

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР
ГОУ ВПО «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ»

Факультет КНТ
Кафедра ПИ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту по дисциплине
«Программирование систем с Базами Данных»
по теме: Информационная система «Нефтеперерабатывающие заводы»

Руководители:

асс. каф. ПИ

Щедрин С.В.

асс. каф. ПИ

Ногтев Е.А.

Выполнил:

ст. гр. ПИ-17в

Петренко Д.А.

Донецк – 2020

РЕФЕРАТ

Пояснительная записка к курсовому проекту содержит: 77 страниц, 45 рисунков, 6 таблиц, 4 источника, 6 приложений.

Цель работы – закрепить практические навыки разработки реляционных баз данных (БД) и проектирования ПО, научиться осуществлять нормализацию таблиц, получить углубленные знания языка структурированных запросов (SQL), приобрести практические навыки самостоятельной разработки основных компонентов прикладного программного обеспечения.

Для достижения поставленной цели курсового проекта необходимо решить следующие задачи:

- выполнить анализ предметной области задания курсовой работы;
- осуществить проектирование модели базы данных;
- разработать программный продукт;
- осуществить тестирование спроектированного программного комплекса.

Объект исследования – информационная система лечебных заведений.

Результат работы – разработанная СУБД обеспечения и учета заказов нефтеперерабатывающих заводов на языке C#, с использованием синтаксиса PostgreSQL.

БАЗА ДАННЫХ, СУБД, POSTGRESQL, СВЯЗИ, ЗАПРОС,
РЕЛЯЦИОННАЯ БАЗА ДАННЫХ

СОДЕРЖАНИЕ

Введение.....	4
1 описание предметной области, постановка задачи.....	6
2 Обоснование выбора СУБД, описание возможностей СУБД.....	7
3 Обоснование выбора инструментальных средств для написания клиентской части, проектирование структуры ПО.....	11
3.1 Невизуальные компоненты для работы с данным.....	13
3.2 Визуальные компоненты для работы с данными	13
3.3 Разработка шаблонов приложений для работы с шаблонами базы данных	15
4 Проектирование базы данных в выбранной СУБД.....	18
4.1 Проектирование концептуальной модели БД.....	18
4.2 Создание таблиц, доменов, индексов	19
4.3 Разработка триггеров	22
4.4 Проектирование запросов к базе данных	26
4.5 Создание представлений и хранимых процедур, функций	27
5 Разработка клиентского приложения.....	51
5.1 Формы и компоненты для работы с основными таблицами	51
5.2 Формы и компоненты для работы со справочниками.....	51
5.3 Формы и компоненты для отображения результатов запросов	52
6 Тестирование разработанной информационной системы.....	54
ЗАКЛЮЧЕНИЕ.....	60
Перечень ссылок.....	61
Приложение А Техническое задание.....	62
Приложение Б Листинг шаблонов.....	63
Приложение В Листинг серверного приложения.....	68
Приложение Г Листинг клиентского приложения.....	74
Приложение Д Руководство пользователя.....	81
Приложение Ж Руководство администратора.....	82

ВВЕДЕНИЕ

На сегодняшний день применение баз данных приобрело весьма важное значение для многих организаций, которые для упрощения своей работы применяют компьютерные технологии.

База данных (БД) — это организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации, преимущественно больших объемов. Базы данных активно используются для проектирования практически любых пользовательских приложений.

В контексте баз данных стоит рассмотреть понятие СУБД. Система управления базами данных (СУБД) — это комплекс программных средств, необходимых для создания структуры новой базы, ее наполнения, редактирования содержимого и отображения информации. Наиболее распространенными СУБД являются MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

Это СУБД примеры типа клиент-сервер, именно такие СУБД встречаются чаще всего в контексте понятия хостинга. Их особенности:

- расположение СУБД на сервере с базами данных;
- непосредственный доступ к БД;
- централизованная обработка клиентских запросов на обработку данных;
- высокий уровень надежности, доступности и безопасности;
- повышенная нагрузка на сервер.

В свою очередь, для удобства работы с СУБД используются специальные веб-приложения, которые позволяют посредством графического интерфейса выполнять администрирование сервера баз данных, запускать специальные команды, а также работать с контентом таблиц и баз данных — действия, которые при отсутствии веб-приложения подлежат выполнению средствами консоли. Примеры: phpMyAdmin используется для администрирования СУБД MySQL, pgAdmin — для PostgreSQL.

Эффективное хранение, обработка и взаимодействие с данными – главные преимущества использования БД сегодня. Один из способов повышения эффективности обработки данных — организовать их эффективное хранение и получение. Самый распространенный подход к хранению данных на сегодня — использовать реляционную базу данных.

Цель исследования состоит в формировании и закреплении навыков применения, комплекса методов управления базами данных, серверами баз данных и создания клиентских приложений для них.

Объектом называется элемент информационной системы, сведения о котором хранятся в базе данных. Иногда объект также называют сущностью (от англ. entity).

Классом объектов называют их совокупность, обладающую одинаковым набором свойств.

Атрибут - это информационное отображение свойств объекта. Каждый объект характеризуется некоторым набором атрибутов.

Ключевым элементом данных называются такой атрибут (или группа атрибутов), который позволяет определить значения других элементов-данных.

Запись данных (от англ. record) - это совокупность значений связанных элементов данных.

Первичный ключ - это атрибут (или группа атрибутов), который уникальным образом идентифицируют каждый экземпляр объекта (запись).

Вторичным ключом называется атрибут (или группа атрибутов), значение которого может повторяться для нескольких записей (экземпляров объекта). Прежде всего, вторичные ключи используются в операциях поиска записей [3].

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ, ПОСТАНОВКА ЗАДАЧИ

В задании курсовой работы требуется разработать базу данных учёта нефтеперерабатывающих заводов. Рассмотрим особенности данной системы, указанные в условии.

Для автоматизации учета работы нефтеперерабатывающих заводов Украины необходима следующая информация: сведения о заводах (название, город, тип собственности (государственный, частный, ЗАО, ООО, ...), год начала функционирования, телефон, страны-поставщики нефти (Казахстан, Россия, Туркмения, Украина,...), виды выпускаемого топлива (вид (дизельное, бензин А76, бензин А80, А92, А95, авиационный керосин,...), годовой объем (в тоннах), цена 1 тонны)), сведения о заказах на нефтепродукты (заказчик (название предприятия, город, телефон), вид топлива, объем (в тоннах), дата заказа).

2 ОБОСНОВАНИЕ ВЫБОРА СУБД, ОПИСАНИЕ ВОЗМОЖНОСТЕЙ СУБД

PostgreSQL не просто реляционная, а объектно-реляционная СУБД. Это даёт ему некоторые преимущества над другими SQL базами данных с открытым исходным кодом, такими как MySQL, MariaDB и Firebird.

Надежность PostgreSQL является проверенным и доказанным фактом и обеспечивается следующими возможностями:

полное соответствие принципам ACID - атомарность, непротиворечивость, изолированность, сохранность данных.

Atomicity - транзакция рассматривается как единая логическая единица, все ее изменения или сохраняются целиком, или полностью откатываются.

Consistency - транзакция переводит базу данных из одного непротиворечивого состояния (на момент старта транзакции) в другое непротиворечивое состояние (на момент завершения транзакции). Непротиворечивым считается состояние базы, когда выполняются все ограничения физической и логической целостности базы данных, при этом допускается нарушение ограничений целостности в течение транзакции, но на момент завершения все ограничения целостности, как физические, так и логические, должны быть соблюдены.

Isolation - изменения данных при конкурентных транзакциях изолированы друг от друга на основе системы версионности

Durability - PostgreSQL заботится о том, что результаты успешных транзакций гарантировано сохраняются на жесткий диск вне зависимости от сбоев аппаратуры.

многоверсионность (Multiversion Concurrency Control, MVCC) используется для поддержания согласованности данных в конкурентных условиях, в то время как в традиционных базах данных используются блокировки. MVCC означает, что каждая транзакция видит копию данных (версию базы данных) на время начала транзакции, несмотря на то, что

состояние базы могло уже измениться. Это защищает транзакцию от несогласованных изменений данных, которые могли быть вызваны (другой) конкурентной транзакцией, и обеспечивает изоляцию транзакций. Основным выигрыш от использования MVCC по сравнению с блокировкой заключается в том, что блокировка, которую ставит MVCC для чтения не конфликтует с блокировкой на запись, и поэтому чтение никогда не блокирует запись и наоборот.

наличие Write Ahead Logging (WAL) - общепринятый механизм протоколирования всех транзакций, что позволяет восстановить систему после возможных сбоев. Основная идея WAL состоит в том, что все изменения должны записываться в файлы на диск только после того, как эти записи журнала, описывающие эти изменения будут и гарантировано записаны на диск. Это позволяет не сбрасывать страницы данных на диск после фиксации каждой транзакции, так как мы знаем и уверены, что сможем всегда восстановить базу данных используя журнал транзакций.

Point in Time Recovery (PITR) - возможность восстановления базы данных (используя WAL) на любой момент в прошлом, что позволяет осуществлять непрерывное резервное копирование кластера PostgreSQL.

Репликация также повышает надежность PostgreSQL. Существует несколько систем репликации, например, Slony, который является свободным и самым используемым решением, поддерживает master-slaves репликацию. Ожидается, что Slony-II будет поддерживать multi-master режим.

Целостность данных является сердцем PostgreSQL. Помимо MVCC, PostgreSQL поддерживает целостность данных на уровне схемы - это внешние ключи (foreign keys), ограничения (constraints).

Модель развития PostgreSQL, которая абсолютно прозрачна для любого, так как все планы, проблемы и приоритеты открыто обсуждаются. Пользователи и разработчики находятся в постоянном диалоге через мэйлинг листы. Все предложения, патчи проходят тщательное тестирование до

принятия их в программное дерево. Большое количество бета-тестеров способствует тестированию версии до релиза и вычищению мелких ошибок.

Открытость кодов PostgreSQL означает их абсолютную доступность для любого, а либеральная BSD лицензия не накладывает никаких ограничений на использование кода.

Производительность PostgreSQL основывается на использовании индексов, интеллектуальном планировщике запросов, тонкой системы блокировок, системой управления буферами памяти и кэширования, превосходной масштабируемости при конкурентной работе.

Поддержка индексов

Стандартные индексы - B-tree, hash, R-tree, GiST (обобщенное поисковое дерево)

Частичные индексы (partial indices)

Функциональные индексы

Планировщик запросов основывается на стоимости различных планов, учитывая множество факторов. Он предоставляет возможность пользователю отлаживать запросы и настраивать систему.

Система блокировок поддерживает блокировки на нижнем уровне, что позволяет сохранять высокий уровень конкурентности при защите целостности данных. Блокировка поддерживается на уровне таблиц и записей. На нижнем уровне, блокировка для общих ресурсов оптимизирована под конкретную ОС и архитектуру.

Управление буферами и кэширование используют сложные алгоритмы для поддержания эффективности использования выделенных ресурсов памяти.

Tablespaces (табличные пространства) позволяют гибкое использование дискового пространства для хранения объектов системы, что также повышает производительность и масштабируемость.

Масштабируемость основывается на описанных выше возможностях. Низкая требовательность PostgreSQL к ресурсам и гибкая система

блокировок обеспечивают его шкалирование, в то время как индексы и управление буферами обеспечивают хорошую управляемость системы даже при высоких нагрузках.

Расширяемость PostgreSQL означает, что пользователь может настраивать систему путем определения новых функций, агрегатов, типов, языков, индексов и операторов. Объектно-ориентированность PostgreSQL позволяет перенести логику приложения на уровень базы данных, что сильно упрощает разработку клиентов, так как вся бизнес логика находится в базе данных. Функции в PostgreSQL однозначно определяются названием, количеством и типами аргументов..

3 ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ДЛЯ НАПИСАНИЯ КЛИЕНТСКОЙ ЧАСТИ, ПРОЕКТИРОВАНИЕ СТРУКТУРЫ ПО

Для реализации клиентской части курсового проекта был выбран язык программирования C# и Framework WPF. Для разработки приложения использовалась среда разработки Microsoft Visual Studio.

Когда говорят C#, нередко имеют в виду технологии платформы .NET (Windows Forms, WPF, ASP.NET, Xamarin). И, наоборот, когда говорят .NET, нередко имеют в виду C#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире.

Как-то Билл Гейтс сказал, что платформа .NET - это лучшее, что создала компания Microsoft. Возможно, он был прав. Фреймворк .NET представляет мощную платформу для создания приложений. Можно выделить следующие ее основные черты:

Поддержка нескольких языков. Основой платформы является общезыковая среда исполнения Common Language Runtime (CLR), благодаря чему .NET поддерживает несколько языков: наряду с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. При компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) - своего рода ассемблер платформы .NET. Поэтому мы можем сделать отдельные модули одного приложения на отдельных языках.

Кроссплатформенность. .NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент .NET Core поддерживается на большинстве современных ОС Windows, MacOS, Linux. Используя различные технологии на платформе .NET, можно разрабатывать приложения на языке C# для самых разных платформ - Windows, MacOS, Linux, Android, iOS, Tizen.

Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. И какое бы приложение мы не собирались писать на С# - текстовый редактор, чат или сложный веб-сайт - так или иначе мы задействуем библиотеку классов .NET.

Разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов являются основой для целого стека технологий, которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных в этом стеке технологий предназначена технология ADO.NET и Entity Framework Core. Для построения графических приложений с богатым насыщенным интерфейсом - технология WPF и UWP, для создания более простых графических приложений - Windows Forms. Для разработки мобильных приложений - Xamarin. Для создания веб-сайтов - ASP.NET и т.д.

Также еще следует отметить такую особенность языка С# и фреймворка .NET, как автоматическая сборка мусора. А это значит, что нам в большинстве случаев не придется, в отличие от С++, заботиться об освобождении памяти. Вышеупомянутая общеязыковая среда CLR сама вызовет сборщик мусора и очистит память..

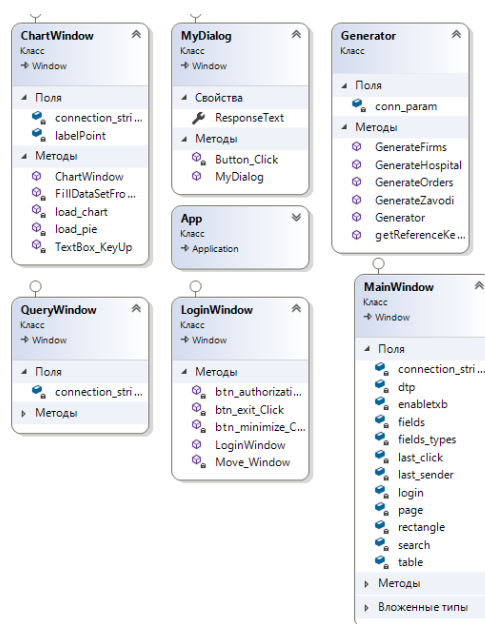


Рисунок 3.1 – Диаграмма классов

3.1 Невизуальные компоненты для работы с данным

Для создания проекта необходимо первым делом скачать драйвер PostgreSQL и добавить его в проект в качестве библиотеки “Npgsql”, что позволит в дальнейшем работать с базами данных.

3.2 Визуальные компоненты для работы с данными

Для отображения данных из таблиц был выбран элемент управления DataGridView из WindowsForms, который является достаточно гибким и позволяет автоматизировать вывод информации пользователю. Пример использования данного элемента управления для отображения таблицы из базы данных представлен на рисунках 3.2 и 3.3.



id_plants	название	тип	город	страна
26	Schuppe LLC	ЗАО	Хабаровск	Украина
27	Torphy - Hartmann	государств...	Сочи	Казахста
28	Balisteri Group	частный	Тольятти	Украина
29	Collins Group	частный	Курск	Казахста
30	Sanford - Feil	ЗАО	Оренбург	Турекме
31	Daniel - Gulowski	частный	Оренбург	ДНР
32	Boyle - Dickinson	ЗАО	Владивосток	Россия
33	Koss - Hettinger	государств...	Санкт-Петербург	Украина
34	Monahan, Windler and Kiehn	частный	Владимир	Украина
35	Pfannerstill - Jones	государств...	Москва	Казахста
36	Purdy - Gulowski	государств...	Махачкала	Казахста
37	Kassulke - Deckow	частный	Красноярск	Турекме
38	Jaskolski, Schneider and Batz	государств...	Магнитогорск	Турекме
39	Dicki, Harris and Ebert	частный	Тольятти	Россия
40	Stiedemann - Gibson	государств...	Красноярск	Казахста
41	Nitzsche - Wunsch	ЗАО	Новокузнецк	Турекме

Рисунок 3.2 – Пример использования DataGridView для отображения данных таблицы

```

private void btn_plants_Click(object sender, RoutedEventArgs e)
{
    last_click = btn_plants_Click;
    last_sender = sender;
    move_rect(sender);

    table = "Заводы";
    fields = new string[]
    {
        "id_plants", "название", "город", "id_type", "год", "телефон", "id_country", "id_fuel", "объём", "цена"
    };

    fields_types = new string[]
    {
        "int", "string", "string", "int", "int", "string", "int", "int", "int", "int"
    };

    FillGridFromDataSet(FillDataSetFromDB(generate_select(fields, table)));

    dataGridview1.Columns["id_type"].Visible = false;
    dataGridview1.Columns["id_country"].Visible = false;
    dataGridview1.Columns["id_fuel"].Visible = false;
    addComboBoxColumn("cmb_type", "тип", "тип собственности", "id_type", "тип", 2);
    addComboBoxColumn("cmb_country", "страна", "страны", "id_country", "страна", 4);
    addComboBoxColumn("cmb_fuel", "топливо", "вид топлива", "id_fuel", "топливо", 5);
    setComboBoxColumn("cmb_type", "id_type");
    setComboBoxColumn("cmb_country", "id_country");
    setComboBoxColumn("cmb_fuel", "id_fuel");
    fix_cmb_width();
}

```

Рисунок 3.3 – Пример использования DataGridView для отображения данных таблицы (код программы)

Для отображения справочников при добавлении/редактировании записей в базе данных был использован элемент управления ComboBox, который обеспечивает удобное отображение и выбор данных из справочника. Примеры использования данного элемента управления приведен на рисунках 3.4 и 3.5.

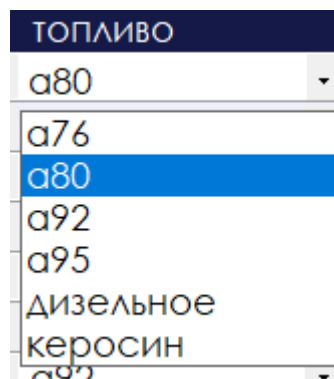


Рисунок 3.4 – Пример использования ComboBox для отображения справочников

```

void addComboBoxColumn(string name_cmbCol, string headerText, string table,
    string ValueMember, string DisplayMember, int Xindex)
{
    System.Windows.Forms.DataGridViewComboBoxColumn cmbCol = new System.Windows.Forms.DataGridViewComboBoxColumn();
    cmbCol.HeaderText = headerText;
    cmbCol.Name = name_cmbCol;

    string fields = $"";
    cmbCol.DataSource = fillComboBox(fields, table, DisplayMember);
    cmbCol.ValueMember = ValueMember;

    cmbCol.DisplayMember = DisplayMember;
    cmbCol.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

    dataGridView1.Columns.Add(cmbCol);
    dataGridView1.Columns[name_cmbCol].DisplayIndex = Xindex;
    dataGridView1.Columns[Xindex].AutoSizeMode = System.Windows.Forms.DataGridViewAutoSizeMode.AllCells;
}

```

Рисунок 3.5 – Пример использования ComboBox для отображения справочников (код программы)

3.3 Разработка шаблонов приложений для работы с шаблонами базы данных

Данный программный продукт оснащён простым, интуитивно понятным пользовательским интерфейсом. В программе были созданы окна авторизации пользователя, работы с таблицами, работы с запросами и графиками.

Начало работы происходит в окне авторизации. Если авторизация происходит успешно, появляется окно для работы (добавление, удаление, изменение записей) с таблицами. Работа с таблицами происходит на основе разрешений конкретного пользователя.

Из окна для работы с таблицами пользователь может перейти на окно с перечнем запросов. В нём пользователь может выполнить запрос, получить результат, экспортировать результаты в excel, просмотреть некоторые итоговые запросы в виде графиков.

Рассмотрим особенности каждой формы на рисунках 3.6 - 3.10.

Авторизуйтесь

Имя пользователя

Дима

Пароль

•••••

Авторизоваться

Рисунок 3.6 – Форма авторизации

id_client	название	город	телефон
1	Weber, Zieme and Schamberger	Хабаровск	+380715469278
2	Cronin, Waters and Barrows	Саратов	+380716770904
3	Ritchie Inc	Омск	+380717056067
4	Smith and Sons	Тула	+380715242123
5	Rutherford, Kris and Farrell	Москва	+380718071374
6	Schmidt, Macejkovic and Runolfsson	Ростов-на-Дону	+380711226522
7	Kihn, Parisian and Mohr	Екатеринбург	+380715783278
8	Brown, Mertz and Armstrong	Тюмень	+380716097669
9	Bins, Bartell and Reichert	Белгород	+380710504319
10	Friesen LLC	Тольятти	+380713945062
11	Auer, MacGyver and Dach	Томск	+380719921460
12	Gorczyan Group	Томск	+380716280243
13	Huel, Howell and Okuneva	Краснодар	+380710344943
14	Schoen and Sons	Астрахань	+380710809102
15	Torphy and Sons	Краснодар	+380716718125
16	Rutherford - Brakus	Санкт-Петербург	+380715846666
17	Stamm, West and Lona	Тольятти	+380714716511

Заводы Вид топлива Заказы **Заказчики** Тип собственности Страны Запросы

Рисунок 3.7 – Форма работы с таблицами

Запросы			Экспорт в Excel
название	дата	объём	
Bayer - Zulauf	26.03.2020 6:32:34	803	Заказы
Russel Group	30.12.2015 11:01:12	429	Заводы
Hartmann, Osinski and Fay	16.08.1990 12:38:46	430	Заказчики
Cronin - Kilback	09.09.1987 22:02:34	670	Заказы объём больше
Maggio, Predovic and Mraz	02.04.2003 17:49:24	683	Заказы объём меньше
Price - Dibbert	04.12.1991 17:01:26	872	Заказы за пр. месяцы
Marvin - Harris	02.07.1994 16:10:40	463	Заказы за пр. года
Thompson Inc	01.12.1998 9:33:55	787	Заводы без заказов
Altenwerth, Botsford and Gislason	24.05.1995 18:39:03	766	Клиенты без заказов
Jacobs - Morar	30.08.2011 22:42:25	756	Клиенты без заказов v2
Walter, Schaefer and Kuhic	05.11.2007 12:43:10	465	ср. объём заказов
Frami - Quitzon	09.06.2013 9:21:43	354	
Trantow - Metz	21.07.2009 12:05:49	612	
Zieme - Goyette	02.08.2002 1:14:47	400	
Hodkiewicz, Barton and Zulauf	29.04.2005 21:05:01	491	
Watsica - Bosco	12.01.2008 8:52:35	434	
Ryan LLC	17.02.1991 9:19:51	452	

Рисунок 3.8 – Форма запросов

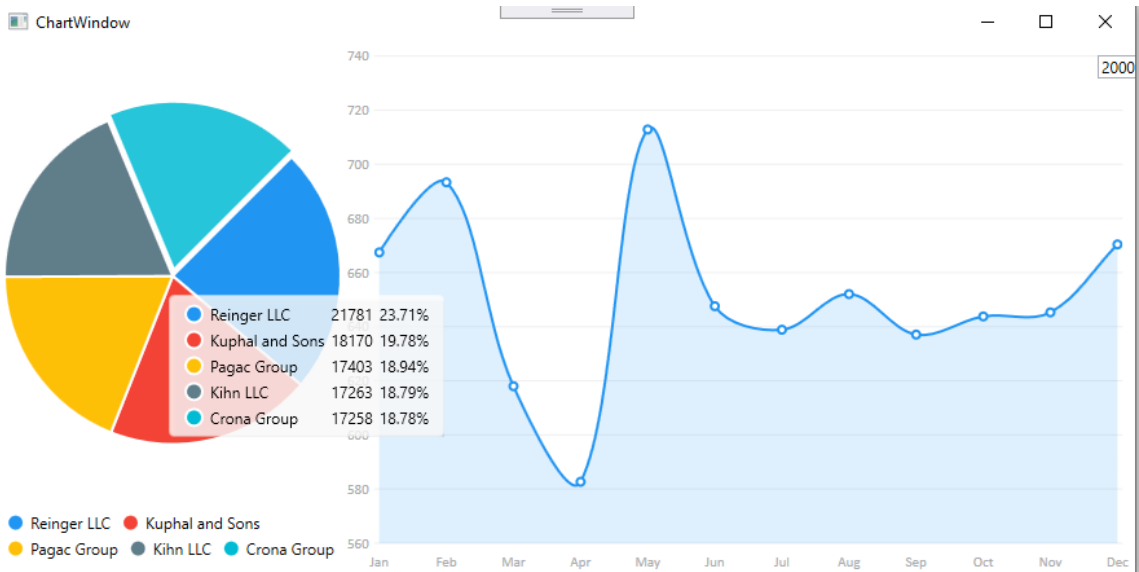


Рисунок 3.9 – Форма с графиками

Месяцы

Введите за сколько последних месяцев

Подтвердить

Рисунок 3.10 – Окно ввода параметров запроса

4 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ В ВЫБРАННОЙ СУБД

4.1 Проектирование концептуальной модели БД

Процесс проектирования БД с использованием метода нормальных форм (НФ) является итерационным и заключается в последовательном переводе отношения из 1НФ в НФ более высокого порядка по определенным правилам. Каждая следующая НФ ограничивается определенным типом функциональных зависимостей и устранением соответствующих аномалий при выполнении операций над отношениями БД, а также сохранении свойств, предшествующих НФ.

Нормализация базы данных сводит к минимуму количество избыточной информации. Ее целью является сохранять данные только один раз, но в нужном месте. Нормализованная база данных исключает дублирование и многократное обслуживание данных, а также появление проблем с целостностью данных, возникающих при повторном вводе одинаковых данных. Первоначально доктором Эдгаром Коддом были определены только 3 нормальные формы. Дальнейшая разработка реляционной теории привела к появлению еще нескольких форм и на данный момент их насчитывается 6.

На практике соответствие базы данных правилам 3-ей нормальной формы вполне достаточно.

В результате выполнения нормализации базы данных (приведение в 3НФ) были выделены следующие таблицы:

1. Заказы (таблица 1.1);
2. Заводы (таблица 1.2);
3. Заказчики (таблица 1.3);
4. Страны (таблица 1.4);
5. Тип собственности (таблица 1.5);
6. вид топлива (таблица 1.6);

4.2 Создание таблиц, доменов, индексов

Таблица 4.1 – Заказы

№	Поле	Тип	Размер	Описание
1	id_order	Числовой	4	Уникальный идентификатор
2	id_client	Числовой	4	Уникальный идентификатор
3	id_fuel	Числовой	4	Уникальный идентификатор
4	объем	Числовой	4	Объем
5	дата	Дата	4	Дата составления заказа
6	id_plants	Числовой	4	Уникальный идентификатор
7	login	Текстовый	64	Для Row level security

Таблица 4.2 – Заводы

№	Поле	Тип	Размер	Описание
1	id_plants	Числовой	4	Уникальный идентификатор
2	название	Текстовый	64	Форма выпуска
3	город	Текстовый	64	Город
4	id_type	Числовой	4	Уникальный идентификатор
5	год	Числовой	4	Год создания
6	телефон	Domain	12	Телефон
7	id_country	Числовой	4	Уникальный идентификатор
8	id_fuel	Числовой	4	Уникальный идентификатор
9	объем	Числовой	4	Объем
10	цена	Числовой	4	Цена

Таблица 4.3 – Заказчики

№	Поле	Тип	Размер	Описание
1	id_client	Числовой	4	Уникальный идентификатор
2	название	Текстовый	64	Группа
3	город	Текстовый	64	Город
4	телефон	Domain	12	Телефон

Таблица 4.4 – страны

№	Поле	Тип	Размер	Описание
1	id_country	Числовой	4	Уникальный идентификатор
2	страна	Текстовый	64	Страна

Таблица 4.5 – тип собственности

№	Поле	Тип	Размер	Описание
1	id_type	Числовой	4	Уникальный идентификатор
2	тип	Текстовый	64	Тип собственности

Таблица 4.6 – вид топлива

№	Поле	Тип	Размер	Описание
1	id_fuel	Числовой	4	Уникальный идентификатор
2	топливо	Текстовый	64	Вид топлива

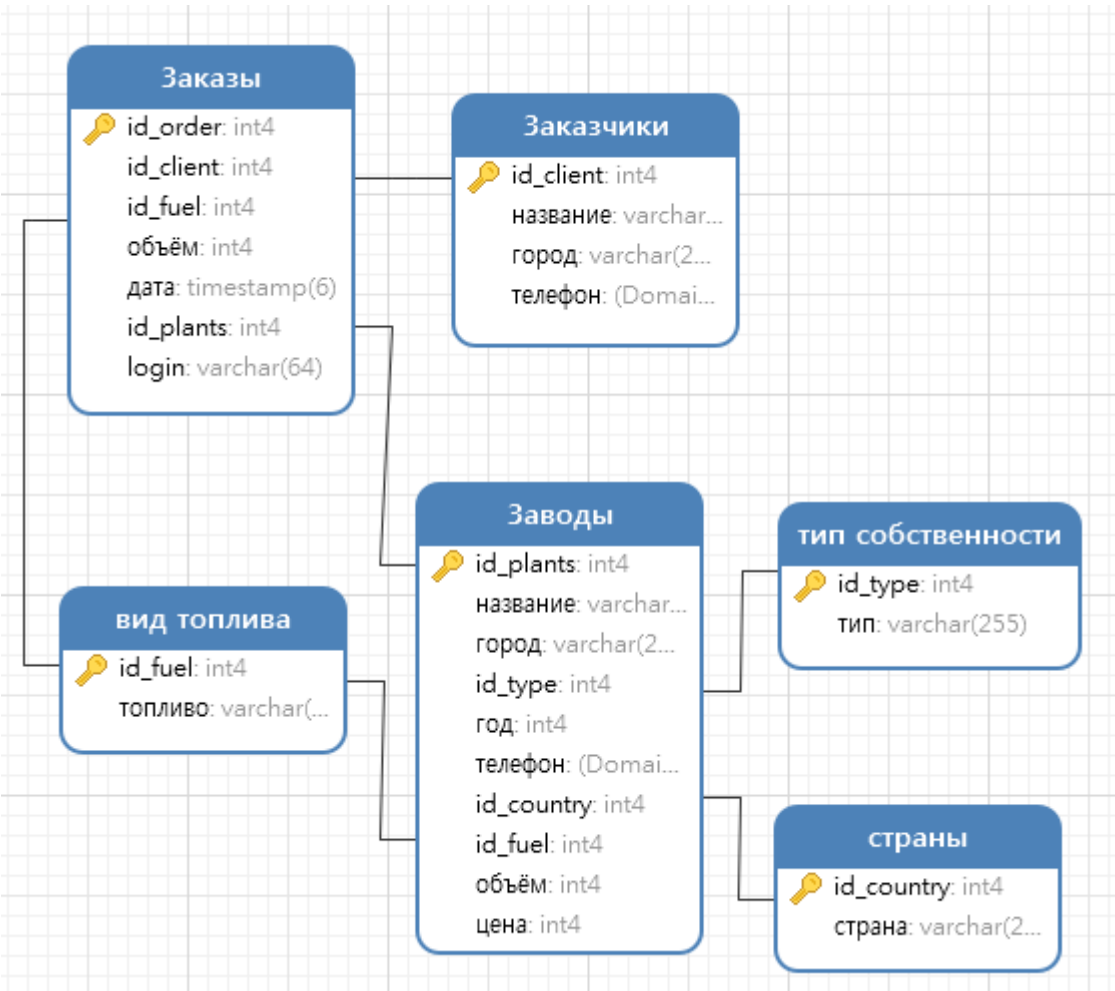


Рисунок 4.1 – Схема базы данных

В ходе разработки базы данных был создан домен `phone`, который представляет собой телефонный номер и имеющий проверку на корректность.

`CREATE DOMAIN` создаёт новый домен. Домен по сути представляет собой тип данных с дополнительными условиями (ограничивающими допустимый набор значений). Владельцем домена становится пользователь его создавший.

Если задаётся имя схемы (например, `CREATE DOMAIN myschema.mydomain ...`), домен создаётся в указанной схеме, в противном случае — в текущей. Имя домена должно быть уникальным среди имён типов и доменов, существующих в этой схеме.

Домены полезны для абстрагирования и вынесения общих характеристик разных полей в единое место для упрощения сопровождения. Например, в нескольких таблицах может присутствовать столбец, содержащий электронный адрес, и для всех требуются одинаковые ограничения `CHECK`, проверяющие синтаксис адреса. В этом случае лучше определить домен, а не задавать для каждой таблицы отдельные ограничения.

Чтобы создать домен, необходимо иметь право `USAGE` для нижележащего типа.

Objects | phone @postgres.public (Bobe...)

Save

General | Checks | Comment | SQL Preview

Underlying Type Category: Base Type

Underlying Type: pg_catalog text

Dimensions: 0

Length: 0

Scale: 0

Collate: pg_catalog default

Default:

☐ Not null

Owner: postgres

Save | Add Check | Delete Check

General | Checks | Comment | SQL Preview

Name	Check	Comment
a_check()	(VALUE ~ '^(\s*)?(\+)?([- _0:]=+)?\d[- _0:]=+]?(\d{10,14})(\s*)?\$':text)	

Рисунок 4.2 – Домен phone

Во время анализа запросов к базе данных возникла необходимость добавить индекс b-tree к полю дата таблицы Заказы для эффективного поиска и фильтрации данных.

Save

General | Advanced | Comment | SQL Preview

☐ Unique

Table name: Заказы

Method: B-Tree

Columns/Expression

	Name/Expression	Collation Schema	Collation	Operator Class Schema	Operator Class	Sort Order	Nulls Order
<input checked="" type="checkbox"/>	дата			pg_catalog	timestamp_ops	ASC	NULLS LAST

Рисунок 4.3 – Индекс поля дата таблицы Заказы

4.3 Разработка триггеров

Были разработаны триггеры before insert для всех таблиц и триггеры для каждого типа события.

`CREATE TRIGGER` создаёт новый триггер. Триггер будет связан с указанной таблицей, представлением или сторонней таблицей и будет выполнять заданную функцию имя_функции при определённых событиях.

Триггер можно настроить так, чтобы он срабатывал до операции со строкой (до проверки ограничений и попытки выполнить `INSERT`, `UPDATE` или `DELETE`) или после её завершения (после проверки ограничений и выполнения `INSERT`, `UPDATE` или `DELETE`), либо вместо операции (при добавлении, изменении и удалении строк в представлении). Если триггер срабатывает до или вместо события, он может пропустить операцию с текущей строкой, либо изменить добавляемую строку (только для операций `INSERT` и `UPDATE`). Если триггер срабатывает после события, он «видит» все изменения, включая результат действия других триггеров.

Триггер с пометкой `FOR EACH ROW` вызывается один раз для каждой строки, изменяемой в процессе операции. Например, операция `DELETE`, удаляющая 10 строк, приведёт к срабатыванию всех триггеров `ON DELETE` в целевом отношении 10 раз подряд, по одному разу для каждой удаляемой строки. Триггер с пометкой `FOR EACH STATEMENT`, напротив, вызывается только один раз для конкретной операции, вне зависимости от того, как много строк она изменила (в частности, при выполнении операции, изменяющей ноль строк, всё равно будут вызваны все триггеры `FOR EACH STATEMENT`). Заметьте, что при выполнении `INSERT` с предложением `ON CONFLICT DO UPDATE` сработают оба триггера уровня операторов, для `INSERT` и для `UPDATE`.

Триггеры, срабатывающие в режиме `INSTEAD OF`, должны быть помечены `FOR EACH ROW` и могут быть определены только для представлений. Триггеры `BEFORE` и `AFTER` для представлений должны быть помечены `FOR EACH STATEMENT`.

Чтобы создать триггер, пользователь должен иметь право `TRIGGER` для этой таблицы. Также пользователь должен иметь право `EXECUTE` для триггерной функции.

Для удаления триггера применяется команда DROP TRIGGER.

Триггер для избранных столбцов (определённый с помощью UPDATE OF имя_столбца) будет срабатывать, когда его столбцы перечислены в качестве целевых в списке SET команды UPDATE. Изменения, вносимые в строки триггерами BEFORE UPDATE, при этом не учитываются, поэтому значения столбцов можно изменить так, что триггер не сработает. И наоборот, при выполнении команды UPDATE ... SET x = x ... триггер для столбца x сработает, хотя значение столбца не меняется.

В триггере BEFORE условие WHEN вычисляется непосредственно перед возможным вызовом функции, поэтому проверка WHEN существенно не отличается от проверки того же условия в начале функции триггера. В частности, учтите, что строка NEW, которую видит ограничение, содержит текущие значения, возможно изменённые предыдущими триггерами. Кроме того, в триггере BEFORE условие WHEN не может проверять системные столбцы в строке NEW (например, oid), так как они ещё не установлены.

В триггере AFTER условие WHEN проверяется сразу после изменения строки, и если оно выполняется, событие запоминается, чтобы вызвать триггер в конце оператора. Если же для триггера AFTER условие WHEN не выполняется, нет необходимости запоминать событие для последующей обработки или заново перечитывать строку в конце оператора. Это приводит к значительному ускорению операторов, изменяющих множество строк, когда триггер должен срабатывать только для некоторых из них.

Триггеры уровня операторов для представления срабатывают, только если операция с представлением обрабатывается триггером уровня строк INSTEAD OF. Если операция обрабатывается правилом INSTEAD, то вместо исходного оператора, обращающегося к представлению, выполняются те операторы, что генерирует правило, поэтому вызываться будут триггеры, связанные с таблицами, к которым обращаются эти заменяющие операторы. Аналогично, для автоматически изменяемого представления выполнение операции сводится к переписыванию оператора в виде операции с базовой

таблицей представления, так что срабатывать будут триггеры уровня операторов для базовой таблицы.

В PostgreSQL до версии 7.3 обязательно требовалось объявлять триггерные функции, как возвращающие фиктивный тип `opaque`, а не `trigger`. Для поддержки загрузки старых файлов экспорта БД, команда `CREATE TRIGGER` принимает функции с объявленным типом результата `opaque`, но при этом выдаётся предупреждение и тип результата меняется на `trigger`.

```

1 CREATE OR REPLACE FUNCTION "public"."gen_inc_client"()
2 RETURNS "pg_catalog"."trigger" AS $BODY$BEGIN
3     NEW.id_client = (SELECT MAX("Заказчики".id_client)+1 FROM "Заказчики");
4     RETURN NEW;
5 END$BODY$
6 LANGUAGE plpgsql VOLATILE
7 COST 100
  
```

Рисунок 4.4 – Функция триггера before insert для генерации ключа

```

1 CREATE OR REPLACE FUNCTION "public"."gen_inc_country"()
2 RETURNS "pg_catalog"."trigger" AS $BODY$BEGIN
3     NEW.id_country = (SELECT MAX("страны".id_country)+1 FROM "страны");
4     RETURN NEW;
5 END$BODY$
6 LANGUAGE plpgsql VOLATILE
7 COST 100
  
```

Рисунок 4.5 – Функция триггера before insert для генерации ключа

```

1 CREATE OR REPLACE FUNCTION "public"."gen_inc_fuel"()
2 RETURNS "pg_catalog"."trigger" AS $BODY$BEGIN
3     NEW.id_fuel = (SELECT MAX("вид топлива".id_fuel)+1 FROM "вид топлива");
4     RETURN NEW;
5 END$BODY$
6 LANGUAGE plpgsql VOLATILE
7 COST 100
  
```

Рисунок 4.6 – Функция триггера before insert для генерации ключа

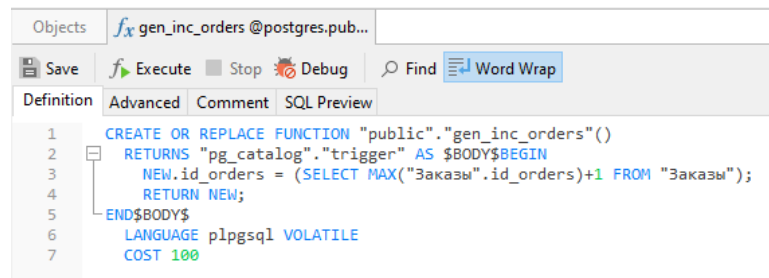


Рисунок 4.7 – Функция триггера before insert для генерации ключа

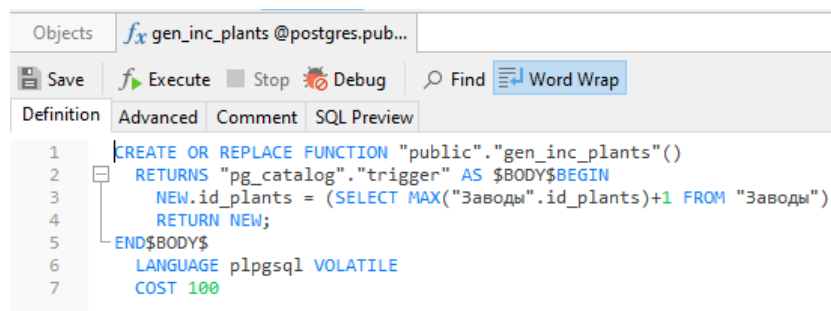


Рисунок 4.8 – Функция триггера before insert для генерации ключа

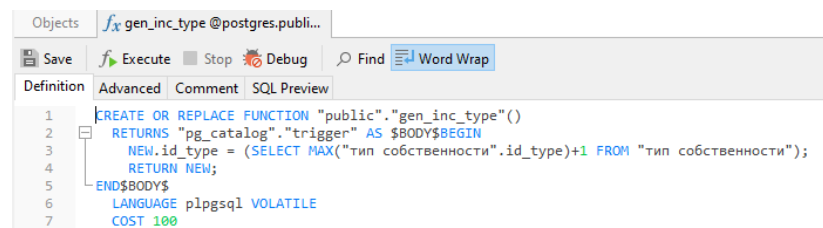


Рисунок 4.9 – Функция триггера before insert для генерации ключа

4.4 Проектирование запросов к базе данных

Во время обучения синтаксису SQL необходимо было разработать запросы следующих типов:

- 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 на группы;
- запрос на запросе по принципу итогового запроса;
- запрос с использованием объединения
- запросы с подзапросами (с использованием in, not in, case).

Запросы — это объект базы данных, который служит для извлечения данных из таблиц и предоставления их пользователю в удобном виде. Особенность запросов состоит в том, что они черпают данные из базовых таблиц и создают на их основе временную таблицу. Применение запросов позволяет избежать дублирования данных в таблицах и обеспечивает максимальную гибкость при поиске и отображении данных в базе данных.

4.5 Создание представлений и хранимых процедур, функций

На базе ранее представленных запросов разработаны отчёты Excel и представлены на рисунке 4.13.

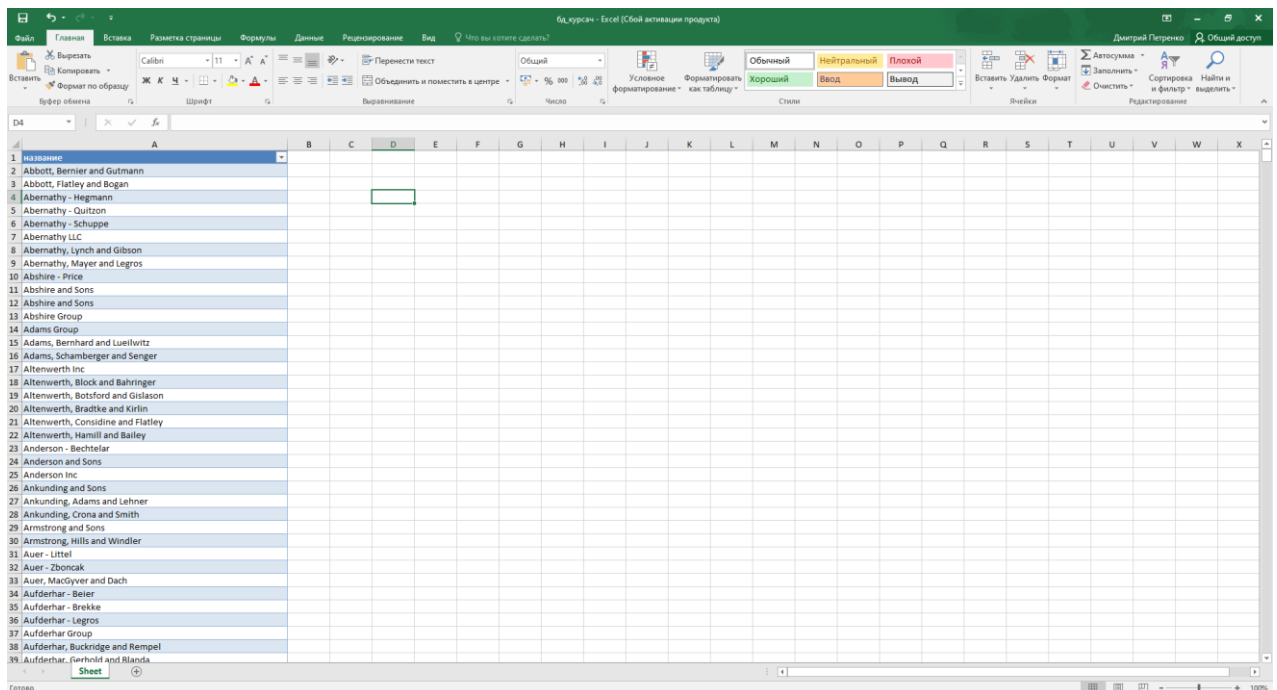


Рисунок 4.13 – Отчет Excel

Objects `fx * orders_bigger_V @postgres.p...`

Save Execute Stop Debug Find Word Wrap

Definition Advanced Comment SQL Preview

```

1 CREATE OR REPLACE FUNCTION "public"."orders_bigger_V"("_count" int4)
2 RETURNS TABLE("id_order" int4, "объём" int4, "дата" timestamp, "топливо" varchar) AS $BODY$
3 BEGIN
4 RETURN QUERY SELECT "Заказы"."id_order", "Заказы"."объём", "Заказы"."дата",
5 "вид топлива"."топливо" FROM "Заказы" NATURAL JOIN "вид топлива" WHERE "Заказы"."объём" > _count
6 ORDER BY "Заказы"."объём" DESC;
7 END $BODY$
8 LANGUAGE plpgsql VOLATILE
9 COST 100
10 ROWS 1000

```

id_order	объём	дата	ТОПЛИВО
11211	949	11.04.1995 11:03:18	а92
8076	949	23.03.2001 1:07:37	а92
12923	949	07.11.1998 11:15:11	а80
308	949	19.10.2018 5:04:38	дизельное
9774	949	09.04.2019 6:32:22	а76
10324	949	28.04.2005 13:42:38	а80
12280	949	02.07.2010 4:59:29	а95
13264	949	30.07.2002 14:16:39	дизельное
8299	949	28.05.2017 1:37:55	а92
7416	949	16.03.1992 4:33:16	а80
7470	949	09.01.2017 17:20:47	а76
5601	949	14.02.2011 22:59:14	а95
892	949	30.03.2002 9:05:15	а80
13802	949	06.01.1998 9:11:46	а76
8210	949	18.09.2017 11:07:26	а95
9481	949	31.03.2005 16:14:42	а92
14065	949	05.01.2017 6:23:39	дизельное

Рисунок 4.14 – Симметричное внутреннее соединение с условием отбора по внешнему ключу (Вывести заказы объем больше)

Objects `f_x * orders_smaller_V @postgres....`

Save Execute Stop Debug Find Word Wrap

Definition Advanced Comment SQL Preview

```

1 CREATE OR REPLACE FUNCTION "public"."orders_smaller_V"("_count" int4)
2 RETURNS TABLE("id_order" int4, "объём" int4, "дата" timestamp, "топливо" varchar) AS $BODY$
3 BEGIN
4 RETURN QUERY SELECT "Заказы"."id_order", "Заказы"."объём", "Заказы"."дата", "вид топлива"."топливо"
5 FROM "Заказы" NATURAL JOIN "вид топлива" WHERE "Заказы"."объём" < _count
6 ORDER BY "Заказы"."объём" DESC;
7 END $BODY$
8 LANGUAGE plpgsql VOLATILE
9 COST 100
10 ROWS 1000

```

id_order	объём	дата	топливо
11880	599	02.04.2020 15:04:20	дизельное
10899	599	26.10.1987 11:57:07	а80
9078	599	19.03.2020 7:21:47	а76
1678	599	03.06.1990 10:40:58	дизельное
695	599	18.02.1991 12:26:46	а92
3821	599	15.07.2013 22:14:01	а92
640	599	01.03.1988 11:15:17	дизельное
2459	599	30.04.2002 13:26:41	а76
3356	599	05.09.1985 13:04:19	а76
13012	599	05.02.1998 22:37:06	а76
12866	599	01.06.2015 13:54:44	а95
2105	599	26.10.2017 12:48:50	а92
4241	599	18.12.2004 6:08:16	дизельное
969	599	22.09.2003 22:56:30	дизельное
12181	599	03.01.2001 10:03:29	дизельное
5166	599	21.11.2005 16:11:05	а95
5584	599	05.04.2002 4:32:55	а92

Рисунок 4.15 – Симметричное внутреннее соединение с условием отбора по внешнему ключу (Вывести заказы объем меньше)

Objects `fx orders_by_last_months @post...`

Save Execute Stop Debug Find Word Wrap

Definition Advanced Comment SQL Preview

```

1 CREATE OR REPLACE FUNCTION "public"."orders_by_last_months"("_months" int4)
2 RETURNS TABLE("id_order" int4, "дата" timestamp, "объём" int4, "топливо" varchar) AS $BODY$
3 BEGIN
4 RETURN QUERY SELECT
5 "Заказы"."id_order",
6 "Заказы"."дата",
7 "Заказы"."объём",
8 "вид топлива"."топливо"
9 FROM "Заказы"
10 NATURAL JOIN "вид топлива"
11 WHERE "Заказы"."дата" > now() - (concat(_months, ' months'))::interval
12 ORDER BY "Заказы"."дата" DESC;
13 END $BODY$
14 LANGUAGE plpgsql VOLATILE
15 COST 100
16 ROWS 1000

```

	id_order	дата	объём	топливо
▶	14340	22.05.2020 0:00:00	555	а80
	2	21.05.2020 0:00:00	689	дизельное
	14342	21.05.2020 0:00:00	333	керосин
	1	20.05.2020 0:00:00	690	а92
	14341	18.05.2020 0:00:00	444	дизельное
	2570	28.04.2020 12:53:53	895	а92
	5838	28.04.2020 10:33:12	525	а76
	13928	26.04.2020 20:25:44	945	а95
	2770	24.04.2020 10:10:15	432	а92
	12371	23.04.2020 17:52:17	751	а92
	8128	22.04.2020 22:58:27	924	а76
*				

Рисунок 4.16 – Симметричное внутреннее соединение с условием отбора по дате (Вывести заказы за последние N месяцев)

Objects *fx* orders_by_last_years @postgre...

Save Execute Stop Debug Find Word Wrap

Definition Advanced Comment SQL Preview

```

1 CREATE OR REPLACE FUNCTION "public"."orders_by_last_years"("_years" int4)
2 RETURNS TABLE("id_order" int4, "дата" timestamp, "объём" int4, "топливо" varchar) AS $BODY$
3 BEGIN
4 RETURN QUERY SELECT
5 "Заказы"."id_order",
6 "Заказы"."дата",
7 "Заказы"."объём",
8 "вид топлива"."топливо"
9 FROM "Заказы"
10 NATURAL JOIN "вид топлива"
11 WHERE "Заказы"."дата" > now() - (concat(_years, ' years'))::interval
12 ORDER BY "Заказы"."дата" DESC;
13 END $BODY$
14 LANGUAGE plpgsql VOLATILE
15 COST 100
16 ROWS 1000

```

id_order	дата	объём	топливо
14340	22.05.2020 0:00:00	555	а80
14342	21.05.2020 0:00:00	333	керосин
2	21.05.2020 0:00:00	689	дизельное
1	20.05.2020 0:00:00	690	а92
14341	18.05.2020 0:00:00	444	дизельное
2570	28.04.2020 12:53:53	895	а92
5838	28.04.2020 10:33:12	525	а76
13928	26.04.2020 20:25:44	945	а95
2770	24.04.2020 10:10:15	432	а92
12371	23.04.2020 17:52:17	751	а92
8128	22.04.2020 22:58:27	924	а76
268	21.04.2020 5:37:09	562	а80
2106	19.04.2020 5:27:24	550	а76
39	18.04.2020 0:00:00	743	а92
5008	17.04.2020 18:51:50	767	а80
536	17.04.2020 9:48:22	670	дизельное
12472	17.04.2020 8:29:28	577	дизельное

Рисунок 4.17 – Симметричное внутреннее соединение с условием отбора по дате (Вывести заказы за последние N лет)

Objectsзаказы_natural_join @postgres...

Save

Preview

Explain

View Builder

Beautify SQL

Definition

Rules

Advanced

Comment

SQL Preview

```
1 | SELECT "Заказы"."объём",
2 |      "Заказы"."дата",
3 |      "Заказы".id_order,
4 |      "Заводы"."название",
5 |      "Заводы"."город",
6 |      "вид топлива"."топливо",
7 |      "Заводы"."цена"
8 | FROM ("Заказы"
9 | JOIN "вид топлива" ON (("Заказы".id_fuel = "вид топлива".id_fuel)))
10 | JOIN "Заводы" ON (((("Заводы".id_fuel = "вид топлива".id_fuel) AND ("Заказы".id_plants = "Заводы".id_plants))))
```

объём	дата	id_order	название
429	30.12.2015 11:01:12	52	Bogan, Sauer and Leuschke
430	16.08.1990 12:38:46	53	Wilkinson - Huels
756	30.08.2011 22:42:25	60	Lang - Berge
491	29.04.2005 21:05:01	65	Medhurst, Crooks and White
542	16.02.1999 13:10:25	76	Grant LLC
430	12.07.2017 7:16:13	83	Conn, Zieme and Ondricka
554	22.01.1998 22:36:12	90	Ankunding Group
863	08.05.2019 20:29:39	104	Sauer - Jacobson
405	15.02.1989 4:38:59	110	Buckridge Group
600	10.06.2016 6:57:10	114	Lemke Inc
534	11.10.2001 23:10:47	119	Johns - Carroll
751	26.08.1999 0:00:00	16	McKenzie, Raynor and Bauc
374	22.04.1999 0:00:00	17	Bode, Greenholt and Swift
788	23.06.1992 0:00:00	33	Kemmer, Jones and Kilback
871	27.01.2008 0:00:00	36	Mante Group
845	11.10.2017 0:00:00	40	Ritchie and Sons

Рисунок 4.18 – Симметричное внутреннее соединение без условия (вывести информацию о заказах)

Objects **заказчики_natural_join @post...**

Save Preview Explain View Builder Beautify SQL

Definition Rules Advanced Comment SQL Preview

```

1 | SELECT "Заказчики"."название",
2 |       "Заказы"."дата",
3 |       "Заказы"."объём"
4 | FROM ("Заказчики"
5 |       JOIN "Заказы" ON (("Заказы".id_client = "Заказчики".id_client)))

```

название	дата	объём
Bayer - Zulauf	26.03.2020 6:32:34	803
Russel Group	30.12.2015 11:01:12	429
Hartmann, Osinski and Fay	16.08.1990 12:38:46	430
Cronin - Kilback	09.09.1987 22:02:34	670
Magglo, Predovic and Mraz	02.04.2003 17:49:24	683
Price - Dibbert	04.12.1991 17:01:26	872
Marvin - Harris	02.07.1994 16:10:40	463
Thompson Inc	01.12.1998 9:33:55	787
Altenwerth, Botsford and Gislason	24.05.1995 18:39:03	766
Jacobs - Morar	30.08.2011 22:42:25	756
Walter, Schaefer and Kuhic	05.11.2007 12:43:10	465
Frami - Quitzon	09.06.2013 9:21:43	354
Trantow - Metz	21.07.2009 12:05:49	612
Zieme - Goyette	02.08.2002 1:14:47	400
Hodkiewicz, Barton and Zulauf	29.04.2005 21:05:01	491
Watsica - Bosco	12.01.2008 8:52:35	434
Ryan LLC	17.02.1991 9:19:51	452

Рисунок 4.19 – Симметричное внутреннее соединение без условия
(вывести информацию о заказчиках)

Objects

заводы_natural_join @postgre...

Save

Preview

Explain

View Builder

Beautify SQL

Definition

Rules

Advanced

Comment

SQL Preview

```
1 SELECT "вид топлива". "топливо",
2       "Заводы". "название",
3       "Заводы". "город",
4       "Заводы". "объём",
5       "Заводы". "год",
6       "Заводы". "телефон",
7       "Заводы". "цена",
8       "страны". "страна"
9 FROM (( "Заводы"
10 JOIN "вид топлива" ON (( "Заводы". id_fuel = "вид топлива". id_fuel )))
11 JOIN "страны" ON (( "Заводы". id_country = "страны". id_country )))
```

топливо	название	город
a76	Torphy - Hartmann	Сочи
a80	Balistreri Group	Тольятти
a95	Collins Group	Курск
a76	Sanford - Feil	Оренбург
a76	Daniel - Gulgowski	Оренбург
a92	Boyle - Dickinson	Владивосток
дизельное	Koss - Hettinger	Санкт-Петербу
a80	Monahan, Windler and Kiehn	Владимир
a80	Pfannerstill - Jones	Москва
a95	Purdy - Gulgowski	Махачкала
a92	Kassulke - Deckow	Красноярск
a76	Jaskolski, Schneider and Batz	Магнитогорск
a76	Dicki, Harris and Ebert	Тольятти
a76	Stiedemann - Gibson	Красноярск
a95	Nitzsche - Wunsch	Новокузнецк
a80	Schumm, Daniel and Krajcik	Пенза

Рисунок 4.20 – Симметричное внутреннее соединение без условия (вывести информацию о заводах)

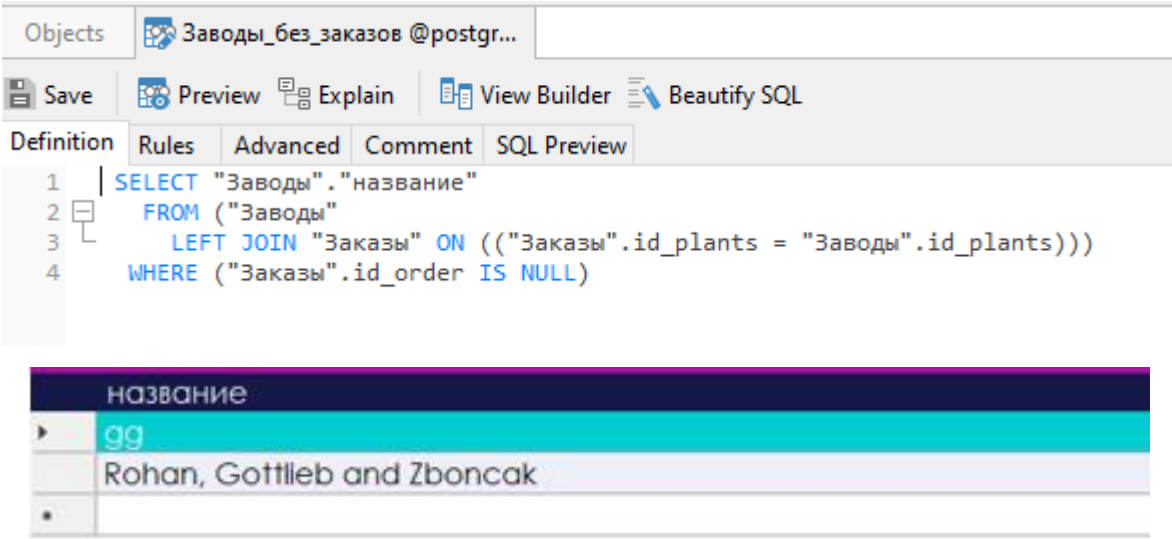


Рисунок 4.21 – Левое внешнее соединение (заводы без заказов)

The screenshot shows a database query editor interface. At the top, there's a tab labeled "Заказчики_без_заказов @post...". Below the tab are buttons for "Save", "Preview", "Explain", "View Builder", and "Beautify SQL". The "Definition" tab is active, displaying a SQL query:

```
1 | SELECT "Заказчики"."название",  
2 |     "Заказчики"."город"  
3 | FROM ("Заказы"  
4 |     RIGHT JOIN "Заказчики" ON (("Заказы".id_client = "Заказчики".id_client)))  
5 | WHERE ("Заказы".id_order IS NULL)
```

Below the query, the results are displayed in a table with two columns: "название" and "город". The first row shows "Homenick, Casper and Bogisich" and "Белгород".

название	город
Homenick, Casper and Bogisich	Белгород
*	

Рисунок 4.22 – Правое внешнее соединение (клиенты без заказов)

Objects emulate_left_join @postgres.p...

Save Preview Explain View Builder Beautify SQL

Definition Rules Advanced Comment SQL Preview

```

1 | SELECT "Заказчики"."название" AS "заказчик",
2 |   ( SELECT "Заказы".id_order
3 |     FROM "Заказы"
4 |     WHERE ("Заказы".id_client = "Заказчики".id_client)
5 |     LIMIT 1) AS "заказы"
6 | FROM "Заказчики"
7 | WHERE (( SELECT "Заказы".id_order
8 |          FROM "Заказы"
9 |          WHERE ("Заказы".id_client = "Заказчики".id_client)
10 |          LIMIT 1) IS NULL)

```

заказчик	заказы
Homenick, Casper and Bogisich	

Рисунок 4.23 – Запрос на запросе по принципу левого соединения
(клиенты без заказов)

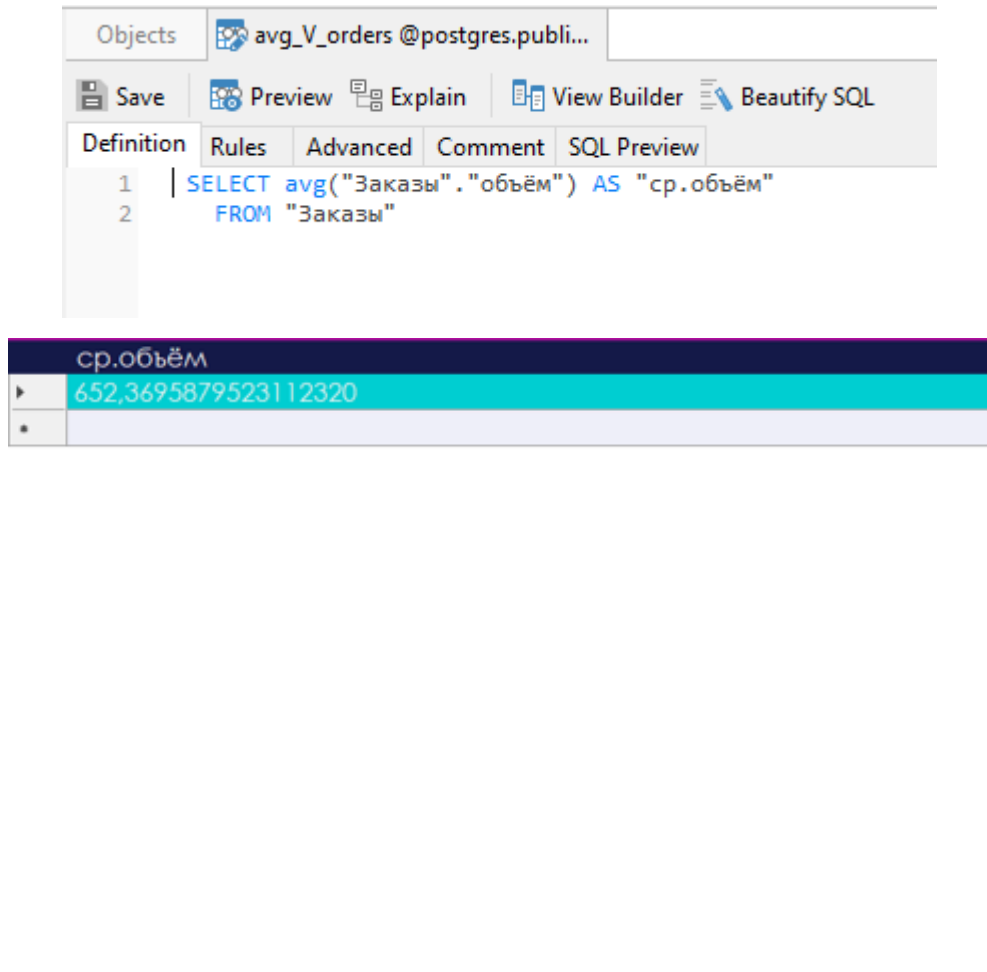


Рисунок 4.24 – Итоговый запрос без условия (Средняя объем заказов)

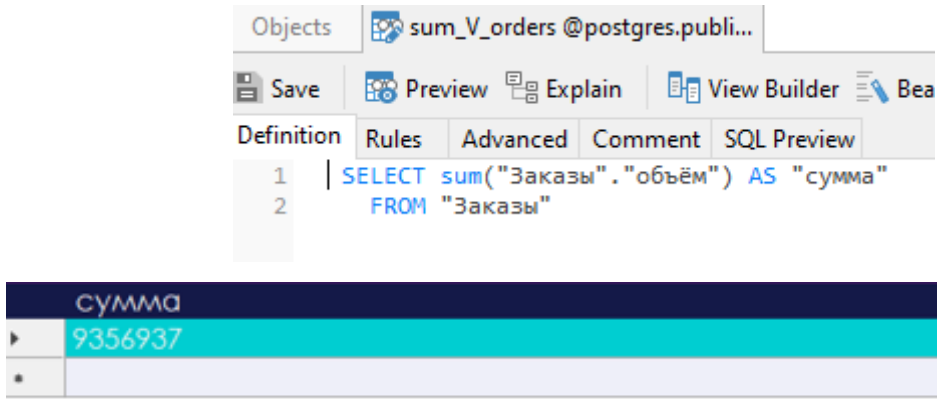


Рисунок 4.25 – Итоговый запрос без условия с итоговыми данными вида «все­го», «в том числе» (Сумма заказов)

The screenshot shows a database IDE interface. At the top, the 'Objects' tab is active, displaying 'orders_sum_where_V_bigger @...'. Below this is a toolbar with buttons for 'Save', 'Execute', 'Stop', 'Debug', 'Find', and 'Word Wrap'. The 'Definition' tab is selected, showing the SQL code for a function. The code is as follows:

```

1 CREATE OR REPLACE FUNCTION "public"."orders_sum_where_V_bigger"("_n" int4)
2 RETURNS TABLE("объём" int8) AS $BODY$
3 BEGIN
4 RETURN QUERY SELECT sum("Заказы"."объём") FROM "Заказы" WHERE "Заказы"."объём" > _n;
5 END$BODY$
6 LANGUAGE plpgsql VOLATILE
7 COST 100
8 ROWS 1000

```

Below the code, the execution result is displayed in a table with a dark blue header and a light blue body. The table has one column labeled 'объём' and one row with the value '6540246'.

объём
6540246

Рисунок 4.26 – Итоговый запрос с условием на данные по значению
(заказы средняя цена больше N)

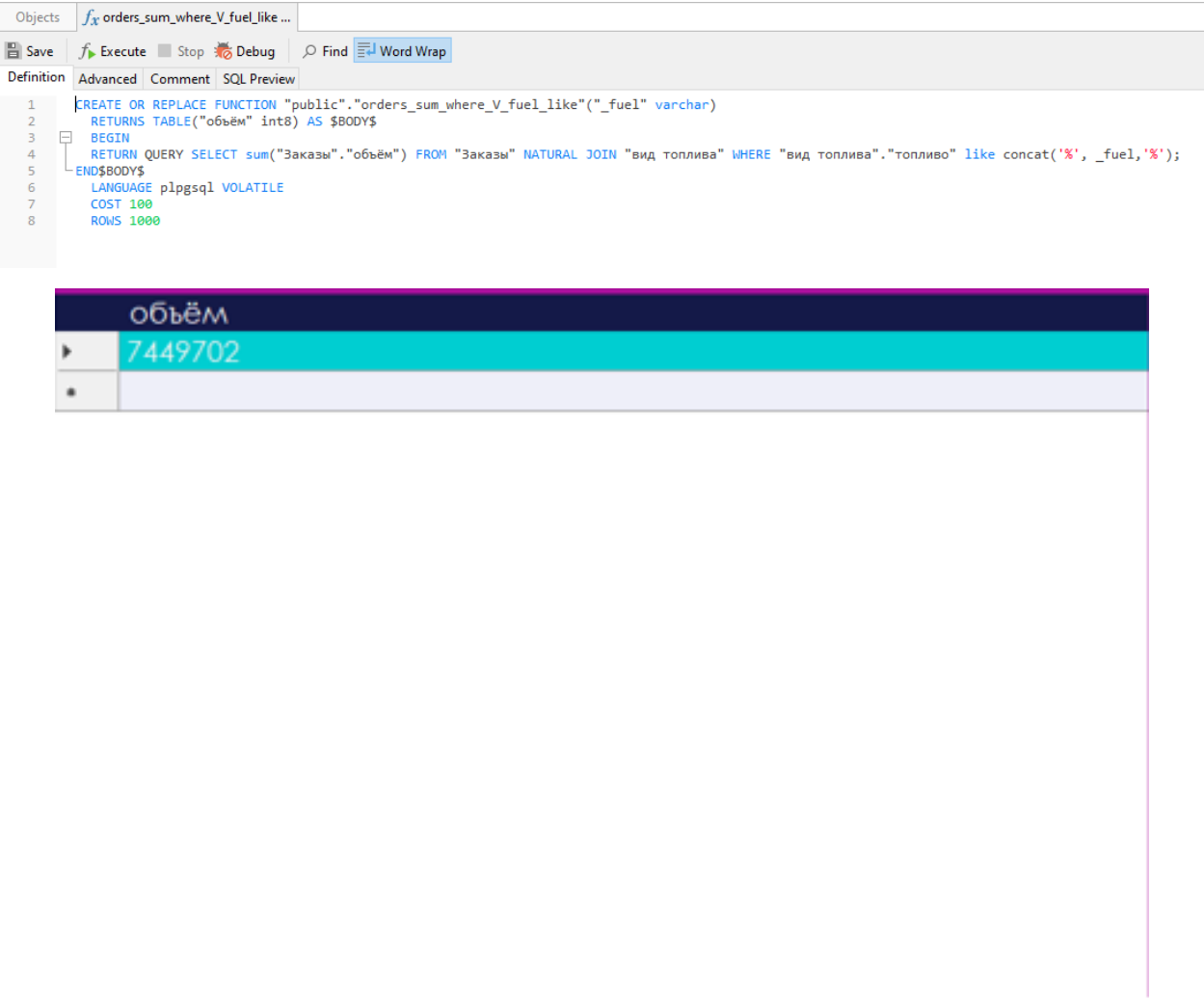


Рисунок 4.27 – Итоговый запрос с условием на данные по маске (объем топлива по маске)

Objects **orders_by_last_months_index ...**

Save Execute Stop Debug Find Word Wrap

Definition Advanced Comment SQL Preview

```

1 CREATE OR REPLACE FUNCTION "public"."orders_by_last_months_index"("_months" int4)
2 RETURNS TABLE("количество" int8) AS $BODY$
3 BEGIN
4 RETURN QUERY SELECT COUNT(*) as "количество" FROM "Заказы" WHERE "Заказы"."дата" > now() - (concat(_months, ' months'))::interval;
5 END $BODY$
6 LANGUAGE plpgsql VOLATILE
7 COST 100
8 ROWS 1000

```

количество
13

Message Result 1

QUERY PLAN

► Aggregate (cost=33.22..33.23 rows=1 width=8) (actual time=0.047..0.047 rows=1 loops=1)

-> Bitmap Heap Scan on "Заказы" (cost=4.37..33.20 rows=9 width=0) (actual time=0.029..0.043 rows=7 loops=1)

Recheck Cond: ("дата" > (now() - (concat(1, ' months'))::interval))

Heap Blocks: exact=7

-> Bitmap Index Scan on date_index (cost=0.00..4.37 rows=9 width=0) (actual time=0.022..0.022 rows=7 loops=1)

Index Cond: ("дата" > (now() - (concat(1, ' months'))::interval))

Planning time: 0.389 ms

Execution time: 0.080 ms

Рисунок 4.28 – Итоговый запрос с условием на данные с индексом
(кол-во заказов за последние N месяцев)

Objects `avg_orders @postgres.public (...)`

Save `Execute` `Stop` `Debug` `Find` `Word Wrap`

Definition `Advanced` `Comment` `SQL Preview`

```

1 CREATE OR REPLACE FUNCTION "public"."avg_orders"()
2   RETURNS TABLE("объём" numeric) AS $BODY$
3 BEGIN
4   RETURN QUERY SELECT
5     AVG("объём") as "объём"
6   FROM "Заказы";
7 END $BODY$
8 LANGUAGE plpgsql VOLATILE
9 COST 100
10 ROWS 1000

```

объём
652,3695879523112320

QUERY PLAN	
► Aggregate	(cost=299.23..299.24 rows=1 width=32) (actual time=2.594..2.594 rows=1 loops=1)
-> Seq Scan on "Заказы"	(cost=0.00..263.38 rows=14338 width=4) (actual time=0.013..1.413 rows=14338 loops=1)
Planning time: 0.227 ms	
Execution time: 2.624 ms	

Рисунок 4.29 – Итоговый запрос с условием на данные без индекса
(средние объем заказов)

Objects `fx avg_orders_bigger @postgres....` `* save @postgres.public (Bobe...` `fx avg_orders`

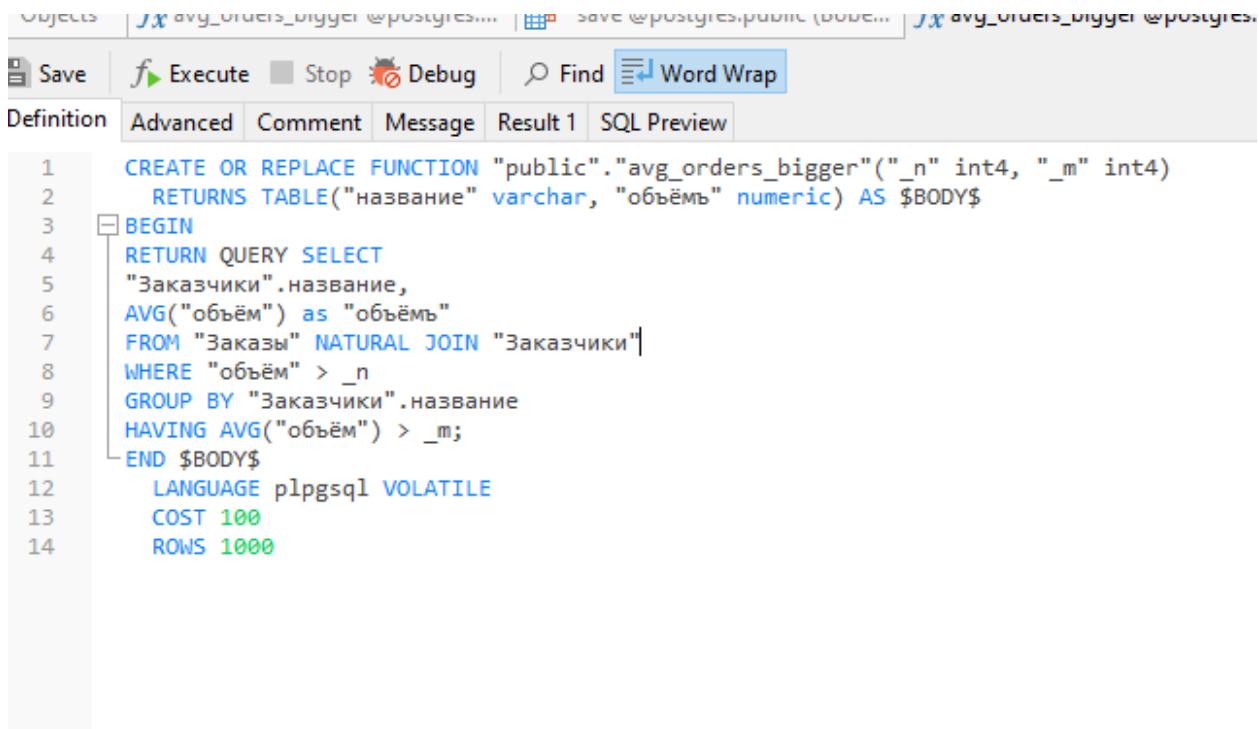
Save `fx` Execute Stop Debug Find Word Wrap

Definition Advanced Comment Message Result 1 SQL Preview

```
1 CREATE OR REPLACE FUNCTION "public"."avg_orders_bigger"("_n" int4)
2 RETURNS TABLE("название" varchar, "объём" numeric) AS $BODY$
3 BEGIN
4 RETURN QUERY SELECT
5 "Заказчики".название,
6 AVG("объём") as "объём"
7 FROM "Заказы" NATURAL JOIN "Заказчики"
8 WHERE "объём" > _n
9 GROUP BY "Заказчики".название;
10 END $BODY$
11 LANGUAGE plpgsql VOLATILE
12 COST 100
13 ROWS 1000
```

название	объём
Crist LLC	636,3846153846153846
Collins - Barton	622,8000000000000000
Powlowski - Wisoky	613,0000000000000000
Roob and Sons	689,8181818181818182
Little Inc	632,5333333333333333
McLaughlin, Bauch and Thiel	608,6666666666666667
Bayer - Zulauf	646,7647058823529412
Cummings, Langosh and Stiedemann	656,8333333333333333
Feeney, Leannon and Considine	602,7692307692307692
Lind Group	575,7142857142857143
Strosin, Reinger and Casper	664,7500000000000000
Ward - Lynch	685,9000000000000000
Brakus - Koss	799,1111111111111111
Frami - Kreiger	649,6363636363636364
Tromp Group	736,3750000000000000
Ferry, Abshire and Emard	727,1428571428571429
Rice - Dach	775,0000000000000000

Рисунок 4.30 – Итоговый запрос с условием на группы



The screenshot shows a database IDE with a menu bar (Save, Execute, Stop, Debug, Find, Word Wrap) and a tab bar (Definition, Advanced, Comment, Message, Result 1, SQL Preview). The main editor displays the following SQL code:

```
1 CREATE OR REPLACE FUNCTION "public"."avg_orders_bigger"("_n" int4, "_m" int4)
2 RETURNS TABLE("название" varchar, "объём" numeric) AS $BODY$
3 BEGIN
4 RETURN QUERY SELECT
5 "Заказчики".название,
6 AVG("объём") as "объём"
7 FROM "Заказы" NATURAL JOIN "Заказчики"
8 WHERE "объём" > _n
9 GROUP BY "Заказчики".название
10 HAVING AVG("объём") > _m;
11 END $BODY$
12 LANGUAGE plpgsql VOLATILE
13 COST 100
14 ROWS 1000
```



название	объём
Crist LLC	791,1428571428571429
Collins - Barton	813,5000000000000000
Powlowski - Wisoky	750,0000000000000000
Roob and Sons	803,5714285714285714
Little Inc	858,7142857142857143
McLaughlin, Bauch and Thiel	749,3333333333333333
Bayer - Zulauf	789,8888888888888889
Cummings, Langosh and Stiedemann	762,2500000000000000
Feeney, Leannon and Considine	742,2857142857142857
Lind Group	858,5000000000000000
Strosin, Reinger and Casper	766,7142857142857143
Ward - Lynch	832,3333333333333333
Brakus - Koss	848,8750000000000000
Frami - Kreiger	856,2000000000000000
Tromp Group	825,5000000000000000
Ferry, Abshire and Emard	815,4000000000000000
Rice - Dach	834,1428571428571429

Рисунок 4.31 – Итоговый запрос с условием на данные и на группы

Objects avg_orders_bigger @postgres... podzapros_itog @postgres.pu...

Save Preview Explain View Builder Beautify SQL

Definition Rules Advanced Comment SQL Preview

```

1 SELECT bb."название",
2       ( SELECT round(avg(aa."объём"), 2) AS round
3         FROM ("Заказы" aa
4              JOIN "Заказчики" cc USING (id_client))
5         WHERE ((cc."название")::text = (bb."название")::text)) AS "среднее",
6       bb."телефон"
7 FROM "Заказчики" bb
8 ORDER BY bb."название"

```

название	среднее	телефон
Abbott, Bernier and Gutmann	544,11	+380710685832
Abbott, Flatley and Bogan	654,71	+380715897416
Abernathy - Hegmann	706,50	+380711151713
Abernathy - Quitzon	806,00	+380710169050
Abernathy - Schuppe	697,65	+380716702658
Abernathy LLC	631,71	+380718601143
Abernathy, Lynch and Gibson	682,20	+380712406796
Abernathy, Mayer and Legros	656,50	+380714725836
Abshire - Price	724,50	+380719506132
Abshire and Sons	648,59	+380718469482
Abshire and Sons	648,59	+380718537673
Abshire Group	659,56	+380712431180
Adams Group	652,33	+380711237059
Adams, Bernhard and Luellwitz	600,15	+380718648395
Adams, Schamberger and Senger	638,50	+380710984357
Altenwerth Inc	716,08	+380716057787
Altenwerth, Block and Bohringer	612,38	+380717184959

Рисунок 4.32 – Запрос на запросе по принципу итогового запроса

The screenshot displays a database management interface for a PostgreSQL database. The top section shows the 'union_view' object. Below it, the 'Definition' tab is active, showing a SQL query that uses a UNION to combine data from two tables: 'вид топлива' and 'тип собственности'. The query selects 'id_fuel' and 'топливо' from the first table, and 'id_type' (aliased as 'id_fuel') and 'тип' (aliased as 'топливо') from the second table. Below the query, a table with two columns, 'id_fuel' and 'топливо', displays the results of the query. The table contains 12 rows of data, including fuel types like 'частный', 'а95', 'керосин', 'а76', 'а80', 'а92', 'ЗАО', 'дизельное', 'государственный', and 'ООО'.

```

1  SELECT "вид топлива".id_fuel,
2      "вид топлива"."топливо"
3  FROM "вид топлива"
4  UNION
5  SELECT "тип собственности".id_type AS id_fuel,
6      "тип собственности"."тип" AS "топливо"
7  FROM "тип собственности"

```

id_fuel	топливо
2	частный
5	а95
6	керосин
2	а76
3	а80
4	а92
3	ЗАО
1	дизельное
1	государственный
4	ООО

Рисунок 4.33 – Запрос с использованием объединения

```
Objects f_x * buyers_last_months @postgr...
Save Execute Stop Debug Find Word Wrap
Definition Advanced Comment SQL Preview
1 CREATE OR REPLACE FUNCTION "public"."buyers_last_months"("_months" int4)
2 RETURNS TABLE("название" varchar) AS $BODY$
3 BEGIN
4 RETURN QUERY SELECT "Заказчики"."название" FROM "Заказчики" WHERE id_client
5 IN (SELECT id_client FROM "Заказы" WHERE "Заказы"."дата" > now() - (concat(_months, ' months'))::interval)
6 ORDER BY "Заказчики"."название";
7 END $BODY$
8 LANGUAGE plpgsql VOLATILE
9 COST 100
10 ROWS 1000
```

НАЗВАНИЕ
Abernathy - Hegmann
Abernathy LLC
Bahringer LLC
Brown, Mertz and Armstrong
Effertz and Sons
Haag LLC
Harber, Macejkovic and Hilpert
Huel, Nikolaus and Langosh
Keeling, Walker and Thiel
Parisian - Jakubowski
Schmidt, Macejkovic and Runolfsson
Stark - Price
Trantow - Metz

Рисунок 4.34 – Запрос с подзапросом in

Objects `fx*no_buyers_last_months@po...`

Save Execute Stop Debug Find Word Wrap

Definition Advanced Comment SQL Preview

```

1 CREATE OR REPLACE FUNCTION "public"."no_buyers_last_months"("_months" int4)
2 RETURNS TABLE("название" varchar) AS $BODY$
3 BEGIN
4 RETURN QUERY SELECT "Заказчики"."название" FROM "Заказчики" WHERE id_client NOT IN
5 (SELECT id_client FROM "Заказы" WHERE "Заказы"."дата" > now() - (concat(_months, ' months'))::interval)
6 ORDER BY "Заказчики"."название";
7 END $BODY$
8 LANGUAGE plpgsql VOLATILE
9 COST 100
10 ROWS 1000

```

НАЗВАНИЕ
Abbott, Bernier and Gutmann
Abbott, Flatley and Bogan
Abernathy - Quitzon
Abernathy - Schuppe
Abernathy, Lynch and Gibson
Abernathy, Mayer and Legros
Abshire - Price
Abshire and Sons
Abshire and Sons
Abshire Group
Adams Group
Adams, Bernhard and Lueilwitz
Adams, Schamberger and Adams, Schamberger and Senger
Altenwerth Inc
Altenwerth, Block and Bahringer
Altenwerth, Botsford and Gislason
Altenwerth, Bradtke and Kirlin

Рисунок 4.35 – Запрос с подзапросом not in

Objects case_view @postgres.public (B...

Save Preview Explain View Builder Beautify SQL

Definition Rules Advanced Comment SQL Preview

```

1 | SELECT a."название",
2 |     a."дата",
3 |     CASE
4 |         WHEN (a."дата" > (now() - '1 mon'::interval)) THEN 'недавно'::text
5 |         ELSE 'давно'::text
6 |     END AS "case"
7 | FROM ("Заказчики"
8 | JOIN "Заказы" USING (id_client)) a
9 | WHERE (a."дата" = ( SELECT max(b."дата") AS max
10 |                     FROM "Заказы" b
11 |                     WHERE (a.id_client = b.id_client)))
12 | ORDER BY a."дата" DESC

```

название	дата	case
Abernathy - Hegmann	22.05.2020 0:00:00	недавно
Keeling, Walker and Thiel	21.05.2020 0:00:00	недавно
Brown, Mertz and Armstrong	21.05.2020 0:00:00	недавно
Schmidt, Macejkovic and Runolfsson	20.05.2020 0:00:00	недавно
Abernathy LLC	19.05.2020 0:00:00	недавно
Effertz and Sons	18.05.2020 0:00:00	недавно
Bahringer LLC	07.05.2020 0:00:00	недавно
Parisian - Jakubowski	28.04.2020 12:53:53	недавно
Harber, Macejkovic and Hilpert	28.04.2020 10:33:12	недавно
Trantow - Metz	26.04.2020 20:25:44	недавно
Huel, Nikolaus and Langosh	24.04.2020 10:10:15	недавно
Haag LLC	23.04.2020 17:52:17	недавно
Stark - Price	22.04.2020 22:58:27	недавно
Bechtelar, Thompson and Casper	21.04.2020 5:37:09	давно
Johnston Group	19.04.2020 5:27:24	давно
Leffler and Sons	18.04.2020 0:00:00	давно

Рисунок 4.36 – Запрос с case

5 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

5.1 Формы и компоненты для работы с основными таблицами

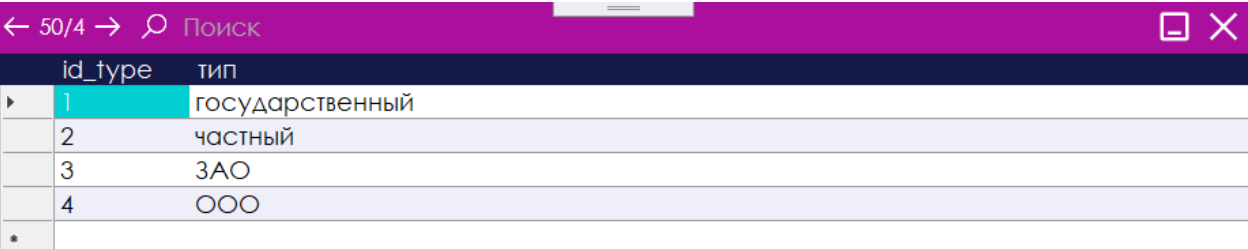


id_plants	название	тип	город	страна
26	Schuppe LLC	ЗАО	Хабаровск	Украина
27	Torphy - Hartmann	государств...	Сочи	Казахста
28	Ballstreri Group	частный	Тольятти	Украина
29	Collins Group	частный	Курск	Казахста
30	Sanford - Feil	ЗАО	Оренбург	Турекме
31	Daniel - Gulgowski	частный	Оренбург	ДНР
32	Boyle - Dickinson	ЗАО	Владивосток	Россия
33	Koss - Hettinger	государств...	Санкт-Петербург	Украина
34	Monahan, Windler and Kiehn	частный	Владимир	Украина
35	Pfannerstill - Jones	государств...	Москва	Казахста
36	Purdy - Gulgowski	государств...	Махачкала	Казахста
37	Kassulke - Deckow	частный	Красноярск	Турекме
38	Jaskolski, Schneider and Batz	государств...	Магнитогорск	Турекме
39	Dicki, Harris and Ebert	частный	Тольятти	Россия
40	Stiedemann - Gibson	государств...	Красноярск	Казахста
41	Nitzsche - Wunsch	ЗАО	Новокузнецк	Турекме

Заводы Вид топлива Заказы Заказчики Тип собственности Страны Запросы

Рисунок 5.1 – форма для работы с основной таблицей

5.2 Формы и компоненты для работы со справочниками



id_type	тип
1	государственный
2	частный
3	ЗАО
4	ООО

Заводы Вид топлива Заказы Заказчики Тип собственности Страны Запросы

Рисунок 5.2 – форма для работы с основной таблицей

← 50/14343 → 🔍 Поиск

id_order	Заказчики	вид топлива	объём	дата	Заводы
1	Schmidt, Ma...	а92	690	20.05.2020	Sanford - Feil
2	Brown, Mertz...	дизельное	689	21.05.2020	Torphy - Hartmann
3	Bins, Bartell a...	а80	813	23.01.2007	Nitzsche - Wunsch
4	Kihn, Parisian ...	а95	786	01.10.2013	Oberbrunner - Boyle
5	Gorczyany Gr...	а76	789	26.10.2018	Kirlin - Corwin
6	Schoen and ...	а76	774	26.05.2001	Kuhn - McGlynn
7	Friesen LLC	а80	389	01.07.2000	Schaden and Sons
8	Cronin, Wate...	а92	563	23.02.1995	Koss - Hettinger
9	Gorczyany Gr...	а95	732	31.07.1998	Yundt, Lehner and Goodwi
10	Gorczyany Gr...	дизельное керосин	518	25.04.1990	Boyle - Dickinson
11	Ritchie Inc	а76	415	20.10.2003	Jacobs, Bode and Murphy
12	Bins, Bartell a...	а92	909	05.01.2006	Schaden and Sons
13	Ritchie Inc	а95	830	17.05.1986	Jaskolski, Schneider and Ba
14	Schmidt, Ma...	дизельное	633	19.01.2015	Collins Group
15	Weber, Ziem...	а95	640	29.11.2002	Kassulke - Deckow
16	Crist and Sons	а92	751	26.08.1999	McKenzie, Raynor and Bau...

Заводы Вид топлива Заказы Заказчики Тип собственности Страны Запросы

Рисунок 5.3 – пример выпадающего списка, заполняемого по справочнику

5.3 Формы и компоненты для отображения результатов запросов

Запросы

объём	дата	id_order	название
429	30.12.2015 11:01:12	52	Bogan, Sauer and Leuschke
430	16.08.1990 12:38:46	53	Wilkinson - Huels
756	30.08.2011 22:42:25	60	Lang - Berge
491	29.04.2005 21:05:01	65	Medhurst, Crooks and White
542	16.02.1999 13:10:25	76	Grant LLC
430	12.07.2017 7:16:13	83	Conn, Zieme and Ondricka
554	22.01.1998 22:36:12	90	Ankunding Group
863	08.05.2019 20:29:39	104	Sauer - Jacobson
405	15.02.1989 4:38:59	110	Buckridge Group
600	10.06.2016 6:57:10	114	Lemke Inc
534	11.10.2001 23:10:47	119	Johns - Carroll
751	26.08.1999 0:00:00	16	McKenzie, Raynor and Bauc
374	22.04.1999 0:00:00	17	Bode, Greenholt and Swift
788	23.06.1992 0:00:00	33	Kemmer, Jones and Kilback
871	27.01.2008 0:00:00	36	Mante Group
845	11.10.2017 0:00:00	40	Ritchie and Sons

Экспорт в Excel

- Заказы
- Заводы
- Заказчики
- Заказы объём больше
- Заказы объём меньше
- Заказы за пр. месяцы
- Заказы за пр. года
- Заводы без заказов
- Клиенты без заказов
- Клиенты без заказов v2
- ср. объём заказов

Рисунок 5.4 – форма для отображения запросов

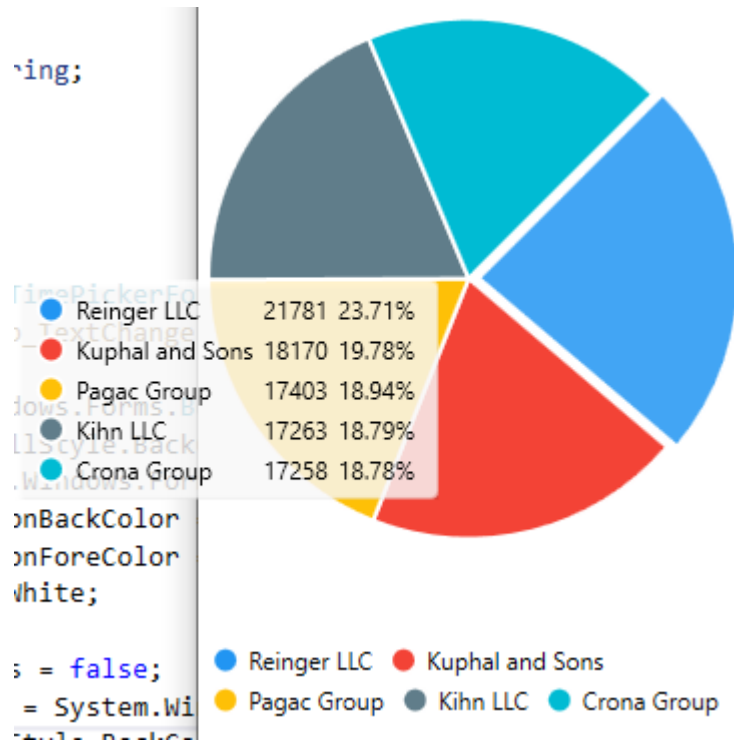


Рисунок 5.5 – круговая диаграмма Топ Заказчиков

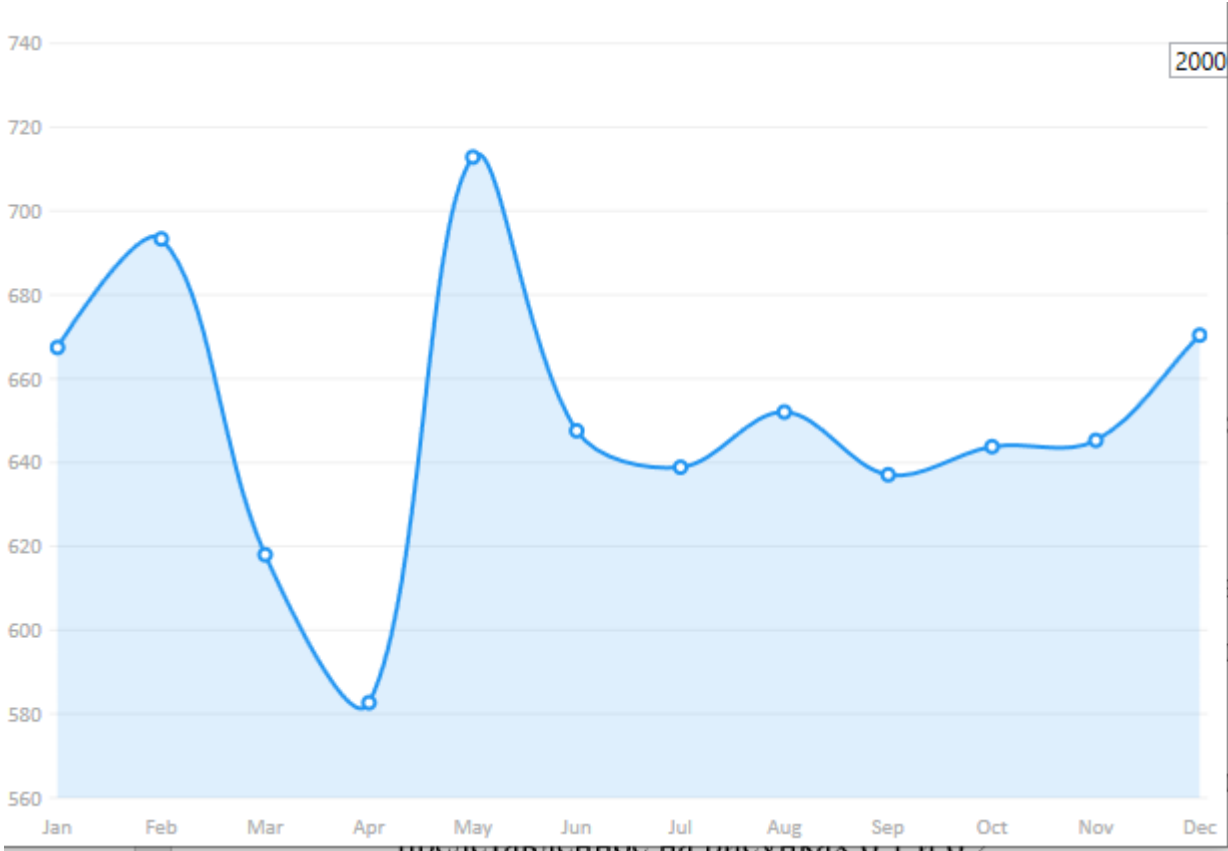


Рисунок 5.6 – график средней цены заказов по месяцам в 2000г

6 ТЕСТИРОВАНИЕ РАЗРАБОТАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

После создания информационной системы было проведено его тестирования. Была проверена возможность одновременной работы с данными, каскадное удаление, изменение.

При ошибочном вводе данных пользователю выводится сообщение, представленное на рисунках 6.1 и 6.2

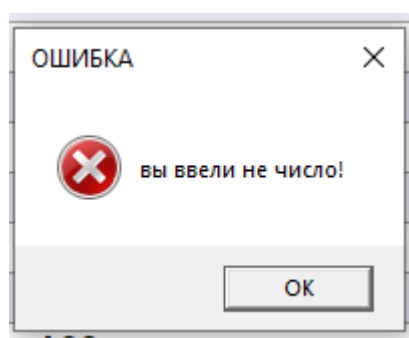


Рисунок 6.1 – Ошибка при вводе неправильных данных

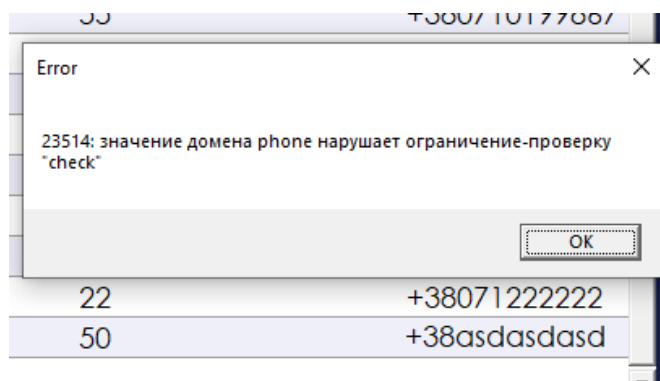


Рисунок 6.2 – Ошибка при вводе не правильного номера телефона

При вводе неправильных входных данных пользователь получает сообщение об ошибке авторизации, представленная на рисунке 6.3

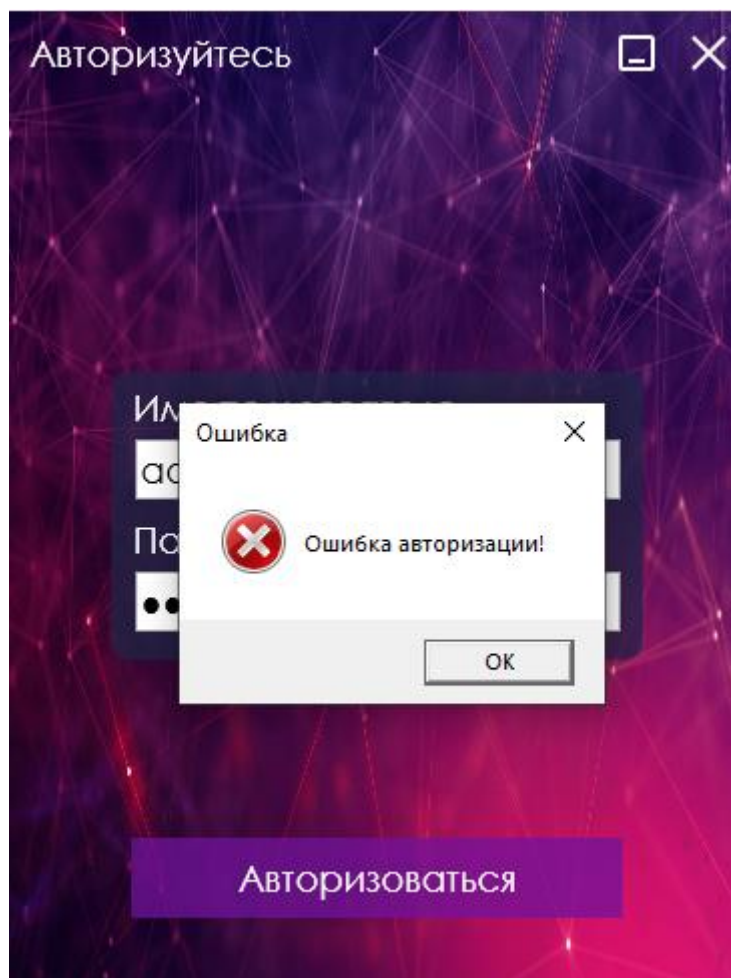




Рисунок 5.3 – Ошибка при некорректно заполненных данных

Было созданы роли: администратора и сотрудников. Администратор имеет доступ почти ко всем элементам созданной базы данных, а у сотрудников есть доступ исключительно к своим записям в таблице Заказы. Остальные сотрудники наследуются от этих ролей.

Objects

 admin (Bober) - Role

 Save

GeneralMember OfMembersPrivilegesCommentSQL Preview

Role Name:

admin

Role ID:

24666

☒ Can login

Password:

••••••••

Confirm Password:

••••••••

Password Encryption:

Connection Limit:

-1

Expiry Date:

☒ Superuser

☐ Can create databases

☐ Can create roles


☒ Inherit privileges


☐ Can replicate

☐ Can bypass RLS

Рисунок 6.4 – пользователь admin

Objects

 employee (Bober) - Role

 Save

GeneralMember OfMembersPrivilegesCommentSQL Preview

Role Name:

employee

Role ID:

24667

☒ Can login

Password:

••••••••

Confirm Password:

••••~•~•~•

Password Encryption:

Connection Limit:

-1

Expiry Date:

☐ Superuser

☐ Can create databases

☐ Can create roles

☒ Inherit privileges

☐ Can replicate

☐ Can bypass RLS

Рисунок 6.5 – пользователь employee

Для таблицы Заказы был включен модификатор ROW LEVEL SECURITY, который защищает данные в таблице на уровне строк.

На рисунке 6.6 сказано, что роль admin имеет доступ ко всем строкам таблицы Заказы, а роль employee имеет доступ только к записям, которые создал он сам. Работу данного механизма обеспечивает триггер before_insert_orders, который перед вставкой в таблицу заполняет поле владельца записи текущего пользователя.

```
1 ALTER TABLE "Заказы" ENABLE ROW LEVEL SECURITY;
2
3 DROP POLICY IF EXISTS admin_orders ON "Заказы";
4 CREATE POLICY admin_orders ON "Заказы"
5 FOR ALL TO admin USING (true);
6
7 DROP POLICY IF EXISTS employee_orders ON "Заказы";
8 CREATE POLICY employee_orders ON "Заказы"
9 FOR ALL TO employee USING (login = CURRENT_USER);
10
11 CREATE OR REPLACE FUNCTION before_insert_orders_fun() RETURNS TRIGGER AS $emp_audit$
12 BEGIN
13     NEW.login = CURRENT_USER;
14     RETURN NEW;
15 END;
16 $emp_audit$ LANGUAGE plpgsql;
17
18
19 CREATE TRIGGER before_insert_orders
20 BEFORE INSERT ON "Заказы"
21 FOR EACH ROW
22 EXECUTE PROCEDURE before_insert_orders_fun();
```

Рисунок 6.6 – ROW LEVEL SECURITY таблицы Заказы

Objects admin (Bober) - Role															
Save Add Privilege Delete Privilege															
General Member Of Members Privileges Comment SQL Preview															
Database: postgres															
Privileges															
	Database	Schema	Name	Connect	Create	Delete	Execute	Insert	References	Select	Temporary	Trigger	Truncate	Update	Usage
▶	postgres	public	страны	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	postgres	public	тип собственности	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	postgres	public	Заводы	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	postgres	public	Заказчики	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	postgres	public	вид топлива	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	postgres	public	Заказы	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	postgres	public	gen_inc_plants()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	gen_inc_fuel()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	before_insert_orders_fun	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	avg_orders()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	orders_by_last_months()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	orders_bigger_V(IN "_co	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	orders_by_last_years(IN "	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	orders_smaller_V(IN "_cc	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	orders_sum_where_V_biq	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	orders_by_last_months_i	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	orders_sum_where_V_fui	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	avg_orders_bigger(IN "_x	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	buyers_last_months(IN "	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	clients_by_last_months()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	avg_orders_bigger(IN "_x	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	no_buyers_last_months()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	sum_zakazi_by_client(IN	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	zakazi_avg_by_month_ys	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	gen_inc_client()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	gen_inc_country()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	gen_inc_orders()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	gen_inc_type()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Рисунок 6.7 - Привилегии пользователя admin

Objects * employee (Bober) - Role															
Save Add Privilege Delete Privilege															
General Member Of Members Privileges Comment SQL Preview															
Database: postgres															
Privileges															
	Database	Schema	Name	Connect	Create	Delete	Execute	Insert	References	Select	Temporary	Trigger	Truncate	Update	Usage
▶	postgres	public	страны	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	тип собственности	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	Заводы	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	postgres	public	Заказчики	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	postgres	public	вид топлива	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	Заказы	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	postgres	public	avg_orders()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	avg_orders_bigger(IN "_x	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	avg_orders_bigger(IN "_x	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	before_insert_orders_fun	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	buyers_last_months(IN "	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	clients_by_last_months()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	gen_inc_client()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	gen_inc_country()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	gen_inc_fuel()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	gen_inc_orders()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	gen_inc_plants()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	gen_inc_type()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	no_buyers_last_months()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	orders_bigger_V(IN "_co	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	orders_by_last_months()	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	orders_by_last_months_i	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	orders_by_last_years(IN "	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	orders_smaller_V(IN "_cc	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	orders_sum_where_V_biq	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	orders_sum_where_V_fui	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	sum_zakazi_by_client(IN	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postgres	public	zakazi_avg_by_month_ys	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Рисунок 6.8 – привилегии пользователя employee

← 50/3 → Поиск						
id_order	Заказчики	вид топлива	объём	дата	Заводы	
14340	Abernathy - ...	а80	555	22.05.2020	Anderson LLC	
14341	Effertz and S...	дизельное	444	18.05.2020	Anderson - Breitenberg	
14342	Keeling, Walk...	керосин	333	21.05.2020	Ankunding Group	
*						

Заводы

Вид топлива

Заказы

Заказчики

Тип собственности

Страны

Запросы

Рисунок 6.9 – Заказы созданные пользователем Дима (RLS)

← 50/4 → Поиск	
id_type	тип
1	государственный
2	частный
3	ЗАО
4	ООО
	а
*	

Заводы

Вид топлива

Заказы

Заказчики

Тип собственности

Страны

Запросы

Error

42501: нет доступа к отношению тип собственности

OK

Рисунок 6.10 – Попытка пользователем Дима работы с запрещенной таблицей

ЗАКЛЮЧЕНИЕ

В течение разработки курсового проекта была изучена предметная область проекта, разработана концептуальная модель БД, были написаны основные запросы и функции для данной предметной области, было разработано клиентское приложение. Также были изучены тонкости проектирования и разработки клиент-серверных приложений.

Система обладает уникальной авторизацией, базовым уровнем защиты данных от несанкционированного доступа.

Недостатками данной программы является то, что отображение данных часто производится с задержкой по причине вычисления данных и не осуществлено разделение ролей пользователей.

В результате создания данной системы требования, изложенные в постановке задачи, выполнены.

У Постгреса множество возможностей. Созданный с использованием объектно-реляционной модели, он поддерживает сложные структуры и широкий спектр встроенных и определяемых пользователем типов данных. Он обеспечивает расширенную ёмкость данных и заслужил доверие бережным отношением к целостности данных. Возможно, вам не понадобятся все те продвинутые функции хранения данных, которые мы исследовали в этой статье, но, поскольку потребности могут быстро возрасти, есть несомненное преимущество в том, чтобы иметь всё это под рукой.

ПЕРЕЧЕНЬ ССЫЛОК

1. PostgreSQL [Электронный ресурс] Режим доступа:
<https://ru.wikipedia.org/wiki/PostgreSQL>, свободный.- Загл. с экрана.
2. Триггер [Электронный ресурс] Режим доступа:
[https://ru.wikipedia.org/wiki/Триггер_\(базы_данных\)](https://ru.wikipedia.org/wiki/Триггер_(базы_данных)) , свободный.- Загл. с экрана.
3. Запрос [Электронный ресурс] Режим доступа:
<http://miheevag.narod.ru/db.htm> , свободный.- Загл. с экрана.
4. Уотсон, К. Visual C# 2008. Базовый курс / Карли Уотсон, Кристиан Нейгел, Якоб Хаммер Педерсен, Джон Д. Рид, Морган Скиннер, Эрик Уайт// «Вильямс», 2009.– 1216с.

ПРИЛОЖЕНИЕ А ТЕХНИЧЕСКОЕ ЗАДАНИЕ

ПРИЛОЖЕНИЕ Б ЛИСТНИГ ШАБЛОНОВ

← 50/571 → Поиск

id_plants	название	тип	город	страна
26	Schuppe LLC	ЗАО	Хабаровск	Украина
27	Torphy - Hartmann	государств...	Сочи	Казахста
28	Balistreri Group	частный	Тольятти	Украина
29	Collins Group	частный	Курск	Казахста
30	Sanford - Feil	ЗАО	Оренбург	Турекме
31	Daniel - Gulowski	частный	Оренбург	ДНР
32	Boyle - Dickinson	ЗАО	Владивосток	Россия
33	Koss - Hettinger	государств...	Санкт-Петербург	Украина
34	Monahan, Windler and Kiehn	частный	Владимир	Украина
35	Pfannerstill - Jones	государств...	Москва	Казахста
36	Purdy - Gulowski	государств...	Махачкала	Казахста
37	Kassulke - Deckow	частный	Красноярск	Турекме
38	Jaskolski, Schneider and Batz	государств...	Магнитогорск	Турекме
39	Dicki, Harris and Ebert	частный	Тольятти	Россия
40	Stiedemann - Gibson	государств...	Красноярск	Казахста
41	Nitzsche - Wunsch	ЗАО	Новокузнецк	Турекме

Заводы Вид топлива Заказы Заказчики Тип собственности Страны Запросы

Рисунок Б.1 – Главная форма клиентского приложения

Запросы

Экспорт в Excel

Заказы
Заводы
Заказчики
Заказы объём больше
Заказы объём меньше
Заказы за пр. месяцы
Заказы за пр. года
Заводы без заказов
Клиенты без заказов
Клиенты без заказов v2
ср. объём заказов

Рисунок Б.2 – Форма запросов

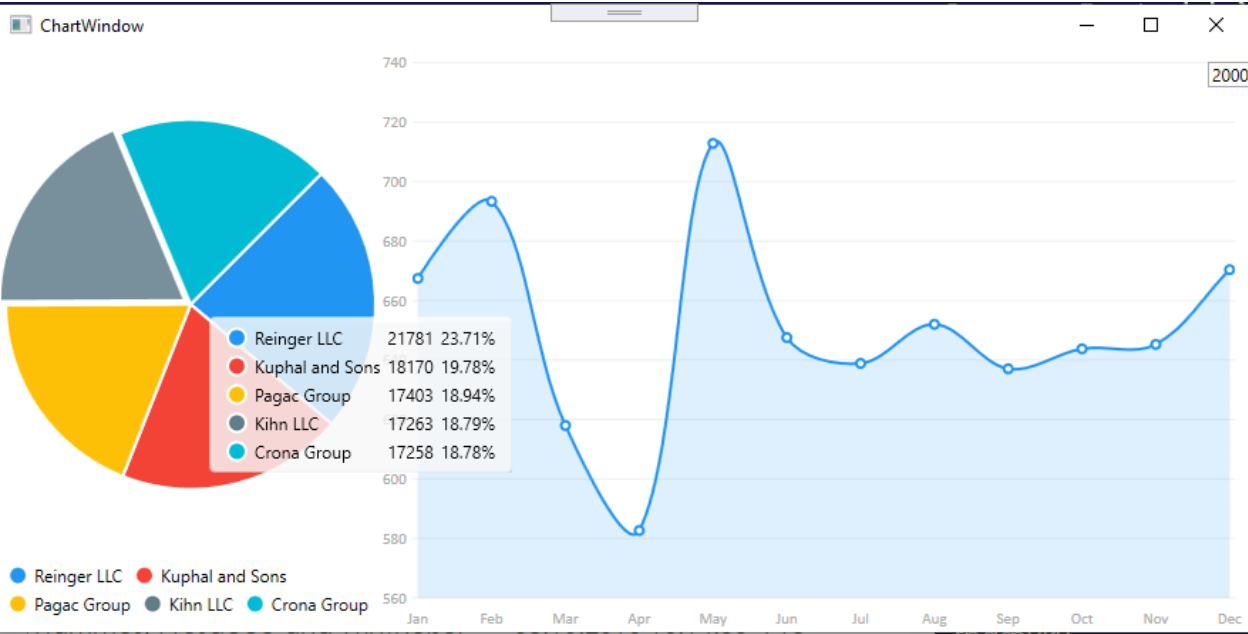


Рисунок Б.3 –Форма графиков

← 50/571 → Поиск

id_plants	название	тип	город	страна
26	Schuppe LLC	ЗАО	Хабаровск	Украина
27	Torphy - Hartmann	государств...	Сочи	Казахста
28	Balistreri Group	частный	Тольятти	Украина
29	Collins Group	частный	Курск	Казахста
30	Sanford - Feil	ЗАО	Оренбург	Турекме
31	Daniel - Gulgowski	частный	Оренбург	ДНР
32	Boyle - Dickinson	...	Владивосток	Россия
33	Koss - Hettinger	...	Санкт-Петербург	Украина
34	Monahan, Windler an	...	Владимир	Украина
35	Pfannerstill - Jones	...	Москва	Казахста
36	Purdy - Gulgowski	...	Махачкала	Казахста
37	Kassulke - Deckow	частный	Красноярск	Турекме
38	Jaskolski, Schneider and Batz	государств...	Магнитогорск	Турекме
39	Dicki, Harris and Ebert	частный	Тольятти	Россия
40	Stiedemann - Gibson	государств...	Красноярск	Казахста
41	Nitzsche - Wunsch	ЗАО	Новокузнецк	Турекме

Внимание!
вы уверены что хотите удалить запись?

Да Нет

Заводы Вид топлива Заказы Заказчики Тип собственности Страны Запросы

Рисунок Б.4– Всплывающее окно при удалении

← 50/29 → 🔍 Влад

id_plants	название	тип	город	страна
32	Boyle - Dickinson	ЗАО	Владивосток	Россия
34	Monahan, Windler and Kiehn	частный	Владимир	Украина
48	Bergstrom - Pagac	государств...	Владивосток	Казахстан
51	Pfannerstill, Larson and Anderson	частный	Владивосток	Турекмен
94	Kreiger, Ward and Lesch	государств...	Владимир	ДНР
116	Wolf, Breitenberg and Boehm	ЗАО	Владивосток	ДНР
140	Aufderhar, Wunsch and West	государств...	Владимир	Украина
145	Hyatt, Hahn and Christiansen	ЗАО	Владивосток	ДНР
164	Russel LLC	ЗАО	Владивосток	Турекмен
168	Heathcote Group	ЗАО	Владимир	ДНР
202	Bins LLC	частный	Владивосток	Казахстан
211	Schamberger, Stoltenberg and Shanahan	частный	Владимир	Казахстан
236	Abernathy, Herman and Morar	частный	Владивосток	Казахстан
250	Goldner and Sons	государств...	Владимир	Турекмен
366	Hyatt - Cummings	частный	Владивосток	Казахстан
369	Ankunding Group	ЗАО	Владимир	Россия

Заводы Вид топлива Заказы Заказчики Тип собственности Страны Запросы

Рисунок Б.5 – Фильтрация данных

Авторизуйтесь

Имя пользователя

admin

Пароль

•••••

Авторизоваться

Рисунок Б.6 – Форма Login



← 50/3 → 🔍 Поиск

id_order	Заказчики	вид топлива	объём	дата	Заводы
▶ 14340	Abernathy - ...	▼ а80	▼ 555	22.05.2020	Anderson LLC
14341	Effertz and S...	▼ дизельное	▼ 444	18.05.2020	Anderson - Breitenberg
14342	Keeling, Walk...	▼ керосин	▼ 333	21.05.2020	Ankunding Group
*		▼	▼		

Заводы	Вид топлива	Заказы	Заказчики	Тип собственности	Страны	Запросы
--------	-------------	--------	-----------	-------------------	--------	---------

Рисунок Б.7 – Заказы, созданные пользователем Дима (RLS)

← 50/4 → 🔍 Поиск

id_type	тип
1	государственный
2	частный
3	ЗАО
4	ООО
	aaaa
	

Error

✕

42501: нет доступа к отношению тип собственности

OK

Заводы	Вид топлива	Заказы	Заказчики	Тип собственности	Страны	Запросы
--------	-------------	--------	-----------	-------------------	--------	---------

Рисунок Б.8 – Попытка пользователем Дима работы с запрещенной таблицей

ПРИЛОЖЕНИЕ В ЛИСТИНГ СЕРВЕРНОГО ПРИЛОЖЕНИЯ

```

/*
Navicat Premium Data Transfer

Source Server        : Bober
Source Server Type   : PostgreSQL
Source Server Version : 100011
Source Host          : localhost:5432
Source Catalog       : postgres
Source Schema        : public

Target Server Type   : PostgreSQL
Target Server Version : 100011
File Encoding        : 65001

Date: 20/05/2020 18:45:35
*/

-----
-- Sequence structure for Заводы_id_seq
-----
DROP SEQUENCE IF EXISTS "public"."Заводы_id_seq";
CREATE SEQUENCE "public"."Заводы_id_seq"
INCREMENT 1
MINVALUE 1
MAXVALUE 2147483647
START 1
CACHE 1;

-----
-- Sequence structure for Заказчики_id_seq
-----
DROP SEQUENCE IF EXISTS "public"."Заказчики_id_seq";
CREATE SEQUENCE "public"."Заказчики_id_seq"
INCREMENT 1
MINVALUE 1
MAXVALUE 2147483647
START 1
CACHE 1;

-----
-- Sequence structure for Заказы_id_seq
-----
DROP SEQUENCE IF EXISTS "public"."Заказы_id_seq";
CREATE SEQUENCE "public"."Заказы_id_seq"
INCREMENT 1
MINVALUE 1
MAXVALUE 2147483647
START 1
CACHE 1;

-----
-- Sequence structure for вид топлива_id_seq
-----
DROP SEQUENCE IF EXISTS "public"."вид топлива_id_seq";
CREATE SEQUENCE "public"."вид топлива_id_seq"
INCREMENT 1
MINVALUE 1
MAXVALUE 2147483647
START 1
CACHE 1;

-----
-- Sequence structure for страны_id_seq
-----
DROP SEQUENCE IF EXISTS "public"."страны_id_seq";
CREATE SEQUENCE "public"."страны_id_seq"
INCREMENT 1
MINVALUE 1
MAXVALUE 2147483647
START 1
CACHE 1;

-----
-- Sequence structure for тип собственности_id_seq
-----
DROP SEQUENCE IF EXISTS "public"."тип
собственности_id_seq";
CREATE SEQUENCE "public"."тип собственности_id_seq"
INCREMENT 1
MINVALUE 1
MAXVALUE 2147483647
START 1
CACHE 1;

-----
-- Table structure for Заводы
-----
DROP TABLE IF EXISTS "public"."Заводы";
CREATE TABLE "public"."Заводы" (
  "id_plants" int4 NOT NULL GENERATED ALWAYS AS
  IDENTITY (
  INCREMENT 1
  MINVALUE 1
  MAXVALUE 2147483647
  START 1
  ),
  "название" varchar(255) COLLATE "pg_catalog"."default" NOT
  NULL,
  "город" varchar(255) COLLATE "pg_catalog"."default" NOT
  NULL,
  "id_type" int4 NOT NULL,
  "год" int4 NOT NULL,
  "телефон" "public"."phone" COLLATE "pg_catalog"."default"
  NOT NULL,
  "id_country" int4 NOT NULL,
  "id_fuel" int4 NOT NULL,
  "объём" int4 NOT NULL,
  "цена" int4 NOT NULL
)
;

-----
-- Table structure for Заказчики
-----
DROP TABLE IF EXISTS "public"."Заказчики";
CREATE TABLE "public"."Заказчики" (
  "id_client" int4 NOT NULL GENERATED ALWAYS AS
  IDENTITY (
  INCREMENT 1
  MINVALUE 1
  MAXVALUE 2147483647
  START 1
  ),
  "название" varchar(255) COLLATE "pg_catalog"."default" NOT
  NULL,
  "город" varchar(255) COLLATE "pg_catalog"."default" NOT
  NULL,
  "телефон" "public"."phone" COLLATE "pg_catalog"."default"
  NOT NULL
)
;

-----
-- Table structure for Заказы
-----
DROP TABLE IF EXISTS "public"."Заказы";
CREATE TABLE "public"."Заказы" (
  "id_order" int4 NOT NULL GENERATED ALWAYS AS
  IDENTITY (
  INCREMENT 1
  MINVALUE 1
  MAXVALUE 2147483647
  START 1
  ),
  "id_client" int4 NOT NULL,
  "id_fuel" int4 NOT NULL,
  "объём" int4 NOT NULL,

```

```

"дата" timestamp(6) NOT NULL,
"id_plants" int4 NOT NULL,
"login" varchar(64) COLLATE "pg_catalog"."default" NOT
NULL DEFAULT 'admin'::character varying
)
;

-----
-- Table structure for вид топлива
-----
DROP TABLE IF EXISTS "public"."вид топлива";
CREATE TABLE "public"."вид топлива" (
  "id_fuel" int4 NOT NULL GENERATED ALWAYS AS
IDENTITY (
INCREMENT 1
MINVALUE 1
MAXVALUE 2147483647
START 1
),
  "топливо" varchar(255) COLLATE "pg_catalog"."default" NOT
NULL
)
;

-----
-- Table structure for страны
-----
DROP TABLE IF EXISTS "public"."страны";
CREATE TABLE "public"."страны" (
  "id_country" int4 NOT NULL GENERATED ALWAYS AS
IDENTITY (
INCREMENT 1
MINVALUE 1
MAXVALUE 2147483647
START 1
),
  "страна" varchar(255) COLLATE "pg_catalog"."default" NOT
NULL
)
;

-----
-- Table structure for тип собственности
-----
DROP TABLE IF EXISTS "public"."тип собственности";
CREATE TABLE "public"."тип собственности" (
  "id_type" int4 NOT NULL GENERATED ALWAYS AS
IDENTITY (
INCREMENT 1
MINVALUE 1
MAXVALUE 2147483647
START 1
),
  "тип" varchar(255) COLLATE "pg_catalog"."default" NOT
NULL
)
;

-----
-- Function structure for avg_orders
-----
DROP FUNCTION IF EXISTS "public"."avg_orders";
CREATE OR REPLACE FUNCTION "public"."avg_orders"()
  RETURNS TABLE("объём" numeric) AS $BODY$
BEGIN
RETURN QUERY SELECT
AVG("объём") as "объём"
FROM "Заказы"
WHERE "объём" > _n
HAVING AVG("объём") > _m;
END $BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000;

-----
-- Function structure for avg_orders_bigger
-----
DROP FUNCTION IF EXISTS
"public"."avg_orders_bigger"("_n" int4, "_m" int4);
CREATE OR REPLACE FUNCTION
"public"."avg_orders_bigger"("_n" int4)
  RETURNS TABLE("объём" numeric) AS $BODY$
BEGIN
RETURN QUERY SELECT
AVG("объём") as "объём"
FROM "Заказы"
WHERE "объём" > _N;
END $BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000;

-----
-- Function structure for before_insert_orders_fun
-----
DROP FUNCTION IF EXISTS
"public"."before_insert_orders_fun"();
CREATE OR REPLACE FUNCTION
"public"."before_insert_orders_fun"()
  RETURNS "pg_catalog"."trigger" AS $BODY$
BEGIN
  NEW.login = CURRENT_USER;
  RETURN NEW;
END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;

-----
-- Function structure for buyers_last_months
-----
DROP FUNCTION IF EXISTS
"public"."buyers_last_months"("_months" int4);
CREATE OR REPLACE FUNCTION
"public"."buyers_last_months"("_months" int4)
  RETURNS TABLE("название" varchar) AS $BODY$
BEGIN
RETURN QUERY SELECT "Заказчики"."название" FROM
"Заказчики" WHERE id_client IN (SELECT id_client FROM
"Заказы" WHERE "Заказы"."дата" > now() - (concat(_months, '
months'))::interval)
ORDER BY "Заказчики"."название";
END $BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000;

-----
-- Function structure for clients_by_last_months
-----
DROP FUNCTION IF EXISTS
"public"."clients_by_last_months"("_months" int4);
CREATE OR REPLACE FUNCTION
"public"."clients_by_last_months"("_months" int4)
  RETURNS TABLE("название" varchar) AS $BODY$
BEGIN
RETURN QUERY SELECT "Заказчики"."название" FROM
"Заказчики" WHERE id_client IN (SELECT id_client FROM

```

```
"Заказы" WHERE "Заказы"."дата" > now() - (concat(_months, '
months'))::interval)
ORDER BY "Заказчики"."название";
END $BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000;
```

```
-- Function structure for gen_inc_client
```

```
DROP FUNCTION IF EXISTS "public"."gen_inc_client";
CREATE OR REPLACE FUNCTION "public"."gen_inc_client"()
RETURNS "pg_catalog"."trigger" AS $BODY$BEGIN
NEW.id_client = (SELECT MAX("Заказчики".id_client)+1
FROM "Заказчики");
RETURN NEW;
END$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
```

```
-- Function structure for gen_inc_country
```

```
DROP FUNCTION IF EXISTS "public"."gen_inc_country";
CREATE OR REPLACE FUNCTION
"public"."gen_inc_country"()
RETURNS "pg_catalog"."trigger" AS $BODY$BEGIN
NEW.id_country = (SELECT MAX("страны".id_country)+1
FROM "страны");
RETURN NEW;
END$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
```

```
-- Function structure for gen_inc_fuel
```

```
DROP FUNCTION IF EXISTS "public"."gen_inc_fuel";
CREATE OR REPLACE FUNCTION "public"."gen_inc_fuel"()
RETURNS "pg_catalog"."trigger" AS $BODY$BEGIN
NEW.id_fuel = (SELECT MAX("вид топлива".id_fuel)+1
FROM "вид топлива");
RETURN NEW;
END$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
```

```
-- Function structure for gen_inc_orders
```

```
DROP FUNCTION IF EXISTS "public"."gen_inc_orders";
CREATE OR REPLACE FUNCTION "public"."gen_inc_orders"()
RETURNS "pg_catalog"."trigger" AS $BODY$BEGIN
NEW.id_orders = (SELECT MAX("Заказы".id_orders)+1
FROM "Заказы");
RETURN NEW;
END$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
```

```
-- Function structure for gen_inc_plants
```

```
DROP FUNCTION IF EXISTS "public"."gen_inc_plants";
CREATE OR REPLACE FUNCTION "public"."gen_inc_plants"()
RETURNS "pg_catalog"."trigger" AS $BODY$BEGIN
NEW.id_plants = (SELECT MAX("Заводы".id_plants)+1
FROM "Заводы");
RETURN NEW;
END$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
```

```
-- Function structure for gen_inc_type
```

```
DROP FUNCTION IF EXISTS "public"."gen_inc_type";
```

```
CREATE OR REPLACE FUNCTION "public"."gen_inc_type"()
RETURNS "pg_catalog"."trigger" AS $BODY$BEGIN
NEW.id_type = (SELECT MAX("тип
собственности".id_type)+1 FROM "тип собственности");
RETURN NEW;
END$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
```

```
-- Function structure for no_buyers_last_months
```

```
DROP FUNCTION IF EXISTS
"public"."no_buyers_last_months"("_months" int4);
CREATE OR REPLACE FUNCTION
"public"."no_buyers_last_months"("_months" int4)
RETURNS TABLE("название" varchar) AS $BODY$
BEGIN
RETURN QUERY SELECT "Заказчики"."название" FROM
"Заказчики" WHERE id_client NOT IN (SELECT id_client
FROM "Заказы" WHERE "Заказы"."дата" > now() -
(concat(_months, ' months'))::interval)
ORDER BY "Заказчики"."название";
END $BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000;
```

```
-- Function structure for orders_bigger_V
```

```
DROP FUNCTION IF EXISTS
"public"."orders_bigger_V"("_count" int4);
CREATE OR REPLACE FUNCTION
"public"."orders_bigger_V"("_count" int4)
RETURNS TABLE("id_order" int4, "объём" int4, "дата"
timestamp, "топливо" varchar) AS $BODY$
BEGIN
RETURN QUERY SELECT "Заказы"."id_order",
"Заказы"."объём", "Заказы"."дата", "вид топлива"."топливо"
FROM "Заказы" NATURAL JOIN "вид топлива" WHERE
"Заказы"."объём" > _count
ORDER BY "Заказы"."объём" DESC;
END $BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000;
```

```
-- Function structure for orders_by_last_months
```

```
DROP FUNCTION IF EXISTS
"public"."orders_by_last_months"("_months" int4);
CREATE OR REPLACE FUNCTION
"public"."orders_by_last_months"("_months" int4)
RETURNS TABLE("id_order" int4, "дата" timestamp, "объём"
int4, "топливо" varchar) AS $BODY$
BEGIN
RETURN QUERY SELECT
"Заказы"."id_order",
"Заказы"."дата",
"Заказы"."объём",
"вид топлива"."топливо"
FROM "Заказы"
NATURAL JOIN "вид топлива"
WHERE "Заказы"."дата" > now() - (concat(_months, '
months'))::interval
ORDER BY "Заказы"."дата" DESC;
END $BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000;
```

```
-- Function structure for orders_by_last_months_index
```

```
DROP FUNCTION IF EXISTS
"public"."orders_by_last_months_index"("_months" int4);
```

```

CREATE OR REPLACE FUNCTION
"public"."orders_by_last_months_index"("_months" int4)
RETURNS TABLE("количество" int8) AS $BODY$
BEGIN
RETURN QUERY SELECT COUNT(*) as "количество" FROM
"Заказы" WHERE "Заказы"."дата" > now() - (concat(_months, '
months'))::interval;
END $BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000;

```

```
-- Function structure for orders_by_last_years
```

```

DROP FUNCTION IF EXISTS
"public"."orders_by_last_years"("_years" int4);
CREATE OR REPLACE FUNCTION
"public"."orders_by_last_years"("_years" int4)
RETURNS TABLE("id_order" int4, "дата" timestamp, "объём"
int4, "топливо" varchar) AS $BODY$
BEGIN
RETURN QUERY SELECT
"Заказы"."id_order",
"Заказы"."дата",
"Заказы"."объём",
"вид топлива"."топливо"
FROM "Заказы"
NATURAL JOIN "вид топлива"
WHERE "Заказы"."дата" > now() - (concat(_years, '
years'))::interval
ORDER BY "Заказы"."дата" DESC;
END $BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000;

```

```
-- Function structure for orders_smaller_V
```

```

DROP FUNCTION IF EXISTS
"public"."orders_smaller_V"("_count" int4);
CREATE OR REPLACE FUNCTION
"public"."orders_smaller_V"("_count" int4)
RETURNS TABLE("id_order" int4, "объём" int4, "дата"
timestamp, "топливо" varchar) AS $BODY$
BEGIN
RETURN QUERY SELECT "Заказы"."id_order",
"Заказы"."объём", "Заказы"."дата", "вид топлива"."топливо"
FROM "Заказы" NATURAL JOIN "вид топлива" WHERE
"Заказы"."объём" < _count
ORDER BY "Заказы"."объём" DESC;
END $BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000;

```

```
-- Function structure for orders_sum_where_V_bigger
```

```

DROP FUNCTION IF EXISTS
"public"."orders_sum_where_V_bigger"("_n" int4);
CREATE OR REPLACE FUNCTION
"public"."orders_sum_where_V_bigger"("_n" int4)
RETURNS TABLE("объём" int8) AS $BODY$
BEGIN
RETURN QUERY SELECT sum("Заказы"."объём")
FROM "Заказы" WHERE "Заказы"."объём" > _n;
END $BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000;

```

```
-- Function structure for orders_sum_where_V_fuel_like
```

```

DROP FUNCTION IF EXISTS
"public"."orders_sum_where_V_fuel_like"("_fuel" varchar);

```

```

CREATE OR REPLACE FUNCTION
"public"."orders_sum_where_V_fuel_like"("_fuel" varchar)
RETURNS TABLE("объём" int8) AS $BODY$
BEGIN
RETURN QUERY SELECT sum("Заказы"."объём")
FROM "Заказы" NATURAL JOIN "вид топлива" WHERE "вид
топлива"."топливо" like concat('%', _fuel,'%');
END $BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000;

```

```
-- Function structure for sum_zakazi_by_client
```

```

DROP FUNCTION IF EXISTS
"public"."sum_zakazi_by_client"("_count" int8);
CREATE OR REPLACE FUNCTION
"public"."sum_zakazi_by_client"("_count" int8)
RETURNS TABLE("Клиент" varchar, "сумма" int8) AS
$BODY$
BEGIN
--Последние заказы фирмы 'название фирмы'
RETURN QUERY SELECT "Заказчики"."название" as
"Клиент", SUM(объём) as "сумма" FROM "Заказчики"
NATURAL JOIN "Заказы"
GROUP BY название
HAVING SUM(объём) > _count
ORDER BY "сумма" DESC;
END $BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000;

```

```
-- Function structure for zakazi_avg_by_month_year
```

```

DROP FUNCTION IF EXISTS
"public"."zakazi_avg_by_month_year"("_month" int4, "_year"
int4);
CREATE OR REPLACE FUNCTION
"public"."zakazi_avg_by_month_year"("_month" int4, "_year"
int4)
RETURNS TABLE("ср.цена" numeric) AS $BODY$
BEGIN
RETURN QUERY SELECT ROUND(AVG("Заказы"."объём"),
2) as "ср.цена" FROM "Заказы"
WHERE EXTRACT(YEAR from дата) = _year AND
EXTRACT(MONTH from дата) = _month;
END $BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000;

```

```
-- View structure for Заказчики_без_заказов
```

```

DROP VIEW IF EXISTS "public"."Заказчики_без_заказов";
CREATE VIEW "public"."Заказчики_без_заказов" AS SELECT
"Заказчики"."название",
"Заказчики"."город"
FROM ("Заказы"
RIGHT JOIN "Заказчики" ON (("Заказы".id_client =
"Заказчики".id_client)))
WHERE ("Заказы".id_order IS NULL);

```

```
-- View structure for Заводы_без_заказов
```

```

DROP VIEW IF EXISTS "public"."Заводы_без_заказов";
CREATE VIEW "public"."Заводы_без_заказов" AS SELECT
"Заводы"."название"
FROM ("Заводы"
LEFT JOIN "Заказы" ON (("Заказы".id_plants =
"Заводы".id_plants)))
WHERE ("Заказы".id_order IS NULL);

```

```

-- View structure for emulate_left_join
-----
DROP VIEW IF EXISTS "public"."emulate_left_join";
CREATE VIEW "public"."emulate_left_join" AS SELECT
"Заказчики"."название" AS "заказчик",
  ( SELECT "Заказы".id_order
    FROM "Заказы"
    WHERE ("Заказы".id_client = "Заказчики".id_client)
    LIMIT 1) AS "заказы"
FROM "Заказчики"
WHERE (( SELECT "Заказы".id_order
  FROM "Заказы"
  WHERE ("Заказы".id_client = "Заказчики".id_client)
  LIMIT 1) IS NULL);

-----
-- View structure for podzapros_itog
-----
DROP VIEW IF EXISTS "public"."podzapros_itog";
CREATE VIEW "public"."podzapros_itog" AS SELECT
bb."название",
  ( SELECT round(avg(aa."объём"), 2) AS round
    FROM ("Заказы" aa
      JOIN "Заказчики" cc USING (id_client))
    WHERE ((cc."название")::text = (bb."название")::text)) AS
"среднее",
bb."телефон"
FROM "Заказчики" bb
ORDER BY bb."название";

-----
-- View structure for union_view
-----
DROP VIEW IF EXISTS "public"."union_view";
CREATE VIEW "public"."union_view" AS SELECT "вид
топлива".id_fuel,
  "вид топлива"."топливо"
FROM "вид топлива"
UNION
SELECT "тип собственности".id_type AS id_fuel,
  "тип собственности"."тип" AS "топливо"
FROM "тип собственности";

-----
-- View structure for заказы_natural_join
-----
DROP VIEW IF EXISTS "public"."заказы_natural_join";
CREATE VIEW "public"."заказы_natural_join" AS SELECT
"Заказы"."объём",
"Заказы"."дата",
"Заказы".id_order,
"Заводы"."название",
"Заводы"."город",
"вид топлива"."топливо",
"Заводы"."цена"
FROM (("Заказы"
  JOIN "вид топлива" ON (("Заказы".id_fuel = "вид
топлива".id_fuel)))
  JOIN "Заводы" ON (((("Заводы".id_fuel = "вид
топлива".id_fuel) AND ("Заказы".id_plants =
"Заводы".id_plants))));

-----
-- View structure for заводы_natural_join
-----
DROP VIEW IF EXISTS "public"."заводы_natural_join";
CREATE VIEW "public"."заводы_natural_join" AS SELECT
"вид топлива"."топливо",
"Заводы"."название",
"Заводы"."город",
"Заводы"."объём",
"Заводы"."год",
"Заводы"."телефон",
"Заводы"."цена",
"страны"."страна"
FROM (("Заводы"
  JOIN "вид топлива" ON (("Заводы".id_fuel = "вид
топлива".id_fuel)))
  JOIN "страны" ON (((("Заводы".id_country =
"страны".id_country))));

-----
-- View structure for заказчики_natural_join
-----
DROP VIEW IF EXISTS "public"."заказчики_natural_join";
CREATE VIEW "public"."заказчики_natural_join" AS SELECT
"Заказчики"."название",
"Заказы"."дата",
"Заказы"."объём"
FROM ("Заказчики"
  JOIN "Заказы" ON (((("Заказы".id_client =
"Заказчики".id_client))));

-----
-- View structure for avg_V_orders
-----
DROP VIEW IF EXISTS "public"."avg_V_orders";
CREATE VIEW "public"."avg_V_orders" AS SELECT
avg("Заказы"."объём") AS "ср.объём"
FROM "Заказы";

-----
-- View structure for sum_V_orders
-----
DROP VIEW IF EXISTS "public"."sum_V_orders";
CREATE VIEW "public"."sum_V_orders" AS SELECT
sum("Заказы"."объём") AS "сумма"
FROM "Заказы";

-----
-- View structure for case_view
-----
DROP VIEW IF EXISTS "public"."case_view";
CREATE VIEW "public"."case_view" AS SELECT
a."название",
a."дата",
CASE
  WHEN (a."дата" > (now() - '1 mon'::interval)) THEN
'недавно':text
  ELSE 'давно':text
END AS "case"
FROM ("Заказчики"
  JOIN "Заказы" USING (id_client)) a
WHERE (a."дата" = ( SELECT max(b."дата") AS max
  FROM "Заказы" b
  WHERE (a.id_client = b.id_client)))
ORDER BY a."дата" DESC;

-----
-- Alter sequences owned by
-----
ALTER SEQUENCE "public"."Заводы_id_seq"
OWNED BY "public"."Заводы".id_plants";
SELECT setval("public"."Заводы_id_seq", 626, true);
ALTER SEQUENCE "public"."Заказчики_id_seq"
OWNED BY "public"."Заказчики".id_client";
SELECT setval("public"."Заказчики_id_seq", 1415, true);
ALTER SEQUENCE "public"."Заказы_id_seq"
OWNED BY "public"."Заказы".id_order";
SELECT setval("public"."Заказы_id_seq", 14343, true);
ALTER SEQUENCE "public"."вид топлива_id_seq"
OWNED BY "public"."вид топлива".id_fuel";
SELECT setval("public"."вид топлива_id_seq", 8, true);
ALTER SEQUENCE "public"."страны_id_seq"
OWNED BY "public"."страны".id_country";
SELECT setval("public"."страны_id_seq", 8, true);
ALTER SEQUENCE "public"."тип собственности_id_seq"
OWNED BY "public"."тип собственности".id_type";
SELECT setval("public"."тип собственности_id_seq", 7, true);

-----
-- Uniques structure for table Заводы
-----
ALTER TABLE "public"."Заводы" ADD CONSTRAINT
"Заводы_название_key" UNIQUE ("название");

```



```

-----
-- Primary Key structure for table Заводы
-----
ALTER TABLE "public"."Заводы" ADD CONSTRAINT
"Заводы_pkey" PRIMARY KEY ("id_plants");

-----
-- Primary Key structure for table Заказчики
-----
ALTER TABLE "public"."Заказчики" ADD CONSTRAINT
"Заказчики_pkey" PRIMARY KEY ("id_client");

-----
-- Indexes structure for table Заказы
-----
CREATE INDEX "date_index" ON "public"."Заказы" USING
btree (
    "дата" "pg_catalog"."timestamp_ops" ASC NULLS LAST
);

-----
-- Triggers structure for table Заказы
-----
CREATE TRIGGER "before_insert_orders" BEFORE INSERT
ON "public"."Заказы"
FOR EACH ROW
EXECUTE PROCEDURE "public"."before_insert_orders_fun"();

-----
-- Primary Key structure for table Заказы
-----
ALTER TABLE "public"."Заказы" ADD CONSTRAINT
"Заказы_pkey" PRIMARY KEY ("id_order");

-----
-- Primary Key structure for table вид топлива
-----
ALTER TABLE "public"."вид топлива" ADD CONSTRAINT
"вид топлива_pkey" PRIMARY KEY ("id_fuel");

-----
-- Primary Key structure for table страны

```

```

-----
ALTER TABLE "public"."страны" ADD CONSTRAINT
"страны_pkey" PRIMARY KEY ("id_country");

-----
-- Primary Key structure for table тип собственности
-----
ALTER TABLE "public"."тип собственности" ADD
CONSTRAINT "тип собственности_pkey" PRIMARY KEY
("id_type");

-----
-- Foreign Keys structure for table Заводы
-----
ALTER TABLE "public"."Заводы" ADD CONSTRAINT
"Заводы_id_country_fkey" FOREIGN KEY ("id_country")
REFERENCES "public"."страны" ("id_country") ON DELETE
NO ACTION ON UPDATE NO ACTION;
ALTER TABLE "public"."Заводы" ADD CONSTRAINT
"Заводы_id_fuel_fkey" FOREIGN KEY ("id_fuel")
REFERENCES "public"."вид топлива" ("id_fuel") ON DELETE
NO ACTION ON UPDATE NO ACTION;
ALTER TABLE "public"."Заводы" ADD CONSTRAINT
"Заводы_id_type_fkey" FOREIGN KEY ("id_type")
REFERENCES "public"."тип собственности" ("id_type") ON
DELETE NO ACTION ON UPDATE NO ACTION;

-----
-- Foreign Keys structure for table Заказы
-----
ALTER TABLE "public"."Заказы" ADD CONSTRAINT
"Заказы_id_client_fkey" FOREIGN KEY ("id_client")
REFERENCES "public"."Заказчики" ("id_client") ON DELETE
NO ACTION ON UPDATE NO ACTION;
ALTER TABLE "public"."Заказы" ADD CONSTRAINT
"Заказы_id_fuel_fkey" FOREIGN KEY ("id_fuel")
REFERENCES "public"."вид топлива" ("id_fuel") ON DELETE
NO ACTION ON UPDATE NO ACTION;
ALTER TABLE "public"."Заказы" ADD CONSTRAINT
"Заказы_id_plants_fkey" FOREIGN KEY ("id_plants")
REFERENCES "public"."Заводы" ("id_plants") ON DELETE NO
ACTION ON UPDATE NO ACTION;

```

ПРИЛОЖЕНИЕ Г ЛИСТИНГ КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

	Код
<pre> public partial class LoginWindow : Window { public LoginWindow() { InitializeComponent(); } private void Move_Window(object sender, MouseButtonEventArgs e) { if (e.ChangedButton == MouseButton.Left) this.DragMove(); } private void btn_exit_Click(object sender, RoutedEventArgs e) { this.Close(); } private void btn_minimize_Click(object sender, RoutedEventArgs e) { this.WindowState = WindowState.Minimized; } private void btn_authorization_Click(object sender, RoutedEventArgs e) { string database = "course2020"; string port = "5432"; string host = "127.0.0.1"; string login = txb_login.Text; string password = txb_password.Password; string connection_string = \$"Server={host}; Port={port}; User Id={login}; Password={password}; Database={database}"; NpgsqlConnection connection = new NpgsqlConnection(connection_string); try { connection.Open(); } catch (Npgsql.PostgresException ex) { MessageBox.Show("Ошибка авторизации!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); return; } //FF3E59A0 MainWindow window = new MainWindow(connection_string, login); window.Show(); this.Close(); } } public partial class QueryWindow : Window { string connection_string; public QueryWindow(string connection_string) { InitializeComponent(); this.connection_string = connection_string; dataGridview1.BorderStyle = System.Windows.Forms.BorderStyle.None; dataGridview1.AlternatingRowsDefaultCellStyle.BackColor = System.Drawing.Color.FromArgb(238, 239, 249); dataGridview1.CellBorderStyle = System.Windows.Forms.DataGridViewCellBorderStyle.SingleHorizontal; dataGridview1.DefaultCellStyle.SelectionBackColor = System.Drawing.Color.DarkTurquoise; dataGridview1.DefaultCellStyle.SelectionForeColor = System.Drawing.Color.WhiteSmoke; dataGridview1.BackgroundColor = System.Drawing.Color.White; dataGridview1.EnableHeadersVisualStyles = false; dataGridview1.ColumnHeaderBorderStyle = System.Windows.Forms.DataGridViewHeaderBorderStyle.None; dataGridview1.ColumnHeaderDefaultCellStyle.BackColor = System.Drawing.Color.FromArgb(20, 25, 72); dataGridview1.ColumnHeaderDefaultCellStyle.ForeColor = System.Drawing.Color.White; } private void Move_Window(object sender, MouseButtonEventArgs e) { if (e.ChangedButton == MouseButton.Left) this.DragMove(); } private void btn_exit_Click(object sender, RoutedEventArgs e) { this.Close(); } </pre>	<pre> private void move_rect(object sender) { //rect.Margin = new Thickness((sender as Button).Margin.Left, (sender as Button).Margin.Top, 0, 0); //rect.Height = (sender as Button).Height; } private void btn_minimize_Click(object sender, RoutedEventArgs e) { this.WindowState = WindowState.Minimized; } void FillGridFromDataSet(DataSet dataSet) { try { dataGridview1.Columns.Clear(); for (int i = 0; i < dataSet.Tables[0].Columns.Count; i++) { var col = dataSet.Tables[0].Columns[i]; dataGridview1.Columns.Add(col.ColumnName, col.Caption); if (i == 0) dataGridview1.Columns[col.ColumnName].ReadOnly = true; } for (int i = 0; i < dataSet.Tables[0].Rows.Count; i++) dataGridview1.Rows.Add(dataSet.Tables[0].Rows[i].ItemArray); for (int i = 0; i < dataGridview1.Columns.Count; i++) { dataGridview1.Columns[i].AutoSizeMode = System.Windows.Forms.DataGridViewAutoSizeColumnMode.AllCells; if (i == dataGridview1.Columns.Count - 1) dataGridview1.Columns[i].AutoSizeMode = System.Windows.Forms.DataGridViewAutoSizeColumnMode.Fill; } } catch (Exception e) { } } DataSet FillDataSetFromDB(string sql) { try { NpgsqlConnection connection = new NpgsqlConnection(connection_string); var select = sql; var dataAdapter = new NpgsqlDataAdapter(select, connection); var dataSet = new DataSet(); dataAdapter.Fill(dataSet); connection.Close(); return dataSet; } catch (Exception e) { MessageBox.Show(e.Message, "Error"); return null; } } private void btn_orders_Click(object sender, RoutedEventArgs e) { string sql = "SELECT * FROM Заказы_naturaljoin"; FillGridFromDataSet(FillDataSetFromDB(sql)); } private void btn_orders_by_firm_Click(object sender, RoutedEventArgs e) { var dialog = new MyDialog("Фирма", "Введите фирму"); if (dialog.ShowDialog() == true) { string sql = \$"SELECT * FROM zakazi_by_firm('{dialog.ResponseText}')"; FillGridFromDataSet(FillDataSetFromDB(sql)); } } private void btn_hospitals_Click(object sender, RoutedEventArgs e) { string sql = "SELECT * FROM Заводы_naturaljoin"; FillGridFromDataSet(FillDataSetFromDB(sql)); } private void btn_orders_by_hospital_Click(object sender, RoutedEventArgs e) { var dialog = new MyDialog("Больница", "Введите Номер завода"); if (dialog.ShowDialog() == true) { try </pre>

```

        {
            string sql = $"SELECT * FROM
zakazi_by_hospital({dialog.ResponseText})";
            FillGridFromDataSet(FillDataSetFromDB(sql));
        }
        catch (Exception ee)
        {
            MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK,
            MessageBoxImage.Error);
        }
    }

    private void btn_orders_by_last_months_Click(object sender, RoutedEventArgs e)
    {
        var dialog = new MyDialog("Месяцы", "Введите за сколько последних
месяцев вернуть заказы");
        if (dialog.ShowDialog() == true)
        {
            try
            {
                string sql = $"SELECT * FROM
zakazi_by_last_months({dialog.ResponseText})";
                FillGridFromDataSet(FillDataSetFromDB(sql));
            }
            catch (Exception ee)
            {
                MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK,
                MessageBoxImage.Error);
            }
        }
    }

    private void btn_orders_by_last_years_Click(object sender, RoutedEventArgs e)
    {
        var dialog = new MyDialog("Месяцы", "Введите за сколько последних лет
вернуть заказы");
        if (dialog.ShowDialog() == true)
        {
            try
            {
                string sql = $"SELECT * FROM
zakazi_by_last_years({dialog.ResponseText})";
                FillGridFromDataSet(FillDataSetFromDB(sql));
            }
            catch (Exception ee)
            {
                MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK,
                MessageBoxImage.Error);
            }
        }
    }

    private void btn_hospitals_wiithout_orders_Click(object sender, RoutedEventArgs
e)
    {
        string sql = "SELECT * FROM view_Заводы_без_заказов";
        FillGridFromDataSet(FillDataSetFromDB(sql));
    }

    private void btn_firms_without_orders_Click(object sender, RoutedEventArgs e)
    {
        string sql = "SELECT * FROM view_фирмы_без_заказов";
        FillGridFromDataSet(FillDataSetFromDB(sql));
    }

    private void btn_firms_without_orders_left_join_Click(object sender,
RoutedEventArgs e)
    {
        string sql = "SELECT * FROM view_emulate_left_join";
        FillGridFromDataSet(FillDataSetFromDB(sql));
    }

    private void btn_avg_orders_by_hospital_Click(object sender, RoutedEventArgs e)
    {
        string sql = "SELECT * FROM средняя_стоимость_заказа_для_кажд";
        FillGridFromDataSet(FillDataSetFromDB(sql));
    }

    private void btn_sum_orders_Click(object sender, RoutedEventArgs e)
    {
        string sql = "SELECT * FROM сумма_заказов";
        FillGridFromDataSet(FillDataSetFromDB(sql));
    }

    private void btn_avg_bigger_Click(object sender, RoutedEventArgs e)
    {
        var dialog = new MyDialog("Цена", "Введите больше чего");
        if (dialog.ShowDialog() == true)
        {
            try
            {
                string sql = $"SELECT * FROM
zakazi_avg_bigger({dialog.ResponseText})";
                FillGridFromDataSet(FillDataSetFromDB(sql));
            }
            catch (Exception ee)
            {
                MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK,
                MessageBoxImage.Error);
            }
        }
    }

```

```

    private void btn_firms_in_country_Click(object sender, RoutedEventArgs e)
    {
        var dialog = new MyDialog("Страна", "Введите маску страны");
        if (dialog.ShowDialog() == true)
        {
            string sql = $"SELECT * FROM
firms_country_like({dialog.ResponseText})";
            FillGridFromDataSet(FillDataSetFromDB(sql));
        }
    }

    private void btn_count_firms_by_last_monthes_Click(object sender,
RoutedEventArgs e)
    {
        var dialog = new MyDialog("Месяцы", "Введите за сколько последних
месяцев вернуть заказы");
        if (dialog.ShowDialog() == true)
        {
            try
            {
                string sql = $"SELECT * FROM
zakazi_count_by_last_month({dialog.ResponseText})";
                FillGridFromDataSet(FillDataSetFromDB(sql));
            }
            catch (Exception ee)
            {
                MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK,
                MessageBoxImage.Error);
            }
        }
    }

    private void btn_sum_zakazi_by_firm_where_Click(object sender,
RoutedEventArgs e)
    {
        var dialog = new MyDialog("количество", "Введите количество товаров для
данных");
        if (dialog.ShowDialog() == true)
        {
            try
            {
                string sql = $"SELECT * FROM
sum_zakazi_by_firm_where({dialog.ResponseText})";
                FillGridFromDataSet(FillDataSetFromDB(sql));
            }
            catch (Exception ee)
            {
                MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK,
                MessageBoxImage.Error);
            }
        }
    }

    private void btn_sum_zakazi_by_firm_where2_Click(object sender,
RoutedEventArgs e)
    {
        var dialog = new MyDialog("количество", "Введите количество товаров для
данных");
        var dialog2 = new MyDialog("количество", "Введите количество товаров для
группы");
        if (dialog.ShowDialog() == true && dialog2.ShowDialog() == true)
        {
            try
            {
                string sql = $"SELECT * FROM
sum_zakazi_by_firm_where({dialog.ResponseText}, {dialog2.ResponseText})";
                FillGridFromDataSet(FillDataSetFromDB(sql));
            }
            catch (Exception ee)
            {
                MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK,
                MessageBoxImage.Error);
            }
        }
    }

    private void btn_itog_podzapros_Click(object sender, RoutedEventArgs e)
    {
        string sql = $"SELECT * FROM podzapros_itog()";
        FillGridFromDataSet(FillDataSetFromDB(sql));
    }

    private void btn_union_Click(object sender, RoutedEventArgs e)
    {
        string sql = $"SELECT * FROM union_view";
        FillGridFromDataSet(FillDataSetFromDB(sql));
    }

    private void btn_in_Click(object sender, RoutedEventArgs e)
    {
        var dialog = new MyDialog("Месяцы", "Введите интервал в месяцах");
        if (dialog.ShowDialog() == true)
        {
            try
            {
                string sql = $"SELECT * FROM
hospitals_by_last_months({dialog.ResponseText})";
                FillGridFromDataSet(FillDataSetFromDB(sql));
            }
            catch (Exception ee)
            {
                MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK,
                MessageBoxImage.Error);
            }
        }
    }

```

```

    }
}

private void btn_not_in_Click(object sender, RoutedEventArgs e)
{
    var dialog = new MyDialog("Месяцы", "Введите интервал в месяцах");
    if (dialog.ShowDialog() == true)
    {
        try
        {
            string sql = $"SELECT * FROM no_hospitals_by_last_months({dialog.ResponseText})";
            FillGridFromDataSet(FillDataSetFromDB(sql));
        }
        catch (Exception ee)
        {
            MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK,
            MessageBoxImage.Error);
        }
    }
}

private void btn_case_Click(object sender, RoutedEventArgs e)
{
    string sql = $"SELECT * FROM case_view";
    FillGridFromDataSet(FillDataSetFromDB(sql));
}

private void btn_excel_Click(object sender, RoutedEventArgs e)
{
    //Creating DataTable
    DataTable dt = new DataTable();

    //Adding the Columns
    foreach (System.Windows.Forms.DataGridColumn column in
    dataGridView1.Columns)
    {
        dt.Columns.Add(column.HeaderText);
    }

    //Adding the Rows
    foreach (System.Windows.Forms.DataGridRow row in
    dataGridView1.Rows)
    {
        dt.Rows.Add();
        foreach (System.Windows.Forms.DataGridCell cell in row.Cells)
        {
            if (cell.Value != null)
                dt.Rows[dt.Rows.Count - 1][cell.ColumnIndex] = cell.Value.ToString();
        }
    }
}

SaveFileDialog saveFileDialog1 = new SaveFileDialog();
saveFileDialog1.Filter = "xlsx files (*.xlsx)|*.xlsx|All files (*.*)|*.*";
saveFileDialog1.FilterIndex = 1;
saveFileDialog1.RestoreDirectory = true;

if (saveFileDialog1.ShowDialog() == true)
{
    using (XLWorkbook wb = new XLWorkbook())
    {
        wb.Worksheets.Add(dt, "Sheet");
        wb.SaveAs(saveFileDialog1.FileName);
        MessageBox.Show("Сохранено");
    }
}

private void btn_charts_Click(object sender, RoutedEventArgs e)
{
    ChartWindow cw = new ChartWindow(connection_string);
    cw.Show();
}
}
}
using System;
using System.Collections.Generic;
using System.Data;
using System.Drawing;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using Npgsql;

namespace BDcource2020
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        string connection_string;
        string login;
        public MainWindow(string connection_string, string login)
        {
            InitializeComponent();
            this.connection_string = connection_string;
            this.login = login;
            set_TextBox(tb_search, "Search");

            dataGridView1.Controls.Add(dtp);
            dtp.Visible = false;

```

```

            dtp.Format = System.Windows.Forms.DateTimePickerFormat.Custom;
            dtp.TextChanged += new EventHandler(dtp_TextChange);

            dataGridView1.BorderStyle = System.Windows.Forms.BorderStyle.None;
            dataGridView1.AlternatingRowsDefaultCellStyle.BackColor =
            Color.FromArgb(238, 239, 249);
            dataGridView1.CellBorderStyle =
            System.Windows.Forms.DataGridCellBorderStyle.SingleHorizontal;
            dataGridView1.DefaultCellStyle.SelectionBackColor = Color.DarkTurquoise;
            dataGridView1.DefaultCellStyle.SelectionForeColor = Color.WhiteSmoke;
            dataGridView1.BackgroundColor = Color.White;

            dataGridView1.EnableHeadersVisualStyles = false;
            dataGridView1.ColumnHeadersBorderStyle =
            System.Windows.Forms.DataGridHeaderBorderStyle.None;
            dataGridView1.ColumnHeadersDefaultCellStyle.BackColor =
            Color.FromArgb(20, 25, 72);
            dataGridView1.ColumnHeadersDefaultCellStyle.ForeColor = Color.White;

            btn_hospital_Click(btn_hospital, null);
            new Generator();
        }

        private void Move_Window(object sender, MouseButtonEventArgs e)
        {
            if (e.ChangedButton == MouseButton.Left)
                this.DragMove();
        }

        private void set_TextBox(TextBox tb, string text = "")
        {
            var converter = new System.Windows.Media.BrushConverter();
            tb.Text = text;
            tb.GotKeyboardFocus +=
            KeyboardFocusChangedEventHandler(tb_GotKeyboardFocus);
            tb.LostKeyboardFocus +=
            KeyboardFocusChangedEventHandler(tb_LostKeyboardFocus);
        }
        bool enabletb = false;
        private void tb_GotKeyboardFocus(object sender,
        KeyboardFocusChangedEventArgs e)
        {
            if (sender is TextBox)
            {
                //If nothing has been entered yet.
                var converter = new System.Windows.Media.BrushConverter();

                if (enabletb == false)
                {
                    ((TextBox)sender).Text = "";
                    ((TextBox)sender).Foreground = System.Windows.Media.Brushes.White;
                }
            }
        }

        private void tb_LostKeyboardFocus(object sender,
        KeyboardFocusChangedEventArgs e)
        {
            if (sender is TextBox)
            {
                enabletb = false;
                //If nothing was entered, reset default text.
                if (((TextBox)sender).Text.Trim().Equals(""))
                {
                    var converter = new System.Windows.Media.BrushConverter();
                    ((TextBox)sender).Foreground =
                    (System.Windows.Media.Brush)converter.ConvertFromString("#FFAAAAAA");//K
                    if (((TextBox)sender).Name == "tb_search")
                        ((TextBox)sender).Text = "Search";
                    else
                        ((TextBox)sender).Text = "DefaultText";
                }
            }
        }

        void FillGridFromDataSet(DataSet dataSet)
        {
            try
            {
                dataGridView1.Columns.Clear();
                for (int i = 0; i < dataSet.Tables[0].Columns.Count; i++)
                {
                    var col = dataSet.Tables[0].Columns[i];
                    dataGridView1.Columns.Add(col.ColumnName, col.Caption);
                    if (i == 0) dataGridView1.Columns[col.ColumnName].ReadOnly = true;
                }
                for (int i = 0; i < dataSet.Tables[0].Rows.Count; i++)
                    dataGridView1.Rows.Add(dataSet.Tables[0].Rows[i].ItemArray);

                for (int i = 0; i < dataGridView1.Columns.Count; i++)
                {
                    dataGridView1.Columns[i].AutoSizeMode =
                    System.Windows.Forms.DataGridAutoSizeMode.AllCells;
                    if (i == dataGridView1.Columns.Count - 1)
                        dataGridView1.Columns[i].AutoSizeMode =
                    System.Windows.Forms.DataGridAutoSizeMode.Fill;
                }
            }
            catch (Exception e)
            {

```

```

    }
}

DataSet FillDataSetFromDB(string sql)
{
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(connection_string);
        var select = sql;
        var dataAdapter = new NpgsqlDataAdapter(select, connection);
        var dataSet = new DataSet();
        dataAdapter.Fill(dataSet);
        connection.Close();
        return dataSet;
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Error");
        last_click(last_sender, null);
        return null;
    }
}

void execSql(string sql)
{
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(connection_string);
        connection.Open();
        NpgsqlCommand com = new NpgsqlCommand(sql, connection);
        com.ExecuteNonQuery();
        connection.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Error");
        last_click(last_sender, null);
    }
}

public DataTable fillComboBox(string fields, string table, string display_member)
{
    NpgsqlConnection connection = new NpgsqlConnection(connection_string);
    var select = "SELECT " + fields + " FROM \"" + table + "\" ORDER BY " +
display_member;

    var dataAdapter = new NpgsqlDataAdapter(select, connection);
    var dataTable = new DataTable();

    dataAdapter.Fill(dataTable);
    connection.Close();
    return dataTable;
}

/// <summary>
/// добавить ComboBox в таблицу
/// </summary>
/// <param name="name_cmbCol">имя колонки в таблице</param>
/// <param name="headerText">отображение колонки</param>
/// <param name="table">из какой таблицы брать данные</param>
/// <param name="ValueMember">какой член table брать за индекс</param>
/// <param name="DisplayMember">какой член table показывать</param>
/// <param name="Xindex">каким по порядку разместить</param>
void addComboBoxColumn(string name_cmbCol, string headerText, string table,
    string ValueMember, string DisplayMember, int Xindex)
{
    System.Windows.Forms.DataGridViewComboBoxColumn cmbCol = new
System.Windows.Forms.DataGridViewComboBoxColumn();
    cmbCol.HeaderText = headerText;
    cmbCol.Name = name_cmbCol;

    string fields = $"*";
    cmbCol.DataSource = fillComboBox(fields, table, DisplayMember);
    cmbCol.ValueMember = ValueMember;

    cmbCol.DisplayMember = DisplayMember;
    cmbCol.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

    dataGridView1.Columns.Add(cmbCol);
    dataGridView1.Columns[name_cmbCol].DisplayIndex = Xindex;
    dataGridView1.Columns[Xindex].AutoSizeMode
System.Windows.Forms.DataGridViewAutoSizeMode.AllCells;
}

void setComboBoxColumn(string column, string column_with_id)
{
    foreach (System.Windows.Forms.DataGridViewRow row in
dataGridView1.Rows)
    {
        row.Cells[column].Value = row.Cells[column_with_id].Value;
    }
}

public void StretchLastColumn()
{
    var lastColIndex = dataGridView1.Columns.Count - 1;
    var lastCol = dataGridView1.Columns[lastColIndex];
    lastCol.AutoSizeMode
System.Windows.Forms.DataGridViewAutoSizeMode.Fill;
}

void fix_cmb_width()
{
    if (dataGridView1.Columns.Count != 0)
    {
        dataGridView1.Columns[0].ReadOnly = true;
        for (int i = 0; i < dataGridView1.Columns.Count; i++)
        {
            var sub = dataGridView1.Columns[i].Name.Substring(0, 3);
            if (sub == "cmb")
            {
                dataGridView1.Columns[i].Width = 160;
            }
            else if (sub == "chk")
            {
            }
            else if (dataGridView1.Columns[i].Name == "дата")
            {
                dataGridView1.Columns[i].Width = 130;
            }
            else
            {
                dataGridView1.Columns[i].AutoSizeMode
System.Windows.Forms.DataGridViewAutoSizeMode.AllCells;
            }
        }
    }

    string generate_where()
    {
        string where = "";
        List<string> likes = new List<string>();
        for (int i = 0; i < fields.Length; i++)
            if (fields_types[i] == "string")
                likes.Add($" {fields[i]} LIKE '% {search}%'");

        if (likes.Count == 0 || search == "Search" || search == "")
            where = "1=1";
        else
        {
            bool first = true;
            for (int i = 0; i < likes.Count; i++)
            {
                if (first)
                {
                    where += likes[i];
                    first = false;
                }
                else
                    where += " OR " + likes[i];
            }
        }
        return where;
    }

    string generate_select(string[] fields, string table)
    {
        string where = generate_where();
        string result = "SELECT ";

        foreach (string field in fields)
            result += "\"" + field + "\", ";

        result = result.Remove(result.Length - 1);
        result += $"FROM \"{table}\" ";
        result += $"WHERE {where} ";
        result += $"ORDER BY {fields[0]} ";
        result += $"LIMIT {(page + 1) * 50} ";
        result += $"OFFSET {page * 50} ";
        int count = getCountRowsTable(table, where);
        status_text.Text = " " + ((page + 1) * 50).ToString();
        status_text.Text += "/" + count + " ";

        return result;
    }

    private void btn_exit_Click(object sender, RoutedEventArgs e)
    {
        this.Close();
    }

    private void move_rect(object sender)
    {
        rect.Margin = new Thickness((sender as Button).Margin.Left, (sender as
Button).Margin.Top, 0, 0);
        rect.Height = (sender as Button).Height;
    }

    private void btn_minimize_Click(object sender, RoutedEventArgs e)
    {
        this.WindowState = WindowState.Minimized;
    }

    string[] fields;
    string[] fields_types;
    string table;
    delegate void btn_click(object sender, RoutedEventArgs e);
    btn_click last_click;
    object last_sender;

    private void btn_hospital_Click(object sender, RoutedEventArgs e)
    {
        last_click = btn_hospital_Click;
        last_sender = sender;
    }
}

```

```

move_rect(sender);

table = "Заводы";
fields = new string[]
{
    "id_hospital", "номер", "id_type", "id_district", "год создания", "число мест",
    "количество врачей", "телефон"
};

fields_types = new string[]
{
    "int", "int", "int", "int", "int", "int", "string"
};

FillGridFromDataSet(FillDataSetFromDB(generate_select(fields, table)));

dataGridView1.Columns["id_type"].Visible = false;
dataGridView1.Columns["id_district"].Visible = false;
addComboBoxColumn("cmb_type", "тип", "Типы заводы", "id_type", "тип", 2);
addComboBoxColumn("cmb_district", "район", "Районы города", "id_district",
"район", 3);
setComboBoxColumn("cmb_type", "id_type");
setComboBoxColumn("cmb_district", "id_district");
fix_cmb_width();
}

private void btn_orders_Click(object sender, RoutedEventArgs e)
{
    last_click = btn_orders_Click;
    last_sender = sender;
    move_rect(sender);

    table = "Заказы";
    fields = new string[]
    {
        "id_order", "id_firm", "id_group", "id_form", "дата", "количество", "цена",
        "id_hospital"
    };

    fields_types = new string[]
    {
        "int", "int", "int", "int", "string", "int", "int", "int"
    };

    FillGridFromDataSet(FillDataSetFromDB(generate_select(fields, table)));

    dataGridView1.Columns["id_firm"].Visible = false;
    dataGridView1.Columns["id_group"].Visible = false;
    dataGridView1.Columns["id_form"].Visible = false;
    dataGridView1.Columns["id_hospital"].Visible = false;

    addComboBoxColumn("cmb_firm", "Фирма", "Фирма производитель",
    "id_firm", "название", 8);
    addComboBoxColumn("cmb_group", "Группа", "Фармокологическая
группа", "id_group", "группа", 2);
    addComboBoxColumn("cmb_form", "Форма", "Форма выпуска",
    "id_form", "форма", 3);
    addComboBoxColumn("cmb_hospital", "Больница", "Заводы",
    "id_hospital", "номер", 7);

    setComboBoxColumn("cmb_firm", "id_firm");
    setComboBoxColumn("cmb_group", "id_group");
    setComboBoxColumn("cmb_form", "id_form");
    setComboBoxColumn("cmb_hospital", "id_hospital");

    foreach (System.Windows.Forms.DataGridViewRow row in
dataGridView1.Rows)
    {
        if (row.Cells["дата"].Value != null && row.Cells["дата"].Value.ToString() !=
        "")
            row.Cells["дата"].Value = row.Cells["дата"].Value.ToString().Substring(0,
10);
    }

    fix_cmb_width();
}

private void btn_firms_Click(object sender, RoutedEventArgs e)
{
    last_click = btn_firms_Click;
    last_sender = sender;
    move_rect(sender);

    table = "Фирма производитель";
    fields = new string[]
    {
        "id_firm", "название", "id_type", "страна", "год"
    };

    fields_types = new string[]
    {
        "int", "string", "int", "string", "int"
    };

    FillGridFromDataSet(FillDataSetFromDB(generate_select(fields, table)));

    dataGridView1.Columns["id_type"].Visible = false;
    addComboBoxColumn("cmb_type", "Тип", "Тип собственности", "id_type",
"тип", 2);
    setComboBoxColumn("cmb_type", "id_type");
    fix_cmb_width();
}

```

```

private void btn_departs_Click(object sender, RoutedEventArgs e)
{
    last_click = btn_departs_Click;
    last_sender = sender;
    move_rect(sender);

    table = "Отделения";
    fields = new string[]
    {
        "id_depart", "отделение"
    };

    fields_types = new string[]
    {
        "int", "string"
    };

    FillGridFromDataSet(FillDataSetFromDB(generate_select(fields, table)));

    fix_cmb_width();
    StretchLastColumn();
}

private void btn_jiraf_Click(object sender, RoutedEventArgs e)
{
    last_click = btn_jiraf_Click;
    last_sender = sender;
    move_rect(sender);

    table = "Жираф";
    fields = new string[]
    {
        "id_jiraf", "id_hospital", "id_depart"
    };

    fields_types = new string[]
    {
        "int", "int", "int"
    };

    FillGridFromDataSet(FillDataSetFromDB(generate_select(fields, table)));

    dataGridView1.Columns["id_hospital"].Visible = false;
    dataGridView1.Columns["id_depart"].Visible = false;
    addComboBoxColumn("cmb_hospital", "Больница", "Заводы", "id_hospital",
"номер", 1);
    addComboBoxColumn("cmb_depart", "Отделение", "Отделения", "id_depart",
"отделение", 2);
    setComboBoxColumn("cmb_hospital", "id_hospital");
    setComboBoxColumn("cmb_depart", "id_depart");
    fix_cmb_width();
    StretchLastColumn();
}

private void btn_type_hosp_Click(object sender, RoutedEventArgs e)
{
    last_click = btn_type_hosp_Click;
    last_sender = sender;
    move_rect(sender);

    table = "Типы заводы";
    fields = new string[]
    {
        "id_type", "тип"
    };

    fields_types = new string[]
    {
        "int", "string"
    };

    FillGridFromDataSet(FillDataSetFromDB(generate_select(fields, table)));

    fix_cmb_width();
    StretchLastColumn();
}

private void btn_districts_Click(object sender, RoutedEventArgs e)
{
    last_click = btn_districts_Click;
    last_sender = sender;
    move_rect(sender);

    table = "Районы города";
    fields = new string[]
    {
        "id_district", "район"
    };

    fields_types = new string[]
    {
        "int", "string"
    };

    FillGridFromDataSet(FillDataSetFromDB(generate_select(fields, table)));

    fix_cmb_width();
    StretchLastColumn();
}

private void btn_form_Click(object sender, RoutedEventArgs e)
{
    last_click = btn_form_Click;
    last_sender = sender;
}

```

```

move_rect(sender);

table = "Форма выпуска";
fields = new string[]
{
    "id_form", "форма"
};

fields_types = new string[]
{
    "int", "string"
};

FillGridFromDataSet(FillDataSetFromDB(generate_select(fields, table)));

fix_cmb_width();
StretchLastColumn();
}

private void btn_type_sobs_Click(object sender, RoutedEventArgs e)
{
    last_click = btn_type_sobs_Click;
    last_sender = sender;
    move_rect(sender);

    table = "Тип собственности";
    fields = new string[]
    {
        "id_type", "тип"
    };

    fields_types = new string[]
    {
        "int", "string"
    };

    FillGridFromDataSet(FillDataSetFromDB(generate_select(fields, table)));

    fix_cmb_width();
    StretchLastColumn();
}

private void btn_farmo_group_Click(object sender, RoutedEventArgs e)
{
    last_click = btn_type_sobs_Click;
    last_sender = sender;
    move_rect(sender);

    table = "Фармокологическая группа";
    fields = new string[]
    {
        "id_group", "группа"
    };

    fields_types = new string[]
    {
        "int", "string"
    };

    FillGridFromDataSet(FillDataSetFromDB(generate_select(fields, table)));

    fix_cmb_width();
    StretchLastColumn();
}

private void btn_back_Click(object sender, RoutedEventArgs e)
{
    move_rect(sender);
    QueryWindow window = new QueryWindow(connection_string);
    window.Show();
    move_rect(last_sender);
}

bool readStrFromCell(System.Windows.Forms.DataGridViewCell cell, out string r)
{
    bool result = true;

    if (cell.Value != null)
    {
        r = cell.Value.ToString();
    }
    else
    {
        r = "";
        result = false;
    }

    return result;
}

bool readIntFromCell(System.Windows.Forms.DataGridViewCell cell, out string i)
{
    int temp = 0;
    i = "null";
    if (cell.Value != null)
    {
        if (int.TryParse(cell.Value.ToString(), out temp))
        {
            i = cell.Value.ToString();
        }
        else if (cell.Value.ToString() != "")
        {
            return false;
        }
    }
}

else
{
    return false;
}

return true;
}

private void dataGridView1_CellValueChanged(object sender,
System.Windows.Forms.DataGridViewCellEventArgs e)
{
    var row = dataGridView1.Rows[e.RowIndex];
    string name = dataGridView1.Columns[e.ColumnIndex].Name;

    if (name.Length > 3 && name.Substring(0, 3) == "cmb")
    {
        row.Cells["id" + name.Substring(3)].Value = row.Cells[name].Value;
        return;
    }

    string sql = "";
    if (row.Cells[fields[0]].Value == null)//insert
    {
        string values_string = "";
        for(int i=1; i<fields.Length; i++)
        {
            string temp = "";
            if (fields_types[i] == "int")
            {
                bool result = readIntFromCell(row.Cells[fields[i]], out temp);
                if (result == false && !fields[i].StartsWith("id"))
                {
                    MessageBox.Show("вы ввели не число!", "ОШИБКА",
                    MessageBoxButton.OK, MessageBoxImage.Error);
                    return;
                }
                values_string += $"{temp}";
            }
            else if (fields_types[i] == "string")
            {
                readStrFromCell(row.Cells[fields[i]], out temp);
                values_string += $"{temp}";
            }
            if (temp.Equals("null") || temp.Trim(' ').Equals(""))
                return;

            if (i != fields.Length-1)
                values_string += ", ";
        }

        sql = $"insert into \"{table}\"(\"";
        for (int i = 1; i < fields.Length; i++){
            if (i != fields.Length - 1)
            {
                sql += $"{fields[i]}\"";
            }
            else
            {
                sql += $"{fields[i]}\"";
                sql += ") VALUES (";
            }
        }
        sql += values_string;
        sql += ")";

        execSql(sql);
        last_click(last_sender, null);
    }
    else//update
    {
        sql = $" update \"{table}\" set ";
        for (int i = 1; i < fields.Length; i++)
        {
            string temp = "";
            if (fields_types[i] == "int")
            {
                bool result = readIntFromCell(row.Cells[fields[i]], out temp);
                if (result == false && !fields[i].StartsWith("id"))
                {
                    MessageBox.Show("вы ввели не число!", "ОШИБКА",
                    MessageBoxButton.OK, MessageBoxImage.Error);
                    return;
                }
            }
            else if (fields_types[i] == "string")
            {
                readStrFromCell(row.Cells[fields[i]], out temp);
                temp = $"{temp}";
            }
            if (i != fields.Length - 1)
            {
                sql += $"{fields[i]}\" = {temp}, ";
            }
            else
            {
                sql += $"{fields[i]}\" = {temp} ";
            }
        }
        sql += $" where \"{fields[0]}\" = {row.Cells[fields[0]].Value.ToString()}";
        execSql(sql);
        //last_click(last_sender, null);
    }
}

```

```

        System.Windows.Forms.DateTimePicker dtp = new
        System.Windows.Forms.DateTimePicker();
        Rectangle rectangle;

        private void dtp_TextChange(object sender, EventArgs e)
        {
            dataGridView1.CurrentCell.Value = dtp.Text.ToString();
            dtp.Visible = false;
        }

        private void dataGridView1_ColumnWidthChanged(object sender,
        System.Windows.Forms.DataGridColumnEventArgs e)
        {
            dtp.Visible = false;
        }

        private void dataGridView1_CellClick(object sender,
        System.Windows.Forms.DataGridCellEventArgs e)
        {
            if (e.RowIndex != -1 && e.ColumnIndex != -1)
            {
                if (table == "Заказы")
                {
                    if (dataGridView1.Columns[e.ColumnIndex].Name == "дата")
                    {
                        rectangle = dataGridView1.GetCellDisplayRectangle(e.ColumnIndex,
                        e.RowIndex, true);
                        dtp.Size = new System.Drawing.Size(rectangle.Width, rectangle.Height);
                        dtp.Location = new System.Drawing.Point(rectangle.X, rectangle.Y);
                        dtp.Visible = true;
                    }
                }
            }
        }

        private void dataGridView1_UserDeletingRow(object sender,
        System.Windows.Forms.DataGridRowCancelEventArgs e)
        {
            string sql = $"delete from {table} where \"{fields[0]}\" =
            {e.Row.Cells[fields[0]].Value.ToString()}";
            if (MessageBox.Show("вы уверены что хотите удалить запись?",
            "Внимание!", MessageBoxButtons.YesNo) == DialogResult.No)
            {
                e.Cancel = true;
                return;
            }
            execSql(sql);
        }

        private int getCountRowsTable(string table, string where="1=1")
        {
            NpgsqlConnection connection = new NpgsqlConnection(connection_string);
            connection.Open();

            var dataAdapter = new NpgsqlDataAdapter(
                string.Format("SELECT COUNT(*) FROM \"{0}\" WHERE {1}", table,
            where),
                connection);
            var dataTable = new DataTable();
            dataAdapter.Fill(dataTable);
            connection.Close();
            return int.Parse(dataTable.Rows[0][0].ToString());
        }

        string search = "";
        int page = 0;

        private void tb_search_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Key == Key.Enter)
            {
                search = tb_search.Text;
                last_click(last_sender, null);
            }
        }

        private void btn_left_Click(object sender, RoutedEventArgs e)
        {
            page = Math.Max(0, --page);
            last_click(last_sender, null);
        }

        private void btn_right_Click(object sender, RoutedEventArgs e)
        {
            page++;
            last_click(last_sender, null);
        }
    }
}

```


ПРИЛОЖЕНИЕ Д РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

При запуске программы запускается окно авторизации. Если ввести правильные данные авторизации, программа пропустит пользователя в окно с таблицами.

Для того, чтобы удалить запись в таблице, необходимо выбрать её(нажать, чтобы она стала выделенной) и нажать на клавиатуре кнопку «Delete», если же вы хотите удалить группу записей необходимо зажать «Ctrl» и нажать на записи(чтобы они стали выделенными), а после нажать на клавиатуре кнопку «Delete». Если нажатие на кнопку «Delete» было ошибочно, то сперва у Вас спросят: “Вы действительно хотите удалить запись?”, если нажмёте «Да», то записи удалятся, «Нет» - удаление не произойдёт.

Для того, чтобы добавить запись в таблицу, необходимо заполнить последнюю строчку в таблице. Если формат данных будет соблюден, запись добавится в базу данных.

Для того, чтобы редактировать запись в таблице, необходимо выбрать её в таблице, нажать на ячейку в нужной колонке два раза и ввести данные.

Для поиска, на основной форме необходимо ввести в поле текст и поиск произведётся автоматически.

Для отображения графика, нужно нажать на меню «Графики», которое находится вверху главной формы. В открывшейся форме нажать на кнопку «Кнопочка», возле каждого графика.

Для того, чтобы экспортировать результат запроса в Excel, необходимо зайти на форму с запросами и нажать «Экспортировать в Excel» и выбрать путь сохранения файла.

Для просмотра графиков в окне с запросами необходимо в самом низу нажать на кнопку с графиками.

ПРИЛОЖЕНИЕ Ж РУКОВОДСТВО АДМИНИСТРАТОРА

При запуске программы перед пользователем появляется меню авторизации. Корректно заполним логин и пароль в форме авторизации и нажать войти, то перед вами откроется меню с таблицами.

Для того, чтобы удалить одну запись в таблице, необходимо выбрать её (нажать, чтобы она стала выделенной) и нажать на клавиатуре кнопку «Delete», если же вы хотите удалить группу записей необходимо зажать «Ctrl» и нажать на записи (чтобы они стали выделенными), а после нажать на клавиатуре кнопку «Delete». Если нажатие на кнопку «Delete» было ошибочно, то сперва у Вас спросят: “Вы действительно хотите удалить запись?”, если нажмёте «Да», то записи удалятся, «Нет» - удаление не произойдёт.

Для того, чтобы добавить запись в таблицу, необходимо заполнить последнюю строчку в таблице. Если формат данных будет соблюден, запись добавится в базу данных.

Для того, чтобы редактировать запись в таблице, необходимо выбрать ее в таблице, нажать на ячейку в нужной колонке два раза и ввести данные.

Для поиска, на основной форме необходимо ввести в поле текст и поиск произведётся автоматически.

Для отображения графика, нужно нажать на меню «Графики», которое находится внизу формы запросов.

Для того, чтобы экспортировать результат запроса в Excel, необходимо зайти на форму с запросами и нажать «Экспортировать в Excel» и выбрать путь сохранения файла, а так же задать имя файлу.

Для просмотра графиков в окне с запросами необходимо в самом низу нажать на кнопку с графиками.