**Лабораторная работа № 11 Тема: Хранимые процедуры. Триггеры (типа AFTER)**

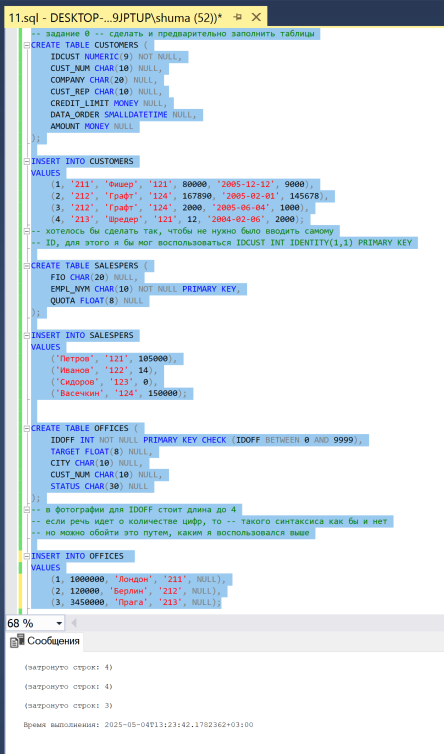
**⇒ Подготовительное задание:**

Средствами SQL Server Management Studio(SSMS) создайте новую БД со следующими нижеуказанными таблицами (связи не устанавливать)

Название БД должно содержать Группу и Фамилию + ЛР\_11 например :( 01\_Ivanov\_LR\_11 )



Итак, требуется создать следующие таблицы



-- задание 0 -- сделать и предварительно заполнить таблицы

CREATE TABLE CUSTOMERS (

IDCUST NUMERIC(9) NOT NULL,

CUST\_NUM CHAR(10) NULL,

COMPANY CHAR(20) NULL,

CUST\_REP CHAR(10) NULL,

CREDIT\_LIMIT MONEY NULL,

DATA\_ORDER SMALLDATETIME NULL,

AMOUNT MONEY NULL

);

INSERT INTO CUSTOMERS

VALUES

(1, '211', 'Фишер', '121', 80000, '2005-12-12', 9000),

(2, '212', 'Графт', '124', 167890, '2005-02-01', 145678),

(3, '212', 'Графт', '124', 2000, '2005-06-04', 1000),

(4, '213', 'Шредер', '121', 12, '2004-02-06', 2000);

-- хотелось бы сделать так, чтобы не нужно было вводить самому

-- ID, для этого я бы мог воспользоваться IDCUST INT IDENTITY(1,1) PRIMARY KEY

CREATE TABLE SALESPERS (

FIO CHAR(20) NULL,

EMPL\_NYM CHAR(10) NOT NULL PRIMARY KEY,

QUOTA FLOAT(8) NULL

);

INSERT INTO SALESPERS

VALUES

('Петров', '121', 105000),

('Иванов', '122', 14),

('Сидоров', '123', 0),

('Васечкин', '124', 150000);

CREATE TABLE OFFICES (

IDOFF INT NOT NULL PRIMARY KEY CHECK (IDOFF BETWEEN 0 AND 9999),

TARGET FLOAT(8) NULL,

CITY CHAR(10) NULL,

CUST\_NUM CHAR(10) NULL,

STATUS CHAR(30) NULL

);

-- в фотографии для IDOFF стоит длина до 4

-- если речь идет о количестве цифр, то -- такого синтаксиса как бы и нет

-- но можно обойти это путем, каким я воспользовался выше

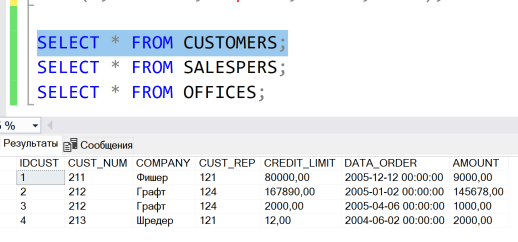
INSERT INTO OFFICES

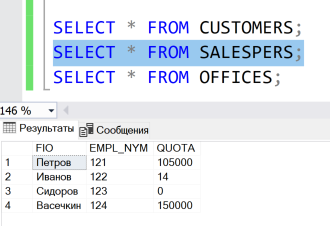
VALUES

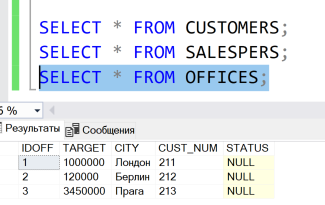
(1, 1000000, 'Лондон', '211', NULL),

(2, 120000, 'Берлин', '212', NULL),

(3, 3450000, 'Прага', '213', NULL);



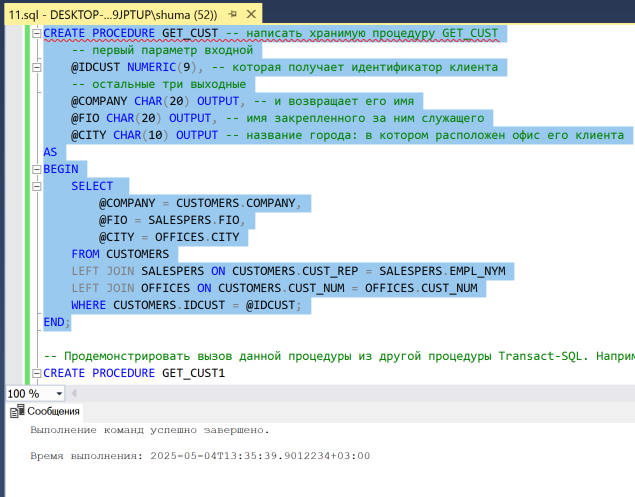




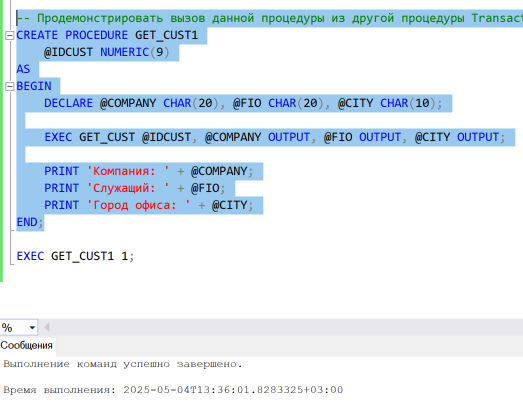
**⇒ Основные задания**

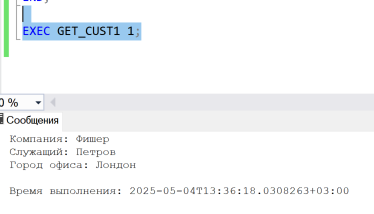
1. Написать хранимую процедуру GET\_CUST, которая получает идентификатор клиента и возвращает его имя (поле COMPANY таблицы CUSTOMERS), имя закрепленного за ним служащего (поле FIO таблицы SALESPERS) и название города, в котором расположен офис этого клиента (поле CITY таблицы OFFICES).

Первый из передаваемых параметров - входной, остальные три – выходные (используются для передачи запрошенных данных вызывающей процедуре).



Продемонстрировать вызов данной процедуры из другой процедуры Transact-SQL. Например, GET\_CUST1 (идентификатор\_клиента).





1. Написать хранимую процедуру CHK\_TOT, которая получает идентификатор клиента и вычисляет общую стоимость его заказов (поле AMOUNT таблицы CUSTOMERS) и в зависимости от того, превысит ли эта сумма 30 000$, заносит в поле STATUS таблицы OFFICES одно из двух примечаний – “большой объем заказов”, ”малый объем заказов”



-- Задание 2

-- нужно учесть то, что у нас есть одинаковые CUST\_NUM

-- Написать хранимую процедуру CHK\_TOT

CREATE OR ALTER PROCEDURE CHK\_TOT

@IDCUST NUMERIC(9) -- которая получает идентификатор клиента

AS

BEGIN

-- объявляем переменные

DECLARE @TotalAmount MONEY = 0;

DECLARE @CUST\_NUM CHAR(10);

DECLARE @COMPANY CHAR(20);

DECLARE @StatusMessage NVARCHAR(100);

DECLARE @OfficeCount INT;

DECLARE @CurrentStatus CHAR(50);

-- Получаем данные клиента

SELECT @CUST\_NUM = CUST\_NUM,

@COMPANY = COMPANY

FROM CUSTOMERS

WHERE IDCUST = @IDCUST;

-- учтем, а есть ли клиент

IF @CUST\_NUM IS NOT NULL

BEGIN

-- и вычисляет общую стоимость его заказов

SELECT @TotalAmount = COALESCE(SUM(AMOUNT), 0)

FROM CUSTOMERS

WHERE CUST\_NUM = @CUST\_NUM;

-- и в зависимости от того, превысит ли эта сумма 30 000$,

-- заносит в поле STATUS таблицы OFFICES

-- одно из двух примечаний – “большой объем заказов”, ”малый объем заказов”

IF @TotalAmount > 30000

SET @CurrentStatus = 'большой объем заказов';

ELSE

SET @CurrentStatus = 'малый объем заказов';

UPDATE OFFICES

SET STATUS = @CurrentStatus

WHERE CUST\_NUM = @CUST\_NUM;

-- Получаем количество обновленных офисов

SELECT @OfficeCount = @@ROWCOUNT;

-- вывод (для отладки)

PRINT '===================================';

PRINT 'Обработка клиента:';

PRINT 'ID клиента: ' + CAST(@IDCUST AS NVARCHAR);

PRINT 'Номер клиента: ' + @CUST\_NUM;

PRINT 'Компания: ' + @COMPANY;

PRINT 'Общая сумма заказов: ' + CAST(@TotalAmount AS NVARCHAR) + '$';

PRINT 'Установлен статус: ' + @CurrentStatus;

PRINT 'Обновлено офисов: ' + CAST(@OfficeCount AS NVARCHAR);

PRINT '===================================';

END

ELSE -- если клиент не найден

BEGIN

PRINT '===================================';

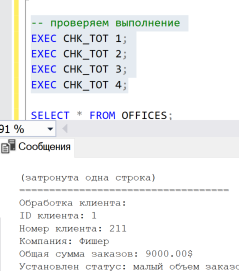
PRINT 'Ошибка: Клиент с ID ' + CAST(@IDCUST AS NVARCHAR) + ' не найден';

PRINT '===================================';

END

END;

Проверяем выполнение



(затронута одна строка)

===================================

Обработка клиента:

ID клиента: 1

Номер клиента: 211

Компания: Фишер

Общая сумма заказов: 9000.00$

Установлен статус: малый объем заказов

Обновлено офисов: 1

===================================

(затронута одна строка)

===================================

Обработка клиента:

ID клиента: 2

Номер клиента: 212

Компания: Графт

Общая сумма заказов: 146678.00$

Установлен статус: большой объем заказов

Обновлено офисов: 1

===================================

(затронута одна строка)

===================================

Обработка клиента:

ID клиента: 3

Номер клиента: 212

Компания: Графт

Общая сумма заказов: 146678.00$

Установлен статус: большой объем заказов

Обновлено офисов: 1

===================================

(затронута одна строка)

===================================

Обработка клиента:

ID клиента: 4

Номер клиента: 213

Компания: Шредер

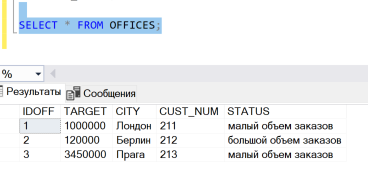
Общая сумма заказов: 2000.00$

Установлен статус: малый объем заказов

Обновлено офисов: 1

===================================

Время выполнения: 2025-05-04T14:13:57.7810176+03:00



1. Написать хранимую процедуру для добавления данных о новом клиенте в таблицу OFFICES.

• Добавить новую строку в таблицу OFFICES. Разрешается добовление только для клиентов, уже имеющих запись в таблице CUSTOMERS.

• Обновить запись в SALESPERS, увеличив поле QUOTA для соответствующего служащего (для каждого, с которым работает клиент) на величину объема продаж добавленной в предыдущем пункте записи. Плановый объем продаж служащего не может быть увеличен более чем на определенную величину.

Если сумма заказов(@TotalAmount) рассматриваемого клиента составляет < 20 000$ (вызов процедуры CHK\_TOT, которая должна возвращать эту сумму, например, в качестве выходного параметра), то величина объема продаж (поле TARGET таблицы OFFICES) будет добавлена к плану служащего.

Если сумма заказов рассматриваемого клиента == 20 000$ к плану будут добавлены фиксированные 20 000$.

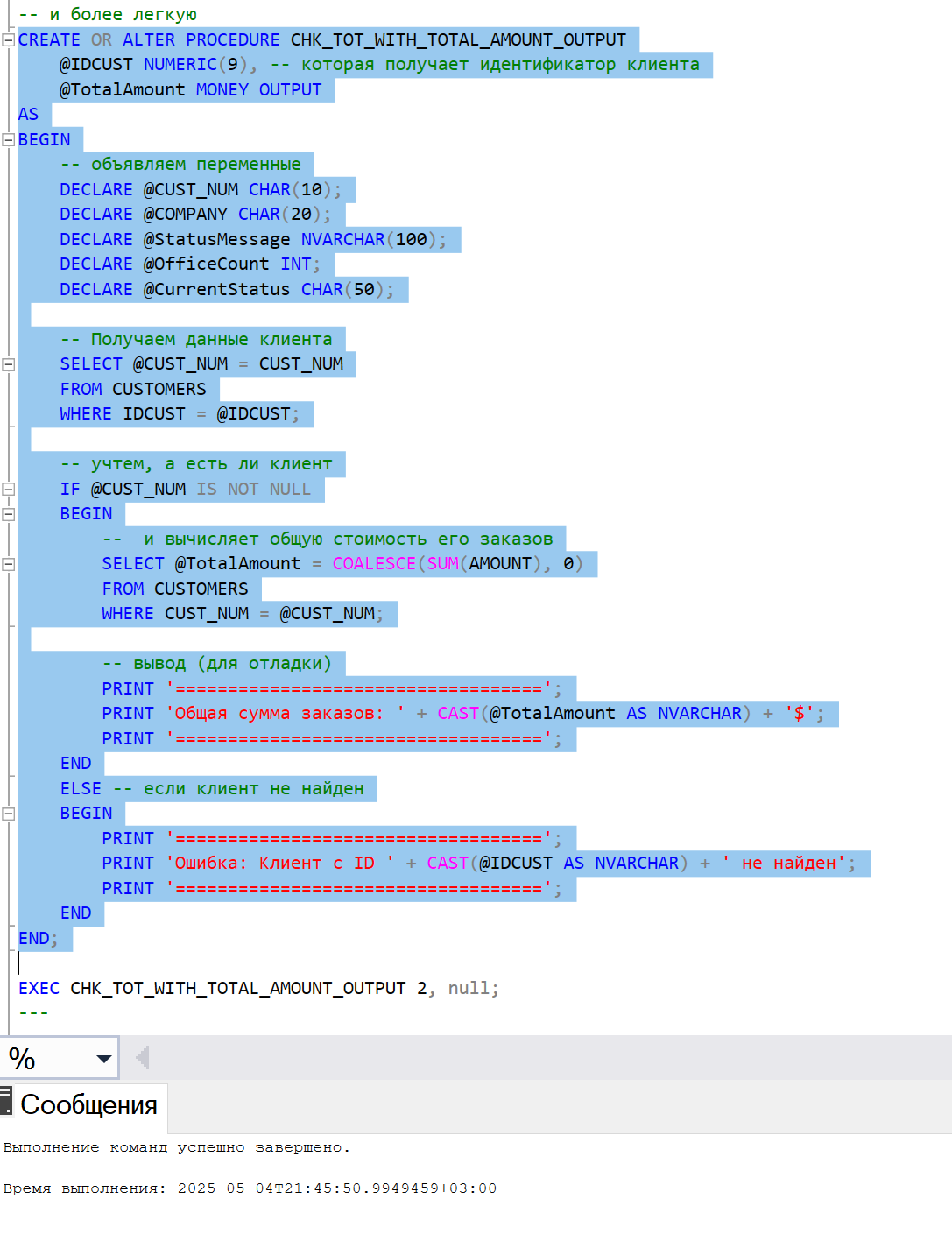
В противном случае – запретить добавление новой записи в OFFICES и обновление в SALESPERS. Предусмотреть промежуточный вывод суммы заказов рассматриваемого клиента.

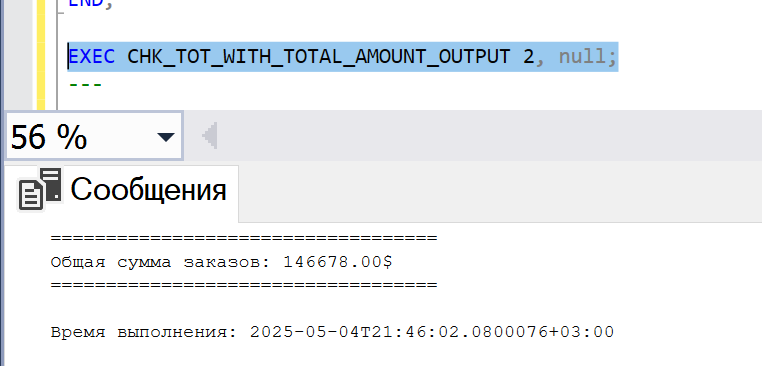
-- по заданию нужно будет воспользоваться функцией из задания 2

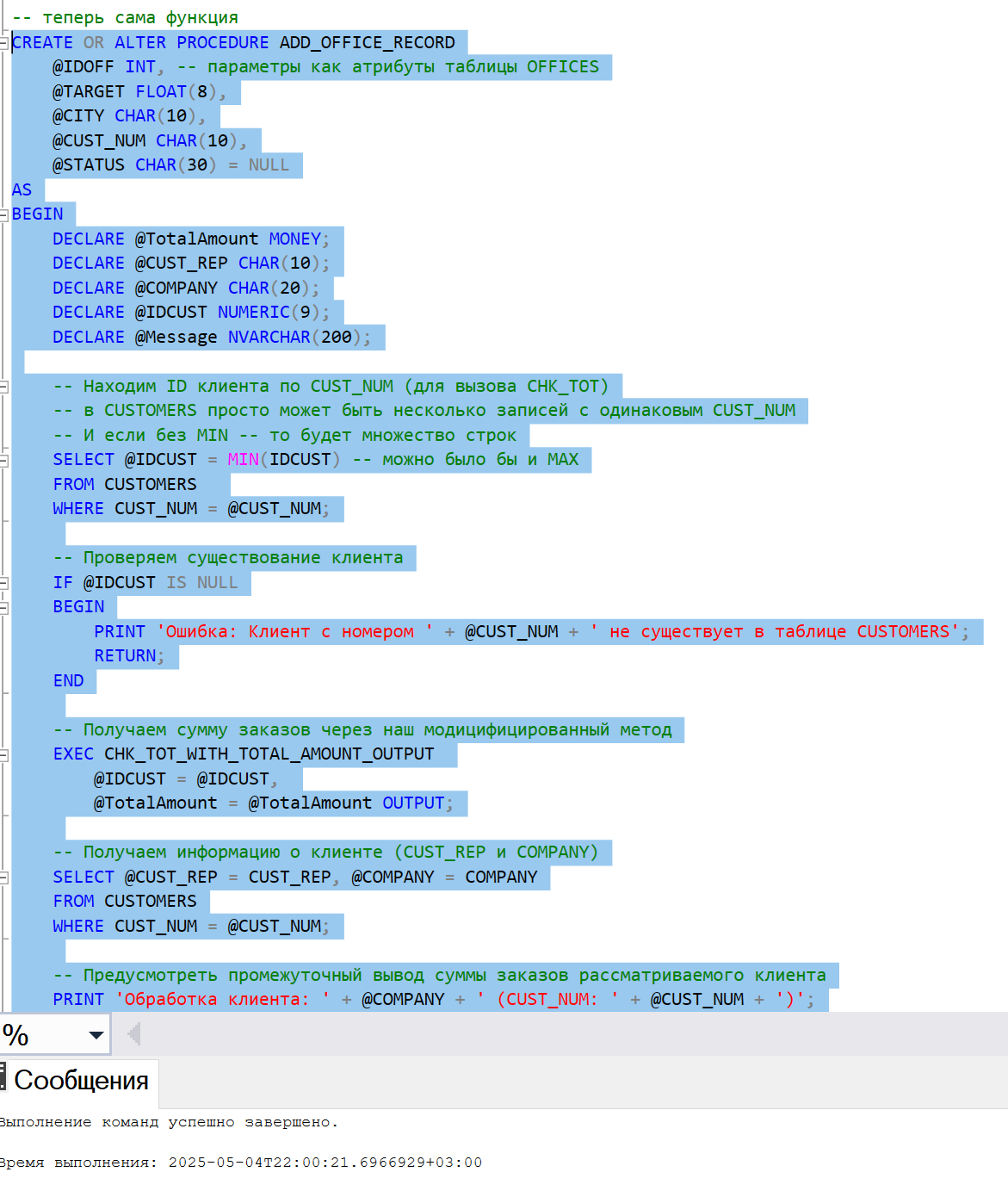
-- и я бы ей воспользвался бы для подсчета @TotalAmount, но -- у нее нет OUTPUT параметра

-- поэтому я решил сделать новую процедуру, что имеет в себе этот параметр

-- и более легкую







CREATE OR ALTER PROCEDURE ADD\_OFFICE\_RECORD

@IDOFF INT, -- параметры как атрибуты таблицы OFFICES

@TARGET FLOAT(8),

@CITY CHAR(10),

@CUST\_NUM CHAR(10),

@STATUS CHAR(30) = NULL

AS

BEGIN

DECLARE @TotalAmount MONEY;

DECLARE @CUST\_REP CHAR(10);

DECLARE @COMPANY CHAR(20);

DECLARE @IDCUST NUMERIC(9);

DECLARE @Message NVARCHAR(200);

-- Находим ID клиента по CUST\_NUM (для вызова CHK\_TOT)

-- в CUSTOMERS просто может быть несколько записей с одинаковым CUST\_NUM

-- И если без MIN -- то будет множество строк

SELECT @IDCUST = MIN(IDCUST) -- можно было бы и MAX

FROM CUSTOMERS

WHERE CUST\_NUM = @CUST\_NUM;

-- Проверяем существование клиента

IF @IDCUST IS NULL

BEGIN

PRINT 'Ошибка: Клиент с номером ' + @CUST\_NUM + ' не существует в таблице CUSTOMERS';

RETURN;

END

-- Получаем сумму заказов через наш модицифицированный метод

EXEC CHK\_TOT\_WITH\_TOTAL\_AMOUNT\_OUTPUT

@IDCUST = @IDCUST,

@TotalAmount = @TotalAmount OUTPUT;

-- Получаем информацию о клиенте (CUST\_REP и COMPANY)

SELECT @CUST\_REP = CUST\_REP, @COMPANY = COMPANY

FROM CUSTOMERS

WHERE CUST\_NUM = @CUST\_NUM;

-- Предусмотреть промежуточный вывод суммы заказов рассматриваемого клиента

PRINT 'Обработка клиента: ' + @COMPANY + ' (CUST\_NUM: ' + @CUST\_NUM + ')';

PRINT 'Сумма заказов: ' + CAST(@TotalAmount AS NVARCHAR) + '$';

-- Проверяем условия для добавления записи

IF @TotalAmount < 20000

BEGIN

-- Добавляем запись в OFFICES

INSERT INTO OFFICES (IDOFF, TARGET, CITY, CUST\_NUM, STATUS)

VALUES (@IDOFF, @TARGET, @CITY, @CUST\_NUM, @STATUS);

-- Обновляем QUOTA в SALESPERS (добавляем TARGET)

UPDATE SALESPERS

SET QUOTA = QUOTA + @TARGET

WHERE EMPL\_NYM = @CUST\_REP;

PRINT 'Добавлена запись в OFFICES. QUOTA служащего увеличен на ' + CAST(@TARGET AS NVARCHAR) + '$';

END

ELSE IF @TotalAmount = 20000

BEGIN

-- Добавляем запись в OFFICES

INSERT INTO OFFICES (IDOFF, TARGET, CITY, CUST\_NUM, STATUS)

VALUES (@IDOFF, @TARGET, @CITY, @CUST\_NUM, @STATUS);

-- Обновляем QUOTA

UPDATE SALESPERS

SET QUOTA = QUOTA + 20000 -- (фиксированная сумма 20000)

WHERE EMPL\_NYM = @CUST\_REP;

PRINT 'Добавлена запись в OFFICES. QUOTA служащего увеличен на 20000$';

END

ELSE

BEGIN

PRINT 'Ошибка: Сумма заказов клиента (' + CAST(@TotalAmount AS NVARCHAR) + '$) превышает 20000$. Добавление запрещено.';

RETURN;

END

-- Выводим итоговую информацию о служащем

SELECT @Message = 'Итоговый QUOTA служащего: ' + FIO + ' (' + EMPL\_NYM + ') = ' + CAST(QUOTA AS NVARCHAR) + '$'

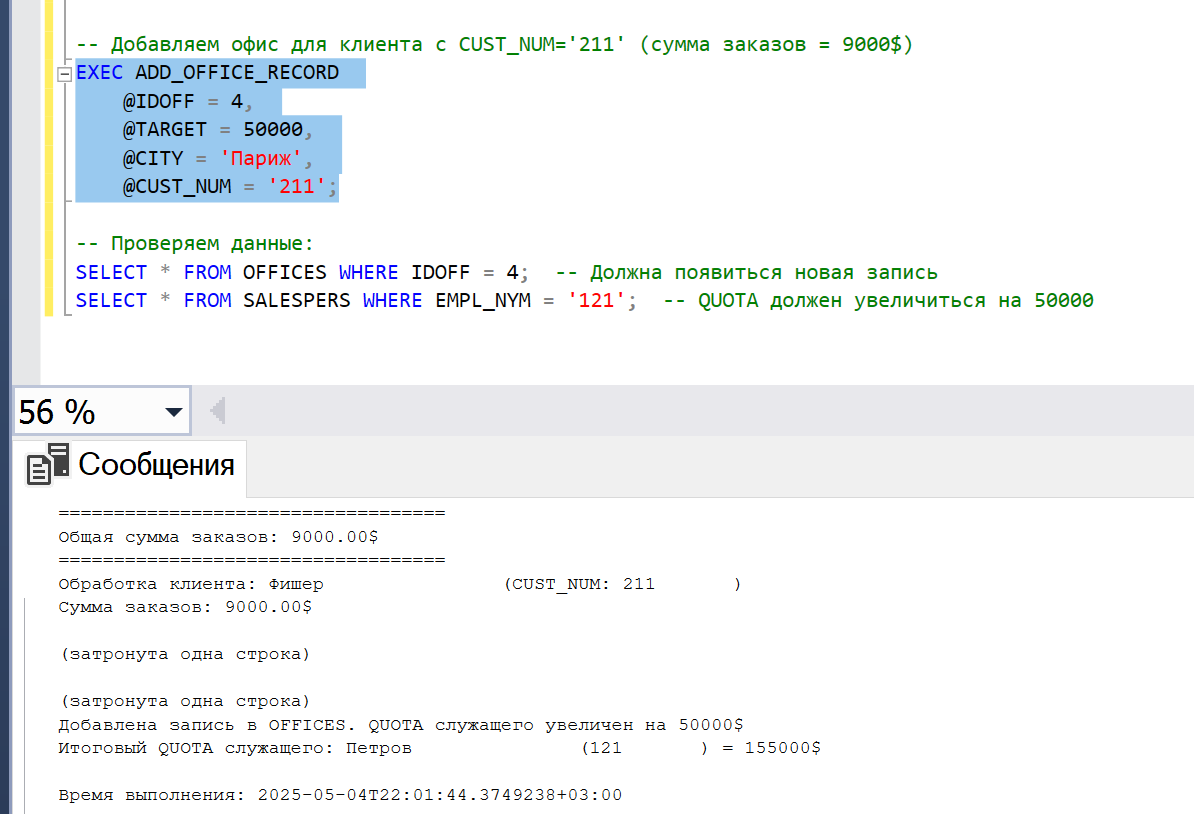
FROM SALESPERS

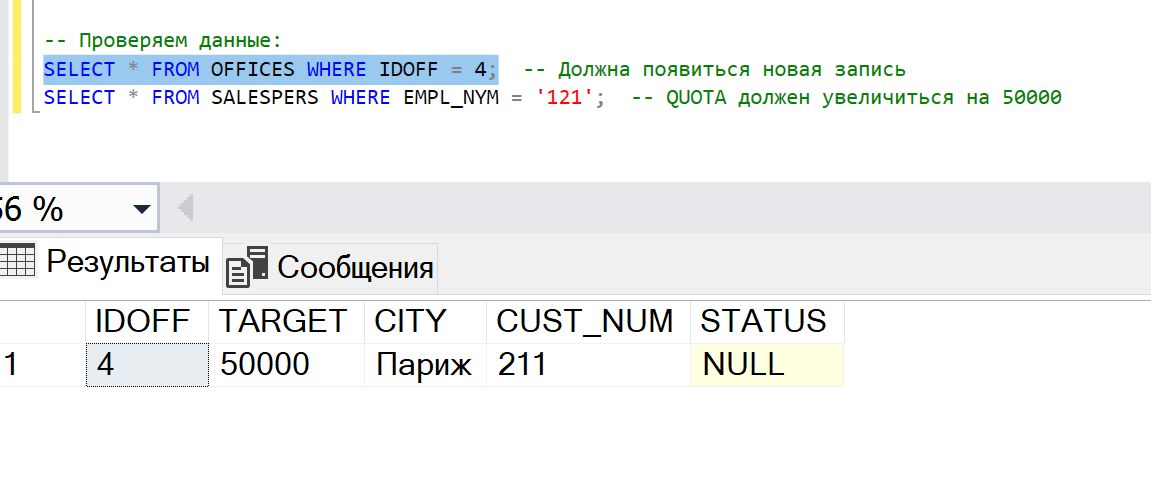
WHERE EMPL\_NYM = @CUST\_REP;

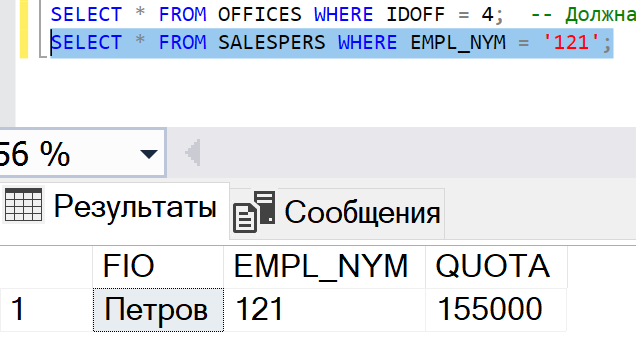
PRINT @Message;

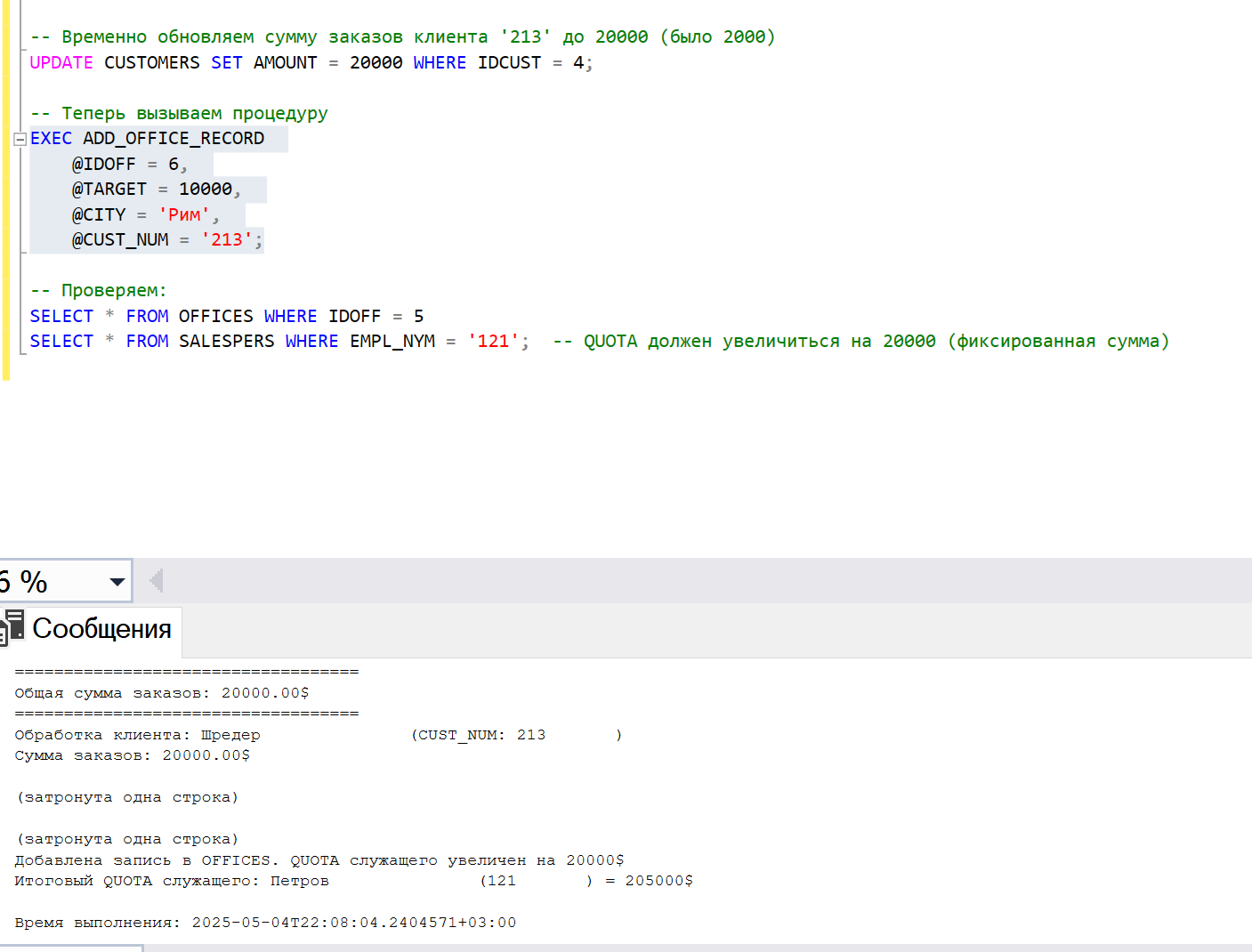
END;

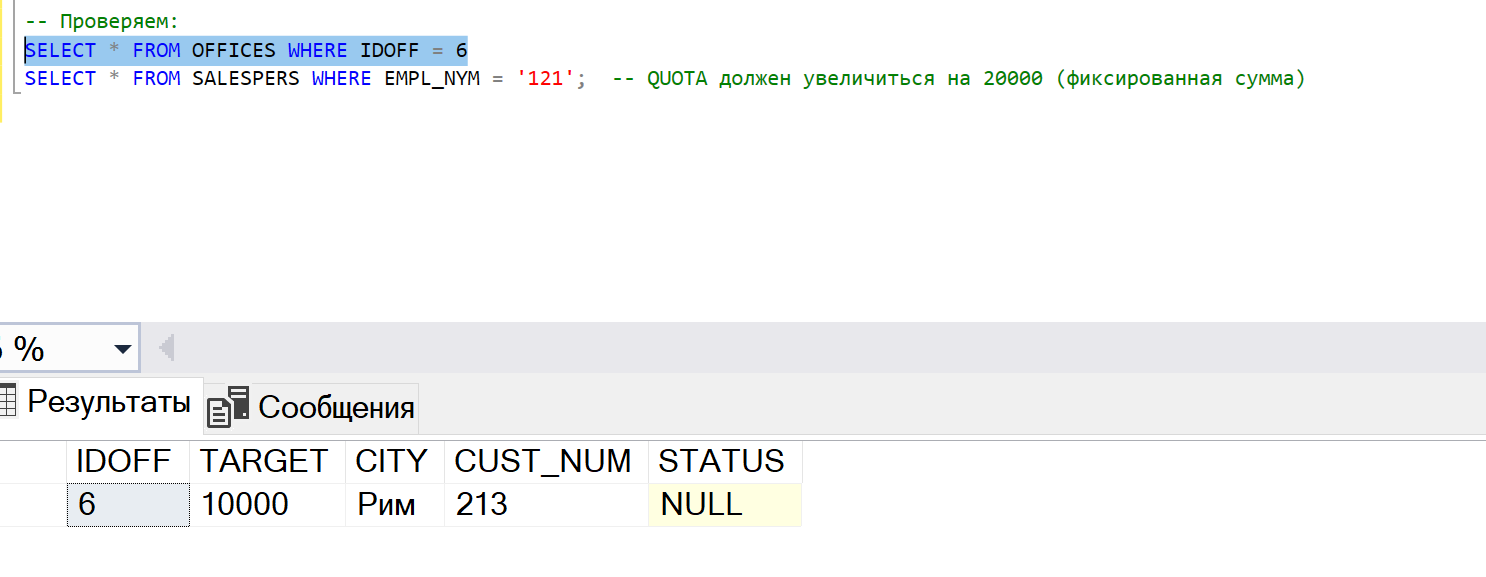
Чтобы проверить, с имитируем несколько случаев

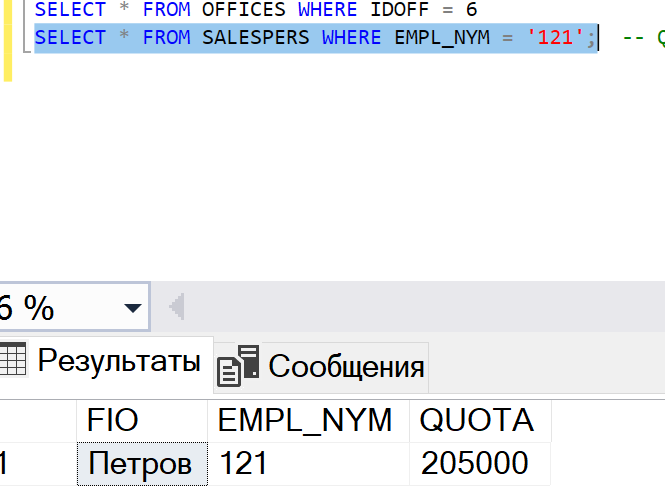




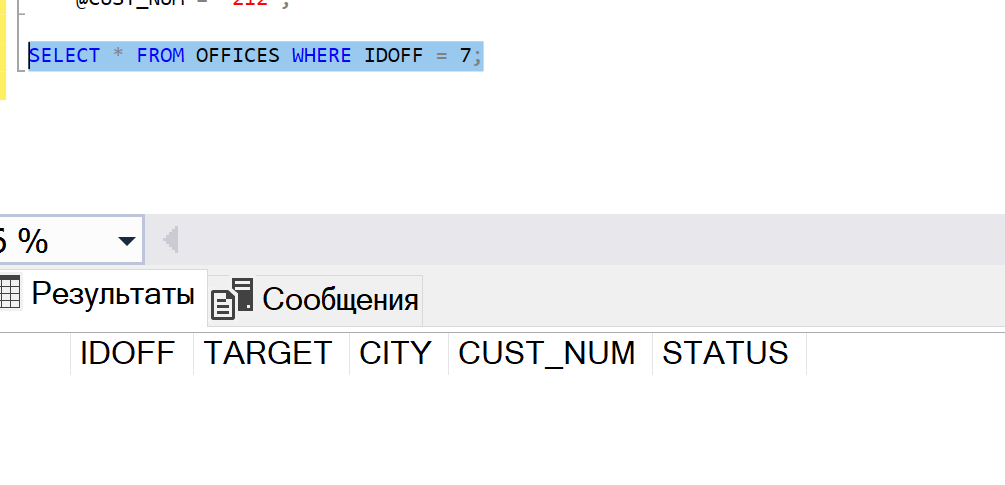




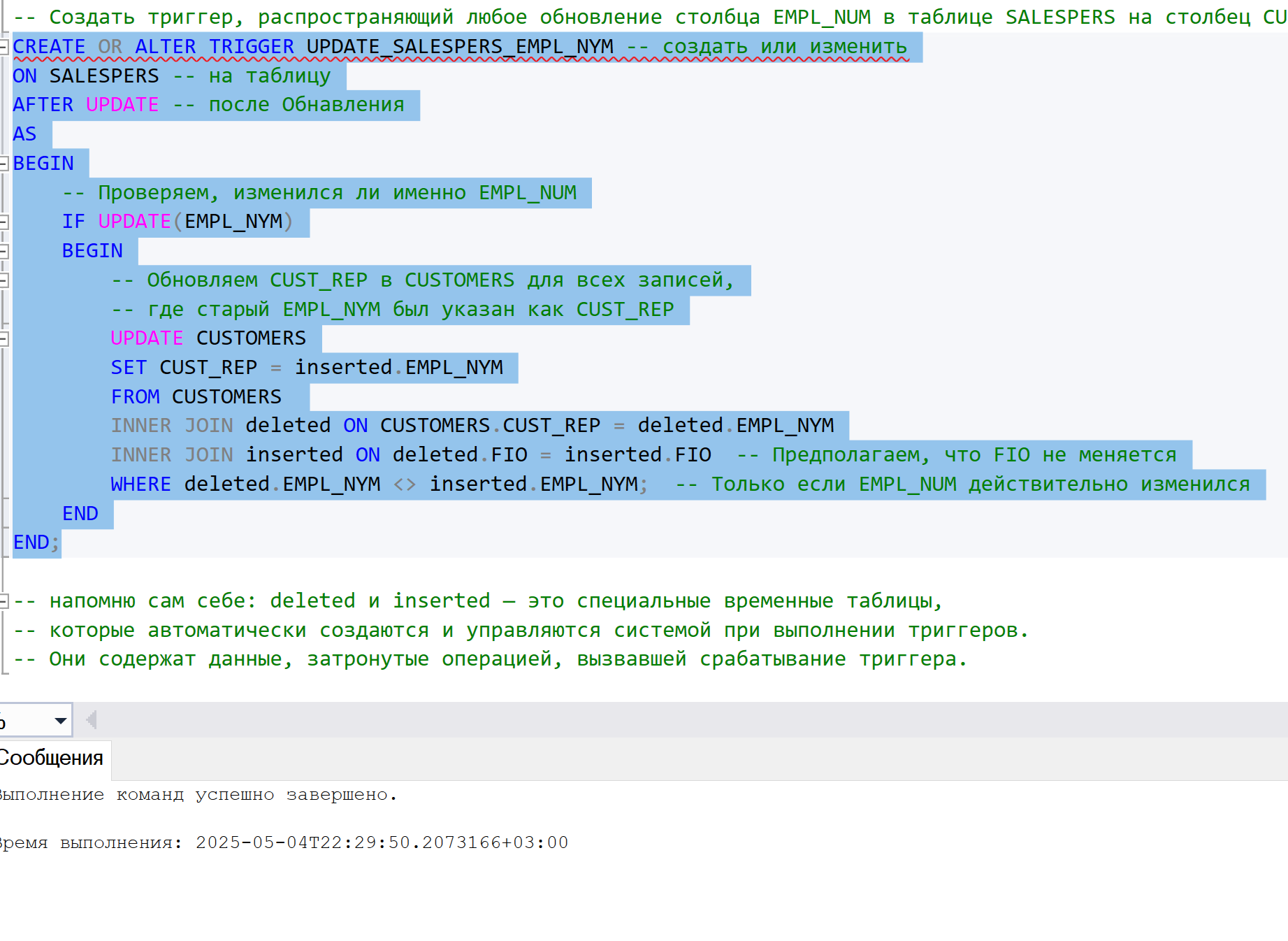








1. Создать триггер, распространяющий любое обновление столбца EMPL\_NUM в таблице SALESPERS на столбец CUST\_REP таблицы CUSTOMERS.



CREATE OR ALTER TRIGGER UPDATE\_SALESPERS\_EMPL\_NYM -- создать или изменить

ON SALESPERS -- на таблицу

AFTER UPDATE -- после Обнавления

AS

BEGIN

-- Проверяем, изменился ли именно EMPL\_NUM

IF UPDATE(EMPL\_NYM)

BEGIN

-- Обновляем CUST\_REP в CUSTOMERS для всех записей,

-- где старый EMPL\_NYM был указан как CUST\_REP

UPDATE CUSTOMERS

SET CUST\_REP = inserted.EMPL\_NYM

FROM CUSTOMERS

INNER JOIN deleted ON CUSTOMERS.CUST\_REP = deleted.EMPL\_NYM

INNER JOIN inserted ON deleted.FIO = inserted.FIO -- Предполагаем, что FIO не меняется

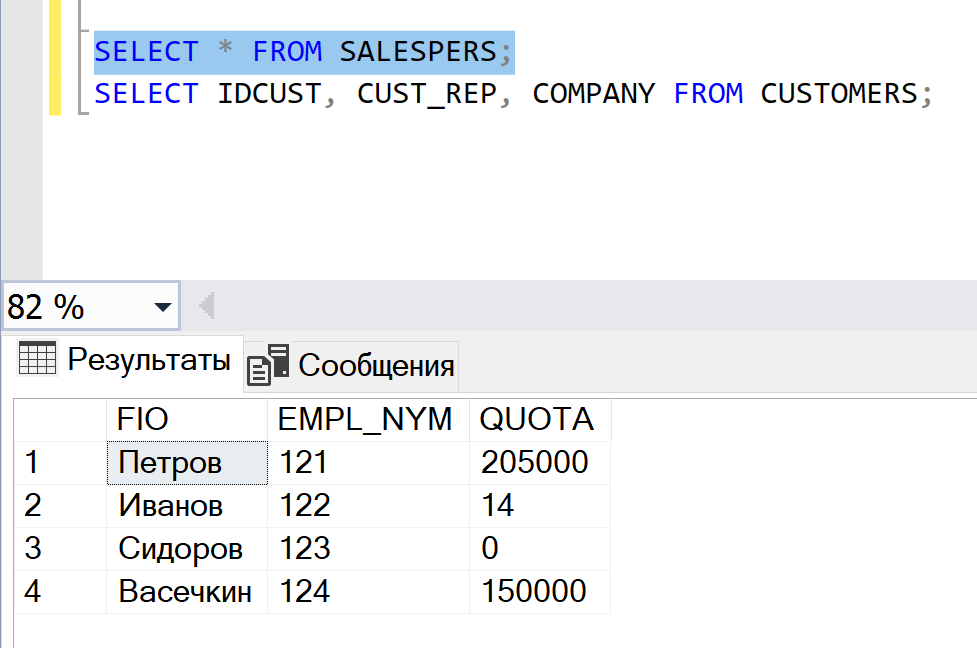
WHERE deleted.EMPL\_NYM <> inserted.EMPL\_NYM; -- Только если EMPL\_NUM действительно изменился

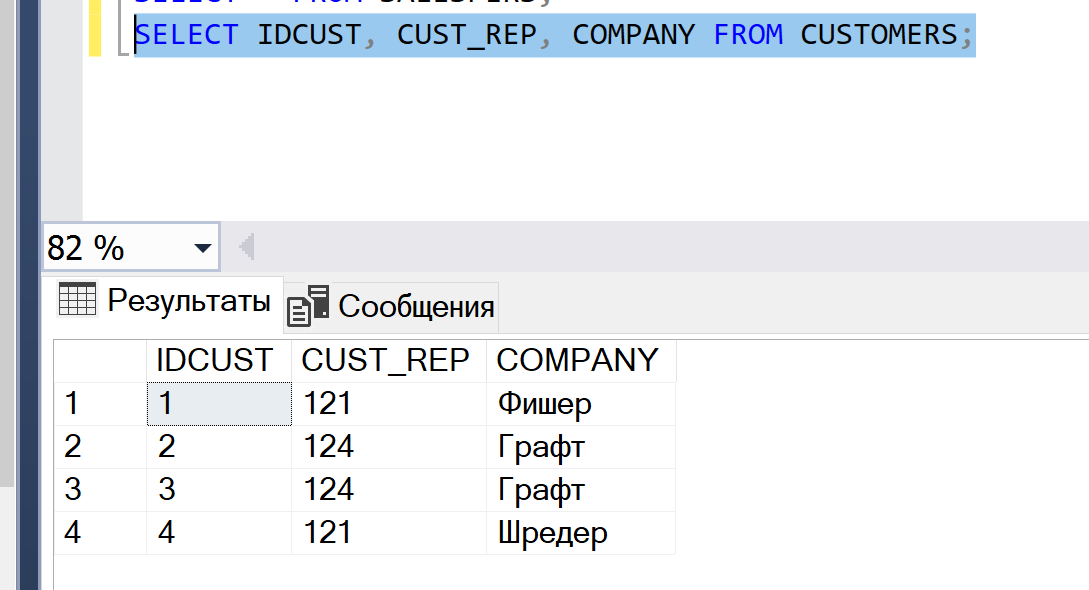
END

END;

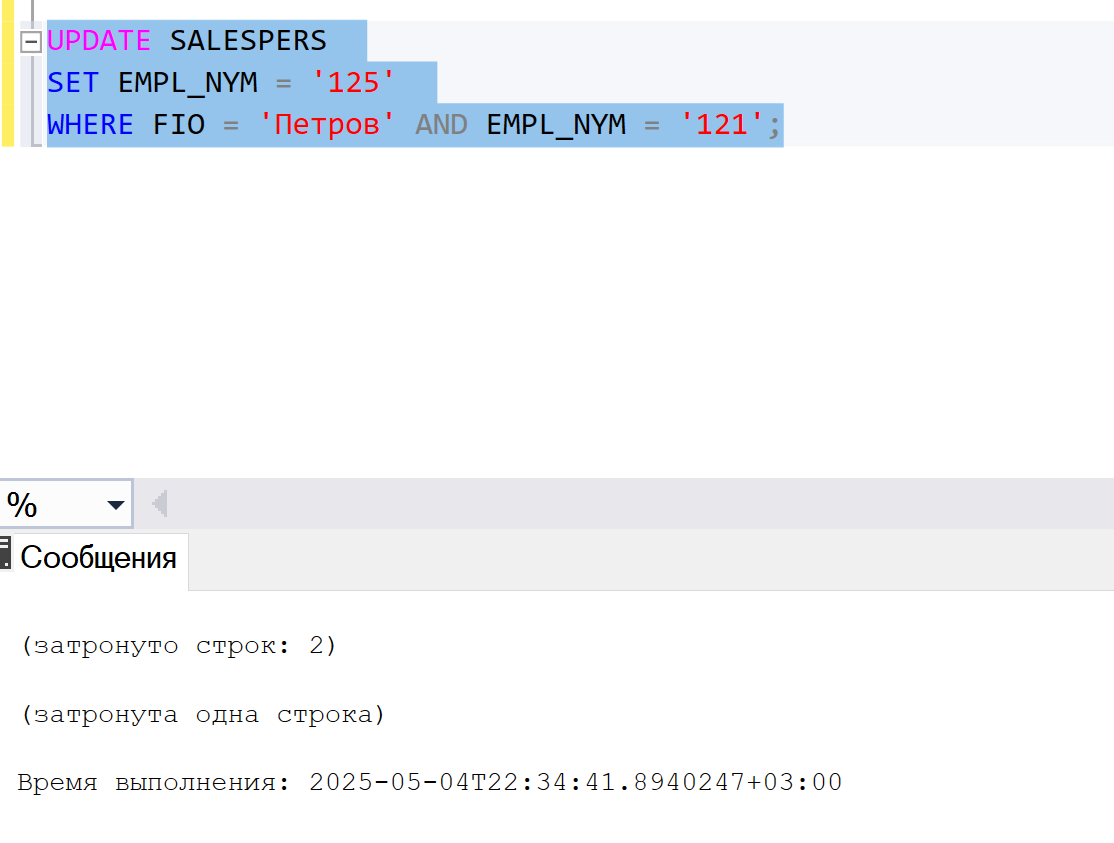
Проверяем

Было

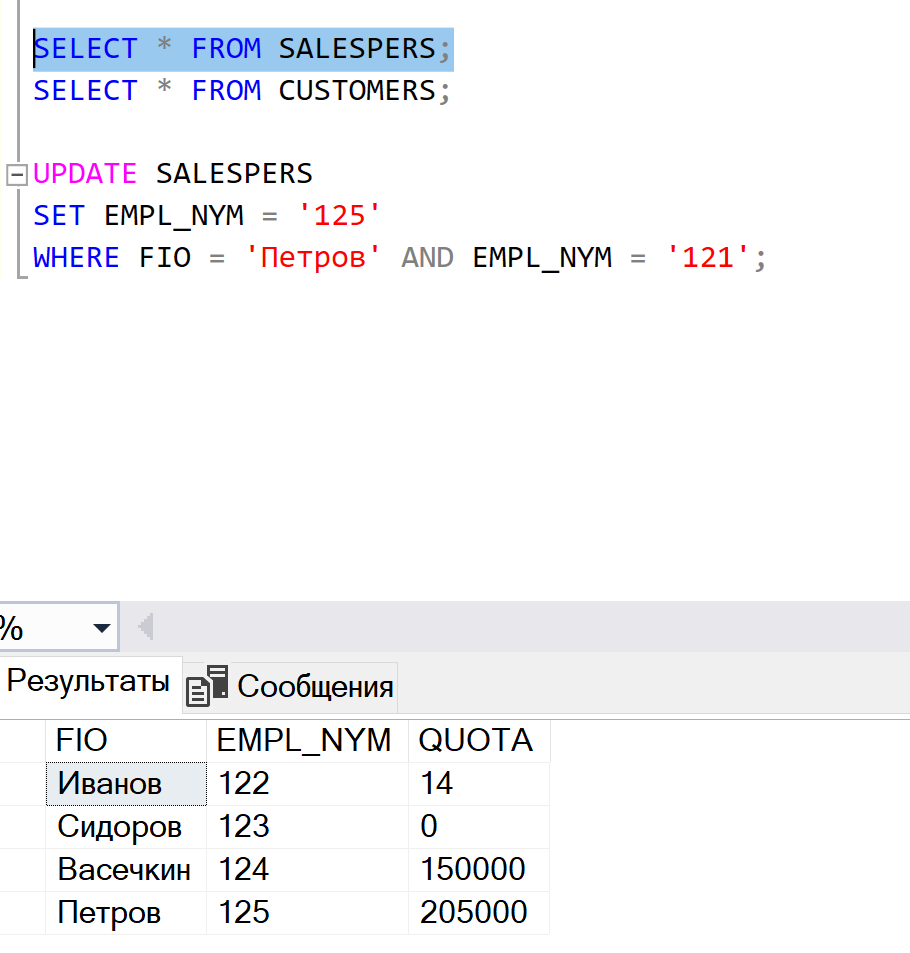
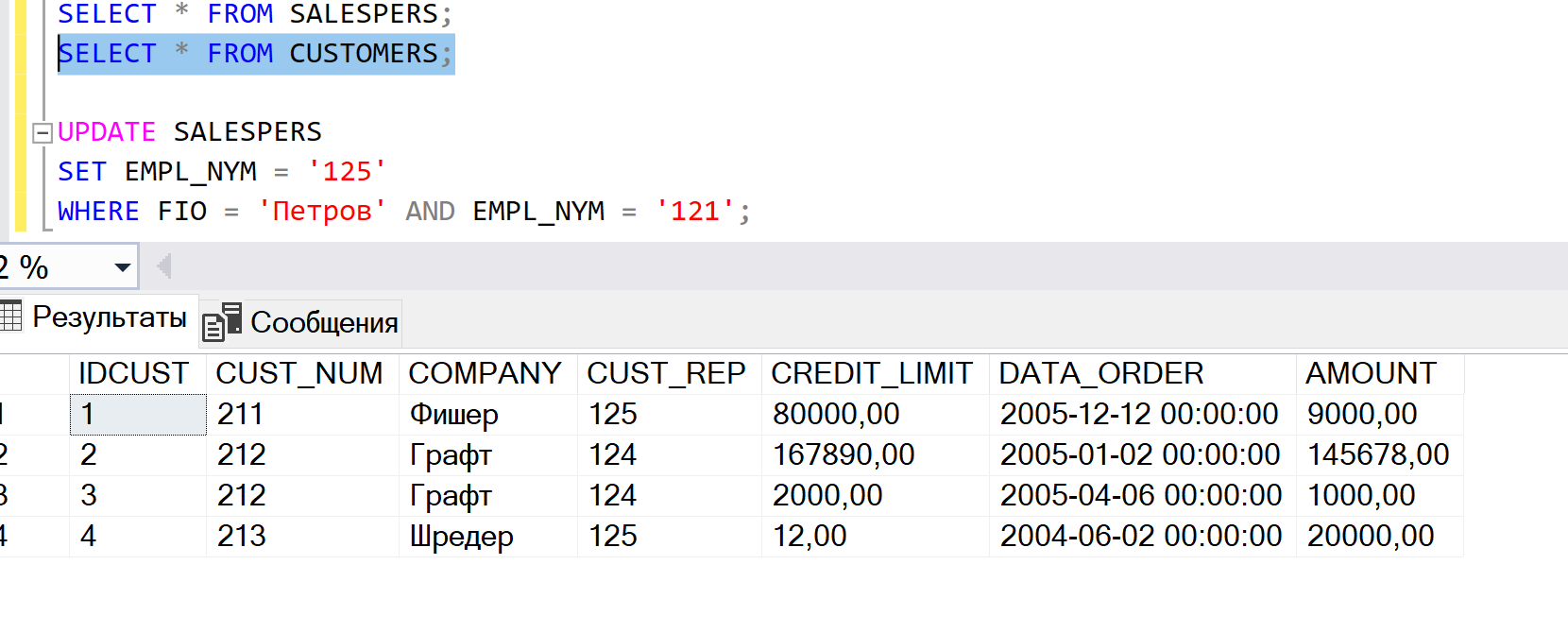




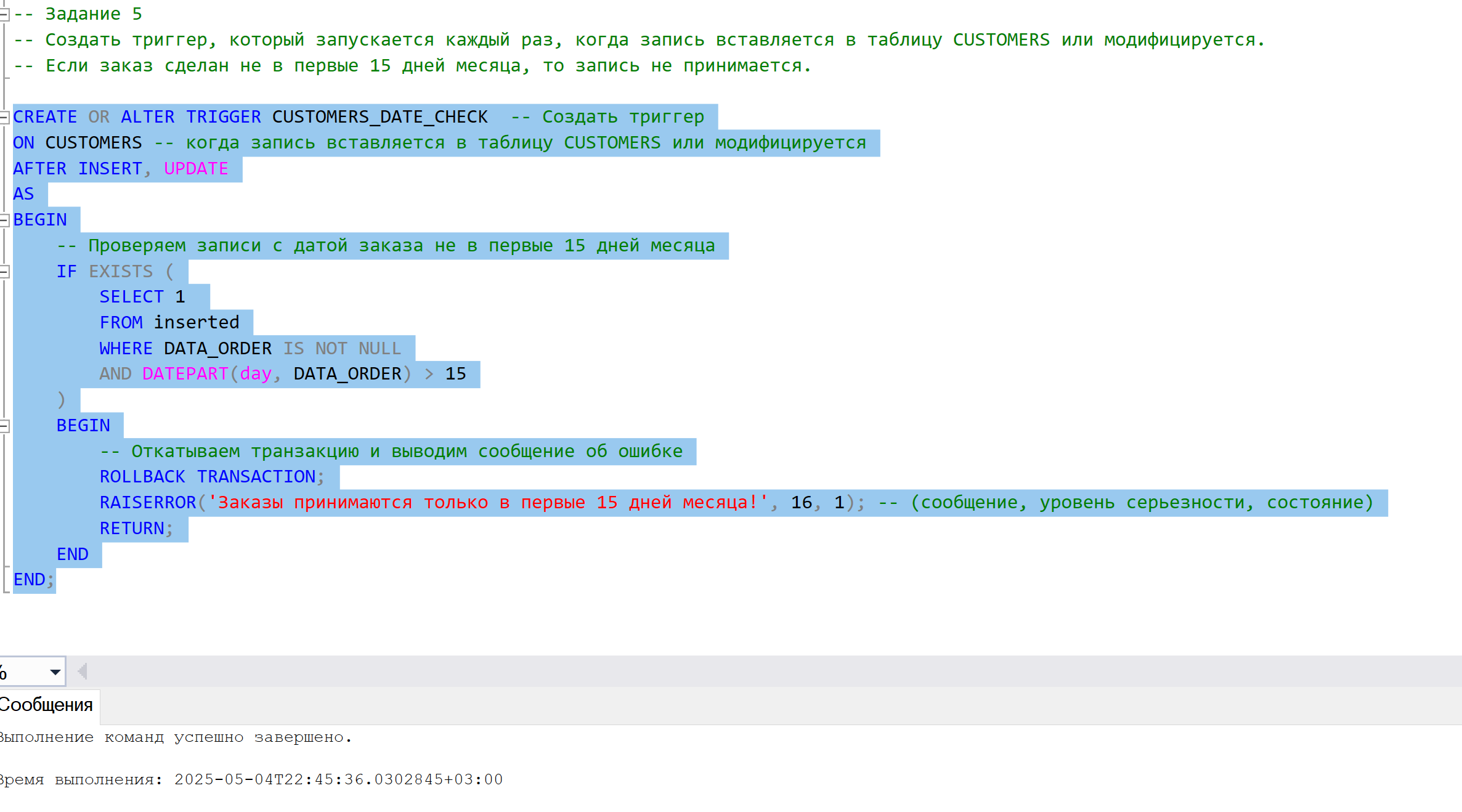
выполняем



Итого имеем

1. Создать триггер, который запускается каждый раз, когда запись вставляется в таблицу CUSTOMERS или модифицируется. Если заказ сделан не в первые 15 дней месяца, то запись не принимается. Для выделения определенной части даты (дня) в виде целого значения используйте функцию DATEPART(date\_part, date), где date\_partday|month|year.



CREATE OR ALTER TRIGGER CUSTOMERS\_DATE\_CHECK -- Создать триггер

ON CUSTOMERS -- когда запись вставляется в таблицу CUSTOMERS или модифицируется

AFTER INSERT, UPDATE

AS

BEGIN

-- Проверяем записи с датой заказа не в первые 15 дней месяца

IF EXISTS (

SELECT 1

FROM inserted

WHERE DATA\_ORDER IS NOT NULL

AND DATEPART(day, DATA\_ORDER) > 15

)

BEGIN

-- Откатываем транзакцию и выводим сообщение об ошибке

ROLLBACK TRANSACTION;

RAISERROR('Заказы принимаются только в первые 15 дней месяца!', 16, 1); -- (сообщение, уровень серьезности, состояние)

RETURN;

END

END;

Тестируем

