLUMNS: 4

COL_NAME	PID	PNAME	PRICE	DISCOUNT
DATA TYPE	NUMBER(2)	VARCHAR (10)	NUMBER(7,2)	NUMBER(6,2)
CONTARINTS	NOTNULL	NOTNULL	NOT NULL	NOT NULL
	UNIQUE			
			CHECK(PRICE>0)	
	PK			

```
CREATE TABLE PRODUCT (
PID NUMBER(2) PRIMARY KEY,
PNAME VARCHAR(10) NOT NULL,
PRICE NUMBER(7,2) NOT NULL CHECK(PRICE>0),
DISCOUNT NUMBER(7,2)
);
```

Syntax to create a table :

```
CREATE TABLE   Table_Name (
Column_Name1 datatype constraint_type ,
Column_Name2 datatype constraint_type ,
Column_Name3 datatype constraint_type ,
.
.
Column_NameN datatype constraint_type
   );
```

EXAMPLE 2:

TABEL NAME= CUSTOMER (OBJECT)
NO OF COL =5

CID	CNAME	PH_NO	ADDRESS	PID
NUMBER (3)	VARCHAR(15)	NUMBER(10)	VARCAHR(10)	NUMBER(2)
	NOT NULL	NOT NULL	NOT NULL	NOT NULL
		UNIQUE		
		CHECK(LENGTH (PH_NO=10))		
PK				
				FK

```
CREATE TABEL CUST (
CID NUMBER(3) PRIMARY KEY,
CNAME VARCHAR(15) NOT NULL,
PH_NO NUMBER(10) NOT NULL UNIQUE CHECK(LENGTH(PH_NO)=10),
ADDRESS VARCHAR(10) NOT NULL,
PID NUMBER(2) NOT NULL,
FOREIGN KEY (PID) REFERENCES PRODUCT(PID)
);
```

(FOREIGN KEY (COL_NAME) REFERENCES TABEL_NAME1 (COL_NAME)

OR

CONSTRAINT PID _FK FOREGIN KEY (PID) PRODUCT (PID)

2. RENAME : "IT IS USED TO CHANGE THE NAME OF THE OBJECT "

Syntax: RENAME Table_Name TO New_Name ;

RENAME CUST TO CUSTOMER;

3. ALTER :" IT IS USED TO MODIFY THE STRUCTURE OF THE TABLE "

- **TO ADD A COLUMN :**

Syntax: ALTER TABLE Table_Name
ADD Column_Name Datatype Constraint_type

EXAMPLE: ALTER TABLE CUSTOMER
ADD EMAIL VARCHAR(15) UNIQUE;

- **TO DROP A COLUMN :**

Syntax: ALTER TABLE Table_Name
DROP COLUMN Column_Name ;

EXAMPLE
ALTER TABLE CUSTOMER
DROP COLUMN EMAIL ;

- **TO RENAME A COLUMN :**

Syntax: ALTER TABLE Table_Name
RENAME COLUMN Column_Name TO new_Column_Name ;

EXAMPLE:
ALTER TABLE PRODUCT
RENAME COLUMN DISCOUNT TO DISC;

- **TO MODIFY THE DATATYPE :**

Syntax: ALTER TABLE Table_Name
MODIFY COLUMN_NAME New_Datatype;

- EXAMPLE:

ALTER TABEL CUSTOMER
MODIFY ADDRESS VARCHAR(15);

- **TO MODIFY NOT NULL CONSTRAINTS :**

Syntax: **ALTER TABLE Table_Name**
MODIFY COLUMN_NAME Existing_datatype [NULL]/NOT NULL;

Example : ALTER TABLE PRODUCT
MODIFY DISC NUMBER(7,2) Not Null ;

TRUNCATE : " IT IS USED TO REMOVE ALL THE RECORDS/ROWS/TUPLE FROM THE TABLE PREMANENTLY "

Syntax:TRUNCATE TABLE Table_Name ;

DROP: " IT IS USED TO REMOVE THE TABLE FROM THE DATABASE "

Syntax: DROP TABLE Table_Name ;

TO RECOVER THE TABLE:

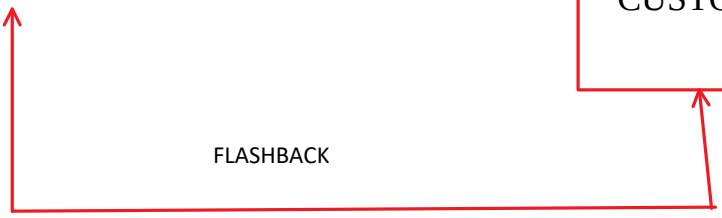
Syntax: FLASHBACK TABLE Table_Name
TO BEFORE DROP ;

EXAMPLE

ADDRESS: BIN$123QASSDAD

BIN FOLDER

CUSTOMER



FLASHBACK TABLE CUSTOMER
TO BEFORE DROP;

TO DELETE THE TABLE FROM BIN FOLDER :

Syntax: **PURGE TABLE Table_Name ;**

NOTE : DDL STATEMENTS ARE AUTO-COMMIT STATEMENTS

DAY 23

10 July 2021 03:01 PM

DATA MANIPULATION LANGUAGE (DML)

It is used to Manipulate the Object by performing insertion , updating and deletion .

1. INSERT
2. UPDATE
3. DELETE

1. **INSERT :** It is used to insert / create records in the table .

   Syntax: INSERT INTO Table_Name VALUES(v1 , v2 , v3 ……) ;

   PRODUCT:

PID	PNAME	PRICE	DISCOUNT
NUMBER(2)	VARCHAR(10)	NUMBER(7,2)	NUMBER(7,2

   EXAMPLE;

   INSERT INTO PRODUCT VALUES(11, 'REALME' , 5000.00, 1000.00);

2. **UPDATE :**It is used to modify an existing value .

   Syntax: UPDATE Table_Name
   SET Col_Name = Value , Col_Name = Value ,,,,,
   [WHERE stmt] ;

   EXAMPLE:
   CUSTOMER:

CID	CNAME	PH_NO	ADDRESS	PID
NUMBER(3)	VARCHAR(15)	NUMBER(10)	VARCHAR(15)	NUMBER(2)
1	DINGA	9874561231	BANGLORE	12
2	DINGI	9874654123	BANGLORE	22

EXAMPLE:

WAQTD UPADTE OF PHONE OF DINGA TO 1234567890

```
UPADTE  CUSTOMER
 SET PH_NO=1234567890, ADDRESS = 'MANGLORE'
 WHERE CNAME='DINGA';
```

3. **DELETE :** It is used to remove a particular record from the table .

Syntax: **DELETE**
FROM Table_Name
[WHERE stmt];

CUSTOMER:

CID	CNAME	PH_NO	ADDRESS	PID
NUMBER(3)	VARCHAR(15)	NUMBER(10)	VARCHAR(15)	NUMBER(2)
1	DINGA	9874561231	BANGLORE	12
2	DINGI	9874654123	BANGLORE	22

EXAAMPLE:
WAQTD REMOVE DINGA FROM THE LIST OF CUSTOMER
DELETE FROM CUSTOMER
WHERE CNAME='DINGA';

ASSIGNMENT ON DML STATEMENTS :

1.WAQT update the salary of employee to double their salary if He is working as a manager .

2.WAQT change the name of SMITH to SMIITH .

3.WAQT modify the job of KING to 'PRESIDENT' .

4.WAQT to change name of ALLEN to ALLEN MORGAN .

5.WAQT hike the salary of the employee to 10% . If employees earn less than 2000 as a salesman .

6.WAQ TO delete the employees who don’t earn commission .

7.WAQ to remove all the employees hired before 1987 in dept 20

8.Differentiate between TRUNCATE and DELETE statements .

3. TRANSACTION CONTROL LANGUAGE (TCL)

"It is used to control the transactions done on the database ".

The DML Operations performed on the Database are known as Transactions such as Insertion ,

Updating and Deletion .

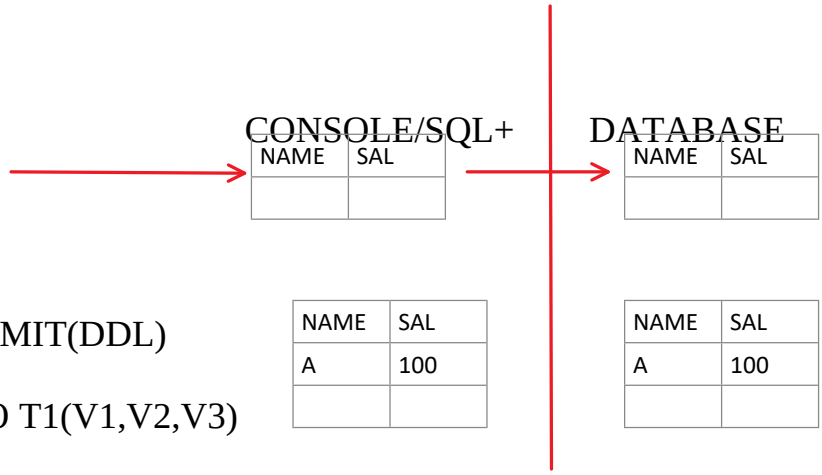
We have 3 Statements :

1. COMMIT
2. ROLLBACK
3. SAVEPOINT

1.COMMIT : "This statement is used to SAVE the transactions into the DB ".

Syntax: COMMIT ;

EXAMPLE:



COMMIT;

2.ROLLBACK :

This statement is used to Obtain only the saved data from the DB .
It will bring you to the point where you have committed for the last time .

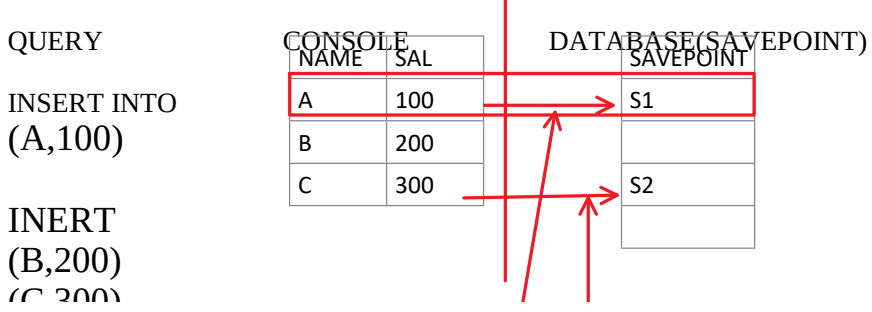
SYNTAX: ROLLBACK ;

3. SAVEPOINT :

This statement is used to mark the positions or restoration points . (nothing related to DB) .

SYNTAX: SAVEPOINT Savepoint_Name ;

EXAMPLE:



INERT
(B,200)
(C,300)

ROLLBACK TO S1;

ROLLBACK TO S1
ROLLBACK TO S2

SYNATX : ROLLBACK TO SAVEPOINT_NAME

4. DATA CONTROL LANGUAGE :

"This statement is used to control the flow of data between the users ".

We have 2 statements :
1. GRANT
2. REVOKE

1.GRANT : THIS STATEMENT IS USED TO GIVE PERMISSION TO A USER .

```
SYNTAX: GRANT SQL_STATEMENT
        ON TABLE_NAME
        TO USER_NAME ;
```

2.REVOKE : THIS STATEMENT IS USED TO TAKE BACK THE PERMISSION FROM THE USER .

```
SYNTAX: REVOKE SQL_STATEMENT
        ON TABLE_NAME
        FROM USER_NAME ;
```

EXAMPLE:

USER1: SCOTT

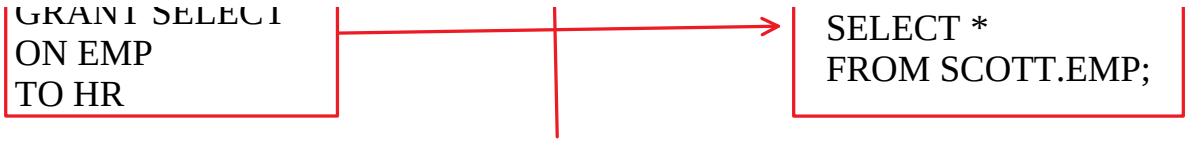
EMP	SAL

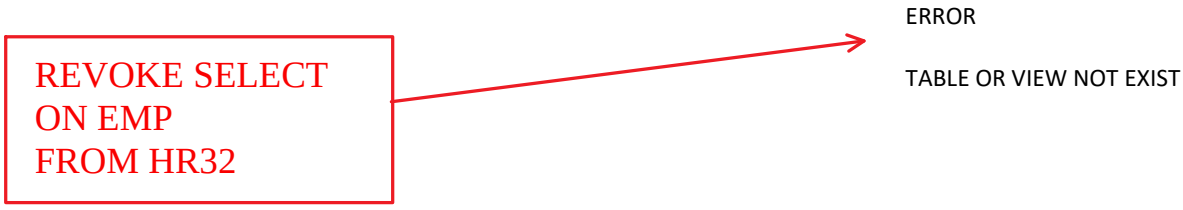
USER 2: HR

```
SELECT *
FROM EMP;
```

```
GRANT SELECT
ON EMP
TO HR
```

```
SELECT *
FROM SCOTT.EMP;
```





SQL> SHOW USER ;

USER is "SCOTT" SQL> CONNECT
Enter user-name: HR
Enter password: *****
Connected.

SQL> SHOW USER ; USER is "HR"

Connected.
SQL> SHOW USER ; USER is "HR"
SQL> SELECT *
FROM SCOTT.EMP;

FROM SCOTT.EMP
*
ERROR at line 2:
ORA-00942: table or view does not exist

SQL> CONNECT
Enter user-name: SCOTT Enter password: ***** Connected.
SQL> GRANT SELECT ON EMP TO HR;

Grant succeeded.

SQL> CONNECT
Enter user-name: HR
Enter password: *****

Connected.

SQL> SELECT *
2 FROM SCOTT.EMP;

DAY 24

14 July 2021 05:50 PM

ATTRIBUTES & FUNCTIONAL DEPENDENCIES

ATTRIBUTES : Are the properties which defines the entity .

Key attribute / Candidate key : An attribute which is used to identify a record uniquely from a table is known as key attribute .

Ex: *Phone_No , mail_id , aadhar , pan , ration , passport , dl , bank a/c*

Non key attribute : All the attributes other than key attributes .

Ex : *Name , age , gender , dob*

Prime key attribute : Among the key attributes an attribute is chosen to be the main attribute to identify a record uniquely from the table is known as prime key attribute .

Ex: *Phone_No.*

Non-prime key attribute : All the key attributes other than Prime key attributes

Ex : *mail_id , Aadhar , pan , ration , passport , dl , bank a/c* .

Composite key attribute : It is combination of two or more *non key attributes* which is used to identify a record uniquely from the table .

- Composite key is found whenever there is no key attribute .

Ex: *(name , age , dob , address)*

Super key attribute : It is a set of all key attributes .

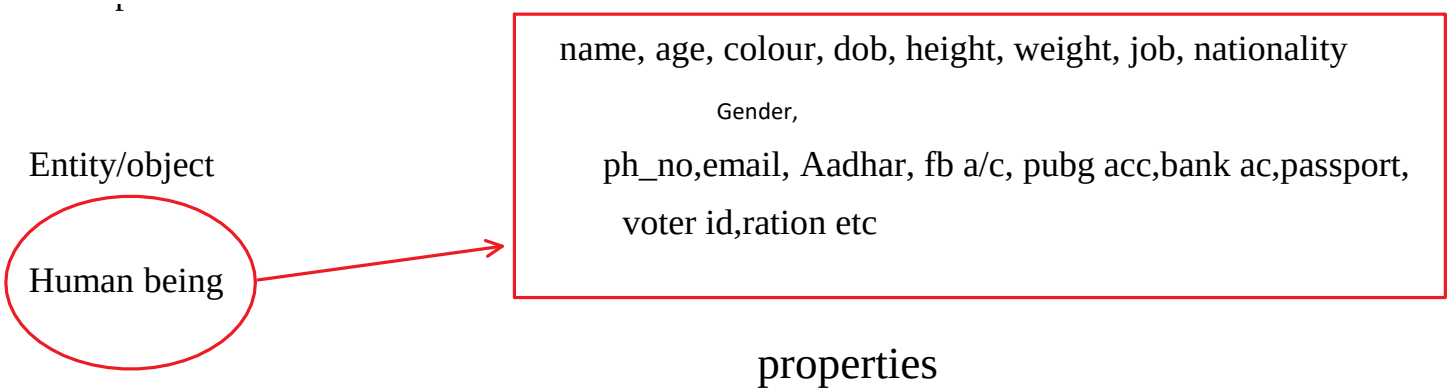
Ex: *{Phone_No , mail_id , Aadhar , pan , ration , passport , dl , bank a/c }*

Foreign key attribute : It is an attributes which behaves as an attribute of another entity to represent the relationship .

Ex: deptno

example:

name, age, colour, dob, height, weight, job, nationality



FUNCTIONAL DEPENDENCY :

"THERE EXISTS A DEPENECY SUCH THAT AN ATTRIBUTE IN A RELATION DETERMINES ANOTHER ATTRIBUTE ".

Example : **EMP** - (EID , ENAME)

EID ----> ENAME : *functional dependency* .

TYPES OF FUNCTIONAL DEPENDENCIES :

1. TOTAL FUNCTIONAL DEPENDENCY
2. PARTIAL FUNCTIONAL DEPENDENCY
3. TRANSITIVE FUNCTIONAL DEPENDENCY

1. TOTAL FUNCTIONAL DEPENDENCY:

If an attribute in a relation determines all the other attributes it is known as TFD

OR

If all the attributes are dependent on a single attribute then it is known as TFD

EMP - (EID , ENAME , SAL , DOB)

EID—KEY ATTRIBUTE

EID - > ENAME

EID - > SAL

EID - > DOB

:- **EID** ---> (ENAME , SAL , DOB) :- *total functional dependency.*

2. PARTIAL FUNCTIONAL DEPENDENCY:

There exists a dependency such that a part of composite key attributes determines another attribute uniquely .

CUSTOMER - (CNAME , ADDRESS , MAIL_ID , PHONE_NO)

Customer

CNAME	ADDRESS	MAIL_ID	PHONE_NO
Smith	Mysore	smith@gmail.com	
Miller	Bangalore		1001
Scott	Mangalore	scott@yahoo.com	
Adams	Mysore		2002
Scott	Delhi	scott@yahoo.com	3003

(**PHONE_NO , MAIL_ID**) ---- *Composite key attribute*

PHONE_NO ---> CNAME , ADDRESS

MAIL_ID --- > CNAME , ADDRESS :- *partial functional dep .*

3. TRANSITIVE FUNCTIONAL DEPENDENCY

There exists a dependency such that an attribute is determined by a non-key attribute, which is intern determined by a key attribute .

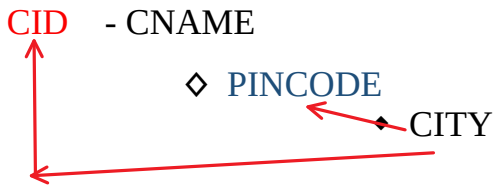
CUSTOMER - (CID , CNAME , PINCODE , CITY)

Customer

CID	CNAME	PINCODE	CITY
1	Smith	560019	Bangalore
2	Miller	560019	Bangalore

3	Scott	312121	Pune
4	Adams	123456	Mumbai
5	Scott	123456	Mumbai

KEY ATTRIBUTE NON KEY ATTRIBUTE



Redundancy : *The repetition of unwanted data is known as redundancy .*

Anomaly : *The side effects caused during DML operations is known as Anomaly.*

DAY 25

10 July 2021 03:04 PM

What is Normalization ?

" It is the process of reducing a large table into smaller tables in order to remove redundancies and anomalies by identifying their functional dependencies is known as Normalization . "

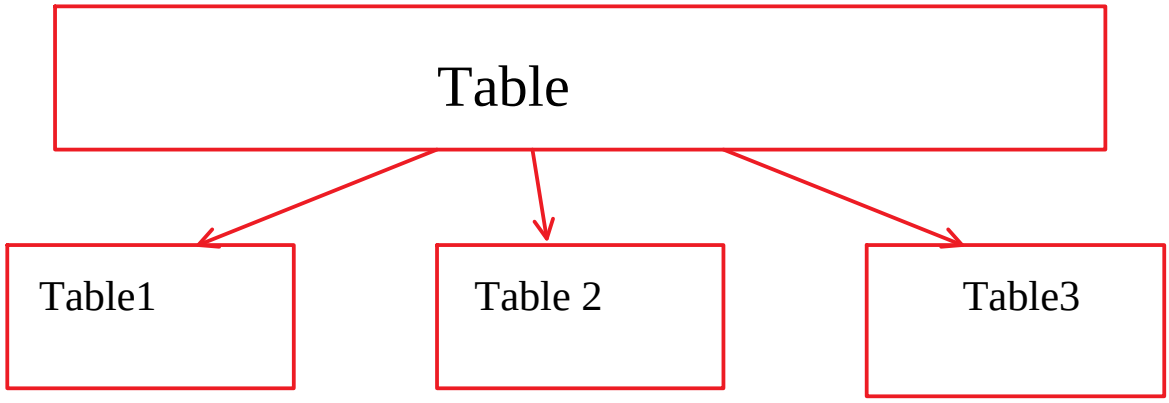
Or

"The process of decomposing a large table into smaller table is known as Normalization ."

Or

"Reducing a table to its Normal Form is known as Normalization . "

Example



What is **Normal Form** ?

A table without redundancies and anomalies are said to be in Normal Form .

Levels of Normal From .

1. **First Normal Form (1NF)**
2. **Second Normal Form (2NF)**
3. **Third Normal Form (3NF)**
4. **Boyce - Codd Normal Form (BCNF 3.5NF)**
5. **Fifth normal form (5F)**
6. **Sixth normal form (6F)**

Note : **If any Table / entity is reduced to 3NF , then the table is said to be normalized.**

1. **First Normal Form (1NF) :**

The table is said to be first normal form if it is satisfied following conditions

- No duplicates/repeated records .
- Multivalued data should not be present .

QSPIDERS

QID	NAME	COURSE
1	A	JAVA
2	B	JAVA , SQL
3	C	MT , SQL
1	A	MT

QID	NAME	C1	C2	C3
1	A	JAVA		MT
2	B	JAVA	SQL	
3	C		SQL	MT

1. **Second Normal Form (2NF)**

The table is said to be second normal form if it is satisfied following conditions

- Table should be in 1NF
- Table should not have Partial Functional Dependency .

EMPLOYEE - (EID , ENAME , SAL , DEPTNO , DNAME ,LOC)

EID	ENAME	SAL	DEPTNO	DNAME	LOC
1	A	100	10	D1	L1
2	B	120	20	D2	L2
3	C	320	10	D1	L1
4	D	251	10	D1	L1

EID- ename, sal
DEPTNO – dname, loc

(Eid, deptno)→(ename, sal, dname, loc) composite key attribute
Results in PFD

R1 – (eid, ename,sal)
R2 – (deptno, dname, loc)

Eid	**ename**	**sal**	**DNO**
1	A	100	10
2	B	120	20
3	C	320	10
4	D	251	10

Deptno	**dname**	**Loc**
10	D1	L1
20	D2	L2

1. **Third Normal Form (3NF)**

The table is said to be third normal form if it is satisfied following conditions

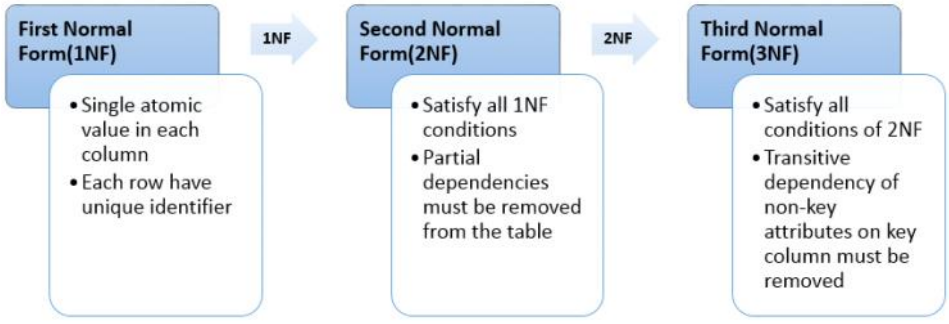
- Table should be in 2NF .
- Table should not have Transitive Functional Dependency .

Employee - (EID , Ename , Sal , comm , Pincode , state , country)

EID -> ENAME
SAL
COMM
PINCODE -> STATE
COUNTRY

R1 – (eid, ename, comm)
R2 – (pincode, state, country)

Note



EXTRA (DAY1)

24 July 2021 08:52 PM

VIEW

"In SQL view is a virtual table based on the result set of an SQL statement"

SYNTAX

```
CREATE VIEW VIEW_NAME
AS
SELECT COLUMN _NAME
FROM TABLE_NAME
WHERE <CONDTION>
```

SYNATX 2

```
DROP VIEW VIEW_NAME;
```

NOTE:

To give permission to create view !!

LOGIN AS SYSTEM

CONNECT ;

USER : SYSTEM
PWD: Oracle1234

GRANT CREATE VIEW TO SCOTT; (GIVING THE PERMISSION)

REVOKE CREATE VIEW FROM SCOTT; (TAKE BACK PERMISSON)

EXAMPLE :

EMPNO	ENAME	SAL	JOB
1	A	2000	MANAGER
2	B	1000	CLERK
3	C	5000	MANGER
4	D	1000	SLAESMAN

DAY 2

24 July 2021 08:52 PM

COMMENTS

Comments are used to explain the sections of SQL statements.

It has 2 comments

1) SINGLE LINE COMMENTS
2) MULTI LINE COMMENTS

1)SINGLE LINE COMMENTS:

SELECT *
FROM EMP; __ IT DISPLAYS ALL THE DETAILS

comments starts and end within single line are called as single line comments (individual line comments).

Single line comments start with -- and [ends with :]

__WAQTD ALL THE DETAILS OF THE EMP

SELECT *
FROM EMP;

syntax

```
-- single line comment
   select column name/ expression
   from table name
```

Example:

```
--select all the details of the employees :
  SELECT * FROM EMP;
```

2) MULTI LINES COMMETS:

Comments starts with first line but ends with different line are called multi lines comments

Multi line comments starts with /* and ends with */

syntax

```
/*multi
  line
  comments */
  select column name/ expression
  from table name
```

Example:

```
/*select all the
  details of the
  employees : */
  SELECT * FROM EMP;
```

DAY 3

24 July 2021 08:52 PM

SET OPERATOR

1. UNION
2. UNION ALL
3. INTERSECT
4. MINUS

UNION

The union operator could to be used to find the result set or combination of two/more table without returning any duplicate rows

Note :

1. The columns must have same datatypes.
2. The columns in each table must be in the same order (but they need not have to in the same length).
3. Each table used within union must have the same number of columns.
4. It provides the unique values by default.

Syntax:

```
SELECT COL_NAME
FROM TABLE _NAME1
[WHERE<FILTER CODITION>]

UNION

SELECT COL_NAME
FROM TABLE_NAME2
[WHERE<FILTER CODITION>];
```

Order of execution

1. From
2. [Where] (if used) [ROW BY ROW]
3. Select

UNION ALL

The union all operator could to be used to find the result set or combination of two/more table including returning any duplicate rows

Note :

1. The columns must have same datatypes.
2. The columns in each table must be in the same order (but they need not have to in the same length).
3. Each table used within union must have the same number of columns.

Syntax:

```
SELECT COL_NAME
FROM TABLE _NAME1
[WHERE<FILTER CODITION>]

UNION ALL

SELECT COL_NAME
FROM TABLE_NAME2
[WHERE<FILTER CODITION>]
```

Order of execution

1. From
2. [Where] (if used) [ROW BY ROW]
3. Select

DAY 3

22 August 2021 11:43 AM

INTERSECT

Intersect operator is used to combine two select statements ,it returns only common value [equal value] present in the table[column_name].

SYNTAX:

```
SELECT COLUMN_NAME
FROM TABLE_NAME
[WHERE <CONDITION>]

INTERSECT

SELECT COLUMN_NAME
FROM TABLE_NAME
[WHERE <CONDITION>];
```

ORDER OF EXECUTION:

1)FROM
2)[WHERE]
3)SELECT

Note :

1. The columns must have same datatypes.
2. The columns in each table must be in the same order (but they need not have to in the same length).
3. Each table used within intersect must have the same number of columns.

MINUS

MINUS operator is used to compare to select statements, it returns only the row that are present in the first statement , but are not in second statement ,[distinct value]. MINUS elements the duplicate rows.

SYNTAX:

```
SELECT COLUMN_NAME
FROM TABLE_NAME1
[WHERE <CONDITION>]

MINUS

SELECT COLUMN_NAME
FROM TABLE_NAME2
[WHERE  <CONDITION>];
```

ORDER OF EXECUTION:

1)FROM

2)[WHERE]
3)SELECT

Note :

1. The columns must have same datatypes.
2. The columns in each table must be in the same order (but they need not have to in the same length).
3. Each table used within minus must have the same number of columns.

DAY 4

22 August 2021 11:23 AM

Escape character
Escape character is used to remove the special behavior of the special character and treated as normal character (character).

Note:
1)Escape character must be defined.
2)Escape character must be written before the special character which has to be treated as normal character.
3)The recommended character for the escape character

'!, $, /, \' etc.

Syntax;

```
COLUMN_NAME/EXPRESSION LIKE 'PATTERN' ESCAPE 'ESCAPE_CHAR' ;
```

EXAMPLE:
1)WAQTD names of the employee if the names starts with an underscore.

```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE '%!_%' ESCAPE '!';
```

2)WAQTD name of the employee having % as the first character and underscore has the 4th character.

```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE '% ! %_ _ _%' ESCAPE '!' ;
```

3)WAQTD names of the emp if the emp name ends with % or _ .

```
SELECT ENAME
FROM EMP
WHERE ENAME LIKE '% ! %' OR ENAME LIKE '% ! _'  ESCAPE  '!' ;
```

DAY 5

12 September 2021 11:24 AM

PSEUDO COLUMN

Pseudo columns are the false column that are present in each and every table and must be called explicitly.

Pseudo columns cannot be seen without calling them.

We have 2 types:

1) ROWID
2) ROWNUM.

1) ROWID

ROWID IS AN 18 DIGIT ADDRESS IN WHICH THE RECORD IS PRESENT (OR) THE RECORD IS STORED IN THE MEMORY (VIRTUAL MEMEORY).

Note:

1) ROWID is one of the way to access or delete the record
2) ROWID is unique.
3) ROWID is present for each and every record .
4) ROWID is generated at the time of insertion of records.
5) ROWID cannot be inserted, updated or deleted .
6) Empty table will not be having ROWID
7) ROWID is static in nature .
8) ROWID can be used to identify a record uniquely from the table when there is no key attribute or primary key

SYNTAX:

TO FETCH TOP RECORD

```
SELECT */COLUMN_NAME/EXPRESSION
FROM EMP E1
WHERE ( SELECT COUNT([ DISTINCT] ROWID )
          FROM EMP E2
        WHERE E1.ROWID >= E2.ROWID) = N ;
```

TO FETCH BOTTOM RECORD

```
SELECT */COLUMN_NAME/EXPRESSION
FROM EMP E1
WHERE ( SELECT COUNT([ DISTINCT] ROWID )
          FROM EMP E2
        WHERE E1.ROWID< =E2.ROWID) = N ;
```

```
FROM EMP E2
WHERE E1.ROWID< =E2.ROWID) = N ;
```

EXAMPLE:

```
SELECT ROWID,EMP.*
FROM EMP;
```

ASSIGNMENT ON ROWID:

- WAQTD FETCH THE FIRST RECORD FROM THE EMP TABLE
- WAQTD FETCH THE 2ND RECORD FROM THE EMP TABLE
- WAQTD FETCH THE 4TH,5TH,8TH RECORD FROM THE EMP TABLE
- WAQTD FETCH THE LAST RECORD FROM THE EMP TABLE
- WAQTD FETCH THE SECOND LAST RECORD FROM THE EMP TABLE

2) *ROWNUM*

ROWNUM acts as serial number to the result table

Note:

1) ROWNUM is used as record number that is assigned to the result table
2) ROWNUM is dynamic in nature.
3) ROWNUM is generated at the time of execution.
4) ROWNUM always starts with 1.
5) ROWNUM cannot be duplicated .
6) ROWNUM get incremented after it is assigned .

Example:

1) WAQTD ROWNUM ALONG WITH ALL THE DETAILS OF EMP.

```
SELECT ROWNUM, EMP.*
FROM EMP;
```

2) WAQTD FIRST 3 RECORDS.

EMP

ENAME	SAL
A	2000
B	1000
C	3000
D	2500
E	4000

```
SELECT *
```

```
FROM EMP
WHERE ROWNUM<4;
```

TO MAKE ROWNUM AS STATIC:

1) Take a table and assign ROWNUM to a given table
2) Change the ROWNUM to any other name by using ALIAS .
3) Use this as a subquery in form clause of outer query
4) In the outer query use the alias name in the condition

Example:

1) WAQTD THIRD 3 RECORD

EMP

ENAME	SAL
A	2000
B	1000
C	3000
D	2500
E	4000

```
SELECT ROUNUM SLNO,EMP.*
FROM EMP;
```

SL NO	ENAME	SAL
1	A	2000
2	B	1000
3	C	3000
4	D	2500
5	E	4000

```
SELECT *
FROM (SELECT ROWNUM SLNO,EMP.*
        FROM EMP)
WHERE SLNO=3;
```

EXAMPLE:

1) WAQTD THE FETCH FIRST RECORD FROM THE EMP TABLE
2) WAQTD THE FETCH 2ND AND 3RD RECORD FROM THE EMP TABLE
3) WAQTD THE FETCH 7TH RECORD FROM THE EMP TABLE
4) WAQTD THE FETCH FIRST FIVE RECORDS FROM THE EMP TABLE
5) WAQTD THE FETCH FIRST SEVEN RECORDS FROM THE EMP TABLE