Name :- Nagaraja. Shivappa. Aralikatti
BE :- CSE
Sub :- SQL
Mobile NO: 6361153251

SQL

INDEX

INDEX

SINO	CONCEPTS	MOCK CONDUCT
6	FUNCTIONS IN SQL (1) Built-In Functions (a) SRF (b) MRF (2) User-Defined functions	
7	CLAUSE STATEMENTS (a) select clause (b) from clause (c) where clause (d) groupby clause (e) Having clause (f) orderby clause.	
8	SUB QUERY (1) Types of sub query (a) SRF (b) MRF	
9	~~CO-RELATED SUB QUERY~~ JOIN'S	
10	CO-RELATION SUB QUERY	
11	PSUEDO CODE (a) ROWID (b) ROWNUM	
12	NORMALIZATION	Final mock

STRUCTRUED QUERY LANGUAGE

1 INTRODUCTION TO SQL :-

(R) SQL means Structured Query Language. it's Just language.

(R) SQL is used to communicate with the database.

(R) SQL is computer programming language. ~~to~~ SQL is use to Conversion of database.

" Structured Query Language is a computer programming language it's Just language it is used to communicate with database is called as Structured Query language."

1.1 Histroy of SQL :-

(R) SQL invented in 1970, in IBM (International Business Machine) company.

(R) The father of SQL is ~~"Boycee"~~ "RAYMOND BOYCEE".

(R) The co-father of SQL is (Edugar - Frank) "EF - Codd."

(R) ANSI (American National Standard Institute) This Institute calling ~~converted~~ the Name SEQUEL.

(R) After, ANSI is converted SEQUEL to SQL.

DATA :-

(*) Data : Data is a collection of information such as text, images, audio, video different type of data.

(*) Data is a raw facts which describes properties of an objects.

for example :

Pen → object
Pen name :- Bright
Pen color :- black
pen height :- 10 cm
pen width :- 5 cm

DATA BASE :-

(*) Data base is a large amount of data, we collected and store in a storage area with ascending order / in rows and columns wise is called as Data base.

(*) Data base is it is placed where we store the data in systemmatically / organized manner.

(*) Database we perform some operation like, Create, Read, update, delete CRUD operation.

(*) Database example : In a college the number of students studying the admission section they take all students information. all the information / Data store in a storage area in rows & columns. That is student Database.

(*) Students data base, hospital database, company database, Industrial database,

	col1	col2	col3	col4
Entity name →	STNO	S. NAME	S. GENDER	S. PHONO
1st record →	1	NSA	MALE	6361153251
2nd record →	2	NAVEEN	MALE	87[illegible]
3rd record →	3	KEERTHY	FEMALE	86000087
4th record →	4	KAVY	FEMALE	[illegible]

WHY WE USE SQL :-

(R) SQL is we to communicate with Database and perform some CRUD operations. ⟶ [CREATE, READ, UPDATE, DELETE]

(R) we can create a table in database.

(R) we can ~~read~~ Insert Data into a table in database.

(R) we can update Data in a table in the database.

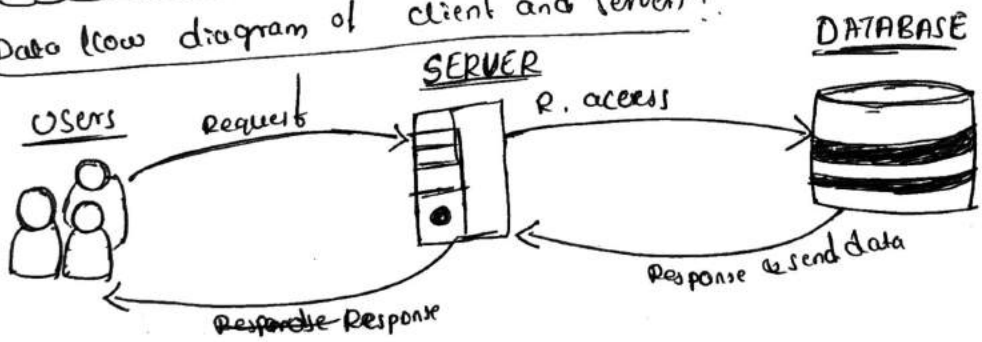
(R) we can Read Data from database.

(R) we can modify Data ~~from~~ in a table from database.

(R) we can delete records / data in a table from database.

How Users access data from server, and database :-

Data flow diagram of client and server :-



(R) User :-
(R) SERVER
(R) DATABASE

"Peoples / we are users, to use computer and searching information in a google search engine. when we enter address and give the search button, that is direct go to server and server will accept request from user and server have data then response the client. if the server doesn't have data. The server will access data from database and give the data to client This all about how users access data from server and database."

2 SQL: have two software

1) DBMS software.
2) RDBMS software.

1) DBMS (Data Base Management System) :-

(*) DBMS means Data Base Management System. DBMS is a software.

(*) DBMS is a software it is use to collect data and store data ~~in a storage area~~ and maintain data is called as DBMS.

(*) Data Base Management System what we collecting data, that data store in byte formate 0's | 1's in a memory location.

(*) DBMS are Data can communicate with any Query Languages.

(*) DBMS Data only single person can access Data and perform the CRUD Operation.

(*) DBMS have two keys 1) Authorization 2) security.

1) Authorization :- authorization is one of the key of DBMS software. It is used to access Data only authorized persons. Who know their username and password can access data.

2) security :- security the name indicates protection of Database. and safe data unauthorized can't access data.

(*) DBMS software work flow diagram.

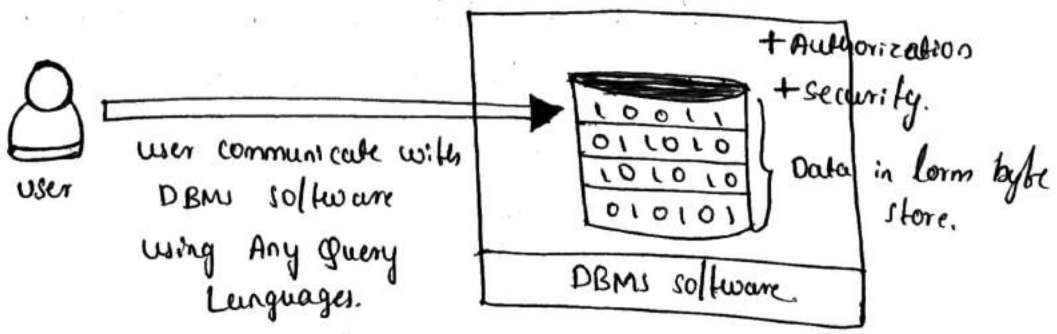


fig: DBMS software.

2] RDBMS (Relational Data Base Management System) :-

(*) RDBMS It's stands for Relational Data Base Management System.

(*) "RDBMS is a software, it is used to collection of information to and store data, and maintain data, with relation (realted Database we are collecting and store data in form of rows and columns is called as RDBMS."

(*) RDBMS also have two keys 1] Authorization 2] security.

1] Authorization :- whom know the usernames and password only that Persons can access data that are called authorized Persons.

2] security :- protecting database maintain database.

(*) RDBMS software can communicate with multiple users. with SQL Language only.

(*) RDBMS & Data can store in a Table wise.

(*) RDBMS is easy to use software and process is easy.

(*) RDBMS software work flow diagram bellow.

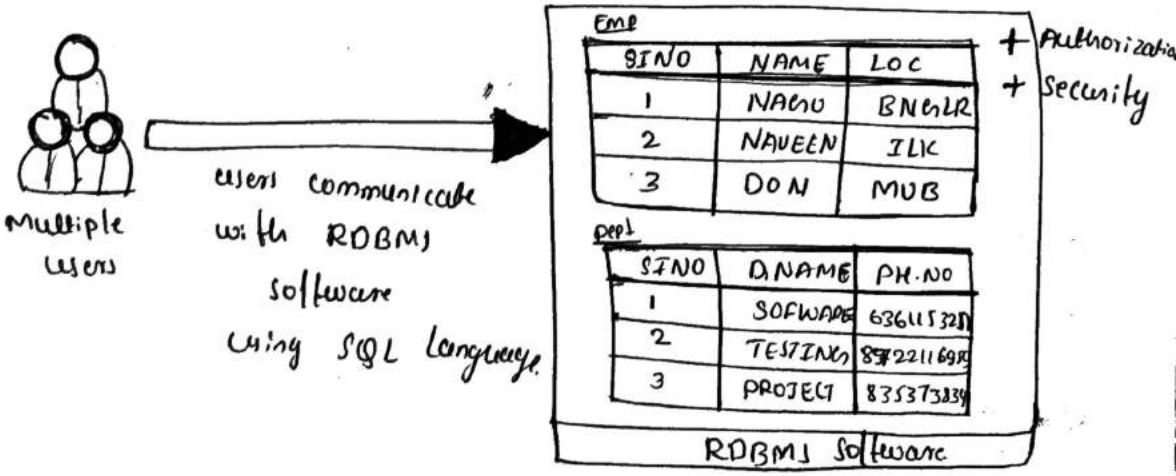


Fig: RDBMS software.

Difference between DBMS and RDBMS software :-

DBMS	RDMS
✓ DBMS is Data Base Management System.	✓ RDBMS is Relational Data Base Management System.
✓ DBMS is a software it is used to collecting data, store data & main tain data.	✓ RDBMS is a software it is used to collection of information, and store data and main tain data with redational data can store it.
✓ DBMS is store data in form of byte.	✓ RDBMS Data can store in Table wise (rows and columns.
✓ DBMS can communicate with single person and any Query Languages.	✓ RDBMS can communicate with multiple users with SQL Language. only.
✓ DBMS have two keys 1) Authori-zation and 2) security.	✓ RDBMS have also two key 1) Authori-zation and 2) security.
✓ DBMS software Draw backs :-	✓ RDBMS software Draw backs :-
✓ DBMS software Data modification is possible but difficult.	✓ RDBMS software the Data modification is easy.
✓ single persons can access data.	✓ multiple users can Access Data.
✓ very less amount of data.	✓ very large Amount of Data we store.
✓ DBMS software execution speed is less.	✓ RDBMS software execution speed good.
✓ small scale industries can use DBMS software.	✓ Large Industries can use - RDBMS software.
✓ Normalization not possible	✓ Normalization is Possible.

[3] TYPE'S OF VALIDATIONS

(R) validation means it is a process of selecting correct data and store Data.

(R) validation is valid data we are using and perform some operations.

There are two types of validation. that is following bellow :-

1] DATA TYPES

2] CONSTRAINS

1] DATA TYPE'S :- Data type is a one of validation type. and Data type is a process of what type of data it identifies is called Data type.

There are 5 data types is there :-

1] CHAR

2] VARCHAR / VARCHAR2

3] NUMBER

4] DATE

5] LOB (Large objects)

(R) Character Large object (CLOB).

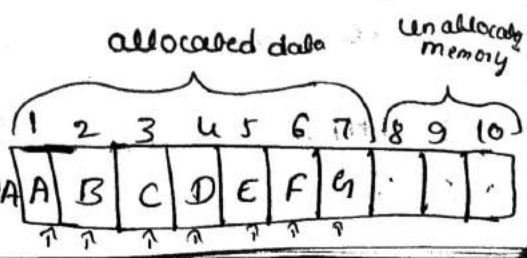
(R) Binary Large object (BLOB).

1] CHAR :- CHAR is a one of the Data type. CHAR Data type is used to store characters Data. The characters type of Data like ' ', 'A - Z', 'a-z', '0-9' and special characters also store data.

(!, @, #, (), $).

SYNTAX :- Data type [size];

ex:- CHAR [10];



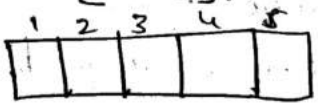
(R) CHAR Data type is store characters Data ~~its~~ it is a FLMA ~~fixed Memory~~ Fixed Length Memory Allocation.

Whenever we store characters Data in a memory it's take FLMA fixed Length Memory Allocation.

(K) The size of 10 it's create 10 characters Digit space, we store only 7 characters in a memory That is allocated memory remaining space 3 digit that is unallocated memory we can't use that space, because, The char data type only we store data it's Fixed Length memory allocation. we can't Change the memory ~~location~~ space / reused the memory space.

SIZE:- size it's indicates what, / How much Character you will store in a memory that digits of characters you will specify in the SIZE section.

[SIZE].

CHAR [5] ⟶ 

(K) it's create 5 digit characters size will create in memory location.

(R) CHAR is simple data type, But once data we insert into memory that is Fixed length memory Allocation we can't change it and we can't reuse it.

(R) CHAR data type... the characters data store Range is 2000

(A) Max 2000 characters we store.

2] VARCHAR / VARCHAR2 :-

(R) VARCHAR | VARCHAR2 it is one of the Data type.

(R) VARCHAR it is used to store Data. form of variable characters.

(R) VARCHAR Data store word / Data we can store using varchar Data type.

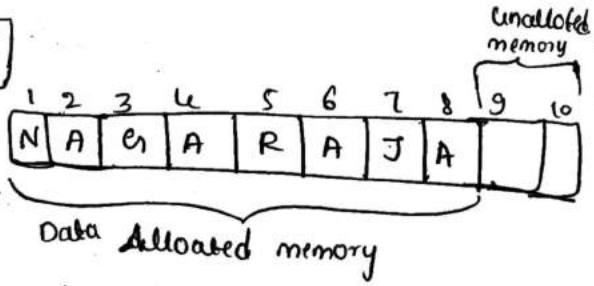
(R) VARCHAR we can store Data max 2000 variable characters.

(R) VARCHAR2 ~~it~~ it is one of the Data type and it's latest version, we this varchar2 data type along with number and characters also it's support it.

(R) VARCHAR2 we can store Data max 4000 variable characters.

SYNTAX :- DATA TYPE [SIZE];

Ex:- VARCHAR [10];



(R) VARCHAR / VARCHAR2 Data type store variable characters.

(R) VARCHAR data type [size] it is use to create space in a memory.

(R) VARCHAR [10]; this create 10 digit ~~memory~~ space in memory

(R) VARCHAR data type is VLMA Variable Length Memory Allocation. we insert data in a memory we can reuse the unallocted memory.

(R) VARCHAR is easy to use and store Data in a memory.

(R) VARCHAR is used to store variable Characters.

(R) we can reuse unalloated memory ~~allocation~~ in VARCHAR data type.

3) NUMBER :-

(*) NUMBER is a Data type. it is used to store Number type of Data / Integer Numbers.

(*) NUMBER it's store Numerical values. and it's have two Arguments.

1] PRECISION
2] SCALE

SYNTAX :- DATATYPE (PRECISION , [SCALE]); → optional.

ex:- NUMBER(5) ; // integer value
NUMBER(5.2); // decimal values.

1] PRECISION :- Precision is used to determined the Number of digit ~~Integer value~~ store integer values.

(*) The Range of Precision is 1 to 38

2] SCALE :- Scale is used to determined the Number of digit store Decimal point values.

(*) The default value of scale is '0'.

(*) The Range of scale is -84 to 127

for more examples :-

(1) NUMBER(7); // ± 1111111
7 digit Enter values. type of integers.

(2) NUMBER(5); // ±12346

(3) NUMBER(3.5); // ± 000999

(4) NUMBER(5,-3); // 55555
↓ ↓ ↓
0 0 0
55,000

(5) NUMBER(6.2); 1111.11

4] DATE :-

(R) DATE is a one of the Data type.

(N) DATE is used to store Data in form of Date like Day-month-and year

(R) The DATE have two formates that is following bellow.

Format :-

1] 'DD-MM-YY' → '01-JAN-2021'

2] 'DD-MM-YY' → '01-JAN-21'

SYNTAX :-

DATE → DATE

5] LOB : Large objects :-

(P) The Name indicates large objects. It's have two type.

1] CHARACTER LARGE OBJECT (CLOB) :- character Large object it is used to store Large Amount of character objects. and CLOB data store upto 4GB size.

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

for example :- we can store large object of character in ~~Dictionary~~ Dictionary it's start from Alphabets.

2] BINARY LARGE OBJECT (BLOB) :-

Binary Large object it is used to store Binary Data a large. Amount of Binary Data we can store. The BLOB data we can store upto 4GB size.

SYNTAX :- [{BLOB | BINARY LARGE OBJECT} | LENGTH | {K|M|G}}]

for example :- Binary Data pdf, Documents, image, audio, video,

2] CONSTRAINS :-

(P) Constrain is one of the type of validation.

(R) Constrain means specify some values. / It is a rule given to column for validation.

There are five type of constrains. :-

1] UNIQUE ——→ content value
2] NOT NULL ——→ Empty and No empty value
3] CHECK ——→ Check the status of column
4] PRIMARY KEY ——→ Unique and it column is a P.K.
5] FOREIGN KEY ——→ Linking of one / more tables.

1] UNIQUE :- Unique is one of the type of constrain. Unique the name. Itself it used to avoid duplicate value in the column. It's Checks the unique value in the column.

ex:- `UNIQUE (NAME (NAGARAJA));`

2] NOT NULL :- It is used to avoid NULL value.

(1) NULL means → Empty.
(2) NOT NULL means → No empty.

3] CHECK :- It is an extra validation with condition if the condition is satified. then the value is accepted else Rejected.
Check it is used to check the status of column like phone, `CHECK (LENGTH (PHNO));`

4] PRIMARY KEY :- primary key is a unique key it is used to access all column value if first column is a primary key. each and every table only one primary key have.

5] FOREIGN KEY :- Foreign key is a key it used to connection established of one or more table many foreign key in the tables.

[4] SQL STATEMENT'S :-

There are five SQL statements. those all following bellows.

Overview of SQL statements :-

1] DDL (Data Definition Language)

2] DML (Data Manipulation Language)

3] TCL (Transcation control language)

4] DCL (Data control language)

5] DQL (Data query language)

1] DDL (Data Definition Language) :-

(1) DDL stands for Data Definition language. it is one of the SQL statement.

(2) DDL the name it indicates Data Definition Language. the Data will Defining and perform some operations in this DDL.

(3) DDL is used to construct an object in the database and deals with structure of the objects.

Its have five statements / commands :-

1] CREATE

2] RENAME

3] ALTER

4] TRUNCATE

5] DROP

1) CREATE :- Create is a one of the DDL stmt. create : used to creating tables by using create stmt. / command

SYNTAX :-

CREATE TABLE TABLENAME (COLUMN1_NAME DATATYPE (SIZE). PRIMARY KEY, COLUMN2_NAME DATA TYPE(SIZE), COLUMN3_NAME DATA TYPE (SIZE), ...);

OR

```
CREATE TABLE TABLE NAME (COLUMN_NAME1 DATA TYPE(SIZE) PRIMARY
                         COLUMN_NAME2 DATATYPE(SIZE),
                         COLUMN_NAME3 DATATYPE (SIZE),
                         COLUMN_NAME4 DATA TYPE (SIZE),
                              .              .
                              .              .
                         COLUMN_NAMEN DATATYPEN (SIZE));
```

For example :- student table :-

```
CREATE TABLE STUDENT (SINO NUMBER(10). PRIMARY KEY,
                      S_NAME VARCHAR(15),
                      S_GENDER VARCHAR (10),
                      S_PHONO NUMBER (10),
                      S-ADDRES VARCHAR (20));
```

(*) after writhes this, we will get like bellow:

Table created.

(*) To view table created / Not check using desc operator.

```
DESC TABLE TABLENAME;
DESC STUDENT;
```

(*) It describes the student table and show column names along with constraint columns values.

2] RENAME :- RENAME is one of the DDL stmt. it is use to change the column name / table name, reassign the name, rename.

(1) RENAME command is used to change the name of the table or database object.

(2) RENAME stmt it renaming the column name by using ALTER command.

SYNTAX :-

```
ALTER TABLE TABLENAME RENAME OLDCOL_NAME TO
                                NEW COL_NAME ;
```

3] ALTER :- ALTER is one of the DDL stmt it is use to Alter the values. Add column, rename column, modify column, delete column.

SYNTAX :- add column to table using ADD command.

ex:
```
ALTER TABLE TABLENAME ADD ~~COLUMN_NAME~~
                          NEW COL-NAME ;
```

ex:
```
ALTER TABLE STUDENT ADD EMAID. ;
```

4] TRUNCATE :- It remove all the records in table, except column names
TRUNCATE it delete the table permentaly.

SYNTAX :
```
TRUNCATE TABLE TABLENAME;
```
```
TRUNCATE TABLE STUDENT ;
```

5] DROP :- DROP it is used to delete the tables. it not delete table permentaly it in recycle bin
we delete table permenently by we will use PURGE command.

SYNTAX :
```
DROP TABLE TABLENAME ;
```
ex:
```
DROP TABLE STUDENT ;
```

2] DML (Data Manipulation Language)

(*) DML stands for Data Manipulation Language. It is one of the SQL stmt.

(*) DML is used to manipulated the object by performing Insertion, updating, and Deletion.

It's have Three commands :-

1] INSERT
2] UPDATE
3] DELETE

1] **INSERT** :- INSERT is a one of the DML stmt. / command. INSERT it is used to Inserting the values in to a tables / create records in the table.

SYNTAX :-

```
INSERT INTO TABLE NAME VALUES('VALUE1', 'VALUE2'
                                'VALUE3', ...);
```

ex:-

```
INSERT INTO STUDENT VALUES(1, 'NAGU', 'MALE', 63611
                            -53251, 'ILKAL');
```

1 row Inserted.

2] UPDATE :- UPDATE is one of the DML stmt / command.

UPDATE it is used to modifying an existing values. and changes the values.

SYNTAX :

```
UPDATE TABLE NAME
SET Col_Name = value, Col_Name = value
WHERE stmt;
```

Ex :-

```
UPDATE STUDENT
SET S_PHONO = 8722116959, ADDRESS = 'ILKAL'
WHERE S_NAME = 'NAGU';
```

```
UPDATE STUDENT
SET S_NAME = 'NAGARAJA'
WHERE S_ID = 1;
```

3] DELETE :-

(1) DELETE is one of the DML stmt. / command.

(2) DELETE command it is used to remove particular records from table.

SYNTAX :

```
DELETE
FROM TABLE NAME
WHERE STMT;
```

```
DELETE
FROM STUDENT
WHERE S_ID = 1;
```

```
DELETE
FROM STUDENT
WHERE S_NAME = 'NAGU';
```

3] TCL (Transcation Control Language) :-

(*) TCL stand for Transcation control language it's one of the SQL stmt.

(*) TCL it is used perform commit, rollbacking, save point operations.

It's have three stmt/commands :-

1] COMMIT

2] ROLLBACK

3] SAVE POINT

1] COMMIT :-

(*) COMMIT it is used to save the database. whenever we insert some records into table after, insertion compelete, we just put commit, it's save the records what you inserted records into table.

(*) commit is a just saves, the database.

SYNTAX :- `COMMIT;`

2] ROLLBACK :-

(*) ROLLBACK it is getting back the data which we saved data we will get back.

(*) ROLLBACK whenever we commit, after we delete some records we want to backup record we used roll back command.

(*) without commit and delete the records we can't backup the data.

(*) first we commit after delete record, we get back data through rollback. SYNTAX : `ROLLBACK;`

3] SAVE POINT :-

- SAVE POINT is one of the TCL stmt / command.
- SAVE POINT is don't realted anything to the database
- SAVE POINT is remembering point where I stopped it. or
- SAVE POINT is Reloration Point.

  - remeber point
  - or
  - reloration point.

- Whenever, we will insert some records to database, after we will do commit. and after insert some more records and make commit it that time we don't know where I gave commit. it checks the which level I gave commit it. it remember. that's why we used SAVE POINT.

SYNTAX :-

```
SAVEPOINT SAVEPOINT_NAME;
```

ex:

```
SAVEPOINT RAJ;
```

4] DCL (Data Control Language) :-

(1) DCL stands for Data Control Language.

(2) DCL is one of the SQL stmt.

(3) "Data control language is used to control the flow of data between the users."

DCL have two stmt / commands :-

1] GRANT

2] REVOKE

1] GRANT :- It is a command it is used to give permission to a user.

SYNTAX :-

```
GRANT SQL_STMT
ON TABLENAME
TO USER_NAME;
```

2] REVOKE :- It is a command it is used to take back the permission from users.

SYNTAX :-

```
REVOKE SQL_STMT
ON TABLE_NAME
FROM USER_NAME;
```

5] DQL (Data Query Language) :-

① DQL stands for "Data query Language"

② DQL is used to retrieves the data from database.

It's have four statements :-

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

1] SELECT :- it is used to retrieves data from table and display it.

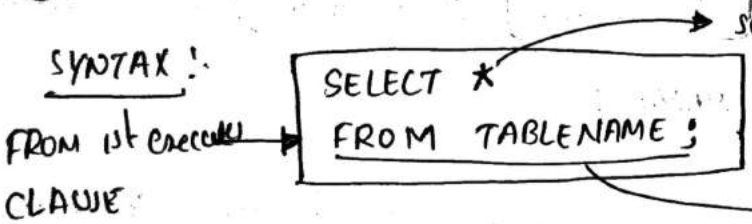
② SELECT stmt selecting the data from table and display it.

① SELECT have one Arugument → Asterisk (*)

② Asterisk (*) it means select all data from table.

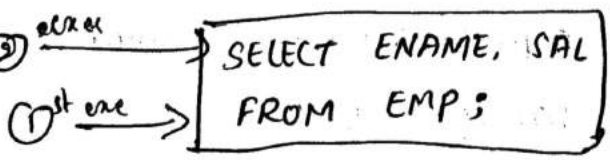
① semicolon (;) it means end / Terminates the query.

SYNTAX :-



② CLAUSE :- CLAUSE is a stmt it executes in order wise.

" first FROM CLAUSE will be execute that select table, after SELECT CLAUSE will get executes all data. "



2] PROJECTION :-

① "PROJECTION is a process of retrieving the data by selecting only the columns is known as PROJECTIONS."

3] SELECTION :-

② "SELECTION it is a process of retrieving data by selecting both columns and rows is known as a SELECTION."

4] JOIN :-

② "JOIN it is used to retrieving data from multiple tables simultaneously is called as JOIN."

SYNTAX :-

```
SELECT * / [DISTINCT] column-name/Expression [ALIAS]
FROM TABLE NAME ;
```

① column-name :- How to display column name by using select stmt.

```
SELECT ENAM, SAL,
FROM EMP;
```

(2nd ② , 1st ①)

ENAME	SAL
A	200
B	30
C	40

② DISTINCT :- DISTINCT stmt (command) it is used to avoid unwanted names or remove repeated values, is called as DISTINCT.

```
SELECT DISTINCT ENAME
FROM EMP;
```

(② , ① st or)

(R) ALIAS :- "ALIAS is stmt / command it is used to Rename the columns names."

SYNTAX :-

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

→ goto Emp table select all columns (*) and rename EMPID to SINO.

ORDER OF EXECUTION :-

1] FROM CLAUSE → will get executed.
2] SELECT CLAUSE → will get executed.

Example :-

1] Write a query to display of all the students names.

student

S_Id	S-Name	S_Phone_no
1	NAGARJA	6361153251
2	NAVEEN	9822116959
3	MANJU	9353787189
4	KING	8123976874

(2nd) (1st)

```
SELECT S-Name
FROM student;
```

O/P →

S-Name
NAGARAJA
NAVEEN
MANJU
KING

5 OPERATORS IN SQL

1] ARITHMATICS OPERATORS (+, -, *, /)

2] CONCATENATION OPERATOR (||)

3] COMPARATION OPERATOR (=, !=, or < >)

4] REALATIONAL OPERATOR (>, <, >=, <=)

5] LOGICAL OPERATOR (AND, OR, NOT)

6] SPECIAL OPERATORS

- (1) IN NOTIN
- (2) BETWEEN NOTBETWEEN
- (3) IS IS NOT
- (4) LIKE NOT LIKE

7] SUBQUERY OPERATORS

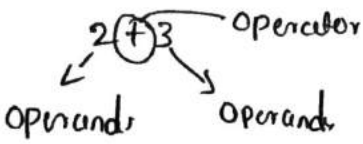
- (1) ALL.
- (2) ANY.
- (3) EXIST.
- (4) NOT EXIST.

1) ARITHMATICS OPERATORS :- (+, −, *, /) :-

(*) Arithamatics operator is one of the SQL operator.

(*) Arithamatics operators are Addition, subtraction, multiplication, Division This all operators are Arithamatics operator. This operator is we to perform some operation using SQL queries.

(*) Addition (+) :- Add two values we will get total values.



to add two values 2+3 = 5

(*) Using + operator we will get total values.

(*) In case SQL queries we want total Emp salary we use + operator.

for example :

WAQD: total/sum of all Employees salary.

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

// + we will get to sum of all Emp salary.

(*) subtraction :- deduction of Emp salary.

```
SELECT SAL-SAL, SAL
FROM EMP;
```

// − Deduction of Emp salary.

(*) Multiplication (*) :-

(*) Multiplication (*) operator is used to multiply by two values of double of values. for ex: 2*2=4 like.

how much Employe 1 year salary ~~how much~~ we used (*) *.

WAQD Display Employee 1 year salary.

~~SELECT SAL * SAL, SAL~~

```
SELECT SAL, SAL* 12/100 ;
FROM EMP ;
```

WAQD Display hall month Employe salary.

```
SELECT SAL, SAL * 6/100 ;
FROM EMP;
```

(*) Division :- Division operator :- is used to dividing of values or percentage of employee salary. we Division operator.

```
SELECT SAL, SAL * 10/100 :
FROM EMP;
```

→ 10% salary of employee

2] CONCATENATION OPERATOR (||) :-

(R) Concatenation operator is one of the SQL operator.

(R) concatenation means Join two values. or

(R) Concatenation is combination of two sections.

(R) concatenation it is used to Join strings

symbol : (||)

Example :-

```
SELECT ENAME
FROM EMP
WHERE JOB="MANAGER"
```

Ename
ALLEN
MARTIN
SMITH

→ concatenation operators Join Two strings.

```
SELECT 'Hi' || ename
FROM EMP
WHERE JOB='MANAGER';
```

O/P

Ename
Hi ALLEN
Hi MARTIN
Hi SMITH

3] COMPARISION OPERATOR :- (=, !=, or <>) 4] and Relational operator

(*) COMPARISION OPERATORS are equal (=), not equal (!=), less than (<), greater than (>). operators we used and write query.

1] WAQD all employee salary, and whose name is 'SMITH'.

(*) equal operator

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

(1) → FROM EMP
(2) → WHERE ENAME = 'SMITH';
= → equal operator.

(*) WHERE CLAUSE :- WHERE is a CLAUSE stmt, it is used to filter condition. in the query condition is there we will use WHERE CLAUSE stmt.

(*) Not equal operator

```
SELECT SAL
FROM EMP
WHERE ENAME != 'SMITH';
```

!= → not equal operator

(*) less than (<) :- less salary of employees (MINIMUM SAL)
WAQD ~~whose employee~~ employees salary and whose employees less salary:

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

< → Less sal

(*) greater than (>) :- greater salary of employees (MAXIMUM SAL),
WAQD employees salary whose salary is maximum.

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

> → greater sal.

(*) less than equal to (<=) :- less than equal to salary.

```
SELECT SAL
FROM EMP
WHERE SAL <= SAL;
```

// whose salary is less than equal to

(*) greater than equal to (>=) :- greater than equal to

```
SELECT SAL
FROM EMP
WHERE SAL >= SAL;
```

// whose salary is greater than equal to

(*) Equal Equal to (==) :-

```
SELECT SAL
FROM EMP
WHERE SAL == SAL;
```

// whose salary is equal equal to.

5] LOGICAL OPERATOR :-

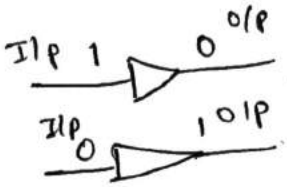
(1) Logical operators are like operators perform different tasks.

(2) Logical operator are NOT, AND, OR.

(3) NOT :- NOT is a logical operator. it is used to 1 input and 1 output

0'1 or 1'0

A	B
0	1
1	0



NOT operator it is used to check the Data is there are not in a Table. use NOT stmt.

1] WAQD employee names whose name 'MARTIN' not working Dept 10.

```
SELECT ENAME
FROM EMP
WHERE ~~ENAME~~ 'MARTIN' ~~AND~~ AND DEPT NOT IN(10);
```

or

WAQD employees whose not working in Dept 20.

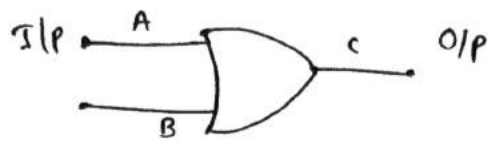
```
SELECT ENAME
FROM EMP
WHERE DEPT NOT IN (20);
```

NOT operator it is used to check either data available or not in a Table.

and NOT it indicates not available of data in a Table.

(1) AND :- AND is a logical operator it is used to perform two i/p's and one o/p Multiplication operations.

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1



(2) AND operator it is used to take one more condition in WHERE CLAUSE.

(3) In WHERE CLAUSE we use more AND operator to apply filter condition.

For example :

WAQD all employees name and salary, whose name is ALLEN and working in Dept no 10 and location is Newyork

```
SELECT ENAME, SAL
FROM EMP
WHERE ENAME='ALLEN' AND DEPTNO='10' AND
LOC='NEWYORK';
```

AND Operator.

(4) AND operator it is we to apply more filter condition in WHERE CLAUSE.

(K) OR operator :- OR operator it is used both condition check either one or two. (TRUE / FALSE).

(A) OR operator it's check condition first or second condition.

WAQD Employee name and salary whose taking Maximum salary.

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

it's checks both condition it's execute 1 or 2.

OR Operator :- Addition operation

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

" OR operator it is used to either 1 or 2nd executes. "

6] SPECIAL OPERATORS :-

- (1) IN NOT IN ⟶ it is used to IN inside the data and not in not inside the data
- (2) BETWEEN NOT BETWEEN ⟶ it checks salary ranges or Dates
- (3) IS IS NOT ⟶ this compare to NULL values & Not NULL
- (4) LIKE NOT LIKE ⟶ this use to pattern matching.

1] IN OPERATOR :- IN OPERATOR it is used to checks the data the data is available in a Table.

Example:

WAQD Employee name & salary whose working in Dept 30.

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

2] NOT IN operator :. NOT IN Operator it is used to checks the data, if the data not available, then use this operator.

WAQD Employee name & salary whose not working in Dept 20

```
SELECT ENAME, SAL
FROM EMP
WHERE DEPTNO NOT IN (20);
```

2] BETWEEN OPERATOR :-

(*) BETWEEN OPERATOR it is used to check ranges of Joining Dates and resigning dates in company.

(*) start to end ranges.

WAQD all employee names whose Hired date '~~01-JAN~~ 1970 to

Dates
[JAN - DEC

```
SELECT ENAME
FROM EMP
WHERE HIRE_DATE BETWEEN '01-JAN-1970'
  AND '31-DEC-1970';
```

(*) NOT BETWEEN OPERATOR :- This operator, it is used to not hiered within the Dates. NOT BETWEEN Operator - not Joining and resigning within Date, it ~~range~~ checks ranges.

WAQD all employee name whose Not Hired date 1980

```
SELECT ENAME
FROM EMP
WHERE HIRED_DATE NOT BETWEEN '01-JAN-198
  AND '31-DEC-1980';
```

3] IS OPERATOR :- IS OPERATOR it is used to compare NULL values. empty values. there is no data.

SYNTAX :

```
COLUMN NAME IS NULL;
```

Ex:

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

WAQD whose commission is NULL

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

(4) IS NOT :- IS NOT operator is comparing Not NULL values plus checks the data and NOT EMPTY. The Data is there.

SYNTAX :

```
COLUMN NAME IS NOT NULL;
```

WAQD whose NOt getting commission is Not NULL

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

4] LIKE OPERATOR :-

(R) LIKE OPERATOR it is used to pattern matching

(R) LIKE OPERATOR this matches character pattern matching it's have two parts / special character.

1] percentage (%) :- It is a special character, which is used to match any number of character, any number of time / number of character. (match character %A) %A%) A%)

2] Underscore (_) :- It is a special character which is used to match exactly one, but any character.

(___) after 3 space character.

(_) — after one space character

SYNTAX :

```
COLUMN-NAME / Exp LIKE 'Pattern';
```

Examples :

① WAQD Details of all employees whose name starts with S.

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

② WAQD Details of all employee whose name end with s

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

③ WAQD Details of all employee whose name repeated value in a name. s.

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

Note:

The name start with s → %S

The name end with s → S%

The name repeated with s → %S%S%

(4) NOT LIKE OPERATOR :- NOT LIKE OPERATOR is unmatches the pattern and it is also opposite work of LIKE OPERARTOR. if is unmatches the pattern.

(1) WAQD Details of all employee whose name not start with s.

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

(2) WAQD Details of all employee whose name not end with s.

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

(3) WAQD Details of all employee whose name not repeated with s.

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

(4) WAQD Details of all employee whose name is not in second digit start with s.

```
SELECT ENAME
FROM EMP
WHERE ENAME NOT LIKE '_%S';
```

7] SUB-QUERY OPERATORS :-

1] ALL

2] ANY

3] EXIST

4] NOT EXIST

1] ALL :- ALL is a sub-query operator it act like AND operator. it satisfied both condition All condition.

Example :

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

2] ANY :- ANY is a sub query operator it is used to either one/ two condition is true like OR operator.

Example :

```
SELECT ENAME
FROM EMP
WHERE DEPT NO ANY (SELECT DEPTNO
                   FROM DEPT
                   WHERE LOC = 'NEW YORK';
```

3] EXIST :- EXIST is a sub query operator it is used to check the data is available / exist data in a table to matches the data and display it.

for example:

```
SELECT ENAME, SAL
FROM EMP
WHERE DEPTNO EXIST ( SELECT DEPTNO
                     FROM DEPT
                     WHERE DNAME = 'ACCONTING');
```

4] NOT EXIST :- NOT EXIT is a sub query operator it is used to not available data in a table unmatched record it check

for example :

```
SELECT ENAME
FROM EMP
WHERE DEPTNO NOT EXIT ( SELECT DEPTNO
                        FROM DEPT
                        WHERE DEPTNO = '50';
```

6 FUNCTION'S IN SQL :-

"Functions are block of code or list of Instructions which are used to perform some specific Tasks."

Functions components :-

1] Function Name
2] Number of Arguments (no's - Inputs)
3] Return Type.

Function Types :-

(*) Function have two main types that is following bellow.

1] Built-In Function.
   (*) SRF (Single row function)
   (*) MRF (Multi row function).
2] User Defined Function.

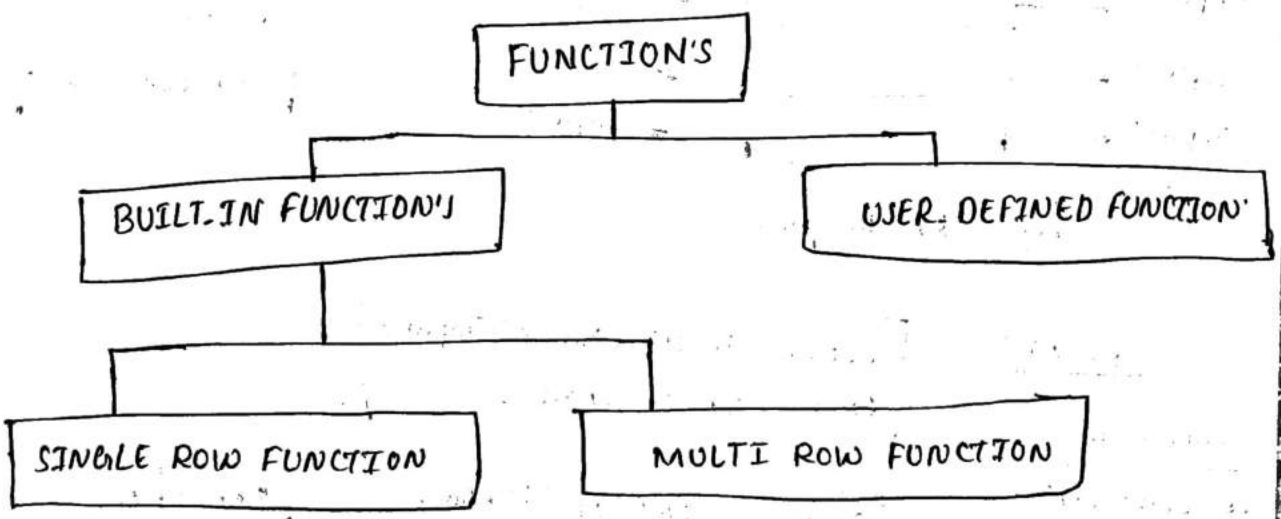


fig: Function Types flow diagram.

1) BUILT-IN FUNCTION :-

(N) Built-In function are one of the function type.

(R) Built-In function have two types.

- (R) Single row function (SRF)
- (R) Multi row function (MRF).

(*) SINGLE ROW FUNCTION (SRF) :-

(R) Single row function is one of the built-In function type.

(N) single row function if single row by row execute.

(R) It takes only one Input, one output, single line know execute.

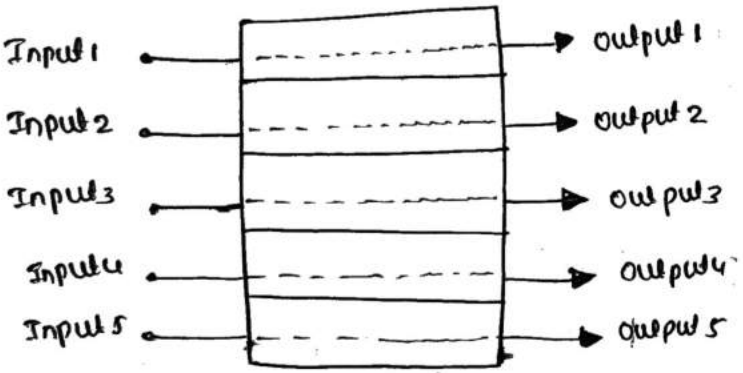


Fig: SRF Diagram

E2 : LENGTH()

ENAME	SAL
NAGARAJA	1,40,000
MANJU	20,000
KIRAN	30,000
NAVEEN	2,10,000

SYNTAX :

```
SELECT Function Name (Input)
FROM TABLE NAME ;
```

Functions

(R) LENGTH ✓
(R) MAX ✓
(R) MIN ✓
(R) COUNT ✓
(R) AVG ✓

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

ENAME	SAL
KIRAN	30,000

I/P → O/P

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

ENAME
6

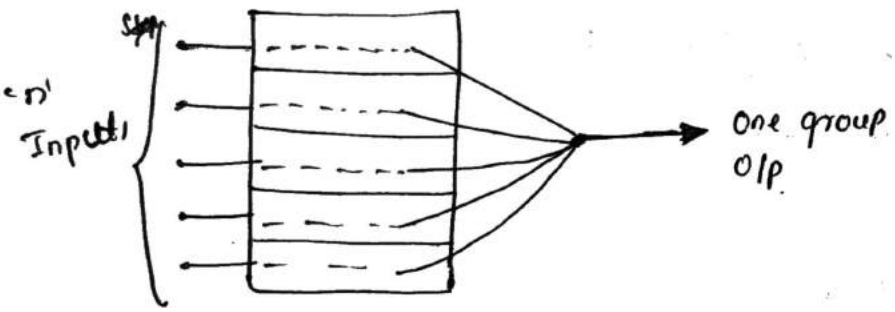
1] MULTI ROW FUNCTION (MRF) :-

(R) Multi row function is one of the built-in function type.

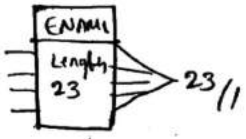
(R) Multi row function it execute group by group.

(R) Multi row function it takes all inputs at once, and then execute group by group give output.

(R) If we pass n number of inputs in MRF() it return 1 group output.



ENAME	SAL
NAGARAJA	1,80,000
MANJU	20,000
KIRAN	30,000
NAVEEN	2,80,000



SYNTAX :-

```
SELECT FUNCTION NAME (Inputs)
FROM   TABLE NAME
```

Functions

(R) LENGTH
(R) MAX
(R) MIN
(R) COUNT
(R) AVG

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

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

SAL
2,80,000 → 2,80,000

2] USER DEFINED FUNCTION :- User can predefined functions are user defined functions.

STRINGS :- How to print first 3 digits of strings using substring function();

(1) string is a stmt / collection of words.

(2) sub-string is a words. this words can make sub string.

SYNTAX : `substr('original string', 'p', 'L');`

Example : I/p: JSPIDER

O/p: JSP

```
substr('JSPIDER', 1, 7/2);
substr('JSPIDER', 1, 3);
```

output

JSP

Example:

I/p = POWER

O/p = POW

```
substr('POWER', 1, 5);
substr('POWER', 1, 5/2);
substr('POWER', 1, 3);
```

O/P

POW

[7] CLAUSE & STATEMENT'S :-

1] SELECT CLAUSE

2] FROM CLAUSE

3] WHERE CLAUSE

4] GROUPBY CLAUSE

5] HAVING CLAUSE

6] ORDERBY CLAUSE

1] SELECT CLAUSE :- it execute ~~row by row~~. group by group

(R) SELECT CLAUSE is a one stmt it is used to selecting data from data base / Table.

(R) SELECT → selecting process. data base

(R) SELECT * → selecting whole data, * (Asterisk) selecting all.

SYNTAX :-

```
SELECT *
FROM TABLE NAME;
```

```
2nd → SELECT *
1st → FROM EMP;
```

"it direct goto EMP Table select all data & display it."

2] FROM CLAUSE :- it execute ~~group by group~~. row by row.

(R) FROM CLAUSE is where we fetching the database Address of Table. we used FROM CLAUSE.

(R) FROM CLAUSE will get execute first.

SYNTAX :-

```
2nd → SELECT ENAME, SAL
1st → FROM EMP;
```

" it execute first from clause in From EMP Table it's direct goto EMP Table and select two column ENAME, SAL display it."

3] WHERE CLAUSE :-

(R) WHERE CLAUSE is a stmt, it is used to filter conditions.

(R) WHERE CLAUSE is write condition in this where clause stmt..

(R) WHERE CLAUSE stmt we will write one / more conditions.

(R) It executes row by row.

SYNTAX :-

```
SELECT COLUMN_NAME
FROM TABLE.NAME
WHERE Condition ;
```

Example :

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

2nd

1st

(R) first WHERE CLAUSE will get executes & it checks condition

(R) after, FROM CLAUSE will get executes.

Ex:

```
SELECT SAL
FROM EMP
WHERE ENAME = 'ALLEN';
```

2nd

1st

4] GROUPBYCLAUSE :- it executes row by row.

(K) GROUPBYCLAUSE is a stmt clause.

(K) GROUP BY CLAUSE is used to group by group the records.

SYNTAX :-

```
SELECT group_by expression / group function
FROM table name
WHERE <filter condition>
GROUP BY column_name / expression
```

Emp

E.ID	E.NAME	SAL	DEPT_NO
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

DEPT_NO
10
20
30

WAQD Number of employee working in each Dept.

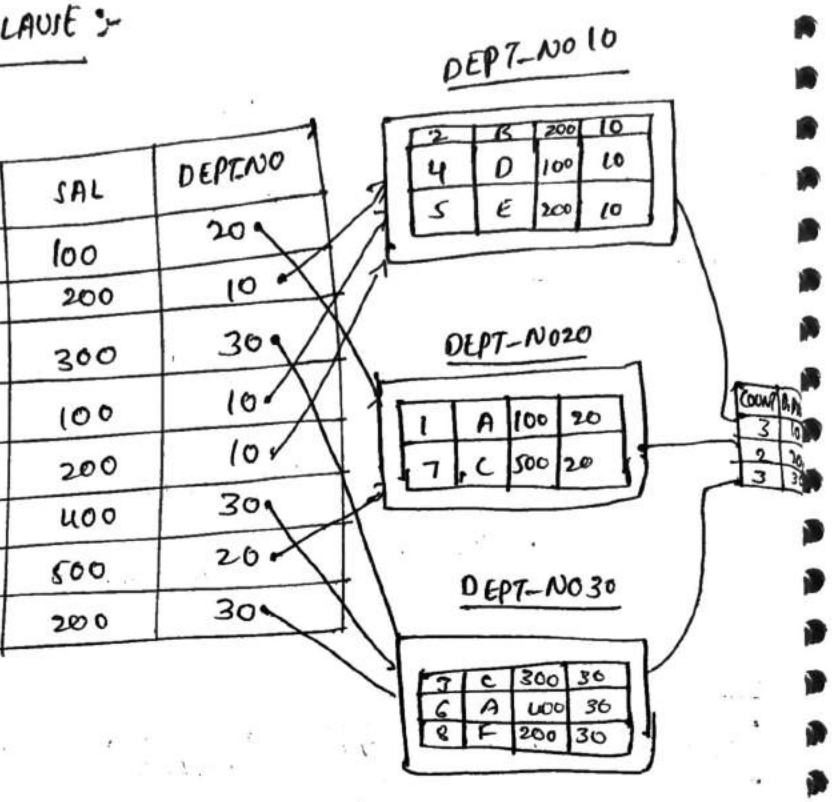
```
SELECT COUNT(*), DEPT_NO
FROM EMP
GROUP BY DEPT_NO;
```

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

OUTPUT of FROM CLAUSE :-

EMP

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



(1) Group by clause is used to execute group by group the records.

(2) Group by clause is execute row by row.

(3) After, execution of group by clause we will get groups.

(4) Group by clause execute row by row. records.

Queries on group by clause :-

1) WAQD Number of employees working in each Department ~~except~~ except the employee working as Analyst.

```
SELECT COUNT(*), DEPT_NO
FROM EMP
WHERE JOB = 'ANALYST'
GROUP BY DEPT_NO;
```

2] WAQD Max salary given to each Job —

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

3] WAQD Number of employees working in each Job if the employee have Character 'A' in their Names.

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

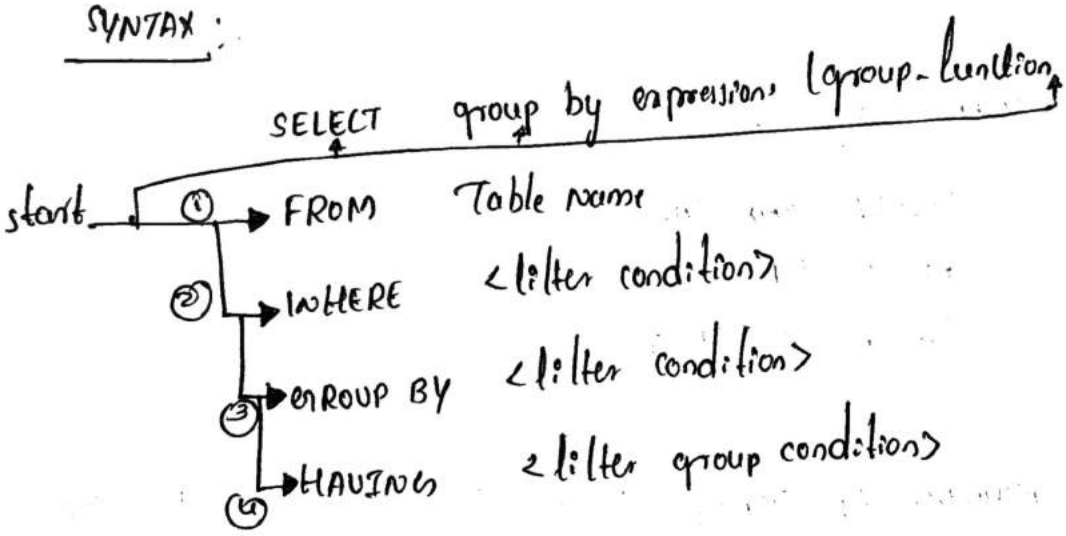
4] WAQD Number of Employees getting Commission in each department.

```
SELECT COUNT(*), DEPT_NO
FROM EMP
WHERE COMM IS NULL
GROUP BY DEPT_NO;
```

5] HAVING CLAUSE :-

(1) HAVING CLAUSE is a stmt of is used to filter the group.
(2) HAVING CLAUSE depends on group by clause.
(3) without groupby clause we can't write Having clause.
(4) but, without Having clause we will write group by clause.
(5) Having clause pf execute group by group.

SYNTAX :



HAVING CLAUSE SYNTAX :-

```
SELECT column name / group by function.
FROM Table Name
[WHERE <l_condition >
 GROUP BY <l- condition group>
 HAVING <l- condition group>
```

Query:

Write a query to display (WAQD) to find the Number of employees working in each dept. if there are at least 3 employees in each dept.

start →

```
SELECT COUNT(*), DEPT-NO
FROM EMP
GROUP BY DEPT-NO
HAVING COUNT(*)>=3;
```

OUTPUT:

EID	E.NAME	SAL	DEPT-NO
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	G	500	20
8	F	200	30

DEPT-NO 10

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

3>=3

DEPT-NO 20

1	A	100	20
7	G	500	20

2>=3

DEPT-NO 30

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

3>=3

COUNT(*)	DEPT-NO
3	10
3	30

6] ORDER BY CLAUSE :-

(1) ORDER BY CLAUSE is stmt.

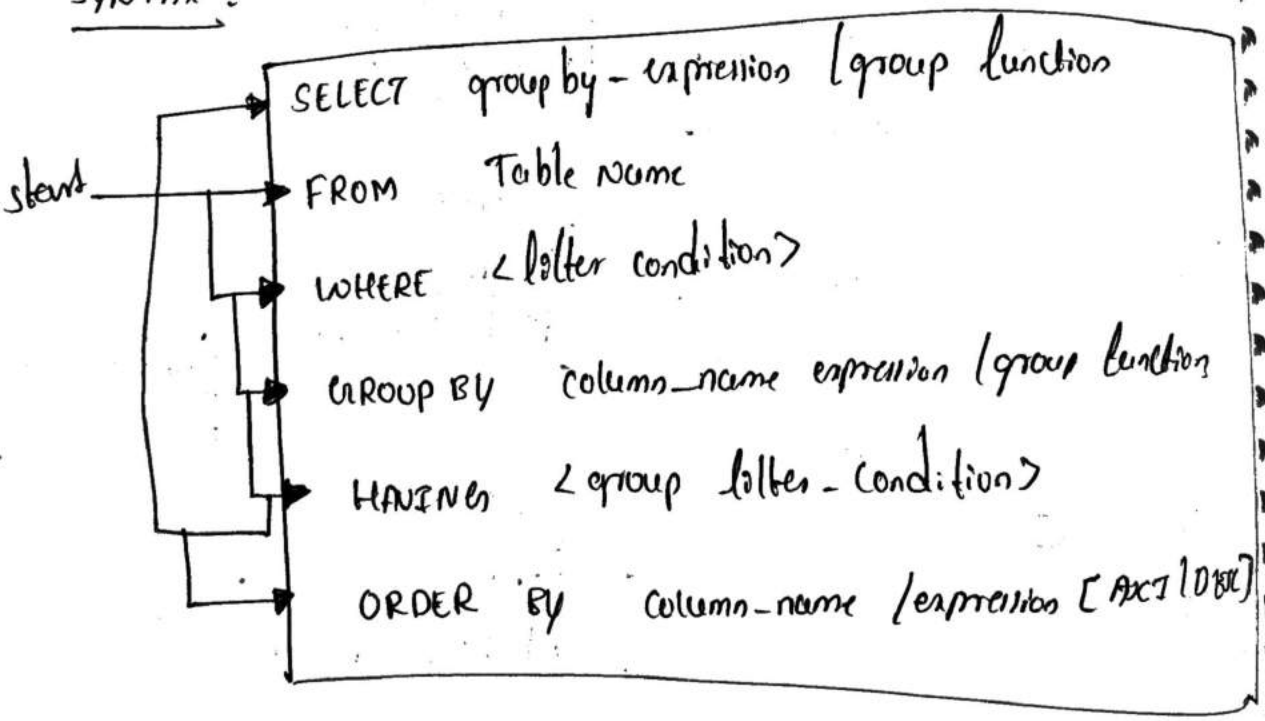
(2) ORDER BY CLAUSE is used to Arrange the records, either in ascending order or descending order.

(3) ORDER BY CLAUSE is execute after the select clause execute as last.

(4) ORDER BY CLAUSE ~~exit~~ execute Always last line.

(5) By default order by clause sorts the records in ascending order.

SYNTAX :-



Example:

WAQD total number of employees names in order wise.

```
SELECT ENAME
FROM EMP
ORDER BY ENAME = 'ASCI';
```

WAQD number of employees name is unorder wise.

```
SELECT ENAME
FROM EMP
ORDER BY ENAME='DESCI';
```

Differentiate Between WHERE CLAUSE and HAVING CLAUSE.

WHERE CLAUSE	HAVING CLAUSE
① WHERE CLAUSE is used to filter the condition	① HAVING CLAUSE is used to filter the group by clause
② WHERE CLAUSE is execute Row by row	② HAVING CLAUSE is depend on Group by clause
③ WHERE CLAUSE we will write one or more filter condition.	③ HAVING ~~FILTER~~ CLAUSE execute the group by group.
④ WHERE CLAUSE is SINGLE ROW FUNCTION (SRF).	④ HAVING CLAUSE is MULTI ROW FUNCTION (MRF).

3] SUB-QUERY :-

(*) Sub query :- a query written inside one more query is known as sub query.

(*) sub query contains A query inside one more query we will write that :- sub query.

working principle of sub query :-

(*) query inside one more sub query that have two parts

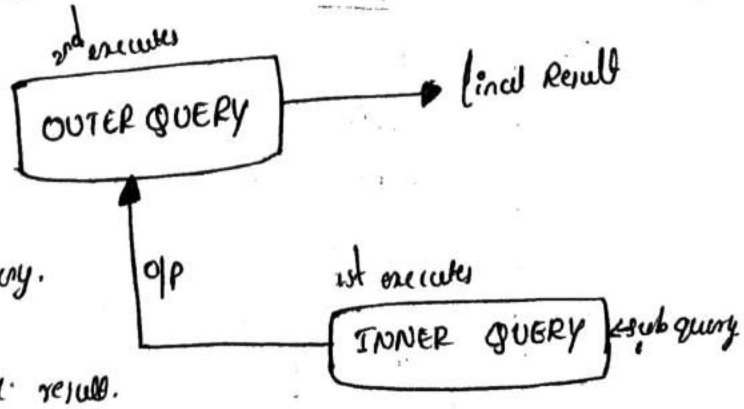
1] OUTER QUERY.

2] INNER QUERY.

1] OUTER QUERY :- outer query :- first part of sub query this outer query will get execute after sub query. it depends on Inner query.

2] INNER Query :- The Inner query is second part of sub query. Inner Query :- first execute after outer query will get execute.

Flow diagram :-



(*) Inner query will get executes

(*) Inner gives i/p as outer query.

(*) after outer query will get executes and give final result.

Why / when, do we use sub-query :-

CASE 1 :-

" Whenever we have unknown present in the question we will use sub query to find the unknown values.

for example :-

E.ID	E. Name	E.SAL	DEPT-NO
1	ALLEN	1000	20
2	BLAKE	2000	10
3	CLERK	3000	30
4	MILLER	1500	10
5	SMITH	2500	10

CASE 2 :-

" whenever the data to be selected, and the condition to be executed, the data are present in different Tables."

for example :

EMP

E.ID	E.Name	E.SAL	DEPT_NO
1	ALLEN	1000	20
2	BLAKE	2000	10
3	CLERK	3000	30
4	MILLER	1500	10
5	SMITH	2500	10

DEPT

DEPT-NO	D.NAME	LOC
10	D1	L1
20	D2	L2
30	D3	L3

Queries :-

1] WAQTD dept of the employee whose name is MILLER.

```
SELECT DEPT-NO
FROM EMP
WHERE ENAME = 'MILLER';
```

2] WAQD dname of the employee whose name is MILLER

```
SELECT dname                    → outer query.
FROM DEPT
WHERE deptno = (SELECT deptno.
                FROM EMP
                WHERE ENAME = 'MILLER';)
                                → Inner Query
```

3] WAQTD LOCATION of ADMI

```
SELECT LOC
FROM DEPT
WHER deptno = (SELECT dept_NO
               FROM EMP
               WHERE ENAME = 'ADMI';)
```

3] WAQTD All the Details of employee working in the same Designation as Miller and work in Location New york.

```
SELECT *
FROM EMP
WHERE JOB = (SELECT JOB
             FROM EMP
             WHERE ENAME='MILLER') AND
      DEPTNO = (SELECT DEPTNO
                FROM DEPT
                WHERE LOC='NEW YORK')
```

4] WAQTD All the Details of employees working in the same designation as MILLER, but work in Dept Department is ACCOUNTANT.

```
SELECT *
FROM EMP
WHERE JOB = (SELECT JOB
             FROM EMP
             WHERE DEPTNO = (SELECT DEPTNO
                             FROM DEPT
                             WHERE D.NAME='ACCOUNTANT'))
```

NESTED SUB QUERY :-

(1) A sub query written inside one more sub query is known as Nested sub query.

(2) We can Nest 255 sub queries.

NOTE:
MAX >
MIN <

Queries :-

1] WAQTD Maximum salary given to an Employees.

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

SAL
1000
2000
4000
3000
5000

2] WAQTD second maximum salary given to an Employees

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

(1) 4000 (2) 5000

3] WAQTD 3rd max salary given to an Employees.

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

(1) 3000 (2) 4000 (3) 5000

4] WAQ TO Deptname of the employees getting 2nd minimum salary

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

1st min 2000
2nd min 1000

Employees and managers Relationship :-

Note :

(1) Employees have manager each employee have one manager.
(2) All employees are working under managers.
(3) Manager have Handling all employees.
(4) Each employee have employee ID.
(5) Each Manager have mgr_NO

Note :

1] To find employees mangers by using [emp_id | Mgr_No] → Emp
2] To find Managers employees by using [Mgr_No | Emp_id] → Mgr No

Connection of Employee and Managers :-

CASE 1 → To find Emp managers we mgr-No → who's Employee NO same their is their emp manager.

EMP

Emp-NO	E_NAME	Mgr-NO
1	ALLEN	3
2	SMITH	1
3	JAMES	2
4	KING	3

(1) ALLEN manager is JAMES

(2) SMITH manager is ALLEN.

Queries :-

1) WAQTD name of the ALLEN manager.

```
SELECT E_NAME
FROM EMP
WHERE Emp-NO = (SELECT Mgr-NO
                FROM EMP
                WHERE ENAME = "ALLEN");
```

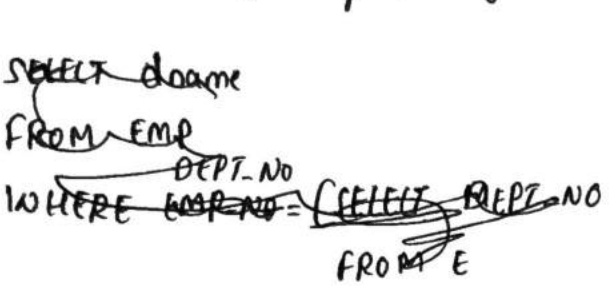
3 JAMES 3 ALLEN

2) WAQTD names of smith manager.

```
SELECT E_NAME
FROM EMP
WHERE EMP-NO = (SELECT Mgr-NO
                FROM EMP
                WHERE EMP-NO = (SELECT Mgr-NO
                                FROM EMP
                                WHERE ENAME = "SMITH"));
```

ALLEN

4) WAQTD dname of king manager.



```
SELECT dname
FROM DEPT
WHERE DEPT_NO = (SELECT DEPT_NO
                 FROM EMP
                 WHERE EMP_NO = (SELECT MGR_NO
                                 FROM EMP
                                 WHERE ENAME="KING"));
```

5) WAQTD Location of Adams manager's manager.

```
SELECT LOC
FROM DEPT
WHERE DEPT_NO = (SELECT DEPT_NO
                 FROM EMP
                 WHERE EMP_NO = (SELECT MGR_NO
                                 FROM EMP
                                 WHERE EMP_NO = (SELECT MGR_NO
                                                 FROM EMP
                                                 WHERE
                                                 ENAME='Adm
                                                 )));
```

CASE2: To find manager employee by using Emp_No → through mgr_no

Emp

Emp_No	ENAME	Mgr_No
1	ALLEN	3
2	SMITH	1
3	JAMES	2
4	KING	3

Queries

1] WAQTD Names of the employees reporting to king.

```
SELECT ENAME
FROM EMP A ALLEN
WHERE Mgr_NO = (SELECT EMP_NO
                FROM EMP
                WHERE ENAME = 'KING');
```

2] WAQTD. Name and salary given to the employees reporting to JAMES.

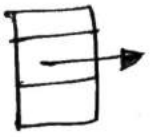
```
SELECT E.NAME, SAL
FROM EMP A
WHERE Mgr_NO = (SELECT EMP_NO
                FROM EMP
                WHERE ENAME = 'KING');
```

4) TYPES OF SUB-QUERY :-

There are two types of sub-query that is following below

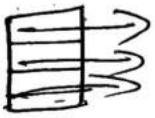
1) SINGLE ROW SUBQUERY
2) MULTI ROW SUBQUERY

1] SINGLE ROW SUBQUERY :-



(*) Single row sub query means, A query inside one more subquery that sub query executes with one values is called as single row subquery.

2] MULTI ROW SUBQUERY :-



(*) Multi row sub query means, A query inside one more subquery that subquery execute with one or more values is called Multi row sub query.

5] L

Example :-

Emp

Emp-Id	EName	Dept_No
1	ROLEX	2
2	NSA	3
3	APPU	4
4	BOSS	5
5	KIRAN	1

DEPT

DEPT-NO	D-NAME	LOC
2	S/Ftware	BLR
3	S/ Designing	MAS
4	S/ Analyst.	DVG
5	Electrical	Bider
1	Robotic	Belagavi

Queries:

1] WAQTD who working in Dept no 2 //single value

```
SELECT Dname
FROM DEPT
WHERE DEPT-NO =(SELECT DEPT_NO
                FROM EMP
                WHERE ENAME = 'ROLEX');
```

2] WAQTD who working in Dept no 2, and Dept NO 3 //more value,

```
SELECT Dname
FROM DEPT
WHERE DEPT-NO =(SELECT DEPTNO
                FROM EMP
                WHERE ENAME IN ('ROLEX, NJA'));
```

[9] JOIN'S :-

(R) Join is a connecting of two sections.

(R) Join is a linking of one table to another table.

(R) Join is a merging of two tables is called as Join.

JOIN have five types :-

1] ~~SCANNER~~ CROSS JOIN / CARTISION JOIN.

2] INNER JOIN

3] OUTER JOIN → left outer join
→ Right outer join
→ Full outer join

4] ~~NATURAL~~ NATURAL JOIN

5] SELF JOIN.

1] CROSS JOIN / CARTISION JOIN :-

(R) cross Join / cartision Join is a type of Join.

(R) cross Join is a column with pattern matching

SYNTAX :

oracle :

```
SELECT T1 colname , T2 colname , T3 colname ...
FROM   T1 , T2 , T3 ... ;
```

ANSI :

```
SELECT T1 colname , T2, colname, T3 colname
FROM   T1 CROSS JOIN T2 , T2 CROSS JOIN T3 ;
```

Example :-

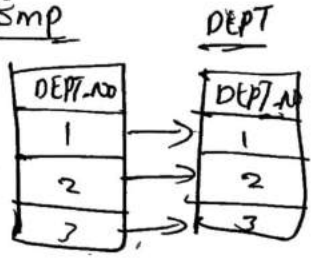
```
SELECT E,NAME, SAL, D_NAME
FROM EMP, DEPT ;
```

```
SELECT E_NAME, D_NAME
FROM EMP CROSS JOIN DEPT ;
```

2] INNER JOIN :- "INNER JOIN is also join column with pattern matching is called as INNER JOIN,"

OR

" Inner Joins mean combine two tables along with pattern matching with same values is called as inner join "



SYNTAX :

```
SELECT T1 colname, T2 colname
FROM   T1, T2
WHERE  < filter condition >
```

Example :

```
SELECT E_Name, D_name
FROM   Emp , DEPT
WHERE  DEPT_NO = DEPT_NO ;
```

3] OUTER JOIN :-

(*) Outer Join is type of Join. "the outer Join it is used to unmatched the records".

it's have three types

1] Left outer Join
2] Right outer Join
3] Full outer Join

1] Left Outer Join :- it is a type of outer Join.

"Left outer Join it is used to unmatched records with left table, along with matching records is called left outer join".

Emp

Left

E-NO	E-NAME	DEPT-NO
1	NSA	10
2	APPU	20
3	NAVEEN	30

DEPT

Right

DEPT-NO	D-NAME	LOC
10	S/w	BNG
20	H/w	DVG
40	IT	BNG

O/P

E.NO	E-NAME	DEPT-NO	DEPTNO	D-NAME	LOC
3	NAVEEN	30	NULL	NULL	NULL
1	NSA	10	10	S/w	BNG
2	APPU	20	20	H/w	DVG

unmatching record

matching record

SYNTAX :-

ANSI :

```
SELECT  column_Name
FROM  Table Name | LEFT | OUTER | JOIN  Table Name
ON  JOIN_CONDITION
```

Oracle :

```
SELECT  * column_name
FROM  Table_Name1 ,  Table_Name2
WHERE  Table1 . ColName = Table2 columnname (+) ;
```

Example :

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

2] Right Outer Join: It is one of the type of outer join.

"Right Outer Join it is used to unmatched records from right side table, along with matching records is called as Right Outer Join"

Emp

E.NO	E.NAME	DEPTNO
1	NJA	10
2	APPU	20
3	NAVEEN	30

Left

DEPT

DEPTNO	ENAME	LOC
10	SLW	BNG
20	HLW	DVG
40	IT	BNG

Right

QP

E.NO	ENAME	DEPTNO	DEPTNO	E.NAME	LOC
NULL	NULL	NULL	40	IT	BNG
1	NJA	10	10	SLW	BNG
2	APPU	20	20	HLW	DVG

unmatching record

matching record

Example:

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

3] Full outer Join : it is a type of outer join

"Full outer Join is it is used to unmatched record both left and Right from tables along with matching records is called as Full outer Join."

Emp

E_NO	E_NAME	DEPTNO
1	NSA	10
2	APPO	20
3	NAVEEN	30

DEPT

DEPT_NO	D_NAME	LOC
10	S/W	BNG
20	H/W	DVG
40	IT	BNG

E_NO	E_NAME	DEPT_NO	DEPTNO	D_NAME	LOC
3	NAVEEN	30	40	IT	BNG
1	NSA	10	10	S/W	BNG
2	APPO	20	20	H/W	DVG

Un matching record

matching record

Ex:

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

4] NATURAL JOIN :- It is one of type of Join.

(*) Natural Join plz act like cross Join, and there is a Relationship blw two tables thats act like inner Join is called as Natural Join.

Fruit

F.code	f-Name	cost
11	Orange	250
22	Apple	100

color

C-code	C-NAME	cost
80	SILVER	250
60	WHITE	100

Relationship blw two tables

SYNTAX :

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

Ex :

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

5] SELF JOIN :- It is a type of Join.

" self Join is a merging two tables with self Join is called as self Join ".

Emp

E-ID	E-NAME	MGR-NO
101	Nagendra	103
102	Nikil	104
103	Ramya	102
104	Krishna	101

Ex

```
SELECT  E1. E-NAME , E2. ENAME
FROM    EMP E1 , EMP E2 ;
```

EMP E1

EID	E-NAME	MGR-NO
100	Nagendra	103
102	Nikil	104
103	Ramya	102
104	Krishna	101

Emp E2

E-ID	ENAME	MGR-NO
101	Nagendra	103
102	Nikil	104
103	Ramya	102
104	Krishna	101

E-ID	E-NAME	MGR
101	Ramya	102
104	Krishna	101
102	Nikil	104
101	Nagendra	103

```
SELECT E1. ENAME, E2. ENAME
FROM EMP E1 , EMP E2
WHERE E1. MGR-NO = E2. EMPID
```

10 CO-RELATION SUB-QUERY

(1) Co-Relation means mutually related sub query.

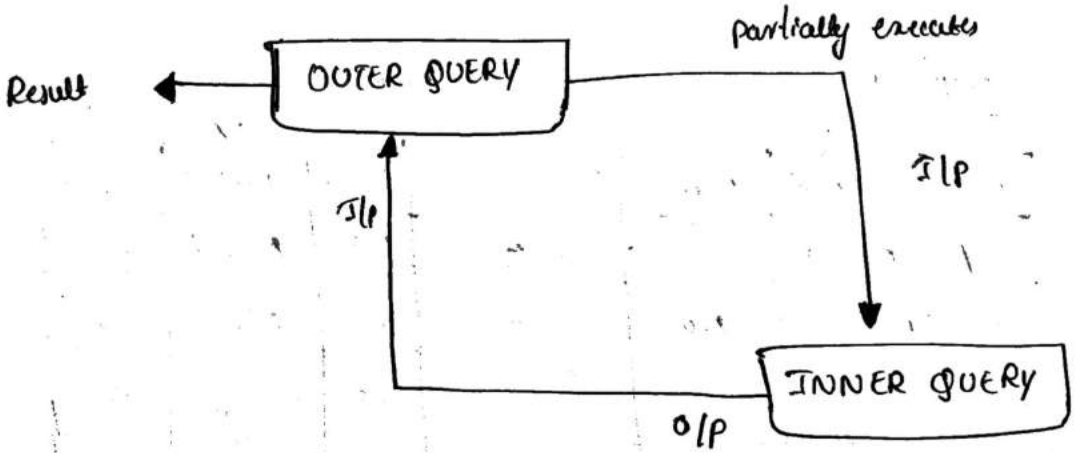
(2) "Co-Relation sub query is same as sub query. A query inside one more subquery but, the outer query will get execute first after, Inner query execute is called co-Relation sub-query."

(3) Co-Relation sub-query have two parts

1) OUTER QUERY
2) INNER QUERY

(4) first OUTER QUERY will get execute. after Inner query will get execute

Co-Relation sub-query flow diagram :-



(1) co-Relation's sub-query is a query inside one more sub query. but Outer Query will get execute first, after Inner Query will get execute

(2) co-Relation sub-query is partially execute.

Difference b/w Sub Query and Co-Relation Sub Query.

Sub-Query	Co-Relation Sub-Query
ⓚ Sub query is query inside one more query is called as sub query. In this Sub-Query have two parts outer Query & Inner Query. The Inner Query will get execute first after outer Query will get execute	ⓚ Co-Relation Sub Query is mutually Related to sub Query. Same as sub Query but outer Query will get execute first, after Inner Query will get execute is called as co-Relation Sub-Query.
ⓚ outer Query will dependent on Inner query.	ⓚ Co-Relation Sub query is partically executes • Both are independent.
ⓚ Join condition is not mandatory	ⓚ Join condition is mandatory any must be written in inner Query
ⓚ outer Query executes once.	ⓚ outer Query executes twice.

Queries on Co-Relation subquery:-

MAX = <
MIN = >

1) WAQTD find 2nd Max salary of Employees.

```
SELECT DISTINCT SAL        -> outer query will get executed
FROM EMP E1
WHERE (SELECT (DISTINCT SAL)
       FROM EMP E2
       WHERE E1.SAL <= E2.SAL) = 2;
```

EMP E1

100 <= 100 ✓

E1.ENAME	E1.SAL
A	100 → 2nd max
B	90
C	300 → 1st max
D	50

EMP E2

✓

E2.E-NAME	E2.SAL
A	100 → 2nd max
B	90
C	300 → 1st max
D	50

SAL
100
90
300
50

2) WAQDT find 3rd MIN. salary of Employees.

```
SELECT DISTINCT SAL
FROM EMP E1
WHERE (SELECT (DISTINCT SAL)
       FROM EMP E2
       WHERE E1.SAL >= E2.SAL) = 3;
```

NOTE: 2nd, 3rd max min condition we will give last or WHERE ~~after~~ condition after.

example:

```
SELECT DISTINCT SAL
FROM EMP E1
WHERE = 3 (SELECT (DISTINCT SAL)
          FROM EMP E2
          WHERE E1.SAL >= E2.SAL);
```

difference

will give where condition after, or last.

II PSUEDO CODE

(R) Psuedo code is a moces of step by step solve the given problem is called as psuedo code.

(R) Psuedo code columns are the false columns. are present in the each and every table must be called explicity.

(R) psuedo code columns ~~and~~ cannot be seen without calling them.

psuedo code have two types :-

1] ROW ID.

2] ROW NUM.

1] ROWID :- static ROW ID

ACDEFBZXYLMNSTUOPQ → it's unic & 18 digit.

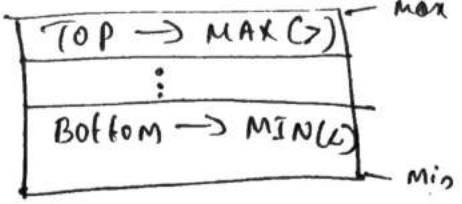
(R) ROWID is type of psuedo code.

(R) ROWID is the name itself it is ID, each & every Row have unic ROWID that is 18 digits is there that is called ROWID.

(R) ROWID each & every Row have ROW ID that is automatically generated when data insert to the table.

(R) ROWID is static that fixed values.

(R) ROWID can be used to Identify a records uniquely from the table when there is no Primary key Or foreign key.



SYNTAX of ROWID :-

To Fetch Top Records :

```
SELECT */ COLUMN-NAME /EXPRESSION
FROM EMP E1
WHERE (SELECT COUNT (DISTINCT ROWID)
       FROM EMP E2
       WHERE E1.ROWID >= E2.ROWID) = N;
```

Queries :-

1) WAQTD Display the 1st rowid

```
SELECT *
FROM EMP E1
WHERE (SELECT COUNT (DISTINCT ROWID)
       FROM EMP E2
       WHERE E1.ROWID >= E2.ROWID) = 1;
                                   = 2;
                                   = 3;

                                   IN (1,2,3,1);
```

2) WAQTD Display the last rowid

```
SELECT *
FROM EMP E1
WHERE ( SELECT COUNT (DISTINCT ROWID)
        FROM EMP E2
        WHERE E1.ROWID <= E2.ROWID) = 14;
```

2] ROW NUM :-

(1) ROW NUM is a type of pseudo code.

(2) ROW NUM is a serial / sequencial Number that is available in every row. when data in the row. or Table.

(3) ROW NUM is dynamic in nature. it's keep changing the serial number.

(4) ROW NUM always starts with 1 and cannot be duplicated.

(5) ROW NUM is used to fetch top and bottom records.

SYNTAX of ROW NUM :-

```
SELECT *
FROM ( SELECT ROWNUM SINO, EMP.*
       FROM EMP)
WHERE SINO = NO ;
```

Queries :-

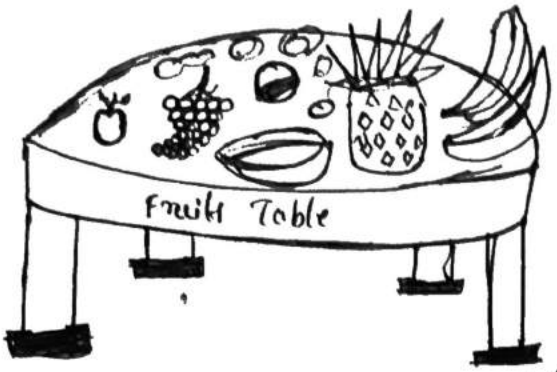
WAQTD it rownum of records.

```
SELECT *
FROM ( SELECT ROWNUM SINO, EMP.*
       FROM EMP)
WHERE SINO = 1;
      = 2;
      = 3;
      IN (1, 3, 7);
```

12 NORMALIZATION

(A) Normalization : Bringing larger table to smaller table is called as Normalization.

(K) Normalization is a filter the tables bring to Bigger tables to small tables is also known as Normalization.



gathered fruits.

each & every fruits sepearated.

(K) The fruits are before all in one Table after.

(I) The fruits are separated Tables small, small Tables. is called as normalization.

~~Normalization have~~.

Functional Dependency :- It is dependent on functionals.

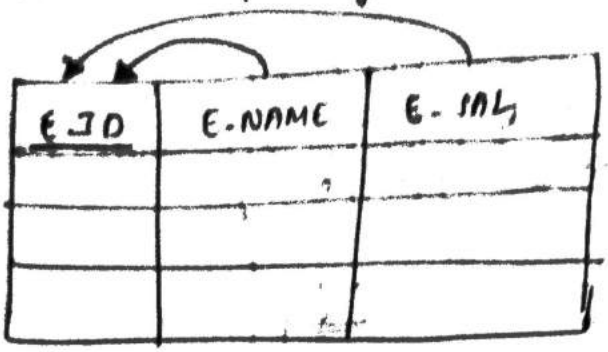
Its have Three types :-

1) Total Functional Dependency.
2) Partial Functional Dependency.
3) Transitive Functional Dependency.

1) Total Functional Dependency :-

(A) Total Functional Dependency is type of Functional Dependency

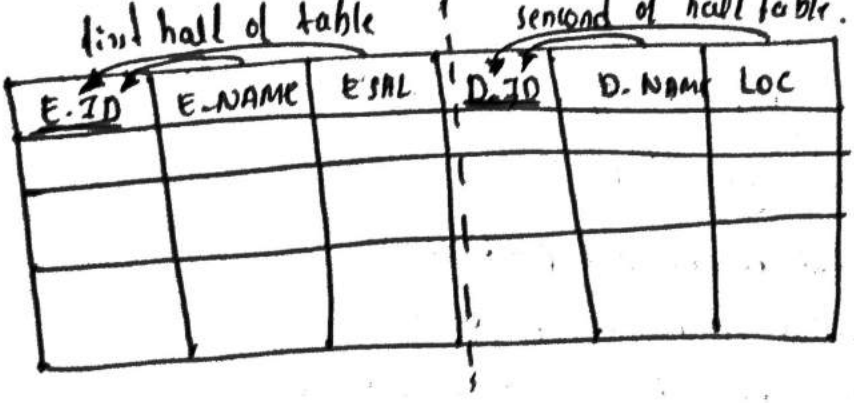
(A) Total Functional Dependency is All columns is dependent on first column that is primary key column is called as Total Functional Dependency.



2) Partial Functional Dependency :-

(A) Partial Functional Dependency is type of Functional Dependency.

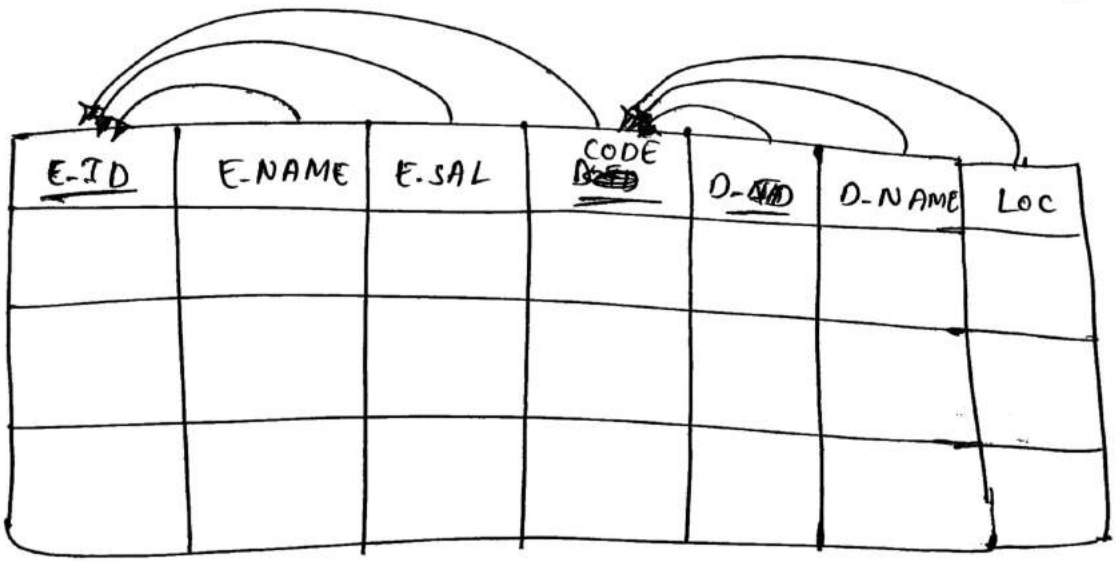
(A) partial Functional Dependency is larger table it's first half of table all columns dependent on first half of table first column and 2nd half of table all columns is dependent on 2nd half table first columns. is called as partial Functional Dependency



3] Transitive Functional Dependency :-

(Q) Transitive Functional Dependency is type of Functional Dependency.

" Transitive functional Dependency is a A larger table half of first table and 2nd half of table all columns are dependent their 1st half of table all columns dependent on first half of table of 1st column: and second half of table all column are dependent on 2nd half of the 1st column and that column also dependent on first of half of table first column is called as Transitive Functional Dependency.



Stored procedure :-

step1 :-

```
Create procedure XYZ
AS
FROM EMP
end,
```

//