

# SQL

INDEX.....

S.NO	TOPIC NO
1	APPLICATIONS
2	DATA
3	DATABASE
4	BASIC OPERATIONS PERFORMED ON DATABASE
5	DBMS SOFTWARE
6	RELATIONAL MODEL
7	TABLE
8	COLUMNS
9	ROWS
10	CELL
11	RULES OF EF. CODD
12	DATATYPES
13	CONSTRAINTS
14	STATEMENTS IN SQL
15	SHORTCUT TABLE FORMS
16	EXPRESSION
17	OPERAND & LITERALS
18	ALIAS
19	DISTINCT
20	SELECTIONS
21	OPERATORS
22	IN OPERATOR
23	NOT IN OPERATOR
24	BETWEEN OPERATOR
25	NOT BETWEEN
26	IS OPERATOR
27	LIKE OPERATOR
28	CONCATINATION OPERATOR
29	FUNCTIONS
30	LIST OF MULTIROW FUNCTIONS/GROUP/AGGRIGATE FUNCTIONS
31	SINGLE ROW FUNCTIONS
32	SUBSTRING
33	REPLACE
34	INSTR
35	SYSDATE
36	CURRENT DATE
37	SYSTIMESTAMP
38	TO_CHAR()
39	FORMAT MODELS
40	NVL(); [NULL VLAUE LOGIC]
41	GROUP BY CLAUSE

42	HAVING CLAUSE
43	ORDER BY
44	SUB-QUERY (CASE-1,2)
45	JOINS
46	TYPES OF JOINS
47	TABLE CREATION 1. DDL 2. DML 3. TCL 4. DCL 5. DQL
48	FUNCTIONAL DEPENDENCY
49	PARTIAL FUNCTIONAL DEPENDENCY
50	TRANSITIVE FUNCTIONAL DEPENDENCY
51	SET THEORY
52	SET OPERATORS
53	VIEW
54	DIFF B/W TABLE & VIEW & ADV OF VIEW
55	INDEX
56	PROCEDURE/STORED PROCEDURE
57	CURSOR
58	DIFF B/W JOINS AND SET OPERATORS

SQL

- **SQL is nothing but DATABASE.**

### **STAND ALONE APPLICATION :**

It is an application that runs locally on the device and doesn't require anything else to be functional.

All the logic is built into the app, So it doesn't need an Internet connection nor any other services installed. They need no browser to run, but often demand a device to be installed on.

**E.g.:** Paint, PowerPoint, Excel, Vlc,.etc.

### **WEB APPLICATION:**

It is a application software that runs on a webserver, unlike computer-based software programs that are run locally on the operating system of the device.

Web application are accessed by the user through a web browser with an active network connection.

**E.g.:** F.B, WhatsApp, TikTok, etc.

- Every application divided into three categories.

- ❖ FRONT-END.
- ❖ MIDDLEWARE.
- ❖ BACK-END.

SQL

### **FRONT-END:**

Whatever we see on the screen is called front end.

- ❖ Front end web development is the development of the graphical user interface of a website, through the use of HTML,CSS & JAVASCRIPT, So that users can view and interact with web-site.

### **MIDDLEWARE:**

It is a software that enables one or more kinds of communication /connectivity between two or more application in a distributed network.

There are many types fo middleware. Some, such as message brokers or transaction processing monitors, focus on one type of communication.

- ❖ There are some set of programs written all those were responsible for middleware

**E.g.:** Java,Python,C,C++,JavaScript ,Note.....etc.

### **BACK-END:**

It is also called the server side, consists of the server which provides data on request, the application which channels it, and the database which organize the information.

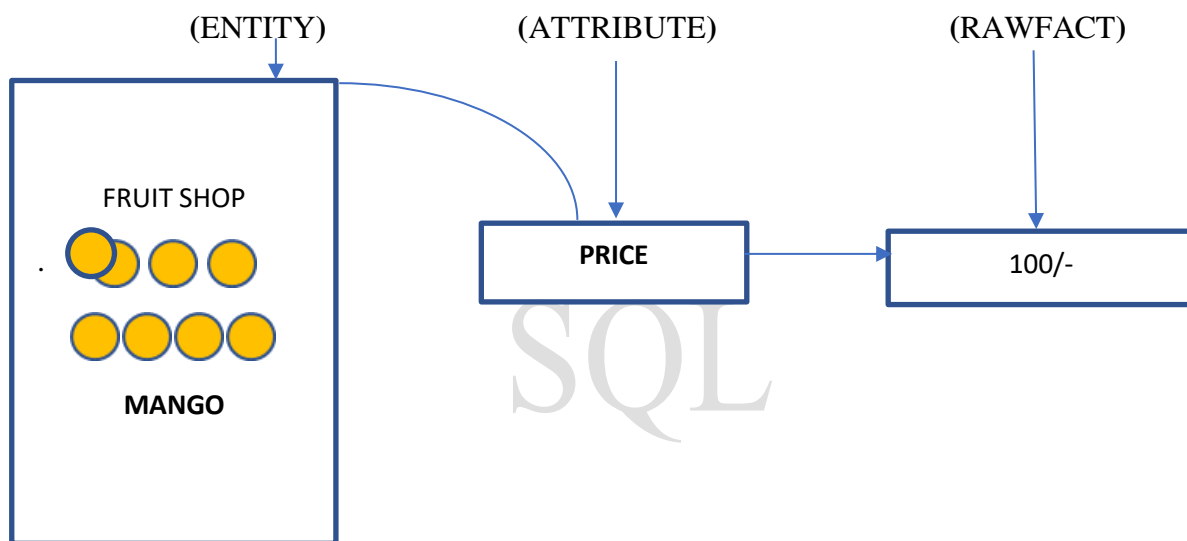
- ❖ Every application has own database that database is nothing but backend.  
Backend there are two types 1) **SQL** 2) MANGO DB.  
SQL is most preferred.

**E.g.:** When a customer browses shoes on website, they are interacting with front end.

### **DATA:**

It is the Raw fact that describes the **Attributes** of an **Entity**.

**E.g.;**



### **DATABASE:**

Place where Data is stored in Symmetric and organized manner.

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually connection by a Database Management System (**DBMS**). Most databases use Structured Query Language (**SQL**) for writing and querying data.

### BASIC OPERATIONS PERFORMED ON DATABASE:

- Create/Insert
- Read/Retrieve
- Update/Modify
- Delete/Drop



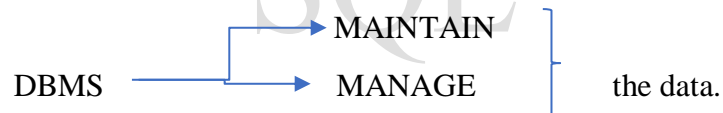
Data was stored in Database and it was managed in Database Management System (DBMS).

### DBMS SOFTWARE:

DBMS is a software we use to maintain & manage database.

Two important factors are: -

1. SECURITY.
  2. AUTHORIZATION.
- We use “**Query Language**” to communicate.



### DBMS FEATURES

- SECURITY.
- AUTHORIZATION.

### DBMS → MANY TYPES

- NETWORK DBMS
  - HIERARCHY DBMS
  - OBJECT ORIENT DBMS
  - **RELATIONAL DBMS → RDBMS**
- MOSTLY NOT USED**

**RDBMS** is a type of DBMS software.

## **RDBMS**

It's a type of DBMS software where Data will be stored in the form of Tables.

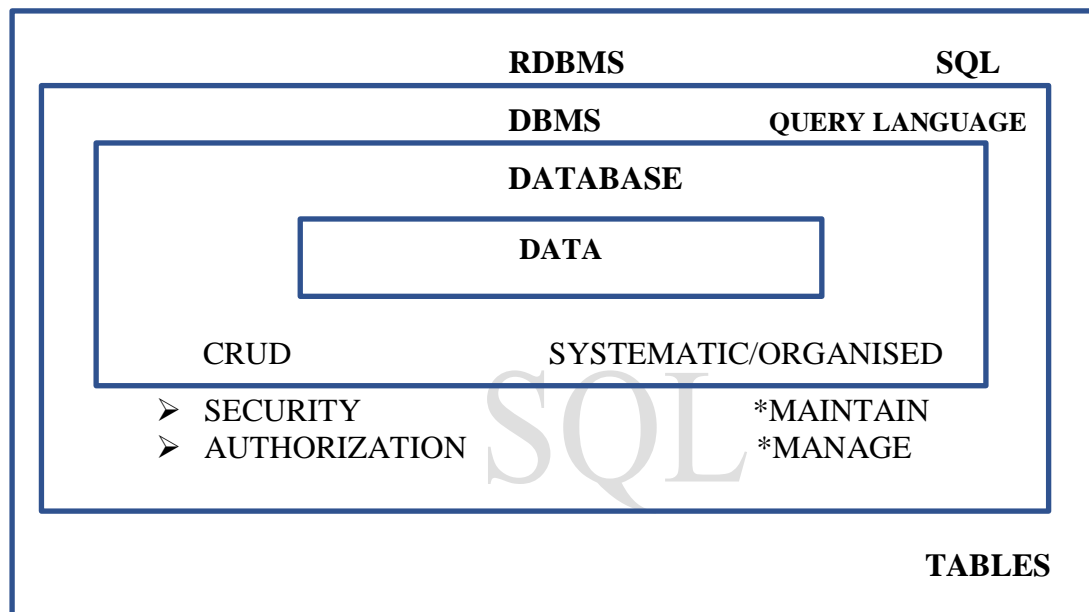
- WHY?
- WHY RDBMS?

It was under **Relational Model Theory** introduced by **E.F. CODD**.

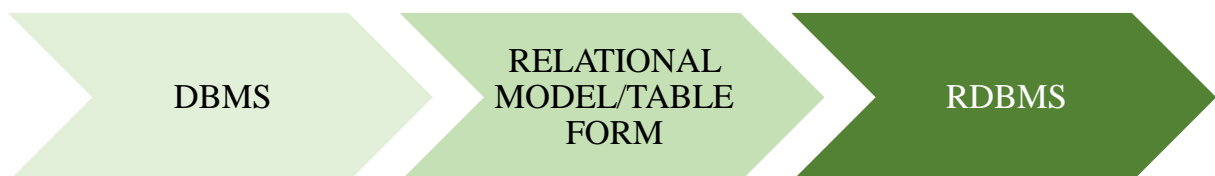
Two important Factors are: -

- Security.
- Authorization.

We use **STRUCTURED QUERY LANGUAGE (SQL)** to communicate.



Any DBMS if it follows Relational model/data is stored in table form is called RDBMS.





## **RELATIONAL MODEL:**

It is a concept designed by Data Scientist "**E.F.CODD**".

In Relational Model we store the data in the form of **TABLES**.

A DBMS software which follows Relational model becomes Relational DBMS (**RDBMS**).

<b>ID</b>	<b>NAME</b>	<b>BRANCH</b>	<b>PERCENTAGE</b>	<b>DOB</b>
1	AKHIL	ECE	72	1-1-2000
2	NIKHIL	CSE	78	1-2-2000
3	KRISH	ECE	72	1-3-2000
4	VISH	EEE	89	1-4-2000

- ✓ YELLOW – CELL
- ✓  - ROWS
- ✓  - COLUMNS
- ✓ **BOLD** words are Attributes.

In RDBMS we cannot store more than **ONE VALUE**.

### **TABLE:**

Table is a logical organization of data which consists of Rows & Columns.

### **COLUMNS:**

Columns is also referred as attributes or fields.

A column is used to represent one property of all entities.

### **ROWS:**

Rows is also referred as records or Tuples.

A row is used to represent all the properties of single entity.

### **CELL:**

Cell is the smallest unit in table in which we store data.

The inter-section of Rows & columns Generate Cells.

### **RULES OF E.F.CODD:**

- The Data is stored in the cell must be **SINGLE VALUE DATA**.
- In RDBMS we store everything in the form of tables including metadata.

(THE DETAILS ABOUT THE DATA IS **METADATA**).

- According to E.F.CODD we can store Data in Multiple tables, If needed we can Establish connection between two tables using attributes.
- We can validate the data entered into the table in two steps.
  - 1) By assigning **Data type**.
  - 2) By assigning **Constraint**.

#### ❖ **NOTE :**

- Here **Datatypes** are mandatory whereas **Constraints** are optional.

## DATATYPES:

It is used to determine what type or kind of data will be stored in a particular memory location.

- CHAR
- VARCHAR/VARCHAR2
- NUMBER
- DATE
- LARGE OBJECT.
  - 1) CHARACTER LARGE OBJECT. (CLOB)
  - 2) BINARY LARGE OBJECT. (BLOB)

## CHAR:

Char datatype can accept characters

- 'A-Z'
- 'a-z'
- '0-9' or
- Special Characters (#,\*,\$,...)

Syntax: -

## **CHAR (SIZE)**

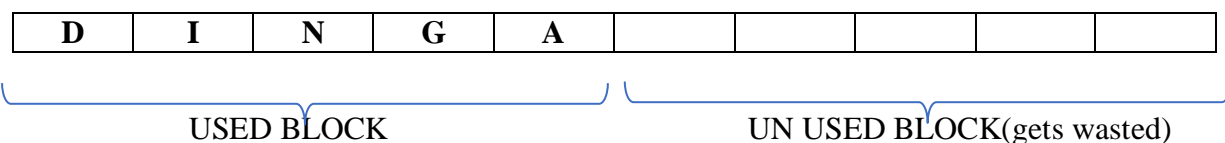
### SIZE:

It is used to determine the number of characters that we can store.

- ❖ Whenever we mention char datatype, we have to mention size for it.
- ❖ The max size that we can store is "2000".
- ❖ It is a type of "FIXED LENGTH MEMORY ALLOCATION"

E.g.:

**CHAR (10)** we can save up to 2000.



To overcome char, we have varchar.

## VARCHAR:

Varchar datatype can accept characters such as

- 'A-Z'
- 'a-z'
- '0-9' or
- Special Characters (#,\*,\$,...)



SYNTAX:

**VARCHAR(SIZE)**

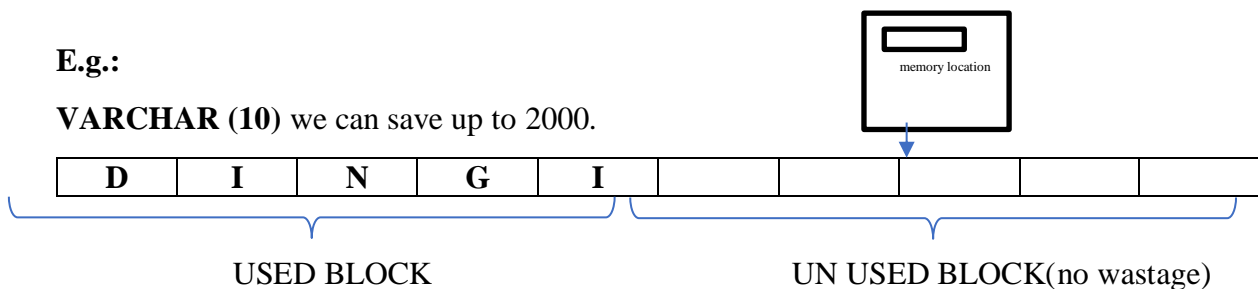
**SIZE:**

It is used to determine the number of characters that we can store.

- ❖ Whenever we mention char datatype, we have to mention size for it.
- ❖ The max size that we can store is “**2000**”.
- ❖ It is a type of “**VARIABLE LENGTH MEMORY ALLOCATION**”

**E.g.:**

**VARCHAR (10)** we can save up to 2000.



**VARCHAR – 2:**

It is the updated version of varchar i.e., size is updated from **2000** to **4000**.

SYNTAX:

**VARCHAR2(SIZE)**

The, Max size we can store is ‘**4000**’.

### INTERVIEW QUESTIONS

#### ❖ DIFFERENCE BETWEEN CHAR AND VARCHAR.

CHAR	VARCHAR
➤ Wastage of Memory	➤ No wastage of Memory.
➤ Fixed length Memory allocation.	➤ Variable length memory allocation
➤ CHAR(SIZE)	➤ VARCHAR(SIZE)

#### ❖ DIFFERENCE BETWEEN VARCHAR AND VARCHAR2.

VARCHAR	VARCHAR2
➤ Size is 2000	➤ Size is 4000
➤ VARCHAR(SIZE)	➤ VARCHAR2(SIZE)

## NUMBER:


The datatype is used to store numerical values.

It can accept two arguments.

- a) Precision.
- b) Scale.

SYNTAX:

**NUMBER(PRESCION[SCALE])**



NOT MANDATORY

## PRECISION:

It is used to determine the digits we are going to store in numerical place.

## SCALE:

It is used to determine the number of digits we are going to store in decimal place within the precision.

- ❖ The maximum precision we can store is '38'.
- ❖ The maximum scale we can store is '127'.
- ❖ CASE-1:      NUMBER (5)  
                    E.g., =    +9 9 9 9 9
- ❖ CASE

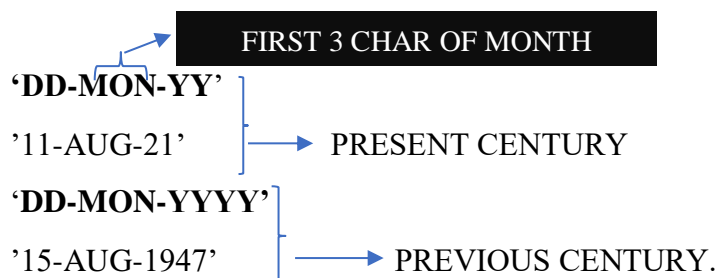
## DATE:

SYNTAX – DATE

The two oracle specified data formats are

- a) 'DD-MON-YY'
- b) 'DD-MON-YYYY'
- 🚦 Date should always be enclosed with 'SINGLE QUOTES'.

E.g.:



'DD-MON-YY' } FIRST 3 CHAR OF MONTH

'11-AUG-21' } PRESENT CENTURY

'DD-MON-YYYY' } PREVIOUS CENTURY.

'15-AUG-1947' }

## **NUMBER:**

The Datatype is used to store numerical values.

It can accept two arguments.

- Precision.
- Scale.

## **SYNTAX:**

NUMBER(PRESCISION,[SCALE])

## **PRECISION:**

It is used to determine the digits we are going to store in numerical place.

## **SCALE:**

It is used to determine the no of digits we are going to store in decimal place within the prescion.

The max precision we can store is '38'.

The max scale we can store is '127'.

**CASE** – 1 : NUMBER(5)

+9 9 9 9 9

0 { + 9 9 9 9 9  
- 9 9 9 9 9

**CASE** – 2 : P>S (PRECION IS GREATER THAN SCALE)

NUMBER (5,2)

+ - 999,99

**CASE** – 3: P=C

NUMBER (4,4)

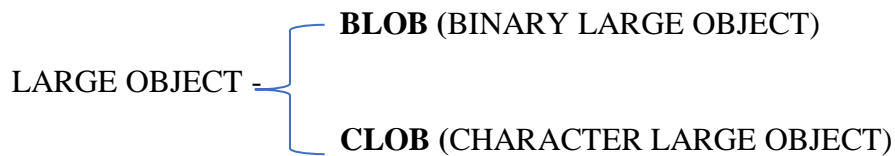
+ - 0.9999

**CASE** – 4 : P<S

NUMBER (3,5)

+ - 0.00999

## LARGE OBJECT:



**BLOB** – PICTURE, VIDEOS, FILES ETC....

➤ ‘**4GB**’ – STORAGE SPACE.

❖ If we want to store more than 4000 characters we use CLOB data type. More than 4GB we go for CLOB data types.

## NOTE:

- CLOB is used to store the “CHARACTERS”.
- BLOB is used to store the “PHOTOS, IMAGES” ...ETC.

a) CLOB (CHARACTER LARGE OBJECT)

➤ This is used to store the characters up to 4GB of size.

**SYNTAX: - CLOB**

b) BLOB (BINARY LARGE OBJECT)

➤ This is used to store the Img, Videos, file, etc.

**SYNTAX :- BLOB**

NUMBER (33)	VARCHAR (20)	VARCHAR(15)	NUMBER(7,2)	NUMBER(10)
EID	NAME	DESIGNATION	SALARY	CONTACT NO
101	MUKESH	TEST ENGG	40000.75	7410258963 (101,102 SAME NUM)
102	SURESH	TEST ENGG	(EMPTY SALARY)	7410258963 (101,102 SAME NUM)
103 (SAME ID)	RAMESH	(EMPTY JOB)	-36000.75 (-VE SALARY)	987654123 (INCOMPLETE)
103 (SAME ID)	KAMESH	ANALYST	75000.44	7410235689
105	SARVESH	PRESIDENT	96325	74102356 (INCOMPLETE)

THESE ERRORS ARE REMOVED BY “CONSTRAINTS”.

### **CONSTRAINTS:** -

Constraints are the conditions are assigned to a particular column to validate data.

### **TYPES OF CONSTRAINTS:** -

- UNIQUE
- NOT NULL
- CHECK
- PRIMARY KEY
- FOREIGN KEY

### **UNIQUE:** -

Unique is s a constraint which is assigned to a particular column which cannot accept repeated or duplicate value.

### **NOT NULL:** -

It is a constraint which is assigned to a particular column which cannot be null or which are mandatory.

### **CHECK:-**

It is a constraint which is assigned to a particular column for extra validations.

Check constraint is assigned with a condition, If the condition is true the value gets accepted, else rejected.



E.g.

- 1) AADHAR CARD
- 2) PAN CARD
- 3) PHONE NUMBER
- 4) EMP ID
- 5) SR NO
- 6) MAIL ID
- 7) USER NAME

### **PRIMARY KEY:** -

Primary key is a constraint which is used to assign to a column to identify a record uniquely from the table.

### **CHARACTERISTICS OF PRIMARY KEY:** -

- We can have only one primary key in a table.
- It cannot accept repeated or duplicate values.
- It cannot accept null.
- It's a combination of unique and not.
- Primary key is not mandatory but recommended to have one in table.

### **FOREIGN KEY:** -

It is constraint which is used to establish the connection between two tables.

### **CHARACTERISTICS OF FOREIGN KEY:** -

- We can have 'n' no of foreign keys in table.
- It can accept repeated or duplicate values.
- It can accept null.
- It is not a combination of unique and not null.
- It is present in child table but actually belongs to parent table.
- It is also referred as '**REFERENTIAL INTEGRITY CONSTRAINT**'.
- .....
- Only primary key can travel to another table.
- When it travels it becomes foreign key.

### **STATEMENTS IN SQL:**

- |                                 |       |
|---------------------------------|-------|
| 1) DATA DEFINITION LANGUAGE     | [DDL] |
| 2) DATA MANIPULATION LANGUAGE   | [DML] |
| 3) DATA CONTROL LANGUAGE        | [DCL] |
| 4) TRANSACTION CONTROL LANGUAGE | [TCL] |
| 5) DATA QUERY LANGUAGE          | [DQL] |

### **DATA QUERY LANGUAGE:** -

This statement is used to retrieve the data from database.

- ❖ There are 4 statements
  - 1) SELECT
  - 2) PROJECTION
  - 3) SELECTION
  - 4) JOIN

### **SELECT:** -

This statement is used to retrieve the data from database and display it.

### **PROJECTION:** -

This statement is used to retrieve the data from database by selection only column.

All values in the column will be selected by default.

#### ❖ **SYNTAX**

SELECT \*/[DISTINCT] COLUMN\_NAME/EXPRESSION [ALIAS] FROM  
TABLE\_NAME.

### **SELECTION:** -

This statement is used to retrieve data from database y selection both columns as well as recorder.

### **JOINS: -**

This statement is used to retrieve the data from multiple tables simultaneously.

CLAUSES {  
SELECT NAME → COLUMN NAME  
FROM STUDENT; → TABLE NAME

### **STUDENT**

SID	NAME	BRANCH	PERCENTAGE
101	AMAR	CSE	72
102	NAYAN	CSE	73
103	KAMAL	ECE	77
104	SRUTHI	IT	89

### **Q. WRITE A QUERY TO DISPLAY STUDENT NAME?**

A) SELECT NAME (COLUMN NAME)  
FROM STUDENT;(TABLE NAME)

O/P:-

NAME
AMAR
NAYAN
KAMAL
SRUTHI

### ❖ **NOTE: -**

- 1) From clause starts the execution.
- 2) For from clause, we can pass table name as argument.
- 3) The job of from clause is to go to the database & search for the table and put the table under execution.
- 4) Select clause executes after the execution of from clause.
- 5) For select clause we can pass asterisk column name & expression as argument.
- 6) The job of select clause is to go to the table which is under execution & select data and display.
- 7) Select clause is responsible for the result table.

**Q. WAQTD NAME & BRANCH OF ALL THE STUDENTS?**

A)     SELECT NAME, BRANCH  
       FROM STUDENT;

**O/P:-**

NAME	BRANCH
AMAR	CSE
NAYAN	CSE
KAMAL	ECE
SRUTHI	IT

**Q. WAQTD BRANCH & PERCENTAGE?**

A)     SELECT BRANCH, PERCENTAGE  
       FROM STUDENT;

**O/P: -**

BRANCH	PERCENTAGE
CSE	72
CSE	73
ECE	77
IT	89

❖ WE HAVE TO WRITE SAME NAME THAT IS GIVEN.

**Q. WAQTD ALL THE DETAILS FROM THE TABLE?**

A)     SELECT SID, NAME, BRANCH, PERCENTAGE  
       FROM STUDENT;

(OR)

SELECT \*  
FROM STUDENT; } (MAIN PREFERENCE)

❖ WE CANNOT CHAGE THE ORDER.

- If we want to write more than one column, we separate it by using comma (,).
- We have to write some column name, table name presents in database.
- If we want to select all the columns we have to go for (\*) asterisk.
- We cannot change the order of given query.



**Q. WAQTD SID, NAME, PERCENTAGE & BRANCH OF ALL STUDENT?**

A) SELECT SID, NAME, PERCENTAGE, BRANCH  
FROM STUDENT;

**O/P:-**

SID	NAME	PERCENTAGE	BRANCH
101	AMAR	72	CSE
102	NAYAN	73	CSE
103	KAMAL	77	ECE
104	SRUTHI	89	IT

- ❖ To find the column name in table
- ❖ We use DESC DEPT (DESC-DESCRIBE)

**SHORT CUT TABLE FULL FORMS:-**

- EMP NO → EMPLOYEE NO
- ENAME → EMPLOYEE NAME
- JOB → DESIGNATION
- MGR → MANAGER
- HIREDATE → DATE OF JOINING
- SAL → SALARY
- COMM → COMMISSION
- DEPT NO → DEPARTMENT NUMBER

**DESC DEPT:-**

- DEPT NO → DEPARTEMENT NUMBER
- D NAME → DEPARTMENT NAME
- LOC → LOCATION

**Q. WAQTD ALL THE DETAILS OF EMPLOYEE TABLE?**

A) SELECT \*  
FROM EMP;

**Q. WAQTD NAMES OF ALL THE EMPLOYEES?**

A) SELECT ENAME  
FROM EMP;

**Q. WAQTD NAME AND SALARY GIVEN TO ALL THE EMPLOYEE**

A) SELECT ENAME, SAL  
FROM EMP;

**Q. WAQTD NAME AND ANNAUAL SALARY GIVEN TO ALL THE EMPLOYEE?**

A) SELECT ENAME, SAL\*12  
FROM EMP;

**Q. WAQTD EMPLOYEE ID & DEPT NO OF ALL EMPLOYEES IN EMP TABLE?**

A) SELECT EMP NO, DEP NO  
FROM EMP;

**Q. WAQTD ENAME & HIREDATE OF ALL THE EMPLOYEE?**

A) SELECT ENAME, HIREDATE  
FROM EMP;

**Q. WAQTD NAME & DESIGNATION OF ALL EMP?**

A) SELECT ENAME, JOB  
FROM EMP;

**Q. WAQTD NAME & HALFTERM SALARY OF ALL THE EMPLOYEE?**

A) SELECT E NAME, SAL\*6 OR SAL\*12/2  
FROM EMP;

**Q. WAQTD ENAME & SAL & ALSO A SAL WITH 25% HIKE?**

A) SELECT ENAME, SAL, SAL+SAL\*25%  
FROM EMP;

**Q. WAQTD NAME, SAL & SAL DED OF 12%?**

A) SELECT ENAME, SAL, SAL-SAL\*12  
FROM EMP;

**EXPRESSION:** -

A statement which gives us result is known as expression.

- Expression consists of 2 types
  - 1) Operand
  - 2) Operators (+, -, \*, /)

**OPERAND:** -

Operand consists of two types.

- COLUMN NAME
- LITERALS (DIRECT VALUES)

### **LITERALS:** -

Literals are of three types.

- Number literal
- Character literal
- Date literal
- CHARACTER LITERAL & DATE LITERAL SHOULD BE ENCLOSED WITH SINGLE QUOTES (‘’)

OPERAND	OPERATOR	OPERAND
COLUMN NAME		COLUMN NAME
LITERALS		LITERALS
➤ NUMBER LITERAL	+, -, *, /	➤ NUMBER LITERAL
➤ CHARACTER LITERAL		➤ CHARACTER LITERAL
➤ DATE LITERAL		➤ DATE LITERAL

Anything which is written in the format operand, operator & operand we call it as ‘Expression’.

**E.g.**

Sal + 12 / sal + sal \* 25 / 100

**RED COLOUR = OPERAND**

**BLACK COLOUR = OPERATORS**

**Q. WAQTD ENAME & SAL & SAL WITH 18% HIKE?**

**A)**

```
SELECT ENAME, SAL, SAL+SAL*18/100
FROM EMP;
```

**Q. WAQTD ENAME & SAL & ALSO SAL WITH 40% DEDUCTION?**

**A)**

```
SELECT ENAME, SAL, SAL-SAL*40/100
FROM EMP;
```

**Q. WAQTD ENAME & SAL & ALSO SAL WITH 7% HIKE?**

**A)**

```
SELECT ENAME, SAL, SAL+SAL*7/100
FROM EMP;
```

**Q. WAQTD ENAME & ANNUAL SAL & ALSO SAL WITH 34% HIKE?**

A)

```
SELECT ENAME, SAL, SAL+SAL*34/100
FROM EMP;
```

**Q. WAQTD ENAME & ANNUAL SAL WITH BONUS OF 5000?**

A)

```
SELECT ENAME, SAL*12+5000
FROM EMP;
```

- ❖ WE CALL ALTERNATE NAMES IN SQL AS ALIAS.
- ❖ ALIAS IS NOT THE PERMENATN NAME THIS IS TEMPORAR NAME.

E.g.

SAL\*12 AS ANNUAL SAL

SAL\*12 AS ANNUAL\_SAL

SAL\*12 AS "ANNUAL SAL"

- ❖ WRITING AS IS NOT MANDATORY.
- ❖ WRITING ALIAS NAMES ALSO NOT MANDATORY BUT HIGHLY RECOMMENDED.

**ALIAS: -**

Alias is an alternative name given to a column or an expression in the result table.

Alias name can be used with or without using 'AS' keyword.

Alias name should be a single word or a string enclose within double quotes.

Alias is not mandatory but recommend to provide.

**QUESTIONS ON ALIAS**

**Q. WAQTD NAME, SAL OF THE EMPLOYEE?**

A)

```
SELECT ENAME, SAL AS SALARY, SAL*12 AS ANNUALSAL
FROM EMP;
```

**Q. WAQTD ENAME & JOB FOR ALL THE EMPLOYEE WITH THEIR HALF TERM SAL?**

A)

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

**Q. WAQTD ALL THE DETAILS OF THE EMPLOYEES ALONG WITH AN ANNUAL BONUS OF 2000?**

A) SELECT EMP.\*, SAL\*12+2000

FROM EMP;

❖ IN SQL IF WE WRITEEN \* IT DOESN'T NEED OR CHECK WITH OTHERS (COLUMN NAME/EXPRESSION) IT DIRECTLY GOES.

**Q. WAQTD NAME SALARY AND SALARY WITH HIKE OF 10%?**

A)

```
SELECT ENAME, SAL, SAL+SAL*10/100 AS HIKE  
FROM EMP;
```

**Q. WAQTD NAME & SALARY WITH DED OF 25%?**

A)

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

**Q. WAQTD NAME & SALARY WITH MONTHLY HIKE OF Rs.50/-?**

A)

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

**Q. WAQTD NAME & ANNUAL ALARY WITH DED OF 10%?**

A)

```
SELECT ENAME,SAL*12,SAL-SAL*10/100  
FROM EMP;
```

**Q. WAQTD TOTAL SALARY GIVEN TO EACH EMPLOYEE?**

A)

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

**Q. WAQTD TOTAL NAME & DESIGNATION ALONG WITH 100RS PENALTY IN SALARY?**

A)

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

**DISTINCT: -**

Remove the duplicate/repeated value.

- To remove repeated values or duplicated values in result table we use distinct values.
- For distinct clause we can pass column name or an expression as an argument.
- Distinct clause should be used as the first argument in the select clause.
- We can pass multiple columns for distinct clause.
- It removes the combination of duplicates from all the columns.

**STUDENT**

ID	NAME	BRANCH	YOP
101	A	CSE	2020
102	B	ECE	2021
103	C	IT	2019
104	D	EEE	2020
105	E	MECH	2021

**Q. WAQTD DIFFERENT BRANCHES IN STUDENT TABLE?**

A)

```
SELECT DISTINCT BRANCH  
FROM STUDENT;
```

**Q. WAQTD DIFF YOP FROM STUDENT TABLE?**

A)

```
SELECT DISTINCT YOP  
FROM STUDENT;
```

**Q. WAQTD DIFFERENT BRANCHES & YOP?**

A)

```
SELECT DIFFERENT BRANCH, YOP  
FROM STUDENT;
```

**Q. WAQTD ONLY DIFFERENT SALARIES GIVEN TO EMPLOYEE?**

A)

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

**Q. WAQTD THE DIFFERENT DESIGNATION HAT ARE PRESENT IN EMP TABLE?**

**A)**

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

**Q. WAQTD DIFFERENT DEPT NO AS WELL AS & SALARIES THAT ARE PRESENT IN TABLE?**

**A)**

```
SELECT DISTINCT DEPTNO,SAL  
FROM EMP;
```

**Q. WAQTD DIFFERENT DEPT NO PRESENT IN EMP TABLE?**

**Q. WAQTD NAME & EMP ID OF ALL EMPLOYEE**

**Q. WAQTD ANNUAL SALARY ALONG WITH SALARY HIKE IN 33%?**

**Q. WAQTD DIFFERENT NAMES WE HAVE IN EMPLOYEE?**

**Q. WAQTD ALL THE DETAILS OF EMPLOYEE ALONG WITH MONTHLY HIKE OF RS 100?**

**Q. WAQTD DEP NO OF ALL THE EMPLOYEES ALONG WITH THEIR NAME?**

**Q. WAQTD DESIGNATION OF ALL THE EMPLOYEE?**

**Q. WAQTD DATE OF JOIN OF ALL EMPLOYESS**

**Q. WAQTD SAL AS SALARY & HIRE DATE AS DATE OF JOINING?**

**SELECTIONS: -**

Selections is one which select the data from the table by using both column name & row name.

**WHERE CLAUSE: -**

This is used to filter the records

**NOTE: -**

- ❖ For where clause we can pass filter condition as an argument.
- ❖ Where clause executes Row-By-Row.
- ❖ Where clause after the execution of from clause.
- ❖ We can pass multiple condition for where clause using logical operators
- ❖ We cannot able to use ALIAS name in where clause

**SYNTAX: -**

SELECT \*/[DISTINCT]COLUMN\_NAME/EXPRESSION[ALIAS]FROM<TABLE-  
\_NAME]

Where <FILTER CONDITION>

❖ THE MAIN CONDITION OF WHERE CLAUSE IS REMOVES ALL FALSE VALUES.

**ORDER OF EXECUTION**

1. FROM CLAUSE
2. WHERE CLAUSE
3. SELECT CLAUSE

**Q. WAQTD EMPNAME, SAL GIVEN TO SUNDRI?**

**A)**

```
SELECT ENAME,SAL  
FROM EMP  
WHERE ENAME='SUNDRI';
```

**Q. WAQTD DETAILS OF AN EMP IF THE EMP NAME IS KING?**

**Q. WAQTD NAME, SAL GIVEN TO ALL EMP WHO ARE GETTING SAL MORE THAN 1200?**

**Q. WAQTD NAME, SAL GIVEN TO ALL EMP WHO ARE GETTING SAL LESS THAN 1200?**

**WAQTD NAME & JOB OF ALL EMP IF THEY ARE WORKING AS SALESMAN?**

**WAQTD ALL THE DETAILS OF AN EMP WHO ARE WORKING DEPTNO 30?**

**WAQTD ENAME FROM EMP TABLE IF THEY ARE WORKING IN DEPT NO 20?**

**WAQTD SAL OF ALL THE EMP?**

**WAQTD SAL OF EMP WHOSE NAME IS SMITH?**

**ASSIGNMENT – I**

**Q. WAQTD EMPNAME, EMPNO, JOB ALONG WITH DEPTNO IF EMPNO IS 7788?**

**Q. WAQTD DETAILS OF EMP IF THEY ARE HIRED AFTER 81?**

**Q. WAQTD NAME & HIREDATE IF EMP HIRED BEFORE 86?**

**Q. WAQTD DETAILS OF EMP WHO ARE NOT WORKING AS ANALYST?**

**Q. WAQTD DETAILS OF EMP IF THEY ARE WORKING AS CLERK IN DEPTNO 20?**



## **OPERATORS: -**

There are seven types of Operators.

1. **Athematic Operator** (+, -, \*, /)
2. **Comparison Operator** (=, != or < >)
3. **Relational Operator** (>, <, >=, <=)
4. **Logical Operator** (and, or, not)
5. **Concatenation Operator** (||)
6. **Special Operator** (in, notin, between, not between, is, isnot, like, notlike)
7. **Subquery Operator** (all, any, exist, not exist)

## **AND: -**

And operators return true if both the condition are true.

## **OR: -**

OR operators return true when any one of the conditions is true.

AND	O/P
1*1	1
1*0	0
0*1	0
0*0	0

OR	O/P
1+1	1
1+0	1
0+1	1
0+0	0

**Q. WAQTD DETAILS OF AN EMP WHO ARE NOT WORKING AS SALESMAN OR MANAGER?**

**Q. WAQTD DETAILS OF AN EMP WHO ARE WORKING AS SALESMAN OR MANAGER?**

**Q. WAQTD ENAME, AND DEPTNO IF THEY ARE WORKING IN DEPTNO 10 OR 30 AS A PRESIDENT?**

**Q. WAQTD SAL GIVEN TO ALL THE EMPLOYEE WHO ARE GETTING SAL MORE THAN 1250 & LESS THAN 5000?**

**Q. WAQTD DETAILS OF AN EMP WHO ARE GETTING COMM 0 OR 500 IN DEPTNO 30 OR 20?**

**Q. WAQTD DETAILS OF AN EMP EXCEPT PRESIDENT IF THEY ARE HIRED AFTER 82?**

## **IN OPERATOR: -**

It is multivalued operator which can accept multiple values at RHS & single value at LHS.

## **SYNTAX: -**

COLUMN\_NAME/EXPRESSION IN (V1,V2,.....VN)

**Q. WAQTD NAME, JOB & DEPTNO OF ALL THE MEP WHO ARE WORKING AS ANALYST OR CLERK OR MANAGER IN DEPTNO 10 OR 20 OR 40?**

**Q. WAQTD DETAILS OF EMP IF THEY ARE WORKING IN DEPTNO 10 OR 20 OR 30 OR 40?**

**Q. WAQTD NAME, JOB, SAL & DEPTNO OF AN EMP IF THEY ARE GETTING SAL 1250 OR 3000 OR 5000 IN DEPTNO 10 OR 30 EXCEPT MANAGER, PRESIDENT? (SAME QUE IN NOT IN)**

**NOT IN OPERATOR : -**

It is multivalued operator which can accept multiple values at RHS & single value at LHS.

**SYNTAX : -**

COLUMN\_NAME/EXPRESSION NOT IN (V1,V2,.....VN)

**Q. WAQTD DETAILS OF AN EMPLOYEE WHO ARE WORKING IN DEPTNO 20 OR 30 EXCEPT WHO ARE GETTING SAL 800,3000,1250?**

**Q. WAQTD SAL GIVEN TO ALL EMP IF THEY ARE GETTING SAL MORE THAN OR EQUAL TO 1250 BUT LESS THAN OR EQUAL TO 5000?**

**BETWEEN OPERATOR**

Whenever we have range of values then we can go for between operator.

➔ Between operator includes the ranges.

**SYNTAX : -**

COLUMN\_NAME/EXPRESSION BETWEEN LOWER\_RANGE AND HIGHER\_RANGE

**Q. WAQTD SAL GIVEN TO ALL EMP IF THEY ARE GETTING SAL MORE THAN OR EQUAL TO 1250 BUT LESS THAN OR EQUAL TO 5000? (USING BETWEEN OPERATOR)**

**Q. WAQTD COMM GIVEN TO SALESMAN OR ANALYST IF THEY ARE GETTING COMM BETWEEN 300 TO 400?**

**Q. WAQTD DETAILS OF EMPLOYEE EXCEPT WHO ARE GETTING SAL 1500-3000?**

**NOT BETWEEN: -**

Whenever we had excluded the range of values we can go for not between operators.

➔ Not between operators works by excluding the ranges.

**SYNTAX: -**

COLUMN\_NAME/EXPRESSION NOT BETWEEN LOWER RANGE AND HIGHER-NAME.

**Q. WAQTD SAL GIVEN TO ALL THE EMP WHO ARE GETTING SAL LESS THAN 1250 OR MORE THAN 3000?**

**Q. WAQTD EMPNAME AND COMM GIVEN TO ALL EMP WHO ARE GETTING COMM MORE THAN 0 & LESS THAN OR EQUAL TO 1400?**

### **IS OPERATOR: -**

It is used to check the value present in LHS is NULL/NOT NULL.

### **SYNTAX :-**

COLUMN\_NAME/EXPRESSION IS NULL/NOTNULL

**Q. WAQTD DETAIL FO EMP IF THEY ARE GETTING COMM?**

**A) SELECT \***

**FROM EMP**

**WHERE COMM>0; OR WHERE COMM IS NOT NULL;**

**Q. WAQTD NAME, JOB AND COMM OF ALL THE EMP WHO ARE WORKING AS SALESMAN OR ANALYST AND THEY ARE GETTING COMM?**

**Q. WAQTD DETAILS OF AN EMP IF THE EMPNAME IS SMITH?**

### **LIKE OPERATORS: -**

Like operator is used to perform pattern matching.

### **SYNTAX:-**

COLUMN\_NAME/EXPRESSION LIKE 'PATTERN-TO-MATCHING';

To perform pattern matching we have two special characters.

1. PERCENTAGE (%)
2. UNDERSCORE (\_)

### **PERCENTAGE (%): -**

It can accept any character any no of times ranges 0 to n.

### **UNDERSCORE:-**

It can accept any character only once.

E.G:-

<b>TEJASHREE</b>	<b>LIKE '%A';</b>	<b>//APOORVA</b>
<b>NAVIN</b>	<b>LIKE 'A%';</b>	<b>//ANKIT, APOORVA</b>
<b>PALLAVI</b>	<b>LIKE '%A%';</b>	<b>//TEJASHREE, NAVIN, PALLAVI, JANANI, ANKIT, APOORVA</b>
<b>JANANI</b>	<b>LIKE '%A%A%';</b>	<b>//APOORVA, PALLAVI, JANANI</b>
<b>ANKIT</b>	<b>LIKE '%EE%';</b>	<b>//TEJASHREE</b>
<b>APOORVA</b>	<b>LIKE '%I_';</b>	<b>//ANKIT, NAVIN</b>
<b>DILEEP</b>	<b>LIKE '%A_A%';</b>	<b>//DILEEP</b>

**Q. WAQTD DETAILS OF EMP IF EMPNAME STARTS WITH S?**

**Q. WAQTD DETAILS OF AN EMP IF EMPNAME ENDS WITH R?**

**Q. WAQTD DETAILS OF AN EMP IF EMP NAME CONTAINS A IN IT?**

**Q. WAQTD NAME, JOB OF AN EMP IF ENAME STARTS WITH M OR B?**

**Q. WAQTD DETAILS OF EMP IF THEY ARE WORKING AS SALESMAN OR MANAGER & GETTING 3 DIGIT COMM?**

**Q. WAQTD DETAILS OF AN EMP WHO ARE HIRED DURING 81?**

**ESCAPE:** -

It is used to remove the special behavior of special character and treat it as normal char which is present next to it. Only once.

**Q. WAQTD DETAILS OF EMP IF EMPNAME CONTAINS % IN IT?**

A) SELECT \*

FROM EMP

WHERE ENAME LIKE '%! % %' ESCAPE '!';

❖ ' % ! \_ % ' = 'UNDERSCORE'

❖ ' % ! % % ' = 'PERCENTAGE'

**CONCATINATION OPERATOR:** -

To join strings.

SYNTAX: CONCAT('STR','STR')

E.g.:-

MR SMITH YOUR SAL IS 800 RS

SELECT CONCAT('Mr',CONCAT(ENAME,CONCAT('YOUR SALARY IS',CONCAT(SAL,'RS'))))

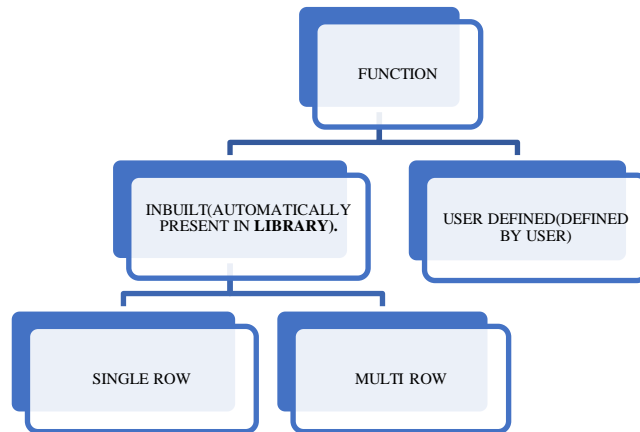
FROM EMP;

**SELECT 'Mr' || ENAME||'YOUR SALARY IS' || SAL|| 'Rs'**

**FROM EMP;**

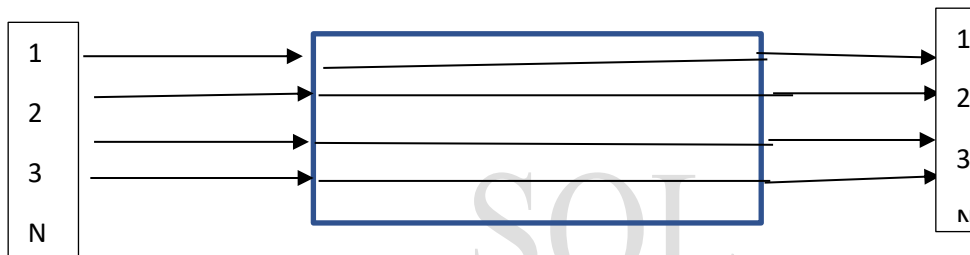
**FUNCTIONS:** -

It is the list of instructions or block of codes which is used to perform task.



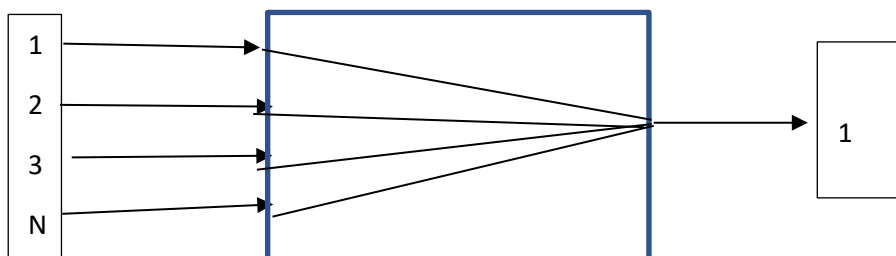
### **SINGLE ROW FUNCTIONS: -**

- It executes row-by-row.
- It takes one input executes and generates one O/p then goes to the next input.
- If we pass 'n' no of inputs to single row function it returns 'n' no of outputs.



### **MULTI ROW FUNCTIONS**

- It is also known as Group function, Aggregate function.
- It executes group by group.
- It takes all the inputs at once aggregate it and generates the output.
- If we pass 'n' no of inputs to multi row function, It returns a single output.



## **LIST OF MULTI ROW FUNCTIONS/GROUP/AGGRIGATE FUNCTIONS: -**

### **1. MAX()**

It is used to obtain max value from given table.

#### **SYNTAX**

**MAX(COLUMN\_NAME/EXPRESSION)**

### **2. MIN()**

It is used to obtain min value from given table.

#### **SYNTAX**

**MIN(COLUMN\_NAME/EXPRESSION)**

### **3. SUM()**

It is used to obtain total value from given table.

#### **SYNTAX**

**SUM(COLUMN\_NAME/EXPRESSION)**

### **4. AVG()**

It is used to obtain average value from given table.

#### **SYNTAX**

**AVG(COLUMN\_NAME/EXPRESSION)**

### **5. COUNT()**

It is used to obtain number of values from given table.

#### **SYNTAX**

**COUNT(COLUMN\_NAME/EXPRESSION)**

## **NOTE: -**

- Multi row function can accept only a single argument that is a column name or an expression.
- MAX () and MIN| () functions can be used for all the following datatypes problem.

i.e., CHAR, VARCHAR, NUMBER AND DATE.

- SUM () AND AVG () functions can only take number column of an argument.
- Multi-row functions will ignore the null value.

ENAME	SAL	COMM
CHARN	200	20
BHARAT	400	NULL
SHIVAM	600	40

**SELECT SUM(COMM)**

**FROM EMP;**

- We cannot use multirow functions in where clause.
- We cannot use any column name with multirow functions in select clause.
- COUNT () is the only MRF TO which we can pass '\*' as an argument.

**Q. WAQTD MAX SAL ALONG WITH MAX COMM GIVEN TO AN EMP?**

**Q. WAQTD TOTAL SAL GIVENT TO ALL THE SALESMAN FROM EMP?**

**Q. WAQTD MAX SAL OF ALL EMP EXCEPT PRESIDENT?**

**Q. WAQTD NO OF EMP WORKING AS MANAGER?**

**Q. WAQTD NO OF EMP WHO ARE NOT WORKING AS SALESMAN OR ANALYST AND DEPTNO NOT BELONG TO 10 OR 20?**

**A) SELECT COUNT (\*)**

**FROM EMP**

**WHERE JOB NOT IN ('SALESMAN','ANALYST') AND DEPTNO (10,20);**

**Q. WAQTD HIREDATE OF FIRST HIRED EMP FROM EMP?**

**A)**

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

**Q. WAQTD RECENTLY/LASTHIRED EMP FROM EMP TABLE?**

**Q. WAQTD NO OF DEPTNO PRESENT IN EMP TABLE?**

**SINGLE ROW FUNCTION: -**

We can nest single row function using **DUAL**.

**DUAL: -**

Dummy table which is used to get output.

**LOWER: -**

To convert given string into lower case.

**SYNTAX: -**

**LOWER(STR);**

**E.g. SELECT LOWER('JSPIDERS')**

**FROM DUAL;**

**UPPER: -**

Convert given string into upper case.

**SYNTAX: -**

```
SELECT UPPER('JSPIDERS')
FROM DUAL;
```

**INITCAP**

To convert first letter of a word into appreciate.

**SYNTAX: -**

```
INIT CAP(STR)
SELECT INITCAP ('CHARAN IS A GOOD BOY')
FROM DUAL;
```

**LENGTH: -**

To obtain the length of the string.

**SYNTAX: -**

```
LENGTH ('STR')
```

E.g

```
SELECT LEGTH('JSPIDERS')
FROM DUAL;
```

**SUBSTRING**

It is used to obtain part of a string from original string.

**SYNTAX: -**

```
SUBSTR('ORIGINAL_STR', LENGTH,[LENGTH])
```

[] ➔ Anything which is written in square brackets is **OPTIONAL**.

E.g.: -

SUBSTR ('BANGALORE',2,4)		ANGA
SUBSTR ('BANGALORE',4,1)		G
SUBSTR ('BANGALORE',10,2)		NULL
SUBSTR ('BANGALORE',6)		LORE
SUBSTR ('BANGALORE',7,5)		ORE
SUBSTR ('BANGALORE',0,3)		BAN
SUBSTR ('BANGALORE',-3,2)		OR
SUBSTR ('BANGALORE',-6)		GA LORE
SUBSTR ('BANGALORE',-1)		E



SUBSTR ('BANGALORE',-7,3)	NGALO
SUBSTR ('BANGALORE',-1,2)	E

**Q. WAQTD NAME OF EMP IF THE EMP NAME START WITH 'M'?/BY USING SUBTRING? A (WHERE SUBSTR(ENAME,1,1)='M');**

**Q. WAQTD DETAILS OF EMP IF EMPNAME ENDS WITH R?**

**A) WHERE SUBSTR(ENAME,-1,1)='R';**

**Q. WAQTD NAME & JOB OF EMP IF JOB STARTING WITH MAN OR SAL?**

**A)WHERE SUBSTR(JOB,1,3) IN('MAN','SAL');**

**Q. WAQTD FIRST HALF OF EMPNAME IN LOWER CASE?**

**A)**

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

**Q. WAQTD SECOND HALF OF EMPNAME IN LOWER CASE?**

**A)**

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

**Q. WAQTD FIRST HALF OF THE EMPNAME IN LOWER CASE & 2<sup>ND</sup> HALF OF THE EMPNAME IN UPPER CASE?**

**A)**

```
SELECT
ENAME,LOWER(SUBSTR(ENAME,1,(LENGTH(ENAME)/2)))||UPPER(SUBSTR(EN
AME,LENGTH(ENAME)/2+1)
FROM EMP;
```

**REPLACE: -**

It is used to replace the substring from a given string in original string

**SYNTAX: - REPLACE('ORIGINAL\_STR','SUB\_STR','NEW\_STR');**

**E.g.**

**SELECT REPLACE('JSPIDERS','J','Q')**

**FROM DUAL;**

**O/P: Q SPIDERS.**

**SELECT REPLACE ('DANGER','A')**

**FROM DUAL;**

**O/P; 'DNGER'**

**Q. WAQTD DETAILS OF EMP IF EMPNAME NOT START WITH M?**

**A) WHERE SUBSTR(ENAME,1,1)!='M';**

**Q. WAQTD DETAILS OF EMP IF EMP CONTAIN 6 CHAR & GETTING 3 DIGIT SALARY?**

**Q. WAQTD DETAILS OF EMP IF EMPNAME CONTAINS EXACTLY 1A?**

### **INSTR**

It is used to obtain index value of substring which is present in original string.

SYNTAX: -

INSTR ('ORIGINAL-STR','SUB\_STR', POSITION, [Nth Occurrence]);

❖ If we are not giving any occurrence means it takes 1<sup>st</sup> occurrence only.

INSTR('MALAYALAM','A',1)		2
INSTR('MALAYALAM','L',3)	1 2 3 4 5 6 7 8 9	3
INSTR('MALAYALAM','A',1,2)	M A L A Y A L A M	4
INSTR('MALAYALAM','L',1,2)		7
INSTR('MALAYALAM','A',1,3)		6

**NOTE: -**

❖ IN INSTR WE ARE COMPARING WITH >0 OR =0.

**Q. WAQTD DETAILS OF EMP IF ENAME CONTAINS ATLEAST 1A?**

**A) WHERE INSTR(ENAME,'A',1)>0;**

**Q. WAQTD DETAILS OF EMP IF EMP NAME DOES NOT CONTAIN A?**

**A) WHERE INSTR(ENAME,'A',1)=0;**

**Q. WAQTD DETAILS OF EMP IF ENAME CONTAINS 2A?**

**A) WHERE INSTR(ENAME,'A',2)>0;**

### **SYSDATE**

To obtain the current date of a system

SELECTT SYSDATE

FROM DUAL;

**CURRENT DATE: -**

```
SELECT CURRENT_DATE  
FROM DUAL;
```

**SYSTIMESTAMP: -**

```
SELECT SYSTIMESTAMP  
FROM DUAL;
```

**TO\_CHAR ()**

This function is used to convert the given date to string format.

**SYNTAX**

TO\_CHAR (DATE, 'FORMAT\_MODELS')

**FORMAT MODELS**

1. YEAR
2. YYYY
3. YY
4. MONTH
5. MON
6. MM
7. DAY
8. DY
9. DD
10. D
11. HH24
12. HH12
13. MI
14. SS

SQL

DISPLAY THE TIME

HH12:MI: SS

**Q. WAQTD DETAILS OF EMP IF EMP HIRED ON DEC?**

**A)**

```
SELECT *  
FROM EMP  
WHERE TO_CHAR(HIREDATE, 'MON')='DEC';
```

**Q. WAQTD DETAILS OF EMP WHO ARE HIRED DURING 81?**

**Q. WAQTD NAME & HIREDATE OF EMP IF EMP HIRED ON FEB,NOV,DEC?**

**Q. WAQTD DETAILS OF EMP WHO ARE HIRED ON 22 & 17?**

**Q. WAQTD DETAILS OF EMP IF EMPNAME NOT START WITH A & THEY ARE HIRED ON SATUARDAY, FRIDAY & SUNDAY?**

**NVL (); [NULL VLAUE LOGIC]**

It is used to overcome null problem.

**SYNTAX: -**

NVL (ARG1, ARG2)

IT CAN ACCEPT 2 ARGUMENTS.

- In ARG1 we must write a column name or expression that can be null.
- In ARG2 we must write a value that can be substituted in place of null.
- In arg1 is not null, NVL returns same value present in ARG1.

ENAME	SAL	COMM
CHINNA	200	40
CHAITU	400	NULL
ICECREAM	600	20

SELECT SAL+COMM

FROM EMP;

**O/P: -**

240

ERROR

620.

**IN NVL**

**SELECT NVL(SAL,0) +NVL(COMM,0)**

**FROM EMP;**

**O/P: -**

**200+40 = 240**

**400+~~NULL~~ 0 = 400**

**600+20 = 620**

SQL

### **GROUP BY CLAUSE: -**

- We use group by clause to group the records.
- **IT EXECUTES ROW-BY-ROW.**
- For group by clause, we can pass column name or an expression as an argument
- .....
- We can write Group\_By\_Expression.
- After the execution of group by clause it create groups and if any clause executes after group by clause it executes group\_by\_group.

### **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 [GROLUP-BY-GROUP]

**Q. WAQTD MAX SAL GIVEN TO ALL EMP IN EACH DEPT?**

A)

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

**Q. WAQTD MIN SAL GIVEN TO ALL EMP IN EACH DEPT**

A)

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

❖ **IN QUESTION IF THEY ASK EACH MEANS, WE SHOULD USE GROUP BY.**

**Q. WAQTD TOTAL SAL NEED TO PAY FOR ALL THE EMP IN EACH JOB?**

A)

```
SELECT SUM(SAL)
FROM EMP
WHERE GROUP BY JOB;
```

## ASSIGNMEN QUE

**Q. WAQTD NO OF EMP WORKING AS MANAGER IN EACH DEPT?**

A)

```
SELECT COUNT(JOB)
FROM EMP
WHERE JOB='MANAGER'
GROUP BY JOB;
```

**Q. WAQTD AVG SAL AND TOTAL SAL NEED TO PAY FOR ALL EMP EXCEPT MANAGER IN EACH DEPT?**

A)

```
SELECT AVG(SAL), SUM(SAL)
FROM EMP
WHERE JOB != MANAGER
GROUP BY DEPTNO;
```

**Q. WAQTD NO OF EMP WORKING IN EACH DEPT IF THEY ARE GETTING SAL MORE THAN 2000?**

A)

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

### **HAVING CLAUSE: -**

- We use having clause to filter the groups.
- We can pass multi-row function condition in having clause
- It executes group by group.
- If Ur using having clause it should be used after group by clause.
- It cannot be used without group by clause.

### **SYNTAX: -**

SELECT GROUP\_BY\_EXPRESSION/GROUP\_FUNCTION

FROM TABLE\_NAME

[WHERE <FILTER\_CONDITION>]

GROUP BY COLUMN\_NAME/EXPRESSION

HAVING<GROUP\_FILTER\_CONDITION>

GROUP\_BY\_EXPRESSION CONDITION/MULTI\_ROW\_FUNCTION CONDITION

**ORDER OF EXECUTION: -**

1. FROM
2. WHERE(IF-USED) [ROW-BY-ROW]
3. GROUP BY [ROW-BY-ROW]
4. HAVING [ GROUP\_BY\_GROUP]
5. SELECT [GROUP\_BY\_GROUP]

**Q. WAQTD FIND MAX SAL, DEPTNO ALONG WITH IN EACH DEP WHO ARE GETTING SAL MORE THA 400?**

**A)**

```
SELECT MAX(SAL), DEPTNO
FROM EMP
HAVING SAL>400;
```

**Q. WAQTD MIN SAL GIVEN TO ALL THE MANAGER IF THEYARE GETTING MIN SAL MORE THAN 500 IN EACH DEPT?**

**A)**

```
SELECT MIN(SAL), DEPTNO
FROM EMP
WHERE JOB='MANAGER'
GROUP BY DEPTNO
HAVING MIN(SAL)>500;
```

SQL

**Q. WAQTD NOT OF EMP WORKING IN EACH JOB IN WHICH ATLEAST 2 EMP ARE WORKING?**

**A)**

```
SELECT COUNT (*)
FROM EMP
GROUP BY JOB
HAVING COUNT(JOB)>2;
```

**Q. WAQTD REPEATED SALARAY OR DUPLICATE SALARY?**

**A)**

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

**Q. WAQTD NO OF EMP HIRED ON SAMEDATE?**

A)

**Q. WAQTD NO OF EMP HIRED ON SAME DAY 1 TO SAME DEPTNO?**

A)

**ORDER BY: -**

- It is used to sort the records in ascending or descending order.
- Order by clause must be written as last clause in statement
- Order by clause executes after the select clause.
- By default, order by clause sort the records in ascending order
- We can pass column name or expression as an argument in order by clause.
- We can pass alias name in order by clause.

**SYNTAX: -**

```
SELECT GROUP_BY_EXPRESSION/GROUP_FUNC  
FROM TABLE_NAME  
[WHERE<FILTER_CONDITION>]  
[GROUP BY COLUMN_NAME/EXPRESSION]  
[HAVING <GROUP_FILTER CONDITION>]  
ORDER BY COLUMN_NAME[ASC]/DESC;
```

**ORDER BY 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

ENAME	SAL
A	100
B	300
C	200
D	800
E	600



SELECT SAL FROM EMP ORDER BY SAL;	SELECT SAL AS SALARY FROM EMP ORDER BY SALARY;	SELECT SAL FROM EMP ORDER BY SAL ASC;	SELECT SAL FROM EMP ORDER BY SAL DESC;
---	--	--	---

### Q. WAQTD SAL FROM EMP TABLE IN DESC ORDER?

A)

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

### Q. WAQTD DEPTNO OF EMP IN ASC?

A)

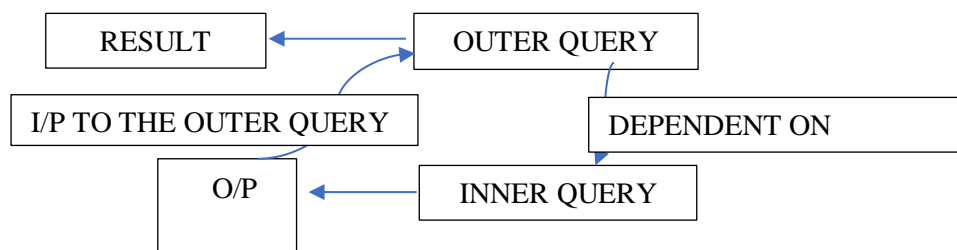
```
SELECT SAL
FROM EMP
ORDER BY SAL ASC;
```

### SUB -QUERY: -

1. A query which is written inside another query is known as subquery.

#### Working Procedure

- Outer Query
  - Inner Query/sub-query
2. Inner query will execute first and generated the O/P.
  3. The O/p generated by the inner query will be given as I/p to outer query.
  4. The outer query will execute and generate the output.
  5. This output will be result.
  6. By this we can say the outer query is dependent on inner query.



Why and where we need to go for subquery.

CASE: - 1

When we come across unknowns value we need to prefer subquery.

**CONDITIONS/RULES TO WRITE SUBQUERY: -**

1. We can select any one column name in inner query.
2. The column selected in inner query and the column written in outer query must have some data type.

Where comm> (select deptno);



NUM



NUM

Where comm> (select ename);



NUM



VARCHAR

**Q. WAQTD DETAILS OF EMP IF EMP GETS SAL MORE THAN SUNDRI SAL?**

**A)**

```
SELECT *  
FROM EMP  
WHERE SAL>(SELECT SAL  
FROM EMP  
WHERE ENAME='SUNDRI');
```

**Q. WAQTD NAME OF EMP IF EMP GETTING SAL MORE THAN BLANKE SAL?**

**A)**

```
SELECT *  
FROM EMP  
WHERE HIREDATE> (SELECT HIREDATE  
FROM EMP  
WHERE ENAME='MILLER');
```

**Q. WAQTD NAME & JOB OF EMP IF THE EMP WORKING IN SAME JOB AS ALLEN?**

**A)**

```
SELECT ENAME, JOB  
FROM EMP  
WHERE JOB= (SELECT JOB  
FROM EMP  
WHERE ENAME='ALLEN');
```

**Q. WAQTD DETAILS OF EMP IF THE EMP WORKING SAME DEPT AS TURNER DEPT?**

**A)**

```
SELECT *  
FROM EMP  
WHERE DEPTNO=(SELECT DEPTNO  
FROM EMP  
WHERE ENAME='TURNER');
```

**Q. WAQTD DETAILS OF EMP IF THE EMP WORKING AS SALESMAN & GETTING SAL MORE THAN TURNER SAL?**

**A)**

```
SELECT *  
FROM EMP  
WHERE JOB='SALESMAN' AND SAL>(SELECT SAL  
FROM EMP  
WHERE ENAME='TURNER');
```

**Q. WAQTD ENAME, JOB, COMM OF ALL THE EMP IF THEY ARE GETTING COMM AND THEY ARE WORKING AS SAME DEPT AS TURNER DEPT?**

**A)**

```
SELECT ENAME, JOB, COMM  
FROM EMP  
WHERE COMM IS NOT NULL AND DEPTNO= (SELECT DEPTNO  
FROM EMP  
WHERE ENAME='TURNER');
```

**Q. WAQTD DETAILS OF EMP WHO ARE NOT WORKING AS MANAGER AND THEY ARE GETTING SAL LESS THAN 5000?**

**A)**

```
SELECT *  
FROM EMP  
WHERE (JOB! ='MANAGER') AND (SAL)<5000;
```

**CASE: - 2 FOR SUBQUERY: -**

Whenever the data to be selected and the condition to be executed are present in different, tables we use sub query.

ID	NAME	DNO
1	SUNDRI	20
2	MACHA	10
3	SUNDRA	30
4	MACHI	10

DNO	DNAME	LOC
10	D1	X
20	D2	Y
30	D3	Z

**Q. WAQTD DEPTNAME OF SUNDRA?**

A)

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

**Q. WAQTD DEPT NAME OF MILLER?**

A)

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

**Q. WAQTD LOC OF MILLER?**

A)

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

**Q. WAQTD NO OF EMP WORKING IN SALES DEPT?**

A)

```
SELECT COUNT (*)
FROM EMP
WHERE DEPTNO=(SELECT DEPTNO
FROM DEPT
WHERE DNAME='SALES');
```

## ASSIGNMENT

**Q. WAQTD NAME OF EMP WHO ARE WORKING IN CHICAGO?**

A)

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

**Q. WAQTD DEPTNO, LOC OF EMP IF THE ENAME IS KING?**

A)

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

**Q. WAQTD DETAILS OF EMP IF THEY ARE WORKING IN ACCOUNTING DEPT?**

A)

```
SELECT *
FROM EMP
WHERE DEPTNO IN (SELECT DEPTNO
FROM DEPT
WHERE DNAME='ACCOUNTING');
```

**Q. WAQTD DETAILS OF EMP IF THEY ARE WORKING IN ACCOUNTING AND SALES DEPT?**

A)

```
SELECT *
FROM EMP
WHERE DEPTNO IN (SELECT DEPTNO
FROM DEPT
WHERE DNAME IN('SALES','ACCOUNTING'));
```

**Q. WAQTD NAME OF EMP WHO ARE WORKING IN CHICAGO & DALLAS?**

A)

```
SELECT ENAME
FROM EMP
WHERE DEPTNO IN (SELECT DEPTNO
FROM DEPT
WHERE LOC IN('CHICAGO','DALLAS'));
```

**Q. WAQTD NAME, JOB, SAL, GIVEN TO ALL THE EMP IF THEY ARE WORKING AS SALES DEPT EXCEPT PRESIDENT?**

**A)**

```
SELECT ENAME, JOB, SAL
FROM EMP
WHERE JOB <> 'PRESIDENT' AND DEPTNO IN (SELECT DEPTNO FROM DEPT
WHERE DNAME = 'SALES');
```

**Q. WAQTD DETAILS OF EMP WHO ARE GETTING SAL MORE THAN SMITH?**

**A)**

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

**Q. WAQTD DEPTNAME OF EMP IF THEY ARE WORKING AS SALESMAN & GETTING SAL MORE THEN SMITH SALARY?**

**A)**

```
SELECT DNAME
FROM DEPT
WHERE DEPTNO IN (SELECT DEPTNO
FROM EMP
WHERE JOB = 'SALESMAN' AND SAL > (SELECT SAL
FROM EMP
WHERE ENAME = 'SMITH'));
```

**Q. WAQTD DETAILS OF EMP WHO ARE WORKING IN SALES DEPT & THEY ARE GETTING SAL LESS THAN KING?**

**A)**

```
SELECT *
FROM EMP
WHERE DEPTNO = (SELECT DEPTNO
FROM DEPT
WHERE DNAME = 'SALES' AND SAL < (SELECT SAL
FROM EMP
WHERE ENAME = 'KING'));
```

**Q. WAQTD 2<sup>ND</sup> MAX SAL?**

**A)**

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

**Q. WAQTD 3<sup>RD</sup> MIN SAL?**

A)

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

**NESTED SUBQUERY: -**

- A subquery written inside another subquery is known as nested subquery.
- We can nest about 255 subqueries.
- To find nth max or min, sal, we will write(N-1) sub-queries.

**EMPLOYEE MANAGER RELATIONSHIP: -**

TYPE 1: - TO FIND MANAGER/REPORTING MANAGER

EMPNO	ENAME	MGR
1	A	2
2	B	4
3	C	NULL
4	D	1

**Q. WAQTD MANAGER NAME OF A?**

A)

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

**Q. WAQTD MANAGER NAME OF BLACK FROM EMP TABLE/**

A)

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

**Q. WAQTD DETAIL OF PRESIDENT MANAGER FROM EMP?**

A)

```
SELECT *  
FROM EMP  
WHERE EMPNO=(SELECT MGR  
FROM EMP  
WHERE JOB='PRESIDENT');
```

**Q. WAQTD LOC NAME OF CLERK MANAGER?**

A)

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

**Q. WAQTD DEPTNAME OF TURNER MANAGER?**

A)

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

**Q. WAQTD DETAILS OF SMITH MANAGER MANAGER?**

A)

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



**Q. WAQTD LOC NAME OF TURNER MANAGER MANAGER?**

A)

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

**TYPE – 2 TO FIND REPORTER/REPORTINGS**

EMPNO	ENAME	MGR
1	A	2
2	B	4
3	C	NULL
4	D	1

**Q. WAQTD NAME OF EMP WHO ARE REPORTING TO A?**

A)

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

**Q. WAQTD NO OF EMP REPORTING TO KING?**

A)

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

**Q. WAQTD DETAILS OF EMP WHO ARE REPORTING TO BLACK?**

A)

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

**ASSIGNMENT: -**

**Q. WAQTD NO OF EMP REPORTING TO ANALYST?**

**Q. WAQTD DETAILS OF EMP WHO ARE ACTING AS REPORTING MANAGER?**

**Q. WAQTD NAME OF EMP WHO ARE HAVING ATLEAST 2 REPORTING?**

**Q. WAQTD NAME OF EMP WHO ARE REPORTING AND INDIRECTLY OR DIRECTLY TO THE KING?**

**PSEUDO COLUMN: -**

- These 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.
- Types of pseudo columns
  1. ROWID
  2. ROWNUM

**1. ROWID: -**

It is a 18 digit address in which the record is present or the record is stored in the memory.

**SYNTAX:-**

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

**NOTE: -**

- ❖ ROWID is one of the ways to access/delete the record.
- ❖ ROWID is unique.
- ❖ ROWID is present for each & every record.
- ❖ ROWID is generated at the time of insertion of records.
- ❖ ROWID cannot be inserted ,updated/deleted.
- ❖ Empty table will not be having ROWID.
- ❖ ROWID is static in nature.(CONSTANT).
- ❖ ROWID can be used to identify a record uniquely from the table when there is no key attribute/primary key.

## 2. **ROWNUM**: -

It acts as serial number to the result table.

### **SYNTAX**: -

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

### **NOTE:-**

- ❖ It is used as record num that is assigned to the result table.
- ❖ It is dynamic in nature(keeps on changing).
- ❖ It is generated at the time of execution.
- ❖ It always starts with 1
- ❖ Cannot be duplicated.
- ❖ Gets incremented after it is assigned.

### **Q. WAQTD FIRST 3 RECORDS FROM EMP TABLE?**

A)

```
SELECT ROWNUM,EMP.*  
FROM EMP  
WHERE ROWNUM<4;
```

### **Q. WAQTD TOP 5 RECORDS?**

A)

```
SELECT ROWNUM,EMP.*  
FROM EMP  
WHERE ROWNUM<6;
```

### **Q. WAQTD TO 10 RECORDS?**

A)

```
SELECT ROWNUM,EMP.*  
FROM EMP  
WHERE ROWNUM<11;
```

### **Q. WAQTD 1<sup>ST</sup> RECORD FORM THE TABLE?**

A)

```
SELECT ROWNUM,EMP.*  
FROM EMP  
WHERE ROWNUM=1;
```

**Q. WAQTD 2<sup>ND</sup> RECORD FROM THE EMP TABLE?**

A)

```
SELECT *  
FROM (SELECT ROWNUM AS SLNO, EMP.*  
FROM EMP)  
WHERE SLNO=2;
```

**TO MAKE ROWNUM AS STATIC: -**

- Take a table and assign rownum to the given table.
- Change the rownum to any other name by using ALIAS(SLNO).
- Use this as a SUB-QUERY in from clause of outer query.
- In the outer query the alias name in the condition.

**Q. WAQTD NTH MAX AND NTH MIN USING ROWNUM?**

A)

```
SELECT SAL  
FROM (SELECT ROWNUM AS SLNO,SAL  
FROM(SELECT DISTINCT SAL  
FROM EMP  
ORDER BY SAL DESC))  
WHERE SLNO='N';
```

**Q. WAQTD 3<sup>RD</sup> / 8<sup>TH</sup> / 6<sup>TH</sup> MAX/MIN SAL?**

A)

```
SELECT SAL  
FROM (SELECT ROWNUM AS SLNO,SAL  
FROM(SELECT DISTINCT SAL  
FROM EMP  
ORDER BY SAL DESC/ASC))  
WHERE SLNO='3';
```

**JOINS: -**

This statement is used to retrieve the data from multiple table simultaneously.

**TYPES OF JOINS: -**

- CARTESIAN JOINS/CROSS JOIN
- INNER JOIN/EQUI JOIN
- OUTER JOIN
  1. LEFT OVER JOIN
  2. RIGHT OVER JOIN
  3. FULL OUTER JOIN
- SELF JOIN

➤ NATURAL JOIN

**CARTESIAN JOIN/CROSS JOIN:** -

- In certain join a record from table will be merged with all the records of table2.
- No of columns in result table will be summation of columns present in table1 & table 2.
- TOTAL NUM OF COLUMN = T1+T2.
- No fo records in result table will be the product of records present in table 1 and table 2.
- TOTAL NUM OF RECORDS = T1\*T2.
- .....
- IN THIS JOIN WE WILL BE GETTING ERROR RECORDS.

**SYNTAX:** -

```
1. ANSI:
SELECT COLUMN_NAME
FROM TABLE_NAME/CROSSJOIN TABLE_NAME2;

2. ORACLE:
SELECT COLUMN_NAME
FROM TABLE_NAME1,TABLE_NAME2;
```

**INNER JOIN:** -

We use inner join to obtain only the matched records or the records which has pair.

- .....
- We use join condition to obtain the matched records.
- .....

**SYNTAX:** -

```
1. ANSI:
SELECT COLUMN_NAME
FROM TABLE_NAME1 INNER JOIN TABLE_NAME2
ON<JOIN_CONDITION>;
```

**E.G.**

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

**SYNTAX:** -

```
2. ORACLE
SELECT COLUMN_NAME
FROM TABLE_NAME1,TABLE_NAME2
WHERE <JOIN_CONDITION>;
```

**E.G.**

## BOYS

BID	BNAME	GID
1	MUNN	2
2	RAJA	1
3	CHINNA	3

## GIRLS

GID	GNAME
1	RANI
2	MUNNI
3	CHINNI

### ORACLE

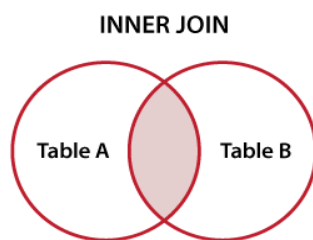
```
SELECT *  
FROM BOYS, GIRLS  
WHERE BOYS.GID=GIRLS.GID;
```

### ANSI

```
SELECT *  
FROM BOYS INNER JOIN GIRLS  
ON BOYS.GID=GIRLS.GID;
```

BID	BNAME	GID	GID	GNAME
1	MUNNA	2	2	MUNNI
2	RAJA	1	1	RANI
3	CHINNA	3	3	CHINNI

## VENN DIAGRAM



# SQL

### JOIN CONDITION: -

It is a condition on which we merge two table to get only the matched records.

### SYNTAX: -

TABLE_NAME 1.COL_NAME=TABLE_NAME 2.COL_NAME;
---

E.G.

EMP.DNO=DEPT.DNO
------------------

**Q. WAQTD EMPNAME & THEIR LOC IF THEY ARE WORKING AS SALESMAN USING JOINS?**

**A)**

**ORACLE**

```
SELECT ENAME, LOC  
FROM EMP, DEPT  
WHERE JOB='SALESMAN' AND EMP.DEPTNO=DEPT.DEPTNO;
```

**ANSI**

```
SELECT ENAME, LOC  
FROM EMP INNER JOIN DEPT ON EMP.DEPTNO=DEPT.DEPTNO  
WHERE JOB='SALESMAN';
```

**Q. WAQTD EMPNAME, JOB, SAL ALONG WITH LOC IF THE EMP WORKING IN 'CHICAGO' OR 'DALLAS' & THEIR SAL IS GREATER THAN 1250?**

**A)**

```
SELECT ENAME, JOB, SAL, LOC  
FROM EMP, DEPT WHERE EMP.DEPTNO=DEPT.DEPTNO AND LOC  
IN('CHICAGO','DALLAS') AND SAL>1250;
```

**NATURAL JOIN:** -

- In natural join we won't be writing any join condition.
- If the table contains similar columns, we get the O/P of inner join.
- If the table is not having similar columns, we will get the O/P of cartesian join.

**Q. WHY OR WHEN WE USE NATURAL JOIN?**

**A) Whenever there is no table structure, we use natural join.**

**COLUMNS THAT ARE PRESENT ----- TABLE STRUCTURE**

**SYNTAX:**

**ANSI:**

```
SELECT COLUMN_NAME  
FROM TABLE_NAME1 NATURAL JOIN TABLE_NAME2;
```

**E.G.**

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

**OR**

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

### **OUTER JOIN:** -

In outer join we get the unmatched records along with the matched records.

### **LEFT OUTER JOIN:** -

In left outer join we get unmatched records of left table along with matching records.

### **SYNTAX**

#### **LEFT OUTER JOIN:**

##### 1. ANSI

```
SELECT COLUMN_NAME  
FROM TABLE_NAME/LEFT [OUTER] JOIN ON <JOIN_CONDITION>;
```

##### **E.G.**

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

##### 2. ORACLE

```
SELECT COLUMN_NAME  
FROM TABLE_NAME1,TABLE_NAME2  
WHERE TABLE_NAME1.COL_NAME=TABLE_NAME2.COL_NAME(+);
```

##### **E.G.**

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

### **BOY**

BID	BNAME	GID
1	MUNNA	2
2	RAJA	1
3	CHINNA	3
4	KIDDO	NULL

### **GIRLS**

GID	GNAME
1	RANI
2	MUNNI
3	CHINNI
4	MANCHI

### **Q. WAQTD MATCHED & UNMATCHED RECORD FROM BOYS TABLE?**

#### **A)**

##### **ORACLE**

```
SELECT *  
FROM BOYS,GIRLS  
WHERE BOYS.GID=GIRLS.GID(+);
```

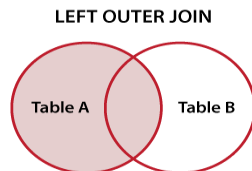
##### **ANSI**

```
SELECT *  
FROM BOYS LEFT OUTER JOIN GIRLS  
ON BOYS.GID=GIRLS.GID;
```



BID	BNAME	GID	GID	GNAME
1	MUNNA	2	2	MUNNI
2	RAJA	1	1	RANI
3	CHINNA	3	3	CHINNI
4	KIDDO	NULL	NULL	NULL

## VENN DIAGRAM FROM LEFT OUTER JOIN



## RIGHT OUTER JOIN

In right outer join we get unmatched records of right table along with matched records.

BID	BNAME	GID
1	MUNNA	2
2	RAJA	1
3	CHINNA	3
4	MACHA	NULL

GID	GNAME
1	RANI
2	MUNNI
3	CHINNI
4	MACHI

BID	BNAME	GID	GID	GNAME
1	MUNNA	2	2	MUNNI
2	RAJA	1	1	RANI
3	CHINNA	3	3	CHINNI
NULL	NULL	NULL	4	MACHI

MATCHED  
RECORD

UNMATCHED RECORDS

```
SELECT *
FROM BOYS,GIRLS
WHERE BOYS.GID(+)=GIRLS,GID;
```

## SYNTAX

### ANSI

```
SELECT COLUMN_NAME
FROM TABLE_NAME1 RIGHT[OUTER] JOIN TABLE_NAME2
ON <JOIN_CONDITION>;
```

### E.G.

```
SELECT *
FROM EMP E RIGHT OUTER JOIN DEPT D
ON E.DEPTNO.D.DEPTNO;
```

### ORACLE

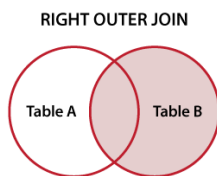
```
SELECT COLUMN_NAME
```

```
FROM TABLE_NAME1, TABLE_NAME2
WHERE TABLE_NAME1.COL_NAME(+) = TABLE_NAME2 , COL_NAME;
```

E.G.

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

## VENN DIAGRAM FOR RIGHT OUTER JOIN



## FULL OUTER JOIN

To obtain unmatched records of both the table along with matched records.

## SYNTAX

**ANSI**

```
SELECT COLUMN_NAME
FROM TABLE_NAME1 FULL[OUTER] JOIN TABLE_NAME 2 ON
<JOIN_CONDITION>;
```

E.G.

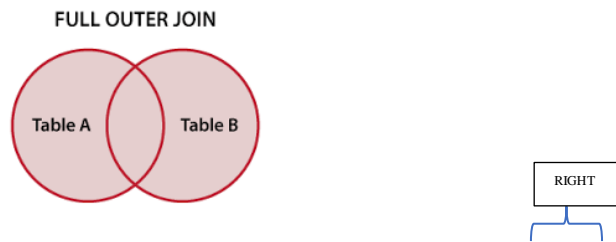
```
SELECT *
FROM EMP E DEPT D ON EMP E.DEPTNO=D.DEPTNO;
```

BID	BNAME	GID
1	MUNNA	2
2	RAJA	1
3	CHINNA	3
4	MACHA	NULL

GID	GNAME
1	RANI
2	MUNNI
3	CHINNI
4	MACHI

BID	BNAME	GID	GID	GNAME
1	MUNNA	2	2	MUNNI
2	RAJA	1	1	RANI
3	CHINNA	3	3	CHINNI
4	MACHA	NULL	NULL	NULL

## VENN DIAGRAM FOR FULL OUTER JOIN



**Q. WAQTD EMPNAME, DEPTNAME IF EMP WORKING IN SOME DEPT/NOT?**

**A)**

```
3 SELECT E.ENAME,D.DNAME
1 FROM EMP E,DEPT.D
2 WHERE E.DEPTNO=D.DEPTNO(+); OR
WHERE EMP.DEPTNO=DEPT.DEPTNO(+);
```

**Q. WAQTD EMPNAME, DEPTNAME IF DEPT HAVE SOME EMP WORKING IN DEPT OR NOT?**

**A)**

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

**Q. WAQTD DETAILS OF EMP ALONG WITH DEPTNAME IF EMP NOT WORKING IN ANY DEPT?**

**A)**

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

## **SELF JOIN**

Self join is used to join the same two table or the table itself.

**Q. WHY WE USE SELF JOIN?**

**A)** If the data to be selected and condition to be selected and condition to be executed is present in same table but in different record we use self-join.

## **SYNTAX**

**ANSI**

```
SELECT COLUMN_NAME
```

FROM TABLE\_NAME T1,JOIN TABLE\_NAME T2  
ON <JOIN\_CONDITION>;

**E.G.**

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

**ORACLE**

SELECT COLUMN\_NAME  
FROM TABLE\_NAME T1,TABLE\_NAME T2  
WHERE<JOIN\_CONDITION>;

**E.G.**

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

EMPNO	ENAME	MGR	EMPNO	ENAME	MGR
1	A	3	1	A	3
2	B	NULL	2	B	NULL
3	C	2	3	C	2
4	D	3	4	D	3

**EMP E1**

**EMP E2**

EMPNO	ENAME	MGR	EMPNO	ENAME	MGR
1	A	3	3	C	2
3	C	2	2	B	NULL
4	D	3	3	C	2

EMPNO

MGR NAME

**Q. WAQTD TO FIND THE MGR NAME OF A?**

**A)**

SELECT \*  
FROM EMP E1,EMP E2  
WHERE E1.MGR=E2.EMPNO AND E1.ENAME='A';

**Q. WAQTD EMPNAME, JOB AND HIS MANAGER'S NAME?**

A)

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

**NOTE: -**

- ❖ If we want to join 'n' no of table we will be writing (n-1) join condition.
- ❖ We can join upto 256 table using inner join.

**Q. WAQTD EMPNAME, JOB.SAL ALONG WITH MGR SAL IF MGR GETTING SAL MORE THAN 2999?**

A)

```
SELECT E1.ENAME,E1.JOB,E1.SAL,
E2.SAL
FROM EMP E1,EMP E2
WHERE E1.MGR=E2.EMPNO
AND E2.SAL>2999;
```

**Q. WAQTD EMPNAME & HIS MGR NAME(ALONG BOT EMP & MGR) WITH THEIR DEPT NAME?**

A)

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

**Q. WAQTD ENAME & MGR NAME ALONG WITH EMP DEPTNAME?**

A)

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

**Q. WAQTD ENAME,MGRNAME ALONG WITH THEIR DNAME IF EMP EARNING MORE THAN 2000 & MGR WORKING IN DEPT NO 20?**

A)

```
SELECT E1.ENAME,E2.ENAME,D1.DNAME,D2.DNAME
FROM EMP E1,EMP E2,DEPT D1,DEPT D2
WHERE E1.MGR=E2.EMPNO AND E1.DEPTNO=D1.DEPTNO AND
E2.DEPTNO=D2.DEPTNO AND E1.SAL>2000 AND E2.DEPTNO=20;
```

**Q. WAQTD EMP NAME,MGR NAME AND THEIR DEPTNAME IF EMP EARNING MORE THAN SMITH AND MANAGER EARNING THAN MILLER?**

A)

SELECT D1.DNAME E1.ENAME	E2.ENAME D2.DNAME
FROM DEPT D1 EMP E1	EMP E2 DEPT D2
E1.ENAME=D1.DNAME	E2.ENAME=D2.DEPTNO

```
SELECT E1.ENAME,E2.ENAME,D1.DNAME,D2.DNAME
FROM EMP E1,EMP E2,DEPT D1,DEPT D2
WHERE E1.MGR=E2.EMPNO AND E1.DEPTNO=D1.DEPTNO AND
E2.DEPTNO=D2.DEPTNO AND E1.SAL>(SELECT SAL FROM EMP WHERE
ENAME='SMITH') AND E2.SAL>(SELECT SAL FROM EMP WHERE
ENAME='MILLER');
```

**Q. WAQTD ENAME,MGRNAME AND THEIR LOC IF EMP WORKNG DEPTNO 10/30 AND MGR EARNING MORE THAN FORD AND EMP WORKING IN LOC NEWYORK OR CHICAGO?**

A)

```
SELECT E1.ENAME,E2.ENAME,D1.LOC,D2.LOC
FROM EMP E1,EMP E2,DEPT D1,DEPTD2
WHERE E1.MGR=E2.EMPNO AND E1.DEPTNO IN(10,20) D1.LOC
IN('NEWYORK','CHICAGO') AND E2.SAL>(SELECT SAL FROM EMP WHERE
ENAME='FORD');
```

## TABLE CREATION

### DATA DEFINATION LANGUAGE


#### L CREATE: -

#### SYNTAX

```
CREATE TABLE
    table
_name
(
    COLUMN_NAME_1 DATATYPE NOT NULL/[NULL],
    COLUMN_NAME_2 DATATYPE NOT NULL / [NULL],
    .
    .
    COLUMN_NAME_n    DATATYPE    NOT NULL / [ NULL],
    CONSTRAINT constraint_ref_name UNIQUE(COLUMN_NAME),
    CONSTRAINT constraint_ref_name CHECK(CONDITION),
    CONSTRAINT constraint_ref_name PRIMARY KEY(COLUMN_NAME),
    CONSTRAINT constraint_ref_name FOREIGN KEY(COLUMN_NAME)
    REFERENCES parent_table_name (COLUMN_NAME)
);
```

#### CUSTOMER TABLE

##### TOP DOWN APPROACH



COLUMN _NAME	CID	CNAME	PHONE
DATA TYPE	NUMBER(2)	VARCHAR(20)	NUMBER(10)
NULL/NO TNUL	NOT NULL	NOT NULL	NOT NULL
CONSTR AINTS	CID_PK (PRIMARY KEY)		PHNO UNIQUE(PHONE) CHECK(LENGTH(PHONE)=10)

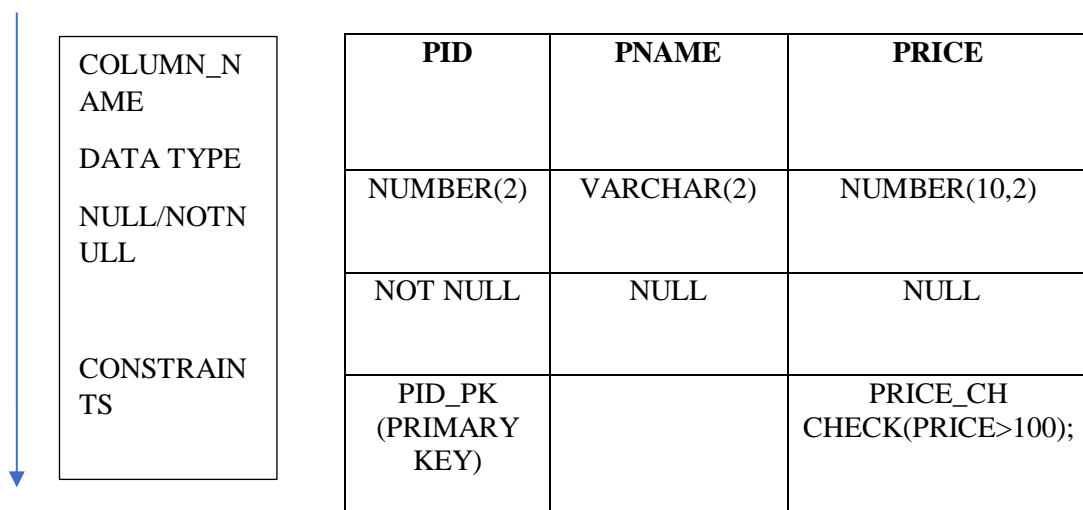
## CREATE TABLE

```
CREATE TABLE CUSTOMER
(
  CID NUMBER(2) NOT NULL,
  CNAME VARCHAR(20) NOT NULL,
  PHONE NUMBER(10) NOT NULL,
  CONSTRAINTS CID_PK PRIMARY KEY(CID),
  CONSTRAINTS PNO UNIQUE (PHONE),CHECK (LENGTH(PHONE)=10)
);
```

## CREATE TABLE OF PRODUCT

CUSTOMER TABLE

TOP DOWN APPROACH



COLUMN_NAME	PID	PNAME	PRICE
DATA TYPE	NUMBER(2)	VARCHAR(2)	NUMBER(10,2)
NULL/NOT NULL	NOT NULL	NULL	NULL
CONSTRAINTS	PID_PK (PRIMARY KEY)		PRICE_CH CHECK(PRICE>100);

## PRODUCT TABLE

```
CREATE TABLE PRODUCT
(
  PID NUMBER(2) NOT NULL,
  PNAME VARCHAR(20) NOT NULL,
  PRICE NUMBER(10,2) NULL,
  CONSTRAINTS PID_PK PRIMARY KEY(PID),
  CONSTRAINTS PRICE_CH CHECK(PRICE>100)
);
```



## **RENAME:** -

### **SYNTAX**

```
RENAME current_table_name TO New_table_name;
```

E.G.

```
RENAME PRODUCT TO PROD;
```

## **ALTER:** -

### **SYNTAX**

1. TO ADD A COL :

```
ALTER TABLE table_name  
ADD COLUMN_NAME DATATYPE[NULL/NOT NULL];
```

E.G.

```
ALTER TABLE CUSTOMER  
ADD PID NUMBER(2) NOT NULL;
```

2. TO DROP A COL :

```
ALTER TABLE table_name  
DROP COLUMN COLUMN_NAME ;
```

E.G.

```
ALTER TABLE CUSTOMER  
DROP COLUMN PID;
```

3. TO CHANGE THE  
DATATYPE:  
ALTER TABLE table\_name

```
MODIFY COLUMN_NAME  
new_datatype;
```

4. TO CHANGE THE NOT  
NULL CONSTRAINT:

```
ALTER TABLE table_name  
  
MODIFY COLUMN_NAME existing_datatype  
NULL/NOTNULL;
```

5. TO RENAME THE COLUMN:

```
ALTER TABLE table_name  
  
RENAME COLUMN current_name TO new_name;
```

E.G.

```
ALTER TABLE CUSTOMER  
RENAME COLUMN CID TO PID;
```

#### 6.TO MODIFY

##### CONSTRAINTS:

a) ALTER TABLE table\_name

```
ADD CONSTRAINT constraint_ref_name UNIQUE(column_name);
```

b) ALTER TABLE table\_name

```
ADD CONSTRAINT constraint_ref_name CHECK(condition);
```

c) ALTER TABLE table\_name

```
ADD CONSTRAINT constraint_ref_name PRIMARY KEY(column_name);
```

d) ALTER TABLE table\_name

```
ADD CONSTRAINT constraint_ref_name FOREIGN KEY(column_name) REFERENCES  
parent_table_name(column_name);
```

## **TRUNCATE**

### SYNTAX:

```
TRUNCATE TABLE table_name;
```

- Used to remove all the records without altering table structure/cannot get back.

## **DROP**

### SYNTAX

```
DROP TABLE table_name;
```

## **TO RECOVER THE TABLE (ONLY IN ORACLE): -**

### SYNTAX

```
FLASHBACK TABLE table_name  
TO BEFORE DROP  
[RENAME TO new_name];
```

## **TO DROP THE TABLE FROM RECYCLE BIN:-**

### SYNTAX

```
PURGE TABLE table_name;
```

- It is used to remove all the records along with table structure.
- It can be get back.

### **DELETE:** -

- To delete the particular records from the table.
- Can get by using ROLL BACK.

### **INTERVIEW QUESTIONS**

<b>TRUNCATE</b>	<b>DROP</b>	<b>DELETE</b>
<ul style="list-style-type: none"><li>➤ Used to remove all the records without altering table structure.</li><li>➤ cannot get back.</li></ul>	<ul style="list-style-type: none"><li>➤ It is used to remove all the records along with table structure.</li><li>➤ It can be get back.</li></ul>	<ul style="list-style-type: none"><li>➤ To delete particular records from the table.</li><li>➤ Can get back by using ROLL BACK.</li></ul>

### **DATA MANIPULATION LANGUAGE:** -

#### **INSERT**

#### **SYNTAX**

```
INSERT INTO table_name VALUES(V1,V2,...VN)
INSERT INTO table_name(COL1,COL2,...COLN) VALUES(V1,V2,...VN)
```

**OR**

```
INSERT INTO table_name(COL1,COL2,...COLN)
VALEU(&COL1,&COL2,.....&COLN);
```

#### **1<sup>ST</sup> WAY**

```
INSERT INTO CUSTOMER VALUES(1,'MIRCHI', 0000000000);
```

#### **2<sup>ND</sup> WAY**

```
INSERT INTO CUSTOMER(CID,CNAME,PHONE) VALUES(2,'KRISHNA',9999999999);
```

#### **3<sup>RD</sup> WAY**

```
INSERT INTO CUSTOMER VALUES(&CID,&CNAME,&PHONE);
```

```
SQL> /
```

### **UPDATE**

#### **SYNTAX**

```
UPDATE table_name
SET COL=V1,COL2=V2,...COLN=VN
[WHERE <FILETER_CONDITION>];
```

E.G.

```
UPDATE CUSTOMER
SET CNAME='CHUBBY'
WHERE CID=5;
```

## **DELETE**

### **SYNTAX**

```
DELETE  
FROM table_name  
[WHERE <filter_condition>];
```

E.G.

```
DELETE  
FROM CUSTOMER  
WHERE CID/CNAME/PHONE;
```

## **TRANSITION CONTROL LANGUAGE: -**

### **COMMIT: -**

This statement is used to save all the transitions in the database.

### **SYNTAX: -**

```
COMMIT
```

### **SAVE POINT: -**

It is used to make the position in the database.

- The data will be stored in the table but not inside database.
- We can rollback to the particular position by using save point.

### **SYNTAX**

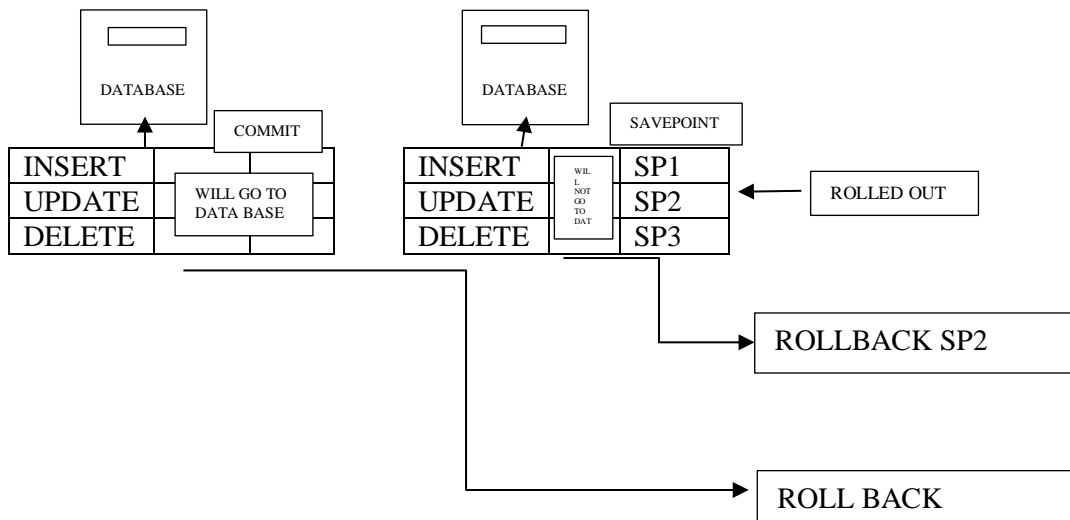
```
SAVEPOINT savepoint_name;
```

### **ROLLBACK: -**

- It is used to go back the previously commit point.
- It is also used to get back deleted.

### **SYNTAX**

```
ROLLBACK  
ROLLBACK TO SAVEPOINT  
ROLLBACK TO savepoint_name;
```



### **DATA CONTROL LANGUAGE:** -

#### **GRANT:** -

- It is used to give the permission to user.

#### **SYNTAX**

GRANT sql\_statment on table\_ name to user\_name;

#### **REVOKE:-**

- It is used to get back permission from user.

#### **SYNTAX**

REVOKE sql\_statment ON table\_name FROM user\_name;

#### **ATTRIBUTES:** -

#### **KEY ATTRIBUTES:** -

Which is used to identify a record uniquely from the table is called key attribute.

e.g.

AADHAR NO,PAN,DL,PASSPORT,PHNO,TIKTOK,GMAIL.

#### **NON-KEY ATTRIBUTE:** -

All the attributes except key attributes are referred as non-key attributes.

e.g.

NAME, AGE, DOB,HEIGHT,WEIGHT,B.P

### **PRIME-KEY ATTRIBUTES: -**

Along the key attribute an attribute is chosen to be the main attribute to identify the record uniquely from the table.

e.g.

AADHAR NO

### **NON-PRIME-KEY ATTRIBUTE: -**

All the key attributes except prime key attribute is referred non-primekey attribute.

e.g.

PAN,DL,PASSPORT,PHNO,TIKTOK ID,GMAIL

### **COMPOSITE KEY ATTRIBUTE: -**

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

e.g.

NAME, AGE, DOB

### **SUPER KEY ATTRIBUTE: -**

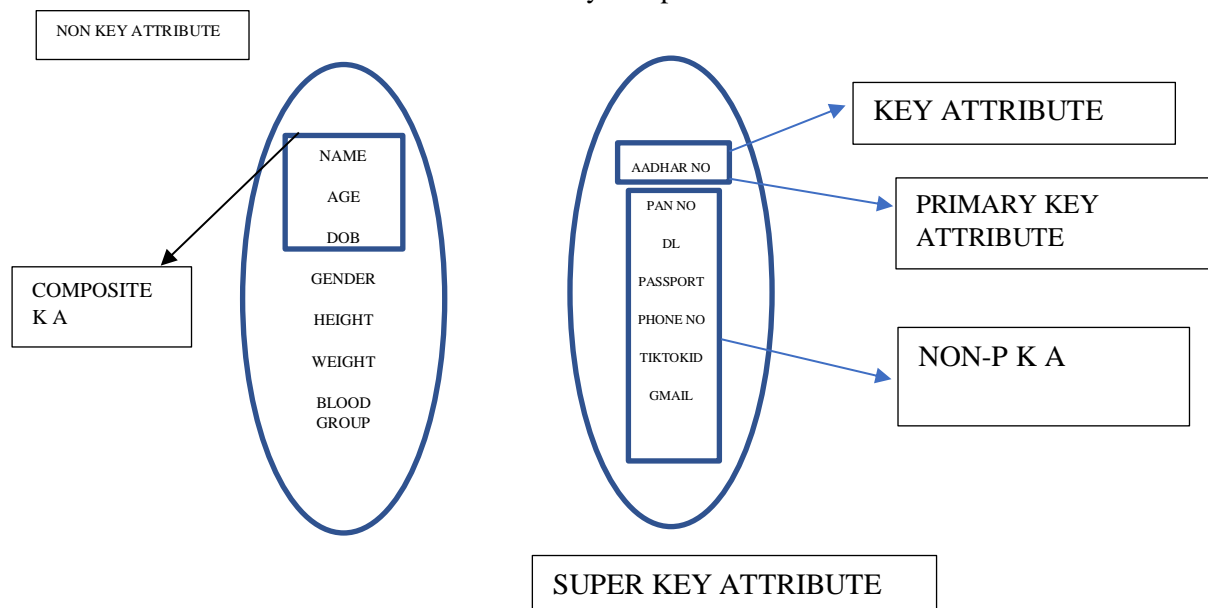
It is the set of all the key attributes.

e.g.

KEY ATTRIBUTES (AADHAR NO,PAN,DL,PASSPORT,PHNO,TIKTOK ID,GMAIL)

### **FOREIGN KEY ATTRIBUTE: -**

It behaves as an attribute of another entity to represent the relation.



### **FUNCTIONAL DEPENDENCY: -**

Let us consider the relation 'R' with two ATTRIBUTES 'x' AND 'y' respectively. In which attribute 'X' determines attribute 'Y'. or IN OTHER WORDS, 'Y' IS DEPENDENT ON 'X'.

There exists functional dependency there are no anomaly are redundancy.

### **ANOMALY**

There are the side effects which are cause during the DML operators.

### **REDUNDANCY: -**

There are the Repeated or Duplicate.

e.g.

let us consider a relation with 4 attributes A, B, C & D in which 'A' is a key attribute.

R-----→ (A,B,C,D)

A IS K,A

A----→ B

A----→ C

A---→ D

THERE EXISTS T.F.D

A- [B,C,D]

### **PARTIAL FUNCTIONAL DEPENDENCY: -**

For a partial functional dependency to exist there must be a composite key attribute.

- One of the attribute in composite key relation determine another attribute separately, and this known as partial functional dependency.
- In partial functional dependency we have redundancy and ANOMALY.

e.g.

Let us consider a relation 'R' with 4 attributes A, B, C, D in which A & B are composite key attributes.

R -----→ [A,B,C,D]

A & B --→ C.K.A

(A,B) -----→ (C,D)

B-----→ (C)

THERE EXIST P.F.D

### **TRANSITIVE FUNCTIONAL DEPENDENCY: -**

If an attribute is determined by a non-key attribute which in turn is determined by a key attribute, then there exists transitive functional dependency.

❖ In transitive functional dependency we have redundancy and ANAMOLY.

e.g.

Let us consider a relation with attributes A, B, C & D in which A is a key attribute.

R----->(A,B,C,D)

A IS K.A

A---->B

A---->D

D----->C

A---->C

THERE EXIST T.F.D

### **SET THEORY: -**

SETS in SQL is used to merge the tables vertically.

**NOTE: -**

❖ The elements that are selected in both the tables must correspond to each other.

\\

SQL

### **SET OPERATORS**

WE HAVE 4 SET OPERATORS

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

A={1,2,3,4,5} B={5,6,7}

1)UNION

A UNION B

={1,2,3,4,5,6,7}

DISTINCT

2)UNION ALL

A UNION ALL B

={1,2,3,4,5,5,6,7}

DISTINCT+DUPLICATES



### 3)INTERSECTION

```
A INTERSECTION B
={5}
COMMON
```

### 4)MINUS

```
A MINUS B = {1,2,3,4}
T1 MINUS T2

B MINUS A={6,7}
T2 MINUS T1
```

### **VIEW:** -

VIEWS are the virtual table which can be created and re-used when ever we are dealing with a part of a table.

### **SYNTAX**

```
CREATE VIEW view name
AS
SELECT stmt;
```

### **Q. DIFF B/W TABLE & VIEW & ADVANTAGES OF VIEW?**

A)

TABLE	VIEW
A table contains data	A view is just a SELECT statement which has been saved in the database (more or less, depending on your database).
A table is structured with columns and rows. Table is an independent object	Adv: it can join data from several tables thus creating a new view of it.

### **INDEX:** -

When we have INDEX, it enhance (INCREASES) the speed of searching in the DATABASE.  
i.e., THERE ARE 1000 tables in DATABASE.

So for easy of accesses, we use index.

### **INDEX ARE OF 2 TYPES:-**

1. CLUSTURED
2. NON-CLUSTURED

**PROCEDURE/STORED PROCEDURE: -**

These are PRE-COMPILED programs.

(ALREADY COMPILED AND WHICH WILL BE STORED IN EXECUTABLE FORM).

-----

NOTES

-----

**CURSOR: -**

CURSOR means memory (INTERNAL STORAGE)

(WHILE EXECUTING THESE THINGS SHOULD BE STORED SOMEWHERE

i.e., CURSOR).

**Q. DIFFERENCE BETWEEN JOINS AND SET OPERATORS?**

A) JOINS ARE USED TO JOIN TWO TABLES HORIZANTALLY

BUT,

SET OPERATORS ARE USED TO JOIN TWO TABLES VERETICALLY.

\*\*\*\*\* **THE END** \*\*\*\*\*

### **IMPORTANT TOPICS**

<b>TOPIC NO</b>	<b>TOPIC NAME</b>
1	PRIMARY KEY
2	FOREIGN KEY
3	STATEMENTS IN SQL
4	DDL
5	DML 1. INSERT 2. UPDATE
6	DQL
7	JOINS 1. INNER JOIN 2. SELF JOIN
8	SUB-QUERY
9	NORMILIZATION
10	S.R.F
11	PSEUDO COLUMNS ROWNUM
12	CO-RELATED SUBQUERY
13	VIEW
14	INDEX
15	PROCEDURE
16	CURSOR

SQL