Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**Национальный исследовательский университет «МЭИ»**

**Задание на типовой расчет**

*МЕБЕЛЬНАЯ ФИРМА*

Выполнил

Студент Артемьев А.И.

Группа А-07-18

Дата 29.09.2020

**Москва 2020 НИУ МЭИ**

Компания занимается производством и продажей мебели.

**1)База данных должна хранить следующие данные:**

1) Каталог мебели: идентификатор, категория(отдел), вид, цвет, материал, цена

Например: (id1234, мягкая мебель, угловой диван, серый, ткань, 40000)

2) Список работников: идентификатор, ФИО, отдел, зарплата

Например: (id5678, Иванов Иван Иванович, мягкая мебель, 35000)

3) Список заказов: идентификатор заказа, идентификатор заказанной мебели(из каталога мебели), ФИО заказчика, адрес заказчика, статус товара.

Например: (id4321, id1234, Петров Петр Петрович, ул. Ленина 15, готов к отправлению)

**2)Порядок работы фирмы:**

Пользователь выбирает в каталоге мебель, которую он хочет заказать и указывает свое ФИО и адрес доставки.

В отделе приема и отправки рассматривают заказ пользователя и если все указано верно, отправляют заказ в соответствующий отдел производства фирмы, в зависимости от вида мебели.

Заказ добавляется в базу данных и для него устанавливают статус *На изготовке*

После изготовления работник отдела приема и отправки уведомляет заказчика, что товар готов посредством изменения его статуса на *Готов к доставке*. Далее он выбирает свободного водителя и отправляет заказ, следовательно статус водителя становится *занят* пока он не доставит товар, а статус товара *В пути.*

После получения мебели заказчиком работник отдела приема и отправки отмечает, что водитель свободен и товар доставлен.

**3)Требуется реализовать следующие группы пользователей:**

Пользователь:

-просматривает каталог товаров

-выбирает нужную мебель и оформляет заказ

-просматривает информацию о статусе готовности товара

Работник отдела приема и отправки товаров:

-устанавливает в какой отдел товар будет отправлен на изготовление

-находит свободного водителя, который будет доставлять товар

-устанавливает статус готовности товара на данный момент

-устанавливает, что водитель доставил товар и свободен

Администратор:

-просматривает каталог товаров

-вносит поправки в каталог (добавляет и удаляет товары, изменяет цены на них)

-увольняет/добавляет работников, изменяет их зарплаты

-просматривает список всех заказов

**4)** **Концептуальное проектирование: ER-диаграмма. Таблицы (не менее 3), индексы (обосновать), ключи (обосновать), связи, ограничения. ER-диаграмма.**

Проведем нормализацию

Таблица заказы

Первичный ключ IdOrder

До нормализации имеется одна таблица, хранящая все данные, при этом поля не являются атомарными.

Произведем нормализацию к первой нормальной форме. Создадим первичный ключ id, сделаем все поля атомарными. Получим следующую таблицу:

|  |
| --- |
| Id |
| FIOcustomer |
| DeliveryAddress |
| StatusOrdre |
| СategoryProduct |
| Type |
| color |
| materyal |
| price |
| FIOcustomer |
| DeliveryAddress |
| StatusOrdre |
| FIOWorker |
| Сategoryworker |
| salaries |
| FIODriver |
| Statusdriver |

Приведем ко 2 нормальной форме

Первичный ключ IdOrder

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IdOrder | FirstNameCustomer | MiddleCustomer | LastNameCustomer | DeliveryAddress |
| Сategory | Type | color | materyal | price | Status |

Таблица работники

Первичный ключ IdWorker

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| IdWorker | FirstNameWorker | MiddleWorker | LastNameWorker | Сategory | salaries | TypeWorker |

Таблица Водители

Первичный ключ IdDriver

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| IdDriver | FirstNameDriver | MiddleDriver | LastNameDriver | Status | OrderId |

Приведем к 3 нормальной форме

Таблица заказы

Первичный ключ: IdOrder

Внешний ключ: CustomerId, ProductId, DriverId ссылаются связью многие к одному на таблицу заказчик и продукт.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IdOrder | CustomerId | ProductId | status | DriverId |

Таблица заказчик

Первичный ключ: IdCustomer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IdCustomer | FirstNameCustomer | MiddleCustomer | LastNameCustomer | DeliveryAddress |

Таблица товар

Первичный ключ: IdProduct

Внешний ключ: СategoryId ссылается на таблицу категории с связью один ко многим.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| IdProduct | СategoryId | Type | color | materyal | price |

Таблица Водители

Первичный ключ: IdDriver

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IdDriver | FirstNameDriver | MiddleDriver | LastNameDriver | Status |

Таблица категориия продукта

Первичный ключ: IdСategory ссылается на таблицу категории с связью один ко многим.

|  |  |
| --- | --- |
| IdСategory | Name |

Таблица работники

Первичный ключ IdWorker

Внешний ключ СategoryId ссылается на таблицу категории с связью один ко многим.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| IdWorker | FirstNameWorker | MiddleWorker | LastNameWorker | СategoryId | salaries | TypeWorker |

Таблица отдел работника

Первичный ключ: IdСategory

|  |  |
| --- | --- |
| IdСategory | Name |

Для реализации права доступа добавим таблицу для администраторов и работников которая хранит: логин, пароль, уровень доступа.

Таблица администратор

Первичный ключ: IdAdmin

Внешний Ключ WorkerId ссылается на таблицу Worker связью один к одному.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IdAdmin | Login | Parol | Dostup | WorkerId |

Для поиска и сортировки, будут использоваться следящие индексы:

Индекс по категории товара, нужен для поиска товаров по категории

Индекс по статусу водителя нужен при поиске не занятого водителя

Индекс по ФИО сотрудника для поиска сотрудника.

Ограничения:

Логин должен быть уникальным

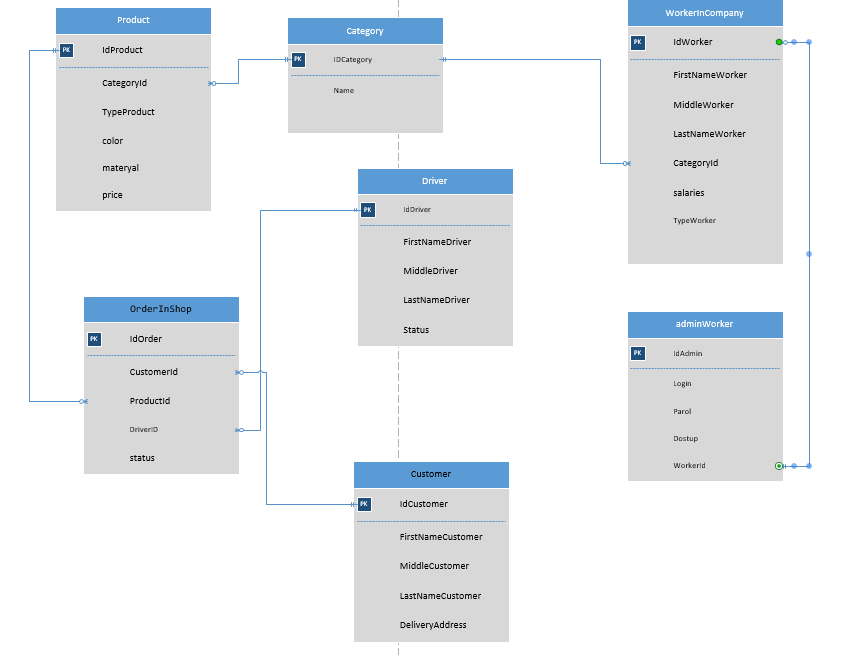
Если работник относится к отделам производства, то он должен относиться к одному из отделов производства.

Все поля кроме: CategoryId у таблицы WorckerInCompany, StatusDriver у таблицы Driver не могут быть пустыми.

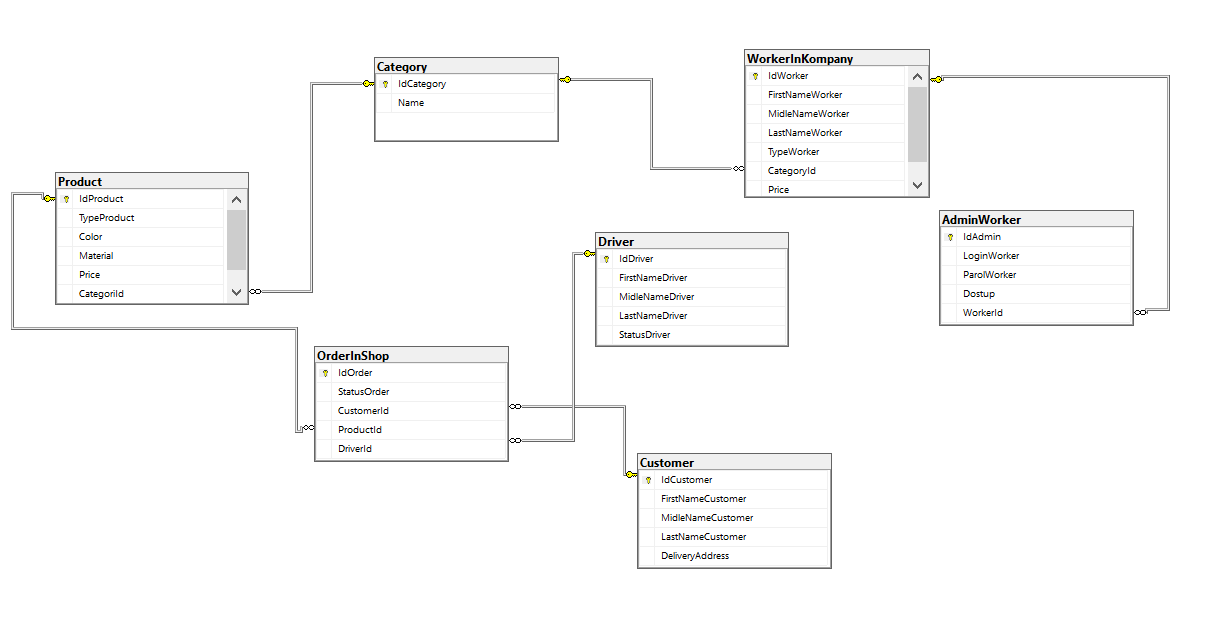
Цена и зарплата должны быть больше нуля.

Пока водитель занят доставкой товара, он не может брать новые товары.

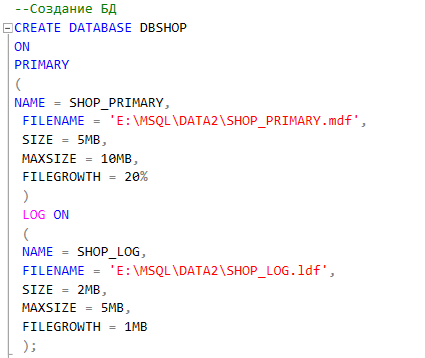
ER Диаграмма:



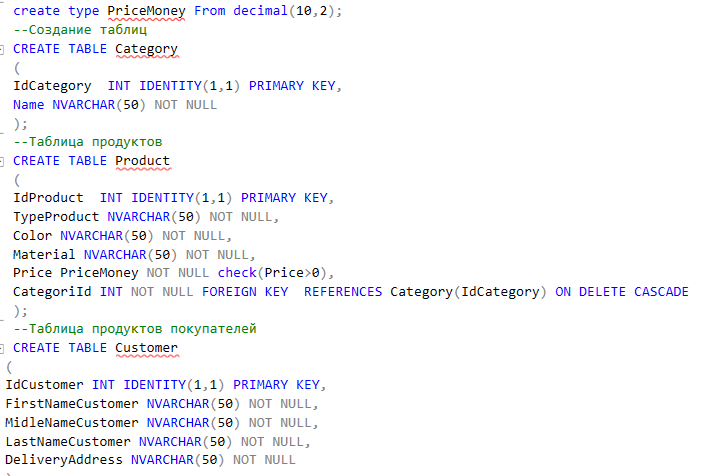
**4)Физическая модель БД**

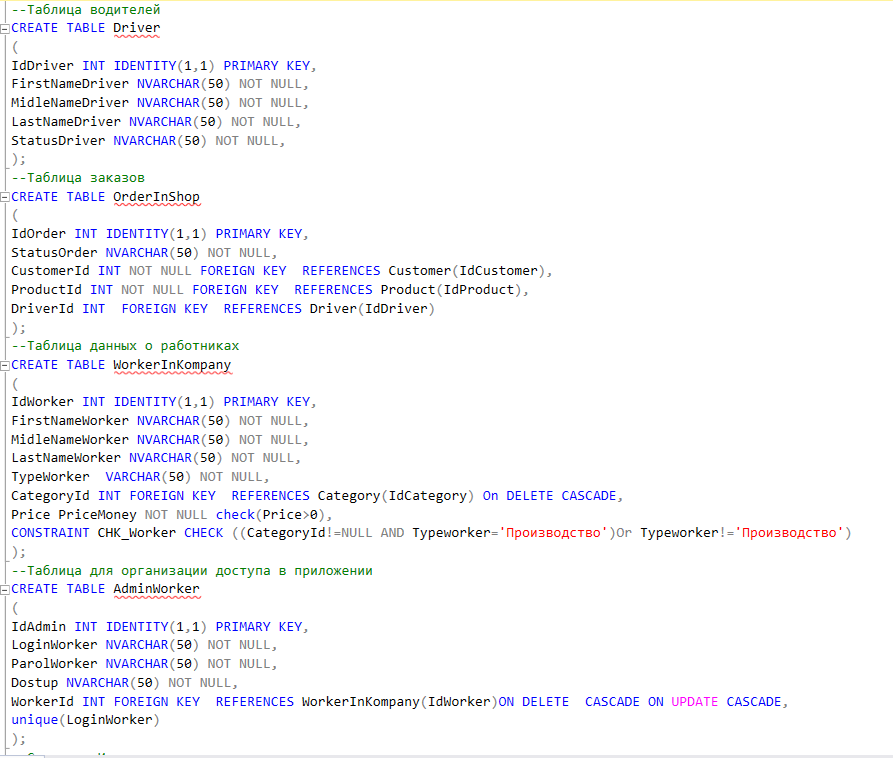


Код создания БД:

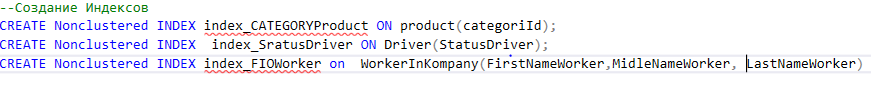


Код Создания таблиц:

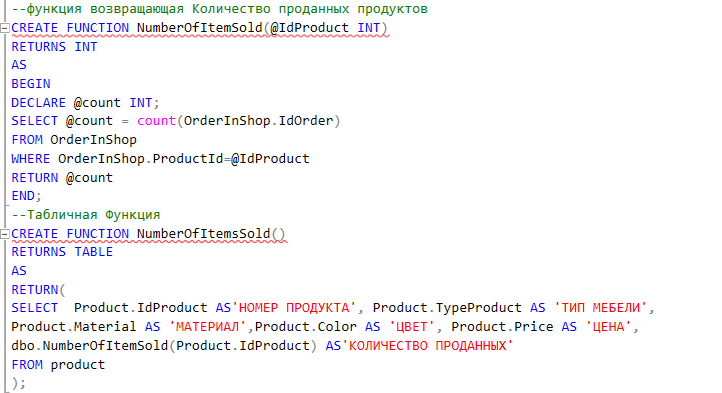


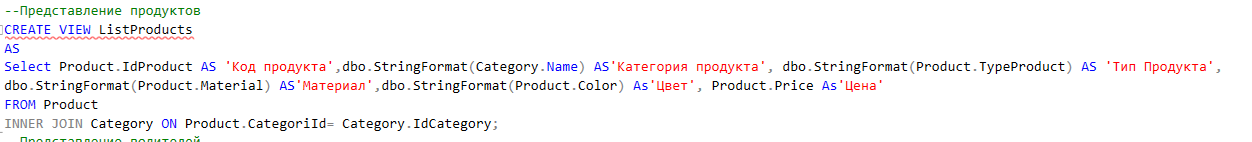


Создание Индексов:

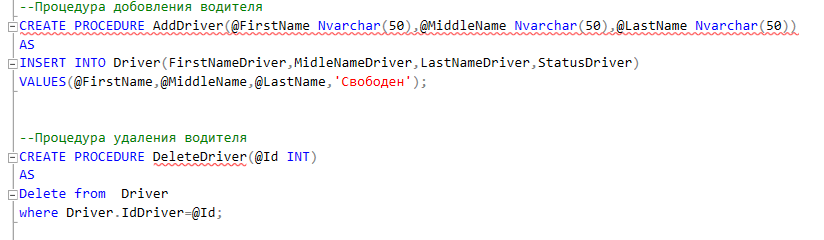


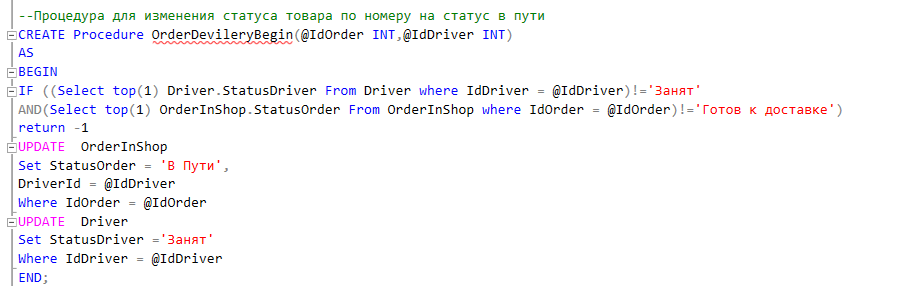
**6)Создать 1 табличную,1 скалярную функцию и 1 представление.**





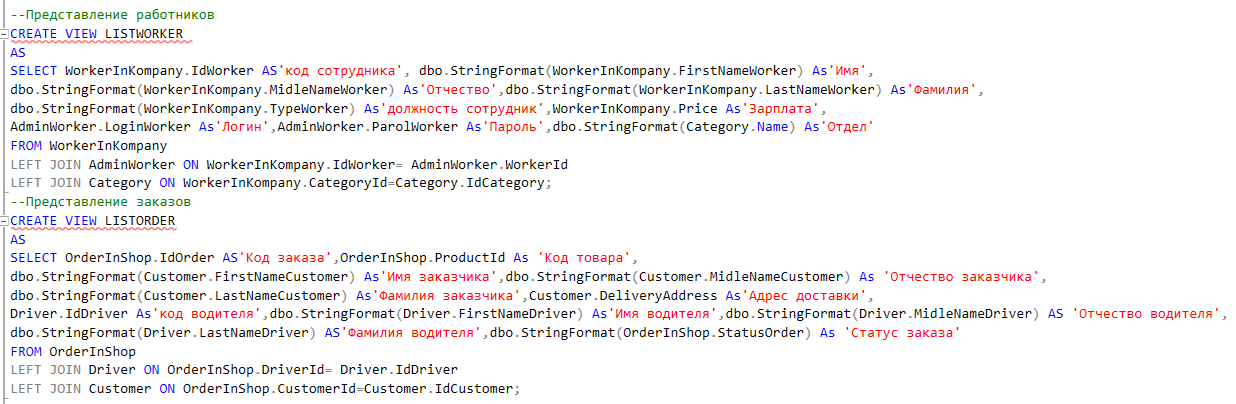
**7)Создать все необходимые процедуры для манипуляции данных с БД:**



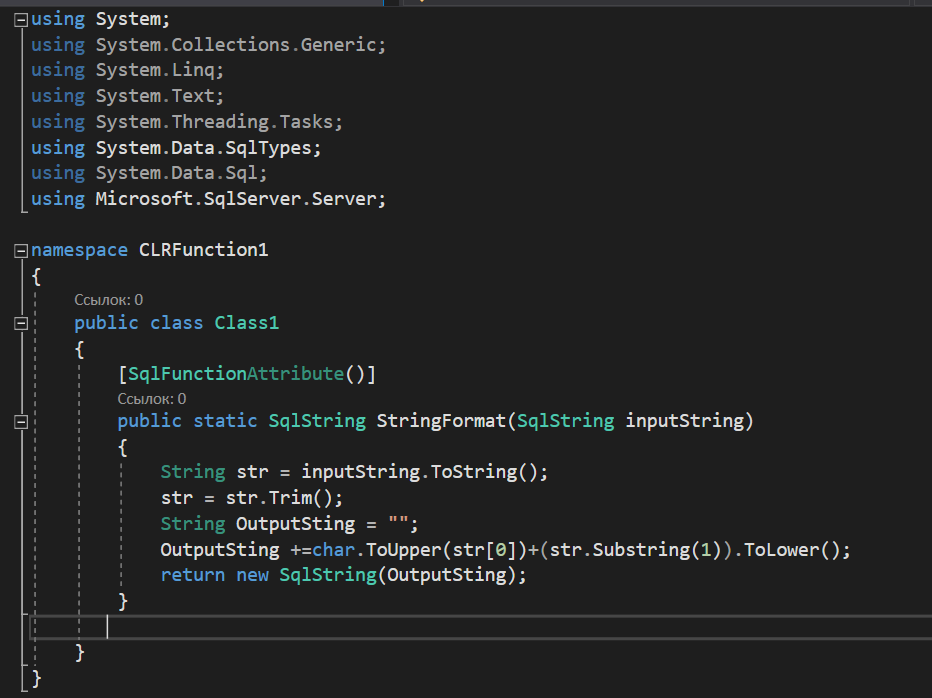


**8)Создать представления для визуализации данных в соответствии с пунктом 3.**

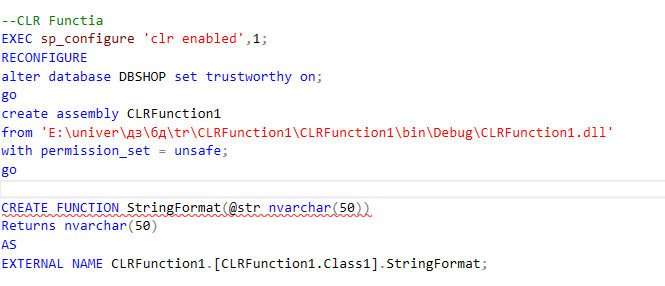




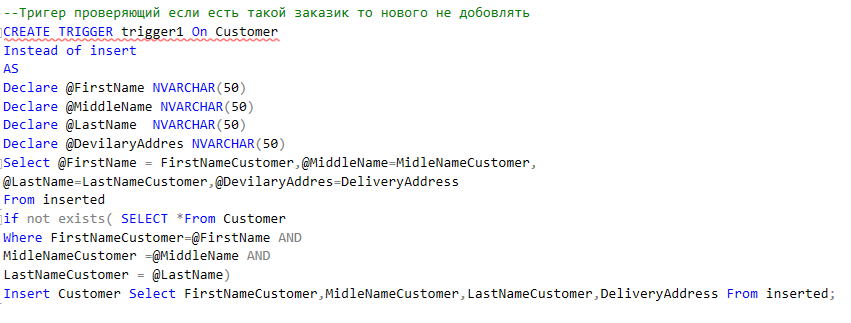
9)**Создать собственную CLR функцию.**



Подключение в БД

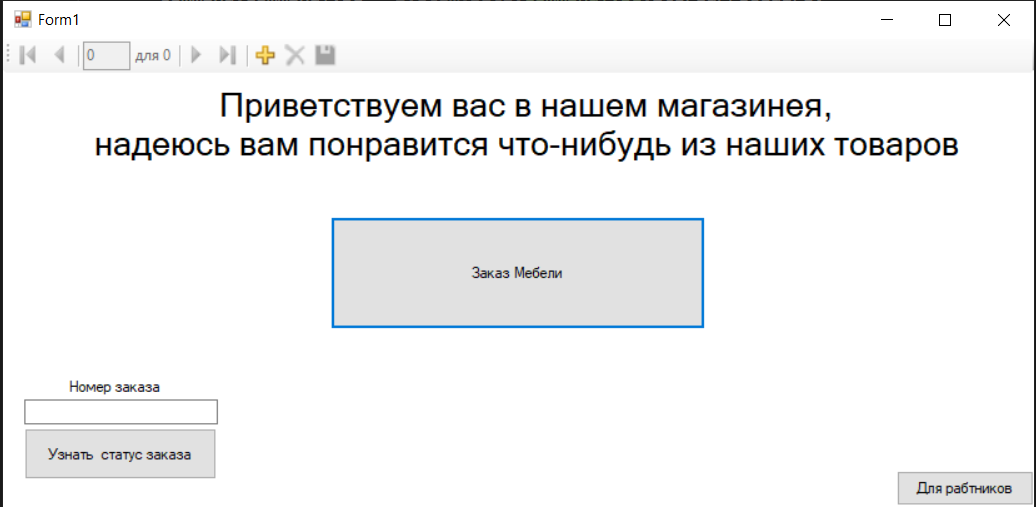


**10)Создать Хотя бы 1 триггер на таблицу.**

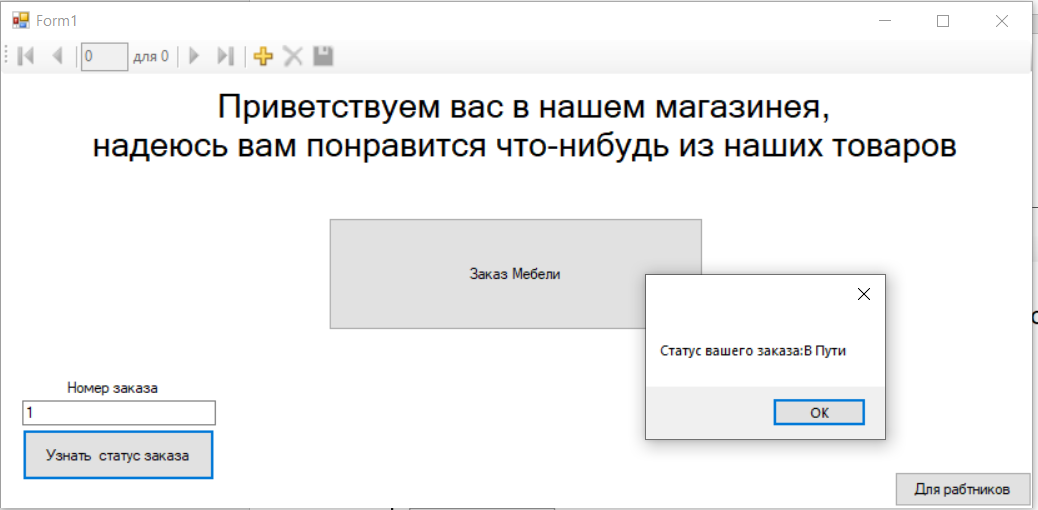


**Пример работы программы:**

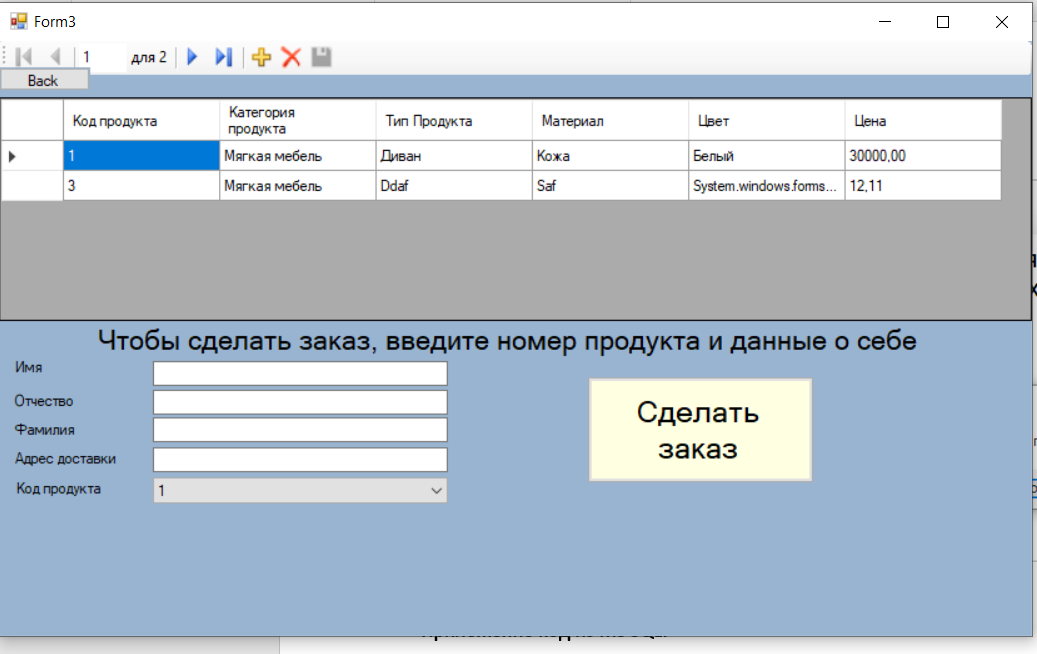
Главное меню программы:



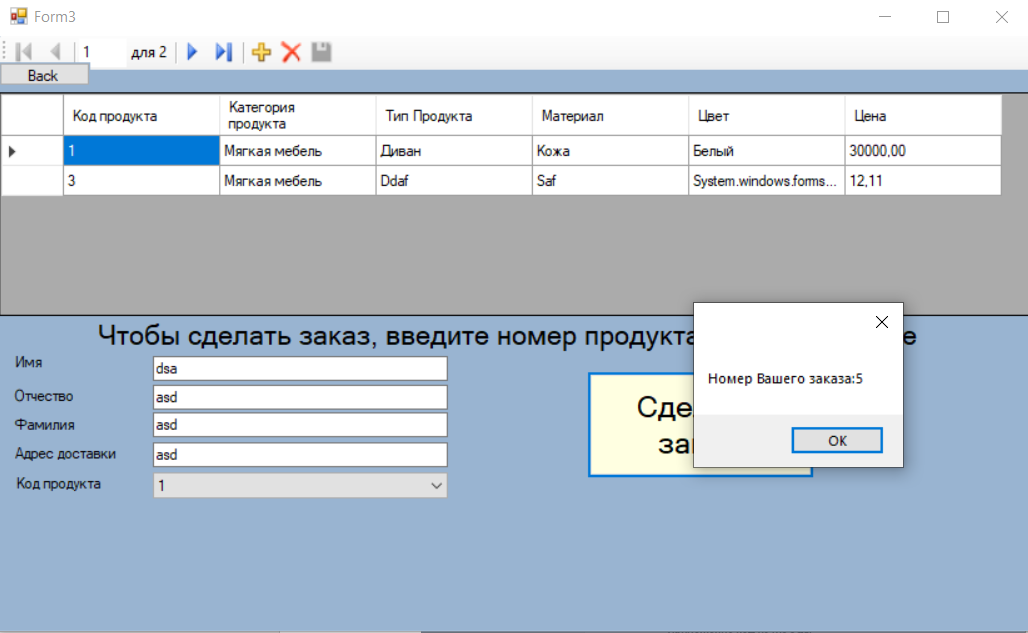
Ввод номера для узнавания статуса заказа:



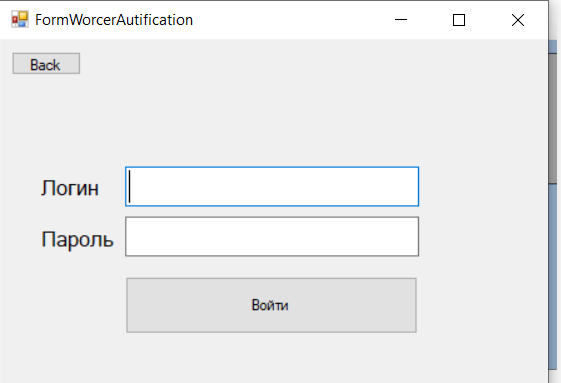
Меню для пользователя, заказывающего мебель:



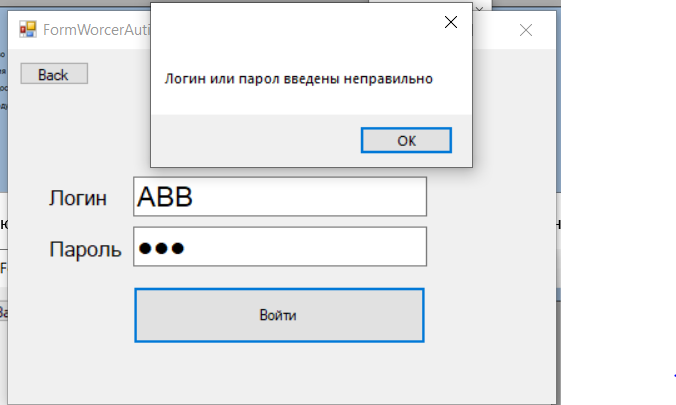
Когда пользователь совершает заказ:



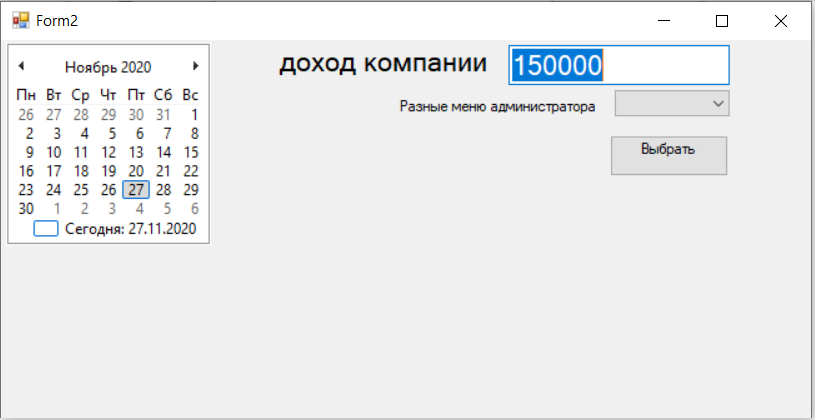
Меню ввода данных для входа в приложение для работников компании



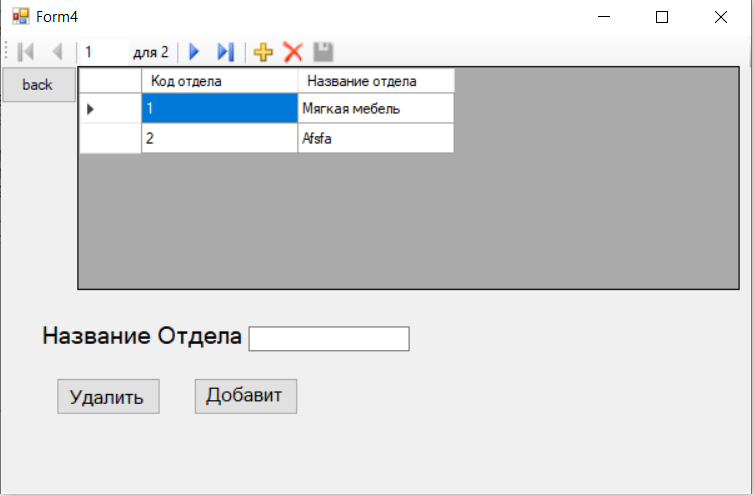
Когда работник ввел не верный логин или пароль.



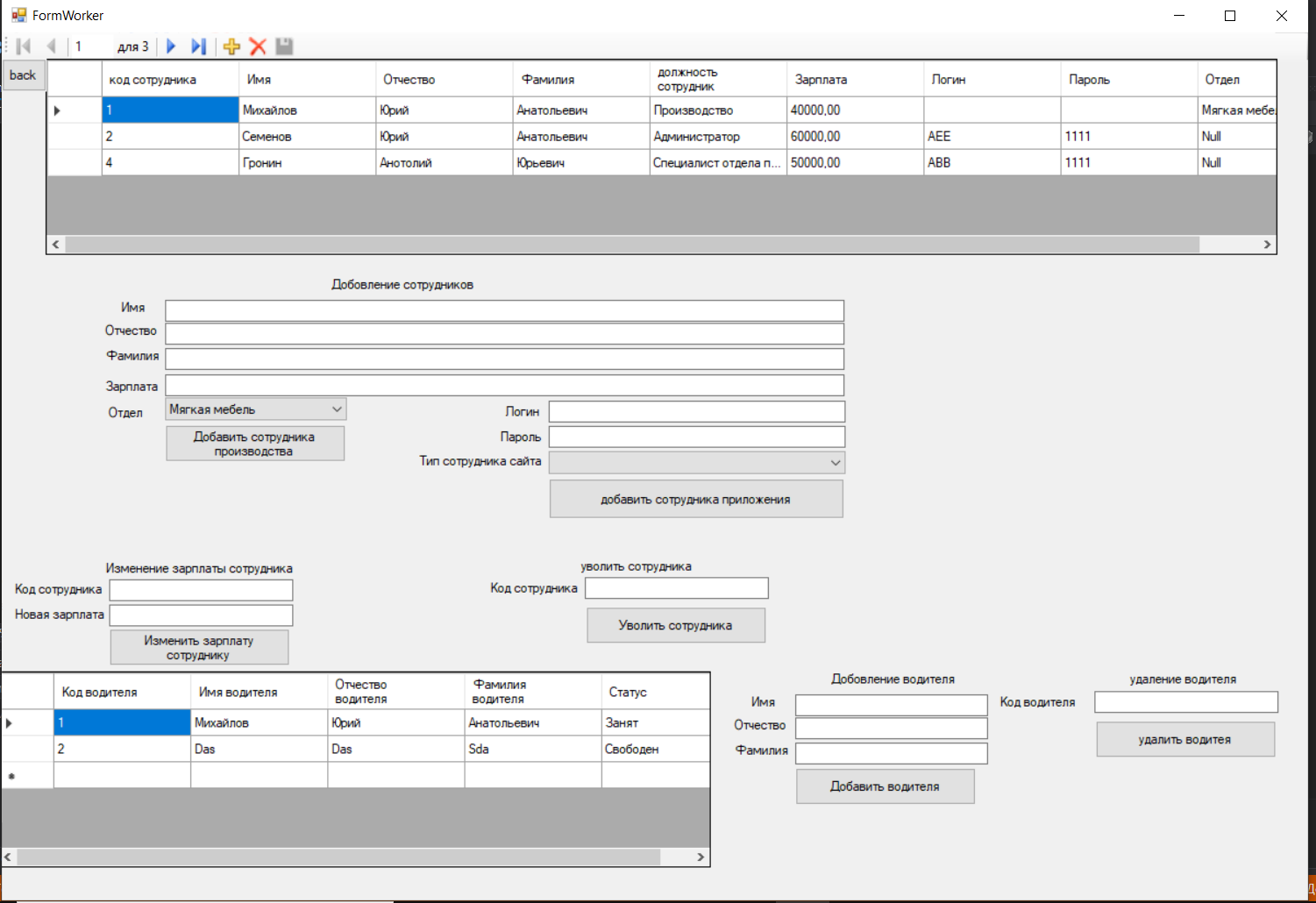
Главное меню администратора:



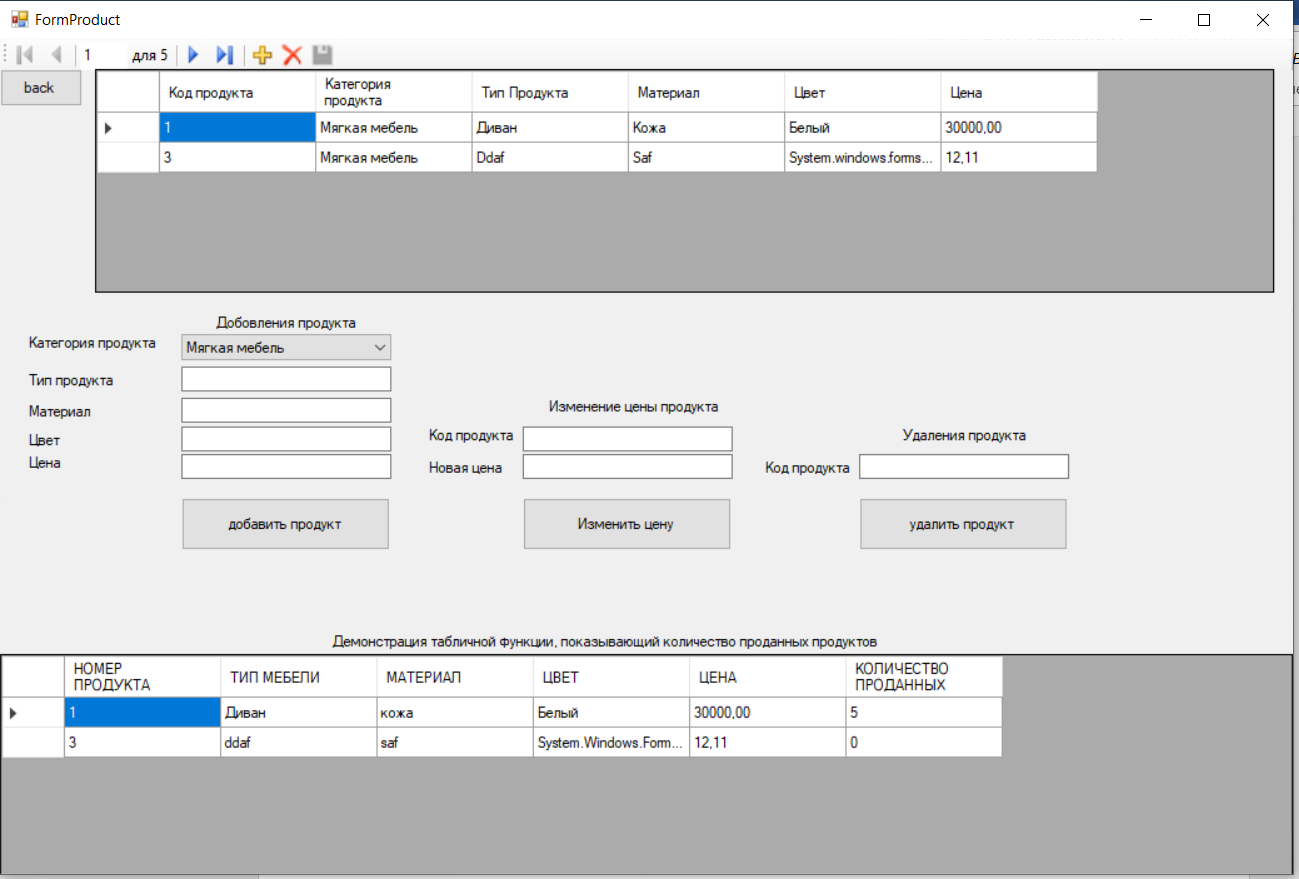
Меню изменения отделов у администратора:

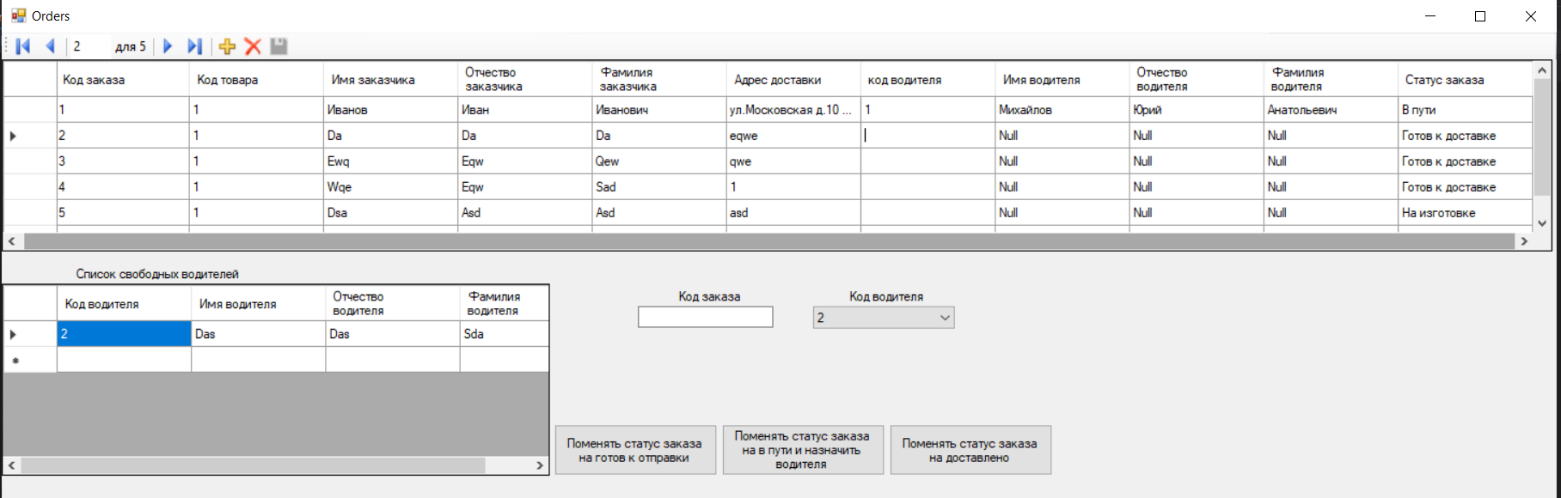


Меню работников у администратора



Меню продуктов у администратора:



Меню работника отдела приема и отправки товара

**Приложение код из MS SQL:**

--Создание БД  
CREATE DATABASE DBSHOP  
ON  
PRIMARY  
(  
NAME = SHOP\_PRIMARY,  
 FILENAME = 'E:\MSQL\DATA2\SHOP\_PRIMARY.mdf',  
 SIZE = 5MB,  
 MAXSIZE = 10MB,  
 FILEGROWTH = 20%  
 )  
 LOG ON  
 (  
 NAME = SHOP\_LOG,  
 FILENAME = 'E:\MSQL\DATA2\SHOP\_LOG.ldf',  
 SIZE = 2MB,  
 MAXSIZE = 5MB,  
 FILEGROWTH = 1MB  
 );  
 create type PriceMoney From decimal(10,2);  
 --Создание таблиц  
 CREATE TABLE Category  
 (  
 IdCategory  INT IDENTITY(1,1) PRIMARY KEY,  
 Name NVARCHAR(50) NOT NULL  
 );  
 --Таблица продуктов  
 CREATE TABLE Product  
 (  
 IdProduct  INT IDENTITY(1,1) PRIMARY KEY,  
 TypeProduct NVARCHAR(50) NOT NULL,  
 Color NVARCHAR(50) NOT NULL,  
 Material NVARCHAR(50) NOT NULL,  
 Price PriceMoney NOT NULL check(Price>0),  
 CategoriId INT NOT NULL FOREIGN KEY  REFERENCES Category(IdCategory) ON DELETE CASCADE   
 );  
 --Таблица продуктов покупателей  
 CREATE TABLE Customer  
(  
IdCustomer INT IDENTITY(1,1) PRIMARY KEY,  
FirstNameCustomer NVARCHAR(50) NOT NULL,  
MidleNameCustomer NVARCHAR(50) NOT NULL,  
LastNameCustomer NVARCHAR(50) NOT NULL,  
DeliveryAddress NVARCHAR(50) NOT NULL  
);  
--Таблица водителей  
CREATE TABLE Driver   
(  
IdDriver INT IDENTITY(1,1) PRIMARY KEY,  
FirstNameDriver NVARCHAR(50) NOT NULL,  
MidleNameDriver NVARCHAR(50) NOT NULL,  
LastNameDriver NVARCHAR(50) NOT NULL,  
StatusDriver NVARCHAR(50) NOT NULL,  
);  
--Таблица заказов  
CREATE TABLE OrderInShop  
(  
IdOrder INT IDENTITY(1,1) PRIMARY KEY,  
StatusOrder NVARCHAR(50) NOT NULL,  
CustomerId INT NOT NULL FOREIGN KEY  REFERENCES Customer(IdCustomer),  
ProductId INT NOT NULL FOREIGN KEY  REFERENCES Product(IdProduct),  
DriverId INT  FOREIGN KEY  REFERENCES Driver(IdDriver)  
);  
--Таблица данных о работниках  
CREATE TABLE WorkerInKompany  
(  
IdWorker INT IDENTITY(1,1) PRIMARY KEY,  
FirstNameWorker NVARCHAR(50) NOT NULL,  
MidleNameWorker NVARCHAR(50) NOT NULL,  
LastNameWorker NVARCHAR(50) NOT NULL,  
TypeWorker  VARCHAR(50) NOT NULL,  
CategoryId INT FOREIGN KEY  REFERENCES Category(IdCategory) On DELETE CASCADE,  
Price PriceMoney NOT NULL check(Price>0),  
CONSTRAINT CHK\_Worker CHECK ((CategoryId!=NULL AND Typeworker='Производство')Or Typeworker!='Производство')  
);  
--Таблица для организации доступа в приложении  
CREATE TABLE AdminWorker  
(  
IdAdmin INT IDENTITY(1,1) PRIMARY KEY,  
LoginWorker NVARCHAR(50) NOT NULL,  
ParolWorker NVARCHAR(50) NOT NULL,  
Dostup NVARCHAR(50) NOT NULL,  
WorkerId INT FOREIGN KEY  REFERENCES WorkerInKompany(IdWorker)ON DELETE  CASCADE ON UPDATE CASCADE,  
unique(LoginWorker)  
);  
--Создание Индексов  
CREATE Nonclustered INDEX index\_CATEGORYProduct ON product(categoriId);  
CREATE Nonclustered INDEX  index\_SratusDriver ON Driver(StatusDriver);  
CREATE Nonclustered INDEX index\_FIOWorker on  WorkerInKompany(FirstNameWorker,MidleNameWorker, LastNameWorker)  
   
--Создание процедур  
--Процедура добовления водителя  
CREATE PROCEDURE AddDriver(@FirstName Nvarchar(50),@MiddleName Nvarchar(50),@LastName Nvarchar(50))  
AS  
INSERT INTO Driver(FirstNameDriver,MidleNameDriver,LastNameDriver,StatusDriver)   
VALUES(@FirstName,@MiddleName,@LastName,'Свободен');  
   
   
--Процедура удаления водителя  
CREATE PROCEDURE DeleteDriver(@Id INT)  
AS  
Delete from  Driver   
where Driver.IdDriver=@Id;  
   
--Процедура добовления покупателя   
CREATE PROCEDURE AddCustomer(@FirstName Nvarchar(50),@MiddleName Nvarchar(50),@LastName Nvarchar(50),@Address Nvarchar(50))  
AS  
INSERT INTO Customer(FirstNameCustomer,MidleNameCustomer,LastNameCustomer,DeliveryAddress)   
VALUES(@FirstName,@MiddleName,@LastName,@Address);  
   
--Процедура добовления продукта  
CREATE PROCEDURE AddProduct  
(@TypeProduct NVARCHAR(50),@Color NVARCHAR(50),@Material NVARCHAR(50),@Price PriceMoney,@Categori NVARCHAR(50))  
AS  
begin  
DECLARE @CategoriId INT  
Select @CategoriId=  IdCategory from Category where Name = @Categori  
If(@CategoriId=NULL)return - 1  
INSERT INTO Product(TypeProduct,Color,Material,Price,CategoriId)  
VALUES(@TypeProduct,@Color,@Material,@Price,@CategoriId)  
end;  
   
--Процедура удаления продукта  
CREATE PROCEDURE DeleteProduct(@id INT )  
AS  
begin  
DELETE FROM Product   
WHERE product.IdProduct=@id  
end;  
   
--Процедура удобовления категории  
CREATE PROCEDURE AddCategory(@Name Nvarchar(50))  
AS  
INSERT INTO Category(Name)   
VALUES(@Name);  
   
--Процедура удаления категории  
CREATE PROCEDURE DeleteCategory(@Name Nvarchar(50))  
AS  
DELETE From Category WHERE Category.Name=@Name;  
--Процедура добовления продукта  
CREATE PROCEDURE AddWorkerInProduction  
(@FirstName Nvarchar(50),@MiddleName Nvarchar(50),@LastName Nvarchar(50),@Category Nvarchar(50),@Price PriceMoney)  
AS  
BEGIN  
DECLARE  @CategoriId INT  
Select @CategoriId=  IdCategory from Category where Name = @Category  
INSERT INTO WorkerInKompany(FirstNameWorker,MidleNameWorker,LastNameWorker,TypeWorker,CategoryId,Price)  
VALUES(@FirstName,@MiddleName,@LastName,'Производство',@CategoriId,@Price)  
END;  
   
   
-- Процедура удаления раотника  
CREATE PROCEDURE DeleteWorkerInProduction(@ID int)  
AS  
BEGIN  
DELETE FROM WorkerInKompany WHERE IdWorker=@ID;  
END;  
--Процедура добавления данных о работнике  
CREATE PROCEDURE AddWorkerOrAdmin --  
(@FirstName Nvarchar(50),@MiddleName Nvarchar(50),@LastName Nvarchar(50),  
@Price PriceMoney,@TypeWorker Nvarchar(50),@Login Nvarchar(50), @Parol Nvarchar(50))  
AS  
BEGIN  
INSERT INTO WorkerInKompany(FirstNameWorker,MidleNameWorker,LastNameWorker,TypeWorker,Price)  
VALUES(@FirstName,@MiddleName,@LastName,@TypeWorker,@Price)  
DECLARE @IdWorker INT  
SET  @IdWorker= SCOPE\_IDENTITY()  
INSERT INTO AdminWorker(LoginWorker,ParolWorker,WorkerId,Dostup)  
VALUES(@Login,@Parol,@IdWorker,@TypeWorker)  
END;  
   
   
  
--Добавление заказа  
CREATE PROCEDURE AddOrder  
(@FirstName Nvarchar(50),@MiddleName Nvarchar(50),@LastName Nvarchar(50),@Address Nvarchar(50),@IDProduct INT)  
AS  
BEGIN  
EXEC AddCustomer @FirstName,@MiddleName,@LastName,@Address  
DECLARE @IdCustomer INT  
SELECT @IdCustomer = Customer.IdCustomer From Customer  
Where FirstNameCustomer=@FirstName AND  
MidleNameCustomer =@MiddleName AND  
LastNameCustomer = @LastName  
INSERT INTO  OrderInShop(CustomerId,ProductId,StatusOrder)  
VALUES(@IdCustomer,@IDProduct,'На изготовке')  
DECLARE @IdOrder INT  
SET  @IdOrder= SCOPE\_IDENTITY()  
RETURN @IdOrder  
END;  
--Процедура для изменения статуса товара по номеру на статус изготовка  
CREATE PROCEDURE OrderAreManufactured(@Id INT)  
AS  
IF ((Select top(1) OrderInShop.StatusOrder From OrderInShop where IdOrder = @Id)!='На изготовке')  
return -1  
UPDATE  OrderInShop  
Set StatusOrder = 'Готов к доставке'  
Where IdOrder = @Id;  
   
--Процедура для изменения статуса товара по номеру на статус в пути  
CREATE Procedure OrderDevileryBegin(@IdOrder INT,@IdDriver INT)  
AS  
BEGIN  
IF ((Select top(1) Driver.StatusDriver From Driver where IdDriver = @IdDriver)!='Занят'  
AND(Select top(1) OrderInShop.StatusOrder From OrderInShop where IdOrder = @IdOrder)!='Готов к доставке')  
return -1  
UPDATE  OrderInShop  
Set StatusOrder = 'В Пути',  
DriverId = @IdDriver  
Where IdOrder = @IdOrder  
UPDATE  Driver  
Set StatusDriver ='Занят'  
Where IdDriver = @IdDriver  
END;  
--Процедура для измения статуса товара на доставлен  
CREATE Procedure OrderDevileryEnd(@IdOrder INT)  
AS  
BEGIN  
IF ((Select top(1) OrderInShop.StatusOrder From OrderInShop where IdOrder = @IdOrder)!='В Пути')  
return -1  
UPDATE  Driver  
Set StatusDriver ='Свободен'  
Where IdDriver = (Select OrderInShop.DriverId From OrderInShop  
Where IdOrder=@IdOrder)  
Update OrderInShop  
Set StatusOrder = 'Доставлен'  
Where IdOrder=@IdOrder  
END;  
--Процедура для изменения зарплаты работника  
CREATE PROCEDURE NewPriseWorker(@Id INT,@newPrice PriceMoney)  
AS  
BEGIN  
UPDATE WorkerInKompany  
Set Price = @newPrice  
Where IdWorker = @Id  
END;  
--Процедура для изменения цены продукта  
CREATE PROCEDURE NewPriceProduct(@Id INT,@newPrice PriceMoney)  
AS  
BEGIN  
UPDATE Product  
Set Price = @newPrice  
Where IdProduct = @Id  
END;  
   
--Функция возвращающая доступ по логину и поролю  
CREATE FUNCTION GetDostup(@Login NVARCHAR(50),@Parol NVARCHAR(50))  
RETURNS NVARCHAR(50)  
AS  
BEGIN  
DECLARE @DOSTUP NVARCHAR(50)  
Select top(1) @DOSTUP = Dostup From AdminWorker   
where LoginWorker= @Login AND ParolWorker =@Parol;  
RETURN @DOSTUP  
END;  
-- Скалярная функция возвращающая заработок компании  
CREATE FUNCTION IncomeCompany()  
RETURNS INT  
AS  
BEGIN  
DECLARE @sum INT;  
SELECT @sum = SUM(Product.Price) FROM OrderInShop   
INNER JOIN Product ON OrderInShop.ProductId = Product.IdProduct  
Group BY Product.IdProduct  
RETURN @sum  
END;  
--функция возвращающая Количество проданных продуктов  
CREATE FUNCTION NumberOfItemSold(@IdProduct INT)  
RETURNS INT  
AS  
BEGIN  
DECLARE @count INT;  
SELECT @count = count(OrderInShop.IdOrder)    
FROM OrderInShop  
WHERE OrderInShop.ProductId=@IdProduct  
RETURN @count  
END;  
--Табличная Функция  
CREATE FUNCTION NumberOfItemsSold()  
RETURNS TABLE  
AS  
RETURN(  
SELECT  Product.IdProduct AS'НОМЕР ПРОДУКТА', Product.TypeProduct AS 'ТИП МЕБЕЛИ',  
Product.Material AS 'МАТЕРИАЛ',Product.Color AS 'ЦВЕТ', Product.Price AS 'ЦЕНА',  
dbo.NumberOfItemSold(Product.IdProduct) AS'КОЛИЧЕСТВО ПРОДАННЫХ'  
FROM product  
);  
--Функция Возвращающая Статус по номеру заказа  
CREATE FUNCTION OrderCustomerStautus(@IdOrder INT)  
RETURNS Varchar(50)  
 AS  
BEGIN   
DECLARE @Name NVarchar(50);  
Select top(1) @Name = OrderInShop.StatusOrder From OrderInShop  
Where IdOrder=@IdOrder;  
RETURN @Name  
END;  
   
   
--Представление продуктов  
CREATE VIEW ListProducts  
AS   
Select Product.IdProduct AS 'Код продукта',dbo.StringFormat(Category.Name) AS'Категория продукта', dbo.StringFormat(Product.TypeProduct) AS 'Тип Продукта',  
dbo.StringFormat(Product.Material) AS'Материал',dbo.StringFormat(Product.Color) As'Цвет', Product.Price As'Цена'  
FROM Product  
INNER JOIN Category ON Product.CategoriId= Category.IdCategory;  
--Представление водителей  
CREATE VIEW ListDriver  
As  
Select IdDriver As'Код водителя',dbo.StringFormat(FirstNameDriver) As'Имя водителя',dbo.StringFormat(MidleNameDriver) AS 'Отчество водителя',  
dbo.StringFormat(LastNameDriver) AS'Фамилия водителя',dbo.StringFormat(StatusDriver) As Статус From Driver;  
--Представление Свободных водителей   
CREATE VIEW ListDriverOpen   
AS  
Select IdDriver As'Код водителя',dbo.StringFormat(FirstNameDriver) As'Имя водителя',dbo.StringFormat(MidleNameDriver) AS 'Отчество водителя',  
dbo.StringFormat(LastNameDriver) AS'Фамилия водителя' From Driver  
where StatusDriver ='Свободен';  
--Представление категорий  
CREATE VIEW ListCategory  
AS  
Select IdCategory As'Код отдела',dbo.StringFormat(category.Name) As'Название отдела' From Category;  
   
--Представление работников  
CREATE VIEW LISTWORKER   
AS  
SELECT WorkerInKompany.IdWorker AS'код сотрудника', dbo.StringFormat(WorkerInKompany.FirstNameWorker) As'Имя',  
dbo.StringFormat(WorkerInKompany.MidleNameWorker) As'Отчество',dbo.StringFormat(WorkerInKompany.LastNameWorker) As'Фамилия',  
dbo.StringFormat(WorkerInKompany.TypeWorker) As'должность сотрудник',WorkerInKompany.Price As'Зарплата',  
AdminWorker.LoginWorker As'Логин',AdminWorker.ParolWorker As'Пароль',dbo.StringFormat(Category.Name) As'Отдел'  
FROM WorkerInKompany  
LEFT JOIN AdminWorker ON WorkerInKompany.IdWorker= AdminWorker.WorkerId  
LEFT JOIN Category ON WorkerInKompany.CategoryId=Category.IdCategory;  
--Представление заказов  
CREATE VIEW LISTORDER  
AS  
SELECT OrderInShop.IdOrder AS'Код заказа',OrderInShop.ProductId As 'Код товара',  
dbo.StringFormat(Customer.FirstNameCustomer) As'Имя заказчика',dbo.StringFormat(Customer.MidleNameCustomer) As 'Отчество заказчика',  
dbo.StringFormat(Customer.LastNameCustomer) As'Фамилия заказчика',Customer.DeliveryAddress As'Адрес доставки',  
Driver.IdDriver As'код водителя',dbo.StringFormat(Driver.FirstNameDriver) As'Имя водителя',dbo.StringFormat(Driver.MidleNameDriver) AS 'Отчество водителя',  
dbo.StringFormat(Driver.LastNameDriver) AS'Фамилия водителя',dbo.StringFormat(OrderInShop.StatusOrder) As 'Статус заказа'  
FROM OrderInShop  
LEFT JOIN Driver ON OrderInShop.DriverId= Driver.IdDriver  
LEFT JOIN Customer ON OrderInShop.CustomerId=Customer.IdCustomer;  
   
   
--Тригер проверяющий если есть такой заказик то нового не добовлять  
CREATE TRIGGER trigger1 On Customer  
Instead of insert  
AS  
Declare @FirstName NVARCHAR(50)  
Declare @MiddleName NVARCHAR(50)  
Declare @LastName  NVARCHAR(50)  
Declare @DevilaryAddres NVARCHAR(50)  
Select @FirstName = FirstNameCustomer,@MiddleName=MidleNameCustomer,  
@LastName=LastNameCustomer,@DevilaryAddres=DeliveryAddress  
From inserted  
if not exists( SELECT \*From Customer  
Where FirstNameCustomer=@FirstName AND  
MidleNameCustomer =@MiddleName AND  
LastNameCustomer = @LastName)  
Insert Customer Select FirstNameCustomer,MidleNameCustomer,LastNameCustomer,DeliveryAddress From inserted;  
   
   
   
--CLR Functia  
EXEC sp\_configure 'clr enabled',1;  
RECONFIGURE  
alter database DBSHOP set trustworthy on;  
go  
create assembly CLRFunction1  
from 'E:\univer\дз\бд\tr\CLRFunction1\CLRFunction1\bin\Debug\CLRFunction1.dll'  
with permission\_set = unsafe;  
go  
   
CREATE FUNCTION StringFormat(@str nvarchar(50))  
Returns nvarchar(50)  
AS  
EXTERNAL NAME CLRFunction1.[CLRFunction1.Class1].StringFormat;

**Приложение код С#**

**Главное меню программы:**

using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Data.SqlClient;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Windows.Forms;  
   
namespace TRBD  
{//Форма главного меню  
    public partial class Form1 : Form  
    {  
        static String ConnectString = "Data Source=DESKTOP-900BQME;Initial Catalog=DBSHOP;Integrated Security=True";  
        public Form1()  
        {  
            InitializeComponent();  
        }  
   
        private void Form1\_Load(object sender, EventArgs e)  
        {   
        }  
   
        private void label1\_Click(object sender, EventArgs e)  
        {  
        }  
        //Переход в меню покуупки товара  
        private void userMenu\_Click(object sender, EventArgs e)  
        {  
            FormCustomer newForm = new FormCustomer();  
            newForm.Show();  
            Hide();  
        }  
        //Кнопка для перехода в меню входа для работников  
        private void CustomerButton\_Click(object sender, EventArgs e)  
        {  
            FormWorcerAutification newForm = new FormWorcerAutification();  
            newForm.Show();  
            Hide();  
        }  
        //Кнопка проверки статуса товара  
        private void StatusByIdOrder\_Click(object sender, EventArgs e)  
        {  
            string sqlExpression = "OrderCustomerStautus";  
            try  
            {  
                using (SqlConnection connection = new SqlConnection(ConnectString))  
                {  
                    connection.Open();  
                    SqlCommand command = new SqlCommand(sqlExpression, connection);  
                    command.CommandType = System.Data.CommandType.StoredProcedure;  
                    SqlParameter nameParam1 = new SqlParameter  
                    {  
                        ParameterName = "@IdOrder",  
                        Value = int.Parse(IdOrder.Text)  
                    };  
                    command.Parameters.Add(nameParam1);  
                    var returnValue = command.Parameters.Add("@Return", SqlDbType.Int);  
                    returnValue.Direction = ParameterDirection.ReturnValue;  
                    command.ExecuteNonQuery();  
                    var result = (returnValue.Value).ToString();  
                    if(result=="")  
                    MessageBox.Show($"Заказа под номером :{IdOrder.Text} нет");  
                    else  
                    MessageBox.Show($"Статус вашего заказа:{result}");  
                    connection.Close();  
                }  
            }  
            catch (SqlException ex)  
            {  
                MessageBox.Show(ex.Message);  
            }  
        }  
    }  
}

**Меню для пользователя, заказывающего мебель:**

//Форма для заказа продукта из магазина  
    public partial class FormCustomer : Form  
    {  
        static String ConnectString = "Data Source=DESKTOP-900BQME;Initial Catalog=DBSHOP;Integrated Security=True";  
   
   
        public FormCustomer()  
        {  
            InitializeComponent();  
        }  
        //Процедура добавления заказа  
        private void AddOrder(string FirstName, string MiddleName, string LastName, string Addres, string IdProduct)  
        {  
            string sqlExpression = "AddOrder";  
            try  
            {  
                using (SqlConnection connection = new SqlConnection(ConnectString))  
                {  
                    connection.Open();  
   
                    SqlCommand command = new SqlCommand(sqlExpression,connection);  
                    command.CommandType = System.Data.CommandType.StoredProcedure;  
                    SqlParameter nameParam1 = new SqlParameter  
                    {  
                        ParameterName = "@FirstName",  
                        Value = FirstName  
                    };  
                    SqlParameter nameParam2 = new SqlParameter  
                    {  
                        ParameterName = "@MiddleName",  
                        Value = MiddleName  
                    };  
                    SqlParameter nameParam3 = new SqlParameter  
                    {  
                        ParameterName = "@LastName",  
                        Value = LastName  
                    };  
                    SqlParameter nameParam4 = new SqlParameter  
                    {  
                        ParameterName = "@Address",  
                        Value = Addres  
                    };  
                    SqlParameter nameParam5 = new SqlParameter  
                    {  
                        ParameterName = "@IDProduct",  
                        Value = int.Parse(IdProduct)  
                    };  
                    command.Parameters.Add(nameParam1);  
                    command.Parameters.Add(nameParam2);  
                    command.Parameters.Add(nameParam3);  
                    command.Parameters.Add(nameParam4);  
                    command.Parameters.Add(nameParam5);  
                    var returnValue = command.Parameters.Add("@Return", SqlDbType.Int);  
                    returnValue.Direction = ParameterDirection.ReturnValue;  
                    command.ExecuteNonQuery();  
                    var result = (int)returnValue.Value;  
                    MessageBox.Show($"Номер Вашего заказа:{result}");  
                    connection.Close();  
                }  
            }  
            catch (SqlException ex)  
            {  
                MessageBox.Show(ex.Message);  
            }  
        }  
   
   
        private void Form3\_Load(object sender, EventArgs e)  
        {  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.ListProducts". При необходимости она может быть перемещена или удалена.  
            this.listProductsTableAdapter.Fill(this.dBSHOPDataSet1.ListProducts);  
   
        }  
   
        //Кнопка добовления заказа  
        private void MakeOrder\_Click(object sender, EventArgs e)  
        {  
            AddOrder(NameCustomer.Text, MiddleNameCustomer.Text,  
                LastNameCustomer.Text, AddresCustomer.Text, CodProdut.Text);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.ListProducts". При необходимости она может быть перемещена или удалена.  
            this.listProductsTableAdapter.Fill(this.dBSHOPDataSet1.ListProducts);  
        }  
        private void label1\_Click(object sender, EventArgs e)  
        {  
        }  
   
        private void NameCustomer\_TextChanged(object sender, EventArgs e)  
        {  
        }  
   
        private void label2\_Click(object sender, EventArgs e)  
        {  
        }  
   
        private void Back\_Click(object sender, EventArgs e)  
        {  
   
                Form1 newForm = new Form1();  
                newForm.Show();  
                Hide();  
   
        }  
    }

**Меню ввода данных для входа в приложение для работников компании**

public partial class FormWorcerAutification : Form  
    {  
        static String ConnectString = "Data Source=DESKTOP-900BQME;Initial Catalog=DBSHOP;Integrated Security=True";  
   
        public FormWorcerAutification()  
        {  
   
            InitializeComponent();  
        }  
   
        private void Parol\_TextChanged(object sender, EventArgs e)  
        {  
            Parol.UseSystemPasswordChar = true;//делаем тест зашифрованным  
        }  
   
        private void Login\_TextChanged(object sender, EventArgs e)  
        {  
   
        }  
   
        private string AutificationWorcker(String Login,String Parol)  
        {  
            string sqlExpression = "GetDostup";  
            try  
            {  
                using (SqlConnection connection = new SqlConnection(ConnectString))  
                {  
                    connection.Open();  
                    SqlCommand command = new SqlCommand(sqlExpression, connection);  
                    command.CommandType = System.Data.CommandType.StoredProcedure;  
                    SqlParameter nameParam1 = new SqlParameter  
                    {  
                        ParameterName = "@Login",  
                        Value = Login  
                    };  
                    SqlParameter nameParam2 = new SqlParameter  
                    {  
                        ParameterName = "@Parol",  
                        Value = Parol  
                    };  
                    command.Parameters.Add(nameParam1);  
                    command.Parameters.Add(nameParam2);  
                    var returnValue = command.Parameters.Add("@Return", SqlDbType.NVarChar);  
                    returnValue.Direction = ParameterDirection.ReturnValue;  
                    command.ExecuteNonQuery();  
                    var result = (returnValue.Value).ToString();  
                    connection.Close();  
                    return result;  
                }  
            }  
            catch (SqlException ex)  
            {  
                MessageBox.Show(ex.Message);  
            }  
            return "";  
        }  
   
        private void buttonAutification\_Click(object sender, EventArgs e)  
        {  
            String dostup = AutificationWorcker(Login.Text, Parol.Text);  
            if (dostup == "")  
            {  
                MessageBox.Show("Логин или парол введены неправильно");  
            }else if(dostup== "Администратор")  
            {  
                FormAdministrator newForm = new FormAdministrator();  
                newForm.Show();  
                Hide();  
            }  
            else if (dostup == "Специалист отдела приема и отправки")  
            {  
                FormOrders newForm = new FormOrders();  
                newForm.Show();  
                Hide();  
            }  
        }  
   
        private void FormWorcerAutification\_Load(object sender, EventArgs e)  
        {  
   
        }  
   
        private void Back\_Click(object sender, EventArgs e)  
        {  
            Form1 newForm = new Form1();  
            newForm.Show();  
            Hide();  
        }  
    }

**Главное меню администратора:**

//Главное меню у администратора  
    public partial class FormAdministrator : Form  
    {  
        static String ConnectString = "Data Source=DESKTOP-900BQME;Initial Catalog=DBSHOP;Integrated Security=True";  
        public FormAdministrator()  
        {  
            InitializeComponent();  
        }  
   
        private void label2\_Click(object sender, EventArgs e)  
        {  
   
        }  
        //Функция возвращающая доход компании  
        private int GetCompanyProfit()  
        {  
            string sqlExpression = "IncomeCompany";  
            try  
            {  
                using (SqlConnection connection = new SqlConnection(ConnectString))  
                {  
                    connection.Open();  
                    SqlCommand command = new SqlCommand(sqlExpression, connection);  
                    command.CommandType = System.Data.CommandType.StoredProcedure;  
                    var returnValue = command.Parameters.Add("@Return", SqlDbType.Int);  
                    returnValue.Direction = ParameterDirection.ReturnValue;  
                    command.ExecuteNonQuery();  
                    var result = (int)returnValue.Value;  
                    connection.Close();  
                    return result;  
                }  
            }  
            catch (SqlException ex)  
            {  
                MessageBox.Show(ex.Message);  
            }  
            return 0;  
        }  
   
        private void FormAdministrator\_Load(object sender, EventArgs e)  
        {  
            textBox1.Text = GetCompanyProfit().ToString();  
        }  
   
        private void textBox1\_TextChanged(object sender, EventArgs e)  
        {  
            textBox1.Text = GetCompanyProfit().ToString();  
        }  
   
        private void buttonChangemode\_Click(object sender, EventArgs e)  
        {  
            if(comboBox2.Text == "Отделы")  
            {  
                FormCategory newForm = new FormCategory();  
                newForm.Show();  
                Hide();  
            }else if(comboBox2.Text == "Продукты")  
            {  
                FormProduct newForm = new FormProduct();  
                newForm.Show();  
                Hide();  
            }  
            else if (comboBox2.Text == "Работники")  
            {  
                FormWorker newForm = new FormWorker();  
                newForm.Show();  
                Hide();  
            }  
        }  
    }

**Меню изменения отделов у администратора:**

public partial class FormCategory : Form  
    {  
        static String ConnectString = "Data Source=DESKTOP-900BQME;Initial Catalog=DBSHOP;Integrated Security=True";  
        public FormCategory()  
        {  
            InitializeComponent();  
        }  
   
        private void FormCategory\_Load(object sender, EventArgs e)  
        {  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.ListCategory". При необходимости она может быть перемещена или удалена.  
            this.listCategoryTableAdapter.Fill(this.dBSHOPDataSet1.ListCategory);  
   
        }  
   
        private void buttonDeleteCategory\_Click(object sender, EventArgs e)  
        {  
            string sqlExpression = "DeleteCategory";  
            try  
            {  
                using (SqlConnection connection = new SqlConnection(ConnectString))  
                {  
                    connection.Open();  
                    SqlCommand command = new SqlCommand(sqlExpression, connection);  
                    command.CommandType = System.Data.CommandType.StoredProcedure;  
                    SqlParameter nameParam1 = new SqlParameter  
                    {  
                        ParameterName = "@Name",  
                        Value = textBox1.Text  
                    };  
                    command.Parameters.Add(nameParam1);  
                    command.ExecuteNonQuery();  
   
                }  
            }  
            catch (SqlException ex)  
            {  
                MessageBox.Show(ex.Message);  
            }  
            this.listCategoryTableAdapter.Fill(this.dBSHOPDataSet1.ListCategory);  
        }  
   
        private void buttonAddCategory\_Click(object sender, EventArgs e)  
        {  
            string sqlExpression = "AddCategory";  
            try  
            {  
                using (SqlConnection connection = new SqlConnection(ConnectString))  
                {  
                    connection.Open();  
                    SqlCommand command = new SqlCommand(sqlExpression, connection);  
                    command.CommandType = System.Data.CommandType.StoredProcedure;  
                    SqlParameter nameParam1 = new SqlParameter  
                    {  
                        ParameterName = "@Name",  
                        Value = textBox1.Text  
                    };  
                    command.Parameters.Add(nameParam1);  
                    command.ExecuteNonQuery();  
   
   
                    connection.Close();  
                }  
            }  
            catch (SqlException ex)  
            {  
                MessageBox.Show(ex.Message);  
            }  
            this.listCategoryTableAdapter.Fill(this.dBSHOPDataSet1.ListCategory);  
        }  
   
        private void back\_Click(object sender, EventArgs e)  
        {  
            FormAdministrator newForm = new FormAdministrator();  
            newForm.Show();  
            Hide();  
        }  
    }

**Меню работников у администратора**

public partial class FormWorker : Form  
    {  
        static String ConnectString = "Data Source=DESKTOP-900BQME;Initial Catalog=DBSHOP;Integrated Security=True";  
   
        public FormWorker()  
        {  
            InitializeComponent();  
        }  
   
        private void FormWorker\_Load(object sender, EventArgs e)  
        {  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet2.ListDriver". При необходимости она может быть перемещена или удалена.  
            this.listDriverTableAdapter1.Fill(this.dBSHOPDataSet2.ListDriver);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.ListDriver". При необходимости она может быть перемещена или удалена.  
            this.listDriverTableAdapter.Fill(this.dBSHOPDataSet1.ListDriver);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.ListCategory". При необходимости она может быть перемещена или удалена.  
            this.listCategoryTableAdapter.Fill(this.dBSHOPDataSet1.ListCategory);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.LISTWORKER". При необходимости она может быть перемещена или удалена.  
            this.lISTWORKERTableAdapter.Fill(this.dBSHOPDataSet1.LISTWORKER);  
   
        }  
   
        private void label2\_Click(object sender, EventArgs e)  
        {  
   
        }  
   
        private void label19\_Click(object sender, EventArgs e)  
        {  
   
        }  
   
        private void back\_Click(object sender, EventArgs e)  
        {  
            FormAdministrator newForm = new FormAdministrator();  
            newForm.Show();  
            Hide();  
        }  
   
        private void AddWorkerProizvod(String FirsName, String MiddleName, String LastName, String Price, String Category)  
        {  
            try  
            {  
                using (SqlConnection connection = new SqlConnection(ConnectString))  
                {  
                    connection.Open();  
                    string commandtext = $"EXEC AddWorkerInProduction '{FirsName}','{MiddleName}','{LastName}','{Category}',{Price}";  
                    SqlCommand command = new SqlCommand(commandtext, connection);  
                    command.ExecuteReader();  
                    connection.Close();  
                }  
            }  
            catch (SqlException ex)  
            {  
                MessageBox.Show(ex.Message);  
            }  
        }  
            private void DeleteWorker(String Id)  
            {  
                try  
                {  
                    using (SqlConnection connection = new SqlConnection(ConnectString))  
                    {  
                        connection.Open();  
                        string commandtext = $"EXEC DeleteWorkerInProduction {int.Parse(Id)}";  
                        SqlCommand command = new SqlCommand(commandtext, connection);  
                        command.ExecuteReader();  
                        connection.Close();  
                    }  
                }  
                catch (SqlException ex)  
                {  
                    MessageBox.Show(ex.Message);  
                }  
   
            }  
   
        private void ChangePriWorker(string Id, string priseString)  
        {  
            string sqlExpression = "NewPriseWorker";  
            try  
            {  
                using (SqlConnection connection = new SqlConnection(ConnectString))  
                {  
                    var price = Decimal.Parse(priseString);  
                    connection.Open();  
                    SqlCommand command = new SqlCommand(sqlExpression, connection);  
                    command.CommandType = System.Data.CommandType.StoredProcedure;  
                    SqlParameter nameParam1 = new SqlParameter  
                    {  
                        ParameterName = "@Id",  
                        Value = int.Parse(Id)  
                    };  
                    SqlParameter nameParam2 = new SqlParameter  
                    {  
                        ParameterName = "@newPrice",  
                        Value = price  
                    };  
                    command.Parameters.Add(nameParam1);  
                    command.Parameters.Add(nameParam2);  
                    command.ExecuteNonQuery();  
                    connection.Close();  
                }  
            }  
            catch (SqlException ex)  
            {  
                MessageBox.Show(ex.Message);  
            }  
        }  
   
        private void AddWorkerOrAdmin(String FirsName, String MiddleName, String LastName, String Price,String Type, String login,String parol)  
        {  
            try  
            {  
                using (SqlConnection connection = new SqlConnection(ConnectString))  
                {  
                    connection.Open();  
                    string commandtext = $"EXEC AddWorkerOrAdmin '{FirsName}','{MiddleName}','{LastName}',{Price},'{Type}'" +  
                        $",'{login}','{Parol}'";  
                    SqlCommand command = new SqlCommand(commandtext, connection);  
                    command.ExecuteReader();  
                    connection.Close();  
                }  
            }  
            catch (SqlException ex)  
            {  
                MessageBox.Show(ex.Message);  
            }  
        }  
        private void AddDriver(String FirsName, String MiddleName, String LastName)  
        {  
            try  
            {  
                using (SqlConnection connection = new SqlConnection(ConnectString))  
                {  
                    connection.Open();  
                    string commandtext = $"EXEC AddDriver '{FirsName}','{MiddleName}','{LastName}'";  
                    SqlCommand command = new SqlCommand(commandtext, connection);  
                    command.ExecuteReader();  
                    connection.Close();  
                }  
            }  
            catch (SqlException ex)  
            {  
                MessageBox.Show(ex.Message);  
            }  
        }  
        private void DeleteDriver(String Id)  
        {  
            try  
            {  
                using (SqlConnection connection = new SqlConnection(ConnectString))  
                {  
                    connection.Open();  
                    string commandtext = $"EXEC DeleteDriver {int.Parse(Id)}";  
                    SqlCommand command = new SqlCommand(commandtext, connection);  
                    command.ExecuteReader();  
                    connection.Close();  
                }  
            }  
            catch (SqlException ex)  
            {  
                MessageBox.Show(ex.Message);  
            }  
   
        }  
   
        private void buttonAddWorkerProizvod\_Click(object sender, EventArgs e)  
        {  
            AddWorkerProizvod(FirstNameWorker.Text, MiddleNameWorker.Text, LastNameWorker.Text,  
               Price.Text, categoryWorker.Text);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.LISTWORKER". При необходимости она может быть перемещена или удалена.  
            this.lISTWORKERTableAdapter.Fill(this.dBSHOPDataSet1.LISTWORKER);  
   
        }  
   
        private void buttonChangePice\_Click(object sender, EventArgs e)  
        {  
            ChangePriWorker(CodeWorkerChangePrice.Text, NewPrice.Text);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.LISTWORKER". При необходимости она может быть перемещена или удалена.  
            this.lISTWORKERTableAdapter.Fill(this.dBSHOPDataSet1.LISTWORKER);  
        }  
   
        private void AddWorkerAdmin\_Click(object sender, EventArgs e)  
        {  
            AddWorkerOrAdmin(FirstNameWorker.Text, MiddleNameWorker.Text, LastNameWorker.Text,  
               Price.Text, typeWorker.Text, Login.Text, Parol.Text);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.LISTWORKER". При необходимости она может быть перемещена или удалена.  
            this.lISTWORKERTableAdapter.Fill(this.dBSHOPDataSet1.LISTWORKER);  
        }  
   
        private void buttonDeleteWorker\_Click(object sender, EventArgs e)  
        {  
            DeleteWorker(CodeWorkerDelete.Text);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.LISTWORKER". При необходимости она может быть перемещена или удалена.  
            this.lISTWORKERTableAdapter.Fill(this.dBSHOPDataSet1.LISTWORKER);  
   
        }  
   
        private void buttonAddDriver\_Click(object sender, EventArgs e)  
        {  
            AddDriver(FirstNameDriver.Text, MiddleNameDriver.Text, LastNameDriver.Text);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet2.ListDriver". При необходимости она может быть перемещена или удалена.  
            this.listDriverTableAdapter1.Fill(this.dBSHOPDataSet2.ListDriver);  
   
        }  
   
        private void buttonDeleteDriver\_Click(object sender, EventArgs e)  
        {  
            DeleteDriver(CodeDeleteDriver.Text);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet2.ListDriver". При необходимости она может быть перемещена или удалена.  
            this.listDriverTableAdapter1.Fill(this.dBSHOPDataSet2.ListDriver);  
        }  
    }

**Меню продуктов у администратора:**

//Форма продуктов у администратора  
    public partial class FormProduct : Form  
    {  
        static String ConnectString = "Data Source=DESKTOP-900BQME;Initial Catalog=DBSHOP;Integrated Security=True";  
        public FormProduct()  
        {  
            InitializeComponent();  
        }  
   
        private void FormProduct\_Load(object sender, EventArgs e)  
        {  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet2.NumberOfItemsSold". При необходимости она может быть перемещена или удалена.  
            this.numberOfItemsSoldTableAdapter.Fill(this.dBSHOPDataSet2.NumberOfItemsSold);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.ListCategory". При необходимости она может быть перемещена или удалена.  
            this.listCategoryTableAdapter.Fill(this.dBSHOPDataSet1.ListCategory);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.ListProducts". При необходимости она может быть перемещена или удалена.  
            this.listProductsTableAdapter.Fill(this.dBSHOPDataSet1.ListProducts);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.LISTORDER". При необходимости она может быть перемещена или удалена.  
            this.lISTORDERTableAdapter.Fill(this.dBSHOPDataSet1.LISTORDER);  
   
        }  
        private void AddOproduct(string Category, string type, string material, string color, string priseString)  
        {  
            try  
            {  
                using (SqlConnection connection = new SqlConnection(ConnectString))  
                {  
                    connection.Open();  
                    string commandtext = $"EXEC AddProduct '{type}','{Color}','{material}',{priseString},'{Category}'";  
                    SqlCommand command = new SqlCommand(commandtext, connection);   
                    command.ExecuteReader();  
                    connection.Close();  
                }  
            }  
            catch (SqlException ex)  
            {  
                MessageBox.Show(ex.Message);  
            }  
        }  
        private void Deleteproduct(string Id)  
        {  
            string sqlExpression = "DeleteProduct";  
            try  
            {  
                using (SqlConnection connection = new SqlConnection(ConnectString))  
                {  
   
                    connection.Open();  
   
                    SqlCommand command = new SqlCommand(sqlExpression, connection);  
                    command.CommandType = System.Data.CommandType.StoredProcedure;  
                    SqlParameter nameParam1 = new SqlParameter  
                    {  
                        ParameterName = "@Id",  
                        Value = int.Parse(Id)  
                    };  
                    command.Parameters.Add(nameParam1);  
                    command.ExecuteNonQuery();  
                }  
            }  
            catch (SqlException ex)  
            {  
                MessageBox.Show(ex.Message);  
            }  
        }  
        private void ChangePriceproduct(string Id, string priseString)  
        {  
            string sqlExpression = "NewPriceProduct";  
            try  
            {  
                using (SqlConnection connection = new SqlConnection(ConnectString))  
                {  
                    var price = Decimal.Parse(priseString);  
                    connection.Open();  
                    SqlCommand command = new SqlCommand(sqlExpression, connection);  
                    command.CommandType = System.Data.CommandType.StoredProcedure;  
                    SqlParameter nameParam1 = new SqlParameter  
                    {  
                        ParameterName = "@Id",  
                        Value = int.Parse(Id)  
                    };  
                    SqlParameter nameParam2 = new SqlParameter  
                    {  
                        ParameterName = "@newPrice",  
                        Value = price  
                    };  
                    command.Parameters.Add(nameParam1);  
                    command.Parameters.Add(nameParam2);  
                    command.ExecuteNonQuery();  
                    connection.Close();  
                }  
            }  
            catch (SqlException ex)  
            {  
                MessageBox.Show(ex.Message);  
            }  
        }  
   
   
        private void label1\_Click(object sender, EventArgs e)  
        {  
   
        }  
        private void buttonAddProduct\_Click(object sender, EventArgs e)  
        {  
            AddOproduct(comboBox1.Text, typeProduct.Text, material.Text, Color.Text, Price.Text);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.ListProducts". При необходимости она может быть перемещена или удалена.  
            this.listProductsTableAdapter.Fill(this.dBSHOPDataSet1.ListProducts);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.LISTORDER". При необходимости она может быть перемещена или удалена.  
            this.lISTORDERTableAdapter.Fill(this.dBSHOPDataSet1.LISTORDER);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet2.NumberOfItemsSold". При необходимости она может быть перемещена или удалена.  
            this.numberOfItemsSoldTableAdapter.Fill(this.dBSHOPDataSet2.NumberOfItemsSold);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.ListCategory". При необходимости она может быть перемещена или удалена.  
            this.listCategoryTableAdapter.Fill(this.dBSHOPDataSet1.ListCategory);  
        }  
   
        private void buttonChangePrice\_Click(object sender, EventArgs e)  
        {  
            ChangePriceproduct(codeChagePrise.Text, newPrice.Text);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.ListProducts". При необходимости она может быть перемещена или удалена.  
            this.listProductsTableAdapter.Fill(this.dBSHOPDataSet1.ListProducts);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.LISTORDER". При необходимости она может быть перемещена или удалена.  
            this.lISTORDERTableAdapter.Fill(this.dBSHOPDataSet1.LISTORDER);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet2.NumberOfItemsSold". При необходимости она может быть перемещена или удалена.  
            this.numberOfItemsSoldTableAdapter.Fill(this.dBSHOPDataSet2.NumberOfItemsSold);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.ListCategory". При необходимости она может быть перемещена или удалена.  
            this.listCategoryTableAdapter.Fill(this.dBSHOPDataSet1.ListCategory);  
        }  
   
        private void buttonDeleteProduct\_Click(object sender, EventArgs e)  
        {  
            Deleteproduct(codeDeleteProduct.Text);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.ListProducts". При необходимости она может быть перемещена или удалена.  
            this.listProductsTableAdapter.Fill(this.dBSHOPDataSet1.ListProducts);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.LISTORDER". При необходимости она может быть перемещена или удалена.  
            this.lISTORDERTableAdapter.Fill(this.dBSHOPDataSet1.LISTORDER);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet2.NumberOfItemsSold". При необходимости она может быть перемещена или удалена.  
            this.numberOfItemsSoldTableAdapter.Fill(this.dBSHOPDataSet2.NumberOfItemsSold);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.ListCategory". При необходимости она может быть перемещена или удалена.  
            this.listCategoryTableAdapter.Fill(this.dBSHOPDataSet1.ListCategory);  
        }  
   
        private void back\_Click(object sender, EventArgs e)  
        {  
            FormAdministrator newForm = new FormAdministrator();  
            newForm.Show();  
            Hide();  
        }  
    }

**Меню работника отдела приема и отправки товара**

public partial class FormOrders : Form  
    {  
        static String ConnectString = "Data Source=DESKTOP-900BQME;Initial Catalog=DBSHOP;Integrated Security=True";  
        public FormOrders()  
        {  
            InitializeComponent();  
        }  
   
        private void Orders\_Load(object sender, EventArgs e)  
        {  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet2.ListDriverOpen". При необходимости она может быть перемещена или удалена.  
            this.listDriverOpenTableAdapter.Fill(this.dBSHOPDataSet2.ListDriverOpen);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.LISTORDER". При необходимости она может быть перемещена или удалена.  
            this.lISTORDERTableAdapter.Fill(this.dBSHOPDataSet1.LISTORDER);  
   
        }  
   
        private  void ReadyOrder(String idOrder)  
        {  
            string sqlExpression = "OrderAreManufactured";  
            try  
            {  
                using (SqlConnection connection = new SqlConnection(ConnectString))  
                {  
   
                    connection.Open();  
                    SqlCommand command = new SqlCommand(sqlExpression, connection);  
                    command.CommandType = System.Data.CommandType.StoredProcedure;  
                    SqlParameter nameParam1 = new SqlParameter  
                    {  
                        ParameterName = "@Id",  
                        Value = int.Parse(idOrder)  
                    };  
                    command.Parameters.Add(nameParam1);  
                    var returnValue = command.Parameters.Add("@Return", SqlDbType.Int);  
                    returnValue.Direction = ParameterDirection.ReturnValue;  
                    command.ExecuteNonQuery();  
                    var result = (int)returnValue.Value;  
                    if (result ==-1)  
                    MessageBox.Show($"Неверный номер заказа");  
                    connection.Close();  
                }  
            }  
            catch (SqlException ex)  
            {  
                MessageBox.Show(ex.Message);  
            }  
        }  
   
   
        private void GoDevilary(String idOrder,String IdDriver)  
        {  
            string sqlExpression = "OrderDevileryBegin";  
            try  
            {  
                using (SqlConnection connection = new SqlConnection(ConnectString))  
                {  
   
                    connection.Open();  
                    SqlCommand command = new SqlCommand(sqlExpression, connection);  
                    command.CommandType = System.Data.CommandType.StoredProcedure;  
                    SqlParameter nameParam1 = new SqlParameter  
                    {  
                        ParameterName = "@IdOrder",  
                        Value = int.Parse(idOrder)  
                    };  
   
                    SqlParameter nameParam2 = new SqlParameter  
                     {  
                        ParameterName = "@IdDriver",  
                        Value = int.Parse(idOrder)  
                     };  
                    command.Parameters.Add(nameParam1);  
                    command.Parameters.Add(nameParam2);  
   
                    var returnValue = command.Parameters.Add("@Return", SqlDbType.Int);  
                    returnValue.Direction = ParameterDirection.ReturnValue;  
                    command.ExecuteNonQuery();  
                    var result = (int)returnValue.Value;  
                    if (result == -1)  
                        MessageBox.Show($"Неверные данные");  
                    connection.Close();  
                }  
            }  
            catch (SqlException ex)  
            {  
                MessageBox.Show(ex.Message);  
            }  
        }  
   
        private void OrderDevileryFinish(String idOrder)  
        {  
            string sqlExpression = "OrderDevileryEnd";  
            try  
            {  
                using (SqlConnection connection = new SqlConnection(ConnectString))  
                {  
   
                    connection.Open();  
                    SqlCommand command = new SqlCommand(sqlExpression, connection);  
                    command.CommandType = System.Data.CommandType.StoredProcedure;  
                    SqlParameter nameParam1 = new SqlParameter  
                    {  
                        ParameterName = "@Id",  
                        Value = int.Parse(idOrder)  
                    };  
                    command.Parameters.Add(nameParam1);  
                    var returnValue = command.Parameters.Add("@Return", SqlDbType.Int);  
                    returnValue.Direction = ParameterDirection.ReturnValue;  
                    command.ExecuteNonQuery();  
                    var result = (int)returnValue.Value;  
                    if (result == -1)  
                        MessageBox.Show($"Неверный номер заказа");  
                    connection.Close();  
                }  
            }  
            catch (SqlException ex)  
            {  
                MessageBox.Show(ex.Message);  
            }  
        }  
   
        private void buttonReadyDriver\_Click(object sender, EventArgs e)  
        {  
   
            ReadyOrder(CodeOrder.Text);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet2.ListDriverOpen". При необходимости она может быть перемещена или удалена.  
            this.listDriverOpenTableAdapter.Fill(this.dBSHOPDataSet2.ListDriverOpen);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.LISTORDER". При необходимости она может быть перемещена или удалена.  
            this.lISTORDERTableAdapter.Fill(this.dBSHOPDataSet1.LISTORDER);  
        }  
   
        private void buttonGoDevillery\_Click(object sender, EventArgs e)  
        {  
            GoDevilary(CodeOrder.Text, CodeDriver.Text);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet2.ListDriverOpen". При необходимости она может быть перемещена или удалена.  
            this.listDriverOpenTableAdapter.Fill(this.dBSHOPDataSet2.ListDriverOpen);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.LISTORDER". При необходимости она может быть перемещена или удалена.  
            this.lISTORDERTableAdapter.Fill(this.dBSHOPDataSet1.LISTORDER);  
        }  
   
        private void buttonFinnishDevillery\_Click(object sender, EventArgs e)  
        {  
            OrderDevileryFinish(CodeOrder.Text);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet2.ListDriverOpen". При необходимости она может быть перемещена или удалена.  
            this.listDriverOpenTableAdapter.Fill(this.dBSHOPDataSet2.ListDriverOpen);  
            // TODO: данная строка кода позволяет загрузить данные в таблицу "dBSHOPDataSet1.LISTORDER". При необходимости она может быть перемещена или удалена.  
            this.lISTORDERTableAdapter.Fill(this.dBSHOPDataSet1.LISTORDER);  
        }  
    }  
    }

**Главный модуль программы:**

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Threading.Tasks;  
using System.Windows.Forms;  
   
namespace TRBD  
{  
    static class Program  
    {  
        /// <summary>  
        /// Главная точка входа для приложения.  
        /// </summary>  
        [STAThread]  
        static void Main()  
        {  
            Application.EnableVisualStyles();  
            Application.SetCompatibleTextRenderingDefault(false);  
            Application.Run(new Form1());  
        }  
    }  
}