CSCI-5408

DATA MANAGEMENT, WAREHOUSING, & ANALYTICS

LAB ASSIGNMENT - 3

Banner ID: B00977669

GitLab Assignment Link:
https://git.cs.dal.ca/saji/csci5408_w24_b00977669_christin_saji

Table of Contents

Task 1......3
Task 2......6
Task 3......8

Task: Design a banking application database for managing customer’s debit and credit.

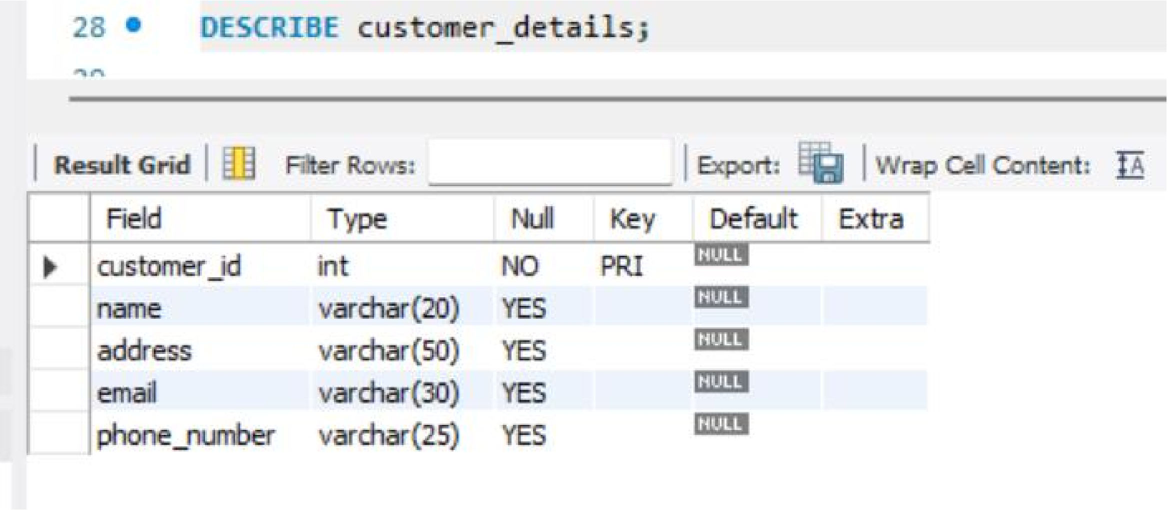
Task 1:

Design a banking application database with the **tables**:

- **Customer details** with attributes: customer id, name, address, email, phone number
- **Account details** with attributes: accountnumber, account balance
- **Account transfer details** with attributes: sender account number, date of transfer, recipient account number, transaction status (could be waiting/accepted/declined).

QUERY:

```
CREATE TABLE customer_details (

   customer_id INT PRIMARY KEY,

   name VARCHAR(20),

   address VARCHAR(50),

   email VARCHAR(30),

   phone_number VARCHAR(25)

);
```

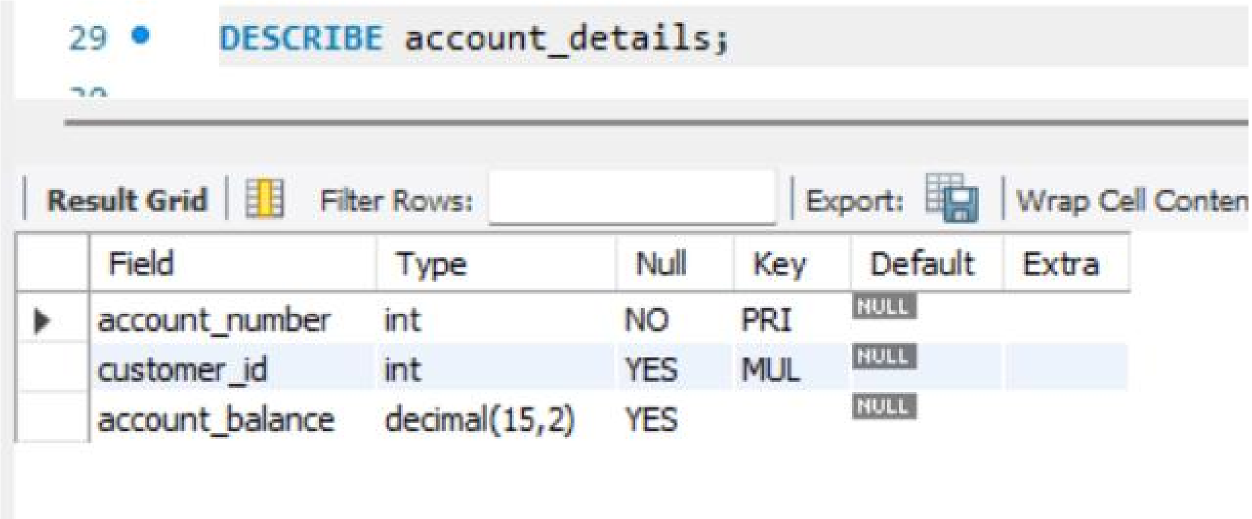


```
DESCRIBE customer_details;
```

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	NULL	
name	varchar(20)	YES		NULL	
address	varchar(50)	YES		NULL	
email	varchar(30)	YES		NULL	
phone_number	varchar(25)	YES		NULL	

Figure 1 customer_details Table Information

```
CREATE TABLE account_details (
    account_number INT PRIMARY KEY,
    customer_id INT,
    account_balance DECIMAL(15,2),
    FOREIGN KEY (customer_id) REFERENCES customer_details(customer_id)
);
```

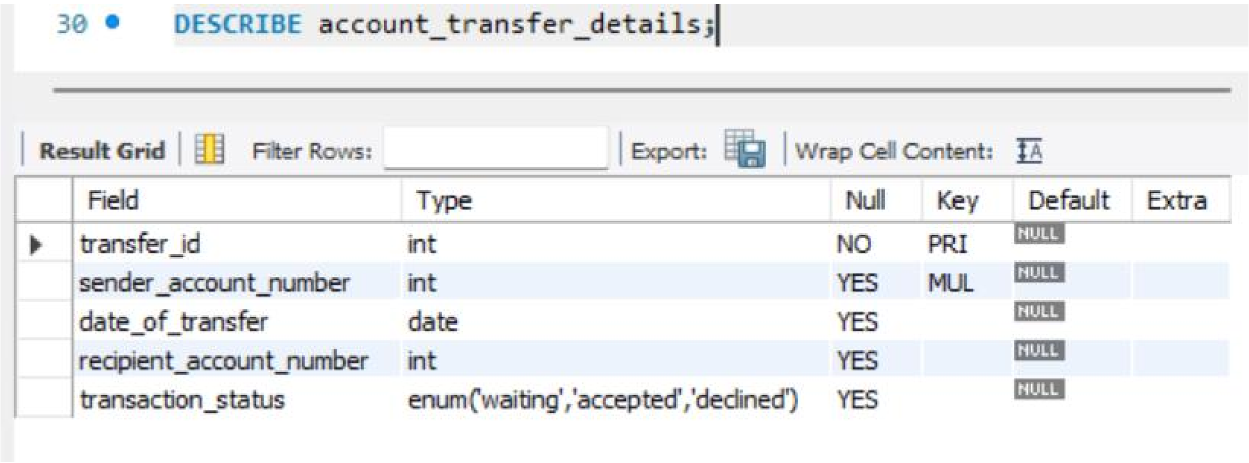


```
DESCRIBE account_details;
```

Field	Type	Null	Key	Default	Extra
account_number	int	NO	PRI	NULL	
customer_id	int	YES	MUL	NULL	
account_balance	decimal(15,2)	YES		NULL	

Figure 2 account_details Table Information

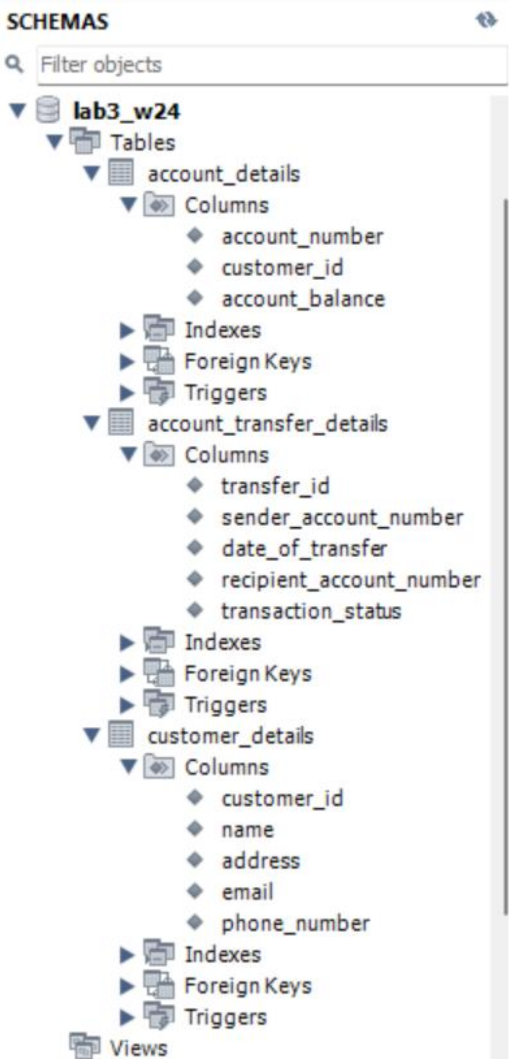
```
CREATE TABLE account_transfer_details (
    transfer_id INT PRIMARY KEY,
    sender_account_number INT,
    date_of_transfer DATE,
    recipient_account_number INT,
    transaction_status ENUM('waiting', 'accepted', 'declined'),
    FOREIGN KEY (sender_account_number) REFERENCES account_details(account_number)
);
```



```sql
DESCRIBE account_transfer_details;
```

Field	Type	Null	Key	Default	Extra
transfer_id	int	NO	PRI	NULL	
sender_account_number	int	YES	MUL	NULL	
date_of_transfer	date	YES		NULL	
recipient_account_number	int	YES		NULL	
transaction_status	enum('waiting','accepted','declined')	YES		NULL	

Figure 3 account_transfer_details Table Information



```
lab3_w24
  Tables
    account_details
      Columns
        account_number
        customer_id
        account_balance
      Indexes
      Foreign Keys
      Triggers
    account_transfer_details
      Columns
        transfer_id
        sender_account_number
        date_of_transfer
        recipient_account_number
        transaction_status
      Indexes
      Foreign Keys
      Triggers
    customer_details
      Columns
        customer_id
        name
        address
        email
        phone_number
      Indexes
      Foreign Keys
      Triggers
  Views
```

Figure 4 Banking Application Database Schema

Task 2:

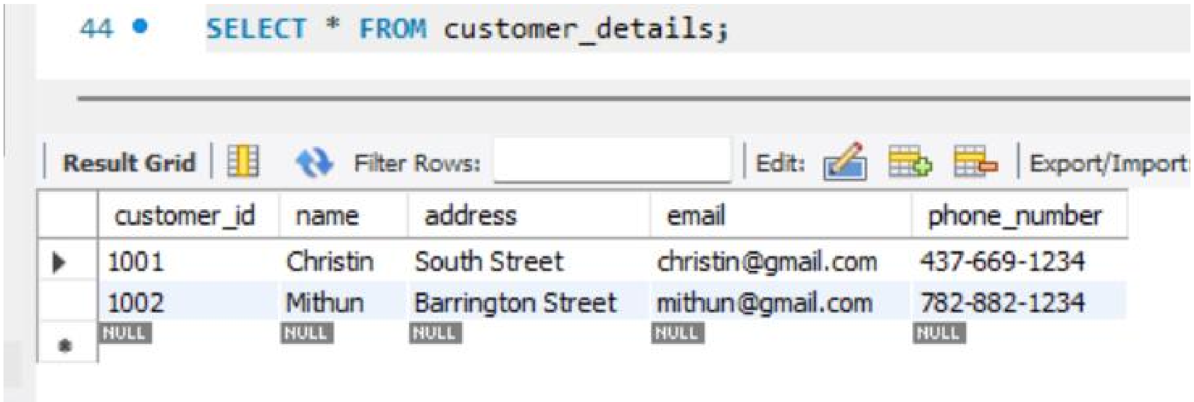
Insert some sample/dummy data in the above tables.

QUERY:

INSERT INTO customer_details (customer_id, name, address, email, phone_number) VALUES

(1001, 'Christin', 'South Street', 'christin@gmail.com', '437-669-1234'),

(1002, 'Mithun', 'Barrington Street', 'mithun@gmail.com', '782-882-1234');



```
SELECT * FROM customer_details;
```

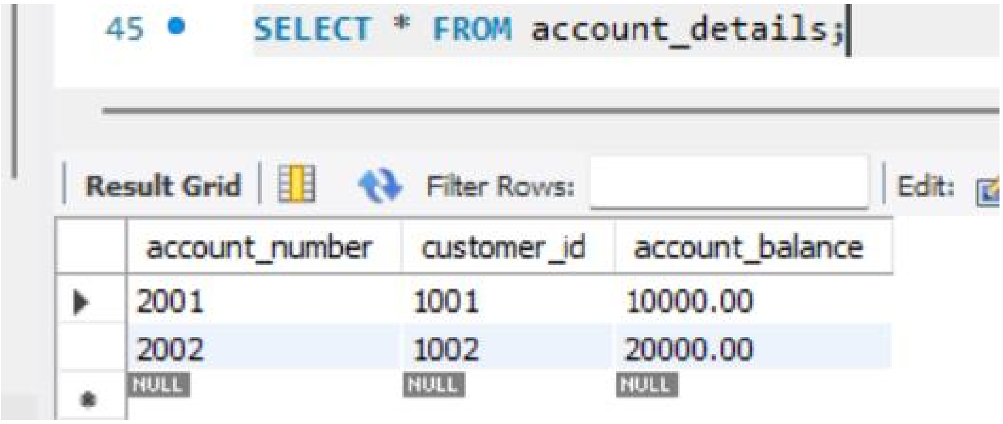
customer_id	name	address	email	phone_number
1001	Christin	South Street	christin@gmail.com	437-669-1234
1002	Mithun	Barrington Street	mithun@gmail.com	782-882-1234
NULL	NULL	NULL	NULL	NULL

Figure 5 customer_details Table with sample data

INSERT INTO account_details (account_number, customer_id, account_balance) VALUES

(2001, 1001, 10000.00),

(2002, 1002, 20000.00);



```
SELECT * FROM account_details;
```

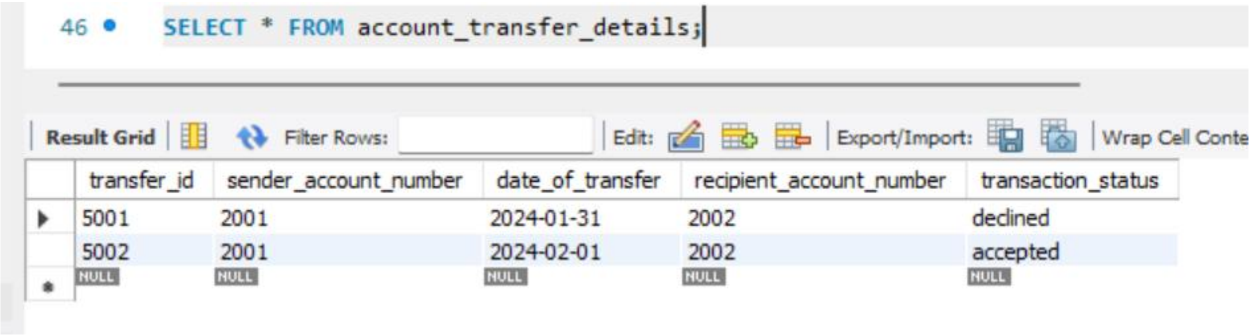
account_number	customer_id	account_balance
2001	1001	10000.00
2002	1002	20000.00
NULL	NULL	NULL

Figure 6 account_details Table with sample data

INSERT INTO account_transfer_details (transfer_id, sender_account_number, date_of_transfer, recipient_account_number, transaction_status) VALUES

(5001, 2001, '2024-01-31', 2002, 'declined'),

(5002, 2001, '2024-02-01', 2002, 'accepted');



```
SELECT * FROM account_transfer_details;
```

transfer_id	sender_account_number	date_of_transfer	recipient_account_number	transaction_status
5001	2001	2024-01-31	2002	declined
5002	2001	2024-02-01	2002	accepted
NULL	NULL	NULL	NULL	NULL

Figure 7 account_transfer_details Table with sample data

Task 3:

Create transactions for the below two scenarios:

A. Scenario-1 (Transaction “**Accepted**” state):

Initially, a debit operation takes place on the sender’s account, followed by the insertion of a record in the account transfer details table indicating the transfer is in “waiting” state. Now, assume that the transaction has **successfully passed** the security check by some X business logic (pure assumptions, no need to implement logic). Based on this status handle the credit operation for the receiver account. Lastly, the transaction status updates from “waiting” to “accepted” state.

QUERY:

```
SET autocommit = 0;


START TRANSACTION;


UPDATE account_details

SET account_balance = account_balance - 1000

        WHERE account_number = 2002;


INSERT into account_transfer_details (transfer_id, sender_account_number, date_of_transfer,
recipient_account_number, transaction_status) VALUES

(5003, 2002, CURDATE(), 2001, 'waiting');


UPDATE account_details

SET account_balance = account_balance + 1000

        WHERE account_number = 2001;


UPDATE account_transfer_details

SET transaction_status = 'accepted'

        WHERE transfer_id = 5003;


COMMIT;
```

Step 1: Set autocommit to 0 in MySQL, as it is set to 1 by default.

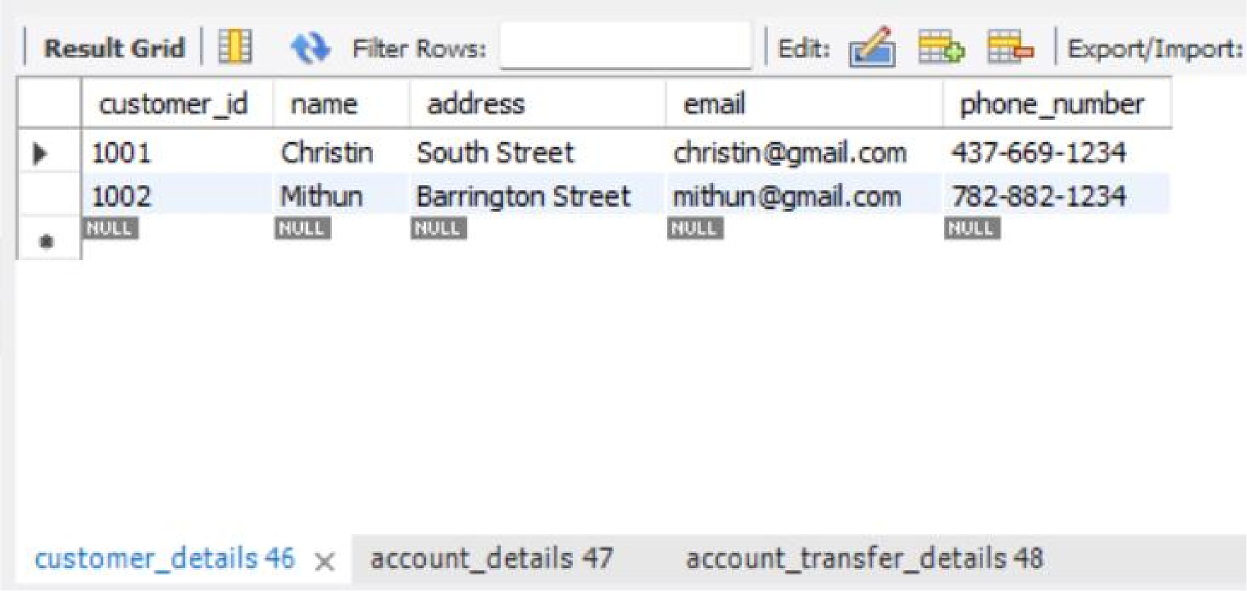
Step 2: Begin the transaction using START TRANSACTION.

Step 3: Initially debit 1000 CAD from Mithun's account.

Step 4: Update the account_transfer_details with the sender and recipient account number, transaction date, and transaction status set to 'waiting'. Assign a unique Transfer ID for reference.

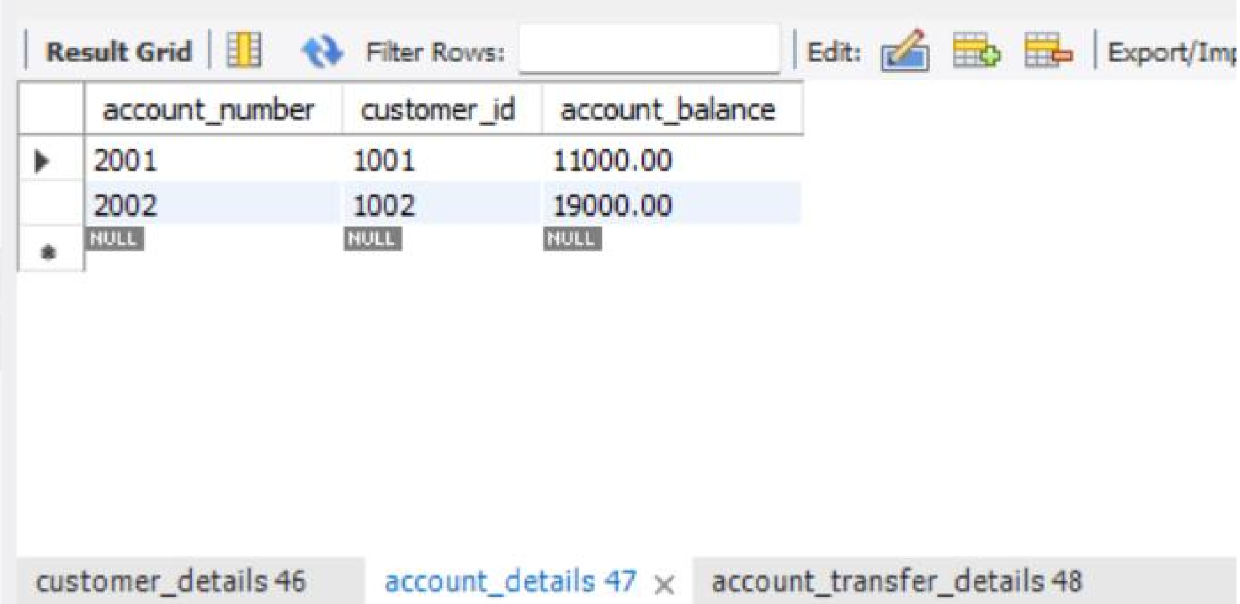
Step 5: Credit the debited amount to Christin's account.

Step 6: If the transaction is successful, change the transaction status to 'accepted', and commit it, ensuring that the transaction is reflected in the database.



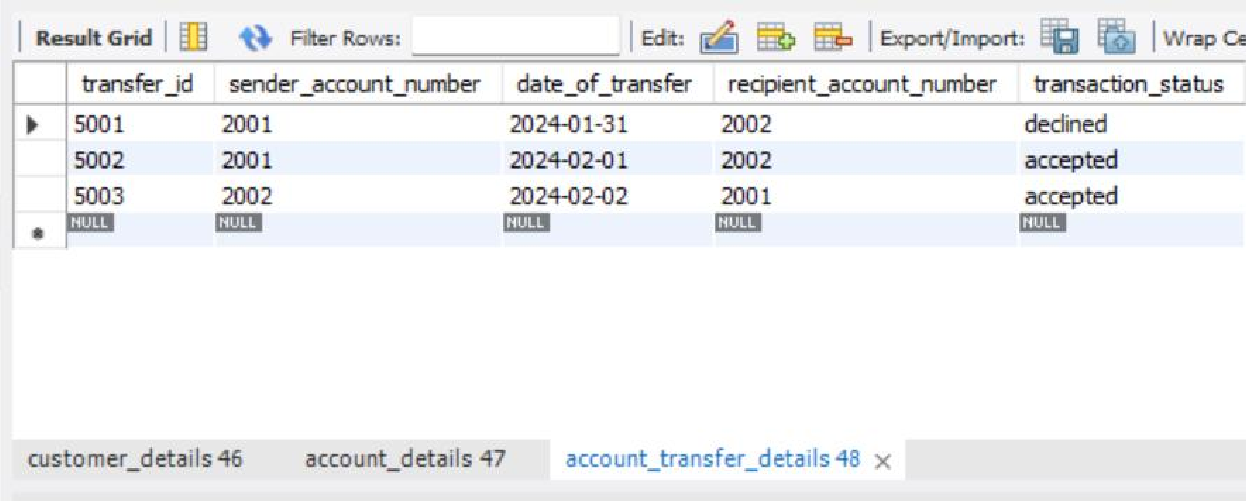
	customer_id	name	address	email	phone_number
▶	1001	Christin	South Street	christin@gmail.com	437-669-1234
	1002	Mithun	Barrington Street	mithun@gmail.com	782-882-1234
*	NULL	NULL	NULL	NULL	NULL

Figure 8 Transaction Status Accepted: customer_details Table



	account_number	customer_id	account_balance
▶	2001	1001	11000.00
	2002	1002	19000.00
*	NULL	NULL	NULL

Figure 9 Transaction Status Accepted: account_details Table



	transfer_id	sender_account_number	date_of_transfer	recipient_account_number	transaction_status
▶	5001	2001	2024-01-31	2002	declined
	5002	2001	2024-02-01	2002	accepted
	5003	2002	2024-02-02	2001	accepted
*	NULL	NULL	NULL	NULL	NULL

Figure 10 Transaction Status Accepted: account_transfer_details Table

B. Scenario-2 (Transaction “**Declined**” state):

Initially, a debit operation takes place on the sender’s account, followed by the insertion of a record in the account transfer details table indicating the transfer is in “waiting” state. Now, assume that the transaction has failed the security check by some X business logic (pure assumptions, no need to implement logic). Based on this status handle the operation for the credit operation debited on the sender account (Here, use the concept of Rollback and Save points to perform the undo operation). Lastly, the transaction status updates from “waiting” to “declined” state.

QUERY:

```
SET autocommit = 0;


START TRANSACTION;


UPDATE account_details

SET account_balance = account_balance - 2000

        WHERE account_number = 2001;


INSERT into account_transfer_details (transfer_id, sender_account_number, date_of_transfer,
recipient_account_number, transaction_status) VALUES

(5004, 2001, CURDATE(), 2002, 'waiting');


SAVEPOINT before_credit;


UPDATE account_details

SET account_balance = account_balance + 2000

        WHERE account_number = 2002;


ROLLBACK TO SAVEPOINT before_credit;


UPDATE account_details

SET account_balance = account_balance + 2000
```

```
        WHERE account_number = 2001;


UPDATE account_transfer_details

SET transaction_status = 'declined'

        WHERE transfer_id = 5004;


COMMIT;
```

Step 1: Set autocommit to 0 in MySQL, as it is set to 1 by default.

Step 2: Begin the transaction using START TRANSACTION.

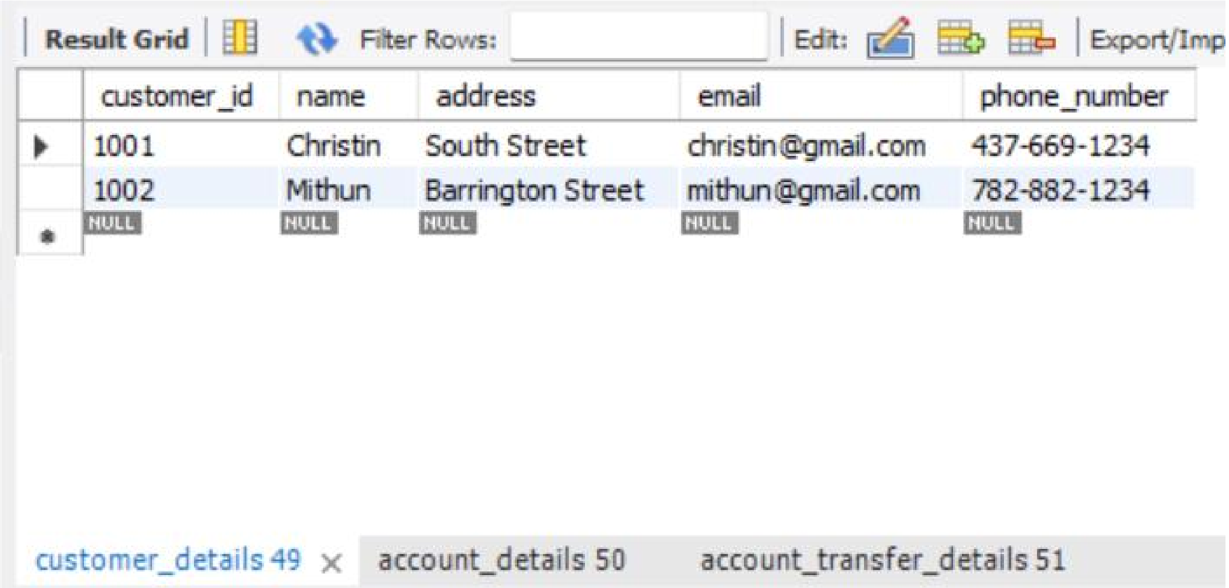
Step 3: Initially debit 2000 CAD from Christin's account.

Step 4: Update the account_transfer_details with the sender and recipient account number, transaction date, and transaction status set to 'waiting'. Assign a unique Transfer ID for reference.

Step 5: Added a SAVEPOINT before the credit operation.

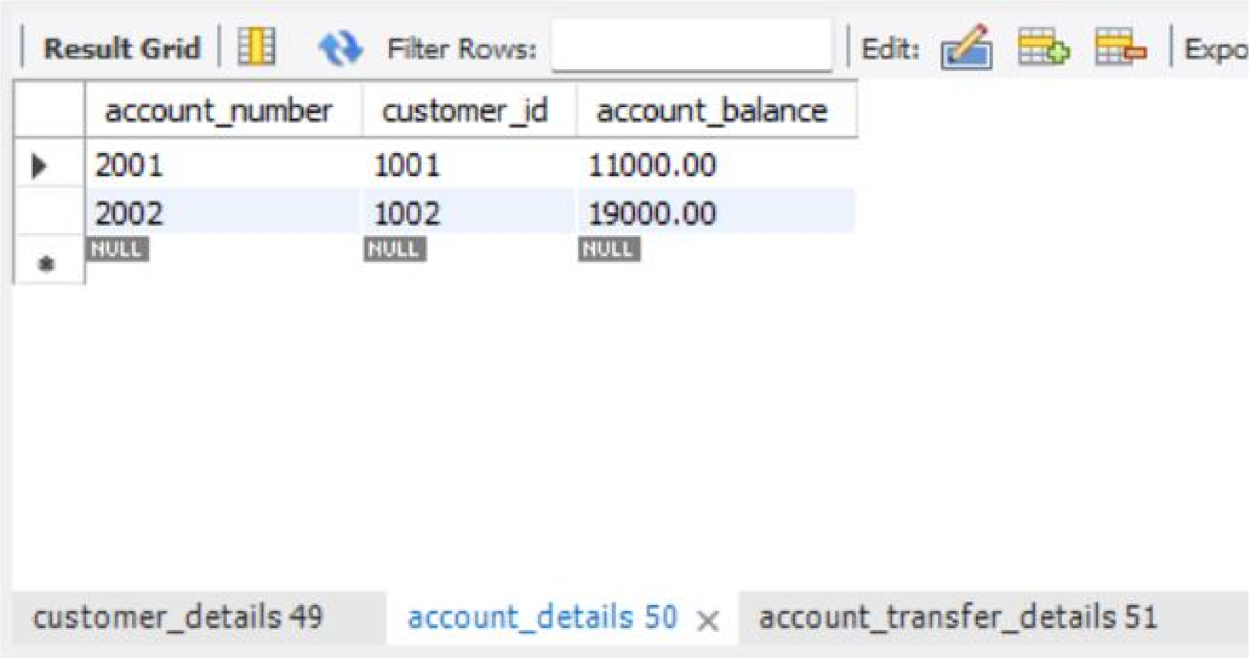
Step 5: Credit the debited amount to Christin's account if the business logic is passed; otherwise, ROLLBACK to the SAVEPOINT before the credit, ensuring the credit operation is undone.

Step 6: If the transaction fails, credit back the amount debited from Christin's account, update the transaction status to 'declined', and commit the changes to reflect in the database.



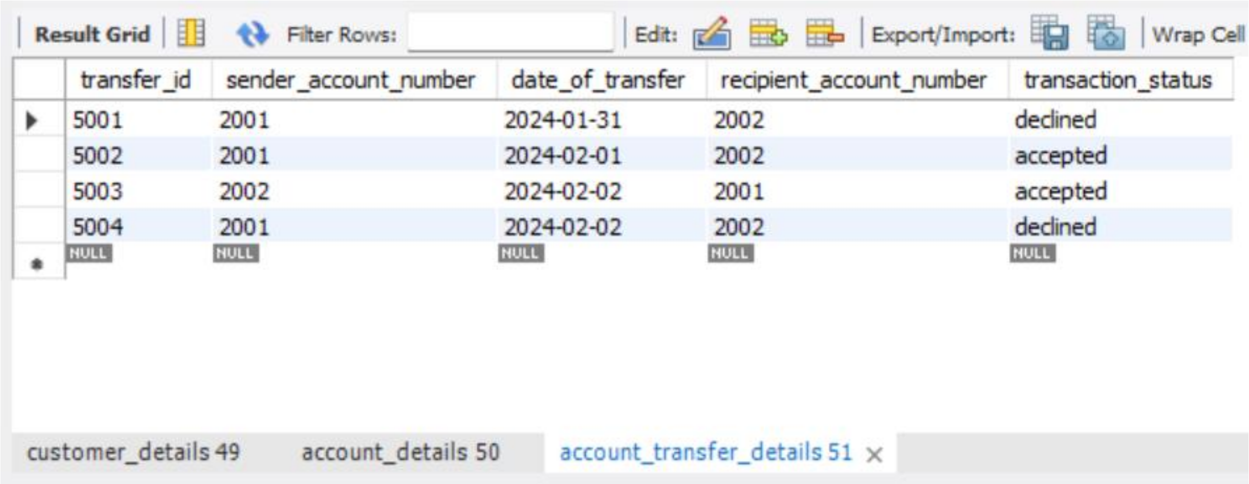
customer_id	name	address	email	phone_number
1001	Christin	South Street	christin@gmail.com	437-669-1234
1002	Mithun	Barrington Street	mithun@gmail.com	782-882-1234
NULL	NULL	NULL	NULL	NULL

Figure 11 Transaction Status Declined: customer_details Table



account_number	customer_id	account_balance
2001	1001	11000.00
2002	1002	19000.00
NULL	NULL	NULL

Figure 12 Transaction Status Declined: account_details Table



transfer_id	sender_account_number	date_of_transfer	recipient_account_number	transaction_status
5001	2001	2024-01-31	2002	declined
5002	2001	2024-02-01	2002	accepted
5003	2002	2024-02-02	2001	accepted
5004	2001	2024-02-02	2002	declined
NULL	NULL	NULL	NULL	NULL

Figure 13 Transaction Status Declined: account_transfer_details Table