B.M.S. COLLEGE OF ENGINEERING, BANGALORE-19

(Autonomous College under VTU)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DATABASE MANAGEMENT SYSTEM LABORATORY RECORD

NAME:NEHA CATHRIN A
USN:1BM19CS099
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

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)

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

a.Update the damage amount for the car with a specific Regno in the accident with report number 12 to

25000.

b. Add a new accident to the database.

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

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

```
show databases;

use insurance;

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

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

```
create table accident(report_number int,accd_date date,location
varchar(20),primary key(report_number));

create table owns(driver_id varchar(10),regno varchar(10),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(10),regno
varchar(10),report_number int,

damage_amt float, foreign key (driver_id,regno) references
owns(driver_id,regno) on  delete cascade,

foreign key (report_number) references accident(report_number) on
delete cascade);


insert into person values(1111,"ramu","k s layout");

insert into person values(2222,"john","indiranagar");

insert into person values(3333,"priya","jayanagar");

insert into person values(4444,"gopal","whitefield");

insert into person values(5555,"latha","vijaynagar");

select * from person;


insert into car values("KA04Q2301","MARUTHI-DX",2000);

insert into car values("KA05P1000","FORDICON",2000);

insert into car values("KA03L1234","ZEN-VXI",1999);

insert into car values("KA03L9999","MARUTHI-DX",2002);

insert into car values("KA01P4020","INDICA-VX",2002);

select * from car;
```

```
desc accident;
insert into accident values(12,'2002-06-01','m g road');
insert into accident values(200,'2002-12-10','doubleroad');
insert into accident values(300,'1999-07-23','m g road');
insert into accident values(25000,'2000-06-11','residency road');
insert into accident values(26500,'2001-10-01','richmond road');
select * from accident;

insert into owns values(1111,"KA04Q2301");
insert into owns values(1111,"KA05P1000");
insert into owns values(2222,"KA03L1234");
insert into owns values(3333,"KA03L9999");
insert into owns values(4444,"KA01P4020");

insert into participated values(1111,"KA04Q2301",12,20000);
insert into participated values(2222,"KA03L1234",200,500);
insert into participated values(3333,"KA03L9999",300,10000);
insert into participated values(4444,"KA01P4020",25000,2375);
insert into participated values(1111,"KA05P1000",26500,70000);
insert into participated values(1111,"KA05P1000",300,50000);
select * from participated;


update participated set damage_amt=25000 where report_number=12
and regno='KA04Q2301';
```

insert into accident values (5555,'2009-09-10','brigade road');

select * from accident;

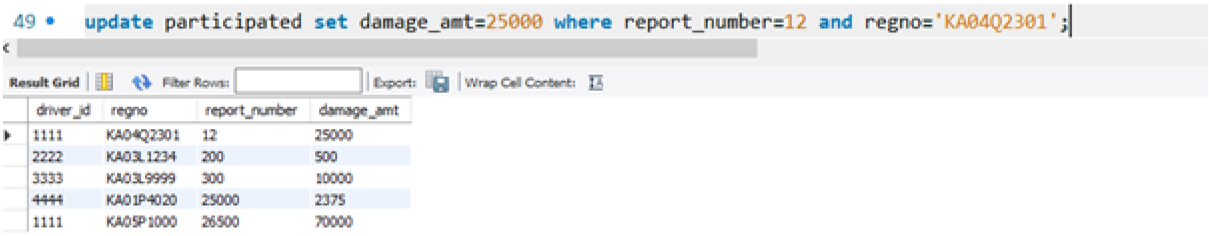
select count(*) from accident where accd_date like '2008-__-__' ;

select count(A.report_number) from accident A,participated P,car C

 where A.report_number=P.report_number and

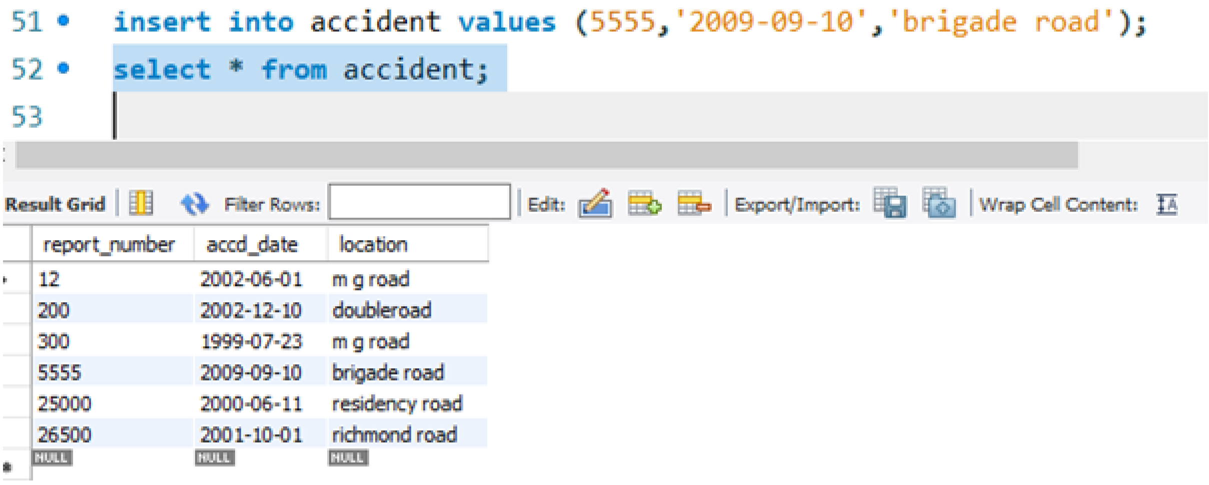
 P.regno=C.regno and

 C.model='maruthi-dx';



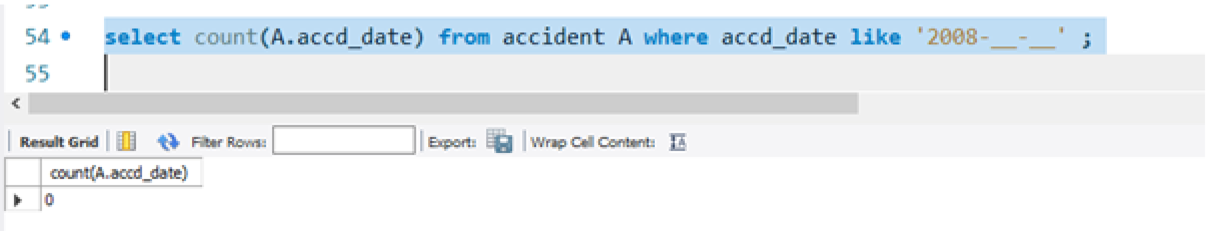
```
update participated set damage_amt=25000 where report_number=12 and regno='KA04Q2301';
```

driver_id	regno	report_number	damage_amt
1111	KA04Q2301	12	25000
2222	KA03L1234	200	500
3333	KA03L9999	300	10000
4444	KA01P4020	25000	2375
1111	KA05P1000	26500	70000



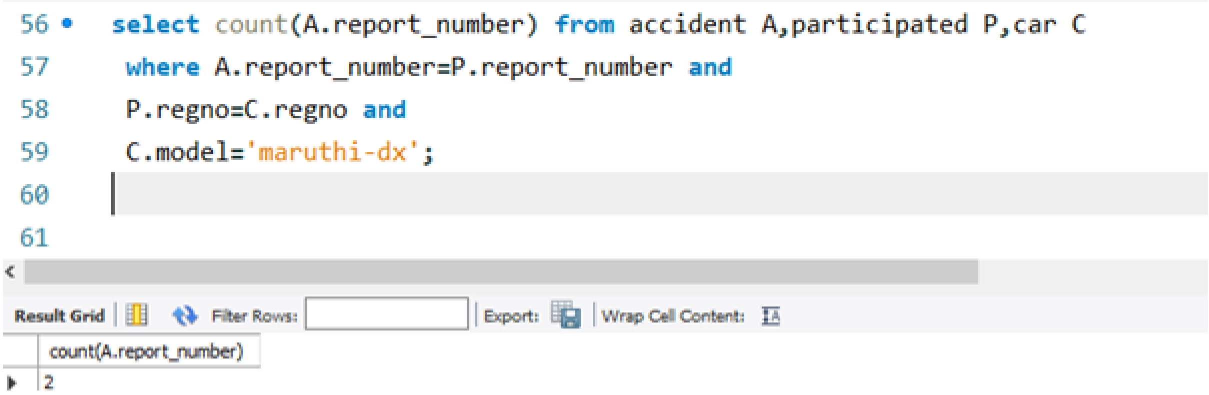
```
insert into accident values (5555,'2009-09-10','brigade road');
select * from accident;
```

report_number	accd_date	location
12	2002-06-01	m g road
200	2002-12-10	doubleroad
300	1999-07-23	m g road
5555	2009-09-10	brigade road
25000	2000-06-11	residency road
26500	2001-10-01	richmond road
NULL	NULL	NULL



```
select count(A.accd_date) from accident A where accd_date like '2008-__-__' ;
```

count(A.accd_date)
0



```
select count(A.report_number) from accident A,participated P,car C
 where A.report_number=P.report_number and
 P.regno=C.regno and
 C.model='maruthi-dx';
```

```
count(A.report_number)
2
```

PROGRAM 2. BOOK DEALER DATABASE

The following tables are maintained by a book dealer:

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

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

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

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

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

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

ii. Enter at least five tuples for each relation.

iii. Give the details of the authors who have 2 or more books in the catalog and the price of the books in the

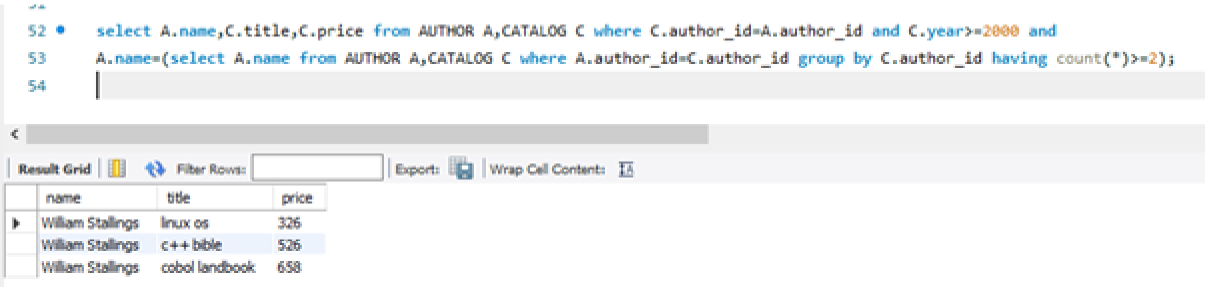
catalog and the year of publication is after 2000.

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

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

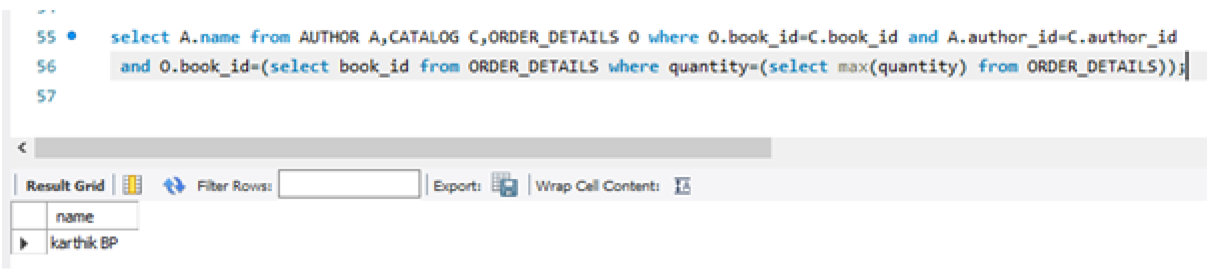
```
create database Book_dealer;

use Book_dealer;

create table AUTHOR( author_id int primary key, name varchar(30), city
varchar(20), country varchar(20));

create table PUBLISHER( publisher_id int primary key, name varchar(30),
city varchar(20), country varchar(20));


create table CATEGORY( category_id int, description varchar(50), 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,

primary key(book_id), foreign key(author_id) references
AUTHOR(author_id), foreign key(publisher_id) references
PUBLISHER(publisher_id),

 foreign key(category_id) references CATEGORY(category_id));


create table ORDER_DETAILS( order_no int primary key, book_id int,
quantity int, foreign key(book_id) references CATALOG(book_id));

show tables;


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 BP","New York","USA");
insert into AUTHOR values(1005,"William Stallings","Las Vegas","USA");
select * from AUTHOR;

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,"Willy","Berlin","Germany");
insert into PUBLISHER values(5,"MGH","New York","USA");
select * from PUBLISHER;

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 system");
select * from CATEGORY;

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 landbook",1005,4,1001,2000,658);
select * from CATALOG;
```

```
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);

select * from ORDER_DETAILS;


select A.name,C.title,C.price from AUTHOR A,CATALOG C where
C.author_id=A.author_id and C.year>=2000 and

A.name=(select A.name from AUTHOR A,CATALOG C where
A.author_id=C.author_id group by C.author_id having count(*)>=2);


select A.name from AUTHOR A,CATALOG C,ORDER_DETAILS O where
O.book_id=C.book_id and A.author_id=C.author_id

 and O.book_id=(select book_id from ORDER_DETAILS where
quantity=(select max(quantity) from ORDER_DETAILS));


 update CATALOG set price=price+(10*price/100)  where publisher_id=5
;

select * from CATALOG;
```



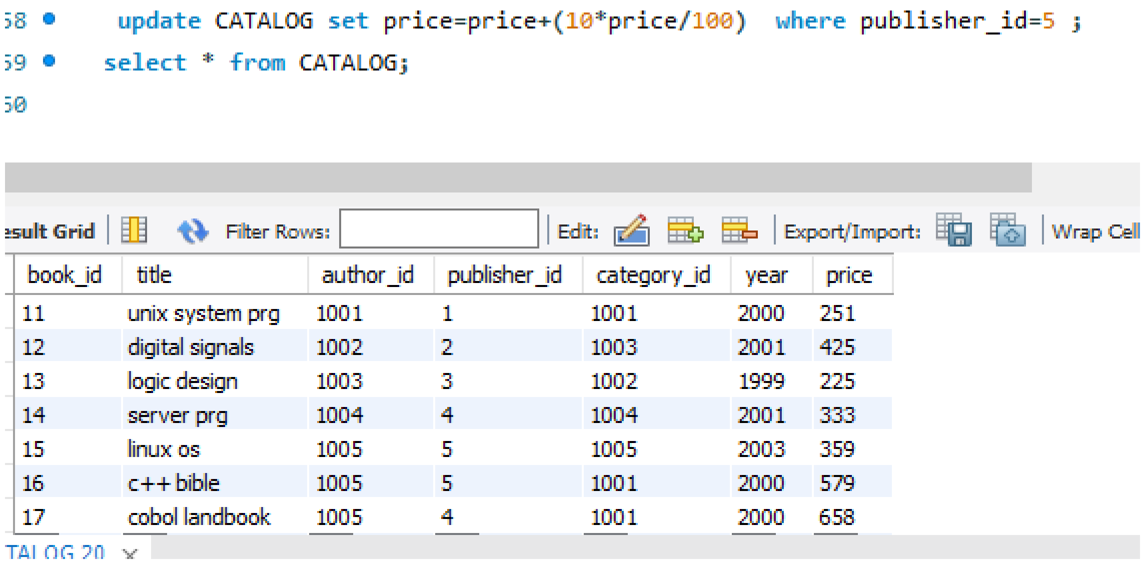
```
select A.name,C.title,C.price from AUTHOR A,CATALOG C where C.author_id=A.author_id and C.year>=2000 and
A.name=(select A.name from AUTHOR A,CATALOG C where A.author_id=C.author_id group by C.author_id having count(*)>=2);
```

name	title	price
William Stallings	linux os	326
William Stallings	c++ bible	526
William Stallings	cobol landbook	658



```sql
select A.name from AUTHOR A,CATALOG C,ORDER_DETAILS O where O.book_id=C.book_id and A.author_id=C.author_id
 and O.book_id=(select book_id from ORDER_DETAILS where quantity=(select max(quantity) from ORDER_DETAILS));
```

name
karthik BP



```sql
 update CATALOG set price=price+(10*price/100)  where publisher_id=5 ;
select * from CATALOG;
```

book_id	title	author_id	publisher_id	category_id	year	price
11	unix system prg	1001	1	1001	2000	251
12	digital signals	1002	2	1003	2001	425
13	logic design	1003	3	1002	1999	225
14	server prg	1004	4	1004	2001	333
15	linux os	1005	5	1005	2003	359
16	c++ bible	1005	5	1001	2000	579
17	cobol landbook	1005	4	1001	2000	658

PROGRAM 3. ORDER PROCESSING DATABASE

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

CUSTOMER (CUST #: int, cname: String, city: String)

ORDER (order #: int, odate: date, cust #: int, ord-Amt: int)

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

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

WAREHOUSE (warehouse #: int, city: String)

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

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

ii. Enter at least five tuples for each relation.

iii. Produce a listing: CUSTNAME, #oforders, AVG_ORDER_AMT, where the middle column is the total

numbers of orders by the customer and the last column is the average order amount for that customer.

iv. List the order# for orders that were shipped from all warehouses that the company has in a specific city.

v. Demonstrate how you delete item# 10 from the ITEM table and make that field null in the ORDER_ITEM

table.

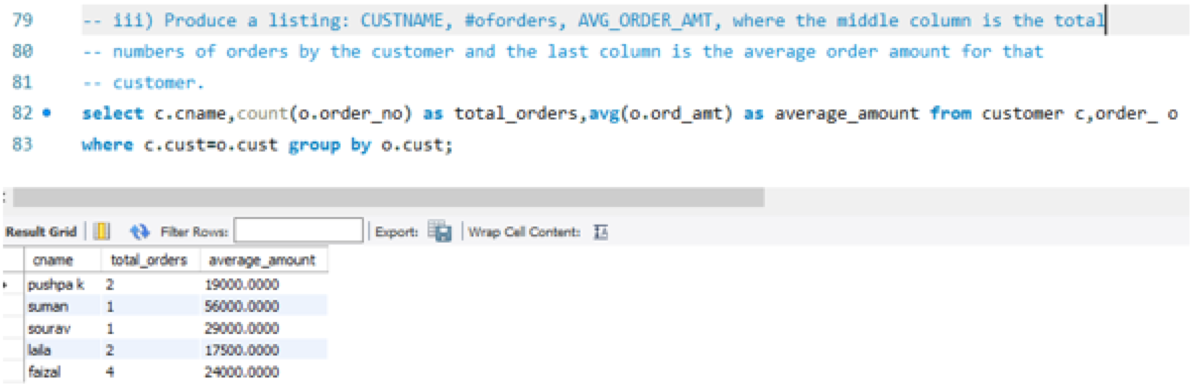
```
create database order_processing;

use order_processing;

create table customer(cust int primary key,cname varchar(20),city varchar(20));

create table order_(order_no int primary key,odate date,cust int ,ord_amt int,

foreign key(cust) references customer(cust) on delete cascade);

create table item(item_no int primary key,unit_price int);

create table order_item(order_no int,item_no int ,qty int,

foreign key(order_no) references order_(order_no)on delete cascade,

foreign key(item_no) references item(item_no)on delete cascade);

create table warehouse(warehouse_no int primary key,city varchar(20));

create table shipment(order_no int,warehouse_no int ,ship_date date,

foreign key(order_no) references order_(order_no) on delete cascade,

foreign key(warehouse_no) references warehouse(warehouse_no) on delete cascade);

show tables;
```

```
drop table order_item;
insert into customer values(771,"pushpa k","bangalore");
insert into customer values(772,"suman","mumbai");
insert into customer values(773,"sourav","calicut");
insert into customer values(774,"laila","hyderabad");
insert into customer values(775,"faizal","bangalore");
select * from customer;

insert into order_ values(111,'2002-01-22',771,18000);
insert into order_ values(112,'2002-07-30',774,6000);
insert into order_ values(113,'2003-04-03',775,9000);
insert into order_ values(114,'2003-11-03',775,29000);
insert into order_ values(115,'2003-12-10',773,29000);
insert into order_ values(116,'2004-08-19',772,56000);
insert into order_ values(117,'2004-09-10',771,20000);
insert into order_ values(118,'2004-11-20',775,29000);
insert into order_ values(119,'2005-02-13',774,29000);
insert into order_ values(120,'2005-10-13',775,29000);
select * from order_;

insert into item values(5001,503);
insert into item values(5002,750);
insert into item values(5003,150);
insert into item values(5004,600);
insert into item values(5005,890);
```

```
select * from item;

insert into order_item values(111,5001,50);
insert into order_item values(112,5003,20);
insert into order_item values(113,5002,50);
insert into order_item values(114,5005,60);
insert into order_item values(115,5004,90);
insert into order_item values(116,5001,10);
insert into order_item values(117,5003,80);
insert into order_item values(118,5005,50);
insert into order_item values(119,5002,10);
insert into order_item values(120,5004,45);
select * from order_item;

insert into warehouse values(1,"delhi");
insert into warehouse values(2,"bombay");
insert into warehouse values(3,"chennai");
insert into warehouse values(4,"bangalore");
insert into warehouse values(5,"bangalore");
insert into warehouse values(6,"delhi");
insert into warehouse values(7,"bombay");
insert into warehouse values(8,"chennai");
insert into warehouse values(9,"delhi");
insert into warehouse values(10,"bangalore");
select * from warehouse;
```

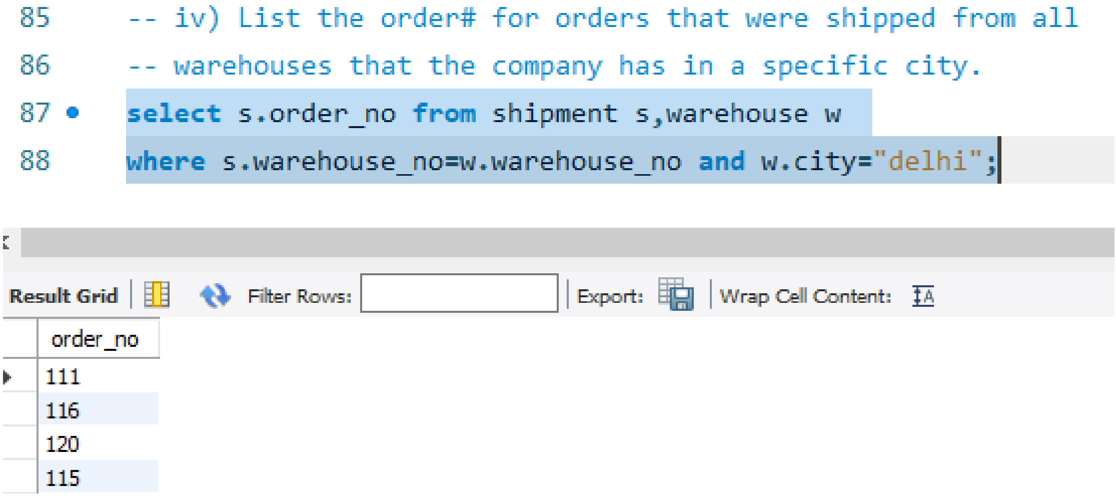
```
insert into shipment values(111,1,'2002-02-10');
insert into shipment values(112,5,'2002-09-10');
insert into shipment values(113,8,'2003-02-10');
insert into shipment values(114,3,'2003-12-10');
insert into shipment values(115,9,'2004-01-19');
insert into shipment values(116,1,'2004-09-20');
insert into shipment values(117,5,'2004-09-10');
insert into shipment values(118,7,'2004-11-30');
insert into shipment values(119,7,'2005-04-30');
insert into shipment values(120,6,'2005-12-21');
select * from shipment;


update shipment set ship_date='2001-11-20' where warehouse_no=6;


-- 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_no) as total_orders,avg(o.ord_amt) as average_amount from customer c,order_ o
where c.cust=o.cust group by o.cust;


-- iv) List the order# for orders that were shipped from all
-- warehouses that the company has in a specific city.
```

```
select s.order_no from shipment s,warehouse w

where s.warehouse_no=w.warehouse_no and w.city="delhi";

-- v) Demonstrate how you delete item# 10 from the ITEM table and

-- make that field null in theORDER_ITEM table.

delete from item where item_no=5005;

select * from order_item;
```



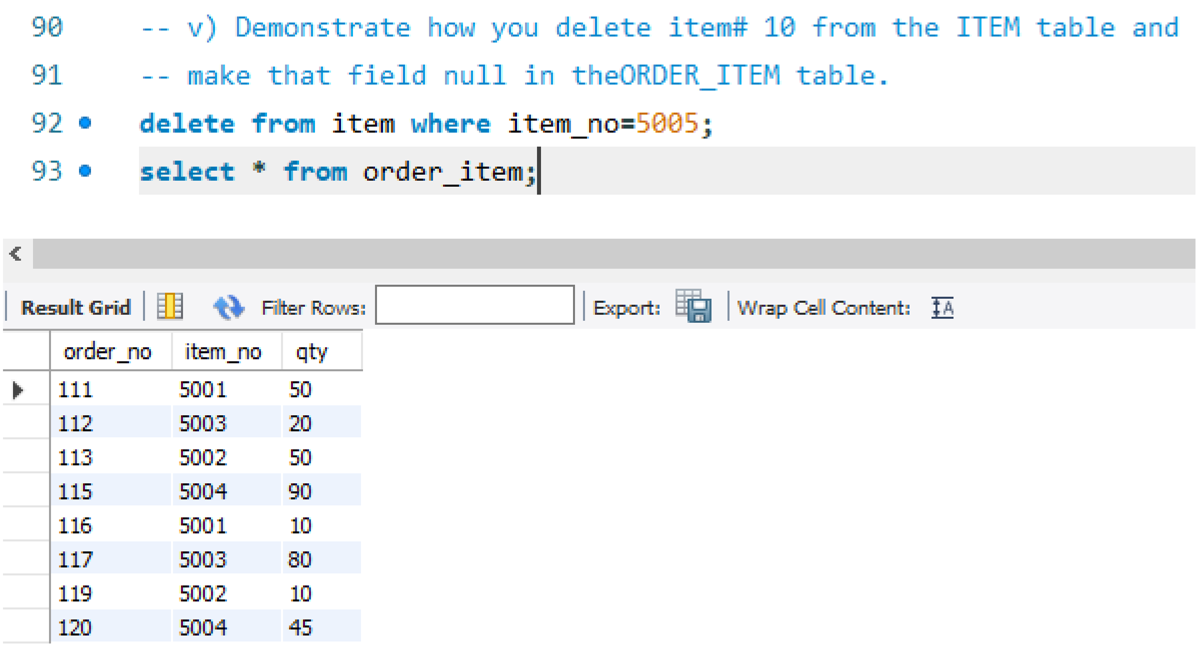
```
-- iii) Produce a listing: CUSTNAME, #oforders, AVG_ORDER_AMT, where the middle column is the total
-- numbers of orders by the customer and the last column is the average order amount for that
-- customer.
select c.cname,count(o.order_no) as total_orders,avg(o.ord_amt) as average_amount from customer c,order_ o
where c.cust=o.cust group by o.cust;
```

cname	total_orders	average_amount
pushpa k	2	19000.0000
suman	1	56000.0000
sourav	1	29000.0000
laila	2	17500.0000
faizal	4	24000.0000



```
-- iv) List the order# for orders that were shipped from all
-- warehouses that the company has in a specific city.
select s.order_no from shipment s,warehouse w
where s.warehouse_no=w.warehouse_no and w.city="delhi";
```

order_no
111
116
120
115



```
-- v) Demonstrate how you delete item# 10 from the ITEM table and
-- make that field null in theORDER_ITEM table.
delete from item where item_no=5005;
select * from order_item;
```

order_no	item_no	qty
111	5001	50
112	5003	20
113	5002	50
115	5004	90
116	5001	10
117	5003	80
119	5002	10
120	5004	45

PROGRAM 4. 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)

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.

```
create database banking_enterprise;

use banking_enterprise;

create table branch(branch_name varchar(20) primary key,branch_city varchar(20),assets real);

create table accounts(acc_no int primary key,branch_name varchar(20),balance real, foreign key(branch_name)
references branch(branch_name) on delete cascade);

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

create table depositor(customer_name varchar(20),acc_no int,
foreign key(customer_name) references customer(customer_name) on delete cascade,
foreign key(acc_no) references accounts(acc_no) on delete cascade);

create table loan(loan_number int primary key,branch_name varchar(20),amount int,
foreign key(branch_name) references branch(branch_name) on delete cascade);


create table borrower(customer_name varchar(20),loan_number int,

foreign key(customer_name) references customer(customer_name) on delete cascade,

foreign key(loan_number) references loan(loan_number) on delete cascade);

show tables;



insert into branch values("SBI PD Nagar","Bangalore",200000);

insert into branch values("SBI Rajaji Nagar","Bangalore",500000);

insert into branch values("SBI Jayanagar","Delhi",660000);

insert into branch values("SBI Vijay Nagar","Chennai",870000);

insert into branch values("SBI Hosakerehalli","Bangalore",550000);

select * from branch;


insert into accounts values(11,"SBI Hosakerehalli",5000);

insert into accounts values(22,"SBI Vijay Nagar",5000);

insert into accounts values(33,"SBI Jayanagar",5000);

insert into accounts values(44,"SBI Rajaji Nagar",10000);

insert into accounts values(55,"SBI Vijay Nagar",40000);

insert into accounts values(66,"SBI PD Nagar",4000);

insert into accounts values(77,"SBI PD Nagar",40000);

insert into accounts values(88,"SBI Rajaji Nagar",4000);
```

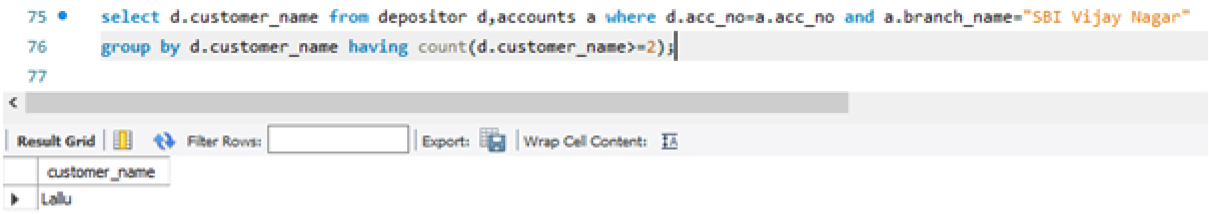
```
select * from accounts;


insert into customer values("Kezar","MG road","Bangalore");

insert into customer values("Lal Krishna","ST MKS road","Bangalore");

insert into customer values("Rahul","Augsten road","Bangalore");

insert into customer values("Lallu","V S road","Bangalore");

insert into customer values("Faizal","Resedency road","Bangalore");

insert into customer values("Rajeev","Dicknsn road","Bangalore");

select * from customer;


insert into depositor values("Rahul",11);

insert into depositor values("Lallu",22);

insert into depositor values("Rahul",33);

insert into depositor values("Faizal",44);

insert into depositor values("Lallu",55);

insert into depositor values("Kezar",66);

insert into depositor values("Rajeev",77);

insert into depositor values("Lal Krishna",88);

select * from depositor;



insert into loan values(10011,"SBI Jayanagar",10000);

insert into loan values(10012,"SBI Vijay Nagar",5000);

insert into loan values(10013,"SBI Hosakerehalli",20000);

insert into loan values(10014,"SBI PD Nagar",15000);
insert into loan values(10015,"SBI Rajaji Nagar",25000);

select * from loan;


insert into borrower values("Kezar",10011);

insert into borrower values("Lal Krishna",10012);

insert into borrower values("Rahul",10013);

insert into borrower values("Lallu",10014);

insert into borrower values("Lal Krishna",10015);

select * from borrower;


-- iii) 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.acc_no=a.acc_no and a.branch_name="SBI Vijay Nagar"

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.acc_no = depositor.acc_no

join branch on branch.branch_name = accounts.branch_name

where branch.branch_city = "Bangalore"

GROUP BY depositor.customer_name;



-- v) Demonstrate how you delete all account tuples at every
```

-- branch located in a specific city.

delete from accounts where branch_name in

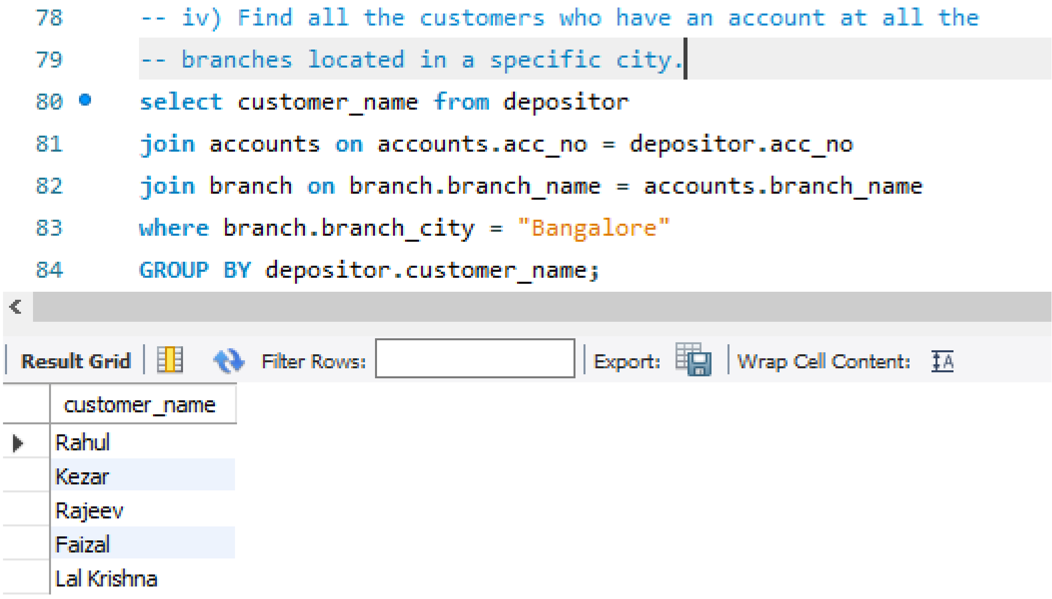
(select branch_name from branch where branch_city="delhi");

select * from accounts;



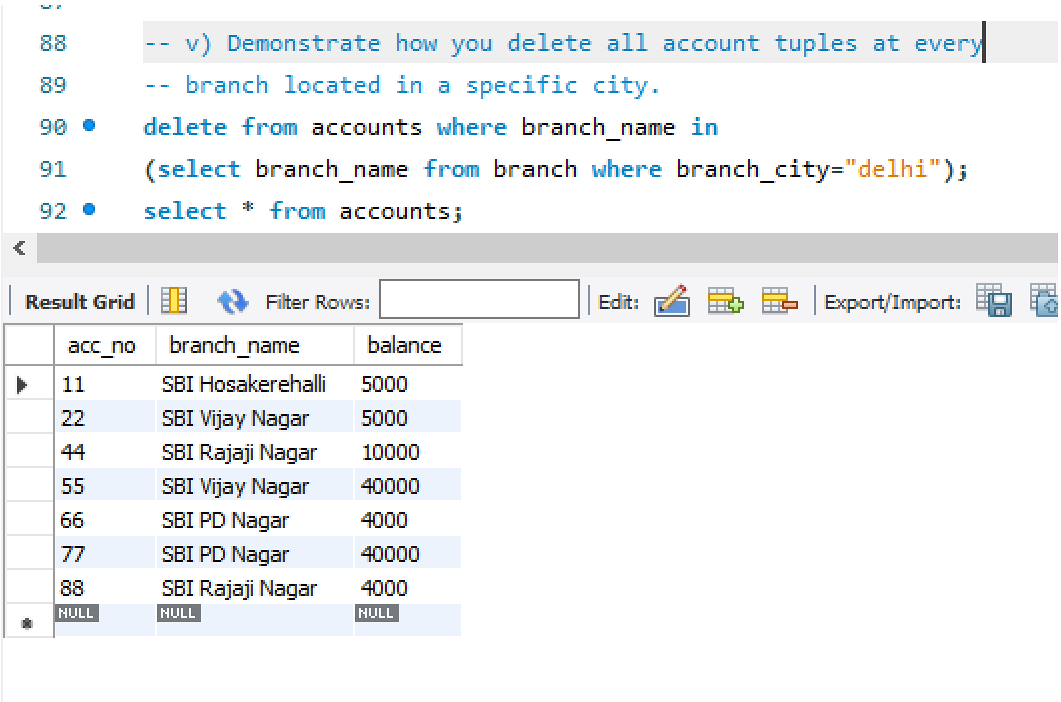
```
select d.customer_name from depositor d,accounts a where d.acc_no=a.acc_no and a.branch_name="SBI Vijay Nagar"
group by d.customer_name having count(d.customer_name>=2);
```

customer_name
Lallu



```
-- 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.acc_no = depositor.acc_no
join branch on branch.branch_name = accounts.branch_name
where branch.branch_city = "Bangalore"
GROUP BY depositor.customer_name;
```

customer_name
Rahul
Kezar
Rajeev
Faizal
Lal Krishna



```
-- 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;
```

acc_no	branch_name	balance
11	SBI Hosakerehalli	5000
22	SBI Vijay Nagar	5000
44	SBI Rajaji Nagar	10000
55	SBI Vijay Nagar	40000
66	SBI PD Nagar	4000
77	SBI PD Nagar	40000
88	SBI Rajaji Nagar	4000
NULL	NULL	NULL

PROGRAM 5. STUDENT ENROLLMENT DATABASE

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

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

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

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

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

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

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

ii. Enter at least five tuples for each relation.

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

iv. Produce a list of text books (include Course #, Book-ISBN, Book-title) in the alphabetical order for courses

offered by the ‘CS’ department that use more than two books.

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

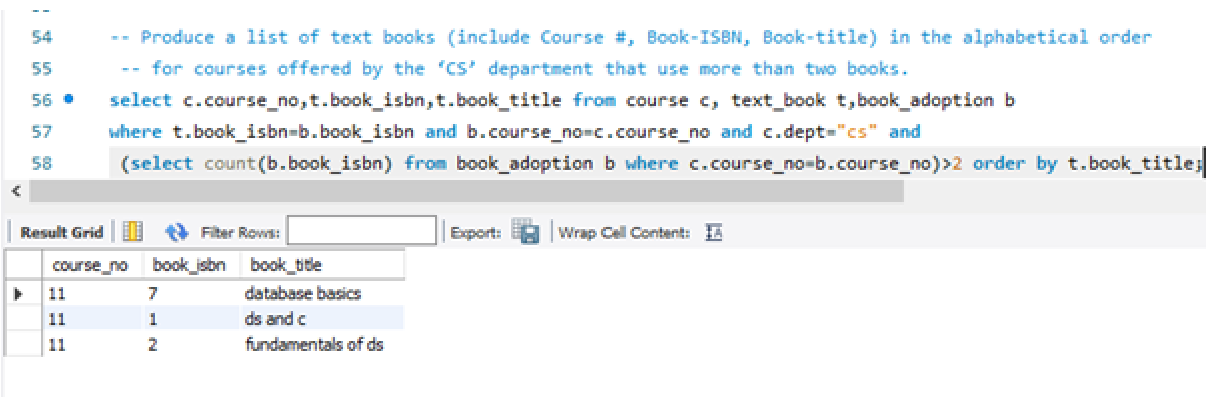
```
create database Student_Enrollment;

use Student_enrollment;

create table student(regno varchar(10) primary key,name varchar(10),major varchar(10),bdate date);

create table course(course_no int primary key,cname varchar(10),dept varchar(10));

create table enroll(regno varchar(10),course_no int,sem int, marks int,

foreign key(regno) references student(regno) on delete cascade,

foreign key(course_no) references course(course_no) on delete cascade);

create table text_book(book_isbn int primary key,book_title varchar(20),publisher varchar(10),author varchar(10));

create table book_adoption(course_no int,sem int,book_isbn int ,

foreign key(course_no) references course(course_no) on delete cascade,

foreign key(book_isbn) references text_book(book_isbn) on delete cascade);


insert into student(regno,name,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');
select * from student;

insert into course(course_no,cname,dept) values
(11,"ds","cs"),
(22,"usp","is"),
(33,"sns","ec"),
(44,"dbms","cs"),
(55,"ec","tc");
select * from course;

insert into enroll(regno,course_no,sem,marks) values
("cs01",11,4,85),
("is02",22,6,80),
("ec03",33,2,80),
("cs03",44,6,75),
("tc05",55,2,80);
select * from enroll;

insert into text_book(book_isbn,book_title,publisher,author) values
(1,"ds and c","princeton","padma"),
(2,"fundamentals of ds","princeton","godse"),
(3,"fundamentals of dbms","princeton","navathe"),
(4,"sql","princeton","foley"),
(5,"electronic circuits","tmh","elmarsi"),

(6,"adv unix program","tmh","stevens");

select * from text_book;


insert into book_adoption(course_no,sem,book_isbn) values

(11,4,1),(11,4,2),(44,6,3),(44,6,4),(55,2,5),(22,6,6);

select * from book_adoption;


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

insert into text_book values(7,"database basics","princeton","shawn");

insert into book_adoption values(11,4,7);


-- 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_no,t.book_isbn,t.book_title from course c, text_book t,book_adoption b

where t.book_isbn=b.book_isbn and b.course_no=c.course_no and c.dept="cs" and

 (select count(b.book_isbn) from book_adoption b where c.course_no=b.course_no)>2 order by t.book_title;


-- 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_book t where c.course_no=b.course_no
```

and t.book_isbn=b.book_isbn and t.publisher="tmh")

and c.dept not in (select c.dept

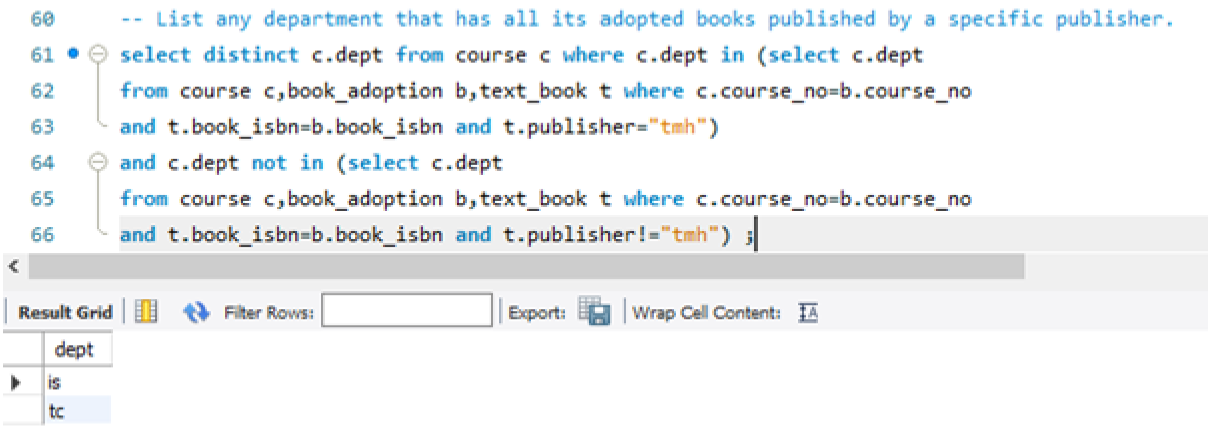
from course c,book_adoption b,text_book t where c.course_no=b.course_no

and t.book_isbn=b.book_isbn and t.publisher!="tmh") ;



```
--
-- 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_no,t.book_isbn,t.book_title from course c, text_book t,book_adoption b
where t.book_isbn=b.book_isbn and b.course_no=c.course_no and c.dept="cs" and
 (select count(b.book_isbn) from book_adoption b where c.course_no=b.course_no)>2 order by t.book_title;
```

course_no	book_isbn	book_title
11	7	database basics
11	1	ds and c
11	2	fundamentals of ds



```
 -- 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_book t where c.course_no=b.course_no
and t.book_isbn=b.book_isbn and t.publisher="tmh")
and c.dept not in (select c.dept
from course c,book_adoption b,text_book t where c.course_no=b.course_no
and t.book_isbn=b.book_isbn and t.publisher!="tmh") ;
```

dept
is
tc

Program 6: Movie database :

Consider the schema for Movie Database:

ACTOR(Act_id, Act_Name, Act_Gender)

DIRECTOR(Dir_id, Dir_Name, Dir_Phone)

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

MOVIE_CAST(Act_id, Mov_id, Role)

RATING(Mov_id, Rev_Stars)

Write SQL queries to

i. List the titles of all movies directed by ‘Hitchcock’.

ii. Find the movie names where one or more actors acted in two or more movies.

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

iv. Find the title of movies and number of stars for each movie that has at least one rating and find the highest

number of stars that movie received. Sort the result by movie title.

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

```
CREATE DATABASE MOVIE;

USE MOVIE;


CREATE TABLE ACTOR(ACT_ID INT PRIMARY KEY ,ACT_NAME
VARCHAR(30),ACT_GENDER VARCHAR(30)  );


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, 'Leonardo','MALE'),

(3, 'Robert Downey', 'MALE'),

 (4, 'Jennifer Lawrence','FEMALE'),

(5, 'Emma Stone','FEMALE');

select * from ACTOR;


INSERT INTO DIRECTOR(DIR_ID, DIR_NAME, PHONE_NO) VALUES

(1, 'Steven Spielberg', 99988776600),

(2, 'Christopher', 9988776611),

(3, 'Alfred Hitchcock', 9988776622),

(4, 'Tim Burton', 9988776633),

(5, 'James Cameron', 9988776644);

select * from DIRECTOR;
```

```
INSERT INTO
MOVIES(MOVIE_ID,MOVIE_TITLE,MOVIE_YEAR,MOVIE_LANG,DIR_ID)
VALUES
(1,'War of the Worlds', 2005, 'ENG', 1),
(2,'Titanic', 1997, 'ENG', 1),
(3,'Iron Man', 2008, 'ENG', 2),
(4,'Red Sparrow', 2018, 'ENG', 3),
(5,'Spider Man',2015, 'ENG', 4),
(6, 'Avatar', 2009, 'ENG', 5),
(7,'Mission Impossible',2017,'ENG',3);
select * from MOVIES;


INSERT INTO MOVIE_CAST(ACT_ID, MOVIE_ID,ROLE) VALUES
(1, 1, 'LEAD'),
(1, 7, 'LEAD'),
(2, 2, 'LEAD'),
(3, 3, 'LEAD'),
(4, 4, 'LEAD'),
(5, 5, 'LEAD'),
 (5,6,'CO-STAR');
 select * FROM MOVIE_CAST;


INSERT INTO RATING(MOVIE_ID, RATING_STARS) VALUES
(1, 3),
(2, 4),
(3, 5),
```

```sql
(4, 3),

(5, 4),

(6, 4),

(7, 5);


SELECT * FROM RATING;




-- 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(*)>=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 FROM MOVIES M, RATING R

WHERE M.MOVIE_ID = R.MOVIE_ID GROUP BY M.MOVIE_TITLE HAVING COUNT(R.RATING_STARS>=1) ORDER BY M.MOVIE_TITLE;

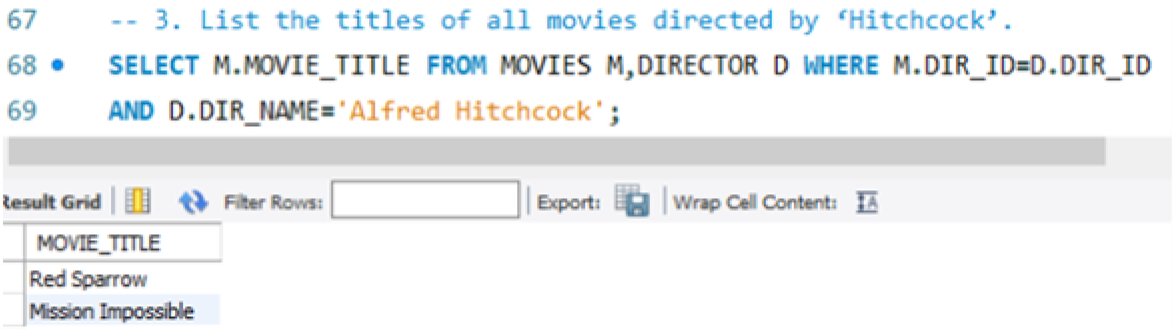
-- 7. Update rating of all movies directed by ‘Steven Spielberg’ to 5.

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

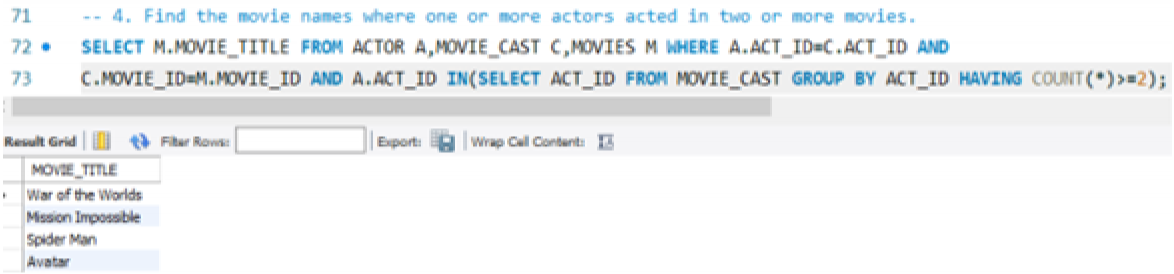
AND D.DIR_NAME='Steven Spielberg');

SELECT * FROM RATING;



```
-- 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';
```

```
MOVIE_TITLE
Red Sparrow
Mission Impossible
```



```
-- 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(*)>=2);
```

```
MOVIE_TITLE
War of the Worlds
Mission Impossible
Spider Man
Avatar
```

Program 7: Airlines Database :

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

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

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

CERTIFIED (eid: integer, aid: integer)

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

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

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

Write each of the following queries in SQL.

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

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

the aircraft for which she or he is certified.

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

Frankfurt.

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

all pilots certified for this aircraft.

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

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

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

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

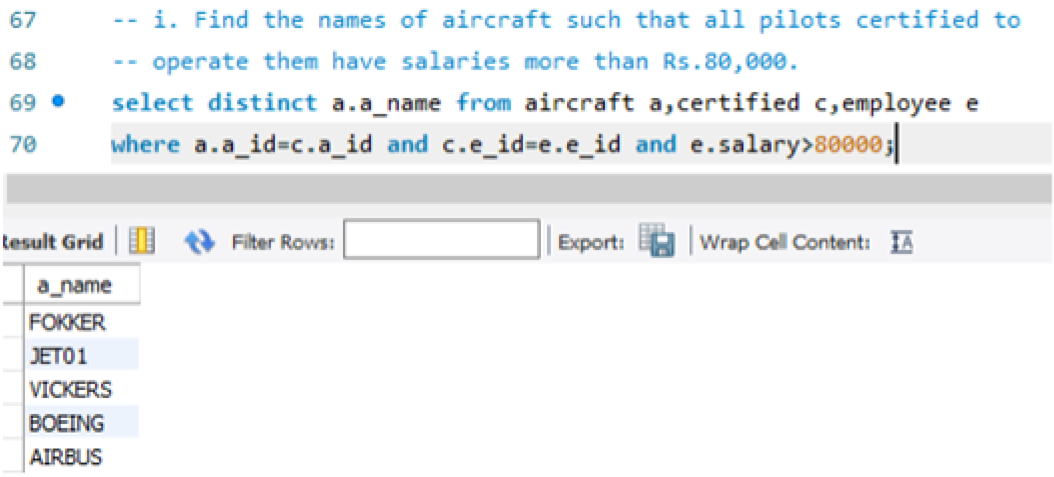
```
create database AIRLINE;

use AIRLINE;


create table flights(flno int ,from_city varchar(20),to_city
varchar(20),distance int,

departs time, arrives time ,price int );

create table aircraft(a_id int primary key ,a_name
varchar(20),cruisingrange int );

create table employee(e_id int primary key ,e_name varchar(20),salary
int);

create table certified(e_id int,a_id int,

foreign key(a_id) references aircraft(a_id) on delete cascade,

foreign key(e_id) references employee(e_id) on delete cascade);


insert into
flights(flno,from_city,to_city,distance,departs,arrives,price)values

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

(2,'BANGALORE','DELHI',5000,'12:15:00','04:30:00',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','05:30:00',90000),

(6,'Mumbai','Delhi',1200,'10:30:00','12:30:00',28000),

(7,'BANGALORE','FRANKFURT',17000,'12:00:00','06:30:00',99000),
(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),
(12,'LONDON','NEW YORK', 30000, '14:05:00', '17:50:00', 50000),
(11,'LONDON','NEW YORK', 31000, '14:06:00', '18:05:00', 51000),
(12,'LONDON','BERLIN', 15000, '14:06:00', '16:05:00', 17000);
select * from flights;

insert into aircraft(a_id,a_name,cruisingrange)values
(111,'AIRBUS',1000),
(222,'BOEING',5000),
(333,'JET01',5000),
(444,'DOUGLAS',8000),
(555,'ANTONOV',500),
(666,'VICKERS',800),
(777,'FOKKER',1000);
select * from aircraft;

insert into employee(e_id,e_name,salary)values (10,'DANNY',80000),
(1,'ARJUN',30000),
(2,'ARPITH',85000),
(3,'BHOOMI',50000),
(4,'HENRY',45000),
(5,'JOMIE',90000),
(6,'ANOSH',75000),
(7,'RICK',100000),

(8,'JANE',70000),

(9,'SOFIE',80000);

select * from employee;


insert into certified(e_id,a_id) values (9,222),

(1,111),

(2,777),

(2,333),

(3,555),

(4,222),

(5,666),

(5,222),

(6,333),

(6,111),

(7,111),

(8,444),

(9,555),

(9,333);

select * from certified;



-- i. Find the names of aircraft such that all pilots certified to

-- operate them have salaries more than Rs.80,000.

select distinct a.a_name from aircraft a,certified c,employee e
```

```
where a.a_id=c.a_id and c.e_id=e.e_id 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.e_id,max(a.cruisingrange) from aircraft a,employee e,certified c
where a.a_id=c.a_id and e.e_id=c.e_id group by e.e_id having count(e.e_id)>3;


-- iii. Find the names of pilots whose salary is less than the price of the
-- cheapest route from Bengaluru to Frankfurt.
select e.e_name from employee e where e.e_id in(select e_id from certified)
and salary<(select min(price) from flights where from_city="BANGALORE" and
to_city="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.a_name,avg(e.salary) from aircraft a,employee e,certified c
where a.a_id=c.a_id and e.e_id=c.e_id and a.cruisingrange>1000 group by a.a_name;


-- v. Find the names of pilots certified for some Boeing aircraft.
select e.e_name from aircraft a,employee e,certified c
where a.a_id=c.a_id and e.e_id=c.e_id and a.a_name="BOEING";
```

```
-- vi. Find the aids of all aircraft that can be used on

--  routes from Bengaluru to New Delhi.

select a_id from aircraft where cruisingrange>=(select distance from flights

where from_city="BANGALORE" and to_city="DELHI");
```

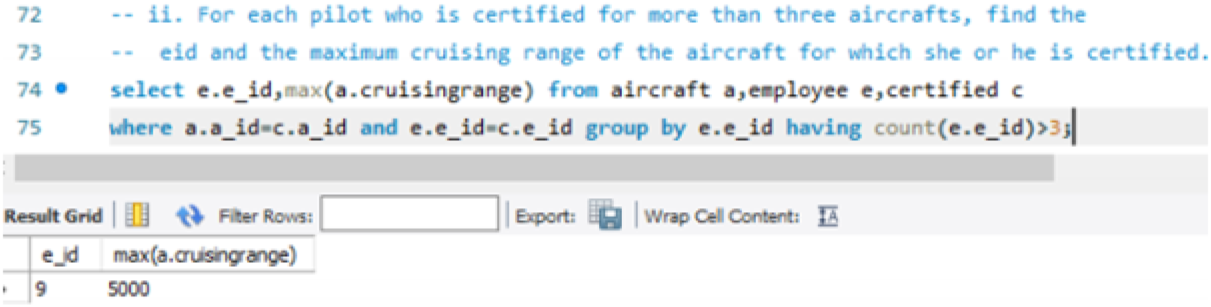
```
-- vii. A customer wants to travel from Madison to New York with no

--  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.flno ,f.departs from flights f where f.flno in ( ( select f1.flno

from flights f1 where f1.from_city="MADISON" AND f1.to_city="NEW YORK" and f1.arrives<'18:00:00')

union ( select f1.flno from flights f1,flights f2 where f1.from_city="MADISON"

and f1.to_city!="NEW YORK" and f1.to_city=f2.from_city and f2.to_city="NEW YORK"

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 e_name from employee where e_id not in(select e_id from certified)

and salary>(select avg(salary) from employee where e_id in(select e_id

from certified));
```



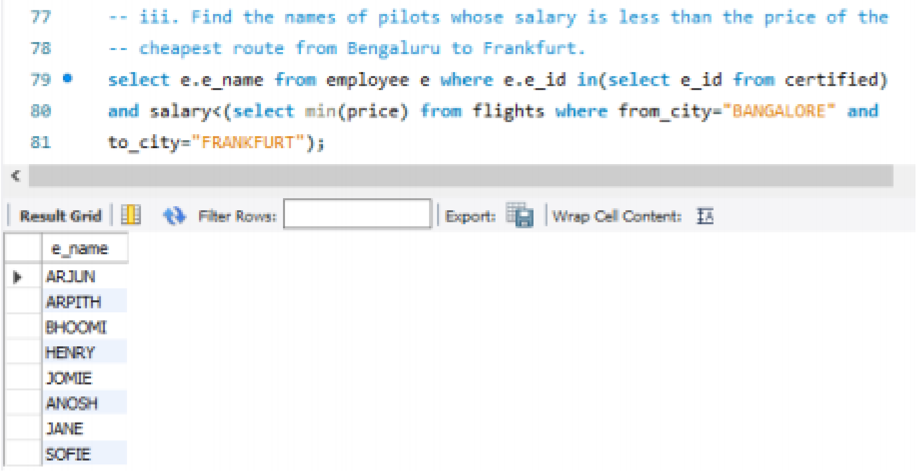
```
-- i. Find the names of aircraft such that all pilots certified to
-- operate them have salaries more than Rs.80,000.
select distinct a.a_name from aircraft a,certified c,employee e
where a.a_id=c.a_id and c.e_id=e.e_id and e.salary>80000;
```

```
a_name
FOKKER
JET01
VICKERS
BOEING
AIRBUS
```



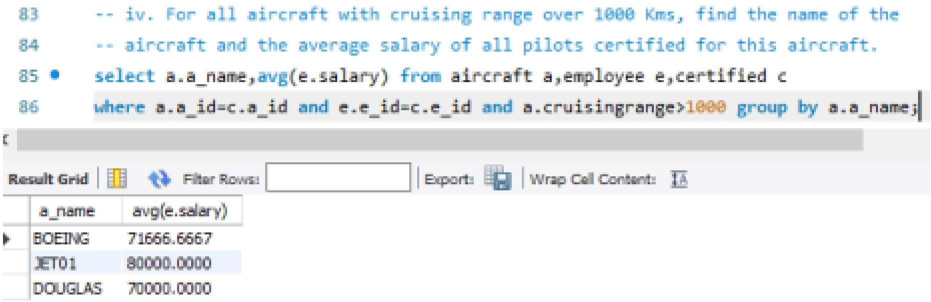
```
-- ii. For each pilot who is certified for more than three aircrafts, find the
--  eid and the maximum cruising range of the aircraft for which she or he is certified.
select e.e_id,max(a.cruisingrange) from aircraft a,employee e,certified c
where a.a_id=c.a_id and e.e_id=c.e_id group by e.e_id having count(e.e_id)>3;
```

```
e_id   max(a.cruisingrange)
9      5000
```



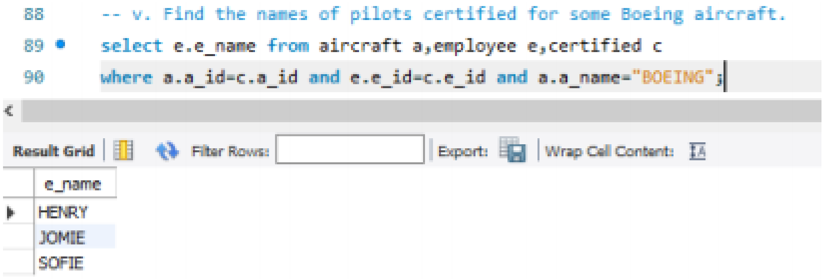
```sql
-- iii. Find the names of pilots whose salary is less than the price of the
-- cheapest route from Bengaluru to Frankfurt.
select e.e_name from employee e where e.e_id in(select e_id from certified)
and salary<(select min(price) from flights where from_city="BANGALORE" and
to_city="FRANKFURT");
```

e_name
ARJUN
ARPITH
BHOOMI
HENRY
JOMIE
ANOSH
JANE
SOFIE



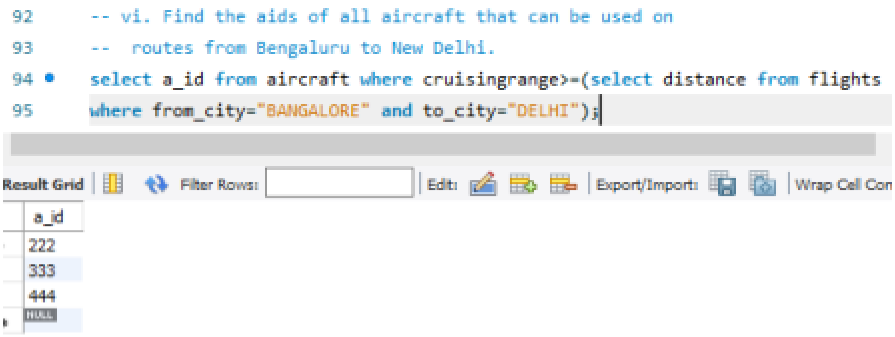
```sql
-- iv. For all aircraft with cruising range over 1000 Kms, find the name of the
-- aircraft and the average salary of all pilots certified for this aircraft.
select a.a_name,avg(e.salary) from aircraft a,employee e,certified c
where a.a_id=c.a_id and e.e_id=c.e_id and a.cruisingrange>1000 group by a.a_name;
```

a_name	avg(e.salary)
BOEING	71666.6667
JET01	80000.0000
DOUGLAS	70000.0000



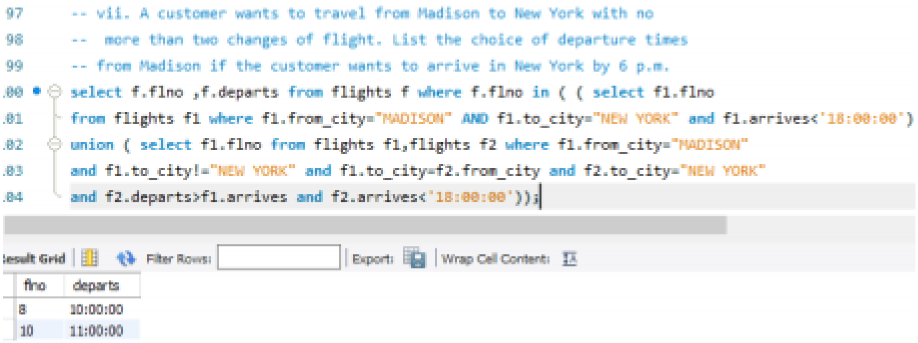
```sql
-- v. Find the names of pilots certified for some Boeing aircraft.
select e.e_name from aircraft a,employee e,certified c
where a.a_id=c.a_id and e.e_id=c.e_id and a.a_name="BOEING";
```

e_name
HENRY
JOMIE
SOFIE



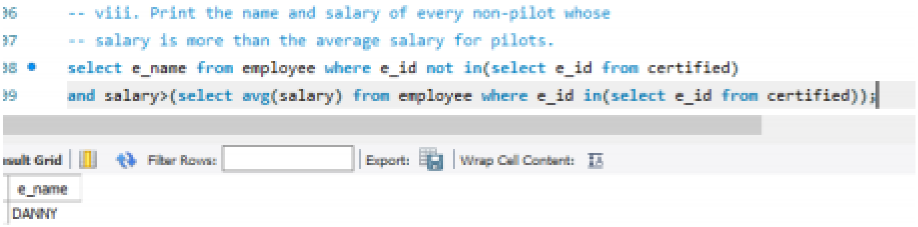
```
-- vi. Find the aids of all aircraft that can be used on
--  routes from Bengaluru to New Delhi.
select a_id from aircraft where cruisingrange>=(select distance from flights
where from_city="BANGALORE" and to_city="DELHI");
```

```
a_id
222
333
444
NULL
```



```
-- vii. A customer wants to travel from Madison to New York with no
--  more than two changes of flight. List the choice of departure times
-- from Madison if the customer wants to arrive in New York by 6 p.m.
select f.flno ,f.departs from flights f where f.flno in ( ( select f1.flno
from flights f1 where f1.from_city="MADISON" AND f1.to_city="NEW YORK" and f1.arrives<'18:00:00')
union ( select f1.flno from flights f1,flights f2 where f1.from_city="MADISON"
and f1.to_city!="NEW YORK" and f1.to_city=f2.from_city and f2.to_city="NEW YORK"
and f2.departs>f1.arrives and f2.arrives<'18:00:00'));
```

```
flno  departs
8     10:00:00
10    11:00:00
```



```
-- viii. Print the name and salary of every non-pilot whose
-- salary is more than the average salary for pilots.
select e_name from employee where e_id not in(select e_id from certified)
and salary>(select avg(salary) from employee where e_id in(select e_id from certified));
```

```
e_name
DANNY
```

Program 8 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) PRIMARY KEY,

SNAME VARCHAR(25),

ADDRESS VARCHAR(25),

PHONE int,

GENDER CHAR(1));

CREATE TABLE SEMSEC(

SSID VARCHAR(5) PRIMARY KEY,

SEM integer(2),
SEC CHAR(1));
CREATE TABLE CLASS(
USN VARCHAR(10),
SSID VARCHAR(5),
PRIMARY KEY(USN,SSID),
FOREIGN KEY(USN) REFERENCES STUDENT(USN),
FOREIGN KEY(SSID) REFERENCES SEMSEC(SSID));
CREATE TABLE SUBJECT(
SUBCODE VARCHAR(8),
TITLE VARCHAR(20),
SEM int,
CREDITS integer(2),
PRIMARY KEY(SUBCODE));
CREATE TABLE IAMARKS(
USN VARCHAR(10),
SUBCODE VARCHAR(8), SSID VARCHAR(5),
TEST1 integer(2),
TEST2 integer(2),
TEST3 integer(2),
FINALIA integer(2),
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 (‘1RN13CS020’,’AKSHAY’,’BELAGAVI’,
8877881,’M’);

INSERT INTO STUDENT VALUES
(‘1RN13CS062’,’SANDHYA’,’BENGALURU’,7722829,’F’);

INSERT INTO STUDENT VALUES (‘1RN13CS091’,’TEESHA’,’BENGALURU’,
7712312,’F’);

INSERT INTO STUDENT VALUES
(‘1RN13CS066’,’SUPRIYA’,’MANGALURU’,8877882,’F’);

INSERT INTO STUDENT VALUES (‘1RN14CS010’,’ABHAY’,’BENGALURU’,
9900211,’M’);

INSERT INTO STUDENT VALUES
(‘1RN14CS032’,’BHASKAR’,’BENGALURU’,9923219,’M’);

INSERT INTO STUDENT VALUES (‘1RN14CS025’,’ASMI’,’BENGALURU’,
7894737,’F’);

INSERT INTO STUDENT VALUES (‘1RN15CS011’,’AJAY’,’TUMKUR’,
9845091,’M’);

INSERT INTO STUDENT VALUES (‘1RN15CS029’,’CHITRA’,’DAVANGERE’,
7696772,’F’);

INSERT INTO STUDENT VALUES (‘1RN15CS045’,’JEEVA’,’BELLARY’,
9944850,’M’);

INSERT INTO STUDENT VALUES
(‘1RN15CS091’,’SANTOSH’,’MANGALURU’,8812332,’M’);

INSERT INTO STUDENT VALUES (‘1RN16CS045’,’ISMAIL’,’KALBURGI’,
9900232,’M’);

INSERT INTO STUDENT VALUES (‘1RN16CS088’,’SAMEERA’,’SHIMOGA’,
9905542,’F’);

INSERT INTO STUDENT VALUES
(‘1RN16CS122’,’VINAYAKA’,’CHIKAMAGALUR’, 8800880,’M’);

INSERT INTO SEMSEC VALUES (‘CSE8A’, 8,’A’);

INSERT INTO SEMSEC VALUES (‘CSE8B’, 8,’B’);

INSERT INTO SEMSEC VALUES (‘CSE8C’, 8,’C’);
INSERT INTO SEMSEC VALUES (‘CSE7A’, 7,’A’);
INSERT INTO SEMSEC VALUES (‘CSE7B’, 7,’B’); INSERT INTO SEMSEC
VALUES (‘CSE7C’, 7,’C’);
INSERT INTO SEMSEC VALUES (‘CSE6A’, 6,’A’);
INSERT INTO SEMSEC VALUES (‘CSE6B’, 6,’B’);
INSERT INTO SEMSEC VALUES (‘CSE6C’, 6,’C’);
INSERT INTO SEMSEC VALUES (‘CSE5A’, 5,’A’);
INSERT INTO SEMSEC VALUES (‘CSE5B’, 5,’B’); INSERT
INTO SEMSEC VALUES (‘CSE5C’, 5,’C’);
INSERT INTO SEMSEC VALUES (‘CSE4A’, 4,’A’);
INSERT INTO SEMSEC VALUES (‘CSE4B’, 4,’B’); INSERT
INTO SEMSEC VALUES (‘CSE4C’, 4,’C’);
INSERT INTO SEMSEC VALUES (‘CSE3A’, 3,’A’);
INSERT INTO SEMSEC VALUES (‘CSE3B’, 3,’B’);
INSERT INTO SEMSEC VALUES (‘CSE3C’, 3,’C’);
INSERT INTO SEMSEC VALUES (‘CSE2A’, 2,’A’);
INSERT INTO SEMSEC VALUES (‘CSE2B’, 2,’B’);
INSERT INTO SEMSEC VALUES (‘CSE2C’, 2,’C’);
INSERT INTO SEMSEC VALUES (‘CSE1A’, 1,’A’);
INSERT INTO SEMSEC VALUES (‘CSE1B’, 1,’B’);
INSERT INTO SEMSEC VALUES (‘CSE1C’, 1,’C’);
INSERT INTO CLASS VALUES (‘1RN13CS020’,’CSE8A’);
INSERT INTO CLASS VALUES (‘1RN13CS062’,’CSE8A’); INSERT
INTO CLASS VALUES (‘1RN13CS066’,’CSE8B’); INSERT INTO
CLASS VALUES (‘1RN13CS091’,’CSE8C’); INSERT INTO CLASS
VALUES (‘1RN14CS010’,’CSE7A’);
```

```
INSERT INTO CLASS VALUES (‘1RN14CS025’,’CSE7A’);

INSERT INTO CLASS VALUES (‘1RN14CS032’,’CSE7A’);

INSERT INTO CLASS VALUES (‘1RN15CS011’,’CSE4A’);

INSERT INTO CLASS VALUES (‘1RN15CS029’,’CSE4A’); INSERT INTO
CLASS VALUES (‘1RN15CS045’,’CSE4B’); INSERT INTO CLASS VALUES

(‘1RN15CS091’,’CSE4C’); INSERT INTO CLASS VALUES
(‘1RN16CS045’,’CSE3A’); INSERT INTO

CLASS VALUES (‘1RN16CS088’,’CSE3B’); INSERT INTO CLASS VALUES
(‘1RN16CS122’,’CSE3C’);

INSERT INTO SUBJECT VALUES (‘10CS81’,’ACA’, 8, 4);

INSERT INTO SUBJECT VALUES (‘10CS82’,’SSM’, 8, 4);

INSERT INTO SUBJECT VALUES (‘10CS83’,’NM’, 8, 4);

INSERT INTO SUBJECT VALUES (‘10CS84’,’CC’, 8, 4);

INSERT INTO SUBJECT VALUES (‘10CS85’,’PW’, 8, 4);

INSERT INTO SUBJECT VALUES (‘10CS71’,’OOAD’, 7, 4);

INSERT INTO SUBJECT VALUES (‘10CS72’,’ECS’, 7, 4);

INSERT INTO SUBJECT VALUES (‘10CS73’,’PTW’, 7, 4);

INSERT INTO SUBJECT VALUES (‘10CS74’,’DWDM’, 7, 4);

INSERT INTO SUBJECT VALUES (‘10CS75’,’JAVA’, 7, 4);

INSERT INTO SUBJECT VALUES (‘10CS76’,’SAN’, 7, 4);

INSERT INTO SUBJECT VALUES (‘15CS51’,’ME’, 5, 4);

INSERT INTO SUBJECT VALUES (‘15CS52’,’CN’, 5, 4);

INSERT INTO SUBJECT VALUES (‘15CS53’,’DBMS’, 5, 4);

INSERT INTO SUBJECT VALUES (‘15CS54’,’ATC’, 5, 4);

INSERT INTO SUBJECT VALUES (‘15CS55’,’JAVA’, 5, 3);

INSERT INTO SUBJECT VALUES (‘15CS56’,’AI’, 5, 3);
INSERT INTO SUBJECT VALUES (‘15CS41’,’M4’, 4, 4);

INSERT INTO SUBJECT VALUES (‘15CS42’,’SE’, 4, 4);

INSERT INTO SUBJECT VALUES (‘15CS43’,’DAA’, 4, 4);

INSERT INTO SUBJECT VALUES (‘15CS44’,’MPMC’, 4, 4);

INSERT INTO SUBJECT VALUES (‘15CS45’,’OOC’, 4, 3);

INSERT INTO SUBJECT VALUES (‘15CS46’,’DC’, 4, 3);

INSERT INTO SUBJECT VALUES (‘15CS31’,’M3’, 3, 4);

INSERT INTO SUBJECT VALUES (‘15CS32’,’ADE’, 3, 4); INSERT INTO
SUBJECT VALUES

(‘15CS33’,’DSA’, 3, 4); INSERT INTO SUBJECT VALUES (‘15CS34’,’CO’, 3,
4); INSERT INTO SUBJECT VALUES (‘15CS35’,’USP’, 3, 3);

INSERT INTO SUBJECT VALUES (‘15CS36’,’DMS’, 3, 3);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3)
VALUES

(‘1RN13CS091’,’10CS81’,’CSE8C’, 15, 16, 18);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3)
VALUES

(‘1RN13CS091’,’10CS82’,’CSE8C’, 12, 19, 14);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3)
VALUES

(‘1RN13CS091’,’10CS83’,’CSE8C’, 19, 15, 20);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3)
VALUES

(‘1RN13CS091’,’10CS84’,’CSE8C’, 20, 16, 19);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3)
VALUES

(‘1RN13CS091’,’10CS85’,’CSE8C’, 15, 15, 12);

-- 3
```

```
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’;

-- 4)

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;

-- 5)

CREATE VIEW STU_TEST1_MARKS_VIEW

AS

SELECT TEST1, SUBCODE

FROM IAMARKS

WHERE USN = '1RN13CS091';

-- 6)

update IAMARKS set
FINALIA=((TEST1+TEST2+TEST3)-LEAST(TEST1,TEST2,TEST3))/2;

-- 7)

SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER,

(CASE

WHEN IA.FINALIA BETWEEN 17 AND 20 THEN ‘OUTSTANDING’

WHEN IA.FINALIA BETWEEN 12 AND 16 THEN ‘AVERAGE’
```

ELSE ‘WEAK’

END) AS CAT

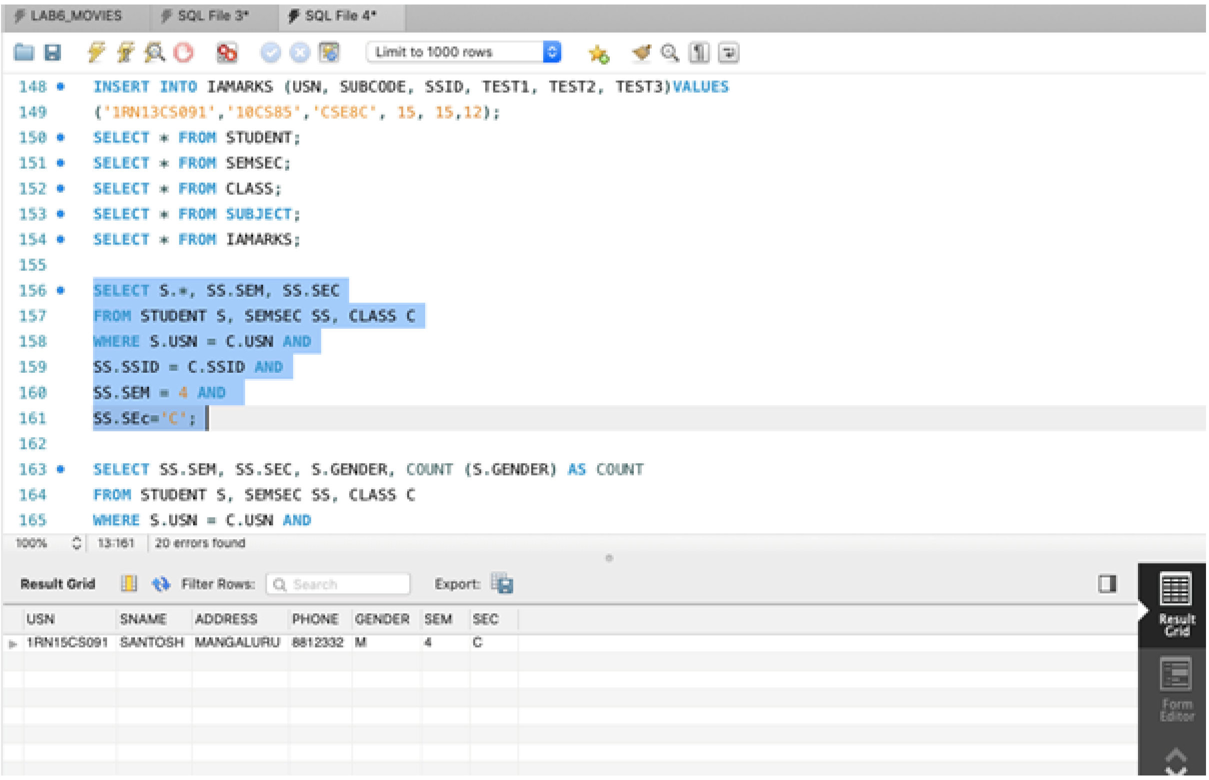
FROM STUDENT S, SEMSEC SS, IAMARKS IA, SUBJECT SUB

WHERE S.USN = IA.USN AND

SS.SSID = IA.SSID AND

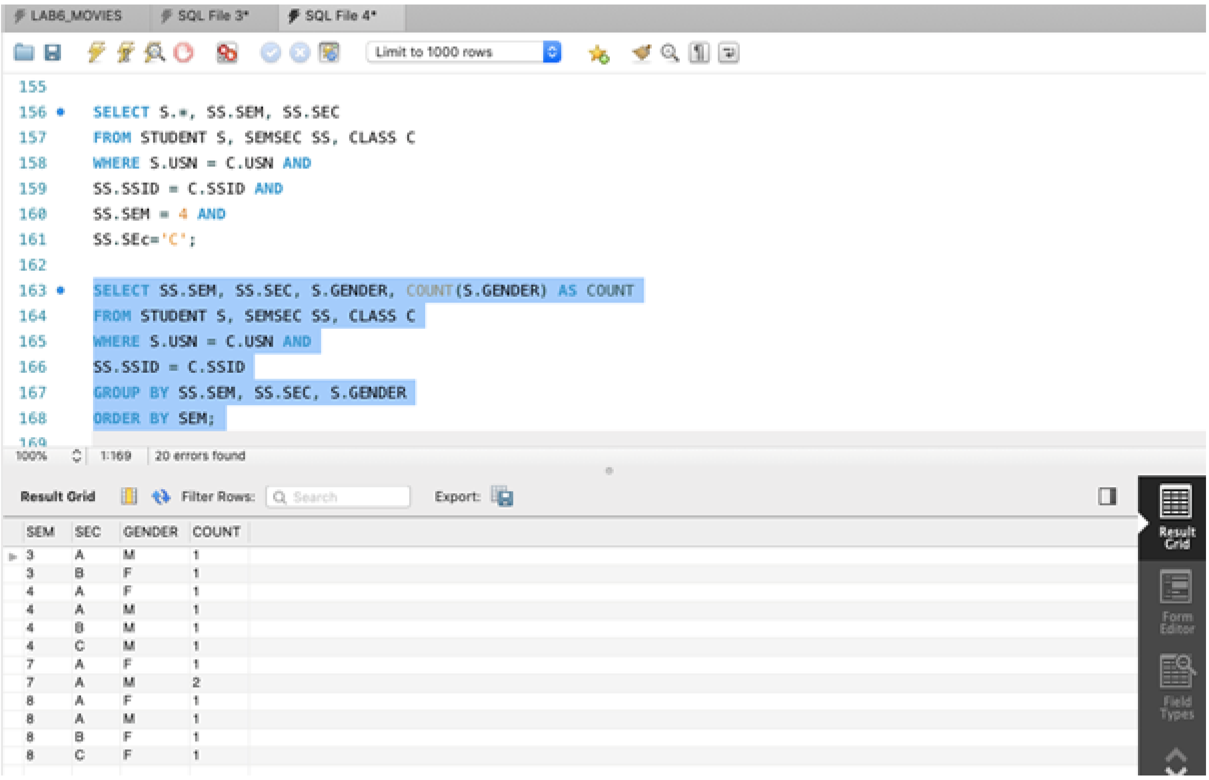
SUB.SUBCODE = IA.SUBCODE AND

SUB.SEM = 8;



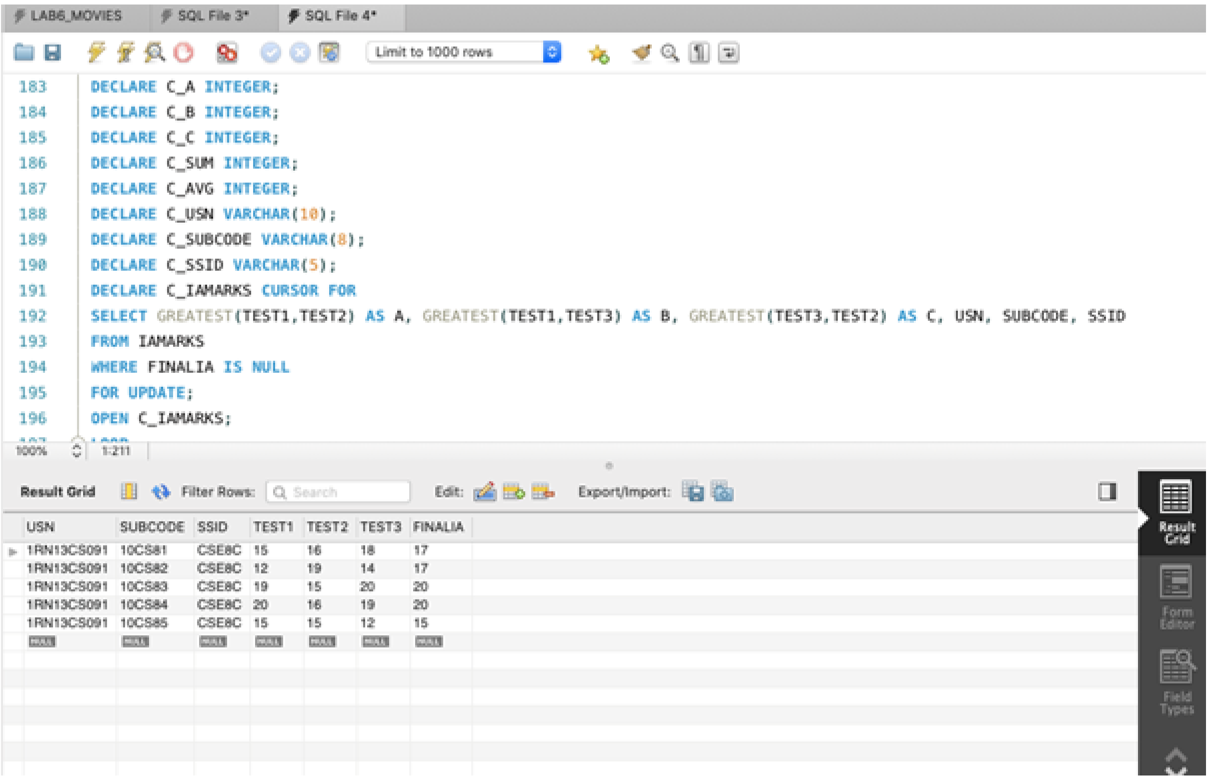
```
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3)VALUES
('1RN13CS091','10CS85','CSE8C', 15, 15,12);
SELECT * FROM STUDENT;
SELECT * FROM SEMSEC;
SELECT * FROM CLASS;
SELECT * FROM SUBJECT;
SELECT * FROM IAMARKS;

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';

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
```

USN	SNAME	ADDRESS	PHONE	GENDER	SEM	SEC
1RN15CS091	SANTOSH	MANGALURU	8812332	M	4	C



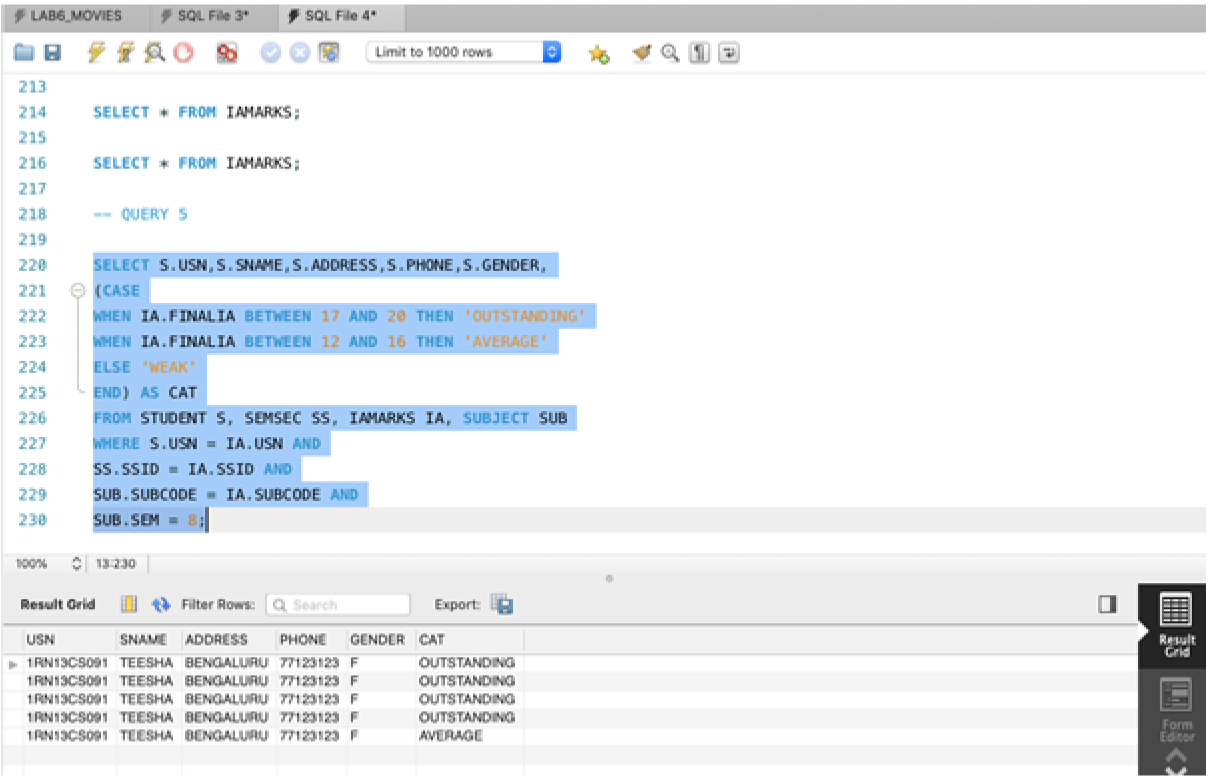
```sql
SELECT S.*, SS.SEM, SS.SEC
FROM STUDENT S, SEMSEC SS, CLASS C
WHERE S.USN = C.USN AND
SS.SSID = C.SSID AND
SS.SEM = 4 AND
SS.SEc='C';

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;
```

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



```sql
DECLARE C_A INTEGER;
DECLARE C_B INTEGER;
DECLARE C_C INTEGER;
DECLARE C_SUM INTEGER;
DECLARE C_AVG INTEGER;
DECLARE C_USN VARCHAR(10);
DECLARE C_SUBCODE VARCHAR(8);
DECLARE C_SSID VARCHAR(5);
DECLARE C_IAMARKS CURSOR FOR
SELECT GREATEST(TEST1,TEST2) AS A, GREATEST(TEST1,TEST3) AS B, GREATEST(TEST3,TEST2) AS C, USN, SUBCODE, SSID
FROM IAMARKS
WHERE FINALIA IS NULL
FOR UPDATE;
OPEN C_IAMARKS;
```

USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINALIA
1RN13CS091	10CS81	CSE8C	15	16	18	17
1RN13CS091	10CS82	CSE8C	12	19	14	17
1RN13CS091	10CS83	CSE8C	19	15	20	20
1RN13CS091	10CS84	CSE8C	20	16	19	20
1RN13CS091	10CS85	CSE8C	15	15	12	15
NULL	NULL	NULL	NULL	NULL	NULL	NULL



```sql
SELECT * FROM IAMARKS;

SELECT * FROM IAMARKS;

-- QUERY 5

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

```
USN         SNAME   ADDRESS    PHONE     GENDER  CAT
1RN13CS091  TEESHA  BENGALURU  77123123  F       OUTSTANDING
1RN13CS091  TEESHA  BENGALURU  77123123  F       OUTSTANDING
1RN13CS091  TEESHA  BENGALURU  77123123  F       OUTSTANDING
1RN13CS091  TEESHA  BENGALURU  77123123  F       OUTSTANDING
1RN13CS091  TEESHA  BENGALURU  77123123  F       AVERAGE
```

Program 9 Student Faculty:

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 p 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).

```
CREATE DATABASE Student_Faculty;

use  Student_Faculty;


 CREATE TABLE student(

snum INT,
sname VARCHAR(10),
major VARCHAR(2),
level 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),
meets_at VARCHAR(10),
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 (snum,sname,major,level,age)
 VALUES(1,'jhon','CS','Sr',19),
(2,'smith','CS','Jr',20),
(3,'jacob','CV','Sr',20),
(4,'tom','CS','Jr',20),
(5,'sid','CS','Jr',20),
(6,'harry','CS','Sr',21);

SELECT * FROM student;

INSERT INTO faculty (fid,fname, deptid)
 VALUES(11,'Harshith',1000),
(12,'Mohan',1000),
(13,'Kumar',1001),
(14,'Shobha',1002),
(15,'Shan',1000);
```

```
SELECT * FROM faculty;

INSERT INTO class (cname,meets_at,room,fid)
VALUES('class1','noon','room1',14),
('class10','morning','room128',14),
('class2','morning','room2',12),
('class3','morning','room3',11),
('class4','evening','room4',14),
('class5','night','room3',15),
('class6','morning','room2',14),
('class7','morning','room3',14);

INSERT INTO enrolled (snum,cname)
VALUES(1,'class1'),
(2,'class1'),
(4,'class3'),
(3,'class3'),
(5,'class4'),
(1,'class5'),
(2,'class5'),
(3,'class5'),
(4,'class5'),
(5,'class5'),
(6,'class5');
```

```
SELECT * FROM enrolled;
```

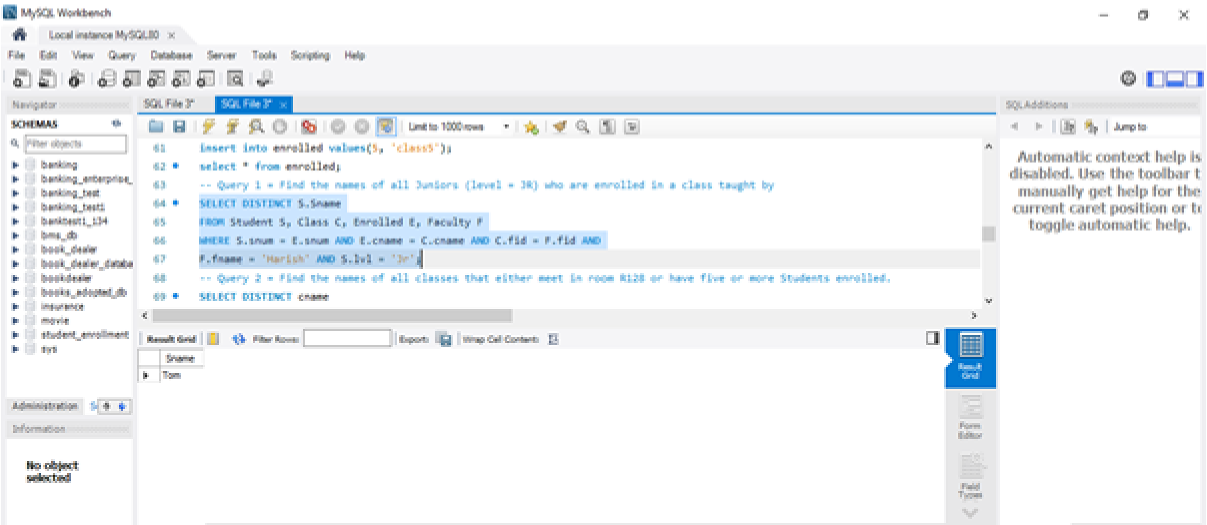
```
-- Query 1: Find the names of all juniors (level=Jr) who are enrolled for
class taught by professor Harshith.

SELECT DISTINCT s.sname

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

WHERE  s.snum=e.snum        AND

        e.cname=c.cname        AND

        s.level='jr'   AND

        f.fname='Harshith' AND

        f.fid=c.fid;
```

```
-- Query 2: Find the names of all classes that either meet in room128 or
have 5 or more students enrolled.

SELECT DISTINCT cname

FROM class

WHERE room='room128'

OR

cname IN (SELECT e.cname FROM enrolled e GROUP BY e.cname HAVING
COUNT(*)>=5);
```

```
-- Query 3: Find the names of all students who are enrolled in two classes
that meet at 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.meets_at=c2.meets_at  );


-- Query 4: 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) );




-- Query 5: Find the names of the faculty members for whome the combined enrollment of the classes that they teach is less then five.

 SELECT DISTINCT f.fname

FROM faculty f

WHERE f.fid IN (  SELECT c.fid

 FROM class c, enrolled e
```

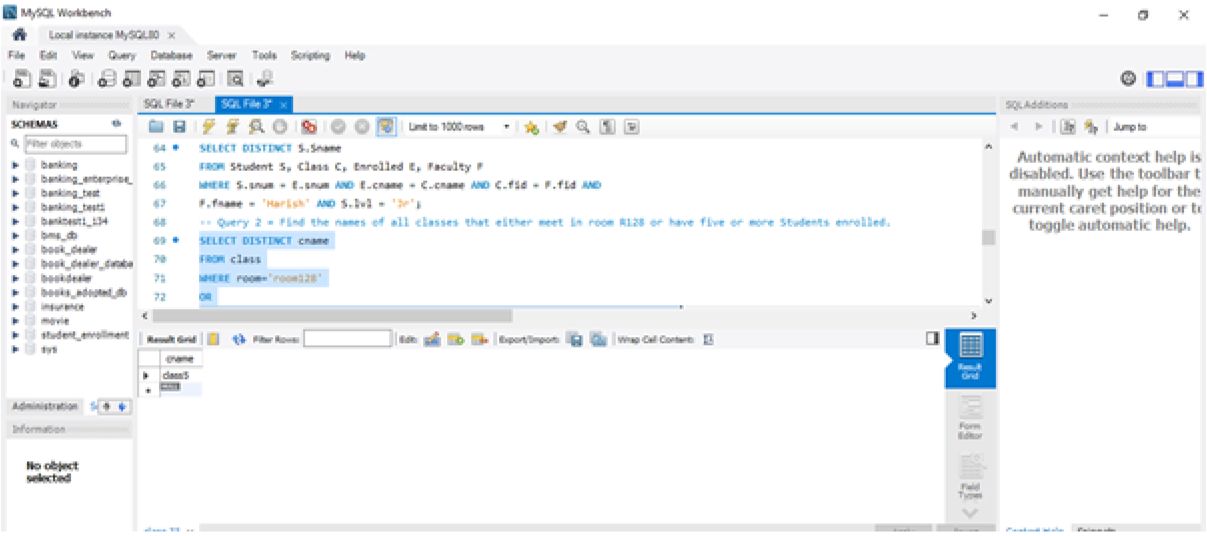
WHERE c.cname = e.cname GROUP BY c.cname HAVING COUNT(c.cname)< 5);



```
insert into enrolled values(5, 'class5');
select * from enrolled;
-- Query 1 - 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';
-- Query 2 - Find the names of all classes that either meet in room R128 or have five or more Students enrolled.
SELECT DISTINCT cname
```

Sname
Tom

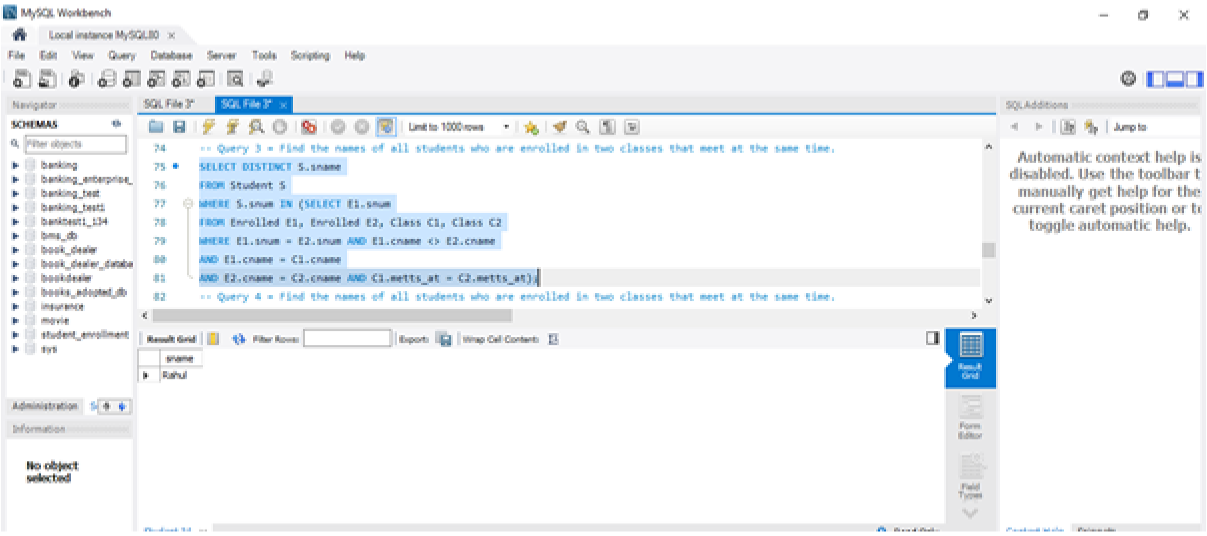
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.



```
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';
-- Query 2 - 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='room128'
OR
```

cname
class5
NULL

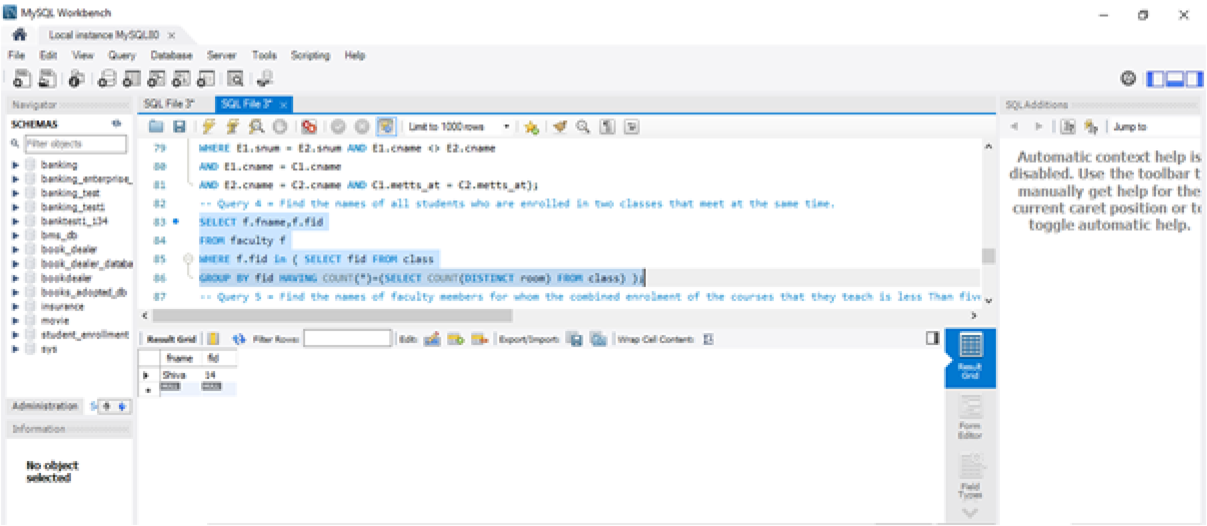
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.



```
-- Query 3 - 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);
-- Query 4 - Find the names of all students who are enrolled in two classes that meet at the same time.
```

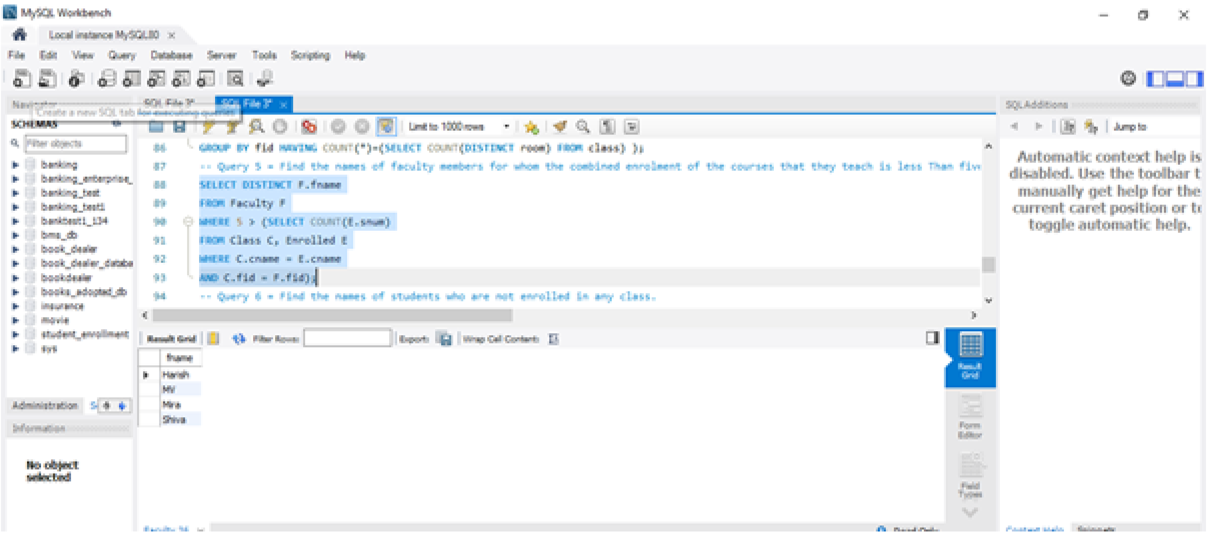
sname
Rahul

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.



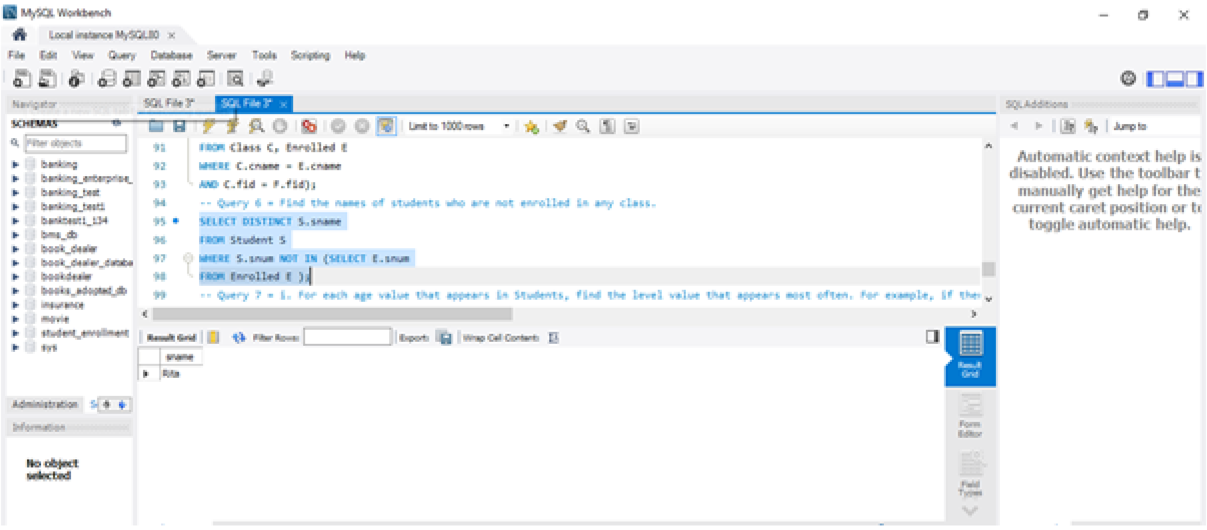
```sql
WHERE E1.snum = E2.snum AND E1.cname <> E2.cname
AND E1.cname = C1.cname
AND E2.cname = C2.cname AND C1.metts_at = C2.metts_at);
-- Query 4 = Find the names of all students who are enrolled in two classes that meet at the same time.
SELECT f.fname,f.fid
FROM faculty f
WHERE f.fid in ( SELECT fid FROM class
GROUP BY fid HAVING COUNT(*)=(SELECT COUNT(DISTINCT room) FROM class) );
-- Query 5 = Find the names of faculty members for whom the combined enrolment of the courses that they teach is less Than fiv
```

```
fname   fid
Shiva   14
NULL    NULL
```



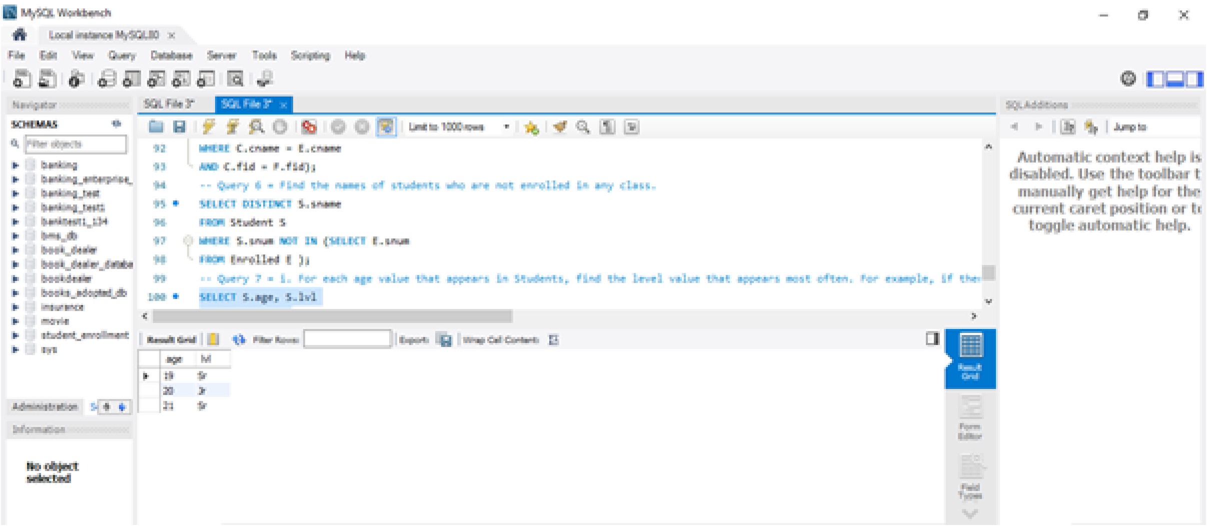
```sql
GROUP BY fid HAVING COUNT(*)=(SELECT COUNT(DISTINCT room) FROM class) );
-- Query 5 = Find the names of faculty members for whom the combined enrolment of the courses that they teach is less Than fiv
SELECT DISTINCT F.fname
FROM Faculty F
WHERE 5 > (SELECT COUNT(E.snum)
FROM Class C, Enrolled E
WHERE C.cname = E.cname
AND C.fid = F.fid);
-- Query 6 = Find the names of students who are not enrolled in any class.
```

```
fname
Harish
MV
Mira
Shiva
```



```sql
FROM Class C, Enrolled E
WHERE C.cname = E.cname
AND C.fid = F.fid);
-- Query 6 = 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 );
-- Query 7 = 1. For each age value that appears in Students, find the level value that appears most often. For example, if the
```

```
sname
Rita
```



```
WHERE C.cname = E.cname
AND C.fid = F.fid);
-- Query 6 - 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 );
-- Query 7 - i. For each age value that appears in Students, find the level value that appears most often. For example, if the
SELECT S.age, S.lvl
```

```
age  lvl
19   Sr
20   Jr
21   Sr
```

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.

Program 10 Supllier database:

Consider the following schema:

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

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

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

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

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

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

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

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

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

all the suppliers who supply that part).

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

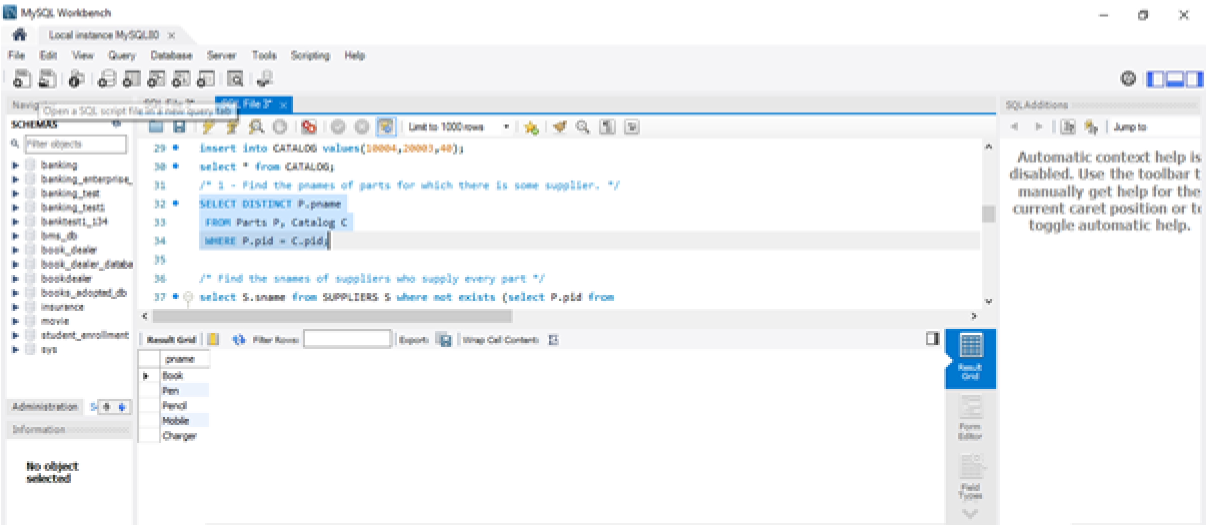
vii. Find the sids of suppliers who supply only red parts.

```
CREATE DATABASE SUPPLIER;
USE SUPPLIER;
CREATE TABLE SUPPLIERS(SID BIGINT(5) PRIMARY KEY, SNAME
VARCHAR(20), CITY VARCHAR(20));
INSERT INTO SUPPLIERS VALUES(10001,'ACME WIDGET','BANGALORE');
INSERT INTO SUPPLIERS VALUES(10002,'JOHNS ','KOLKATA');
INSERT INTO SUPPLIERS VALUES(10003,'VIMAL','MUMBAI');
INSERT INTO SUPPLIERS VALUES(10004,'RELIANCE ','DELHI');
SELECT * FROM SUPPLIERS;
CREATE TABLE PARTS(PID BIGINT(5) PRIMARY KEY, PNAME VARCHAR(20),
COLOR VARCHAR(10));
INSERT INTO PARTS VALUES(20001,'BOOK','RED');
INSERT INTO PARTS VALUES(20002,'PEN','RED');
INSERT INTO PARTS VALUES(20003,'PENCIL','GREEN');
INSERT INTO PARTS VALUES(20004,'MOBILE ','GREEN');
INSERT INTO PARTS VALUES(20005,'CHARGER','BLACK');
SELECT * FROM PARTS;
CREATE TABLE CATALOG(SID BIGINT(5), PID BIGINT(5), FOREIGN KEY(SID)
REFERENCES SUPPLIERS(SID), FOREIGN KEY(PID) REFERENCES PARTS(PID),
COST FLOAT(6), PRIMARY KEY(SID, PID));
INSERT INTO CATALOG VALUES(10001,20001,10);
INSERT INTO CATALOG VALUES(10001,20002,10);
INSERT INTO CATALOG VALUES(10001,20003,30);
INSERT INTO CATALOG VALUES(10001,20004,10);
INSERT INTO CATALOG VALUES(10001,20005,10);
INSERT INTO CATALOG VALUES(10002,20001,10);
INSERT INTO CATALOG VALUES(10002,20002,20);
INSERT INTO CATALOG VALUES(10003,20003,30);
INSERT INTO CATALOG VALUES(10004,20003,40);
SELECT * FROM CATALOG;
/* 1 - FIND THE PNAMES OF PARTS FOR WHICH THERE IS SOME SUPPLIER. */
SELECT DISTINCT P.PNAME
 FROM PARTS P, CATALOG C
 WHERE P.PID = C.PID;

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

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

/* FIND THE PNAMES OF PARTS SUPPLIED BY ACME WIDGET SUPPLIERS AND
BY NO ONE ELSE */
SELECT P.PNAME FROM PARTS P, CATALOG C, SUPPLIERS S WHERE P.PID
= C.PID AND C.SID = S.SID AND S.SNAME = 'ACME WIDGET' AND NOT EXISTS
(SELECT * FROM CATALOG C1, SUPPLIERS S1 WHERE P.PID = C1.PID AND
C1.SID = S1.SID AND S1.SNAME <> 'ACME WIDGET');
```

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

ALL THE SUPPLIERS WHO SUPPLY THAT PART).

 */

SELECT DISTINCT C.SID FROM CATALOG C

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

FROM CATALOG C1

WHERE C1.PID = C.PID );
```

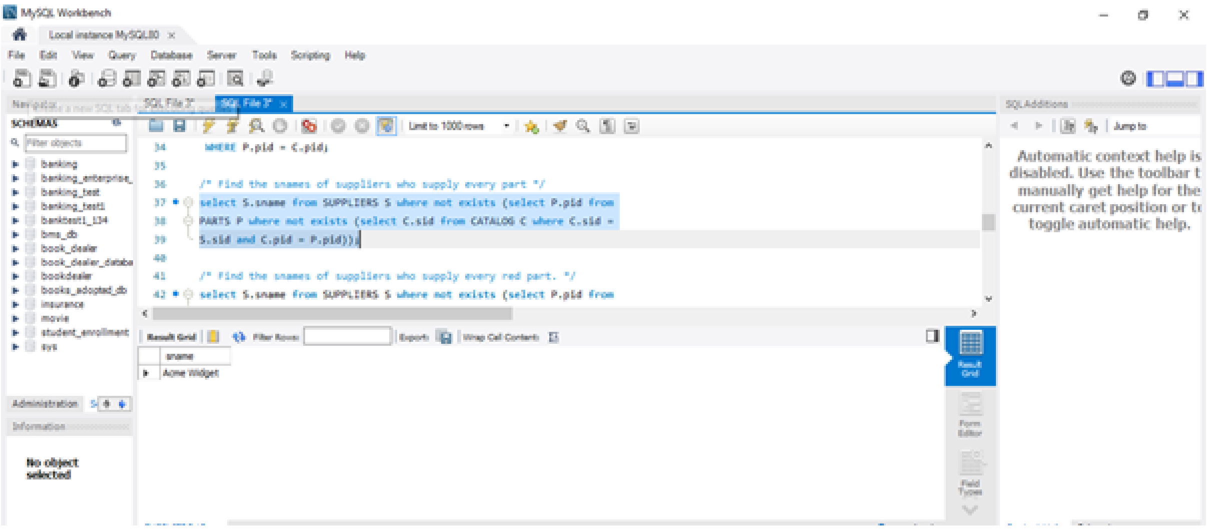
```
/* FOR EACH PART, FIND THE SNAME OF THE SUPPLIER WHO CHARGES THE
MOST FOR THAT PART.*/

SELECT P.PID, S.SNAME

FROM PARTS P, SUPPLIERS S, CATALOG C

WHERE C.PID = P.PID

AND C.SID = S.SID

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

FROM CATALOG C1

WHERE C1.PID = P.PID);
```

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

SELECT DISTINCT C.SID

FROM CATALOG C

WHERE NOT EXISTS ( SELECT *

FROM PARTS P

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



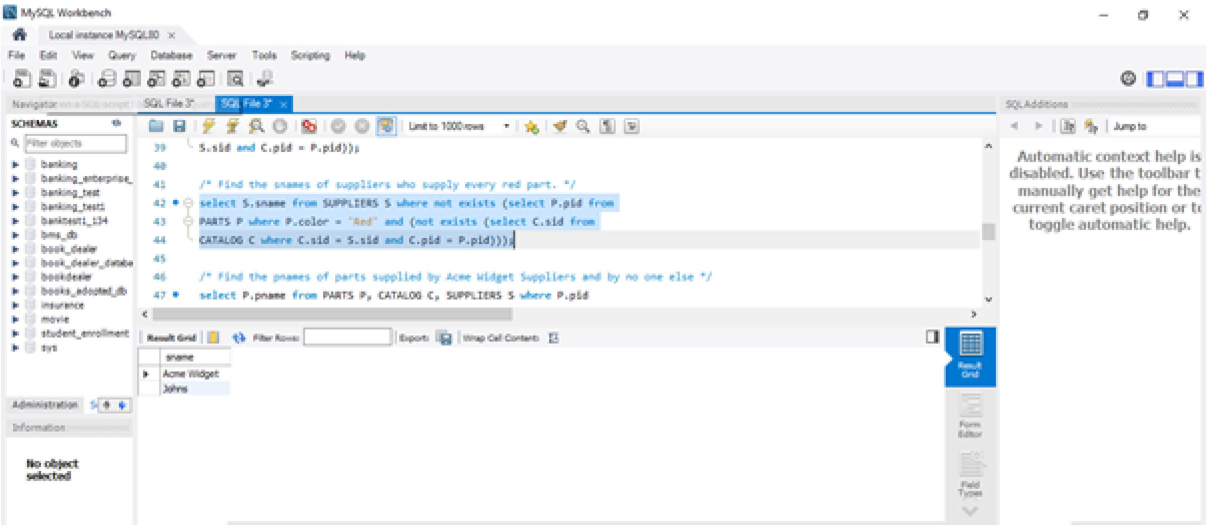
```
insert into CATALOG values(10004,20003,40);
select * from CATALOG;
/* 1 - Find the pnames of parts for which there is some supplier. */
SELECT DISTINCT P.pname
 FROM Parts P, Catalog C
 WHERE P.pid = C.pid;

/* Find the snames of suppliers who supply every part */
select S.sname from SUPPLIERS S where not exists (select P.pid from
```

```
pname
Book
Pen
Pencil
Mobile
Charger
```



```
 WHERE P.pid = C.pid;

/* Find the snames of suppliers who supply every part */
select S.sname from SUPPLIERS S where not exists (select P.pid from
PARTS P where not exists (select C.sid from CATALOG C where C.sid =
S.sid and C.pid = P.pid));

/* Find the snames of suppliers who supply every red part. */
select S.sname from SUPPLIERS S where not exists (select P.pid from
```

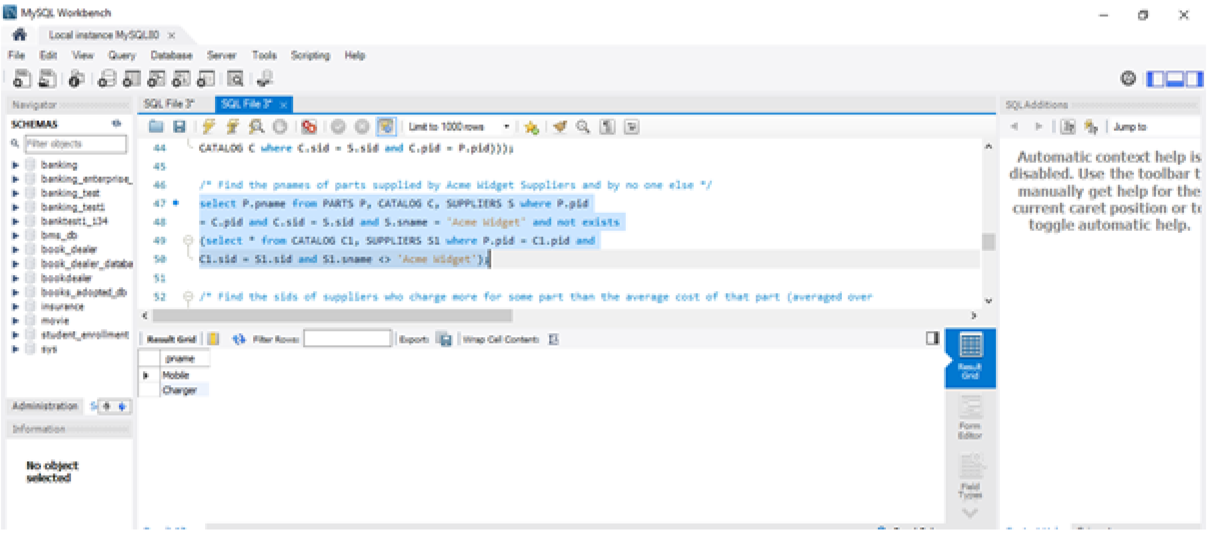
```
sname
Acme Widget
```



```
S.sid and C.pid = P.pid));

/* Find the snames of suppliers who supply every red part. */
select S.sname from SUPPLIERS S where not exists (select P.pid from
PARTS P where P.color = 'Red' and (not exists (select C.sid from
CATALOG C where C.sid = S.sid and C.pid = P.pid)));

/* Find the pnames of parts supplied by Acme Widget Suppliers and by no one else */
select P.pname from PARTS P, CATALOG C, SUPPLIERS S where P.pid
```

sname
Acme Widget
Johns

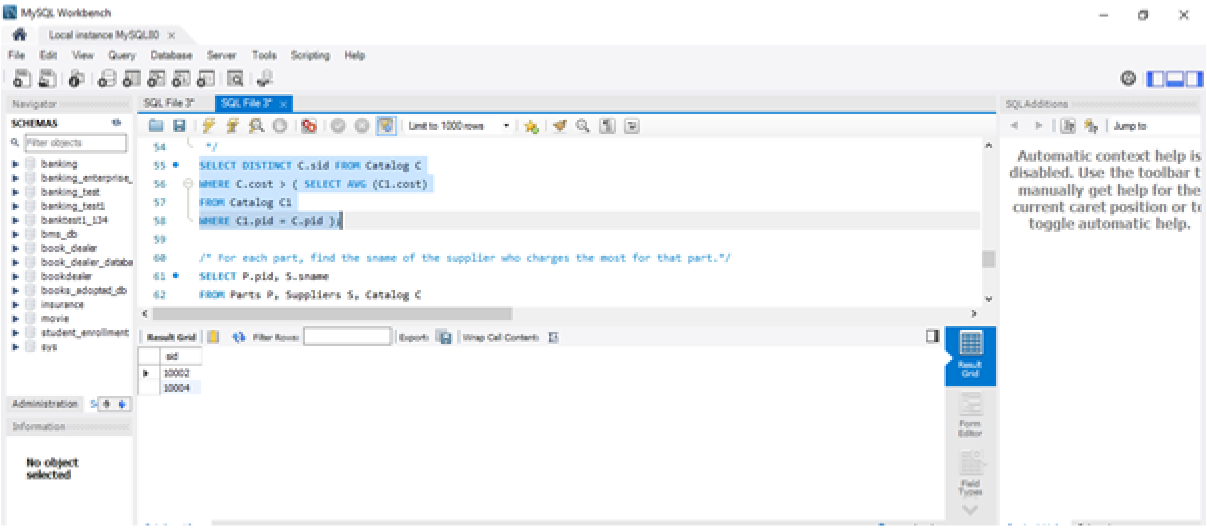
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.



```
CATALOG C where C.sid = S.sid and C.pid = P.pid)));

/* Find the pnames of parts supplied by Acme Widget Suppliers and by no one else */
select P.pname from PARTS P, CATALOG C, SUPPLIERS S where P.pid
= C.pid and C.sid = S.sid and S.sname = 'Acme Widget' and not exists
(select * from CATALOG C1, SUPPLIERS S1 where P.pid = C1.pid and
C1.sid = S1.sid and S1.sname <> 'Acme Widget');

/* Find the sids of suppliers who charge more for some part than the average cost of that part (averaged over
```

pname
Mobile
Charger

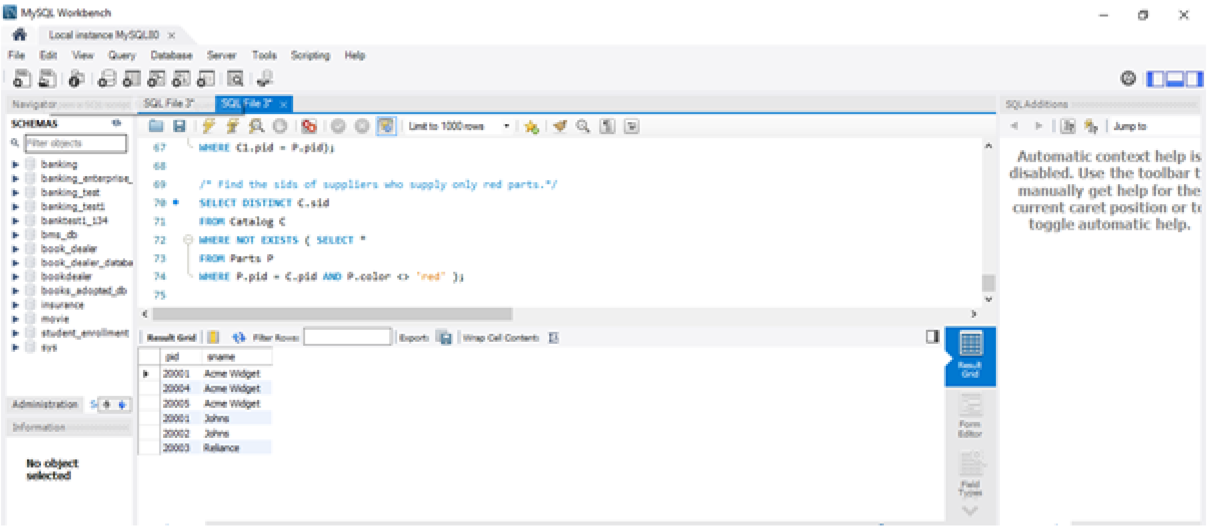
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.



```
 */
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
```

sid
10002
10004

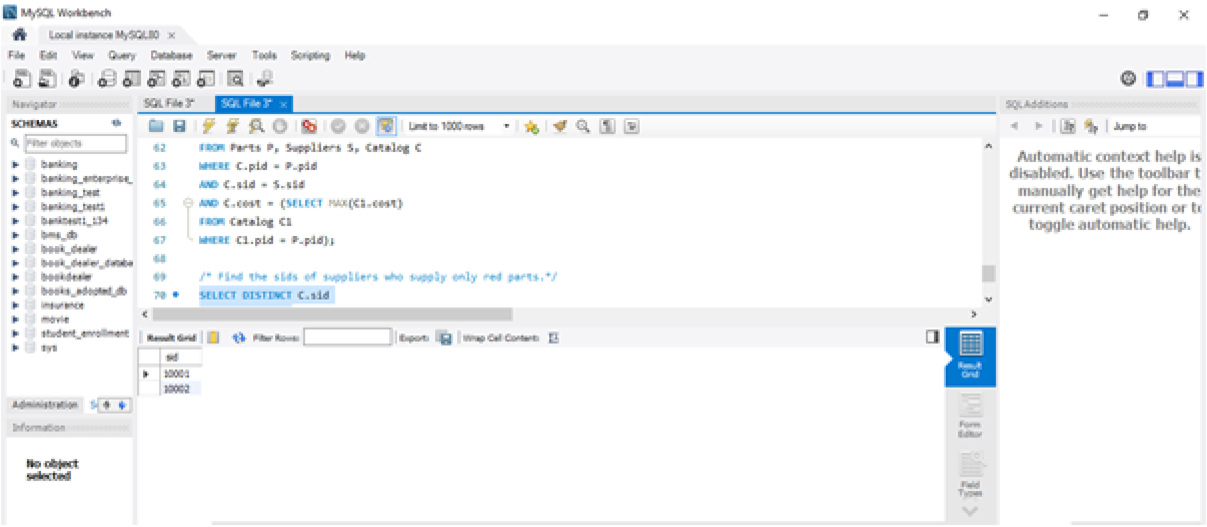
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or to toggle automatic help.



```
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' );

```

pid	sname
20001	Acme Widget
20004	Acme Widget
20005	Acme Widget
20001	Johns
20002	Johns
20003	Reliance

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or to toggle automatic help.



```
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
```

sid
10001
10002

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or to toggle automatic help.