Name: SaiPraveen Marni

USN: 1BM19CS138

Dept: CSE

Section: C

Course Name: Database Management Systems

Lab_batch: C-3

PROGRAM 6: ORDER PROCESSING DATABASE

Consider the following relations for an Order Processing database application in a company.

CUSTOMER (CUST #: int, cname: String, city: String) ORDER (order #: int, odate: date, cust #: int, ord-Amt: int)

ITEM (item #: int, unit-price: int)

ORDER-ITEM (order #: int, item #: int, qty: int)

WAREHOUSE (warehouse #: int, city: String)

SHIPMENT (order #: int, warehouse #: int, ship-date: date)

- i. Create the above tables by properly specifying the primary keys and the foreign keys and the foreign keys.
- ii. Enter at least five tuples for each relation.
- iii. Produce a listing: CUSTNAME, #oforders, AVG_ORDER_AMT, where the middle column is the total numbers of orders by the customer and the last column is the average order amount for that customer.
- iv. List the order# for orders that were shipped from all warehouses that the company has in a specific city.
- v. Demonstrate how you delete item# 10 from the ITEM table and make that field null in the ORDER_ITEM table.

```
create database orderpro;
use orderpro;

CREATE TABLE CUSTOMERS(
    CUSTNO INT,
    CNAME VARCHAR2(50),
    CITY VARCHAR2(50),
    PRIMARY KEY(CUSTNO)
);
```

```
CREATE TABLE ORDERS(
 ORDERNO INT,
 ODATE DATE,
 CUSTNO INT,
 ORD_AMT INT,
 PRIMARY KEY(ORDERNO),
 FOREIGN KEY(CUSTNO) REFERENCES CUSTOMERS(CUSTNO)
 );
CREATE TABLE ITEM(
 ITEMNO INT,
 UNIT_PRICE INT,
 PRIMARY KEY(ITEMNO)
 );
CREATE TABLE ORDER_ITEM(
 ORDERNO INT,
 ITEMNO INT,
 QTY INT,
 PRIMARY KEY(ORDERNO, ITEMNO),
 FOREIGN KEY(ORDERNO) REFERENCES ORDERS(ORDERNO),
 FOREIGN KEY(ITEMNO) REFERENCES ITEM(ITEMNO)
 );
CREATE TABLE WAREHOUSE(
 WAREHOUSENO INT,
 CITY VARCHAR2(50),
 PRIMARY KEY(WAREHOUSENO)
 );
```

```
CREATE TABLE SHIPMENT(
 ORDERNO INT,
 WAREHOUSENO INT,
 SHIP DATE DATE,
 PRIMARY KEY(ORDERNO, WAREHOUSENO),
 FOREIGN KEY(ORDERNO) REFERENCES ORDERS(ORDERNO),
 FOREIGN KEY(WAREHOUSENO) REFERENCES WAREHOUSE(WAREHOUSENO)
 );
INSERT INTO CUSTOMERS VALUES(CUSTNO, 'CNAME', 'CITY');
INSERT INTO ORDERS VALUES(ORDERNO, 'ODATE', CUSTNO, ORD AMT);
INSERT INTO ITEM VALUES(ITEMNO, UNIT_PRICE);
INSERT INTO ORDER_ITEM VALUES(ORDERNO, ITEMNO, QTY);
INSERT INTO WAREHOUSE VALUES(WAREHOUSENO, 'CITY');
INSERT INTO SHIPMENT VALUES(ORDERNO, WAREHOUSENO, 'SHIP DATE');
SELECT * FROM CUSTOMERS;
SELECT * FROM ORDERS;
SELECT * FROM ITEM;
SELECT * FROM ORDER_ITEM;
SELECT * FROM WAREHOUSE;
SELECT * FROM SHIPMENT;
INSERT INTO customers (custno, `cname`, `city`) VALUES
(111, 'hari', 'bangalore'),(112, 'siddu', 'bangalore'),(113, 'praveen', 'bangalore'),(114, 'yash',
'bangalore'),(115, 'anil', 'bangalore'),(116, 'piyush', 'bangalore');
```



INSERT INTO item (itemno, unit_price) VALUES

(11, 500),(12, 600),(14, 300),(15, 200),(16, 1000);



INSERT INTO orders (orderno, `odate`, custno, ord_amt) VALUES (2001, '2020-06-11', 111, 1200),(2002, '2020-05-12', 112, 1300), (2003, '2020-08-13', 113, 1100),(2004, '2020-11-14', 114, 1400), (2005, '2020-06-15', 115, 1500),(2006, '2020-12-16', 116, 1100);



INSERT INTO order_item (orderno, itemno, qty) VALUES (2001, 11, 2),(2002, 16, 1),(2003, 15, 5),(2004, 14, 3),(2006, 12, 1);



INSERT INTO shipment (orderno, warehouseno, `ship_date`) VALUES (2001, 31, '2020-06-11'),(2002, 32, '2020-05-12'),

(2003, 33, '2020-08-13'),(2004, 34, '2020-11-14'),

(2005, 35, '2020-06-15');

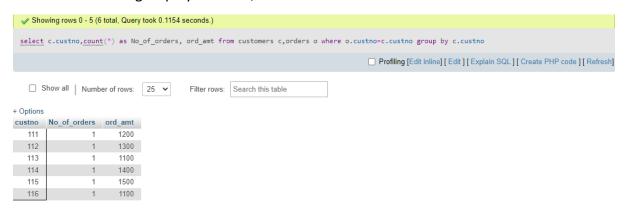


INSERT INTO warehouse (warehouseno, `city`) VALUES

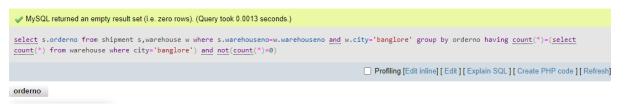
(31, 'ranchi'),(32, 'bangalore'),(33, 'mumbai'),(34, 'delhi'),(35, 'punjab');



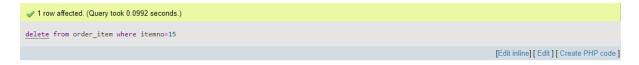
select c.custno,count(*) as No_of_orders,Avg_order_amt from customers c,orders o where o.custno=c.custno group by c.custno; #1



select s.orderno from shipment s,warehouse w where s.warehouseno=w.warehouseno and w.city='banglore' group by orderno having count(*)=(select count(*) from warehouse where city='banglore') and not(count(*)=0); #2



delete from item where itemno=15; #3



PROGRAM 7: BOOK DEALER DATABASE

The following tables are maintained by a book dealer:

AUTHOR(author-id: int, name: String, city: String, country: String)

PUBLISHER(publisher-id: int, name: String, city: String, country: String)

CATALOG (book-id: int, title: String, author-id: int, publisher-id: int, category-id:

int, year: int, price: int)

CATEGORY(category-id: int, description: String)

ORDER-DETAILS(order-no: int, book-id: int, quantity: int)

- i. Create the above tables by properly specifying the primary keys and the foreign keys.
- ii. Enter at least five tuples for each relation.
- iii. Give the details of the authors who have 2 or more books in the catalog and the price of the books in the catalog and the year of publication is after 2000.
- iv. Find the author of the book which has maximum sales.
- v. Demonstrate how you increase the price of books published by a specific publisher by 10%.

```
CREATE DATABASE BOOK;
USE BOOK;

CREATE TABLE AUTHORS(
   AUTHOR_ID INT,
   A_NAME VARCHAR(10),
   CITY VARCHAR(10),
   COUNTRY VARCHAR(10),
   PRIMARY KEY(AUTHOR_ID)
  );
```

CREATE TABLE PUBLISHERS(

```
PUBLISHER_ID INT,
 P NAME VARCHAR(10),
 CITY VARCHAR(20),
 COUNTRY VARCHAR(10),
 PRIMARY KEY(PUBLISHER_ID)
 );
CREATE TABLE CATEGORY(
 CATEGORY_ID INT,
 DESCRIPTIONS VARCHAR(10),
 PRIMARY KEY(CATEGORY_ID)
 );
CREATE TABLE CATALOG(
 BOOK ID INT,
 TITLE VARCHAR(10),
 AUTHOR_ID INT,
 PUBLISHER_ID INT,
 CATEGORY ID INT,
 P YEAR INT,
 PRICE INT,
 PRIMARY KEY(BOOK_ID),
 FOREIGN KEY(AUTHOR_ID) REFERENCES AUTHORS(AUTHOR_ID),
 FOREIGN KEY(PUBLISHER_ID) REFERENCES PUBLISHERS(PUBLISHER_ID),
 FOREIGN KEY(CATEGORY_ID) REFERENCES CATEGORY(CATEGORY_ID)
 );
CREATE TABLE ORDER_DETAILS(
 ORDER_NO INT,
```

```
BOOK_ID INT,

QTY INT,

PRIMARY KEY(ORDER_NO),

FOREIGN KEY (BOOK_ID) REFERENCES CATALOG(BOOK_ID)

);
```

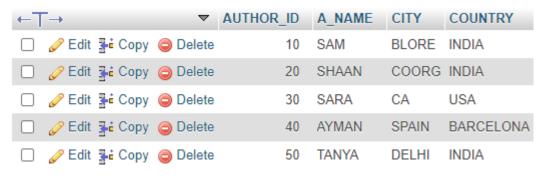
INSERT INTO AUTHORS VALUES(10,'SAM','BLORE','INDIA');

INSERT INTO AUTHORS VALUES(20, 'SHAAN', 'COORG', 'INDIA');

INSERT INTO AUTHORS VALUES(30, 'SARA', 'CA', 'USA');

INSERT INTO AUTHORS VALUES(40, 'AYMAN', 'SPAIN', 'BARCELONA');

INSERT INTO AUTHORS VALUES(50, 'TANYA', 'DELHI', 'INDIA');



SELECT * FROM AUTHORS;

INSERT INTO PUBLISHERS VALUES(1,'PHI','NY','USA');

INSERT INTO PUBLISHERS VALUES(2,'EEE','LA','USA');

INSERT INTO PUBLISHERS VALUES(3,'SWAPNA',",'INDIA');

INSERT INTO PUBLISHERS VALUES(4, 'MGH', 'NY', 'USA');

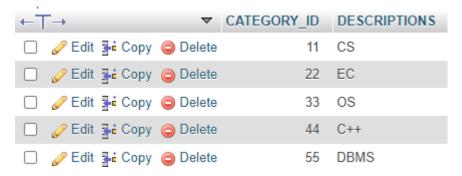
INSERT INTO PUBLISHERS VALUES(5, 'PEARSON', 'DELHI', 'INDIA');



SELECT * FROM PUBLISHERS;

```
INSERT INTO CATEGORY VALUES(11,'CS');
INSERT INTO CATEGORY VALUES(22,'EC');
INSERT INTO CATEGORY VALUES(33,'OS');
INSERT INTO CATEGORY VALUES(44,'C++');
```

INSERT INTO CATEGORY VALUES(55, 'DBMS');



SELECT * FROM CATEGORY;

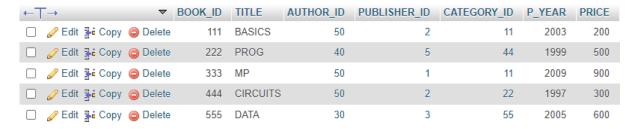
INSERT INTO CATALOG VALUES(111, BASICS', 50, 2, 11, 2003, 200);

INSERT INTO CATALOG VALUES(222, 'PROG', 40, 5, 44, 1999, 500);

INSERT INTO CATALOG VALUES(333, MP', 50, 1, 11, 2009, 900);

INSERT INTO CATALOG VALUES(444, 'CIRCUITS', 50, 2, 22, 1997, 300);

INSERT INTO CATALOG VALUES(555, 'DATA', 30, 3, 55, 2005, 600);



SELECT * FROM CATALOG;

INSERT INTO ORDER_DETAILS VALUES(123,222,100);
INSERT INTO ORDER_DETAILS VALUES(231,111,150);
INSERT INTO ORDER_DETAILS VALUES(143,333,90);

INSERT INTO ORDER_DETAILS VALUES(156,555,200);

INSERT INTO ORDER DETAILS VALUES(218,444,70);



SELECT * FROM ORDER DETAILS;

SELECT A.A_NAME,C.TITLE,C.PRICE FROM AUTHORS A,CATALOG C WHERE C.AUTHOR_ID=A.AUTHOR_ID AND C.P_YEAR>=2000 AND A.A_NAME=(SELECT A.A_NAME FROM AUTHORS A,CATALOG C WHERE A.AUTHOR_ID=C.AUTHOR_ID GROUP BY C.AUTHOR ID HAVING COUNT(*)>=2); #1



SELECT A.A_NAME FROM AUTHORS A,CATALOG C,ORDER_DETAILS O WHERE
O.BOOK_ID=C.BOOK_ID AND A.AUTHOR_ID=C.AUTHOR_ID AND O.BOOK_ID=(SELECT
BOOK_ID FROM ORDER_DETAILS WHERE QTY=(SELECT MAX(QTY) FROM ORDER_DETAILS));
#2



UPDATE CATALOG SET PRICE=1.10*PRICE;

SELECT * FROM CATALOG;

← T→ '	■ BOOK_ID	TITLE	AUTHOR_ID	PUBLISHER_ID	CATEGORY_ID	P_YEAR	PRICE
☐ Ø Edit ♣ Copy Dele	te 111	BASICS	50	2	11	2003	220
☐ Ø Edit ♣ Copy □ Dele	te 222	PROG	40	5	44	1999	550
☐	te 333	MP	50	1	11	2009	990
☐ Ø Edit ♣ Copy ⊜ Dele	te 444	CIRCUITS	50	2	22	1997	330
☐ Ø Edit ♣ Copy ⊜ Dele	te 555	DATA	30	3	55	2005	660

PROGRAM 08: STUDENT ENROLLMENT DATABASE

Consider the following database of student enrollment in courses and books adopted for each course.

STUDENT (regno: String, name: String, major: String, bdate: date)

COURSE (course #: int, cname: String, dept: String)

ENROLL (regno: String, cname: String, sem: int, marks: int)

BOOK_ADOPTION (course #: int, sem: int, book-ISBN: int)

TEXT(book-ISBN:int, book-title:String, publisher:String, author:String)

- i. Create the above tables by properly specifying the primary keys and the foreign keys.
- ii. Enter at least five tuples for each relation.
- iii. Demonstrate how you add a new text book to the database and make this book be adopted by some department.
- iv. Produce a list of text books (include Course #, Book-ISBN, Book-title) in the alphabetical order for courses offered by the 'CS' department that use more than two books.
- v. List any department that has all its adopted books published by a specific publisher.

CREATE DATABASE STUDENTENROLL;

USE STUDENTENROLL;

CREATE TABLE STUDENT(

REGNO VARCHAR(20),

NAME VARCHAR2(20),

MAJOR VARCHAR2(5),

BDATE DATE,

```
PRIMARY KEY(REGNO)
 );
CREATE TABLE COURSE(
 COURSE_ID INT,
 CNAME VARCHAR(20),
 DEPT VARCHAR(4),
 PRIMARY KEY(COURSE_ID)
 );
CREATE TABLE TEXT(
  ISBN INT,
  BOOK_TITLE VARCHAR(50),
 PUBLISHER VARCHAR(20),
 AUTHOR VARCHAR(20),
 PRIMARY KEY(ISBN)
 );
CREATE TABLE ADOPTION(
 COURSE ID INT,
 SEM INT,
 ISBN INT,
 PRIMARY KEY(COURSE_ID, ISBN),
 FOREIGN KEY(COURSE_ID) REFERENCES COURSE(COURSE_ID),
 FOREIGN KEY(ISBN) REFERENCES TEXT(ISBN)
 );
CREATE TABLE ENROLL(
  REGNO VARCHAR(20),
```

```
COURSE ID INT,
 SEM INT,
 MARKS INT,
 PRIMARY KEY(REGNO, COURSE ID),
 FOREIGN KEY(REGNO) REFERENCES STUDENT(REGNO),
 FOREIGN KEY(COURSE ID) REFERENCES COURSE(COURSE ID)
 );
INSERT INTO STUDENT VALUES('REGNO', 'NAME', 'MAJOR', 'DATE');
INSERT INTO COURSE VALUES(COURSE ID, 'CNAME', 'DEPT');
INSERT INTO TEXT VALUES(ISBN, 'BOOK TITLE', 'PUBLISHER', 'AUTHOR');
INSERT INTO ADOPTION VALUES(COURSE_ID, SEM, ISBN);
INSERT INTO ENROLL VALUES('REGNO', COURSE_ID, SEM, MARKS);
SELECT * FROM STUDENT;
SELECT * FROM COURSE;
SELECT * FROM TEXT;
SELECT * FROM ADOPTION;
SELECT * FROM ENROLL;
INSERT INTO STUDENT (REGNO, NAME, MAJOR, DATE) VALUES
  ('1PE11CS002','SAM','SR','19930924'),
  ('1PE11CS003','CAMERON','SR','19931127'),
  ('1PE11CS004','DANNY','SR','19930413'),
  ('1PE11CS005','ETHER','JR','19940824');
                     ▼ REGNO
                                NAME
                                          MAJOR BDATE
&MAJO 0000-00-00
                                                1993-09-24
☐ Ø Edit ♣ Copy 	 Delete 1PE11CS002 SAM
                                           SR
☐ Ø Edit № Copy  Delete 1PE11CS003 CAMERON SR 1993-11-27
                                           SR
                                                 1993-04-13

  ☐ 
  Ø Edit 
  Copy 
  Delete 1PE11CS005 ETHER

                                          JR
                                                  1994-08-24
```

INSERT INTO COURSE (COURSE_ID, CNAME, DEPT) VALUES (111,'OS','CSE'), (112,'EC','CSE'), (113,'SS','ISE'), (114,'DBMS','CSE'), (115,'SIGNALS','ECE');

←T→	$\overline{}$	COURSE_ID	CNAME	DEPT
☐ Ø Edit Gopy Gopy	Delete	111	OS	CSE
☐ 🔗 Edit 👫 Copy	Delete	112	EC	CSE
☐ Ø Edit Gopy Gopy	Delete	113	SS	ISE
☐ 🔗 Edit 🛂 Copy	Delete	114	DBMS	CSE
☐ 🖉 Edit 🛂 Copy	Delete	115	SIGNALS	ECE

INSERT INTO TEXT (ISBN, BOOK_TITLE, PUBLISHER, AUTHOR) VALUES

(10, 'DATABASE SYSTEMS', 'PEARSON', 'SCHIELD'),

(900, 'OPERATING SYS', 'PEARSON', 'LELAND'),

(901, 'CIRCUITS', 'HALL INDIA', 'BOB'),

(902, 'SYSTEM SOFTWARE', 'PETERSON', 'JACOB'),

(903, 'SCHEDULING', 'PEARSON', 'PATIL'),

(904, 'DATABASE SYSTEMS', 'PEARSON', 'JACOB'),

(905, 'DATABASE MANAGER', 'PEARSON', 'BOB'),

(906, 'SIGNALS', 'HALL INDIA', 'SUMIT');

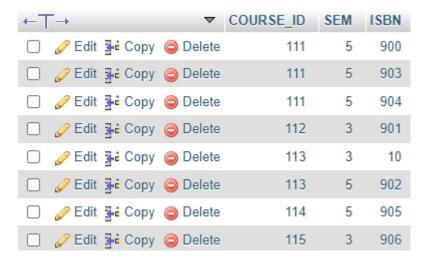
← + → ▼	ISBN	BOOK_TITLE	PUBLISHER	AUTHOR
☐ Ø Edit ¾ Copy Delete	10	DATABASE SYSTEMS	PEARSON	SCHIELD
☐ Ø Edit ♣ Copy Delete	900	OPERATING SYS	PEARSON	LELAND
☐	901	CIRCUITS	HALL INDIA	BOB
☐ Ø Edit ♣ Copy Delete	902	SYSTEM SOFTWARE	PETERSON	JACOB
☐ Ø Edit ♣ Copy Delete	903	SCHEDULING	PEARSON	PATIL
☐ Ø Edit ♣ Copy Delete	904	DATABASE SYSTEMS	PEARSON	JACOB
☐ Ø Edit ♣ Copy Delete	905	DATABASE MANAGER	PEARSON	BOB
☐ Ø Edit ♣ Copy Delete	906	SIGNALS	HALL INDIA	SUMIT

INSERT INTO ENROLL (REGNO, COURSE_ID, SEM, MARKS) VALUES ('1PE11CS002',114,5,100), ('1PE11CS003',113,5,100),

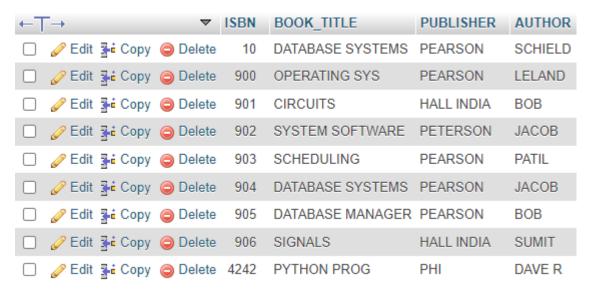
('1PE11CS004',111,5,100), ('1PE11CS005',112,3,100);

← ⊤→	~	REGNO	COURSE_ID	SEM	MARKS
☐ Ø Edit ¾ Copy	Delete	1PE11CS002	114	5	100
☐ 🖉 Edit 👫 Copy	Delete	1PE11CS003	113	5	100
☐ Ø Edit Gopy Gopy	Delete	1PE11CS004	111	5	100
☐ Ø Edit ≩ Copy	Delete	1PE11CS005	112	3	100

INSERT INTO ADOPTION (COURSE_ID, SEM, ISBN) VALUES (111,5,900), (111,5,903), (111,5,904), (112,3,901), (113,3,10), (114,5,905), (113,5,902), (115,3,906);



INSERT INTO TEXT VALUES('4242', 'PYTHON PROG', 'PHI', 'DAVE R');
INSERT INTO ADOPTION VALUES(40, 4, 4242); #1



SELECT C.CNAME, T.ISBN, T.BOOK_TITLE

FROM COURSE C, TEXT T

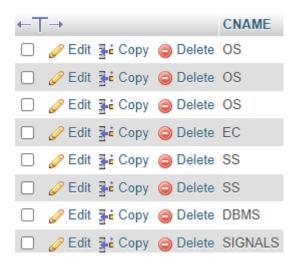
ORDER BY C.CNAME; #2

DBMS 901 CIRCUITS DBMS 906 SIGNALS DBMS 903 SCHEDULING	SYS
55110	SYS
DBMS 903 SCHEDULING	SYS
DBMS 900 OPERATING	ANAGER
DBMS 905 DATABASE M	
DBMS 902 SYSTEM SOF	TWARE
DBMS 10 DATABASE S	YSTEMS
DBMS 4242 PYTHON PRO	OG
DBMS 904 DATABASE S	YSTEMS
EC 903 SCHEDULING	3
EC 900 OPERATING	SYS
EC 905 DATABASE M	IANAGER
EC 902 SYSTEM SOF	TWARE
EC 10 DATABASE S	YSTEMS
EC 4242 PYTHON PRO	OG
EC 904 DATABASE S	YSTEMS
EC 901 CIRCUITS	
EC 906 SIGNALS	
OS 900 OPERATING	SYS
OS 905 DATABASE M	IANAGER
OS 902 SYSTEM SOF	TWARE
OS 4242 PYTHON PRO	OG
OS 10 DATABASE S	YSTEMS
OS 904 DATABASE S	YSTEMS

SELECT C.CNAME

FROM COURSE C, ADOPTION A, TEXT T

WHERE T.ISBN = A.ISBN AND C.COURSE_ID = A.COURSE_ID; #3



PROGRAM 9: MOVIE DATABASE

Consider the schema for Movie Database:

```
ACTOR(Act_id, Act_Name, Act_Gender)

DIRECTOR(Dir_id, Dir_Name, Dir_Phone)

MOVIES(Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id)

MOVIE_CAST(Act_id, Mov_id, Role)

RATING(Mov_id, Rev_Stars)

Write SQL queries to
```

- i. List the titles of all movies directed by 'Speed'.
- ii. Find the movie names where one or more actors acted in two or more movies.
- iii. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).
- iv. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.
- v. Update rating of all movies directed by 'Steven Spielberg' to 5

```
CREATE DATABASE MOVIE;
USE MOVIE;

CREATE TABLE ACTOR(
    ACT_ID INT,
    ACT_NAME VARCHAR(10),
    ACT_GENDER VARCHAR(10),
    PRIMARY KEY(ACT_ID)
    );
```

CREATE TABLE DIRECTOR(

```
DIR_ID INT,
 DIR NAME VARCHAR(10),
 DIR PHONE INT,
 PRIMARY KEY(DIR_ID)
 );
CREATE TABLE MOVIES(
  MOV_ID INT,
 MOV_TITLE VARCHAR(10),
 MOV_YEAR VARCHAR(10),
 MOV_LANG VARCHAR(10),
  DIR_ID INT, PRIMARY KEY(MOV_ID),
 FOREIGN KEY(DIR_ID) REFERENCES DIRECTOR(DIR_ID)
 );
CREATE TABLE MOVIE_CAST(
 ACT_ID INT, MOV_ID INT,
 ROLE_PLAYED VARCHAR(10),
 FOREIGN KEY(ACT ID) REFERENCES ACTOR(ACT ID),
 FOREIGN KEY(MOV ID) REFERENCES MOVIES(MOV ID)
 );
CREATE TABLE RATING(
 MOV_ID INT,REV_STARS FLOAT,
 FOREIGN KEY(MOV_ID) REFERENCES MOVIES(MOV_ID)
 );
INSERT INTO ACTOR VALUES(2,'Reeves','MALE');
INSERT INTO ACTOR VALUES(1,'Ice T','MALE');
```

INSERT INTO ACTOR VALUES(5,'Rai','FEMALE');
INSERT INTO ACTOR VALUES(3,'Khan','MALE');

INSERT INTO ACTOR VALUES(4, 'Pitt', 'MALE');

← T→	▼ ACT_ID	ACT_NAME	ACT_GENDER
☐ Ø Edit ¾ Copy €	Delete 1	Ice T	MALE
☐ 🖉 Edit 👫 Copy 🥥	Delete 2	Reeves	MALE
☐ Ø Edit ¾ Copy €	Delete 3	Khan	MALE
☐ Ø Edit ♣ Copy €	Delete 4	Pitt	MALE
☐ Ø Edit Gopy Gopy Gopy Gopy Gopy Gopy Gopy Gopy Gopy Gopy Gopy	Delete 5	Rai	FEMALE

SELECT * FROM ACTOR;

INSERT INTO DIRECTOR VALUES(30, 'Clint', 247382);

INSERT INTO DIRECTOR VALUES(20, 'Steven', 241829);

INSERT INTO DIRECTOR VALUES(10, 'Speed', 241342);

INSERT INTO DIRECTOR VALUES(50, 'Kevin', 241456);

INSERT INTO DIRECTOR VALUES(40, 'Downey', 241666);

←ͳ	→	$\overline{}$	DIR_ID	DIR_NAME	DIR_PHONE
	🥜 Edit 🛂 Copy	Delete	10	Speed	241342
	Ø Edit ₃ Copy	Delete	20	Steven	241829
	Ø Edit 👫 Copy	Delete	30	Clint	247382
	Ø Edit ₃ Copy	Delete	40	Downey	241666
	Ø Edit 3 € Copy	Delete	50	Kevin	241456

SELECT * FROM DIRECTOR;

INSERT INTO MOVIES VALUES(44, 'GUILTY', 2020, 'HINDI', 10);

INSERT INTO MOVIES VALUES(33,'1917',1999,'ENGLISH',10);

INSERT INTO MOVIES VALUES(11, 'NOTEBOOK', 2018, 'HINDI', 20);

INSERT INTO MOVIES VALUES(22, 'AVENGERS', 2007, 'ENGLISH', 20);

INSERT INTO MOVIES VALUES(55, 'WAR', 2018, 'HINDI', 50);



SELECT * FROM MOVIES;

INSERT INTO MOVIE_CAST VALUES(5,55,'VILLAIN');
INSERT INTO MOVIE_CAST VALUES(1,55,'MAIN LEAD');
INSERT INTO MOVIE_CAST VALUES(5,22,'SISTER');
INSERT INTO MOVIE_CAST VALUES(4,22,'MAIN LEAD');
INSERT INTO MOVIE_CAST VALUES(4,55,'BROTHER');
INSERT INTO MOVIE_CAST VALUES(2,44,'VILLAIN');

ACT_ID	MOV_ID	ROLE_PLAYED
5	55	VILLAIN
1	55	MAIN LEAD
5	22	SISTER
4	22	MAIN LEAD
4	55	BROTHER
2	44	VILLAIN

SELECT * FROM MOVIE CAST;

INSERT INTO RATING VALUES(11,4);

INSERT INTO RATING VALUES(55,3.5);

INSERT INTO RATING VALUES(22,2.5);

INSERT INTO RATING VALUES(11,3.5);

INSERT INTO RATING VALUES(11,4);

MOV_ID	REV_STARS
11	4
55	3.5
22	2.5
11	3.5
11	4

SELECT * FROM RATING;

SELECT D.DIR_NAME,M.MOV_TITLE FROM DIRECTOR D,MOVIES M WHERE D.DIR_ID=M.DIR_ID AND D.DIR_NAME='Speed'; #1

SELECT D.DIR_NAME,M.MOV_TITLE FROM DIRECTOR D,MOVIES M WHERE D.DIR_ID=M.DIR_ID AND D.D	DIR_NAME='Speed'
	☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh
☐ Show all Number of rows: 25 ▼ Filter rows: Search this table	
+ Options	
DIR_NAME MOV_TITLE	
Speed 1917	
Speed GUILTY	

SELECT M.MOV_TITLE FROM MOVIES M,MOVIE_CAST MC WHERE M.MOV_ID=MC.MOV_ID AND MC.ACT_ID IN(SELECT ACT_ID FROM MOVIE_CAST GROUP BY ACT_ID HAVING COUNT(*)>1) GROUP BY M.MOV_TITLE HAVING COUNT(*)>1; #2

Showing rows 0 - 1 (2 total, Query took 0.075	3 seconds.)	
SELECT M.MOV_TITLE FROM MOVIES M,MOVIE_ COUNT(*)>1) GROUP BY M.MOV_TITLE HAVING		ACT_ID IN(SELECT ACT_ID FROM MOVIE_CAST GROUP BY ACT_ID HAVING
		Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
☐ Show all Number of rows: 25 ▼	Filter rows: Search this table	
+ Options		
←⊤→ MOV_TITLE		
☐ Ø Edit ¾ Copy Delete AVENGERS		
☐ Ø Edit 👫 Copy 🔘 Delete WAR		

SELECT ACT_NAME FROM (MOVIES NATURAL JOIN MOVIE_CAST) NATURAL JOIN ACTOR WHERE MOV_YEAR>2015 OR MOV_YEAR<2000; #3

SELECT ACT_NAME FROM (MOVIES NATURAL JOIN MOVIE_CAST) NATURAL JOIN ACTOR WHERE MOV_YEAR>20	15 <u>OR</u> MOV_YEAR<2000
	Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh
☐ Show all Number of rows: 25 ▼ Filter rows: Search this table	
+ Options	
ACT_NAME	
Rai	
Ice T	
Pitt	
Reeves	

SELECT M.MOV_TITLE,MAX(R.REV_STARS) AS MAX_RATING FROM MOVIES M,RATING R WHERE M.MOV_ID=R.MOV_ID GROUP BY R.MOV_ID ORDER BY M.MOV_TITLE; #4



UPDATE RATING SET REV_STARS=5 WHERE MOV_ID IN (SELECT M.MOV_ID FROM MOVIES M,DIRECTOR D WHERE M.DIR_ID=D.DIR_ID AND D.DIR_NAME='Steven');

SELECT * FROM RATING; #5

MOV_ID	REV_STARS
11	5
55	3.5
22	5
11	5
11	5

PROGRAM 10: COLLEGE DATABASE

Consider the schema for College Database:

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

SEMSEC(SSID, Sem, Sec) CLASS(USN, SSID)

SUBJECT(Subcode, Title, Sem, Credits)

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

Write SQL queries to

- i. List all the student details studying in fourth semester 'C' section.
- ii. Compute the total number of male and female students in each semester and in each section.
- iii. Create a view of Test1 marks of student USN '22' in all subjects.
- iv. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.
- v. Categorize students based on the following criterion: If FinalIA = 17 to 20 then CAT = 'Outstanding' If FinalIA = 12 to 16 then CAT = 'Average' If FinalIA < 12 then CAT = 'Weak' Give these details only for 8th semester A, B, and C section students.

```
CREATE DATABASE COLLEGE;
USE COLLEGE;

CREATE TABLE STUDENT(

USN INT,S_NAME VARCHAR(10),

ADDRESS VARCHAR(20),

PHONE INT,

GENDER VARCHAR(10),

PRIMARY KEY(USN)

);
```

```
CREATE TABLE SEM_SEC(
 SSID INT,
 SEM INT,
 SEC VARCHAR(5),
 PRIMARY KEY(SSID)
 );
CREATE TABLE CLASS(
 USN INT,
 SSID INT,
 FOREIGN KEY(USN) REFERENCES STUDENT(USN),
 FOREIGN KEY(SSID) REFERENCES SEM_SEC(SSID)
 );
CREATE TABLE SUBJECTS(
 SUBCODE INT,
 TITLE VARCHAR(20),
 SEM INT,
 CREDITS INT,
 PRIMARY KEY(SUBCODE)
 );
CREATE TABLE MARKS(
 USN INT,
 SUBCODE INT,
 SSID INT,
 TEST1 INT,
 TEST2 INT,
```

```
TEST3 INT,

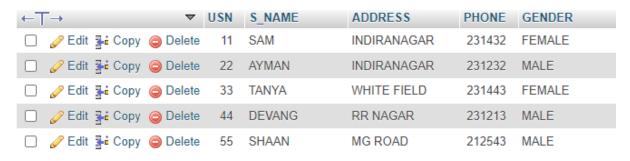
FOREIGN KEY(USN) REFERENCES STUDENT(USN),

FOREIGN KEY(SSID) REFERENCES SEM_SEC(SSID),

FOREIGN KEY(SUBCODE) REFERENCES SUBJECTS(SUBCODE)

);
```

INSERT INTO STUDENT VALUES(44,'DEVANG','RR NAGAR',231213,'MALE');
INSERT INTO STUDENT VALUES(22,'AYMAN','INDIRANAGAR',231232,'MALE');
INSERT INTO STUDENT VALUES(11,'SAM','INDIRANAGAR',231432,'FEMALE');
INSERT INTO STUDENT VALUES(55,'SHAAN','MG ROAD',212543,'MALE');
INSERT INTO STUDENT VALUES(33,'TANYA','WHITE FIELD',231443,'FEMALE');



SELECT * FROM STUDENT;

INSERT INTO SEM_SEC VALUES(2,6,'B');
INSERT INTO SEM_SEC VALUES(4,4,'C');
INSERT INTO SEM_SEC VALUES(5,4,'B');
INSERT INTO SEM_SEC VALUES(3,4,'A');
INSERT INTO SEM_SEC VALUES(1,2,'B');



SELECT * FROM SEM SEC;

INSERT INTO CLASS VALUES(33,5);

INSERT INTO CLASS VALUES(11,4);

INSERT INTO CLASS VALUES(55,2);

INSERT INTO CLASS VALUES(22,4);

INSERT INTO CLASS VALUES(44,4);

USN	SSID
33	5
11	4
55	2
22	4
44	4

SELECT * FROM CLASS;

INSERT INTO SUBJECTS VALUES(20, 'DBMS', 2, 4);

INSERT INTO SUBJECTS VALUES(10, 'MP',4,4);

INSERT INTO SUBJECTS VALUES(40, 'ADA', 1,4);

INSERT INTO SUBJECTS VALUES(30,'LD',5,3);

INSERT INTO SUBJECTS VALUES(50,'COA',3,3);



SELECT * FROM SUBJECTS;

INSERT INTO MARKS VALUES(33,10,5,19,19,20);

INSERT INTO MARKS VALUES(22,50,4,16,15,12);

INSERT INTO MARKS VALUES(55,30,2,19,19,19);

INSERT INTO MARKS VALUES(22,40,4,12,18,16);

INSERT INTO MARKS VALUES(44,10,4,10,12,11);

INSERT INTO MARKS VALUES(11,20,4,15,14,13);

USN	SUBCODE	SSID	TEST1	TEST2	TEST3
33	10	5	19	19	20
22	50	4	16	15	12
55	30	2	19	19	19
22	40	4	12	18	16
44	10	4	10	12	11
11	20	4	15	14	13

SELECT * FROM MARKS;

SELECT * FROM STUDENT S WHERE S.USN IN (SELECT C.USN FROM CLASS C,SEM_SEC S WHERE S.SSID=C.SSID AND S.SEM=4 AND S.SEC='C'); #1

← +	$\overline{}$	USN	S_NAME	ADDRESS	PHONE	GENDER
☐ Ø Edit ¾ Copy €	Delete	11	SAM	INDIRANAGAR	231432	FEMALE
☐ 🖉 Edit 👫 Copy 🧯	Delete	22	AYMAN	INDIRANAGAR	231232	MALE
☐ 🥜 Edit 👫 Copy 🧯	Delete	44	DEVANG	RR NAGAR	231213	MALE

SELECT S.GENDER,SS.SEM,SS.SEC,COUNT(*) FROM STUDENT S,SEM_SEC SS,CLASS C WHERE C.USN=S.USN AND C.SSID=SS.SSID GROUP BY SS.SSID; #2

GENDER	SEM	SEC	COUNT(*)
MALE	6	В	1
FEMALE	4	С	3
FEMALE	4	В	1

CREATE VIEW USN_22(USN,SUB,MARKS) AS SELECT M.USN,S.TITLE,M.TEST1 FROM MARKS M,SUBJECTS S WHERE M.SUBCODE=S.SUBCODE AND M.USN=22;

SELECT * FROM USN 22; #3

USN	SUB	MARKS
22	COA	16
22	ADA	12

ALTER TABLE MARKS ADD COLUMN FINAL ALL FLOAT;

UPDATE MARKS SET FINAL_ALL=((TEST1+TEST2+TEST3)-LEAST(TEST1,TEST2,TEST3))/2;

SELECT * FROM MARKS; #4

USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINAL_ALL
33	10	5	19	19	20	19.5
22	50	4	16	15	12	15.5
55	30	2	19	19	19	19
22	40	4	12	18	16	17
44	10	4	10	12	11	11.5
11	20	4	15	14	13	14.5

ALTER TABLE MARKS ADD COLUMN CATEGORY VARCHAR(20);

UPDATE MARKS SET CATEGORY=

CASE

WHEN FINAL_ALL>=17 AND FINAL_ALL<=20 THEN

'OUTSTANDING'

WHEN FINAL_ALL>=12 AND FINAL_ALL<17 THEN 'AVERAGE'
WHEN FINAL_ALL<12 THEN 'WEAK'

END;

SELECT * FROM MARKS; #5

USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINAL_ALL	CATEGORY
33	10	5	19	19	20	19.5	OUTSTANDING
22	50	4	16	15	12	15.5	AVERAGE
55	30	2	19	19	19	19	OUTSTANDING
22	40	4	12	18	16	17	OUTSTANDING
44	10	4	10	12	11	11.5	WEAK
11	20	4	15	14	13	14.5	AVERAGE