CSCI 5408

DATA MANAGEMENT AND WAREHOUSING

LAB ASSIGNMENT - 3

Banner ID: B00952865

GitLab Assignment Link:
https://git.cs.dal.ca/apurohit/CSCI5408_F23_B00952865_AdityaMaheshbhai_Purohit/-/tree/main/Lab3

Table of Contents

Create Tables: Customer, Account, Transfer

Queries:

```
CREATE DATABASE lab3_transactions;
USE lab3_transactions;

CREATE TABLE customer(
    c_id INT PRIMARY KEY,
    name VARCHAR(30),
    mailing_address VARCHAR(200),
    permanent_address VARCHAR(200),
    email VARCHAR(50) NOT NULL UNIQUE,
    primary_phone_number VARCHAR(10) NOT NULL UNIQUE);

CREATE TABLE account(
    acc_number INT PRIMARY KEY,
    c_id INT,
    acc_balance DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (c_id) REFERENCES customer(c_id));

CREATE TABLE transfer(
    transfer_id INT PRIMARY KEY,
    sender_acc_number INT,
    receiver_acc_number INT,
    date_of_transfer DATE,
    recipient_name VARCHAR(100),
    status VARCHAR(10),
    FOREIGN KEY (sender_acc_number) REFERENCES account(acc_number),
```

FOREIGN KEY (receiver_acc_number) REFERENCES account(acc_number));

INSERT INTO customer VALUES(1, 'Aditya', 'Halifax', 'Halifax', 'aditya.purohit@dal.ca', 9876543219);

INSERT INTO account VALUES(1, 1, 100.00);

INSERT INTO customer VALUES(2, 'Joy', 'Halifax', 'Halifax', 'joy@gmail.com', 7776543219);

INSERT INTO account VALUES(2, 2, 50.00);

SELECT * FROM CUSTOMER;

SELECT * FROM ACCOUNT;

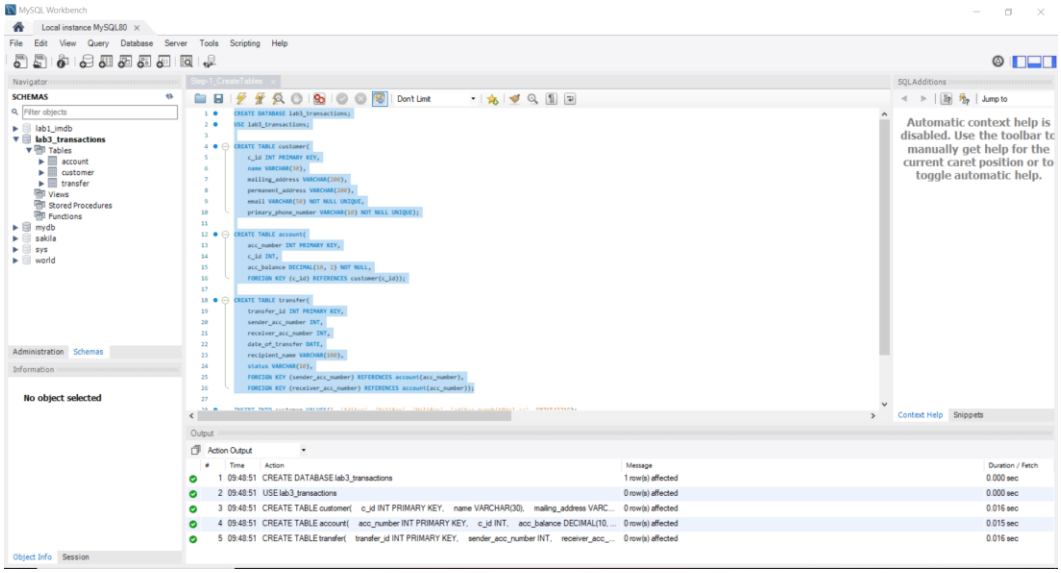

Figure 1: Creating 3 Tables [1]

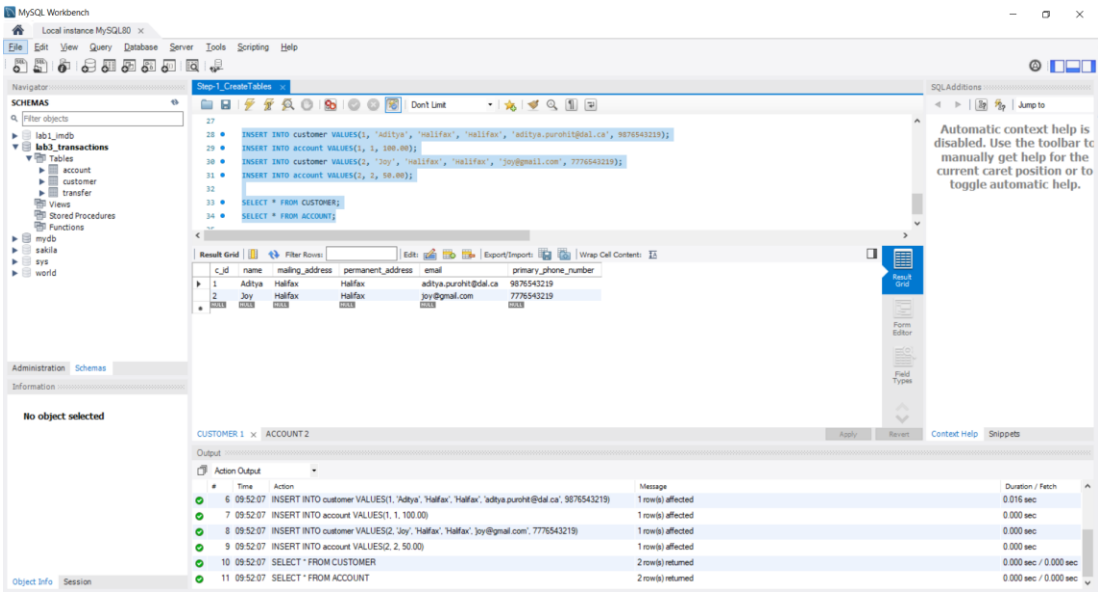

Figure 2: Inserting 2 Customer Records, 2 Accounts, Display customer records. [1]

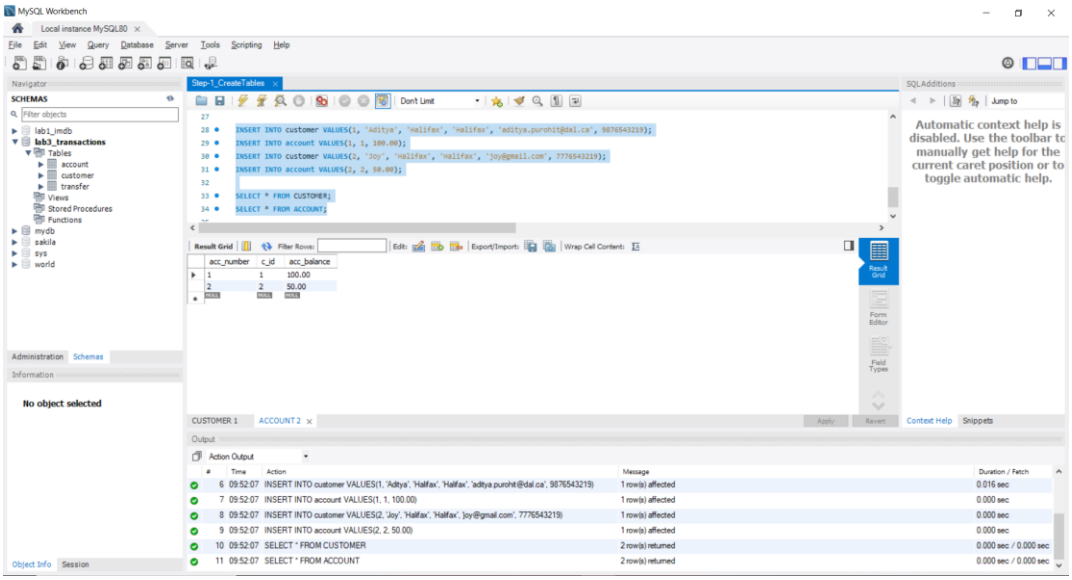

Figure 3: Display account records. [1]

Explanation:

I have created 3 tables and also inserted 2 customers and 2 accounts, so that the transfer transaction can take place in next step. The new data of both tables can be seen as the result of SELECT query.

Transaction Case 1 Assuming Business logic updates status to ‘accepted’:

Query:

```
SET autocommit=0;


-- Transfer 30 Dollars from Account number 1 to Account number 2. Assuming Business Logic will set status to accepted.

START TRANSACTION;


SAVEPOINT before_update_account;

-- Debit account number 1.

UPDATE account

SET acc_balance = acc_balance - 30 WHERE acc_number = 1;


SAVEPOINT before_insert_transfer;

-- Insert a new Transfer record, with waiting status.

INSERT INTO transfer values(435, 1, 2, '2023-11-04', 'Joy', 'waiting');


-- Assuming some business logic sets the transfer status to 'accepted'

UPDATE transfer

SET status = 'accepted'

WHERE transfer_id=435;


-- Credit receiver with 30 dollars and commit Transaction as transfer status is accepted.

UPDATE account

SET acc_balance = acc_balance + 30 WHERE acc_number = 2;


-- Commit these all updates, as transfer was accepted.
```

COMMIT;

Step-1: Running the transaction partial to show that the commit is turned off and changes aren't reflected yet in actual database:

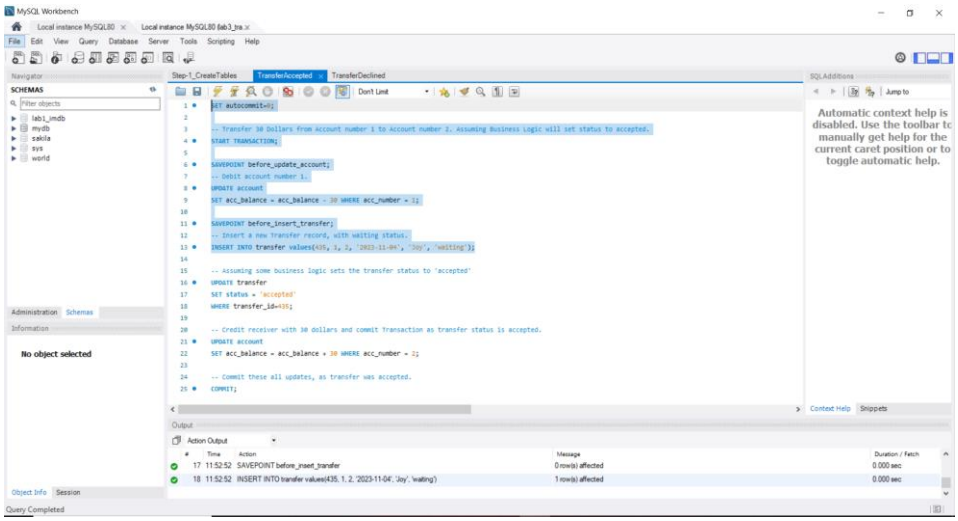

Figure 4: Executed part-1 of transaction: debit account & insert transfer record. [1]

Step-2: Now, I have opened new tab of MySQL instance in workbench to check records in account & transfer table. It can be seen that account table has still 100 dollars for aditya customer and transfer table has no entry yet. This is due to pending commit.

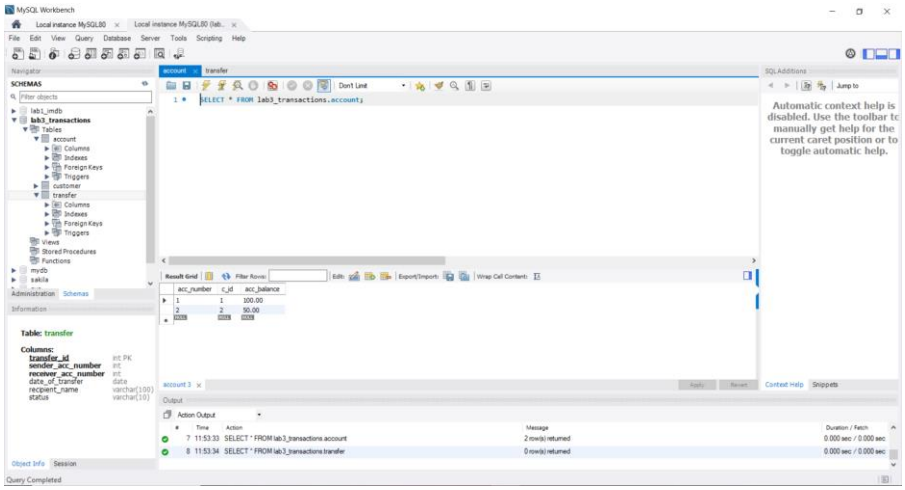

Figure 5: No change in account table. [1]

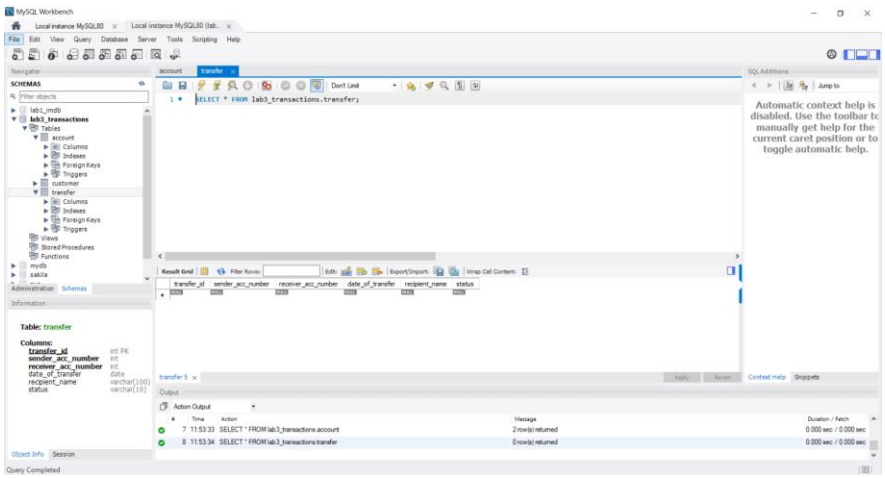

Figure 6: No change in transfer table. [1]

Step-3: But now, when I run the part-2 of this transaction, it can be seen that commit has taken place and both the tables are updated.

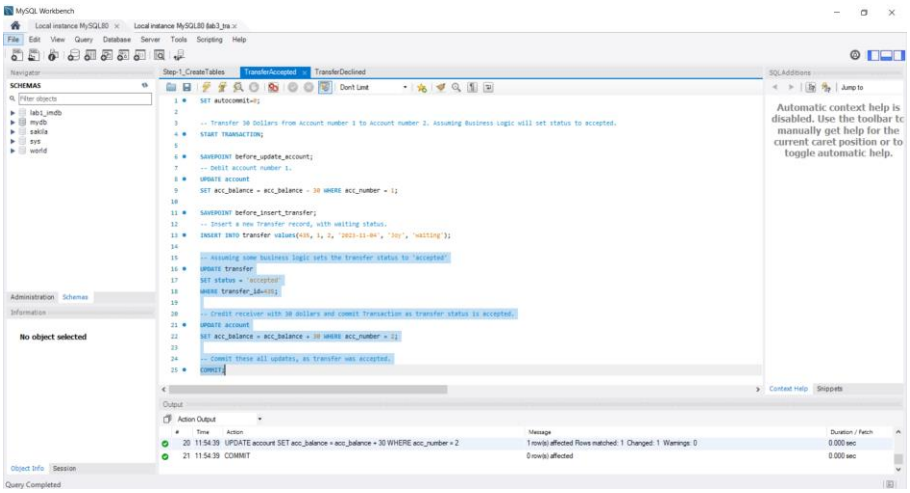

Figure 7: Executed part-2 of transaction: business logic, credit, update status & commit. [1]

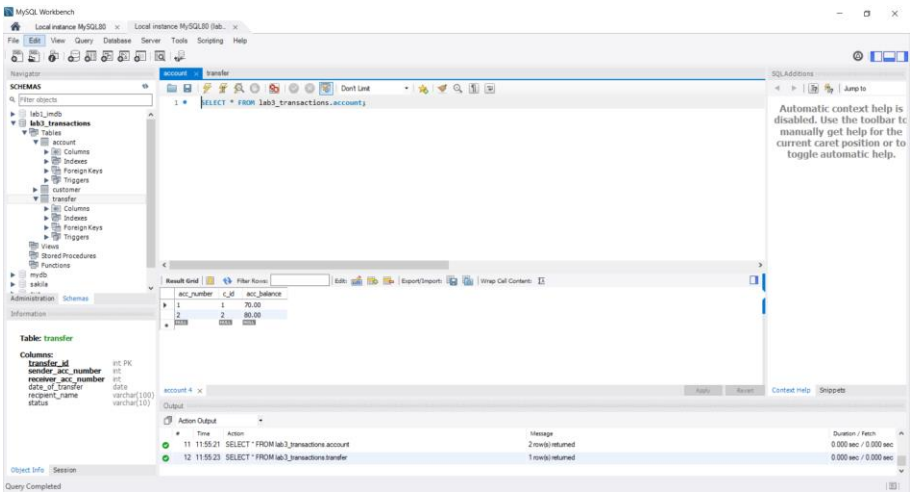

Figure 8: Account table has updated balances now. [1]

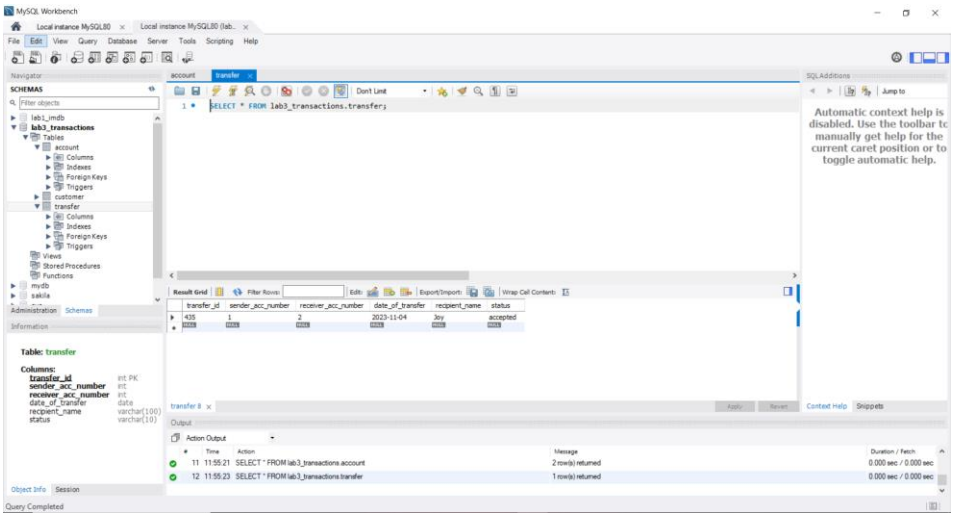

Figure 9: Transfer table has a new entry with accepted status. [1]

Transaction Case 2 Assuming Business logic updates status to ‘declined:

Before running the transaction case 2, I ran create table queries again with drop database, just to reset the original state which was affected by the case 1 execution.

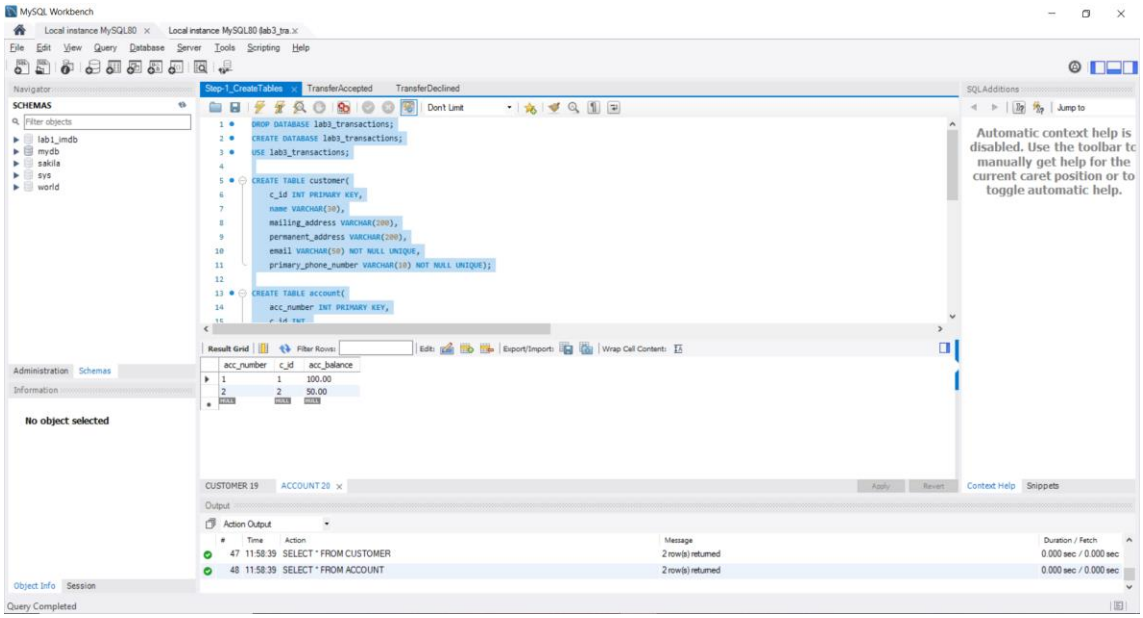

Figure 10: Resetting the original database state. [1]

Query:

```
SET autocommit=0;


-- Transfer 30 Dollars from Account number 1 to Account number 2. Assuming Business Logic will set status to declined.

START TRANSACTION;


SAVEPOINT before_update_account;

-- Debit account number 1.

UPDATE account

SET acc_balance = acc_balance - 30 WHERE acc_number = 1;


SAVEPOINT before_insert_transfer;

-- Insert a new Transfer record, with waiting status.
```

INSERT INTO transfer values(435, 1, 2, '2023-11-04', 'Joy', 'waiting');

-- Assuming some business logic sets the transfer status to 'declined'

UPDATE transfer

SET status = 'declined'

WHERE transfer_id=435;

-- Rollback to savepoint, to undo the debit as transfer was declined.

ROLLBACK TO before_update_account;

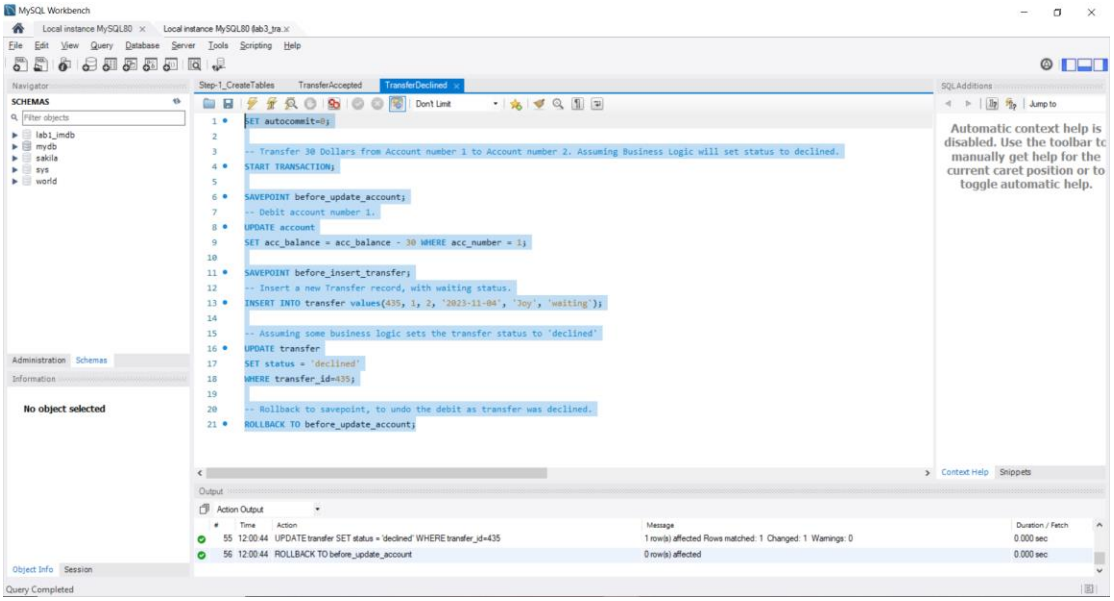

Figure 11: Running the whole transaction case-2 [1]

Here, there would be no updates in the transfer or account tables as there a rollback to savepoint happening due to the declined status set by business logic.

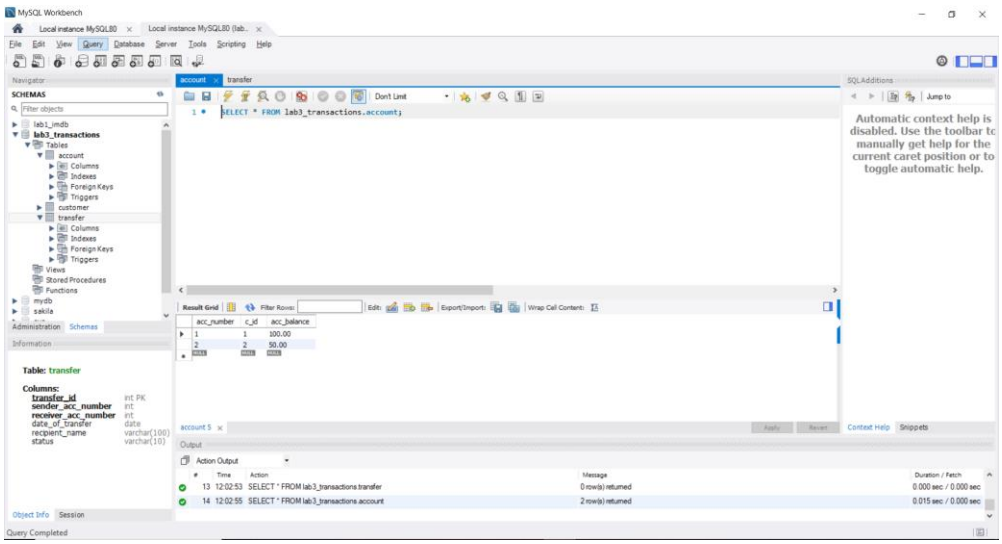

Figure 12: No Updates in Account Table. [1]

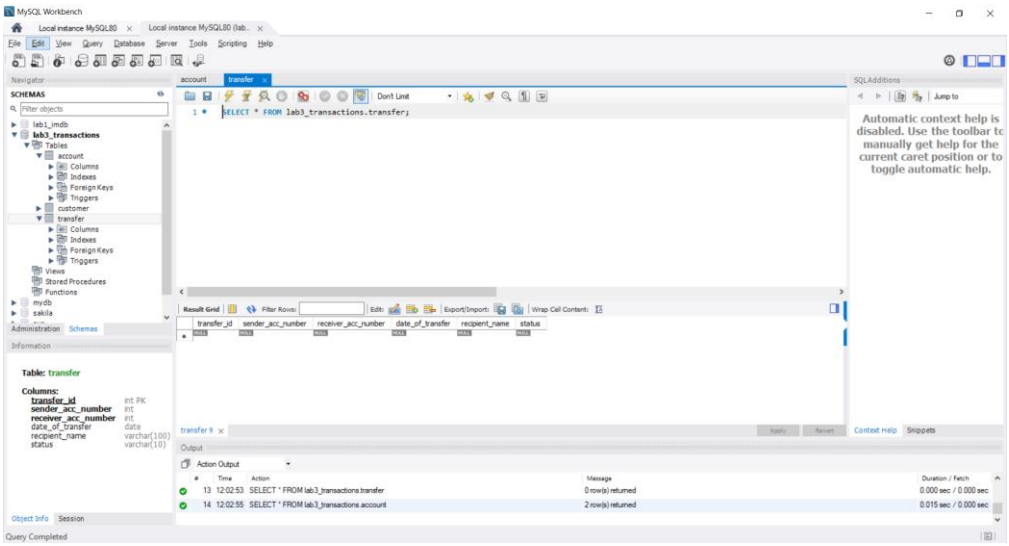

Figure 13: No Updates in Transfer Table. [1]

References:

[1] “MySQL Workbench",” MySQL, https://www.mysql.com/products/workbench/ (accessed Oct. 04, 2023).