DBMS Practical File

By: - Aman Sharma

(03017002021)

BCA Shift-1

**PRACTICAL -1**

**Q1 Create database**

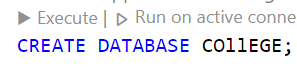
1. COLLEGE DATABASE:

STUDENT (USN, SName, Address, Phone, Gender)

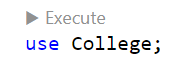
SEMSEC (SSID, Sem, Sec)

CLASS (USN, SSID) SUBJECT (Subcode, Title, Sem, Credits)

IAMARKS (USN, Subcode, SSID, Test1, Test2, Test3, FinalIA)

****

**Use Database**

****

**Create table -1 Student**

CREATE TABLE student(

    SSN int NOT NULL PRIMARY KEY,

    SName VARCHAR(50),

    Address VARCHAR(250),

    Phone BIGINT,

    Gender CHAR(10),

);

**Insert values**

INSERT into student VALUES (1,"Aman","Hauz Khas, New Delhi, Delhi",9524324356,"M");

INSERT into student VALUES (2,"Salman","khajuri Khas, New Delhi, Delhi",9524324356,"M");

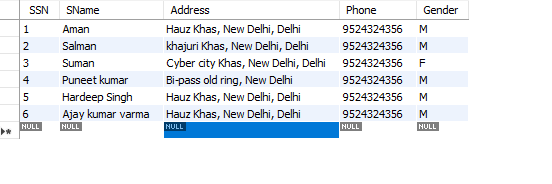
INSERT into student VALUES (3,"Suman","Cyber city Khas, New Delhi, Delhi",9524324356,"F");

INSERT into student VALUES (4,"Puneet kumar", "Bi-pass old ring, New Delhi",9524324356,"M");

INSERT into student VALUES (5,"Hardeep Singh", "Hauz Khas, New Delhi, Delhi",9524324356,"M");

INSERT into student VALUES (6,"Ajay kumar varma","Hauz Khas, New Delhi, Delhi",9524324356,"M");

SELECT \* FROM student;



**Create table-2 SEMSEC**

create table semsec(SSID int,Sem VARCHAR(10),Sec CHAR(5));

INSERT into semsec VALUES(101,"2nd","A");

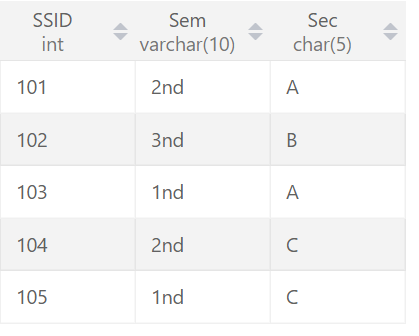
INSERT into semsec VALUES(102,"3nd","B");

INSERT into semsec VALUES(103,"1nd","A");

INSERT into semsec VALUES(104,"2nd","C");

INSERT into semsec VALUES(105,"1nd","C");

SELECT \*FROM semsec;



**Create table - 3 CLASS**

CREATE Table CLASS(USN int,SSID int);

INSERT INTO CLASS VALUES(1,101);

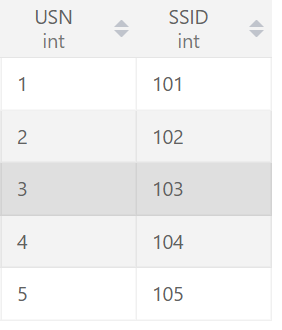
INSERT INTO CLASS VALUES(2,102);

INSERT INTO CLASS VALUES(3,103);

INSERT INTO CLASS VALUES(4,104);

INSERT INTO CLASS VALUES(5,105);

SELECT \* FROM CLASS;



**Create table – 4 Subject**

CREATE TABLE subject(Subcode BIGINT,Title VARCHAR(88),Sem VARCHAR(20),Credits int);

INSERT INTO subject VALUES(201,"Accounting","2nd",4);

INSERT INTO subject VALUES(202,"Art","2nd",2);

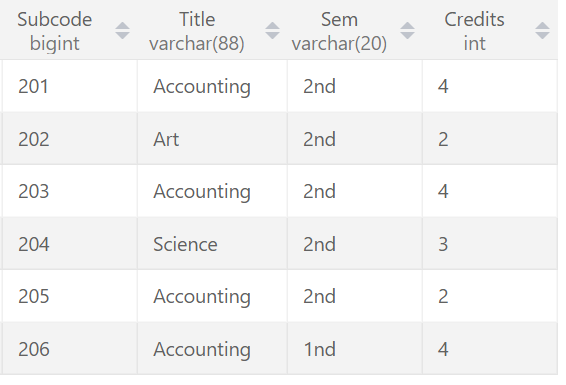
INSERT INTO subject VALUES(203,"Accounting","2nd",4);

INSERT INTO subject VALUES(204,"Science","2nd",3);

INSERT INTO subject VALUES(205,"Accounting","2nd",2);

INSERT INTO subject VALUES(206,"Accounting","2nd",4);

SELECT \*from subject;



**Create table - 5 IAMARKS**

CREATE TABLE IAMARKS(USN int,Subcode BIGINT,SSID int,Test1 int,test2 int,Test3 int, FinallA BIGINT);

INSERT into iamarks VALUES(1,201,101,99,99,99,297);

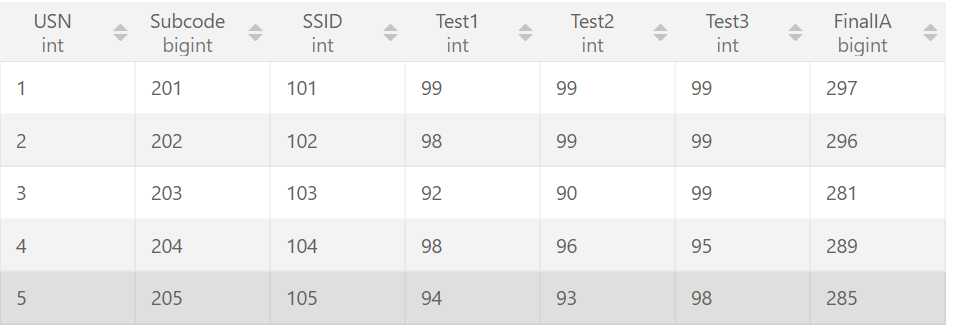
INSERT into iamarks VALUES(2,202,102,98,99,99,296);

INSERT into iamarks VALUES(3,203,103,92,90,99,281);

INSERT into iamarks VALUES(4,204,104,98,96,95,289);

INSERT into iamarks VALUES(5,205,105,94,93,98,285);

SELECT \* from IAMARKS;

****

**2.COMPANY DATABASE:**

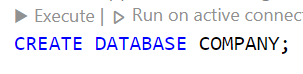
EMPLOYEE (SSN, Name, Address, Sex, Salary, SuperSSN, DNo)

DEPARTMENT (DNo, DName, MgrSSN, MgrStartDate)

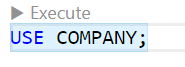
DLOCATION (DNo,DLoc)

PROJECT (PNo, PName, PLocation, DNo)

WORKS\_ON (SSN, PNo, Hours)

Create database Company ;

**Use Database**



**Create table - 1 Employee**

CREATE TABLE EMPLOYEE(

    SSN int NOT NULL PRIMARY KEY ,

    Name VARCHAR(50),

    Address VARCHAR(250),

    Sex CHAR(10),

    Salary BIGINT,

    SuperSSN INT,

    DNO INT

);

**Insert values in Employee table**

INSERT into employee VALUES(1,"Aman","Trilok Puri Delhi pin -110091","M",150000,101,201);

INSERT into employee VALUES(2,"Puneet","lok Puri Delhi pin -110091","M",140000,102,202);

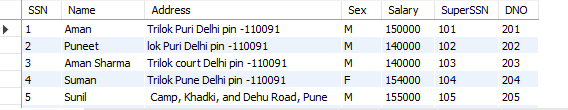
INSERT into employee VALUES(3,"Aman Sharma","Trilok court Delhi pin -110091","M",140000,103,203);

INSERT into employee VALUES(4,"Suman","Trilok Pune Delhi pin -110091","F",154000,104,204);

INSERT into employee VALUES(5,"Sunil"," Camp, Khadki, and Dehu Road, Pune","M",155000,105,205);

INSERT into employee VALUES(6,"bhavya","Khadki, and Dehu Road, Pune","F",150600,106,206);

SELECT \* FROM employee;

****

**Create table – 2 Department and Insert values**

CREATE Table DEPARTMENT(DNO INT,DName VARCHAR(60),MgrSSN BIGINT,MgrStartDate DATE);

INSERT into department VALUES(201,"Aman",101,"2016-02-20");

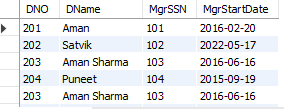
INSERT into department VALUES(202,"Satvik",102,"2022-05-17");

INSERT into department VALUES(203,"Aman Sharma",103,"2016-06-16");

INSERT into department VALUES(204,"Puneet",104,"2015-09-19");

INSERT into department VALUES(205,"bhavya",105,"2021-02-22");

SELECT \*from department;

****

**Create table - 3 DLOCATION and Insert values**

use company;

CREATE TABLE DLOCATION(DNO int,DLoc int);

INSERT into DLOCATION VALUES(201,301);

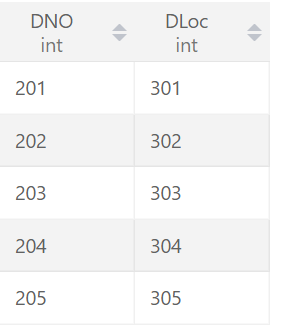
INSERT into DLOCATION VALUES(202,302);

INSERT into DLOCATION VALUES(203,303);

INSERT into DLOCATION VALUES(204,304);

INSERT into DLOCATION VALUES(205,305);

SELECT \*FROM dlocation;



**Create table - 4 PROJECT and Insert values**

CREATE Table PROJECT(PNo int, PName VARCHAR(50), PLocation VARCHAR(60),DNO int);

INSERT into project values (1,"Ai Power", "Model Town",201);

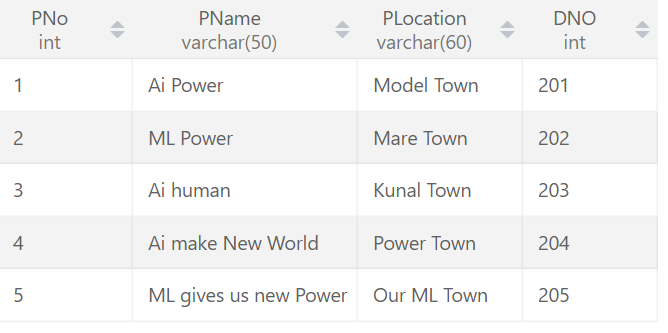
INSERT into project values (2,"ML Power", "Mare Town",202);

INSERT into project values (3,"Ai human", "Kunal Town",203);

INSERT into project values (4,"Ai make New World", "Power Town",204);

INSERT into project values(5,"ML gives us new Power", "Our ML Town",205);

select \*from PROJECT;



**Create table – 5 WORKS\_ON and Insert values**

CREATE Table WORKS\_ON(SSN INT,PNO int,Hours time);

desc WORKS\_ON;

INSERT into WORKS\_ON VALUES(1,1,'10:30:10');

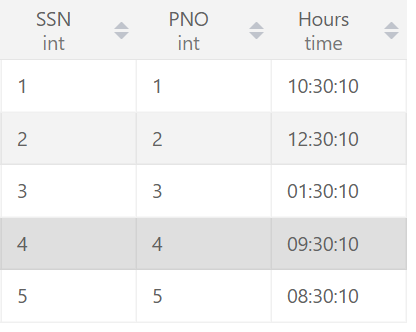
INSERT into WORKS\_ON VALUES(2,2,'12:30:10');

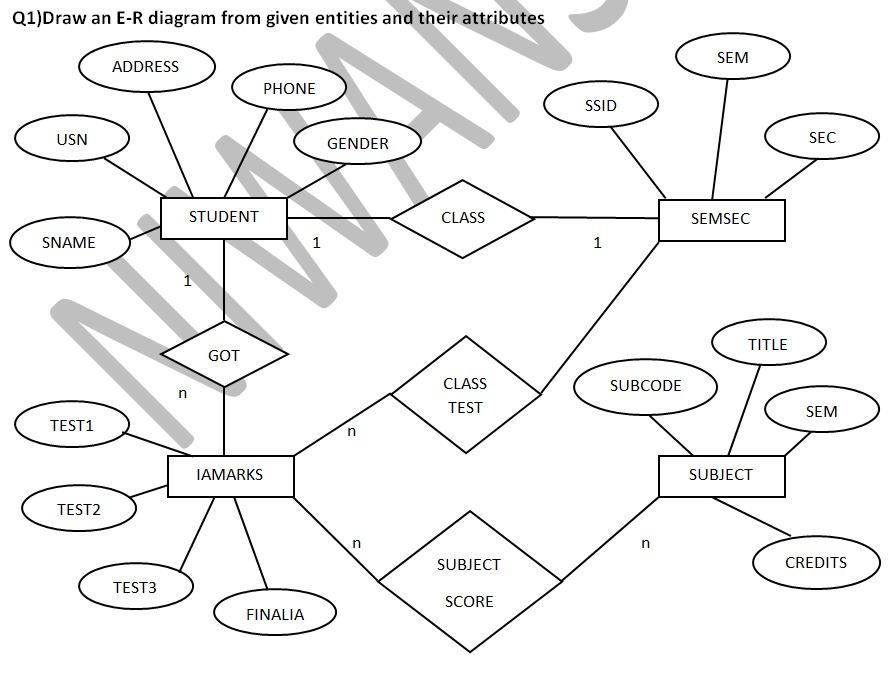
INSERT into WORKS\_ON VALUES(3,3,'01:30:10');

INSERT into WORKS\_ON VALUES(4,4,'09:30:10');

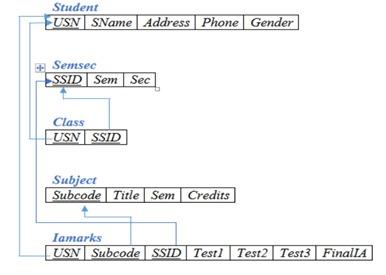
INSERT into WORKS\_ON VALUES(5,5,'08:30:10');

SELECT \* FROM WORKS\_ON;



****

**PRACTICAL -2**

****

**Program 3:-**Write queries to execute following DDL commands :

1. CREATE :
2. ALTER:
3. DROP:

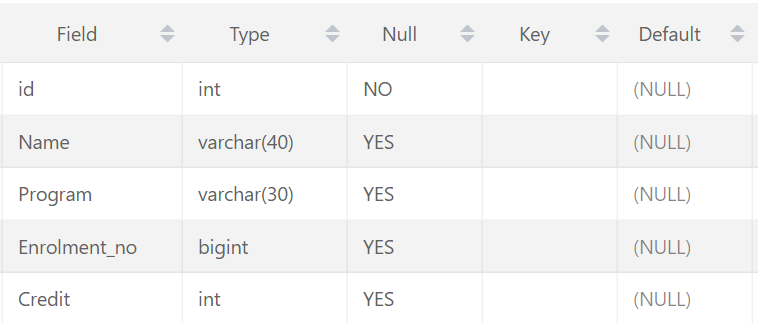
Create command query

CREATE DATABASE test;

use test;

CREATE table Student(id int, Name VARCHAR(40),Program VARCHAR(30),Enrolment\_no BIGINT, Credit int);

desc Student;



SELECT \* FROM student;

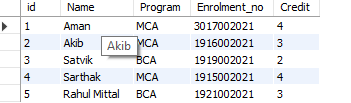
INSERT into student VALUES(1,'Aman','MCA',03017002021,4);

INSERT into student VALUES(2,'Akib','MCA',01916002021,3);

INSERT into student VALUES(3,'Satvik','BCA',01919002021,2);

INSERT into student VALUES(4,'Sarthak','MCA',01915002021,4);

INSERT into student VALUES(5,'Rahul Mittal','BCA',01921002021,3);



Q4 Alter command query

Change the size of a particular column.

alter TABLE student

MODIFY COLUMN Program char(50);



Add a new column to the existing table.

ALTER TABLE student

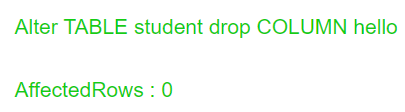
add COLUMN hello int;



Remove a column from the table.

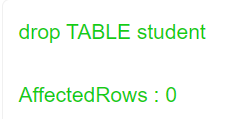
Alter TABLE student

drop COLUMN hello;



DROP: Destroy the table along with its data.

drop TABLE student;



**Program 4:-**Write queries to execute following DML commands :

INSERT: Insert five records in each table.

UPDATE: Modify data in single and multiple columns in a table

DELETE: Delete selective and all records from a table.

Insert five records in table.

INSERT into student VALUES(1,'Aman','MCA',03017002021,4);

INSERT into student VALUES(2,'Akib','MCA',01916002021,3);

INSERT into student VALUES(3,'Satvik','BCA',01919002021,2);

INSERT into student VALUES(4,'Sarthak','MCA',01915002021,4);

INSERT into student VALUES(5,'Rahul Mittal','BCA',01921002021,3);

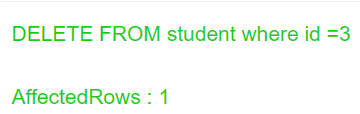
Modify data in single and multiple columns in a table

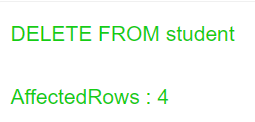
UPDATE student set Program ="Btech" WHERE id = 1;



DELETE: Delete selective and all records from a table.

DELETE FROM student where id =3;

  
DELETE FROM student;



**Program 5:-**Write queries to execute following DML command :

SELECT: Retrieve the entire contents of the table.

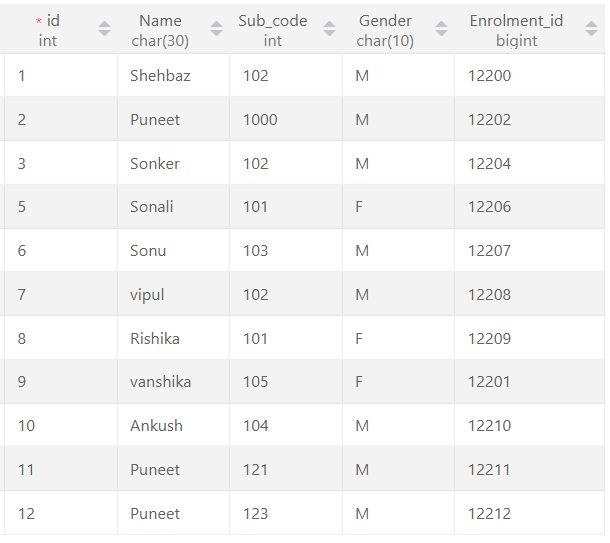
Retrieve the selective contents (based on provided conditions) from a table.

Retrieve contents from a table based on various operators

i.e. string operators, logical operators and conditional operators, Boolean operators.

Sort the data in ascending and descending order in a table on the basis of one column or more than one column.

SELECT \* FROM pin;



use hello;

SELECT \*from best1;

-- String Operations

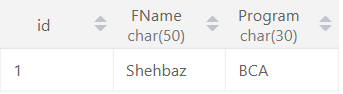
-- There are the two string operators:

--          Percent (%)  :  This Operator (%) matches any substring.

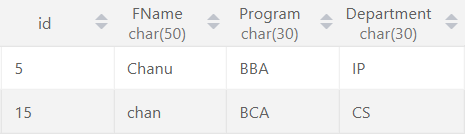
--          Underscore ( \_ ) : This Operator ( \_ ) matches any character.

-- String Operator

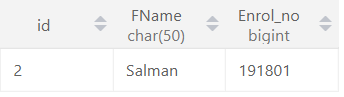
SELECT id, FName, Program FROM best1 WHERE FName LIKE 'Sh%';



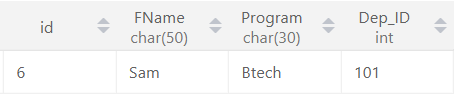
SELECT id, FName, Program, Department FROM best1 WHERE FName LIKE '\_ha%';



SELECT id, FName, Enrol\_no FROM best1 WHERE FName LIKE 'Sal\_%';



SELECT id, FName, Program, Dep\_ID FROM best1 WHERE FName LIKE 'S\_\_';

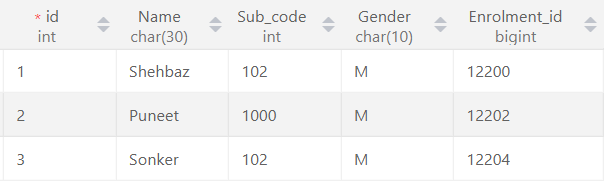


-- ......Logical Opreators........

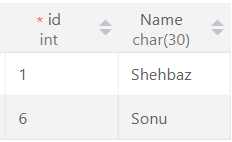
-- .....also We can use And(&&),OR(||) NOT(!).....

SELECT \* from pin;

SELECT \* FROM pin WHERE id <= 3 AND Gender = "M";

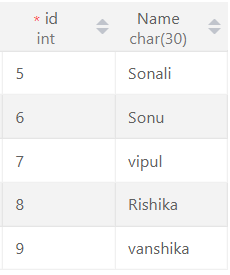


SELECT id, Name FROM pin WHERE id = 6 || id = 1;



SELECT id, Name FROM pin WHERE  not(id = 6 OR id = 1);

SELECT id, Name FROM pin WHERE id  BETWEEN 5 AND 9;

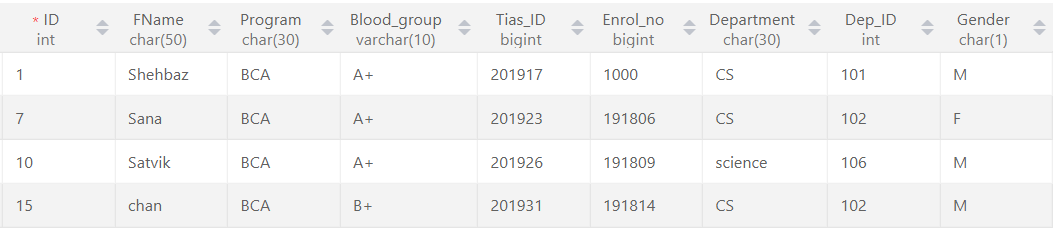


SELECT id, FName, Program FROM best1 WHERE Program IN('BCA' ,'BBA');

SELECT id, FName, Program FROM best1 WHERE FName LIKE 'S%';

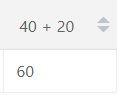
SELECT id, FName, Program FROM best1 WHERE FName LIKE '\_aa%';

SELECT \* FROM best1 WHERE Program = ALL (SELECT Program FROM best1 WHERE Program = 'BCA');



-- Arithmetic Operators ( + - \* / % )

SELECT 40 + 20;

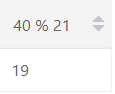


SELECT 40 - 20;

SELECT 40 \* 20;

SELECT 40 / 20;

SELECT 40 % 21;



-- Comparision Operator < > >= <= !> <! == != ,.....,

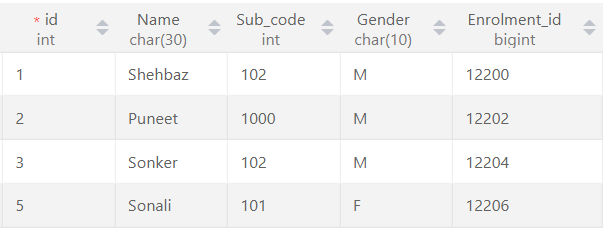
SELECT \* FROM pin ;

SELECT \*FROM Pin where id >5;

SELECT \*FROM Pin where id <5;

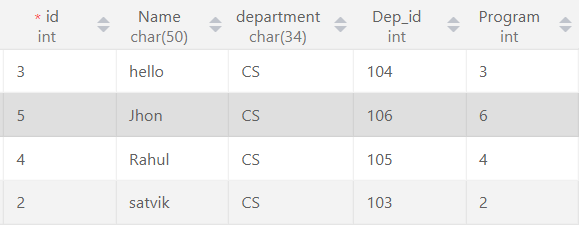
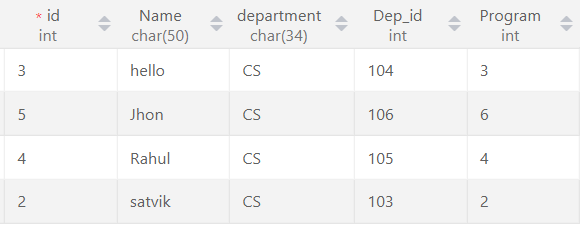
SELECT \*FROM Pin where id >=5;

SELECT \*FROM Pin where id <= 5;



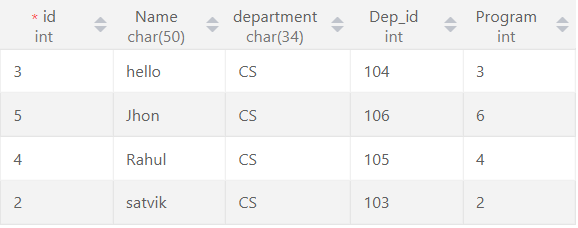
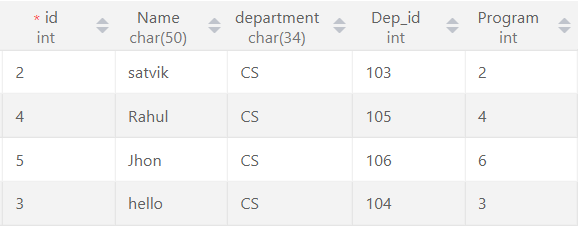
SELECT \*FROM Pin where id != 5;

\*\*\*Sort Table In Ascending Order by name basis of one column or more than one column.

SELECT \*from best ORDER BY Name; SELECT \*from best ORDER BY Name ,Program;   


Sort Table In descending Order by name basis of one column or more than one column.

SELECT \*from best ORDER BY Name DESC; SELECT \*from best ORDER BY Name ,Program DESC;



**Practical:-6** Create table using following integrity constraints:

1. Primary Key
2. Unique Key
3. Not Null
4. Check Default
5. Foreign Key

-- Child table for REFERENCES FOREIGN KEY;

CREATE TABLE City(

    ID INT ,

    City VARCHAR(160),

    PRIMARY KEY(ID)

    );

insert INTO city VALUES(1,"Agra");

insert INTO city VALUES(2,"Rohini");

insert INTO city VALUES(3,"Rithala");

insert INTO city VALUES(4,"Ankur vihar");

insert INTO city VALUES(5,"Kashmiri gate");

SELECT \*from city;

ALTER TABLE city  RENAME  COLUMN  ID to C\_ID;

-- /////////////////////////////////Main table

CREATE Table Personal(

    ID INT NOT NULL UNIQUE,

    Code\_NO INT UNIQUE,

    FNAME VARCHAR(60) NOT NULL,

    Age INT NOT NULL check (Age >=18),

    Phone\_no VARCHAR(10) UNIQUE,

    MyArea VARCHAR(80) NOT null DEFAULT 'Agra',

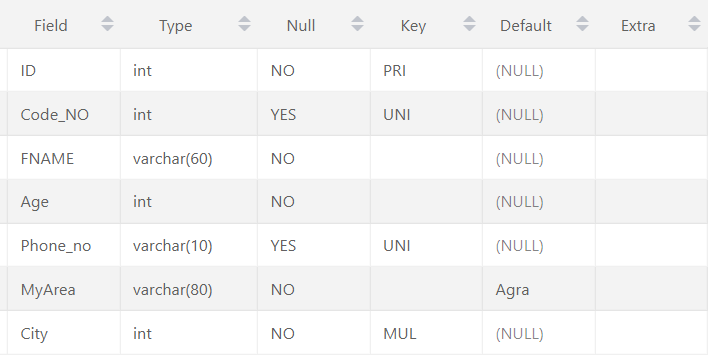
    City INT NOT NULL,

    PRIMARY KEY(ID),

    FOREIGN KEY (City)REFERENCES City(C\_ID)

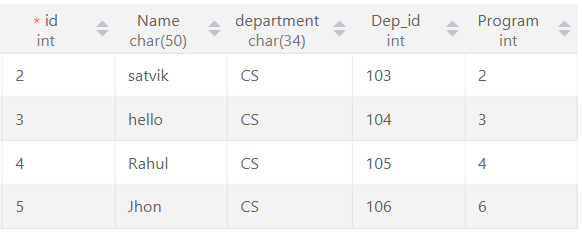
);

DESC Personal;

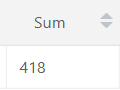


**Practical:-7** Write queries to execute following Aggregate functions Sum, Avg, Count, Minimum and Maximum value of a numeric column of a table using aggregate function.

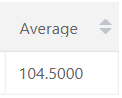
SELECT \*from best;



SELECT SUM(Dep\_id) As Sum  FROM best;



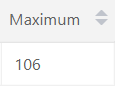
SELECT AVG(Dep\_id) As Average  FROM best;



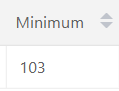
SELECT COUNT(Dep\_id) FROM best;



SELECT MAX(Dep\_id) As Maximum  FROM best;



SELECT MIN(Dep\_id) As Minimum  FROM best;



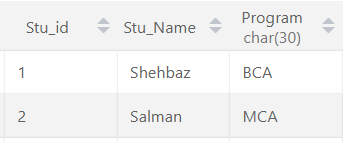
**Practical:-8** Retrieve data from a table using alias names .

**SELECT column as alias\_name FROM table\_name;**

**column:** fields in the table

**alias\_name:** temporary alias name to be used in replacement of original column name

SELECT id as Stu\_id, FName as Stu\_Name, Program FROM best1 WHERE FName LIKE 'S%';

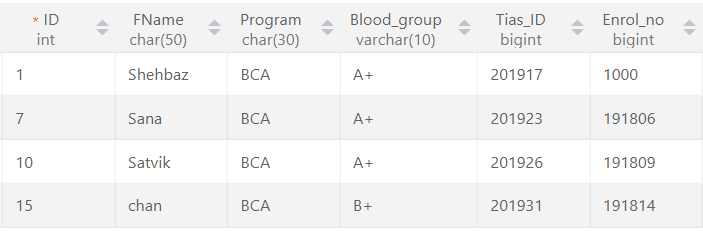


SELECT MIN(Dep\_id) As Minimum  FROM best;

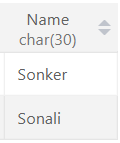


**Practical:- 9** Retrieve data of a table using nested queries.

SELECT \* FROM best1 WHERE Program = ALL (SELECT Program FROM best1 WHERE Program = 'BCA');



SELECT Name from pin WHERE id = ANY(SELECT id FROM best WHERE  Dep\_id>103);

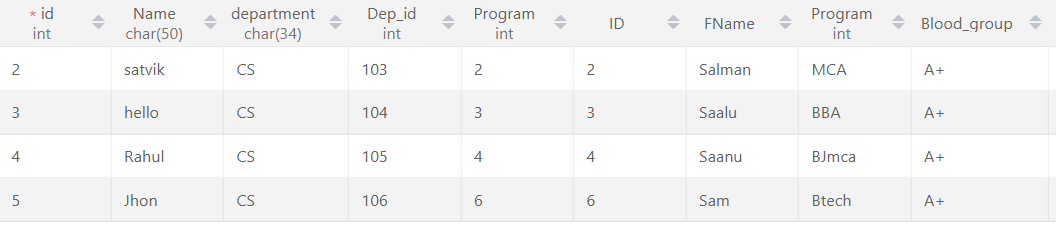
****

**Practical:-10** Retrieve data from more than one table using inner join, left outer, right outer and full outer joins

\*\*\*INNER JOIN means intersection of two table

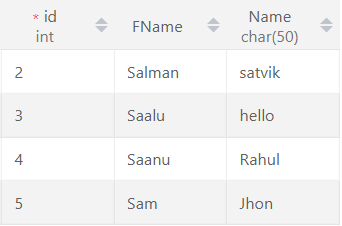
SELECT \* from best INNER JOIN best1

ON best.Program = best1.ID;

****

SELECT b.id,b1.FName,b.Name from best b INNER JOIN best1 b1

ON b.Program = b1.ID;

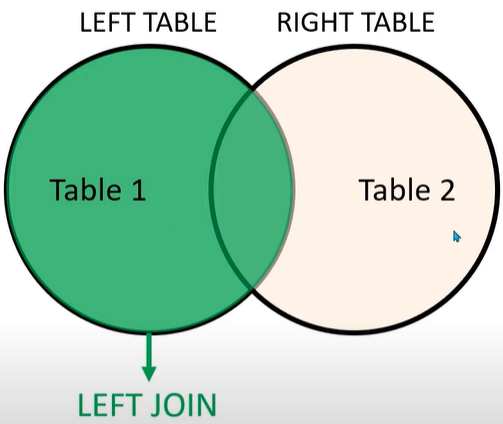
****

SELECT b.id,b.Name,b1.Program,b1.FName from best b INNER JOIN best1 b1

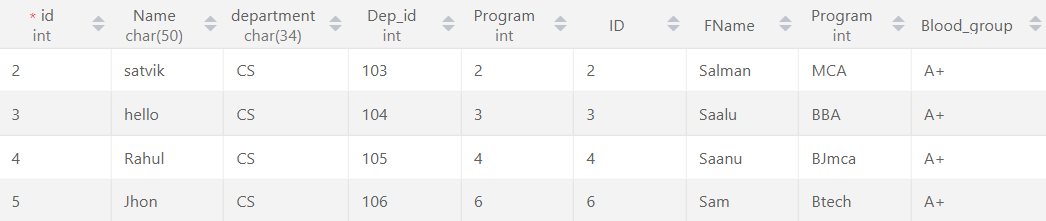
ON b.Program = b1.ID WHERE b1.Program = "BBA";

****

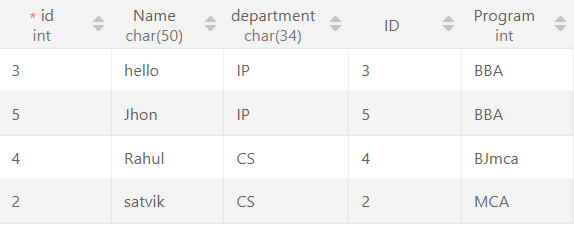
**\*\*\*\*The left join returns all records from the left table(t1), and the matched records from right table(t2)**

****

SELECT \* FROM best LEFT JOIN best1 ON best.Program = best1.ID;

****

SELECT b.id,b.Name,b1.department,b1.ID,b1.Program FROM best b LEFT JOIN best1 b1 ON b.id = b1.ID ORDER BY Name ;

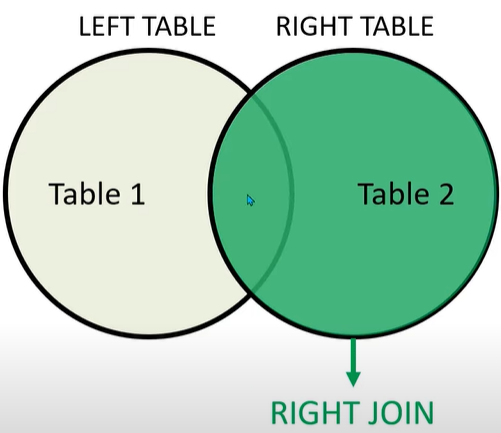
****

SELECT b.id,b.Name,b1.department,b1.ID,b1.Program FROM best b LEFT JOIN best1 b1

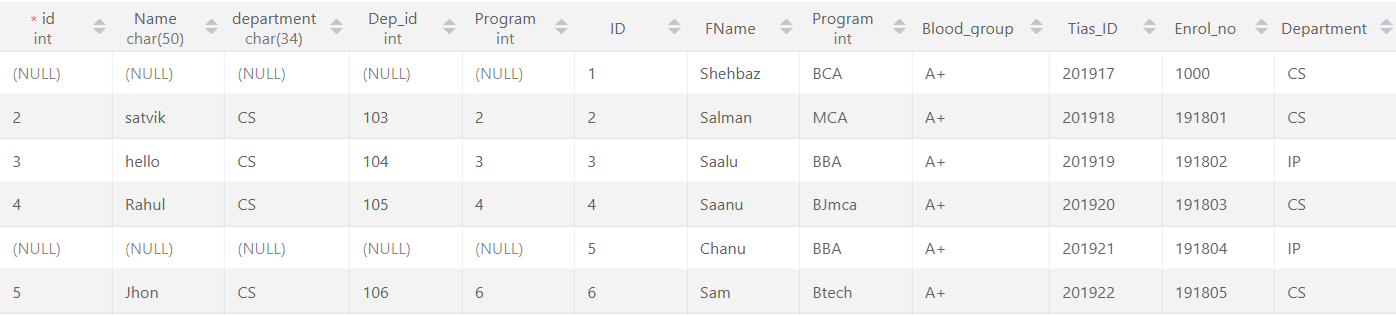
ON b.id = b1.ID WHERE b1.Program="MCA";

****

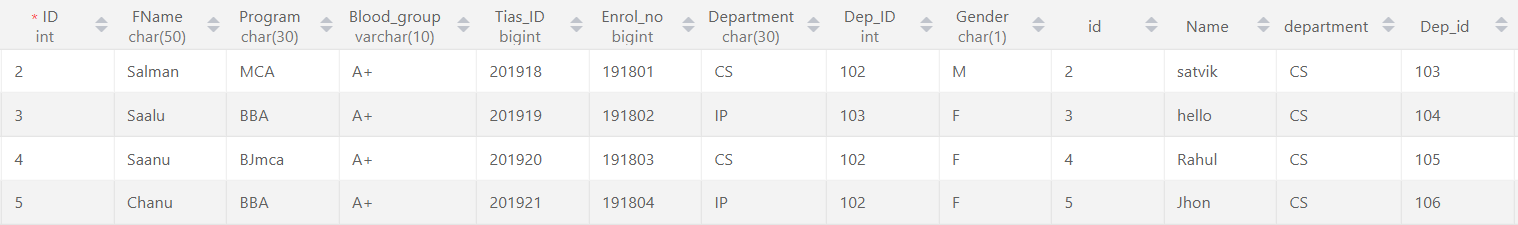
**\*\*\*\*The Right join returns all records from the right table(t2), and the matched records from left table(t1)**

****

SELECT \* FROM best RIGHT JOIN best1 ON best.Program = best1.ID;

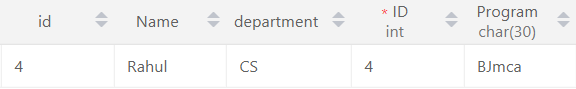
****

SELECT \* FROM best1 b1 RIGHT JOIN best b ON b1.ID = b.id;



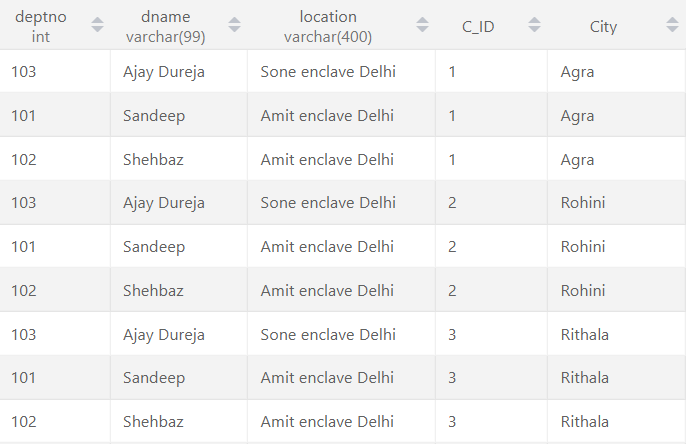
SELECT b.id,b.Name,b1.department,b1.ID,b1.Program FROM best1 b1 RIGHT JOIN best b

ON b1.ID = b.id WHERE b1.Program="Bjmca";



\*\*\*Cross join means every 1 row of t1 repeat with respect to t2 \*\*\*\*

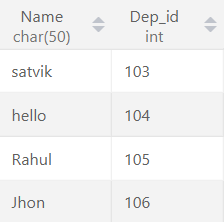
SELECT \* FROM dept CROSS JOIN city;

****

**Practical:- 11** Create view from one table and more than one table.

CREATE VIEW Stu\_info as select Name, Dep\_id FROM best ,Program\_  WHERE  best.Program = program\_.program\_id;

SELECT \* FROM stu\_info;

****

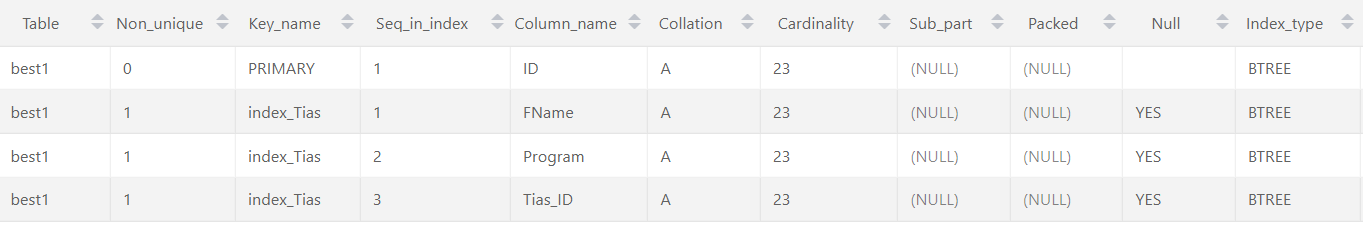
**Practical:- 12** Create index on a column of a table.

use hello;

SELECT \* FROM best1;

create index index\_Tias on best1(FName, Program, Tias\_ID);

show index from best1;



drop index index\_Tias on best1;

**Practical:- 13** Consider the Insurance company’s Database given below.

The primary keys are underlined and the data typesare specified.

PERSON(driver\_id# : string, name : string, address : string)

CAR(regno : string, model : string, year : int)

ACCIDENT(report\_number : int, acc\_date : date, location : string)

OWNS(driver\_id# : string, regno : string)

PARTICIPATED(driver\_id# : string, regno : string, report\_number : int, damage\_amount :number(10,2) )

1. Create the above tables by properly specified the primary key and the foreign key
2. Enter at least five tuples for each relation
3. (A)Demonstrate how you can a. Update the damage amount for the car with a specific regno, the accident with report number 12 to 25000.

(B). Add a new accident to the database.

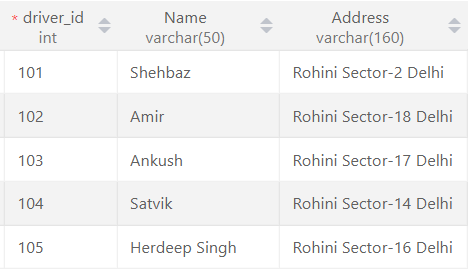
1. Find the total number of people who owned cars that were involved in accident in2002.
2. Find the number of accident in which cars belonging to a specific models were involved

CREATE DATABASE  Insurance\_company;

use Insurance\_company;

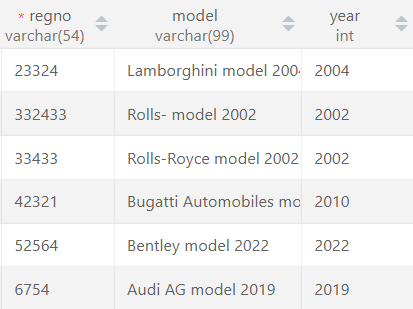
CREATE TABLE PERSON(driver\_id int PRIMARY KEY,Name VARCHAR(50),Address VARCHAR(160));

SELECT \* FROM PERSON;

****

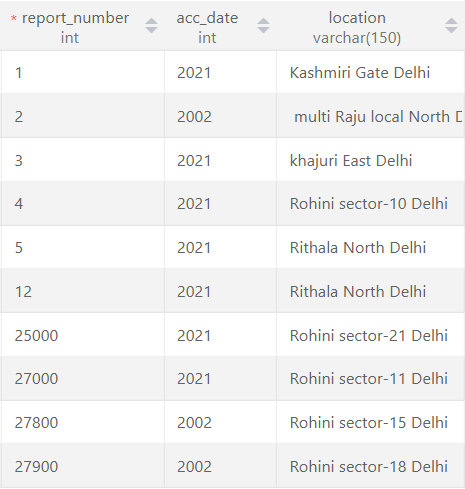
CREATE TABLE CAR(regno VARCHAR(54) PRIMARY KEY, model VARCHAR(99), year int);

SELECT \* FROM CAR;

****

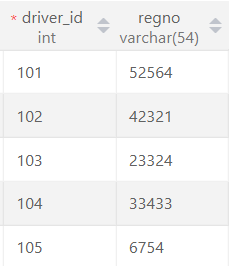
CREATE TABLE ACCIDENT(report\_number int PRIMARY KEY, acc\_date INT, location VARCHAR(150));

SELECT \* FROM accident;

****

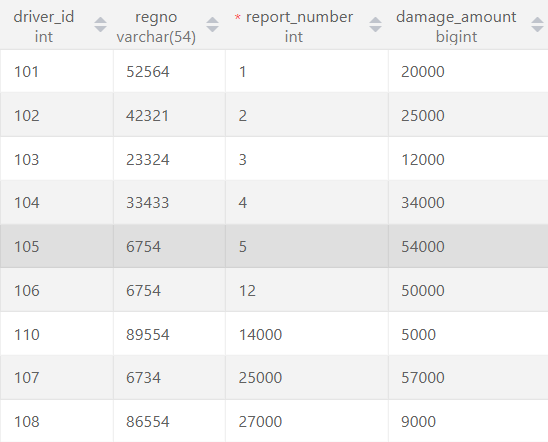
CREATE TABLE OWNS(driver\_id INT PRIMARY KEY, regno VARCHAR(54));

SELECT \* FROM owns;

****

CREATE Table PARTICIPATED(driver\_id int UNIQUE KEY , regno VARCHAR(54), report\_number int PRIMARY KEY, damage\_amount BIGINT );

SELECT \* FROM participated;

****

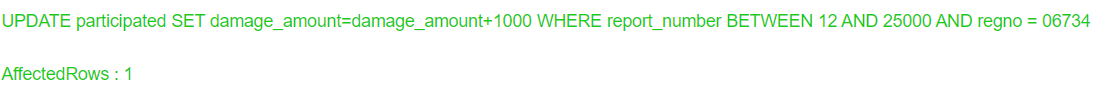
--(i) Create the above tables by properly specified the primary key and the foreign key

ALTER TABLE  accident ADD FOREIGN KEY (report\_number) REFERENCES participated(report\_number);

--(ii) (A) Update the damage amount for the car with a specific regno, the

-- accident with report number 12 to 25000.

UPDATE participated SET damage\_amount=damage\_amount+1000  WHERE report\_number BETWEEN 12 AND 25000 AND regno = 06734;

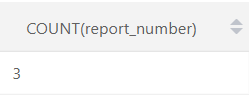


--(ii) (B) Add a new accident to the database.

INSERT INTO accident VALUES(28000,'2021-01-13',"Rohini sector-11 Delhi");

--(iii) Find the total number of people who owned cars that were involved in accident in2002

SELECT COUNT(report\_number) FROM accident WHERE acc\_date =2002;

****

-- (iv) Find the number of accident in which cars belonging to a specific models were involved

Select count(report\_number) as accident\_ from ACCIDENT  where  report\_number  IN (select report\_number from car where model='Rolls-Royce model\_ ' OR model="Audi AG model\_ ");



**Practical:- 14**

-- Consider the following schema of a library\_management system.Write the SQL

-- queries for the questions given below;

-- Student(Stud\_no : integer, Stud\_name: string)

-- Membership(Mem\_no: integer, Stud\_no: integer)

-- Book\_(book\_no: integer, book\_name:string, author: string)

-- lss\_rec\_(iss\_no:integer, iss\_date: date, Mem\_no: integer, book\_no: integer)

-- (i) Create the tables with the appropriate integrity constraints

-- (ii) Insert around 10 records in each of the tables

-- (iii)Display all records for all tables

-- (iv)List all the student names with their membership numbers

-- (v) List all the issues for the current date with student and Book names

-- (vi) List the details of students who borrowed book whose author is Elmarsi & Navathe

-- (vii) Give a count of how many books have been bought by each student

-- (viii) Give a list of books taken by student with stud\_no as 1005

-- (ix) Delete the List of books details which are issued as of today

-- (x) Create a view which lists out the iss\_no, iss \_date, stud\_name, book name

CREATE DATABASE library\_management;

use library\_management;

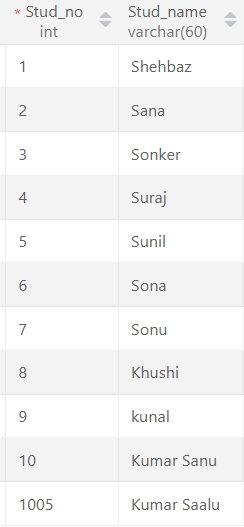
(i) Create the tables with the appropriate integrity constraints

(ii) Insert around 10 records in each of the tables

(iii)Display all records for all tables

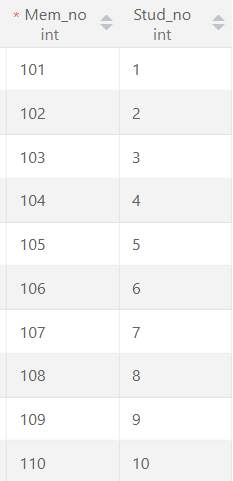
CREATE TABLE Student(Stud\_no  int PRIMARY KEY, Stud\_name VARCHAR(60));

SELECT \* FROM student;



CREATE TABLE Membership(Mem\_no int PRIMARY KEY, Stud\_no int);

SELECT \* FROM membership;



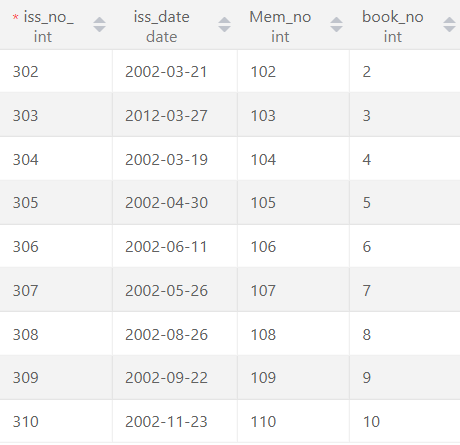
CREATE TABLE Book\_(book\_no int, book\_name VARCHAR(200), author VARCHAR(200));

TRUNCATE TABLE Book\_;

SELECT \*FROM book\_;

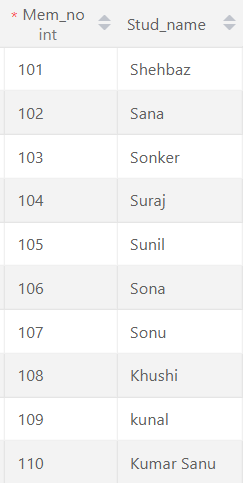


CREATE TABLE lss\_rec(iss\_no\_ int PRIMARY KEY, iss\_date date, Mem\_no int, book\_no int);



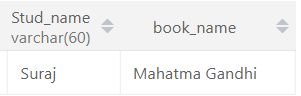
-- (iv)List all the student names with their membership numbers

SELECT Mem\_no,Stud\_name FROM membership JOIN student WHERE membership.Stud\_no =student.Stud\_no;



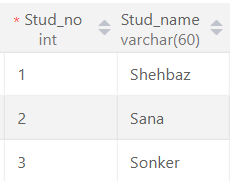
-- (v) List all the issues for the current date with student and Book names

select Stud\_name, book\_name from student, book\_, lss\_rec ,membership where iss\_date= '2002-03-19' AND student.Stud\_no= membership.Stud\_no AND  lss\_rec.Mem\_no= membership.Mem\_no AND lss\_rec.book\_no= book\_.book\_no;

****

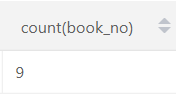
-- (vi)List the details of students who borrowed book whose author is Elmarsi & Navathe

Select \*from student where Stud\_no IN(select book\_no from book\_ where author = 'Elmarsi & Navathe');



-- (vii) Give a count of how many books have been bought by each student

select count(book\_no) from lss\_rec;



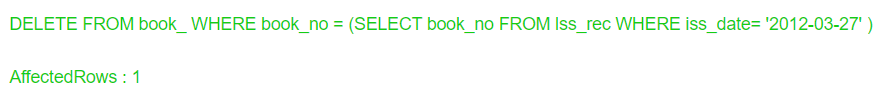
-- (viii) Give a list of books taken by student with stud\_no as 1005

select book\_name from book\_ where book\_no=(select book\_no from lss\_rec where Mem\_no=(select Mem\_no from membership where Stud\_no= 1005));



-- (ix) Delete the List of books details which are issued as of today

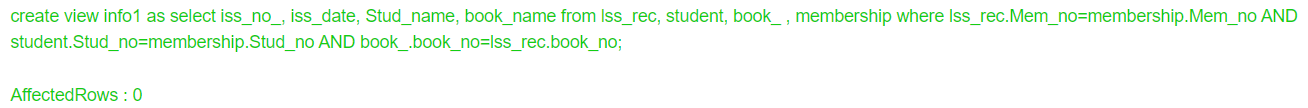
DELETE FROM book\_ WHERE book\_no = (SELECT book\_no FROM lss\_rec WHERE iss\_date= '2012-03-27' );

****

-- (x) Create a view which lists out the iss\_no, iss \_date, stud\_name, book name

create view info1 as select iss\_no\_, iss\_date, Stud\_name, book\_name from lss\_rec, student, book\_ , membership where lss\_rec.Mem\_no=membership.Mem\_no AND student.Stud\_no=membership.Stud\_no AND book\_.book\_no=lss\_rec.book\_no;

select \*from info1;

****

**Program 15**

Use the relations below to write SQL queries to solve the business problems specified.

CLIENT (clientno#,name, client\_referred\_by#)

ORDER (orderno#, clientno#, order\_date, empid#)

ORDER\_LINE (orderno#, order line number#, item\_number#, no\_of\_items, item\_ cost,shipping\_date)

ITEM (item\_number#, item\_type, cost)

EMPLOYEE (empid#, emp\_type#, deptno, salary, firstname, lastname)

Notes:

a. Column followed by # is the primary key of the table.

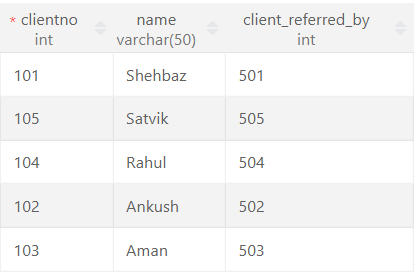
b. Each client may be referred by another client. If so, the client number of the referring client is stored in referred\_by.

c. The total cost for a particular order line = no\_of\_items \* item\_cost.c. Write queries for the following

1. Create all the above tables.
2. Insert at least five records.
3. Display all the rows and columns in the CLIENT table. Sort by client name in reverse alphabetical order.
4. Display the item number and total cost for each order line (total cost = no of items X item cost). Name the calculated column TOTAL COST.
5. Display all the client numbers in the ORDER table. Remove duplicates.
6. Display the order number and client number from the ORDER table. Output the result in the format. Client ordered
7. Display full details from the ORDER\_LINE table where the item number is (first condition) between 1 and 200 (no > or < operators) OR the item number is greater than1000 AND (second condition) the item cost is not in the list 1000, 2000, 3000 OR the order number is not equal to 1000.
8. Display the client name and order date for all orders.
9. Repeat query (6) but also display all clients who have never ordered anything.
10. Display the client name and order date for all orders using the join keywords.
11. Display the client name and order date for all orders using the JOIN method.
12. Display the client number, order date and shipping date for all orders where the shipping date is between three and six months after the order date.
13. (xiii) Display the client number and name and the client number and name of the person who referred that client.
14. (xiv) Display the client name in upper case only and in lower case only.
15. (xv) Display the second to fifth characters in each client name.

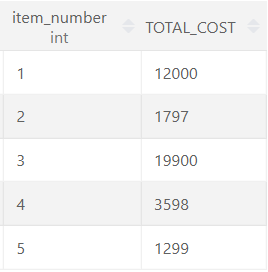
-- (iii) Display all the rows and columns in the CLIENT table. Sort by client name in reverse alphabetical order.

SELECT \* FROM client ORDER BY name DESC;

****

--(iv) Display the item number and total cost for each order line (total cost = no of items X item cost). Name the calculated column TOTAL COST.

SELECT item\_number,order\_line\_ AS TOTAL\_COST FROM ORDER\_LINE;

****

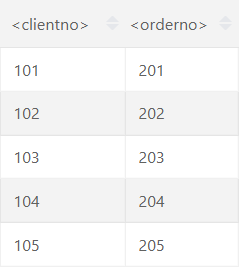
-- (v) Display all the client numbers in the ORDER table. Remove duplicates.

SELECT DISTINCT clientno AS client\_numbers FROM client;

****

-- (vi) Display the order number and client number from the ORDER table. Output the result in the format. Client <clientno> ordered <orderno>

SELECT clientno AS "<clientno>",orderno AS "<orderno>" from order\_;

****

--(vii) Display full details from the ORDER\_LINE table where the item

-- number is (first condition) between 1 and 200 (no > or < operators)

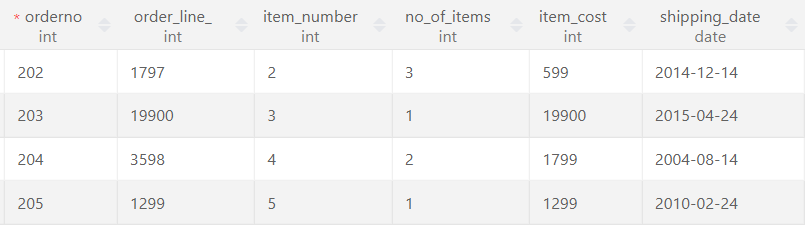
--OR the item number is greater than1000 AND

-- (second condition) the item

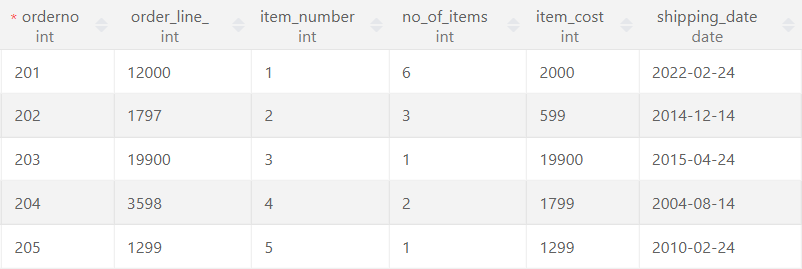
-- cost is not in the list 1000, 2000, 3000 OR the order number is not

-- equal to 1000.

SELECT \* FROM order\_line where item\_number BETWEEN 2 AND 5 OR item\_number>2;

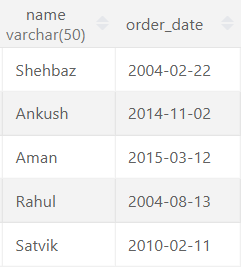
****

SELECT \* FROM order\_line where item\_cost NOT IN(1000,2000,3000) OR NOT(orderno=1000);

****

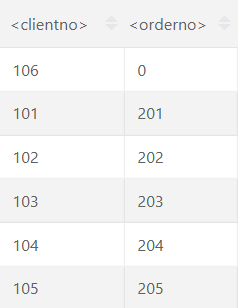
-- (viii) Display the client name and order date for all orders.

SELECT name,order\_date FROM client,order\_ WHERE client.clientno = order\_.clientno ;

****

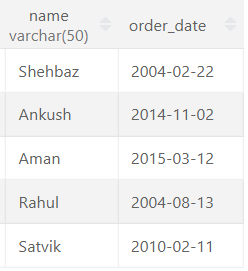
(ix) Repeat query (6) but also display all clients who have never ordered anything.

SELECT clientno AS "<clientno>",orderno AS "<orderno>" from order\_;

****

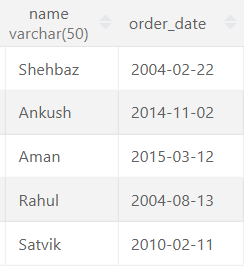
-- (x) Display the client name and order date for all orders using the join keywords.

SELECT name,order\_date FROM client JOIN order\_ ON client.clientno = order\_.clientno ;

**4**

-- (xi) Display the client name and order date for all orders using the JOIN method.

SELECT name,order\_date FROM client INNER JOIN order\_ ON client.clientno = order\_.clientno ;

****

-- (xiv) Display the client name in upper case only and in lower case only.

SELECT lower(name) FROM client;

SELECT UPPER(name) FROM client;



-- (xv) Display the second to fifth characters in each client name.

SELECT SUBSTRING(name,2,5) FROM client;

