Database:

A database is an organized collection of data that is stored and accessed electronically. The data is structured in a way that allows easy retrieval, management, and updating. Databases are essential for handling large amounts of information in a structured and efficient manner.

Database management system:

A Database Management System (DBMS) is software that allows user to define, create, maintain, and control access to database. It acts as an intermediary between the user and database, ensuring that the data is stored electrically, retrieved quickly, and managed securely.

Types of Databases:

1. Centralized Database
2. Distributed Database
3. NoSQL Database
4. Cloud Database
5. Relational Database
6. Network Database
7. Object-Oriented Database
8. Hierarchical Database

Relational Database: Data is Stored in Tables form. Like NoSQL.

Non - Relational (NoSQL): Data is not stored in Tables form, like stored in documents form, like MongoDB.

SQL🡪Structured Query Language.

CRUD🡪Create, Read, Update, Delete.

Structure of Table: combination of rows and columns.

SQL Commands:

🡪DDL (Data Definition Language) 🡪Create, Drop, Alter, Truncate, Rename.

🡪DQL (Data Query Language) 🡪 select.

🡪DML (Data Manipulation Language) 🡪Insert, Update, Delete, Lock.

🡪DCL (Data Control Language) 🡪Grant, Revoke. (Giving permissions to users)

🡪TCL (Transaction Control Language) 🡪Begin, Transaction, Commit, roll back, Save, Print, etc.,



SQL Data Types:

Creating a Database:

Query 🡪 Create database db\_name;

Deleting Database:

Query 🡪Drop database db\_name;

Use Database:

Query 🡪use db\_name;

🡪 CREATE DATABASE movies;🡪

|  | **#** | **Time** | **Action** | **Message** | **Duration / Fetch** |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 14:10:45 | CREATE DATABASE movies | 1 row(s) affected | 0.016 sec |

🡪 SHOW databases;🡪

|  |
| --- |
| ab123 |
| information\_schema |
| movies |
| mydb |
| mysql |
| performance\_schema |
| sys |
| vishnu |
| vishnu10 |

🡪 DROP database vishnu;🡪

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 5 | 14:15:19 | DROP database vishnu | 0 row(s) affected | 0.031 sec |

Creating Table:CREATE TABLE table\_name(column1 datatype1,column2 datatype2,---);

🡪CREATE TABLE movies (

movie\_id INT, tittle VARCHAR(225),release\_year YEAR,genre varchar (100),language varchar(50),duration\_minutes INT,rating decimal (3,1),director\_id INT);

🡪show tables;

Primary Key: It is a unique identifier for each record in a database table.

Foreign Key: It is a field in a one table, that refers to the primary key in another table.

NOTE:

🡪unique primary key and NOT null.

🡪foreign keys can be multiple and null.

use movies;

create table Directors (

director\_id INT PRIMARY KEY auto\_increment,

name varchar (225) not null,

dob date,

nationality varchar (100),

awards text);

create table Movies123 (

movie\_id int primary key auto\_increment,

title varchar (225) not null,

release\_year year not null,

genre varchar (100) not null,

language varchar (50) default "Telugu",

duration\_minutes int not null,

rating decimal (3,1),

director\_id int,

foreign key (director\_id) references Directors(director\_id)

);

create table Actors (

actor\_id int primary key auto\_increment,

name varchar (225) not null,

dob date,

genre char(1),

nationality varchar(100),

debut\_year year

);

create table Movie\_Cast (

movie\_id int,

actor\_id int,

role\_name varchar(255),

screen\_time\_minutes int,

foreign key (movie\_id) references Movies123(movie\_id),

foreign key (actor\_id) references Actors(actor\_id),

primary key(movie\_id,actor\_id));

create table Box\_Office (

movie\_id int , budget bigint,

box\_office\_collection bigint,

domestic\_collection bigint,

international\_collection bigint,

foreign key (movie\_id) references Movies123(movie\_id),

primary key(movie\_id));

show tables;

insert into Directors(name,dob,nationality,awards) values

('vishnu','2003-08-11','indian','no awards');

insert into Directors(name,dob,nationality,awards) values

('vishnu123','2003-08-11','indian','no awards');

select \* from Directors;

insert into Directors (name,dob,nationality,awards) values

('S.S Rajamouli','1973-10-10','indian','Padma Shri, Nandi Award'),

('Trivikram Srinivas','1971-11-07','indian','Nandi Award, SIIMA Award'),

('Puri Jagannath','1966-09-26','indian','Nandi Award, Filmfare Award'),

('Sukumar','1970-01-11','indian','Filmfare Award, Nandi Award'),

('V.V Vinayak','1974-10-09','indian','Nandi Award'),

('Koratala Siva','1975-06-15','indian',null),

('Boyapati Srinu','1971-04-01','indian','Nandi Award'),

('Harish Shankar','1982-03-31','indian',null),

('Srinu Vitla','1972-09-24','indian',null),

('Gautham Vasudev Menon','1973-02-25','indian','Nandi Award');

insert into Movies123 (title,release\_year,genre,duration\_minutes,rating,

director\_id) values

('Baahubali:The Beginning',2015,'Action',159,8.1,1),

('Ala Vaikuntapurramuloo',2020,'Drama',165,7.8,2),

('Pokiri',2006,'Action',155,7.9,3),

('Rangasthalam',2018,'Drama',179,8.4,4),

('Tagore',2003,'Action',180,7.5,5),

('Bharath Ane Nenu',2018,'Political',173,7.7,6),

('Sarrinodu',2016,'Action',160,6.8,7),

('Gabbar Singh',2012,'Action',152,7.2,8),

('Dookudu',2011,'Comedy',175,7.5,9),

('Yennai Arindhaal',2015,'Action',176,7.3,10);

insert into Actors (name,dob,nationality,debut\_year) values

('Prabhas','1979-10-23','Indian',2002),

('Allu Arjun','1983-04-08','Indian',2003),

('Mahesh Babu','1975-08-09','Indian',1999),

('Ram Charan','1985-03-27','Indian',2007),

('Chiranjeevi','1955-08-22','Indian',1978),

('Jr.NTR','1983-05-20','Indian',1996),

('Pawan Kalyan','1971-09-02','Indian',1996),

('Nagarjuna','1959-02-24','Indian',2008),

('Nani','1984-02-24','Indian',2008),

('Samantha Ruthu Prabhu','1987-04-28','Indian',2010);

insert into Movie\_Cast (movie\_id,actor\_id,role\_name,screen\_time\_minutes)

values (1,1,'Shivudu',120),

(2,2,'Bantu',140),

(3,3,'Pandu',130),

(4,4,'Chitti Babu',150),

(5,5,'Tagore',160),

(6,6,'Bharath',145),

(7,2,'Gana',135),

(8,7,'Gabbar singh',175),

(9,3,'Ajay',150),

(10,9,'Satyadev',160);

insert into Box\_Office (movie\_id,budget,box\_office\_collection,

domestic\_collection,international\_collection) values

(1,1800000000,6000000000,450000000,1500000000),

(2,10000000000,2700000000,21000000000,600000000),

(3,120000000000,660000000000,5000000000,160000000),

(4,15000000000,3650000000000,380000000000,85000000000),

(5,20000000000,30000000000,2500000000,50000000000),

(6,120000000,45000000000,3400000000000,78000000000),

(7,56000000000,860000000000,400000000,4500000000),

(8,3400000000000,5600000000000,45000000,340000000),

(9,35000000000,1950000000,56000000000,340000000000),

(10,500000000,85000000000,70000000000,1500000000000);

select \* from Box\_Office;

Insertion Syantax:

🡪Inserting Single Value:

Insert into table\_name (col1,col2,,,,,,) values (val1,val2,,,,,);

🡪Inserting Multiple Values:

Insert into table\_name (col1,col2,,,,,) values (val1,val2,val3,,,,),(val4,val5,val6,,,,);

How to see tables?

🡪Select: SELET statement in SQL is used to retrieve data from one or more tables in a database.

Basic Syntax:

🡪SELECT col1,col2,----,FROM table\_name;

select budget,box\_office\_collection from Box\_Office;

Selecting all columns:

🡪SELECT \* FROM table\_name;

🡪Selecting Specific columns🡪SELECT title,release\_year,rating FROM Movies123;

🡪Selecting all columns🡪SELECT \* FROM Movies123;

Practice Time:

QS🡪Create a table named books with the following columns book\_id, title, author, genre, price, published\_year, in\_stock.Insert few random values, display tittle, published\_year columns.

Ans 🡪 create table books (

book\_id int,

title varchar(100),

author varchar(100),

genre varchar(100),

price int,

published\_year year,

in\_stock boolean);

insert into books values

(20,'abc','vishnu','science fiction',2000,2024,true);

select title,published\_year from books;

create table books (

book\_id int,

title varchar(100),

author varchar(100),

genre varchar(100),

price int,

published\_year year,

in\_stock boolean);

insert into books values

(20,'abc','vishnu','science fiction',2000,2024,true);

insert into books values

(20,'abc','vishnu123','science fiction',2000,2024,false);

select title,published\_year,in\_stock from books;

select \* from books;

Constraints:Specifiy rules for data in a table.

\*NOT NULL🡪Ensures that a column cannot contain NULL values.

For example name varchar(100) not null;

create table Employees (

employee\_id int primary key,

name varchar(50) not null,

salary decimal(10,2));

insert into Employees values(

1,'vishnu',10000);

insert into Employees values(

2,'',10000);

insert into Employees (employee\_id,salary) values

(1,10000);

O/P🡪field ‘name’ doesn’t have a default value.

insert into Employees (name,salary) values

('vishnu',10000);

O/P🡪field ‘employee\_id’ doesn’t have a default value.

UNIQUE:Ensures that all the values in a column (or a combination of columns) are unique across the rows of the table.

create table Users (

user\_id int primary key,

user\_name varchar(100) unique);

insert into Users values(1,'vishnu');

insert into Users values(2,'vishnu');

O/P🡪Duplicate entry ‘vishnu’,for key user\_name.

DEFAULT: Provides a default value for a column when no value is specified during insertion.

create table orders (

order\_id int primary key,

order\_date date default'2024-10-20');

insert into orders values (100,'2023-04-05');

select \* from orders;

insert into orders (order\_id) values(200);

select \* from orders;🡪Here default data is added.

CHECK:Ensures that all values in a column satisfy a specific condition. It can be used to limit the range of values that can be inserted into a column.

create table products (

prod\_id int primary key,

price int,

quantity int check(quantity>0)

);

insert into products (prod\_id,price,quantity) values(1,100,10);

insert into products (prod\_id,price,quantity) values(1,100,-1);🡪Check constrain is violated.

WHERE:IT is used to filter records in a query based on specific condition.

Syntax: SELECT col1,col2 FROM table\_name WHERE condition;

Basic Example🡪SELECT tittle,release\_year FROM Movies123 WHERE release\_year=2022;

SELECT title,release\_year FROM Movies123 WHERE release\_year=2011;

SELECT title,release\_year FROM Movies123 WHERE release\_year>2011;

Using operators with WHERW:

Arithmetic operators🡪+,-,\*,/,%.

Comparison operators🡪=,!=,>,>=,<,<=.

Logical operators🡪AND, OR,|NOT,IN,BETWEEN,ALL,LIKE,ANY.

Bitwise operators🡪&,|.

Arithmetic operators🡪profit of a movie = boxoffice collection-budget

i.e SELECT movie\_id,box\_office\_collection-budget as profit\_or\_loss from Box\_Office;

Doubling a movie budget🡪doubled budget=budget\*2

i.e SELECT movie\_id,budget\*2 as doubled\_budget FROM Box\_Office;

Order By:SELECT col1,col2,---- FROM table\_name ORDER BY col1 Asc/Desc, col2 Asc/Desc;

Example:Select title,release\_year FROM Movies123 ORDER BY releasing\_year DESC;

Select title,release\_year FROM Movies123 ORDER BY release\_year ASC;

LIMIT: SELECT col1,col2,--- FROM table\_name LIMIT number\_of\_rows;

EXAMPLE: SELECT title,release\_year from Movies123 LIMIT 5;

AGGREGATE FUNCTIONS:

1. Count🡪Select Count (\*) From Movies123;🡪return no.of rows.
2. Max🡪Select Max(budget) From Box\_Office;
3. Min🡪Select Min(rating) From Movies123;
4. Sum🡪Select Sum(box\_office\_collections) From Box\_Office;
5. Avg🡪Select Avg(rating) From Movies123;

GROUP BY:SELECT col1,col2 aggregate\_function(col2) FROM table\_name;

EXAMPLE:SELECT language,sum(box\_office\_collection) AS total\_colection FROM Movies123 m JOIN Box\_Office b on m.movie\_id=b.movie\_id GROUP BY language;

HAVING:SELETC col1,aggregate\_function(col2) FROM table\_name GROUP BY col1 HAVING condition;

EXAMPLE:SELECT language,Sum(total\_office\_collection) as total\_collection from Movies123 m JOIN Box\_Office b on m.movie\_id=b.movie\_id GROP BY language HAVING sum(box\_office\_collection)>5000000;

Order of mentioning SQL Clauses:

SELECT🡪FROM🡪JOIN🡪WHERE🡪GROUP BY🡪HAVING🡪ORDER BY🡪LIMIT.

UPDATE:Modify existing records.

Update table\_name set col1=value1,col2=value2 ,--- where condition;

Example🡪Update Movies123 set rating=9.0,where title=’rrr’;

DELETE:Remove existing records.

Delete from table\_name where condition;

Example🡪Delete from Movies123 where tittle=’kalki 2898 AD’;

Csacading for Foreign Keys:

🡪ON DELETE CASCADE:🡪Automatically deletes all rows in the child table that reference the deleted row in the parent table.

ALTER TABLE:Alter table table\_name ADD column\_name datatype constraints;🡪Adding new columns,Modify existing column,Drop columns.

DROP TABLE:DROP table table\_name;🡪removing table from database.

RENAME TABLE:RENAME table old\_table\_name to new\_table\_name;🡪renaming the table.

TRUNCATE TABLE:TRUNCATE table table\_name ;🡪remove all records in table without deleting actual table.

INNER JOIN🡪returns common rows in both tables.

🡪Select columns from table1 inner join table2 on table1.column=table2.column;

LEFT JOIN:returns all rows from the left table and the matched rows from the right table.if there is no match,NULL values are returned for columns from the right table.

🡪Select columns from table 1 left join table2 on table1.column=table2.column;

RIGHT JOIN:Similar to left join, but it retuns all rows from the right table and the matched rows from the left table.if there is no match ,NULL values are returned for columns from the left table.

🡪Select columns from table 1 right join table2 on table1.column=table2.column;

CROSS JOIN:return the cartesian product of the two tables,meaning that it returns all possible combinations of rows .this type of join is rarely used because it can produce a very large number of rows.

🡪Select columns from table1 cross join table2;

SELF JOIN:It is ajoin of a table with itself.this is useful for comparing rows within the same table.

🡪Select a.column,b.column from table a table b where condition;

UNION:It is used to combine the result-set of two or more select statements.

🡪Select columns from table1 union select columns from table2;

Sub Queries:

use movies;

select \* from Box\_Office;

select budget from Box\_Office where budget>(select avg(budget) from Box\_Office);

PRACTISING WORK:

use movies;

create table Directors (

director\_id INT PRIMARY KEY auto\_increment,

name varchar (225) not null,

dob date,

nationality varchar (100),

awards text);

create table Movies123 (

movie\_id int primary key auto\_increment,

title varchar (225) not null,

release\_year year not null,

genre varchar (100) not null,

language varchar (50) default "Telugu",

duration\_minutes int not null,

rating decimal (3,1),

director\_id int,

foreign key (director\_id) references Directors(director\_id)

);

create table Actors (

actor\_id int primary key auto\_increment,

name varchar (225) not null,

dob date,

genre char(1),

nationality varchar(100),

debut\_year year

);

create table Movie\_Cast (

movie\_id int,

actor\_id int,

role\_name varchar(255),

screen\_time\_minutes int,

foreign key (movie\_id) references Movies123(movie\_id),

foreign key (actor\_id) references Actors(actor\_id),

primary key(movie\_id,actor\_id));

create table Box\_Office (

movie\_id int , budget bigint,

box\_office\_collection bigint,

domestic\_collection bigint,

international\_collection bigint,

foreign key (movie\_id) references Movies123(movie\_id),

primary key(movie\_id));

show tables;

insert into Directors(name,dob,nationality,awards) values

('vishnu','2003-08-11','indian','no awards');

insert into Directors(name,dob,nationality,awards) values

('vishnu123','2003-08-11','indian','no awards');

select \* from Directors;

insert into Directors (name,dob,nationality,awards) values

('S.S Rajamouli','1973-10-10','indian','Padma Shri, Nandi Award'),

('Trivikram Srinivas','1971-11-07','indian','Nandi Award, SIIMA Award'),

('Puri Jagannath','1966-09-26','indian','Nandi Award, Filmfare Award'),

('Sukumar','1970-01-11','indian','Filmfare Award, Nandi Award'),

('V.V Vinayak','1974-10-09','indian','Nandi Award'),

('Koratala Siva','1975-06-15','indian',null),

('Boyapati Srinu','1971-04-01','indian','Nandi Award'),

('Harish Shankar','1982-03-31','indian',null),

('Srinu Vitla','1972-09-24','indian',null),

('Gautham Vasudev Menon','1973-02-25','indian','Nandi Award');

insert into Movies123 (title,release\_year,genre,duration\_minutes,rating,

director\_id) values

('Baahubali:The Beginning',2015,'Action',159,8.1,1),

('Ala Vaikuntapurramuloo',2020,'Drama',165,7.8,2),

('Pokiri',2006,'Action',155,7.9,3),

('Rangasthalam',2018,'Drama',179,8.4,4),

('Tagore',2003,'Action',180,7.5,5),

('Bharath Ane Nenu',2018,'Political',173,7.7,6),

('Sarrinodu',2016,'Action',160,6.8,7),

('Gabbar Singh',2012,'Action',152,7.2,8),

('Dookudu',2011,'Comedy',175,7.5,9),

('Yennai Arindhaal',2015,'Action',176,7.3,10);

insert into Actors (name,dob,nationality,debut\_year) values

('Prabhas','1979-10-23','Indian',2002),

('Allu Arjun','1983-04-08','Indian',2003),

('Mahesh Babu','1975-08-09','Indian',1999),

('Ram Charan','1985-03-27','Indian',2007),

('Chiranjeevi','1955-08-22','Indian',1978),

('Jr.NTR','1983-05-20','Indian',1996),

('Pawan Kalyan','1971-09-02','Indian',1996),

('Nagarjuna','1959-02-24','Indian',2008),

('Nani','1984-02-24','Indian',2008),

('Samantha Ruthu Prabhu','1987-04-28','Indian',2010);

insert into Movie\_Cast (movie\_id,actor\_id,role\_name,screen\_time\_minutes)

values (1,1,'Shivudu',120),

(2,2,'Bantu',140),

(3,3,'Pandu',130),

(4,4,'Chitti Babu',150),

(5,5,'Tagore',160),

(6,6,'Bharath',145),

(7,2,'Gana',135),

(8,7,'Gabbar singh',175),

(9,3,'Ajay',150),

(10,9,'Satyadev',160);

insert into Box\_Office (movie\_id,budget,box\_office\_collection,

domestic\_collection,international\_collection) values

(1,1800000000,6000000000,450000000,1500000000),

(2,10000000000,2700000000,21000000000,600000000),

(3,120000000000,660000000000,5000000000,160000000),

(4,15000000000,3650000000000,380000000000,85000000000),

(5,20000000000,30000000000,2500000000,50000000000),

(6,120000000,45000000000,3400000000000,78000000000),

(7,56000000000,860000000000,400000000,4500000000),

(8,3400000000000,5600000000000,45000000,340000000),

(9,35000000000,1950000000,56000000000,340000000000),

(10,500000000,85000000000,70000000000,1500000000000);

select \* from Box\_Office;

select budget,box\_office\_collection from Box\_Office;

create table books (

book\_id int,

title varchar(100),

author varchar(100),

genre varchar(100),

price int,

published\_year year,

in\_stock boolean);

insert into books values

(20,'abc','vishnu','science fiction',2000,2024,true);

insert into books values

(20,'abc','vishnu123','science fiction',2000,2024,false);

select title,published\_year,in\_stock from books;

select \* from books;

create table Employees (

employee\_id int primary key,

name varchar(50) not null,

salary decimal(10,2));

insert into Employees values(

1,'vishnu',10000);

insert into Employees values(

2,'',10000);

insert into Employees (employee\_id,salary) values

(1,10000);

insert into Employees (name,salary) values

('vishnu',10000);

create table Users (

user\_id int primary key,

user\_name varchar(100) unique);

insert into Users values(1,'vishnu');

insert into Users values(2,'vishnu');

insert into Users values(2,'vishnu1');

create table orders (

order\_id int primary key,

order\_date date default'2024-10-20');

insert into orders values (100,'2023-04-05');

select \* from orders;

insert into orders (order\_id) values(200);

select \* from orders;

create table products (

prod\_id int primary key,

price int,

quantity int check(quantity>0)

);

insert into products (prod\_id,price,quantity) values(1,100,10);

insert into products (prod\_id,price,quantity) values(1,100,-1);

select \* from products;

SELECT title,release\_year FROM Movies123 WHERE release\_year=2011;

SELECT title,release\_year FROM Movies123 WHERE release\_year>2011;

/\*Arithmetic\*/

SELECT movie\_id,box\_office\_collection-budget as profit\_or\_loss from Box\_Office;

SELECT movie\_id,budget,budget-budget,budget\*2 as doubled\_budget FROM Box\_Office;

select \* from Actors;

select \* from Actors where debut\_year!=2001;

select title,release\_year from Movies123 where language='Telugu' and release\_year=2012;

select title,release\_year from Movies123 where language='Telugu' or release\_year=2012;

insert into Movies123(title,release\_year,genre,language,duration\_minutes,rating) values('Dabang',2012,'Action','Hindi',140,9.0);

select title,release\_year from Movies123 where language='Telugu' or release\_year=2012;

select \* from Movies123 where not language='Telugu';

insert into Movies123(title,release\_year,genre,language,duration\_minutes,rating) values('Premam',2012,'Romance','Malayalam',140,9.0);

select \* from Movies123 where language in ('Malayalam');

select \* from Movies123 where language in ('Malayalam','Hindi');

select \* from Movies123 where language ='Malayalam';

select title,release\_year from Movies123 where release\_year between 2015 and 2022;

select \* from Movies123;

select \* from Movies123 where title like 'Baahu%';

select \* from Movies123 where title like '%ubali:The%';

select \* from Movies123 where title like '%ubali:The';

select \* from Movies123 where title like 'P\_kiri';

select \* from Movies123 where title like 'P\_k\_i';

select \* from Movies123 where title like 'P\_\_kiri';

select \* from Movies123;

Select title,release\_year FROM Movies123 ORDER BY release\_year DESC;

Select title,release\_year FROM Movies123 ORDER BY release\_year ASC;

SELECT title,release\_year from Movies123 limit 2;

SELECT title,release\_year from Movies123 limit 11;

SELECT title,release\_year from Movies123 limit 10;

/\*Aggregate Function\*/

select count(\*) from Movies123;

select count(\*) as total\_count from Movies123;

select max(budget) from Box\_Office;

select max(budget) as max\_budget from Box\_Office;

select min(budget) from Box\_Office;

select min(budget) as min\_budget from Box\_Office;

select sum(budget) from Box\_Office;

select sum(box\_office\_collection) as tottal\_sum from Box\_Office;

select avg(rating) from Movies123;

select language,sum(box\_office\_collection) as total\_collection

from Movies123 m join Box\_Office b on m.movie\_id=b.movie\_id

group by language;

select \* from Movies123;

insert into Box\_Office (movie\_id,budget,box\_office\_collection,

domestic\_collection,international\_collection) values

(11,18000000,60000000,4500000,15000000),

(12,1800000,6000000,45000000,150000);

select language,sum(box\_office\_collection) as total\_collection from

Movies123 m join Box\_Office b on m.movie\_id=b.movie\_id group by language;

/\*works like whwere function\*/

select language,sum(box\_office\_collection) as total\_collection from

Movies123 m JOIN Box\_Office b on m.movie\_id=b.movie\_id group by language

HAVING sum(box\_office\_collection)>50000000000;

select \* from Movies123;

update Movies123 set rating=3.5 where title='pokiri';/\*safe mode is on now\*/

SET SQL\_SAFE\_UPDATES=0;/\*SAFE MOPDE OFF NOW\*/

SET SQL\_SAFE\_UPDATES=1;

update Movies123 set rating=3.5 where title='pokiri';

SET SQL\_SAFE\_UPDATES=0;

delete from Movies123 where title='Pokiri';/\* Its a foreign key we cannot delete\*/

create table dummy1 (

name varchar(100));

insert into dummy1 values('vishnu');

select \* from dummy1;

delete from dummy1 where name='vishnu';

select \* from dummy1;

create database devara1;

use devara1;

create table Movies1 (

movie\_id int primary key auto\_increment,

title varchar (225),

release\_year year,

genre varchar (100),

language varchar (50) default "Telugu",

duration\_minutes int,

rating decimal (3,1)

);

create table Box\_Office12 (

box\_office\_id int primary key auto\_increment,

movie\_id int , budget bigint,

box\_office\_collection bigint,

domestic\_collection bigint,

foreign key (movie\_id) references Movies1(movie\_id)

on delete cascade

);

/\*Inserting data\*/

insert into Movies1(title,release\_year,genre,language,duration\_minutes,rating)

values ('movie1',2020,'Action','telugu',150,7.8),

('movie2',2022,'Action','telugtu',155,7.9),

('movie3',2023,'Action','telugu',150,7.9),

('movie4',2021,'Action','telugu',159,7.0),

('movie5',2023,'Action','telugu',1180,9.9);

insert into Box\_Office12 (movie\_id,budget) values

(1,2500000000),

(2,544000000),

(3,6000000000),

(4,560000000),

(5,6700000000000);

delete from Movies1 where title='movie2';

create database pushpa1;

use pushpa1;

create table Movies1 (

movie\_id int primary key auto\_increment,

title varchar (225),

release\_year year,

genre varchar (100),

language varchar (50) default "Telugu",

duration\_minutes int,

rating decimal (3,1)

);

create table Box\_Office12 (

box\_office\_id int primary key auto\_increment,

movie\_id int , budget bigint,

box\_office\_collection bigint,

domestic\_collection bigint,

foreign key (movie\_id) references Movies1(movie\_id)

on update cascade

);

insert into Movies1(title,release\_year,genre,language,duration\_minutes,rating)

values ('movie1',2020,'Action','telugu',150,7.8),

('movie2',2022,'Action','telugtu',155,7.9),

('movie3',2023,'Action','telugu',150,7.9),

('movie4',2021,'Action','telugu',159,7.0),

('movie5',2023,'Action','telugu',1180,9.9);

insert into Box\_Office12 (movie\_id,budget) values

(1,2500000000),

(2,544000000),

(3,6000000000),

(4,560000000),

(5,6700000000000);

show tables;

SET SQL\_SAFE\_UPDATES=0;

update Movies1 set movie\_id=143 where title='movie1';

select \* from Movies1;

/\*Other Cascading Options

1.SET NULL

2.SET DEFAULT

3.NO ACTION

4.RESTRICT\*/

select \* from Movies1;

Alter table Movies1 add director\_id int;

select \* from Movies1;

alter table Movies1 modify rating decimal(4,2);

select \* from Movies1;

alter table Movies1 drop column director\_id;

select \* from Movies1;

show tables;

DROP table box\_office12;

rename table Movies1 to Films;

show tables;

select \* from Films;

truncate table Films;

select \* from Films;

/\*INNER JOIN\*/

drop table films;

create table movies (

movie\_id int, title varchar(20),director\_id int);

create table directors (

director\_id int,name varchar(20));

insert into movies values

(1,'RRR',101),

(2,'Pushpa',102),

(3,'KGF',103);

insert into directors values

(101,'s.s rajamouli'),

(102,'sukumar'),

(104,'prashanth neel');

select \* from movies;

select \* from directors;

select \* from movies m inner join directors d

on m.director\_id=d.director\_id;

select m.title,d.name from movies m inner join directors d

on m.director\_id=d.director\_id;

/\*LEFT JOIN\*/

select m.title,d.name from movies m left join directors d

on m.director\_id=d.director\_id;

select \* from movies m left join directors d

on m.director\_id=d.director\_id;

/\*RIGHT JOIN\*/

select m.title,d.name from movies m right join directors d

on m.director\_id=d.director\_id;

/\*CROSS JOIN\*/

select \* from movies m cross join directors d;

/\*SELF JOIN\*/

select a.title as movie1,b.title as movie2,a.release\_year from

movies a ,movies b where a.release\_yaer=b.release\_year and a.movie\_id;

Use movies;

/\*UNION\*/

create database union\_test;

use union\_test;

create table telugu\_movies (

movie\_id int,

title varchar(20),

release\_year year

);

insert into telugu\_movies values

(1,'RRR',2022),

(2,'Pushpa',2021),

(3,'KGF',2018);

create table hindi\_movies (

movie\_id int,

title varchar(20),

release\_year year

);

insert into hindi\_movies values

(1,'Dangal',2016),

(2,'Bajarangi Bhaijaan',2015),

(3,'KGF',2018);

select title,release\_year from telugu\_movies

union

select title,release\_year from hindi\_movies;

select title,release\_year from telugu\_movies

union all /\*-->Allows Duplicates\*/

select title,release\_year from hindi\_movies;

use movies;

select \* from Box\_Office;

select budget from Box\_Office where budget>(select avg(budget) from Box\_Office);