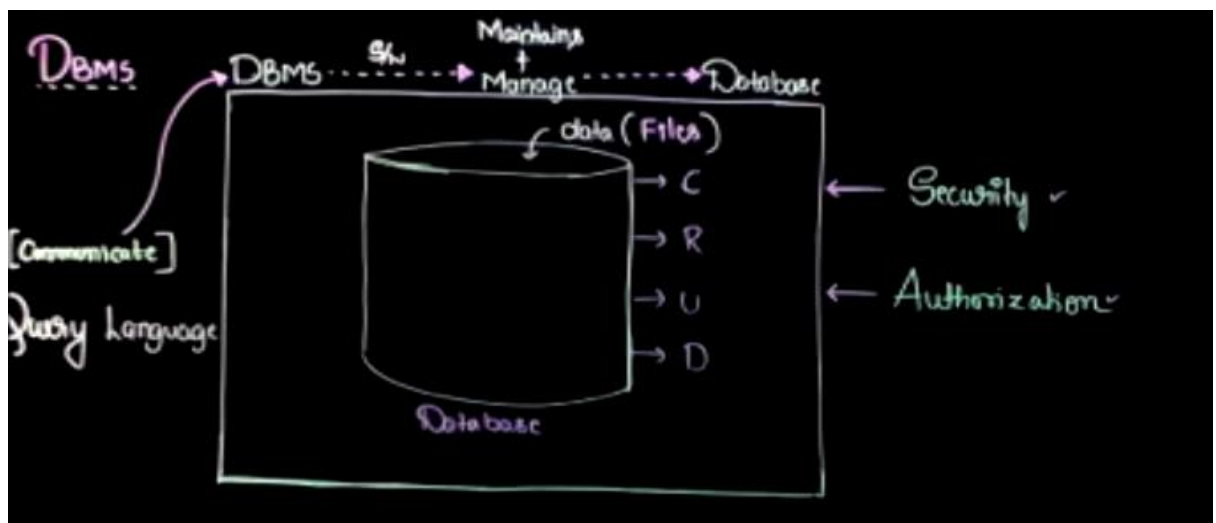


# PENTAGON SPACE

## SQL session

### DBMS (Database Management System):

- ✓ It is a software that maintains and manages database
- ✓ Provides two important features security and authorization(verification)
- ✓ The data in DBMS will be stored in the form of files
- ✓ To communicate with the DBMS software, we need to make of use language called query language.

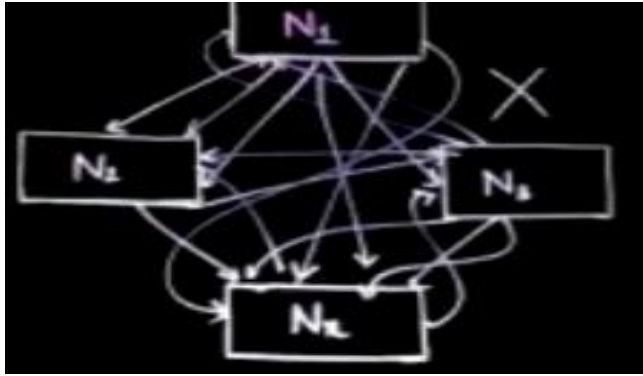


### Types of DBMS

1. Network DBMS
2. Object Oriented DBMS
3. Hierarchal DBMS
4. RDBMS

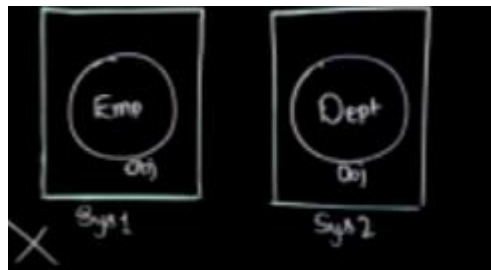
#### 1. Network DBMS:

- The data is stored in the form of network
- Data can be shared from one network to another network
- It is very much difficult to find the source of the data
- Less secure, more cost



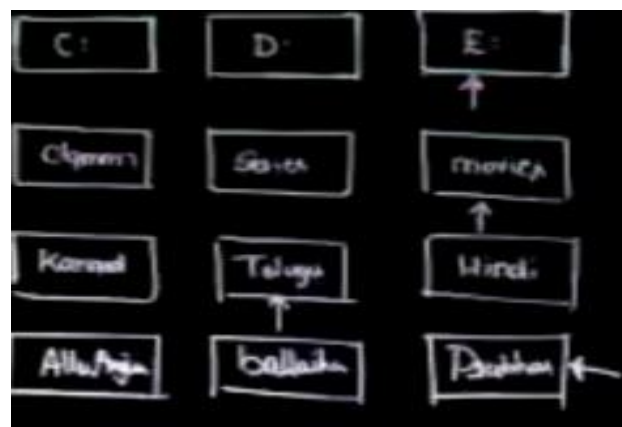
## 2. Object Oriented DBMS [OODBMS]

- Consumes more space
- stores data in form of object
- can't establish connection



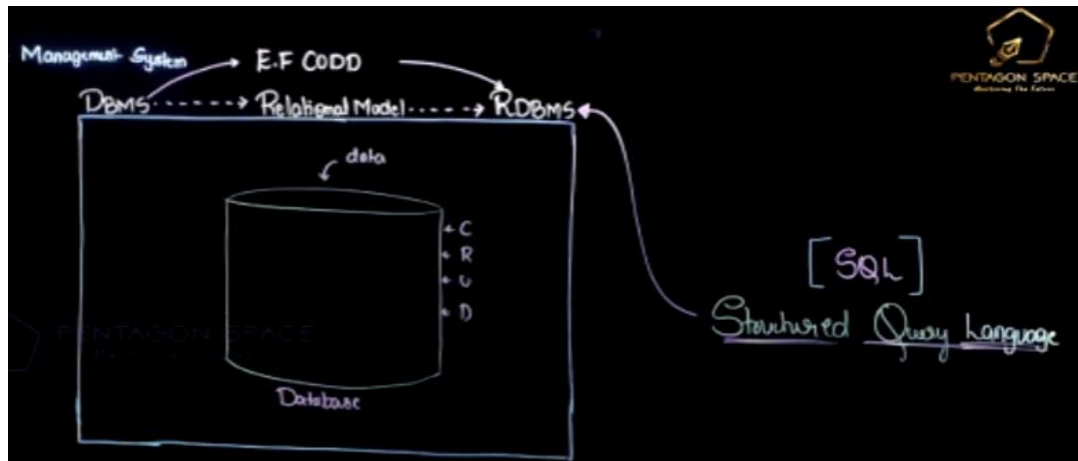
## 3. Hierarchical DBMS

- Storing data in hierarchical format
- Here in hierarchical data one data is there inside another data like one in file there is another data, but in companies there is huge data here it is very difficult for the companies to use hierarchical data
- It is not efficient manner of storing data
- Can't relate the data



#### 4. RDBMS (Relational Database management System)

- It is the software to store the data in the form of tables
- If the DBMS follows relational model then we call it as RDBMS
- If DBMS follows EF CODD rules it is RDBMS
- To communicate with RDBMS, we use structured query language (SQL)



#### Relational Model

- ✓ Relational model was introduced by E.F CODD
- ✓ In relational model data will be stored in the form of tables/relations
- ✓ In Relational model we can store meta-data

**Meta-data:** It is data about the data

#### Difference between DBMS and RDBMS

DBMS	RDBMS
➤ DBMS is a software which maintains and manages the database	➤ RDBMS is a software which is used to store the data in the form of tables /relations
➤ DBMS stores data in the form of files	➤ RDBMS stores the data in the form of tables/relations
➤ To communicate with DBMS, we use query language	➤ To communicate with DBMS, we use structured query language
➤ Supports single user access	➤ Supports multi user access
➤ Normalization cannot be performed	➤ We can perform Normalization
➤ Less Secured	➤ More Secured
➤ Ex: Microsoft access	➤ Ex: Oracle, My SQL

- ❖ Data is a raw fact which describes the attribute of the entity

### EF CODD Rules

1. The data enter into the database should be single value or atomic.
2. We can store in multiple tables and can establish the connection between those tables by using key attribute.
3. We can assign datatypes and constraints to validate the data to enter into the table, datatypes are mandatory whereas constraints are optional.

### Write a difference between RDBMS and Excel Sheet ( Spread Sheet )

Aspect	RDBMS (Relational Database Management System)	Excel Sheet (Spreadsheet)
Purpose	Designed to store and manage structured data efficiently	Used for simple data entry, calculations, and analysis
Data Storage	Stores data in related tables with defined schema	Stores data in cells arranged in rows and columns
Data Relationships	Supports complex relationships (primary/foreign keys)	Does not support relational integrity between sheets
Data Volume Handling	Efficiently handles large volumes of data	Limited handling of large datasets
Multi-user Support	Supports concurrent access by multiple users	Limited or no real-time multi-user support
Data Integrity	Enforces strict data types and constraints	Minimal data validation and enforcement
Querying	Uses SQL for complex queries and operations	Uses simple functions and filters
Security	Advanced user roles and access control	Basic password protection
Scalability	Highly scalable for enterprise applications	Not suitable for large-scale applications

## Data Types

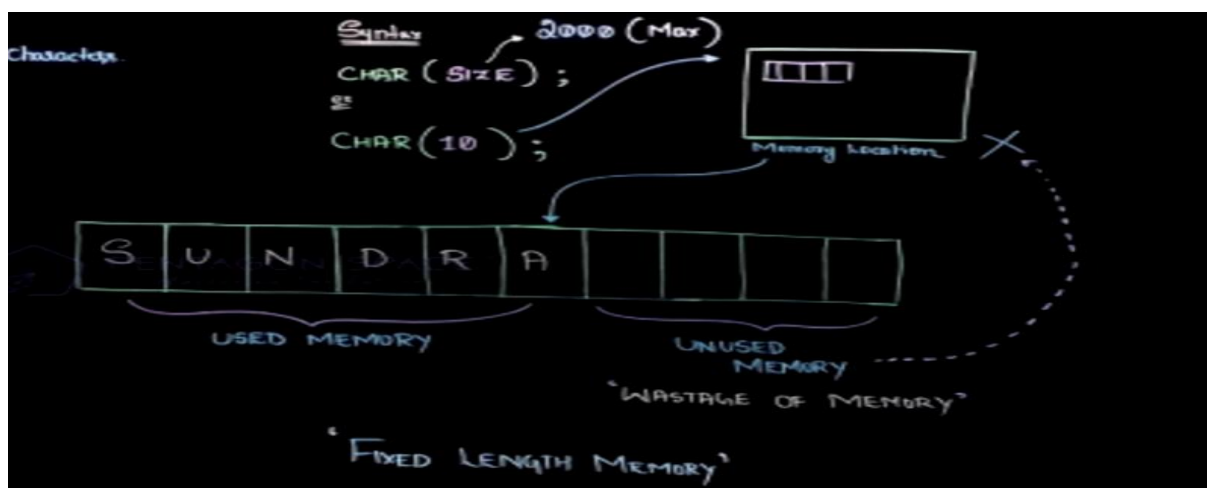
Before storing any data into the memory location, we have to specify the type of data to specify the type of data we use data types. Data types and constraints are used to validate the data.

### Types of Data Types

1. CHAR
2. VARCHAR
3. LARGE OBJECTS (LOB)
  - a. CHARACTER LARGE OBJECT (CLOB)
  - b. BINARY LARGE OBJECT (BLOB)
4. DATA
5. NUMBER

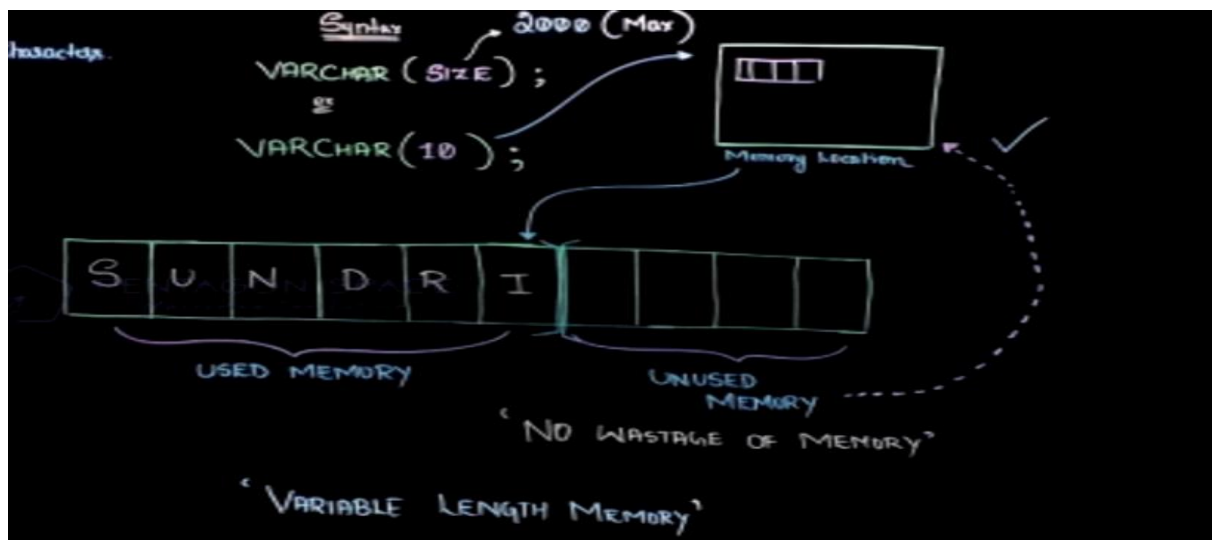
### CHAR

- It is used to store characters
- Allows upper case, lower case, special character, '0-9'
- Char datatype is also called as fixed length memory location
- The max capacity is 2000 for char.
- Char data type size as an argument
- In case of char the unused memory cannot be given back to the memory location for reuse purpose
- Hence there is a wastage of memory.



## VARCHAR

- It is used to store characters.
- Allows upper case, lower case, special character, '0-9'.
- Takes size as an argument.
- Varchar is called as variable length memory location.
- The max capacity is 2000 for varchar.
- Unused memory will be given back to memory location for reuse purpose.
- No wastage of memory



## VARCHAR 2

- It is an updated version of varchar
- It stores maximum character up to 4000

## LARGE OBJECTS (LOB)

### CHARACTER LARGE OBJECT (CLUB)

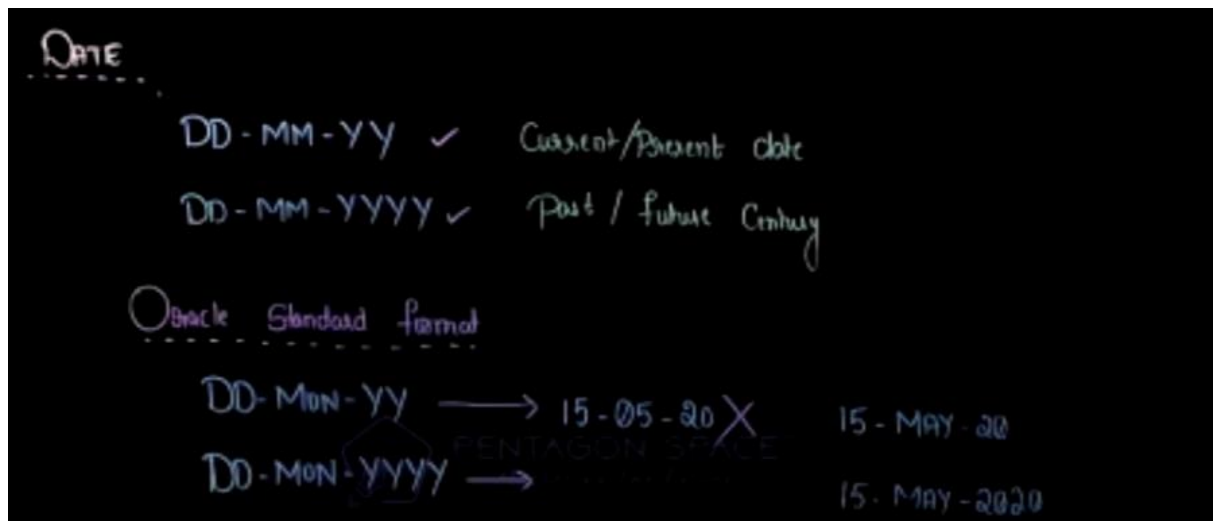
- It is used to store large amount of characters up to 4GB
- Syntax: Club;

### BINARY LARGE OBJECT (BLOB)

- It is used to store the binary objects such as audio, video, image, document, pdf in the form of binary format.
- Syntax: Blob;

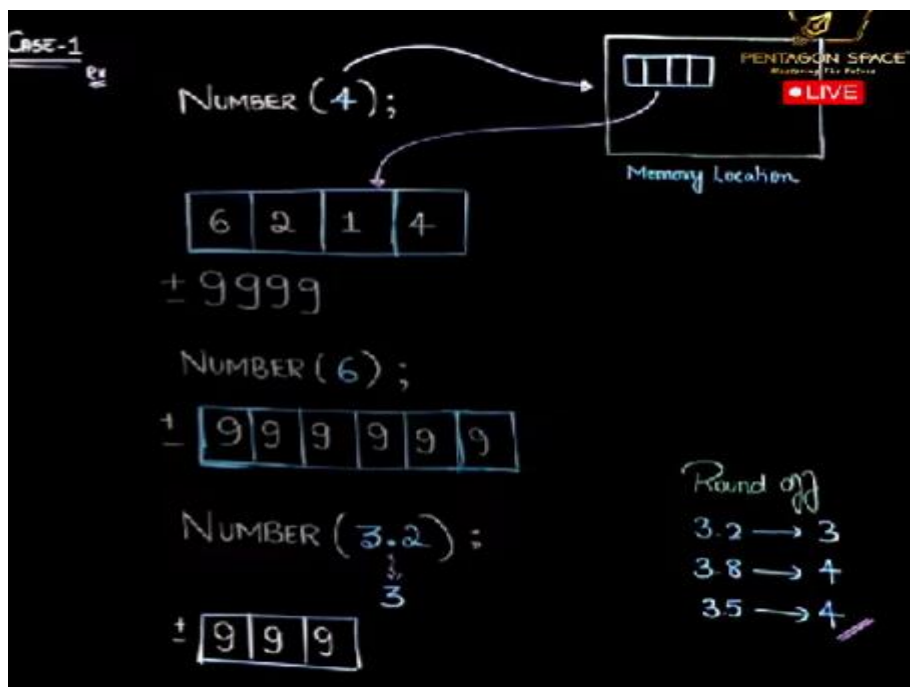
## DATE

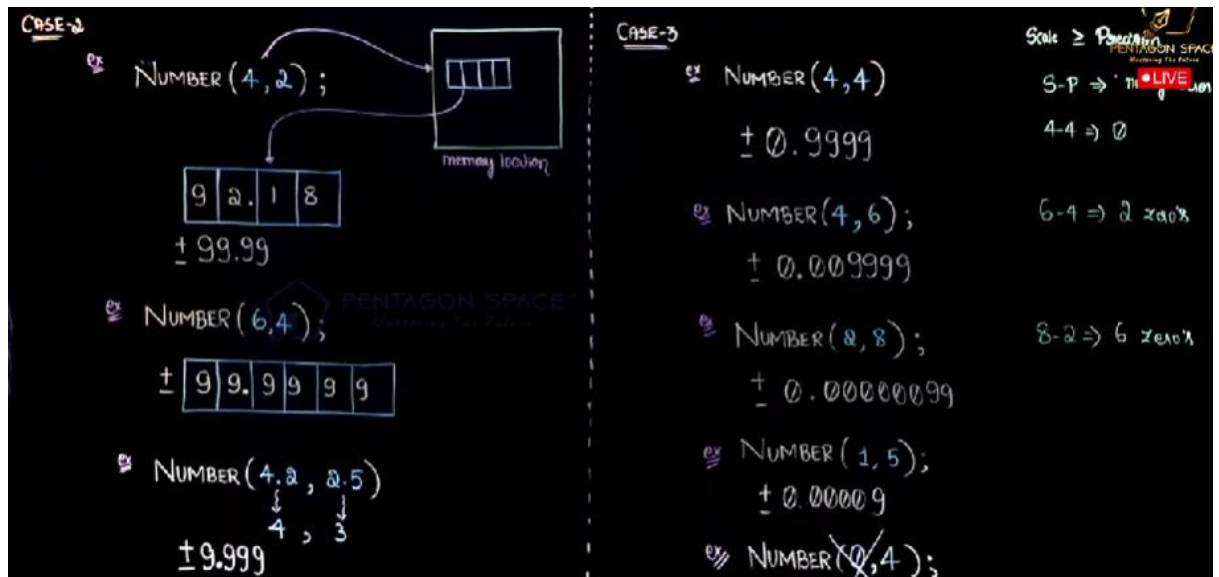
- It is used to store the date



## NUMBER

- It is used to store numbers
- Syntax: NUMBER (precision, [scale]); the scale is optional
- Precision: used to store integer value (numeric non - decimal value)
- Scale: It is used to store decimal values





## CONSTRAINTS

It is a rule or condition given to a table to validate the data.

### Types of Constraints

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

### UNIQUE Constraint

Unique Constraint is used to avoid duplicate values entered into a table

**Note:** Null means Empty, Null is a keyword not a constraint.

### NOT NULL Constraint

It is used to avoid duplicate values which are entered into a table

### CHECK Constraint

- It is used to provide user defined condition
- Syntax:  
 CHECK (condition)
- Example: `CHECK (Sal >= 100)`  
`CHECK (perc >= 0 && per <= 100)`



## **PRIMARY KEY Constraint**

It is used to uniquely identify records from the table

### **Characteristics of Primary key**

- To represent the table among the scheme we use primary key
- Primary key must be a combination unique and not null
- A table can have only one primary key as per the database standards
- Primary key is not mandatory for a table, but design wise it is referable

## **FOREIGN KEY Constraint**

Foreign key is used to establish the connection between two tables

### **Characteristics of Foreign key**

- A table can have multiple foreign key
- A primary key of a table can be eligible to foreign key of another table
- Foreign key can be null and can accept duplicate values, and also it can be unique and not null but nothing is mandatory.
- Foreign key is also known as Referential Integrity Constraint
- Foreign key is present in child table but always belongs to parent table
- Foreign key is not mandatory for a table, but design wise it is referable

## **SQL Statements**

### **1. Data Definition Language (DDL)**

- i. CREATE
- ii. RENAME
- iii. ALTER
- iv. TRUNCATE
- v. DROP

### **2. Data Manipulation Language (DML)**

- i. INSERT
- ii. UPDATE
- iii. DELETE

### **3. Data Control Language (DCL)**

- i. GRANT
- ii. REVOKE

### **4. Transaction Control Language (TCL)**

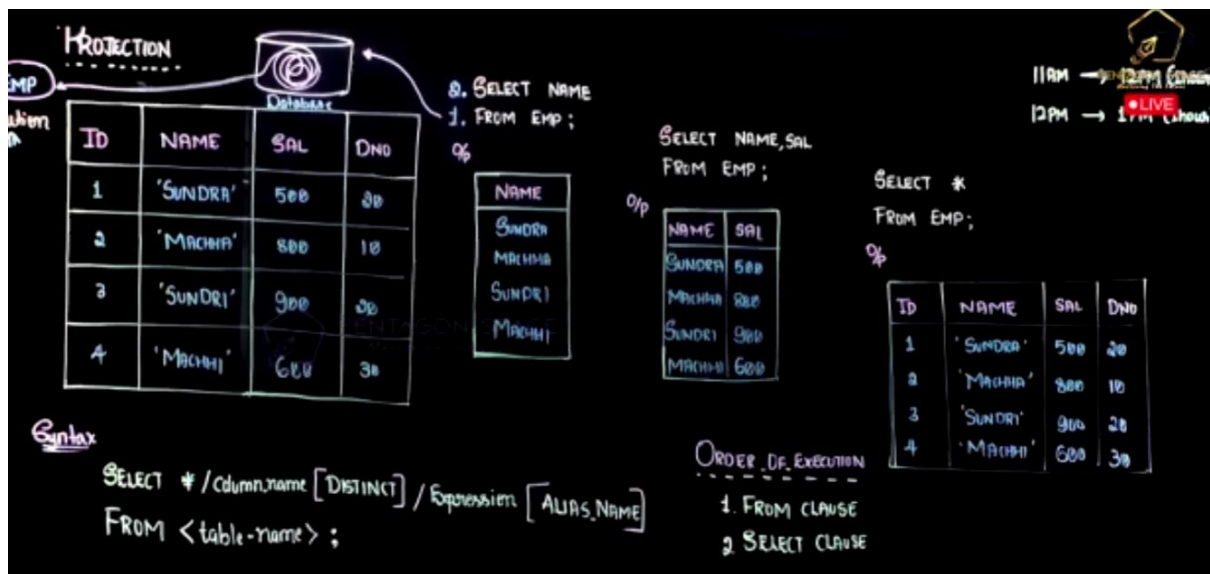
- i. COMMIT
- ii. ROLL BACK
- iii. SAVE POINT

### **5. Data Query Language (DQL)**

- i. SELECT
- ii. PROJECTION
- iii. SELECTION
- iv. JOINS

### **Data Query Language (DQL)**

- i. SELECT:**
  - It is used to retrieve data and display the output
- ii. PROJECTION:**
  - It is used to retrieve data from the given table by selecting column name
  - By default, in projection all records get selected
- iii. SELECTION:**
  - It is used to retrieve the data from the table by selecting column name and by providing condition
- iv. JOINS:**
  - It is used to retrieve the data from multiple tables simultaneously



## FROM CLAUSE:

In the query from class executes first

## SELECT CLAUSE:

It is used to retrieve the data

## Number format Datatype

- **Int:** To store integer values
- **Big int:** To store large number of integer value data
- **Decimal:** To store both integer and decimal value.

## Data Format Data type

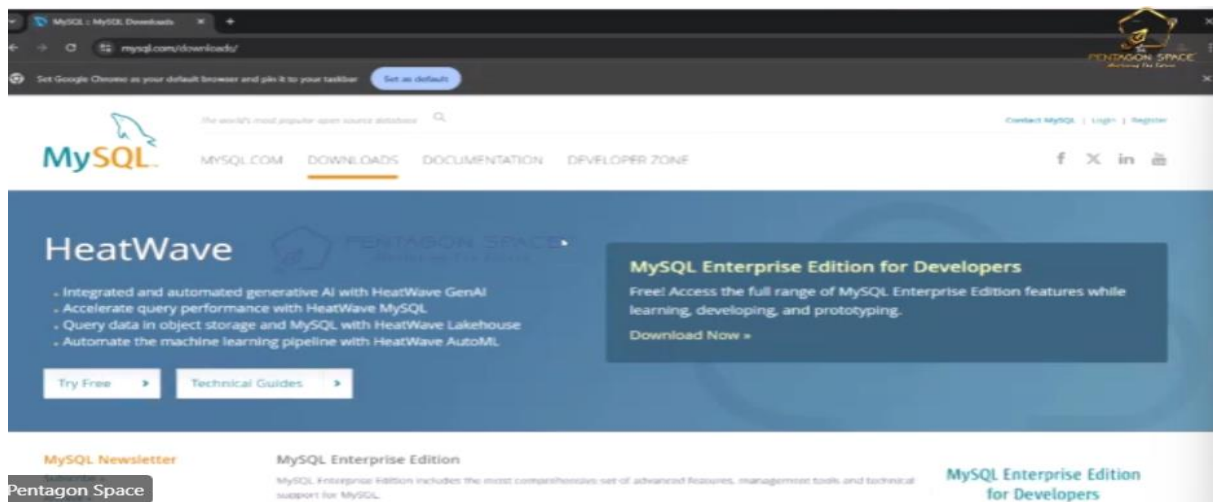
- **DATE:** To store data values
  - **MY SQL FORMAT:** 'YYYY – MM – DD'
- **DATETIME:** To store data along with time
  - **MYSQL FORMAT :** 'YYYY – MM – DD HH:MI:SS '
- **TIME:** To store Time values
  - **MYSQL FORMAT :** 'HH : MI : SS '
- **TIMESTAMP:** To store date, time along Timezone
  - **MYSQL FORMAT:** 'YYYY – MM – DD HH:MI:SS UTC '
  - **UTC:** Co-ordinated universal Time

## CONSTRAINTS

- **AUTO\_INCREMENT:** To generate unique value for a column automatically
  - Default is 1
  - Only for primary key column we can use auto\_increment
- **ENUM:** It act as a datatype, to set some limited set of values we use enum
- **Default:** To set a default value for a column we use default

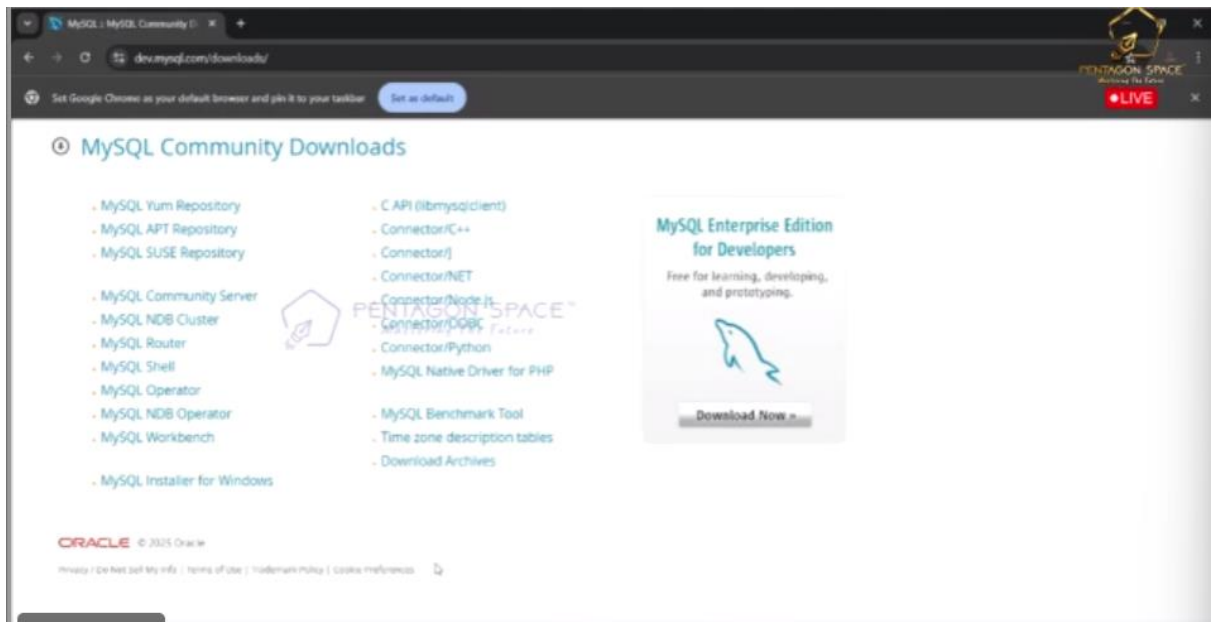
## Steps to install My SQL:

- open chrome
- search for my sql download
- you will get interface
- open the first link



- Scroll down we have my sql community downloads
- No thanks, just start my download





- For windows here we have my sql installer for windows, click on it
- Next click on the first download option
- Click on the file you downloaded
- Click yes, then next
- Use custom, click on first option (my sql server → my sql server → my sql 8.0 → then select first option), next open application (choose my sql work bench → my sql workbench 8.0 → select the first option)
- Then click next
- If we forget my sql password uninstall and install.

## Keywords

**Signed:** It will allow both positive and negative numbers for a column

**Unsigned:** It will allow only positive numbers for a column

- ❖ If we don't use these keywords by default, it is signed keyword.

## SQL STATEMENTS

### Data Definition Language (DDL)

#### i. **CREATE:**

- It is use to create a database and its objects such as table, view, procedure, Trigger and User.
- To create a database, **syntax:** create database database\_name;

- To view all databases, present in my SQL, **syntax:** show databases;
- To access one database, **syntax:** use database\_name;
- To create a table,

TO CREATE TABLE

SYNTAX:1

```
CREATE TABLE TABLE_NAME
(COLUMN_NAME_1 DATATYPE SIGNED/UNSIGNED NULL/NOT NULL,
 COLUMN_NAME_2 DATATYPE SIGNED/UNSIGNED NULL/NOT NULL,
,
,
,
,
COLUMN_NAME_N DATATYPE SIGNED/UNSIGNED NULL/NOT NULL,
CONSTRAINT CONSTRAINT_NAME PRIMARY KEY(COLUMN_NAME),
CONSTRAINT CONSTRAINT_NAME UNIQUE(COLUMN_NAME),
CONSTRAINT CONSTRAINT_NAME CHECK(CONDITION),
CONSTRAINT CONSTRAINT_NAME FOREIGN KEY(COLUMN_NAME) REFERENCES PARENT_TABLE_NAME(COLUMN_NAME))
);
```

- Example:

Student

column	SID	SNAME	PHONE
Datatype	Unsigned Int	Varchar(10)	Unsigned Int
null/notnull	NN	NN	NN
constraint	Primary Key	-	Unique check(length(Phone)=10)

P\_SID                      U\_Ph  
   C\_Ph

(create table Student

```
( SID Int Unsigned Not Null,
  SNAME Varchar(10) Not Null,
  PHONE Int Unsigned Not Null,
  constraint P_SID Primary key (Sid),
  constraint U_Ph Unique(Phone),
  constraint C_Ph check( length(Phone)=10)
);
```

```
CREATE TABLE STUDENT
(SID INT UNSIGNED NOT NULL,
 SNAME VARCHAR(10) NOT NULL,
 PHONE INT UNSIGNED NOT NULL,
 CONSTRAINT P_SID PRIMARY KEY(SID),
 CONSTRAINT U_PH UNIQUE(PHONE),
 CONSTRAINT C_PH CHECK(LENGTH(PHONE)=10)
);
```

BRANCH:

BID	INT	UNSIGNED NN	PRIMARY KEY:P_BID
BNAME	VARCHAR(10)	NN	
LOCATION	VARCHAR(20)	NN	
PINCODE	INT	UNSIGNED NN	UNIQUE,CHECK(LENGTH(PINCODE)=6)
			U_PIN   C_PIN

```

CREATE TABLE BRANCH
(BID INT UNSIGNED NOT NULL,
 BNAME VARCHAR(10) NOT NULL,
 LOCATION VARCHAR(20) NOT NULL,
 PINCODE INT UNSIGNED NOT NULL,
 CONSTRAINT P_BID PRIMARY KEY(BID),
 CONSTRAINT U_PIN UNIQUE(PINCODE),
 CONSTRAINT C_PIN CHECK(LENGTH(PINCODE)=6)
);

```

SYNTAX:2 TO CREATE TABLE:

-----

```

CREATE TABLE TABLE_NAME
(
 COLUMN_NAME_1 DATATYPE SIGNED/UNSIGNED NULL/NOT NULL CONSTRAINT,
 COLUMN_NAME_2 DATATYPE SIGNED/UNSIGNED NULL/NOT NULL CONSTRAINT,
 ,
 ,
 ,
 ,
 COLUMN_NAME_N DATATYPE SIGNED/UNSIGNED NULL/NOT NULL CONSTRAINT,
 CONSTRAINT FOREIGN KEY(COLUMN_NAME)REFERENCES PARENT_TABLE_NAME(COLUMN_NAME)
);

```

**Example:**

FID	INT	UNSIGNED	NN	PRIMARY KEY
FNAME	VARCHAR(10)		NN	
SUBJECT	VARCHAR(10)		NN	
DNAME	VARCHAR(10)		NULL	
PHONE	INT	UNSIGNED	NN	UNIQUE, CHECK(LENGTH(PHONE)=10)

```

CREATE TABLE FACULTY
(
 FID INT UNSIGNED NOT NULL PRIMARY KEY,
 FNAME VARCHAR(10) NOT NULL,
 SUBJECT VARCHAR(10) NOT NULL,
 DNAME VARCHAR(10),
 PHONE INT UNSIGNED NOT NULL UNIQUE, CHECK(LENGTH(PHONE)=10)
);

```



## CUSTOMER

-----

Field	Field Type	Field Properties
CID	INT	UNSIGNED NN PRIMARY KEY AUTO_INCREMENT
CNAME	VARCHAR(10)	NN
AGE	INT	UNSIGNED NN
ORDER_ID	INT	UNSIGNED NULL
CITY	VARCHAR(10)	NN
BALANCE	DECIMAL(10,2)	UNSIGNED NN DEFAULT '100.00'
SID	INT	UNSIGNED FOREIGN KEY

## CREATE TABLE CUSTOMER

```
(
CID INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
CNAME VARCHAR(10) NOT NULL,
AGE INT UNSIGNED NOT NULL,
ORDER_ID INT UNSIGNED,
CITY VARCHAR(10) NOT NULL,
BALANCE DECIMAL(10,2) UNSIGNED NOT NULL DEFAULT '100.00',
SID INT UNSIGNED,
CONSTRAINT FOREIGN KEY(SID) REFERENCES STUDENT(SID)
);
```

## BANK

-----

Field	Field Type	Field Properties
ACCOUNT_NUMBER	BIGINT	NN PRIMARY KEY
ACCOUNT_HOLDER_NAME	VARCHAR(20)	NN
GENDER	ENUM('MALE', 'FEMALE')	NN
BALANCE	DECIMAL(7,2)	NN DEFAULT '500.00'
IFSC_CODE	VARCHAR(20)	NN CHECK(LENGTH(IFSC_CODE)=11)
BID	INT UNSIGNED	FOREIGN KEY

## CREATE TABLE BANK

```
(ACCOUNT_NUMBER BIGINT NOT NULL PRIMARY KEY,
ACCOUNT_HOLDER_NAME VARCHAR(20) NOT NULL,
GENDER ENUM('MALE', 'FEMALE') NOT NULL,
BALANCE DECIMAL(7,2) NOT NULL DEFAULT '500.00',
IFSC_CODE VARCHAR(20) NOT NULL CHECK(LENGTH(IFSC_CODE)=11),
BID INT UNSIGNED,
CONSTRAINT FOREIGN KEY(BID) REFERENCES BRANCH(BID)
);
```

- ❖ To view structure of the table,
  - **syntax:** Desc table\_name;
- ❖ To clear screen in my SQL
  - **syntax:** system cls
- ❖ To Exit from my SQL server



- **syntax:** EXIT
- ❖ To **view constraint name** of the constraints
  - **Step1:** Use information\_schema;
  - **Step2:** Select \* from table\_constraints;

## ii. **ALTER:**

- It is used to modify the structure of the table

### 1. **To add a new column:**

#### **Syntax:**

```
alter table table_name
add column_name datatype null/not null;
```

#### **Example:**

```
Email varchar (10) not null --- student
alter table student
add email varchar (20) not null;
```

### 2. **To add a column after a particular column:**

#### **Syntax:**

```
alter table table_name
add column_name datatype null/not null after column_name;
```

#### **Example:**

```
Balance int null after sname---student
alter table student
add balance int null after sname;
```

### 3. **To drop column\_name:**

#### **Syntax:**

```
alter table table_name
drop column_name;
```

#### **Example:**

```
ifsc_code ---- bank
alter table bank
drop ifsc_code;
```

### 4. **To change the datatype:**

#### **Syntax:**

```
alter table table_name
modify existing_column_name new_datatype null/not null;
```

**Example:**

Change dname datatype in faculty table from varchar to char

```
alter table faculty
```

```
modify dname char (20) null;
```

**5. To change null/not null:****Syntax:**

```
alter table table_name
```

```
modify existing_column_name existing_datatype null/not null;
```

**6. To change table\_name:****Syntax:**

```
alter table table_name
```

```
rename new_table_name;
```

**Example:**

```
Student---student_data
```

```
alter table student
```

```
rename student_data;
```

**7. To change column\_name****Syntax:**

```
alter table table_name
```

```
change existing_column_name new_column_name existing_datatype null/not  
null;
```

**Example:**

```
alter table faculty
```

```
change fname name varchar(20) not null;
```

❖ We cannot drop/rename a column which is assigned with check constraint

**8. To add constraints:****i. To add primary key:****Syntax:**

```
alter table table_name
```

```
add constraint primary key(column_name);
```

**ii. To add unique:****Syntax:**

```
alter table table_name
```

```
add constraint unique(column_name);
```

**Example:**

```
alter table branch  
add constraint unique(location);
```

**iii. To add check constraint:**

**Syntax:**

```
alter table table_name  
add constraint check(condition);
```

**iv. To add foreign key:**

**Syntax:**

```
alter table table_name  
add constraint foreign key(column_name)references  
parent_table_name(column_name);
```

**v. To add default constraint for existing column:**

**Syntax:**

```
alter table table_name  
modify existing_column_name existing_datatype null/not null  
default'default_value';
```

**Example:**

```
alter table product  
modify quantity int unsigned not null default'100';
```

**vi. To add default constraint for new column:**

**Syntax:**

```
alter table table_name  
add column_name datatype null/not null default'default_value';
```

**Example:**

```
alter table faculty  
add balance int not null default'100';
```

**vii. To add auto\_increment for existing column**

**Syntax:**

```
alter table table_name  
Modify existing_column_name existing_datatype null/not null  
auto_increment;
```

**Example:**

```
alter table product  
modify pid int unsigned not null auto_increment;
```

**viii. To add auto\_increment for a new column:****Syntax:**

```
alter table table_name  
add column_name datatype null/not null primary key auto_increment;
```

**9. To Drop constraints:****i. To drop primary key:****Syntax:**

```
alter table table_name  
drop primary key;
```

- ❖ we can't drop primary key from a table if primary is acting as a foreign key in another table
- ❖ we can't drop primary key from a table if a primary key column is assigned with auto\_increment

**ii. To drop default****Syntax:**

```
alter table table_name  
alter column column_name  
drop default;
```

**Example:**

```
alter table faculty  
alter balance  
drop default;
```

**iii. To drop auto\_increment:****Syntax:**

```
alter table table_name  
modify existing column_name existing datatype null/not null ;
```

- ❖ If we don't pass auto increment automatically auto increment is removed

**Example:**

```
alter table customer  
modify cid int unsigned not null;
```

**iv. To drop other constraints (unique, check, foreign key)**

**Syntax:**

Alter table table\_name

Drop constraint constraint\_name;

**iii. RENAME**

**iv. TRUNCATE:**

It is used to delete all the records from a table without affecting the structure of a table.

**Syntax:**

truncate table table\_name;

**Example:**

truncate table product;

**v. DROP :**

It is used to drop database and its objects such as table, view, procedure, trigger and user from MySQL

**Syntax:**

drop table table\_name;

**To drop database from MySQL**

**Syntax:**

drop database database\_name;

**Example:**

drop database demo;

**Data Manipulation Language (DML)**

**i. INSERT:**

It is used to insert/add records into existing tables

**Syntax 1 (when we know column name along with column order from the table)**

insert into table\_name values (v1, v2, ....., vn);

**Example:**

insert product values(1,'iphone',45000,1);

**Syntax 2(if we know column name):**

Insert into table\_name (col1, col2, ....., coln) values (v1, v2, ....., vn);

**Example:**

```
insert into product (pname, pid, price, quantity) values ('kitkat',7,20,5),  
('munch',8,20,5);
```

**Syntax 3:**

```
insert into table_name (select statement);
```

**ii. UPDATE**

It is used to modify the existing records in a table

**Syntax:**

```
update table_name  
set column_name_1=v1, column_name_2=v2,....., column_name_n=vn  
[where condition];
```

**Example:**

```
update product  
set pname='kabab', price=150  
where pid=5;
```

**iii. DELETE**

It is use to delete existing from a table

**Syntax:**

```
delete  
from table_name  
[where condition];
```

**Example:**

```
delete  
from product  
where pid=9;
```

**write a query to display employee fname and lname from employees table**

```
select fname, lname from emps;
```

**write a query to display employee and dob along with their status**

```
select doj, dob, status from emps;
```

**write a query to display details of employees from emps table**

```
select * from emps;
```

**wqtd employee fname,lname,job and commission with 10% deduction**

**wqtd employee fname, job, sal with 15% deduction and annual salary with 11% hike**

```
select fname ,job,sal-(sal*15/100),(sal*12)+((sal*12)*11/100) from emps;
```

### **Aliasing:**

It is used to provide alternative name for a column in resultant table

### **Rules:**

- With or without using a keyword we can use write alias name.
- We can use multiple words as alias name by using inside quotes or by connecting it with underscore
- **Example:** select sal as salary;
- select sal salary from emps;
- select sal\*12 as annual\_salary from emps;
- select sal\*12 as “Annual Salary” from emps;

### **Distinct**

It is used to avoid duplicate values from the resultant table

### **Rules:**

- Either \* or distinct must be the first argument in select clause
- We can use multiple columns along with distinct keyword, it will avoid combination of duplicate values

### **Wqtd different job roles present in emps table**

```
Select distinct job from emps;
```

### **Wqtd unique mgr from emp table**

```
select distinct mgr from emps;
```

### **Wqtd unique combination of salary an mgr**

```
select distinct sal, mgr from emps;
```



## Selection

It is used to retrieve the from a table by selecting a column\_name/ Expression and providing some condition

### Syntax:

Select column name/expression

From the table

Where condition;

### Where:

To filter the records from a table

### Characteristics of where clause

1. It executes after from clause
2. It executes row by row
3. It evaluates true or false conditions
4. We can write multiple condition inside where clause
5. We can't use alias name inside where clause

### Order of execution

1. FROM
2. WHERE
3. SELECT

Emp

Int Id	Varchar(n) Name	Decimal(s) Salary	Int Dno
1	Sachin	1000	10
2	Dhoni	2000	20
3	Rohit	5000	30
4	Virat	4000	20

RR  
RR  
RR  
RR

WQTD Employee Name And Salary if Employee Name is Rohit.

③ Select Name, Salary  
① From Emp  
② Where Name = 'Rohit';

Sachin = Rohit F  
Dhoni = Rohit F  
Rohit = Rohit T  
Virat = Rohit F

o/p

Name	Salary
Rohit	5000

### **Wqtd details of emps if emp fname is kiran**

```
select *from emps
```

```
where fname="kiran";
```

### **WQTD fname as first name, lname as last name and job from emps table if the emp is working as waiter**

```
select fname 'first name' ,lname 'last name',job
```

```
from emps
```

```
where job='waiter';
```

### **WQTD details of the emps who are getting sal more than 45000**

```
Select * from emps where sal>45000;
```

### **WQTD fname, lname , sal as salary and dob if emp born after the year 1993**

```
select fname, lname, sal, dob from emps where dob>='1994-01-01' or where dob >'1993-12-31';
```

### **WQTD fname, job, doj if emp hired before the year 2019**

```
select fname, job, doj from emps where doj < '2019-12-31' ;
```

### **WQTD details of the Emps along with annual salary if emp is getting annual salary more than 500000**

```
select *, sal *12 as 'annual salary'
```

```
from emps
```

```
where sal*12 >500000;
```

### **Operators**

1. Arithmetic operators (+, -, \*, /, %)
2. Relational operators (<, >, <=, >=, =, !=)
3. Logical operators (AND, OR, NOT)
4. Special operators (IN, NOT IN, IS, BETWEEN, LIKE, NOT LIKE)
5. Subquery operators (ALL, ANY)

## Logical Operators

**OR:** It returns true if any one input is true

**AND:** It returns true if any all the inputs is true

Truth table

X	Y	Z
0	1	1
1	0	1
0	0	0
1	1	1

X	Y	Z
0	1	0
1	0	0
0	0	0
1	1	1

Assume,  
1 = true  
0 = false

**WQTD details of emps if emps are working as chef or cashier**

```
select * from emps where job = "chef" OR job= "cashier";
```

**WQTD fname, lname, sal, dob from emps table if employee born in year 1995**

```
select fname, lname, sal, dob
```

```
from emps
```

```
where dob >= '1995-01-01' and dob<='1995-12-31';
```

**WQTD fname,job and salary if the meps are working as waiter and getting salary more than 50000**

select fname, job, sal

from emps

where job="waiter" and sal>50000;

**WQTD details of the emps if the emps are working as security or manager and their status is available**

select \*

from emps

where (job="security" or job="manager") and status="available";

**IN**

- It is a multi value operator which takes multiple values at the rhs and dingle value at lhs

**SYNTAX:**

- **COLUMN\_NAME/EXPRESSION** **IN**(V1,V2,.....VN);
- In operator works on “or” condition

WQTD NAME of the Emp if Employee is getting salary as 100, 200, 300 or 400

Emps	Name	Sal
R-R	Ramu	300
R-R	Raj4	500

o/p

Name	False	500=100 F	300=100 F	True
Ramu		500=200 F	300=200 F	
		500=300 F	300=300 T	
		500=400 F	300=400 F	

When:- Whenever we need to compare multiple values At a time based on OR condition we use In operator.

Insta: its\_me\_raksit.\_

**WQTD details of the employees who are working as security, chef, delivery or waiter and getting salary more than 45000**

select \*

from emps

where (job="security" or job="chef" or job = "delivery" or job="waiter") and sal >45000;

OR

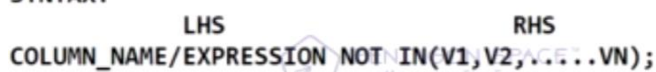
Select \* from emps

Where job in ('security', 'chef', 'delivery', 'waiter') and sal>45000;

**NOT IN**

- It is a multi value operator which takes multiple values at the rhs single value at the lhs
- It works on “AND” condition

SYNTAX:



- Whenever we need to avoid comparison between multiple values based on AND operation

**WQTD name of Employee if employee is not getting salary as 100,200,300,400**

select fname

from emps

where sal not in(100,200,300,400);

**WQTD fname, lname, sal, job if the emps are not working as chef, manager, waiter (without special character)**

select fname,lname,sal,job

from emps

where job not in('chef','manager','waiter');

## BETWEEN

- Whenever we need to include some range of values we use BETWEEN operator

**SYNTAX:**

- $\text{COLUMN\_NAME/EXPRESSION BETWEEN LOWER\_RANGE\_VALUE AND HIGHER\_RANGE\_VALUE;}$

**WQTD details of employees if the employees are getting salary more than or equals to 35000 or less than or equals to 50000**

select \*

from emps

where sal between 35000 and 50000;

**WQTD details of employees those who are born in the year 1995**

select \*

from emps

where dob between '1995-01-01' and '1995-12-31';

## NOT BETWEEN

- Whenever we need to exclude some range of values, we use NOT BETWEEN

**SYNTAX:**

- $\text{COLUMN\_NAME/EXPRESSION NOT BETWEEN LOWER\_RANGE\_VALE AND HIGHER\_RANGE\_VALUE;}$

**WQTD details of the employees those who are not getting salary in the range of 35000 to 45000**

select \*

from emps

where sal not between 35000 and 45000;

**WQTD details of emps those who are not joined in the year 2019**

select \* from emps

where doj not between '2019-01-01' and '2019-12-31';

## IS

- It is used to check whether the column is null or not null
- Syntax: Column\_name / expression is null/not null;

### WQTD details of the employees who are getting some commission

select \*

from emps

where sal is not null;

### WQTD details of the emps who are acting as customer for their company

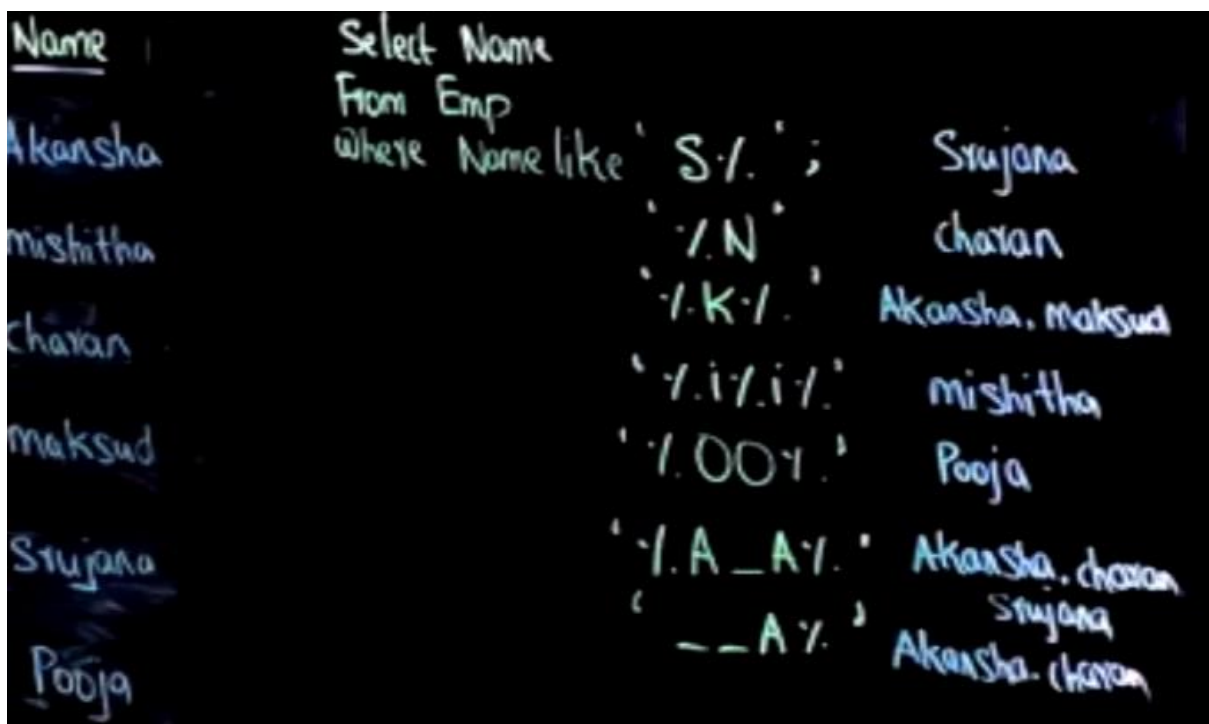
select \*

from emps

where cid is not null;

## LIKE

- It is used for pattern matching
- **Syntax:** Column\_name/Expression Like 'pattern\_to\_match';
- **%:** It takes any character n number of time
- **\_:** It takes any character but only one time



**WQTD details of the emps if employee fname is starting with k.**

```
select * from emps
```

```
where fname like 'k%';
```

**WQTD fname, lname if employee fname is ending with i**

```
select fname, lname from emps
```

```
where fname like '%i';
```

**WQTD fname, lname and sal if employee fname contains atleast 2a and salary more than 25000**

```
select fname, lname, sal from emps
```

```
where fname like '%a%a%' and sal>25000;
```

**WQTD details of the emps if job is starting with sec**

```
select * from emps
```

```
where job like 'sec%';
```

**WQTD details of the emps if fname starts with s or a**

```
select * from emps
```

```
where fname like 's%' or fname like 'a%';
```

**WQTD details of the emps if fname starts with vowels**

```
select * from emps
```

```
where fname like 'a%' or fname like 'e%' or fname like 'i%' or fname like 'o%' or fname like 'u%';
```

**WQTD details of emps if the emps hired in the year 2019**

```
select *
```

```
from emps
```

```
where doj like '2019%';
```



**WQTD fname, lname, dob if emps were born in the month of jan, feb or march**

```
select fname,lname,dob
```

```
from emps
```

```
where dob like '%-01-%' or dob like '%-02-%' or dob like '%-03-%';
```

**WQTD details of emps who were hired in the date of 12 or 01**

```
select *
```

```
from emps
```

```
where doj like '%12' or doj like '%01';
```

**NOT LIKE**

- It is also used for pattern matching
- Here, it will exclude the details based on the pattern
- **Syntax:** column\_name/ expression not like 'pattern\_to\_match';

**WQTD fname from emp table if the emp fname is not starting with s**

```
select fname
```

```
from emps
```

```
where fname not like 's%';
```

**WQTD details of emps if the emps were not born during the year 1995**

```
select *
```

```
from emps
```

```
where dob not like '1995%';
```

**WQTD details of emps whose fname is not starting with m and k**

```
select *
```

```
from emps
```

```
where fname not like 'm%' and fname not like 'k%';
```

**WQTD fname, lname if emp name is starting with vowels and lname is not ending with vowels**

```
select fname, lname
```

```
from emps
```

```
where (fname like 'a%' or fname like 'e%' or fname like 'i%' or fname like 'o%' or fname like 'u%') and lname not like '%a' and lname not like '%e' and lname not like '%i' and lname not like '%o' and lname not like '%u';
```

**WQTD fname, lname if lname contains exactly 3 characters**

```
select fname, lname
```

```
from emps
```

```
where lname like '___';
```

**WQTD details of emps if the emp fname last 3<sup>rd</sup> character is e and lname last character is r and working as waiter or manager but not as female.**

```
select *
```

```
from emps
```

```
where fname like '%e__' and lname like '%r' and job in ('waiter', 'manager') and gender! = 'F';
```

**WQTD details of emps from pentagon table if name consists at least 1% in it**

```
+-----+-----+
```

```
| id | name |
```

```
+-----+-----+
```

```
| 1 | rak%shith |
```

```
| 2 | su_hail |
```

```
| 3 | sa%ndh%ya |
```

```
+-----+-----+
```

**Using this table the output should be rakshith and sandhya because it contains percentage symbol**

```
select *  
  
from pentagon  
  
where name like '%\%%\%%';
```

**Note:**

\ : To remove special behaviour from special character we need to use use backslash(/) in front of the special character.

**Wqtd name of emps from pentagon table if name contains atleast 2%**

```
select name from pentagon  
  
where name like '%\%%\%%\%%';
```

**Wqtd name of emps from pentagon table if name 3<sup>rd</sup> character is \_**

```
select name from pentagon  
  
where name like '__\_%';
```

---

**FUNCTION**

It is a set of instructions/ block od codes to perform some specific task

**Function**

**User defined Function**

- Procedure
- Trigger

**Pre defined Function**

- Aggregate Function
- Character Function
- Number Function
- Date Function

**Aggregate Function/ group Function/ multi-row function**

**Types of Aggregate Function**

**1. Max ()**

- It is used to obtain maximum value from given column
- **Syntax:** select max(column\_name) from table\_name;
- **Example:**  
select max(sal)

```
from emps;
```

## 2. Min ()

➤ It is used to obtain minimum value from given column

➤ **Syntax:** min(column\_name/Expression)

➤ **Example:**

```
select min(sal)
```

```
from emps;
```

## 3. Avg ()

➤ It is used to obtain average value from the given column

➤ **Syntax:** avg(column\_name/Expression)

➤ **Example:**

```
select avg(sal)
```

```
from emps;
```

## 4. Sum ()

➤ It is used to obtain total value from the given column

➤ **Syntax:** sum(column\_name/Expression)

➤ **Example:**

```
select sum(sal)
```

```
from emps;
```

## 5. Count ()

➤ It is used to obtain number of values present in given column

➤ **Syntax:** count(\* /column\_name/Expression)

➤ **Example:**

```
select count (*)
```

```
from emps;
```

**Note:** Only for Count () we can use \* as an argument

## Characteristics of Aggregate Functions

1. It takes n number of inputs but generates single output
2. It executes Group by Group
3. We can't use normal columns along with aggregate function inside select clause.
4. We can use only one column as a argument for aggregate function.
5. We can't nest aggregate functions.

6. It ignores null values.
7. We can't use aggregate function in where clause
8. We can use group by expression along with aggregate function inside select clause

**WQTD Total salary given to the employees who are working as delivery**

```
select sum(sal)

from emps

where job='delivery';
```

**WQTD average salary, total salary, minimum salary and maximum salary given to the employees whose fname starts with k or a**

```
select avg(sal), sum(sal), min (sal), max(sal)

from emps

where fname like 'k%' or fname like 'a%';
```

**WQTD number of emps working as delivery or manager**

```
select count (*)

from emps

where job="delivery" or job="manager";

Or

where job in ('delivery', 'manager');
```

**WQTD number of unique job roles from emps table**

```
select count (distinct job)

from emps;
```

**WQTD number of employees getting salary more than 50000 and born during the date 12 and not getting commission**

```
select count (*)
```

from emps

where sal > 50000 and dob like '%12' and comm is null;

---

## Group by:

It is used to create groups

### Characteristics of group by clause

- It executes row by row
- It executes after from clause (if there is no where clause)
- With or without using where clause we can use group by clause
- It executes row by row after execution it will create group only
- We can use multiple columns along with group by clause, it will create the groups based on combination
- Any clause which executes after group by clause will execute group by group only.

### Group by Expression

The columns which are passing inside group by clause will be considered as group by expression

### Order of Execution

1. From
2. [where]
3. Group by
4. Select

**GROUP BY :-** It is used to create groups.

Emp	ID	NAME	JOB	SAL
P-R	1	Romeo	waiter	10000
P-R	2	Majnu	CHEF	3000
P-R	3	Juliet	waiter	2000
P-R	4	Laila	CHEF	8000
P-R	5	Derdas	Salesman	4000

**Job = Waiter**

1	Romeo	waiter	10000
3	Juliet	waiter	2000

2

**Job = CHEF**

2	Majnu	chef	3000
4	Laila	chef	8000

2

**Job = Salesman**

5	Derdas	Salesman	4000
---	--------	----------	------

1

**WQTD Number of Emps work in each Job role?**

③ Select (count(\*), Job  
① From Emp  
② Group by Job :

Group by Expression

d/p

count(*)	Job
2	waiter
2	chef
1	Salesman

### **WQTD number of emps working in each job role**

```
select job, count(*)  
  
from emps  
  
group by job;
```

### **wqtd number of employees per status**

```
select status, count(*)  
  
from emps  
  
group by status;
```

### **wqtd total salary spent by the company in each job if job roles are delivery, manger or waiter**

```
select sum(sal),job  
  
from emps  
  
where job in('delivery','manager','waiter')  
  
group by job;
```

### **Wqtd number of emps working in each location and getting salary more than 32000 and less than 50000**

```
select count(sal),lid  
  
from emps  
  
where sal>32000 and sal<50000  
  
group by lid;
```

## **HAVING**

It is used to filter the group function

### **Characteristics of Having clause**

- It executes group by group
- It executes after group by clause

- It evaluates true or false condition
- We can pass multiple condition inside having clause
- We can't use normal columns inside having clause

### Order of execution

1. From
2. [where]
3. Group by
4. Having
5. Select

**HAVING :- To filter Aggregate Fn.**

Emp	Name	Job	Sal	dp where	dp Group by	dp Having
P-R	Nani	chef	10000	T	Job = chef Nani chef 10000 SRK chef 2000	10000 > 4000 T
P-R	Ram	waiter	1000	F		
P-R	SRK	chef	2000	T	Job = waiter Prabhas waiter 8000	8000 > 4000 T
P-R	Prabhas	waiter	8000	T		
P-R	balayya	Security	4000	T	Job = Security balayya Security 4000	4000 > 4000 F

**World Maximum Salary** (PENTAGON SQL LIVE)  
in each Job if Employee Salary is more than 1000 And maximum Salary is more than 4000.

- ⑤ Select max(sal), Job
- ① From Emp
- ② Where Sal > 1000
- ③ Group by Job
- ④ Having max(sal) > 4000 ;

dp	max(sal)	Job
	10000	chef
	8000	waiter

**Wqtd maximum salary of emps per lid if the emp is getting salary more than 25000 and maximum salary more than 50000**

select max(sal), lid

from emps

where sal > 25000

group by lid

having max(sal) > 50000;



**wqtd average salary and total salary obtained in each location if the average salary of the location is more than 40000**

```
select avg(sal),sum(sal),lid
```

-> from emps

-> group by lid

-> having avg(sal)>40000;

**Wqtd total salary and number of emps working in each job if more than 2 emps working in each job role**

```
select sum(sal),count(*),job
```

-> from emps

-> group by job

-> having count(\*)>2;

**Wqtd maximum salary and minimum salary and number of emps working in each location if the location contains atleast 2 emps working in it and employee salary must be more than 320000**

```
select max(sal),min(sal),count(*),lid
```

-> from emps

-> where sal>32000

-> group by lid

-> having count(\*)>=2;

**Wqtd number of emps who are getting same salary /repeated salary**

```
select count(*),sal
```

-> from emps

-> group by sal

-> having count(\*)>1;

## Wqtd number of emps having same gender and working in same job

select count(\*)

-> from emps

-> group by job,gender

-> having count(\*) >1;

## Write difference between where and having clause

Aspect	WHERE Clause	HAVING Clause
Usage	Filters rows <b>before</b> grouping	Filters groups <b>after</b> grouping
Used With	SELECT, UPDATE, DELETE	SELECT (with GROUP BY)
Aggregation	Cannot use aggregate functions (like SUM, AVG)	Can use aggregate functions
Performance	Generally faster, as it filters early	Slower if large groups are created
Example	SELECT * FROM sales WHERE region = 'East';	SELECT region, SUM(sales) FROM sales GROUP BY region HAVING SUM(sales) > 1000;

## Order by

- It is used to arrange records either in ascending order or descending order
- Syntax:

Select column\_name/expression

From table\_name

Order by column\_name asc/desc;

## **Characteristics of order by**

- It is a last executable clause in a query
- It executes after select clause
- By default it will consider ascending order for columns
- Normally in all the table records are arranged in ascending order based on primary key column
- We can use alias name inside order by clause
- We can use multiple columns inside order by clause, it will give the priority for first column order if the values are same for the first column order then it will give the priority for 2<sup>nd</sup> column order

## **Order of Execution**

1. From
2. [Where]
3. Group by
4. Having
5. Select
6. Order by

## **Wqtd details of emps based on their salary maximum to minimum order**

select \*

-> from emps

-> order by sal desc;

## **Wqtd fname,lname and job if the emps are working as security or manager or cleaner and arrange the records according to alphabetical order of their fname**

select fname,lname,job

-> from emps

-> where job='security' or job='manager' or job='cleaner'

-> order by fname asc;

**Wqtd number of emps who are having same gender and working in same job role and emps are getting salary more than 30000 and arrange the job in alphabetical order**

```
select count(*),gender,job
```

-> from emps

-> group by gender,job

-> having count(\*)>1

-> order by job asc;

### **Limit**

➤ It is used to display specific number of records from resultant table

➤ **Syntax:**

Select column\_name/Expression

From table\_name

Limit value;

**Wqtd details of first 3 records from emps table**

```
select *
```

-> from emps

-> limit 3;

**Wqtd details of first record from emps table**

```
select *
```

-> from emps

-> limit 1;

### **Offset**

➤ It is used to skip/ignore specific number of records from resultant table

➤ **Syntax:**

Select column\_name /expression

from table-name

Limit value offset value;

**Wqtd details of 2<sup>nd</sup> record from emps table**

select \*

-> from emps

-> limit 1 offset 1;

**Wqtd details of 5<sup>th</sup> and 6<sup>th</sup> record from the emps table**

select \*

-> from emps

-> limit 2 offset 4;

**Wqtd details of top 5 maximum salary holders**

select \*

-> from emps

-> order by sal desc

-> limit 5;

**Wqtd last 3 records details from emps table**

select \*

-> from emps

-> order by eid desc

-> limit 3;

**Wqtd 2<sup>nd</sup> maximum salary from emps table**

select distinct sal

-> from emps

-> order by sal desc

-> limit 1 offset 1;

### **Wqtd 4<sup>th</sup> minimum salary from emps table**

select distinct sal

-> from emps

-> order by sal asc

-> limit 1 offset 3;

---

## **Character function**

### **Characteristics of character function**

- It takes n number of inputs and generates n number of outputs
- It executes row by row
- We can nest character functions
- We can use character functions inside where clause

### **Types of Character functions**

#### **1. Lower ():**

It is used to convert given string value into lower case

##### **Example:**

Select lower('INDIA'); india

#### **2. Upper () :**

It is used to convert given string value into upper case

##### **Example:**

Select upper('india'); INDIA

#### **3. Length ():**

It is used to obtain total number of characters present in given string

##### **Example:**

Select length('pentagon space');14

**Wqtd 2<sup>nd</sup> longest city as well as their respective length from locations table if there is more than 1 2<sup>nd</sup> longest city choose the one that comes first when we ordered alphabetically**

**Sample i/p:**

**City:**

**Abc**

**Pqrs**

**Abcd**

**Pqr**

**Abcde**

**o/p:**

**Abcd 4**

**select city,length(city)**

**-> from locations**

**-> order by length(city) desc, city asc**

**-> limit 1 offset 1**

#### **4. Reverse ():**

It is used to display given string in reverse format

##### **Example:**

Select reverse ('pentagon'); nogatenep

#### **5. Concat ():**

It is used to add/combine two or more string values

##### **Syntax:**

Concat ('str\_1', 'str\_2');

##### **Example:**

MR/MISS	SHAKILA	YOUR	SALARY	IS	1000	RS.
FNAME			SAL			

*PENTAGON SPACE  
Mastering The Future*

```
select concat('Mr/Miss ',fname,' your salary is ',sal,' rs.')
```

```
-> from emps;
```

### **Wqtd fname and lname as full name from emps table**

```
select concat (fname,' ',lname)
```

```
-> from emps;
```

**30/6/25**

#### **6. substr ():**

It is used to extract some part of the string in original string

##### **Syntax:**

substr ('original string', position, [length]); length is optional

##### **Case 1:**

B	E	N	G	A	L	U	R	U
1	2	3	4	5	6	7	8	9

**Note:** in sql index value starts from 1

##### **Example:**

Select substr('BENGALURU',7,2); UR

Select substr('BENGALURU',5); ALURU

##### **Case 2:**

We also have negative index value in SQL, it starts from right to left

B	E	N	G	A	L	U	R	U
-9	-8	-7	-6	-5	-4	-3	-2	-1

##### **Example:**

Select substr ('BENGALURU', -4,4); LURU

Select substr ('BENGALURU', -7); NGALURU

### **Wqtd details of emps if employee fname is starting with 'a'**

```
select *
```

```
from emps
```

```
where substr(fname,1,1)='a';
```



**Wqtd details of emps if employee lname is ending with 'i'**

```
select *  
  
from emps  
  
where substr(lname,-1,1)='i';
```

**Wqtd fname and job if employee job is starting with sec or man**

```
select fname,job  
  
from emps  
  
where substr(job,1,3)='sec' or substr(job,1,3)='man';  
  
or
```

```
select fname,job  
  
from emps  
  
where substr(job,1,3)in('sec','man');
```

**Wqtd fname and lname if fname is not starting with vowels**

```
select fname,job  
  
from emps  
  
where substr(fname,1,1) not in('a','e','i','o','u');
```

**Wqtd details of employee who are born in the year 1995**

```
select *  
  
from emps  
  
where substr(dob,1,4)='1995';
```

**Wqtd fname,lname,dof if employee joined in the month of April, may, June or July**

```
select fname,lname,dof  
  
from emps  
  
where substr(dof,6,2)in(04,05,06,07);
```

**Wqtd details of employees whose reversed is matching with string 'nama'**

```
select *
```

```
from emps
```

```
where reverse(fname)='nama';
```

**Wqtd extract initials from full name in below format**

**Fname: puneeth**

**Lname:rajkumar**

**Full name: puneeth rajkumar**

**o/p: p.r.**

```
select concat(substr(fname,1,1),',',substr(lname,1,1),',') as initials
```

```
from emps;
```

**Wqtd fname, lname and job together in below format**

**Fname: Sharukh**

**Lname: khan**

**o/p: Sharukh khan(actor)**

```
select concat(fname,',',lname,',',(' ',job,','))
```

```
from emps;
```

**Wqtd first half of fname from emps table**

```
select substr(fname,1,length(fname)/2)
```

```
from emps;
```

**Wqtd second half of fname from emps table**

```
select substr(fname,length(fname)/2+1)
```

```
from emps;
```

**wqtd first half of fname in lowercase and second half of fname in reverse format**

**samp o/p:**

**Sameer: samree**

select

```
concat(lower(substr(fname,1,length(fname)/2)),reverse(substr(fname,length(fname)/2+1))) as  
name from emps;
```

**Wqtd fname ans password for emps,password must conatin below conditions**

- i. First 3 characters of fname**
- ii. Length of fname**
- iii. Last 3 digits od their job role**

**samp i/p:**

**Fname: Shakila**

**Job: waiter**

**Samp o/p:**

**Fname: Shakila**

**Password:sha7ter**

```
select fname,concat(substr(fname,1,3),length(fname),substr(job,-3)) as password  
from emps;
```

## **7. REPLACE**

It is used to replace substring from new string in original string

**Syntax:**

```
replace('original string', 'sub string', new string');
```

**Example:**

```
Select replace('pentagon','pent','hex'); hexagon
```

```
select replace('penty','n','') as replaces; petyy
```

**wqtd replace a and I with [a] and [i] in kiran ??**

**o/p: k[i]r[a]n**

```
select replace(replace('kiran','a','[a]'),'i','[i]);
```

**wqtd count of character a in Malayalam**

```
select length('malayalam')- length(replace('malayalam','a',''));
```

**wqtd details of emps if name contains exactly 1**

```
select * from emps
```

```
-> where length(fname) - length(replace(fname,'A','')) = 1;
```

---

## **Number Functions**

### **Characteristics of Number Functions**

- It takes n number of inputs and generates n number of outputs
- It executes row by row
- We can nest number functions
- We can use number functions inside where clause

### **Types of Number Functions**

#### **1. Abs ():**

It is used to convert a negative number to positive number

##### **Example:**

```
Select abs (-18); 18
```

```
Select abs (18); 18
```

#### **2. Mod ():**

- It is used to obtain remainder value

- **Syntax:**

```
Mod (m,d);
```

- **Example:**

```
Mod (8,2);0
```

## Wqtd details of emps if emps are having even eid

select \*

-> from emps

-> where mod(eid,2)=0;

### 3. Round():

It is used to round off a number upto specified number of decimal places

#### Syntax:

Round(number, decimal\_places)

#### Example:

Select round (123.4);123

Select round (123.5);124

Select round (123.456,2);123.46

Select round(123.45645,3);123.456

## Wqtd average salary obtained in each job and round off the average salary upto 2<sup>nd</sup> decimal place

select round(avg(sal),2), job

-> from emps

-> group by job;

### 4. Ceil ():

➤ It will obtain next integer value from the given decimal value (if it is a positive number)

➤ It will obtain current integer value from the given decimal value ( if it is a negative number)

#### ➤ Example:

select ceil(4.5); 5

select ceil(-4.5); -4

### 5. Floor() :

➤ It will obtain current integer value from the given decimal value (if it is a positive number)

- It will obtain next integer value from the given decimal value ( if it is a negative number)

- **Example:**

```
select floor (5.1); 5  
select floor (-5.1); -6
```

## 6. Truncate():

- It is used to cut off a number upto specified number of decimal places without rounding it.

- Syntax:

```
Truncate(number, decimal_place);
```

- **Example:**

```
select truncate(123.4567,2); 123.45  
select truncate(123.456733,4); 123.4567  
select truncate(123.456733);  
ERROR 1064 (42000): You have an error in your SQL syntax;
```

## 7. Pow():

- It is used to obtain power value of a number

- **Example:**

```
select pow(8,2);64  
select pow(2,3);8
```

## 8. Sqrt():

- It is used to obtain root value of a non negative number

- **Example:**

```
select sqrt(4);2  
select sqrt(-4); NULL
```

---

## Date Functions

### Characteristics of Number Functions

- It takes n number of inputs and generates n number of outputs
- It executes row by row
- We can nest date functions
- We can use date functions inside where clause

## Types of Date Functions

### 1. Curdate()

- It is used to obtain current date from the system

- **Example:**

```
select curdate (); 2025-07-09
```

### 2. Sysdate () / now():

- It is used to obtain current date and time from the system

- **Example:**

```
select sysdate();
```

```
select now(); 2025-08-05 17:38:30
```

### 3. Year ():

- It is used to extract year from given date expression

- **Example:**

```
select year('2024-10-12'); 2024
```

```
select year(dob)
```

```
from emps;
```

### 4. Month():

- It is used to extract month from given date expression

- **Example:**

```
select month('2024-10-12'); 10
```

```
select month(dob)
```

```
from emps;
```

### 5. Day():

- It is used to extract day from given date expression

- **Example:**

```
select day('2024-10-12'); 12
```

```
select day(dob)
```

```
-> from emps;
```

### Wqtd details of the emps if the emps before the year 1995

select \*

-> from emps

-> where year(dob)<1995;

### Wqtd details of emps if the employees ired in the month of April, janor feb

select \*

-> from emps

-> where month(doj) in (4,1,2);

### Wqtd details of emps if employees hired in leap year

select \*

-> from emps

-> where mod(year(doj),4)=0;

-----  
10/7/25

#### 6. Datediff ():

It is used to obtain day difference between two values

##### Example:

```
select datediff('2025-07-10','2025-07-09');
```

```
+-----+
| datediff('2025-07-10','2025-07-09') |
+-----+
|                1 |
+-----+
```

#### 7. Date\_add ():

It is used to add some time interval for given dataae value

##### Syntax:

```
date_add ('date value', interval value unit);
```

**Interval:** It is a keyword used to add/subtract some time interval



**Value:** The amount of time which we are adding

**Unit:** unit of time interval( ex: year, month, day)

**Wqtd add 3 year for below date**

2020-10-11

```
select date_add ('2020-10-11', interval 3 year);
```

**Wqtd add 5 year 6 months for below date**

'2001-12-10'

```
select date_add('2001-12-10',interval 66 month);
```

or

```
select date_add(date_add('2001-12-10',interval 5 year), interval 6 month);
```

**8. Date\_sub:**

It is used to subtract some time interval from given date value

**Syntax:**

Date\_sub('date value', interval value unit);

**Wqtd subtract 3 year 10 months and 15 days from below date**

```
select date_sub(date_sub(date_sub('2025-01-24',interval 3 year),interval 10 month), interval 15 day);
```

**9. Date\_format() :**

It is used to extract individual characters from given date-time expression

**Syntax:**

Date\_Format ('date value', 'date\_Format\_pattern');

YYYY	: '%Y'	SELECT DATE_FORMAT(NOW(), '%Y');	2025
YY	: '%y'	SELECT DATE_FORMAT(NOW(), '%y');	25
MONTH	: '%M'	SELECT DATE_FORMAT(NOW(), '%M');	JULY
MM	: '%m'	SELECT DATE_FORMAT(NOW(), '%m');	07
MON	: '%b'	SELECT DATE_FORMAT(NOW(), '%b');	JUL
DD	: '%d'	SELECT DATE_FORMAT(NOW(), '%d');	11
DAY	: '%W'	SELECT DATE_FORMAT(NOW(), '%W');	FRIDAY
DAY	: '%a'	SELECT DATE_FORMAT(NOW(), '%a');	FRI
HH24	: '%H'	SELECT DATE_FORMAT(NOW(), '%H');	17
HH12	: '%h'	SELECT DATE_FORMAT(NOW(), '%h');	05
MINUTE	: '%i'	SELECT DATE_FORMAT(NOW(), '%i');	58
SECONDS	: '%s'	SELECT DATE_FORMAT(NOW(), '%s');	05
TIME	: '%T'	SELECT DATE_FORMAT(NOW(), '%T');	17:59:47
AM/PM	: '%p'	SELECT DATE_FORMAT(NOW(), '%p');	PM
TIME			
AM/PM	: '%r'	SELECT DATE_FORMAT(NOW(), '%r');	06:01:34 PM

**Wqtd current date and time in below format**

**'25-July-11 05 pm Friday'**

```
select date_format(now(), '%y-%M-%d %h %p %W');
```

**wqtd fname and dob in us date format**

**us date format: MM-dd-yyyy**

```
select fname, date_format(dob, '%m-%d-%Y')
```

-> from emps;

**Wqtd details of the emps were hired on Friday, Saturday or Sunday**

```
select *
```

-> from emps

-> where date\_format(doj, '%W') in ('friday', 'saturday', 'sunday');

Or

```
select *
```

-> from emps

-> where date\_format(doj,'%a') in ('fri','sat','sun');

**Wqtd fname and current experience of all the emps in terms of year**

```
SELECT FNAME,DATE_FORMAT(CURDATE(),'%Y')-DATE_FORMAT(DOJ,'%Y') EXP  
FROM EMPS;
```

**Wqtd number of emps hired in each month and display the numbers maximum to minimum order based on their count**

select count(\*), month(doj)

-> from emps

-> group by month(doj)

-> order by count(\*) desc;

**Wqtd last hired employee doj**

select max(doj)

-> from emps;

Or

select doj

-> from emps

-> order by doj desc

-> limit 1;

**Wqtd add 2 year for first hired employee doj**

select date\_Add(min(doj),interval 2 year)

-> from emps;