# Bank Service Management System

Project Description:

Database Testing with MySQL: Verify the integrity of data by executing SQL queries to interact with the MySQL database. Check for data consistency, accuracy, and proper data manipulation. And create a documentation for it.

Project Components:

Total Tables: [7] (Customer, Accounts, Transactions, Loans, Fund Transfers, Online Payments and Mobile top-Ups)

Database Design:

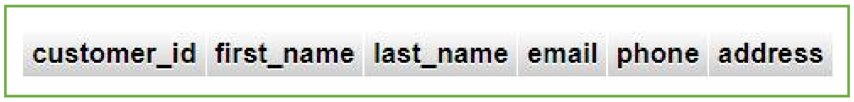
Create a database called "Bank Service Management"

```
CREATE DATABASE BankServiceManagement
```

CREATE TABLES:

1. Customers:

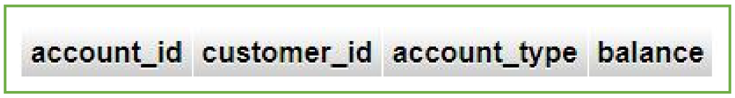
```
CREATE TABLE customers (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    phone VARCHAR(20),
    address VARCHAR(255)
);
```



| customer_id | first_name | last_name | email | phone | address |
|---|---|---|---|---|---|

2. Accounts:

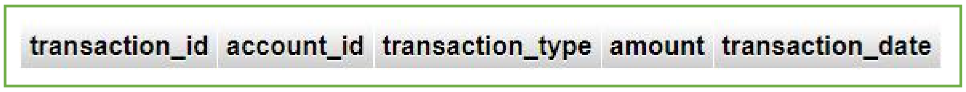
```
CREATE TABLE accounts (
   account_id INT AUTO_INCREMENT PRIMARY KEY,
   customer_id INT,
   account_type VARCHAR(20),
   balance DECIMAL(10, 2),
   FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
); #here customer_id allows bank to link customer data with Account data table
```



| account_id | customer_id | account_type | balance |
|---|---|---|---|

3. Transactions:

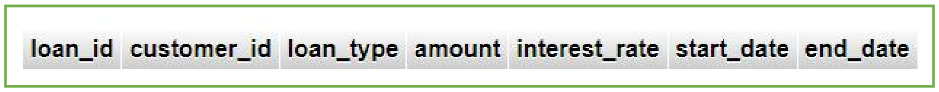
```
CREATE TABLE transactions (
   transaction_id INT AUTO_INCREMENT PRIMARY KEY,
   account_id INT,
   transaction_type VARCHAR(10),
   amount DECIMAL(10, 2),
   transaction_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   FOREIGN KEY (account_id) REFERENCES accounts(account_id)
 ); #here customer_id allows bank to link customer data with Transactions data table
```



| transaction_id | account_id | transaction_type | amount | transaction_date |
|---|---|---|---|---|

4. Loans:

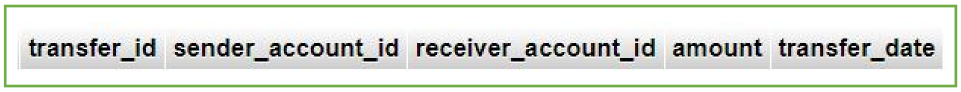
```
CREATE TABLE loans (
    loan_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    loan_type VARCHAR(50),
    amount DECIMAL(10, 2),
    interest_rate DECIMAL(5, 2),
    start_date DATE,
    end_date DATE,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
); #here customer_id allows bank to link customer data with Loans data table
```



| loan_id | customer_id | loan_type | amount | interest_rate | start_date | end_date |
|---|---|---|---|---|---|---|

5. Fund Transfers:

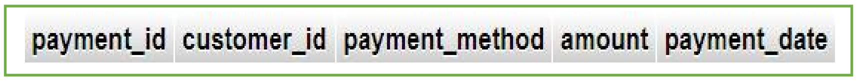
```
CREATE TABLE fund_transfers (
    transfer_id INT AUTO_INCREMENT PRIMARY KEY,
    sender_account_id INT,
    receiver_account_id INT,
    amount DECIMAL(10, 2),
    transfer_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (sender_account_id) REFERENCES accounts(account_id),
    FOREIGN KEY (receiver_account_id) REFERENCES accounts(account_id)
); #here customer_id allows bank to link customer data with Fund Transfers data table
```



| transfer_id | sender_account_id | receiver_account_id | amount | transfer_date |
|---|---|---|---|---|

6. Online Payments:

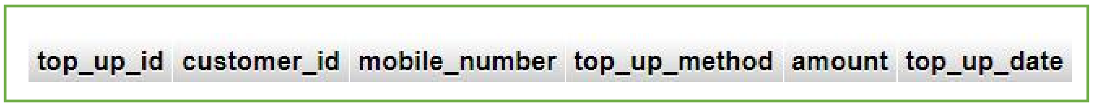
```
CREATE TABLE online_payments (
    payment_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    payment_method VARCHAR(50),
    amount DECIMAL(10, 2),
    payment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
); #here customer_id allows bank to link customer data with Online Payments data table
```



| payment_id | customer_id | payment_method | amount | payment_date |
|---|---|---|---|---|

7. Mobile Top_Ups:

```
CREATE TABLE mobile_top_ups (

    top_up_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    mobile_number VARCHAR(20),
    top_up_method VARCHAR(50),
    amount DECIMAL(10, 2),
    top_up_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
); #here customer_id allows bank to link customer data with Mobile Top_Ups data table
```

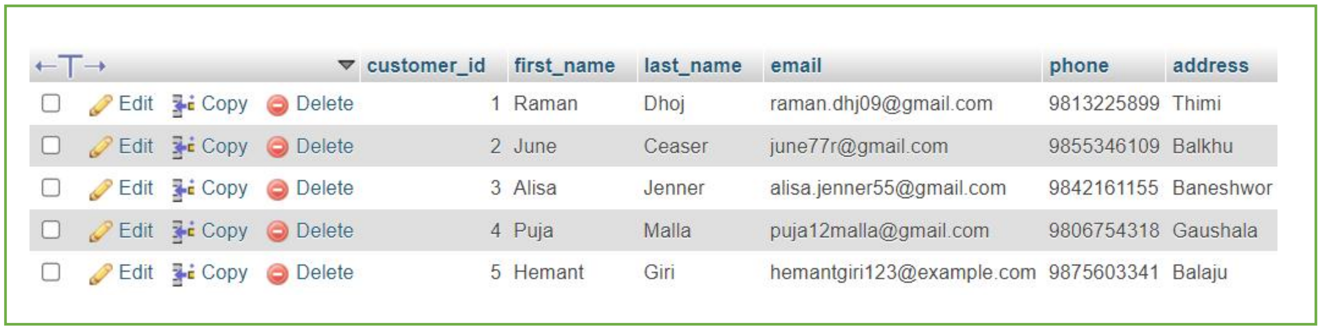


| top_up_id | customer_id | mobile_number | top_up_method | amount | top_up_date |
|---|---|---|---|---|---|

## INSERT INTO TABLE

1. Customers:

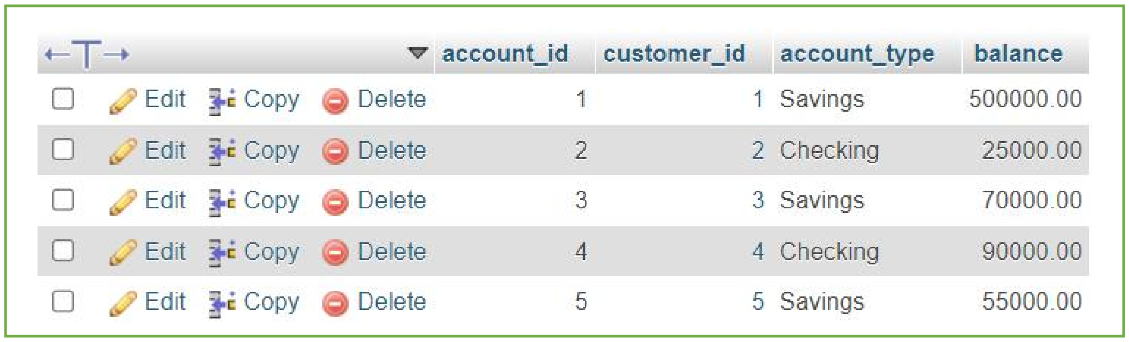
```
INSERT INTO customers (first_name, last_name, email, phone, address)
 VALUES
('Raman', 'Dhoj', 'raman.dhj09@gmail.com', '9813225899', 'Thimi'),
('June', 'Ceaser', 'june77r@gmail.com', '9855346109', 'Balkhu'),
('Alisa', 'Jenner', 'alisa.jenner55@gmail.com', '9842161155', 'Baneshwor'),
('Puja', 'Malla', 'puja12malla@gmail.com', '9806754318', 'Gaushala'),
('Hemant', 'Giri', 'hemantgiri123@example.com', '9875603341', 'Balaju');
```



| | customer_id | first_name | last_name | email | phone | address |
|---|---|---|---|---|---|---|
| Edit Copy Delete | 1 | Raman | Dhoj | raman.dhj09@gmail.com | 9813225899 | Thimi |
| Edit Copy Delete | 2 | June | Ceaser | june77r@gmail.com | 9855346109 | Balkhu |
| Edit Copy Delete | 3 | Alisa | Jenner | alisa.jenner55@gmail.com | 9842161155 | Baneshwor |
| Edit Copy Delete | 4 | Puja | Malla | puja12malla@gmail.com | 9806754318 | Gaushala |
| Edit Copy Delete | 5 | Hemant | Giri | hemantgiri123@example.com | 9875603341 | Balaju |

2. Accounts:

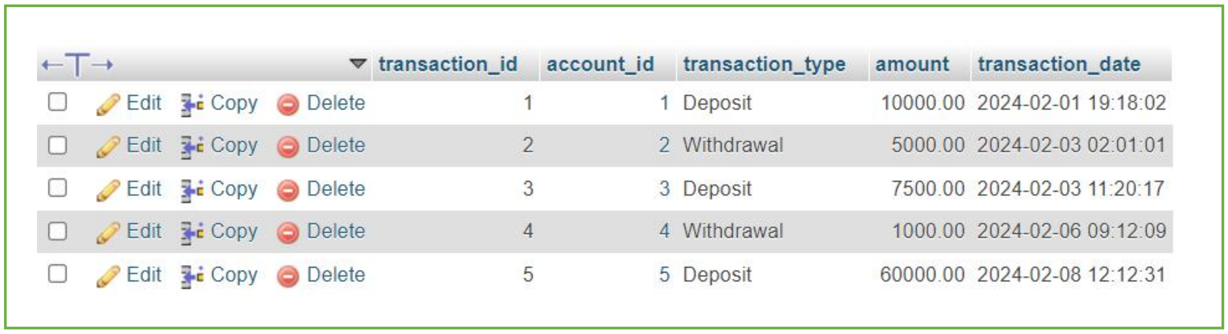
```
INSERT INTO accounts (customer_id, account_type, balance)
VALUES
(1, 'Savings', 500000.00),
(2, 'Checking', 25000.00),
(3, 'Savings', 70000.00),
(4, 'Checking', 90000.00),
(5, 'Savings', 55000.00);
```



| | account_id | customer_id | account_type | balance |
|---|---|---|---|---|
| Edit Copy Delete | 1 | 1 | Savings | 500000.00 |
| Edit Copy Delete | 2 | 2 | Checking | 25000.00 |
| Edit Copy Delete | 3 | 3 | Savings | 70000.00 |
| Edit Copy Delete | 4 | 4 | Checking | 90000.00 |
| Edit Copy Delete | 5 | 5 | Savings | 55000.00 |

3. Transactions:

```
INSERT INTO transactions (account_id, transaction_type, amount)
VALUES
(1, 'Deposit', 10000.00),
(2, 'Withdrawal', 5000.00),
(3, 'Deposit', 7500.00),
(4, 'Withdrawal', 1000.00),
(5, 'Deposit', 60000.00);
```

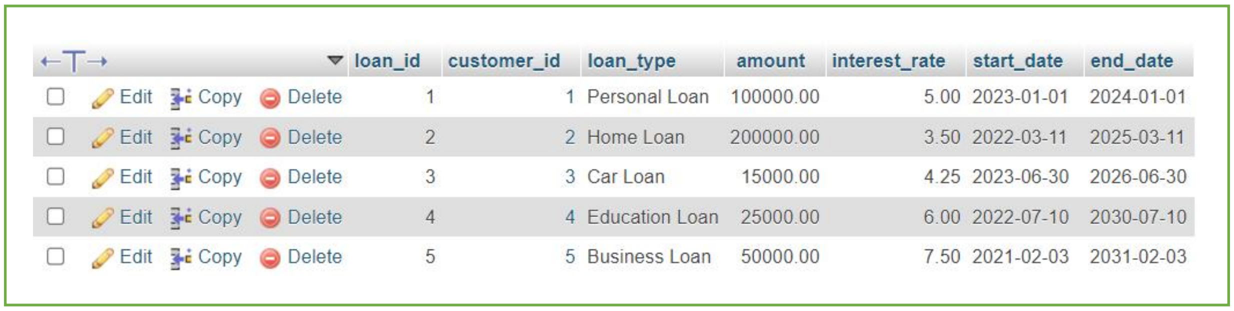


| | transaction_id | account_id | transaction_type | amount | transaction_date |
|---|---|---|---|---|---|
| Edit Copy Delete | 1 | 1 | Deposit | 10000.00 | 2024-02-01 19:18:02 |
| Edit Copy Delete | 2 | 2 | Withdrawal | 5000.00 | 2024-02-03 02:01:01 |
| Edit Copy Delete | 3 | 3 | Deposit | 7500.00 | 2024-02-03 11:20:17 |
| Edit Copy Delete | 4 | 4 | Withdrawal | 1000.00 | 2024-02-06 09:12:09 |
| Edit Copy Delete | 5 | 5 | Deposit | 60000.00 | 2024-02-08 12:12:31 |

4. Loans:

```
INSERT INTO loans (customer_id, loan_type, amount, interest_rate, start_date, end_date)
VALUES
(1, 'Personal Loan', 100000.00, 5.0, '2023-01-01', '2024-01-01'),
(2, 'Home Loan', 200000.00, 3.5, '2022-03-11', '2025-03-11'),
(3, 'Car Loan', 15000.00, 4.25, '2023-06-30', '2026-06-30'),
```

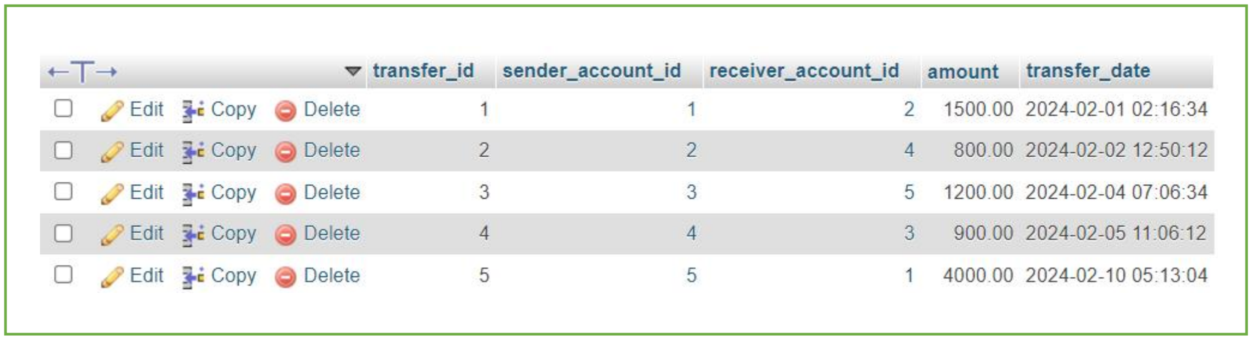
(4, 'Education Loan', 25000.00, 6.0, '2022-07-10', '2030-07-10'),
(5, 'Business Loan', 50000.00, 7.5, '2021-02-03', '2031-02-03');



| | loan_id | customer_id | loan_type | amount | interest_rate | start_date | end_date |
|---|---|---|---|---|---|---|---|
| Edit Copy Delete | 1 | 1 | Personal Loan | 100000.00 | 5.00 | 2023-01-01 | 2024-01-01 |
| Edit Copy Delete | 2 | 2 | Home Loan | 200000.00 | 3.50 | 2022-03-11 | 2025-03-11 |
| Edit Copy Delete | 3 | 3 | Car Loan | 15000.00 | 4.25 | 2023-06-30 | 2026-06-30 |
| Edit Copy Delete | 4 | 4 | Education Loan | 25000.00 | 6.00 | 2022-07-10 | 2030-07-10 |
| Edit Copy Delete | 5 | 5 | Business Loan | 50000.00 | 7.50 | 2021-02-03 | 2031-02-03 |

5. Fund Transfers:

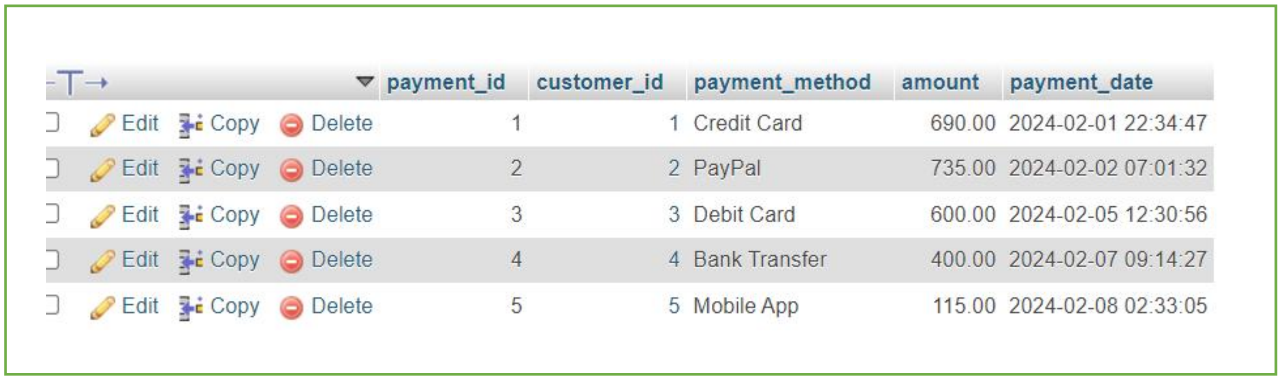
INSERT INTO fund_transfers (sender_account_id, receiver_account_id, amount)
VALUES
(1, 2, 1500.00),
(2, 4, 800.00),
(3, 5, 1200.00),
(4, 3, 900.00),
(5, 1, 4000.00);



| | transfer_id | sender_account_id | receiver_account_id | amount | transfer_date |
|---|---|---|---|---|---|
| Edit Copy Delete | 1 | 1 | 2 | 1500.00 | 2024-02-01 02:16:34 |
| Edit Copy Delete | 2 | 2 | 4 | 800.00 | 2024-02-02 12:50:12 |
| Edit Copy Delete | 3 | 3 | 5 | 1200.00 | 2024-02-04 07:06:34 |
| Edit Copy Delete | 4 | 4 | 3 | 900.00 | 2024-02-05 11:06:12 |
| Edit Copy Delete | 5 | 5 | 1 | 4000.00 | 2024-02-10 05:13:04 |

6. Online Payments:

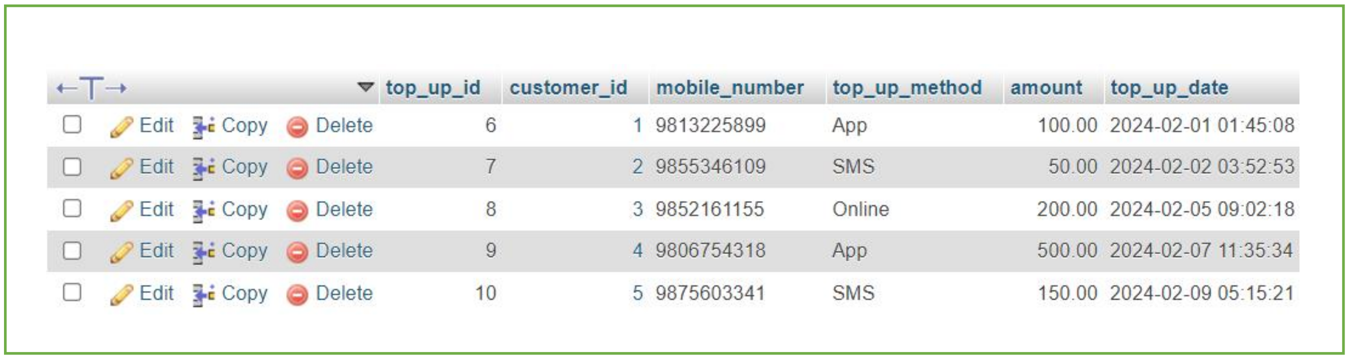
```
INSERT INTO online_payments (customer_id, payment_method, amount)
VALUES
(1, 'Credit Card', 690.00),
(2, 'PayPal', 735.00),
(3, 'Debit Card', 600.00),
(4, 'Bank Transfer', 400.00),
(5, 'Mobile App', 115.00);
```



| | payment_id | customer_id | payment_method | amount | payment_date |
|---|---|---|---|---|---|
| Edit Copy Delete | 1 | 1 | Credit Card | 690.00 | 2024-02-01 22:34:47 |
| Edit Copy Delete | 2 | 2 | PayPal | 735.00 | 2024-02-02 07:01:32 |
| Edit Copy Delete | 3 | 3 | Debit Card | 600.00 | 2024-02-05 12:30:56 |
| Edit Copy Delete | 4 | 4 | Bank Transfer | 400.00 | 2024-02-07 09:14:27 |
| Edit Copy Delete | 5 | 5 | Mobile App | 115.00 | 2024-02-08 02:33:05 |

7. Mobile Top_Ups:

```
INSERT INTO mobile_top_ups (customer_id, mobile_number, top_up_method, amount)

VALUES

(1, '9813225899', 'App', 100.00),

(2, '9855346109', 'SMS', 50.00),

(3, '9852161155', 'Online', 200.00),

(4, '9806754318', 'App', 500.00),

(5, '9875603341', 'SMS', 150.00);
```



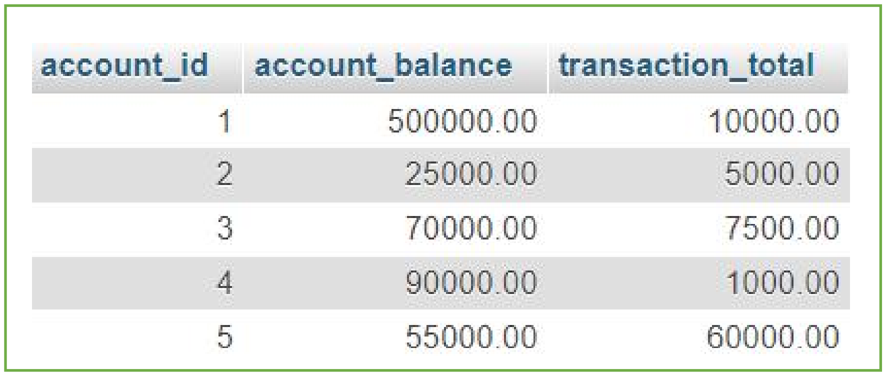
| | top_up_id | customer_id | mobile_number | top_up_method | amount | top_up_date |
|---|---|---|---|---|---|---|
| Edit Copy Delete | 6 | 1 | 9813225899 | App | 100.00 | 2024-02-01 01:45:08 |
| Edit Copy Delete | 7 | 2 | 9855346109 | SMS | 50.00 | 2024-02-02 03:52:53 |
| Edit Copy Delete | 8 | 3 | 9852161155 | Online | 200.00 | 2024-02-05 09:02:18 |
| Edit Copy Delete | 9 | 4 | 9806754318 | App | 500.00 | 2024-02-07 11:35:34 |
| Edit Copy Delete | 10 | 5 | 9875603341 | SMS | 150.00 | 2024-02-09 05:15:21 |

## Data Consistency:

### Verifying Account Balances with Transaction Records:

```
SELECT
    a.account_id,
    a.balance AS account_balance,
    COALESCE(SUM(t.amount), 0) AS transaction_total
FROM
    accounts a
    LEFT JOIN transactions t ON a.account_id = t.account_id
GROUP BY
    a.account_id
HAVING
    a.balance != COALESCE(SUM(t.amount), 0)
```

/**this query compares the balance of each account with the sum of all transaction amounts associated with that account. If the two values do not match for any account, it indicates inconsistency between the account balance and transaction records.**/



| account_id | account_balance | transaction_total |
|---|---|---|
| 1 | 500000.00 | 10000.00 |
| 2 | 25000.00 | 5000.00 |
| 3 | 70000.00 | 7500.00 |
| 4 | 90000.00 | 1000.00 |
| 5 | 55000.00 | 60000.00 |

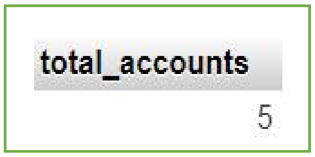
Note: Since, the values obtained above is from two different tables that are identical and matches to its original value. This verifies data consistency.

Data Accuracy:

Verify total number of accounts:

```
SELECT COUNT(*) AS total_accounts FROM accounts;
```

```
/**this query counts the total number of bank accounts that matches the expected count provided by business requirements. Any deviation from the expected count may indicate inaccuracies in the data**/
```



| total_accounts |
|---|
| 5 |

Note: since, the total accounts in database matches the count this verifies data accuracy.

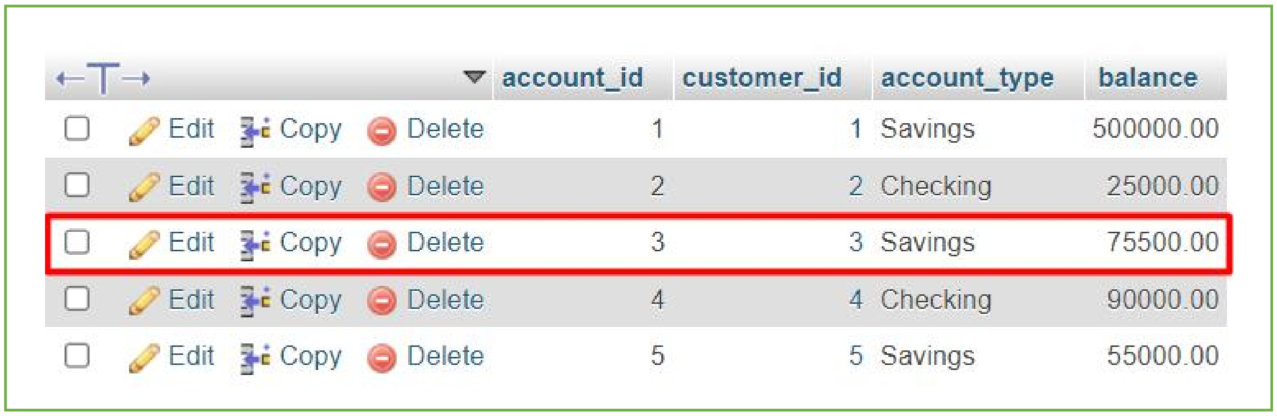
## Data Manipulation:

Data Manipulation provides operations that handle user requests, offering a way to access and manipulate the data that users store within a database. Its common functions include inserting, updating, retrieving and deleting data from the database.

Perform a deposit transaction of 5500 for a specific account and verify that the resulting balance is as expected.

```
UPDATE accounts SET balance = balance + 5500 WHERE account_id = 3;
```

```
/**this query updates amount in account_id = 3 as deposit transaction. If it does not function in database it fails to manipulate data**/
```

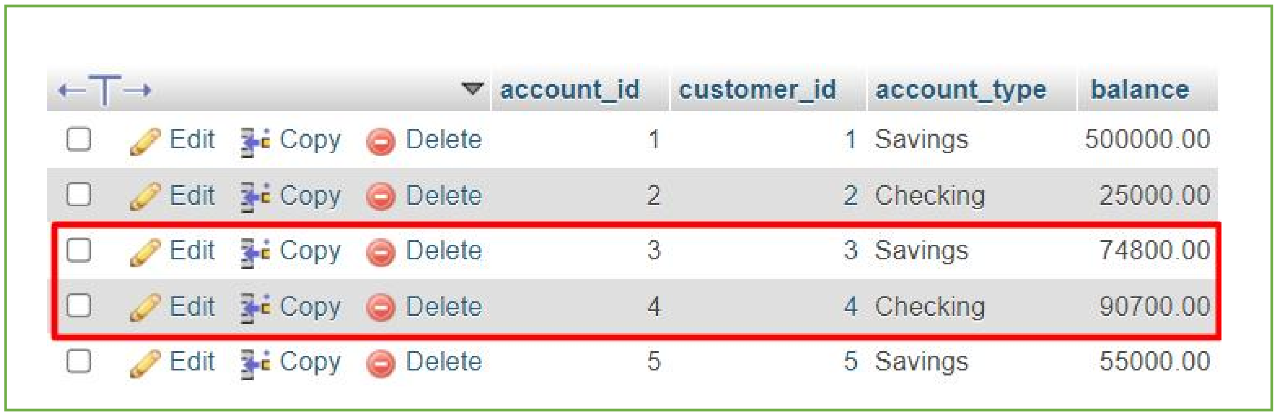


| | account_id | customer_id | account_type | balance |
|---|---|---|---|---|
| Edit Copy Delete | 1 | 1 | Savings | 500000.00 |
| Edit Copy Delete | 2 | 2 | Checking | 25000.00 |
| Edit Copy Delete | 3 | 3 | Savings | 75500.00 |
| Edit Copy Delete | 4 | 4 | Checking | 90000.00 |
| Edit Copy Delete | 5 | 5 | Savings | 55000.00 |

Note: The original balance in acoount_id = 3 was 70000. After deposit transaction it is updated to 75500. This verifies data manipulation.

Perform a Fund Transfer of 700 for specific account and verify that the resulting balance is as expected.

```
UPDATE accounts

SET balance = balance - 700

WHERE account_id = 3;


UPDATE accounts

SET balance = balance + 700

WHERE account_id = 4;


/**these SQL statements update the balances of two accounts involved in a fund transfer transaction. 700 is deducted from the account with account_id = 3 and added to the account with account_id = 4**/
```



| | account_id | customer_id | account_type | balance |
|---|---|---|---|---|
| Edit Copy Delete | 1 | 1 | Savings | 500000.00 |
| Edit Copy Delete | 2 | 2 | Checking | 25000.00 |
| Edit Copy Delete | 3 | 3 | Savings | 74800.00 |
| Edit Copy Delete | 4 | 4 | Checking | 90700.00 |
| Edit Copy Delete | 5 | 5 | Savings | 55000.00 |

Note: Since, the transaction between two accounts is successful therefore it verifies data manipulation.

## Data Integrity:

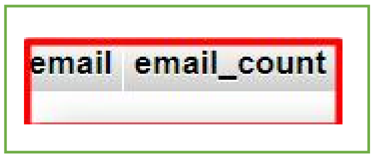
It is a concept and process that ensures the accuracy, completeness, consistency, and validity of an organization's data. By following the process, organizations not only ensure the integrity of the data but guarantee they have accurate and correct data in their database.

### Unique Email Addresses for Customers:

Verify each customer's email address is unique in the database to prevent multiple customers from sharing the same email address.

```
SELECT email, COUNT(*) AS email_count

FROM customers

GROUP BY email

HAVING email_count > 1;
```

/**this query checks for duplicate email addresses in the customers table. If any email address has a count greater than 1, it indicates that there are multiple customers sharing the same email address, violating the data integrity constraint of unique email addresses for each customer**/



| email | email_count |
|---|---|

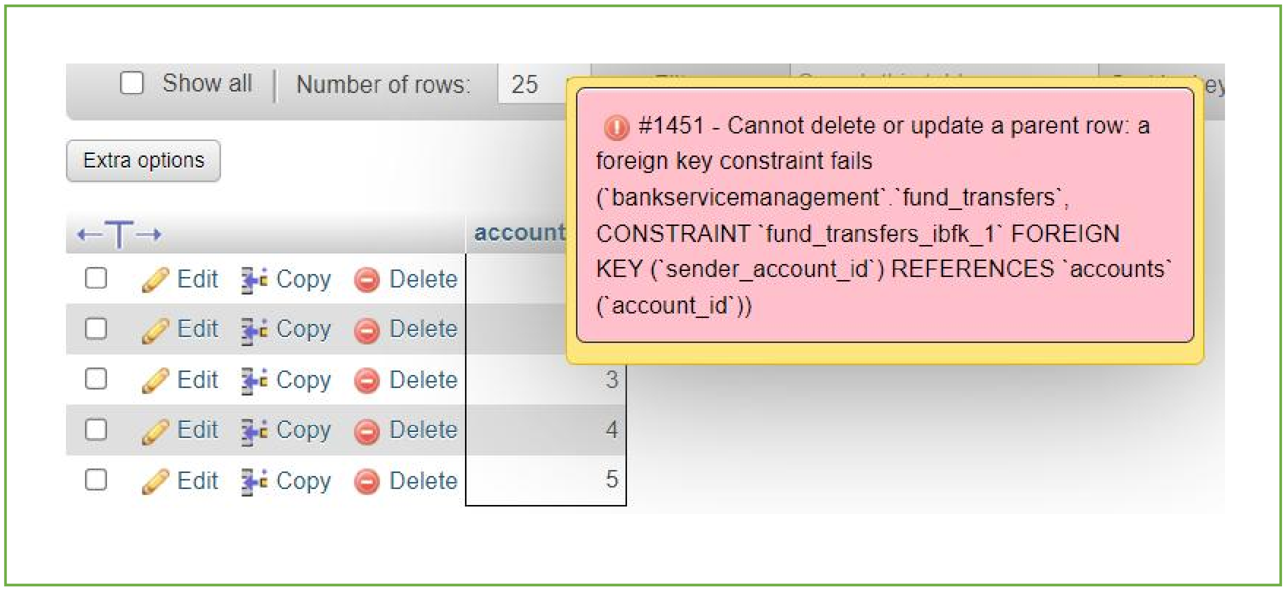
Note: Since, email_count is empty there is no customers sharing same email address in the database. This verifies data integrity

Preventing Account Deletion If Transactions Exist:

Prevent the deletion of accounts if there are associated transactions to maintain data integrity.

```
SELECT account_id
FROM accounts
WHERE account_id IN (SELECT DISTINCT account_id FROM transactions);
```

/**this query identifies accounts that have associated transactions. If any accounts are returned, it indicates that there are transactions associated with those accounts, and therefore, the deletion of these accounts should be prevented to maintain data integrity**/



```
#1451 - Cannot delete or update a parent row: a foreign key constraint fails (`bankservicemanagement`.`fund_transfers`, CONSTRAINT `fund_transfers_ibfk_1` FOREIGN KEY (`sender_account_id`) REFERENCES `accounts` (`account_id`))
```

Note: Since, the account_id cannot be deleted due to foreign key constraints therefore it maintains data integrity.

## Conclusion:

In conclusion, the database queries conducted on the bank service database provided valuable insights into the integrity, accuracy, consistency, and manipulation of data.

Data Consistency:

Account balances were compared with transaction records to ensure accuracy, detecting any discrepancies in balance calculations.

Data Accuracy:

The total number of accounts was verified, ensuring that they align with the expected counts based on business requirements.

Data Manipulation:

Deposit transactions was verified to a specified account.

Fund transfer transactions were examined to verify the integrity of transferred amounts between sender and receiver accounts.

Data Integrity:

The queries revealed that there are no orphaned accounts ensuring that all accounts are associated with valid customers.

Foreign key constraints were verified to ensure that transactions are linked to existing accounts, maintaining referential integrity.

Overall, the database queries helped ensure the reliability and integrity of the bank service database, highlighting areas of improvement and validating the accuracy of data operations. By adhering to these principles of data management, the bank can maintain a robust and trustworthy database system, enhancing customer satisfaction and operational efficiency.