CSCI 5408

DATA MANAGEMENT AND WAREHOUSING

DALHOUSIE UNIVERSITY

LAB ASSIGNMENT - 4

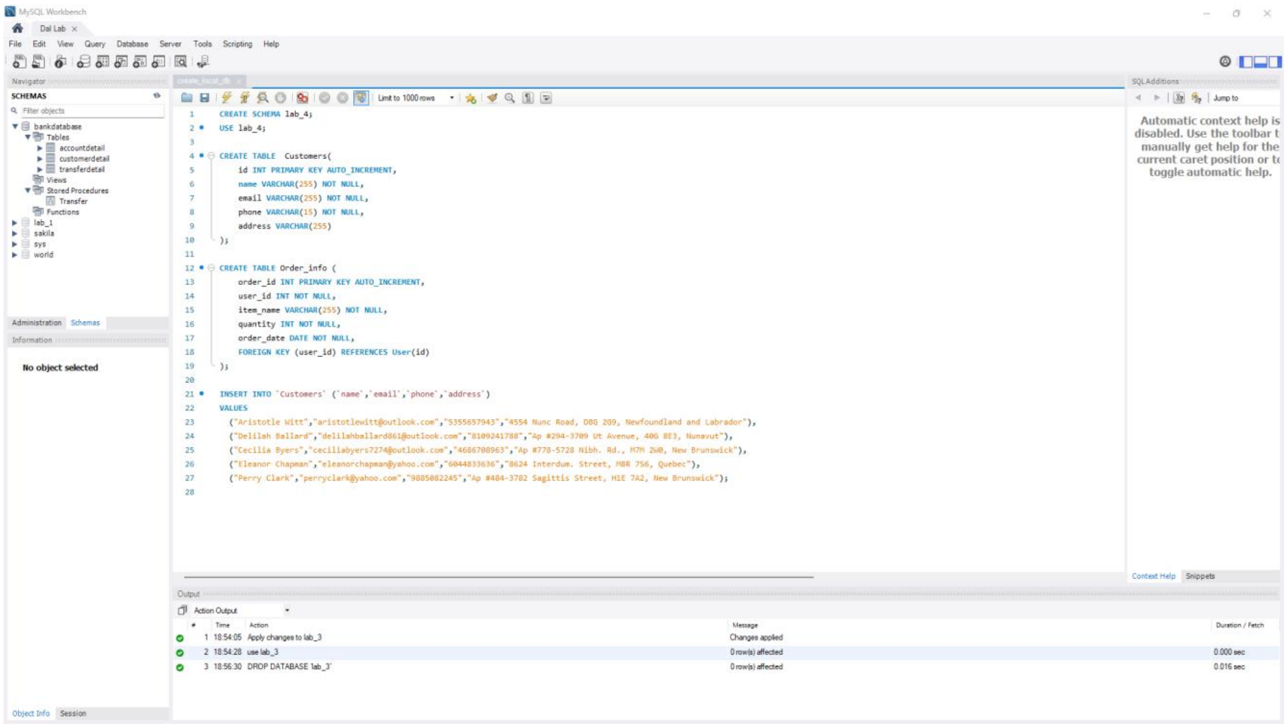
Submitted By:	Kenil Shaileshkumar Patel (kenil.patel@dal.ca)
Banner ID:	B00954251
Submitted On:	October 25, 2023

Gitlab Repository Link

https://git.cs.dal.ca/kenil/csci5408_f23_b00954251_kenil_patel/-/tree/main/Lab4

Table of Contents

Setting up database on local and remote server for Problem



```sql
CREATE SCHEMA lab_4;
USE lab_4;

CREATE TABLE  Customers(
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    phone VARCHAR(15) NOT NULL,
    address VARCHAR(255)
);

CREATE TABLE Order_info (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    item_name VARCHAR(255) NOT NULL,
    quantity INT NOT NULL,
    order_date DATE NOT NULL,
    FOREIGN KEY (user_id) REFERENCES User(id)
);

INSERT INTO `Customers` (`name`,`email`,`phone`,`address`)
VALUES
  ("Aristotle Witt","aristotlewitt@outlook.com","5355657943","4554 Nunc Road, D8G 2G9, Newfoundland and Labrador"),
  ("Delilah Ballard","delilahballard861@outlook.com","8109241788","Ap #294-3709 Ut Avenue, 40G 8E3, Nunavut"),
  ("Cecilia Byers","ceciliabyers7274@outlook.com","4686708963","Ap #778-5728 Nibh. Rd., M7M 2W0, New Brunswick"),
  ("Eleanor Chapman","eleanorchapman@yahoo.com","6044833636","8624 Interdum. Street, M8R 7S6, Quebec"),
  ("Perry Clark","perryclark@yahoo.com","9885082245","Ap #484-3782 Sagittis Street, H1E 7A2, New Brunswick");
```

```
#  Time      Action                      Message             Duration / Fetch
1  18:54:05  Apply changes to lab_3      Changes applied
2  18:54:28  use lab_3                   0 row(s) affected   0.000 sec
3  18:56:30  DROP DATABASE `lab_3`       0 row(s) affected   0.016 sec
```

Fig 1: Query to create and populate the Customers and Order_info table.

Below is the proof of the schema and its table and the database that was created.

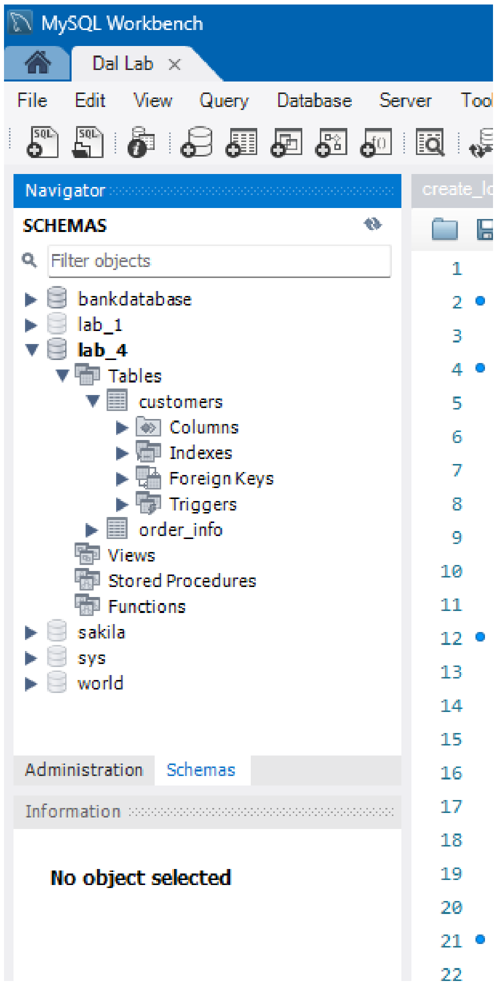
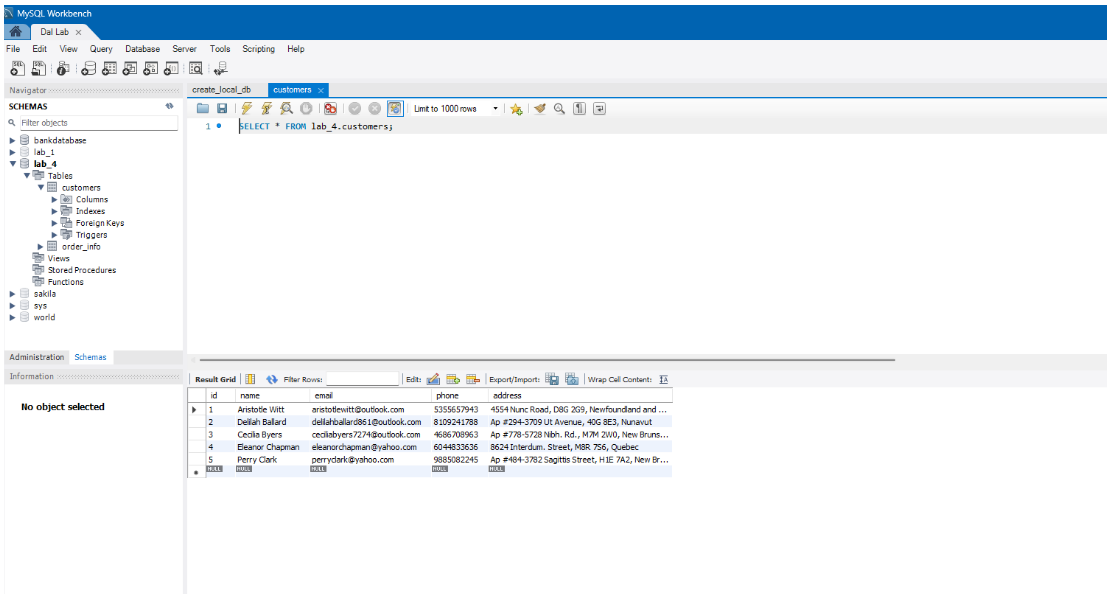


Fig 2: Proof of table creation.

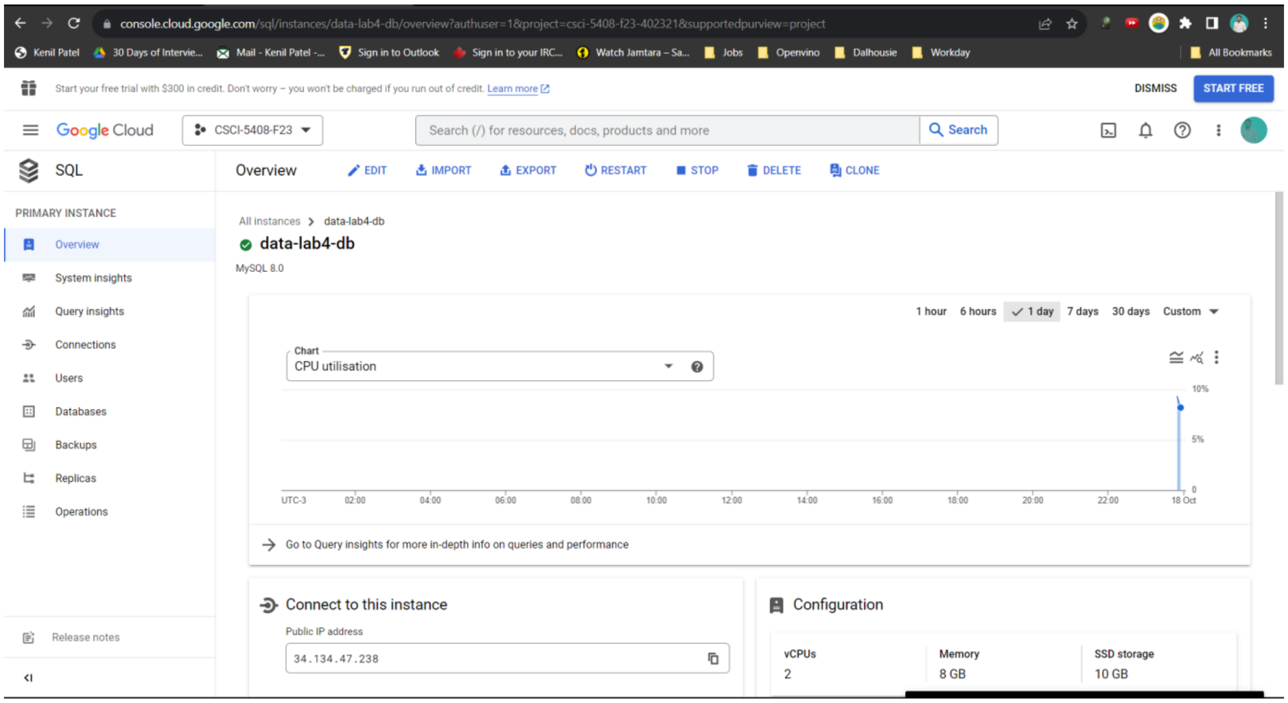


```
SELECT * FROM lab_4.customers;
```

id	name	email	phone	address
1	Aristotle Witt	aristotlewitt@outlook.com	5355657943	4554 Nunc Road, D8G 2G9, Newfoundland and ...
2	Delilah Ballard	delilahballard861@outlook.com	8109241788	Ap #294-3709 Ut Avenue, 40G 8E3, Nunavut
3	Cecilia Byers	ceciliabyers7274@outlook.com	4686708963	Ap #778-5728 Nibh. Rd., M7M 2W0, New Bruns...
4	Eleanor Chapman	eleanorchapman@yahoo.com	6044833636	8624 Interdum. Street, M8R 7S6, Quebec
5	Perry Clark	perryclark@yahoo.com	9885082245	Ap #484-3782 Sagittis Street, H1E 7A2, New Br...
NULL	NULL	NULL	NULL	NULL

Fig 3: Result when the customer table is selected in local database.

Creating an SQL instance on Google Cloud so we can create the Inventory table on the SQL instance which would be remotely present at Google Cloud.



All instances > data-lab4-db

data-lab4-db

MySQL 8.0

Connect to this instance

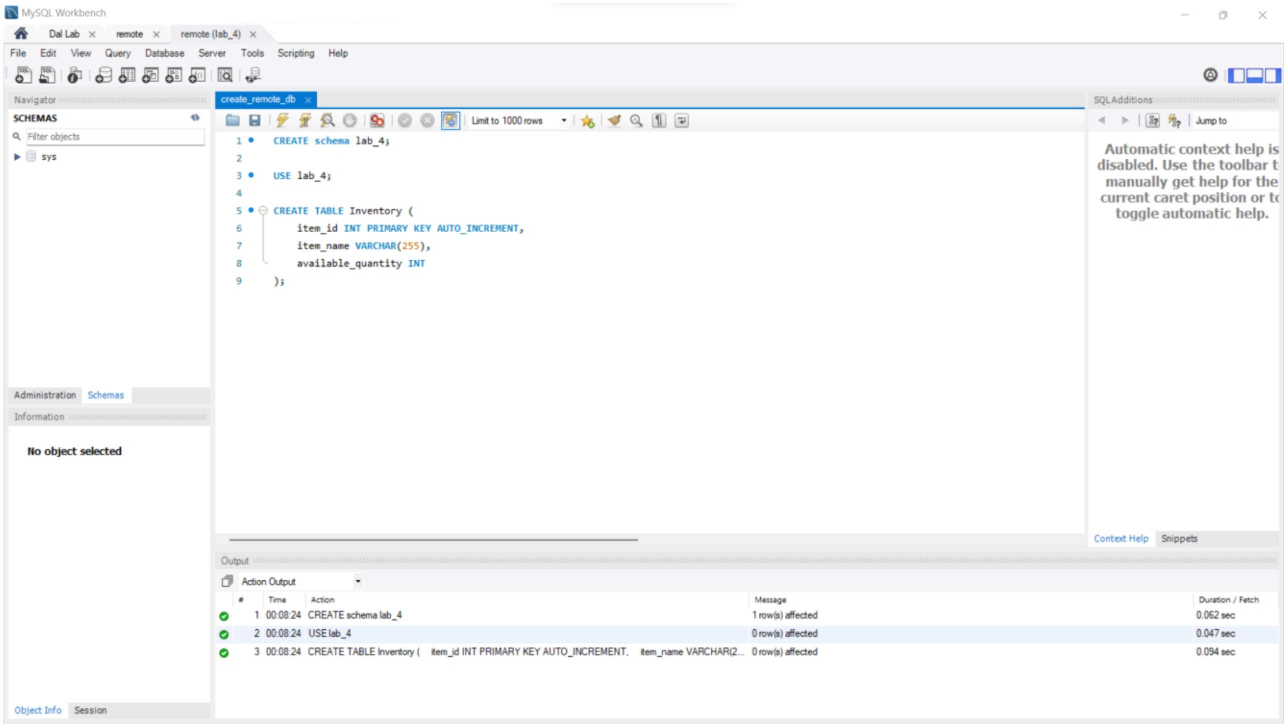
Public IP address: 34.134.47.238

Configuration

vCPUs	Memory	SSD storage
2	8 GB	10 GB

Fig 4: Cloud dashboard when SQL instance is created.

Creating the table Inventory on the SQL instance that we just created.

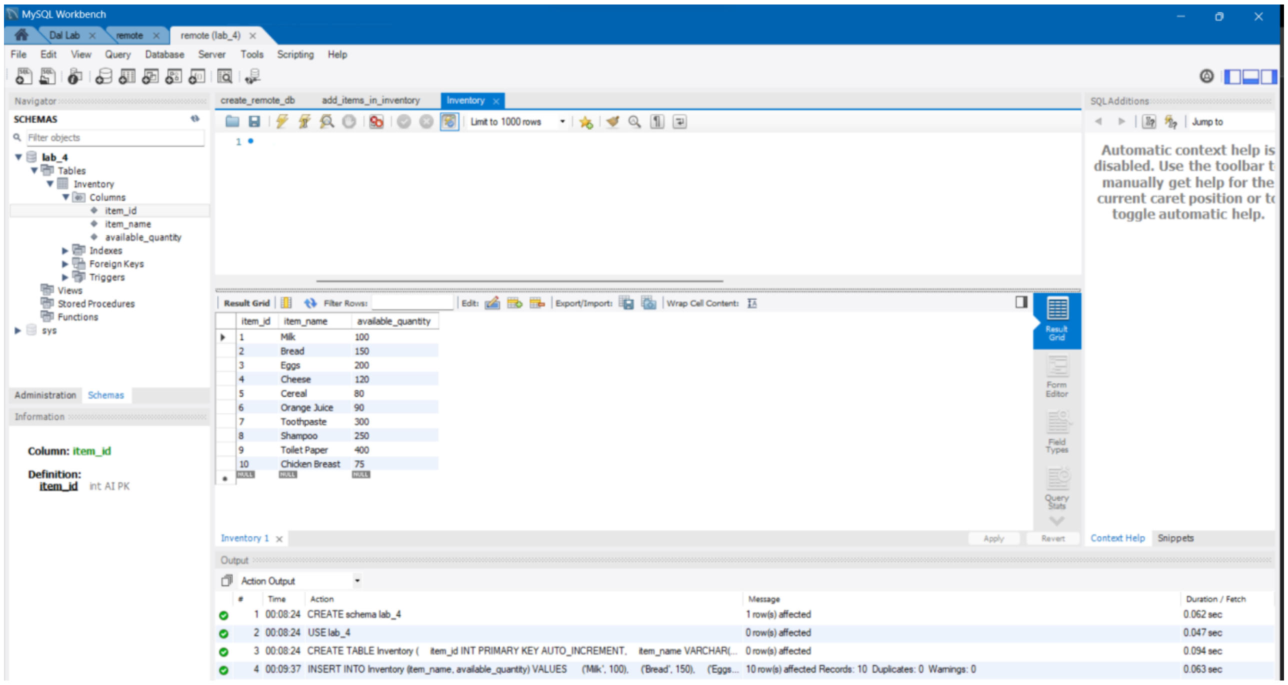

MySQL Workbench
Dal Lab
remote
remote (lab_4)
File Edit View Query Database Server Tools Scripting Help
create_remote_db
Limit to 1000 rows
SCHEMAS
sys
No object selected
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
CREATE schema lab_4;

USE lab_4;

CREATE TABLE Inventory (
    item_id INT PRIMARY KEY AUTO_INCREMENT,
    item_name VARCHAR(255),
    available_quantity INT
);
```

Action Output

#	Time	Action	Message	Duration / Fetch
1	00:08:24	CREATE schema lab_4	1 row(s) affected	0.062 sec
2	00:08:24	USE lab_4	0 row(s) affected	0.047 sec
3	00:08:24	CREATE TABLE Inventory (item_id INT PRIMARY KEY AUTO_INCREMENT, item_name VARCHAR(2...	0 row(s) affected	0.094 sec

Fig 5: Creating and populating the table on the Remote Cloud instance.



item_id	item_name	available_quantity
1	Milk	100
2	Bread	150
3	Eggs	200
4	Cheese	120
5	Cereal	80
6	Orange Juice	90
7	Toothpaste	300
8	Shampoo	250
9	Toilet Paper	400
10	Chicken Breast	75
NULL	NULL	NULL

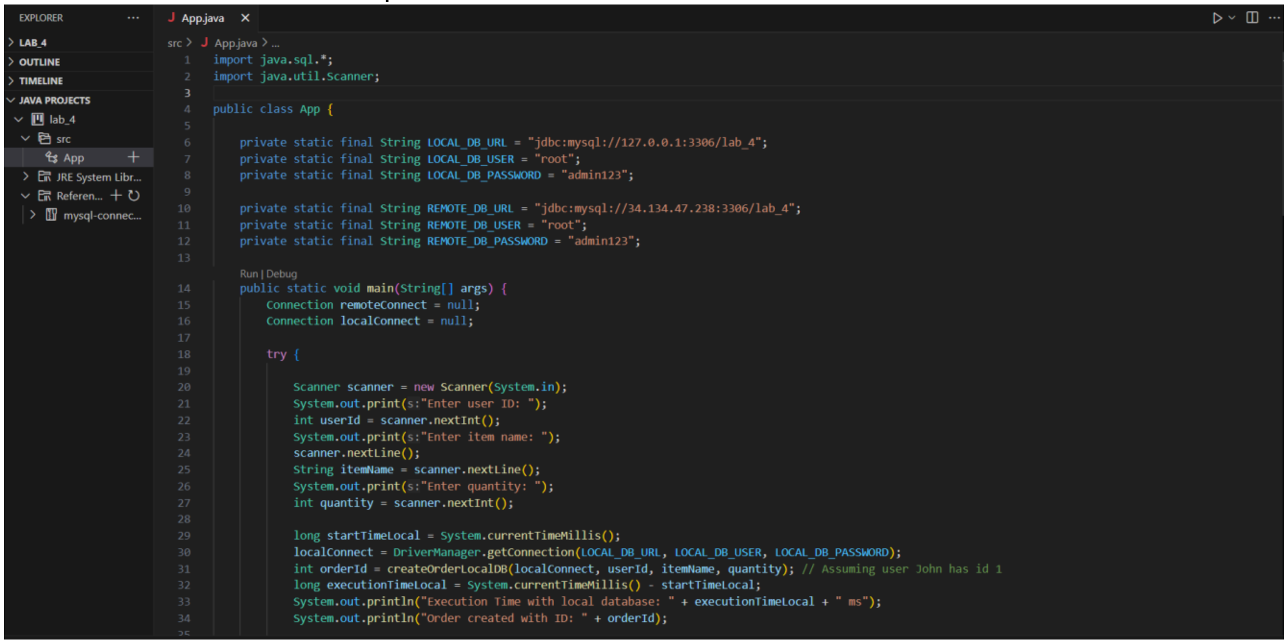
```
#  Time      Action                                                                                      Message                                                     Duration / Fetch
1  00:08:24  CREATE schema lab_4                                                                         1 row(s) affected                                           0.062 sec
2  00:08:24  USE lab_4                                                                                   0 row(s) affected                                           0.047 sec
3  00:08:24  CREATE TABLE Inventory (    item_id INT PRIMARY KEY AUTO_INCREMENT,    item_name VARCHAR(...  0 row(s) affected                                           0.094 sec
4  00:09:37  INSERT INTO Inventory (item_name, available_quantity) VALUES    ('Milk', 100),    ('Bread', 150),    ('Eggs...  10 row(s) affected Records: 10  Duplicates: 0  Warnings: 0  0.063 sec
```

Fig 6: Inventory table on remote SQL instance.

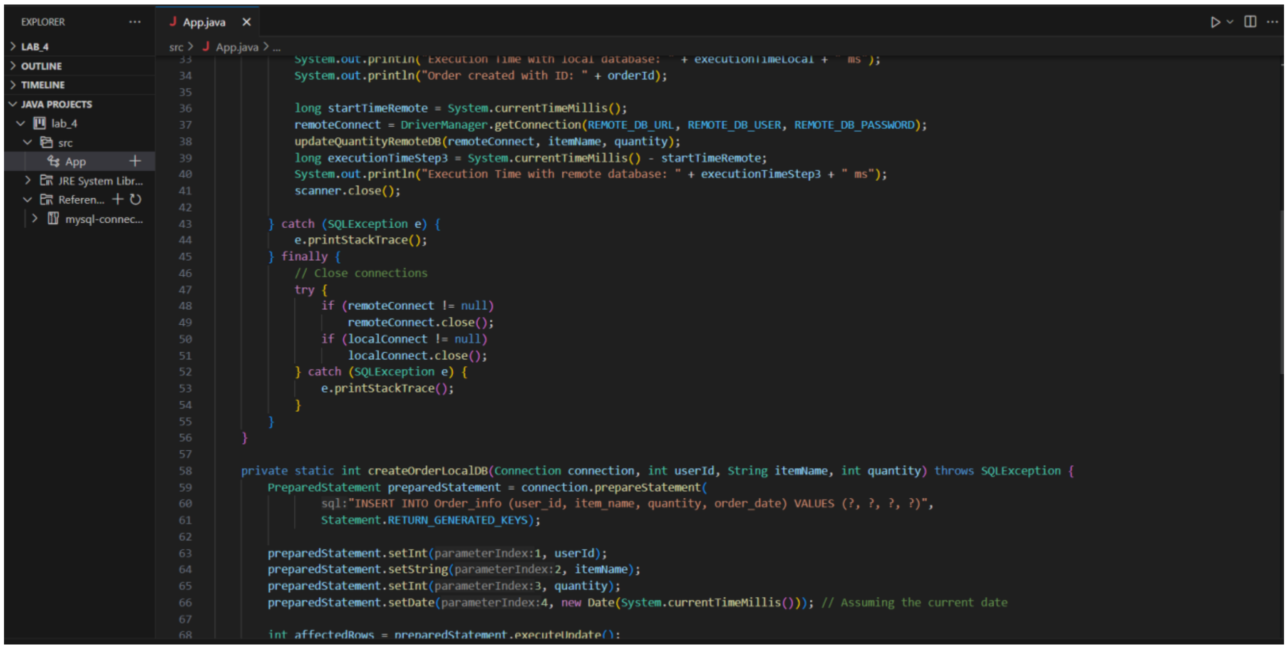
Problem Statement :

1. Fetches item details from the remote database.
2. Creates an order in a local database.
3. Write the updated quantity back to the remote database upon order creation.

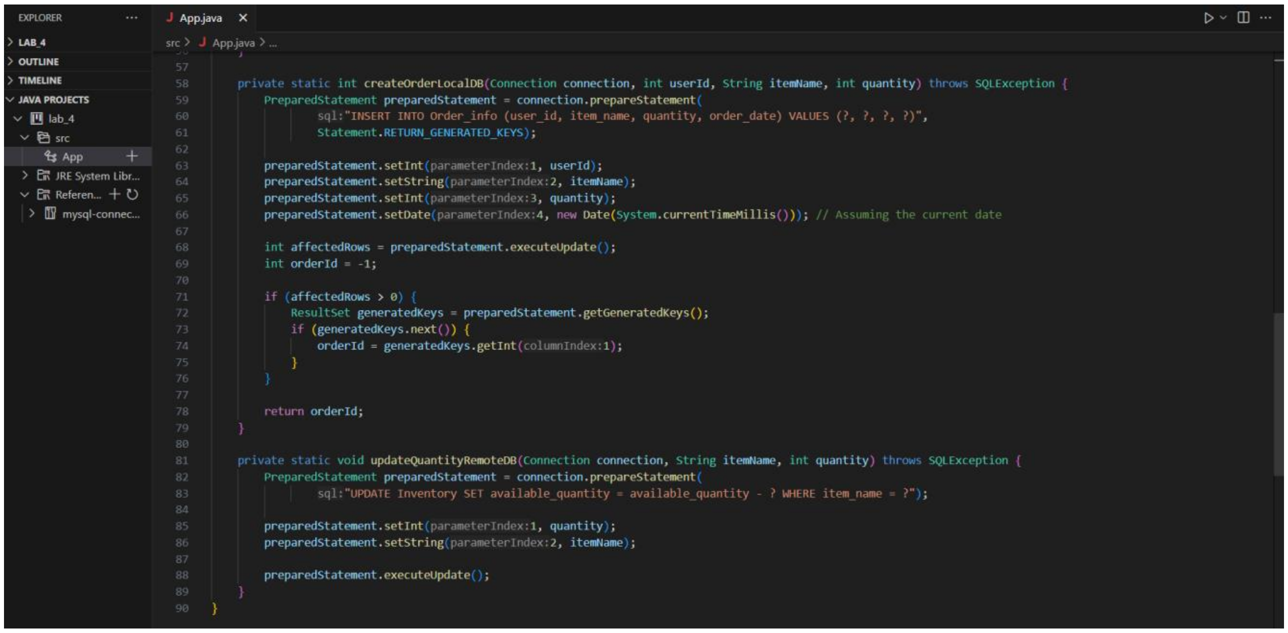
The Java code for the above problem statement is as follows:



```java
import java.sql.*;
import java.util.Scanner;

public class App {

    private static final String LOCAL_DB_URL = "jdbc:mysql://127.0.0.1:3306/lab_4";
    private static final String LOCAL_DB_USER = "root";
    private static final String LOCAL_DB_PASSWORD = "admin123";

    private static final String REMOTE_DB_URL = "jdbc:mysql://34.134.47.238:3306/lab_4";
    private static final String REMOTE_DB_USER = "root";
    private static final String REMOTE_DB_PASSWORD = "admin123";

    public static void main(String[] args) {
        Connection remoteConnect = null;
        Connection localConnect = null;

        try {

            Scanner scanner = new Scanner(System.in);
            System.out.print(s:"Enter user ID: ");
            int userId = scanner.nextInt();
            System.out.print(s:"Enter item name: ");
            scanner.nextLine();
            String itemName = scanner.nextLine();
            System.out.print(s:"Enter quantity: ");
            int quantity = scanner.nextInt();

            long startTimeLocal = System.currentTimeMillis();
            localConnect = DriverManager.getConnection(LOCAL_DB_URL, LOCAL_DB_USER, LOCAL_DB_PASSWORD);
            int orderId = createOrderLocalDB(localConnect, userId, itemName, quantity); // Assuming user John has id 1
            long executionTimeLocal = System.currentTimeMillis() - startTimeLocal;
            System.out.println("Execution Time with local database: " + executionTimeLocal + " ms");
            System.out.println("Order created with ID: " + orderId);
```



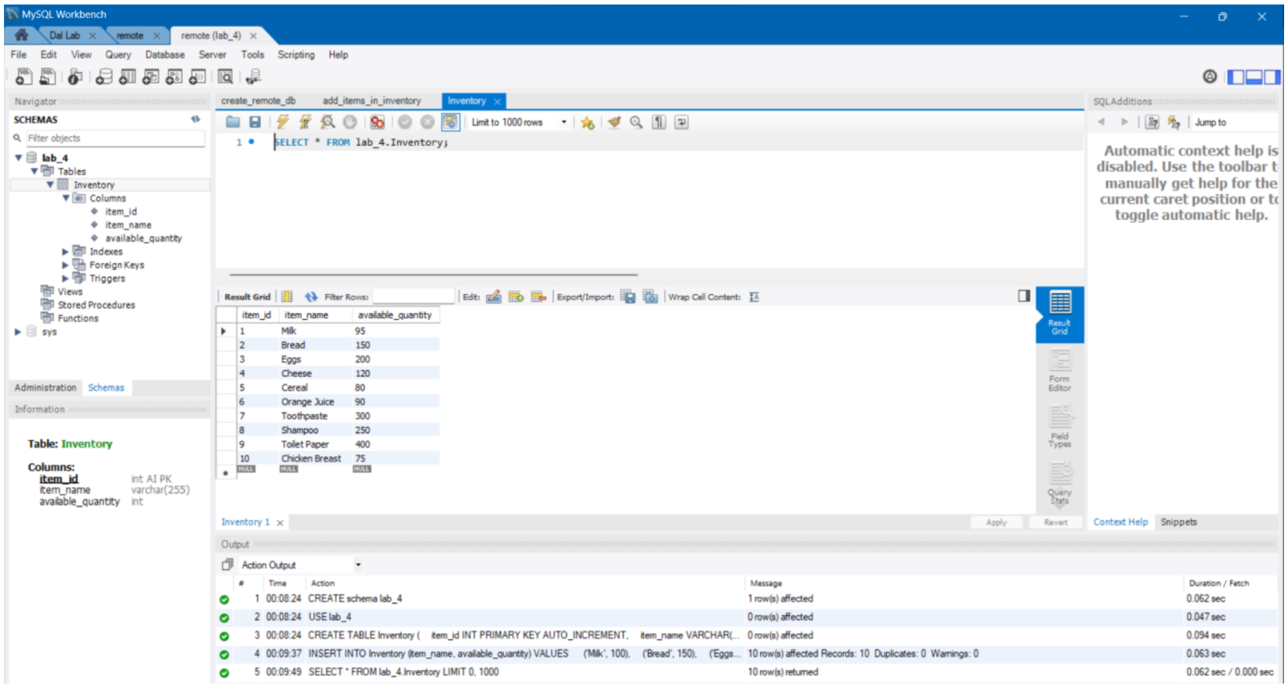
```java
            System.out.println("Execution Time with local database: " + executionTimeLocal + " ms");
            System.out.println("Order created with ID: " + orderId);

            long startTimeRemote = System.currentTimeMillis();
            remoteConnect = DriverManager.getConnection(REMOTE_DB_URL, REMOTE_DB_USER, REMOTE_DB_PASSWORD);
            updateQuantityRemoteDB(remoteConnect, itemName, quantity);
            long executionTimeStep3 = System.currentTimeMillis() - startTimeRemote;
            System.out.println("Execution Time with remote database: " + executionTimeStep3 + " ms");
            scanner.close();

        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            // Close connections
            try {
                if (remoteConnect != null)
                    remoteConnect.close();
                if (localConnect != null)
                    localConnect.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    private static int createOrderLocalDB(Connection connection, int userId, String itemName, int quantity) throws SQLException {
        PreparedStatement preparedStatement = connection.prepareStatement(
                sql:"INSERT INTO Order_info (user_id, item_name, quantity, order_date) VALUES (?, ?, ?, ?)",
                Statement.RETURN_GENERATED_KEYS);

        preparedStatement.setInt(parameterIndex:1, userId);
        preparedStatement.setString(parameterIndex:2, itemName);
        preparedStatement.setInt(parameterIndex:3, quantity);
        preparedStatement.setDate(parameterIndex:4, new Date(System.currentTimeMillis())); // Assuming the current date

        int affectedRows = preparedStatement.executeUpdate();
```



```java
    private static int createOrderLocalDB(Connection connection, int userId, String itemName, int quantity) throws SQLException {
        PreparedStatement preparedStatement = connection.prepareStatement(
                sql:"INSERT INTO Order_info (user_id, item_name, quantity, order_date) VALUES (?, ?, ?, ?)",
                Statement.RETURN_GENERATED_KEYS);

        preparedStatement.setInt(parameterIndex:1, userId);
        preparedStatement.setString(parameterIndex:2, itemName);
        preparedStatement.setInt(parameterIndex:3, quantity);
        preparedStatement.setDate(parameterIndex:4, new Date(System.currentTimeMillis())); // Assuming the current date

        int affectedRows = preparedStatement.executeUpdate();
        int orderId = -1;

        if (affectedRows > 0) {
            ResultSet generatedKeys = preparedStatement.getGeneratedKeys();
            if (generatedKeys.next()) {
                orderId = generatedKeys.getInt(columnIndex:1);
            }
        }

        return orderId;
    }

    private static void updateQuantityRemoteDB(Connection connection, String itemName, int quantity) throws SQLException {
        PreparedStatement preparedStatement = connection.prepareStatement(
                sql:"UPDATE Inventory SET available_quantity = available_quantity - ? WHERE item_name = ?");

        preparedStatement.setInt(parameterIndex:1, quantity);
        preparedStatement.setString(parameterIndex:2, itemName);

        preparedStatement.executeUpdate();
    }
}
```

Fig 7-9: Java solution code for problem statement.

Following is the snip of the inventory table before we run the code



```sql
SELECT * FROM lab_4.Inventory;
```

item_id	item_name	available_quantity
1	Milk	95
2	Bread	150
3	Eggs	200
4	Cheese	120
5	Cereal	80
6	Orange Juice	90
7	Toothpaste	300
8	Shampoo	250
9	Toilet Paper	400
10	Chicken Breast	75
NULL	NULL	NULL

Table: Inventory

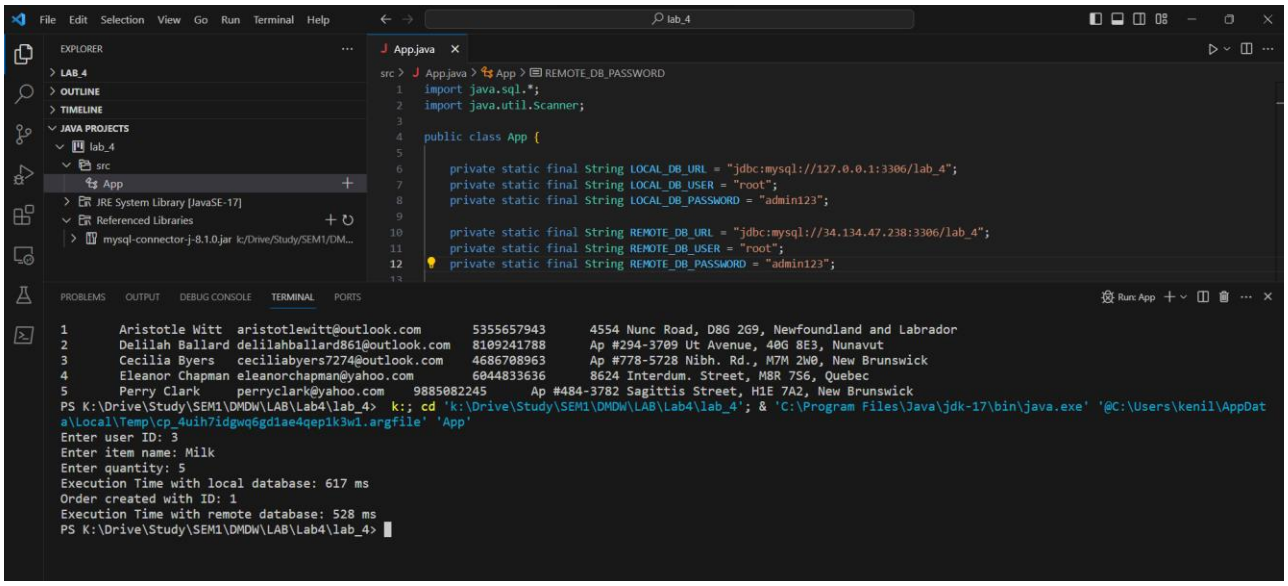
Columns:

item_id	int AI PK
item_name	varchar(255)
available_quantity	int

```
#  Time      Action                                                                                              Message                                                        Duration / Fetch
1  00:08:24  CREATE schema lab_4                                                                                 1 row(s) affected                                              0.062 sec
2  00:08:24  USE lab_4                                                                                           0 row(s) affected                                              0.047 sec
3  00:08:24  CREATE TABLE Inventory (    item_id INT PRIMARY KEY AUTO_INCREMENT,    item_name VARCHAR(...       0 row(s) affected                                              0.094 sec
4  00:09:37  INSERT INTO Inventory (item_name, available_quantity) VALUES    ('Milk', 100),    ('Bread', 150),    ('Eggs...   10 row(s) affected Records: 10  Duplicates: 0  Warnings: 0   0.063 sec
5  00:09:49  SELECT * FROM lab_4.Inventory LIMIT 0, 1000                                                         10 row(s) returned                                             0.062 sec / 0.000 sec
```

Fig 10: Initial value in inventory table at remote SQL instance

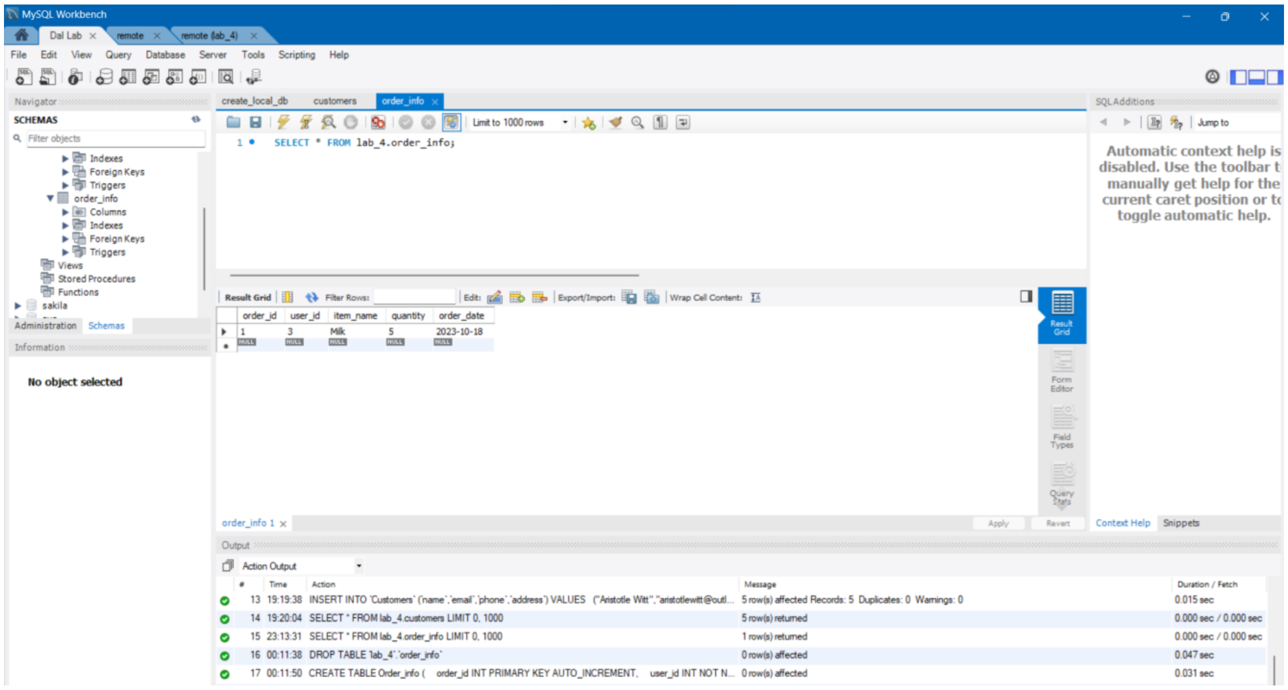
On running the java code we create an order in the local database and the output is shown below:



```java
import java.sql.*;
import java.util.Scanner;

public class App {

    private static final String LOCAL_DB_URL = "jdbc:mysql://127.0.0.1:3306/lab_4";
    private static final String LOCAL_DB_USER = "root";
    private static final String LOCAL_DB_PASSWORD = "admin123";

    private static final String REMOTE_DB_URL = "jdbc:mysql://34.134.47.238:3306/lab_4";
    private static final String REMOTE_DB_USER = "root";
    private static final String REMOTE_DB_PASSWORD = "admin123";
```

```
1       Aristotle Witt  aristotlewitt@outlook.com       5355657943      4554 Nunc Road, D8G 2G9, Newfoundland and Labrador
2       Delilah Ballard delilahballard861@outlook.com   8109241788      Ap #294-3709 Ut Avenue, 40G 8E3, Nunavut
3       Cecilia Byers   ceciliabyers7274@outlook.com    4686708963      Ap #778-5728 Nibh. Rd., M7M 2W0, New Brunswick
4       Eleanor Chapman eleanorchapman@yahoo.com        6044833636      8624 Interdum. Street, M8R 7S6, Quebec
5       Perry Clark     perryclark@yahoo.com    9885082245      Ap #484-3782 Sagittis Street, H1E 7A2, New Brunswick
PS K:\Drive\Study\SEM1\DMDW\LAB\Lab4\lab_4>  k:; cd 'k:\Drive\Study\SEM1\DMDW\LAB\Lab4\lab_4'; & 'C:\Program Files\Java\jdk-17\bin\java.exe' '@C:\Users\kenil\AppDat
a\Local\Temp\cp_4uih7idgwq6gd1ae4qep1k3w1.argfile' 'App'
Enter user ID: 3
Enter item name: Milk
Enter quantity: 5
Execution Time with local database: 617 ms
Order created with ID: 1
Execution Time with remote database: 528 ms
PS K:\Drive\Study\SEM1\DMDW\LAB\Lab4\lab_4>
```

Fig 11: Java code output on creating an order in the local database

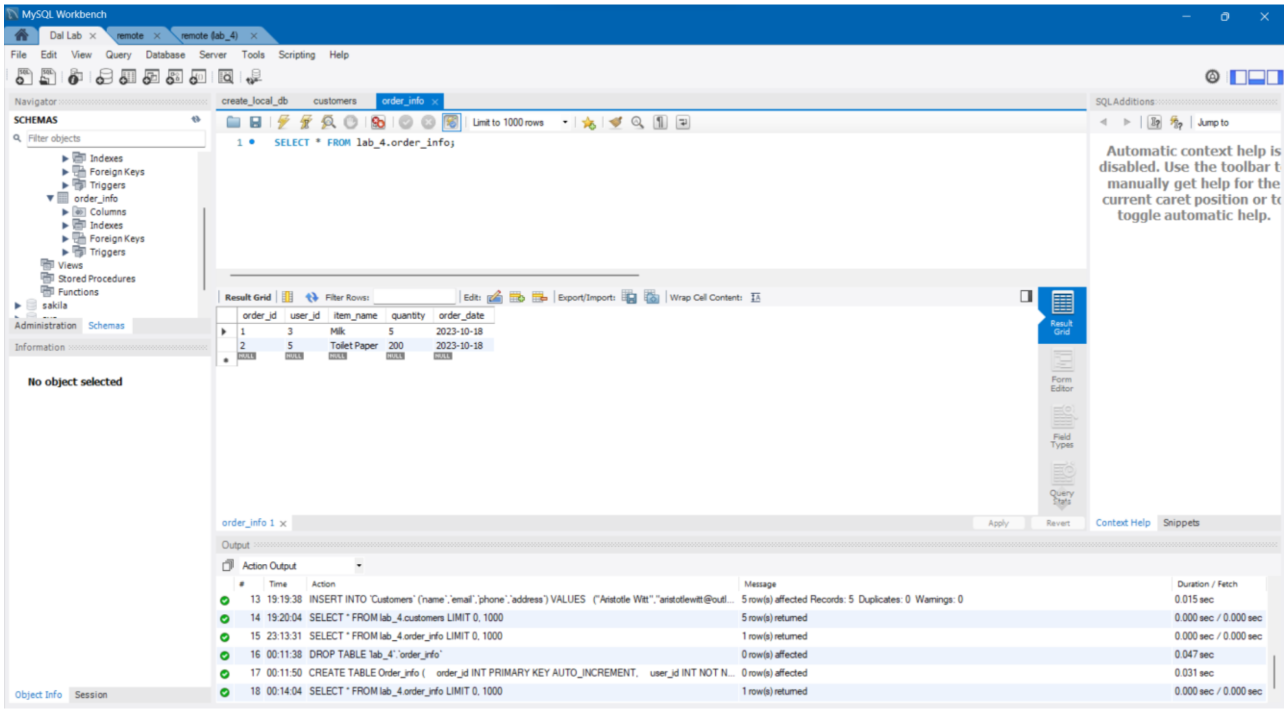


```sql
SELECT * FROM lab_4.order_info;
```

order_id	user_id	item_name	quantity	order_date
1	3	Milk	5	2023-10-18
NULL	NULL	NULL	NULL	NULL

#	Time	Action	Message	Duration / Fetch
13	19:19:38	INSERT INTO `Customers` (`name`,`email`,`phone`,`address`) VALUES ("Aristotle Witt","aristotlewitt@outl...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.015 sec
14	19:20:04	SELECT * FROM lab_4.customers LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
15	23:13:31	SELECT * FROM lab_4.order_info LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
16	00:11:38	DROP TABLE `lab_4`.`order_info`	0 row(s) affected	0.047 sec
17	00:11:50	CREATE TABLE Order_info (order_id INT PRIMARY KEY AUTO_INCREMENT, user_id INT NOT N...	0 row(s) affected	0.031 sec

Fig 12: Order created in Order_info table in local database.

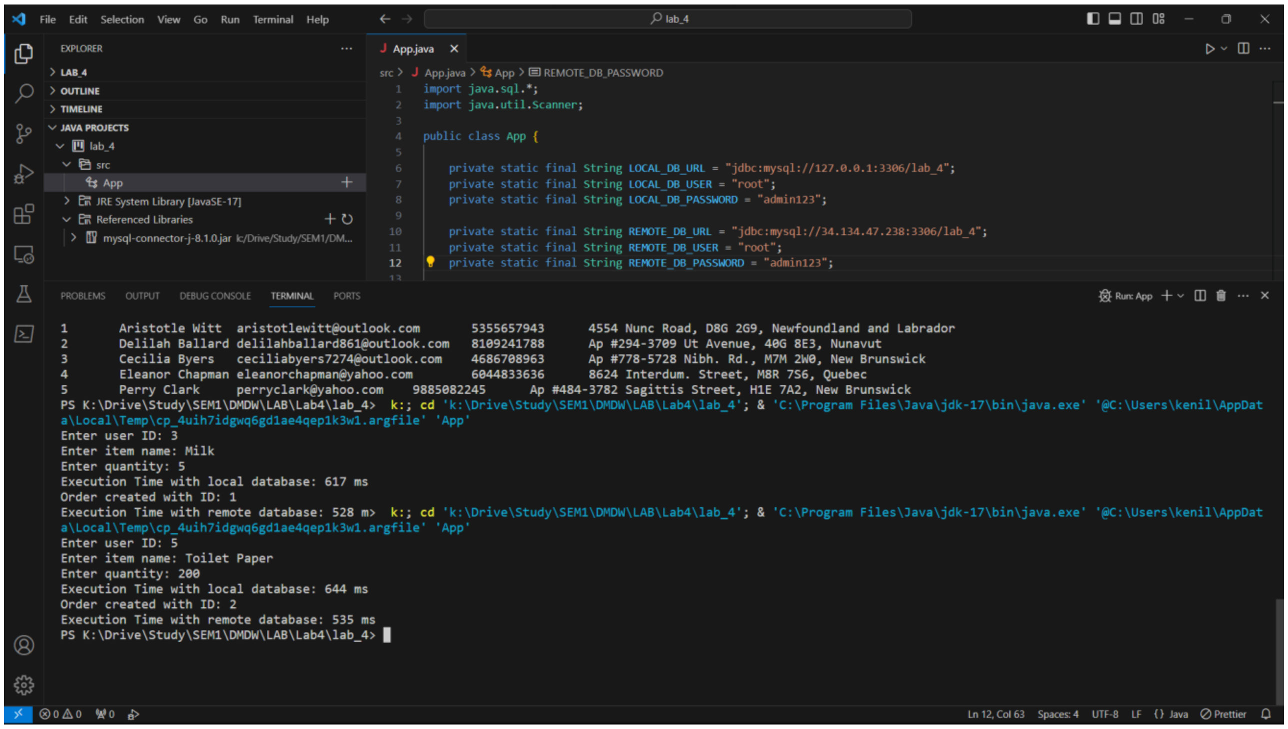


```sql
SELECT * FROM lab_4.order_info;
```

order_id	user_id	item_name	quantity	order_date
1	3	Milk	5	2023-10-18
2	5	Toilet Paper	200	2023-10-18
NULL	NULL	NULL	NULL	NULL

```
#   Time      Action                                                                            Message                                                      Duration / Fetch
13  19:19:38  INSERT INTO `Customers` (`name`,`email`,`phone`,`address`) VALUES ("Aristotle Witt","aristotlewitt@outl...  5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0  0.015 sec
14  19:20:04  SELECT * FROM lab_4.customers LIMIT 0, 1000                                       5 row(s) returned                                            0.000 sec / 0.000 sec
15  23:13:31  SELECT * FROM lab_4.order_info LIMIT 0, 1000                                      1 row(s) returned                                            0.000 sec / 0.000 sec
16  00:11:38  DROP TABLE `lab_4`.`order_info`                                                   0 row(s) affected                                            0.047 sec
17  00:11:50  CREATE TABLE Order_info (    order_id INT PRIMARY KEY AUTO_INCREMENT,    user_id INT NOT N...  0 row(s) affected                             0.031 sec
18  00:14:04  SELECT * FROM lab_4.order_info LIMIT 0, 1000                                      1 row(s) returned                                            0.000 sec / 0.000 sec
```

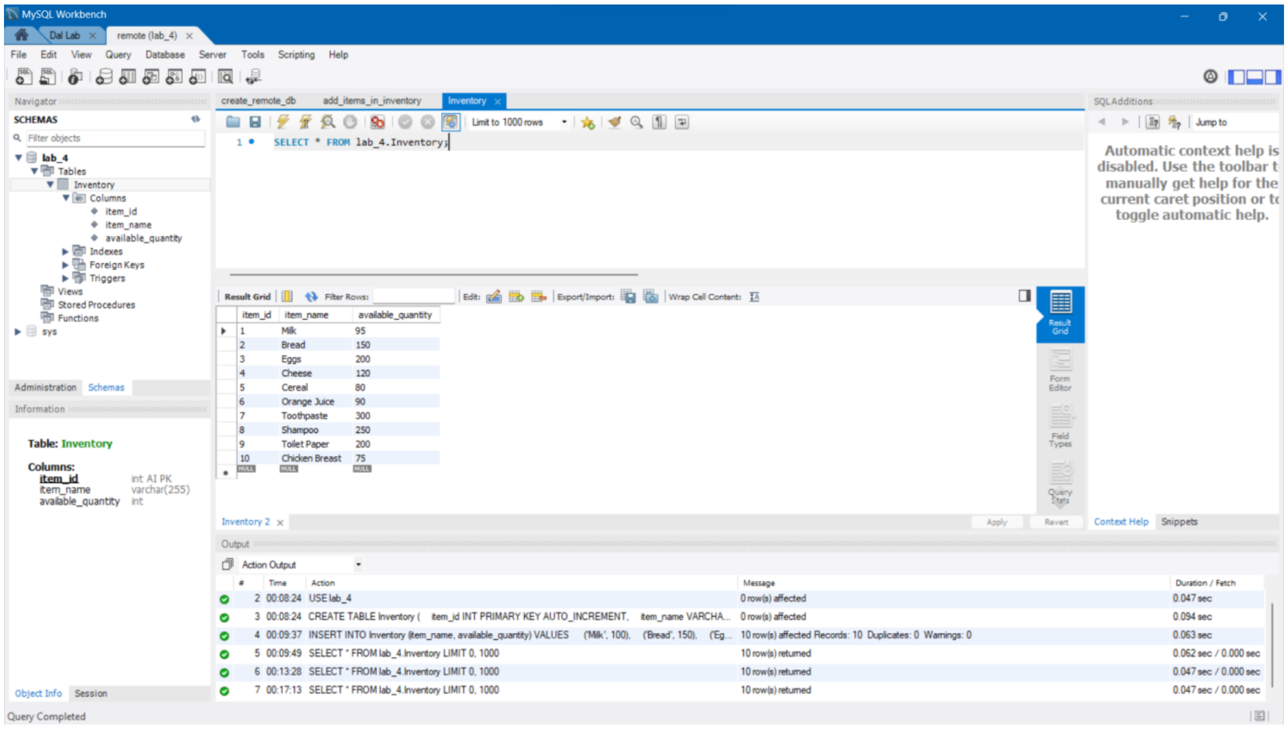
Fig 13: order_info table on creating second order



```java
import java.sql.*;
import java.util.Scanner;

public class App {

    private static final String LOCAL_DB_URL = "jdbc:mysql://127.0.0.1:3306/lab_4";
    private static final String LOCAL_DB_USER = "root";
    private static final String LOCAL_DB_PASSWORD = "admin123";

    private static final String REMOTE_DB_URL = "jdbc:mysql://34.134.47.238:3306/lab_4";
    private static final String REMOTE_DB_USER = "root";
    private static final String REMOTE_DB_PASSWORD = "admin123";
```

```
1       Aristotle Witt  aristotlewitt@outlook.com       5355657943      4554 Nunc Road, D8G 2G9, Newfoundland and Labrador
2       Delilah Ballard delilahballard861@outlook.com   8109241788      Ap #294-3709 Ut Avenue, 40G 8E3, Nunavut
3       Cecilia Byers   ceciliabyers7274@outlook.com    4686708963      Ap #778-5728 Nibh. Rd., M7M 2W0, New Brunswick
4       Eleanor Chapman eleanorchapman@yahoo.com        6044833636      8624 Interdum. Street, M8R 7S6, Quebec
5       Perry Clark     perryclark@yahoo.com    9885082245      Ap #484-3782 Sagittis Street, H1E 7A2, New Brunswick
PS K:\Drive\Study\SEM1\DMDW\LAB\Lab4\lab_4>  k:; cd 'k:\Drive\Study\SEM1\DMDW\LAB\Lab4\lab_4'; & 'C:\Program Files\Java\jdk-17\bin\java.exe' '@C:\Users\kenil\AppDat
a\Local\Temp\cp_4uih7idgwq6gd1ae4qep1k3w1.argfile' 'App'
Enter user ID: 3
Enter item name: Milk
Enter quantity: 5
Execution Time with local database: 617 ms
Order created with ID: 1
Execution Time with remote database: 528 m>  k:; cd 'k:\Drive\Study\SEM1\DMDW\LAB\Lab4\lab_4'; & 'C:\Program Files\Java\jdk-17\bin\java.exe' '@C:\Users\kenil\AppDat
a\Local\Temp\cp_4uih7idgwq6gd1ae4qep1k3w1.argfile' 'App'
Enter user ID: 5
Enter item name: Toilet Paper
Enter quantity: 200
Execution Time with local database: 644 ms
Order created with ID: 2
Execution Time with remote database: 535 ms
PS K:\Drive\Study\SEM1\DMDW\LAB\Lab4\lab_4>
```

Fig 14: Java output on creating another order



```
SELECT * FROM lab_4.Inventory;
```

item_id	item_name	available_quantity
1	Milk	95
2	Bread	150
3	Eggs	200
4	Cheese	120
5	Cereal	80
6	Orange Juice	90
7	Toothpaste	300
8	Shampoo	250
9	Toilet Paper	200
10	Chicken Breast	75
NULL	NULL	NULL

```
#  Time      Action                                                                                     Message                                                  Duration / Fetch
2  00:08:24  USE lab_4                                                                                  0 row(s) affected                                        0.047 sec
3  00:08:24  CREATE TABLE Inventory (    item_id INT PRIMARY KEY AUTO_INCREMENT,    item_name VARCHA...  0 row(s) affected                                        0.094 sec
4  00:09:37  INSERT INTO Inventory (item_name, available_quantity) VALUES    ('Milk', 100),   ('Bread', 150),   ('Eg...  10 row(s) affected Records: 10  Duplicates: 0  Warnings: 0  0.063 sec
5  00:09:49  SELECT * FROM lab_4.Inventory LIMIT 0, 1000                                                10 row(s) returned                                       0.062 sec / 0.000 sec
6  00:13:28  SELECT * FROM lab_4.Inventory LIMIT 0, 1000                                                10 row(s) returned                                       0.047 sec / 0.000 sec
7  00:17:13  SELECT * FROM lab_4.Inventory LIMIT 0, 1000                                                10 row(s) returned                                       0.047 sec / 0.000 sec
```

Table: Inventory

Columns:
item_id int AI PK
item_name varchar(255)
available_quantity int

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

Query Completed

Fig 15: Final values in the inventory table at the remote database

In my code local database takes more time to execute queries than the remote database because of hardware I am using on the local database is of less configuration and low an end laptop. Whereas remote database server has good hardware specifications on Google Cloud.

References

[1] Mysql.com. [Online]. Available: https://dev.mysql.com/doc/workbench/en/wb-forward-engineering-live-server.html. [Accessed: 18-Oct-2023].

[2] “Google Cloud Platform,” Google.com. [Online]. Available: https://cloud.google.com/?hl=en. [Accessed: 18-Oct-2023].