

DBMS LAB EXPERIMENTS

Experiment 1:

You need to create new database and insert below data to three different tables.

Data Dictionary:

Create three tables with below schema:

Sailors (sid: integer, sname: string, rating: integer, age: real)

Boats (bid: integer, bname:string, color:string)

Reserves (sid: integer, bid: integer, day: date)

Source code:

Create database record;

use record;

```
create table sailors (sid integer not null, sname varchar(32), rating integer, age real, CONSTRAINT PK_sailors PRIMARY KEY (sid) );
```

```
INSERT INTO sailors( sid, sname, rating, age ) VALUES ( 22, 'Rahul', 10, 25.0 ), (29, 'Anand', 9, 26.0), (31, 'Niket', 7, 45.0), (32, 'Neha', 5, 56.6),
```

```
(58, 'Bijay', 4, 23.5), (64, 'Thomas', 8, 35.0), (71, 'Rusty', 8, 29.5), (74, 'Venky', 6, 63.5), (85, 'Alfred', 1, 42.5), (95, 'Vikky', 9, 24.5);
```

```
create table boats (bid integer not null, boatname varchar(45), color varchar(32), constraint PK_boats primary key (bid) );
```

```
INSERT INTO boats ( bid, boatname, color ) VALUES (101, 'Waterking', 'Red'), (102, 'Waterking', 'Blue'), (103, 'Marine', 'Red'), (104, 'Seaway', 'Green');
```

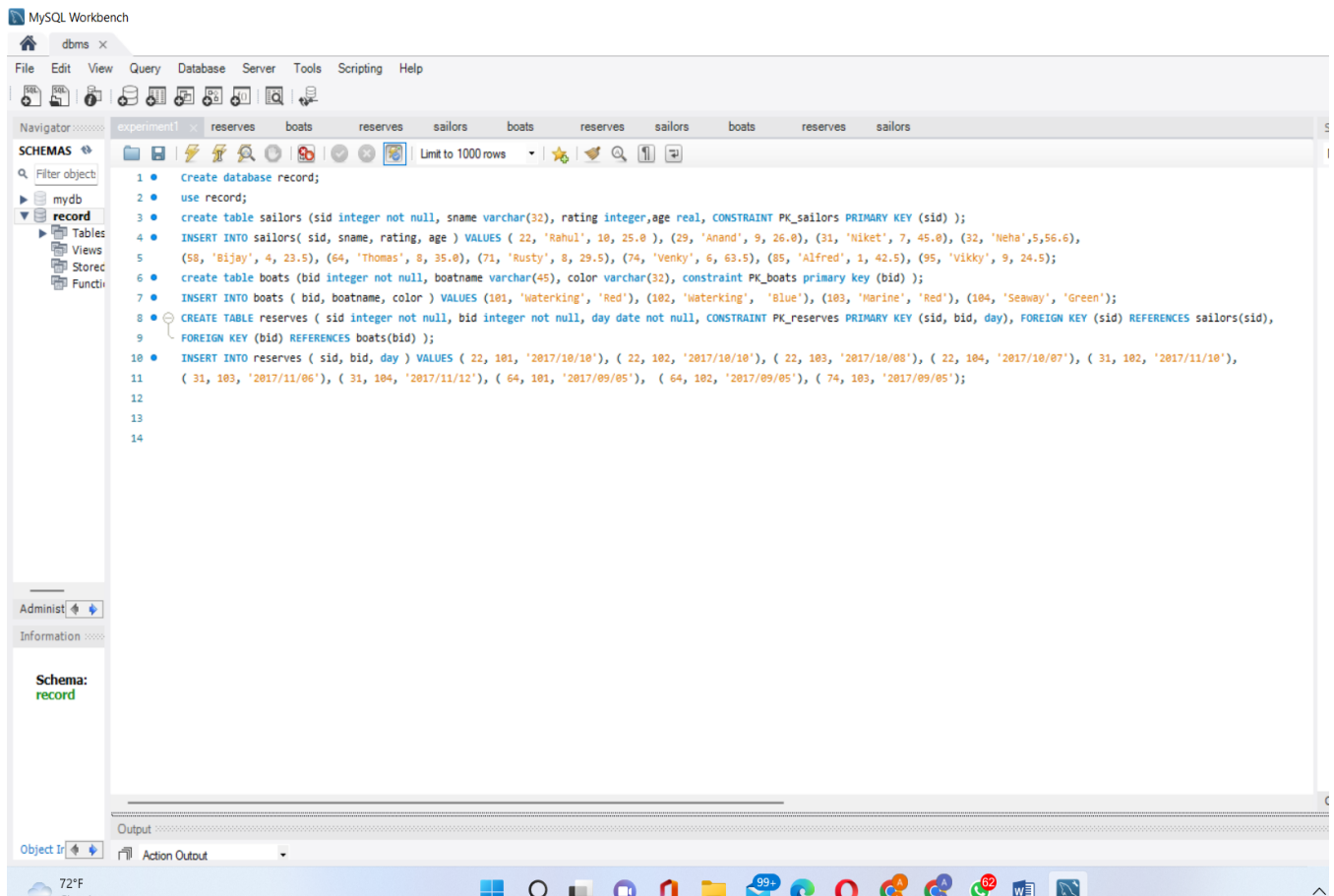
```
CREATE TABLE reserves ( sid integer not null, bid integer not null, day date not null, CONSTRAINT PK_reserves PRIMARY KEY (sid, bid, day), FOREIGN KEY (sid) REFERENCES sailors(sid),
```

```
FOREIGN KEY (bid) REFERENCES boats(bid) );
```

```
INSERT INTO reserves ( sid, bid, day ) VALUES ( 22, 101, '2017/10/10'), ( 22, 102, '2017/10/10'), ( 22, 103, '2017/10/08'), ( 22, 104, '2017/10/07'), ( 31, 102, '2017/11/10'),
```

```
( 31, 103, '2017/11/06'), ( 31, 104, '2017/11/12'), ( 64, 101, '2017/09/05'), ( 64, 102, '2017/09/05'), ( 74, 103, '2017/09/05');
```

Output:



The screenshot shows the MySQL Workbench interface with a SQL editor containing the following queries:

```
1 • Create database record;
2 • use record;
3 • create table sailors (sid integer not null, sname varchar(32), rating integer, age real, CONSTRAINT PK_sailors PRIMARY KEY (sid) );
4 • INSERT INTO sailors( sid, sname, rating, age ) VALUES ( 22, 'Rahul', 10, 25.0 ), ( 29, 'Anand', 9, 26.0), ( 31, 'Niket', 7, 45.0), ( 32, 'Neha', 5, 56.6),
5 ( 58, 'Bijay', 4, 23.5), ( 64, 'Thomas', 8, 35.0), ( 71, 'Rusty', 8, 29.5), ( 74, 'Venky', 6, 63.5), ( 85, 'Alfred', 1, 42.5), ( 95, 'Vikky', 9, 24.5);
6 • create table boats (bid integer not null, boatname varchar(45), color varchar(32), constraint PK_boats primary key (bid) );
7 • INSERT INTO boats ( bid, boatname, color ) VALUES ( 101, 'Waterking', 'Red'), ( 102, 'Waterking', 'Blue'), ( 103, 'Marine', 'Red'), ( 104, 'Seaway', 'Green');
8 • CREATE TABLE reserves ( sid integer not null, bid integer not null, day date not null, CONSTRAINT PK_reserves PRIMARY KEY (sid, bid, day), FOREIGN KEY (sid) REFERENCES sailors(sid),
9 FOREIGN KEY (bid) REFERENCES boats(bid) );
10 • INSERT INTO reserves ( sid, bid, day ) VALUES ( 22, 101, '2017/10/10'), ( 22, 102, '2017/10/10'), ( 22, 103, '2017/10/08'), ( 22, 104, '2017/10/07'), ( 31, 102, '2017/11/10'),
11 ( 31, 103, '2017/11/06'), ( 31, 104, '2017/11/12'), ( 64, 101, '2017/09/05'), ( 64, 102, '2017/09/05'), ( 74, 103, '2017/09/05');
12
13
14
```

The interface includes a Navigator pane on the left showing the 'record' schema, a bottom status bar with '72°F', and a Windows taskbar at the very bottom.

Boats Table:

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: experiment1 reserves boats reserves sailors boats x

SCHEMAS

Filter objects

mydb

record

Tables

boats

reserves

sailors

Views

Stored Procedures

Functions

1 • SELECT * FROM record.boats;

Limit to 1000 rows

Result Grid

Filter Rows:

Edit: Export/Import: Wrap Cell Content:

bid	boatname	color
101	Waterking	Red
102	Waterking	Blue
103	Marine	Red
104	Seaway	Green
NULL	NULL	NULL

boats 1 x

Reserves Table:

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: experiment1 reserves boats reserves sailors boats reserves x

SCHEMAS

Filter objects

mydb

record

Tables

boats

reserves

sailors

Views

Stored Procedures

Functions

1 • SELECT * FROM record.reserves;

Limit to 1000 rows

Result Grid

Filter Rows:

Edit: Export/Import: Wrap Cell Content:

sid	bid	day
22	101	2017-10-10
64	101	2017-09-05
22	102	2017-10-10
31	102	2017-11-10
64	102	2017-09-05
22	103	2017-10-08
31	103	2017-11-06
74	103	2017-09-05
22	104	2017-10-07
31	104	2017-11-12
NULL	NULL	NULL

Administration Schemas

Information

Sailors table:

MySQL Workbench

dbms x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

mydb

record

Tables

boats

reserves

sailors

Views

Stored Procedures

Functions

experiment1 reserves boats reserves sailors boats reserves sailors x

Limit to 1000 rows

1 • `SELECT * FROM record.sailors;`

Result Grid

Filter Rows:

Edit: Export/Import: Wrap Cell Content:

sid	sname	rating	age
22	Rahul	10	25
29	Anand	9	26
31	Niket	7	45
32	Neha	5	56.6
58	Bijay	4	23.5
64	Thomas	8	35
71	Rusty	8	29.5
74	Venky	6	63.5
85	Alfred	1	42.5
95	Vikky	9	24.5
•	NULL	NULL	NULL

No object selected

Lab Experiment 2:

You need to find solution of below questions based on Experiment 1.

1. Find the names and sids of sailors who have reserved a red or a Green boat?

`SELECT R.sid FROM boats B, reserves R WHERE R.bid = B.bid AND B.color = "red" UNION SELECT R2.sid FROM boats B2, reserves R2 WHERE R2.bid = B2.bid AND B2.color = "green" ;`

The screenshot shows the MySQL Workbench interface. The 'Schemas' pane on the left displays a tree view of the database structure, including tables like 'account', 'borrower', 'branch', 'customer', 'depositor', 'loan', 'reserves', and 'sailors'. The 'Query' editor in the center contains the following SQL query:

```
1 • SELECT R.sid FROM boats B, reserves R WHERE R.bid = B.bid AND B.color = "red" UNION SELECT
2   R2.sid FROM boats B2, reserves R2 WHERE R2.bid = B2.bid AND
3     B2.color = "green" ;
4
5
```

The 'Result Grid' at the bottom shows the output of the query, which is a list of sailor IDs (sid) who have reserved a red or green boat. The results are displayed in a table with one column, 'sid', and four rows of data.

sid
22
64
31
74

2. Find the names and sids of sailors who have reserved a red and a Green boat?
3. Find all sids of sailors who have a rating of 10 or reserved boat 104?
4. Find the names of sailors who have reserved boat 103?
5. Find the names of sailors who have reserved a red boat?
6. Find the names of sailors who have not reserved a red boat?
7. Find sailors whose rating is better than some sailor called Horatio?
8. Find the sailors with the highest rating?
9. Find the names of sailors who have reserved both a red and a green boat?
10. Find the names of sailors who have reserved boat no 103?
11. Find the names of sailors who have reserved all boats?

Lab Experiment 3:

Experiment 3 to Experiment 8 are related to each other. You need to follow same data and table details for the solution. You need create new database with below table details and have to insert data based on below tables.

Branch Schema <branch-name, branch-city, assets>

Customer Schema <customer-name, customer-street, customer-city>

Loan Schema <loan-number, branch-name, amount>

Borrower Schema <customer-name, loan-number>

Account Scheme <account-number, branch-name, balance>

Depositor Scheme <customer-name, account-number>

Source code:

```
create database mydb;
```

```
use mydb;
```

Create table branch (branch_name varchar(20), branch_city varchar(20), assets integer);

insert into branch(branch_name, branch_city, assets) values("Brighton", "Brooklyn", 7100000),("Downtown", "Brooklyn", 9000000),("Mianus", "Horseneck", 3700000),("NorthTown", "Rye", 400000),("Perryridge", "Horseneck", 1700000),("Pownal", "Bennington", 300000),("Redwood", "Palo Alto", 2100000),("Round Hill", "Horseneck", 800000);

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

insert into customer(customer_name, customer_street, customer_city) values("Adams", "Spring", "Pittsfield"),("Brooks", "Senator", "Brooklyn"),("Curry", "North", "Rye"),("Glenn", "Sand Hill", "Woodside"),("Green", "Walnut", "Stamford"),("Hayes", "Main", "Harrison"),("Johnson", "Alma", "Palo Alto"),("Jones", "Main", "Harrison"),("Lindsay", "Park", "Pittsfield"),("Turner", "Putnam", "Stamford"),("Smith", "North", "Rye"),("Williams", "Nassau", "Princeton");

create table loan(loan_number varchar(20) primary key, branch_name varchar(20), amount integer);

insert into loan(loan_number, branch_name, amount) values("L-11", "Round Hill", 900),("L-14", "Downtown", 1500), ("L-15", "Perryridge", 1500),("L-16", "Perryridge", 1300),("L-17", "Downtown", 1000), ("L-23", "Redwood", 2000), ("L-93", "Mianus", 500);

create table borrower(customer_name varchar(20), loan_number varchar(20), foreign key (loan_number) References loan(loan_number));

insert into borrower(customer_name, loan_number) values("Adams", "L-16"), ("Curry", "L-93"), ("Hayes", "L-15"), ("Jackson", "L-14"), ("Jones", "L-17"), ("Smith", "L-11"), ("Smith", "L-23"), ("Williams", "L-17");

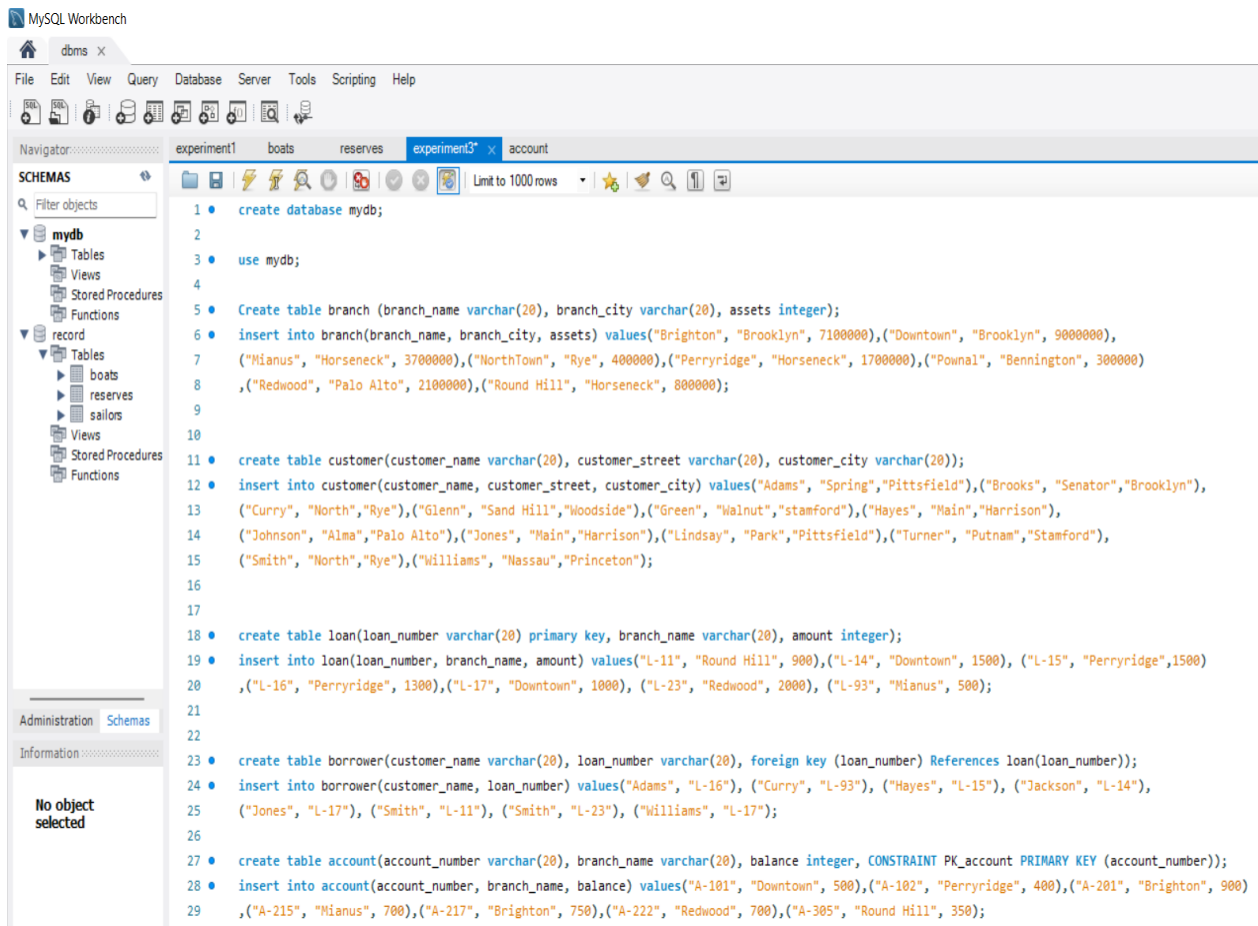
create table account(account_number varchar(20), branch_name varchar(20), balance integer, CONSTRAINT PK_account PRIMARY KEY (account_number));

insert into account(account_number, branch_name, balance) values("A-101", "Downtown", 500),("A-102", "Perryridge", 400),("A-201", "Brighton", 900),("A-215", "Mianus", 700),("A-217", "Brighton", 750),("A-222", "Redwood", 700),("A-305", "Round Hill", 350);

create table depositor (customer_name varchar(20), account_number varchar(20), foreign key (account_number) references account(account_number));

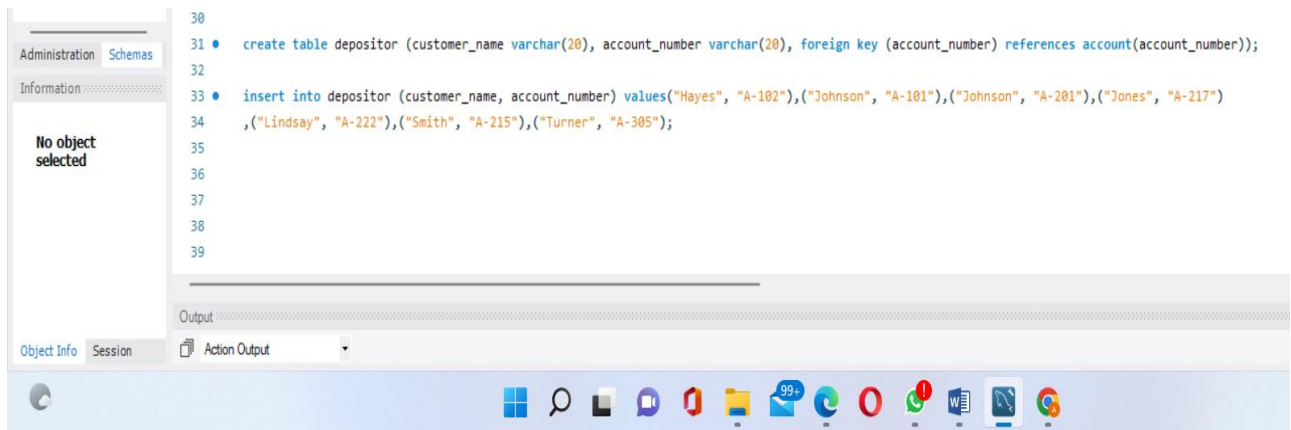
insert into depositor (customer_name, account_number) values("Hayes", "A-102"),("Johnson", "A-101"),("Johnson", "A-201"),("Jones", "A-217"),("Lindsay", "A-222"),("Smith", "A-215"),("Turner", "A-305");

Output:

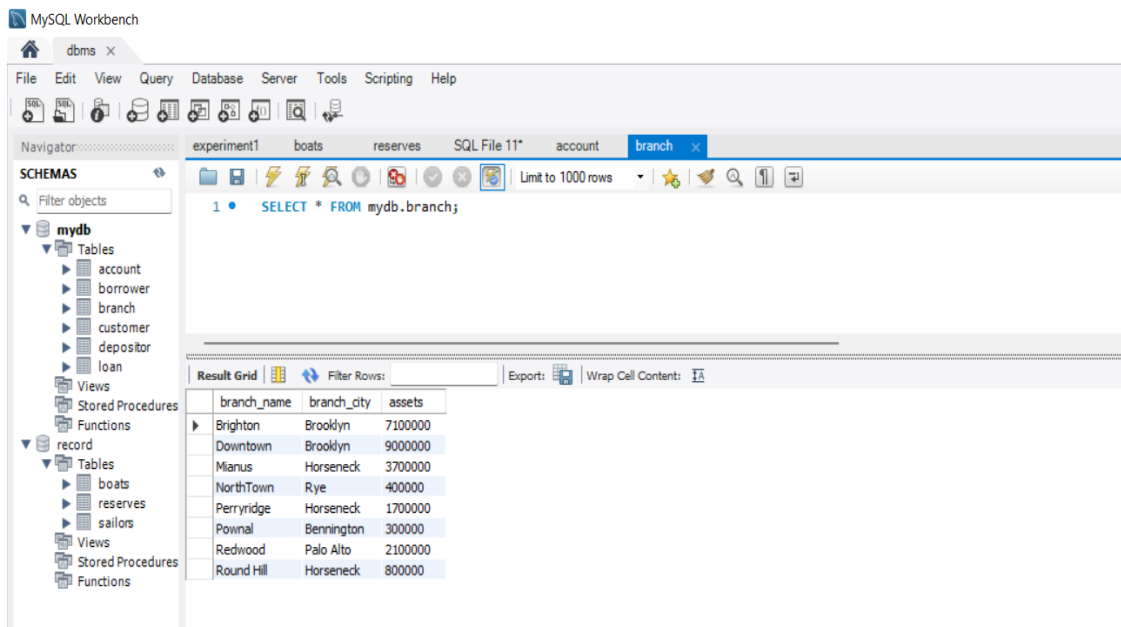


The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view showing the 'mydb' database and its tables: 'boats', 'reserves', 'sailors', 'customer', 'loan', 'borrower', and 'account'. The main editor window shows a SQL script with the following content:

```
1 • create database mydb;
2
3 • use mydb;
4
5 • Create table branch (branch_name varchar(20), branch_city varchar(20), assets integer);
6 • insert into branch(branch_name, branch_city, assets) values("Brighton", "Brooklyn", 7100000),("Downtown", "Brooklyn", 9000000),
7 ("Mianus", "Horseneck", 3700000),("NorthTown", "Rye", 400000),("Perryridge", "Horseneck", 1700000),("Pownal", "Bennington", 300000)
8 ,("Redwood", "Palo Alto", 2100000),("Round Hill", "Horseneck", 800000);
9
10
11 • create table customer(customer_name varchar(20), customer_street varchar(20), customer_city varchar(20));
12 • insert into customer(customer_name, customer_street, customer_city) values("Adams", "Spring", "Pittsfield"),("Brooks", "Senator", "Brooklyn"),
13 ("Curry", "North", "Rye"),("Glenn", "Sand Hill", "Woodside"),("Green", "Walnut", "Stamford"),("Hayes", "Main", "Harrison"),
14 ("Johnson", "Alma", "Palo Alto"),("Jones", "Main", "Harrison"),("Lindsay", "Park", "Pittsfield"),("Turner", "Putnam", "Stamford"),
15 ("Smith", "North", "Rye"),("Williams", "Nassau", "Princeton");
16
17
18 • create table loan(loan_number varchar(20) primary key, branch_name varchar(20), amount integer);
19 • insert into loan(loan_number, branch_name, amount) values("L-11", "Round Hill", 900),("L-14", "Downtown", 1500), ("L-15", "Perryridge", 1500)
20 ,("L-16", "Perryridge", 1300),("L-17", "Downtown", 1000), ("L-23", "Redwood", 2000), ("L-93", "Mianus", 500);
21
22
23 • create table borrower(customer_name varchar(20), loan_number varchar(20), foreign key (loan_number) References loan(loan_number));
24 • insert into borrower(customer_name, loan_number) values("Adams", "L-16"), ("Curry", "L-93"), ("Hayes", "L-15"), ("Jackson", "L-14"),
25 ("Jones", "L-17"), ("Smith", "L-11"), ("Smith", "L-23"), ("Williams", "L-17");
26
27 • create table account(account_number varchar(20), branch_name varchar(20), balance integer, CONSTRAINT PK_account PRIMARY KEY (account_number));
28 • insert into account(account_number, branch_name, balance) values("A-101", "Downtown", 500),("A-102", "Perryridge", 400),("A-201", "Brighton", 900)
29 ,("A-215", "Mianus", 700),("A-217", "Brighton", 750),("A-222", "Redwood", 700),("A-305", "Round Hill", 350);
```

Branch table:



Customer table:

MySQL Workbench

dbms x

File Edit View Query Database Server Tools Scripting Help

Navigators: experiment1 boats reserves SQL File 11* account branch customer x

Limit to 1000 rows

1 • `SELECT * FROM mydb.customer;`

Result Grid

customer_name	customer_street	customer_city
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Turner	Putnam	Stamford
Smith	North	Rye
Williams	Nassau	Princeton

Loan Table:

MySQL Workbench

dbms x

File Edit View Query Database Server Tools Scripting Help

Navigators: experiment1 boats reserves SQL File 11* account branch customer loan x

Limit to 1000 rows

1 • `SELECT * FROM mydb.loan;`

Result Grid

loan_number	branch_name	amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500
NULL	NULL	NULL

Borrow Table:

MySQL Workbench

dbms x

File Edit View Query Database Server Tools Scripting Help

Navigator: experiment1 boats reserves SQL File 11* account branch customer loan borrower x

Limit to 1000 rows

1 • `SELECT * FROM mydb.borrower;`

Result Grid Filter Rows: Export: Wrap Cell Content: [FA](#)

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

SCHEMAS

Filter objects

mydb

- Tables
 - account
 - borrower
 - branch
 - customer
 - depositor
 - loan
- Views
- Stored Procedures
- Functions

record

- Tables
 - boats
 - reserves
 - sailors
- Views
- Stored Procedures
- Functions

Deposit Table:

MySQL Workbench

dbms x

File Edit View Query Database Server Tools Scripting Help

Navigator: experiment1 boats reserves experiment3* account depositor x

Limit to 1000 rows

1 • `SELECT * FROM mydb.depositor;`

SCHEMAS

Filter objects

- mydb
 - Tables
 - account
 - borrower
 - branch
 - customer
 - depositor
 - loan
 - Views
 - Stored Procedures
 - Functions
- record
 - Tables
 - boats
 - reserves
 - sailors
 - Views
 - Stored Procedures
 - Functions

Administration Schemas

Information

No object selected

Result Grid

Filter Rows:

Export: Wrap Cell Content: `1`

	customer_name	account_number
▶	Hayes	A-102
	Johnson	A-101
	Johnson	A-201
	Jones	A-217
	Lindsay	A-222
	Smith	A-215
	Turner	A-305

Lab Experiment 4:

You need to find solution of below questions based on tables using Experiment Number 03.

1.To list all the fields from the table Customer.

`SELECT * from customer;`

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of databases, with 'mydb' expanded to show tables including 'customer'. The main editor window has a query tab titled 'dbmsexperiment 4' containing the SQL query: `SELECT * from customer;`. Below the query editor, the 'Result Grid' displays the data from the 'customer' table. The table has three columns: 'customer_name', 'customer_street', and 'customer_city'. The data is as follows:

customer_name	customer_street	customer_city
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Turner	Putnam	Stamford
Smith	North	Rye
Williams	Nassau	Princeton

2.To list all the fields after applying arithmetic operations on column amount (amount*100).

```
SELECT loan_number, branch_name, amount *100 FROM Loan;
```

MySQL Workbench

dbms x

File Edit View Query Database Server Tools Scripting Help

Navigator: account experiment3 dbmsexperiment 4*

Limit to 1000 rows

1 • SELECT * from customer;

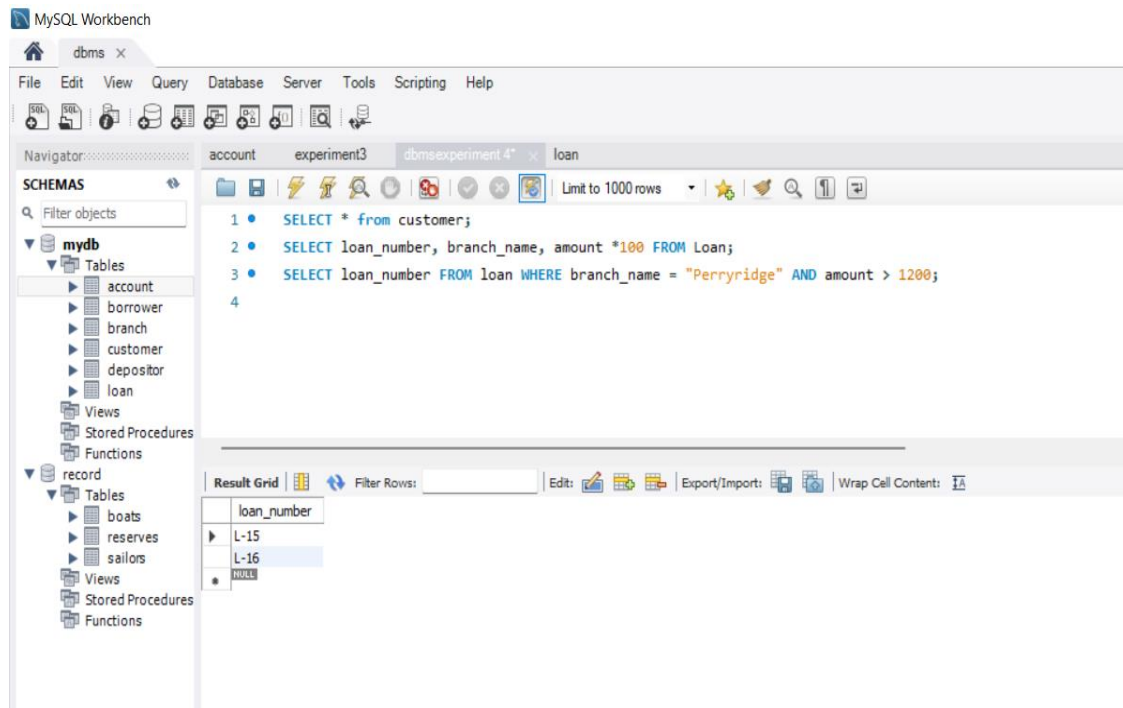
2 • SELECT loan_number, branch_name, amount *100 FROM Loan;

Result Grid

	loan_number	branch_name	amount *100
▶	L-11	Round Hill	90000
	L-14	Downtown	150000
	L-15	Perryridge	150000
	L-16	Perryridge	130000
	L-17	Downtown	100000
	L-23	Redwood	200000
	L-93	Mianus	50000

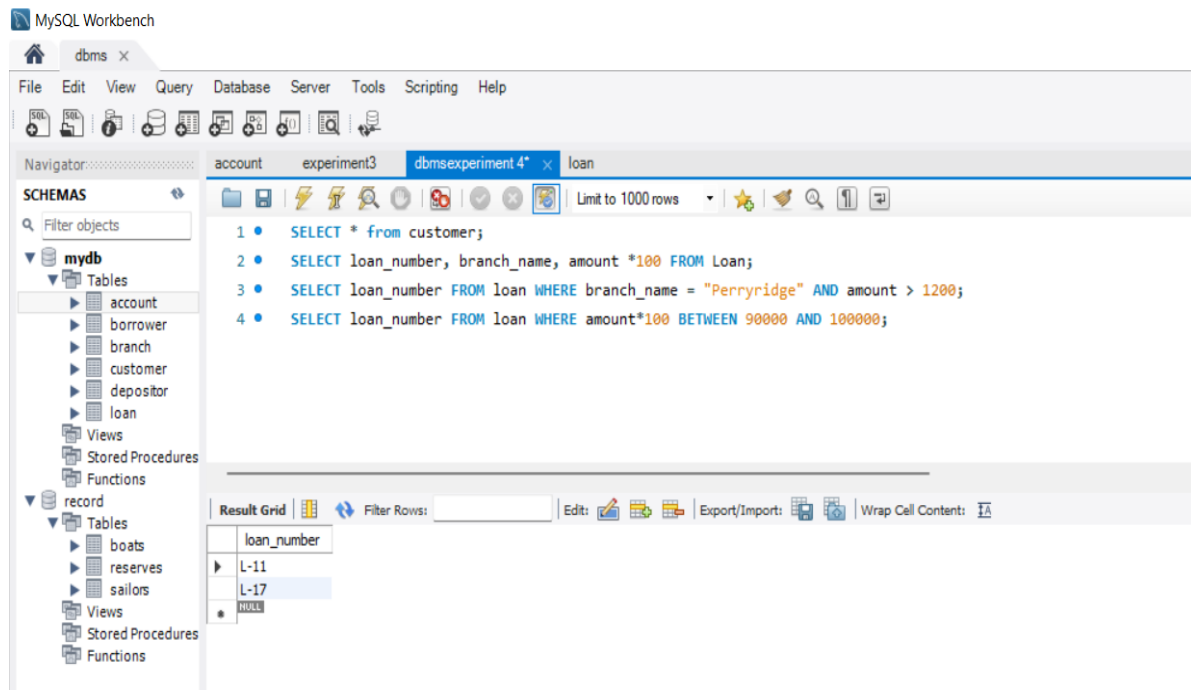
3. Find all loan numbers for loans made at the Perryridge branch with loan amounts greater than Rs1200.

```
SELECT loan_number FROM loan WHERE branch_name = "Perryridge" AND amount > 1200;
```



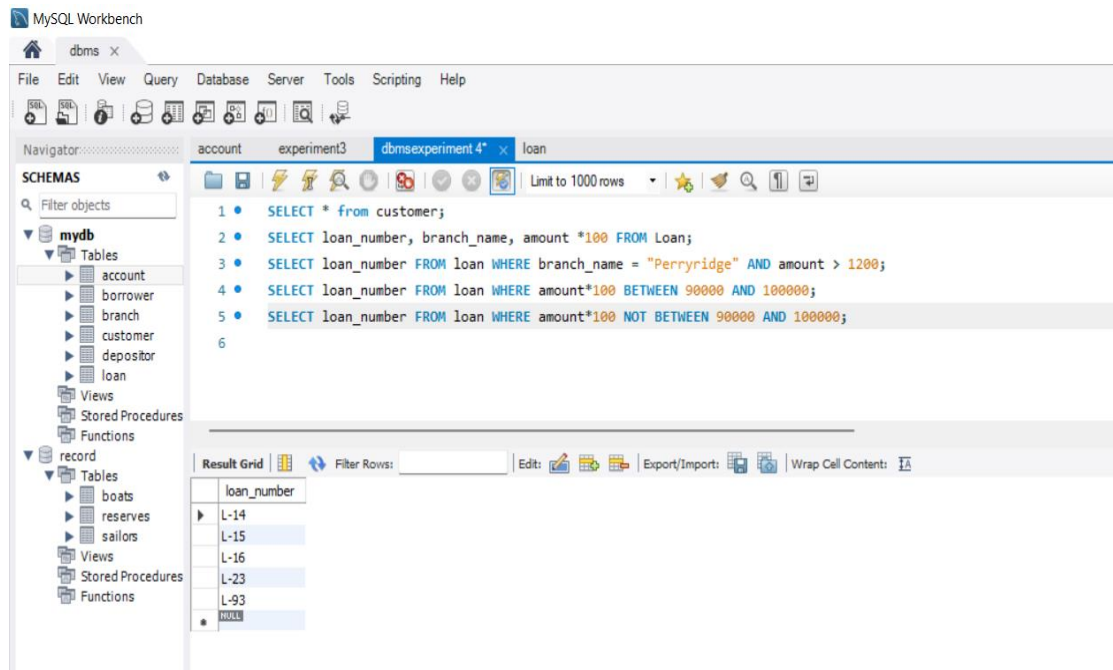
4. Find all loan numbers for loans with loan amounts between Rs90,000 and Rs100,000. (amount*100).

SELECT loan_number FROM loan WHERE amount*100 BETWEEN 90000 AND 100000;



5. Find all loan numbers for loans with loan amounts not between Rs90,000 and Rs100,000.
(amount*100)

SELECT loan_number FROM loan WHERE amount*100 NOT BETWEEN 90000 AND 100000;



6. For all customers who have a loan from the bank, find their names, loan numbers and loan amounts.

```
SELECT customer_name, borrower.loan_number, amount FROM borrower, loan WHERE  
borrower.loan_number = loan.loan_number;
```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with 'mydb' expanded, showing tables like 'account', 'borrower', 'branch', 'customer', 'depositor', and 'loan'. The main editor window contains a SQL query with eight lines. The 'Result Grid' at the bottom shows the output of the query, displaying columns 'customer_name', 'loan_number', and 'amount' with eight rows of data.

```
1 • SELECT * from customer;  
2 • SELECT loan_number, branch_name, amount *100 FROM loan;  
3 • SELECT loan_number FROM loan WHERE branch_name = "Perryridge" AND amount > 1200;  
4 • SELECT loan_number FROM loan WHERE amount*100 BETWEEN 90000 AND 100000;  
5 • SELECT loan_number FROM loan WHERE amount*100 NOT BETWEEN 90000 AND 100000;  
6 • SELECT customer_name, borrower.loan_number, amount FROM borrower, loan WHERE borrower.loan_number = loan.loan_number;  
7  
8
```

customer_name	loan_number	amount
Smith	L-11	900
Jackson	L-14	1500
Hayes	L-15	1500
Adams	L-16	1300
Jones	L-17	1000
Williams	L-17	1000
Smith	L-23	2000
Curry	L-93	500

Lab Experiment 5:

You need to find solution of below questions based on tables using Experiment Number 03.

1. Find the customer names, loan numbers and loan amounts for all loans at the Perryridge branch.

SELECT customer_name, borrower.loan_number, amount **FROM** borrower, loan **WHERE** borrower.loan_number = loan.loan_number **AND** branch_name = "Perryridge";

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 SELECT customer_name, borrower.loan_number, amount FROM borrower, loan WHERE borrower.loan_number = loan.loan_number
2 AND branch_name = "Perryridge";
```

The Results window displays the following data:

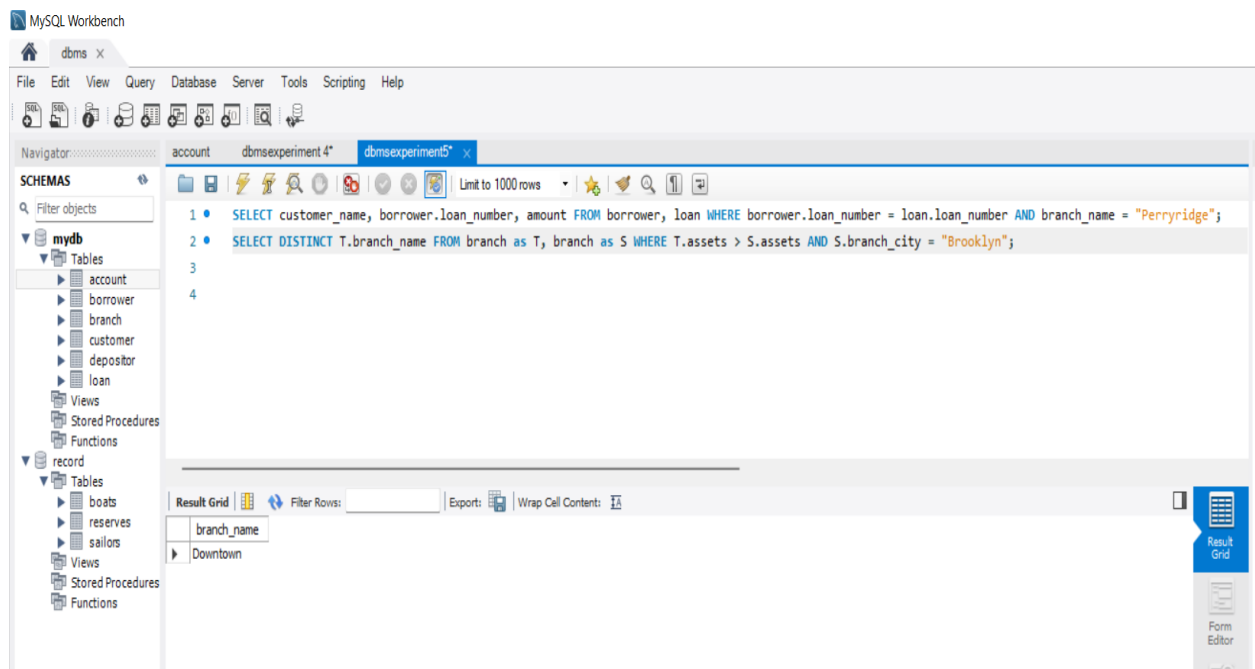
customer_name	loan_number	amount
Hayes	L-15	1500
Adams	L-16	1300

The bottom status bar shows the query execution details:

#	Time	Action	Message	Duration / Fetch
8	16:19:33	SELECT customer_name, borrower.loan_number, amount FROM borrower, loan WHERE borrower.loan_number = L...	8 row(s) returned	0.187 sec / 0.000 sec
9	16:30:38	SELECT customer_name, borrower.loan_number, amount FROM borrower, loan WHERE borrower.loan_number = L...	2 row(s) returned	0.094 sec / 0.000 sec

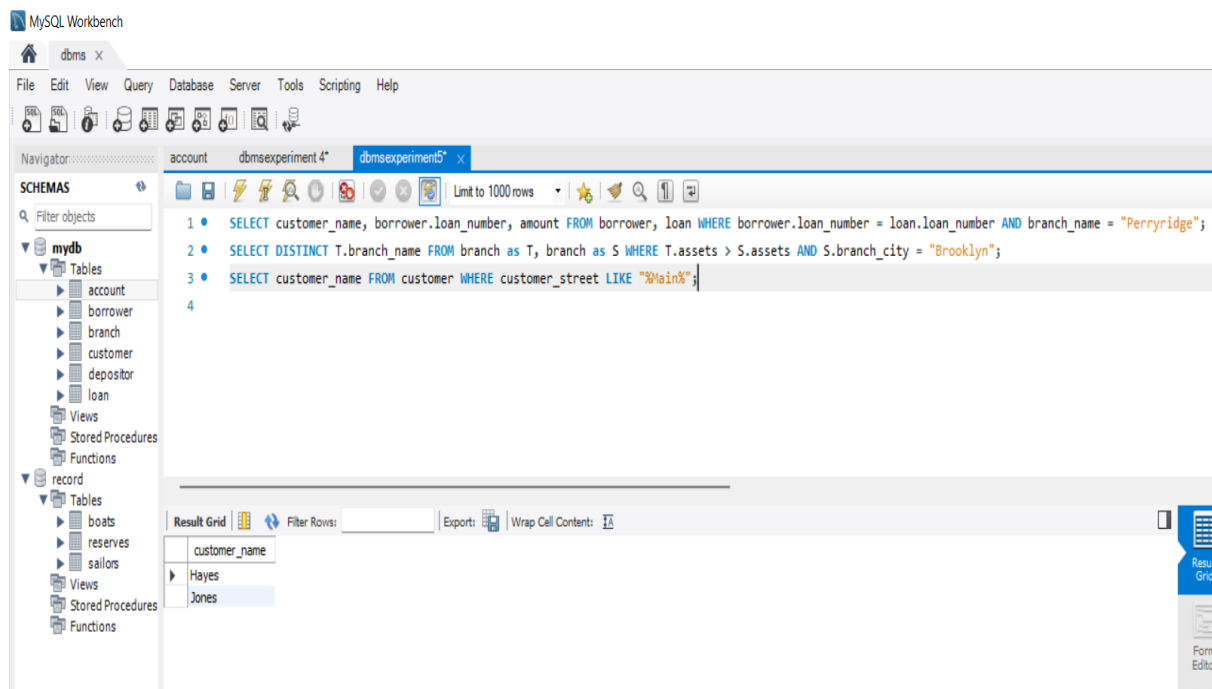
2. Find the names of all branches that have assets greater than atleast one branch located in Brooklyn.

```
SELECT DISTINCT T.branch_name FROM branch as T, branch as S WHERE T.assets > S.assets  
AND S.branch_city = "Brooklyn";
```



3. Find the names of all customers whose street address includes the substring 'Main'.

```
SELECT customer_name FROM customer WHERE customer_street LIKE "%Main%";
```



4. To list in alphabetic order all customers who have a loan at the Perryridge branch.

```
SELECT DISTINCT customer_name FROM borrower B, loan L WHERE B.loan_number =  
L.loan_number AND branch_name = "Perryridge" ORDER BY customer_name;
```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view of the 'mydb' database, including tables like 'account', 'borrower', 'branch', 'customer', 'depositor', and 'loan'. The main editor window contains a SQL query with four lines, the fourth of which is the query from the previous block. Below the query editor, the 'Result Grid' is visible, showing a table with two rows: 'Adams' and 'Hayes' under the 'customer_name' column. The interface also includes a menu bar (File, Edit, View, Query, Database, Server, Tools, Scripting, Help) and a toolbar with various icons for file operations, query execution, and navigation.

```
1 • SELECT customer_name, borrower.loan_number, amount FROM borrower, loan WHERE borrower.loan_number = loan.loan_number AND branch_name = "Perryridge";  
2 • SELECT DISTINCT T.branch_name FROM branch as T, branch as S WHERE T.assets > S.assets AND S.branch_city = "Brooklyn";  
3 • SELECT customer_name FROM customer WHERE customer_street LIKE "Main%";  
4 • SELECT DISTINCT customer_name FROM borrower B, loan L WHERE B.loan_number = L.loan_number AND branch_name = "Perryridge" ORDER BY customer_name;
```

customer_name
Adams
Hayes

5. To list the entire loan info in descending order of amount.

```
SELECT * FROM loan ORDER BY amount DESC, loan_number ASC;
```

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including tables like 'account', 'borrower', 'branch', 'customer', 'depositor', and 'loan'. The main query editor window contains a SQL query with five statements. The fifth statement is the one being executed: `SELECT * FROM loan ORDER BY amount DESC, loan_number ASC;`. Below the query editor, the 'Result Grid' pane shows the output of the query, displaying a table with three columns: 'loan_number', 'branch_name', and 'amount'. The results are sorted by amount in descending order.

loan_number	branch_name	amount
L-23	Redwood	2000
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-11	Round Hill	900
L-93	Mianus	500
NULL	NULL	NULL

6. To find all customers having a loan, an account or both at the bank, without duplicates.

(SELECT customer_name FROM depositor) UNION (SELECT customer_name FROM borrower)

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view of the 'mydb' database, including tables like 'account', 'borrower', 'branch', 'customer', 'depositor', and 'loan'. The main editor window shows a SQL query with six lines, the last of which is the UNION query. The 'Result Grid' at the bottom displays the output of the query, listing customer names: Hayes, Johnson, Jones, Lindsay, Smith, Turner, Adams, Curry, Jackson, and Williams.

```
1 • SELECT customer_name, borrower.loan_number, amount FROM borrower, loan WHERE borrower.loan_number = loan.loan_number AND branch_name = "Perryridge";
2 • SELECT DISTINCT T.branch_name FROM branch as T, branch as S WHERE T.assets > S.assets AND S.branch_city = "Brooklyn";
3 • SELECT customer_name FROM customer WHERE customer_street LIKE "%Main%";
4 • SELECT DISTINCT customer_name FROM borrower B, loan L WHERE B.loan_number = L.loan_number AND branch_name = "Perryridge" ORDER BY customer_name;
5 • SELECT * FROM loan ORDER BY amount DESC, loan_number ASC;
6 • (SELECT customer_name FROM depositor) UNION (SELECT customer_name FROM borrower)
```

customer_name
Hayes
Johnson
Jones
Lindsay
Smith
Turner
Adams
Curry
Jackson
Williams

Lab Experiment 6:

You need to find solution of below questions based on tables using Experiment Number 03.

1. To find all customers having a loan, an account or both at the bank, with duplicates.

(SELECT customer_name FROM depositor) UNION ALL (SELECT customer_name FROM borrower);

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'mydb' expanded, showing tables like 'account', 'borrower', 'branch', 'customer', 'depositor', and 'loan'. The main window shows a SQL query in the 'Query' tab:

```
(SELECT customer_name FROM depositor) UNION ALL (SELECT customer_name FROM borrower);
```

The 'Result Grid' at the bottom shows the output of the query, listing customer names with duplicates:

customer_name
Hayes
Johnson
Johnson
Jones
Lindsay
Smith
Turner
Adams
Curry
Hayes
Jackson
Jones
Smith
Smith
Williams

2. To find all customers having both a loan and an account at the bank, without duplicates.

SELECT depositor.customer_name FROM depositor WHERE depositor.customer_name IN
(SELECT customer_name FROM Borrower);

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'mydb' expanded, showing tables like 'account', 'borrower', 'branch', 'customer', 'depositor', and 'loan'. The main query editor contains the following SQL code:

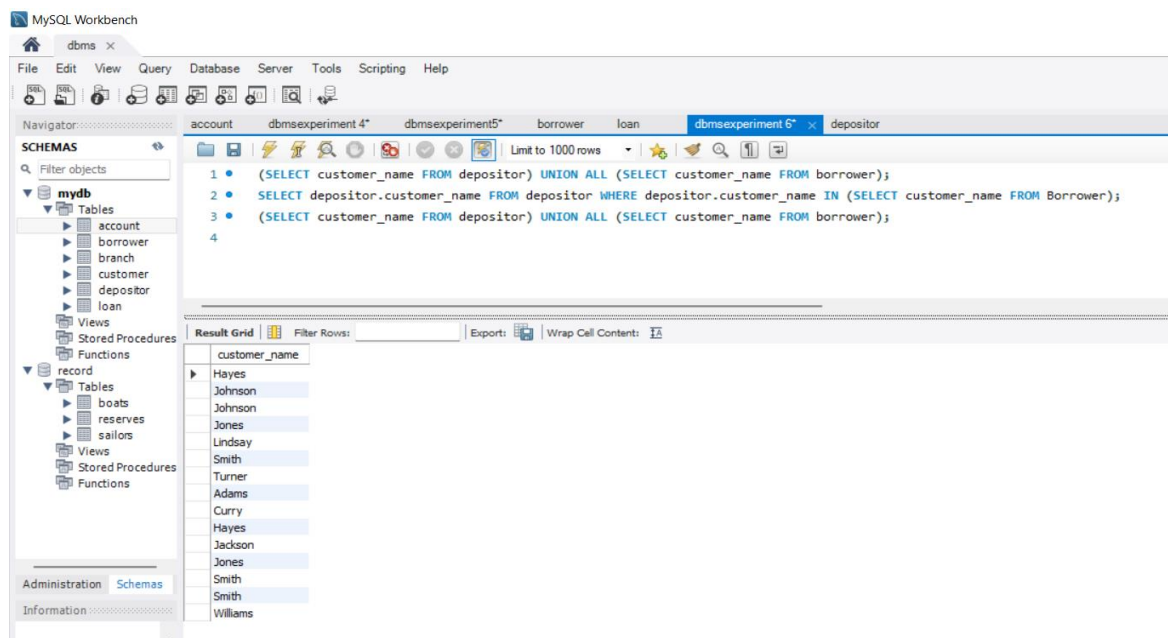
```
1 • (SELECT customer_name FROM depositor) UNION ALL (SELECT customer_name FROM borrower);
2 • SELECT depositor.customer_name FROM depositor WHERE depositor.customer_name IN (SELECT customer_name FROM Borrower);
3
```

The 'Result Grid' at the bottom shows the output of the query:

customer_name
Hayes
Jones
Smith

3. To find all customers having a loan, an account or both at the bank, with duplicates.

(SELECT customer_name FROM depositor) UNION ALL (SELECT customer_name FROM borrower);



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'mydb' expanded, showing tables like 'account', 'borrower', 'branch', 'customer', 'depositor', and 'loan'. The main editor window contains a SQL query in a tab labeled 'dbmsexperiment 6*'. The query is as follows:

```
1 • (SELECT customer_name FROM depositor) UNION ALL (SELECT customer_name FROM borrower);
2 • SELECT depositor.customer_name FROM depositor WHERE depositor.customer_name IN (SELECT customer_name FROM borrower);
3 • (SELECT customer_name FROM depositor) UNION ALL (SELECT customer_name FROM borrower);
4
```

Below the query editor, the 'Result Grid' is visible, showing a single column 'customer_name' with the following values:

customer_name
Hayes
Johnson
Johnson
Jones
Lindsay
Smith
Turner
Adams
Curry
Hayes
Jackson
Jones
Smith
Smith
Williams

4. To find all customers who have an account but no loan at the bank, without duplicates.

`SELECT DISTINCT customer_name FROM depositor WHERE customer_name NOT IN (SELECT customer_name FROM borrower);`

MySQL Workbench

dbms x

File Edit View Query Database Server Tools Scripting Help

Navigator: account dbmsexperiment 4* dbmsexperiment5* borrower loan dbmsexperiment 6* x depositor

SCHEMAS

Filter objects

mydb

Tables

- account
- borrower
- branch
- customer
- depositor
- loan

Views

Stored Procedures

Functions

record

Tables

- boats
- reserves
- sailors

Limit to 1000 rows

- 1 • `(SELECT customer_name FROM depositor) UNION ALL (SELECT customer_name FROM borrower);`
- 2 • `SELECT depositor.customer_name FROM depositor WHERE depositor.customer_name IN (SELECT customer_name FROM Borrower);`
- 3 • `(SELECT customer_name FROM depositor) UNION ALL (SELECT customer_name FROM borrower);`
- 4 • `SELECT DISTINCT customer_name FROM depositor WHERE customer_name NOT IN (SELECT customer_name FROM borrower);`

Result Grid

customer_name
Johnson
Lindsay
Turner

5. To find all customers who have an account but no loan at the bank, with duplicates.

```
SELECT customer_name FROM depositor WHERE customer_name NOT IN (SELECT customer_name FROM borrower);
```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'mydb' expanded, showing tables like 'account', 'borrower', 'branch', 'customer', 'depositor', and 'loan'. The main editor window contains a SQL query with six lines, including UNION ALL and NOT IN clauses. The 'Result Grid' at the bottom shows the output of the query, displaying a list of customer names: Johnson, Johnson, Lindsay, and Turner.

```
1 • (SELECT customer_name FROM depositor) UNION ALL (SELECT customer_name FROM borrower);
2 • SELECT depositor.customer_name FROM depositor WHERE depositor.customer_name IN (SELECT customer_name FROM Borrower);
3 • (SELECT customer_name FROM depositor) UNION ALL (SELECT customer_name FROM borrower);
4 • SELECT DISTINCT customer_name FROM depositor WHERE customer_name NOT IN (SELECT customer_name FROM borrower);
5 • SELECT customer_name FROM depositor WHERE customer_name NOT IN (SELECT customer_name FROM borrower);
6
```

customer_name
Johnson
Johnson
Lindsay
Turner

6. Find the average account balance at the Perryridge branch.

```
SELECT branch_name, AVG(balance) FROM Account WHERE branch_name = "Perryridge";
```

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view for the 'mydb' database, including tables like 'account', 'borrower', 'branch', 'customer', 'depositor', and 'loan'. The main editor window contains a SQL query with seven numbered lines. Line 6 is highlighted, showing the query: `SELECT branch_name, AVG(balance) FROM Account WHERE branch_name = "Perryridge";`. Below the query editor, the 'Result Grid' is visible, showing a single row with the columns 'branch_name' and 'AVG(balance)', and the values 'Perryridge' and '400.0000' respectively.

```
1 • (SELECT customer_name FROM depositor) UNION ALL (SELECT customer_name FROM borrower);
2 • SELECT depositor.customer_name FROM depositor WHERE depositor.customer_name IN (SELECT customer_name FROM Borrower);
3 • (SELECT customer_name FROM depositor) UNION ALL (SELECT customer_name FROM borrower);
4 • SELECT DISTINCT customer_name FROM depositor WHERE customer_name NOT IN (SELECT customer_name FROM borrower);
5 • SELECT customer_name FROM depositor WHERE customer_name NOT IN (SELECT customer_name FROM borrower);
6 • SELECT branch_name, AVG(balance) FROM Account WHERE branch_name = "Perryridge";
7
```

branch_name	AVG(balance)
Perryridge	400.0000