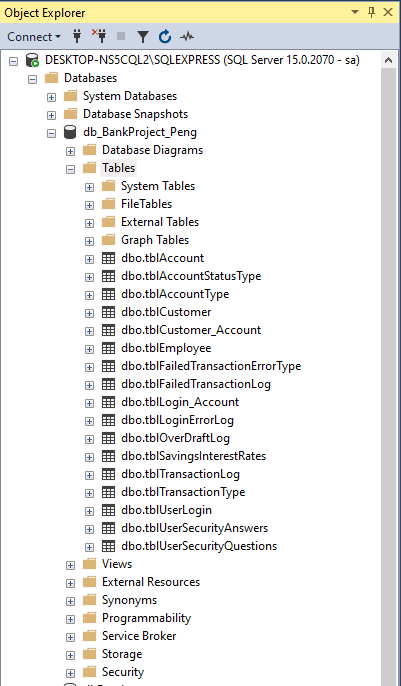
****

**SQL Programming**

**Project Phase 1**

Total # of Questions: [4]

**Question/Problem 1**

1. **Create a database for a banking application called “Bank”. [Basic]**
2. **Create all the tables mentioned in the database diagram. [Moderate]**
3. **Create all the constraints based on the database diagram. [Advanced]**
4. **Insert at least 5 rows in each table. [Basic]**
5. **Create a database for a banking application called “Bank”. [Basic]**

**2. Create all the tables mentioned in the database diagram. [Moderate]**

--CREATE TABLES--

/\*

TABLES FOR ACCOUNT:

-------------------

Table #1: AccountType, parent table for Account

Table #2: AccountStatusType, parent table for Account

Table #3: SavingInterestRate, parent table for Account

Table #4: Account, parent table for OverDraftLog

Table #5: OverDraftLog

\*/

Table #1: AccountType

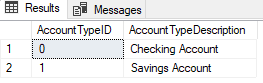


Table #2: AccountStatusType

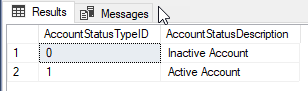


Table #3: SavingInterestRate

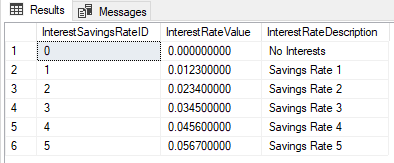


Table #4: Account

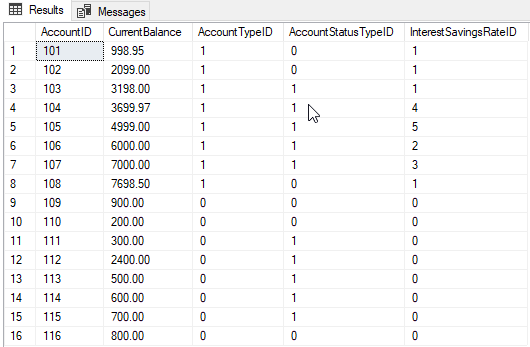
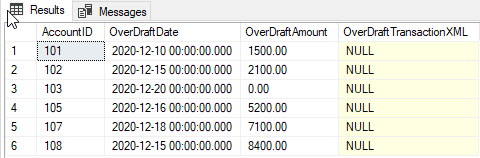


Table #5: OverDraftLog



/\*

TABLES FOR USER LOGIN:

----------------------

Table #6: UserLogin, parent table for UserSecurityAnswers AND Login-Account

Table #7: UserSecurityQuestions, parent table for UserSecurityAnswers

Table #8: UserSecurityAnswers

Table #9: Login\_Account

Table #10: LoginErrorLog

\*/

Table #6: UserLogin

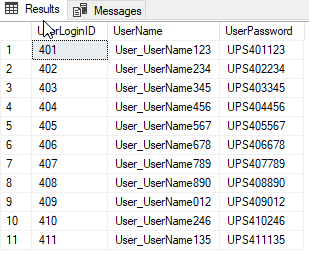


Table #7: UserSecurityQuestions

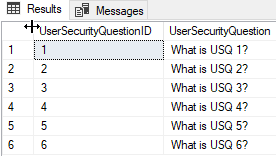


Table #8: UserSecurityAnswers

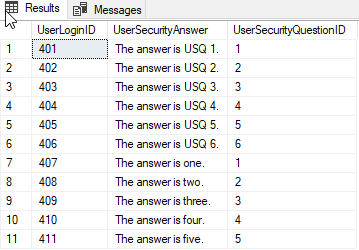


Table #9: Login\_Account

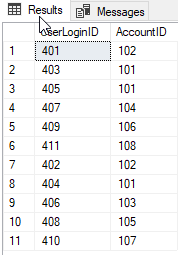
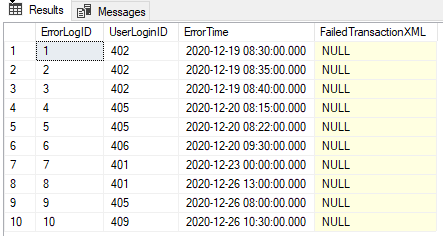


Table #10: LoginErrorLog



/\*

TABLES FOR CUSTOMER:

----------------------

Table #11: Customer, parent table for Customer\_Account

Table #12: Customer\_Account

\*/

Table #11: Customer

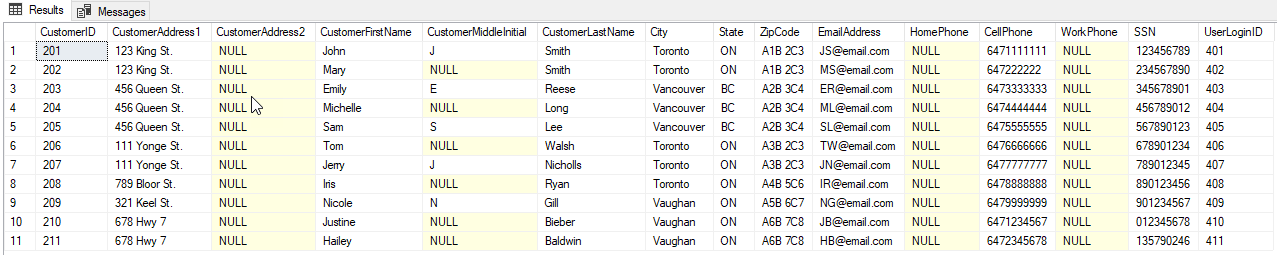
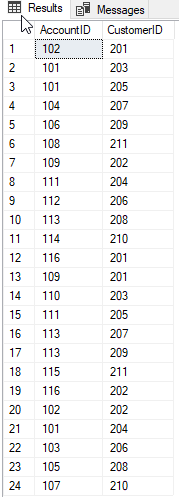


Table #12: Customer\_Account



/\*

TABLES FOR EMPLOYEE AND TRANSACTIONS:

----------------------

Table #13: Employee, parent table for TransactionLog

Table #14: TransactionType, parent table for TransactionLog

Table #15: TransactionLog

Table #16: FailedTransactionErrorType, parent table for FailedTransactionLog

Table #17: FailedTransactionLog

\*/

Table #13: Employee

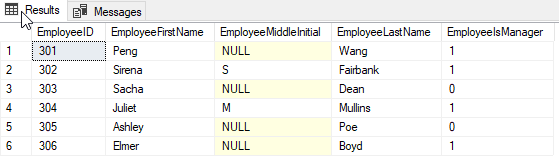


Table #14: TransactionType

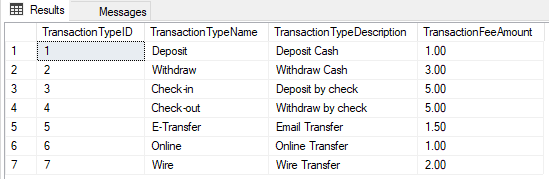


Table #15: TransactionLog

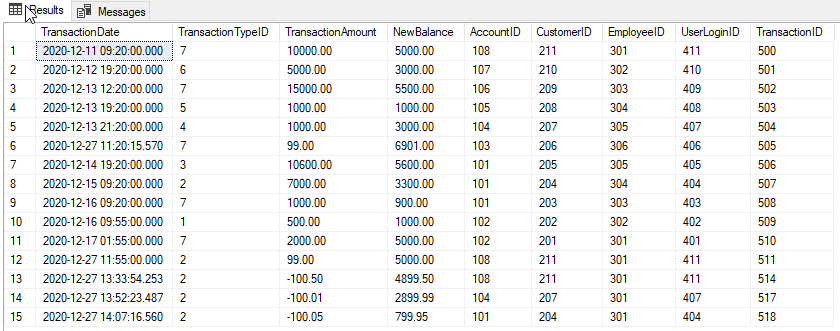


Table #16: FailedTransactionErrorType

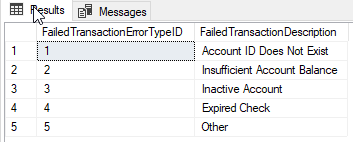
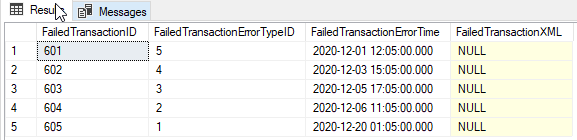


Table #17: FailedTransactionLog



**3. Create all the constraints based on the database diagram. [Advanced]**

****

****

**SQL Programming**

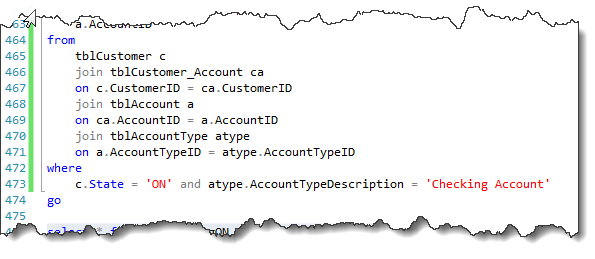
**Project Phase 2**

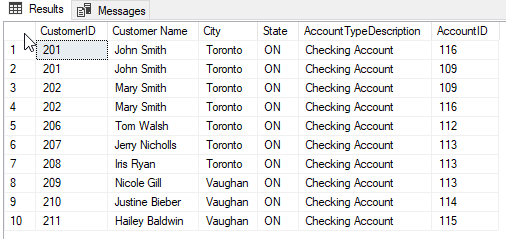
Total # of Questions: [5]

**Question/Problem 1**

1. **Create a view to get all customers with checking account from ON province. [Moderate]**
2. **Create a view to get all customers with total account balance (including interest rate) greater than 5000. [Advanced]**
3. **Create a view to get counts of checking and savings accounts by customer. [Moderate]**
4. **Create a view to get any particular user’s login and password using AccountId. [Moderate]**
5. **Create a view to get all customers’ overdraft amount. [Moderate]**
6. **Create a stored procedure to add “User\_” as a prefix to everyone’s login (username). [Moderate]**
7. **Create a stored procedure that accepts AccountId as a parameter and returns customer’s full name. [Advanced]**
8. **Create a stored procedure that returns error logs inserted in the last 24 hours. [Advanced]**
9. **Create a stored procedure that takes a deposit as a parameter and updates CurrentBalance value for that particular account. [Advanced]**
10. **Create a stored procedure that takes a withdrawal amount as a parameter and updates**
11. **Prepare a report to describe the project. [Moderate]**
12. **Prepare a presentation for the project. [Moderate]**

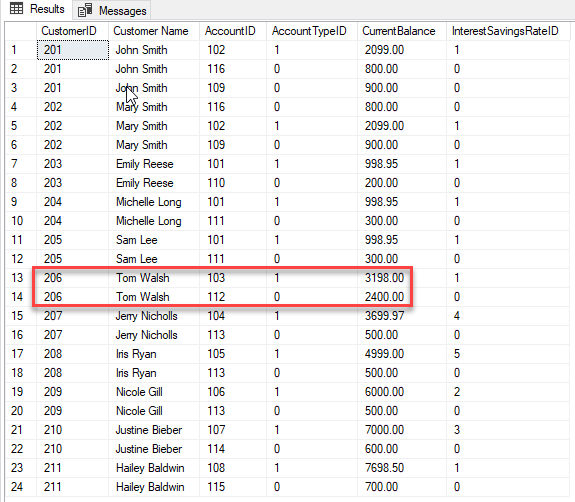
**1. Create a view to get all customers with checking account from ON province. [Moderate]**

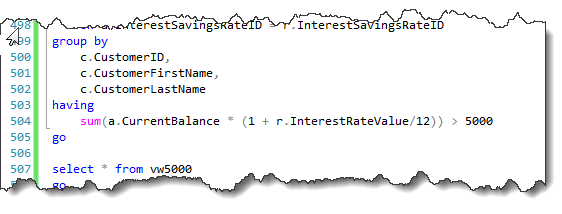


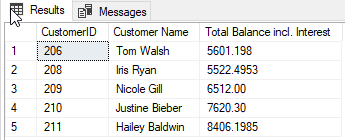


**2. Create a view to get all customers with total account balance (including interest rate) greater than 5000. [Advanced]**

From the below table, I expected to have Tom Walsh to be selected as his total account balance exceeds 5000.



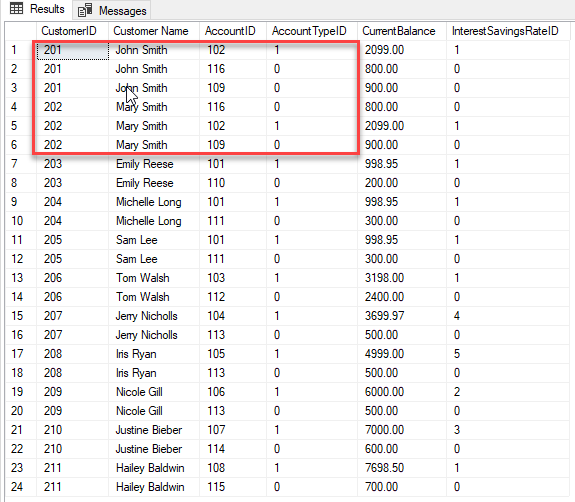


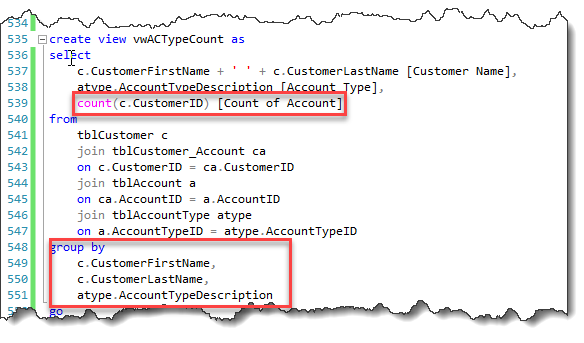


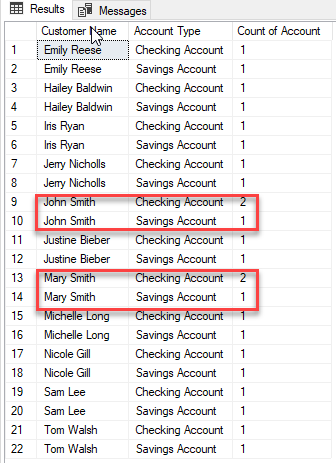
**3. Create a view to get counts of checking and savings accounts by customer. [Moderate]**

I expected to see the correct count for John Smith and Mary Smith.

Count customer ID from below joined table then group by customer name and account type will bring the result.

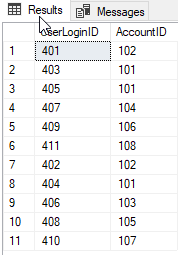
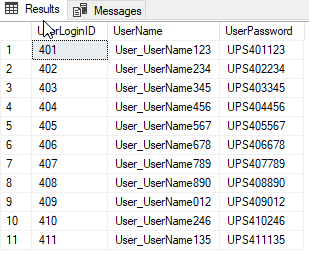




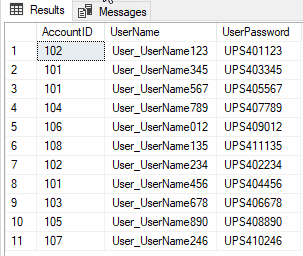


**4. Create a view to get any particular user’s login and password using AccountId. [Moderate]**

Join below two tables will get the required list.



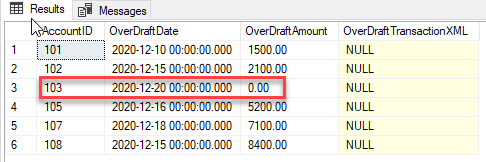
**Result:**



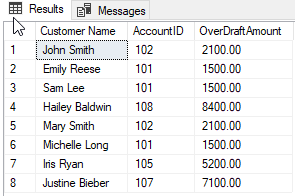
**5. Create a view to get all customers’ overdraft amount. [Moderate]**

Join tables Customer, Account and OverDraftLog.

**Pay attention to exclude zero amount.**



**Result:**



**6. Create a stored procedure to add “User\_” as a prefix to everyone’s login (username). [Moderate]**

This is the syntext to add prefix.

set UserName = concat('User\_', UserName)

However we need to avoid keep adding the same prefix for multiple times.

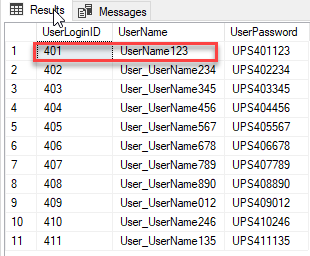
where left(UserName, 5) != 'User\_'

To test the result, firstly I removed ‘Use\_’ from a random loginID.

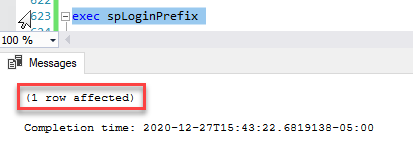
update tblUserLogin

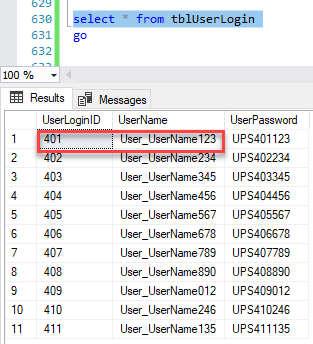
set UserName = 'UserName123'

where UserLoginID = 401



I then executed proc and expected to see the proc will add it back without impact other loginIDs that already have that prefix.

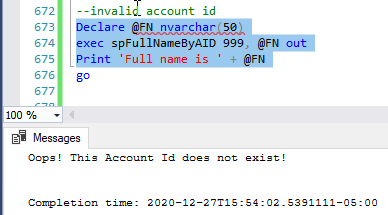
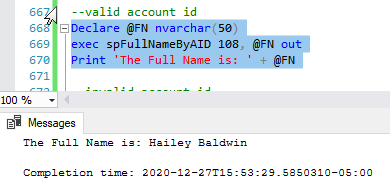
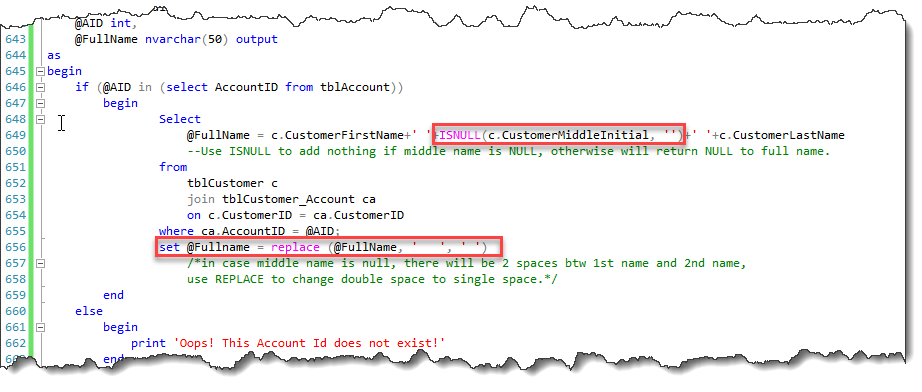




**7. Create a stored procedure that accepts AccountId as a parameter and returns customer’s full name. [Advanced]**

Challenge is that full name will return as NULL if middlename is null. Use ISNULL function will resolve the issue.

The 2nd issue is that once ISNULL returned nothing, it left the full name with 2 spaces inbtween the first name and the second name. Use REPLACE function to change double space to single space fixed the problem.



**8. Create a stored procedure that returns error logs inserted in the last 24 hours. [Advanced]**

After testing the different scenarios, DATEADD(day, -1, getdate()) brought the expected calculation for about 24 hours againt Current Date & Time. No need to separate into Hours, Minutes, even Seconds.

At this moment I draft this report, the current time is 16:09 Dec 27th 2020. I just inserted a new item with the date/time of 17:00 Dec 26th 2020, which is within 24 hours, therefore should be selected.

insert into tblLoginErrorLog values

(11, 402, convert(datetime, '2020-12-26 17:00', 120), null)

--within 24 hrs from now (2020-12-27 16:05)

--expect to be selected

Create proc spErrorLog24 as

Begin

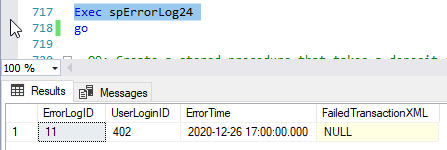
select \*

from tblLoginErrorLog e

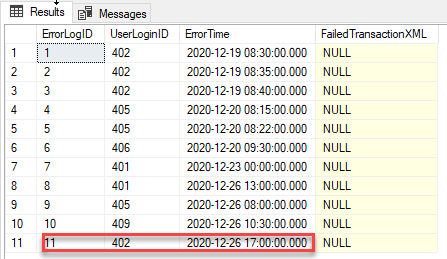
where e.ErrorTime > DATEADD(day, -1, getdate())

--DATEADD(interval, increment int, expression smalldatetime) RETURNS smalldatetime.

End



**Compare other records in the log:**



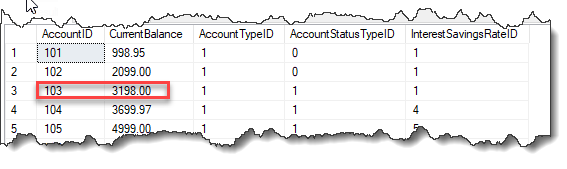
**9. Create a stored procedure that takes a deposit as a parameter and updates CurrentBalance value for that particular account. [Advanced]**

As per question, this proc only update the table Account with increased Current Balance.

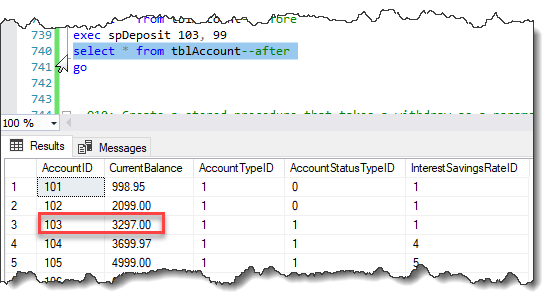
Take account 103 as example:

exec spDeposit 103, 99

Before:



After:

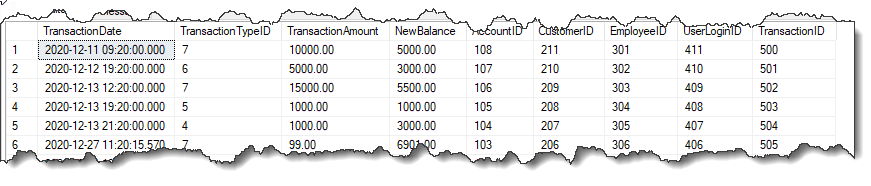


**10. Create a stored procedure that takes a withdrawal amount as a parameter and updates**

Different from Q9, this time will update two tables:

1. Increase Account Current Balance; and
2. Add new row/record in table Transaction Log.

Ref below for table structure.



The amount after withdraw should be the New Balance to be inserted into the transaction log.

The logic used is New$ = Old$ - Withdraw$.

* Finding the old$ required to locate the latest balance for a given account & customer.
* Especially when multiple records exist for the same account and same customer.
* The latest entry for that account and customer should be considered.

Therefore I use:

select max(transactiondate) from tblTransactionLog where AccountID = @AcID and CustomerID = @CustID)

Above returns the latest date for the given account and customer. The next step is to locate the balance on this date:

--assign the original balance to a variable(OldBalance) - logic: Old$ - withdraw$ = new$

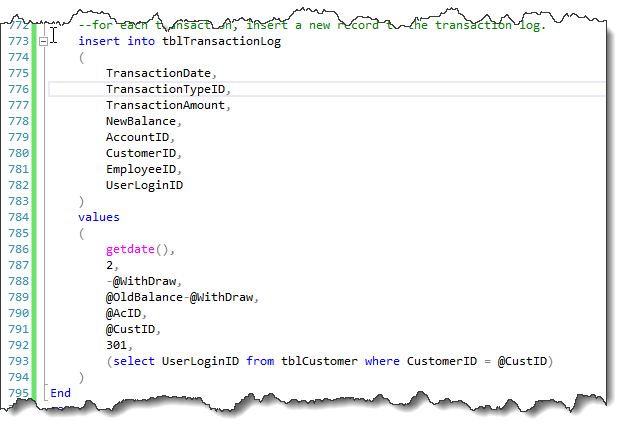
declare @OldBalance money

set @OldBalance = (

select NewBalance from tblTransactionLog where TransactionDate = (

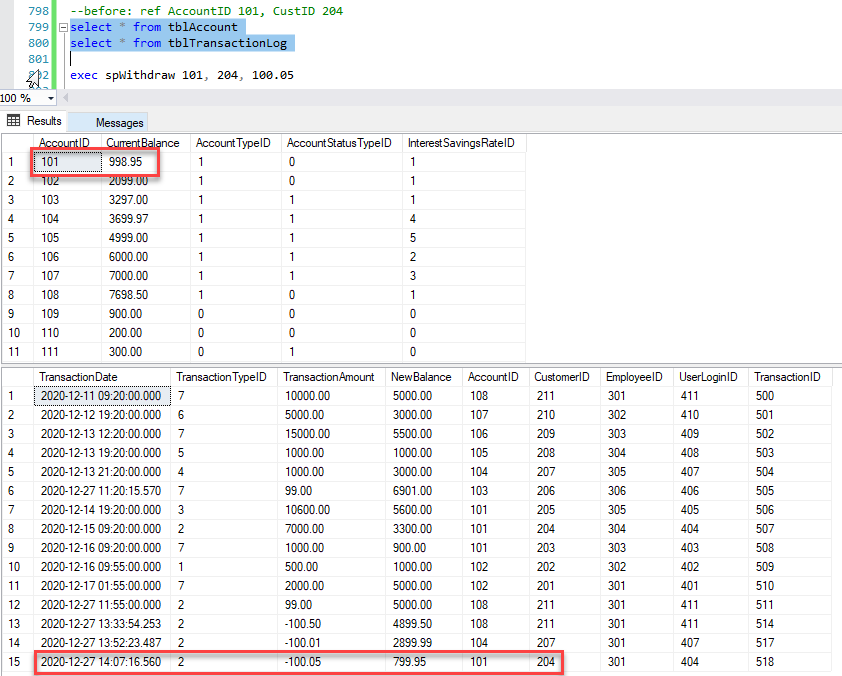
select max(transactiondate) from tblTransactionLog where AccountID = @AcID and CustomerID = @CustID))

I did not figure out how to enter TransactionTypeID and EmployeeID bypassing parameters. At this moment, I manually entered.

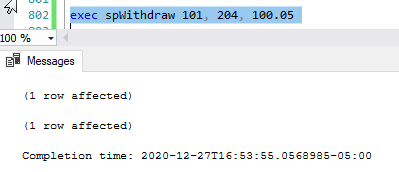


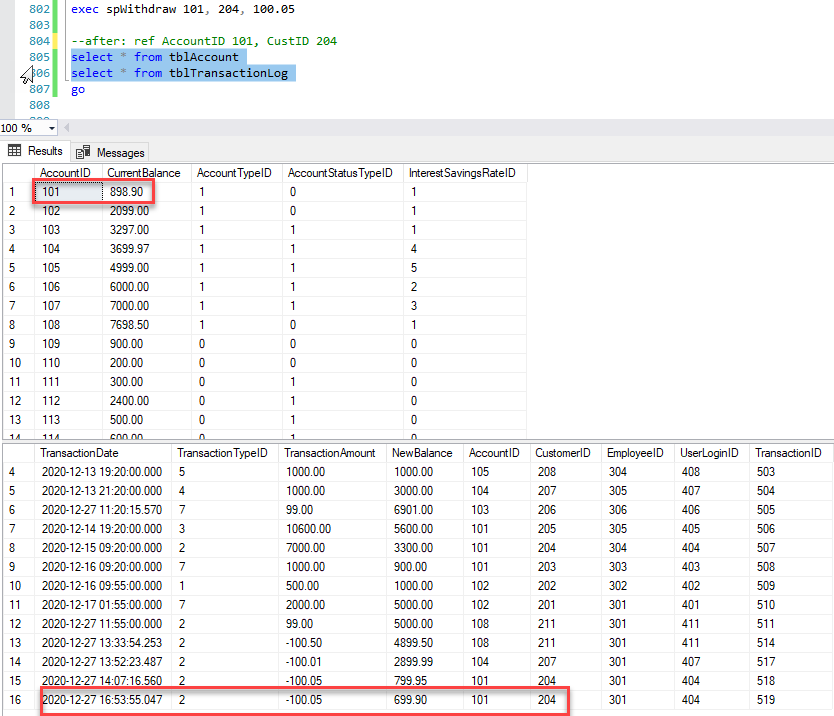
Use account 101, Customer 204 as an example:

Before withdraw 100.05:



After withdraw:





**11. Prepare a report to describe the project. [Moderate]**

Report done on Dec 27th 2020.

**12. Prepare a presentation for the project. [Moderate]**

Presentation done on Dec 23rd 2020.