



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

**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING**

**DATABASE MANAGEMENT SYSTEM
LABORATORY RECORD**

**NAME: Sushmitha Y V
USN:1BM19CS165**

PROGRAM: BACHELOR OF ENGINEERING

SEMESTER: IV

SESSION: APR-JUL 2021

COURSE CODE: 19CS4PCDBM

COURSE TITLE: DATABASE MANAGEMENT SYSTEM

CREDITS: 4

DBMS Lab List

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

DBMS LAB RECORD

(ALL 10 PROGRAMS)

PROGRAM 1: INSURANCE DATABASE

QUESTION:

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.

PROGRAM CODE:

```
create database Insurance;
```

```
use Insurance;
```

```
CREATE TABLE PERSON(DRIVER_ID VARCHAR(10),NAME VARCHAR(20),ADDRESS  
VARCHAR(15),PRIMARY KEY(DRIVER_ID));
```

```
show tables;
```

```
desc PERSON;
```

```
SELECT *FROM PERSON;
```

```
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);

show tables;

insert into PERSON(DRIVER_ID,NAME,ADDRESS)values('1111','RAMU', 'K.S.LAYOUT');

insert into PERSON(DRIVER_ID,NAME,ADDRESS)values('2222','JOHN', 'INDIRANAGAR');

insert into PERSON(DRIVER_ID,NAME,ADDRESS)values('3333','PRIYA', 'JAYANAGAR');

insert into PERSON(DRIVER_ID,NAME,ADDRESS)values('4444','GOPAL', 'WHITEFIELD');

insert into PERSON(DRIVER_ID,NAME,ADDRESS)values('5555','LATHA', ' VIJAYANAGAR');

COMMIT;

desc PERSON;

SELECT *FROM PERSON;

insert into car(regno,Model,Year)values('KA04Q2301','MARUTHI-DX', '2000-10-11');

insert into car(regno,Model,Year)values('KA05P1000',' FORDICON','2000-09-08');

insert into car(regno,Model,Year)values('KA03L1234','ZEN-VXI', '1999-07-06');

insert into car(regno,Model,Year)values('KA03L9999',' MARUTH-DX', '2002-06-05');

insert into car(regno,Model,Year)values('KA01P4020',' INDICA-VX', '2002-05-04');

COMMIT;

desc car;

SELECT *FROM car;

insert into Accident(report_no,ADATE,Location)values('12',' 2002-06-02',' M G ROAD');

insert into Accident(report_no,ADATE,Location)values('200',' 2002-12-10',' DOUBLEROAD');

insert into Accident(report_no,ADATE,Location)values('300',' 1999-07-10','M G ROAD');

insert into Accident(report_no,ADATE,Location)values('25000',' 2000-06-11',' RESIDENCY ROAD');

insert into Accident(report_no,ADATE,Location)values('26500',' 2001-08-12',' RICHMOND ROAD');

COMMIT;

desc Accident;

```

```

SELECT *FROM Accident;

insert into owns(driver_id,regno)values('1111', 'KA04Q2301');

insert into owns(driver_id,regno)values('1111','KA05P1000');

insert into owns(driver_id,regno)values('2222','KA03L1234');

insert into owns(driver_id,regno)values('3333','KA03L9999');

insert into owns(driver_id,regno)values('4444','KA01P4020');

COMMIT;

desc owns;

SELECT *FROM owns;

insert into PARTICIPATED(driver_id,regno,report_no,damage_amt)values('1111', 'KA04Q2301',' 12','20000');

insert into PARTICIPATED(driver_id,regno,report_no,damage_amt)values('2222','KA03L1234','200','500');

insert into PARTICIPATED(driver_id,regno,report_no,damage_amt)values('3333','KA03L9999','300','10000');

insert into PARTICIPATED(driver_id,regno,report_no,damage_amt)values('4444','KA01P4020','25000','2375');

insert into
PARTICIPATED(driver_id,regno,report_no,damage_amt)values('1111','KA05P1000','26500','70000');

COMMIT;

desc PARTICIPATED ;

SELECT *FROM PARTICIPATED;

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

*/
UPDATE PARTICIPATED SET DAMAGE_AMT=25000 WHERE REPORT_NO =12 AND
REGNO='KA04Q2301';

COMMIT;

desc PARTICIPATED ;

SELECT *FROM PARTICIPATED;

/*

```

b. Add a new accident to the database

```
*/
```

```
insert into Accident(report_no,ADATE,Location)values('500','2005-06-02','Mysore Road');
```

```
desc Accident;
```

```
SELECT *FROM Accident;
```

```
/*
```

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

```
*/
```

```
select count(*) from Accident where year(ADATE)=2008;
```

```
/*
```

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

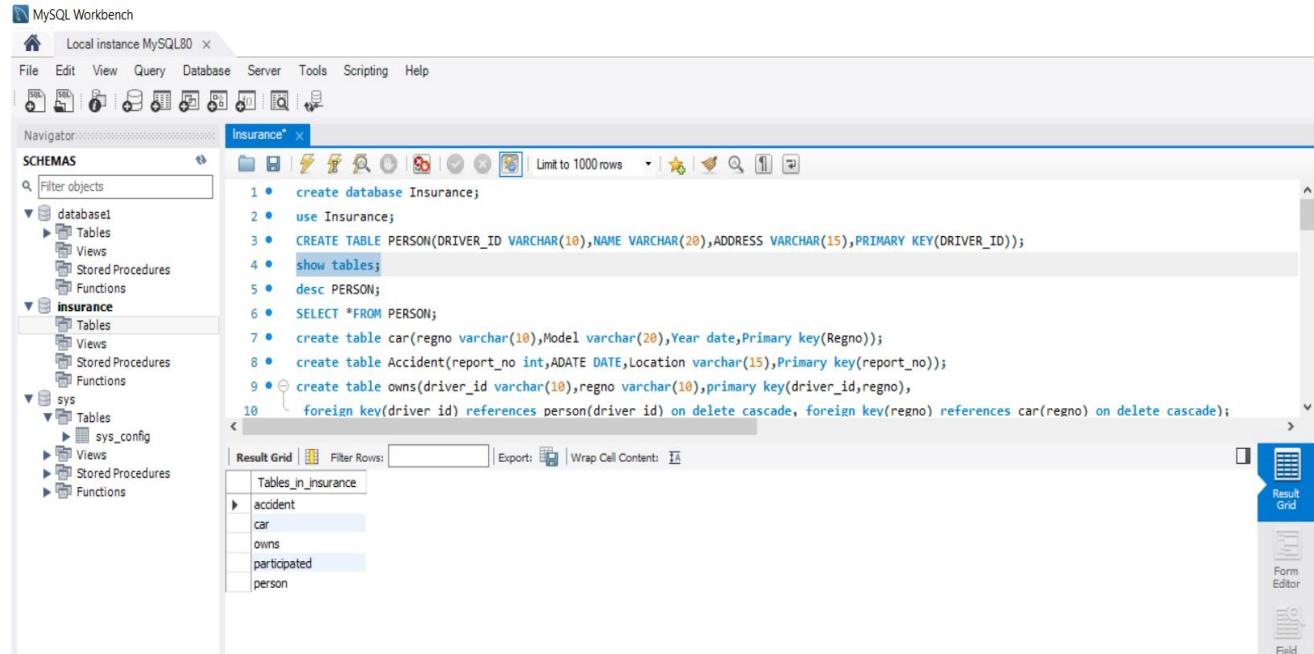
```
*/
```

```
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';
```

SCREENSHOTS OF OUTPUT:



The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar has a Navigator tab and a Schemas section showing databases like database1 and insurance. The main area has an 'Insurance' tab selected. The SQL editor contains the following code:

```
1 • create database Insurance;
2 • use Insurance;
3 • CREATE TABLE PERSON(DRIVER_ID VARCHAR(10),NAME VARCHAR(20),ADDRESS VARCHAR(15),PRIMARY KEY(DRIVER_ID));
4 • show tables;
5 • desc PERSON;
6 • SELECT *FROM PERSON;
7 • create table car(regno varchar(10),Model varchar(20),Year date,Primary key(Regno));
8 • create table Accident(report_no int,ADATE DATE,Location varchar(15),Primary key(report_no));
9 • create table owns(driver_id varchar(10),regno varchar(10),primary key(driver_id,regno),
10   foreign key(driver_id) references person(driver_id) on delete cascade, foreign key(regno) references car(regno) on delete cascade);
```

The results grid below the editor shows a table named 'Tables_in_insurance' with four rows: accident, car, owns, and participated.

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance*

SCHEMAS

- Filter objects
- database1
 - Tables
 - Views
 - Stored Procedures
 - Functions
- insurance
 - Tables
 - Views
 - Stored Procedures
 - Functions
- sys
 - Tables
 - sys_config
 - Views
 - Stored Procedures
 - Functions

Script Editor

```

16 • insert into PERSON(DRIVER_ID,NAME,ADDRESS)values('2222','JOHN', 'INDIRANAGAR');
17 • insert into PERSON(DRIVER_ID,NAME,ADDRESS)values('3333','PRIYA', 'JAYANAGAR');
18 • insert into PERSON(DRIVER_ID,NAME,ADDRESS)values('4444','GOPAL', 'WHITEFIELD');
19 • insert into PERSON(DRIVER_ID,NAME,ADDRESS)values('5555','LATHA', 'VIJAYANAGAR');
20 • COMMIT;
21 • desc PERSON;
22 • SELECT *FROM PERSON;
23 • insert into car(regno,Model,Year)values('KA04Q2301', 'MARUTHI-DX', '2000-10-11');
24 • insert into car(regno,Model,Year)values('KA05P1000', 'FORDICON', '2000-09-08');
25 • insert into car(regno,Model,Year)values('KA03L1234', 'ZEN-VXI', '1999-07-06');

```

Result Grid

DRIVER_ID	NAME	ADDRESS
1111	RAMU	K.S.LAYOUT
2222	JOHN	INDIRANAGAR
3333	PRIYA	JAYANAGAR
4444	GOPAL	WHITEFIELD
5555	LATHA	VIJAYANAGAR
NULL	NULL	NULL

Right Panel

- Result Grid
- Form Editor
- Field Types

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance*

SCHEMAS

- Filter objects
- database1
 - Tables
 - Views
 - Stored Procedures
 - Functions
- insurance
 - Tables
 - Views
 - Stored Procedures
 - Functions
- sys
 - Tables
 - sys_config
 - Views
 - Stored Procedures
 - Functions

Script Editor

```

28 • COMMIT;
29 • desc car;
30 • SELECT *FROM car;
31 • insert into Accident(report_no,ADATE,Location)values('12', ' 2002-06-02', ' M G ROAD');
32 • insert into Accident(report_no,ADATE,Location)values('200', ' 2002-12-10', ' DOUBLEROAD');
33 • insert into Accident(report_no,ADATE,Location)values('300', ' 1999-07-10', 'M G ROAD');
34 • insert into Accident(report_no,ADATE,Location)values('25000', ' 2000-06-11', ' RESIDENCY ROAD');
35 • insert into Accident(report_no,ADATE,Location)values('26500', ' 2001-08-12', ' RICHMOND ROAD');
36 • COMMIT;
37 • desc Accident;

```

Result Grid

regno	Model	Year
KA01P4020	INDICA-VX	2002-05-04
KA03L1234	ZEN-VXI	1999-07-06
KA03L9999	MARUTHI-DX	2002-06-05
KA04Q2301	MARUTHI-DX	2000-10-11
KA05P1000	FORDICON	2000-09-08
NULL	NULL	NULL

Right Panel

- Result Grid
- Form Editor
- Field Types

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance*

SCHEMAS

database1

insurance

sys

Result Grid

Report No. ADATE Location

report_no	ADATE	Location
12	2002-06-02	M G ROAD
200	2002-12-10	DOUBLEROAD
300	1999-07-10	M G ROAD
25000	2000-06-11	RESIDENCY ROAD
26500	2001-08-12	RICHMOND ROAD
*	NULL	NULL

Form Editor

Field Types

```
37 • desc Accident;
38 • SELECT *FROM Accident;
39 • insert into owns(driver_id,regno)values('1111','KA04Q2301');
40 • insert into owns(driver_id,regno)values('1111','KA05P1000');
41 • insert into owns(driver_id,regno)values('2222','KA03L1234');
42 • insert into owns(driver_id,regno)values('3333','KA03L9999');
43 • insert into owns(driver_id,regno)values('4444','KA01P4020');
44 • COMMIT;
45 • desc owns;
46 • SELECT *FROM owns;
```

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance*

SCHEMAS

database1

insurance

sys

Result Grid

Driver ID Regno

driver_id	regno
4444	KA01P4020
2222	KA03L1234
3333	KA03L9999
1111	KA04Q2301
1111	KA05P1000
*	NULL

Form Editor

Field Types

```
38 • SELECT *FROM Accident;
39 • insert into owns(driver_id,regno)values('1111','KA04Q2301');
40 • insert into owns(driver_id,regno)values('1111','KA05P1000');
41 • insert into owns(driver_id,regno)values('2222','KA03L1234');
42 • insert into owns(driver_id,regno)values('3333','KA03L9999');
43 • insert into owns(driver_id,regno)values('4444','KA01P4020');
44 • COMMIT;
45 • desc owns;
46 • SELECT *FROM owns;
47 • insert into PARTICIPATED(driver_id,regno,report_no,damage_amt)values('1111','KA04Q2301','12','2000');
```

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance*

SCHEMAS

- database1
 - Tables
 - Views
 - Stored Procedures
 - Functions
- insurance
 - Tables
 - Views
 - Stored Procedures
 - Functions
- sys
 - Tables
 - sys_config
 - Views
 - Stored Procedures
 - Functions

Limit to 1000 rows

```

46 • SELECT *FROM owns;
47 • insert into PARTICIPATED(driver_id,regno,report_no,damage_amt)values('1111','KA04Q2301','12','20000');
48 • insert into PARTICIPATED(driver_id,regno,report_no,damage_amt)values('2222','KA03L1234','200','500');
49 • insert into PARTICIPATED(driver_id,regno,report_no,damage_amt)values('3333','KA03L9999','300','10000');
50 • insert into PARTICIPATED(driver_id,regno,report_no,damage_amt)values('4444','KA01P4020','25000','2375');
51 • insert into PARTICIPATED(driver_id,regno,report_no,damage_amt)values('1111','KA05P1000','26500','70000');
52 • COMMIT;
53 • desc PARTICIPATED ;
54 • SELECT *FROM PARTICIPATED;
55

```

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

driver_id	regno	report_no	damage_amt
1111	KA04Q2301	12	20000
2222	KA03L1234	200	500
3333	KA03L9999	300	10000
4444	KA01P4020	25000	2375
1111	KA05P1000	26500	70000

Result Grid Form Editor Field Types

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance*

SCHEMAS

- database1
 - Tables
 - Views
 - Stored Procedures
 - Functions
- insurance
 - Tables
 - Views
 - Stored Procedures
 - Functions
- sys
 - Tables
 - sys_config
 - Views
 - Stored Procedures
 - Functions

Limit to 1000 rows

```

50 • insert into PARTICIPATED(driver_id,regno,report_no,damage_amt)values('4444','KA01P4020','25000','2375');
51 • insert into PARTICIPATED(driver_id,regno,report_no,damage_amt)values('1111','KA05P1000','26500','70000');
52 • COMMIT;
53 • desc PARTICIPATED ;
54 • SELECT *FROM PARTICIPATED;
55 • UPDATE PARTICIPATED SET DAMAGE_AMT=25000 WHERE REPORT_NO =12 AND REGNO='KA04Q2301';
56 • COMMIT;
57 • desc PARTICIPATED ;
58 • SELECT *FROM PARTICIPATED;
59

```

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

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 Field Types

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance x

SCHEMAS

Filter objects

- database1
 - Tables
 - Views
 - Stored Procedures
 - Functions
- insurance
 - Tables
 - Views
 - Stored Procedures
 - Functions
- sys
 - Tables
 - sys_config
 - Views
 - Stored Procedures
 - Functions

Limit to 1000 rows

```

59 • UPDATE PARTICIPATED SET DAMAGE_AMT=25000 WHERE REPORT_NO =12 AND REGNO='KA04Q2301';
60 • COMMIT;
61 • desc PARTICIPATED ;
62 • SELECT *FROM PARTICIPATED;
63 • /*
64 • b. Add a new accident to the database
65 */
66 • insert into Accident(report_no,ADATE,Location)values('500',' 2005-06-02','Mysore Road');
67 • desc Accident;
68 • SELECT *FROM Accident;
69

```

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

	report_no	ADATE	Location
▶	12	2002-06-02	M G ROAD
	200	2002-12-10	DOUBLEROAD
	300	1999-07-10	M G ROAD
	500	2005-06-02	Mysore Road
	25000	2000-06-11	RESIDENCY ROAD
	26500	2001-08-12	RICHMOND ROAD
●	NULL	NULL	NULL

Result Grid | Form Editor

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance x

SCHEMAS

Filter objects

- database1
 - Tables
 - Views
 - Stored Procedures
 - Functions
- insurance
 - Tables
 - Views
 - Stored Procedures
 - Functions
- sys
 - Tables
 - sys_config
 - Views
 - Stored Procedures
 - Functions

Limit to 1000 rows

```

66 • insert into Accident(report_no,ADATE,Location)values('500',' 2005-06-02','Mysore Road');
67 • desc Accident;
68 • SELECT *FROM Accident;
69
70 • /*
71 • iv. Find the total number of people who owned cars that involved in accidents in 2008
72 */
73 • select count(*) from Accident where year(ADATE)=2008;
74
75 • /*
76 • V. Find the number of accidents in which cars belonging to a specific model were involved

```

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

	count(*)
▶	0

Result Grid | Form Editor

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Navigator' with 'SCHEMAS' expanded, showing 'database1' and 'insurance'. The main area is titled 'Insurance' and contains a query editor with the following SQL code:

```

70  /*
71   iv. Find the total number of people who owned cars that involved in accidents in 2008
72   */
73 • select count(*) from Accident where year(ADATE)=2008;
74
75  /*
76   V. Find the number of accidents in which cars belonging to a specific model were involved
77  */
78 • 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';

```

The result grid shows the output of the last query:

COUNT(A.REPORT_NO)
1

PROGRAM-2:BOOK DEALER DATABASE

QUESTION:

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%.

PROGRAM CODE:

```
create database bookdealer;
use bookdealer;
create table AUTHOR (
author_id int,
name varchar(20),
city varchar(15),
country varchar(15),
primary key(author_id)
);
show tables;
desc AUTHOR;
SELECT *FROM AUTHOR;
create table PUBLISHER (
publisher_id int,
name varchar(20),
city varchar(15),
country varchar(15),
primary key(publisher_id)
);
create table CATEGORY(
category_id int,
description varchar(20),
primary key(category_id)
);
show tables;
desc CATEGORY;
```

```
SELECT *FROM CATEGORY;

create table CATALOG (
    book_id int,
    title varchar(15),author_id int,publisher_id int,category_id 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,
    year int,
    price int,
    primary key(book_id)
);

show tables;

desc CATALOG;

SELECT *FROM CATALOG;

create table ORDER_DETAILS (
    order_no int,book_id int,
    foreign key(book_id) references CATALOG(book_id) on delete cascade,
    quantity int
);

show tables;

desc ORDER_DETAILS;

SELECT *FROM ORDER_DETAILS;

insert into AUTHOR(author_id,name,city,country)values(1001,'TERAS CHAN','CA','USA');

insert into
AUTHOR(author_id,name,city,country)values(1002,'STEVENS','ZOMBI','UGANDA');

insert into AUTHOR(author_id,name,city,country)values(1003,'M MANO','CAIR','CANADA');

insert into AUTHOR(author_id,name,city,country)values(1004,'KARTHIK B.P','NEW
YORK','USA');

insert into AUTHOR(author_id,name,city,country)values(1005,'WILLIAM STALLINGS','LAS
VEGAS','USA');
```

```
COMMIT;

desc AUTHOR;

SELECT *FROM AUTHOR;

insert into PUBLISHER(publisher_id,name,city,country)values(1,'PEARSON','NEW YORK','USA');

insert into PUBLISHER(publisher_id,name,city,country)values(2,'EEE','NEW SOUTH VALES','USA');

insert into PUBLISHER(publisher_id,name,city,country)values(3,'PHI','DELHI','INDIA');

insert into
PUBLISHER(publisher_id,name,city,country)values(4,'WILLEY','BERLIN','GERMANY');

insert into PUBLISHER(publisher_id,name,city,country)values(5,'MGH ','NEW YORK','USA');

COMMIT;

desc PUBLISHER;

SELECT *FROM PUBLISHER;

insert into CATEGORY(category_id,description)values(1001,'COMPUTER SCIENCE');

insert into CATEGORY(category_id,description)values(1002,'ALGORITHM DESIGN');

insert into CATEGORY(category_id,description)values(1003,'ELECTRONICS');

insert into CATEGORY(category_id,description)values(1004,'PROGRAMMING');

insert into CATEGORY(category_id,description)values(1005,'OPERATING SYSTEMS');

COMMIT;

desc CATEGORY;

SELECT *FROM CATEGORY;

insert into
CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(11,'Unix System Prg',1001,1,1001,2000,251);

insert into
CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(12,'Digital Signals',1002,2,1003,2001,425);

insert into
CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(13,'Logic Design',1003,3,1002,1999,225);
```

```
insert into
CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(14,'Server
Prg',1004,4,1004,2001,333);

insert into
CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(15,'Linux
OS',1005,5,1005,2003,326);

insert into
CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(16,'C++
Bible',1005,5 ,1001,2000,526);

insert into
CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(17,'COBOL
Handbook',1005,4,1001,2000,658);

COMMIT;

desc CATALOG;

SELECT *FROM CATALOG;

insert into ORDER_DETAILS(order_no,book_id,quantity)values(1,11,5);

insert into ORDER_DETAILS(order_no,book_id,quantity)values(2,12,8);

insert into ORDER_DETAILS(order_no,book_id,quantity)values(3,13,15);

insert into ORDER_DETAILS(order_no,book_id,quantity)values(4,14,22);

insert into ORDER_DETAILS(order_no,book_id,quantity)values(5,15,3);

insert into ORDER_DETAILS(order_no,book_id,quantity)values(2,17,10);

COMMIT;

desc ORDER_DETAILS;

SELECT *FROM ORDER_DETAILS;

SELECT AUTHOR.author_id,name,city,country FROM AUTHOR,CATALOG where
AUTHOR.author_id=CATALOG.author_id group by CATALOG.author_id having
count(CATALOG.author_id)>=2;

SELECT PRICE FROM CATALOG where year>2000;

select name from AUTHOR,CATALOG where AUTHOR.author_id=CATALOG.author_id and
book_id in(select book_id from ORDER_DETAILS where quantity=(select max(quantity) from
ORDER_DETAILS));

update CATALOG set price=1.1*price where publisher_id in(select publisher_id from
PUBLISHER where name='PEARSON');

COMMIT;
```

SELECT *FROM CATALOG;

OUTPUT SCREENSHOTS:

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer

SCHEMAS

43 | foreign key(book_id) references CATALOG(book_id) on delete cascade,
44 | quantity int
45 |);
46 • show tables;
47 • desc ORDER_DETAILS;
48 • SELECT *FROM ORDER_DETAILS;
49 • insert into AUTHOR(author_id,name,city,country)values(1001,'TERAS CHAN','CA','USA');
50 • insert into AUTHOR(author_id,name,city,country)values(1002,'STEVENS','ZOMBI','UGANDA');
51 • insert into AUTHOR(author_id,name,city,country)values(1003,'M MANO','CAIR','CANADA');
52 • insert into AUTHOR(author_id,name,city,country)values(1004,'KARTHIK B.P','NEW YORK','USA');

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

Tables_in_bookdealer

author catalog category order_details publisher

Administration Schemas

This screenshot shows the MySQL Workbench interface. The 'Bookdealer' tab is active. In the central pane, a SQL script is being run, creating a table named ORDER_DETAILS with a foreign key constraint referencing the CATALOG table. The script also lists all tables in the Bookdealer schema. The 'Result Grid' tab is selected, showing the results of the 'show tables' command.

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer

SCHEMAS

52 • insert into AUTHOR(author_id,name,city,country)values(1004,'KARTHIK B.P','NEW YORK','USA');
53 • insert into AUTHOR(author_id,name,city,country)values(1005,'WILLIAM STALLINGS','LAS VEGAS','USA');
54 • COMMIT;
55 • desc AUTHOR;
56 • SELECT *FROM AUTHOR;
57 • insert into PUBLISHER(publisher_id,name,city,country)values(1,'PEARSON','NEW YORK','USA');
58 • insert into PUBLISHER(publisher_id,name,city,country)values(2,'EEE','NEW SOUTH VALES','USA');
59 • insert into PUBLISHER(publisher_id,name,city,country)values(3,'PHI','DELHI','INDIA');
60 • insert into PUBLISHER(publisher_id,name,city,country)values(4,'WILLEY','BERLIN','GERMANY');
61 • insert into PUBLISHER(publisher_id,name,city,country)values(5,'MGH ','NEW YORK','USA');

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

author_id	name	city	country
1001	TERAS CHAN	CA	USA
1002	STEVENS	ZOMBI	UGANDA
1003	M MANO	CAIR	CANADA
1004	KARTHIK B.P	NEW YORK	USA
1005	WILLIAM STALLINGS	LAS VEGAS	USA
*	NULL	NULL	NULL

Administration Schemas

This screenshot shows the MySQL Workbench interface. The 'Bookdealer' tab is active. A SQL script is running, inserting data into the AUTHOR and PUBLISHER tables. The 'Result Grid' tab is selected, displaying the inserted data for the AUTHOR table.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Bookdealer

SCHEMAS

- database1
- insurance
- Tables
- Views
- Stored Procedures
- Functions
- sys

```

58 • insert into PUBLISHER(publisher_id,name,city,country)values(2,'EEE','NEW SOUTH VALES','USA');
59 • insert into PUBLISHER(publisher_id,name,city,country)values(3,'PHI','DELHI','INDIA');
60 • insert into PUBLISHER(publisher_id,name,city,country)values(4,'WILLEY','BERLIN','GERMANY');
61 • insert into PUBLISHER(publisher_id,name,city,country)values(5,'MGH ','NEW YORK','USA');
62 • COMMIT;
63 • desc PUBLISHER;
64 • SELECT *FROM PUBLISHER;
65 • insert into CATEGORY(category_id,description)values(1001,'COMPUTER SCIENCE');
66 • insert into CATEGORY(category_id,description)values(1002,'ALGORITHM DESIGN');
67 • insert into CATEGORY(category_id,description)values(1003,'ELECTRONICS');

```

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

publisher_id	name	city	country
1	PEARSON	NEW YORK	USA
2	EEE	NEW SOUTH VALES	USA
3	PHI	DELHI	INDIA
4	WILLEY	BERLIN	GERMANY
5	MGH	NEW YORK	USA
*	HULL	HULL	HULL

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Bookdealer

SCHEMAS

- database1
- insurance
- Tables
- Views
- Stored Procedures
- Functions
- sys

```

70 • COMMIT;
71 • desc CATEGORY;
72 • SELECT *FROM CATEGORY;
73 • insert into CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(11,'Unix System Prg',10
74 • insert into CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(12,'Digital Signals',10
75 • insert into CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(13,'Logic Design',1003,
76 • insert into CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(14,'Server Prg',1004,4,
77 • insert into CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(15,'Linux OS',1005,5,10
78 • insert into CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(16,'C++ Bible',1005,5,
79 • insert into CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(17,'COBOL Handbook',100

```

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

category_id	description
1001	COMPUTER SCIENCE
1002	ALGORITHM DESIGN
1003	ELECTRONICS
1004	PROGRAMMING
1005	OPERATING SYSTEMS
*	HULL

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer x

SCHEMAS

- Filter objects
- database1
- insurance
- Tables
- Views
- Stored Procedures
- Functions
- sys

```

76 • insert into CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(14,'Server Prg',1004,4,1004,4,4)
77 • insert into CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(15,'Linux OS',1005,5,1005,5,10)
78 • insert into CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(16,'C++ Bible',1005,5,1005,5,5)
79 • insert into CATALOG(book_id,title,author_id,publisher_id,category_id,year,price)values(17,'COBOL Handbook',100,100,100,100)
80 • COMMIT;
81 • desc CATALOG;
82 • SELECT *FROM CATALOG;
83 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(1,11,5);

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor | Field Types

book_id	title	author_id	publisher_id	category_id	year	price
11	Unix System Prg	1001	1	1001	2000	251
12	Digital Signals	1002	2	1003	2001	425
13	Logic Design	1003	3	1002	1999	225
14	Server Prg	1004	4	1004	2001	333
15	Linux OS	1005	5	1005	2003	326
16	C++ Bible	1005	5	1001	2000	526
17	COBOL Handbook	1005	4	1001	2000	658
*	NULL	NULL	NULL	NULL	NULL	NULL

Administration Schemas

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer x

SCHEMAS

- Filter objects
- database1
- insurance
- Tables
- Views
- Stored Procedures
- Functions
- sys

```

84 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(2,12,8);
85 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(3,13,15);
86 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(4,14,22);
87 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(5,15,3);
88 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(2,17,10);
89 • COMMIT;
90 • desc ORDER_DETAILS;
91 • SELECT *FROM ORDER_DETAILS;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Result Grid | Form Editor | Field Types

order_no	book_id	quantity
1	11	5
2	12	8
3	13	15
4	14	22
5	15	3
2	17	10

Administration Schemas

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer

SCHEMAS

Filter objects

database1 insurance

Tables Views Stored Procedures Functions

sys

Query Editor:

```
83 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(1,11,5);
84 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(2,12,8);
85 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(3,13,15);
86 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(4,14,22);
87 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(5,15,3);
88 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(2,17,10);
89 • COMMIT;
90 • desc ORDER_DETAILS;
91 • SELECT *FROM ORDER_DETAILS;
92 • SELECT AUTHOR.author_id,name,city,country FROM AUTHOR,CATALOG where AUTHOR.author_id=CATALOG.author_id group by
93 • SELECT PRICE FROM CATALOG where year>2000;
```

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

author_id	name	city	country
1005	WILLIAM STALLINGS	LAS VEGAS	USA

Administration Schemas

Result Grid

Form Editor

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer

SCHEMAS

Filter objects

database1 insurance

Tables Views Stored Procedures Functions

sys

Query Editor:

```
84 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(2,12,8);
85 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(3,13,15);
86 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(4,14,22);
87 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(5,15,3);
88 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(2,17,10);
89 • COMMIT;
90 • desc ORDER_DETAILS;
91 • SELECT *FROM ORDER_DETAILS;
92 • SELECT AUTHOR.author_id,name,city,country FROM AUTHOR,CATALOG where AUTHOR.author_id=CATALOG.author_id group by
93 • SELECT PRICE FROM CATALOG where year>2000;
```

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

PRICE
425
333
326

Administration Schemas

CATALOG 30

Result Grid

Form Editor

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer

SCHEMAS

Filter objects

- database1
- insurance
 - Tables
 - Views
 - Stored Procedures
 - Functions
- sys

```

85 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(3,13,15);
86 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(4,14,22);
87 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(5,15,3);
88 • insert into ORDER_DETAILS(order_no,book_id,quantity)values(2,17,10);
89 • COMMIT;
90 • desc ORDER_DETAILS;
91 • SELECT *FROM ORDER_DETAILS;
92 • SELECT AUTHOR.author_id,name,city,country FROM AUTHOR,CATALOG where AUTHOR.author_id=CATALOG.author_id group b
93 • SELECT PRICE FROM CATALOG where year>2000;
94 • select name from AUTHOR,CATALOG where AUTHOR.author_id=CATALOG.author_id and book_id in(select book_id from OR

```

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

name
KARTHIK.B.P

Administration Schemas

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer*

SCHEMAS

Filter objects

- database1
- insurance
 - Tables
 - Views
 - Stored Procedures
 - Functions
- sys

```

91 • SELECT *FROM ORDER_DETAILS;
92 • SELECT AUTHOR.author_id,name,city,country FROM AUTHOR,CATALOG where AUTHOR.author_id=CATALOG.author_id group b
93 • SELECT PRICE FROM CATALOG where year>2000;
94 • select name from AUTHOR,CATALOG where AUTHOR.author_id=CATALOG.author_id and book_id in(select book_id from OR
95 • update CATALOG set price=1.1*price where publisher_id in(select publisher_id from PUBLISHER where name='PEARSO
96 • COMMIT;
97 • SELECT *FROM CATALOG;
98

```

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

book_id	title	author_id	publisher_id	category_id	year	price
11	Unix System Prg	1001	1	1001	2000	276
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

Administration Schemas

PROGRAM-3:ORDER PROCESSING DATABASE

QUESTION:

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.

PROGRAM CODE:

```
create database Order_processing;
use Order_processing;
CREATE TABLE CUSTOMER
(
    cust_no int,
    cname VARCHAR(15),
    city VARCHAR(15),
    PRIMARY KEY(cust_no)
);

CREATE TABLE ORDERS(
    order_no int,
    odate date,
    cust_no int,
    foreign key(cust_no) references CUSTOMER(cust_no) on delete cascade,
    ord_Amt int,
    primary key(order_no)
);

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

create table ORDER_ITEM (
    order_no int,
    item_no int,
    qty int,
    foreign key(order_no) references ORDERS(order_no) on delete cascade,
```

```
foreign key(item_no) references ITEM(item_no) on delete SET NULL
);

create table WAREHOUSE(
warehouseno int,
city varchar(30),
primary key(warehouseno)
);

create table SHIPMENT(
order_no int,
warehouseno int,
ship_date date,
foreign key(order_no) references ORDERS(order_no) on delete cascade,
foreign key(warehouseno) references WAREHOUSE(warehouseno) on delete cascade
);

show tables;

insert into CUSTOMER(cust_no,cname,city)values(771,'PUSHPA K','BANGALORE');
insert into CUSTOMER(cust_no,cname,city)values(772,'SUMAN','MUMBAI');
insert into CUSTOMER(cust_no,cname,city)values(773,'SOURAV','CALICUT');
insert into CUSTOMER(cust_no,cname,city)values(774,'LAILA','HYDERABAD');
insert into CUSTOMER(cust_no,cname,city)values(775,'FAIZAL','BANGALORE');

COMMIT;

desc CUSTOMER;

SELECT *FROM CUSTOMER;

insert into ORDERS(order_no,odate,cust_no,ord_Amt)values(111,'22-01-
02',771,18000);

insert into ORDERS(order_no,odate,cust_no,ord_Amt)values(112,'30-07-
02',774,6000);

insert into ORDERS(order_no,odate,cust_no,ord_Amt)values(113,'03-04-
03',775,9000);
```

```
insert into ORDERS(order_no,odate,cust_no,ord_Amt)values(114,'03-11-03',775,29000);

insert into ORDERS(order_no,odate,cust_no,ord_Amt)values(115,'10-12-03',773,29000);

insert into ORDERS(order_no,odate,cust_no,ord_Amt)values(116,'19-08-04',772,56000);

insert into ORDERS(order_no,odate,cust_no,ord_Amt)values(117,'10-09-04',771,20000);

insert into ORDERS(order_no,odate,cust_no,ord_Amt)values(118,'20-11-04',775,29000);

insert into ORDERS(order_no,odate,cust_no,ord_Amt)values(119,'13-02-05',774,29000);

insert into ORDERS(order_no,odate,cust_no,ord_Amt)values(120,'13-10-05',775,29000);

COMMIT;

desc ORDERS;

SELECT *FROM ORDERS;

insert into ITEM(item_no,unit_price)values(5001,503);

insert into ITEM(item_no,unit_price)values(5002,750);

insert into ITEM(item_no,unit_price)values(5003,150);

insert into ITEM(item_no,unit_price)values(5004,600);

insert into ITEM(item_no,unit_price)values(5005,890);

COMMIT;

desc ITEM;

SELECT *FROM ITEM;

insert into ORDER_ITEM(order_no,item_no,qty)values(111,5001,50);

insert into ORDER_ITEM(order_no,item_no,qty)values(112,5003,20);

insert into ORDER_ITEM(order_no,item_no,qty)values(113,5002,50);

insert into ORDER_ITEM(order_no,item_no,qty)values(114,5005,60);

insert into ORDER_ITEM(order_no,item_no,qty)values(115,5004,90);
```

```
insert into ORDER_ITEM(order_no,item_no,qty)values(116,5001,10);
insert into ORDER_ITEM(order_no,item_no,qty)values(117,5003,80);
insert into ORDER_ITEM(order_no,item_no,qty)values(118,5005,50);
insert into ORDER_ITEM(order_no,item_no,qty)values(119,5002,10);
insert into ORDER_ITEM(order_no,item_no,qty)values(120,5004,45);
```

```
COMMIT;
```

```
desc ORDER_ITEM;
```

```
SELECT *FROM ORDER_ITEM;
```

```
insert into WAREHOUSE(warehouseno,city)values(1,'DELHI');
```

```
insert into WAREHOUSE(warehouseno,city)values(2,'BOMBAY');
```

```
insert into WAREHOUSE(warehouseno,city)values(3,'CHENNAI');
```

```
insert into WAREHOUSE(warehouseno,city)values(4,'BANGALORE');
```

```
insert into WAREHOUSE(warehouseno,city)values(5,'BANGALORE');
```

```
insert into WAREHOUSE(warehouseno,city)values(6,'DELHI');
```

```
insert into WAREHOUSE(warehouseno,city)values(7,'BOMBAY');
```

```
insert into WAREHOUSE(warehouseno,city)values(8,'CHENNAI');
```

```
insert into WAREHOUSE(warehouseno,city)values(9,'DELHI');
```

```
insert into WAREHOUSE(warehouseno,city)values(10,'BANGALORE');
```

```
COMMIT;
```

```
desc WAREHOUSE;
```

```
SELECT *FROM WAREHOUSE;
```

```
insert into SHIPMENT(order_no,warehouseno,ship_date)values(111,1,'10-02-02');
```

```
insert into SHIPMENT(order_no,warehouseno,ship_date)values(112,5,'10-09-02');
```

```
insert into SHIPMENT(order_no,warehouseno,ship_date)values(113,8,'10-02-03');
```

```
insert into SHIPMENT(order_no,warehouseno,ship_date)values(114,3,'10-12-03');
```

```
insert into SHIPMENT(order_no,warehouseno,ship_date)values(115,9,'19-01-04');
```

```
insert into SHIPMENT(order_no,warehouseno,ship_date)values(116,1,'20-09-04');
```

```
insert into SHIPMENT(order_no,warehouseno,ship_date)values(117,5,'10-09-04');
```

```
insert into SHIPMENT(order_no,warehouseno,ship_date)values(118,7,'30-11-04');
insert into SHIPMENT(order_no,warehouseno,ship_date)values(119,7,'30-04-05');
insert into SHIPMENT(order_no,warehouseno,ship_date)values(120,6,'21-12-05');
COMMIT;
desc SHIPMENT;
SELECT *FROM SHIPMENT;
/*Produce a listing: CUSTNAME, #oforders, AVG_ORDER_AMT, where the middle
column
is the total numbers of orders by the customer and the last column is the average
order amount for that customer.*/
SELECT C.CNAME as CUSTNAME, COUNT(*) as no_of_orders,AVG(O.ord_Amt) as
AVG_ORDER_AMT FROM CUSTOMER C,
ORDERS O WHERE C.cust_no=O.cust_no GROUP BY C.CNAME;
/*List the order# for orders that were shipped from all warehouses that the company
has in a specific city.*/
SELECT order_no FROM WAREHOUSE W, SHIPMENT S WHERE
W.warehouseno=S.warehouseno AND CITY='BANGALORE';
/*Demonstrate how you delete item# 10 from the ITEM table and make that field
null in the ORDER_ITEM table.*/
delete from ITEM where item_no=5005;
select *from ITEM;
select *from ORDER_ITEM;
```

OUTPUT SCREENSHOTS:

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator Insurance Bookdealer OrderProcessing

SCHEMAS Filter objects

- bookdealer
- database1
- insurance
 - Tables
 - Views
 - Stored Procedures
 - Functions
- sys

38 ship_date date,
39 foreign key(order_no) references ORDERS(order_no) on delete cascade,
40 foreign key(warehouseno) references WAREHOUSE(warehouseno) on delete cascade
41);
42 • show tables;
43 • insert into CUSTOMER(cust_no,cname,city)values(771,'PUSHPA K','BANGALORE');
44 • insert into CUSTOMER(cust_no,cname,city)values(772,'SUMAN','MUMBAI');
45 • insert into CUSTOMER(cust_no,cname,city)values(773,'SOURAV','CALICUT');

Result Grid Filter Rows: Export: Wrap Cell Content:

Tables_in_order_processing		
	customer	item
▶	customer	item
	order_item	orders
	shipment	warehouse

Administration Schemas

Information

Result Grid Form Editor Field Types

```
ship_date date,
foreign key(order_no) references ORDERS(order_no) on delete cascade,
foreign key(warehouseno) references WAREHOUSE(warehouseno) on delete cascade
);
show tables;
insert into CUSTOMER(cust_no,cname,city)values(771,'PUSHPA K','BANGALORE');
insert into CUSTOMER(cust_no,cname,city)values(772,'SUMAN','MUMBAI');
insert into CUSTOMER(cust_no,cname,city)values(773,'SOURAV','CALICUT');
```

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator Insurance Bookdealer OrderProcessing

SCHEMAS Filter objects

- bookdealer
- database1
- insurance
 - Tables
 - Views
 - Stored Procedures
 - Functions
- sys

44 • insert into CUSTOMER(cust_no,cname,city)values(772,'SUMAN','MUMBAI');
45 • insert into CUSTOMER(cust_no,cname,city)values(773,'SOURAV','CALICUT');
46 • insert into CUSTOMER(cust_no,cname,city)values(774,'LAILA','HYDERABAD');
47 • insert into CUSTOMER(cust_no,cname,city)values(775,'FAIZAL','BANGALORE');
48 • COMMIT;
49 • desc CUSTOMER;
50 • SELECT *FROM CUSTOMER;
51 • insert into ORDERS(order_no,odate,cust_no,ord_Amt)values(111,'22-01-02',771,18000);

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

cust_no	cname	city
771	PUSHPA K	BANGALORE
772	SUMAN	MUMBAI
773	SOURAV	CALICUT
774	LAILA	HYDERABAD
775	FAIZAL	BANGALORE
*	NULL	NULL

Administration Schemas

Result Grid Form Editor Field Types

```
insert into CUSTOMER(cust_no,cname,city)values(772,'SUMAN','MUMBAI');
insert into CUSTOMER(cust_no,cname,city)values(773,'SOURAV','CALICUT');
insert into CUSTOMER(cust_no,cname,city)values(774,'LAILA','HYDERABAD');
insert into CUSTOMER(cust_no,cname,city)values(775,'FAIZAL','BANGALORE');
COMMIT;
desc CUSTOMER;
SELECT *FROM CUSTOMER;
insert into ORDERS(order_no,odate,cust_no,ord_Amt)values(111,'22-01-02',771,18000);
```

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer OrderProcessing

SCHEMAS

Filter objects

bookdealer

database1

insurance

Tables

Views

Stored Procedures

Functions

sys

Limit to 1000 rows

59 • insert into ORDERS(order_no,odate,cust_no,ord_Amt)values(119,'13-02-05',774,29000);
60 • insert into ORDERS(order_no,odate,cust_no,ord_Amt)values(120,'13-10-05',775,29000);
61 • COMMIT;
62 • desc ORDERS;
63 • SELECT *FROM ORDERS;
64 • insert into ITEM(item_no,unit_price)values(5001,503);

Result Grid

Filter Rows:

order_no	odate	cust_no	ord_Amt
111	2022-01-02	771	18000
112	2030-07-02	774	6000
113	2003-04-03	775	9000
114	2003-11-03	775	29000
115	2010-12-03	773	29000
116	2019-08-04	772	56000
117	2010-09-04	771	20000
118	2020-11-04	775	29000
119	2013-02-05	774	29000
120	2013-10-05	775	29000
*	HULL	HULL	HULL

Administration Schemas

Result Grid

Form Editor

Field Types

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer OrderProcessing

SCHEMAS

Filter objects

bookdealer

database1

insurance

Tables

Views

Stored Procedures

Functions

sys

Limit to 1000 rows

68 • insert into ITEM(item_no,unit_price)values(5005,890);
69 • COMMIT;
70 • desc ITEM;
71 • SELECT *FROM ITEM;
72 • insert into ORDER_ITEM(order_no,item_no,qty)values(111,5001,50);
73 • insert into ORDER_ITEM(order_no,item_no,qty)values(112,5003,20);
74 • insert into ORDER_ITEM(order_no,item_no,qty)values(113,5002,50);
75 • insert into ORDER_ITEM(order_no,item_no,qty)values(114,5005,60);

Result Grid

Filter Rows:

item_no	unit_price
5001	503
5002	750
5003	150
5004	600
5005	890
*	HULL

Administration Schemas

Information

Result Grid

Form Editor

Field Types

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer OrderProcessing

SCHEMAS

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

```
79 • insert into ORDER_ITEM(order_no,item_no,qty)values(118,5005,50);
80 • insert into ORDER_ITEM(order_no,item_no,qty)values(119,5002,10);
81 • insert into ORDER_ITEM(order_no,item_no,qty)values(120,5004,45);
82 • COMMIT;
83 • desc ORDER_ITEM;
84 • SELECT *FROM ORDER_ITEM;
85 • insert into WAREHOUSE(warehouseno,city)values(1,'DELHI');
```

order_no	item_no	qty
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

Administration Schemas

Information

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer OrderProcessing

SCHEMAS

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

```
94 • insert into WAREHOUSE(warehouseno,city)values(10,'BANGALORE');
95 • COMMIT;
96 • desc WAREHOUSE;
97 • SELECT *FROM WAREHOUSE;
98 • insert into SHIPMENT(order_no,warehouseno,ship_date)values(111,1,'10-02-02');
99 • insert into SHIPMENT(order_no,warehouseno,ship_date)values(112,5,'10-09-02');
100 • insert into SHIPMENT(order_no,warehouseno,ship date)values(113,8,'10-02-03');
```

warehouseno	city
1	DELHI
2	BOMBAY
3	CHENNAI
4	BANGALORE
5	BANGALORE
6	DELHI
7	BOMBAY
8	CHENNAI
9	DELHI
10	BANGALORE
NULL	NULL

Administration Schemas

Information

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer OrderProcessing

SCHEMAS

- bookdealer
- database1
- insurance
 - Tables
 - Views
 - Stored Procedures
 - Functions
- sys

105 • insert into SHIPMENT(order_no,warehouseno,ship_date)values(118,7,'30-11-04');

106 • insert into SHIPMENT(order_no,warehouseno,ship_date)values(119,7,'30-04-05');

107 • insert into SHIPMENT(order_no,warehouseno,ship_date)values(120,6,'21-12-05');

108 • COMMIT;

109 • desc SHIPMENT;

110 • **SELECT *FROM SHIPMENT;**

111 /*Produce a listing: CUSTNAME, #oforders, AVG ORDER AMT, where the middle column

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

order_no	warehouseno	ship_date
111	1	2010-02-02
112	5	2010-09-02
113	8	2010-02-03
114	3	2010-12-03
115	9	2019-01-04
116	1	2020-09-04
117	5	2010-09-04
118	7	2030-11-04
119	7	2030-04-05
120	6	2021-12-05

Administration Schemas

Result Grid Form Editor Field Types

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer OrderProcessing*

SCHEMAS

- bookdealer
- database1
- insurance
 - Tables
 - Views
 - Stored Procedures
 - Functions
- sys

107 • insert into SHIPMENT(order_no,warehouseno,ship_date)values(120,6,'21-12-05');

108 • COMMIT;

109 • desc SHIPMENT;

110 • **SELECT *FROM SHIPMENT;**

111 /*Produce a listing: CUSTNAME, #oforders, AVG_ORDER_AMT, where the middle column

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

113 • **SELECT C.CNAME as CUSTNAME, COUNT(*) as no_of_orders, AVG(O.ord_Amt) as AVG_ORDER_AMT FROM CUSTOMER C,**

114 **ORDERS O WHERE C.cust_no=O.cust_no GROUP BY C.CNAME;**

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

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

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

Administration Schemas

Information

Result Grid Form Editor

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer OrderProcessing*

```

109 • desc SHIPMENT;
110 • SELECT *FROM SHIPMENT;
111 /*Produce a listing: CUSTNAME, #oforders, AVG_ORDER_AMT, where the middle column
112 is the total numbers of orders by the customer and the last column is the average order amount for that custom
113 • SELECT C.CNAME as CUSTNAME, COUNT(*) as no_of_orders,AVG(0.ord_Amt) as AVG_ORDER_AMT FROM CUSTOMER C,
114 ORDERS O WHERE C.cust_no=O.cust_no GROUP BY C.CNAME;
115 /*List the order# for orders that were shipped from all warehouses that the company has in a specific city.*/
116 • SELECT order_no FROM WAREHOUSE W, SHIPMENT S WHERE W.warehouseno=S.warehouseno AND CITY='BANGALORE';
117 /*Demonstrate how you delete item# 10 from the ITEM table and make that field null in the ORDER_ITEM table.*/

```

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

order_no
112
117

Administration Schemas

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer OrderProcessing*

```

111 /*Produce a listing: CUSTNAME, #oforders, AVG_ORDER_AMT, where the middle column
112 is the total numbers of orders by the customer and the last column is the average order amount for that custom
113 • SELECT C.CNAME as CUSTNAME, COUNT(*) as no_of_orders,AVG(0.ord_Amt) as AVG_ORDER_AMT FROM CUSTOMER C,
114 ORDERS O WHERE C.cust_no=O.cust_no GROUP BY C.CNAME;
115 /*List the order# for orders that were shipped from all warehouses that the company has in a specific city.*/
116 • SELECT order_no FROM WAREHOUSE W, SHIPMENT S WHERE W.warehouseno=S.warehouseno AND CITY='BANGALORE';
117 /*Demonstrate how you delete item# 10 from the ITEM table and make that field null in the ORDER_ITEM table.*/
118 • delete from ITEM where item_no=5005;
119 • select *from ITEM;

```

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

item_no	unit_price
5001	503
5002	750
5003	150
5004	600
NULL	NULL

Administration Schemas

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar has sections for Navigator, Schemas, Administration, and Information. The main area shows a query editor titled 'OrderProcessing' with the following SQL code:

```

115  /*List the order# for orders that were shipped from all warehouses that the company has in a specific city.*/
116  • SELECT order_no FROM WAREHOUSE W, SHIPMENT S WHERE W.warehouseno=S.warehouseno AND CITY='BANGALORE';
117  /*Demonstrate how you delete item# 10 from the ITEM table and make that field null in the ORDER_ITEM table.*/
118  • delete from ITEM where item_no=5005;
119  • select *from ITEM;
120  • select *from ORDER_ITEM;

```

Below the code is a 'Result Grid' showing the output of the last query:

order_no	item_no	qty
111	5001	50
112	5003	20
113	5002	50
114	NULL	60
115	5004	90
116	5001	10
117	5003	80
118	NULL	50
119	5002	10
120	5004	45

PROGRAM-4:BANKING DATABASE

Question:

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.

vi) Generate suitable reports.

vii) Create suitable front end for querying and displaying the results.

PROGRAM CODE:

```
create database banking;
```

```
use banking;
```

```
create table branch(  
branch_name varchar(30) primary key,  
branch_city varchar(30),  
assets real);
```

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

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

```
create table depositor(  
customer_name varchar(30),
```

```
accno int,  
primary key(customer_name ,accno),  
foreign key (accno) references accounts(accno) on delete cascade on update cascade,  
foreign key (customer_name) references customer(customer_name) on delete cascade on  
update  
cascade);
```

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

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

```
show tables;  
insert into branch(branch_name,branch_city,assets) values  
('A','Bangalore',190000),  
('B','Bangalore',200000),  
('C','Delhi',235344),  
('D','Chennai',1050560),  
('E','Chennai',678909);  
select *from branch;
```

```
insert into accounts(accno,branch_name,balance) VALUES  
(1001,'A',10000),  
(1002,'B',5000),  
(1003,'C',7500),  
(1004,'D',50000),  
(1005,'D',75000),  
(1006,'E',560),  
(1007,"B",500),  
(1008,"B",1500);  
select *from accounts;
```

```
insert into customer(customer_name,customer_street,customer_city) VALUES  
("Ravi","Dasarahalli","Bangalore"),  
("Shyam","Indiranagar","Delhi"),  
("Seema","Vasantnagar","Chennai"),  
("Arpita","Church Street","Bangalore"),  
("Vinay","MG Road","Chennai");  
select *from customer;
```

```
insert into depositor(customer_name,accno) VALUES  
("Ravi",1001),  
("Ravi",1002),  
("Shyam",1003),  
("Seema",1004),  
("Seema",1005),  
("Arpita",1006),  
("Vinay",1007),  
("Vinay",1008);
```

```
select *from depositor;
```

```
insert into loan(loan_number,branch_name,amount) VALUES  
(001,'A',10000),  
(002,'B',25000),  
(003,'B',250000),  
(004,'C',5000),  
(005,'E',90000);
```

```
select *from loan;
```

```
insert into borrower(customer_name,loan_number) VALUES  
("Arpita",001),  
("Ravi",002),  
("Arpita",003),  
("Shyam",004),  
("Vinay",005);
```

```
select *from borrower;
```

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

```
select customer_name from depositor  
join accounts on depositor.accno = accounts.accno where accounts.branch_name = "D"  
group by depositor.customer_name having count(depositor.customer_name) >=2;
```

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

```
select customer_name from depositor  
join accounts on accounts.accno = depositor.accno
```

```

join branch on branch.branch_name = accounts.branch_name
where branch.branch_city = "Bangalore"
GROUP BY depositor.customer_name
having count(DISTINCT branch.branch_name) = (SELECT COUNT(branch_name)
FROM branch
WHERE branch_city = 'Bangalore');

```

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

```

delete from accounts where branch_name in
(select branch_name from branch where branch_city="Delhi");
select *from accounts;

```

OUTPUT SCREENSHOTS:

The screenshot shows the MySQL Workbench interface with the 'BankingDatabase' selected. In the central query editor, the following SQL code is visible:

```

97  /*iii. Find all the customers who have at least two accounts at the Main branch */
98
99 • select customer_name from depositor
100 join accounts on depositor.accno = accounts.accno where accounts.branch_name = "D"
101 group by depositor.customer_name having count(depositor.customer_name) >=2;
102
103 /* iv. Find all the customers who have an account at all the branches located in a specific city.*/
104

```

The result grid shows a single row with the customer name 'Seema'.

The screenshot shows the MySQL Workbench interface with the 'BankingDatabase' selected. In the central query editor, the following SQL code is visible:

```

104
105 • select customer_name from depositor
106 join accounts on accounts.accno = depositor.accno
107 join branch on branch.branch_name = accounts.branch_name
108 where branch.branch_city = "Bangalore"
109 GROUP BY depositor.customer_name
110 having count(DISTINCT branch.branch_name) = (SELECT COUNT(branch_name)
111 FROM branch
112 WHERE branch_city = 'Bangalore');

```

The result grid shows a single row with the customer name 'Ravi'.

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer OrderProcessing* BankingDatabase ×

SCHEMAS: Filter objects

- bookdealer
- database1
- insurance
- order_processing
 - Tables
 - Views
 - Stored Procedures
 - Functions
- sys

```

111   FROM branch
112   WHERE branch_city = 'Bangalore');

113
114 /*v. Demonstrate how you delete all account tuples at every branch located in a specific city.*/
115 • delete from accounts where branch_name in
116   (select branch_name from branch where branch_city="Delhi");
117 • select *from accounts;

```

Result Grid | Filter Rows: | Edits: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor

accno	branch_name	balance
1001	A	10000
1002	B	5000
1004	D	50000
1005	D	75000
1006	E	560
1007	B	500
1008	B	1500
• NULL	NULL	NULL

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer OrderProcessing* BankingDatabase ×

SCHEMAS: Filter objects

- bookdealer
- database1
- insurance
- order_processing
 - Tables
 - Views
 - Stored Procedures
 - Functions
- sys

```

39 foreign key (customer_name) references customer(customer_name) on delete cascade on update cascade,
40 foreign key (loan_number) references loan(loan_number) on delete cascade on update cascade);
41 • show tables;
42 • insert into branch(branch_name,branch_city,assets) values
43 ('A','Bangalore',190000),
44 ('B','Bangalore',200000),
45 ('C','Delhi',235344),
46 ('D','Chennai',1050560),
47 ('E','Chennai',678909);
48 select *from branch;
49
50 insert into accounts(accno,branch name,balance) VALUES

```

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

Tables_in_banking		
accounts		
borrower		
branch		
customer		
depositor		
loan		

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Insurance Bookdealer OrderProcessing* BankingDatabase ×

SCHEMAS: Filter objects

- bookdealer
- database1
- insurance
- order_processing
 - Tables
 - Views
 - Stored Procedures
 - Functions
- sys

```

43 ('A','Bangalore',190000),
44 ('B','Bangalore',200000),
45 ('C','Delhi',235344),
46 ('D','Chennai',1050560),
47 ('E','Chennai',678909);
48 select *from branch;
49
50 insert into accounts(accno,branch name,balance) VALUES

```

Result Grid | Filter Rows: | Edits: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor

branch_name	branch_city	assets
A	Bangalore	190000
B	Bangalore	200000
C	Delhi	235344
D	Chennai	1050560
E	Chennai	678909
• NULL	NULL	NULL

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

bookdealer

database1

insurance

order_processing

Tables

Views

Stored Procedures

Functions

sys

BankingDatabase

Limit to 1000 rows

55 (1005,'D',75000),
56 (1006,'E',560),
57 (1007,"B",500),
58 (1008,"B",1500);
59 select *from accounts;
60

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

acco	branch_name	balance
1001	A	10000
1002	B	5000
1003	C	7500
1004	D	50000
1005	D	75000
1006	E	560
1007	B	500
1008	B	1500
*	HULL	HULL

Result Grid Form Editor Field Types

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

bookdealer

database1

insurance

order_processing

Tables

Views

Stored Procedures

Functions

sys

BankingDatabase

Limit to 1000 rows

61 insert into customer(customer_name,customer_street,customer_city) VALUES
62 ("Ravi","Dasarahalli","Bangalore"),
63 ("Shyam","Indiranagar","Delhi"),
64 ("Seema","Vasanthnagar","Chennai"),
65 ("Arpita","Church Street","Bangalore"),
66 ("Vinay","MG Road","Chennai");
67 select *from customer;

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

customer_name	customer_street	customer_city
Arpita	Church Street	Bangalore
Ravi	Dasarahalli	Bangalore
Seema	Vasanthnagar	Chennai
Shyam	Indiranagar	Delhi
Vinay	MG Road	Chennai
*	HULL	HULL

Result Grid Form Editor

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

bookdealer

database1

insurance

order_processing

Tables

Views

Stored Procedures

Functions

sys

BankingDatabase

Limit to 1000 rows

76 ("Vinay",1007),
77 ("Vinay",1008);
78 select *from depositor;

80

81 insert into loan(loan_number,branch_name,amount) VALUES
82 (001,'A',10000),

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

customer_name	acco
Ravi	1001
Ravi	1002
Shyam	1003
Seema	1004
Seema	1005
Arpita	1006
Vinay	1007
Vinay	1008
*	HULL

Result Grid Form Editor

The image shows two separate sessions in MySQL Workbench. Both sessions are connected to 'Local instance MySQL80' and are using the 'BankingDatabase' schema.

Session 1:

```

82 (001,'A',10000),
83 (002,'B',25000),
84 (003,'B',250000),
85 (004,'C',5000),
86 (005,'E',90000);
87 select *from loan;
88

```

Result Grid:

loan_number	branch_name	amount
1	A	10000
2	B	25000
3	B	250000
4	C	5000
5	E	90000
*	HULL	HULL

Session 2:

```

94 ("Vinay",005);
95 • select *from borrower;
96
97 /*iii. Find all the customers who have at least two accounts at the Main branch */
98
99 • select customer_name from depositor
100 join accounts on depositor.accno = accounts.accno where accounts.branch_name = "D"
101 group by depositor.customer_name having count(depositor.customer_name) >=2;

```

Result Grid:

customer_name	loan_number
Arpta	1
Ravi	2
Arpta	3
Shyam	4
Vinay	5
*	HULL

PROGRAM 5-STUDENT ENROLLMENT DATABASE

QUESTION:

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.

PROGRAM CODE:

```
CREATE DATABASE STUDENTENROLLMENT;
```

```
USE STUDENTENROLLMENT;
```

```
CREATE TABLE STUDENT(
```

```
    REG_NO VARCHAR(30),
```

```
    SNAME VARCHAR(30),
```

```
    MAJOR VARCHAR(30),
```

```
    BDATE DATE,
```

```
    PRIMARY KEY(REG_NO)
```

```
);
```

```
CREATE TABLE COURSE(
```

```
    COURSE_ID INT,
```

```
    CNAME VARCHAR(30),
```

```
    DEPT VARCHAR(30),
```

```
    PRIMARY KEY(COURSE_ID)
```

```
);
```

```
CREATE TABLE ENROLL(
```

```
    REG_NO VARCHAR(30),
```

```
COURSE_ID INT,  
SEM INT,  
MARKS INT,  
FOREIGN KEY(REG_NO) REFERENCES STUDENT(REG_NO) ON DELETE CASCADE ON  
UPDATE CASCADE,  
FOREIGN KEY(COURSE_ID) REFERENCES COURSE(COURSE_ID) ON DELETE CASCADE  
ON UPDATE CASCADE  
);
```

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

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

```
show tables;
```

```
INSERT INTO STUDENT(REG_NO, SNAME, MAJOR, BDATE) VALUES ('CS01', 'RAM', 'DS',  
'1986-03-12');
```

```
INSERT INTO STUDENT(REG_NO, SNAME, MAJOR, BDATE) VALUES ('IS02', 'SMITH', 'USP',  
'1987-12-23');
```

```
INSERT INTO STUDENT(REG_NO, SNAME, MAJOR, BDATE) VALUES ('EC03', 'AHMED', 'SNS',  
'1985-04-17');  
  
INSERT INTO STUDENT(REG_NO, SNAME, MAJOR, BDATE) VALUES ('CS03', 'SNEHA', 'DBMS',  
'1987-01-01');  
  
INSERT INTO STUDENT(REG_NO, SNAME, MAJOR, BDATE) VALUES ('TC05', 'AKHILA', 'EC',  
'1986-10-06');  
  
SELECT * FROM STUDENT;
```

```
INSERT INTO COURSE(COURSE_ID, CNAME, DEPT) VALUES (11, 'DS', 'CS');  
  
INSERT INTO COURSE(COURSE_ID, CNAME, DEPT) VALUES (22, 'USP', 'IS');  
  
INSERT INTO COURSE(COURSE_ID, CNAME, DEPT) VALUES (33, 'SNS', 'EC');  
  
INSERT INTO COURSE(COURSE_ID, CNAME, DEPT) VALUES (44, 'DBMS', 'CS');  
  
INSERT INTO COURSE(COURSE_ID, CNAME, DEPT) VALUES (55, 'EC', 'TC');  
  
SELECT * FROM COURSE;
```

```
INSERT INTO ENROLL(REG_NO, COURSE_ID, SEM, MARKS) VALUES ('CS01', 11, 4, 85);  
  
INSERT INTO ENROLL(REG_NO, COURSE_ID, SEM, MARKS) VALUES ('IS02', 22, 6, 80);  
  
INSERT INTO ENROLL(REG_NO, COURSE_ID, SEM, MARKS) VALUES ('EC03', 33, 2, 80);  
  
INSERT INTO ENROLL(REG_NO, COURSE_ID, SEM, MARKS) VALUES ('CS03', 44, 6, 75);  
  
INSERT INTO ENROLL(REG_NO, COURSE_ID, SEM, MARKS) VALUES ('TC05', 55, 2, 8);  
  
SELECT * FROM ENROLL;
```

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

```
INSERT INTO TEXT(BOOK_ISBN, BOOK_TITLE, PUBLISHER, AUTHOR) VALUES (1, 'DS and C',  
'Princeton', 'Padma Reddy');  
  
INSERT INTO TEXT(BOOK_ISBN, BOOK_TITLE, PUBLISHER, AUTHOR) VALUES (2,  
'Fundamentals of DS', 'Princeton', 'Godse');  
  
INSERT INTO TEXT(BOOK_ISBN, BOOK_TITLE, PUBLISHER, AUTHOR) VALUES (3,  
'Fundamentals of DBMS', 'Princeton', 'Navathe');  
  
INSERT INTO TEXT(BOOK_ISBN, BOOK_TITLE, PUBLISHER, AUTHOR) VALUES (4, 'SQL',  
'Princeton', 'Foley');  
  
INSERT INTO TEXT(BOOK_ISBN, BOOK_TITLE, PUBLISHER, AUTHOR) VALUES (5, 'Electronic  
circuits', 'TMH', 'Elmasri');  
  
INSERT INTO TEXT(BOOK_ISBN, BOOK_TITLE, PUBLISHER, AUTHOR) VALUES (6, 'Adv unix  
prog', 'TMH', 'Stevens');
```

```
SELECT * FROM TEXT;
```

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

```
INSERT INTO TEXT VALUES(7, "TREES & GRAPHS", "PRINCETON", "SADGE");
```

```
INSERT INTO BOOK_ADOPTION VALUES(11, 4, 8);
```

```
SELECT * FROM BOOK_ADOPTION;
```

```
SELECT * FROM TEXT;
```

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

```
SELECT C.COURSE_ID,T.BOOK_ISBN,T.BOOK_TITLE FROM TEXT T,COURSE  
C,BOOK_ADOPTION B WHERE T.BOOK_ISBN=B.BOOK_ISBN AND  
B.COURSE_ID=C.COURSE_ID AND C.DEPT="CS" AND (SELECT COUNT(B.BOOK_ISBN) FROM  
BOOK_ADOPTION B WHERE  
C.COURSE_ID=B.COURSE_ID)>=2 ORDER BY T.BOOK_TITLE;
```

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

```
SELECT DISTINCT C.DEPT  
FROM COURSE C  
WHERE C.DEPT IN  
( SELECT C.DEPT  
FROM COURSE C,BOOK_ADOPTION B,TEXT T  
WHERE C.COURSE_ID=B.COURSE_ID  
AND T.BOOK_ISBN=B.BOOK_ISBN  
AND T.PUBLISHER='Princeton')  
AND C.DEPT NOT IN  
(SELECT C.DEPT  
FROM COURSE C,BOOK_ADOPTION B,TEXT T  
WHERE C.COURSE_ID=B.COURSE_ID  
AND T.BOOK_ISBN=B.BOOK_ISBN  
AND T.PUBLISHER != 'Princeton');
```

OUTPUT SCREENSHOTS:

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The main window has tabs for OrderProcessing, BankingDatabase, and StudentEnrollment. The left sidebar shows the Navigator with Schemas (banking, bookdealer, database1, insurance, order_processing) and Tables (customer, item, order_item, orders, shipment, warehouse). The central area contains a SQL editor with the following code:

```
86 • INSERT INTO TEXT VALUES(7, "TREES & GRAPHS", "PRINCETON", "SADGE");  
87 • INSERT INTO BOOK_ADOPTION VALUES(11, 4, 8);  
88  
89 • SELECT * FROM BOOK_ADOPTION;  
90  
91 • SELECT * FROM TEXT;  
92  
93 -- Produce a list of text books (include Course #, Book-ISBN, Book-title) in the alphabetical order for courses offered by t
```

Below the SQL editor is a Result Grid showing the data from the BOOK_ADOPTION table:

COURSE_ID	SEM	BOOK_ISBN
11	4	1
11	4	2
44	6	3
44	6	4
55	2	5
22	6	6
55	2	7
11	4	8
HOLE	HOLE	HOLE

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- Filter objects
- banking
- bookdealer
- database1
- insurance
- order_processing
- Tables
- customer
- item
- order_item
- orders
- shipment
- warehouse
- Views
- Stored Procedures
- Functions
- sys

Administration Schemas Information

OrderProcessing BankingDatabase StudentEnrollment*

```

86 • INSERT INTO TEXT VALUES(7, "TREES & GRAPHS", "PRINCETON", "SADGE");
87 • INSERT INTO BOOK_ADOPTION VALUES(11, 4, 8);
88
89 • SELECT * FROM BOOK_ADOPTION;
90
91 • SELECT * FROM TEXT;
92
93 -- Produce a list of text books (include Course #, Book-ISBN, Book-title) in the alphabetical order for courses offered by t

```

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

BOOK_ISBN	BOOK_TITLE	PUBLISHER	AUTHOR
1	DS and C	Princeton	Padma Reddy
2	Fundamentals of DS	Princeton	Gode
3	Fundamentals of DBMS	Princeton	Navathe
4	SQL	Princeton	Foley
5	Electronic circuits	TMH	Elnasri
6	Adv unix prog	TMH	Stevens
7	TREES & GRAPHS	PRINCETON	SADGE

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- Filter objects
- banking
- bookdealer
- database1
- insurance
- order_processing
- Tables
- customer
- item
- order_item
- orders
- shipment
- warehouse
- Views
- Stored Procedures
- Functions
- sys

Administration Schemas Information

OrderProcessing BankingDatabase StudentEnrollment*

```

89 • SELECT * FROM BOOK_ADOPTION;
90
91 • SELECT * FROM TEXT;
92
93 -- Produce a list of text books (include Course #, Book-ISBN, Book-title) in the alphabetical order for courses offered by t
94 • SELECT C.COURSE_ID,T.BOOK_ISBN,T.BOOK_TITLE FROM TEXT T,COURSE C,BOOK_ADOPTION B WHERE T.BOOK_ISBN=B.BOOK_ISBN AND
95 B.COURSE_ID=C.COURSE_ID AND C.DEP="CS" AND (SELECT COUNT(B.BOOK_ISBN) FROM BOOK_ADOPTION B WHERE
96 C.COURSE_ID=B.COURSE_ID)>=2 ORDER BY T.BOOK_TITLE;
97
98

```

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

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

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- Filter objects
- banking
- bookdealer
- database1
- insurance
- order_processing
- Tables
- customer
- item
- order_item
- orders
- shipment
- warehouse
- Views
- Stored Procedures
- Functions
- sys

Administration Schemas Information

OrderProcessing BankingDatabase StudentEnrollment*

```

103 ( SELECT C.DEP
104 FROM COURSE C,BOOK_ADOPTION B,TEXT T
105 WHERE C.COURSE_ID=B.COURSE_ID
106 AND T.BOOK_ISBN=B.BOOK_ISBN
107 AND T.PUBLISHER='Princeton')
108 AND C.DEP NOT IN
109 (SELECT C.DEP
110 FROM COURSE C,BOOK_ADOPTION B,TEXT T
111 WHERE C.COURSE_ID=B.COURSE_ID
112 AND T.BOOK_ISBN=B.BOOK_ISBN
113 AND T.PUBLISHER != 'Princeton');

```

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

DEPT
CS

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: OrderProcessing BankingDatabase StudentEnrollment*

SCHEMAS: Filter objects

- banking
- bookdealer
- database1
- insurance
- order_processing
- Tables: customer, item, order_item, orders, shipment, warehouse
- Views
- Stored Procedures
- Functions
- sys

Administration Schemas Information

```

55 • INSERT INTO COURSE(COURSE_ID, CNAME, DEPT) VALUES (22, 'USP', 'IS');
56 • INSERT INTO COURSE(COURSE_ID, CNAME, DEPT) VALUES (33, 'SNS', 'EC');
57 • INSERT INTO COURSE(COURSE_ID, CNAME, DEPT) VALUES (44, 'DBMS', 'CS');
58 • INSERT INTO COURSE(COURSE_ID, CNAME, DEPT) VALUES (55, 'EC', 'TC');
59 • SELECT * FROM COURSE;
60
61 • INSERT INTO ENROLL(REG_NO, COURSE_ID, SEM, MARKS) VALUES ('CS01', 11, 4, 85);
62 • INSERT INTO ENROLL(REG_NO, COURSE_ID, SEM, MARKS) VALUES ('IS02', 22, 6, 80);
63 • INSERT INTO ENROLL(REG_NO, COURSE_ID, SEM, MARKS) VALUES ('EC03', 33, 2, 80);

```

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

COURSE_ID	CNAME	DEPT
11	DS	CS
22	USP	IS
33	SNS	EC
44	DBMS	CS
55	EC	TC
NULL	NULL	NULL

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: OrderProcessing BankingDatabase StudentEnrollment*

SCHEMAS: Filter objects

- banking
- bookdealer
- database1
- insurance
- order_processing
- Tables: customer, item, order_item, orders, shipment, warehouse
- Views
- Stored Procedures
- Functions
- sys

Administration Schemas Information

```

39 • CREATE TABLE TEXT(
    BOOK_ISBN INT,
    BOOK_TITLE VARCHAR(30),
    PUBLISHER VARCHAR(30),
    AUTHOR VARCHAR(30),
    FOREIGN KEY(BOOK_ISBN) REFERENCES BOOK_ADOPTION(BOOK_ISBN) ON DELETE CASCADE ON UPDATE CASCADE
);
46 • show tables;
47 • INSERT INTO STUDENT(REG_NO, SNAME, MAJOR, BDATE) VALUES ('CS01', 'RAM', '1986-03-12'), ('CS02', 'SNEHA', 'DBMS', '1987-01-01'), ('EC03', 'AHMED', 'SNS', '1985-04-17'), ('IS02', 'SMITH', 'USP', '1987-12-23'), ('TC05', 'AKHILA', 'EC', '1986-10-06');

```

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

Tables_in_studentenrollment			
book_adoption	course	enroll	student
text			

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: OrderProcessing BankingDatabase StudentEnrollment*

SCHEMAS: Filter objects

- banking
- bookdealer
- database1
- insurance
- order_processing
- Tables: customer, item, order_item, orders, shipment, warehouse
- Views
- Stored Procedures
- Functions
- sys

Administration Schemas Information

```

49 • INSERT INTO STUDENT(REG_NO, SNAME, MAJOR, BDATE) VALUES ('EC03', 'AHMED', 'SNS', '1985-04-17');
50 • INSERT INTO STUDENT(REG_NO, SNAME, MAJOR, BDATE) VALUES ('CS03', 'SNEHA', 'DBMS', '1987-01-01');
51 • INSERT INTO STUDENT(REG_NO, SNAME, MAJOR, BDATE) VALUES ('TC05', 'AKHILA', 'EC', '1986-10-06');
52 • SELECT * FROM STUDENT;
53
54 • INSERT INTO COURSE(COURSE_ID, CNAME, DEPT) VALUES (11, 'DS', 'CS');
55 • INSERT INTO COURSE(COURSE_ID, CNAME, DEPT) VALUES (22, 'USP', 'IS');
56 • INSERT INTO COURSE(COURSE_ID, CNAME, DEPT) VALUES (33, 'SNS', 'EC');
57 • INSERT INTO COURSE(COURSE_ID, CNAME, DEPT) VALUES (44, 'DBMS', 'CS');

```

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

REG_NO	SNAME	MAJOR	BDATE
CS01	RAM	DS	1986-03-12
CS03	SNEHA	DBMS	1987-01-01
EC03	AHMED	SNS	1985-04-17
IS02	SMITH	USP	1987-12-23
TC05	AKHILA	EC	1986-10-06
NULL	NULL	NULL	NULL

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: OrderProcessing BankingDatabase StudentEnrollment*

SCHEMAS: Filter objects

- banking
- bookdealer
- database1
- insurance
- order_processing
- Tables: customer, item, order_item, orders, shipment, warehouse, Views, Stored Procedures, Functions
- sys

```

58 • INSERT INTO COURSE(COURSE_ID, CNAME, DEPT) VALUES (55, 'EC', 'TC');
59 • SELECT * FROM COURSE;
60
61 • INSERT INTO ENROLL(REG_NO, COURSE_ID, SEM, MARKS) VALUES ('CS01', 11, 4, 85);
62 • INSERT INTO ENROLL(REG_NO, COURSE_ID, SEM, MARKS) VALUES ('IS02', 22, 6, 80);
63 • INSERT INTO ENROLL(REG_NO, COURSE_ID, SEM, MARKS) VALUES ('EC03', 33, 2, 80);
64 • INSERT INTO ENROLL(REG_NO, COURSE_ID, SEM, MARKS) VALUES ('CS03', 44, 6, 75);
65 • INSERT INTO ENROLL(REG_NO, COURSE_ID, SEM, MARKS) VALUES ('TC05', 55, 2, 8);
66 • SELECT * FROM ENROLL;

```

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

REG_NO	COURSE_ID	SEM	MARKS
CS01	11	4	85
IS02	22	6	80
EC03	33	2	80
CS03	44	6	75
TC05	55	2	8

Administration Schemas Information

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: OrderProcessing BankingDatabase StudentEnrollment*

SCHEMAS: Filter objects

- banking
- bookdealer
- database1
- insurance
- order_processing
- Tables: customer, item, order_item, orders, shipment, warehouse, Views, Stored Procedures, Functions
- sys

```

70 • INSERT INTO BOOK_ADOPTION(COURSE_ID, SEM, BOOK_ISBN) VALUES (44, 6, 3);
71 • Execute the selected portion of the script or everything, if there is no selection;
72 • INSERT INTO BOOK_ADOPTION(COURSE_ID, SEM, BOOK_ISBN) VALUES (44, 6, 4);
73 • INSERT INTO BOOK_ADOPTION(COURSE_ID, SEM, BOOK_ISBN) VALUES (55, 2, 5);
74 • INSERT INTO BOOK_ADOPTION(COURSE_ID, SEM, BOOK_ISBN) VALUES (22, 6, 6);
75 • INSERT INTO BOOK_ADOPTION(COURSE_ID, SEM, BOOK_ISBN) VALUES (55, 2, 7);
76
77 • INSERT INTO TEXT(BOOK_ISBN, BOOK_TITLE, PUBLISHER, AUTHOR) VALUES (1, 'DS and C', 'Princeton', 'Padma Reddy');
78 • INSERT INTO TEXT(BOOK_ISBN, BOOK_TITLE, PUBLISHER, AUTHOR) VALUES (2, 'Fundamentals of DS', 'Princeton', 'Godse');

```

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

COURSE_ID	SEM	BOOK_ISBN
11	4	1
11	4	2
44	6	3
44	6	4
55	2	5
22	6	6
55	2	7
HULL	HULL	HULL

BOOK_ADOPTION 5 x

Apply | Revert

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: OrderProcessing BankingDatabase StudentEnrollment*

SCHEMAS: Filter objects

- banking
- bookdealer
- database1
- insurance
- order_processing
- Tables: customer, item, order_item, orders, shipment, warehouse, Views, Stored Procedures, Functions
- sys

```

79 • INSERT INTO TEXT(BOOK_ISBN, BOOK_TITLE, PUBLISHER, AUTHOR) VALUES (3, 'Fundamentals of DBMS', 'Princeton', 'Navathe');
80 • INSERT INTO TEXT(BOOK_ISBN, BOOK_TITLE, PUBLISHER, AUTHOR) VALUES (4, 'SQL', 'Princeton', 'Foley');
81 • INSERT INTO TEXT(BOOK_ISBN, BOOK_TITLE, PUBLISHER, AUTHOR) VALUES (5, 'Electronic circuits', 'TMH', 'Elmasri');
82 • INSERT INTO TEXT(BOOK_ISBN, BOOK_TITLE, PUBLISHER, AUTHOR) VALUES (6, 'Adv unix prog', 'TMH', 'Stevens');
83 • SELECT * FROM TEXT;
84
85 -- Demonstrate how you add a new text book to the database and make this book be adopted by some department.
86 • INSERT INTO TEXT VALUES(7, "TREES & GRAPHS", "PRINCETON", "SADGE");
87 • INSERT INTO BOOK_ADOPTION VALUES(11, 4, 8);

```

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

BOOK_ISBN	BOOK_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

Administration Schemas Information

PROGRAM-6:MOVIE DATABASE

QUESTION:

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.

PROGRAM CODE:

```
create database movie;
```

```
use movie;
```

```
CREATE TABLE ACTOR (
```

```
ACT_ID INT,
```

```
ACT_NAME VARCHAR (20),
```

```
ACT_GENDER CHAR (1),
```

```
PRIMARY KEY (ACT_ID));
```

```
CREATE TABLE DIRECTOR (
```

```
DIR_ID INT,
```

```
DIR_NAME VARCHAR (20),  
DIR_PHONE real,  
PRIMARY KEY (DIR_ID));
```

```
CREATE TABLE MOVIES (  
MOV_ID INT,  
MOV_TITLE VARCHAR (25),  
MOV_YEAR INT,  
MOV_LANG VARCHAR (12),  
DIR_ID INT,  
PRIMARY KEY (MOV_ID),  
FOREIGN KEY (DIR_ID) REFERENCES DIRECTOR (DIR_ID));
```

```
CREATE TABLE MOVIE_CAST (  
ACT_ID INT,  
MOV_ID INT,  
ROLE VARCHAR(10),  
PRIMARY KEY (ACT_ID, MOV_ID),  
FOREIGN KEY (ACT_ID) REFERENCES ACTOR (ACT_ID),  
FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));
```

```
CREATE TABLE RATING (  
MOV_ID INT,  
REV_STARS VARCHAR (25),  
PRIMARY KEY (MOV_ID),  
FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));
```

```
INSERT INTO ACTOR VALUES (301,'ANUSHKA','F');
```

```
INSERT INTO ACTOR VALUES (302,'PRABHAS','M');
INSERT INTO ACTOR VALUES (303,'PUNITH','M');
INSERT INTO ACTOR VALUES (304,'JERMY','M');

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

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

INSERT INTO MOVIE_CAST VALUES (301, 1002, 'HEROINE');
INSERT INTO MOVIE_CAST VALUES (301, 1001, 'HEROINE');
INSERT INTO MOVIE_CAST VALUES (303, 1003, 'HERO');
INSERT INTO MOVIE_CAST VALUES (303, 1002,'GUEST');
INSERT INTO MOVIE_CAST VALUES (304, 1004, 'HERO');

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

SELECT * FROM ACTOR;

/*1. List the titles of all movies directed by'Hitchcock'.*/
SELECT MOV_TITLE
```

```
FROM MOVIES  
WHERE DIR_ID IN (SELECT DIR_ID  
FROM DIRECTOR  
WHERE DIR_NAME = 'HITCHCOCK');
```

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

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

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

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

/*4. Find the title of movies and number of stars for each movie that has at least
one rating and find the highest number of stars that movie received. Sort the
result by movie title. */

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

```
BY MOV_TITLE  
HAVING MAX(REV_STARS)>0  
ORDER BY MOV_TITLE;
```

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

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

OUTPUT SCREENSHOTS:

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Labtest_Banking project MovieDatabase ×

SCHEMAS

Filter objects

- banking
- bookdealer
- database1
- insurance
- labtest_banking
- onlineprojectsubmissionportal
- order_processing
- Tables
- Views
- Stored Procedures
- Functions
- studentenrollment
- sys

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

```

86
87 • SELECT ACT_NAME, MOV_TITLE, MOV_YEAR
88 FROM ACTOR A JOIN
89 MOVIE_CAST C
90 ON A.ACT_ID=C.ACT_ID
91 JOIN MOVIES M
92 ON C.MOV_ID=M.MOV_ID
93 WHERE M.MOV_YEAR NOT BETWEEN 2000 AND 2015;

```

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

ACT_NAME	MOV_TITLE	MOV_YEAR
ANUSHKA	BAHUBALI-2	2017

Administration Schemas

Information

No object selected

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Labtest_Banking project MovieDatabase ×

SCHEMAS

Filter objects

- banking
- bookdealer
- database1
- insurance
- labtest_banking
- onlineprojectsubmissionportal
- order_processing
- Tables
- Views
- Stored Procedures
- Functions
- studentenrollment
- sys

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

```

96
97
98 • SELECT MOV_TITLE, MAX(REV_STARS) FROM MOVIES
99 INNER JOIN RATING USING (MOV_ID) GROUP BY MOV_TITLE
100 HAVING MAX(REV_STARS)>0
101 ORDER BY MOV_TITLE;
102
103
104 /* Indexes creation of all tables affected by following statements to EX */

```

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

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

Administration Schemas

Information

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Labtest_Banking project MovieDatabase* ×

```

102 ORDER BY MOV_TITLE;

103

104 /*5. Update rating of all movies directed by 'Steven Spielberg' to 5*/
105 • UPDATE RATING
106 SET REV_STARS='5'
107 WHERE MOV_ID = (SELECT MOV_ID FROM MOVIES
108 WHERE DIR_ID = (SELECT DIR_ID FROM DIRECTOR
109 WHERE DIR_NAME ='STEVEN SPIELBERG'));
110 • select * from rating;

```

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

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

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Labtest_Banking project MovieDatabase* ×

```

28 PRIMARY KEY (ACT_ID, MOV_ID),
29 FOREIGN KEY (ACT_ID) REFERENCES ACTOR (ACT_ID),
30 FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID);

31

32 • CREATE TABLE RATING (
33 MOV_ID INT,
34 REV_STARS VARCHAR (25),
35 PRIMARY KEY (MOV_ID),
36 FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));

```

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

Tables_in_movie
actor
director
movie_cast
movies
rating

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Labtest_Banking project MovieDatabase* ×

```

37 • show tables;
38

39 • INSERT INTO ACTOR VALUES (301,'ANUSHKA','F');
40 INSERT INTO ACTOR VALUES (302,'PRABHAS','M');
41 INSERT INTO ACTOR VALUES (303,'PUNITH','M');
42 • INSERT INTO ACTOR VALUES (304,'JERMY','M');
43 • SELECT * FROM ACTOR;
44

45 INSERT INTO DIRECTOR VALUES (60,'RAJAMOULI', 8751611001);
46 INSERT INTO DIRECTOR VALUES (61,'UTTAMARAJ', 7756420011);

```

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

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

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Labtest_Banking project MovieDatabase

SCHEMAS

- banking
- bookdealer
- database1
- insurance
- labtest_banking
- onlineprojectsubmissionportal
- order_processing
 - Tables
 - Views
 - Stored Procedures
 - Functions
- studentenrollment
- sys

Limit to 1000 rows ▾

```

46 • INSERT INTO DIRECTOR VALUES (61,'HITCHCOCK', 7766138911);
47 • INSERT INTO DIRECTOR VALUES (62,'FARAN', 9986776531);
48 • INSERT INTO DIRECTOR VALUES (63,'STEVEN SPIELBERG', 8989776530);
49 • SELECT * FROM DIRECTOR;
50
51 • INSERT INTO MOVIES VALUES (1001,'BAHUBALI-2', 2017, 'TELUGU', 60);
52 • INSERT INTO MOVIES VALUES (1002,'BAHUBALI-1', 2015, 'TELUGU', 60);
53 • INSERT INTO MOVIES VALUES (1003,'AKASH', 2008, 'KANNADA', 61);
54 • INSERT INTO MOVIES VALUES (1004,'WAR HORSE', 2011, 'ENGLISH', 63);
55 • SELECT * FROM MOVIES;

```

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

DIR_ID	DIR_NAME	DIR_PHONE
60	RAJAMOLI	8751611001
61	HITCHCOCK	7766138911
62	FARAN	9986776531
63	STEVEN SPIELBERG	8989776530
•	MULL	MULL

Administration Schemas Information

Result Grid Form Editor

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Labtest_Banking project MovieDatabase

SCHEMAS

- banking
- bookdealer
- database1
- insurance
- labtest_banking
- onlineprojectsubmissionportal
- order_processing
 - Tables
 - Views
 - Stored Procedures
 - Functions
- studentenrollment
- sys

Limit to 1000 rows ▾

```

52 • INSERT INTO MOVIES_VALUES (1002,'BAHUBALI-1', 2015, 'TELUGU', 60);
53 • INSERT INTO MOVIES_VALUES (1003,'AKASH', 2008, 'KANNADA', 61);
54 • INSERT INTO MOVIES_VALUES (1004,'WAR HORSE', 2011, 'ENGLISH', 63);
55 • SELECT * FROM MOVIES;
56
57 • INSERT INTO MOVIE_CAST VALUES (301, 1002, 'HEROINE');
58 • INSERT INTO MOVIE_CAST VALUES (301, 1001, 'HEROINE');
59 • INSERT INTO MOVIE_CAST VALUES (303, 1003, 'HERO');
60 • INSERT INTO MOVIE_CAST VALUES (303, 1002, 'GUEST');
61 • INSERT INTO MOVIE_CAST VALUES (304, 1004, 'HERO');

```

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

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

Administration Schemas Information

Result Grid Form Editor

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Labtest_Banking project MovieDatabase

SCHEMAS

- banking
- bookdealer
- database1
- insurance
- labtest_banking
- onlineprojectsubmissionportal
- order_processing
 - Tables
 - Views
 - Stored Procedures
 - Functions
- studentenrollment
- sys

Limit to 1000 rows ▾

```

61 • INSERT INTO MOVIE_CAST VALUES (304, 1004, 'HERO');
62 • SELECT * FROM MOVIE_CAST;
63
64 • INSERT INTO RATING VALUES (1001,'4');
65 • INSERT INTO RATING VALUES (1002,'2');
66 • INSERT INTO RATING VALUES (1003, '5');
67 • INSERT INTO RATING VALUES (1004, '4');
68 • SELECT * FROM RATING;
69
70

```

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

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

Administration Schemas Information

Result Grid Form Editor

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Labtest_Banking project MovieDatabase ×

Limit to 1000 rows

```

61 • INSERT INTO MOVIE_CAST VALUES (304, 1004, 'HERO');
62 • SE Execute the selected portion of the script or everything, if there is no selection
63
64 • INSERT INTO RATING VALUES (1001,'4');
65 • INSERT INTO RATING VALUES (1002,'2');
66 • INSERT INTO RATING VALUES (1003, '5');
67 • INSERT INTO RATING VALUES (1004, '4');
68 • SELECT * FROM RATING;
69
70 /*1. List the titles of all movies directed by 'Kusturka' */

```

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

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

Administration Schemas Information

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Labtest_Banking project MovieDatabase ×

Limit to 1000 rows

```

70 /*1. List the titles of all movies directed by 'Hitchcock'.*/
71 • SELECT MOV_TITLE
72   FROM MOVIES
73   WHERE DIR_ID IN (SELECT DIR_ID
74     FROM DIRECTOR
75     WHERE DIR_NAME = 'HITCHCOCK');
76
77 /*2. Find the movie names where one or more actors acted in two or more movies.*/
78 • SELECT MOV_TITLE
79   FROM MOVIES M, MOVIE_CAST MV

```

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

MOV_TITLE
AKASH

Administration Schemas Information

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator: Labtest_Banking project MovieDatabase ×

Limit to 1000 rows

```

76
77 /*2. Find the movie names where one or more actors acted in two or more movies.*/
78 • SELECT MOV_TITLE
79   FROM MOVIES M, MOVIE_CAST MV
80   WHERE M.MOV_ID=MV.MOV_ID AND ACT_ID IN (SELECT ACT_ID
81     FROM MOVIE_CAST GROUP BY ACT_ID HAVING
82     COUNT(ACT_ID)>1)
83   GROUP BY MOV_TITLE HAVING COUNT(MOV_TITLE)>1;
84
85 /*3. List all actors who acted in a movie before 2000 and also in a movie after

```

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

MOV_TITLE
BAHUBALI-1

Administration Schemas Information

PROGRAM 7: AIRLINE FLIGHT DATABASE

QUESTION:

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.

PROGRAM CODE:

```
create database airline;
use airline;
CREATE TABLE flights(
flno Int,
`from` Varchar(20),
`to` Varchar(20),
distance INT,
departs time,
arrives time,
price Int,
PRIMARY KEY(flno) );
CREATE TABLE aircraft(
aid INT,
aname VARCHAR(20),
cruisingrange INT,
PRIMARY KEY (aid) );
CREATE TABLE employees(
eid INT,
ename Varchar(20),
salary INT,
PRIMARY KEY (eid) );
CREATE TABLE certified(
eid INT,
aid INT,
PRIMARY KEY (eid,aid),
FOREIGN KEY (eid) REFERENCES employees (eid),
FOREIGN KEY (aid) REFERENCES aircraft (aid) );
show tables;
INSERT INTO flights (flno,`from`, `to`,distance,departs,arrives,price) VALUES
(1,'Bangalore','Chennai',360,'08:45','10:00',10000),
(2,'Bangalore','Delhi',1700,'12:15','15:00',37000),
(3,'Bangalore','Kolkata',1500,'15:15','05:25',30000),
(4,'Mumbai','Delhi',1200,'10:30','12:30',28000),
(5,'Bangalore','New york',14000,'05:45','02:30',90000),
(6,'Delhi','Chicago',12000,'10:00','05:45',95000),
(7,'Bangalore','Frankfurt',15000,'12:00','06:30',98000),
(8,'Madison','New york',1500,'10:15','14:25',30000);
SELECT * FROM flights;
INSERT INTO aircraft (aid,aname,cruisingrange) values
(1,'Airbus 380',1000),
(2,'Boeing 737',4000),
(3,'Lockheed',5500),
(4,'Airbus A220',9500),
```

```

(5,'Boeing 747',800),
(6,'Douglas DC3',900);
SELECT * FROM aircraft;
INSERT INTO employees (eid,ename,salary) VALUES
(1,'Zoya',95000),
(2,'Akshay',65000),
(3,'Niveditha',70000),
(4,'Safan',45000),
(5,'Peter',95000),
(6,'Nayan',100000),
(7,'Ajay',50000);
SELECT * FROM employees;
INSERT INTO certified (eid,aid) VALUES
(1,1),
(1,3),
(1,4),
(5,4),
(5,3),
(1,2),
(2,6),
(2,5),
(4,5),
(6,4),
(6,3),
(3,6),
(3,2);
SELECT * FROM certified;
#i. Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.80,000.
SELECT DISTINCT A.aname
FROM Aircraft A
WHERE A.Aid IN (SELECT C.aid
FROM Certified C, Employees E
WHERE C.eid = E.eid AND
NOT EXISTS ( SELECT *
FROM Employees E1
WHERE E1.eid = E.eid AND E1.salary < 80000 ));
#ii. For each pilot who is certified for more than three aircrafts, find the eid and the maximum cruising range of the aircraft for which she or he is certified.
SELECT C.eid, MAX(A.cruisingrange)
FROM Certified C, Aircraft A
WHERE C.aid = A.aid

```

GROUP BY C.eid

HAVING COUNT(*) > 3;

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

Bengaluru toFrankfurt.

SELECT DISTINCT e.ename

FROM employees e

WHERE e.salary<

(SELECT MIN(f.price)

FROM flights f

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

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

average salary of all pilots certified for this aircraft.

SELECT a.aid,a.ename,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.ename;

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

SELECT distinct e.ename

FROM employees e,aircraft a,certified c

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

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

SELECT a.aid

FROM aircraft a

WHERE a.cruisingrange>

(SELECT MIN(f.distance)

FROM flights f

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

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

flight. List the choice of departure times from Madison if the customer wants to arrive in New

York by 6 p.m.

SELECT F.departs

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

FROM Flights F0

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

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

for pilots.0

```

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

```

OUTPUT SCREENSHOTS:

The screenshot shows the MySQL Workbench interface. The top window is titled "Airline" and contains a SQL query editor with the following code:

```

29 FOREIGN KEY (aid) REFERENCES aircraft (aid) ;
30
31 • show tables;
32
33 • INSERT INTO flights (flno,`from`, `to`,distance,departs,arrives,price) VALUES
34 (1,'Bangalore','Chennai',360,'08:45','10:00',10000),
35 (2,'Bangalore','Delhi',1700,'12:15','15:00',37000),
36 (3,'Bangalore','Kolkata',1500,'15:15','05:25',30000),
37 (4,'Mumbai','Delhi',1200,'10:30','12:30',28000),
38 (5,'Bangalore','New york',14000,'05:45','02:30',90000),
-- -----

```

The "Result Grid" tab is selected at the bottom, showing the results of the "show tables;" command:

Tables_in_airline
aircraft
certified
employees
flights

Airline x

35 (2, 'Bangalore', 'Delhi', 1700, '12:15', '15:00', 37000),
 36 (3, 'Bangalore', 'Kolkata', 1500, '15:15', '05:25', 30000),
 37 (4, 'Mumbai', 'Delhi', 1200, '10:30', '12:30', 28000),
 38 (5, 'Bangalore', 'New york', 14000, '05:45', '02:30', 90000),
 39 (6, 'Delhi', 'Chicago', 12000, '10:00', '05:45', 95000),
 40 (7, 'Bangalore', 'Frankfurt', 15000, '12:00', '06:30', 98000),
 41 (8, 'Madison', 'New york', 1500, '10:15', '14:25', 30000);
 42 • SELECT * FROM flights;

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

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

flights 2 x

Airline x

47 (3, 'Lockheed', 5500),
 48 (4, 'Airbus A220', 9500),
 49 (5, 'Boeing 747', 800),
 50 (6, 'Douglas DC3', 900);
 51 • SELECT * FROM aircraft;
 52
 53 • INSERT INTO employees (eid,ename,salary) VALUES
 (1, 'Zoya', 95000),
 54 (2, 'John', 150000);
 55 • UPDATE employees SET salary = 100000 WHERE eid = 2;

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

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

aircraft 3 x

Airline x

```
56      (3,'Niveditha',70000),
57      (4,'Safan',45000),
58      (5,'Peter',95000),
59      (6,'Nayan',100000),
60      (7,'Ajay',50000);
61 •  SELECT * FROM employees;
62
63 •  INSERT INTO certified (eid,aid) VALUES
  
```

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

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

employees 4 x

Airline x

```
74      (6,3),
75      (3,6),
76      (3,2);
77 •  SELECT * FROM certified;
78
79
  
```

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

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

certified 5 x

Airline x

```

80      #i. Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.80,000.
81 •   SELECT DISTINCT A.aname
82     FROM Aircraft A
83     WHERE A.aid IN (SELECT C.aid
84       FROM Certified C, Employees E
85       WHERE C.eid = E.eid AND
86       NOT EXISTS ( SELECT *
87         FROM Employees E1
88         WHERE E1.eid = E.eid AND E1.salary < 80000 ));
89
90      #ii. For each pilot who is certified for more than three aircrafts, find the eid and the maximum cruising range of .

```

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

	aname
▶	Airbus 380
	Boeing 737
	Lockheed
	Airbus A220

Aircraft 6 x

Airline x

```

89
90      #ii. For each pilot who is certified for more than three aircrafts, find the eid and
91 •   SELECT C.eid, MAX(A.cruisingrange)
92     FROM Certified C, Aircraft A
93     WHERE C.aid = A.aid
94     GROUP BY C.eid
95     HAVING COUNT(*) > 3;
96
97      #iii. Find the names of pilots whose salary is less than the price of the cheapest rc
98 •   SELECT DISTINCT e.ename
99     FROM employees e

```

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

	eid	MAX(A.cruisingrange)
▶	1	9500

Result 7 x

Airline x

```

95      HAVING COUNT(*) > 3;
96
97      #iii. Find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru toFrankfurt.
98 •   SELECT DISTINCT e.ename
99      FROM employees e
100     WHERE e.salary<
101        (SELECT MIN(f.price)
102         FROM flights f
103        WHERE f.from='Bangalore' AND f.to='Frankfurt');
104
105      #iv. For all aircraft with cruising range over 1000 Kms, find the name of the aircraft and the average salary of all i

```

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

	ename
▶	Zoya
	Akshay
	Niveditha
	Safan
	Peter
	Ajay

employees 8 x

Airline x

```

104
105      #iv. For all aircraft with cruising range over 1000 Kms, find the name
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 names of pilots certified for some Boeing aircraft.
114 •   SELECT distinct e.ename

```

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

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

Result 9 x

Airline ×

113 #v. Find the names of pilots certified for some Boeing aircraft.
114 • SELECT distinct e.ename
115 FROM employees e,aircraft a,certified c
116 WHERE e.eid=c.eid AND c.aid=a.aid AND a.ename like 'Boeing%';
117
118 #vi. Find the aids of all aircraft that can be used on routes from Bengaluru
119 • SELECT a.aid
120 FROM aircraft a
121 WHERE a.cruisingrange>
122 (SELECT MIN(f.distance)
123 FROM flights f

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

	ename
▶	Zoya
	Niveditha
	Akshay
	Safan

Result 10 ×

Airline ×

116 WHERE e.eid=c.eid AND c.aid=a.aid AND a.ename like 'Boeing%';
117
118 #vi. Find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi.
119 • SELECT a.aid
120 FROM aircraft a
121 WHERE a.cruisingrange>
122 (SELECT MIN(f.distance)
123 FROM flights f
124 WHERE f.from='Bangalore' AND f.to='Delhi');
125
126 #vii. A customer wants to travel from Madison to New York with no more than two changes of f

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

	aid
▶	2
	3
	4
*	NULL

aircraft 11 ×

Airline x

```

122  (SELECT MIN(f.distance)
123   FROM flights f
124   WHERE f.from='Bangalore' AND f.to='Delhi');
125
126  #vii. A customer wants to travel from Madison to New York with no more than two changes
127 •  SELECT F.departs
128   FROM Flights F WHERE F.flno IN (  SELECT F0.flno
129   FROM Flights F0
130   WHERE F0.from = 'Madison' AND F0.to = 'New york' AND F0.arrives < '18:00' );
131
132  #viii. Print the name and salary of every non-pilot whose salary is more than the avera
<

```

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

departs
10:15:00

Flights 12 x

Airline x

```

132  #viii. Print the name and salary of every non-pilot whose salary is more than the average salary for pilots.0
133 •  SELECT E.ename, E.salary
134   FROM Employees E
135   WHERE E.eid NOT IN ( SELECT DISTINCT C.eid
136   FROM Certified C )
137   AND E.salary > ( SELECT AVG (E1.salary)
138   FROM Employees E1
139   WHERE E1.eid IN
140   ( SELECT DISTINCT C1.eid
141   FROM Certified C1 ) );

```

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

ename	salary
-------	--------

Employees 13 x

PROGRAM-8:SUPPLIERS DATABASE

QUESTION :

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.

PROGRAM 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 );

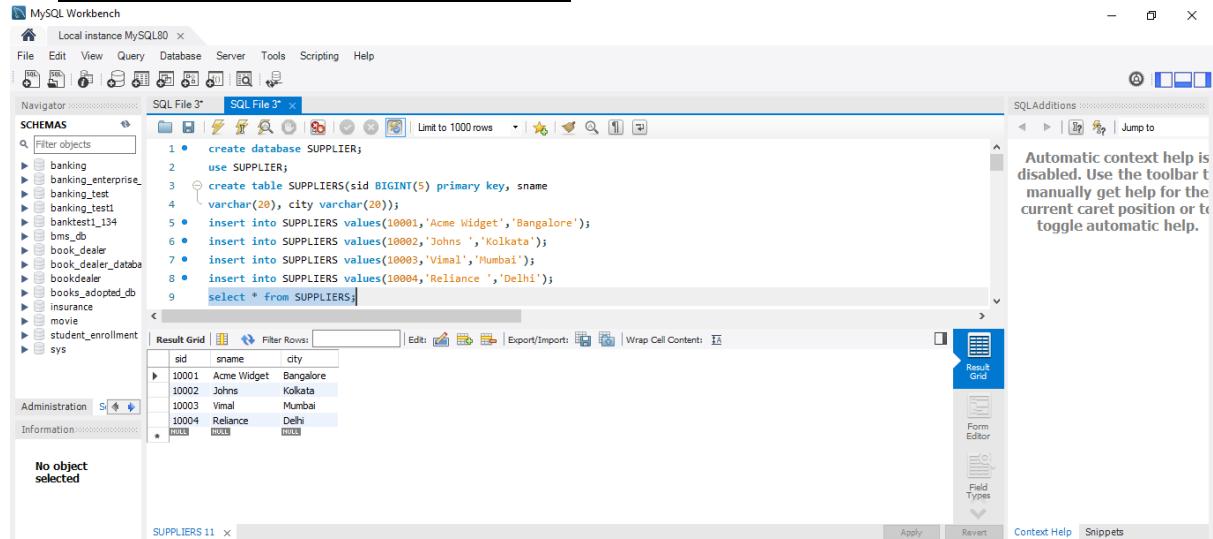
/* FOR EACH PART, FIND THE SNAME OF THE SUPPLIER WHO CHARGES THE MOST FOR THAT PART.*/
SELECT P.PID, S.SNAME
FROM PARTS P, SUPPLIERS S, CATALOG C
WHERE C.PID = P.PID
AND C.SID = S.SID
AND C.COST = (SELECT MAX(C1.COST)
FROM CATALOG C1
WHERE C1.PID = P.PID);

/* 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' );

```

SCREENSHOTS OF THE PROGRAM OUTPUT :

- CREATION AND INSERTION OF VALUES**



The screenshot shows the MySQL Workbench interface with the following details:

- File Menu:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard MySQL Workbench toolbar.
- Navigator:** Shows the database schema with the 'SUPPLIERS' table selected.
- SQL Editor:** Contains the SQL code for creating the 'SUPPLIERS' table and inserting four rows of data.
- Result Grid:** Displays the inserted data in a grid format.
- SQLAdditions:** A panel on the right with the message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

sid	sname	city
10001	Acme Widget	Bangalore
10002	Johns	Kolkata
10003	Vimal	Mumbai
10004	Reliance	Delhi
NULL	NULL	NULL

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator Create a new SQL tab for executing queries SQL File 3*

Limit to 1000 rows

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

SCHEMAS

- banking
- banking_enterprise_
- banking_test
- banking_test1
- banktest1_134
- bms_db
- book_dealer
- book_dealer_database
- bookdealer
- books_adopted_db
- insurance
- movie
- student_enrollment
- sys

No object selected

Administration Information

Result Grid

pid	pname	color
20001	Book	RED
20002	Pen	RED
20003	Pencil	Green
20004	Mobile	Green
20005	Charger	Black
*	NULL	NULL

SQLAdditions

Jump to

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator Create a new SQL tab for executing queries SQL File 3*

Limit to 1000 rows

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

SCHEMAS

- banking
- banking_enterprise_
- banking_test
- banking_test1
- banktest1_134
- bms_db
- book_dealer
- book_dealer_database
- bookdealer
- books_adopted_db
- insurance
- movie
- student_enrollment
- sys

No object selected

Administration Information

Result Grid

sid	pid	cost
10001	20001	10
10001	20002	10
10001	20003	30
10001	20005	10
10002	20001	10
10002	20002	20
10003	20003	30
10004	20003	40
*	NULL	NULL

SQLAdditions

Jump to

1. FIND THE PNAMEs OF PARTS FOR WHICH THERE IS SOME SUPPLIER.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator Open a SQL script file in a new query tab SQL File 3*

Limit to 1000 rows

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

SCHEMAS

- banking
- banking_enterprise_
- banking_test
- banking_test1
- banktest1_134
- bms_db
- book_dealer
- book_dealer_database
- bookdealer
- books_adopted_db
- insurance
- movie
- student_enrollment
- sys

No object selected

Administration Information

Result Grid

pname
Book
Pen
Pencil
Mobile
Charger

SQLAdditions

Jump to

2 - FIND THE SNAMEs OF SUPPLIERS WHO SUPPLY EVERY PART.

MySQL Workbench - Local instance MySQL80

```

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

```

Result Grid

sname
Acme Widget

No object selected

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

MySQL Workbench - Local instance MySQL80

```

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

```

Result Grid

sname
Acme Widget
Johns

No object selected

4 - FIND THE PNAMES OF PARTS SUPPLIED BY ACME WIDGET SUPPLIERS AND BY NO ONE ELSE

MySQL Workbench - Local instance MySQL80

```

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

```

Result Grid

pname
Mobile
Charger

No object selected

5 - FIND THE SIDS OF SUPPLIERS WHO CHARGE MORE FOR SOME PART THAN THE AVERAGE COST OF THAT PART (AVERAGED OVER ALL THE SUPPLIERS WHO SUPPLY THAT PART).

```

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help
Navigator SQL File 3* SQL File 3*
SCHEMAS Filter objects
banking banking_enterprise_ banking_test banking_testl banking1_134 bms_db book_dealer book_dealer_database bookdealer books_adopted_db insurance movie student_enrollment sys
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

```

No object selected

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

sid
10002
10004

SQLAdditions | Jump to

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

6 - FOR EACH PART, FIND THE SNAME OF THE SUPPLIER WHO CHARGES THE MOST FOR THAT PART

```

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help
Navigator SQL File 3* SQL File 3*
SCHEMAS Filter objects
banking banking_enterprise_ banking_test banking_testl banking1_134 bms_db book_dealer book_dealer_database bookdealer books_adopted_db insurance movie student_enrollment sys
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

```

No object selected

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

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

SQLAdditions | Jump to

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

7 - FIND THE SIDS OF SUPPLIERS WHO SUPPLY ONLY RED PARTS

```

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help
Navigator SQL File 3* SQL File 3*
SCHEMAS Filter objects
banking banking_enterprise_ banking_test banking_testl banking1_134 bms_db book_dealer book_dealer_database bookdealer books_adopted_db insurance movie student_enrollment sys
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

```

No object selected

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

sid
10001
10002

SQLAdditions | Jump to

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

PROGRAM-9:STUDENT-FACULTY DATABASE

QUESTION :

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).

PROGRAM CODE :

```
CREATE DATABASE STUDENT_FACULTY;
USE STUDENT_FACULTY;
CREATE TABLE STUDENT(
    SNUM INT,
    SNAME VARCHAR(10),
    MAJOR VARCHAR(2),
    LVL VARCHAR(2),
    AGE INT, PRIMARY KEY(SNUM));

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

CREATE TABLE CLASS(
    CNAME VARCHAR(20),
    METTS_AT TIMESTAMP,
    ROOM VARCHAR(10),
    FID INT,
    PRIMARY KEY(CNAME),
```

```

FOREIGN KEY(FID) REFERENCES FACULTY(FID));

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

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

```

SCREENSHOTS OF THE PROGRAM OUTPUT :

• CREATION AND INSERTION OF VALUES

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local instance MySQL80' and 'MySQL Model' are selected. The main area displays a SQL file named 'SQL File 8' containing the following code:

```
31 • INSERT INTO STUDENT VALUES(2, 'Smith', 'CS', 'Sr', 20);
32 • INSERT INTO STUDENT VALUES(3, 'Jacob', 'CV', 'Sr', 20);
33 • INSERT INTO STUDENT VALUES(4, 'Tom', 'CS', 'Jr', 20);
34 • INSERT INTO STUDENT VALUES(5, 'Rahul', 'CS', 'Jr', 20);
35 • INSERT INTO STUDENT VALUES(6, 'Rita', 'CS', 'Sr', 21);
```

Below the code, a Result Grid shows the inserted data:

snun	sname	major	lM	age
1	Jhon	CS	Sr	19
2	Smith	CS	Jr	20
3	Jacob	CV	Sr	20
4	Tom	CS	Jr	20
5	Rahul	CS	Jr	20
6	Rita	CS	Sr	21

In the bottom pane, the 'Output' tab is active, showing the execution history of the queries:

#	Time	Action	Message	Duration / Fetch
155	12:28:56	SELECT DISTINCT F.fname FROM Faculty F WHERE 5 > (SELECT COUNT(E.enum) FROM Class C, Enrolled E WHERE E.cid = C.cid AND E.sid = F.sid)	4 row(s) returned	0.000 sec / 0.000 sec
156	12:29:03	SELECT DISTINCT S.sname FROM Student S WHERE S.anum NOT IN (SELECT E.enum FROM Enrolled E)	1 row(s) returned	0.000 sec / 0.000 sec
157	12:30:40	SELECT DISTINCT cname FROM class WHERE room=room128 OR cname IN (SELECT e cname FROM enr)	1 row(s) returned	0.000 sec / 0.000 sec
158	12:34:27	SELECT f.fname,f.id FROM faculty f WHERE f.id IN (SELECT fid FROM class GROUP BY fid HAVING COUNT(fid)=1)	1 row(s) returned	0.000 sec / 0.000 sec
159	12:45:08	SELECT S.age, S.M FROM Student S GROUP BY S.age, S.M HAVING S.M IN (SELECT S1.M FROM Student S1 WHERE S1.M < S.M)	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'S.M' at line 1	0.000 sec / 0.000 sec
160	12:48:22	SELECT S.age, S.level FROM STUDENT S GROUP BY S.age, S.level HAVING S.level IN (SELECT S1.level FROM STUDENT S1 WHERE S1.level < S.level)	Error Code: 1054. Unknown column 'S.level' in field list	0.000 sec / 0.000 sec
161	12:48:33	SELECT S.age, S.level FROM STUDENT S GROUP BY S.age, S.level HAVING S.level IN (SELECT S1.level FROM STUDENT S1 WHERE S1.level < S.level)	Error Code: 1054. Unknown column 'S.level' in field list	0.000 sec / 0.000 sec
162	12:49:31	SELECT S.age, S.level FROM STUDENT S GROUP BY S.age, S.level HAVING S.M IN (SELECT S1.M FROM STUDENT S1 WHERE S1.M < S.M)	Error Code: 1054. Unknown column 'S.M' in field list	0.000 sec / 0.000 sec
163	12:49:52	SELECT S.age, S.M FROM STUDENT S GROUP BY S.age, S.M HAVING S.M IN (SELECT S1.M FROM STUDENT S1 WHERE S1.M < S.M)	3 row(s) returned	0.000 sec / 0.000 sec
164	12:56:04	select * from STUDENT LIMIT 0,1000	6 row(s) returned	0.000 sec / 0.000 sec

MySQL Workbench

Local instance MySQL80 x MySQL Model x

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance MySQL80

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

Administration Schemas Information

No object selected

Object Info Session

Query Completed

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance MySQL80

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

Administration Schemas Information

No object selected

Object Info Session

Query Completed

MySQL Workbench

SQL File B

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

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

```

40 • INSERT INTO FACULTY VALUES(14, 'Shiva', 1000);
41 • INSERT INTO FACULTY VALUES(15, 'Nupur', 1000);
42 • select * from FACULTY;
43 • insert into class values('class1', '12/11/15 10:15:16', 'R1', 14);
44 • insert into class values('class10', '12/11/15 10:15:16', 'R12B', 14);

```

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

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

FACULTY 13 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
156	12:29:03	SELECT DISTINCT Sname FROM Student S WHERE Sname NOT IN (SELECT Enam FROM Enrolld E)...	1 row(s) returned	0.000 sec / 0.000 sec
157	12:30:12	SELECT DISTINCT crname FROM class WHERE room=room128 OR crname IN (SELECT e.name FROM enr...	1 row(s) returned	0.000 sec / 0.000 sec
158	12:30:27	SELECT fname,fid FROM faculty f WHERE fid IN (SELECT fid FROM class GROUP BY fid HAVING C...	1 row(s) returned	0.000 sec / 0.000 sec
159	12:45:08	SELECT S.age, SM FROM Student S GROUP BY S.age, SM HAVING SM IN (SELECT S1.M FROM Studen...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se...	0.000 sec
160	12:48:22	SELECT S.age, S.level FROM STUDENT S GROUP BY S.age, S.level HAVING S.level IN (SELECT S1.level...	Error Code: 1054. Unknown column 'S.level' in field list'	0.000 sec
161	12:48:33	SELECT S.age, S.level FROM STUDENT S GROUP BY S.age, S.level HAVING S.level IN (SELECT S1.level...	Error Code: 1054. Unknown column 'S.level' in field list'	0.000 sec
162	12:49:31	SELECT S.age, S.level FROM STUDENT S GROUP BY S.age, S.level HAVING S.level IN (SELECT S1.level...	Error Code: 1054. Unknown column 'S.level' in field list'	0.000 sec
163	12:49:52	SELECT S.age, SM FROM STUDENT S GROUP BY S.age, SM HAVING SM IN (SELECT S1.M FROM STUDENT S...	3 row(s) returned	0.000 sec / 0.000 sec
164	12:56:04	select *from STUDENT LIMIT 0,1000	6 row(s) returned	0.000 sec / 0.000 sec
165	12:56:21	select *from FACULTY LIMIT 0,1000	5 row(s) returned	0.000 sec / 0.000 sec

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

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

class 14 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
157	12:30:40	SELECT DISTINCT crname FROM class WHERE room=room128 OR crname IN (SELECT e.name FROM enr...	1 row(s) returned	0.000 sec / 0.000 sec
158	12:30:27	SELECT f.fname,fid FROM faculty f WHERE fid IN (SELECT fid FROM class GROUP BY fid HAVING C...	1 row(s) returned	0.000 sec / 0.000 sec
159	12:45:08	SELECT S.age, SM FROM Student S GROUP BY S.age, SM HAVING SM IN (SELECT S1.M FROM Studen...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se...	0.000 sec
160	12:48:22	SELECT S.age, S.level FROM STUDENT S GROUP BY S.age, S.level HAVING S.level IN (SELECT S1.level...	Error Code: 1054. Unknown column 'S.level' in field list'	0.000 sec
161	12:48:33	SELECT S.age, S.level FROM STUDENT S GROUP BY S.age, S.level HAVING S.level IN (SELECT S1.level...	Error Code: 1054. Unknown column 'S.level' in field list'	0.000 sec
162	12:49:31	SELECT S.age, S.level FROM STUDENT S GROUP BY S.age, S.level HAVING S.level IN (SELECT S1.level...	Error Code: 1054. Unknown column 'S.level' in field list'	0.000 sec
163	12:49:52	SELECT S.age, SM FROM STUDENT S GROUP BY S.age, SM HAVING SM IN (SELECT S1.M FROM STUDENT S...	3 row(s) returned	0.000 sec / 0.000 sec
164	12:56:04	select *from STUDENT LIMIT 0,1000	6 row(s) returned	0.000 sec / 0.000 sec
165	12:56:21	select *from FACULTY LIMIT 0,1000	5 row(s) returned	0.000 sec / 0.000 sec
166	12:56:30	select *from class LIMIT 0,1000	8 row(s) returned	0.000 sec / 0.000 sec

QUERY 1 = FIND THE NAMES OF ALL JUNIORS (LEVEL = JR) WHO ARE ENROLLED IN A CLASS TAUGHT BY

```

MySQL Workbench - Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator SQL File 3* SQL File 3* 
SCHEMAS Filter objects
banking banking_enterprise_ banking_test banking_test1 banktest1_134 bms_db book_dealer book_dealer_database bookdealer books_adopted_db insurance movie student_enrollment sys
Result Grid Filter Rows: Export: Wrap Cell Content: 
Sname
Tom
SQLAdditions Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```

QUERY 2 = FIND THE NAMES OF ALL CLASSES THAT EITHER MEET IN ROOM R128 OR HAVE FIVE OR MORE STUDENTS ENROLLED.

```

MySQL Workbench - Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator SQL File 3* SQL File 3* 
SCHEMAS Filter objects
banking banking_enterprise_ banking_test banking_test1 banktest1_134 bms_db book_dealer book_dealer_database bookdealer books_adopted_db insurance movie student_enrollment sys
Result Grid Filter Rows: Export/Import: Wrap Cell Content: 
cname
class5
NULL
SQLAdditions Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```

QUERY 3 = FIND THE NAMES OF ALL STUDENTS WHO ARE ENROLLED IN TWO CLASSES THAT MEET AT THE SAME TIME.

```

MySQL Workbench - Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator SQL File 3* SQL File 3* 
SCHEMAS Filter objects
banking banking_enterprise_ banking_test banking_test1 banktest1_134 bms_db book_dealer book_dealer_database bookdealer books_adopted_db insurance movie student_enrollment sys
Result Grid Filter Rows: Export: Wrap Cell Content: 
sname
Rahul
SQLAdditions Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```

QUERY 4 = FIND THE NAMES OF ALL STUDENTS WHO ARE ENROLLED IN TWO CLASSES THAT MEET AT THE SAME TIME.

MySQL Workbench - Local instance MySQL80

```

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 = Find the names of all students who are enrolled in two classes that meet at the same time.
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 = Find the names of faculty members for whom the combined enrolment of the courses that they teach is less Than five.

```

Result Grid

fname	fid
Shiva	14
	15

No object selected

QUERY 5 = FIND THE NAMES OF FACULTY MEMBERS FOR WHOM THE COMBINED ENROLMENT OF THE COURSES THAT THEY TEACH IS LESS THAN FIVE.

MySQL Workbench - Local instance MySQL80

```

86 GROUP BY fid HAVING COUNT(*)=(SELECT COUNT(DISTINCT room) FROM class) );
87 -- Query 5 = Find the names of faculty members for whom the combined enrolment of the courses that they teach is less Than five.
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 = Find the names of students who are not enrolled in any class.

```

Result Grid

fname
Harish
MV
Mira
Shiva

No object selected

QUERY 6 = FIND THE NAMES OF STUDENTS WHO ARE NOT ENROLLED IN ANY CLASS.

MySQL Workbench - Local instance MySQL80

```

91 FROM Class C, Enrolled E
92 WHERE C.cname = E cname
93 AND C.fid = F.fid;
94 -- Query 6 = Find the names of students who are not enrolled in any class.
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 = i. For each age value that appears in Students, find the level value that appears most often. For example, if the

```

Result Grid

sname
Rita

No object selected

QUERY 7 = i. 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)

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local instance MySQL80' is selected. The 'File', 'Edit', 'View', 'Query', 'Database', 'Server', 'Tools', and 'Help' menus are available. Below the menu is a toolbar with icons for file operations like Open, Save, Print, and Database management. The 'Navigator' pane on the left lists various database schemas such as 'banking', 'banking_enterprise', 'banking_test', 'banking_test1', 'banktest1_134', 'bms_db', 'book_dealer', 'book_dealer_database', 'bookdealer', 'books_adopted_db', 'insurance', 'movie', 'student_enrollment', and 'sys'. The 'Information' pane at the bottom indicates 'No object selected'. The main area contains a 'SQL File 3*' tab with the following SQL code:

```

92 WHERE C cname = E cname;
93 AND C fid = F fid;
94 -- Query 6 = Find the names of students who are not enrolled in any class.
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 = 1. For each age value that appears in Students, find the level value that appears most often. For example, if the
100 • SELECT S age, S lvl

```

The 'Result Grid' pane shows the output of the last query:

age	lvl
19	Sr
20	Jr
21	Sr

A status message in the top right corner says: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

PROGRAM-10:COLLEGE DATABASE

QUESTION:

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, FinallA)

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

v. Categorize students based on the following criterion:

If FinallA = 17 to 20 then CAT = 'Outstanding'

If FinallA = 12 to 16 then CAT = 'Average'

If FinallA < 12 then CAT = 'Weak'

Give these details only for 8th semester A, B, and C section students.

PROGRAM CODE:

```

create database college;
use college;
CREATE TABLE STUDENT (
USN VARCHAR (10) PRIMARY KEY,
SNAME VARCHAR (25),
ADDRESS VARCHAR (25),
PHONE real,
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',8877881122,'M');
INSERT INTO STUDENT
VALUES('1RN13CS062','SANDHYA','BENGALURU',7722829912,'F');
INSERT INTO STUDENT
VALUES('1RN13CS091','TEESHA','BENGALURU',7712312312,'F');
INSERT INTO STUDENT
VALUES('1RN13CS066','SUPRIYA','MANGALURU',8877881122,'F');
INSERT INTO STUDENT
VALUES('1RN14CS010','ABHAY','BENGALURU',9900211201,'M');
INSERT INTO STUDENT
VALUES('1RN14CS032','BHASKAR','BENGALURU',9923211099,'M');
INSERT INTO STUDENT VALUES ('1RN14CS025','ASMI','BENGALURU', 7894737377,'F');
INSERT INTO STUDENT VALUES ('1RN15CS011','AJAY','TUMKUR', 9845091341,'M');
INSERT INTO STUDENT VALUES
('1RN15CS029','CHITRA','DAVANGERE',7696772121,'F');
INSERT INTO STUDENT VALUES ('1RN15CS045','JEEVA','BELLARY', 9944850121,'M');
INSERT INTO STUDENT VALUES
('1RN15CS091','SANTOSH','MANGALURU',8812332201,'M');
INSERT INTO STUDENT VALUES('1RN16CS045','ISMAIL','KALBURGI',9900232201,'M');
INSERT INTO STUDENT VALUES
('1RN16CS088','SAMEERA','SHIMOGA',9905542212,'F');
INSERT INTO STUDENT VALUES
('1RN16CS122','VINAYAKA','CHIKAMAGALUR',8800880011,'M');

INSERT INTO SEMSEC VALUES ('CSE8A', 8,'A');
INSERT INTO SEMSEC VALUES ('CSE8B', 8,'B');
INSERT INTO SEMSEC VALUES ('CSE8C',8,'C');
INSERT INTO SEMSEC VALUES ('CSE7A',7,'A');

```

```
INSERT INTO SEMSEC VALUES ('CSE7B',7,'B');
INSERT INTO SEMSEC VALUES ('CSE7C',7,'C');
INSERT INTO SEMSEC VALUES ('CSE6A',6,'A');
INSERT INTO SEMSEC VALUES ('CSE6B', 6,'B');
INSERT INTO SEMSEC VALUES ('CSE6C', 6,'C');
INSERT INTO SEMSEC VALUES ('CSE5A', 5,'A');
INSERT INTO SEMSEC VALUES ('CSE5B', 5,'B');
INSERT INTO SEMSEC VALUES ('CSE5C', 5,'C');
INSERT INTO SEMSEC VALUES ('CSE4A',4,'A');
INSERT INTO SEMSEC VALUES ('CSE4B', 4,'B');
INSERT INTO SEMSEC VALUES ('CSE4C', 4,'C');
INSERT INTO SEMSEC VALUES ('CSE3A', 3,'A');
INSERT INTO SEMSEC VALUES ('CSE3B', 3,'B');
INSERT INTO SEMSEC VALUES ('CSE3C', 3,'C');
INSERT INTO SEMSEC VALUES ('CSE2A', 2,'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;
-- 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
SUBSEM = 8;

OUTPUT SCREENSHOTS:

LAB6_MOVIES SQL File 3* SQL File 4*

148 • INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3)VALUES
 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
 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'; |

162

163 • 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

100% 13:161 | 20 errors found

Result Grid Filter Rows: Search Export:

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



155

156 • 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';

162

163 • 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;

100% 1:169 | 20 errors found

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



LAB6_MOVIES SQL File 3* SQL File 4* Limit to 1000 rows

```
183 DECLARE C_A INTEGER;
184 DECLARE C_B INTEGER;
185 DECLARE C_C INTEGER;
186 DECLARE C_SUM INTEGER;
187 DECLARE C_AVG INTEGER;
188 DECLARE C_USN VARCHAR(10);
189 DECLARE C_SUBCODE VARCHAR(8);
190 DECLARE C_SSID VARCHAR(5);
191 DECLARE C_IAMARKS CURSOR FOR
192 SELECT GREATEST(TEST1,TEST2) AS A, GREATEST(TEST1,TEST3) AS B, GREATEST(TEST3,TEST2) AS C, USN, SUBCODE, SSID
193 FROM IAMARKS
194 WHERE FINALIA IS NULL
195 FOR UPDATE;
196 OPEN C_IAMARKS;
```

100% 1:211

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

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

Form Editor Field Types

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