**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

**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;