KAZAKH KBTU BRITISH
T E C H N I C A L
U N I V E R S I T Y

JSC "Kazakh British Technical University"
School of Mathematematic and Cybernetics

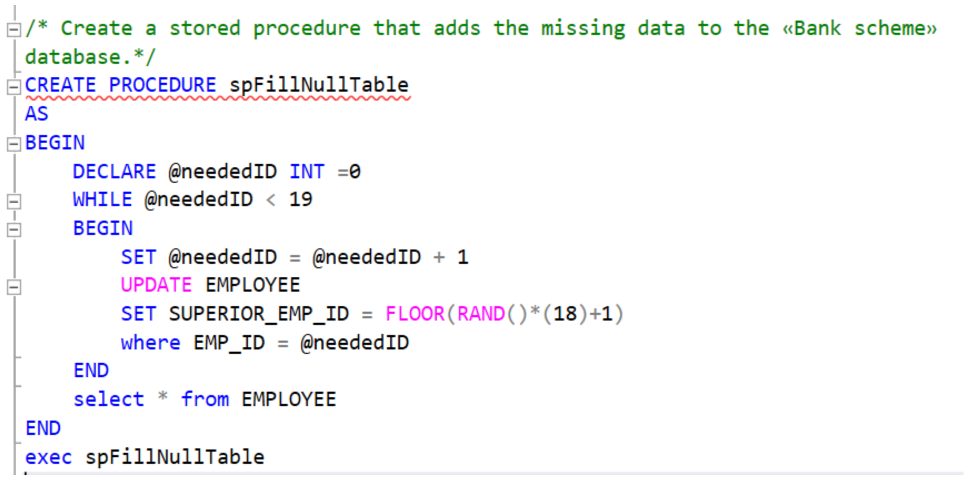
Analysis of Data Bases

Laboratory Work #9

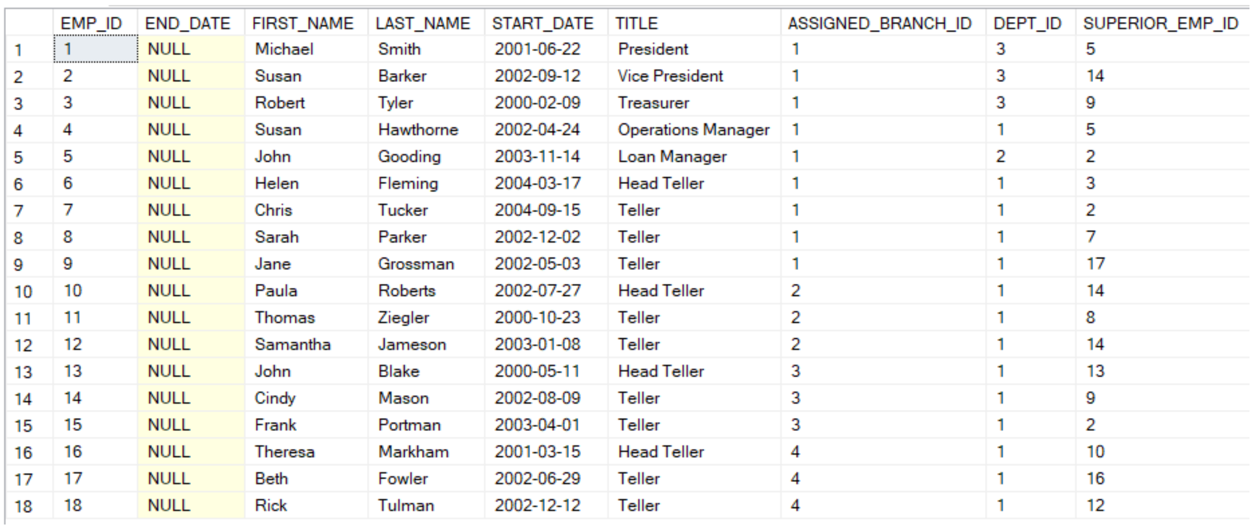
Prepared by: Maratuly Temirbolat

Almaty 2021

1. **Create a stored procedure that adds the missing data to the "Bank scheme" database.** This time in order to demonstrate the above statement I used the loop and filled in the SUPERIOR_EMP_ID column with Random numbers in range from 1 to 18 in EMPLOYEE TABLE.

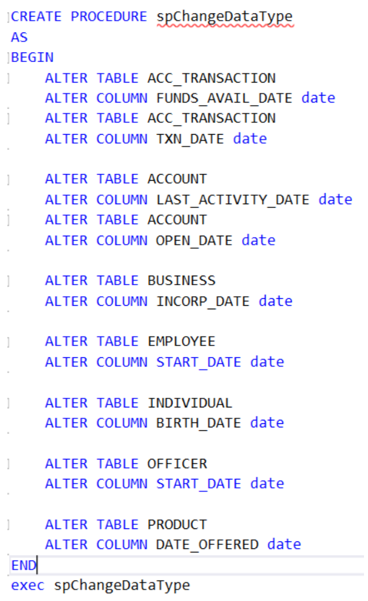


```
/* Create a stored procedure that adds the missing data to the «Bank scheme»
database.*/
CREATE PROCEDURE spFillNullTable
AS
BEGIN
    DECLARE @neededID INT =0
    WHILE @neededID < 19
    BEGIN
        SET @neededID = @neededID + 1
        UPDATE EMPLOYEE
        SET SUPERIOR_EMP_ID = FLOOR(RAND()*(18)+1)
        where EMP_ID = @neededID
    END
    select * from EMPLOYEE
END
exec spFillNullTable
```

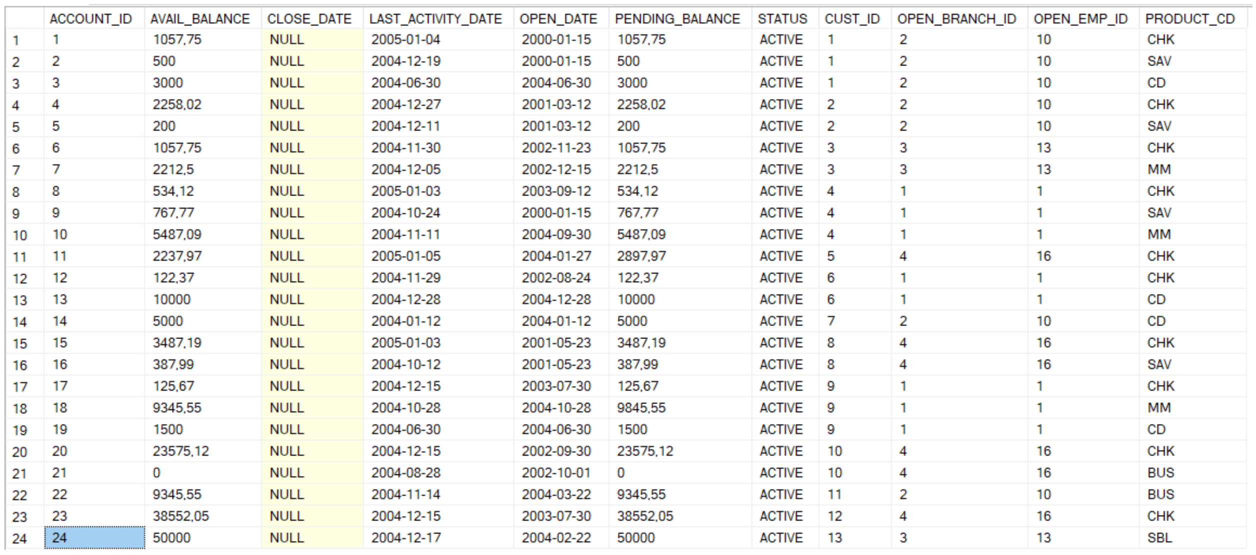


	EMP_ID	END_DATE	FIRST_NAME	LAST_NAME	START_DATE	TITLE	ASSIGNED_BRANCH_ID	DEPT_ID	SUPERIOR_EMP_ID
1	1	NULL	Michael	Smith	2001-06-22	President	1	3	5
2	2	NULL	Susan	Barker	2002-09-12	Vice President	1	3	14
3	3	NULL	Robert	Tyler	2000-02-09	Treasurer	1	3	9
4	4	NULL	Susan	Hawthorne	2002-04-24	Operations Manager	1	1	5
5	5	NULL	John	Gooding	2003-11-14	Loan Manager	1	2	2
6	6	NULL	Helen	Fleming	2004-03-17	Head Teller	1	1	3
7	7	NULL	Chris	Tucker	2004-09-15	Teller	1	1	2
8	8	NULL	Sarah	Parker	2002-12-02	Teller	1	1	7
9	9	NULL	Jane	Grossman	2002-05-03	Teller	1	1	17
10	10	NULL	Paula	Roberts	2002-07-27	Head Teller	2	1	14
11	11	NULL	Thomas	Ziegler	2000-10-23	Teller	2	1	8
12	12	NULL	Samantha	Jameson	2003-01-08	Teller	2	1	14
13	13	NULL	John	Blake	2000-05-11	Head Teller	3	1	13
14	14	NULL	Cindy	Mason	2002-08-09	Teller	3	1	9
15	15	NULL	Frank	Portman	2003-04-01	Teller	3	1	2
16	16	NULL	Theresa	Markham	2001-03-15	Head Teller	4	1	10
17	17	NULL	Beth	Fowler	2002-06-29	Teller	4	1	16
18	18	NULL	Rick	Tulman	2002-12-12	Teller	4	1	12

2. **Create a stored procedure that changes the datetime data type to date for all the corresponding columns of the «Bank scheme».**
Remark: I have changed the datatype from datetime to date only for NONNULL columns. Demonstrate the result illustrating the columns from ACCOUNT table.

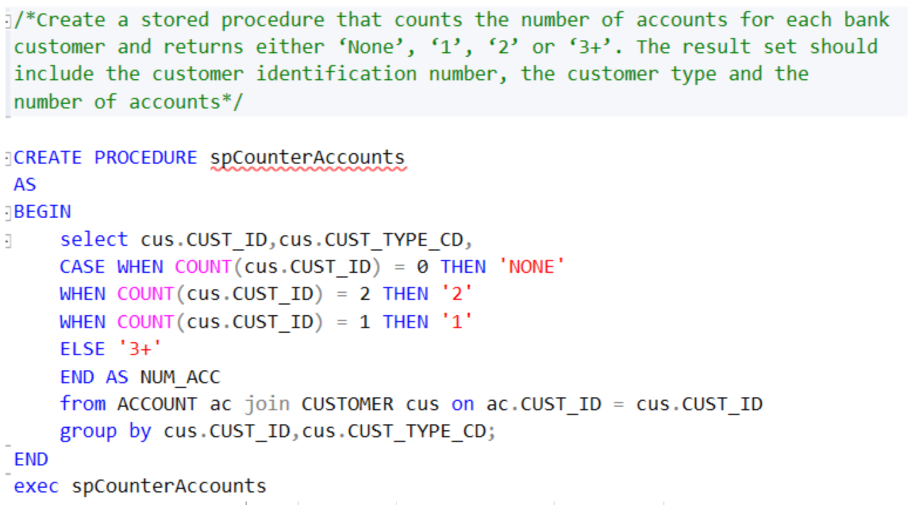


```
CREATE PROCEDURE spChangeDataType
AS
BEGIN
    ALTER TABLE ACC_TRANSACTION
    ALTER COLUMN FUNDS_AVAIL_DATE date
    ALTER TABLE ACC_TRANSACTION
    ALTER COLUMN TXN_DATE date

    ALTER TABLE ACCOUNT
    ALTER COLUMN LAST_ACTIVITY_DATE date
    ALTER TABLE ACCOUNT
    ALTER COLUMN OPEN_DATE date

    ALTER TABLE BUSINESS
    ALTER COLUMN INCORP_DATE date

    ALTER TABLE EMPLOYEE
    ALTER COLUMN START_DATE date

    ALTER TABLE INDIVIDUAL
    ALTER COLUMN BIRTH_DATE date

    ALTER TABLE OFFICER
    ALTER COLUMN START_DATE date

    ALTER TABLE PRODUCT
    ALTER COLUMN DATE_OFFERED date
END
exec spChangeDataType
```

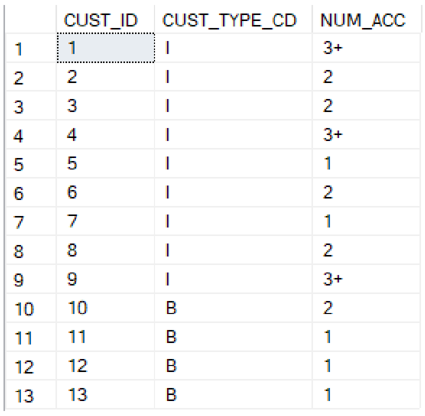


	ACCOUNT_ID	AVAIL_BALANCE	CLOSE_DATE	LAST_ACTIVITY_DATE	OPEN_DATE	PENDING_BALANCE	STATUS	CUST_ID	OPEN_BRANCH_ID	OPEN_EMP_ID	PRODUCT_CD
1	1	1057,75	NULL	2005-01-04	2000-01-15	1057,75	ACTIVE	1	2	10	CHK
2	2	500	NULL	2004-12-19	2000-01-15	500	ACTIVE	1	2	10	SAV
3	3	3000	NULL	2004-06-30	2004-06-30	3000	ACTIVE	1	2	10	CD
4	4	2258,02	NULL	2004-12-27	2001-03-12	2258,02	ACTIVE	2	2	10	CHK
5	5	200	NULL	2004-12-11	2001-03-12	200	ACTIVE	2	2	10	SAV
6	6	1057,75	NULL	2004-11-30	2002-11-23	1057,75	ACTIVE	3	3	13	CHK
7	7	2212,5	NULL	2004-12-05	2002-12-15	2212,5	ACTIVE	3	3	13	MM
8	8	534,12	NULL	2005-01-03	2003-09-12	534,12	ACTIVE	4	1	1	CHK
9	9	767,77	NULL	2004-10-24	2000-01-15	767,77	ACTIVE	4	1	1	SAV
10	10	5487,09	NULL	2004-11-11	2004-09-30	5487,09	ACTIVE	4	1	1	MM
11	11	2237,97	NULL	2005-01-05	2004-01-27	2897,97	ACTIVE	5	4	16	CHK
12	12	122,37	NULL	2004-11-29	2002-08-24	122,37	ACTIVE	6	1	1	CHK
13	13	10000	NULL	2004-12-28	2004-12-28	10000	ACTIVE	6	1	1	CD
14	14	5000	NULL	2004-01-12	2004-01-12	5000	ACTIVE	7	2	10	CD
15	15	3487,19	NULL	2005-01-03	2001-05-23	3487,19	ACTIVE	8	4	16	CHK
16	16	387,99	NULL	2004-10-12	2001-05-23	387,99	ACTIVE	8	4	16	SAV
17	17	125,67	NULL	2004-12-15	2003-07-30	125,67	ACTIVE	9	1	1	CHK
18	18	9345,55	NULL	2004-10-28	2004-10-28	9845,55	ACTIVE	9	1	1	MM
19	19	1500	NULL	2004-06-30	2004-06-30	1500	ACTIVE	9	1	1	CD
20	20	23575,12	NULL	2004-12-15	2002-09-30	23575,12	ACTIVE	10	4	16	CHK
21	21	0	NULL	2004-08-28	2002-10-01	0	ACTIVE	10	4	16	BUS
22	22	9345,55	NULL	2004-11-14	2004-03-22	9345,55	ACTIVE	11	2	10	BUS
23	23	38552,05	NULL	2004-12-15	2003-07-30	38552,05	ACTIVE	12	4	16	CHK
24	24	50000	NULL	2004-12-17	2004-02-22	50000	ACTIVE	13	3	13	SBL

3. **Create a stored procedure that counts the number of accounts for each bank customer and returns either ‘None’, ‘1’, ‘2’ or ‘3+’. The result set should include the customer identification number, the customer type and the number of accounts.**

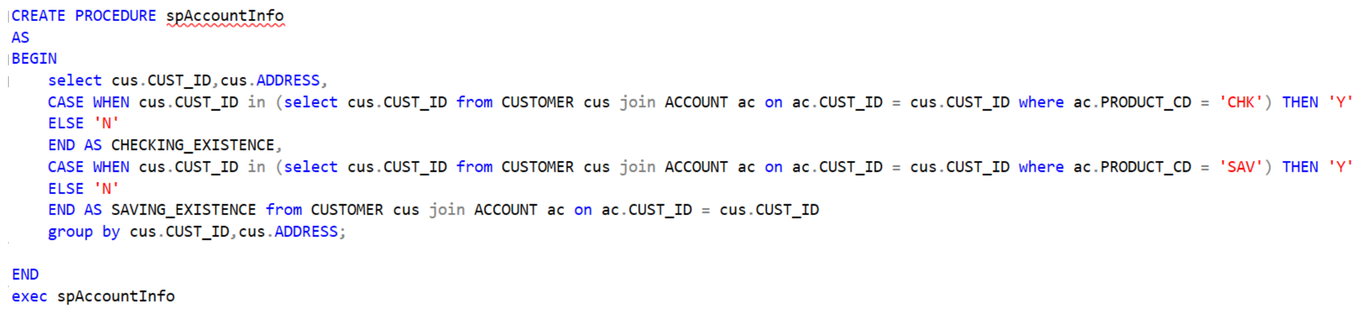


```
/*Create a stored procedure that counts the number of accounts for each bank
customer and returns either ‘None’, ‘1’, ‘2’ or ‘3+’. The result set should
include the customer identification number, the customer type and the
number of accounts*/

CREATE PROCEDURE spCounterAccounts
AS
BEGIN
    select cus.CUST_ID,cus.CUST_TYPE_CD,
    CASE WHEN COUNT(cus.CUST_ID) = 0 THEN 'NONE'
    WHEN COUNT(cus.CUST_ID) = 2 THEN '2'
    WHEN COUNT(cus.CUST_ID) = 1 THEN '1'
    ELSE '3+'
    END AS NUM_ACC
    from ACCOUNT ac join CUSTOMER cus on ac.CUST_ID = cus.CUST_ID
    group by cus.CUST_ID,cus.CUST_TYPE_CD;
END
exec spCounterAccounts
```

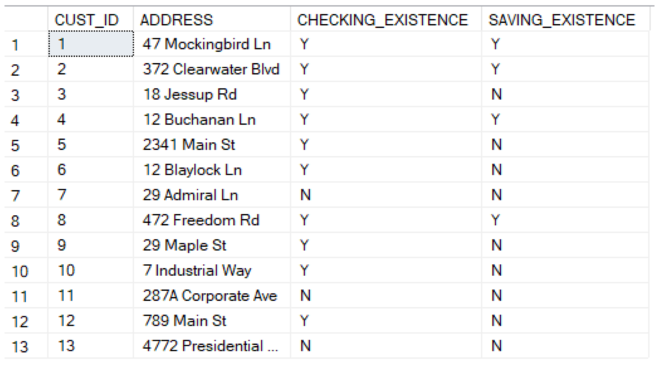


	CUST_ID	CUST_TYPE_CD	NUM_ACC
1	1	I	3+
2	2	I	2
3	3	I	2
4	4	I	3+
5	5	I	1
6	6	I	2
7	7	I	1
8	8	I	2
9	9	I	3+
10	10	B	2
11	11	B	1
12	12	B	1
13	13	B	1

4. **Create a stored procedure that uses two CASE expressions to generate two output columns, one to show whether the customer has any checking accounts and the other to show whether the customer has any savings accounts. If the customer has the account, print ‘Y’, otherwise print ‘N’. The result set should include the following information: the customer ID, their home address, the existence of checking accounts and the existence of savings accounts.**

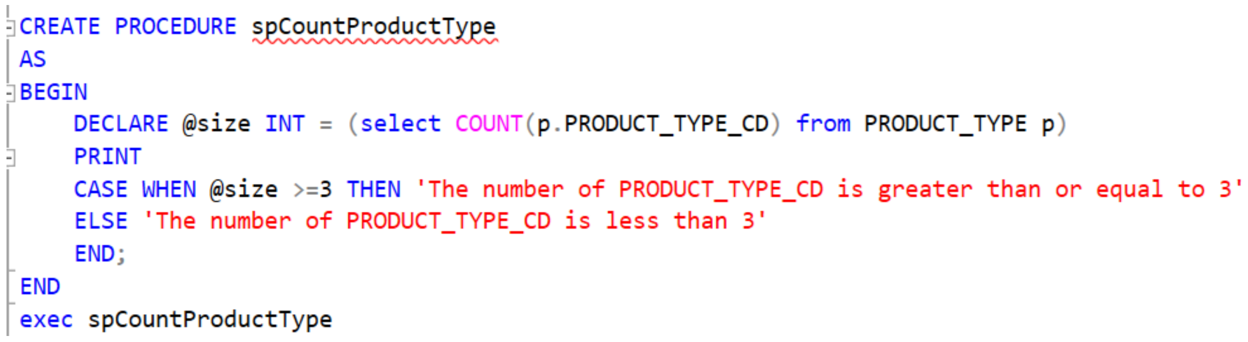


```
CREATE PROCEDURE spAccountInfo
AS
BEGIN
    select cus.CUST_ID,cus.ADDRESS,
    CASE WHEN cus.CUST_ID in (select cus.CUST_ID from CUSTOMER cus join ACCOUNT ac on ac.CUST_ID = cus.CUST_ID where ac.PRODUCT_CD = 'CHK') THEN 'Y'
    ELSE 'N'
    END AS CHECKING_EXISTENCE,
    CASE WHEN cus.CUST_ID in (select cus.CUST_ID from CUSTOMER cus join ACCOUNT ac on ac.CUST_ID = cus.CUST_ID where ac.PRODUCT_CD = 'SAV') THEN 'Y'
    ELSE 'N'
    END AS SAVING_EXISTENCE from CUSTOMER cus join ACCOUNT ac on ac.CUST_ID = cus.CUST_ID
    group by cus.CUST_ID,cus.ADDRESS;

END
exec spAccountInfo
```

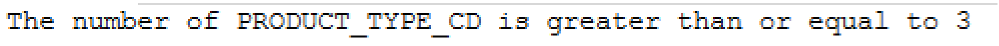


	CUST_ID	ADDRESS	CHECKING_EXISTENCE	SAVING_EXISTENCE
1	1	47 Mockingbird Ln	Y	Y
2	2	372 Clearwater Blvd	Y	Y
3	3	18 Jessup Rd	Y	N
4	4	12 Buchanan Ln	Y	Y
5	5	2341 Main St	Y	N
6	6	12 Blaylock Ln	Y	N
7	7	29 Admiral Ln	N	N
8	8	472 Freedom Rd	Y	Y
9	9	29 Maple St	Y	N
10	10	7 Industrial Way	Y	N
11	11	287A Corporate Ave	N	N
12	12	789 Main St	Y	N
13	13	4772 Presidential ...	N	N

5. **Create a stored procedure that declares a variable and set it to the count of all PRODUCT_TYPE_CD in the Product_Type table. If the count is greater than or equal to 3, the stored procedure should display a message that says, "The number of PRODUCT_TYPE_CD is greater than or equal to 3". Otherwise, it should say, "The number of PRODUCT_TYPE_CD is less than 3".**

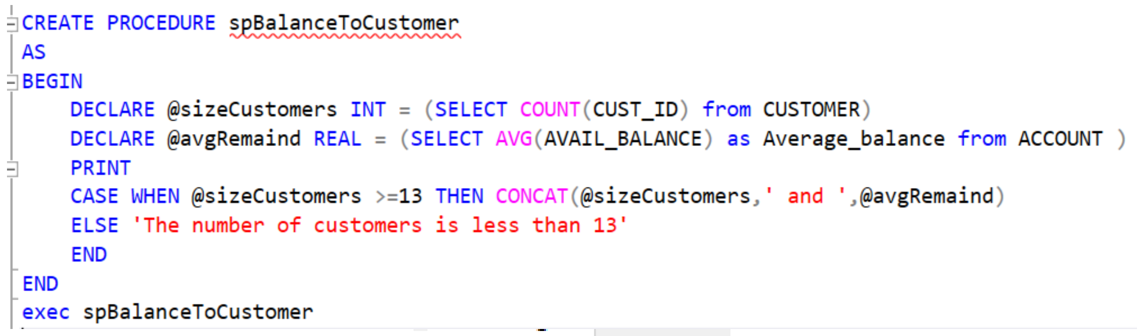


```
CREATE PROCEDURE spCountProductType
AS
BEGIN
    DECLARE @size INT = (select COUNT(p.PRODUCT_TYPE_CD) from PRODUCT_TYPE p)
    PRINT
    CASE WHEN @size >=3 THEN 'The number of PRODUCT_TYPE_CD is greater than or equal to 3'
    ELSE 'The number of PRODUCT_TYPE_CD is less than 3'
    END;
END
exec spCountProductType
```

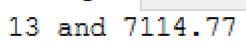


```
The number of PRODUCT_TYPE_CD is greater than or equal to 3
```

6. **Create a stored procedure that uses two variables to store:**
   a) the count of all of the customers in the Customer table;
   b) the average avail balance for each customer. If the customers count is greater than or equal to 13, the stored procedure should display a result set that displays the values of both variables. Otherwise, the procedure should display a result set that displays a message that says, "The number of customers is less than 13".

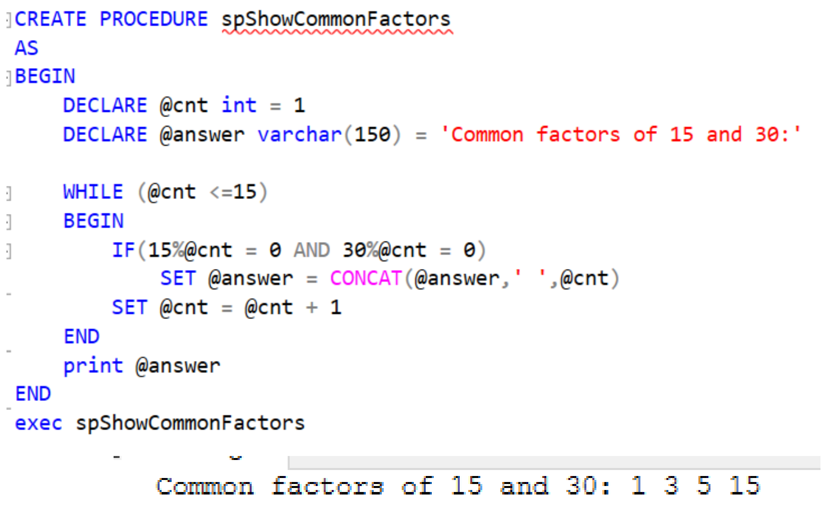


```
CREATE PROCEDURE spBalanceToCustomer
AS
BEGIN
    DECLARE @sizeCustomers INT = (SELECT COUNT(CUST_ID) from CUSTOMER)
    DECLARE @avgRemaind REAL = (SELECT AVG(AVAIL_BALANCE) as Average_balance from ACCOUNT )
    PRINT
    CASE WHEN @sizeCustomers >=13 THEN CONCAT(@sizeCustomers,' and ',@avgRemaind)
    ELSE 'The number of customers is less than 13'
    END
END
exec spBalanceToCustomer
```



```
13 and 7114.77
```

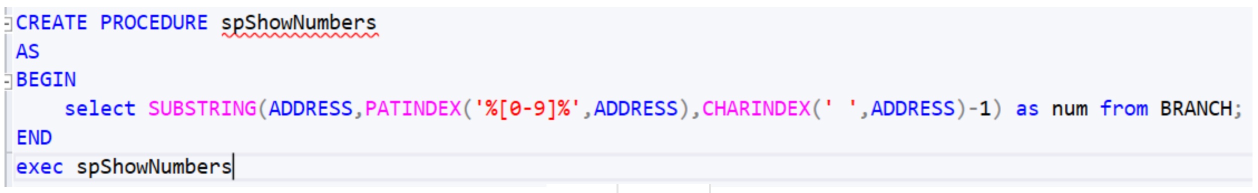
7. **Create a stored procedure that calculates the common factors between 15 and 30. This procedure should display a string that displays the common factors in this form: Common factors of 15 and 30: 1 3 5 15**



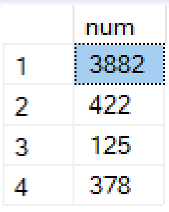
```
CREATE PROCEDURE spShowCommonFactors
AS
BEGIN
    DECLARE @cnt int = 1
    DECLARE @answer varchar(150) = 'Common factors of 15 and 30:'

    WHILE (@cnt <=15)
    BEGIN
        IF(15%@cnt = 0 AND 30%@cnt = 0)
            SET @answer = CONCAT(@answer,' ',@cnt)
        SET @cnt = @cnt + 1
    END
    print @answer
END
exec spShowCommonFactors
```

```
Common factors of 15 and 30: 1 3 5 15
```

8. **Create a stored procedure that shows all numeric characters from the entire string. You can use the ADDRESS columns in the «Bank scheme» database or any row of your choice.**

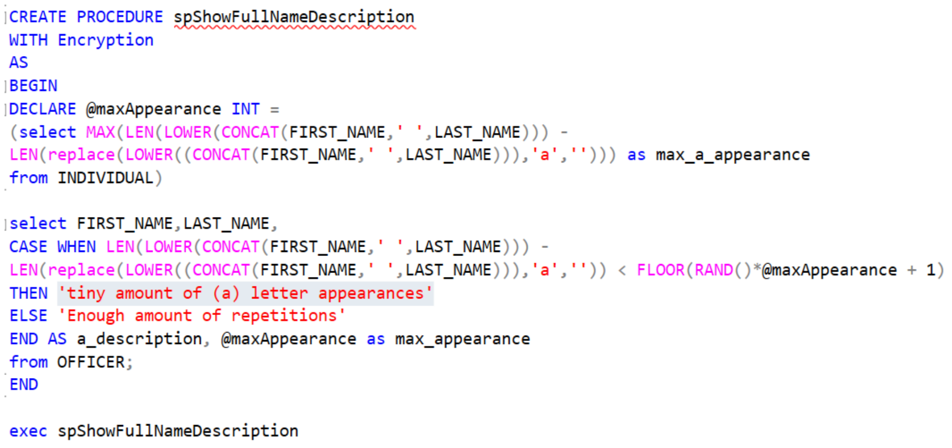


```
CREATE PROCEDURE spShowNumbers
AS
BEGIN
    select SUBSTRING(ADDRESS,PATINDEX('%[0-9]%',ADDRESS),CHARINDEX(' ',ADDRESS)-1) as num from BRANCH;
END
exec spShowNumbers
```

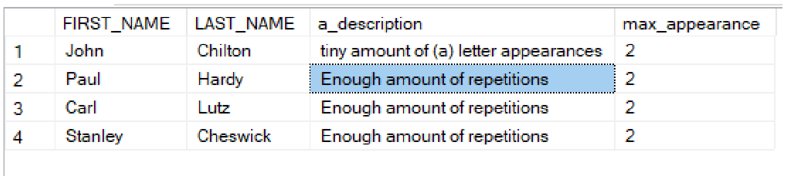


	num
1	3882
2	422
3	125
4	378

9. **Create a stored procedure for the «Bank scheme» database of your choice. Condition: the procedure must be encrypted.**
Description of the Task: Create a stored procedure which is encrypted and shows the First names, last names, description of their full names and maximum appearance of the ‘a’ letter of the OFFICER Table. Take the max possible appearance of the ‘a’ letter from the CONCATINATION of the FIRST_NAME and Last_NAME from INDIVIDUAL table. If the appearance of the ‘a’ exceeds of equal the quantity that is considered as random number from 0 to amount of maximum appearance from the INDIVIDUALS then it shows ‘Enough amount of repetitions’ otherwise 'tiny amount of (a) letter appearances'. Floor the random number.



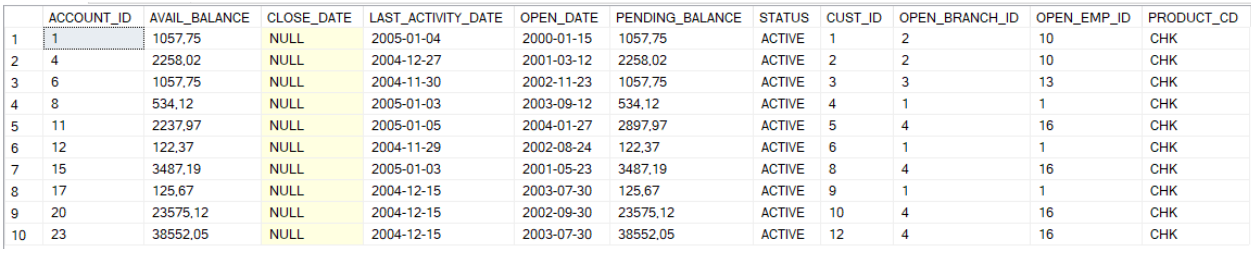
```
CREATE PROCEDURE spShowFullNameDescription
WITH Encryption
AS
BEGIN
DECLARE @maxAppearance INT =
(select MAX(LEN(LOWER(CONCAT(FIRST_NAME,' ',LAST_NAME))) -
LEN(replace(LOWER((CONCAT(FIRST_NAME,' ',LAST_NAME))),'a',''))) as max_a_appearance
from INDIVIDUAL)

select FIRST_NAME,LAST_NAME,
CASE WHEN LEN(LOWER(CONCAT(FIRST_NAME,' ',LAST_NAME))) -
LEN(replace(LOWER((CONCAT(FIRST_NAME,' ',LAST_NAME))),'a','')) < FLOOR(RAND()*@maxAppearance + 1)
THEN 'tiny amount of (a) letter appearances'
ELSE 'Enough amount of repetitions'
END AS a_description, @maxAppearance as max_appearance
from OFFICER;
END

exec spShowFullNameDescription
```



	FIRST_NAME	LAST_NAME	a_description	max_appearance
1	John	Chilton	tiny amount of (a) letter appearances	2
2	Paul	Hardy	Enough amount of repetitions	2
3	Carl	Lutz	Enough amount of repetitions	2
4	Stanley	Cheswick	Enough amount of repetitions	2

10. **Create two stored procedures for the «Bank scheme» database. Condition: one procedure must call another.**
    Description of the own exercise: Create two stored procedures where one DECLARES 2 variables (First variable is responsible to keep the maximum number of the most popular account and Second to store the type(short description) of the most popular account) then this procedure calls another one to show all the accounts with the appropriate type of the account that was sent from the last procedure.

```
CREATE PROCEDURE spFindPopularTypeAccount
AS
BEGIN
    DECLARE @maxNumb INT = (select max(numb_account) from  (select COUNT(ACCOUNT_ID) as numb_account from ACCOUNT group by PRODUCT_CD)a);
    DECLARE @popType varchar(10) =( select TOP(1) PRODUCT_CD from ACCOUNT group by PRODUCT_CD having COUNT(ACCOUNT_ID) = @maxNumb);
    exec spShowAccounts @typeAccount = @popType
END

CREATE PROCEDURE spShowAccounts
@typeAccount varchar(10)
AS
BEGIN
    select * from ACCOUNT where PRODUCT_CD = @typeAccount;
END

exec spFindPopularTypeAccount
```



	ACCOUNT_ID	AVAIL_BALANCE	CLOSE_DATE	LAST_ACTIVITY_DATE	OPEN_DATE	PENDING_BALANCE	STATUS	CUST_ID	OPEN_BRANCH_ID	OPEN_EMP_ID	PRODUCT_CD
1	1	1057,75	NULL	2005-01-04	2000-01-15	1057,75	ACTIVE	1	2	10	CHK
2	4	2258,02	NULL	2004-12-27	2001-03-12	2258,02	ACTIVE	2	2	10	CHK
3	6	1057,75	NULL	2004-11-30	2002-11-23	1057,75	ACTIVE	3	3	13	CHK
4	8	534,12	NULL	2005-01-03	2003-09-12	534,12	ACTIVE	4	1	1	CHK
5	11	2237,97	NULL	2005-01-05	2004-01-27	2897,97	ACTIVE	5	4	16	CHK
6	12	122,37	NULL	2004-11-29	2002-08-24	122,37	ACTIVE	6	1	1	CHK
7	15	3487,19	NULL	2005-01-03	2001-05-23	3487,19	ACTIVE	8	4	16	CHK
8	17	125,67	NULL	2004-12-15	2003-07-30	125,67	ACTIVE	9	1	1	CHK
9	20	23575,12	NULL	2004-12-15	2002-09-30	23575,12	ACTIVE	10	4	16	CHK
10	23	38552,05	NULL	2004-12-15	2003-07-30	38552,05	ACTIVE	12	4	16	CHK