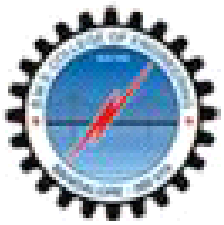


B. M. S. COLLEGE OF ENGINEERING, BENGALURU
Autonomous Institute, Affiliated to VTU

DEPARTMENT OF CSE

2021

Lab Report of Database Management System

Database Management System -19CS4PCDBM

Submitted by
Name:Yashaswini Shah
USN:1BM19CS216
Semester : 4

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

PROGRAM 1: INSURANCE DATABASE

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)

```
create database insurance;
use insurance;

create table person(
   driver_id varchar(10),
   name varchar(20),
   address varchar(30),
   primary key(driver_id)
);
```

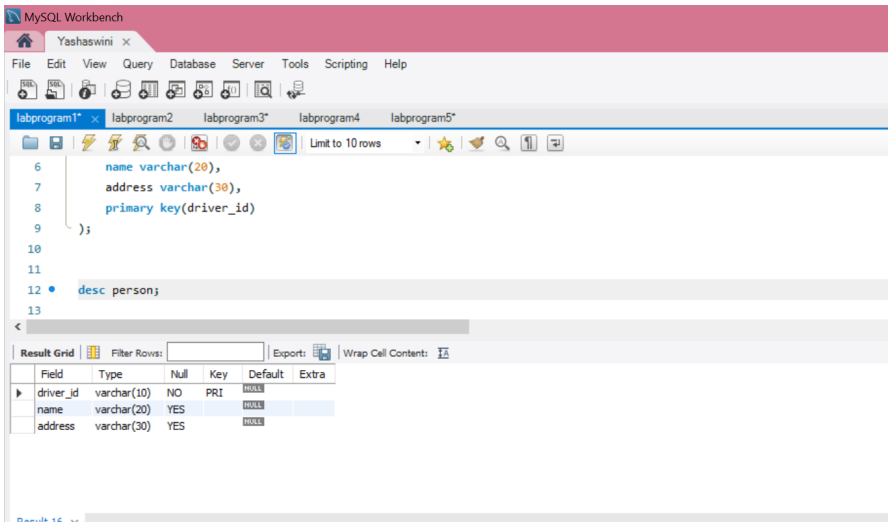
```
desc person;

create table car(
   reg_num varchar(10),
   model varchar(10),
   year int,
   primary key(reg_num)
);

desc car;

create table accident(
   report_num int,
   accident_date date,
   location varchar(20),
   primary key(report_num)
);
desc accident;

create table owns(
   driver_id varchar(10),
   reg_num varchar(10),
   primary key(driver_id,reg_num),
   foreign key(driver_id) references person(driver_id),
   foreign key(reg_num) references car(reg_num)
);

desc owns;

create table participated(
   driver_id varchar(10),
   reg_num varchar(10),
   report_num int,
   damage_amount int,
   primary key(driver_id,reg_num,report_num),
   foreign key(driver_id) references person(driver_id),
   foreign key(reg_num) references car(reg_num),
   foreign key(report_num) references accident(report_num)
);

desc participated;

insert into person values('A01','Raghu','Electronic City');
insert into person values('A02','Rishab','Orange County');
insert into person values('A03','Rufus','NR Colony');
insert into person values('A04','Jamal','Lawrence Park');
insert into person values('A05','Kevin','Rosedale');

commit;
```

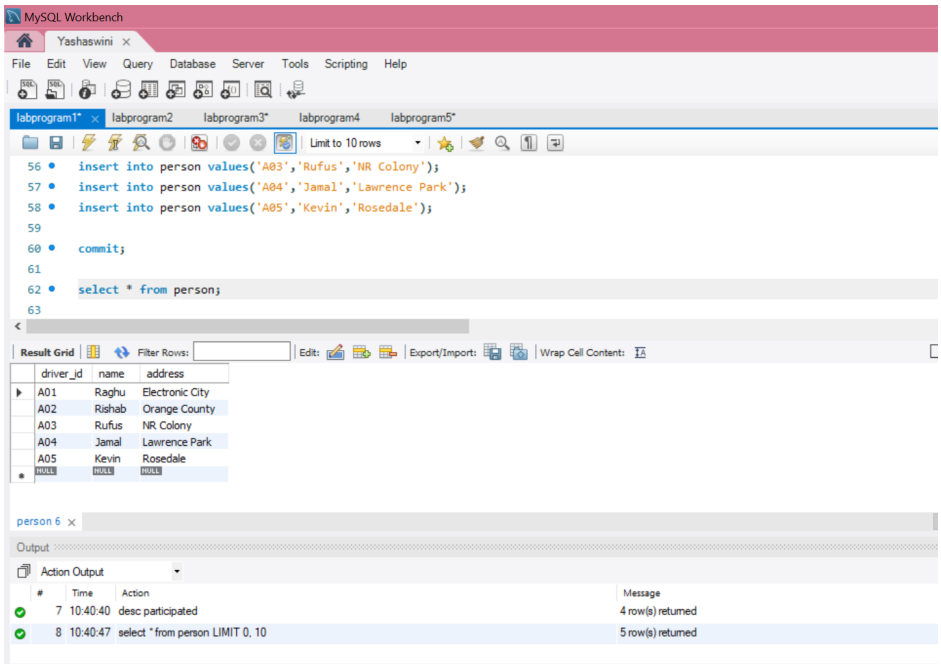
```
select * from person;

insert into car values('KA031111','Accord',2005);
insert into car values('KA041122','MX-5',2019);
insert into car values('KA051133','Indica',2010);
insert into car values('KA061144','Prius',2015);
insert into car values('KA071155','Camry',2020);

commit;
select * from car;

insert into accident values(111,'2020-01-01','NR Road');
insert into accident values(122,'2020-02-02','Dalhousie Road');
insert into accident values(133,'2020-03-03','Henry Road');
insert into accident values(144,'2020-04-04','Beehive Road');
insert into accident values(155,'2020-05-05','Orange Street');
commit;

select * from accident;

insert into owns values ('A01','KA031111');
insert into owns values ('A02','KA041122');
insert into owns values ('A03','KA051133');
insert into owns values ('A04','KA061144');
insert into owns values ('A05','KA071155');
commit;

select * from owns;
insert into participated values ('A01','KA031111',111, 10000);
insert into participated values ('A02','KA041122',122, 20000);
insert into participated values ('A03','KA051133',133, 30000);
insert into participated values ('A04','KA061144',144, 40000);
insert into participated values ('A05','KA071155',155, 50000);
commit;

select * from participated;
```

i. Create the above tables by properly specifying the primary keys and the foreign keys.

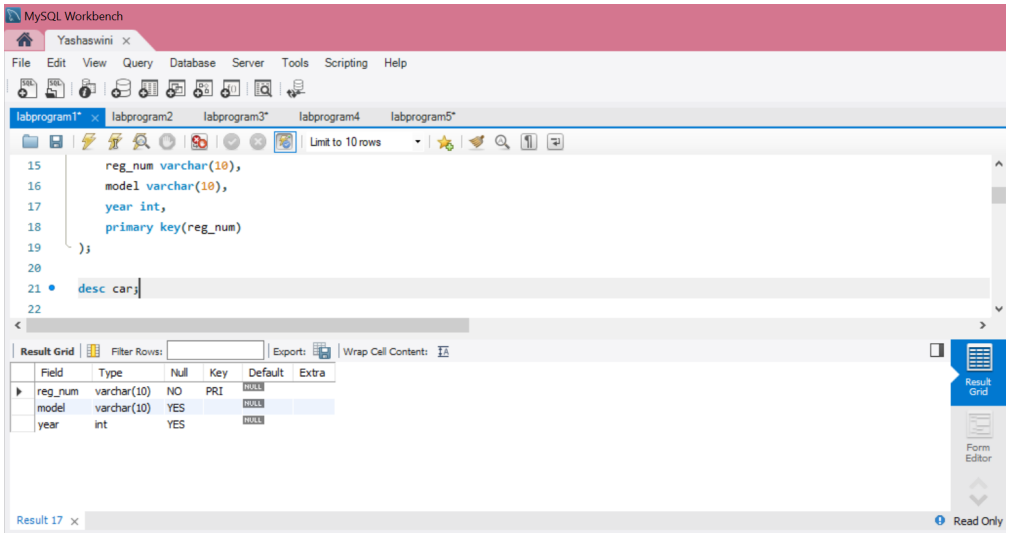
ii. Enter at least five tuples for each relation.

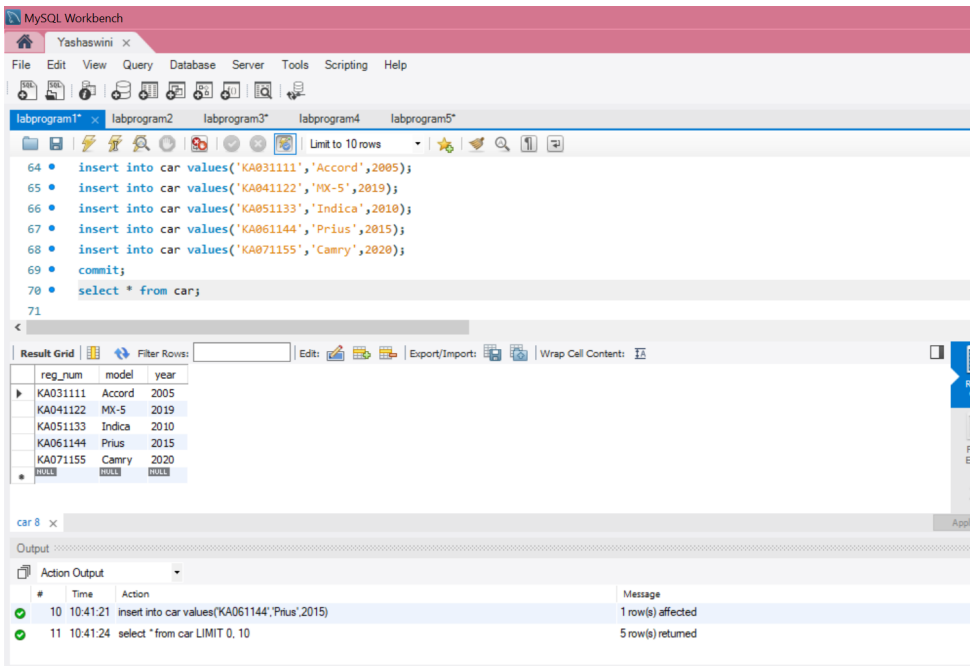
PERSON TABLE



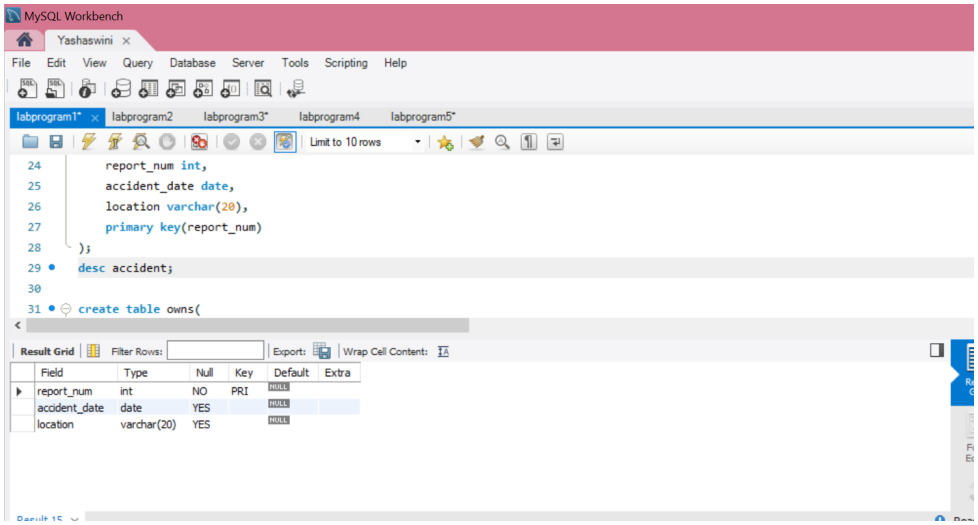


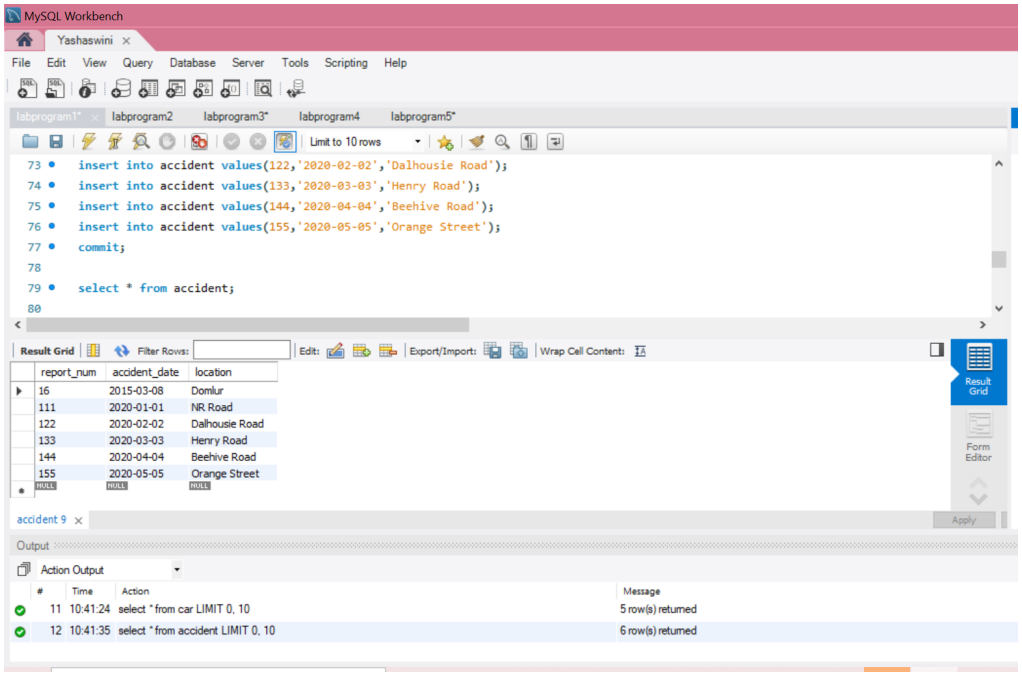
CAR TABLE


MySQL Workbench
Yashaswini
File Edit View Query Database Server Tools Scripting Help
labprogram1* labprogram2 labprogram3* labprogram4 labprogram5*
Limit to 10 rows
15 reg_num varchar(10),
16 model varchar(10),
17 year int,
18 primary key(reg_num)
19);
20
21 desc car;
22
Result Grid Filter Rows: Export: Wrap Cell Content:
Field Type Null Key Default Extra
reg_num varchar(10) NO PRI NULL
model varchar(10) YES NULL
year int YES NULL
Result Grid
Form Editor
Result 17
Read Only

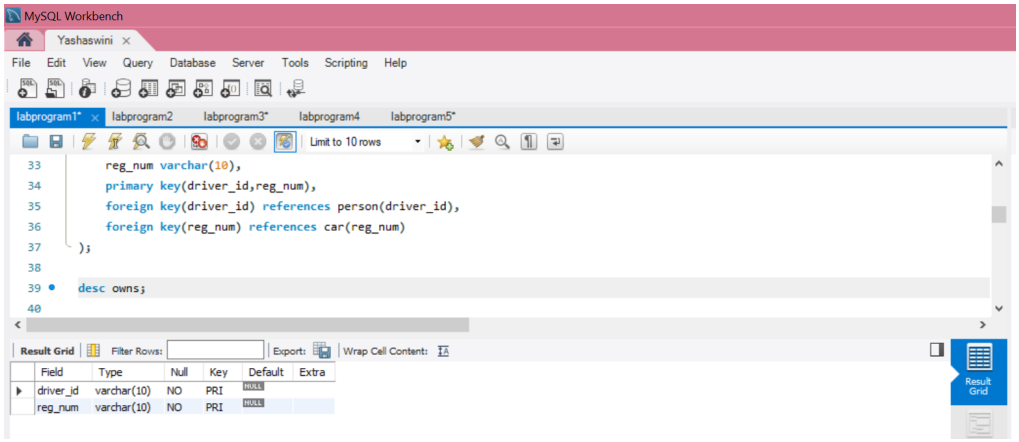

MySQL Workbench
Yashaswini
File Edit View Query Database Server Tools Scripting Help
labprogram1* labprogram2 labprogram3* labprogram4 labprogram5*
Limit to 10 rows
64 insert into car values('KA031111','Accord',2005);
65 insert into car values('KA041122','MX-5',2019);
66 insert into car values('KA051133','Indica',2010);
67 insert into car values('KA061144','Prius',2015);
68 insert into car values('KA071155','Camry',2020);
69 commit;
70 select * from car;
71
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
reg_num model year
KA031111 Accord 2005
KA041122 MX-5 2019
KA051133 Indica 2010
KA061144 Prius 2015
KA071155 Camry 2020
car 8
Output
Action Output
Time Action Message
10 10:41:21 insert into car values('KA061144','Prius',2015) 1 row(s) affected
11 10:41:24 select * from car LIMIT 0, 10 5 row(s) returned

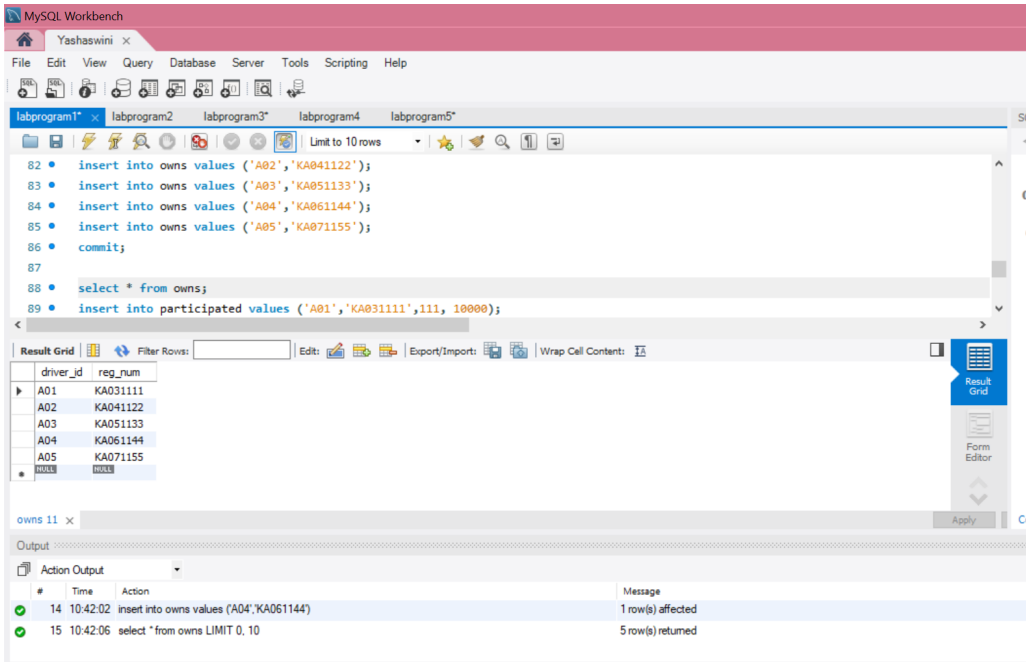
ACCIDENT TABLE



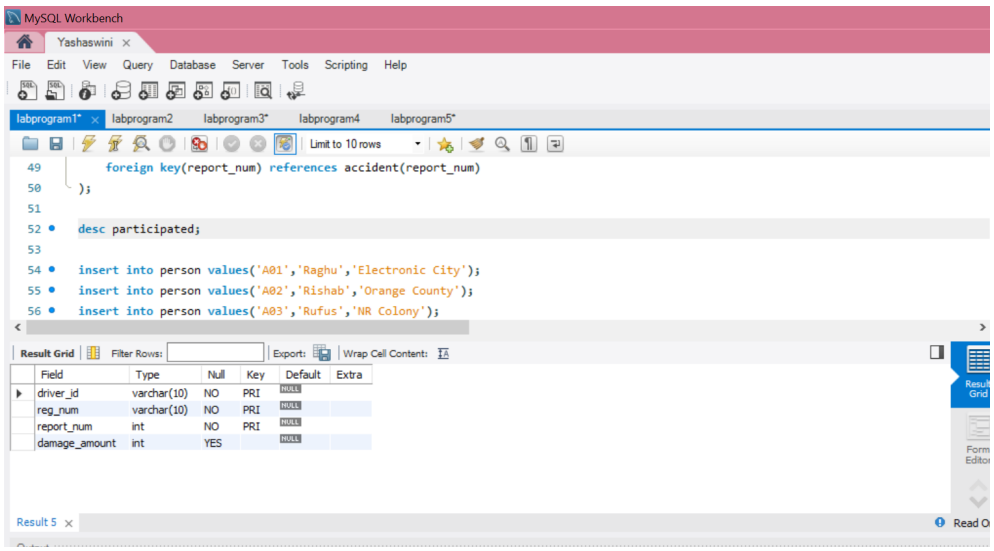

MySQL Workbench
Yashaswini
File Edit View Query Database Server Tools Scripting Help
labprogram1* labprogram2 labprogram3* labprogram4 labprogram5*
Limit to 10 rows
insert into accident values(122,'2020-02-02','Dalhousie Road');
insert into accident values(133,'2020-03-03','Henry Road');
insert into accident values(144,'2020-04-04','Beehive Road');
insert into accident values(155,'2020-05-05','Orange Street');
commit;
select * from accident;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
report_num accident_date location
16 2015-03-08 Domlur
111 2020-01-01 NR Road
122 2020-02-02 Dalhousie Road
133 2020-03-03 Henry Road
144 2020-04-04 Beehive Road
155 2020-05-05 Orange Street
accident 9
Apply
Output
Action Output
Time Action Message
11 10:41:24 select * from car LIMIT 0, 10 5 row(s) returned
12 10:41:35 select * from accident LIMIT 0, 10 6 row(s) returned

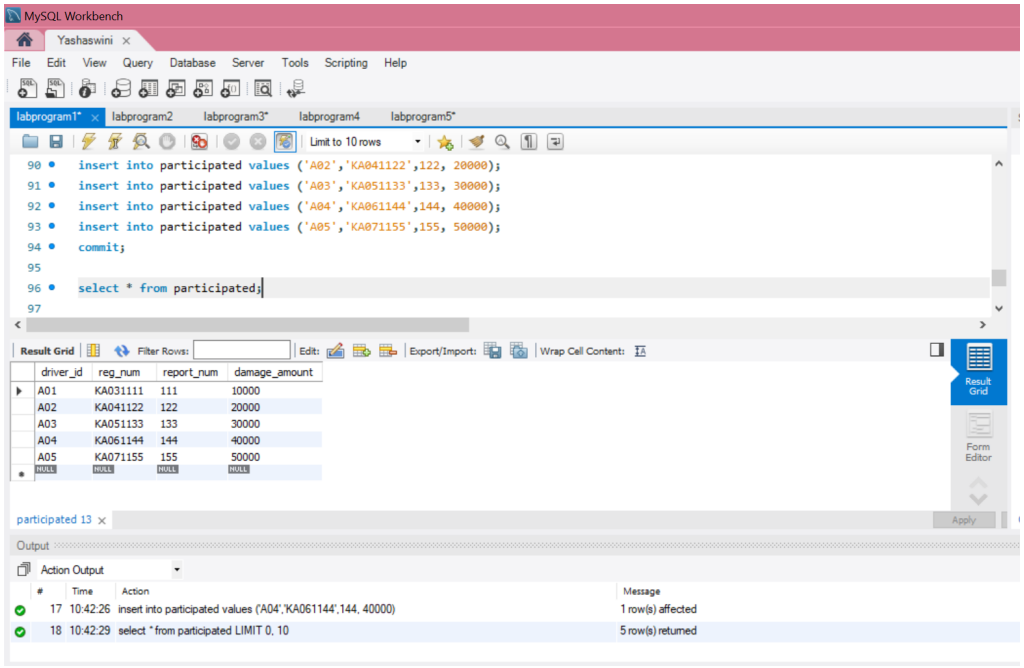
OWNS TABLE


MySQL Workbench
Yashaswini
File Edit View Query Database Server Tools Scripting Help
labprogram1* labprogram2 labprogram3* labprogram4 labprogram5*
Limit to 10 rows
reg_num varchar(10),
primary key(driver_id,reg_num),
foreign key(driver_id) references person(driver_id),
foreign key(reg_num) references car(reg_num)
);
desc owns;
Result Grid Filter Rows: Export: Wrap Cell Content:
Field Type Null Key Default Extra
driver_id varchar(10) NO PRI NULL
reg_num varchar(10) NO PRI NULL



PARTICIPATED TABLE





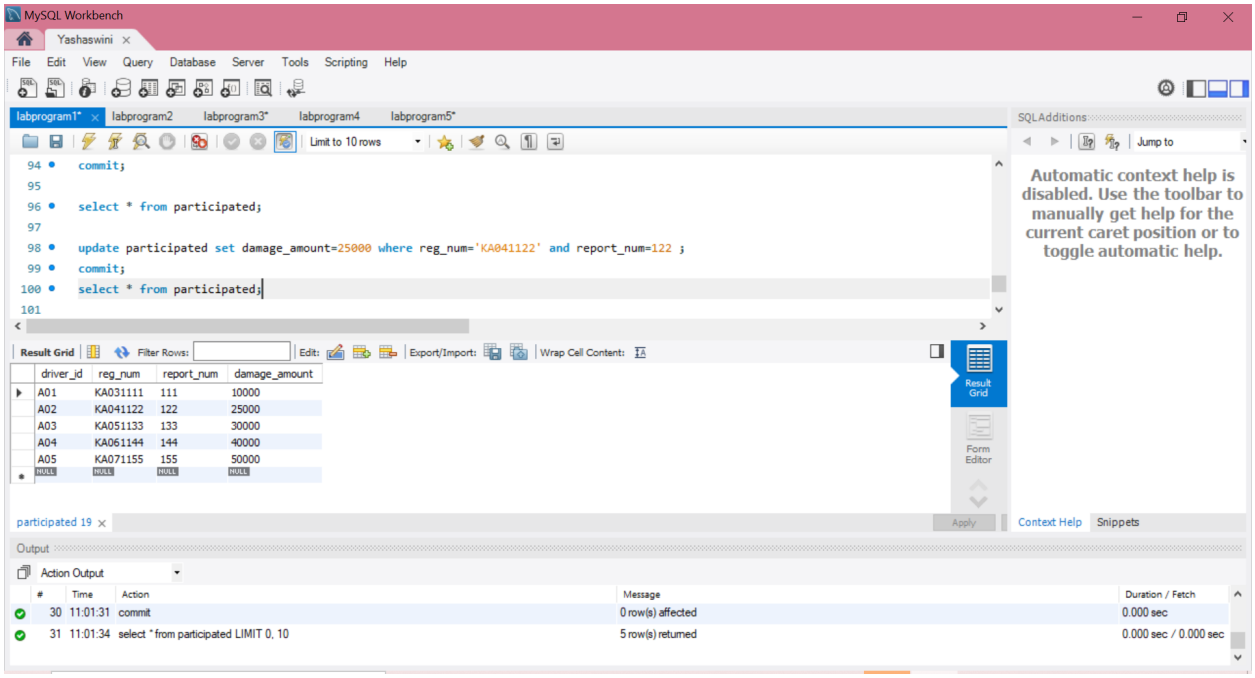
iii.Demonstrate how you

a.Update the damage amount for the car with a specific Regno in the accident with report number 122 to 25000.

update participated set damage_amount=25000 where reg_num='KA041122' and report_num=122 ;

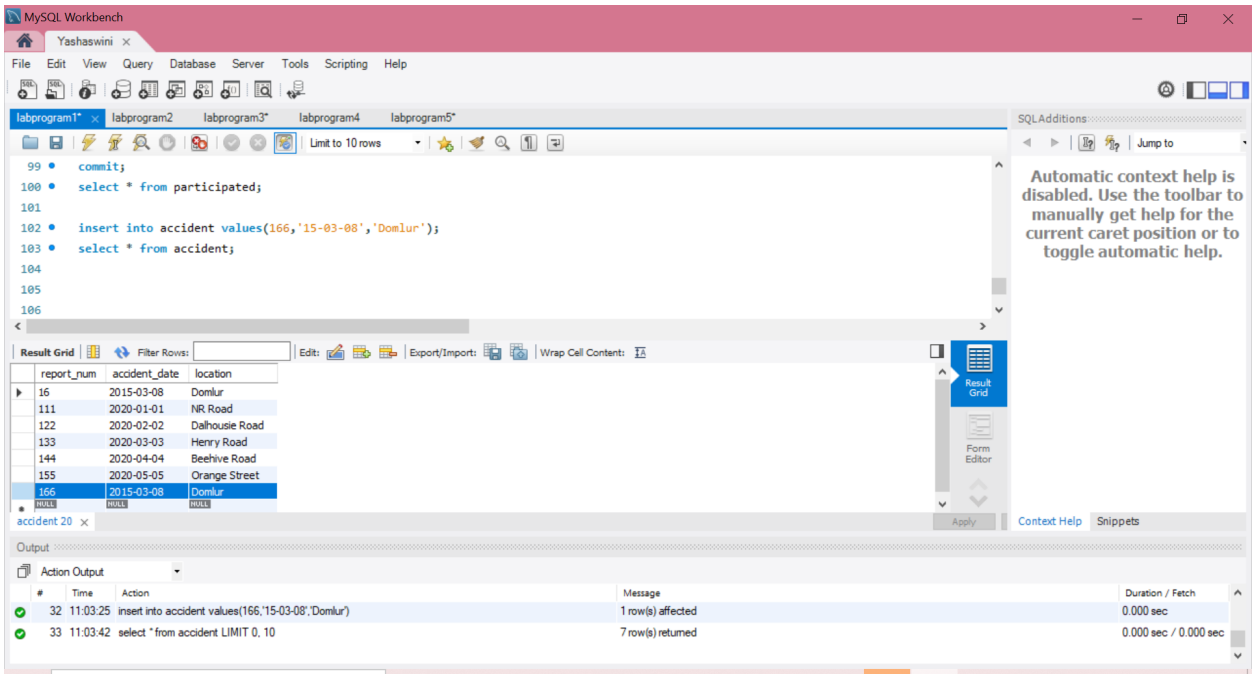
commit;

select * from participated;



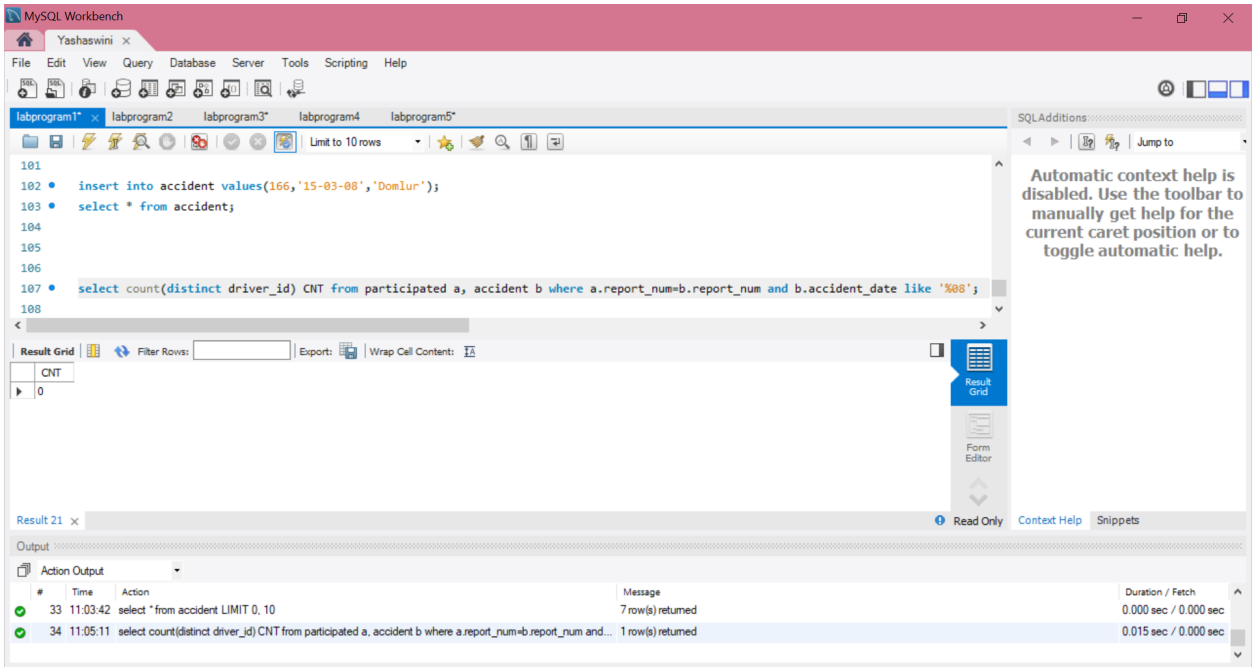
b. Add a new accident to the database.

insert into accident values(166,'15-03-08','Domlur');
select * from accident;



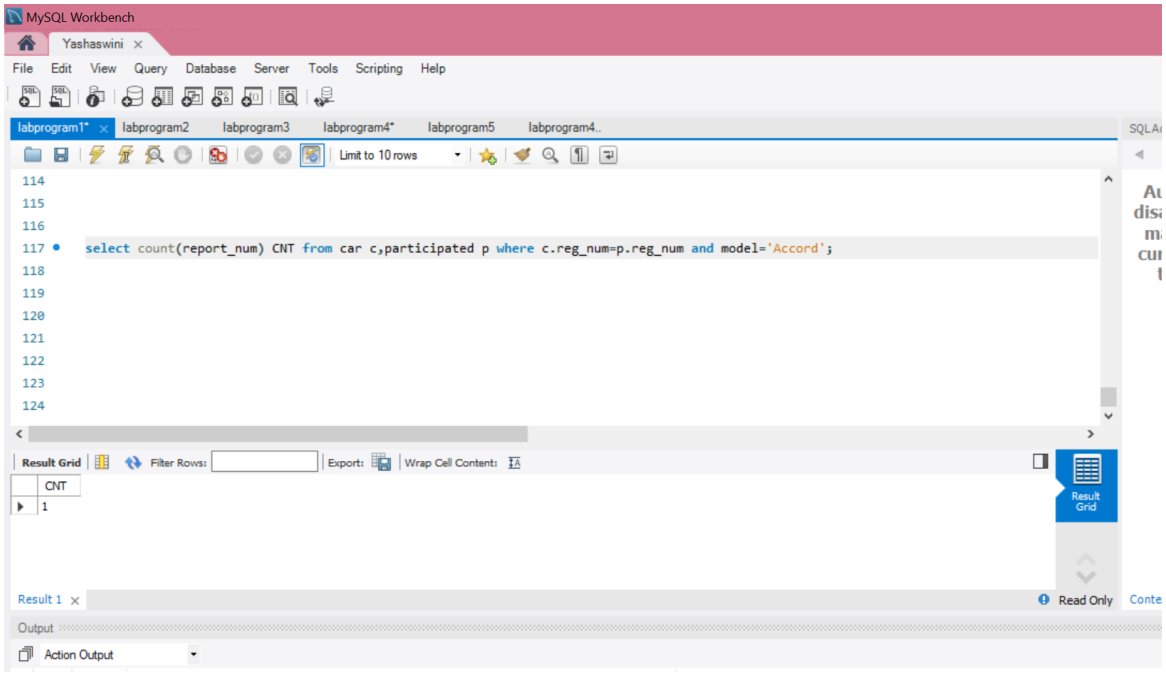
iv. Find the total number of people who owned cars that were involved in accidents in 2008.

```
select count(distinct driver_id) CNT from participated a, accident b where a.report_num=b.report_num
    and b.accident_date like '%08';
```



v. Find the number of accidents in which cars belonging to a specific model were involved.

```
select count(report_num) CNT from car c,participated p where c.reg_num=p.reg_num and
model='Accord';
```



PROGRAM 2: BANKING ENTERPRISE DATABASE

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)

---

create database Banking;

use Banking;

create table Branch(

```
branchname varchar(30),
branchcity varchar(30),
assests real,
primary key (branchname)
);
desc Branch;

create table BankAccount(
accno integer,
branchname varchar(30),
balance real,
primary key (accno),
foreign key (branchname) references Branch(branchname)
);

desc BankAccount;

create table BankCustomer(
customername varchar(30),
customerstreet varchar(30),
customercity varchar(30),
primary key(customername)
);

desc BankCustomer;

create table Depositer(
customername varchar(30),
```

```
accno integer,
primary key (customername, accno),
foreign key (customername) references BankCustomer (customername),
foreign key(accno) references BankAccount(accno)
);

desc Depositer;

create table Loan (
loannumber int,
branchname varchar(30),
amount real,
primary key (loannumber),
foreign key (branchname) references Branch(branchname)
);

desc Loan;

insert into Branch values ('SBI_Chamrajpet','Bangalore',50000);
insert into Branch values ('SBI_ResidencyRoad','Bangalore',10000);
insert into Branch values ('SBI_ShivajiRoad','Bombay',20000);
insert into Branch values ('SBI_ParlimentRoad','Delhi',10000);
insert into Branch values ('SBI_Jantarmantar','Delhi',20000);

select * from Branch;

insert into Loan values(2,'SBI_ResidencyRoad',2000);
```

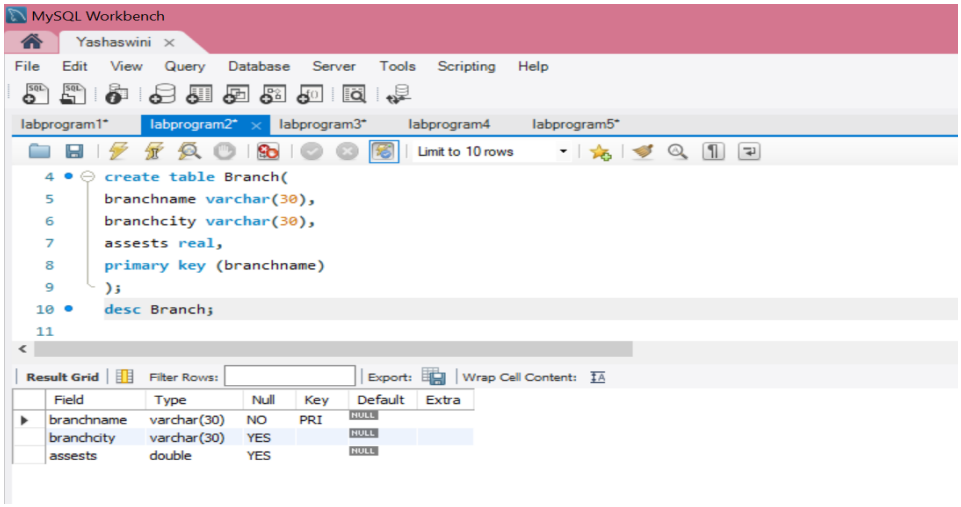
```
insert into Loan values(1,'SBI_Chamrajpet',1000);
insert into Loan values(3,'SBI_ShivajiRoad', 3000);
insert into Loan values(4,'SBI_ParlimentRoad',4000);
insert into Loan values(5,'SBI_Jantarmantar',5000);

select * from Loan;

insert into BankAccount values(11,'SBI_Jantarmantar',2000);
insert into BankAccount values(10,'SBI_ResidencyRoad',5000);
insert into BankAccount values(9,'SBI_ParlimentRoad',3000);
insert into BankAccount values(8,'SBI_ResidencyRoad',4000);
insert into BankAccount values(6,'SBI_ShivajiRoad',4000);
insert into BankAccount values(5,'SBI_Jantarmantar',8000);
insert into BankAccount values(4,'SBI_ParlimentRoad',9000);
insert into BankAccount values(3,'SBI_ShivajiRoad',6000);
insert into BankAccount values(2,'SBI_ResidencyRoad',5000);
insert into BankAccount values(1,'SBI_Chamrajpet',2000);

commit;
select * from BankAccount;

insert into Depositer values('Avinash',1);
insert into Depositer values('Dinesh',2);
insert into Depositer values('Nikil',4);
insert into Depositer values('Ravi',5);
insert into Depositer values('Avinash',8);
insert into Depositer values('Nikil',9);
insert into Depositer values('Dinesh',10);
```

```
insert into Depositer values('Nikil',11);

select * from Depositer;

insert into BankCustomer values('Avinash','Bull_Temple_Road','Bangalore');
insert into BankCustomer values('Dinesh','Bannergatta_Road','Bangalore');
insert into BankCustomer values('Mohan','NationalColege_Road','Bangalore');
insert into BankCustomer values('Nikil','Akbar_Road','Delhi');
insert into BankCustomer values('Ravi','Prithiviraj_Road','Delhi');

select *from BankCustomer;
```

i. Create the above tables by properly specifying the primary keys and the foreign keys.

ii. Enter at least five tuples for each relation.

TABLE BRANCH



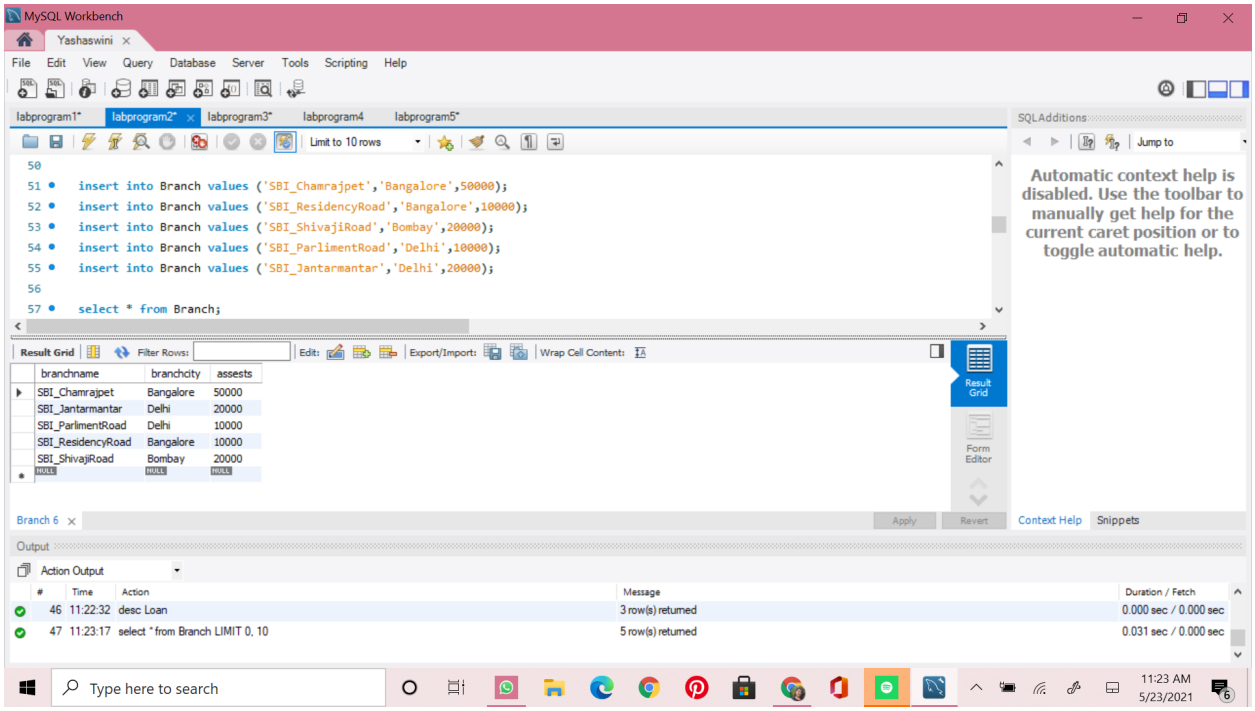
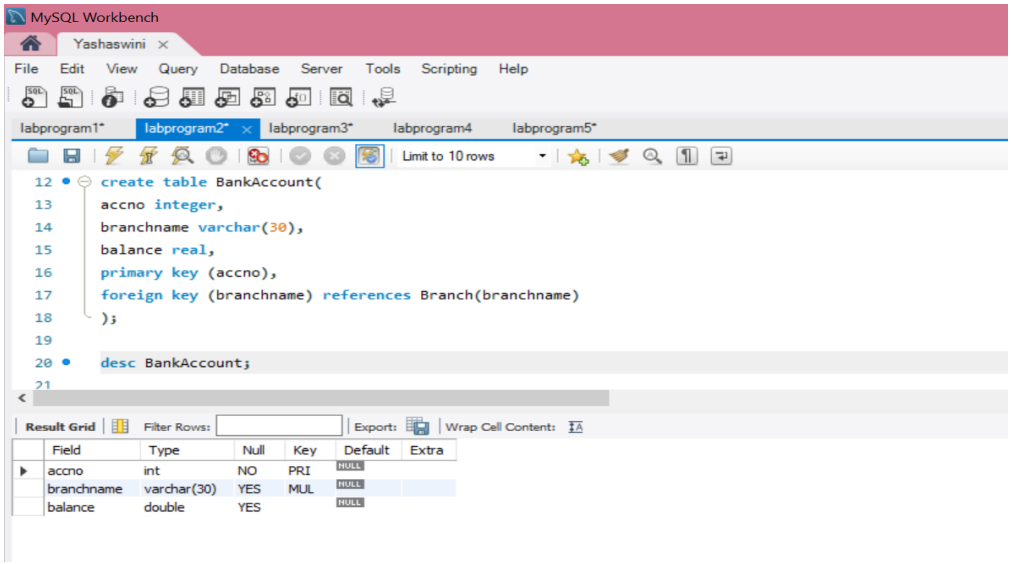


TABLE BANK ACCOUNT



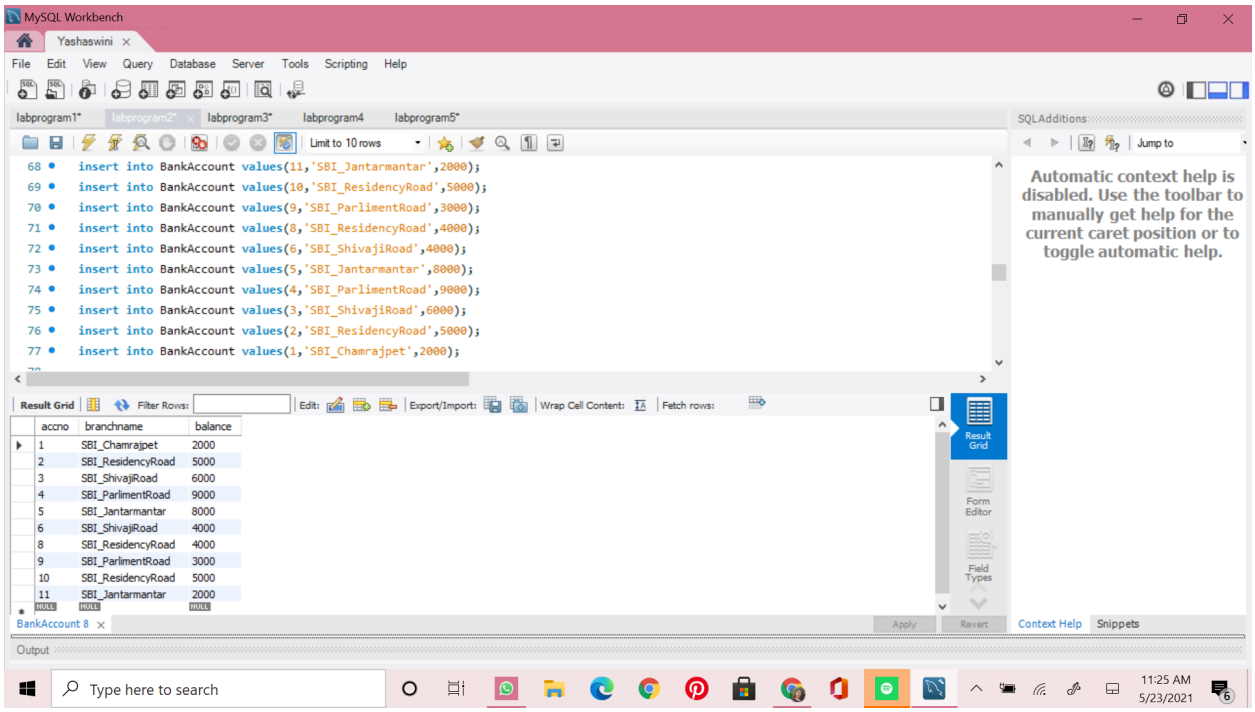
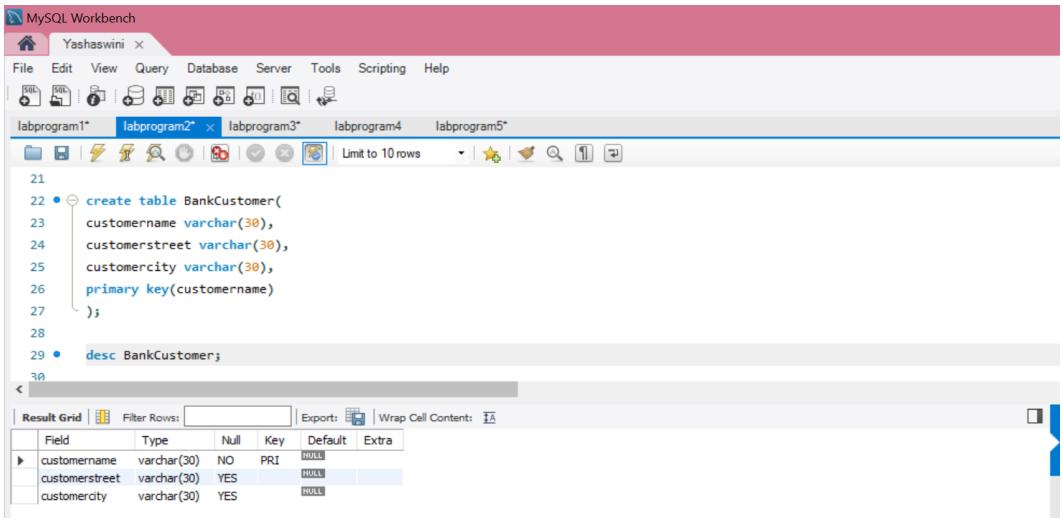


TABLE BANK CUSTOMER



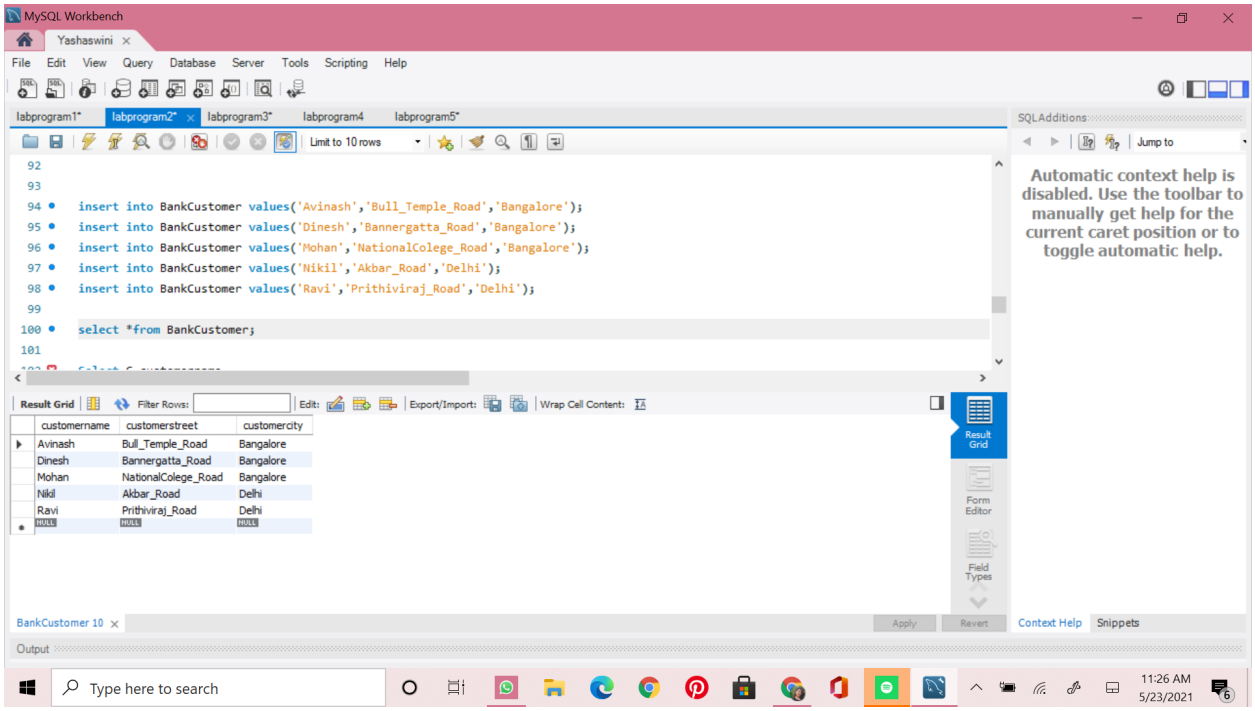
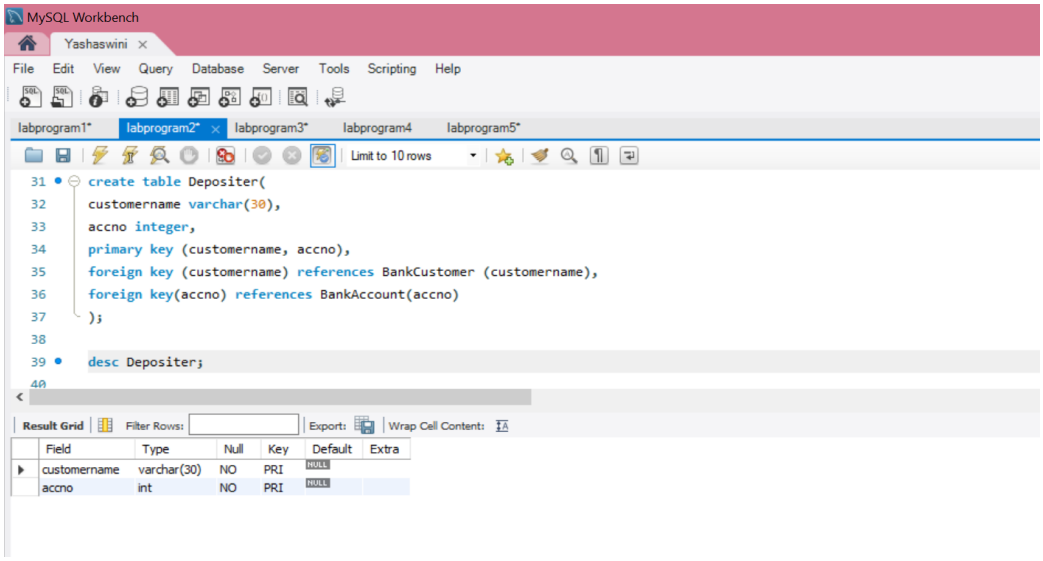


TABLE DEPOSITOR


MySQL Workbench
Yashaswini
File Edit View Query Database Server Tools Scripting Help
labprogram1* labprogram2* labprogram3* labprogram4 labprogram5*
Limit to 10 rows
create table Depositer(
customername varchar(30),
accno integer,
primary key (customername, accno),
foreign key (customername) references BankCustomer (customername),
foreign key(accno) references BankAccount(accno)
);
desc Depositer;
Result Grid Filter Rows: Export: Wrap Cell Content:
Field Type Null Key Default Extra
customername varchar(30) NO PRI NULL
accno int NO PRI NULL

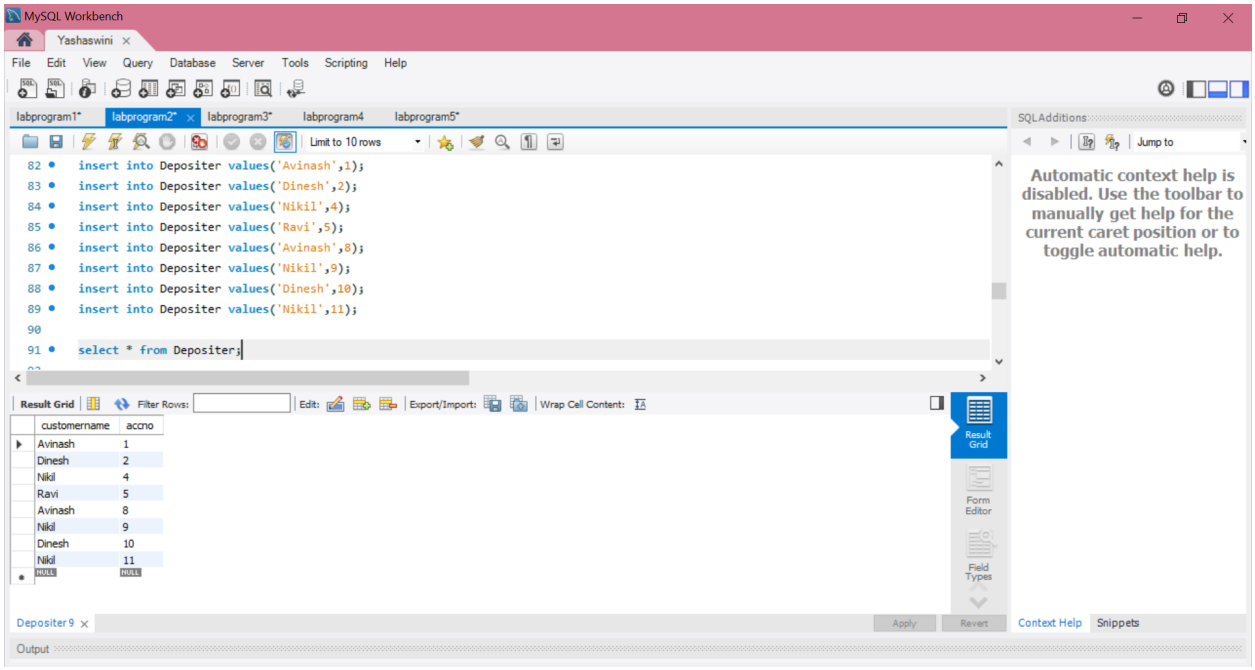
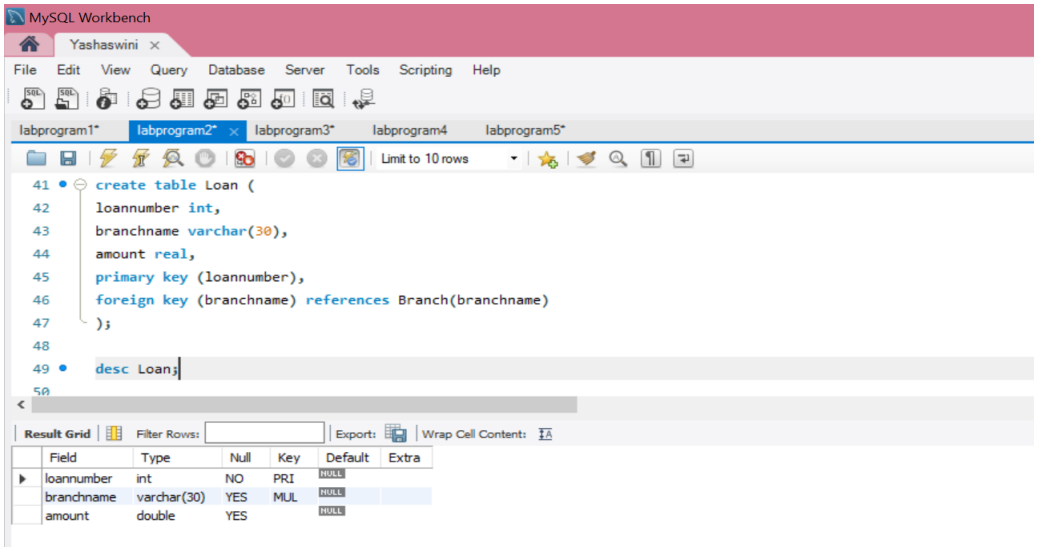
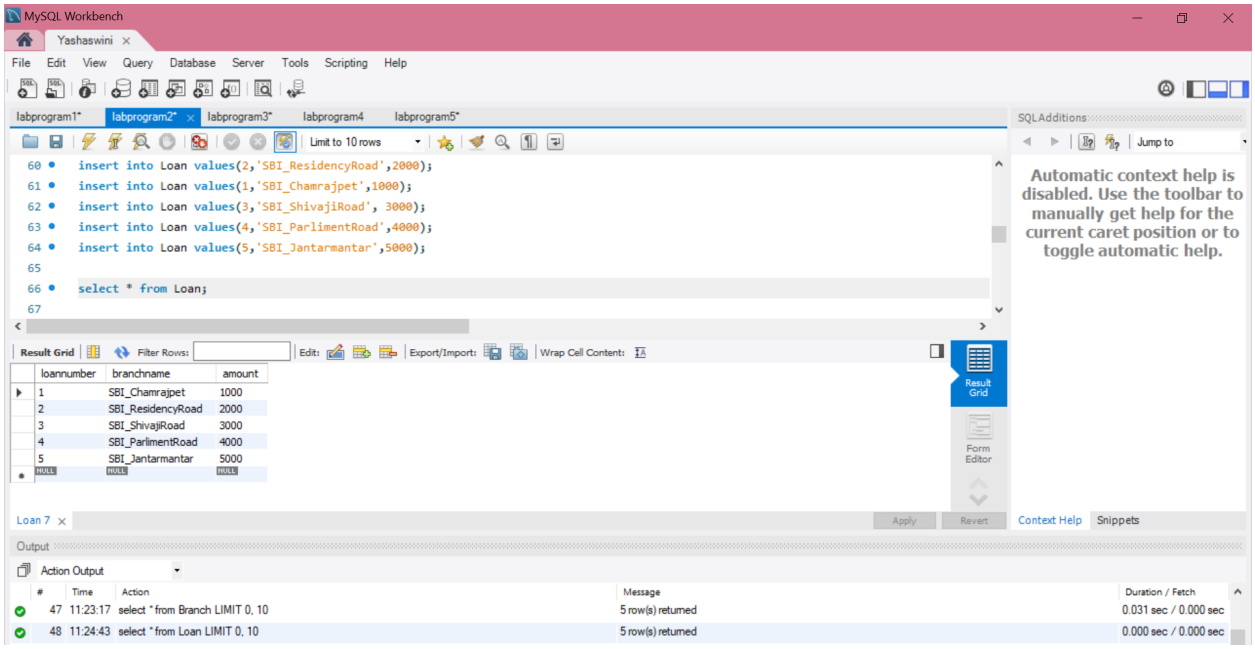

MySQL Workbench
Yashaswini
File Edit View Query Database Server Tools Scripting Help
labprogram1* labprogram2* labprogram3* labprogram4 labprogram5*
Limit to 10 rows
insert into Depositer values('Avinash',1);
insert into Depositer values('Dinesh',2);
insert into Depositer values('Nikil',4);
insert into Depositer values('Ravi',5);
insert into Depositer values('Avinash',8);
insert into Depositer values('Nikil',9);
insert into Depositer values('Dinesh',10);
insert into Depositer values('Nikil',11);
select * from Depositer;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
customername accno
Avinash 1
Dinesh 2
Nikil 4
Ravi 5
Avinash 8
Nikil 9
Dinesh 10
Nikil 11
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Result Grid
Form Editor
Field Types
Depositer 9
Apply
Revert
Context Help
Snippets
Output

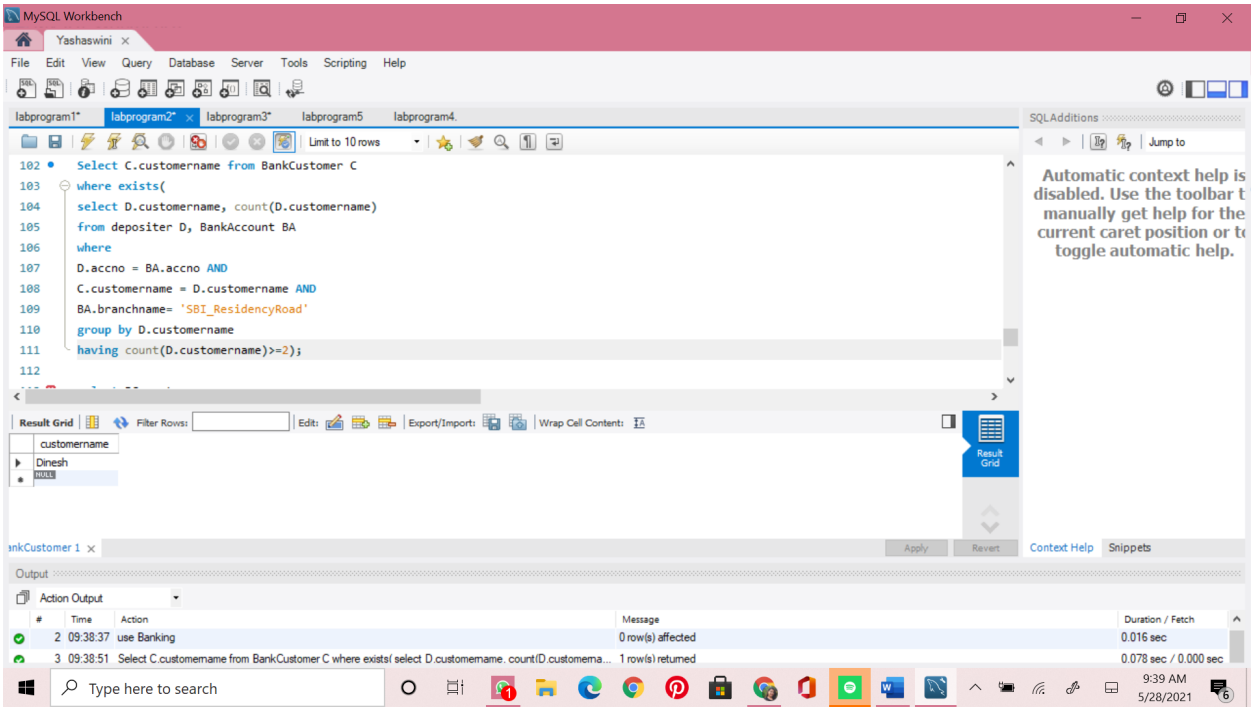
TABLE LOAN


MySQL Workbench
Yashaswini
File Edit View Query Database Server Tools Scripting Help
labprogram1* labprogram2* labprogram3* labprogram4 labprogram5*
Limit to 10 rows
41 create table Loan (
42 loannumber int,
43 branchname varchar(30),
44 amount real,
45 primary key (loannumber),
46 foreign key (branchname) references Branch(branchname)
47);
48
49 desc Loan;
50
Result Grid Filter Rows: Export: Wrap Cell Content:
Field Type Null Key Default Extra
loannumber int NO PRI NULL
branchname varchar(30) YES MUL NULL
amount double YES NULL


MySQL Workbench
Yashaswini
File Edit View Query Database Server Tools Scripting Help
labprogram1* labprogram2* labprogram3* labprogram4 labprogram5*
Limit to 10 rows
60 insert into Loan values(2,'SBI_ResidencyRoad',2000);
61 insert into Loan values(1,'SBI_Chamrajpet',1000);
62 insert into Loan values(3,'SBI_ShivajiRoad', 3000);
63 insert into Loan values(4,'SBI_ParlimentRoad',4000);
64 insert into Loan values(5,'SBI_Jantarmantar',5000);
65
66 select * from Loan;
67
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
loannumber branchname amount
1 SBI_Chamrajpet 1000
2 SBI_ResidencyRoad 2000
3 SBI_ShivajiRoad 3000
4 SBI_ParlimentRoad 4000
5 SBI_Jantarmantar 5000
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Result Grid
Form Editor
Loan 7
Apply Revert
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
47 11:23:17 select * from Branch LIMIT 0, 10 5 row(s) returned 0.031 sec / 0.000 sec
48 11:24:43 select * from Loan LIMIT 0, 10 5 row(s) returned 0.000 sec / 0.000 sec

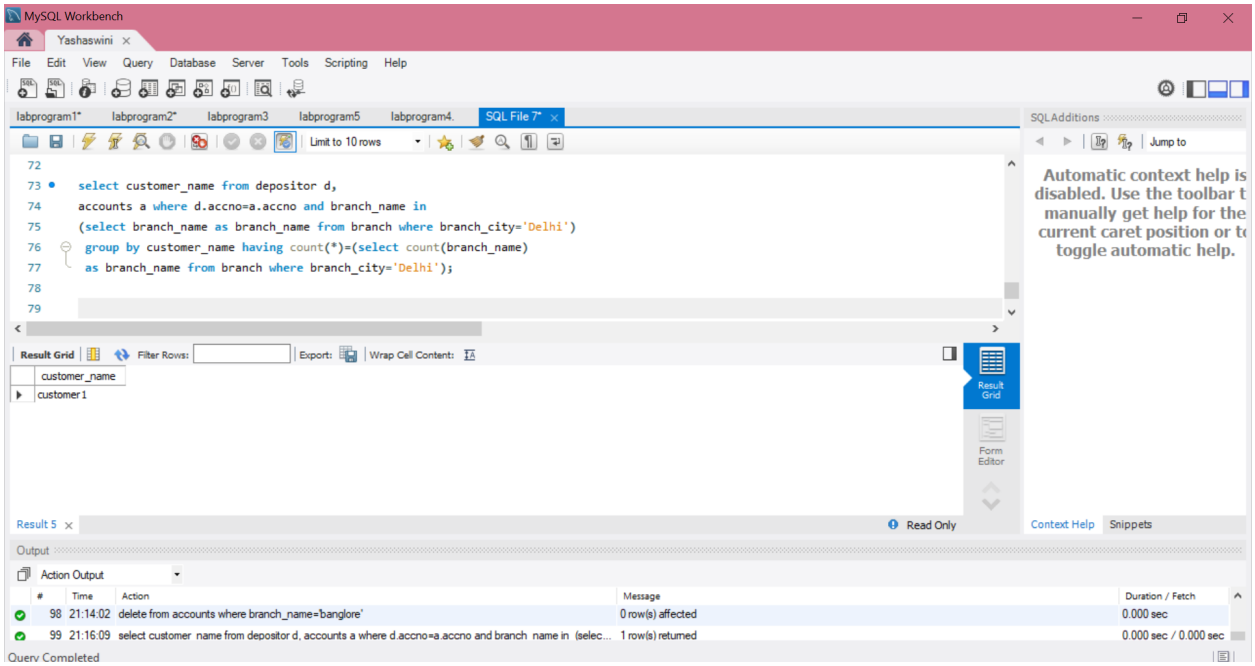
iii. Find all the customers who have at least two accounts at the Main branch.

```
Select C.customername from BankCustomer C
where exists(
select D.customername, count(D.customername)
from depositer D, BankAccount BA
where
D.accno = BA.accno AND
C.customername = D.customername AND
BA.branchname= 'SBI_ResidencyRoad'
group by D.customername
having count(D.customername)>=2);
```



iv. Find all the customers who have an account at all the branches located in a specific city. v.

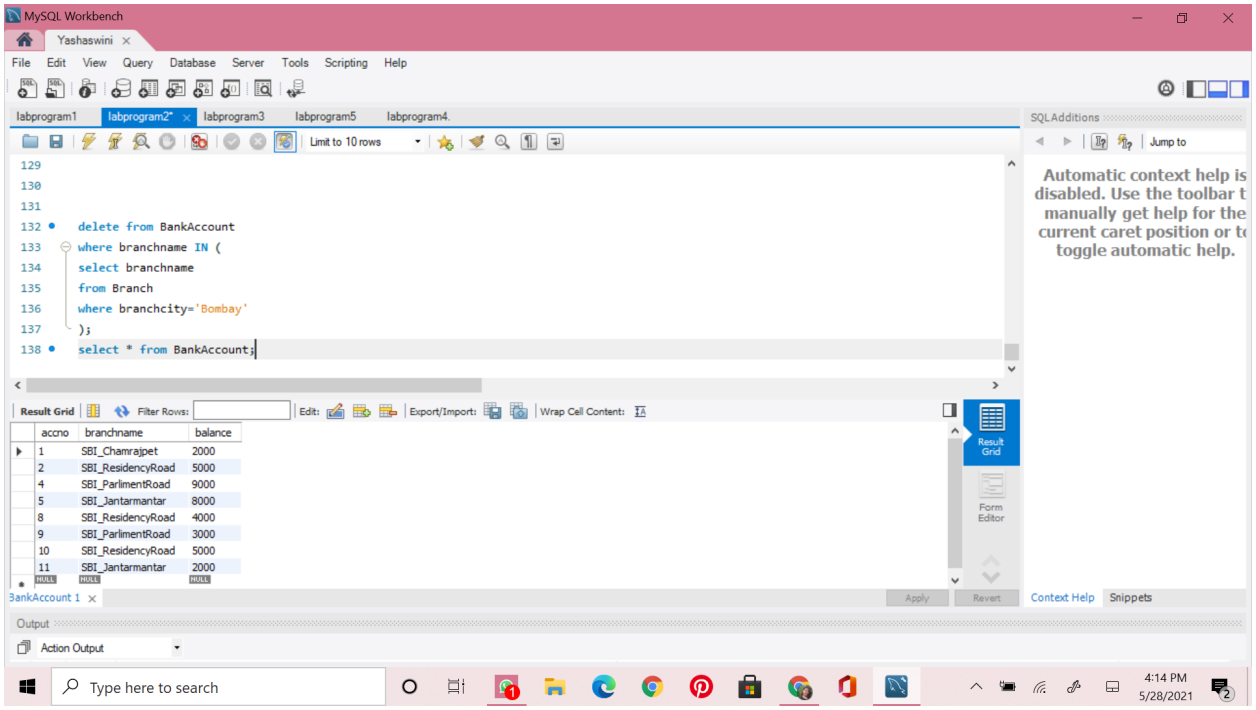
```
select customer_name from depositor d,
accounts a where d.accno=a.accno and branch_name in
(select branch_name as branch_name from branch where branch_city='Delhi')
 group by customer_name having count(*)=(select count(branch_name)
 as branch_name from branch where branch_city='Delhi');
```



v.Demonstrate how you delete all account tuples at every branch located in a specific city.

(BOMBAY)

```
delete from BankAccount
where branchname IN (
select branchname
from Branch
where branchcity='Bombay'
);
select * from BankAccount;
```



PROGRAM 3: 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.

```
create database Supplier;

use Supplier;


create table SUPPLIERS (

sid integer
```

```
primary key,
sname varchar(20),
city varchar(20)
);

desc SUPPLIERS;

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

desc PARTS;

create table CATALOG (
sid integer,
pid integer,
foreign key(sid) references SUPPLIERS(sid),
foreign key(pid) references PARTS(pid),
cost float(6),
primary key(sid,pid)
);

desc CATALOG;

insert into suppliers value (10001,'Acme Widget','Bangalore');
insert into suppliers value (10002,'Johns','Kolkata');
```

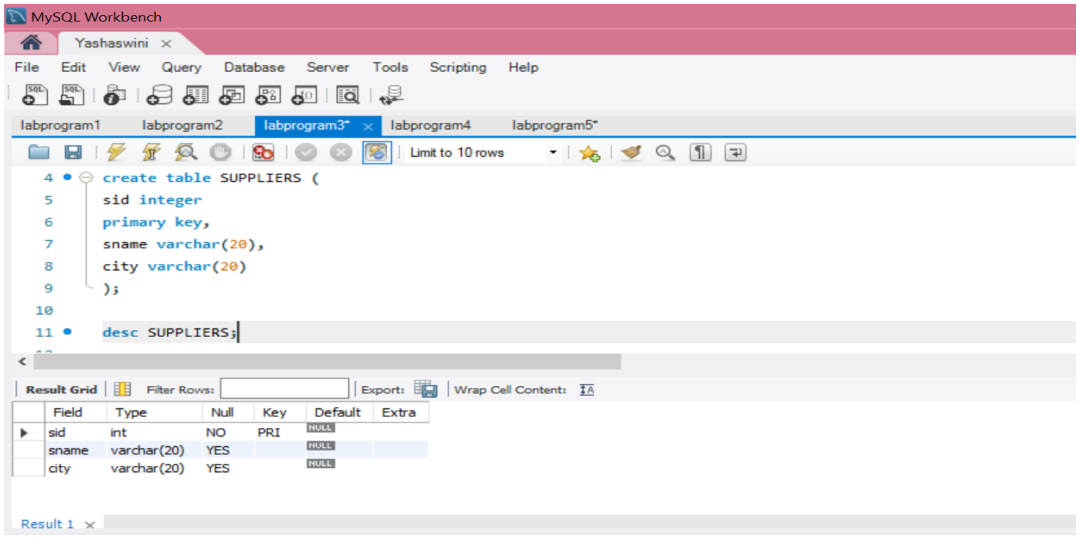
```
insert into suppliers value (10003,'Vimal','Mumbai');
insert into suppliers value (10004,'Reliance','Delhi');
insert into suppliers value(10005,'Mahindra','Mumbai');

select * from SUPPLIERS;

commit;

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;

commit;

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');
```

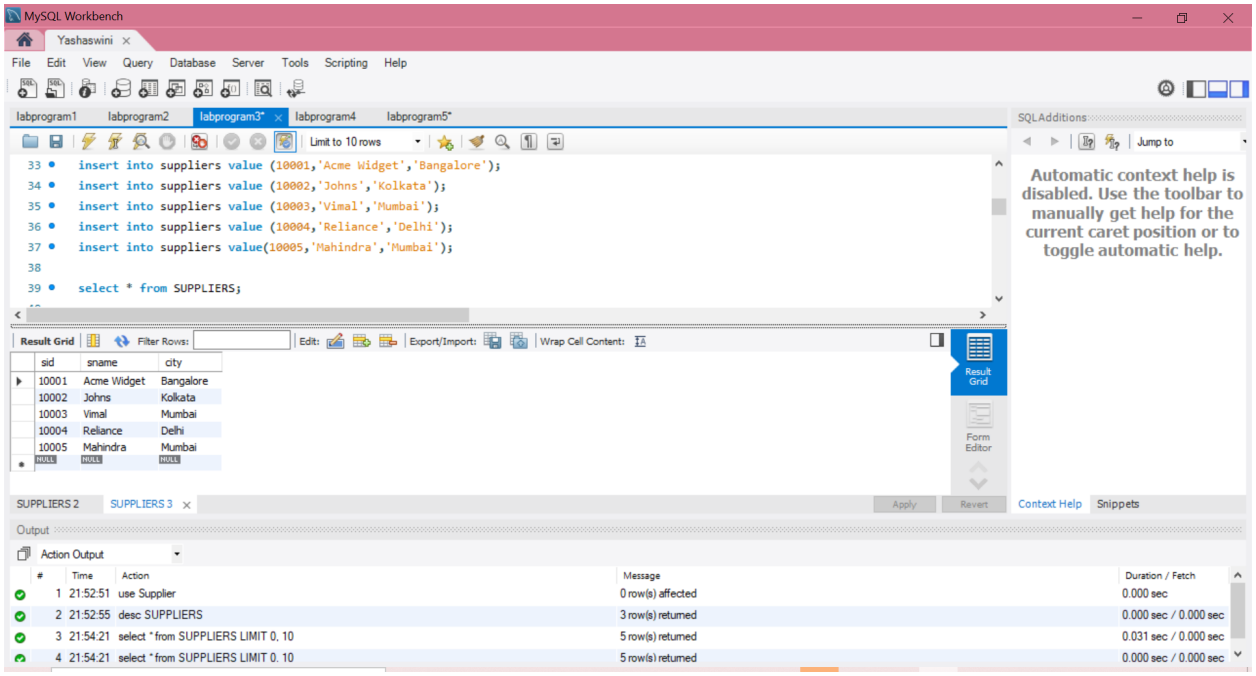
insert into CATALOG values(10004,'20003','40');

select * from CATALOG;

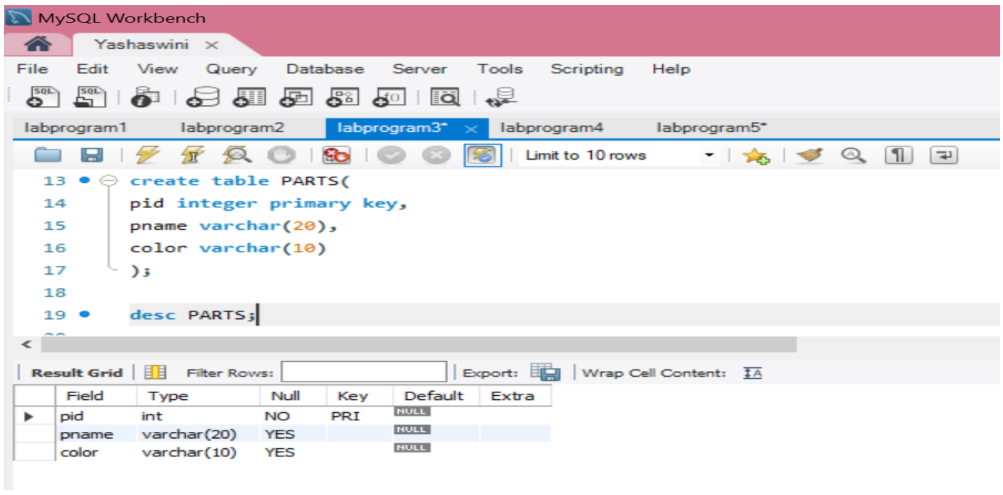
commit;

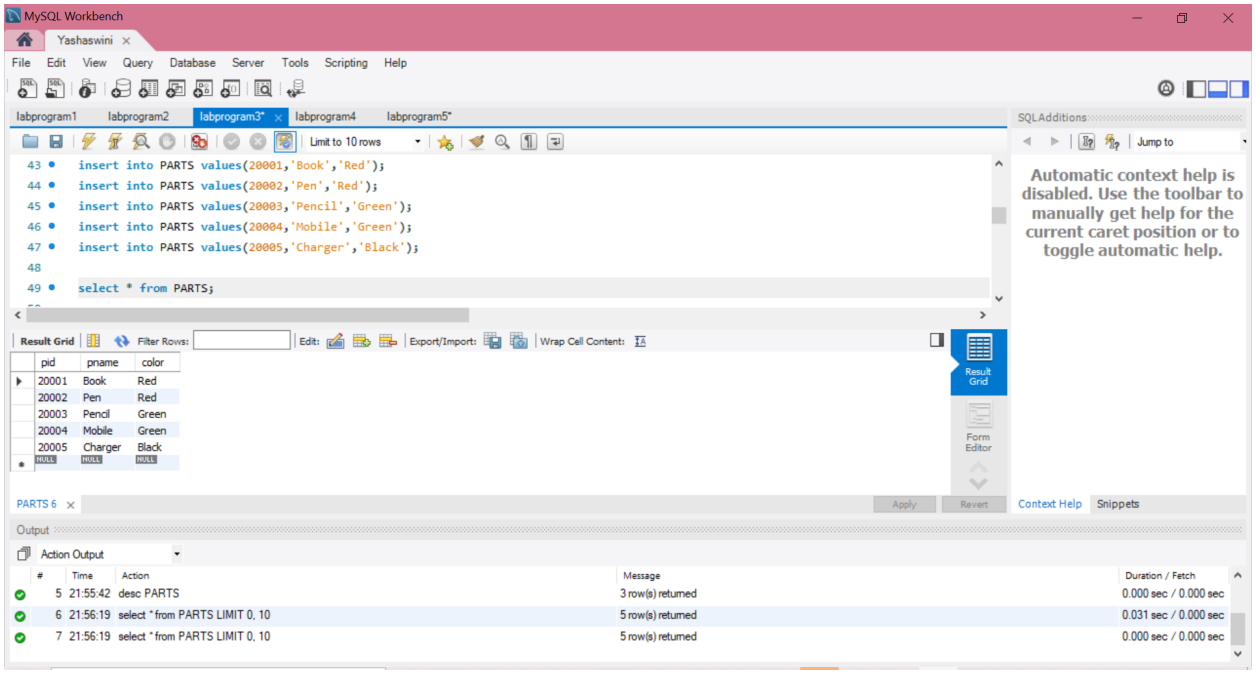
1) SUPPLIERS



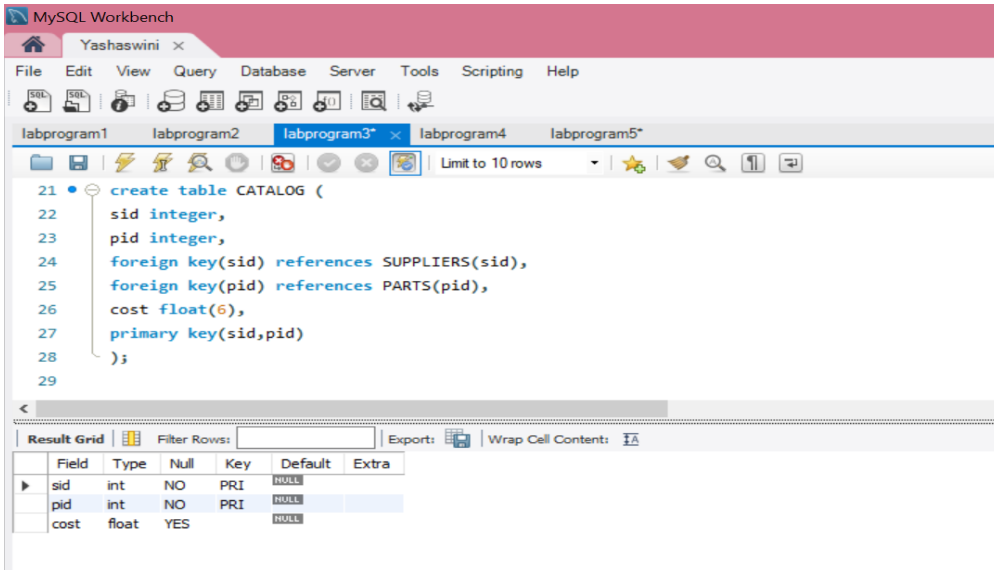


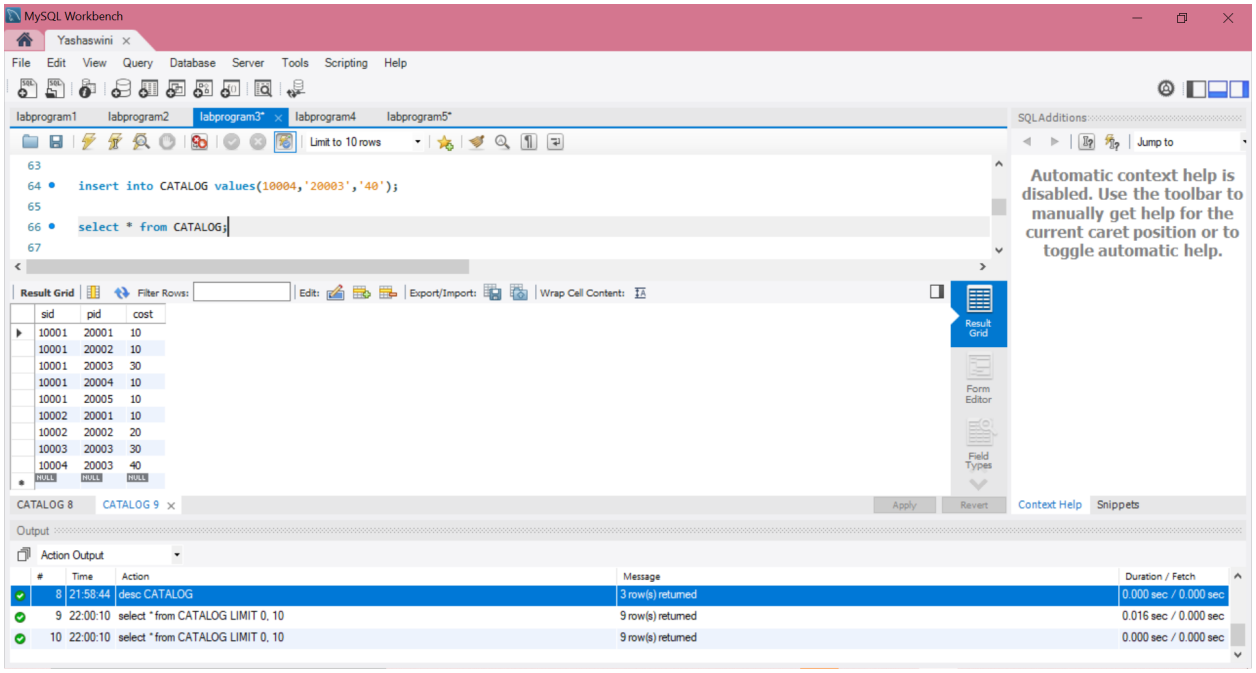
2) PARTS TABLE





3) CATALOG TABLE





Write the following queries in SQL:

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

insert into parts values(5,'tiles','blue');

select p.pname from parts p where p.pid in

(select pid from catalog c group by c.pid having count(c.sid)>0);

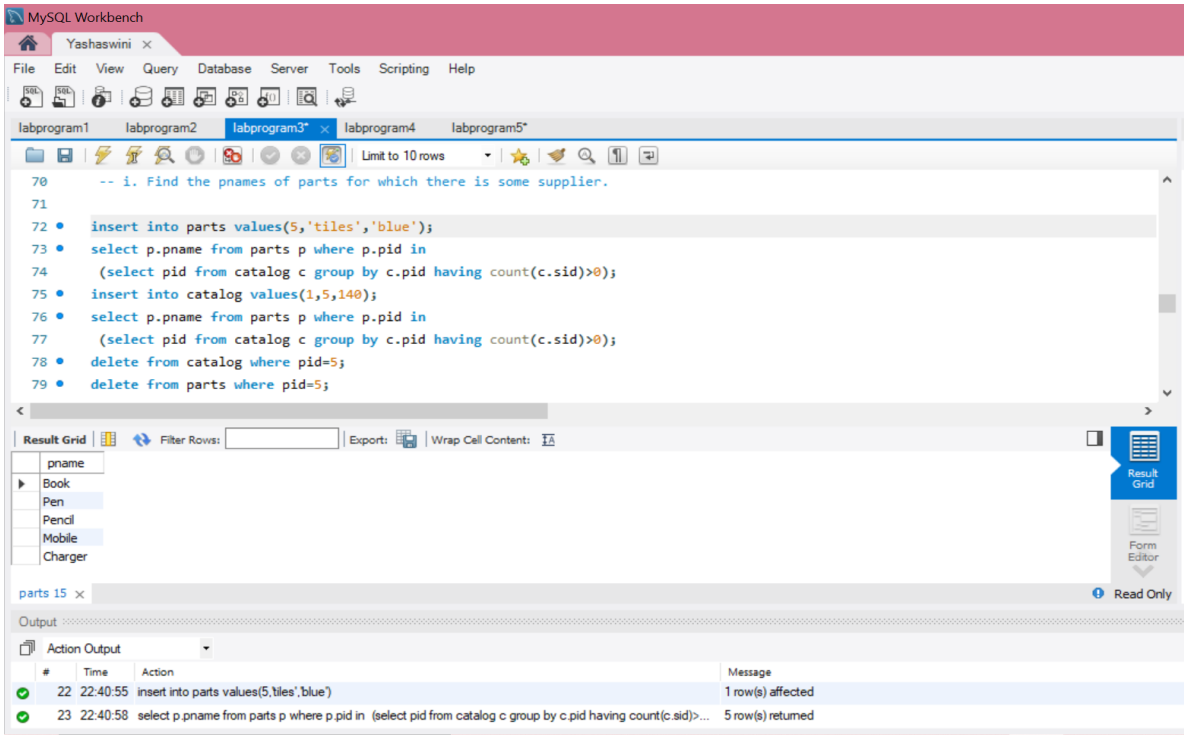
insert into catalog values(1,5,140);

select p.pname from parts p where p.pid in

(select pid from catalog c group by c.pid having count(c.sid)>0);

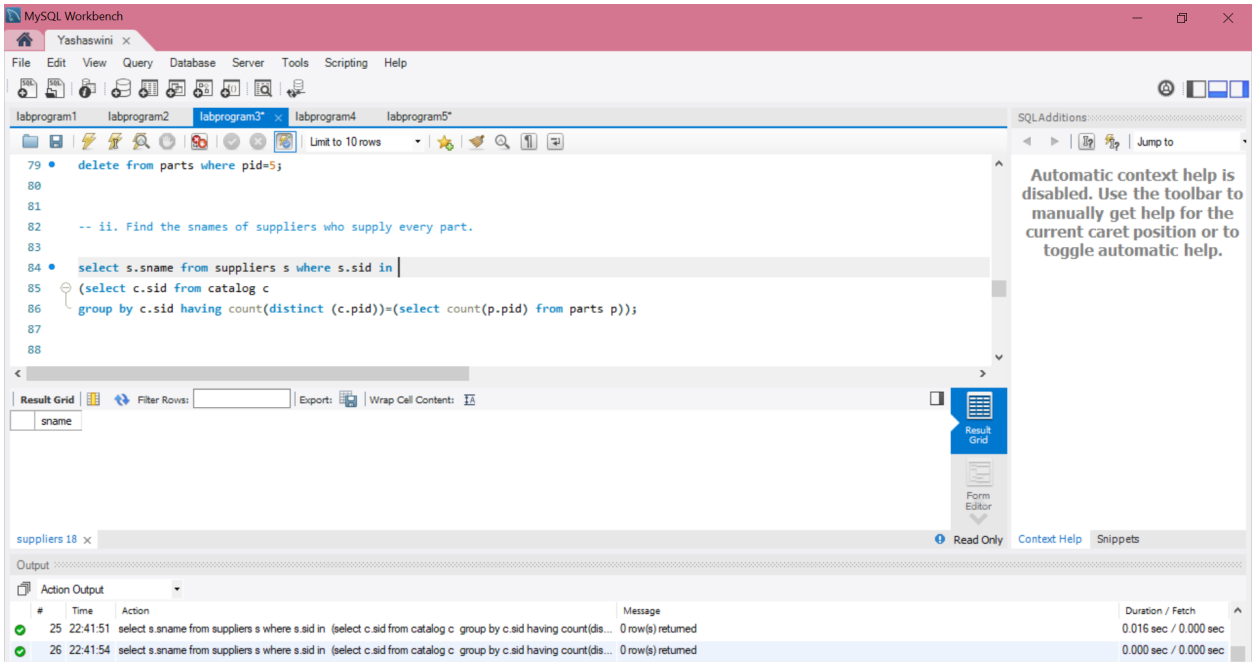
delete from catalog where pid=5;

delete from parts where pid=5;



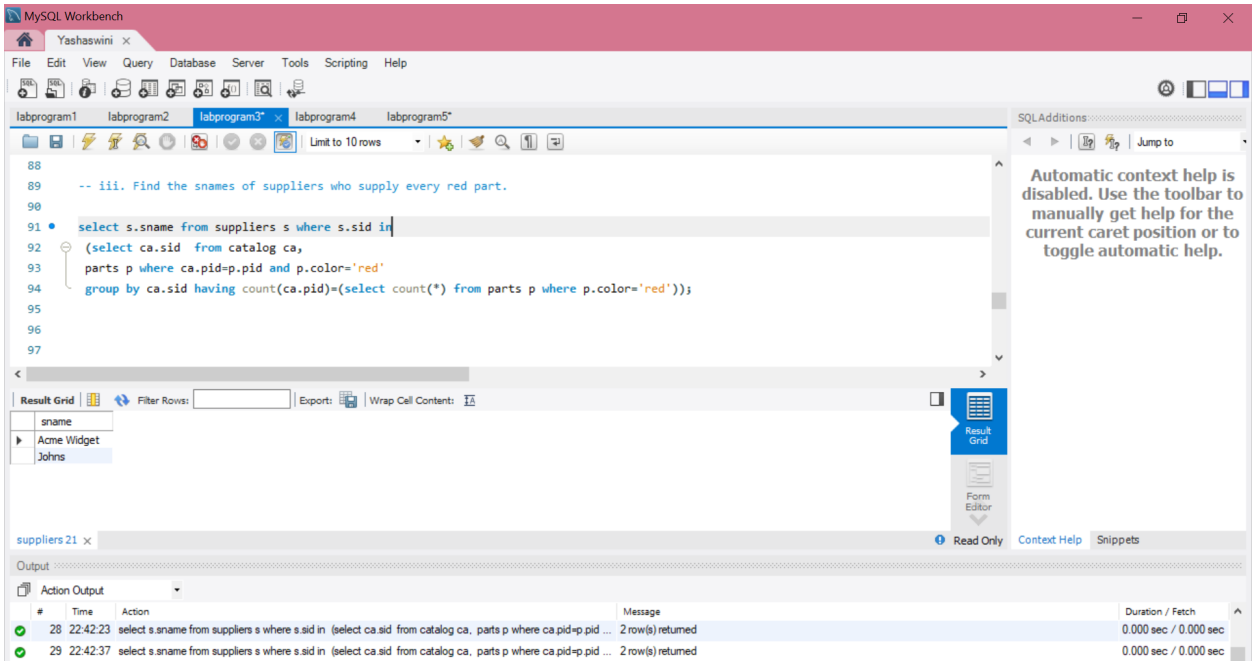
ii. Find the snames of suppliers who supply every part.

select s.sname from suppliers s where s.sid in
(select c.sid from catalog c
group by c.sid having count(distinct (c.pid))=(select count(p.pid) from parts p));



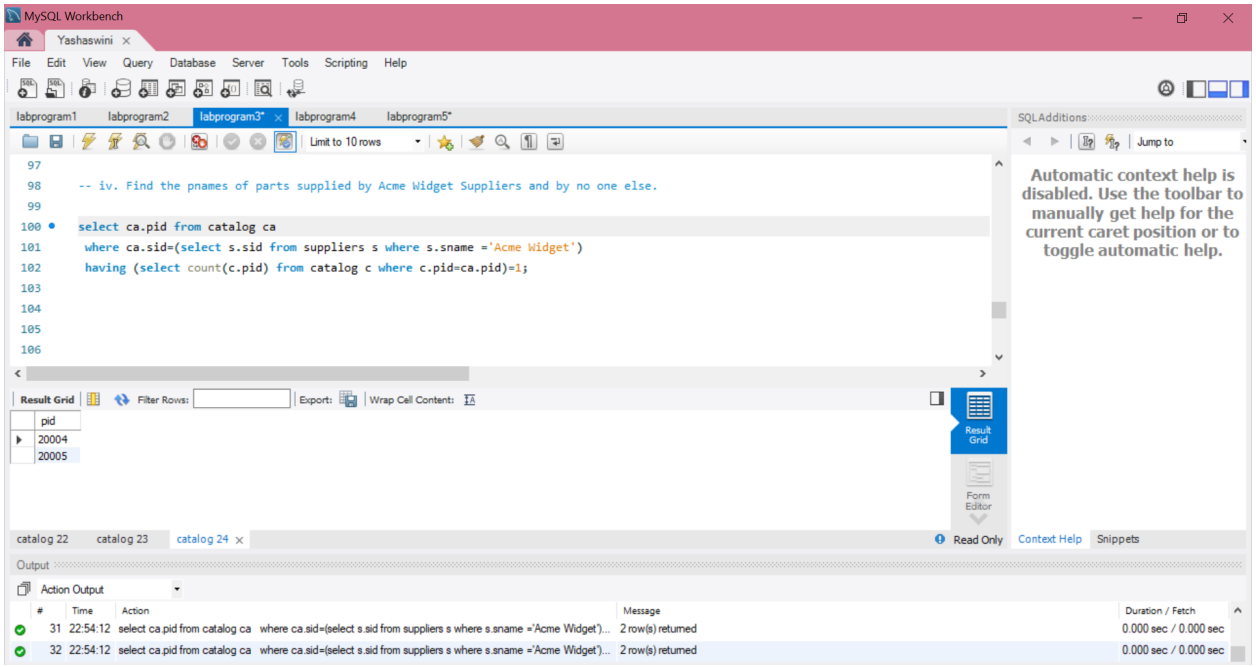
iii. Find the snames of suppliers who supply every red part.

```
select s.sname from suppliers s where s.sid in
 (select ca.sid  from catalog ca,
 parts p where ca.pid=p.pid and p.color='red'
 group by ca.sid having count(ca.pid)=(select count(*) from parts p where p.color='red'));
```



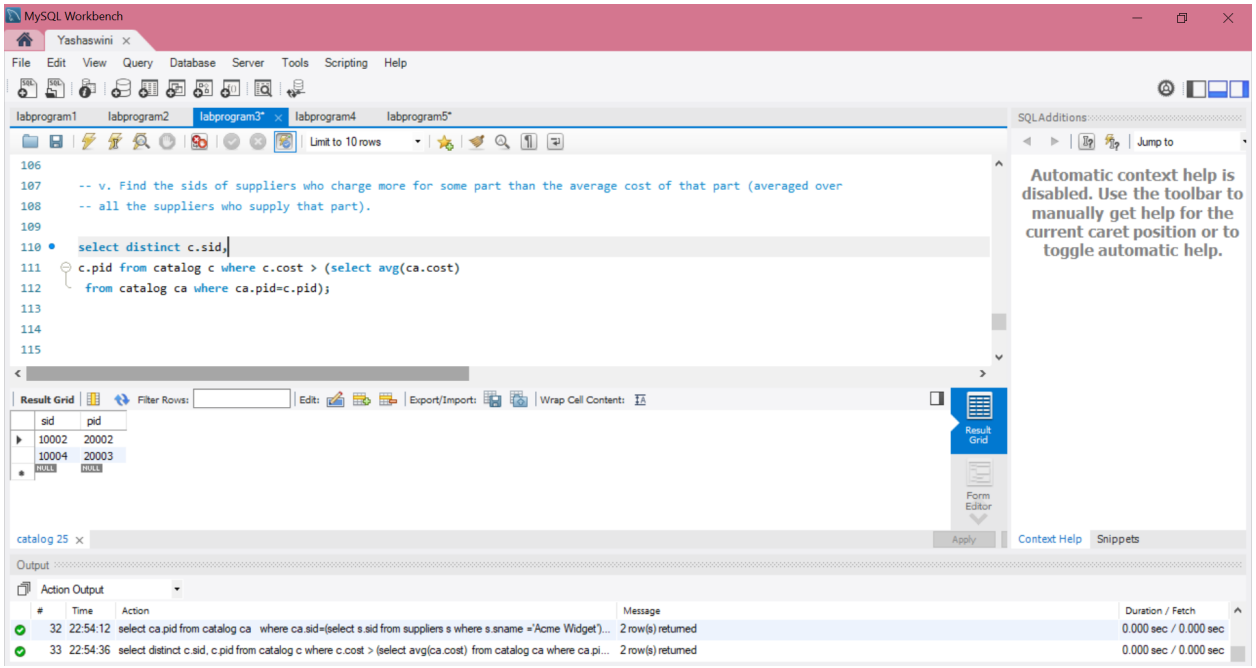
iv. Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.

```
select ca.pid from catalog ca
 where ca.sid=(select s.sid from suppliers s where s.sname ='Acme Widget')
 having (select count(c.pid) from catalog c where c.pid=ca.pid)=1;
```



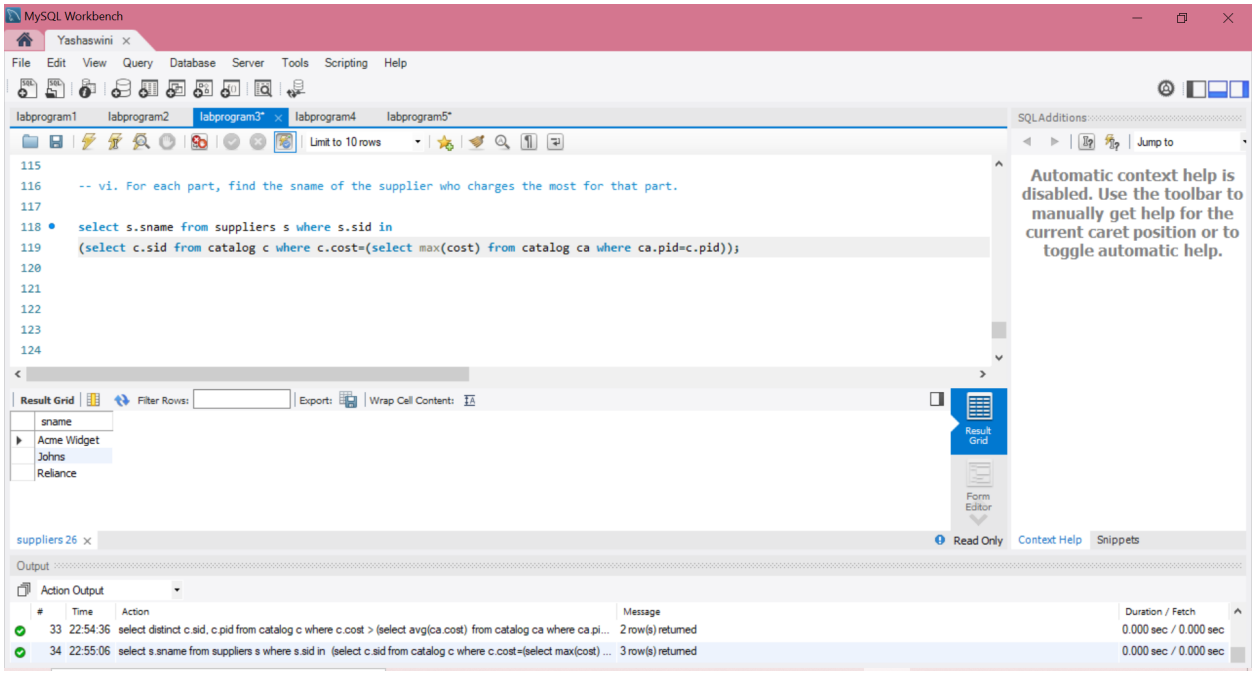
v. Find the sids of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part).

```
select distinct c.sid, c.pid from catalog c where c.cost > (select avg(ca.cost) from catalog ca where ca.pid=c.pid);
```



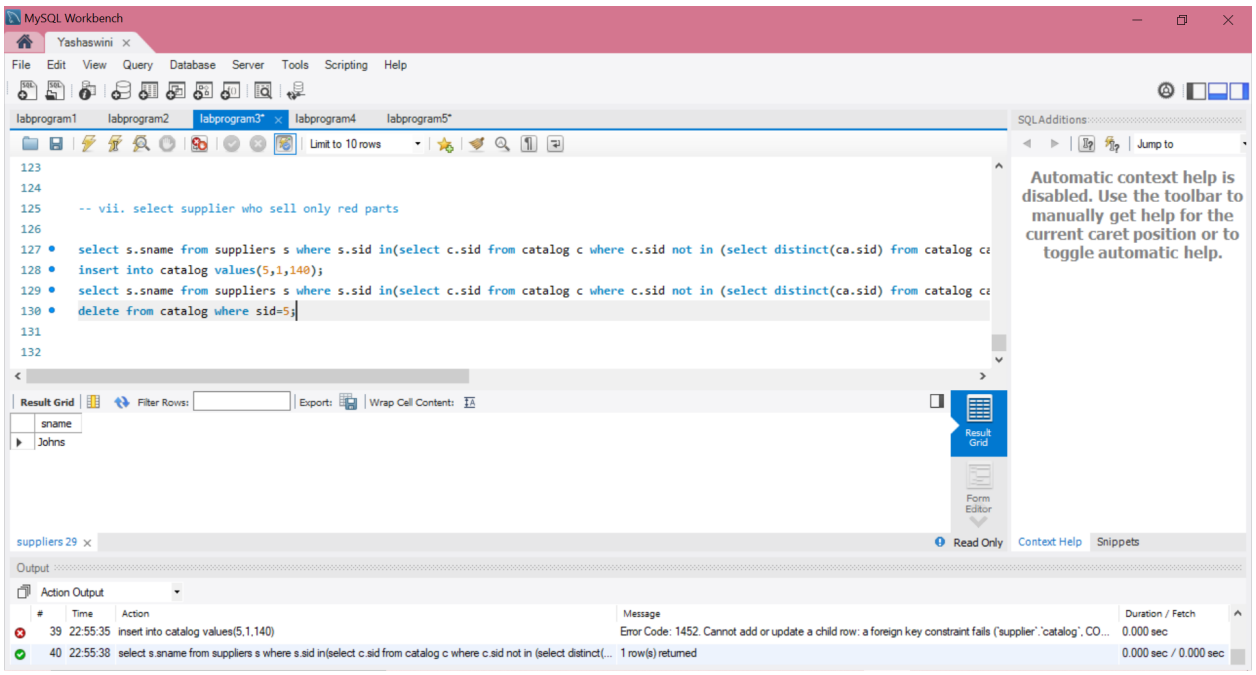
vi. For each part, find the sname of the supplier who charges the most for that part.

select s.sname from suppliers s where s.sid in
(select c.sid from catalog c where c.cost=(select max(cost) from catalog ca where ca.pid=c.pid));



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

```
select s.sname from suppliers s where
s.sid in(select c.sid from catalog c where c.sid not in (select distinct(ca.sid) from catalog ca,parts p
where ca.pid=p.pid and p.color!='red'));
insert into catalog values(5,1,140);
select s.sname from suppliers s where s.sid in(select c.sid from catalog c where c.sid not in (select
distinct(ca.sid) from catalog ca,parts p where ca.pid=p.pid and p.color!='red'));
delete from catalog where sid=5;
```



Name: Yashaswini Shah

USN:1BM19CS216

PROGRAM 4:STUDENT FACULTY 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.

```
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));

desc student;

create table faculty(
fid INT,
fname VARCHAR(20),
deptid INT,
PRIMARY KEY(fid));

desc faculty;
```

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

desc class;

CREATE TABLE enrolled(
snum INT,
cname VARCHAR(20),
PRIMARY KEY(snum,cname),
FOREIGN KEY(snum) REFERENCES student(snum),
FOREIGN KEY(cname) REFERENCES class(cname));

desc enrolled;

INSERT INTO STUDENT VALUES(1, 'John', '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);
commit;

select * from student;
```

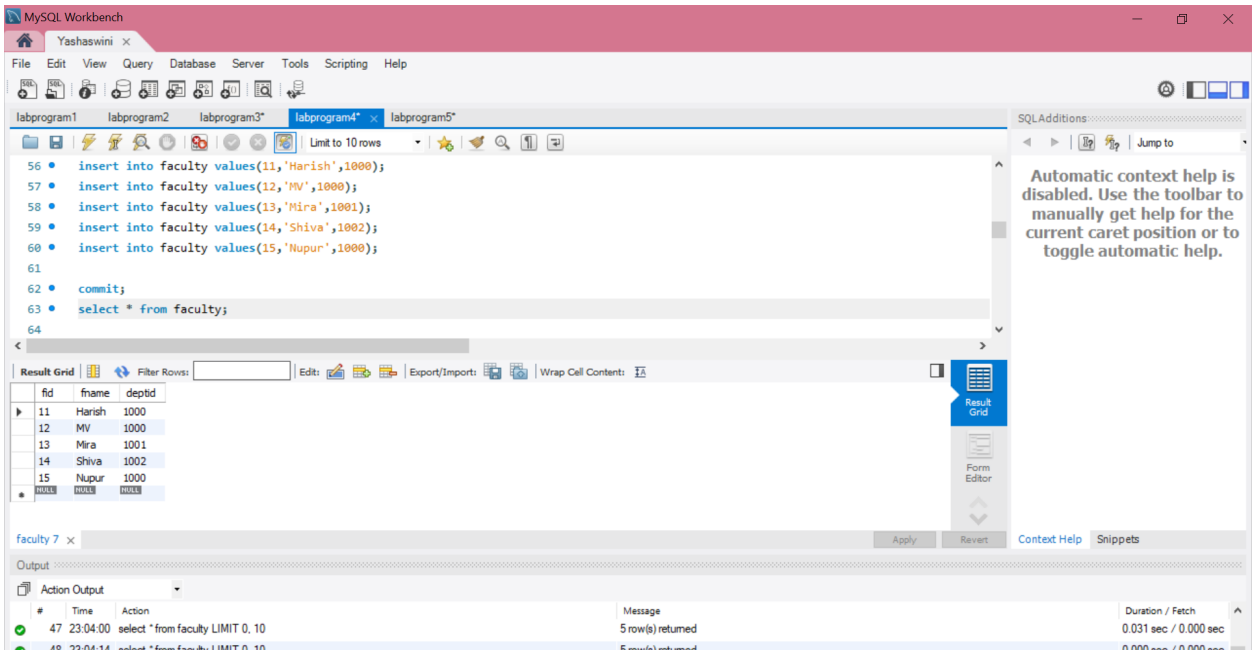
```
INSERT INTO FACULTY VALUES(11, 'Harish', 1000);
INSERT INTO FACULTY VALUES(12, 'MV', 1000);
INSERT INTO FACULTY VALUES(13 , 'Mira', 1001);
INSERT INTO FACULTY VALUES(14, 'Shiva', 1002);
INSERT INTO FACULTY VALUES(15, 'Nupur', 1000);
commit;

select * from faculty;

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);
commit;

select * from class;

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');

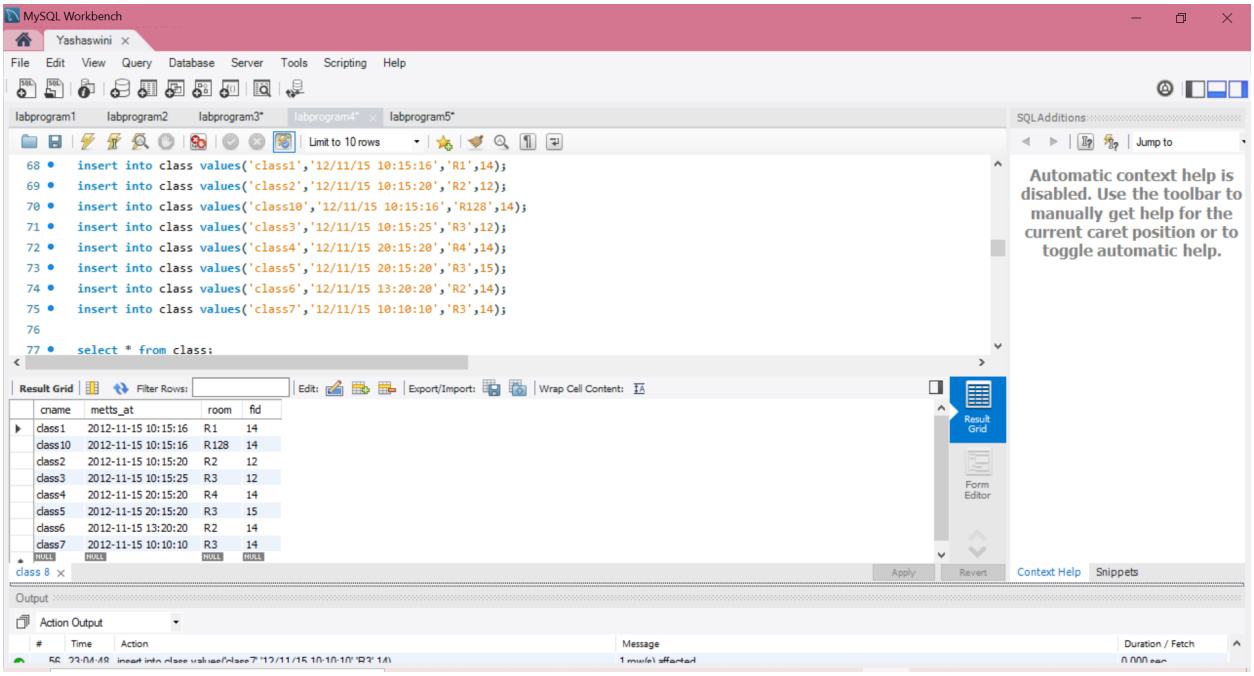
commit;

select * from enrolled;

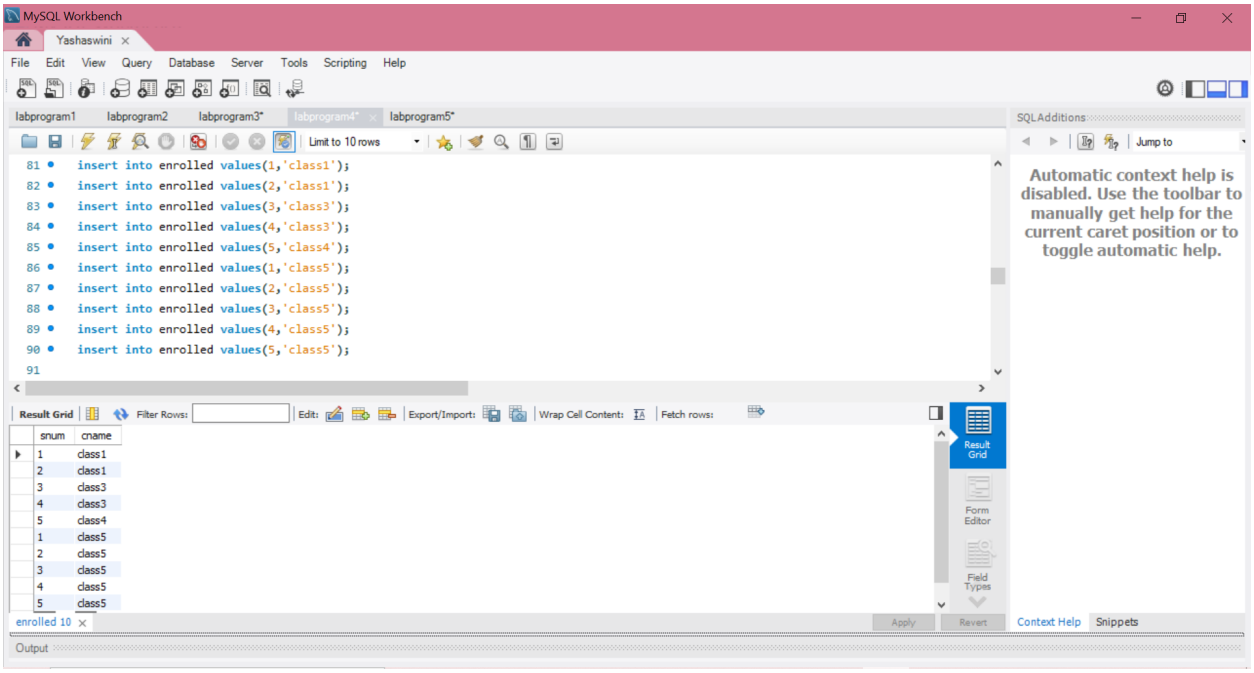
STUDENT TABLE



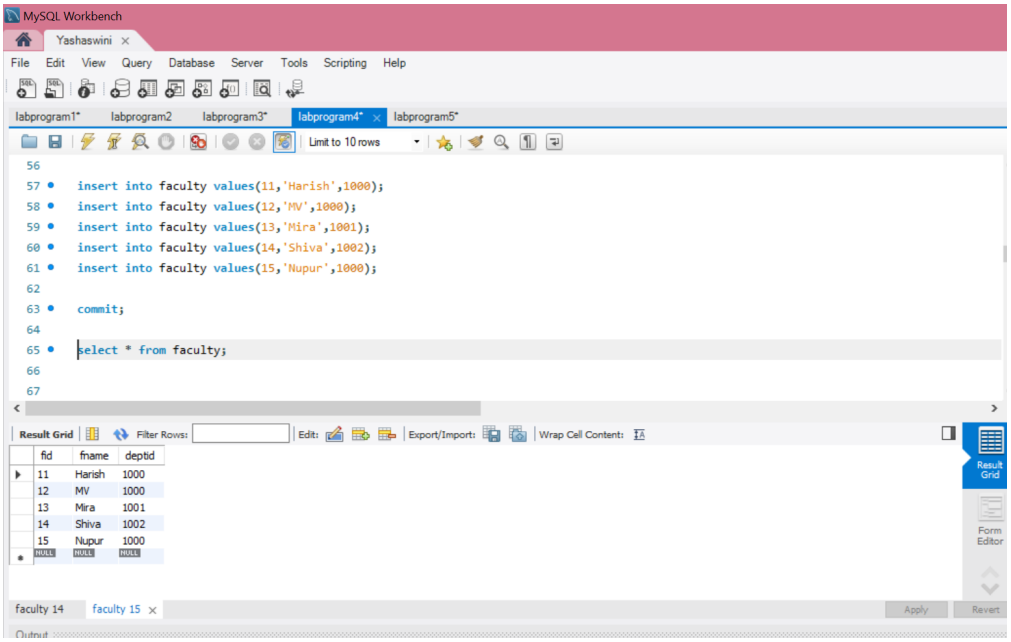
CLASS TABLE



ENROLLED TABLE



FACULTY TABLE



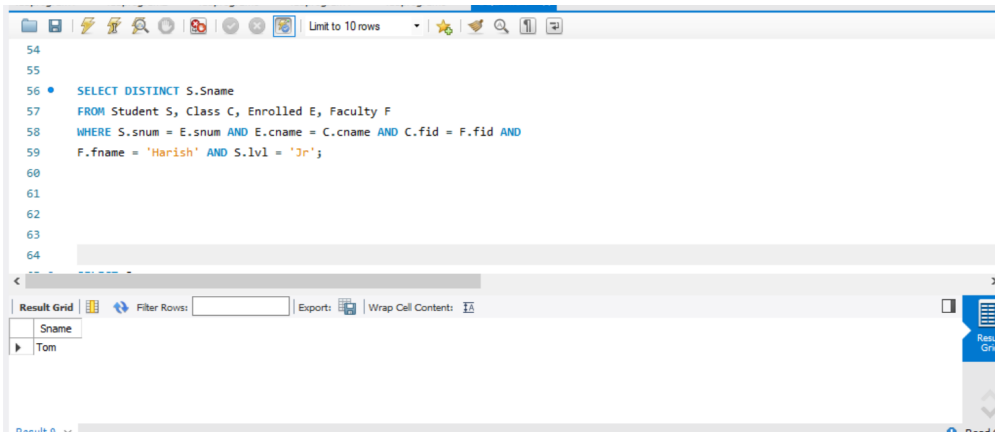
i. Find the names of all Juniors (level = JR) who are enrolled in a class taught by

SELECT DISTINCT S.Sname

FROM Student S, Class C, Enrolled E, Faculty F

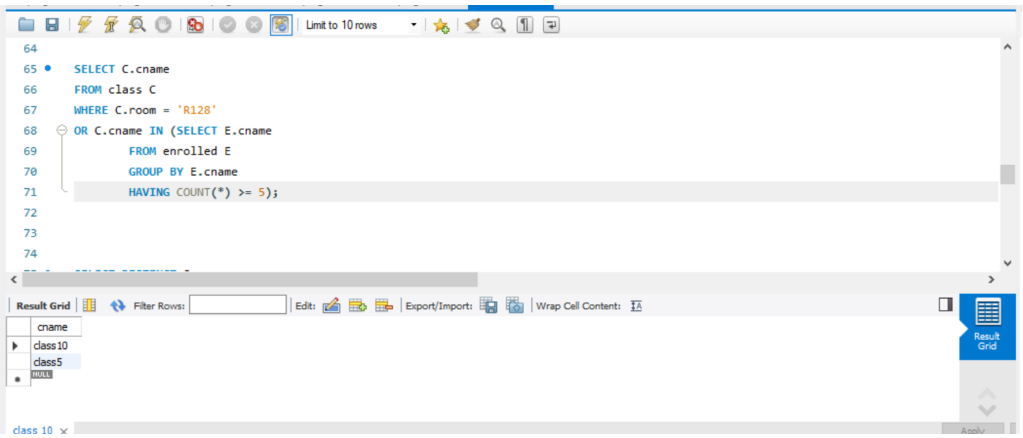
WHERE S.snum = E.snum AND E.cname = C.cname AND C.fid = F.fid AND

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(*) >= 5);
```



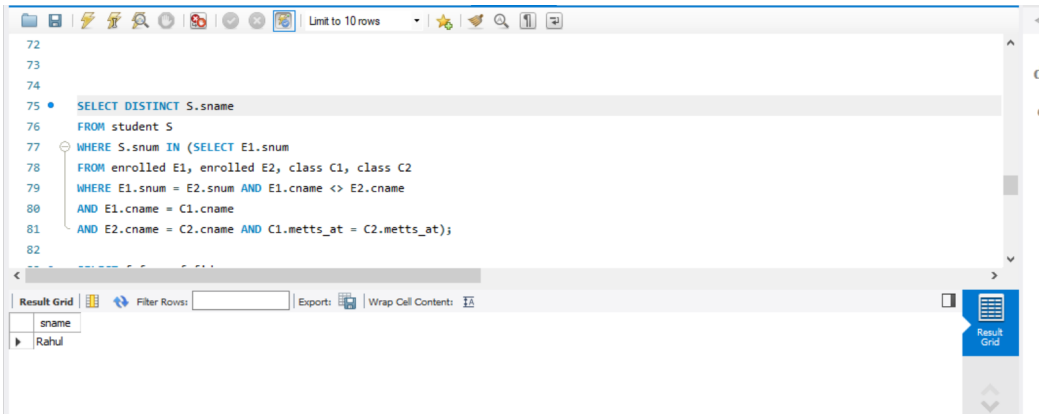
iii. Find the names of all students who are enrolled in two classes that meet at the same time.

```
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);



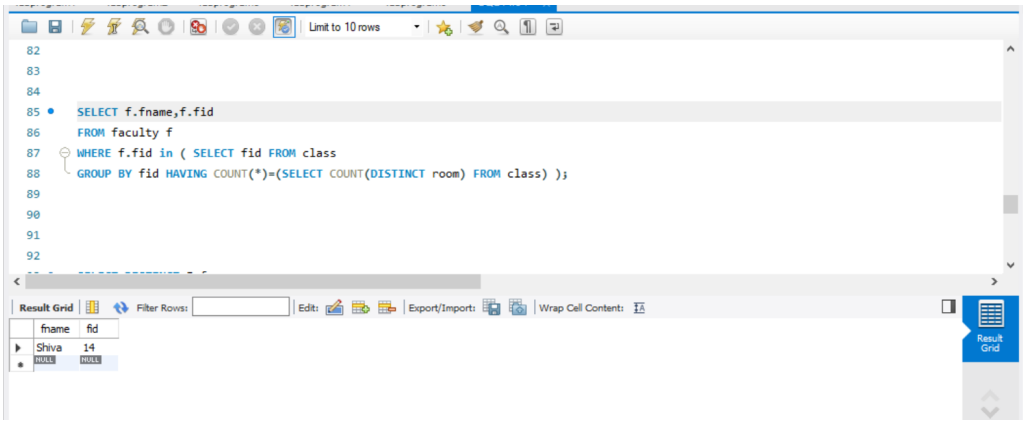
iv. Find the names of faculty members who teach in every room in which some class is taught.

SELECT f.fname,f.fid

FROM faculty f

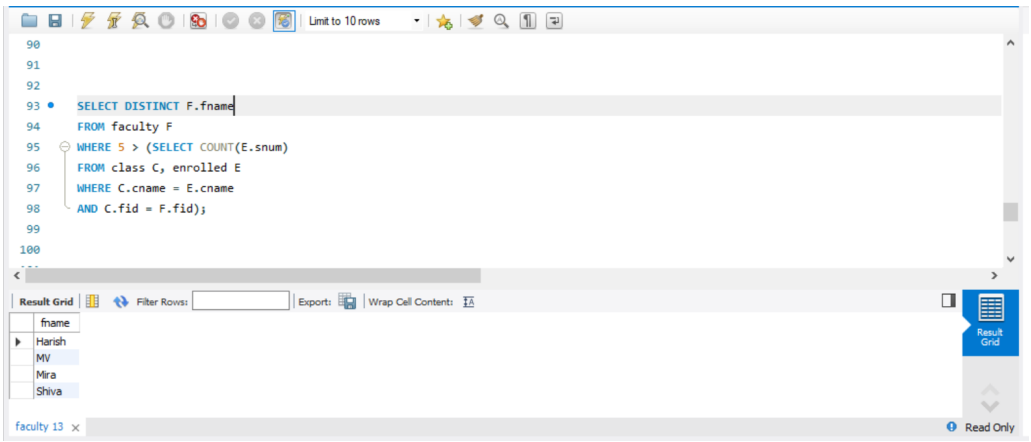
WHERE f.fid in (SELECT fid FROM class

GROUP BY fid HAVING COUNT(*)=(SELECT COUNT(DISTINCT room) FROM class));



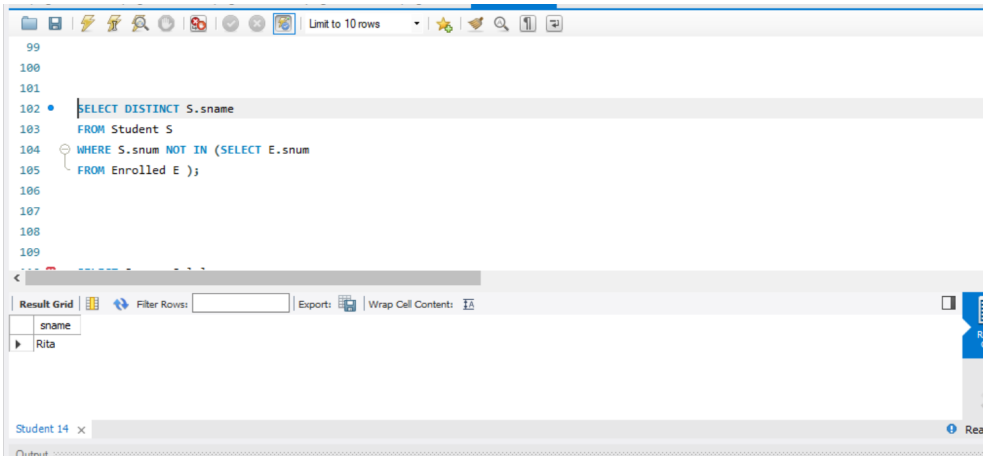
v. Find the names of faculty members for whom the combined enrollment of the courses that they teach is less than five.

```
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);
```



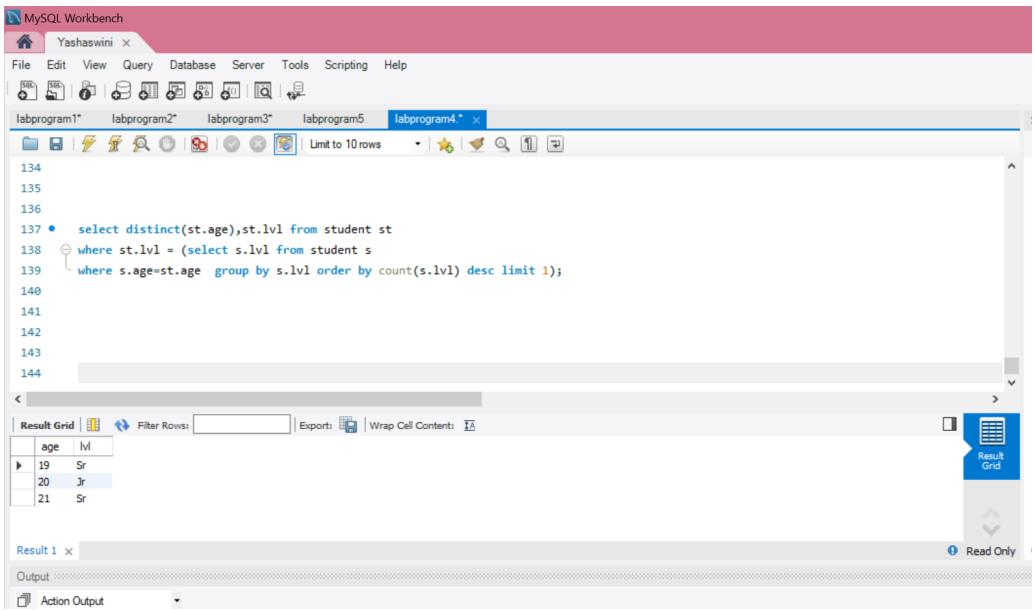
vi. Find the names of students who are not enrolled in any class.

```
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 distinct(st.age),st.lvl from student st
where st.lvl = (select s.lvl from student s
where s.age=st.age  group by s.lvl order by count(s.lvl) desc limit 1);
```



PROGRAM 5: 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)

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

```
create database AirlineFlight;
 use AirlineFlight;

 CREATE TABLE FLIGHTS
  (FLNO INTEGER PRIMARY KEY,
  FFROM VARCHAR(15) ,
  TTO VARCHAR(15) ,
  DISTANCE INTEGER,
  DEPARTS TIMESTAMP,
  ARRIVES TIMESTAMP,
  PRICE INTEGER );

DESC FLIGHTS;

CREATE TABLE AIRCRAFT
  (AID INTEGER PRIMARY KEY,
  ANAME VARCHAR(10),
  CRUISINGRANGE INTEGER);

 DESC AIRCRAFT;
```

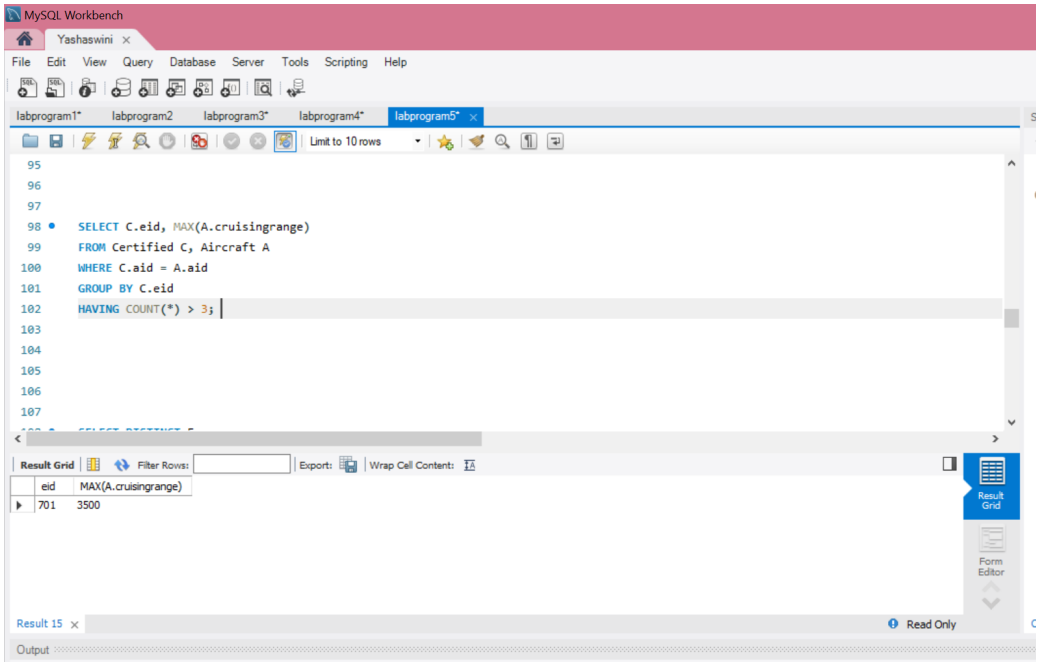
```
 CREATE TABLE EMPLOYEES
  (EID INTEGER PRIMARY KEY,
  ENAME VARCHAR(15),
  SALARY INTEGER );

DESC EMPLOYEES;

CREATE TABLE CERTIFIED
  (EID INTEGER NOT NULL,
  AID INTEGER NOT NULL,
  PRIMARY KEY (EID, AID),
  FOREIGN KEY (EID) REFERENCES EMPLOYEES (EID),
  FOREIGN KEY (AID) REFERENCES AIRCRAFT (AID));

 DESC CERTIFIED;
COMMIT;

insert into aircraft values(101,'747',3000);
insert into aircraft values(102,'Boeing',900);
insert into aircraft values(103,'647',800);
insert into aircraft values(104,'Dreamliner',10000);
insert into aircraft values(105,'Boeing',3500);
insert into aircraft values(106,'707',1500);
insert into aircraft values(107,'Dream', 120000);

select * from aircraft;

insert into employees values(701,'A',50000);
insert into employees values(702,'B',100000);
insert into employees values(703,'C',150000);
insert into employees values(704,'D',90000);
insert into employees values(705,'E',40000);
insert into employees values(706,'F',60000);
insert into employees values(707,'G',90000);
```

```
select * from employees;

insert into certified values(701,101);
insert into certified values(701,102);
insert into certified values(701,106);
insert into certified values(701,105);
insert into certified values(702,104);
insert into certified values(703,104);
insert into certified values(704,104);
insert into certified values(702,107);
insert into certified values(703,107);
insert into certified values(704,107);
insert into certified values(702,101);
insert into certified values(703,105);
insert into certified values(704,105);
insert into certified values(705,103);

select * from certified;

insert into flights values(101,'Bangalore','Delhi',2500,TIMESTAMP '2005-05-13 07:15:31',TIMESTAMP '2005-05-13 17:15:31',5000);
insert into flights values(102,'Bangalore','Lucknow',3000,TIMESTAMP '2005-05-13 07:15:31',TIMESTAMP '2005-05-13 11:15:31',6000);
insert into flights values(103,'Lucknow','Delhi',500,TIMESTAMP '2005-05-13 12:15:31',TIMESTAMP ' 2005-05-13 17:15:31',3000);
insert into flights values(107,'Bangalore','Frankfurt',8000,TIMESTAMP '2005-05-13 07:15:31',TIMESTAMP '2005-05-13 22:15:31',60000);
insert into flights values(104,'Bangalore','Frankfurt',8500,TIMESTAMP '2005-05-13 07:15:31',TIMESTAMP '2005-05-13 23:15:31',75000);
insert into flights values(105,'Kolkata','Delhi',3400,TIMESTAMP '2005-05-13 07:15:31',TIMESTAMP '2005-05-13 09:15:31',7000);

select * from Flights;
```

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.

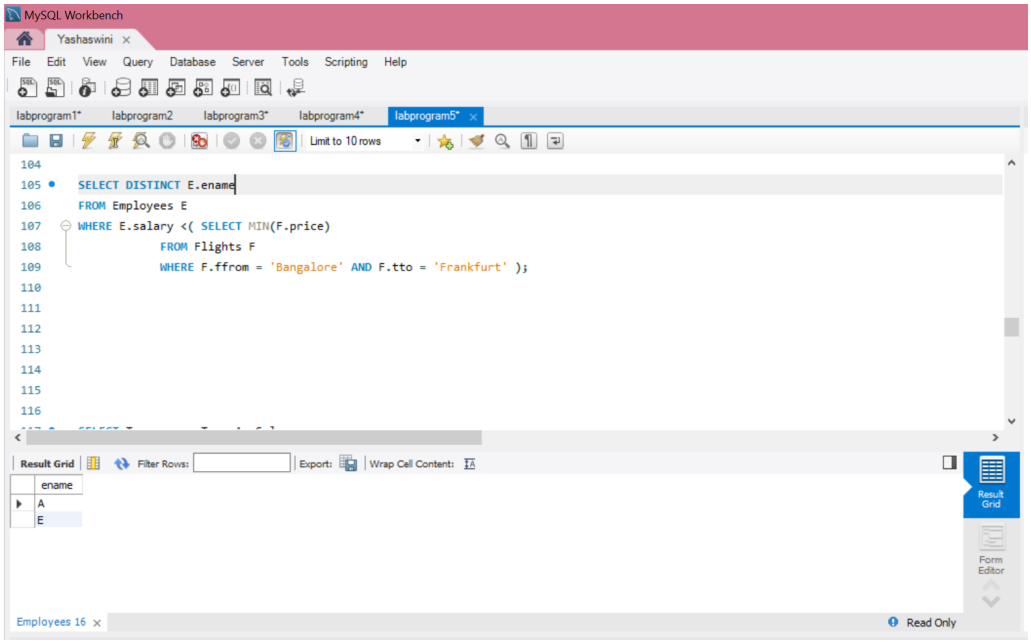
i. **Find the names of aircraFor each pilot who is certified for more than three aircrafts, find the eid and the maximum cruisingrange of the aircraft for which she or he is certified.**

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



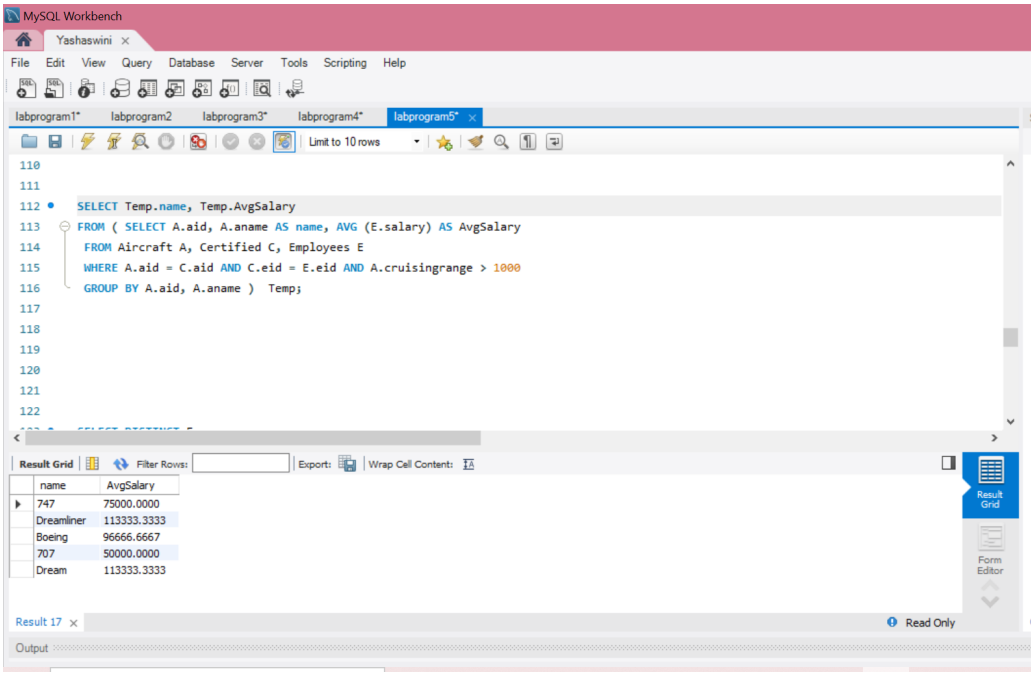
ii. Find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru to Frankfurt.

```
SELECT DISTINCT E.ename
FROM Employees E
WHERE E.salary <( SELECT MIN(F.price)
FROM Flights F
WHERE F.ffrom = 'Bangalore' AND F.tto = 'Frankfurt' );
```



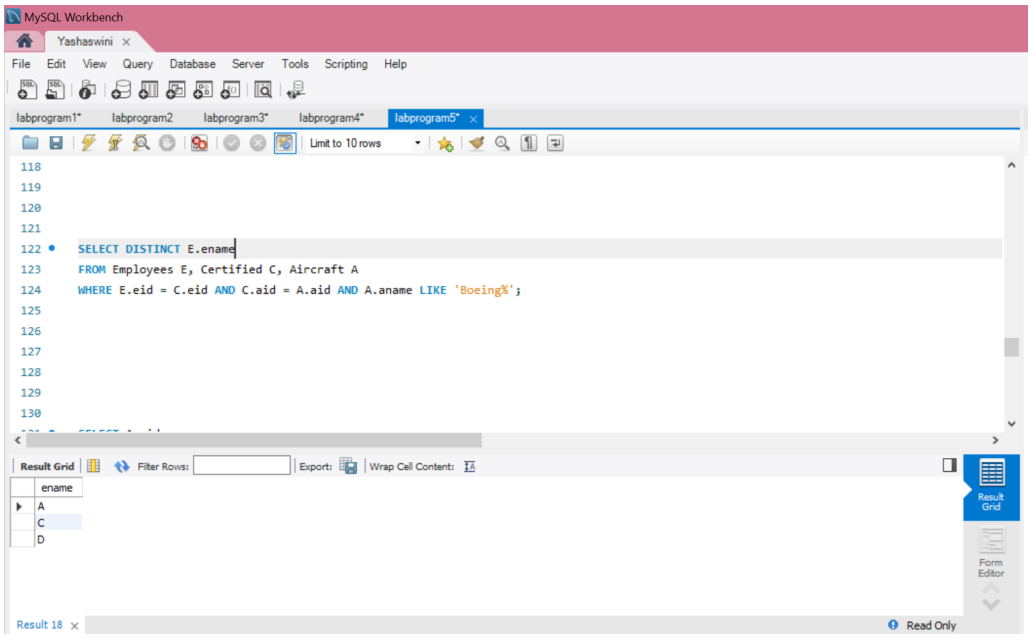
iii. For all aircraft with cruisingrange over 1000 Kms, find the name of the aircraft and the average salary of all pilots certified for this aircraft.

```
SELECT Temp.name, Temp.AvgSalary
FROM ( SELECT A.aid, A.aname AS name, AVG (E.salary)  AS AvgSalary
 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 )  Temp;
```



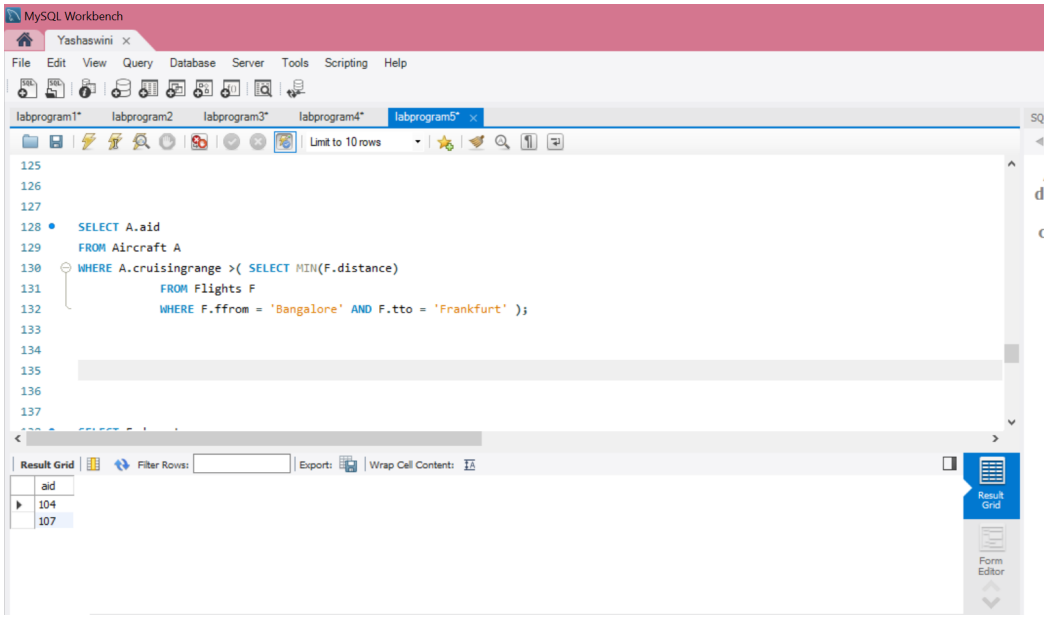
iv. Find the names of pilots certified for some Boeing aircraft.

```
SELECT DISTINCT E.ename
FROM Employees E, Certified C, Aircraft A
WHERE E.eid = C.eid AND C.aid = A.aid AND A.aname  LIKE 'Boeing%';
```



v. **Find the aids of all aircraft that can be used on routes from Bengaluru to Frankfurt.**

```
SELECT A.aid
FROM Aircraft A
WHERE A.cruisingrange >( SELECT MIN(F.distance)
                         FROM Flights F
                         WHERE F.ffrom = 'Bangalore' AND F.tto = 'Frankfurt' );
```



vi. **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.ffrom = 'Bangalore' AND F0.tto
= 'Delhi'  AND extract(hour from F0.arrives) < 18 )

 UNION
```

(SELECT F0.flno FROM Flights F0, Flights F1 WHERE F0.ffrom = 'Bangalore' AND F0.tto <> 'Delhi'

AND F0.tto = F1.ffrom AND F1.tto = 'Delhi' AND F1.departs > F0.arrives AND extract(hour from F1.arrives) < 18)

UNION

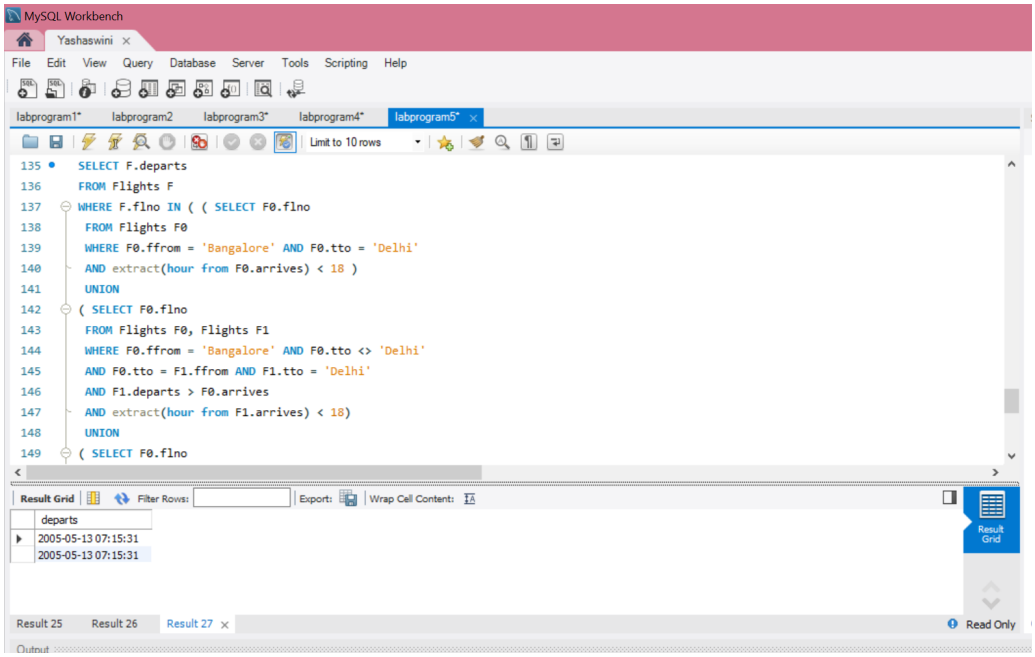
(SELECT F0.flno FROM Flights F0, Flights F1, Flights F2 WHERE F0.ffrom = 'Bangalore'

AND F0.tto = F1.ffrom AND F1.tto = F2.ffrom

AND F2.tto = 'Delhi' AND F0.tto <> 'Delhi'

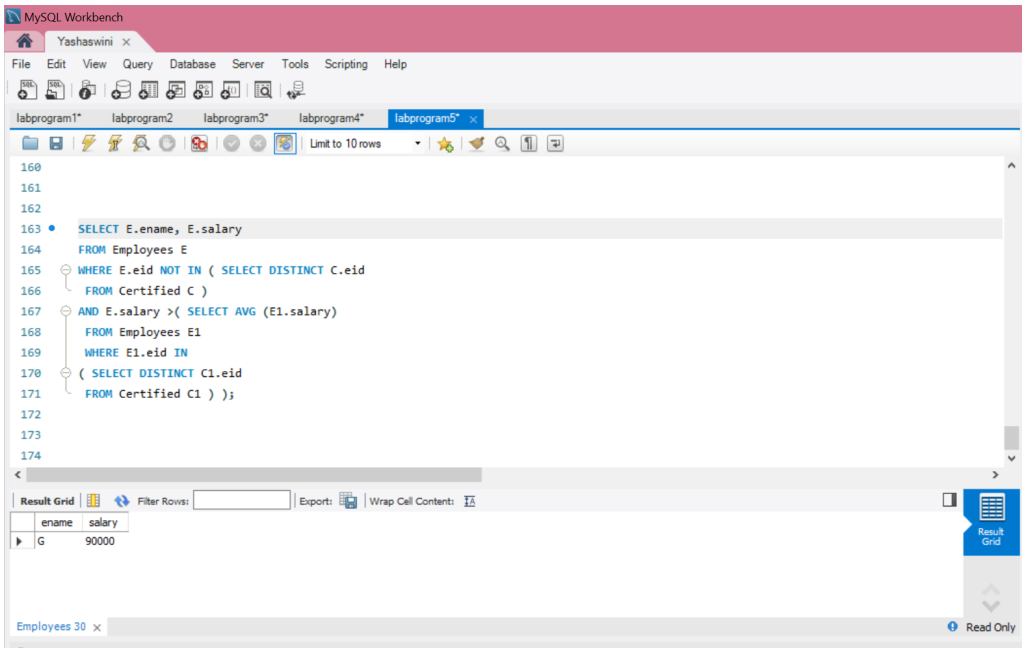
AND F1.tto <> 'Delhi' AND F1.departs > F0.arrives

AND F2.departs > F1.arrives AND extract(hour from F2.arrives) < 18));



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

```
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 ) );
```



PROGRAM 6: ORDER DATABASE

SALESMAN (*Salesman_id, Name, City, Commission*)
CUSTOMER (*Customer_id, Cust_Name, City, Grade, Salesman_id*)
ORDERS (*Ord_No, Purchase_Amt, Ord_Date, Customer_id, Salesman_id*)
Write SQL queries to
1. Count the customers with grades above Bangalore's average.

2. Find the name and numbers of all salesmen who had more than one customer.

3. List all salesmen and indicate those who have and don't have customers in their cities (Use UNION operation.)

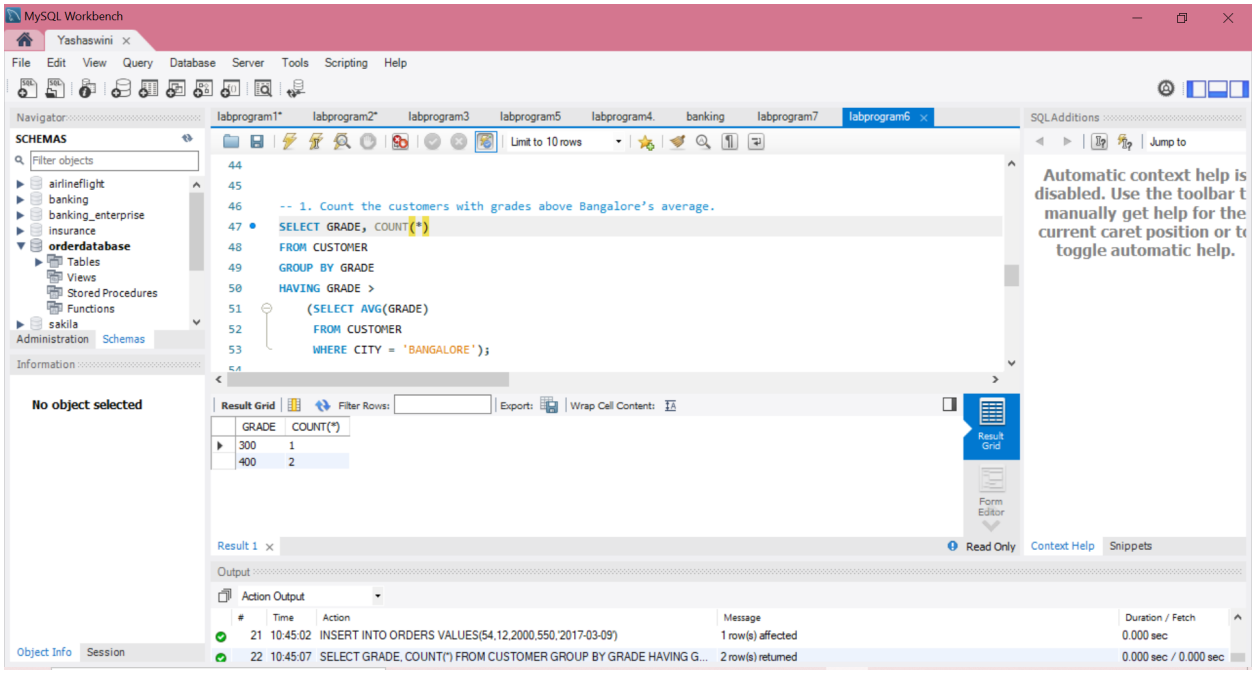
4. Create a view that finds the salesman who has the customer with the highest order of a day.

```
CREATE DATABASE ORDERDATABASE;
USE ORDERDATABASE;
CREATE TABLE SALESMAN(
SALESMANID INT,
SNAME VARCHAR(30),
CITY VARCHAR(30),
COMMISSION VARCHAR(30),
PRIMARY KEY (SALESMANID));
CREATE TABLE CUSTOMER(
CUSTOMERID INT,
SALESMANID INT,
CNAME VARCHAR(30),
CITY VARCHAR(20),
GRADE INT,
PRIMARY KEY (CUSTOMERID),
FOREIGN KEY (SALESMANID) REFERENCES SALESMAN(SALESMANID));
CREATE TABLE ORDERS(
ORDERNO INT,
CUSTOMERID INT,
SALESMANID INT,
PURCHASEAMT INT,
ORDERDATE DATE,
PRIMARY KEY (ORDERNO),
FOREIGN KEY (SALESMANID) REFERENCES SALESMAN(SALESMANID),
FOREIGN KEY (CUSTOMERID) REFERENCES CUSTOMER(CUSTOMERID));

INSERT INTO SALESMAN VALUES(1000,'JOHN','BANGALORE','25%');
```

```
INSERT INTO SALESMAN VALUES(2000,'RAVI','BANGALORE','20%');
INSERT INTO SALESMAN VALUES(3000,'KUMAR','MYSORE','15%');
INSERT INTO SALESMAN VALUES(4000,'SMITH','DELHI','30%');
INSERT INTO SALESMAN VALUES(5000,'HARSHA','HYDERABAD','15%');

INSERT INTO CUSTOMER VALUES(10,1000,'PREETHI','BANGALORE',100);
INSERT INTO CUSTOMER VALUES(11,2000,'VIVEK','MANGALORE',300);
INSERT INTO CUSTOMER VALUES(12,2000,'BHASKAR','CHENNAI',400);
INSERT INTO CUSTOMER VALUES(13,3000,'CHETAN','BANGALORE',200);
INSERT INTO CUSTOMER VALUES(14,2000,'MAMTHA','BANGALORE',400);

INSERT INTO ORDERS VALUES(50,10,1000,5000,'2017-05-04');
INSERT INTO ORDERS VALUES(51,10,2000,450,'2017-01-20');
INSERT INTO ORDERS VALUES(52,13,2000,1000,'2017-02-24');
INSERT INTO ORDERS VALUES(53,14,3000,3500,'2017-04-13');
INSERT INTO ORDERS VALUES(54,12,2000,550,'2017-03-09');
```

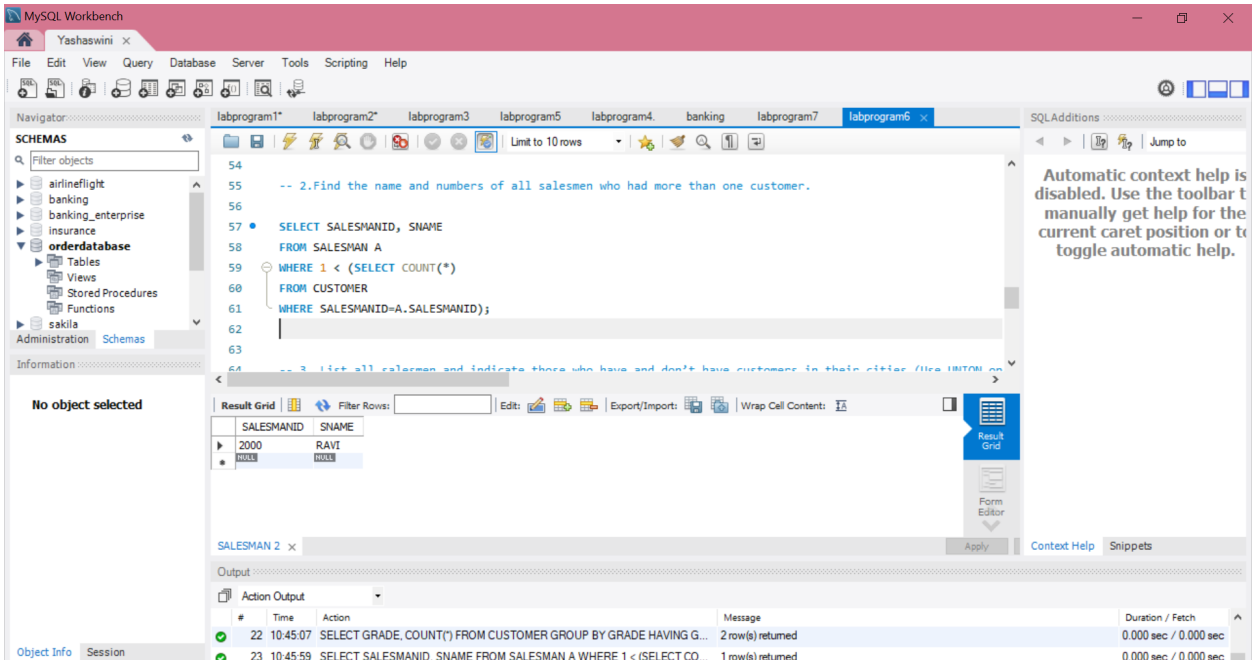
-- 1. Count the customers with grades above Bangalore’s average.

```
SELECT GRADE, COUNT(*)
FROM CUSTOMER
GROUP BY GRADE
HAVING GRADE >
   (SELECT AVG(GRADE)
    FROM CUSTOMER
    WHERE CITY = 'BANGALORE');
```



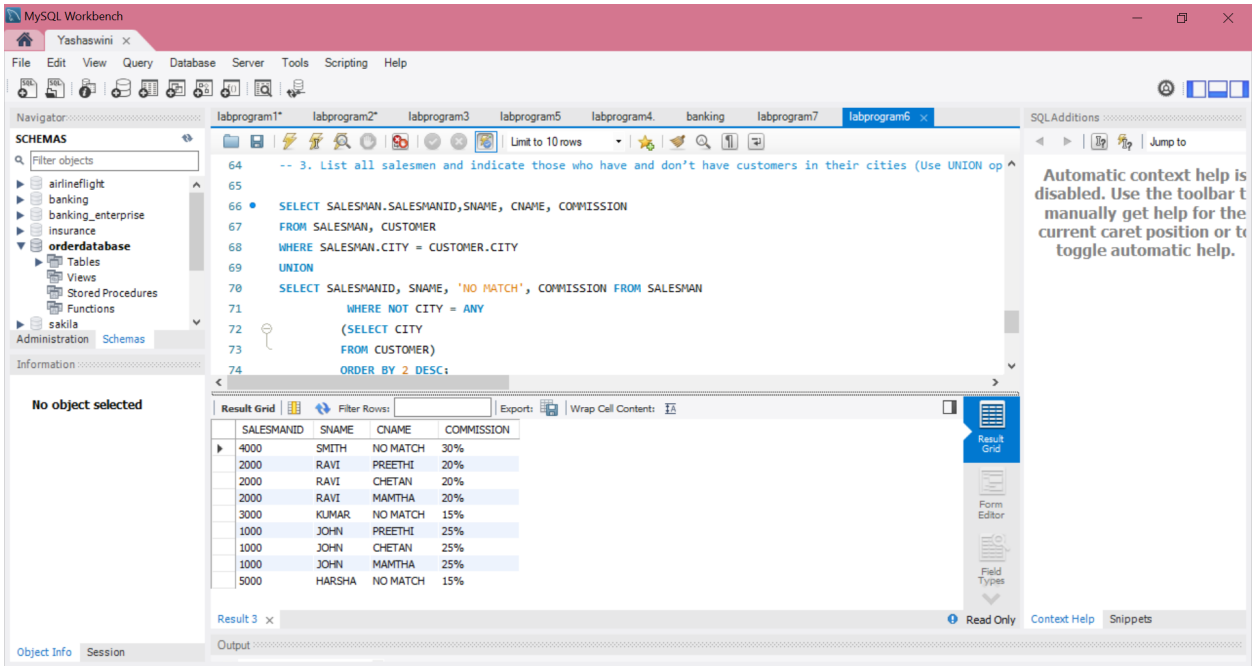
-- 2.Find the name and numbers of all salesmen who had more than one customer.

```
SELECT SALESMANID, SNAME
FROM SALESMAN A
WHERE 1 < (SELECT COUNT(*)
FROM CUSTOMER
WHERE SALESMANID=A.SALESMANID);
```



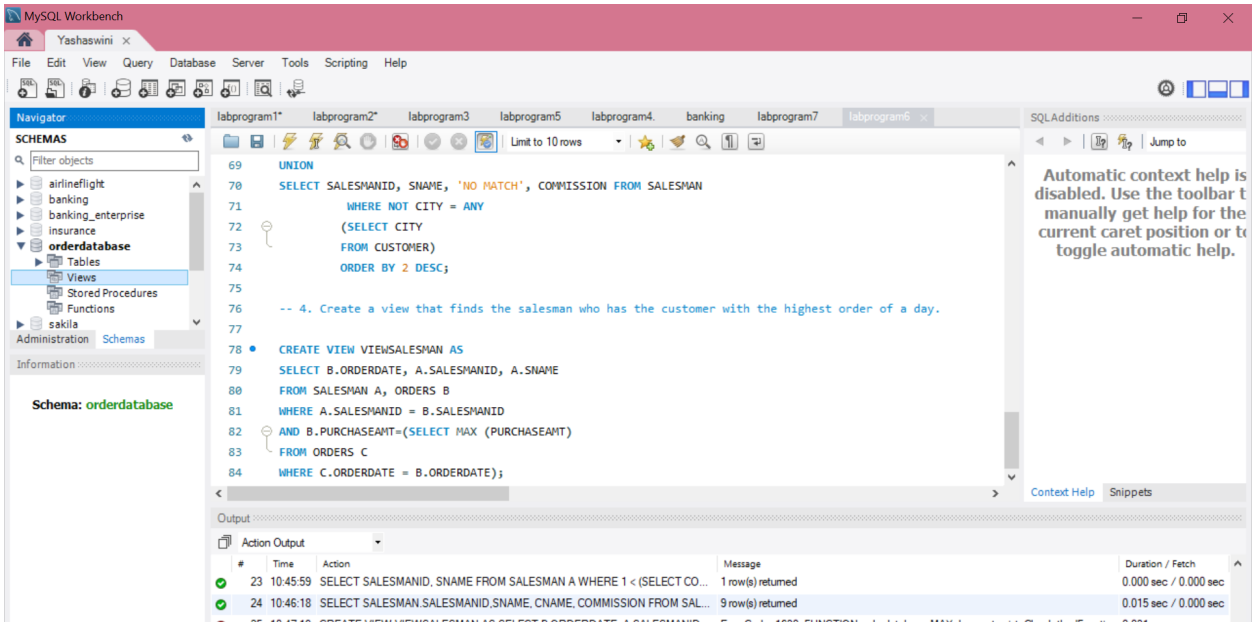
-- 3. List all salesmen and indicate those who have and don't have customers in their cities (Use UNION operation.)

```
SELECT SALESMAN.SALESMANID,SNAME, CNAME, COMMISSION
FROM SALESMAN, CUSTOMER
WHERE SALESMAN.CITY = CUSTOMER.CITY
UNION
SELECT SALESMANID, SNAME, 'NO MATCH', COMMISSION FROM SALESMAN
        WHERE NOT CITY = ANY
        (SELECT CITY
        FROM CUSTOMER)
        ORDER BY 2 DESC;
```



-- 4. Create a view that finds the salesman who has the customer with the highest order of a day.

```
CREATE VIEW VIEWSALESMAN AS
SELECT B.ORDERDATE, A.SALESMANID, A.SNAME
FROM SALESMAN A, ORDERS B
WHERE A.SALESMANID = B.SALESMANID
AND B.PURCHASEAMT=(SELECT MAX (PURCHASEAMT)
FROM ORDERS C
WHERE C.ORDERDATE = B.ORDERDATE);
```



PROGRAM NUMBER 7: BOOK DEALER

AUTHOR(author-id: int, name: String, city: String, country: String)
PUBLISHER(publisher-id: int, name: String, city: String, country: String)
CATALOG (book-id: int, title: String, author-id: int, publisher-id: int, category-id: int, year: int, price: int)
CATEGORY(category-id: int, description: String)
ORDER-DETAILS(order-no: int, book-id: int, quantity: int)

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

```
create database book_dealer;


use book_dealer;
create table author(
author_id int,
name varchar(30),
```

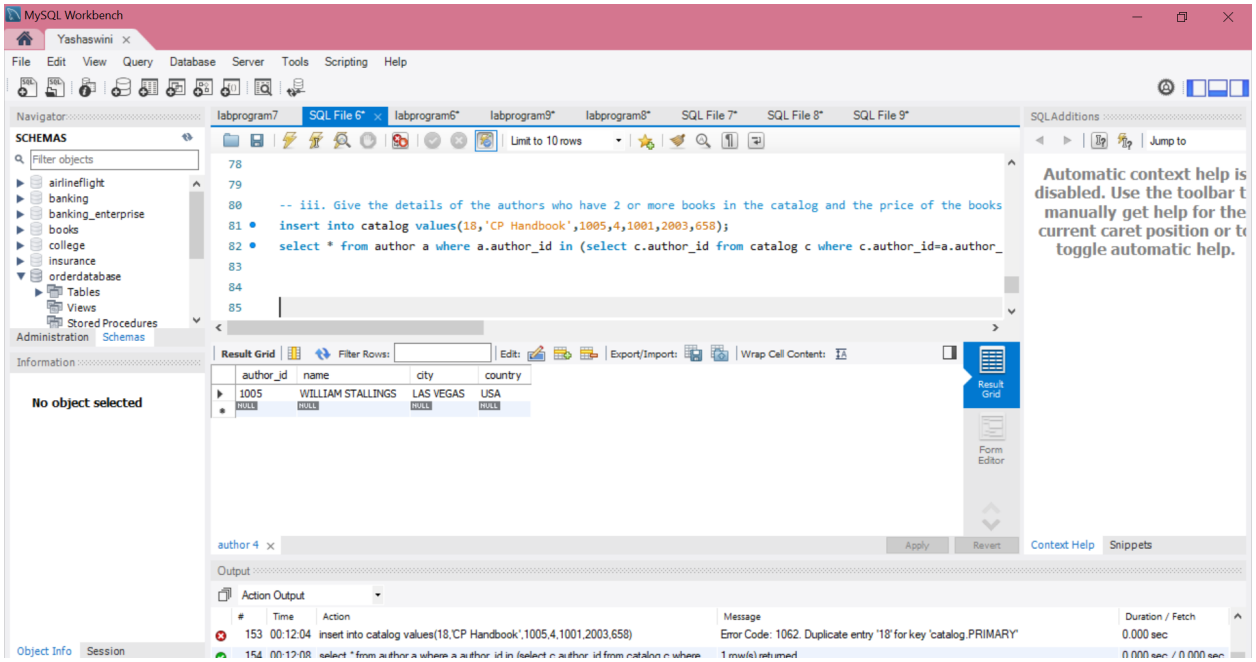
```
city varchar(20),
country varchar(20),
constraint auth_id primary key (author_id)
);
create table publisher(
publisher_id int,
name varchar(30),
city varchar (30),
country varchar(30),
constraint pub_id primary key (publisher_id)
);

create table category(
category_id int,
description varchar(30),
constraint primary key (category_id)
);
create table catalog(
book_id int ,
title varchar(30),
author_id int,
publisher_id int ,
category_id int,
year int,
price int,
constraint cat_id primary key(book_id),
constraint cat_auth foreign key (author_id) references author(author_id) on delete cascade on
update cascade,
constraint cat_pub foreign key (publisher_id) references publisher(publisher_id) on delete
cascade on update cascade ,
constraint cat_log foreign key (category_id) references category(category_id) on delete cascade
on update cascade
 );
create table order_details (
order_no int,
book_id int,
quantity int,
constraint ord_no primary key (order_no),
constraint  ord_id foreign key (book_id) references catalog(book_id) on delete cascade on
update cascade
);
```

-- ii. Inserting tuples

```
insert into author values(1001,'TERAS CHAN','CA','USA');
insert into author values(1002,'STEVENS','ZOMBI','UGANDA');
insert into author values(1003,'M MANO','CAIR','CANADA');
insert into author values(1004,'KARTHIK B.P.','NEW YORK','USA');
insert into author values(1005,'WILLIAM STALLINGS','LAS VEGAS','USA');

insert into publisher values(1,'PEARSON','NEW YORK','USA');
insert into publisher values(2,'EEE','NEW SOUTH VALES','USA');
insert into publisher values(3,'PHI','DELHI','INDIA');
insert into publisher values(4,'WILLEY','BERLIN','GERMANY');
insert into publisher values(5,'MGH','NEW YORK','USA');

insert into category values(1001,'COMPUTER SCIENCE');
insert into category values(1002,'ALGORITHM DESIGN');
insert into category values(1003,'ELECTRONICS');
insert into category values(1004,'PROGRAMMING');
insert into category values(1005,'OPERATING SYSTEMS');

insert into catalog values(11,'Unix System Prg',1001,1,1001,2000,251);
insert into catalog values(12,'Digital Signals',1002,2,1003,2001,425);
insert into catalog values(13,'Logic Design',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 Handbook',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(6,17,10);
```

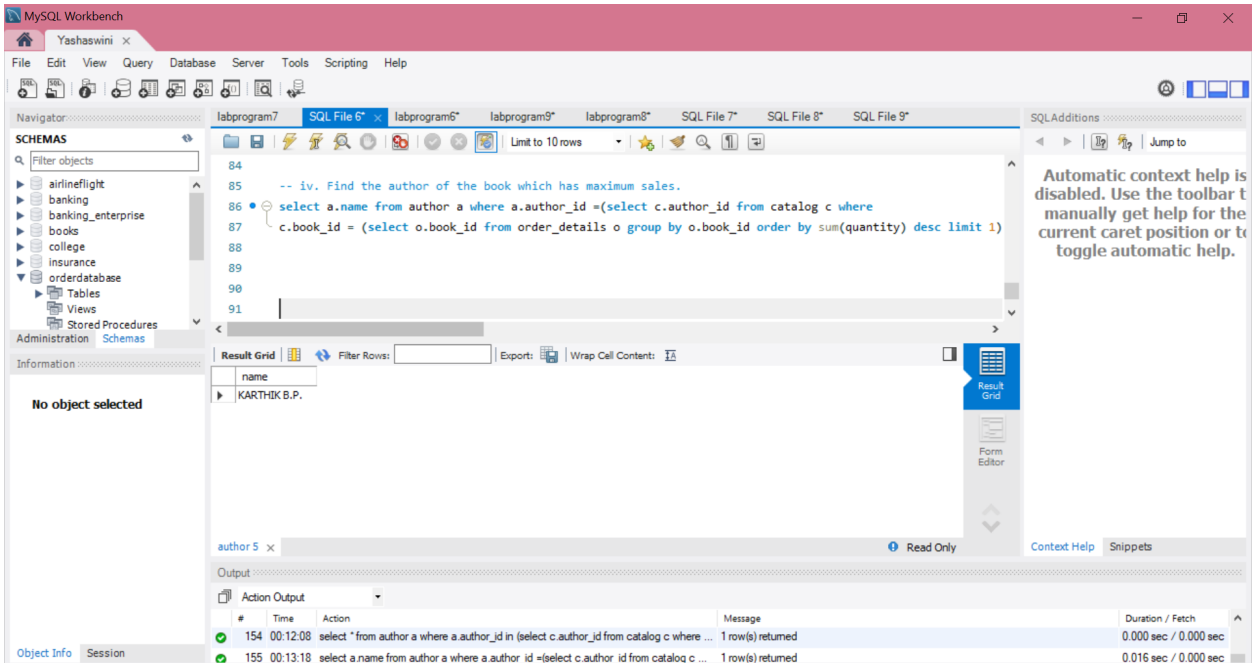
-- iii. Give the details of the authors who have 2 or more books in the catalog and the price of the books in the catalog and the year of publication is aftcataloger 2000.select * from author a where a.author_id in (select c.author_id from catalog c where c.author_id=a.author_id and year>2000 having count(*)>=2);

```
insert into catalog values(18,'CP Handbook',1005,4,1001,2003,658);
select * from author a where a.author_id in (select c.author_id from catalog c where
c.author_id=a.author_id and year>2000 having count(*)>=2);
```



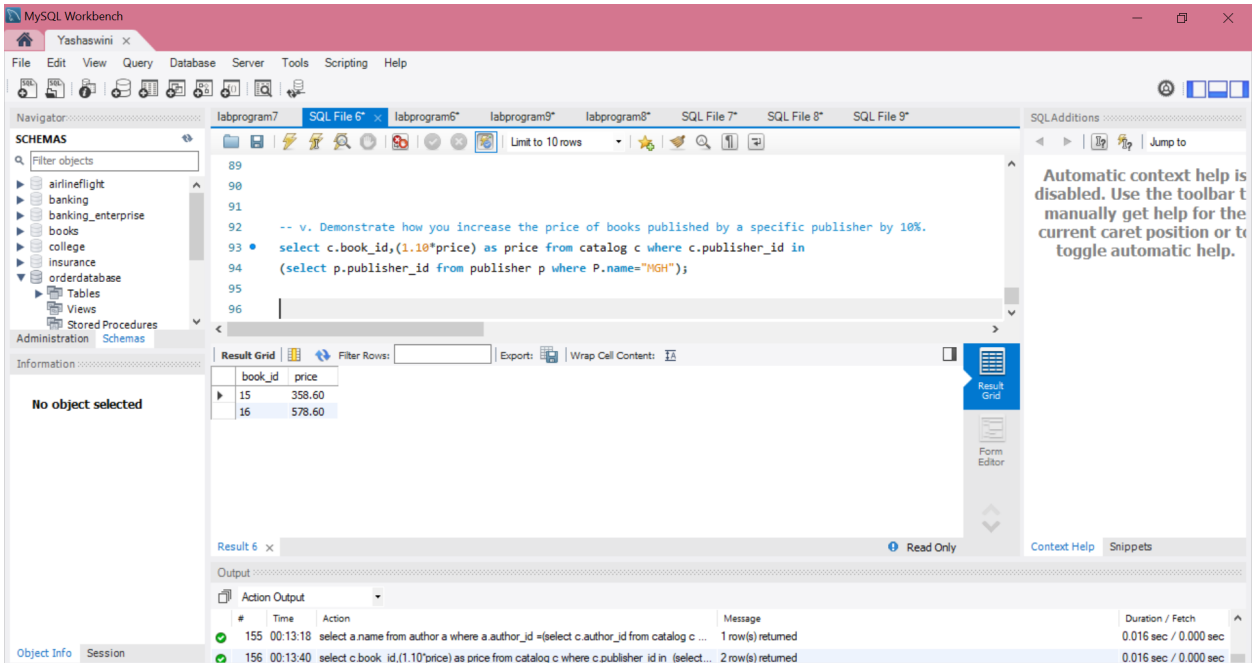
-- iv. Find the author of the book which has maximum sales.

select a.name from author a where a.author_id =(select c.author_id from catalog c where c.book_id = (select o.book_id from order_details o group by o.book_id order by sum(quantity) desc limit 1));



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

select c.book_id,(1.10*price) as price from catalog c where c.publisher_id in (select p.publisher_id from publisher p where P.name="MGH");



PROGRAM 8: STUDENT ENROLLMENT DATABASE

Consider the following database of student enrollment in courses and books adopted for each course.
STUDENT (regno: String, name: String, major: String, bdate: date)
COURSE (course #: int, cname: String, dept: String)
ENROLL (regno: String, cname: String, sem: int, marks: int)
BOOK_ADOPTION (course #: int, sem: int, book-ISBN: int)
TEXT(book-ISBN:int, book-title:String, publisher:String, author:String)
i. Create the above tables by properly specifying the primary keys and the foreign keys.
ii. Enter at least five tuples for each relation.
iii. Demonstrate how you add a new text book to the database and make this book be adopted by some department.
iv. Produce a list of textbooks (include Course #, Book-ISBN, Book-title) in the alphabetical order for courses offered by the 'CS' department that use more than two books.
v. List any department that has all its adopted books published by a specific publisher

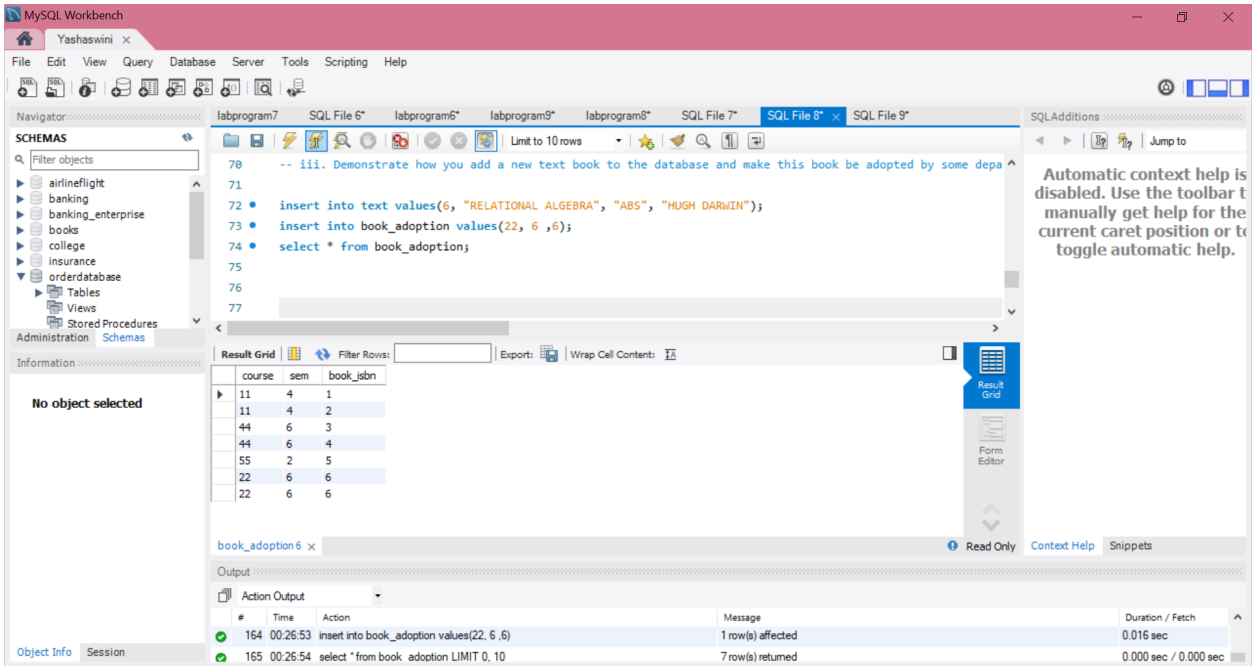
create database student_enrollment;

```
use student_enrollment;

-- i) creating table
create table student(
regno varchar(10) ,
name varchar(30),
major varchar(10),
bdate date,
constraint stu_reg primary key (regno)
);
create table course (
course int ,
cname varchar(30),
dept varchar (30),
constraint cou_cou primary key (course)
);
create table enroll (
regno varchar(10) ,
cname varchar (30),
sem int ,
marks int,
constraint en_reg foreign key (regno) references student(regno)
);
create table text (
book_isbn int,
book_title varchar(30),
publisher varchar(30),
author varchar (30),
constraint book_book primary key (book_isbn)
);
create table book_adoption (
course int,
sem int,
book_isbn int,
constraint book_cou foreign key (course)
references course(course) on delete cascade on update cascade ,
constraint book_book foreign key (book_isbn)
references text(book_isbn) on delete cascade on update cascade
);

-- ii. Inserting tuples
insert into student values("CS01", "PRANAV", "DS", "1986-03-12");
insert into student values("IS02", "PRATEEK", "USP", "1987-12-23");
insert into student values("EC03", "SAURAB", "SNS", "1985-04-17");
```

```
insert into student values("CS03", "ARKA", "DBMS", "1987-01-01");
insert into student values("TC05", "PRANSHU", "EC", "1986-10-06");
insert into course values(11,"DS","CS");
insert into course values(22,"USP","IS");
insert into course values(33,"SNS","EC");
insert into course values(44,"DBMS","CS");
insert into course values(55,"EC","TC");
insert into enroll values("CS01", 11, 4, 85);
insert into enroll values("IS02", 22, 6, 80);
insert into enroll values("EC03", 33, 2, 80);
insert into enroll values("CS03", 44, 6, 75);
insert into enroll values("TC05", 11, 4, 85);
insert into text values(1, "DS AND C", "PRINCETON","PADMA REDDY");
insert into text values(2, "FUNDAMENTALS OF DS", "SPRINGER", "GODSE");
insert into text values(3, "FUNDAMENTALS OF DBMS", "SPRINGER", "NAVATHE");
insert into text values(4, "SQL", "PRINCETON", "FOLEY");
insert into text values(5, "ELECTRONIC CIRCUITS", "TMH", "ELMASRI");
insert into book_adoption values(11, 4 ,1);
insert into book_adoption values(11, 4 ,2);
insert into book_adoption values(44, 6 ,3);
insert into book_adoption values(44, 6 ,4);
insert into book_adoption values(55, 2 ,5);
```

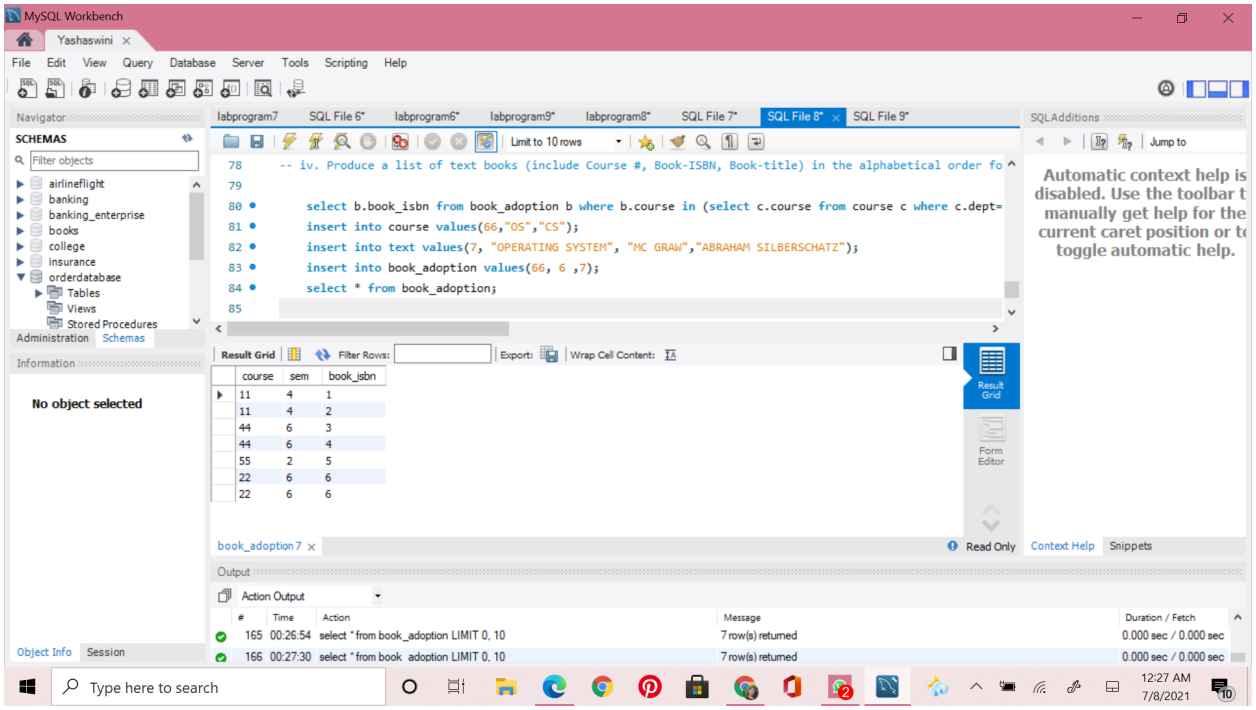
-- iii. Demonstrate how you add a new text book to the database and make this book be adopted by some department.

```
insert into text values(6, "RELATIONAL ALGEBRA", "ABS", "HUGH DARWIN");
insert into book_adoption values(22, 6 ,6);
select * from book_adoption;
```



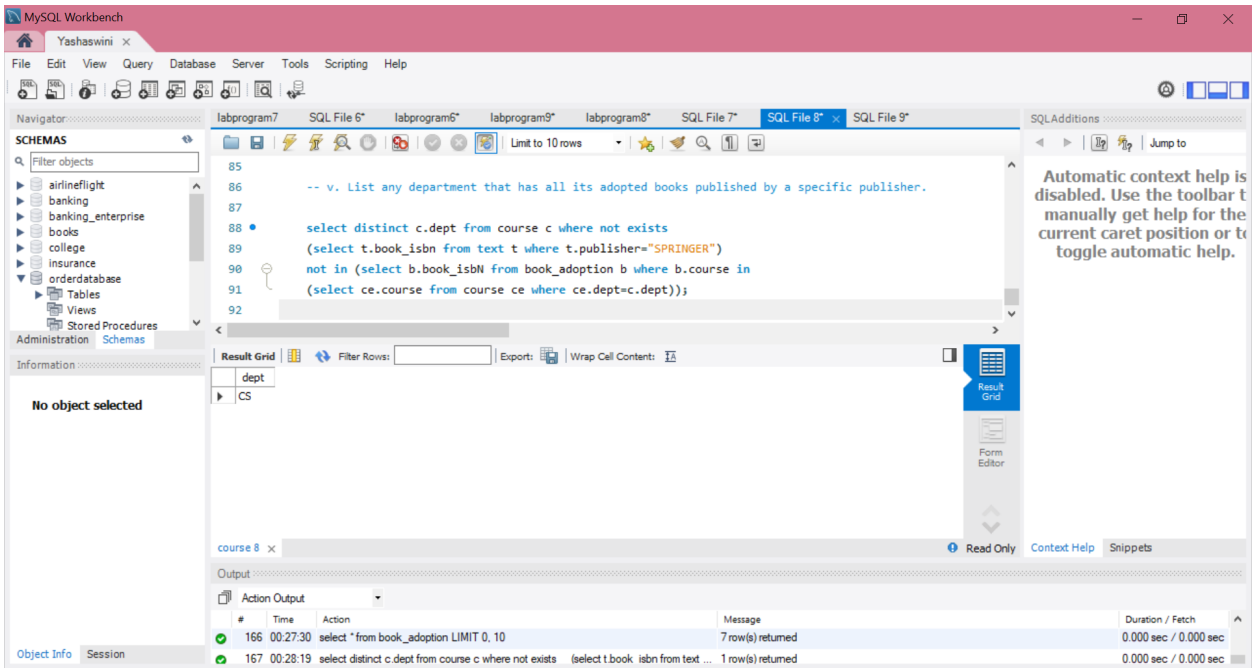
-- iv. Produce a list of text books (include Course #, Book-ISBN, Book-title) in the alphabetical order for courses offered by the 'CS' department that use more than two books.

```
    select b.book_isbn from book_adoption b where b.course in (select c.course from course c
where c.dept="CS") and (select count(*) from book_adoption be where be.course=b.course)>=2
;
    insert into course values(66,"OS","CS");
    insert into text values(7, "OPERATING SYSTEM", "MC GRAW","ABRAHAM
SILBERSCHATZ");
            insert into book_adoption values(66, 6 ,7);
    select * from book_adoption;
```



-- v. List any department that has all its adopted books published by a specific publisher.

select distinct c.dept from course c where not exists (select t.book_isbn from text t where t.publisher="SPRINGER") not in (select b.book_isbN from book_adoption b where b.course in (select ce.course from course ce where ce.dept=c.dept));



USN:1BM19CS216

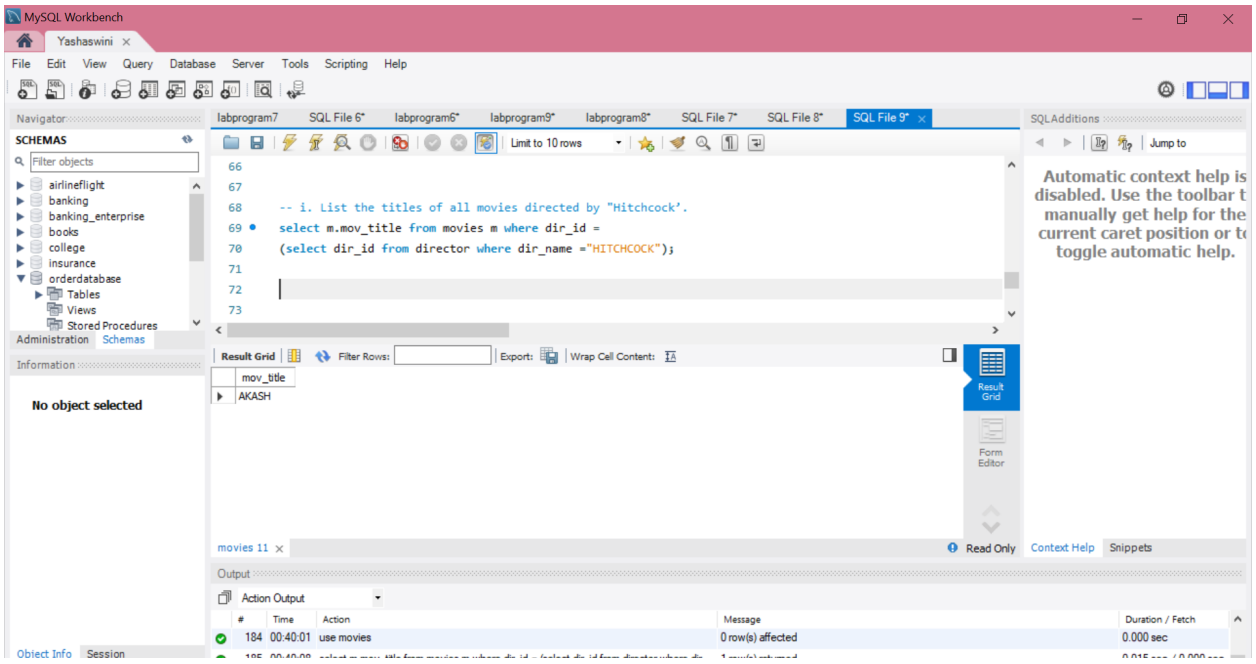
PROGRAM 9 : MOVIE DATABASE

Consider the schema for Movie Database:
ACTOR(Act_id, Act_Name, Act_Gender)
DIRECTOR(Dir_id, Dir_Name, Dir_Phone)
MOVIES(Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id)
MOVIE_CAST(Act_id, Mov_id, Role)
RATING(Mov_id, Rev_Stars)
Write SQL queries to
i. List the titles of all movies directed by '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.

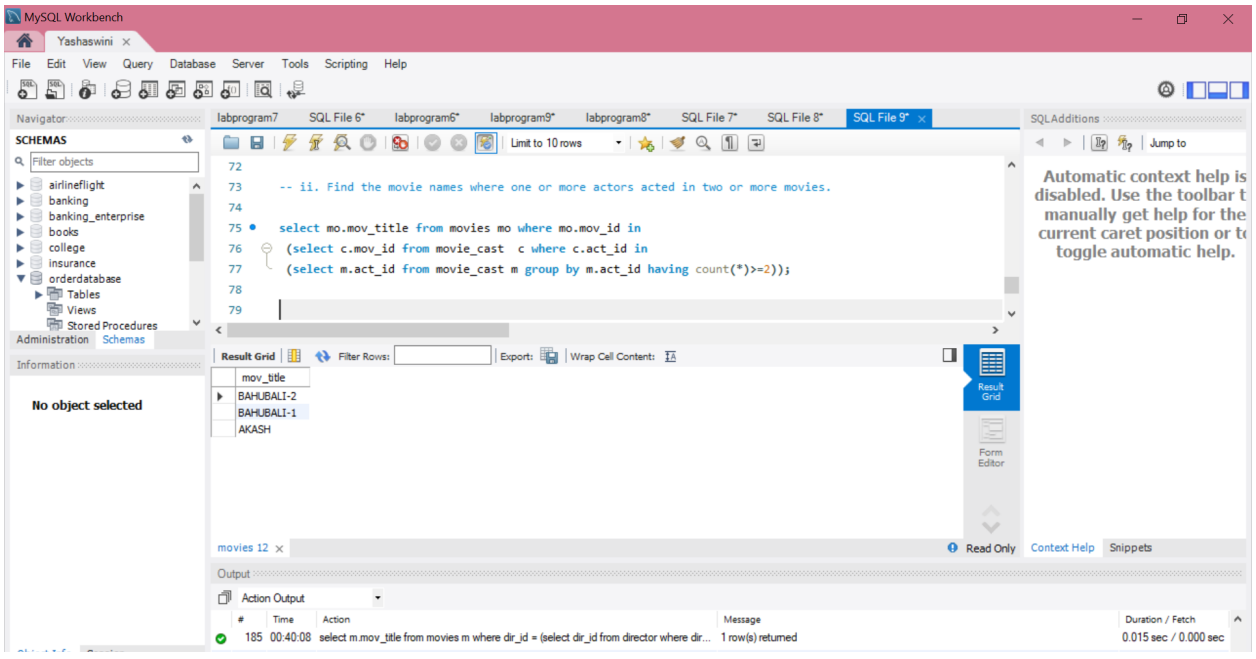
```
create database movies;
use movies;

create table Actor(
act_id int,
act_name varchar(30),
act_gender varchar(1),
constraint act_id primary key (act_id)
);
create table Director(
dir_id int,
dir_name varchar(30),
dir_phone varchar(30),
constraint dir_id primary key (dir_id)
);
create table Movies(
mov_id int,
mov_title varchar(30),
mov_year varchar(4),
mov_lang varchar(12),
dir_id int ,
```

```
constraint mov_id primary key (mov_id),
constraint mov_id foreign key (dir_id) references director(dir_id)
);
create table movie_cast (
act_id int,
mov_id int,
role varchar(10),
constraint mov_pr primary key(act_id ,mov_id),
constraint mov_act foreign key (act_id) references actor(act_id),
constraint mov_mov foreign key (mov_id) references movies(mov_id)
);
create table rating (
mov_id int ,
rev_stars varchar(25),
constraint rat primary key (mov_id),
constraint rat_mov foreign key (mov_id) references movies(mov_id));

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,'jeremy','m');

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);

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

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");
INSERT INTO MOVIE_CAST VALUES (301, 1005, "HEROINE");
INSERT INTO MOVIE_CAST VALUES (303, 1005, "HERO");

INSERT INTO RATING VALUES (1001, 4);
INSERT INTO RATING VALUES (1002, 2);
INSERT INTO RATING VALUES (1003, 5);
INSERT INTO RATING VALUES (1004, 4);
```

-- i. List the titles of all movies directed by "Hitchcock'.
select m.mov_title from movies m where dir_id = (select dir_id from director where dir_name ="HITCHCOCK");



-- ii. Find the movie names where one or more actors acted in two or more movies.
select mo.mov_title from movies mo where mo.mov_id in (select c.mov_id from movie_cast c where c.act_id in (select m.act_id from movie_cast m group by m.act_id having count(*)>=2));

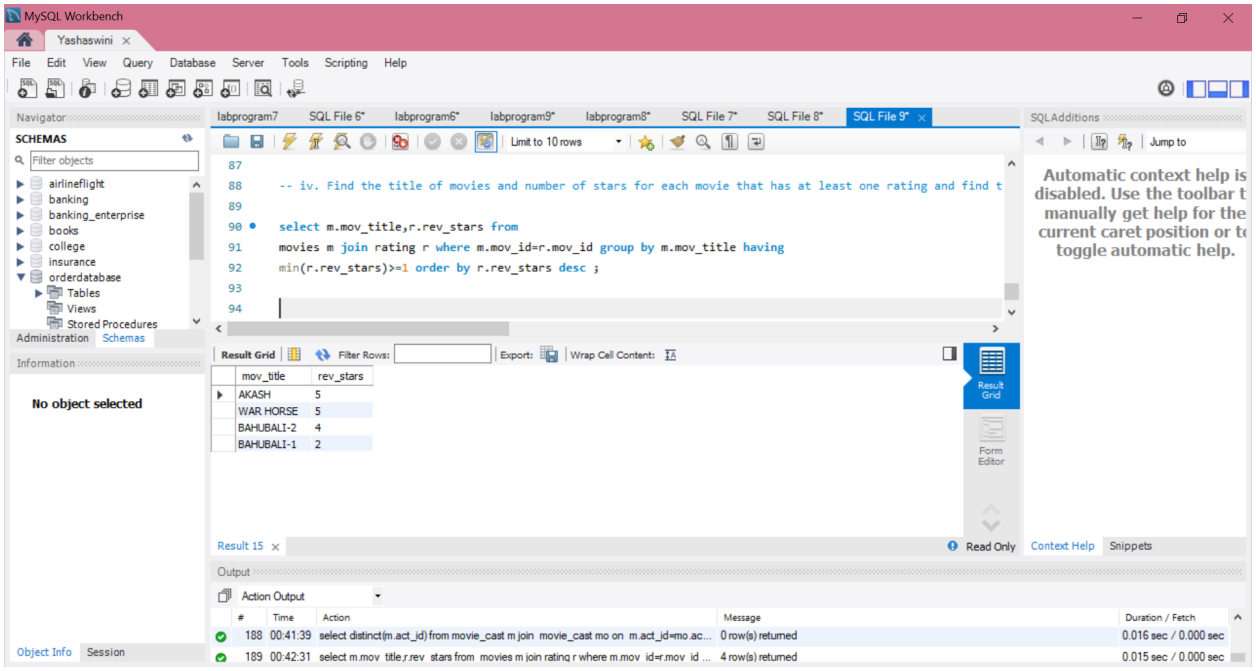


-- iii. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).

select distinct(m.act_id) from movie_cast m join movie_cast mo on m.act_id=mo.act_id and m.mov_id!=mo.mov_id where m.mov_id in (select mo.mov_id from movies mo where mo.mov_year>=2015) and mo.mov_id in (select mo.mov_id from movies mo where mo.mov_year<2000);

-- 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.

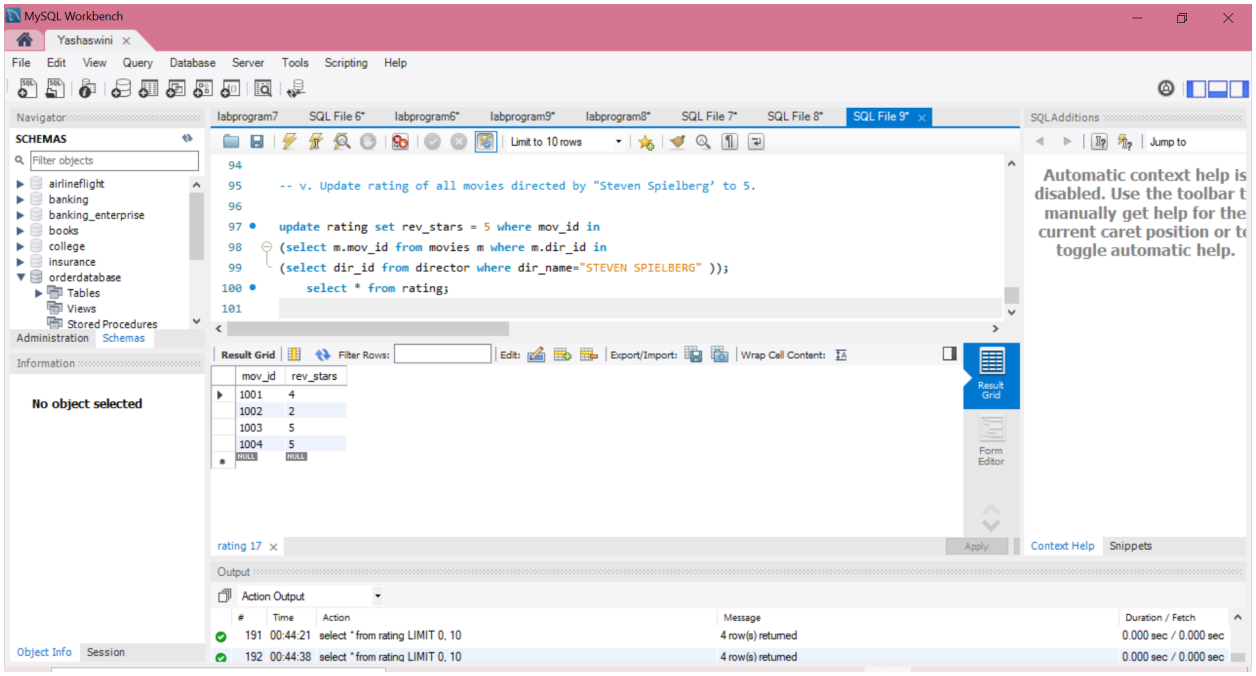
select m.mov_title,r.rev_stars from movies m join rating r where m.mov_id=r.mov_id group by m.mov_title having min(r.rev_stars)>=1 order by r.rev_stars desc ;



-- v. Update rating of all movies directed by "Steven Spielberg' to 5.

update rating set rev_stars = 5 where mov_id in (select m.mov_id from movies m where m.dir_id in (select dir_id from director where dir_name="STEVEN SPIELBERG"));

select * from rating;



PROGRAM 10:COLLEGE DATABASE

Consider the schema for College Database:
STUDENT(USN, SName, Address, Phone, Gender)
SEMSEC(SSID, Sem, Sec) CLASS(USN, SSID)
SUBJECT(Subcode, Title, Sem, Credits)
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.

```
create database college;
        use college;
        create table student (
        usn varchar(10),
        sname varchar(10),
```

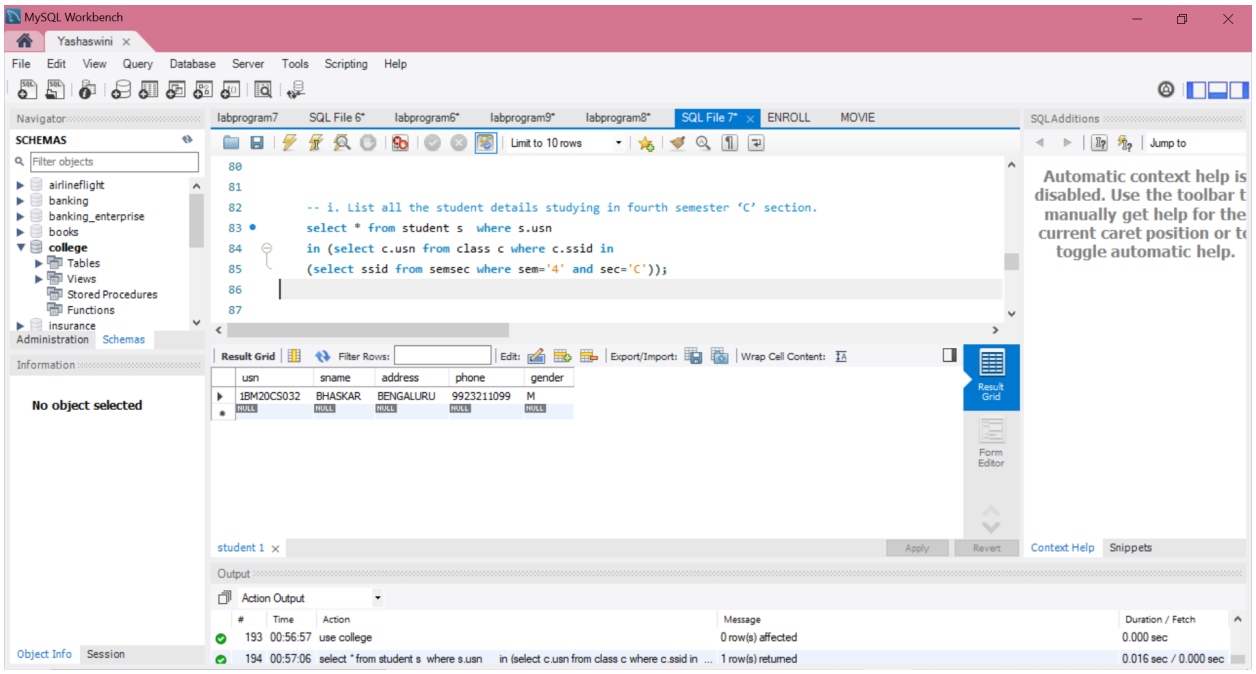
```
        address varchar (30),
        phone varchar(10),
        gender varchar(1),
        constraint stu_usn primary key (usn)
        );
        create table semsec (
                ssid varchar(5),
                sem varchar(2),
                sec varchar(1),
                constraint sem_ssid primary key (ssid)
        );
        create table class(
        usn varchar(10),
        ssid varchar(5),
        constraint class_usn primary key (usn),
        constraint class_usn foreign key (usn) references student (usn) on delete cascade on
update cascade,
        constraint class_ssid foreign key (ssid) references  semsec(ssid) on delete cascade on
update cascade
        );
        create table subjects(
        subcode varchar(10),
        title varchar (20),
        sem int,
        credits int,
        constraint sub_sub primary key (subcode)
        );
        create table iamarks(
        usn varchar(10),
        subcode varchar(10),
        ssid varchar(5),
        test1 int,
        test2 int,
        test3 int,
        finalia int,
        constraint ia primary key (usn,subcode,ssid),
        constraint ia_usn foreign key (usn) references  student(usn) on delete cascade on
update cascade ,
        constraint ia_subcode foreign key (subcode) references  subjects(subcode) on delete
cascade on update cascade,
        constraint ia_ssid foreign key (ssid) references  semsec(ssid) on delete cascade on
update cascade
        );
```

```
INSERT INTO STUDENT VALUES ('1BM19CS020','AKSHAY','BELAGAVI',
8877881122,'M');
INSERT INTO STUDENT VALUES ('1BM19CS062','SANDHYA','BENGALURU',
7722829912,'F');
INSERT INTO STUDENT VALUES ('1BM19CS091','TEESHA','BENGALURU',
7712312312,'F');
INSERT INTO STUDENT VALUES ('1BM19CS066','SUPRIYA','MANGALURU',
8877881122,'F');
INSERT INTO STUDENT VALUES ('1BM20CS010','ABHAY','BENGALURU',
9900211201,'M');
INSERT INTO STUDENT VALUES ('1BM20CS032','BHASKAR','BENGALURU',
9923211099,'M');
INSERT INTO STUDENT VALUES ('1BM20CS025','ASMI','BENGALURU',
7894737377,'F');
INSERT INTO STUDENT VALUES ('1BM20CS011','AJAY','TUMKUR', 9845091941,'M');


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 ('CSE6A', 6,'A');
INSERT INTO SEMSEC VALUES ('CSE6B', 6,'B');
INSERT INTO SEMSEC VALUES ('CSE6C', 6,'C');
INSERT INTO SEMSEC VALUES ('CSE8A', 8,'A');
INSERT INTO SEMSEC VALUES ('CSE8B', 8,'B');
INSERT INTO SEMSEC VALUES ('CSE8C', 8,'C');

INSERT INTO CLASS VALUES ('1BM19CS020','CSE8A');
INSERT INTO CLASS VALUES ('1BM19CS062','CSE4A');
INSERT INTO CLASS VALUES ('1BM19CS066','CSE8B');
INSERT INTO CLASS VALUES ('1BM19CS091','CSE8C');
INSERT INTO CLASS VALUES ('1BM20CS010','CSE4A');
INSERT INTO CLASS VALUES ('1BM20CS025','CSE4B');
INSERT INTO CLASS VALUES ('1BM20CS032','CSE4C');

INSERT INTO SUBJECTS VALUES ('10CS81','ACA', 8, 4);
INSERT INTO SUBJECTS VALUES ('10CS82','SSM', 8, 4);
INSERT INTO SUBJECTS VALUES ('10CS83','NM', 8, 4);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1BM19CS091','10CS81','CSE8C', 15, 16, 18);
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1BM19CS091','10CS82','CSE8C', 12, 19, 14);
```

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1BM19CS091','10CS83','CSE8C', 19, 15, 20);

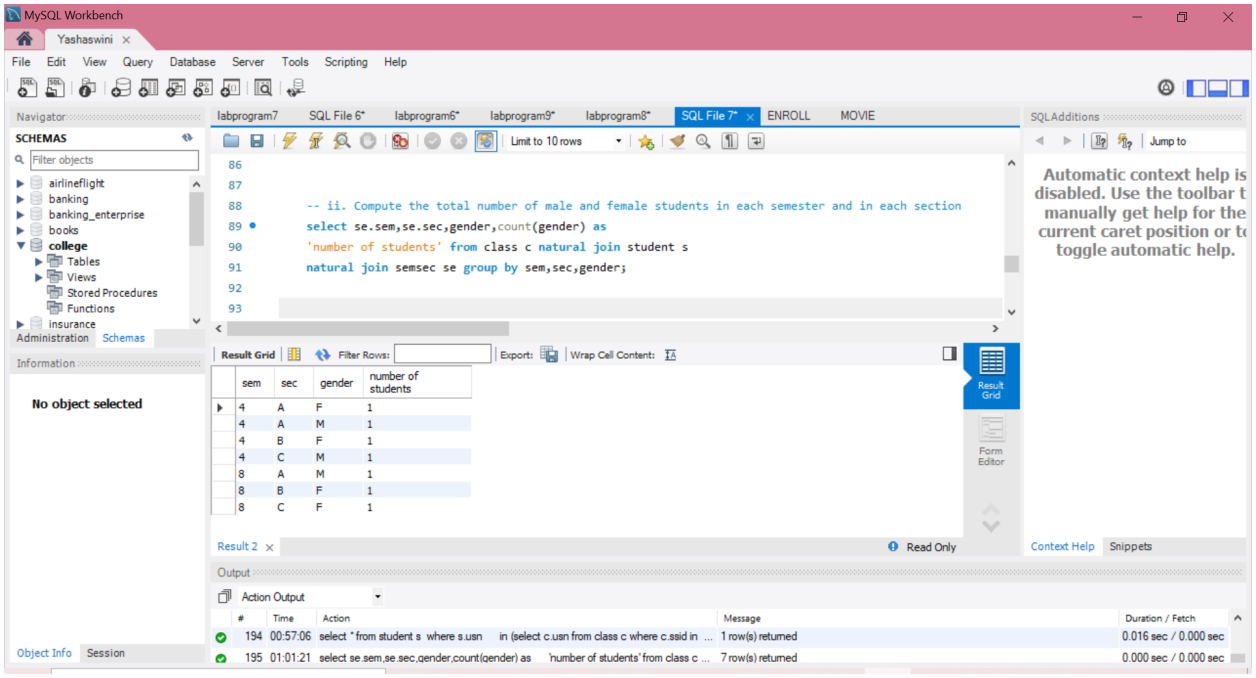
-- i. List all the student details studying in fourth semester ‘C’ section.

```
select * from student s  where s.usn
in (select c.usn from class c where c.ssid in
(select ssid from semsec where sem='4' and sec='C'));
```



-- ii. Compute the total number of male and female students in each semester and in each section

```
select se.sem,se.sec,gender,count(gender) as 'number of students' from class c natural join student s natural join semsec se group by sem,sec,gender;
```



-- iii. Create a view of Test1 marks of student USN '1BI15CS101' in all subjects.

create view marks(test1) as select i.test1 from iamarks i where i.usn='1BM19CS091';

-- iv.calculate finalia marks and update the respective column for all student;

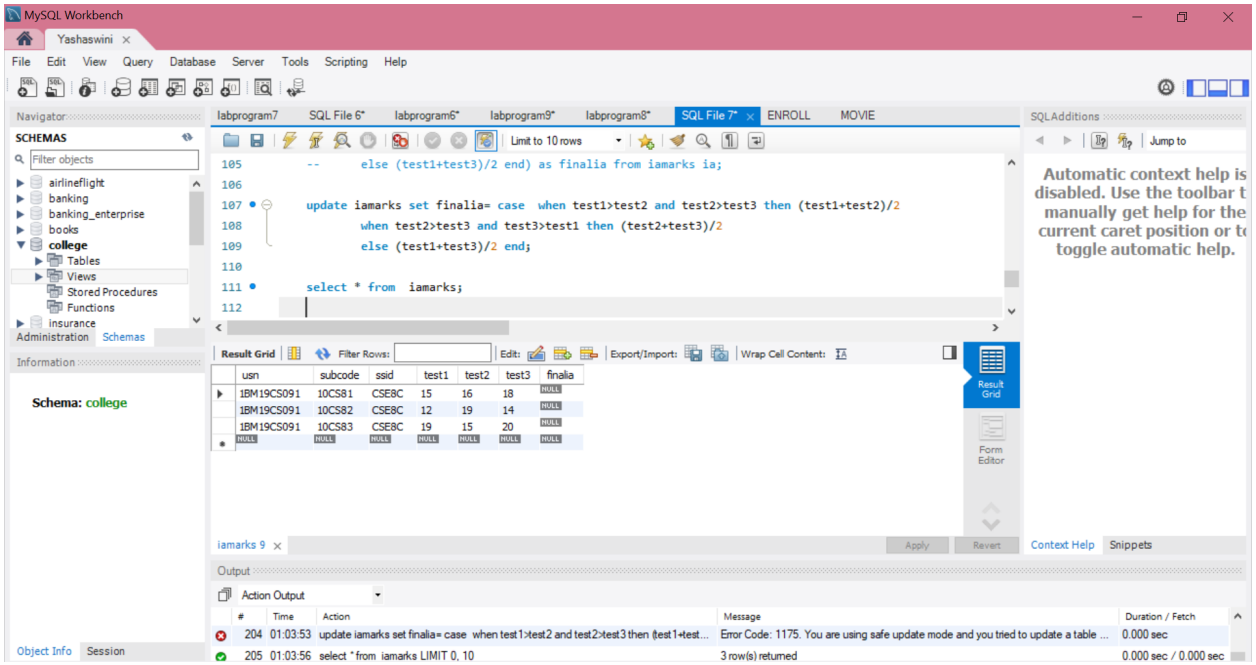
-- select ia.usn,test1,test2,test3,(case

-- when test1>test3 and test2>test3 then (test1+test2)/2

-- when test2>test1 and test3>test1 then (test2+test3)/2

-- else (test1+test3)/2 end) as finalia from iamarks ia;

```
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:

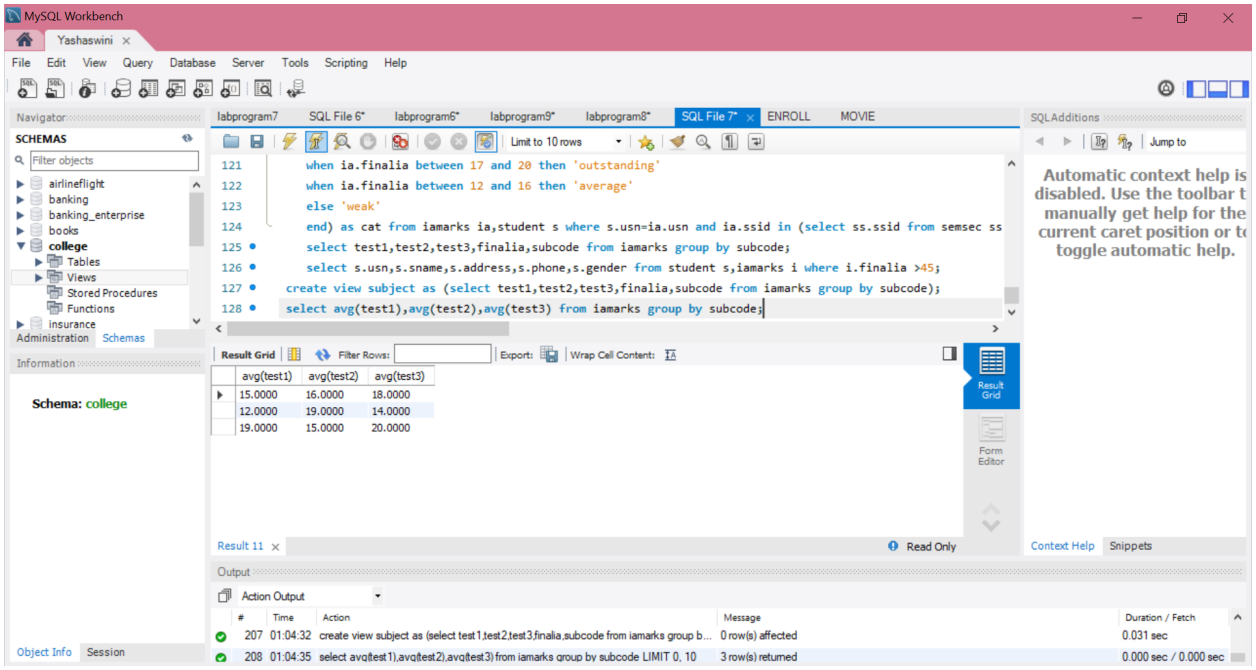
-- If FinalIA = 17 to 20 then CAT = 'Outstanding'

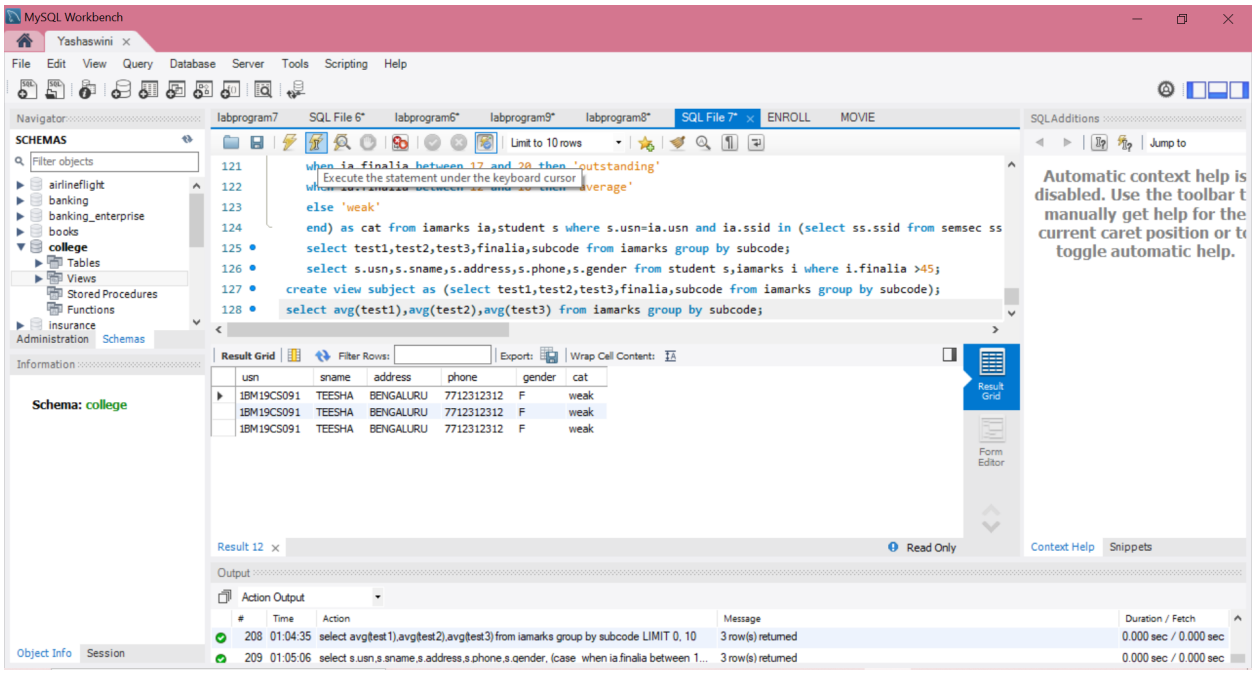
-- If FinalIA = 12 to 16 then CAT = 'Average'

-- 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.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 iamarks ia,student s where s.usn=ia.usn and ia.ssid in (select ss.ssid
from semsec ss where sem='8');
   select test1,test2,test3,finalia,subcode from iamarks group by subcode;
   select s.usn,s.sname,s.address,s.phone,s.gender from student s,iamarks i where i.finalia
>45;
 create view subject as (select test1,test2,test3,finalia,subcode from iamarks group by subcode);
 select avg(test1),avg(test2),avg(test3) from iamarks group by subcode;
```





1BM19CS216

THANK YOU