**Day 12: 31-10-2025:**

**This below link contains sample tables**

[**https://github.com/Kaleakash/mysql\_tables.git**](https://github.com/Kaleakash/mysql_tables.git)

Where clause: filter the data

1. Relational operator >, <, >=, <=, =, !=

select \* from employee where salary > 15000;

1. Between operator : range of data

select \* from employee where salary between 5000 and 10000;

select \* from employee where hire\_date between '1995-01-01' and '1998-12-31';

1. In operator : multiple value with condition

select \* from employee where job\_id in('SA\_REP','ST\_CLERK');

1. Like : like equal with special features as start with, end with, contains etc with %

select \* from employee where first\_name ='steven';

select \* from employee where first\_name like 'steven';

% means 0 or 1 or many

Start with s character

select \* from employee where first\_name like 's%';

end with a character

select \* from employee where first\_name like '%a';

contains g character

select \* from employee where first\_name like '%g%';

\_

Only one character anything

select \* from employee where first\_name like 's\_e%';

1. is null

select \* from employee where manager\_id is null;

select \* from employee where manager\_id is not null

**Order by clause**

It is use to sort the data in ascending or descending order

select \* from employee order by salary asc;

select \* from employee order by salary desc;

select \* from employee order by first\_name asc;

select \* from employee order by first\_name asc;

Join:

Join is use to retrieve more than one column from more than one table with or without conditions.

1. Cross join or cartesian product

M \* N

select first\_name,salary,department\_name from employee, department;

select first\_name,salary,department\_id,department\_name from employee, department;

error

select employee.first\_name,employee.salary,department.department\_id,department.department\_name from employee, department;

**we can table alias**

select emp.first\_name,emp.salary,dept.department\_id,dept.department\_name from employee emp, department dept;

**join types**

1. **Inner join (Equi join) :**

it display common records present in both the tables.

select emp.first\_name,emp.salary,dept.department\_id,dept.department\_name from employee emp inner join department dept on emp.department\_id=dept.department\_id;

1. Outer join
   1. Left outer join

Common as well as left table remaining record

**select emp.first\_name,emp.salary,dept.department\_id,dept.department\_name from employee emp left outer join department dept on emp.department\_id=dept.department\_id;**

* 1. Right outer join

Common as well as right table remaining record

**select emp.first\_name,emp.salary,dept.department\_id,dept.department\_name from employee emp right outer join department dept on emp.department\_id=dept.department\_id;**

* 1. Full join

Common + left and right remaining records

**Self join :** joining the same table is known as self join.

**select e1.first\_name as EmployeeName,e2.first\_name as ManagerName from employee e1 inner join employee e2 on e1.manager\_id=e2.employee\_id;**

**My SQL functions**

Function contains set of instruction to perform specific task.

Function mainly divided into 2 types

1. Pre defined or built in function
2. User defined function (PL SQL )

Pre defined function : mainly divided into 2 types

1. Single row function

The function functionality apply for each row independently.

1. String function

select length('Welcome to Java Training') as Result;

select upper('Welcome to Java Training') as Result;

select lower('Welcome to Java Training') as Result;

1. Number function

select round(4567.567,2) as Result;

select round(4567.567,1) as Result;

select round(4567.567,0) as Result;

select truncate(4567.567,2) as Result;

select truncate(4567.567,1) as Result;

1. Date function

select now();

select curdate();

select curtime();

select round(datediff(now(),hire\_date)/365,0) from employee;

select date\_format(now(),'%d-%m-%y');

select date\_format(now(),'%d-%m-%Y');

select date\_format(now(),'%d-%M-%Y');

1. Multi row function or aggregate functions

The function functionality apply for more than one records using group by clause. By default whole table consider as one group.

Sum()

Max()

Min()

Avg()

Count()

select sum(salary) as total\_salary from employee;

select max(salary) as max\_salary from employee;

select min(salary) as min\_salary from employee;

select avg(salary) as avg\_salary from employee;

select count(salary) as number\_of\_emp from employee;

select count(employee\_id) from employee;

select count(\*) from employee;

whole table one group consider.

if we want to make sub group then we need to use group by clause.

select sum(salary) from employee group by department\_id;

select sum(salary) from employee group by job\_id;

select count(salary) from employee group by manager\_id;

select department\_id,sum(salary) from employee group by department\_id;

select manager\_id,count(salary) from employee group by manager\_id;

group by with where clause

select department\_id,sum(salary) from employee where department\_id is not null group by department\_id;

where clause vs having clause: both are same while applying the condition. But where clause apply for individual records which we can use before group by or without group by. Having clause apply individual as well as group of records which we can use after group by.

Query with where clause, group by and having clause

select department\_id,sum(salary) from employee where department\_id is not null group by department\_id having sum(salary) > 55000;

all clause simple example

select department\_id,sum(salary) from employee where department\_id is not null group by department\_id having sum(salary) > 55000 order by sum(salary) asc;

DDL

Syntax

create table tableName(

columnName dataType1,

columnName dataType2,

columnName dataType3

);

While creating table to insert valid data you need to use constraints.

Primary key

Not null

Check constraints

Unique

Enum

Foreign key

Default

Employees

Id FName LName Salary gender age city phonenumber

create table employees(employee\_id int primary key,fname varchar(20) not null, lname varchar(30), salary float check (salary>20000),gender enum('Male','Female'), city varchar(30) default 'Bangalore',phonenumer varchar(10) unique);

insert into employees(employee\_id,fname,lname,salary,gender,city,phonenumer) values(100,'Raj','Deep',22000,'Male','Mumbai','9876536787');

mysql> insert into employees(employee\_id,fname,lname,salary,gender,city,phonenumber) values(100,'Raj','Deep',22000,'Male','Mumbai','9876536787');

ERROR 1054 (42S22): Unknown column 'phonenumber' in 'field list'

mysql> insert into employees(employee\_id,fname,lname,salary,gender,city,phonenumer) values(100,'Raj','Deep',22000,'Male','Mumbai','9876536787');

Query OK, 1 row affected (0.00 sec)

mysql> insert into employees(employee\_id,fname,lname,salary,gender,city,phonenumer) values(100,'Raj','Deep',22000,'Male','Mumbai','9876536787');

ERROR 1062 (23000): Duplicate entry '100' for key 'employees.PRIMARY'

mysql> insert into employees(employee\_id,fname,lname,salary,gender,city,phonenumer) values(101,'Raj','Deep',22000,'Male','Mumbai','9876536787');

ERROR 1062 (23000): Duplicate entry '9876536787' for key 'employees.phonenumer'

mysql> insert into employees(employee\_id,fname,lname,salary,gender,city,phonenumer) values(101,'Raj','Deep',22000,'Male','Mumbai','9876536785');

Query OK, 1 row affected (0.01 sec)

mysql> insert into employees(employee\_id,fname,salary,gender,city,phonenumer) values(102,'Ravi',22000,'Male','Mumbai','9876536785');

ERROR 1062 (23000): Duplicate entry '9876536785' for key 'employees.phonenumer'

mysql> insert into employees(employee\_id,fname,salary,gender,city,phonenumer) values(102,'Ravi',22000,'Male','Mumbai','9876536783');

Query OK, 1 row affected (0.00 sec)

mysql> insert into employees(employee\_id,fname,salary,gender,city,phonenumer) values(103,'Ajay',12000,'Male','Mumbai','9876536383');

ERROR 3819 (HY000): Check constraint 'employees\_chk\_1' is violated.

mysql> insert into employees(employee\_id,fname,salary,gender,city,phonenumer) values(103,'Ajay',27000,'Male','Mumbai','9876536383');

Query OK, 1 row affected (0.01 sec)

mysql> insert into employees(employee\_id,fname,salary,gender,city,phonenumer) values(104,'Balaji',27000,'Unkown','Mumbai','9476536383');

ERROR 1265 (01000): Data truncated for column 'gender' at row 1

mysql> insert into employees(employee\_id,fname,salary,gender,city,phonenumer) values(104,'Veena',27000,'female','Mumbai','9476536383');

Query OK, 1 row affected (0.01 sec)

mysql> insert into employees(employee\_id,fname,salary,gender,phonenumer) values(104,'Veena',27000,'female','9446536383');

ERROR 1062 (23000): Duplicate entry '104' for key 'employees.PRIMARY'

mysql> insert into employees(employee\_id,fname,salary,gender,phonenumer) values(105,'Neena',27000,'female','9446536383');

Query OK, 1 row affected (0.01 sec)

DML

Insert

Delete

Update

Update query

update employees set lname='Kumar' where employee\_id=102;

delete from employee where employee\_id=105;

**drop table tableName;**

drop table sample; it remove all records as well as table structure

**truncate table tableName;**

truncate table sample;

delete Vs truncate vs drop

DML DDL DDL

All records delete all records delete all record as well

Not able structure Structure of table.

Can use where clause can’t can’t

We can use TCL on DML but can’t on DDL.

Alter command

Using alter we can modify the table structure

Adding new column to existing table

alter table employees add desg varchar(2);

modify the existing column information

alter table employees modify desg varchar(10);

drop column from existing table

alter table employees drop column lname;

rename column name in existing table

alter table employees rename column desg to designation

**Relationship**

One to One -🡪 PK 🡪 FK or Shared PK

One person has one pancard

create table person(pid varchar(10) primary key,pname varchar(10));

create table pancard(panid varchar(10) primary key, pannumber varchar(10), foreign key(panid) references person(pid));

mysql> insert into person values('A1','Raj');

Query OK, 1 row affected (0.00 sec)

mysql> insert into person values('A2','Ravi');

Query OK, 1 row affected (0.00 sec)

mysql> insert into pancard values('A1','AA4567JP');

Query OK, 1 row affected (0.01 sec)

insert into pancard values('A2','AA5353GH');

Query OK, 1 row affected (0.00 sec)

**One to Many 🡪 PK 🡪 FK**

Trainer and Student

**create table trainer(tid int primary key,tname varchar(30), tech varchar(30));**

**create table student(sid int primary key,sname varchar(30), age int, tsid int, foreign key(tsid) references trainer(tid));**

mysql> insert into trainer values(1,'Raj','Java');

Query OK, 1 row affected (0.01 sec)

mysql> insert into trainer values(2,'Ravi','Python');

Query OK, 1 row affected (0.01 sec)

insert into trainer values(3,'Ram','ReactJS');

Query OK, 1 row affected (0.01 sec)

mysql> insert into student values(100,'Meeta',21,1);

Query OK, 1 row affected (0.01 sec)

mysql> insert into student values(101,'Veeta',22,1);

Query OK, 1 row affected (0.01 sec)

mysql> insert into student values(102,'Leeta',23,2);

Query OK, 1 row affected (0.01 sec)

insert into student values(103,'Geeta',24,null);

Query OK, 1 row affected (0.00 sec)

select t.tname,t.tech,s.sname from trainer t inner join student s on t.tid =s.tsid;

Many to One -🡪

Many to Many -🡪

One to many 🡪 bidirectional

Students

Sid PK

SName

1. Raj
2. Ravi

Technologies

TId PK

TName

100 Java

101 Python

StudentTechnologeis

STId int PK auto increment 1 2

Sid 🡪 FK 1 1 2 2

TId -🡪 FK 100 101 100 101

**TCL: Transactional control language**

**Commit;**

**Rollback**

**Savepoint**

Limit and offset

**Limit to display top record**

select first\_name,salary from employee limit 2;

offset skip number of records

**select employee\_id,first\_name,salary from employee limit 2 offset 4;**

select employee\_id,first\_name,salary from employee order by salary desc limit 1;

select employee\_id,first\_name,salary from employee order by salary asc limit 1;

**row\_number()** it provide unique number for each employee under the partition of department/job\_id/manager\_id order of salary may be asc and desc.

select first\_name,department\_id,salary,row\_number() over (partition by department\_id order by salary desc) as row\_id from employee;

**rank():** gives same rank for same salary, but skips the next number

select first\_name,department\_id,salary,rank() over (partition by department\_id order by salary desc) as row\_id from employee;

dense\_ran() : gives same rank for ties, but doesn’t skip the next rank