Колледж ТОО "Astana IT University"

**Отчет**

**по производственному обучению**

Группа: ПО2207

ФИО студентов: Турсунбаев Нурсултан и Нурпеисов Мансур

ФИО преподавателя: Тулабоев К.С.

г. Астана

# Содержание

[Введение..........................................................................................................3](#_gjdgxs)

Создание схемы в draw.io............................................................................. .4

Создание базы данных car\_rent.....................................................................6

Заполнение столбцов......................................................................................7

Вывод данных из БД с помощью ЯП Python................................................8

Библиотека psycopg2......................................................................................9

Описание связей между сущностями "Заказ", "Автомобиль", "Клиент" и "Сотрудник":……………………………………………………………….11

Структуристика Базы данных для автосалона…………………………..12

Цели проекта……………………………………………………………….17

Описание интерфейса базы данных для автосалона…………………….21

Заключение....................................................................................................24

# Введение

Задача и цель поставленные перед студентами по предмету «Организация обработки больших данных»

Задачи:

1. Выбрать тему для БД
2. Создание схемы в Draw.io на основе выбранной темы
3. Создание БД на основе схемы из Draw.io
4. Заполнение таблиц данными (не менее 100 элементов в каждом столбце)
5. Найти способ вывести данные через ЯП Python

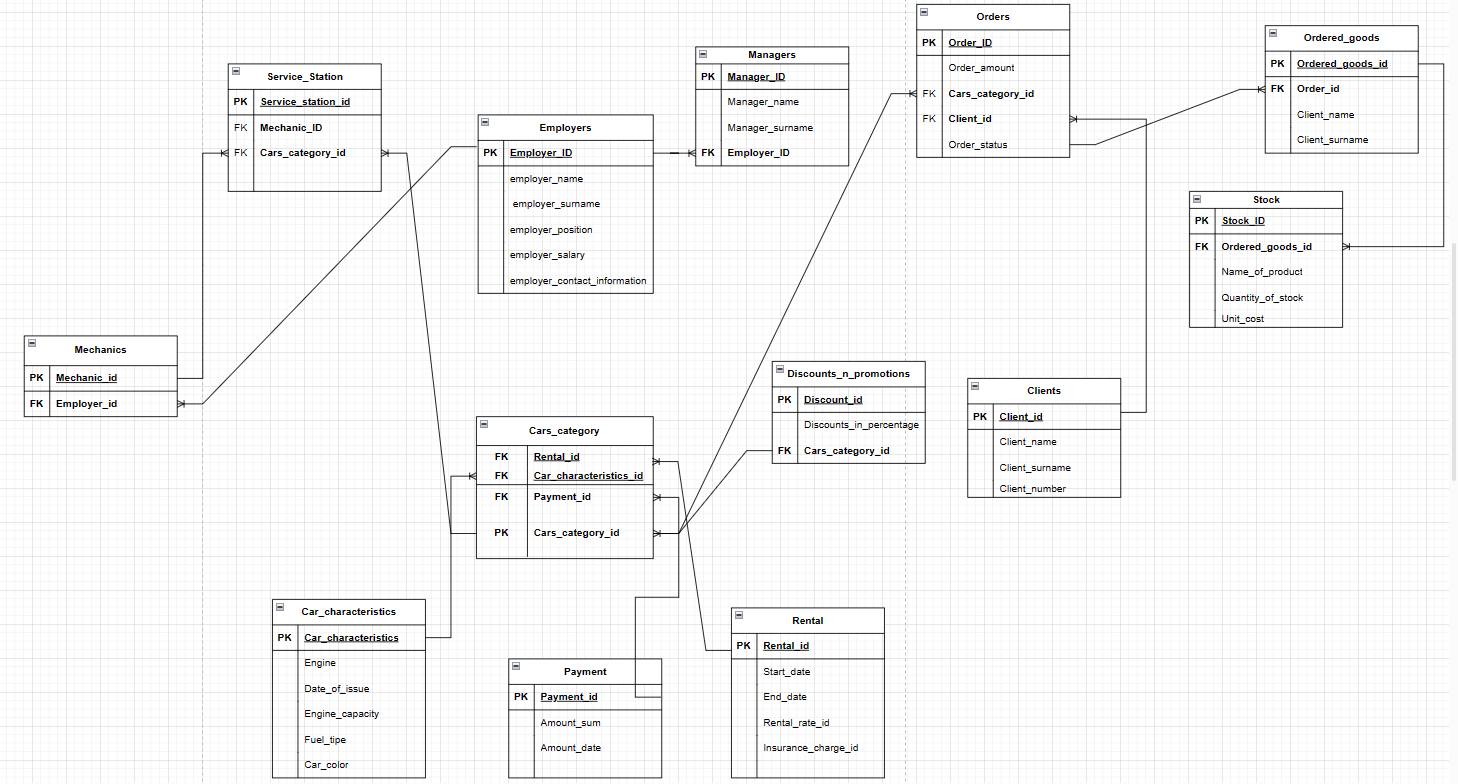
Цель производственного обучения заключалась в создании БД в PostgreSQL на основе выбранной темы, а также вывода данных с помощью языка Python

Задачей практики является подготовка будущих специалистов к практической реализации различных БД

**Создание схемы в Draw.io**

Для того, чтобы создать хорошую БД, всегда требуется сначала представить себе примерный вид ваших таблиц и то, как они будут связаны, для этого лучше всего использовать диаграммы. После того как вы создадите таблицы, нужно заполнить их данными – столбцами (с названием и типом данных). Далее нужно указать связи между таблицами с помощью указания ключей (Primary key и Foreign key). Мы создали 13 таблиц по тематике Автосалона с возможностью ремонта и аренды автомобилей.

**Сама схема**



**Таблицы, которые мы создали:**

* Mechanics
* Service\_station
* Employers
* Managers
* Orders
* Ordered\_goods
* Stock
* Discounts\_n\_promotions
* Clients – содержит информацию о клиентах
* Cars\_category
* Car\_characteristics
* Payment
* Rental

# Создание БД в PostgreSQL

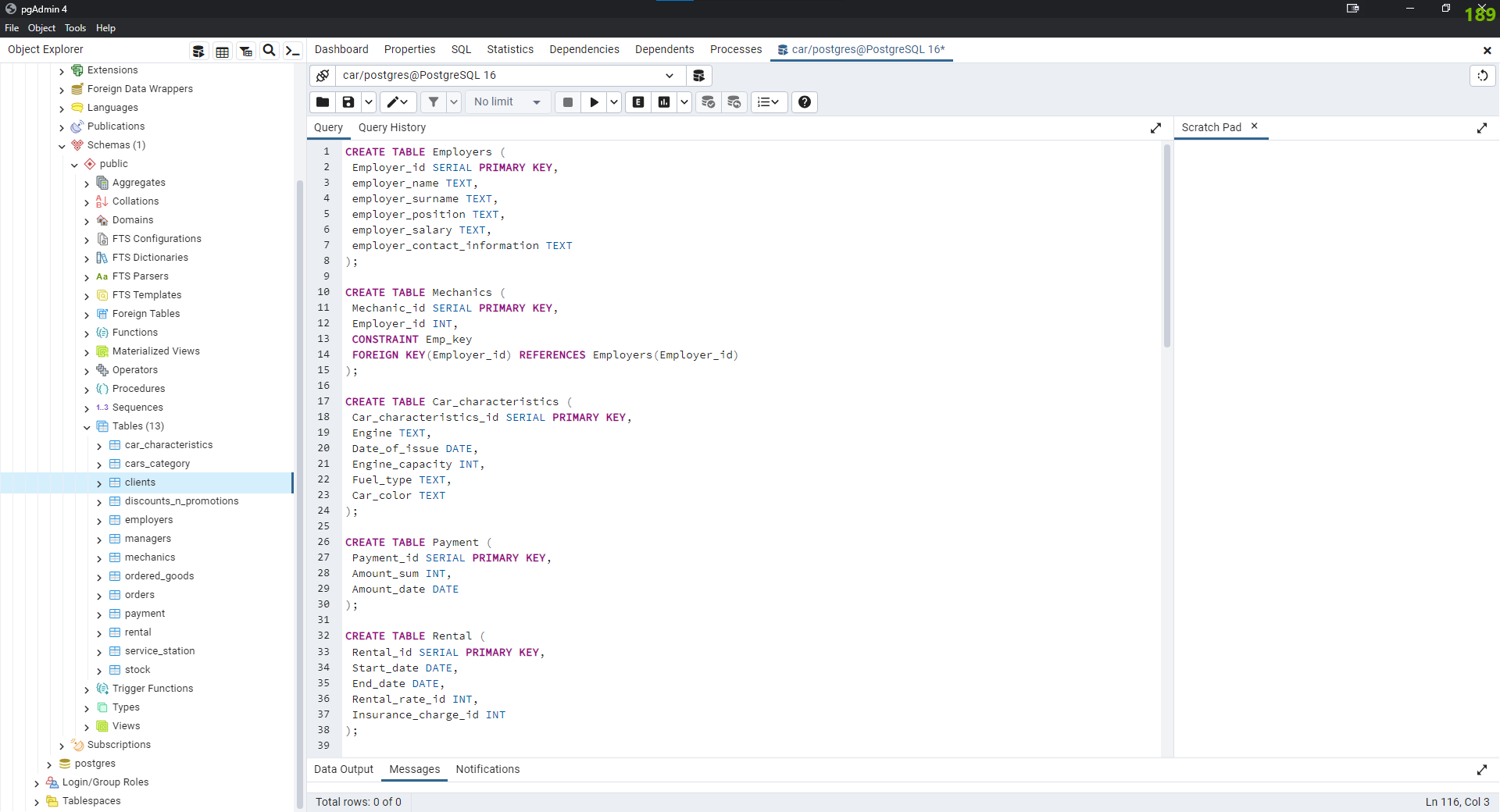
Имея схему нашей БД, было начато создание таблиц в PostgreSQL

Создано 13 таблиц в базе данных car\_rent с использованием команды SQL (CREATE TABLE). И наполнено их столбцами, присвоены типы данных в зависимости от столбца

Для обеспечения целостности данных, и установки связи, было установлено

соответствующие ограничения, такие как PRIMARY KEY и FOREIGN KEY.

**Скрин части кода**



**Заполнение базы данных**

Заполнение базы данных происходило с помощью онлайн - генератора Mockaroo, включая создание соединения с базой данных и загрузку

данных.

Mockaroo дает создать данные различных типов, включая имена,

адреса, номера телефонов, даты, валюты, названия стран и т.п.

Также он сам с помощью AI может понять, что хочет запросить пользователь

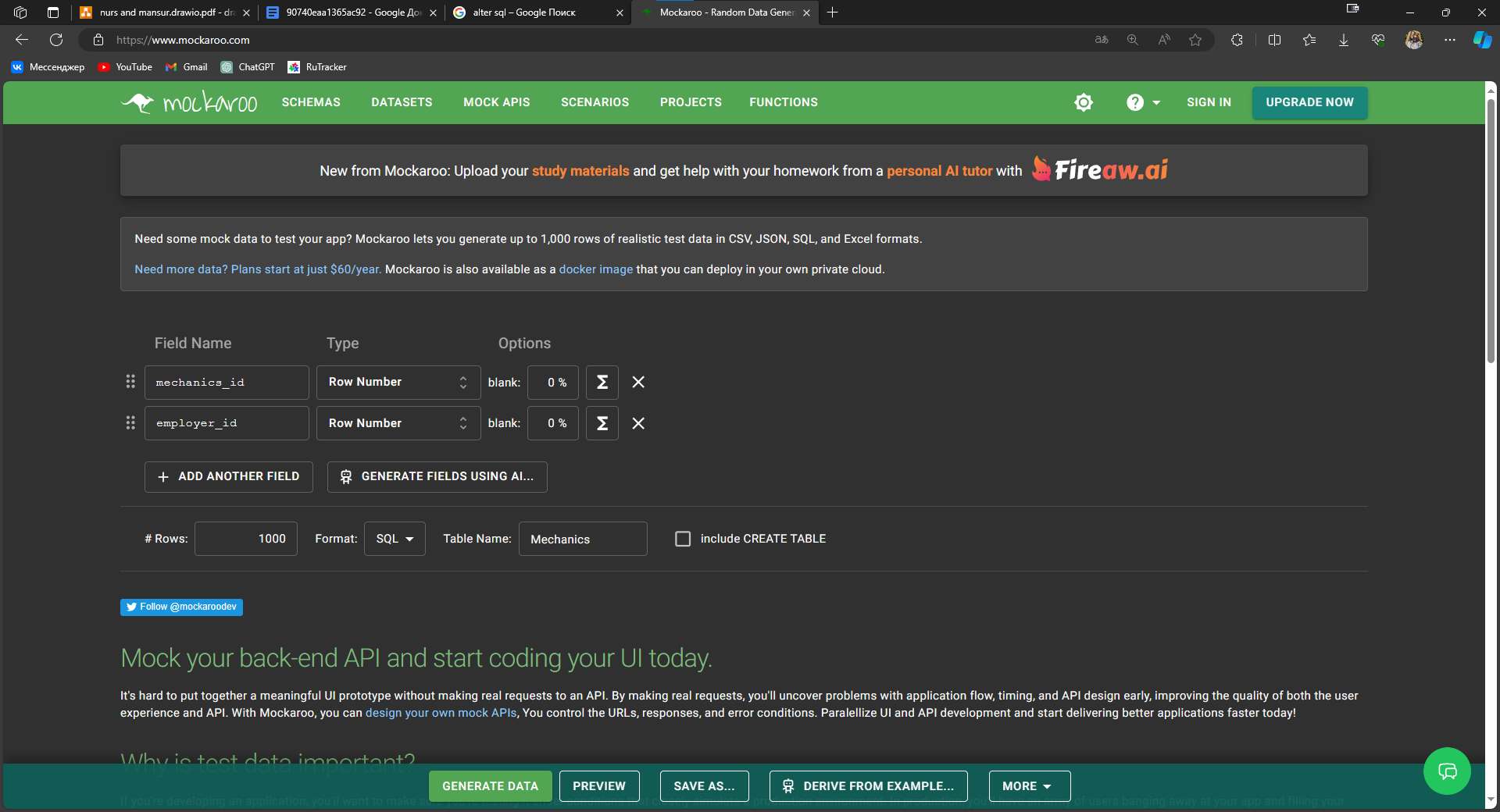
После генерации данных в Mockaroo, мы скачали их в sql формате,

после чего проверив правильность типов данных вставили в БД через QueryTool,

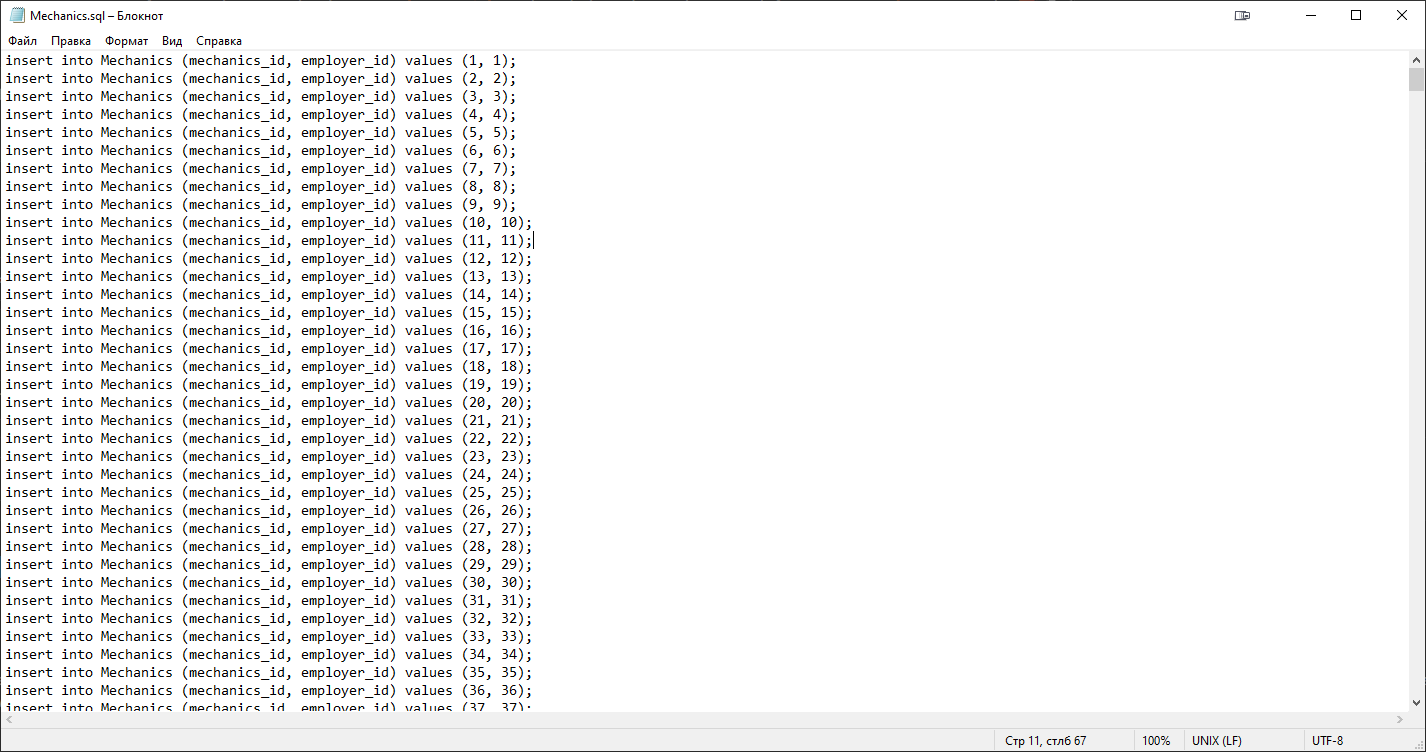
Mockaroo использует правильный синтаксис заполнения таблиц, поэтому проблем не возникло

После загрузки данных мы проверили правильность введенных данных

**Процесс генерации данных**



**Запрос, который вывел Mockaroo (используется команда INSERT INTO)**



**Вывод данных с помощью ЯП Python**

Для начала в среде разработки (было использовано Pycharm) нужно установить сам ЯП Python, затем установлено в библиотеку psycopg2, с помощью команды import, эта библиотека нужна для взаимодействия с PostgreSQL

Psycopg2

**Библиотека psycopg2 и структура кода:**

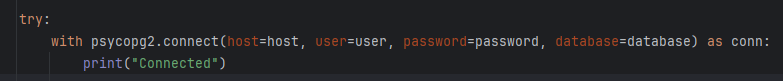
Выбрана данную библиотеку чтобы связать Python проект с БД

(Структурой Базой Данных):

**Объяснение кода**



Импорт параметров подключения из файла config:



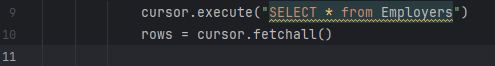
try используется для попытки установить соединение с базой данных и выполнить SQL-запрос

В этой части кода мы пытаемся установить соединение с базой данных PostgreSQL, используя параметры подключения, которые были импортированы из файла config.py. conn представляет собой объект соединения с базой данных.

Если соединение было успешно установлено, этот фрагмент кода выведет "Connected" в консоль.



Здесь создается курсор, который будет использоваться для выполнения SQL-запросов к базе данных.

****

Здесь создается курсор, который будет использоваться для выполнения SQL-запросов к базе данных, затем запрашиваются все строки из таблицы с именем "Employers" в базе данных.



Результаты запроса (результирующие строки) сохраняются в переменной rows.



Здесь результаты, полученные в виде кортежей, преобразуются в список кортежей result\_list, где каждый кортеж представляет одну строку из результата запроса.



Выводятся результаты



Если возникнут ошибки (например, не удастся установить соединение), то выводит:

"[INFO] ERROR while working with PostgreSQL"

**Полный код**

main.py

import psycopg2

from config import host, user, password, database

try:

with psycopg2.connect(host=host, user=user, password=password, database=database) as conn:

print("Connected")

with conn.cursor() as cursor:

cursor.execute("SELECT \* from Employers")

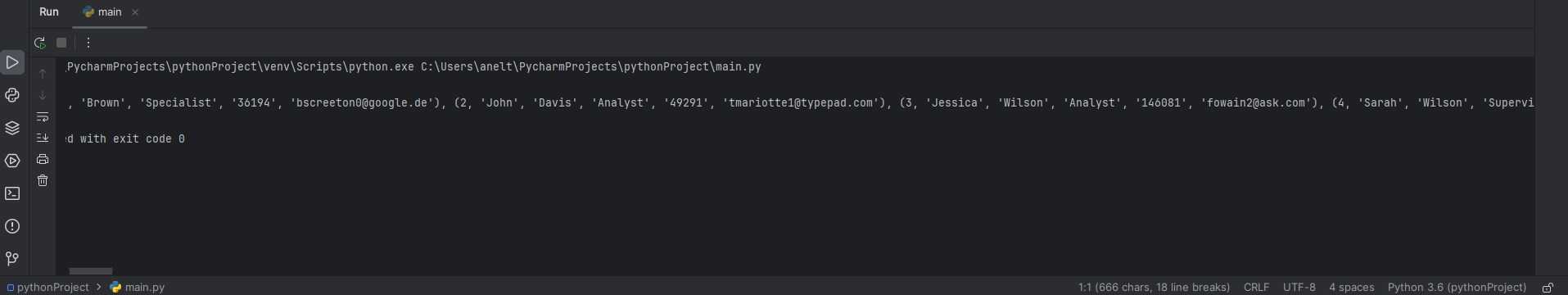
rows = cursor.fetchall()

result\_list = [row for row in rows]

print(result\_list)

except psycopg2.Error as ex:

print("[INFO] ERROR while working with PostgreSQL", ex)

**Вывод кода**  


**Описание связей между сущностями "Заказ", "Автомобиль", "Клиент" и "Сотрудник":**

**Заказ и Автомобиль:**

Каждый заказ привязан к конкретному автомобилю.

У каждого заказа есть связь с одним конкретным автомобилем, что означает, что заказ относится к определенному транспортному средству.

**Заказ и Клиент:**

Каждый заказ привязан к конкретному клиенту.

У каждого заказа есть связь с одним конкретным клиентом, что позволяет идентифицировать заказ в контексте определенного клиента.

**Заказ и Сотрудник:**

Каждый заказ привязан к конкретному сотруднику.

У каждого заказа есть связь с одним конкретным сотрудником, что указывает на ответственного исполнителя, обрабатывающего этот заказ.

**Клиент и Заказы:**

У каждого клиента может быть несколько заказов.

Связь "один ко многим" между клиентом и заказами позволяет однозначно идентифицировать все заказы, принадлежащие конкретному клиенту.

**Сотрудник и Заказы:**

Каждый сотрудник может обрабатывать несколько заказов.

Связь "один ко многим" между сотрудником и заказами означает, что у каждого сотрудника может быть множество заказов, которые он обрабатывает.

Эти связи позволяют эффективно моделировать взаимосвязи между заказами, автомобилями, клиентами и сотрудниками в вашей системе.

**Структуристика Базы данных для автосалона.**

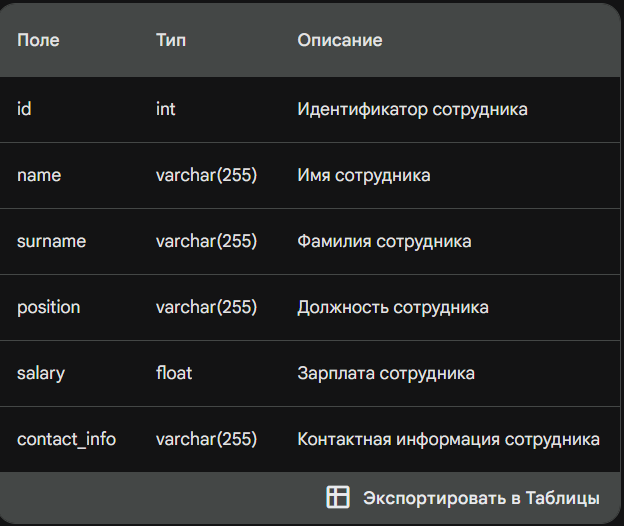
Этот репозиторий содержит SQL скрипты и схемы базы данных для управления автосалоном. База данных

разработана для хранения информации о автомобилях, клиентах, заказах и сотрудниках автосалона.

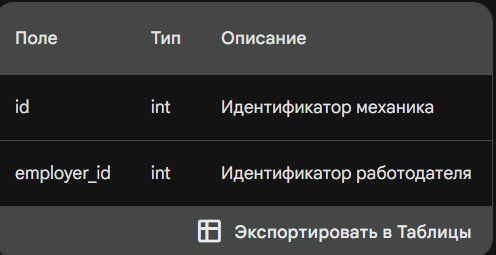
**Структура базы данных**

**База данных состоит из следующих таблиц:**

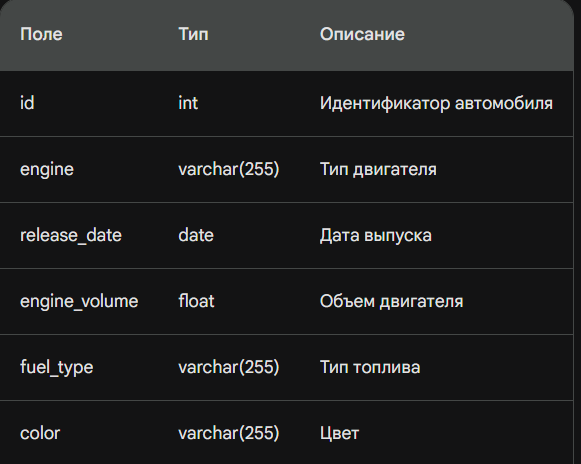
**Employers** - Содержит информацию о сотрудниках автосалона, их идентификатор, имя, фамилию, должность, зарплату и контактную информацию.



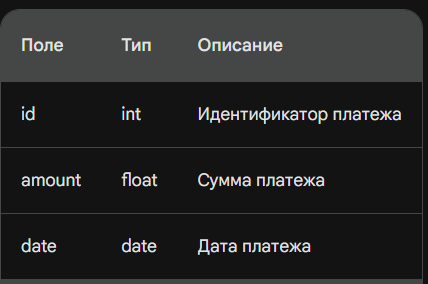
**Mechanics** - Таблица для механиков, связана с таблицей Employers по идентификатору сотрудника,



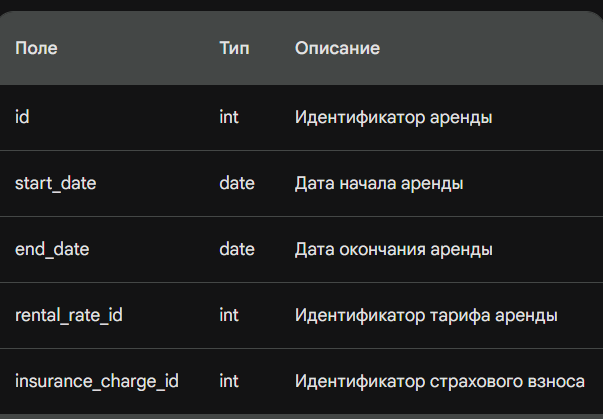
**Car characteristics** - Хранит характеристики автомобилей, такие как двигатель, дата выпуска, объем двигателя, тип топлива и цвет.



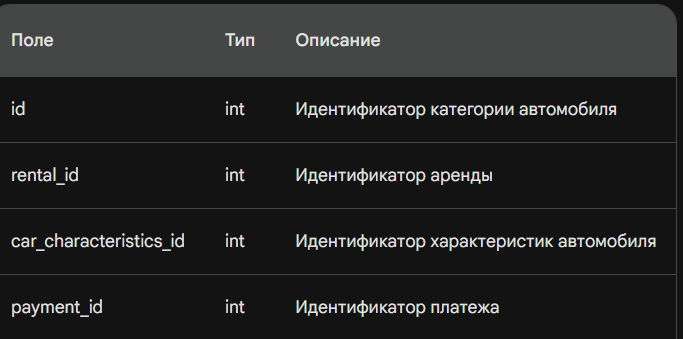
**Payment** - Содержит информацию о платежах, сумму и дату.



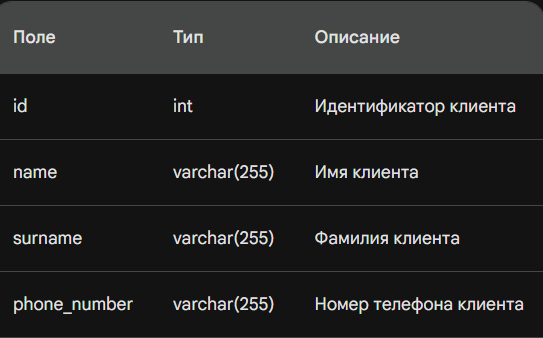
**Rental** - Таблица для аренды автомобилей, содержит дату начала и окончания аренды, а также связана с таблицами Rental\_rate и Insurance\_charge.



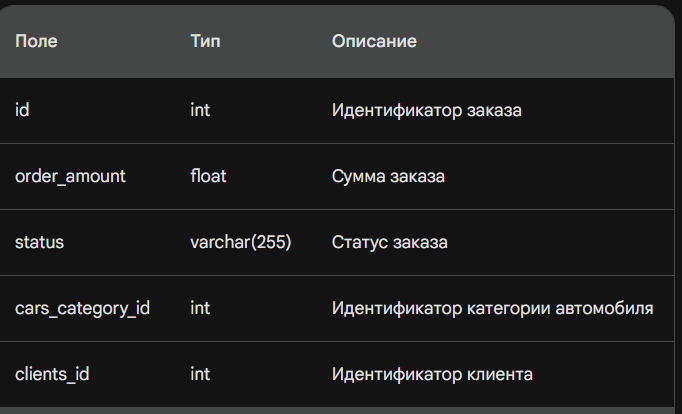
**Cars category** - Связывает информацию об аренде автомобиля с его характеристиками и платежами.



**Clients** - Хранит данные о клиентах: их идентификатор, имя, фамилию и номер телефона.



**Orders** - Таблица для заказов, содержит информацию о заказе, сумму, статус, а также связана с таблицами Cars\_category и Clients.



**Цели проекта**

Целью проекта является создание базы данных, которая может использоваться для управления автосалоном. База данных должна обеспечивать хранение и доступ к информации о следующих объектах:

**Автомобили**

База данных должна хранить информацию о всех автомобилях, представленных в автосалоне. Эта информация должна включать в себя следующие данные:

\* Идентификатор автомобиля

\* Марка автомобиля

Модель автомобиля

\* Год выпуска

\* Объем двигателя

\* Тип топлива

\* Цвет

\* Цена

Клиенты

База данных должна хранить информацию о всех клиентах, которые обращались в автосалон. Эта информация должна включать в себя следующие данные:

\* Идентификатор клиента

\* Имя клиента

Фамилия клиента

\* Номер телефона

\* Адрес электронной почты

**Заказы**

База данных должна хранить информацию о всех заказах, которые были оформлены в автосалоне. Эта информация должна включать в себя следующие данные:

\* Идентификатор заказа

\* Дата заказа

\* Сумма заказа

\* Статус заказа

\* Категория автомобиля

\* Идентификатор клиента

**Сотрудники**

База данных должна хранить информацию о всех сотрудниках автосалона. Эта информация должна включать в себя следующие данные:

\* Идентификатор сотрудника

\* Имя сотрудника

Фамилия сотрудника

\* Должность сотрудника

Заработная плата

\* Контактная информация

Более подробно о целях проекта

Цели проекта можно разделить на следующие группы:

**Хранение данных**

База данных должна обеспечивать хранение данных о всех объектах, перечисленных выше. Для этого в базе данных должны быть созданы соответствующие таблицы.

**Доступ к данным**

База данных должна обеспечивать доступ к данным для сотрудников автосалона. Для этого необходимо определить права доступа к данным для различных категорий пользователей.

**Аналитика данных**

База данных должна позволять сотрудникам автосалона выполнять аналитику данных. Для этого необходимо создать соответствующие запросы и отчеты.

**Примеры использования базы данных**

База данных может использоваться для следующих целей:

**Управление запасом автомобилей**

База данных может использоваться для отслеживания количества автомобилей на складе, а также для управления ценами на автомобили.

**Обслуживание клиентов**

База данных может использоваться для хранения информации о клиентах, а также для отслеживания их заказов. Это позволяет сотрудникам автосалона предоставлять клиентам более качественный сервис.

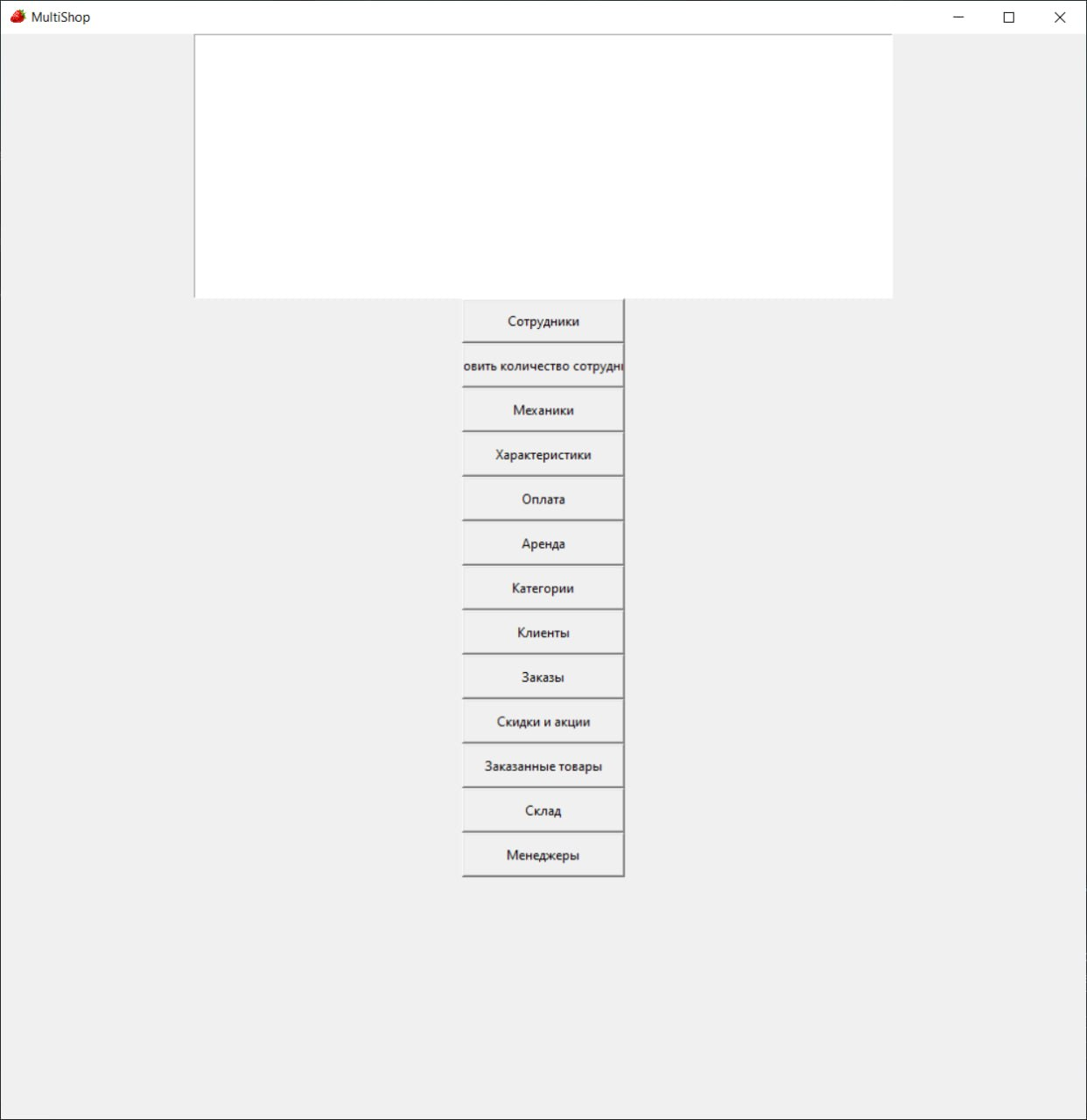
**Маркетинг**

База данных может использоваться для анализа данных о клиентах, а также для создания маркетинговых кампаний. Это позволяет автосалону повысить эффективность своих маркетинговых усилий.

Механизм безопасности. Это позволит защитить базу данных от несанкционированного доступа.

Средства резервного копирования и восстановления. Это позволит восстановить базу данных в случае ее повреждения или потери.

При добавлении этих возможностей база данных станет еще более эффективной и полезной для автосалона.



**Описание интерфейса базы данных для автосалона**

На изображении представлен интерфейс для базы данных автосалона. Интерфейс реализован на основе веб-технологий и позволяет пользователям выполнять различные операции с базой данных, такие как просмотр, добавление, изменение и удаление данных.

Интерфейс состоит из следующих элементов:

Меню - расположено в верхней части экрана и содержит ссылки на различные разделы интерфейса.

Панель инструментов - расположена под меню и содержит кнопки, которые позволяют пользователям выполнять различные операции с базой данных.

Список таблиц - расположен в центре экрана и отображает список таблиц, содержащихся в базе данных.

Форма - расположена в нижней части экрана и используется для ввода данных в базу данных.

Меню

Меню содержит ссылки на следующие разделы интерфейса:

Главная - содержит общую информацию о базе данных.

Таблица - позволяет просматривать, добавлять, изменять и удалять данные в таблицах базы данных.

Запросы - позволяет создавать и выполнять запросы к базе данных.

Отчеты - позволяет создавать отчеты на основе данных из базы данных.

Панель инструментов

Панель инструментов содержит следующие кнопки:

Новая таблица - позволяет создать новую таблицу в базе данных.

Новый запрос - позволяет создать новый запрос к базе данных.

Новый отчет - позволяет создать новый отчет на основе данных из базы данных.

Список таблиц

Список таблиц отображает список таблиц, содержащихся в базе данных. Для каждой таблицы в списке отображаются следующие сведения:

Название таблицы

Количество записей в таблице

Размер таблицы

Форма

Форма используется для ввода данных в базу данных. Форма содержит поля для ввода данных для каждого столбца в таблице.

Примеры использования интерфейса

Вот несколько примеров использования интерфейса:

Просмотр данных - чтобы просмотреть данные в таблице, выберите таблицу в списке таблиц и нажмите кнопку "Просмотреть".

Добавление данных - чтобы добавить данные в таблицу, выберите таблицу в списке таблиц и нажмите кнопку "Добавить". Затем введите данные в форму и нажмите кнопку "Сохранить".

Изменение данных - чтобы изменить данные в таблице, выберите таблицу в списке таблиц и нажмите кнопку "Изменить". Затем введите новые данные в форму и нажмите кнопку "Сохранить".

Удаление данных - чтобы удалить данные из таблицы, выберите таблицу в списке таблиц и нажмите кнопку "Удалить". Затем подтвердите удаление данных.

Создание запроса - чтобы создать запрос к базе данных, выберите раздел "Запросы" в меню и нажмите кнопку "Новый запрос". Затем введите текст запроса в форму и нажмите кнопку "Выполнить".

Создание отчета - чтобы создать отчет на основе данных из базы данных, выберите раздел "Отчеты" в меню и нажмите кнопку "Новый отчет". Затем выберите таблицу, на основе которой необходимо создать отчет, и нажмите кнопку "Создать".

Общие возможности интерфейса

Интерфейс имеет следующие общие возможности:

Поддержка различных типов баз данных - интерфейс поддерживает различные типы баз данных, включая MySQL, PostgreSQL, Oracle и Microsoft SQL Server.

Поддержка различных языков - интерфейс поддерживает различные языки, включая английский, русский, французский и немецкий.

Доступность - интерфейс доступен для пользователей с ограниченными возможностями.

**Заключение**

База данных для автосалона является важным инструментом, который может использоваться для повышения эффективности работы автосалона. Она позволяет хранить и обрабатывать информацию о различных объектах, связанных между собой, таких как автомобили, клиенты, заказы и сотрудники автосалона.

В данном проекте база данных была разработана на основе реляционной модели данных. Реляционная модель данных является эффективной моделью для хранения больших объемов данных. Она также позволяет выполнять различные операции над данными, такие как добавление, изменение, удаление и поиск данных.

В будущем базу данных можно улучшить, добавив следующие возможности:

Поддержку многопользовательского доступа. Это позволит нескольким пользователям одновременно работать с базой данных.

Механизм безопасности. Это позволит защитить базу данных от несанкционированного доступа. Средства резервного копирования и восстановления. Это позволит восстановить базу данных в случае ее повреждения или потери.

При добавлении этих возможностей база данных станет еще более эффективной и полезной для автосалона.