

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)**

**Факультет
Образовательная программа**

**Инфокоммуникационных технологий
11.03.02 Программирование в
инфокоммуникационных системах**

ОТЧЕТ
по лабораторной работе 6
по дисциплине «Разработка баз данных»

Выполнил: студент группы К33202
Рогозина Вероника Сергеевна
Проверил: ст. преподаватель Осетрова И.С.

Санкт-Петербург
2024

1. Цель работы

Целью данной лабораторной работы является реализация пользовательских функций и процедур.

2. Задачи, решаемые при выполнении работы

- 2.1. Создание хранимой процедуры в SSMS.
- 2.2. Создание хранимой процедуры с помощью Query Editor.
- 2.3. Создание скалярной пользовательской функции.
- 2.4. Функция, возвращающая табличное значение.

3. Объект исследования

Объектом исследования в данной лабораторной работе являются способы реализации пользовательских функций и процедур.

4. Исходные данные

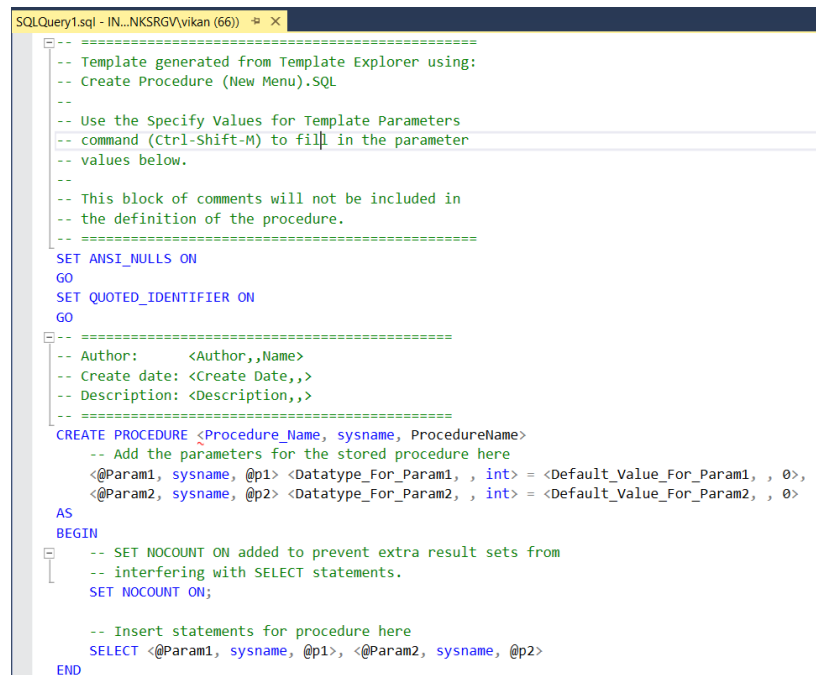
Инструкция к лабораторной работе, инструмент Microsoft SQL Server Management Studio 2019, база данных, таблицы, индексы, представления, созданные в предыдущих лабораторных работах, встроенные шаблоны SSMS.

5. Выполнение работы

5.1 Создание хранимой процедуры в SSMS

В ходе выполнения данной задачи необходимо ознакомиться со способом создания хранимой процедуры с помощью инструмента Microsoft SQL Server Management Studio.

5.1.1. Код основного шаблона хранимой процедуры представлен на рисунке 1.



```
SQLQuery1.sql - IN...NKSRGV\vikon (66)
--
-- Template generated from Template Explorer using:
-- Create Procedure (New Menu).SQL
--
-- Use the Specify Values for Template Parameters
-- command (Ctrl-Shift-M) to fill in the parameter
-- values below.
--
-- This block of comments will not be included in
-- the definition of the procedure.
--
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
--
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
--
CREATE PROCEDURE <Procedure_Name, sysname>, ProcedureName>
-- Add the parameters for the stored procedure here
    <@Param1, sysname>, @p1> <Datatype_For_Param1, , int> = <Default_Value_For_Param1, , 0>,
    <@Param2, sysname>, @p2> <Datatype_For_Param2, , int> = <Default_Value_For_Param2, , 0>
AS
BEGIN
-- SET NOCOUNT ON added to prevent extra result sets from
-- interfering with SELECT statements.
    SET NOCOUNT ON;

-- Insert statements for procedure here
    SELECT <@Param1, sysname>, @p1>, <@Param2, sysname>, @p2>
END
```

Рисунок 1 – Код шаблона хранимой процедуры

5.1.2. Открытие ОД «Specify Values for Template Parameters» и ввод необходимых настроек в данном ОД представлены на рисунке 2. Для корректной работы шаблона необходимо заполнить поля Author – имя и фамилия автора, Create Date – дата создания процедуры, Description – описание процедуры, Procedure_Name – имя создаваемой процедуры, @Param1, @Param2 – столбцы, для которых будут использованы значения двух первых входных параметров, а также Datatype_For_Param1 и Datatype_For_Param2 – типы данных двух первых входных столбцов. Значения по умолчанию для данных столбцов необходимо удалить.

Parameter	Type	Value
Author		Rogozina Veronika
Create Date		09-04-2024
Description		My first procedure
Procedure_Name	sysname	usp_insCustomer
@Param1	sysname	@FirstName
Datatype_For_Param1		nvarchar(50)
Default_Value_For_P...		
@Param2	sysname	@LastName
Datatype_For_Param2		nvarchar(50)
Default_Value_For_P...		

Рисунок 2 – Настройка ОД «Specify Values for Template Parameters»

5.1.3. Шаблон, получившийся после применения всех настроек их пункта 5.1.2 представлен на рисунке 3.

```

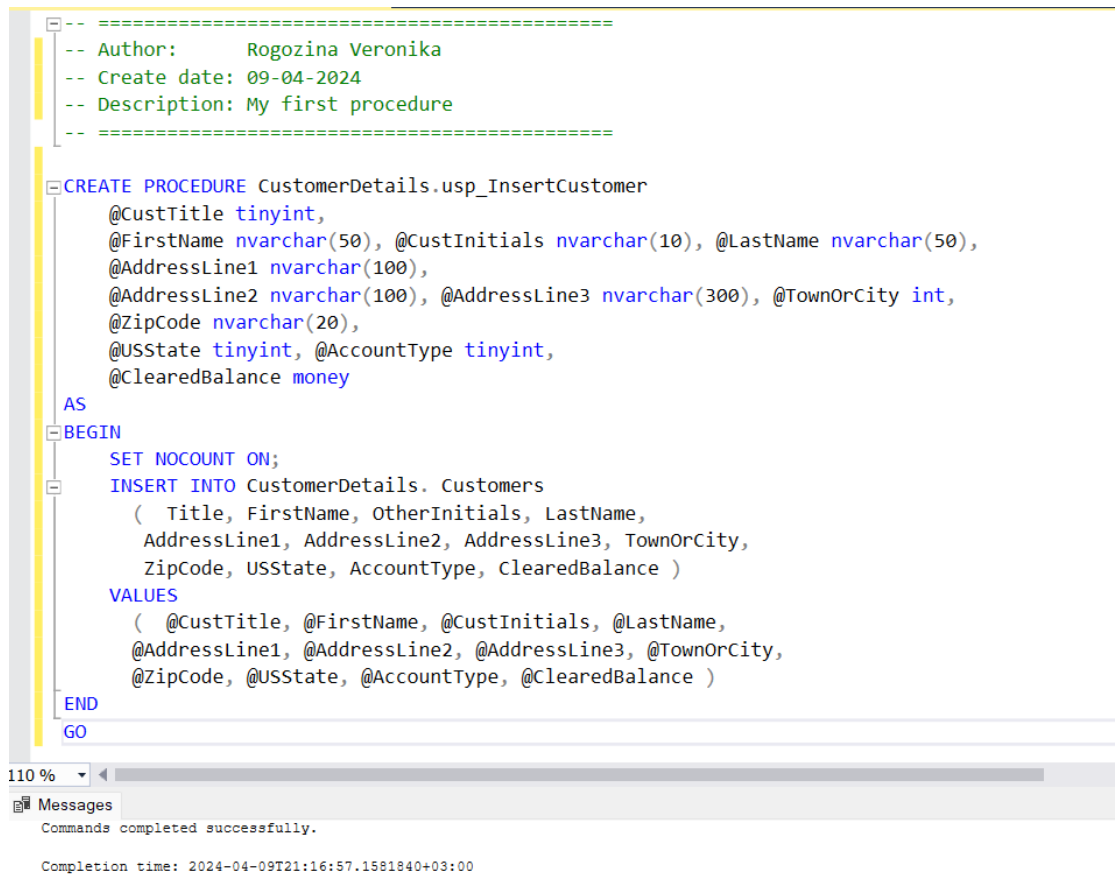
-- -----
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- -----
-- Author:      Rogozina Veronika
-- Create date: 09-04-2024
-- Description: My first procedure
-- -----
CREATE PROCEDURE usp_insCustomer
-- Add the parameters for the stored procedure here
    @FirstName nvarchar(50) = ,
    @LastName nvarchar(50) = 
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT @FirstName, @LastName
END
GO

```

Рисунок 3 – Шаблон

5.1.4. Для полной версии шаблона необходимо добавить к имени процедуры имя схемы и определить оставшиеся параметры, которые следует сгруппировать. Процедура при этом будет состоять из одной инструкции INSERT. Итоговый вариант кода шаблона представлен на рисунке 4.



```
-- =====
-- Author:      Rogozina Veronika
-- Create date: 09-04-2024
-- Description: My first procedure
-- =====

CREATE PROCEDURE CustomerDetails.usp_InsertCustomer
    @CustTitle tinyint,
    @FirstName nvarchar(50), @CustInitials nvarchar(10), @LastName nvarchar(50),
    @AddressLine1 nvarchar(100),
    @AddressLine2 nvarchar(100), @AddressLine3 nvarchar(300), @TownOrCity int,
    @ZipCode nvarchar(20),
    @USState tinyint, @AccountType tinyint,
    @ClearedBalance money
AS
BEGIN
    SET NOCOUNT ON;
    INSERT INTO CustomerDetails.Customers
        ( Title, FirstName, OtherInitials, LastName,
          AddressLine1, AddressLine2, AddressLine3, TownOrCity,
          ZipCode, USState, AccountType, ClearedBalance )
    VALUES
        ( @CustTitle, @FirstName, @CustInitials, @LastName,
          @AddressLine1, @AddressLine2, @AddressLine3, @TownOrCity,
          @ZipCode, @USState, @AccountType, @ClearedBalance )
END
GO
```

110 %

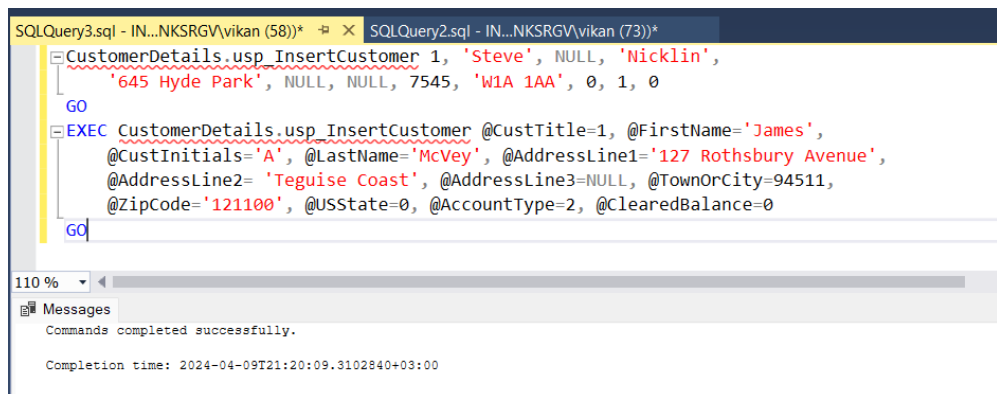
Messages

Commands completed successfully.

Completion time: 2024-04-09T21:16:57.1581840+03:00

Рисунок 4 – Код шаблона

5.1.5. Код запроса, передающего в процедуру входные параметры в том же порядке, в котором они определены в хранимой процедуре, а затем перечисляющего имена параметров и их значения, представлен на рисунке 5.



```
SQLQuery3.sql - IN...NKSRRGV\vikon (58))*  SQLQuery2.sql - IN...NKSRRGV\vikon (73))*

CustomerDetails.usp_InsertCustomer 1, 'Steve', NULL, 'Nicklin',
    '645 Hyde Park', NULL, NULL, 7545, 'W1A 1AA', 0, 1, 0
GO

EXEC CustomerDetails.usp_InsertCustomer @CustTitle=1, @FirstName='James',
    @CustInitials='A', @LastName='McVey', @AddressLine1='127 Rothsbury Avenue',
    @AddressLine2='Teguisse Coast', @AddressLine3=NULL, @TownOrCity=94511,
    @ZipCode='121100', @USState=0, @AccountType=2, @ClearedBalance=0
GO
```

110 %

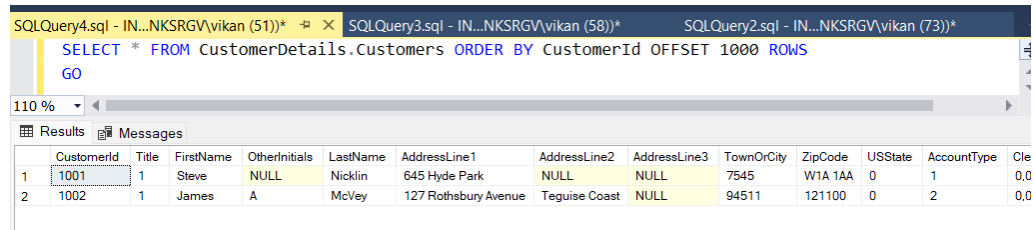
Messages

Commands completed successfully.

Completion time: 2024-04-09T21:20:09.3102840+03:00

Рисунок 5 – Код запроса

5.1.6. Код запроса, который выводит отсортированные по полю CustomerId значения из таблицы CustomerDetails.Customers, и вывод значений из таблицы представлены на рисунке 6.



The screenshot shows a SQL Server Query Editor window with a query and its results. The query is:

```
SELECT * FROM CustomerDetails.Customers ORDER BY CustomerId OFFSET 1000 ROWS  
GO
```

The results are displayed in a table with the following columns: CustomerId, Title, FirstName, OtherInitials, LastName, AddressLine1, AddressLine2, AddressLine3, TownOrCity, ZipCode, USState, AccountType, and Cle. The results are sorted by CustomerId.

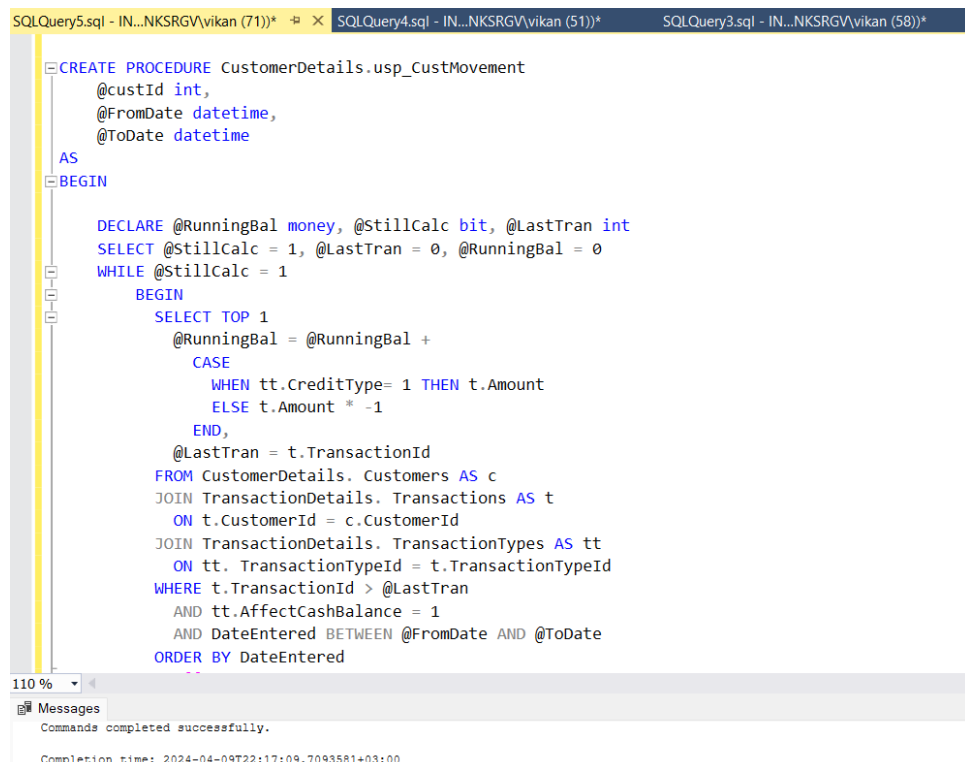
	CustomerId	Title	FirstName	OtherInitials	LastName	AddressLine1	AddressLine2	AddressLine3	TownOrCity	ZipCode	USState	AccountType	Cle
1	1001	1	Steve	NULL	Nicklin	645 Hyde Park	NULL	NULL	7545	W1A 1AA	0	1	0,0
2	1002	1	James	A	McVey	127 Rothsbury Avenue	Teguisse Coast	NULL	94511	121100	0	2	0,0

Рисунок 6 – Код запроса и вывод значений из таблицы CustomerDetails.Customers

5.2. Создание хранимой процедуры с помощью Query Editor.

В ходе выполнения данной задачи необходимо ознакомиться со способом создания хранимой процедуры с помощью Query Editor.

5.2.1. Код запроса, который создает хранимую процедуру с тремя входными параметрами, которая возвращает итоговый баланс по счету клиента за указанный период времени для дальнейшего начисления процентов по кредиту или вкладу, представлен на рисунке 7.



The screenshot shows a SQL Server Query Editor window with the code for creating a stored procedure. The code is:

```
CREATE PROCEDURE CustomerDetails.usp_CustMovement  
    @custId int,  
    @FromDate datetime,  
    @ToDate datetime  
AS  
BEGIN  
    DECLARE @RunningBal money, @StillCalc bit, @LastTran int  
    SELECT @StillCalc = 1, @LastTran = 0, @RunningBal = 0  
    WHILE @StillCalc = 1  
    BEGIN  
        SELECT TOP 1  
            @RunningBal = @RunningBal +  
                CASE  
                    WHEN tt.CreditType= 1 THEN t.Amount  
                    ELSE t.Amount * -1  
                END,  
            @LastTran = t.TransactionId  
        FROM CustomerDetails. Customers AS c  
        JOIN TransactionDetails. Transactions AS t  
            ON t.CustomerId = c.CustomerId  
        JOIN TransactionDetails. TransactionTypes AS tt  
            ON tt.TransactionTypeId = t.TransactionTypeId  
        WHERE t.TransactionId > @LastTran  
            AND tt.AffectCashBalance = 1  
            AND DateEntered BETWEEN @FromDate AND @ToDate  
        ORDER BY DateEntered  
    END  
END
```

The Messages pane at the bottom shows the following messages:

```
Commands completed successfully.  
Completion time: 2024-04-09T22:17:09.7093581+03:00
```

Рисунок 7 – Код запроса для создания хранимой процедуры

5.2.2. Код запроса, который добавляет данные в таблицу для тестирования процедуры, представлен на рисунке 8.

```

SQLQuery6.sql - IN...NKSRGV\vikan (65))* X SQLQuery5.sql - IN...NKSRGV\vikan (71))* SQLQuery4.sql - IN...NKSRGV\vikan (70))*
INSERT INTO TransactionDetails.Transactions (CustomerId,
TransactionTypeId, DateEntered, Amount, RelatedProductId)
VALUES (1,1, '01-08-2015', 100.00,1), (1,1, '03-08-2015', 75.67,1),
(1,2, '05-08-2015', 35.20,1), (1,2, '06-08-2015', 20.00, 1)
GO

```

110 %

Messages

(4 rows affected)

Completion time: 2024-04-09T22:20:16.2445598+03:00

Рисунок 8 – Код запроса для добавления данных в таблицу

5.2.3. Выполнение процедуры и получившиеся значения представлены на рисунке 9. Данное значение соответствует заданному в условии, запрос выполнен корректно.

```

SQLQuery7.sql - IN...NKSRGV\vikan (77))* X SQLQuery6.sql - IN...NKSRGV\vikan (65))* SQLQuery5.sql - IN...NKSRGV\vikan (70))*
EXEC CustomerDetails.usp_CustMovement 1, '01-08-2015', '31-08-2015'
GO

```

110 %

Results Messages

	End Balance
1	-120.47

Рисунок 9 – Выполнение процедуры

5.3. Создание скалярной пользовательской функции.

В ходе выполнения данной задачи необходимо ознакомиться со способом создания скалярной пользовательской функции.

5.3.1. Код запроса, создающего скалярную пользовательскую функцию, которая осуществляет начисление процентов по определенной ставке на заданную сумму денежных средств за определенный период времени, представлен на рисунке 10.

```

SQLQuery8.sql - IN...NKSRGV\vikan (81))* X SQLQuery7.sql - IN...NKSRGV\vikan (77))* SQLQuery6.sql - IN...NKSRGV\vikan (70))*
CREATE FUNCTION TransactionDetails.ufn_IntCalc
( @InterestRate decimal(6,3) = 10, @Amount money, @FromDate date, @ToDate date )
RETURNS money
WITH EXECUTE AS CALLER
AS
BEGIN
DECLARE @IntCalculated money
SELECT @IntCalculated = @Amount * (((@InterestRate/100.00) *
(DATEDIFF(d,@FromDate, @ToDate) / 365.00))
RETURN (ISNULL(@IntCalculated, 0))
END
GO

```

110 %

Messages

Commands completed successfully.

Completion time: 2024-04-09T22:27:03.5955015+03:00

Рисунок 10 – Код запроса для создания функции

5.3.2. Тестирование функции на определённом наборе значений представлено на рисунке 11. Полученное значение соответствует заданному в тексте инструкции к лабораторной работе, функция работает корректно.

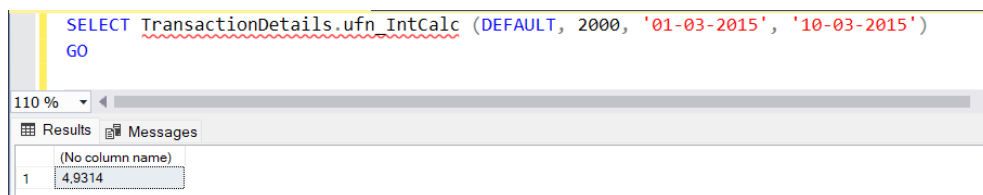


Рисунок 11 – Тестирование функции

5.4. Создание функции, возвращающей табличное значение.

В ходе выполнения данной задачи необходимо ознакомиться со способом создания функции, возвращающей табличное значение.

5.4.1. Код запроса, создающего функцию, принимающую в качестве входного параметра CustomerId и возвращающую таблицу строк TransactionDetails.Transactions, представлен на рисунке 12.

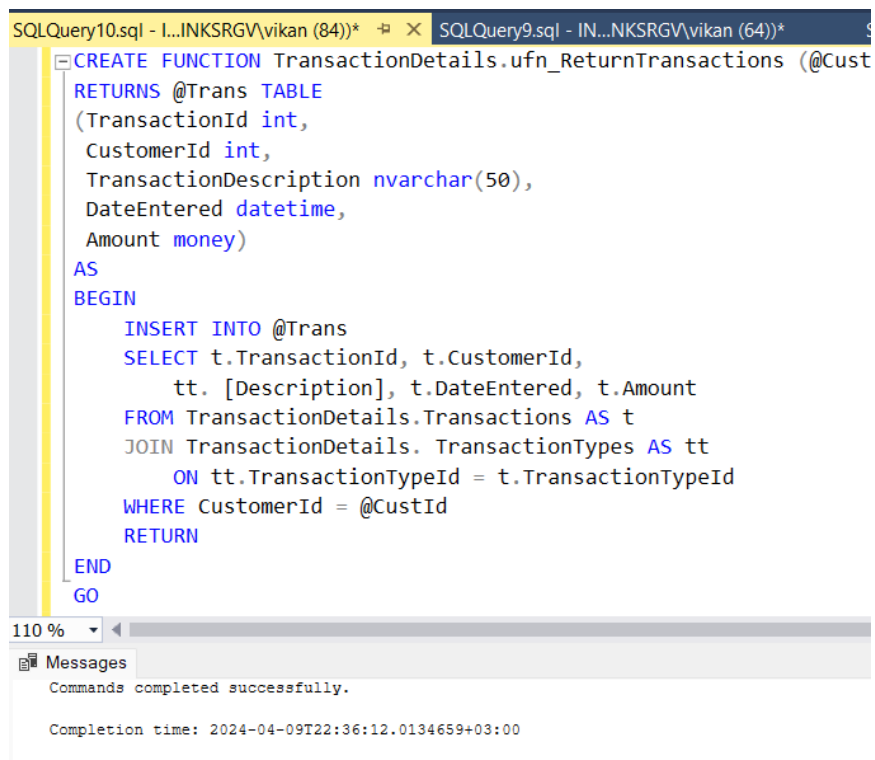


Рисунок 12 – Код запроса для создания функции, возвращающей табличное значение

5.4.2. Код запросов, которые тестируют функцию на двух наборах значений, представлен на рисунке 13 для первого набора и на рисунке 14 для второго. Инструкция SELECT в коде запросов идентичная, однако они имеют различие в операторах: в первом запросе представлен оператор CROSS APPLY, во втором - OUTER APPLY. Оператор CROSS APPLY «применяет»

функцию `ufn_ReturnTransactions` к каждой строке таблицы `CustomerDetails.Customers`. Таким образом данный оператор выводит все данные о клиентах и соответствующих им транзакциях, если они у клиента есть. В противном случае строка с данными о клиенте не будет выведена. Оператор `OUTER APPLY` выводит все значения таблицы `CustomerDetails.Customers`, даже если для некоторых клиентов нет транзакций. В случае, когда у клиента есть транзакции, строка будет отображена полностью и будет содержать данные о клиенте и транзакциях. В противном случае будут выведены только значения о клиенте, а столбцы, соответствующие данным о транзакциях, будут заполнены значениями `NULL`.

SQLQuery11.sql - I...\INKSRGV\vikan (56))* SQLQuery10.sql - I...\INKSRGV\vikan (84))* SQLQuery9

```

SELECT c.FirstName, c.LastName, t.TransactionId,
t.TransactionDescription, t.DateEntered, t.Amount
FROM CustomerDetails.Customers AS c
CROSS APPLY TransactionDetails.ufn_ReturnTransactions(c.CustomerId) AS
GO

```

110 %

Results Messages

	FirstName	LastName	TransactionId	TransactionDescription	DateEntered	Amount
1	Noel	Morgala	1001	Buy	2015-08-01 00:00:00.000	100,00
2	Noel	Morgala	1002	Buy	2015-08-03 00:00:00.000	75,67
3	Noel	Morgala	1003	Sell	2015-08-05 00:00:00.000	35,20
4	Noel	Morgala	1004	Sell	2015-08-06 00:00:00.000	20,00
5	Noel	Morgala	509	Buy	2011-12-05 13:54:54.880	62521,5101
6	Noel	Morgala	836	Cash Deposit	2012-03-10 16:38:08.960	48033,0788
7	Aubrey	Lomas	244	Buy	2012-03-22 22:04:05.440	14149,7153
8	Aubrey	Lomas	607	Sell	2012-03-06 14:20:18.770	34215,274
9	Bernie	McGee	926	Sell	2011-11-10 02:01:26.640	75316,5828
10	Jane	Harper	6	Sell	2012-01-21 16:14:49.450	41124,6195
11	Terence	Madden	121	Cash Withdrawal	2012-01-29 09:22:33.250	52477,0824
12	Terence	Madden	511	Buy	2012-01-11 22:57:02.670	1037,0005
13	Anne	Mather	529	Cash Withdrawal	2012-03-23 05:47:13.830	98415,5152
14	Mickey	Ferguson	786	Sell	2012-03-19 09:42:09.840	73237,3412
15	Mickey	Ferguson	964	Buy	2011-11-09 18:41:01.320	5132,1456

Рисунок 13 – Код запроса для тестирования функции

SQLQuery12.sql - I...\INKSRGV\vikan (74))* SQLQuery11.sql - I...\INKSRGV\vikan (56))* SQLQuery10.sql

```

SELECT c.FirstName, c.LastName, t.TransactionId,
t.TransactionDescription, t.DateEntered, t.Amount
FROM CustomerDetails.Customers AS c
OUTER APPLY TransactionDetails.ufn_ReturnTransactions (c.CustomerId) AS
GO

```

110 %

Results Messages

	FirstName	LastName	TransactionId	TransactionDescription	DateEntered	Amount
1	Noel	Morgala	1001	Buy	2015-08-01 00:00:00.000	100,00
2	Noel	Morgala	1002	Buy	2015-08-03 00:00:00.000	75,67
3	Noel	Morgala	1003	Sell	2015-08-05 00:00:00.000	35,20
4	Noel	Morgala	1004	Sell	2015-08-06 00:00:00.000	20,00
5	Noel	Morgala	509	Buy	2011-12-05 13:54:54.880	62521,5101
6	Noel	Morgala	836	Cash Deposit	2012-03-10 16:38:08.960	48033,0788
7	Aubrey	Lomas	244	Buy	2012-03-22 22:04:05.440	14149,7153
8	Aubrey	Lomas	607	Sell	2012-03-06 14:20:18.770	34215,274
9	Bernie	McGee	926	Sell	2011-11-10 02:01:26.640	75316,5828
10	Jane	Harper	6	Sell	2012-01-21 16:14:49.450	41124,6195
11	Mark	Vernon-Smith	NULL	NULL	NULL	NULL

Рисунок 14 – Код запроса для тестирования функции

6. Выводы и анализ результатов работы

В ходе выполнения данной лабораторной работы были получены практические и теоретические навыки по реализации пользовательских функций и процедур, а также теоретические знания по применению операторов CROSS APPLY и OUTER APPLY в инструменте SQL Server Management Studio.

Все поставленные задачи были выполнены с помощью подробной инструкции, представленной в тексте лабораторной работы, поэтому в ходе выполнения работы никаких трудностей не возникло.