**LAB 4**

Abhisha Thaker B00937694

Github repo link for java code:

**1. Create a local database with–**

**• User table (attributes – id,name,email,phone,address)**

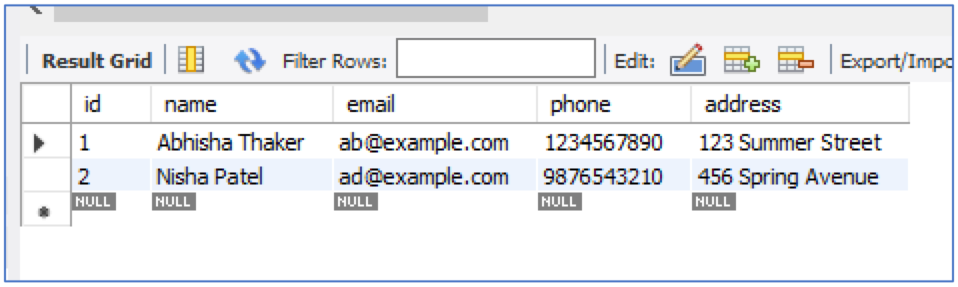
**• Order_info table (order_id, user_id,item_name, quantity,order_date)**

```
CREATE SCHEMA ecommerce;
USE ecommerce;
CREATE TABLE User (
   id INTEGER PRIMARY KEY,
   name VARCHAR(20),
   email VARCHAR(50),
   phone VARCHAR(20),
   address VARCHAR(200)
);

CREATE TABLE Order_info (
   order_id INT PRIMARY KEY,
   user_id INT,
   item_name VARCHAR(255),
   quantity INT,
   order_date DATE
);
```

INSERT INTO User (id, name, email, phone, address) VALUES (1, 'Abhisha Thaker', 'ab@example.com', '1234567890', '123 Summer Street');

INSERT INTO User (id, name, email, phone, address) VALUES (2, 'Nisha Patel', 'ad@example.com', '9876543210', '456 Spring Avenue');

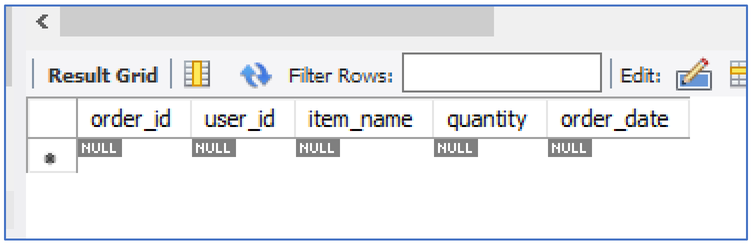
select * from User;



| id | name | email | phone | address |
|---|---|---|---|---|
| 1 | Abhisha Thaker | ab@example.com | 1234567890 | 123 Summer Street |
| 2 | Nisha Patel | ad@example.com | 9876543210 | 456 Spring Avenue |
| NULL | NULL | NULL | NULL | NULL |

User#1

Select * from order_info;



| order_id | user_id | item_name | quantity | order_date |
|---|---|---|---|---|
| NULL | NULL | NULL | NULL | NULL |

Order_info#1

**2. Create a remote database in GCP with – • Inventory table (item_id,item_name, available_quantity)**

CREATE SCHEMA INVENTORY;

use INVENTORY;

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

);

INSERT INTO Inventory (item_id, item_name, available_quantity)

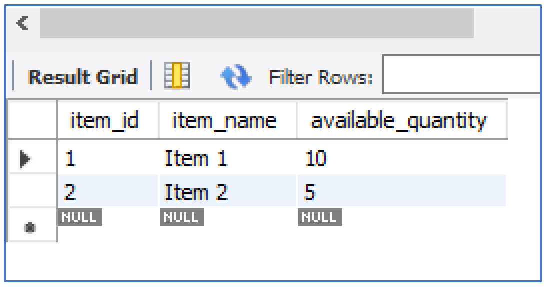
VALUES (1, 'Item 1', 10);

-- Insert multiple rows using a single INSERT statement

INSERT INTO Inventory (item_id, item_name, available_quantity)

VALUES (2, 'Item 2', 5);

select * from Inventory;



| | item_id | item_name | available_quantity |
|---|---|---|---|
| ▶ | 1 | Item 1 | 10 |
| | 2 | Item 2 | 5 |
| * | NULL | NULL | NULL |

inventory#1

**Write a Python/Java/any language program that –**

**• Fetches item details from the remote database**

**• Creates an order in local database**

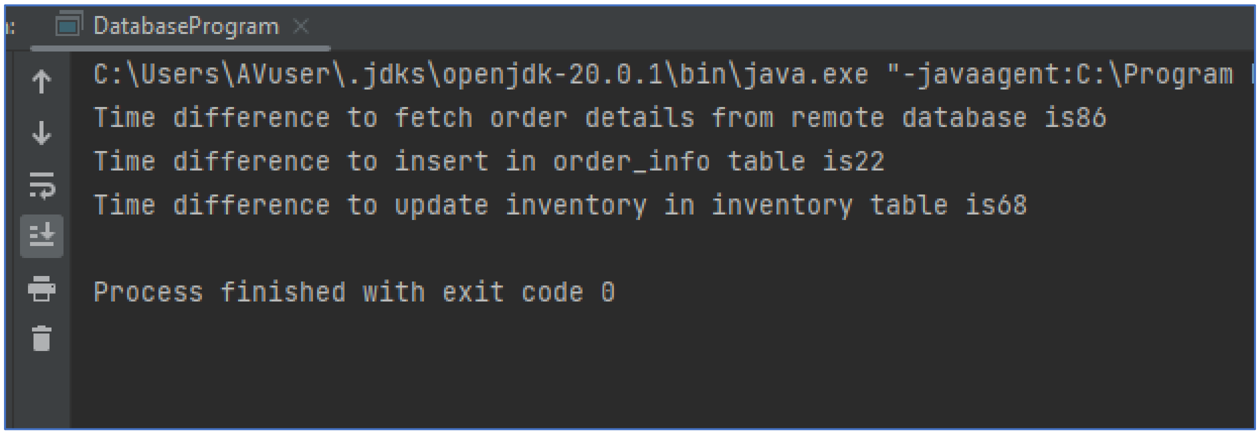
**• Writes the updated quantity back to the remote database upon order creation**

```
package org.example;

// Press Shift twice to open the Search Everywhere dialog
and type `show whitespaces`,
// then press Enter. You can now see whitespace characters
in your code.
import java.sql.*;
import java.util.Scanner;

public class DatabaseProgram {
    public static void main(String[] args) {
        try {
            // Connect to the local database
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter the local username");
            String localUsername = sc.next();
            System.out.println("Enter the local password");
            String localPassword = sc.next();
            Connection localConnect =
DriverManager.getConnection("jdbc:mysql://localhost:3306/ec
ommerce", localUsername, localPassword);

            // Connect to the remote database in GCP
            System.out.println("Enter the remote
username");
            String remoteUsername = sc.next();
            System.out.println("Enter the remote
password");
            String remotePassword = sc.next();
            Connection remoteConnect =
DriverManager.getConnection("jdbc:mysql://35.184.144.54:330
6/INVENTORY", remoteUsername, remotePassword);

            /*
            Fetch Item details from the Inventory table in
the remote database
            and display the time it took to carry out this
operation
             */
            // Fetch item details from the remote database
            Statement remoteStatement =
remoteConnect.createStatement();

            // set the start time before running the first
query
            long startTime = System.currentTimeMillis();
            ResultSet remoteResultSet =
remoteStatement.executeQuery("SELECT item_id, item_name,
available_quantity FROM Inventory");
            // log the end time after running the query
            long endTime = System.currentTimeMillis();

            long timediff = endTime - startTime;
            System.out.println("Time difference to fetch
order details from remote database is"+""+timediff);
```

```
            /*
            Insert Order_Info details in the Order_Info
table in the local database
            and display the time it took to carry out this
operation
             */
            // Create an order in the local database
            Statement localStatement =
localConnect.createStatement();

            String orderQuery = "INSERT INTO Order_info
(order_id, user_id, item_name, quantity, order_date) VALUES
(4, 1, 'Product A', 2, '2023-06-23')";

            startTime = System.currentTimeMillis();
            localStatement.executeUpdate(orderQuery);
            endTime = System.currentTimeMillis();
            timediff = endTime - startTime;
            System.out.println("Time difference to insert
in order_info table is"+""+timediff);


            // Write the updated quantity back to the
remote database

            String updateQuery = "UPDATE Inventory SET
available_quantity = 4 WHERE item_id = 1";
            Statement updateStatement =
remoteConnect.createStatement();

            startTime = System.currentTimeMillis();
            updateStatement.executeUpdate(updateQuery);
            endTime = System.currentTimeMillis();
            timediff = endTime - startTime;
            System.out.println("Time difference to update
inventory in inventory table is"+""+timediff);

            // Close the database connections
            remoteResultSet.close();
            remoteStatement.close();
            updateStatement.close();

            localStatement.close();
```

```
                remoteConnect.close();
                localConnect.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
}
}
```

Print query execution time at every step (try to deduce the reason behind the time differences)



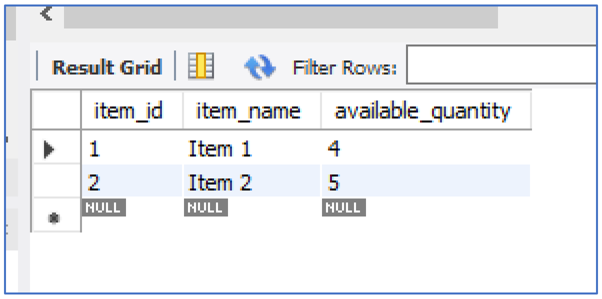
```
C:\Users\AVuser\.jdks\openjdk-20.0.1\bin\java.exe "-javaagent:C:\Program
Time difference to fetch order details from remote database is86
Time difference to insert in order_info table is22
Time difference to update inventory in inventory table is68

Process finished with exit code 0
```

Java_output

Time difference is the least in inserting in the order_info table which is in the local database. Because local database requires less time in connection and subsequent handshakes. While fetching and updating details in the remote database has an overhead because it's located remotely, so due to time consumed by network latency and due to connection handshakes that happen over remote network, that's the reason why it took less time ( ie. 22 ms ) in local database while 86 ms and 68 ms in remote database.

After running the Java Code, write the following commands in mysqlworkbench just to doublecheck

select * from Inventory;



| | item_id | item_name | available_quantity |
|---|---|---|---|
| ▶ | 1 | Item 1 | 4 |
| | 2 | Item 2 | 5 |
| ✱ | NULL | NULL | NULL |