DBMS Lab Report 2

1BM19CS080- M VAMSHI KRISHNA

LAB PROGRAM 6:

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 ‘Hitchcock’.

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.

CODE:

```
create database movie;

use movie;

CREATE TABLE ACTOR (

ACT_ID INT,

ACT_NAME VARCHAR (20),

ACT_GENDER CHAR (1),
```

```
PRIMARY KEY (ACT_ID));

CREATE TABLE DIRECTOR (
DIR_ID INT,
DIR_NAME VARCHAR (20),
DIR_PHONE real,
PRIMARY KEY (DIR_ID));

CREATE TABLE MOVIES (
MOV_ID INT,
MOV_TITLE VARCHAR (25),
MOV_YEAR INT,
MOV_LANG VARCHAR (12),
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 VARCHAR(10),
PRIMARY KEY (ACT_ID, MOV_ID),
FOREIGN KEY (ACT_ID) REFERENCES ACTOR (ACT_ID),
FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));

CREATE TABLE RATING (
MOV_ID INT,
```

```
REV_STARS VARCHAR (25),
PRIMARY KEY (MOV_ID),
FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));
show tables;

INSERT INTO ACTOR VALUES (301,'ANUSHKA','F');
INSERT INTO ACTOR VALUES (302,'PRABHAS','M');
INSERT INTO ACTOR VALUES (303,'PUNITH','M');
INSERT INTO ACTOR VALUES (304,'JERMY','M');
SELECT * FROM ACTOR;

INSERT INTO DIRECTOR VALUES (60,'RAJAMOULI', 8751611001);
INSERT INTO DIRECTOR VALUES (61,'HITCHCOCK', 7766138911);
INSERT INTO DIRECTOR VALUES (62,'FARAN', 9986776531);
INSERT INTO DIRECTOR VALUES (63,'STEVEN SPIELBERG',
8989776530);
SELECT * FROM DIRECTOR;

INSERT INTO MOVIES VALUES (1001,'BAHUBALI-2', 2017, 'TELUGU',
60);
INSERT INTO MOVIES VALUES (1002,'BAHUBALI-1', 2015,'TELUGU',
60);
INSERT INTO MOVIES VALUES (1003,'AKASH', 2008,'KANNADA', 61);
INSERT INTO MOVIES VALUES (1004,'WAR HORSE', 2011, 'ENGLISH',
63);
SELECT * FROM MOVIES;

INSERT INTO MOVIE_CAST VALUES (301, 1002, 'HEROINE');
INSERT INTO MOVIE_CAST VALUES (301, 1001, 'HEROINE');
```

```
INSERT INTO MOVIE_CAST VALUES (303, 1003, 'HERO');
INSERT INTO MOVIE_CAST VALUES (303, 1002,'GUEST');
INSERT INTO MOVIE_CAST VALUES (304, 1004, 'HERO');
SELECT * FROM MOVIE_CAST;

INSERT INTO RATING VALUES (1001,'4');
INSERT INTO RATING VALUES (1002,'2');
INSERT INTO RATING VALUES (1003, '5');
INSERT INTO RATING VALUES (1004, '4');
SELECT * FROM RATING;

/*1. List the titles of all movies directed by‘Hitchcock’.*/
SELECT MOV_TITLE
FROM MOVIES
WHERE DIR_ID IN (SELECT DIR_ID
FROM DIRECTOR
WHERE DIR_NAME = 'HITCHCOCK');

/*2. Find the movie names where one or more actors acted in two or more
movies.*/
SELECT MOV_TITLE
FROM MOVIES M, MOVIE_CAST MV
WHERE M.MOV_ID=MV.MOV_ID AND ACT_ID IN (SELECT ACT_ID
FROM MOVIE_CAST GROUP BY ACT_ID HAVING
COUNT(ACT_ID)>1)
GROUP BY MOV_TITLE HAVING COUNT(MOV_TITLE)>1;

/*3. List all actors who acted in a movie before 2000 and also in a movie after
```

2015 (use JOIN operation).*/

```
SELECT ACT_NAME, MOV_TITLE, MOV_YEAR
FROM ACTOR A JOIN
MOVIE_CAST C
ON A.ACT_ID=C.ACT_ID
JOIN MOVIES M
ON C.MOV_ID=M.MOV_ID
WHERE M.MOV_YEAR NOT BETWEEN 2000 AND 2015;
```

/*4. 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. */

```
SELECT MOV_TITLE, MAX(REV_STARS) FROM MOVIES
INNER JOIN RATING USING (MOV_ID) GROUP
BY MOV_TITLE
HAVING MAX(REV_STARS)>0
ORDER BY MOV_TITLE;
```

/*5. Update rating of all movies directed by ‘Steven Spielberg’ to 5*/

```
UPDATE RATING
SET REV_STARS='5'
WHERE MOV_ID = (SELECT MOV_ID FROM MOVIES
WHERE DIR_ID = (SELECT DIR_ID FROM DIRECTOR
WHERE DIR_NAME ='STEVEN
SPIELBERG'));
SELECT * FROM RATING;
```

Output: Tables:

SCHEMAS

Filter objects

- airline
- banking
- book_dealer
- insurance
- movie
  - Tables
  - Views
  - Stored Procedures
  - Functions

Info | Tables | Columns | Indexes | Triggers | Views | Stored Procedures | Functions | Grants | Events

Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length	Max Data Length	Index Length	Data Free	Auto Incre...	Create Time	Update Time
actor	InnoDB	10	Dynamic	4	4096	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	0	2021-06-30 10:37:21	2021-06-30 10:37:21
director	InnoDB	10	Dynamic	4	4096	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	0	2021-06-30 10:37:21	2021-06-30 10:37:21
movie_cast	InnoDB	10	Dynamic	5	3276	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	0	2021-06-30 10:37:21	2021-06-30 10:37:21
movies	InnoDB	10	Dynamic	4	4096	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	0	2021-06-30 10:37:21	2021-06-30 10:37:21
rating	InnoDB	10	Dynamic	4	4096	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	0	2021-06-30 10:37:21	2021-06-30 10:37:21

Columns:

SCHEMAS

Filter objects

- airline
- banking
- book_dealer
- insurance
- movie
  - Tables
  - Views
  - Stored Procedures
  - Functions
- order_processing
- s_tudent
- student_enrol
- supplier
- sys

Info | Tables | Columns | Indexes | Triggers | Views | Stored Procedures | Functions | Grants | Events

Table	Column	Type	Default Value	Nullable	Character Set	Collation	Privileges	Extra	Comments
actor	ACT_ID	int		NO			select,insert,update,references		
actor	ACT_NAME	varchar(20)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references		
actor	ACT_GENDER	char(1)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references		
director	DIR_ID	int		NO			select,insert,update,references		
director	DIR_NAME	varchar(20)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references		
director	DIR_PHONE	double		YES			select,insert,update,references		
movie_cast	ACT_ID	int		NO			select,insert,update,references		
movie_cast	MOV_ID	int		NO			select,insert,update,references		
movie_cast	ROLE	varchar(10)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references		
movies	MOV_ID	int		NO			select,insert,update,references		
movies	MOV_TITLE	varchar(25)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references		
movies	MOV_YEAR	int		YES			select,insert,update,references		
movies	MOV_LANG	varchar(12)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references		
movies	DIR_ID	int		YES			select,insert,update,references		
rating	MOV_ID	int		NO			select,insert,update,references		
rating	REV_STARS	varchar(25)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references		

Result 1:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Tables_in_movie
actor
director
movie_cast
movies
rating

Administration | Schemas

Information

No object selected

Result 1 | ACTOR 2 | DIRECTOR 3 | MOVIES 4 | MOVIE_CAST 5 | RATING 6

Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
136	10:37:21	SELECT * FROM MOVIE_CAST LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
137	10:37:21	INSERT INTO RATING VALUES (1001,'4')	1 row(s) affected	0.000 sec
138	10:37:21	INSERT INTO RATING VALUES (1002,'2')	1 row(s) affected	0.000 sec
139	10:37:21	INSERT INTO RATING VALUES (1003, '5')	1 row(s) affected	0.000 sec
140	10:37:21	INSERT INTO RATING VALUES (1004, '4')	1 row(s) affected	0.016 sec
141	10:37:21	SELECT * FROM RATING LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

Object Info | Session

Actor 2:

Result Grid | Filter Rows: | Edi

ACT_ID	ACT_NAME	ACT_GENDER
301	ANUSHKA	F
302	PRABHAS	M
303	PUNITH	M
304	JERMY	M
NULL	NULL	NULL

Director 3:

Result Grid | Filter Rows: | Edit:

DIR_ID	DIR_NAME	DIR_PHONE
60	RAJAMOULI	8751611001
61	HITCHCOCK	7766138911
62	FARAN	9986776531
63	STEVEN SPIELBERG	8989776530
NULL	NULL	NULL

Movies 4:

MOV_ID	MOV_TITLE	MOV_YEAR	MOV_LANG	DIR_ID
1001	BAHUBALI-2	2017	TELUGU	60
1002	BAHUBALI-1	2015	TELUGU	60
1003	AKASH	2008	KANNADA	61
1004	WAR HORSE	2011	ENGLISH	63
NULL	NULL	NULL	NULL	NULL

Movie_cast 5:

Result Grid | Filter Rows:

ACT_ID	MOV_ID	ROLE
301	1001	HEROINE
301	1002	HEROINE
303	1002	GUEST
303	1003	HERO
304	1004	HERO
NULL	NULL	NULL

Rating 6:

MOV_ID	REV_STARS
1001	4
1002	2
1003	5
1004	4
NULL	NULL

/*1. List the titles of all movies directed by‘Hitchcock’.*/

SELECT MOV_TITLE

FROM MOVIES

WHERE DIR_ID IN (SELECT DIR_ID

FROM DIRECTOR

WHERE DIR_NAME = 'HITCHCOCK');

	MOV_TITLE
▶	AKASH

/*2. Find the movie names where one or more actors acted in two or more movies.*/

SELECT MOV_TITLE

FROM MOVIES M, MOVIE_CAST MV

WHERE M.MOV_ID=MV.MOV_ID AND ACT_ID IN (SELECT ACT_ID

FROM MOVIE_CAST GROUP BY ACT_ID HAVING

COUNT(ACT_ID)>1)

GROUP BY MOV_TITLE HAVING COUNT(MOV_TITLE)>1;

Result Grid | Filter Rows:

	MOV_TITLE
▶	BAHUBALI-1

/*3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).*/

SELECT ACT_NAME, MOV_TITLE, MOV_YEAR

FROM ACTOR A JOIN

MOVIE_CAST C

ON A.ACT_ID=C.ACT_ID

JOIN MOVIES M

ON C.MOV_ID=M.MOV_ID

WHERE M.MOV_YEAR NOT BETWEEN 2000 AND 2015;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	ACT_NAME	MOV_TITLE	MOV_YEAR
▶	ANUSHKA	BAHUBALI-2	2017

Administration Schemas

Information

/*4. 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. */

```
SELECT MOV_TITLE, MAX(REV_STARS) FROM MOVIES
INNER JOIN RATING USING (MOV_ID) GROUP
BY MOV_TITLE
HAVING MAX(REV_STARS)>0
ORDER BY MOV_TITLE;
```

Result Grid | Filter Rows:

MOV_TITLE	MAX(REV_STARS)
AKASH	5
BAHUBALI-1	2
BAHUBALI-2	4
WAR HORSE	4

/*5. Update rating of all movies directed by ‘Steven Spielberg’ to 5*/

```
UPDATE RATING
SET REV_STARS='5'
WHERE MOV_ID = (SELECT MOV_ID FROM MOVIES
WHERE DIR_ID = (SELECT DIR_ID FROM DIRECTOR
WHERE DIR_NAME ='STEVEN
SPIELBERG'));
SELECT * FROM RATING;
```

Result Grid | Filter Rows:

MOV_ID	REV_STARS
1001	4
1002	2
1003	5
1004	4
NULL	NULL

PROGRAM 7:

AIRLINE FLIGHT DATABASE

Consider the following database that keeps track of airline flight information:

FLIGHTS (flno: integer, from: string, to: string, distance: integer, departs: time, arrives: time, price: integer)

AIRCRAFT (aid: integer, aname: string, cruisingrange: integer)

CERTIFIED (eid: integer, aid: integer)

EMPLOYEE (eid: integer, ename: string, salary: integer)

Note that the Employees relation describes pilots and other kinds of employees as well; Every pilot is certified

for some aircraft, and only pilots are certified to fly.

Write each of the following queries in SQL.

i. Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.80,000.

ii. For each pilot who is certified for more than three aircrafts, find the eid and the maximum cruising range of

the aircraft for which she or he is certified.

iii. Find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru to

Frankfurt.

iv. For all aircraft with cruising range over 1000 Kms, find the name of the aircraft and the average salary of

all pilots certified for this aircraft.

v. Find the names of pilots certified for some Boeing aircraft.

vi. Find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi.

vii. A customer wants to travel from Madison to New York with no more than two changes of flight. List the

choice of departure times from Madison if the customer wants to arrive in New York by 6 p.m.

viii. Print the name and salary of every non-pilot whose salary is more than the average salary for pilots.

Code: create database airline;

```
use airline;
CREATE TABLE flights(
flno Int,
`from` Varchar(20),
`to` Varchar(20),
distance INT,
departs time,
arrives time,
price Int,
PRIMARY KEY(flno) );
 CREATE TABLE aircraft(
 aid INT,
 aname VARCHAR(20),
 cruisingrange INT,
PRIMARY KEY (aid) );

CREATE TABLE employees(
 eid INT,
 ename Varchar(20),
 salary INT,
 PRIMARY KEY (eid) );

CREATE TABLE certified(
eid INT,
aid INT,
PRIMARY KEY (eid,aid),
```

```
FOREIGN KEY (eid) REFERENCES employees (eid),
FOREIGN KEY (aid) REFERENCES aircraft (aid) );

show tables;

INSERT INTO flights (flno,`from`,`to`,distance,departs,arrives,price) VALUES
(1,'Bangalore','Chennai',360,'08:45','10:00',10000),
(2,'Bangalore','Delhi',1700,'12:15','15:00',37000),
(3,'Bangalore','Kolkata',1500,'15:15','05:25',30000),
(4,'Mumbai','Delhi',1200,'10:30','12:30',28000),
(5,'Bangalore','New york',14000,'05:45','02:30',90000),
(6,'Delhi','Chicago',12000,'10:00','05:45',95000),
(7,'Bangalore','Frankfurt',15000,'12:00','06:30',98000),
(8,'Madison','New york',1500,'10:15','14:25',30000);
SELECT * FROM flights;

INSERT INTO aircraft (aid,aname,cruisingrange) values
	(1,'Airbus 380',1000),
	(2,'Boeing 737',4000),
	(3,'Lockheed',5500),
	(4,'Airbus A220',9500),
	(5,'Boeing 747',800),
	(6,'Douglas DC3',900);
SELECT * FROM aircraft;

INSERT INTO employees (eid,ename,salary) VALUES
(1,'Zoya',95000),
```

```
(2,'Akshay',65000),
(3,'Niveditha',70000),
(4,'Safan',45000),
(5,'Peter',95000),
(6,'Nayan',100000),
(7,'Ajay',50000);
SELECT * FROM employees;

INSERT INTO certified (eid,aid) VALUES
(1,1),
(1,3),
(1,4),
(5,4),
(5,3),
(1,2),
 (2,6),
(2,5),
(4,5),
(6,4),
(6,3),
(3,6),
(3,2);
SELECT * FROM certified;
```

#i. Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.80,000.

```
SELECT DISTINCT A.aname
```

```
FROM Aircraft A
WHERE A.Aid IN (SELECT C.aid
FROM Certified C, Employees E
WHERE C.eid = E.eid AND
NOT EXISTS ( SELECT *
FROM Employees E1
WHERE E1.eid = E.eid AND E1.salary < 80000 ));
```

#ii. For each pilot who is certified for more than three aircrafts, find the eid and the maximum cruising range of the aircraft for which she or he is certified.

```
SELECT C.eid, MAX(A.cruisingrange)
FROM Certified C, Aircraft A
WHERE C.aid = A.aid
GROUP BY C.eid
HAVING COUNT(*) > 3;
```

#iii. Find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru toFrankfurt.

```
SELECT DISTINCT e.ename
FROM employees e
WHERE e.salary<
(SELECT MIN(f.price)
FROM flights f
WHERE f.from='Bangalore' AND f.to='Frankfurt');
```

#iv. For all aircraft with cruising range over 1000 Kms, find the name of the aircraft and the average salary of all pilots certified for this aircraft.

```
SELECT a.aid,a.aname,AVG(e.salary)
```

```
FROM aircraft a,certified c,employees e
WHERE a.aid=c.aid
AND c.eid=e.eid
AND a.cruisingrange>1000
GROUP BY a.aid,a.aname;
```

#v. Find the names of pilots certified for some Boeing aircraft.

```
SELECT distinct e.ename
FROM employees e,aircraft a,certified c
WHERE e.eid=c.eid AND c.aid=a.aid AND a.aname like 'Boeing%';
```

#vi. Find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi.

```
SELECT a.aid
FROM aircraft a
WHERE a.cruisingrange>
(SELECT MIN(f.distance)
FROM flights f
WHERE f.from='Bangalore' AND f.to='Delhi');
```

#vii. A customer wants to travel from Madison to New York with no more than two changes of flight. List the choice of departure times from Madison if the customer wants to arrive in New York by 6 p.m.

```
SELECT F.departs
FROM Flights F WHERE F.flno IN (  SELECT F0.flno
FROM Flights F0
WHERE F0.from = 'Madison' AND F0.to = 'New york' AND F0.arrives < '18:00'
);
```

#viii. Print the name and salary of every non-pilot whose salary is more than the average salary for pilots.0

```
SELECT E.ename, E.salary
FROM Employees E
WHERE E.eid NOT IN ( SELECT DISTINCT C.eid
FROM Certified C )
AND E.salary > ( SELECT AVG (E1.salary)
FROM Employees E1
WHERE E1.eid IN
( SELECT DISTINCT C1.eid
FROM Certified C1 ) );
```

Output:

Tables:

Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length	Max Data Length	Index Length	Data Free	Auto Incre...	Create Time	Update Time
aircraft	InnoDB	10	Dynamic	6	2730	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	0	2021-06-30 10:07:42	2021-06-30 10:07:42
certified	InnoDB	10	Dynamic	13	1260	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	0	2021-06-30 10:07:42	2021-06-30 10:07:42
employees	InnoDB	10	Dynamic	7	2340	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	0	2021-06-30 10:07:42	2021-06-30 10:07:42
flights	InnoDB	10	Dynamic	8	2048	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	0	2021-06-30 10:07:42	2021-06-30 10:07:42

Columns:

Info | Tables | Columns | Indexes | Triggers | Views | Stored Procedures | Functions | Grants | Events

Table	Column	Type	Default Value	Nullable	Character Set	Collation	Privileges	Extra	Comments
aircraft	aid	int		NO			select,insert,update,references		
aircraft	aname	varchar(20)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references		
aircraft	cruisingrange	int		YES			select,insert,update,references		
certified	eid	int		NO			select,insert,update,references		
certified	aid	int		NO			select,insert,update,references		
employees	eid	int		NO			select,insert,update,references		
employees	ename	varchar(20)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references		
employees	salary	int		YES			select,insert,update,references		
flights	flno	int		NO			select,insert,update,references		
flights	from	varchar(20)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references		
flights	to	varchar(20)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references		
flights	distance	int		YES			select,insert,update,references		
flights	departs	time		YES			select,insert,update,references		
flights	arrives	time		YES			select,insert,update,references		
flights	price	int		YES			select,insert,update,references		

```
SELECT * FROM airline.aircraft;
```

Result Grid | Filter Rows:

aid	aname	cruisingrange
1	Airbus 380	1000
2	Boeing 737	4000
3	Lockheed	5500
4	Airbus A220	9500
5	Boeing 747	800
6	Douglas DC3	900
NULL	NULL	NULL

SELECT * FROM airline.certified;

Result Grid

eid	aid
1	1
1	2
3	2
1	3
5	3
6	3
1	4
5	4
6	4
2	5
4	5
2	6
3	6
NULL	NULL

SELECT * FROM airline.employees;

Result Grid | Filter Rows

eid	ename	salary
1	Zoya	95000
2	Akshay	65000
3	Niveditha	70000
4	Safan	45000
5	Peter	95000
6	Nayan	100000
7	Ajay	50000
NULL	NULL	NULL

SELECT * FROM airline.flights;

Result Grid | Filter Rows: | Edit: | Expo

flno	from	to	distance	departs	arrives	price
1	Bangalore	Chennai	360	08:45:00	10:00:00	10000
2	Bangalore	Delhi	1700	12:15:00	15:00:00	37000
3	Bangalore	Kolkata	1500	15:15:00	05:25:00	30000
4	Mumbai	Delhi	1200	10:30:00	12:30:00	28000
5	Bangalore	New york	14000	05:45:00	02:30:00	90000
6	Delhi	Chicago	12000	10:00:00	05:45:00	95000
7	Bangalore	Frankfurt	15000	12:00:00	06:30:00	98000
8	Madison	New york	1500	10:15:00	14:25:00	30000
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Queries:

/*i. Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.80,000.*/

```
SELECT DISTINCT A.aname

FROM Aircraft A

WHERE A.Aid IN (SELECT C.aid

FROM Certified C, Employees E

WHERE C.eid = E.eid AND

NOT EXISTS ( SELECT *

FROM Employees E1

WHERE E1.eid = E.eid AND E1.salary < 80000 ));
```

aname
Airbus 380
Boeing 737
Lockheed
Airbus A220

/*ii. For each pilot who is certified for more than three aircrafts, find the eid and the maximum cruising range of the aircraft for which she or he is certified.*/

```
SELECT C.eid, MAX(A.cruisingrange)

FROM Certified C, Aircraft A

WHERE C.aid = A.aid

GROUP BY C.eid

HAVING COUNT(*) > 3;
```

eid	MAX(A.cruisingrange)
1	9500

/*iii. Find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru toFrankfurt.*/

```
SELECT DISTINCT e.ename

FROM employees e

WHERE e.salary<

(SELECT MIN(f.price)

FROM flights f

WHERE f.from='Bangalore' AND f.to='Frankfurt');
```

ename
Zoya
Akshay
Niveditha
Safan
Peter
Ajay

/*iv. For all aircraft with cruising range over 1000 Kms, find the name of the aircraft and the average salary of all pilots certified for this aircraft. */

```
SELECT a.aid,a.aname,AVG(e.salary)

FROM aircraft a,certified c,employees e

WHERE a.aid=c.aid

AND c.eid=e.eid

AND a.cruisingrange>1000

GROUP BY a.aid,a.aname;
```

aid	aname	AVG(e.salary)
2	Boeing 737	82500.0000
3	Lockheed	96666.6667
4	Airbus A220	96666.6667

/*v. Find the names of pilots certified for some Boeing aircraft. */

```
SELECT distinct e.ename

FROM employees e,aircraft a,certified c
```

WHERE e.eid=c.eid AND c.aid=a.aid AND a.aname like 'Boeing%';

Result Grid

ename
Zoya
Niveditha
Akshay
Safan

/*vi. Find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi. */

SELECT a.aid

FROM aircraft a

WHERE a.cruisingrange>

(SELECT MIN(f.distance)

FROM flights f

WHERE f.from='Bangalore' AND f.to='Delhi');

Result Grid

aid
2
3
4
NULL

/*vii. A customer wants to travel from Madison to New York with no more than two changes of flight. List the choice of departure times from Madison if the customer wants to arrive in New York by 6 p.m. */

SELECT F.departs

FROM Flights F WHERE F.flno IN (SELECT F0.flno

FROM Flights F0

WHERE F0.from = 'Madison' AND F0.to = 'New york' AND F0.arrives < '18:00');

departs
10:15:00

/*viii. Print the name and salary of every non-pilot whose salary is more than the average salary for pilots.0*/

```
SELECT E.ename, E.salary

FROM Employees E

WHERE E.eid NOT IN ( SELECT DISTINCT C.eid

FROM Certified C )

AND E.salary > ( SELECT AVG (E1.salary)

FROM Employees E1

WHERE E1.eid IN

( SELECT DISTINCT C1.eid

FROM Certified C1 ) );
```

ename	salary

LAB PROGRAM 8:

STUDENT FACLUTY DATABASE:

Consider the following database for student enrolment for course:

STUDENT (snum: integer, sname: string, major: string, level: string, age: integer)

CLASS (name: string, meets at: time, room: string, fid: integer)

ENROLLED (snum: integer, cname: string)

FACULTY (fid: integer, fname: string, deptid: integer)

The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair

such that the student is enrolled in the class. Level is a two character code with 4 different values (example:

Junior: JR etc)

Write the following queries in SQL. No duplicates should be printed in any of the answers.

i. Find the names of all Juniors (level = JR) who are enrolled in a class taught by

ii. Find the names of all classes that either meet in room R128 or have five or more Students enrolled.

iii. Find the names of all students who are enrolled in two classes that meet at the same time.

iv. Find the names of faculty members who teach in every room in which some class is taught.

v. Find the names of faculty members for whom the combined enrolment of the courses that they teach is less

than five.

vi. Find the names of students who are not enrolled in any class.

vii. For each age value that appears in Students, find the level value that appears most often. For example, if

there are more FR level students aged 18 than SR, JR, or SO students aged 18, you should print the pair (18,

FR).

CODE:

```
create database s_tudent;

use s_tudent;
```

```
CREATE TABLE student(
snum INT,
sname VARCHAR(10),
major VARCHAR(2),
lvl VARCHAR(2),
age INT,
primary key(snum));

CREATE TABLE faculty(
fid INT,
fname VARCHAR(20),
deptid INT,
PRIMARY KEY(fid));

CREATE TABLE class(
cname VARCHAR(20),
metts_at TIMESTAMP,
room VARCHAR(10),
fid INT,
PRIMARY KEY(cname),
FOREIGN KEY(fid) REFERENCES faculty(fid));

CREATE TABLE enrolled(
snum INT,
cname VARCHAR(20),
```

```
PRIMARY KEY(snum,cname),
FOREIGN KEY(snum) REFERENCES student(snum),
FOREIGN KEY(cname) REFERENCES class(cname));

INSERT INTO STUDENT
VALUES(1,'jhon','CS','Sr',19),(2,'Smith','CS','Jr',20),(3,'Jacob','CV','Sr',
20),(4,'Tom','CS','Jr',20),(5,'Rahul','CS','Jr',20),(6,'Rita','CS','Sr',21);

INSERT INTO FACULTY
VALUES
(11,'Harish',1000),(12,'MV',1000),(13,'Mira',1001),(14,'Shiva',1002),(15
,'Nupur',1000);

insert into class
values ('class1', '12/11/15 10:15:16', 'R1', 14),('class10', '12/11/15
10:15:16', 'R128', 14),('class2', '12/11/15 10:15:20', 'R2', 12),('class3',
'12/11/15 10:15:25', 'R3', 11),('class4', '12/11/15 20:15:20', 'R4',
14),('class5', '12/11/15 20:15:20', 'R3', 15),('class6', '12/11/15 13:20:20',
'R2', 14),('class7', '12/11/15 10:10:10', 'R3', 14);

insert into enrolled
values (1, 'class1'),(2, 'class1'),(3, 'class3'),(4, 'class3'),(5, 'class4'),(1,
'class5'),(2, 'class5'),(3, 'class5'),(4, 'class5'),(5, 'class5');

select distinct s.sname
from student s, class c, enrolled e, faculty f
where s.snum = e.snum and e.cname = c.cname and c.fid = f.fid and
```

```
f.fname ='Harish' and s.lvl ='Jr';

select C.cname
from Class C
where C.room = 'R128'
or C.cname in (select E.cname
            FROM Enrolled E
            GROUP BY E.cname
            HAVING COUNT(E.snum) >= 5);

select distinct S.sname
from Student S
where S.snum in (select E1.snum
                from Enrolled E1, Enrolled E2, Class C1, Class C2
                where E1.snum = E2.snum AND E1.cname <> E2.cname
                AND E1.cname = C1.cname
                AND E2.cname = C2.cname AND C1.metts_at =
C2.metts_at);

select distinct F.fname
from Faculty F,Class C
where F.fid=C.fid
group by C.fid
having count(distinct room) = 5;
```

```
select distinct F.fname
from Faculty F
where 5 > (select COUNT(E.snum)
from Class C, Enrolled E
where C.cname = E.cname
and C.fid = F.fid);

select distinct S.sname
from Student S
where S.snum NOT IN (select E.snum
from Enrolled E );

select S.age, S.lvl
from Student S
group by S.age, S.lvl
having S.lvl IN (select S1.lvl from Student S1
    where S1.age = S.age
group by S1.lvl, S1.age
having COUNT(S1.snum) >= ALL (select COUNT(S2.snum)
from Student S2
where s1.age = S2.age
group by S2.lvl, S2.age));
```

/*i. Find the names of all Juniors (level = JR) who are enrolled in a class taught by Harish.*/

```
select distinct s.sname
from student s, class c, enrolled e, faculty f
where s.snum = e.snum and e.cname = c.cname and c.fid = f.fid and
f.fname ='Harish' and s.lvl ='Jr';
```

/*ii. Find the names of all classes that either meet in room R128 or have five or more Students enrolled.*/

```
select C.cname
from Class C
where C.room = 'R128'
or C.cname in (select E.cname
        FROM Enrolled E
        GROUP BY E.cname
        HAVING COUNT(E.snum) >= 5);
```

/*iii. Find the names of all students who are enrolled in two classes that meet at the same time.*/

```
select distinct S.sname
from Student S
where S.snum in (select E1.snum
            from Enrolled E1, Enrolled E2, Class C1, Class C2
            where E1.snum = E2.snum AND E1.cname <> E2.cname
            AND E1.cname = C1.cname
            AND E2.cname = C2.cname AND C1.metts_at =
C2.metts_at);
```

/*iv. Find the names of faculty members who teach in every room in which some class is taught.*/

```
select distinct F.fname
from Faculty F,Class C
where F.fid=C.fid
group by C.fid
having count(distinct room) = 5;
```

/*v. Find the names of faculty members for whom the combined enrollment of the courses that they teach is less than five.*/

```
select distinct F.fname
from Faculty F
where 5 > (select COUNT(E.snum)
from Class C, Enrolled E
where C.cname = E.cname
and C.fid = F.fid);
```

/*vi. Find the names of students who are not enrolled in any class. */

```
select distinct S.sname
from Student S
where S.snum NOT IN (select E.snum
from Enrolled E );
```

/*vii. For each age value that appears in Students, find the level value that appears most often. For example, if there are more FR level students aged 18 than

SR, JR, or SO students aged 18, you should print the pair (18, FR).*/

```
select S.age, S.lvl
from Student S
group by S.age, S.lvl
having S.lvl IN (select S1.lvl from Student S1
```

```
    where S1.age = S.age
group by S1.lvl, S1.age
having COUNT(S1.snum) >= ALL (select COUNT(S2.snum)
from Student S2
where s1.age = S2.age
group by S2.lvl, S2.age));
```

output:

Tables:

Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length	Max Data Length	Index Length	Data Free	Auto Incre...	Create Time	Update Time
class	InnoDB	10	Dynamic	8	2048	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	0	2021-06-30 09:45:45	2021-06-30 09:45:45
enrolled	InnoDB	10	Dynamic	10	1638	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	0	2021-06-30 09:45:45	2021-06-30 09:45:45
faculty	InnoDB	10	Dynamic	5	3276	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	0	2021-06-30 09:45:45	2021-06-30 09:45:45
student	InnoDB	10	Dynamic	6	2730	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	0	2021-06-30 09:45:44	2021-06-30 09:45:45

Columns:

Table	Column	Type	Default Value	Nullable	Character Set	Collation	Privileges	Extra
class	cname	varchar(20)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references	
class	metts_at	timestamp		YES			select,insert,update,references	
class	room	varchar(10)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references	
class	fid	int		YES			select,insert,update,references	
enrolled	snum	int		NO			select,insert,update,references	
enrolled	cname	varchar(20)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references	
faculty	fid	int		NO			select,insert,update,references	
faculty	fname	varchar(20)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references	
faculty	deptid	int		YES			select,insert,update,references	
student	snum	int		NO			select,insert,update,references	
student	sname	varchar(10)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references	
student	major	varchar(2)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references	
student	lvl	varchar(2)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references	
student	age	int		YES			select,insert,update,references	

Indexes :

Table	Name	Unique	Index...	Index Comment	Column	Seq in Index	Packed	Collat...	Cardi...	Sub p...	NULL	Comment
class	PRIMARY	Yes	BTREE		cname	1		A	3			
class	fid	No	BTREE		fid	1		A	4		YES	
enrolled	PRIMARY	Yes	BTREE		snum	1		A	5			
enrolled	PRIMARY	Yes	BTREE		cname	2		A	10			
enrolled	cname	No	BTREE		cname	1		A	4			
faculty	PRIMARY	Yes	BTREE		fid	1		A	5			
student	PRIMARY	Yes	BTREE		snum	1		A	6			

SELECT * FROM s_tudent.class;

Result Grid | Filter Rows:

cname	metts_at	room	fid
class1	2012-11-15 10:15:16	R1	14
class10	2012-11-15 10:15:16	R128	14
class2	2012-11-15 10:15:20	R2	12
class3	2012-11-15 10:15:25	R3	11
class4	2012-11-15 20:15:20	R4	14
class5	2012-11-15 20:15:20	R3	15
class6	2012-11-15 13:20:20	R2	14
class7	2012-11-15 10:10:10	R3	14
NULL	NULL	NULL	NULL

SELECT * FROM s_tudent.enrolled;

Result Grid | Filter Rows:

snum	cname
1	class1
2	class1
3	class3
4	class3
5	class4
1	class5
2	class5
3	class5
4	class5
5	class5
NULL	NULL

enrolled 1

SELECT * FROM s_tudent.faculty;

fid	fname	deptid
11	Harish	1000
12	MV	1000
13	Mira	1001
14	Shiva	1002
15	Nupur	1000
NULL	NULL	NULL

SELECT * FROM s_tudent.student;

Result Grid | Filter Rows:

snum	sname	major	lvl	age
1	jhon	CS	Sr	19
2	Smith	CS	Jr	20
3	Jacob	CV	Sr	20
4	Tom	CS	Jr	20
5	Rahul	CS	Jr	20
6	Rita	CS	Sr	21
NULL	NULL	NULL	NULL	NULL

i. Find the names of all Juniors (level = JR) who are enrolled in a class taught by Harish.

select distinct s.sname

from student s, class c, enrolled e, faculty f

where s.snum = e.snum and e.cname = c.cname and c.fid = f.fid and

f.fname ='Harish' and s.lvl ='Jr';

Result Grid | Filter Rows:

sname
Tom

ii. Find the names of all classes that either meet in room R128 or have five or more Students enrolled.

```
select C.cname
from Class C
where C.room = 'R128'
or C.cname in (select E.cname
               FROM Enrolled E
               GROUP BY E.cname
               HAVING COUNT(E.snum) >= 5);
```

Result Grid | Filter Rows:

cname
class10
class5
NULL

iii. Find the names of all students who are enrolled in two classes that meet at the same time.

```
select distinct S.sname

from Student S

where S.snum in (select E1.snum

                 from Enrolled E1, Enrolled E2, Class C1, Class C2

                 where E1.snum = E2.snum AND E1.cname <> E2.cname

                 AND E1.cname = C1.cname
```

AND E2.cname = C2.cname AND C1.metts_at = C2.metts_at);

Result Grid

	sname
▶	Rahul

iv. Find the names of faculty members who teach in every room in which some class is taught.

```
select distinct F.fname
from Faculty F,Class C
where F.fid=C.fid
group by C.fid
having count(distinct room) = 5;
```

Result Grid

	fname
▶	Shiva

v. Find the names of faculty members for whom the combined enrollment of the courses that they teach is less than five.

```
select distinct F.fname

from Faculty F

where 5 > (select COUNT(E.snum)

from Class C, Enrolled E

where C.cname = E.cname

and C.fid = F.fid);
```

Result Grid

fname
Harish
MV
Mira
Shiva

vi. Find the names of students who are not enrolled in any class.

```
select distinct S.sname

from Student S

where S.snum NOT IN (select E.snum

from Enrolled E );
```

Result Grid

sname
Rita

vii. For each age value that appears in Students, find the level value that appears most often. For example, if there are more FR level students aged 18 than SR, JR, or SO students aged 18, you should print the pair (18, FR).

```
select S.age, S.lvl

from Student S

group by S.age, S.lvl

having S.lvl IN (select S1.lvl from Student S1

    where S1.age = S.age

group by S1.lvl, S1.age
```

```
having COUNT(S1.snum) >= ALL (select COUNT(S2.snum)

from Student S2

where s1.age = S2.age

group by S2.lvl, S2.age));
```

Result Grid

age	lvl
19	Sr
20	Jr
21	Sr

PROGRAM 9:

SUPPLIER DATABASE:

Consider the following schema:

SUPPLIERS (sid: integer, sname: string, address: string)

PARTS (pid: integer, pname: string, color: string)

CATALOG (sid: integer, pid: integer, cost: real)

The Catalog relation lists the prices charged for parts by Suppliers. Write the following queries in SQL:

i. Find the pnames of parts for which there is some supplier.

ii. Find the snames of suppliers who supply every part.

iii. Find the snames of suppliers who supply every red part.

iv. Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.

v. Find the sids of suppliers who charge more for some part than the average cost of that part (averaged over

all the suppliers who supply that part).

vi. For each part, find the sname of the supplier who charges the most for that part.

vii. Find the sids of suppliers who supply only red parts.

Code:

```
CREATE DATABASE SUPPLIER;
USE SUPPLIER;
CREATE TABLE SUPPLIERS(SID BIGINT(5) PRIMARY KEY, SNAME
VARCHAR(20), CITY VARCHAR(20));
INSERT INTO SUPPLIERS VALUES(10001,'ACME WIDGET','BANGALORE');
INSERT INTO SUPPLIERS VALUES(10002,'JOHNS ','KOLKATA');
INSERT INTO SUPPLIERS VALUES(10003,'VIMAL','MUMBAI');
INSERT INTO SUPPLIERS VALUES(10004,'RELIANCE ','DELHI');
SELECT * FROM SUPPLIERS;
CREATE TABLE PARTS(PID BIGINT(5) PRIMARY KEY, PNAME VARCHAR(20),
COLOR VARCHAR(10));
INSERT INTO PARTS VALUES(20001,'BOOK','RED');
INSERT INTO PARTS VALUES(20002,'PEN','RED');
INSERT INTO PARTS VALUES(20003,'PENCIL','GREEN');
INSERT INTO PARTS VALUES(20004,'MOBILE ','GREEN');
```

```
INSERT INTO PARTS VALUES(20005,'CHARGER','BLACK');
SELECT * FROM PARTS;
CREATE TABLE CATALOG(SID BIGINT(5), PID BIGINT(5), FOREIGN KEY(SID)
REFERENCES SUPPLIERS(SID), FOREIGN KEY(PID) REFERENCES PARTS(PID),
COST FLOAT(6), PRIMARY KEY(SID, PID));
INSERT INTO CATALOG VALUES(10001,20001,10);
INSERT INTO CATALOG VALUES(10001,20002,10);
INSERT INTO CATALOG VALUES(10001,20003,30);
INSERT INTO CATALOG VALUES(10001,20004,10);
INSERT INTO CATALOG VALUES(10001,20005,10);
INSERT INTO CATALOG VALUES(10002,20001,10);
INSERT INTO CATALOG VALUES(10002,20002,20);
INSERT INTO CATALOG VALUES(10003,20003,30);
INSERT INTO CATALOG VALUES(10004,20003,40);
SELECT * FROM CATALOG;
/* 1 - FIND THE PNAMES OF PARTS FOR WHICH THERE IS SOME SUPPLIER. */
SELECT DISTINCT P.PNAME
 FROM PARTS P, CATALOG C
 WHERE P.PID = C.PID;

/* FIND THE SNAMES OF SUPPLIERS WHO SUPPLY EVERY PART */
SELECT S.SNAME FROM SUPPLIERS S WHERE NOT EXISTS (SELECT P.PID FROM
PARTS P WHERE NOT EXISTS (SELECT C.SID FROM CATALOG C WHERE C.SID =
S.SID AND C.PID = P.PID));

/* FIND THE SNAMES OF SUPPLIERS WHO SUPPLY EVERY RED PART. */
SELECT S.SNAME FROM SUPPLIERS S WHERE NOT EXISTS (SELECT P.PID FROM
PARTS P WHERE P.COLOR = 'RED' AND (NOT EXISTS (SELECT C.SID FROM
CATALOG C WHERE C.SID = S.SID AND C.PID = P.PID)));
```

```
/* FIND THE PNAMES OF PARTS SUPPLIED BY ACME WIDGET SUPPLIERS AND
BY NO ONE ELSE */

SELECT P.PNAME FROM PARTS P, CATALOG C, SUPPLIERS S WHERE P.PID

= C.PID AND C.SID = S.SID AND S.SNAME = 'ACME WIDGET' AND NOT EXISTS

(SELECT * FROM CATALOG C1, SUPPLIERS S1 WHERE P.PID = C1.PID AND

C1.SID = S1.SID AND S1.SNAME <> 'ACME WIDGET');


/* FIND THE SIDS OF SUPPLIERS WHO CHARGE MORE FOR SOME PART THAN
THE AVERAGE COST OF THAT PART (AVERAGED OVER

ALL THE SUPPLIERS WHO SUPPLY THAT PART).

 */

SELECT DISTINCT C.SID FROM CATALOG C

WHERE C.COST > ( SELECT AVG (C1.COST)

FROM CATALOG C1

WHERE C1.PID = C.PID );


/* FOR EACH PART, FIND THE SNAME OF THE SUPPLIER WHO CHARGES THE
MOST FOR THAT PART.*/

SELECT P.PID, S.SNAME

FROM PARTS P, SUPPLIERS S, CATALOG C

WHERE C.PID = P.PID

AND C.SID = S.SID

AND C.COST = (SELECT MAX(C1.COST)

FROM CATALOG C1

WHERE C1.PID = P.PID);


/* FIND THE SIDS OF SUPPLIERS WHO SUPPLY ONLY RED PARTS.*/

SELECT DISTINCT C.SID

FROM CATALOG C

WHERE NOT EXISTS ( SELECT *

FROM PARTS P
```

WHERE P.PID = C.PID AND P.COLOR <> 'RED');

Output:

Suppliers 1:

Result Grid | Filter Rows:

SID	SNAME	CITY
10001	ACME WIDGET	BANGALORE
10002	JOHNS	KOLKATA
10003	VIMAL	MUMBAI
10004	RELIANCE	DELHI
NULL	NULL	NULL

Parts 2:

PID	PNAME	COLOR
20001	BOOK	RED
20002	PEN	RED
20003	PENCIL	GREEN
20004	MOBILE	GREEN
20005	CHARGER	BLACK
NULL	NULL	NULL

Catalog3:

SID	PID	COST
10001	20001	10
10001	20002	10
10001	20003	30
10001	20004	10
10001	20005	10
10002	20001	10
10002	20002	20
10003	20003	30
10004	20003	40
NULL	NULL	NULL

/* 1 - FIND THE PNAMES OF PARTS FOR WHICH THERE IS SOME SUPPLIER. */

SELECT DISTINCT P.PNAME

FROM PARTS P, CATALOG C

WHERE P.PID = C.PID;

PNAME
BOOK
PEN
PENCIL
MOBILE
CHARGER

/* 2.FIND THE SNAMES OF SUPPLIERS WHO SUPPLY EVERY PART */

SELECT S.SNAME FROM SUPPLIERS S WHERE NOT EXISTS (SELECT P.PID FROM PARTS P WHERE NOT EXISTS (SELECT C.SID FROM CATALOG C WHERE C.SID = S.SID AND C.PID = P.PID));

Result Grid

	SNAME
▶	ACME WIDGET

/* 3.FIND THE SNAMES OF SUPPLIERS WHO SUPPLY EVERY RED PART. */

SELECT S.SNAME FROM SUPPLIERS S WHERE NOT EXISTS (SELECT P.PID FROM PARTS P WHERE P.COLOR = 'RED' AND (NOT EXISTS (SELECT C.SID FROM CATALOG C WHERE C.SID = S.SID AND C.PID = P.PID)));

Result Grid

	SNAME
▶	ACME WIDGET
	JOHNS

/* 4.FIND THE PNAMES OF PARTS SUPPLIED BY ACME WIDGET SUPPLIERS AND BY NO ONE ELSE */

SELECT P.PNAME FROM PARTS P, CATALOG C, SUPPLIERS S WHERE P.PID = C.PID AND C.SID = S.SID AND S.SNAME = 'ACME WIDGET' AND NOT EXISTS (SELECT * FROM CATALOG C1, SUPPLIERS S1 WHERE P.PID = C1.PID AND C1.SID = S1.SID AND S1.SNAME <> 'ACME WIDGET');

Result Grid

	PNAME
▶	MOBILE
	CHARGER

/*5. FIND THE SIDS OF SUPPLIERS WHO CHARGE MORE FOR SOME PART THAN THE AVERAGE COST OF THAT PART (AVERAGED OVER

ALL THE SUPPLIERS WHO SUPPLY THAT PART).

```
 */
SELECT DISTINCT C.SID FROM CATALOG C
WHERE C.COST > ( SELECT AVG (C1.COST)
FROM CATALOG C1
WHERE C1.PID = C.PID );
```

Result Grid

	SID
▶	10002
	10004

```
/* 6.FOR EACH PART, FIND THE SNAME OF THE SUPPLIER WHO CHARGES THE
MOST FOR THAT PART.*/
SELECT P.PID, S.SNAME
FROM PARTS P, SUPPLIERS S, CATALOG C
WHERE C.PID = P.PID
AND C.SID = S.SID
AND C.COST = (SELECT MAX(C1.COST)
FROM CATALOG C1
WHERE C1.PID = P.PID);
```

Result Grid Filter Ro

	PID	SNAME
▶	20001	ACME WIDGET
	20004	ACME WIDGET
	20005	ACME WIDGET
	20001	JOHNS
	20002	JOHNS
	20003	RELIANCE

```
/*7. FIND THE SIDS OF SUPPLIERS WHO SUPPLY ONLY RED PARTS.*/
SELECT DISTINCT C.SID
FROM CATALOG C
WHERE NOT EXISTS ( SELECT *
```

```
FROM PARTS P
WHERE P.PID = C.PID AND P.COLOR <> 'RED' );
```

PROGRAM 10:

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)

IAMARKS(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 ‘1BI15CS101’ 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.

Code:

```
CREATE DATABASE COLLEGEDB;
USE COLLEGEDB;
```

```
CREATE TABLE STUDENT (
USN VARCHAR (10),
SNAME VARCHAR (25),
ADDRESS VARCHAR (25),
PHONE LONG,
GENDER CHAR (1),
PRIMARY KEY (USN));

select * from student;

CREATE TABLE SEMSEC (
SSID VARCHAR (5),
SEM INT,
SEC CHAR (1),
PRIMARY KEY (SSID));
select * from semsec;

CREATE TABLE CLASS (
USN VARCHAR (10),
SSID VARCHAR (5),
PRIMARY KEY (USN, SSID),
FOREIGN KEY (USN) REFERENCES STUDENT (USN),
FOREIGN KEY (SSID) REFERENCES SEMSEC (SSID));
select * from class;
```

```
CREATE TABLE SUBJECT (
SUBCODE VARCHAR (8),
TITLE VARCHAR (20),
SEM INT,
CREDITS INT,
PRIMARY KEY (SUBCODE));
select * from subject;

CREATE TABLE IAMARKS (
USN VARCHAR (10),
SUBCODE VARCHAR (8),
SSID VARCHAR (5),
TEST1 INT,
TEST2 INT,
TEST3 INT,
FINALIA INT,
PRIMARY KEY (USN, SUBCODE, SSID),
FOREIGN KEY (USN) REFERENCES STUDENT (USN),
FOREIGN KEY (SUBCODE) REFERENCES SUBJECT (SUBCODE),
FOREIGN KEY (SSID) REFERENCES SEMSEC (SSID));
select * from iamarks;

INSERT INTO STUDENT VALUES
('1RN13CS020','AKSHAY','BELAGAVI', 8877881122,'M');
```

```
INSERT INTO STUDENT VALUES
('1RN13CS062','SANDHYA','BENGALURU', 7722829912,'F');

INSERT INTO STUDENT VALUES
('1RN13CS091','TEESHA','BENGALURU', 7712312312,'F');

INSERT INTO STUDENT VALUES
('1RN13CS066','SUPRIYA','MANGALURU', 8877881122,'F');

INSERT INTO STUDENT VALUES
('1RN14CS010','ABHAY','BENGALURU', 9900211201,'M');

INSERT INTO STUDENT VALUES
('1RN14CS032','BHASKAR','BENGALURU', 9923211099,'M');

INSERT INTO STUDENT VALUES
('1RN14CS025','ASMI','BENGALURU', 7894737377,'F');

INSERT INTO STUDENT VALUES
('1RN15CS011','AJAY','TUMKUR', 9845091341,'M');


INSERT INTO STUDENT VALUES
('1RN15CS029','CHITRA','DAVANGERE', 7696772121,'F');

INSERT INTO STUDENT VALUES
('1RN15CS045','JEEVA','BELLARY', 9944850121,'M');

INSERT INTO STUDENT VALUES
('1RN15CS091','SANTOSH','MANGALURU', 8812332201,'M');

INSERT INTO STUDENT VALUES
('1RN16CS045','ISMAIL','KALBURGI', 9900232201,'M');

INSERT INTO STUDENT VALUES
('1RN16CS088','SAMEERA','SHIMOGA', 9905542212,'F');

INSERT INTO STUDENT VALUES
('1RN16CS122','VINAYAKA','CHIKAMAGALUR', 8800880011,'M');
```

```
INSERT INTO SEMSEC VALUES ('CSE8A', 8,'A');
INSERT INTO SEMSEC VALUES ('CSE8B', 8,'B');
INSERT INTO SEMSEC VALUES ('CSE8C', 8,'C');
INSERT INTO SEMSEC VALUES ('CSE7A', 7,'A');
INSERT INTO SEMSEC VALUES ('CSE7B', 7,'B');
INSERT INTO SEMSEC VALUES ('CSE7C', 7,'C');
INSERT INTO SEMSEC VALUES ('CSE6A', 6,'A');
INSERT INTO SEMSEC VALUES ('CSE6B', 6,'B');
INSERT INTO SEMSEC VALUES ('CSE6C', 6,'C');
INSERT INTO SEMSEC VALUES ('CSE5A', 5,'A');
INSERT INTO SEMSEC VALUES ('CSE5B', 5,'B');
INSERT INTO SEMSEC VALUES ('CSE5C', 5,'C');
INSERT INTO SEMSEC VALUES ('CSE4A', 4,'A');
INSERT INTO SEMSEC VALUES ('CSE4B', 4,'B');
INSERT INTO SEMSEC VALUES ('CSE4C', 4,'C');
INSERT INTO SEMSEC VALUES ('CSE3A', 3,'A');
INSERT INTO SEMSEC VALUES ('CSE3B', 3,'B');
INSERT INTO SEMSEC VALUES ('CSE3C', 3,'C');
INSERT INTO SEMSEC VALUES ('CSE2A', 2,'A');
INSERT INTO SEMSEC VALUES ('CSE2B', 2,'B');
INSERT INTO SEMSEC VALUES ('CSE2C', 2,'C');
INSERT INTO SEMSEC VALUES ('CSE1A', 1,'A');
INSERT INTO SEMSEC VALUES ('CSE1B', 1,'B');
```

```
INSERT INTO SEMSEC VALUES ('CSE1C', 1,'C');

INSERT INTO CLASS VALUES ('1RN13CS020','CSE8A');
INSERT INTO CLASS VALUES ('1RN13CS062','CSE8A');
INSERT INTO CLASS VALUES ('1RN13CS066','CSE8B');
INSERT INTO CLASS VALUES ('1RN13CS091','CSE8C');
INSERT INTO CLASS VALUES ('1RN14CS010','CSE7A');
INSERT INTO CLASS VALUES ('1RN14CS025','CSE7A');
INSERT INTO CLASS VALUES ('1RN14CS032','CSE7A');
INSERT INTO CLASS VALUES ('1RN15CS011','CSE4A');
INSERT INTO CLASS VALUES ('1RN15CS029','CSE4A');
INSERT INTO CLASS VALUES ('1RN15CS045','CSE4B');
INSERT INTO CLASS VALUES ('1RN15CS091','CSE4C');
INSERT INTO CLASS VALUES ('1RN16CS045','CSE3A');
INSERT INTO CLASS VALUES ('1RN16CS088','CSE3B');
INSERT INTO CLASS VALUES ('1RN16CS122','CSE3C');


INSERT INTO SUBJECT VALUES ('10CS81','ACA', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS82','SSM', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS83','NM', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS84','CC', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS85','PW', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS71','OOAD', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS72','ECS', 7, 4);
```

```
INSERT INTO SUBJECT VALUES ('10CS73','PTW', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS74','DWDM', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS75','JAVA', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS76','SAN', 7, 4);
INSERT INTO SUBJECT VALUES ('15CS51', 'ME', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS52','CN', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS53','DBMS', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS54','ATC', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS55','JAVA', 5, 3);
INSERT INTO SUBJECT VALUES ('15CS56','AI', 5, 3);
INSERT INTO SUBJECT VALUES ('15CS41','M4', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS42','SE', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS43','DAA', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS44','MPMC', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS45','OOC', 4, 3);
INSERT INTO SUBJECT VALUES ('15CS46','DC', 4, 3);
INSERT INTO SUBJECT VALUES ('15CS31','M3', 3, 4);
INSERT INTO SUBJECT VALUES ('15CS32','ADE', 3, 4);
INSERT INTO SUBJECT VALUES ('15CS33','DSA', 3, 4);
INSERT INTO SUBJECT VALUES ('15CS34','CO', 3, 4);
INSERT INTO SUBJECT VALUES ('15CS35','USP', 3, 3);
INSERT INTO SUBJECT VALUES ('15CS36','DMS', 3, 3);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2,
TEST3) VALUES ('1RN13CS091','10CS81','CSE8C', 15, 16, 18);
```

```
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1RN13CS091','10CS82','CSE8C', 12, 19, 14);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1RN13CS091','10CS83','CSE8C', 19, 15, 20);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1RN13CS091','10CS84','CSE8C', 20, 16, 19);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1RN13CS091','10CS85','CSE8C', 15, 15, 12);


/*1. List all the student details studying in fourth semester ‘C’ section. */


SELECT S.*, SS.SEM, SS.SEC

FROM STUDENT S, SEMSEC SS, CLASS C

WHERE S.USN = C.USN AND

SS.SSID = C.SSID AND

SS.SEM = 4 AND SS.SEC='C';


/*2. Compute the total number of male and female students in each semester and in each section. */

SELECT SS.SEM, SS.SEC, S.GENDER, COUNT(S.GENDER) AS COUNT

FROM STUDENT S, SEMSEC SS, CLASS C

WHERE S.USN = C.USN AND

SS.SSID = C.SSID

GROUP BY SS.SEM, SS.SEC, S.GENDER

ORDER BY SEM;
```

/*3. Create a view of Test1 marks of student USN ‘1BI15CS101’ in all subjects. */

```
CREATE VIEW STU_TEST1_MARKS_VIEW
AS
SELECT TEST1, SUBCODE
FROM IAMARKS
WHERE USN = '1RN13CS091';

SELECT * FROM STU_TEST1_MARKS_VIEW;
```

/*5. 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. */

```
SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER,
(CASE
WHEN IA.FINALIA BETWEEN 17 AND 20 THEN 'OUTSTANDING'
WHEN IA.FINALIA BETWEEN 12 AND 16 THEN 'AVERAGE'
ELSE 'WEAK'
END) AS CAT
FROM STUDENT S, SEMSEC SS, IAMARKS IA, SUBJECT SUB
```

WHERE S.USN = IA.USN AND

SS.SSID = IA.SSID AND

SUB.SUBCODE = IA.SUBCODE AND

SUB.SEM = 8;

Output:

Student 1:

USN	SNAME	ADDRESS	PHONE	GENDER
NULL	NULL	NULL	NULL	NULL

Semsec 2:

Result Grid | Filter Ro

SSID	SEM	SEC
NULL	NULL	NULL

Class 3:

Result Grid

USN	SSID
NULL	NULL

Subject 4:

Result Grid | Filter Rows:

SUBCODE	TITLE	SEM	CREDITS
NULL	NULL	NULL	NULL

Iamarks 5:

Result Grid | Filter Rows: | Edit:

USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINALIA
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Query 1:

```
SELECT S.*, SS.SEM, SS.SEC
FROM STUDENT S, SEMSEC SS, CLASS C
WHERE S.USN = C.USN AND
SS.SSID = C.SSID AND
SS.SEM = 4 AND SS.SEC='C';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

USN	SNAME	ADDRESS	PHONE	GENDER	SEM	SEC
1RN15CS091	SANTOSH	MANGALURU	8812332201	M	4	C

Query 2 :

```
SELECT SS.SEM, SS.SEC, S.GENDER, COUNT(S.GENDER) AS COUNT
FROM STUDENT S, SEMSEC SS, CLASS C
WHERE S.USN = C.USN AND
SS.SSID = C.SSID
GROUP BY SS.SEM, SS.SEC, S.GENDER
ORDER BY SEM;
```

Result Grid | Filter Rows:

SEM	SEC	GENDER	COUNT
3	A	M	1
3	B	F	1
3	C	M	1
4	A	F	1
4	A	M	1
4	B	M	1
4	C	M	1
7	A	F	1
7	A	M	2
8	A	F	1
8	A	M	1
8	B	F	1
8	C	F	1

Query 3:

```
CREATE VIEW STU_TEST1_MARKS_VIEW
AS
SELECT TEST1, SUBCODE
FROM IAMARKS
```

WHERE USN = '1RN13CS091';

SELECT * FROM STU_TEST1_MARKS_VIEW;

Result Grid | Filter Rows:

TEST1	SUBCODE
15	10CS81
12	10CS82
19	10CS83
20	10CS84
15	10CS85

Query 4:

```
SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER,
(CASE
WHEN IA.FINALIA BETWEEN 17 AND 20 THEN 'OUTSTANDING'
WHEN IA.FINALIA BETWEEN 12 AND 16 THEN 'AVERAGE'
ELSE 'WEAK'
END) AS CAT
FROM STUDENT S, SEMSEC SS, IAMARKS IA, SUBJECT SUB
WHERE S.USN = IA.USN AND
SS.SSID = IA.SSID AND
SUB.SUBCODE = IA.SUBCODE AND
SUB.SEM = 8;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

USN	SNAME	ADDRESS	PHONE	GENDER	CAT
1RN13CS091	TEESHA	BENGALURU	7712312312	F	WEAK
1RN13CS091	TEESHA	BENGALURU	7712312312	F	WEAK
1RN13CS091	TEESHA	BENGALURU	7712312312	F	WEAK
1RN13CS091	TEESHA	BENGALURU	7712312312	F	WEAK
1RN13CS091	TEESHA	BENGALURU	7712312312	F	WEAK

----------**********----------