B. M. S. COLLEGE OF ENGINEERING

Record of Practical Work
Subject: DataBase Management System (MySQL)

Name	**Kiran M K**
USN	**1BM19CS073**
Semester	**4**
Section	**B**
Dept.	**CSE**

CONTENTS

Experiment 1:

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, adate: 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.

Code:

```
create database insurance;
use insurance;
CREATE TABLE PERSON (DRIVER_ID VARCHAR(10),NAME
VARCHAR(20),ADDRESS
VARCHAR(15),PRIMARY KEY(DRIVER_ID));
create table car(regno varchar(10),Model varchar(20),Year date,Primary key(Regno));
create table Accident(report_no int,ADATE DATE,Location varchar(15),Primary
key(report_no));

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_no int,
damage_amt float, foreign key (driver_id,regno) references OWNS(driver_id,regno)
ON DELETE CASCADE,
foreign key (REPORT_NO) references ACCIDENT(REPORT_NO) ON DELETE
CASCADE);
```

```
INSERT INTO PERSON VALUES("1111","RAMU","K.S.LAYOUT");
INSERT INTO PERSON VALUES("2222","JOHN","INDIRANAGAR"),
("3333","PRIYA","JAYNAGAR"), ("4444","GOPAL","WHITEFIELD"),
("5555","LATHA","VIJAYNAGAR");

INSERT INTO CAR VALUES("KA04Q2301","MARUTHI-DX","2000-01-01"),
("KA05P1000","FORDICON","2000-01-01"),("KA03L1234","ZEN-
VXI","1999-01-01"),("KA03L9999","MARUTHI-DX","2002-01-01"),
("KA01P4020","INDICA-VX","2002-01-01");

INSERT INTO ACCIDENT VALUES(12,"2002-06-01","M G ROAD"),
(200,"2002-12-10","DOUBLE ROAD"),(300,"1999-07-23","M G ROAD"),
(25000,"2000-06-11","RESIDENCY ROAD"),(26500,"2001-10-16","RICHMOND
ROAD");

INSERT INTO OWNS VALUES("1111","KA04Q2301"),("1111","KA05P1000"),
("2222","KA03L1234"),("3333","KA03L9999"),("4444","KA01P4020");

INSERT INTO PARTICIPATED VALUES("1111","KA04Q2301",12,20000),
("2222","KA03L1234",200,500),("3333","KA03L9999",300,10000),
("4444","KA01P4020",25000,2375),("1111","KA05P1000",26500,70000);

UPDATE PARTICIPATED SET DAMAGE_AMT=25000 WHERE
REPORT_NO=12 AND REGNO="KA04Q2301";

SELECT * FROM PARTICIPATED;
SELECT COUNT(*) FROM ACCIDENT WHERE ADATE LIKE "2002-__-__";

SELECT COUNT(A.REPORT_NO) FROM ACCIDENT A,PARTICIPATED P,CAR
C WHERE A.REPORT_NO = P.REPORT_NO AND P.REGNO=C.REGNO AND
C.MODEL="MARUTHI-DX";
```

Output:

```
INSERT INTO PERSON VALUES("2222","JOHN","INDIRANAGAR"),("3333","PRIYA","JAYNAGAR"), ("4444","GOPAL","WHITEFIELD"),("55

INSERT INTO CAR VALUES("KA04Q2301","MARUTHI-DX","2000-01-01"),("KA05P1000","FORDICON","2000-01-01"),("KA03L1234","ZEN-

INSERT INTO ACCIDENT VALUES(12,"2002-06-01","M G ROAD"),(200,"2002-12-10","DOUBLE ROAD"),(300,"1999-07-23","M G ROAD")

INSERT INTO OWNS VALUES("1111","KA04Q2301"),("1111","KA05P1000"),("2222","KA03L1234"),("3333","KA03L9999"),("4444","KA

INSERT INTO PARTICIPATED VALUES("1111","KA04Q2301",12,20000),("2222","KA03L1234",200,500),("3333","KA03L9999",300,1000

UPDATE PARTICIPATED SET DAMAGE_AMT=25000 WHERE REPORT_NO=12 AND REGNO="KA04Q2301";

SELECT * FROM PARTICIPATED;
SELECT COUNT(*) FROM ACCIDENT WHERE ADATE LIKE "2002-__-__";

SELECT COUNT(A.REPORT_NO) FROM ACCIDENT A,PARTICIPATED P,CAR C WHERE A.REPORT_NO = P.REPORT_NO AND P.REGNO=C.REGNO AND
```

100% 28:32

Result Grid | Filter Rows: Search | Export:

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

Result Grid | Form Editor

```
foreign key (REPORT_NO) references ACCIDENT(REPORT_NO) ON DELETE CASCADE);


INSERT INTO PERSON VALUES("1111","RAMU","K.S.LAYOUT");
INSERT INTO PERSON VALUES("2222","JOHN","INDIRANAGAR"),("3333","PRIYA","JAYNAGAR"), ("4444","GOPAL","WHITEFIELD"),("55

INSERT INTO CAR VALUES("KA04Q2301","MARUTHI-DX","2000-01-01"),("KA05P1000","FORDICON","2000-01-01"),("KA03L1234","ZEN-

INSERT INTO ACCIDENT VALUES(12,"2002-06-01","M G ROAD"),(200,"2002-12-10","DOUBLE ROAD"),(300,"1999-07-23","M G ROAD")

INSERT INTO OWNS VALUES("1111","KA04Q2301"),("1111","KA05P1000"),("2222","KA03L1234"),("3333","KA03L9999"),("4444","KA

INSERT INTO PARTICIPATED VALUES("1111","KA04Q2301",12,20000),("2222","KA03L1234",200,500),("3333","KA03L9999",300,1000

UPDATE PARTICIPATED SET DAMAGE_AMT=25000 WHERE REPORT_NO=12 AND REGNO="KA04Q2301";

SELECT * FROM PARTICIPATED;
SELECT COUNT(*) FROM ACCIDENT WHERE ADATE LIKE "2002-__-__";
```

100% 61:33

Result Grid | Filter Rows: Search | Export:

COUNT(*)
2

Result Grid | Form Editor

```
UES("1111","RAMU","K.S.LAYOUT");
UES("2222","JOHN","INDIRANAGAR"),("3333","PRIYA","JAYNAGAR"), ("4444","GOPAL","WHITEFIELD"),("5555","LATHA","VIJAYNAGA

("KA04Q2301","MARUTHI-DX","2000-01-01"),("KA05P1000","FORDICON","2000-01-01"),("KA03L1234","ZEN-VXI","1999-01-01"),("K

ALUES(12,"2002-06-01","M G ROAD"),(200,"2002-12-10","DOUBLE ROAD"),(300,"1999-07-23","M G ROAD"),(25000,"2000-06-11","

S("1111","KA04Q2301"),("1111","KA05P1000"),("2222","KA03L1234"),("3333","KA03L9999"),("4444","KA01P4020");

ED VALUES("1111","KA04Q2301",12,20000),("2222","KA03L1234",200,500),("3333","KA03L9999",300,10000),("4444","KA01P4020"

T DAMAGE_AMT=25000 WHERE REPORT_NO=12 AND REGNO="KA04Q2301";

ATED;
CCIDENT WHERE ADATE LIKE "2002-__-__";

NO) FROM ACCIDENT A,PARTICIPATED P,CAR C WHERE A.REPORT_NO = P.REPORT_NO AND P.REGNO=C.REGNO AND C.MODEL="MARUTHI-DX";
```

100% 141:35

Result Grid Filter Rows: Search Export:

COUNT(A.REPORT_NO)
2

Result Grid

Form Editor

Experiment 2:

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%.

Code:

```
CREATE DATABASE BOOKDEALER;
USE BOOKDEALER;
CREATE TABLE AUTHOR(AUTHOR_ID INT,NAME VARCHAR(30),CITY
VARCHAR(30),COUNTRY VARCHAR(30));
CREATE TABLE PUBLISHER(PUBLISHER_ID INT,NAME VARCHAR(30),CITY
VARCHAR(30),COUNTRY VARCHAR(30));
CREATE TABLE AUTHOR(AUTHOR_ID INT,NAME VARCHAR(30),CITY
VARCHAR(30),COUNTRY VARCHAR(30),PRIMARY KEY(AUTHOR_ID));
CREATE TABLE PUBLISHER(PUBLISHER_ID INT,NAME VARCHAR(30),CITY
VARCHAR(30),COUNTRY VARCHAR(30),PRIMARY KEY(PUBLISHER_ID));
CREATE TABLE CATEGORY(CATEGORY_ID INT,DESCRIPTION
VARCHAR(200), PRIMARY KEY(CATEGORY_ID));
CREATE TABLE CATALOG(BOOK_ID INT,TITLE VARCHAR(50), AUTHOR_ID
INT, PUBLISHER_ID INT,CATEGORY_ID INT,AYEAR INT,PRICE
INT,FOREIGN KEY(AUTHOR_ID) REFERENCES AUTHOR(AUTHOR_ID) ON
DELETE CASCADE, FOREIGN KEY(PUBLISHER_ID) REFERENCES
PUBLISHER(PUBLISHER_ID) ON DELETE CASCADE, FOREIGN
KEY(CATEGORY_ID) REFERENCES CATEGORY(CATEGORY_ID) ON
DELETE CASCADE);
CREATE TABLE CATALOG(BOOK_ID INT,TITLE VARCHAR(50), AUTHOR_ID
INT, PUBLISHER_ID INT,CATEGORY_ID INT,AYEAR INT,PRICE
INT,FOREIGN KEY(AUTHOR_ID) REFERENCES AUTHOR(AUTHOR_ID) ON
DELETE CASCADE, FOREIGN KEY(PUBLISHER_ID) REFERENCES
PUBLISHER(PUBLISHER_ID) ON DELETE CASCADE, FOREIGN
```

```
KEY(CATEGORY_ID) REFERENCES CATEGORY(CATEGORY_ID) ON
DELETE CASCADE, PRIMARY KEY(BOOK_ID));
CREATE TABLE ORDERDETAILS(ORDER_NO INT,BOOK_ID INT,
QUANTITY INT, FOREIGN KEY(BOOK_ID) REFERENCES
CATALOG(BOOK_ID));

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,"Willian Stalling","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",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 orderdetails values(1,11,5);
insert into orderdetails values(2,12,8);
insert into orderdetails values(3,13,15);
insert into orderdetails values(4,14,22);
insert into orderdetails values(5,15,3);
insert into orderdetails values(6,17,10);

/*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.*/
```

SELECT * FROM AUTHOR WHERE AUTHOR_ID IN (SELECT AUTHOR_ID FROM CATALOG WHERE AYEAR>2000 AND PRICE> (SELECT AVG(PRICE) FROM CATALOG) GROUP BY AUTHOR_ID HAVING COUNT(*)>1);

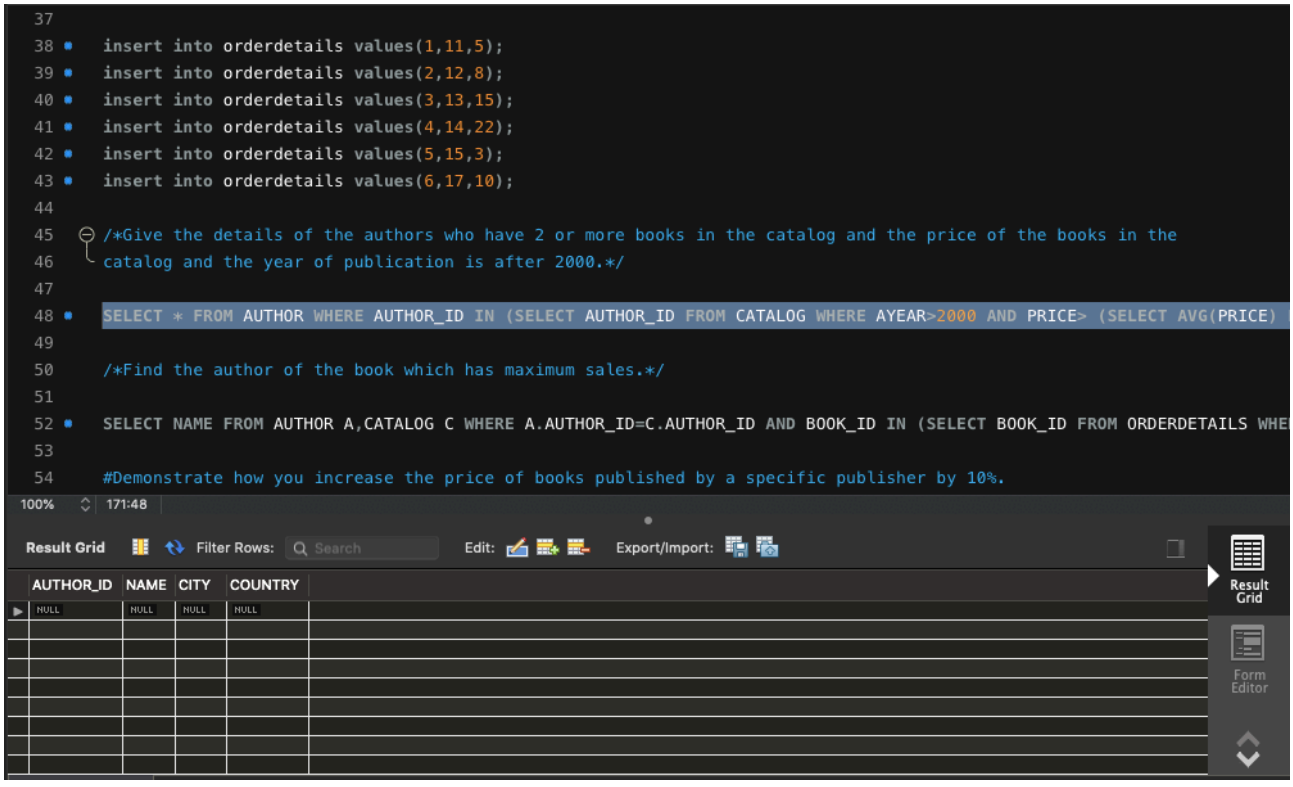
/*Find the author of the book which has maximum sales.*/

SELECT NAME FROM AUTHOR A,CATALOG C WHERE A.AUTHOR_ID=C.AUTHOR_ID AND BOOK_ID IN (SELECT BOOK_ID FROM ORDERDETAILS WHERE QUANTITY= (SELECT MAX(QUANTITY) FROM ORDERDETAILS));

#Demonstrate how you increase the price of books published by a specific publisher by 10%.
UPDATE CATALOG SET PRICE=1.1*PRICE WHERE PUBLISHER_ID IN (SELECT PUBLISHER_ID FROM PUBLISHER WHERE NAME='PEARSON');

SELECT * FROM AUTHOR;

Output:



```
40  insert into orderdetails values(3,13,15);
41  insert into orderdetails values(4,14,22);
42  insert into orderdetails values(5,15,3);
43  insert into orderdetails values(6,17,10);
44
45  /*Give the details of the authors who have 2 or more books in the catalog and the price of the books in the
46  catalog and the year of publication is after 2000.*/
47
48  SELECT * FROM AUTHOR WHERE AUTHOR_ID IN (SELECT AUTHOR_ID FROM CATALOG WHERE AYEAR>2000 AND PRICE> (SELECT AVG(PRICE)
49
50  /*Find the author of the book which has maximum sales.*/
51
52  SELECT NAME FROM AUTHOR A,CATALOG C WHERE A.AUTHOR_ID=C.AUTHOR_ID AND BOOK_ID IN (SELECT BOOK_ID FROM ORDERDETAILS WHE
53
54  #Demonstrate how you increase the price of books published by a specific publisher by 10%.
55  UPDATE CATALOG SET PRICE=1.1*PRICE WHERE PUBLISHER_ID IN (SELECT PUBLISHER_ID FROM PUBLISHER WHERE NAME='PEARSON');
56
57  SELECT * FROM AUTHOR;
```

100% 174:52

Result Grid | Filter Rows: Search | Export:

NAME
Karthik B P

Result Grid | Form Editor

```
40  insert into orderdetails values(3,13,15);
41  insert into orderdetails values(4,14,22);
42  insert into orderdetails values(5,15,3);
43  insert into orderdetails values(6,17,10);
44
45  /*Give the details of the authors who have 2 or more books in the catalog and the price of the books in the
46  catalog and the year of publication is after 2000.*/
47
48  SELECT * FROM AUTHOR WHERE AUTHOR_ID IN (SELECT AUTHOR_ID FROM CATALOG WHERE AYEAR>2000 AND PRICE> (SELECT AVG(PRICE)
49
50  /*Find the author of the book which has maximum sales.*/
51
52  SELECT NAME FROM AUTHOR A,CATALOG C WHERE A.AUTHOR_ID=C.AUTHOR_ID AND BOOK_ID IN (SELECT BOOK_ID FROM ORDERDETAILS WHE
53
54  #Demonstrate how you increase the price of books published by a specific publisher by 10%.
55  UPDATE CATALOG SET PRICE=1.1*PRICE WHERE PUBLISHER_ID IN (SELECT PUBLISHER_ID FROM PUBLISHER WHERE NAME='PEARSON');
56
57  SELECT * FROM CATALOG;
```

100% 23:57

Result Grid | Filter Rows: Search | Edit: | Export/Import:

BOOK_ID	TITLE	AUTHOR_ID	PUBLISHER_ID	CATEGORY_ID	AYEAR	PRICE
11	Unix System	1001	1	1001	2000	367
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	326
16	C++ Bible	1005	5	1001	2000	526
17	Cobol Handbook	1005	4	1001	2000	658
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid | Form Editor

Experiment 3:

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.

Code:

```
CREATE DATABASE ORDERPROCESSING;
USE ORDERPROCESSING;

CREATE TABLE CUSTOMER(CUST_NO INT,CNAME VARCHAR(30),CITY
VARCHAR(30), PRIMARY KEY(CUST_NO));
CREATE TABLE ORDERDET(ORDER_NO INT,ODATE DATE, CUST_NO
INT,ORDER_AMT INT,PRIMARY KEY(ORDER_NO), FOREIGN
KEY(CUST_NO) REFERENCES CUSTOMER(CUST_NO) ON DELETE
CASCADE);
CREATE TABLE ITEM(ITEM_NO INT, UNIT_PRICE INT, PRIMARY
KEY(ITEM_NO));
CREATE TABLE ORDERITEM(ORDER_NO INT, ITEM_NO INT, QTY INT,
FOREIGN KEY(ORDER_NO) REFERENCES ORDERDET(ORDER_NO) ON
DELETE CASCADE, FOREIGN KEY(ITEM_NO) REFERENCES
ITEM(ITEM_NO) ON DELETE CASCADE);
```

```
CREATE TABLE WAREHOUSE(WAREHOUSE_NO INT, CITY VARCHAR(30),
PRIMARY KEY(WAREHOUSE_NO));
CREATE TABLE SHIPMENT(ORDER_NO INT,WAREHOUSE_NO INT,
SHIP_DATE DATE,FOREIGN KEY(ORDER_NO) REFERENCES
ORDERDET(ORDER_NO) ON DELETE CASCADE, FOREIGN
KEY(WAREHOUSE_NO) REFERENCES WAREHOUSE(WAREHOUSE_NO) ON
DELETE CASCADE);

INSERT INTO CUSTOMER VALUES(771,"PUSHPA K","BANGALORE"),
(772,"SUMAN","MUMBAI"),(773,"SOURAV", "CALICUT"),
(774 ,"LAILA","HYDERABAD"),(775 ,"FAIZAL" ,"BANGALORE");

INSERT INTO ORDERDET VALUES(111," 2002-01-22", 771, 18000),
(112 ,"2002-07-30", 774, 6000),
(113, "2003-04-03", 775, 9000),
(114, "2003-11-03", 775, 29000),
(115, "2003-12-10", 773, 29000),
(116 ,"2004-08-19", 772, 56000),
(117, "2004-09-10", 771, 20000),
(118, "2004-11-20" ,775, 29000),
(119, "2005-02-13", 774, 29000),
(120, "2005-10-13", 775, 29000);

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

INSERT INTO ORDERITEM VALUES(111, 5001, 50),
(112, 5003, 20),
(113, 5002, 50),
(114, 5005, 60),
(115, 5004, 90),
(116, 5001, 10),
(117, 5003, 80),
(118, 5005, 50),
(119, 5002, 10),
(120, 5004, 45);

INSERT INTO WAREHOUSE VALUES(1, "DELHI"),
(2, "BOMBAY"),
(3 ,"CHENNAI"),
(4 ,"BANGALORE"),
(5 ,"BANGALORE"),
```

```
(6 ,"DELHI"),
(7 ,"BOMBAY"),
(8 ,"CHENNAI"),
(9 ,"DELHI"),
(10 ,"BANGALORE");


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

/*List the order# for orders that were shipped from all warehouses that the company
has in a
specific city.*/
SELECT O.ORDER_NO FROM ORDERDET O, WAREHOUSE W, SHIPMENT S
WHERE W.WAREHOUSE_NO = S.WAREHOUSE_NO AND W.CITY =
"BANGALORE";

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

DELETE FROM ITEM WHERE ITEM_NO = 5005;

SELECT * FROM ORDERITEM;

/*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.*/

CREATE VIEW DIFFTABLE(CUSTNAME, NO_OF_ORDERS,
AVG_ORDER_AMT) AS SELECT C.CNAME,COUNT(O.ORDER_NO) ,
AVG(O.ORDER_AMT) FROM CUSTOMER C, ORDERDET O WHERE
C.CUST_NO = O.CUST_NO GROUP BY O.CUST_NO;
```

SELECT * FROM DIFFTABLE;

Output:

```
59    (117, 5, "2004-09-10"),
60    (118, 7, "2004-11-30"),
61    (119, 7, "2005-04-30"),
62    (120, 6, "2005-12-21");
63
64    /*List the order# for orders that were shipped from all warehouses that the company has in a
65    specific city.*/
66    SELECT O.ORDER_NO FROM ORDERDET O, WAREHOUSE W, SHIPMENT S WHERE W.WAREHOUSE_NO = S.WAREHOUSE_NO AND W.CITY = "BANGALO
67
68    /*Demonstrate how you delete item# 10 from the ITEM table and make that field null in the
69    ORDER_ITEM*/
70
71    DELETE FROM ITEM WHERE ITEM_NO = 5005;
72
73    SELECT * FROM ORDERITEM;
```

100% 123:66

Result Grid | Filter Rows: Search | Export:

ORDER_NO
111
111
117
117
116
116
115
115
112
112
119
119
113
113

Result Grid | Form Editor | Field Types

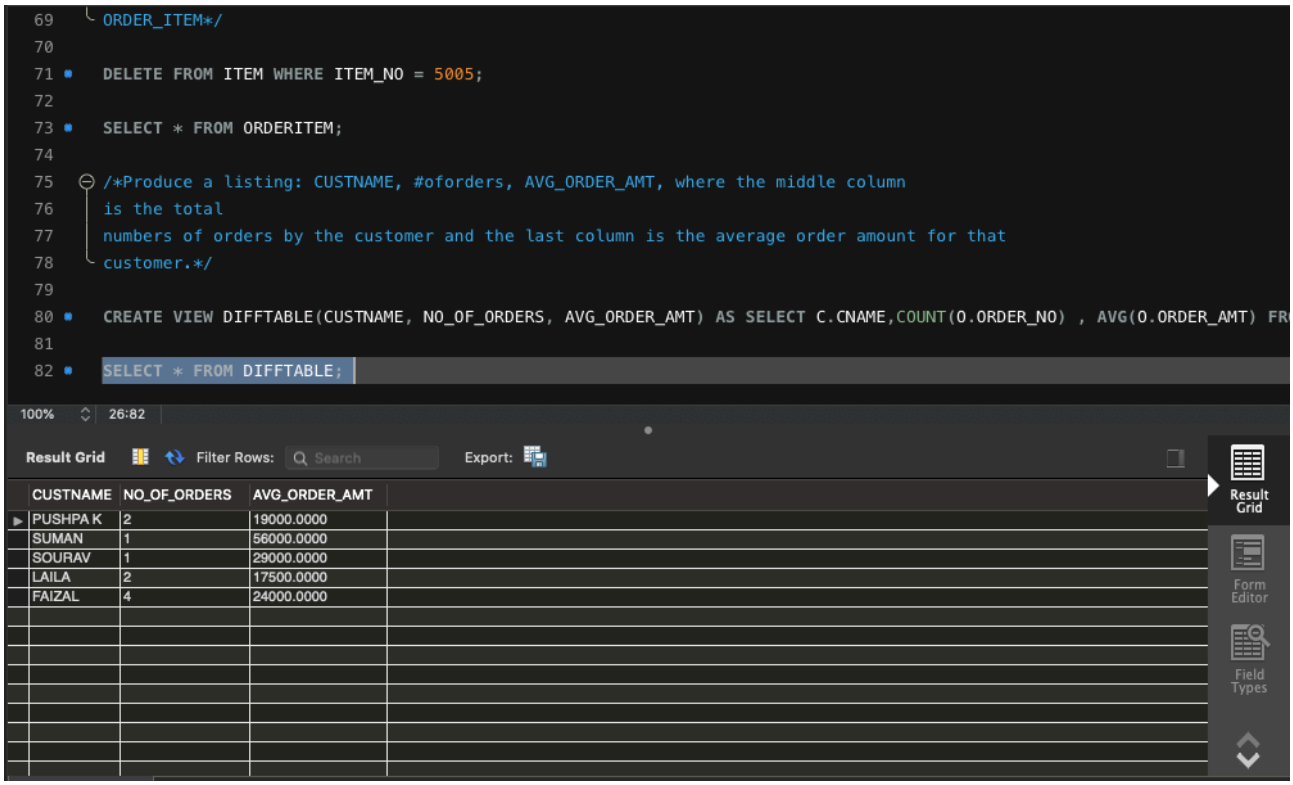
```
60    (118, 7, "2004-11-30"),
61    (119, 7, "2005-04-30"),
62    (120, 6, "2005-12-21");
63
64    /*List the order# for orders that were shipped from all warehouses that the company has in a
65    specific city.*/
66    SELECT O.ORDER_NO FROM ORDERDET O, WAREHOUSE W, SHIPMENT S WHERE W.WAREHOUSE_NO = S.WAREHOUSE_NO AND W.CITY = "BANGALO
67
68    /*Demonstrate how you delete item# 10 from the ITEM table and make that field null in the
69    ORDER_ITEM*/
70
71    DELETE FROM ITEM WHERE ITEM_NO = 5005;
72
73    SELECT * FROM ORDERITEM;
74
```

100% 25:73

Result Grid | Filter Rows: Search | Export:

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

Result Grid | Form Editor | Field Types


69 ORDER_ITEM*/
70
71 DELETE FROM ITEM WHERE ITEM_NO = 5005;
72
73 SELECT * FROM ORDERITEM;
74
75 /*Produce a listing: CUSTNAME, #oforders, AVG_ORDER_AMT, where the middle column
76 is the total
77 numbers of orders by the customer and the last column is the average order amount for that
78 customer.*/
79
80 CREATE VIEW DIFFTABLE(CUSTNAME, NO_OF_ORDERS, AVG_ORDER_AMT) AS SELECT C.CNAME,COUNT(O.ORDER_NO) , AVG(O.ORDER_AMT) FR
81
82 SELECT * FROM DIFFTABLE;
100%
26:82
Result Grid
Filter Rows:
Search
Export:
CUSTNAME NO_OF_ORDERS AVG_ORDER_AMT
PUSHPA K 2 19000.0000
SUMAN 1 56000.0000
SOURAV 1 29000.0000
LAILA 2 17500.0000
FAIZAL 4 24000.0000
Result Grid
Form Editor
Field Types

Experiment 4:

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.

Code:

```
CREATE DATABASE BANKING;
USE BANKING;
CREATE TABLE BRANCH(BRANCH_NAME VARCHAR(30), BRANCH_CITY VARCHAR(30), ASSETS FLOAT, PRIMARY KEY(BRANCH_NAME));
CREATE TABLE ACCOUNTS(ACC_NO INT, BRANCH_NAME VARCHAR(30), BALANCE FLOAT, PRIMARY KEY(ACC_NO),FOREIGN KEY(BRANCH_NAME) REFERENCES BRANCH(BRANCH_NAME) ON DELETE CASCADE);
CREATE TABLE CUSTOMER(ACC_NO INT,CUST_NAME VARCHAR(30), CUST_STREET VARCHAR(100), CUST_CITY VARCHAR(30), PRIMARY KEY(CUST_NAME), FOREIGN KEY ACCOUNTS(ACC_NO) REFERENCES ACCOUNTS(ACC_NO) ON DELETE CASCADE ON UPDATE CASCADE);
CREATE TABLE LOAN(LOAN_NO INT, BRANCH_NAME VARCHAR(30), AMOUNT FLOAT, PRIMARY KEY(LOAN_NO), FOREIGN KEY(BRANCH_NAME) REFERENCES BRANCH(BRANCH_NAME) ON DELETE CASCADE);
CREATE TABLE BORROWER(CUST_NAME VARCHAR(30), LOAN_NO INT, FOREIGN KEY(CUST_NAME) REFERENCES CUSTOMER(CUST_NAME) ON DELETE CASCADE, FOREIGN KEY(LOAN_NO) REFERENCES LOAN(LOAN_NO) ON DELETE CASCADE);
```

#ENTERING THE DATA

```
INSERT INTO BRANCH VALUES("SBI PD NAGAR", "BANGALORE", 200000),
("SBI RAJAJI NAGAR", "BANGALORE" ,500000),
("SBI JAYANAGAR" ,"BANGALORE", 660000),
("SBI VIJAY NAGAR" ,"BANGALORE", 870000),
("SBI HOSAKEREHALLI" ,"BANGALORE", 550000);

INSERT INTO BRANCH VALUES("CANARA J C ROAD", "BANGALORE",
3000000),
("CANARA-SYNDICATE MANIPAL","MANIPAL",2000000),
("CANARA K M MARG","UDUPI",100000),
("CANARA PAHARGANJ","DELHI",200000),
("CANARA LAJPATHNAGAR","DELHI",300000);

INSERT INTO ACCOUNTS VALUES(1234602, "SBI HOSAKEREHALLI", 5000),
(1234603, "SBI VIJAY NAGAR", 5000),
(1234604, "SBI JAYANAGAR", 5000),
(1234605, "SBI RAJAJI NAGAR", 10000),
(1234503, "SBI VIJAY NAGAR", 40000),
(1234504, "SBI PD NAGAR", 4000);

INSERT INTO ACCOUNTS VALUES(282577, "CANARA J C ROAD", 15000),
(235887, "CANARA J C ROAD", 25000),
(367733,"CANARA J C ROAD",56300),
(462822,"CANARA J C ROAD",23500),
(127298,"CANARA-SYNDICATE MANIPAL", 13000),
(877373,"CANARA-SYNDICATE MANIPAL",19000),
(122933,"CANARA-SYNDICATE MANIPAL",16770),
(544556,"CANARA K M MARG",12000),
(896565,"CANARA K M MARG",12300),
(453433,"CANARA PAHARGANJ",67000),
(453462,"CANARA PAHARGANJ", 34000),
(232377,"CANARA LAJPATHNAGAR", 12000),
(655665,"CANARA LAJPATHNAGAR", 23000);

INSERT INTO CUSTOMER VALUES(1234602,"KEZAR", "M G ROAD",
"BANGALORE"),
(1234603,"LAL KRISHNA" ,"ST MKS ROAD" ,"BANGALORE"),
(1234604,"RAHUL" ,"AUGSTEN ROAD" ,"BANGALORE"),
(1234605,"LALLU" ,"V S ROAD" ,"BANGALORE"),
(1234503,"FAIZAL" ,"RESEDENCY ROAD" ,"BANGALORE"),
(1234504,"RAJEEV" ,"DICKNSN ROAD" ,"BANGALORE");

INSERT INTO CUSTOMER
VALUES(282577,"KRISHNA","VIJAYNAGAR","BANGALORE"),
(235887,"ANIRUDH","BANASHANKARI 5TH STAGE","BANGALORE"),
```

(367733,"CHIRANTH","HOSAHALLI 5TH MAIN","BANGALORE"),
(462822,"MAITHILI","RAJAJINAGAR 4TH BLOCK","BANGALORE"),
(127298,"SAHANA","GANDHI ROAD","MANIPAL"),
(877373,"SARIKA","KASTURBA ROAD","MANIPAL"),
(122933,"SANJAY","PAI MARG","MANIPAL"),
(544556,"SHREYA","NEW THARAGUPET","UDUPI"),
(896565,"UDISHA","COURT ROAD","UDUPI"),
(453433,"SHAMBHAVI","KAROLBAGH","DELHI"),
(453462,"PRANAV","PAHARGANJ","DELHI"),
(655665,"VISHVESH","LAJPATHNAGAR","DELHI"),
(232377,"VAISHAK","LAJPATHNAGAR","DELHI");

INSERT INTO LOAN VALUES(10011 ,"SBI JAYANAGAR" ,10000),
(10012 ,"SBI VIJAY NAGAR", 5000),
(10013 ,"SBI HOSAKEREHALLI", 20000),
(10014 ,"SBI PD NAGAR" ,15000),
(10015 ,"SBI RAJAJI NAGAR" ,25000);

INSERT INTO BORROWER VALUES("KEZAR", 10011),
("LAL KRISHNA", 10012),
("RAHUL", 10013),
("LALLU", 10014),
("LAL KRISHNA" ,10015);

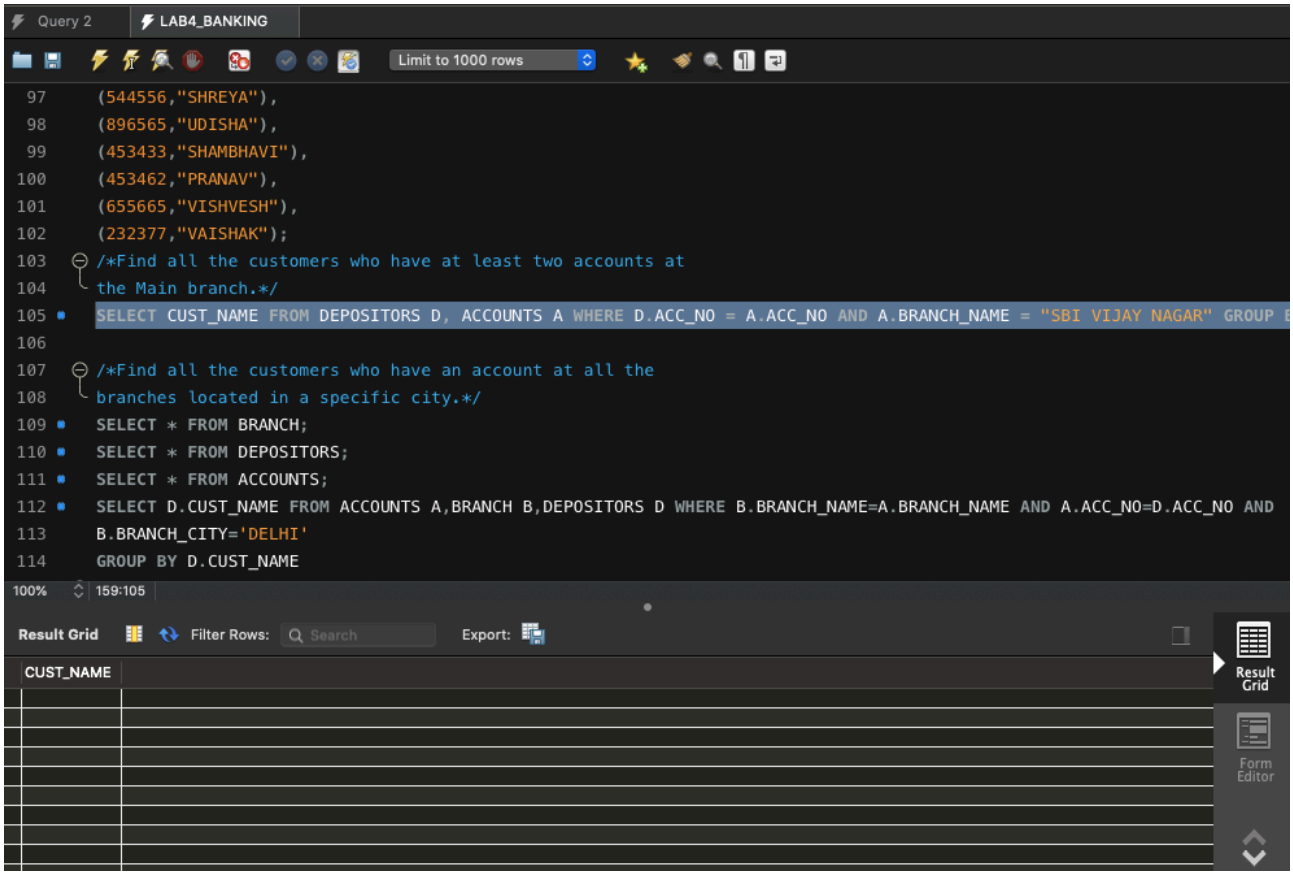
CREATE TABLE DEPOSITORS(ACC_NO INT, CUST_NAME VARCHAR(30), FOREIGN KEY(ACC_NO) REFERENCES ACCOUNTS(ACC_NO) ON DELETE CASCADE, FOREIGN KEY(CUST_NAME) REFERENCES CUSTOMER(CUST_NAME) ON DELETE CASCADE);

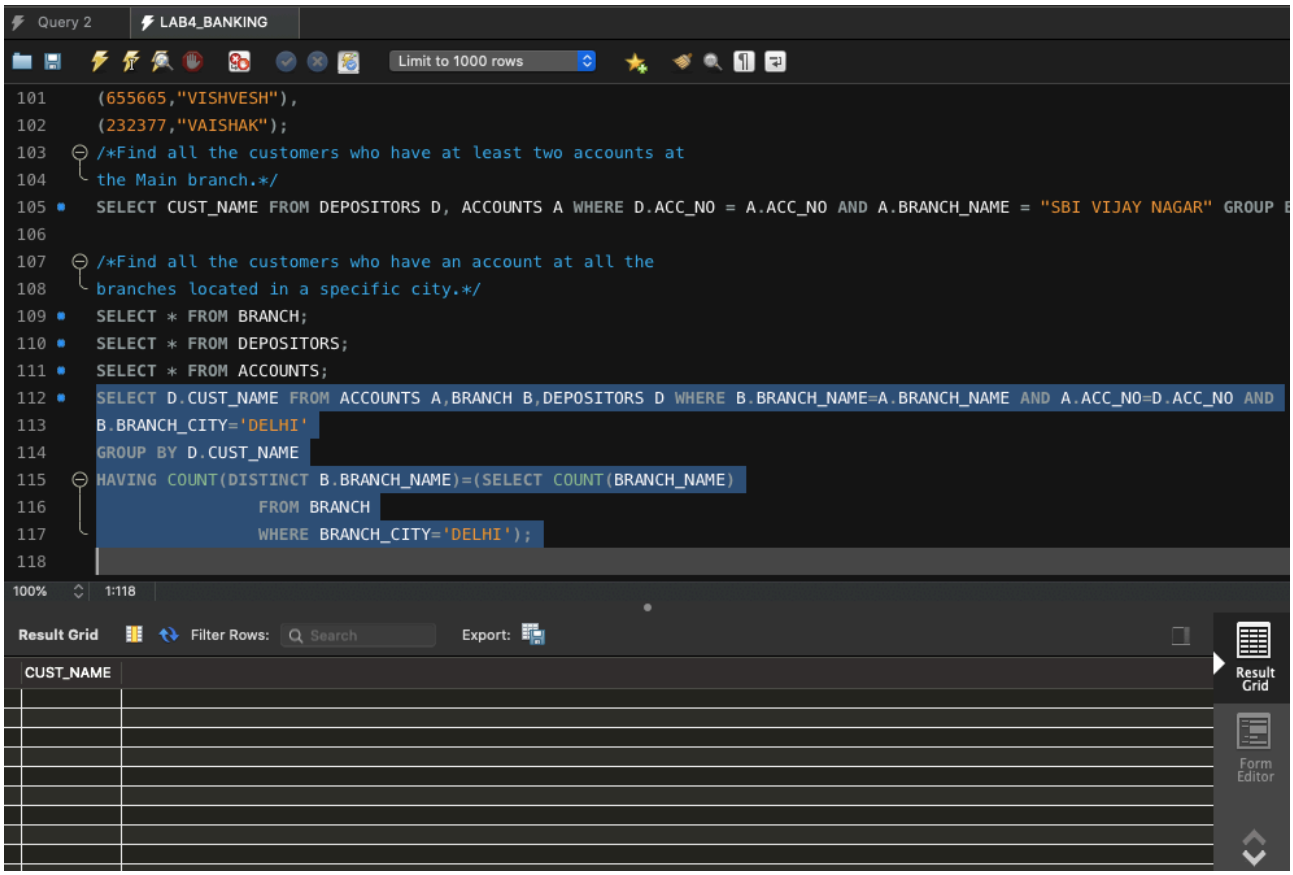
INSERT INTO DEPOSITORS VALUES(1234602, "KEZAR"),
(1234603, "LAL KRISHNA"),
(1234604,"RAHUL"),
(1234605,"LALLU"),
(1234503,"FAIZAL"),
(1234504,"RAJEEV");

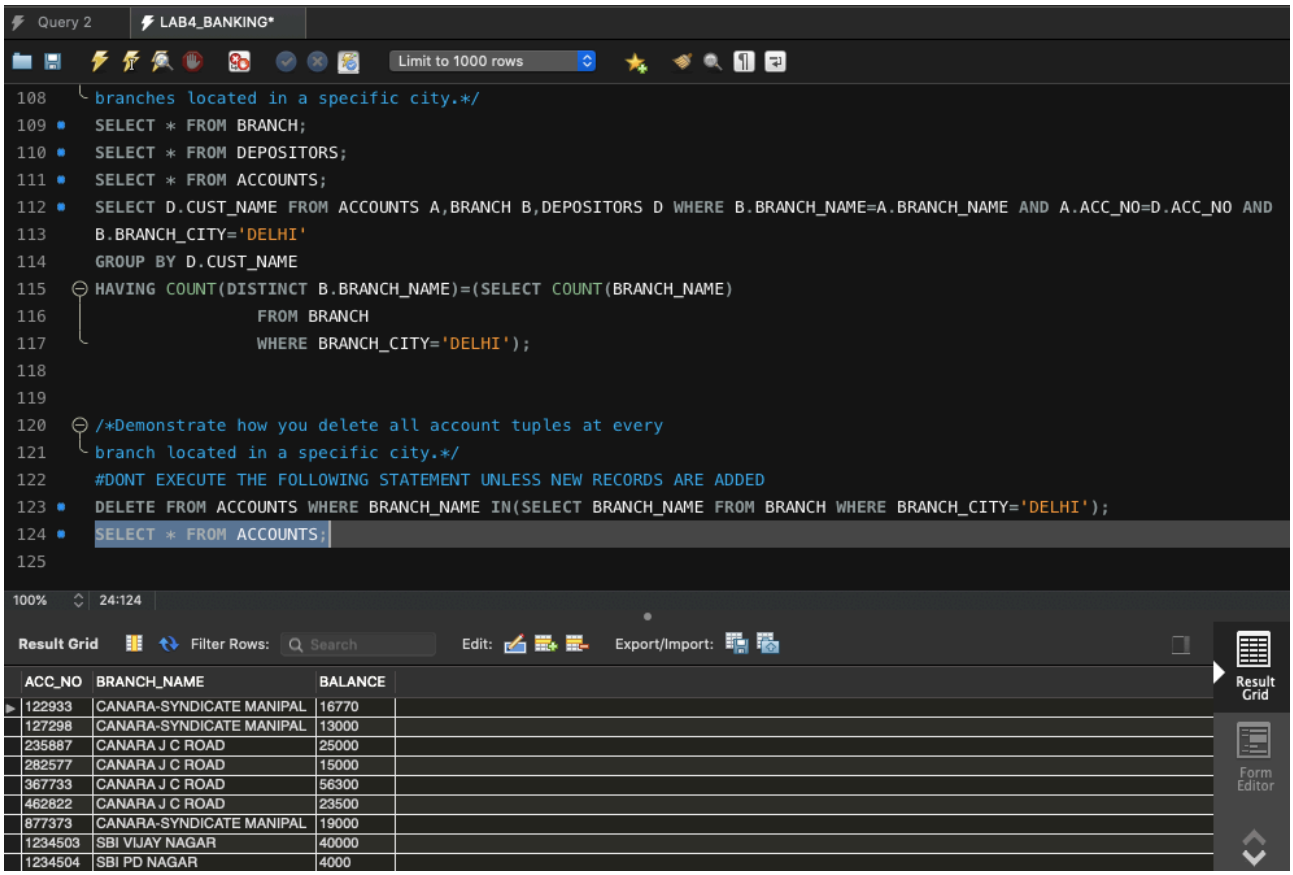
INSERT INTO DEPOSITORS VALUES(282577,"KRISHNA"),
(235887,"ANIRUDH"),
(367733,"CHIRANTH"),
(462822,"MAITHILI"),
(127298,"SAHANA"),
(877373,"SARIKA"),
(122933,"SANJAY"),
(544556,"SHREYA"),

```
(896565,"UDISHA"),
(453433,"SHAMBHAVI"),
(453462,"PRANAV"),
(655665,"VISHVESH"),
(232377,"VAISHAK");
/*Find all the customers who have at least two accounts at
the Main branch.*/
SELECT CUST_NAME FROM DEPOSITORS D, ACCOUNTS A WHERE
D.ACC_NO = A.ACC_NO AND A.BRANCH_NAME = "SBI VIJAY NAGAR"
GROUP BY D.CUST_NAME HAVING COUNT(A.ACC_NO)>=2;

/*Find all the customers who have an account at all the
branches located in a specific city.*/
SELECT * FROM BRANCH;
SELECT * FROM DEPOSITORS;
SELECT * FROM ACCOUNTS;
SELECT D.CUST_NAME FROM ACCOUNTS A,BRANCH B,DEPOSITORS D
WHERE B.BRANCH_NAME=A.BRANCH_NAME AND
A.ACC_NO=D.ACC_NO AND
B.BRANCH_CITY='DELHI'
GROUP BY D.CUST_NAME
HAVING COUNT(DISTINCT B.BRANCH_NAME)=(SELECT
COUNT(BRANCH_NAME)
            FROM BRANCH
            WHERE BRANCH_CITY='DELHI');


/*Demonstrate how you delete all account tuples at every
branch located in a specific city.*/
#DONT EXECUTE THE FOLLOWING STATEMENT UNLESS NEW RECORDS
ARE ADDED
DELETE FROM ACCOUNTS WHERE BRANCH_NAME IN(SELECT
BRANCH_NAME FROM BRANCH WHERE BRANCH_CITY='UDUPI');
SELECT * FROM ACCOUNTS;
```

OUTPUT:


Query 2
LAB4_BANKING
Limit to 1000 rows
(544556,"SHREYA"),
(896565,"UDISHA"),
(453433,"SHAMBHAVI"),
(453462,"PRANAV"),
(655665,"VISHVESH"),
(232377,"VAISHAK");
/*Find all the customers who have at least two accounts at
the Main branch.*/
SELECT CUST_NAME FROM DEPOSITORS D, ACCOUNTS A WHERE D.ACC_NO = A.ACC_NO AND A.BRANCH_NAME = "SBI VIJAY NAGAR" GROUP
/*Find all the customers who have an account at all the
branches located in a specific city.*/
SELECT * FROM BRANCH;
SELECT * FROM DEPOSITORS;
SELECT * FROM ACCOUNTS;
SELECT D.CUST_NAME FROM ACCOUNTS A,BRANCH B,DEPOSITORS D WHERE B.BRANCH_NAME=A.BRANCH_NAME AND A.ACC_NO=D.ACC_NO AND
B.BRANCH_CITY='DELHI'
GROUP BY D.CUST_NAME
100%
159:105
Result Grid
Filter Rows:
Search
Export:
CUST_NAME
Result Grid
Form Editor


Query 2
LAB4_BANKING
Limit to 1000 rows
(655665,"VISHVESH"),
(232377,"VAISHAK");
/*Find all the customers who have at least two accounts at
the Main branch.*/
SELECT CUST_NAME FROM DEPOSITORS D, ACCOUNTS A WHERE D.ACC_NO = A.ACC_NO AND A.BRANCH_NAME = "SBI VIJAY NAGAR" GROUP
/*Find all the customers who have an account at all the
branches located in a specific city.*/
SELECT * FROM BRANCH;
SELECT * FROM DEPOSITORS;
SELECT * FROM ACCOUNTS;
SELECT D.CUST_NAME FROM ACCOUNTS A,BRANCH B,DEPOSITORS D WHERE B.BRANCH_NAME=A.BRANCH_NAME AND A.ACC_NO=D.ACC_NO AND
B.BRANCH_CITY='DELHI'
GROUP BY D.CUST_NAME
HAVING COUNT(DISTINCT B.BRANCH_NAME)=(SELECT COUNT(BRANCH_NAME)
FROM BRANCH
WHERE BRANCH_CITY='DELHI');
100%
1:118
Result Grid
Filter Rows:
Search
Export:
CUST_NAME
Result Grid
Form Editor



ACC_NO	BRANCH_NAME	BALANCE
122933	CANARA-SYNDICATE MANIPAL	16770
127298	CANARA-SYNDICATE MANIPAL	13000
235887	CANARA J C ROAD	25000
282577	CANARA J C ROAD	15000
367733	CANARA J C ROAD	56300
462822	CANARA J C ROAD	23500
877373	CANARA-SYNDICATE MANIPAL	19000
1234503	SBI VIJAY NAGAR	40000
1234504	SBI PD NAGAR	4000

Experiment 5:

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)

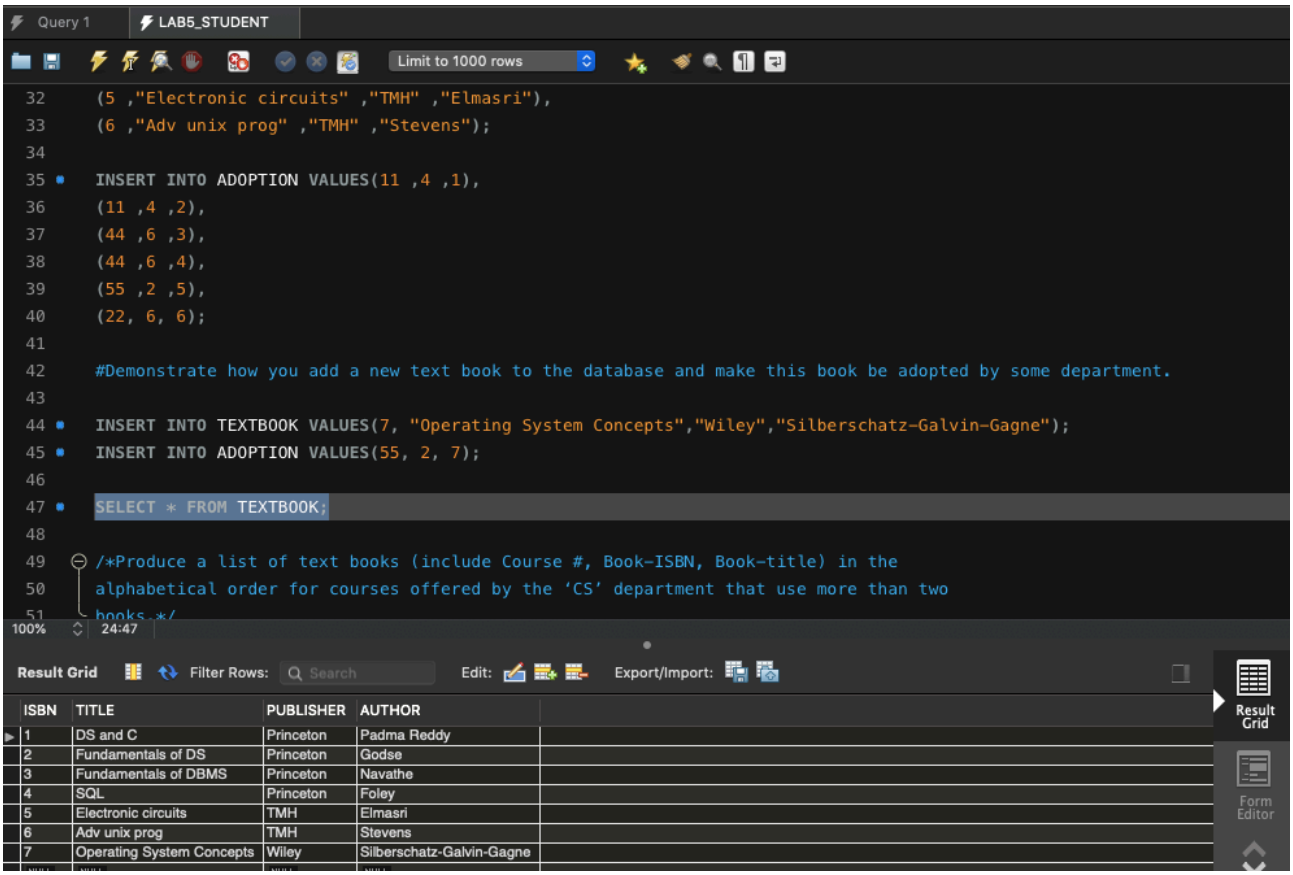
i) Create the above tables by properly specifying the primary keys and the foreign keys.
ii) Enter at least five tuples for each relation.
iii) Demonstrate how you add a new text book to the database and make this book be adopted by some department.
iv) Produce a list of text books (include Course #, 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.

Code:

```
CREATE DATABASE STUDENT_ENROLLMENT;
USE STUDENT_ENROLLMENT;

CREATE TABLE STUDENT(REGNO VARCHAR(10), NAME VARCHAR(30), MAJOR
VARCHAR(10), BDATE DATE, PRIMARY KEY(REGNO));
CREATE TABLE COURSE(COURSE_NO INT, CNAME VARCHAR(30), DEPT VARCHAR(4),
PRIMARY KEY(COURSE_NO));
CREATE TABLE ENROLL(REGNO VARCHAR(10), COURSE_NO INT, SEM INT, MARKS
INT, FOREIGN KEY(REGNO) REFERENCES STUDENT(REGNO) ON DELETE CASCADE
ON UPDATE CASCADE, FOREIGN KEY(COURSE_NO) REFERENCES
COURSE(COURSE_NO) ON DELETE CASCADE ON UPDATE CASCADE);
CREATE TABLE TEXTBOOK(ISBN INT, TITLE VARCHAR(30), PUBLISHER VARCHAR(30),
AUTHOR VARCHAR(30), PRIMARY KEY(ISBN));
CREATE TABLE ADOPTION(COURSE_NO INT, SEM INT, ISBN INT, FOREIGN
KEY(COURSE_NO) REFERENCES COURSE(COURSE_NO) ON DELETE CASCADE ON
UPDATE CASCADE, FOREIGN KEY(ISBN) REFERENCES TEXTBOOK(ISBN) ON DELETE
CASCADE ON UPDATE CASCADE);

INSERT INTO STUDENT VALUES("CS01", "RAM" ,"DS" ,"1986-03-12"),
("IS02" ,"SMITH" ,"USP" ,"1987-12-23"),
("EC03" ,"AHMED" ,"SNS" ,"1985-04-17"),
("CS03" ,"SNEHA" ,"DBMS" ,"1987-01-01"),
("TC05" ,"AKHILA" ,"EC" ,"1986-10-06");

INSERT INTO COURSE VALUES(11 ,"DS" ,"CS"),
(22 ,"USP" ,"IS"),
```

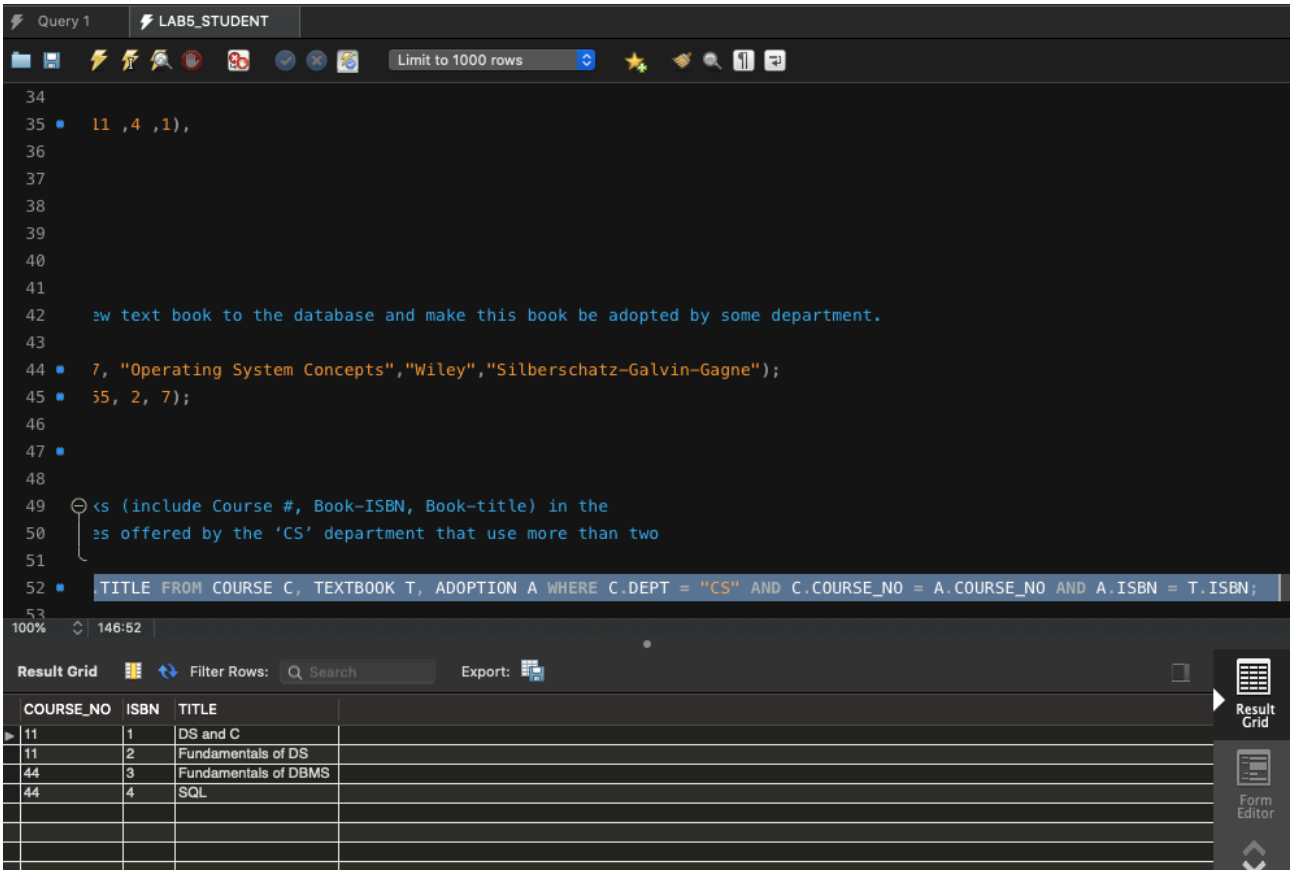
```
(33 ,"SNS" ,"EC"),
(44 ,"DBMS" ,"CS"),
(55 ,"EC" ,"TC");

INSERT INTO ENROLL VALUES("CS01" ,11 ,4 ,85),
("IS02" ,22, 6, 80),
("EC03", 33, 2, 80),
("CS03", 44, 6, 75),
("TC05", 55, 2, 8);

INSERT INTO TEXTBOOK VALUES(1 ,"DS and C" ,"Princeton" ,"Padma Reddy"),
(2 ,"Fundamentals of DS" ,"Princeton", "Godse"),
(3 ,"Fundamentals of DBMS" ,"Princeton" ,"Navathe"),
(4 ,"SQL" ,"Princeton" ,"Foley"),
(5 ,"Electronic circuits" ,"TMH" ,"Elmasri"),
(6 ,"Adv unix prog" ,"TMH" ,"Stevens");

INSERT INTO ADOPTION VALUES(11 ,4 ,1),
(11 ,4 ,2),
(44 ,6 ,3),
(44 ,6 ,4),
(55 ,2 ,5),
(22, 6, 6);

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

INSERT INTO TEXTBOOK VALUES(7, "Operating System Concepts","Wiley","Silberschatz-
Galvin-Gagne");
INSERT INTO ADOPTION VALUES(55, 2, 7);

SELECT * FROM TEXTBOOK;

/*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.ISBN,T.TITLE FROM COURSE C, TEXTBOOK T, ADOPTION A
WHERE C.DEPT = "CS" AND C.COURSE_NO = A.COURSE_NO AND A.ISBN = T.ISBN;

/*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, TEXTBOOK T, ADOPTION A
WHERE T.PUBLISHER = "PRINCETON" AND C.COURSE_NO = A.COURSE_NO AND
A.ISBN = T.ISBN)
AND C.DEPT NOT IN (SELECT C.DEPT FROM COURSE C, TEXTBOOK T, ADOPTION A
WHERE T.PUBLISHER != "PRINCETON" AND C.COURSE_NO = A.COURSE_NO AND
A.ISBN = T.ISBN);
```

OUTPUT:



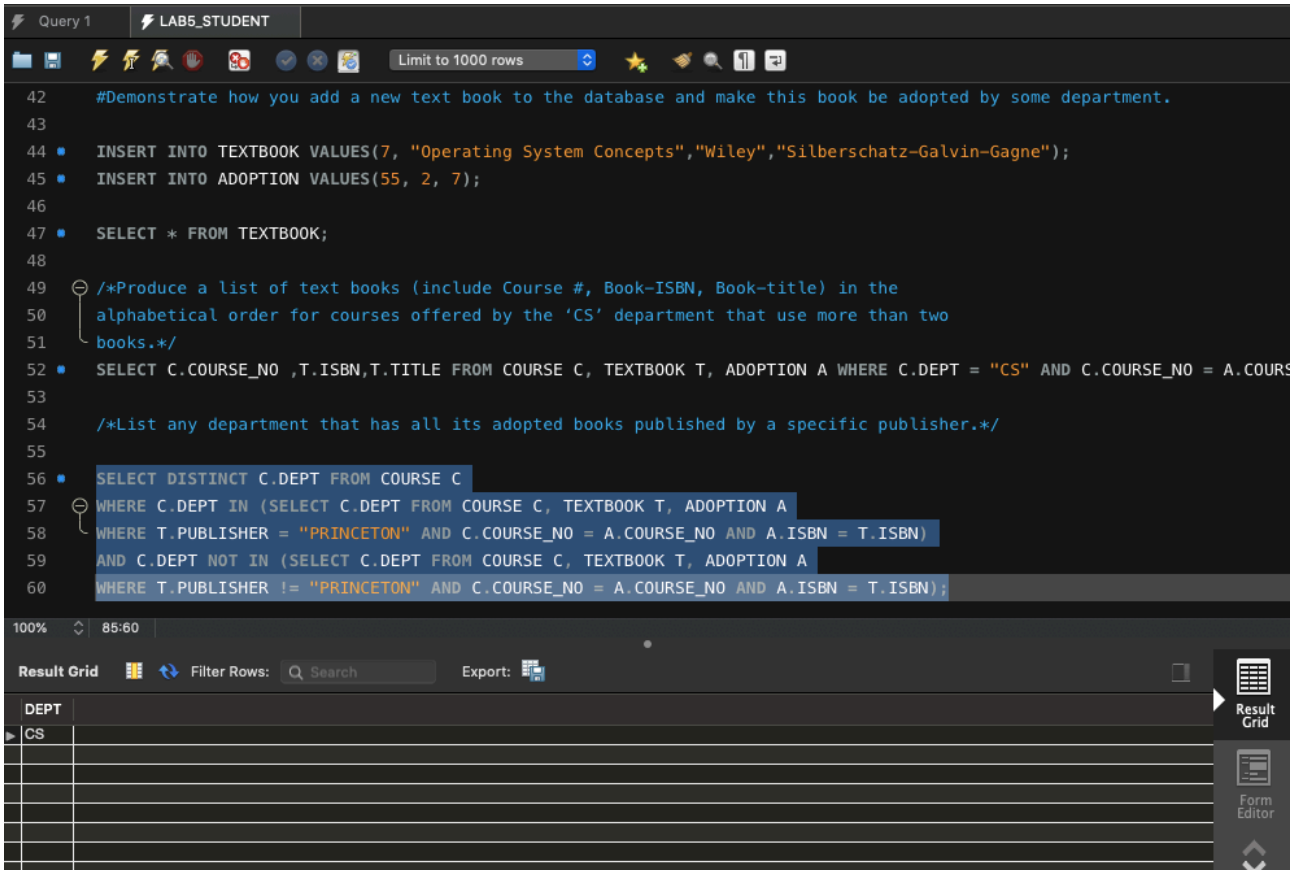
```
(5 ,"Electronic circuits" ,"TMH" ,"Elmasri"),
(6 ,"Adv unix prog" ,"TMH" ,"Stevens");

INSERT INTO ADOPTION VALUES(11 ,4 ,1),
(11 ,4 ,2),
(44 ,6 ,3),
(44 ,6 ,4),
(55 ,2 ,5),
(22, 6, 6);

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

INSERT INTO TEXTBOOK VALUES(7, "Operating System Concepts","Wiley","Silberschatz-Galvin-Gagne");
INSERT INTO ADOPTION VALUES(55, 2, 7);

SELECT * FROM TEXTBOOK;

/*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 */
```

ISBN	TITLE	PUBLISHER	AUTHOR
1	DS and C	Princeton	Padma Reddy
2	Fundamentals of DS	Princeton	Godse
3	Fundamentals of DBMS	Princeton	Navathe
4	SQL	Princeton	Foley
5	Electronic circuits	TMH	Elmasri
6	Adv unix prog	TMH	Stevens
7	Operating System Concepts	Wiley	Silberschatz-Galvin-Gagne



```
11 ,4 ,1),

ew text book to the database and make this book be adopted by some department.

7, "Operating System Concepts","Wiley","Silberschatz-Galvin-Gagne");
55, 2, 7);

ks (include Course #, Book-ISBN, Book-title) in the
es offered by the 'CS' department that use more than two

TITLE FROM COURSE C, TEXTBOOK T, ADOPTION A WHERE C.DEPT = "CS" AND C.COURSE_NO = A.COURSE_NO AND A.ISBN = T.ISBN;
```

COURSE_NO	ISBN	TITLE
11	1	DS and C
11	2	Fundamentals of DS
44	3	Fundamentals of DBMS
44	4	SQL


Query 1
LAB5_STUDENT
Limit to 1000 rows
#Demonstrate how you add a new text book to the database and make this book be adopted by some department.
INSERT INTO TEXTBOOK VALUES(7, "Operating System Concepts","Wiley","Silberschatz-Galvin-Gagne");
INSERT INTO ADOPTION VALUES(55, 2, 7);
SELECT * FROM TEXTBOOK;
/*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.ISBN,T.TITLE FROM COURSE C, TEXTBOOK T, ADOPTION A WHERE C.DEPT = "CS" AND C.COURSE_NO = A.COURS
/*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, TEXTBOOK T, ADOPTION A
WHERE T.PUBLISHER = "PRINCETON" AND C.COURSE_NO = A.COURSE_NO AND A.ISBN = T.ISBN)
AND C.DEPT NOT IN (SELECT C.DEPT FROM COURSE C, TEXTBOOK T, ADOPTION A
WHERE T.PUBLISHER != "PRINCETON" AND C.COURSE_NO = A.COURSE_NO AND A.ISBN = T.ISBN);
100%
85:60
Result Grid
Filter Rows:
Search
Export:
DEPT
CS
Result Grid
Form Editor

Experiment 6:

Consider the schema for Movie Database:
ACTOR(Act_id, Act_Name, Act_Gender)

DIRECTOR(Dir_id, Dir_Name, Dir_Phone)

MOVIES(Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id) MOVIE_CAST(Act_id, Mov_id, Role)

RATING(Mov_id, Rev_Stars)

Write SQL queries to

i. List the titles of all movies directed by 'Hitchcock'.

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

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

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

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

Code:

```
CREATE DATABASE MOVIES;

USE MOVIES;


CREATE TABLE ACTOR(ACT_ID INT, ACT_NAME VARCHAR(30), ACT_GENDER
    VARCHAR(6), PRIMARY KEY(ACT_ID));

CREATE TABLE DIRECTOR(DIR_ID INT, DIR_NAME VARCHAR(50), DIR_PHONE INT,
    PRIMARY KEY(DIR_ID));

CREATE TABLE MOVIES(MOV_ID INT, MOV_TITLE VARCHAR(100), MOV_YEAR INT,
    MOV_LANG VARCHAR(30), DIR_ID INT, PRIMARY KEY(MOV_ID), FOREIGN
    KEY(DIR_ID) REFERENCES DIRECTOR(DIR_ID) ON DELETE CASCADE ON
    UPDATE CASCADE);

CREATE TABLE MOVIE_CAST(ACT_ID INT, MOV_ID INT, ROLEP VARCHAR(30),

FOREIGN KEY(ACT_ID) REFERENCES ACTOR(ACT_ID) ON DELETE CASCADE ON
    UPDATE CASCADE,

FOREIGN KEY(MOV_ID) REFERENCES MOVIES(MOV_ID) ON DELETE CASCADE ON
    UPDATE CASCADE);

CREATE TABLE RATING(MOV_ID INT, STARS FLOAT,
```

```
FOREIGN KEY(MOV_ID) REFERENCES MOVIES(MOV_ID) ON DELETE CASCADE ON
        UPDATE CASCADE);



#DATA ENTRY

#FIRST 3 IN MURDER! DIR HITCHCOOK 1930 ENGLISH 4.8 001->ACTOR-MANAGER,
        002->ACTRESS 003->STAGE-MANAGER

#THE MAN WHO KNEW TOO MUCH, HITCHCOOK 1934 ENGLISH 4.5 4->HERO 5-
        >VILLAIN

#JURASSIC PARK STEVEN SPIELBERG 1993 ENGLISH 6->HERO 7-HEROINE

INSERT INTO ACTOR VALUES(001, "HERBERT MARSHALL", "MALE"),

(002, "NORAH BURING", "FEMALE"),

(003, "EDWARD CHAPMAN" , "MALE"),

(004, "LESLIE BANKS", "MALE"),

(005, "PETER LORRE", "MALE"),

(006, "SAM NEILL", "MALE"),

(007, "LAURA DERN", "FEMALE"),

(008, "RAJ KUMAR" ,"MALE"),

(009, "ANANT NAG", "MALE");


INSERT INTO ACTOR VALUES(010, "KALPANA", "FEMALE");


INSERT INTO DIRECTOR VALUES(001, "HITCHCOOK", 1234567890),

(002, "STEVEN SPIELBERG", 1345167890),

(003, "DORAI BHAGWAN", 1876543210),

(004, "RISHAB SHETTY", 1789012345);


INSERT INTO MOVIES VALUES(001, "MURDER!", 1930, "ENGLISH", 001),

(002, "THE MAN WHO KNEW TOO MUCH", 1934, "ENGLISH", 001),

(003, "JURASSIC PARK", 1993, "ENGLISH", 002),

(004, "ERADU KANASU", 1974, "KANNADA", 003),

(005, "BAYALU DAARI", 1976, "KANNADA", 003),
```

(006, "SARKARI HIRIYA PRATHAMIKA SHAALE KASARGOD", 2018, "KANNADA", 004);

INSERT INTO MOVIE_CAST VALUES(001, 001, "ACTOR-MANAGER"),

(002, 001, "ACTRESS"),

(003, 001, "STAGE-MANAGER"),

(004, 002, "HERO"),

(005, 002, "VILLAIN"),

(006, 003, "HERO"),

(007, 003, "HEROINE"),

(008, 004, "HERO"),

(010, 004, "HEROINE"),

(009, 005, "HERO"),

(010, 005, "HEROINE"),

(009, 006, "LAWYER");

INSERT INTO RATING VALUES(001, 4),

(001, 5),

(001, 5),

(002, 4),

(002, 4),

(002, 4),

(003, 5),

(003, 5),

(003, 5),

(004, 4),

(004, 4),

(005, 4),

(005, 4),

(006, 5),

(006, 5),

(006, 5);

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

```
SELECT M.MOV_TITLE FROM MOVIES M, DIRECTOR D
WHERE M.DIR_ID = D.DIR_ID AND D.DIR_NAME = "HITCHCOOK";
```

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

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

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

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

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

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

```
SELECT MOV_TITLE , MAX(STARS)
FROM MOVIES
INNER JOIN RATING USING (MOV_ID)
GROUP BY MOV_TITLE
```

HAVING MAX (STARS)>0

ORDER BY MOV_TITLE;

UPDATE RATING

SET STARS=5

WHERE MOV_ID IN (SELECT MOV_ID FROM MOVIES

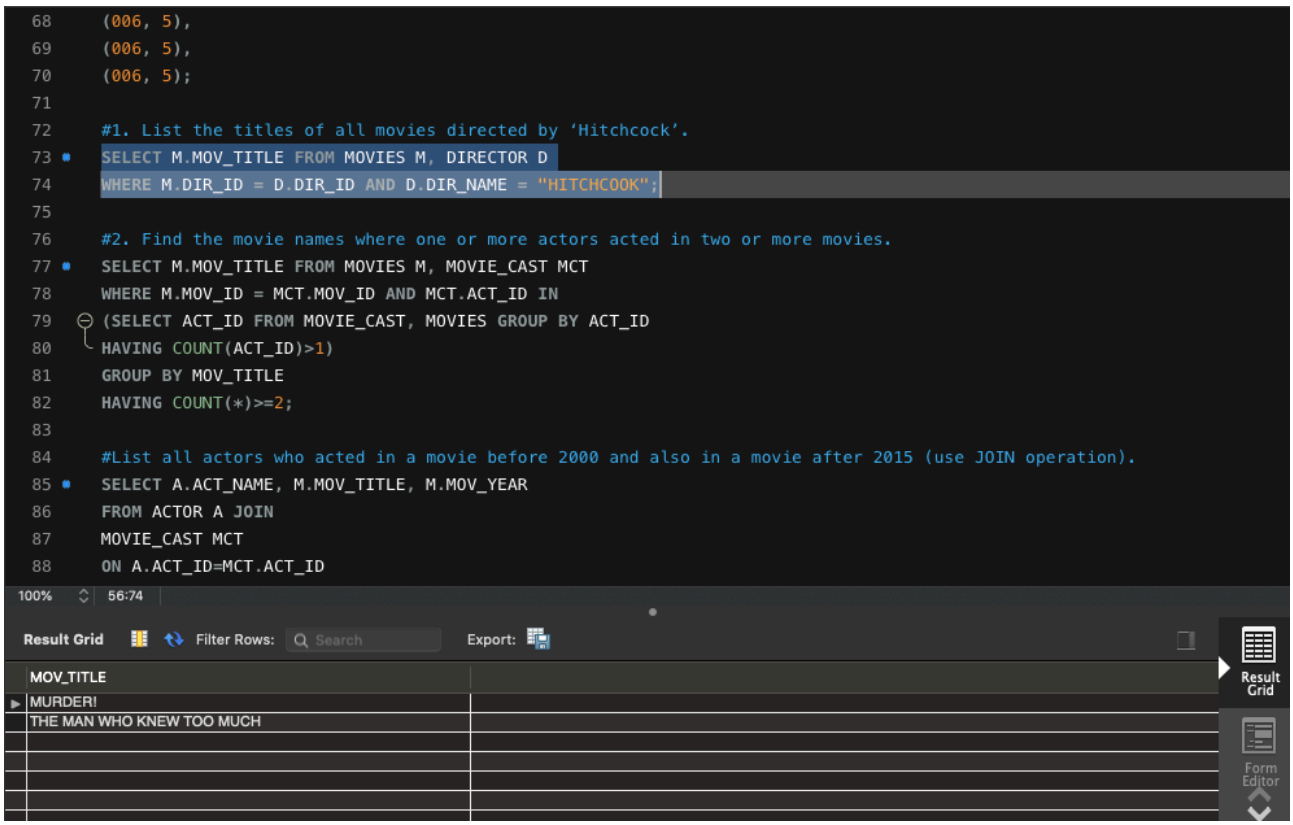
WHERE DIR_ID IN (SELECT DIR_ID

FROM DIRECTOR

WHERE DIR_NAME='STEVEN SPIELBERG'));

SELECT * FROM RATING;

Output:



```
68    (006, 5),
69    (006, 5),
70    (006, 5);
71
72    #1. List the titles of all movies directed by 'Hitchcock'.
73    SELECT M.MOV_TITLE FROM MOVIES M, DIRECTOR D
74    WHERE M.DIR_ID = D.DIR_ID AND D.DIR_NAME = "HITCHCOOK";
75
76    #2. Find the movie names where one or more actors acted in two or more movies.
77    SELECT M.MOV_TITLE FROM MOVIES M, MOVIE_CAST MCT
78    WHERE M.MOV_ID = MCT.MOV_ID AND MCT.ACT_ID IN
79    (SELECT ACT_ID FROM MOVIE_CAST, MOVIES GROUP BY ACT_ID
80    HAVING COUNT(ACT_ID)>1)
81    GROUP BY MOV_TITLE
82    HAVING COUNT(*)>=2;
83
84    #List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).
85    SELECT A.ACT_NAME, M.MOV_TITLE, M.MOV_YEAR
86    FROM ACTOR A JOIN
87    MOVIE_CAST MCT
88    ON A.ACT_ID=MCT.ACT_ID
```

100% 20:82

Result Grid | Filter Rows: Search | Export:

MOV_TITLE
MURDER!
THE MAN WHO KNEW TOO MUCH
JURASSIC PARK
ERADU KANASU
BAYALU DAARI

Result Grid | Form Editor

```
81    GROUP BY MOV_TITLE
82    HAVING COUNT(*)>=2;
83
84    #List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).
85    SELECT A.ACT_NAME, M.MOV_TITLE, M.MOV_YEAR
86    FROM ACTOR A JOIN
87    MOVIE_CAST MCT
88    ON A.ACT_ID=MCT.ACT_ID
89    JOIN MOVIES M
90    ON MCT.MOV_ID=M.MOV_ID
91    WHERE M.MOV_YEAR NOT BETWEEN 2000 AND 2015;
92
93    /*Find the title of movies and number of stars for each movie that has at least one rating and find the highest
94    number of stars that movie received. Sort the result by movie title.*/
95    SELECT MOV_TITLE , MAX(STARS)
96    FROM MOVIES
97    INNER JOIN RATING USING (MOV_ID)
98    GROUP BY MOV_TITLE
99    HAVING MAX (STARS)>0
100   ORDER BY MOV_TITLE;
101
```

100% 44:91

Result Grid | Filter Rows: Search | Export:

MOV_TITLE
MURDER!
THE MAN WHO KNEW TOO MUCH
JURASSIC PARK
ERADU KANASU
BAYALU DAARI

Result Grid | Form Editor

```
86    FROM ACTOR A JOIN
87    MOVIE_CAST MCT
88    ON A.ACT_ID=MCT.ACT_ID
89    JOIN MOVIES M
90    ON MCT.MOV_ID=M.MOV_ID
91    WHERE M.MOV_YEAR NOT BETWEEN 2000 AND 2015;
92
93    /*Find the title of movies and number of stars for each movie that has at least one rating and find the highest
94    number of stars that movie received. Sort the result by movie title.*/
95    SELECT MOV_TITLE , MAX(STARS)
96    FROM MOVIES
97    INNER JOIN RATING USING (MOV_ID)
98    GROUP BY MOV_TITLE
99    HAVING MAX(STARS)>0
100   ORDER BY MOV_TITLE;
101
102
103   UPDATE RATING
104   SET STARS=5
105   WHERE MOV_ID IN (SELECT MOV_ID FROM MOVIES
106   WHERE DIR_ID IN (SELECT DIR_ID
```

100% 1:102

Result Grid Filter Rows: Search Export:

MOV_TITLE	MAX(STARS)
BAYALU DAARI	4
ERADU KANASU	4
JURASSIC PARK	5
MURDER!	5
SARKARI HIRIYA PRATHAMIKA SHAALE KASARGOD	5
THE MAN WHO KNEW TOO MUCH	4

Result Grid Form Editor

```
90    ON MCT.MOV_ID=M.MOV_ID
91    WHERE M.MOV_YEAR NOT BETWEEN 2000 AND 2015;
92
93    /*Find the title of movies and number of stars for each movie that has at least one rating and find the highest
94    number of stars that movie received. Sort the result by movie title.*/
95    SELECT MOV_TITLE , MAX(STARS)
96    FROM MOVIES
97    INNER JOIN RATING USING (MOV_ID)
98    GROUP BY MOV_TITLE
99    HAVING MAX(STARS)>0
100   ORDER BY MOV_TITLE;
101
102
103   UPDATE RATING
104   SET STARS=5
105   WHERE MOV_ID IN (SELECT MOV_ID FROM MOVIES
106   WHERE DIR_ID IN (SELECT DIR_ID
107   FROM DIRECTOR
108   WHERE DIR_NAME='STEVEN SPIELBERG'));
109
110   SELECT * FROM RATING;
```

100% 22:110

Result Grid Filter Rows: Search Export:

MOV_ID	STARS
1	4
1	5
1	5
2	4
2	4
2	4
3	5

Result Grid Form Editor

Experiment 7:

Consider the following database that keeps track of airline flight information:
FLIGHTS (flno: integer, from: string, to: string, distance: integer, departs: time, arrives: time, price: integer)

AIRCRAFT (aid: integer, aname: string, cruisingrange: integer)
CERTIFIED (eid: integer, aid: integer)
EMPLOYEE (eid: integer, ename: string, salary: integer)
Note that the Employees relation describes pilots and other kinds of employees as well; Every pilot is certified for some aircraft, and only pilots are certified to fly.
Write each of the following queries in SQL.

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

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

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

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

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

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

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

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

Code:

```
CREATE DATABASE AIRLINES;

USE AIRLINES;

CREATE TABLE FLIGHTS(

 FLNO INT,

 `FROM` VARCHAR(20),
```

```
 `TO` VARCHAR(20),
 DISTANCE INT,
DEPARTS TIME,
 ARRIVES TIME,
PRICE INT,
 PRIMARY KEY(FLNO) );
 CREATE TABLE AIRCRAFT(
 AID INT,
 ANAME VARCHAR(20),
 CRUISINGRANGE INT,
  PRIMARY KEY (AID) );

CREATE TABLE EMPLOYEES(
 EID INT,
 ENAME VARCHAR(20),
 SALARY INT,
 PRIMARY KEY (EID) );

CREATE TABLE CERTIFIED(
  EID INT,
  AID INT,
  PRIMARY KEY (EID,AID),
 FOREIGN KEY (EID) REFERENCES EMPLOYEES (EID),
FOREIGN KEY (AID) REFERENCES AIRCRAFT (AID) );

INSERT INTO FLIGHTS (FLNO,`FROM`,`TO`,DISTANCE,DEPARTS,ARRIVES,PRICE)
VALUES
 (1,'BANGALORE','CHENNAI',360,'08:45','10:00',10000),
```

```
 (2,'BANGALORE','DELHI',1700,'12:15','15:00',37000),
 (3,'BANGALORE','KOLKATA',1500,'15:15','05:25',30000),
 (4,'MUMBAI','DELHI',1200,'10:30','12:30',28000),
(5,'BANGALORE','NEW YORK',14000,'05:45','02:30',90000),
(6,'DELHI','CHICAGO',12000,'10:00','05:45',95000),
(7,'BANGALORE','FRANKFURT',15000,'12:00','06:30',98000),
 (8,'MADISON','NEW YORK',1500,'10:15','14:25',30000);
SELECT * FROM FLIGHTS;

INSERT INTO AIRCRAFT (AID,ANAME,CRUISINGRANGE) VALUES
      (1,'AIRBUS 380',1000),
      (2,'BOEING 737',4000),
      (3,'LOCKHEED',5500),
      (4,'AIRBUS A220',9500),
      (5,'BOEING 747',800),
      (6,'DOUGLAS DC3',900);

INSERT INTO EMPLOYEES (EID,ENAME,SALARY) VALUES
(1,'ZOYA',95000),
(2,'AKSHAY',65000),
(3,'NIVEDITHA',70000),
(4,'SAFAN',45000),
(5,'PETER',95000),
 (6,'NAYAN',100000),
(7,'AJAY',50000);

INSERT INTO CERTIFIED (EID,AID) VALUES
(1,1),
```

```
(1,3),

(1,4),

(5,4),

(5,3),

(1,2),

 (2,6),

(2,5),

(4,5),

 (6,4),

 (6,3),

 (3,6),

 (3,2);


/* I. 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 DISTINCT A.ANAME

FROM AIRCRAFT A

WHERE A.AID IN (SELECT C.AID

FROM CERTIFIED C, EMPLOYEES E

WHERE C.EID = E.EID AND

NOT EXISTS ( SELECT *

FROM EMPLOYEES E1

WHERE E1.EID = E.EID AND E1.SALARY < 80000 )) ;


/* II. FIND THE NAMES OF PILOTS WHOSE SALARY IS LESS THAN THE PRICE OF THE CHEAPEST ROUTE FROM BENGALURU TO

FRANKFURT.*/

SELECT C.EID, MAX(A.CRUISINGRANGE)
```

```
FROM CERTIFIED C, AIRCRAFT A

WHERE C.AID = A.AID

GROUP BY C.EID

HAVING COUNT(*) > 3;
```

/* III. 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 DISTINCT E.ENAME

    FROM EMPLOYEES E

   WHERE E.SALARY<

   (SELECT MIN(F.PRICE)

   FROM FLIGHTS F

   WHERE F.FROM='BANGALORE'

   AND F.TO='FRANKFURT');
```

/* IV. FIND THE NAMES OF PILOTS CERTIFIED FOR SOME BOEING AIRCRAFT.*/

```
SELECT A.AID,A.ANAME,AVG(E.SALARY)

    FROM AIRCRAFT A,CERTIFIED C,EMPLOYEES E

   WHERE A.AID=C.AID

   AND C.EID=E.EID

   AND A.CRUISINGRANGE>1000

   GROUP BY A.AID,A.ANAME;
```

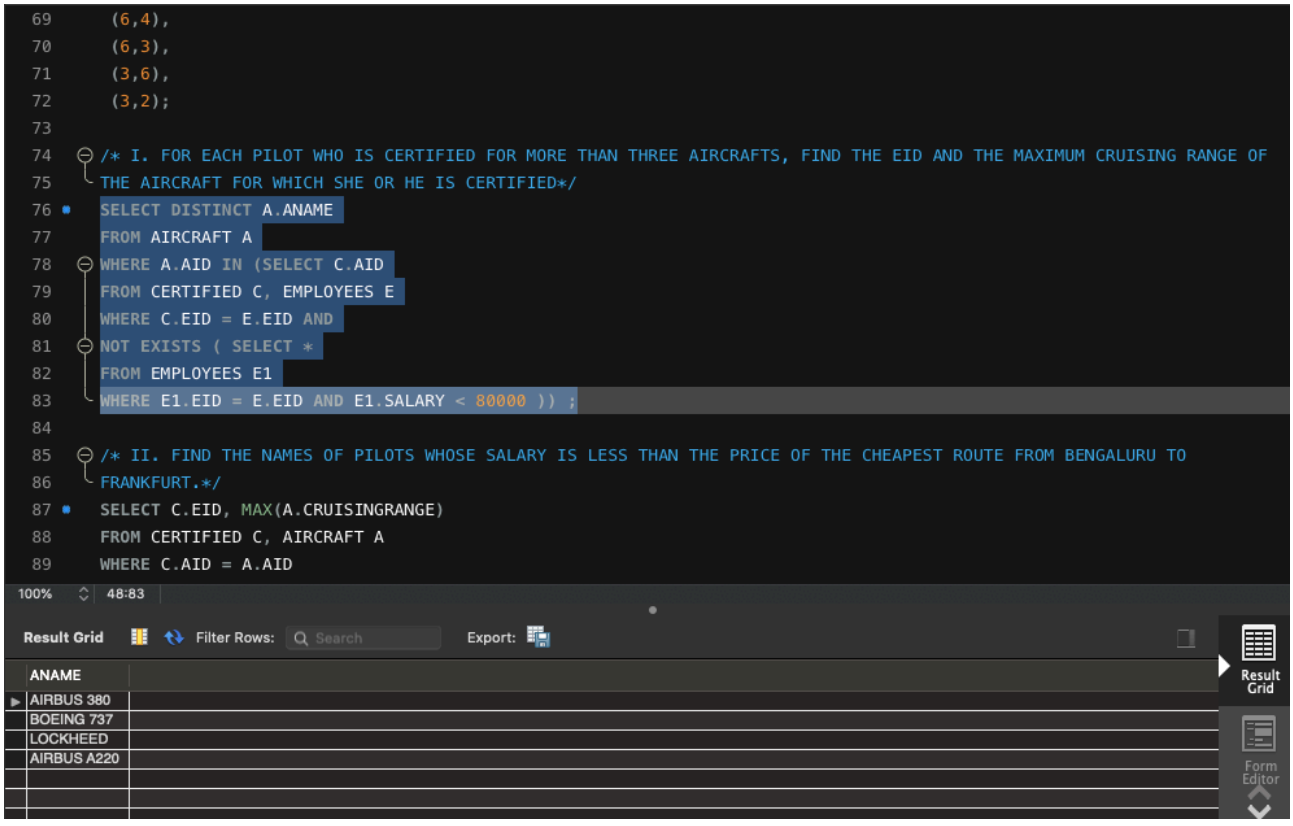
/* V. FIND THE AIDS OF ALL AIRCRAFT THAT CAN BE USED ON ROUTES FROM BENGALURU TO NEW DELHI.*/

```
SELECT DISTINCT E.ENAME
```

```
    FROM EMPLOYEES E,AIRCRAFT A,CERTIFIED C

   WHERE E.EID=C.EID

   AND C.AID=A.AID

  AND A.ANAME LIKE 'BOEING%';


/*VI. 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 A.AID

    FROM AIRCRAFT A

    WHERE A.CRUISINGRANGE>

    (SELECT MIN(F.DISTANCE)

   FROM FLIGHTS F

   WHERE F.FROM='BANGALORE'

   AND F.TO='DELHI');


/* VII. A CUSTOMER WANTS TO TRAVEL FROM MADISON TO NEW YORK WITH NO
MORE THAN TWO CHANGES OF FLIGHT. LIST THE

CHOICE OF DEPARTURE TIMES FROM MADISON IF THE CUSTOMER WANTS TO
ARRIVE IN NEW YORK BY 6 P.M.*/

SELECT F.DEPARTS

FROM FLIGHTS F WHERE F.FLNO IN (  SELECT F0.FLNO

FROM FLIGHTS F0

WHERE F0.FROM = 'MADISON' AND F0.TO = 'NEW YORK' AND F0.ARRIVES < '18:00' );


/* VIII. PRINT THE NAME AND SALARY OF EVERY NON-PILOT WHOSE SALARY IS
MORE THAN THE AVERAGE SALARY FOR PILOTS.

*/
```

```
SELECT E.ENAME, E.SALARY
FROM EMPLOYEES E
WHERE E.EID NOT IN ( SELECT DISTINCT C.EID
FROM CERTIFIED C )
AND E.SALARY > ( SELECT AVG (E1.SALARY)
FROM EMPLOYEES E1
WHERE E1.EID IN
( SELECT DISTINCT C1.EID
FROM CERTIFIED C1 ) );
```

Output:



```
78  WHERE A.AID IN (SELECT C.AID
79  FROM CERTIFIED C, EMPLOYEES E
80  WHERE C.EID = E.EID AND
81  NOT EXISTS ( SELECT *
82  FROM EMPLOYEES E1
83  WHERE E1.EID = E.EID AND E1.SALARY < 80000 )) ;
84
85  /* II. FIND THE NAMES OF PILOTS WHOSE SALARY IS LESS THAN THE PRICE OF THE CHEAPEST ROUTE FROM BENGALURU TO
86  FRANKFURT.*/
87  SELECT C.EID, MAX(A.CRUISINGRANGE)
88  FROM CERTIFIED C, AIRCRAFT A
89  WHERE C.AID = A.AID
90  GROUP BY C.EID
91  HAVING COUNT(*) > 3;
92
93
94  /* III. FOR ALL AIRCRAFT WITH CRUISING RANGE OVER 1000 KMS, FIND THE NAME OF THE AIRCRAFT AND THE AVERAGE SALARY OF
95  ALL PILOTS CERTIFIED FOR THIS AIRCRAFT.*/
96  SELECT DISTINCT E.ENAME
97       FROM EMPLOYEES E
98      WHERE E.SALARY<
```

100% 21:91

Result Grid | Filter Rows: Search | Export:

ANAME
AIRBUS 380
BOEING 737
LOCKHEED
AIRBUS A220

```
86  FRANKFURT.*/
87  SELECT C.EID, MAX(A.CRUISINGRANGE)
88  FROM CERTIFIED C, AIRCRAFT A
89  WHERE C.AID = A.AID
90  GROUP BY C.EID
91  HAVING COUNT(*) > 3;
92
93
94  /* III. FOR ALL AIRCRAFT WITH CRUISING RANGE OVER 1000 KMS, FIND THE NAME OF THE AIRCRAFT AND THE AVERAGE SALARY OF
95  ALL PILOTS CERTIFIED FOR THIS AIRCRAFT.*/
96  SELECT DISTINCT E.ENAME
97       FROM EMPLOYEES E
98      WHERE E.SALARY<
99      (SELECT MIN(F.PRICE)
100     FROM FLIGHTS F
101     WHERE F.FROM='BANGALORE'
102     AND F.TO='FRANKFURT');
103
104
105 /* IV. FIND THE NAMES OF PILOTS CERTIFIED FOR SOME BOEING AIRCRAFT.*/
106 SELECT A.AID,A.ANAME,AVG(E.SALARY)
```

100% 27:102

Result Grid | Filter Rows: Search | Export:

ENAME
ZOYA
AKSHAY
NIVEDITHA
SAFAN
PETER
AJAY

```
95    ALL PILOTS CERTIFIED FOR THIS AIRCRAFT.*/
96    SELECT DISTINCT E.ENAME
97         FROM EMPLOYEES E
98        WHERE E.SALARY<
99        (SELECT MIN(F.PRICE)
100       FROM FLIGHTS F
101       WHERE F.FROM='BANGALORE'
102       AND F.TO='FRANKFURT');
103
104
105   /* IV. FIND THE NAMES OF PILOTS CERTIFIED FOR SOME BOEING AIRCRAFT.*/
106   SELECT A.AID,A.ANAME,AVG(E.SALARY)
107        FROM AIRCRAFT A,CERTIFIED C,EMPLOYEES E
108       WHERE A.AID=C.AID
109       AND C.EID=E.EID
110       AND A.CRUISINGRANGE>1000
111       GROUP BY A.AID,A.ANAME;
112
113   /* V. FIND THE AIDS OF ALL AIRCRAFT THAT CAN BE USED ON ROUTES FROM BENGALURU TO NEW DELHI.*/
114   SELECT DISTINCT E.ENAME
115        FROM EMPLOYEES E,AIRCRAFT A,CERTIFIED C
```

100% 28:111

Result Grid | Filter Rows: Search | Export:

ENAME
ZOYA
AKSHAY
NIVEDITHA
SAFAN
PETER
AJAY

```
110       AND A.CRUISINGRANGE>1000
111       GROUP BY A.AID,A.ANAME;
112
113   /* V. FIND THE AIDS OF ALL AIRCRAFT THAT CAN BE USED ON ROUTES FROM BENGALURU TO NEW DELHI.*/
114   SELECT DISTINCT E.ENAME
115        FROM EMPLOYEES E,AIRCRAFT A,CERTIFIED C
116       WHERE E.EID=C.EID
117       AND C.AID=A.AID
118       AND A.ANAME LIKE 'BOEING%';
119
120   /*VI. A CUSTOMER WANTS TO TRAVEL FROM MADISON TO NEW YORK WITH NO MORE THAN TWO CHANGES OF FLIGHT. LIST THE
121   CHOICE OF DEPARTURE TIMES FROM MADISON IF THE CUSTOMER WANTS TO ARRIVE IN NEW YORK BY 6 P.M.
122   */
123    SELECT A.AID
124        FROM AIRCRAFT A
125        WHERE A.CRUISINGRANGE>
126       (SELECT MIN(F.DISTANCE)
127       FROM FLIGHTS F
128       WHERE F.FROM='BANGALORE'
129       AND F.TO='DELHI');
130
```

100% 31:118

Result Grid | Filter Rows: Search | Export:

ENAME
ZOYA
AKSHAY
NIVEDITHA
SAFAN
PETER
AJAY

```
117         AND C.AID=A.AID
118        AND A.ANAME LIKE 'BOEING%';
119
120  /*VI. A CUSTOMER WANTS TO TRAVEL FROM MADISON TO NEW YORK WITH NO MORE THAN TWO CHANGES OF FLIGHT. LIST THE
121  CHOICE OF DEPARTURE TIMES FROM MADISON IF THE CUSTOMER WANTS TO ARRIVE IN NEW YORK BY 6 P.M.
122  */
123   SELECT A.AID
124       FROM AIRCRAFT A
125       WHERE A.CRUISINGRANGE>
126       (SELECT MIN(F.DISTANCE)
127      FROM FLIGHTS F
128      WHERE F.FROM='BANGALORE'
129      AND F.TO='DELHI');
130
131  /* VII. A CUSTOMER WANTS TO TRAVEL FROM MADISON TO NEW YORK WITH NO MORE THAN TWO CHANGES OF FLIGHT. LIST THE
132  CHOICE OF DEPARTURE TIMES FROM MADISON IF THE CUSTOMER WANTS TO ARRIVE IN NEW YORK BY 6 P.M.*/
133  SELECT F.DEPARTS
134  FROM FLIGHTS F WHERE F.FLNO IN (  SELECT F0.FLNO
135  FROM FLIGHTS F0
136  WHERE F0.FROM = 'MADISON' AND F0.TO = 'NEW YORK' AND F0.ARRIVES < '18:00' );
137
```

100% 23:129

Result Grid | Filter Rows: Search | Export:

ENAME
ZOYA
AKSHAY
NIVEDITHA
SAFAN
PETER
AJAY

Result Grid | Form Editor

```
126       (SELECT MIN(F.DISTANCE)
127      FROM FLIGHTS F
128      WHERE F.FROM='BANGALORE'
129      AND F.TO='DELHI');
130
131  /* VII. A CUSTOMER WANTS TO TRAVEL FROM MADISON TO NEW YORK WITH NO MORE THAN TWO CHANGES OF FLIGHT. LIST THE
132  CHOICE OF DEPARTURE TIMES FROM MADISON IF THE CUSTOMER WANTS TO ARRIVE IN NEW YORK BY 6 P.M.*/
133  SELECT F.DEPARTS
134  FROM FLIGHTS F WHERE F.FLNO IN (  SELECT F0.FLNO
135  FROM FLIGHTS F0
136  WHERE F0.FROM = 'MADISON' AND F0.TO = 'NEW YORK' AND F0.ARRIVES < '18:00' );
137
138  /* VIII. PRINT THE NAME AND SALARY OF EVERY NON-PILOT WHOSE SALARY IS MORE THAN THE AVERAGE SALARY FOR PILOTS.
139  */
140  SELECT E.ENAME, E.SALARY
141  FROM EMPLOYEES E
142  WHERE E.EID NOT IN ( SELECT DISTINCT C.EID
143  FROM CERTIFIED C )
144  AND E.SALARY > ( SELECT AVG (E1.SALARY)
145  FROM EMPLOYEES E1
146  WHERE E1.EID IN
```

100% 77:136

Result Grid | Filter Rows: Search | Export:

ENAME
ZOYA
AKSHAY
NIVEDITHA
SAFAN
PETER
AJAY

Result Grid | Form Editor

```
128         WHERE F.FROM='BANGALORE'
129         AND F.TO='DELHI');
130
131 /* VII. A CUSTOMER WANTS TO TRAVEL FROM MADISON TO NEW YORK WITH NO MORE THAN TWO CHANGES OF FLIGHT. LIST THE
132 CHOICE OF DEPARTURE TIMES FROM MADISON IF THE CUSTOMER WANTS TO ARRIVE IN NEW YORK BY 6 P.M.*/
133 SELECT F.DEPARTS
134 FROM FLIGHTS F WHERE F.FLNO IN (  SELECT F0.FLNO
135 FROM FLIGHTS F0
136 WHERE F0.FROM = 'MADISON' AND F0.TO = 'NEW YORK' AND F0.ARRIVES < '18:00' );
137
138 /* VIII. PRINT THE NAME AND SALARY OF EVERY NON-PILOT WHOSE SALARY IS MORE THAN THE AVERAGE SALARY FOR PILOTS.
139 */
140 SELECT E.ENAME, E.SALARY
141 FROM EMPLOYEES E
142 WHERE E.EID NOT IN ( SELECT DISTINCT C.EID
143 FROM CERTIFIED C )
144 AND E.SALARY > ( SELECT AVG (E1.SALARY)
145 FROM EMPLOYEES E1
146 WHERE E1.EID IN
147 ( SELECT DISTINCT C1.EID
148 FROM CERTIFIED C1 ) );
```

100% 23:148

Result Grid Filter Rows: Search Export:

ENAME	SALARY

Result Grid

Form Editor

Experiment 8:

Consider the schema for College Database:
STUDENT(USN, SName, Address, Phone, Gender)

SEMSEC(SSID, Sem, Sec)
CLASS(USN, SSID)
SUBJECT(Subcode, Title, Sem, Credits)
IAMARKS(USN, Subcode, SSID, Test1, Test2, Test3, FinalIA)

Write SQL queries to

i. List all the student details studying in fourth semester 'C' section.

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

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

iv. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.

v. Categorize students based on the following criterion:
If FinalIA = 17 to 20 then CAT = 'Outstanding'
If FinalIA = 12 to 16 then CAT = 'Average'
If FinalIA< 12 then CAT = 'Weak'
Give these details only for 8th semester A, B, and C section students.

Code:

```
CREATE DATABASE STUDENT_FACULTY;

USE STUDENT_FACULTY;

CREATE TABLE STUDENT (

USN VARCHAR (10) PRIMARY KEY,

SNAME VARCHAR (25),

ADDRESS VARCHAR (25),

PHONE INT (10),

GENDER CHAR (1));

CREATE TABLE SEMSEC (

SSID VARCHAR (5) PRIMARY KEY,

SEM INT (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 (2),
CREDITS INT (2),
PRIMARY KEY (SUBCODE));
CREATE TABLE IAMARKS (
USN VARCHAR (10),
SUBCODE VARCHAR (8),
SSID VARCHAR(5),
TEST1 INT(2),
TEST2 INT(2),
TEST3 INT(2),
FINALIA INT (2),
PRIMARY KEY (USN, SUBCODE, 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',
88778811,'M');
```

```
INSERT INTO STUDENT VALUES('1RN13CS062','SANDHYA','BENGALURU',
77228299,'F');
INSERT INTO STUDENT VALUES('1RN13CS091','TEESHA','BENGALURU',
77123123,'F');
INSERT INTO STUDENT VALUES('1RN13CS066','SUPRIYA','MANGALURU',
88778811,'F');
INSERT INTO STUDENT VALUES('1RN14CS010','ABHAY','BENGALURU',
99002112,'M');
INSERT INTO STUDENT VALUES('1RN14CS032','BHASKAR','BENGALURU',
99232110,'M');
INSERT INTO STUDENT VALUES ('1RN14CS025','ASMI','BENGALURU', 78947373,'F');
INSERT INTO STUDENT VALUES ('1RN15CS011','AJAY','TUMKUR', 98450913,'M');

INSERT INTO STUDENT VALUES ('1RN15CS029','CHITRA','DAVANGERE',
76967721,'F');
INSERT INTO STUDENT VALUES ('1RN15CS045','JEEVA','BELLARY', 99448501,'M');
INSERT INTO STUDENT VALUES ('1RN15CS091','SANTOSH','MANGALURU',
8812332,'M');
INSERT INTO STUDENT VALUES('1RN16CS045','ISMAIL','KALBURGI',
99002322,'M');
INSERT INTO STUDENT VALUES ('1RN16CS088','SAMEERA','SHIMOGA',
99055422,'F');
INSERT INTO STUDENT VALUES ('1RN16CS122','VINAYAKA','CHIKAMAGALUR',
88008800,'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,'C');
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);
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
SS.SSID = C.SSID
GROUP BY SS.SEM, SS.SEC, S.GENDER
ORDER BY SEM;


CREATE VIEW STU_TEST1_MARKS_VIEW
AS
SELECT TEST1, SUBCODE
```

```
FROM IAMARKS

WHERE USN = '1RN13CS091';


-- QUERY 4


DELIMITER //


CREATE PROCEDURE AVG_MARKS()

BEGIN

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;

LOOP

FETCH C_IAMARKS INTO C_A, C_B, C_C, C_USN, C_SUBCODE, C_SSID;

IF (C_A != C_B) THEN
```

```
 SET C_SUM=C_A+C_B;
ELSE
 SET C_SUM=C_A+C_C;
END IF;
SET C_AVG=C_SUM/2;
UPDATE IAMARKS SET FINALIA = C_AVG
WHERE USN = C_USN AND SUBCODE = C_SUBCODE AND SSID = C_SSID;
END LOOP;
CLOSE C_IAMARKS;
END;
//

CALL AVG_MARKS();

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;

Output:

```
110         AND A.CRUISINGRANGE>1000
111         GROUP BY A.AID,A.ANAME;
112
113     /* V. FIND THE AIDS OF ALL AIRCRAFT THAT CAN BE USED ON ROUTES FROM BENGALURU TO NEW DELHI.*/
114     SELECT DISTINCT E.ENAME
115         FROM EMPLOYEES E,AIRCRAFT A,CERTIFIED C
116         WHERE E.EID=C.EID
117         AND C.AID=A.AID
118         AND A.ANAME LIKE 'BOEING%';
119
120     /*VI. A CUSTOMER WANTS TO TRAVEL FROM MADISON TO NEW YORK WITH NO MORE THAN TWO CHANGES OF FLIGHT. LIST THE
121     CHOICE OF DEPARTURE TIMES FROM MADISON IF THE CUSTOMER WANTS TO ARRIVE IN NEW YORK BY 6 P.M.
122     */
123     SELECT A.AID
124         FROM AIRCRAFT A
125         WHERE A.CRUISINGRANGE>
126         (SELECT MIN(F.DISTANCE)
127         FROM FLIGHTS F
128         WHERE F.FROM='BANGALORE'
129         AND F.TO='DELHI');
130
```

100% 31:118

Result Grid | Filter Rows: Search | Export:

ENAME
ZOYA
AKSHAY
NIVEDITHA
SAFAN
PETER
AJAY

```
149     ('1RN13CS091','10CS85','CSE8C', 15, 15,12);
150     SELECT * FROM STUDENT;
151     SELECT * FROM SEMSEC;
152     SELECT * FROM CLASS;
153     SELECT * FROM SUBJECT;
154     SELECT * FROM IAMARKS;
155
156     SELECT S.*, SS.SEM, SS.SEC
157     FROM STUDENT S, SEMSEC SS, CLASS C
158     WHERE S.USN = C.USN AND
159     SS.SSID = C.SSID AND
160     SS.SEM = 4 AND
161     SS.SEc='C';
162
163     SELECT SS.SEM, SS.SEC, S.GENDER, COUNT(S.GENDER) AS COUNT
164     FROM STUDENT S, SEMSEC SS, CLASS C
165     WHERE S.USN = C.USN AND
166     SS.SSID = C.SSID
167     GROUP BY SS.SEM, SS.SEC, S.GENDER
168     ORDER BY SEM;
169
```

100% 13:161

Result Grid | Filter Rows: Search | Export:

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

```
153  SELECT * FROM SUBJECT;
154  SELECT * FROM IAMARKS;
155
156  SELECT S.*, SS.SEM, SS.SEC
157  FROM STUDENT S, SEMSEC SS, CLASS C
158  WHERE S.USN = C.USN AND
159  SS.SSID = C.SSID AND
160  SS.SEM = 4 AND
161  SS.SEc='C';
162
163  SELECT SS.SEM, SS.SEC, S.GENDER, COUNT(S.GENDER) AS COUNT
164  FROM STUDENT S, SEMSEC SS, CLASS C
165  WHERE S.USN = C.USN AND
166  SS.SSID = C.SSID
167  GROUP BY SS.SEM, SS.SEC, S.GENDER
168  ORDER BY SEM;
169
```

100% 14:168

Result Grid | Filter Rows: Search | Export:

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

Result Grid | Form Editor | Field Types

```
166  SS.SSID = C.SSID
167  GROUP BY SS.SEM, SS.SEC, S.GENDER
168  ORDER BY SEM;
169
170
171  CREATE VIEW STU_TEST1_MARKS_VIEW
172  AS
173  SELECT TEST1, SUBCODE
174  FROM IAMARKS
175  WHERE USN = '1RN13CS091';
176
177  SELECT * FROM STU_TEST1_MARKS_VIEW;
178  -- QUERY 4
179
180  DELIMITER //
181
182  CREATE PROCEDURE AVG_MARKS()
183  BEGIN
184  DECLARE C_A INTEGER;
185  DECLARE C_B INTEGER;
186  DECLARE C_C INTEGER;
```

100% 36:177

Result Grid | Filter Rows: Search | Export:

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

Result Grid | Form Editor

```
201     SET C_SUM=C_A+C_B;
202    ELSE
203     SET C_SUM=C_A+C_C;
204    END IF;
205    SET C_AVG=C_SUM/2;
206    UPDATE IAMARKS SET FINALIA = C_AVG
207    WHERE USN = C_USN AND SUBCODE = C_SUBCODE AND SSID = C_SSID;
208    END LOOP;
209    CLOSE C_IAMARKS;
210    END;
211    //
212
213    CALL AVG_MARKS();
214
215    SELECT * FROM IAMARKS;
216
217    SELECT * FROM IAMARKS;
218
219    -- QUERY 5
220
221    SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER,
```

100% 23:215

Result Grid | Filter Rows: Search | Edit: | Export/Import:

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

Result Grid | Form Editor

```
212
213    CALL AVG_MARKS();
214
215    SELECT * FROM IAMARKS;
216
217    SELECT * FROM IAMARKS;
218
219    -- QUERY 5
220
221    SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER,
222    (CASE
223    WHEN IA.FINALIA BETWEEN 17 AND 20 THEN 'OUTSTANDING'
224    WHEN IA.FINALIA BETWEEN 12 AND 16 THEN 'AVERAGE'
225    ELSE 'WEAK'
226    END) AS CAT
227    FROM STUDENT S, SEMSEC SS, IAMARKS IA, SUBJECT SUB
228    WHERE S.USN = IA.USN AND
229    SS.SSID = IA.SSID AND
230    SUB.SUBCODE = IA.SUBCODE AND
231    SUB.SEM = 8;
```

100% 13:231

Result Grid | Filter Rows: Search | Export:

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

Result Grid | Form Editor

Experiment 9:

Consider the following database for student enrolment for course:
STUDENT (snum: integer, sname: string, major: string, level: string, age: integer)
CLASS (name: string, meets at: time, room: string, fid: integer)
ENROLLED (snum: integer, cname: string)
FACULTY (fid: integer, fname: string, deptid: integer)
The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair such that the student is enrolled in the class. Level is a two character code with 4 different values (example: Junior: JR etc)
Write the following queries in SQL. No duplicates should be printed in any of the answers.

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

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

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

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

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

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

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

Code:

```
CREATE DATABASE STUDENT_FACULTY2;

USE STUDENT_FACULTY2;

CREATE TABLE STUDENT(

SNUM INT,

SNAME VARCHAR(10),

MAJOR VARCHAR(2),

LVL VARCHAR(2),

AGE INT, PRIMARY KEY(SNUM));
```

```
CREATE TABLE FACULTY(
FID INT,FNAME VARCHAR(20),
DEPTID INT,
 PRIMARY KEY(FID));

CREATE TABLE CLASS(
CNAME VARCHAR(20),
METTS_AT TIMESTAMP,
ROOM VARCHAR(10),
 FID INT,
PRIMARY KEY(CNAME),
FOREIGN KEY(FID) REFERENCES FACULTY(FID));

CREATE TABLE ENROLLED(
SNUM INT,
CNAME VARCHAR(20),
PRIMARY KEY(SNUM,CNAME),
FOREIGN KEY(SNUM) REFERENCES STUDENT(SNUM),
FOREIGN KEY(CNAME) REFERENCES CLASS(CNAME));

INSERT INTO STUDENT VALUES(1, 'JHON', 'CS', 'SR', 19);
INSERT INTO STUDENT VALUES(2, 'SMITH', 'CS', 'JR', 20);
INSERT INTO STUDENT VALUES(3 , 'JACOB', 'CV', 'SR', 20);
INSERT INTO STUDENT VALUES(4, 'TOM ', 'CS', 'JR', 20);
INSERT INTO STUDENT VALUES(5, 'RAHUL', 'CS', 'JR', 20);
INSERT INTO STUDENT VALUES(6, 'RITA', 'CS', 'SR', 21);
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);
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);
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');
SELECT * FROM ENROLLED;
-- QUERY 1
```

```
SELECT DISTINCT S.SNAME

FROM STUDENT S, CLASS C, ENROLLED E, FACULTY F

WHERE S.SNUM = E.SNUM AND E.CNAME = C.CNAME AND C.FID = F.FID AND

F.FNAME = 'HARISH' AND S.LVL = 'JR';

-- QUERY 2

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

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

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

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
SELECT DISTINCT S.SNAME
FROM STUDENT S
WHERE S.SNUM NOT IN (SELECT E.SNUM
FROM ENROLLED E );
-- QUERY 7
SELECT S.AGE, S.LVL
FROM STUDENT S
GROUP BY S.AGE, S.LVL
HAVING S.LVL IN (SELECT S1.LVL
FROM STUDENT S1
WHERE S1.AGE=S.AGE
GROUP BY S1.AGE, S1.LVL
HAVING COUNT(*) >= ALL (SELECT COUNT(*)
FROM STUDENT S2
WHERE S1.AGE=S2.AGE
GROUP BY S2.LVL, S2.AGE))
ORDER BY S.AGE;
```

Output:

```
57  INSERT INTO ENROLLED VALUES(1, 'CLASS5');
58  INSERT INTO ENROLLED VALUES(2, 'CLASS5');
59  INSERT INTO ENROLLED VALUES(3, 'CLASS5');
60  INSERT INTO ENROLLED VALUES(4, 'CLASS5');
61  INSERT INTO ENROLLED VALUES(5, 'CLASS5');
62  SELECT * FROM ENROLLED;
63  -- QUERY 1
64  SELECT DISTINCT S.SNAME
65  FROM STUDENT S, CLASS C, ENROLLED E, FACULTY F
66  WHERE S.SNUM = E.SNUM AND E.CNAME = C.CNAME AND C.FID = F.FID AND
67  F.FNAME = 'HARISH' AND S.LVL = 'JR';
68  -- QUERY 2
69  SELECT DISTINCT CNAME
70  FROM CLASS
71  WHERE ROOM='ROOM128'
72  OR
73  CNAME IN (SELECT E.CNAME FROM ENROLLED E GROUP BY E.CNAME HAVING COUNT(*)>=5);
74  -- QUERY 3
75  SELECT DISTINCT S.SNAME
76  FROM STUDENT S
77  WHERE S.SNUM IN (SELECT E1.SNUM
```

100% 37:67

Result Grid | Filter Rows: Search | Export:

SNAME
TOM

```
65  FROM STUDENT S, CLASS C, ENROLLED E, FACULTY F
66  WHERE S.SNUM = E.SNUM AND E.CNAME = C.CNAME AND C.FID = F.FID AND
67  F.FNAME = 'HARISH' AND S.LVL = 'JR';
68  -- QUERY 2
69  SELECT DISTINCT CNAME
70  FROM CLASS
71  WHERE ROOM='ROOM128'
72  OR
73  CNAME IN (SELECT E.CNAME FROM ENROLLED E GROUP BY E.CNAME HAVING COUNT(*)>=5);
74  -- QUERY 3
75  SELECT DISTINCT S.SNAME
76  FROM STUDENT S
77  WHERE S.SNUM IN (SELECT E1.SNUM
78  FROM ENROLLED E1, ENROLLED E2, CLASS C1, CLASS C2
79  WHERE E1.SNUM = E2.SNUM AND E1.CNAME <> E2.CNAME
80  AND E1.CNAME = C1.CNAME
81  AND E2.CNAME = C2.CNAME AND C1.METTS_AT = C2.METTS_AT);
82  -- QUERY 4
83  SELECT F.FNAME,F.FID
84  FROM FACULTY F
85  WHERE F.FID IN ( SELECT FID FROM CLASS
```

100% 79:73

Result Grid | Filter Rows: Search | Edit: | Export/Import:

CNAME
CLASS5
NULL

```
65    FROM STUDENT S, CLASS C, ENROLLED E, FACULTY F
66    WHERE S.SNUM = E.SNUM AND E.CNAME = C.CNAME AND C.FID = F.FID AND
67    F.FNAME = 'HARISH' AND S.LVL = 'JR';
68    -- QUERY 2
69    SELECT DISTINCT CNAME
70    FROM CLASS
71    WHERE ROOM='ROOM128'
72    OR
73    CNAME IN (SELECT E.CNAME FROM ENROLLED E GROUP BY E.CNAME HAVING COUNT(*)>=5);
74    -- QUERY 3
75    SELECT DISTINCT S.SNAME
76    FROM STUDENT S
77    WHERE S.SNUM IN (SELECT E1.SNUM
78    FROM ENROLLED E1, ENROLLED E2, CLASS C1, CLASS C2
79    WHERE E1.SNUM = E2.SNUM AND E1.CNAME <> E2.CNAME
80    AND E1.CNAME = C1.CNAME
81    AND E2.CNAME = C2.CNAME AND C1.METTS_AT = C2.METTS_AT);
82    -- QUERY 4
83    SELECT F.FNAME,F.FID
84    FROM FACULTY F
85    WHERE F.FID IN ( SELECT FID FROM CLASS
```

100% 1:82

Result Grid | Filter Rows: Search | Export:

SNAME
RAHUL

```
78    FROM ENROLLED E1, ENROLLED E2, CLASS C1, CLASS C2
79    WHERE E1.SNUM = E2.SNUM AND E1.CNAME <> E2.CNAME
80    AND E1.CNAME = C1.CNAME
81    AND E2.CNAME = C2.CNAME AND C1.METTS_AT = C2.METTS_AT);
82    -- QUERY 4
83    SELECT F.FNAME,F.FID
84    FROM FACULTY F
85    WHERE F.FID IN ( SELECT FID FROM CLASS
86    GROUP BY FID HAVING COUNT(*)=(SELECT COUNT(DISTINCT ROOM) FROM CLASS) );
87    -- QUERY 5
88    SELECT DISTINCT F.FNAME
89    FROM FACULTY F
90    WHERE 5 > (SELECT COUNT(E.SNUM)
91    FROM CLASS C, ENROLLED E
92    WHERE C.CNAME = E.CNAME
93    AND C.FID = F.FID);
94    -- QUERY 6
95    SELECT DISTINCT S.SNAME
96    FROM STUDENT S
97    WHERE S.SNUM NOT IN (SELECT E.SNUM
98    FROM ENROLLED E );
```

100% 73:86

Result Grid | Filter Rows: Search | Edit: | Export/Import:

FNAME	FID
SHIVA	14
NULL	NULL

```
83  SELECT F.FNAME,F.FID
84  FROM FACULTY F
85  WHERE F.FID IN ( SELECT FID FROM CLASS
86  GROUP BY FID HAVING COUNT(*)=(SELECT COUNT(DISTINCT ROOM) FROM CLASS) );
87  -- QUERY 5
88  SELECT DISTINCT F.FNAME
89  FROM FACULTY F
90  WHERE 5 > (SELECT COUNT(E.SNUM)
91  FROM CLASS C, ENROLLED E
92  WHERE C.CNAME = E.CNAME
93  AND C.FID = F.FID);
94  -- QUERY 6
95  SELECT DISTINCT S.SNAME
96  FROM STUDENT S
97  WHERE S.SNUM NOT IN (SELECT E.SNUM
98  FROM ENROLLED E );
99  -- QUERY 7
100 SELECT S.AGE, S.LVL
101 FROM STUDENT S
102 GROUP BY S.AGE, S.LVL
103 HAVING S.LVL IN (SELECT S1.LVL
```

100% 20:93

Result Grid | Filter Rows: Search | Export:

FNAME
HARISH
MV
MIRA
SHIVA

Result Grid

Form Editor

```
83  SELECT F.FNAME,F.FID
84  FROM FACULTY F
85  WHERE F.FID IN ( SELECT FID FROM CLASS
86  GROUP BY FID HAVING COUNT(*)=(SELECT COUNT(DISTINCT ROOM) FROM CLASS) );
87  -- QUERY 5
88  SELECT DISTINCT F.FNAME
89  FROM FACULTY F
90  WHERE 5 > (SELECT COUNT(E.SNUM)
91  FROM CLASS C, ENROLLED E
92  WHERE C.CNAME = E.CNAME
93  AND C.FID = F.FID);
94  -- QUERY 6
95  SELECT DISTINCT S.SNAME
96  FROM STUDENT S
97  WHERE S.SNUM NOT IN (SELECT E.SNUM
98  FROM ENROLLED E );
99  -- QUERY 7
100 SELECT S.AGE, S.LVL
101 FROM STUDENT S
102 GROUP BY S.AGE, S.LVL
103 HAVING S.LVL IN (SELECT S1.LVL
```

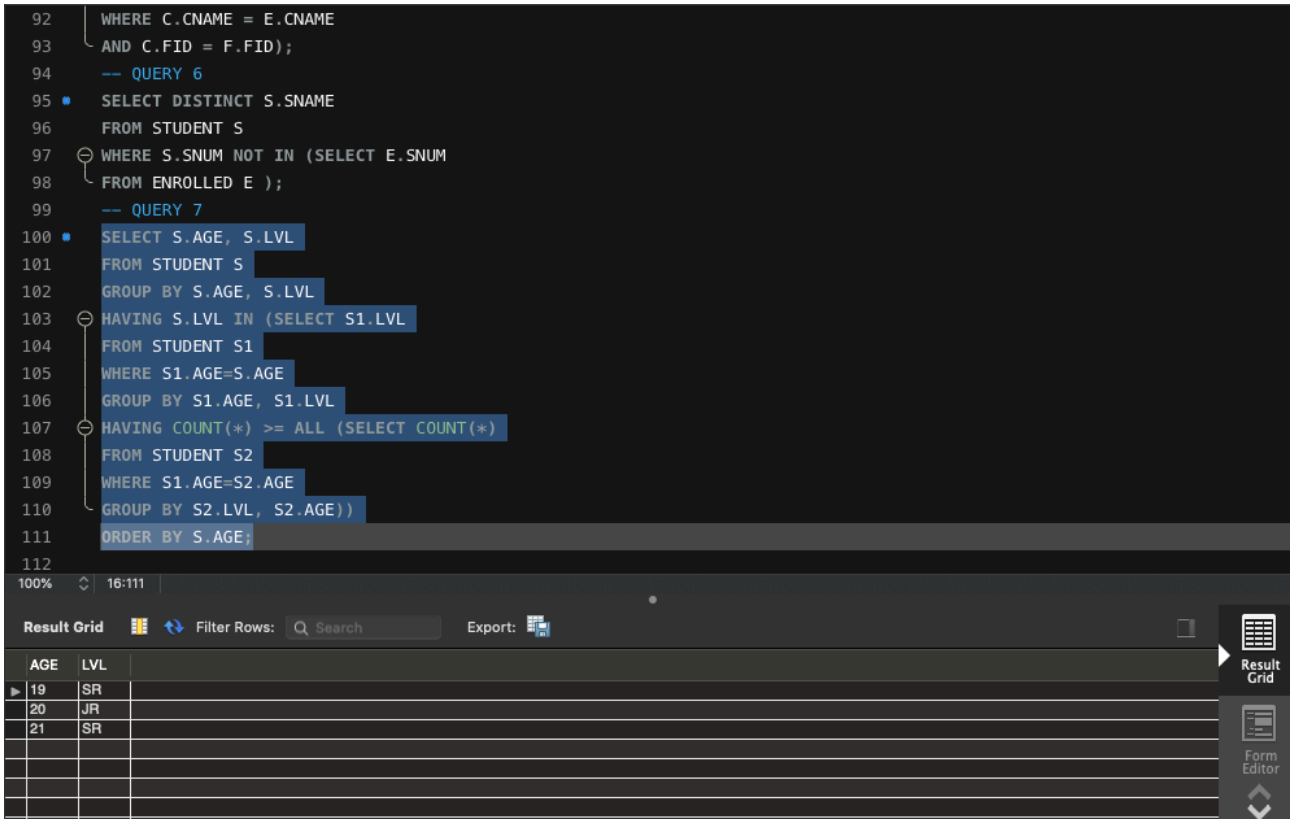
100% 19:98

Result Grid | Filter Rows: Search | Export:

SNAME
RITA

Result Grid

Form Editor


92 WHERE C.CNAME = E.CNAME
93 AND C.FID = F.FID);
94 -- QUERY 6
95 SELECT DISTINCT S.SNAME
96 FROM STUDENT S
97 WHERE S.SNUM NOT IN (SELECT E.SNUM
98 FROM ENROLLED E);
99 -- QUERY 7
100 SELECT S.AGE, S.LVL
101 FROM STUDENT S
102 GROUP BY S.AGE, S.LVL
103 HAVING S.LVL IN (SELECT S1.LVL
104 FROM STUDENT S1
105 WHERE S1.AGE=S.AGE
106 GROUP BY S1.AGE, S1.LVL
107 HAVING COUNT(*) >= ALL (SELECT COUNT(*)
108 FROM STUDENT S2
109 WHERE S1.AGE=S2.AGE
110 GROUP BY S2.LVL, S2.AGE))
111 ORDER BY S.AGE;
112
100%
16:111
Result Grid
Filter Rows:
Search
Export:
AGE
LVL
19
SR
20
JR
21
SR
Result Grid
Form Editor

Experiment 10:

Consider the following schema:
SUPPLIERS (sid: integer, sname: string, address: string)
PARTS (pid: integer, pname: string, color: string)
CATALOG (sid: integer, pid: integer, cost: real)
The Catalog relation lists the prices charged for parts by Suppliers. Write the following queries in SQL:

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

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

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

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

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

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

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

Code:

```
CREATE DATABASE SUPPLIER;

USE SUPPLIER;

CREATE TABLE SUPPLIERS(SID BIGINT(5) PRIMARY KEY, SNAME

VARCHAR(20), CITY VARCHAR(20));

INSERT INTO SUPPLIERS VALUES(10001,'ACME WIDGET','BANGALORE');

INSERT INTO SUPPLIERS VALUES(10002,'JOHNS ','KOLKATA');

INSERT INTO SUPPLIERS VALUES(10003,'VIMAL','MUMBAI');

INSERT INTO SUPPLIERS VALUES(10004,'RELIANCE ','DELHI');

SELECT * FROM SUPPLIERS;

CREATE TABLE PARTS(PID BIGINT(5) PRIMARY KEY, PNAME VARCHAR(20),

COLOR VARCHAR(10));

INSERT INTO PARTS VALUES(20001,'BOOK','RED');

INSERT INTO PARTS VALUES(20002,'PEN','RED');
```

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

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

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

SELECT S.SNAME FROM SUPPLIERS S WHERE NOT EXISTS (SELECT P.PID FROM

PARTS P WHERE P.COLOR = 'RED' AND (NOT EXISTS (SELECT C.SID FROM

CATALOG C WHERE C.SID = S.SID AND C.PID = P.PID)));


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

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

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

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

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


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

ALL THE SUPPLIERS WHO SUPPLY THAT PART).

 */

SELECT DISTINCT C.SID FROM CATALOG C

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

FROM CATALOG C1

WHERE C1.PID = C.PID );


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

SELECT P.PID, S.SNAME

FROM PARTS P, SUPPLIERS S, CATALOG C

WHERE C.PID = P.PID

AND C.SID = S.SID

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

FROM CATALOG C1

WHERE C1.PID = P.PID);

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

SELECT DISTINCT C.SID

FROM CATALOG C

WHERE NOT EXISTS (SELECT *

FROM PARTS P

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

Output:

```
26  INSERT INTO CATALOG VALUES(10002,20001,10);
27  INSERT INTO CATALOG VALUES(10002,20002,20);
28  INSERT INTO CATALOG VALUES(10003,20003,30);
29  INSERT INTO CATALOG VALUES(10004,20003,40);
30  SELECT * FROM CATALOG;
31  /* 1 - FIND THE PNAMES OF PARTS FOR WHICH THERE IS SOME SUPPLIER. */
32  SELECT DISTINCT P.PNAME
33   FROM PARTS P, CATALOG C
34   WHERE P.PID = C.PID;
35
36  /* FIND THE SNAMES OF SUPPLIERS WHO SUPPLY EVERY PART */
37  SELECT S.SNAME FROM SUPPLIERS S WHERE NOT EXISTS (SELECT P.PID FROM
38  PARTS P WHERE NOT EXISTS (SELECT C.SID FROM CATALOG C WHERE C.SID =
39  S.SID AND C.PID = P.PID));
40
41  /* FIND THE SNAMES OF SUPPLIERS WHO SUPPLY EVERY RED PART. */
42  SELECT S.SNAME FROM SUPPLIERS S WHERE NOT EXISTS (SELECT P.PID FROM
43  PARTS P WHERE P.COLOR = 'RED' AND (NOT EXISTS (SELECT C.SID FROM
44  CATALOG C WHERE C.SID = S.SID AND C.PID = P.PID)));
45
46  /* FIND THE PNAMES OF PARTS SUPPLIED BY ACME WIDGET SUPPLIERS AND BY NO ONE ELSE */
47  SELECT P.PNAME FROM PARTS P, CATALOG C, SUPPLIERS S WHERE P.PID
48  = C.PID AND C.SID = S.SID AND S.SNAME = 'ACME WIDGET' AND NOT EXISTS
49  (SELECT * FROM CATALOG C1, SUPPLIERS S1 WHERE P.PID = C1.PID AND
50  C1.SID = S1.SID AND S1.SNAME <> 'ACME WIDGET');
51
52  /* FIND THE SIDS OF SUPPLIERS WHO CHARGE MORE FOR SOME PART THAN THE AVERAGE COST OF THAT PART (AVERAGED OVER
53  ALL THE SUPPLIERS WHO SUPPLY THAT PART).
54   */
55  SELECT DISTINCT C.SID FROM CATALOG C
```

```
/* 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');
```

100% 27:39

Result Grid | Filter Rows: Search | Export:

SNAME
ACME WIDGET

Result Grid | Form Editor

```
/* 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)
```

100% 52:44

Result Grid | Filter Rows: Search | Export:

SNAME
ACME WIDGET

Result Grid | Form Editor

```sql
/* 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
```

100% 48:50

Result Grid | Filter Rows: Search | Export:

PNAME
MOBILE
CHARGER

Result Grid | Form Editor

```sql
/* 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
```

100% 24:58

Result Grid | Filter Rows: Search | Export:

SID
10002
10004

Result Grid | Form Editor

```
48      = C.PID AND C.SID = S.SID AND S.SNAME = 'ACME WIDGET' AND NOT EXISTS
49    (SELECT * FROM CATALOG C1, SUPPLIERS S1 WHERE P.PID = C1.PID AND
50    C1.SID = S1.SID AND S1.SNAME <> 'ACME WIDGET');
51
52    /* FIND THE SIDS OF SUPPLIERS WHO CHARGE MORE FOR SOME PART THAN THE AVERAGE COST OF THAT PART (AVERAGED OVER
53    ALL THE SUPPLIERS WHO SUPPLY THAT PART).
54     */
55    SELECT DISTINCT C.SID FROM CATALOG C
56    WHERE C.COST > ( SELECT AVG (C1.COST)
57    FROM CATALOG C1
58    WHERE C1.PID = C.PID );
59
60    /* FOR EACH PART, FIND THE SNAME OF THE SUPPLIER WHO CHARGES THE MOST FOR THAT PART.*/
61    SELECT P.PID, S.SNAME
62    FROM PARTS P, SUPPLIERS S, CATALOG C
63    WHERE C.PID = P.PID
64    AND C.SID = S.SID
65    AND C.COST = (SELECT MAX(C1.COST)
66    FROM CATALOG C1
67    WHERE C1.PID = P.PID);
68
```

100% 23:67

Result Grid | Filter Rows: Search | Export:

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

Result Grid | Form Editor

```
56    WHERE C.COST > ( SELECT AVG (C1.COST)
57    FROM CATALOG C1
58    WHERE C1.PID = C.PID );
59
60    /* FOR EACH PART, FIND THE SNAME OF THE SUPPLIER WHO CHARGES THE MOST FOR THAT PART.*/
61    SELECT P.PID, S.SNAME
62    FROM PARTS P, SUPPLIERS S, CATALOG C
63    WHERE C.PID = P.PID
64    AND C.SID = S.SID
65    AND C.COST = (SELECT MAX(C1.COST)
66    FROM CATALOG C1
67    WHERE C1.PID = P.PID);
68
69    /* FIND THE SIDS OF SUPPLIERS WHO SUPPLY ONLY RED PARTS.*/
70    SELECT DISTINCT C.SID
71    FROM CATALOG C
72    WHERE NOT EXISTS ( SELECT *
73    FROM PARTS P
74    WHERE P.PID = C.PID AND P.COLOR <> 'RED' );
75
```

100% 1:75

Result Grid | Filter Rows: Search | Export:

SID
10001
10002

Result Grid | Form Editor

****************************END OF DBMS LAB PROGRAMS************************