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

Заключение....................................................................................................11

# 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

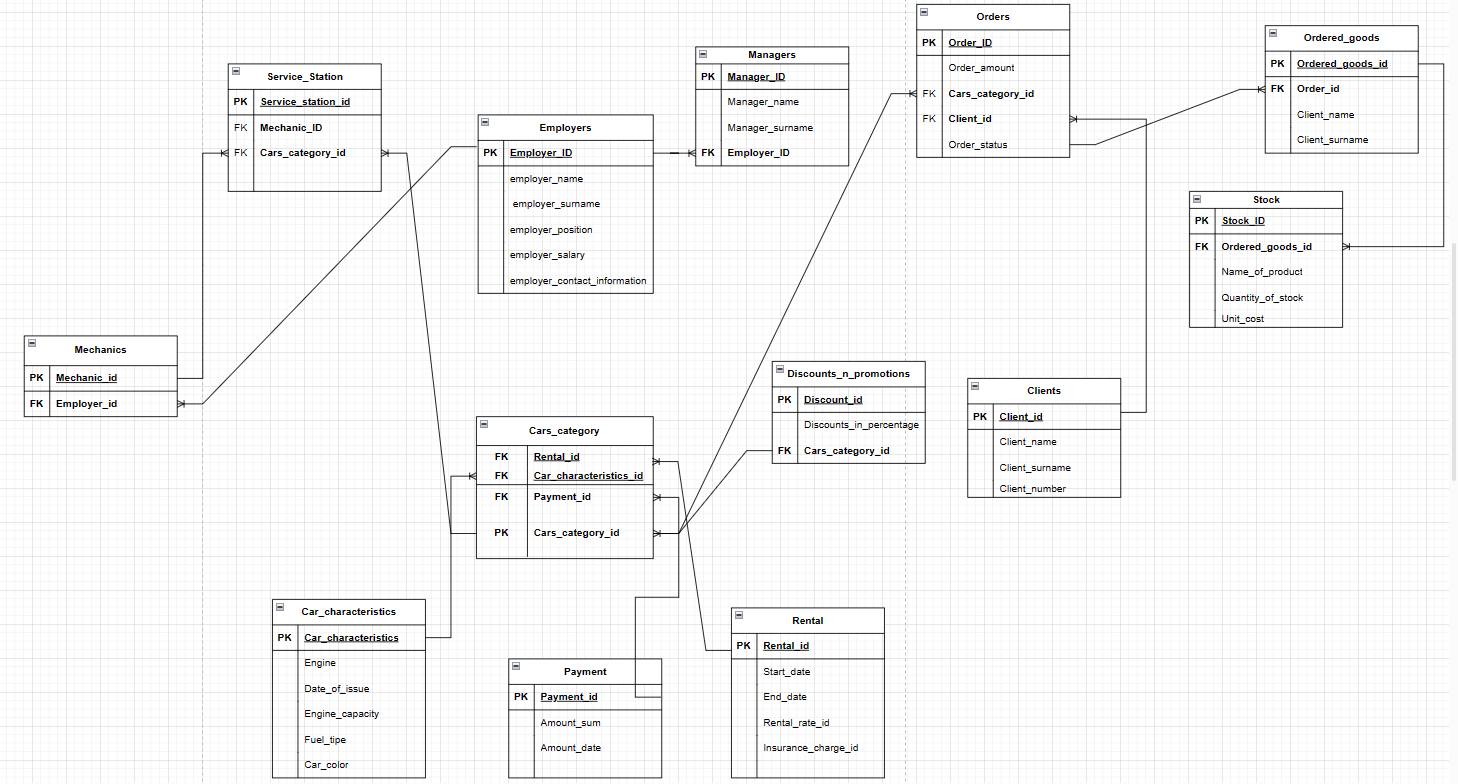
Цель производственного обучения заключалась в создании БД в PostgreSQL на основе выбранной темы, а также вывода данных с помощью языка Python

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

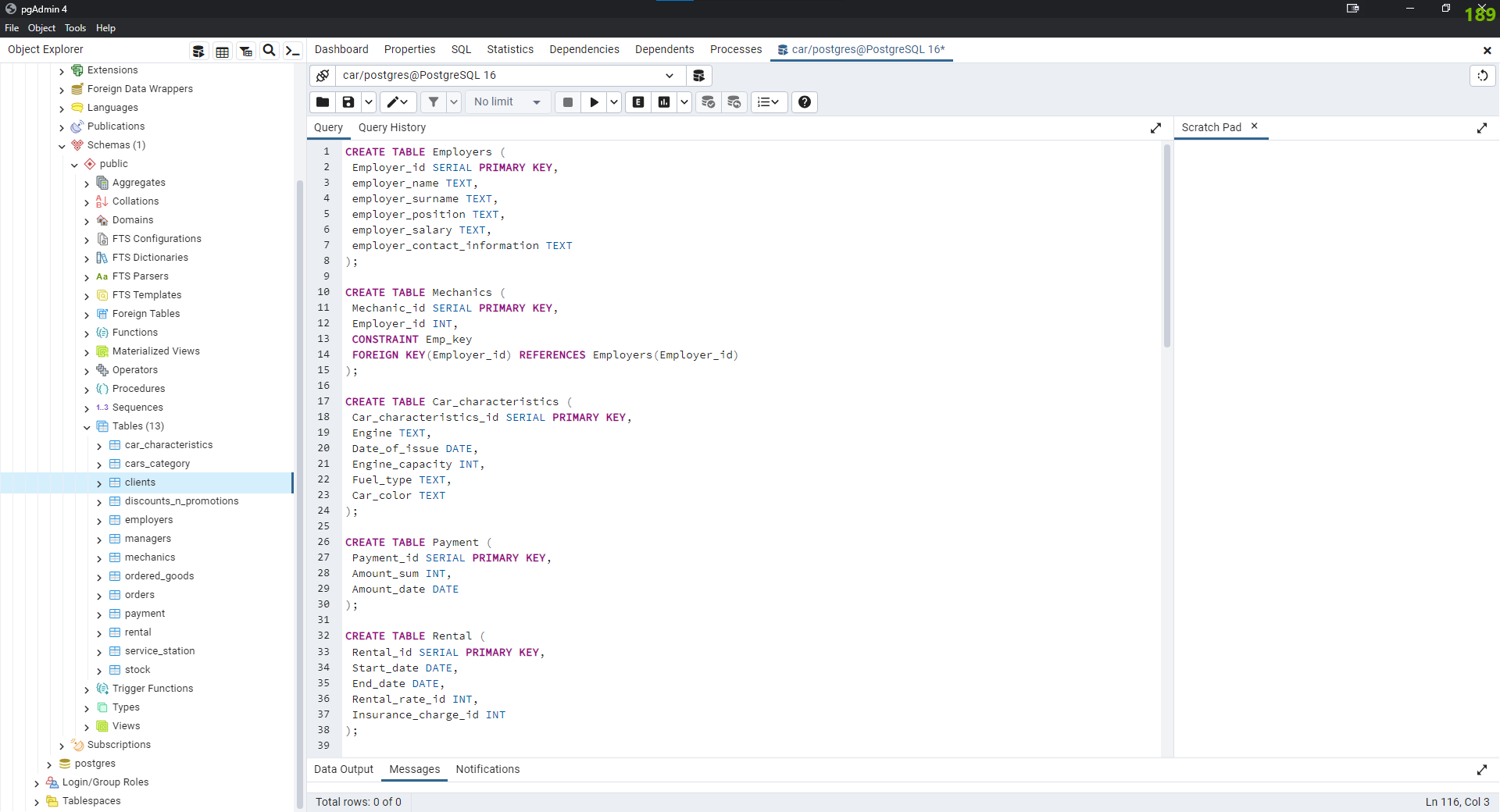
Имея схему нашей БД, мы приступили к созданию таблиц в PostgreSQL

Мы создали 13 таблиц в базе данных car\_rent с использованием команды SQL (CREATE TABLE). И наполнили их столбцами, присвоили типы данных в зависимости от столбца

Для обеспечения целостности данных, и установки связи, мы установили

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

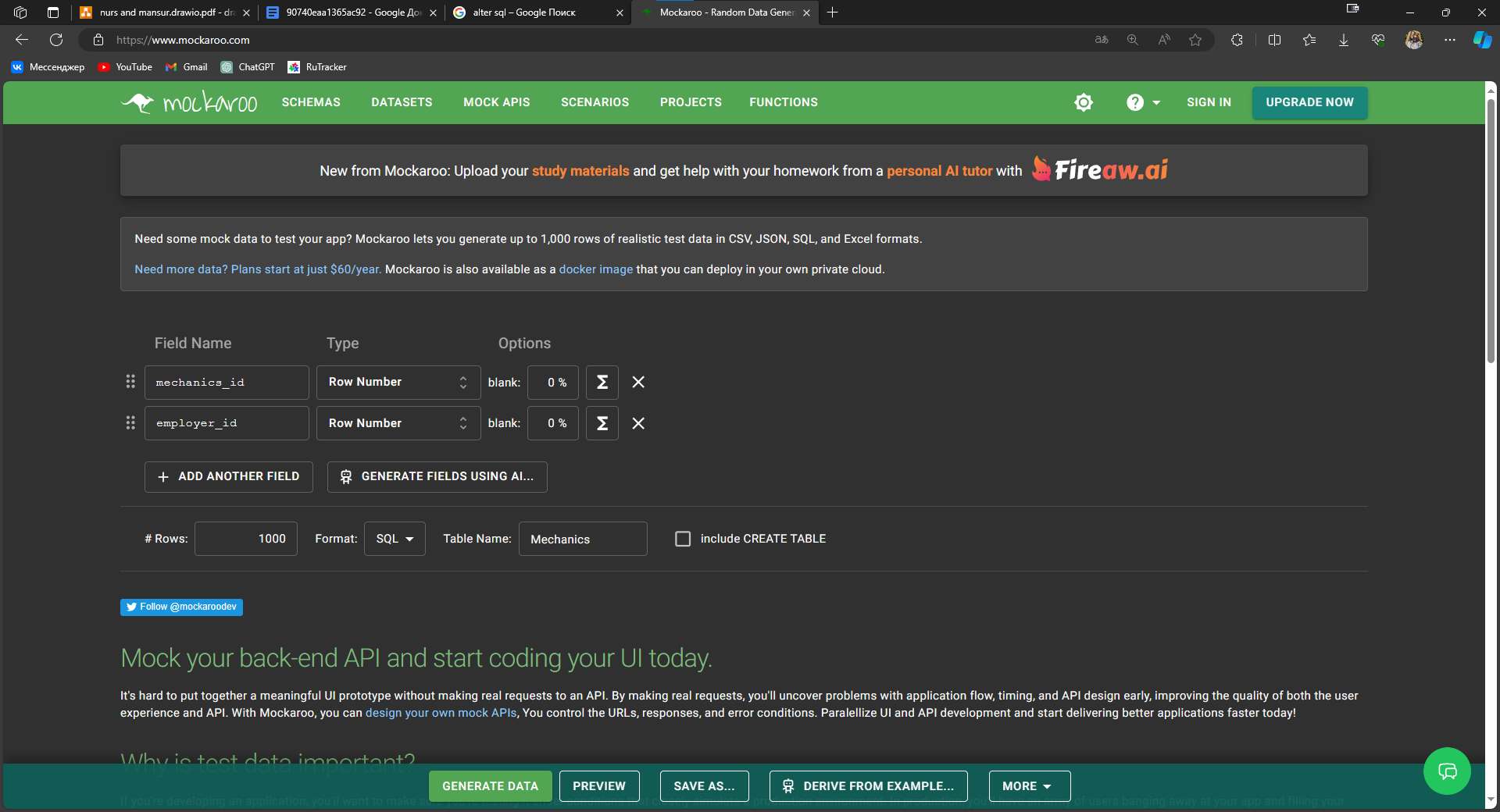
После генерации данных в Mockaroo, мы скачали их в sql формате,

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

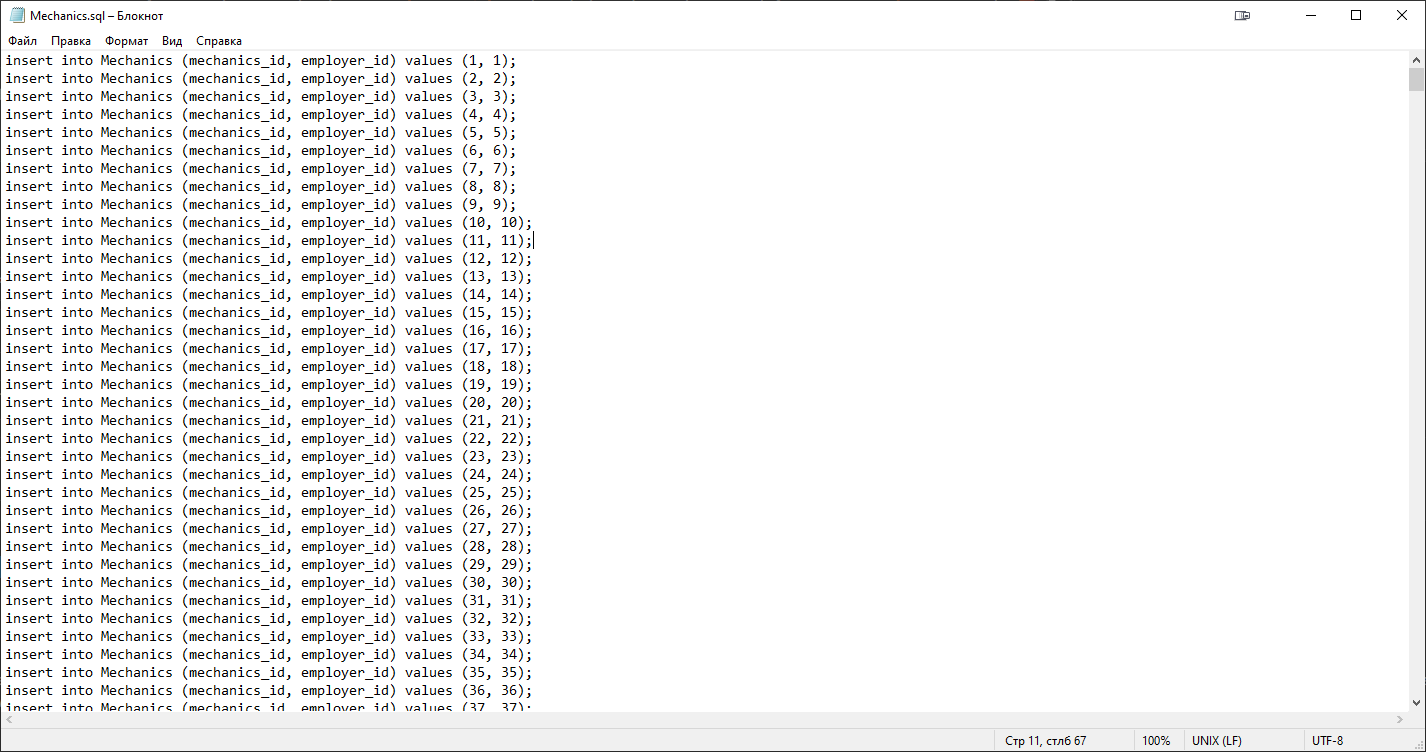
Mockaroo использует правильный синтаксис заполнения таблиц, поэтому проблем не возникло

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 (мы использовали Pycharm) нужно установить сам ЯП Python, затем мы установили библиотеку psycopg2, с помощью команды import, эта библиотека нужна для взаимодействия с PostgreSQL

Psycopg2

**Библиотека psycopg2 и структура кода:**

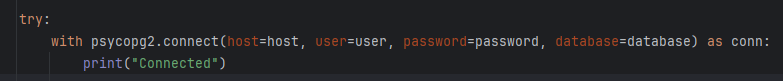
Мы выбрали данную библиотеку чтобы связать наш Python проект с БД

(Структурой Базой Данных):

**Объяснение кода**



Импорт параметров подключения из файла config:



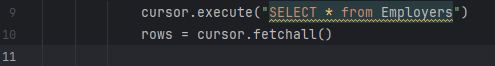
try используется для попытки установить соединение с базой данных и выполнить SQL-запрос

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 мы создаем курсор, который будет использоваться для выполнения SQL-запросов к базе данных.

****

Здесь мы создаем курсор, который будет использоваться для выполнения SQL-запросов к базе данных, затем запрашиваем 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.



Выводим 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")

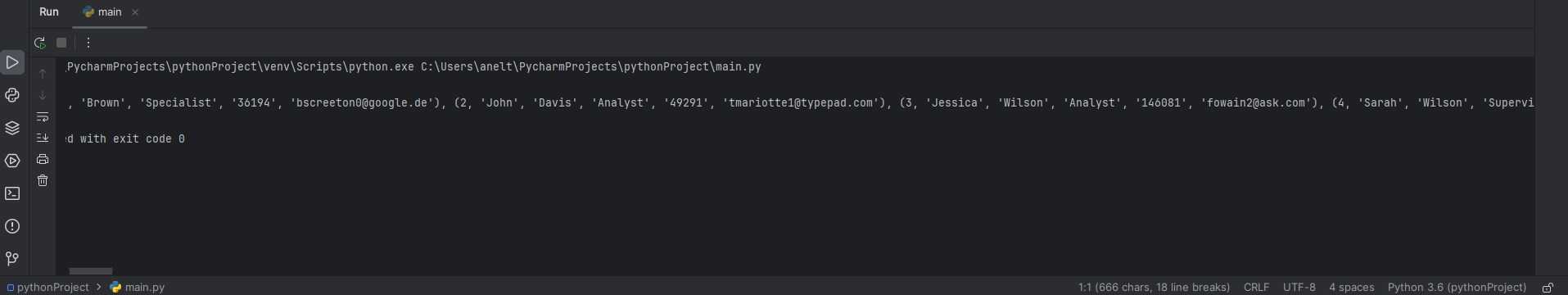
rows = cursor.fetchall()

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**  


**Заключение:**

В ходе выполнения задач по предмету "Организация обработки больших данных", мы успешно освоили ключевые аспекты работы с базами данных и использования программирования на Python.

В заключении, следует подчеркнуть следующие ключевые моменты:

Цель производственного обучения, заключавшаяся в создании БД и выводе данных с использованием Python, была достигнута успешно.