

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных технологий

Кафедра информатики

КУРСОВАЯ РАБОТА

по дисциплине «Управление данными»

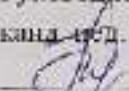
Проектирование и разработка автоматизированной
информационной системы предприятия

Пояснительная записка


ОГУ 09.03.02.3023.858 ПЗ

Руководитель

канд. техн. наук, доцент

 Т.Е. Тлеглова
« 05 » 06 2023 г.

Студент группы 21ИСТ(б)СИЦ

 К.А. Вардугин
« 5 » 11.2023 2023 г.

Оренбург 2023

Утверждаю
заведующий кафедрой информатики
М.А. Токарева
«9» февраля 2023 г.

ЗАДАНИЕ

на выполнение курсовой работы
студенту Варлаутину Кириллу Андреевичу
по направлению подготовки 09.03.02 Информационные системы и технологии
по дисциплине «Управление данными»

1 Тема работы Проектирование и разработка автоматизированной информационной системы предприятия

2 Срок сдачи студентом работы «1» июня 2023 г.

3 Цель и задачи работы

Цель: систематизация, закрепление и расширение полученных теоретических знаний, приобретение практических навыков и умений для творческого применения в решении конкретных задач в области использования современных СУБД.
Задачи: изучение, анализ и решение вопросов, связанных с проектированием и использованием конкретной базы данных, а также администрированием системы управления базами данных.

4 Исходные данные к работе конкретный вариант предметной области, подлежащий исследованию

5 Перечень вопросов, подлежащих разработке

1 Анализ предметной области

1.1 Описание основных сущностей предметной области

2 Информатическое проектирование БД, построение диаграмм БД

3 Обоснование и выбор системы управления БД

3.1 Выбор технологии работы с БД (архитектура БД) и СУБД

3.2 Обоснование выбора средства разработки диалогового приложения

4 Информатическое проектирование БД

5 Код формирования БД на SQL

5.1 Теоретические сведения о SQL-запросах

5.2 Создание запросов к базе данных

6 Приложение для работы с БД

6.1 Структура приложения

6.2 Проектирование графического интерфейса

6.3 Тестирование разработанного приложения

6 Перечень графического (иллюстративного) материала схематические изображения

Дата выдачи и получения задания

Руководитель

«9» февраля 2023 г.

Т.Е. Тлеглова

Студент

«9» февраля 2023 г.

К.А. Варлаутин

Аннотация

Курсовая работа посвящена проектированию и разработке базы данных предметной области «Торговое предприятие» в СУБД SQL Server и Visual Studio.

Актуальность данной темы обусловлено тем, что современные торговые предприятия оперируют большим объемом данных, требующих эффективного хранения, управления и обработки. База данных, разработанная с использованием SQL Server и Visual Studio, может обеспечить надежную и гибкую основу для хранения и анализа данных торгового предприятия, что позволит повысить эффективность управления бизнесом и принимать обоснованные решения. Кроме того, такая работа может дать студенту необходимые знания и навыки для успешной работы с СУБД и инструментами разработки, что может быть полезно в его будущей профессиональной деятельности.

Пояснительная записка состоит из шести разделов.

В первом разделе пояснительной записки производится анализ предметной области, а именно описываются основные сущности данной предметной области.

Во втором разделе пояснительной записки изложены сведения об инфологическом проектировании базы данных и производится построение диаграмм базы данных.

В третьем разделе пояснительной записки обосновываются выбор технологии работы с базой данных и средства разработки диалогового приложения.

В четвертом разделе пояснительной записки рассматривается даталогическое проектирование базы данных.

В пятом разделе пояснительной записки излагаются краткие теоретические сведения о SQL-запросах, а также код формирования базы данных на SQL.

В заключительном, шестом разделе пояснительной записки описывается разработанное приложение для работы с созданной базой данных, а также проводится его тестирование.

Работа содержит 96 листов текста, 67 рисунков, 12 таблиц, 2 приложения.

					ОГУ 09.03.02.3023.858 ПЗ				
Изм.	Лист	№ документа	Подп.	Дата	Проектирование и разработка БД автоматизированной информационной системы предприятия	Лит.	Лист	Листов	
Разраб.		Вардугин К.А.		05.06		К	3	96	
Пров.		Глегинова Т.Е.		05.06					
Н.контр.									
Зав. каф.		Токарева М.А.		05.06			21 ИСТ(6) СИИ		

Содержание

Введение.....	7
1 Анализ предметной области.....	8
1.1 Описание основных сущностей предметной области.....	8
2 Инфологическое проектирование БД, построение диаграмм БД	13
3 Обоснование и выбор системы управления БД	15
3.1 Выбор технологии работы с БД (архитектура БД) и СУБД	15
3.2 Обоснование выбора средств разработки диалогового приложения	16
4 Даталогическое проектирование БД.....	18
5 Код формирования БД на SQL.....	22
5.1 Теоретические сведения о SQL-запросах	22
5.2 Создание запросов к базе данных.....	23
6 Приложение для работы с БД	35
6.1 Структура приложения	35
6.2 Проектирование графического интерфейса	35
6.3 Тестирование разработанного приложения	54
Заключение	66
Список использованных источников	67
Приложение А	68
Приложение Б	95

Введение

В настоящее время автоматизация бизнес-процессов стала неотъемлемой частью современного мира. Это явление охватило многие отрасли экономики, включая торговлю и розничную продажу. Разработка информационной системы для автоматизации процессов продаж парфюмерии и косметики позволит торговой организации эффективно управлять своими ресурсами, контролировать движение товаров на складе и повысить уровень обслуживания клиентов.

Эта информационная система будет включать в себя базу данных, которая будет хранить и обрабатывать информацию об ассортименте продукции, ее поставщиках, сотрудниках магазина и продажах. Такая база данных позволит решать множество задач, таких как учет поставленного и реализованного товара, назначение розничных цен на товары с возможностью автоматического добавления наценки, назначение скидки по дисконтной карте на весь чек целиком и печать товарного чека.

Система обеспечивает автоматический учет движения товаров на складе, что позволяет контролировать наличие товара и его количество в режиме реального времени. Также система позволяет назначать розничные цены на продукцию с учетом наценки, что упрощает управление ценами и повышает точность расчетов.

Для удобства покупателей система предоставляет возможность назначать скидки по дисконтной карте на весь чек целиком. Это повышает уровень удовлетворенности клиентов и стимулирует их к повторным покупкам.

Для печати товарного чека система использует информацию о продуктах, их количестве и стоимости, а также информацию о сумме скидки и общей сумме к оплате. Это позволяет обеспечить быстрое и точное оформление заказа и ускорить процесс обслуживания покупателей.

Для реализации такой информационной системы необходимо использовать современные технологии программирования и базы данных. В данной курсовой работе будет создана база данных и приложение в среде Visual Studio, используя формы на языке программирования C#, которые предоставляют удобный графический интерфейс для взаимодействия с пользователем. торговой организации значительно повысить эффективность своей работы, контролировать движение товаров на складе и улучшить уровень обслуживания клиентов.

Целью данной курсовой работы является освоение и укрепление практических навыков работы с базами данных.

1 Анализ предметной области

Разрабатываемая информационная система предназначена для автоматизации бизнес-процессов торговой организации, специализирующейся на продаже парфюмерии и косметики. База данных, создаваемая для этой системы, позволит эффективно хранить и обрабатывать информацию об ассортименте продукции, ее поставщиках, сотрудниках магазина и продажах.

Основными задачами, решаемыми разрабатываемой системой, являются учет поставленного и реализованного товара, назначение розничных цен на товары с возможностью автоматического добавления наценки, назначение скидки по дисконтной карте на весь чек целиком и печать товарного чека.

Система обеспечивает автоматический учет движения товаров на складе, что позволяет в режиме реального времени контролировать наличие товара и его количество. Также система позволяет назначать розничные цены на продукцию с учетом наценки, что упрощает управление ценами и повышает точность расчетов.

Для удобства покупателей система предоставляет возможность назначать скидки по дисконтной карте на весь чек целиком. Дисконтная карта также позволяет магазину получать дополнительную информацию о покупателях, их предпочтениях и покупательском поведении.

Для печати товарного чека система использует информацию о продуктах, их количестве и стоимости, а также информацию о сумме скидки и общей сумме к оплате.

Все эти функциональные возможности системы позволяют торговой организации оптимизировать работу, увеличить эффективность управления и повысить уровень обслуживания покупателей.

1.1 Описание основных сущностей предметной области

Описание основных сущностей является важной частью проектирования базы данных. Оно помогает определить структуру базы данных, определяя, какие данные будут храниться в таблицах и как эти таблицы будут связаны между собой. Для информационной системы торговой организации, занимающейся продажей парфюмерии и косметики, основные сущности включают товары, поставщиков, сотрудников, заказы и счета. Каждая из этих сущностей имеет свои характеристики и атрибуты, которые необходимо учитывать при создании базы данных.

Таблица 1 – Список сущностей базы данных

№	Название	Тип	Назначение
1	Товары	Стержневой	Хранение данных об ассортименте товаров, подлежащих продаже

Продолжение таблицы 1

2	Поставщики	Стержневой	Хранение информации о компаниях-поставщиках продукции
3	Сотрудники	Стержневой	Хранение личных и рабочих данных о сотрудниках организации
4	Заказы	Ассоциативный	Связующая таблица для хранения данных о заказах покупателей
5	Счета	Ассоциативный	Связующая таблица для хранения данных о продажах и оплате

Таблицы «Товары», «Поставщики» и «Сотрудники» являются стержневыми таблицами, которые содержат основную информацию о соответствующих объектах. Таблицы «Заказы» и «Счета» являются связующими таблицами и несут функцию связи между другими таблицами базы данных.

Для каждой таблицы (сущности) приведем описание ее атрибутов. Атрибут на физическом уровне – это колонки таблицы и выражает определенное свойство объекта (таблица 2-6).

Таблица 2 – Список атрибутов таблицы «Товары»

Ключевое поле	Название	Назначение
ПК (первичный ключ)	№	Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждому товару. Это целое число, т.е. для идентификации каждого товара будет применяться не названия самих товаров, а определенный номер. Этот номер может быть случайным целым числом или счетчик по порядку.

Продолжение таблицы 2

	Код товара	
	Категория	
	Наименование	
	Единица измерения	
	Цена закупочная	
	Наценка, %	
	Цена реализации	
	Количество на складе	
	Срок реализации	
	Код поставщика	

Таблица 3 – Список атрибутов таблицы «Поставщики»

Ключевое поле	Название	Назначение
ПК (первичный ключ)	Код поставщика	Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждому поставщику. Это целое число, т.е. для идентификации каждого поставщика будет применяться не названия самих поставщиков, а определенный номер. Этот номер может быть случайным целым числом или счетчик по порядку.
	Название	
	Юридический адрес	
	Телефон	
	Факс	

Таблица 4 – Список атрибутов таблицы «Сотрудники»

Ключевое поле	Название	Назначение
ПК (первичный ключ)	Код сотрудника	Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждому сотруднику. Это целое число, т.е. для идентификации каждого сотрудника будет применяться не названия самих сотрудников, а определенный номер. Этот номер может быть случайным целым числом или счетчик по порядку.
	Номер паспорта	
	Имя	
	Отчество	
	Фамилия	
	Семейное положение	
	Должность	
	Электронная почта	
	Дата рождения	
	Дата найма	
	Город	
	Почтовый индекс	
	Адрес	
	Контактный телефон	

Таблица 5 – Список атрибутов таблицы «Заказы»

Ключевое поле	Название	Назначение
	Номер чека	
	Код товара	
	Количество	
	Цена	
	Стоимость	

Таблица 6 – Список атрибутов таблицы «Счета»

Ключевое поле	Название	Назначение
ПК (первичный ключ)	Номер чека	Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждому чеку. Это целое число, т.е. для идентификации каждого чека будет применяться не названия самих чеков, а определенный номер. Этот номер может быть случайным целым числом или счетчик по порядку.
	Код сотрудника	
	Дата	
	Дисконтная карта	
	Скидка	
	Сумма к оплате	

2 Инфологическое проектирование БД, построение диаграмм БД

Инфологическая модель является важным инструментом при проектировании баз данных. Она позволяет описать структуру данных и связи между ними, не учитывая аспекты физической реализации в базе данных. Построение инфологической модели является первым этапом при проектировании баз данных и предшествует созданию физической модели. Она помогает определить сущности, их атрибуты и связи между ними, что позволяет более точно определить требования к базе данных и избежать ошибок при ее создании. В данном разделе мы построим инфологическую модель для базы данных, содержащей информацию о товарах, поставщиках, сотрудниках, заказах и счетах, которая позволяет описать структуру данных и связи между ними в данной предметной области.

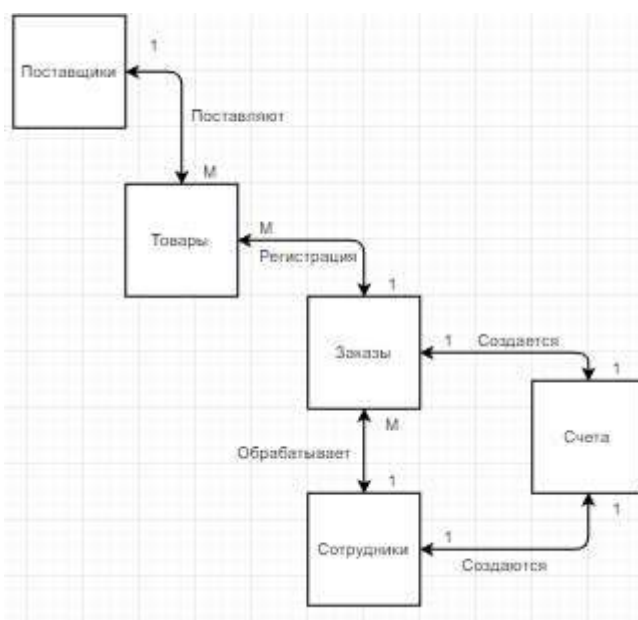


Рисунок 1 – Инфологическая модель базы данных

Для выявленных связей заполним таблицу. Полная информация о связях таблиц базы данных «Торговое предприятие» представлено в таблице 7.

Таблица 7 – Информация о связях таблиц

Название связи	Сущности	Назначение
1 ко многим	Поставщики (один) к Товары (много)	Один поставщик может поставлять много товаров
1 ко многим	Сотрудники (один) к Заказы (много)	Один сотрудник может обрабатывать много заказов

Продолжение таблицы 7

1 ко многим	Заказы (один) к Товары (много)	Много товаров может быть в одном заказе
1 к 1	Заказы (один) к Счета (один)	Один заказ может быть оплачен только одним счетом, а каждый счет может содержать информацию только об одном заказе
1 к 1	Счета (один) к Сотрудники (один)	Каждый счет может быть выставлен только одним сотрудником

Когда созданы отношения (связи) между таблицами, база данных достигла той точки, когда данные в одной таблице начинают зависеть от данных в другой таблице. SQL Server дает возможность увидеть, зависит ли некая таблица от других или нет. Отображение зависимостей можно получить при помощи диаграммы базы данных. Диаграмма базы данных в простейшей форме отображает таблицы (с перечислением атрибутов этих таблиц) и отношения между таблицами. На рисунке 2 показана диаграмма базы данных «Торговое предприятие».

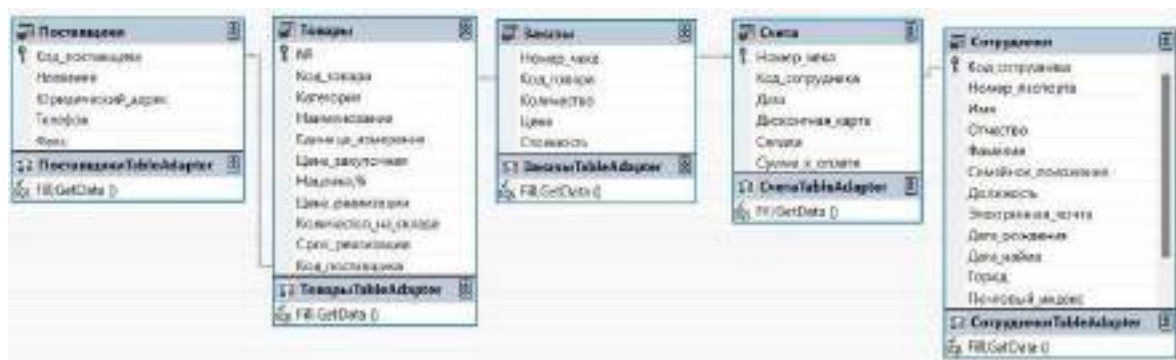


Рисунок 2 – Диаграмма базы данных

3 Обоснование и выбор системы управления БД

3.1 Выбор технологии работы с БД (архитектура БД) и СУБД

СУБД (Система Управления Базами Данных) — это специальное программное обеспечение, которое используется для хранения, организации, управления и обработки информации в базах данных. Она позволяет пользователям создавать, изменять и извлекать данные, а также обеспечивает надежность, целостность и безопасность этих данных.

Существует множество различных СУБД, каждая из которых имеет свои особенности, преимущества и недостатки. В этом разделе мы сравним несколько из них и рассмотрим, почему я выбрал SQL Server Management Studio (SQL SMS) в качестве приложения для работы с базами данных.

MySQL Workbench

MySQL Workbench — это инструмент для работы с базами данных MySQL. Он предоставляет пользователю широкий набор функций для создания, изменения и управления базами данных, включая возможность проектирования схемы базы данных, написания запросов на языке SQL, управления пользователями и многое другое.

Однако, по мнению многих пользователей, интерфейс MySQL Workbench не всегда интуитивно понятен, а также он может быть несколько медленным и неэффективным при работе с большими объемами данных.

PostgreSQL

PostgreSQL — это мощная СУБД с открытым исходным кодом, которая используется для хранения и обработки данных различных типов и объемов. Она предоставляет множество возможностей для управления данными, включая поддержку многопользовательской работы, полнотекстовый поиск, геоданные и многое другое.

Однако, PostgreSQL может быть несколько сложным в использовании, особенно для новичков, которые только начинают работу с базами данных. Также установка и настройка PostgreSQL может быть несколько сложной и затратной по времени.

Oracle SQL Developer

Oracle SQL Developer — это инструмент для работы с базами данных Oracle. Он предоставляет широкий набор функций для работы с базами данных, включая создание, изменение и управление таблицами, написание и выполнение запросов на языке SQL, анализ производительности и многое другое.

Однако, Oracle SQL Developer может быть несколько медленным и громоздким при работе с большими объемами данных. Кроме того, для работы с Oracle SQL Developer необходимо иметь определенные знания и навыки работы с базами данных Oracle.

Microsoft Access

Microsoft Access - это относительно простой инструмент для работы с базами данных, разработанный для использования в Windows-среде. Он позволяет пользователям быстро создавать и изменять базы данных, создавать формы и отчеты для отображения данных, а также использовать мощные инструменты для анализа и управления данными.

Однако, Microsoft Access имеет ограничения по размеру базы данных и производительности, что делает его менее подходящим для работы с большими объемами данных. Кроме того, Microsoft Access работает только в операционных системах Windows и не поддерживает многопользовательский режим работы.

SQL Server Management Studio (SQL SMS)

SQL Server Management Studio (SQL SMS) - это бесплатное приложение для работы с базами данных Microsoft SQL Server. Он предоставляет широкий набор функций для управления базами данных, включая создание, изменение и удаление таблиц, написание и выполнение запросов на языке SQL, анализ производительности и многое другое.

Одним из преимуществ SQL SMS является его удобный и интуитивно понятный интерфейс, который позволяет пользователям легко освоить приложение даже без специальных знаний в области баз данных. Кроме того, SQL SMS поддерживает работу с большими объемами данных и обеспечивает высокую производительность при выполнении запросов.

В целом, каждая из рассмотренных СУБД имеет свои преимущества и недостатки, и выбор технологии для работы с базами данных зависит от конкретных потребностей и задач пользователя. Однако, по моему мнению, SQL Server Management Studio является одним из лучших бесплатных приложений для работы с базами данных, благодаря своей функциональности, удобству использования и высокой производительности.

3.2 Обоснование выбора средств разработки диалогового приложения

Выбор Visual Studio с использованием Windows Forms и языка программирования C# обосновывается следующими преимуществами:

1) Широкие возможности. Visual Studio поддерживает множество языков программирования и технологий, что позволяет создавать различные приложения с разным функционалом. Windows Forms предоставляет широкий набор элементов управления, которые можно использовать для создания интерфейса приложения, а язык C# имеет множество возможностей для написания эффективного и читабельного кода.

2) Простота использования. Visual Studio имеет интуитивно понятный интерфейс и предоставляет все необходимые инструменты для создания приложения. Windows Forms также прост в использовании и предоставляет готовые элементы управления, что упрощает создание интерфейса для пользователей без

необходимости вручную писать каждый компонент. Язык C# также имеет простой синтаксис, который легко понять и использовать даже новичкам в программировании.

3) Большое сообщество. Visual Studio, Windows Forms и C# имеют большое сообщество пользователей, что обеспечивает доступ к огромному количеству документации, учебных пособий и форумов для получения помощи и поддержки при разработке приложения. Также это значит, что существует множество сторонних библиотек и инструментов, которые можно использовать для расширения функционала приложения.

4) Высокая производительность. Использование языка C# позволяет создавать эффективный код, который работает быстро и потребляет меньше ресурсов. Windows Forms также обеспечивает высокую производительность благодаря использованию нативных компонентов операционной системы.

5) Широкая совместимость. Созданные в Visual Studio приложения могут работать на широком диапазоне операционных систем и платформ, включая Windows, MacOS и Linux. Windows Forms также обеспечивает совместимость с различными версиями Windows и .NET Framework.

В целом, выбор Visual Studio с использованием Windows Forms и языка программирования C# обеспечивает широкие возможности для создания диалогового приложения с простым и интуитивно понятным интерфейсом, высокой производительностью и совместимостью с различными платформами.

4 Дatalogическое проектирование БД

Дatalogическая модель является важным этапом проектирования базы данных, она описывает структуру данных на уровне конкретных таблиц и их связей.

В дatalogической модели определяются атрибуты таблиц, их типы данных, а также связи между таблицами. Кроме того, дatalogическая модель содержит информацию о первичных и внешних ключах, что позволяет обеспечить целостность данных в базе.

В этом разделе мы составим дatalogическую модель на основе инфологической модели, рассмотренной ранее, для уточнения структуры таблиц и связей между ними.

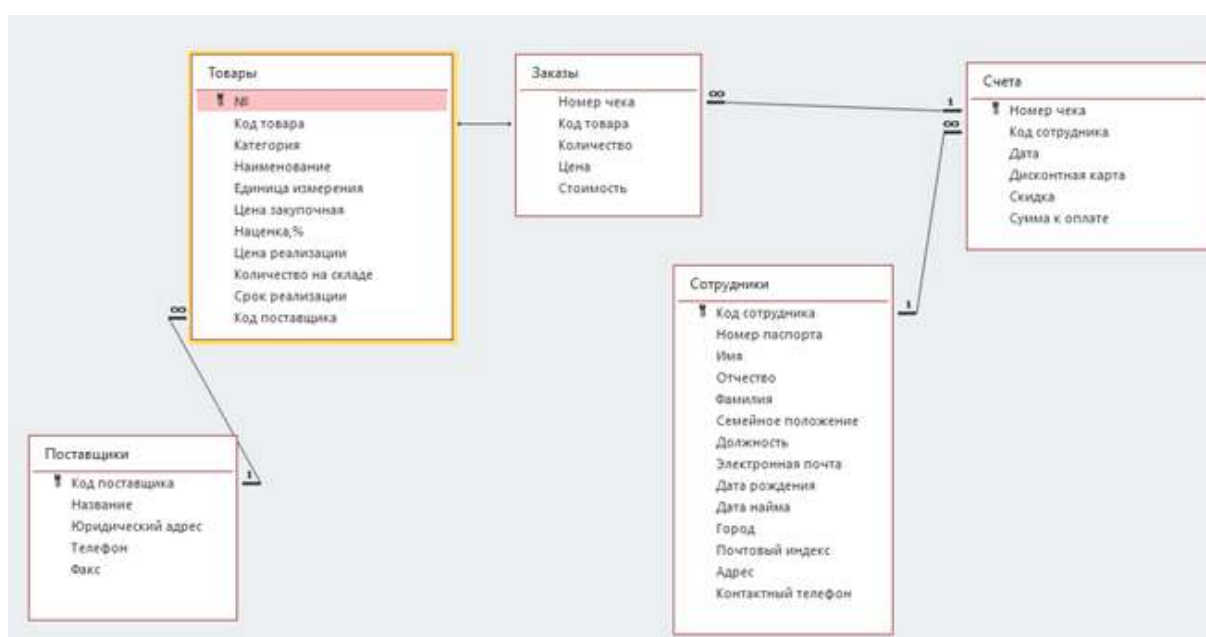


Рисунок 3 – Диаграмма базы данных

Дatalogическая модель БД представляется в виде набора таблиц специальной формы, в которых указываются наименование атрибута, идентификатор, тип, длина, формат, ограничения (8-12).

Таблица 8 – Список атрибутов таблицы «Поставщики»

№	Название	Идентификатор	Тип	Не пусто	Ограничение
1	Код_поставщика	Код_поставщика	Счетчик	Да	ПК (первичный ключ)
2	Название	Название	Короткий текст	Нет	

Продолжение таблицы 8

3	Юридический_адрес	Юридический_адрес	Короткий текст	Нет	
4	Телефон	Телефон	Короткий текст	Нет	\(000\)000\-\00\-\00;;*
5	Факс	Факс	Короткий текст	Нет	\(000\)000\-\00\-\00;;*

Таблица 9 – Список атрибутов таблицы «Товары»

№	Название	Идентификатор	Тип	Не пусто	Ограничение
1	№	№	Счетчик	Да	ПК (первичный ключ)
2	Код_товара	Код_товара	Числовой	Да	ВК (внешний ключ)
3	Категория	Категория	Короткий текст	Нет	
4	Наименование	Наименование	Короткий текст	Нет	
5	Единица_измерения	Единица_измерения	Короткий текст	Нет	
6	Цена_закупочная	Цена_закупочная	Денежный	Нет	# ##0,00»р.»;-# ##0,00»р.»
7	Наценка	Наценка	Числовой	Нет	Проценты (число от 0 до 1). Например 0.3 это будет 30%
8	Цена_реализации	Цена_реализации	Денежный	Нет	# ##0,00»р.»;-# ##0,00»р.»
9	Количество_на_складе	Количество_на_складе	Числовой	Нет	
10	Срок_реализации	Срок_реализации	Дата и время	Нет	00.00.0000;0;#
11	Код_поставщика	Код_поставщика	Числовой	Да	ВК(внешний ключ) (supplier_id) ссылается на «поставщики»(supplier_id)

Таблица 10 – Список атрибутов таблицы «Заказы»

№	Название	Идентификатор	Тип	Не пуст о	Ограничение
1	Номер_чека	Номер_чека	Числовой	Да	
2	Код_товара	Код_товара	Числовой	Да	
3	Количество	Количество	Числовой	Нет	
4	Цена	Цена	Денежный	Нет	# ##0,00»р.»;-# ##0,00»р.»
5	Стоимость	Стоимость	Денежный	Нет	# ##0,00»р.»;-# ##0,00»р.»

Таблица 11 – Список атрибутов таблицы «Сотрудники»

№	Название	Идентификатор	Тип	Не пуст о	Ограничение
1	Код_сотрудни ка	Код_сотрудника	Короткий текст	Да	ПК (первичный ключ)
2	Номер_паспор та	Номер_паспорта	Короткий текст	Нет	00\ -00» №»000000;0;*
3	Имя	Имя	Короткий текст	Нет	
4	Отчество	Отчество	Короткий текст	Нет	
5	Фамилия	Фамилия	Короткий текст	Нет	
6	Семейное_пол ожение	Семейное_поло жение	Короткий текст	Нет	
7	Должность	Должность	Короткий текст	Нет	
8	Электронная_ почта	Электронная_по чта	Короткий текст	Нет	
9	Дата_рождени я	Дата_рождения	Дата и время	Нет	00.00.0000;0;*
10	Дата_найма	Дата_найма	Дата и время	Нет	00.00.0000;0;*
11	Город	Город	Короткий текст	Нет	
12	Почтовый_ин декс	Почтовый_инде кс	Короткий текст	Нет	
13	Адрес	Адрес	Короткий текст	Нет	
14	Контактный_т елефон	Контактный_тел ефон	Короткий текст	Нет	\(000\)000\ -00\ 00;0;*

Таблица 12 – Список атрибутов таблицы «Счета»

№	Название	Идентификатор	Тип	Не пуст о	Ограничение
1	Номер_чека	Номер_чека	Числовой	Да	ПК (Первичный ключ)
2	Код_сотрудни ка	Код_сотрудника	Короткий текст	Да	ВК (Внешний ключ)
3	Дата	Дата	Дата и время	Нет	00.00.0000;0;*
4	Дисконтная_к арта	Дисконтная_кар та	Логически й	Нет	Да/нет
5	Скидка	Скидка	Числовой	Нет	Проценты
6	Сумма_к_опл ате	Сумма_к_оплате	Денежный	Нет	00.00.0000;0;*

5 Код формирования БД на SQL

5.1 Теоретические сведения о SQL-запросах

SQL (Structured Query Language) — это язык программирования, который используется для работы с реляционными базами данных. Он позволяет создавать, изменять и удалять данные в базе данных, а также извлекать данные с помощью запросов.

SQL-запросы — это команды, которые выполняются на базе данных, чтобы получить нужные данные или изменить существующие записи. Они могут содержать различные операторы, условия, функции и ключевые слова.

Основные операторы SQL-запросов:

SELECT - используется для извлечения данных из таблицы или нескольких таблиц. Он может содержать условия, сортировку, группировку и многое другое.

INSERT - используется для добавления новых записей в таблицу.

UPDATE - используется для обновления существующих записей в таблице.

DELETE - используется для удаления записей из таблицы.

JOIN - используется для объединения данных из нескольких таблиц по условию соответствия.

GROUP BY - используется для группировки данных по определенному столбцу.

ORDER BY - используется для сортировки данных по определенному столбцу.

WHERE - используется для фильтрации данных по определенному условию.

Функции SQL-запросов:

COUNT - используется для подсчета количества записей в таблице.

SUM - используется для подсчета суммы значений столбца в таблице.

AVG - используется для подсчета среднего значения столбца в таблице.

MAX - используется для нахождения максимального значения столбца в таблице.

MIN - используется для нахождения минимального значения столбца в таблице.

DISTINCT - используется для извлечения уникальных значений из столбца.

SQL-запросы могут быть сложными и содержать множество условий и операторов. Они могут также быть оптимизированы для повышения производительности базы данных и ускорения выполнения запроса. Хорошее знание SQL-запросов и возможностей языка SQL в целом позволяет эффективно работать с базами данных и извлекать нужные данные в нужный момент.

5.2 Создание запросов к базе данных

Для каждой таблицы приведем примеры запросов на создание таблиц, на заполнение из данными (строками), а также запросы на выборку:

1) Создадим на языке Transact-SQL файл базы данных согласно своему варианту. Разработаем базу данных на основе спроектированной концептуальной модели данных.

SQL – запрос на создание БД с именем trading_company2 будет выглядеть следующим образом (рисунок 4):

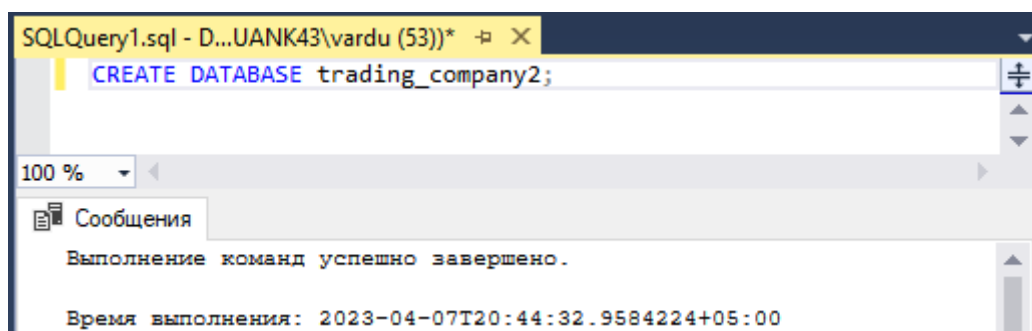


Рисунок 4 – Создание БД с помощью SQL- запроса

Данный запрос создает новую БД с именем «trading_company2» в системе управления базами данных (СУБД), которую мы используем.

После выполнения этого SQL-запроса, база данных «trading_company 2» будет создана и готова к использованию для хранения данных нашей «Торговой организации». Мы можем создавать таблицы, определять структуру данных и выполнять другие операции внутри этой базы данных.

2) Создадим программно на языке SQL все таблицы, с указанием первичных и внешних ключей и ограничения целостности:

2.1 Таблица «Заказы»

Создадим таблицу «Заказы» со следующими полями и их ограничениями, и типами данных (рисунок 5):



Рисунок 5 – Создание таблицы «Заказы»

В этом запросе создается таблица «Заказы» с пятью полями:

1. Номер_чека - представленное в виде целого числа (INT) и не допускающий значений NULL.
2. Код_товара - представленное в виде целого числа (INT) и также не допускающий значений NULL.
3. Количество - представленное в виде целого числа (INT) и допускающее значение NULL.
4. Цена - представленное в виде денежного типа данных (MONEY) и также допускающее значение NULL.
5. Стоимость - также представленное в виде денежного типа данных (MONEY) и допускающее значение NULL.

2.2 Таблица «Товары»

Создадим таблицу «Товары» со следующими полями и их ограничениями, и типами данных (рисунок 6):

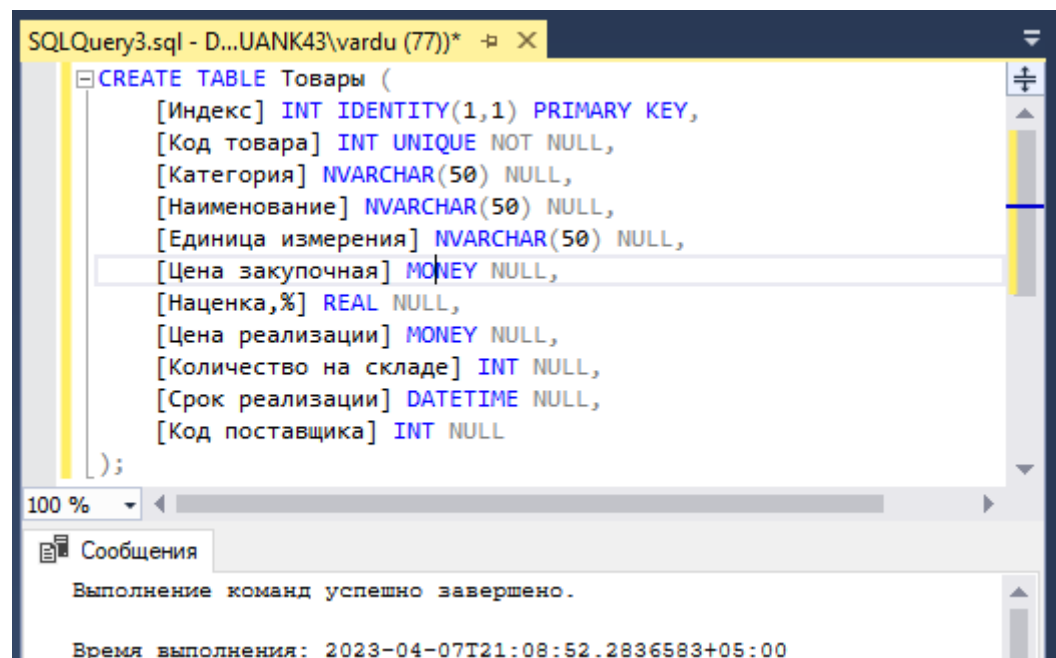


Рисунок 6 – Создание таблицы «Товары»

В этом запросе создается таблица «Товары» с одиннадцатью полями:

1. Индекс - определено как целое число (INT) и автоматически инкрементируется (IDENTITY) с начальным значением 1 и приращением 1 (1,1). Это поле служит в качестве индекса или идентификатора товаров и указано как первичный ключ (PRIMARY KEY) для таблицы.
2. Код товара - уникальное поле (UNIQUE), представленное в виде целого числа (INT) и также не допускающее значений NULL. Значения этого поля не должны повторяться.
3. Категория - представленная в виде строкового значения переменной длины (NVARCHAR(50)) и допускающая значение NULL.

4. Наименование - также представленное в виде строкового значения переменной длины (NVARCHAR(50)) и допускающее значение NULL.

5. Единица измерения - также представленная в виде строкового значения переменной длины (NVARCHAR(50)) и допускающая значение NULL.

6. Цена закупочная - представленная в виде денежного типа данных (MONEY) и также допускающая значение NULL.

7. Наценка,% - представленная в виде вещественного числа (REAL) и допускающая значение NULL.

8. Цена реализации - также представленная в виде денежного типа данных (MONEY) и допускающая значение NULL.

9. Количество на складе - представленное в виде целого числа (INT) и допускающее значение NULL.

10. Срок реализации - представленный в виде значения даты и времени (DATETIME) и допускающий значение NULL.

11. Код поставщика - представленный в виде целого числа (INT) и допускающий значение NULL.

2.3 Таблица «Сотрудники»

Создадим таблицу «Сотрудники» со следующими полями и их ограничениями, и типами данных (рисунок 7):

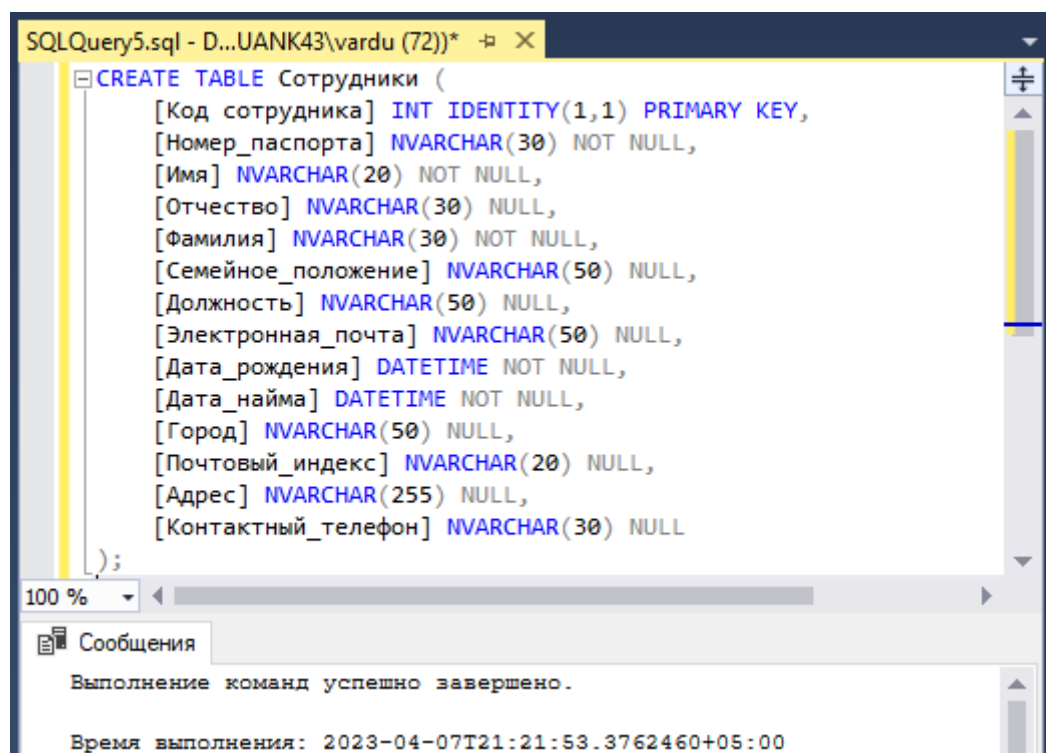


Рисунок 7 – Создание таблицы «Сотрудники»

В этом запросе создается таблица «Сотрудники» с четырнадцатью полями:

1. Код сотрудника - первичный ключ (PRIMARY KEY), представленный в виде целого числа (INT) и автоматически генерирующийся с помощью

инкрементного значения (IDENTITY(1,1)). Это уникальное поле, которое идентифицирует каждого сотрудника в таблице.

2. Номер_паспорта - строковое значение переменной длины (NVARCHAR(30)), которое не допускает значение NULL. Это поле содержит номер паспорта сотрудника.

3. Имя - строковое значение переменной длины (NVARCHAR(20)), которое не допускает значение NULL. Это поле содержит имя сотрудника.

4. Отчество - строковое значение переменной длины (NVARCHAR(30)), которое допускает значение NULL. Это поле содержит отчество сотрудника.

5. Фамилия - строковое значение переменной длины (NVARCHAR(30)), которое не допускает значение NULL. Это поле содержит фамилию сотрудника.

6. Семейное_положение - строковое значение переменной длины (NVARCHAR(50)), которое допускает значение NULL. Это поле содержит семейное положение сотрудника.

7. Должность - строковое значение переменной длины (NVARCHAR(50)), которое допускает значение NULL. Это поле содержит должность сотрудника.

8. Электронная_почта - строковое значение переменной длины (NVARCHAR(50)), которое допускает значение NULL. Это поле содержит электронную почту сотрудника.

9. Дата_рождения - значение даты и времени (DATETIME), которое не допускает значение NULL. Это поле содержит дату рождения сотрудника.

10. Дата_найма - значение даты и времени (DATETIME), которое не допускает значение NULL. Это поле содержит дату найма сотрудника.

11. Город - строковое значение переменной длины (NVARCHAR(50)), которое допускает значение NULL. Это поле содержит город проживания сотрудника.

12. Почтовый_индекс - строковое значение переменной длины (NVARCHAR(20)), которое допускает значение NULL. Это поле содержит почтовый индекс места проживания сотрудника.

13. Адрес - строковое значение переменной длины (NVARCHAR(255)), которое допускает значение NULL. Это поле содержит адрес места проживания сотрудника.

14. Контактный_телефон - строковое значение переменной длины (NVARCHAR(30)), которое допускает значение NULL. Это поле содержит контактный телефон сотрудника.

2.4 Таблица «Счета»

Создадим таблицу «Счета» со следующими полями и их ограничениями, и типами данных (рисунок 8):

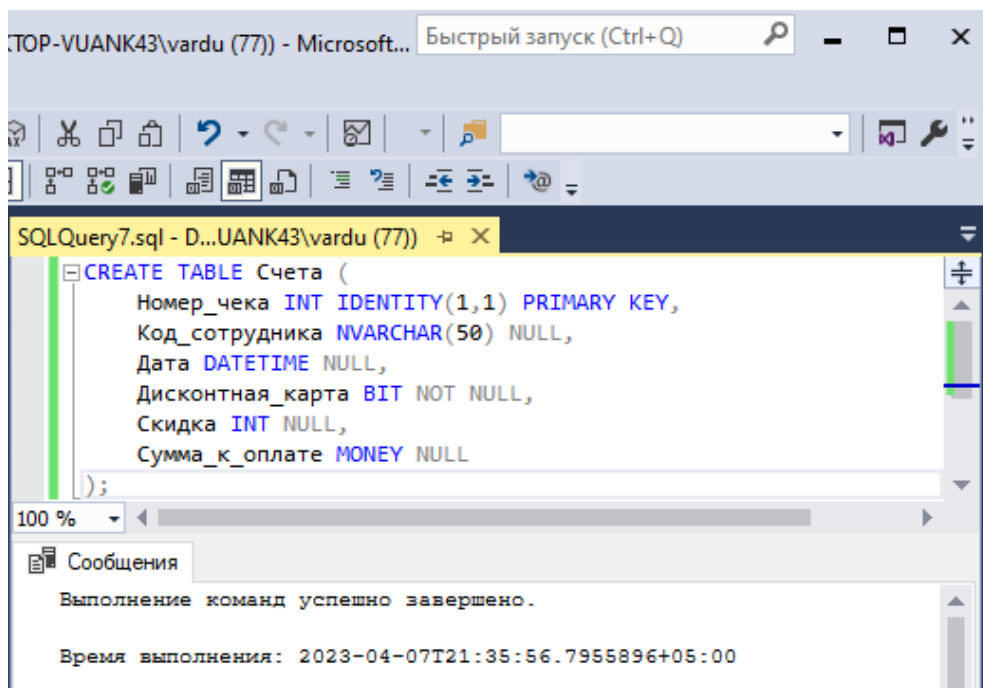


Рисунок 8 – Создание таблицы «Счета»

В этом запросе создается таблица «Счета» с шестью полями:

1. Номер_чека - первичный ключ (PRIMARY KEY), представленный в виде целого числа (INT) и автоматически генерирующийся с помощью инкрементного значения (IDENTITY(1,1)). Это уникальное поле, которое идентифицирует каждый чек в таблице.

2. Код_сотрудника - строковое значение переменной длины (NVARCHAR(50)) NULL, которое содержит код сотрудника, ответственного за этот чек. Это поле может содержать значение NULL, если код сотрудника не указан.

3. Дата - поле типа DATETIME, которое содержит дату создания чека. Это поле может содержать значение NULL, если дата неизвестна.

4. Дисконтная_карта - поле типа BIT, которое указывает, есть ли дисконтная карта у покупателя. Значение 1 означает наличие дисконтной карты, а значение 0 - отсутствие.

5. Скидка - поле типа INT, которое содержит значение скидки, предоставленной покупателю на этом чеке. Это поле может содержать значение NULL, если скидка не предоставлена.

6. Сумма_к_оплате - поле типа MONEY, которое содержит сумму к оплате по этому чеку. Это поле может содержать значение NULL, если сумма неизвестна.

2.5 Таблица «Поставщики»

Создадим таблицу «Поставщики» со следующими полями и их ограничениями, и типами данных (рисунок 9):

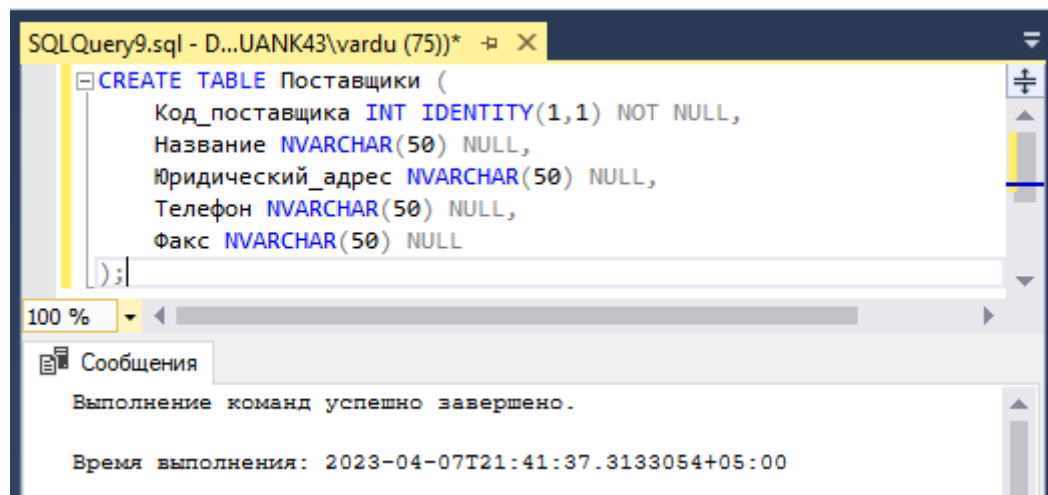


Рисунок 9 – Создание таблицы «Поставщики»

В этом запросе создается таблица «Поставщики» с пятью полями:

1. Код_поставщика - поле типа INT, которое не может содержать значение NULL и инкрементируется автоматически. Оно представляет собой уникальный код (идентификатор) поставщика.
2. Название - поле типа NVARCHAR(50), которое может содержать значение NULL. Оно содержит название поставщика, представленное в виде текстовой строки длиной до 50 символов.
3. Юридический_адрес - поле типа NVARCHAR(50), которое может содержать значение NULL. Оно содержит юридический адрес поставщика, представленный в виде текстовой строки длиной до 50 символов.
4. Телефон - поле типа NVARCHAR(50), которое может содержать значение NULL. Оно содержит контактный телефон поставщика, представленный в виде текстовой строки длиной до 50 символов.
5. Факс - поле типа NVARCHAR(50), которое может содержать значение NULL. Оно содержит номер факса поставщика, представленный в виде текстовой строки длиной до 50 символов.

3) Теперь свяжем таблицы с помощью SQL – запросов:

3.1 «Поставщики» – «Товары»

Данный запрос создаст внешний ключ «Код_поставщика» в таблице «Товары», который будет ссылаться на поле «Код_поставщика» в таблице «Поставщики» (рисунок 10).

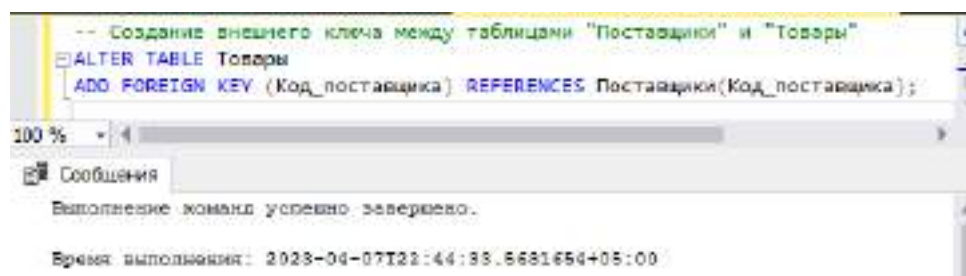


Рисунок 10 – Связывание таблиц с помощью SQL-запроса

3.2 «Товары» - «Заказы»

Данный запрос создаст внешний ключ «Код_товара» в таблице «Товары», который будет ссылаться на поле «Код_товара» в таблице «Заказы» (Рисунок 11).

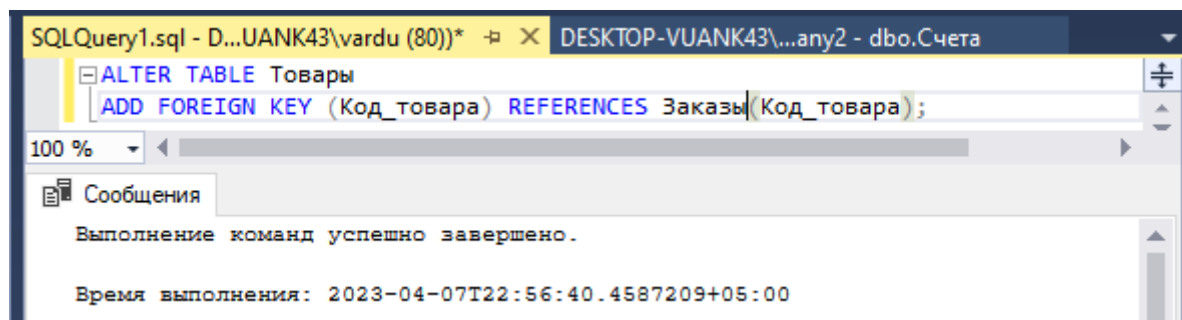


Рисунок 11 – Связывание таблиц с помощью SQL-запроса

3.3 «Заказы» - «Счета»

Данный запрос создаст внешний ключ «Номер_чека» в таблице «Заказы», который будет ссылаться на поле «Номер_чека» в таблице «Счета» (Рисунок 12).

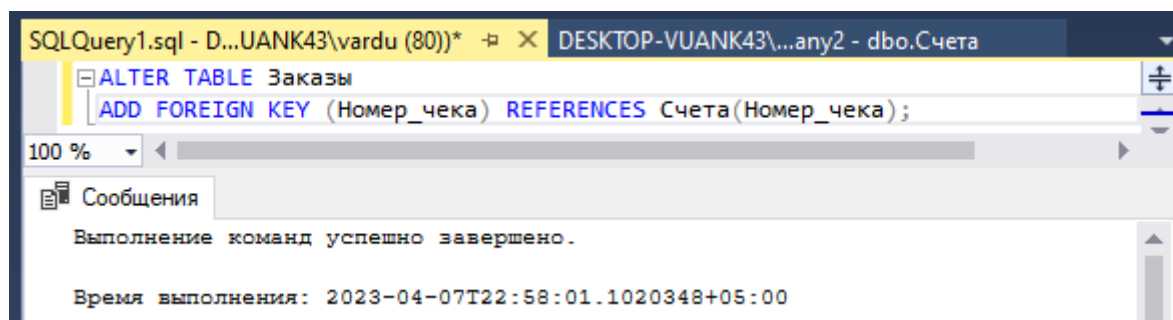


Рисунок 12 – Связывание таблиц с помощью SQL-запроса

3.4 «Счета» - «Сотрудники»

Данный запрос создаст внешний ключ «Код_сотрудника» в таблице «Счета», который будет ссылаться на поле «Код_сотрудника» в таблице «Сотрудники» (Рисунок 13).

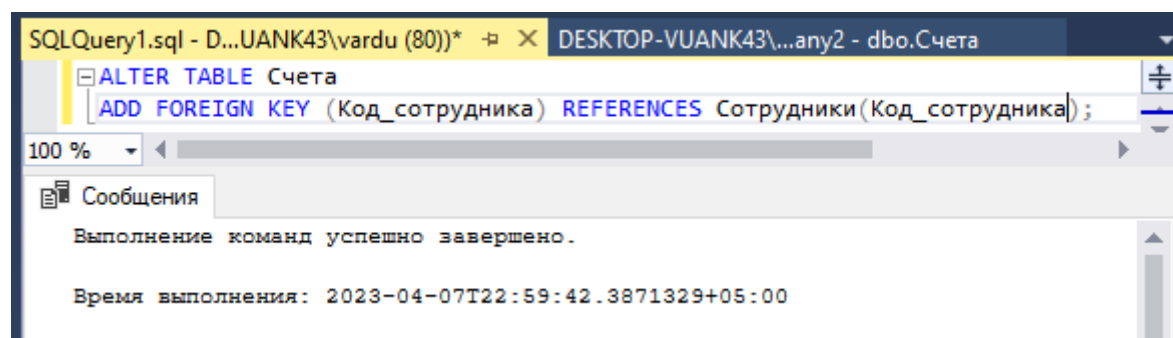
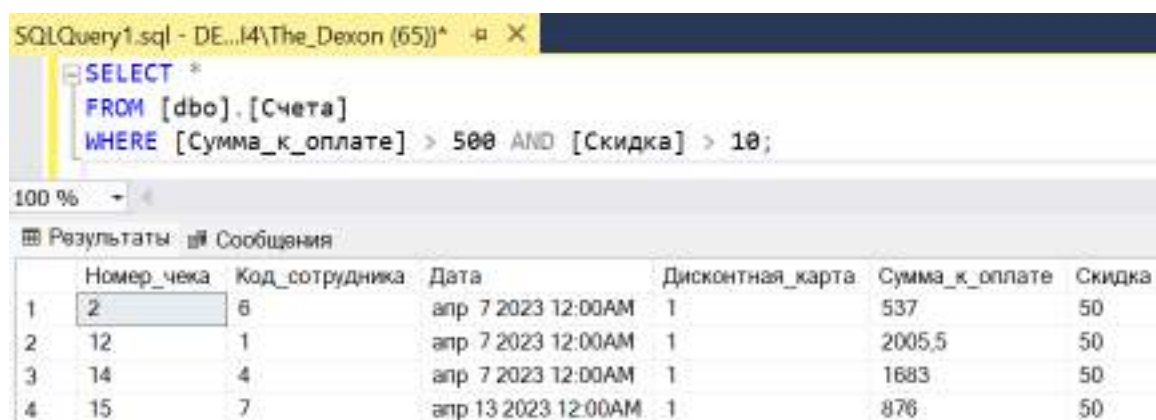


Рисунок 13 – Связывание таблиц с помощью SQL-запроса

4) Создадим запросы на выборку, а также отбор строк по условию:

4.1 простейшие запросы с использованием операторов сравнения

В данном запросе мы можем получить данные всех счетов, у которых сумма к оплате больше 5000 рублей и скидка больше 10% (Рисунок 14).



SQLQuery1.sql - DE...I4\The_Dexon (65))

```
SELECT *  
FROM [dbo].[Счета]  
WHERE [Сумма_к_оплате] > 5000 AND [Скидка] > 10;
```

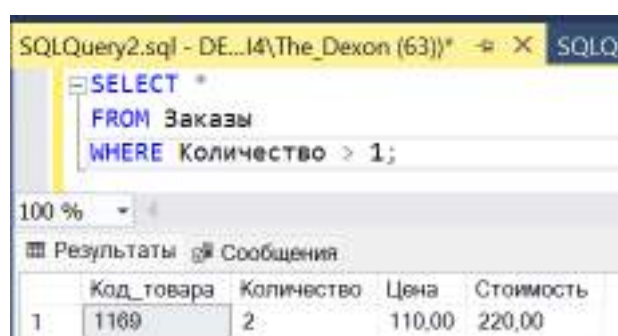
100 %

Результаты Сообщения

	Номер_чека	Код_сотрудника	Дата	Дисконтная_карта	Сумма_к_оплате	Скидка
1	2	6	апр 7 2023 12:00AM	1	537	50
2	12	1	апр 7 2023 12:00AM	1	2005,5	50
3	14	4	апр 7 2023 12:00AM	1	1683	50
4	15	7	апр 13 2023 12:00AM	1	876	50

Рисунок 14 – Результат запроса

Данный запрос позволяет сделать вывод всех заказов, где количество товара больше 1 единицы (Рисунок 15).



SQLQuery2.sql - DE...I4\The_Dexon (63))

```
SELECT *  
FROM Заказы  
WHERE Количество > 1;
```

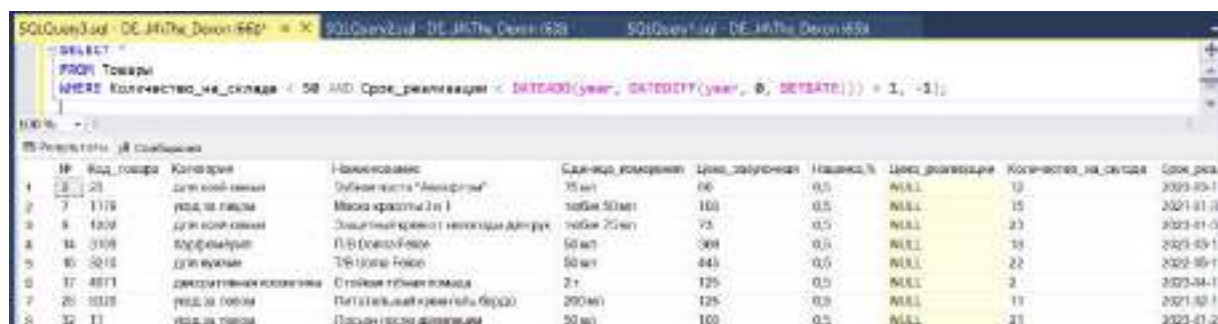
100 %

Результаты Сообщения

	Код_товара	Количество	Цена	Стоимость
1	1169	2	110,00	220,00

Рисунок 15 – Результат запроса

Данный запрос позволяет сделать вывод всех товаров, у которых количество на складе меньше 50 и срок реализации до конца текущего года (Рисунок 16).



SQLQuery3.sql - DE...I4\The_Dexon (66))

```
SELECT *  
FROM Товары  
WHERE Количество_на_складе < 50 AND Срок_реализации < DATEADD(year, DATEDIFF(year, 0), GETDATE()) + 1, -1);
```

100 %

Результаты Сообщения

	ИД	Код_товара	Категория	Наименование	Единица_измерения	Цена_закупочная	Наценка, %	Цена_продажная	Количество_на_складе	Срок_реализации
1	2	20	длиннокопеечные	Длиннокопеечные "Амстердам"	75 мл	80	0,5	80,5	10	2023-05-1
2	7	1179	уход за кожей	Маска красоты 3 в 1	таблетки 50мл	160	0,5	160,5	15	2023-01-3
3	8	1209	длиннокопеечные	Длиннокопеечные "Амстердам"	таблетки 20мл	95	0,5	95,5	23	2023-01-3
4	14	3109	уход за кожей	П.В. (Солнечный)	50 мл	300	0,5	300,5	10	2023-05-1
5	15	5218	длиннокопеечные	Т.В. (Солнечный)	50 мл	440	0,5	440,5	22	2023-05-1
6	17	4871	длиннокопеечные	Стойкая помада	2 г	125	0,5	125,5	2	2023-04-1
7	20	1033	уход за кожей	Питательный крем (Солнечный)	200 мл	125	0,5	125,5	11	2023-05-1
8	32	11	уход за кожей	Питательный крем (Солнечный)	50 мл	160	0,5	160,5	21	2023-01-3

Рисунок 16 – Результат запроса

4.2 запросы с использованием логических операторов AND, OR и NOT

В этом запросе мы выбираем все записи из таблицы «Сотрудники», где значение столбца «Должность» равно «Менеджер» ИЛИ «Директор» (Рисунок 17).



The screenshot shows a SQL query window with the following query: `SELECT * FROM Сотрудники WHERE Должность = 'Менеджер' OR Должность = 'Директор'`. The results table contains three rows of employee data.

№	Код_сотрудника	Идентификационный номер	Имя	Отчество	Фамилия	Семейное_имя	Должность	Пол	Дата_рождения	Дата_найма
1	2	22-02-19912421	Наталья	Ивановна	Захарова	захарова	менеджер	NULL	1975-11-25 00:00:00.000	2019-03-04 00:00:00.000
2	8	22-04-19912404	Екатерина	Ивановна	Сидорова	сидорова	менеджер	NULL	1979-07-12 00:00:00.000	2020-02-21 00:00:00.000
3	11	22-01-19912042	Нарзан	Андреевич	Варданян	назарян	Директор	мужской_18_февраля_1975	2001-10-26 00:00:00.000	2020-04-06 00:00:00.000

Рисунок 17 – Результат запроса

Здесь мы выбираем все записи из таблицы «Товары», где значение столбца «Категория» НЕ равно «уход за лицом» (Рисунок 18).



The screenshot shows a SQL query window with the following query: `SELECT * FROM Товары WHERE NOT Категория = 'уход за лицом'`. The results table contains five rows of product data.

№	Код_товара	Категория	Наименование	Единица_измерения	Цена_закупочная	Наценка_%	Цена_реализации	Количество
1	12	уход за телом	Антицеллюлитный гидрококоскреат	200 мл	152	0.5	NULL	10
2	21	для волос	Шампунь для волос "Ароматный"	75 мл	66	0.5	NULL	12
3	1201	для волос	Защитный крем от солнца для лица	пакет 50 мл	88	0.5	NULL	72
4	1202	для волос	Защитный крем от солнца для тела	пакет 75 мл	75	0.5	NULL	23
5	1402	для мужчин	Гель-крем для умывания	150 мл	88	0.5	NULL	52

Рисунок 18 – Результат запроса

Данный запрос выбирает все записи из таблицы «Сотрудники», где значение столбца «Город» равно «Москва» И значение столбца «Дата_рождения» больше 1986-12-25 (Рисунок 19).



The screenshot shows a SQL query window with the following query: `SELECT * FROM Сотрудники WHERE Город = 'Москва' AND Дата_рождения > '1986-12-25'`. The results table contains two rows of employee data.

Отчество	Фамилия	Семейное_имя	Должность	Пол	Дата_рождения	Дата_найма	Город	Почтовый_индекс
Андреевич	Горбачев	назарян	менеджер	мужской	1976-05-10 00:00:00.000	2020-05-20 00:00:00.000	Москва	817407
Владимировна	Малова	захарова	менеджер	женский	1986-12-24 00:00:00.000	2020-02-15 00:00:00.000	Москва	804192

Рисунок 19 – Результат запроса

4.3 запрос на использование комбинации логических операторов

В этом запросе мы выбираем все записи из таблицы «Товары», где значение столбца «Категория» равно «уход за телом» И значение столбца «Цена_закупочная» больше 50 ИЛИ значение столбца «Код_поставщика» равно «3», И значение столбца «Количество_на_складе» больше 10. Здесь также используется комбинация операторов AND и OR, а также круглые скобки для определения порядка выполнения условий (Рисунок 20).

SQL Query 1 - DE...The.Demon (117) - X

```
SELECT *
FROM Товары
WHERE Категория = 'Уход за кожей' AND Цена_закупочная > 50 AND Код_поставщика = '3' AND Количество_на_складе > 10
```

№	Код_товара	Категория	Наименование	Единица_измерения	Цена_закупочная	Наценка,%	Цена_реализации	Количество_на_складе
1	11	2016	Масло для лица увлажняющее	150 мл	85	8,5	100,5	30
2	12	2016	Крем для лица питательный для сухой кожи	150 мл	95	8,5	103,25	12
3	33	2016	Средство для лица увлажняющее	200 мл	105	8,5	114,75	15
4	31	1101	Лосьон после бритья	50 мл	103	8,5	111,75	12
5	32	11	Пена после бритья	50 мл	103	8,5	111,75	20

Рисунок 20 – Результат запроса

4.4 запрос на использование выражений над столбцами

На рисунке 21 демонстрируется SQL-запрос, который выбирает данные из таблицы «Товары» и создает новый столбец с именем «[Цена_реализации]», который содержит результат выражения «Товары.[Цена_закупочная] * Товары.[Наценка,%] + Товары.[Цена_закупочная]»:

SQL Query 2 - DE...The.Demon (156) - X

```
SELECT Товары.[Код_товара],
Товары.[Категория],
Товары.[Наименование],
Товары.[Единица_измерения],
Товары.[Цена_закупочная],
Товары.[Наценка,%],
(Товары.[Цена_закупочная]*Товары.[Наценка,%]+Товары.[Цена_закупочная])
AS [Цена_реализации],
Товары.[Количество_на_складе],
Товары.[Срок_реализации],
Товары.[Код_поставщика]
FROM Товары
```

№	Код_товара	Категория	Наименование	Единица_измерения	Цена_закупочная	Наценка,%	Цена_реализации	Количество
1	12	уход за кожей	Антифолликулярный термеконцентр	250 мл	132	8,5	143,8	10
2	23	для всей семьи	Зубная паста "Акварель"	75 мл	86	8,5	92,65	12
3	340	уход за кожей	Тонизирующий гель для лица	гель 15 мл	88	8,5	94,8	5
4	332	уход за кожей	Средство для лица	гель 50 мл	81	8,5	87,25	25
5	1167	уход за кожей	Увлажняющий крем для лица с глицерином	флакон 50 мл	85	8,5	91,25	14
6	1199	уход за кожей	Гель для лица "Бодарость"	гель 50 мл	110	8,5	119,5	11
7	1170	уход за кожей	Масло для лица	гель 50 мл	102	8,5	109,75	15
8	1201	для всей семьи	Защитный крем от насекомых для лица	гель 50 мл	83	8,5	90,25	12
9	1202	для всей семьи	Защитный крем от насекомых для лица	гель 75 мл	73	8,5	79,25	23
10	1452	для мужчин	Гель для лица	150 мл	88	8,5	94,8	12
11	2016	уход за кожей	Масло для лица увлажняющее	150 мл	95	8,5	103,25	33
12	2016	уход за кожей	Крем для лица питательный для сухой кожи	150 мл	95	8,5	103,25	12
13	3102	парфюмерия	ПВ Aloga	флакон 50 мл	340	8,5	368,5	30
14	3109	парфюмерия	ПВ Donna Falso	50 мл	380	8,5	408,5	18
15	3123	парфюмерия	ПВ Viole	50 мл	221	8,5	239,75	16
16	3210	для мужчин	ПВ Uomo Falso	50 мл	443	8,5	478,75	22
17	4871	декоративная косметика	Стойкий губный помада	2 г	129	8,5	139,75	2
18	4933	декоративная косметика	Блеск для губ обильно блестящий	18 мл	73	8,5	79,25	11

Рисунок 21 – Результат запроса

4.5 запросы с проверкой на принадлежность множеству

На рисунке 22 демонстрируется SQL-запрос, который выводит всех сотрудников, у которых должность содержит слово «менеджер» и контактный телефон не равен NULL.

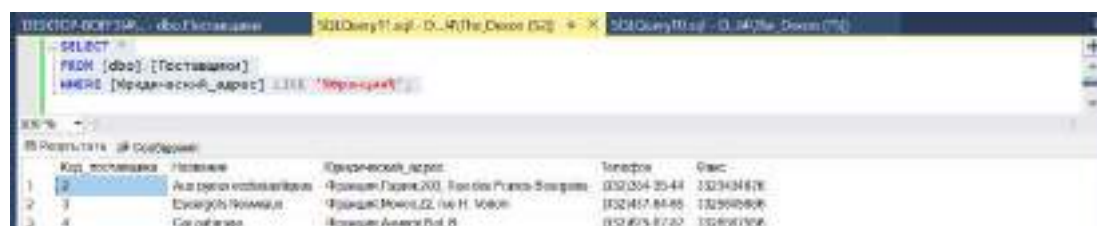
SQL Query 1 - DE...The.Demon (175) - X

```
SELECT *
FROM [dbo].[Сотрудники]
WHERE [Должность] LIKE 'менеджер%' AND [Контактный_телефон] IS NOT NULL;
```

№	Код_сотрудника	Имя_фамилия	ВМ	Отечество	Полное_имя	Семейное_имя	Должность	Контактный_телефон	Дата_рождения	Дата_адреса
1	2	22-03-1981-1421	Мужчина	Иванович	Иванов	Иванов	менеджер	NULL	1979-11-23 00:00:00.000	2010-03-04 00:00:00.000
2	9	22-04-1981-1444	Мужчина	Александрович	Сидоров	Сидоров	менеджер	NULL	1979-12-12 00:00:00.000	2010-03-04 00:00:00.000

Рисунок 22 – Результат запроса

На рисунке 23 демонстрируется SQL-запрос на выборку всех поставщиков, чьи юридические адреса содержат определенное ключевое слово.



The screenshot shows a SQL query window with the following query:

```
SELECT *
FROM [dbo].[Поставщики]
WHERE [Юридический_адрес] LIKE 'Москва'
```

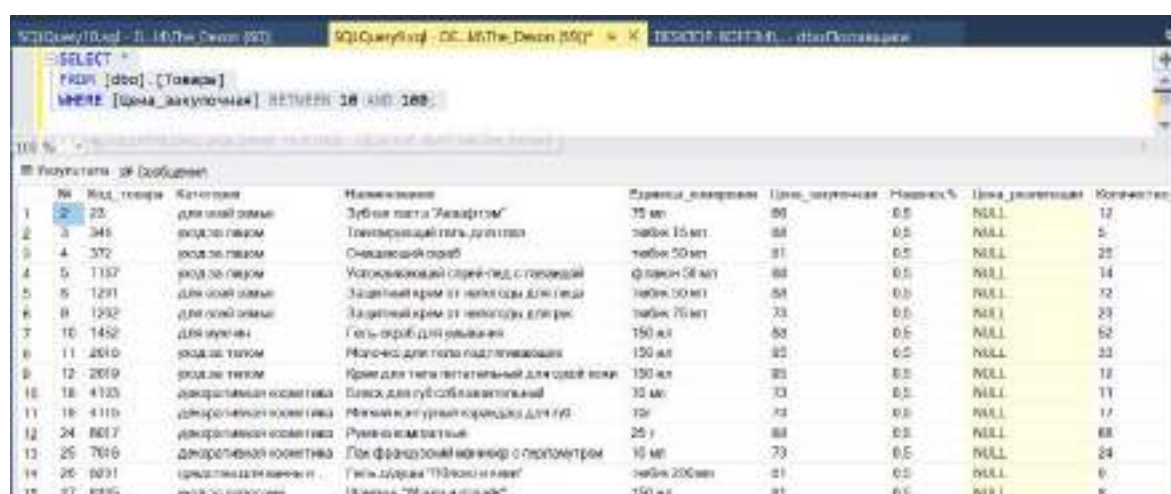
The results table shows the following data:

ID	Код поставщика	Наименование	Юридический_адрес	Телефон	Факс
1	2	Автосервис "Скорая помощь"	Москва, Ленинский пр-д, д. 100	(495) 234-56-78	(495) 234-56-79
2	3	Специализированный сервис	Москва, Мещеряковская ул, д. 10	(495) 234-56-78	(495) 234-56-79
3	4	Сервис-центр	Москва, Ленинский пр-д, д. 100	(495) 234-56-78	(495) 234-56-79

Рисунок 23 – Результат запроса

4.6 запросы с проверкой на принадлежность диапазону значений

На рисунке 24 демонстрируется SQL-запрос на выборку всех товаров, у которых цена закупочная находится в определенном диапазоне.



The screenshot shows a SQL query window with the following query:

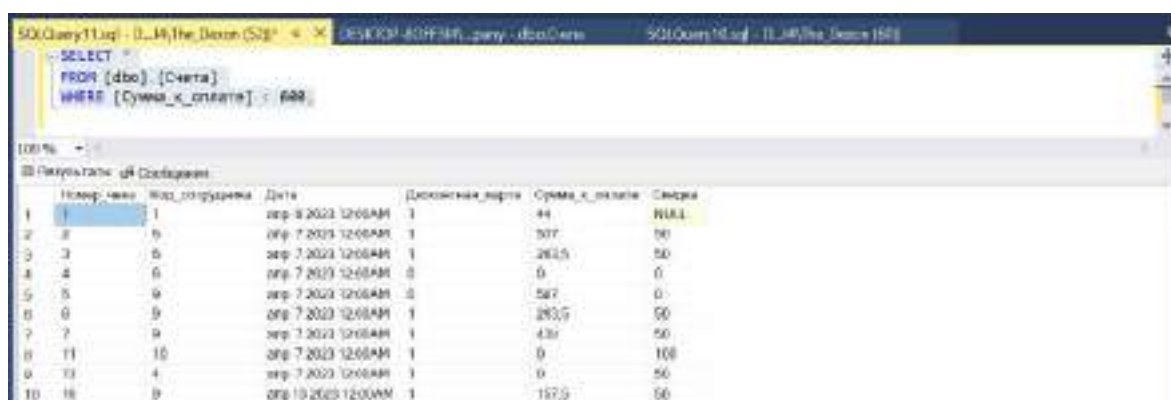
```
SELECT *
FROM [dbo].[Товары]
WHERE [Цена_закупочная] BETWEEN 10 AND 100
```

The results table shows the following data:

ID	Код товара	Категория	Наименование	Единица измерения	Цена_закупочная	Налог_%,	Цена_реализации	Количество
1	23	для ошейников	Зубная паста "Аквадент"	75 мл	80	0.5	NULL	12
2	346	родовые пакеты	Термоизоляционный материал	материал 15 м	88	0.5	NULL	5
3	372	родовые пакеты	Специальный материал	материал 50 м	81	0.5	NULL	25
4	1132	для ошейников	Увлажняющий крем-гель с глицерином	флакон 50 мл	88	0.5	NULL	14
5	1291	для ошейников	Защитный крем от ультрафиолета	материал 50 м	88	0.5	NULL	72
6	1292	для ошейников	Защитный крем от холода и ветра	материал 75 м	73	0.5	NULL	23
7	1452	для ошейников	Гель-крем для ошейников	150 мл	88	0.5	NULL	52
8	2010	родовые пакеты	Материал для упаковки	150 мл	82	0.5	NULL	33
9	2019	родовые пакеты	Крем для тела с натуральными маслами	150 мл	88	0.5	NULL	12
10	4103	для ошейников	Паста для зубной пасты	75 мл	73	0.5	NULL	11
11	4110	для ошейников	Материал для упаковки	75	73	0.5	NULL	17
12	8017	для ошейников	Ручная мыльница	25 г	88	0.5	NULL	88
13	7016	для ошейников	Полупроводниковый датчик	10 м	73	0.5	NULL	24
14	8031	для ошейников	Гель-крем "Тропический"	материал 200 м	81	0.5	NULL	6
15	8285	родовые пакеты	Материал "Пластик"	150 мл	81	0.5	NULL	8

Рисунок 24 – Результат запроса

На рисунке 25 демонстрируется SQL-запрос на выборку всех счетов, у которых сумма к оплате меньше определенного значения.



The screenshot shows a SQL query window with the following query:

```
SELECT *
FROM [dbo].[Счета]
WHERE [Сумма_к_оплате] < 600
```

The results table shows the following data:

ID	Номер счета	Код контрагента	Дата	Договорная_сумма	Сумма_к_оплате	Сумма
1	1	1	01.07.2023 12:00AM	1	44	NULL
2	2	5	01.07.2023 12:00AM	1	307	50
3	3	5	01.07.2023 12:00AM	1	263.5	50
4	4	5	01.07.2023 12:00AM	0	0	0
5	5	5	01.07.2023 12:00AM	0	587	0
6	6	5	01.07.2023 12:00AM	1	280.5	50
7	7	5	01.07.2023 12:00AM	1	431	50
8	11	10	01.07.2023 12:00AM	1	0	100
9	13	4	01.07.2023 12:00AM	1	0	50
10	16	5	01.07.2023 12:00AM	1	157.5	50

Рисунок 25 – Результат запроса

4.7 запросы с проверкой на соответствие шаблону

На рисунке 26 демонстрируется SQL-запрос, который выбирает все записи из таблицы «Сотрудники», у которых значение в столбце «Должность» содержит указанный шаблон «менеджер».

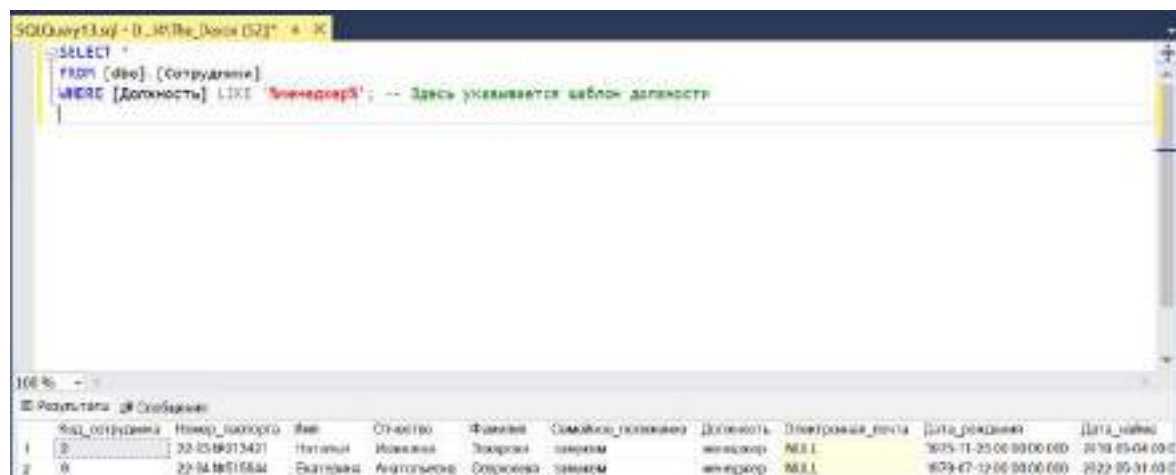


Рисунок 26 – Результат запроса

На рисунке 27 демонстрируется SQL-запрос, который выбирает все записи из таблицы «Сотрудники», у которых значение в столбце «Электронная_почта» заканчивается на указанное доменное имя, например «@gmail.com».



Рисунок 27 – Результат запроса

4.8 запрос с проверкой на неопределенное значение

На рисунке 28 демонстрируется SQL-запрос, который выбирает все записи из таблицы «Сотрудники», у которых значение в столбце «Электронная_почта» отсутствует, то есть является неопределенным (NULL).

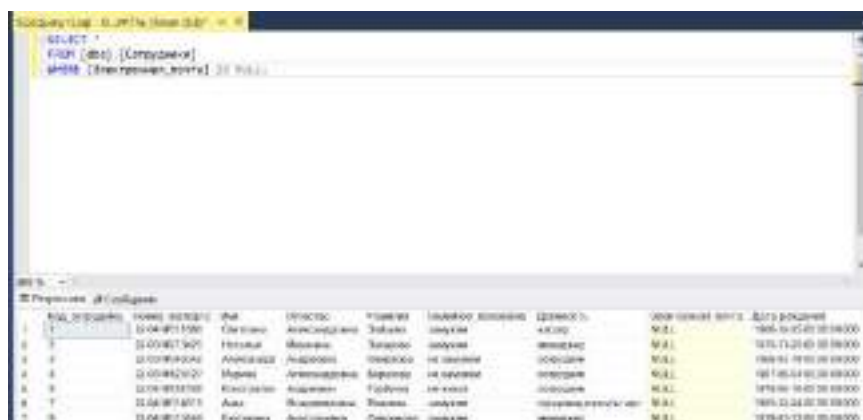


Рисунок 28 – Результат запроса

6 Приложение для работы с БД

6.1 Структура приложения

Структура приложения для БД, разработанной в Windows Forms с использованием C#, обычно состоит из трех основных компонентов:

Интерфейс пользователя (UI) - этот компонент отвечает за отображение данных, взаимодействие с пользователем и передачу команд в приложение. Интерфейс пользователя может содержать формы, кнопки и другие элементы интерфейса, которые пользователь может использовать для работы с базой данных.

База данных (БД) — это компонент приложения, который отвечает за хранение данных. База данных может быть создана с использованием различных СУБД, таких как SQL Server, MySQL, PostgreSQL и других. БД должна содержать таблицы, в которых хранятся данные, и связи между таблицами.

Код приложения — это компонент, который связывает пользовательский интерфейс и базу данных. Код приложения может включать в себя классы и методы, которые отвечают за получение и обработку данных из базы данных, а также за передачу этих данных на пользовательский интерфейс.

6.2 Проектирование графического интерфейса

Проектирование графического интерфейса (GUI) является одной из важнейших задач в создании диалогового приложения. Графический интерфейс должен быть удобным и интуитивно понятным для пользователей, что обеспечит эффективное взаимодействие с приложением.

При проектировании GUI для приложения Windows Forms в Visual Studio используется конструктор форм, который позволяет быстро и просто создавать различные элементы интерфейса, такие как кнопки, текстовые поля, таблицы и другие элементы управления.

Главным элементом интерфейса приложения является главное окно. В окне могут быть размещены различные элементы управления, такие как кнопки, текстовые поля, таблицы и т.д. Окно можно настроить на различные режимы отображения, такие как полноэкранный или оконный режим.

Кроме главного окна, в приложении могут быть созданы другие формы, которые открываются при определенных действиях пользователя.

Также в приложении может быть использовано меню, которое содержит набор команд для управления приложением. Меню можно создать вручную или с помощью конструктора меню, который также доступен в Visual Studio.

Важно помнить, что проектирование графического интерфейса должно быть ориентировано на конечного пользователя, и учитывать его потребности и

ожидания. Хорошо спроектированный интерфейс сделает работу с приложением более комфортной и продуктивной.

В разработанном приложении используется внутреннее подключение к базе данных с помощью строки подключения: `SqlConnection connection = new SqlConnection(«DataSource=DESKTOP-80FF3I4\SQLEXPRESS02; InitialCatalog=Trading_company;IntegratedSecurity=True»);`

На рисунке 29 представлено главное окно приложения, из которого мы можем получить доступ к таким диалоговым окнам как: «Информация о товарах», «Поставщики», «Оформление заказов», «Сведения о сотрудниках»

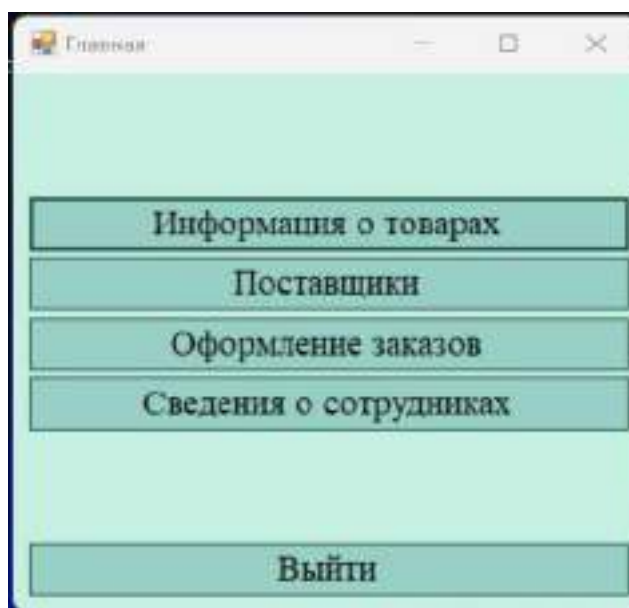


Рисунок 29 – Главное окно приложения

При нажатии на кнопку «Информация о товарах» откроется новое окно с таблицей «Товары» (рисунок 30) и различными кнопками с разными функциями:



Рисунок 30 – Диалоговое окно «Товары»

При нажатии на кнопку «Рассчитать цену реализации» откроется новое диалоговое окно «Расчет цены реализации» (Рисунок 31):

№	Код_товара	Категория	Наименование	Единица_измерения	Цена_закупочная	Наценка, %	Цена_реализации	Количество_на_складе	Срок_реализации
1	12	для мойки	Детекторная...	шт	142	0,5	147,7	10	12.01.2009
2	31	для мойки	Зачисточная...	шт	86	0,5	88,3	10	13.01.2009
3	340	для мойки	Тонировочная...	шт	88	0,5	90,4	3	14.01.2009
4	357	для мойки	Очистительная...	шт	81	0,5	83,5	35	27.02.2009
5	1167	для мойки	Мойка...	шт	88	0,5	90,4	18	12.02.2009
6	1169	для мойки	Гель для мойки...	шт	110	0,5	111,5	11	12.01.2009
7	1173	для мойки	Маска...	шт	100	0,5	101,5	13	31.01.2009

Рисунок 31 – Диалоговое окно «Расчет цены реализации»

«Расчет цены реализации» — это SQL-запрос, который в коде выглядит следующим образом:

```
SqlCommand command = new SqlCommand(«SELECT Товары.[№],
Товары.[Код_товара], Товары.Категория, Товары.Наименование,
Товары.[Единица_измерения], Товары.[Цена_закупочная], Товары.[Наценка,%],
(Товары.[Цена_закупочная]*Товары.[Наценка,%]+Товары.[Цена_закупочная]) AS
[Цена_реализации], Товары.[Количество_на_складе], Товары.[Срок_реализации],
Товары.[Код_поставщика] FROM Товары», connection);
```

В столбце «Цена_реализации» происходит вычисление значения по формуле: $\text{Цена_закупочная} * \text{Наценка, \%} + \text{Цена_закупочная}$.

Таким образом, данный запрос предоставляет информацию о товарах, хранящихся в таблице, включая информацию о цене реализации, которая вычисляется на основе цены закупки и наценки.

При нажатии на кнопку «Списанные товары» откроется новое диалоговое окно «Списанные товары» (Рисунок 32):

№	Код_товара	Категория	Наименование	Единица_измерения	Цена_реализации	Наценка, %	Цена_реализации	Количество_на_складе	Срок_реализации
1	12	для мойки	Детекторная...	шт	142	0,5	147,7	10	12.01.2009
2	31	для мойки	Зачисточная...	шт	86	0,5	88,3	10	13.01.2009
3	340	для мойки	Тонировочная...	шт	88	0,5	90,4	3	14.01.2009
4	357	для мойки	Очистительная...	шт	81	0,5	83,5	35	27.02.2009
5	1167	для мойки	Мойка...	шт	88	0,5	90,4	18	12.02.2009
6	1169	для мойки	Гель для мойки...	шт	110	0,5	111,5	11	12.01.2009
7	1173	для мойки	Маска...	шт	100	0,5	101,5	13	31.01.2009
8	1174	для мойки	Маска...	шт	100	0,5	101,5	13	31.01.2009
9	1175	для мойки	Маска...	шт	100	0,5	101,5	13	31.01.2009
10	1176	для мойки	Маска...	шт	100	0,5	101,5	13	31.01.2009

Рисунок 32 – Диалоговое окно «Списанные товары»

«Списанные товары» — это SQL-запрос, который в коде выглядит следующим образом:

```
SqlCommand command = new SqlCommand(«SELECT * FROM Товары WHERE
[Срок_реализации] < @Today», connection);
```

Данный код создает экземпляр класса SqlCommand, который содержит SQL-запрос «SELECT * FROM Товары WHERE [Срок_реализации] < @Today». Здесь @Today является параметром запроса и может быть заменен на определенное значение при выполнении запроса.

Запрос выбирает все записи из таблицы «Товары», где значение поля «Срок_реализации» меньше значения параметра @Today. При выполнении запроса необходимо будет указать значение параметра @Today, чтобы получить результаты запроса.

При нажатии на кнопку «Поиск товара по коду» откроется новое диалоговое окно «Поиск товара», где можно ввести в поле «Код_товара» и получить данные о нем (Рисунок 33):

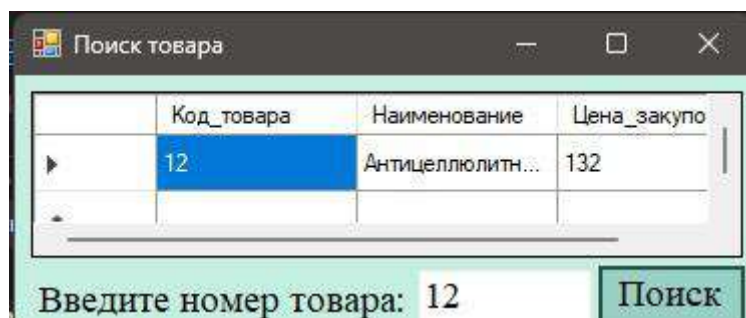


Рисунок 33 – Диалоговое окно «Поиск товара»

«Поиск товара» — это SQL-запрос, который в коде выглядит следующим образом:

```
string query = «SELECT [Код_товара], Наименование, [Цена_закупочная]  
FROM Товары WHERE [Код_товара] = @код_товара»;
```

Этот код объявляет строку запроса SQL, которая выбирает из таблицы «Товары» код товара, наименование и цену закупки для товара, у которого код товара равен заданному параметру «@код_товара», который нужно ввести в поле «Введите номер товара:».

При нажатии на кнопку «Просмотр и печать ценников» откроется новое диалоговое окно «Просмотр и печать» (Рисунок 34):

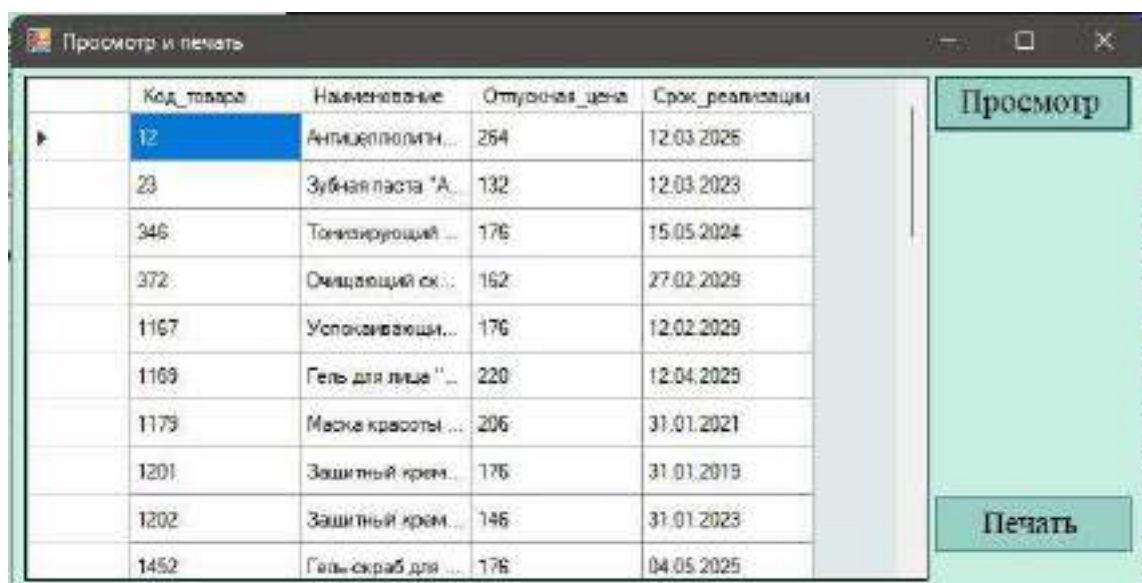


Рисунок 34 – Диалоговое окно «Просмотр и печать»

В данном диалоговом окне, если нажать на кнопку «Просмотр», то сработает SQL – запрос и появится таблица, код выглядит следующим образом:

```
string query = «SELECT Товары.[Код_товара], Товары.Наименование,  
(Товары.[Цена_закупочная]*Товары.[Наценка,%]+Товары.[Цена_закупочная])  
AS[Отпускная_цена], Товары.[Срок_реализации] FROM Товары»;
```

Данный запрос выбирает данные из таблицы «Товары» и производит вычисления для получения отпускной цены, используя цену закупки и наценку в процентах, а также выбирает срок реализации товара. Запрос выбирает следующие столбцы:

- Код товара (Код_товара)
- Наименование товара (Наименование)
- Отпускная цена (вычисляемый столбец с именем «Отпускная_цена»)
- Срок реализации товара (Срок_реализации)

Отметим, что с использованием оператора AS столбец «Отпускная_цена» получает собственное имя в результате выполнения запроса.

При нажатии на кнопку «Печать» в данном диалоговом окне – распечатается данная таблица.

При нажатии на кнопку «Поставщики» на главном диалоговом окне откроется новое диалоговое окно «Информация о поставщиках», где отобразится таблица «Поставщики» и при вводе в поле «Ведите код поставщика:» «Код поставщика» отобразится информация о том, какие товары он поставляет, данные кнопки имеют в коде SQL-запросы (Рисунок 35):

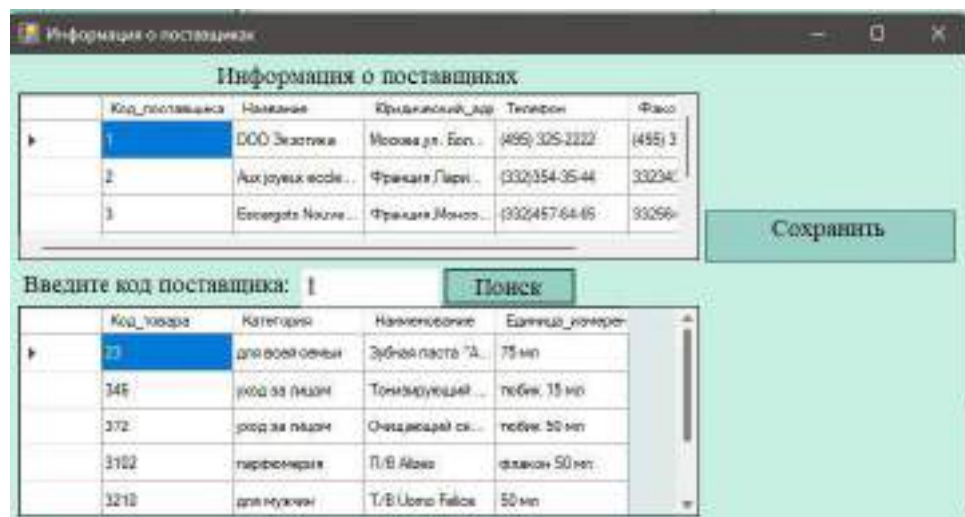


Рисунок 35 – Диалоговое окно «Информация о поставщиках»

Запрос на кнопке «Сохранить»:

```
SqlDataAdapter adapter = new SqlDataAdapter(«SELECT * FROM Поставщики»,  
connection);
```


Этот код создает объект SqlDataAdapter, который будет использоваться для получения данных из таблицы Поставщики базы данных, находящейся в соединении connection.

SqlDataAdapter - это объект, который используется для заполнения DataSet и обновления базы данных в зависимости от изменений, внесенных в DataSet. В данном случае он используется только для заполнения DataSet.

Конструктор SqlDataAdapter принимает два параметра: строку запроса для выборки данных и соединение с базой данных. В данном случае строка запроса «SELECT * FROM Поставщики» выбирает все строки из таблицы Поставщики.

Запрос на кнопке «Поиск»:

```
string query = $"SELECT Код_товара, Категория, Наименование, Единица_измерения FROM Товары WHERE Код_поставщика = {supplierId}»;
```

Этот запрос выбирает из таблицы «Товары» информацию о товарах, у которых значение поля «Код_поставщика» равно заданному значению «supplierId». Запрос использует интерполяцию строк, чтобы вставить значение «supplierId» в строку запроса. Выбранные поля - «Код_товара», «Категория», «Наименование» и «Единица_измерения».

При нажатии на кнопку «Оформление заказов» на главном диалоговом окне откроется новое диалоговое окно «Оформление заказа», где отобразится таблица «Заказы». В данной форме при выборе из поля «Выберите код товара:» будут отображаться «Код_товара» из таблицы «Товары» и при выборе будут отображаться данные о нем: Название, Кол-во на складе, цена за шт. (все эти данные появляются автоматически из таблицы товары по полю «Код_товара»). При заполнении поля «Выберите кол-во» (если выбрать больше, чем на складе, то появится диалоговое окно, которое будет сообщать о том, что выбранное количество не может превышать количество на складе) поле «Стоимость» будет автоматически заполнено. Затем нажимаем кнопку «Добавить в таблицу «Заказы» и ждем «Обновить данные», после этого таблица будет заполнена (Рисунок 36):

Код_товара	Количество	Цена	Стоимость
172	21	81.000	1863.000

Выберите код товара: 372

Наименование: Спичечный скраб

Кол-во на складе: 21

Цена за шт.: 81

Выберите кол-во: 23

Стоимость: 1863,00

Добавить в таблицу "Заказы" Обновить данные Удалить данные

Оформление счета: 21.04.2023

Укажите номер сотрудника: 8

☒ Дисконтная карта

Напишите число скидок(если есть дисконтная карта), иначе оставьте поле пустым: 45

Добавить в таблицу "Счета" Открыть таблицу "Счета"

Рисунок 36 – Диалоговое окно «Оформление заказа»

Также есть возможность удалить данные в таблице «Заказы», для этого нужно выбрать строку (нажать в таблице) и нажать на кнопку «Удалить данные».

Весь вышеописанный функционал кнопок и полей имеют следующие SQL-запросы в коде:

Поле «Выберите код товара»:

```
e) private void comboBox1_SelectedIndexChanged(object sender, EventArgs
{
    // При выборе кода товара в ComboBox1 отображаем его название в Label2
    try
    {
        connection.Open();
        SqlCommand cmd = new SqlCommand(«SELECT Наименование FROM Товары WHERE
Код_товара = @Код_товара», connection);
        cmd.Parameters.AddWithValue(«@Код_товара»,
comboBox1.SelectedItem.ToString());
        SqlDataReader reader = cmd.ExecuteReader();
        if (reader.Read())
        {
            label2.Text = reader[«Наименование»].ToString();
        }
        else
        {
            label2.Text = «;
        }
        reader.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(«Ошибка при загрузке данных из базы данных: « +
ex.Message);
    }
    finally
    {
        connection.Close();
    }
    try
    {
        connection.Open();
        SqlCommand cmd = new SqlCommand(«SELECT Наименование,
Количество_на_складе, Цена_закупочная FROM Товары WHERE Код_товара =
@Код_товара», connection);
        cmd.Parameters.AddWithValue(«@Код_товара»,
comboBox1.SelectedItem.ToString());
        SqlDataReader reader = cmd.ExecuteReader();
        if (reader.Read())
        {
            label2.Text = reader[«Наименование»].ToString();
            label3.Text = reader[«Количество_на_складе»].ToString();
        }
    }
}
```

```

        label4.Text = reader[«Цена_закупочная»].ToString();
    }
    else
    {
        label2.Text = «;
        label3.Text = «;
        label4.Text = «;
    }
    reader.Close();
}
catch (Exception ex)
{
    MessageBox.Show(«Ошибка при загрузке данных из базы данных: « +
ex.Message);
}
finally
{
    connection.Close();
}
}

```

Данный код отслеживает событие «SelectedIndexChanged» (изменение выбранного элемента) в объекте ComboBox1 и выполняет два запроса к базе данных, чтобы отобразить информацию о выбранном товаре в трех разных Label'ах.

В первом блоке try-catch выполняется запрос к базе данных для получения наименования выбранного товара. Запрос содержит параметр «@Код_товара», который задается значением выбранного элемента ComboBox1. Если запрос выполнен успешно, то наименование товара записывается в Label2.

Во втором блоке try-catch выполняется запрос к базе данных для получения дополнительной информации о выбранном товаре (количество на складе и цена закупки). Запрос также содержит параметр «@Код_товара», заданный значением выбранного элемента ComboBox1. Если запрос выполнен успешно, то полученные данные записываются в Label3 и Label4 соответственно.

Если запросы выполнены неуспешно, то отображается соответствующее сообщение об ошибке. В блоке finally происходит закрытие подключения к базе данных в любом случае.

Поле «Стоимость»:

```

private void textBox2_TextChanged(object sender, EventArgs e)
{
    float availableQuantity = Convert.ToSingle(label3.Text); // Получаем
доступное количество товара из label3
    float enteredQuantity = 0;
    if (!string.IsNullOrEmpty(textBox2.Text))
    {
        enteredQuantity = Convert.ToSingle(textBox2.Text); // Получаем
введенное количество товара из textBox2
    }
}

```



```

        if (enteredQuantity > availableQuantity)
        {
            MessageBox.Show(«Вы не можете указать количество товара больше, чем на
            складе!»);
            textBox2.Text = availableQuantity.ToString(); // Устанавливаем в
            textBox2 максимально доступное количество товара
        }

        float currentQuantity = 0;
        if (float.TryParse(label4.Text.Replace(«,», «.»), out
        currentQuantity)) // Преобразуем значение в label4 в число, заменяя запятую
        на точку
        {
            float multipliedQuantity = currentQuantity * enteredQuantity; //
            Выполняем умножение

            textBox4.Text = multipliedQuantity.ToString(«#,##0.00»); // Записываем
            результат в textBox4 с форматом «--,--»
        }
        else
        {
            // Выводим сообщение об ошибке
            MessageBox.Show(«Значение в label4 не может быть преобразовано в
            число.»);
        }
    }

    private void textBox4_TextChanged(object sender, EventArgs e)
    {
        float currentQuantity = 0;
        if (float.TryParse(label4.Text, out currentQuantity)) // Преобразуем
        значение в label4 в число
        {
            float enteredQuantity = 0;
            if (float.TryParse(textBox2.Text, out enteredQuantity)) // Преобразуем
            значение в textBox2 в число
            {

            }
        }
        else
        {
            // Выводим сообщение об ошибке
            MessageBox.Show(«Вы ввели некорректное значение в textBox2. Пожалуйста,
            введите число.»);
        }
    }

    else
    {
        // Выводим сообщение об ошибке
        MessageBox.Show(«Значение в label4 не может быть преобразовано в
        число.»);
    }
}

```

```
}
```

Данный код относится к обработке событий изменения текста в текстовых полях формы.

Метод `textBox2_TextChanged` вызывается при изменении текста в текстовом поле `textBox2`. Он считывает количество товара, доступное на складе, из `label3` и введенное пользователем количество из `textBox2`. Если введенное количество больше доступного, то выводится сообщение об ошибке и в `textBox2` устанавливается максимально доступное количество.

Затем метод преобразует значение в `label4` в число с помощью метода `float.TryParse`, заменяя запятую на точку, если это необходимо, и выполняет умножение этого числа на введенное количество товара. Результат записывается в `textBox4` с форматированием числа в виде «--,--».

Метод `textBox4_TextChanged` вызывается при изменении текста в текстовом поле `textBox4`. Он также преобразует значение в `label4` в число, затем преобразует значение в `textBox2` в число и выполняет умножение этих чисел.

Кнопка «Добавить в таблицу «Заказы»:

```
private void button1_Click(object sender, EventArgs e)
{
    // Проверяем, что выбран элемент в comboBox1
    if (comboBox1.SelectedItem != null)
    {
        // Получаем выбранный элемент в comboBox1
        string selectedProduct = comboBox1.SelectedItem.ToString();

        // Проверяем, что введено значение в textBox2
        if (!string.IsNullOrEmpty(textBox2.Text))
        {
            // Парсим значение из textBox2 в число
            if (float.TryParse(textBox2.Text, out float enteredQuantity))
            {
                // Проверяем, что значение в label4 может быть преобразовано в число
                if (float.TryParse(label4.Text, out float currentPrice))
                {
                    // Вычисляем стоимость
                    float totalCost = enteredQuantity * currentPrice;

                    // Проверяем, что введено значение в textBox4
                    if (!string.IsNullOrEmpty(textBox4.Text))
                    {
                        // Парсим значение из textBox4 в число
                        if (float.TryParse(textBox4.Text, out float enteredTotalCost))
                        {
                            // Добавляем данные в таблицу «Заказы»
                            string sql = «INSERT INTO Заказы (Код_товара, Количество, Цена,
                                Стоимость) VALUES (@Код_товара, @Количество, @Цена, @Стоимость)»;
                        }
                    }
                }
            }
        }
    }
}
```

```

        using (SqlConnection connection = new SqlConnection(«Data
Source=DESKTOP-8OFF3I4\\SQLEXPRESS02;Initial
Catalog=Trading_company;Integrated Security=True»))
        {
            using (SqlCommand command = new SqlCommand(sql, connection))
            {
                // Добавляем параметры в команду
                command.Parameters.AddWithValue(«@Код_товара», selectedProduct);
                command.Parameters.AddWithValue(«@Количество», enteredQuantity);
                command.Parameters.AddWithValue(«@Цена», currentPrice);
                command.Parameters.AddWithValue(«@Стоимость», enteredTotalCost);

                // Открываем соединение
                connection.Open();

                // Выполняем команду
                command.ExecuteNonQuery();

                // Закрываем соединение
                connection.Close();
            }
        }

        // Выводим сообщение об успешном добавлении данных
        MessageBox.Show(«Данные успешно добавлены в таблицу 'Заказы'.»);
    }
    else
    {
        // Выводим сообщение об ошибке
        MessageBox.Show(«Вы ввели некорректное значение в textBox4. Пожалуйста,
введите число.»);
    }
}
else
{
    // Выводим сообщение об ошибке
    MessageBox.Show(«Значение в textBox4 не может быть пустым.»);
}
}
else
{
    // Выводим сообщение об ошибке
    MessageBox.Show(«Значение в label4 не может быть преобразовано в
число.»);
}
}
else
{
    // Выводим сообщение об ошибке
    MessageBox.Show(«Вы ввели некорректное значение в textBox2. Пожалуйста,
введите число.»);
}
}

```

```

    }
    }
    else
    {
        // Выводим сообщение об ошибке
        MessageBox.Show(«Введите количество !»);
    }
    }
    else
    {
        // Выводим сообщение об ошибке
        MessageBox.Show(«Необходимо выбрать товар из списка.»);
    }
    }
}

```

Этот код является обработчиком события нажатия кнопки button1. Когда пользователь нажимает на кнопку, код выполняет ряд проверок и действий, связанных с добавлением заказа в базу данных. Рассмотрим код по шагам:

1) Проверяем, выбран ли элемент в comboBox1. Если не выбран, выводим сообщение об ошибке и выходим из метода.

2) Получаем выбранный элемент в comboBox1 и сохраняем его в переменную selectedProduct.

3) Проверяем, что введено значение в textBox2. Если не введено, выводим сообщение об ошибке и выходим из метода.

4) Парсим значение из textBox2 в число и сохраняем его в переменную enteredQuantity. Если введено некорректное значение, выводим сообщение об ошибке и выходим из метода.

5) Проверяем, что значение в label4 может быть преобразовано в число. Если не может быть, выводим сообщение об ошибке и выходим из метода.

6) Вычисляем стоимость заказа, умножив введенное количество (enteredQuantity) на цену товара (currentPrice), сохраненную в label4.

7) Проверяем, что введено значение в textBox4. Если не введено, выводим сообщение об ошибке и выходим из метода.

8) Парсим значение из textBox4 в число и сохраняем его в переменную enteredTotalCost. Если введено некорректное значение, выводим сообщение об ошибке и выходим из метода.

9) Создаем строку запроса SQL для добавления данных в таблицу «Заказы» с помощью параметров.

10) Устанавливаем соединение с базой данных и выполняем команду, используя параметры, добавленные в шаге 9.

11) Если команда выполнена успешно, выводим сообщение об успешном добавлении данных в таблицу «Заказы».

12) Если команда не выполнена успешно, выводим сообщение об ошибке.

13 Закрываем соединение с базой данных.

Примечание:

«Парсить» (от английского parse) означает преобразование данных из одного формата в другой или из строки в числовой или другой тип данных. В программировании, это часто используется для преобразования пользовательского ввода (например, введенного пользователем значения в текстовое поле) в нужный формат данных, который можно использовать для дальнейшей обработки в приложении. Например, в коде, который мы предоставили, парсинг используется для преобразования введенного пользователем значения в числовой тип данных.

Кнопка «Обновить данные»:

```
private void button4_Click(object sender, EventArgs e)
{
    using (SqlConnection connection = new SqlConnection(«Data
Source=DESKTOP-8OFF3I4\\SQLEXPRESS02;Initial
Catalog=Trading_company;Integrated Security=True»))
    {
        connection.Open();
        using (SqlDataAdapter adapter = new SqlDataAdapter(«SELECT * FROM
Заказы», connection))
        {
            // Создаем новый DataTable
            DataTable dt = new DataTable();
            // Заполняем DataTable данными из таблицы «Заказы»
            adapter.Fill(dt);
            // Привязываем DataTable к DataGridView
            dataGridView1.DataSource = dt;
        }
    }
}
```

Данный код отвечает за отображение данных из таблицы «Заказы» в DataGridView компоненте на форме приложения.

Сначала создается новый экземпляр класса SqlConnection, который устанавливает соединение с базой данных Trading_company. Затем происходит открытие соединения и создание нового экземпляра класса SqlDataAdapter для выполнения запроса на выборку данных из таблицы «Заказы».

Затем создается новый экземпляр класса DataTable, который будет использоваться для хранения выбранных данных из таблицы. Выбранные данные из таблицы «Заказы» заполняются в DataTable с помощью метода Fill класса SqlDataAdapter.

Наконец, DataGridView компонент на форме привязывается к DataTable, чтобы отобразить выбранные данные из таблицы «Заказы». После завершения работы с базой данных, соединение закрывается.

Кнопка «Удалить данные»:

```
private void button5_Click(object sender, EventArgs e)
{
```

```

// Проверяем, есть ли выделенные строки в DataGridView
if (dataGridView1.SelectedRows.Count > 0)
{
    // Если есть выделенные строки, то удаляем их из таблицы «Заказы»
    foreach (DataGridViewRow row in dataGridView1.SelectedRows)
    {
        // Получаем значение в ячейке «Код_товара» текущей строки
        int orderId = Convert.ToInt32(row.Cells[«Код_товара»].Value);
        // Удаляем строку из таблицы «Заказы» по «Код_товара»
        DeleteOrder(orderId);
    }
}
else
{
    // Если нет выделенных строк, то выводим сообщение
    MessageBox.Show(«Выделите строку(и) для удаления.»);
}

// Обновляем таблицу «Заказы» после удаления
UpdateOrderTable();
}

```

Этот код обрабатывает клик по кнопке «Удалить» и удаляет выделенные строки из таблицы «Заказы».

Вначале происходит проверка, есть ли выделенные строки в DataGridView. Если есть, то для каждой выделенной строки получаем значение в ячейке «Код_товара» и удаляем строку из таблицы «Заказы» по этому значению, используя метод DeleteOrder.

Если выделенных строк нет, то выводится сообщение об ошибке.

После удаления строк из таблицы «Заказы», таблица обновляется вызовом метода Update Order Table(), который обновляет DataGridView.

Теперь перейдем во вторую нижнюю часть (после черной линии) диалогового окна «Оформление заказа» (Рисунок 37):

Оформление счета:
 21.04.2023
 Укажите номер сотрудника: [dropdown]
☒ Дисконтная карта
 Напишите число скидки (если есть дисконтная карта), иначе оставьте поле пустым: 45
 [Добавить в таблицу "Счета"] [Открыть таблицу "Счета"]

Рисунок 37 – Вторая часть диалогового окна «Оформление заказа»

В данном диалоговом окне указывается настоящая дата, в поле «Укажите номер сотрудника» отображается поле «Код_сотрудника» из таблицы «Сотрудники». При нажатии на «Дисконтную карту» необходимо указывать число (например, 45 это 45%). После того, как указали данные, можем нажимать на

кнопку «Добавить в таблицу «Счета» и открыть ее с помощью кнопки «Открыть таблицу «Счета», где будут отображаться все данные.

Весь вышеописанный функционал кнопок и полей имеют следующие SQL-запросы в коде:

Поле даты:

```
private void label8_Click(object sender, EventArgs e)
{
    label8.Text = DateTime.Now.ToString("dd.MM.yyyy");
}
```

Этот код относится к обработчику события «Click» для элемента управления Label с именем «label8». Когда пользователь кликает на этот элемент, выполняется код, который устанавливает значение текста этого элемента как текущую дату в формате «dd.MM.yyyy». То есть это действие обновляет текст, отображаемый в label8, на текущую дату.

Кнопка «Добавить в таблицу «Счета»:

```
private void button2_Click(object sender, EventArgs e)
{
    // Проверяем заполнение label10_Click
    if (string.IsNullOrEmpty(label10.Text))
    {
        MessageBox.Show("Пожалуйста, укажите дату!");
        return;
    }

    // Получаем значения из элементов управления
    string номерЧека = textBox1.Text;
    int кодСотрудника;
    if (comboBox3.SelectedValue != null &&
        int.TryParse(comboBox3.SelectedValue.ToString(), out кодСотрудника))
    {
        DateTime дата = DateTime.Parse(label8.Text);
        bool дисконтнаяКарта = checkBox1.Checked;
        decimal скидка = 0m;
        if (дисконтнаяКарта)
        {
            скидка = Convert.ToDecimal(textBox3.Text);
        }

        // Считываем значение стоимости из таблицы «Заказы»
        decimal стоимость = 0m;
        foreach (DataGridViewRow row in dataGridView1.Rows)
        {
            if (!row.IsNewRow)
            {
                стоимость += Convert.ToDecimal(row.Cells["«Стоимость»"].Value);
            }
        }
    }
}
```

```

}

// Вычисляем сумму к оплате с учетом скидки
decimal суммаКОплате = 0m;
if (дисконтнаяКарта)
{
    суммаКОплате = стоимость - (стоимость * скидка / 100);
}
else
{
    суммаКОплате = стоимость;
}

// Выполняем вставку данных в таблицу «Счета»
using (SqlConnection connection = new SqlConnection(«Data Source=DESKTOP-
8OFF3I4\SQLEXPRESS02;Initial Catalog=Trading_company;Integrated
Security=True»))
{
    connection.Open();
    using (SqlCommand command = new SqlCommand(«INSERT INTO Счета
(Код_сотрудника, Дата, Дисконтная_карта, Скидка, Сумма_к_оплате) VALUES
(@КодСотрудника, @Дата, @ДисконтнаяКарта, @Скидка, @СуммаКОплате)»,
connection))
    {
        command.Parameters.AddWithValue(«@КодСотрудника», кодСотрудника);
        command.Parameters.AddWithValue(«@Дата», дата);
        command.Parameters.AddWithValue(«@ДисконтнаяКарта», дисконтнаяКарта);
        command.Parameters.AddWithValue(«@Скидка», скидка);
        command.Parameters.AddWithValue(«@СуммаКОплате», суммаКОплате);
        command.ExecuteNonQuery();
    }
}

MessageBox.Show(«Данные успешно сохранены в таблицу 'Счета'.»);
}
else
{
    MessageBox.Show(«Пожалуйста, выберите сотрудника и убедитесь, что значение
является числом»);
}
}

```

Данный код обрабатывает нажатие на кнопку «Сохранить» в форме. Он выполняет следующие действия:

- 1) Проверяет, заполнена ли дата покупки в label10. Если дата не заполнена, выводит сообщение об ошибке и прекращает выполнение кода.
- 2) Получает значения из элементов управления формы: номер чека из textBox1, код сотрудника из comboBox3, дату покупки из label8, флаг дисконтной карты из checkBox1, значение скидки из textBox3 (если дисконтная карта выбрана) и считывает значения стоимости из таблицы «Заказы».

3) Вычисляет сумму к оплате, учитывая скидку, если дисконтная карта выбрана.

4) Выполняет вставку данных в таблицу «Счета» в базе данных.

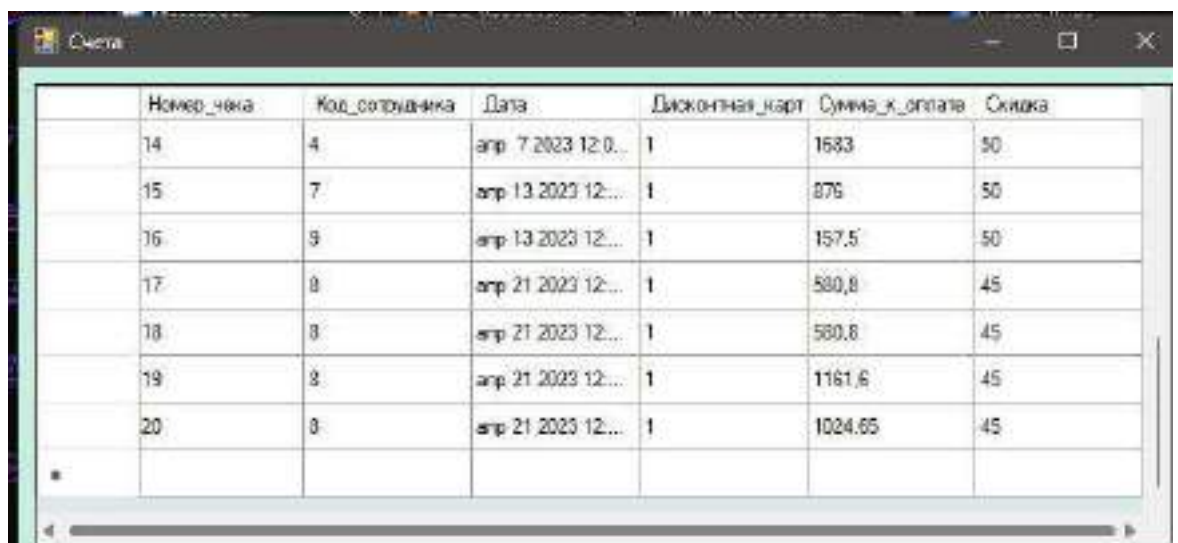
5) Подключается к базе данных Trading_company на сервере DESKTOP-8OFF3I4\SQLEXPRESS02, используя Windows-аутентификацию.

Используя SqlCommand, выполняет операцию INSERT, вставляя значения кода сотрудника, даты покупки, флага дисконтной карты, значения скидки и суммы к оплате в соответствующие поля таблицы «Счета».

5) Выводит сообщение об успешном сохранении данных в таблицу «Счета».

Если код сотрудника не выбран или не является числом, выводит сообщение об ошибке.

После того, как мы указали все данные в форме «Оформление заказа» и нажали кнопку «Добавить в таблицу «Счета» - мы можем открыть таблицу «Счета» с помощью кнопки «Открыть таблицу «Счета» и посмотреть на нее (рисунок 38):



Номер_чека	Код_сотрудника	Дата	Дисконтная_карт	Сумма_к_оплате	Скидка
14	4	апр 7 2023 12:0...	1	1633	50
15	7	апр 13 2023 12:...	1	876	50
16	9	апр 13 2023 12:...	1	157,5	50
17	8	апр 21 2023 12:...	1	580,8	45
18	8	апр 21 2023 12:...	1	580,8	45
19	8	апр 21 2023 12:...	1	1161,6	45
20	8	апр 21 2023 12:...	1	1024,65	45

Рисунок 38 – Диалоговое окно «Счета»

При открытии данной формы срабатывает следующий код с SQL-запросом:

```
private void Form10_Load(object sender, EventArgs e)
{
    try
    {
        using (SqlConnection connection = new SqlConnection(«Data
Source=DESKTOP-8OFF3I4\\SQLEXPRESS02;Initial
Catalog=Trading_company;Integrated Security=True»))
        {
            connection.Open();
            using (SqlCommand command = new SqlCommand(«SELECT * FROM Счета»,
connection))
            {
                SqlDataAdapter adapter = new SqlDataAdapter(command);
                DataTable dt = new DataTable();
                adapter.Fill(dt);
            }
        }
    }
}
```

```

dataGridView1.DataSource = dt;
}
}
}
catch (Exception ex)
{
    MessageBox.Show(«Ошибка при загрузке данных из базы данных: « +
ex.Message);
}
}

```

Этот код выполняется при загрузке формы (Form10). Он устанавливает соединение с базой данных, создает объект SqlCommand с запросом на выборку всех данных из таблицы «Счета» и использует объект SqlDataAdapter для заполнения объекта DataTable. Затем результаты выборки отображаются в DataGridView на форме.

Если возникает исключение во время выполнения кода (например, ошибка подключения к базе данных), то обработчик исключений отображает сообщение об ошибке в окне MessageBox.

При нажатии на кнопку «Сведения о сотрудниках» откроется новое диалоговое окно «Информация о сотрудниках», которое будет содержать таблицу «Сотрудники» и две кнопки: «Обновить данные» и «Сохранить изменения» (Рисунок 39):



Рисунок 39 – Диалоговое окно «Информация о сотрудниках»

В данном диалоговом окне можно добавлять или изменять данные прямо в таблице, после чего нажать на кнопку «Сохранить изменения».

Код с SQL-запросами данной формы имеет следующий вид:

```

private void Form11_Load(object sender, EventArgs e)
{
    try
    {
        using (SqlConnection connection = new SqlConnection(«Data
Source=DESKTOP-8OFF3I4\SQLEXPRESS02;Initial
Catalog=Trading_company;Integrated Security=True»))
        {
            connection.Open();

```

```

        using (SqlCommand command = new SqlCommand(«SELECT * FROM Сотрудники»,
connection))
        {
            SqlDataAdapter adapter = new SqlDataAdapter(command);
            DataTable dt = new DataTable();
            adapter.Fill(dt);
            dataGridView1.DataSource = dt;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(«Ошибка при загрузке данных из базы данных: « +
ex.Message);
    }
}

private void button1_Click(object sender, EventArgs e)
{
    // Сохраняем изменения в таблице «Сотрудники»
    SqlConnection connection = new SqlConnection(«Data Source=DESKTOP-
80FF3I4\\SQLEXPRESS02;Initial Catalog=Trading_company;Integrated
Security=True»);
    SqlDataAdapter adapter = new SqlDataAdapter(«SELECT * FROM Сотрудники»,
connection);
    SqlCommandBuilder builder = new SqlCommandBuilder(adapter);
    adapter.Update((DataTable)dataGridView1.DataSource);
    // выводим сообщение об успешном сохранении
    MessageBox.Show(«Данные успешно сохранены!»);
}

private void button2_Click(object sender, EventArgs e)
{
    string connectionString = «Data Source=DESKTOP-
80FF3I4\\SQLEXPRESS02;Initial Catalog=Trading_company;Integrated
Security=True»;
    string query = «SELECT * FROM Сотрудники»;
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        SqlDataAdapter adapter = new SqlDataAdapter(query, connection);
        DataSet ds = new DataSet();
        adapter.Fill(ds, «Сотрудники»);
        dataGridView1.DataSource = ds.Tables[«Сотрудники»];
    }
}

```

Код разделен на три части, каждая связана с нажатием кнопки:

1) Метод `Form1_Load` выполняется при загрузке формы и заполняет таблицу `dataGridView1` данными из таблицы «Сотрудники» в базе данных.

2) Метод `button1_Click` сохраняет изменения, внесенные в таблицу `dataGridView1`, обратно в базу данных. Для этого создается экземпляр класса `SqlConnection` для подключения к базе данных, а затем создается экземпляр класса `SqlDataAdapter` для выборки данных из таблицы «Сотрудники». Далее создается экземпляр класса `SqlCommandBuilder`, который генерирует команды SQL для обновления таблицы. Наконец, вызывается метод `Update` объекта `adapter` с источником данных из таблицы `dataGridView1`, чтобы сохранить изменения в базе данных.

3) Метод `button2_Click` выбирает все данные из таблицы «Сотрудники» и отображает их в `dataGridView1`. Для этого создается экземпляр класса `SqlConnection` для подключения к базе данных, затем создается экземпляр класса `SqlDataAdapter` для выборки данных из таблицы «Сотрудники», и наконец, данные сохраняются в `DataSet`, который затем используется для отображения данных в `dataGridView1`.

6.3 Тестирование разработанного приложения

Для тестирования программы производились различные манипуляции с данными.

Тест №1

Рассмотрим в качестве теста все функции, которые есть в диалоговом окне «Товары», которое открывается через главное диалоговое окно с помощью кнопки «Информация о товарах».

Итак. Запускаем программу (Рисунок 40):

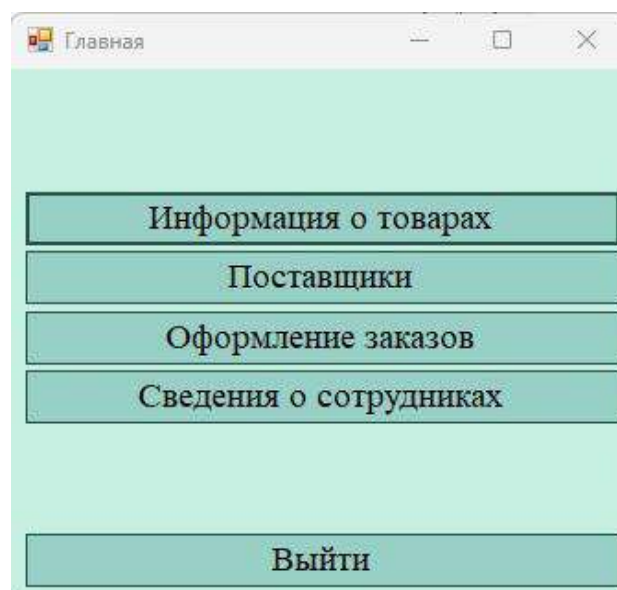


Рисунок 40 – Главное диалоговое окно

Нажимаем на кнопку «Информация о товарах» и открываем окно «Товары» с таблицей «Товары» и листаем таблицу в конец (Рисунок 41):



Рисунок 41 – Окно «Товары»

Добавим в 33 строку таблицы новые данные, где Поле «№» заполняется автоматически (Рисунок 42):

33	12321	уход за волосами	Шампунь "Шам"	50мл	100	28.10.2025	1
----	-------	------------------	---------------	------	-----	------------	---

Рисунок 42 – Новые данные в 33 строке таблицы «Товары»

Далее нажимаем на кнопку «Сохранить изменения» и появляется диалоговое окно, которое сообщает о том, что данные успешно сохранены (Рисунок 43):

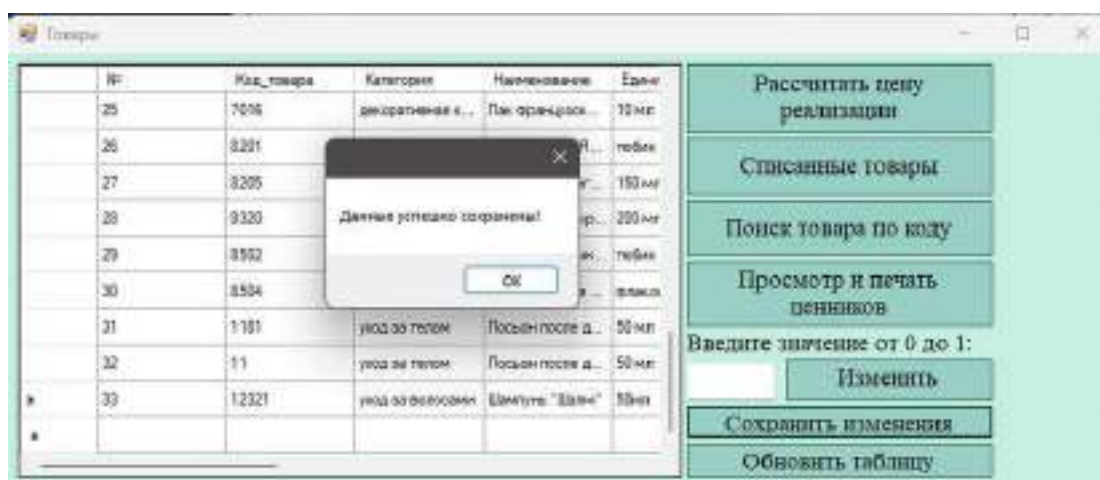


Рисунок 43 – Успешное добавление новых данных

Изменим столбец «Наценка,%» в каждой строке одним нажатием. Для этого введем число 0.5 (50% соответственно) в поле и нажмем на кнопку «Изменить» (Рисунок 44):

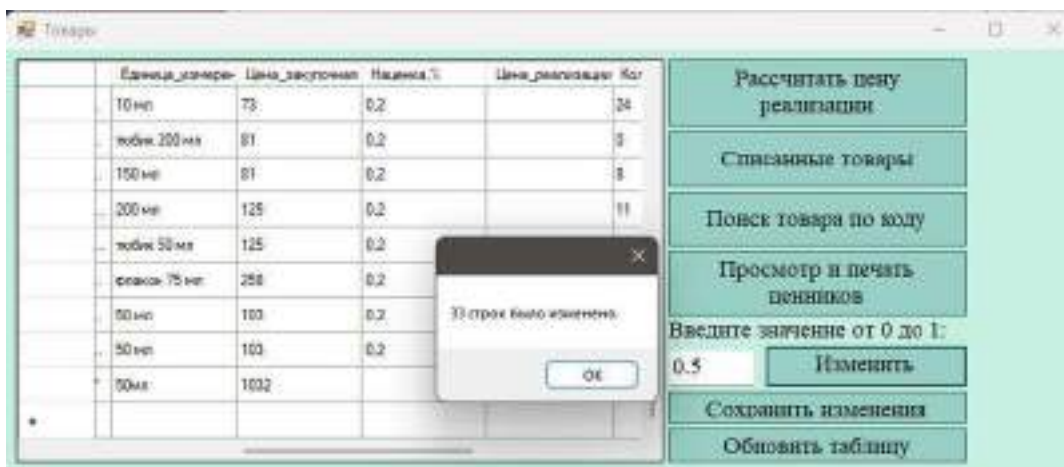


Рисунок 44 – Изменение всех строк поля «Наценка,%»

Обновляем таблицу и видим, что значения в поле «Наценка,%» изменились (Рисунок 45):



Рисунок 45 – Изменение всех строк поля «Наценка,%»

Рассчитаем теперь цену реализации и проверим правильность расчета с помощью калькулятора. Для этого нажмем на кнопку «Рассчитать цену реализации» (Рисунок 46):



Рисунок 46 – Диалоговое окно «Расчет цены реализации»

Проверим, правильно ли рассчитаны значения в поле «Цена_реализации». Для проверки возьмем 5 строку поле «Цена_закупочная» умножим на «Наценка,%»

и прибавим «Цена_закупочная». Как можем заметить, расчёты были проведены верно (Рисунок 47):

№	Код_товара	Категория	Наименование	Единица_измерения	Цена_закупочная	Наценка, %	Цена_реализации
1	12	уход за телом	Антицеллюлитн...	200 мл	132	0,5	198
2	23	для всей семьи	Зубная паста "А...	75 мл	88	0,5	99
3	346	уход за лицом	Тонизирующий л...	тобик 15 мл	88	0,5	132
4	372	уход за лицом	Очищающий скр...	тобик 50 мл	81	0,5	121,5
5	1167	уход за лицом	Успокаивающ...	флакон 50 мл	88	0,5	132

DEG RAD $88 \times 0,5 + 88 = 132$

Рисунок 47 – Проверка расчета

Нажмем теперь на кнопку «Списанные товары», где должны отображаться те товары, у которых поле «Срок_реализации» меньше сегодняшней даты, на данный момент дата 26.04.2023 (Рисунок 48):

№	Код_товара	Категория	Наименование	Единица_измерения	Цена_закупочная	Наценка, %	Цена_реализации	Количество_на_ск	Срок_реализации	Код_пошива
1	23	для всей семьи	Зубная паста "А...	75 мл	88	0,5	99	12	19.03.2023	1
2	1178	уход за лицом	Маска увлажня...	тобик 30 мл	120	0,5	165	16	20.01.2021	3
3	1281	для всей семьи	Заживляющее...	тобик 10 мл	88	0,5	92	72	20.01.2019	4
4	1282	для всей семьи	Заживляющее...	тобик 75 мл	72	0,5	75	29	20.01.2020	3
14	3738	парфюмерия	ПФ Doria Parfa...	50 мл	168	0,5	176	16	19.03.2023	4
16	2043	для мужчин	ПФ Doria Parfa...	50 мл	140	0,5	147	28	12.06.2022	1
17	1079	деodorant	Стайки туфта...	20	125	0,5	130	2	18.04.2020	1
28	1029	уход за телом	Питательный кр...	200 мл	125	0,5	130	11	12.02.2021	3
32	11	уход за телом	Посып после д...	50 мл	100	0,5	105	21	20.01.2020	3

Рисунок 48 – Диалоговое окно «Списанные товары»

Проверим кнопку «Поиск товара по коду» и откроем диалоговое окно «Поиск товара» (Рисунок 49):

Цена_реализации	Количество_на_ск	Срок_реализации	Код_пошива
	10	12.03.2025	3
	12	12.09.2023	1
	5	15.06.2024	1

Введите номер товара: Поиск

Введите значение от 0 до 1: 0.5 Изменить

Сохранить изменения Обновить таблицу

Рисунок 49 – Диалоговое окно «Поиск товара»

Введем в поле число 12 и нажмем на кнопку «Поиск», после чего отобразится информация о товаре с данным номером или полем «Код_товара» из таблицы «Товары» (Рисунок 50):

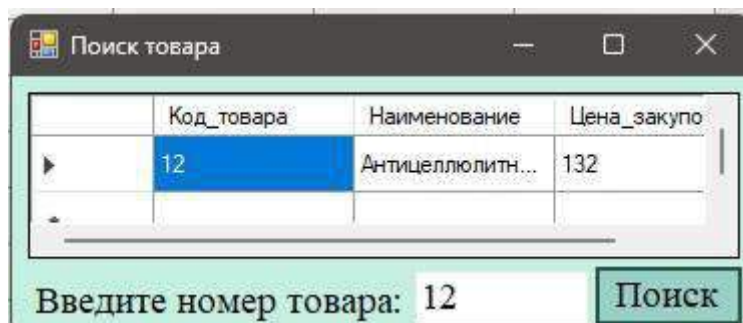


Рисунок 50 – Диалоговое окно «Поиск товара»

Проверим, что будет, если ввести номер товара, которого не существует. Результатом будет новое диалоговое окно, которое сообщит, что данного товара не существует (Рисунок 51):



Рисунок 51 – Сообщение «Товар не найден»

Тест №2

Рассмотрим в качестве теста все функции, которые есть в диалоговом окне «Информация о поставщиках», которое открывается через главное диалоговое окно с помощью кнопки «Поставщики».

Нажимаем на главном окне кнопку «Поставщики» и открывается новое диалоговое окно «Информация о поставщиках», где будет отображаться таблица «Поставщики» (Рисунок 52):

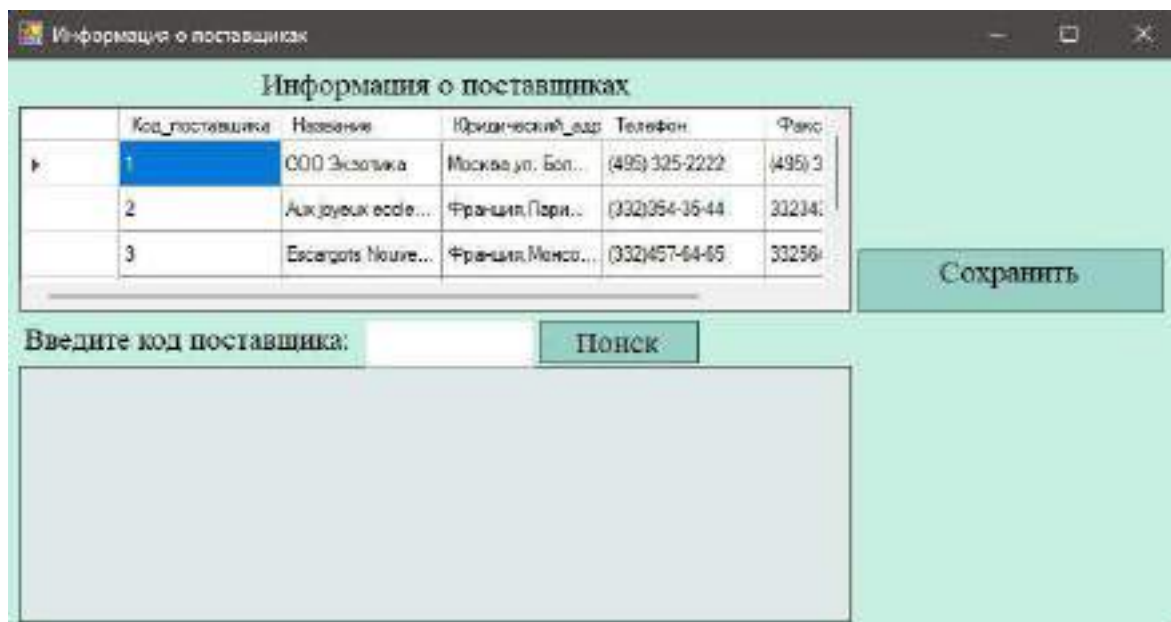


Рисунок 52 – Диалоговое окно «Информация о поставщиках»

Добавим в главную таблицу новые данные в пятую строку и нажмем на кнопку сохранить. Как мы видим, данные успешно сохранены (Рисунок 53):

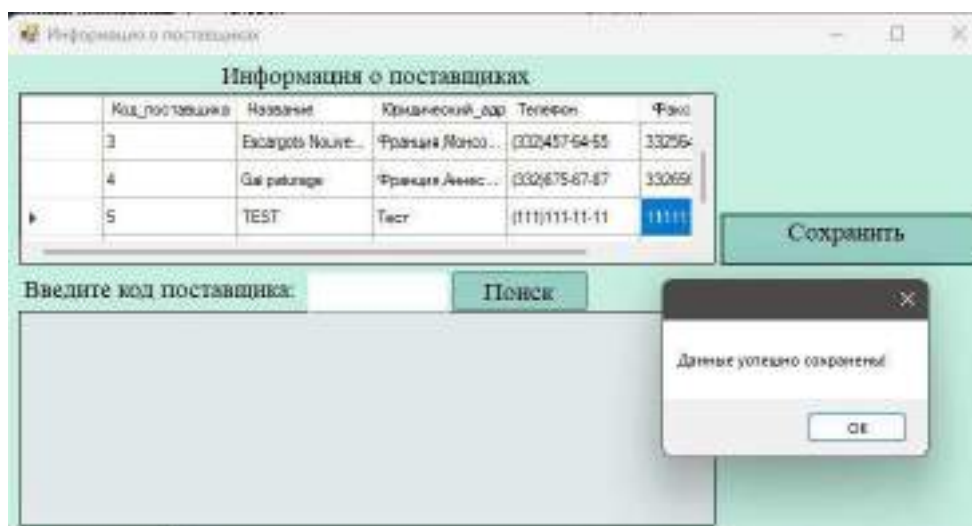


Рисунок 53 – Успешное добавление новых данных

Введем в поле код поставщика из таблицы «Поставщики», после чего отобразится новая таблица, где будет информация о том, какой товар поставщик поставляет:

3	Escargots Nouve...	Франция, Монсо...	(332)457-64-65	332564
---	--------------------	-------------------	----------------	--------

Введите код поставщика: 3 Поиск

	Код_товара	Категория	Наименование	Единица_измерен
▶	12	уход за телом	Антицеллюлитн...	200 мл
	1202	для всей семьи	Защитный крем...	тюбик 75 мл
	1452	для мужчин	Гель-скраб для ...	150 мл
	2018	уход за телом	Молочко для те...	150 мл
	3123	парфюмерия	T/B Violet	50 мл

Рисунок 54 – Информация о поставщике в таблице

Тест №3

Рассмотрим в качестве теста все функции, которые есть в диалоговом окне «Оформление заказа», которое открывается через главное диалоговое окно с помощью кнопки «Оформление заказов».

Откроется диалоговое окно «Оформление заказа», где имеется таблица «Заказы» (Рисунок 55):

Оформление заказа

Заказ:

	Код_товара	Количество	Цена	Стоимость
▶				

Выберите код товара:

Название:

Кол-во на складе:

Цена за шт.:

Выберите кол-во:

Стоимость:

Добавить в таблицу "Заказы"

Обновить данные

Удалить данные

Оформление счета:

-->Нажмите, чтобы указать дату <--

Укажите номер сотрудника: 1

☐ Дисконтная карта

Напишите число скидки(если есть дисконтная карта), иначе оставьте поле нулевым: 0

Добавить в таблицу "Счета"

Открыть таблицу "Счета"

Рисунок 55 – Диалоговое окно «Оформление заказа»

Оформим заказ, для начала выберем код товара из уже имеющихся (Рисунок 56):

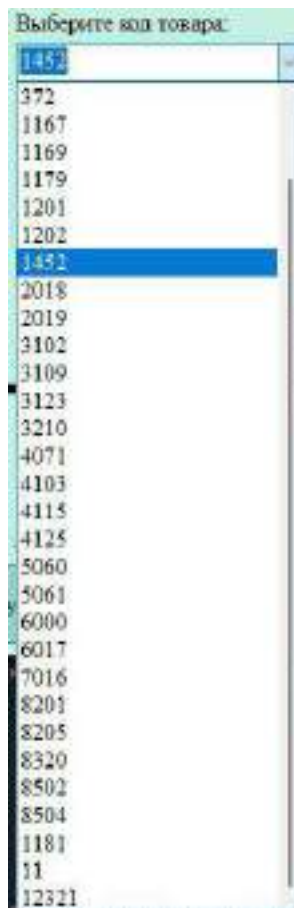


Рисунок 56 – Список кодов товара

Выберем код товара под номером 1452, после чего отобразится информация о нем (Рисунок 57):

Название:	Гель-скраб для умывания
Кол-во на складе:	52
Цена за шт.:	88

Рисунок 57 – Информация о товаре по коду товара

Введем количество и поле стоимость, которое высчитывается автоматически (Рисунок 58):

Выберите код товара:

1452

Название: Гель-скраб для умывания

Кол-во на складе: 52

Цена за шт.: 88

Выберите кол-во: 2

Стоимость: 176,00

Рисунок 58 – Автоматический расчет стоимости товара

Примечание: если написать в поле «Выберите кол-во» больше, чем поле «Кол-во на складе», то будет ошибка в диалоговом окне, которая укажет на это (Рисунок 59):

Выберите код товара:

1452

Название: Гель-скраб для умывания

Кол-во на складе: 52

Цена за шт.: 88

Выберите кол-во: 222

Стоимость: 1 936,00

Вы не можете указать количество товара больше, чем на складе!

OK

Рисунок 59 – Сообщение о неверном заполнении данных

Теперь нажмем на кнопки «Добавить в таблицу «Заказы» и «Обновить данные», чтобы данные отобразились в таблице «Заказы» (Рисунок 60):

Заказ:

	Код товара	Количество	Цена	Стоимость
▶	1452	2	88,0000	176,0000
*				

Выберите код товара: 1452

Наименование: Гель-спрей для укладки волос

Кол-во на складе: 52

Цена за шт.: 88

Выберите кол-во: 2

Стоимость: 176,00

Добавить в таблицу "Заказы" Обновить данные Удалить данные

Оформление счета:

-->Нажмите,чтобы указать дату<--

Укажите номер сотрудника: 1

☐ Дисконтная карта

Назначьте число скидки(если есть дисконтная карта), иначе оставьте поле пустым: 0

Добавить в таблицу "Счета" Открыть таблицу "Счета"

Рисунок 60 – Отображение данных в таблице «Заказы»

Примечание: для надежного тестирования был добавлен еще один товар в таблицу «Заказы» (Рисунок 61):

Заказ:

	Код товара	Количество	Цена	Стоимость
▶	1452	2	88,0000	176,0000
	1179	12	103,0000	1236,0000
*				

Рисунок 61 – Данные в таблице «Заказы»

Теперь оформим счет, для этого сначала укажем дату, для этого нужно нажать на поле «-->Нажмите,чтобы указать дату<--», затем укажем в поле «Укажите номер сотрудника» его номер (Рисунок 62):

Заказ:

	Код товара	Количество	Цена	Стоимость
▶	1452	2	88,0000	176,0000
	1179	12	103,0000	1236,0000
*				

Выберите код товара:

Наименование:

Кол-во на складе:

Цена за шт.:

Выберите кол-во:

Стоимость:

Добавить в таблицу "Заказы" Обновить данные Удалить данные

Оформление счета:

20.04.2021

Укажите номер сотрудника: 3

☐ Дисконтная карта

Назначьте число скидки(если есть дисконтная карта), иначе оставьте поле пустым: 0

Добавить в таблицу "Счета" Открыть таблицу "Счета"

Рисунок 62 – Номера сотрудников

Затем, если хотим в счет добавить скидку, нужно поставить галочку у текста «Дисконтная карта» и ввести в поле скидку. Любое число будет процентом, т.е, например число 24 это и есть 24%. Добавим скидку в 24% (Рисунок 63):

Оформление счета:
26.04.2023

Укажите номер сотрудника:

☒ Дисконтная карта

Назначение: число скидок (если есть дисконтная карта), иначе оставшее поле пустым:

Рисунок 63 – Добавление дисконтной карты и скидки

Нажмем теперь «Добавить в таблицу «Счета», а затем откроем ее. Как мы можем заметить, в 23 строке таблицы были добавлены новые данные (Рисунок 64):

Номер_чека	Код_сотрудника	Дата	Дисконтная_карт	Сумма_к_оплате	Скидка
17	8	апр 21 2023 12:...	1	500.8	45
18	8	апр 21 2023 12:...	1	500.8	45
19	8	апр 21 2023 12:...	1	1161.6	45
20	8	апр 21 2023 12:...	1	1024.85	45
21	8	апр 21 2023 12:...	1	1024.85	45
22	2	апр 21 2023 12:...	1	484	50
23	1	апр 26 2023 12:...	1	1073.12	24

Рисунок 64 – Таблица «Счета»

Проверим теперь правильность расчета с учетом скидки. Как мы видим, расчеты верны (Рисунок 65):

Введите:

Код_чека	Код_сотрудника	Дата	Дисконтная_карт	Сумма_к_оплате	Скидка
17	8	апр 21 2023 12:...	1	500.8	45
18	8	апр 21 2023 12:...	1	500.8	45
19	8	апр 21 2023 12:...	1	1161.6	45
20	8	апр 21 2023 12:...	1	1024.85	45
21	8	апр 21 2023 12:...	1	1024.85	45
22	2	апр 21 2023 12:...	1	484	50
23	1	апр 26 2023 12:...	1	1073.12	24
24	1	апр 26 2023 12:...	1	1073.12	0

Выберите вид отчета:

Назначение:

Баллы за оплату:

Дисконтная карта:

Вычисления:

$$(88 \times 2) + (103 \times 12) = 1073.12$$

$$(88 \times 2) + (103 \times 12) - 24\%$$

Рисунок 65 – Проверка расчетов

Примечание: были также добавлены такие же данные в 24 строку, но без учета скидки.

Тест №3

Рассмотрим в качестве теста все функции, которые есть в диалоговом окне «Информация о сотрудниках», которое открывается через главное диалоговое окно с помощью кнопки «Сведения о сотрудниках» (Рисунок 66).

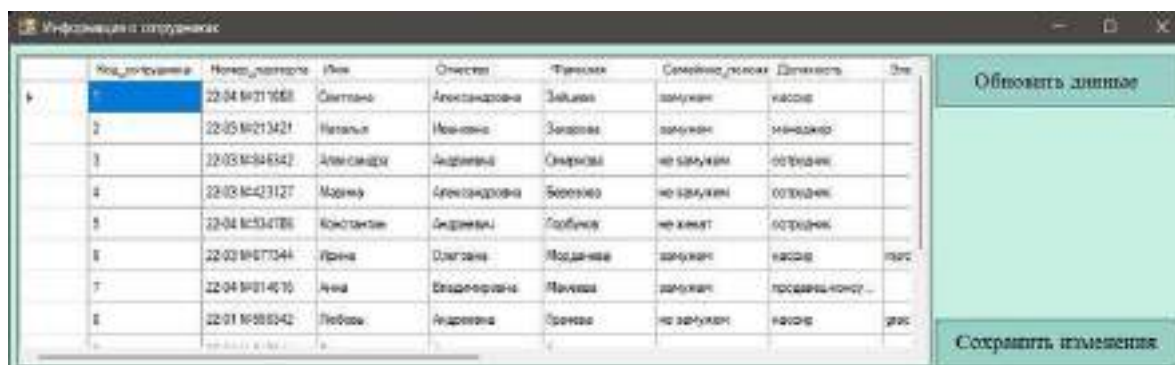


Рисунок 66 – Диалоговое окно «Информация о сотрудниках»

Добавим новые данные в таблицу строки 12 и нажимаем «Сохранить данные», а затем «Обновить данные». После чего, появится диалоговое окно, которое сообщает о том, что данные успешно сохранены (Рисунок 67):



Рисунок 67 – Успешное изменение данных в таблице «Сотрудники»

В результате тестирования было подтверждено, что приложение полностью соответствует требованиям курсовой работы и способно эффективно управлять базой данных торгового предприятия.

Заключение

В данной курсовой работе была спроектирована и разработана база данных «Торговое предприятие». Данная предметная область была исследована в соответствии с заданием. Анализируя ее, были выявлены такие объекты этого вида деятельности, как «Сотрудники», «Заказы», «Счета», «Поставщики», «Товары» со всеми их составляющими и характеристиками.

Также была разработана программа с использованием компонентов среды визуального проектирования MS Visual C# и MS SQL Management Studio 2019.

Курсовая работа выполнялась в пять этапов:

- 1) анализ теоретического материала и методических рекомендаций;
- 2) разработка и создание основных модели проектируемой базы данных;
- 3) формирование SQL-запросов к базе данных;
- 4) разработка программы средствами MS Visual C#, которая позволила решить поставленные задачи;
- 5) тестирование разработанного приложения, в ходе которого не было выявлено никаких ошибок, все ожидаемые результаты совпали с полученными, формирование таблиц, занесение, удаление и изменение в них данных осуществлялось корректно.

С помощью базы данных «Торговое предприятие» можно получить следующие, актуальные в данной работе сведения: о товарах и их категориях, поставщиках и заказах, продажах и складах. Также можно получить список оплаченных заказов и отчёты по продажам.

Разработанная система позволяет решать задачи поиска информации в базе данных по запросу пользователя и выдавать информацию в удобном для работы и наглядном для непрофессионала виде.

Современные базы данных являются основой многочисленных информационных систем. Информация, накопленная в них, является чрезвычайно ценным материалом, и в настоящий момент широко распространяются методы обработки баз данных с точки зрения извлечения из них дополнительных знаний, методов, которые связаны с обобщением и различными дополнительными способами обработки данных.

В проекте представлены все разработанные модели системы, а также все используемые в её интерфейсе формы. При проектировании запросов были использованы SQL-запросы.

Итогом курсовой работы можно считать закрепление знаний в области проектирования баз данных, навыков программирования на языке C# и создании приложения.

Список использованных источников

- 1 Макаренко О.В., Зарубина Т.Н. Программирование баз данных на С#. – СПб.: БХВ-Петербург, 2012. – 352 с.
- 2 Шилдт Г. С# 4.0. Полное руководство. – М.: Вильямс, 2011. – 1056 с.
- 3 Кузнецов В.В. Современные информационные технологии в управлении торговым предприятием. – М.: Издательский дом "Инфра-М", 2014. – 320 с.
- 4 Леонтьев А. В., Марченков Д. А. Разработка информационных систем с использованием SQL и С#. – М.: Издательство "Лань", 2015. – 256 с.
- 5 Бондарчук М.В., Чернов С.В., Шаталова О.В. Проектирование баз данных в Microsoft SQL Server с использованием Entity Framework. – СПб.: Питер, 2016. – 336 с.
- 6 Касаткина Е.В., Лапин Е.С. SQL и проектирование баз данных. – М.: Издательский дом "ДМК Пресс", 2018. – 304 с.
- 7 Каляев И.Г., Колесников И.В., Феоктистов А.В. Microsoft SQL Server. Курс для разработчиков. – М.: Издательство "Бином. Лаборатория знаний", 2018. – 400 с.
- 8 Робинсон Э., Уоррен Д. Программирование на С# 8.0 и .NET Core 3.0 для профессионалов. – М.: ДМК Пресс, 2019. – 1200 с.
- 9 Клименко С.А., Колесникова Е.В., Попова Ю.В. Создание баз данных в Microsoft SQL Server с помощью языка программирования T-SQL. – СПб.: БХВ-Петербург, 2020. – 448 с.
- 10 Безуглый А.В., Мосин В.В. Введение в SQL Server Integration Services. – М.: ДМК Пресс, 2021. – 416 с.
- 11 Алексеев В.В., Мамыкин О.А. Создание информационной системы для торгового предприятия // Информатика и ее применения. 2018. Т. 12. № 4. С. 22-28.
- 12 Лукин А.В. Проектирование баз данных на примере информационной системы торгового предприятия // Научные ведомости Белгородского государственного университета. Серия: Компьютерные технологии. 2019. Т. 29. № 10. С. 87-96.
- 13 Гаранин А.А., Маркин С.А. Применение объектно-ориентированного подхода при разработке информационной системы торгового предприятия // Научно-технический вестник информационных технологий, механики и оптики. 2018. Т. 18. № 5. С. 943-948.
- 14 Кожевников А.В., Шестаков А.В. Разработка информационной системы для управления торговым предприятием // Информационные технологии в образовании и науке. 2017. № 4 (31). С. 122-128.
- 15 Мельников М.Н., Жукова Е.И. Проектирование информационной системы торгового предприятия на основе технологии клиент-сервер // Информационные технологии и вычислительные системы. 2017. № 1. С. 103-108.

Приложение А (обязательное)

Листинг программного кода

App.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
<configSections>
</configSections>
<connectionStrings>
<add name="Kurs2.Properties.Settings.Trading_companyConnectionString"
connectionString="Data Source=DESKTOP-80FF3I4\SQLEXPRESS02;Initial
Catalog=Trading_company;Integrated Security=True"
providerName="System.Data.SqlClient" />
</connectionStrings>
<startup>
<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
</startup>
</configuration>
```

DataGridViewPrinter.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing.Printing;
using System.Windows.Forms;
using System.Drawing;
using Kurs2;
namespace Kurs2
{
public class DataGridViewPrinter
{
public DataGridView dataGridView;
public PrintDocument printDocument;
public DataGridView dgv;
public DataGridViewPrinter(DataGridView dgv)
{
this.dgv = dgv;
}

public void PrintDocument_PrintPage(object sender, PrintPageEventArgs
e)
{
Bitmap bm = new Bitmap(this.dgv.Width, this.dgv.Height);
```

```

        this.dgv.DrawToBitmap(bm, new Rectangle(0, 0, this.dgv.Width,
this.dgv.Height));
        e.Graphics.DrawImage(bm, 0, 0);
    }

    public void Print()
    {
        PrintDialog printDialog = new PrintDialog();
        printDialog.Document = printDocument;
        if (printDialog.ShowDialog() == DialogResult.OK)
        {
            printDocument.Print();
        }
    }
}
}
}

```

Form1.cs

```

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 Kurs2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

        }

        private void button6_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            // Создаем новую форму

```

```

Form2 form2 = new Form2();

// Получаем данные из таблицы "Товары" и передаем их на новую форму
SqlConnection connection = new SqlConnection("Data Source = DESKTOP-
80FF3I4\\SQLEXPRESS02; Initial Catalog = Trading_company; Integrated
Security = True");
SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM Товары",
connection);
DataTable table = new DataTable();
adapter.Fill(table);
form2.dataGridView1.DataSource = table;

// Отображаем новую форму
form2.Show();
}

private void button2_Click(object sender, EventArgs e)
{
Form7 form7 = new Form7(); // Создаем экземпляр новой формы
form7.Show(); // Отображаем новую форму
}

private void button3_Click(object sender, EventArgs e)
{
Form8 form8 = new Form8(); // Создаем экземпляр новой формы
form8.Show(); // Отображаем новую форму
}

private void button5_Click(object sender, EventArgs e)
{
Form11 form11 = new Form11(); // Создаем экземпляр новой формы
form11.Show(); // Отображаем новую форму
}

private void Form1_Load(object sender, EventArgs e)
{
}
}
}

```

Form2.cs

```

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 Kurs2
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            // Сохраняем изменения в таблице "Товары"
            SqlConnection connection = new SqlConnection("Data Source=DESKTOP-80FF3I4\\SQLEXPRESS02;Initial Catalog=Trading_company;Integrated Security=True");
            SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM Товары", connection);
            SqlCommandBuilder builder = new SqlCommandBuilder(adapter);
            adapter.Update((DataTable)dataGridView1.DataSource);
            // выводим сообщение об успешном сохранении
            MessageBox.Show("Данные успешно сохранены!");
        }

        private void button2_Click(object sender, EventArgs e)
        {
            // Создаем новый экземпляр формы Form2
            Form3 form3 = new Form3();

            // Выполняем запрос и получаем результаты в виде DataTable
            DataTable dataTable = new DataTable();
            using (SqlConnection connection = new SqlConnection("Data Source=DESKTOP-80FF3I4\\SQLEXPRESS02;Initial Catalog=Trading_company;Integrated Security=True"))
            {
                connection.Open();
                SqlCommand command = new SqlCommand("SELECT Товары.[№], Товары.[Код_товара], Товары.Категория, Товары.Наименование, Товары.[Единица_измерения], Товары.[Цена_закупочная], Товары.[Наценка,%], (Товары.[Цена_закупочная]*Товары.[Наценка,%]+Товары.[Цена_закупочная]) AS [Цена_реализации], Товары.[Количество_на_складе], Товары.[Срок_реализации], Товары.[Код_поставщика] FROM Товары", connection);
                SqlDataAdapter adapter = new SqlDataAdapter(command);
                adapter.Fill(dataTable);
            }

            // Устанавливаем DataTable в качестве источника данных для dataGridView1 на форме Form2

```

```

form3.dataGridView3.DataSource = dataTable;

// Отображаем форму Form2
form3.Show();
}

private void button3_Click(object sender, EventArgs e)
{
    // Создание подключения к базе данных
    SqlConnection connection = new SqlConnection("Data Source=DESKTOP-
80FF3I4\\SQLEXPRESS02;Initial Catalog=Trading_company;Integrated
Security=True");

    // Создание команды SQL для выполнения запроса
    SqlCommand command = new SqlCommand("SELECT * FROM Товары WHERE
[Срок_реализации] < @Today", connection);

    // Добавление параметра даты в запрос
    command.Parameters.AddWithValue("@Today", DateTime.Today);

    // Создание адаптера данных и заполнение таблицы
    SqlDataAdapter adapter = new SqlDataAdapter(command);
    DataTable table = new DataTable();
    adapter.Fill(table);

    // Отображение данных в DataGridView на форме Form3
    Form4 form4 = new Form4();
    form4.dataGridView3.DataSource = table;
    form4.Show();
}

private void button4_Click(object sender, EventArgs e)
{
    Form5 form5 = new Form5();
    form5.ShowDialog();
}

private void button5_Click(object sender, EventArgs e)
{
    Form6 form6 = new Form6();
    form6.Show();
}

private void button6_Click(object sender, EventArgs e)
{
    // Соединение с базой данных
    using (SqlConnection connection = new SqlConnection("Data
Source=DESKTOP-80FF3I4\\SQLEXPRESS02;Initial
Catalog=Trading_company;Integrated Security=True"))
    {
        // Запрос на обновление значения столбца "Наценка,%" в таблице "Товары"
    }
}

```

```

string query = "UPDATE Товары SET [Наценка, %] = @Markup";

// Создание команды для выполнения запроса
using (SqlCommand command = new SqlCommand(query, connection))
{
    try
    {
        // Открытие соединения
        connection.Open();

        // Установка значения параметра запроса
        command.Parameters.AddWithValue("@Markup", textBox1.Text);

        // Выполнение запроса
        int rowsAffected = command.ExecuteNonQuery();

        // Вывод сообщения об успешном выполнении запроса
        MessageBox.Show($"{rowsAffected} строк было изменено.");
    }
    catch (Exception ex)
    {
        // Вывод сообщения об ошибке выполнения запроса
        MessageBox.Show(ex.Message);
    }
}

private void button7_Click(object sender, EventArgs e)
{
    string connectionString = "Data Source=DESKTOP-80FF3I4\\SQLEXPRESS02;Initial Catalog=Trading_company;Integrated Security=True";
    //Обновляем таблицу "Товары"
    string query = "SELECT * FROM Товары";
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        SqlDataAdapter adapter = new SqlDataAdapter(query, connection);
        DataSet ds = new DataSet();
        adapter.Fill(ds, "Товары");
        dataGridView1.DataSource = ds.Tables["Товары"];
    }
}

private void Form2_Load(object sender, EventArgs e)
{
}
}
}

```


Form3.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Kurs2
{
    public partial class Form3 : Form
    {
        public Form3()
        {
            InitializeComponent();
        }

        private void Form3_Load(object sender, EventArgs e)
        {

        }
    }
}
```

Form4.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Kurs2
{
    public partial class Form4 : Form
    {
        public Form4()
        {
            InitializeComponent();
        }

        private void Form4_Load(object sender, EventArgs e)
```

$\{ \}$

Form5.cs

```
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 Kurs2
{
    public partial class Form5 : Form
    {
        public Form5()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string connectionString = "Data Source=DESKTOP-8OFF3I4\\SQLEXPRESS02;Initial Catalog=Trading_company;Integrated Security=True";
            string query = "SELECT [Код_товара], Наименование, [Цена_закупочная] FROM Товары WHERE [Код_товара] = @код_товара";
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand command = new SqlCommand(query, connection))
                {
                    command.Parameters.AddWithValue("@код_товара", textBox1.Text);
                    connection.Open();
                    SqlDataReader reader = command.ExecuteReader();
                    if (reader.HasRows)
                    {
                        DataTable dataTable = new DataTable();
                        dataTable.Load(reader);
                        dataGridView4.DataSource = dataTable;
                    }
                    else
                    {

```

```

MessageBox.Show("Товар не найден");
}
}
}

}

private void Form5_Load(object sender, EventArgs e)
{

}
}
}

```

Form6.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Data.SqlClient;
using System.Drawing;
using System.Drawing.Printing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Kurs2;
namespace Kurs2

{
    public partial class Form6 : Form
    {
        public Form6()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string query = "SELECT Товары.[Код_товара], Товары.Наименование,
(Товары.[Цена_закупочная]*Товары.[Наценка,%]+Товары.[Цена_закупочная])
AS[Отпускная_цена], Товары.[Срок_реализации] FROM Товары";
            using (SqlConnection connection = new SqlConnection("Data
Source=DESKTOP-80FF3I4\\SQLEXPRESS02;Initial
Catalog=Trading_company;Integrated Security=True"))
            {
                SqlCommand command = new SqlCommand(query, connection);
                SqlDataAdapter adapter = new SqlDataAdapter(command);
            }
        }
    }
}

```

```

        DataTable dataTable = new DataTable();
        adapter.Fill(dataTable);
        dataGridView6.DataSource = dataTable;
    }

    private void button2_Click(object sender, EventArgs e)
    {
        string connectionString = "Data Source=DESKTOP-80FF3I4\\SQLEXPRESS02;Initial Catalog=Trading_company;Integrated Security=True";
        string query = "SELECT Код_товара, Наименование, Цена_реализации, Срок_реализации FROM Товары";

        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            SqlCommand command = new SqlCommand(query, connection);
            SqlDataAdapter adapter = new SqlDataAdapter(command);
            DataTable dataTable = new DataTable();
            adapter.Fill(dataTable);
            dataGridView6.DataSource = dataTable;
        }

        PrintDocument printDoc = new PrintDocument();
        DataGridViewPrinter dgvPrinter = new
        DataGridViewPrinter(this.dataGridView6);
        printDoc.PrintPage += new
        PrintPageEventHandler(dgvPrinter.PrintDocument_PrintPage);
        printDoc.Print();
    }

    private void Form6_Load(object sender, EventArgs e)
    {
    }
}

```

Form7.cs

```

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 Kurs2
{
    public partial class Form7 : Form
    {
        SqlConnection conn = new SqlConnection(@"Data Source=DESKTOP-
80FF3I4\SQLEXPRESS02;Initial Catalog=Trading_company;Integrated
Security=True");
        public Form7()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            // Проверяем, что в TextBox введено число
            if (!int.TryParse(textBox1.Text, out int supplierId))
            {
                MessageBox.Show("Введите корректный код поставщика!");
                return;
            }

            // Проверяем, что в TextBox не пустое значение
            if (string.IsNullOrEmpty(textBox1.Text))
            {
                MessageBox.Show("Введите код поставщика!");
                return;
            }

            string query = $"SELECT Код_товара, Категория, Наименование,
Единица_измерения FROM Товары WHERE Код_поставщика = {supplierId}";
            SqlDataAdapter adapter = new SqlDataAdapter(query, conn);
            DataSet ds = new DataSet();
            adapter.Fill(ds, "Товары");
            dataGridView7.DataSource = ds.Tables["Товары"];
        }

        private void Form7_Load(object sender, EventArgs e)
        {
            string connectionString = "Data Source=DESKTOP-
80FF3I4\SQLEXPRESS02;Initial Catalog=Trading_company;Integrated
Security=True";
            string query = "SELECT * FROM Поставщики";
            using (SqlConnection conn = new SqlConnection(connectionString))
            {
                SqlDataAdapter adapter = new SqlDataAdapter(query, conn);
                DataSet ds = new DataSet();
                adapter.Fill(ds, "Поставщики");
                dataGridView9.DataSource = ds.Tables["Поставщики"];
            }
        }
    }
}

```

```

    }
    }

    private void button2_Click(object sender, EventArgs e)
    {
        // Сохраняем изменения в таблице "Товары"
        SqlConnection connection = new SqlConnection("Data Source=DESKTOP-80FF3I4\\SQLEXPRESS02;Initial Catalog=Trading_company;Integrated Security=True");
        SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM Поставщики", connection);
        SqlCommandBuilder builder = new SqlCommandBuilder(adapter);
        adapter.Update((DataTable)dataGridView9.DataSource);
        // выводим сообщение об успешном сохранении
        MessageBox.Show("Данные успешно сохранены!");
    }
}

```

Form8.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Kurs2
{
    public partial class Form8 : Form
    {
        private SqlConnection connection; // Объект подключения к базе данных

        public Form8()
        {
            InitializeComponent();
            connection = new SqlConnection("Data Source=DESKTOP-80FF3I4\\SQLEXPRESS02;Initial Catalog=Trading_company;Integrated Security=True"); // Здесь указываем строку подключения к вашей базе данных
        }

        private void Form8_Load(object sender, EventArgs e)
        {

```

// TODO: данная строка кода позволяет загрузить данные в таблицу "trading_companyDataSet.Сотрудники". При необходимости она может быть перемещена или удалена.

```
this.сотрудникиTableAdapter.Fill(this.trading_companyDataSet.Сотрудники);  
string connectionString = "Data Source=DESKTOP-80FF3I4\\SQLEXPRESS02;Initial Catalog=Trading_company;Integrated Security=True";
```

```
string query = "SELECT * FROM Заказы";  
using (SqlConnection connection = new SqlConnection(connectionString))  
{  
    SqlDataAdapter adapter = new SqlDataAdapter(query, connection);  
    DataSet ds = new DataSet();  
    adapter.Fill(ds, "Заказы");  
    dataGridView1.DataSource = ds.Tables["Заказы"];  
}
```

// Загружаем данные в ComboBox1 из таблицы "Товары"

```
try  
{  
    connection.Open();  
    SqlCommand cmd = new SqlCommand("SELECT Код_товара FROM Товары",  
connection);  
    SqlDataReader reader = cmd.ExecuteReader();  
    while (reader.Read())  
    {  
        comboBox1.Items.Add(reader["Код_товара"].ToString());  
    }  
    reader.Close();  
}  
catch (Exception ex)  
{  
    MessageBox.Show("Ошибка при загрузке данных из базы данных: " +  
ex.Message);  
}
```

finally

```
{  
    connection.Close();  
}
```

try

```
{  
    connection.Open();  
    SqlCommand cmd = new SqlCommand("SELECT Код_сотрудника FROM  
Сотрудники", connection);  
    SqlDataReader reader = cmd.ExecuteReader();  
    while (reader.Read())  
    {  
        comboBox2.Items.Add(reader["Код_сотрудника"].ToString());  
    }  
    reader.Close();  
}  
catch (Exception ex)
```



```

        {
            MessageBox.Show("Ошибка при загрузке данных из базы данных: " +
ex.Message);
        }
        finally
        {
            connection.Close();
        }

    }

    private void comboBox1_SelectedIndexChanged(object sender, EventArgs
e)
    {
        // При выборе кода товара в ComboBox1 отображаем его название в Label2
        try
        {
            connection.Open();
            SqlCommand cmd = new SqlCommand("SELECT Наименование FROM Товары WHERE
Код_товара = @Код_товара", connection);
            cmd.Parameters.AddWithValue("@Код_товара",
comboBox1.SelectedItem.ToString());
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())
            {
                label2.Text = reader["Наименование"].ToString();
            }
            else
            {
                label2.Text = «;
            }
            reader.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Ошибка при загрузке данных из базы данных: " +
ex.Message);
        }
        finally
        {
            connection.Close();
        }
        try
        {
            connection.Open();
            SqlCommand cmd = new SqlCommand("SELECT Наименование,
Количество_на_складе, Цена_закупочная FROM Товары WHERE Код_товара =
@Код_товара", connection);
            cmd.Parameters.AddWithValue("@Код_товара",
comboBox1.SelectedItem.ToString());

```

```

        SqlDataReader reader = cmd.ExecuteReader();
        if (reader.Read())
        {
            label2.Text = reader["Наименование"].ToString();
            label3.Text = reader["Количество_на_складе"].ToString();
            label4.Text = reader["Цена_закупочная"].ToString();
        }
        else
        {
            label2.Text = «;
            label3.Text = «;
            label4.Text = «;
        }
        reader.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка при загрузке данных из базы данных: " +
ex.Message);
    }
    finally
    {
        connection.Close();
    }
}

private void label8_Click(object sender, EventArgs e)
{
    label8.Text = DateTime.Now.ToString("dd.MM.yyyy");
}

private void comboBox2_SelectedIndexChanged(object sender, EventArgs
e)
{
}

private void button3_Click(object sender, EventArgs e)
{
    Form10 form10 = new Form10(); // Создаем экземпляр новой формы
    form10.Show(); // Отображаем новую форму
}

private void textBox2_TextChanged(object sender, EventArgs e)
{
    float availableQuantity = Convert.ToSingle(label3.Text); // Получаем
доступное количество товара из label3
    float enteredQuantity = 0;
    if (!string.IsNullOrEmpty(textBox2.Text))

```

```

        {
            enteredQuantity = Convert.ToSingle(textBox2.Text); // Получаем
введенное количество товара из textBox2
        }
        if (enteredQuantity > availableQuantity)
        {
            MessageBox.Show("Вы не можете указать количество товара больше, чем на
складе!");
            textBox2.Text = availableQuantity.ToString(); // Устанавливаем в
textBox2 максимально доступное количество товара
        }

        float currentQuantity = 0;
        if (float.TryParse(label4.Text.Replace(",", "."), out
currentQuantity)) // Преобразуем значение в label4 в число, заменяя запятую
на точку
        {
            float multipliedQuantity = currentQuantity * enteredQuantity; //
Выполняем умножение

            textBox4.Text = multipliedQuantity.ToString("#,##0.00"); // Записываем
результат в textBox4 с форматом "--,--"
        }
        else
        {
            // Выводим сообщение об ошибке
            MessageBox.Show("Значение в label4 не может быть преобразовано в
число.");
        }
    }

    private void label15_Click(object sender, EventArgs e)
    {

    }

    private void textBox4_TextChanged(object sender, EventArgs e)
    {
        float currentQuantity = 0;
        if (float.TryParse(label4.Text, out currentQuantity)) // Преобразуем
значение в label4 в число
        {
            float enteredQuantity = 0;
            if (float.TryParse(textBox2.Text, out enteredQuantity)) // Преобразуем
значение в textBox2 в число
            {
                // Пропущено умножение
            }
            else
            {
                // Выводим сообщение об ошибке
            }
        }
    }

```

```

        MessageBox.Show("Вы ввели некорректное значение в textBox2. Пожалуйста,
введите число.");
    }
}
else
{
    // Выводим сообщение об ошибке
    MessageBox.Show("Значение в label4 не может быть преобразовано в
число.");
}
}

private void label4_Click(object sender, EventArgs e)
{

}

private void button1_Click(object sender, EventArgs e)
{
    // Проверяем, что выбран элемент в comboBox1
    if (comboBox1.SelectedItem != null)
    {
        // Получаем выбранный элемент в comboBox1
        string selectedProduct = comboBox1.SelectedItem.ToString();

        // Проверяем, что введено значение в textBox2
        if (!string.IsNullOrEmpty(textBox2.Text))
        {
            // Парсим значение из textBox2 в число
            if (float.TryParse(textBox2.Text, out float enteredQuantity))
            {
                // Проверяем, что значение в label4 может быть преобразовано в число
                if (float.TryParse(label4.Text, out float currentPrice))
                {
                    // Вычисляем стоимость
                    float totalCost = enteredQuantity * currentPrice;

                    // Проверяем, что введено значение в textBox4
                    if (!string.IsNullOrEmpty(textBox4.Text))
                    {
                        // Парсим значение из textBox4 в число
                        if (float.TryParse(textBox4.Text, out float enteredTotalCost))
                        {
                            // Добавляем данные в таблицу "Заказы"
                            string sql = "INSERT INTO Заказы (Код_товара, Количество, Цена,
Стоимость) VALUES (@Код_товара, @Количество, @Цена, @Стоимость)";
                            using (SqlConnection connection = new SqlConnection("Data
Source=DESKTOP-8OFF3I4\\SQLEXPRESS02;Initial
Catalog=Trading_company;Integrated Security=True"))
                            {
                                using (SqlCommand command = new SqlCommand(sql, connection))

```

```

{
// Добавляем параметры в команду
command.Parameters.AddWithValue("@Код_товара", selectedProduct);
command.Parameters.AddWithValue("@Количество", enteredQuantity);
command.Parameters.AddWithValue("@Цена", currentPrice);
command.Parameters.AddWithValue("@Стоимость", enteredTotalCost);

// Открываем соединение
connection.Open();

// Выполняем команду
command.ExecuteNonQuery();

// Закрываем соединение
connection.Close();
}
}

// Выводим сообщение об успешном добавлении данных
MessageBox.Show("Данные успешно добавлены в таблицу 'Заказы'.");
}
else
{
// Выводим сообщение об ошибке
MessageBox.Show("Вы ввели некорректное значение в textBox4. Пожалуйста,
введите число.");
}
}
else
{
// Выводим сообщение об ошибке
MessageBox.Show("Значение в textBox4 не может быть пустым.");
}
}
else
{
// Выводим сообщение об ошибке
MessageBox.Show("Значение в label4 не может быть преобразовано в
число.");
}
}
else
{
// Выводим сообщение об ошибке
MessageBox.Show("Вы ввели некорректное значение в textBox2. Пожалуйста,
введите число.");
}
}
else
{
// Выводим сообщение об ошибке

```

```

        MessageBox.Show("Введите количество !");
    }
}
else
{
    // Выводим сообщение об ошибке
    MessageBox.Show("Необходимо выбрать товар из списка.");
}
}

private void button4_Click(object sender, EventArgs e)
{
    using (SqlConnection connection = new SqlConnection("Data
Source=DESKTOP-8OFF3I4\\SQLEXPRESS02;Initial
Catalog=Trading_company;Integrated Security=True"))
    {
        connection.Open();
        using (SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM
Заказы", connection))
        {
            // Создаем новый DataTable
            DataTable dt = new DataTable();
            // Заполняем DataTable данными из таблицы "Заказы"
            adapter.Fill(dt);
            // Привязываем DataTable к DataGridView
            dataGridView1.DataSource = dt;
        }
    }
}

private void button5_Click(object sender, EventArgs e)
{
    // Проверяем, есть ли выделенные строки в DataGridView
    if (dataGridView1.SelectedRows.Count > 0)
    {
        // Если есть выделенные строки, то удаляем их из таблицы "Заказы"
        foreach (DataGridViewRow row in dataGridView1.SelectedRows)
        {
            // Получаем значение в ячейке "Код_товара" текущей строки
            int orderId = Convert.ToInt32(row.Cells["Код_товара"].Value);
            // Удаляем строку из таблицы "Заказы" по "Код_товара"
            DeleteOrder(orderId);
        }
    }
    else
    {
        // Если нет выделенных строк, то выводим сообщение
        MessageBox.Show("Выделите строку(и) для удаления.");
    }
}

// Обновляем таблицу "Заказы" после удаления

```

```

UpdateOrderTable();
}

// Метод для удаления заказа по "Код_заказа"
private void DeleteOrder(int orderId)
{
    using (SqlConnection connection = new SqlConnection("Data
Source=DESKTOP-8OFF3I4\\SQLEXPRESS02;Initial
Catalog=Trading_company;Integrated Security=True"))
    {
        connection.Open();
        using (SqlCommand cmd = new SqlCommand("DELETE FROM Заказы WHERE
Код_товара = @orderId", connection))
        {
            cmd.Parameters.AddWithValue("@orderId", orderId);
            cmd.ExecuteNonQuery();
        }
    }
}

// Метод для удаления всех заказов
private void DeleteAllOrders()
{
    using (SqlConnection connection = new SqlConnection("Data
Source=DESKTOP-8OFF3I4\\SQLEXPRESS02;Initial
Catalog=Trading_company;Integrated Security=True"))
    {
        connection.Open();
        using (SqlCommand cmd = new SqlCommand("DELETE FROM Заказы",
connection))
        {
            cmd.ExecuteNonQuery();
        }
    }
}

// Метод для обновления таблицы "Заказы"
private void UpdateOrderTable()
{
    using (SqlConnection connection = new SqlConnection("Data
Source=DESKTOP-8OFF3I4\\SQLEXPRESS02;Initial
Catalog=Trading_company;Integrated Security=True"))
    {
        connection.Open();
        using (SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM
Заказы", connection))
        {
            DataTable dt = new DataTable();
            adapter.Fill(dt);
            dataGridView1.DataSource = dt;
        }
    }
}

```



```

    }
    }

    private void textBox1_TextChanged(object sender, EventArgs e)
    {

    }

    private void button2_Click(object sender, EventArgs e)
    {
        // Проверяем заполнение label10_Click
        if (string.IsNullOrEmpty(label10.Text))
        {
            MessageBox.Show("Пожалуйста, укажите дату!");
            return;
        }

        // Получаем значения из элементов управления
        string номерЧека = textBox1.Text;
        int кодСотрудника;
        if (comboBox3.SelectedValue != null &&
int.TryParse(comboBox3.SelectedValue.ToString(), out кодСотрудника))
        {
            DateTime дата = DateTime.Parse(label8.Text);
            bool дисконтнаяКарта = checkBox1.Checked;
            decimal скидка = 0m;
            if (дисконтнаяКарта)
            {
                скидка = Convert.ToDecimal(textBox3.Text);
            }

            // Считываем значение стоимости из таблицы "Заказы"
            decimal стоимость = 0m;
            foreach (DataGridViewRow row in dataGridView1.Rows)
            {
                if (!row.IsNewRow)
                {
                    стоимость += Convert.ToDecimal(row.Cells["Стоимость"].Value);
                }
            }

            // Вычисляем сумму к оплате с учетом скидки
            decimal суммаКОплате = 0m;
            if (дисконтнаяКарта)
            {
                суммаКОплате = стоимость - (стоимость * скидка / 100);
            }
            else
            {
                суммаКОплате = стоимость;
            }
        }
    }

```

```

        // Выполняем вставку данных в таблицу "Счета"
        using (SqlConnection connection = new SqlConnection("Data
Source=DESKTOP-8OFF3I4\\SQLEXPRESS02;Initial
Catalog=Trading_company;Integrated Security=True"))
        {
            connection.Open();
            using (SqlCommand command = new SqlCommand("INSERT INTO Счета
(Код_сотрудника, Дата, Дисконтная_карта, Скидка, Сумма_к_оплате) VALUES
(@КодСотрудника, @Дата, @ДисконтнаяКарта, @Скидка, @СуммаКОплате)",
connection))
            {
                command.Parameters.AddWithValue("@КодСотрудника", кодСотрудника);
                command.Parameters.AddWithValue("@Дата", дата);
                command.Parameters.AddWithValue("@ДисконтнаяКарта", дисконтнаяКарта);
                command.Parameters.AddWithValue("@Скидка", скидка);
                command.Parameters.AddWithValue("@СуммаКОплате", суммаКОплате);
                command.ExecuteNonQuery();
            }
        }

        MessageBox.Show("Данные успешно сохранены в таблицу 'Счета'.");
    }
    else
    {
        MessageBox.Show("Пожалуйста, выберите сотрудника и убедитесь, что
значение является числом");
    }
}

e) private void comboBox3_SelectedIndexChanged(object sender, EventArgs
{
}

private void label10_Click(object sender, EventArgs e)
{
}

private void label2_Click(object sender, EventArgs e)
{
}

private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
}

```

```
private void textBox3_TextChanged(object sender, EventArgs e)
{

}
}
```

Form9.cs

```
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 Kurs2
{
    public partial class Form9 : Form
    {
        public Form9()
        {
            InitializeComponent();

            private void Form9_Load(object sender, EventArgs e)
            {
                SqlConnection connection = new SqlConnection("Data Source=DESKTOP-80FF3I4\\SQLEXPRESS02;Initial Catalog=Trading_company;Integrated Security=True");
                SqlDataAdapter adapter = new SqlDataAdapter("SELECT Номер_чека, Код_сотрудника, Дата, Дисконтная_карта, Скидка, Сумма_к_оплате FROM Счета", connection);
                DataTable table = new DataTable();
                adapter.Fill(table);
                dataGridView11111.DataSource = table;
                dataGridView11111.AutoGenerateColumns = true;
                dataGridView11111.ReadOnly = true;
                dataGridView11111.Dock = DockStyle.Fill;
            }

            private void button1_Click(object sender, EventArgs e)
            {
                // Запрос для удаления всех строк из таблицы "Счета"
                string query = "DELETE FROM Счета";
            }
        }
    }
}
```

```

        using (SqlConnection connection = new SqlConnection("Data
Source=DESKTOP-80FF3I4\\SQLEXPRESS02;Initial
Catalog=Trading_company;Integrated Security=True"))
        {
            SqlCommand command = new SqlCommand(query, connection);

            connection.Open();

            int result = command.ExecuteNonQuery();

            if (result > 0)
            {
                MessageBox.Show("Таблица успешно очищена.");
            }
            else
            {
                MessageBox.Show("Ошибка при очистке таблицы.");
            }
        }
    }
}

```

Form10.cs

```

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 Kurs2
{
    public partial class Form10 : Form
    {
        public Form10()
        {
            InitializeComponent();
        }

        private void Form10_Load(object sender, EventArgs e)

```

```

{
    try
    {
        using (SqlConnection connection = new SqlConnection("Data
Source=DESKTOP-8OFF3I4\\SQLEXPRESS02;Initial
Catalog=Trading_company;Integrated Security=True"))
        {
            connection.Open();
            using (SqlCommand command = new SqlCommand("SELECT * FROM Счета",
connection))
            {
                SqlDataAdapter adapter = new SqlDataAdapter(command);
                DataTable dt = new DataTable();
                adapter.Fill(dt);
                dataGridView1.DataSource = dt;
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка при загрузке данных из базы данных: " +
ex.Message);
    }
}

private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
}
}
}

```

Form11.cs

```

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 Kurs2
{
    public partial class Form11 : Form
    {

```

```

public Form11()
{
    InitializeComponent();
}

private void Form11_Load(object sender, EventArgs e)
{
    try
    {
        using (SqlConnection connection = new SqlConnection("Data
Source=DESKTOP-8OFF3I4\\SQLEXPRESS02;Initial
Catalog=Trading_company;Integrated Security=True"))
        {
            connection.Open();
            using (SqlCommand command = new SqlCommand("SELECT * FROM Сотрудники",
connection))
            {
                SqlDataAdapter adapter = new SqlDataAdapter(command);
                DataTable dt = new DataTable();
                adapter.Fill(dt);
                dataGridView1.DataSource = dt;
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка при загрузке данных из базы данных: " +
ex.Message);
    }
}

private void button1_Click(object sender, EventArgs e)
{
    {
        // Сохраняем изменения в таблице "Сотрудники"
        SqlConnection connection = new SqlConnection("Data Source=DESKTOP-
8OFF3I4\\SQLEXPRESS02;Initial Catalog=Trading_company;Integrated
Security=True");
        SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM Сотрудники",
connection);
        SqlCommandBuilder builder = new SqlCommandBuilder(adapter);
        adapter.Update((DataTable)dataGridView1.DataSource);
        // выводим сообщение об успешном сохранении
        MessageBox.Show("Данные успешно сохранены!");
    }
}

private void button2_Click(object sender, EventArgs e)
{

```

```

        string connectionString = "Data Source=DESKTOP-80FF3I4\\SQLEXPRESS02;Initial Catalog=Trading_company;Integrated Security=True";
        string query = "SELECT * FROM Сотрудники";
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            SqlDataAdapter adapter = new SqlDataAdapter(query, connection);
            DataSet ds = new DataSet();
            adapter.Fill(ds, "Сотрудники");
            dataGridView1.DataSource = ds.Tables["Сотрудники"];
        }
    }

    private void button3_Click(object sender, EventArgs e)
    {

    }
}

```

Program.cs

```

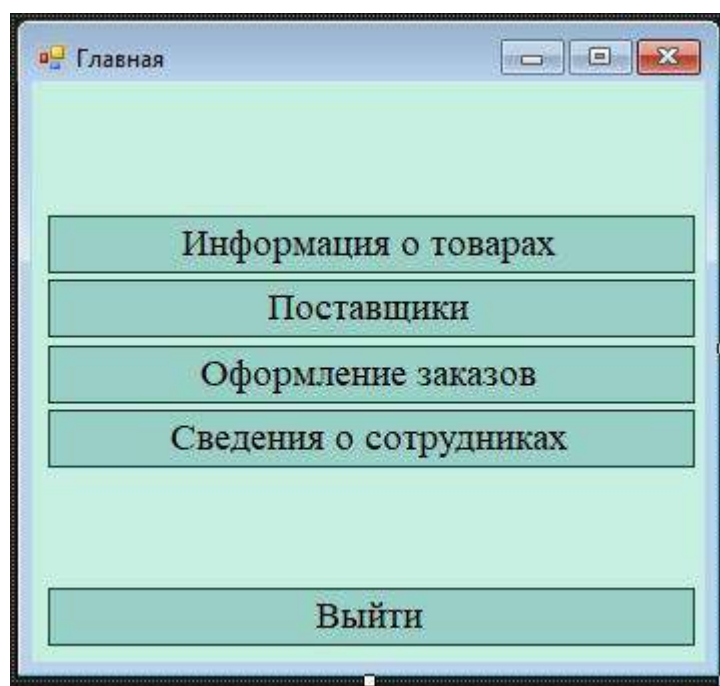
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

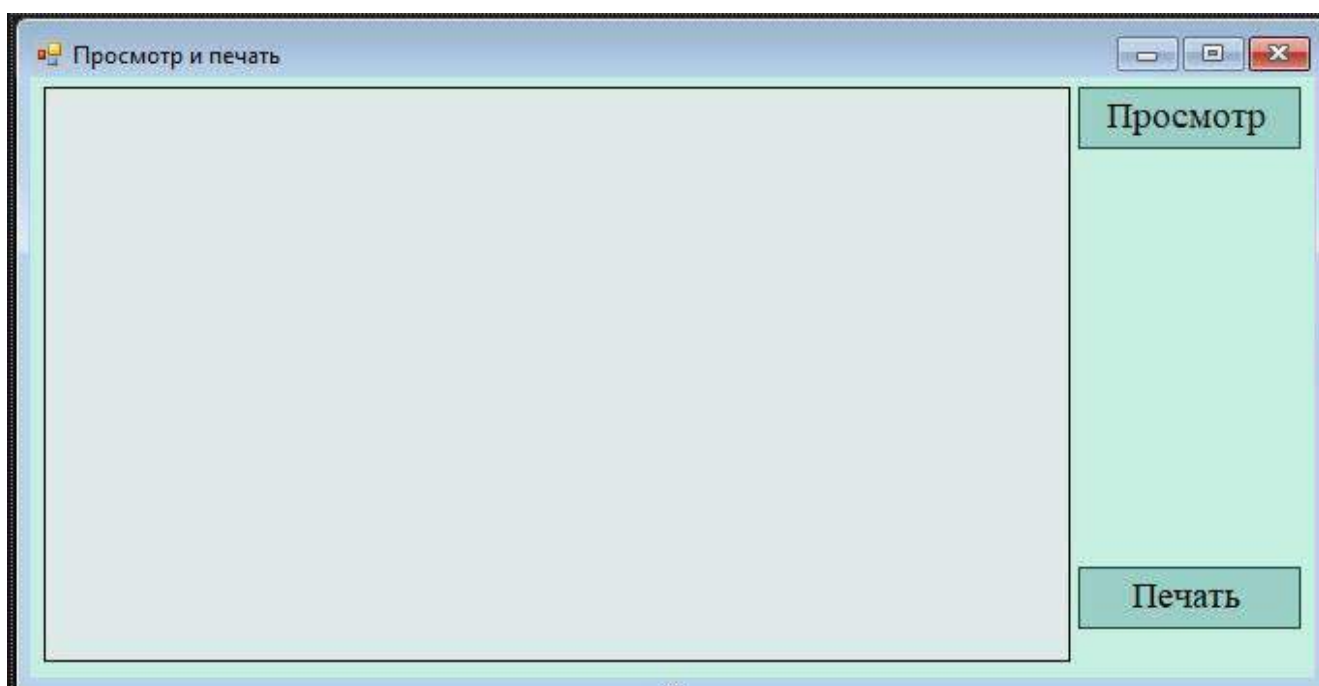
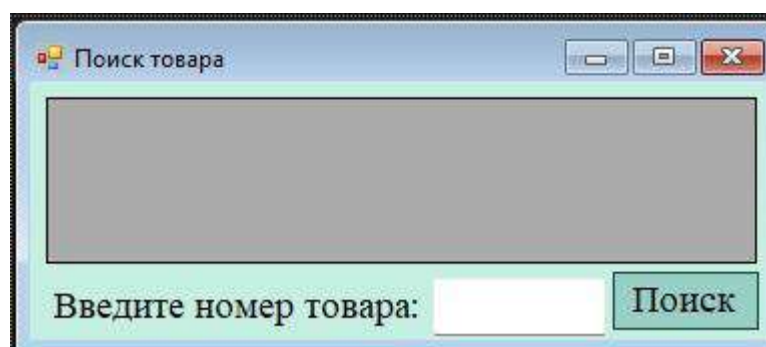
namespace Kurs2
{
    internal static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}

```


Приложение Б (справочное)

Экранные формы диалогового приложения





Информация о поставщиках

Информация о поставщиках

Сохранить

Введите код поставщика:

Поиск

Оформление заказа

Заказ:

Выберите код товара:

Название:

Кол-во на складе:

Цена за шт.:

Выберите кол-во:

Стоимость:

Добавить в таблицу "Заказы"

Обновить данные

Удалить данные

Оформление счета:

-->Нажмите, чтобы указать дату <--

Укажите номер сотрудника:

☐ Дисконтная карта

Нажмите число скидки(если есть дисконтная карта), иначе оставьте поле нулевым:

Добавить в таблицу "Счета"

Открыть таблицу "Счета"

