Отчёт по курсовой работе

по дисциплине «Базы данных»

Разработка многопользовательской автоматизированной системы управления организацией. Объект автоматизации – пивной завод.

|  |  |
| --- | --- |
| Студент гр. 3530904/00105 | Цапов Н. В. |
| Преподаватель | Гасанова И. А. |

Оглавление

[ЗАДАНИЕ 3](#_Toc121245970)

[Анализ предметной области 4](#_Toc121245971)

[Описание таблиц и связей 10](#_Toc121245972)

[Схема 10](#_Toc121245973)

[Описание таблиц 11](#_Toc121245974)

[Запросы 15](#_Toc121245975)

[Представления 15](#_Toc121245976)

[Запросы 17](#_Toc121245977)

[Описание программы 27](#_Toc121245978)

[Скриншоты системы 28](#_Toc121245979)

[Авторизация и регистрация 28](#_Toc121245980)

[Окно администратора ИС 29](#_Toc121245981)

[Окно рабочего завода (Менеджер по продажам) 30](#_Toc121245982)

[Окно менеджера по закупкам (розничные магазины) 35](#_Toc121245983)

[Заключение 37](#_Toc121245984)

[Список литературы 38](#_Toc121245985)

[Приложение (скрипт создания бд) 39](#_Toc121245986)

## ЗАДАНИЕ

НА ВЫПОЛНЕНИЕ КУРСОВОГО ПРОЕКТА (КУРСОВОЙ РАБОТЫ)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| cтуденту группы | 3530904/00105 | |  | Цапов Никита Владимирович | | | |
|  | *(номер группы)* | |  | *(фамилия, имя, отчество)* | | | |
| ***1. Тема проекта (работы)*** | | | *Разработка многопользовательской* | | | | |
| *автоматизированной системы управления организацией.* | | | | | | | |
| Объект автоматизации - пивной завод*.  Задание № 19* | | | | | | | |
| ***2. Срок сдачи студентом законченного проекта (работы)*** | | | | | | | 01.12.2022 |
| ***3. Исходные данные к проекту (работе)*** | | | | | |  | |
| *Описание предметной области* | | | | | | | |
|  | | | | | | | |
|  | | | | | | | |
| ***4.Содержание пояснительной записки***(перечень подлежащих разработке вопросов: введение, основная часть (раскрывается структура основной части), заключение, список использованных источников, приложения)*.* | | | | | | | |
| *Введение. Анализ предметной области. Проектирование схемы данных.* | | | | | | | |
| *Реализация базы данных в среде  SQL Server. Разработка представлений и* | | | | | | | |
| *хранимых процедур. Разработка клиентского приложения. Тестирование.* | | | | | | | |
| *Заключение. Список использованных источников.* | | | | | | | |
| Примерный объем пояснительной записки | | | | *15-20* | | | страниц машинописного |
| текста | | | | | | | |
| 5. Перечень графического материала (с указанием обязательных чертежей и | | | | | | | |
| плакатов) | *не предоставляется* | | | | | | |
|  | | | | | | | |
| 6. Консультанты | |  | | | | | |
|  | | | | | | | |
| 7. Дата получения задания: «11» \_\_сентября\_\_ 2022 г. | | | | | | | |
|  | | | | | | | |
| Руководитель | | | |  |  | | Гасанова И.А. |
|  | | | | *(подпись)* |  | | *(инициалы, фамилия)* |
| Задание принял к исполнению | | | |  |  | | Цапов Н. В. |
|  | | | | *(подпись)* |  | | *(инициалы, фамилия)* |
|  | | | | | | | 11.09.2022 |

(дата)

## Анализ предметной области

Группы пользователей разрабатываемой информационной системы (ИС)

|  |  |
| --- | --- |
| № гп | Наименование пользователя |
| 1 | Менеджер по продажам (завод) |
| 2 | Менеджер по закупкам (Розничные магазины) |
| 3 | Администратор ИС |

Функции групп пользователей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № гп | Выполняемая функция | Входные данные | Выходные данные | Функции, которые должны быть реализованы в ИС |
| Менеджер по продажам (завод) | | | | |
| 1 | Просмотр базы товаров |  | * Наименование продукта(product\_name) * Тип продукта(product\_type) * UPS-код(upc\_code) * Себестоимость продукта(prime\_price) * Розничная цена(retail\_price) * Розничная цена с учетом скидки(discount\_price) * Количество товара на складе(amount) | Просмотр всех записей и полей таблицы products, объединенной с discounts, stock |
| 2 | Добавление товара в каталог | * Наименование продукта(product\_name) * Тип продукта(product\_type) * UPС-код(upc\_code) * Розничная цена(retail\_price) * Скидка на товар(discount) * Количество товара на складе(amount) | Таблица products | Добавление записи в таблицу products |
| 3 | Обновление информации о товаре | * Id продукта(product\_id): * Изменяемые поля записи: * Наименование продукта(product\_name) * Розничная цена(retail\_price) | Таблица products | Редактирование записи в таблице products |
| 4 | Изменение скидки на товар | * Id товара(product\_id), * Скидка(discount) | Таблицы discounts, products | Редактирование записи в таблице discounts  Изменение вычисляемого значения discount\_price в таблице products |
| 5 | Изменить количество товара на складе | * Id товара(product\_id) * Количество(amount) | Таблица stock | Редактирование записи в таблице stock |
| 6 | Просмотр информации о заказах | Фильтры:   * Статус заказа(order\_status) * Название организации (customer\_name) * Сортировка по дате (Last week, Last month, Last year) | * Номер заказа(order\_id) * Название организации (customer\_name) * Дата заказа(order\_date) * Статус заказа(order\_status) * Себестоимость заказа(total\_prime\_cost) * Стоимость заказа(total\_cost) | Просмотр записей таблиц orders, customers |
| 7 | Просмотр содержимого заказа |  | * Название товара(product\_name) * Количество товара(amount) * Цена (price) * Стоимость заказа(cost) | Просмотр записей таблиц orders, products, order\_datails, customers |
| 8 | Просмотр информации о заказчиках |  | * Название организации customer\_name * Email(email) * Номер телефона(phone\_number) * Адрес(address) | Просмотр всех записей таблицы customers |
| 9 | Получение значения прибыли завода за все время |  | Вычисляемое значение | Вывод на экран прибыли завода за все время |
| 10 | Получение значения прибыли завода от конкретного заказчика | * Id заказчика(customer\_id) | Вычисляемое значение | Вывод на экран прибыли завода за все время от заказов конкретного заказчика |
| 11 | Получить информацию о недостающих товарах на складе для конкретного заказа | * Id заказа(order\_id) | Каждый товар:   * Id товара(product\_id) * Вычисляемое значение: order\_details.amount – stock.amount | Просмотр записей таблиц order\_details по конкретному значению order\_id и вычисление количества недостающих товаров на складе |
| 12 | Получить информацию о недостающих товарах для конкретного заказчика | * Id заказчика(customer\_id) | Каждый товар:   * Id товара(product\_id) * Вычисляемое значение: order\_details.amount – stock.amount | Просмотр записей таблиц order\_details по конкретному значению order\_id для каждого заказа заказчика(customer\_id) и вычисление количества недостающих товаров на складе |
| 13 | Получить информацию о недостающих товарах для всех существующих заказов |  | Каждый товар:   * Id товара(product\_id) * Вычисляемое значение: order\_details.amount – stock.amount | Просмотр записей таблиц order\_details по каждому значению order\_id и вычисление количества недостающих товаров на складе |
| Менеджер по закупкам (Розничные магазины) | | | | |
| 1 | Просмотр каталога товаров |  | * Наименование продукта(product\_name) * Тип продукта(product\_type) * Розничная цена(retail\_price) * Розничная цена с учетом скидки(discount\_price) | Просмотр записей таблиц products, discounts(кроме cost\_price) |
| 2 | Создание учетной записи заказчика | * Название организации customer\_name * Email(email), * Номер телефона(phone\_number), * Адрес(address) * Пароль(user\_password) | Таблицы customers, users | Добавление записи в таблицы customers, users |
| 3 | Изменение данных об учетной записи заказчика | * Id заказчика(customer\_id): * Номер телефона(phone\_number) * Адрес(address) | Таблица customers | Редактирование записи в таблице customers, |
| 4 | Формирование заказа | * Id заказчика(customer\_id): * Id товара(product\_id), * Количество(amount) | Таблицы order\_details,  orders | Добавление записей в таблицы orders, order\_details |
| 5 | Отмена заказа | * Id заказа(order\_id) | Таблица orders | Установить статус заказа(order\_state) в “cancelled” в таблице orders |
| 6 | Просмотр своих заказов | * Id закакзчика(customer\_id)   Фильтры:   * Статус заказа(order\_status) * Сортировка по дате (Last week, Last month, Last year) | * Номер заказа(order\_id) * Дата заказа(order\_date) * Статус заказа(order\_state) * Стоимость заказа(total\_cost) | Просмотр записей таблицы orders |
| 7 | Просмотр содержимого заказа | * Id заказа(order\_id) | * Название товара(product\_name) * Количество товара(amount) * Цена (price) * Стоимость заказа(cost) | Просмотр записей таблиц orders, order\_details |
| Администратор ИС | | | | |
| 1 | Создание учетной записи работника | * Адрес электронной почты(email) * Имя(first\_name) * Фамилия(second\_name) * Пароль(user\_password) * Роль(user\_role) | Таблицы users, workers | Добавление записей в таблицы users, workers |
| 2 | Удаление учетной записи работника | * Id работника(worker\_id) | Таблицы users, workers | Удаление записей из таблиц users, workers |
| 3 | Просмотр учетных записей работников завода | Фильтр по роли (worker/admin) | * Ник в системе(username) * Адрес электронной почты(email) * Имя(first\_name) * Фамилия(second\_name) * Пароль(user\_password) * Роль(user\_role) | Просмотр записей таблиц users, workers |

Хранимые данные

Группы пользователей разрабатываемой информационной системы (ИС)

|  |  |  |
| --- | --- | --- |
| № гп | Наименование пользователя |  |
| 1 | Менеджер по продажам (завод) | МП |
| 3 | Менеджер по закупкам (Розничные магазины) | МЗ |
| 4 | Администратор ИС | АИС |

|  |  |  |  |
| --- | --- | --- | --- |
| Описание поля | Название поля | Пользователи, которым разрешен доступ на изменение | Ограничения по типу и значению |
| **Пользователи(userss)** | | | |
| ID пользователя | user\_id |  | int primary key identity(1, 1) not null |
| Имя пользователя | username |  | Varchar(30), not null  CHECK(username !=''), UNIQUE |
| Пароль | user\_password |  | varbinary(256), not null  CONSTRAINT CK\_USER\_PASSWORD CHECK(user\_password !='') |
| Роль | user\_role |  | Varchar(10), not null |
| **Рабочие завода(workers)** | | | |
| ID пользователя | user\_id |  | foreign key(user\_id) references users (user\_id) |
| ID рабочего | worker\_id |  | int primary key identity(1, 1) not null |
| Электронная почта рабочего | email |  | Varchar(30), not null  CONSTRAINT CK\_WORKER\_EMAIL CHECK(dbo.isValidEmail(email)=1)  CONSTRAINT UQ\_WORKER\_EMAIL UNIQUE |
| Имя рабочего | first\_name |  | Varchar(30), not null CHECK(first\_name !='') |
| Фамилия рабочего | second\_name |  | Varchar(30), not null CHECK(second\_name !='') |
| **Заказчики(customers)** | | | |
| ID пользователя | user\_id |  | foreign key(user\_id) references users (user\_id) |
| ID заказчика | customer\_id |  | int primary key identity(1, 1) not null |
| Имя заказчика | customer\_name |  | Varchar(30), not null |
| Электронная почта заказчика | email |  | Varchar(30), not null  CONSTRAINT CK\_CUSTOMER\_EMAIL CHECK(dbo.isValidEmail(email) = 1)  CONSTRAINT UQ\_CUSTOMER\_EMAIL UNIQUE |
| Номер телефона заказчика | phone\_number | МЗ | Varchar(30), not null  CONSTRAINT CK\_CUSTOMER\_PHONE\_NUMBER CHECK(dbo.isValidPhoneNumber(phone\_number)=1)  CONSTRAINT UQ\_CUSTOMER\_PHONE\_NUMBER UNIQUE |
| Адрес | address | МЗ | Varchar(30), not null CHECK(address !='') |
| **Заказы(orders)** | | | |
| ID заказчика | customer\_id |  | foreign key(customer\_id) references customers (customer\_id) |
| ID заказа | order\_id |  | int primary key identity(1, 1) not null |
| Дата заказа | order\_date |  | Datetime, not null |
| Статус заказа | order\_status | МП, МЗ(отменить) | Varchar(30), not null default ‘pending’ |
| Итоговая себестоимость заказа | total\_prime\_cost |  | Decimal(8, 2) |
| Итоговая стоимость заказа | total\_cost |  | Decimal(8, 2) |
| **Состав заказа(order\_details)** | | | |
| ID заказа | order\_id |  | foreign key(order\_id) references orders (order\_id) |
| ID продукта | product\_id |  | foreign key(product\_id) references products (product\_id) |
| Количество продукта | amount |  | Int, not null |
| Цена товара | price |  | Decimal(8, 2) |
| Стоимость товаров в заказе | cost |  | as price \* amount |
| **Товары(products)** | | | |
| ID товара | product\_id |  | int primary key identity(1, 1) not null |
| Наименование продукта | product\_name | МП | Varchar(30), not null UNIQUE |
| Тип товара | product\_type | МП | Varchar(30), not null |
| UPC код | Upc\_code | МП | Varchar(12)  CONSTRAINT UQ\_PRODUCT\_UPC UNIQUE  CONSTRAINT CK\_PRODUCT\_UPC CHECK(LEN(upc\_code) = 12) |
| Себестоимость продукта | prime\_price | МП | Decimal(8, 2), not null |
| Цена единицы продукта в розницу | retail\_price |  | as prime\_price \* 1.38 |
| Розничная цена с учетом скидки | discount\_price |  | Decimal(8, 2), not null |
| **Скидки(discounts)** | | | |
| ID товара | product\_id |  | foreign key(product\_id) references products (product\_id) |
| Действующая скидка на продукт | discount | МП | Decimal(3, 2), not null |
| **Склад(stock)** | | | |
| ID товара | product\_id |  | foreign key(product\_id) references products (product\_id) |
| Количество продукта на складе | amount | МП | Int, not null |

## Описание таблиц и связей

### Схема

****

**Таблицы:**

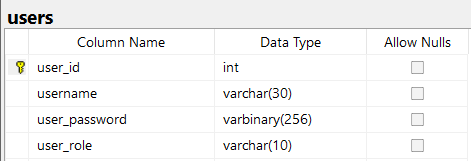
1. **users** – таблица содержит информацию о пользователях ИС (Менеджер по продажам (завод), Менеджер по закупкам (Розничные магазины)). С помощью поля user\_role можно понять заказчик или сотрудник завода авторизирован.
2. **workers** – таблица содержит информацию о каждом работнике пивного завода (Менеджер по продажам (завод)).
3. **customers** – таблица содержит информацию о сторонних заказчиках (Менеджер по закупкам (Розничные магазины)).
4. **orders** – таблица содержит информацию о сформированных заказах от сторонних заказчиков.
5. **order\_details** – таблица содержит информацию о составе каждого сформированного заказа из таблицы orders.
6. **products** – таблица содержит информацию о продукции завода.
7. **discounts** – таблица содержит информацию о скидках на каждый товар в формате (“0.15”=15%, “0.10”=15% и т.д).
8. **stock** - таблица склада, содержащая информацию о количестве каждого товара на складе

**Связи таблицы:**

|  |  |  |  |
| --- | --- | --- | --- |
| Таблицы | Тип | Primary key | Foreign\_key |
| users, workers | One-to-one | users.user\_id | workers.user\_id |
| users, cusotmers | One-to-one | users.user\_id | customers.user\_id |
| customers,orders | One-to-many | cutomers.customer\_id | orders.customer\_id |
| orders, order\_details | One-to-many | orders.order\_id | order\_details.order\_id |
| order\_details, products | Many-to-many | products.product\_id | order\_details.product\_id |
| stock, products | One-to-one | products.product\_id | stock.product\_id |
| discounts, products | One-to-one | products.product\_id | discounts.product\_id |

### Описание таблиц

**Users**



    CREATE TABLE users (

        [user\_id] [int] PRIMARY KEY IDENTITY(1,1) NOT NULL,

        [username] [varchar](30) NOT NULL CHECK(username !='') UNIQUE,

        [user\_password] varbinary(256) NOT NULL

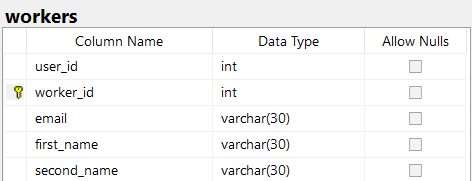
            CONSTRAINT CK\_USER\_PASSWORD CHECK(user\_password !=''),

        [user\_role] [varchar](10) NOT NULL

    );

* user\_id - ID пользователя
* username - Ник пользователя в системе
* user\_password - Пароль
* user\_role – роль

**Workers**



    CREATE TABLE workers (

        [user\_id] [int] NOT NULL FOREIGN KEY([user\_id]) REFERENCES [dbo].[users] ([user\_id]) on delete cascade,

        [worker\_id] [int] PRIMARY KEY IDENTITY(1,1) NOT NULL,

        [email] [varchar](30) NOT NULL

            CONSTRAINT CK\_WORKER\_EMAIL CHECK(dbo.isValidEmail(email) = 1)

            CONSTRAINT UQ\_WORKER\_EMAIL UNIQUE,

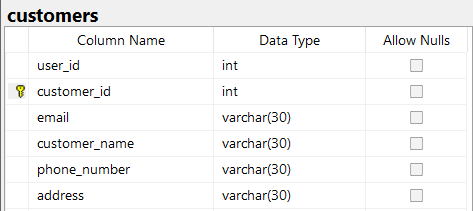
        [first\_name] [varchar](30) NOT NULL CHECK(first\_name !=''),

        [second\_name] [varchar](30) NOT NULL CHECK(second\_name !='')

    );

* + user\_id - ID пользователя
  + worker\_id - ID рабочего
  + email - Электронная почта рабочего
  + first\_name - Имя рабочего
* second\_name - Фамилия рабочего

**Customers**



    CREATE TABLE customers (

        [user\_id] [int] NOT NULL FOREIGN KEY([user\_id]) REFERENCES [dbo].[users] ([user\_id]),

        [customer\_id] [int] PRIMARY KEY IDENTITY(1,1) NOT NULL,

        [email] [varchar](30) NOT NULL

            CONSTRAINT CK\_CUSTOMER\_EMAIL CHECK(dbo.isValidEmail(email) = 1)

            CONSTRAINT UQ\_CUSTOMER\_EMAIL UNIQUE,

        [customer\_name] [varchar](30) NOT NULL CHECK(customer\_name !=''),

        [phone\_number] [varchar](30) NOT NULL

            CONSTRAINT CK\_CUSTOMER\_PHONE\_NUMBER CHECK(dbo.isValidPhoneNumber(phone\_number) = 1)

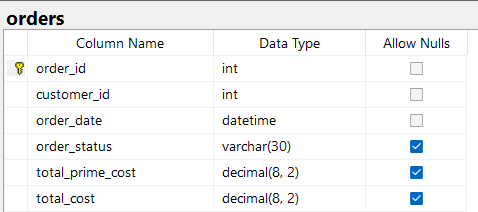
            CONSTRAINT UQ\_CUSTOMER\_PHONE\_NUMBER UNIQUE,

        [address] [varchar](30) NOT NULL CHECK(address !='')

    );

* user\_id - ID пользователя
* customer\_id - ID заказчика
* customer\_name – Наименование заказчика
* email - Электронная почта заказчика
* phone\_number - Номер телефона заказчика
* address – Адрес

**Orders**



    CREATE TABLE orders (

        [order\_id] [int] PRIMARY KEY IDENTITY(1,1) NOT NULL,

        [customer\_id] [int] NOT NULL FOREIGN KEY([customer\_id]) REFERENCES [dbo].[customers] ([customer\_id]),

        [order\_date] [datetime] NOT NULL,

        [order\_status] [varchar](30) DEFAULT 'pending',

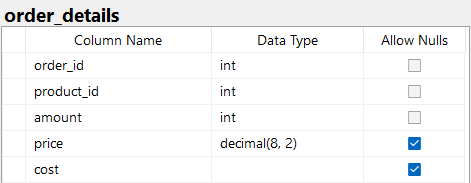
        [total\_prime\_cost] [decimal](8, 2) NULL,

        [total\_cost] [decimal](8, 2) NULL

    );

* + order\_id - ID заказа
  + customer\_id - ID заказчика
  + order\_date - Дата заказа
  + order\_status - Статус заказа
* total\_prime\_cost - Итоговая себестоимость заказа(вычисляемое по триггеру значение)
* total\_cost - Итоговая стоимость заказа(вычисляемое по триггеру значение)

**Order\_details**



    CREATE TABLE order\_details (

        [order\_id] [int] NOT NULL FOREIGN KEY([order\_id]) REFERENCES [dbo].[orders] ([order\_id]),

        [product\_id] [int] NOT NULL FOREIGN KEY([product\_id]) REFERENCES [dbo].[products] ([product\_id]),

        [amount] [int] NOT NULL,

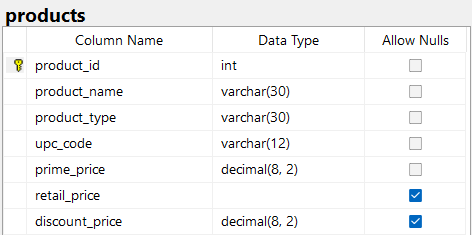
        [price] [decimal](8, 2) NULL,

        [cost] as price \* amount

    );

* + order\_id – ID заказа
  + product\_id - ID продукта
* amount - Количество продукта
* price – Цена товара
* cost – Стоимость товаров

**Products**



    CREATE TABLE products (

        [product\_id] [int] PRIMARY KEY IDENTITY(1,1) NOT NULL,

        [product\_name] [varchar](30) NOT NULL CHECK(product\_name !='')

            CONSTRAINT UQ\_PRODUCT\_NAME UNIQUE,

        [product\_type] [varchar](30) NOT NULL,

        [upc\_code] [varchar](12) NOT NULL

            CONSTRAINT UQ\_PRODUCT\_UPC UNIQUE(upc\_code)

            CONSTRAINT CK\_PRODUCT\_UPC CHECK(LEN(upc\_code) = 12),

        [prime\_price] [decimal](8, 2) NOT NULL,

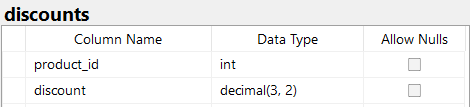
        [retail\_price] as prime\_price \* 1.38,

        [discount\_price] [decimal](8, 2)

    );

* product\_id - ID товара
* product\_name - Наименование товара
* product\_type - Тип товара
* upc\_code – UPC код товара
* prime\_price - Себестоимость товара
* retail\_price - Цена единицы товара в розницу
* discount\_price - Розничная цена со скидкой

**Discounts**



    CREATE TABLE discounts (

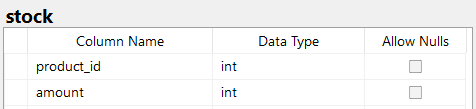
        [product\_id] [int] NOT NULL FOREIGN KEY([product\_id]) REFERENCES [dbo].[products] ([product\_id]),

        [discount] [decimal](3, 2) NOT NULL

    );

* product\_id - ID товара
* discount - Действующая скидка на продукт

**Stock**



    CREATE TABLE stock (

        [product\_id] [int] NOT NULL FOREIGN KEY([product\_id]) REFERENCES [dbo].[products] ([product\_id]),

        [amount] [int] NOT NULL

    );

* product\_id - ID товара
* amount - Количество товара на складе

## Запросы

Все представления и запросы хранятся в процедурах

Представления:

1) Вывод каталога товаров, его количества на складе, и цены со скидкой (сложный: таблицы products, stock)

create view ProductsStockView as

    select product\_name,

           product\_type,

           upc\_code,

           prime\_price,

           retail\_price,

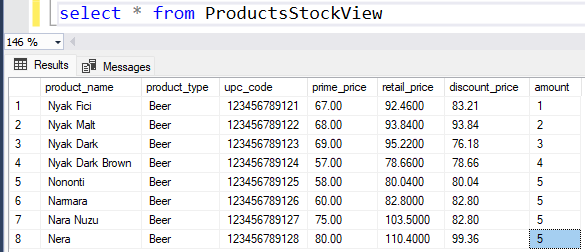
           discount\_price,

           amount

           from products

                join stock on products.product\_id=stock.product\_id

GO



2) Вывод всех заказов и их содержимого (сложный: таблицы orders, order\_details, customers)

create view OrdersOrderDetailsCustomersView as

    select orders.order\_id,

       customer\_name,

       order\_date,

       orders.order\_status,

       products.product\_id,

       product\_name,

       amount,

       price,

       cost

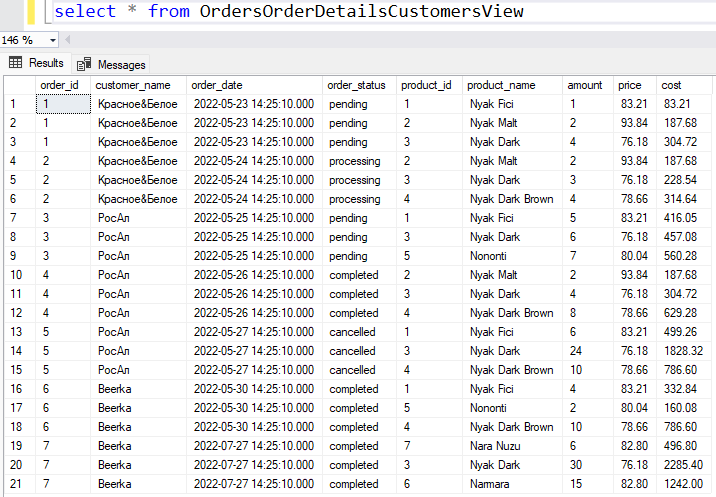
        from orders

            join order\_details on orders.order\_id=order\_details.order\_id

            join products on order\_details.product\_id=products.product\_id

            join customers on orders.customer\_id=customers.customer\_id

go



### Запросы

#### Вывод

1. Вывод базы товаров для сотрудника завода с количеством товара на складе и себестоимости продукта(сложный)

create or alter procedure ShowProductsBase

as

select product\_name,

        product\_type,

        upc\_code,

        prime\_price,

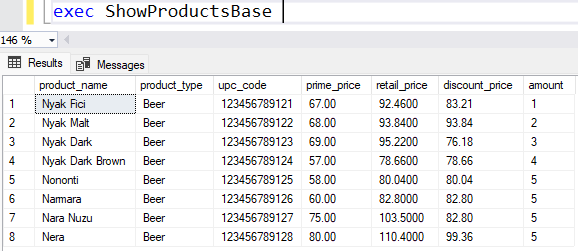
        retail\_price,

        discount\_price,

        amount

from ProductsStockView

go



1. Вывод каталога товаров для заказчика(сложный)

create or alter procedure ShowProductsCatalog

as

select product\_name,

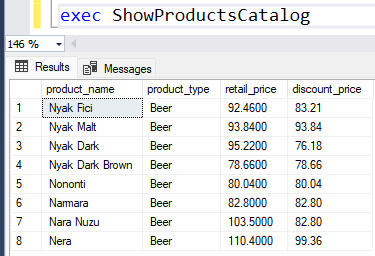
        product\_type,

        retail\_price,

        discount\_price

from ProductsStockView

go



1. Получить информацию о недостающих товарах на складе для существующих заказов(сложный)

create or alter procedure ShowLackProducts

as

select product\_name,

       abs(amount\_on\_stock - ordered) as lack\_on\_stock

from (select product\_name,

             sum(OrdersOrderDetailsCustomersView.amount) as ordered,

             stock.amount as amount\_on\_stock

      from OrdersOrderDetailsCustomersView

          join stock

              on OrdersOrderDetailsCustomersView.product\_id=stock.product\_id

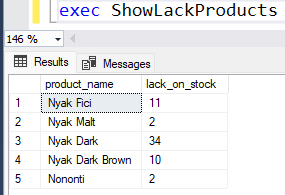
      where order\_status != 'completed'

      group by product\_name, stock.amount)

as a

where (amount\_on\_stock - ordered) < 0

go



1. Вывод информации о недостающих товарах на складе для существующих заказов для конкретного заказа (сложный)

create or alter procedure ShowLackProductsByOrder\_id

    @order\_id int

as

select product\_name,

    abs(amount\_on\_stock - ordered) as lack\_on\_stock

from

    (select product\_name,

        sum(OrdersOrderDetailsCustomersView.amount) as ordered,

        stock.amount as amount\_on\_stock

    from OrdersOrderDetailsCustomersView

        join stock on OrdersOrderDetailsCustomersView.product\_id=stock.product\_id

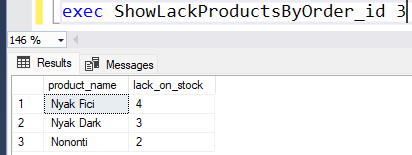
    where order\_status != 'completed' and order\_id = @order\_id

group by product\_name, stock.amount)

as a

where (amount\_on\_stock - ordered) < 0

go



1. Вывод информации о недостающих товарах на складе для существующих заказов для конкретного заказчика (сложный)

create or alter procedure ShowLackProductsByCustomer\_name

    @customer\_name varchar(30)

as

select product\_name,

    abs(amount\_on\_stock - ordered) as lack\_on\_stock

from

    (select product\_name,

        sum(OrdersOrderDetailsCustomersView.amount) as ordered,

        stock.amount as amount\_on\_stock

    from OrdersOrderDetailsCustomersView join stock on OrdersOrderDetailsCustomersView.product\_id=stock.product\_id

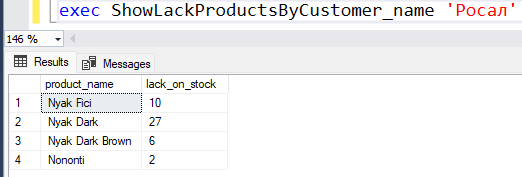
    where order\_status != 'completed' and customer\_name = @customer\_name

    group by product\_name, stock.amount)

as a

where (amount\_on\_stock - ordered) < 0

go



1. Вывод информации обо всех заказах(сложный)

create or alter procedure ShowAllOrders

as

select order\_id,

       customer\_name,

       order\_date,

       order\_status,

       total\_prime\_cost,

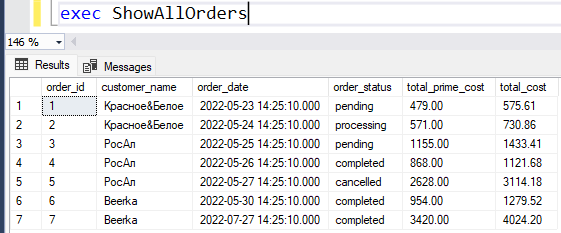
       total\_cost

from Orders

    join customers

        on orders.customer\_id=customers.customer\_id

go



1. Вывод информации обо всех заказах по различным фильтрам (customer\_name, order\_status, date) (сложный)

ShowOrdersByCustomer\_nameAndOrder\_status\_ForWorker,

ShowOrdersByOrder\_statusAnd\_Period\_ForWorker,

ShowOrdersByOrder\_status\_ForWorker,

ShowOrdersByCustomer\_nameAnd\_Period\_ForWorker,

ShowOrdersByCustomer\_name\_ForWorker,

ShowAllOrders\_Period\_ForWorker,

Запросы практически идентичны, отличия только в фильтре

create or alter procedure ShowOrdersByCustomer\_nameAndOrder\_statusAnd\_Period\_ForWorker

    @customer\_name varchar(30),

    @order\_status varchar(30),

    @period int

as

select order\_id,

       customer\_name,

       order\_date,

       order\_status,

       total\_prime\_cost,

       total\_cost

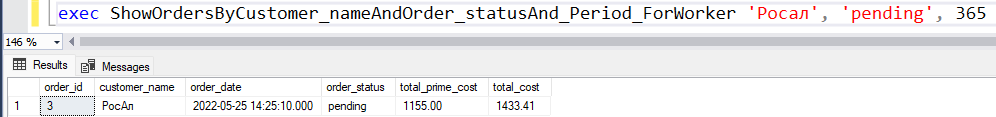
from Orders

    join customers

        on orders.customer\_id=customers.customer\_id

where customer\_name = @customer\_name and order\_status = @order\_status and DATEDIFF(day, order\_date, GETDATE()) < @period

go



1. Вывод информации о своих заказах для заказчика

create or alter procedure ShowOrdersByCustomer\_id\_ForCustomer

    @customer\_id int

as

select order\_id,

    order\_date,

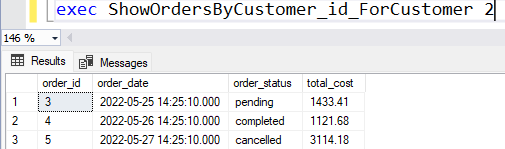
    order\_status,

    total\_cost

from Orders

where customer\_id = @customer\_id

go



1. Вывод информации о своих заказах для заказчика по различным фильтрам (order\_status, date)

ShowOrdersByCustomer\_idAndOrder\_status\_ForCustomer,

ShowOrdersByCustomer\_id\_Period\_ForCustomer

Запросы практически идентичны, отличия только в фильтре

create or alter procedure ShowOrdersByCustomer\_idAndOrder\_status\_Period\_ForCustomer

    @customer\_id int,

    @order\_status varchar(30),

    @period int

as

select order\_id,

    order\_date,

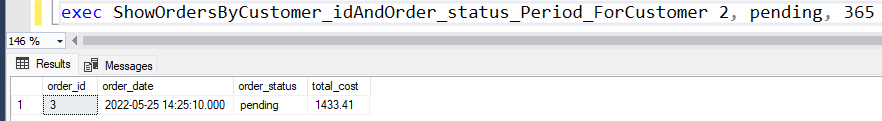
    order\_status,

    total\_cost

from Orders

where customer\_id = @customer\_id and order\_status = @order\_status and DATEDIFF(day, order\_date, GETDATE()) < @period

go



1. Вывод содержимого заказа (сложный)

create or alter procedure ShowOrderDetails

@order\_id int

as

select product\_name,

        amount,

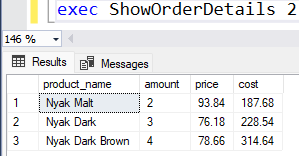
        price,

        cost

    from OrdersOrderDetailsCustomersView

    where order\_id = @order\_id

GO



1. Вывод информации о заказчиках

create or alter procedure ShowCustomers

as

select customer\_name,

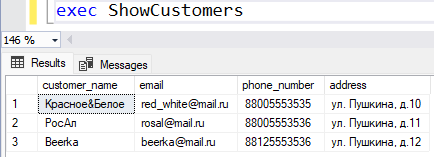
    email,

    phone\_number,

    address

from customers

go



1. Вывод значения прибыли завода за все время

create or alter procedure ShowAllIncome

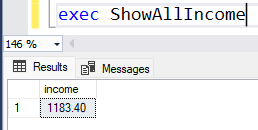
as

select sum(total\_cost - total\_prime\_cost) as income

    from Orders

where order\_status='completed'

go



1. Вывод значения прибыли завода от конкретного заказчика

create or alter procedure ShowIncomeByCustomer\_name

    @customer\_name varchar(30)

as

select sum(total\_cost - total\_prime\_cost) as income

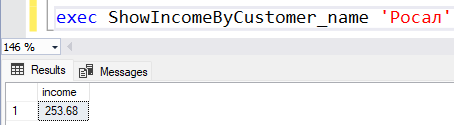
    from Orders

        join customers

            on orders.customer\_id=customers.customer\_id

where order\_status='completed' and customer\_name=@customer\_name

go



1. Вывод информации о сотрудниках завода

create or alter procedure ShowWorkers

as

select username,

    email,

    first\_name,

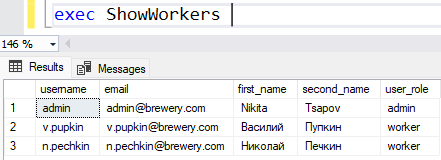
    second\_name,

    user\_role

from workers

    join users on workers.user\_id=users.user\_id

go



1. Вывод информации о сотрудниках завода по фильтру роли

create or alter procedure ShowWorkersByRole

    @user\_role varchar(10)

as

select username,

    email,

    first\_name,

    second\_name,

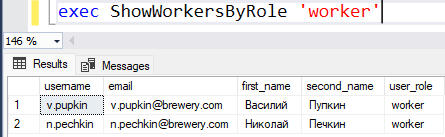
    user\_role

from workers

    join users on workers.user\_id=users.user\_id

    where user\_role = @user\_role

go



#### Вставка

1. Вставка товара в базу товара

create or alter procedure InsertProduct

    @product\_name varchar(30),

    @product\_type varchar(30),

    @upc\_code varchar(12),

    @prime\_price decimal(8, 2),

    @discount decimal(8, 2),

    @amount int

as

BEGIN TRANSACTION

    INSERT INTO products(product\_name, product\_type, upc\_code, prime\_price)

        VALUES (@product\_name, @product\_type, @upc\_code, @prime\_price)

    Declare @product\_id INT

    Set @product\_id=(select product\_id from products where product\_name=@product\_name)

    EXEC UpdateDiscount @product\_id, @discount

    EXEC UpdateAmountOnStock @product\_id, @amount

    IF (@@error <> 0)

        ROLLBACK

COMMIT

GO

1. Вставка учетной записи в таблицу сотрудников завода(сложный)

create or alter procedure InsertWorker

    @email varchar(30),

    @first\_name varchar(30),

    @second\_name varchar(30),

    @user\_password varchar(30)

as

BEGIN

BEGIN TRANSACTION

    INSERT INTO users(username, user\_password, user\_role)

        VALUES (left(@email, charindex('@', @email) - 1), HASHBYTES('SHA2\_256', @user\_password), 'worker');

    INSERT INTO workers(user\_id, email, first\_name, second\_name)

        VALUES (

            (select user\_id from users

                where username=left(@email,  charindex('@', @email) - 1)),

            @email,

            @first\_name,

            @second\_name

        )

COMMIT

END

GO

1. Вставка учетной записи в таблицу заказчиков(сложный)

create or alter procedure InsertCustomer

    @customer\_name varchar(30),

    @email varchar(30),

    @phone\_number varchar(30),

    @address varchar(30),

    @user\_password varchar(30)

as

    BEGIN TRANSACTION

        INSERT INTO users(username,

                user\_password,

                user\_role)

            VALUES (left(@email, charindex('@', @email) - 1),

                    HASHBYTES('SHA2\_256', @user\_password),

                    'customer')

        INSERT INTO customers(user\_id,

                email,

                customer\_name,

                phone\_number,

                address)

            VALUES ((select user\_id from users

                    where username=left(@email,  charindex('@', @email) - 1)),

                @email,

                @customer\_name,

                @phone\_number,

                @address

            )

        IF (@@error <> 0)

            ROLLBACK

    COMMIT

GO

1. Вставка нового заказа(сложный)
   1. В таблицу orders

create or alter procedure InsertOrder

    @customer\_id INT,

    @order\_date DATETIME

as

    INSERT INTO orders(customer\_id, order\_date)

        VALUES (@customer\_id, @order\_date)

go

* 1. В таблицу order\_details

create or alter procedure InsertOrder\_details

    @order\_id INT,

    @product\_id INT,

    @amount INT

as

    INSERT INTO order\_details(order\_id, product\_id, amount)

        VALUES (@order\_id, @product\_id, @amount)

GO

## Описание программы

Задание заключается в разработке многопользовательской автоматизированной системы управления организацией. Объект автоматизации – пивной завод.

В качестве СУБД была предложена Microsoft SQL Server.

Microsoft SQL Server — система управления реляционными базами данных, разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Для реализации логики системы было подготовлено множество запросов на языке Transact-SQL. “Обертка” была написана на языке Python. Связь приложения с базой была реализована с помощью модуля QtSql библиотеки PyQt6, в качестве драйвера выступает драйвер SQL Server ODBC.

Python - высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как C или C++.

Графический интерфейс для системы был так же написан на языке Python с использованием библиотеки PyQt6.

PyQt — набор расширений (биндингов) графического фреймворка Qt для языка программирования Python, выполненный в виде расширения Python.

PyQt разработан британской компанией Riverbank Computing. PyQt работает на всех платформах, поддерживаемых Qt: Linux и другие UNIX-подобные ОС, macOS и Windows. Существует 3 версии: PyQt6, PyQt5 и PyQt4, поддерживающие соответствующие версии Qt. PyQt распространяется под лицензиями GPL (2 и 3 версии) и коммерческой.

В качестве среды разработки был использован многофункциональный инструмент Visual Studio Code.

Visual Studio Code (VS Code) — текстовый редактор, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией.

## Скриншоты системы

### Авторизация и регистрация

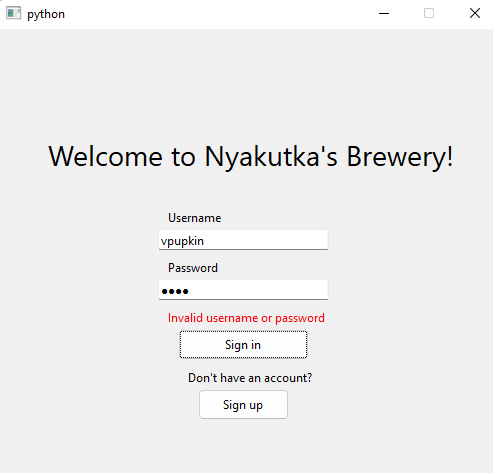
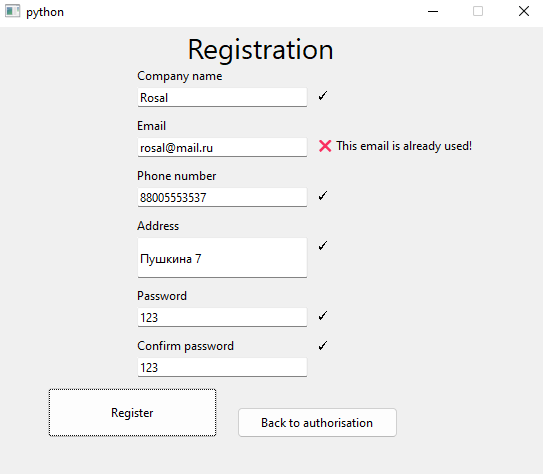
 

Рисунок 1. Неуспешная попытка авторизации Рисунок 2. Неуспешная попытка регистрации

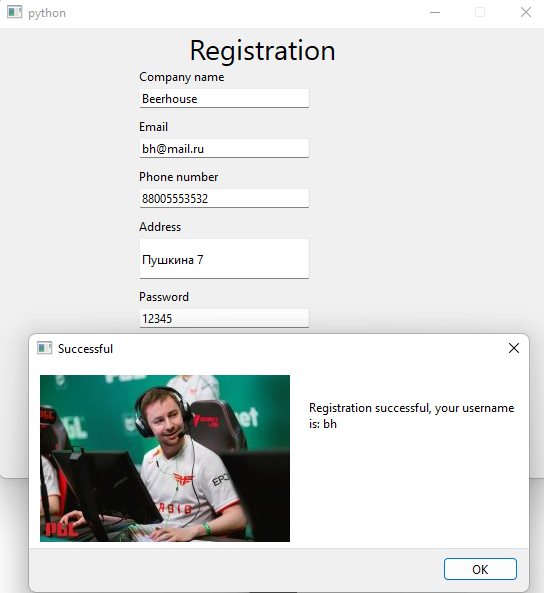


Рисунок 3. Успешная регистрация

### Окно администратора ИС

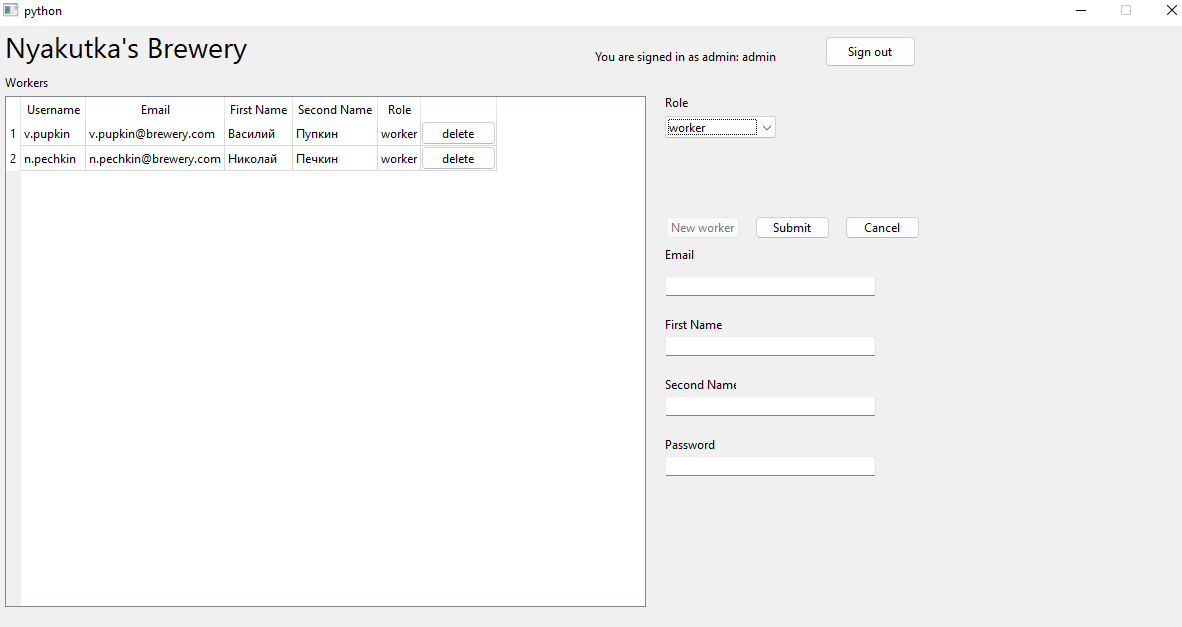


Рисунок 4. Окно администратора системы в режиме создания учетной записи рабочего

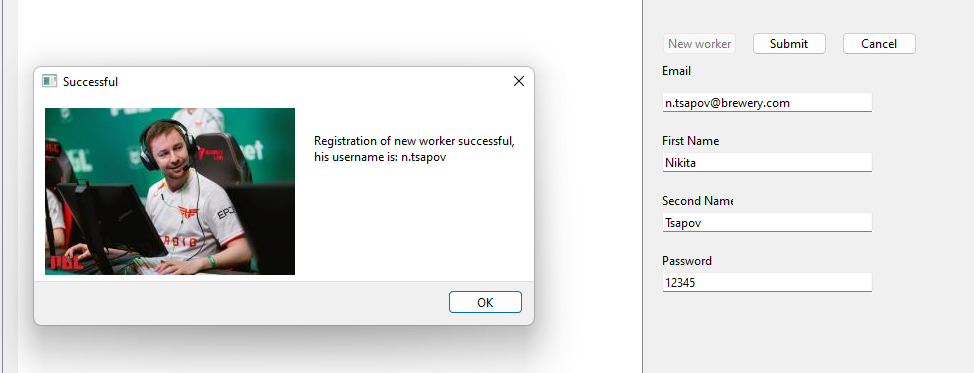


Рисунок 5. Успешное создание учетной записи рабочего

### Окно рабочего завода (Менеджер по продажам)

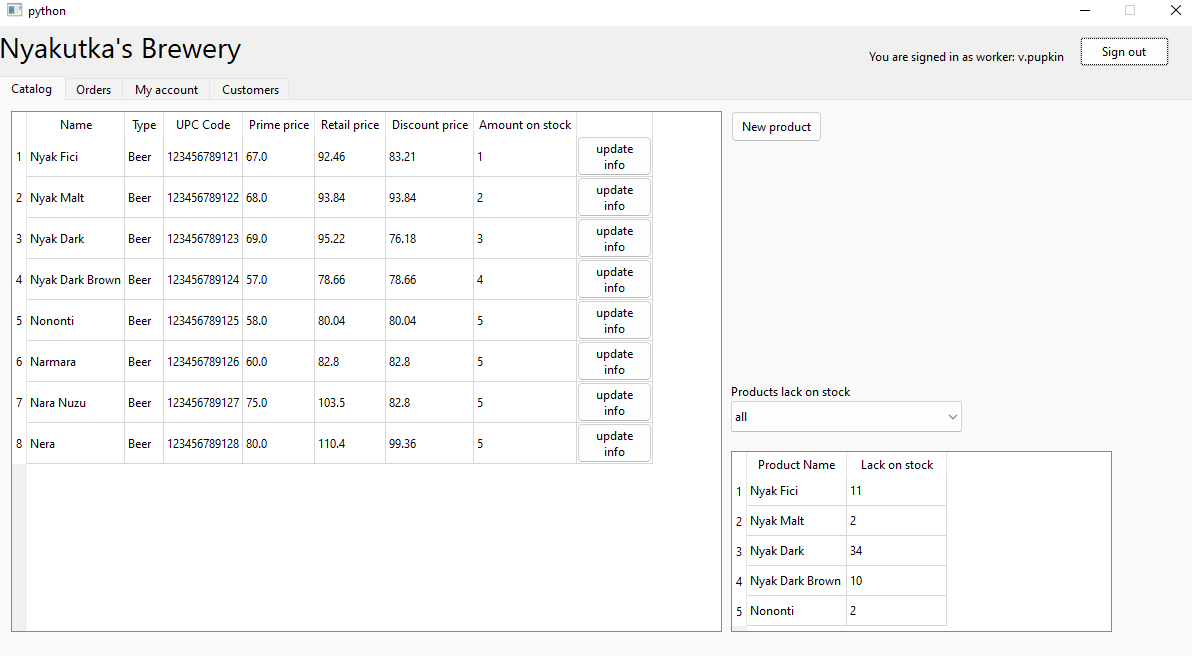


Рисунок 6. Окно рабочего завода, вкладка Catalog

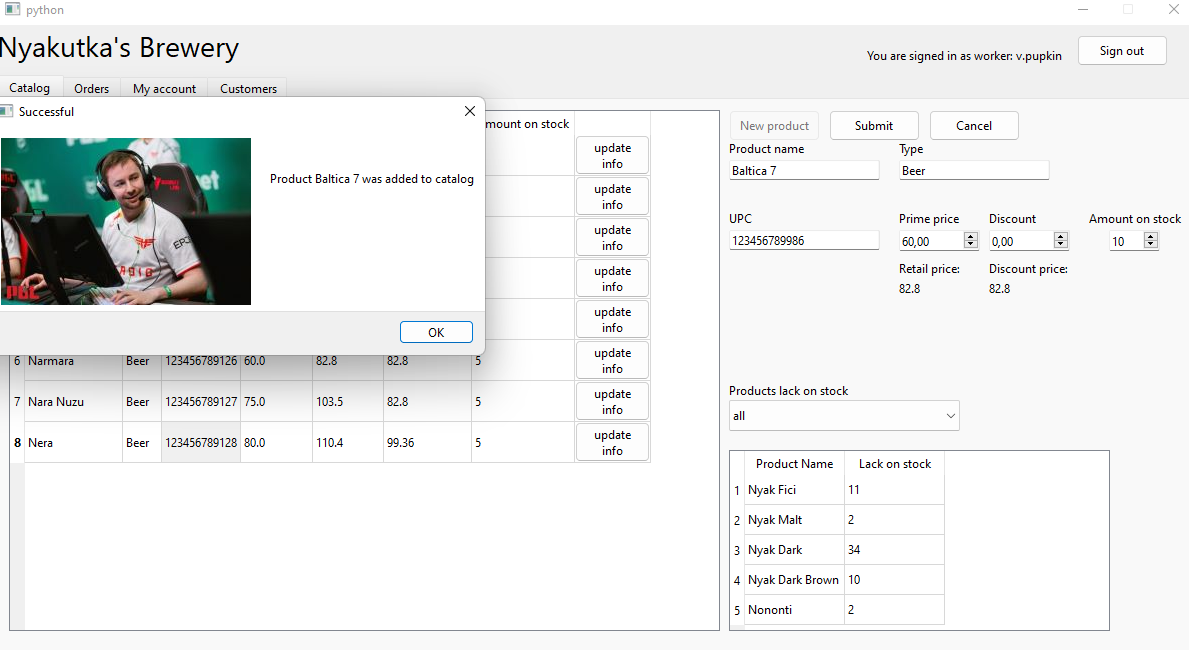


Рисунок 7. Успешное добавление нового продукта в базу

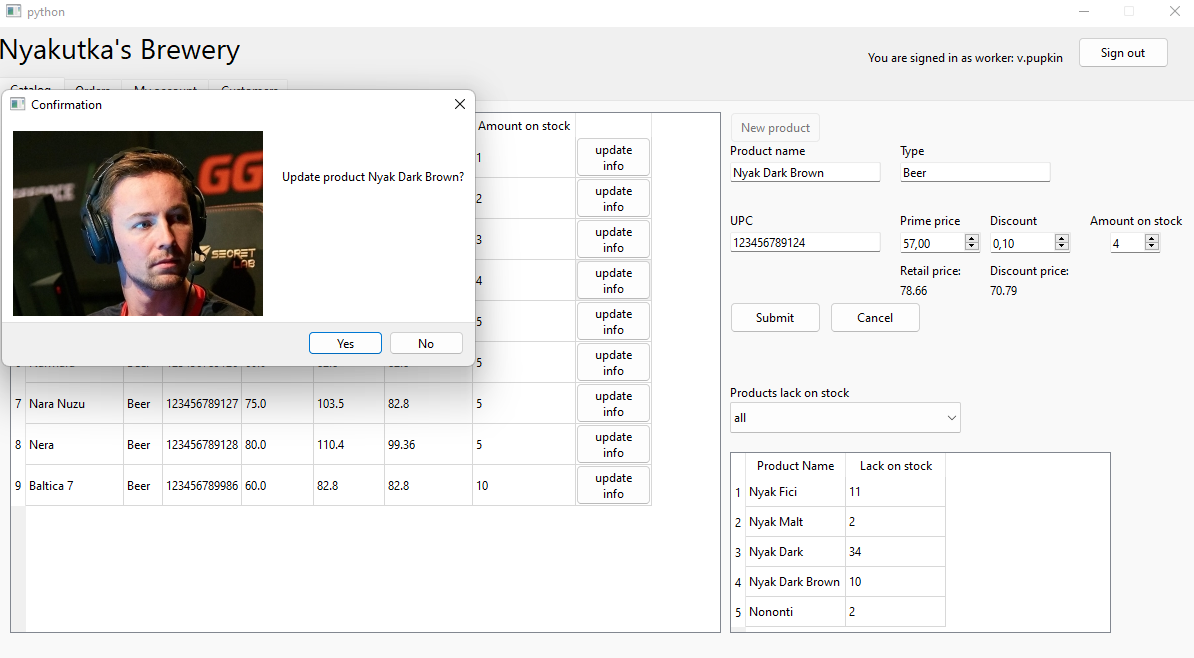


Рисунок 8. Обновление данных о продукте, запрос на подтверждение

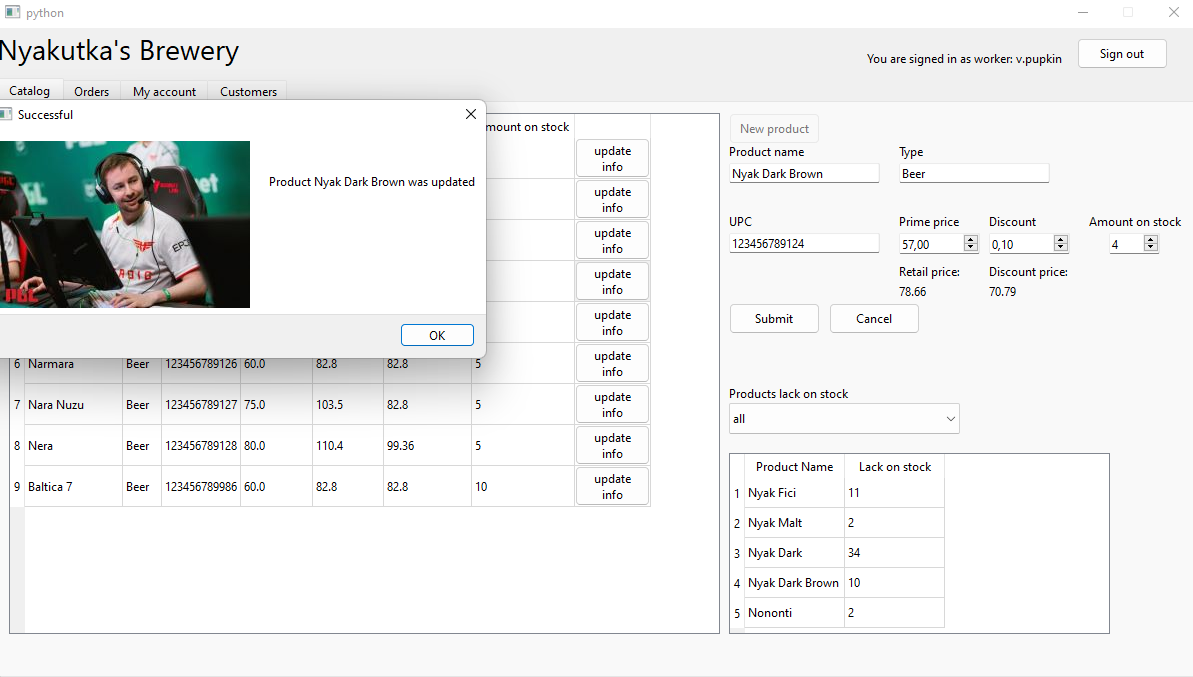


Рисунок 9. Обновление данных о продукте прошло успешно

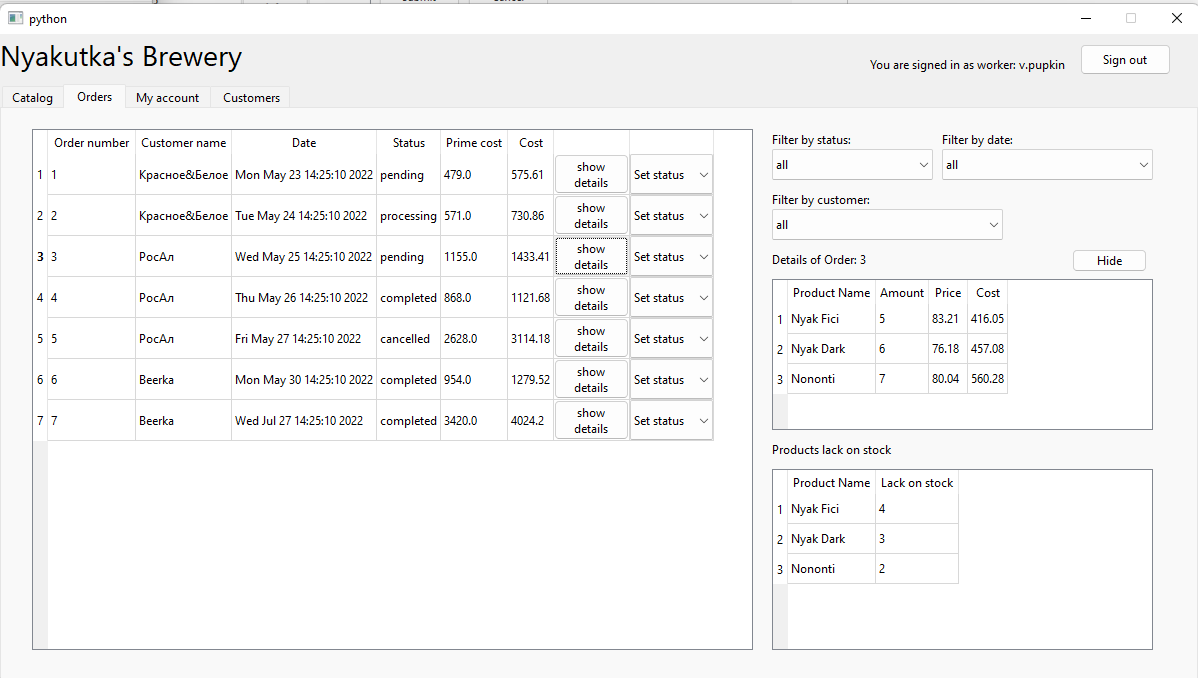


Рисунок 10. Окно рабочего завода, вкладка Orders

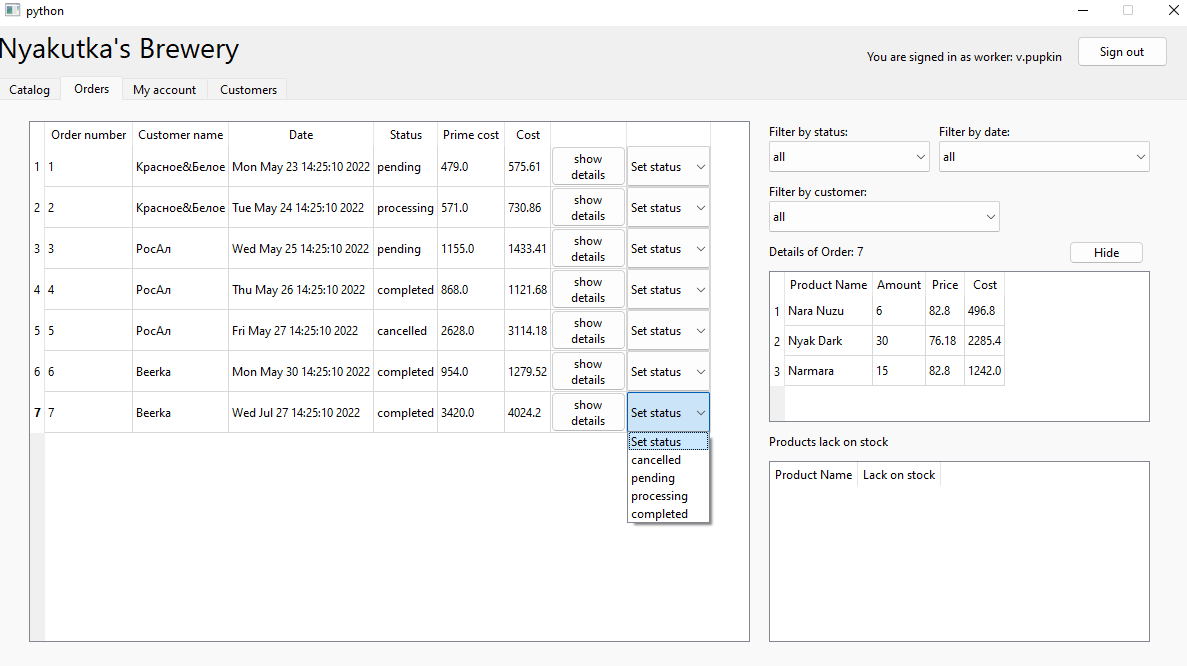


Рисунок 11. Окно рабочего завода, вкладка Catalog, изменение статуса заказа

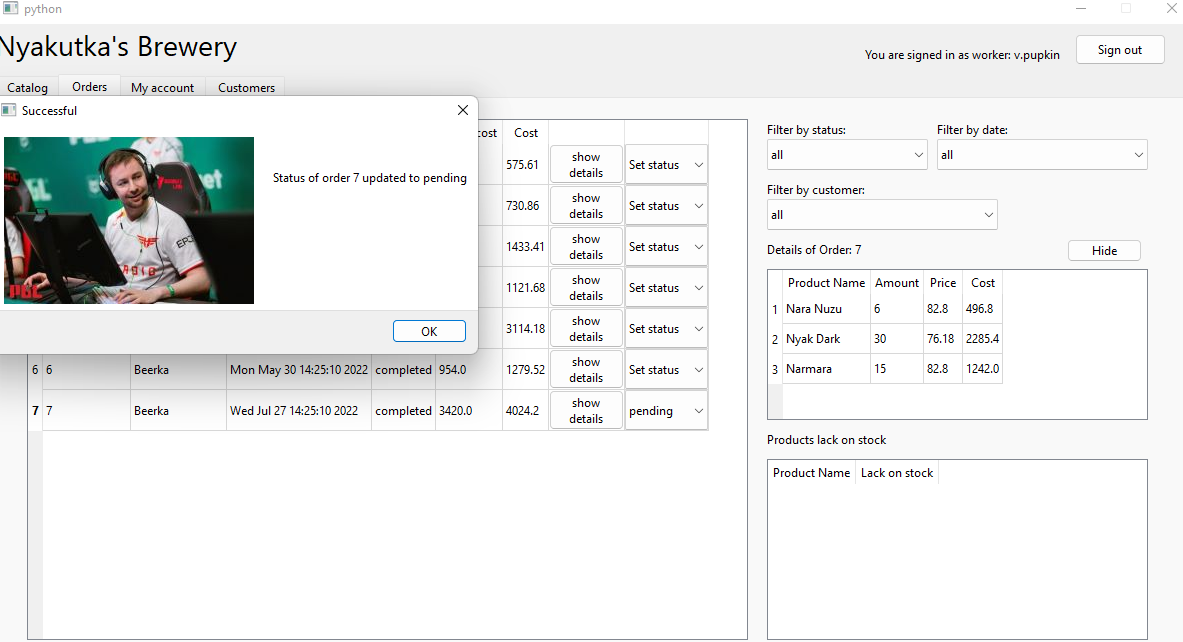


Рисунок 12. Окно рабочего завода, вкладка Catalog, изменение статуса заказа прошло успешно

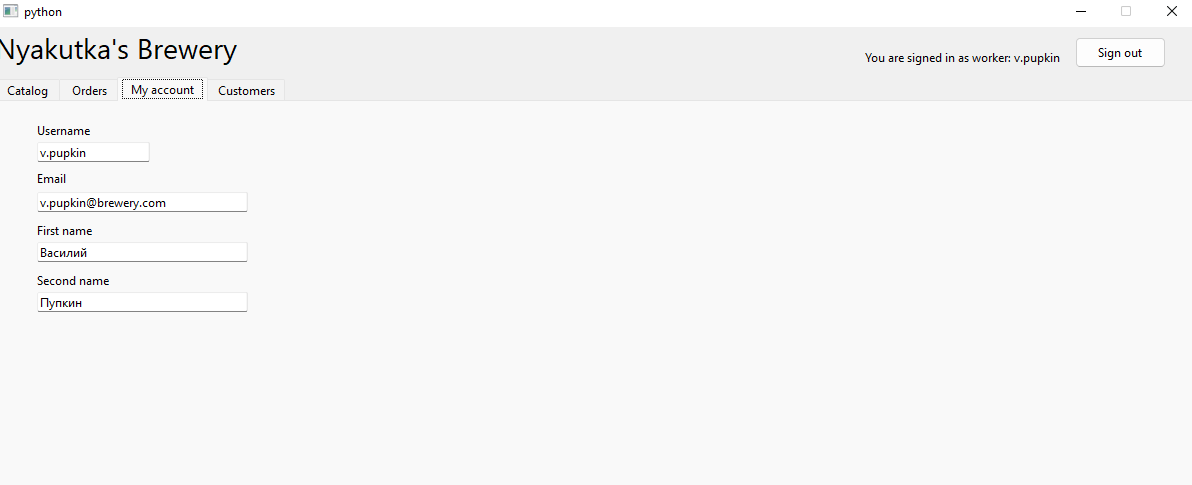


Рисунок 13. Окно рабочего завода, вкладка My account

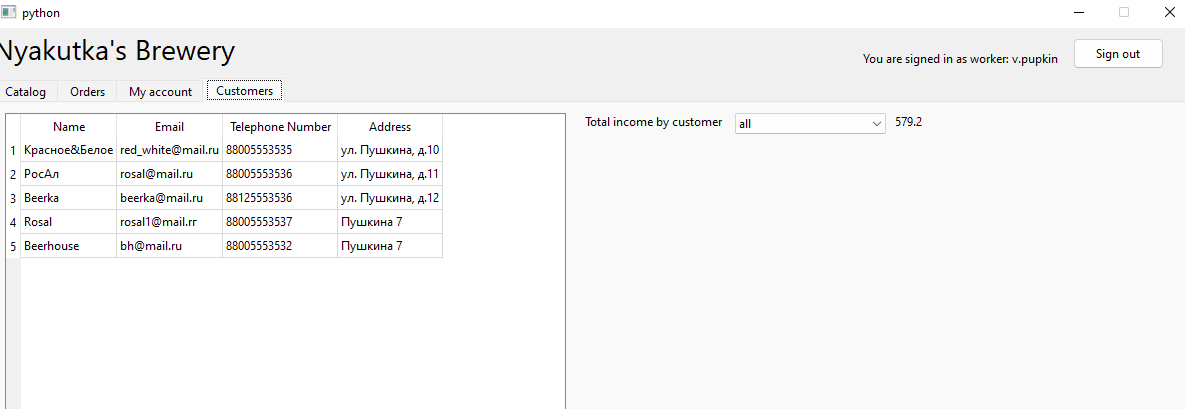


Рисунок 14. Окно рабочего завода, вкладка Customers

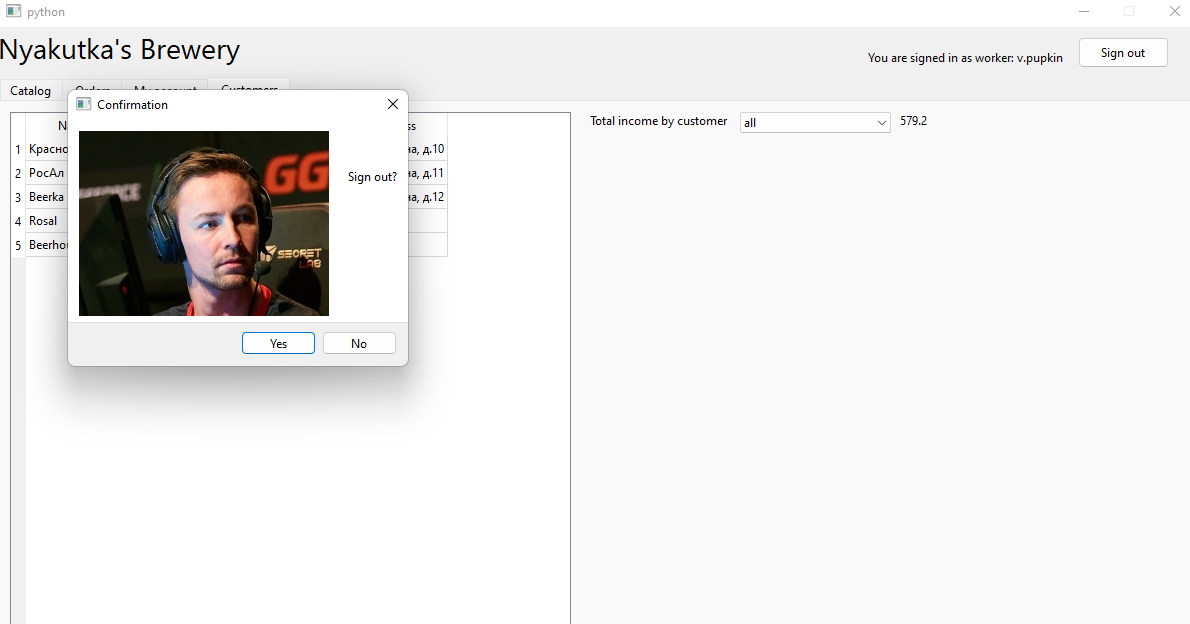


Рисунок 15. Подтверждение выхода из системы

### Окно менеджера по закупкам (розничные магазины)

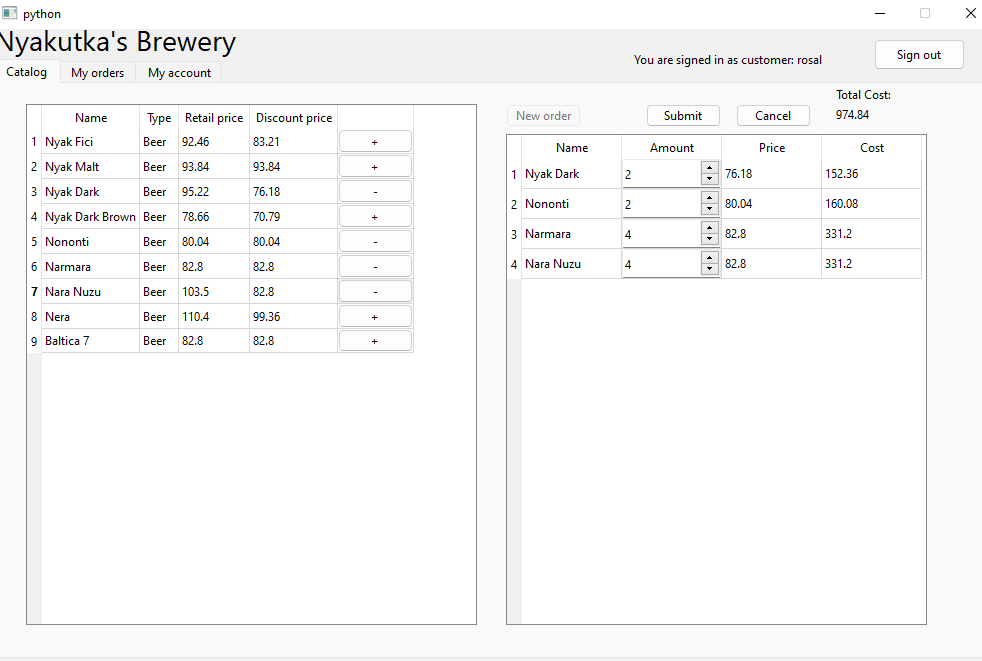


Рисунок 16. Окно пользователя, вкладка Catalog, формирование нового заказа

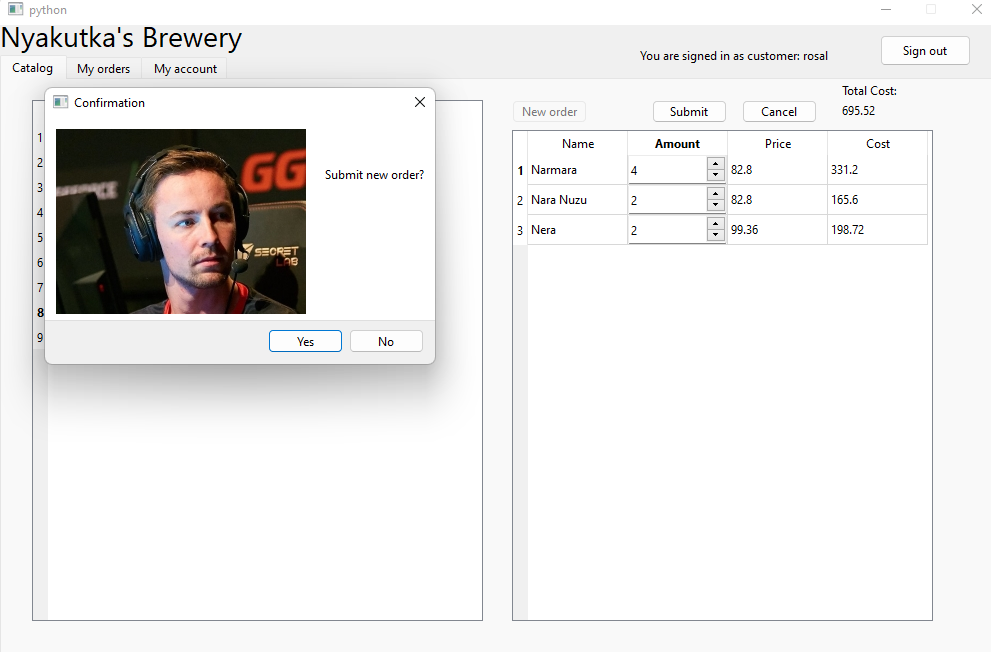


Рисунок 17. Окно пользователя, вкладка Catalog, формирование нового заказа, подтверждение

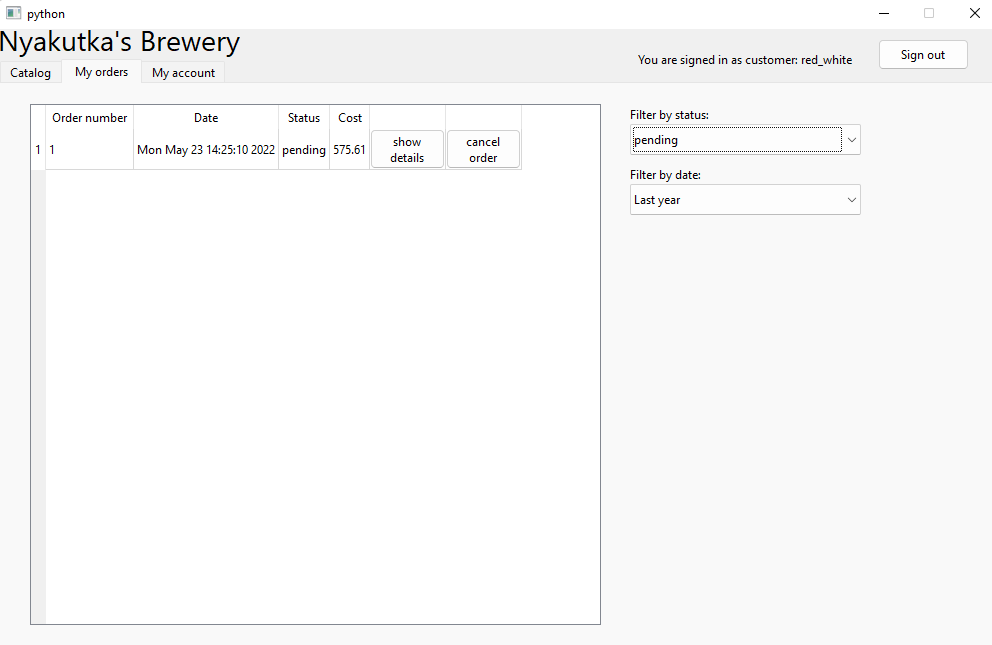


Рисунок 18. Окно пользователя, вкладка My orders, филътры по статусу заказа и по дате

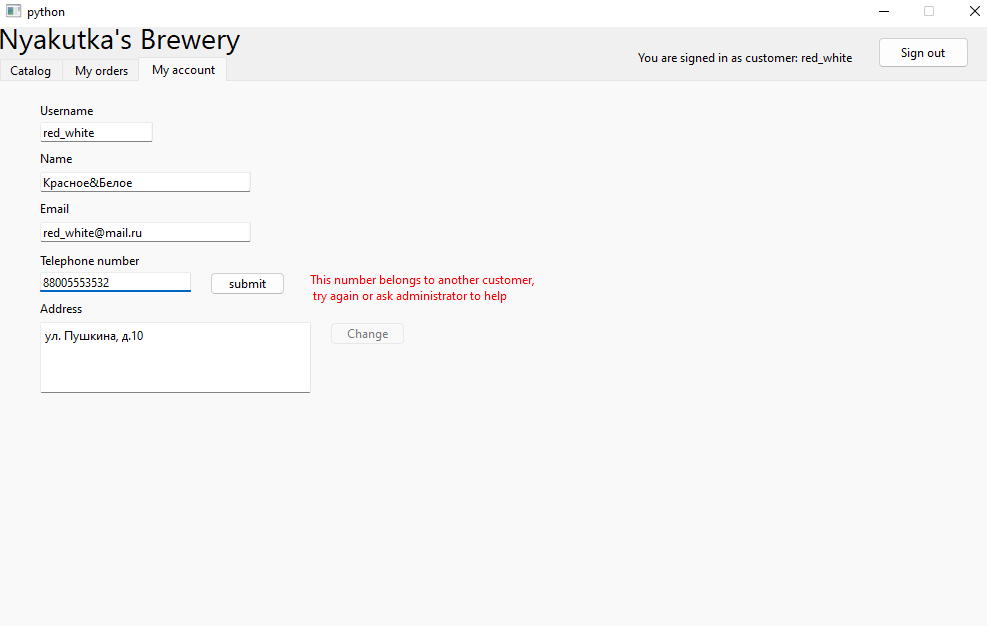


Рисунок 19. Окно пользователя, вкладка My account, неуспешная попытка изменить данные

## Заключение

Задача была решена в полной мере, а именно:

1. Сформирован анализ предметной области, где были объявлены все функции и особенности приложения
2. Составлена структура базы данных
3. Реализованы запросы к базе данных
4. Создано приложение с графическим интерфейсом пользователя

В процессе написания запросов убедился, что хоть инструменты ORM во многих случаях удобны, но существуют такие задачи, которые требуют сырых запросов на языке SQL. Так же я познакомился с инструментами для создания графического интерфейса, которые в начале работы давались мне нелегко.

Ссылка на код проекта в github: <https://github.com/Nyakutka/brewery>

## Список литературы

1. Введение в MS SQL Server и T-SQL // METANIT URL: https://metanit.com/sql/sqlserver/1.1.php (дата обращения: 21.10.2022).
2. Справочник по Transact-SQL (ядро СУБД) // Microsoft learn URL: https://learn.microsoft.com/ru-ru/sql/t-sql/language-reference?view=sql-server-ver16 (дата обращения: 21.10.2022).
3. Qt for Python // PyQt Documentation URL: https://doc.qt.io/qtforpython/ (дата обращения: 21.10.2022).

## Приложение (скрипт создания бд)

USE [master]

GO

/\*\*\*\*\*\* Object:  Database [brewery]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

CREATE DATABASE [brewery]

 CONTAINMENT = NONE

 ON  PRIMARY

( NAME = N'brewery', FILENAME = N'D:\Programs\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\brewery.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )

 LOG ON

( NAME = N'brewery\_log', FILENAME = N'D:\Programs\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\brewery\_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )

 WITH CATALOG\_COLLATION = DATABASE\_DEFAULT

GO

ALTER DATABASE [brewery] SET COMPATIBILITY\_LEVEL = 150

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

EXEC [brewery].[dbo].[sp\_fulltext\_database] @action = 'enable'

end

GO

ALTER DATABASE [brewery] SET ANSI\_NULL\_DEFAULT OFF

GO

ALTER DATABASE [brewery] SET ANSI\_NULLS OFF

GO

ALTER DATABASE [brewery] SET ANSI\_PADDING OFF

GO

ALTER DATABASE [brewery] SET ANSI\_WARNINGS OFF

GO

ALTER DATABASE [brewery] SET ARITHABORT OFF

GO

ALTER DATABASE [brewery] SET AUTO\_CLOSE OFF

GO

ALTER DATABASE [brewery] SET AUTO\_SHRINK OFF

GO

ALTER DATABASE [brewery] SET AUTO\_UPDATE\_STATISTICS ON

GO

ALTER DATABASE [brewery] SET CURSOR\_CLOSE\_ON\_COMMIT OFF

GO

ALTER DATABASE [brewery] SET CURSOR\_DEFAULT  GLOBAL

GO

ALTER DATABASE [brewery] SET CONCAT\_NULL\_YIELDS\_NULL OFF

GO

ALTER DATABASE [brewery] SET NUMERIC\_ROUNDABORT OFF

GO

ALTER DATABASE [brewery] SET QUOTED\_IDENTIFIER OFF

GO

ALTER DATABASE [brewery] SET RECURSIVE\_TRIGGERS OFF

GO

ALTER DATABASE [brewery] SET  ENABLE\_BROKER

GO

ALTER DATABASE [brewery] SET AUTO\_UPDATE\_STATISTICS\_ASYNC OFF

GO

ALTER DATABASE [brewery] SET DATE\_CORRELATION\_OPTIMIZATION OFF

GO

ALTER DATABASE [brewery] SET TRUSTWORTHY OFF

GO

ALTER DATABASE [brewery] SET ALLOW\_SNAPSHOT\_ISOLATION OFF

GO

ALTER DATABASE [brewery] SET PARAMETERIZATION SIMPLE

GO

ALTER DATABASE [brewery] SET READ\_COMMITTED\_SNAPSHOT OFF

GO

ALTER DATABASE [brewery] SET HONOR\_BROKER\_PRIORITY OFF

GO

ALTER DATABASE [brewery] SET RECOVERY FULL

GO

ALTER DATABASE [brewery] SET  MULTI\_USER

GO

ALTER DATABASE [brewery] SET PAGE\_VERIFY CHECKSUM

GO

ALTER DATABASE [brewery] SET DB\_CHAINING OFF

GO

ALTER DATABASE [brewery] SET FILESTREAM( NON\_TRANSACTED\_ACCESS = OFF )

GO

ALTER DATABASE [brewery] SET TARGET\_RECOVERY\_TIME = 60 SECONDS

GO

ALTER DATABASE [brewery] SET DELAYED\_DURABILITY = DISABLED

GO

ALTER DATABASE [brewery] SET ACCELERATED\_DATABASE\_RECOVERY = OFF

GO

EXEC sys.sp\_db\_vardecimal\_storage\_format N'brewery', N'ON'

GO

ALTER DATABASE [brewery] SET QUERY\_STORE = OFF

GO

USE [brewery]

GO

/\*\*\*\*\*\* Object:  UserDefinedFunction [dbo].[isValidEmail]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--TABLES

CREATE FUNCTION [dbo].[isValidEmail](@EMAIL varchar(30))

RETURNS bit as

BEGIN

    DECLARE @bitRetVal as Bit

    IF (@EMAIL = '' OR @EMAIL NOT LIKE '\_%@\_\_%.\_\_%')

        SET @bitRetVal = 0  -- Invalid

    ELSE

        SET @bitRetVal = 1   -- Valid

    RETURN @bitRetVal

END

GO

/\*\*\*\*\*\* Object:  UserDefinedFunction [dbo].[isValidPhoneNumber]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE FUNCTION [dbo].[isValidPhoneNumber](@PHONE\_NUMBER varchar(30))

RETURNS bit as

BEGIN

    DECLARE @bitRetVal as Bit

    IF (LEN(@PHONE\_NUMBER) != 11 or iSNUMERIC(@PHONE\_NUMBER) = 0)

        SET @bitRetVal = 0  -- Invalid

    ELSE

        SET @bitRetVal = 1   -- Valid

    RETURN @bitRetVal

END

GO

/\*\*\*\*\*\* Object:  Table [dbo].[products]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[products](

    [product\_id] [int] IDENTITY(1,1) NOT NULL,

    [product\_name] [varchar](30) NOT NULL,

    [product\_type] [varchar](30) NOT NULL,

    [upc\_code] [varchar](12) NOT NULL,

    [prime\_price] [decimal](8, 2) NOT NULL,

    [retail\_price]  AS ([prime\_price]\*(1.38)),

    [discount\_price] [decimal](8, 2) NULL,

PRIMARY KEY CLUSTERED

(

    [product\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

 CONSTRAINT [UQ\_PRODUCT\_NAME] UNIQUE NONCLUSTERED

(

    [product\_name] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

 CONSTRAINT [UQ\_PRODUCT\_UPC] UNIQUE NONCLUSTERED

(

    [upc\_code] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[stock]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[stock](

    [product\_id] [int] NOT NULL,

    [amount] [int] NOT NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  View [dbo].[ProductsStockView]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--VIEWS

create view [dbo].[ProductsStockView] as

    select product\_name,

           product\_type,

           upc\_code,

           prime\_price,

           retail\_price,

           discount\_price,

           amount

           from products

                join stock on products.product\_id=stock.product\_id

GO

/\*\*\*\*\*\* Object:  Table [dbo].[customers]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[customers](

    [user\_id] [int] NOT NULL,

    [customer\_id] [int] IDENTITY(1,1) NOT NULL,

    [email] [varchar](30) NOT NULL,

    [customer\_name] [varchar](30) NOT NULL,

    [phone\_number] [varchar](30) NOT NULL,

    [address] [varchar](30) NOT NULL,

PRIMARY KEY CLUSTERED

(

    [customer\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

 CONSTRAINT [UQ\_CUSTOMER\_EMAIL] UNIQUE NONCLUSTERED

(

    [email] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

 CONSTRAINT [UQ\_CUSTOMER\_PHONE\_NUMBER] UNIQUE NONCLUSTERED

(

    [phone\_number] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[orders]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[orders](

    [order\_id] [int] IDENTITY(1,1) NOT NULL,

    [customer\_id] [int] NOT NULL,

    [order\_date] [datetime] NOT NULL,

    [order\_status] [varchar](30) NULL,

    [total\_prime\_cost] [decimal](8, 2) NULL,

    [total\_cost] [decimal](8, 2) NULL,

PRIMARY KEY CLUSTERED

(

    [order\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[order\_details]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[order\_details](

    [order\_id] [int] NOT NULL,

    [product\_id] [int] NOT NULL,

    [amount] [int] NOT NULL,

    [price] [decimal](8, 2) NULL,

    [cost]  AS ([price]\*[amount])

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  View [dbo].[OrdersOrderDetailsCustomersView]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create view [dbo].[OrdersOrderDetailsCustomersView] as

    select orders.order\_id,

       customer\_name,

       order\_date,

       orders.order\_status,

       products.product\_id,

       product\_name,

       amount,

       price,

       cost

        from orders

            join order\_details on orders.order\_id=order\_details.order\_id

            join products on order\_details.product\_id=products.product\_id

            join customers on orders.customer\_id=customers.customer\_id

GO

/\*\*\*\*\*\* Object:  Table [dbo].[discounts]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[discounts](

    [product\_id] [int] NOT NULL,

    [discount] [decimal](3, 2) NOT NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[users]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[users](

    [user\_id] [int] IDENTITY(1,1) NOT NULL,

    [username] [varchar](30) NOT NULL,

    [user\_password] [varbinary](256) NOT NULL,

    [user\_role] [varchar](10) NOT NULL,

PRIMARY KEY CLUSTERED

(

    [user\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

UNIQUE NONCLUSTERED

(

    [username] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[workers]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[workers](

    [user\_id] [int] NOT NULL,

    [worker\_id] [int] IDENTITY(1,1) NOT NULL,

    [email] [varchar](30) NOT NULL,

    [first\_name] [varchar](30) NOT NULL,

    [second\_name] [varchar](30) NOT NULL,

PRIMARY KEY CLUSTERED

(

    [worker\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

 CONSTRAINT [UQ\_WORKER\_EMAIL] UNIQUE NONCLUSTERED

(

    [email] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[orders] ADD  DEFAULT ('pending') FOR [order\_status]

GO

ALTER TABLE [dbo].[customers]  WITH CHECK ADD FOREIGN KEY([user\_id])

REFERENCES [dbo].[users] ([user\_id])

GO

ALTER TABLE [dbo].[discounts]  WITH CHECK ADD FOREIGN KEY([product\_id])

REFERENCES [dbo].[products] ([product\_id])

GO

ALTER TABLE [dbo].[order\_details]  WITH CHECK ADD FOREIGN KEY([order\_id])

REFERENCES [dbo].[orders] ([order\_id])

GO

ALTER TABLE [dbo].[order\_details]  WITH CHECK ADD FOREIGN KEY([product\_id])

REFERENCES [dbo].[products] ([product\_id])

GO

ALTER TABLE [dbo].[orders]  WITH CHECK ADD FOREIGN KEY([customer\_id])

REFERENCES [dbo].[customers] ([customer\_id])

GO

ALTER TABLE [dbo].[stock]  WITH CHECK ADD FOREIGN KEY([product\_id])

REFERENCES [dbo].[products] ([product\_id])

GO

ALTER TABLE [dbo].[workers]  WITH CHECK ADD FOREIGN KEY([user\_id])

REFERENCES [dbo].[users] ([user\_id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[customers]  WITH CHECK ADD CHECK  (([address]<>''))

GO

ALTER TABLE [dbo].[customers]  WITH CHECK ADD CHECK  (([customer\_name]<>''))

GO

ALTER TABLE [dbo].[customers]  WITH CHECK ADD  CONSTRAINT [CK\_CUSTOMER\_EMAIL] CHECK  (([dbo].[isValidEmail]([email])=(1)))

GO

ALTER TABLE [dbo].[customers] CHECK CONSTRAINT [CK\_CUSTOMER\_EMAIL]

GO

ALTER TABLE [dbo].[customers]  WITH CHECK ADD  CONSTRAINT [CK\_CUSTOMER\_PHONE\_NUMBER] CHECK  (([dbo].[isValidPhoneNumber]([phone\_number])=(1)))

GO

ALTER TABLE [dbo].[customers] CHECK CONSTRAINT [CK\_CUSTOMER\_PHONE\_NUMBER]

GO

ALTER TABLE [dbo].[products]  WITH CHECK ADD CHECK  (([product\_name]<>''))

GO

ALTER TABLE [dbo].[products]  WITH CHECK ADD  CONSTRAINT [CK\_PRODUCT\_UPC] CHECK  ((len([upc\_code])=(12)))

GO

ALTER TABLE [dbo].[products] CHECK CONSTRAINT [CK\_PRODUCT\_UPC]

GO

ALTER TABLE [dbo].[users]  WITH CHECK ADD CHECK  (([username]<>''))

GO

ALTER TABLE [dbo].[users]  WITH CHECK ADD  CONSTRAINT [CK\_USER\_PASSWORD] CHECK  (([user\_password]<>''))

GO

ALTER TABLE [dbo].[users] CHECK CONSTRAINT [CK\_USER\_PASSWORD]

GO

ALTER TABLE [dbo].[workers]  WITH CHECK ADD CHECK  (([first\_name]<>''))

GO

ALTER TABLE [dbo].[workers]  WITH CHECK ADD CHECK  (([second\_name]<>''))

GO

ALTER TABLE [dbo].[workers]  WITH CHECK ADD  CONSTRAINT [CK\_WORKER\_EMAIL] CHECK  (([dbo].[isValidEmail]([email])=(1)))

GO

ALTER TABLE [dbo].[workers] CHECK CONSTRAINT [CK\_WORKER\_EMAIL]

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[DeleteWorker]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--DELETE--

create   procedure [dbo].[DeleteWorker]

    @user\_id int

as

    DELETE from users

        where user\_id=@user\_id

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[InsertCustomer]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create   procedure [dbo].[InsertCustomer]

    @customer\_name varchar(30),

    @email varchar(30),

    @phone\_number varchar(30),

    @address varchar(30),

    @user\_password varchar(30)

as

    BEGIN TRANSACTION;

        INSERT INTO users(username,

                user\_password,

                user\_role)

            VALUES (left(@email, charindex('@', @email) - 1),

                    HASHBYTES('SHA2\_256', @user\_password),

                    'customer')

        IF (@@error <> 0)

            ROLLBACK

        INSERT INTO customers(user\_id,

                email,

                customer\_name,

                phone\_number,

                address)

            VALUES ((select user\_id from users

                    where username=left(@email,  charindex('@', @email) - 1)),

                @email,

                @customer\_name,

                @phone\_number,

                @address

            )

    COMMIT TRANSACTION;

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[InsertOrder]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create   procedure [dbo].[InsertOrder]

    @customer\_id INT,

    @order\_date DATETIME

as

    INSERT INTO orders(customer\_id, order\_date)

        VALUES (@customer\_id, @order\_date)

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[InsertOrder\_details]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create   procedure [dbo].[InsertOrder\_details]

    @order\_id INT,

    @product\_id INT,

    @amount INT

as

    INSERT INTO order\_details(order\_id, product\_id, amount)

        VALUES (@order\_id, @product\_id, @amount)

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[InsertProduct]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

---

--INSERT--

create   procedure [dbo].[InsertProduct]

    @product\_name varchar(30),

    @product\_type varchar(30),

    @upc\_code varchar(12),

    @prime\_price decimal(8, 2),

    @discount decimal(8, 2),

    @amount int

as

BEGIN TRANSACTION

    INSERT INTO products(product\_name, product\_type, upc\_code, prime\_price)

        VALUES (@product\_name, @product\_type, @upc\_code, @prime\_price)

    Declare @product\_id INT

    Set @product\_id=(select product\_id from products where product\_name=@product\_name)

    EXEC UpdateDiscount @product\_id, @discount

    EXEC UpdateAmountOnStock @product\_id, @amount

    IF (@@error <> 0)

        ROLLBACK

COMMIT

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[InsertWorker]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

---

create   procedure [dbo].[InsertWorker]

    @email varchar(30),

    @first\_name varchar(30),

    @second\_name varchar(30),

    @user\_password varchar(30)

as

BEGIN

BEGIN TRANSACTION

    INSERT INTO users(username, user\_password, user\_role)

        VALUES (left(@email, charindex('@', @email) - 1), HASHBYTES('SHA2\_256', @user\_password), 'worker');

    INSERT INTO workers(user\_id, email, first\_name, second\_name)

        VALUES (

            (select user\_id from users

                where username=left(@email,  charindex('@', @email) - 1)),

            @email,

            @first\_name,

            @second\_name

        )

COMMIT

END

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowAllIncome]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

---

create   procedure [dbo].[ShowAllIncome]

as

select sum(total\_cost - total\_prime\_cost) as income

    from Orders

where order\_status='completed'

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowAllOrders]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--

create   procedure [dbo].[ShowAllOrders]

as

select order\_id,

       customer\_name,

       order\_date,

       order\_status,

       total\_prime\_cost,

       total\_cost

from Orders

    join customers

        on orders.customer\_id=customers.customer\_id

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowAllOrders\_Period\_ForWorker]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--

create   procedure [dbo].[ShowAllOrders\_Period\_ForWorker]

    @period int

as

select order\_id,

       customer\_name,

       order\_date,

       order\_status,

       total\_prime\_cost,

       total\_cost

from Orders

    join customers

        on orders.customer\_id=customers.customer\_id

where DATEDIFF(day, order\_date, GETDATE()) < @period

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowCustomers]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

---

create   procedure [dbo].[ShowCustomers]

as

select customer\_name,

    email,

    phone\_number,

    address

from customers

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowIncomeByCustomer\_name]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

---

create   procedure [dbo].[ShowIncomeByCustomer\_name]

    @customer\_name varchar(30)

as

select sum(total\_cost - total\_prime\_cost) as income

    from Orders

        join customers

            on orders.customer\_id=customers.customer\_id

where order\_status='completed' and customer\_name=@customer\_name

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowLackProducts]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--

create   procedure [dbo].[ShowLackProducts]

as

select product\_name,

       abs(amount\_on\_stock - ordered) as lack\_on\_stock

from (select product\_name,

             sum(OrdersOrderDetailsCustomersView.amount) as ordered,

             stock.amount as amount\_on\_stock

      from OrdersOrderDetailsCustomersView

          join stock

              on OrdersOrderDetailsCustomersView.product\_id=stock.product\_id

      where order\_status != 'completed'

      group by product\_name, stock.amount)

as a

where (amount\_on\_stock - ordered) < 0

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowLackProductsByCustomer\_name]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

---

create   procedure [dbo].[ShowLackProductsByCustomer\_name]

    @customer\_name varchar(30)

as

select product\_name,

    abs(amount\_on\_stock - ordered) as lack\_on\_stock

from

    (select product\_name,

        sum(OrdersOrderDetailsCustomersView.amount) as ordered,

        stock.amount as amount\_on\_stock

    from OrdersOrderDetailsCustomersView join stock on OrdersOrderDetailsCustomersView.product\_id=stock.product\_id

    where order\_status != 'completed' and customer\_name = @customer\_name

    group by product\_name, stock.amount)

as a

where (amount\_on\_stock - ordered) < 0

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowLackProductsByOrder\_id]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

---

create   procedure [dbo].[ShowLackProductsByOrder\_id]

    @order\_id int

as

select product\_name,

    abs(amount\_on\_stock - ordered) as lack\_on\_stock

from

    (select product\_name,

        sum(OrdersOrderDetailsCustomersView.amount) as ordered,

        stock.amount as amount\_on\_stock

    from OrdersOrderDetailsCustomersView

        join stock on OrdersOrderDetailsCustomersView.product\_id=stock.product\_id

    where order\_status != 'completed' and order\_id = @order\_id

group by product\_name, stock.amount)

as a

where (amount\_on\_stock - ordered) < 0

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowOrderDetails]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

---

create   procedure [dbo].[ShowOrderDetails]

@order\_id int

as

select product\_name,

        amount,

        price,

        cost

    from OrdersOrderDetailsCustomersView

    where order\_id = @order\_id

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowOrdersByCustomer\_id\_ForCustomer]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--

create   procedure [dbo].[ShowOrdersByCustomer\_id\_ForCustomer]

    @customer\_id int

as

select order\_id,

    order\_date,

    order\_status,

    total\_cost

from Orders

where customer\_id = @customer\_id

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowOrdersByCustomer\_id\_Period\_ForCustomer]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

---

create   procedure [dbo].[ShowOrdersByCustomer\_id\_Period\_ForCustomer]

    @customer\_id int,

    @period int

as

select order\_id,

    order\_date,

    order\_status,

    total\_cost

from Orders

where customer\_id = @customer\_id and DATEDIFF(day, order\_date, GETDATE()) < @period

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowOrdersByCustomer\_idAndOrder\_status\_ForCustomer]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--

create   procedure [dbo].[ShowOrdersByCustomer\_idAndOrder\_status\_ForCustomer]

    @customer\_id int,

    @order\_status varchar(30)

as

select order\_id,

    order\_date,

    order\_status,

    total\_cost

from Orders

where customer\_id = @customer\_id and order\_status = @order\_status

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowOrdersByCustomer\_idAndOrder\_status\_Period\_ForCustomer]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

---

create   procedure [dbo].[ShowOrdersByCustomer\_idAndOrder\_status\_Period\_ForCustomer]

    @customer\_id int,

    @order\_status varchar(30),

    @period int

as

select order\_id,

    order\_date,

    order\_status,

    total\_cost

from Orders

where customer\_id = @customer\_id and order\_status = @order\_status and DATEDIFF(day, order\_date, GETDATE()) < @period

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowOrdersByCustomer\_name\_ForWorker]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--

create   procedure [dbo].[ShowOrdersByCustomer\_name\_ForWorker]

    @customer\_name varchar(30)

as

select order\_id,

       customer\_name,

       order\_date,

       order\_status,

       total\_prime\_cost,

       total\_cost

from Orders

    join customers

        on orders.customer\_id=customers.customer\_id

where customer\_name = @customer\_name

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowOrdersByCustomer\_nameAnd\_Period\_ForWorker]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

---

create   procedure [dbo].[ShowOrdersByCustomer\_nameAnd\_Period\_ForWorker]

    @customer\_name varchar(30),

    @period int

as

select order\_id,

       customer\_name,

       order\_date,

       order\_status,

       total\_prime\_cost,

       total\_cost

from Orders

    join customers

        on orders.customer\_id=customers.customer\_id

where customer\_name = @customer\_name and DATEDIFF(day, order\_date, GETDATE()) < @period

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowOrdersByCustomer\_nameAndOrder\_status\_ForWorker]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--

create   procedure [dbo].[ShowOrdersByCustomer\_nameAndOrder\_status\_ForWorker]

    @customer\_name varchar(30),

    @order\_status varchar(30)

as

select order\_id,

       customer\_name,

       order\_date,

       order\_status,

       total\_prime\_cost,

       total\_cost

from Orders

    join customers

        on orders.customer\_id=customers.customer\_id

where customer\_name = @customer\_name and order\_status = @order\_status

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowOrdersByCustomer\_nameAndOrder\_statusAnd\_Period\_ForWorker]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

---

create   procedure [dbo].[ShowOrdersByCustomer\_nameAndOrder\_statusAnd\_Period\_ForWorker]

    @customer\_name varchar(30),

    @order\_status varchar(30),

    @period int

as

select order\_id,

       customer\_name,

       order\_date,

       order\_status,

       total\_prime\_cost,

       total\_cost

from Orders

    join customers

        on orders.customer\_id=customers.customer\_id

where customer\_name = @customer\_name and order\_status = @order\_status and DATEDIFF(day, order\_date, GETDATE()) < @period

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowOrdersByOrder\_status\_ForWorker]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--

create   procedure [dbo].[ShowOrdersByOrder\_status\_ForWorker]

    @order\_status varchar(30)

as

select order\_id,

       customer\_name,

       order\_date,

       order\_status,

       total\_prime\_cost,

       total\_cost

from Orders

    join customers

        on orders.customer\_id=customers.customer\_id

where order\_status = @order\_status

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowOrdersByOrder\_statusAnd\_Period\_ForWorker]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

---

create   procedure [dbo].[ShowOrdersByOrder\_statusAnd\_Period\_ForWorker]

    @order\_status varchar(30),

    @period int

as

select order\_id,

       customer\_name,

       order\_date,

       order\_status,

       total\_prime\_cost,

       total\_cost

from Orders

    join customers

        on orders.customer\_id=customers.customer\_id

where order\_status = @order\_status and DATEDIFF(day, order\_date, GETDATE()) < @period

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowProductsBase]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--PROCEDURES

---Просмотр базы товаров

create   procedure [dbo].[ShowProductsBase]

as

select product\_name,

        product\_type,

        upc\_code,

        prime\_price,

        retail\_price,

        discount\_price,

        amount

from ProductsStockView

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowProductsCatalog]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

---

create   procedure [dbo].[ShowProductsCatalog]

as

select product\_name,

        product\_type,

        retail\_price,

        discount\_price

from ProductsStockView

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowWorkers]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

---

---

---

create   procedure [dbo].[ShowWorkers]

as

select username,

    email,

    first\_name,

    second\_name,

    user\_role

from workers

    join users on workers.user\_id=users.user\_id

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[ShowWorkersByRole]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

---

create   procedure [dbo].[ShowWorkersByRole]

    @user\_role varchar(10)

as

select username,

    email,

    first\_name,

    second\_name,

    user\_role

from workers

    join users on workers.user\_id=users.user\_id

    where user\_role = @user\_role

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[UpdateAmountOnStock]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create   procedure [dbo].[UpdateAmountOnStock]

    @product\_id int,

    @amount int

as

    UPDATE stock

        set amount = @amount

    where product\_id=@product\_id

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[UpdateCustomer]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create   procedure [dbo].[UpdateCustomer]

    @customer\_id int,

    @phone\_number varchar(30),

    @address varchar(30)

as

    UPDATE customers

        set phone\_number = @phone\_number,

            address = @address

    where customer\_id=@customer\_id

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[UpdateDiscount]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create   procedure [dbo].[UpdateDiscount]

    @product\_id int,

    @discount decimal(3, 2)

as

    UPDATE discounts

        set discount = @discount

    where product\_id=@product\_id

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[UpdateOrderStatus]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create   procedure [dbo].[UpdateOrderStatus]

    @order\_id int,

    @order\_status varchar(30)

as

    UPDATE orders

        set order\_status = @order\_status

    where order\_id=@order\_id

GO

/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[UpdateProduct]    Script Date: 06.12.2022 19:04:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

--UPDATE--

create   procedure [dbo].[UpdateProduct]

    @product\_id int,

    @product\_name varchar(30),

    @product\_type varchar(30),

    @upc\_code varchar(30),

    @prime\_price varchar(30),

    @discount decimal(8, 2),

    @amount int

as

begin TRANSACTION

    UPDATE products

        set product\_name = @product\_name,

            product\_type = @product\_type,

            upc\_code = @upc\_code,

            prime\_price = @prime\_price

    where product\_id=@product\_id

    EXEC UpdateDiscount @product\_id = @product\_id, @discount = @discount

    EXEC UpdateAmountOnStock @product\_id = @product\_id, @amount = @amount

    IF (@@error <> 0)

            ROLLBACK

COMMIT

GO

USE [master]

GO

ALTER DATABASE [brewery] SET  READ\_WRITE

GO