B.M.S. COLLEGE OF ENGINEERING, BANGALORE-19
(Autonomous College under VTU)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DATABASE MANAGEMENT SYSTEM LABORATORY RECORD

NAME: NITHIN.C
USN: 1BM19CS106

PROGRAM: BACHELOR OF ENGINEERING
SEMESTER: IV
SESSION: APR-JUL 2021
COURSE CODE: 19CS4PCDBM
COURSE TITLE: DATABASE MANAGEMENT SYSTEM
CREDITS: 4

DBMS Lab List

Experiment #	Name of Experiment
1	Insurance Database
2	Banking Enterprise Database
3	Supplier Database
4	Student Faculty Database
5	Airline Flight Database
6	Order Processing Database
7	Book dealer Database
8	Student Enrolment Database
9	Movie Database
10	College Database

PROGRAM1:

Consider the Insurance database given below. The primary keys are underlined and the data types are specified.
PERSON (driver-id #: String, name: String, address: String)
CAR (Regno: String, model: String, year: int)
ACCIDENT (report-number: int, date: date, location: String)
OWNS (driver-id #: String, Regno: String)
PARTICIPATED (driver-id: String, Regno: String, report-number: int, damage-amount: int)

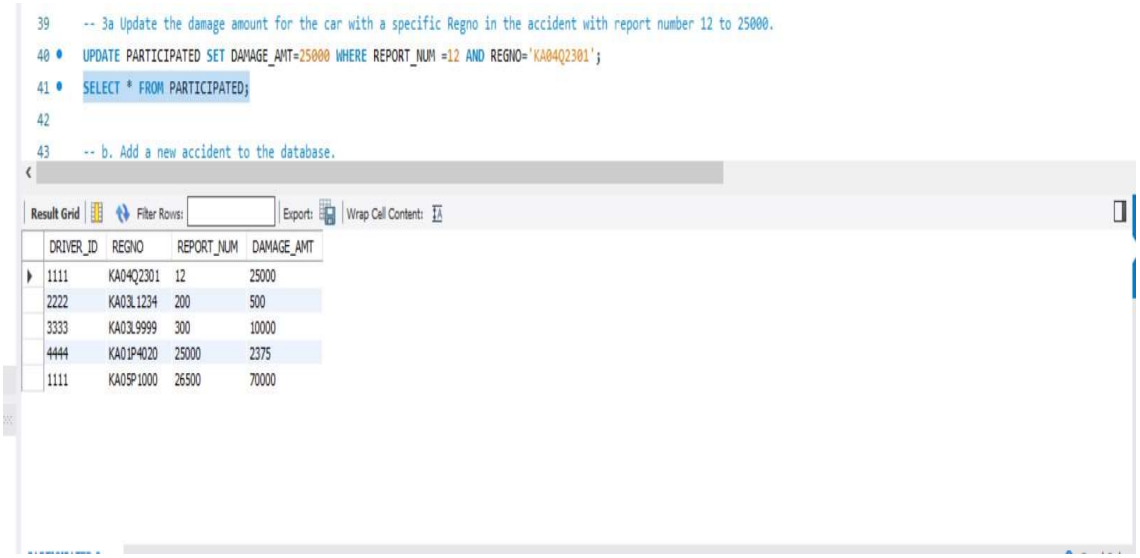
SQL SCRIPT:

```
-- i. Create the above tables by properly specifying the
primary keys and the foreign keys.
-- ii. Enter at least five tuples for each relation.
CREATE DATABASE INSURANCE;
USE INSURANCE;
SHOW DATABASES;
CREATE TABLE PERSON(DRIVER_ID VARCHAR(20),PNAME
VARCHAR(30),ADDRESSS VARCHAR(20),PRIMARY KEY(DRIVER_ID));
CREATE TABLE CAR(REGNO VARCHAR(20),MODEL VARCHAR(20),CYEAR
INT,PRIMARY KEY(REGNO));
CREATE TABLE ACCIDENT(REPORT_NUM INT,ADATE DATE,LOCATION
VARCHAR(30),PRIMARY KEY(REPORT_NUM));
SHOW TABLES;
CREATE TABLE OWNS(DRIVER_ID VARCHAR(20),REGNO
VARCHAR(20),PRIMARY KEY(DRIVER_ID,REGNO),
FOREIGN KEY(DRIVER_ID) REFERENCES PERSON(DRIVER_ID) ON
DELETE CASCADE,
FOREIGN KEY(REGNO) REFERENCES CAR(REGNO) ON DELETE CASCADE);
```

```
CREATE TABLE PARTICIPATED(DRIVER_ID VARCHAR(20),REGNO
VARCHAR(20),REPORT_NUM INT,DAMAGE_AMT DOUBLE,
FOREIGN KEY(DRIVER_ID,REGNO) REFERENCES
OWNS(DRIVER_ID,REGNO) ON DELETE CASCADE,
FOREIGN KEY(REPORT_NUM) REFERENCES ACCIDENT(REPORT_NUM) ON
DELETE CASCADE);
DESC PARTICIPATED;
INSERT INTO PERSON VALUES('1111','RAMU','K.S LAYOUT');
INSERT INTO PERSON(DRIVER_ID, PNAME,ADDRESSS) VALUES
(2222,'JOHN', 'INDIRANAGAR'),
(3333, 'PRIYA', 'JAYANAGAR'),
(4444, 'GOPAL', 'WHITEFIELD'),
(5555, 'LATHA', 'VIJAYANAGAR');
INSERT INTO CAR(REGNO,MODEL,CYEAR) VALUES
('KA04Q2301','MARUTHI-DX', 2000),
('KA05P1000', 'FORDICON', 2000),
('KA03L1234', 'ZEN-VXI', 1999),
('KA03L9999', 'MARUTHI-DX', 2002),
('KA01P4020', 'INDICA-VX', 2002);
INSERT INTO ACCIDENT(REPORT_NUM,ADATE,LOCATION) VALUES (12,
'2002-06-01', 'M G ROAD'),
(200, '2002-12-10', 'DOUBLEROAD'),
(300, '1999-07-23', 'M G ROAD'),
(25000, '2000-06-11', 'RESIDENCY ROAD'),
(26500, '2001-10-16', 'RICHMOND ROAD');
INSERT INTO OWNS(DRIVER_ID,REGNO) VALUES ('1111',
'KA04Q2301'), ('1111', 'KA05P1000'), ('2222', 'KA03L1234'),
('3333', 'KA03L9999'), ('4444', 'KA01P4020');
INSERT INTO PARTICIPATED(DRIVER_ID, REGNO, REPORT_NUM,
DAMAGE_AMT) VALUES ('1111', 'KA04Q2301', 12, 20000),
 ('2222', 'KA03L1234', 200, 500),
 ('3333', 'KA03L9999', 300, 10000),
 ('4444', 'KA01P4020', 25000, 2375),
 ('1111', 'KA05P1000', 26500, 70000);

-- 3a Update the damage amount for the car with a specific
Regno in the accident with report number 12 to 25000.
UPDATE PARTICIPATED SET DAMAGE_AMT=25000 WHERE REPORT_NUM
=12 AND REGNO='KA04Q2301';
SELECT * FROM PARTICIPATED;

-- b. Add a new accident to the database.
INSERT INTO ACCIDENT VALUE('20000','2003-05-13','Frazer
Town');
SELECT * FROM ACCIDENT;
```

```
-- iv. Find the total number of people who owned cars that
involved in accidents in 2008.
SELECT COUNT(*) FROM ACCIDENT WHERE YEAR(ADATE)=2002;

-- v. Find the number of accidents in which cars belonging
to a specific model were involved.
SELECT COUNT(A.REPORT_NUM) FROM ACCIDENT A, PARTICIPATED P,
CAR C
WHERE A.REPORT_NUM=P.REPORT_NUM
AND
P.REGNO=C.REGNO
AND
C.MODEL='MARUTHI-DX';
```

Output:



```
50    -- v. Find the number of accidents in which cars belonging to a specific model were involved.
51 •  SELECT COUNT(A.REPORT_NUM) FROM ACCIDENT A, PARTICIPATED P, CAR C
52    WHERE A.REPORT_NUM=P.REPORT_NUM
53    AND
54    P.REGNO=C.REGNO
55    AND
56    C.MODEL='MARUTHI-DX';
57
58
```

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

COUNT(A.REPORT_NUM)
2

```
43    -- b. Add a new accident to the database.
44 •  INSERT INTO ACCIDENT VALUE('20000','2003-05-13','Frazer Town');
45 •  SELECT * FROM ACCIDENT;
46
47    -- iv. Find the total number of people who owned cars that involved in accidents in 2008.
48 •  SELECT COUNT(*) FROM ACCIDENT WHERE YEAR(ADATE)=2002;
49
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

REPORT_NUM	ADATE	LOCATION
12	2002-06-01	M G ROAD
200	2002-12-10	DOUBLEROAD
300	1999-07-23	M G ROAD
20000	2003-05-13	Frazer Town
25000	2000-06-11	RESIDENCY ROAD
26500	2001-10-16	RICHMOND ROAD
NULL	NULL	NULL

PROGRAM 2:

Consider the following database for a banking enterprise.
BRANCH (branch-name: String, branch-city: String, assets: real)
ACCOUNTS (accno: int, branch-name: String, balance: real)

DEPOSITOR (customer-name: String, customer-street: String, customer-city: String)
LOAN (loan-number: int, branch-name: String, amount: real)
BORROWER (customer-name: String, loan-number: 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. Find all the customers who have at least two accounts at the Main branch.
iv. Find all the customers who have an account at all the branches located in a specific city.
v. Demonstrate how you delete all account tuples at every branch located in a specific city.

SQL SCRIPT:

```
-- i. Create the above tables by properly specifying the
primary keys and the foreign keys.
-- ii. Enter at least five tuples for each relation.
CREATE DATABASE BANKING_ENTERPRISE;
USE BANKING_ENTERPRISE;
-- SHOW DATABASES;
create table branch(
branch_name varchar(30) primary key,
branch_city varchar(30),
assets real);

create table accounts(
```

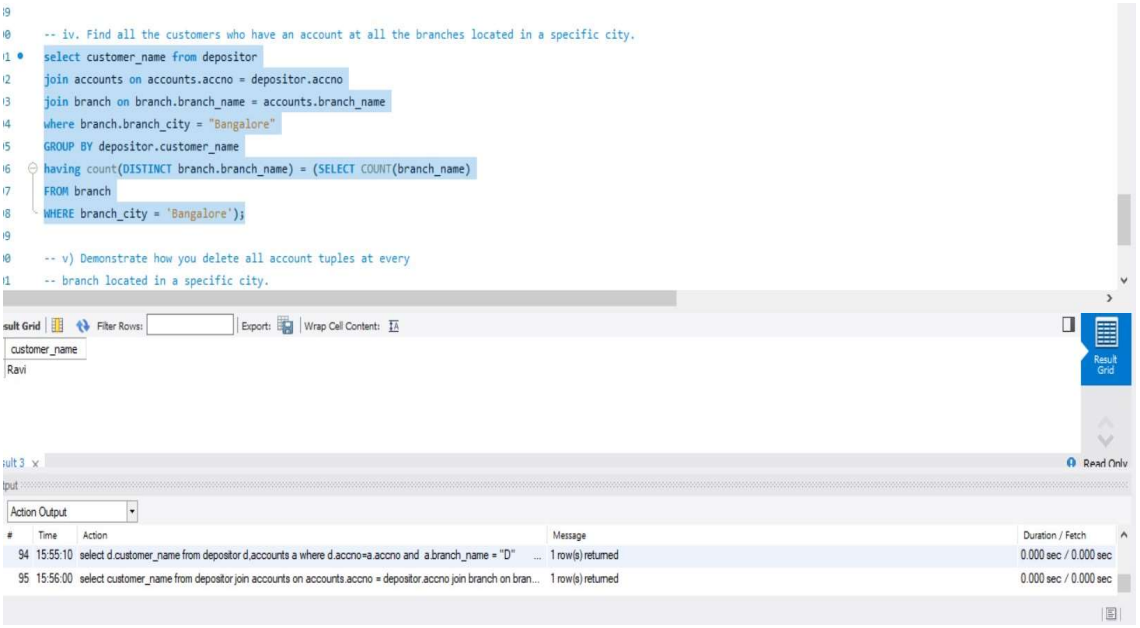
```
accno int primary key,
branch_name varchar(30),
balance real,
foreign key (branch_name) references branch(branch_name)
on delete cascade on update cascade);

create table customer(
customer_name varchar(30) primary key,
customer_street varchar(20),
customer_city varchar(20));

create table depositor(
customer_name varchar(30),
accno int,
primary key(customer_name ,accno),
foreign key (accno) references accounts(accno) on delete
cascade on update cascade,
foreign key (customer_name) references
customer(customer_name) on delete cascade on update
cascade);

create table loan(
loan_number int primary key,
branch_name varchar(30),
amount real,
foreign key (branch_name) references branch(branch_name)
);

create table borrower (
customer_name varchar(30),
loan_number int,
primary key(customer_name, loan_number),
foreign key (customer_name) references
customer(customer_name) on delete cascade on update
cascade,
foreign key (loan_number) references loan(loan_number)
on delete cascade on update cascade);



insert into branch(branch_name,branch_city,assets)
values ('A','Bangalore',190000),

('B','Bangalore',200000),

('C','Delhi',235344),
```

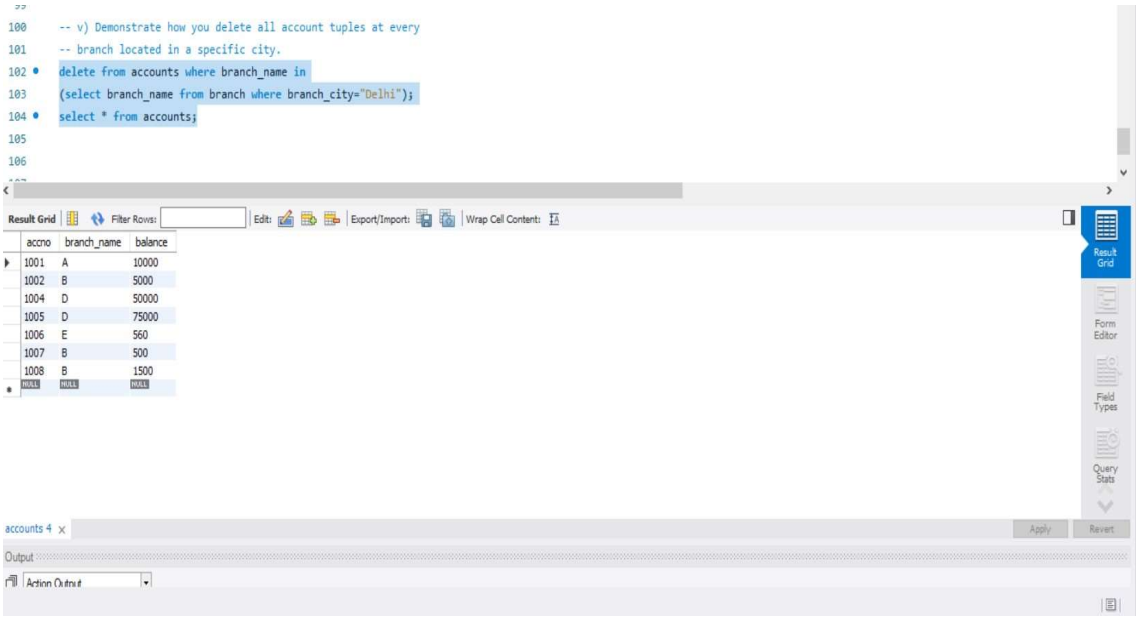
```
('D','Chennai',1050560),

('E','Chennai',678909);
insert into accounts(accno,branch_name,balance) VALUES
(1001,'A',10000),

(1002,'B',5000),

(1003,'C',7500),

(1004,'D',50000),
                                        (1005,'D',75000),
                                        (1006,'E',560),

(1007,"B",500),

(1008,"B",1500);
insert into
customer(customer_name,customer_street,customer_city)
VALUES ("Ravi","Dasarahalli","Bangalore"),

("Shyam","Indiranagar","Delhi"),

("Seema","Vasantnagar","Chennai"),

("Arpita","Church Street","Bangalore"),

("Vinay","MG Road","Chennai");
insert into depositor(customer_name,accno) VALUES
("Ravi",1001),

("Ravi",1002),

("Shyam",1003),
                                    ("Seema",1004),

("Seema",1005),

("Arpita",1006),
                                    ("Vinay",1007),

("Vinay",1008);

insert into loan(loan_number,branch_name,amount) VALUES
(001,'A',10000),
```

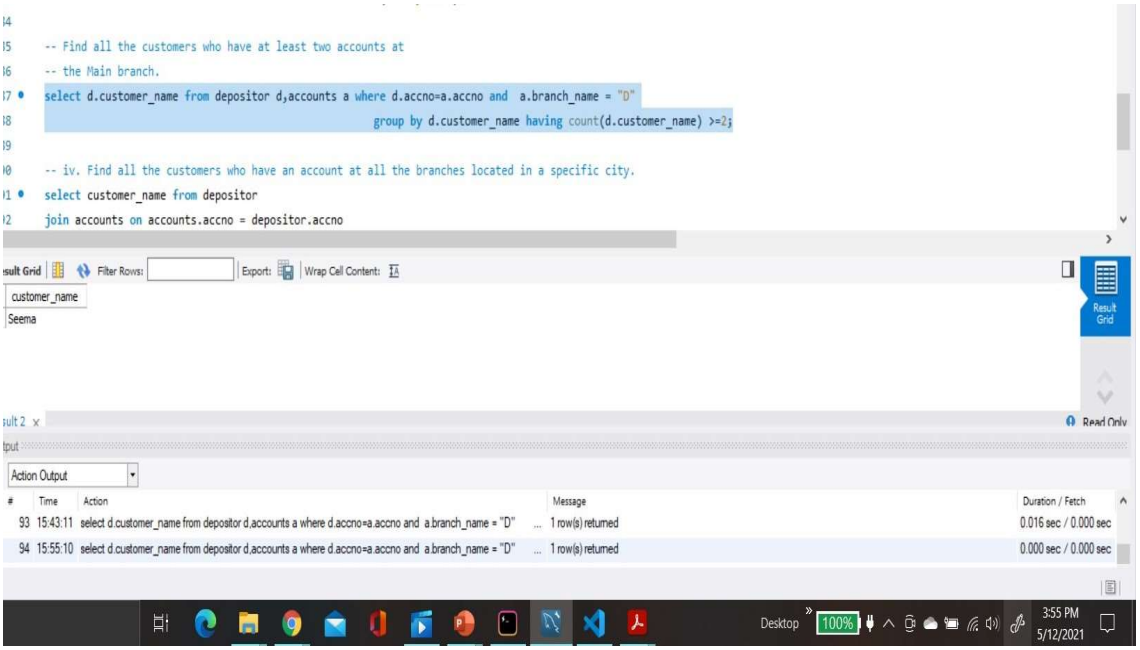
```
(002,'B',25000),

(003,'B',250000),

(004,'C',5000),
                                        (005,'E',90000);

insert into borrower(customer_name,loan_number) VALUES
("Arpita",001),

("Ravi",002),

("Arpita",003),

("Shyam",004),

("Vinay",005);

-- Find all the customers who have at least two accounts
at
-- the Main branch.
select d.customer_name from depositor d,accounts a where
d.accno=a.accno and  a.branch_name = "D"

group by d.customer_name having count(d.customer_name)
>=2;

-- iv. Find all the customers who have an account at all
the branches located in a specific city.
select customer_name from depositor
join accounts on accounts.accno = depositor.accno
join branch on branch.branch_name = accounts.branch_name
where branch.branch_city = "Bangalore"
GROUP BY depositor.customer_name
having count(DISTINCT branch.branch_name) = (SELECT
COUNT(branch_name)
FROM branch
WHERE branch_city = 'Bangalore');

-- v) Demonstrate how you delete all account tuples at
every
-- branch located in a specific city.
delete from accounts where branch_name in
(select branch_name from branch where
branch_city="Delhi");
```

```
select * from accounts;
```

Output:







PROGRAM 3:

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.

SQL SCRIPT:

```
Create Database Supplier;

use Supplier;

-- Create tables
create table suppliers(
sid integer primary key,
sname varchar(20),
address varchar(50));
```

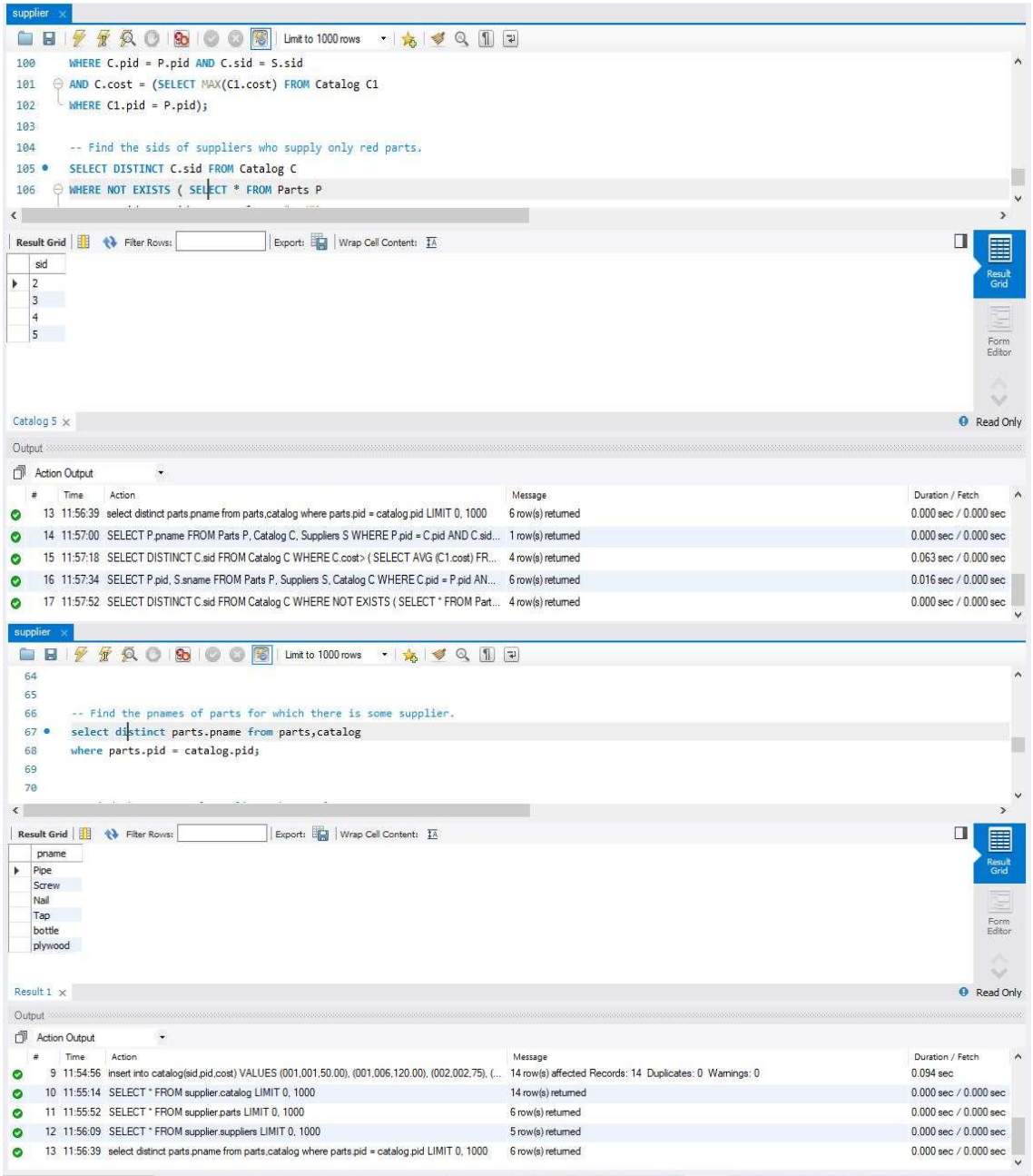
```
create table parts(
pid integer primary key,
pname varchar(20),
color varchar(10));

create table catalog(
sid integer,
pid integer,
cost real,
primary key(sid,pid),
foreign key(sid) references suppliers(sid) on
delete cascade on update cascade,
foreign key(pid) references parts(pid) on
delete cascade on update cascade);

-- Insert values
insert into suppliers(sid,sname,address)
VALUES
(001,'Rohan','Mangalore'),
(002,'Avni','Bangalore'),
(003,'Pratibha','Bagalkot'),
(004,'Rahul','Udupi'),
(005,'Prithvi','Hassan');

insert into parts(pid,pname,color) VALUES
(001,'Pipe','white'),
(002,'Screw','red'),
(003,'Nail','black'),
```

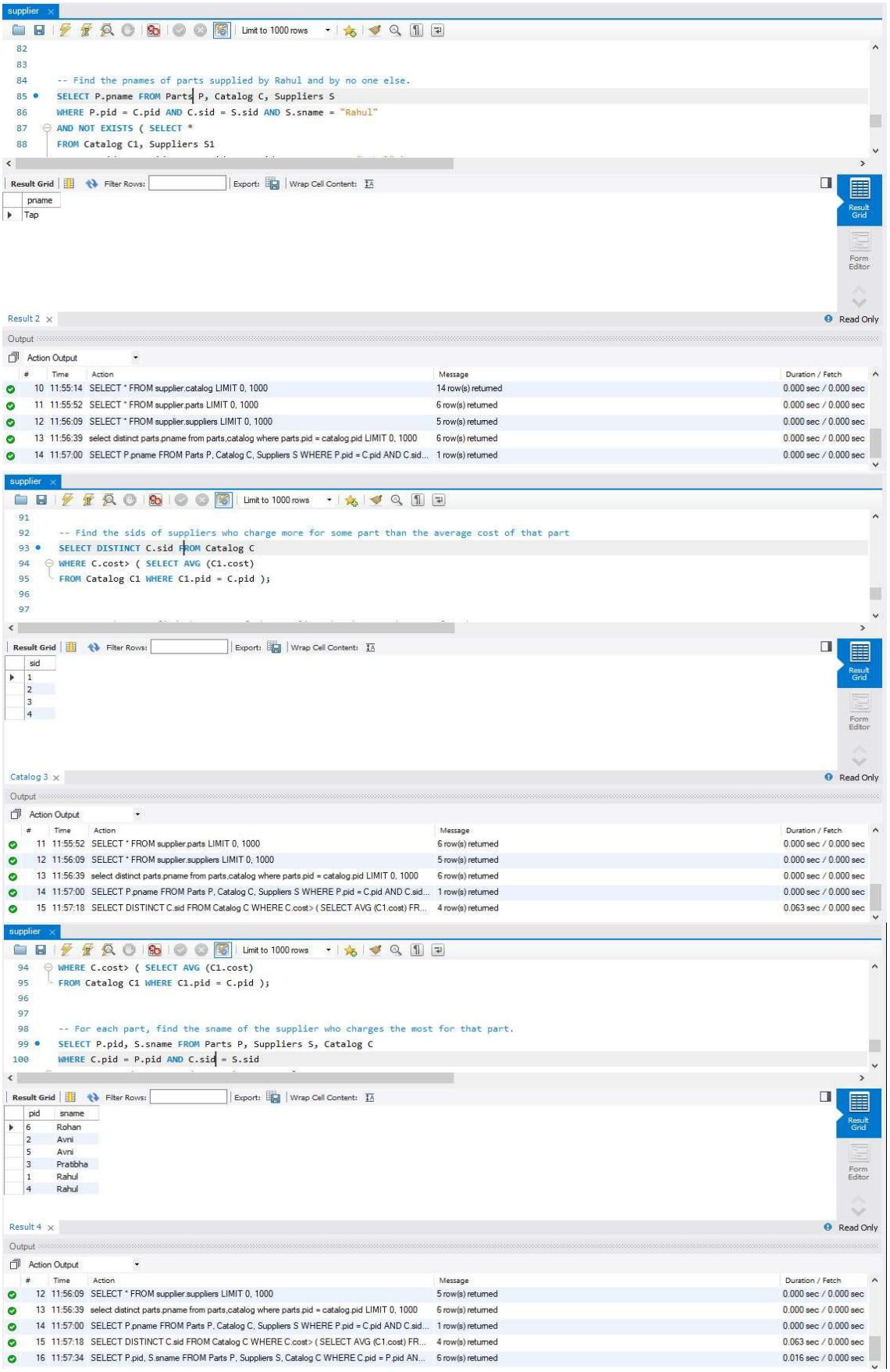
```
(004,'Tap','grey'),
(005,'bottle','red'),
(006,'plywood','brown');

insert into catalog(sid,pid,cost) VALUES
(001,001,50.00),
(001,006,120.00),
(002,002,75),
(002,005,100),
(003,002,45),
(003,003,75),
(004,001,140),
(004,002,38),
(004,003,42),
(004,004,310),
(004,005,79),
(004,006,110),
(005,002,50),
(005,003,48);




-- Find the pnames of parts for which there
is some supplier.
select distinct parts.pname from
parts,catalog
where parts.pid = catalog.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)
except
(selectc.pid from catalog c where c.sid =
s.sid));


-- 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") except
(select c.pid from catalogc,parts p
where c.sid = s.sid and c.pid = p.pid and
p.color = "red"));


-- Find the pnames of parts supplied by
Rahul 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 = "Rahul"
AND NOT EXISTS ( SELECT *
```

```
FROM Catalog C1, Suppliers S1
WHERE P.pid = C1.pid AND C1.sid = S1.sid
AND S1.sname<>"Rahul" );


-- Find the sids of suppliers who charge
more for some part than the average cost of
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:


supplier
Limit to 1000 rows
100 WHERE C.pid = P.pid AND C.sid = S.sid
101 AND C.cost = (SELECT MAX(C1.cost) FROM Catalog C1
102 WHERE C1.pid = P.pid);
103
104 -- Find the sids of suppliers who supply only red parts.
105 SELECT DISTINCT C.sid FROM Catalog C
106 WHERE NOT EXISTS (SELECT * FROM Parts P
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
sid
2
3
4
5
Result Grid
Form Editor
Catalog 5
Read Only
Output
Action Output
Time Action Message Duration / Fetch
13 11:56:39 select distinct parts.pname from parts,catalog where parts.pid = catalog.pid LIMIT 0, 1000 6 row(s) returned 0.000 sec / 0.000 sec
14 11:57:00 SELECT P.pname FROM Parts P, Catalog C, Suppliers S WHERE P.pid = C.pid AND C.sid... 1 row(s) returned 0.000 sec / 0.000 sec
15 11:57:18 SELECT DISTINCT C.sid FROM Catalog C WHERE C.cost> (SELECT AVG (C1.cost) FR... 4 row(s) returned 0.063 sec / 0.000 sec
16 11:57:34 SELECT P.pid, S.sname FROM Parts P, Suppliers S, Catalog C WHERE C.pid = P.pid AN... 6 row(s) returned 0.016 sec / 0.000 sec
17 11:57:52 SELECT DISTINCT C.sid FROM Catalog C WHERE NOT EXISTS (SELECT * FROM Part... 4 row(s) returned 0.000 sec / 0.000 sec
supplier
Limit to 1000 rows
64
65
66 -- Find the pnames of parts for which there is some supplier.
67 select distinct parts.pname from parts,catalog
68 where parts.pid = catalog.pid;
69
70
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
pname
Pipe
Screw
Nail
Tap
bottle
plywood
Result Grid
Form Editor
Result 1
Read Only
Output
Action Output
Time Action Message Duration / Fetch
9 11:54:56 insert into catalog(sid,pid,cost) VALUES (001,001,50.00), (001,006,120.00), (002,002,75), (... 14 row(s) affected Records: 14 Duplicates: 0 Warnings: 0 0.094 sec
10 11:55:14 SELECT * FROM supplier.catalog LIMIT 0, 1000 14 row(s) returned 0.000 sec / 0.000 sec
11 11:55:52 SELECT * FROM supplier.parts LIMIT 0, 1000 6 row(s) returned 0.000 sec / 0.000 sec
12 11:56:09 SELECT * FROM supplier.suppliers LIMIT 0, 1000 5 row(s) returned 0.000 sec / 0.000 sec
13 11:56:39 select distinct parts.pname from parts,catalog where parts.pid = catalog.pid LIMIT 0, 1000 6 row(s) returned 0.000 sec / 0.000 sec



```
supplier
Limit to 1000 rows

82
83
84      -- Find the pnames of parts supplied by Rahul and by no one else.
85      SELECT P.pname FROM Parts P, Catalog C, Suppliers S
86      WHERE P.pid = C.pid AND C.sid = S.sid AND S.sname = "Rahul"
87      AND NOT EXISTS ( SELECT *
88      FROM Catalog C1, Suppliers S1
```

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

pname
Tap

Result 2 — Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
10	11:55:14	SELECT * FROM supplier.catalog LIMIT 0, 1000	14 row(s) returned	0.000 sec / 0.000 sec
11	11:55:52	SELECT * FROM supplier.parts LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
12	11:56:09	SELECT * FROM supplier.suppliers LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
13	11:56:39	select distinct parts.pname from parts,catalog where parts.pid = catalog.pid LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
14	11:57:00	SELECT P.pname FROM Parts P, Catalog C, Suppliers S WHERE P.pid = C.pid AND C.sid...	1 row(s) returned	0.000 sec / 0.000 sec

```
supplier
Limit to 1000 rows

91
92      -- Find the sids of suppliers who charge more for some part than the average cost of that part
93      SELECT DISTINCT C.sid FROM Catalog C
94      WHERE C.cost> ( SELECT AVG (C1.cost)
95      FROM Catalog C1 WHERE C1.pid = C.pid );
96
97
```

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

sid
1
2
3
4

Catalog 3 — Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
11	11:55:52	SELECT * FROM supplier.parts LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
12	11:56:09	SELECT * FROM supplier.suppliers LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
13	11:56:39	select distinct parts.pname from parts,catalog where parts.pid = catalog.pid LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
14	11:57:00	SELECT P.pname FROM Parts P, Catalog C, Suppliers S WHERE P.pid = C.pid AND C.sid...	1 row(s) returned	0.000 sec / 0.000 sec
15	11:57:18	SELECT DISTINCT C.sid FROM Catalog C WHERE C.cost> (SELECT AVG (C1.cost) FR...	4 row(s) returned	0.063 sec / 0.000 sec

```
supplier
Limit to 1000 rows

94      WHERE C.cost> ( SELECT AVG (C1.cost)
95      FROM Catalog C1 WHERE C1.pid = C.pid );
96
97
98      -- For each part, find the sname of the supplier who charges the most for that part.
99      SELECT P.pid, S.sname FROM Parts P, Suppliers S, Catalog C
100     WHERE C.pid = P.pid AND C.sid = S.sid
```

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

pid	sname
6	Rohan
2	Avni
5	Avni
3	Pratibha
1	Rahul
4	Rahul

Result 4 — Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
12	11:56:09	SELECT * FROM supplier.suppliers LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
13	11:56:39	select distinct parts.pname from parts,catalog where parts.pid = catalog.pid LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
14	11:57:00	SELECT P.pname FROM Parts P, Catalog C, Suppliers S WHERE P.pid = C.pid AND C.sid...	1 row(s) returned	0.000 sec / 0.000 sec
15	11:57:18	SELECT DISTINCT C.sid FROM Catalog C WHERE C.cost> (SELECT AVG (C1.cost) FR...	4 row(s) returned	0.063 sec / 0.000 sec
16	11:57:34	SELECT P.pid, S.sname FROM Parts P, Suppliers S, Catalog C WHERE C.pid = P.pid AN...	6 row(s) returned	0.016 sec / 0.000 sec

PROGRAM 4:

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).

SQL SCRIPT:

```
CREATE DATABASE STUDENT_FACULTY;
USE STUDENT_FACULTY;

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

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

insert into class values("class1", "12/11/15
10:15:16", "R1", 14);
insert into class values("class10", "12/11/15
10:15:16", "R128", 14);
```

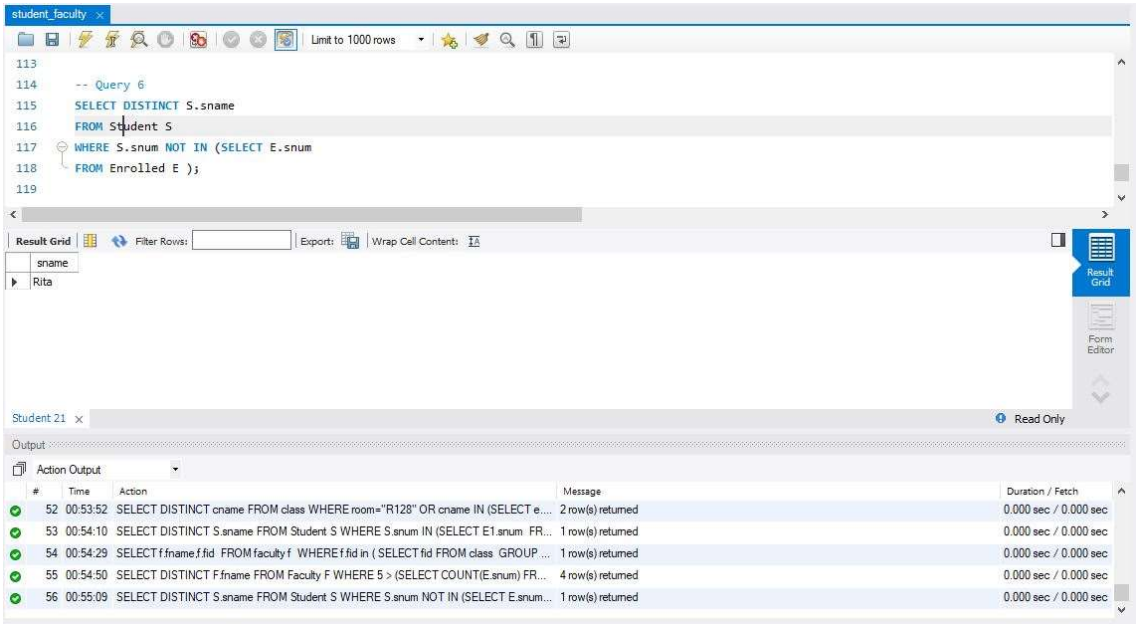
```
insert into class values("class2", "12/11/15
10:15:20", "R2", 12);
insert into class values("class3", "12/11/15
10:15:25", "R3", 11);
insert into class values("class4", "12/11/15
20:15:20", "R4", 14);
insert into class values("class5", "12/11/15
20:15:20", "R3", 15);
insert into class values("class6", "12/11/15
13:20:20", "R2", 14);
insert into class values("class7", "12/11/15
10:10:10", "R3", 14);

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

select * from STUDENT;
select * from FACULTY;
select * from class;
select * from enrolled;
```

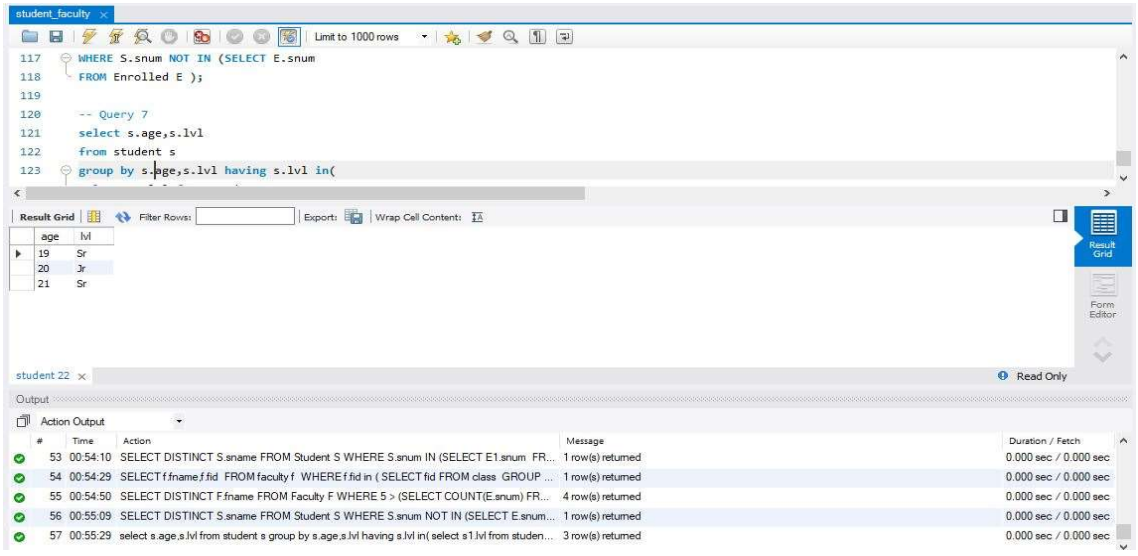
```
-- i. Find the names of all Juniors (level =
JR) who are enrolled in a class taught by
Harish
SELECT DISTINCT S.Sname
FROM Student S, Class C, Enrolled E,
Faculty F
WHERE S.snum = E.snum AND E.cname =
C.cname AND C.fid = F.fid AND
F.fname = "Harish" AND S.lvl = "Jr";


-- ii. Find the names of all classes that
either meet in room R128 or have five or
more Students enrolled.
SELECT DISTINCT cname
FROM class
WHERE room="R128"
OR
cname IN (SELECT e.cname FROM enrolled
e GROUP BY e.cname HAVING
COUNT(*)>=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
```

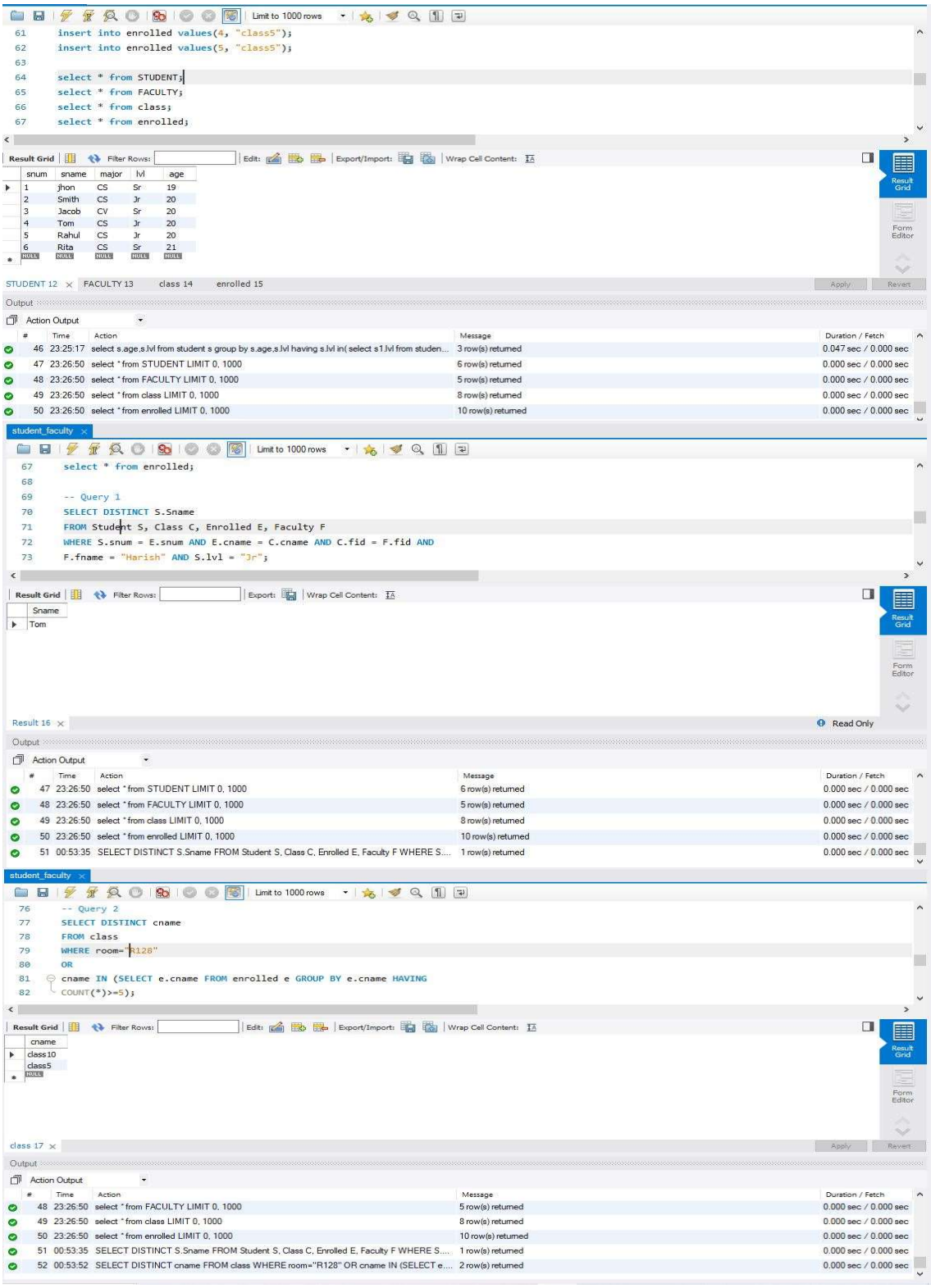
```
WHERE S.snum IN (SELECT E1.snum
FROM Enrolled E1, Enrolled E2, Class C1,
Class C2
WHERE E1.snum = E2.snum AND E1.cname
<> E2.cname
AND E1.cname = C1.cname
AND E2.cname = C2.cname AND
C1.metts_at = C2.metts_at);


-- iv. Find the names of faculty members
who teach in every room in which some
class is taught.
SELECT f.fname,f.fid
FROM faculty f
WHERE f.fid in ( SELECT fid FROM class
GROUP BY fid HAVING COUNT(*)=(SELECT
COUNT(DISTINCT
room) FROM class) );

-- Find the names of faculty members for
whom the combined enrolment 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(*)>=all(select count(*) from
student s2 where s1.age=s2.age
group by s2.lvl,s2.age));
```

OUTPUT:


student_faculty
Limit to 1000 rows
113
114 -- Query 6
115 SELECT DISTINCT S.sname
116 FROM Student S
117 WHERE S.snum NOT IN (SELECT E.snum
118 FROM Enrolled E);
119
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
sname
Rita
Result Grid
Form Editor
Student 21
Read Only
Output
Action Output
Time Action Message Duration / Fetch
52 00:53:52 SELECT DISTINCT cname FROM class WHERE room="R128" OR cname IN (SELECT e.... 2 row(s) returned 0.000 sec / 0.000 sec
53 00:54:10 SELECT DISTINCT S.sname FROM Student S WHERE S.snum IN (SELECT E1.snum FR... 1 row(s) returned 0.000 sec / 0.000 sec
54 00:54:29 SELECT f.fname,f.fid FROM faculty f WHERE f.fid in (SELECT fid FROM class GROUP ... 1 row(s) returned 0.000 sec / 0.000 sec
55 00:54:50 SELECT DISTINCT F.fname FROM Faculty F WHERE 5 > (SELECT COUNT(E.snum) FR... 4 row(s) returned 0.000 sec / 0.000 sec
56 00:55:09 SELECT DISTINCT S.sname FROM Student S WHERE S.snum NOT IN (SELECT E.snum... 1 row(s) returned 0.000 sec / 0.000 sec


student_faculty
Limit to 1000 rows
117 WHERE S.snum NOT IN (SELECT E.snum
118 FROM Enrolled E);
119
120 -- Query 7
121 select s.age,s.lvl
122 from student s
123 group by s.age,s.lvl having s.lvl in(
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
age lvl
19 Sr
20 Jr
21 Sr
Result Grid
Form Editor
student 22
Read Only
Output
Action Output
Time Action Message Duration / Fetch
53 00:54:10 SELECT DISTINCT S.sname FROM Student S WHERE S.snum IN (SELECT E1.snum FR... 1 row(s) returned 0.000 sec / 0.000 sec
54 00:54:29 SELECT f.fname,f.fid FROM faculty f WHERE f.fid in (SELECT fid FROM class GROUP ... 1 row(s) returned 0.000 sec / 0.000 sec
55 00:54:50 SELECT DISTINCT F.fname FROM Faculty F WHERE 5 > (SELECT COUNT(E.snum) FR... 4 row(s) returned 0.000 sec / 0.000 sec
56 00:55:09 SELECT DISTINCT S.sname FROM Student S WHERE S.snum NOT IN (SELECT E.snum... 1 row(s) returned 0.000 sec / 0.000 sec
57 00:55:29 select s.age,s.lvl from student s group by s.age,s.lvl having s.lvl in(select s1.lvl from studen... 3 row(s) returned 0.000 sec / 0.000 sec



```
insert into enrolled values(4, "class5");
insert into enrolled values(5, "class5");

select * from STUDENT;
select * from FACULTY;
select * from class;
select * from enrolled;
```

Result Grid

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

Output

Action Output

#	Time	Action	Message	Duration / Fetch
46	23:25:17	select s.age,s.lvl from student s group by s.age,s.lvl having s.lvl in(select s1.lvl from studen...	3 row(s) returned	0.047 sec / 0.000 sec
47	23:26:50	select * from STUDENT LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
48	23:26:50	select * from FACULTY LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
49	23:26:50	select * from class LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
50	23:26:50	select * from enrolled LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

student_faculty

```
select * from enrolled;

-- Query 1
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

Sname
Tom

Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
47	23:26:50	select * from STUDENT LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
48	23:26:50	select * from FACULTY LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
49	23:26:50	select * from class LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
50	23:26:50	select * from enrolled LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
51	00:53:35	SELECT DISTINCT S.Sname FROM Student S, Class C, Enrolled E, Faculty F WHERE S....	1 row(s) returned	0.000 sec / 0.000 sec

student_faculty

```
-- Query 2
SELECT DISTINCT cname
FROM class
WHERE room="R128"
OR
cname IN (SELECT e.cname FROM enrolled e GROUP BY e.cname HAVING
COUNT(*)>=5);
```

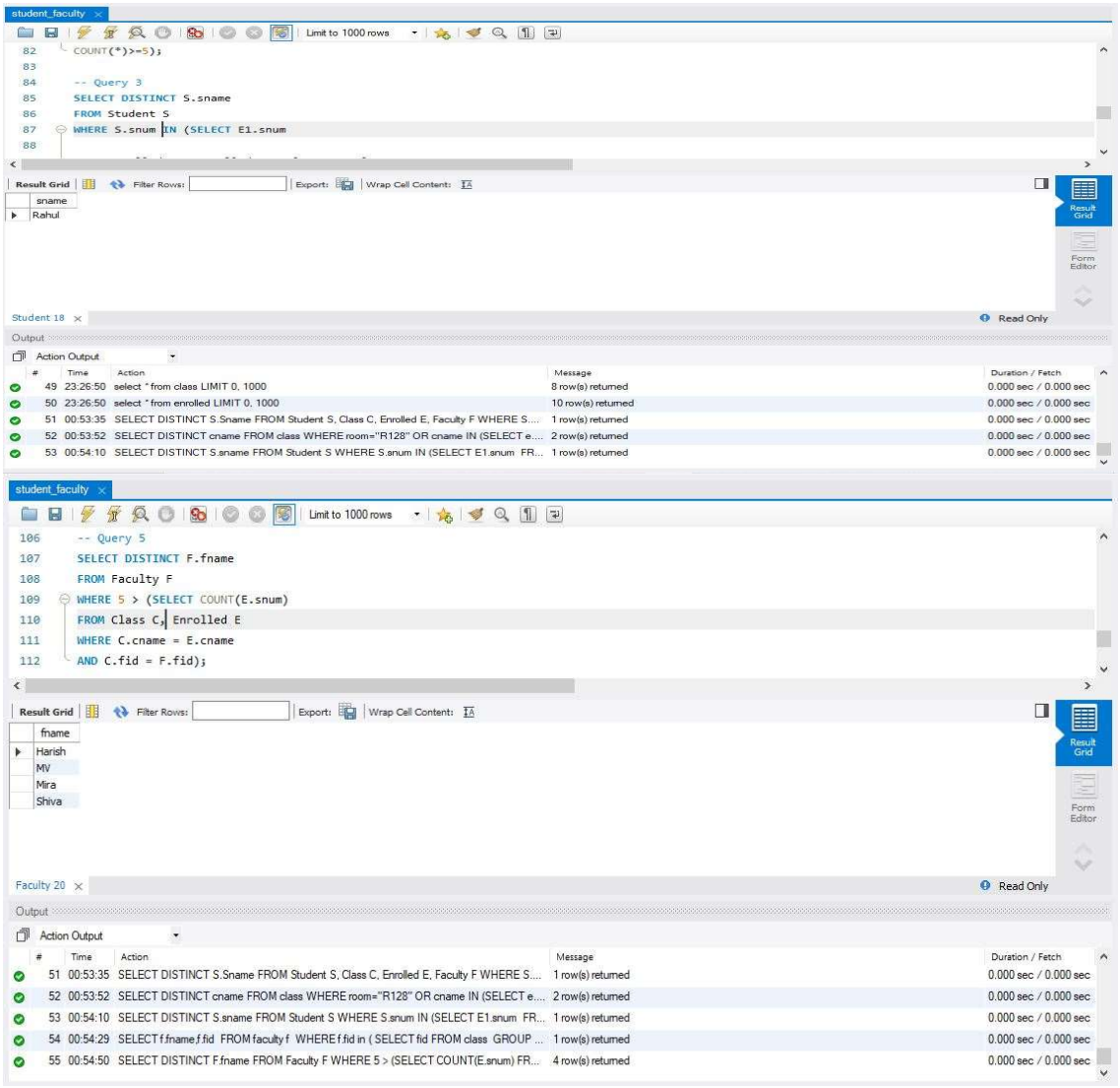
Result Grid

cname
class10
class5

Output

Action Output

#	Time	Action	Message	Duration / Fetch
48	23:26:50	select * from FACULTY LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
49	23:26:50	select * from class LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
50	23:26:50	select * from enrolled LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
51	00:53:35	SELECT DISTINCT S.Sname FROM Student S, Class C, Enrolled E, Faculty F WHERE S....	1 row(s) returned	0.000 sec / 0.000 sec
52	00:53:52	SELECT DISTINCT cname FROM class WHERE room="R128" OR cname IN (SELECT e....	2 row(s) returned	0.000 sec / 0.000 sec



PROGRAM 5:

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.

SQL SCRIPT:

```
--  Create the above tables by properly specifying the primary keys and the foreign keys.
--  Enter at least five tuples for each relation.
CREATE DATABASE AIRLINE;
USE AIRLINE;
-- SHOW DATABASES;
CREATE TABLE FLIGHTS(FL_NO INT,FFROM VARCHAR(30),FTO VARCHAR(30),DISTANCE INT,DEPARTS TIME,ARRIVES TIME,PRICE INT);
```

```
CREATE TABLE AIRCRAFT(AID INT,ANAME
VARCHAR(30),CRUISINGRANGE
INT,PRIMARY KEY(AID));

CREATE TABLE CERTIFIED(EID INT,AID INT,
                        FOREIGN KEY(EID)
REFERENCES EMPLOYEE(EID) ON UPDATE
CASCADE,
                         FOREIGN KEY(AID)
REFERENCES AIRCRAFT(AID) ON UPDATE
CASCADE);

CREATE TABLE EMPLOYEE(EID INT,ENAME
VARCHAR(30),SALARY INT,PRIMARY
KEY(EID));

INSERT INTO FLIGHTS (FL_NO, FFROM,
FTO, DISTANCE, DEPARTS, ARRIVES,
PRICE) VALUES

(1,'BANGALORE','MANGALORE',360,'10:45:0
0','12:00:00',10000),

(2,'BANGALORE','DELHI',5000,'12:15:00','0
4:30:00',25000),

(3,'BANGALORE','MUMBAI',3500,'02:15:00','
05:25:00',30000),
```

```
(4,'DELHI','MUMBAI',4500,'10:15:00','12:05
:00',35000),

(5,'DELHI','FRANKFURT',18000,'07:15:00','0
5:30:00',90000),

(6,'BANGALORE','FRANKFURT',19500,'10:00
:00','07:45:00',95000),

(7,'BANGALORE','FRANKFURT',17000,'12:00
:00','06:30:00',99000);

INSERT INTO FLIGHTS (FL_NO, FFROM,
FTO, DISTANCE, DEPARTS, ARRIVES,
PRICE) VALUES
        (8,'MADISON','NEW YORK', 19000,
'10:00:00', '17:00:00', 100000),
        (9,'MADISON','NEW YORK', 29000,
'10:00:00', '18:30:00', 100000),
        (10,'MADISON','LONDON', 30000,
'11:00:00', '14:00:00', 55000),
        (10,'LONDON','NEW YORK', 30000,
'14:05:00', '17:50:00', 50000),
        (11,'LONDON','NEW YORK', 31000,
'14:06:00', '18:05:00', 51000),
        (11,'LONDON','BERLIN', 15000,
'14:06:00', '16:05:00', 17000),
```

```
        (11,'BERLIN','NEW YORK', 18000,
'16:06:00', '17:59:00', 17401);


INSERT INTO AIRCRAFT (AID, ANAME,
CRUISINGRANGE) VALUES
            (123,'AIRBUS',1000),
            (302,'BOEING',5000),
            (306,'JET01',5000),
            (378,'AIRBUS380',8000),
            (456,'AIRCRAFT',500),
            (789,'AIRCRAFT02',800),
            (951,'AIRCRAFT03',1000);

INSERT INTO EMPLOYEE (EID, ENAME,
SALARY) VALUES
            (1,'AJAY',30000),
            (2,'AJITH',85000),
            (3,'ARNAB',50000),
            (4,'HARRY',45000),
            (5,'RON',90000),
            (6,'JOSH',75000),
            (7,'RAM',100000);

INSERT INTO EMPLOYEE (EID, ENAME,
SALARY) VALUES
            (8,'RAMESH',70000),
            (9,'SURESH',80000);
```

```
INSERT INTO CERTIFIED (EID, AID) VALUES
                (1,123),
                (2,123),
                (1,302),
                (5,302),
                (7,302),
                (1,306),
                (2,306),
                (1,378),
                (2,378),
                (4,378),
                (6,456),
                (3,456),
                (5,789),
                (6,789),
                (3,951),
                (1,951),
                (1,789);

-- 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,CERTIFIED C,EMPLOYEE E
```

```
WHERE A.AID=C.AID AND C.EID=E.EID
AND E.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 E.EID,MAX(A.CRUISINGRANGE)
FROM AIRCRAFT A,CERTIFIED C,EMPLOYEE
E WHERE A.AID=C.AID AND C.EID=E.EID
GROUP BY E.EID HAVING COUNT(E.EID)>3;

-- iii. Find the names of pilots whose salary
is less than the price of the cheapest route
from Bengaluru to
-- Frankfurt.
SELECT ENAME FROM EMPLOYEE WHERE
EID IN(SELECT EID FROM CERTIFIED) AND
SALARY<(SELECT MIN(PRICE) FROM
FLIGHTS WHERE FFROM="BANGALORE"
AND FTO="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.
```

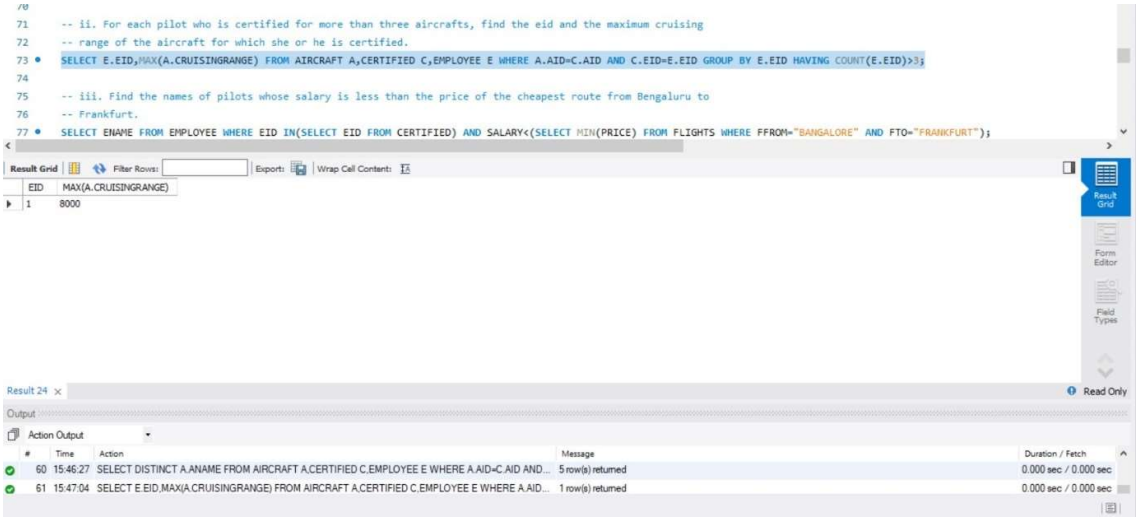
```
SELECT A.ANAME,AVG(E.SALARY) FROM
AIRCRAFT A,CERTIFIED C,EMPLOYEE E
WHERE A.AID=C.AID AND C.EID=E.EID
AND A.CRUISINGRANGE>1000 GROUP BY
A.ANAME;

-- v. Find the names of pilots certified for
some Boeing aircraft.
SELECT E.ENAME FROM AIRCRAFT
A,CERTIFIED C,EMPLOYEE E WHERE
A.AID=C.AID AND C.EID=E.EID AND
A.ANAME="BOEING";

-- vi. Find the aids of all aircraft that can be
used on routes from Bengaluru to New
Delhi.
SELECT AID FROM AIRCRAFT WHERE
CRUISINGRANGE>=(SELECT DISTANCE
FROM FLIGHTS  WHERE
FFROM="BANGALORE" AND FTO="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.FL_NO IN ( ( SELECT F0.FL_NO
 FROM FLIGHTS F0
 WHERE F0.FFROM = 'MADISON' AND
F0.FTO = 'NEW YORK'
 AND F0.ARRIVES < '18:00:00' )
 UNION
 ( SELECT F0.FL_NO
 FROM FLIGHTS F0, FLIGHTS F1
 WHERE F0.FFROM = 'MADISON' AND
F0.FTO != 'NEW YORK'
 AND F0.FTO = F1.FFROM AND F1.FTO =
'NEW YORK'
 AND F1.DEPARTS > F0.ARRIVES
 AND F1.ARRIVES < '18:00:00' )
 UNION
 ( SELECT F0.FL_NO
 FROM FLIGHTS F0, FLIGHTS F1, FLIGHTS
F2
 WHERE F0.FFROM = 'MADISON'
 AND F0.FTO = F1.FFROM
 AND F1.FTO = F2.FFROM
 AND F2.FTO = 'NEW YORK'
 AND F0.FTO != 'NEW YORK'
 AND F1.FTO != 'NEW YORK'
 AND F1.DEPARTS > F0.ARRIVES
 AND F2.DEPARTS > F1.ARRIVES
 AND F2.ARRIVES < '18:00:00' ));
```

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

```
SELECT ENAME FROM EMPLOYEE WHERE EID NOT IN(SELECT EID FROM CERTIFIED) AND SALARY>(SELECT AVG(SALARY) FROM EMPLOYEE WHERE EID IN(SELECT EID FROM CERTIFIED));
```

OUTPUT:



```
74
75      -- iii. Find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru to
76      -- Frankfurt.
77 •    SELECT ENAME FROM EMPLOYEE WHERE EID IN(SELECT EID FROM CERTIFIED) AND SALARY<(SELECT MIN(PRICE) FROM FLIGHTS WHERE FFROM="BANGALORE" AND FTO="FRANKFURT");
78
79      -- iv. For all aircraft with cruising range over 1000 Kms, find the name of the aircraft and the average
80      -- salary of all pilots certified for this aircraft.
81 •    SELECT A.ANAME,AVG(E.SALARY) FROM AIRCRAFT A,CERTIFIED C,EMPLOYEE E WHERE A.AID=C.AID AND C.EID=E.EID AND A.CRUISINGRANGE>1000 GROUP BY A.ANAME;
82
83      -- v. Find the names of pilots certified for some Boeing aircraft.
```

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

ENAME
AJAY
AJITH
ARNAB
HARRY
RON
JOSH

EMPLOYEE 25 × Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
61	15:47:04	SELECT E.EID,MAX(A.CRUISINGRANGE) FROM AIRCRAFT A,CERTIFIED C,EMPLOYEE E WHERE A.AID...	1 row(s) returned	0.000 sec / 0.000 sec
62	15:47:34	SELECT ENAME FROM EMPLOYEE WHERE EID IN(SELECT EID FROM CERTIFIED) AND SALARY<(SEL...	6 row(s) returned	0.000 sec / 0.000 sec

```
79      -- iv. For all aircraft with cruising range over 1000 Kms, find the name of the aircraft and the average
80      -- salary of all pilots certified for this aircraft.
81 •    SELECT A.ANAME,AVG(E.SALARY) FROM AIRCRAFT A,CERTIFIED C,EMPLOYEE E WHERE A.AID=C.AID AND C.EID=E.EID AND A.CRUISINGRANGE>1000 GROUP BY A.ANAME;
82
83      -- v. Find the names of pilots certified for some Boeing aircraft.
84 •    SELECT E.ENAME FROM AIRCRAFT A,CERTIFIED C,EMPLOYEE E WHERE A.AID=C.AID AND C.EID=E.EID AND A.ANAME="BOEING";
85
86      -- vi. Find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi.
```

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

ANAME	AVG(E.SALARY)
BOEING	73333.3333
JET01	57500.0000
AIRBUS380	53333.3333

Result 26 × Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
62	15:47:34	SELECT ENAME FROM EMPLOYEE WHERE EID IN(SELECT EID FROM CERTIFIED) AND SALARY<(SEL...	6 row(s) returned	0.000 sec / 0.000 sec
63	15:47:58	SELECT A.ANAME,AVG(E.SALARY) FROM AIRCRAFT A,CERTIFIED C,EMPLOYEE E WHERE A.AID=C.AI...	3 row(s) returned	0.000 sec / 0.000 sec

```
83      -- v. Find the names of pilots certified for some Boeing aircraft.
84 •    SELECT E.ENAME FROM AIRCRAFT A,CERTIFIED C,EMPLOYEE E WHERE A.AID=C.AID AND C.EID=E.EID AND A.ANAME="BOEING";
85
86      -- vi. Find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi.
87 •    SELECT AID FROM AIRCRAFT WHERE CRUISINGRANGE>=(SELECT DISTANCE FROM FLIGHTS  WHERE FFROM="BANGALORE" AND FTO="DELHI");
88
89      -- vii. A customer wants to travel from Madison to New York with no more than two changes of flight. List
```

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

ENAME
AJAY
RON
RAM

Result 27 × Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
63	15:47:58	SELECT A.ANAME,AVG(E.SALARY) FROM AIRCRAFT A,CERTIFIED C,EMPLOYEE E WHERE A.AID=C.AI...	3 row(s) returned	0.000 sec / 0.000 sec
64	15:48:19	SELECT E.ENAME FROM AIRCRAFT A,CERTIFIED C,EMPLOYEE E WHERE A.AID=C.AID AND C.EID=E.E...	3 row(s) returned	0.000 sec / 0.000 sec

```
85
86      -- vi. Find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi.
87 •    SELECT AID FROM AIRCRAFT WHERE CRUISINGRANGE>=(SELECT DISTANCE FROM FLIGHTS  WHERE FFROM="BANGALORE" AND FTO="DELHI");
88
89      -- vii. A customer wants to travel from Madison to New York with no more than two changes of flight. List
90      -- the choice of departure times from Madison if the customer wants to arrive in New York by 6 p.m.
91 •    SELECT F.DEPARTS
92      FROM FLIGHTS F
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

AID
302
306
378
NULL

AIRCRAFT 28 ×

Output

Action Output

#	Time	Action	Message	Duration / Fetch
64	15:48:19	SELECT E.ENAME FROM AIRCRAFT A,CERTIFIED C,EMPLOYEE E WHERE A.AID=C.AID AND C.EID=E.E...	3 row(s) returned	0.000 sec / 0.000 sec
65	15:48:40	SELECT AID FROM AIRCRAFT WHERE CRUISINGRANGE>=(SELECT DISTANCE FROM FLIGHTS WHE...	3 row(s) returned	0.000 sec / 0.000 sec

```
116
117     -- viii. Print the name and salary of every non-pilot whose salary is more than the average salary for pilots.
118 •   SELECT ENAME FROM EMPLOYEE WHERE EID NOT IN(SELECT EID FROM CERTIFIED) AND SALARY>(SELECT AVG(SALARY) FROM EMPLOYEE WHERE EID IN(SELECT EID FROM CERTIFIED));
119
120
```

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

ENAME
RAMESH
SURESH

EMPLOYEE 29 × Read Only

Output

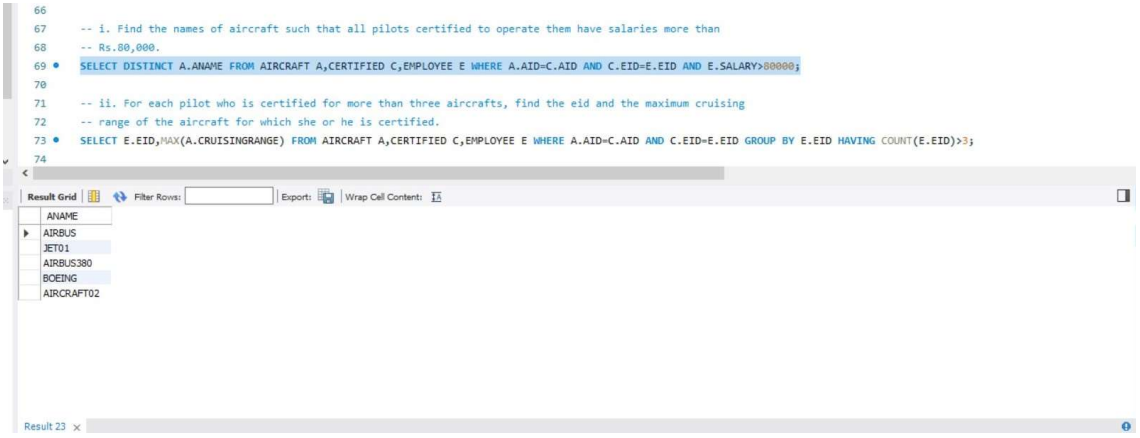
Action Output

#	Time	Action	Message	Duration / Fetch
65	15:48:40	SELECT AID FROM AIRCRAFT WHERE CRUISINGRANGE>=(SELECT DISTANCE FROM FLIGHTS WHE...	3 row(s) returned	0.000 sec / 0.000 sec
66	15:49:05	SELECT ENAME FROM EMPLOYEE WHERE EID NOT IN(SELECT EID FROM CERTIFIED) AND SALARY>...	2 row(s) returned	0.000 sec / 0.000 sec

```
101 •   SELECT F.DEPARTS
102     FROM FLIGHTS F
103     WHERE F.Fl_NO IN ( ( SELECT F0.Fl_NO
104      FROM FLIGHTS F0
105      WHERE F0.FFROM = 'MADISON' AND F0.FTO = 'NEW YORK'
106      AND F0.ARRIVES < '18:00:00' )
107      UNION
108      ( SELECT F0.Fl_NO
109      FROM FLIGHTS F0, FLIGHTS F1
110      WHERE F0.FFROM = 'MADISON' AND F0.FTO != 'NEW YORK'
111      AND F0.FTO = F1.FFROM AND F1.FTO = 'NEW YORK'
112      AND F1.DEPARTS > F0.ARRIVES
113      AND F1.ARRIVES < '18:00:00' )
114      UNION
115      ( SELECT F0.Fl_NO
116      FROM FLIGHTS F0, FLIGHTS F1, FLIGHTS F2
117      WHERE F0.FFROM = 'MADISON'
118      AND F0.FTO = F1.FFROM
119      AND F1.FTO = F2.FFROM
120      AND F2.FTO = 'NEW YORK'
121      AND F0.FTO != 'NEW YORK'
122      AND F1.FTO != 'NEW YORK'
123      AND F1.DEPARTS > F0.ARRIVES
124      AND F2.DEPARTS > F1.ARRIVES
125      AND F2.ARRIVES < '18:00:00' ));
126
127     -- viii. Print the name and salary of every non-pilot whose salary is more than the average salary for pilots.
```

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

DEPARTS
10:00:00
11:00:00
14:05:00



```
66
67      -- i. Find the names of aircraft such that all pilots certified to operate them have salaries more than
68      -- Rs.80,000.
69 •    SELECT DISTINCT A.ANAME FROM AIRCRAFT A,CERTIFIED C,EMPLOYEE E WHERE A.AID=C.AID AND C.EID=E.EID AND E.SALARY>80000;
70
71      -- ii. For each pilot who is certified for more than three aircrafts, find the eid and the maximum cruising
72      -- range of the aircraft for which she or he is certified.
73 •    SELECT E.EID,MAX(A.CRUISINGRANGE) FROM AIRCRAFT A,CERTIFIED C,EMPLOYEE E WHERE A.AID=C.AID AND C.EID=E.EID GROUP BY E.EID HAVING COUNT(E.EID)>3;
74
```

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

ANAME
AIRBUS
JET01
AIRBUS380
BOEING
AIRCRAFT02

Result 23

PROGRAM 6:

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)

v. 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.

SQL SCRIPT:

```
-- i. Create the above tables by properly specifying the
primary keys and the foreign keys.
-- ii. Enter at least five tuples for each relation.
CREATE DATABASE ORDER_PROCESSING;
USE ORDER_PROCESSING;
-- SHOW DATABASES;
CREATE TABLE CUSTOMER(CUST_ID INT,CNAME VARCHAR(30),CITY
VARCHAR(30),PRIMARY KEY(CUST_ID));
CREATE TABLE ORDERS(ORDER_ID INT,ORDER_DATE DATE,CUST_ID
INT,ORDER_AMOUNT INT,PRIMARY KEY(ORDER_ID),
                            FOREIGN KEY(CUST_ID)
REFERENCES CUSTOMER(CUST_ID) ON DELETE CASCADE);
```

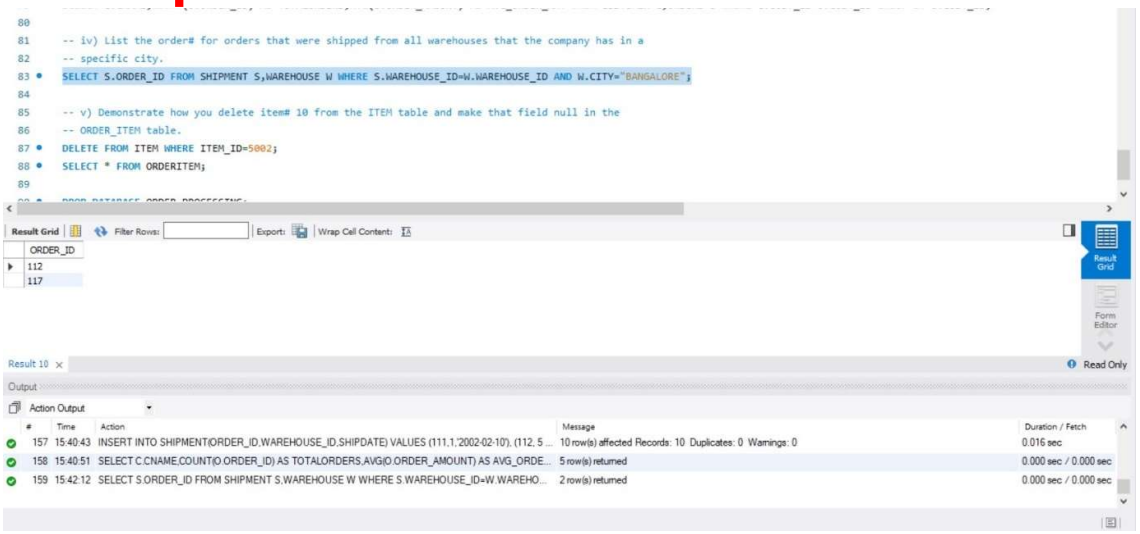
```
CREATE TABLE ITEM(ITEM_ID INT,UNITPRICE INT,PRIMARY
KEY(ITEM_ID));
-- SHOW TABLES;
CREATE TABLE ORDERITEM(ORDER_ID INT,ITEM_ID INT,QTY INT,
                  FOREIGN KEY(ORDER_ID) REFERENCES
ORDERS(ORDER_ID) ON DELETE SET NULL,
                  FOREIGN KEY(ITEM_ID) REFERENCES
ITEM(ITEM_ID) ON DELETE SET NULL);
CREATE TABLE WAREHOUSE(WAREHOUSE_ID INT,CITY
VARCHAR(30),PRIMARY KEY(WAREHOUSE_ID));
CREATE TABLE SHIPMENT(ORDER_ID INT,WAREHOUSE_ID
INT,SHIPDATE DATE,
                        FOREIGN KEY(ORDER_ID) REFERENCES
ORDERS(ORDER_ID) ON DELETE CASCADE,
                  FOREIGN KEY(WAREHOUSE_ID) REFERENCES
WAREHOUSE(WAREHOUSE_ID) ON DELETE CASCADE);


INSERT INTO CUSTOMER(CUST_ID,CNAME,CITY) VALUES (771,
‘PUSHPA K’,’BANGALORE’),
                                (772, ‘SUMAN’,
‘MUMBAI’),

(773,’SOURAV’,’CALICUT’),
                                (774, ‘LAILA’,
‘HYDERABAD’),
                                (775, ‘FAIZAL’,’
BANGALORE’);
INSERT INTO
ORDERS(ORDER_ID,ORDER_DATE,CUST_ID,ORDER_AMOUNT) VALUES
(111, ‘2002-01-22’, 771, 18000),
                                                (112,
‘2002-07-30’, 774, 6000),
                                                (113,
‘2003-04-03’, 775, 9000),
                                                (114,
‘2003-11-03’, 775, 29000),
                                                  (115,
‘2003-12-10’, 773, 29000),

(116, ‘2004-08-19’, 772, 56000),
                                                (117,
‘2004-09-10’, 771, 20000),
                                                (118,
‘2004-11-20’,775, 29000),
```

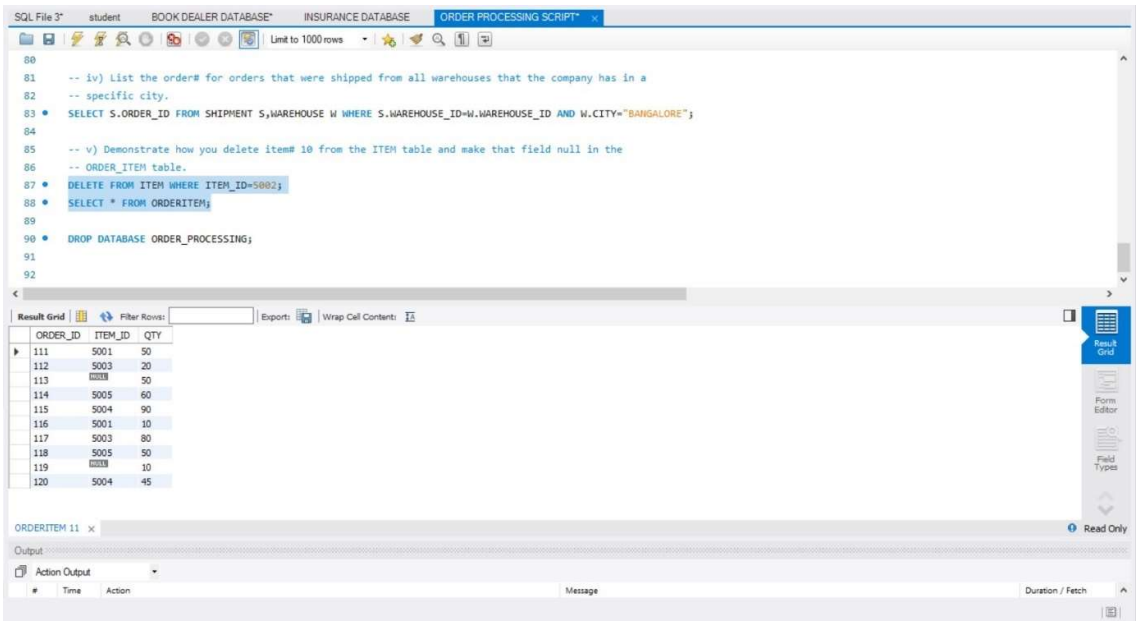
```
                                                    (119,
        ‘2005-02-13’, 774, 29000),
                                                    (120,
        ‘2005-10-13’, 775 ,29000);

        INSERT INTO ITEM(ITEM_ID, UNITPRICE) VALUES (5001, 503),
                                                    (5002, 750),
                                                    (5003, 150),
                                                    (5004, 600),
                                                    (5005, 890);

        INSERT INTO ORDERITEM(ORDER_ID,ITEM_ID,QTY) VALUES
        (111, 5001, 50),

        (112, 5003, 20),

        (113, 5002, 50),

        (114, 5005, 60),

        (115, 5004, 90),

        (116, 5001, 10),

        (117, 5003, 80),

        (118, 5005, 50),

        (119, 5002, 10),

        (120, 5004, 45);

        INSERT INTO WAREHOUSE(WAREHOUSE_ID,CITY) VALUES
        (1,’DELHI’),

        (2,’BOMBAY’),

        (3,’CHENNAI’),

        (4,’BANGALORE’),

        (5,’BANGALORE’),

        (6,’DELHI’),

        (7,’BOMBAY’),
```

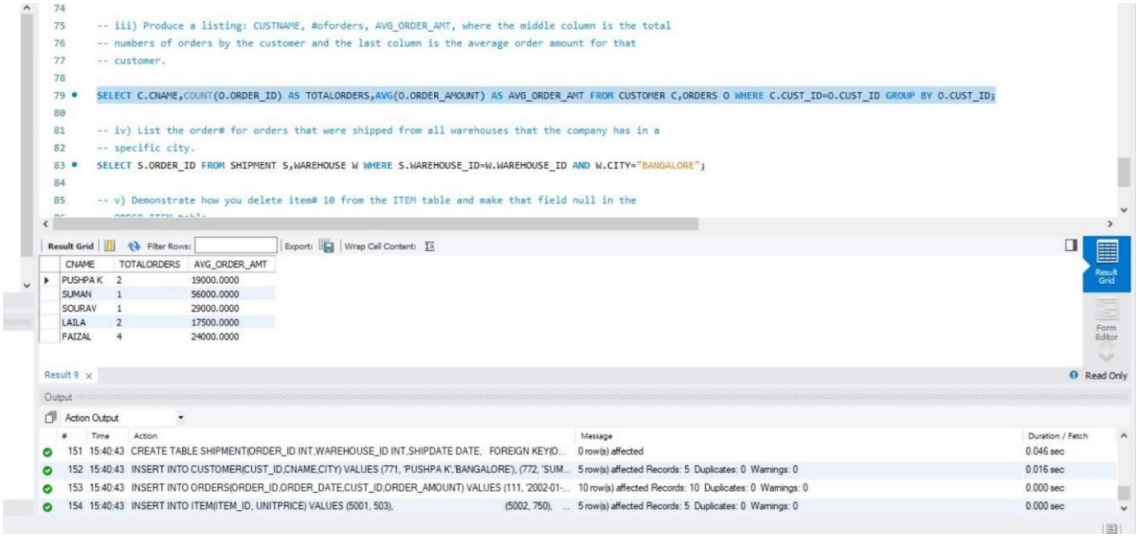
```
(8,’CHENNAI’),

(9,’DELHI’),

(10,’BANGALORE’);

INSERT INTO SHIPMENT(ORDER_ID,WAREHOUSE_ID,SHIPDATE)
VALUES (111,1,’2002-02-10’),
                                        (112, 5
,’2002-09-10’),
                                        (113, 8
,’2003-02-10’),
                                        (114, 3
,’2003-12-10’),
                                          (115,9
,’2004-01-19’),
                                        (116, 1,
‘2004-09-20’),
                                          (117, 5
,’2004-0910’),
                                        (118, 7
,’2004-11-30’),
                                        (119, 7
,’2005-04-30’),
                                        (120, 6
,’2005-12-21’);

-- 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.

SELECT C.CNAME,COUNT(O.ORDER_ID) AS
TOTALORDERS,AVG(O.ORDER_AMOUNT) AS AVG_ORDER_AMT FROM
CUSTOMER C,ORDERS O WHERE C.CUST_ID=O.CUST_ID GROUP BY
O.CUST_ID;

-- iv) List the order# for orders that were shipped from
all warehouses that the company has in a
-- specific city.
SELECT S.ORDER_ID FROM SHIPMENT S,WAREHOUSE W WHERE
S.WAREHOUSE_ID=W.WAREHOUSE_ID AND W.CITY=”BANGALORE”;
```

```
-- v) Demonstrate how you delete item# 10 from the ITEM
table and make that field null in the
-- ORDER_ITEM table.
DELETE FROM ITEM WHERE ITEM_ID=5002;
SELECT * FROM ORDERITEM;
```

Output:







PROGRAM 7:

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)

v. 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%.

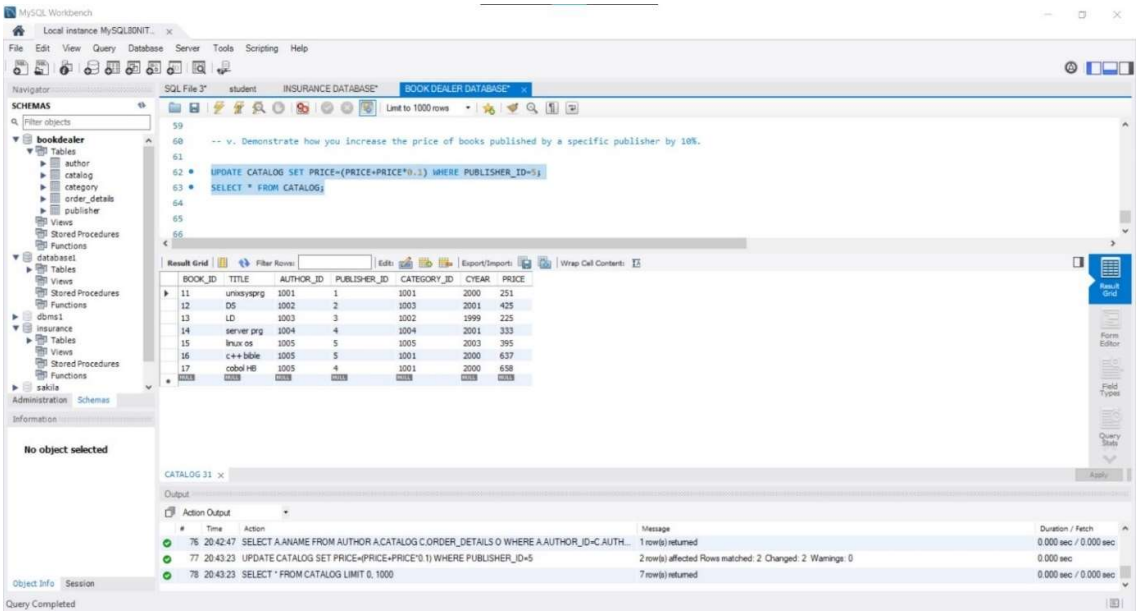
SQL SCRIPT:

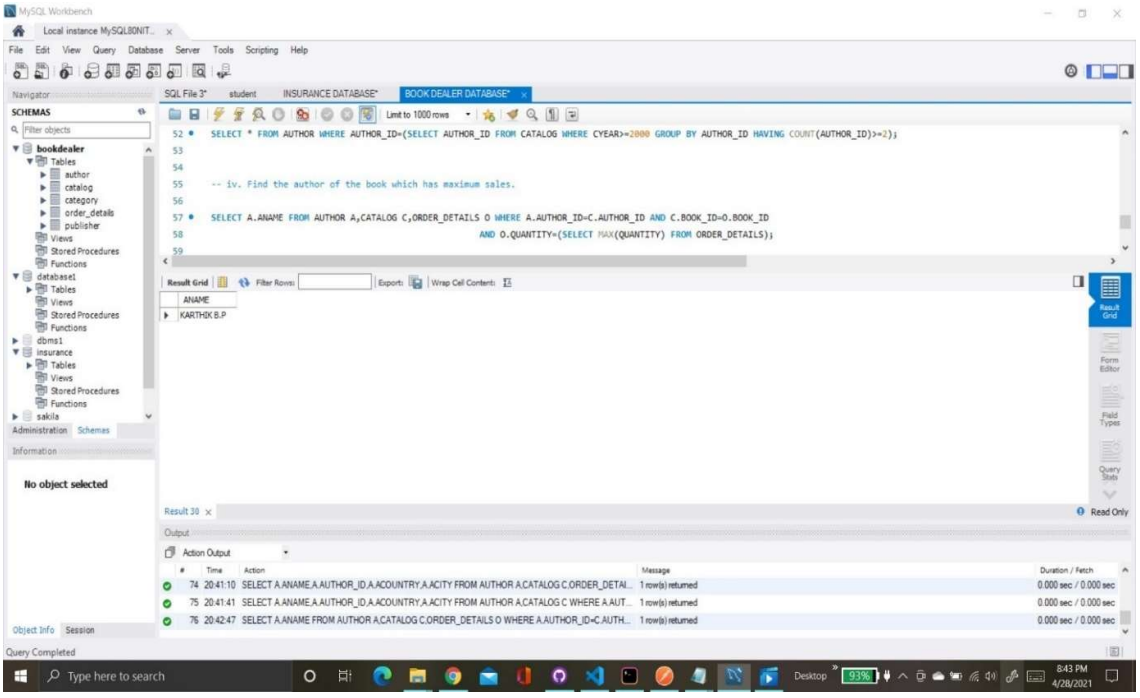
```
-- i. Create the above tables by properly specifying the
primary keys and the foreign keys.
-- ii. Enter at least five tuples for each relation.
CREATE DATABASE BOOKDEALER;
USE BOOKDEALER;
SHOW DATABASES;
CREATE TABLE AUTHOR(AUTHOR_ID INT,ANAME VARCHAR(30),ACITY
VARCHAR(30),ACOUNTRY VARCHAR(30),PRIMARY KEY(AUTHOR_ID));
CREATE TABLE PUBLISHER(PUBLISHER_ID INT,PNAME
VARCHAR(20),PCITY VARCHAR(30),PCOUNTRY VARCHAR(30),PRIMARY
KEY(PUBLISHER_ID));
CREATE TABLE CATALOG(BOOK_ID INT,TITLE VARCHAR(30),AUTHOR_ID
INT,PUBLISHER_ID INT,CATEGORY_ID INT,CYEAR INT,PRICE
INT,PRIMARY KEY(BOOK_ID),

FOREIGN KEY(AUTHOR_ID) REFERENCES AUTHOR(AUTHOR_ID) ON
DELETE CASCADE,

FOREIGN KEY(PUBLISHER_ID) REFERENCES PUBLISHER(PUBLISHER_ID)
ON DELETE CASCADE,

                                    FOREIGN KEY(CATEGORY_ID)
REFERENCES CATEGORY(CATEGORY_ID) ON DELETE CASCADE);
-- SHOW TABLES;
CREATE TABLE CATEGORY(CATEGORY_ID INT,DESCRIPTION
VARCHAR(30),PRIMARY KEY(CATEGORY_ID));
```

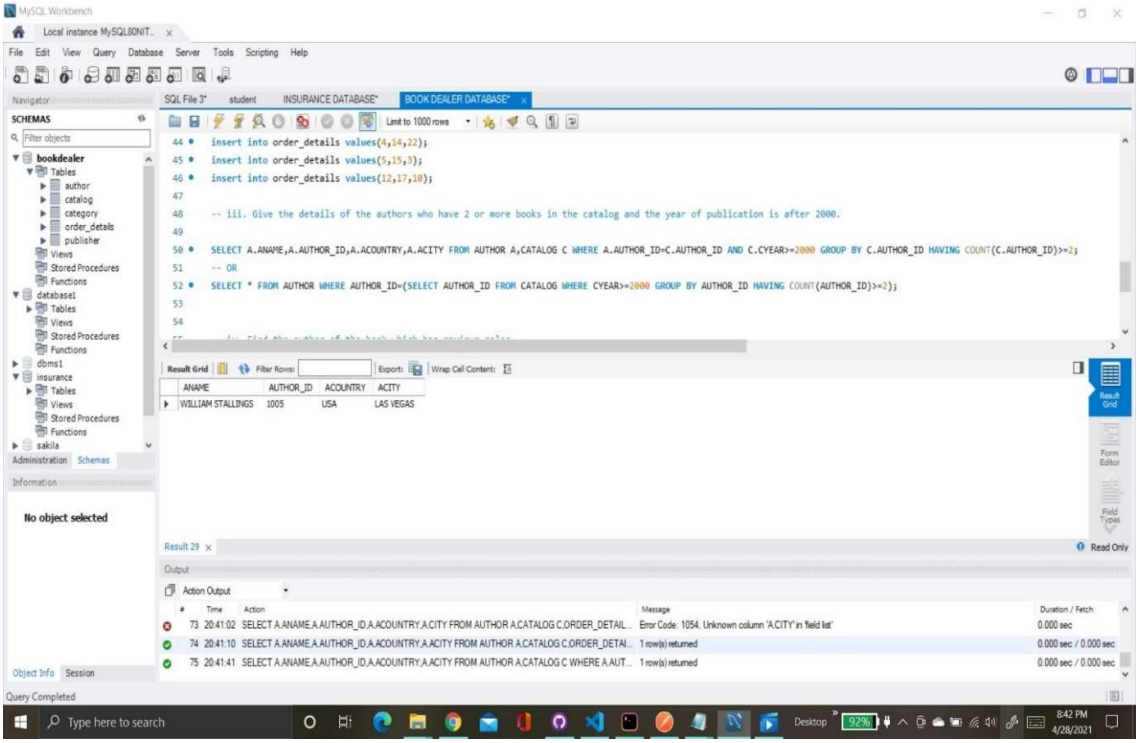
```
CREATE TABLE ORDER_DETAILS(ORDER_NO INT,BOOK_ID INT,QUANTITY
INT,PRIMARY KEY(ORDER_NO),
                          FOREIGN KEY(BOOK_ID) REFERENCES
CATALOG(BOOK_ID) ON DELETE CASCADE);
INSERT INTO AUTHOR(AUTHOR_ID,ANAME,ACITY,ACOUNTRY) VALUES
(1001,'TERAS CHAN', 'CA','USA'),
                                        (1002,'STEVENS',
'ZOMBI','UGANDA'),

(1003,'M MANO', 'CAIR','CANADA'),
                                        (1004,'KARTHIK B.P',
'NEW YORK','USA'),

(1005,'WILLIAM STALLINGS', 'LAS VEGAS','USA');
INSERT INTO PUBLISHER(PUBLISHER_ID,PNAME,PCITY,PCOUNTRY)
VALUES (1,'PEARSON', 'NEW YORK','USA'),
                                              (2,'EEE','NEW
SOUTH WALES','USA'),

     (3,'PHI','DELHI','INDIA'),
                                              (4,'WILLEY',
'BERLIN','GERMANY'),
                                                   (5,'MGH',
'NEW YORK','USA');

insert into category values(1001,'CSE');
insert into category values(1002,'ADA');
insert into category values(1003,'ECE');
insert into category values(1004,'PROGRAMING');
insert into category values(1005,'OS');

insert into catalog values(11,'unixsysprg',1001,1,1001,2000
,251);
insert into catalog values(12,'DS',1002,2,1003, 2001 ,425);
insert into catalog values(13,'LD',1003,3,1002, 1999 ,225);
insert into catalog values(14,'server prg',1004,4,1004, 2001
,333);
insert into catalog values(15,'linux os',1005,5,1005, 2003
,326);
insert into catalog values(16,'c++ bible',1005,5,1001, 2000
,526);
insert into catalog values(17,'cobol HB',1005,4,1001, 2000
,658);

insert into order_details values(1,11,5);
insert into order_details values(2,12,8);
```

```
insert into order_details values(3,13,15);
insert into order_details values(4,14,22);
insert into order_details values(5,15,3);
insert into order_details values(12,17,10);

-- iii. Give the details of the authors who have 2 or more
books in the catalog and the year of publication is after
2000.

SELECT A.ANAME,A.AUTHOR_ID,A.ACOUNTRY,A.ACITY FROM AUTHOR
A,CATALOG C WHERE A.AUTHOR_ID=C.AUTHOR_ID AND C.CYEAR>=2000
GROUP BY C.AUTHOR_ID HAVING COUNT(C.AUTHOR_ID)>=2;
-- OR
SELECT * FROM AUTHOR WHERE AUTHOR_ID=(SELECT AUTHOR_ID FROM
CATALOG WHERE CYEAR>=2000 GROUP BY AUTHOR_ID HAVING
COUNT(AUTHOR_ID)>=2);


-- iv. Find the author of the book which has maximum sales.

SELECT A.ANAME FROM AUTHOR A,CATALOG C,ORDER_DETAILS O WHERE
A.AUTHOR_ID=C.AUTHOR_ID AND C.BOOK_ID=O.BOOK_ID
                                                    AND
O.QUANTITY=(SELECT MAX(QUANTITY) FROM ORDER_DETAILS);

-- v. Demonstrate how you increase the price of books
published by a specific publisher by 10%.

UPDATE CATALOG SET PRICE=(PRICE+PRICE*0.1) WHERE
PUBLISHER_ID=5;
SELECT * FROM CATALOG;
```

Output:


MySQL Workbench
Local instance MySQL80NIT...
File Edit View Query Database Server Tools Scripting Help
SQL File 3* student INSURANCE DATABASE* BOOK DEALER DATABASE*
-- v. Demonstrate how you increase the price of books published by a specific publisher by 10%.
UPDATE CATALOG SET PRICE=(PRICE+PRICE*0.1) WHERE PUBLISHER_ID=5;
SELECT * FROM CATALOG;
BOOK_ID TITLE AUTHOR_ID PUBLISHER_ID CATEGORY_ID CYEAR PRICE
11 unixsysprg 1001 1 1001 2000 251
12 DS 1002 2 1003 2001 425
13 LD 1003 3 1002 1999 225
14 server prg 1004 4 1004 2001 333
15 linux os 1005 5 1005 2003 395
16 c++ bible 1005 5 1001 2000 637
17 cobol HB 1005 4 1001 2000 658
No object selected
Action Output
76 20:42:47 SELECT A.ANAME FROM AUTHOR A,CATALOG C,ORDER_DETAILS O WHERE A.AUTHOR_ID=C.AUTH... 1 row(s) returned 0.000 sec / 0.000 sec
77 20:43:23 UPDATE CATALOG SET PRICE=(PRICE+PRICE*0.1) WHERE PUBLISHER_ID=5 2 row(s) affected Rows matched: 2 Changed: 2 Warnings: 0 0.000 sec
78 20:43:23 SELECT * FROM CATALOG LIMIT 0, 1000 7 row(s) returned 0.000 sec / 0.000 sec
Query Completed


MySQL Workbench
Local instance MySQL80NIT...
File Edit View Query Database Server Tools Scripting Help
SQL File 3* student INSURANCE DATABASE* BOOK DEALER DATABASE*
SELECT * FROM AUTHOR WHERE AUTHOR_ID=(SELECT AUTHOR_ID FROM CATALOG WHERE CYEAR>=2000 GROUP BY AUTHOR_ID HAVING COUNT(AUTHOR_ID)>=2);
-- iv. Find the author of the book which has maximum sales.
SELECT A.ANAME FROM AUTHOR A,CATALOG C,ORDER_DETAILS O WHERE A.AUTHOR_ID=C.AUTHOR_ID AND C.BOOK_ID=O.BOOK_ID
AND O.QUANTITY=(SELECT MAX(QUANTITY) FROM ORDER_DETAILS);
ANAME
KARTHIK B.P
No object selected
Action Output
74 20:41:10 SELECT A.ANAME,A.AUTHOR_ID,A.ACOUNTRY,A.ACITY FROM AUTHOR A,CATALOG C,ORDER_DETAI... 1 row(s) returned 0.000 sec / 0.000 sec
75 20:41:41 SELECT A.ANAME,A.AUTHOR_ID,A.ACOUNTRY,A.ACITY FROM AUTHOR A,CATALOG C WHERE A.AUT... 1 row(s) returned 0.000 sec / 0.000 sec
76 20:42:47 SELECT A.ANAME FROM AUTHOR A,CATALOG C,ORDER_DETAILS O WHERE A.AUTHOR_ID=C.AUTH... 1 row(s) returned 0.000 sec / 0.000 sec
Query Completed
Type here to search
8:43 PM
4/28/2021



PROGRAM 8:

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.

SQL SCRIPT:

```
-- i. Create the above tables by properly specifying the
primary keys and the foreign keys
-- ii. Enter at least five tuples for each relation.
CREATE DATABASE STUDENT_ENROLLMENT;
USE STUDENT_ENROLLMENT;
-- SHOW DATABASES;
```

```
CREATE TABLE STUDENT(REG_NO VARCHAR(30),SNAME
VARCHAR(30),MAJOR VARCHAR(30),BDATE DATE,PRIMARY
KEY(REG_NO));

CREATE TABLE COURSE(COURSE_ID INT,CNAME VARCHAR(30),DEPT
VARCHAR(30),PRIMARY KEY(COURSE_ID));

CREATE TABLE ENROLL(REG_NO VARCHAR(30),COURSE_ID INT,SEM
INT,MARKS INT,
                        FOREIGN KEY(REG_NO) REFERENCES
STUDENT(REG_NO) ON UPDATE CASCADE,
                   FOREIGN KEY(COURSE_ID) REFERENCES
COURSE(COURSE_ID) ON UPDATE CASCADE);

CREATE TABLE BOOK_ADPOTION(COURSE_ID INT,SEM INT,BOOK_ISBN
INT,
                   FOREIGN KEY(COURSE_ID) REFERENCES
COURSE(COURSE_ID) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE TEXT(BOOK_ISBN INT,BOOK_TITLE
VARCHAR(30),PUBLISHER VARCHAR(30),AUTHOR VARCHAR(30),
                         FOREIGN KEY(BOOK_ISBN) REFERENCES
COURSE(BOOK_ISBN) ON UPDATE CASCADE);

INSERT INTO STUDENT(REG_NO,SNAME,MAJOR,BDATE) VALUES
('CS01', 'RAM', 'DS', '1986-03-12'),
                                      ('IS02', 'SMITH',
'USP', '1987-12-23'),
                                        ('EC03', 'AHMED',
'SNS', '1985-04-17'),
                                        ('CS03', 'SNEHA',
'DBMS', '1987-01-01'),
                                      ('TC05', 'AKHILA',
'EC', '1986-10-06');
INSERT INTO COURSE(COURSE_ID,CNAME,DEPT) VALUES (11, 'DS',
'CS'),
                                  (22, 'USP', 'IS'),
                                  (33, 'SNS', 'EC'),
                                  (44, 'DBMS', 'CS'),
                                  (55, 'EC', 'TC');
INSERT INTO ENROLL(REG_NO,COURSE_ID,SEM,MARKS) VALUES
('CS01', 11, 4, 85),
                                       ('IS02', 22, 6, 80),
                                       ('EC03', 33, 2, 80),
                                       ('CS03', 44, 6, 75),
                                       ('TC05', 55, 2, 8);
```

```
INSERT INTO BOOK_ADOPTION(COURSE_ID,SEM,BOOK_ISBN) VALUES
(11,4,1),
(11,4,2),
(44,6,3),
(44,6,4),
(55,2,5),
(22,6,6),
(55,2,7);

-- iii) Demonstrate how you add a new text book to the
database and make this book be
-- adopted by some department.(ec department)
INSERT INTO BOOK_ADOPTION(COURSE_ID,SEM,BOOK_ISBN) VALUE
(33,4,8);
INSERT INTO TEXT(BOOK_ISBN, BOOK_TITLE, PUBLISHER, AUTHOR)
VALUE(8,'JAVA 14','ORACLE','NITHIN');
SELECT * FROM TEXT;

-- 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.
SELECT C.COURSE_ID,T.BOOK_ISBN,T.BOOK_TITLE FROM TEXT
T,COURSE C,BOOK_ADOPTION B WHERE T.BOOK_ISBN=B.BOOK_ISBN AND
B.COURSE_ID=C.COURSE_ID AND C.DEPT="CS" AND

                      (SELECT COUNT(B.BOOK_ISBN) FROM
BOOK_ADOPTION B WHERE C.COURSE_ID=B.COURSE_ID)>=2 ORDER BY
T.BOOK_TITLE;

-- --v) List any department that has all its adopted books
published by a specific publisher.
SELECT DISTINCT C.DEPT
     FROM COURSE C
    WHERE C.DEPT IN
    ( SELECT C.DEPT
    FROM COURSE C,BOOK_ADOPTION B,TEXT T
    WHERE C.COURSE_ID=B.COURSE_ID
    AND T.BOOK_ISBN=B.BOOK_ISBN
    AND T.PUBLISHER='Princeton')
    AND C.DEPT NOT IN
    (SELECT C.DEPT
    FROM COURSE C,BOOK_ADOPTION B,TEXT T
    WHERE C.COURSE_ID=B.COURSE_ID
    AND T.BOOK_ISBN=B.BOOK_ISBN
```

```
            AND T.PUBLISHER != 'Princeton');
```

Output:

```
53
54    -- iv) Produce a list of text books (include Course #, Book-ISBN, Book-title) in the
55    -- alphabetical order for courses offered by the 'CS' department that use more than two
56    -- books.
57 •  SELECT C.COURSE_ID,T.BOOK_ISBN,T.BOOK_TITLE FROM TEXT T,COURSE C,BOOK_ADOPTION B WHERE T.BOOK_ISBN=B.BOOK_ISBN AND B.COURSE_ID=C.COURSE_ID AND C.DEPT="CS" AND
58                                                  (SELECT COUNT(B.BOOK_ISBN) FROM BOOK_ADOPTION B WHERE C.COURSE_ID=B.COURSE_ID)>=2 ORDER BY T.BOOK_TITLE;
```

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

COURSE_ID	BOOK_ISBN	BOOK_TITLE
11	1	DS and C
44	3	Fundamentals of DBMS
11	2	Fundamentals of DS
44	4	SQL

```
59
60    -- --v) List any department that has all its adopted books published by a specific publisher.
61 •  SELECT DISTINCT C.DEPT
62        FROM COURSE C
63        WHERE C.DEPT IN
64        ( SELECT C.DEPT
65        FROM COURSE C,BOOK_ADOPTION B,TEXT T
66        WHERE C.COURSE_ID=B.COURSE_ID
67        AND T.BOOK_ISBN=B.BOOK_ISBN
68        AND T.PUBLISHER='Princeton')
69        AND C.DEPT NOT IN
70        (SELECT C.DEPT
71        FROM COURSE C,BOOK_ADOPTION B,TEXT T
72        WHERE C.COURSE_ID=B.COURSE_ID
73        AND T.BOOK_ISBN=B.BOOK_ISBN
74        AND T.PUBLISHER != 'Princeton');
```

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

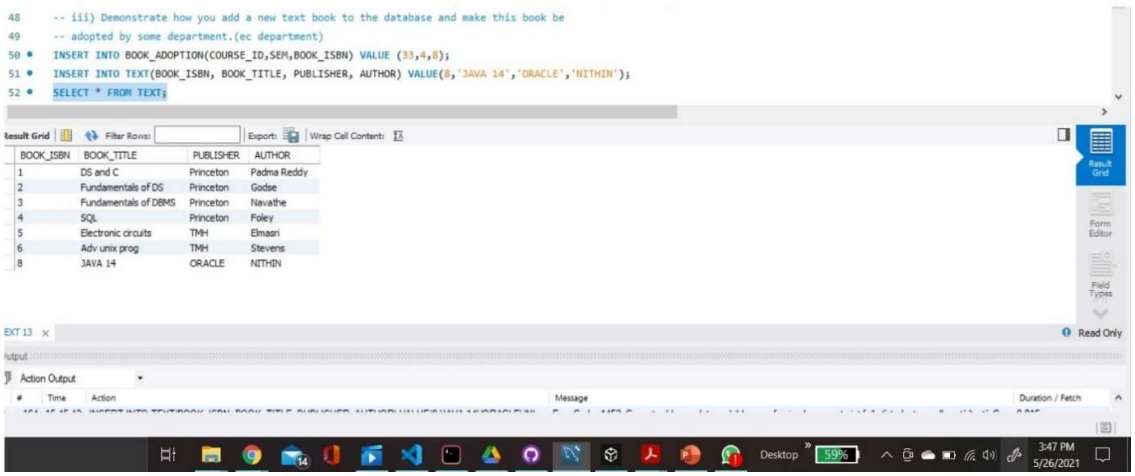
DEPT
CS

Result Grid | Form Editor

COURSE 15 x | Read Only

Output

Action Output



PROGRAM 9:

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.

SQL SCRIPT:

```
-- i. Create the above tables by properly specifying the primary keys and the foreign keys.
-- ii. Enter at least five tuples for each relation.
CREATE DATABASE MOVIE;
USE MOVIE;
-- SHOW DATABASES;
CREATE TABLE ACTOR(ACT_ID INT,ACT_NAME VARCHAR(30),ACT_GENDER VARCHAR(30),PRIMARY KEY(ACT_ID));

CREATE TABLE DIRECTOR(DIR_ID INT,DIR_NAME VARCHAR(30),PHONE_NO LONG,PRIMARY KEY(DIR_ID));
```

```
CREATE TABLE MOVIES(MOVIE_ID
INT,MOVIE_TITLE
VARCHAR(30),MOVIE_YEAR
INT,MOVIE_LANG VARCHAR(30),DIR_ID
INT,PRIMARY KEY(MOVIE_ID),
                    FOREIGN KEY(DIR_ID)
REFERENCES DIRECTOR(DIR_ID) ON
UPDATE CASCADE);

CREATE TABLE MOVIE_CAST(ACT_ID
INT,MOVIE_ID INT,ROLE VARCHAR(30),
                FOREIGN KEY(ACT_ID)
REFERENCES ACTOR(ACT_ID) ON DELETE
CASCADE ON UPDATE CASCADE,
                FOREIGN KEY(MOVIE_ID)
REFERENCES MOVIES(MOVIE_ID) ON
DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE RATING(MOVIE_ID
INT,RATING_STARS INT CHECK
(RATING_STARS<=5),
                    FOREIGN KEY(MOVIE_ID)
REFERENCES MOVIES(MOVIE_ID) ON
UPDATE CASCADE);
```

```
INSERT INTO
ACTOR(ACT_ID,ACT_NAME,ACT_GENDER)
VALUES (1, 'Tom Cruise','MALE' ),

      (2, 'Jamie Foxx','MALE'),

      (3, 'Robert De Niro', 'MALE'),

      (4, 'Zoe Saldana','FEMALE'),

      (5, 'Kim Novak','FEMALE');

INSERT INTO DIRECTOR(DIR_ID,
DIR_NAME, PHONE_NO) VALUES (1, 'Steven
Spielberg', 9110626411),

         (2, 'Quentin Tarantino',
9110626422),
                                        (3,
'Alfred Hitchcock', 9110626433),
                                        (4,
'Martin Scorsese', 9110626444),
                                        (5,
'James Cameron', 9110626455);

INSERT INTO
MOVIES(MOVIE_ID,MOVIE_TITLE,MOVIE_YE
```

```
AR,MOVIE_LANG,DIR_ID) VALUES(1,'War of
the Worlds', 2005, 'ENG', 1),

                              (2,'Minority
Report', 2002, 'ENG', 1),

(3,'Django Unchained', 2012, 'ENG', 2),

(4,'Vertigo', 1958, 'ENG', 3),

(5,'Goodfellas',1990, 'ENG', 4),

(6, 'Avatar', 2009, 'ENG', 5);

INSERT INTO MOVIE_CAST(ACT_ID,
MOVIE_ID,ROLE) VALUES(1, 1, 'LEAD'),


    (1, 2, 'LEAD'),
                                        (2, 3,
'LEAD'),
                                        (3, 5,
'LEAD'),
                                        (4, 6,
'CO-STAR'),
                                        (5, 4,
'LEAD');
```

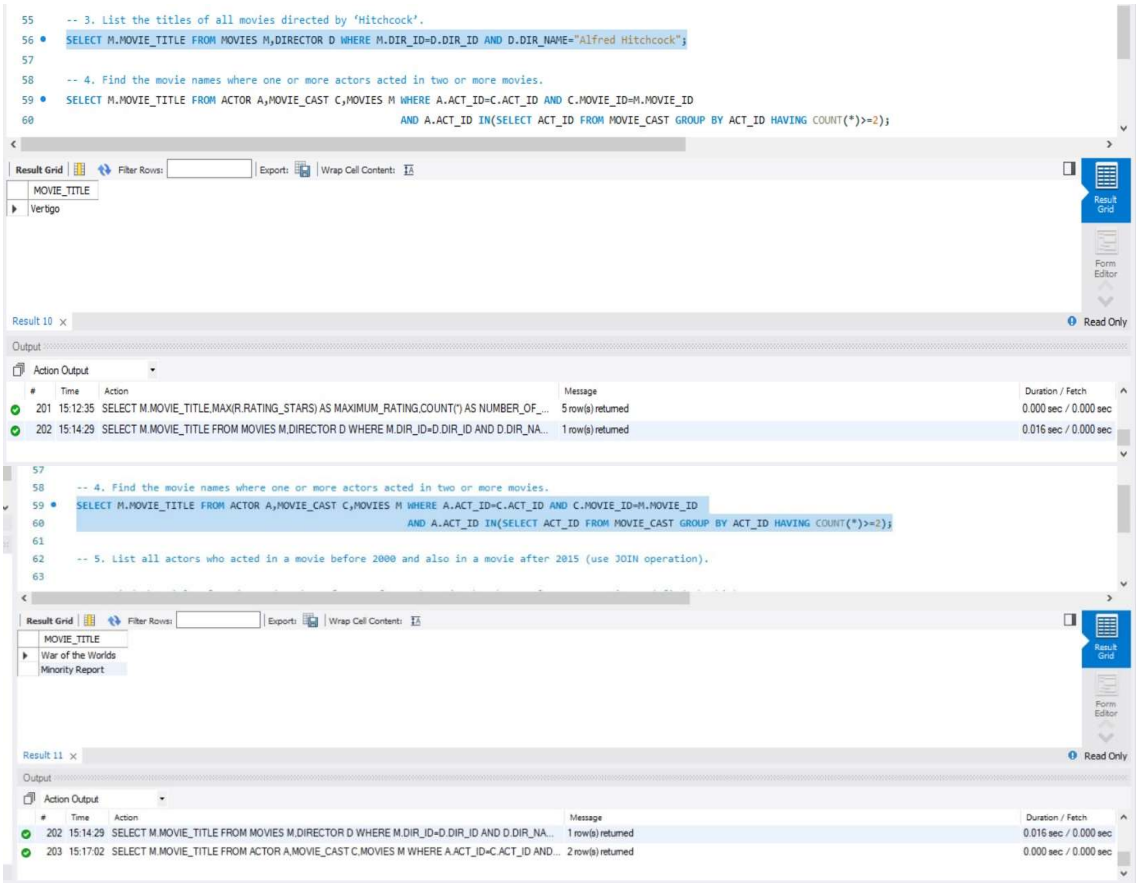
```
INSERT INTO
RATING(MOVIE_ID,RATING_STARS)
VALUES (1, 3),

(2, 4),
                                        (3, 5),
                                        (4, 4),
                                        (5, 5),
                                        (1, 5),
                                        (2, 3),
                                        (5, 4);

-- 3. List the titles of all movies directed by
‘Hitchcock’.
SELECT M.MOVIE_TITLE FROM MOVIES
M,DIRECTOR D WHERE
M.DIR_ID=D.DIR_ID AND
D.DIR_NAME="Alfred Hitchcock";

-- 4. Find the movie names where one or
more actors acted in two or more movies.
SELECT M.MOVIE_TITLE FROM ACTOR
A,MOVIE_CAST C,MOVIES M WHERE
A.ACT_ID=C.ACT_ID AND
C.MOVIE_ID=M.MOVIE_ID
                                              AND
A.ACT_ID IN(SELECT ACT_ID FROM
```

```
MOVIE_CAST GROUP BY ACT_ID HAVING
COUNT(MOVIE_ID)>=2);

-- 5. List all actors who acted in a movie
before 2000 and also in a movie after 2015
(use JOIN operation).
SELECT A.ACT_NAME FROM ACTOR A
JOIN MOVIE_CAST MC ON
A.ACT_ID=MC.ACT_ID
JOIN MOVIES M ON
MC.MOVIE_ID=M.MOVIE_ID
WHERE M.MOVIE_YEAR NOT BETWEEN
2000 AND 2015;

-- 6. 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
M.MOVIE_TITLE,MAX(R.RATING_STARS) AS
MAXIMUM_RATING,COUNT(*) AS
NUMBER_OF_RATINGS FROM MOVIES
M,RATING R
                                    WHERE
M.MOVIE_ID=R.MOVIE_ID  GROUP BY
R.MOVIE_ID HAVING COUNT(*)>0 ORDER
BY M.MOVIE_TITLE;
```

-- 7. Update rating of all movies directed by ‘Steven Spielberg’ to 5.

```
UPDATE RATING SET RATING_STARS = 5
WHERE MOVIE_ID IN
(SELECT M.MOVIE_ID FROM MOVIES M,
DIRECTOR D WHERE M.DIR_ID = D.DIR_ID
AND D.DIR_NAME='Steven Spielberg');
SELECT * FROM RATING;
```

OUTPUT:



```
64    -- 6. Find the title of movies and number of stars for each movie that has at least one rating and find the highest
65    -- number of stars that movie received. Sort the result by movie title.
66 •  SELECT M.MOVIE_TITLE,MAX(R.RATING_STARS) AS MAXIMUM_RATING,COUNT(*) AS NUMBER_OF_RATINGS FROM MOVIES M,RATING R
67                                          WHERE M.MOVIE_ID=R.MOVIE_ID  GROUP BY R.MOVIE_ID HAVING COUNT(*)>0 ORDER BY M.MOVIE_TITLE;
68
```

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

MOVIE_TITLE	MAXIMUM_RATING	NUMBER_OF_RATINGS
Django Unchained	5	1
Goodfellas	5	2
Minority Report	4	2
Vertigo	4	1
War of the Worlds	5	2

Result 12 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
203	15:17:02	SELECT M.MOVIE_TITLE FROM ACTOR A,MOVIE_CAST C,MOVIES M WHERE A.ACT_ID=C.ACT_ID AND...	2 row(s) returned	0.000 sec / 0.000 sec
204	15:17:33	SELECT M.MOVIE_TITLE,MAX(R.RATING_STARS) AS MAXIMUM_RATING,COUNT(*) AS NUMBER_OF_...	5 row(s) returned	0.000 sec / 0.000 sec

```
68
69    -- 7. Update rating of all movies directed by 'Steven Spielberg' to 5.
70 •  UPDATE RATING SET RATING_STARS = 5 WHERE MOVIE_ID IN
71    (SELECT M.MOVIE_ID FROM MOVIES M, DIRECTOR D WHERE M.DIR_ID = D.DIR_ID AND D.DIR_NAME='Steven Spielberg');
72 •  SELECT * FROM RATING;
73
74
```

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

MOVIE_ID	RATING_STARS
1	5
2	5
3	5
4	4
5	5
1	5
2	5
5	4

RATING 13 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
202	15:14:29	SELECT M.MOVIE_TITLE FROM MOVIES M,DIRECTOR D WHERE M.DIR_ID=D.DIR_ID AND D.DIR_NA...	1 row(s) returned	0.016 sec / 0.000 sec
203	15:17:02	SELECT M.MOVIE_TITLE FROM ACTOR A,MOVIE_CAST C,MOVIES M WHERE A.ACT_ID=C.ACT_ID AND...	2 row(s) returned	0.000 sec / 0.000 sec
204	15:17:33	SELECT M.MOVIE_TITLE,MAX(R.RATING_STARS) AS MAXIMUM_RATING,COUNT(*) AS NUMBER_OF_...	5 row(s) returned	0.000 sec / 0.000 sec
205	15:19:38	UPDATE RATING SET RATING_STARS = 5 WHERE MOVIE_ID IN (SELECT M.MOVIE_ID FROM MOVIES ...	3 row(s) affected Rows matched: 4 Changed: 3 Warnings: 0	0.016 sec
206	15:20:03	SELECT * FROM RATING LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec

```
62    -- 5. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).
63 •  SELECT A.ACT_NAME FROM ACTOR A
64    JOIN MOVIE_CAST MC ON A.ACT_ID=MC.ACT_ID
65    JOIN MOVIES M ON MC.MOVIE_ID=M.MOVIE_ID
66    WHERE M.MOVIE_YEAR NOT BETWEEN 2000 AND 2015;
67
68    -- 6. Find the title of movies and number of stars for each movie that has at least one rating and find the highest
69    -- number of stars that movie received. Sort the result by movie title.
70 •  SELECT M.MOVIE_TITLE,MAX(R.RATING_STARS) AS MAXIMUM_RATING,COUNT(*) AS NUMBER_OF_RATINGS FROM MOVIES M,RATING R
71                                          WHERE M.MOVIE_ID=R.MOVIE_ID  GROUP BY R.MOVIE_ID HAVING COUNT(*)>0 ORDER BY M.MOVIE_TITLE;
72
73    -- 7. Update rating of all movies directed by 'Steven Spielberg' to 5.
```

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

ACT_NAME
Kim Novak
Robert De Niro

Result 21 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
211	15:56:22	SELECT A.ACT_NAME FROM ACTOR A JOIN MOVIE_CAST MC ON A.ACT_ID=MC.ACT_ID JOIN MOVIES...	6 row(s) returned	0.016 sec / 0.000 sec
212	15:57:04	SELECT A.ACT_NAME FROM ACTOR A JOIN MOVIE_CAST MC ON A.ACT_ID=MC.ACT_ID JOIN MOVIES...	2 row(s) returned	0.000 sec / 0.000 sec
213	15:57:12	SELECT A.ACT_NAME FROM ACTOR A JOIN MOVIE_CAST MC ON A.ACT_ID=MC.ACT_ID JOIN MOVIES...	2 row(s) returned	0.000 sec / 0.000 sec
214	15:59:01	SELECT M.MOVIE_TITLE FROM ACTOR A,MOVIE_CAST C,MOVIES M WHERE A.ACT_ID=C.ACT_ID AND...	2 row(s) returned	0.000 sec / 0.000 sec
215	15:59:10	SELECT A.ACT_NAME FROM ACTOR A JOIN MOVIE_CAST MC ON A.ACT_ID=MC.ACT_ID JOIN MOVIES...	2 row(s) returned	0.016 sec / 0.000 sec

PROGRAM 10:

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.

SQL SCRIPT:

```
CREATE DATABASE COLLEGE;
USE COLLEGE;

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

CREATE TABLE SEMSEC(
SSID VARCHAR(5) PRIMARY KEY,
SEM INTEGER,
SEC CHAR(1));

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

```
CREATE TABLE SUBJECT(
SUBCODE VARCHAR(8) PRIMARY KEY,
TITLE VARCHAR(20),
SEM INTEGER,
CREDITS INTEGER);

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


INSERT INTO STUDENT VALUES
('1BI13CS020','ANAND','UDUPI','944920792
7','M'),
('1BI13CS062','AMITA','BENGALURU','94824
43852','F'),
```

```
('1BI15CS101','CHETAN','BENGALURU','776
6554433','M'),
('1BI13CS066','DIVYA','MANGALURU','94824
43857','F'),
('1BI14CS010','ESHA','BENGALURU','778244
3857','F'),
('1BI14CS032','GANESH','MYSURU','944900
9017','M'),
('1BI14CS025','HARISH','BENGALURU','9449
009017','M'),
('1BI15CS011','ISHA','TUMKUR','987900901
7','F'),
('1BI15CS029','JAY','DAVANGERE','7749009
017','M'),
('1BI15CS045','KAVYA','BELLARY','7649009
017','F'),
('1BI15CS091','MALINI','MANGALURU','9679
009017','F'),
('1BI16CS045','NEEL','KALBURGI','9449780
017','M'),
('1BI16CS088','PARTHA','SHIMOGA','946900
9017','M'),
('1BI16CS122','REEMA','CHIKAMAGALUR','7
754646789','F');
INSERT INTO SEMSEC VALUES
('CSE8A', 8,'A'),
('CSE8B', 8,'B'),
('CSE8C', 8,'C'),
```

```
('CSE7A', 7,'A'),
('CSE7B', 7,'B'),
('CSE7C', 7,'C'),
('CSE6A', 6,'A'),
('CSE6B', 6,'B'),
('CSE6C', 6,'C'),
('CSE5A', 5,'A'),
('CSE5B', 5,'B'),
('CSE5C', 5,'C'),
('CSE4A', 4,'A'),
('CSE4B', 4,'B'),
('CSE4C', 4,'C'),
('CSE3A', 3,'A'),
('CSE3B', 3,'B'),
('CSE3C', 3,'C'),
('CSE2A', 2,'A'),
('CSE2B', 2,'B'),
('CSE2C', 2,'C'),
('CSE1A', 1,'A'),
('CSE1B', 1,'B'),
('CSE1C', 1,'C');
INSERT INTO CLASS VALUES
('1BI13CS020','CSE8A'),
('1BI13CS062','CSE8A'),
('1BI13CS066','CSE8B'),
('1BI15CS101','CSE8C'),
('1BI14CS010','CSE7A'),
('1BI14CS025','CSE7A'),
```

```
('1BI14CS032','CSE7A'),
('1BI15CS011','CSE4A'),
('1BI15CS029','CSE4A'),
('1BI15CS045','CSE4B'),
('1BI15CS091','CSE4C'),
('1BI16CS045','CSE3A'),
('1BI16CS088','CSE3B'),
('1BI16CS122','CSE3C');
INSERT INTO SUBJECT VALUES
('10CS81','ACA', 8, 4),
('10CS82','SSM', 8, 4),
('10CS83','NM', 8, 4),
('10CS84','CC', 8, 4),
('10CS85','PW', 8, 4),
('10CS71','OOAD', 7, 4),
('10CS72','ECS', 7, 4),
('10CS73','PTW', 7, 4),
('10CS74','DWDM', 7, 4),
('10CS75','JAVA', 7, 4),
('10CS76','SAN', 7, 4),
('15CS51','ME', 5, 4),
('15CS52','CN', 5, 4),
('15CS53','DBMS', 5, 4),
('15CS54','ATC', 5, 4),
('15CS55','JAVA', 5, 3),
('15CS56','AI', 5, 3),
('15CS41','M4', 4, 4),
('15CS42','SE', 4, 4),
```

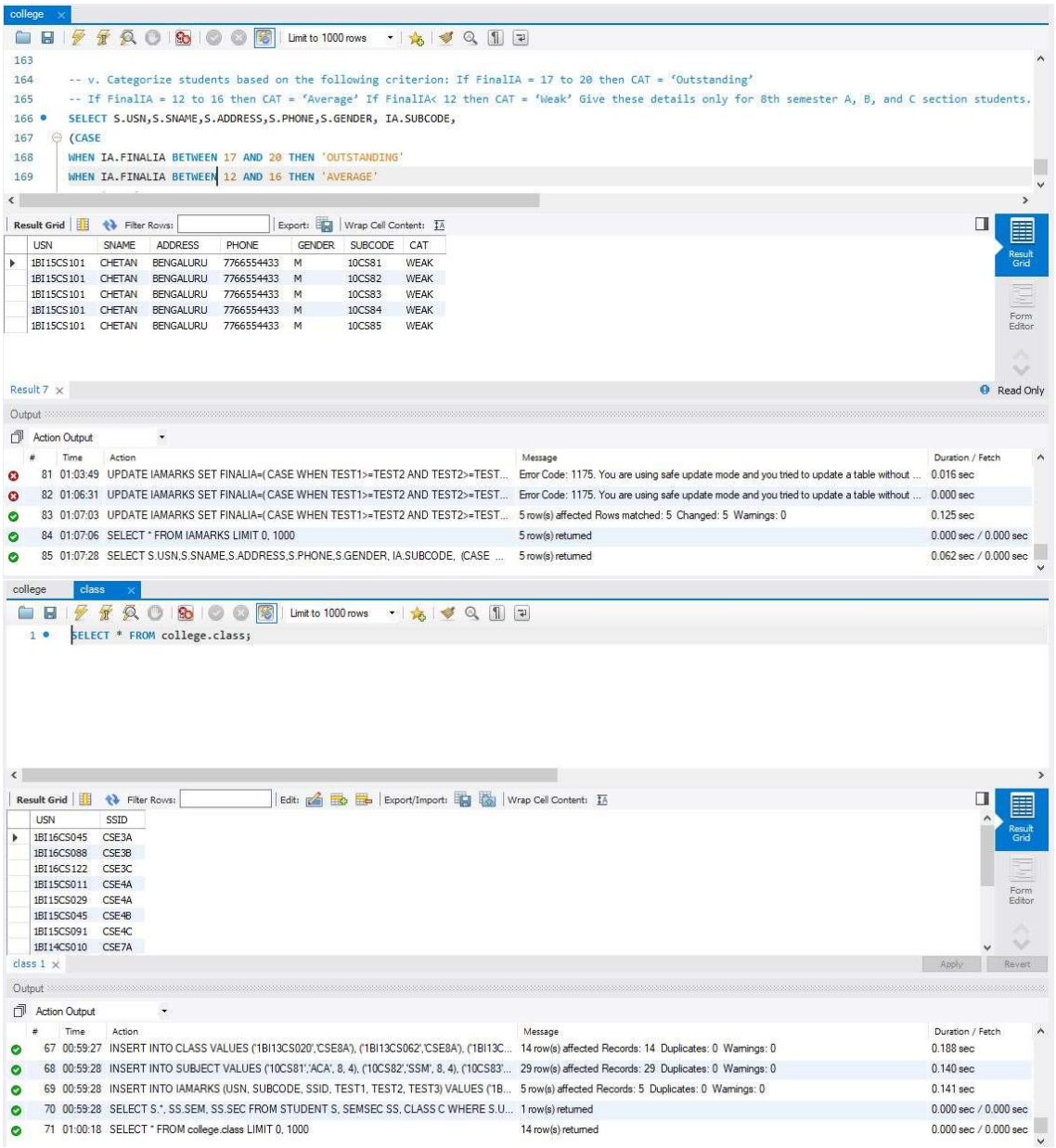
```
('15CS43','DAA', 4, 4),
('15CS44','MPMC', 4, 4),
('15CS45','OOC', 4, 3),
('15CS46','DC', 4, 3),
('15CS31','M3', 3, 4),
('15CS32','ADE', 3, 4),
('15CS33','DSA', 3, 4),
('15CS34','CO', 3, 4),
('15CS35','USP', 3, 3),
('15CS36','DMS', 3, 3);
INSERT INTO IAMARKS (USN, SUBCODE,
SSID, TEST1, TEST2, TEST3) VALUES
('1BI15CS101','10CS81','CSE8C', 15, 16,
18),
('1BI15CS101','10CS82','CSE8C', 12, 19,
14),
('1BI15CS101','10CS83','CSE8C', 19, 15,
20),
('1BI15CS101','10CS84','CSE8C', 20, 16,
19),
('1BI15CS101','10CS85','CSE8C', 15, 15,
12);

-- i. List all the student details studying in
fourth semester ‘C’ section.
SELECT S.*, SS.SEM, SS.SEC FROM
STUDENT S, SEMSEC SS, CLASS C
```

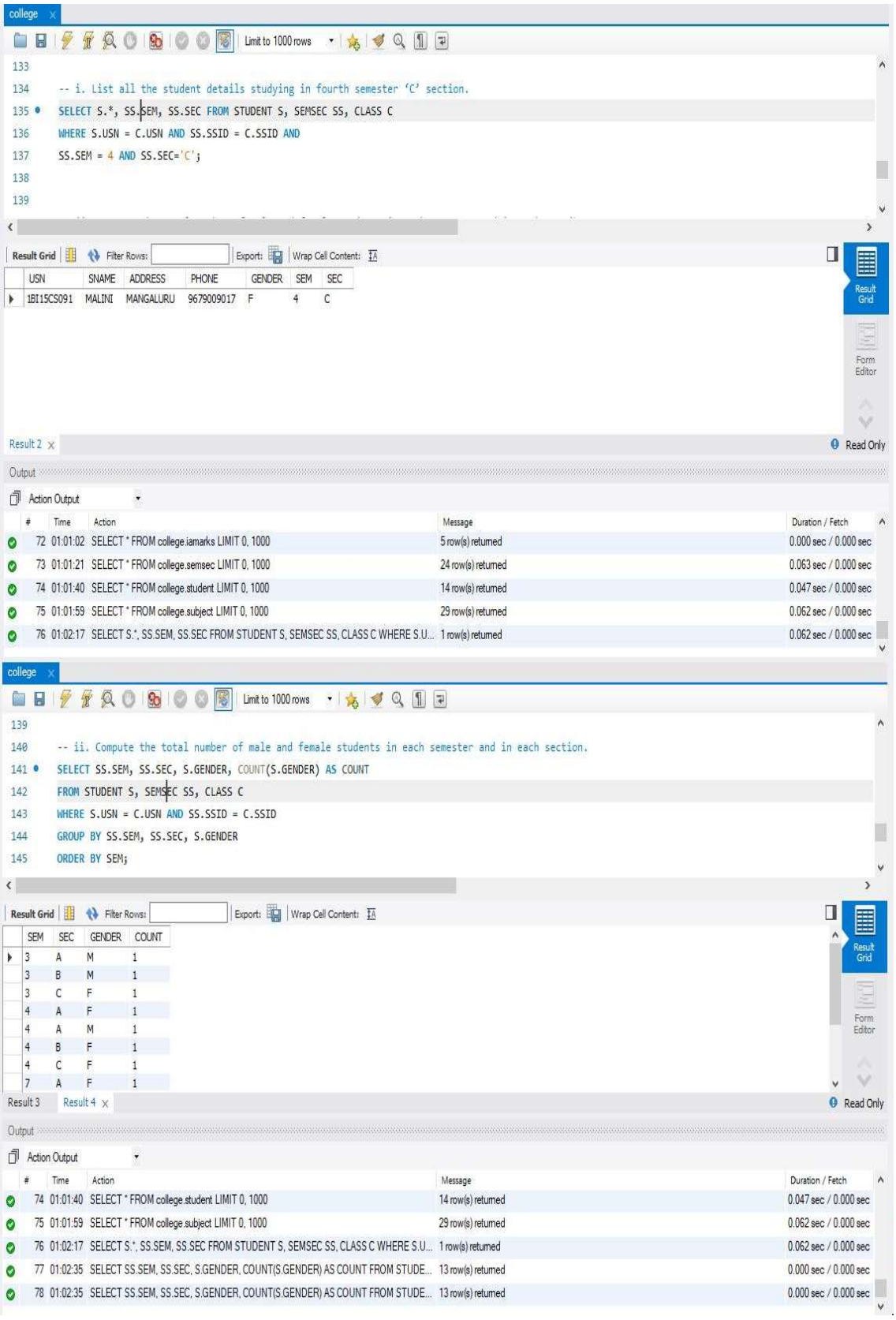
```
WHERE S.USN = C.USN AND SS.SSID =
C.SSID AND
SS.SEM = 4 AND SS.SEC='C';


-- ii. 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;

-- iii. Create a view of Test1 marks of
student USN ‘1BI15CS101’ in all subjects.
CREATE VIEW STUDENT_TEST1_MARKS_V
AS
SELECT TEST1, SUBCODE FROM IAMARKS
WHERE USN = '1BI15CS101';

-- iv. Calculate the FinalIA (average of best
two test marks) and update the
corresponding table for all students.
UPDATE IAMARKS SET FINALIA=(
CASE
```

```
WHEN TEST1>=TEST2 AND TEST2>=TEST3
THEN TEST1+TEST2/2
WHEN TEST2>=TEST3 AND TEST3>=TEST1
THEN TEST2+TEST3/2
ELSE TEST1+TEST3/2
END);
SELECT * FROM IAMARKS;


-- 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.
SELECT
S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GE
NDER, IA.SUBCODE,
(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:





college

Limit to 1000 rows

```
133
134    -- i. List all the student details studying in fourth semester 'C' section.
135    SELECT S.*, SS.SEM, SS.SEC FROM STUDENT S, SEMSEC SS, CLASS C
136    WHERE S.USN = C.USN AND SS.SSID = C.SSID AND
137    SS.SEM = 4 AND SS.SEC='C';
138
139
```

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

USN	SNAME	ADDRESS	PHONE	GENDER	SEM	SEC
1BI15CS091	MALINI	MANGALURU	9679009017	F	4	C

Result Grid | Form Editor | Result 2 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
72	01:01:02	SELECT * FROM college.iamarks LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
73	01:01:21	SELECT * FROM college.semsec LIMIT 0, 1000	24 row(s) returned	0.063 sec / 0.000 sec
74	01:01:40	SELECT * FROM college.student LIMIT 0, 1000	14 row(s) returned	0.047 sec / 0.000 sec
75	01:01:59	SELECT * FROM college.subject LIMIT 0, 1000	29 row(s) returned	0.062 sec / 0.000 sec
76	01:02:17	SELECT S.*, SS.SEM, SS.SEC FROM STUDENT S, SEMSEC SS, CLASS C WHERE S.U...	1 row(s) returned	0.062 sec / 0.000 sec

college

Limit to 1000 rows

```
139
140    -- ii. Compute the total number of male and female students in each semester and in each section.
141    SELECT SS.SEM, SS.SEC, S.GENDER, COUNT(S.GENDER) AS COUNT
142    FROM STUDENT S, SEMSEC SS, CLASS C
143    WHERE S.USN = C.USN AND SS.SSID = C.SSID
144    GROUP BY SS.SEM, SS.SEC, S.GENDER
145    ORDER BY SEM;
```

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

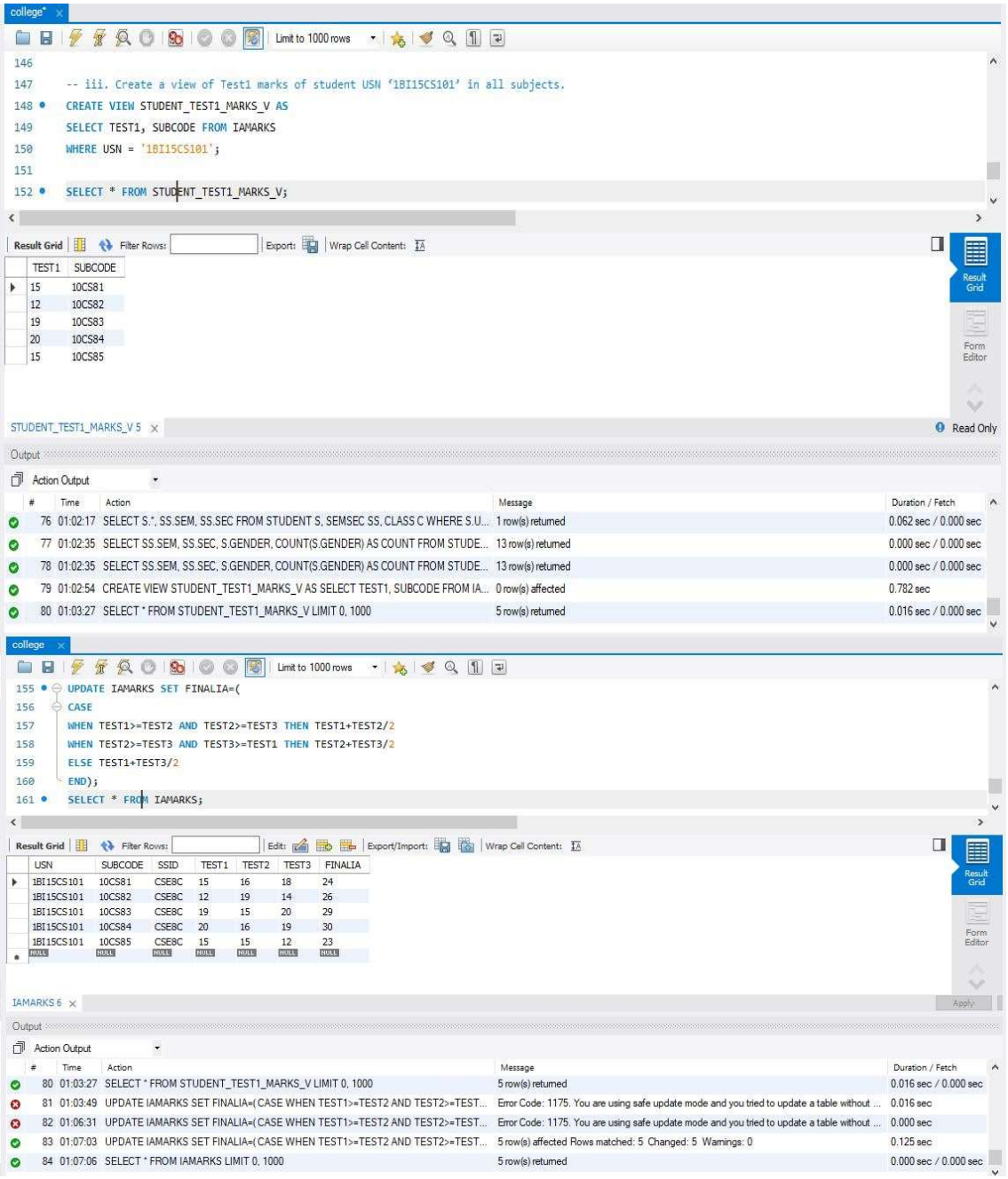
SEM	SEC	GENDER	COUNT
3	A	M	1
3	B	M	1
3	C	F	1
4	A	F	1
4	A	M	1
4	B	F	1
4	C	F	1
7	A	F	1

Result Grid | Form Editor | Result 3 | Result 4 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
74	01:01:40	SELECT * FROM college.student LIMIT 0, 1000	14 row(s) returned	0.047 sec / 0.000 sec
75	01:01:59	SELECT * FROM college.subject LIMIT 0, 1000	29 row(s) returned	0.062 sec / 0.000 sec
76	01:02:17	SELECT S.*, SS.SEM, SS.SEC FROM STUDENT S, SEMSEC SS, CLASS C WHERE S.U...	1 row(s) returned	0.062 sec / 0.000 sec
77	01:02:35	SELECT SS.SEM, SS.SEC, S.GENDER, COUNT(S.GENDER) AS COUNT FROM STUDE...	13 row(s) returned	0.000 sec / 0.000 sec
78	01:02:35	SELECT SS.SEM, SS.SEC, S.GENDER, COUNT(S.GENDER) AS COUNT FROM STUDE...	13 row(s) returned	0.000 sec / 0.000 sec



```
-- iii. Create a view of Test1 marks of student USN '1BI15CS101' in all subjects.
CREATE VIEW STUDENT_TEST1_MARKS_V AS
SELECT TEST1, SUBCODE FROM IAMARKS
WHERE USN = '1BI15CS101';

SELECT * FROM STUDENT_TEST1_MARKS_V;
```

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

#	Time	Action	Message	Duration / Fetch
76	01:02:17	SELECT S.*, SS.SEM, SS.SEC FROM STUDENT S, SEMSEC SS, CLASS C WHERE S.U...	1 row(s) returned	0.062 sec / 0.000 sec
77	01:02:35	SELECT SS.SEM, SS.SEC, S.GENDER, COUNT(S.GENDER) AS COUNT FROM STUDE...	13 row(s) returned	0.000 sec / 0.000 sec
78	01:02:35	SELECT SS.SEM, SS.SEC, S.GENDER, COUNT(S.GENDER) AS COUNT FROM STUDE...	13 row(s) returned	0.000 sec / 0.000 sec
79	01:02:54	CREATE VIEW STUDENT_TEST1_MARKS_V AS SELECT TEST1, SUBCODE FROM IA...	0 row(s) affected	0.782 sec
80	01:03:27	SELECT * FROM STUDENT_TEST1_MARKS_V LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec

```
UPDATE IAMARKS SET FINALIA=(
CASE
WHEN TEST1>=TEST2 AND TEST2>=TEST3 THEN TEST1+TEST2/2
WHEN TEST2>=TEST3 AND TEST3>=TEST1 THEN TEST2+TEST3/2
ELSE TEST1+TEST3/2
END);
SELECT * FROM IAMARKS;
```

USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINALIA
1BI15CS101	10CS81	CSE8C	15	16	18	24
1BI15CS101	10CS82	CSE8C	12	19	14	26
1BI15CS101	10CS83	CSE8C	19	15	20	29
1BI15CS101	10CS84	CSE8C	20	16	19	30
1BI15CS101	10CS85	CSE8C	15	15	12	23

#	Time	Action	Message	Duration / Fetch
80	01:03:27	SELECT * FROM STUDENT_TEST1_MARKS_V LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
81	01:03:49	UPDATE IAMARKS SET FINALIA=(CASE WHEN TEST1>=TEST2 AND TEST2>=TEST...	Error Code: 1175. You are using safe update mode and you tried to update a table without ...	0.016 sec
82	01:06:31	UPDATE IAMARKS SET FINALIA=(CASE WHEN TEST1>=TEST2 AND TEST2>=TEST...	Error Code: 1175. You are using safe update mode and you tried to update a table without ...	0.000 sec
83	01:07:03	UPDATE IAMARKS SET FINALIA=(CASE WHEN TEST1>=TEST2 AND TEST2>=TEST...	5 row(s) affected Rows matched: 5 Changed: 5 Warnings: 0	0.125 sec
84	01:07:06	SELECT * FROM IAMARKS LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec