



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

**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING**

**DATABASE MANAGEMENT SYSTEM
LABORATORY RECORD**

NAME: Prithvi.J
USN: 1BM19CS122

PROGRAM: BACHELOR OF ENGINEERING

SEMESTER: IV

SESSION: APR-JUL 2021

COURSE CODE: 19CS4PCDBM

COURSE TITLE: DATABASE MANAGEMENT SYSTEM

CREDITS: 4

DBMS Lab List

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

PROGRAM 1: INSURANCE DATABASE

Consider the Insurance database given below. The primary keys are underlined and the data types are specified.

PERSON (driver-id #: String, name: String, address: String)

CAR (Regno: String, model: String, year: int)

ACCIDENT (report-number: int, date: date, location: String)

OWNS (driver-id #: String, Regno: String)

PARTICIPATED (driver-id: String, Regno: String, report-number: int, damage-amount: int)

Queries Executed:

- Create the above tables by properly specifying the primary keys and the foreign keys.
- Enter at least five tuples for each relation.

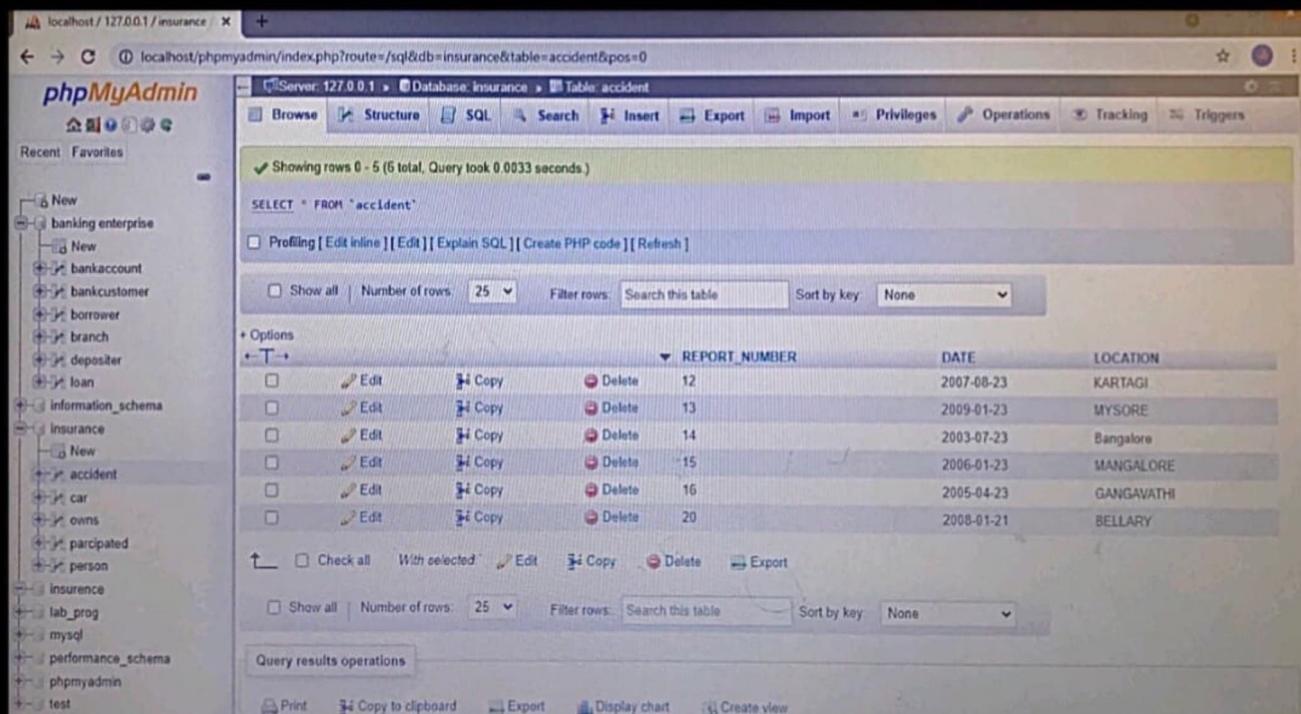
The screenshot shows the phpMyAdmin interface for the 'insurance' database. The left sidebar lists the database structure with tables: accident, car, owns, participated, and person. The main area displays the 'Structure' tab for the 'person' table. The table has the following columns:

Field	Type	Collation	Size	Overhead
driver-id	String	utf8mb4_general_ci	16,0 K18	-
name	String	utf8mb4_general_ci	16,0 K18	-
address	String	utf8mb4_general_ci	16,0 K18	-

Below the table structure, there are buttons for Browse, Structure, Search, Insert, Empty, Drop, and Operations. The 'Rows' section shows 5 rows in the table.

iii. Demonstrate how you

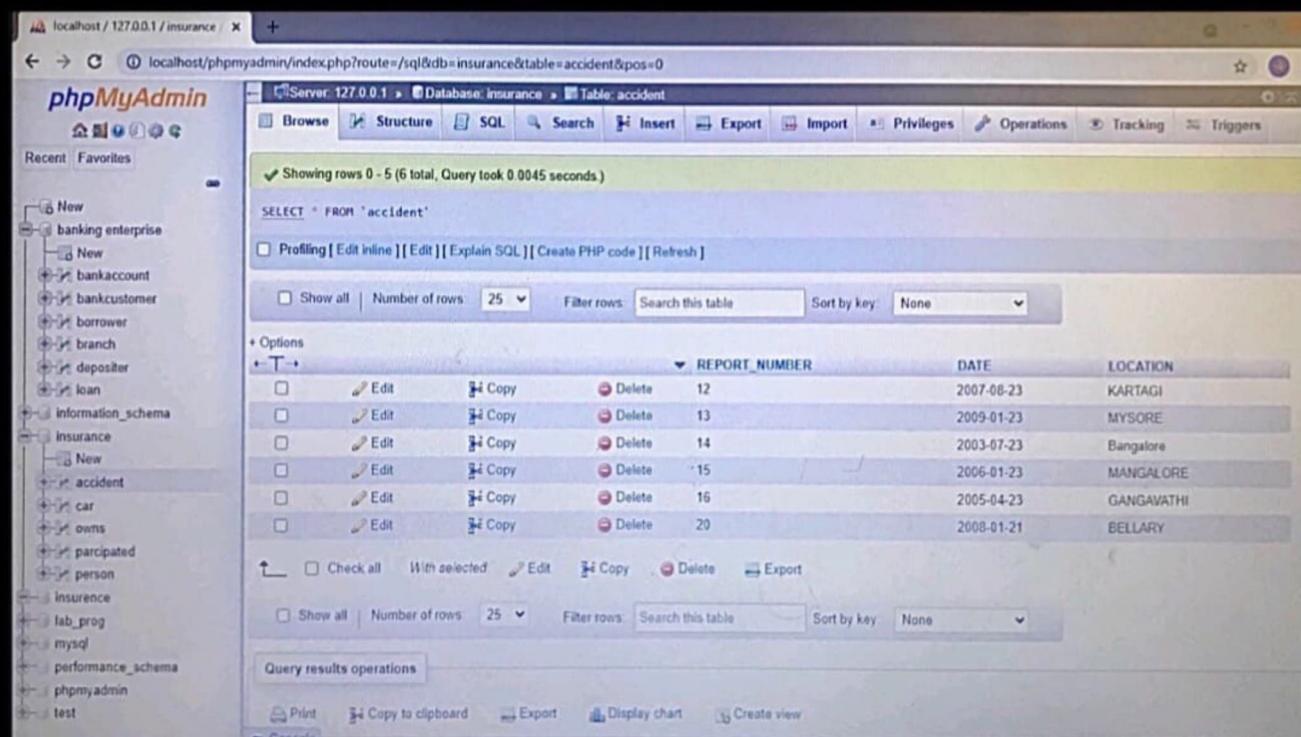
- a. Update the damage amount for the car with a specific Regno in the accident with report number 12 to 25000



The screenshot shows the phpMyAdmin interface for the 'insurance' database. The 'accident' table is selected. The table has columns: REPORT_NUMBER, DATE, and LOCATION. The data is as follows:

REPORT_NUMBER	DATE	LOCATION
12	2007-08-23	KARTAGI
13	2009-01-23	mysore
14	2003-07-23	Bangalore
15	2006-01-23	MANGALORE
16	2005-04-23	GANGAVATHI
20	2008-01-21	BELLARY

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



The screenshot shows the phpMyAdmin interface for the 'insurance' database. The 'accident' table is selected. The table has columns: REPORT_NUMBER, DATE, and LOCATION. The data is as follows:

REPORT_NUMBER	DATE	LOCATION
12	2007-08-23	KARTAGI
13	2009-01-23	mysore
14	2003-07-23	Bangalore
15	2006-01-23	MANGALORE
16	2005-04-23	GANGAVATHI
20	2008-01-21	BELLARY

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

The screenshot shows the phpMyAdmin interface for the 'insurance' database. The left sidebar lists various tables under the 'Insurance' schema, including 'New', 'accident', 'car', 'owns', and 'participated'. The 'participated' table is currently selected. The main area displays the contents of the 'participated' table with the following data:

	DRIVER_ID	REG_NO	REPORT_NUMBER	DAMAGE_AMOUNT
<input type="checkbox"/>	A1	KA37 0011	14	2500
<input type="checkbox"/>	A2	KA37 0012	16	7000
<input type="checkbox"/>	A3	KA37 0013	12	5000
<input type="checkbox"/>	A4	KA37 0014	15	4000
<input type="checkbox"/>	A5	KA37 0015	13	8000

Below the table, there are buttons for 'Check all', 'With selected:', 'Edit', 'Copy', 'Delete', and 'Export'. The top navigation bar includes links for 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', 'Import', 'Privileges', 'Operations', 'Tracking', and 'Triggers'.

PROGRAM 2. BANKING ENTERPRISE DATABASE

Consider the following database for a banking enterprise.

BRANCH (branch-name: String, branch-city: String, assets: real)

ACCOUNTS (accno: int, branch-name: String, balance: real)

DEPOSITOR (customer-name: String, customer-street: String, customer-city: String)

LOAN (loan-number: int, branch-name: String, amount: real)

BORROWER (customer-name: String, loan-number: int)

Queries:

- i. Create the above tables by properly specifying the primary and the foreign keys.
- ii. Enter at least five tuples for each relation.

The screenshot shows the phpMyAdmin interface for the 'banking enterprise' database. The left sidebar lists the database structure with tables: bankaccount, bankcustomer, borrower, branch, depositor, and loan. The main area displays the 'Structure' tab for the 'branch' table, showing columns: branch-name (String), branch-city (String), and assets (real). The table has 5 rows. Below the table, there are buttons for Browse, Structure, Search, Insert, Empty, Drop, and a summary row indicating 5 InnoDB tables with a total size of 144.0 KIB.

Table	Action	Rows	Type	Collation	Size	Overhead
bankaccount	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	32.0 KIB	-
bankcustomer	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	16.0 KIB	-
borrower	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	16.0 KIB	-
branch	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	16.0 KIB	-
depositor	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	32.0 KIB	-
loan	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	32.0 KIB	-
6 tables	Sum	30	InnoDB	utf8mb4_general_ci	144.0 KIB	0 B

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

The screenshot shows the phpMyAdmin interface for the 'banking enterprise' database. The left sidebar shows the schema structure with 'bankcustomer' selected. The main area displays the 'bankcustomer' table with 5 rows. The columns are 'customername', 'customerstreet', and 'customercity'. The data is as follows:

customername	customerstreet	customercity
ashwini	bannerghatta_road	bangalore
deepak	cg_road	delhi
nisha	church_street	delhi
prithvi	mg_road	bangalore
suraj	national_road	bangalore

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

The screenshot shows the phpMyAdmin interface for the 'banking enterprise' database. The left sidebar shows the schema structure with 'branch' selected. The main area displays the 'branch' table with 5 rows. The columns are 'branchname', 'branchcity', and 'assests'. The data is as follows:

branchname	branchcity	assests
UBI_Chamarajpet	Bangalore	40000
UBI_MgRoad	Bangalore	25000
UBI_ParliamentRoad	delhi	30000
UBI_ResidencyRoad	Bangalore	10000
UBI_ShivajiRoad	Bombay	20000

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

The screenshot shows the phpMyAdmin interface for the 'banking enterprise' database. The left sidebar lists databases, schemas, and tables. The main area displays the 'Tables' section for the 'banking enterprise' schema. A table lists seven tables: bankaccount, bankcustomer, borrower, branch, branchtotalloan, depositer, and loan. The table includes columns for Action, Rows, Type, Collation, Size, and Overhead. Below the table, there are buttons for Check all and With selected, and links for Print, Data dictionary, Create table, and Consola.

Table	Action	Rows	Type	Collation	Size	Overhead
bankaccount	<input type="button" value="Browse"/> <input type="button" value="Structure"/> <input type="button" value="Search"/> <input type="button" value="Insert"/> <input type="button" value="Empty"/> <input type="button" value="Drop"/>	5	InnoDB	utf8mb4_general_ci	32.0 Kib	-
bankcustomer	<input type="button" value="Browse"/> <input type="button" value="Structure"/> <input type="button" value="Search"/> <input type="button" value="Insert"/> <input type="button" value="Empty"/> <input type="button" value="Drop"/>	5	InnoDB	utf8mb4_general_ci	16.0 Kib	-
borrower	<input type="button" value="Browse"/> <input type="button" value="Structure"/> <input type="button" value="Search"/> <input type="button" value="Insert"/> <input type="button" value="Empty"/> <input type="button" value="Drop"/>	5	InnoDB	utf8mb4_general_ci	16.0 Kib	-
branch	<input type="button" value="Browse"/> <input type="button" value="Structure"/> <input type="button" value="Search"/> <input type="button" value="Insert"/> <input type="button" value="Empty"/> <input type="button" value="Drop"/>	5	InnoDB	utf8mb4_general_ci	16.0 Kib	-
branchtotalloan	<input type="button" value="Browse"/> <input type="button" value="Structure"/> <input type="button" value="Search"/> <input type="button" value="Insert"/> <input type="button" value="Edit"/> <input type="button" value="Drop"/>	~0	<input type="button" value="View"/>	---	---	-
depositor	<input type="button" value="Browse"/> <input type="button" value="Structure"/> <input type="button" value="Search"/> <input type="button" value="Insert"/> <input type="button" value="Empty"/> <input type="button" value="Drop"/>	5	InnoDB	utf8mb4_general_ci	32.0 Kib	-
loan	<input type="button" value="Browse"/> <input type="button" value="Structure"/> <input type="button" value="Search"/> <input type="button" value="Insert"/> <input type="button" value="Empty"/> <input type="button" value="Drop"/>	5	InnoDB	utf8mb4_general_ci	32.0 Kib	0 B
Sum		~30	InnoDB	utf8mb4_general_ci	144.0 Kib	

PROGRAM 3. SUPPLIER DATABASE

Consider the following schema:

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

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

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

The Catalog relation lists the prices charged for parts by Suppliers.

Write the following queries in SQL:

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

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1
- Database:** supplier database
- Table:** CATALOG
- Query Result:**

```
SELECT pname FROM PARTS,CATALOG WHERE CATALOG.pid=PARTS.pid AND CATALOG.sid IS NOT NULL
```

Showing rows 0 - 4 (5 total, Query took 0.1164 seconds.)

pname
screw
bolt
screw
nail
bolt
- Operations:** Print, Copy to clipboard, Export, Display chart, Create view, Console

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

The screenshot shows the phpMyAdmin interface for the 'supplier database'. The left sidebar lists databases like 'New', 'banking enterprise', 'Information_schema', 'Insurance', 'Insurance', 'lab_prog', 'mysql', 'performance_schema', 'phpmyadmin', and 'supplier database'. The 'supplier database' is selected, and its tables 'catalog', 'parts', and 'suppliers' are visible. The 'suppliers' table is currently selected, displaying 5 rows of data:

	sid	sname	address
<input type="checkbox"/>	122	alien	mysore
<input type="checkbox"/>	132	ikea	bangalore
<input type="checkbox"/>	142	Acme Widget	dehl
<input type="checkbox"/>	152	alexa	mangalore
<input type="checkbox"/>	162	grid	kerala

Below the table, there are buttons for 'Check all', 'With selected:', 'Edit', 'Copy', 'Delete', and 'Export'. The top navigation bar includes links for 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', 'Import', 'Privileges', 'Operations', 'Tracking', and 'Triggers'.

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

The screenshot shows the phpMyAdmin interface for the 'supplier database'. The left sidebar lists the same databases as the previous screenshot. The 'supplier database' is selected, and its tables 'catalog', 'parts', and 'suppliers' are visible. A complex SQL query is entered in the query editor:

```
SELECT S.sname FROM SUPPLIERS S WHERE NOT EXISTS (( SELECT P.pid FROM PARTS P WHERE P.color = 'Red') EXCEPT ( SELECT C.pid FROM CATALOG C, PARTS P WHERE C.sid = S.sid AND C.pid = P.pid AND P.color = 'Red'))
```

The results show 0 rows found, indicating no suppliers supply every red part.

Below the query, there are buttons for 'Check all', 'With selected:', 'Edit', 'Copy', 'Delete', and 'Export'. The top navigation bar includes links for 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', 'Import', 'Privileges', 'Operations', 'Tracking', and 'Triggers'. At the bottom, there is a 'Bookmark this SQL query' button and a 'Label' input field.

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

The screenshot shows the phpMyAdmin interface for the 'parts' table in the 'supplier database'. The table has columns: pid, pname, and color. The data is as follows:

pid	pname	color
1	screw	red
2	bolt	yellow
3	screw	black
4	nail	white
5	bolt	blue

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).

The screenshot shows the phpMyAdmin interface for the 'catalog' table in the 'supplier database'. The table has columns: sid, pid, and cost. The data is as follows:

sid	pid	cost
122	1	800.00
132	2	3000.00
142	3	4500.00
152	4	2000.00
162	5	5000.00

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

The screenshot shows the phpMyAdmin interface for the 'supplier' database. The 'suppliers' table is selected. The table data is as follows:

		sid	sname	address
<input type="checkbox"/>	<input type="checkbox"/> Edit	122	allen	mysore
<input type="checkbox"/>	<input type="checkbox"/> Edit	132	ikea	bangalore
<input type="checkbox"/>	<input type="checkbox"/> Edit	142	Acme Widget	delhi
<input type="checkbox"/>	<input type="checkbox"/> Edit	152	alexa	mangalore
<input type="checkbox"/>	<input type="checkbox"/> Edit	162	grid	kerala

Below the table, there are buttons for 'Check all', 'With selected', 'Edit', 'Copy', 'Delete', and 'Export'. There are also buttons for 'Print', 'Copy to clipboard', 'Export', 'Display chart', and 'Create view'.

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

The screenshot shows the phpMyAdmin interface for the 'supplier' database. A complex SQL query is run against the CATALOG table to find suppliers who supply only red parts. The results show zero rows.

Query results:

```
MySQL returned an empty result set (i.e. zero rows) (Query took 0.0097 seconds.)
```

```
SELECT S.sname FROM SUPPLIERS S WHERE NOT EXISTS((SELECT P.pid FROM PARTS P ) EXCEPT (SELECT C.pid FROM CATALOG C WHERE C.sid = S.sid))
```

Below the results, there are buttons for 'Create view', 'Bookmark this SQL query', and 'Label' (with a 'Let every user access this bookmark' checkbox). The bottom section shows the query again with the results:

Showing rows 0 - 0 (1 total, Query took 0.0205 seconds.)

```
SELECT S.sname FROM SUPPLIERS S WHERE NOT EXISTS (( SELECT P.pid FROM PARTS P WHERE P.color = 'Red') EXCEPT ( SELECT C.pid FROM CATALOG C, PARTS P WHERE C.sid = S.sid AND C.pid = P.pid AND P.color = 'Red' ))
```

PROGRAM 4. STUDENT FACULTY DATABASE

Consider the following database for student enrolment for course:

STUDENT (snum: integer, sname: string, major: string, level: string, age: integer)

CLASS (name: string, meets at: time, room: string, fid: integer)

ENROLLED (snum: integer, cname: string)

FACULTY (fid: integer, fname: string, deptid: integer)

The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair such that the student is enrolled in the class. Level is a two character code with 4 different values (example:

Junior: JR etc)

Write the following queries in SQL. No duplicates should be printed in any of the answers.

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

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

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1 | **Database:** student faculty database | **Table:** faculty
- Query Results:**

```
Showing rows 0 - 0 (1 total, Query took 0.0126 seconds)

SELECT DISTINCT s.sname FROM student s,class c,faculty f,enrolled e WHERE s.snum=e.snum AND e.cname=c.cname AND s.level='JR' AND f.fname='Marshitn' AND f.fid=c.fid
```
- Table Structure:** A table named "faculty" is shown with columns: sname (tom).
- Operations:** Buttons for Edit, Copy, Delete, Show all, Number of rows (set to 25), Filter rows, and Search this table.
- Query results operations:** Buttons for Print, Copy to clipboard, Export, Display chart, and Create view.
- Bookmark this SQL query:** A link to bookmark the current query.
- Label:** A placeholder for a label.

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

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1 | **Database:** student faculty database | **Table:** faculty
- Query Results:**

```
Showing rows 0 - 1 (2 total, Query took 0.0622 seconds)

SELECT DISTINCT cname FROM class WHERE room='room128' OR cname IN (SELECT e cname FROM enrolled e GROUP BY e cname HAVING COUNT(e)>=5)
```
- Table Structure:** A table named "faculty" is shown with columns: cname (class10, class5).
- Operations:** Buttons for Edit, Copy, Delete, Show all, Number of rows (set to 25), Filter rows, and Search this table.
- Query results operations:** Buttons for Print, Copy to clipboard, Export, Display chart, and Create view.
- Bookmark this SQL query:** A link to bookmark the current query.
- Label:** A placeholder for a label.

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

The screenshot shows the phpMyAdmin interface for a database named "student faculty database". The left sidebar lists various databases and their tables. The main area shows an SQL query:

```
MySQL returned an empty result set (i.e. zero rows) (Query took 0.0381 seconds)

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.meets_at=c2.meets_at )
```

Below the query, there are options to "Bookmark this SQL query" and "Label" it. The results section shows "Showing rows 0 - 0 (1 total; Query took 0.0169 seconds)".

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

The screenshot shows the phpMyAdmin interface for the same database. The left sidebar lists tables. The main area shows an SQL query:

```
Showing rows 0 - 0 (1 total, Query took 0.0169 seconds)

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) )
```

The results section shows a table with one row for Shobha, with fields fname, fid, and room. There are buttons for Edit, Copy, Delete, and Export.

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

The screenshot shows the phpMyAdmin interface for a database named 'student faculty database'. The left sidebar lists various databases and their tables. The main area displays the results of a SQL query run against the 'faculty' table. The query is:

```
SELECT DISTINCT F.fname FROM Faculty F WHERE F.fid IN ( SELECT c.fid FROM class c, enrolled e WHERE c.cname = e cname GROUP BY c.cname HAVING COUNT(c.cname) < 5 )
```

The results show two rows of data:

fname
Harshith
Shobha

Below the results, there are options for printing, copying to clipboard, exporting, displaying a chart, and creating a view.

PROGRAM 5. AIRLINE FLIGHT DATABASE

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

FLIGHTS (flno: integer, from: string, to: string, distance: integer, departs: time, arrives: time, price: integer) AIRCRAFT (aid: integer, fname: 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.

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

The screenshot shows the phpMyAdmin interface for a MySQL database named 'flights'. The current table is 'aircraft'. The table has three columns: 'aid' (primary key), 'fname', and 'cruisingrange'. There are three rows: 'Aircraft' (aid=1, fname='Airbus320', cruisingrange=5000), 'Airbus320' (aid=2, fname='Boeing747', cruisingrange=7000), and 'Lufthansa' (aid=3, fname='Airbus380', cruisingrange=8000). The 'Structure' tab is visible at the top, and the 'Query results operations' bar is at the bottom.

aid	fname	cruisingrange
1	Aircraft	5000
2	Airbus320	7000
3	Lufthansa	8000

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.

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1
- Database:** Rights
- Table:** certified
- Query:** SELECT c.eid, MAX(cruisingrange) FROM certified c,aircraft a WHERE c.eid=a.eid GROUP BY c.eid HAVING COUNT(*)>3
- Results:** One row is shown: eid 1 with MAX(cruisingrange) 7000.

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

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1
- Database:** Rights
- Table:** employees
- Query:** SELECT DISTINCT e.ename FROM employees e WHERE e.salary < (SELECT MIN(f.price) FROM flight f WHERE f.frm='Bangalore' AND f.too='Frankfurt')
- Results:** Five rows are shown:

ename
Lucus
Chen
Felix
Harry
Cole

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

localhost/phpmyadmin/index.php?route=/table/sql&db=flights&table=employees

Server: 127.0.0.1 - Database: Rights - Table: employees

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Show query box

Showing rows 0 - 1 (2 total, Query took 0.0118 seconds.)

SELECT distinct e.name FROM employees e,aircraft a,certified c WHERE e.eid=c.eid AND c.aid=a.aid AND a.name='Boeing'

Profiling | Edit inline | Edit | Explain SQL | Create PHP code | Refresh |

Show all Number of rows: 25 Filter rows: Search this table

ename
Lucas
Chen

Options

Check all With selected Edit Copy Delete Export

Show all Number of rows: 25 Filter rows: Search this table

Query results operations

Print Copy to clipboard Export Display chart Create view

Bookmark this SQL query

Label Let every user access this bookmark

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

localhost/phpmyadmin/index.php?route=/table/sql&db=flights&table=aircraft

Server: 127.0.0.1 - Database: Rights - Table: aircraft

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Show query box

Showing rows 0 - 2 (3 total, Query took 0.0098 seconds.)

SELECT a.aid FROM aircraft a WHERE a.cruisingrange <= (SELECT MIN(f.distance) FROM flight f WHERE f.frm='Bangalore' AND f.too='Delhi')

Profiling | Edit inline | Edit | Explain SQL | Create PHP code | Refresh |

Show all Number of rows: 25 Filter rows: Search this table Sort by key None

aid
201
450
879

Options

Check all With selected Edit Copy Delete Export

Show all Number of rows: 25 Filter rows: Search this table Sort by key None

Query results operations

Print Copy to clipboard Export Display chart Create view

Bookmark this SQL query

Label Let every user access this bookmark

PROGRAM 6. ORDER PROCESSING DATABASE

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

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

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

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

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

WAREHOUSE (warehouse #: int, city: String)

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

- i. Create the above tables by properly specifying the primary keys and the foreign keys and the foreign keys.
- ii. Enter at least five tuples for each relation.

The screenshot shows the phpMyAdmin interface for a database named 'order'. The 'Structure' tab is selected. It displays three tables: 'customer', 'orders', and 'salesman'. The 'customer' table has 5 rows, 'orders' has 5 rows, and 'salesman' has 5 rows. All three tables are InnoDB type with utf8mb4_general_ci collation. The 'customer' table has a size of 32.0 KiB, 'orders' has 48.0 KiB, and 'salesman' has 16.0 KiB. The 'Overhead' column is listed as 0 B for all three tables. Below the table list, there are buttons for 'Create table' and 'Go'.

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

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

- Server:** 127.0.0.1
- Database:** order
- Table:** Orders
- Query:** select distinct c.salesman_id, s.salesman_name from Customer c, Salesman s where c.salesman_id = s.salesman_id and 1 < (select count(customer_id) from Customer where salesman_id = c.salesman_id)
- Results:** A table showing salesman_id and salesman_name. It contains two rows: 1000 (John) and 2000 (Ravi).
- Operations:** Buttons for Print, Copy to clipboard, Export, Display chart, Create view, and Bookmark this SQL query.

iii. List all salesmen and indicate those who have and don't have customers in their cities

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

- Server:** 127.0.0.1
- Database:** order
- Table:** Orders
- Query:** select s.salesman_name, c.customer_name from Salesman s, Customer c where s.salesman_id = c.salesman_id and c.city = s.city union select s.salesman_name, 'No Match' from Salesman s, Customer c where s.salesman_id = c.salesman_id and c.city != s.city
- Results:** A table showing salesman_name and customer_name. It lists salesmen with their corresponding customers and salesmen with 'No Match'.
- Operations:** Buttons for Print, Copy to clipboard, Export, Display chart, Create view, and Bookmark this SQL query.

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

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

- Top Bar:** Shows the connection path: Server: 127.0.0.1 > Database: order > Table: Orders.
- Toolbar:** Includes Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, Operations, Tracking, and Triggers.
- Query Editor (Top Panel):** Displays the creation of a view named salesman_view based on the Orders and Customer tables.
- Query Editor (Second Panel):** Shows a delete operation from the Salesman table where salesman_id = 1000.
- Query Editor (Third Panel):** Displays a select query from the Salesman table.
- Table View:** Shows the data for the Salesman table with columns: salesman_id, salesman_name, city, commission. The data is as follows:

salesman_id	salesman_name	city	commission
2000	Ravi	Bangalore	20
3000	Kumar	Mysore	15
4000	Smith	Delhi	30
5000	Harsha	Hyderabad	15

PROGRAM 7. BOOK DEALER DATABASE

The following tables are maintained by a book dealer:

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

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

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

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

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

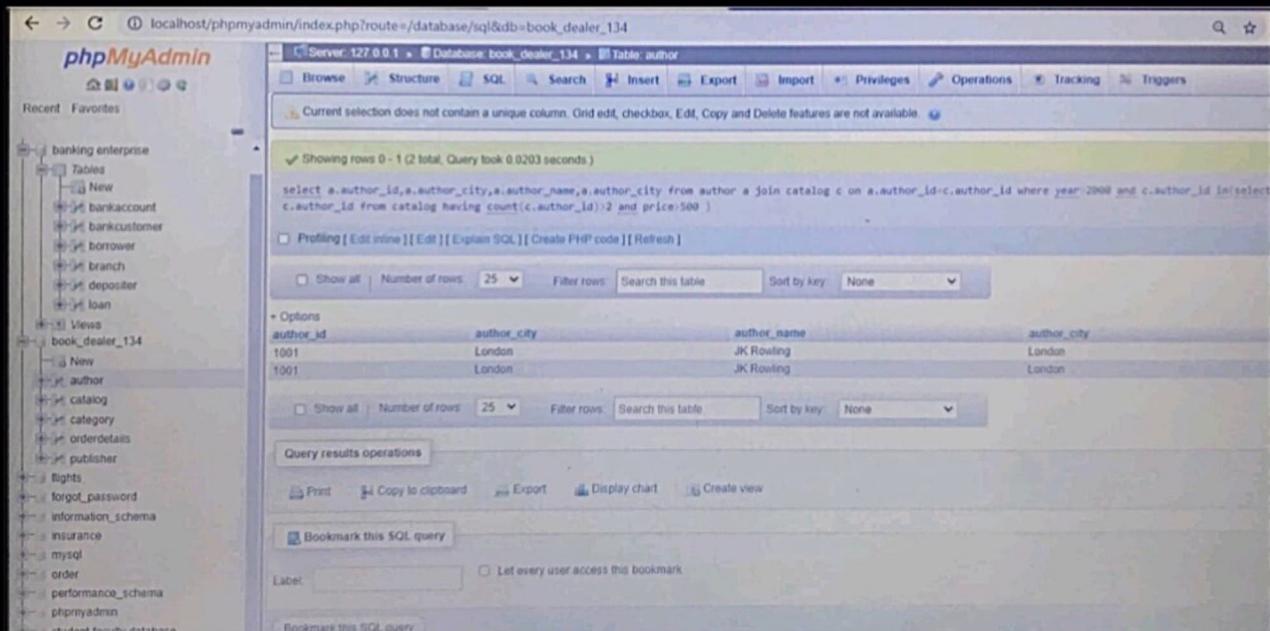
i. Create the above tables by properly specifying the primary keys and the foreign keys.

ii. Enter at least five tuples for each relation.

The screenshot shows the phpMyAdmin interface for the 'book_dealer_134' database. The left sidebar lists databases and tables. The main area displays the table structure for 'author', 'catalog', 'category', 'orderdetails', and 'publisher'. A summary at the bottom indicates 5 tables and 28 rows.

Table	Action	Rows	Type	Collation	Size	Overhead
author	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 Kib	-
catalog	Browse Structure Search Insert Empty Drop	7	InnoDB	utf8mb4_general_ci	54.0 Kib	-
category	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	16.0 Kib	-
orderdetails	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_general_ci	32.0 Kib	-
publisher	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 Kib	-

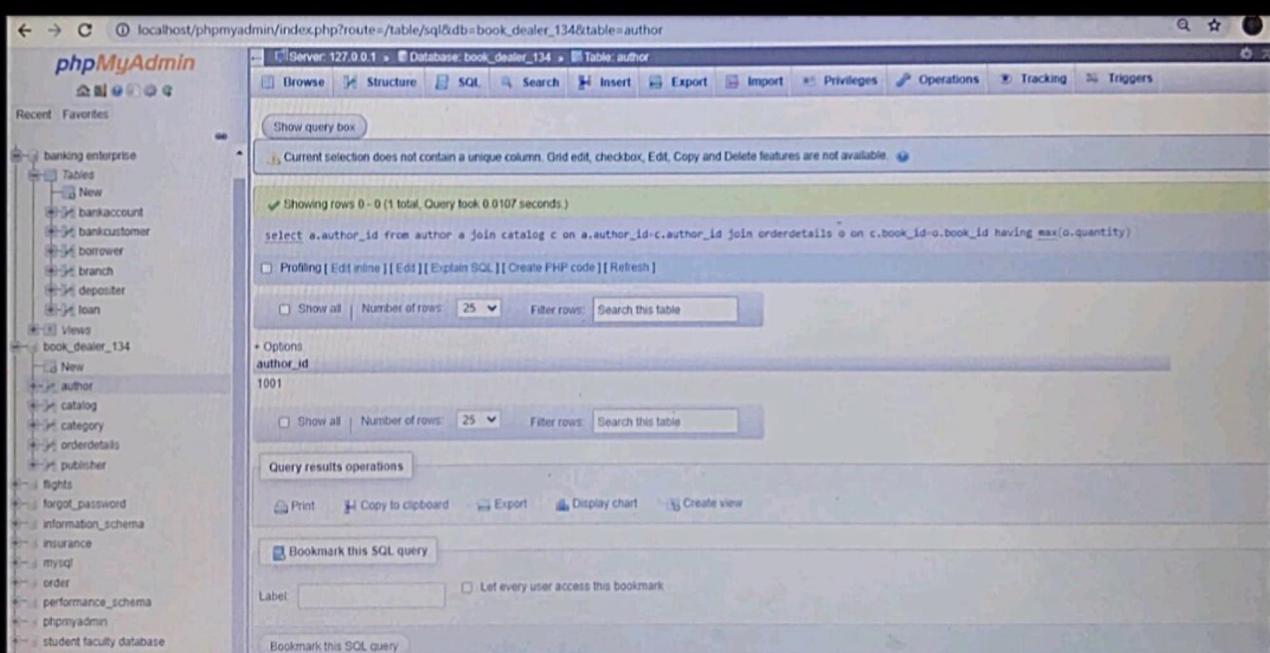
iii. Give the details of the authors who have 2 or more books in the catalog and the price of the books in the catalog and the year of publication is after 2000.



```
select a.author_id,a.author_name,a.author_city from author a join catalog c on a.author_id=c.author_id where year>2000 and c.author_id in(select c.author_id from catalog having count(c.author_id)>2 and price<500 )
```

author_id	author_name	author_city
1001	JK Rowling	London
1001	JK Rowling	London

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



```
select a.author_id from author a join catalog c on a.author_id=c.author_id join orderdetails o on c.book_id=o.book_id having max(o.quantity)
```

author_id
1001

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

The screenshot shows the phpMyAdmin interface for a MySQL database named 'book_dealer_134'. The left sidebar lists various databases and tables. The 'catalog' table is selected in the main area. A SQL query is entered in the top text box:

```
update catalog set price=(0.1*price)+price where publisher_id=2001
```

The results of the query are displayed below, showing 7 rows affected:

book_id	book_title	author_id	publisher_id	category_id	year	price
4001	HP and Goblet Of Fire	1001	101	3001	2002	600
4002	HP and Order Of Phoenix	1001	102	3001	2005	650
4003	Two States	1002	104	3001	2009	65
4004	3 Mistakes of my life	1002	104	3001	2007	55
4005	Da Vinci Code	1004	103	3001	2004	450
4006	Angels and Demons	1004	103	3001	2003	350
4007	Artificial Intelligence	1003	102	3002	1970	500

PROGRAM 8. STUDENT ENROLLMENT DATABASE

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

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

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

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

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

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

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

ii. Enter at least five tuples for each relation.

The screenshot shows the phpMyAdmin interface for the 'stud_enrollment_134' database. The left sidebar lists databases and the current database's structure. The main area displays the table structure with columns: Table, Action, Rows, Type, Collation, Size, and Overhead. The tables listed are book_adoption, course, enroll, student, and text. The 'book_adoption' table has 8 rows, 'course' has 5, 'enroll' has 4, 'student' has 4, and 'text' has 8. All tables use InnoDB storage engine and utf8mb4_general_ci collation. The 'text' table has a larger size of 112.0 Kib.

Table	Action	Rows	Type	Collation	Size	Overhead
book_adoption	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_general_ci	32.0 Kib	-
course	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	16.0 Kib	-
enroll	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	32.0 Kib	-
student	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 Kib	-
text	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_general_ci	16.0 Kib	-
5 tables	Sum	29	InnoDB	utf8mb4_general_ci	112.0 Kib	0 B

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

The screenshot shows the phpMyAdmin interface for the 'stud_enrollment_134' database. The left sidebar lists various databases and the current one selected. The main area shows the 'Structure' tab for the 'stud_enrollment_134' database. It displays a table list with the following details:

Table	Action	Rows	Type	Collation	Size	Overhead
book_adoption	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_general_ci	32.0 Kib	-
course	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	16.0 Kib	-
enroll	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	32.0 Kib	-
student	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 Kib	-
text	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_general_ci	16.0 Kib	-
5 tables	Sum	29	InnoDB	utf8mb4_general_ci	112.0 Kib	0 B

At the bottom, there is a 'Create Table' form with 'Name:' set to 'text' and 'Number of columns:' set to '4'. A 'Go' button is present below it.

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.

The screenshot shows the phpMyAdmin interface for the 'stud_enrollment_134' database, specifically the 'BOOK_ADOPTION' table. The top status bar indicates '1 row inserted.' The main area shows the table structure and the inserted data:

book_isbn	book_title	publisher	author
10	DATABASE SYSTEMS	PEARSON	SCHIELD
23	ADA	PEARSON	SCHIELD
900	OPERATING SYS	PEARSON	LELAND
901	CIRCUITS	HALL INDIA	BOB
902	SYSTEM SOFTWARE	PETERSON	JACOB
903	SCHEDULING	PEARSON	PATIL
904	DATABASE SYSTEMS	PEARSON	JACOB
905	DATABASE MANAGER	PEARSON	BOB
906	SIGNALS	HALL INDIA	SUMIT

v. List any department that has all its adopted

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

The screenshot shows the phpMyAdmin interface for a MySQL database named 'stud_enrollment_134'. The current table is 'BOOK_ADOPTION'. The query results show four rows of data:

courseno	book_isbn	book_title
111	23	ADA
111	904	DATABASE SYSTEMS
111	900	OPERATING SYS
111	903	SCHEDULING

The SQL query executed is:

```
SELECT c.courseno,t.book_isbn,t.book_title FROM course c,book_adoption ba,text t WHERE c.courseno=ba.courseno AND ba.book_isbn=t.book_isbn AND c.dept='CSE' AND 2<( SELECT COUNT(book_isbn) FROM book_adoption b WHERE c.courseno=b.courseno) ORDER BY t.book_title
```

PROGRAM 9: MOVIE DATABASE

Consider the schema for Movie Database:

ACTOR(Act_id, Act_Name, Act_Gender)

DIRECTOR(Dir_id, Dir_Name, Dir_Phone)

MOVIES(Mov_id, Mov_Title, Mov_Year,

Mov_Lang, Dir_id)

MOVIE_CAST(Act_id, Mov_id, Role)

RATING(Mov_id, Rev_Stars)

Write SQL queries to

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

The screenshot shows the phpMyAdmin interface for a database named 'movie'. The left sidebar lists various databases. The main area is titled 'Table: MOVIES' with a query result showing two rows of movie titles: 'Red Sparrow' and 'Mission Impossible'.

MOVIE_TITLE
Red Sparrow
Mission Impossible

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

The screenshot shows the phpMyAdmin interface for a database named 'movie'. The current table is 'MOVIES'. The SQL query executed was:

```
SELECT M.MOVIE_TITLE FROM ACTOR A,MOVIE_CAST C,MOVIES M WHERE A.ACT_ID=C.ACT_ID AND C.MOVIE_ID=M.MOVIE_ID AND A.ACT_ID IN(SELECT ACT_ID FROM MOVIE_CAST GROUP BY ACT_ID HAVING COUNT(>)=2)
```

The results table displays four rows of movie titles:

MOVIE_TITLE
War of the Worlds
Mission Impossible
Spider Man
Avatar

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

The screenshot shows the phpMyAdmin interface for a database named 'movie'. The current table is 'ACTOR'. The SQL query executed was:

```
SELECT A.ACT_NAME FROM ACTOR A JOIN MOVIE_CAST MC ON A.ACT_ID=MC.ACT_ID JOIN MOVIES M ON MC.MOVIE_ID=M.MOVIE_ID WHERE M.MOVIE_YEAR NOT BETWEEN 2000 AND 2015
```

The results table displays three actor names:

ACT_NAME
Tom Cruise
Leonardo
Jennifer Lawrence

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.

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1
- Database:** movie
- Table:** MOVIES
- Query Result:**

```
SELECT M.MOVIE_TITLE, MAX(R.RATING_STARS) AS MAXIMUM_RATING FROM MOVIES M, RATING R WHERE M.MOVIE_ID = R.MOVIE_ID GROUP BY M.MOVIE_TITLE HAVING COUNT(R.RATING_STARS)>1 ORDER BY M.MOVIE_TITLE
```

Showing rows 0 - 6 (7 total, Query took 0.0678 seconds.) [MOVIE_TITLE: AVATAR... - WAR OF THE WORLDS...]

MOVIE_TITLE	MAXIMUM_RATING
Avatar	4
Iron Man	5
Mission Impossible	5
Red Sparrow	3
Spider Man	4
Titanic	4
War of the Worlds	3

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

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1
- Database:** movie
- Table:** RATING
- Query Result:**

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

2 rows affected. (Query took 0.0580 seconds.)

Current selection does not contain a unique column. Grid edit, checkbox, Edit and Delete features are not available.

Showing rows 0 - 6 (7 total, Query took 0.0041 seconds.)

MOVIE_ID	RATING_STARS
1	5
2	5
3	5
4	3
5	4
6	4
7	5

PROGRAM 10:COLLEGE DATABASE

Consider the schema for College Database:

STUDENT(USN, SName, Address, Phone, Gender)

SEMSEC(SSID, Sem, Sec)

CLASS(USN, SSID)

SUBJECT(Subcode, Title, Sem, Credits)

IAMARKS(USN, Subcode, SSID, Test1, Test2, Test3, FinallA)

Write SQL queries to

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

The screenshot shows the phpMyAdmin interface with the following details:

- Left Panel:** Shows the database structure with a tree view of tables and views under the "New" database.
- Top Bar:** Includes tabs for Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Events, Triggers, and Tracking.
- Query Results:**
 - SQL Query: `SELECT S.USN, S.SNAME, S.ADDRESS, S.PHONE, S.GENDER, 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 = 'A'`
 - Results:

USN	SNAME	ADDRESS	PHONE	GENDER	SEM	SEC
1RN15CS011	AJAY	TUMKUR	1232654578	M	4	A
1RN15CS029	CHITRA	DAVANGERE	1232654578	F	4	A
 - Operations: Print, Copy to clipboard, Export, Display chart, Create view.
 - Bookmark this SQL query: Label input field, Let every user access this bookmark checkbox.

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

The screenshot shows the phpMyAdmin interface with the following details:

- Left sidebar:** Shows the database structure with tables like banking enterprise, book_dealer_134, and stu_enrollment_134.
- Query results:** A table showing the count of male (M) and female (F) students across different semesters (SEM) and sections (SEC).
- Table Data:**

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

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

The screenshot shows the phpMyAdmin interface with the following details:

- Left sidebar:** Shows the database structure with tables like banking enterprise, book_dealer_134, and stu_enrollment_134.
- View creation:** A view named STU_TEST1_MARKS_VIEW is being created with the following SQL query:

```
CREATE VIEW STU_TEST1_MARKS_VIEW AS SELECT TEST1, SUBCODE FROM IAMARKS WHERE USN = '1RN13CS091'
```

- View results:** The view contains 5 rows of data.
- Table Data:**

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

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

The screenshot shows the phpMyAdmin interface for the 'college_134' database. The left sidebar lists various databases and tables. The main area is titled 'Table: STUDENT' and displays the following SQL query results:

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

The table has the following columns: USN, SNAME, ADDRESS, PHONE, GENDER, and CAT. There are five rows of data, all showing the same values: USN 1RN13CS091, SNAME TEESHA, ADDRESS BENGALURU, PHONE 1232654578, GENDER F, and CAT WEAK.

USN	SNAME	ADDRESS	PHONE	GENDER	CAT
1RN13CS091	TEESHA	BENGALURU	1232654578	F	WEAK
1RN13CS091	TEESHA	BENGALURU	1232654578	F	WEAK
1RN13CS091	TEESHA	BENGALURU	1232654578	F	WEAK
1RN13CS091	TEESHA	BENGALURU	1232654578	F	WEAK