8 days:

**Day 8**

**Database:**

**MySQL :**

MySQL function : function is use to write the set of instruction to perform a specific task.

My SQL provided different types of function. Mainly divided into 2 types.

1. pre defined function
2. user defined function (PL SQL).

**Pre defined or built in function**

Mainly divided into 2 types.

1. Single row functions

The function functionality apply for each record individually.

String function

Upper

Lower

Substring

Concat

Length

Etc

**select pname,upper(pname) as upperCase, lower(pname) as lowerCase,substring(pname,2,5) substring from product;**

**select length('welcome to mysql trainining');**

Number function

select abs(-10);

select round(56000.4678,3);

select round(56000.4678,2);

select round(56000.4678,1);

select round(56000.4678,0);

select truncate(56000.4678,3);

select truncate(56000.4678,2);

select truncate(56000.4678,1);

select truncate(56000.4678,0);

Date function

Now() this display current date and time

curDate() this display date

curtime() this display time

date\_format(date,format):

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

select date\_format(now(),'%d-%m-%Y %h:%i:%s');

datediff(): this function display number of days between two dates

select datediff('2025-06-01','2025-04-01');

select datediff(now(),'2025-04-01');

employee table contains hire\_date.

Please take the help of datediff function and display first\_name and hire\_date in dd-mm-yyyy format of only those employee whose experience is > 30 year.

select upper(first\_name) employee\_name,date\_format(hire\_date,'%d-%M-%Y') hire\_date,round(datediff(now(),hire\_date)/365,0) yearOfExp from employee;

<https://dev.mysql.com/doc/refman/8.4/en/built-in-function-reference.html>

**Multi row function or aggregate function**

These function functionalities apply for more than one records using group concept. By default whole table consider as one group. Using group by clause we can make sub 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 total\_number\_of\_Employee from employee;

count function can ignore null value.

While using count function better pass the column name which contains pk or \*.

select count(\*) as total\_number\_of\_Employee from employee;

**group by clause**

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

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

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

multi group

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

**having clause**

having clause is like a where clause. But Having clause we use after group by. Where clause we use before group by. Where clause apply the condition for individual record and having clause apply the condition for group of records.

Below query cover all DRL or DQL clause

**select department\_id,sum(salary) from employee where department\_id is not null group by department\_id having sum(salary) > 50000 order by department\_id desc;**

**Relationship**

**Primary key** : single table we can create only one column as PK. Pk column doesn’t allow duplicate as well as it doesn’t allow null values.

**Foreign key**: Foreign key is use to connect to PK of different as well as same table. If column is FK that column allow only those values which present in PK. It can allow duplicate but only those values present in PK column. It can allow null values.

One to many -🡪 One Trainer and Student. One trainer can handle many students.

**Trainer**

**TId(PK)** TName tech

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

mysql> **insert into trainer values(100,'Raj','Java');**

Query OK, 1 row affected (0.01 sec)

mysql> **insert into trainer values(101,'Ravi','Python');**

Query OK, 1 row affected (0.00 sec)

mysql> **insert into trainer values(102,'Rajesh','ReactJS');**

Query OK, 1 row affected (0.00 sec)

**Student**

**SID(PK)** SName age **TSID(FK)**

**create table student(sid int primary key,sname varchar(30), age int,**

**tsid int, foreign key(tsid) references trainer(tid));**

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

Query OK, 1 row affected (0.01 sec)

mysql> **insert into student values(2,'Veena',22,100);**

Query OK, 1 row affected (0.00 sec)

mysql> **insert into student values(3,'Teena',23,102);**

Query OK, 1 row affected (0.00 sec)

mysql> **insert into student values(4,'Leena',24,null);**

Query OK, 1 row affected (0.00 sec)

**Select \* from trainer**

**Select \* from student**

Inner join with Trainer and student

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

left outer join with Trainer and Student

**select t.pname,t.tech,s.sname from trainer t left outer join student s on t.tid=s.tsid;**

right outer join with Trainer and Student

**select t.pname,t.tech,s.sname from trainer t right outer join student s on t.tid=s.tsid;**

One to One -🡪 One Person has only one passport

Using foreign key

Using shared primary key

Passport

**PPID(PK) expiry**

**create table passport(pid varchar(30) primary key,expiry\_date timestamp);**

mysql> insert into passport values('ABC123321',now());

Query OK, 1 row affected (0.01 sec)

mysql> insert into passport values('XYZ123321',now());

Query OK, 1 row affected (0.00 sec)

mysql> select \* from passport;

Person

**PID(PK as well as FK) Pname**

**create table person(pid varchar(30) primary key,pname varchar(30), foreign key(pid) references passport(pid));**

**insert into person values('ABC123321','Steven');**

**insert into person values('XYZ123321','John');**

Many to Many 🡪 Many students known more than one technologies or courses.

Students

SID(PK) SName

1. John
2. Steven

create table students(sid int primary key,sname varchar(30));

insert into students values(1,'John');

insert into students values(2,'Steven');

Courses

CI(PK) CName

100 Java

101 Python

create table courses(cid int primary key,cname varchar(30));

insert into courses values(100,'Java');

insert into courses values(101,'Python');

Students\_Courses\_relationship

Srno(PK) Sid(FK) CI(FK)

111 1 100

222 1 101

333 2 100

create table students\_courses\_relationship(srno int primary key,sid int, cid int, foreign key(sid) references student(sid), foreign key(cid) references courses(cid));

insert into students\_courses\_relationship values(111,1,100);

insert into students\_courses\_relationship values(222,1,101);

insert into students\_courses\_relationship values(333,2,100);

Many to One 🡪 Many Employees working in one project or part of one department.

1. Multi row functions