**Day 6: 23 February 2025**

create database databasename

create database mytestdb; it is use to create new database.

use mytestdb; now we are inside a new database.

show tables;

**DML :**

Update query:

**Syntax**

Update tablename set columname =value;

update employee set salary=45000;

All employee salary updates to 45000

Update with where clause

**Syntax**

update tableName set columnname = value where columName = value;

update employee set salary=48000 where employee\_id=100;

update employee set salary=50000 where dob between '2020-01-01' and '2022-12-31';

update employee set age=48 where age >=45 and employee\_id=100;

update employee set first\_name = 'John seena',age=50 where employee\_id=100;

**Delete query**

Syntax

Delete from tableName; it delete all records from a table

delete from employee;

delete with where clause

syntax

delete from tableName where clause

delete from employee where employee\_id=109;

delete from employee where age >=45;

DDL language

Sample

Id msg

Int pk varchar(500)

**create table sample(id int primary key,msg varchar(500));**

**insert into sample values(1,'Hi');**

**insert into sample values(2,'Hello');**

**insert into sample values(3,'How r you');**

**insert into sample values(4,'What are you doing');**

if we want to remove the table

drop table tableName; it use to remove table structure as well as all records present insdie that table.

drop table sample;

delete from tableName; it remove all records from table but table structure present in database. alter : alter is part of DDL which is use to change structure of the table.

1. Add new column without drop the table.

Syntax

alter table tableName add columnname datatype

alter table employee add desg varchar(2);

after added you can update value for new column using update query

update employee set desg='HR' where employee\_id=101;

1. Modify column data types

Syntax

alter table tableName modify columName datatype

alter table employee modify desg varchar(10);

update employee set desg ='Developer' where employee\_id in(104,105,106);

1. Remove column

Syntax

alter table tablename drop column columnName

1. Rename column name

Syntax

alter table tableName rename column oldcolumname to recolumname

alter table employee rename column dob to doj;

alter table employee rename column desg to designation;

truncate : it is use to remove all records from a table but still it maintain the table structure.

Syntax

truncate table tableName;

truncate table sample;

**delete vs truncate vs drop**

1. Delete is a part of DML and truncate and drop are part of DDL.
2. Drop remove all records as well as table structure but delete and truncate remove only records table structure maintains in database.
3. Using delete we can use where clause but with truncate we can’t use where clause
4. If we use truncate to remove all records those records removed permanently. But if we deleted records using delete query we can get back those records (we can do undo).
5. On truncate we can’t use TCL command but DML we can use TCL commands.

TCL : Transactional control language

TCL contains two commands commit and rollback.

If we execute more than one DML query on same table or different table. If all query executed successfully then you can do commit under one truncation.

If any of the query fail you need to do rollback that truncation.

Table -🡪 Account

accno 🡪 Pk auto\_increment, name varchar(30), amount float

create table account(accno int primary key auto\_increment, name varchar(30), amount float);

insert into account(name,amount) values('Steven',500);

insert into account(name,amount) values('John',500);

insert into account(name,amount) values('Lex',500);

insert into account(name,amount) values('Bob',700);

**1st successful transaction**

start transaction;

update account set amount = amount-100 where accno=1;

update account set amount = amount+100 where accno=2;

commit;

**1stfailure transaction**

start transaction;

update account set amount = amount-100 where accno=3;

update account set amount = amount+100 where accno=10;

rollback;

**Database constraints and relationship**

Employees

Empid(PK), fname (not null) lname , emailid (unique) with not null,

Constrains mainly use to restrict the user or programmer to inset invalid data.

1. Primary key : primary key is use to maintain the unique record in particular column. In single table we can use only one column as PK. If column is PK that column doesn’t allow duplicate as well as null value (we can’t leave blank)).
2. Not null : not null contains we can’t leave blank or we can’t insert null value.
3. Unique : unique contains is use to store unique value like primary key. Single table we can create more than one column as unique. Unique constraints by default allow null value.
4. Check constraints : we can use where clause with check contains. If condition true then only it allow to insert the records example. Example salary > 15000 or age > 21 etc.
5. Default constraints: if user or programmer doesn’t insert any value then what default value consider for that record we can use with default constraint. If we need we can override that value.
6. Foreign key : Foreign key is use to connect primary key of same or different table. FK allow only those values which present in PK column. FK can allow null value as well as it allow duplicate.

**Table relationship**

4 types of relationship

1. One to many -🡪 Department and employees : in one department many employee working.

Trainer and students : one trainer can handle many students

1. One to one -🡪 Person to passport : one person has only one passport.

One user has only one userprofile

1. Many to many 🡪 many employees or student know more than one skillset or technologies.
2. Many to one

**One to many with contains on Department with employees tables.**

Department 🡪Table

Department\_id -🡪 Pk with auto increment

Department\_name 🡪 varchar(50) not null,

Location -🡪 varchar(50) not null with default as new York

create table department(department\_id int primary key auto\_increment,

department\_name varchar(100) not null,

location varchar(100) not null default 'New York');

insert into department(department\_name) values('IT');

insert into department(department\_name) values('Marketing');

insert into department(department\_name,location) values('Sales','Texas');

insert into department(department\_name,location) values('HR','California');

Employees -🡪 table

Employee\_id 🡪int with primary key

First\_name -🡪 varchar(30) not null

Last\_name -🡪 varchar(30)

Gender -🡪 enum(‘Male’,’Female’)

Age --🡪 int with check constraints > 21

Salary -🡪 float with check constraints 🡪 25000 to 50000

Emailid 🡪varchar(30) unique not null

Doj -🡪date

Department\_id 🡪 int this column as FK link to Pk of department table with department\_id

**Employees table creation**

create table employees(employee\_id int primary key,

first\_name varchar(30) not null,

last\_name varchar(30),

gender enum('male','female'),

age int check (age>21),

salary float check (salary between 25000 and 50000),

email varchar(30) unique not null,

doj date,

department\_id int,

foreign key(department\_id) references department(department\_id));

insert into employees values(100,'John','Deo','Male',24,27000,'john.deo@gmail.com','2018-10-05',1);

insert into employees values(101,'Jane','Smith','Male',27,32000,'jane.smith@gmail.com','2019-12-05',2);

insert into employees values(102,'Emaily','Davis','Female',23,32000,'emaily.davis@gmail.com','2020-04-15',2);

insert into employees values(103,'Charlotte','King','Male',26,45000,'charlotte.king@gmail.com','2021-10-20',3);

insert into employees values(104,'Mia','Adams','female',28,50000,'mai.adams@gmail.com','2021-10-20',3);

insert into employees values(105,'Steven',null,'male',34,48000,'steven@gmail.com','2022-12-22',1);

insert into employees values(106,'James','Miller','male',36,45000,'james.miller@gmail.com','2018-10-25',null);

One to One relationship

1. Shared primary key
2. Pk with FK

Users -🡪 table primary table

Uid -🡪 int primary key with auto\_increment

Email 🡪 varchar(30) unique not null

Password 🡪 varchar(30) not null

Account\_created 🡪 timestamp default current\_timestamp

create table users(uid int primary key auto\_increment,

email varchar(30) not null,

password varchar(30) not null,

account\_create timestamp default current\_timestamp);

insert into users(email,password) values('john.deo@gmail.com','john@123');

insert into users(email,password) values('jane.smith@gmail.com','jane@123');

Users\_Profile 🡪 table secondary table

Profile\_id 🡪 int primary key,

First\_name🡪varchar(30) not null

Last\_name varchar(30)

Age 🡪 int

Dob 🡪date,

Foreign key(Profile\_id) references Users(uid)

create table user\_profile(profile\_id int primary key,

first\_name varchar(30) not null,

last\_name varchar(30),

age int,

dob date,

foreign key(profile\_id) references users(uid));

insert into user\_profile values(1,'john','deo',34,'2022-10-01');

insert into user\_profile values(2,'jane','smith',36,'2008-01-12');

Many to many relationship

**Student**

Sid (PK) SName

100 John

101 Jane

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

insert into student values(100,'john');

insert into student values(101,'jane');

**Course**

Cid(PK) CName

1000 Java

1001 Python

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

insert into course values(1000,'Java');

insert into course values(1001,'Python');

**StudentCourseRelationship**

SCID int auto\_increment, SID(FK), Cid(FK)

1 100 1000

2 100 1001

3 101 1001

create table student\_course\_relationship(scrid int primary key auto\_increment, sid int, cid int, foreign key(sid) references student(sid), foreign key(cid) references course(cid));

insert into student\_course\_relationship(sid,cid) values(100,1000);

insert into student\_course\_relationship(sid,cid) values(100,1001);

insert into student\_course\_relationship(sid,cid) values(101,1001);

Department with Employee many to many relationship

Department

Did(PK) others columns

1

2

Employees

EmpId(PK) others columns

100

101

EmployeeDepartmentRelationship

EDRID Did(FK) EmpId(FK)

1111 1 100

2222 2 100

3333 1 101

4444 1 100