

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
**"ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ"**

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по курсу

“Программирование систем с серверами баз данных”

Тема работы: «Комиссионные магазины города»

Руководители: _____ Щедрин С.В.
_____ Ногтев Е.А.
_____ Филипишин Д.А.
(подпись) (дата)

Разработал: _____ Моргунов А.Г.
ст. гр. ПИ-186 (подпись) (дата)

РЕФЕРАТ

Пояснительная записка содержит 100 страниц, 98 рисунков, 6 источников, 7 приложения.

Объектом исследования данной курсовой работы являются комиссионные магазины города.

Цель работы – проектирование и реализация системы учета деятельности комиссионных магазинов, включающей в себя серверную и клиентскую части.

Результатом работы является многопользовательская база данных и клиентское приложение для взаимодействия с ней. Система может выполнять такие функции как: добавление, удаление, поиск записей в таблицах и справочниках БД, составление однотобличного отчета в Microsoft Excel, создание диаграммы, основанной на статистических данных, выполнение запросов различного уровня сложности. Система имеет интуитивно понятный интерактивный интерфейс. Различные роли имеют разграничение возможностей.

БАЗА ДАННЫХ, СУБД, C++, QT, SQL, КОМИССИОННЫЙ
МАГАЗИН, РАБОТНИК, ПРОЦЕДУРА, ТАБЛИЦА, POSTGRESQL,
БЕЗОПАСНОСТЬ

СОДЕРЖАНИЕ

Введение.....	5
1 Описание предметной области, постановка задачи	6
2 Система управления базой данных	7
3 Обоснование выбора инструментальные средств для написания клиентской части, проектирование структуры ПО	8
3.1 Не визуальные компоненты для работы с данными	9
3.2 Визуальные компоненты отображения данных.....	10
4 Проектирование базы данных в выбранной СУБД	12
4.1 Проектирование концептуальной модели БД.....	12
4.2 Создание таблиц, доменов, индексов	14
4.3 Разработка триггеров.....	18
4.4 Разработка ограничений доступа	21
4.4.1 Права доступа работника (роль worker)	22
4.4.2 Права доступа администратора (роль administrator)	23
4.4.3 Права доступа для владельца (роль holder).....	24
4.5 Проектирование запросов к базе данных	24
5 Разработка клиентского приложения.....	51
5.1 Работник(worker).....	53
5.2 Администратор (administrator).....	55
5.3 Владелец (holder).....	56
6 Тестирование разработанной информационной системы (в т.ч. включая защиту от несанкционированного доступа, каскадное удаление).....	60
6.1 Ошибки ввода.....	60
6.2 Каскадное удаление	61
Заключение	63
Список литературы	64
Приложение А. Техническое задание.....	65
Приложение Б. Листинг клиентского приложения	70

Приложение В. Листинг серверного приложения.....	76
Приложение Г. Антиплагиат	95
Приложение Д. Руководство работника.....	96
Приложение Е. Руководство Администратора	98
Приложение Ё. Руководство владельца.....	100

ВВЕДЕНИЕ

В современном мире технологии все больше и больше вливаются в жизнь людей. В том числе и в бизнес – сферу. Многие предприниматели и организации используют современные методы обработки информации. Основным средством для быстрой обработки информации являются базы данных. Это средство позволяет обрабатывать, выводить, хранить и манипулировать огромными объемами данных. Поэтому такое средство востребовано и используется повсеместно.

Система управления базами данных(СУБД) – это программный комплекс, обеспечивающий централизованное хранение данных и предоставляющий приложениям услуги по обработке данных.

Совокупность данных, хранимых под управлением СУБД, называется базой данных. В оригинальном английском варианте словосочетание data base означает «основание, состоящее из данных». Этот смысл несколько искажается в русском словосочетании «база данных». На самом деле это – фундамент, на котором строятся приложения и который состоит из данных. Действительно, данные (а, следовательно, база данных) являются очень существенной частью практически любой информационной системы. [1]

Целью разработки является создание базы данных комиссионных магазинов города с помощью СУБД PostgreSQL и создание клиентского приложения для взаимодействия с этой базой данных на языке программирования C++.

Разработанная система может применяться для учета деятельности комиссионных магазинов города.

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ, ПОСТАНОВКА ЗАДАЧИ

Система учета работы комиссионного магазина должна содержать данные о комитентах (ФИО, район проживания, социальное положение (рабочий, служащий, пенсионер, домохозяйка, предприниматель...), место работы, дата рождения, телефон), принимаемых товарах (номер квитанции, группа товара (посуда, одежда, обувь, игрушки,...), наименование товара, дата приема, количество единиц, цена за единицу) и о реализации товаров (товар, дата, количество проданных единиц).

Задача: разработать программный комплекс для автоматизации управления предметной областью. Разработать роли: работник, администратор, владелец сети магазинов.

Работник может просматривать товары в своем магазине, добавлять новые товары, реализовывать товары, добавлять новых клиентов.

Администратор – это управляющая роль. Он имеет такие возможности как добавление и удаление ролей (работников и владельцев), перевод работников из одного магазина в другой, увольнение работников, редактирование справочников.

Владелец может просматривать данные по всем товарам, реализациям работникам и клиентам. Также может просматривать заранее скомпонованные выборки данных (запросы), а также имеет доступ к просмотру статистики в виде диаграммы и экспорт статистики в формат MicrosoftExcel.

2 СИСТЕМА УПРАВЛЕНИЯ БАЗОЙ ДАННЫХ

В качестве СУБД был выбран PostgreSQL. PostgreSQL идеально подходит для решения задачи, т.к. имеет функциональность:

создание многопользовательской системы, в которой есть возможность разграничения прав различных пользователей;

поддерживает БД неограниченного размера;

возможность описания всей логики на сервере позволяет перенести все вычисления и обработки данных на сервер, что существенно упрощает поддержку системы и внесение в нее изменений;

существует поддержка большого количество встроенных типов, а также возможность создавать свои типы с поддержкой встраивания ограничения в тип (домены);

существует возможность оптимизировать запросы при помощи индексов в следствии анализа предметной области и выявления данных, к которым имеется большое количество запросов и по каким данным происходит сортировка и фильтрация, при помощи индексов;

партиционирование позволяет задумываться о возможности масштабирования системы без потери эффективности.

Открытый исходный код позволяет не беспокоиться о неожиданных исходах операций и проблем с безопасностью, поскольку все процессы, проходящие внутри системы, полностью открыты.

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

Для разработки клиентской части приложения был выбран язык C++. Он обладает множеством преимуществ, которые выгодно выделяют его на фоне остальных языков программирования.

Во-первых, он очень быстрый. Программа, которая грамотно написана на C++ в подавляющем большинстве случаев окажется быстрее программы с тем же функционалом, написанной на другом языке программирования.

Во-вторых, он обладает завидной универсальностью. Поскольку компилятор C++ есть на каждой операционной системе, появляется возможность написать одну программу, которая сможет скомпилироваться на любой платформе.

В-третьих, до сих пор идет активная поддержка и модификация C++. Разрабатывают и выпускают новые стандарты, что позволяет языку не отставать от своих конкурентов, и даже опережать их как в функциональности, так и в удобстве написания кода.

В-четвертых, он до сих пор остается одним из самых востребованных языков программирования, потому что имеет широкие возможности, позволяющие оптимизировать программируемые системы до такого состояния, до которого невозможно добраться, используя более высокоуровневые языки программирования.

Также C++ обладает невероятно мощной библиотекой Qt. Эта библиотека позволяет создавать удобные и эффективные пользовательские интерфейсы, а также позволяет удобно связывать приложение с сервером базы данных.

3.1 Не визуальные компоненты для работы с данными

Для взаимодействия с базой данных библиотека Qt предоставляет класс `QSqlQuery`, который позволяет формировать запросы, принимать ответы сервера, экранировать переменные (рис. 3.1). Для начала необходимо создать запрос, и ввести команду на языке SQL, затем при необходимости задаются значения. После этого запрос выполняется.

```
QSqlQuery query;
query.prepare("select * from products where id = :quittance");
query.bindValue(":quittance", quittance);
query.exec();
```

Рисунок 3.1 – Шаблон работы с запросом

Для взаимодействия с графическим интерфейсом используется класс `QSqlQueryModel`. Он позволяет сохранять результаты запроса, чтобы потом отобразить их пользователю (рис. 3.2).

```
QSqlQueryModel* query_model = new QSqlQueryModel(this);
query_model->setQuery(query);
```

Рисунок 3.2 – Создание новой модели и сохранение запроса, результаты которого она будет отображать

При разработке программы было принято решение реализовать вспомогательные функции для ускорения и облегчения разработки. Одной из таких функций является `sendQuery`. Она позволяет в удобном виде отправлять запрос и дает на выходе курсор, который позволяет работать с результатом запроса (рис. 3.3).

```

 QSqlQuery sendQuery(const QString& query) {
    QSqlQuery sql_api;
    sql_api.prepare(query);
    if (!sql_api.exec()) {
        QMessageBox::warning(nullptr, "Ошибка", sql_api.lastError().text());
    }
    return sql_api;
}

```

Рисунок 3.3 – Функция sendQuery

3.2 Визуальные компоненты отображения данных

Для визуализации результатов запросов используется класс `QTableView`. Он имеет возможность принимать модель и отображать ее содержимое. Эта особенность позволяет нам выводить данные, которые мы получили ранее, при выполнении запроса.

```

 ui->products_table2->setModel(query_model);

```

Рисунок 3.4 – Задать модель, которую будет отображать таблица

Помимо отображения данных в классе `QTableView` имеется возможность изменения отображения, что позволяет более гибко настраивать выводимую пользователю информацию.

Например, если мы не хотим показывать пользователю некоторые столбцы таблицы, однако данные в них нам еще нужны для дальнейшей работы, то мы можем просто скрыть их от пользователя, при этом использовать их в программе дальше (рис. 3.5).

```

 ui->products_table2->hideColumn(0);
 ui->products_table2->hideColumn(6);
 ui->products_table2->hideColumn(7);

```

Рисунок 3.5 – Скрытие столбцов из таблицы

Для отображения неизменяемого текста используется класс `QLabel`.
Для ввода данных пользователем используется класс `QLineEdit`.

Для отображения сообщений об ошибках используется класс `QMessageBox` (рис. 3.6)

```
QMessageBox::warning(nullptr, "Ошибка", "Неверный логин или пароль");
```

Рисунок 3.6 – Использование всплывающего сообщения

4 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ В ВЫБРАННОЙ СУБД

4.1 Проектирование концептуальной модели БД

Концептуальное проектирование базы данных – процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты [2].

Как правило, основной целью концептуального проектирования базы данных является описание способа физической реализации логического проекта базы данных [3].

Приступая к физическому проектированию, прежде всего, необходимо проанализировать и хорошо усвоить информацию об отношениях, собранную на этапе анализа предметной области [5].

В модели базы данных присутствуют 4 справочника:

- «shops» - справочник, содержащий идентификационный номер (id) и имя магазина (name);
- «product_types» - справочник, содержащий идентификационный номер (id) и тип продукта (посуда, одежда и т.д.) (product_type);
- «statuses» справочник, содержащий идентификационный номер (id) и имя социальный статус клиента (студент, пенсионер и т.д.) (status);
- «districts» справочник, содержащий идентификационный номер (id) и название района города (district).

В модели присутствуют 4 таблицы:

- «customers» - таблица, содержащая идентификационный номер клиента (id), имя (first_name), фамилию клиента (second_name), идентификатор района, в котором проживает клиент (id_district), идентификатор социального статуса клиента (id_status), место работы

клиента (work), дату рождения клиента (birthday), номер телефона клиента (phone);

– «workers» - таблица, содержащая идентификационный номер клиента (id), логин работника (login), идентификатор магазина, в котором работает работник (id_shop), его состояние (активен или уволен) (is_available);

– «products» - таблица, содержащая идентификатор продукта (id), номер квитанции, которая была выписана при получении товара (quittance), идентификатор типа продукта (id_product_type), название продукта (product_name), дату приема продукта на склад (reception), количество продукта при приеме (count), оставшееся количество продукта на складе (reception), цена за единицу продукта (price), идентификатор клиента, сдавшего этот продукт (id_customer), идентификатор магазина, в котором хранится продукт (id_shop), идентификатор работника, принявшего продукт на склад (id_worker);

– «realization» - идентификатор продажи товара (id), номер чека, который был выписан при продаже товара (ticket), идентификационный номер продукта, который был куплен (id_product), дата продажи (realization_date), количество проданного товара (realization_count), идентификатор работника, который продал товар (id_worker).

После проведенного анализа и предварительного проектирования было решено остановиться на варианте концептуальной модели базы данных представленной на рисунке 4.1.

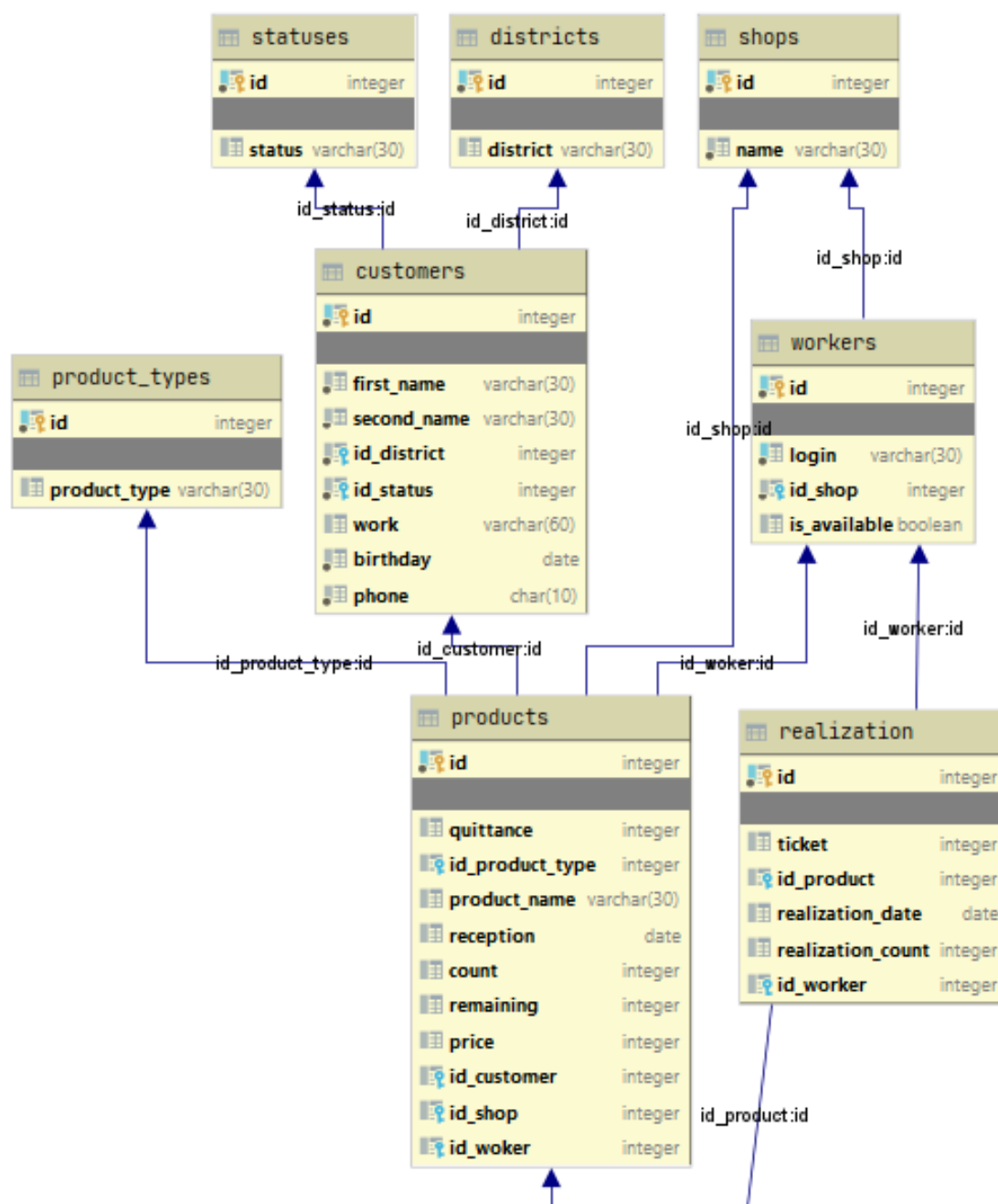


Рисунок 4.1 – Концептуальная модель базы данных

4.2 Создание таблиц, доменов, индексов

Таблицы создаются по уже готовой концептуальной модели базы данных. При создании необходимо указывать название полей, их типы, ограничения, а также можно указать внешние ключи, которые связывают разные таблицы.

При создании таблиц были использованы следующие SQL запросы.
(см. рис. 4.2 – 4.9)

```
create table customers
(
    id          serial      not null
        constraint customers_pkey
            primary key,
    first_name  varchar(30) not null,
    second_name varchar(30) not null,
    id_district integer      not null
        constraint customers_id_district_fkey
            references districts
            on update cascade on delete cascade,
    id_status   integer      not null
        constraint customers_id_status_fkey
            references statuses
            on update cascade on delete cascade,
    work        varchar(60),
    birthday    date         not null
        constraint years_18
            check (age((CURRENT_DATE)::timestamp with time zone, (birthday)::timestamp with time zone) >=
                '18 years'::interval),
    phone       char(10)     not null
);
```

Рисунок 4.2 – Создание таблицы пользователей

```
create table districts
(
    id          serial not null
        constraint districts_pkey
            primary key,
    district    varchar(30)
);
```

Рисунок 4.3 – Создание справочника районы

```
create table product_types
(
    id          serial not null
        constraint product_types_pkey
            primary key,
    product_type varchar(30)
);
```

Рисунок 4.4 – Создание справочника типы продуктов

```

create table products
(
    id serial not null
        constraint products_pkey
            primary key,
    quittance integer,
    id_product_type integer
        constraint products_id_product_type_fkey
            references product_types
            on update cascade on delete cascade,
    product_name varchar(30),
    reception date,
    count integer,
    remaining integer,
    price integer,
    id_customer integer
        constraint products_id_customer_fkey
            references customers
            on update cascade on delete cascade,
    id_shop integer
        constraint products_id_shop_fkey
            references shops
            on update cascade on delete cascade,
    id_woker integer
        constraint products_id_woker_fkey
            references workers
            on update cascade on delete cascade,
    constraint remaining_less_or_equal_count
        check (remaining <= count)
);

```

Рисунок 4.5 – Создание таблицы продуктов


```

create table realization
(
    id serial not null
        constraint realization_pkey
        primary key,
    ticket integer,
    id_product integer
        constraint realization_id_product_fkey
        references products
        on update cascade on delete cascade,
    realization_date date,
    realization_count integer,
    id_worker integer
        constraint realization_id_worker_fkey
        references workers
        on update cascade on delete cascade
);

```

Рисунок 4.6 – Создание таблицы реализаций

```

create table shops
(
    id serial not null
        constraint shops_pkey
        primary key,
    name varchar(30) not null
);

```

Рисунок 4.7 – Создание справочника магазинов

```

create table statuses
(
    id serial not null
        constraint statuses_pkey
        primary key,
    status varchar(30)
);

```

Рисунок 4.8 – Создание справочника социальных статусов

```

create table workers
(
    id          serial      not null
        constraint workers_pkey
            primary key,
    login       varchar(30) not null,
    id_shop     integer     not null
        constraint workers_shops_id_fk
            references shops
            on update cascade on delete cascade,
    is_available boolean default true
);

```

Рисунок 4.9 – Создание таблицы работников

```

create domain phone_number as char(10)
constraint phone_number_check check (value ~ similar_to_escape('071[0-9]{7}'));

```

Рисунок 4.10 – Создание домена для ввода номера телефона

```

create index realization_ticket_index
on realization (id_product);

```

Рисунок 4.11 – Создание индекса в таблице реализаций

```

create index id_product_types_idx on products using btree (id_product_types);

```

Рисунок 4.12 – Создание индекса в таблице продуктов

4.3 Разработка триггеров

Триггеры необходимы в проекте для добавления большей логики при стандартных действиях (удаления, изменения, вставки), а также для того чтобы указать базе данных как обновлять таблицы при обновлении представлений [6]. Также триггеры позволяют реализовать ограничения, которые используют подзапросы.

В программе при помощи триггеров выполнялись различные действия с данными. Первое их применение – это проверка вводимого значения в таблицу (рис. 4.13 – 4.14)

```

create trigger product_count_check
    before update
    on products
    for each row
execute function count_check();

create or replace function count_check() returns trigger as
$$
begin
    if new.count < 0 then
        raise exception 'Нельзя столько продавать!';
    else
        return new;
    end if;
end;
$$ language plpgsql;

```

Рисунок 4.13 – Триггер проверяющие количество продаваемого товара

```

create trigger workers_is_available_before_update
    before update
    on workers
    for each row
execute function is_available_check();

create or replace function is_available_check() returns trigger as
$$
begin
    if old.is_available = false then
        return old;
    else
        return new;
    end if;
end;
$$ language plpgsql;

```

Рисунок 4.14 – Триггер, проверяющий, чтобы не изменялись данные уже уволенных сотрудников

Также триггеры применялись для обеспечения целостности данных при проведении операций, которые осуществлялись над связанными по смыслу таблицами (рис 4.15).

```
create trigger product_count_decrease_on_realization_before_insert
  before insert
  on realization
  for each row
  execute function decrease_products();

create or replace function decrease_products() returns trigger as
$$
begin
  update products
  set remaining = (select remaining from products where id = new.id_product) - new.realization_count
  where id = new.id_product;
  return new;
end;
$$ language plpgsql;
```

Рисунок 4.15 – Триггер, который уменьшает количество товара при его продаже

Еще одним применением триггеров стала реализация модифицируемого представления. Исходя из анализа предметной области было решено позволить работникам только вставлять данные в таблицу. Возможность изменения и удаления данных может стать инструментом в руках злоумышленников. Важным условием при создании модифицируемого представления (рис. 3.16) является указание в триггерах параметра `instead of`, который и позволяет реализовывать такие удобные инструменты. Создание триггера показано на рисунке 3.17.

```

create or replace view products_modify_view with (security_barrier) as
select p.quittance,
       pt.product_type,
       p.product_name,
       p.reception,
       p.remaining,
       p.price,
       p.id_customer,
       c.first_name,
       c.second_name,
       c.phone,
       s.name as shop
from products p
       inner join product_types pt on p.id_product_type = pt.id
       inner join customers c on p.id_customer = c.id
       inner join shops s on p.id_shop = s.id
order by reception desc, p.quittance desc;

```

Рисунок 4.16– Создание модифицируемого представления

```

create trigger products_modify_view_instead_of_insert
  instead of insert
  on products_modify_view
  for each row
execute procedure instead_of_insert();

create or replace function instead_of_insert() returns trigger
as
$$
begin
  insert into products (quittance, id_product_type, product_name, reception, count, remaining, price, id_customer,
                       id_shop, id_woker)
  values (nextval('quittance_num'), (select id from product_types where product_type = new.product_type),
         new.product_name, now()::date, new.remaining, new.remaining, new.price,
         new.id_customer, (select workers.id_shop from workers where login = session_user),
         (select id from workers where login = session_user));
  return new;
end;
$$ language plpgsql;

```

Рисунок 4.17 – Триггер, обеспечивающий взаимодействие с
модифицируемым представлением

4.4 Разработка ограничений доступа

При разработке базы данных необходимо разграничить права доступа для таких ролей, как администратор, рабочий, владелец.

Для разграничения прав использовался следующий подход: предоставить grant на таблицу для роли к которой будет осуществляться доступ, далее определить политику, то есть создать row level security (RLS), если данный подход не практичен, например из-за того, что в условии RLS необходим доступ к таблицам к которым нет доступа, то создается представления view с условием where и далее дается право grant select для данной таблицы, если и этот метод не подходит, то создается функция с параметром security definer и дается grant execute на данную функцию роли.

Для того, чтобы включить защиту строк у таблцы необходимо прописать команду: alter table «имя таблицы» enable row level security.

4.4.1 Права доступа работника (роль worker)

Работник – это роль, которой необходим доступ к просмотру товаров, реализаций, заказчиков, для того, чтобы отвечать на вопросы покупателей. Также ему нужны права на последовательности и вставку в таблицы из-за того, что он должен добавлять новые товары, новых клиентов, а также реализовывать товар (рис. 4.18).

```
create role worker;

grant usage on sequence products_id_seq, quittance_num, customers_id_seq,
    realization_id_seq, ticket_num to worker;
grant select, insert on table products, realization, customers to worker;
grant select on table workers, shops, product_types, customers_view, districts, statuses to worker;
grant select, insert on table products_modify_view to worker;
```

Рисунок 4.18 – Права для работника

Политики рабочего обеспечивают ему доступ к продуктам и реализациям только в своем магазине. Также в талице работников он может видеть только себя. В таблице магазинов работник видит только свой магазин (рис. 4.19).

```

-- Видит только себя в таблице работников
create policy workers_id_policy on workers for select to worker using (current_user = login);

-- Просмотр товаров только в своем магазине
create policy workers_shop_policy on products for all to worker using
(
    id_shop = (select workers.id_shop
               from workers
               where login = current_user));

drop policy workers_shop_policy on products;

-- Просмотр реализации только своего магазина
create policy realization_shop_policy on realization for all to worker using
(
    exists(select id
             from products
             where (products.id = id_product)
                  and id_shop = (select id_shop from workers)));

drop policy realization_shop_policy on realization;

-- Просмотр информации о своем магазине
create policy shop_id_policy on shops for select to worker using (id = (select id_shop
                                                                    from workers
                                                                    where login = current_user));

```

Рисунок 4.19 – Политики для работника

4.4.2 Права доступа администратора (роль administrator)

Администратор имеет доступ к добавлению и удалению в справочники, а также полный доступ к таблице работников потому, что он через него проходят все изменения среди персонала (рис.4.20). Администратор имеет доступ ко всем записям в таблицах работников и магазинов (рис. 4.21).

```

create role administrator;
alter role administrator createrole;

grant usage on sequence districts_id_seq, product_types_id_seq, statuses_id_seq, shops_id_seq to administrator;
grant update, insert, select, delete on table workers, districts, product_types, statuses to administrator;
grant select on table shops to administrator;

```

Рисунок 4.20 – Права для администратора

```
create policy administrator_realization_policy on workers for all to administrator using (true);  
create policy shop_to_admin_policy on shops for select to administrator using (true);
```

Рисунок 4.21 – Политики для администратора

4.4.3 Права доступа для владельца (роль holder)

Владелец имеет право смотреть на все таблицы, а также имеет доступ к запросам и статистике (рис. 4.22).

```
create role holder;  
  
grant select on all tables in schema public to holder;
```

Рисунок 4.22 – Права для владельца

4.5 Проектирование запросов к базе данных

Запросы с симметричным внутренним соединением предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON.

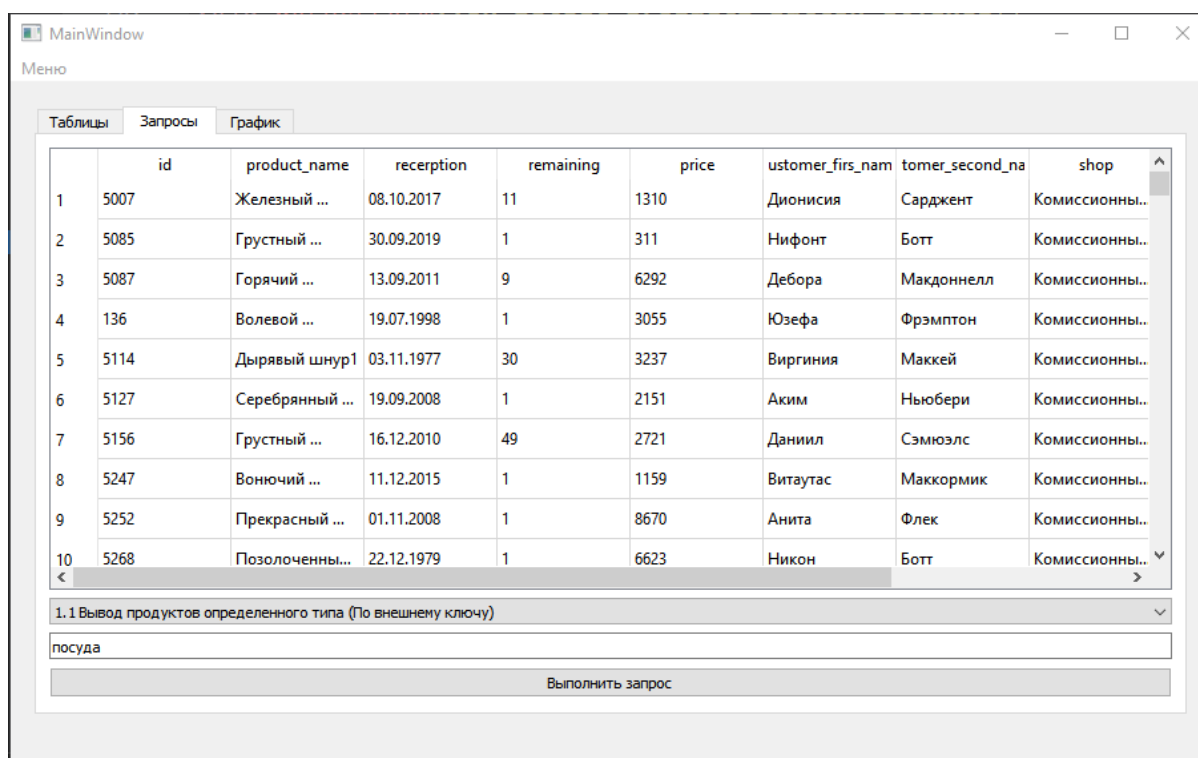
Запрос, показывающий информацию об идентификаторе продукта, названии продукта, дате према продукта на склад, количестве оставшихся единиц товара, цене за единицу товара, имя, фамилия заказчика, название магазина, типе продукта, у продуктов, которые относятся к указанному типу продуктов. Тип продукта задает пользователь. Код предоставлен на рисунке 4.23, а результат – 4.24.


```

select products.id,
       products.product_name,
       products.reception,
       products.remaining,
       products.price,
       c.first_name,
       c.second_name,
       s.name,
       pt.product_type
from products
       inner join customers c on products.id_customer = c.id
       inner join product_types pt on products.id_product_type = pt.id
       inner join shops s on products.id_shop = s.id
where id_product_type =
       (select product_types.id from product_types
        where product_types.product_type = product_t);

```

Рисунок 4.23 – Симметричное внутреннее соединение с условием по внешнему ключу



	id	product_name	reception	remaining	price	ustomer_firs_nam	tomer_second_na	shop
1	5007	Железный ...	08.10.2017	11	1310	Дионисия	Сарджент	Комиссионны..
2	5085	Грустный ...	30.09.2019	1	311	Нифонт	Ботт	Комиссионны..
3	5087	Горячий ...	13.09.2011	9	6292	Дебора	Макдоннелл	Комиссионны..
4	136	Волевой ...	19.07.1998	1	3055	Юзефа	Фрэмpton	Комиссионны..
5	5114	Дырявый шнур1	03.11.1977	30	3237	Виргиния	Маккей	Комиссионны..
6	5127	Серебряный ...	19.09.2008	1	2151	Аким	Ньюбери	Комиссионны..
7	5156	Грустный ...	16.12.2010	49	2721	Даниил	Сэмюэлс	Комиссионны..
8	5247	Вонючий ...	11.12.2015	1	1159	Витаутас	Маккормик	Комиссионны..
9	5252	Прекрасный ...	01.11.2008	1	8670	Анита	Флек	Комиссионны..
10	5268	Позолоченны...	22.12.1979	1	6623	Никон	Ботт	Комиссионны..

1.1 Вывод продуктов определенного типа (По внешнему ключу)

посуда

Выполнить запрос

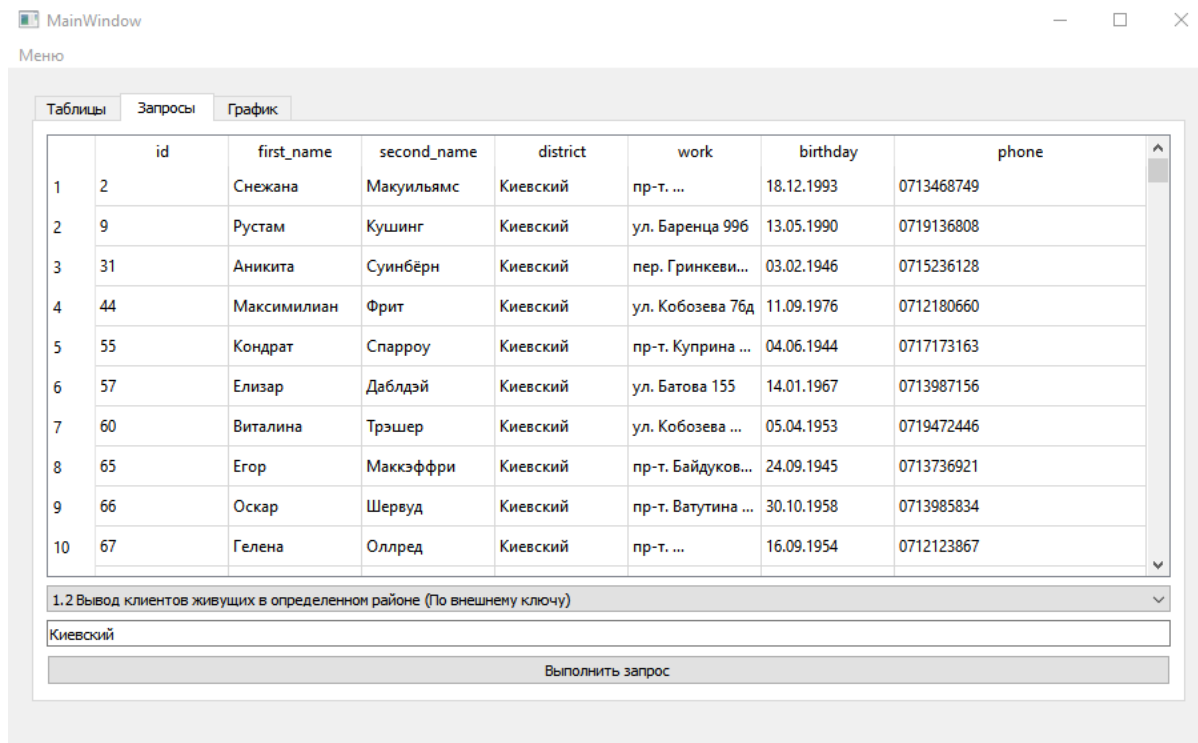
Рисунок 4.24– Продукты типа посуда

Запрос, показывающий информацию об идентификаторе пользователя, имени, фамилии заказчика, названии района его

проживания, месте его работы, дате его рождения, номере его телефона, проживающего в указанном районе. Район задает пользователь. Код предоставлен на рисунке 4.25, а результат – 4.26.

```
select customers.id,
       customers.first_name,
       customers.second_name,
       d.district,
       customers.work,
       customers.birthday,
       customers.phone
from customers
      inner join districts d on customers.id_district = d.id
where id_district = (select districts.id from districts
                    where districts.district = district_name);
```

Рисунок 4.25– Симметричное внутреннее соединение с условием по
внешнему ключу



	id	first_name	second_name	district	work	birthday	phone
1	2	Снежана	Макуильямс	Киевский	пр-т. ...	18.12.1993	0713468749
2	9	Рустам	Кушинг	Киевский	ул. Баренца 996	13.05.1990	0719136808
3	31	Аникита	Суинбёрн	Киевский	пер. Гринкеви...	03.02.1946	0715236128
4	44	Максимилиан	Фрит	Киевский	ул. Кобозева 76д	11.09.1976	0712180660
5	55	Кондрат	Спарроу	Киевский	пр-т. Куприна ...	04.06.1944	0717173163
6	57	Елизар	Даблдэй	Киевский	ул. Батова 155	14.01.1967	0713987156
7	60	Виталина	Трэшер	Киевский	ул. Кобозева ...	05.04.1953	0719472446
8	65	Егор	Маккэффри	Киевский	пр-т. Байдуков...	24.09.1945	0713736921
9	66	Оскар	Шереуд	Киевский	пр-т. Ватутина ...	30.10.1958	0713985834
10	67	Гелена	Оллред	Киевский	пр-т. ...	16.09.1954	0712123867

1.2 Вывод клиентов живущих в определенном районе (По внешнему ключу)

Киевский

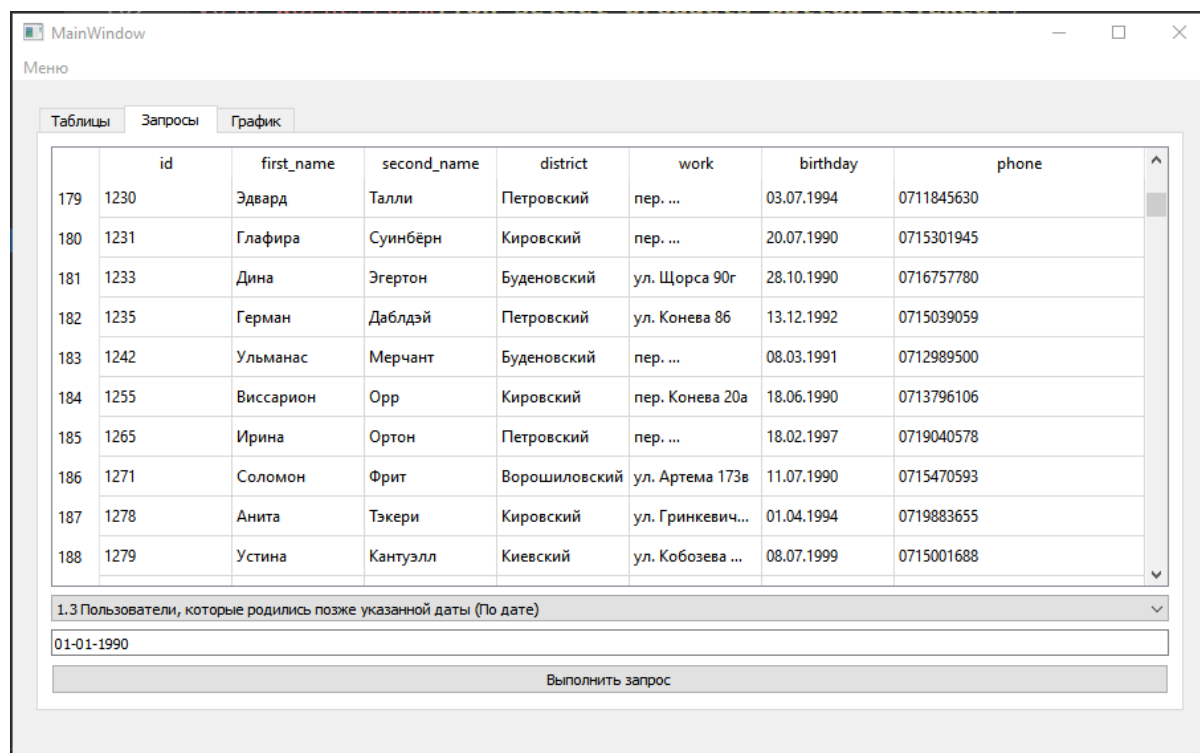
Выполнить запрос

Рисунок 4.26 – Клиенты, живущие в киевском районе

Запрос, показывающий информацию об идентификаторе пользователя, имени, фамилии заказчика, названии района его проживания, месте его работы, дате его рождения, номере его телефона, у пользователей, которые родились позже указанной даты. Дату задаёт пользователь. Код предоставлен на рисунке 4.27, а результат – 4.28.

```
select customers.id,
       customers.first_name,
       customers.second_name,
       d.district,
       customers.work,
       customers.birthday,
       customers.phone
from customers
      inner join districts d on customers.id_district = d.id
where customers.birthday > birth;
```

Рисунок 4.27– Симметричное внутреннее соединение с условием на дату



	id	first_name	second_name	district	work	birthday	phone
179	1230	Эдвард	Талли	Петровский	пер. ...	03.07.1994	0711845630
180	1231	Глафира	Суинбёрн	Кировский	пер. ...	20.07.1990	0715301945
181	1233	Дина	Эгертон	Буденовский	ул. Щорса 90г	28.10.1990	0716757780
182	1235	Герман	Даблдэй	Петровский	ул. Конева 8б	13.12.1992	0715039059
183	1242	Ульманас	Мерчант	Буденовский	пер. ...	08.03.1991	0712989500
184	1255	Виссарион	Орр	Кировский	пер. Конева 20а	18.06.1990	0713796106
185	1265	Ирина	Ортон	Петровский	пер. ...	18.02.1997	0719040578
186	1271	Соломон	Фрит	Ворошиловский	ул. Артема 173в	11.07.1990	0715470593
187	1278	Анита	Тэкери	Кировский	ул. Гринкевич...	01.04.1994	0719883655
188	1279	Устина	Кантуэлл	Киевский	ул. Кобозева ...	08.07.1999	0715001688

1.3 Пользователи, которые родились позже указанной даты (По дате)

01-01-1990

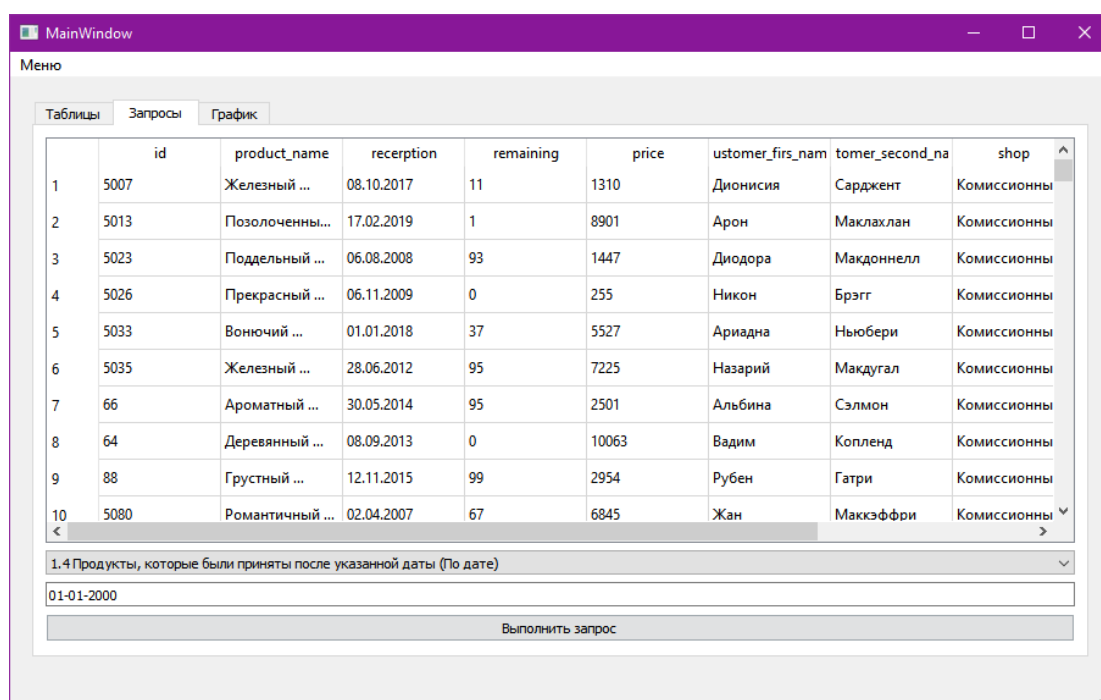
Выполнить запрос

Рисунок 4.28 – Клиенты, родившиеся в 1990 году и позже

Запрос, показывающий информацию об идентификаторе продукта, названии продукта, дате приема продукта на склад, количестве оставшихся единиц товара, цене за единицу товара, имени, фамилии заказчика, названии магазина, типе продукта, у продуктов, которые были приняты на склад после указанной даты. Дату задает пользователь. Код предоставлен на рисунке 4.29, а результат – 4.30.

```
select products.id,
       products.product_name,
       products.reception,
       products.remaining,
       products.price,
       c.first_name,
       c.second_name,
       s.name,
       pt.product_type
from products
       inner join customers c on products.id_customer = c.id
       inner join product_types pt on products.id_product_type = pt.id
       inner join shops s on products.id_shop = s.id
where reception > reception_date;
```

Рисунок 4.29– Симметричное внутреннее соединение с условием на дату



	id	product_name	reception	remaining	price	ustomer_firs_nam	tomer_second_na	shop
1	5007	Железный ...	08.10.2017	11	1310	Дионисия	Сарджент	Комиссионны
2	5013	Позолоченны...	17.02.2019	1	8901	Арон	Маклахлан	Комиссионны
3	5023	Поддельный ...	06.08.2008	93	1447	Диодора	Макдоннелл	Комиссионны
4	5026	Прекрасный ...	06.11.2009	0	255	Никон	Брэгг	Комиссионны
5	5033	Вонючий ...	01.01.2018	37	5527	Ариадна	Ньюбери	Комиссионны
6	5035	Железный ...	28.06.2012	95	7225	Назарий	Макдугал	Комиссионны
7	66	Ароматный ...	30.05.2014	95	2501	Альбина	Сэлмон	Комиссионны
8	64	Деревянный ...	08.09.2013	0	10063	Вадим	Копленд	Комиссионны
9	88	Грустный ...	12.11.2015	99	2954	Рубен	Гатри	Комиссионны
10	5080	Романтичный ...	02.04.2007	67	6845	Жан	Маккэффри	Комиссионны

1.4 Продукты, которые были приняты после указанной даты (По дате)

01-01-2000

Выполнить запрос

Рисунок 4.30 –Продукты, которые были приняты в 2000 году и позже

Рассмотрим примеры запросов аналогичного типа, но без условий. Запрос, показывающий информацию об имени, фамилии заказчика, месте его работы, номере квитанции сданного им товара, количестве единиц товара оставшихся на складе, названии магазина, в котором находится товар. Код предоставлен на рисунке 4.31, а результат – 4.32.

```
select customers.first_name || ' ' || customers.second_name as customer_name,
       customers.work,
       p.quittance,
       p.remaining,
       s.name
from customers
      inner join products p on customers.id = p.id_customer
      inner join shops s on p.id_shop = s.id
order by customer_name;
```

Рисунок 4.31 – Симметричное внутреннее соединение без условия

	customer_name	work	quittance	remaining	name
1	Абрам Барлоу	пр-т. Зайцева ...	32966	1	Комиссионный №119286
2	Абрам Барлоу	пр-т. Зайцева ...	29089	0	Комиссионный №657298
3	Абрам Барлоу	пр-т. Зайцева ...	33106	0	Комиссионный №924275
4	Абрам Биддер	пер. Баренца ...	49943	6	Комиссионный №97168
5	Абрам Бойер	ул. Зайцева 39а	7499	19	Комиссионный №134943
6	Абрам Брюстер	пер. Горбатова...	48492	0	Комиссионный №665739
7	Абрам Грейвз	пер. Калинина ...	3256	0	Комиссионный №389819
8	Абрам Грейвз	пер. Калинина ...	2349	1	Комиссионный №789147
9	Абрам Грейвз	пер. Калинина ...	20160	0	Комиссионный №794828
10	Абрам Грейвз	пер. Калинина ...	37462	1	Комиссионный №661434

2.1 Информация о клиентах и состояниях сданных ими товаров

01-01-2000

Выполнить запрос

Рисунок 4.32 – Информация о клиентах и состоянии сданных ими товаров

Запрос, показывающий информацию о номере квитанции, с которой товар был принят, имени продукта, дате приема, количестве проданных экземпляров, дате каждой покупки, имени магазина, в котором находится. Код предоставлен на рисунке 4.33, а результат – 4.34.

```
select products.quittance,
       products.product_name,
       products.reception,
       r.realization_count,
       r.realization_date,
       s.name as "shop"
from products
      inner join realization r on products.id = r.id_product
      inner join shops s on products.id_shop = s.id
order by s.name, products.reception, r.realization_date, r.realization_count;
```

Рисунок 4.33 – Симметричное внутреннее соединение без условия

	quittance	product_name	reception	realization_count	realization_date	shop
1	37754	Холодный ...	21.09.1942	1	15.07.1951	Комиссионный №101975
2	32415	Позолоченны...	01.02.1947	1	01.04.1959	Комиссионный №101975
3	32415	Позолоченны...	01.02.1947	1	29.03.1989	Комиссионный №101975
4	32415	Позолоченны...	01.02.1947	1	19.10.2010	Комиссионный №101975
5	6713	Поддельный ...	08.10.1951	5	12.06.2002	Комиссионный №101975
6	38533	Железный ...	25.07.1952	1	21.11.1961	Комиссионный №101975
7	38533	Железный ...	25.07.1952	1	03.05.1964	Комиссионный №101975
8	38533	Железный ...	25.07.1952	5	17.01.1975	Комиссионный №101975
9	45114	Горячий ...	06.06.1953	3	19.01.1972	Комиссионный №101975
10	30532	Веселый ...	15.03.1954	2	06.05.1992	Комиссионный №101975

2.2 Информация о продуктах и их реализации

01-01-2000

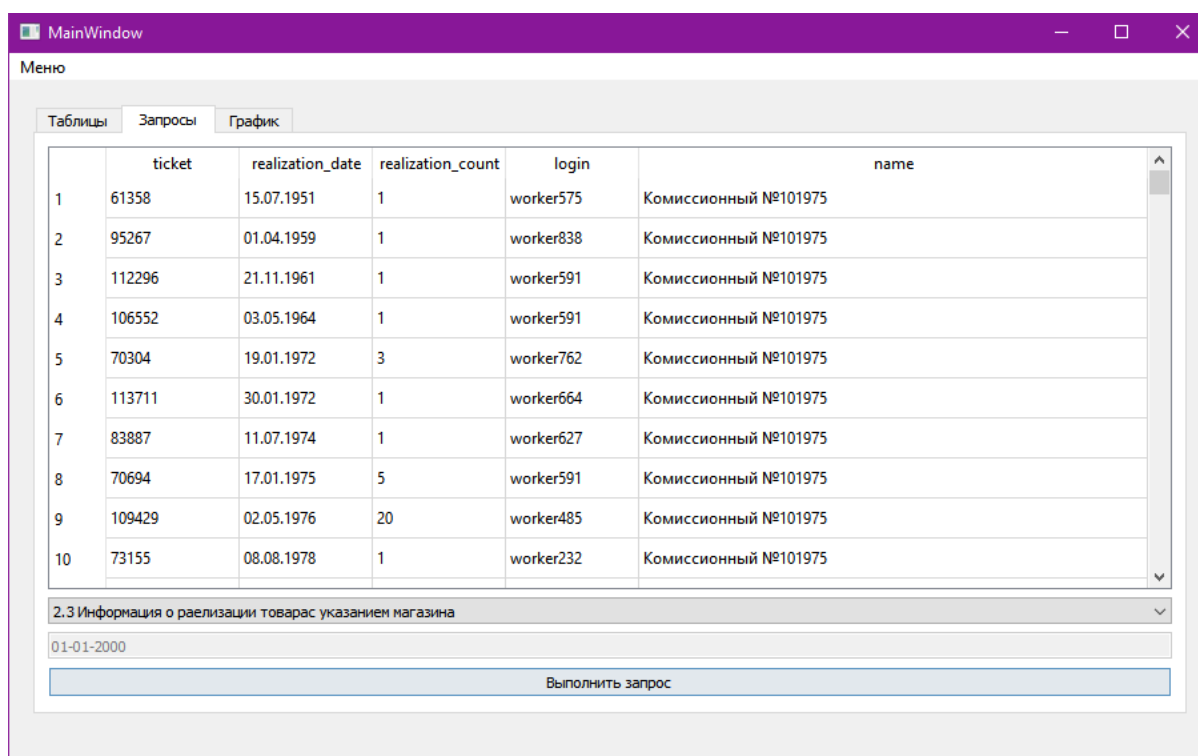
Выполнить запрос

Рисунок 4.34 – Информация о продуктах и их реализации

Запрос, показывающий информацию о номере чека, выданного при продаже, дате продажи, количества проданного товара, логине работника, который принял товар, названии магазина, в котором находится товар. Код предоставлен на рисунке 4.35, а результат – 4.36.

```
select realization.ticket, realization.realization_date,
       realization.realization_count, w.login, s.name
from realization
       inner join products p on realization.id_product = p.id
       inner join shops s on p.id_shop = s.id
       inner join workers w on p.id_woker = w.id
order by s.name, realization.realization_date, w.login;
```

Рисунок 4.35 – Симметричное внутреннее соединение без условия



	ticket	realization_date	realization_count	login	name
1	61358	15.07.1951	1	worker575	Комиссионный №101975
2	95267	01.04.1959	1	worker838	Комиссионный №101975
3	112296	21.11.1961	1	worker591	Комиссионный №101975
4	106552	03.05.1964	1	worker591	Комиссионный №101975
5	70304	19.01.1972	3	worker762	Комиссионный №101975
6	113711	30.01.1972	1	worker664	Комиссионный №101975
7	83887	11.07.1974	1	worker627	Комиссионный №101975
8	70694	17.01.1975	5	worker591	Комиссионный №101975
9	109429	02.05.1976	20	worker485	Комиссионный №101975
10	73155	08.08.1978	1	worker232	Комиссионный №101975

2.3 Информация о реализации товаров с указанием магазина

01-01-2000

Выполнить запрос

Рисунок 4.36 – Информация о реализации товаров с указанием магазина

Запрос с оператором LEFT OUTER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON, и

дополняются записями из первой по порядку (левой) таблицы, даже если они не соответствуют условию. У записей левой таблицы, которые не соответствуют условию, значение столбца из правой таблицы будет NULL (неопределённым).

Запрос, показывающий информацию о номере квитанции товара, с которой он был принят на склад, количестве товара, которое было при преме, оставшееся количество товара, номер чека, который был выписан при продаже товара. Код запроса представлен на рисунке 4.37, а результат – 4.38.

```
select products.quittance, products.count, products.remaining, r.ticket
from products
      left join realization r on products.id = r.id_product
order by quittance;
```

Рисунок 4.37 – Левое внешнее соединение

	quittance	count	remaining	ticket
24695	15479	1	0	72982
24696	15480	75	5	75388
24697	15481	1	0	87251
24698	15482	1	0	55444
24699	15483	4	4	
24700	15484	8	0	69856
24701	15485	1	0	72978
24702	15486	1	0	56811
24703	15487	5	3	109091
24704	15488	15	0	111254

3.1 Все продукты и их реализация

01-01-2000

Выполнить запрос

Рисунок 4.38 – Все продукты и их реализация (даже те, что еще не были реализованы)

Запрос с оператором RIGHT OUTER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON, и дополняются записями из второй по порядку (правой) таблицы, даже если они не соответствуют условию. У записей правой таблицы, которые не соответствуют условию, значение столбца из левой таблицы будет NULL (неопределённым).

Запрос, показывающий информацию о номере чека, который был продан товар, дате продажи, количестве проданного товара, логине работника, который продал товар. Код запроса представлен на рисунке 4.39, а результат – 4.40.

```
select r.ticket, r.realization_date, r.realization_count, workers.login
from workers
right join realization r on workers.id = r.id_worker;
```

Рисунок 4.39 – правое внешнее соединение

	ticket	realization_date	realization_count	login
24695	73714	16.04.2007	1	worker98
24696	73715	05.06.1996	1	worker617
24697	73716	02.05.2013	19	worker608
24698	73717	31.01.2017	1	worker115
24699	73718	13.10.2004	21	worker124
24700	73719	14.03.2009	3	worker736
24701	73720	08.07.2017	14	worker699
24702	73721	09.04.2016	1	worker999
24703	73722	21.02.2017	12	worker778
24704	73723	30.03.2019	2	worker300

4.1 Реализация продукта с указанием работника

01-01-2000

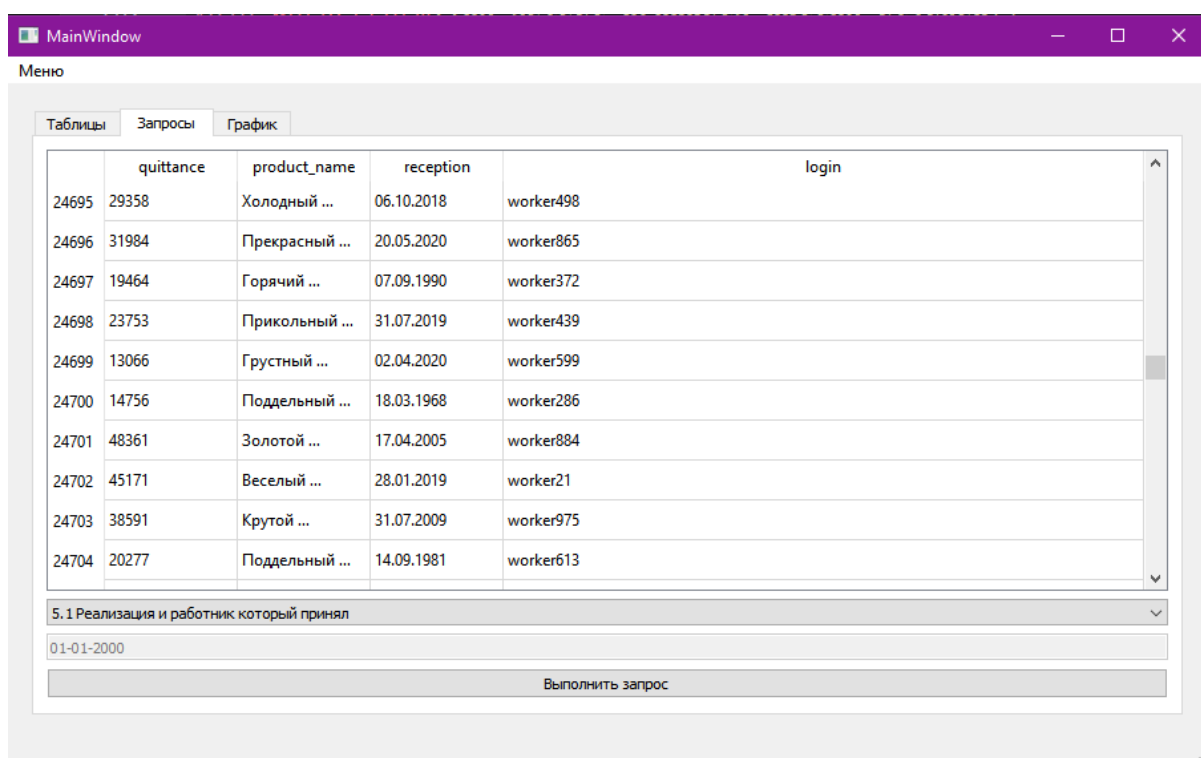
Выполнить запрос

Рисунок 4.40– Реализация продукта с указанием работника

Рассмотрим запрос на запросе по принципу левого. С помощью него мы находим информацию о квитанции, которая была выписана при приеме товара, имени продукта, дате приема продукта, логине работника, который принял товар на склад. Код предоставлен на рисунке 4.41, а результат – 4.42.

```
select products.quittance,
       products.product_name,
       products.reception,
       (select login from workers where workers.id = products.id_woker)
from products;
```

Рисунок 4.41 – Запрос на запросе по принципу левого соединения



	quittance	product_name	reception	login
24695	29358	Холодный ...	06.10.2018	worker498
24696	31984	Прекрасный ...	20.05.2020	worker865
24697	19464	Горячий ...	07.09.1990	worker372
24698	23753	Прикольный ...	31.07.2019	worker439
24699	13066	Грустный ...	02.04.2020	worker599
24700	14756	Поддельный ...	18.03.1968	worker286
24701	48361	Золотой ...	17.04.2005	worker884
24702	45171	Веселый ...	28.01.2019	worker21
24703	38591	Крутой ...	31.07.2009	worker975
24704	20277	Поддельный ...	14.09.1981	worker613

5.1 Реализация и работник который принял

01-01-2000

Выполнить запрос

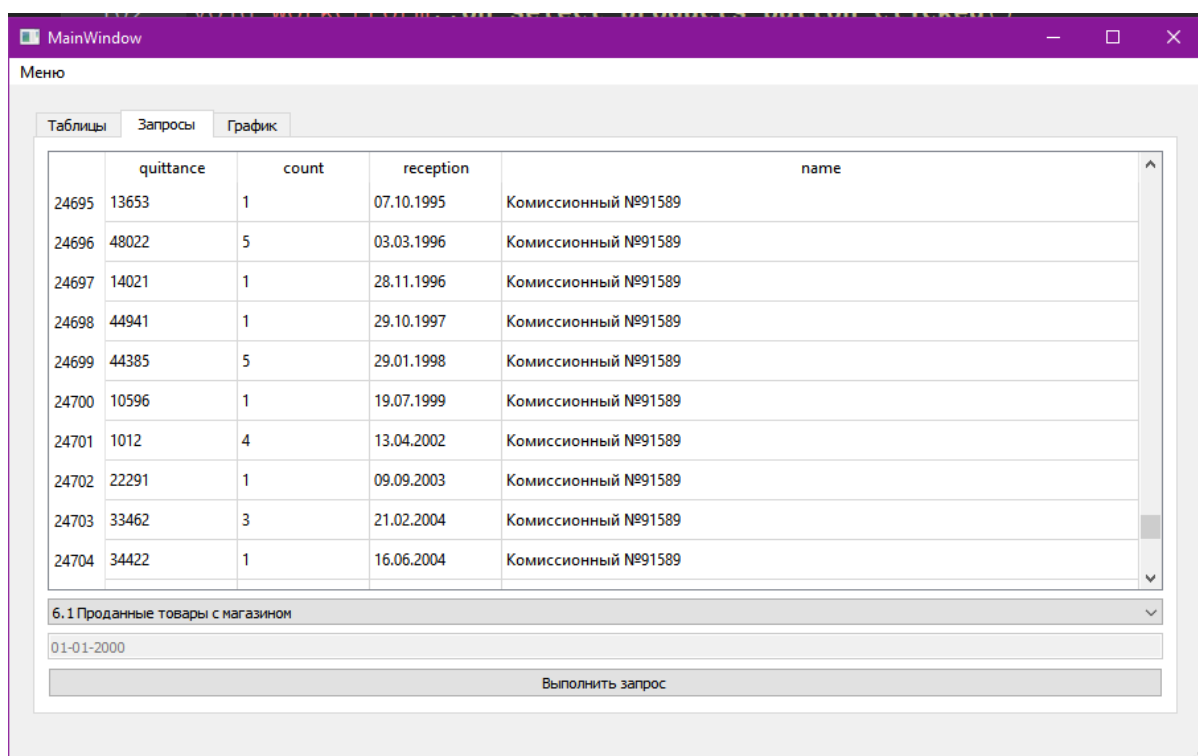
Рисунок 4.42 – Реализация и работник, который принял товар

С помощью итогового запроса без условия пользователь может получить информацию о квитанциях товаров, количестве чеков,

выписанных на этот товар, дате приема товара на склад, имени магазина, в котором товар был принят (см. рис. 4.43 – 4.44).

```
select sold_products.quittance, sold_products.count, sold_products.reception, s.name
from (select * from products where remaining = 0) as sold_products
     inner join shops s on sold_products.id_shop = s.id
order by s.name, sold_products.reception;
```

Рисунок 4.43 – Итоговый запрос без условия



	quittance	count	reception	name
24695	13653	1	07.10.1995	Комиссионный №91589
24696	48022	5	03.03.1996	Комиссионный №91589
24697	14021	1	28.11.1996	Комиссионный №91589
24698	44941	1	29.10.1997	Комиссионный №91589
24699	44385	5	29.01.1998	Комиссионный №91589
24700	10596	1	19.07.1999	Комиссионный №91589
24701	1012	4	13.04.2002	Комиссионный №91589
24702	22291	1	09.09.2003	Комиссионный №91589
24703	33462	3	21.02.2004	Комиссионный №91589
24704	34422	1	16.06.2004	Комиссионный №91589

6.1 Проданные товары с магазином

01-01-2000

Выполнить запрос

Рисунок 4.44 – Проданные товары с указанием магазина

Итоговый запрос без условия с итоговыми данными вида «всего», «в том числе» используются для того, чтобы получить информацию об общем количестве принятого товара и количество полностью проданного товара (больше не осталось на складе) (см. рис. 4.45 – 4.46).

```
select count(*), (select count(*) as sold from products where remaining = 0)
from products;
```

Рисунок 4.45 - Итоговый запрос без условия с итоговыми данными выда:
"всего", "в том числе"

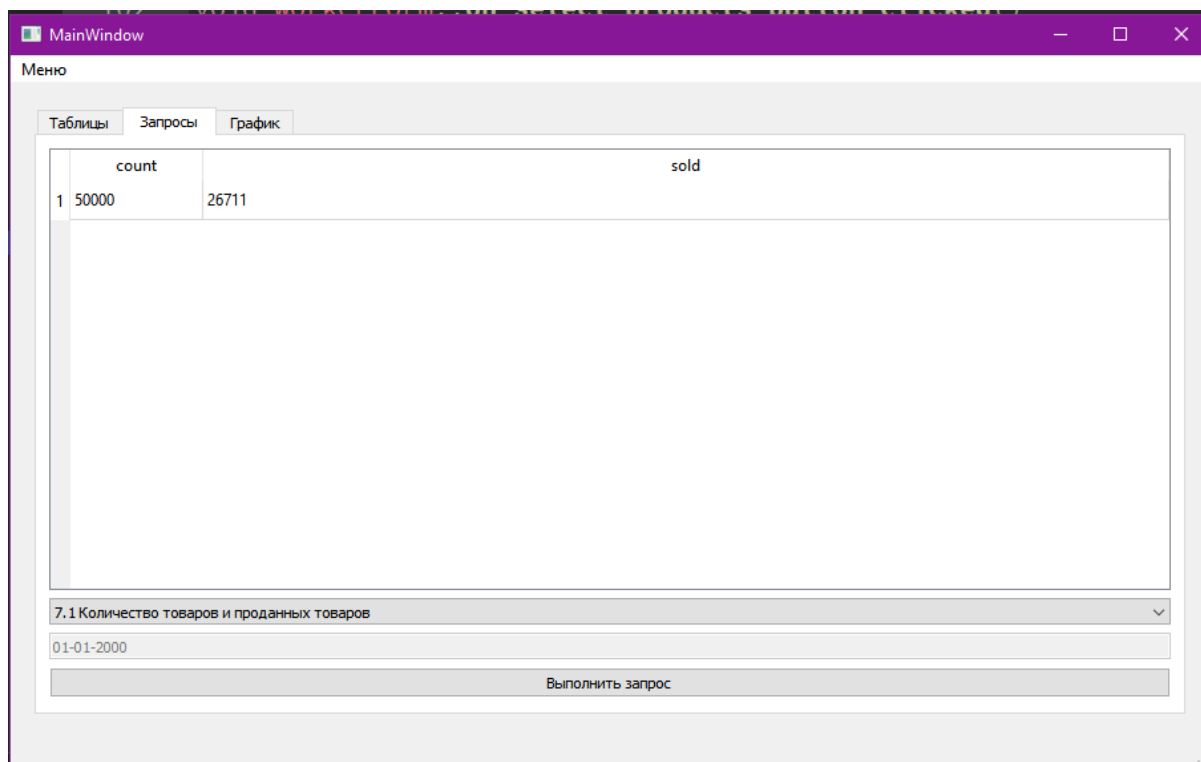


Рисунок 4.46 – Количество товаров и проданных товаров

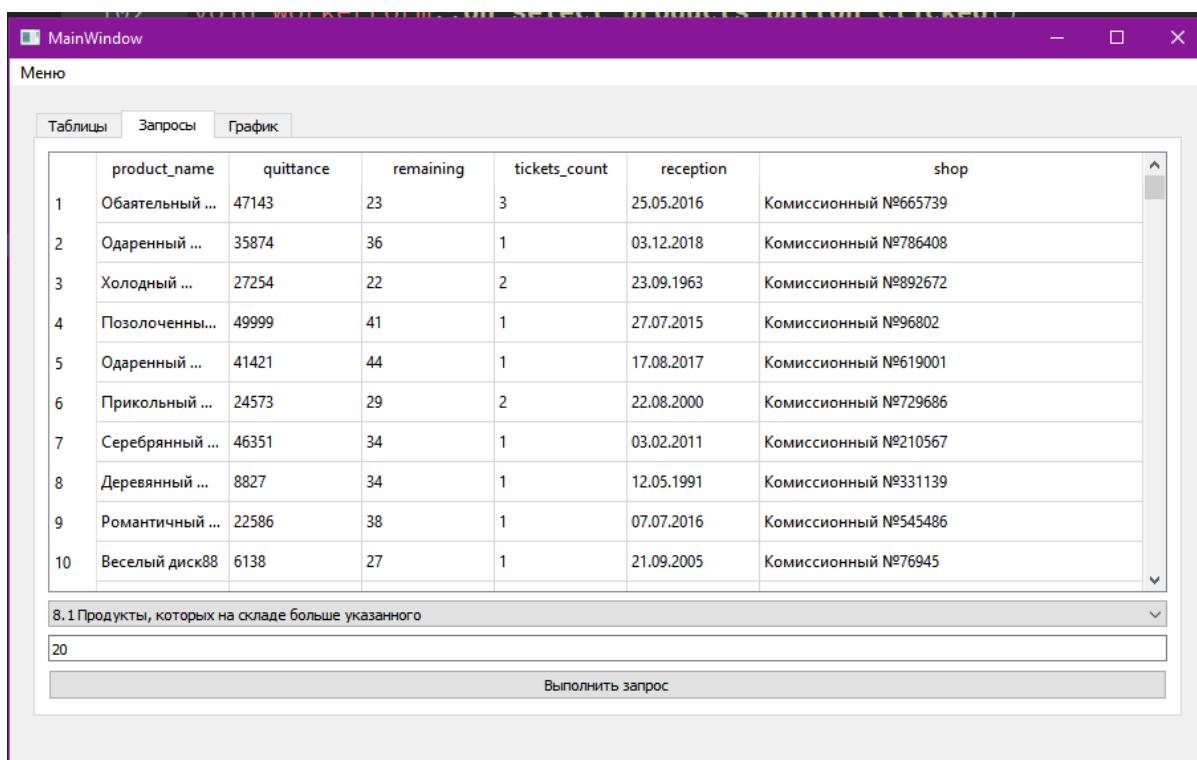
Итоговый запрос с условием на данные используем, чтобы получить информацию об имени товара, квитанции, с которой был принят товар, оставшееся количество товара, количестве чеков, оформленных на товар, дате приема, имени магазина, в котором находится товар у товаров, количество которых на складе больше указанного. Количество задаёт пользователь. Код представлен на рисунке 4.47, результат – 4.48.

```

select p.product_name,
       p.quittance,
       p.remaining,
       count(r.ticket):: integer as tickets_count,
       p.reception,
       s.name                as shop
from realization r
       inner join products p on r.id_product = p.id
       inner join shops s on p.id_shop = s.id
where p.remaining > remaining_count
group by r.id_product, p.product_name, p.quittance, p.remaining,
         p.reception, s.name

```

Рисунок 4.47 - Итоговые запросы с условием на данные по значению



MainWindow

Меню

Таблицы Запросы График

	product_name	quittance	remaining	tickets_count	reception	shop
1	Обаятельный ...	47143	23	3	25.05.2016	Комиссионный №665739
2	Одаренный ...	35874	36	1	03.12.2018	Комиссионный №786408
3	Холодный ...	27254	22	2	23.09.1963	Комиссионный №892672
4	Позолоченны...	49999	41	1	27.07.2015	Комиссионный №96802
5	Одаренный ...	41421	44	1	17.08.2017	Комиссионный №619001
6	Прикольный ...	24573	29	2	22.08.2000	Комиссионный №729686
7	Серебрянный ...	46351	34	1	03.02.2011	Комиссионный №210567
8	Деревянный ...	8827	34	1	12.05.1991	Комиссионный №331139
9	Романтичный ...	22586	38	1	07.07.2016	Комиссионный №545486
10	Веселый диск88	6138	27	1	21.09.2005	Комиссионный №76945

8.1 Продукты, которых на складе больше указанного

20

Выполнить запрос

Рисунок 4.48 – Товары, которых на складе больше 20

Итоговый запрос с условием на данные, но в качестве проверки используется маска. Такой запрос используется для получения информации об имени и фамилии клиента, его телефоне, количестве товаров, сданных им, у клиентов телефонный номер которых содержит

определенный набор цифр. Набор цифр задает пользователь. (см. рис. 4.49 – 4.50).

```
select t1.first_name, t1.second_name, t1.phone, t1.count
from (select c.first_name, c.second_name, c.phone,
count(p.quittance) :: integer as count
      from customers c
           inner join products p on c.id = p.id_customer
      where c.phone similar to '%' || part_of_phone || '%'
      group by c.phone, c.second_name, c.first_name
      order by count desc) as t1;
```

Рисунок 4.49- Итоговый запрос с условием на данные по маске

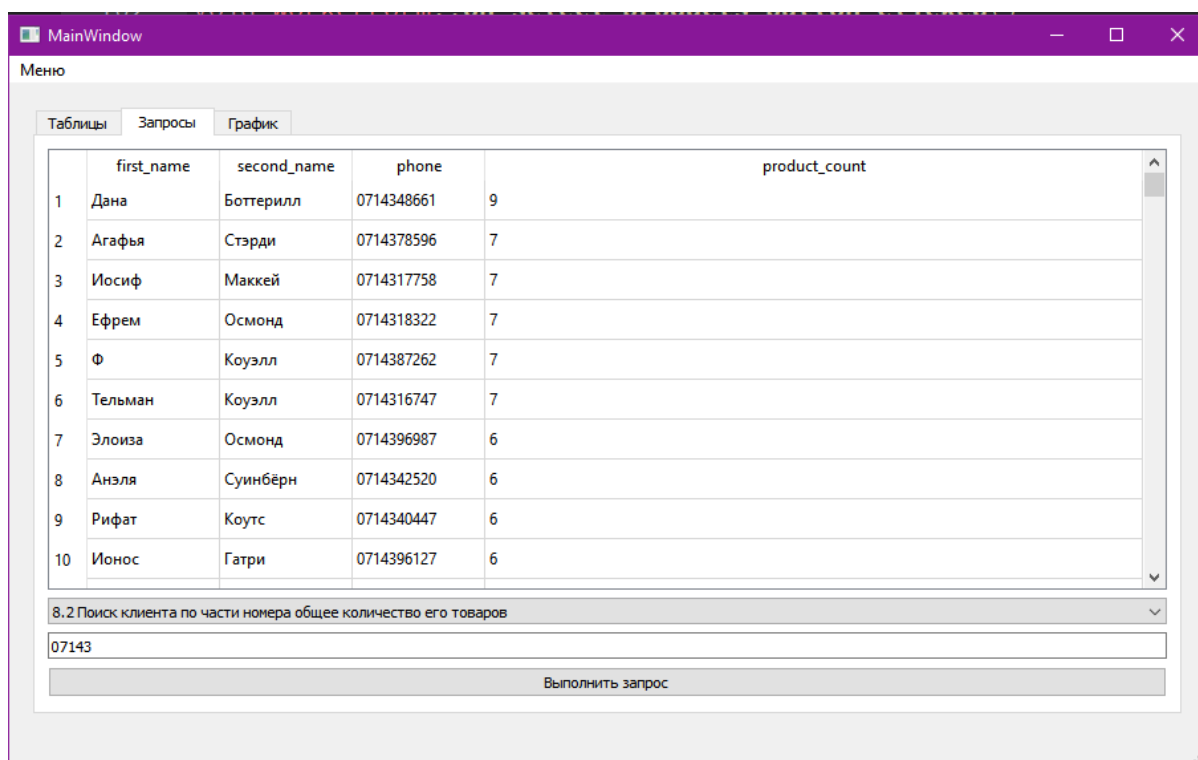


Рисунок 4.50– Клиент, номер которого содержит «07143» и общее количество его товаров

Итоговый запрос с условием на данные с использованием индекса используется для получения информации об имени продукта, номере квитанции, с которой был принят товар, имени магазина, в котором

находится товар, количестве чеков, которые были выписаны на этот товар, у товара с указанным индексом. Индекс товара задает пользователь. (рис. 4.51 – 4.53)

```
create index realization_ticket_index
on realization (id_product);
```

Рисунок 4.51– Создание индекса для таблицы реализации

```
select t1.product_name, t1.quittance, t1.shop, t1.realizations_count
from (select p.product_name,
            p.quittance,
            s.name                      as shop,
            count(r.realization_count)::integer as realizations_count
from realization r
    right join products p on r.id_product = p.id
    inner join shops s on p.id_shop = s.id
where r.id_product = product_id
group by p.id, s.name, p.quittance) as t1;
```

Рисунок 4.52– Итоговый запрос с условием на данные с использованием индекса

MainWindow

Меню

Таблицы Запросы График

	product_name	quittance	shop	realizations_count
1	Дырявый аксессуарб	123	Комиссионный №285920	3

8.3 Количество чеков выписанных, на этот товар

123

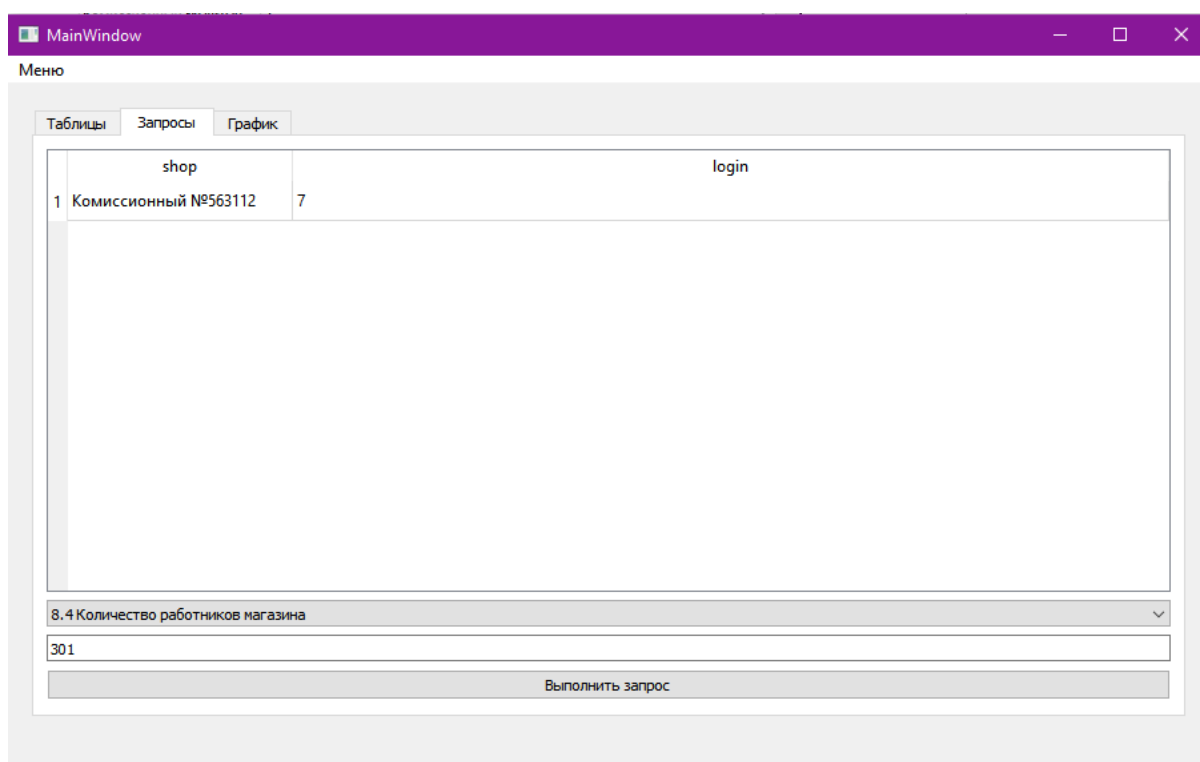
Выполнить запрос

Рисунок 4.53 – Количество чеков, выписанных на товар, имеющий номер квитанции 123

Итоговый запрос с условием на данные используется для получения информации об имени магазина, количестве работников в магазине, у магазина с указанным индексом. Индекс магазина указывается пользователем. (рис. 4.54 – 4.55)

```
select t1.name, count(t1.login) :: integer
from (select s.name, w.login
      from shops s
      right join workers w on s.id = w.id_shop
      where s.id = shop_id) as t1
group by t1.name;
```

Рисунок 4.54- Итоговый запрос с условием на данные



shop	login
1 Комиссионный №563112	7

8.4 Количество работников магазина

301

Выполнить запрос

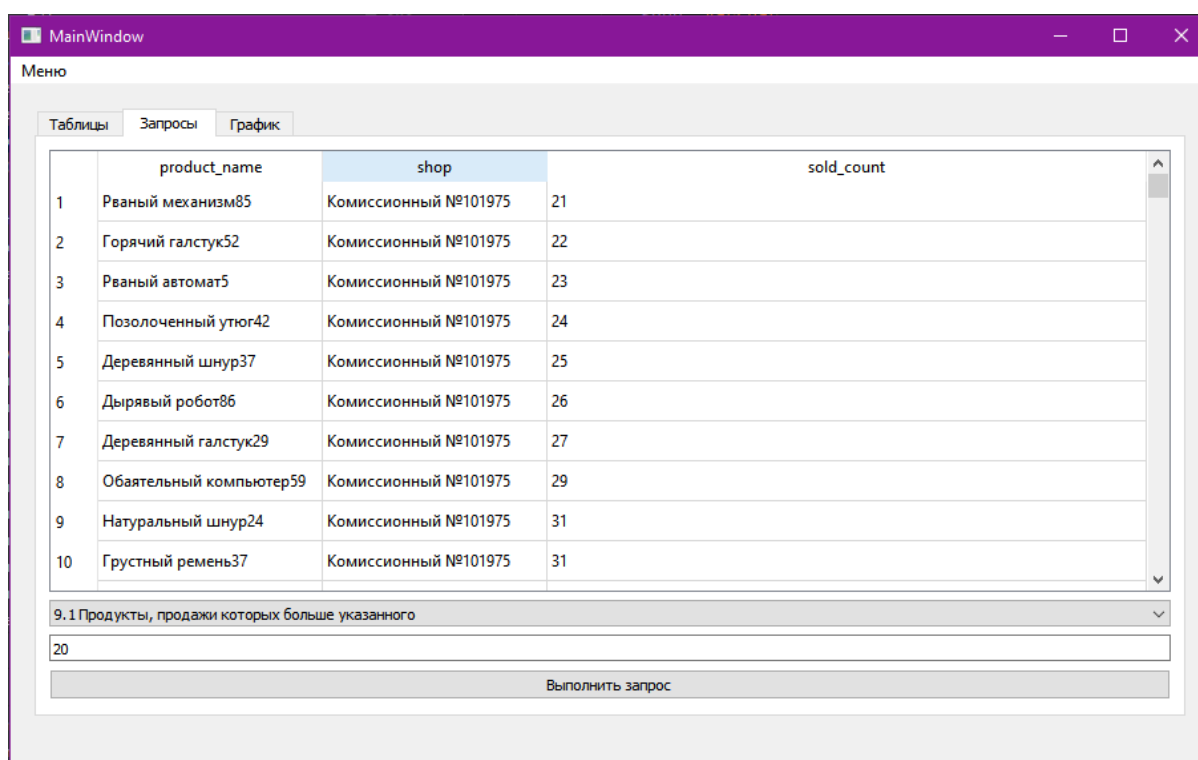
Рисунок 4.55 – Количество работников магазина с идентификационным номером 301

Итоговый запрос с условием на группы показывает пользователю информацию об имени продукта, названии магазина, в котором находится

продукт, общем количестве реализаций продукта, для продуктов, общее количество реализаций которых больше указанного. Количество реализаций указывает пользователь. Код представлен на рисунке 4.56, а результат – 4.57.

```
select t1.product_name, t1.shop, t1.sold
from (select p.product_name, s.name as shop, sum(r.realization_count)::integer as sold
      from realization r
           inner join products p on r.id_product = p.id
           inner join shops s on p.id_shop = s.id
      group by p.id, p.product_name, shop
      having sum(r.realization_count) > min_sold_count
      order by shop, sold) as t1;
```

Рисунок 4.56– Итоговый запрос с условием на группы



	product_name	shop	sold_count
1	Рванный механизм85	Комиссионный №101975	21
2	Горячий галстук52	Комиссионный №101975	22
3	Рванный автомат5	Комиссионный №101975	23
4	Позолоченный утюг42	Комиссионный №101975	24
5	Деревянный шнур37	Комиссионный №101975	25
6	Дырявый робот86	Комиссионный №101975	26
7	Деревянный галстук29	Комиссионный №101975	27
8	Обаятельный компьютер59	Комиссионный №101975	29
9	Натуральный шнур24	Комиссионный №101975	31
10	Грустный ремень37	Комиссионный №101975	31

9.1 Продукты, продажи которых больше указанного

20

Выполнить запрос

Рисунок 4.57 – Продукты, продажи которых больше 20

Итоговый запрос с условием на данные и группы используем для вывода информации об имени и фамилии пользователя, имени магазина, социальном статусе, количестве продуктов, которые пользователь сдал в

магазин, для пользователей с указанным социальным статусом и количеством сданных товаров больше указанного (см. рис. 4.58 – 4.59). Социальный статус и количество на вход подаёт пользователь.

```
select t1.customer, t1.shop, t1.status, t1.products_count
from (select (c.first_name || ' ' || c.second_name) :: varchar as customer,
            s.name                                     as shop,
            s2.status,
            count(p.quittance)::integer               as products_count
      from products p
      inner join customers c on p.id_customer = c.id
      inner join shops s on p.id_shop = s.id
      inner join statuses s2 on c.id_status = s2.id
     where s2.status = status1
     group by p.id_customer, customer, shop, s2.status
    having count(p.quittance) > prod_count
   order by count(p.quittance) desc) as t1;
```

Рисунок 4.58 – Итоговый запрос с условием на данные и на группы

	customer	shop	status	products_count
1	Ждан Ньюболд	Комиссионны...	предпринимат...	2
2	Изабелла Лейк	Комиссионны...	предпринимат...	2
3	Таисия Манселл	Комиссионны...	предпринимат...	2
4	Парамон Орр	Комиссионны...	предпринимат...	2
5	Анита ...	Комиссионны...	предпринимат...	2
6	Адис Кэмпиион	Комиссионны...	предпринимат...	2
7	Ф Дэнсон	Комиссионны...	предпринимат...	2
8	Руслан Бетелл	Комиссионны...	предпринимат...	2
9	Василиса Лейк	Комиссионны...	предпринимат...	2
10	Ратмир Даблдэй	Комиссионны...	предпринимат...	2

10.1 Пользователи с указанным социальным статусом и количеством сданных товаров больше указанного

предприниматель 1

Выполнить запрос

Рисунок 4.59 – Предприниматели, у которых сданных товаров больше 1

Запрос на запросе по принципу итогового запроса в данной системе создаётся для демонстрации средней цены товара в магазине и мени магазина. (см. рис. 4.60 – 4.61).

```
select (select avg(price) from products
where id_shop = s.id)::integer as average_price, s.name as shop
from shops s
order by average_price desc;
```

Рисунок 4.60– Запрос на запросе по принципу итогового запроса

	average_price	shop
1	5667	Комиссионный №285996
2	5609	Комиссионный №40528
3	5564	Комиссионный №478208
4	5549	Комиссионный №611106
5	5546	Комиссионный №640341
6	5544	Комиссионный №742011
7	5519	Комиссионный №692274
8	5514	Комиссионный №769753
9	5499	Комиссионный №114652
10	5498	Комиссионный №881033
11.1 Средняя стоимость всех товаров магазина		
предприниматель	1	
Выполнить запрос		

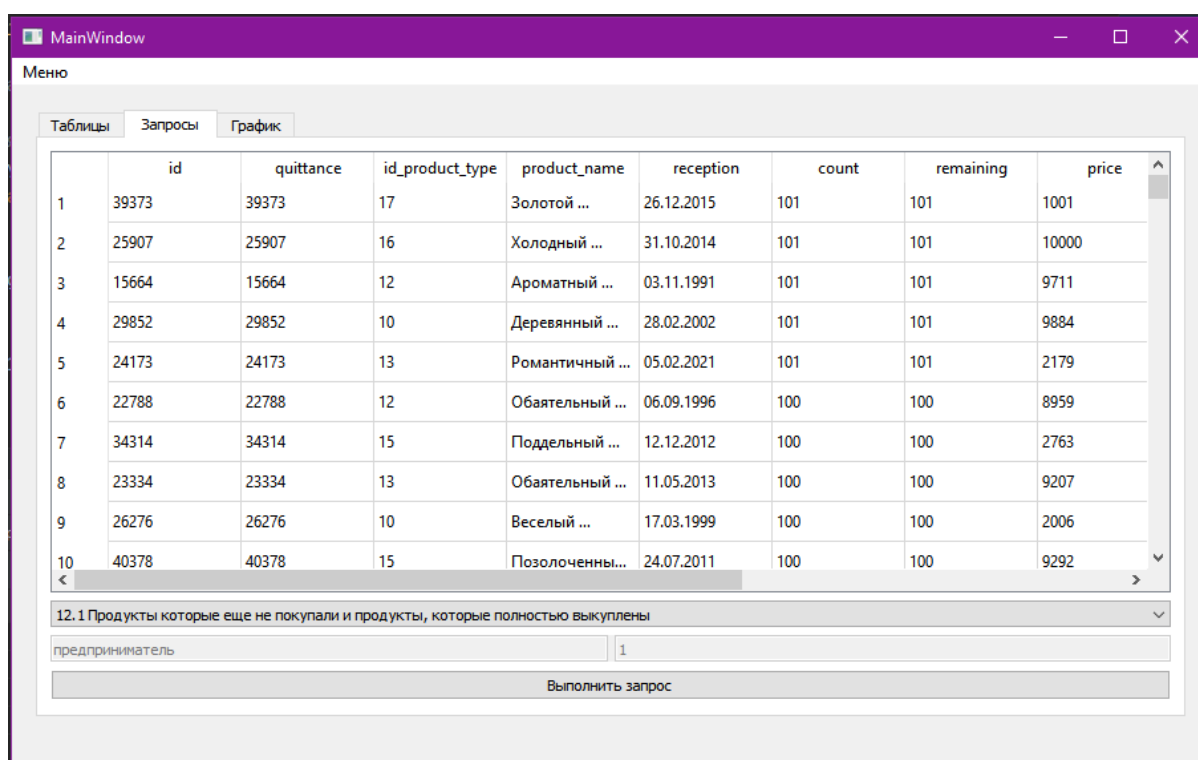
Рисунок 4.61 – Средняя стоимость всех товаров магазинов

Оператор языка UNION предназначен для объединения результирующих таблиц базы данных, полученных с применением слова SELECT. Условие объединения результирующих таблиц: совпадение числа, порядка следования и типа данных столбцов.

Используем данный тип запроса для того, чтобы получить полную информацию о продуктах, которые проданы полностью и продуктах, которые не проданы ни разу. Код представлен на рисунке 4.62, результат работы – 4.63.

```
select *
from products
where count = remaining
union all
select *
from products
where remaining = 0
order by remaining desc;
```

Рисунок 4.62 – Запрос с использованием объединения



	id	quittance	id_product_type	product_name	reception	count	remaining	price
1	39373	39373	17	Золотой ...	26.12.2015	101	101	1001
2	25907	25907	16	Холодный ...	31.10.2014	101	101	10000
3	15664	15664	12	Ароматный ...	03.11.1991	101	101	9711
4	29852	29852	10	Деревянный ...	28.02.2002	101	101	9884
5	24173	24173	13	Романтичный ...	05.02.2021	101	101	2179
6	22788	22788	12	Обаятельный ...	06.09.1996	100	100	8959
7	34314	34314	15	Поддельный ...	12.12.2012	100	100	2763
8	23334	23334	13	Обаятельный ...	11.05.2013	100	100	9207
9	26276	26276	10	Веселый ...	17.03.1999	100	100	2006
10	40378	40378	15	Позолоченны...	24.07.2011	100	100	9292

12.1 Продукты которые еще не покупали и продукты, которые полностью выкуплены

предприниматель 1

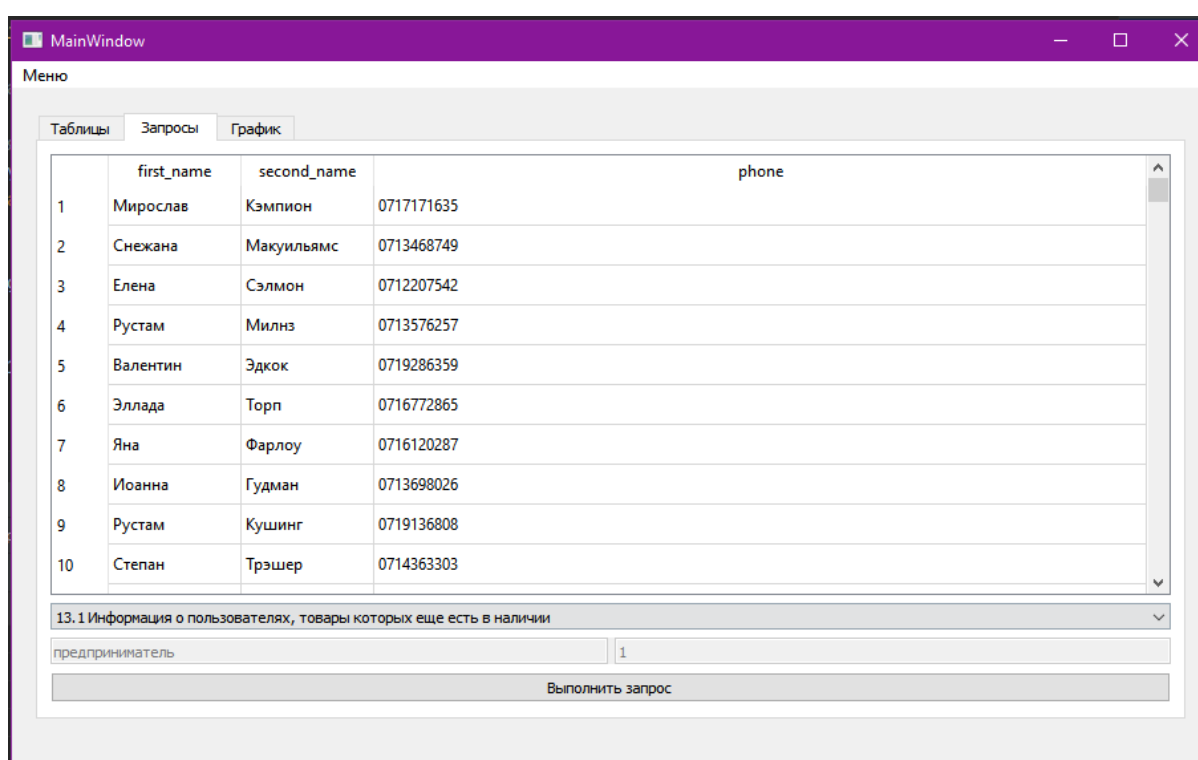
Выполнить запрос

Рисунок 4.63 – Товары, которые полностью скупили и товары, которые еще не покупали

Запрос с подзапросом с использованием `in`. Запрос позволяет получить информацию об имени и фамилии клиента, номере телефона, для клиентов, товары которых еще остались на складе. (рис. 4.64 – 4.65)

```
select first_name, second_name, phone
from customers
where id in (select id_customer from products where remaining > 0);
```

Рисунок 4.64 – Запрос с подзапросом с использованием `in`



	first_name	second_name	phone
1	Мирослав	Кэмпиион	0717171635
2	Снежана	Макуильямс	0713468749
3	Елена	Сэлмон	0712207542
4	Рустам	Милнз	0713576257
5	Валентин	Эдкок	0719286359
6	Эллада	Торп	0716772865
7	Яна	Фарлоу	0716120287
8	Иоанна	Гудман	0713698026
9	Рустам	Кушинг	0719136808
10	Степан	Трэшер	0714363303

13.1 Информация о пользователях, товары которых еще есть в наличии

предприниматель 1

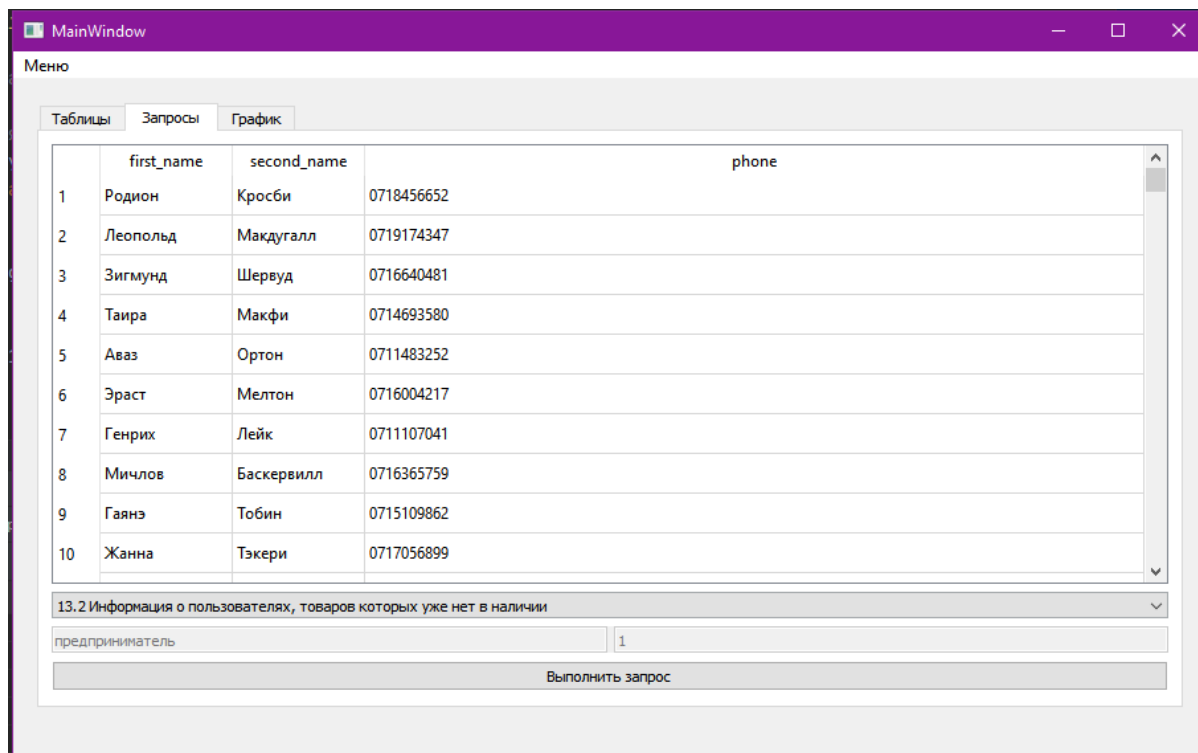
Выполнить запрос

Рисунок 4.65 – Пользователи, товары которых еще есть в наличии

Запрос с подзапросом с использованием `in`. Запрос позволяет получить информацию об имени и фамилии клиента, его номере телефона, для клиентов, товаров которых уже не осталось на складе. (рис. 4.66 – 4.67)

```
create or replace view query_13_2 as
select first_name, second_name, phone
from customers
where id not in (select id_customer from products where remaining > 0);
```

Рисунок 4.66 - Запрос с подзапросом с использованием not in



	first_name	second_name	phone
1	Родион	Кросби	0718456652
2	Леопольд	Макдугалл	0719174347
3	Зигмунд	Шервуд	0716640481
4	Таира	Макфи	0714693580
5	Аваз	Ортон	0711483252
6	Эраст	Мелтон	0716004217
7	Генрих	Лейк	0711107041
8	Мичлов	Баскервилл	0716365759
9	Гаянэ	Тобин	0715109862
10	Жанна	Тэкери	0717056899

13.2 Информация о пользователях, товаров которых уже нет в наличии

предприниматель: 1

Выполнить запрос

Рисунок 4.67 – Пользователи, товаров которых уже нет в наличии

Запрос с подзапросом с использованием case. Запрос позволяет получить информацию о номере квитанции, с которой был принят товар, имени товара, цене товара, оставшемся количестве товара на складе, количестве товара при поступлении, популярности продукта у покупателей. Популярность определяется по отношению оставшегося количества товара к исходному количеству товара (0% - раскупили, 1-20% - очень хорошо покупается, 21-40% - хорошо покупается, 41-60% - покупается, 61-80% - плохо покупается, 80-99% - очень плохо покупается, 100% - не покупается). (рис. 4.68 – 4.69)

```

select quittance,
       product_name,
       price,
       remaining,
       count,
       case
         when (remaining::float / count * 100)::integer = 100
           then 'Не покупается (' || (remaining::float / count * 100)::integer || '%)'
         when (remaining::float / count * 100)::integer between 80 and 99 then
           'Очень плохо покупается (' || (remaining::float / count * 100)::integer || '%)'
         when (remaining::float / count * 100)::integer between 60 and 80
           then 'Плохо покупается (' || (remaining::float / count * 100)::integer || '%)'
         when (remaining::float / count * 100)::integer between 40 and 60
           then 'Покупается (' || (remaining::float / count * 100)::integer || '%)'
         when (remaining::float / count * 100)::integer between 20 and 40
           then 'Хорошо покупается (' || (remaining::float / count * 100)::integer || '%)'
         when (remaining::float / count * 100)::integer between 1 and 20 then
           'Очень хорошо покупается (' || (remaining::float / count * 100)::integer || '%)'
         when (remaining::float / count * 100)::integer = 0
           then 'Раскупили (' || (remaining::float / count * 100)::integer || '%)'
         else (remaining::float / count * 100)::integer || ' error' end as state
from products
order by (remaining::float / count * 100)::integer, count desc;

```

Рисунок 4.68 - Запрос с подзапросом с использованием case

MainWindow

Меню

Таблицы Запросы График

	quittance	product_name	price	remaining	count	state
1	4070	Железный ...	8038	0	101	Раскупили (0%)
2	11125	Вонючий ...	9041	0	101	Раскупили (0%)
3	48032	Романтичный ...	9437	0	101	Раскупили (0%)
4	3288	Грустный ...	4088	0	101	Раскупили (0%)
5	8734	Обаятельный ...	2855	0	101	Раскупили (0%)
6	19794	Деревянный ...	4976	0	101	Раскупили (0%)
7	10246	Дырявый диск69	5710	0	101	Раскупили (0%)
8	41966	Ароматный ...	9667	0	101	Раскупили (0%)
9	2530	Вонючий ...	3709	0	101	Раскупили (0%)
10	16313	Призрачный ...	1338	0	101	Раскупили (0%)

13.3 Популярность продукта у покупателей

предприниматель 1

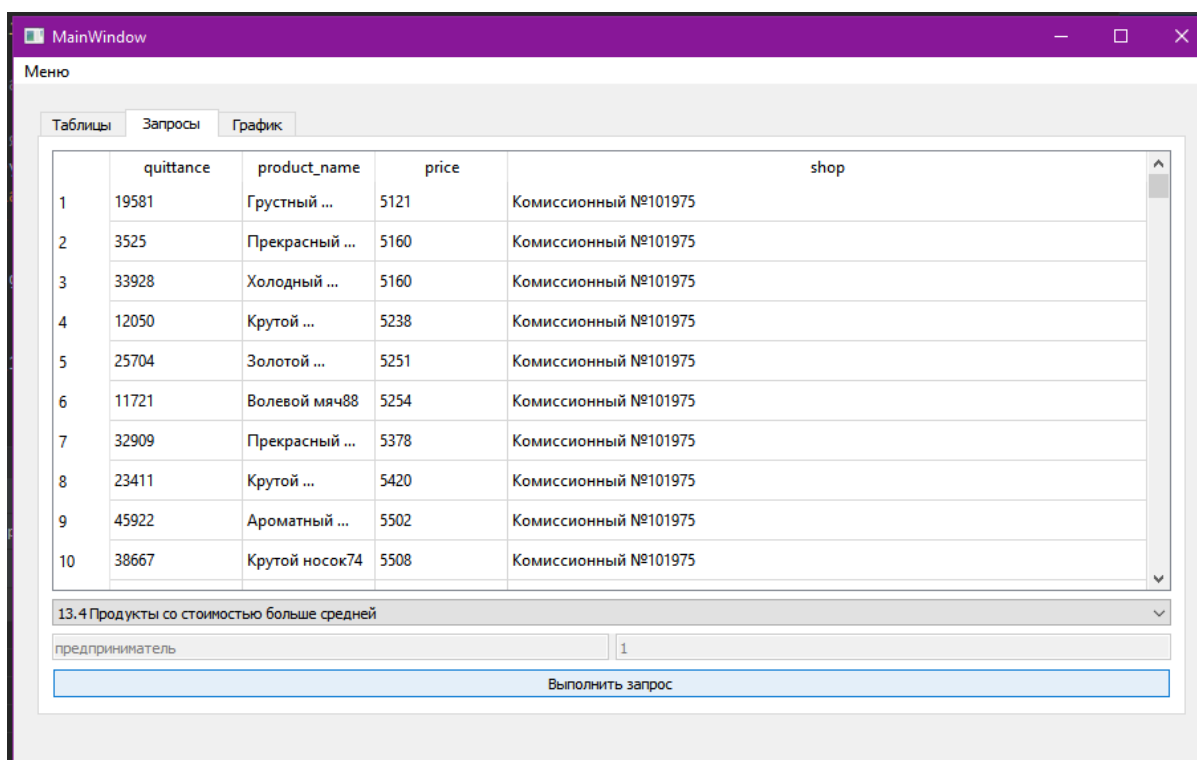
Выполнить запрос

Рисунок 4.69 – Популярность продуктов у покупателей

Запрос с подзапросом с использованием операций над итоговыми данными. Запрос позволяет получить информацию о номере квитанции, с которой был принят товар, имени товара, цене товара, имени магазина, в котором находится . (рис. 4.70 – 4.71)

```
select quittance, product_name, price, s.name as shop
from products
      inner join shops s on products.id_shop = s.id
where price > (select avg(price) from products)
order by shop, price;
```

Рисунок 4.70 - Запрос с подзапросом с использованием операций над
ИТОГОВЫМИ ДАННЫМИ



	quittance	product_name	price	shop
1	19581	Грустный ...	5121	Комиссионный №101975
2	3525	Прекрасный ...	5160	Комиссионный №101975
3	33928	Холодный ...	5160	Комиссионный №101975
4	12050	Крутой ...	5238	Комиссионный №101975
5	25704	Золотой ...	5251	Комиссионный №101975
6	11721	Волевой мяч88	5254	Комиссионный №101975
7	32909	Прекрасный ...	5378	Комиссионный №101975
8	23411	Крутой ...	5420	Комиссионный №101975
9	45922	Ароматный ...	5502	Комиссионный №101975
10	38667	Крутой носок74	5508	Комиссионный №101975

13.4 Продукты со стоимостью больше средней

предприниматель 1

Выполнить запрос

Рисунок 4.71 – Продукты со стоимостью больше средней

4.5 Создание представлений и хранимых функций

Представления – это таблицы чье содержание выбирается или получается из других таблиц, при этом они не содержат никаких собственных данных.

Представление – это фактически запрос, который выполняется всякий раз, когда представление становится темой команды. Вывод запроса при этом в каждый момент становится содержанием представления [6].

CREATE VIEW создаёт представление запроса.

Все запросы, которые выполняются без параметров, реализованы с помощью представлений. Рассмотрим пример создания представления (см. рис. 4.72).

```
create or replace view query_13_2 as
select first_name, second_name, phone
from customers
where id not in (select id_customer from products where remaining > 0);
```

Рисунок 4.72 – Представление для запроса с левым соединением

Остальные запросы реализованы аналогичным способом.

Хранимая процедура – это объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Postgres позволяет вызывать функции с именованными параметрами. Команда CREATE FUNCTION определяет новую функцию.

Рассмотрим пример создания функции для запроса (см. рис. 4.73).

```

create or replace function query_8_4(shop_id integer)
    returns table
    (
        shop varchar,
        login integer
    )
as
$$
begin
    return query select t1.name, count(t1.login) :: integer
        from (select s.name, w.login
            from shops s
                right join workers w on s.id = w.id_shop
            where s.id = shop_id) as t1
        group by t1.name;
end;
$$ language plpgsql SECURITY DEFINER;

```

Рисунок 4.73 – Пример создания функции для итогового запроса с условием на данные и группы

В данном случае отличие от представлений в том, что мы можем передать параметры; указывает тип возвращаемых данных; также нужно указывать язык, на котором, пишется функция.

Остальные параметризованные запросы аналогичным способом помещаются в такую «оболочку».

5 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

При разработке клиентского приложения была необходимость сделать разные интерфейсы для разных видов пользователей. Это было достигнуто благодаря определению роли пользователя после авторизации, после чего производится выбор интерфейса в зависимости от определенной при авторизации роли.

Первой формой всегда является форма авторизации (см. рис. 5.1).

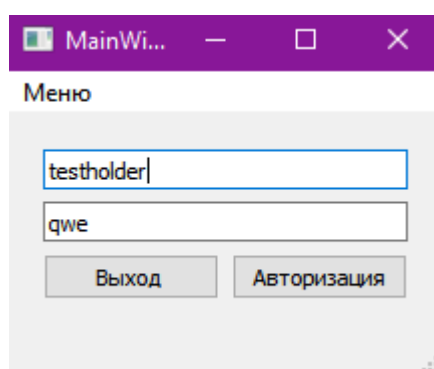


Рисунок 5.1 – Форма авторизации

В этой форме пользователь может авторизоваться либо выйти. Меню содержит один пункт – выйти из профиля, и сохраняется для любой формы, но в форме авторизации при нажатии на нее ничего не произойдет.

Для определения роли входящего используется следующий механизм: сначала форма авторизации испускает сигнал, который запрашивает проверку роли в базе данных по введенному логину и паролю (рис.).

```
void Authorization::on_login_button_clicked()
{
    emit TryToLogin(ui->login_le->text(), ui->password_le->text());
}
```

Рисунок 5.2 – Запрос на проверку роли по логину и паролю

Главная форма обрабатывает этот сигнал и выполняет проверку роли по присланному логину и паролю (рис. 5.3). В зависимости от результата выполняется либо подключение к базе данных (рис. 5.4) и переход на нужную форму, либо, если введенный логин или пароль не верен, то пользователю выводится предупреждение об ошибке аутентификации.

```
connect(auth, &Authorization::TryToLogin, this, [this](const QString& login,
    const QString& password){
    ConnectDB(login, password);
    QSqlQuery q;
    q.exec("select current_user");
    q.first();
    qDebug() << q.value(0).toString();
    if (CheckUserGroup("worker")){
        SetPage(new WorkerForm);
    } else if (CheckUserGroup("administrator")) {
        SetPage(new AdministratorForm);
    } else if (CheckUserGroup("holder")){
        SetPage(new HolderForm);
    }
});
```

Рисунок 5.3 – Проверка роли пользователя

```
void MainWindow::ConnectDB(QString login, QString password){
    database = QSqlDatabase::addDatabase("QPSQL");
    database.setDatabaseName("store");
    database.setHostName("localhost");
    database.setPassword(password);
    database.setPort(5432);
    database.setUserName(login);
    if (database.open()) {
        qDebug() << "Database have been open";
    } else {
        QMessageBox::warning(nullptr, "Ошибка", "Неверный логин или пароль");
        qDebug() << "Database have not been open";
        qDebug() << database.lastError();
    }
}
```

Рисунок 5.4– Подключение к базе данных

5.1 Работник(worker)

Для работников был разработан интерфейс, который разделен на 3 части: «Продукты» (рис. 5.5), «Реализация» (рис. 5.6) и «Клиенты» (рис. 5.7). Подробнее про работу с формой работника можно прочитать в руководстве работника (приложение Д).

The screenshot displays the 'MainWindow' application interface. The 'Продукты' (Products) tab is active. The main area contains a table with the following data:

	product_type	product_name	reception	remaining	price
1	посуда	Позолоченный...	25.09.2020	25	6004
2	инструменты	Ароматный ...	07.06.2020	4	8977
3	компьютерная...	Позолоченный...	02.06.2020	2	9738
4	одежда	Романтичный ...	15.05.2020	35	5435
5	игрушки	Позолоченный...	30.04.2020	2	8372
6	инструменты	Горячий ...	22.01.2020	5	2221
7	игрушки	Романтичный ...	03.02.2019	1	8871
8	компьютерная...	Веселый ...	10.01.2019	1	8504
9	товары для дома	Дырявый ...	08.10.2018	8	10083
10	одежда	Прикольный ...	25.09.2018	56	9011
11	обувь	Призрачный ...	30.01.2018	1	6085
12	товары для ...	Дырявый ...	05.10.2017	11	9113
13	инструменты	Золотой мяч68	05.09.2017	4	2651
14	товары для ...	Холодный ...	23.07.2017	69	1901
15	игрушки	Поддельный ...	06.02.2017	15	4192

On the right, the 'Подобная информация' (Similar information) panel shows details for the selected product (ID 13):

- Дата приема: 2017-09-05
- Количество при приеме: 63
- Информация о заказчике: Имя: Терентий, Фамилия: Блэр, Район: Киевский
- Социальный статус: предприниматель
- Место работы: пр-т. Мажковского 5в
- Дата рождения: 1990-05-13
- Номер телефона: 0719314084

At the bottom, there are search and action buttons: 'Имя товара' (search), 'Поиск по имени', 'Просмотр товаров в наличии', 'Имя продукта' (dropdown), 'Тип продукта' (dropdown), 'Количество' (input), 'Цена' (input), 'Идентификатор пользователя' (input), 'Продать выделенный товар', and 'Добавить товар'.

Рисунок 5.5 – Форма работника для взаимодействия с продуктами

MainWindow

Меню

Продукты Реализация Клиенты

Информация о реализации

	ticket	product_name	quittance	realization_date	realization_count
1	44	Позолоченны...	217	12.08.1967	1
2	38	Железный ...	367	06.09.1988	1
3	950	Крутой диск67	582	21.04.2017	3
4	4258	Крутой диск67	582	12.08.2002	1
5	87017	Крутой диск67	582	29.03.2001	1
6	2558	Крутой диск67	582	16.08.2017	5
7	1917	Веселый ...	656	27.09.2016	1
8	3797	Железный ...	1046	21.02.2020	1
9	1705	Железный ...	1046	07.06.2020	1
10	911	Железный ...	1046	24.10.2019	1
11	114403	Дырявый утюг64	1266	07.04.1998	7
12	112424	Дырявый утюг64	1266	29.12.1997	34
13	2860	Дырявый утюг64	1266	06.09.2008	2
14	3252	Горячий ...	1278	02.08.2004	18
15	69956	Горячий ...	1278	21.08.2018	1

Количество реализованных (Больше указанного): 0

Поиск по реализации

Информация о товаре

	product_type	product_name	reception	remaining	price
1	посуда	Обаятельный ...	19.02.2021	0	6200
2	товары для дома	Прекрасный ...	09.11.2020	0	2833
3	посуда	Позолоченны...	25.09.2020	25	6004
4	одежда	Прикольный ...	06.08.2020	0	650
5	инструменты	Ароматный ...	07.06.2020	4	8977
6	компьютерная...	Позолоченны...	02.06.2020	2	9738
7	одежда	Романтичный ...	15.05.2020	35	5435
8	игрушки	Позолоченны...	30.04.2020	2	8372
9	компьютерная...	Одаренный ...	18.04.2020	0	8460
10	посуда	Романтичный ...	30.03.2020	0	1978
11	инструменты	Горячий ...	22.01.2020	5	2221
12	электроника	Горячий ...	11.10.2019	0	9220
13	одежда	Ароматный ...	13.06.2019	0	4799

Имя продукта

Поиск по товарам

Рисунок 5.6 – Форма работника для взаимодействия между реализациями и товарами

MainWindow

Меню

Продукты Реализация Клиенты

Информация о клиенте

	id	first_name	second_name	phone	work
46	14836	Игнат	Кэмпиион	0716091422	пер. Ткаченко 139
47	14781	Христя	Маккей	0712085823	ул. Ватутина 73
48	14617	Гюзель	Гудман	0719752542	ул. Конева 186г
49	14166	Геннадий	Мелтон	0719181545	пер. Байдукова ...
50	13620	Измаил	Бродерик	0719518896	пер. Шекспира 15г
51	13603	Гарри	Эмерсон	0713340011	ул. Лагутенко 1036
52	13438	Василий	Талли	0714043274	пр-т. Орешкова ...
53	13262	Венедикт	Бутби	0712345941	пр-т. Орешкова ...
54	13176	Алиса	Фрэмплтон	0713420022	ул. Калинина 182г
55	13097	Иоанна	Мэнселл	0719244245	ул. Калинина 72г
56	12933	Ярослав	Лейк	0714985572	ул. Байдукова 28в
57	12670	Архип	Макдугал	0714542867	ул. Конева 24

Поиск клиентов

Имя

Фамилия

Поиск по клиентам магазина

Поиск по всем клиентам

Добавление клиента

Имя

Фамилия

Киевооий

пенсионер

Место работы

01.01.2000

Телефон

Добавить клиента

Товары клиента

	product_type	product_name	reception	remaining	price
1	посуда	Обаятельный ...	19.02.2021	0	6200
2	товары для дома	Прекрасный ...	09.11.2020	0	2833
3	посуда	Позолоченны...	25.09.2020	25	6004
4	одежда	Прикольный ...	06.08.2020	0	650
5	инструменты	Ароматный ...	07.06.2020	4	8977
6	компьютерная...	Позолоченны...	02.06.2020	2	9738
7	одежда	Романтичный ...	15.05.2020	35	5435
8	игрушки	Позолоченны...	30.04.2020	2	8372
9	компьютерная...	Одаренный ...	18.04.2020	0	8460
10	посуда	Романтичный ...	30.03.2020	0	1978
11	инструменты	Горячий ...	22.01.2020	5	2221
12	электроника	Горячий ...	11.10.2019	0	9220
13	одежда	Ароматный ...	13.06.2019	0	4799
14	товары для дома	Веселый ...	20.05.2019	0	1015
15	игрушки	Романтичный ...	03.02.2019	1	8871

Название товара

Поиск по товарам

Рисунок 5.7 – Форма работника для взаимодействия с клиентами и их товарами

5.2 Администратор (administrator)

Для администратора был разработан интерфейс, который разделен на 2 части: «Роли» (рис. 5.8) и «Справочники» (рис. 5.9). Подробнее про работу с формой администратора можно прочитать в руководстве администратора (приложение Е).

The screenshot shows the 'Роли' (Roles) form. It features a table of roles with columns: id, login, is_available, and name. Below the table are input fields for 'Логин работника' (Worker login), 'Пароль работника' (Worker password), and a dropdown for 'Комиссионный №' (Commission number). There are buttons for 'Найти работника' (Find worker), 'Добавить работника' (Add worker), 'Перевести в другой магазин' (Move to another store), and 'Уволить работника' (Dismiss worker). On the right, there is a list of owners with a column 'rolname' and a button 'Найти владельца' (Find owner). Below this are input fields for 'Логин владельца' (Owner login) and 'Пароль владельца' (Owner password), along with buttons 'Добавить владельца' (Add owner) and 'Удалить владельца' (Delete owner).

	id	login	is_available	name
1	2001	worker1000	true	Комиссионный №694661
2	2000	worker999	true	Комиссионный №369450
3	1999	worker998	true	Комиссионный №460812
4	1998	worker997	true	Комиссионный №348291
5	1997	worker996	true	Комиссионный №553609
6	1996	worker995	true	Комиссионный №43131
7	1995	worker994	true	Комиссионный №2712

rolname
1 testholder

Рисунок 5.8 – Форма для работы с ролями

MainWindow

Меню

Роли Справочники

Районы

	id	district
1	8	Киевский
2	9	Буденовский
3	10	Петровский
4	11	Куйбышевский
5	12	Ворошиловский
6	13	Кировский
7	14	Калининский

Найти Добавить Удалить выбранное

Рисунок 5.9 – Форма для работы со справочниками

5.3 Владелец (holder)

Для владельца был разработан интерфейс, который разделен на 3 части: «Таблицы» (рис 5.10 – 5.11), «Запросы» (рис. 5.12) и «График» (рис. 5.13 – 5.15). Подробнее про работу с формой владельца можно прочитать в руководстве владельца (приложение Ё).

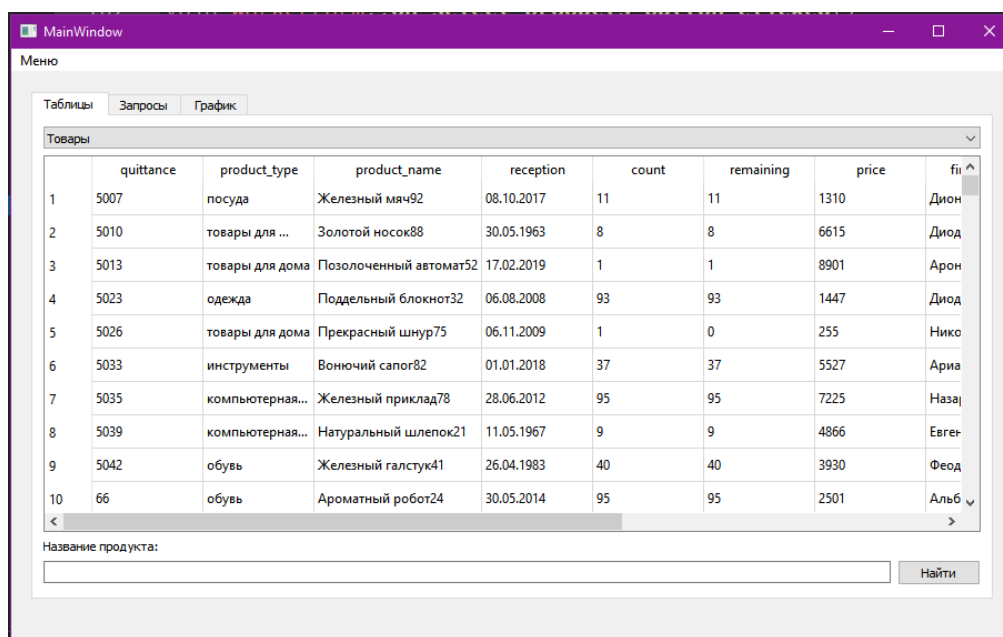


Рисунок 5.10 – Форма для работы с таблицами

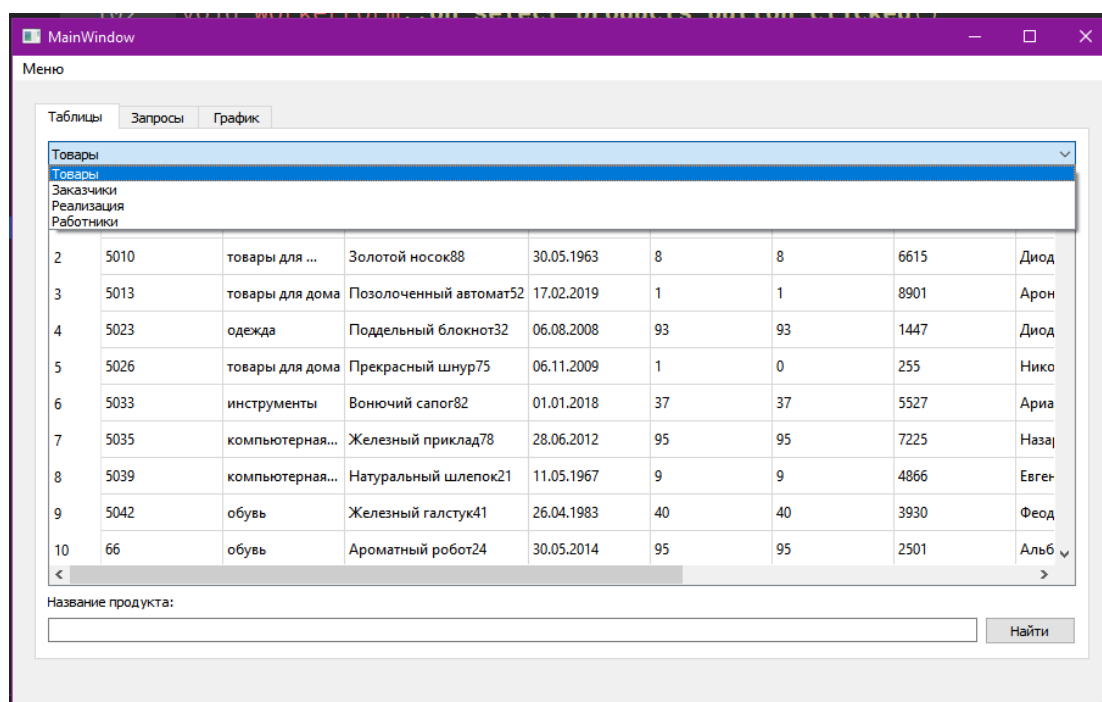


Рисунок 5.11– Выбор таблицы

MainWindow

Меню

Таблицы Запросы График

	quittance	product_name	price	shop
1	19581	Грустный ...	5121	Комиссионный №101975
2	3525	Прекрасный ...	5160	Комиссионный №101975
3	33928	Холодный ...	5160	Комиссионный №101975
4	12050	Крутой ...	5238	Комиссионный №101975
5	25704	Золотой ...	5251	Комиссионный №101975
6	11721	Волевой мяч88	5254	Комиссионный №101975
7	32909	Прекрасный ...	5378	Комиссионный №101975
8	23411	Крутой ...	5420	Комиссионный №101975
9	45922	Ароматный ...	5502	Комиссионный №101975
10	38667	Крутой носок74	5508	Комиссионный №101975

13.4 Продукты со стоимостью больше средней

предприниматель 1

Выполнить запрос

Рисунок 5.12 – Просмотр результатов запроса

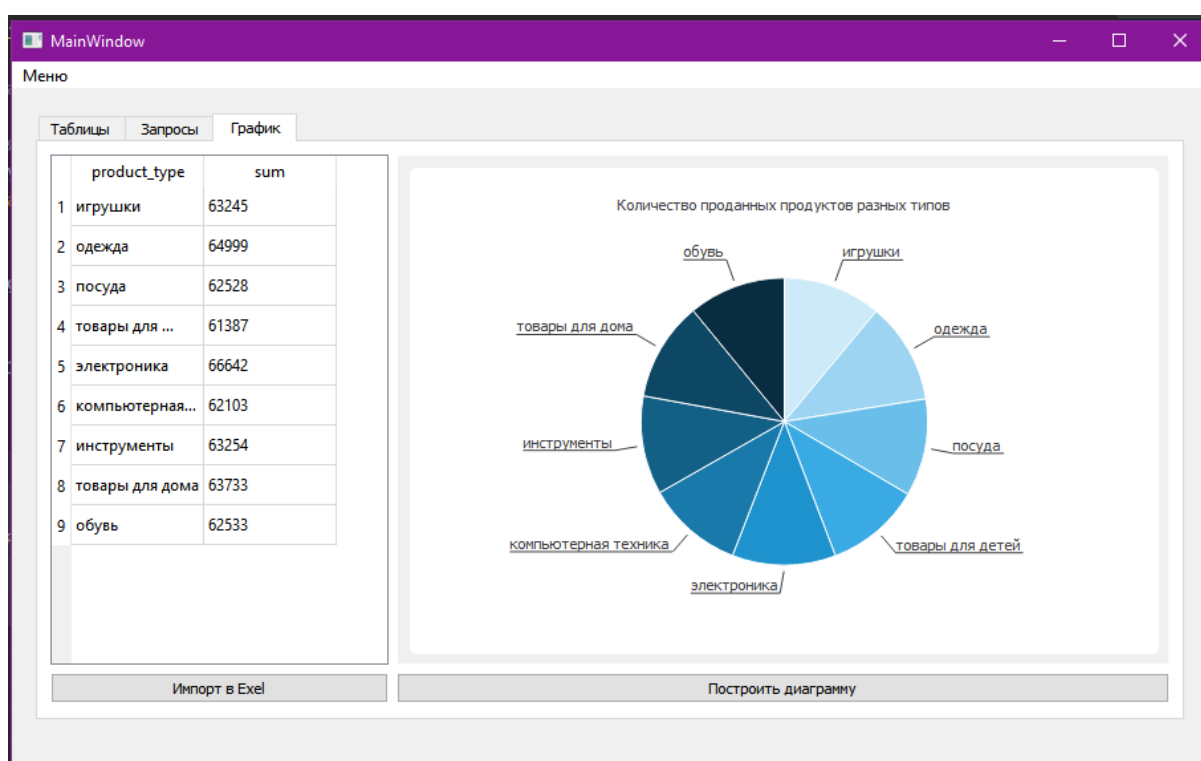


Рисунок 5.13 – Количество проданных продуктов разных типов

При нажатии на кнопку «Импорт в Excel» открывается форма выбора пути сохранения файла (рис. 5.10), после чего будет создан файл Excel содержащий записанную статистику (рис. 5.11).

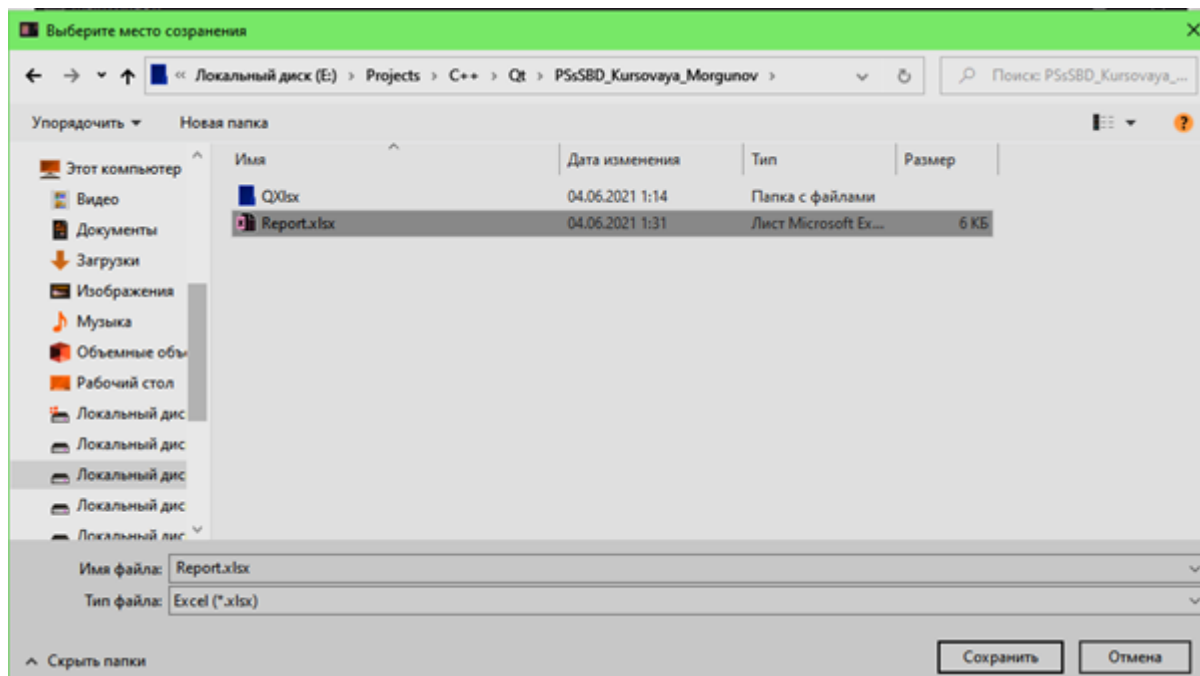


Рисунок 5.14 – Выбор места сохранения Excel

	A	B	C
1	product_type	sum	
2	игрушки	63245	
3	одежда	64999	
4	посуда	62528	
5	товары для детей	61387	
6	электроника	66642	
7	компьютерная тех	62103	
8	инструменты	63254	
9	товары для дома	63733	
10	обувь	62533	
11			

Рисунок 5.15– Результат экспорта в Excel

6 ТЕСТИРОВАНИЕ РАЗРАБОТАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ (В Т.Ч. ВКЛЮЧАЯ ЗАЩИТУ ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА, КАСКАДНОЕ УДАЛЕНИЕ)

6.1 Ошибки ввода

При разработке учитывались такие ошибки как: несанкционированный доступ (рис. 6.1).

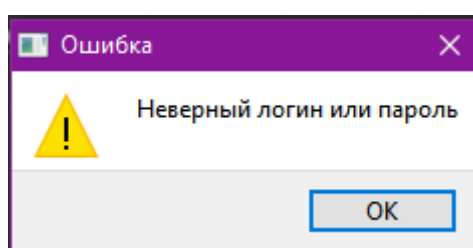


Рисунок 6.1 – Несанкционированный доступ

Клиентское приложение спроектировано так, чтобы предугадать ошибки пользователя и не дать ему их совершить. Например, пока пользователь не выберет товар, то кнопка «Продать товар» будет неактивна.

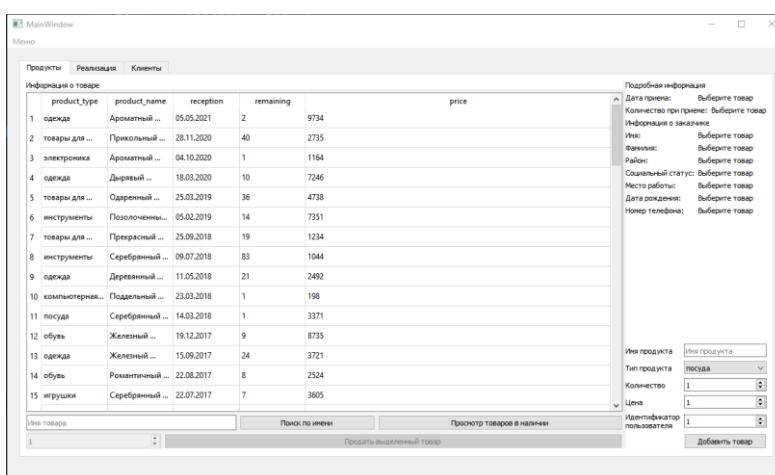


Рисунок 6.2 – Пока не выбран товар кнопка продажи неактивна

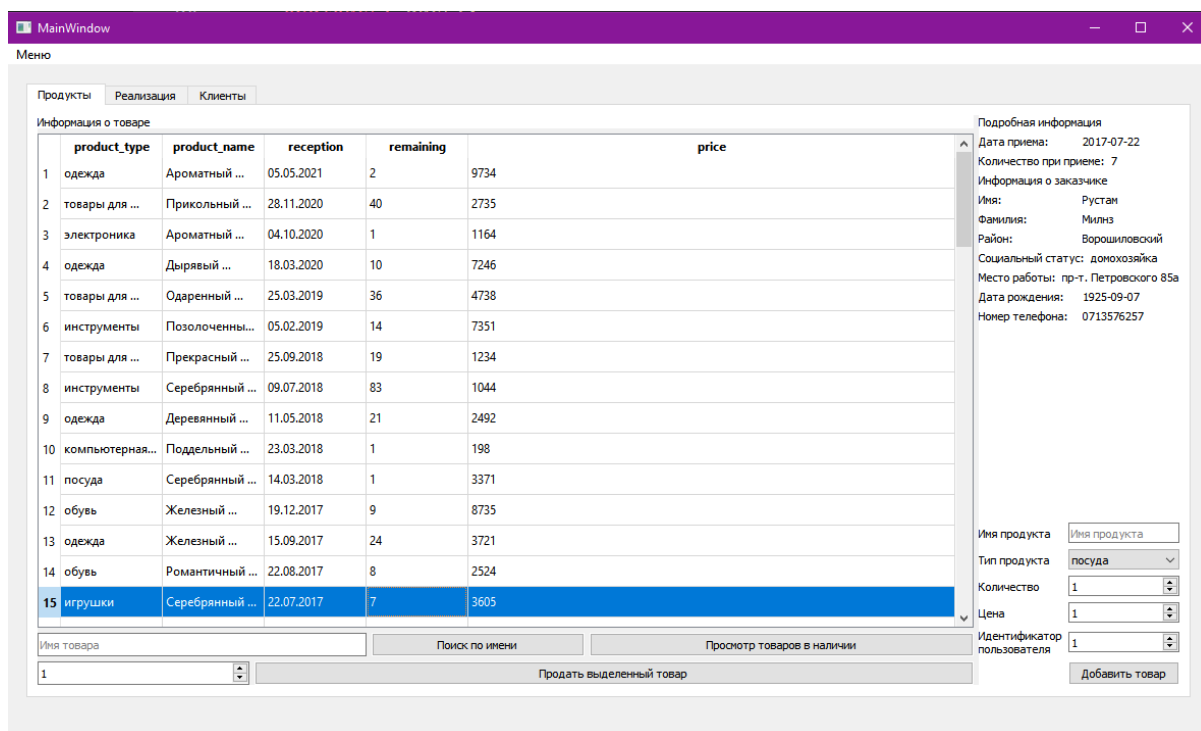


Рисунок 6.3 – При выборе товара она становится активна

Также обрабатывается возможная ошибка пользователя (рис. 6.4).

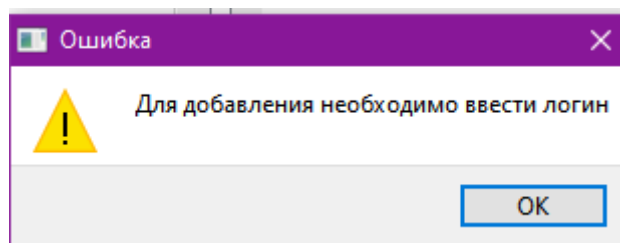


Рисунок 6.4 – Администратор пытается добавить пользователя с пустым логином

6.2 Каскадное удаление

При проектировании приложения был сделан упор на то, что никто не должен уметь затирать или изменять операции, которые уже были

выполнены. Это делается для того, чтобы минимизировать риск мошенничества.

Каскадное удаление реализовано при создании таблиц (п. 4.2). Но продемонстрировать его не представляется возможным по вышеописанной причине.

ЗАКЛЮЧЕНИЕ

Результат работы – система, реализующая систему для учета деятельности комиссионных магазинов города, реализованная при помощи языка программирования C++ и системы управления базами данных PostgreSQL.

К преимуществам программы относятся: удобный пользовательский интерфейс, реализация и управление базой данных с помощью клиентского интерфейса, возможность добавлять, удалять, редактировать, выбирать данные из таблиц. Возможен экспорт данных в excel. Безопасность для пользователя от случайного удаления данных. Возможен вывод диаграмм со статистическими данными. Просмотр результатов запросов различной сложности. Разграничение возможностей различных ролей. Возможность просматривать диаграммы.

При дальнейшей работе над системой можно смасштабировать ее до более крупных размеров, а также оптимизировать внутреннюю логику сервера для более эффективной работы с данными.

СПИСОК ЛИТЕРАТУРЫ

1. Основы технологий баз данных: учеб. пособие/ Б.А. Новиков, Е.А.Горшкова; под ред. Е.В. Рогова. – М.:ДМК Пресс, 2019. -240с
2. Малыхина, М. П. / Базы данных. Основы, проектирование, использование. — СПб.: БХВ-Петербург, 2004. — 512 с.
3. Сибилев В. Д. / Моделирование и проектирование баз данных. В 2 ч.: Учеб. пособие для вузов. — Томск: Изд-во ТУСУР, 2002. — 144 с.
4. Петров В. Н. / Информационные системы: Учеб. для вузов. — СПб.: Питер, 2003. — 688 с.
5. Хомоненко А. Д., Цыганков В. М., Мальцев М. Г. / Базы данных. Учебник для вузов — 4-е издание, доп. и перераб. — СПб.: КОРОНА принт, 2004. — 736 с.
6. Черенков А. П. / Информационные системы для экономистов: Учеб. пособие для вузов. — М.: Экзамен, 2004. — 192 с.

ПРИЛОЖЕНИЕ А. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

ПРИЛОЖЕНИЕ Б. ЛИСТИНГ КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

```

administratorform.cpp
#include "administratorform.h"
#include "ui_administratorform.h"
#include "QSql"
#include "QSqlQueryModel"
#include "helper.hpp"
#include <QDebug>
#include <QSqlError>
#include <QMessageBox>

struct dictionary_data{
    QString name;
    QString column_name;
};

Q_DECLARE_METATYPE(dictionary_data);

AdministratorForm::AdministratorForm(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::AdministratorForm)
{
    ui->setupUi(this);
    helper::setupTableView(ui->owners_table1);
    helper::setupTableView(ui->workers_table1);
    helper::setupTableView(ui->dictionary_table2);
    QSqlQuery role_query("set role administrator");

    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery(QString("select r.rolname from pg_roles r,
    pg_group g where g.groname = 'holder' and ARRAY[oid] && g.grolist "
    "and r.rolname like '%1%'");.arg(ui->owner_login_le1->text()));
    ui->owners_table1->setModel(query_model);

    QSqlQueryModel* query_model2 = new QSqlQueryModel(this);
    query_model2->setQuery("select id, login, is_available, "
    "(select name from shops where id = id_shop) from workers "
    "order by id desc");
    ui->workers_table1->setModel(query_model2);

    auto query = helper::sendQuery("select * from shops");
    while (query.next()) {
        ui->shop_cb1->addItem(query.value(1).toString(), query.value(0));
    }

    QVariant v;
    v.setValue(dictionary_data{"districts", "district"});
    ui->current_dictionary_cb2->addItem("Районы", v);
    v.setValue(dictionary_data{"statuses", "status"});
    ui->current_dictionary_cb2->addItem("Социальное положение", v);
    v.setValue(dictionary_data{"product_types", "product_type"});
    ui->current_dictionary_cb2->addItem("Типы товаров", v);
    v.setValue(dictionary_data{"shops", "name"});
    ui->current_dictionary_cb2->addItem("Магазины", v);

    int index = ui->current_dictionary_cb2->currentIndex();
    QString dictionary_name = qvariant_cast<dictionary_data>
    (ui->current_dictionary_cb2->itemData(index)).name;
    QString column_name = qvariant_cast<dictionary_data>
    (ui->current_dictionary_cb2->itemData(index)).column_name;

    QSqlQueryModel* query_model3 = new QSqlQueryModel(this);
    query_model3->setQuery(QString("select * from %1 where %2 like
    '%3%'").arg(
        dictionary_name,
        column_name,
        ui->name_le2->text()));
    ui->dictionary_table2->setModel(query_model3);
}

AdministratorForm::~AdministratorForm()
{
    delete ui;
}

void AdministratorForm::on_find_owner_b1_clicked()
{
    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery(QString("select r.rolname from pg_roles r,
    pg_group g where g.groname = 'holder' and ARRAY[oid] && g.grolist "
    "and r.rolname like '%1%'");.arg(ui->owner_login_le1->text()));
    ui->owners_table1->setModel(query_model);
}

void AdministratorForm::on_add_owner_b1_clicked()
{
    QSqlQuery query(QString("create role %1 login password '%2' in role
    holder").arg(
        ui->owner_login_le1->text(), ui-
    >owner_password_le1->text()));
    helper::reloadModel(ui->owners_table1);
}

void AdministratorForm::on_delete_owner_b1_clicked()
{
    // Как выбрать нужную запись в выбранной строке?
    // auto model = index.model()->index(index.row(), 0);
    // auto selected_owner = helper::data(ui->owners_table1-
    >currentIndex(), 0);
    QSqlQuery query(QString("drop role
    %1").arg(selected_owner.toString()));
    helper::reloadModel(ui->owners_table1);
    qDebug() << selected_owner;
}

void AdministratorForm::on_owners_table1_clicked(const QModelIndex &index)
{
}

void AdministratorForm::on_find_worker_b1_clicked()
{
    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery("select id, login, is_available, "
    "(select name from shops where id = id_shop) from workers "
    "order by id desc");
    ui->workers_table1->setModel(query_model);
}

void AdministratorForm::on_add_worker_b1_clicked()
{
    if (ui->worker_login_le1->text() == ""){
        QMessageBox::warning(nullptr, "Ошибка", "Для добавления необходимо
    ввести логин");
    }else{
        QSqlQuery query(QString("select create_worker('%1', '%2',
    %3)").arg(
            ui->worker_login_le1->text(),
            ui->worker_password_le1->text(),
            ui->shop_cb1->currentData().toString()));
        qDebug() << query.lastError();
        helper::reloadModel(ui->workers_table1);
    }
}

void AdministratorForm::on_delete_worker_b1_clicked()
{
    QSqlQuery query(QString("select fire_worker('%1')").arg(
    >currentIndex(), 1).toString());
    qDebug() << query.lastError();
    helper::reloadModel(ui->workers_table1);
}

void AdministratorForm::on_add_worker_b1_2_clicked()
{
    QSqlQuery query(QString("update workers set id_shop = %1 "
    "where id = %2").arg(ui->shop_cb1->currentData().toString(),
    helper::data(ui->workers_table1-
    >currentIndex(), 0).toString()));
    qDebug() << query.lastError();
    helper::reloadModel(ui->workers_table1);
}

void AdministratorForm::on_find_b2_clicked()
{
    int index = ui->current_dictionary_cb2->currentIndex();
    QString dictionary_name = qvariant_cast<dictionary_data>
    (ui->current_dictionary_cb2->itemData(index)).name;
    QString column_name = qvariant_cast<dictionary_data>
    (ui->current_dictionary_cb2->itemData(index)).column_name;

    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery(QString("select * from %1 where %2 like
    '%3%'").arg(
        dictionary_name,
        column_name,
        ui->name_le2->text()));
    ui->dictionary_table2->setModel(query_model);
}

void AdministratorForm::on_add_b2_clicked()
{
    int index = ui->current_dictionary_cb2->currentIndex();
    QString dictionary_name = qvariant_cast<dictionary_data>
    (ui->current_dictionary_cb2->itemData(index)).name;
    QString column_name = qvariant_cast<dictionary_data>
    (ui->current_dictionary_cb2->itemData(index)).column_name;

    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery(QString("insert into %1 (%2)
    values('%3')").arg(
        dictionary_name,
        column_name,
        ui->new_name_le2->text()));
    helper::reloadModel(ui->dictionary_table2->model());
    qDebug() << query_model->lastError();
    // ui->dictionary_table2->setModel(query_model);
}

void AdministratorForm::on_current_dictionary_cb2_activated(int index)
{
    QString dictionary_name = qvariant_cast<dictionary_data>
    (ui->current_dictionary_cb2->itemData(index)).name;

    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery(QString("select * from %1").arg(
    dictionary_name));
    ui->dictionary_table2->setModel(query_model);
}

```

```

void AdministratorForm::on_delete_b2_clicked()
{
    {
        int index = ui->current_dictionary_cb2->currentIndex();
        QString dictionary_name = qvariant_cast<dictionary_data>
            (ui->current_dictionary_cb2->itemData(index)).name;
        QString column_name = qvariant_cast<dictionary_data>
            (ui->current_dictionary_cb2->itemData(index)).column_name;

        QSqlQueryModel* query_model = new QSqlQueryModel(this);

        query_model->setQuery(QString("delete from %1 where id =
            \'%2\'").arg(
                dictionary_name,
                helper::data(ui->dictionary_table2-
                    >currentIndex(),0).toString()));
        helper::reloadModel(ui->dictionary_table2->model());
        qDebug() << query_model->lastError();
        // ui->dictionary_table2->setModel(query_model);
    }
}

administratorform.h
#ifndef ADMINISTRATORFORM_H
#define ADMINISTRATORFORM_H

#include <QWidget>

namespace Ui {
class AdministratorForm;
}

class AdministratorForm : public QWidget
{
    Q_OBJECT

public:
    explicit AdministratorForm(QWidget *parent = nullptr);
    ~AdministratorForm();

private slots:
    void on_find_owner_b1_clicked();
    void on_add_owner_b1_clicked();
    void on_delete_owner_b1_clicked();
    void on_owners_table1_clicked(const QModelIndex &index);
    void on_find_worker_b1_clicked();
    void on_add_worker_b1_clicked();
    void on_delete_worker_b1_clicked();
    void on_add_worker_b1_2_clicked();
    void on_find_b2_clicked();
    void on_add_b2_clicked();
    void on_current_dictionary_cb2_activated(int index);
    void on_delete_b2_clicked();

private:
    Ui::AdministratorForm *ui;
};

#endif // ADMINISTRATORFORM_H

authorization.cpp
#include "authorization.h"
#include "ui_authorization.h"

Authorization::Authorization(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Authorization)
{
    ui->setupUi(this);
}

Authorization::~Authorization()
{
    delete ui;
}

void Authorization::ClearForm()
{
    ui->login_le->clear();
    ui->password_le->clear();
}

void Authorization::on_login_button_clicked()
{
    emit TryToLogin(ui->login_le->text(), ui->password_le->text());
}

void Authorization::on_exit_button_clicked()
{
    exit(0);
}

authorization.h
#ifndef AUTHORIZATION_H
#define AUTHORIZATION_H

#include <QWidget>

namespace Ui {
class Authorization;
}

class Authorization : public QWidget
{
    Q_OBJECT

public:
    explicit Authorization(QWidget *parent = nullptr);
    ~Authorization();
    void ClearForm();

signals:
    void TryToLogin(QString login, QString password);

private slots:
    void on_login_button_clicked();
    void on_exit_button_clicked();

private:
    Ui::Authorization *ui;
};

#endif // AUTHORIZATION_H

helper.cpp
#pragma once
#include "helper.hpp"
#include <QtSql>
#include <QHeaderView>
#include <QMessageBox>

namespace helper {
void debugQuery(QSqlQuery& query) {
    if (!query.exec()) {
        qDebug() << "Query error:";
        qDebug() << "Last query: " << query.lastQuery();
        qDebug() << "Error: " << query.lastError().text();
        return;
    }

    auto record = query.record();
    auto record_size = record.count();
    while(query.next()) {
        for (int i = 0; i < record_size; ++i) {
            qDebug().nospace() << record.field(i).name() << ": " <<
                query.value(i);
            qDebug().noquote() << "\n";
        }
        qDebug() << "Last query: " << query.lastQuery();
    }

    void setupTableView(QTableView* table_view) {
        table_view->horizontalHeader()->setStretchLastSection(true);
        table_view->setSelectionMode(QTableView::SingleSelection);
        table_view->setSelectionBehavior(QTableView::SelectRows);
        // table_view->verticalHeader()->hide();
    }

    QVariant data(const QModelIndex& index, int column_number) {
        auto result_index = index.model()->index(index.row(), column_number);
        auto result_data = result_index.data();
        return result_data;
    }

    void reloadModel(QAbstractItemModel* model) {
        auto query_model = dynamic_cast<QSqlQueryModel*>(model);
        assert(query_model);
        query_model->setQuery(query_model->query().lastQuery());
    }

    void reloadModel(QTableView* view) {
        reloadModel(view->model());
    }

    QSqlQuery sendQuery(const QString& query) {
        QSqlQuery sql_api;
        sql_api.prepare(query);
        if (!sql_api.exec()) {
            QMessageBox::warning(nullptr, "Ошибка",
                sql_api.lastError().text());
        }
        return sql_api;
    }
}

helper.hpp
#ifndef HELPER_HPP
#define HELPER_HPP
#include <QSqlQuery>
#include <QTableView>

namespace helper {
void debugQuery(QSqlQuery& query);
void setupTableView(QTableView* table_view);
QVariant data(const QModelIndex& index, int column_number);
void reloadModel(QAbstractItemModel* model);
void reloadModel(QTableView* view);
QSqlQuery sendQuery(const QString& query);
}

#endif // HELPER_HPP

holderform.cpp
#include "holderform.h"
#include "ui_holderform.h"
#include "helper.hpp"
#include <QDebug>
#include <QtSql>
#include <QVariant>
#include <QMessageBox>
#include <QtCharts>
#include "xlsxdocument.h"

```

```

#include "xlsxchartsheet.h"
#include "xlsxcellrange.h"
#include "xlsxchart.h"
#include "xlsxrichstring.h"
#include "xlsxworkbook.h"

void setupQuery_cb2(Ui::HolderForm* ui);

struct query_data{
    bool have_arguments;
    QString query;
};

struct table_data{
    QString name;
    QString column_name;
    QString label_text;
};

Q_DECLARE_METATYPE(query_data);
Q_DECLARE_METATYPE(table_data);

HolderForm::HolderForm(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::HolderForm)
{
    ui->setupUi(this);

    helper::setupTableView(ui->query_table2);
    helper::setupTableView(ui->tables_table2);

    setupQuery_cb2(ui);

    QVariant v;
    v.setValue(table_data{"products_view", "product_name", "Название
продукта:"});
    ui->current_table_cb2->addItem("Товары", v);
    v.setValue(table_data{"customers_view", "second_name", "Фамилия:"});
    ui->current_table_cb2->addItem("Заказчики", v);
    v.setValue(table_data{"realization_view", "product_name",
        "Название товара"});
    ui->current_table_cb2->addItem("Реализация", v);
    v.setValue(table_data{"workers_view", "login", "Логин:"});
    ui->current_table_cb2->addItem("Работники", v);

    int index = ui->current_table_cb2->currentIndex();
    QString table_name = qvariant_cast<table_data>
        (ui->current_table_cb2->itemData(index)).name;
    QString column_name = qvariant_cast<table_data>
        (ui->current_table_cb2->itemData(index)).column_name;

    QSqlQueryModel* query_model = new QSqlQueryModel(this);

    query_model->setQuery(QString("select * from %1").arg(
        table_name));
    ui->tables_table2->setModel(query_model);
}

HolderForm::~HolderForm()
{
    delete ui;
}

void setupQuery_cb2(Ui::HolderForm* ui){
    QVariant v;
    v.setValue(query_data{true, "query_1_1"});
    ui->query_cb2->addItem("1.1 Вывод продуктов определенного типа (По
внешнему ключу)", v);
    v.setValue(query_data{true, "query_1_2"});
    ui->query_cb2->addItem("1.2 Вывод клиентов живущих в определенном
районе (По внешнему ключу)", v);
    v.setValue(query_data{true, "query_1_3"});
    ui->query_cb2->addItem("1.3 Пользователи, которые родились позже
указанной даты (По date)", v);
    v.setValue(query_data{true, "query_1_4"});
    ui->query_cb2->addItem("1.4 Продукты, которые были приняты после
указанной даты (По date)", v);
    v.setValue(query_data{false, "query_2_1"});
    ui->query_cb2->addItem("2.1 Информация о клиентах и состояниях сданных
ими товаров", v);
    v.setValue(query_data{false, "query_2_2"});
    ui->query_cb2->addItem("2.2 Информация о продуктах и их реализации",
v);
    v.setValue(query_data{false, "query_2_3"});
    ui->query_cb2->addItem("2.3 Информация о реализации товаров указанием
магазина", v);
    v.setValue(query_data{false, "query_3_1"});
    ui->query_cb2->addItem("3.1 Все продукты и их реализация", v);
    v.setValue(query_data{false, "query_4_1"});
    ui->query_cb2->addItem("4.1 Реализация продукта с указанием работника",
v);
    v.setValue(query_data{false, "query_5_1"});
    ui->query_cb2->addItem("5.1 Реализация и работник который принял", v);
    v.setValue(query_data{false, "query_6_1"});
    ui->query_cb2->addItem("6.1 Проданные товары с магазином", v);
    v.setValue(query_data{false, "query_7_1"});
    ui->query_cb2->addItem("7.1 Количество товаров и проданных товаров",
v);
    v.setValue(query_data{true, "query_8_1"});
    ui->query_cb2->addItem("8.1 Продукты, которых на складе больше
указанного", v);
    v.setValue(query_data{true, "query_8_2"});
    ui->query_cb2->addItem("8.2 Поиск клиента по части номера общее
количество его товаров", v);
    v.setValue(query_data{true, "query_8_3"});
    ui->query_cb2->addItem("8.3 Количество чеков выписанных, на этот
товар", v);
    v.setValue(query_data{true, "query_8_4"});
    ui->query_cb2->addItem("8.4 Количество работников магазина", v);
    v.setValue(query_data{true, "query_9_1"});
    ui->query_cb2->addItem("9.1 Продукты, продажи которых больше
указанного", v);
    v.setValue(query_data{true, "query_10_1"});
    ui->query_cb2->addItem("10.1 Пользователи с указанным социальным
статусом и количеством сданных товаров больше указанного", v);
    v.setValue(query_data{false, "query_11_1"});
    ui->query_cb2->addItem("11.1 Средняя стоимость всех товаров магазина",
v);
    v.setValue(query_data{false, "query_12_1"});
    ui->query_cb2->addItem("12.1 Продукты которые еще не покупали и
продукты, которые полностью выкуплены", v);
    v.setValue(query_data{false, "query_13_1"});
    ui->query_cb2->addItem("13.1 Информация о пользователях, товары которых
еще есть в наличии", v);
    v.setValue(query_data{false, "query_13_2"});
    ui->query_cb2->addItem("13.2 Информация о пользователях, товаров
которых уже нет в наличии", v);
    v.setValue(query_data{false, "query_13_3"});
    ui->query_cb2->addItem("13.3 Популярность продукта у покупателей", v);
    v.setValue(query_data{false, "query_13_4"});
    ui->query_cb2->addItem("13.4 Продукты со стоимостью больше средней",
v);
}

void HolderForm::on_query_cb2_currentIndexChanged(int index)
{
    bool have_arguments = qvariant_cast<query_data>
        (ui->query_cb2->itemData(index)).have_arguments;
    QString name = qvariant_cast<query_data>
        (ui->query_cb2->itemData(index)).query;

    ui->argument_le2->setEnabled(have_arguments);
    if (name == "query_10_1"){
        ui->argument2_le2->setEnabled(true);
    } else {
        ui->argument2_le2->setEnabled(false);
    }
}

void HolderForm::on_query_b2_clicked()
{
    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    int index = ui->query_cb2->currentIndex();
    bool have_arguments = qvariant_cast<query_data>
        (ui->query_cb2->itemData(index)).have_arguments;
    QString name = qvariant_cast<query_data>
        (ui->query_cb2->itemData(index)).query;

    if(have_arguments){
        QSqlQuery query;
        if (name == "query_10_1"){
            query.prepare(QString("select * from %1(\\'%2\\', %3)").arg(
                name,
                ui->argument_le2->text(),
                ui->argument2_le2->text()
            ));
        } else {
            query.prepare(QString("select * from %1(\\'%2\\')").arg(
                name,
                ui->argument_le2->text()
            ));
        }

        if(!query.exec()){
            QMessageBox::warning(nullptr, "Ошибка", "Введите параметр,
подходящий под запрос");
        }

        query_model->setQuery(query);
        ui->query_table2->setModel(query_model);

        qDebug() << "Function " << name;

    } else {
        query_model->setQuery(QString("select * from %1").arg(name));
        qDebug() << "View" << name;
        ui->query_table2->setModel(query_model);
    }
}

void HolderForm::on_find_b2_clicked()
{
    int index = ui->current_table_cb2->currentIndex();
    QString table_name = qvariant_cast<table_data>
        (ui->current_table_cb2->itemData(index)).name;
    QString column_name = qvariant_cast<table_data>
        (ui->current_table_cb2->itemData(index)).column_name;

    QSqlQueryModel* query_model = new QSqlQueryModel(this);

    query_model->setQuery(QString("select * from %1 where %2 like
\\'%%3%'").arg(
        table_name,
        column_name,
        ui->name_le2->text()));
    ui->tables_table2->setModel(query_model);
}

void HolderForm::on_current_table_cb2_activated(int index)
{
    QString table_name = qvariant_cast<table_data>
        (ui->current_table_cb2->itemData(index)).name;
    QString label_text = qvariant_cast<table_data>
        (ui->current_table_cb2->itemData(index)).label_text;

    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    ui->label->setText(label_text);
    query_model->setQuery(QString("select * from %1").arg(
        table_name));
    ui->tables_table2->setModel(query_model);
}

```



```

void HolderForm::on_chart_button_clicked()
{
    auto series = new QPieSeries();
    auto cursor = helper::sendQuery("select * from test_view");
    while (cursor.next()) {
        series->append(cursor.value(0).toString(),
            cursor.value(1).toInt());
    }
    series->setLabelsVisible(true);
    auto chart = new QChart;
    chart->addSeries(series);
    chart->setAnimationOptions(QChart::AllAnimations);
    chart->setTitle("Количество проданных продуктов разных типов");
    // chart->legend()->setAlignment(Qt::AlignRight);
    chart->legend()->hide();
    ui->chart->setChart(chart);

    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery("select * from test_view");
    ui->view_table3->setModel(query_model);
    ui->chart->setRenderHint(QPainter::Antialiasing);
}

void HolderForm::on_import_button_clicked()
{
    QXlsx::Document xlsxw;

    auto query = helper::sendQuery("select * from test_view");
    auto record = query.record();
    auto record_size = record.count();
    for (int i = 1; i <= record_size; ++i) {
        xlsxw.write(1, i, record.fieldName(i - 1));
    }
    for (int i = 2; query.next(); ++i) {
        for (int j = 1; j <= record_size; ++j) {
            xlsxw.write(i, j, query.value(j - 1));
        }
    }
    const auto path = QFileDialog::getSaveFileName(nullptr, "Выберите место
    созранения", "", "Excel (*.xlsx)");
    xlsxw.saveAs(path);
}

holderform.h
#ifndef HOLDERFORM_H
#define HOLDERFORM_H

#include <QWidget>

namespace Ui {
class HolderForm;
}

class HolderForm : public QWidget
{
    Q_OBJECT

public:
    explicit HolderForm(QWidget *parent = nullptr);
    ~HolderForm();

private slots:
    void on_query_cb2_currentIndexChanged(int index);

    void on_query_b2_clicked();

    void on_find_b2_clicked();

    void on_current_table_cb2_activated(int index);

    void on_chart_button_clicked();

    void on_import_button_clicked();

private:
    Ui::HolderForm *ui;
};

#endif // HOLDERFORM_H

main.cpp
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}

mainwindow.cpp
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "authorization.h"
#include "workerform.h"
#include "administratorform.h"
#include "holderform.h"
#include <QDebug>
#include <QMessageBox>

#include <QtSql>

MainWindow::MainWindow(QWidget *parent)
: QMainWindow(parent)
, ui(new Ui::MainWindow) {
    ui->setupUi(this);
    auto auth = new Authorization(this);
    SetPage(auth);
    setGeometry(300, 200, 218, 99+40);

    connect(auth, &Authorization::TryToLogin, this, [this](const QString&
    login,
        const QString& password){
        ConnectDB(login, password);
        QSqlQuery q;
        q.exec("select current_user");
        q.first();
        qDebug() << q.value(0).toString();
        if (CheckUserGroup("worker")){
            SetPage(new WorkerForm);
        } else if (CheckUserGroup("administrator")) {
            SetPage(new AdministratorForm);
        } else if (CheckUserGroup("holder")){
            SetPage(new HolderForm);
        }

        // SetPage(new Formtest);
    });
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::SetPage(QWidget *w){
    resize(w->width()+18,w->height()+60);
    ui->stackedWidget->addWidget(w);
    ui->stackedWidget->setCurrentWidget(w);
    qDebug() << ui->stackedWidget->currentWidget()->objectName();
}

bool MainWindow::PreviousPage(){
    int last = ui->stackedWidget->count() - 1;
    auto* last_widget = ui->stackedWidget->widget(last);
    ui->stackedWidget->removeWidget(last_widget);
    return last;
}

void MainWindow::ConnectDB(QString login, QString password){
    database = QSqlDatabase::addDatabase("QPSQL");
    database.setDatabaseName("store");
    database.setHostName("localhost");
    database.setPassword(password);
    database.setPort(5432);
    database.setUserName(login);
    if (database.open()) {
        qDebug() << "Database have been open";
    } else {
        QMessageBox::warning(nullptr, "Ошибка", "Неверный логин или
        пароль");
        qDebug() << "Database have not been open";
        qDebug() << database.lastError();
    }
}

bool MainWindow::CheckUserGroup(const QString& role){
    QSqlQuery query;
    query.prepare("select pg_has_role(current_user, :role, 'USAGE')");
    query.bindValue(":role", role);
    query.exec();
    query.first();
    return query.value(0).toBool();
}

void MainWindow::on_stackedWidget_currentChanged(int)
{
}

void MainWindow::on_logout_triggered()
{
    database.close();
    QSize sz(218, 99+40);
    for (int i = 1; i < ui->stackedWidget->count(); i++) {
        auto* widget = ui->stackedWidget->widget(i);
        ui->stackedWidget->removeWidget(widget);
        delete widget;
    }
    auto* authorization = ui->stackedWidget->widget(0);
    auto* authorization_widget =
    dynamic_cast<Authorization*>(authorization);
    assert(authorization_widget);
    authorization_widget->ClearForm();

    resize(sz);
}

mainwindow.h
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QStackedWidget>
#include <QSqlDatabase>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_stackedWidget_currentChanged(int arg1);

```

```

void on_logout_triggered();

private:
    Ui::MainWindow *ui;
    QSqlDatabase database;
    void SetPage(QWidget *w);
    bool PreviousPage();
    void ConnectDB(QString login, QString password);
    bool CheckUserGroup(const QString& role);
};
#endif // MAINWINDOW_H

workerform.cpp
#include "workerform.h"
#include "ui_workerform.h"
#include <QSqlQuery>
#include <QSql>
#include <QSqlQueryModel>
#include <QDebug>
#include <QSqlError>
#include "helper.hpp"
#include <QMessageBox>

WorkerForm::WorkerForm(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::WorkerForm)
{
    ui->setupUi(this);
    helper::setupTableView(ui->realizations_table2);
    helper::setupTableView(ui->products_table_view);
    helper::setupTableView(ui->products_table2);
    helper::setupTableView(ui->products_table3);
    helper::setupTableView(ui->clients_table3);

    auto query = helper::sendQuery("select * from districts");
    while (query.next()) {
        ui->district_cb3->addItem(query.value(1).toString(),
            query.value(0));
    }

    query = helper::sendQuery("select * from statuses");
    while (query.next()) {
        ui->status_cb3->addItem(query.value(1).toString(), query.value(0));
    }

    query = helper::sendQuery("select * from product_types");
    while (query.next()) {
        ui->product_type_cb1->addItem(query.value(1).toString(),
            query.value(0));
    }

    // Setup products_view_table
    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery("select * from products_modify_view where
remaining > 0");
    ui->products_table_view->setModel(query_model);
    ui->sell_product_button->setEnabled(false);
    ui->realization_count_spin1->setEnabled(false);
    ui->products_table_view->hideColumn(0);
    ui->products_table_view->hideColumn(6);
    ui->products_table_view->hideColumn(7);
    ui->products_table_view->hideColumn(8);
    ui->products_table_view->hideColumn(9);
    ui->products_table_view->hideColumn(10);

    // Setup client_table3
    auto client_q = QString("select c.id, c.first_name, c.second_name,
c.phone, c.work "
"from customers c "
"where first_name like '%"%'%"%' "
"and second_name like '%"%'%"%' "
"group by c.id order by c.id desc").arg(ui->first_name_search_le3->text(),
ui->second_name_search_le3->text());
    QSqlQueryModel* client_qm = new QSqlQueryModel(this);
    client_qm->setQuery(client_q);
    ui->clients_table3->setModel(client_qm);

    ui->phone_add_le3->setValidator(new QRegExpValidator(QRegExp("[0-9]{1,10}"), this));
}

WorkerForm::~WorkerForm()
{
    delete ui;
}

void WorkerForm::SelectCustomer(int id_customer)
{
    QSqlQuery query;
    query.prepare("select * from customers_view where id = :id_customer");
    query.bindValue(":id_customer", id_customer);
    query.exec();
    query.next();
    qDebug() << query.lastError();

    ui->first_name_label->setText(query.value("first_name").toString());
    ui->second_name_label->setText(query.value("second_name").toString());
    ui->district_label->setText(query.value("district").toString());
    ui->status_label->setText(query.value("status").toString());
    ui->work_label->setText(query.value("work").toString());
    ui->birthday_label->setText(query.value("birthday").toString());
    ui->phone_label->setText(query.value("phone").toString());
}

int WorkerForm::SelectProduct(int quittance)
{
    QSqlQuery query;
    query.prepare("select * from products where id = :quittance");
    query.bindValue(":quittance", quittance);
    query.exec();
    query.next();
    ui->reception_label->setText(query.value("reception").toString());
    ui->count_label->setText(query.value("count").toString());
    return query.value("id_customer").toInt();
}

```

```

}

void WorkerForm::on_select_products_button_clicked()
{
    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery("select * from products_modify_view where
remaining > 0");
    ui->products_table_view->setModel(query_model);
    ui->sell_product_button->setEnabled(false);
    ui->realization_count_spin1->setEnabled(false);
    ui->products_table_view->hideColumn(0);
    ui->products_table_view->hideColumn(6);
    ui->products_table_view->hideColumn(7);
    ui->products_table_view->hideColumn(8);
    ui->products_table_view->hideColumn(9);
    ui->products_table_view->hideColumn(10);
    qDebug() << query_model->lastError();
}

void WorkerForm::on_products_table_view_clicked(const QModelIndex &index)
{
    auto model = index.model()->index(index.row(), 0);
    int customer_id = SelectProduct(model.data().toInt());
    SelectCustomer(customer_id);
    model = index.model()->index(index.row(), 4);
    ui->realization_count_spin1->setMaximum(model.data().toInt());
    ui->sell_product_button->setEnabled(true);
    ui->realization_count_spin1->setEnabled(true);
}

void WorkerForm::on_search_button_clicked()
{
    auto query = QString("select * from products_modify_view "
"where product_name like '%"%'%"%' and remaining > 0").arg(ui-
>product_name_le->text());

    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery(query);
    ui->products_table_view->setModel(query_model);
    ui->sell_product_button->setEnabled(false);
    ui->realization_count_spin1->setEnabled(false);
    ui->products_table_view->hideColumn(0);
    ui->products_table_view->hideColumn(6);
    ui->products_table_view->hideColumn(7);
    ui->products_table_view->hideColumn(8);
    ui->products_table_view->hideColumn(9);
    ui->products_table_view->hideColumn(10);
}

void WorkerForm::on_sell_product_button_clicked()
{
    QSqlQuery query;
    QString query_str = QString("select sell_product(%1, %2)").arg(
        helper::data(ui->products_table_view->currentIndex(),
0).toString(),
        ui->realization_count_spin1->text());
    query.exec(query_str);
    helper::reloadModel(ui->products_table_view->model());
    ui->realization_count_spin1->setEnabled(false);
    ui->sell_product_button->setEnabled(false);
    // qDebug() << query.lastError();

    // qDebug() << helper::data(ui->products_table_view->currentIndex(), 0);
}

void WorkerForm::on_products_find_button2_clicked()
{
    auto query = QString("select * from products_modify_view "
"where product_name like '%"%'%"%'").arg(ui->product_name_le2->text());
    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery(query);
    ui->products_table2->setModel(query_model);
    ui->products_table2->hideColumn(0);
    ui->products_table2->hideColumn(6);
    ui->products_table2->hideColumn(7);
    ui->products_table2->hideColumn(8);
    ui->products_table2->hideColumn(9);
    ui->products_table2->hideColumn(10);
}

void WorkerForm::on_products_table2_clicked(const QModelIndex &index)
{
    auto model = index.model()->index(index.row(), 0);
    int quittance = model.data().toInt();
    auto query = QString("select r.ticket, p.product_name, p.quittance,
r.realization_date, r.realization_count "
"from realization r "
"inner join products p on r.id_product = p.id "
"where r.id_product = %1 "
"order by quittance").arg(quittance);
    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery(query);
    ui->realizations_table2->setModel(query_model);
}

void WorkerForm::on_realization_find_button2_clicked()
{
    auto query = QString("select r.ticket, p.product_name, p.quittance,
r.realization_date, r.realization_count "
"from realization r "
"inner join products p on r.id_product = p.id "
"where realization_count > %1 "
"order by quittance").arg(ui->realization_count_sb2->text());
    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery(query);
    ui->realizations_table2->setModel(query_model);
}

```

```

void WorkerForm::on_realizations_table2_clicked(const QModelIndex &index)
{
    auto model = index.model()->index(index.row(), 2);
    int quittance = model.data().toInt();
    auto query = QString("select * from products_modify_view "
    "where quittance = %1").arg(quittance);
    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery(query);
    ui->products_table2->setModel(query_model);
    ui->products_table2->hideColumn(0);
    ui->products_table2->hideColumn(6);
    ui->products_table2->hideColumn(7);
    ui->products_table2->hideColumn(8);
    ui->products_table2->hideColumn(9);
    ui->products_table2->hideColumn(10);
}

void WorkerForm::on_find_client_button3_clicked()
{
    auto query = QString("select c.id, c.first_name, c.second_name,
    c.phone, c.work "
    "from customers c "
    "inner join products p on c.id = p.id_customer where "
    "first_name like \'%1%\'' "
    "and second_name like \'%2%\'' "
    "order by c.id desc").arg(ui->first_name_search_le3->text(), ui-
    >second_name_search_le3->text());
    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery(query);
    ui->clients_table3->setModel(query_model);
    // ui->clients_table3->hideColumn(0);

    // auto query = QString("select c.id, c.first_name, c.second_name,
    c.phone, c.work "
    //"from customers c "
    //"where first_name like \'%1%\'' "
    //"and second_name like \'%2%\'' "
    //"group by c.id order by c.id desc").arg(ui->first_name_search_le3-
    >text(), ui->second_name_search_le3->text());
    // QSqlQueryModel* query_model = new QSqlQueryModel(this);
    // query_model->setQuery(query);
    // ui->clients_table3->setModel(query_model);
}

void WorkerForm::on_clients_table3_clicked(const QModelIndex &index)
{
    auto model = index.model()->index(index.row(), 0);
    int customer_id = model.data().toInt();
    auto query = QString("select * from products_modify_view "
    "where quittance in (select quittance from products where id_customer =
    %1)").arg(customer_id);
    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery(query);
    ui->products_table3->setModel(query_model);
    ui->products_table3->hideColumn(0);
    ui->products_table3->hideColumn(6);
    ui->products_table3->hideColumn(7);
    ui->products_table3->hideColumn(8);
    ui->products_table3->hideColumn(9);
    ui->products_table3->hideColumn(10);
}

void WorkerForm::on_find_products_button3_clicked()
{
    auto query = QString("select * from products_modify_view "
    "where product_name like \'%1%\''").arg(ui->product_name_le3->text());
    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery(query);
    ui->products_table3->setModel(query_model);
    ui->products_table3->hideColumn(0);
    ui->products_table3->hideColumn(6);
    ui->products_table3->hideColumn(7);
    ui->products_table3->hideColumn(8);
    ui->products_table3->hideColumn(9);
    ui->products_table3->hideColumn(10);
}

void WorkerForm::on_products_table3_clicked(const QModelIndex &index)
{
    auto model = index.model()->index(index.row(), 0);
    int quittance = model.data().toInt();
    auto query = QString("select c.id, c.first_name, c.second_name,
    c.phone, c.work "
    "from customers c "
    "inner join products p on c.id = p.id_customer "
    "where p.id_shop = (select id from shops) "
    "and p.quittance = %1 "
    "group by c.id").arg(quittance);
    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery(query);
    ui->clients_table3->setModel(query_model);
    // ui->clients_table3->hideColumn(0);
}

void WorkerForm::on_add_client_button3_clicked()
{
    auto query = QString("insert into customers (first_name, second_name,
    id_district, id_status, work, birthday, phone) "
    "VALUES (\'%1\', \'%2\', %3, %4, \'%5\', \'%6\', \'%7%\')").arg(ui-
    >first_name_add_le3->text(),
    ui->second_name_add_le3->text(),
    ui->id_district_cb3-
    >currentData().toString(),
    ui->status_cb3-
    >currentData().toString(),
    ui->work_add_le3->text(),
    ui->birthday_de3->text(),
    ui->phone_add_le3->text());

    QSqlQuery q(query);
    helper::reloadModel(ui->clients_table3);
    qDebug() << q.lastError();
    q.exec();
}

void WorkerForm::on_find_all_client_button3_clicked()
{
    auto query = QString("select c.id, c.first_name, c.second_name,
    c.phone, c.work "
    "from customers c "
    "where first_name like \'%1%\'' "
    "and second_name like \'%2%\'' "
    "group by c.id order by c.id desc").arg(ui->first_name_search_le3->text(),
    ui->second_name_search_le3->text());
    QSqlQueryModel* query_model = new QSqlQueryModel(this);
    query_model->setQuery(query);
    ui->clients_table3->setModel(query_model);
    // ui->clients_table3->hideColumn(0);
}

void WorkerForm::on_add_product_button_clicked()
{
    QSqlQuery query;
    QString query_str{
        "insert into products_modify_view(product_type, product_name, "
        "remaining, price, id_customer) "
        "values (\'%1\', \'%2\', %3, %4, %5)"};
    query.prepare(query_str.arg(ui->product_type_cb1->currentText(),
    ui->product_name_le1->text(),
    ui->count_sb1->text(),
    ui->price_sb1->text(),
    ui->id_customer_sb1->text()
    ));
    if (!query.exec()) {
        QMessageBox::warning(nullptr, "Ошибка", query.lastError().text());
        qDebug() << query.lastQuery();
        return;
    }
    ui->sell_product_button->setEnabled(false);
    ui->realization_count_spin1->setEnabled(false);
    helper::reloadModel(ui->products_table_view->model());
}

workerform.h
#ifndef WORKERFORM_H
#define WORKERFORM_H

#include <QWidget>

namespace Ui {
    class WorkerForm;
}

class WorkerForm : public QWidget
{
    Q_OBJECT

public:
    explicit WorkerForm(QWidget *parent = nullptr);
    ~WorkerForm();
    void SelectCustomer(int id_customer);
    int SelectProduct(int quittance);

private slots:
    void on_select_products_button_clicked();

    void on_products_table_view_clicked(const QModelIndex &index);

    void on_search_button_clicked();

    void on_sell_product_button_clicked();

    void on_products_find_button2_clicked();

    void on_products_table2_clicked(const QModelIndex &index);

    void on_realization_find_button2_clicked();

    void on_realizations_table2_clicked(const QModelIndex &index);

    void on_find_client_button3_clicked();

    void on_clients_table3_clicked(const QModelIndex &index);

    void on_find_products_button3_clicked();

    void on_products_table3_clicked(const QModelIndex &index);

    void on_add_client_button3_clicked();

    void on_find_all_client_button3_clicked();

    void on_add_product_button_clicked();

private:
    Ui::WorkerForm *ui;
};

#endif // WORKERFORM_H

```

ПРИЛОЖЕНИЕ В. ЛИСТИНГ СЕРВЕРНОГО ПРИЛОЖЕНИЯ

```

--
-- PostgreSQL database dump
--

-- Dumped from database version 13.2
-- Dumped by pg_dump version 13.2

-- Started on 2021-06-04 08:56:05

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;

--
-- TOC entry 763 (class 1247 OID 25129)
-- Name: phone_number; Type: DOMAIN; Schema: public; Owner: postgres
CREATE DOMAIN public.phone_number AS character(10)
    CONSTRAINT phone_number_check CHECK ((length(VALUE) = 10));

ALTER DOMAIN public.phone_number OWNER TO postgres;

--
-- TOC entry 788 (class 1247 OID 25257)
-- Name: positive_num; Type: DOMAIN; Schema: public; Owner: postgres
CREATE DOMAIN public.positive_num AS integer
    CONSTRAINT positive CHECK ((VALUE > 0));

ALTER DOMAIN public.positive_num OWNER TO postgres;

--
-- TOC entry 259 (class 1255 OID 25259)
-- Name: count_check(); Type: FUNCTION; Schema: public; Owner: postgres
CREATE FUNCTION public.count_check() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
begin
    if new.count < 0 then
        raise exception 'Нельзя столько продавать!';
    else
        return new;
    end if;
end;
$$;

ALTER FUNCTION public.count_check() OWNER TO postgres;

--
-- TOC entry 265 (class 1255 OID 16972)
-- Name: create_worker(text, text, integer); Type: FUNCTION; Schema: public; Owner: postgres
CREATE FUNCTION public.create_worker(username text, password
text, shop_id integer) RETURNS void
    LANGUAGE plpgsql SECURITY DEFINER
    AS $$
begin
    insert into workers values (default, username, shop_id,
true);

    execute format('create role %s login password ''%s'' in
role worker', username, password);
end;
$$;

ALTER FUNCTION public.create_worker(username text, password
text, shop_id integer) OWNER TO postgres;

--
-- TOC entry 238 (class 1255 OID 16841)
-- Name: decrease_products(); Type: FUNCTION; Schema: public; Owner: postgres
CREATE FUNCTION public.decrease_products() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
begin
    update products set remaining = (select remaining from
products where id = new.id_product)-new.realization_count
    where id = new.id_product;

    return new;
end;
$$;

ALTER FUNCTION public.decrease_products() OWNER TO postgres;

--
-- TOC entry 267 (class 1255 OID 16931)
-- Name: delete_from_all_tables(); Type: FUNCTION; Schema: public; Owner: postgres
CREATE FUNCTION public.delete_from_all_tables() RETURNS void
    LANGUAGE plpgsql
    AS $$
begin
    delete
    from customers
    where true;
    alter sequence customers_id_seq restart 1;

    delete
    from districts
    where true;
    alter sequence districts_id_seq restart 1;

    delete
    from statuses
    where true;
    alter sequence statuses_id_seq restart 1;

    delete
    from pg_roles
    where rolcanlogin = true
    and ARRAY [oid] && (select grolist from pg_group
where groname = 'worker');

    delete
    from workers
    where true;
    alter sequence workers_id_seq restart 1;

```

```

delete
from product_types
where true;
alter sequence product_types_id_seq restart 1;

delete
from shops
where true;
alter sequence shops_id_seq restart 1;

delete
from products
where true;
alter sequence products_id_seq restart 1;
alter sequence quittance_num restart 1;

delete
from realization
where true;
alter sequence ticket_num restart 1;
alter sequence realization_id_seq restart 1;

end;
$$;

ALTER FUNCTION public.delete_from_all_tables() OWNER TO
postgres;

--
-- TOC entry 262 (class 1255 OID 26303)
-- Name: delete_workers(); Type: FUNCTION; Schema: public;
Owner: postgres
--

CREATE FUNCTION public.delete_workers() RETURNS void
LANGUAGE plpgsql
AS $$
declare
count integer;
begin
count = (select count(*) from workers);

for i in 1..count
loop
execute format('drop role ' || (select login from
workers limit 1));

end loop;

end;
$$;

ALTER FUNCTION public.delete_workers() OWNER TO postgres;

--
-- TOC entry 242 (class 1255 OID 25229)
-- Name: fire_worker(character varying); Type: FUNCTION;
Schema: public; Owner: postgres
--

CREATE FUNCTION public.fire_worker(login_ character varying)
RETURNS void
LANGUAGE plpgsql SECURITY DEFINER
AS $$
begin
update workers set is_available = false where login =
login_;

execute format('drop role %s ', login_);
end;
$$;

ALTER FUNCTION public.fire_worker(login_ character varying)
OWNER TO postgres;

--
-- TOC entry 268 (class 1255 OID 16932)

-- Name: generate_all_tables(integer, integer, integer,
integer, integer); Type: FUNCTION; Schema: public; Owner: postgres
--

CREATE FUNCTION public.generate_all_tables(shops_count
integer, workers_count integer, products_count integer, customers_count
integer, realization_count integer) RETURNS void
LANGUAGE plpgsql
AS $$
begin
perform generate_statuses();
perform generate_districts();
perform generate_product_types();
perform generate_shops(shops_count);
perform generate_workers(workers_count);
perform generate_customers(customers_count);
perform generate_products(products_count);
perform generate_realization(realization_count);

end;
$$;

ALTER FUNCTION public.generate_all_tables(shops_count integer,
workers_count integer, products_count integer, customers_count integer,
realization_count integer) OWNER TO postgres;

--
-- TOC entry 239 (class 1255 OID 16787)
-- Name: generate_customers(integer); Type: FUNCTION; Schema:
public; Owner: postgres
--

CREATE FUNCTION public.generate_customers(count integer)
RETURNS void
LANGUAGE plpgsql
AS $$
declare
first_names varchar[];
second_names varchar[];
prefixes varchar[];
streets varchar[];
letters varchar[];
begin
first_names = array [ 'Августина', 'Авора', 'Агафья',
'Агнесса', 'Агния', 'Ада', 'Аделия', 'Аза',
'Азиза', 'Аида', 'Алана', 'Алевтина', 'Александра',
'Алико', 'Алина', 'Алиса', 'Алла', 'Альберта', 'Альбина', 'Альжбета',
'Амелия', 'Амина', 'Анастасия', 'Ангелина', 'Ангела,
Ангелика', 'Анисья', 'Анита', 'Анна', 'Антонина', 'Анфиса',
'Анэля', 'Ариадна', 'Арина', 'Архелия', 'Астра',
'Ася', 'Аурелия', 'Беатриса', 'Белла',
'Береслава', 'Берта', 'Биргит', 'Богдана', 'Боженa',
'Борислава', 'Бронислава', 'Валентина', 'Валерия', 'Ванда',
'Варвара', 'Василиса', 'Венера', 'Вера', 'Вероника',
'Веселина', 'Веста', 'Вета, Иветта', 'Вида, Видана', 'Виктория',
'Вилора', 'Виолетта', 'Виргиния', 'Виталина',
'Владислава', 'Галина', 'Гаянэ', 'Гелена', 'Гелла', 'Генриетта',
'Георгина', 'Гера', 'Гертруда', 'Глафира', 'Глория',
'Гражина', 'Грета', 'Гюзель', 'Дайна', 'Дана', 'Даниэла', 'Данута',
'Дарина', 'Дарья', 'Дебора', 'Джемма', 'Джулия',
'Джульетта', 'Диана', 'Дина', 'Динара', 'Диодора', 'Дионисия', 'Доля',
'Доминика', 'Ева', 'Евгения', 'Евдокия', 'Екатерина',
'Елена', 'Елизавета', 'Жанна', 'Зара', 'Земфира',
'Зинаида', 'Злата', 'Зоя', 'Иветта', 'Ивона',
'Изабелла', 'Изольда', 'Илзе', 'Инара', 'Инга', 'Инесса', 'Инна',
'Иоанна', 'Иоланта', 'Ираида', 'Ирина', 'Ольга',
'Сабина', 'Санта', 'Сарра', 'Светлана', 'Северина', 'Серафима',
'Сильва, Сильвия', 'Сима', 'Симона', 'Снежана',
'Софья', 'Станислава', 'Стелла', 'Стефания', 'Сусанна', 'Таира', 'Таисия',
'Тала', 'Тамара', 'Татьяна', 'Тереза', 'Томила',
'Ульяна', 'Юлиана', 'Устина', 'Фаиза', 'Фаина', 'Фаня', 'Фая',
'Фелиция', 'Флора', 'Франсуаза', 'Фрида', 'Хильда',
'Христина', 'Христя', 'Цветана', 'Чеслава', 'Эдда',
'Эдита', 'Элеонора', 'Элина, Элина', 'Элла',
'Эллада', 'Элоиза', 'Эльвира', 'Эльга', 'Эльза', 'Эльмира', 'Эмилия',
'Эмма',
'Эрика', 'Эсмеральда', 'Юзефа', 'Юлия', 'Юна',
'Юнона', 'Юстина', 'Ядвига', 'Яна', 'Янита', 'Янка', 'Ярослава',

```

```

        'Абрам', 'Аваз', 'Август', 'Авдей', 'Автандил',
        'Адам', 'Адис', 'Адолф', 'Адриан', 'Азарий', 'Аким', 'Алан',
        'Александр', 'Алексей', 'Альберт', 'Альфред',
        'Амадей', 'Амадеус', 'Амаяк', 'Анатолий', 'Ангел', 'Андоим', 'Андрей',
        'Аникита', 'Антон', 'Ануфрий', 'Арам', 'Арий',
        'Аристарх', 'Аркадий', 'Арно', 'Арнольд', 'Арон', 'Арсений',
        'Артем', 'Артемий', 'Артур', 'Архип', 'Аскольд',
        'Афанасий', 'Ахмет', 'Ашот', 'Бежен', 'Бенедикт', 'Берек',
        'Бернар', 'Богдан', 'Боголюб', 'Болеслав',
        'Бонифаций', 'Борис', 'Борислав', 'Боян', 'Бронислав', 'Бруно', 'Вадим',
        'Валентин', 'Валерий', 'Вальтер', 'Василий',
        'Велизар', 'Венедикт', 'Вениамин', 'Виктор', 'Вилен', 'Вилли', 'Вильгельм',
        'Виссарион', 'Виталий', 'Витаутас', 'Витольд',
        'Владимир', 'Владислав', 'Владлен', 'Володар', 'Вольдемар', 'Всеволод',
        'Вячеслав', 'Г', 'Гавриил', 'Гарри', 'Гастон',
        'Геннадий', 'Генрих', 'Георгий', 'Геральд', 'Герасим', 'Герман', 'Глеб',
        'Гордей', 'Гордон', 'Градимир', 'Григорий', 'Гурий',
        'Давид', 'Даниил', 'Демид', 'Демьян',
        'Денис', 'Джордан', 'Дмитрий', 'Дональд', 'Донат',
        'Донатос', 'Дорофей', 'Евгений', 'Евграф', 'Евдоким', 'Евстафий',
        'Егор', 'Елизар', 'Елисей', 'Емельян', 'Ермолай',
        'Ерофей', 'Ефим', 'Ефимий', 'Ефрем', 'Жан', 'Ждан', 'Жорж',
        'Захар', 'Захария', 'Зигмунд', 'Зиновий', 'Ибрагим',
        'Иван', 'Игнат', 'Игорь', 'Измаил', 'Израиль', 'Илиан',
        'Илларион', 'Илья', 'Иннокентий', 'Ион', 'Ионос',
        'Иосиф', 'Ираклий', 'Иржи', 'Исай', 'Казимир', 'Карен', 'Карл',
        'Ким', 'Кирилл', 'Клавдий', 'Клемент', 'Клим', 'Клод',
        'Кондрат', 'Конкордий', 'Константин', 'Кузьма', 'Лазарь',
        'Лев', 'Леван', 'Леонард', 'Леонид', 'Леонтий',
        'Леопольд', 'Лука', 'Любомир', 'Людвиг', 'Люсьен', 'Мадлен', 'Май',
        'Макар', 'Максим', 'Максимилиан', 'Мануил', 'Марат',
        'Мариан', 'Марк', 'Мартин', 'Матвей', 'Мераб', 'Мечеслав',
        'Милан', 'Мирон', 'Мирослав', 'Михаил', 'Мичлов',
        'Модест', 'Моисей', 'Мурат', 'Муслим', 'Назар', 'Назарий', 'Натан',
        'Наум', 'Никита', 'Никифор', 'Николай', 'Никон',
        'Нисон', 'Нифонт', 'Олан', 'Олег', 'Олесь', 'Онисим', 'Орест', 'Осип',
        'Оскар', 'Павел', 'Парамон', 'Петр', 'Платон',
        'Порфирий', 'Прохор', 'Равиль', 'Радий', 'Радомир', 'Раис',
        'Раймонд', 'Ратмир', 'Рафаил', 'Рафик', 'Рашид',
        'Рем', 'Ренольд', 'Ринат', 'Рифат', 'Ричард', 'Роберт', 'Родион',
        'Ролан', 'Роман', 'Ростислав', 'Рубен', 'Рудольф',
        'Руслан', 'Рустам', 'Савва', 'Савел', 'Самсон', 'Святослав',
        'Севастьян', 'Северин', 'Семен', 'Серафим', 'Сергей',
        'Сократ', 'Соломон', 'Спартак', 'Стакрат', 'Станислав', 'Степан',
        'Стивен', 'Стоян', 'Таис', 'Талик', 'Тамаз', 'Тарас',
        'Тельман', 'Теодор', 'Терентий', 'Тибор', 'Тигран', 'Тигрий',
        'Тимофей', 'Тимур', 'Тит', 'Тихон', 'Трифон',
        'Трофим', 'Ульманас', 'Устин', 'Ф', 'Фаддей', 'Федор', 'Феликс',
        'Феодосий',
        'Фидель', 'Филимон', 'Филипп', 'Флорентий', 'Фома',
        'Франц', 'Фридрих', 'Харитон', 'Христиан', 'Христос', 'Христофор',
        'Эдвард', 'Эдуард', 'Эльдар', 'Эмиль', 'Эммануил',
        'Эраст', 'Эрик', 'Эрнест', 'Юлиан', 'Юрий', 'Юхим', 'Яким',
        'Яков', 'Ян', 'Яромир', 'Ярослав', 'Ясон'];
        second_names
        = array ['Барлоу', 'Баскервил',
        'Батчелор', 'Бетелл', 'Биддер', 'Бичем', 'Блэр', 'Бойер',
        'Ботт', 'Боттерилл', 'Бродерик',

```

```

        'Бромфилд', 'Брэгг', 'Брюстер', 'Бутби', 'Гатри', 'Грейвз', 'Грейнджер', 'Гудман',
        'Даблзэй', 'Даунинг', 'Додвелл', 'Дэнсон', 'Калвер',

```

```

        'Кантуэлл', 'Кларксон', 'Клауд', 'Клиленд', 'Клоуз', 'Кокрил', 'Коллингвуд', 'Копл
        енд', 'Косгроув', 'Коупленд', 'Коутс', 'Коуэлл', 'Кросби',

```

```

        'Кушинг', 'Кэмпион', 'Лампкин', 'Лейк', 'Листон', 'Лэнгтон', 'Макгуайр', 'Макдонне
        лл', 'Макдугал', 'Макдугалл', 'Маккей', 'Маккинни', 'Макклейн',

```

```

        'Маккормик', 'Маккэффри', 'Маклахлан', 'Маклохлин', 'Макмаллен', 'Макморрей', 'Ма
        куильямс', 'Макфи', 'Макыээн', 'Манселл', 'Мелтон', 'Мерриман',

```

```

        'Мерритт', 'Мерчант', 'Милнз', 'Мэнселл', 'Несбит', 'Несбитт', 'Ньюбери', 'Ньюберр
        и', 'Ньюболд', 'Ньюолл', 'Ньюэлл', 'Оллред', 'Орр', 'Ортон',

```

```

        'Осборн', 'Осмонд', 'Рэтлифф', 'Сарджент', 'Сеймур', 'Симм', 'Спарроу', 'Спунер', '
        Стэндинг', 'Стэнтон', 'Стэрди', 'Суинбёрн', 'Сэлмон',

```

```

        'Сэмюэлс', 'Таббс', 'Талли', 'Тейт', 'Тернбулл', 'Тобин', 'Толмен', 'Торнхилл', 'То
        рп', 'Трэверс', 'Трэшер', 'Турнер', 'Тэкери', 'Тэннер', 'Фарлоу',

```

```

        'Фейтфулл', 'Флек', 'Флетчер', 'Фрит', 'Фрэмpton', 'Шервуд', 'Эбби', 'Эгертон', 'Эд
        ди', 'Эдкок', 'Эйтчисон', 'Эмерсон', 'Эшби', 'Эшкрофт'];

```

```

        prefixes = array ['ул.', 'нр-т.', 'пер.'];
        streets
        = array
        ['Гринкевича', 'Гурова', 'Ильича', 'Артема', 'Байдукова', 'Батова', 'Баренца', 'Ва
        тугина', 'Зайцева', 'Ермошенко',

```

```

        'Горбатова', 'Ионова', 'Калинина', 'Кобозева', 'Ковалы', 'Конева', 'Куприна', 'Лаг
        утенко', 'Маяковского', 'Орешкова', 'Петровского', 'Прокофьева',

```

```

        'Ткаченко', 'Флеровского', 'Шекспира', 'Щорса'];
        letters = array ['', 'а', 'б', 'в', 'г', 'д'];

```

```

        for i in 1..count
        loop
            insert into customers (first_name, second_name,
            id_district, id_status, work, birthday, phone)
            values (first_names[trunc(random()
            * array_length(first_names, 1)) + 1],
            second_names[trunc(random()
            * array_length(second_names, 1)) + 1],
            (select id from districts limit 1 offset
            trunc(random()*(select count(*) from districts))),
            (select id from statuses limit 1 offset
            trunc(random()*(select count(*) from statuses))),

```

```

            prefixes[trunc(random()*array_length(prefixes,1))+1] ||
            streets[trunc(random()*array_length(streets,1))+1] || ' ' ||
            trunc(random() * 200) ||
            letters[trunc(random()*array_length(letters,1))+1],
            '01-01-2002'::date - trunc(random() *
            29220)::integer,
            '071' || trunc(random() * (9999999-
            1000000)+1000000));

```

```

        end loop;

    end ;
$$;

```

```

ALTER FUNCTION public.generate_customers(count integer) OWNER
TO postgres;

```

```

--
-- TOC entry 243 (class 1255 OID 16804)
-- Name: generate_districts(); Type: FUNCTION; Schema: public;
Owner: postgres
--

```

```

CREATE FUNCTION public.generate_districts() RETURNS void
LANGUAGE plpgsql
AS $$
begin
    insert into districts (district)
    values
    ('Киевский'),('Буденовский'),('Петровский'),('Куйбышевский'),
    ('Ворошиловский'),('Кировский'),('Калининский');
end;
$$;

```

```

ALTER FUNCTION public.generate_districts() OWNER TO postgres;

```

```

--
-- TOC entry 245 (class 1255 OID 16811)
-- Name: generate_product_types(); Type: FUNCTION; Schema:
public; Owner: postgres
--

```

```

CREATE FUNCTION public.generate_product_types() RETURNS void
LANGUAGE plpgsql
AS $$
begin
    insert into product_types (product_type)
    values ('посуда'),('одежда'),('обувь'),
    ('игрушки'),('товары для дома'),('товары для
    детей'),('инструменты'),

```

```

        ('компьютерная техника'),('электроника'));
    end;
    $$;

ALTER FUNCTION public.generate_product_types() OWNER TO
postgres;

--
-- TOC entry 269 (class 1255 OID 16820)
-- Name: generate_products(integer); Type: FUNCTION; Schema:
public; Owner: postgres
--

CREATE FUNCTION public.generate_products(count integer)
RETURNS void
    LANGUAGE plpgsql
    AS $$
    declare
        product_count integer;
        customer_id integer;
        part1 varchar[] = array ['Горячий', 'Прекрасный',
'Холодный', 'Крутой', 'Вонючий', 'Ароматный', 'Веселый', 'Грустный',
'Обаятельный',
        'Прикольный', 'Одаренный', 'Романтичный', 'Волевой',
'Призрачный', 'Дырявый', 'Рванный',
        'Железный', 'Деревянный', 'Золотой', 'Серебрянный',
'Позолоченный', 'Поддельный', 'Натуральный'];
        part2 varchar[] = array ['capor', 'диск', 'шлепок',
'утюг', 'компьютер', 'носок', 'галстук', 'робот', 'автомат',
        'приклад', 'блокнот', 'шнур', 'монитор', 'механизм',
'эспандер', 'провод', 'удлинитель',
        'переходник', 'аксессуар', 'штатив', 'микрофон',
'ремень', 'кроссовок', 'мяч', 'конструктор'];

        -- Совершеннолетие пользователя (товар можно сдавать
только с 18 лет)
        customer_majority date;
    begin
        for i in 1..count
            loop
                product_count = (random() ^ 5) * 100 + 1;
                customer_id = (select id from customers limit 1
offset trunc(random() * (select count(*) from customers)));
                customer_majority = (select birthday from
customers where id = customer_id) + 6575;
                insert into products(quittance, id_product_type,
product_name, reception, count, remaining, price,
                                id_customer, id_shop,
id_woker)
                VALUES ((select nextval('quittance_num')),
                (select id
                from product_types
                limit 1
                offset
                trunc(random() * (select count(*) from
product_types)))),
                part1[trunc(random() * array_length(part1,
1)) + 1] || ' ' ||
                part2[trunc(random() * array_length(part2,
1)) + 1] ||
                trunc(random() * 95 + 1) ,
                customer_majority + trunc(random() * ('10-
05-2021'::date - customer_majority))::integer,
                product_count,
                product_count,
                random() * 10000 + 100,
                customer_id,
                (select id from shops limit 1 offset
trunc(random() * (select count(*) from shops))),
                (select id from workers limit 1 offset
trunc(random() * (select count(*) from workers))));
            end loop;
        end;
    $$;

```

```

ALTER FUNCTION public.generate_products(count integer) OWNER
TO postgres;

--
-- TOC entry 266 (class 1255 OID 16847)
-- Name: generate_realization(integer); Type: FUNCTION;
Schema: public; Owner: postgres
--

CREATE FUNCTION public.generate_realization(count integer)
RETURNS integer
    LANGUAGE plpgsql
    AS $$
    declare
        real_count integer = 0;
        random_id_product integer;
        reception_date date;

    begin
        for i in 1..count
            loop
                random_id_product =
                (select id
                from products
                where remaining > 0
                limit 1
                offset
                trunc(random() * (select count(*) from
products where remaining > 0)));

                exit when random_id_product is null;
                reception_date = (select reception from products
where products.id = random_id_product);

                insert into realization (ticket, id_product,
realization_date, realization_count, id_worker)
                VALUES (nextval('ticket_num'),
                random_id_product,
                reception_date + trunc(random() * ('10-05-
2021'::date - reception_date))::integer,
                (trunc(random() * (select remaining from
products where products.id = random_id_product)) + 1),
                (select id from workers limit 1 offset
trunc(random() * (select count(*) from workers))));
                real_count = real_count + 1;
            end loop;
        return real_count;
    end;
    $$;

ALTER FUNCTION public.generate_realization(count integer)
OWNER TO postgres;

--
-- TOC entry 246 (class 1255 OID 16817)
-- Name: generate_shops(integer); Type: FUNCTION; Schema:
public; Owner: postgres
--

CREATE FUNCTION public.generate_shops(count integer) RETURNS
void
    LANGUAGE plpgsql
    AS $$
    begin
        for i in 1..count
            loop
                insert into shops(name) values ('Комиссионный №'
|| trunc(random()*1000000));
            end loop;
        end;
    $$;

ALTER FUNCTION public.generate_shops(count integer) OWNER TO
postgres;

--

```

```

-- TOC entry 244 (class 1255 OID 16806)
-- Name: generate_statuses(); Type: FUNCTION; Schema: public;
Owner: postgres
--

CREATE FUNCTION public.generate_statuses() RETURNS void
LANGUAGE plpgsql
AS $$
begin
    insert into statuses (status)
    values
('пенсионер'),('предприниматель'),('домохозяйка'),
('госслужащий'),('студент'),('безработный');
end;
$$;

ALTER FUNCTION public.generate_statuses() OWNER TO postgres;

--
-- TOC entry 264 (class 1255 OID 16808)
-- Name: generate_workers(integer); Type: FUNCTION; Schema:
public; Owner: postgres
--

CREATE FUNCTION public.generate_workers(count integer) RETURNS
void
LANGUAGE plpgsql
AS $$
begin
    for i in 1..count
    loop
        perform create_worker('worker' || i, 'qwe',
            (select id from shops limit
1 offset trunc(random() * (select count(*) from shops))));
    end loop;
end;
$$;

ALTER FUNCTION public.generate_workers(count integer) OWNER TO
postgres;

--
-- TOC entry 260 (class 1255 OID 25126)
-- Name: instead_of_insert(); Type: FUNCTION; Schema: public;
Owner: postgres
--

CREATE FUNCTION public.instead_of_insert() RETURNS trigger
LANGUAGE plpgsql SECURITY DEFINER
AS $$
begin
    insert into products (quittance, id_product_type,
product_name, reception, count, remaining, price, id_customer,
id_shop, id_woker)
values (nextval('quittance_num'), (select id from
product_types where product_type = new.product_type),
new.product_name, now()::date, new.remaining,
new.remaining, new.price,
new.id_customer, (select workers.id_shop from
workers where login = session_user),
(select id from workers where login =
session_user));
    return new;
end;
$$;

ALTER FUNCTION public.instead_of_insert() OWNER TO postgres;

--
-- TOC entry 263 (class 1255 OID 25262)
-- Name: is_available_check(); Type: FUNCTION; Schema: public;
Owner: postgres
--

CREATE FUNCTION public.is_available_check() RETURNS trigger
LANGUAGE plpgsql

```

```

AS $$
begin
    if old.is_available = false then return old; else return
new;
    end if;
end;
$$;

ALTER FUNCTION public.is_available_check() OWNER TO postgres;

--
-- TOC entry 277 (class 1255 OID 17149)
-- Name: query_10_1(character varying, integer); Type:
FUNCTION; Schema: public; Owner: postgres
--

CREATE FUNCTION public.query_10_1(status1 character varying,
prod_count integer) RETURNS TABLE(customer character varying, shop
character varying, status character varying, products_count integer)
LANGUAGE plpgsql SECURITY DEFINER
AS $$
begin
    return query select t1.customer, t1.shop, t1.status,
t1.products_count
        from (select (c.first_name || ' ' ||
c.second_name) :: varchar as customer,
s.name
as shop,
s2.status,
count(p.quittance)::integer
as products_count
        from products p
        inner join customers c on
p.id_customer = c.id
        inner join shops s on
p.id_shop = s.id
        inner join statuses s2 on
c.id_status = s2.id
        where s2.status = status1
        group by p.id_customer, customer, shop,
s2.status
        having count(p.quittance) > prod_count
        order by count(p.quittance) desc) as
t1;
end;
$$;

ALTER FUNCTION public.query_10_1(status1 character varying,
prod_count integer) OWNER TO postgres;

--
-- TOC entry 270 (class 1255 OID 17055)
-- Name: query_1_1(character varying); Type: FUNCTION; Schema:
public; Owner: postgres
--

CREATE FUNCTION public.query_1_1(product_t character varying)
RETURNS TABLE(id integer, product_name character varying, reception date,
remaining integer, price integer, customer_firs_name character varying,
customer_second_name character varying, shop character varying,
product_type character varying)
LANGUAGE plpgsql SECURITY DEFINER
AS $$
begin
    return query select products.id,
products.product_name,
products.reception,
products.remaining,
products.price,
c.first_name,
c.second_name,
s.name,
pt.product_type
from products
        inner join customers c on
products.id_customer = c.id

```



```

                                inner join product_types pt on
products.id_product_type = pt.id
                                inner join shops s on
products.id_shop = s.id
                                where id_product_type =
                                (select product_types.id from
product_types where product_types.product_type = product_t);
end;
$$;

ALTER FUNCTION public.query_1_1(product_t character varying)
OWNER TO postgres;

--
-- TOC entry 271 (class 1255 OID 17057)
-- Name: query_1_2(character varying); Type: FUNCTION; Schema:
public; Owner: postgres
--

CREATE FUNCTION public.query_1_2(district_name character
varying) RETURNS TABLE(id integer, first_name character varying,
second_name character varying, district character varying, work character
varying, birthday date, phone character)
LANGUAGE plpgsql SECURITY DEFINER
AS $$
begin
return query select customers.id,
customers.first_name,
customers.second_name,
d.district,
customers.work,
customers.birthday,
customers.phone
from customers
inner join districts d on
customers.id_district = d.id
where id_district = (select districts.id from
districts where districts.district = district_name);
end;
$$;

ALTER FUNCTION public.query_1_2(district_name character
varying) OWNER TO postgres;

--
-- TOC entry 272 (class 1255 OID 17059)
-- Name: query_1_3(date); Type: FUNCTION; Schema: public;
Owner: postgres
--

CREATE FUNCTION public.query_1_3(birth date) RETURNS TABLE(id
integer, first_name character varying, second_name character varying,
district character varying, work character varying, birthday date, phone
character)
LANGUAGE plpgsql SECURITY DEFINER
AS $$
begin
return query select customers.id,
customers.first_name,
customers.second_name,
d.district,
customers.work,
customers.birthday,
customers.phone
from customers
inner join districts d on
customers.id_district = d.id
where customers.birthday > birth;
end;
$$;

ALTER FUNCTION public.query_1_3(birth date) OWNER TO postgres;

--
-- TOC entry 274 (class 1255 OID 17060)

```

```

-- Name: query_1_4(date); Type: FUNCTION; Schema: public;
Owner: postgres
--

CREATE FUNCTION public.query_1_4(reception_date date) RETURNS
TABLE(id integer, product_name character varying, reception date,
remaining integer, price integer, customer_first_name character varying,
customer_second_name character varying, shop character varying,
product_type character varying)
LANGUAGE plpgsql SECURITY DEFINER
AS $$
begin
return query select products.id,
products.product_name,
products.reception,
products.remaining,
products.price,
c.first_name,
c.second_name,
s.name,
pt.product_type
from products
inner join customers c on
products.id_customer = c.id
inner join product_types pt on
products.id_product_type = pt.id
inner join shops s on
products.id_shop = s.id
where reception > reception_date;
end;
$$;

ALTER FUNCTION public.query_1_4(reception_date date) OWNER TO
postgres;

--
-- TOC entry 275 (class 1255 OID 17152)
-- Name: query_8_1(integer); Type: FUNCTION; Schema: public;
Owner: postgres
--

CREATE FUNCTION public.query_8_1(remaining_count integer)
RETURNS TABLE(product_name character varying, quittance integer, remaining
integer, tickets_count integer, reception date, shop character varying)
LANGUAGE plpgsql SECURITY DEFINER
AS $$
begin
return query select t1.product_name, t1.quittance,
t1.remaining, t1.tickets_count, t1.reception, t1.shop
from (select p.product_name,
p.quittance,
p.remaining,
count(r.ticket):: integer as
tickets_count,
p.reception,
s.name as
shop
from realization r
inner join products p on
r.id_product = p.id
inner join shops s on
p.id_shop = s.id
where p.remaining > remaining_count
group by r.id_product, p.product_name,
p.quittance, p.remaining, p.reception, s.name
) as t1;
end;
$$;

ALTER FUNCTION public.query_8_1(remaining_count integer) OWNER
TO postgres;

--
-- TOC entry 276 (class 1255 OID 17151)
-- Name: query_8_2(character varying); Type: FUNCTION; Schema:
public; Owner: postgres

```

```

--
CREATE FUNCTION public.query_8_2(part_of_phone character
varying) RETURNS TABLE(first_name character varying, second_name character
varying, phone character, product_count integer)
LANGUAGE plpgsql SECURITY DEFINER
AS $$
begin
return query select t1.first_name, t1.second_name,
t1.phone, t1.count
from (select c.first_name, c.second_name,
c.phone, count(p.quittance) :: integer as count
from customers c
inner join products p on c.id
= p.id_customer
where c.phone similar to '%' ||
part_of_phone || '%'
group by c.phone, c.second_name,
c.first_name
order by count desc) as t1;

end;
$$;

ALTER FUNCTION public.query_8_2(part_of_phone character
varying) OWNER TO postgres;

--
-- TOC entry 273 (class 1255 OID 17146)
-- Name: query_8_3(integer); Type: FUNCTION; Schema: public;
Owner: postgres
--

CREATE FUNCTION public.query_8_3(product_id integer) RETURNS
TABLE(product_name character varying, quittance integer, shop character
varying, realizations_count integer)
LANGUAGE plpgsql SECURITY DEFINER
AS $$
begin
return query select t1.product_name, t1.quittance,
t1.shop, t1.realizations_count
from (select p.product_name,
p.quittance,
s.name
as shop,
count(r.realization_count)::integer as realizations_count
from realization r
right join products p on
r.id_product = p.id
inner join shops s on
p.id_shop = s.id
where r.id_product = product_id
group by p.id, s.name, p.quittance) as
t1;

end;
$$;

ALTER FUNCTION public.query_8_3(product_id integer) OWNER TO
postgres;

--
-- TOC entry 278 (class 1255 OID 17150)
-- Name: query_8_4(integer); Type: FUNCTION; Schema: public;
Owner: postgres
--

CREATE FUNCTION public.query_8_4(shop_id integer) RETURNS
TABLE(shop character varying, login integer)
LANGUAGE plpgsql SECURITY DEFINER
AS $$
begin
return query select t1.name, count(t1.login) :: integer
from (select s.name, w.login
from shops s
right join workers w on s.id =
w.id_shop
where s.id = shop_id) as t1
group by t1.name;

end;
$$;

ALTER FUNCTION public.query_8_4(shop_id integer) OWNER TO
postgres;

--
-- TOC entry 261 (class 1255 OID 17148)
-- Name: query_9_1(integer); Type: FUNCTION; Schema: public;
Owner: postgres
--

CREATE FUNCTION public.query_9_1(min_sold_count integer)
RETURNS TABLE(product_name character varying, shop character varying,
sold_count integer)
LANGUAGE plpgsql SECURITY DEFINER
AS $$
begin
return query select t1.product_name, t1.shop, t1.sold
from (select p.product_name, s.name as shop,
sum(r.realization_count)::integer as sold
from realization r
inner join products p on
r.id_product = p.id
inner join shops s on
p.id_shop = s.id
group by p.id, p.product_name, shop
having sum(r.realization_count) >
min_sold_count
order by shop, sold) as t1;

end;
$$;

ALTER FUNCTION public.query_9_1(min_sold_count integer) OWNER
TO postgres;

--
-- TOC entry 247 (class 1255 OID 16987)
-- Name: regenerate_all_tebles(integer, integer, integer,
integer, integer); Type: FUNCTION; Schema: public; Owner: postgres
--

CREATE FUNCTION public.regenerate_all_tebles(shops_count
integer, workers_count integer, products_count integer, customers_count
integer, realization_count integer) RETURNS void
LANGUAGE plpgsql
AS $$
begin
perform delete_from_all_tables();
perform generate_all_tables(shops_count, workers_count,
products_count,
customers_count,
realization_count);

end;
$$;

ALTER FUNCTION public.regenerate_all_tebles(shops_count
integer, workers_count integer, products_count integer, customers_count
integer, realization_count integer) OWNER TO postgres;

--
-- TOC entry 240 (class 1255 OID 16960)
-- Name: replace_worker(text); Type: FUNCTION; Schema: public;
Owner: postgres
--

CREATE FUNCTION public.replace_worker(username text) RETURNS
void
LANGUAGE plpgsql SECURITY DEFINER
AS $$
begin
execute format('drop role %s', username);

```

```

        delete from workers where login = username;
    end;
    $$;

ALTER FUNCTION public.replace_worker(username text) OWNER TO
postgres;

--
-- TOC entry 241 (class 1255 OID 25221)
-- Name: sell_product(integer, integer); Type: FUNCTION;
Schema: public; Owner: postgres
--

CREATE FUNCTION public.sell_product(quitance_ integer, count_
integer) RETURNS void
    LANGUAGE plpgsql SECURITY DEFINER
    AS $$
    begin
        insert into realization (ticket, id_product,
realization_date, realization_count, id_worker)
            VALUES (nextval('ticket_num'),
                    quitance_,
                    now()::date,
                    count_,
                    (select id from workers where login =
session_user));
    end;
    $$;

ALTER FUNCTION public.sell_product(quitance_ integer, count_
integer) OWNER TO postgres;

SET default_tablespace = '';

SET default_table_access_method = heap;

--
-- TOC entry 200 (class 1259 OID 16627)
-- Name: customers; Type: TABLE; Schema: public; Owner:
postgres
--

CREATE TABLE public.customers (
    id integer NOT NULL,
    first_name character varying(30) NOT NULL,
    second_name character varying(30) NOT NULL,
    id_district integer NOT NULL,
    id_status integer NOT NULL,
    work character varying(60),
    birthday date NOT NULL,
    phone character(10) NOT NULL,
    CONSTRAINT years_18 CHECK ((age((CURRENT_DATE)::timestamp
with time zone, (birthday)::timestamp with time zone) >= '18
years'::interval))
);

ALTER TABLE public.customers OWNER TO postgres;

--
-- TOC entry 201 (class 1259 OID 16631)
-- Name: customers_id_seq; Type: SEQUENCE; Schema: public;
Owner: postgres
--

CREATE SEQUENCE public.customers_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE public.customers_id_seq OWNER TO postgres;

--

```

```

-- TOC entry 3240 (class 0 OID 0)
-- Dependencies: 201
-- Name: customers_id_seq; Type: SEQUENCE OWNED BY; Schema:
public; Owner: postgres
--

ALTER SEQUENCE public.customers_id_seq OWNED BY
public.customers.id;

--
-- TOC entry 202 (class 1259 OID 16633)
-- Name: districts; Type: TABLE; Schema: public; Owner:
postgres
--

CREATE TABLE public.districts (
    id integer NOT NULL,
    district character varying(30)
);

ALTER TABLE public.districts OWNER TO postgres;

--
-- TOC entry 212 (class 1259 OID 16659)
-- Name: statuses; Type: TABLE; Schema: public; Owner:
postgres
--

CREATE TABLE public.statuses (
    id integer NOT NULL,
    status character varying(30)
);

ALTER TABLE public.statuses OWNER TO postgres;

--
-- TOC entry 231 (class 1259 OID 25189)
-- Name: customers_view; Type: VIEW; Schema: public; Owner:
postgres
--

CREATE VIEW public.customers_view AS
    SELECT c.id,
           c.first_name,
           c.second_name,
           d.district,
           s.status,
           c.work,
           c.birthday,
           c.phone
    FROM ((public.customers c
          JOIN public.statuses s ON ((c.id_status = s.id)))
          JOIN public.districts d ON ((c.id_district = d.id)));

ALTER TABLE public.customers_view OWNER TO postgres;

--
-- TOC entry 203 (class 1259 OID 16636)
-- Name: districts_id_seq; Type: SEQUENCE; Schema: public;
Owner: postgres
--

CREATE SEQUENCE public.districts_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE public.districts_id_seq OWNER TO postgres;

--
-- TOC entry 3245 (class 0 OID 0)

```

```

-- Dependencies: 203
-- Name: districts_id_seq; Type: SEQUENCE OWNED BY; Schema:
public; Owner: postgres
--

ALTER SEQUENCE public.districts_id_seq OWNED BY
public.districts.id;

--
-- TOC entry 204 (class 1259 OID 16638)
-- Name: product_types; Type: TABLE; Schema: public; Owner:
postgres

CREATE TABLE public.product_types (
    id integer NOT NULL,
    product_type character varying(30)
);

ALTER TABLE public.product_types OWNER TO postgres;

--
-- TOC entry 205 (class 1259 OID 16641)
-- Name: product_types_id_seq; Type: SEQUENCE; Schema: public;
Owner: postgres
--

CREATE SEQUENCE public.product_types_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE public.product_types_id_seq OWNER TO postgres;

--
-- TOC entry 3248 (class 0 OID 0)
-- Dependencies: 205
-- Name: product_types_id_seq; Type: SEQUENCE OWNED BY;
Schema: public; Owner: postgres
--

ALTER SEQUENCE public.product_types_id_seq OWNED BY
public.product_types.id;

--
-- TOC entry 206 (class 1259 OID 16643)
-- Name: products; Type: TABLE; Schema: public; Owner:
postgres
--

CREATE TABLE public.products (
    id integer NOT NULL,
    quittance integer,
    id_product_type integer,
    product_name character varying(30),
    reception date,
    count integer,
    remaining integer,
    price integer,
    id_customer integer,
    id_shop integer,
    id_woker integer,
    CONSTRAINT remaining_less_or_equal_count CHECK ((remaining
<= count))
);

ALTER TABLE public.products OWNER TO postgres;

--
-- TOC entry 207 (class 1259 OID 16647)

```

```

-- Name: products_id_seq; Type: SEQUENCE; Schema: public;
Owner: postgres
--

CREATE SEQUENCE public.products_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE public.products_id_seq OWNER TO postgres;

--
-- TOC entry 3251 (class 0 OID 0)
-- Dependencies: 207
-- Name: products_id_seq; Type: SEQUENCE OWNED BY; Schema:
public; Owner: postgres
--

ALTER SEQUENCE public.products_id_seq OWNED BY
public.products.id;

--
-- TOC entry 210 (class 1259 OID 16654)
-- Name: shops; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.shops (
    id integer NOT NULL,
    name character varying(30) NOT NULL
);

ALTER TABLE public.shops OWNER TO postgres;

--
-- TOC entry 232 (class 1259 OID 25214)
-- Name: products_modify_view; Type: VIEW; Schema: public;
Owner: testworker
--

CREATE VIEW public.products_modify_view WITH
(security_barrier='true') AS
    SELECT p.quittance,
           pt.product_type,
           p.product_name,
           p.reception,
           p.remaining,
           p.price,
           p.id_customer,
           c.first_name,
           c.second_name,
           c.phone,
           s.name AS shop
    FROM (((public.products p
            JOIN public.product_types pt ON ((p.id_product_type =
pt.id)))
            JOIN public.customers c ON ((p.id_customer = c.id)))
            JOIN public.shops s ON ((p.id_shop = s.id)))
    ORDER BY p.reception DESC, p.quittance DESC;

ALTER TABLE public.products_modify_view OWNER TO testworker;

--
-- TOC entry 214 (class 1259 OID 16664)
-- Name: workers; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.workers (
    id integer NOT NULL,
    login character varying(30) NOT NULL,
    id_shop integer NOT NULL,
    is_available boolean DEFAULT true
);

```

```

ALTER TABLE public.workers OWNER TO postgres;

--
-- TOC entry 234 (class 1259 OID 25236)
-- Name: products_view; Type: VIEW; Schema: public; Owner:
postgres
--

CREATE VIEW public.products_view AS
SELECT p.quittance,
       pt.product_type,
       p.product_name,
       p.reception,
       p.count,
       p.remaining,
       p.price,
       c.first_name,
       c.second_name,
       s.name,
       w.login
FROM (((public.products p
        JOIN public.customers c ON ((p.id_customer = c.id)))
      JOIN public.shops s ON ((p.id_shop = s.id)))
      JOIN public.workers w ON ((p.id_woker = w.id)))
      JOIN public.product_types pt ON ((p.id_product_type =
pt.id)));

ALTER TABLE public.products_view OWNER TO postgres;

--
-- TOC entry 233 (class 1259 OID 25222)
-- Name: query_11_1; Type: VIEW; Schema: public; Owner:
postgres
--

CREATE VIEW public.query_11_1 AS
SELECT (( SELECT avg(products.price) AS avg
          FROM public.products
          WHERE (products.id_shop = s.id))::integer AS
average_price,
        s.name AS shop
FROM public.shops s
ORDER BY ((( SELECT avg(products.price) AS avg
              FROM public.products
              WHERE (products.id_shop = s.id)))::integer) DESC;

ALTER TABLE public.query_11_1 OWNER TO postgres;

--
-- TOC entry 226 (class 1259 OID 17200)
-- Name: query_12_1; Type: VIEW; Schema: public; Owner:
postgres
--

CREATE VIEW public.query_12_1 AS
SELECT products.id,
       products.quittance,
       products.id_product_type,
       products.product_name,
       products.reception,
       products.count,
       products.remaining,
       products.price,
       products.id_customer,
       products.id_shop,
       products.id_woker
FROM public.products
WHERE (products.count = products.remaining)
UNION ALL
SELECT products.id,
       products.quittance,
       products.id_product_type,
       products.product_name,
       products.reception,
       products.count,

```

```

       products.remaining,
       products.price,
       products.id_customer,
       products.id_shop,
       products.id_woker
FROM public.products
WHERE (products.remaining = 0)
ORDER BY 7 DESC;

ALTER TABLE public.query_12_1 OWNER TO postgres;

--
-- TOC entry 227 (class 1259 OID 17205)
-- Name: query_13_1; Type: VIEW; Schema: public; Owner:
postgres
--

CREATE VIEW public.query_13_1 AS
SELECT customers.first_name,
       customers.second_name,
       customers.phone
FROM public.customers
WHERE (customers.id IN ( SELECT products.id_customer
                        FROM public.products
                        WHERE (products.remaining > 0)));

ALTER TABLE public.query_13_1 OWNER TO postgres;

--
-- TOC entry 228 (class 1259 OID 17214)
-- Name: query_13_2; Type: VIEW; Schema: public; Owner:
postgres
--

CREATE VIEW public.query_13_2 AS
SELECT customers.first_name,
       customers.second_name,
       customers.phone
FROM public.customers
WHERE (NOT (customers.id IN ( SELECT products.id_customer
                              FROM public.products
                              WHERE (products.remaining > 0))));

ALTER TABLE public.query_13_2 OWNER TO postgres;

--
-- TOC entry 229 (class 1259 OID 17249)
-- Name: query_13_3; Type: VIEW; Schema: public; Owner:
postgres
--

CREATE VIEW public.query_13_3 AS
SELECT products.quittance,
       products.product_name,
       products.price,
       products.remaining,
       products.count,
       CASE
         WHEN (((((products.remaining)::double precision /
(products.count)::double precision) * (100)::double precision))::integer =
100) THEN (('He nokynaetsя ('::text || (((products.remaining)::double
precision / (products.count)::double precision) * (100)::double
precision))::integer) || '%')::text)
         WHEN (((((products.remaining)::double precision /
(products.count)::double precision) * (100)::double precision))::integer >=
80) AND (((products.remaining)::double precision /
(products.count)::double precision) * (100)::double precision))::integer <=
99)) THEN (('Очень плохо nokynaetsя ('::text ||
((((products.remaining)::double precision / (products.count)::double
precision) * (100)::double precision))::integer) || '%')::text)
         WHEN (((((products.remaining)::double precision /
(products.count)::double precision) * (100)::double precision))::integer >=
60) AND (((products.remaining)::double precision /
(products.count)::double precision) * (100)::double precision))::integer <=
80)) THEN (('Плохо nokynaetsя ('::text || (((products.remaining)::double

```

```

precision / (products.count)::double precision) * (100)::double
precision)::integer) || '%'::text)
        WHEN ((((((products.remaining)::double precision /
(products.count)::double precision) * (100)::double precision)::integer >=
40) AND ((((((products.remaining)::double precision /
(products.count)::double precision) * (100)::double precision)::integer <=
60)) THEN (('Покынаецца ('::text || ((((((products.remaining)::double
precision / (products.count)::double precision) * (100)::double
precision)::integer) || '%'::text)
        WHEN ((((((products.remaining)::double precision /
(products.count)::double precision) * (100)::double precision)::integer >=
20) AND ((((((products.remaining)::double precision /
(products.count)::double precision) * (100)::double precision)::integer <=
40)) THEN (('Хорошо покынаецца ('::text || ((((((products.remaining)::double
precision / (products.count)::double precision) * (100)::double
precision)::integer) || '%'::text)
        WHEN ((((((products.remaining)::double precision /
(products.count)::double precision) * (100)::double precision)::integer >=
1) AND ((((((products.remaining)::double precision /
(products.count)::double precision) * (100)::double precision)::integer <=
20)) THEN (('Очень хорошо покынаецца ('::text ||
((((products.remaining)::double precision / (products.count)::double
precision) * (100)::double precision)::integer) || '%'::text)
        WHEN ((((((products.remaining)::double precision /
(products.count)::double precision) * (100)::double precision)::integer =
0) THEN (('Раскупили ('::text || ((((((products.remaining)::double precision
/ (products.count)::double precision) * (100)::double precision)::integer)
|| '%'::text)
        ELSE ((((((products.remaining)::double precision /
(products.count)::double precision) * (100)::double precision)::integer ||
' error'::text)
    END AS state
FROM public.products
ORDER BY ((((((products.remaining)::double precision /
(products.count)::double precision) * (100)::double precision)::integer),
products.count DESC;

```

```

ALTER TABLE public.query_13_3 OWNER TO postgres;

--
-- TOC entry 230 (class 1259 OID 17262)
-- Name: query_13_4; Type: VIEW; Schema: public; Owner:
postgres
--

```

```

CREATE VIEW public.query_13_4 AS
SELECT products.quittance,
       products.product_name,
       products.price,
       s.name AS shop
FROM (public.products
      JOIN public.shops s ON ((products.id_shop = s.id)))
WHERE ((products.price)::numeric > ( SELECT
avg(products_1.price) AS avg
      FROM public.products products_1))
ORDER BY s.name, products.price;

```

```

ALTER TABLE public.query_13_4 OWNER TO postgres;

--
-- TOC entry 220 (class 1259 OID 17094)
-- Name: query_2_1; Type: VIEW; Schema: public; Owner:
postgres
--

```

```

CREATE VIEW public.query_2_1 AS
SELECT (((customers.first_name)::text || ' '::text) ||
(customers.second_name)::text) AS customer_name,
       customers.work,
       p.quittance,
       p.remaining,
       s.name
FROM ((public.customers
      JOIN public.products p ON ((customers.id =
p.id_customer)))
      JOIN public.shops s ON ((p.id_shop = s.id)))

```

```

ORDER BY (((customers.first_name)::text || ' '::text) ||
(customers.second_name)::text);

```

```

ALTER TABLE public.query_2_1 OWNER TO postgres;

--
-- TOC entry 208 (class 1259 OID 16649)
-- Name: realization; Type: TABLE; Schema: public; Owner:
postgres
--

```

```

CREATE TABLE public.realization (
    id integer NOT NULL,
    ticket integer,
    id_product integer,
    realization_date date,
    realization_count integer,
    id_worker integer
);

```

```

ALTER TABLE public.realization OWNER TO postgres;

--
-- TOC entry 218 (class 1259 OID 17078)
-- Name: query_2_2; Type: VIEW; Schema: public; Owner:
postgres
--

```

```

CREATE VIEW public.query_2_2 AS
SELECT products.quittance,
       products.product_name,
       products.reception,
       r.realization_count,
       r.realization_date,
       s.name AS shop
FROM ((public.products
      JOIN public.realization r ON ((products.id =
r.id_product)))
      JOIN public.shops s ON ((products.id_shop = s.id)))
ORDER BY s.name, products.reception, r.realization_date,
r.realization_count;

```

```

ALTER TABLE public.query_2_2 OWNER TO postgres;

--
-- TOC entry 219 (class 1259 OID 17086)
-- Name: query_2_3; Type: VIEW; Schema: public; Owner:
postgres
--

```

```

CREATE VIEW public.query_2_3 AS
SELECT realization.ticket,
       realization.realization_date,
       realization.realization_count,
       w.login,
       s.name
FROM (((public.realization
      JOIN public.products p ON ((realization.id_product =
p.id)))
      JOIN public.shops s ON ((p.id_shop = s.id)))
      JOIN public.workers w ON ((p.id_woker = w.id)))
ORDER BY s.name, realization.realization_date, w.login;

```

```

ALTER TABLE public.query_2_3 OWNER TO postgres;

--
-- TOC entry 221 (class 1259 OID 17108)
-- Name: query_3_1; Type: VIEW; Schema: public; Owner:
postgres
--

```

```

CREATE VIEW public.query_3_1 AS
SELECT products.quittance,
       products.count,
       products.remaining,

```

```

        r.ticket
    FROM (public.products
        LEFT JOIN public.realization r ON ((products.id =
r.id_product)))
    ORDER BY products.quittance;

ALTER TABLE public.query_3_1 OWNER TO postgres;

--
-- TOC entry 222 (class 1259 OID 17112)
-- Name: query_4_1; Type: VIEW; Schema: public; Owner:
postgres

--

CREATE VIEW public.query_4_1 AS
    SELECT r.ticket,
           r.realization_date,
           r.realization_count,
           workers.login
    FROM (public.workers
        RIGHT JOIN public.realization r ON ((workers.id =
r.id_worker)));

ALTER TABLE public.query_4_1 OWNER TO postgres;

--
-- TOC entry 223 (class 1259 OID 17116)
-- Name: query_5_1; Type: VIEW; Schema: public; Owner:
postgres

--

CREATE VIEW public.query_5_1 AS
    SELECT products.quittance,
           products.product_name,
           products.reception,
           ( SELECT workers.login
             FROM public.workers
            WHERE (workers.id = products.id_woker)) AS login
    FROM public.products;

ALTER TABLE public.query_5_1 OWNER TO postgres;

--
-- TOC entry 224 (class 1259 OID 17127)
-- Name: query_6_1; Type: VIEW; Schema: public; Owner:
postgres

--

CREATE VIEW public.query_6_1 AS
    SELECT sold_products.quittance,
           sold_products.count,
           sold_products.reception,
           s.name
    FROM (( SELECT products.id,
           products.quittance,
           products.id_product_type,
           products.product_name,
           products.reception,
           products.count,
           products.remaining,
           products.price,
           products.id_customer,
           products.id_shop,
           products.id_woker
          FROM public.products
         WHERE (products.remaining = 0)) sold_products
    JOIN public.shops s ON ((sold_products.id_shop = s.id)))
    ORDER BY s.name, sold_products.reception;

ALTER TABLE public.query_6_1 OWNER TO postgres;

--
-- TOC entry 225 (class 1259 OID 17133)
-- Name: query_7_1; Type: VIEW; Schema: public; Owner:
postgres

--

        r.ticket
    FROM (public.products
        LEFT JOIN public.realization r ON ((products.id =
r.id_product)))
    ORDER BY products.quittance;

ALTER TABLE public.query_7_1 OWNER TO postgres;

--
-- TOC entry 216 (class 1259 OID 16818)
-- Name: quittance_num; Type: SEQUENCE; Schema: public; Owner:
postgres

--

CREATE SEQUENCE public.quittance_num
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE public.quittance_num OWNER TO postgres;

--
-- TOC entry 209 (class 1259 OID 16652)
-- Name: realization_id_seq; Type: SEQUENCE; Schema: public;
Owner: postgres

--

CREATE SEQUENCE public.realization_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE public.realization_id_seq OWNER TO postgres;

--
-- TOC entry 3273 (class 0 OID 0)
-- Dependencies: 209
-- Name: realization_id_seq; Type: SEQUENCE OWNED BY; Schema:
public; Owner: postgres

--

ALTER SEQUENCE public.realization_id_seq OWNED BY
public.realization.id;

--
-- TOC entry 235 (class 1259 OID 25241)
-- Name: realization_view; Type: VIEW; Schema: public; Owner:
postgres

--

CREATE VIEW public.realization_view AS
    SELECT r.ticket,
           p.product_name,
           r.realization_date,
           r.realization_count,
           w.login
    FROM ((public.realization r
    JOIN public.products p ON ((r.id_product = p.id)))
    JOIN public.workers w ON ((r.id_worker = w.id)))
    ORDER BY r.id_product;

ALTER TABLE public.realization_view OWNER TO postgres;

--
-- TOC entry 211 (class 1259 OID 16657)

```

```

-- Name: shops_id_seq; Type: SEQUENCE; Schema: public; Owner:
postgres
--
CREATE SEQUENCE public.shops_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE public.shops_id_seq OWNER TO postgres;

--
-- TOC entry 3276 (class 0 OID 0)
-- Dependencies: 211
-- Name: shops_id_seq; Type: SEQUENCE OWNED BY; Schema:
public; Owner: postgres
--

ALTER SEQUENCE public.shops_id_seq OWNED BY public.shops.id;

--
-- TOC entry 213 (class 1259 OID 16662)
-- Name: statuses_id_seq; Type: SEQUENCE; Schema: public;
Owner: postgres
--

CREATE SEQUENCE public.statuses_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE public.statuses_id_seq OWNER TO postgres;

--
-- TOC entry 3278 (class 0 OID 0)
-- Dependencies: 213
-- Name: statuses_id_seq; Type: SEQUENCE OWNED BY; Schema:
public; Owner: postgres
--

ALTER SEQUENCE public.statuses_id_seq OWNED BY
public.statuses.id;

--
-- TOC entry 237 (class 1259 OID 25250)
-- Name: test_view; Type: VIEW; Schema: public; Owner:
postgres
--

CREATE VIEW public.test_view AS
    SELECT pt.product_type,
           sum(r.realization_count) AS sum
    FROM ((public.realization r
          JOIN public.products p ON ((r.id_product = p.id)))
          JOIN public.product_types pt ON ((p.id_product_type =
pt.id)))
    GROUP BY pt.product_type;

ALTER TABLE public.test_view OWNER TO postgres;

--
-- TOC entry 217 (class 1259 OID 16843)
-- Name: ticket_num; Type: SEQUENCE; Schema: public; Owner:
postgres
--

CREATE SEQUENCE public.ticket_num
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE public.ticket_num OWNER TO postgres;

--
-- TOC entry 215 (class 1259 OID 16667)
-- Name: workers_id_seq; Type: SEQUENCE; Schema: public;
Owner: postgres
--

CREATE SEQUENCE public.workers_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE public.workers_id_seq OWNER TO postgres;

--
-- TOC entry 3282 (class 0 OID 0)
-- Dependencies: 215
-- Name: workers_id_seq; Type: SEQUENCE OWNED BY; Schema:
public; Owner: postgres
--

ALTER SEQUENCE public.workers_id_seq OWNED BY
public.workers.id;

--
-- TOC entry 236 (class 1259 OID 25246)
-- Name: workers_view; Type: VIEW; Schema: public; Owner:
postgres
--

CREATE VIEW public.workers_view AS
    SELECT w.login,
           s.name,
           w.is_available
    FROM (public.workers w
          JOIN public.shops s ON ((w.id_shop = s.id)))
    ORDER BY w.id DESC;

ALTER TABLE public.workers_view OWNER TO postgres;

--
-- TOC entry 3014 (class 2604 OID 16669)
-- Name: customers id; Type: DEFAULT; Schema: public; Owner:
postgres
--

ALTER TABLE ONLY public.customers ALTER COLUMN id SET DEFAULT
nextval('public.customers_id_seq'::regclass);

--
-- TOC entry 3016 (class 2604 OID 16670)
-- Name: districts id; Type: DEFAULT; Schema: public; Owner:
postgres
--

ALTER TABLE ONLY public.districts ALTER COLUMN id SET DEFAULT
nextval('public.districts_id_seq'::regclass);

--
-- TOC entry 3017 (class 2604 OID 16671)
-- Name: product_types id; Type: DEFAULT; Schema: public;
Owner: postgres
--

```



```
ALTER TABLE ONLY public.product_types ALTER COLUMN id SET
DEFAULT nextval('public.product_types_id_seq'::regclass);
```

```
--
-- TOC entry 3018 (class 2604 OID 16672)
-- Name: products id; Type: DEFAULT; Schema: public; Owner:
postgres
```

```
ALTER TABLE ONLY public.products ALTER COLUMN id SET DEFAULT
nextval('public.products_id_seq'::regclass);
```

```
--
-- TOC entry 3020 (class 2604 OID 16673)
-- Name: realization id; Type: DEFAULT; Schema: public; Owner:
postgres
```

```
ALTER TABLE ONLY public.realization ALTER COLUMN id SET
DEFAULT nextval('public.realization_id_seq'::regclass);
```

```
--
-- TOC entry 3021 (class 2604 OID 16674)
-- Name: shops id; Type: DEFAULT; Schema: public; Owner:
postgres
```

```
ALTER TABLE ONLY public.shops ALTER COLUMN id SET DEFAULT
nextval('public.shops_id_seq'::regclass);
```

```
--
-- TOC entry 3022 (class 2604 OID 16675)
-- Name: statuses id; Type: DEFAULT; Schema: public; Owner:
postgres
```

```
ALTER TABLE ONLY public.statuses ALTER COLUMN id SET DEFAULT
nextval('public.statuses_id_seq'::regclass);
```

```
--
-- TOC entry 3023 (class 2604 OID 16676)
-- Name: workers id; Type: DEFAULT; Schema: public; Owner:
postgres
```

```
ALTER TABLE ONLY public.workers ALTER COLUMN id SET DEFAULT
nextval('public.workers_id_seq'::regclass);
```

```
--
-- TOC entry 3216 (class 0 OID 16627)
-- Dependencies: 200
-- Data for Name: customers; Type: TABLE DATA; Schema: public;
Owner: postgres
```

```
--
-- TOC entry 3026 (class 2606 OID 16678)
-- Name: customers customers_pkey; Type: CONSTRAINT; Schema:
public; Owner: postgres
```

```
ALTER TABLE ONLY public.customers
ADD CONSTRAINT customers_pkey PRIMARY KEY (id);
```

```
--
-- TOC entry 3028 (class 2606 OID 16680)
```

```
-- Name: districts districts_pkey; Type: CONSTRAINT; Schema:
public; Owner: postgres
```

```
ALTER TABLE ONLY public.districts
ADD CONSTRAINT districts_pkey PRIMARY KEY (id);
```

```
--
-- TOC entry 3030 (class 2606 OID 16682)
-- Name: product_types product_types_pkey; Type: CONSTRAINT;
Schema: public; Owner: postgres
```

```
ALTER TABLE ONLY public.product_types
ADD CONSTRAINT product_types_pkey PRIMARY KEY (id);
```

```
--
-- TOC entry 3032 (class 2606 OID 16684)
-- Name: products products_pkey; Type: CONSTRAINT; Schema:
public; Owner: postgres
```

```
ALTER TABLE ONLY public.products
ADD CONSTRAINT products_pkey PRIMARY KEY (id);
```

```
--
-- TOC entry 3034 (class 2606 OID 16686)
-- Name: realization realization_pkey; Type: CONSTRAINT;
Schema: public; Owner: postgres
```

```
ALTER TABLE ONLY public.realization
ADD CONSTRAINT realization_pkey PRIMARY KEY (id);
```

```
--
-- TOC entry 3036 (class 2606 OID 16688)
-- Name: shops shops_pkey; Type: CONSTRAINT; Schema: public;
Owner: postgres
```

```
ALTER TABLE ONLY public.shops
ADD CONSTRAINT shops_pkey PRIMARY KEY (id);
```

```
--
-- TOC entry 3038 (class 2606 OID 16690)
-- Name: statuses statuses_pkey; Type: CONSTRAINT; Schema:
public; Owner: postgres
```

```
ALTER TABLE ONLY public.statuses
ADD CONSTRAINT statuses_pkey PRIMARY KEY (id);
```

```
--
-- TOC entry 3042 (class 2606 OID 16692)
-- Name: workers workers_pkey; Type: CONSTRAINT; Schema:
public; Owner: postgres
```

```
ALTER TABLE ONLY public.workers
ADD CONSTRAINT workers_pkey PRIMARY KEY (id);
```

```
--
-- TOC entry 3039 (class 1259 OID 25255)
-- Name: workers_login_idx; Type: INDEX; Schema: public;
Owner: postgres
```

```
CREATE UNIQUE INDEX workers_login_idx ON public.workers USING
btree (lower((login)::text));
```

```

-- TOC entry 3040 (class 1259 OID 25188)
-- Name: workers_login_uindex; Type: INDEX; Schema: public;
Owner: postgres
--

CREATE UNIQUE INDEX workers_login_uindex ON public.workers
USING btree (login);

--

-- TOC entry 3052 (class 2620 OID 25260)
-- Name: products product_count_check; Type: TRIGGER; Schema:
public; Owner: postgres
--

CREATE TRIGGER product_count_check BEFORE UPDATE ON
public.products FOR EACH ROW EXECUTE FUNCTION public.count_check();

--

-- TOC entry 3053 (class 2620 OID 16886)
-- Name: realization
product_count_decrease_on_realization_before_insert; Type: TRIGGER; Schema:
public; Owner: postgres
--

CREATE TRIGGER
product_count_decrease_on_realization_before_insert BEFORE INSERT ON
public.realization FOR EACH ROW EXECUTE FUNCTION
public.decrease_products();

--

-- TOC entry 3055 (class 2620 OID 25219)
-- Name: products_modify_view
products_modify_view_instead_of_insert; Type: TRIGGER; Schema: public;
Owner: testworker
--

CREATE TRIGGER products_modify_view_instead_of_insert INSTEAD
OF INSERT ON public.products_modify_view FOR EACH ROW EXECUTE FUNCTION
public.instead_of_insert();

--

-- TOC entry 3054 (class 2620 OID 25263)
-- Name: workers workers_is_available_before_update; Type:
TRIGGER; Schema: public; Owner: postgres
--

CREATE TRIGGER workers_is_available_before_update BEFORE
UPDATE ON public.workers FOR EACH ROW EXECUTE FUNCTION
public.is_available_check();

--

-- TOC entry 3043 (class 2606 OID 16693)
-- Name: customers customers_id_district_fkey; Type: FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.customers
ADD CONSTRAINT customers_id_district_fkey FOREIGN KEY
(id_district) REFERENCES public.districts(id) ON UPDATE CASCADE ON DELETE
CASCADE;

--

-- TOC entry 3044 (class 2606 OID 16698)
-- Name: customers customers_id_status_fkey; Type: FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.customers
ADD CONSTRAINT customers_id_status_fkey FOREIGN KEY
(id_status) REFERENCES public.statutes(id) ON UPDATE CASCADE ON DELETE
CASCADE;

```

```

--
-- TOC entry 3045 (class 2606 OID 16703)
-- Name: products products_id_customer_fkey; Type: FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.products
ADD CONSTRAINT products_id_customer_fkey FOREIGN KEY
(id_customer) REFERENCES public.customers(id) ON UPDATE CASCADE ON DELETE
CASCADE;

--

-- TOC entry 3046 (class 2606 OID 16708)
-- Name: products products_id_product_type_fkey; Type: FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.products
ADD CONSTRAINT products_id_product_type_fkey FOREIGN KEY
(id_product_type) REFERENCES public.product_types(id) ON UPDATE CASCADE ON
DELETE CASCADE;

--

-- TOC entry 3047 (class 2606 OID 16713)
-- Name: products products_id_shop_fkey; Type: FK CONSTRAINT;
Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.products
ADD CONSTRAINT products_id_shop_fkey FOREIGN KEY (id_shop)
REFERENCES public.shops(id) ON UPDATE CASCADE ON DELETE CASCADE;

--

-- TOC entry 3048 (class 2606 OID 16718)
-- Name: products products_id_woker_fkey; Type: FK CONSTRAINT;
Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.products
ADD CONSTRAINT products_id_woker_fkey FOREIGN KEY
(id_woker) REFERENCES public.workers(id) ON UPDATE CASCADE ON DELETE
CASCADE;

--

-- TOC entry 3049 (class 2606 OID 16723)
-- Name: realization realization_id_product_fkey; Type: FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.realization
ADD CONSTRAINT realization_id_product_fkey FOREIGN KEY
(id_product) REFERENCES public.products(id) ON UPDATE CASCADE ON DELETE
CASCADE;

--

-- TOC entry 3050 (class 2606 OID 16728)
-- Name: realization realization_id_worker_fkey; Type: FK
CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.realization
ADD CONSTRAINT realization_id_worker_fkey FOREIGN KEY
(id_worker) REFERENCES public.workers(id) ON UPDATE CASCADE ON DELETE
CASCADE;

--

-- TOC entry 3051 (class 2606 OID 16812)
-- Name: workers workers_shops_id_fk; Type: FK CONSTRAINT;
Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.workers

```

```

        ADD CONSTRAINT workers_shops_id_fk FOREIGN KEY (id_shop)
REFERENCES public.shops(id) ON UPDATE CASCADE ON DELETE CASCADE;

```

```

--
-- TOC entry 3215 (class 3256 OID 25201)
-- Name: workers administrator_realization_policy; Type:
POLICY; Schema: public; Owner: postgres
--

```

```

        CREATE POLICY administrator_realization_policy ON
public.workers TO administrator USING (true);

```

```

--
-- TOC entry 3206 (class 0 OID 16643)
-- Dependencies: 206
-- Name: products; Type: ROW SECURITY; Schema: public; Owner:
postgres
--

```

```

ALTER TABLE public.products ENABLE ROW LEVEL SECURITY;

```

```

--
-- TOC entry 3208 (class 0 OID 16649)
-- Dependencies: 208
-- Name: realization; Type: ROW SECURITY; Schema: public;
Owner: postgres
--

```

```

ALTER TABLE public.realization ENABLE ROW LEVEL SECURITY;

```

```

--
-- TOC entry 3209 (class 3256 OID 25196)
-- Name: realization realization_shop_policy; Type: POLICY;
Schema: public; Owner: postgres
--

```

```

        CREATE POLICY realization_shop_policy ON public.realization TO
worker USING ((EXISTS ( SELECT products.id
        FROM public.products
        WHERE ((products.id = realization.id_product) AND
(products.id_shop = ( SELECT workers.id_shop
        FROM public.workers))))));

```

```

--
-- TOC entry 3211 (class 3256 OID 25168)
-- Name: shops shop_id_policy; Type: POLICY; Schema: public;
Owner: postgres
--

```

```

        CREATE POLICY shop_id_policy ON public.shops FOR SELECT TO
worker USING ((id = ( SELECT workers.id_shop
        FROM public.workers
        WHERE ((workers.login)::text = CURRENT_USER))));

```

```

--
-- TOC entry 3212 (class 3256 OID 25230)
-- Name: shops shop_to_admin_policy; Type: POLICY; Schema:
public; Owner: postgres
--

```

```

        CREATE POLICY shop_to_admin_policy ON public.shops FOR SELECT
TO administrator USING (true);

```

```

--
-- TOC entry 3210 (class 0 OID 16654)
-- Dependencies: 210
-- Name: shops; Type: ROW SECURITY; Schema: public; Owner:
postgres
--

```

```

ALTER TABLE public.shops ENABLE ROW LEVEL SECURITY;

```

```

--
-- TOC entry 3213 (class 0 OID 16664)

```

```

-- Dependencies: 214
-- Name: workers; Type: ROW SECURITY; Schema: public; Owner:
postgres
--

```

```

ALTER TABLE public.workers ENABLE ROW LEVEL SECURITY;

```

```

--
-- TOC entry 3214 (class 3256 OID 16985)
-- Name: workers workers_id_policy; Type: POLICY; Schema:
public; Owner: postgres
--

```

```

        CREATE POLICY workers_id_policy ON public.workers FOR SELECT
TO worker USING ((CURRENT_USER = (login)::text));

```

```

--
-- TOC entry 3207 (class 3256 OID 16977)
-- Name: products workers_shop_policy; Type: POLICY; Schema:
public; Owner: postgres
--

```

```

        CREATE POLICY workers_shop_policy ON public.products TO worker
USING ((id_shop = ( SELECT workers.id_shop
        FROM public.workers
        WHERE ((workers.login)::text = CURRENT_USER))));

```

```

--
-- TOC entry 3239 (class 0 OID 0)
-- Dependencies: 200
-- Name: TABLE customers; Type: ACL; Schema: public; Owner:
postgres
--

```

```

GRANT SELECT,INSERT ON TABLE public.customers TO worker;
GRANT SELECT ON TABLE public.customers TO holder;

```

```

--
-- TOC entry 3241 (class 0 OID 0)
-- Dependencies: 201
-- Name: SEQUENCE customers_id_seq; Type: ACL; Schema: public;
Owner: postgres
--

```

```

GRANT USAGE ON SEQUENCE public.customers_id_seq TO worker;

```

```

--
-- TOC entry 3242 (class 0 OID 0)
-- Dependencies: 202
-- Name: TABLE districts; Type: ACL; Schema: public; Owner:
postgres
--

```

```

GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE public.districts TO
administrator;
GRANT SELECT ON TABLE public.districts TO holder;
GRANT SELECT ON TABLE public.districts TO worker;

```

```

--
-- TOC entry 3243 (class 0 OID 0)
-- Dependencies: 212
-- Name: TABLE statuses; Type: ACL; Schema: public; Owner:
postgres
--

```

```

GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE public.statuses TO
administrator;
GRANT SELECT ON TABLE public.statuses TO holder;
GRANT SELECT ON TABLE public.statuses TO worker;

```

```

--
-- TOC entry 3244 (class 0 OID 0)
-- Dependencies: 231

```

```

-- Name: TABLE customers_view; Type: ACL; Schema: public;
Owner: postgres
--

GRANT SELECT ON TABLE public.customers_view TO worker;
GRANT SELECT ON TABLE public.customers_view TO holder;

--
-- TOC entry 3246 (class 0 OID 0)
-- Dependencies: 203
-- Name: SEQUENCE districts_id_seq; Type: ACL; Schema: public;
Owner: postgres
--

GRANT USAGE ON SEQUENCE public.districts_id_seq TO
administrator;

--
-- TOC entry 3247 (class 0 OID 0)
-- Dependencies: 204
-- Name: TABLE product_types; Type: ACL; Schema: public;
Owner: postgres
--

GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE
public.product_types TO administrator;
GRANT SELECT ON TABLE public.product_types TO holder;
GRANT SELECT ON TABLE public.product_types TO worker;

--
-- TOC entry 3249 (class 0 OID 0)
-- Dependencies: 205
-- Name: SEQUENCE product_types_id_seq; Type: ACL; Schema:
public; Owner: postgres
--

GRANT USAGE ON SEQUENCE public.product_types_id_seq TO
administrator;

--
-- TOC entry 3250 (class 0 OID 0)
-- Dependencies: 206
-- Name: TABLE products; Type: ACL; Schema: public; Owner:
postgres
--

GRANT SELECT,INSERT ON TABLE public.products TO worker;
GRANT SELECT ON TABLE public.products TO holder;

--
-- TOC entry 3252 (class 0 OID 0)
-- Dependencies: 207
-- Name: SEQUENCE products_id_seq; Type: ACL; Schema: public;
Owner: postgres
--

GRANT SELECT,USAGE ON SEQUENCE public.products_id_seq TO
worker;

--
-- TOC entry 3253 (class 0 OID 0)
-- Dependencies: 210
-- Name: TABLE shops; Type: ACL; Schema: public; Owner:
postgres
--

GRANT SELECT ON TABLE public.shops TO holder;
GRANT SELECT ON TABLE public.shops TO worker;
GRANT SELECT ON TABLE public.shops TO administrator;

--
-- TOC entry 3254 (class 0 OID 0)

```

```

-- Dependencies: 232
-- Name: TABLE products_modify_view; Type: ACL; Schema:
public; Owner: testworker
--

GRANT SELECT ON TABLE public.products_modify_view TO holder;
GRANT SELECT ON TABLE public.products_modify_view TO worker;

--
-- TOC entry 3255 (class 0 OID 0)
-- Dependencies: 214
-- Name: TABLE workers; Type: ACL; Schema: public; Owner:
postgres
--

GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE public.workers TO
administrator;
GRANT SELECT ON TABLE public.workers TO holder;
GRANT SELECT ON TABLE public.workers TO worker;

--
-- TOC entry 3256 (class 0 OID 0)
-- Dependencies: 234
-- Name: TABLE products_view; Type: ACL; Schema: public;
Owner: postgres
--

GRANT SELECT ON TABLE public.products_view TO holder;

--
-- TOC entry 3257 (class 0 OID 0)
-- Dependencies: 233
-- Name: TABLE query_11_1; Type: ACL; Schema: public; Owner:
postgres
--

GRANT SELECT ON TABLE public.query_11_1 TO holder;

--
-- TOC entry 3258 (class 0 OID 0)
-- Dependencies: 226
-- Name: TABLE query_12_1; Type: ACL; Schema: public; Owner:
postgres
--

GRANT SELECT ON TABLE public.query_12_1 TO holder;

--
-- TOC entry 3259 (class 0 OID 0)
-- Dependencies: 227
-- Name: TABLE query_13_1; Type: ACL; Schema: public; Owner:
postgres
--

GRANT SELECT ON TABLE public.query_13_1 TO holder;

--
-- TOC entry 3260 (class 0 OID 0)
-- Dependencies: 228
-- Name: TABLE query_13_2; Type: ACL; Schema: public; Owner:
postgres
--

GRANT SELECT ON TABLE public.query_13_2 TO holder;

--
-- TOC entry 3261 (class 0 OID 0)
-- Dependencies: 229
-- Name: TABLE query_13_3; Type: ACL; Schema: public; Owner:
postgres
--

```

```

GRANT SELECT ON TABLE public.query_13_3 TO holder;

--

-- TOC entry 3262 (class 0 OID 0)
-- Dependencies: 230
-- Name: TABLE query_13_4; Type: ACL; Schema: public; Owner:
postgres

GRANT SELECT ON TABLE public.query_13_4 TO holder;

--

-- TOC entry 3263 (class 0 OID 0)
-- Dependencies: 220
-- Name: TABLE query_2_1; Type: ACL; Schema: public; Owner:
postgres

GRANT SELECT ON TABLE public.query_2_1 TO holder;

--

-- TOC entry 3264 (class 0 OID 0)
-- Dependencies: 208
-- Name: TABLE realization; Type: ACL; Schema: public; Owner:
postgres

GRANT SELECT,INSERT ON TABLE public.realization TO worker;
GRANT SELECT ON TABLE public.realization TO holder;

--

-- TOC entry 3265 (class 0 OID 0)
-- Dependencies: 218
-- Name: TABLE query_2_2; Type: ACL; Schema: public; Owner:
postgres

GRANT SELECT ON TABLE public.query_2_2 TO holder;

--

-- TOC entry 3266 (class 0 OID 0)
-- Dependencies: 219
-- Name: TABLE query_2_3; Type: ACL; Schema: public; Owner:
postgres

GRANT SELECT ON TABLE public.query_2_3 TO holder;

--

-- TOC entry 3267 (class 0 OID 0)
-- Dependencies: 221
-- Name: TABLE query_3_1; Type: ACL; Schema: public; Owner:
postgres

GRANT SELECT ON TABLE public.query_3_1 TO holder;

--

-- TOC entry 3268 (class 0 OID 0)
-- Dependencies: 222
-- Name: TABLE query_4_1; Type: ACL; Schema: public; Owner:
postgres

GRANT SELECT ON TABLE public.query_4_1 TO holder;

--

-- TOC entry 3269 (class 0 OID 0)
-- Dependencies: 223
-- Name: TABLE query_5_1; Type: ACL; Schema: public; Owner:
postgres

--

GRANT SELECT ON TABLE public.query_5_1 TO holder;

--

-- TOC entry 3270 (class 0 OID 0)
-- Dependencies: 224
-- Name: TABLE query_6_1; Type: ACL; Schema: public; Owner:
postgres

GRANT SELECT ON TABLE public.query_6_1 TO holder;

--

-- TOC entry 3271 (class 0 OID 0)
-- Dependencies: 225
-- Name: TABLE query_7_1; Type: ACL; Schema: public; Owner:
postgres

GRANT SELECT ON TABLE public.query_7_1 TO holder;

--

-- TOC entry 3272 (class 0 OID 0)
-- Dependencies: 216
-- Name: SEQUENCE quittance_num; Type: ACL; Schema: public;
Owner: postgres

GRANT USAGE ON SEQUENCE public.quittance_num TO worker;

--

-- TOC entry 3274 (class 0 OID 0)
-- Dependencies: 209
-- Name: SEQUENCE realization_id_seq; Type: ACL; Schema:
public; Owner: postgres

GRANT USAGE ON SEQUENCE public.realization_id_seq TO worker;

--

-- TOC entry 3275 (class 0 OID 0)
-- Dependencies: 235
-- Name: TABLE realization_view; Type: ACL; Schema: public;
Owner: postgres

GRANT SELECT ON TABLE public.realization_view TO holder;

--

-- TOC entry 3277 (class 0 OID 0)
-- Dependencies: 211
-- Name: SEQUENCE shops_id_seq; Type: ACL; Schema: public;
Owner: postgres

GRANT USAGE ON SEQUENCE public.shops_id_seq TO administrator;

--

-- TOC entry 3279 (class 0 OID 0)
-- Dependencies: 213
-- Name: SEQUENCE statuses_id_seq; Type: ACL; Schema: public;
Owner: postgres

GRANT USAGE ON SEQUENCE public.statuses_id_seq TO
administrator;

--

-- TOC entry 3280 (class 0 OID 0)
-- Dependencies: 237
-- Name: TABLE test_view; Type: ACL; Schema: public; Owner:
postgres

```

```
--
--
GRANT SELECT ON TABLE public.test_view TO holder;

--
-- TOC entry 3281 (class 0 OID 0)
-- Dependencies: 217
-- Name: SEQUENCE ticket_num; Type: ACL; Schema: public;
Owner: postgres
--

GRANT USAGE ON SEQUENCE public.ticket_num TO worker;

--

-- TOC entry 3283 (class 0 OID 0)
-- Dependencies: 236
-- Name: TABLE workers_view; Type: ACL; Schema: public; Owner:
postgres
--

GRANT SELECT ON TABLE public.workers_view TO holder;

-- Completed on 2021-06-04 08:56:07

--
-- PostgreSQL database dump complete
--
```

ПРИЛОЖЕНИЕ Г. АНТИПЛАГИАТ



Отчет о проверке на заимствования №1



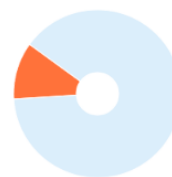
Автор: Моргунов Арсений
Проверяющий: Моргунов Арсений (mag17122000@mail.ru / ID: 7983884)
 Отчет предоставлен сервисом «Антиплагиат» - users.antiplagiat.ru

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 7
 Начало загрузки: 04.06.2021 09:28:33
 Длительность загрузки: 00:00:03
 Имя исходного файла:
 PsSBD_Kursovaya_Morgunov.pdf
 Название документа:
 PsSBD_Kursovaya_Morgunov
 Размер текста: 123 КБ
 Символов в тексте: 125935
 Слов в тексте: 15815
 Число предложений: 1146

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Начало проверки: 04.06.2021 09:28:36
 Длительность проверки: 00:00:09
 Комментарий: не указано
 Модуль поиска: Интернет



ЗАИМСТВОВАНИЯ

11,28%

САМОЦИТИРОВАНИЯ

0%

ЦИТИРОВАНИЯ

0%

ОРИГИНАЛЬНОСТЬ

88,72%

Заимствования — доля всех найденных текстовых пересечений, за исключением тех, которые система отнесла к цитированиям, по отношению к общему объему документа.
 Самоцитирования — доля фрагментов текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника, автором или соавтором которого является автор проверяемого документа, по отношению к общему объему документа.
 Цитирования — доля текстовых пересечений, которые не являются авторскими, но система посчитала их использование корректным, по отношению к общему объему документа. Сюда относятся оформленные по ГОСТу цитаты; общепотребительные выражения; фрагменты текста, найденные в источниках из коллекций нормативно-правовой документации.
 Текстовое пересечение — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.
 Источник — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.
 Оригинальность — доля фрагментов текста проверяемого документа, не обнаруженных ни в одном источнике, по которым шла проверка, по отношению к общему объему документа.
 Заимствования, самоцитирования, цитирования и оригинальность являются отдельными показателями и в сумме дают 100%, что соответствует всему тексту проверяемого документа.
 Обращаем Ваше внимание, что система находит текстовые пересечения проверяемого документа с проиндексированными в системе текстовыми источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности заимствований или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.

№	Доля в отчете	Источник	Актуален на	Модуль поиска
[01]	10,43%	Красным выделена не нужная нам информация, зелёным – нужная, не выделенная – информация об объекте ниже (name и type). — MegaЛекции http://megalektsii.ru	12 Июл 2017	Интернет
[02]	0,49%	Полная версия научной работы 672 КБ http://scienceforum.ru	17 Апр 2018	Интернет
[03]	0,36%	kem_pkd_OBD_lab (2013 год) http://studfiles.ru	14 Июл 2016	Интернет

Рисунок Г.1 – Антиплагиат

ПРИЛОЖЕНИЕ Д. РУКОВОДСТВО РАБОТНИКА

Рассмотрим вкладку «Продукты». В таблице выводится информация о товарах, которые располагаются в магазине работника, который вошел в систему, при этом выводятся только товары, количество которых на складе больше 0, т.е. еще есть на складе (это обеспечивается политиками работников). При нажатии на товар в таблице в блоке интерфейса справа отображается подробная информация о товаре, а также о клиенте, который его сдал в магазин. Также при нажатии на товар становится активной кнопка «Продать выделенный товар», при нажатии на которую продается указанное количество выделенного товара. Для поиска товара по части названия достаточно ввести часть названия товара в поле «Имя товара» и нажать кнопку «Поиск по имени». Нажатие на кнопку «Поиск товаров в наличии» выводит в таблицу все товары, которые располагаются в магазине работника, который вошел в систему, при этом выводятся только товары, количество которых на складе больше 0, т.е. еще есть на складе. Для добавления товара на склад нужно заполнить поля, которые находятся в правой нижней части формы, после чего нажать на кнопку «Добавить товар».

Рассмотрим вкладку «Реализация». В левой части формы расположена таблица, которая отображает информацию о всех продажах, совершенных в этом магазине. Если нажать на элемент в этой таблице, то в таблице, расположенной в правой части формы выведется информация о товаре, который был приобретен выбранным чеком. Если нажать на кнопку «Поиск по реализации», то в левой таблице будут выведены все продажи, в которых количество приобретенного товара будет больше указанного. В правой части формы расположена таблица, содержащая все товары, которые были и есть на складе в текущем магазине. При нажатии на элемент в этой таблице, то в таблице, которая расположена слева,

выведутся все чеки, выписанные на выбранный товар. При нажатии на кнопку «Поиск по имени» в правой таблице выведутся все товары, у которых в названии содержится указанная последовательность символов.

Рассмотрим вкладку «Клиенты». В таблице «Информация о клиенте» выводится информация обо всех клиентах, которые содержатся в базе данных. При нажатии на элемент таблицы «Информация о клиенте» в таблице «Товары клиента» выведутся все товары клиента, которые он сдал в текущий магазин. При нажатии на кнопку «Поиск по всем клиентам» в таблице «Информация о клиенте» выведутся все клиенты, у которых в имени содержится последовательность символов, указанная в поле «Имя» и в фамилии содержится последовательность символов, указанная в поле «Фамилия». При нажатии на кнопку «Поиск по клиентам магазина» в таблице «Информация о клиенте» выведутся клиенты текущего магазина (т.е. они сдали хотя бы один товар в этот магазин), у которых в имени содержится последовательность символов, указанная в поле «Имя» и в фамилии содержится последовательность символов, указанная в поле «Фамилия». Для добавления клиента необходимо заполнить все поля, которые находятся в блоке «Добавление клиента», а затем нажать кнопку «Добавить клиента». В таблице «Товары клиента» выводится информация обо всех товарах, которые были или есть на складе текущего магазина. При нажатии на элемент таблицы «Товары клиента» в таблице «Информация о клиенте» будет выведен клиент, который сдал выделенный товар. При нажатии на кнопку «Поиск по товарам» в таблице «Товары клиента» будут выведены товары, в названии которых содержится последовательность символов, которая указана в поле «Название товара».

ПРИЛОЖЕНИЕ Е. РУКОВОДСТВО АДМИНИСТРАТОРА

Рассмотрим вкладку «Роли». В таблице «Работники» выводятся все работники, находящиеся в базе данных. При нажатии на кнопку «Найти работника» в таблицу «Роли» выводятся работники, у которых в логине есть последовательность символов, которая указана в поле «Логин работника». Чтобы добавить нового работника необходимо ввести его логин, пароль в поля «Логин работника» и «Пароль работника» соответственно, а также выбрать магазин, в который он будет добавлен. После этого нужно нажать на кнопку «Добавить работника». Для увольнения работника необходимо выделить элемент таблицы «Работники», который содержит работника, которого нужно уволить, а затем нажать на кнопку «Уволить работника», после чего значение поля «is_available» изменится на false, а роль работника будет удалена из базы данных.

В таблице «Владельцы» выводятся все владельцы. При нажатии на кнопку «Найти владельца» в таблице «Владельцы» будут выведены владельцы, у которых в логине есть последовательность символов, которая указана в поле «Логин владельца». Чтобы добавить нового владельца необходимо ввести его логин, пароль в поля «Логин владельца» и «Пароль владельца» соответственно. После этого нужно нажать на кнопку «Добавить владельца». Для удаления владельца необходимо выделить элемент таблицы «Владельцы», который содержит владельца, которого нужно удалить, а затем нажать на кнопку «Удалить владельца».

Рассмотрим вкладку «Справочники». Для выбора справочника, необходимо в выпадающем списке выбрать имя справочника. После этого в таблице отобразится содержимое справочника. Для того чтобы найти запись в справочнике нужно ввести часть названия и нажать кнопку «Найти». Для добавления записи в справочник необходимо ввести

название новой записи и нажать кнопку «Добавить». Для удаления записи необходимо выбрать запись в таблице и нажать кнопку «Удалить выбранное».

ПРИЛОЖЕНИЕ Ё. РУКОВОДСТВО ВЛАДЕЛЬЦА

Рассмотрим вкладку «Таблицы». Для выбора таблицы, необходимо в выпадающем списке выбрать имя таблицы. После этого в таблице на экране отобразится содержимое выбранной таблицы. Для того чтобы найти запись в таблице нужно ввести последовательность символов, которую должны включать искомые записи, и нажать кнопку «Найти». Для таблицы работников поиск происходит по логину, для таблицы товары и реализация – по названию товара, для таблицы заказчики по имени.

Рассмотрим вкладку «Запросы». Для выбора запроса необходимо выбрать его из выпадающего списка. Если в запросе есть параметр, то его необходимо записать в поле «Параметр», если есть второй параметр, то его нужно записать в поле «Параметр 2». Если все параметры введены, либо запрос не требует параметров, то нужно нажать кнопку «Выполнить запрос», для отображения результатов запроса в виде таблицы.

Рассмотрим вкладку «График». Для того, чтобы просмотреть статистику в виде графика и таблицы необходимо нажать кнопку «Построить диаграмму». Для того, чтобы сгенерировать Excel файл по статистике необходимо нажать на кнопку «Импорт в Excel», а затем выбрать расположение и имя создаваемого файла, после чего нажать на кнопку «Сохранить».