



SPORTMAG PROJECT

ОГЛАВЛЕНИЕ

Создание базы данных и её заполнение	3
Добавление Триггеров	7
Добавление Процедур.....	15
Добавление Пользовательских функций	21

СОЗДАНИЕ БАЗЫ ДАННЫХ И ЕЁ ЗАПОЛНЕНИЕ

```
CREATE database Sportmag;  
use sportmag;
```

```
CREATE table Goods  
(id int identity(1,1)primary key,  
title nvarchar(100) check(title != '')not null,  
kind nvarchar(100) check(kind != '')not null,  
amount int check(amount>=0) not null default(0),  
selfprice money check(selfprice>=0) not null,  
factory nvarchar(100) check(factory!='')not null,  
price money not null);
```

```
CREATE table Staff  
(id int identity(1,1)primary key,  
Name nvarchar(30) check(Name!='')not null,  
Lname nvarchar(30) check(Lname!='')not null,  
Fname nvarchar(30) check(Fname!='')not null,  
post nvarchar(30) check(post!='')not null,  
first_day date not null CHECK(first_day < = GETDATE()),  
sex bit not null default(0),  
salary money not null);
```

```
CREATE table OldStaff  
(id int identity(1,1)primary key,  
Name nvarchar(30) check(Name!='')not null,  
Lname nvarchar(30) check(Lname!='')not null,  
Fname nvarchar(30) check(Fname!='')not null,  
post nvarchar(30) check(post!='')not null,  
first_day date not null,  
lAst_day date not null,  
sex bit not null default(0));
```

```
CREATE table Client  
(id int identity(1,1)primary key,  
Name nvarchar(30) check(Name!='')not null,  
Lname nvarchar(30) check(Lname!='')not null,  
Fname nvarchar(30) check(Fname!='')not null,  
email nvarchar(30) check(email LIKE ('%@%'))not null,  
phone INT not null CHECK(phone != ''),  
sex bit not null default(0),
```

```
percents DECIMAL (4,2) default(0),
subscribe bit not null default(0));
```

```
CREATE table Sales
(id int identity(1,1)primary key,
goods_id int not null Foreign key (goods_id) REFERENCES
goods(id),
full_price money not null,
amount int check(amount>=0) not null default(0),
sale_day date not null CHECK(sale_day <= GETDATE()),
staff_id int not null default(0) Foreign key (staff_id)
REFERENCES staff(id) ON DELETE CASCADE ,
client_id int not null Foreign key (client_id) REFERENCES
client(id));
```

```
CREATE table Archive
(id int identity(1,1) primary key,
goods_id INT not null Foreign key (goods_id) REFERENCES
goods(id));
```

```
Alter table goods add CONSTRAINT CHECK_SALE
check(price>selfprice);
Alter table staff add CONSTRAINT CHECK_SALARY check(salary>0);
Alter table OldStaff add CONSTRAINT CHECK_FIRST CHECK(first_day !=>
GETDATE());
Alter table OldStaff add CONSTRAINT CHECK_LAST CHECK(last_day >=
first_day);
```

```
INSERT INTO Goods (title, kind, amount, selfprice, factory, price)
VALUES
('Брюшко, гудбай', 'гантели', 77, 500, 'Кировский завод', 2000),
('Шпинделёк', 'унижатор ручной', 1, 100, 'Изба утех для тех и
тех', 5000),
('Жух-жух 17', 'скребок ушной', 1000, 199, 'Когда никто не
слышит', 400),
('Klizma', 'Prazdnik', 44, 200, 'StroyKlizmMASH', 1000),
('Неразжимное колечко', 'унижатор ручной', 15, 500, 'Изба утех для
тех и тех', 3000),
('Пальцевыворачивалка гидравлическая', 'унижатор ручной', 11,
1000, 'Изба утех для тех и тех', 10000);
```

```
INSERT INTO Client ( Name, Lname, Fname, email, phone, sex,
percents,
```

```

subscribe) VALUES
('Кшиштоф', 'Мухин', 'Агафонович', 'fly@mail.com', 656254, 1,
15.00, 1),
('Гвидон', 'Кулебякин', 'Пироксимович', 'gvido@mail.com', 253545,
1, 11.00, 0),
('Don', 'Juan', 'MAximovich', 'gvieo@mail.com', 2533345, 1,
14.00,1),
('Donna-Margarita', 'Juan', 'Nikolavna', 'gvo@mail.com',
2577345,0, 24.00, 1),
('Ivan', 'Hlebov', 'Oristarhovich', 'hlebal@mail.com',
233377345,1, 21.00, 1),
('Gleb', 'Hlebov', 'Oristarhovich', 'Glebal@mail.com',
233377344,1, 25.00, 1);

```

```

INSERT INTO Staff (Name, Lname, Fname, post, first_day, sex,
salary) VALUES
('Ганс', 'Поберушкин', 'Охламонович', 'страшный продаван',
'2014-05-05', 1, 2500),
('Клёпа', 'Фон Амурский', 'Сигизмундович', 'продажник 2
кат.', '2016-02-02', 1, 500),
('Галя', 'Гром', 'Победитовна', 'кассир-продавец', '2000-01-
01',0, 200),
('Стивен', 'Крючковски', 'Джонович', 'консультант-
инсультант', '2018-02-01', 1, 2100),
('Баба', 'Маня', 'Филипповна', 'уборщица-продавец', '1965-01-
01',0, 200),
('Зульфия', 'Наршланбэ', 'Барабековна', 'грузчица-
реализатор', '2021-07-05', 0, 800),
('Barabek', 'Chelovek', 'Poedatel', 'crisis manager',
'2017-07-07', 1, 7300),
('Islambek', 'Chelovek', 'Poedatel', 'manager of judgement
day', '2011-01-01', 1, 17300),
('Bob', 'Hlebov', 'Bobovich', 'bean mASter', '2014-04-01',
1,1300),
('George', 'Hlebov', 'Shprotovich', 'sandvich mASter',
'2013-04-04', 1, 1700);

```

```

INSERT INTO Sales ( goods_id, full_price, amount, sale_day,
staff_id, client_id) VALUES
(1, 4000, 3, '2022-06-10', 1, 1),
(1, 4000, 1, '2022-07-10', 1, 1),
(1, 4000, 5, '2022-06-11', 2, 1),
(2, 2000, 7, '2022-02-03', 3, 2),
(3, 7000, 5, '2021-02-11', 2, 2),
(1, 4000, 5, '2022-02-03', 4, 1),
(4, 8000, 7, '2011-06-10', 4, 3);

```

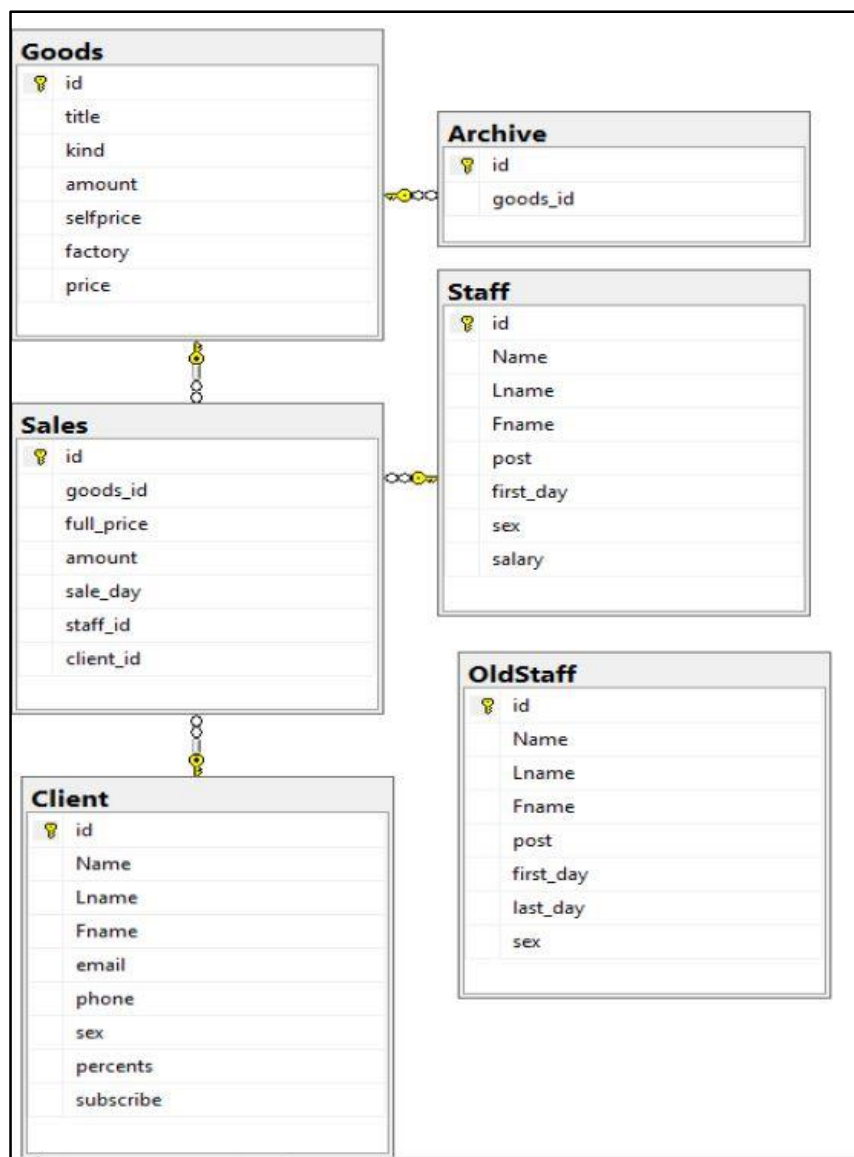


Рис.1 Диаграмма базы данных

ДОБАВЛЕНИЕ ТРИГГЕРОВ

1. При добавлении нового товара триггер проверяет его наличие на складе, если такой товар есть и новые данные о товаре совпадают с уже существующими данными, вместо добавления происходит обновление информации о количестве товара.

```
CREATE trigger Zavoz_starjia on goods
AFTER UPDATE as
BEGIN
IF (@@ROWCOUNT=0)
RETURN;
UPDATE goods SET goods.amount = goods.amount + inserted.amount
FROM inserted
WHERE goods.title = inserted.title AND goods.factory =
inserted.factory
END
```

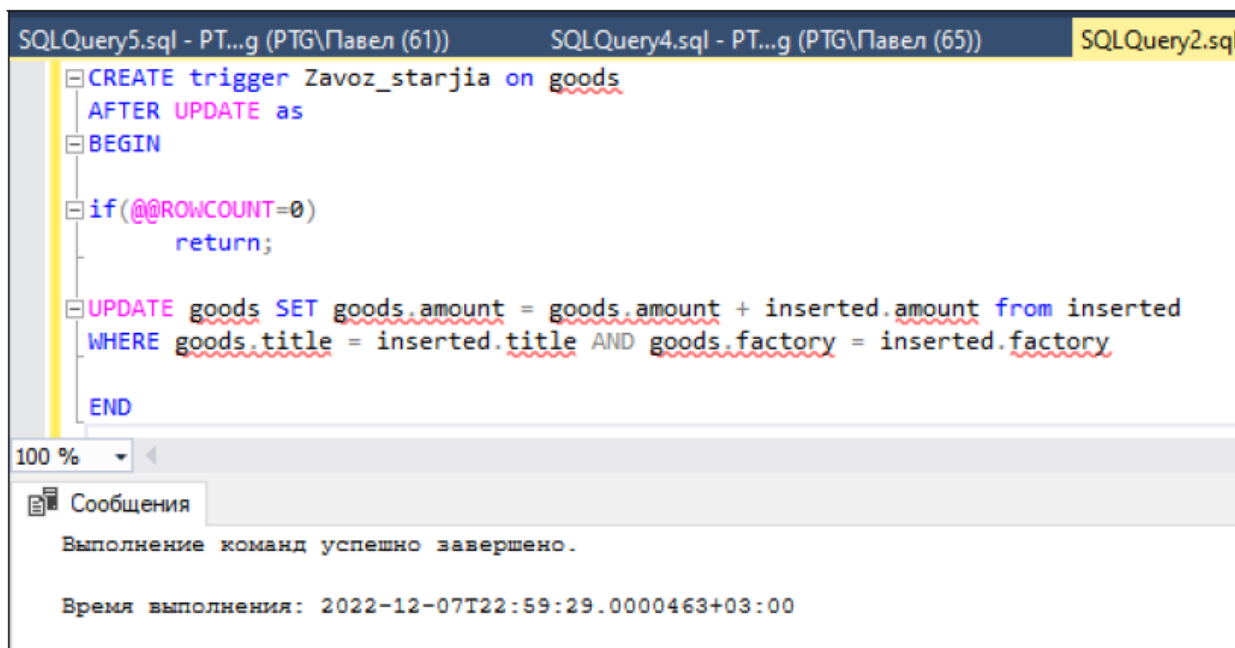


Рис. 2

2. Триггер проверяет что после продажи товара не осталось ни одной единицы данного товара, необходимо перенести информацию о полностью проданном товаре в таблицу «Архив».

```
CREATE trigger Empty_to_Archive on goods
AFTER UPDATE as
BEGIN
IF (@@ROWCOUNT=0)
RETURN;
INSERT INTO Archive (goods_id)  SELECT s.goods_id
FROM sales as s
WHERE s.amount = 0
END
```

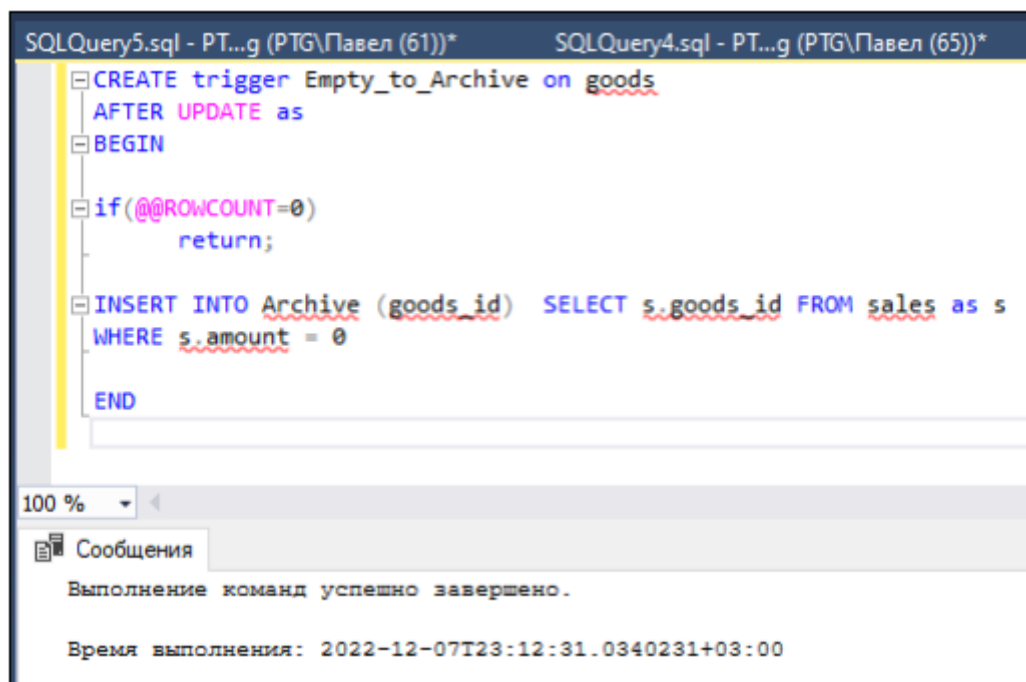


Рис. 3

3. Триггер запрещает добавлять нового продавца, если количество существующих продавцов больше 6.

```
CREATE trigger No_More_Six ON Staff
FOR INSERT
AS BEGIN
declare @how_much int
SET @how_much = (SELECT COUNT(id) FROM Staff)
declare @New_id int
SELECT @New_id = id FROM inserted
IF (@how_much > 6 AND @New_id > @how_much )
BEGIN
raiserror ('NO MORE SIX - Это значит что кто-то тут лишний!', 0,1)
ROLLBACK TRANSACTION
END
ELSE PRINT ('Добавлен новый герой')
END
```

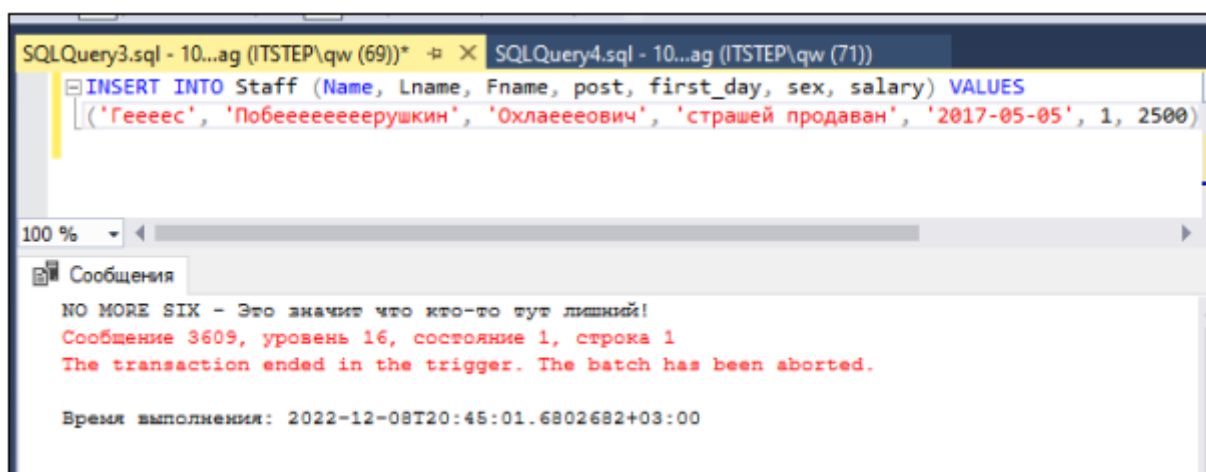


Рис. 4

4. Триггер запрещает удаление сотрудников, принятых на работу до 2015 года.

```
CREATE trigger No_Del_2015 on Staff
FOR DELETE as
BEGIN
IF (@@ROWCOUNT=0)
RETURN;
declare @bad_day date
SET @bad_day = (SELECT first_day FROM deleted)
IF (@bad_day < '2015-01-01' )
BEGIN
RAISERROR ('Такого не сотрёшь!', 0,1)
ROLLBACK TRANSACTION
END
ELSE PRINT ('Гудбай, товарищ!')
END
```

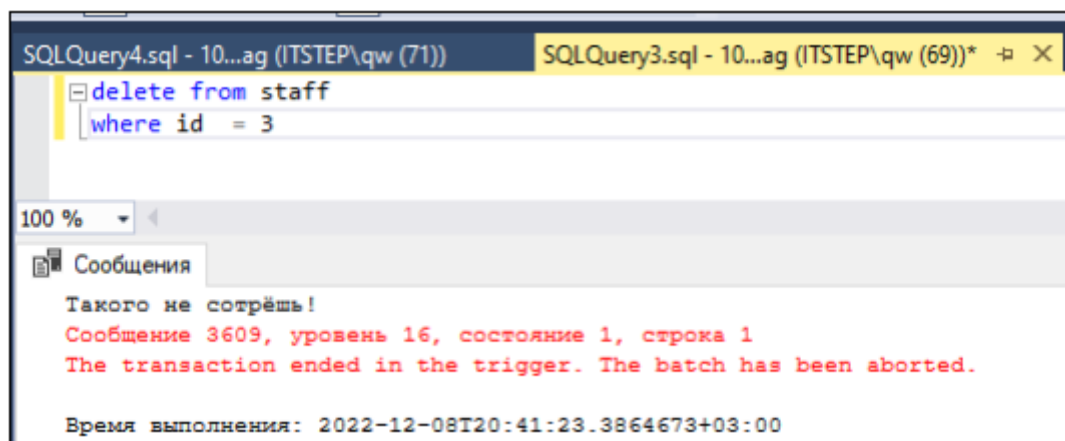


Рис.5

5. При добавлении продавца триггер проверяет есть ли он в таблице покупателей, если запись существует добавление нового продавца отменяется.

```
CREATE trigger Staff_not_client ON Staff
FOR INSERT
AS BEGIN
IF (EXISTS(SELECT staff.name, staff.lname, staff.fname
FROM staff
JOIN inserted ON staff.id= inserted.id
JOIN client on client.name = inserted.name
WHERE client.name = inserted.name
client.lname = inserted.lname
AND client.fname = inserted.fname))
BEGIN
RAISERROR('Не бывать Продавцу покупателем!',0,1)
ROLLBACK TRANSACTION
END
ELSE PRINT ('Добавлен новый герой')
END
```

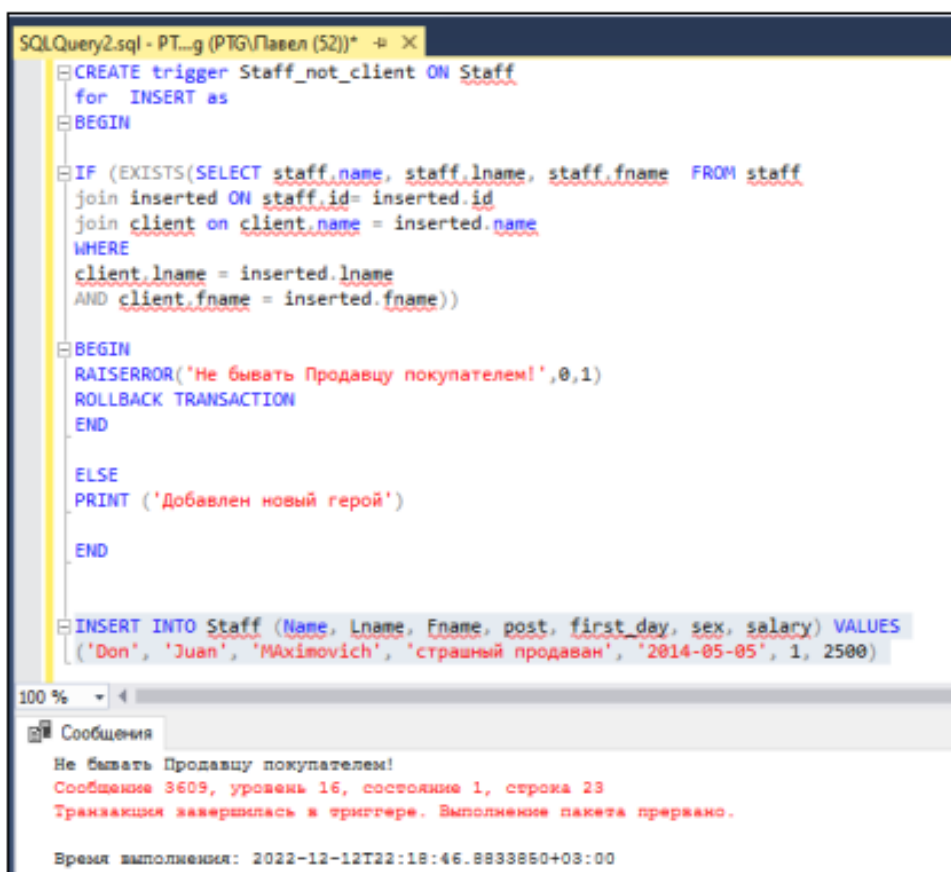


Рис. 6

6. Триггер не позволяет вставлять информацию о продаже таких товаров: яблоки, груши, сливы, кинза.

```
CREATE trigger Destroy_Fruit ON Sales
FOR INSERT as
BEGIN
IF (@@ROWCOUNT=0)
RETURN;
declare @fruit nvarchar(100)
SET @fruit = (SELECT kind FROM Goods
JOIN inserted ON goods.id = inserted.goods_id
WHERE goods.kind in ('яблоки')
OR goods.kind in ('груши')
OR goods.kind in ('сливы')
OR goods.kind in ('кинза'))
IF (@fruit = ('яблоки')
OR @fruit = ('груши')
OR @fruit = ('сливы')
OR @fruit = ('кинза'))
BEGIN
RAISERROR ('Это спортмагазин, а не овощебаза!!',0,1)
ROLLBACK TRANSACTION
END
ELSE PRINT('Если ты не овощ, тогда ты идешь к нам!')
END
```

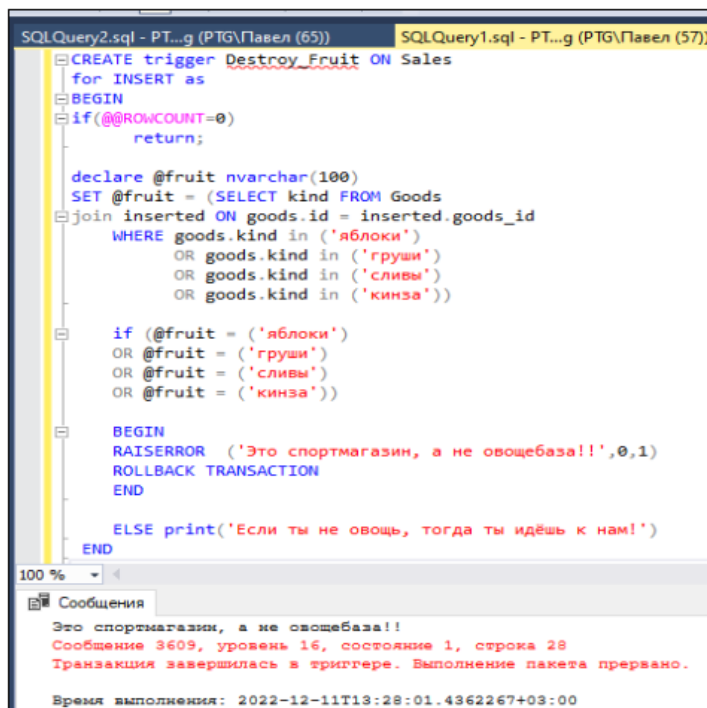


Рис. 7

7. Триггер не позволяет регистрировать уже существующего клиента. При вставке проверять наличие клиента по ФИО и e-mail.

```
CREATE trigger Only_New ON Client
INSTEAD OF INSERT AS
BEGIN
IF EXISTS(SELECT * FROM Client
JOIN inserted ON Client.name = inserted.name
WHERE
Client.name = inserted.name AND
Client.lname = inserted.lname AND
Client.fname = inserted.fname AND
Client.email = inserted.email )
BEGIN
PRINT ('Клиент уже под колпаком!')
ROLLBACK TRANSACTION
END
ELSE print('Падхади дарагой! Всё дешёво-вкусно BAX!')
END
```

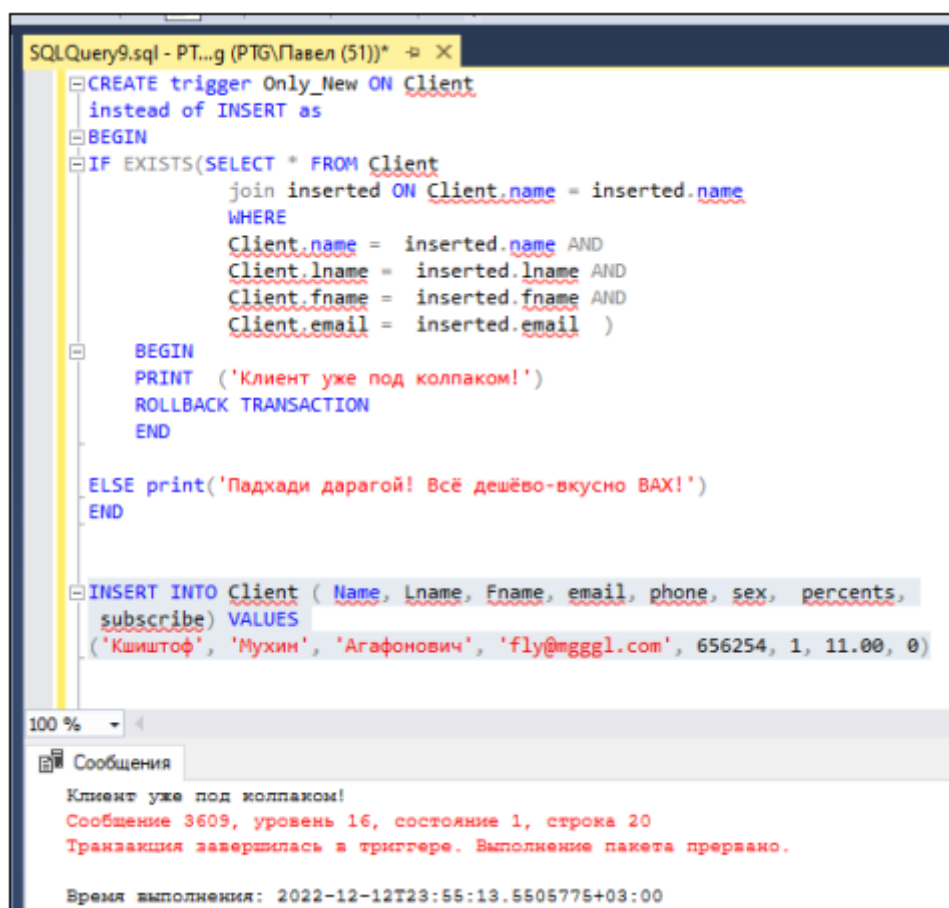


Рис. 8

8. Триггер запрещает удаление существующих клиентов.

```
CREATE TRIGGER Client_must_Live ON dbo.client
FOR DELETE
AS BEGIN
PRINT 'Клиент наш Друг и Кормилец! Ай-Вэй ! Зачем удаляешь,
дарагой?'
ROLLBACK TRANSACTION
END
```

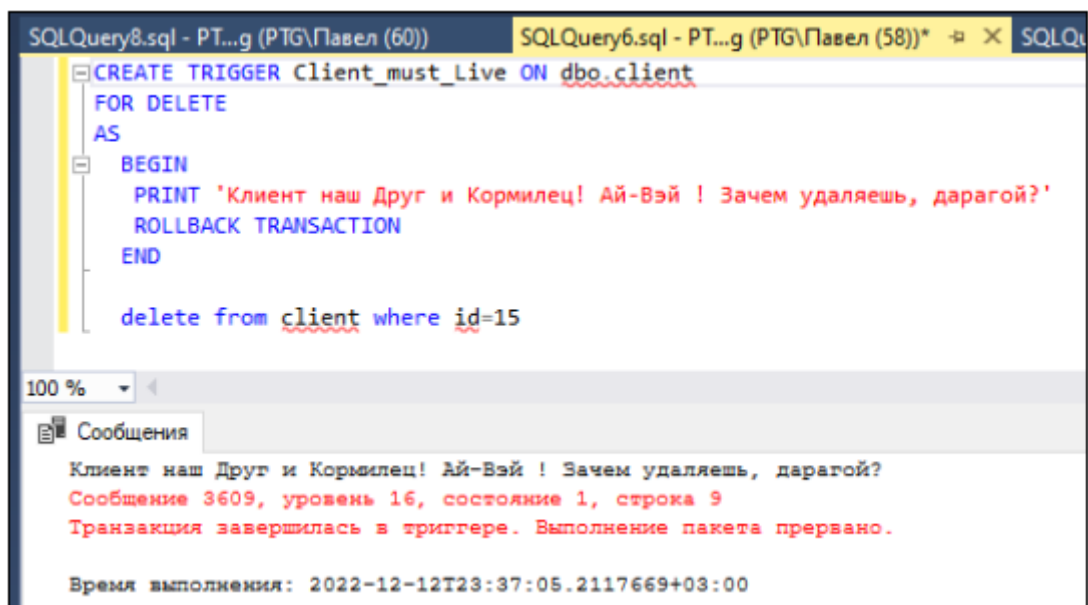


Рис. 9

ДОБАВЛЕНИЕ ПРОЦЕДУР

1. Хранимая процедура показывает информацию о всех продавцах.

```
CREATE procedure ALL_salers AS  
SELECT name + ' ' + lname + ' ' + fname as 'Наш продажный коллектив'  
FROM Staff
```

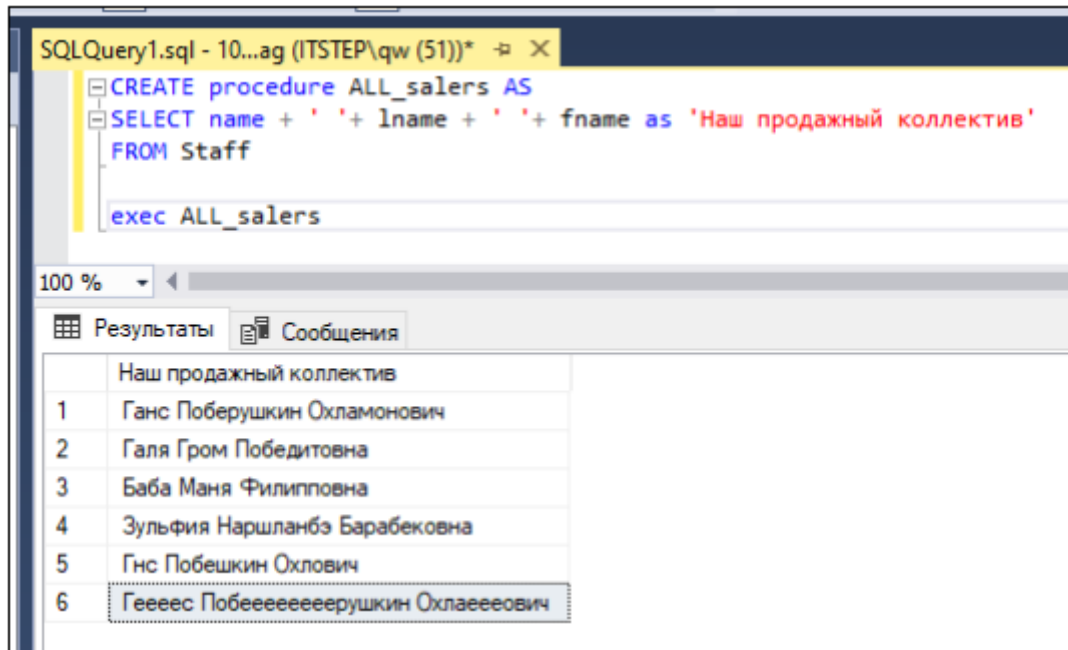


Рис.10

2. Хранимая процедура показывает информацию о всех покупателях.

```
CREATE procedure ALL_clients AS  
SELECT name + ' ' + lname + ' ' + fname as 'Покупатели, Кормильцы,  
Отцы'  
FROM Client
```

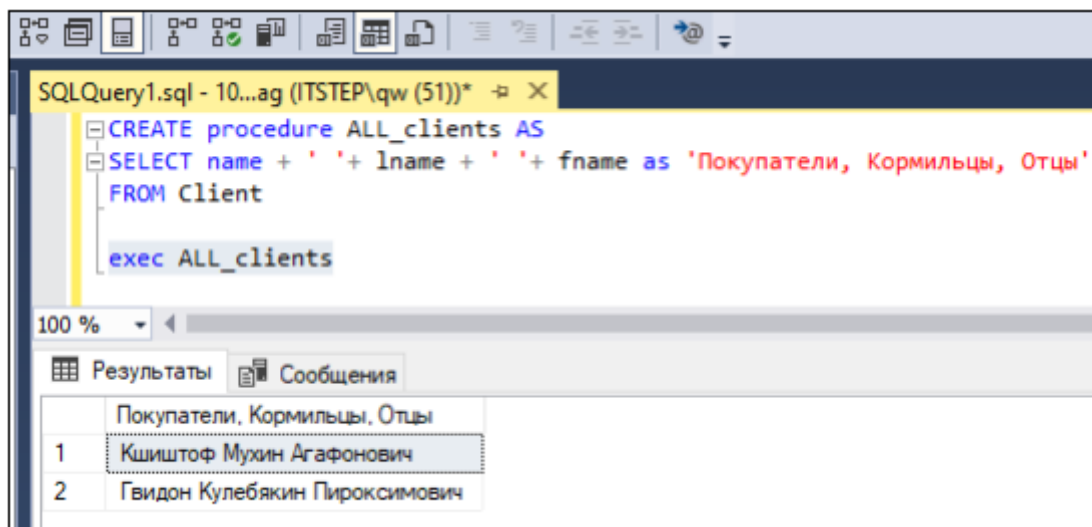


Рис.11

3. Хранимая процедура показывает полную информацию о продажах.

```
CREATE procedure All_Our_Money AS  
SELECT SUM(amount*full_price) as 'Наша выручка'  
FROM Sales
```

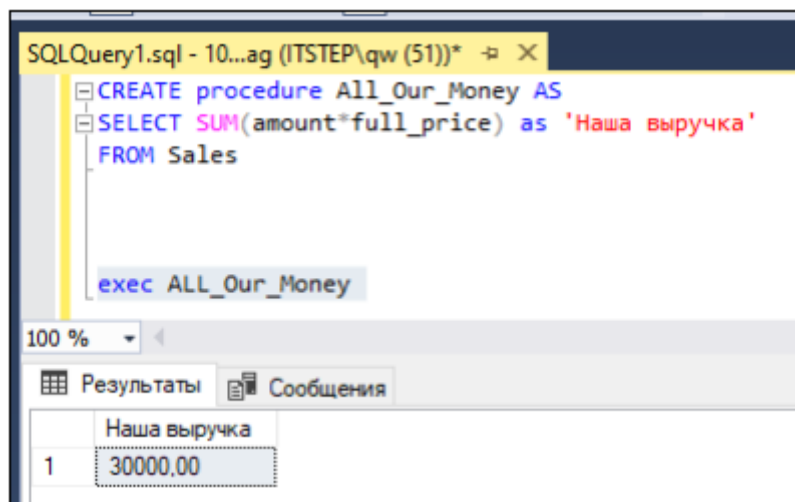


Рис.12

4. Хранимая процедура показывает полную информацию о всех продажах в конкретный день. Дата продажи передается в качестве параметра.

```
CREATE procedure Sales_On_Date (@date date) AS  
BEGIN  
SELECT * FROM Sales  
WHERE sale_day = @date  
END
```

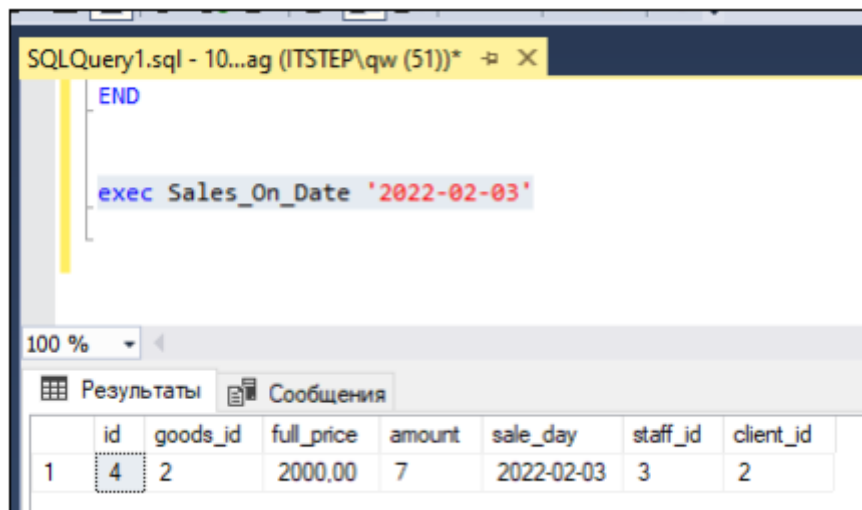


Рис.13

5. Хранимая процедура отображает информацию о продажах конкретного продавца. ФИО продавца передается в качестве параметра хранимой процедуры.

```
CREATE procedure Saler_Statistic (@lname nvarchar(50), @name  
nvarchar(50), @fname nvarchar(50)) AS  
BEGIN  
SELECT g.title as 'Название товара', s.full_price*s.amount as  
'Добыча'  
FROM Sales as s  
JOIN Goods as g ON g.id = s.goods_id  
JOIN Staff as st ON st.id=s.staff_id  
WHERE st.lname = @lname  
AND st.name = @name AND st.fname = @fname  
END
```

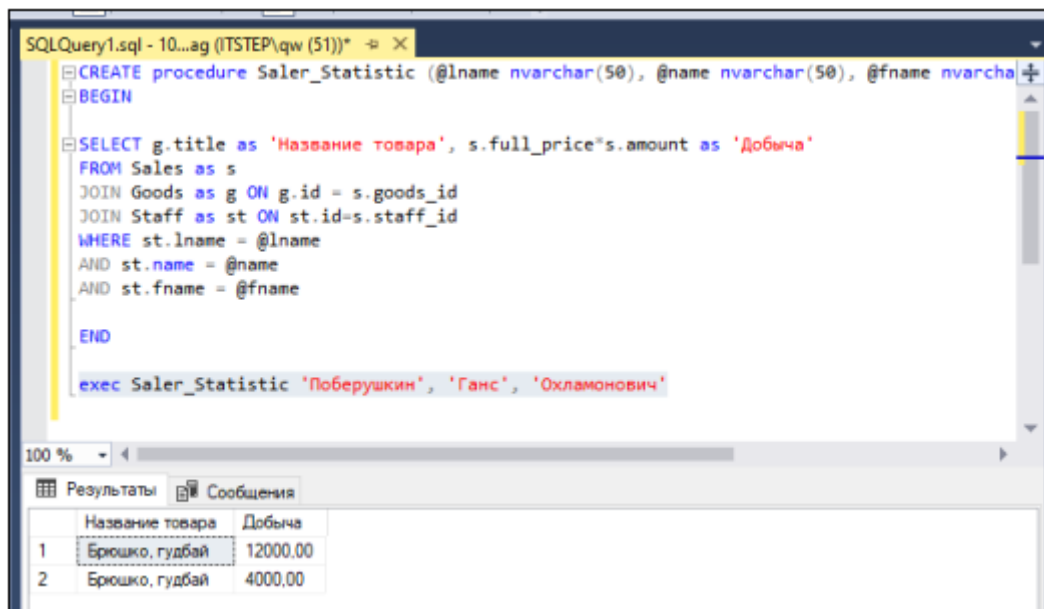


Рис.14

6. Хранимая процедура возвращает среднюю арифметическую цену продажи в конкретный год. Год передаётся в качестве параметра.

```
CREATE procedure AVG_Sales_On_Year (@date date) AS  
BEGIN  
SELECT AVG (full_price)  
FROM Sales  
WHERE DATEDIFF(year, sales.sale_day, @date)=0  
END
```

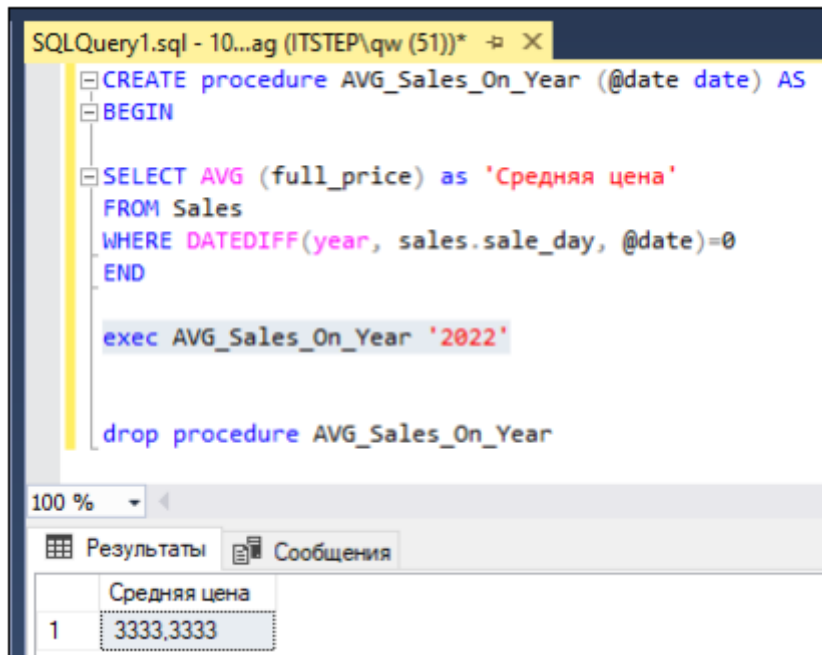


Рис.15

ДОБАВЛЕНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ФУНКЦИЙ

1. Пользовательская функция возвращает дату, когда общая сумма продаж за день была максимальной.

```
CREATE function Best_day()  
RETURNS date  
AS BEGIN  
DECLARE @Best date  
SET @Best = (Select sale_day FROM Sales  
WHERE full_price*amount = (SELECT MAX(full_price*amount)  
FROM sales))  
RETURN @Best  
END
```

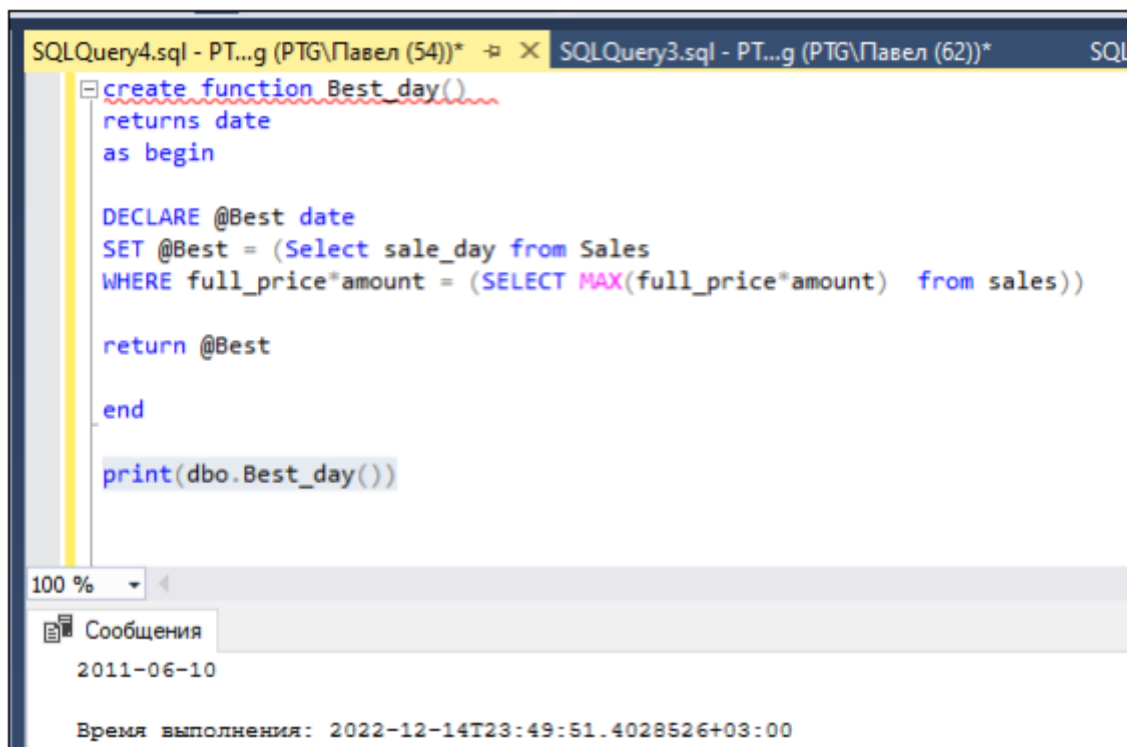


Рис. 16

2. Пользовательская функция возвращает информацию о всех продажах заданного товара. Название товара передаётся в качестве параметра.

```
CREATE function All_of_Saled_Thing(@T nvarchar (50))
RETURNS TABLE AS
RETURN (Select g.title AS Thing, s.sale_day AS Day_of_sale,
s.full_price AS Cost, s.amount AS Amount,
s.full_price*s.amount AS Full_Money
FROM Sales AS S
JOIN goods AS g ON g.id = s.goods_id
WHERE g.title = @T)
```

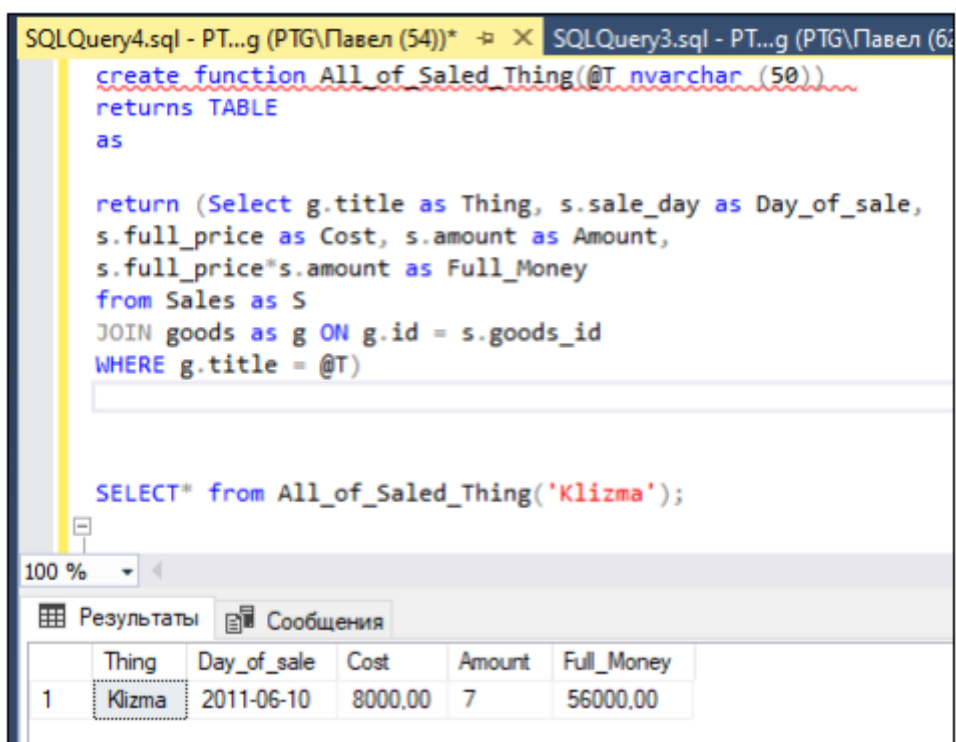


Рис.17

3. Пользовательская функция возвращает информацию о всех продавцах однофамильцах.

```
CREATE function LName_Matchers1 (@LN nvarchar (50))  
RETURNS TABLE AS  
RETURN (Select * FROM Staff AS St  
WHERE st.lname = @LN);
```

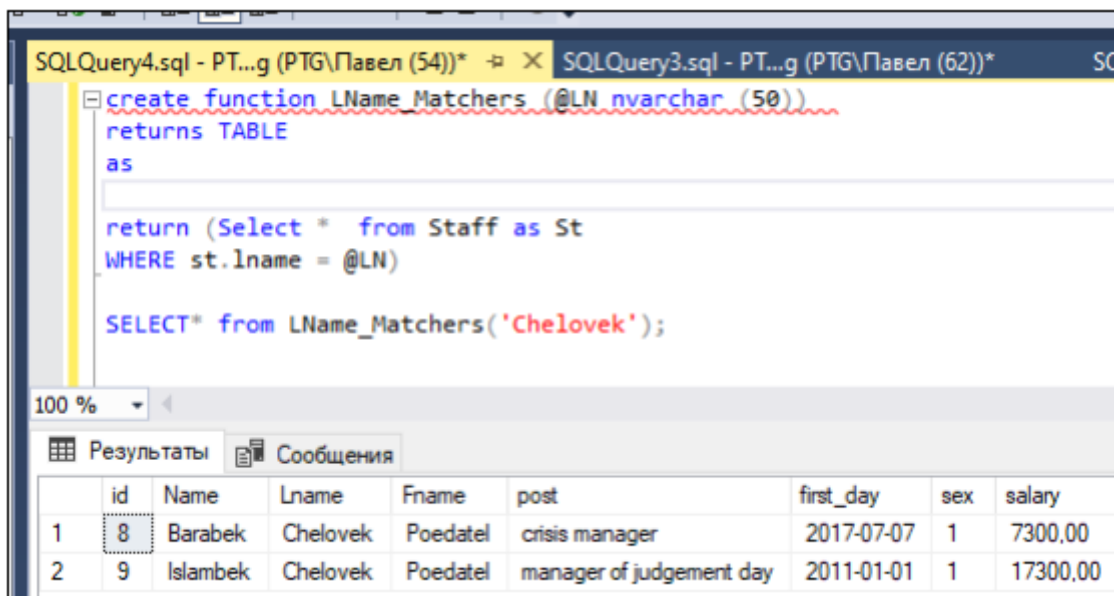


Рис.18

4. Пользовательская функция возвращает информацию о всех покупателях однофамильцах.

```
CREATE function LName_clients (@LNC nvarchar (50))  
RETURNS TABLE AS  
RETURN (Select * FROM Client AS c  
WHERE c.lname = @LNC)
```

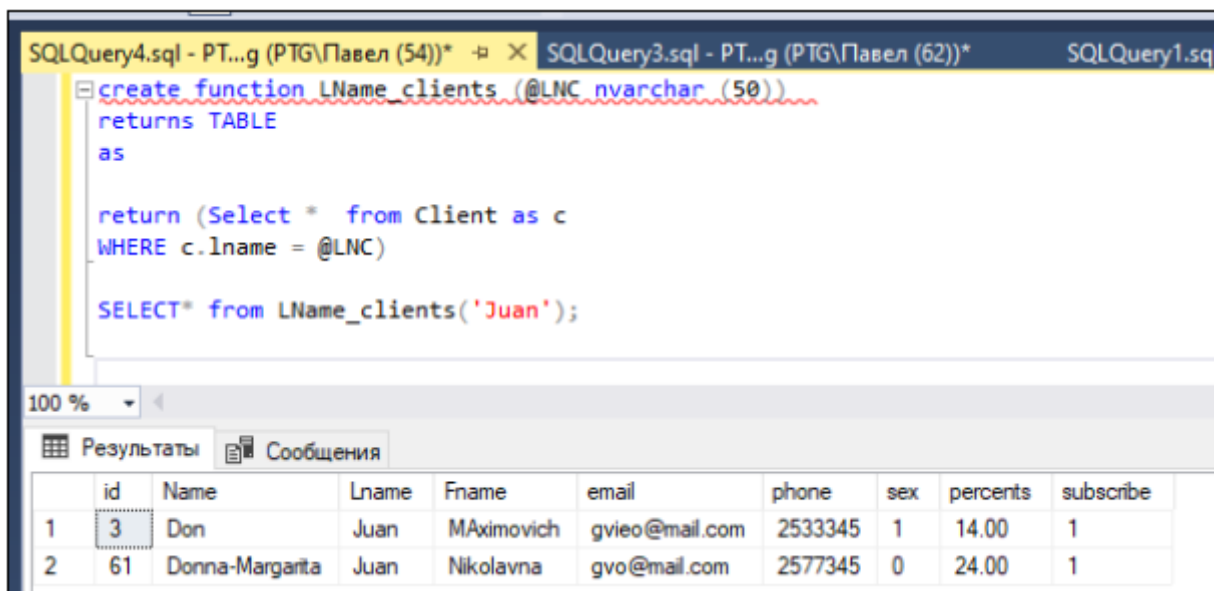


Рис.19

5. Пользовательская функция возвращает информацию о всех покупателях и продавцах однофамильцах.

```
CREATE function ALL_LName_matches (@LA nvarchar (50))
RETURNS @Teskas TABLE (First_name nvarchar (50) not null,
Last_Name nvarchar (50) not null,
Fathers_Name nvarchar (50) not null)
AS BEGIN
DECLARE @tmp_teskas table (Name nvarchar (50) not null,
Lname nvarchar (50) not null, F_Name nvarchar (50) not null)
INSERT @tmp_teskas Select c.Name, c.Lname, c.fname FROM Client AS
C WHERE c.lname = @LA
INSERT @tmp_teskas Select st.Name, st.Lname, st.Fname FROM Staff
AS st WHERE st.lname = @LA
INSERT @Teskas SELECT* FROM @tmp_teskas
RETURN
END;
```

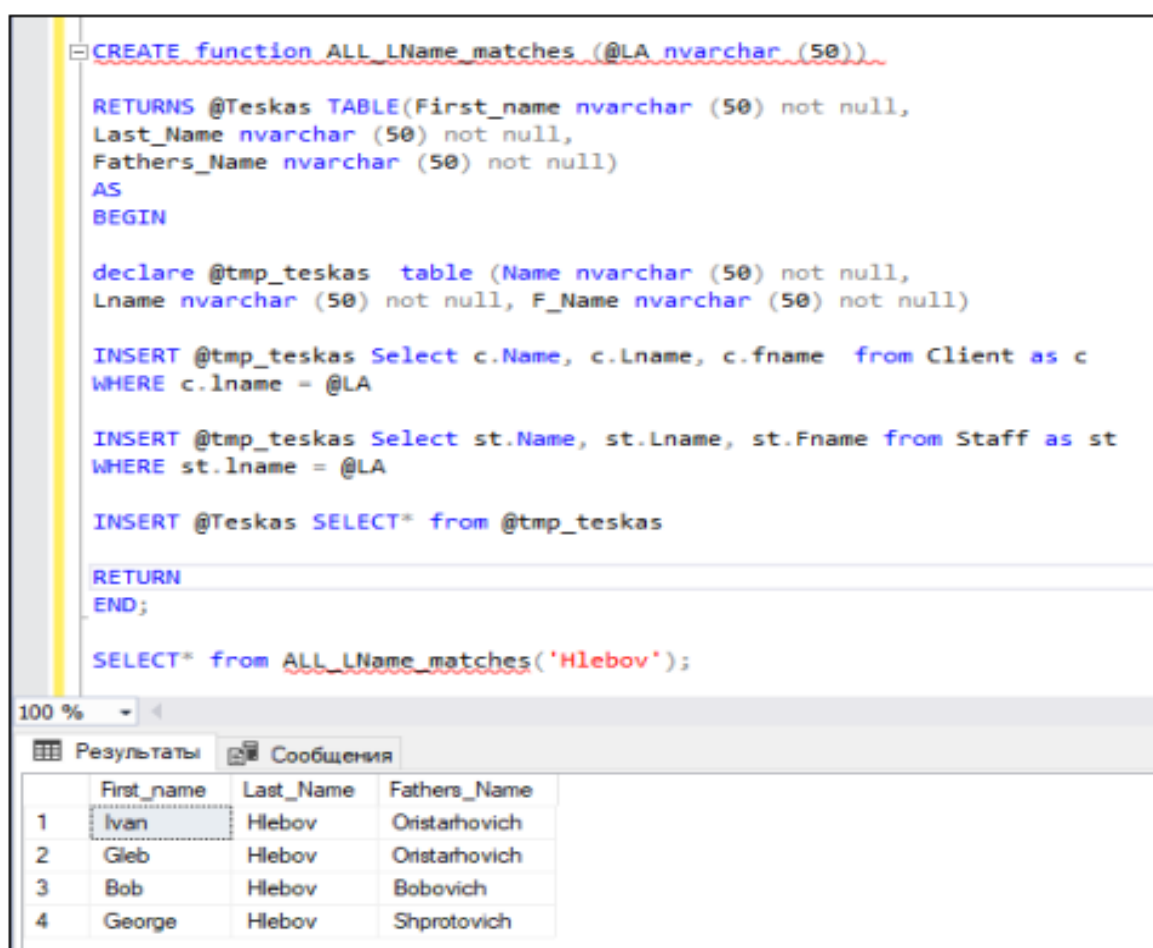


Рис.20

6. Пользовательская функция возвращает количество уникальных покупателей.

```
CREATE function Uni_CL()  
RETURNS int  
AS BEGIN  
DECLARE @temple table (Full_Name nvarchar(100))  
INSERT INTO @temple SELECT DISTINCT c.lname + ' ' + c.name + ' ' +  
c.fname AS 'Full_Name' FROM Client as c  
DECLARE @uniq int  
SET @uniq = (SELECT count(Full_Name) from @temple)  
RETURN @uniq  
END
```

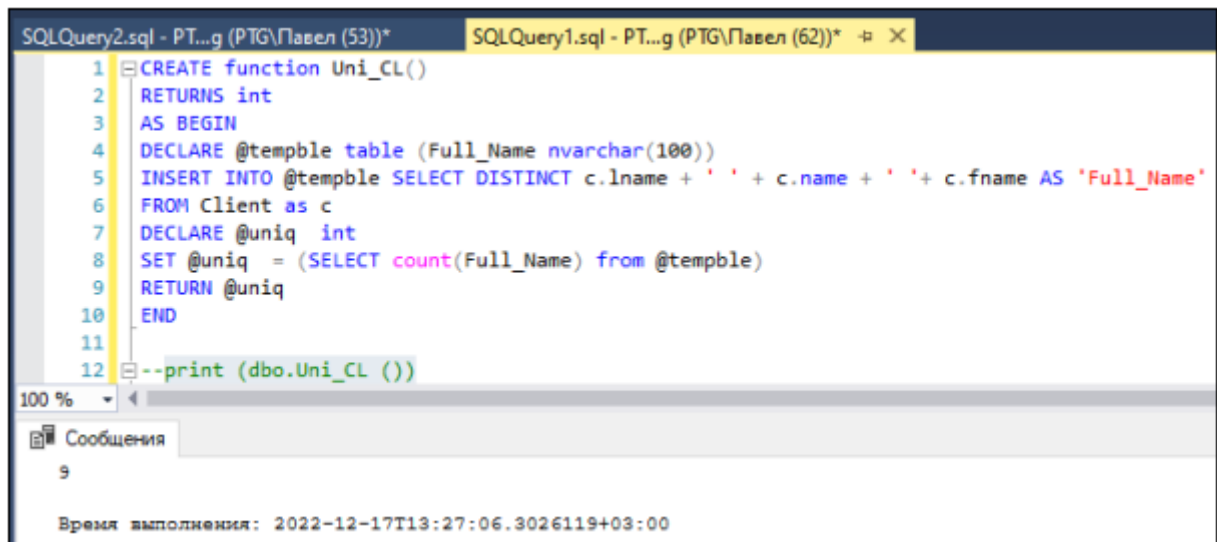


Рис.21

7. Пользовательская функция возвращает среднюю цену товара конкретного вида. Вид товара передаётся в качестве параметра. Например, среднюю цену обуви.

```
CREATE function Sred_price (@kind nvarchar(50))
RETURNS money
AS BEGIN
DECLARE @AVG money
SET @AVG = (select AVG(full_price) FROM Sales as S
JOIN goods as g ON g.id = s.goods_id
WHERE g.kind = @kind)
RETURN @AVG
END
```

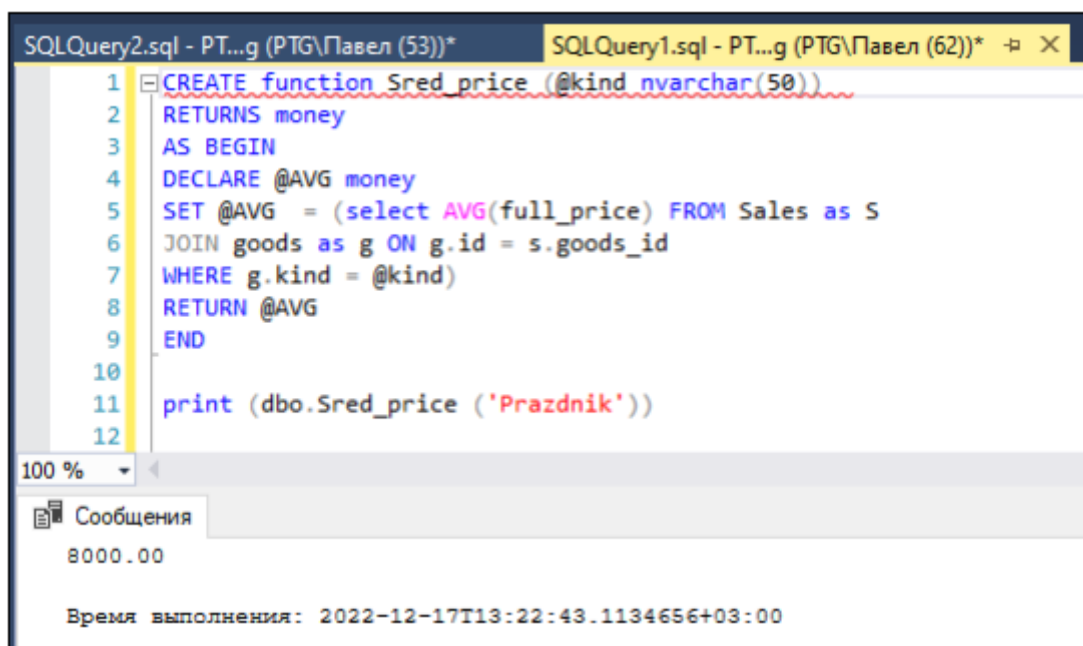
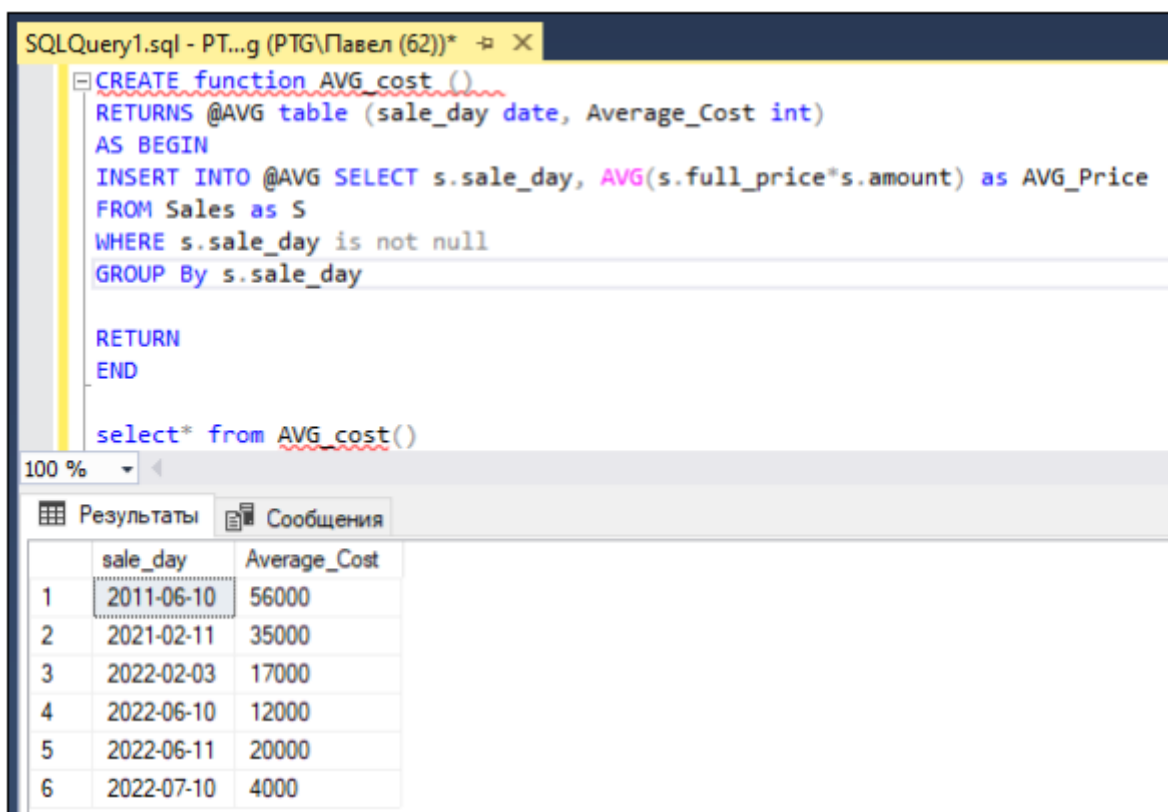


Рис.22

8. Пользовательская функция возвращает среднюю цену продажи по каждой дате, когда осуществлялись продажи.

```
CREATE function AVG_cost ()  
RETURNS @AVG table (sale_day date, Average_Cost int)  
AS BEGIN  
INSERT INTO @AVG SELECT s.sale_day, AVG(s.full_price*s.amount) as  
AVG_Price  
FROM Sales as S  
WHERE s.sale_day is not null  
GROUP By s.sale_day  
RETURN  
END
```



SQLQuery1.sql - PT...g (PTG\Павел (62))

```
CREATE function AVG_cost ()  
RETURNS @AVG table (sale_day date, Average_Cost int)  
AS BEGIN  
INSERT INTO @AVG SELECT s.sale_day, AVG(s.full_price*s.amount) as AVG_Price  
FROM Sales as S  
WHERE s.sale_day is not null  
GROUP By s.sale_day  
  
RETURN  
END  
  
select* from AVG_cost()
```

100 %

Результаты Сообщения

	sale_day	Average_Cost
1	2011-06-10	56000
2	2021-02-11	35000
3	2022-02-03	17000
4	2022-06-10	12000
5	2022-06-11	20000
6	2022-07-10	4000

Рис.23

9. Пользовательская функция возвращает информацию о последнем проданном товаре. Критерий определения последнего проданного товара: дата продажи.

```
CREATE function Last_sale_from (@date date)
RETURNS @last table (Title nvarchar(50), Sale_Amount int, Price
money, Sale_Date date)
AS BEGIN
DECLARE @s_day date
SET @s_day= (select top (1) sale_day from Sales as S
WHERE s.sale_day < @date
ORDER BY sale_day desc);
INSERT INTO @last SELECT g.title, s.amount, s.full_price,
s.sale_day
FROM sales as S
JOIN goods as g ON g.id = s.goods_id
WHERE s.sale_day = @s_day
RETURN
END
```

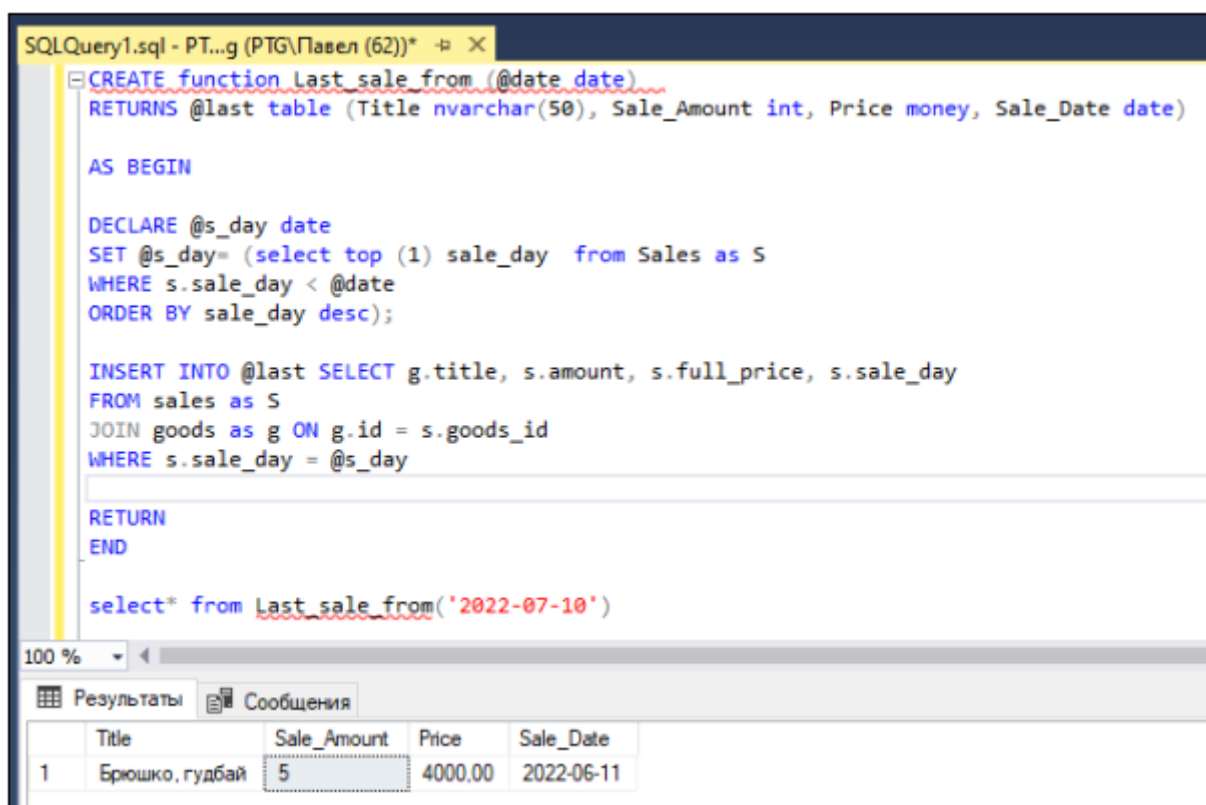


Рис. 24

10. Пользовательская функция возвращает информацию о первом проданном товаре. Критерий определения первого проданного товара: дата продажи.

```
CREATE function First_sale_from (@date date)
RETURNS @first table (Title nvarchar(50), Sale_Amount int, Price
money, Sale_date date)
AS BEGIN
DECLARE @s_day date
SET @s_day= (select top (1) sale_day from Sales as S
WHERE s.sale_day >= @date
order by sale_day);
INSERT INTO @first SELECT g.title, s.amount, s.full_price,
s.sale_day
FROM sales as S
JOIN goods as g ON g.id = s.goods_id
WHERE s.sale_day = @s_day
RETURN
END
```

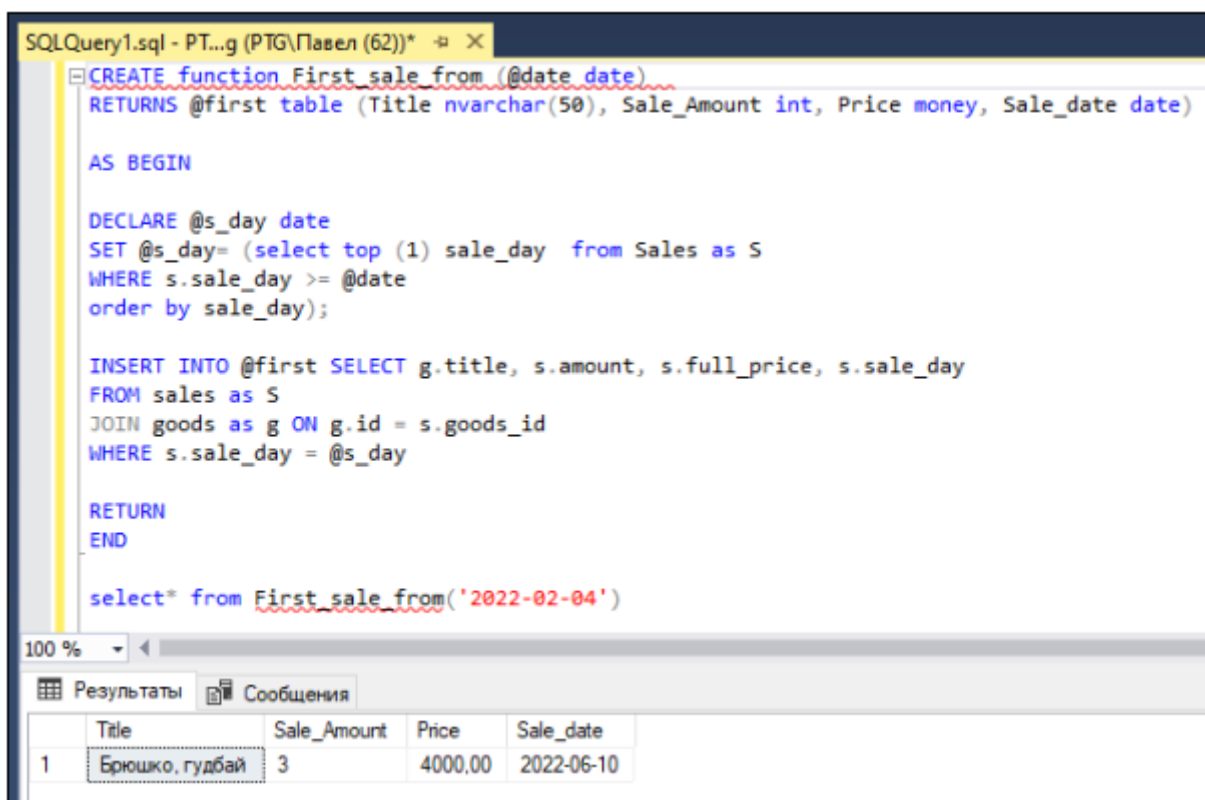


Рис.25

11. Пользовательская функция возвращает информацию о заданном виде товаров конкретного производителя. Вид товара и название производителя передаются в качестве параметров.

```
CREATE function What_about_kind (@kind nvarchar(50), @Produce
nvarchar(50))
RETURNS @about_kind table (Title nvarchar(50), Amount int,
Self_Price money, Price money )
AS
BEGIN
INSERT INTO @about_kind SELECT g.title , g.amount, g.selfprice,
g.price
FROM goods as g
WHERE g.kind = @kind
AND g.factory = @Produce

RETURN
END
```

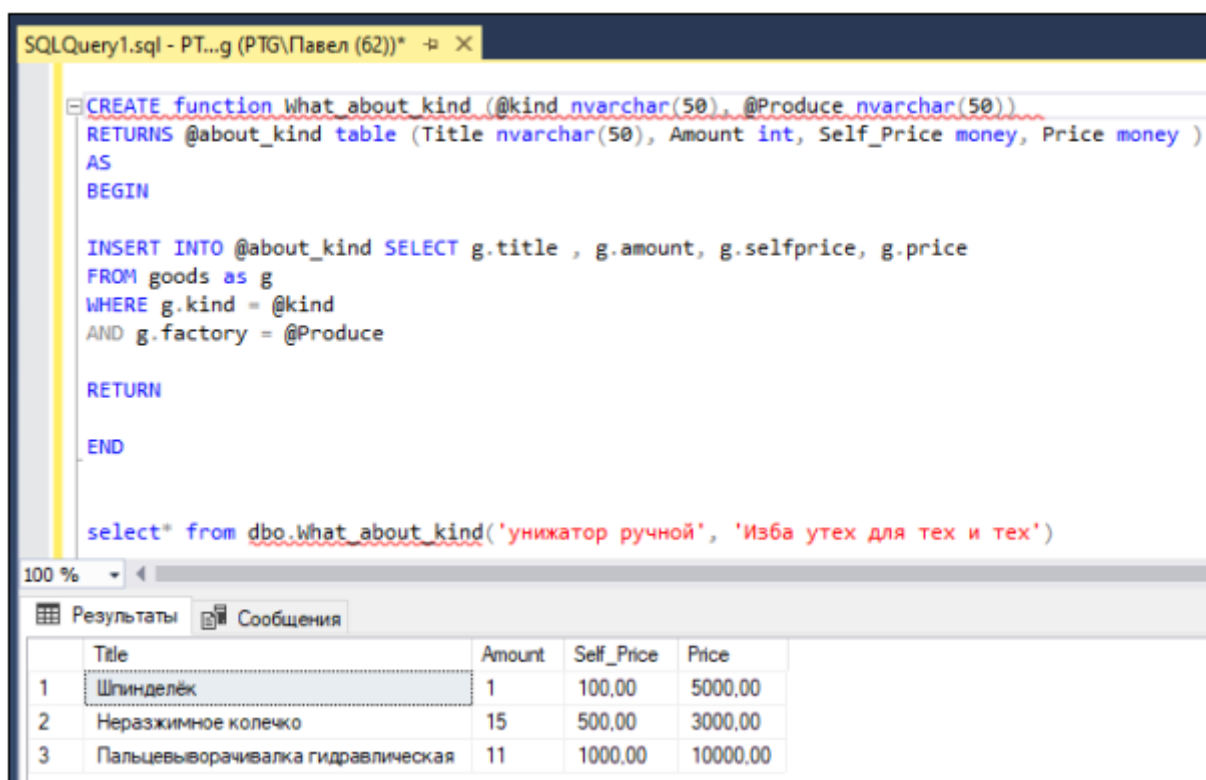


Рис.26



SPORTMAG PROJECT

Павел Макеенко, QA-21
Top Academy SPB, 2022