

---

*CSCI 5408*

*Data Management, Warehousing, And  
Analytics*

---

---

*Lab 4: Introduction to Distributed Database*

---

**Prepared By**

Bhavisha Oza (B00935827)

## Summary

Types of databases include centralized, decentralized, and distributed. Centralized databases store data in a single location, providing simplicity but risking single points of failure.[2] Decentralized databases distribute data across multiple locations, offering better fault tolerance but potentially complex coordination. Distributed databases combine both approaches.[2]

Setting up a MySQL database on Google Cloud Platform (GCP) involves creating a GCP project, enabling Cloud SQL, and configuring the database instance. Once the set-up is done, we can interact with the database using various tools and libraries.

In the hands-on portion, I practiced executing SQL queries, creating tables, inserting data, and retrieving information from the MySQL database on GCP. This provides practical experience with managing and manipulating data in a cloud-based environment.

## Steps followed:

- Completed setup of MySQL on GCP as taught in the lab [2].
- Created a local database with User table, and Order\_info table in MySQL.
- Created a remote database with Inventory table in GCP.
- Created a Java program that –
  - Fetches item details from the remote database
  - Creates an order table in local database.
  - Writes the updated quantity back to the remote database upon order creation.

## Lab exercise:

1. Set up a simple e commerce distributed database system.

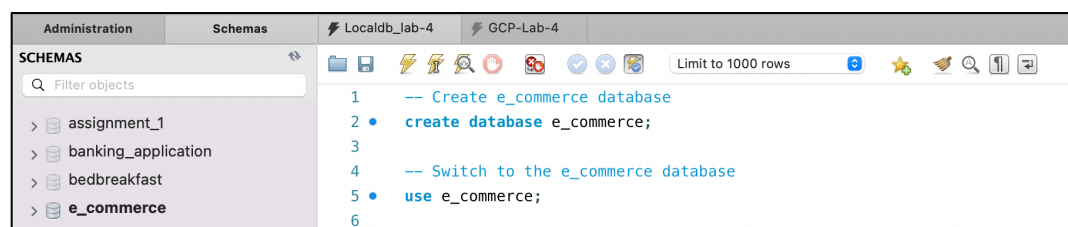
### SQL Queries used:

-- Create e\_commerce database

```
CREATE DATABASE e_commerce;
```

-- Switch to the e\_commerce database

```
USE e_commerce;
```



**Figure 1:** e\_commerce database creation

2. Create a local database with –
  - a. User table (attributes – id,name, email, phone, address)

### SQL Queries used:

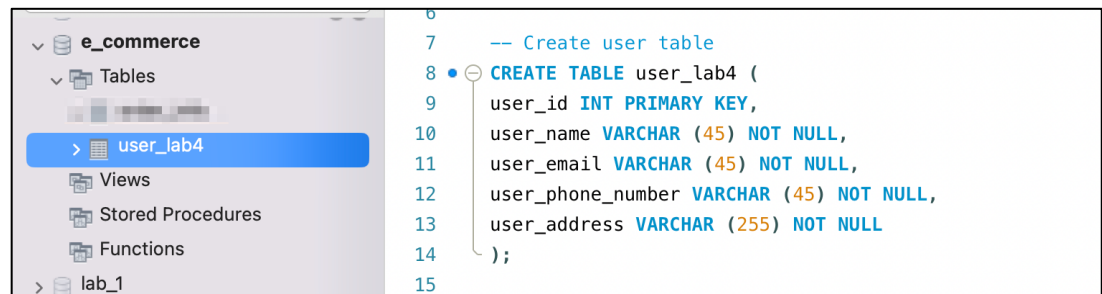
```

-- Create user table
CREATE TABLE user_lab4 (
user_id INT PRIMARY KEY,
user_name VARCHAR (45) NOT NULL,
user_email VARCHAR (45) NOT NULL,
user_phone_number VARCHAR (45) NOT NULL,
user_address VARCHAR (255) NOT NULL
);

-- Insert records into user table
INSERT INTO user_lab4 (user_id, user_name, user_email, user_phone_number,
user_address)
VALUES ('01', 'Bhavisha', 'bhavisha.oza@dal.ca', '4379860129', 'Halifax, NS');

INSERT INTO user_lab4 (user_id, user_name, user_email, user_phone_number,
user_address)
VALUES ('02', 'Bhaumik', 'bhaumik.bhatt@dal.ca', '4379868620', 'Toronto, ON');

```



**Figure 2:** user\_lab4 table creation

- b. Order\_info table (order\_id, user\_id, item\_name, quantity, order\_date).

#### SQL Queries used:

```

-- Create order_info table
CREATE TABLE order_info (
order_id INT PRIMARY KEY,
user_id VARCHAR (45) NOT NULL,
item_name VARCHAR (45) NOT NULL,
quantity VARCHAR (45) NOT NULL,
order_date DATETIME NOT NULL
);

-- Insert records into order_info table
INSERT INTO order_info (order_id, user_id, item_name, quantity, order_date)
VALUES ('01', '01', 'Soap', '50', '2023-06-07');

```

```
INSERT INTO order_info (order_id, user_id, item_name, quantity, order_date)
VALUES ('02', '02', 'Facewash', '67', '2023-05-17');
```

3. Create a remote database in G C P with –
  - a. Inventory table (item\_id,item\_name, available\_quantity)

### SQL Queries used:

```
-- Create e_commerce database
create database e_commerce;
```

```
-- Switch to the e_commerce database
use e_commerce;
```

```
-- Create inventory table
CREATE TABLE inventory (
item_id INT PRIMARY KEY,
item_name VARCHAR (45) NOT NULL,
available_quantity VARCHAR (45) NOT NULL
);
```

```
-- Insert record into inventory table
INSERT INTO inventory (item_id, item_name, available_quantity)
VALUES ('01', 'Shampoo', '23');
```

4. Write a Java language program that:
  - a. Fetches item details from the remote database
    - i. Made the db connection and written the code in java.
    - ii. Started the SQL instance on GCP.
    - iii. Executed the program.

The screenshot shows an IDE with two tabs: DBConnection.java and GCPLab4RunApplication.java. The DBConnection.java tab is active, showing a try block with the following code:

```
try {
    // Establish a connection to the GCP database
    Connection connection = DBConnection.getRemoteDBConnection();

    Statement statement = connection.createStatement();

    // Execute a SELECT query to fetch item details from the inventory table
    String sqlQuery = "SELECT * FROM inventory";
    ResultSet resultSet = statement.executeQuery(sqlQuery);

    // Process the result set
    while (resultSet.next()) {
        int itemId = resultSet.getInt(resultSet.getColumnLabel("item_id"));
        String itemName = resultSet.getString(resultSet.getColumnLabel("item_name"));
        int availableQuantity = resultSet.getInt(resultSet.getColumnLabel("available_quantity"));

        // Print the item details
        System.out.println("Item ID: " + itemId);
        System.out.println("Item Name: " + itemName);
        System.out.println("Available Quantity: " + availableQuantity);
        System.out.println("-----");
    }
}
```

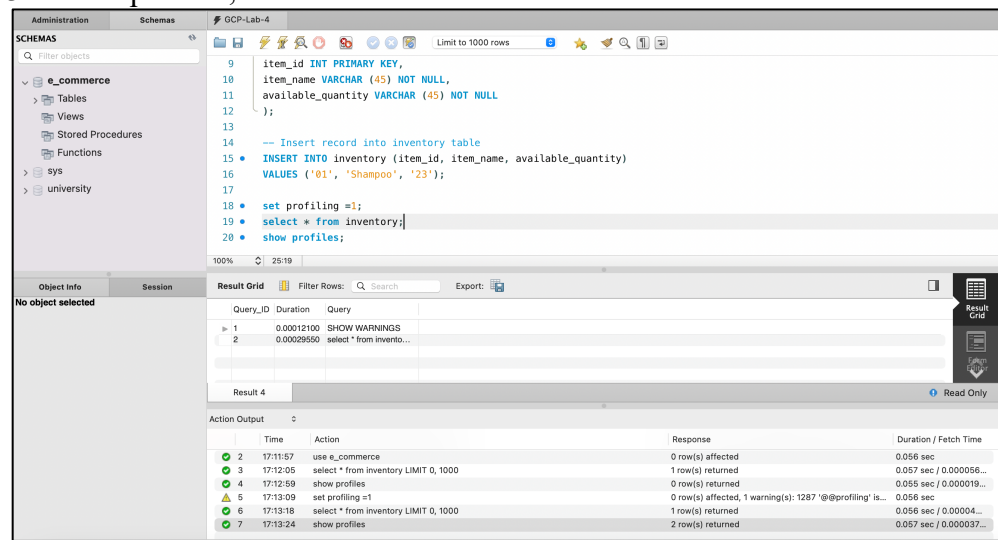
The Run console at the bottom shows the output of the program:

```
Item ID: 1
Item Name: Shampoo
Available Quantity: 23
-----
Process finished with exit code 0
```

**Figure 3:** Code output which fetches item details from the remote database.

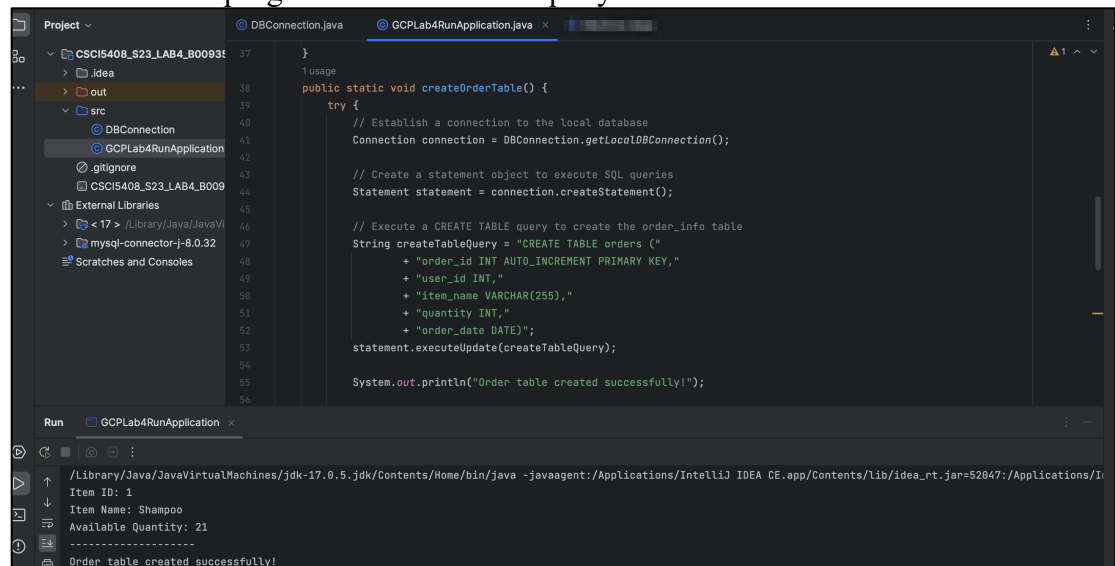
- iv. In MySQL Workbench, followed the steps:
  1. set profiling =1;
  2. select \* from inventory;

### 3. show profiles;



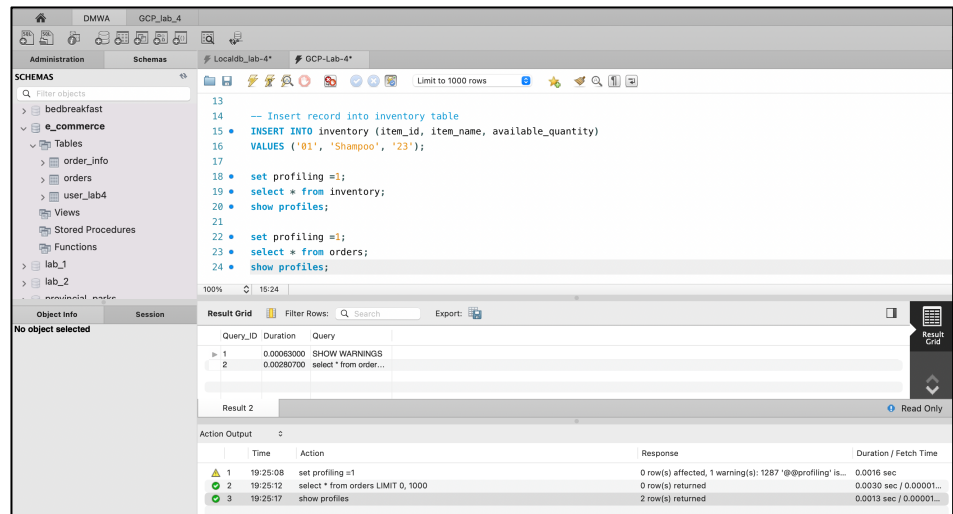
**Figure 4:** SQL output for query execution time which fetches item details from the remote db.

- b. Creates an order in local database.
  - i. Executed the program for create table query.



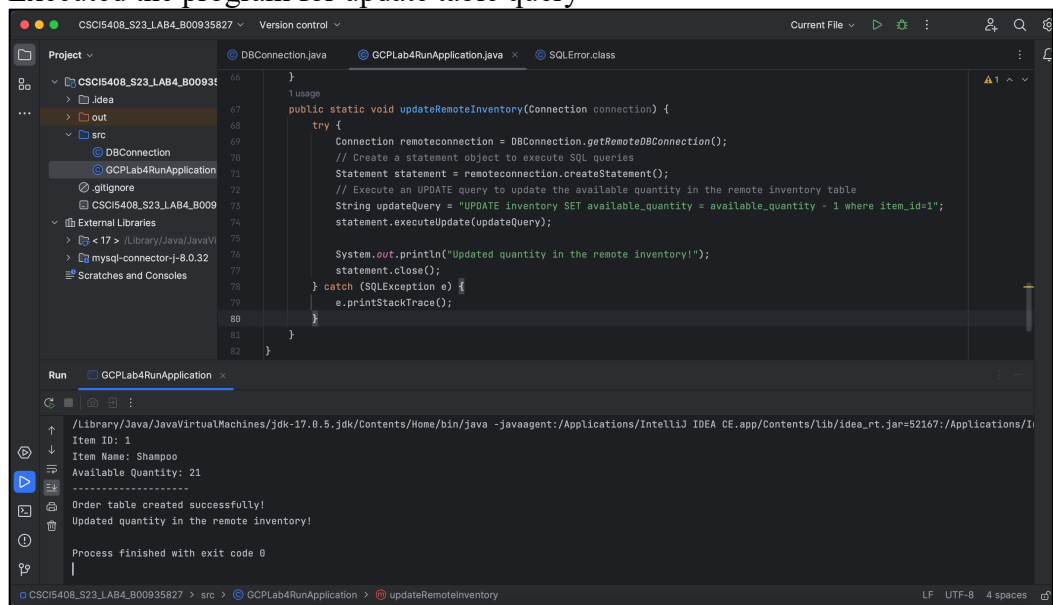
**Figure 5:** Code output which creates an order table in local database.

- ii. In MySQL Workbench, followed the steps:
  1. set profiling =1;
  2. select \* from orders;
  3. show profiles;



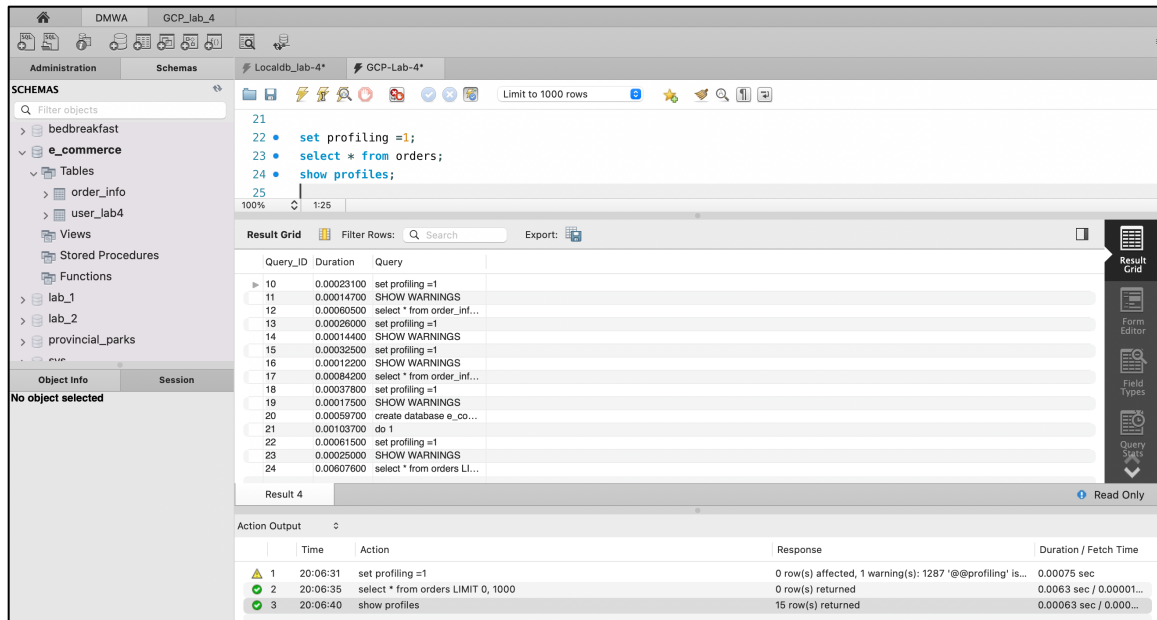
**Figure 6:** SQL output for query execution time which creates the order table in local db.

- c. Writes the updated quantity back to the remote database upon order creation.
  - i. Executed the program for update table query



**Figure 7:** Code output which updates an order table in remote database.

- ii. In MySQL Workbench, followed the steps:
  1. set profiling =1;
  2. select \* from orders;
  3. show profiles;



**Figure 8:** SQL output of query execution time which updates an order table in remote db.

Full code can be found in the given repository: [https://git.cs.dal.ca/boza/csci-5408/-/tree/main/Lab-4/CSCI5408\\_S23\\_LAB4\\_B00935827](https://git.cs.dal.ca/boza/csci-5408/-/tree/main/Lab-4/CSCI5408_S23_LAB4_B00935827)

## References:

- [1] "MySQL Community Downloads," *MySQL* [Online]. Available: <https://dev.mysql.com/downloads/workbench/> [Accessed: May 10, 2023].
- [2] "Lab-4," *Brightspace Dalhousie University* [Online]. Available: <https://dal.brightspace.com/d21/le/content/271677/viewContent/3644625/View> [Accessed: June 7, 2023].