SQL Project Test Cases

Customer Table

1. Test Case 1: Verify Customer Address Retrieval

   Objective: Ensure that given a customer ID, their address data can be retrieved.

   Preconditions: Customer data exists in the database.

   Test Steps:

   1. Execute the query:
      ```
      SELECT address FROM Customers WHERE customer_id = 2;
      ```

   2. Verify that the query returns the correct address for the customer.

   Expected Result: The correct address associated with customer ID 2 is displayed.

2. Test Case 2: Verify Total Number of Customers

   Objective: Ensure that the total number of customers can be retrieved.

   Preconditions: Customers exist in the database.

   Test Steps:

   1. Execute the query:
      ```
      SELECT COUNT(*) AS total_customers FROM Customers;
      ```

   2. Verify that the count matches the expected number of customers.

   Expected Result: The total count of customers is correctly displayed.

Accounts Table

3. Test Case 3: Verify Minimum Two Bank Accounts per Customer
   Objective: Ensure each customer has at least two bank accounts. Preconditions: Customer and account data exist.

Test Steps:

a. Execute the query:

```
  SELECT c.customer_id, c.first_name, c.last_name,
COUNT(a.account_number) AS account_count
  FROM customers c
  JOIN accounts a ON c.customer_id = a.customer_id
  GROUP BY c.customer_id, c.first_name, c.last_name
  HAVING COUNT(a.account_number) >= 2;
```

b. Verify that all customers in the result have at least two accounts.

Expected Result: Each customer has at least two bank accounts.

2. Test Case 4: Verify Customers Have Checking and Savings Accounts

Objective: Ensure each customer has at least one checking and one savings account.

Test Steps:

1. Execute the query:

```
  SELECT customer_id FROM accounts WHERE account_type =
'CHECKING'
  INTERSECT
  SELECT customer_id FROM accounts WHERE account_type =
'SAVINGS';
```

2. Verify that all customers returned have both account types.

Expected Result: Each customer has at least one checking and one savings account.

Transactions Table

5. Test Case 5: Verify Minimum Transactions per Account Objective: Ensure each account has at least three transactions. Preconditions: Transaction data exists.

   Test Steps:

1. Execute the query:

```
SELECT a.customer_id, t.account_number, COUNT(t.transaction_id)
 AS transaction_count
FROM transactions t
JOIN accounts a ON t.account_number = a.account_number
GROUP BY a.customer_id, t.account_number
HAVING COUNT(t.transaction_id) >= 3;
```

2. Verify that all accounts listed have at least three transactions.

Expected Result: Each account has at least three transactions.

6. Test Case 6: Retrieve Transactions for Checking Accounts

Objective: Ensure transactions for a given customer's checking accounts are retrieved correctly.

Test Steps:

1. Execute the query:

```
SELECT t.account_number, t.transaction_date, t.amount, t.transaction_type
FROM transactions t
JOIN accounts a ON t.account_number = a.account_number
WHERE a.customer_id = 3 AND a.account_type = 'CHECKING'
ORDER BY t.transaction_date;
```

2. Verify that only transactions from checking accounts of the specified customer are returned.

Expected Result: Transactions for customer ID 3's checking account are correctly retrieved and ordered by date.

7. Test Case 7: Verify First Transaction is a Deposit

Objective: Ensure that the first transaction recorded for an account is a deposit.

Test Steps:

1. Execute the query:

```
    SELECT account_number, transaction_type
    FROM transactions
    WHERE transaction_id = (SELECT MIN(transaction_id) FROM transactions
  WHERE account_number = transactions.account_number);
```

3. Verify that the first transaction for each account is a deposit.

Expected Result: The first transaction recorded for each account is a deposit.

8. Test Case 8: Retrieve Top 2 Customers by Transaction Amount

Objective: Ensure the query retrieves the top 2 customers by total transaction amount.

Test Steps:

1. Execute the query:

```
SELECT a.customer_id, SUM(t.amount) AS total_amount
FROM transactions t
JOIN accounts a ON t.account_number = a.account_number
WHERE t.transaction_type = 'DEPOSIT'
GROUP BY a.customer_id
ORDER BY total_amount DESC
LIMIT 2;
```

2. Verify that the top 2 customers with the highest transaction amounts are retrieved.

Expected Result: The query correctly returns the top 2 customers based on total transaction amount.

9. Test Case 9: Verify No Account Has More Withdrawals Than Deposits

Objective: Ensure no account has total withdrawals greater than total deposits.

Test Steps:

1. Execute the query:

```
  SELECT account_number,
        SUM(CASE WHEN transaction_type = 'WITHDRAWAL' THEN amount ELSE 0 END) AS
total_withdrawals,
        SUM(CASE WHEN transaction_type = 'DEPOSIT' THEN amount
ELSE 0 END) AS total_deposits
  FROM transactions
  GROUP BY account_number
  HAVING total_withdrawals > total_deposits;
```

2. Verify that no accounts are returned by the query.

Expected Result: No account has more withdrawals than deposits.