Student Name: Peng Wang Date: Dec 27th 2020

METRO COLLEGE OF TECHNOLOGY

SQL Programming
Project Phase 1

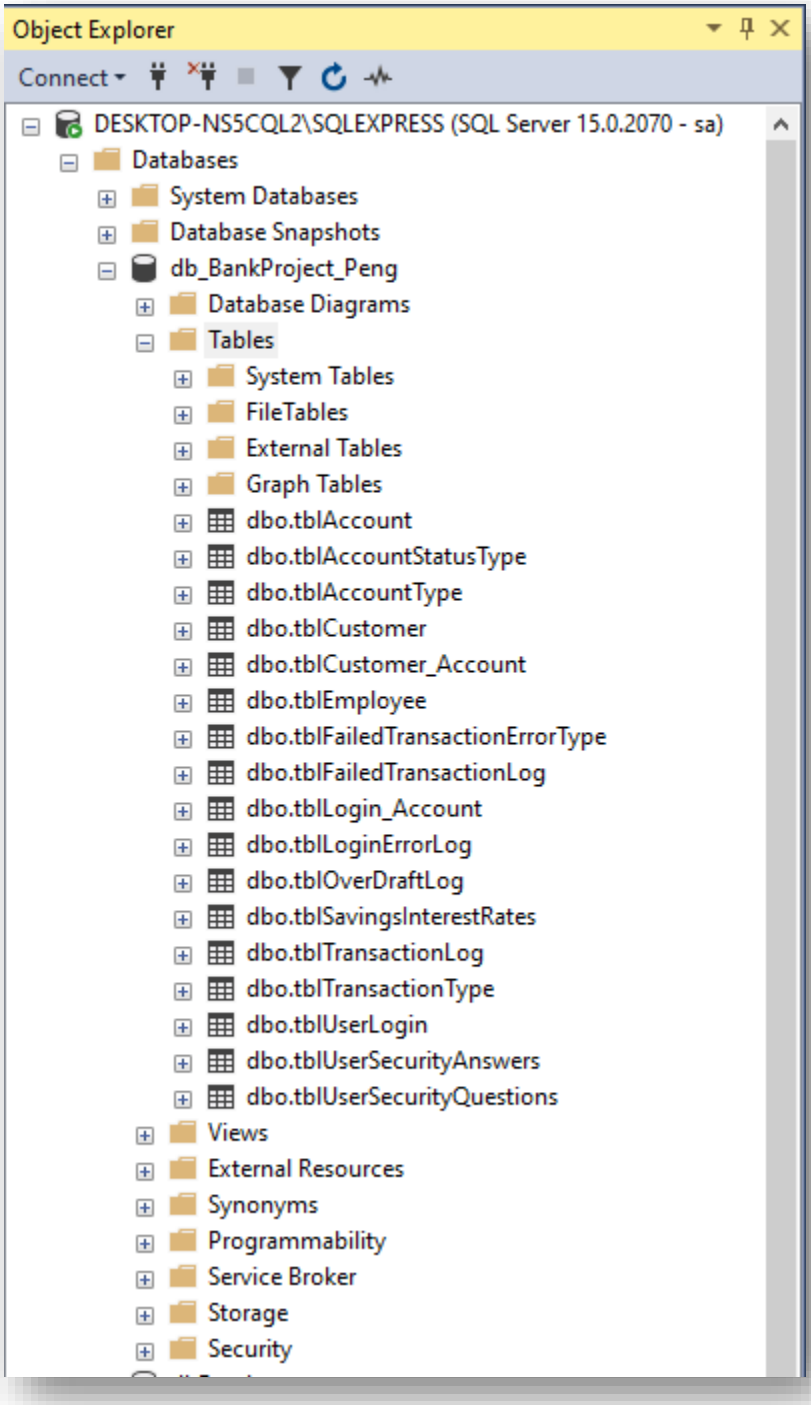
Total # of Questions: [4]

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]

Student Name: Peng Wang Date: Dec 27th 2020

1. Create a database for a banking application called “Bank”. [Basic]



Student Name: Peng Wang Date: Dec 27th 2020

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
```

Results | Messages

	AccountTypeID	AccountTypeDescription
1	0	Checking Account
2	1	Savings Account

```
Table #2: AccountStatusType
```

Results | Messages

	AccountStatusTypeID	AccountStatusDescription
1	0	Inactive Account
2	1	Active Account

```
Table #3: SavingInterestRate
```

Results | Messages

	InterestSavingsRateID	InterestRateValue	InterestRateDescription
1	0	0.000000000	No Interests
2	1	0.012300000	Savings Rate 1
3	2	0.023400000	Savings Rate 2
4	3	0.034500000	Savings Rate 3
5	4	0.045600000	Savings Rate 4
6	5	0.056700000	Savings Rate 5

```
Table #4: Account
```

Student Name: Peng Wang Date: Dec 27th 2020

Results | Messages

	AccountID	CurrentBalance	AccountTypeID	AccountStatusTypeID	InterestSavingsRateID
1	101	998.95	1	0	1
2	102	2099.00	1	0	1
3	103	3198.00	1	1	1
4	104	3699.97	1	1	4
5	105	4999.00	1	1	5
6	106	6000.00	1	1	2
7	107	7000.00	1	1	3
8	108	7698.50	1	0	1
9	109	900.00	0	0	0
10	110	200.00	0	0	0
11	111	300.00	0	1	0
12	112	2400.00	0	1	0
13	113	500.00	0	1	0
14	114	600.00	0	1	0
15	115	700.00	0	1	0
16	116	800.00	0	0	0

Table #5: OverDraftLog

Results | Messages

	AccountID	OverDraftDate	OverDraftAmount	OverDraftTransactionXML
1	101	2020-12-10 00:00:00.000	1500.00	NULL
2	102	2020-12-15 00:00:00.000	2100.00	NULL
3	103	2020-12-20 00:00:00.000	0.00	NULL
4	105	2020-12-16 00:00:00.000	5200.00	NULL
5	107	2020-12-18 00:00:00.000	7100.00	NULL
6	108	2020-12-15 00:00:00.000	8400.00	NULL

Student Name: Peng Wang Date: Dec 27th 2020

```
/*
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

Results | Messages

	UserLoginID	UserName	UserPassword
1	401	User_UserName123	UPS401123
2	402	User_UserName234	UPS402234
3	403	User_UserName345	UPS403345
4	404	User_UserName456	UPS404456
5	405	User_UserName567	UPS405567
6	406	User_UserName678	UPS406678
7	407	User_UserName789	UPS407789
8	408	User_UserName890	UPS408890
9	409	User_UserName012	UPS409012
10	410	User_UserName246	UPS410246
11	411	User_UserName135	UPS411135

Table #7: UserSecurityQuestions

Results | Messages

	UserSecurityQuestionID	UserSecurityQuestion
1	1	What is USQ 1?
2	2	What is USQ 2?
3	3	What is USQ 3?
4	4	What is USQ 4?
5	5	What is USQ 5?
6	6	What is USQ 6?

Table #8: UserSecurityAnswers

Results | Messages

	UserLoginID	UserSecurityAnswer	UserSecurityQuestionID
1	401	The answer is USQ 1.	1
2	402	The answer is USQ 2.	2
3	403	The answer is USQ 3.	3
4	404	The answer is USQ 4.	4
5	405	The answer is USQ 5.	5
6	406	The answer is USQ 6.	6
7	407	The answer is one.	1
8	408	The answer is two.	2
9	409	The answer is three.	3
10	410	The answer is four.	4
11	411	The answer is five.	5

Student Name: Peng Wang Date: Dec 27th 2020

Table #9: Login_Account

Results | Messages

	UserLoginID	AccountID
1	401	102
2	403	101
3	405	101
4	407	104
5	409	106
6	411	108
7	402	102
8	404	101
9	406	103
10	408	105
11	410	107

Table #10: LoginErrorLog

Results | Messages

	ErrorLogID	UserLoginID	ErrorTime	FailedTransactionXML
1	1	402	2020-12-19 08:30:00.000	NULL
2	2	402	2020-12-19 08:35:00.000	NULL
3	3	402	2020-12-19 08:40:00.000	NULL
4	4	405	2020-12-20 08:15:00.000	NULL
5	5	405	2020-12-20 08:22:00.000	NULL
6	6	406	2020-12-20 09:30:00.000	NULL
7	7	401	2020-12-23 00:00:00.000	NULL
8	8	401	2020-12-26 13:00:00.000	NULL
9	9	405	2020-12-26 08:00:00.000	NULL
10	10	409	2020-12-26 10:30:00.000	NULL

Student Name: Peng Wang Date: Dec 27th 2020

```
/*
TABLES FOR CUSTOMER:
----------------------
Table #11: Customer, parent table for Customer_Account
Table #12: Customer_Account
*/
```

Table #11: Customer

Results | Messages

	CustomerID	CustomerAddress1	CustomerAddress2	CustomerFirstName	CustomerMiddleInitial	CustomerLastName	City	State	ZipCode	EmailAddress	HomePhone	CellPhone	WorkPhone	SSN	UserLoginID
1	201	123 King St.	NULL	John	J	Smith	Toronto	ON	A1B 2C3	JS@email.com	NULL	6471111111	NULL	123456789	401
2	202	123 King St.	NULL	Mary	NULL	Smith	Toronto	ON	A1B 2C3	MS@email.com	NULL	647222222	NULL	234567890	402
3	203	456 Queen St.	NULL	Emily	E	Reese	Vancouver	BC	A2B 3C4	ER@email.com	NULL	6473333333	NULL	345678901	403
4	204	456 Queen St.	NULL	Michelle	NULL	Long	Vancouver	BC	A2B 3C4	ML@email.com	NULL	6474444444	NULL	456789012	404
5	205	456 Queen St.	NULL	Sam	S	Lee	Vancouver	BC	A2B 3C4	SL@email.com	NULL	6475555555	NULL	567890123	405
6	206	111 Yonge St.	NULL	Tom	NULL	Walsh	Toronto	ON	A3B 2C3	TW@email.com	NULL	6476666666	NULL	678901234	406
7	207	111 Yonge St.	NULL	Jerry	J	Nicholls	Toronto	ON	A3B 2C3	JN@email.com	NULL	6477777777	NULL	789012345	407
8	208	789 Bloor St.	NULL	Iris	NULL	Ryan	Toronto	ON	A4B 5C6	IR@email.com	NULL	6478888888	NULL	890123456	408
9	209	321 Keel St.	NULL	Nicole	N	Gill	Vaughan	ON	A5B 6C7	NG@email.com	NULL	6479999999	NULL	901234567	409
10	210	678 Hwy 7	NULL	Justine	NULL	Bieber	Vaughan	ON	A6B 7C8	JB@email.com	NULL	6471234567	NULL	012345678	410
11	211	678 Hwy 7	NULL	Hailey	NULL	Baldwin	Vaughan	ON	A6B 7C8	HB@email.com	NULL	6472345678	NULL	135790246	411

Table #12: Customer_Account

Results | Messages

	AccountID	CustomerID
1	102	201
2	101	203
3	101	205
4	104	207
5	106	209
6	108	211
7	109	202
8	111	204
9	112	206
10	113	208
11	114	210
12	116	201
13	109	201
14	110	203
15	111	205
16	113	207
17	113	209
18	115	211
19	116	202
20	102	202
21	101	204
22	103	206
23	105	208
24	107	210

Student Name: Peng Wang Date: Dec 27th 2020

```
/*
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

Results | Messages

	EmployeeID	EmployeeFirstName	EmployeeMiddleInitial	EmployeeLastName	EmployeeIsManager
1	301	Peng	NULL	Wang	1
2	302	Sirena	S	Fairbank	1
3	303	Sacha	NULL	Dean	0
4	304	Juliet	M	Mullins	1
5	305	Ashley	NULL	Poe	0
6	306	Elmer	NULL	Boyd	1

Table #14: TransactionType

Results | Messages

	TransactionTypeID	TransactionTypeName	TransactionTypeDescription	TransactionFeeAmount
1	1	Deposit	Deposit Cash	1.00
2	2	Withdraw	Withdraw Cash	3.00
3	3	Check-in	Deposit by check	5.00
4	4	Check-out	Withdraw by check	5.00
5	5	E-Transfer	Email Transfer	1.50
6	6	Online	Online Transfer	1.00
7	7	Wire	Wire Transfer	2.00

Table #15: TransactionLog

Results | Messages

	TransactionDate	TransactionTypeID	TransactionAmount	NewBalance	AccountID	CustomerID	EmployeeID	UserLoginID	TransactionID
1	2020-12-11 09:20:00.000	7	10000.00	5000.00	108	211	301	411	500
2	2020-12-12 19:20:00.000	6	5000.00	3000.00	107	210	302	410	501
3	2020-12-13 12:20:00.000	7	15000.00	5500.00	106	209	303	409	502
4	2020-12-13 19:20:00.000	5	1000.00	1000.00	105	208	304	408	503
5	2020-12-13 21:20:00.000	4	1000.00	3000.00	104	207	305	407	504
6	2020-12-27 11:20:15.570	7	99.00	6901.00	103	206	306	406	505
7	2020-12-14 19:20:00.000	3	10600.00	5600.00	101	205	305	405	506
8	2020-12-15 09:20:00.000	2	7000.00	3300.00	101	204	304	404	507
9	2020-12-16 09:20:00.000	7	1000.00	900.00	101	203	303	403	508
10	2020-12-16 09:55:00.000	1	500.00	1000.00	102	202	302	402	509
11	2020-12-17 01:55:00.000	7	2000.00	5000.00	102	201	301	401	510
12	2020-12-27 11:55:00.000	2	99.00	5000.00	108	211	301	411	511
13	2020-12-27 13:33:54.253	2	-100.50	4899.50	108	211	301	411	514
14	2020-12-27 13:52:23.487	2	-100.01	2899.99	104	207	301	407	517
15	2020-12-27 14:07:16.560	2	-100.05	799.95	101	204	301	404	518

Table #16: FailedTransactionErrorType

Student Name: Peng Wang Date: Dec 27th 2020

Results | Messages

	FailedTransactionErrorTypeID	FailedTransactionDescription
1	1	Account ID Does Not Exist
2	2	Insufficient Account Balance
3	3	Inactive Account
4	4	Expired Check
5	5	Other

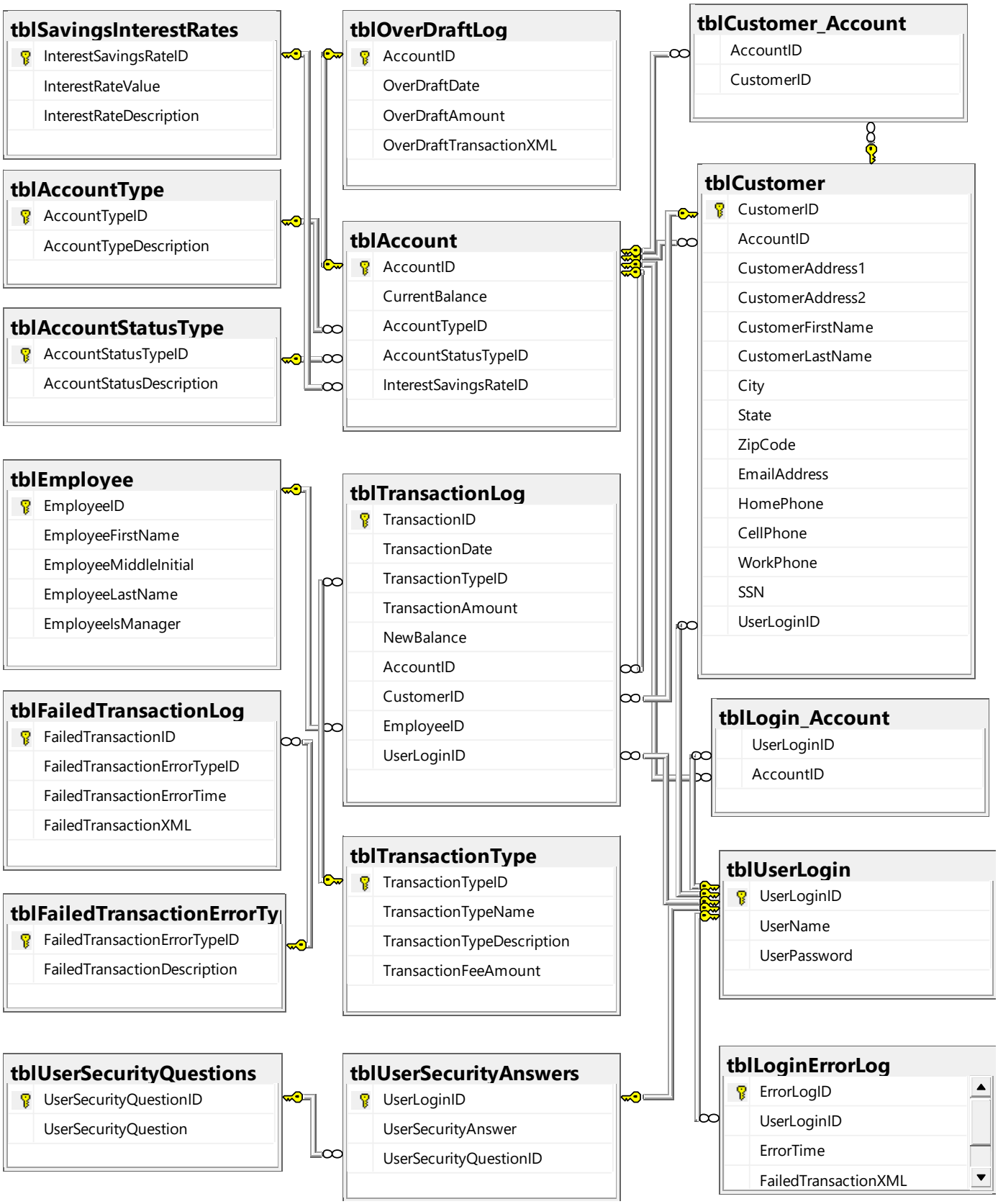
Table #17: FailedTransactionLog

Results | Messages

	FailedTransactionID	FailedTransactionErrorTypeID	FailedTransactionErrorTime	FailedTransactionXML
1	601	5	2020-12-01 12:05:00.000	NULL
2	602	4	2020-12-03 15:05:00.000	NULL
3	603	3	2020-12-05 17:05:00.000	NULL
4	604	2	2020-12-06 11:05:00.000	NULL
5	605	1	2020-12-20 01:05:00.000	NULL

Student Name: Peng Wang Date: Dec 27th 2020

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



Student Name: Peng Wang Date: Dec 27th 2020

METRO COLLEGE OF TECHNOLOGY

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]

Student Name: Peng Wang Date: Dec 27th 2020

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

```
    a.AccountID
from
    tblCustomer c
    join tblCustomer_Account ca
    on c.CustomerID = ca.CustomerID
    join tblAccount a
    on ca.AccountID = a.AccountID
    join tblAccountType atype
    on a.AccountTypeID = atype.AccountTypeID
where
    c.State = 'ON' and atype.AccountTypeDescription = 'Checking Account'
go
```

Results | Messages

	CustomerID	Customer Name	City	State	AccountTypeDescription	AccountID
1	201	John Smith	Toronto	ON	Checking Account	116
2	201	John Smith	Toronto	ON	Checking Account	109
3	202	Mary Smith	Toronto	ON	Checking Account	109
4	202	Mary Smith	Toronto	ON	Checking Account	116
5	206	Tom Walsh	Toronto	ON	Checking Account	112
6	207	Jerry Nicholls	Toronto	ON	Checking Account	113
7	208	Iris Ryan	Toronto	ON	Checking Account	113
8	209	Nicole Gill	Vaughan	ON	Checking Account	113
9	210	Justine Bieber	Vaughan	ON	Checking Account	114
10	211	Hailey Baldwin	Vaughan	ON	Checking Account	115

Student Name: Peng Wang Date: Dec 27th 2020

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.

Results | Messages

	CustomerID	Customer Name	AccountID	AccountTypeID	CurrentBalance	InterestSavingsRateID
1	201	John Smith	102	1	2099.00	1
2	201	John Smith	116	0	800.00	0
3	201	John Smith	109	0	900.00	0
4	202	Mary Smith	116	0	800.00	0
5	202	Mary Smith	102	1	2099.00	1
6	202	Mary Smith	109	0	900.00	0
7	203	Emily Reese	101	1	998.95	1
8	203	Emily Reese	110	0	200.00	0
9	204	Michelle Long	101	1	998.95	1
10	204	Michelle Long	111	0	300.00	0
11	205	Sam Lee	101	1	998.95	1
12	205	Sam Lee	111	0	300.00	0
13	206	Tom Walsh	103	1	3198.00	1
14	206	Tom Walsh	112	0	2400.00	0
15	207	Jerry Nicholls	104	1	3699.97	4
16	207	Jerry Nicholls	113	0	500.00	0
17	208	Iris Ryan	105	1	4999.00	5
18	208	Iris Ryan	113	0	500.00	0
19	209	Nicole Gill	106	1	6000.00	2
20	209	Nicole Gill	113	0	500.00	0
21	210	Justine Bieber	107	1	7000.00	3
22	210	Justine Bieber	114	0	600.00	0
23	211	Hailey Baldwin	108	1	7698.50	1
24	211	Hailey Baldwin	115	0	700.00	0

Student Name: Peng Wang Date: Dec 27th 2020

```
498     a.InterestSavingsRateID = r.InterestSavingsRateID
499 group by
500     c.CustomerID,
501     c.CustomerFirstName,
502     c.CustomerLastName
503 having
504     sum(a.CurrentBalance * (1 + r.InterestRateValue/12)) > 5000
505 go
506
507 select * from vw5000
508 go
```

Results | Messages

	CustomerID	Customer Name	Total Balance incl. Interest
1	206	Tom Walsh	5601.198
2	208	Iris Ryan	5522.4953
3	209	Nicole Gill	6512.00
4	210	Justine Bieber	7620.30
5	211	Hailey Baldwin	8406.1985

Student Name: Peng Wang Date: Dec 27th 2020

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.

Results | Messages

	CustomerID	Customer Name	AccountID	AccountTypeID	CurrentBalance	InterestSavingsRateID
1	201	John Smith	102	1	2099.00	1
2	201	John Smith	116	0	800.00	0
3	201	John Smith	109	0	900.00	0
4	202	Mary Smith	116	0	800.00	0
5	202	Mary Smith	102	1	2099.00	1
6	202	Mary Smith	109	0	900.00	0
7	203	Emily Reese	101	1	998.95	1
8	203	Emily Reese	110	0	200.00	0
9	204	Michelle Long	101	1	998.95	1
10	204	Michelle Long	111	0	300.00	0
11	205	Sam Lee	101	1	998.95	1
12	205	Sam Lee	111	0	300.00	0
13	206	Tom Walsh	103	1	3198.00	1
14	206	Tom Walsh	112	0	2400.00	0
15	207	Jerry Nicholls	104	1	3699.97	4
16	207	Jerry Nicholls	113	0	500.00	0
17	208	Iris Ryan	105	1	4999.00	5
18	208	Iris Ryan	113	0	500.00	0
19	209	Nicole Gill	106	1	6000.00	2
20	209	Nicole Gill	113	0	500.00	0
21	210	Justine Bieber	107	1	7000.00	3
22	210	Justine Bieber	114	0	600.00	0
23	211	Hailey Baldwin	108	1	7698.50	1
24	211	Hailey Baldwin	115	0	700.00	0

Student Name: Peng Wang Date: Dec 27th 2020

```sql
create view vwACTypeCount as
select
    c.CustomerFirstName + ' ' + c.CustomerLastName [Customer Name],
    atype.AccountTypeDescription [Account Type],
    count(c.CustomerID) [Count of Account]
from
    tblCustomer c
    join tblCustomer_Account ca
    on c.CustomerID = ca.CustomerID
    join tblAccount a
    on ca.AccountID = a.AccountID
    join tblAccountType atype
    on a.AccountTypeID = atype.AccountTypeID
group by
    c.CustomerFirstName,
    c.CustomerLastName,
    atype.AccountTypeDescription
go
```

Results | Messages

	Customer Name	Account Type	Count of Account
1	Emily Reese	Checking Account	1
2	Emily Reese	Savings Account	1
3	Hailey Baldwin	Checking Account	1
4	Hailey Baldwin	Savings Account	1
5	Iris Ryan	Checking Account	1
6	Iris Ryan	Savings Account	1
7	Jerry Nicholls	Checking Account	1
8	Jerry Nicholls	Savings Account	1
9	John Smith	Checking Account	2
10	John Smith	Savings Account	1
11	Justine Bieber	Checking Account	1
12	Justine Bieber	Savings Account	1
13	Mary Smith	Checking Account	2
14	Mary Smith	Savings Account	1
15	Michelle Long	Checking Account	1
16	Michelle Long	Savings Account	1
17	Nicole Gill	Checking Account	1
18	Nicole Gill	Savings Account	1
19	Sam Lee	Checking Account	1
20	Sam Lee	Savings Account	1
21	Tom Walsh	Checking Account	1
22	Tom Walsh	Savings Account	1

Student Name: Peng Wang Date: Dec 27th 2020

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.

Results | Messages

	UserLoginID	UserName	UserPassword
1	401	User_UserName123	UPS401123
2	402	User_UserName234	UPS402234
3	403	User_UserName345	UPS403345
4	404	User_UserName456	UPS404456
5	405	User_UserName567	UPS405567
6	406	User_UserName678	UPS406678
7	407	User_UserName789	UPS407789
8	408	User_UserName890	UPS408890
9	409	User_UserName012	UPS409012
10	410	User_UserName246	UPS410246
11	411	User_UserName135	UPS411135

Results | Messages

	UserLoginID	AccountID
1	401	102
2	403	101
3	405	101
4	407	104
5	409	106
6	411	108
7	402	102
8	404	101
9	406	103
10	408	105
11	410	107

Result:

Results | Messages

	AccountID	UserName	UserPassword
1	102	User_UserName123	UPS401123
2	101	User_UserName345	UPS403345
3	101	User_UserName567	UPS405567
4	104	User_UserName789	UPS407789
5	106	User_UserName012	UPS409012
6	108	User_UserName135	UPS411135
7	102	User_UserName234	UPS402234
8	101	User_UserName456	UPS404456
9	103	User_UserName678	UPS406678
10	105	User_UserName890	UPS408890
11	107	User_UserName246	UPS410246

Student Name: Peng Wang Date: Dec 27th 2020

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

Join tables Customer, Account and OverDraftLog.

Pay attention to exclude zero amount.

Results | Messages

	AccountID	OverDraftDate	OverDraftAmount	OverDraftTransactionXML
1	101	2020-12-10 00:00:00.000	1500.00	NULL
2	102	2020-12-15 00:00:00.000	2100.00	NULL
3	103	2020-12-20 00:00:00.000	0.00	NULL
4	105	2020-12-16 00:00:00.000	5200.00	NULL
5	107	2020-12-18 00:00:00.000	7100.00	NULL
6	108	2020-12-15 00:00:00.000	8400.00	NULL

Result:

Results | Messages

	Customer Name	AccountID	OverDraftAmount
1	John Smith	102	2100.00
2	Emily Reese	101	1500.00
3	Sam Lee	101	1500.00
4	Hailey Baldwin	108	8400.00
5	Mary Smith	102	2100.00
6	Michelle Long	101	1500.00
7	Iris Ryan	105	5200.00
8	Justine Bieber	107	7100.00

Student Name: Peng Wang Date: Dec 27th 2020

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
```

Results | Messages

	UserLoginID	UserName	UserPassword
1	401	UserName123	UPS401123
2	402	User_UserName234	UPS402234
3	403	User_UserName345	UPS403345
4	404	User_UserName456	UPS404456
5	405	User_UserName567	UPS405567
6	406	User_UserName678	UPS406678
7	407	User_UserName789	UPS407789
8	408	User_UserName890	UPS408890
9	409	User_UserName012	UPS409012
10	410	User_UserName246	UPS410246
11	411	User_UserName135	UPS411135

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

Student Name: Peng Wang Date: Dec 27th 2020

```
exec spLoginPrefix
```

Messages

```
(1 row affected)

Completion time: 2020-12-27T15:43:22.6819138-05:00
```

```
select * from tblUserLogin
go
```

Results | Messages

	UserLoginID	UserName	UserPassword
1	401	User_UserName123	UPS401123
2	402	User_UserName234	UPS402234
3	403	User_UserName345	UPS403345
4	404	User_UserName456	UPS404456
5	405	User_UserName567	UPS405567
6	406	User_UserName678	UPS406678
7	407	User_UserName789	UPS407789
8	408	User_UserName890	UPS408890
9	409	User_UserName012	UPS409012
10	410	User_UserName246	UPS410246
11	411	User_UserName135	UPS411135

Student Name: Peng Wang Date: Dec 27th 2020

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.

```
    @AID int,
    @FullName nvarchar(50) output
as
begin
    if (@AID in (select AccountID from tblAccount))
        begin
                Select
                    @FullName = c.CustomerFirstName+' '+ISNULL(c.CustomerMiddleInitial, '')+' '+c.CustomerLastName
                    --Use ISNULL to add nothing if middle name is NULL, otherwise will return NULL to full name.
                from
                    tblCustomer c
                    join tblCustomer_Account ca
                    on c.CustomerID = ca.CustomerID
                where ca.AccountID = @AID;
                set @Fullname = replace (@FullName, '  ', ' ')
                    /*in case middle name is null, there will be 2 spaces btw 1st name and 2nd name,
                    use REPLACE to change double space to single space.*/
        end
    else
        begin
            print 'Oops! This Account Id does not exist!'
        end
```

```
--valid account id
Declare @FN nvarchar(50)
exec spFullNameByAID 108, @FN out
Print 'The Full Name is: ' + @FN
```

00 %

Messages

```
The Full Name is: Hailey Baldwin

Completion time: 2020-12-27T15:53:29.5850310-05:00
```

```
--invalid account id
Declare @FN nvarchar(50)
exec spFullNameByAID 999, @FN out
Print 'Full name is ' + @FN
go
```

100 %

Messages

```
Oops! This Account Id does not exist!

Completion time: 2020-12-27T15:54:02.5391111-05:00
```

Student Name: Peng Wang Date: Dec 27th 2020

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
```

```
717 Exec spErrorLog24
718 go
719
```

100 %

Results | Messages

	ErrorLogID	UserLoginID	ErrorTime	FailedTransactionXML
1	11	402	2020-12-26 17:00:00.000	NULL

Compare other records in the log:

Results | Messages

	ErrorLogID	UserLoginID	ErrorTime	FailedTransactionXML
1	1	402	2020-12-19 08:30:00.000	NULL
2	2	402	2020-12-19 08:35:00.000	NULL
3	3	402	2020-12-19 08:40:00.000	NULL
4	4	405	2020-12-20 08:15:00.000	NULL
5	5	405	2020-12-20 08:22:00.000	NULL
6	6	406	2020-12-20 09:30:00.000	NULL
7	7	401	2020-12-23 00:00:00.000	NULL
8	8	401	2020-12-26 13:00:00.000	NULL
9	9	405	2020-12-26 08:00:00.000	NULL
10	10	409	2020-12-26 10:30:00.000	NULL
11	11	402	2020-12-26 17:00:00.000	NULL

Student Name: Peng Wang Date: Dec 27th 2020

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:

	AccountID	CurrentBalance	AccountTypeID	AccountStatusTypeID	InterestSavingsRateID
1	101	998.95	1	0	1
2	102	2099.00	1	0	1
3	103	3198.00	1	1	1
4	104	3699.97	1	1	4
5	105	4999.00	1	1	5

After:

```
exec spDeposit 103, 99
select * from tblAccount--after
go
```

100 %

Results | Messages

	AccountID	CurrentBalance	AccountTypeID	AccountStatusTypeID	InterestSavingsRateID
1	101	998.95	1	0	1
2	102	2099.00	1	0	1
3	103	3297.00	1	1	1
4	104	3699.97	1	1	4
5	105	4999.00	1	1	5

Student Name: Peng Wang Date: Dec 27th 2020

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.

	TransactionDate	TransactionTypeID	TransactionAmount	NewBalance	AccountID	CustomerID	EmployeeID	UserLoginID	TransactionID
1	2020-12-11 09:20:00.000	7	10000.00	5000.00	108	211	301	411	500
2	2020-12-12 19:20:00.000	6	5000.00	3000.00	107	210	302	410	501
3	2020-12-13 12:20:00.000	7	15000.00	5500.00	106	209	303	409	502
4	2020-12-13 19:20:00.000	5	1000.00	1000.00	105	208	304	408	503
5	2020-12-13 21:20:00.000	4	1000.00	3000.00	104	207	305	407	504
6	2020-12-27 11:20:15.570	7	99.00	6901.00	103	206	306	406	505

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.

Student Name: Peng Wang Date: Dec 27th 2020

```
--for each transaction, insert a new record to the transaction log.
insert into tblTransactionLog
(
    TransactionDate,
    TransactionTypeID,
    TransactionAmount,
    NewBalance,
    AccountID,
    CustomerID,
    EmployeeID,
    UserLoginID
)
values
(
    getdate(),
    2,
    -@WithDraw,
    @OldBalance-@WithDraw,
    @AcID,
    @CustID,
    301,
    (select UserLoginID from tblCustomer where CustomerID = @CustID)
)
End
```

Use account 101, Customer 204 as an example:

Student Name: Peng Wang Date: Dec 27th 2020

Before withdraw 100.05:

```
--before: ref AccountID 101, CustID 204
select * from tblAccount
select * from tblTransactionLog

exec spWithdraw 101, 204, 100.05
```

00 %

Results Messages

	AccountID	CurrentBalance	AccountTypeID	AccountStatusTypeID	InterestSavingsRateID
1	101	998.95	1	0	1
2	102	2099.00	1	0	1
3	103	3297.00	1	1	1
4	104	3699.97	1	1	4
5	105	4999.00	1	1	5
6	106	6000.00	1	1	2
7	107	7000.00	1	1	3
8	108	7698.50	1	0	1
9	109	900.00	0	0	0
10	110	200.00	0	0	0
11	111	300.00	0	1	0

	TransactionDate	TransactionTypeID	TransactionAmount	NewBalance	AccountID	CustomerID	EmployeeID	UserLoginID	TransactionID
1	2020-12-11 09:20:00.000	7	10000.00	5000.00	108	211	301	411	500
2	2020-12-12 19:20:00.000	6	5000.00	3000.00	107	210	302	410	501
3	2020-12-13 12:20:00.000	7	15000.00	5500.00	106	209	303	409	502
4	2020-12-13 19:20:00.000	5	1000.00	1000.00	105	208	304	408	503
5	2020-12-13 21:20:00.000	4	1000.00	3000.00	104	207	305	407	504
6	2020-12-27 11:20:15.570	7	99.00	6901.00	103	206	306	406	505
7	2020-12-14 19:20:00.000	3	10600.00	5600.00	101	205	305	405	506
8	2020-12-15 09:20:00.000	2	7000.00	3300.00	101	204	304	404	507
9	2020-12-16 09:20:00.000	7	1000.00	900.00	101	203	303	403	508
10	2020-12-16 09:55:00.000	1	500.00	1000.00	102	202	302	402	509
11	2020-12-17 01:55:00.000	7	2000.00	5000.00	102	201	301	401	510
12	2020-12-27 11:55:00.000	2	99.00	5000.00	108	211	301	411	511
13	2020-12-27 13:33:54.253	2	-100.50	4899.50	108	211	301	411	514
14	2020-12-27 13:52:23.487	2	-100.01	2899.99	104	207	301	407	517
15	2020-12-27 14:07:16.560	2	-100.05	799.95	101	204	301	404	518

Student Name: Peng Wang Date: Dec 27th 2020

After withdraw:

```
802  exec spWithdraw 101, 204, 100.05
```

00 %

Messages

```
(1 row affected)

(1 row affected)

Completion time: 2020-12-27T16:53:55.0568985-05:00
```

```
802  exec spWithdraw 101, 204, 100.05
803
804  --after: ref AccountID 101, CustID 204
805  select * from tblAccount
806  select * from tblTransactionLog
807  go
808
```

00 %

Results | Messages

	AccountID	CurrentBalance	AccountTypeID	AccountStatusTypeID	InterestSavingsRateID
1	101	898.90	1	0	1
2	102	2099.00	1	0	1
3	103	3297.00	1	1	1
4	104	3699.97	1	1	4
5	105	4999.00	1	1	5
6	106	6000.00	1	1	2
7	107	7000.00	1	1	3
8	108	7698.50	1	0	1
9	109	900.00	0	0	0
10	110	200.00	0	0	0
11	111	300.00	0	1	0
12	112	2400.00	0	1	0
13	113	500.00	0	1	0
14	114	600.00	0	1	0

	TransactionDate	TransactionTypeID	TransactionAmount	NewBalance	AccountID	CustomerID	EmployeeID	UserLoginID	TransactionID
4	2020-12-13 19:20:00.000	5	1000.00	1000.00	105	208	304	408	503
5	2020-12-13 21:20:00.000	4	1000.00	3000.00	104	207	305	407	504
6	2020-12-27 11:20:15.570	7	99.00	6901.00	103	206	306	406	505
7	2020-12-14 19:20:00.000	3	10600.00	5600.00	101	205	305	405	506
8	2020-12-15 09:20:00.000	2	7000.00	3300.00	101	204	304	404	507
9	2020-12-16 09:20:00.000	7	1000.00	900.00	101	203	303	403	508
10	2020-12-16 09:55:00.000	1	500.00	1000.00	102	202	302	402	509
11	2020-12-17 01:55:00.000	7	2000.00	5000.00	102	201	301	401	510
12	2020-12-27 11:55:00.000	2	99.00	5000.00	108	211	301	411	511
13	2020-12-27 13:33:54.253	2	-100.50	4899.50	108	211	301	411	514
14	2020-12-27 13:52:23.487	2	-100.01	2899.99	104	207	301	407	517
15	2020-12-27 14:07:16.560	2	-100.05	799.95	101	204	301	404	518
16	2020-12-27 16:53:55.047	2	-100.05	699.90	101	204	301	404	519

Student Name: Peng Wang Date: Dec 27th 2020

Student Name: Peng Wang Date: Dec 27th 2020

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

Report done on Dec 27th 2020.

Student Name: Peng Wang Date: Dec 27th 2020

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

Presentation done on Dec 23rd 2020.

Student Name: Peng Wang Date: Dec 27th 2020