CREATE DATABASE ROOM;

-- //OR

CREATE DATABASE IF NOT EXISTS ROOM;

CREATE DATABASE ROOM1;

DROP DATABASE ROOM1;

-- //OR

DROP DATABASE IF EXISTS ROOM1;

USE ROOM;

SHOW DATABASES;

-- /\*

-- DATATYPE --------------------DESCRIPTION -----------------------------------USAGE

-- CHAR - string(0-255), can store characters of fixed length - CHAR(50)

-- VARCHAR - string (0-255 , can store characters up to given length - VARCHAR(50)

-- BLOB - string (0-65535) - can store binary large object - BLOB(1000)

-- INT - integer (-2,147,483,648 to 2,147,483,647) -INT

-- TINYINT - integer (-128 to 127) -TINYINT

-- BIGINT - integer (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807) -BIGINT

-- BIT - can store x-bit values. x can range from 1 to 64 -BIT(2)

-- FLOAT 0 decimal number - with precision to 23 didgits -FLOAT

-- DOUBLE - Decimal number - with 24 to 53 digits - DOUBLE

-- BOOLEAN - boolean values 0 or 1 - BOOLEAN

-- DATE - date in format of YYYY-MM-DD ranging from - DATE

-- YEAR - year in 4 digits format ranging from 1901-2155 -YEAR

-- TYPES OF SQL COMMANDS

-- DDL (Data Definition Language)- create, alter,rename,truncate & drop

-- DQL (Date query Language) - select

-- DML (Data manipulation Language)- insert,update,delete

-- DCL (Data Control Language) -grant & revoke permission to users

-- TCL (TRANSACTION CONTROL LANGUAGE) - start transaction , commit, rollback

-- \*/

CREATE TABLE student (

id INT PRIMARY KEY,

name VARCHAR (50),

age INT NOT NULL

)

INSERT INTO student VALUES(1,"sai",20);

INSERT INTO student VALUES(2,"ASHU",22);

-- //OR

-- INSERT INTO student (rollno,name) VALUES (105,'tirtha');

SELECT \* FROM student;

SHOW TABLES;

-- Pratice Qs- 1

-- qs- Create a database for your comapny named XYZ.

-- Step 1: create a table inside this DB to store employee info (id,name,salary).

-- step-2 : Add folllwowing information in the Db:

-- 1, "adam",25555

-- 2,"bob",30000,

-- 3,"casey",40000

-- step-3 : Select & view all your table data.

CREATE DATABASE XYZ\_COMPANY;

USE XYZ\_COMPANY;

CREATE TABLE employee(

id INT PRIMARY KEY,

name VARCHAR (50),

salary INT

);

INSERT INTO employee(id,name,salary) VALUES (1,'ADAM',25555),

(2,'BOB',30555),

(3,'CASEY',40555);

SELECT \* FROM employee;

-- Primary Key:- It is a column (or set of columns) in a table that uniquely identifies

-- each row. (a unique id)

-- There is only 1PK & it should be NOT null.

-- Foreign Key:-

-- A foreign Key is a column for (or set of columns) in a table that refers to the primary

-- key . There can be multiple FKs.

-- FKs can have duplicate & null values.

-- Constraints:-

-- SQL constraints are used to specify rules for data in a table.

-- NOT NULL- columns cannot have a null value - (col int NOT NULL)

-- UNIQUE - all values in columns are different - (col int UNIQUE)

-- PRIMARY KEY - makes a column unique & not null but used onl for one. - (id int PRIMARY KEY)

-- FOREIGN KEY - Prevent actions that would destroy links between tables

-- DEFAULT :- sets the default value of a column

CREATE TABLE temp(

id int not null,

PRIMARY KEY (id)

);

CREATE TABLE temp(

cust\_id int,

foreign key (cust\_id) references customer(id)

);

salary INT DEFAULT 25000;

CREATE TABLE emp(

id INT,

salary INT DEFAULT 25000);

INSERT INTO emp (id) VALUES (101);

SELECT \* FROM emp;

-- Constraints:-

-- CHECK- it can limit the values allowed in a column.

CREATE TABLE city(

id INT PRIMARY KEY,

city VARCHAR(50),

age INT,

constraint age\_check check (age >= 18 AND city="Delhi")

);

CREATE database college;

USE college;

CREATE TABLE student1(

rollNo INT PRIMARY KEY,

name VARCHAR(50),

marks INT NOT NULL,

grade VARCHAR(1),

city VARCHAR(20)

);

INSERT INTO student1(rollNo,name,marks,grade,city)

VALUES

(101, 'Sai',75,'A','Delhi'),

(102, 'Suman',70,'B','USA'),

(103, 'Tirtha',90,'A','UK'),

(104, 'Ashu',75,'A','OD'),

(105, 'Bhabani',75,'A','JM'),

(106, 'Ram',75,'A','Patna'),

(107, 'krishna',84,'G','Goa');

SELECT \* FROM student;

SELECT city FROM student;

SELECT distinct city FROM student;

-- distinct - means identify the Unique values

-- Where Clause :- To define some conditions

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

-- Using Opeator in Where:-

-- 1. Arithmetic Operator:- +,-,\*,/,%

-- 2. Comparison Operator:- =,!=,>,>=,<,<=

-- 3. Logical Opeartor:- AND,OR,NOT,IN,BETWEEN,ALL,LIKE,ANY

-- BETWEEN- selects for a given range.

-- IN- matches any values in the list.

-- NOT - to negate the given condition.

-- 3. Bitwise Operator:- &, |

SELECT \* FROM student WHERE marks > 80;

SELECT \* FROM students WHERE city='Delhi';

SELECT \* FROM student WHERE marks+10 > 80;

-- Limit Clause:-

-- Sets an upper limit on number of (tuples) rows to be returned.

SELECT \* FROM student LIMIT 3;

SELECT col1,col2 FROM TABLE\_NAME LIMIT number;

-- Order By Clause:-

-- To sort in ascending (ASC) or descending (DESC)

SELECT \* FROM student ORDER BY city ASC;

SELECT col1,col2 FROM table\_name ORDER BY col\_name(s) ASC;

-- Aggregate Functions:-

-- Aggregate function perform a calculation on a set of values , and return

-- a single value.

-- 1. COUNT()

-- 2. MAX()

-- 3. MIN()

-- 4. SUM()

-- 5. AVG()

SELECT max(marks) FROM student;

SELECT avg(marks) FROM student;

Group By Clause:-

Groups rows that have the same values into summary rows.

It collects data from multiple records and groups the result by one or more column.

Generally we use group by with some aggregation function.

Count number of students in each city:-

Pratiuce Qs - 2:-

Write a Query to find avg marks in each city in ascending Order.

SELECT city, avg(marks) FROM student

GROUP BY city

ORDER BY city;

SELECT grade FROM student group by grade

order by grade;

-- Having Clause:-

-- Similar to where i.e. applies some condition on rows.

-- Used when we want to apply any condition after grouping.

-- Count number of students in each city where max marks cross 90.

SELECT count(name), city FROM student

GROUP BY city

HAVING max(marks)>90;

SELECT city, count(rollNo) FROM student GROUP BY city

having max(marks)>80;

-- General Order:-

-- SELECT column(s)

-- FROM table\_name

-- WHERE condition

-- GROUP BY column(s)

-- HAVING condition

-- ORDER BY column(s) ASC;

SELECT city

FROM student

WHERE grade="A"

GROUP BY city

HAVING max(marks) >= 90

ORDER BY city ASC;

-- Update:- to update existing rows

-- UPDATE table\_name

-- SET col1=var1,col2=var2

-- WHERE condition;

SET SQL\_SAFE\_UPDATES=0;

UPDATE student

SET grade='0'

WHERE grade='A';

SELeCT \* FROM student;

-- Delete:- To delte existing rows

-- DELETE FROM table\_name

-- WHERE condition;

DELETE FROM student

WHERE marks < 33;

-- ---------------------------- FOREIGN KEY EXAMPLE

CREATE TABLE dept(

id INT PRIMARY KEY,

name VARCHAR(50)

);

CREATE TABLE teacher(

id INT PRIMARY KEY,

name VARCHAR(50),

dept\_id INT,

FOREIGN KEY (dept\_id) references dept(id)

);

-- cascading for Foreign key

--

-- On Delete Cascade:-

-- When we create a foreign key using this option, it deletes the refencing rows in the

-- child table when the referenced row is updated in the parent table

-- which has a primary key.

-- On UPDATE Cascade:-

-- When we create a foreign key using UPDATE CASCADE the refencing rows

-- are updated in the child table when the refenced row is updated in the

-- parent table which has a primary key.

CREATE TABLE student(

id INT PRIMARY KEY,

courseID INT,

FOREIGN KEY(courseID) REFERENCES course(id)

ON DELETE CASCADE

ON UPDATE CASCADE

);

-- ALTER:- to change the schema

-- ADD Column:- ALTER TABLEtable\_name

-- ADD COLUMN column\_name datatype constraint;

-- DROP Column:-

-- ALTER TABLE table\_name DROP COLUMN column\_name;

-- RENAME TABLE:-

-- ALTER TABLE table\_name

-- RENAME TO new\_table\_name;

-- CHANGE Column(rename)

-- ALTER TABLE table\_name

-- CHANGE COLUMN old\_name new\_name new\_datatype new\_constraint;

--

-- MODIFY Column(modify datatype/constraint)

-- ALTER TABLE table\_name

-- MODIFY col\_name new\_datatype new\_constraint;

SELECT \* FROM stu;

ALTER TABLE student

ADD COLUMN age INT;

ALTER TABLE student

DROP COLUMN age;

ALTER TABLE student

ADD COLUMN age INT NOT NULL DEFAULT 19;

ALTER TABLE student

MODIFY COLUMN age VARCHAR(2);

ALTER TABLE student

CHANGE age stu\_age INT;

INSERT INTO student

(rollNo,name,marks,stu\_age)

VALUES

(157,'SUNDAR',68,150);

ALTER TABLE student

DROP COLUMN stu\_age;

ALTER TABLE student

RENAME TO stu;

-- TRUNCATE :- (to delete table's data)

-- TRUNCATE TABLE table\_name;

UPDATE stu

SET grade='O'

WHERE grade='A';

-- Pratice Qs:-

-- In the student table:-

-- 1. Change the name of column 'name' to 'fullname'.

-- 2. Delete all the students who scored marks less than 80.

-- 3. Delete the column fro grades.

ALTER TABLE stu

CHANGE name full\_name VARCHAR(50);

SELECT \* FROM stu;

DELETE FROM stu

WHERE marks < 80;

ALTER TABLE stu

DROP COLUMN grade;

-- JOINS IN SQL:-

-- Join is used to combine rows from 2 or more tables , based

-- on a related column between them.

-- 1. Inner Join:-

-- It returns records that have matchingvalues in both tables.

-- Syntax:- SELECT column(s)

-- FROM tableA

-- INNER JOIN tableB

-- ON tableA.col\_name = tableB.col\_name;

-- 2. Outer Join

-- 3. Right Join:-

-- Returns al;l records from th right table , and the matched records from

-- the left table.

-- Syntax:- SELECT column(s)

-- FROM tableA

-- RIGHT JOIN tableB

-- ON tableA.col\_name=tableB.Col\_name;

-- 4. Left join:-

-- Returns all records from the left table, and the matched records from

-- the right table.

-- Syntax:- SELECT COLUMN(S)

-- FROM tableA

-- LEFT JOIN tableB

-- ON tableA.col\_name=tableB.col\_name;

-- 5. Full Join:-

-- Returns all records when there is a match in either left or right table.

-- LEFT JOIN

-- UNION

-- RIGHT JOIN

-- Syntax:-

-- SELECT \* FROM student as a LFT JOIN course as b

-- on a.id=b.id

-- UNION

-- SELECT \* FROM student as a RIGHT JOIN course as b

-- ON a.id=b.id;

Example of Inner join:-

CREATE TABLE student(

id INT PRIMARY KEY,

name VARCHAR(50)

);

INSERT INTO student (id,name)

VALUES

(101,'SAI'),

(102,'suman'),

(103,'ashu'),

(104,'tirtha');

CREATE TABLE course(

id INT PRIMARY KEY,

course VARCHAR(50)

);

INSERT INTO course(id,course)

VALUES

(102,'MATH'),

(105,'PHY'),

(104,"ENG"),

(107,'CSE');

SELECT \* FROM student;

SELECT \* FROM course;

SELECT \* FROM student

INNER JOIN course

ON student.id=course.id;

-- RIGHT & LEFT EXCLUSIVE JOIN:-

SELECT \* FROM student as a

LEFT JOIN COURSE AS b

ON A.id=B.id

WHERE b.id IS NULL;

-- SELF JOIN:-

-- It is a regular join but the table is joined with itself.

--

-- Syntax:-

-- SELECT column(s)

-- FROM table as a

-- JOIN table as b

-- ON a.col\_name=b.col\_name;

-- example:-

SELECT a.name as manager\_name, b.name

FROM employee as a

JOIN employee as b

ON a.id=b.manager\_id;

-- UNION:-

-- It is used to combine the result-set of two or more SELECT statement.

-- Gives UNIQUE records.

-- tO USE IT:-

-- 1. every SELECT should have some no. of columns.

-- 2. columns must have similar data types.

-- 3. columns in every SELECT should be in same order.

--

-- Syntax:-

-- SELECT columns FROM tableA

-- UNION

-- SELECT columns FROM tableB

-- UNION ALL :- IT IS GIVES ALL THE DATA WITH DUPLICATE VALUES

-- Syntax:-

-- SELECT columns FROM tableA

-- UNION ALL

-- SELECT columns FROM tableB

-- SQL SUB QUERIES:-

-- A Subquery or Inner query or a Nested or a query is query with in

-- another SQL query.

-- It Involvs 2 select statements.

-- Syntax:-

-- SELECT columns

-- FROM table\_name

-- WHERE col\_name operator

-- (subquery);

-- Example:- Get name of all students who scored more than class aveage.

-- Step 1:- Find the avg of class

-- Step2 :- Find the names of students with marks > avg

SELECT \* FROM student1;

select avg(marks)

FROM student1;

select name

FROM student1

WHERE marks > (select avg(marks)

FROM student1);

-- Example:-

-- Find the names of all students with even roll numbers.

-- Step 1: Find the even roll numbers

-- Step 2:- Find the names of students with even roll number

SELECT rollNo

FROM student1

WHERE rollNo %2=0;

SELECT name,rollNo

FROM student1

WHERE rollNo IN(SELECT rollNo

FROM student1

WHERE rollNo %2=0);

-- Example:-

-- Find the max marks from the students of Delhi

-- Step 1:- Find the students of Delhi

-- Step 2:- Find their max marks using sublist in step 1

SELECT \* FROM student1

WHERE city="Delhi";

SELECT MAX(marks)

FROM (SELECT \* FROM student1 WHERE city="Delhi") AS temp;

SELECT MAX(marks)

FROM student1

WHERE city="Delhi";

-- MySQL Views:-

-- A view is a virtual table based on the result-set of an SQL statement.

CREATE VIEW view1 As

SELECT rollNo,name FROM student1;

SELECT \* FROM view1;