МИНИСТЕРСТВО НАУКИ И ВЫСЩЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

# Факультет Интеллектуальных систем и программирования Кафедра «Программная инженерия» им Л.П. Фельдмана

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

# к курсовому проекту по курсу «Базы данных»

Тема работы:

# «Ветеринарные клиники»

Руководители: С.В. Щедрин

# О.В. Рычка

Л.В. Незамова (подпись) (дата)

# Разработал:



Ст. гр. ПИ-21В В. Р. Рустамов

# (подпись) (дата)

Донецк - 2023

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДОНЕЦКОЙ НАРОДНОЙ РЕСПУБЛИКИ

ГОУ ВПО «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

# Факультет Интеллектуальных систем и программирования Кафедра «Программная инженерия» им Л.П. Фельдмана

УТВЕРЖДАЮ:

# Зав. кафедрой ПИ им. Л.П. Фельдмана

проф. Зори С.А.

# 14.02.2023 г.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

# На курсовой проект по дисциплине

«Базы данных»

# Тема: «Создание клиент-серверной информационной системы средствами СУБД» Выдано студенту группы ПИ-21В Рустамову В.Р.

Руководители курсового проекта: доц. Рычка О.В., ст. пр. Щедрин С.В., асс. Незамова Л.В.

Описание предметной области:

Разработать БД автоматизации работы ветеринарной службы города. Каждый пункт должен иметь лицензию на оказание услуг.

Один владелец может иметь несколько животных и обращаться за помощью несколько раз в день. По каждой оказанной процедуре оформлять отдельную запись. Стоимость лечения животного, определяется как стоимость оказанной процедуры + стоимость материалов (по предоставленной процедуре). Стоимость одной и той же процедуры, в разных пунктах разная.

# Необходимо:

1. Спроектировать концептуальную модель базы данных (БД) для заданной предметной области и представить ее в виде взаимосвязанных таблиц, находящихся в третьей нормальной форме (в случае донормализации БД – обосновать необходимость). Выделить базовые таблицы и таблицы-справочники, указать для них первичные и внешние ключи.

# Создать базу данных в среде СУБД средствами языка SQL. Добавить таблицы, домены, индексы.

1. Разработать не менее шести триггеров (по одному для каждого типа события), как минимум для двух различных таблиц БД. Триггеры BEFORE INSERT должны быть созданы для всех таблиц и с использованием генераторов задавать значение первичного ключа для вновь добавляемой записи.

# Заполнить таблицы БД с использованием соответствующих запросов на языке SQL (не менее десяти записей в каждом справочнике, не менее 10000- 50000 псевдослучайных записей в таблицах)

1. Сформулировать и реализовать следующие виды запросов:

# симметричное внутреннее соединение с условием (два запроса с условием отбора по внешнему ключу, два – по датам);

* симметричное внутреннее соединение без условия (три запроса);

# левое внешнее соединение;

* правое внешнее соединение;

# запрос на запросе по принципу левого соединения;

* итоговый запрос без условия;

# итоговый запрос без условия с итоговыми данными вида «всего», «в том числе»;

* итоговые запросы с условием на данные (по значению, по маске, с использованием индекса, без использования индекса);

# итоговый запрос с условием на группы;

* итоговый запрос с условием на данные и группы;

# запрос на запросе по принципу итогового запроса;

* запрос с использование объединения;

# запросы с подзапросами (с использованием in, not in, case, операциями над итоговыми данными);

# определить лучший ветеринарный пункт по каждому району и тройку лучших пунктов по городу в целом;

# определить сотрудника предоставившего максимальное количество услуг по каждому пункту, по району и по городу в целом;

# определить доходы каждого пункта в районе, доходы от пунктов по каждому району и по городу в целом за указанный год (в одном районе может находиться несколько пунктов).

# Запросы без параметров реализовать в виде представлений, остальные запросы – в виде хранимых процедур и/или функций. Создать, по меньшей мере, одно модифицированное представление, используя механизм триггеров. ВСЯ логика проектируемого ПО – на сервере

1. Разработать клиентское приложение, которое предоставляет следующие возможности для работы с созданной базой данных:

# добавление, редактирование и удаление записей таблиц и модифицированных представлений;

* работа с наборами данных, находящихся в отношении «один ко многим» (создать составную форму для просмотра и редактирования данных родительской и дочерней таблиц);

# поиск и фильтрация данных отображаемых таблиц;

* просмотр результатов выполнения запросов;

# визуализация одного из итоговых запросов (диаграммы, экспорт в Excel);

1. Обеспечить защиту данных, информации от несанкционированного доступа

# График выполнения курсового проекта

|  |  |
| --- | --- |
| Неделя | Этап выполнения работы |
| 1-2 | Выдача и изучение задания |
| 3 | Анализ требований к системе и способов их реализации |
| 4-5 | Проектирование и реализация БД (таблицы, домены, индексы) |
| 6-7 | Создание триггеров и заполнение таблиц БД |
| 8-9 | Создание представлений и хранимых процедур |
| 10-13 | Разработка клиентского приложения |
| 14 | Тестирование и отладка системы |
| 15 | Оформление пояснительной записки |
| 16-17 | Защита курсового проекта |

Дата выдачи задания 13.02.2023

# Дата защиты курсового проекта 16.05.2023



**Задание принял:** Рустамов Владислав Русланович

# Подпись студента

Согласовано:

# Руководители проекта: Щедрин Сергей Валерьевич

Рычка Ольга Валентиновна Незамова Лариса Викторовна

# РЕФЕРАТ

Отчет по курсовому проекту содержит страниц 94, рисунков 96, таблиц 1, приложения 2.

# Цель курсового проекта – получение практических навыков разработки и проектирования баз данных, а также написание клиентского приложения для взаимодействия с базой данных. Достижение поставленной цели реализуется посредством решения следующих задач:

* + изучение среды управления базами данных MS SQL Server и графической оболочки Windows Forms;
  + освоение языка структурированных запросов SQL;
  + углубление знаний и закрепление практических навыков разработки программного обеспечения на языке C#;

# изучение библиотеки Entity Framework 6 для связи клиентского приложения, написанного на языке C#, с базой данных.

Объектом исследования данной работы является учет деятельности ветеринарных клиник города.

# Результатом работы является клиентское приложения для учета деятельности ветеринарных клиник города и реляционная база данных, предназначенная для хранения учета деятельности и информации ветеринарных клиник города.

БАЗЫ ДАННЫХ, MS SQL SERVER, C#, WINDOWS FORMS, ЗАПРОСЫ, ТАБЛИЦЫ

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ](#_bookmark0) [11](#_bookmark0)

1. [ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ, ПОСТАНОВКА ЗАДАЧИ](#_bookmark1) [12](#_bookmark1)
   1. [Описание входных документов](#_bookmark2) [13](#_bookmark2)
   2. [Выходные данные](#_bookmark3) [13](#_bookmark3)
   3. [Список ограничений](#_bookmark4) [13](#_bookmark4)
2. [ОБОСНОВАНИЕ ВЫБОРА СУБД И ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ДЛЯ](#_bookmark5) [НАПИСАНИЯ КЛИЕНТСКОГО ПРИЛОЖЕНИЯ……………………………….15](#_bookmark5)
   1. [Обоснование выбора СУБД MSSQL](#_bookmark6) SERVER [15](#_bookmark6)
   2. [Обоснование выбора инструментальных средств](#_bookmark7) [15](#_bookmark7)
3. [ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ В ВЫБРАННОЙ СУБД](#_bookmark8) [16](#_bookmark8)
   1. [Проектирование концептуальной модели БД](#_bookmark9) [16](#_bookmark9)
      1. [Инфологическое проектирование](#_bookmark10) [16](#_bookmark10)
      2. [Даталогическое проектирование](#_bookmark11) [17](#_bookmark11)
      3. [Физическое проектирование](#_bookmark12) [18](#_bookmark12)
   2. [Создание таблиц, доменов, индексов](#_bookmark13) [20](#_bookmark13)
      1. [Создание таблиц](#_bookmark14) [21](#_bookmark14)
      2. [Создание доменов](#_bookmark15) [22](#_bookmark15)
      3. [Создание индексов](#_bookmark16) [22](#_bookmark16)
   3. [Разработка триггеров](#_bookmark17) [23](#_bookmark17)
   4. [Проектирование запросов к базе данных](#_bookmark18) [26](#_bookmark18)
      1. [Простые запросы](#_bookmark19) [26](#_bookmark19)
      2. [Итоговые запросы](#_bookmark20) [39](#_bookmark20)
      3. [Запросы из курсового проекта](#_bookmark21) [52](#_bookmark21)
   5. [Создание представлений и модифицированных представлений](#_bookmark22) [55](#_bookmark22)
   6. [Создание функций](#_bookmark23) [57](#_bookmark23)
   7. [Формы и компоненты для работы с таблицами и](#_bookmark24) [справочниками](#_bookmark24)…...[59](#_bookmark24)
   8. [Формы и компоненты для отображения результатов запросов](#_bookmark25) [63](#_bookmark25)
4. [ТЕСТИРОВАНИЕ РАЗРАБОТАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ …66](#_bookmark26)

[ЗАКЛЮЧЕНИЕ](#_bookmark27) [69](#_bookmark27)

[СПИСОК ЛИТЕРАТУРЫ](#_bookmark28) [70](#_bookmark28)

[ПРИЛОЖЕНИЕ А РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ](#_bookmark29) [71](#_bookmark29)

[ПРИЛОЖЕНИЕ Б ЛИСТИНГ СЕРВЕРНОГО ПРИЛОЖЕНИЯ](#_bookmark30) [78](#_bookmark30)

# ВВЕДЕНИЕ

# База данных – упорядоченный набор структурированной информации, которые хранятся в электронном виде в компьютерной системе.

База данных обычно управляется системой управления базами данных – СУБД

# В большинстве баз данных для взаимодействия с данными используется язык структурированных запросов – SQL

SQL – это язык программирования, используемый в большинстве реляционных баз данных для запросов, обработки и определения данных.

# Основная задача курсового проекта - разработать реляционную базу данных

«Ветеринарные клиники» и клиентское приложение для работы с ней, а также получить практические навыки разработки и проектирования баз данных.

# Клиентское приложение позволяет пользователю добавлять, редактировать и удалять данные таблицы, производить сортировку, поиск и фильтрацию данных, выполнять запросы и сохранять в excel результат работы некоторых запросов.

Разработанное приложение может быть использовано для учета ветеринарных клиник города. Его практическая польза состоит в структурированности данных, легкости использования и оптимизации работы ветеринарных клиник.

# ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ, ПОСТАНОВКА ЗАДАЧИ

В ходе курсового проекта необходимо разработать базу данных учета деятельности ветеринарных клиник города.

# Один владелец может иметь несколько животных и обращаться за помощью несколько раз в день. По каждой оказанной процедуре оформлять отдельную запись. Стоимость лечения животного, определяется как стоимость оказанной процедуры + стоимость материалов (по предоставленной процедуре). Стоимость одной и той же процедуры, в разных пунктах разная.

# Ведение дел с помощью компьютерного приложение имеет множество преимуществ по сравнению с ведением дел с помощью бумажной отчетности. Следовательно, разработка приложения для учета ветеринарных клиник города весьма актуальна.

Назначения программы – упрощение учета деятельности ветеринарных клиник города. Учет деятельности ветеринарных станет более качественным и будет отнимать меньшее количество времени.

# Функциональность разрабатываемой системы:

* просмотр, редактирование, удаление записей;

# поиск, сортировка, фильтрация записей;

* просмотр составных форм;

# просмотр результатов выполнения запросов;

* генерация диаграмм для некоторых запросов;

# сохранение отчетов в excel для некоторых запросов.

База данных содержит информацию о ветеринарных клиниках, сотрудниках, животных, процедурах, а также учет ухода за животными.

# Описание входных документов

Входными документами являются информация о ветеринарных клиниках, сотрудниках, животных, процедурах, владельцах

# Информация о вет. клиниках включает в себя название пункта, номер регистрационного пункта, год открытия, телефон, тип собственности, адрес, район города.

Информация о сотрудниках включает ФИО, должность, оклад, дата рождения, название ветеринарной клиники.

# Информация о животных включает ФИО владельца, кличка, вид животного, возраст, условия содержания.

Информация о процедурах включает вид процедуры, дату оказания помощи животному, стоимость процедуры, скидка на процедуру, скидка на процедуру, ФИО сотрудника.

# Информация о владельцах включает ФИО, дату рождения, телефон.

# Выходные данные

# Выходными данными, ожидаемыми от учета деятельности ветеринарных клиник города, являются результаты работы запросов, диаграммы, отчеты, сохранённые в excel. Выходные данные выводятся в удобном для чтения формате.

# Список ограничений

База данных должна предусмотреть следующие ограничения:

* регистрационный номер пункта должен быть больше 0 и меньше 10 000;
* год открытия ветеринарной клиники должен быть больше 1761 и меньше текущей даты;
* оклад должна быть больше 5 000 и меньше 300 000;
* цена процедуры должна быть больше 1000 и меньше 1 000 000;
* скидка на процедуру должна быть больше или равна 0 и меньше или равна 20;
* возраст животного должен быть больше 0;
* цена материала по этой процедуре должна быть больше или равна 0 и меньше или равна 1 000 000;
* стаж должен быть больше или равен 0 и меньше;
* ((текущий год – (дата рождения)) – 18);
* все даты должны быть меньше текущей даты.

# ОБОСНОВАНИЕ ВЫБОРА СУБД И ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ДЛЯ НАПИСАНИЯ КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

# Обоснование выбора СУБД MS SQL Server

# Система управления базами данных – совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

Для разработки приложения учета деятельности ветеринарных клиник города была выбрана реляционная СУБД MS SQL Server и графическая оболочка Windows Forms, позволяющая производить разработку с помощью встроенного конструктора с возможностью дальнейшего редактирования SQL скриптов. [1]

# Выбор MSSQL Server обусловлен удобством, простотой и функциональностью данной СУБД, а также наличием графической оболочки.

# Обоснование выбора инструментальных средств

# Для написания клиентской части приложения выбран язык программирования C#. Данный выбор обусловлен простотой и широкой функциональностью данного языка программирования, наличием библиотек для связи клиентского приложения с БД. Связь клиентской части приложения и базы данных реализована с помощью библиотеки Entity Framework 6.

Приложение будет состоять из множества форм. При запуске приложения появляется форма авторизации. После успешной авторизации пользователь переходит в главное меню, в котором может выбрать просмотр таблиц и справочников, составной формы, запросов и диаграмм. На форме Просмотр таблиц пользователю доступны редактирование, изменение и удаление записей, а также фильтрация, сортировка и поиск записей. На форме Просмотр справочников пользователю доступны редактирование, изменение и удаление записей.

# 3.ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ В ВЫБРАННОЙ СУБД

Проектирование базы данных представляет собой переход от неформального словесного описания информационной структуры предметной области к формализованному описанию объектов предметной области в терминах некоторой модели. [3]

# Проектирование концептуальной модели БД

Проектирование концептуальной модели БД состоит из нескольких основных этапов: инфологическое проектирование, даталогическое проектирование, физическое проектирование

# Инфологическое проектирование

На основании результатов анализа предметной области можно провести инфологическое проектирование информационной системы. В результате инфологического проектирования получим семантическую модель предметной области. Для этого необходимо выделить сущности, свойства сущностей и связи между сущностями. Рассмотрим основные сущности

# Среди свойств сущности Ветеринарные клиники название пункта, номер регистрационного пункта, год открытия, телефон, тип собственности, адрес, район города.

С сущностью Ветеринарные клиники связана сущность Сотрудник. Его свойство – ветеринарная клиника, в которой он работает.

# Среди свойств сущности Животные ФИО владельца, кличка, вид животного, возраст, условия содержания.

Среди свойств сущности Сотрудник – фамилия, имя, отчество, дата рождения, опыт работы, должность, оклад, вет. клиника, в которой он работает

# Таким образом, в результате инфологического проектирования была получена семантическая модель предметной области. В ней выделены сущности модели данных, их свойства

# Даталогическое проектирование

# В процессе даталогического представления выделяются таблицы и отношения между ними на основании построенной ранее семантической модели, схема данных приводится к третьей нормальной форме [4]

На рис. 3.1 изображена реляционная модель данных для предметной области

# «Ветеринарные клиники»

# https://sun9-56.userapi.com/impg/BVWRIxrTuopLRXrNxzML8BHBiRlq0APenUdmgA/47NAdwxyda4.jpg?size=1017x568&quality=95&sign=09e36a6ea866dfa483b22b305aea7c80&type=album

Рисунок 3.1 - Реляционная модель данных

# Основными таблицами в модели данных являются Ветеринарные клиники, Животные, Сотрудники, Процедуры, Владельцы и Лицензии. Справочники – Районы, Виды животных, Типы собственности, Должности, Классы животных, ВидыПроцедур

# Физическое проектирование

# На этапе физического проектирования необходимо выделить общую структуру базы данных и определить, какие типы данных будут использованы для полей таблиц.

Выбранные типы и общая структура указаны на в виде таблицы 3.1. Внешние ключи помечены в, первичные – п.

# Таблица 3.1 – Таблицы и их поля

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица | Поле | Назначение | Тип данных |
| Районы | Districtid | Код района (п) | integer |
| District | Район | text |
| Типы собственности | Type\_of\_possessionid | Код собственности (п) | integer |
| Type\_of\_possession | Тип собственности | text |
| Классы животных | Classid | Код класс (п) | integer |
| Class | Класс животного | text |
| Виды животных | Vidid | Код вида (п) | integer |
| Vid | Вид животного | text |
| Classid | Код семейства (в) | integer |
| Должность | Positiontid | Код должности (п) | integer |
| Position | Должность | text |
| Процедуры | ProcedureId | Код процедуры (п) | integer |
| ProcCost | Стоимость процедур | decimal |
| Discount | Скидка на процедуру | integer |
| MaterialsCost | Стоимость материала | decimal |
| DateHelping | Дата оказания помощи животному | date |
| animalid | Код животного(в) | integer |
| employeeid | Код сотрудника(в) | integer |
| Виды процедуры | Procedureid | Код вида процедуры (п) | integer |
| Procedure | Процедура | text |

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица | Поле | Назначение | Тип данных |
| Ветеринарная клиника | VeterinaryClinicId | Код ветеринарной клиники (п) | integer |
| Registration | Регистрационный номер пункта | integer |
| DistrictId | Код района (в) | integer |
| Type\_of\_possessionid | Код собственности (в) | integer |
| Year | Год основания | integer |
| Phone | Телефон | text |
| Рoint | Название пункта | text |
| Address | Адрес пункта | text |
| Владельцы | Ownerid | Код владельца (п) | integer |
| Name | Имя | text |
| Surname | Фамилия | text |
| Patronymic | Отчество | text |
| Birthday | Дата рождения хозяина | date |
| Telephone | Телефон хозяина | text |
| Животные | Animalid | Код животного (п) | integer |
| Ownerid | Код владельца (в) | integer |
| Vidid | Код вида (в) | integer |
| Age | Возраст животного | integer |
| Terms | Условия содержания животного | text |
| Лицензии | Licenseid | Код лицензии (п) | integer |
| License | Номер лицензии | text |
| Photo | Фото лицензии | byte |
| DateLicense | Срок окончания лицензии | date |

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица | Поле | Назначение | Тип данных |
| Employee | Employeeid | Код сотрудника (п) | integer |
| Name | Имя | text |
| Surname | Фамилия | text |
| Patronymic | Отчество | text |
| Birthday | Дата рождения | date |
| VeterinaryClinicId | Код ветеринарной клиники (в) | integer |
| Positionid | Код должности (в) | integer |
| Experience | Стаж | integer |
| Salary | Оклад | decimal |

# Создание таблиц, доменов, индексов

# На рисунке 3.2 представлен скрипт создания базы данных

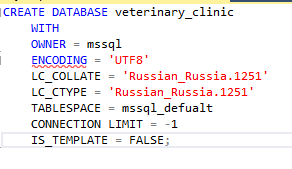
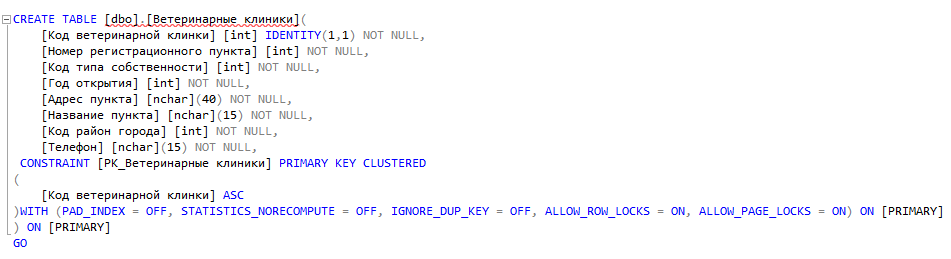


Рисунок 3.2 – Скрипт создания базы данных Ветеринарные клиники

# Создание таблиц

На рисунках 3.3 – 3.7 представлены скрипты создания некоторых таблиц. Первичные ключи генерируются автоматически, с помощью добавления ограничения identity [1]



# Рисунок 3.3 – Скрипт создания таблицы вет. Клиник

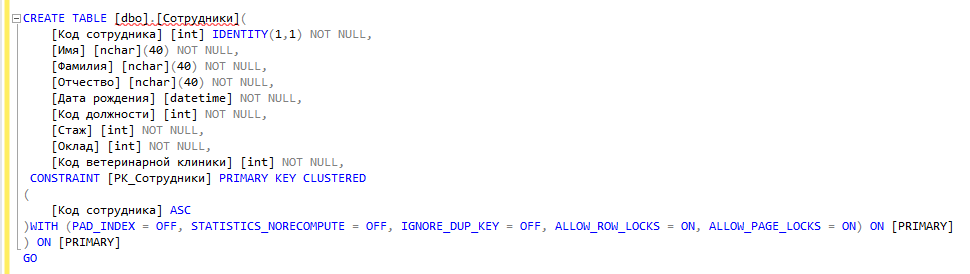
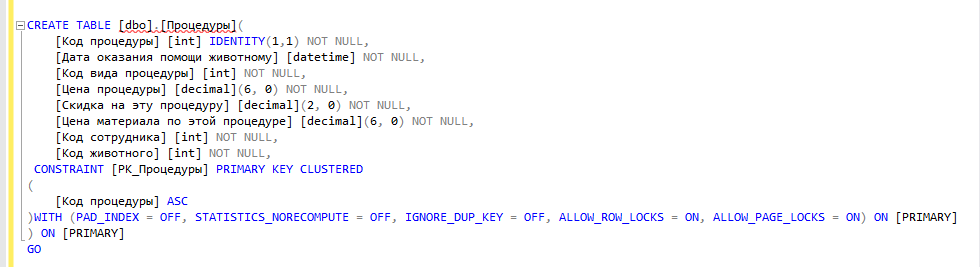


Рисунок 3.4 – Скрипт создания таблицы Сотрудник



# Рисунок 3.5 – Скрипт создания таблицы Процедуры

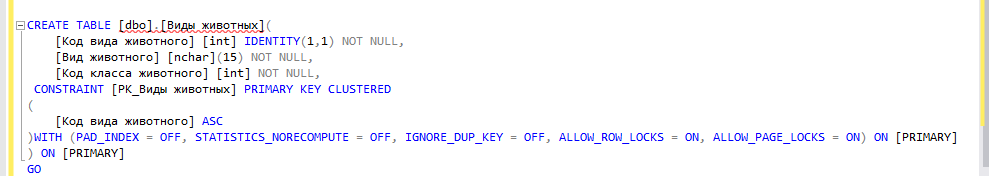
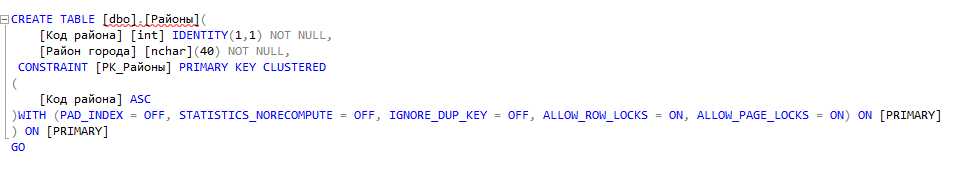


Рисунок 3.6 – Скрипт создания справочника Виды животных

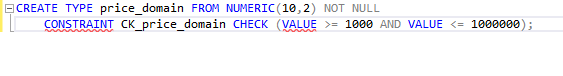


# Рисунок 3.7 – Скрипт создания справочника Районы

# Создание доменов

# Домен определяется заданием некоторого базового типа данных, к которому относятся элементы домена, и произвольного выражения, применяемого к элементу типа данных.

В базе данных Ветеринарные был создан домен для цены, скрипт создания которого показан на рисунке 3.8



# Рисунок 3.8 – Скрипт создания домена для цены

# 2.3.3 Создание индексов

# Индекс – объект базы данных, созданный с целью повышения производительности поиска данных [1]

В базе данных Ветеринарные клиники было создано несколько индексов, показанных на рисунках 3.9 – 3.10



# Рисунок 3.9 – Скрипт создания индекса для поля Кличка Животного Животные



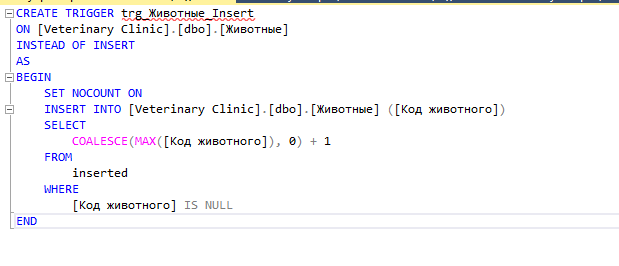
Рисунок 3.10 – Скрипт создания индекса для поля Имя Сотрудники

# Разработка триггеров

Триггер – сочетание хранимой в базе данных процедуры и события, которое заставляет его выполняться. [2]

# В курсовом проекте необходимо разработать триггеры (по одному для каждого типа событий)

Для таблицы Животные были разработаны триггеры AFTER INSERT, AFTER UPDATE, AFTER DELETE которые вводят в поле кличка животного «смена владельца», если владельца сменили, указанные на рисунках 3.11-3.12



# Рисунок 3.11 – Скрипт создания триггера AFTER INSERT

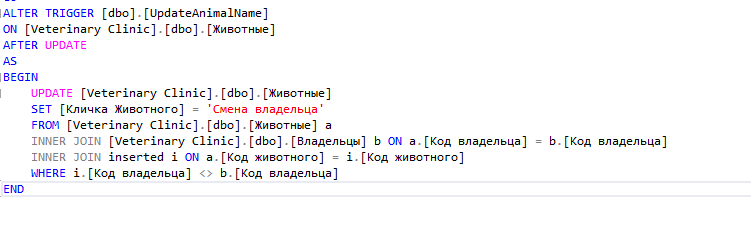
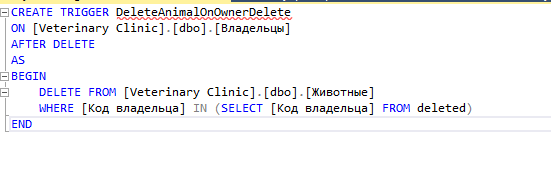


Рисунок 3.12 – Скрипт триггера AFTER UPDATE

Каскадное удаление реализовано с помощью триггера AFTER DELETE для таблицы Животные, представленного на рисунке 3.12. До удаления из таблицы животные удаляются все связанные записи из таблицы процедуры



# Рисунок 3.12 – Скрипты создания триггера AFTER DELETE

# Также все 3 вида триггеров реализованы для таблицы Сотрудники. После изменения даты рождения меняется стаж. Также триггер проверяет значение зарплаты. Создано каскадное удаление, при котором удаление записи Сотрудника повлечёт за собой удаление записей процедуры. Создание триггера и триггерной функции показано на рисунке 3.13

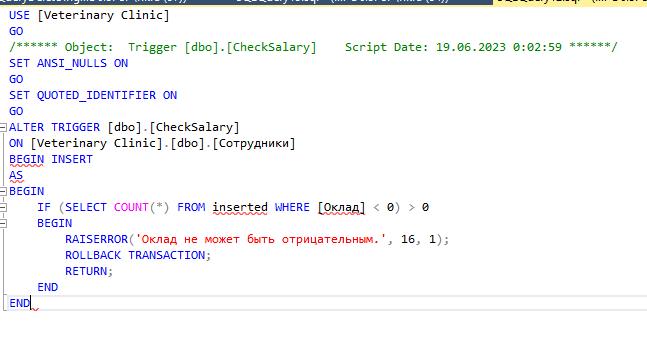


Рисунок 3.13 – Скрипт триггера BEFORE INSERT

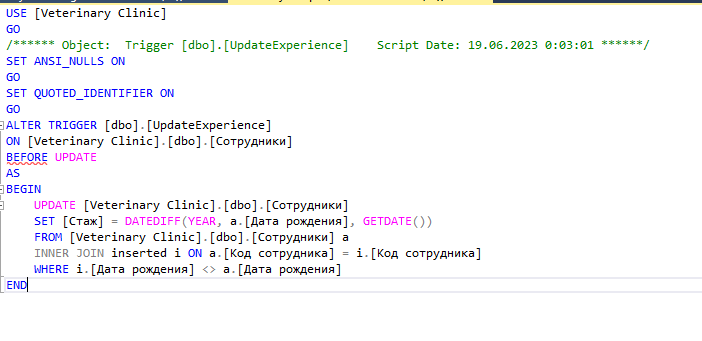


Рисунок 3.14 – Скрипт триггера BEFORE UPDATE

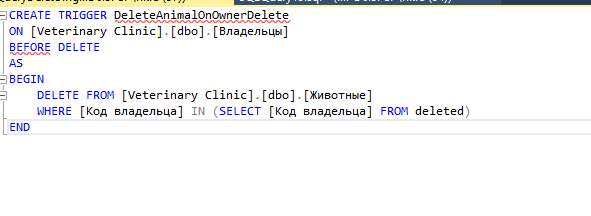


Рисунок 3.15 – Скрипт триггера BEFORE DEL

# Проектирование запросов к базе данных

# Запрос – это обращение к базе данных для получения требуемых данных и выполнения действий над ними. [1]

# Простые запросы

# На рисунке 3.16-3.17 представлено симметричное внутреннее соединение с условием по внешнему ключу - Учёт сотрудников по должности

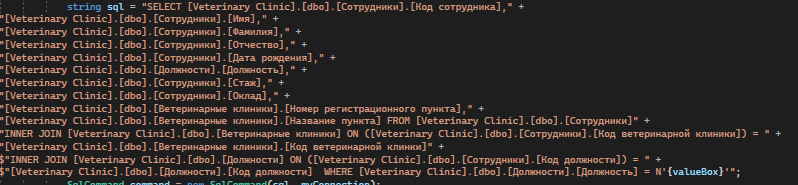
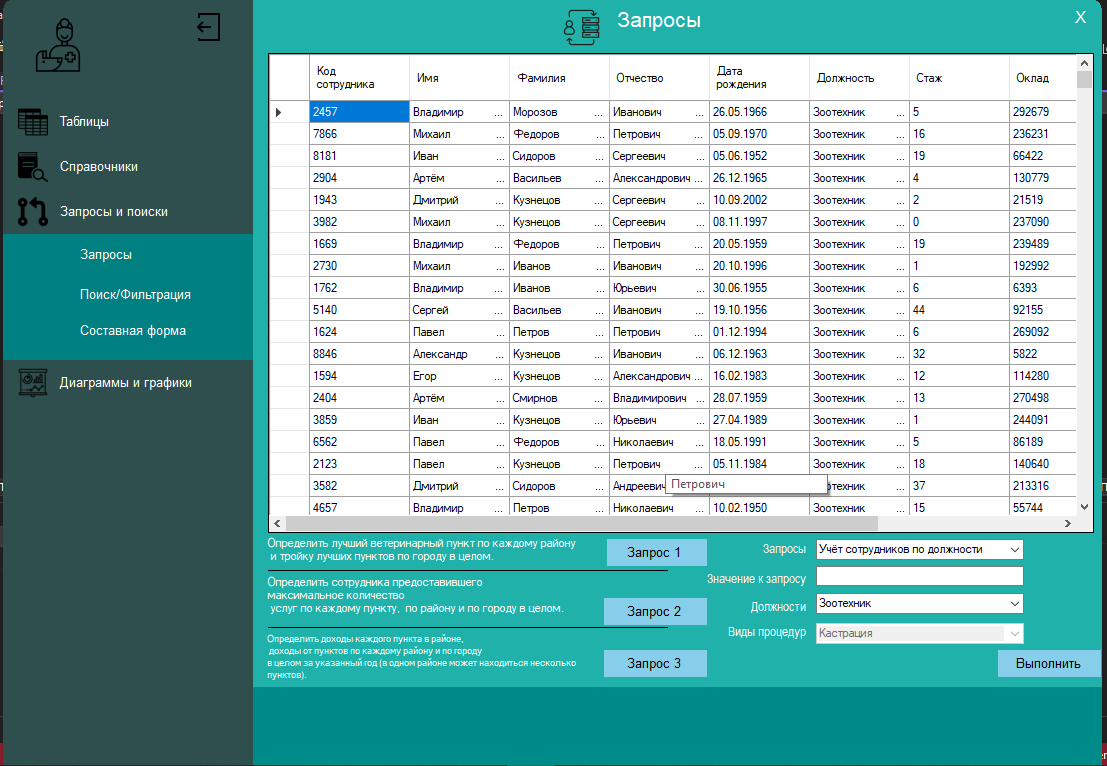
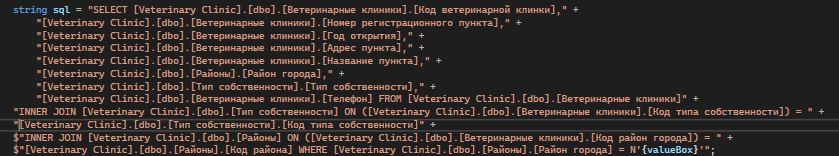


Рисунок 3.16 - Код запроса



# Рисунок 3.17 - Результат выполнения запроса

На рисунке 3.18-3.19 представлено симметричное внутреннее соединение с условием по внешнему ключу – Ветеринарные клиники в определённом районе



# Рисунок 3.18 - Код запроса

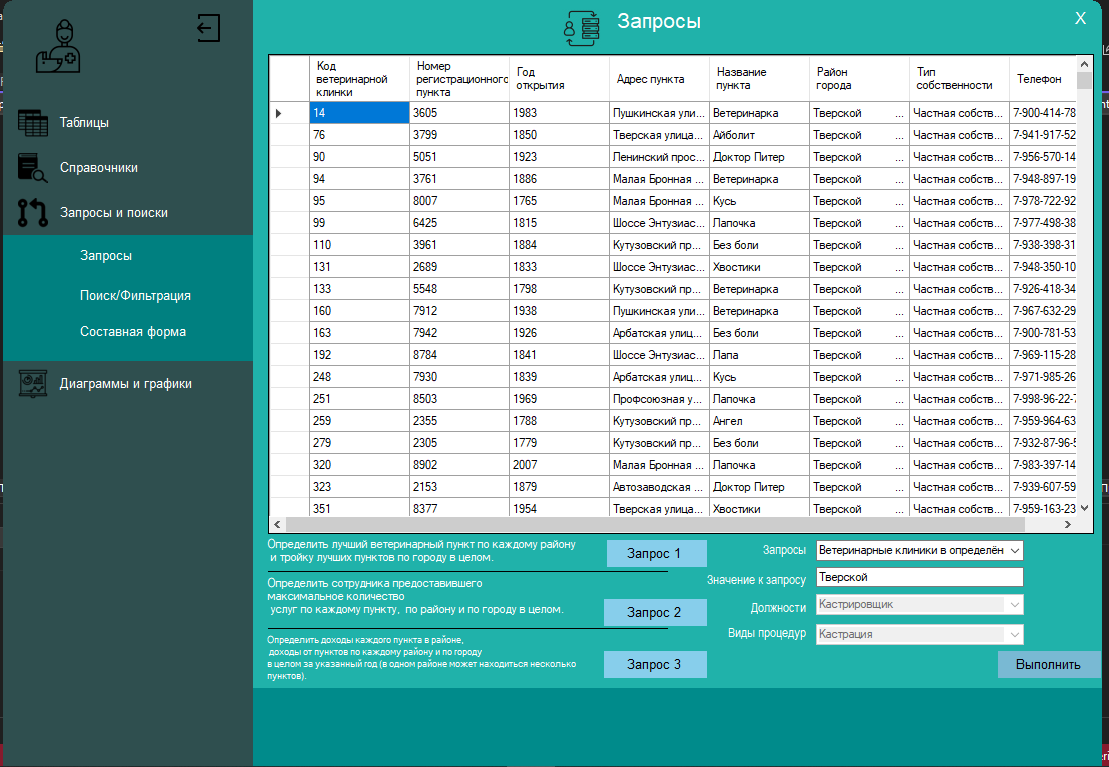


Рисунок 3.19 - Результат выполнения запроса

На рисунке 3.20-3.21 представлено симметричное внутреннее соединение с условием по дате – Процедуры оказанные в опред. дату

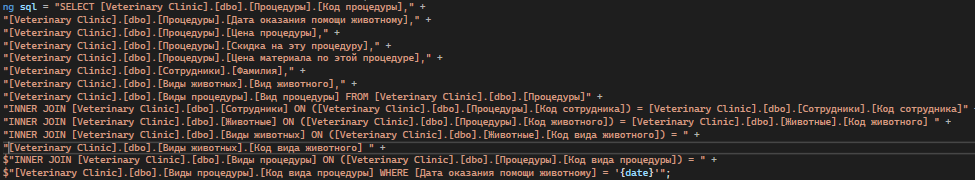
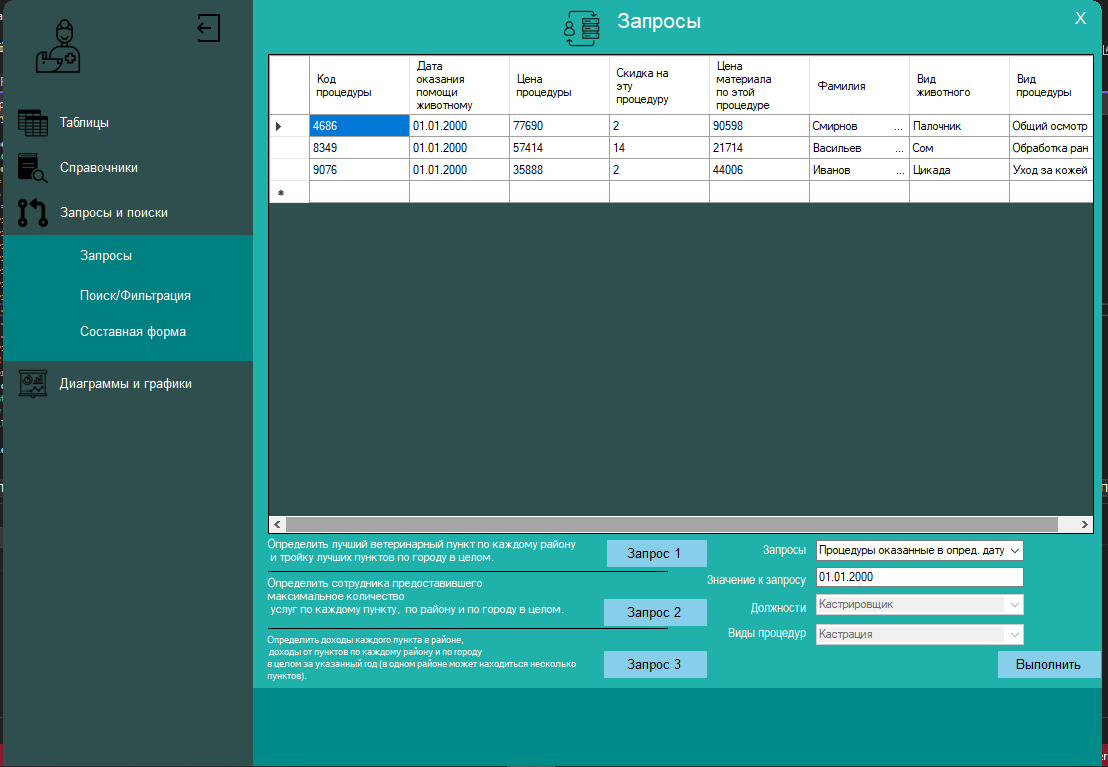
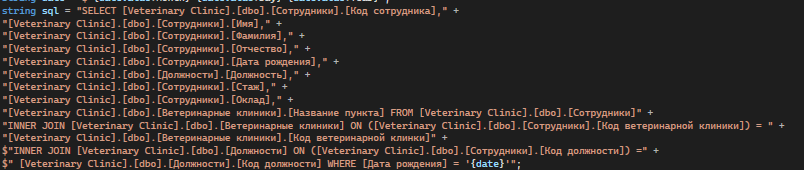


Рисунок 3.20 - Код запроса



# Рисунок 3.21 - Результат выполнения запроса

На рисунке 3.22-3.23 представлено симметричное внутреннее соединение с условием по дате – Сотрудники, родившиеся в опред. дату



# Рисунок 3.22 - Код запроса

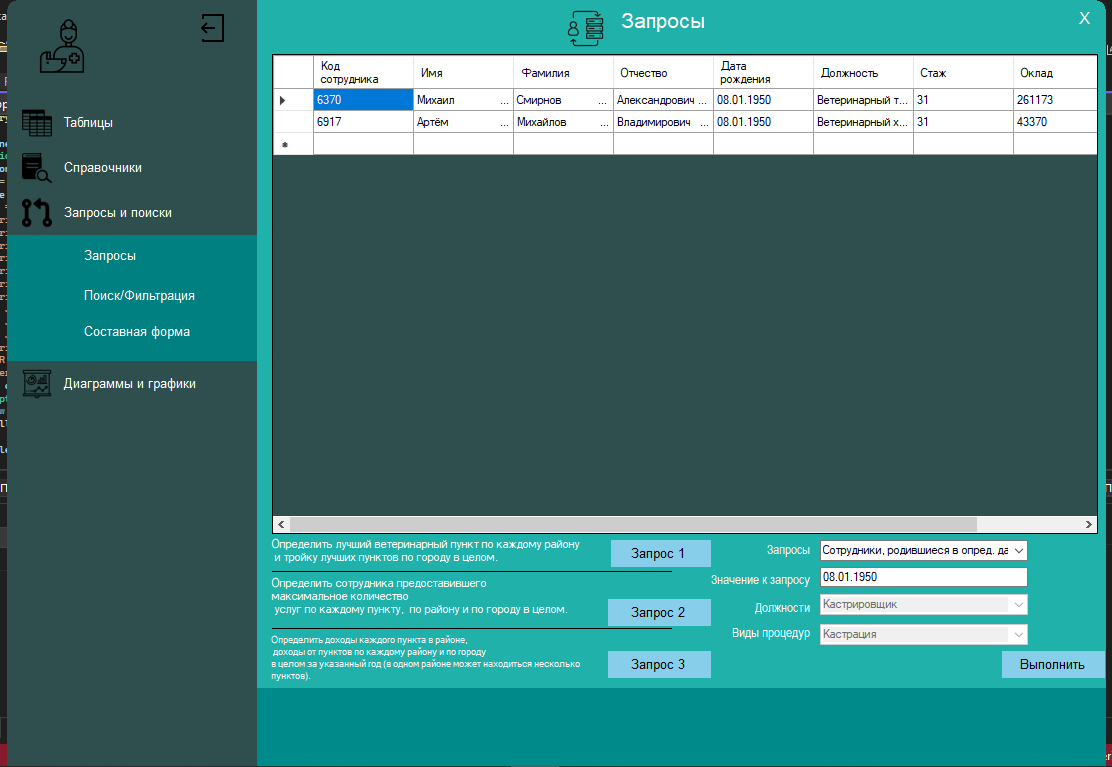


Рисунок 3.23 - Результат выполнения запроса

# На рисунке 3.24-3.29 представлено симметричное внутреннее соединение без условия – вывод информации для таблиц Ветеринарные клиники, Сотрудники и Процедуры

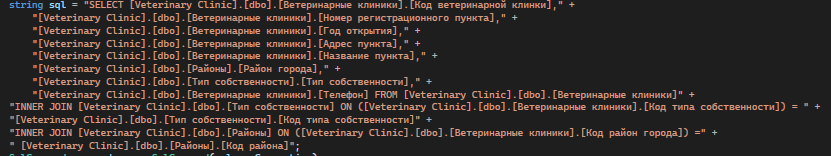
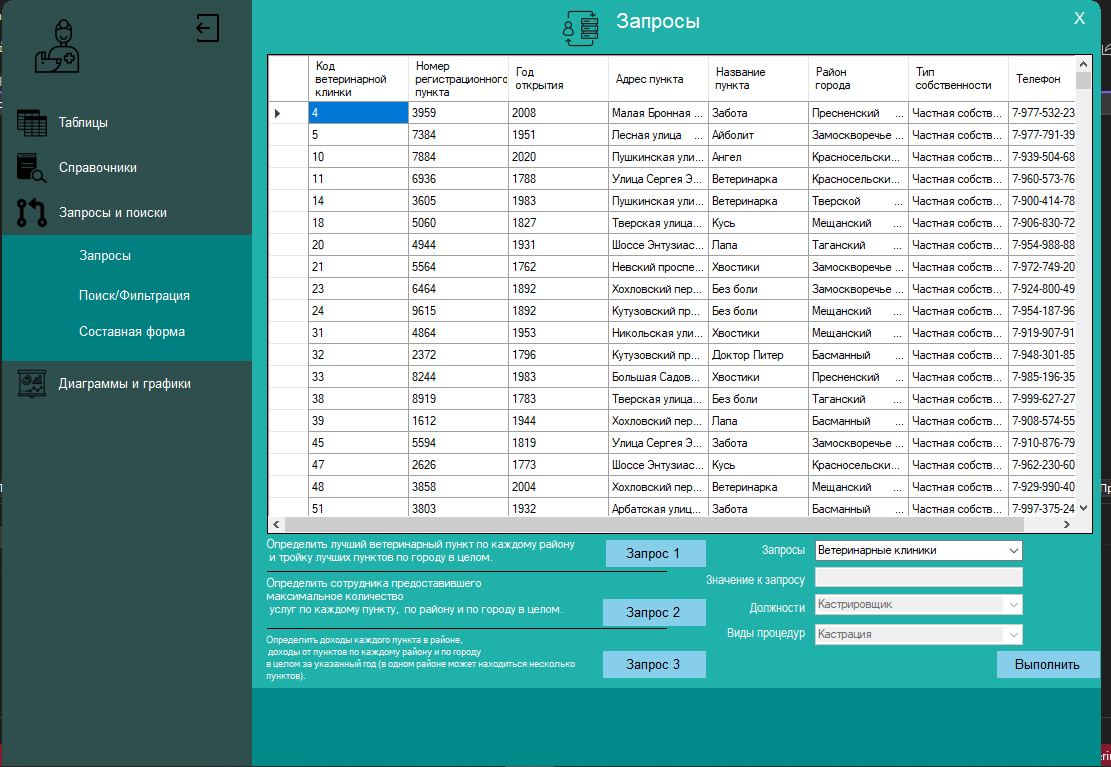


Рисунок 3.24 - Код запроса для таблицы вет. Клиники



# Рисунок 3.25 - Результат выполнения запроса

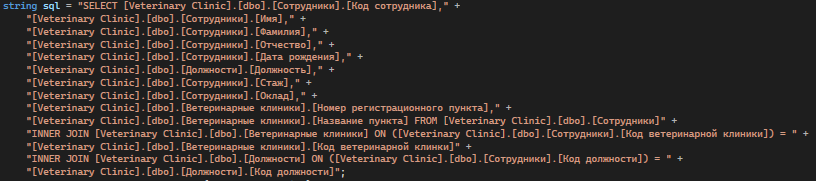
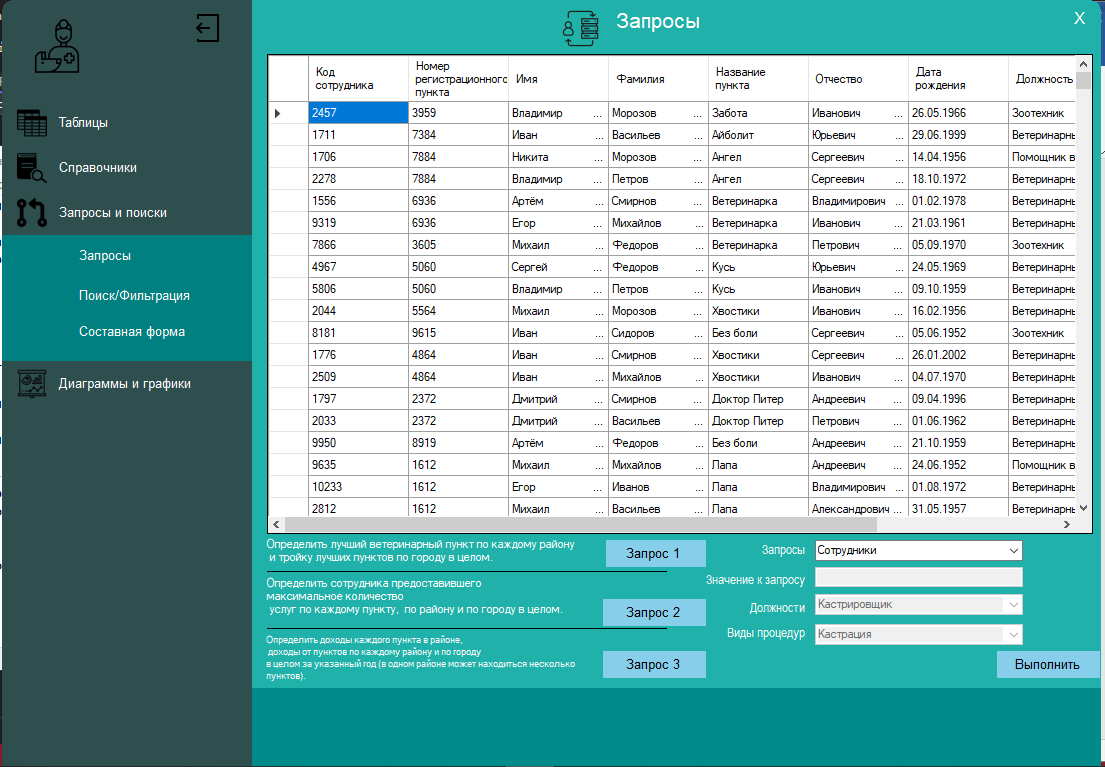


Рисунок 3.26 - Код запроса для таблицы Сотрудники



# Рисунок 3.27 Результат выполнения запроса

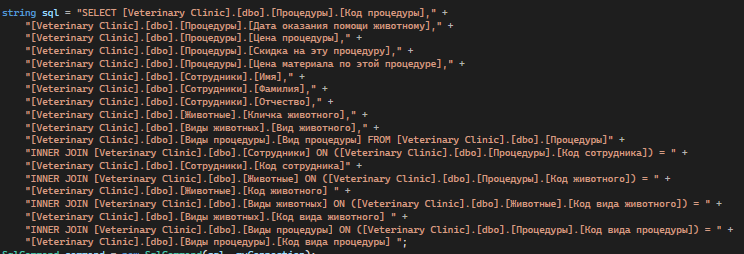
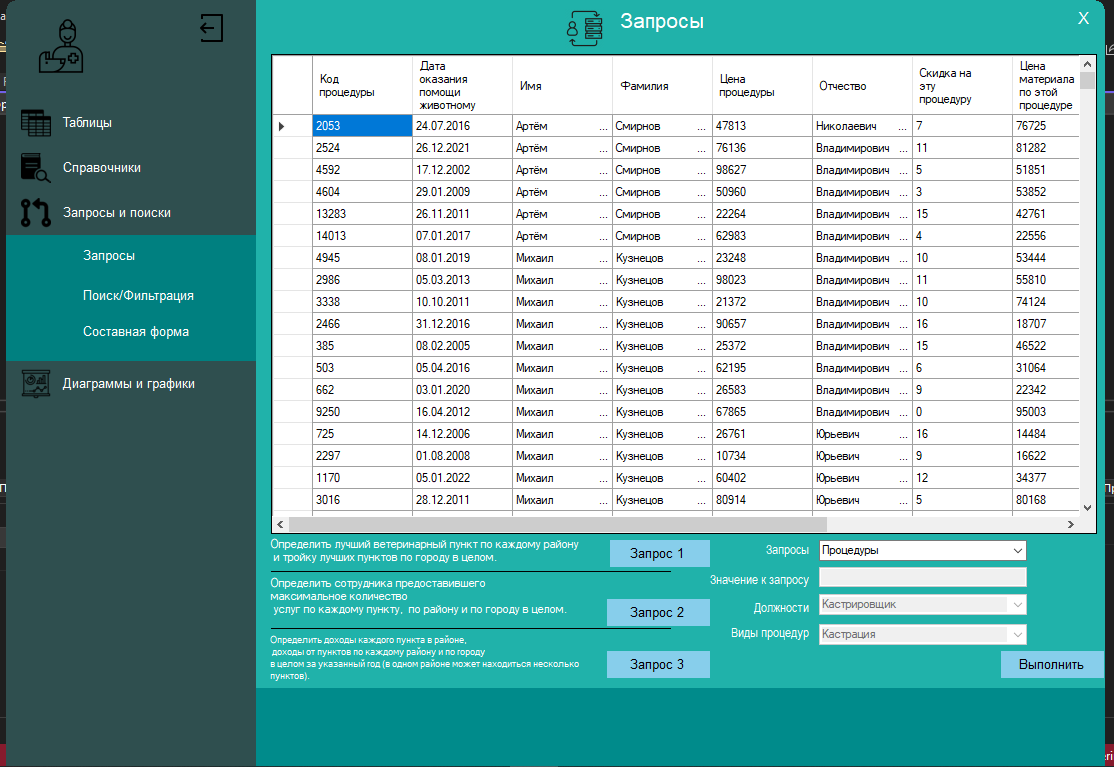
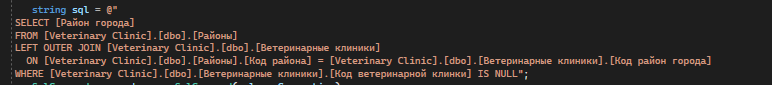


Рисунок 3.28 - Код запроса для таблицы Процедуры



# Рисунок 3.29 - Результат выполнения запроса

На рисунке 3.30-3.31 представлено левое внешнее соединение Районы, в которых нет вет. клиник



# Рисунок 3.30 - Код запроса

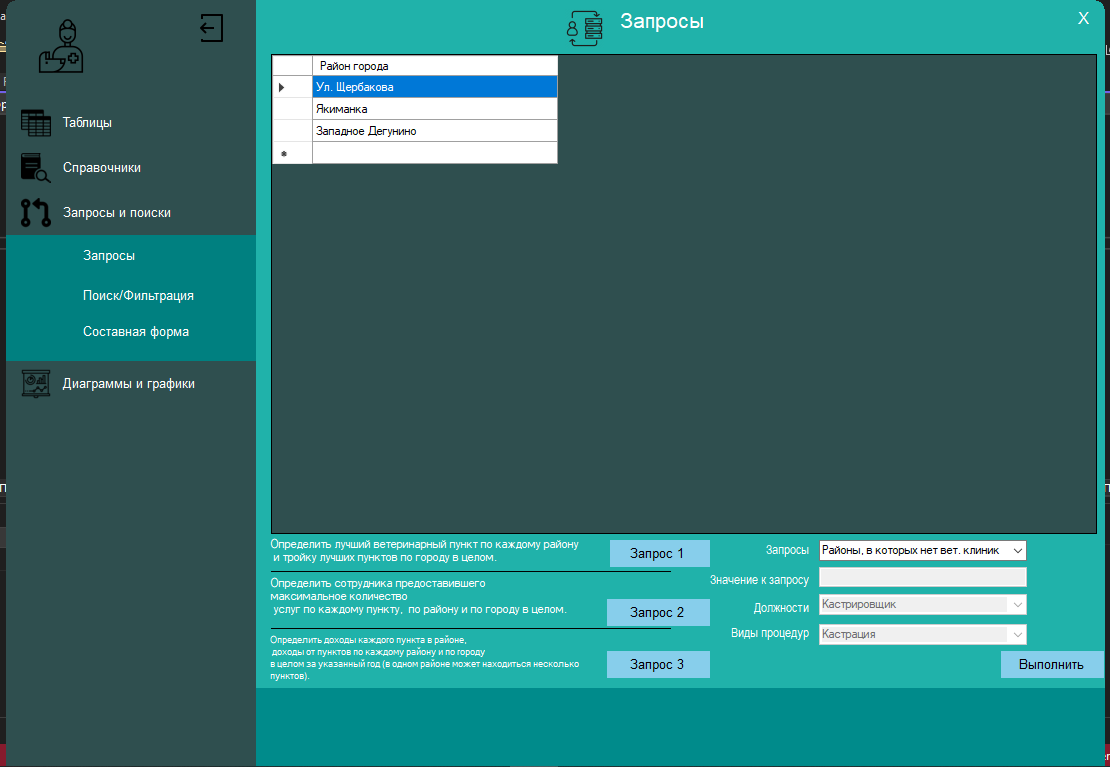
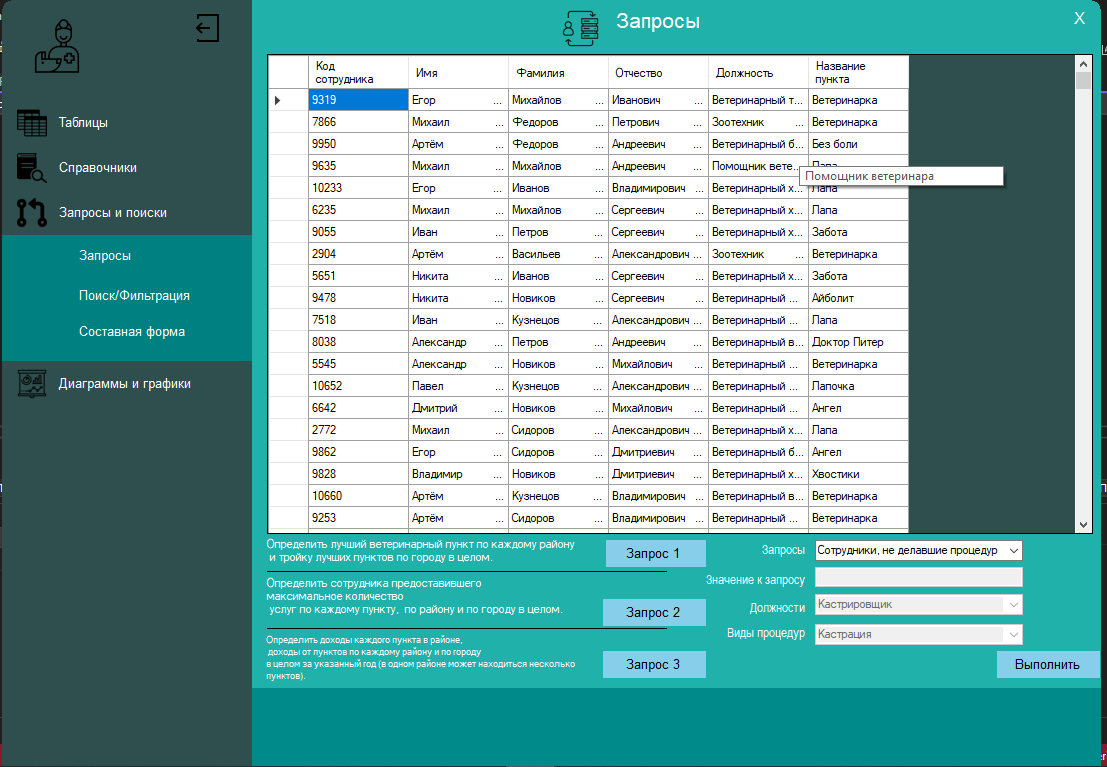


Рисунок 3.31 - Результат выполнения запроса

# На рисунке 3.32-3.33 представлено правое внешнее соединение - Сотрудники, не делавшие процедур

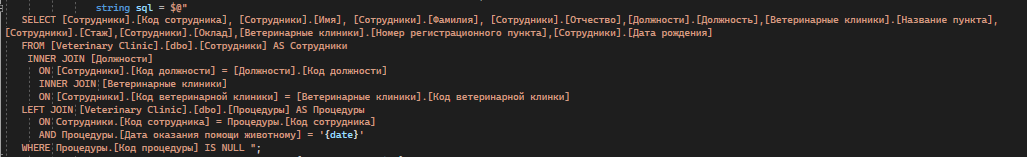
# 

Рисунок 3.32 - Код запроса



# Рисунок 3.33 - Результат выполнения запроса

На рисунке 3.34-3.35 представлен запрос на запросе по принципу левого соединения - Сотрудники, не делавшие процедур опред. даты



# Рисунок 3.34 - Код запроса

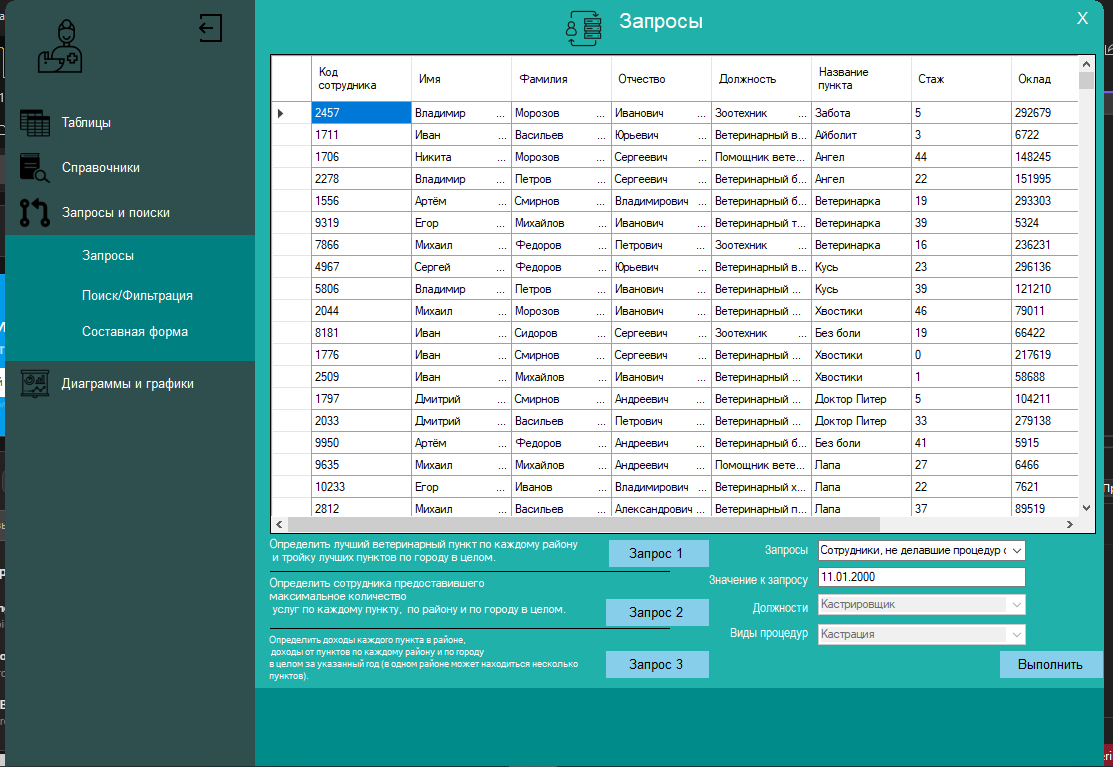
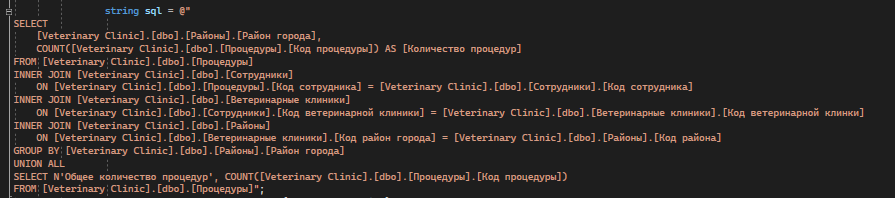


Рисунок 3.45 - Результат выполнения запроса

# Итоговые запросы

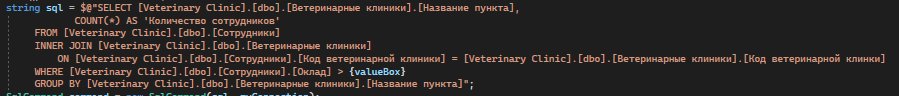
На рисунке 3.36-3.37 представлен итоговый без условия - Количество проведённых процедур всего и в каждом районе

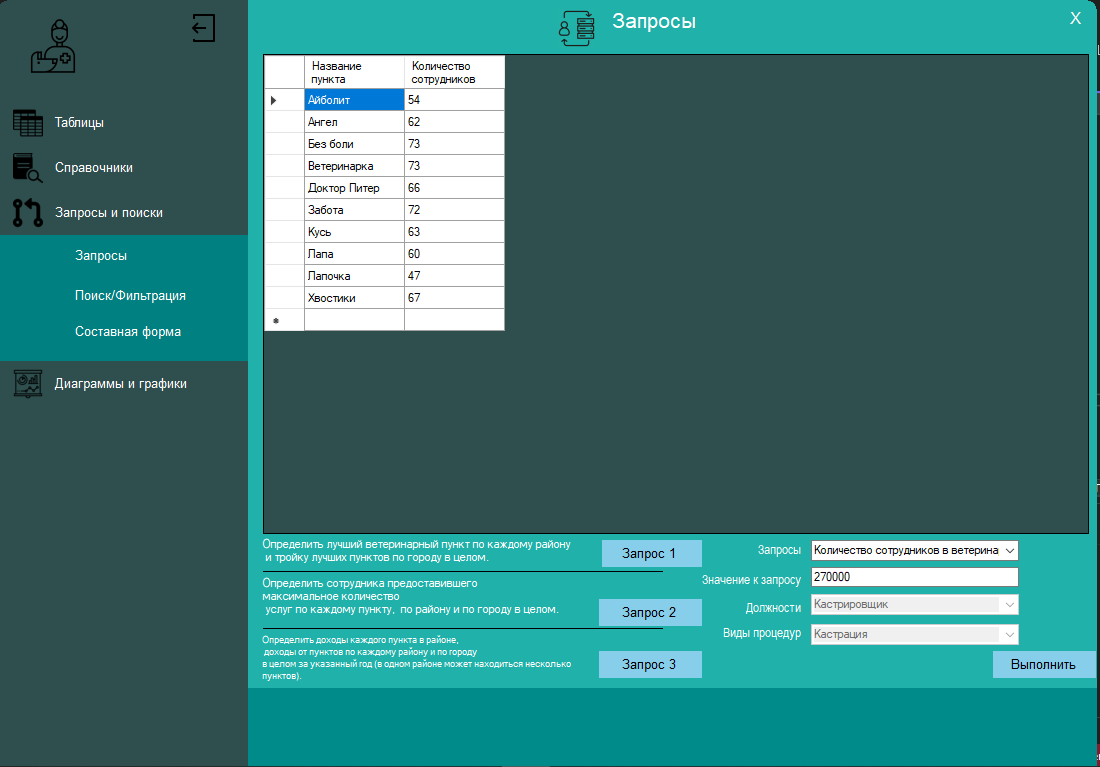


# Рисунок 3.36 - Код запроса

Рисунок 3.37 - Результат выполнения запроса

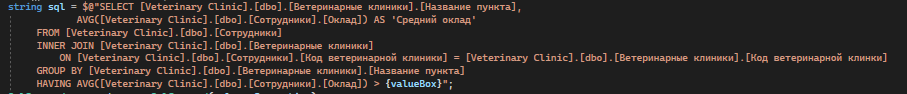
# На рисунке 3.38-3.39 представлен итоговый с условием на данные - Количество сотрудников в ветеринарных клиниках с окладом больше указанного



Рисунок 3.38 - Код запроса

# Рисунок 3.39 - Результат выполнения запроса

На рисунке 3.40-3.41 представлен итоговый с условием на группы - Ветеринарные клиники, где средний оклад сотрудников больше указанного



# Рисунок 3.40 - Код запроса

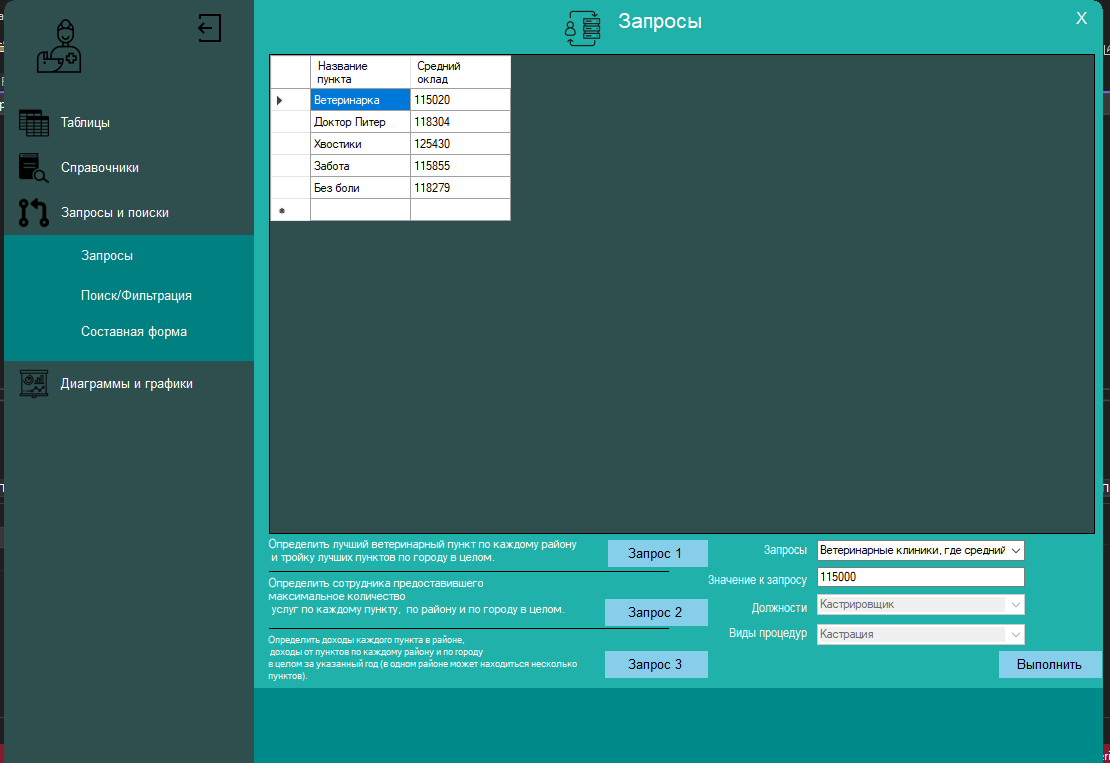
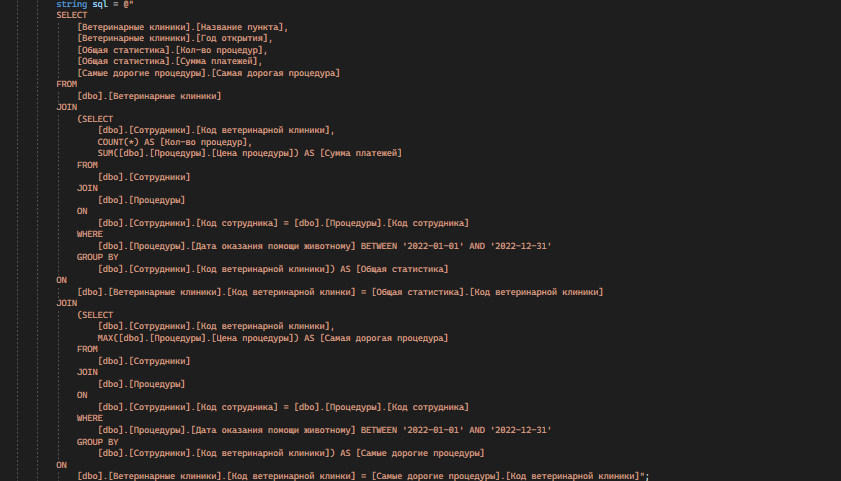


Рисунок 3.41 - Результат выполнения запроса

# На рисунке 3.42-3.43 представлен итоговый с условием на группы и данные

- Список клиник с количеством выполненных процедур, общей суммой процедур и самой дорогой



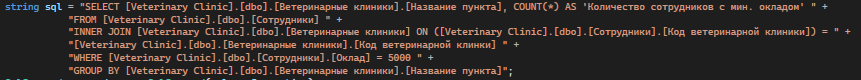
# Рисунок 3.42 - Код запроса

# 

Рисунок 3.43 - Результат выполнения запроса

# На рисунке 3.44-3.45 представлен запрос на запросе по принципу итогового запроса

- Количество сотрудников с минимальным окладом в каждой ветеринарной клинике



# Рисунок 3.44 - Код запроса

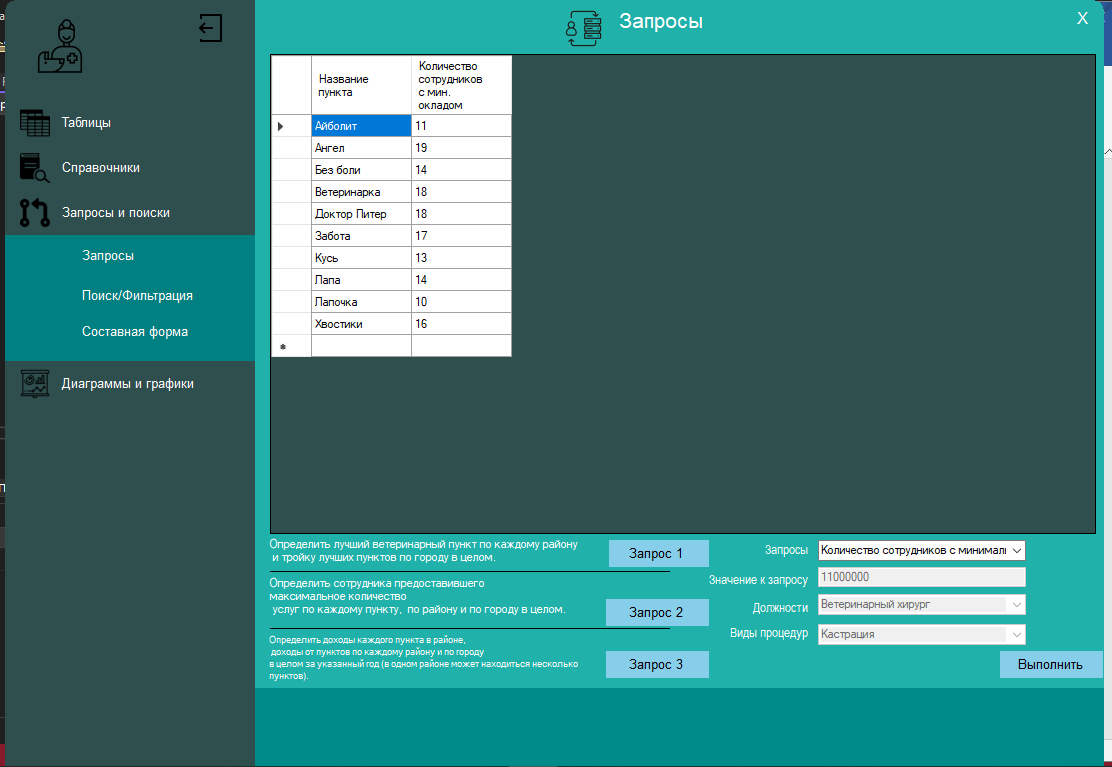


Рисунок 3.45 - Результат выполнения запроса

# На рисунке 3.46-3.47 представлен запрос с подзапросом (IN) - для поиска животных определенных видов(всех собак)

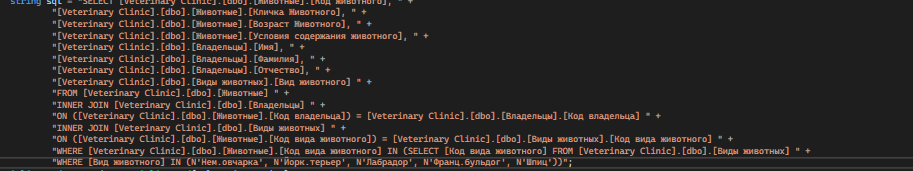
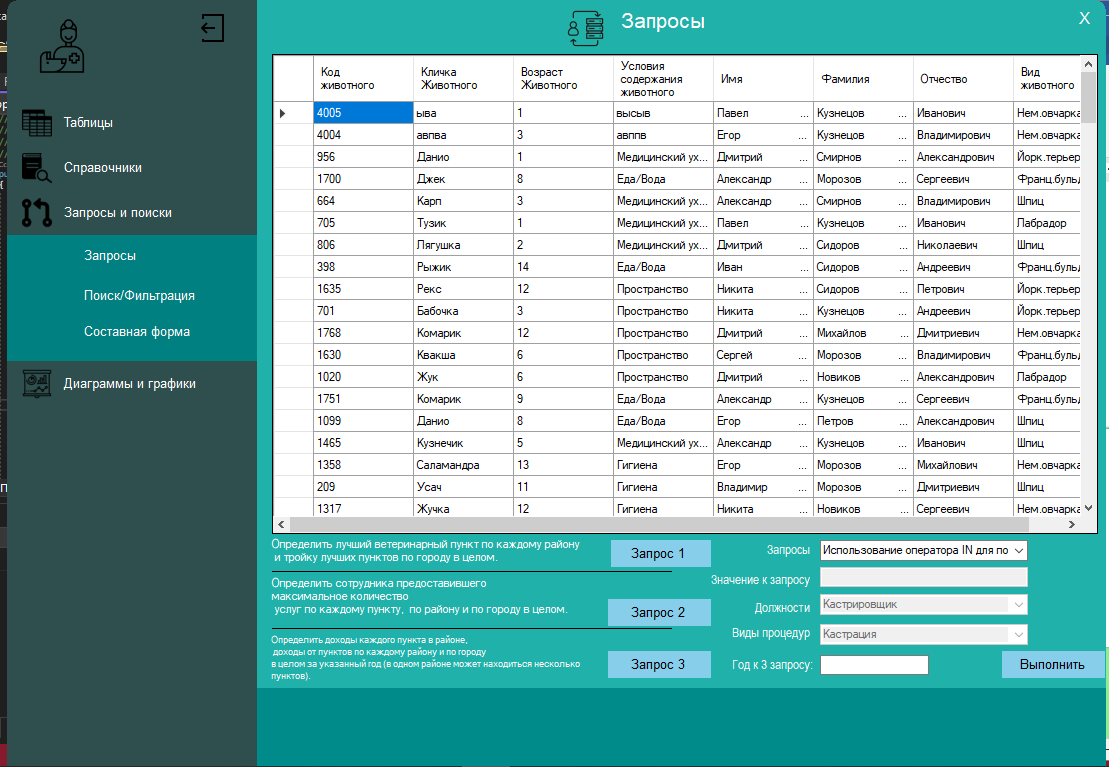
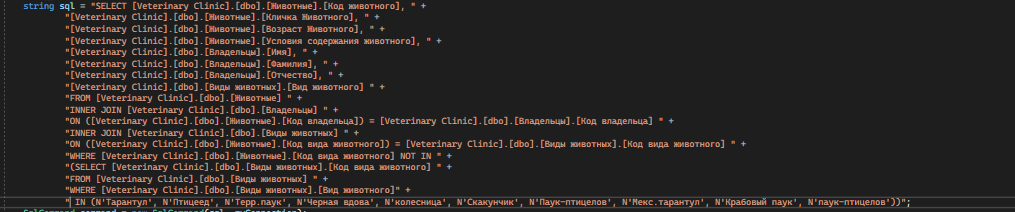


Рисунок 3.46 - Код запроса



# Рисунок 3.47 Результат выполнения запроса

На рисунке 3.48-3.49 представлен запрос с подзапросом (NOT IN) - для поиска животных, которые не относятся к определенным видам(паукообразные)



# Рисунок 3.48 - Код запроса

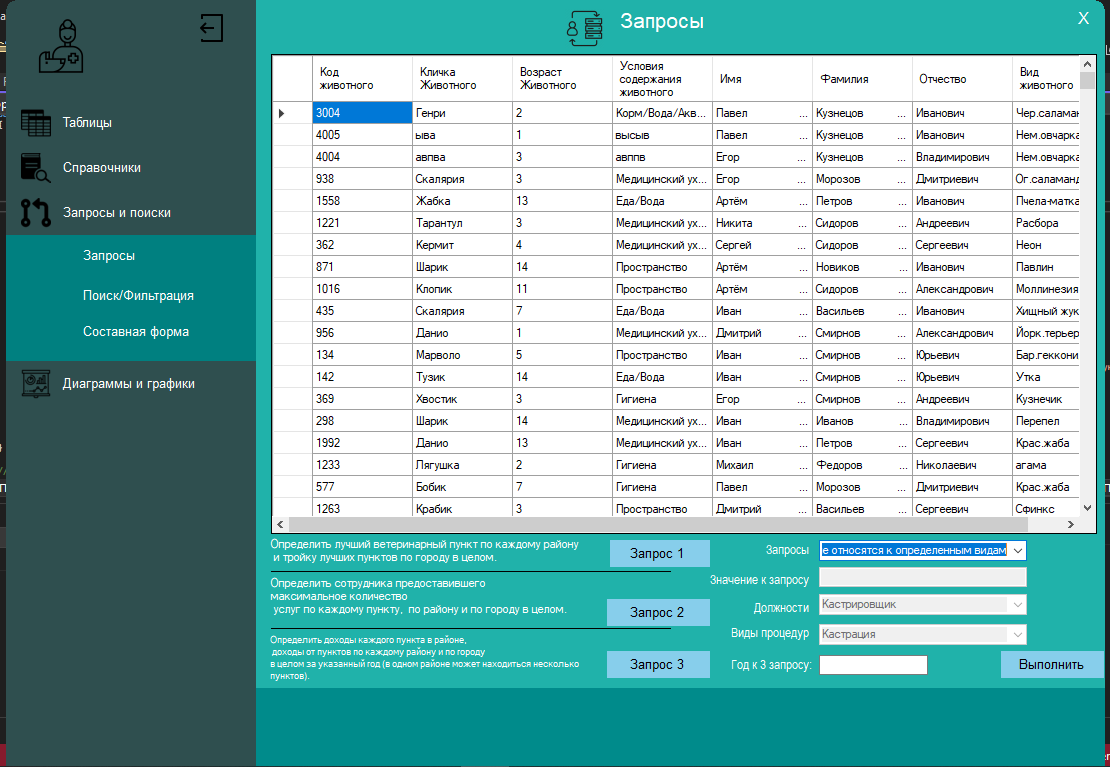


Рисунок 3.49 - Результат выполнения запроса

# На рисунке 3.50-3.51 представлен запрос с подзапросом (CASE) – для вычисления скидки на процедуры

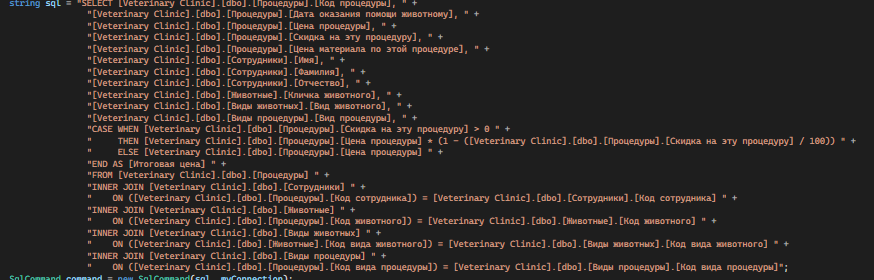
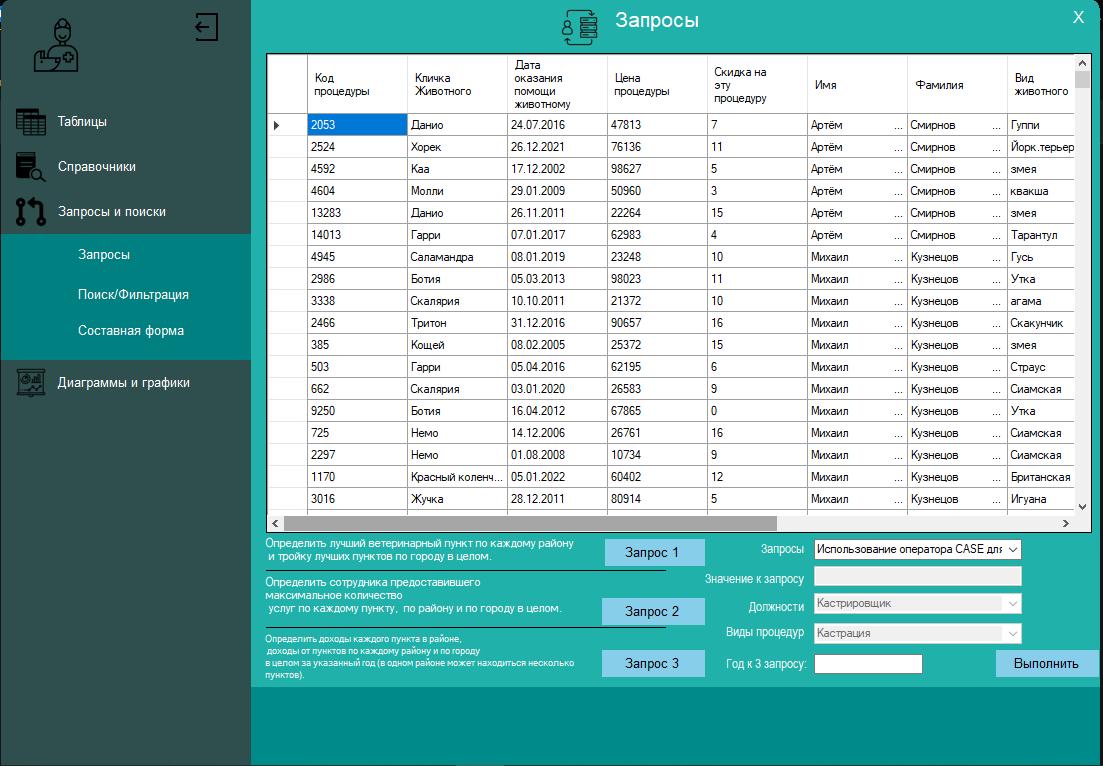
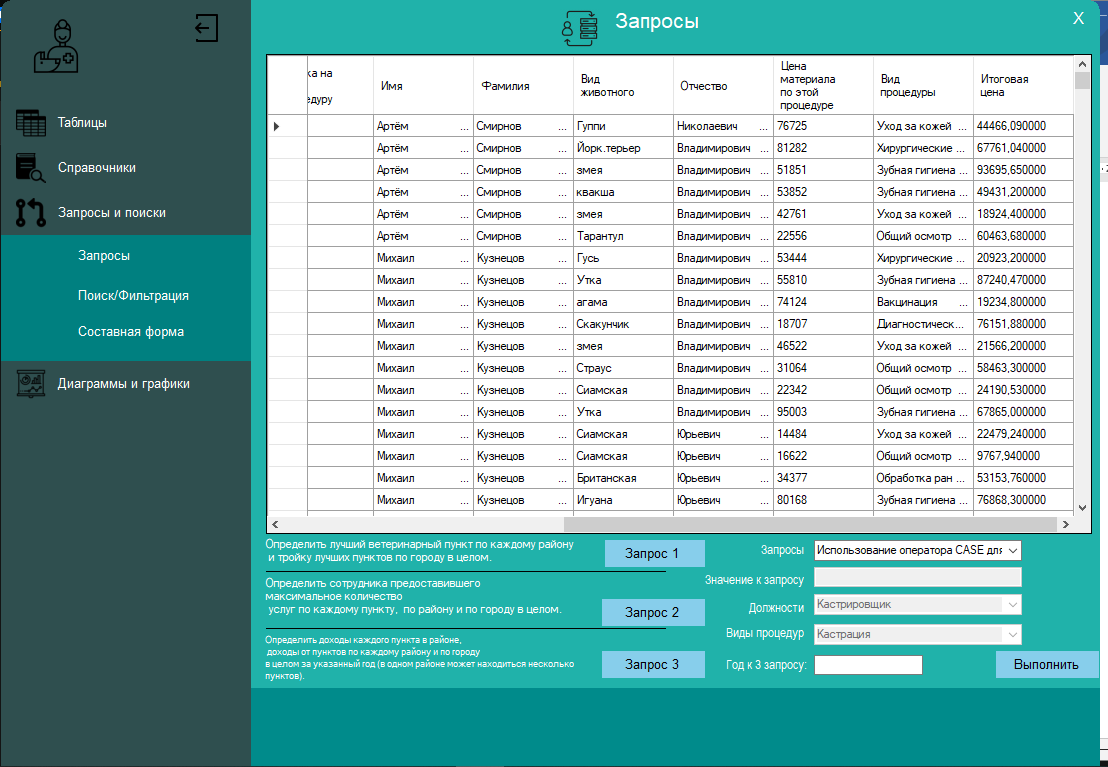


Рисунок 3.50 - Код запроса

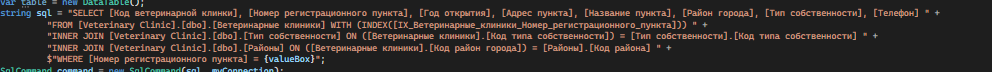


# Рисунок 3.51.1 - Результат выполнения запроса



# Рисунок 3.51.2 - Результат выполнения запроса

На рисунке 3.52-3.53 представлен итоговый запрос с индексом – Процедуры с использованием индекса (по коду животного)



# Рисунок 3.52 - Код запроса



Рисунок 3.53 - Результат выполнения запроса

# На рисунке 3.54-3.55 представлен итоговый запрос с без индекса – Процедуры без использования индекса (по виду процедуры)

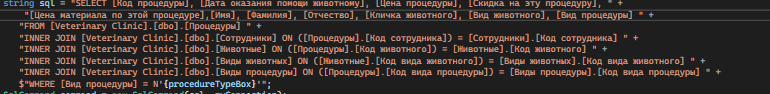
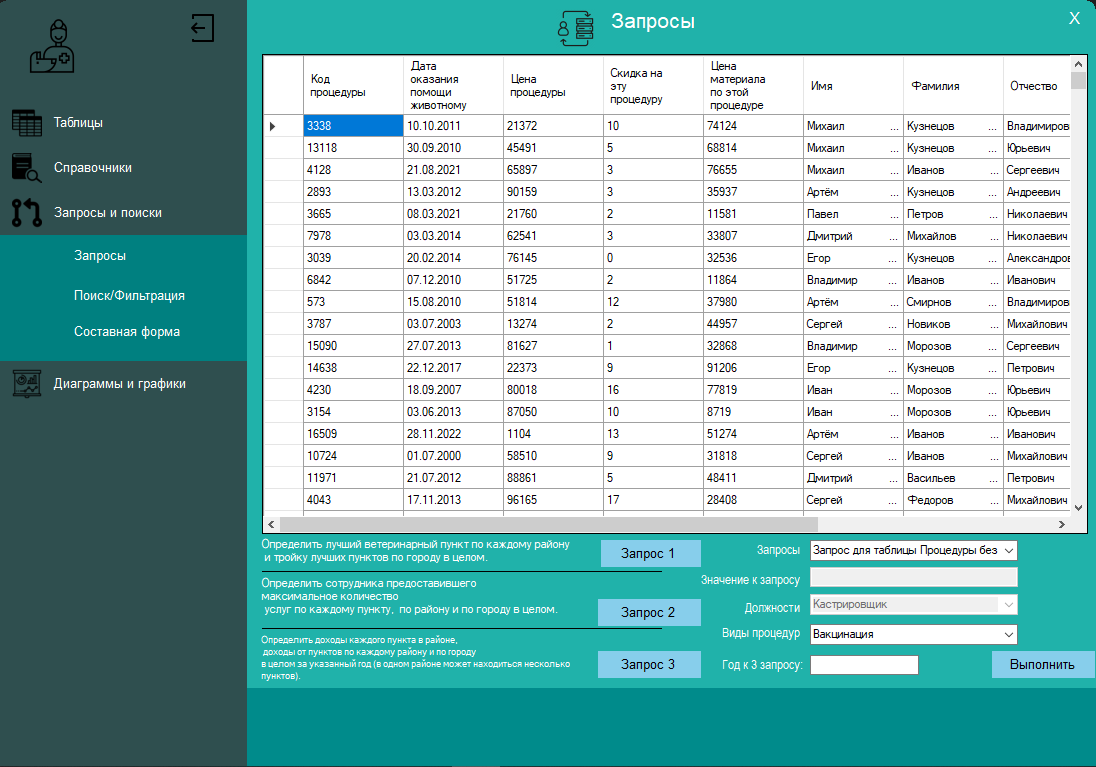
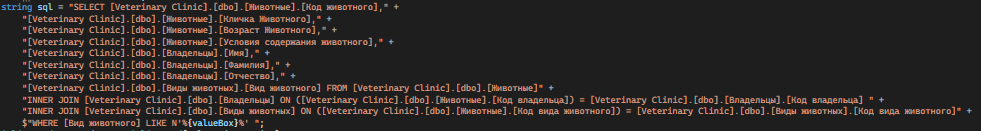


Рисунок 3.54 - Код запроса



# Рисунок 3.55 - Результат выполнения запроса

На рисунке 3.56-3.57 представлен итоговый запрос с условием по маске – Животные с использованием условия по маске (по виду животного)



# Рисунок 3.56 - Код запроса

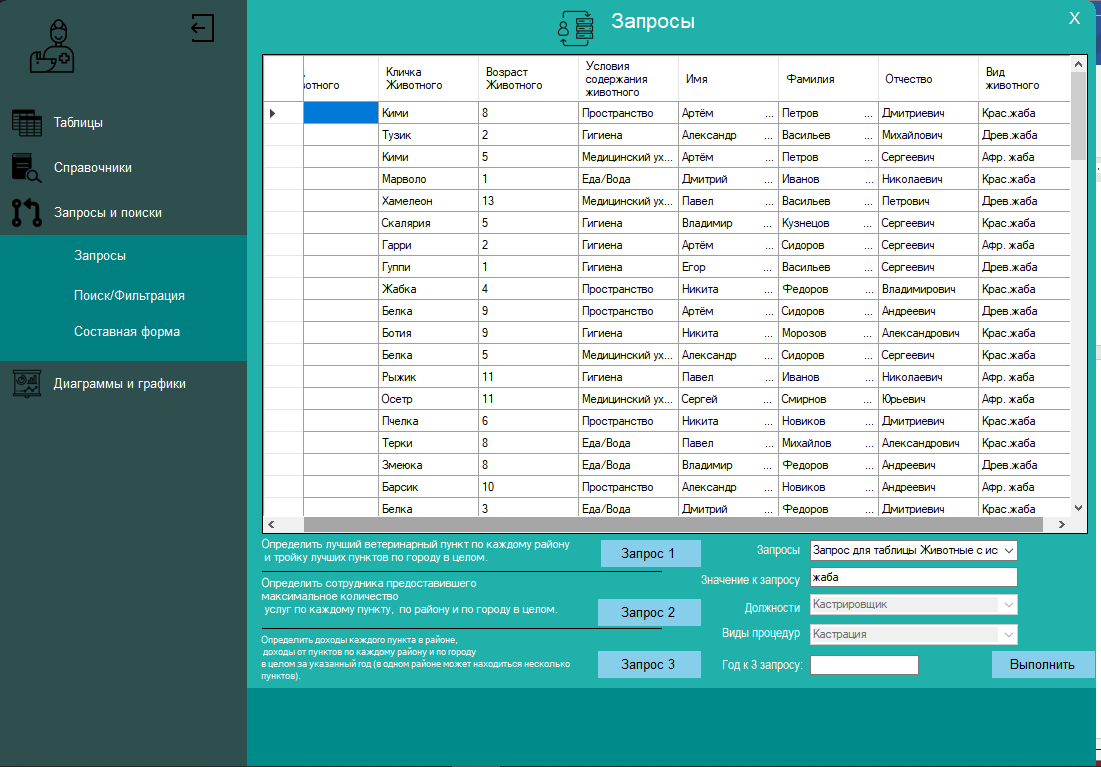
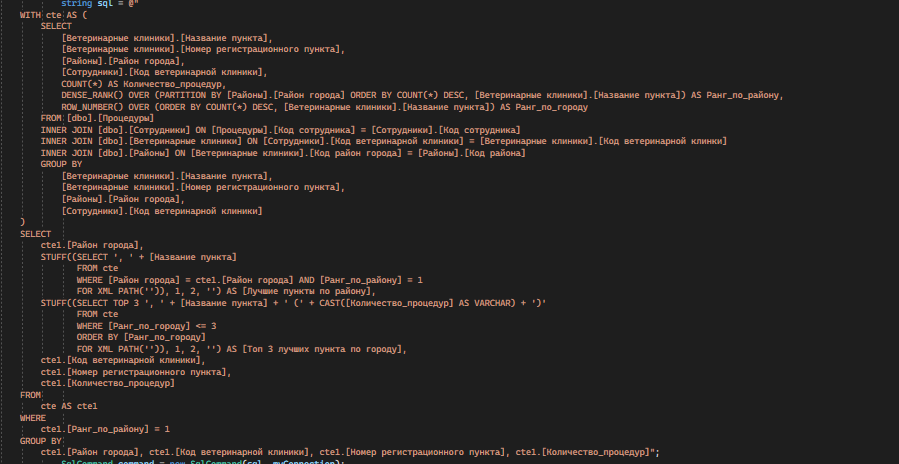


Рисунок 3.57 - Результат выполнения запроса

# Запросы из курсового проекта

На рисунке 3.58-3.59 представлен запрос из курсовой работы – определить лучший ветеринарный пункт по каждому району и тройку лучших пунктов по городу в целом



# Рисунок 3.58 - Код запроса

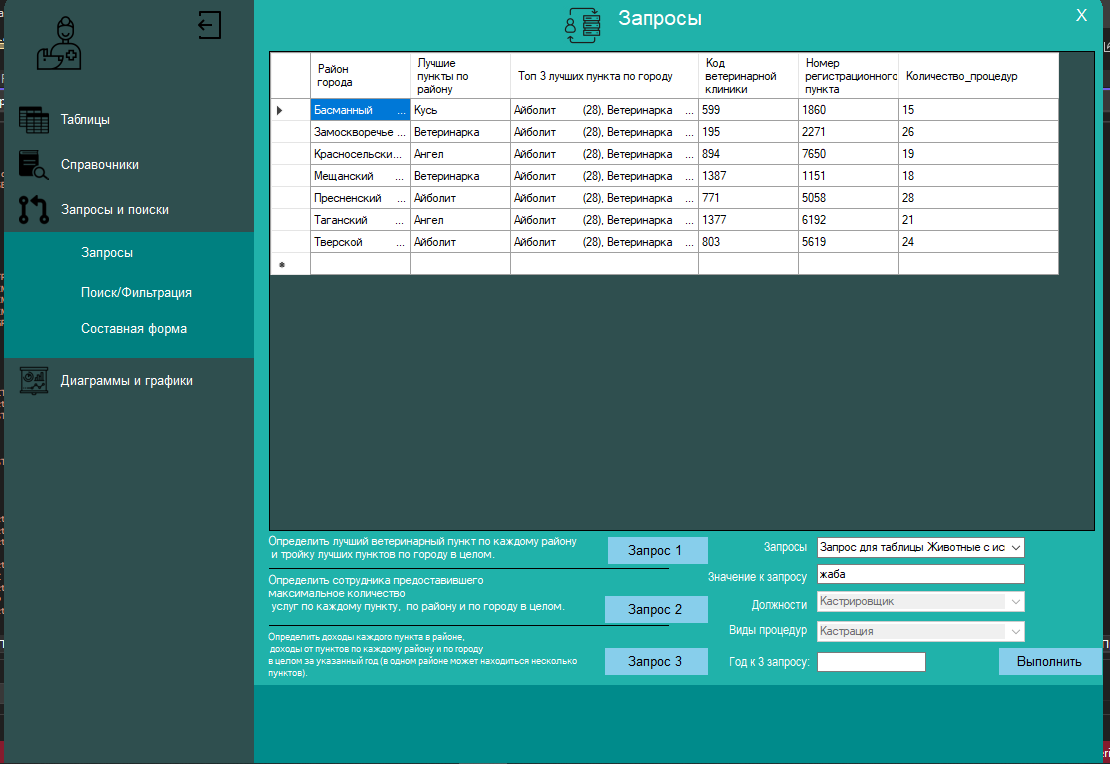


Рисунок 3.59 - Результат выполнения запроса

# На рисунке 3.60-3.61 представлен запрос из курсовой работы – определить сотрудника предоставившего максимальное количество услуг по каждому пункту, по району и по городу в целом

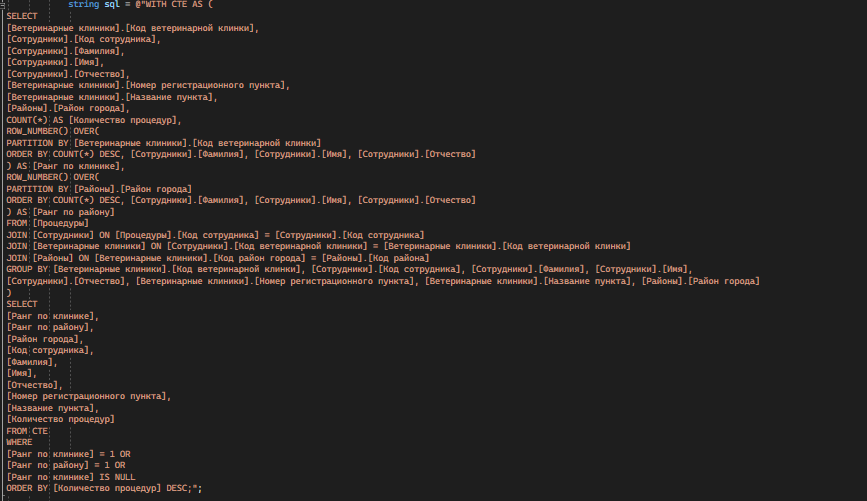
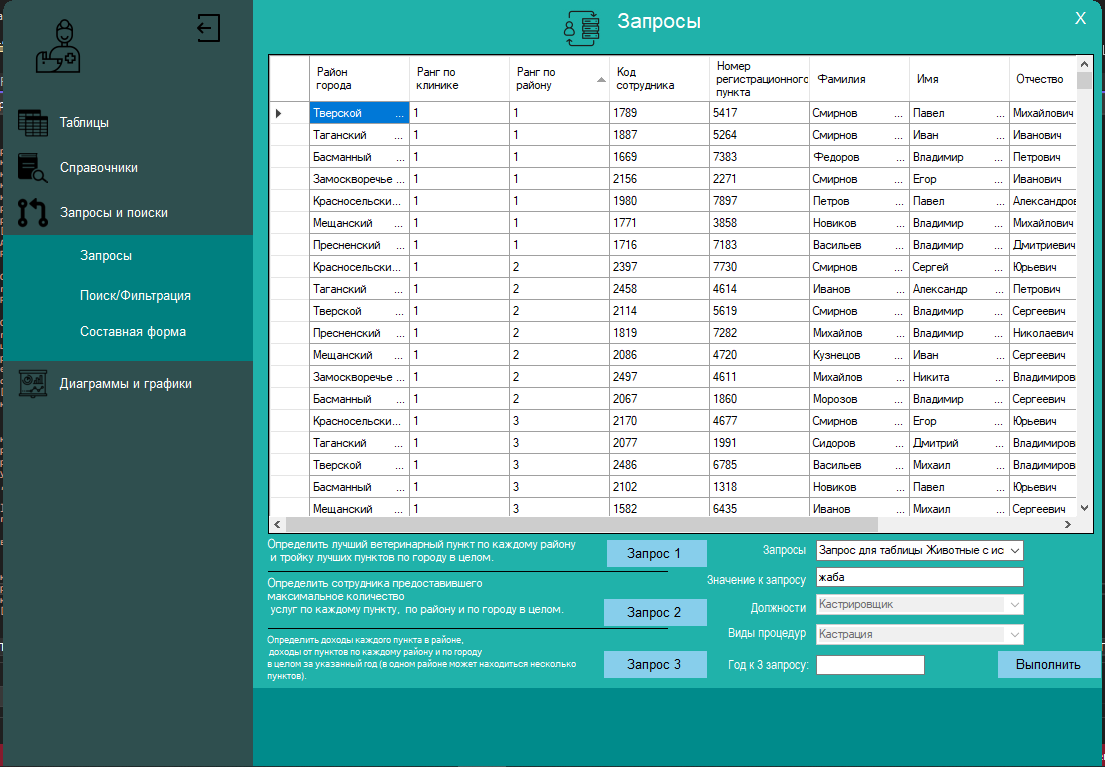


Рисунок 3.60 - Код запроса

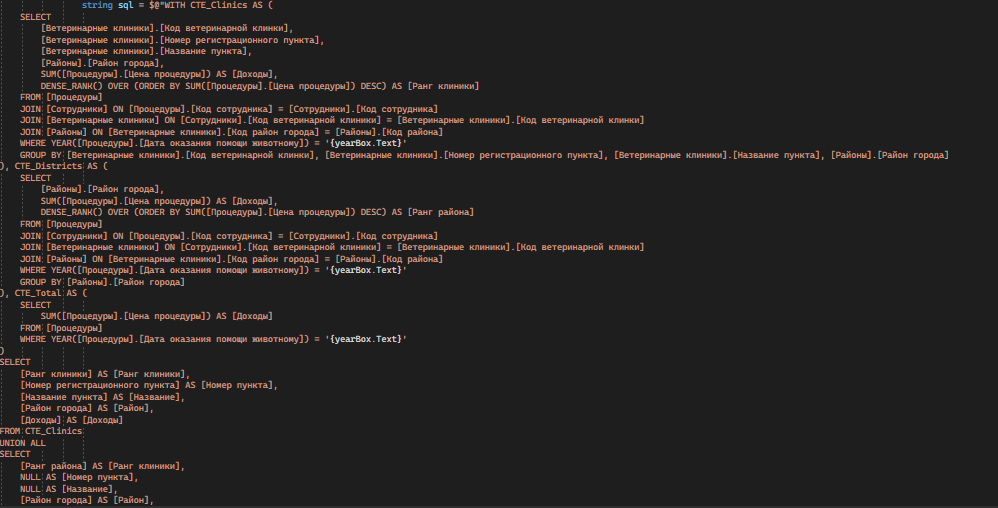


# Рисунок 3.61.1 - Результат выполнения запроса



# Рисунок 3.61.2 - Результат выполнения запроса

На рисунке 3.64-3.65 представлен запрос из курсовой работы – определить доходы каждого пункта в районе, доходы от пунктов по каждому району и по городу в целом за указанный год (в одном районе может находиться несколько пунктов)





# Рисунок 3.64 - Код запроса

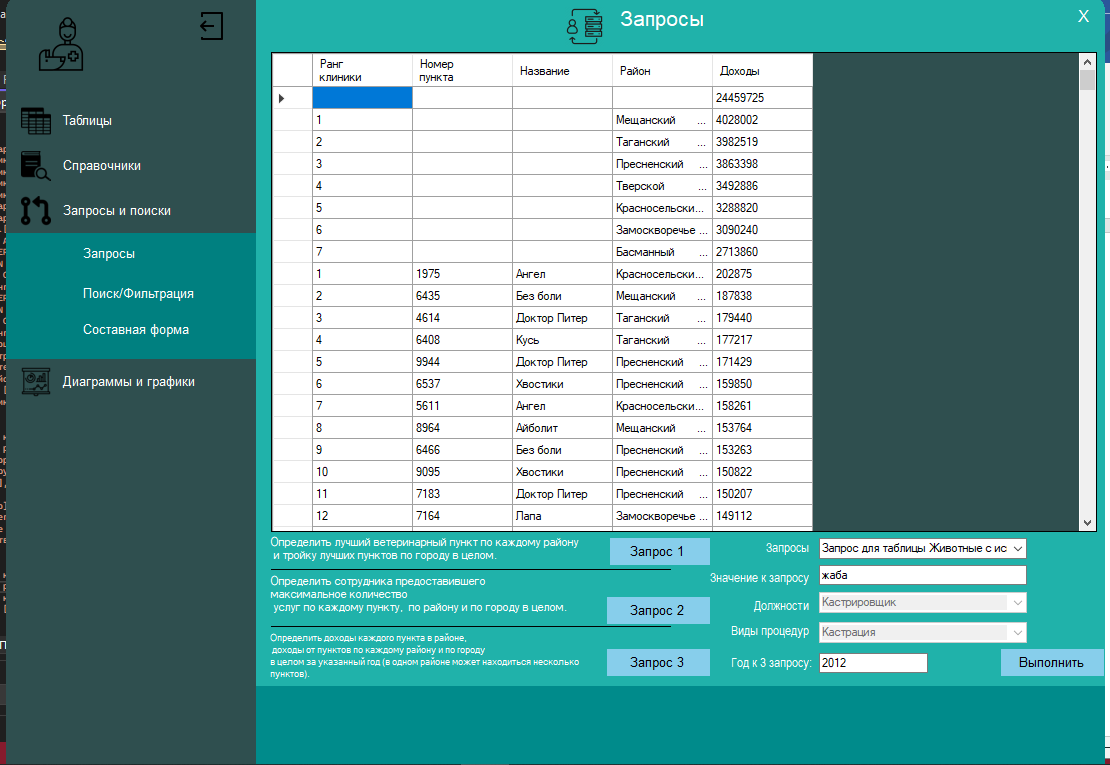


Рисунок 3.65 - Результат выполнения запроса

# Создание представлений и модифицированных представлений

Представление – виртуальная таблица, содержимое которой определяется запросом [1]

# Все запросы без параметров реализованы в виде представлений. Скрипты создания некоторых представления показано на рисунках 3.66-3.68

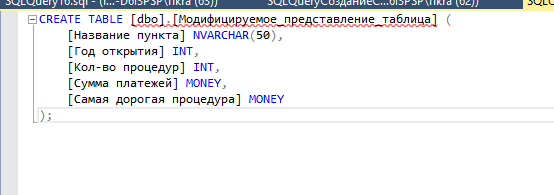
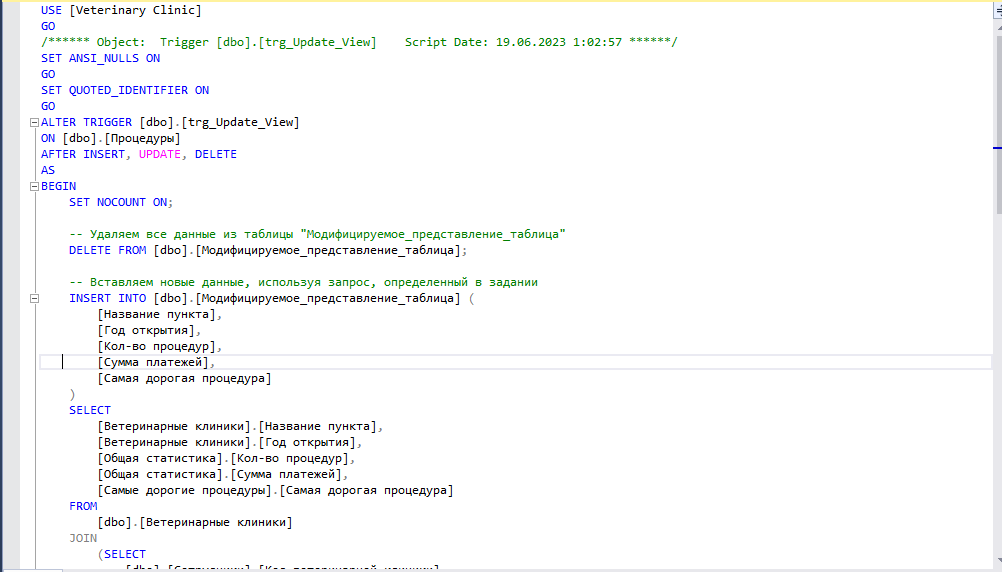
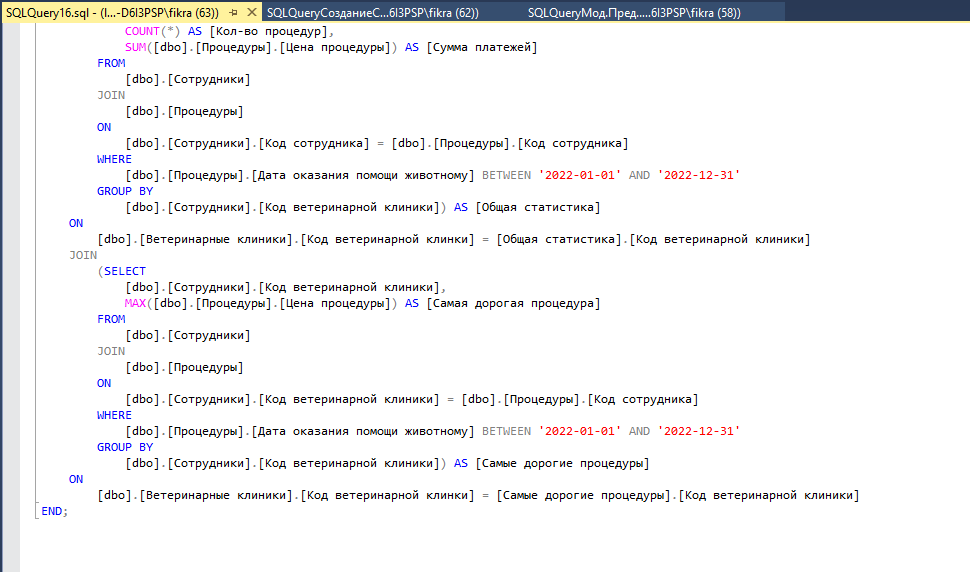


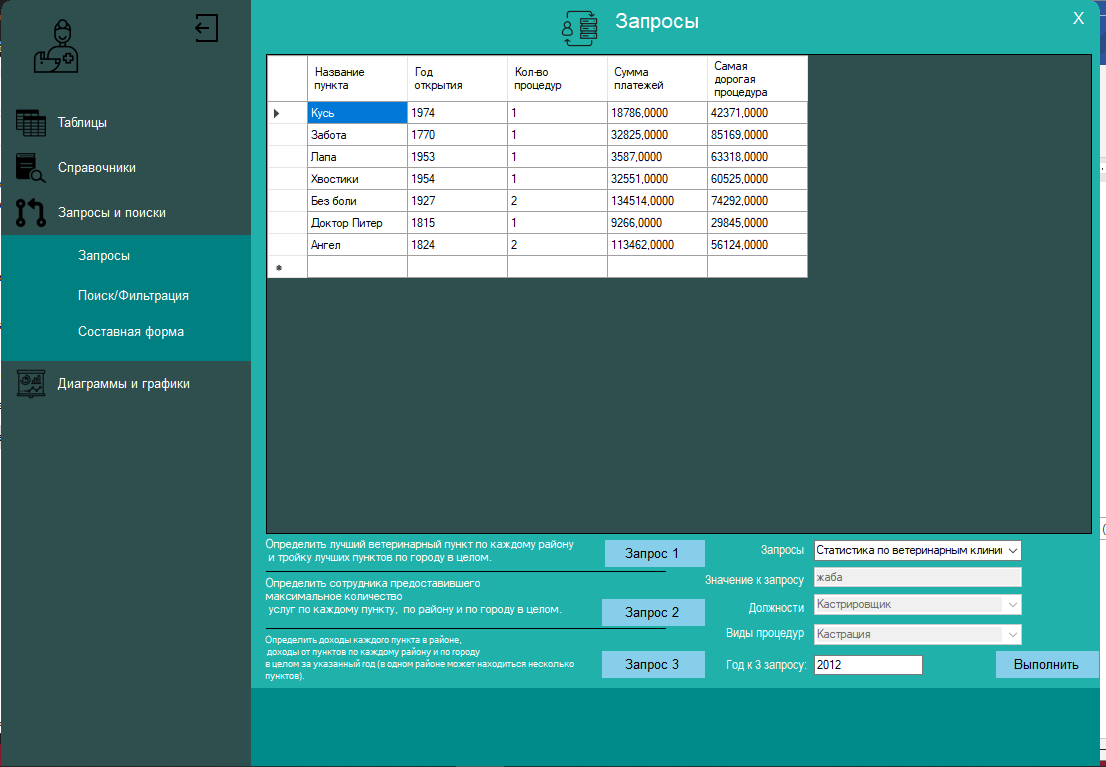
Рисунок 3.66 – Скрипт создания таблицы для мод. Представления



# Рисунок 3.67.1 – Скрипт представления Самых дорогих процедур



# Рисунок 3.67.2 – Скрипт представления Самых дорогих процедур



# Рисунок 3.68 – Результирующие данные таблицы

# Создание функций

Функция – набор операторов SQL, которые принимают входные данные и выполняют с ними действия, а затем возвращают результаты в виде выходных данных

# Все запросы с параметром реализованы в виду функций. Скрипты создания некоторых из них показаны на рисунках 3.73-3.74

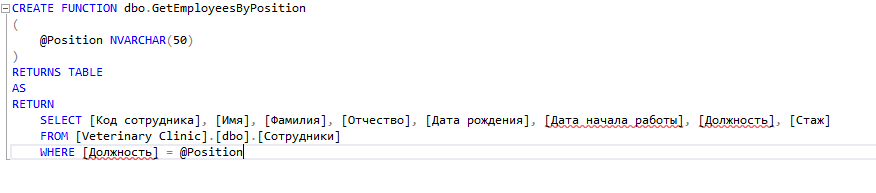
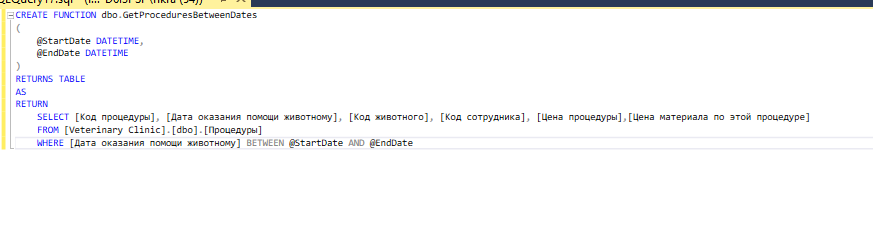


Рисунок 3.73 – Создания функции для запроса Сотрудник определённой должности



# Рисунок 3.74 – Создания функции для запроса Процедуры, сделанные в опред. дату

4.РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

# Клиентское приложение разрабатывается на языке C# с помощью встроенной графической библиотеки Windows Forms. Данная библиотека выбрана из-за простоты использования и функциональности, достаточной для написания клиентского приложения. Соединение базы данных и клиентского приложения произведено с помощью библиотеки Entity Framework 6.

# Формы и компоненты для работы с основными таблицами и справочниками

# Клиентское приложение представляет собой множество форм. Создана одна форма Таблица, в которую происходит подстановка данных для все таблиц. На данной форме расположены все элементы для взаимодействия с таблицей: кнопки добавления, удаления, изменения, фильтрации, сортировки и поиска. Доступны удаление по записи, группе записей, а также каскадное удаление. Форма и результаты сортировки, поиска и фильтрации представлены на рисунках 4.1-4.3

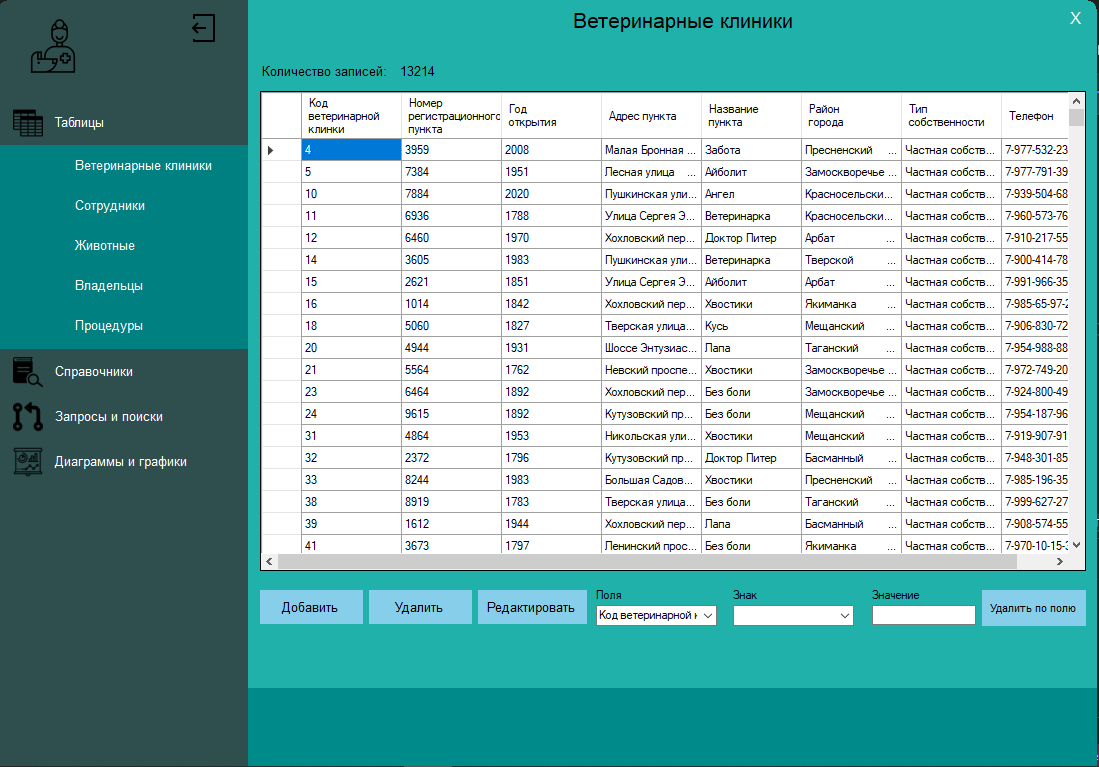
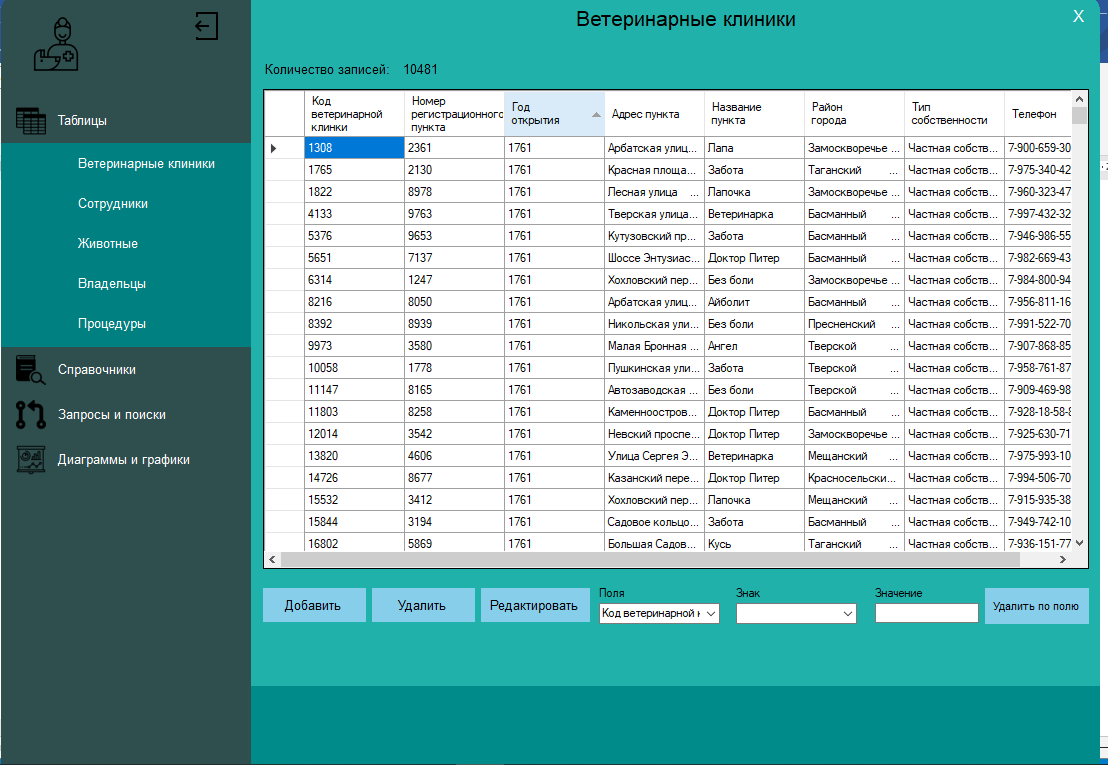


Рисунок 4.1 – Форма Таблица для таблицы Ветеринарные Клиники



# Рисунок 4.2 – Результат сортировки

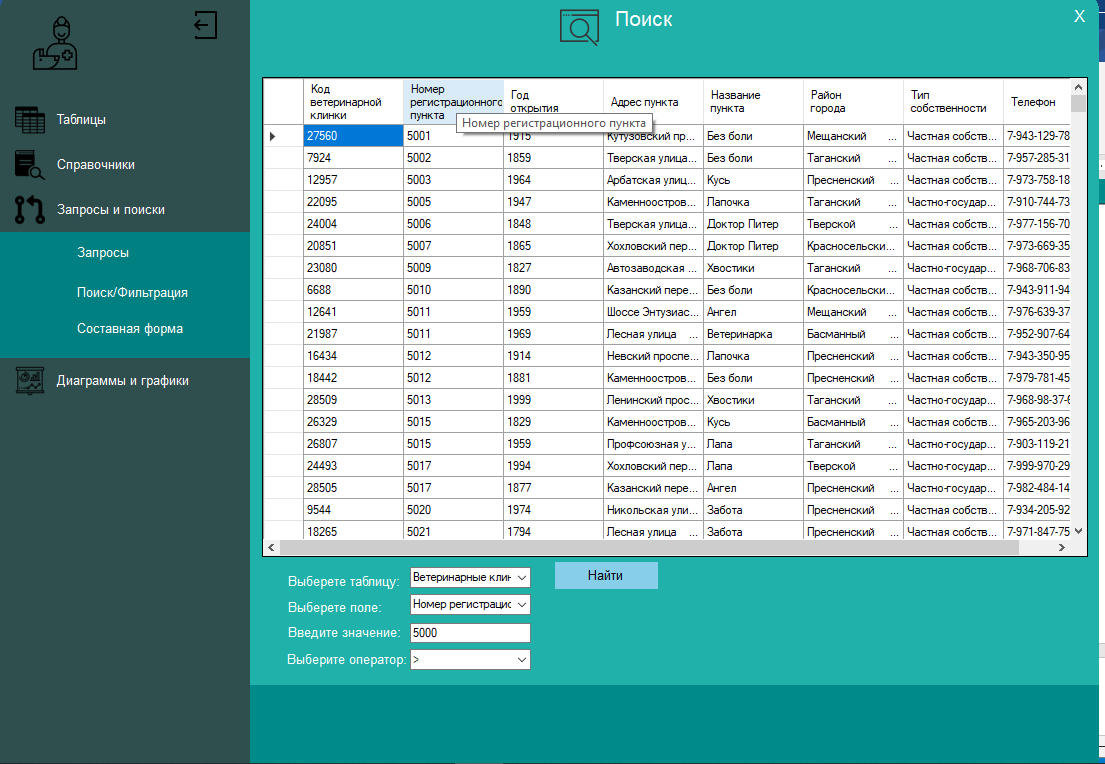
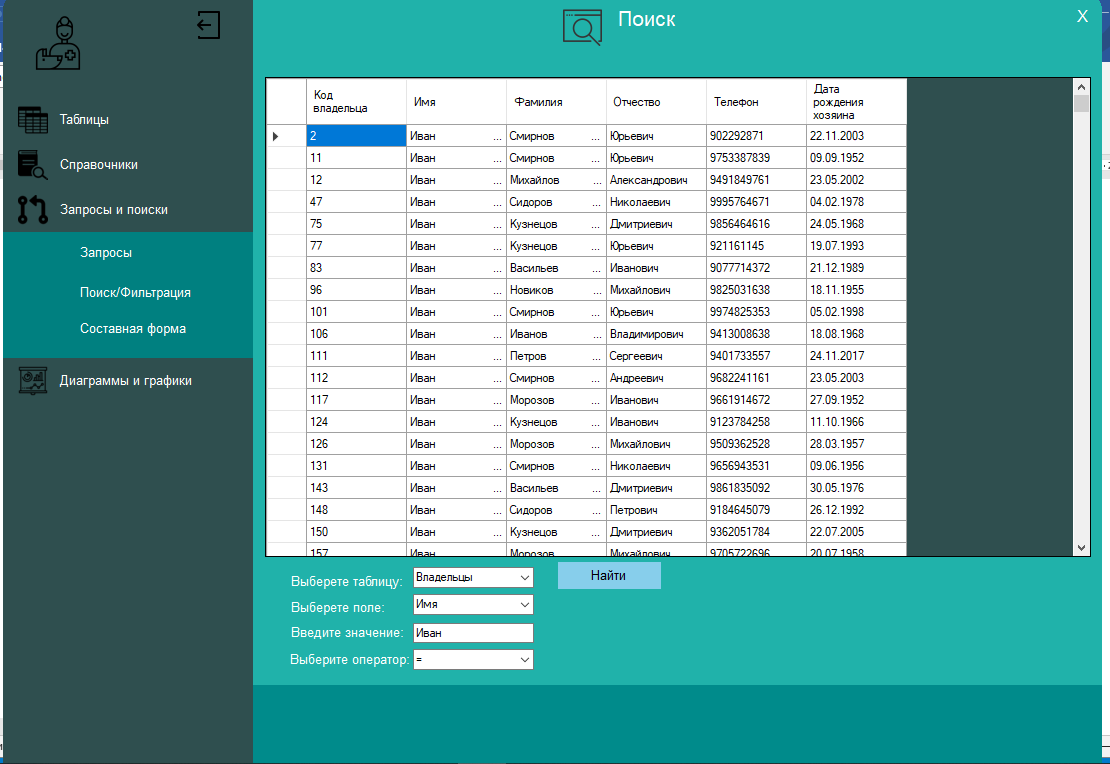
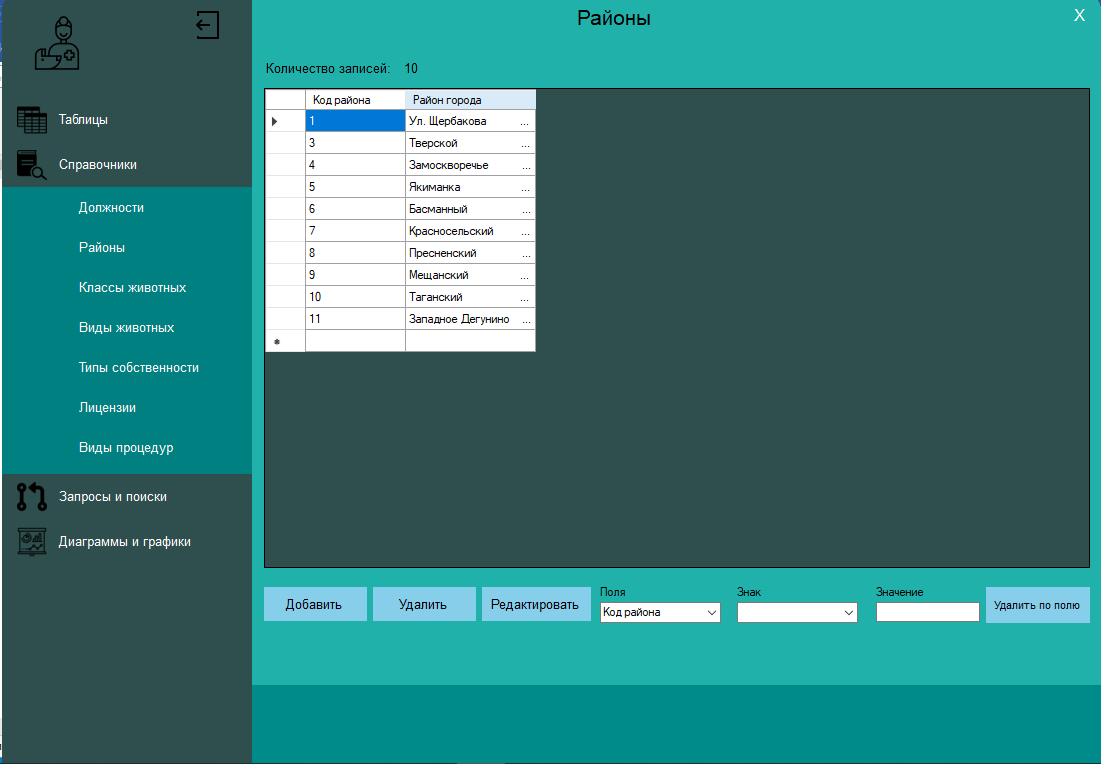


Рисунок 4.2 – Результат фильтрации



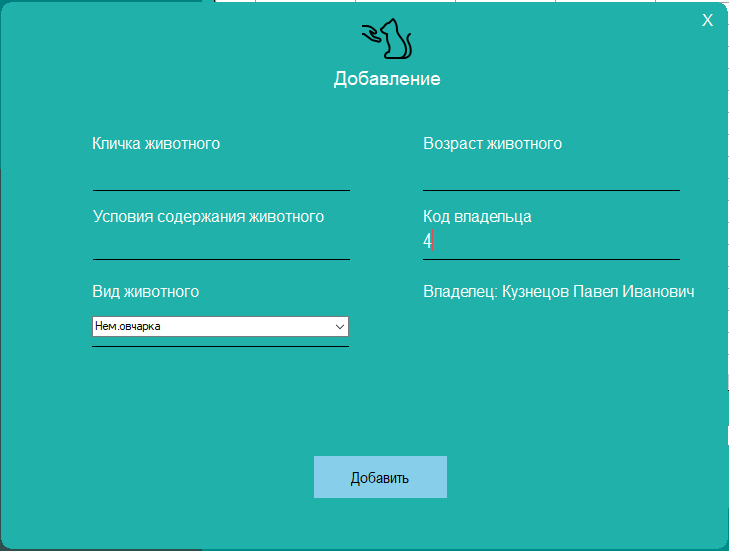
# Рисунок 4.3 – Результат поиска

Также в клиентском приложении создана одна форма для справочников, представленная на рисунке 4.3. Для справочников недоступны поиск, фильтрация и сортировка.



# Рисунок 4.4 – Форма Справочники для справочника Районы

Операции добавления и изменения происходят на отдельных формах, созданных для каждой таблицы, показанных на рисунках 4.5-4.6



# Рисунок 4.5 – Форма Добавление для таблицы Животные

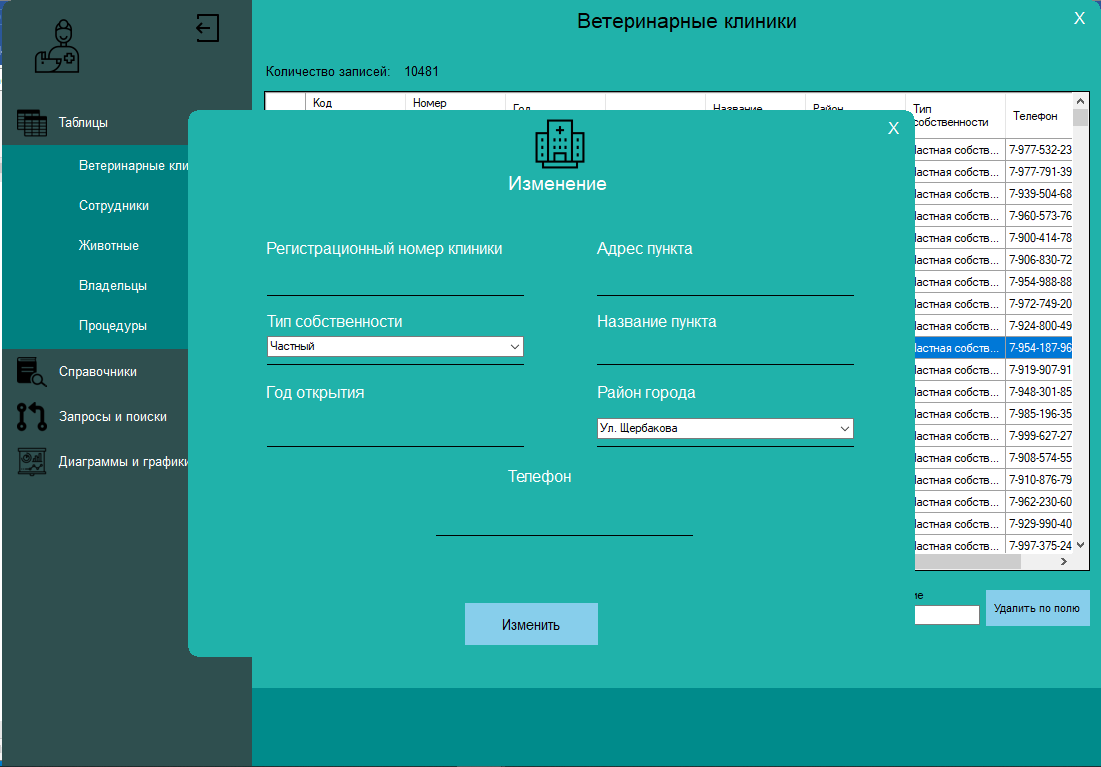


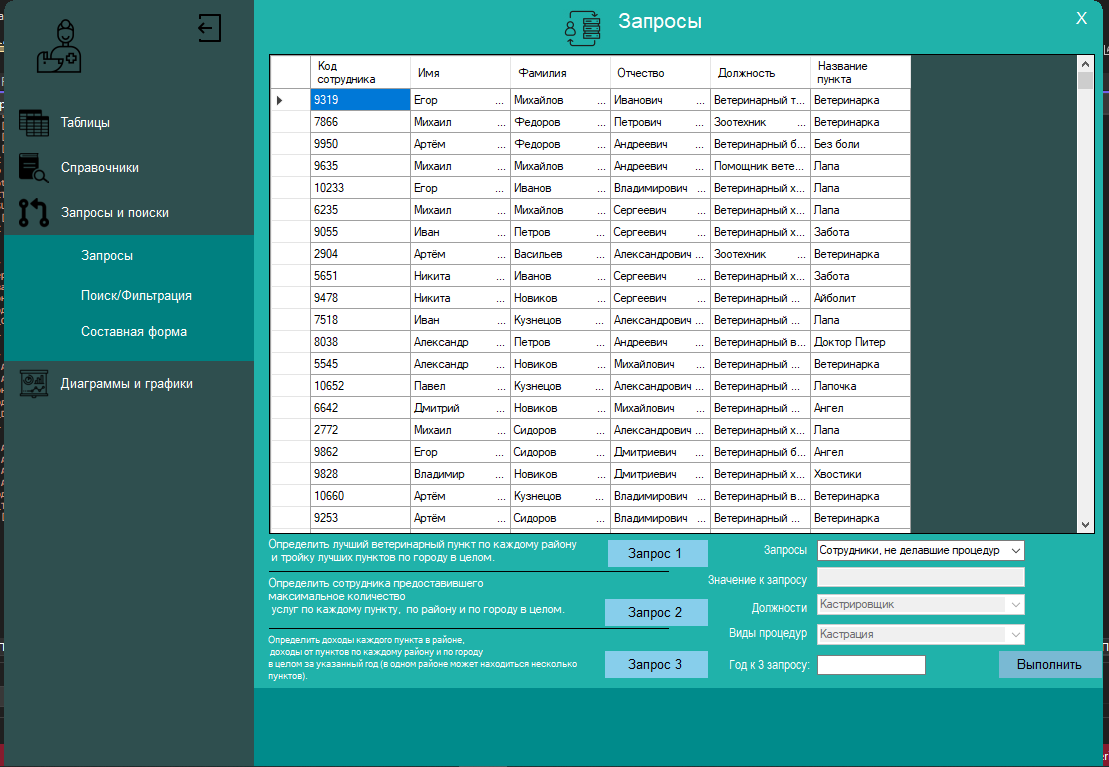
Рисунок 4.6 – Форма Изменение для таблицы Ветеринарные клиники

# Также клиентское приложение содержит составную форму, показывающую сотрудников, работающих в каждом саду, изображенную на рисунке 4.7.

Рисунок 4.7 – Составная форма

# Формы и компоненты для отображения результатов запросов

Для отображения результатов запросов создана форма - для параметрических и непараметрических запросов. В форме происходит подстановка данных, полученных в результате выполнения запроса. Для параметрических запросов присутствует поле, в которое необходимо ввести требуемые данные для выполнения запросы. Форма для параметрических и непараметрических запросов представлены на рисунке 4.8.



# Рисунок 4.8 – Форма для запросов

# Для запроса лучший ветеринарный пункт по каждому району и тройку лучших пунктов по городу в целом по нажатию на соответствующие кнопки доступны просмотр диаграммы и экспорт отчета в Excel, представленные на рисунках 4.9-4.11

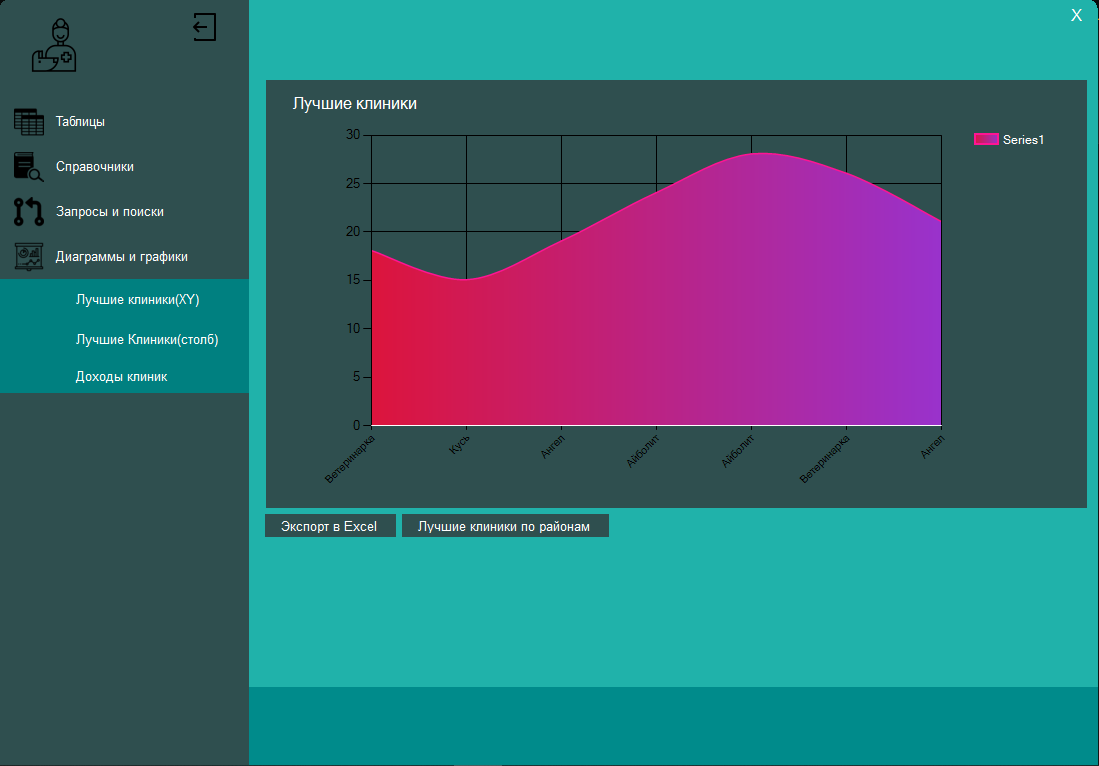


Рисунок 4.9 – Форма для диаграммы-график



# Рисунок 4.10 – Форма для диграммы-столбчатая

Рисунок 4.11 – Результат экспорта отчета в Excel

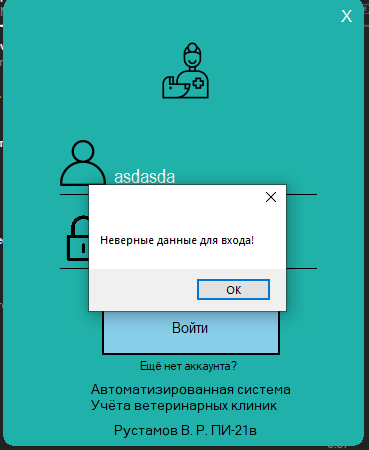
# ТЕСТИРОВАНИЕ РАЗРАБОТАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Для тестирования системы была разработана программа-генератор, выполняющая заполнение справочников и основных таблиц псевдоданными. Для корректной работы программы предварительное заполнение справочников не является необходимым условием.

# Для защиты от несанкционированного доступа перед запуском клиентского приложения пользователь должен авторизироваться. Если пользователь ввел неверные данные, то запрос на подключение отклоняется. На рисунках 5.1-5.2 показана форма авторизации и обработка некорректного ввода данных.

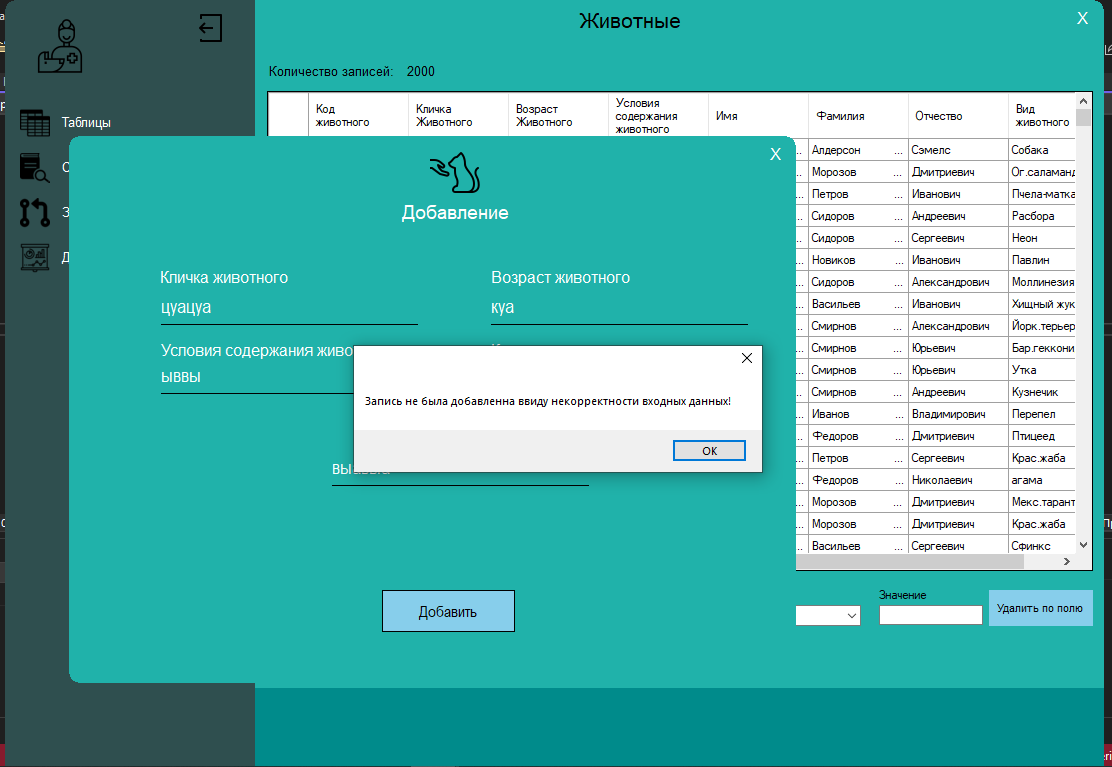
# 

Рисунок 5.1 – Форма Авторизации



# Рисунок 5.2 – Ошибка авторизации

Также в приложении предусмотрена проверка корректности введенных данных для каждой формы добавления и редактирования. Пример ошибки, отображаемой при вводе некорректных данных, представлен на рисунке 5.3



# Рисунок 5.3 – Ошибка обработки ввода

По результатам тестирования программы можно сказать, что она работоспособна.

# При завершении разработки и отладки программного продукта было написано краткое руководство пользователя, в котором описаны основные рекомендации по использованию программы. Руководство пользователя приведено в приложении А.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта были успешно достигнуты все поставленные цели. Основной задачей проекта было создание реляционной базы данных для ветеринарных клиник и соответствующего клиентского приложения для ее управления. В процессе разработки были получены и закреплены практические навыки проектирования и разработки баз данных. Была создана и заполнена база данных, а также разработано клиентское приложение для ее управления, включая запросы, возможность просмотра диаграмм и экспорта результатов в Excel.

В ходе выполнения проекта мы изучили язык SQL, систему управления баз данных MSSQL Server и соответствующую графическую оболочку. Также мы приобрели навыки работы с базами данных на языке C# и изучили библиотеку Windows Form для взаимодействия с базой данных.

Разработанное приложение имеет практическую ценность, поскольку его можно использовать для упрощения ведения учета ветеринарных клиник города.

# СПИСОК ЛИТЕРАТУРЫ

# Mark J. Price. C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development. М.: издательство "ДМК Пресс", 2020. – 816 с.

1. Jon Smith. Entity Framework Core in Action. М.: издательство "ДМК Пресс", 2019. – 520 с.

# Andrew Troelsen, Philip Japikse. Pro C# 7: With .NET and .NET Core. М.: издательство "Вильямс", 2017. - 1416 с.

1. William Assaf, Randolph West, Sven Aelterman. SQL Server 2019 Administration Inside Out. М.: издательство "Вильямс", 2020. - 944 с.

# ПРИЛОЖЕНИЕ А РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

А1. Назначение программы

# Программа предназначена для автоматизации, сокращение времени и повышения точности учета хозяйственной деятельности ботанических садов страны. Приложение позволяет добавлять, редактировать, изменять данные, производить сортировку, поиск и фильтрацию данных, просматривать результат выполнения запросов, сохранять его в Excel, просматривать диаграммы.

А2. Условия выполнения программы

# Для запуска программного продукта необходима операционная система MS Windows (версия не ниже XP). Также необходимо установить MS SQL Server версии не ниже 2017 и C# версии не ниже 7 Объем свободной памяти на жестком диске не менее 600 Мбайт. Процессор не менее производительный, чем Intel Pentium 4 с тактовой частотой 3.00 ГГц.

А3. Выполнение программы

# После запуска приложения открывается форма авторизации. Пользователю необходимо ввести верный логин и пароль, чтобы приступить к работе с приложением. В ином случае пользователю будет выведено сообщение о неудачной авторизации. Форма авторизации и сообщение об ошибке авторизации приведены на рисунках А1-А2.

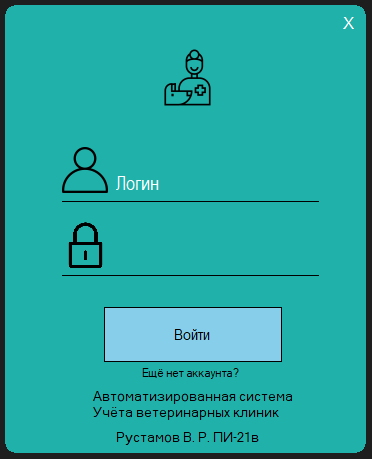
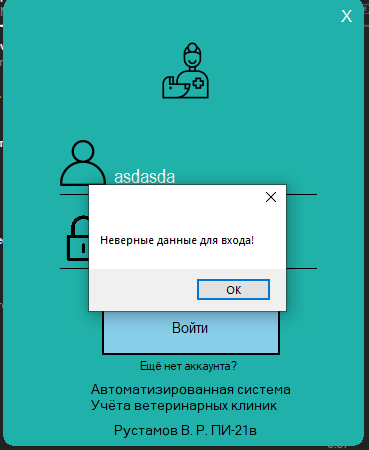
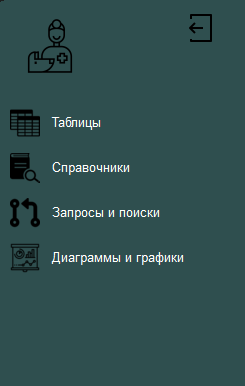


Рисунок А1– Форма авторизации



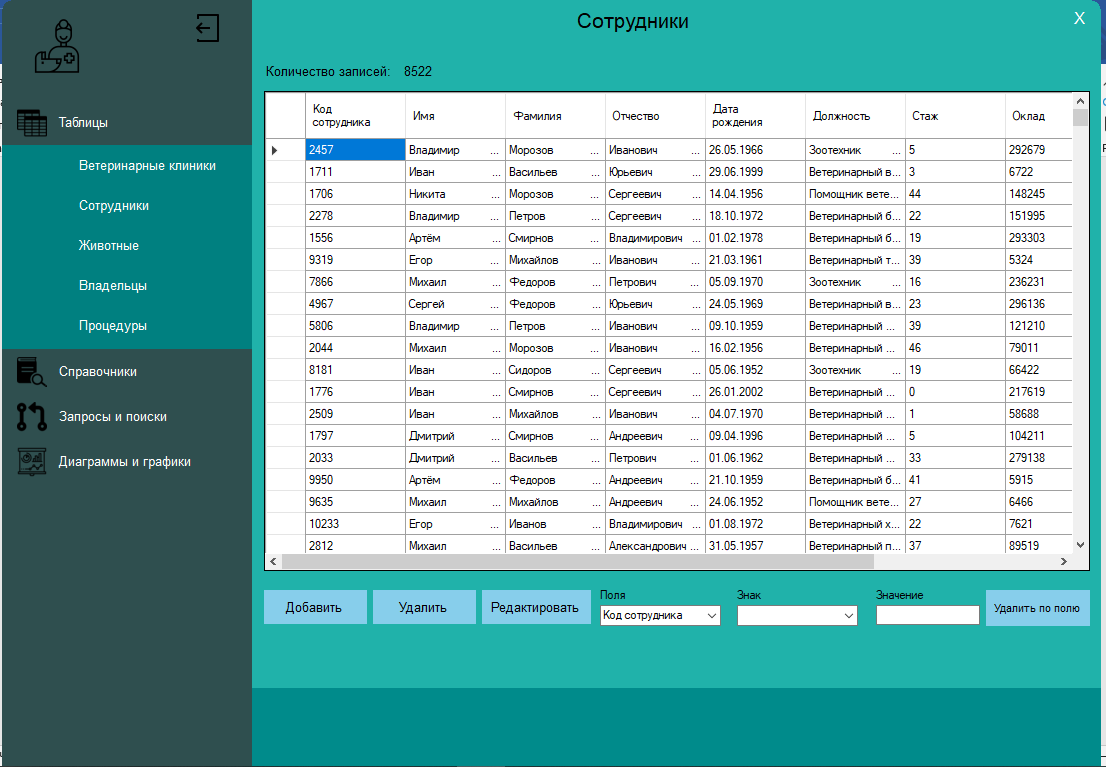
# Рисунок А2 – Ошибка авторизации

В случае успешной авторизации пользователю откроется главное меню, необходимое для навигации по приложению. Пользователь может выбрать работу с основными таблицами, справочниками, просмотр составной формы, работу с запросами, просмотр диаграмм. Главное меню представлено на рисунке А3.



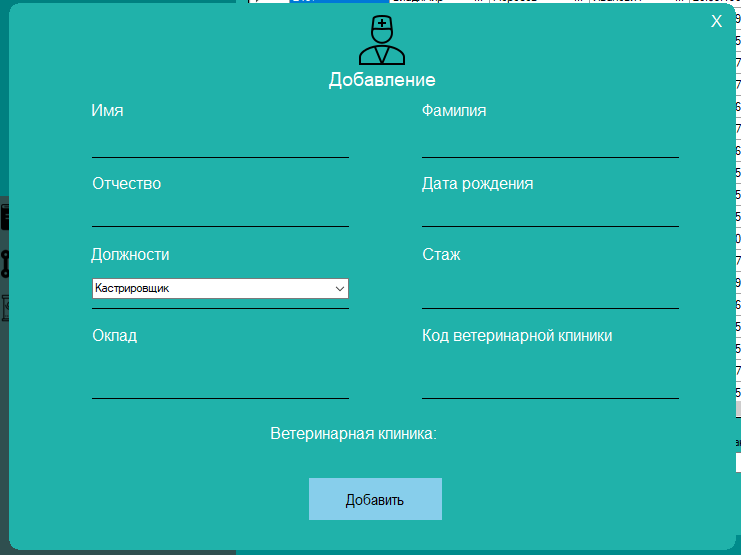
# Рисунок А3 – Главное меню

При нажатии на кнопку Таблицы пользователю будут представлены названия таблиц для дальнейшего выбора. Также происходит выбор справочника. После выбора таблицы для дальнейшей работы будет открыта форма просмотра таблицы, на которой доступны изменение, добавление, удаление данных, а также фильтрация, сортировка и поиск данных при нажатии на соответствующие кнопки. Для поиска, фильтрации и сортировки необходимо выбрать из выпадающего списка поле, для которого будут произведены данные действия. Для справочников возможность фильтрации, сортировки и поиска данных отсутствует. На рисунке А4 приведена форма просмотра таблицы.

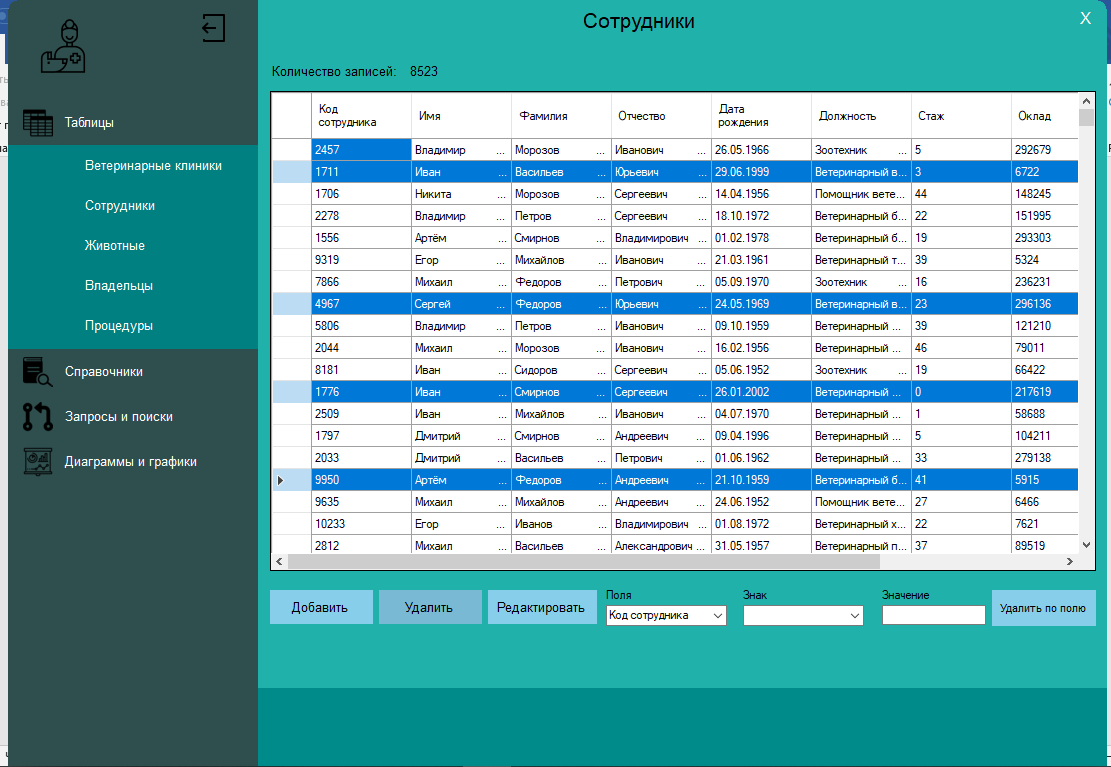


# Рисунок А4 – Форма Просмотра таблиц

Добавление и изменение записей происходит на отдельной форме. Для изменения необходимо кликнуть на запись и нажать кнопку Изменить. Пользователю необходимо заполнить соответствующие поля. В случае ввода некорректных данных будет выведена ошибка. Форма добавления показана на рисунке А5



# Рисунок А5 – Форма добавления

Для удаления пользователю необходимо выбрать запись и нажать кнопку Удалить. Для удаления подряд стоящих записей необходимо выбрать первую и последнюю запись с зажатой клавишей Shift. Для удаления группы не подряд стоящих записей необходимо выбирать записи с зажатой клавишей Ctrl. Выбор записей для удаления показан на рисунке А5.

# Рисунок А6 – Выбор записей для удаления по группе

Каскадное удаление доступно для всех таблиц.

# Для выполнения запросов необходимо перейти в раздел Запросы и выбрать требуемый запрос. После выбора запроса будет выведена форма просмотра запросов. Для выполнения параметрических запросов необходимо сначала указать требуемые данные. Пример выполнения параметрического запроса показан на рисунке А7.

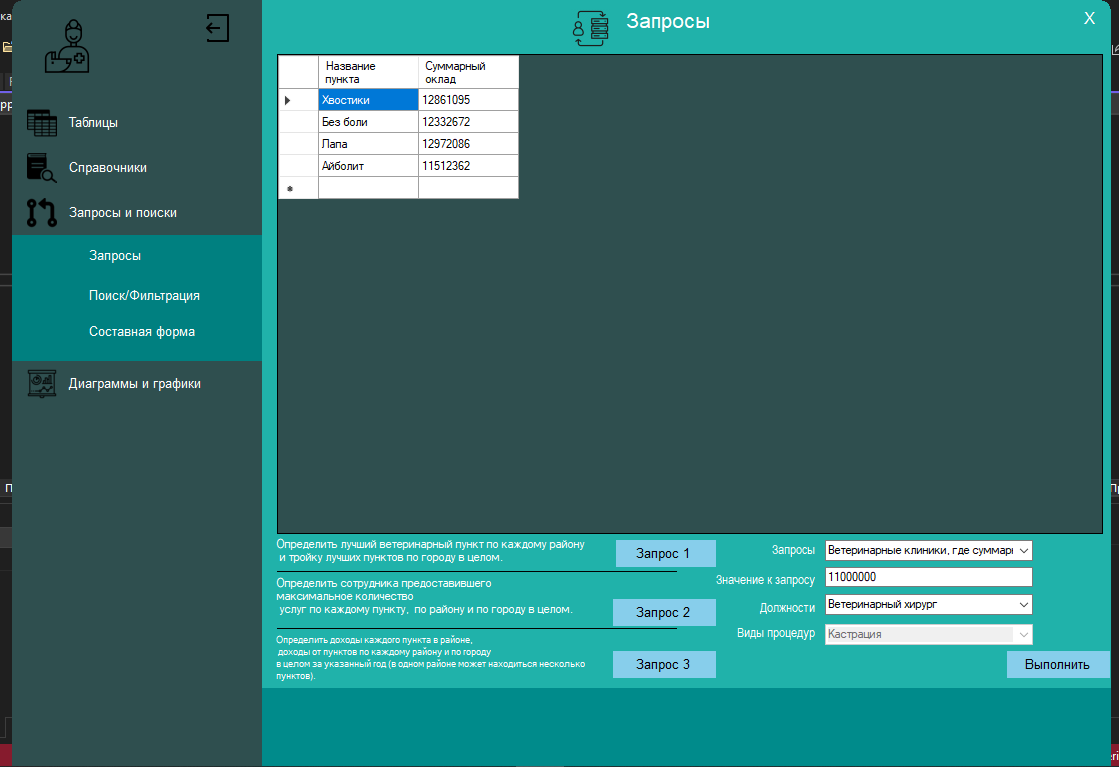


Рисунок А7 – Выполнение параметрического запроса

# Для построения диаграммы и экспорта отчета в Excel необходимо перейти в раздел Диаграммы на открывшейся форме нажать соответствующие кнопки. Форма Диаграммы представлена на рисунке А7.

Рисунок А7 – Форма Диаграммы

# ПРИЛОЖЕНИЕ Б

ЛИСТИНГ СЕРВЕРНОГО ПРИЛОЖЕНИЯ

sing System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Linq.Expressions;

using System.ServiceModel.Channels;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Documents;

using System.Windows.Forms;

using VeterenaryClinicApp.Controller;

using VeterenaryClinicApp.Model;

using VeterenaryClinicApp.View.AnimalClass;

using VeterenaryClinicApp.View.Animals;

using VeterenaryClinicApp.View.AnimalType;

using VeterenaryClinicApp.View.Conditions;

using VeterenaryClinicApp.View.District;

using VeterenaryClinicApp.View.Employees;

using VeterenaryClinicApp.View.Licenses;

using VeterenaryClinicApp.View.Owners;

using VeterenaryClinicApp.View.OwnershipTypes;

using VeterenaryClinicApp.View.Procedure;

using VeterenaryClinicApp.View.ProceduresType;

namespace VeterenaryClinicApp

{

public partial class TableForm : Form

{

public TableForm(DataTable data,string name)

{

InitializeComponent();

dataGridView1.DataSource = data;

nameTable.Text = name;

counter.Text = "" + (dataGridView1.RowCount - 1);

}

bool flag = false;

private void label1\_Click(object sender, EventArgs e)

{

this.Close();

}

private void label1\_MouseMove(object sender, MouseEventArgs e)

{

label1.ForeColor = Color.Red;

}

private void label1\_MouseLeave(object sender, EventArgs e)

{

label1.ForeColor = Color.White;

}

/// <summary>

/// Добавление записи

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button3\_Click(object sender, EventArgs e)

{

switch (nameTable.Text)

{

case "Ветеринарные клиники":

AddOwnerForm addForm = new AddOwnerForm();

addForm.ShowDialog();

while (!addForm.IsDisposed) { }

Refresh();

break;

case "Владельцы":

var addOwnerForm = new AddOwnerAnimalForm();

addOwnerForm.ShowDialog();

while (!addOwnerForm.IsDisposed) { }

Refresh();

break;

case "Процедуры":

var addProcedureForm = new AddProcedureForm();

addProcedureForm.ShowDialog();

while (!addProcedureForm.IsDisposed) { }

Refresh();

break;

case "Сотрудники":

var addEmployeeForm = new AddEmployeeForm();

addEmployeeForm.ShowDialog();

while (!addEmployeeForm.IsDisposed) { }

Refresh();

break;

case "Животные":

var addAnimalForm = new AddAnimalForm();

addAnimalForm.ShowDialog();

while (!addAnimalForm.IsDisposed) { }

Refresh();

break;

case "Виды процедуры":

var addProcedureTypeForm = new AddProcedureTypeForm();

addProcedureTypeForm.ShowDialog();

while (!addProcedureTypeForm.IsDisposed) { }

Refresh();

break;

case "Должности":

var addPositionForm = new AddPositionForm();

addPositionForm.ShowDialog();

while (!addPositionForm.IsDisposed) { }

Refresh();

break;

case "Классы животных":

var addAnimalClassForm = new AddAnimalClassForm();

addAnimalClassForm.ShowDialog();

while (!addAnimalClassForm.IsDisposed) { }

Refresh();

break;

case "Виды животных":

var addAnimalTypeForm = new AddAnimalTypeForm();

addAnimalTypeForm.ShowDialog();

while (!addAnimalTypeForm.IsDisposed) { }

Refresh();

break;

case "Районы":

var addDistrictForm = new AddDistrictForm();

addDistrictForm.ShowDialog();

while (!addDistrictForm.IsDisposed) { }

Refresh();

break;

case "Тип собственности":

var addOwnershipForm = new AddOwnershipTypeForm();

addOwnershipForm.ShowDialog();

while (!addOwnershipForm.IsDisposed) { }

Refresh();

break;

case "Лицензии":

var addLicenseForm = new AddLicenseForm();

addLicenseForm.ShowDialog();

while (!addLicenseForm.IsDisposed) { }

Refresh();

break;

}

}

/// <summary>

/// Удаление записи

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button1\_Click(object sender, EventArgs e)

{

try

{

flag = false;

var selectedRows = dataGridView1.SelectedRows;

List<int> ids = new List<int>();

switch (nameTable.Text)

{

case "Ветеринарные клиники":

for (var i = 0; i < selectedRows.Count; i++)

{

ids.Add((int)selectedRows[i].Cells["Код ветеринарной клинки"].Value);

}

(int, int) countRemoveVetClinic = Remover.CountRemoveVeterinaryClinic(ids);

DialogResult resultVetClinic = MessageBox.Show($"Будут удалены также записи из следующих таблиц:\nСотрудники - {countRemoveVetClinic.Item1}" +

$"\nЛицензии - {countRemoveVetClinic.Item2}\nПродолжить?", "Каскадное удаление", MessageBoxButtons.YesNo);

if (resultVetClinic == DialogResult.No)

{

return;

}

Remover.RemoveVeterinaryClinic(ids);

Refresh();

break;

case "Владельцы":

for (var i = 0; i < selectedRows.Count; i++)

{

ids.Add((int)selectedRows[i].Cells["Код владельца"].Value);

}

int countRemoveOwner = Remover.CountRemoveOwner(ids);

DialogResult resultOwner = MessageBox.Show($"Будут удалены также записи из следующих таблиц:\nЖивотные - {countRemoveOwner}" +

$"\nПродолжить?", "Каскадное удаление", MessageBoxButtons.YesNo);

if (resultOwner == DialogResult.No)

{

return;

}

Remover.RemoveOwner(ids);

Refresh();

break;

case "Процедуры":

for (var i = 0; i < selectedRows.Count; i++)

{

ids.Add((int)selectedRows[i].Cells["Код процедуры"].Value);

}

Remover.RemoveProcedure(ids);

Refresh();

break;

case "Сотрудники":

for (var i = 0; i < selectedRows.Count; i++)

{

ids.Add((int)selectedRows[i].Cells["Код сотрудника"].Value);

}

int countRemoveEmployee = Remover.CountRemoveEmployee(ids);

DialogResult resultEmployee = MessageBox.Show($"Будут удалены также записи из следующих таблиц:\nПроцедуры - {countRemoveEmployee}" +

$"\nПродолжить?", "Каскадное удаление", MessageBoxButtons.YesNo);

if (resultEmployee == DialogResult.No)

{

return;

}

Remover.RemoveEmployee(ids);

Refresh();

break;

case "Животные":

for (var i = 0; i < selectedRows.Count; i++)

{

ids.Add((int)selectedRows[i].Cells["Код животного"].Value);

}

int countRemoveAnimal = Remover.CountRemoveAnimal(ids);

DialogResult resultAnimal = MessageBox.Show($"Будут удалены также записи из следующих таблиц:\nПроцедуры - {countRemoveAnimal}" +

$"\nПродолжить?", "Каскадное удаление", MessageBoxButtons.YesNo);

if (resultAnimal == DialogResult.No)

{

return;

}

Remover.RemoveAnimal(ids);

Refresh();

break;

case "Виды процедуры":

for (var i = 0; i < selectedRows.Count; i++)

{

ids.Add((int)selectedRows[i].Cells["Код вида процедуры"].Value);

}

int countRemoveProcedureType = Remover.CountRemoveProcedureType(ids);

DialogResult resultProcedureType = MessageBox.Show($"Будут удалены также записи из следующих таблиц:\nПроцедуры - {countRemoveProcedureType}" +

$"\nПродолжить?", "Каскадное удаление", MessageBoxButtons.YesNo);

if (resultProcedureType == DialogResult.No)

{

return;

}

Remover.RemoveProcedureType(ids);

Refresh();

break;

case "Должности":

for (var i = 0; i < selectedRows.Count; i++)

{

ids.Add((int)selectedRows[i].Cells["Код должности"].Value);

}

(int, int) countRemovePosition = Remover.CountRemovePosition(ids);

DialogResult resultPosition = MessageBox.Show($"Будут удалены также записи из следующих таблиц:\nСотрудники - {countRemovePosition.Item1}" +

$"\nПроцедуры - {countRemovePosition.Item2}\nПродолжить?", "Каскадное удаление", MessageBoxButtons.YesNo);

if (resultPosition == DialogResult.No)

{

return;

}

Remover.RemovePosition(ids);

Refresh();

break;

case "Классы животных":

for (var i = 0; i < selectedRows.Count; i++)

{

ids.Add((int)selectedRows[i].Cells["Код класса животного"].Value);

}

(int, int, int) countRemoveAnimalClass= Remover.CountRemoveAnimalClass(ids);

DialogResult resultAnimalClass = MessageBox.Show($"Будут удалены также записи из следующих таблиц:\nВиды животных - {countRemoveAnimalClass.Item1}\n" +

$"Животные - {countRemoveAnimalClass.Item2}" +

$"\nПроцедуры - {countRemoveAnimalClass.Item3}" +

$"\nПродолжить?", "Каскадное удаление", MessageBoxButtons.YesNo);

if (resultAnimalClass == DialogResult.No)

{

return;

}

Remover.RemoveAnimalClass(ids);

Refresh();

break;

case "Виды животных":

for (var i = 0; i < selectedRows.Count; i++)

{

ids.Add((int)selectedRows[i].Cells["Код вида животного"].Value);

}

(int, int) countRemoveAnimalType = Remover.CountRemoveAnimalType(ids);

DialogResult resultAnimalType = MessageBox.Show($"Будут удалены также записи из следующих таблиц:\nЖивотные - {countRemoveAnimalType.Item1}" +

$"\nПроцедуры - {countRemoveAnimalType.Item2}\nПродолжить?", "Каскадное удаление", MessageBoxButtons.YesNo);

if (resultAnimalType == DialogResult.No)

{

return;

}

Remover.RemoveAnimalType(ids);

Refresh();

break;

case "Районы":

for (var i = 0; i < selectedRows.Count; i++)

{

ids.Add((int)selectedRows[i].Cells["Код района"].Value);

}

(int, int, int) countRemoveDistrict = Remover.CountRemoveDistrict(ids);

DialogResult resultDistrict = MessageBox.Show($"Будут удалены также записи из следующих таблиц:\nВетеринарные клиники - {countRemoveDistrict.Item1}\n" +

$"Сотрудники - {countRemoveDistrict.Item2}" +

$"\nЛицензии - {countRemoveDistrict.Item3}" +

$"\nПродолжить?", "Каскадное удаление", MessageBoxButtons.YesNo);

if (resultDistrict == DialogResult.No)

{

return;

}

Remover.RemoveDistrict(ids);

Refresh();

break;

case "Тип собственности":

for (var i = 0; i < selectedRows.Count; i++)

{

ids.Add((int)selectedRows[i].Cells["Код типа собственности"].Value);

}

(int, int, int) countRemoveOwnershipType = Remover.CountRemoveOwnershipType(ids);

DialogResult resultOwnershipType = MessageBox.Show($"Будут удалены также записи из следующих таблиц:\nВетеринарные клиники - {countRemoveOwnershipType.Item1}" +

$"\nСотрудники - {countRemoveOwnershipType.Item2}\n" +

$"Лицензии - {countRemoveOwnershipType.Item3}" +

$"\nПродолжить?", "Каскадное удаление", MessageBoxButtons.YesNo);

if (resultOwnershipType == DialogResult.No)

{

return;

}

Remover.RemoveOwnership(ids);

Refresh();

break;

case "Лицензии":

for (var i = 0; i < selectedRows.Count; i++)

{

ids.Add((int)selectedRows[i].Cells["Код лицензии"].Value);

}

Remover.RemoveLicense(ids);

Refresh();

break;

default:

break;

}

}

catch(Exception ex)

{

MessageBox.Show($"{ex.Message}\nВероятно вы не правильно выделили запись для удаление, попробуйте ещё раз.");

}

}

/// <summary>

/// Изменение записи

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button2\_Click(object sender, EventArgs e)

{

try

{

switch (nameTable.Text)

{

case "Ветеринарные клиники":

var selectedRow = dataGridView1.SelectedRows[0];

EditForm editForm = new EditForm(selectedRow);

editForm.ShowDialog();

while (!editForm.IsDisposed) { }

Refresh();

break;

case "Владельцы":

var selectedOwnerRow = dataGridView1.SelectedRows[0];

EditOwnerForm editOwnerForm = new EditOwnerForm(selectedOwnerRow);

editOwnerForm.ShowDialog();

while (!editOwnerForm.IsDisposed) { }

Refresh();

break;

case "Процедуры":

var selectedProcedureRow = dataGridView1.SelectedRows[0];

EditProcedureForm editProcedureForm = new EditProcedureForm(selectedProcedureRow);

editProcedureForm.ShowDialog();

while (!editProcedureForm.IsDisposed) { }

Refresh();

break;

case "Сотрудники":

var selectedEmployeeRow = dataGridView1.SelectedRows[0];

EditEmployeeForm editEmployeeForm = new EditEmployeeForm(selectedEmployeeRow);

editEmployeeForm.ShowDialog();

while (!editEmployeeForm.IsDisposed) { }

Refresh();

break;

case "Животные":

var selectedAnimalRow = dataGridView1.SelectedRows[0];

EditAnimalForm editAnimalForm = new EditAnimalForm(selectedAnimalRow);

editAnimalForm.ShowDialog();

while (!editAnimalForm.IsDisposed) { }

Refresh();

break;

case "Виды процедуры":

var selectedProcedureTypeRow = dataGridView1.SelectedRows[0];

EditProcedureTypeForm editProcedureTypeForm = new EditProcedureTypeForm(selectedProcedureTypeRow);

editProcedureTypeForm.ShowDialog();

while (!editProcedureTypeForm.IsDisposed) { }

Refresh();

break;

case "Должности":

var selectedPositionRow = dataGridView1.SelectedRows[0];

EditPositionForm editConditionForm = new EditPositionForm(selectedPositionRow);

editConditionForm.ShowDialog();

while (!editConditionForm.IsDisposed) { }

Refresh();

break;

case "Классы животных":

var selectedAnimalClassRow = dataGridView1.SelectedRows[0];

EditAnimalClassForm editAnimalClassForm = new EditAnimalClassForm(selectedAnimalClassRow);

editAnimalClassForm.ShowDialog();

while (!editAnimalClassForm.IsDisposed) { }

Refresh();

break;

case "Виды животных":

var selectedAnimalTypeRow = dataGridView1.SelectedRows[0];

EditAnimalTypeForm editAnimalTypeForm = new EditAnimalTypeForm(selectedAnimalTypeRow);

editAnimalTypeForm.ShowDialog();

while (!editAnimalTypeForm.IsDisposed) { }

Refresh();

break;

case "Районы":

var selectedDistrictRow = dataGridView1.SelectedRows[0];

EditDistrictForm editDistrictForm = new EditDistrictForm(selectedDistrictRow);

editDistrictForm.ShowDialog();

while (!editDistrictForm.IsDisposed) { }

Refresh();

break;

case "Тип собственности":

var selectedOwnershipRow = dataGridView1.SelectedRows[0];

EditOwnershipTypeForm editOwnershipForm = new EditOwnershipTypeForm(selectedOwnershipRow);

editOwnershipForm.ShowDialog();

while (!editOwnershipForm.IsDisposed) { }

Refresh();

break;

case "Лицензии":

var selectedLicenseRow = dataGridView1.SelectedRows[0];

EditLicenseForm editLicenseForm = new EditLicenseForm(selectedLicenseRow);

editLicenseForm.ShowDialog();

while (!editLicenseForm.IsDisposed) { }

Refresh();

break;

default:

break;

}

}

catch(Exception ex)

{

MessageBox.Show($"{ex.Message}");

}

}

private void TableForm\_Load(object sender, EventArgs e)

{

labelTemp.Text = nameTable.Text;

string sql;

List<string> columnName = new List<string>();

if (labelTemp.Text != "System.Data.DataRowView" && !string.IsNullOrWhiteSpace(labelTemp.Text))

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

using (var connection = new SqlConnection(connectionString))

{

connection.Open();

if (labelTemp.Text == "Ветеринарные клиники")

{

sql = "SELECT [Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код ветеринарной клинки]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Номер регистрационного пункта]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Год открытия]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Адрес пункта]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Название пункта]," +

"[Veterinary Clinic].[dbo].[Районы].[Район города]," +

"[Veterinary Clinic].[dbo].[Тип собственности].[Тип собственности]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Телефон] FROM [Veterinary Clinic].[dbo].[Ветеринарные клиники]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Тип собственности] ON ([Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код типа собственности]) = [Veterinary Clinic].[dbo].[Тип собственности].[Код типа собственности]" +

$"INNER JOIN [Veterinary Clinic].[dbo].[Районы] ON ([Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код район города]) = [Veterinary Clinic].[dbo].[Районы].[Код района]";

}

else if (labelTemp.Text == "Сотрудники")

{

sql = "SELECT [Veterinary Clinic].[dbo].[Сотрудники].[Код сотрудника]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Имя]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Фамилия]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Отчество]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Дата рождения]," +

"[Veterinary Clinic].[dbo].[Должности].[Должность]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Стаж]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Оклад]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Номер регистрационного пункта]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Название пункта] FROM [Veterinary Clinic].[dbo].[Сотрудники]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Ветеринарные клиники] ON ([Veterinary Clinic].[dbo].[Сотрудники].[Код ветеринарной клиники]) = [Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код ветеринарной клинки]" +

$"INNER JOIN [Veterinary Clinic].[dbo].[Должности] ON ([Veterinary Clinic].[dbo].[Сотрудники].[Код должности]) = [Veterinary Clinic].[dbo].[Должности].[Код должности]";

}

else if (labelTemp.Text == "Животные")

{

sql = "SELECT [Veterinary Clinic].[dbo].[Животные].[Код животного]," +

"[Veterinary Clinic].[dbo].[Животные].[Кличка Животного]," +

"[Veterinary Clinic].[dbo].[Животные].[Возраст Животного]," +

"[Veterinary Clinic].[dbo].[Животные].[Условия содержания животного]," +

"[Veterinary Clinic].[dbo].[Владельцы].[Имя]," +

"[Veterinary Clinic].[dbo].[Владельцы].[Фамилия]," +

"[Veterinary Clinic].[dbo].[Владельцы].[Отчество]," +

"[Veterinary Clinic].[dbo].[Виды животных].[Вид животного] FROM [Veterinary Clinic].[dbo].[Животные]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Владельцы] ON ([Veterinary Clinic].[dbo].[Животные].[Код владельца]) = [Veterinary Clinic].[dbo].[Владельцы].[Код владельца] " +

$"INNER JOIN [Veterinary Clinic].[dbo].[Виды животных] ON ([Veterinary Clinic].[dbo].[Животные].[Код вида животного]) = [Veterinary Clinic].[dbo].[Виды животных].[Код вида животного]";

}

else if (labelTemp.Text == "Процедуры")

{

sql = "SELECT [Veterinary Clinic].[dbo].[Процедуры].[Код процедуры]," +

"[Veterinary Clinic].[dbo].[Процедуры].[Дата оказания помощи животному]," +

"[Veterinary Clinic].[dbo].[Процедуры].[Цена процедуры]," +

"[Veterinary Clinic].[dbo].[Процедуры].[Скидка на эту процедуру]," +

"[Veterinary Clinic].[dbo].[Процедуры].[Цена материала по этой процедуре]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Имя]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Фамилия]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Отчество]," +

"[Veterinary Clinic].[dbo].[Животные].[Кличка животного]," +

"[Veterinary Clinic].[dbo].[Виды животных].[Вид животного]," +

"[Veterinary Clinic].[dbo].[Виды процедуры].[Вид процедуры] FROM [Veterinary Clinic].[dbo].[Процедуры]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Сотрудники] ON ([Veterinary Clinic].[dbo].[Процедуры].[Код сотрудника]) = [Veterinary Clinic].[dbo].[Сотрудники].[Код сотрудника]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Животные] ON ([Veterinary Clinic].[dbo].[Процедуры].[Код животного]) = [Veterinary Clinic].[dbo].[Животные].[Код животного] " +

"INNER JOIN [Veterinary Clinic].[dbo].[Виды животных] ON ([Veterinary Clinic].[dbo].[Животные].[Код вида животного]) = [Veterinary Clinic].[dbo].[Виды животных].[Код вида животного] " +

$"INNER JOIN [Veterinary Clinic].[dbo].[Виды процедуры] ON ([Veterinary Clinic].[dbo].[Процедуры].[Код вида процедуры]) = [Veterinary Clinic].[dbo].[Виды процедуры].[Код вида процедуры]";

}

else if (labelTemp.Text == "Виды животных")

{

sql = "SELECT [Veterinary Clinic].[dbo].[Виды животных].[Код вида животного]," +

"[Veterinary Clinic].[dbo].[Виды животных].[Вид животного]," +

"[Veterinary Clinic].[dbo].[Классы животных].[Класс животного] FROM [Veterinary Clinic].[dbo].[Виды животных]" +

$"INNER JOIN [Veterinary Clinic].[dbo].[Классы животных] ON ([Veterinary Clinic].[dbo].[Виды животных].[Код класса животного]) = [Veterinary Clinic].[dbo].[Классы животных].[Код класса животного]";

}

else if (labelTemp.Text == "Лицензии")

{

sql = "SELECT [Veterinary Clinic].[dbo].[Лицензии].[Код лицензии]," +

"[Veterinary Clinic].[dbo].[Лицензии].[Лицензия №]," +

"[Veterinary Clinic].[dbo].[Лицензии].[Срок окончания лицензии]," +

"[Veterinary Clinic].[dbo].[Лицензии].[Фото лицензии]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Название пункта] FROM [Veterinary Clinic].[dbo].[Лицензии]" +

$"INNER JOIN [Veterinary Clinic].[dbo].[Ветеринарные клиники] ON ([Veterinary Clinic].[dbo].[Лицензии].[Код ветеринарной клинки]) = [Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код ветеринарной клинки]";

}

else

{

sql = $"SELECT \* FROM [{labelTemp.Text}]";

}

var command = new SqlCommand(sql, connection);

using (var reader = command.ExecuteReader())

{

var schemaTable = reader.GetSchemaTable();

foreach (DataRow row in schemaTable.Rows)

{

columnName.Add(row["ColumnName"].ToString());

}

}

fieldsList.DataSource = columnName;

}

}

}

/// <summary>

/// Метод обновления компонента DataGridView

/// </summary>

/// <returns></returns>

private void Refresh()

{

switch (nameTable.Text)

{

case "Ветеринарные клиники":

var dataTable = Refresher.RefreshVeterinaryClinic();

dataGridView1.DataSource = dataTable;

dataGridView1.Invalidate();

counter.Text = "" + (dataGridView1.RowCount);

break;

case "Владельцы":

var dataTableOwner = Refresher.RefreshOwner();

dataGridView1.DataSource = dataTableOwner;

dataGridView1.Invalidate();

counter.Text = "" + (dataGridView1.RowCount);

break;

case "Процедуры":

var dataTableProcedure = Refresher.RefreshProcedure();

dataGridView1.DataSource = dataTableProcedure;

dataGridView1.Invalidate();

counter.Text = "" + (dataGridView1.RowCount);

break;

case "Сотрудники":

var dataTableEmployee = Refresher.RefreshEmployee();

dataGridView1.DataSource = dataTableEmployee;

dataGridView1.Invalidate();

counter.Text = "" + (dataGridView1.RowCount);

break;

case "Животные":

var dataTableAnimal = Refresher.RefreshAnimal();

dataGridView1.DataSource = dataTableAnimal;

dataGridView1.Invalidate();

counter.Text = "" + (dataGridView1.RowCount);

break;

case "Виды процедуры":

var dataTableProcedureType = Refresher.RefreshProcedureType();

dataGridView1.DataSource = dataTableProcedureType;

dataGridView1.Invalidate();

counter.Text = "" + (dataGridView1.RowCount);

break;

case "Должности":

var dataTablePosition = Refresher.RefreshPosition();

dataGridView1.DataSource = dataTablePosition;

dataGridView1.Invalidate();

counter.Text = "" + (dataGridView1.RowCount);

break;

case "Классы животных":

var dataTableAnimalClass = Refresher.RefreshAnimalClass();

dataGridView1.DataSource = dataTableAnimalClass;

dataGridView1.Invalidate();

counter.Text = "" + (dataGridView1.RowCount);

break;

case "Виды животных":

var dataTableAnimalType = Refresher.RefreshAnimalType();

dataGridView1.DataSource = dataTableAnimalType;

dataGridView1.Invalidate();

counter.Text = "" + (dataGridView1.RowCount);

break;

case "Районы":

var dataTableDistrict = Refresher.RefreshDistrict();

dataGridView1.DataSource = dataTableDistrict;

dataGridView1.Invalidate();

counter.Text = "" + (dataGridView1.RowCount);

break;

case "Тип собственности":

var dataTableOwnership = Refresher.RefreshOwnership();

dataGridView1.DataSource = dataTableOwnership;

dataGridView1.Invalidate();

counter.Text = "" + (dataGridView1.RowCount);

break;

case "Лицензии":

var dataTableLicense = Refresher.RefreshLicense();

dataGridView1.DataSource = dataTableLicense;

dataGridView1.Invalidate();

counter.Text = "" + (dataGridView1.RowCount);

break;

default:

break;

}

}

private void removeByFieldButton\_Click(object sender, EventArgs e)

{

try

{

DialogResult result = MessageBox.Show("Могут быть удалены запии из других таблиц.Вы уверены, что хотите продолжить?", "Удаление по полю", MessageBoxButtons.YesNo);

if (result == DialogResult.No)

{

return;

}

else

{

Remover.DeleteNotesByFields(valueBox.Text, nameTable.Text, labelTemp.Text, labelTemp2.Text);

Refresh();

}

}

catch(Exception ex)

{

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Documents;

using System.Windows.Media.Media3D;

using System.Xml;

using VeterenaryClinicApp.Model;

namespace VeterenaryClinicApp.Controller

{

public static class Remover

{

/// <summary>

/// Удаление записей ветеринарных клиник

/// </summary>

/// <param name="ids"></param>

/// <returns>true - если удаление успешно, иначе false</returns>

public static bool RemoveVeterinaryClinic(List<int> ids)

{

try

{

using (var db = new Veterinary\_ClinicEntities())

{

for (var i = 0; i < ids.Count; i++)

{

var recordToDelete = db.Ветеринарные\_клиники.Find(ids[i]);

db.Ветеринарные\_клиники.Remove(recordToDelete);

}

db.SaveChanges();

}

return true;

}

catch(Exception ex)

{

return false;

}

}

/// <summary>

/// Удаление записей владельцев

/// </summary>

/// <param name="ids"></param>

/// <returns>true - если удаление успешно, иначе false</returns>

public static bool RemoveOwner(List<int> ids)

{

try

{

using (var db = new Veterinary\_ClinicEntities())

{

for (var i = 0; i < ids.Count; i++)

{

var recordToDelete = db.Владельцы.Find(ids[i]);

db.Владельцы.Remove(recordToDelete);

}

db.SaveChanges();

}

return true;

}

catch (Exception ex)

{

return false;

}

}

/// <summary>

/// Удаление записей процедур

/// </summary>

/// <param name="ids"></param>

/// <returns>true - если удаление успешно, иначе false</returns>

public static bool RemoveProcedure(List<int> ids)

{

try

{

using (var db = new Veterinary\_ClinicEntities())

{

for (var i = 0; i < ids.Count; i++)

{

var recordToDelete = db.Процедуры.Find(ids[i]);

db.Процедуры.Remove(recordToDelete);

}

db.SaveChanges();

}

return true;

}

catch (Exception ex)

{

return false;

}

}

/// <summary>

/// Удаление записей сотрудников

/// </summary>

/// <param name="ids"></param>

/// <returns>true - если удаление успешно, иначе false</returns>

public static bool RemoveEmployee(List<int> ids)

{

try

{

using (var db = new Veterinary\_ClinicEntities())

{

for (var i = 0; i < ids.Count; i++)

{

var recordToDelete = db.Сотрудники.Find(ids[i]);

db.Сотрудники.Remove(recordToDelete);

}

db.SaveChanges();

}

return true;

}

catch (Exception ex)

{

return false;

}

}

/// <summary>

/// Удаление записей животных

/// </summary>

/// <param name="ids"></param>

/// <returns>true - если удаление успешно, иначе false</returns>

public static bool RemoveAnimal(List<int> ids)

{

try

{

using (var db = new Veterinary\_ClinicEntities())

{

for (var i = 0; i < ids.Count; i++)

{

var recordToDelete = db.Животные.Find(ids[i]);

db.Животные.Remove(recordToDelete);

}

db.SaveChanges();

}

return true;

}

catch (Exception ex)

{

return false;

}

}

/// <summary>

/// Удаление записей животных

/// </summary>

/// <param name="ids"></param>

/// <returns>true - если удаление успешно, иначе false</returns>

public static bool RemoveProcedureType(List<int> ids)

{

try

{

using (var db = new Veterinary\_ClinicEntities())

{

for (var i = 0; i < ids.Count; i++)

{

var recordToDelete = db.Виды\_процедуры.Find(ids[i]);

db.Виды\_процедуры.Remove(recordToDelete);

}

db.SaveChanges();

}

return true;

}

catch (Exception ex)

{

return false;

}

}

/// <summary>

/// Удаление записей должности

/// </summary>

/// <param name="ids"></param>

/// <returns>true - если удаление успешно, иначе false</returns>

public static bool RemovePosition(List<int> ids)

{

try

{

using (var db = new Veterinary\_ClinicEntities())

{

for (var i = 0; i < ids.Count; i++)

{

var recordToDelete = db.Должности.Find(ids[i]);

db.Должности.Remove(recordToDelete);

}

db.SaveChanges();

}

return true;

}

catch (Exception ex)

{

return false;

}

}

/// <summary>

/// Удаление записей класс животного

/// </summary>

/// <param name="ids"></param>

/// <returns>true - если удаление успешно, иначе false</returns>

public static bool RemoveAnimalClass(List<int> ids)

{

try

{

using (var db = new Veterinary\_ClinicEntities())

{

for (var i = 0; i < ids.Count; i++)

{

var recordToDelete = db.Классы\_животных.Find(ids[i]);

db.Классы\_животных.Remove(recordToDelete);

}

db.SaveChanges();

}

return true;

}

catch (Exception ex)

{

return false;

}

}

/// <summary>

/// Удаление записей виды животного

/// </summary>

/// <param name="ids"></param>

/// <returns>true - если удаление успешно, иначе false</returns>

public static bool RemoveAnimalType(List<int> ids)

{

try

{

using (var db = new Veterinary\_ClinicEntities())

{

for (var i = 0; i < ids.Count; i++)

{

var recordToDelete = db.Виды\_животных.Find(ids[i]);

db.Виды\_животных.Remove(recordToDelete);

}

db.SaveChanges();

}

return true;

}

catch (Exception ex)

{

return false;

}

}

/// <summary>

/// Удаление записей районов

/// </summary>

/// <param name="ids"></param>

/// <returns>true - если удаление успешно, иначе false</returns>

public static bool RemoveDistrict(List<int> ids)

{

try

{

using (var db = new Veterinary\_ClinicEntities())

{

for (var i = 0; i < ids.Count; i++)

{

var recordToDelete = db.Районы.Find(ids[i]);

db.Районы.Remove(recordToDelete);

}

db.SaveChanges();

}

return true;

}

catch (Exception ex)

{

return false;

}

}

/// <summary>

/// Удаление записей типа собственности

/// </summary>

/// <param name="ids"></param>

/// <returns>true - если удаление успешно, иначе false</returns>

public static bool RemoveOwnership(List<int> ids)

{

try

{

using (var db = new Veterinary\_ClinicEntities())

{

for (var i = 0; i < ids.Count; i++)

{

var recordToDelete = db.Тип\_собственности.Find(ids[i]);

db.Тип\_собственности.Remove(recordToDelete);

}

db.SaveChanges();

}

return true;

}

catch (Exception ex)

{

return false;

}

}

/// <summary>

/// Удаление записей Лицензии

/// </summary>

/// <param name="ids"></param>

/// <returns>true - если удаление успешно, иначе false</returns>

public static bool RemoveLicense(List<int> ids)

{

try

{

using (var db = new Veterinary\_ClinicEntities())

{

for (var i = 0; i < ids.Count; i++)

{

var recordToDelete = db.Лицензии.Find(ids[i]);

db.Лицензии.Remove(recordToDelete);

}

db.SaveChanges();

}

return true;

}

catch (Exception ex)

{

return false;

}

}

public static (int,int) CountRemoveVeterinaryClinic(List<int> ids)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT COUNT(\*) FROM [Сотрудники] WHERE [Код Ветеринарной клиники] IN (" +

string.Join(",", ids.Select(x => $"'{x}'")) + ")";

SqlCommand command = new SqlCommand(query, myConnection);

int countEmployee = (int)command.ExecuteScalar();

command = new SqlCommand(query, myConnection);

query = "SELECT COUNT(\*) FROM [Лицензии] WHERE [Код Ветеринарной клинки] IN (" +

string.Join(",", ids.Select(x => $"'{x}'")) + ")";

int countLicense = (int)command.ExecuteScalar();

return (countEmployee,countLicense);

}

public static int CountRemoveOwner(List<int> ids)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT COUNT(\*) FROM [Животные] WHERE [Код владельца] IN (" +

string.Join(",", ids.Select(x => $"'{x}'")) + ")";

SqlCommand command = new SqlCommand(query, myConnection);

int count = (int)command.ExecuteScalar();

return count;

}

public static int CountRemoveAnimal(List<int> ids)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT COUNT(\*) FROM [Процедуры] WHERE [Код животного] IN (" +

string.Join(",", ids.Select(x => $"'{x}'")) + ")";

SqlCommand command = new SqlCommand(query, myConnection);

int count = (int)command.ExecuteScalar();

return count;

}

public static int CountRemoveEmployee(List<int> ids)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT COUNT(\*) FROM [Процедуры] WHERE [Код сотрудника] IN (" +

string.Join(",", ids.Select(x => $"'{x}'")) + ")";

SqlCommand command = new SqlCommand(query, myConnection);

int count = (int)command.ExecuteScalar();

return count;

}

public static (int,int) CountRemoveAnimalType(List<int> ids)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT COUNT(\*) FROM [Животные] WHERE [Код вида животного] IN (" +

string.Join(",", ids.Select(x => $"'{x}'")) + ")";

SqlCommand command = new SqlCommand(query, myConnection);

int count = (int)command.ExecuteScalar();

query = "SELECT COUNT(\*) FROM [Процедуры] WHERE [Код животного] IN (SELECT [Код животного] FROM [Животные] WHERE [Код вида животного] IN(" +

string.Join(",", ids.Select(x => $"'{x}'")) + "))";

command = new SqlCommand(query, myConnection);

int count2 = (int)command.ExecuteScalar();

return (count,count2);

}

public static (int,int,int) CountRemoveOwnershipType(List<int> ids)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT COUNT(\*) FROM [Ветеринарные клиники] WHERE [Код типа собственности] IN (" +

string.Join(",", ids.Select(x => $"'{x}'")) + ")";

SqlCommand command = new SqlCommand(query, myConnection);

int count = (int)command.ExecuteScalar();

query = "SELECT COUNT(\*) FROM [Сотрудники] WHERE [Код ветеринарной клиники] IN (SELECT [Код ветеринарной клинки] FROM [Ветеринарные клиники] WHERE [Код типа собственности] IN(" +

string.Join(",", ids.Select(x => $"'{x}'")) + "))";

command = new SqlCommand(query, myConnection);

int count2 = (int)command.ExecuteScalar();

query = "SELECT COUNT(\*) FROM [Лицензии] WHERE [Код ветеринарной клинки] IN (SELECT [Код ветеринарной клинки] FROM [Ветеринарные клиники] WHERE [Код типа собственности] IN(" +

string.Join(",", ids.Select(x => $"'{x}'")) + "))";

command = new SqlCommand(query, myConnection);

int count3 = (int)command.ExecuteScalar();

return (count,count2,count3);

}

public static (int,int,int) CountRemoveDistrict(List<int> ids)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT COUNT(\*) FROM [Ветеринарные клиники] WHERE [Код район города] IN (" +

string.Join(",", ids.Select(x => $"'{x}'")) + ")";

SqlCommand command = new SqlCommand(query, myConnection);

int count = (int)command.ExecuteScalar();

query = "SELECT COUNT(\*) FROM [Сотрудники] WHERE [Код ветеринарной клиники] IN (SELECT [Код ветеринарной клинки] FROM [Ветеринарные клиники] WHERE [Код район города] IN(" +

string.Join(",", ids.Select(x => $"'{x}'")) + "))";

command = new SqlCommand(query, myConnection);

int count2 = (int)command.ExecuteScalar();

query = "SELECT COUNT(\*) FROM [Лицензии] WHERE [Код ветеринарной клинки] IN (SELECT [Код ветеринарной клинки] FROM [Ветеринарные клиники] WHERE [Код район города] IN(" +

string.Join(",", ids.Select(x => $"'{x}'")) + "))";

command = new SqlCommand(query, myConnection);

int count3 = (int)command.ExecuteScalar();

return (count,count2,count3);

}

public static int CountRemoveProcedureType(List<int> ids)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT COUNT(\*) FROM [Процедуры] WHERE [Код вида процедуры] IN (" +

string.Join(",", ids.Select(x => $"'{x}'")) + ")";

SqlCommand command = new SqlCommand(query, myConnection);

int count = (int)command.ExecuteScalar();

return count;

}

public static (int, int) CountRemovePosition(List<int> ids)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT COUNT(\*) FROM [Сотрудники] WHERE [Код должности] IN (" +

string.Join(",", ids.Select(x => $"'{x}'")) + ")";

SqlCommand command = new SqlCommand(query, myConnection);

int count = (int)command.ExecuteScalar();

query = "SELECT COUNT(\*) FROM [Процедуры] WHERE [Код сотрудника] IN (SELECT [Код сотрудника] FROM [Сотрудники] WHERE [Код должности] IN(" +

string.Join(",", ids.Select(x => $"'{x}'")) + "))";

command = new SqlCommand(query, myConnection);

int count2 = (int)command.ExecuteScalar();

return (count, count2);

}

public static (int,int, int) CountRemoveAnimalClass(List<int> ids)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT COUNT(\*) FROM [Виды животных] WHERE [Код класса животного] IN (" +

string.Join(",", ids.Select(x => $"'{x}'")) + ")";

SqlCommand command = new SqlCommand(query, myConnection);

int count = (int)command.ExecuteScalar();

query = "SELECT COUNT(\*) FROM [Животные] WHERE [Код вида животного] IN (SELECT [Код вида животного] FROM [Виды животных] WHERE [Код класса животного] IN(" +

string.Join(",", ids.Select(x => $"'{x}'")) + "))";

command = new SqlCommand(query, myConnection);

int count2 = (int)command.ExecuteScalar();

query = "SELECT COUNT(\*) FROM [Процедуры] WHERE [Код животного] IN (SELECT [Код животного] FROM [Животные] WHERE [Код вида животного] IN (SELECT [Код вида животного] FROM [Виды животных] WHERE [Код класса животного] IN(" +

string.Join(",", ids.Select(x => $"'{x}'")) + ")))";

command = new SqlCommand(query, myConnection);

int count3 = (int)command.ExecuteScalar();

return (count,count2, count3);

}

public static DataTable LoadTables()

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string sql = "SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_TYPE='BASE TABLE'";

SqlCommand command = new SqlCommand(sql, myConnection);

SqlDataAdapter adapter = new SqlDataAdapter(command);

var table = new DataTable();

adapter.Fill(table);

return table;

}

public static void DeleteNotesByFields(string value,string nameTable,string field,string sign)

{

try

{

if (value != null)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string sql;

int intValue;

decimal decValue;

DateTime dateValue;

if (int.TryParse(value, out intValue))

{

sql = $"DELETE FROM [Veterinary Clinic].[dbo].[{nameTable}] " +

$"WHERE [Veterinary Clinic].[dbo].[{nameTable}].[{field}] {sign} {intValue}";

}

else if (decimal.TryParse(value, out decValue))

{

sql = $"DELETE FROM [Veterinary Clinic].[dbo].[{nameTable}] " +

$"WHERE [Veterinary Clinic].[dbo].[{nameTable}].[{field}] {sign} {decValue}";

}

else if (DateTime.TryParse(value, out dateValue))

{

string date = $"{dateValue.Month}-{dateValue.Day}-{dateValue.Year}";

sql = $"DELETE FROM [Veterinary Clinic].[dbo].[{nameTable}] " +

$"WHERE [Veterinary Clinic].[dbo].[{nameTable}].[{field}] {sign} {date}";

}

else

{

if (nameTable == "Ветеринарные клиники")

{

if (field == "Тип собственности")

{

nameTable = "Тип собственности";

sql = "DELETE FROM [Veterinary Clinic].[dbo].[Ветеринарные клиники] " +

$" WHERE [Код типа собственности] IN (SELECT [Код типа собственности] FROM [{nameTable}] WHERE [{field}] = N'{value}')";

}

else if (field == "Район города")

{

nameTable = "Районы";

sql = "DELETE FROM [Veterinary Clinic].[dbo].[Ветеринарные клиники] " +

$" WHERE [Код район города] IN (SELECT [Код района] FROM [{nameTable}] WHERE [{field}] = N'{value}')";

}

else

{

sql = $"DELETE FROM [Veterinary Clinic].[dbo].[{nameTable}] " +

$"WHERE [Veterinary Clinic].[dbo].[{nameTable}].[{field}] = N'{value}'";

}

}

else if (nameTable == "Сотрудники")

{

if (field == "Должность")

{

nameTable = "Должности";

sql = "DELETE FROM [Veterinary Clinic].[dbo].[Сотрудники] " +

$" WHERE [Код должности] IN (SELECT [Код должности] FROM [{nameTable}] WHERE [{field}] = N'{value}')";

}

else if (field == "Номер регистрационного пункта" || field == "Название пункта")

{

nameTable = "Ветеринарные клиники";

sql = "DELETE FROM [Veterinary Clinic].[dbo].[Сотрудники] " +

$" WHERE [Код ветеринарной клинки] IN (SELECT [Код ветеринарной клинки] FROM [{nameTable}] WHERE [{field}] = N'{value}')";

}

else

{

sql = "DELETE FROM [Veterinary Clinic].[dbo].[Ветеринарные клиники] " +

$"WHERE [Veterinary Clinic].[dbo].[{nameTable}].[{field}] = N'{value}'";

}

}

else if (nameTable == "Животные")

{

if (field == "Имя" || field == "Фамилия" || field == "Отчество")

{

nameTable = "Владельцы";

sql = $"DELETE FROM [Veterinary Clinic].[dbo].[Животные] " +

$" WHERE [Код владельца] IN (SELECT [Код владельца] FROM [{nameTable}] WHERE [{field}] = N'{value}')";

}

else if (field == "Вид животного")

{

nameTable = "Виды животных";

sql = $"DELETE FROM [Veterinary Clinic].[dbo].[Животные] " +

$" WHERE [Код вида животного] IN (SELECT [Код вида животного] FROM [{nameTable}] WHERE [{field}] = N'{value}')";

}

else

{

sql = "DELETE FROM [Veterinary Clinic].[dbo].[Ветеринарные клиники] " +

$"WHERE [Veterinary Clinic].[dbo].[{nameTable}].[{field}] = N'{value}'";

}

}

else if (nameTable == "Процедуры")

{

if (field == "Имя" || field == "Фамилия" || field == "Отчество")

{

nameTable = "Сотрудники";

sql = $"DELETE FROM [Veterinary Clinic].[dbo].[Процедуры] " +

$" WHERE [Код сотрудника] IN (SELECT [Код сотрудника] FROM [{nameTable}] WHERE [{field}] = N'{value}')";

}

else if (field == "Кличка животного")

{

nameTable = "Животные";

sql = $"DELETE FROM [Veterinary Clinic].[dbo].[Процедуры] " +

$" WHERE [Код животного] IN (SELECT [Код животного] FROM [{nameTable}] WHERE [{field}] = N'{value}')";

}

else if (field == "Вид животного")

{

nameTable = "Виды животных";

sql = $"DELETE FROM [Veterinary Clinic].[dbo].[Процедуры] " +

$" WHERE [Код животного] IN (SELECT [Код животного] FROM [Животные] WHERE [Код животного] IN (SELECT [Код животного] FROM [{nameTable}] WHERE [Код вида животного] = N'{value}'))";

}

else if (field == "Вид процедуры")

{

nameTable = "Виды процедуры";

sql = $"DELETE FROM [Veterinary Clinic].[dbo].[Процедуры " +

$" WHERE [Код вида процедуры] IN (SELECT [Код вида процедуры] FROM [{nameTable}] WHERE [{field}] = N'{value}')";

}

else

{

sql = "DELETE FROM [Veterinary Clinic].[dbo].[Ветеринарные клиники] " +

$"WHERE [Veterinary Clinic].[dbo].[{nameTable}].[{field}] = N'{value}'";

}

}

else if (nameTable == "Виды животных")

{

if (field == "Класс животного")

{

nameTable = "Классы животных";

sql = $"DELETE FROM [Veterinary Clinic].[dbo].[Виды животных] " +

$" WHERE [Код класса животного] IN (SELECT [Код класса животного] FROM [{nameTable}] WHERE [{field}] = N'{value}')";

}

else

{

sql = "DELETE FROM [Veterinary Clinic].[dbo].[Ветеринарные клиники] " +

$"WHERE [Veterinary Clinic].[dbo].[{nameTable}].[{field}] = N'{value}'";

}

sql = "DELETE FROM [Veterinary Clinic].[dbo].[Виды животных]" +

$"INNER JOIN [Veterinary Clinic].[dbo].[Классы животных] ON ([Veterinary Clinic].[dbo].[Виды животных].[Код класса животного]) = [Veterinary Clinic].[dbo].[Классы животных].[Код класса животного] WHERE [Veterinary Clinic].[dbo].[{nameTable}].[{field}] = N'{value}'";

}

else if (nameTable == "Лицензии")

{

if (field == "Название пункта")

{

nameTable = "Ветеринарные клиники";

sql = $"DELETE FROM [Veterinary Clinic].[dbo].[Лицензии] " +

$" WHERE [Код ветеринарной клинки] IN (SELECT [Код ветеринарной клинки] FROM [{nameTable}] WHERE [{field}] = N'{value}')";

}

else

{

sql = "DELETE FROM [Veterinary Clinic].[dbo].[Ветеринарные клиники] " +

$"WHERE [Veterinary Clinic].[dbo].[{nameTable}].[{field}] = N'{value}'";

}

}

else

{

sql = $"DELETE FROM [Veterinary Clinic].[dbo].[{nameTable}] WHERE [Veterinary Clinic].[dbo].[{nameTable}].[{field}] = N'{value}'";

}

}

SqlCommand command = new SqlCommand(sql, myConnection);

command.ExecuteScalar();

}

}

catch(Exception ex)

{

}

}

}

}

MessageBox.Show($"{ex.Message}!\nПопробуйте иначе!");

}

}

private void fieldsList\_SelectedIndexChanged(object sender, EventArgs e)

{

var fieldName = fieldsList.GetItemText(fieldsList.SelectedItem);

labelTemp.Text = fieldName;

}

private void signList\_SelectedIndexChanged(object sender, EventArgs e)

{

var sign = signList.GetItemText(signList.SelectedItem);

labelTemp2.Text = sign;

}

}

}

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;

using VeterenaryClinicApp.Controller;

using VeterenaryClinicApp.Model;

namespace VeterenaryClinicApp

{

public partial class MainForm : Form

{

protected override void WndProc(ref Message m)

{

if (m.Msg == 0x84)

{

base.WndProc(ref m);

if ((int)m.Result == 0x1)

m.Result = (IntPtr)0x2;

return;

}

base.WndProc(ref m);

}

public MainForm()

{

InitializeComponent();

customizeDesign();

Generator.GenerateDataBase();

}

private void sideMenuPanel\_Paint(object sender, PaintEventArgs e)

{

}

private void MainForm\_Load(object sender, EventArgs e)

{

}

private void customizeDesign()

{

tablesPanel.Visible = false;

dictionaryPanel.Visible = false;

requestsPanel.Visible = false;

}

private void hideSubMenu()

{

if (tablesPanel.Visible)

tablesPanel.Visible = false;

if(dictionaryPanel.Visible)

dictionaryPanel.Visible = false;

if(requestsPanel.Visible)

requestsPanel.Visible = false;

}

private void showSubMenu(Panel subMenu)

{

if (!subMenu.Visible)

{

hideSubMenu();

subMenu.Visible = true;

}

else

{

subMenu.Visible = false;

}

}

private void tablesButton\_Click(object sender, EventArgs e)

{

showSubMenu(tablesPanel);

}

private void dictionaryButton\_Click(object sender, EventArgs e)

{

showSubMenu(dictionaryPanel);

}

private void requestsButton\_Click(object sender, EventArgs e)

{

showSubMenu(requestsPanel);

}

private Form activeForm = null;

private void OpenChildForm(Form childForm)

{

if(activeForm != null)

activeForm.Close();

activeForm = childForm;

childForm.TopLevel = false;

childForm.FormBorderStyle = FormBorderStyle.None;

childForm.Dock = DockStyle.Fill;

ChildFormPanel.Controls.Add(childForm);

ChildFormPanel.Tag = childForm;

childForm.BringToFront();

childForm.Show();

}

/// <summary>

/// Открывает таблицу с Ветеринарными клиниками

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button2\_Click(object sender, EventArgs e)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT [Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код ветеринарной клинки]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Номер регистрационного пункта]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Год открытия]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Адрес пункта]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Название пункта]," +

"[Veterinary Clinic].[dbo].[Районы].[Район города]," +

"[Veterinary Clinic].[dbo].[Тип собственности].[Тип собственности]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Телефон] FROM [Veterinary Clinic].[dbo].[Ветеринарные клиники]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Тип собственности] ON ([Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код типа собственности]) = [Veterinary Clinic].[dbo].[Тип собственности].[Код типа собственности]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Районы] ON ([Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код район города]) = [Veterinary Clinic].[dbo].[Районы].[Код района]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

OpenChildForm(new TableForm(dataTable,"Ветеринарные клиники"));

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

private void pictureBox2\_Click(object sender, EventArgs e)

{

this.Close();

}

private void pictureBox2\_MouseLeave(object sender, EventArgs e)

{

pictureBox2.Image = new Bitmap(@"C:\Users\fikra\Source\Repos\VeterinaryClinic\VeterenaryClinicApp\Resources\3005766\_account\_door\_exit\_logout\_icon.png");

}

private void pictureBox2\_MouseMove(object sender, MouseEventArgs e)

{

pictureBox2.Image = new Bitmap(@"C:\Users\fikra\Source\Repos\VeterinaryClinic\VeterenaryClinicApp\Resources\3005766\_account\_door\_exit\_logout\_icon (1).png");

}

/// <summary>

/// Открывает таблицы с владельцами животных

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void ownerButton\_Click(object sender, EventArgs e)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT \* FROM [Veterinary Clinic].[dbo].[Владельцы]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

OpenChildForm(new TableForm(dataTable, "Владельцы"));

}

/// <summary>

/// Открывает таблицы с сотрудниками ветеринарных клиник

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void employeesButton\_Click(object sender, EventArgs e)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT [Veterinary Clinic].[dbo].[Сотрудники].[Код сотрудника]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Имя]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Фамилия]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Отчество]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Дата рождения]," +

"[Veterinary Clinic].[dbo].[Должности].[Должность]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Стаж]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Оклад]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Номер регистрационного пункта]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Название пункта] FROM [Veterinary Clinic].[dbo].[Сотрудники]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Ветеринарные клиники] ON ([Veterinary Clinic].[dbo].[Сотрудники].[Код ветеринарной клиники]) = [Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код ветеринарной клинки]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Должности] ON ([Veterinary Clinic].[dbo].[Сотрудники].[Код должности]) = [Veterinary Clinic].[dbo].[Должности].[Код должности]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

OpenChildForm(new TableForm(dataTable, "Сотрудники"));

}

/// <summary>

/// Открывает таблицы с животными

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void animalsButton\_Click(object sender, EventArgs e)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT [Veterinary Clinic].[dbo].[Животные].[Код животного]," +

"[Veterinary Clinic].[dbo].[Животные].[Кличка Животного]," +

"[Veterinary Clinic].[dbo].[Животные].[Возраст Животного]," +

"[Veterinary Clinic].[dbo].[Животные].[Условия содержания животного]," +

"[Veterinary Clinic].[dbo].[Владельцы].[Имя]," +

"[Veterinary Clinic].[dbo].[Владельцы].[Фамилия]," +

"[Veterinary Clinic].[dbo].[Владельцы].[Отчество]," +

"[Veterinary Clinic].[dbo].[Виды животных].[Вид животного] FROM [Veterinary Clinic].[dbo].[Животные]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Владельцы] ON ([Veterinary Clinic].[dbo].[Животные].[Код владельца]) = [Veterinary Clinic].[dbo].[Владельцы].[Код владельца] " +

"INNER JOIN [Veterinary Clinic].[dbo].[Виды животных] ON ([Veterinary Clinic].[dbo].[Животные].[Код вида животного]) = [Veterinary Clinic].[dbo].[Виды животных].[Код вида животного]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

OpenChildForm(new TableForm(dataTable, "Животные"));

}

/// <summary>

/// Открывает таблицы с процедурами

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void procButton\_Click(object sender, EventArgs e)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT [Veterinary Clinic].[dbo].[Процедуры].[Код процедуры]," +

"[Veterinary Clinic].[dbo].[Процедуры].[Дата оказания помощи животному]," +

"[Veterinary Clinic].[dbo].[Процедуры].[Цена процедуры]," +

"[Veterinary Clinic].[dbo].[Процедуры].[Скидка на эту процедуру]," +

"[Veterinary Clinic].[dbo].[Процедуры].[Цена материала по этой процедуре]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Имя]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Фамилия]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Отчество]," +

"[Veterinary Clinic].[dbo].[Животные].[Кличка животного]," +

"[Veterinary Clinic].[dbo].[Виды животных].[Вид животного]," +

"[Veterinary Clinic].[dbo].[Виды процедуры].[Вид процедуры] FROM [Veterinary Clinic].[dbo].[Процедуры]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Сотрудники] ON ([Veterinary Clinic].[dbo].[Процедуры].[Код сотрудника]) = [Veterinary Clinic].[dbo].[Сотрудники].[Код сотрудника]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Животные] ON ([Veterinary Clinic].[dbo].[Процедуры].[Код животного]) = [Veterinary Clinic].[dbo].[Животные].[Код животного] " +

"INNER JOIN [Veterinary Clinic].[dbo].[Виды животных] ON ([Veterinary Clinic].[dbo].[Животные].[Код вида животного]) = [Veterinary Clinic].[dbo].[Виды животных].[Код вида животного] " +

"INNER JOIN [Veterinary Clinic].[dbo].[Виды процедуры] ON ([Veterinary Clinic].[dbo].[Процедуры].[Код вида процедуры]) = [Veterinary Clinic].[dbo].[Виды процедуры].[Код вида процедуры] ";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

OpenChildForm(new TableForm(dataTable, "Процедуры"));

}

/// <summary>

/// Открывает справочник с должностями

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void positionsButton\_Click(object sender, EventArgs e)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT \* FROM [Veterinary Clinic].[dbo].[Должности]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

OpenChildForm(new TableForm(dataTable, "Должности"));

}

/// <summary>

/// Открывает справочник с районами

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void disctrictsButton\_Click(object sender, EventArgs e)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT \* FROM [Veterinary Clinic].[dbo].[Районы]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

OpenChildForm(new TableForm(dataTable, "Районы"));

}

/// <summary>

/// Открывает справочник с классами животных

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void classesButton\_Click(object sender, EventArgs e)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT \* FROM [Veterinary Clinic].[dbo].[Классы животных]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

OpenChildForm(new TableForm(dataTable, "Классы животных"));

}

/// <summary>

/// Открывает справочник с видами животных

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void typeButton\_Click(object sender, EventArgs e)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT [Veterinary Clinic].[dbo].[Виды животных].[Код вида животного]," +

"[Veterinary Clinic].[dbo].[Виды животных].[Вид животного]," +

"[Veterinary Clinic].[dbo].[Классы животных].[Класс животного] FROM [Veterinary Clinic].[dbo].[Виды животных]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Классы животных] ON ([Veterinary Clinic].[dbo].[Виды животных].[Код класса животного]) = [Veterinary Clinic].[dbo].[Классы животных].[Код класса животного]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

OpenChildForm(new TableForm(dataTable, "Виды животных"));

}

/// <summary>

/// Открывает справочник с типами собственности

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void ownershipButton\_Click(object sender, EventArgs e)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT \* FROM [Veterinary Clinic].[dbo].[Тип собственности]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

OpenChildForm(new TableForm(dataTable, "Тип собственности"));

}

/// <summary>

/// Открывает справочник с лицензиями

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void licencesButton\_Click(object sender, EventArgs e)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT [Veterinary Clinic].[dbo].[Лицензии].[Код лицензии]," +

"[Veterinary Clinic].[dbo].[Лицензии].[Лицензия №]," +

"[Veterinary Clinic].[dbo].[Лицензии].[Срок окончания лицензии]," +

"[Veterinary Clinic].[dbo].[Лицензии].[Фото лицензии]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Название пункта] FROM [Veterinary Clinic].[dbo].[Лицензии]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Ветеринарные клиники] ON ([Veterinary Clinic].[dbo].[Лицензии].[Код ветеринарной клинки]) = [Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код ветеринарной клинки]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

OpenChildForm(new TableForm(dataTable, "Лицензии"));

}

/// <summary>

/// Открывает справочник с видами процедур

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void typeProcButton\_Click(object sender, EventArgs e)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT \* FROM [Veterinary Clinic].[dbo].[Виды процедуры]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

OpenChildForm(new TableForm(dataTable, "Виды процедуры"));

}

/// <summary>

/// Открывает форму с фильтрацией таблицы

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button1\_Click(object sender, EventArgs e)//фильтрация

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT [Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код ветеринарной клинки]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Номер регистрационного пункта]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Год открытия]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Адрес пункта]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Название пункта]," +

"[Veterinary Clinic].[dbo].[Районы].[Район города]," +

"[Veterinary Clinic].[dbo].[Тип собственности].[Тип собственности]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Телефон] FROM [Veterinary Clinic].[dbo].[Ветеринарные клиники]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Тип собственности] ON ([Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код типа собственности]) = [Veterinary Clinic].[dbo].[Тип собственности].[Код типа собственности]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Районы] ON ([Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код район города]) = [Veterinary Clinic].[dbo].[Районы].[Код района]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTableVetClinic = new DataTable();

adapter.Fill(dataTableVetClinic);

query = "SELECT [Veterinary Clinic].[dbo].[Сотрудники].[Код сотрудника]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Имя]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Фамилия]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Отчество]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Дата рождения]," +

"[Veterinary Clinic].[dbo].[Должности].[Должность]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Стаж]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Оклад]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Номер регистрационного пункта]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Название пункта] FROM [Veterinary Clinic].[dbo].[Сотрудники]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Ветеринарные клиники] ON ([Veterinary Clinic].[dbo].[Сотрудники].[Код ветеринарной клиники]) = [Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код ветеринарной клинки]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Должности] ON ([Veterinary Clinic].[dbo].[Сотрудники].[Код должности]) = [Veterinary Clinic].[dbo].[Должности].[Код должности]";

var dataTableEmployee = new DataTable();

adapter.Fill(dataTableEmployee);

OpenChildForm(new FilterForm(dataTableVetClinic,dataTableEmployee));

}

/// <summary>

/// Открывает форму с запросами к таблице

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void requestButton\_Click(object sender, EventArgs e)

{

OpenChildForm(new RequestForm());

}

/// <summary>

/// Открывает форму с поиском по таблице

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void searchButton\_Click(object sender, EventArgs e)

{

OpenChildForm(new SearchingForm());

}

private void button3\_Click(object sender, EventArgs e)

{

showSubMenu(diagramPanel);

}

private void button6\_Click(object sender, EventArgs e)//диаграмма лучшие клиники

{

OpenChildForm(new StatisticForm());

}

private void button4\_Click(object sender, EventArgs e)//диаграмма лучшие сотрудники

{

OpenChildForm(new StatisticForm2());

}

private void button5\_Click(object sender, EventArgs e)//диаграмма доходы клиник

{

OpenChildForm(new StatisticForm3());

}

}

}using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace VeterenaryClinicApp.Controller

{

public static class Refresher

{

public static DataTable RefreshVeterinaryClinic()

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT [Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код ветеринарной клинки]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Номер регистрационного пункта]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Год открытия]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Адрес пункта]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Название пункта]," +

"[Veterinary Clinic].[dbo].[Районы].[Район города]," +

"[Veterinary Clinic].[dbo].[Тип собственности].[Тип собственности]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Телефон] FROM [Veterinary Clinic].[dbo].[Ветеринарные клиники]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Тип собственности] ON ([Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код типа собственности]) = [Veterinary Clinic].[dbo].[Тип собственности].[Код типа собственности]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Районы] ON ([Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код район города]) = [Veterinary Clinic].[dbo].[Районы].[Код района]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

return dataTable;

}

public static DataTable RefreshOwner()

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT \* FROM [Veterinary Clinic].[dbo].[Владельцы]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

return dataTable;

}

public static DataTable RefreshProcedure()

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT [Veterinary Clinic].[dbo].[Процедуры].[Код процедуры]," +

"[Veterinary Clinic].[dbo].[Процедуры].[Дата оказания помощи животному]," +

"[Veterinary Clinic].[dbo].[Процедуры].[Цена процедуры]," +

"[Veterinary Clinic].[dbo].[Процедуры].[Скидка на эту процедуру]," +

"[Veterinary Clinic].[dbo].[Процедуры].[Цена материала по этой процедуре]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Имя]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Фамилия]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Отчество]," +

"[Veterinary Clinic].[dbo].[Животные].[Кличка животного]," +

"[Veterinary Clinic].[dbo].[Виды животных].[Вид животного]," +

"[Veterinary Clinic].[dbo].[Виды процедуры].[Вид процедуры] FROM [Veterinary Clinic].[dbo].[Процедуры]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Сотрудники] ON ([Veterinary Clinic].[dbo].[Процедуры].[Код сотрудника]) = [Veterinary Clinic].[dbo].[Сотрудники].[Код сотрудника]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Животные] ON ([Veterinary Clinic].[dbo].[Процедуры].[Код животного]) = [Veterinary Clinic].[dbo].[Животные].[Код животного] " +

"INNER JOIN [Veterinary Clinic].[dbo].[Виды животных] ON ([Veterinary Clinic].[dbo].[Животные].[Код вида животного]) = [Veterinary Clinic].[dbo].[Виды животных].[Код вида животного] " +

"INNER JOIN [Veterinary Clinic].[dbo].[Виды процедуры] ON ([Veterinary Clinic].[dbo].[Процедуры].[Код вида процедуры]) = [Veterinary Clinic].[dbo].[Виды процедуры].[Код вида процедуры] ";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

return dataTable;

}

public static DataTable RefreshEmployee()

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT [Veterinary Clinic].[dbo].[Сотрудники].[Код сотрудника]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Имя]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Фамилия]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Отчество]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Дата рождения]," +

"[Veterinary Clinic].[dbo].[Должности].[Должность]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Стаж]," +

"[Veterinary Clinic].[dbo].[Сотрудники].[Оклад]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Номер регистрационного пункта]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Название пункта] FROM [Veterinary Clinic].[dbo].[Сотрудники]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Ветеринарные клиники] ON ([Veterinary Clinic].[dbo].[Сотрудники].[Код ветеринарной клиники]) = [Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код ветеринарной клинки]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Должности] ON ([Veterinary Clinic].[dbo].[Сотрудники].[Код должности]) = [Veterinary Clinic].[dbo].[Должности].[Код должности]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

return dataTable;

}

public static DataTable RefreshAnimal()

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT [Veterinary Clinic].[dbo].[Животные].[Код животного]," +

"[Veterinary Clinic].[dbo].[Животные].[Кличка Животного]," +

"[Veterinary Clinic].[dbo].[Животные].[Возраст Животного]," +

"[Veterinary Clinic].[dbo].[Животные].[Условия содержания животного]," +

"[Veterinary Clinic].[dbo].[Владельцы].[Имя]," +

"[Veterinary Clinic].[dbo].[Владельцы].[Фамилия]," +

"[Veterinary Clinic].[dbo].[Владельцы].[Отчество]," +

"[Veterinary Clinic].[dbo].[Виды животных].[Вид животного] FROM [Veterinary Clinic].[dbo].[Животные]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Владельцы] ON ([Veterinary Clinic].[dbo].[Животные].[Код владельца]) = [Veterinary Clinic].[dbo].[Владельцы].[Код владельца] " +

"INNER JOIN [Veterinary Clinic].[dbo].[Виды животных] ON ([Veterinary Clinic].[dbo].[Животные].[Код вида животного]) = [Veterinary Clinic].[dbo].[Виды животных].[Код вида животного]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

return dataTable;

}

public static DataTable RefreshProcedureType()

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT \* FROM [Veterinary Clinic].[dbo].[Виды процедуры]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

return dataTable;

}

public static DataTable RefreshPosition()

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT \* FROM [Veterinary Clinic].[dbo].[Должности]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

return dataTable;

}

public static DataTable RefreshAnimalClass()

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT \* FROM [Veterinary Clinic].[dbo].[Классы животных]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

return dataTable;

}

public static DataTable RefreshAnimalType()

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT [Veterinary Clinic].[dbo].[Виды животных].[Код вида животного]," +

"[Veterinary Clinic].[dbo].[Виды животных].[Вид животного]," +

"[Veterinary Clinic].[dbo].[Классы животных].[Класс животного] FROM [Veterinary Clinic].[dbo].[Виды животных]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Классы животных] ON ([Veterinary Clinic].[dbo].[Виды животных].[Код класса животного]) = [Veterinary Clinic].[dbo].[Классы животных].[Код класса животного]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

return dataTable;

}

public static DataTable RefreshDistrict()

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT \* FROM [Veterinary Clinic].[dbo].[Районы]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

return dataTable;

}

public static DataTable RefreshOwnership()

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT \* FROM [Veterinary Clinic].[dbo].[Тип собственности]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

return dataTable;

}

public static DataTable RefreshLicense()

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string query = "SELECT [Veterinary Clinic].[dbo].[Лицензии].[Код лицензии]," +

"[Veterinary Clinic].[dbo].[Лицензии].[Лицензия №]," +

"[Veterinary Clinic].[dbo].[Лицензии].[Срок окончания лицензии]," +

"[Veterinary Clinic].[dbo].[Лицензии].[Фото лицензии]," +

"[Veterinary Clinic].[dbo].[Ветеринарные клиники].[Название пункта] FROM [Veterinary Clinic].[dbo].[Лицензии]" +

"INNER JOIN [Veterinary Clinic].[dbo].[Ветеринарные клиники] ON ([Veterinary Clinic].[dbo].[Лицензии].[Код ветеринарной клинки]) = [Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код ветеринарной клинки]";

SqlDataAdapter adapter = new SqlDataAdapter(query, myConnection);

var dataTable = new DataTable();

adapter.Fill(dataTable);

return dataTable;

}

}

}

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;

using VeterenaryClinicApp.Controller;

namespace VeterenaryClinicApp

{

public partial class RequestForm : Form

{

public RequestForm()

{

InitializeComponent();

}

private void RequestForm\_Load(object sender, EventArgs e)

{

posBox.Enabled = false;

procedureTypeBox.Enabled = false;

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

string sql = "SELECT [Должность] FROM [Должности]";

SqlCommand command = new SqlCommand(sql, myConnection);

SqlDataAdapter adapter = new SqlDataAdapter(command);

var table = new DataTable();

adapter.Fill(table);

posBox.DataSource = table;

posBox.DisplayMember = "Должность";

var table2 = new DataTable();

sql = "SELECT [Вид процедуры] FROM [Виды процедуры]";

command = new SqlCommand(sql, myConnection);

adapter = new SqlDataAdapter(command);

adapter.Fill(table2);

procedureTypeBox.DataSource = table2;

procedureTypeBox.DisplayMember = "Вид процедуры";

}

private void pictureBox2\_Click(object sender, EventArgs e)

{

}

//private void pictureBox2\_MouseLeave(object sender, EventArgs e)

//{

// pictureBox2.Image = new Bitmap(@"C:\Users\fikra\source\repos\VeterenaryClinicApp\VeterenaryClinicApp\Resources\511955\_arrow\_before\_beginning\_left\_previous\_icon.png");

//}

//private void pictureBox2\_MouseMove(object sender, MouseEventArgs e)

//{

// pictureBox2.Image = new Bitmap(@"C:\Users\fikra\source\repos\VeterenaryClinicApp\VeterenaryClinicApp\Resources\511955\_arrow\_before\_beginning\_left\_previous\_icon (1).png");

//}

//private void textBox1\_TextChanged(object sender, EventArgs e)

//{

//}

//private void textBox1\_Click(object sender, EventArgs e)

//{

// textBox1.Clear();

//}

private void label1\_Click(object sender, EventArgs e)

{

this.Close();

}

private void label1\_MouseMove(object sender, MouseEventArgs e)

{

label1.ForeColor = Color.Red;

}

private void label1\_MouseLeave(object sender, EventArgs e)

{

label1.ForeColor = Color.White;

}

private void pictureBox3\_Click(object sender, EventArgs e)

{

MessageBox.Show(

"1-й запрос: Определить лучший ветеринарный пункт по каждому району и тройку лучших пунктов по городу в целом.\n" +

"2-й запрос: Определить сотрудника предоставившего максимальное количество услуг по каждому пункту, по району и по городу в целом.\n" +

"3-й запрос: Определить доходы каждого пункта в районе, доходы от пунктов по каждому району и по городу в целом за указанный год (в одном районе может находиться несколько пунктов).",

"FAQ"

);

}

string request;

private void simpleRequestsBox\_SelectedIndexChanged(object sender, EventArgs e)

{

request = simpleRequestsBox.GetItemText(simpleRequestsBox.SelectedItem);

if (request == "Ветеринарные клиники, где суммарный оклад сотрудников на опред. должности выше указанного" || request == "Учёт сотрудников по должности")

posBox.Enabled = true;

else

posBox.Enabled = false;

if (request == "Запрос для таблицы Процедуры без использования индекса (по виду процедуры)")

procedureTypeBox.Enabled = true;

else

procedureTypeBox.Enabled = false;

if (request == "Ветеринарные клиники" || request == "Сотрудники" || request == "Процедуры"

|| request == "Районы, в которых нет вет. клиник" || request == "Сотрудники, не делавшие процедур"

|| request == "Количество проведённых процедур всего и в каждом районе"

|| request == "Список клиник с количеством выполненных процедур"

|| request == "Ветеринарные клиники, которые находятся в районе Басманный"

|| request == "Ветеринарные клиники, которые находятся вне района Басманный"

|| request == "Запрос для таблицы Процедуры без использования индекса (по виду процедуры)"

|| request == "Использование оператора IN для поиска животных определенных видов"

|| request == "Использование оператора NOT IN для поиска животных, которые не относятся к определенным видам"

|| request == "Использование оператора CASE для вычисления скидки на процедуры"

|| request == "Использование подзапроса для выбора владельцев, у которых есть более одного животного"

|| request == "Статистика по ветеринарным клиникам(мод. представление через триггер)")

valueBox.Enabled = false;

else

valueBox.Enabled = true;

}

private void simpleRequestButton\_Click(object sender, EventArgs e)

{

try

{

DataTable table;

switch (request)

{

case "Учёт сотрудников по должности":

table = RequestsExecuter.Request\_1(positionBox);

dataGridView1.DataSource = table;

break;

case "Ветеринарные клиники в определённом районе":

table = RequestsExecuter.Request\_2(valueBox.Text);

dataGridView1.DataSource = table;

break;

case "Процедуры оказанные в опред. дату":

table = RequestsExecuter.Request\_3(valueBox.Text);

dataGridView1.DataSource = table;

break;

case "Сотрудники, родившиеся в опред. дату":

table = RequestsExecuter.Request\_4(valueBox.Text);

dataGridView1.DataSource = table;

break;

case "Ветеринарные клиники":

table = RequestsExecuter.Request\_5();

dataGridView1.DataSource = table;

break;

case "Сотрудники":

table = RequestsExecuter.Request\_6();

dataGridView1.DataSource = table;

break;

case "Процедуры":

table = RequestsExecuter.Request\_7();

dataGridView1.DataSource = table;

break;

case "Районы, в которых нет вет. клиник":

table = RequestsExecuter.Request\_8();

dataGridView1.DataSource = table;

break;

case "Сотрудники, не делавшие процедур":

table = RequestsExecuter.Request\_9();

dataGridView1.DataSource = table;

break;

case "Сотрудники, не делавшие процедур опред. даты":

table = RequestsExecuter.Request\_10(valueBox.Text);

dataGridView1.DataSource = table;

break;

case "Количество проведённых процедур всего и в каждом районе":

table = RequestsExecuter.Request\_11();

dataGridView1.DataSource = table;

break;

case "Количество сотрудников в ветеринарных клиниках с окладом больше указанного":

table = RequestsExecuter.Request\_12(valueBox.Text);

dataGridView1.DataSource = table;

break;

case "Ветеринарные клиники, где средний оклад сотрудников больше указанного":

table = RequestsExecuter.Request\_13(valueBox.Text);

dataGridView1.DataSource = table;

break;

case "Ветеринарные клиники, где суммарный оклад сотрудников на опред. должности выше указанного":

table = RequestsExecuter.Request\_14(valueBox.Text, positionBox);

dataGridView1.DataSource = table;

break;

case "Список клиник с количеством выполненных процедур":

table = RequestsExecuter.Request\_15();

dataGridView1.DataSource = table;

break;

case "Ветеринарные клиники, которые находятся в районе Басманный":

table = RequestsExecuter.Request\_16();

dataGridView1.DataSource = table;

break;

case "Ветеринарные клиники, которые находятся вне района Басманный":

table = RequestsExecuter.Request\_17();

dataGridView1.DataSource = table;

break;

case "Запрос для таблицы Ветеринарные клиники с использованием условия по значению (по коду ветеринарной клиники)":

table = RequestsExecuter.Request\_18(valueBox.Text);

dataGridView1.DataSource = table;

break;

case "Запрос для таблицы Животные с использованием условия по маске (по виду животного)":

table = RequestsExecuter.Request\_19(valueBox.Text);

dataGridView1.DataSource = table;

break;

case "Запрос для таблицы Процедуры с использованием индекса (по коду животного)":

table = RequestsExecuter.Request\_20(valueBox.Text);

dataGridView1.DataSource = table;

break;

case "Запрос для таблицы Процедуры без использования индекса (по виду процедуры)":

table = RequestsExecuter.Request\_21(procedureType);

dataGridView1.DataSource = table;

break;

case "Использование оператора IN для поиска животных определенных видов":

table = RequestsExecuter.Request\_22();

dataGridView1.DataSource = table;

break;

case "Использование оператора NOT IN для поиска животных, которые не относятся к определенным видам":

table = RequestsExecuter.Request\_23();

dataGridView1.DataSource = table;

break;

case "Использование оператора CASE для вычисления скидки на процедуры":

table = RequestsExecuter.Request\_24();

dataGridView1.DataSource = table;

break;

case "Использование подзапроса для выбора владельцев, у которых есть более одного животного":

table = RequestsExecuter.Request\_25();

dataGridView1.DataSource = table;

break;

case "Статистика по ветеринарным клиникам(мод. представление через триггер)":

table = RequestsExecuter.Request\_26();

dataGridView1.DataSource = table;

break;

}

}

catch(Exception ex)

{

MessageBox.Show($"{ex.Message}; Возможно вы ввели не верный параметр");

}

}

string positionBox;

private void posBox\_SelectedIndexChanged(object sender, EventArgs e)

{

positionBox = posBox.GetItemText(posBox.SelectedItem);

}

string procedureType;

private void procedureTypeBox\_SelectedIndexChanged(object sender, EventArgs e)

{

procedureType = procedureTypeBox.GetItemText(procedureTypeBox.SelectedItem);

}

//Запрос 1

private void button3\_Click(object sender, EventArgs e)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

var table = new DataTable();

string sql = @"

WITH cte AS (

SELECT

[Ветеринарные клиники].[Название пункта],

[Ветеринарные клиники].[Номер регистрационного пункта],

[Районы].[Район города],

[Сотрудники].[Код ветеринарной клиники],

COUNT(\*) AS Количество\_процедур,

DENSE\_RANK() OVER (PARTITION BY [Районы].[Район города] ORDER BY COUNT(\*) DESC, [Ветеринарные клиники].[Название пункта]) AS Ранг\_по\_району,

ROW\_NUMBER() OVER (ORDER BY COUNT(\*) DESC, [Ветеринарные клиники].[Название пункта]) AS Ранг\_по\_городу

FROM [dbo].[Процедуры]

INNER JOIN [dbo].[Сотрудники] ON [Процедуры].[Код сотрудника] = [Сотрудники].[Код сотрудника]

INNER JOIN [dbo].[Ветеринарные клиники] ON [Сотрудники].[Код ветеринарной клиники] = [Ветеринарные клиники].[Код ветеринарной клинки]

INNER JOIN [dbo].[Районы] ON [Ветеринарные клиники].[Код район города] = [Районы].[Код района]

GROUP BY

[Ветеринарные клиники].[Название пункта],

[Ветеринарные клиники].[Номер регистрационного пункта],

[Районы].[Район города],

[Сотрудники].[Код ветеринарной клиники]

)

SELECT

cte1.[Район города],

STUFF((SELECT ', ' + [Название пункта]

FROM cte

WHERE [Район города] = cte1.[Район города] AND [Ранг\_по\_району] = 1

FOR XML PATH('')), 1, 2, '') AS [Лучшие пункты по району],

STUFF((SELECT TOP 3 ', ' + [Название пункта] + ' (' + CAST([Количество\_процедур] AS VARCHAR) + ')'

FROM cte

WHERE [Ранг\_по\_городу] <= 3

ORDER BY [Ранг\_по\_городу]

FOR XML PATH('')), 1, 2, '') AS [Топ 3 лучших пункта по городу],

cte1.[Код ветеринарной клиники],

cte1.[Номер регистрационного пункта],

cte1.[Количество\_процедур]

FROM

cte AS cte1

WHERE

cte1.[Ранг\_по\_району] = 1

GROUP BY

cte1.[Район города], cte1.[Код ветеринарной клиники], cte1.[Номер регистрационного пункта], cte1.[Количество\_процедур]";

SqlCommand command = new SqlCommand(sql, myConnection);

SqlDataAdapter adapter = new SqlDataAdapter(command);

table = new DataTable();

adapter.Fill(table);

dataGridView1.DataSource = table;

}

//Запрос 2

private void button1\_Click(object sender, EventArgs e)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

var table = new DataTable();

#region temp

//Общий запрос

// string sql = @"WITH cte AS (

// SELECT

// [Veterinary Clinic].[dbo].[Сотрудники].[Имя],

// [Veterinary Clinic].[dbo].[Сотрудники].[Фамилия],

// [Veterinary Clinic].[dbo].[Сотрудники].[Отчество],

// [Veterinary Clinic].[dbo].[Ветеринарные клиники].[Название пункта],

// [Veterinary Clinic].[dbo].[Районы].[Район города],

// COUNT(\*) AS [Количество процедур],

// DENSE\_RANK() OVER (PARTITION BY [Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код ветеринарной клинки], [Veterinary Clinic].[dbo].[Районы].[Код района] ORDER BY COUNT(\*) DESC) AS [Ранг по пункту],

// DENSE\_RANK() OVER (PARTITION BY [Veterinary Clinic].[dbo].[Районы].[Код района] ORDER BY COUNT(\*) DESC) AS [Ранг по району],

// DENSE\_RANK() OVER (ORDER BY COUNT(\*) DESC) AS [Ранг по городу]

// FROM [Veterinary Clinic].[dbo].[Процедуры]

// INNER JOIN [Veterinary Clinic].[dbo].[Сотрудники] ON [Процедуры].[Код сотрудника] = [Сотрудники].[Код сотрудника]

// INNER JOIN [Veterinary Clinic].[dbo].[Ветеринарные клиники] ON [Сотрудники].[Код ветеринарной клиники] = [Ветеринарные клиники].[Код ветеринарной клинки]

// INNER JOIN [Veterinary Clinic].[dbo].[Районы] ON [Ветеринарные клиники].[Код район города] = [Районы].[Код района]

// GROUP BY

// [Veterinary Clinic].[dbo].[Сотрудники].[Имя],

// [Veterinary Clinic].[dbo].[Сотрудники].[Фамилия],

// [Veterinary Clinic].[dbo].[Сотрудники].[Отчество],

// [Veterinary Clinic].[dbo].[Ветеринарные клиники].[Название пункта],

// [Veterinary Clinic].[dbo].[Районы].[Район города],

// [Veterinary Clinic].[dbo].[Ветеринарные клиники].[Код ветеринарной клинки],

// [Veterinary Clinic].[dbo].[Районы].[Код района]

//)

//SELECT

// [По пункту].[Название пункта],

// [По пункту].[Район города],

// [По пункту].[Максимальное количество процедур],

// [По пункту].[Сотрудник с максимальным количеством процедур]

//FROM

// (

// SELECT

// [cte].[Название пункта],

// [cte].[Район города],

// [cte].[Имя] + ' ' + [cte].[Фамилия] + ' ' + [cte].[Отчество] AS [Сотрудник с максимальным количеством процедур],

// [cte].[Количество процедур],

// MAX([cte].[Количество процедур]) OVER (PARTITION BY [cte].[Название пункта]) AS [Максимальное количество процедур]

// FROM

// [cte]

// WHERE

// [cte].[Ранг по пункту] = 1

// ) AS [По пункту]

// FULL OUTER JOIN

// (

// SELECT

// [cte].[Название пункта],

// [cte].[Район города],

// MAX([cte].[Количество процедур]) AS [Максимальное количество процедур]

// FROM

// [cte]

// WHERE

// [cte].[Ранг по району] = 1

// GROUP BY

// [cte].[Название пункта],

// [cte].[Район города]

// ) AS [По району]

// ON [По пункту].[Название пункта] = [По району].[Название пункта] AND [По пункту].[Район города] = [По району].[Район города]

// FULL OUTER JOIN

// (

// SELECT

// [cte].[Имя] + ' ' + [cte].[Фамилия] + ' ' + [cte].[Отчество] AS [Сотрудник с максимальным количеством процедур],

// MAX([cte].[Количество процедур]) AS [Максимальное количество процедур]

// FROM

// [cte]

// WHERE

// [cte].[Ранг по городу] = 1

// GROUP BY

// [cte].[Имя],

// [cte].[Фамилия],

// [cte].[Отчество]

// ) AS [По городу]

// ON [По пункту].[Сотрудник с максимальным количеством процедур] = [По городу].[Сотрудник с максимальным количеством процедур]";

//По пунктам

// string sql = @"SELECT

// [Ветеринарные клиники].[Номер регистрационного пункта],

// [Ветеринарные клиники].[Код ветеринарной клинки],

// [Сотрудники].[Код сотрудника],

// [Ветеринарные клиники].[Название пункта],

// [Сотрудники].[Имя],

// [Сотрудники].[Фамилия],

// [Сотрудники].[Отчество],

// COUNT(\*) AS [Количество процедур]

//FROM [Процедуры]

//INNER JOIN [Сотрудники] ON [Процедуры].[Код сотрудника] = [Сотрудники].[Код сотрудника]

//INNER JOIN [Ветеринарные клиники] ON [Сотрудники].[Код ветеринарной клиники] = [Ветеринарные клиники].[Код ветеринарной клинки]

//GROUP BY

// [Ветеринарные клиники].[Номер регистрационного пункта],

// [Ветеринарные клиники].[Код ветеринарной клинки],

// [Ветеринарные клиники].[Название пункта],

// [Сотрудники].[Код сотрудника],

// [Сотрудники].[Имя],

// [Сотрудники].[Фамилия],

// [Сотрудники].[Отчество]";

#endregion

//По ветеринарным клиникам

// string sql = @"WITH CTE AS (

// SELECT

// [Ветеринарные клиники].[Код ветеринарной клинки],

// [Сотрудники].[Код сотрудника],

// [Сотрудники].[Фамилия],

// [Сотрудники].[Имя],

// [Сотрудники].[Отчество],

// [Ветеринарные клиники].[Номер регистрационного пункта],

// [Ветеринарные клиники].[Название пункта],

// COUNT(\*) AS [Количество процедур],

// ROW\_NUMBER() OVER(

// PARTITION BY [Ветеринарные клиники].[Код ветеринарной клинки]

// ORDER BY COUNT(\*) DESC, [Сотрудники].[Фамилия], [Сотрудники].[Имя], [Сотрудники].[Отчество]

// ) AS [Ранг]

// FROM [Процедуры]

// JOIN [Сотрудники] ON [Процедуры].[Код сотрудника] = [Сотрудники].[Код сотрудника]

// JOIN [Ветеринарные клиники] ON [Сотрудники].[Код ветеринарной клиники] = [Ветеринарные клиники].[Код ветеринарной клинки]

// GROUP BY [Ветеринарные клиники].[Код ветеринарной клинки], [Сотрудники].[Код сотрудника], [Сотрудники].[Фамилия], [Сотрудники].[Имя], [Сотрудники].[Отчество], [Ветеринарные клиники].[Номер регистрационного пункта], [Ветеринарные клиники].[Название пункта]

//)

//SELECT

// [Ветеринарные клиники].[Код ветеринарной клинки],

// [Сотрудники].[Код сотрудника],

// [Сотрудники].[Фамилия],

// [Сотрудники].[Имя],

// [Сотрудники].[Отчество],

// [Ветеринарные клиники].[Номер регистрационного пункта],

// [Ветеринарные клиники].[Название пункта],

// [Количество процедур]

//FROM CTE

//JOIN [Ветеринарные клиники] ON [CTE].[Код ветеринарной клинки] = [Ветеринарные клиники].[Код ветеринарной клинки]

//JOIN [Сотрудники] ON [CTE].[Код сотрудника] = [Сотрудники].[Код сотрудника]

//WHERE [Ранг] = 1;";

//По районам

//string sql = "WITH CTE AS ( " +

// "SELECT [Районы].[Район города], [Сотрудники].[Код сотрудника], [Сотрудники].[Фамилия], [Сотрудники].[Имя], [Сотрудники].[Отчество], [Ветеринарные клиники].[Номер регистрационного пункта], [Ветеринарные клиники].[Название пункта], " +

// "COUNT(\*) AS [Количество процедур], " +

// "ROW\_NUMBER() OVER(PARTITION BY [Районы].[Район города] ORDER BY COUNT(\*) DESC, [Сотрудники].[Фамилия], [Сотрудники].[Имя], [Сотрудники].[Отчество]) AS [Ранг] " +

// "FROM [Процедуры] " +

// "JOIN [Сотрудники] ON [Процедуры].[Код сотрудника] = [Сотрудники].[Код сотрудника] " +

// "JOIN [Ветеринарные клиники] ON [Сотрудники].[Код ветеринарной клиники] = [Ветеринарные клиники].[Код ветеринарной клинки] " +

// "JOIN [Районы] ON [Ветеринарные клиники].[Код район города] = [Районы].[Код района] " +

// "GROUP BY [Районы].[Район города], [Сотрудники].[Код сотрудника], [Сотрудники].[Фамилия], [Сотрудники].[Имя], [Сотрудники].[Отчество], [Ветеринарные клиники].[Номер регистрационного пункта], [Ветеринарные клиники].[Название пункта] " +

// ") " +

// "SELECT [Район города], [Код сотрудника], [Фамилия], [Имя], [Отчество], [Номер регистрационного пункта], [Название пункта], [Количество процедур] " +

// "FROM CTE " +

// "WHERE [Ранг] = 1";

//Вне зависимости от района

//string sql = "WITH CTE AS ( " +

// "SELECT [Районы].[Район города], [Сотрудники].[Код сотрудника], [Сотрудники].[Фамилия], [Сотрудники].[Имя], [Сотрудники].[Отчество], [Ветеринарные клиники].[Номер регистрационного пункта], [Ветеринарные клиники].[Название пункта], " +

// "COUNT(\*) AS [Количество процедур], " +

// "ROW\_NUMBER() OVER(PARTITION BY [Районы].[Район города] ORDER BY COUNT(\*) DESC, [Сотрудники].[Фамилия], [Сотрудники].[Имя], [Сотрудники].[Отчество]) AS [Ранг] " +

// "FROM [Процедуры] " +

// "JOIN [Сотрудники] ON [Процедуры].[Код сотрудника] = [Сотрудники].[Код сотрудника] " +

// "JOIN [Ветеринарные клиники] ON [Сотрудники].[Код ветеринарной клиники] = [Ветеринарные клиники].[Код ветеринарной клинки] " +

// "JOIN [Районы] ON [Ветеринарные клиники].[Код район города] = [Районы].[Код района] " +

// "GROUP BY [Районы].[Район города], [Сотрудники].[Код сотрудника], [Сотрудники].[Фамилия], [Сотрудники].[Имя], [Сотрудники].[Отчество], [Ветеринарные клиники].[Номер регистрационного пункта], [Ветеринарные клиники].[Название пункта] " +

// ") " +

// "SELECT TOP 1 [Район города], [Код сотрудника], [Фамилия], [Имя], [Отчество], [Номер регистрационного пункта], [Название пункта], [Количество процедур] " +

// "FROM CTE " +

// "ORDER BY [Количество процедур] DESC";

string sql = @"WITH CTE AS (

SELECT

[Ветеринарные клиники].[Код ветеринарной клинки],

[Сотрудники].[Код сотрудника],

[Сотрудники].[Фамилия],

[Сотрудники].[Имя],

[Сотрудники].[Отчество],

[Ветеринарные клиники].[Номер регистрационного пункта],

[Ветеринарные клиники].[Название пункта],

[Районы].[Район города],

COUNT(\*) AS [Количество процедур],

ROW\_NUMBER() OVER(

PARTITION BY [Ветеринарные клиники].[Код ветеринарной клинки]

ORDER BY COUNT(\*) DESC, [Сотрудники].[Фамилия], [Сотрудники].[Имя], [Сотрудники].[Отчество]

) AS [Ранг по клинике],

ROW\_NUMBER() OVER(

PARTITION BY [Районы].[Район города]

ORDER BY COUNT(\*) DESC, [Сотрудники].[Фамилия], [Сотрудники].[Имя], [Сотрудники].[Отчество]

) AS [Ранг по району]

FROM [Процедуры]

JOIN [Сотрудники] ON [Процедуры].[Код сотрудника] = [Сотрудники].[Код сотрудника]

JOIN [Ветеринарные клиники] ON [Сотрудники].[Код ветеринарной клиники] = [Ветеринарные клиники].[Код ветеринарной клинки]

JOIN [Районы] ON [Ветеринарные клиники].[Код район города] = [Районы].[Код района]

GROUP BY [Ветеринарные клиники].[Код ветеринарной клинки], [Сотрудники].[Код сотрудника], [Сотрудники].[Фамилия], [Сотрудники].[Имя],

[Сотрудники].[Отчество], [Ветеринарные клиники].[Номер регистрационного пункта], [Ветеринарные клиники].[Название пункта], [Районы].[Район города]

)

SELECT

[Ранг по клинике],

[Ранг по району],

[Район города],

[Код сотрудника],

[Фамилия],

[Имя],

[Отчество],

[Номер регистрационного пункта],

[Название пункта],

[Количество процедур]

FROM CTE

WHERE

[Ранг по клинике] = 1 OR

[Ранг по району] = 1 OR

[Ранг по клинике] IS NULL

ORDER BY [Количество процедур] DESC;";

SqlCommand command = new SqlCommand(sql, myConnection);

SqlDataAdapter adapter = new SqlDataAdapter(command);

table = new DataTable();

adapter.Fill(table);

dataGridView1.DataSource = table;

}

//Запрос 3

private void button2\_Click(object sender, EventArgs e)

{

string connectionString = @"data source=(localdb)\MSSQLLocalDB;Initial Catalog=Veterinary Clinic;Integrated Security=True;";

SqlConnection myConnection = new SqlConnection(connectionString);

myConnection.Open();

var table = new DataTable();

#region stuff

// string sql = @"WITH CTE AS (

//SELECT

//[Ветеринарные клиники].[Код ветеринарной клинки],

//[Ветеринарные клиники].[Номер регистрационного пункта],

//[Ветеринарные клиники].[Название пункта],

//[Районы].[Район города],

//SUM([Процедуры].[Цена процедуры]) AS [Доходы]

//FROM [Процедуры]

//JOIN [Сотрудники] ON [Процедуры].[Код сотрудника] = [Сотрудники].[Код сотрудника]

//JOIN [Ветеринарные клиники] ON [Сотрудники].[Код ветеринарной клиники] = [Ветеринарные клиники].[Код ветеринарной клинки]

//JOIN [Районы] ON [Ветеринарные клиники].[Код район города] = [Районы].[Код района]

//WHERE YEAR([Процедуры].[Дата оказания помощи животному]) = '2013'

//GROUP BY [Ветеринарные клиники].[Код ветеринарной клинки], [Ветеринарные клиники].[Номер регистрационного пункта], [Ветеринарные клиники].[Название пункта], [Районы].[Район города]

//)

//SELECT

//[Район города],

//[Номер регистрационного пункта],

//[Название пункта],

//[Доходы]

//FROM CTE

//ORDER BY [Доходы] DESC;

//WITH CTE AS (

//SELECT

//[Районы].[Район города],

//SUM([Процедуры].[Цена процедуры]) AS [Доходы]

//FROM [Процедуры]

//JOIN [Сотрудники] ON [Процедуры].[Код сотрудника] = [Сотрудники].[Код сотрудника]

//JOIN [Ветеринарные клиники] ON [Сотрудники].[Код ветеринарной клиники] = [Ветеринарные клиники].[Код ветеринарной клинки]

//JOIN [Районы] ON [Ветеринарные клиники].[Код район города] = [Районы].[Код района]

//WHERE YEAR([Процедуры].[Дата оказания помощи животному]) = '2013'

//GROUP BY [Районы].[Район города]

//)

//SELECT

//[Район города],

//[Доходы]

//FROM CTE

//ORDER BY [Доходы] DESC;

//WITH CTE AS (

//SELECT

//SUM([Процедуры].[Цена процедуры]) AS [Доходы]

//FROM [Процедуры]

//WHERE YEAR([Процедуры].[Дата оказания помощи животному]) = '2013'

//)

//SELECT

//[Доходы]

//FROM CTE;";

#endregion

if (int.TryParse(yearBox.Text, out int year))

{

string sql = $@"WITH CTE\_Clinics AS (

SELECT

[Ветеринарные клиники].[Код ветеринарной клинки],

[Ветеринарные клиники].[Номер регистрационного пункта],

[Ветеринарные клиники].[Название пункта],

[Районы].[Район города],

SUM([Процедуры].[Цена процедуры]) AS [Доходы],

DENSE\_RANK() OVER (ORDER BY SUM([Процедуры].[Цена процедуры]) DESC) AS [Ранг клиники]

FROM [Процедуры]

JOIN [Сотрудники] ON [Процедуры].[Код сотрудника] = [Сотрудники].[Код сотрудника]

JOIN [Ветеринарные клиники] ON [Сотрудники].[Код ветеринарной клиники] = [Ветеринарные клиники].[Код ветеринарной клинки]

JOIN [Районы] ON [Ветеринарные клиники].[Код район города] = [Районы].[Код района]

WHERE YEAR([Процедуры].[Дата оказания помощи животному]) = '{yearBox.Text}'

GROUP BY [Ветеринарные клиники].[Код ветеринарной клинки], [Ветеринарные клиники].[Номер регистрационного пункта], [Ветеринарные клиники].[Название пункта], [Районы].[Район города]

), CTE\_Districts AS (