Department of Computer Science and Engineering

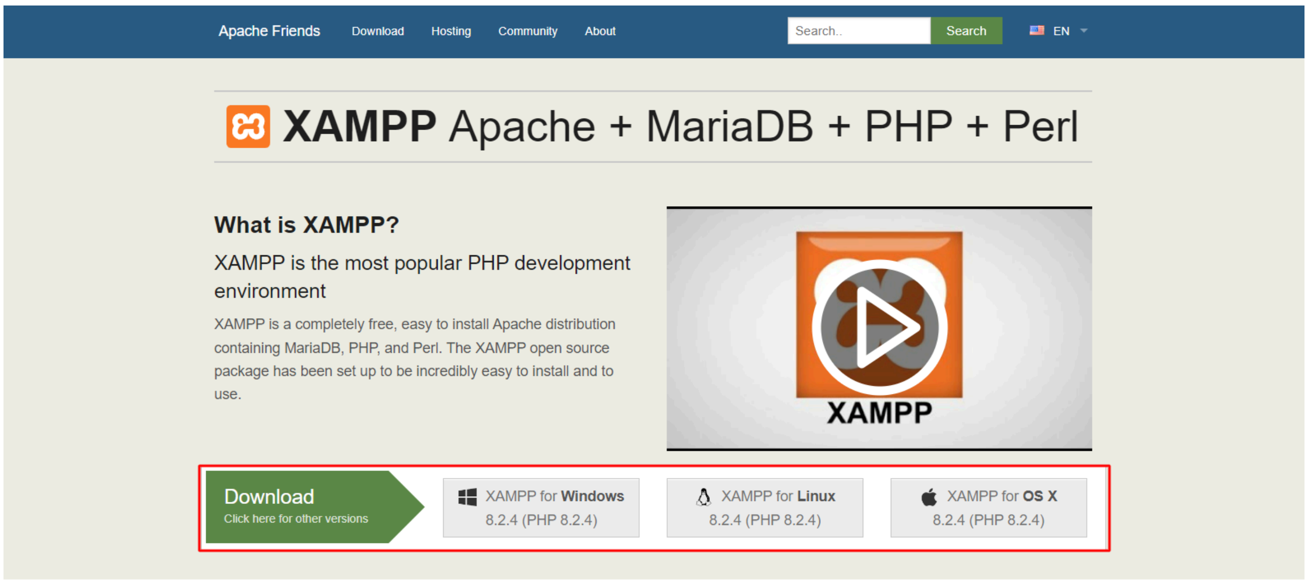
Course Code: CSE370	Credits: 1.5
Course Name: Database Systems	Semester: Fall 2025

Lab 01

Part A: Setting Up and Connecting to the MySQL Server

Activity List for Part A

Step 1: Go to https://www.apachefriends.org/index.html and download XAMPP for your OS.


Apache Friends
Download
Hosting
Community
About
Search..
Search
EN
XAMPP Apache + MariaDB + PHP + Perl
XAMPP
Download
Click here for other versions
XAMPP for Windows 8.2.4 (PHP 8.2.4)
XAMPP for Linux 8.2.4 (PHP 8.2.4)
XAMPP for OS X 8.2.4 (PHP 8.2.4)

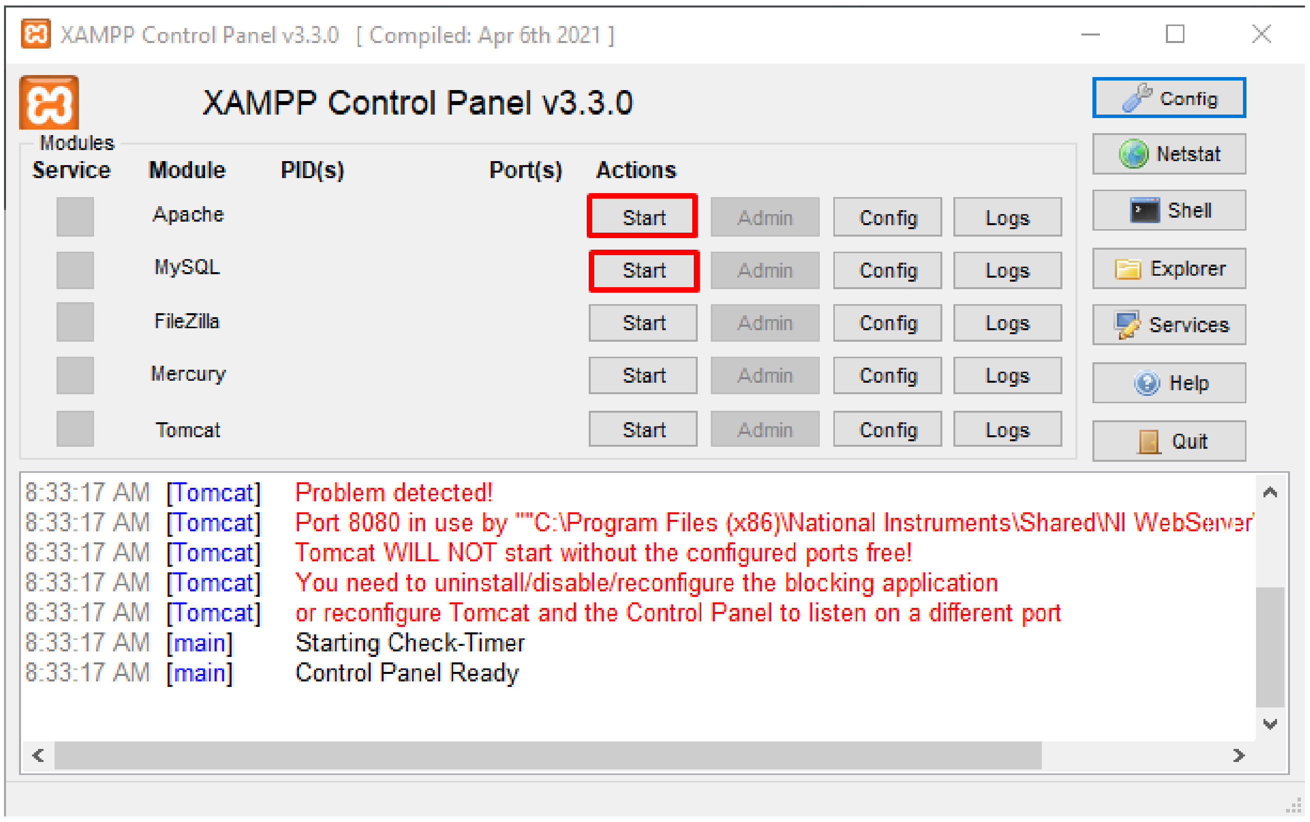
What is XAMPP?

XAMPP is the most popular PHP development environment

XAMPP is a completely free, easy to install Apache distribution containing MariaDB, PHP, and Perl. The XAMPP open source package has been set up to be incredibly easy to install and to use.

Step 2: Install XAMPP according to the installation guide.

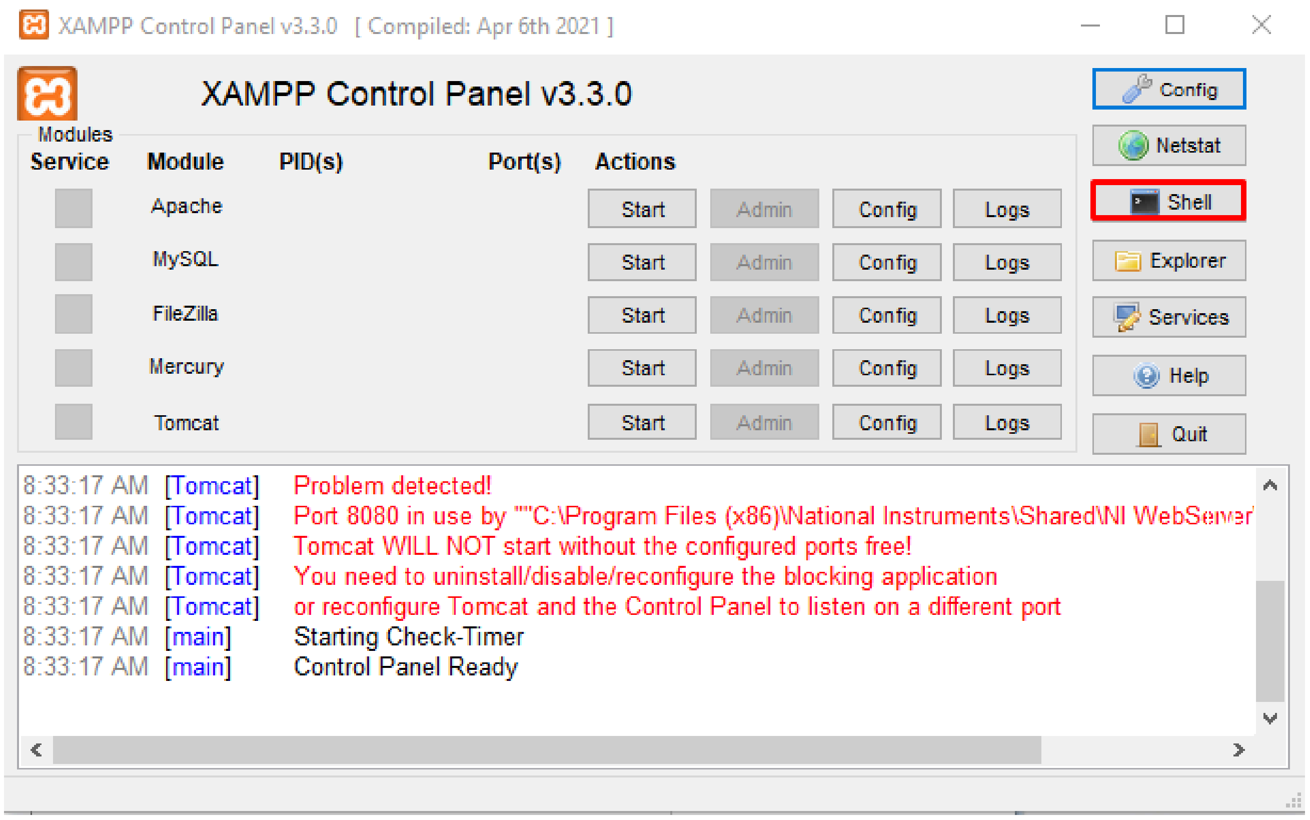
Step 3: Open the XAMPP control panel after installation.



```
8:33:17 AM  [Tomcat]   Problem detected!
8:33:17 AM  [Tomcat]   Port 8080 in use by ""C:\Program Files (x86)\National Instruments\Shared\NI WebServer
8:33:17 AM  [Tomcat]   Tomcat WILL NOT start without the configured ports free!
8:33:17 AM  [Tomcat]   You need to uninstall/disable/reconfigure the blocking application
8:33:17 AM  [Tomcat]   or reconfigure Tomcat and the Control Panel to listen on a different port
8:33:17 AM  [main]     Starting Check-Timer
8:33:17 AM  [main]     Control Panel Ready
```

Open the control panel and click the start buttons (highlighted in red) beside Apache and MySQL.

Step 4: Click on the "shell" button on the right of the window



```
8:33:17 AM  [Tomcat]   Problem detected!
8:33:17 AM  [Tomcat]   Port 8080 in use by ""C:\Program Files (x86)\National Instruments\Shared\NI WebServer
8:33:17 AM  [Tomcat]   Tomcat WILL NOT start without the configured ports free!
8:33:17 AM  [Tomcat]   You need to uninstall/disable/reconfigure the blocking application
8:33:17 AM  [Tomcat]   or reconfigure Tomcat and the Control Panel to listen on a different port
8:33:17 AM  [main]     Starting Check-Timer
8:33:17 AM  [main]     Control Panel Ready
```

Step 5: Connect to the MySQL server

After clicking on the shell, you should see a black window. Type in the following command:

```
mysql -u root -p
```

When you are asked for a password, don't type anything just press enter. **The default password for xampp is an empty string.**

Part B : An Introduction to MySQL Queries

Syntax error in a query might cause the mysql> prompt not to appear after executing the query.

Solutions:

i. Typing one of the following may solve the problem

1. ');
2. `);
3. `;
4. ';
5. Or log out with ctrl+c and log in again

Activity List for Part B

- **All commands are shown in the red boxes.**
- **In the green box, write the response you see after entering each query. Also, write the query for cases where you had to make changes.**
- **The part of the query in bold italic are variables, the rest are keywords. Sometimes, you might need to change the variables as per requirement.**
- **All new queries should be typed in the command window after mysql>**

A Server can have multiple databases, for example, a movie database and a car rental database. So how can you view the list of all databases?

SHOW DATABASES;

If you want to start a new project you should create your own database. After creating check if the new database is in the list now.

CREATE DATABASE ***DB_Name*** ;

Before storing or manipulating any data, you HAVE to select the database you want to work on. All new command will take effect in selected database.

USE ***DB_Name*** ;

All data are stored in tables. Each table will represent 1 entity, for example students_info, the column of the table will be attributes of the students(e.g. student_id, name, department, cgpa, grad_date) and each row will have information about 1 single student. Each attribute has a pre-defined data type such as int, char etc.

```
CREATE TABLE Lab_Grades
(
    std_id char(4),
    name varchar(30),
    major char(3),
    section char(1),
    days_present int,
    project_marks double,
    cgpa decimal(3,2),
    submission_date date
);
```

You can have many tables in database, e.g student_info, teacher_info, course_info etc. So how to view the list of all tables?

```
SHOW TABLES;
```

You might want to check the structure of a table e.g. what columns are there, what are the data types etc.

```
DESCRIBE Table_Name;
```

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16
S008	Farhana	CSE	2	6	15	2.67	2018-08-16
s009	Naima	CSE	2	12	20	3.7	2018-08-14

Link for Table Data: https://docs.google.com/document/d/1YYP8YpRP2gEvWFoCkp3rpkZKdR-CEjmunhR_3-9s18Q/

Now you want to insert the data above in the table you created. There are two commands: a long version and a shorter one! Insert all the data above in the table.

```
INSERT INTO Table_Name
(std_id,name,major, section,
days_present,project_marks,cgpa,
submission_date) values
('s001','Abir','CS','1',10, 18.5,
3.91,'2018-09-15');
```

```
INSERT INTO Table_Name values
('s001','Abir','CS','1', 10, 18.5,
3.91,'2018-09-15');
```

So now you want to view all the data you inserted? For that we will use the select query. More on that later!

SELECT * FROM *Table_Name;*

Part C : SQL Alter, Update, Delete & Basic Select Queries

Task 1: Modifying Columns of a Table:

Add column project_title in the table

ALTER TABLE ***Lab_Grades*** add ***project_title*** char(10);

The data type for Project_title should be varchar(50)

ALTER TABLE ***Lab_Grades*** MODIFY COLUMN ***project_title*** varchar(50);

Now let's delete the column Project_title

ALTER TABLE ***Lab_Grades*** DROP COLUMN ***project_title***;

- How will you change the name of a column from submission_date to sub_date? **[Google it!]**

Task 2: Updating Wrong Data:

Oops! Arafat's major is actually CSE, so update the value in the table

UPDATE ***Lab_Grades*** SET **major** = 'CSE' WHERE **name** = 'Arafat';

Nahid's name is misspelled and also his project marks should be updated to 16.

UPDATE ***Lab_Grades*** SET ***name***='Naheed', ***project_marks*** =16 where ***std_id*** = 's004' ;

- What will happen if the where clause is not included in the update query, e.g . if you typed Update Lab_Grades set Major = 'CSE';? **[Don't try it now, just write the answer]**

Task 3: Deleting Data:

Naima dropped out of the course. So, delete her data from the table.

DELETE FROM ***Lab_Grades*** WHERE ***Name***= 'Naima';

- What would have happened if there was another student named Naima?

Delete the data of everyone who was less than 8 days present.

```
DELETE FROM Lab_Grades WHERE days_present < 8;
```

Task 4: Deleting Table or Database [DO NOT TRY NOW]:

So now if you want to delete a table or database you need the following commands

```
DROP TABLE Table_Name;
```

```
DROP DATABASE DB_Name;
```

Task 5: Retrieving Data from Table:

- What is the [**select * from Lab_grades;]** command used for?

Let's say you want to retrieve only the student id, name and project marks.

```
SELECT  std_id, name, project_marks FROM Lab_Grades;
```

Retrieve the name and total marks of students out of 25 (project + attendance)

```
SELECT name, project_marks+days_present*5/12 AS
total_marks FROM Lab_Grades;
```

- The "as" keyword in the above query is known as an alias. Check out what happens if you remove the "as Total_marks" portion from the above command. State the difference below.

- Try the command below, and state what the Upper() and Lower() functions mean.

```
SELECT UPPER(name), LOWER(name) from Lab_Grades;
```

- Try the two commands below. What is the difference and why is the distinct keyword used?

```
SELECT major FROM Lab_Grades;
```

```
SELECT DISTINCT major FROM Lab_Grades;
```

Now you want to view all the details sorted by name. You can use the order by keyword

```
SELECT * FROM Lab_Grades ORDER BY name;
```

- Was it sorted in ascending or descending order? How can you sort in the opposite order?[Hint: check next command]

Sort all details according to name and then by submission date. There are two students named Tasneem, observe what happens.

```
SELECT * FROM Lab_Grades ORDER BY name DESC,
submission_date ASC;
```

Now, you want to view the name and project marks for only CSE students.

```
SELECT name,project_marks FROM Lab_Grades WHERE
major='CSE' ;
```

- Retrieve the names, days present and marks of students whose project marks are greater than 17

Retrieve the name and marks of students whose marks is between 17 and 19

```
SELECT name,project_marks FROM Lab_Grades WHERE
project_marks BETWEEN 17 and 19 ;
```

Retrieve the details of students who are majoring in either CS or CSE

```
SELECT * FROM Lab_Grades WHERE major in ('CSE', 'CS');
```

- What is the “in” keyword in the above query? In the where clause, you can write the same command using the “or” and “=” operators. Try to figure it out!

Retrieve the details of the students who submitted their project in August and whose marks is greater than 18

```
SELECT * FROM Lab_Grades WHERE project_marks>18 and
submission_date BETWEEN '2018-08-01' and '2018-08-31';
```

- How can you find the students whose Submission_date is not in August?

Retrieve the details of students whose name start with 'a'

SELECT * FROM ***Lab_Grades*** WHERE ***name*** like 'a%';

Retrieve the details of students whose name contains at least 2 a's

SELECT * FROM ***Lab_Grades*** WHERE ***name*** LIKE '%a%a%';

- Try the following command and explain what happens : Select * from ***Lab_Grades*** where ***Name*** like 'a___'; ***[There are 3 underscores]***

<u>Task 6: Basic Select Quiz</u>

Go to https://sqlzoo.net/wiki/SELECT_Quiz and answer the Quiz to test your knowledge of basic select queries.

Department of Computer Science and Engineering

Course Code: CSE 370	**Credits: 3.0**
Course Name: Database Systems	**Semester: Fall 2025**

Lab Assignment 1

Soon after joining Google's elite dev team, Area 120, you were assigned to a project using MySQL since it was your specialty. The project was a social media platform specialized in allowing developers from all over the globe to connect to each other, and it would have features similar to Facebook. For your first task, you have been assigned to work on the tables of one of the project databases.

You will use a database called **Google_<Your8DigitStudentID>.**

Format: CREATE DATABASE Google_<Your8DigitStudentID>; **CREATE DATABASE** Google_12345678;
USE Google_12345678;

The table name is **"Developers"** which is shown below.

member_id	name	email	influence_count	Joining_date	multiplier
1	Taylor Otwell	otwell@laravel.com	739360	2020-6-10	10
2	Ryan Dahl	ryan@nodejs.org	633632	2020-04-22	10
3	Brendan Eich	eich@javascript.com	939570	2020-05-07	8
4	Evan You	you@vuejs.org	982630	2020-06-11	7

5	Rasmus Lerdorf	lerdorf@php.net	937927	2020-06-3	8
6	Guido van Rossum	guido@python.org	968827	2020-07-18	19
7	Adrian Holovaty	adrian@djangoproject.com	570724	2020-05-07	5
8	Simon Willison	simon@djangoproject.com	864615	2020-04-30	4
9	James Gosling	james@java.com	719491	2020-05-18	5
10	Rod Johnson	rod@spring.io	601744	2020-05-18	7
11	Satoshi Nakamoto	nakamoto@blockchain.com	630488	2020-05-10	10

Write the queries of the tasks given below [6 * 2 = 12].

1. Create the above table with the appropriate data type for each column.

2. Change the column name "influence_count". The new name should be "followers," and the data type should be integer.

3. Update the number of followers of each developer by +10.

4. There is a formula to find the efficiency of the developers. Efficiency = ((followers*100/1000000) * (multipliers*100/20))/100. Show the efficiency of each developer in a column named "Efficiency" along with their name.

5. Show the name of the developers in UpperCase and the descending order of their Joining_date.

6. Retrieve the member_id, name, email and followers of the developers who have either ".com" or ".net" in their email address.

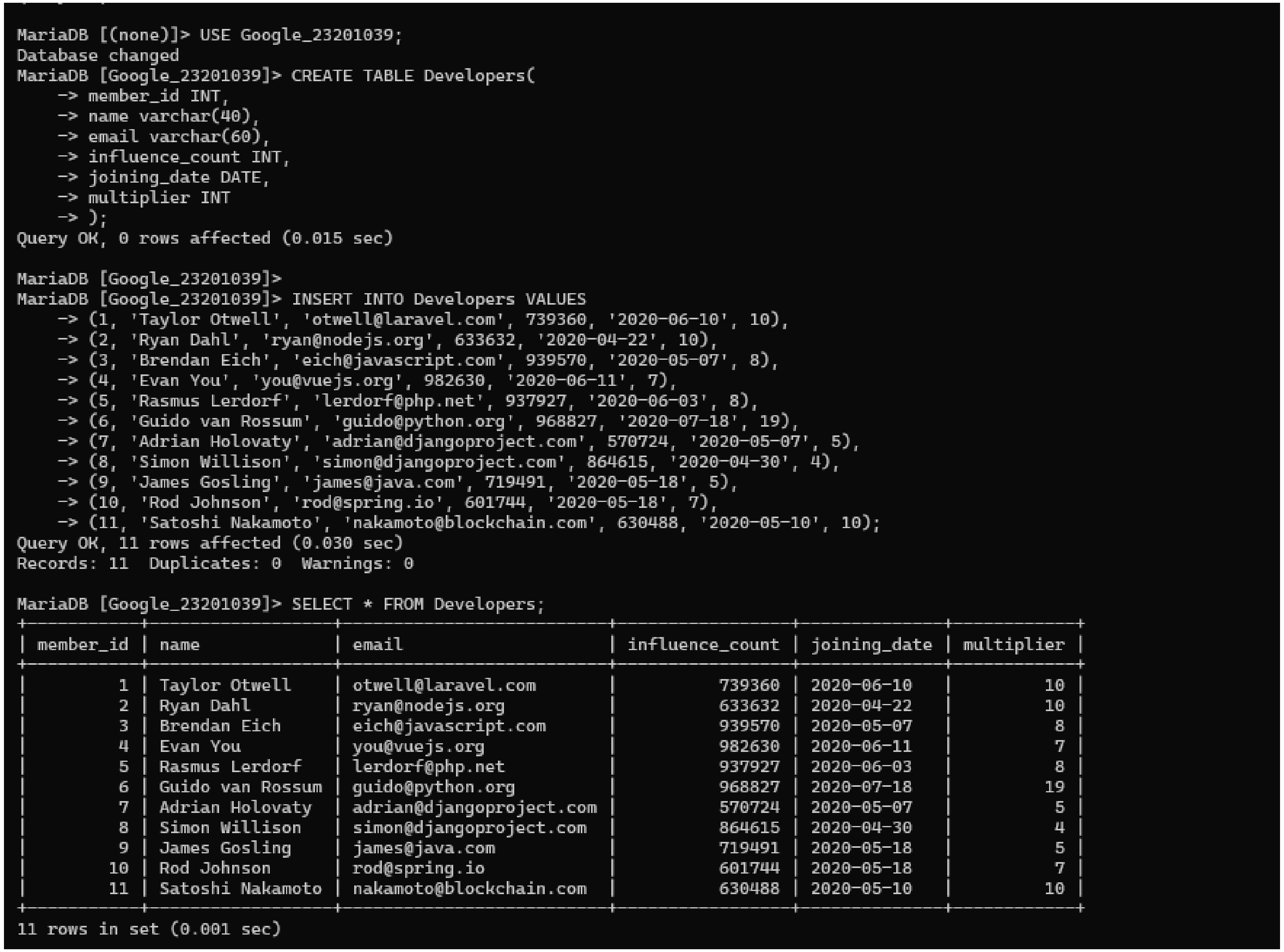
CSE370 : Database Systems

Assignment 01 | Fall 2025

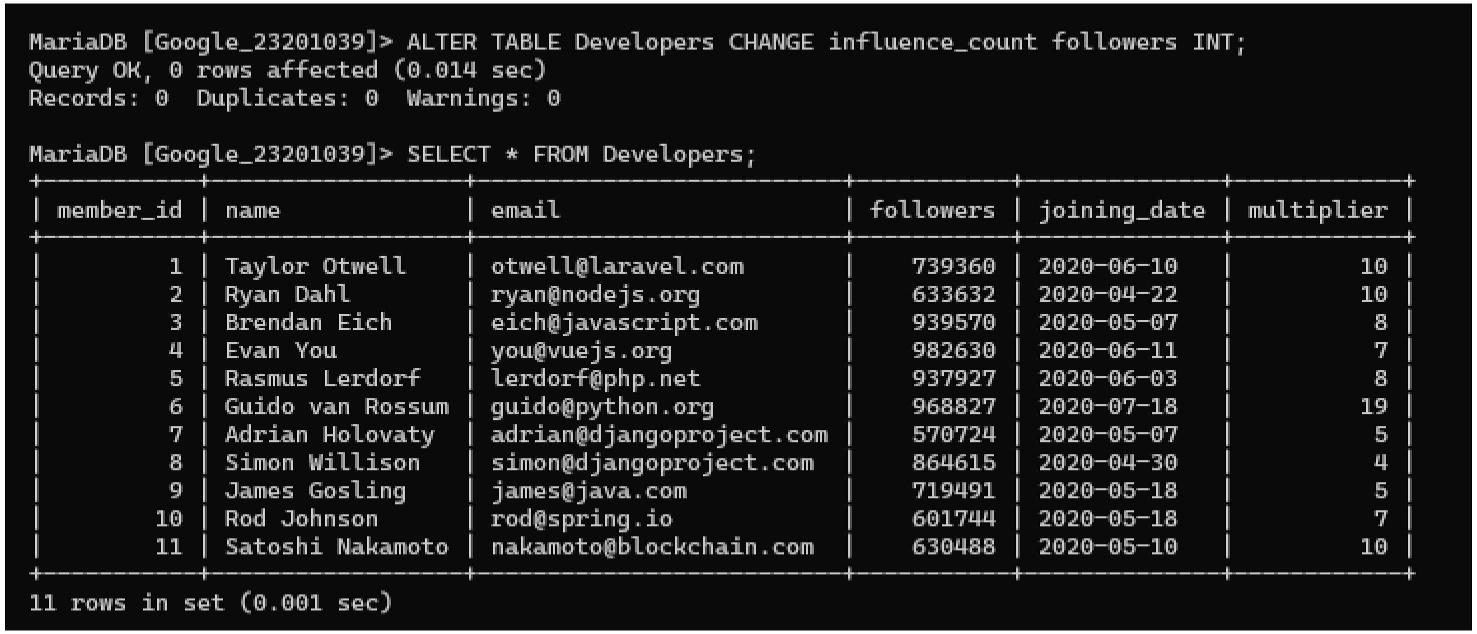
ID : 23201039 | Name : Priyanti Biswas

No 1 Query (as Plain Text)	CREATE TABLE Developers(member_id INT, name varchar(40), email varchar(60), influence_count INT, joining_date DATE, multiplier INT); INSERT INTO Developers VALUES (1, 'Taylor Otwell', 'otwell@laravel.com', 739360, '2020-06-10', 10), (2, 'Ryan Dahl', 'ryan@nodejs.org', 633632, '2020-04-22', 10), (3, 'Brendan Eich', 'eich@javascript.com', 939570, '2020-05-07', 8), (4, 'Evan You', 'you@vuejs.org', 982630, '2020-06-11', 7), (5, 'Rasmus Lerdorf', 'lerdorf@php.net', 937927, '2020-06-03', 8), (6, 'Guido van Rossum', 'guido@python.org', 968827, '2020-07-18', 19), (7, 'Adrian Holovaty', 'adrian@djangoproject.com', 570724, '2020-05-07', 5), (8, 'Simon Willison', 'simon@djangoproject.com', 864615, '2020-04-30', 4), (9, 'James Gosling', 'james@java.com', 719491, '2020-05-18', 5), (10, 'Rod Johnson', 'rod@spring.io', 601744, '2020-05-18', 7), (11, 'Satoshi Nakamoto', 'nakamoto@blockchain.com', 630488, '2020-05-10', 10);

No 1 SS
(of Query & Output in Shell)



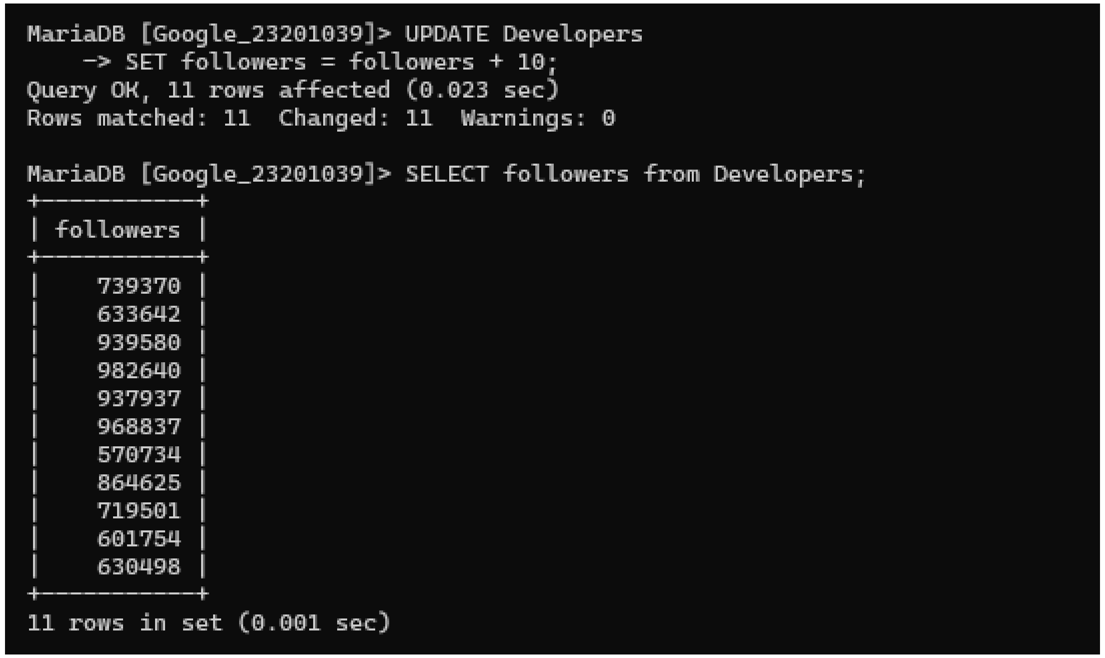
```
MariaDB [(none)]> USE Google_23201039;
Database changed
MariaDB [Google_23201039]> CREATE TABLE Developers(
    -> member_id INT,
    -> name varchar(40),
    -> email varchar(60),
    -> influence_count INT,
    -> joining_date DATE,
    -> multiplier INT
    -> );
Query OK, 0 rows affected (0.015 sec)

MariaDB [Google_23201039]>
MariaDB [Google_23201039]> INSERT INTO Developers VALUES
    -> (1, 'Taylor Otwell', 'otwell@laravel.com', 739360, '2020-06-10', 10),
    -> (2, 'Ryan Dahl', 'ryan@nodejs.org', 633632, '2020-04-22', 10),
    -> (3, 'Brendan Eich', 'eich@javascript.com', 939570, '2020-05-07', 8),
    -> (4, 'Evan You', 'you@vuejs.org', 982630, '2020-06-11', 7),
    -> (5, 'Rasmus Lerdorf', 'lerdorf@php.net', 937927, '2020-06-03', 8),
    -> (6, 'Guido van Rossum', 'guido@python.org', 968827, '2020-07-18', 19),
    -> (7, 'Adrian Holovaty', 'adrian@djangoproject.com', 570724, '2020-05-07', 5),
    -> (8, 'Simon Willison', 'simon@djangoproject.com', 864615, '2020-04-30', 4),
    -> (9, 'James Gosling', 'james@java.com', 719491, '2020-05-18', 5),
    -> (10, 'Rod Johnson', 'rod@spring.io', 601744, '2020-05-18', 7),
    -> (11, 'Satoshi Nakamoto', 'nakamoto@blockchain.com', 630488, '2020-05-10', 10);
Query OK, 11 rows affected (0.030 sec)
Records: 11  Duplicates: 0  Warnings: 0

MariaDB [Google_23201039]> SELECT * FROM Developers;
+-----------+------------------+--------------------------+-----------------+--------------+------------+
| member_id | name             | email                    | influence_count | joining_date | multiplier |
+-----------+------------------+--------------------------+-----------------+--------------+------------+
|         1 | Taylor Otwell    | otwell@laravel.com       |          739360 | 2020-06-10   |         10 |
|         2 | Ryan Dahl        | ryan@nodejs.org          |          633632 | 2020-04-22   |         10 |
|         3 | Brendan Eich     | eich@javascript.com      |          939570 | 2020-05-07   |          8 |
|         4 | Evan You         | you@vuejs.org            |          982630 | 2020-06-11   |          7 |
|         5 | Rasmus Lerdorf   | lerdorf@php.net          |          937927 | 2020-06-03   |          8 |
|         6 | Guido van Rossum | guido@python.org         |          968827 | 2020-07-18   |         19 |
|         7 | Adrian Holovaty  | adrian@djangoproject.com |          570724 | 2020-05-07   |          5 |
|         8 | Simon Willison   | simon@djangoproject.com  |          864615 | 2020-04-30   |          4 |
|         9 | James Gosling    | james@java.com           |          719491 | 2020-05-18   |          5 |
|        10 | Rod Johnson      | rod@spring.io            |          601744 | 2020-05-18   |          7 |
|        11 | Satoshi Nakamoto | nakamoto@blockchain.com  |          630488 | 2020-05-10   |         10 |
+-----------+------------------+--------------------------+-----------------+--------------+------------+
11 rows in set (0.001 sec)
```

No 2 Query (as Plain Text)	ALTER TABLE Developers CHANGE influence_count followers INT;
No 2 SS (of Query & Output in Shell)	

```
MariaDB [Google_23201039]> ALTER TABLE Developers CHANGE influence_count followers INT;
Query OK, 0 rows affected (0.014 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [Google_23201039]> SELECT * FROM Developers;
+-----------+------------------+---------------------------+-----------+--------------+------------+
| member_id | name             | email                     | followers | joining_date | multiplier |
+-----------+------------------+---------------------------+-----------+--------------+------------+
|         1 | Taylor Otwell    | otwell@laravel.com        |    739360 | 2020-06-10   |         10 |
|         2 | Ryan Dahl        | ryan@nodejs.org           |    633632 | 2020-04-22   |         10 |
|         3 | Brendan Eich     | eich@javascript.com       |    939570 | 2020-05-07   |          8 |
|         4 | Evan You         | you@vuejs.org             |    982630 | 2020-06-11   |          7 |
|         5 | Rasmus Lerdorf   | lerdorf@php.net           |    937927 | 2020-06-03   |          8 |
|         6 | Guido van Rossum | guido@python.org          |    968827 | 2020-07-18   |         19 |
|         7 | Adrian Holovaty  | adrian@djangoproject.com  |    570724 | 2020-05-07   |          5 |
|         8 | Simon Willison   | simon@djangoproject.com   |    864615 | 2020-04-30   |          4 |
|         9 | James Gosling    | james@java.com            |    719491 | 2020-05-18   |          5 |
|        10 | Rod Johnson      | rod@spring.io             |    601744 | 2020-05-18   |          7 |
|        11 | Satoshi Nakamoto | nakamoto@blockchain.com   |    630488 | 2020-05-10   |         10 |
+-----------+------------------+---------------------------+-----------+--------------+------------+
11 rows in set (0.001 sec)
```

No 3 Query (as Plain Text)	UPDATE Developers
	SET followers = followers + 10;

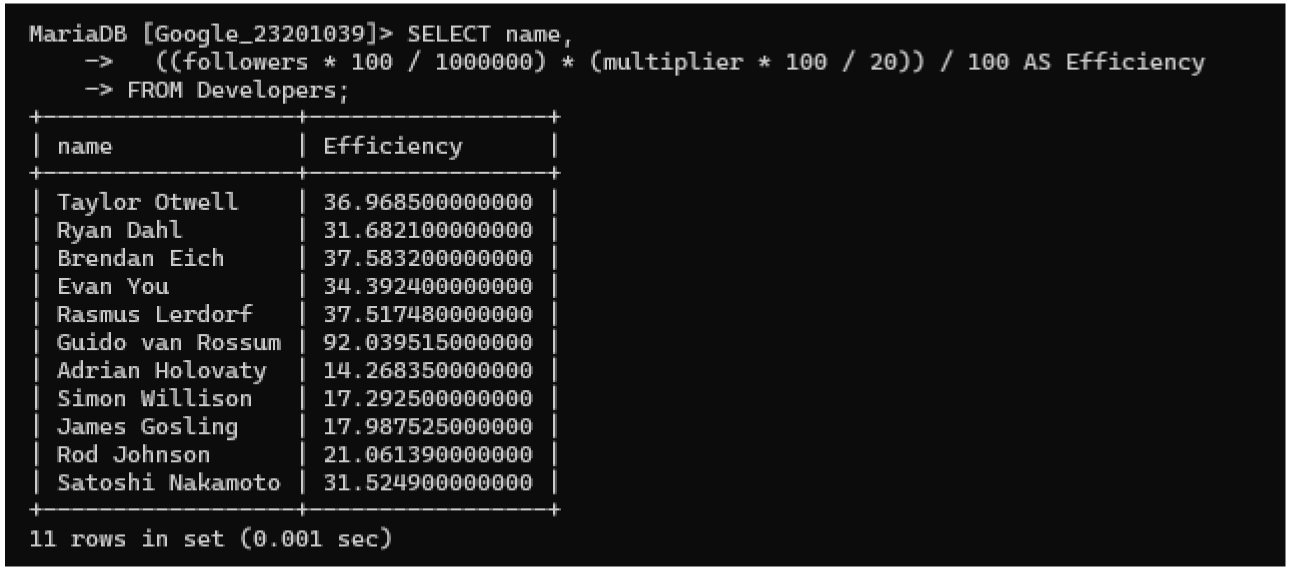
No 3 SS
(of Query & Output
in Shell)



```
MariaDB [Google_23201039]> UPDATE Developers
    -> SET followers = followers + 10;
Query OK, 11 rows affected (0.023 sec)
Rows matched: 11  Changed: 11  Warnings: 0

MariaDB [Google_23201039]> SELECT followers from Developers;
+-----------+
| followers |
+-----------+
|    739370 |
|    633642 |
|    939580 |
|    982640 |
|    937937 |
|    968837 |
|    570734 |
|    864625 |
|    719501 |
|    601754 |
|    630498 |
+-----------+
11 rows in set (0.001 sec)
```

No 4 Query
(as Plain Text)

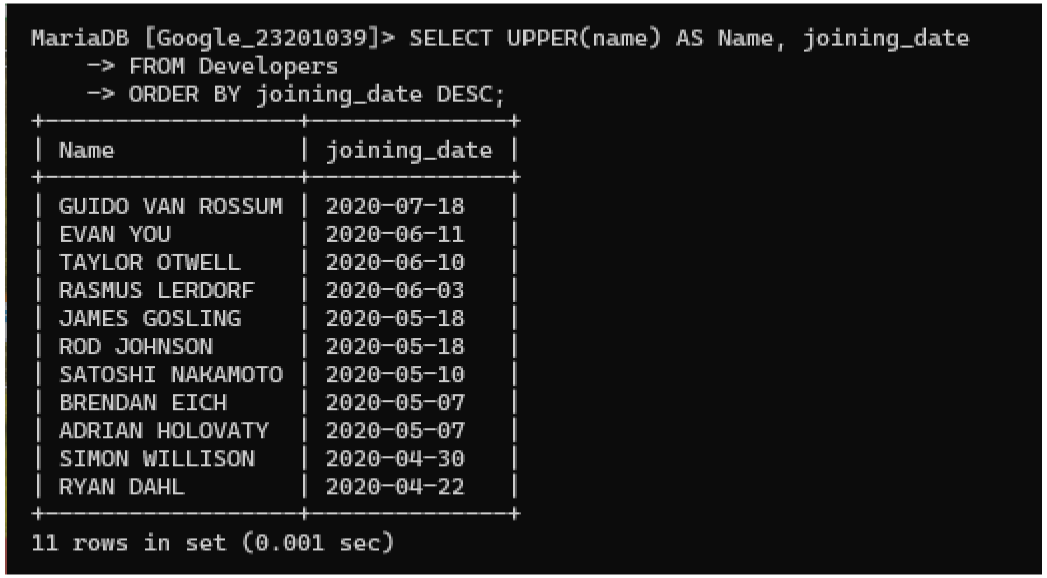
```
SELECT name,
  ((followers * 100 / 1000000) * (multiplier * 100 / 20)) / 100 AS Efficiency
FROM Developers;
```

No 4 SS (of Query & Output in Shell)	

No 5 Query (as Plain Text)	SELECT UPPER(name) AS Name, joining_date
	FROM Developers
	ORDER BY joining_date DESC;

```
MariaDB [Google_23201039]> SELECT name,
    ->   ((followers * 100 / 1000000) * (multiplier * 100 / 20)) / 100 AS Efficiency
    -> FROM Developers;
+-------------------+-----------------+
| name              | Efficiency      |
+-------------------+-----------------+
| Taylor Otwell     | 36.968500000000 |
| Ryan Dahl         | 31.682100000000 |
| Brendan Eich      | 37.583200000000 |
| Evan You          | 34.392400000000 |
| Rasmus Lerdorf    | 37.517480000000 |
| Guido van Rossum  | 92.039515000000 |
| Adrian Holovaty   | 14.268350000000 |
| Simon Willison    | 17.292500000000 |
| James Gosling     | 17.987525000000 |
| Rod Johnson       | 21.061390000000 |
| Satoshi Nakamoto  | 31.524900000000 |
+-------------------+-----------------+
11 rows in set (0.001 sec)
```

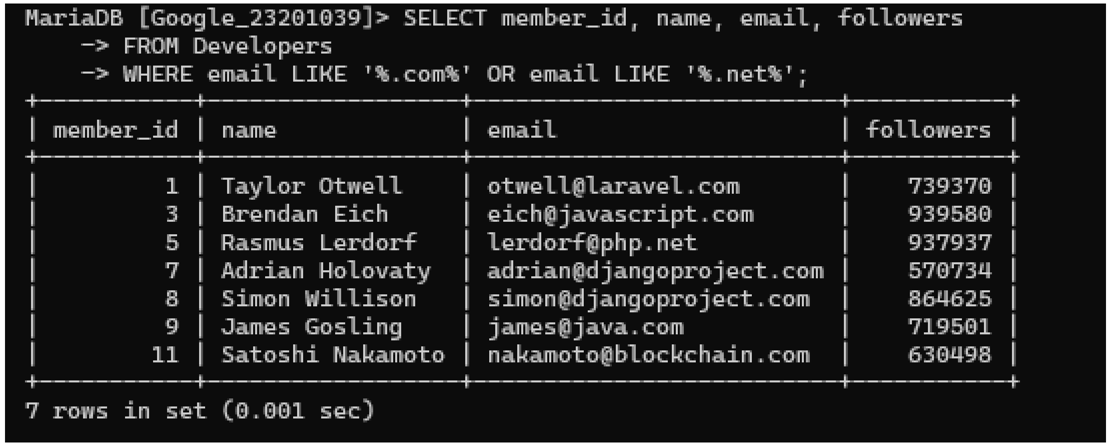
No 5 SS (of Query & Output in Shell)



```
MariaDB [Google_23201039]> SELECT UPPER(name) AS Name, joining_date
    -> FROM Developers
    -> ORDER BY joining_date DESC;
+-------------------+--------------+
| Name              | joining_date |
+-------------------+--------------+
| GUIDO VAN ROSSUM  | 2020-07-18   |
| EVAN YOU          | 2020-06-11   |
| TAYLOR OTWELL     | 2020-06-10   |
| RASMUS LERDORF    | 2020-06-03   |
| JAMES GOSLING     | 2020-05-18   |
| ROD JOHNSON       | 2020-05-18   |
| SATOSHI NAKAMOTO  | 2020-05-10   |
| BRENDAN EICH      | 2020-05-07   |
| ADRIAN HOLOVATY   | 2020-05-07   |
| SIMON WILLISON    | 2020-04-30   |
| RYAN DAHL         | 2020-04-22   |
+-------------------+--------------+
11 rows in set (0.001 sec)
```

No 6 Query (as Plain Text)

SELECT member_id, name, email, followers
FROM Developers
WHERE email LIKE '%.com%' OR email LIKE '%.net%';

No 6 SS (of Query & Output in Shell)	

```
MariaDB [Google_23201039]> SELECT member_id, name, email, followers
    -> FROM Developers
    -> WHERE email LIKE '%.com%' OR email LIKE '%.net%';
+-----------+------------------+--------------------------+-----------+
| member_id | name             | email                    | followers |
+-----------+------------------+--------------------------+-----------+
|         1 | Taylor Otwell    | otwell@laravel.com       |    739370 |
|         3 | Brendan Eich     | eich@javascript.com      |    939580 |
|         5 | Rasmus Lerdorf   | lerdorf@php.net          |    937937 |
|         7 | Adrian Holovaty  | adrian@djangoproject.com |    570734 |
|         8 | Simon Willison   | simon@djangoproject.com  |    864625 |
|         9 | James Gosling    | james@java.com           |    719501 |
|        11 | Satoshi Nakamoto | nakamoto@blockchain.com  |    630498 |
+-----------+------------------+--------------------------+-----------+
7 rows in set (0.001 sec)
```

Department of Computer Science and Engineering

Course Code: CSE 370	**Credits: 3.0**
Course Name: Database Systems	**Semester: Fall 2025**

Lab 02: SQL Subqueries & Aggregate Functions

Activity List

- **All commands are shown in the red boxes.**
- **In the green box, write the appropriate query/answer.**
- **All new queries should be typed in the command window after mysql>**
- **Start by connecting to the server using: mysql -u root -p [password: <just press enter>]**
- **For more MySQL queries, go to www.w3schools.com/sql or google it!**

Initial Table: It's a bit different than Lab 01!

std_id	name	major	section	days_present	project_marks	cgpa	submission_date
s001	Abir	CS	1	10	18.5	3.91	2018-09-15
s002	Nafis	CSE	1	12	20	3.86	2018-08-15
s003	Tasneem	CS	1	8	18	3.57	2018-09-18
s004	Nahid	ECE	2	7	16.5	3.25	2018-08-20
s005	Arafat	CS	2	11	20	4.0	2018-09-13
s006	Tasneem	CSE	1	12	17.5	3.7	2018-08-15
s007	Muhtadi	ECE	1	10	19	3.67	2018-09-16

Link for Table Data: https://docs.google.com/document/d/1j6zmKf3cUr7zQKNxByfRPh0_AaTUn3hTyW6_AKuCFQA/

The purpose of the SELECT statement is to retrieve and display data from one or more database tables. It is an extremely powerful command. SELECT is the most frequently used SQL command and has the following general form:

SELECT [DISTINCT | ALL] {* | [columnExpression [AS newName]] [, . . .]}

FROM TableName [alias] [, . . .]

[WHERE condition]

[GROUP BY columnList] [HAVING condition]

[ORDER BY columnList]

columnExpression represents a column name or an expression, TableName is the name of an existing database table or view that you have access to, and alias is an optional abbreviation for TableName.

The sequence of processing in a SELECT statement is:

	FROM specifies the table or tables to be used
	WHERE filters the rows subject to some condition
↓	**GROUP BY** forms groups of rows with the same column value
	HAVING filters the groups subject to some condition
	SELECT specifies which columns are to appear in the output
	ORDER BY specifies the order of the output

The order of the clauses in the SELECT statement cannot be changed. The only two mandatory clauses are the first two: SELECT and FROM; the remainder are optional. The SELECT operation is closed: the result of a query on a table is another table.

<u>Task 1: Aggregate Functions, Group By and Having:</u>

Retrieve the minimum CGPA/Project_marks from the table	SELECT MIN(*cgpa*) FROM ***Lab_Grades***;
Retrieve the total number of students and the average projects marks	SELECT COUNT(*) as ***total_students***, AVG(***project_marks***) as ***average_project_marks*** FROM ***Lab_Grades;***
Find the sum of the number of days present.	SELECT SUM(***days_present***) FROM ***Lab_Grades;***

- How will you retrieve the last submission date?

Find Minimum and Maximum CGPA of each major	SELECT ***major, MIN***(***CGPA***) as ***minCGPA,*** MAX(***CGPA)*** as ***maxCGPA*** FROM ***Lab_Grades*** GROUP BY ***major;***
Retrieve total number of students for each major	SELECT ***major,*** COUNT(****)*** FROM ***Lab_Grades*** GROUP BY ***major;***

- What is the purpose of the group by keyword? In the above command, if we group by sub_date, instead of major, what will be the output?

For each major find the minimum and maximum CGPA/Project_marks, but only if there were at least 2 students in the major

```
SELECT major, MIN(cgpa) as minCGPA, MAX(cgpa) as maxCGPA
FROM Lab_Grades GROUP BY major HAVING COUNT(*)>=2;
```

For each major find the minimum and maximum CGPA/Project_marks, but consider only students who submitted before or on 15th sep

```
SELECT major, MIN(cgpa) as minCGPA, max(cgpa) as  maxCGPA FROM Lab_Grades
WHERE submission_date<='2018-09-15' GROUP BY major;
```

- The having and where clauses are both used to specify a condition when selecting rows. What is the difference between them?

<u>Task 2: Sub Queries/Nested Queries, Any and All:</u>

- Think about how you can retrieve the names of students who got the highest project marks. Try out your query. Did you get the "correct" response according to the table?

Now, try the nested/sub query on the right

```
SELECT name  FROM Lab_Grades
WHERE project_marks=(SELECT MAX(project_marks)
                        FROM Lab_Grades);
```

Retrieve the CSE students whose CGPA/Project_marks is higher than at least 1 CS students

```
SELECT * from Lab_Grades WHERE major = 'CSE' and cgpa>ANY
(SELECT cgpa FROM Lab_Grades WHERE major = 'CS');
```

Retrieve the CSE students whose CGPA/Project_marks is higher than all CS students

```
SELECT * FROM Lab_Grades WHERE major = 'CSE' and cgpa>ALL
(SELECT cgpa FROM Lab_Grades WHERE major = 'CS');
```

- Did you understand the role of "any" and "all" in the above queries? Explain below.

- Retrieve the names of the students who have received marks greater than at least 1 student doing the same major as them.[Hint: see next command]

Task 3: Correlated Subqueries and Exists:

Select those majors for which at least 1 student has CGPA lower than 3.7

```
SELECT DISTINCT major FROM Lab_Grades L1 WHERE EXISTS (SELECT * FROM Lab_Grades L2 WHERE L2.major=L1.major and L2.cgpa<3.7);
```

- L1 and L2 are temporary aliases and create two separate instances for Lab_Grades; why are they required?

Retrieve the name of student who has obtained maximum marks in project using exists

```
SELECT name FROM Lab_Grades L1 WHERE NOT EXISTS (SELECT * FROM Lab_Grades L2 WHERE L2.std_id!=L1.std_id AND L2.project_marks>L1.project_marks);
```

Retrieve the name of student who has obtained maximum marks in project and who is unique using exists

```
SELECT name FROM Lab_Grades L1 WHERE NOT EXISTS (SELECT * FROM Lab_Grades L2 where L2.std_id!=L1.std_id AND L2.project_marks>=L1.project_marks);
```

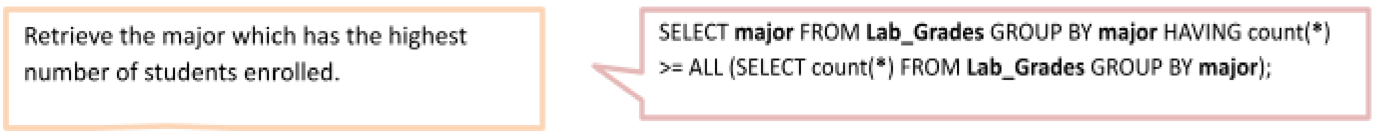
- Please identify the difference between the above two queries. [Hint: 1 asks for unique-only 1 student got the highest and the other didn't]

Retrieve the total number of students who obtained the maximum marks. There are many ways of solving one task; a few ways for this one are shown below.

```
SELECT COUNT(*) FROM Lab_Grades L1 WHERE NOT EXISTS (SELECT * FROM Lab_Grades L2 WHERE L2.std_id!=L1.std_id and L2.project_marks>L1.project_marks);
```

```
SELECT COUNT(*) FROM Lab_Grades WHERE project_marks = (SELECT MAX(project_marks) FROM Lab_Grades);
```

```
SELECT COUNT(*) FROM Lab_Grades WHERE project_marks >=ALL (SELECT project_marks FROM Lab_Grades);
```



Retrieve the major which has the highest number of students enrolled.

```
SELECT major FROM Lab_Grades GROUP BY major HAVING count(*)
>= ALL (SELECT count(*) FROM Lab_Grades GROUP BY major);
```

Task 4: Take a Quiz

Go to https://sqlzoo.net/wiki/Nested_SELECT_Quiz to test your understanding of the queries taught in class.

Department of Computer Science and Engineering

Course Code: CSE 370	**Credits: 3.0**
Course Name: Database Systems	**Semester: Fall 2025**

Lab Assignment 2

Proving yourself worthy of being able to handle more significant tasks, the tech lead has decided to give you a challenging job. However, this time, the data you would be handling is very sensitive and no one wants this data to be leaked. Therefore, instead of getting the entire table, the tech lead has given you the list of attributes that the table contains and the table name. The information given is as follows:

You will use a database called **Company_<Your8DigitStudentID>.**

Format: CREATE DATABASE Company_<Your8DigitStudentID>; **CREATE DATABASE** Company_12345678;
USE Company_12345678;

Table Name: ***Employee***	
Attribute Name	**Attribute type**
employee_id	char(10)
first_name	varchar(20)
last_name	varchar(20)
email	varchar(60)
phone_number	char(14)
hire_date	date
job_id	char(10)
salary	int
commission_pct	decimal(5,3)
manager_id	char(10)
department_id	char(10)

You need to insert rows (around 10 or more to test all of the queries for the following tasks) as per the data types of the attributes/columns and problem statements of the following tasks.

Employee IDs should be ‘EMP001’, ‘EMP002’, etc.
Job IDs should be ‘JOB001’, ‘JOB002’, etc.
Manager IDs should be ‘MNG001’, ‘MNG002’, etc.
Department IDs should be DPT001, ..., DPT005, ... DPT007, etc.

Write down the queries to retrieve the following information: [7 X 2 =14]

1. Find the **first_name, last_name, email, phone_number, hire_date** and **department_id** of all the employees with the latest **hire_date**.

2. Find the ***first_name***, ***last_name***, ***employee_id***, ***phone_number, salary*** and ***department_id*** of all the employees with the lowest **salary** in each department.

3. Find the ***first_name****,* ***last_name****,* ***employee_id****,* ***commission_pct*** *and* ***department_id*** of all the employees in the department 'DPT007' who have a lower commission_pct than all of the department 'DPT005' employees.

4. Find the **department_id** and total number of employees of each department which does not have a single employee under it with a **salary** more than 30,000.

5. For each department, find the ***department_id***, ***job_id*** and ***commission_pct*** with ***commission_pct*** less than at least one other ***job_id*** in that department.

6. Find the ***manager_id*** who does not have any employee under them with a ***salary*** less than 3500.

7. Find the ***first_name***, ***last_name***, ***employee_id***, ***email, salary, department_id*** *and* ***commission_pct*** of the employee with the lowest ***commission_pct*** under each manager.

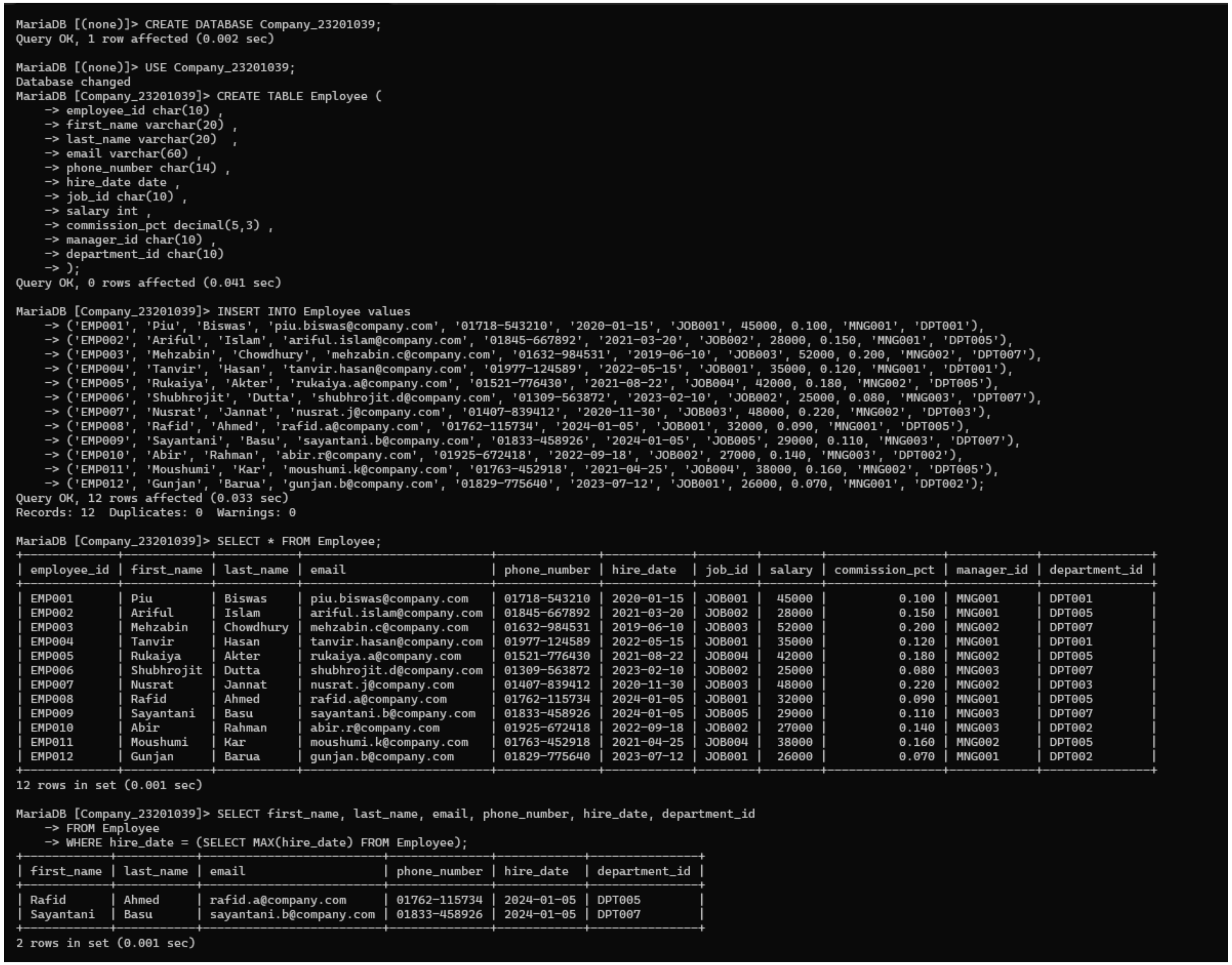
CSE370 : Database Systems

Assignment 02 | Fall 2025

ID : 23201039 | Name : Priyanti Biswas

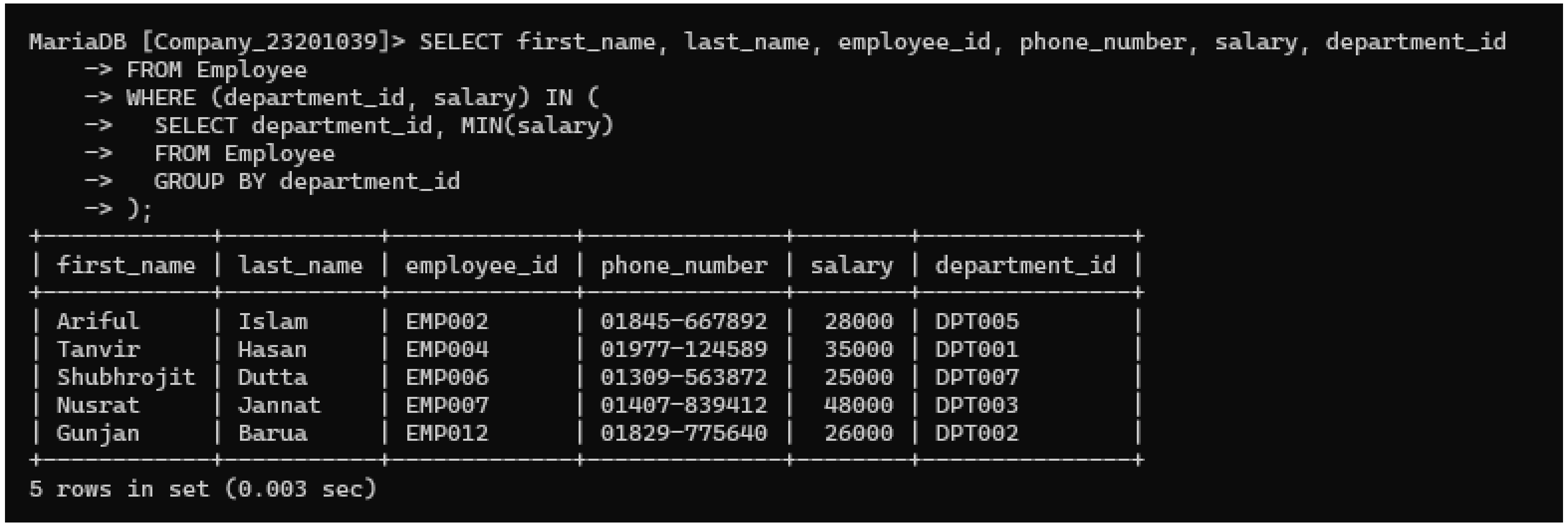
No 1 Query (as Plain Text)	SELECT first_name, last_name, email, phone_number, hire_date, department_id
	FROM Employee
	WHERE hire_date = (SELECT MAX(hire_date) FROM Employee);

No 1 SS
(of Query & Output in Shell)



```
MariaDB [(none)]> CREATE DATABASE Company_23201039;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> USE Company_23201039;
Database changed
MariaDB [Company_23201039]> CREATE TABLE Employee (
    -> employee_id char(10) ,
    -> first_name varchar(20) ,
    -> last_name varchar(20)  ,
    -> email varchar(60) ,
    -> phone_number char(14) ,
    -> hire_date date ,
    -> job_id char(10) ,
    -> salary int ,
    -> commission_pct decimal(5,3) ,
    -> manager_id char(10) ,
    -> department_id char(10)
    -> );
Query OK, 0 rows affected (0.041 sec)

MariaDB [Company_23201039]> INSERT INTO Employee values
    -> ('EMP001', 'Piu', 'Biswas', 'piu.biswas@company.com', '01718-543210', '2020-01-15', 'JOB001', 45000, 0.100, 'MNG001', 'DPT001'),
    -> ('EMP002', 'Ariful', 'Islam', 'ariful.islam@company.com', '01845-667892', '2021-03-20', 'JOB002', 28000, 0.150, 'MNG001', 'DPT005'),
    -> ('EMP003', 'Mehzabin', 'Chowdhury', 'mehzabin.c@company.com', '01632-984531', '2019-06-10', 'JOB003', 52000, 0.200, 'MNG002', 'DPT007'),
    -> ('EMP004', 'Tanvir', 'Hasan', 'tanvir.hasan@company.com', '01977-124589', '2022-05-15', 'JOB001', 35000, 0.120, 'MNG001', 'DPT001'),
    -> ('EMP005', 'Rukaiya', 'Akter', 'rukaiya.a@company.com', '01521-776430', '2021-08-22', 'JOB004', 42000, 0.180, 'MNG002', 'DPT005'),
    -> ('EMP006', 'Shubhrojit', 'Dutta', 'shubhrojit.d@company.com', '01309-563872', '2023-02-10', 'JOB002', 25000, 0.080, 'MNG003', 'DPT007'),
    -> ('EMP007', 'Nusrat', 'Jannat', 'nusrat.j@company.com', '01407-839412', '2020-11-30', 'JOB003', 48000, 0.220, 'MNG002', 'DPT003'),
    -> ('EMP008', 'Rafid', 'Ahmed', 'rafid.a@company.com', '01762-115734', '2024-01-05', 'JOB001', 32000, 0.090, 'MNG001', 'DPT005'),
    -> ('EMP009', 'Sayantani', 'Basu', 'sayantani.b@company.com', '01833-458926', '2024-01-05', 'JOB005', 29000, 0.110, 'MNG003', 'DPT007'),
    -> ('EMP010', 'Abir', 'Rahman', 'abir.r@company.com', '01925-672418', '2022-09-18', 'JOB002', 27000, 0.140, 'MNG003', 'DPT002'),
    -> ('EMP011', 'Moushumi', 'Kar', 'moushumi.k@company.com', '01763-452918', '2021-04-25', 'JOB004', 38000, 0.160, 'MNG002', 'DPT005'),
    -> ('EMP012', 'Gunjan', 'Barua', 'gunjan.b@company.com', '01829-775640', '2023-07-12', 'JOB001', 26000, 0.070, 'MNG001', 'DPT002');
Query OK, 12 rows affected (0.033 sec)
Records: 12  Duplicates: 0  Warnings: 0

MariaDB [Company_23201039]> SELECT * FROM Employee;
+-------------+------------+-----------+---------------------------+--------------+------------+--------+--------+----------------+------------+---------------+
| employee_id | first_name | last_name | email                     | phone_number | hire_date  | job_id | salary | commission_pct | manager_id | department_id |
+-------------+------------+-----------+---------------------------+--------------+------------+--------+--------+----------------+------------+---------------+
| EMP001      | Piu        | Biswas    | piu.biswas@company.com    | 01718-543210 | 2020-01-15 | JOB001 |  45000 |          0.100 | MNG001     | DPT001        |
| EMP002      | Ariful     | Islam     | ariful.islam@company.com  | 01845-667892 | 2021-03-20 | JOB002 |  28000 |          0.150 | MNG001     | DPT005        |
| EMP003      | Mehzabin   | Chowdhury | mehzabin.c@company.com    | 01632-984531 | 2019-06-10 | JOB003 |  52000 |          0.200 | MNG002     | DPT007        |
| EMP004      | Tanvir     | Hasan     | tanvir.hasan@company.com  | 01977-124589 | 2022-05-15 | JOB001 |  35000 |          0.120 | MNG001     | DPT001        |
| EMP005      | Rukaiya    | Akter     | rukaiya.a@company.com     | 01521-776430 | 2021-08-22 | JOB004 |  42000 |          0.180 | MNG002     | DPT005        |
| EMP006      | Shubhrojit | Dutta     | shubhrojit.d@company.com  | 01309-563872 | 2023-02-10 | JOB002 |  25000 |          0.080 | MNG003     | DPT007        |
| EMP007      | Nusrat     | Jannat    | nusrat.j@company.com      | 01407-839412 | 2020-11-30 | JOB003 |  48000 |          0.220 | MNG002     | DPT003        |
| EMP008      | Rafid      | Ahmed     | rafid.a@company.com       | 01762-115734 | 2024-01-05 | JOB001 |  32000 |          0.090 | MNG001     | DPT005        |
| EMP009      | Sayantani  | Basu      | sayantani.b@company.com   | 01833-458926 | 2024-01-05 | JOB005 |  29000 |          0.110 | MNG003     | DPT007        |
| EMP010      | Abir       | Rahman    | abir.r@company.com        | 01925-672418 | 2022-09-18 | JOB002 |  27000 |          0.140 | MNG003     | DPT002        |
| EMP011      | Moushumi   | Kar       | moushumi.k@company.com    | 01763-452918 | 2021-04-25 | JOB004 |  38000 |          0.160 | MNG002     | DPT005        |
| EMP012      | Gunjan     | Barua     | gunjan.b@company.com      | 01829-775640 | 2023-07-12 | JOB001 |  26000 |          0.070 | MNG001     | DPT002        |
+-------------+------------+-----------+---------------------------+--------------+------------+--------+--------+----------------+------------+---------------+
12 rows in set (0.001 sec)

MariaDB [Company_23201039]> SELECT first_name, last_name, email, phone_number, hire_date, department_id
    -> FROM Employee
    -> WHERE hire_date = (SELECT MAX(hire_date) FROM Employee);
+------------+-----------+-------------------------+--------------+------------+---------------+
| first_name | last_name | email                   | phone_number | hire_date  | department_id |
+------------+-----------+-------------------------+--------------+------------+---------------+
| Rafid      | Ahmed     | rafid.a@company.com     | 01762-115734 | 2024-01-05 | DPT005        |
| Sayantani  | Basu      | sayantani.b@company.com | 01833-458926 | 2024-01-05 | DPT007        |
+------------+-----------+-------------------------+--------------+------------+---------------+
2 rows in set (0.001 sec)
```

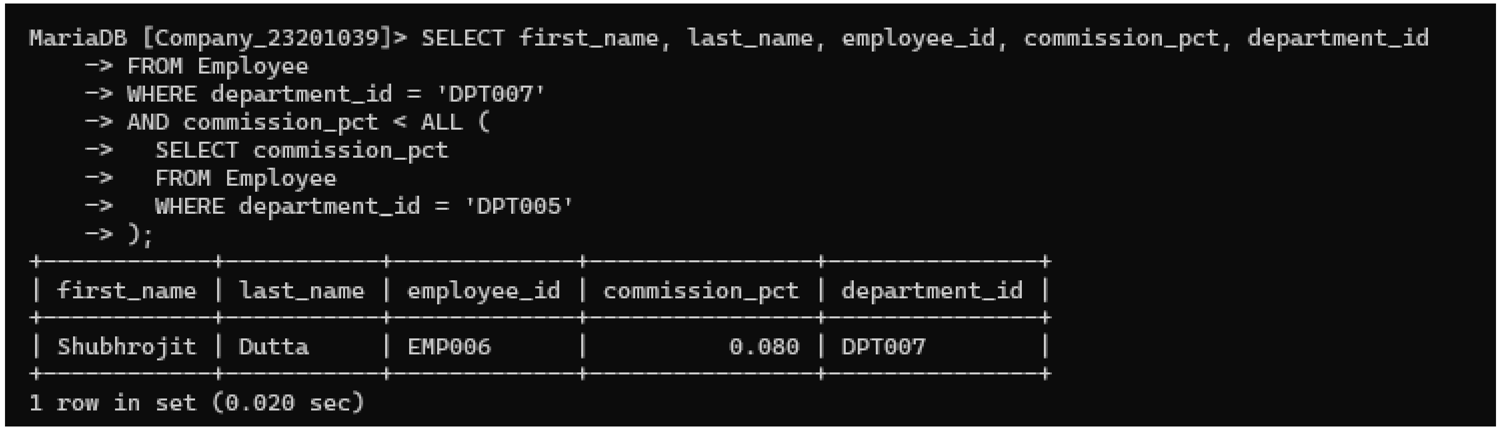
**No 2 Query
(as Plain Text)**
	SELECT first_name, last_name, employee_id, phone_number, salary, department_id
	FROM Employee
	WHERE (department_id, salary) IN (
	  SELECT department_id, MIN(salary)
	  FROM Employee
	  GROUP BY department_id
	);

No 2 SS (of Query & Output in Shell)	

**No 3 Query
(as Plain Text)**
	SELECT first_name, last_name, employee_id, commission_pct, department_id
	FROM Employee
	WHERE department_id = 'DPT007'
	AND commission_pct < ALL (
	  SELECT commission_pct

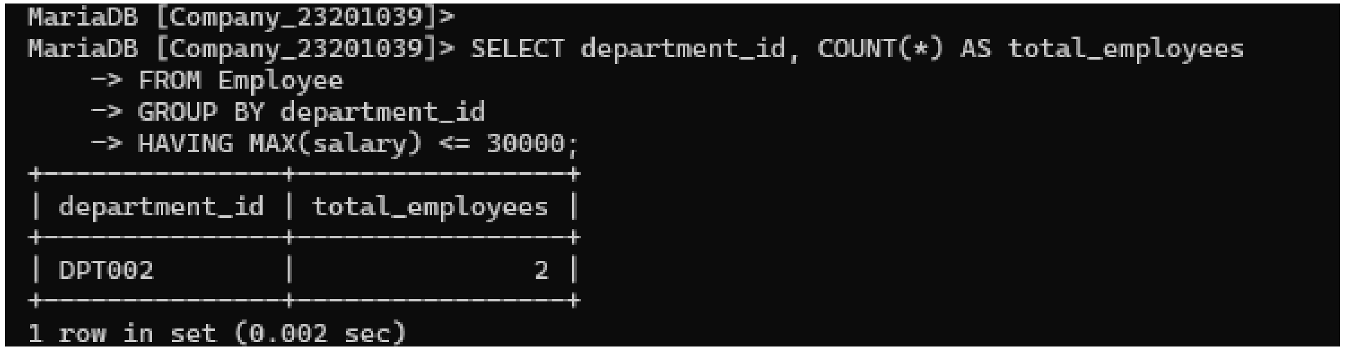
```
MariaDB [Company_23201039]> SELECT first_name, last_name, employee_id, phone_number, salary, department_id
    -> FROM Employee
    -> WHERE (department_id, salary) IN (
    ->   SELECT department_id, MIN(salary)
    ->   FROM Employee
    ->   GROUP BY department_id
    -> );
+------------+-----------+-------------+--------------+--------+---------------+
| first_name | last_name | employee_id | phone_number | salary | department_id |
+------------+-----------+-------------+--------------+--------+---------------+
| Ariful     | Islam     | EMP002      | 01845-667892 |  28000 | DPT005        |
| Tanvir     | Hasan     | EMP004      | 01977-124589 |  35000 | DPT001        |
| Shubhrojit | Dutta     | EMP006      | 01309-563872 |  25000 | DPT007        |
| Nusrat     | Jannat    | EMP007      | 01407-839412 |  48000 | DPT003        |
| Gunjan     | Barua     | EMP012      | 01829-775640 |  26000 | DPT002        |
+------------+-----------+-------------+--------------+--------+---------------+
5 rows in set (0.003 sec)
```

	FROM Employee
	WHERE department_id = 'DPT005'
	);

No 3 SS (of Query & Output in Shell)	

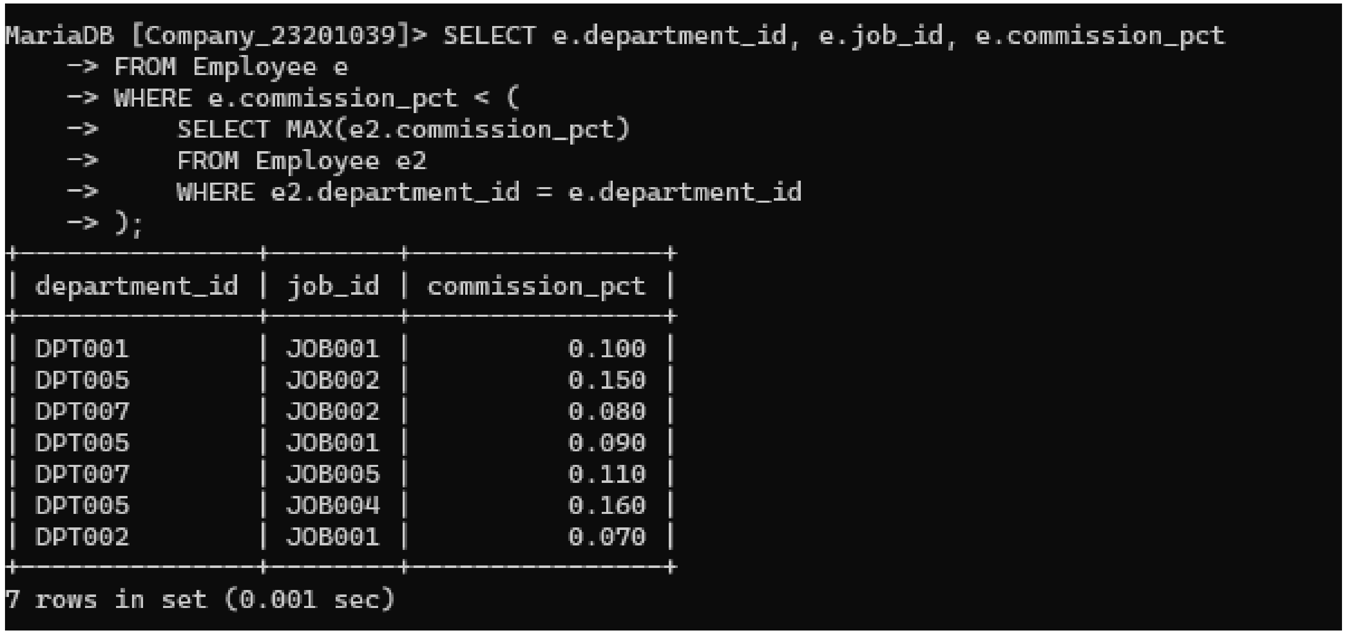
```
MariaDB [Company_23201039]> SELECT first_name, last_name, employee_id, commission_pct, department_id
    -> FROM Employee
    -> WHERE department_id = 'DPT007'
    -> AND commission_pct < ALL (
    ->   SELECT commission_pct
    ->   FROM Employee
    ->   WHERE department_id = 'DPT005'
    -> );
+------------+-----------+-------------+----------------+---------------+
| first_name | last_name | employee_id | commission_pct | department_id |
+------------+-----------+-------------+----------------+---------------+
| Shubhrojit | Dutta     | EMP006      |          0.080 | DPT007        |
+------------+-----------+-------------+----------------+---------------+
1 row in set (0.020 sec)
```

**No 4 Query
(as Plain Text)**
	SELECT department_id, COUNT(*) AS total_employees
	FROM Employee
	GROUP BY department_id
	HAVING MAX(salary) <= 30000;

No 4 SS (of Query & Output in Shell)	

No 5 Query (as Plain Text)	SELECT e.department_id, e.job_id, e.commission_pct
	FROM Employee e
	WHERE e.commission_pct < (
	   SELECT MAX(e2.commission_pct)
	   FROM Employee e2
	   WHERE e2.department_id = e.department_id
	);

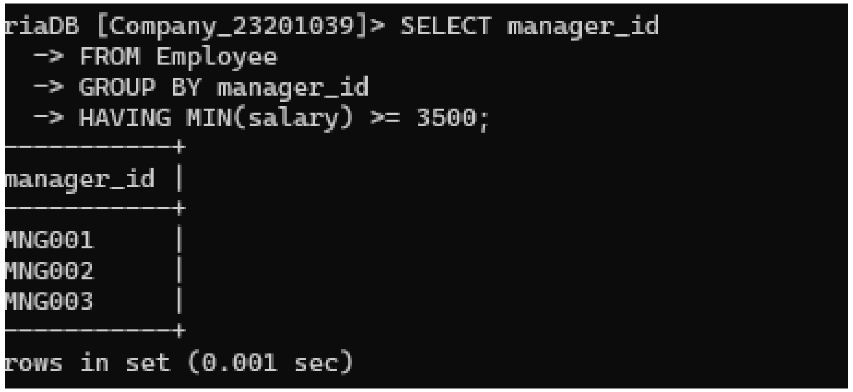
```
MariaDB [Company_23201039]>
MariaDB [Company_23201039]> SELECT department_id, COUNT(*) AS total_employees
    -> FROM Employee
    -> GROUP BY department_id
    -> HAVING MAX(salary) <= 30000;
+---------------+-----------------+
| department_id | total_employees |
+---------------+-----------------+
| DPT002        |               2 |
+---------------+-----------------+
1 row in set (0.002 sec)
```

No 5 SS (of Query & Output in Shell)	

No 6 Query (as Plain Text)	SELECT manager_id
	FROM Employee
	GROUP BY manager_id
	HAVING MIN(salary) >= 3500;

```
MariaDB [Company_23201039]> SELECT e.department_id, e.job_id, e.commission_pct
    -> FROM Employee e
    -> WHERE e.commission_pct < (
    ->     SELECT MAX(e2.commission_pct)
    ->     FROM Employee e2
    ->     WHERE e2.department_id = e.department_id
    -> );
+---------------+--------+----------------+
| department_id | job_id | commission_pct |
+---------------+--------+----------------+
| DPT001        | JOB001 |          0.100 |
| DPT005        | JOB002 |          0.150 |
| DPT007        | JOB002 |          0.080 |
| DPT005        | JOB001 |          0.090 |
| DPT007        | JOB005 |          0.110 |
| DPT005        | JOB004 |          0.160 |
| DPT002        | JOB001 |          0.070 |
+---------------+--------+----------------+
7 rows in set (0.001 sec)
```

No 6 SS
(of Query & Output in Shell)

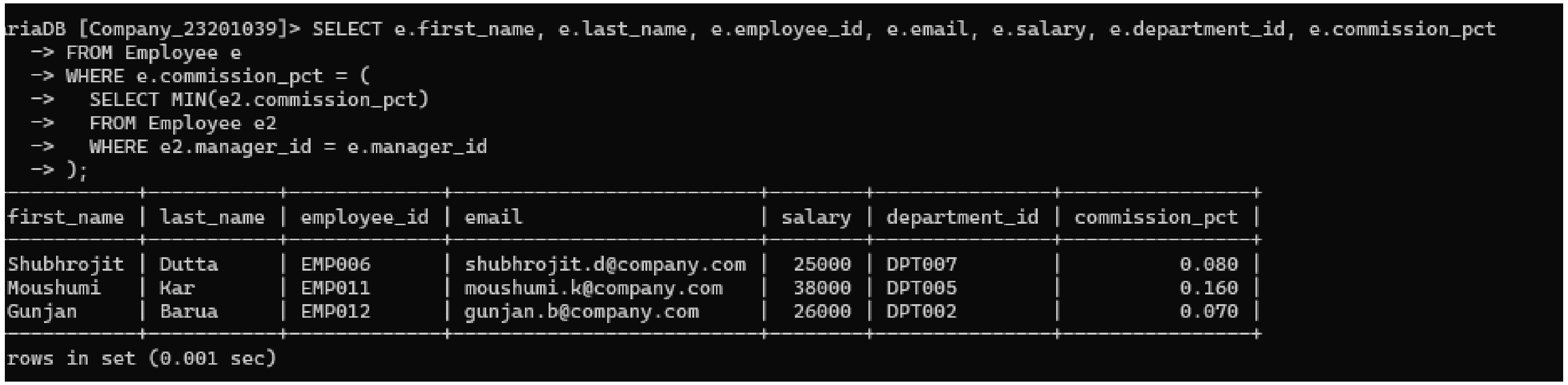


```
riaDB [Company_23201039]> SELECT manager_id
  -> FROM Employee
  -> GROUP BY manager_id
  -> HAVING MIN(salary) >= 3500;
-----------+
manager_id |
-----------+
MNG001     |
MNG002     |
MNG003     |
-----------+
rows in set (0.001 sec)
```

No 7 Query
(as Plain Text)

```
SELECT e.first_name, e.last_name, e.employee_id, e.email, e.salary, e.department_id, e.commission_pct
FROM Employee e
WHERE e.commission_pct = (
  SELECT MIN(e2.commission_pct)
  FROM Employee e2
  WHERE e2.manager_id = e.manager_id
);
```

No 7 SS
(of Query & Output in Shell)



```
riaDB [Company_23201039]> SELECT e.first_name, e.last_name, e.employee_id, e.email, e.salary, e.department_id, e.commission_pct
  -> FROM Employee e
  -> WHERE e.commission_pct = (
  ->   SELECT MIN(e2.commission_pct)
  ->   FROM Employee e2
  ->   WHERE e2.manager_id = e.manager_id
  -> );
-----------+-----------+-------------+--------------------------+--------+---------------+----------------+
first_name | last_name | employee_id | email                    | salary | department_id | commission_pct |
-----------+-----------+-------------+--------------------------+--------+---------------+----------------+
Shubhrojit | Dutta     | EMP006      | shubhrojit.d@company.com |  25000 | DPT007        |          0.080 |
Moushumi   | Kar       | EMP011      | moushumi.k@company.com   |  38000 | DPT005        |          0.160 |
Gunjan     | Barua     | EMP012      | gunjan.b@company.com     |  26000 | DPT002        |          0.070 |
-----------+-----------+-------------+--------------------------+--------+---------------+----------------+
rows in set (0.001 sec)
```

Department of Computer Science and Engineering

Course Code: CSE 370	**Credits: 3.0**
Course Name: Database Systems	**Semester: Fall 2025**

Lab 03 : Introduction to Bank DB, SQL Joins and Constraints

Activity List

Suggestions for this Lab:

- Use a **Text editor** such as Notepad to type and save your program.
- **Copy** and **Paste** your program from the Text editor to the command line. If the program works, save the program. Otherwise, fix the error and save it.
- Save your text file regularly.

Tas Create the "Bank" database and then create all necessary tables below

```
CREATE DATABASE Bank;
```

```
USE Bank;
```

```
create table customer (
customer_id varchar(10) not null,
customer_name varchar(20) not null,
customer_street varchar(30),
customer_city varchar(30),
primary key (customer_id));
```

```
create table branch (
branch_name varchar(15),
branch_city varchar(30),
assets int,
primary key (branch_name),
check (assets >= 0));
```

```
create table account (
branch_name varchar(15),
account_number varchar(10) not null,
balance int,
primary key (account_number),
check (balance >= 0));
```

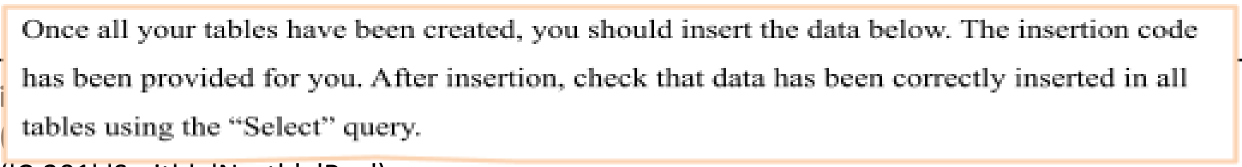
```
create table loan (
loan_number varchar(10) not null,
```

```sql
branch_name varchar(15),
amount int,
primary key (loan_number));
```

```sql
create table depositor (
customer_id varchar(10) not null,
account_number varchar(10) not null,
primary key (customer_id,account_number),
foreign key (customer_id) references customer(customer_id),
foreign key (account_number) references account(account_number));
```

```sql
create table borrower (
customer_id varchar(10) not null,
loan_number varchar(10) not null,
primary key (customer_id, loan_number),
foreign key (customer_id) references customer(customer_id),
foreign key (loan_number) references loan(loan_number));
```

Task 2



Once all your tables have been created, you should insert the data below. The insertion code has been provided for you. After insertion, check that data has been correctly inserted in all tables using the "Select" query.

```sql
('C-201','Smith', 'North', 'Rye'),
('C-211','Hayes', 'Main', 'Harrison'),
('C-212','Curry', 'North', 'Rye'),
('C-215','Lindsay', 'Park', 'Pittsfield'),
('C-220','Turner', 'Putnam', 'Stamford'),
('C-222','Williams', 'Nassau', 'Princeton'),
('C-225','Adams', 'Spring', 'Pittsfield'),
('C-226','Johnson', 'Alma', 'Palo Alto'),
('C-233','Glenn', 'Sand Hill', 'Woodside'),
('C-234','Brooks', 'Senator', 'Brooklyn'),
('C-255','Green', 'Walnut', 'Stamford');
```

```sql
insert into branch values
('Downtown', 'Brooklyn',9000000),
('Redwood', 'Palo Alto',2100000),
('Perryridge', 'Horseneck',1700000),
('Mianus', 'Horseneck',400000),
('Round Hill', 'Horseneck',8000000),
('Pownal', 'Bennington',300000),
('North Town', 'Rye',3700000),
('Brighton', 'Brooklyn',7100000);
```

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

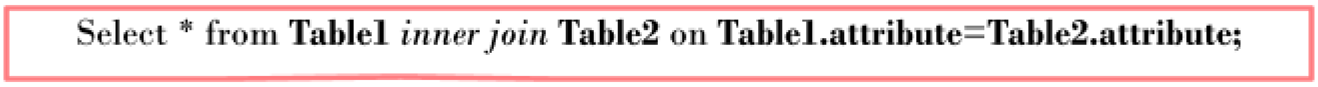
```
insert into loan values
('L-17', 'Downtown', 1000),
('L-23', 'Redwood', 2000),
('L-15', 'Perryridge', 1500),
('L-14', 'Downtown', 1500),
('L-93', 'Mianus', 500),
('L-11', 'Round Hill', 900),
('L-16', 'Perryridge', 1300);
```

```
insert into depositor values
('C-226', 'A-101'),
('C-201', 'A-215'),
('C-211', 'A-102'),
('C-220', 'A-305'),
('C-226', 'A-201'),
('C-101', 'A-217'),
('C-215', 'A-222');
```

```
insert into borrower values
('C-101', 'L-17'),
('C-201', 'L-23'),
('C-211', 'L-15'),
('C-226', 'L-14'),
('C-212', 'L-93'),
('C-201', 'L-11'),
('C-222', 'L-17'),
('C-225', 'L-16');
```

Task 3

The command below is a general format for joining two tables. You can replace “Inner Join” with any of the three other Join operations.



```
Select * from Table1 inner join Table2 on Table1.attribute=Table2.attribute;
```

1. Retrieve all customer’s id, name, city and account number using
   a. Inner Join
   b. Left Join
   c. Right Join

d. Full Join [Not supported by MySQL]

ıo

Task 4

You can join more than two tables using the following format:

```
Select * from ((Table1 inner join Table2 on Table1.attribute=Table2.attribute)
    inner join Table 3 on Table3.attribute = Table1/2.attribute);
```

Retrieve the following information from your database using "join": Customer name, city, account number, balance and branch name.

Task 5

Inner join can also be accomplished without using the "join" keyword in the following way:

```
Select * from T1, T2, T3,.....Tn where T1.attr=T2.attr [.....other conditions] ;
```

Apply the above format on Task 4 and compare your results.

Task 6

Solve all tasks below. Some tasks require multiple tables for which you may use joins as shown in "Task 3 and 4" or without using the join keyword as shown in "Task 5". After joining your required tables, according to your need you may use any other clauses(or keywords) learnt in previous labs, such as, where, group by, having, order by. Some tasks may not need multiple tables at all.

1. Find names and cities of customers who have a loan at Perryridge branch
2. Find the accounts with balances between 700 and 900.
3. Find the names of customers on streets with names ending in "Hill".
4. Find the names of branches whose assets are greater than the assets of some branch in Brooklyn.
5. Find the set of names of branches whose assets are greater than the assets of all branches in Horseneck.
6. Find the set of names of customers at Brighton branch, in alphabetical order.
7. Show the loan data, ordered by decreasing amounts, then increasing loan numbers.
8. Find the names of branches having at least one account, with average balances greater than or equal 700.
9. Find the names and account number of customers who have the 3 highest balances in their accounts. [Hint: https://www.w3schools.com/sql/sql_top.asp]

Task 7

Solve the following tasks:

1. Find the names of customers with accounts at a branch where Johnson has an account.
2. Find the names of customers with an account but not a loan at Mianus branch.
3. Find the names of each branch and the number of customers having at least one account at that branch.
4. Find the average balance of all customers in ‘Palo Alto’ having at least 2 accounts
5. Find the name and account number of the customer who has the 3rd highest balance in their account.

Department of Computer Science and Engineering

Course Code: CSE 370	**Credits: 3.0**
Course Name: Database Systems	**Semester: Fall 2025**

Lab Assignment 3

You will use a database called **Bank_<Your8DigitStudentID>.**

Data for the **Bank_<Your8DigitStudentID>** database is available here :
https://docs.google.com/document/d/1eAxCSgxgBsF1a0Gtwgq9hTGj7mtRtYRjwoHzf_fgwow/

Using the bank database, write MySQL queries for the following tasks:

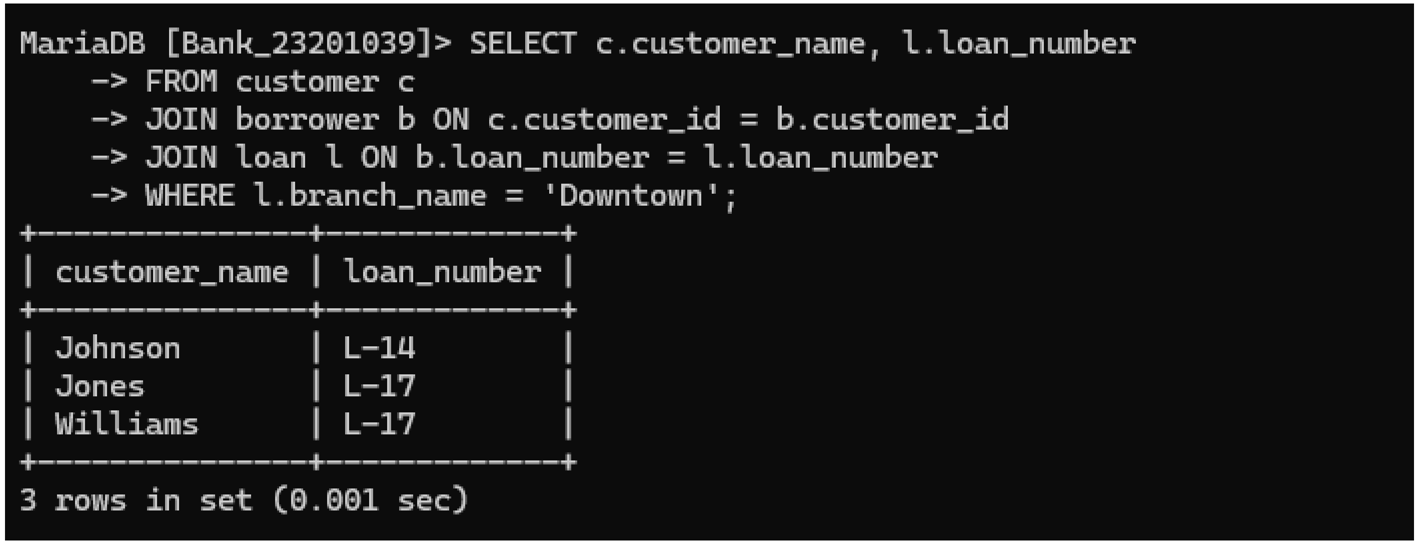
1. Find the name and loan number of all customers having a loan at the Downtown branch. [2]
2. Find all the possible pairs of customers who are from the same city. show in the format Customer1, Customer2, City. [2]
3. If the bank gives out 4% interest to all accounts, show the total interest across each branch. Print Branch_name, Total_Interest [2]
4. Find account numbers with the highest balances for each city in the database [2]
5. Show the loan number, loan amount, and name of customers with the top 5 highest loan amounts. The data should be sorted by increasing amounts, then decreasing loan numbers in case of the same loan amount. [Hint for top 5: Check the "limit" keyword in mysql] [2]
6. Find the names of customers with an account and also a loan at the Perryridge branch. [2]
7. Find the total loan amount of all customers having at least 2 loans from the bank. Show in format customer name, total_loan. [2]

CSE370 : Database Systems

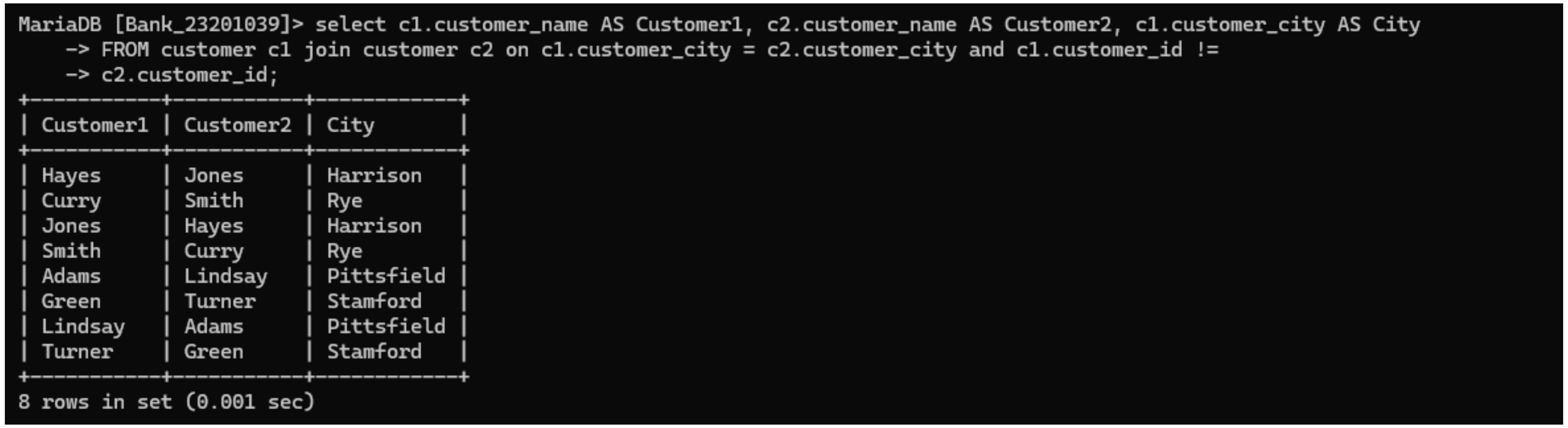
Assignment 03 | Fall 2025

ID : <23201039> | Name : Priyanti Biswas

No 1 Query (as Plain Text)	SELECT c.customer_name, l.loan_number
	FROM customer c
	JOIN borrower b ON c.customer_id = b.customer_id
	JOIN loan l ON b.loan_number = l.loan_number
	WHERE l.branch_name = 'Downtown';

No 1 SS (of Query & Output in Shell)	
No 2 Query (as Plain Text)	select c1.customer_name AS Customer1, c2.customer_name AS Customer2, c1.customer_city AS City FROM customer c1 join customer c2 on c1.customer_city = c2.customer_city and c1.customer_id != c2.customer_id;

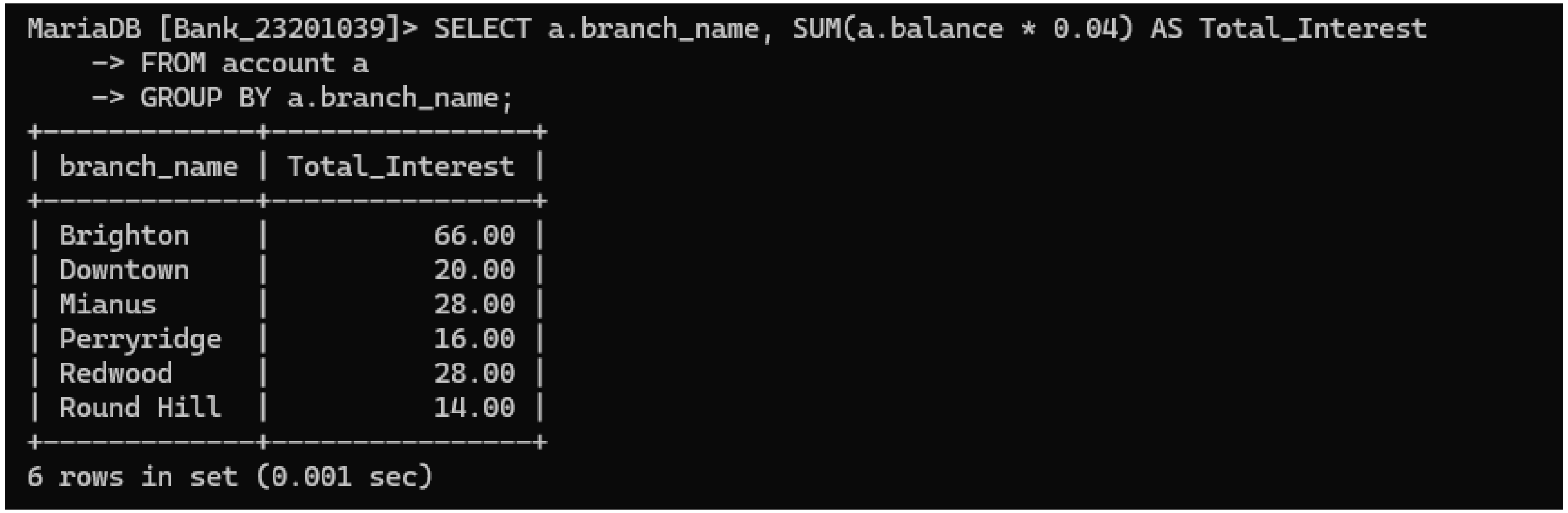
```
MariaDB [Bank_23201039]> SELECT c.customer_name, l.loan_number
    -> FROM customer c
    -> JOIN borrower b ON c.customer_id = b.customer_id
    -> JOIN loan l ON b.loan_number = l.loan_number
    -> WHERE l.branch_name = 'Downtown';
+---------------+-------------+
| customer_name | loan_number |
+---------------+-------------+
| Johnson       | L-14        |
| Jones         | L-17        |
| Williams      | L-17        |
+---------------+-------------+
3 rows in set (0.001 sec)
```

No 2 SS (of Query & Output in Shell)	

No 3 Query (as Plain Text)	SELECT a.branch_name, SUM(a.balance * 0.04) AS Total_Interest
	FROM account a
	GROUP BY a.branch_name;

```
MariaDB [Bank_23201039]> select c1.customer_name AS Customer1, c2.customer_name AS Customer2, c1.customer_city AS City
    -> FROM customer c1 join customer c2 on c1.customer_city = c2.customer_city and c1.customer_id !=
    -> c2.customer_id;
+-----------+-----------+------------+
| Customer1 | Customer2 | City       |
+-----------+-----------+------------+
| Hayes     | Jones     | Harrison   |
| Curry     | Smith     | Rye        |
| Jones     | Hayes     | Harrison   |
| Smith     | Curry     | Rye        |
| Adams     | Lindsay   | Pittsfield |
| Green     | Turner    | Stamford   |
| Lindsay   | Adams     | Pittsfield |
| Turner    | Green     | Stamford   |
+-----------+-----------+------------+
8 rows in set (0.001 sec)
```

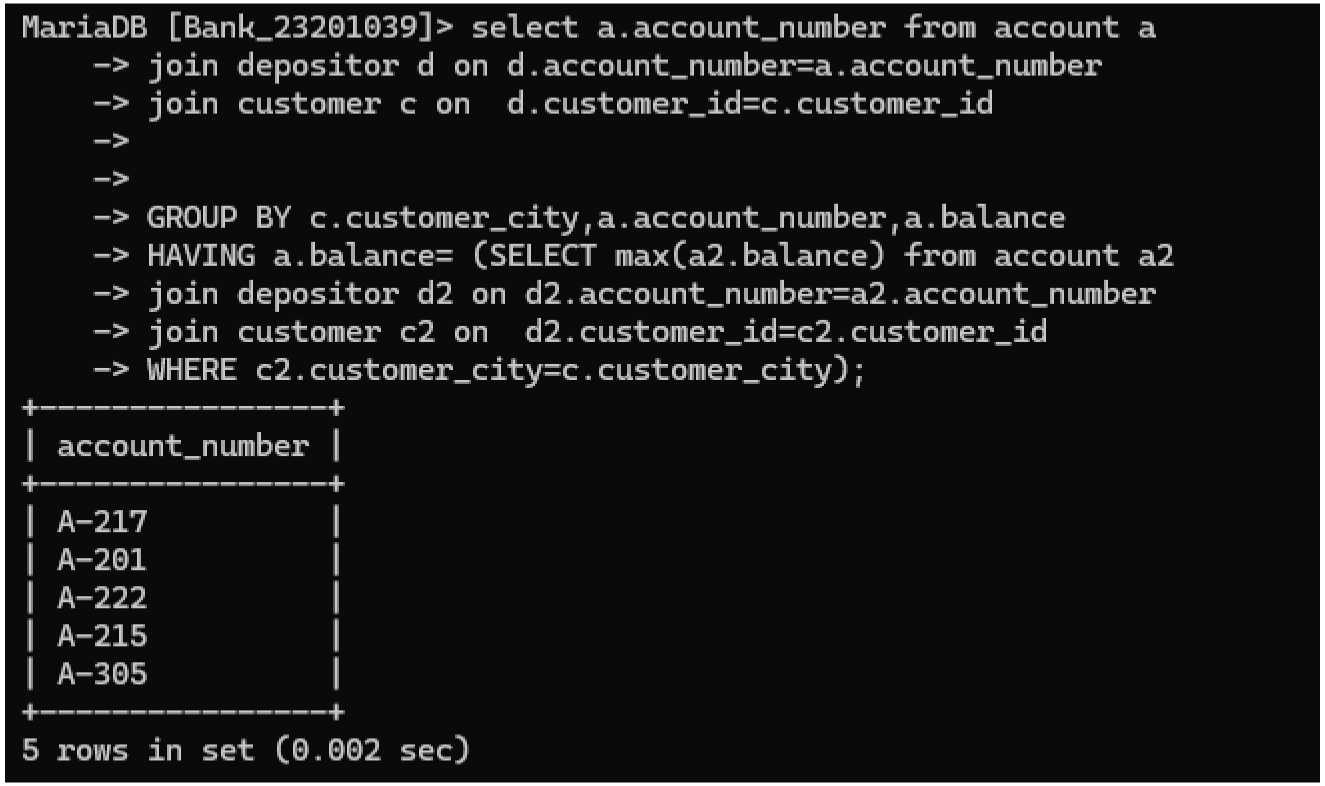
No 3 SS
(of Query & Output
in Shell)



```
MariaDB [Bank_23201039]> SELECT a.branch_name, SUM(a.balance * 0.04) AS Total_Interest
    -> FROM account a
    -> GROUP BY a.branch_name;
+-------------+----------------+
| branch_name | Total_Interest |
+-------------+----------------+
| Brighton    |          66.00 |
| Downtown    |          20.00 |
| Mianus      |          28.00 |
| Perryridge  |          16.00 |
| Redwood     |          28.00 |
| Round Hill  |          14.00 |
+-------------+----------------+
6 rows in set (0.001 sec)
```

No 4 Query
(as Plain Text)

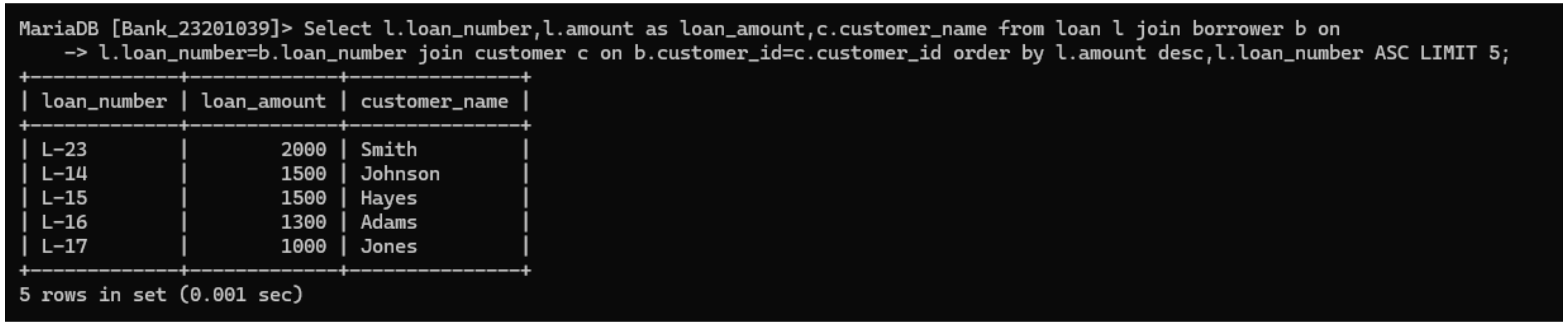
```
select a.account_number from account a
join depositor d on d.account_number=a.account_number
join customer c on  d.customer_id=c.customer_id
GROUP BY c.customer_city,a.account_number,a.balance
HAVING a.balance= (SELECT max(a2.balance) from account a2
join depositor d2 on d2.account_number=a2.account_number
join customer c2 on  d2.customer_id=c2.customer_id
WHERE c2.customer_city=c.customer_city);
```

No 4 SS (of Query & Output in Shell)	

```
MariaDB [Bank_23201039]> select a.account_number from account a
    -> join depositor d on d.account_number=a.account_number
    -> join customer c on  d.customer_id=c.customer_id
    ->
    ->
    -> GROUP BY c.customer_city,a.account_number,a.balance
    -> HAVING a.balance= (SELECT max(a2.balance) from account a2
    -> join depositor d2 on d2.account_number=a2.account_number
    -> join customer c2 on  d2.customer_id=c2.customer_id
    -> WHERE c2.customer_city=c.customer_city);
+----------------+
| account_number |
+----------------+
| A-217          |
| A-201          |
| A-222          |
| A-215          |
| A-305          |
+----------------+
5 rows in set (0.002 sec)
```

No 5 Query (as Plain Text)	Select l.loan_number,l.amount as loan_amount,c.customer_name from loan l join borrower b on l.loan_number=b.loan_number join customer c on b.customer_id=c.customer_id order by l.amount desc,l.loan_number ASC LIMIT 5;

**No 5 SS
(of Query & Output
in Shell)**

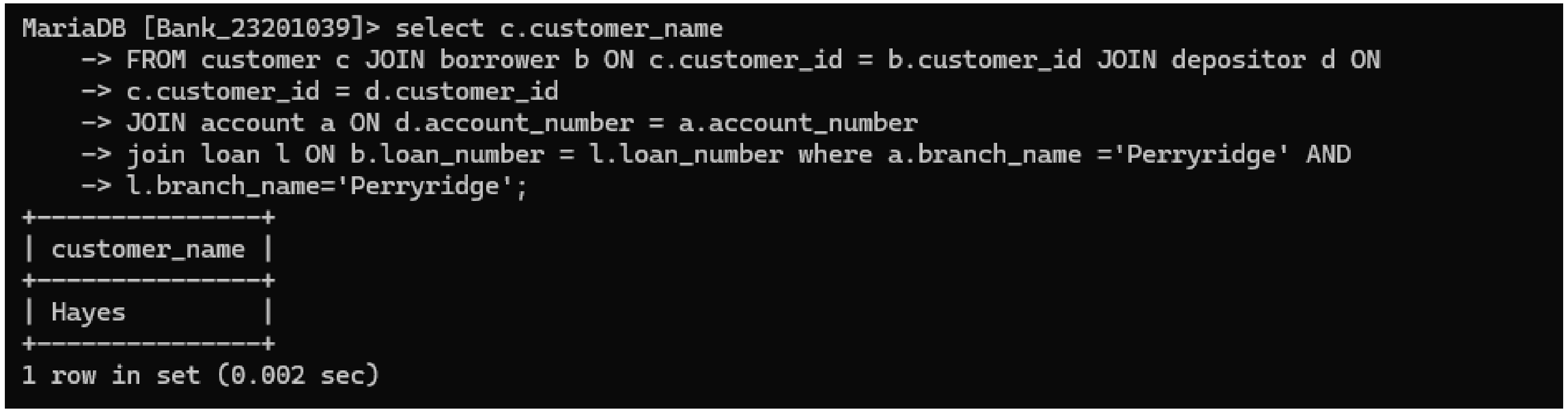


```
MariaDB [Bank_23201039]> Select l.loan_number,l.amount as loan_amount,c.customer_name from loan l join borrower b on
    -> l.loan_number=b.loan_number join customer c on b.customer_id=c.customer_id order by l.amount desc,l.loan_number ASC LIMIT 5;
+-------------+-------------+---------------+
| loan_number | loan_amount | customer_name |
+-------------+-------------+---------------+
| L-23        |        2000 | Smith         |
| L-14        |        1500 | Johnson       |
| L-15        |        1500 | Hayes         |
| L-16        |        1300 | Adams         |
| L-17        |        1000 | Jones         |
+-------------+-------------+---------------+
5 rows in set (0.001 sec)
```

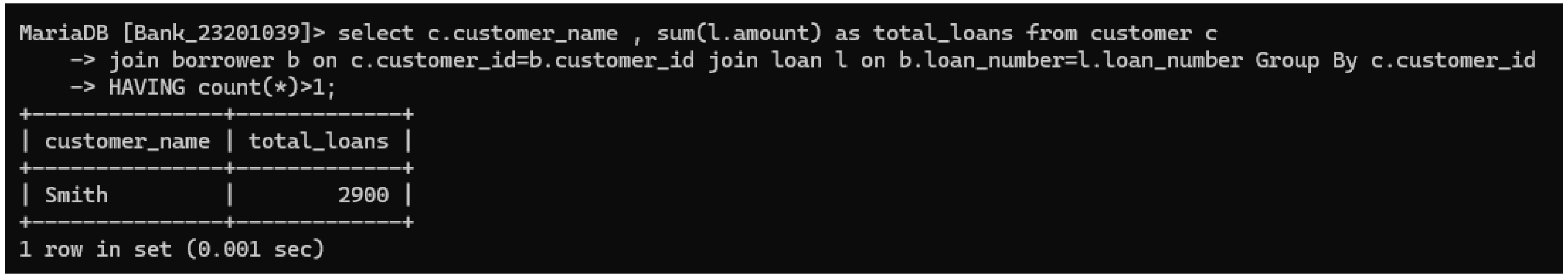
**No 6 Query
(as Plain Text)**

select c.customer_name
FROM customer c JOIN borrower b ON c.customer_id = b.customer_id JOIN depositor d ON
c.customer_id = d.customer_id
JOIN account a ON d.account_number = a.account_number
join loan l ON b.loan_number = l.loan_number where a.branch_name ='Perryridge' AND
l.branch_name='Perryridge';

**No 6 SS
(of Query & Output
in Shell)**



```
MariaDB [Bank_23201039]> select c.customer_name
    -> FROM customer c JOIN borrower b ON c.customer_id = b.customer_id JOIN depositor d ON
    -> c.customer_id = d.customer_id
    -> JOIN account a ON d.account_number = a.account_number
    -> join loan l ON b.loan_number = l.loan_number where a.branch_name ='Perryridge' AND
    -> l.branch_name='Perryridge';
+---------------+
| customer_name |
+---------------+
| Hayes         |
+---------------+
1 row in set (0.002 sec)
```

No 7 Query (as Plain Text)	select c.customer_name , sum(l.amount) as total_loans from customer c join borrower b on c.customer_id=b.customer_id join loan l on b.loan_number=l.loan_number Group By c.customer_id HAVING count(*)>1;
No 7 SS (of Query & Output in Shell)	 See output below.

```
MariaDB [Bank_23201039]> select c.customer_name , sum(l.amount) as total_loans from customer c
    -> join borrower b on c.customer_id=b.customer_id join loan l on b.loan_number=l.loan_number Group By c.customer_id
    -> HAVING count(*)>1;
+---------------+-------------+
| customer_name | total_loans |
+---------------+-------------+
| Smith         |        2900 |
+---------------+-------------+
1 row in set (0.001 sec)
```