Департамент образования и науки города Москвы

Государственное автономное образовательное учреждение

высшего образования города Москвы

«Московский городской педагогический университет»

Институт цифрового образования

Департамент информатики управления и технологий

Кузьмина Дарья Юрьевна БД-241м

**Практическая работа 2 Моделирование данных и SQL для Data Engineering Вариант 11**

Направление подготовки/специальность

38.04.05 - Бизнес-информатика

Бизнес-аналитика и большие данные

(очная форма обучения)

Руководитель дисциплины:

Босенко Т.М., доцент департамента

информатики, управления и технологий,

доктор экономических наук

Москва

2025

Содержание

[Введение 2](#_Toc193670373)

[Основная часть 3](#_Toc193670374)

[Практические задания 11 вариант 15](#_Toc193670375)

[Заключение 19](#_Toc193670376)

## Введение

**Цель**

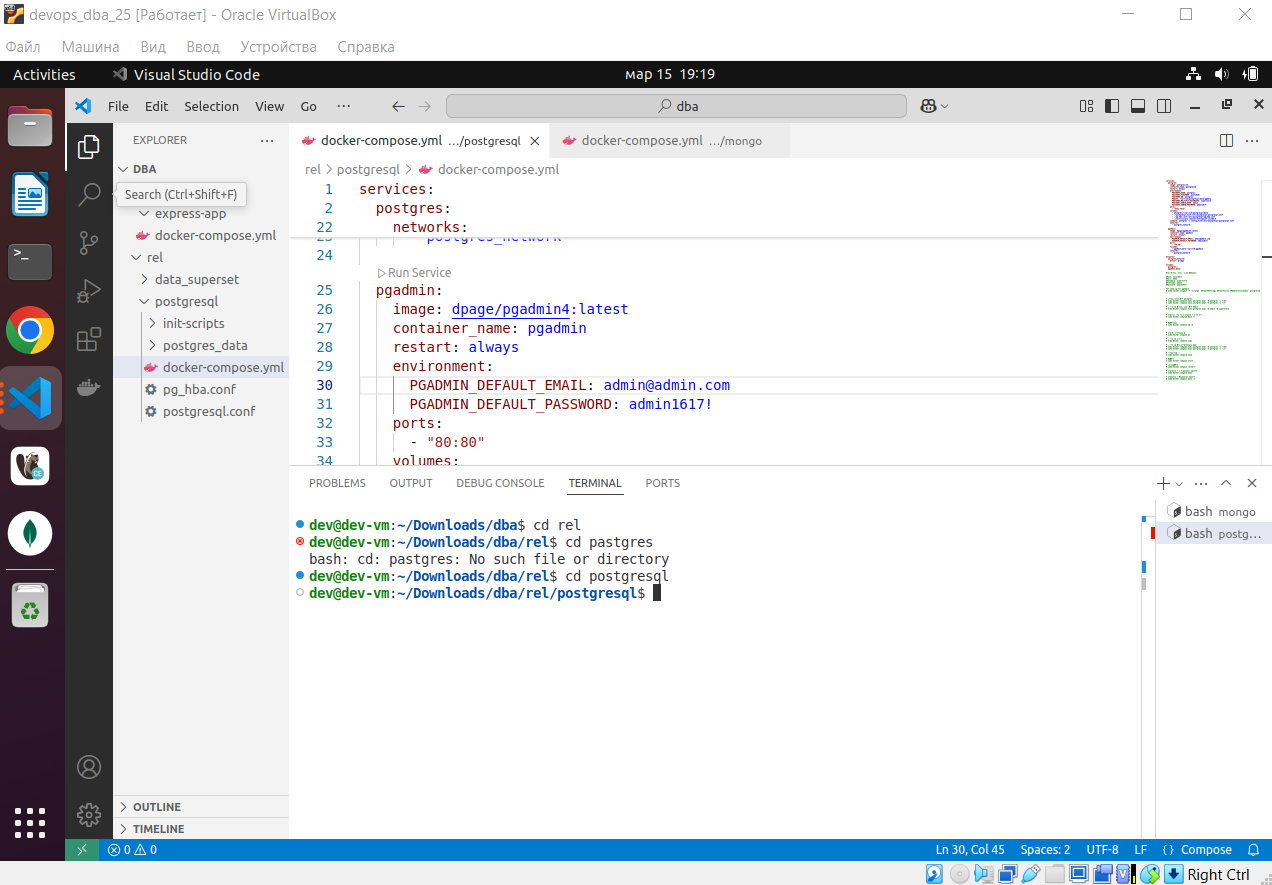
Цель: освоить принципы проектирования баз данных, создания структуры таблиц и загрузки данных в PostgreSQL.

**Задачи**

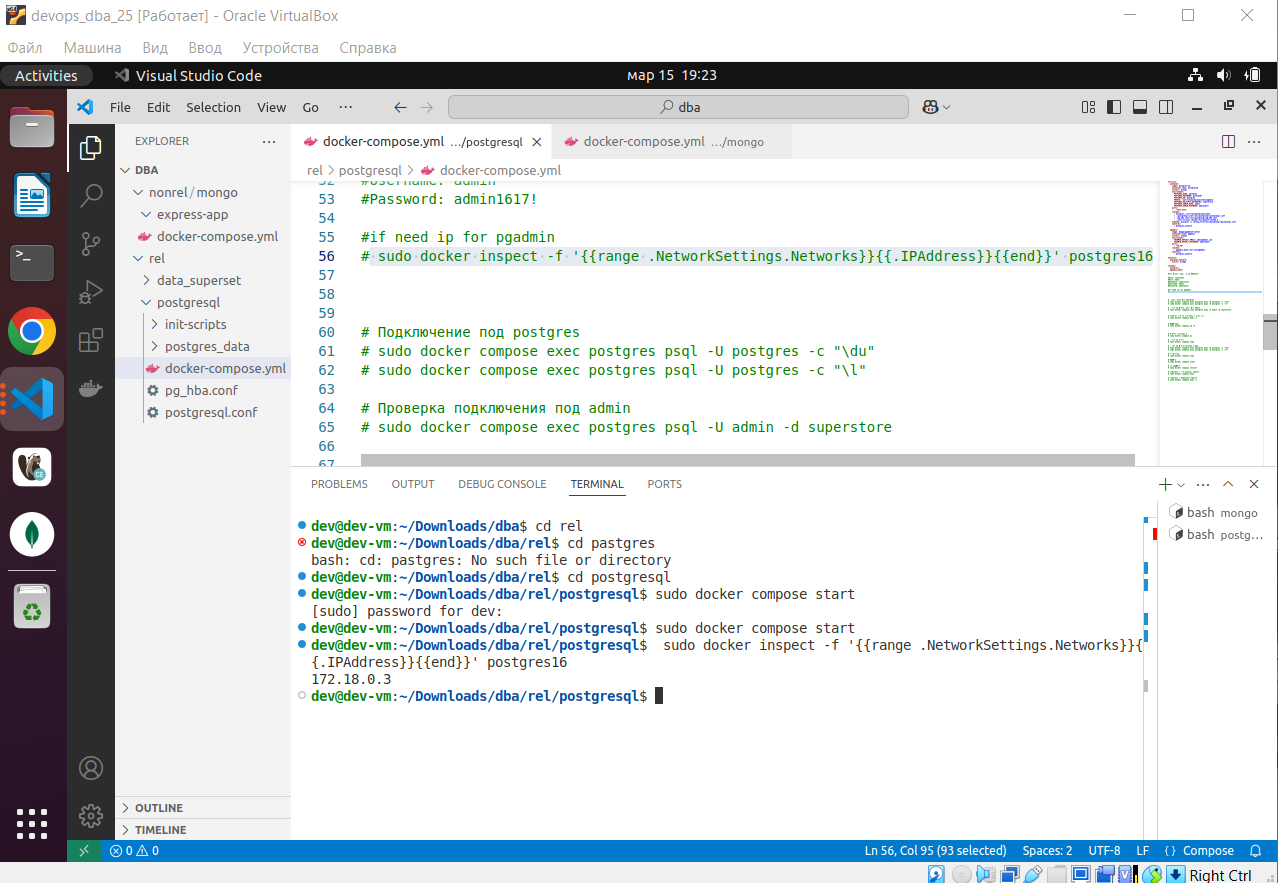
* Установить Postgres базу данных на компьютер. Инструкции по установки Postgres.
* Установить клиент SQL для подключения базы данных. Инструкции по установки DBeaver . Так же вы можете использовать любой другой клиент для подключения к ваше БД.
* Создать 3 таблицы и загрузите данные из Superstore Excel файл в базу данных. Сохраните в GitHub скрипт загрузки данных и создания таблиц. Вы можете использовать готовый пример sql файлов .
* Напишите запросы, чтобы ответить на индивидуальные задания. Сохраните в вашем GitHub скрипт загрузки данных и создания таблиц.
* Нарисовать модель данных для файла Superstore :
* Концептуальную.
* Логическую.
* Физическую. Вы можете использовать бесплатную версию SqlDBM, SQL Developer Data Modeler 24+ или любой другой софт для создания моделей данных баз данных.
* Скопировать DDL и выполнить его в SQL-клиенте.
* Необходимо сделать INSERT INTO SQL, чтобы заполнить Dimensions таблицы и Sales Fact таблицу. Сначала заполняем Dimensions таблицы, где в качестве id генерим последовательность чисел, а затем Sales Fact таблицу, в которую вставляем id из Dimensions таблиц.

## Основная часть

Зашли в папку



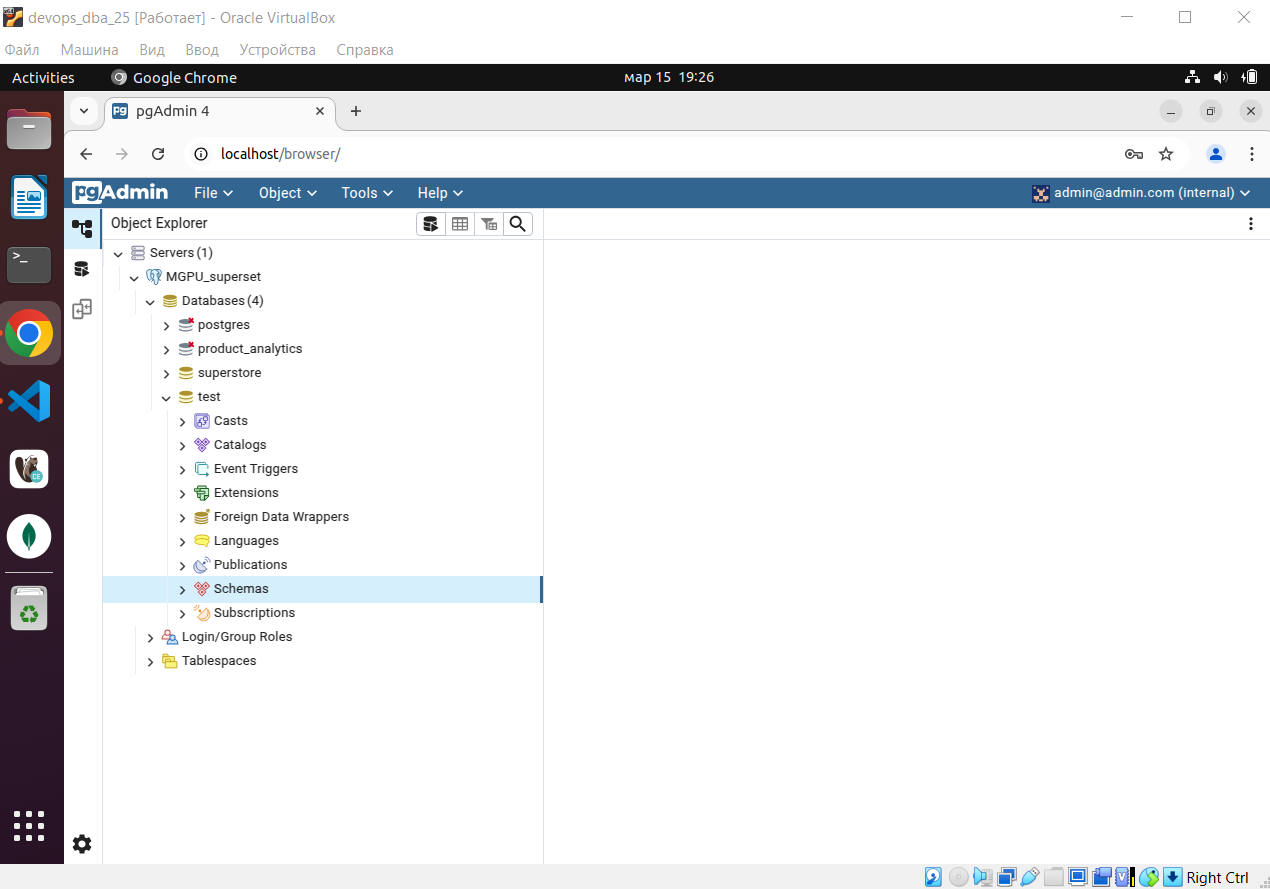
Запустила докер композер, проверила ip базы



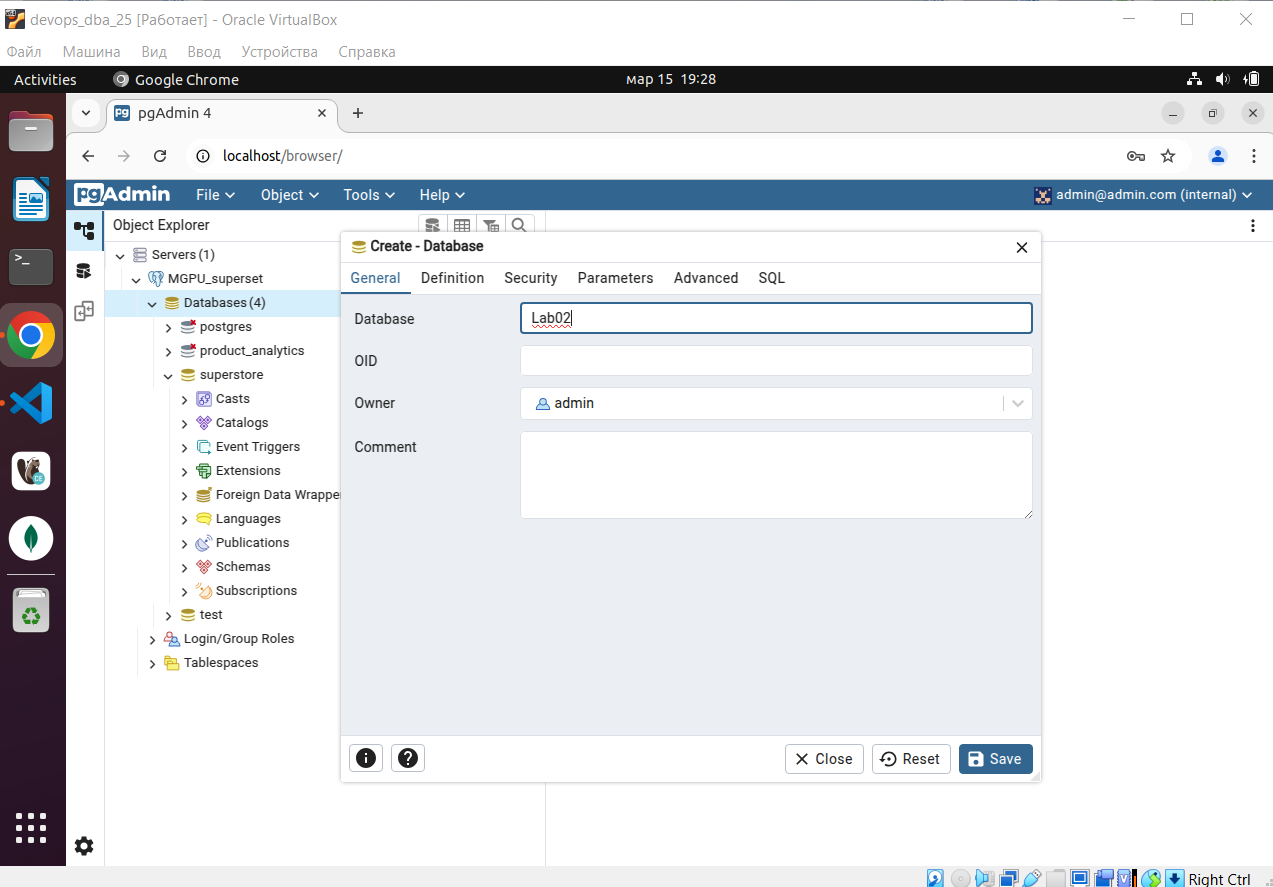
Вхожу в PGadmin



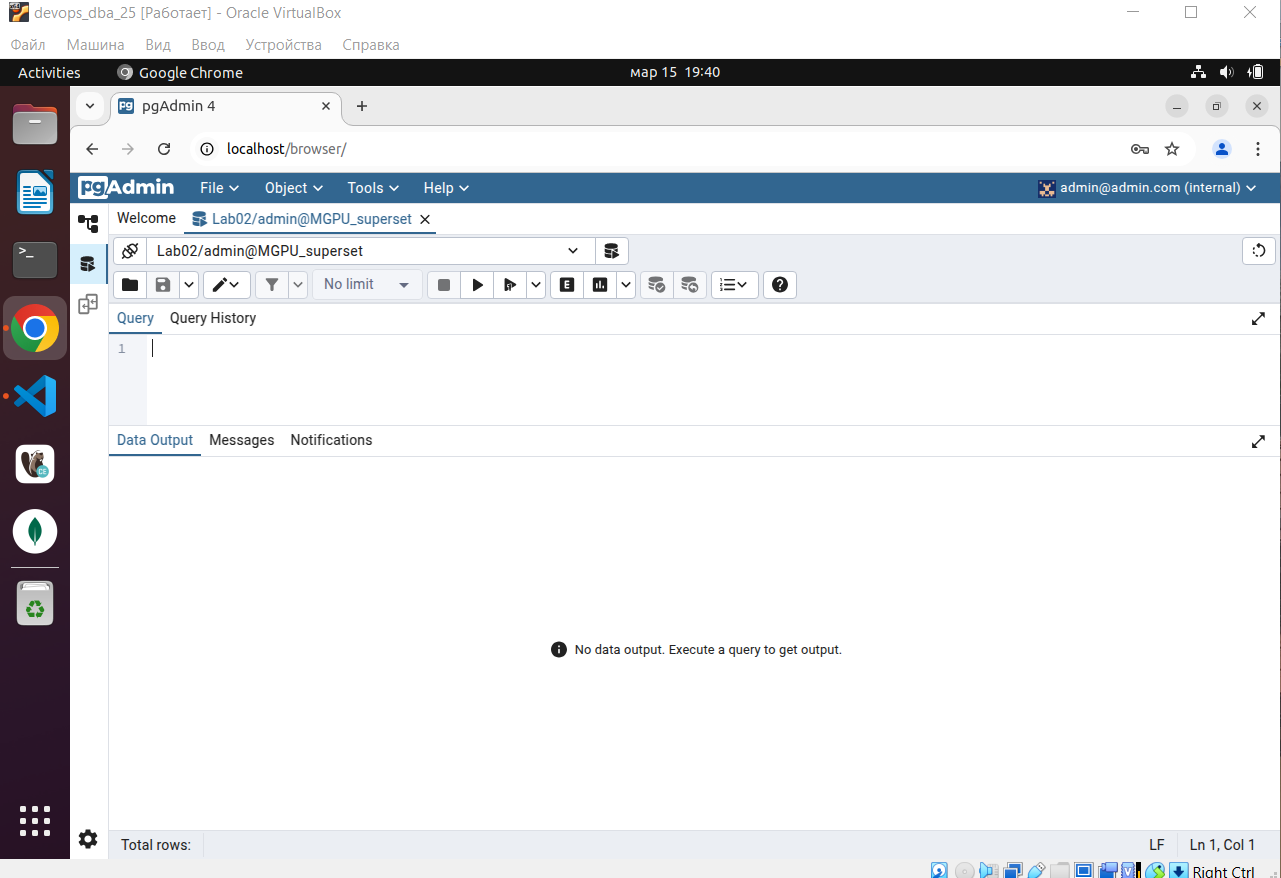
Подключилась к СУБД



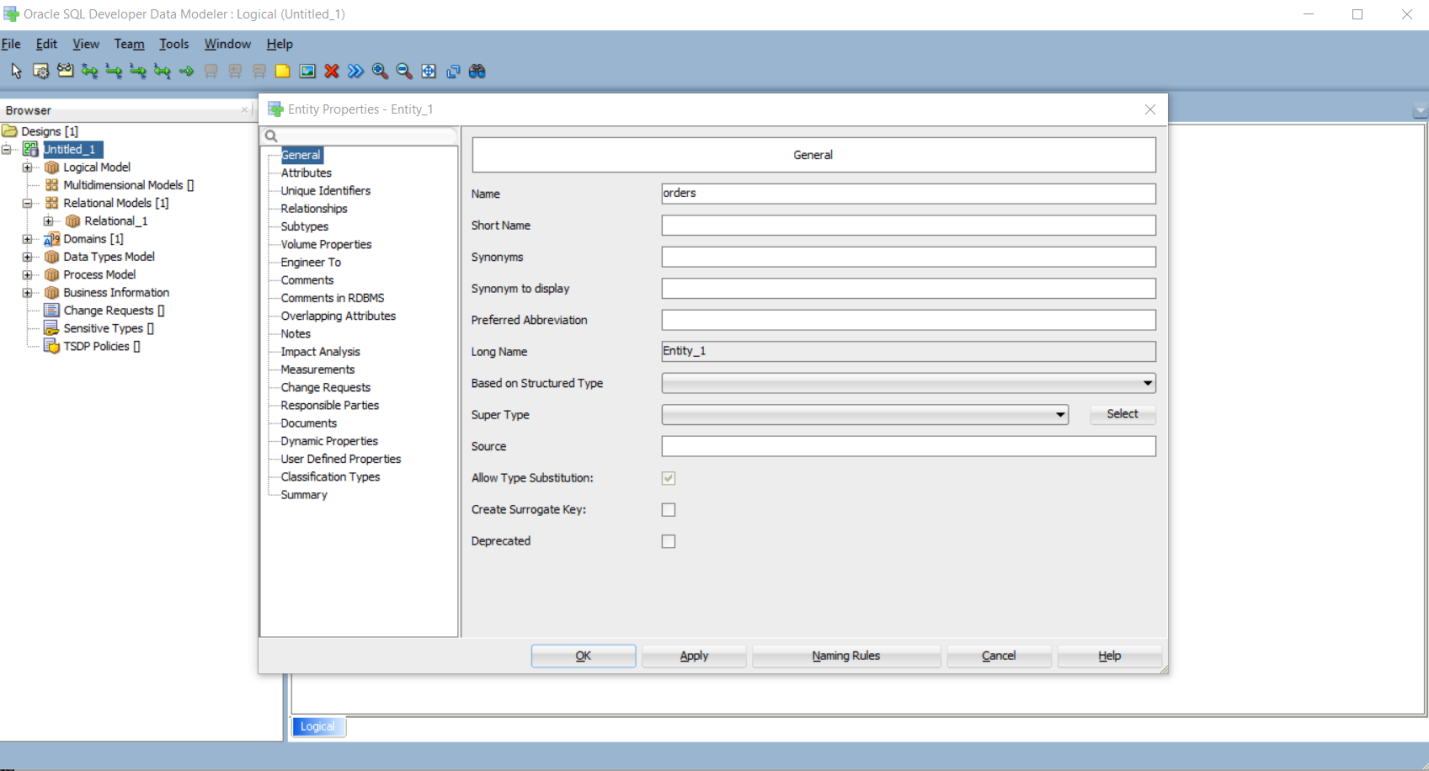
Создаем свою базу данных для работы



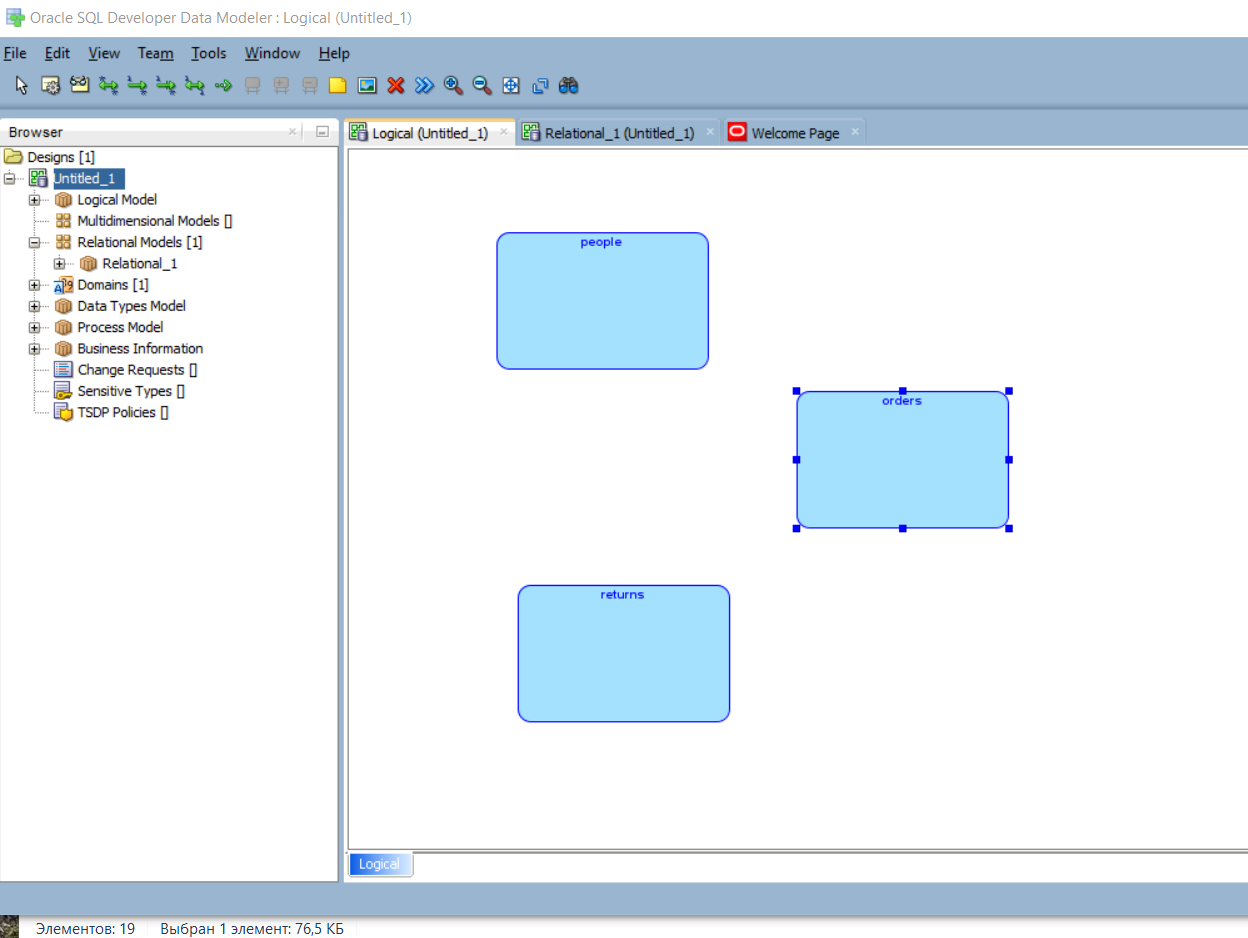
Открыла sql-окно, где будем делать запросы к базе данных



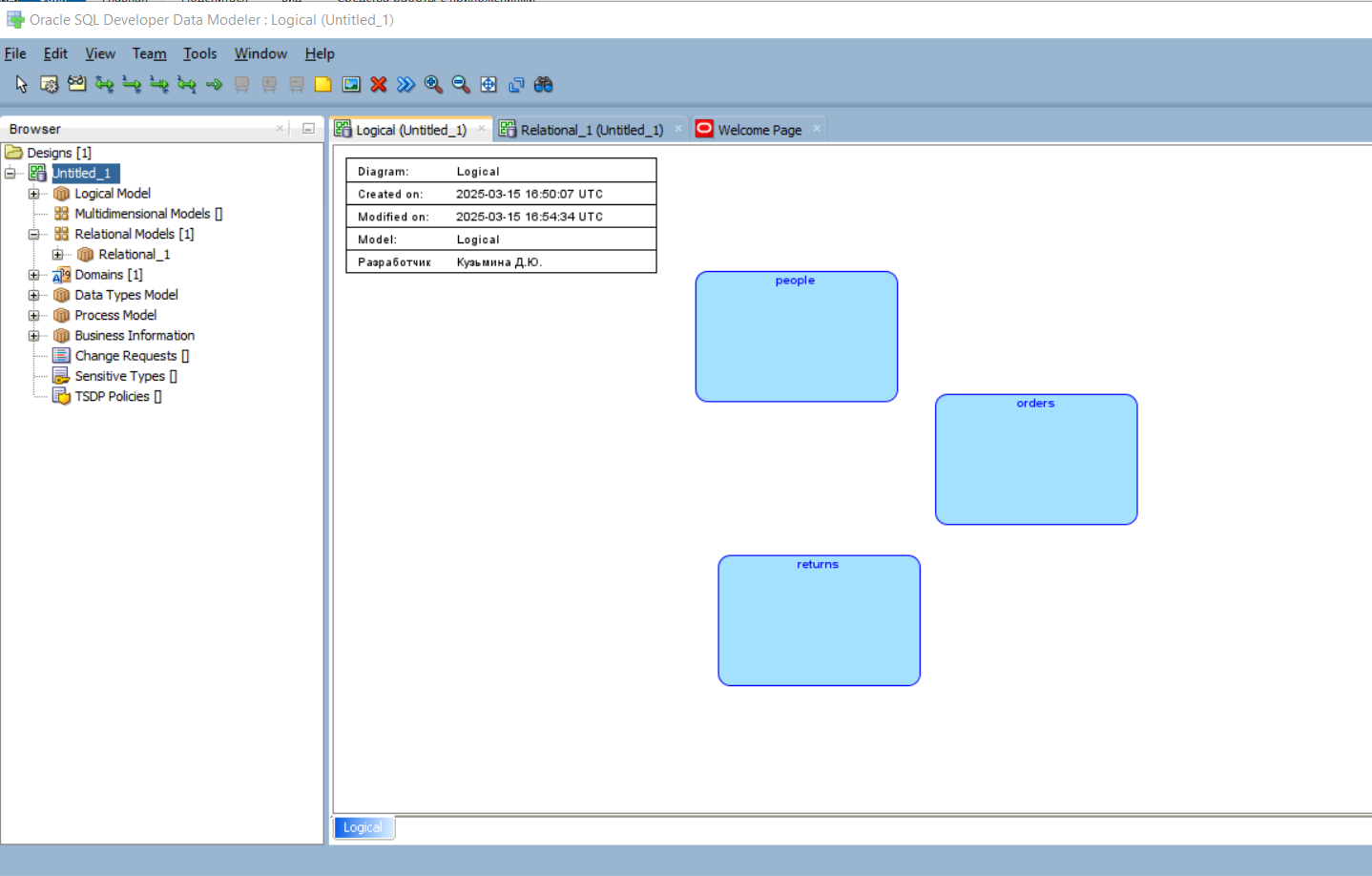
Создаем модель



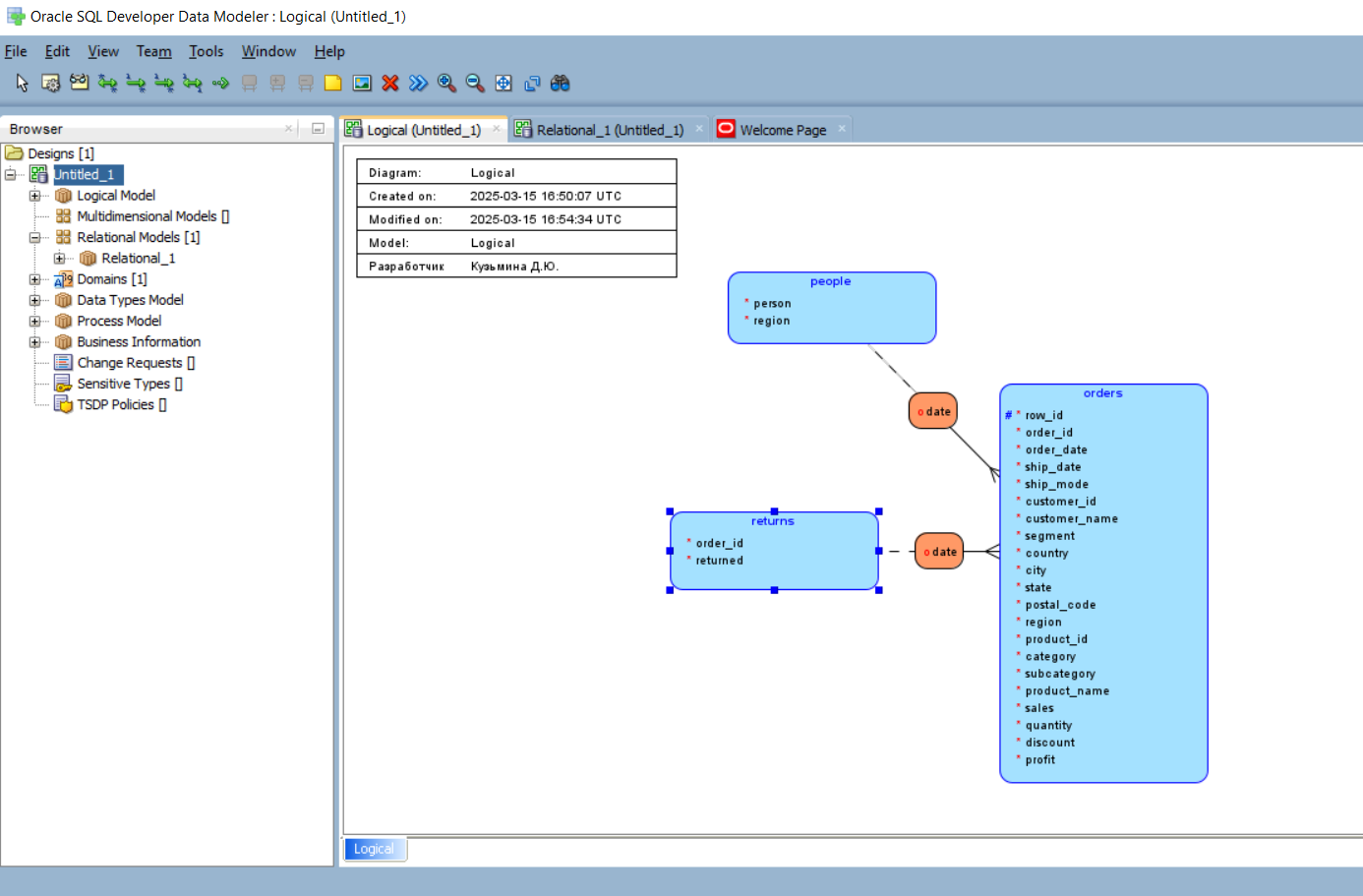
Создаем 3 сущности на основе внешнего источника данных superset excel



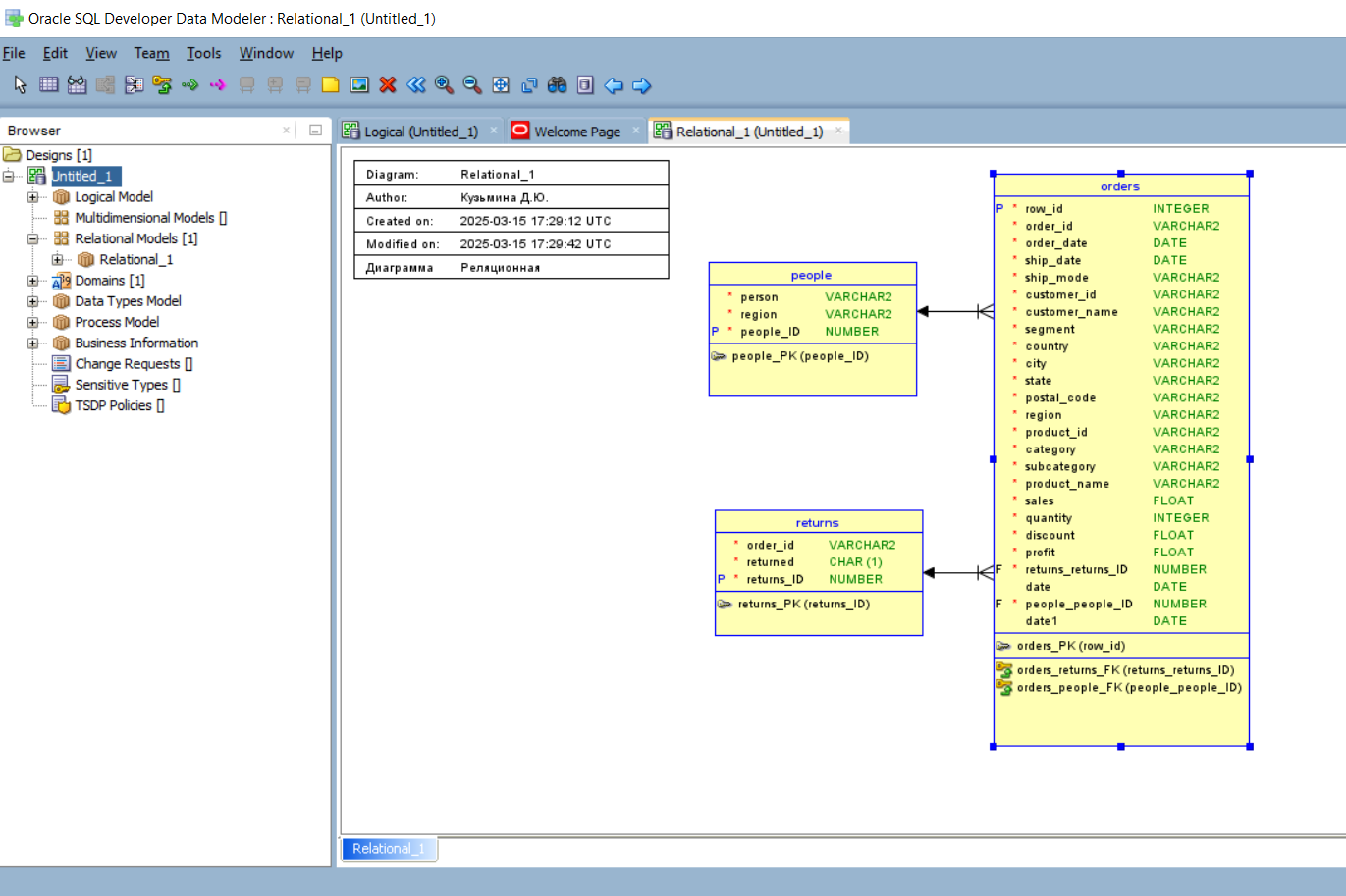
Настроим легенду



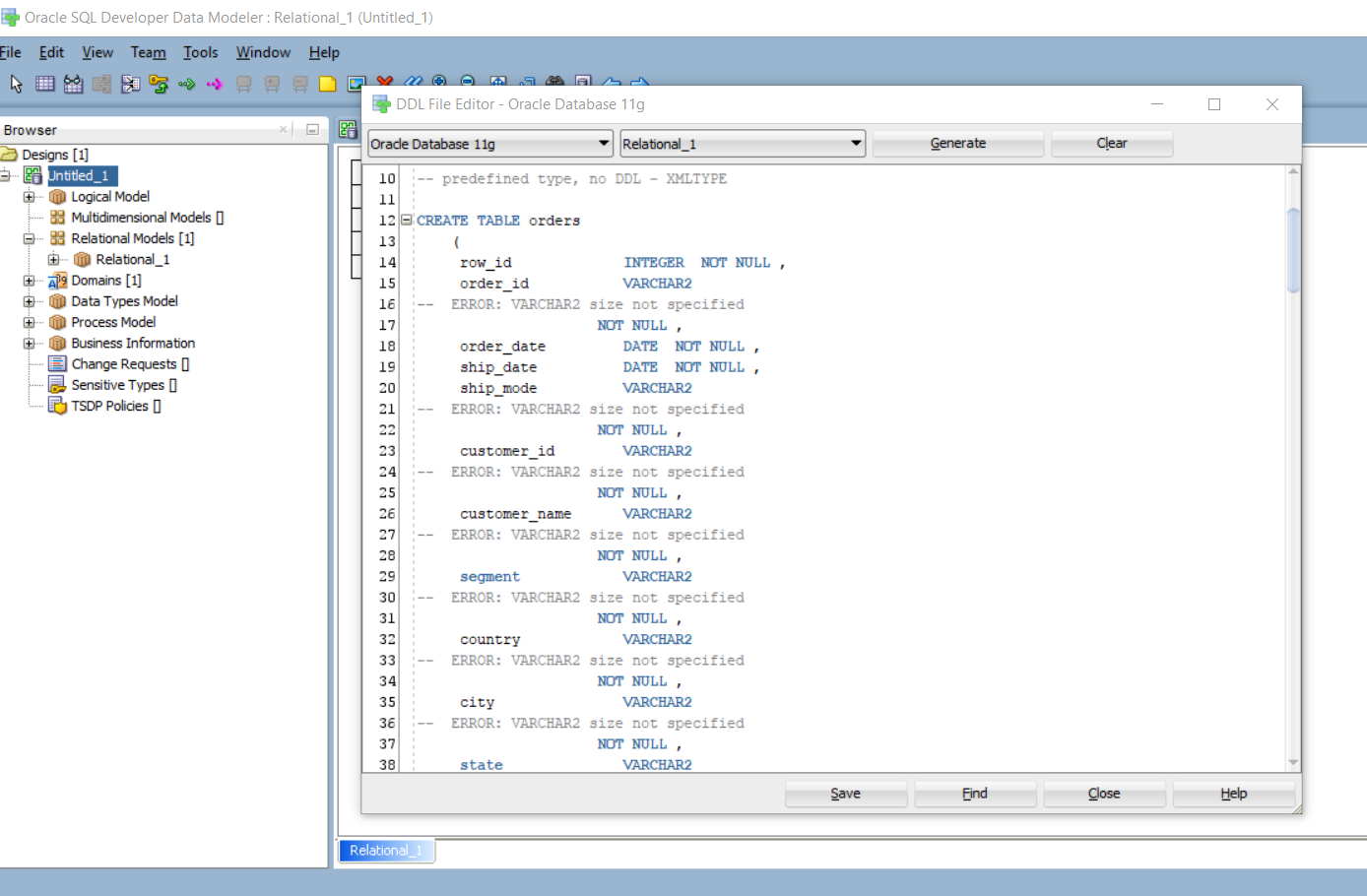
Добавим все поля в концептуальном проектировании



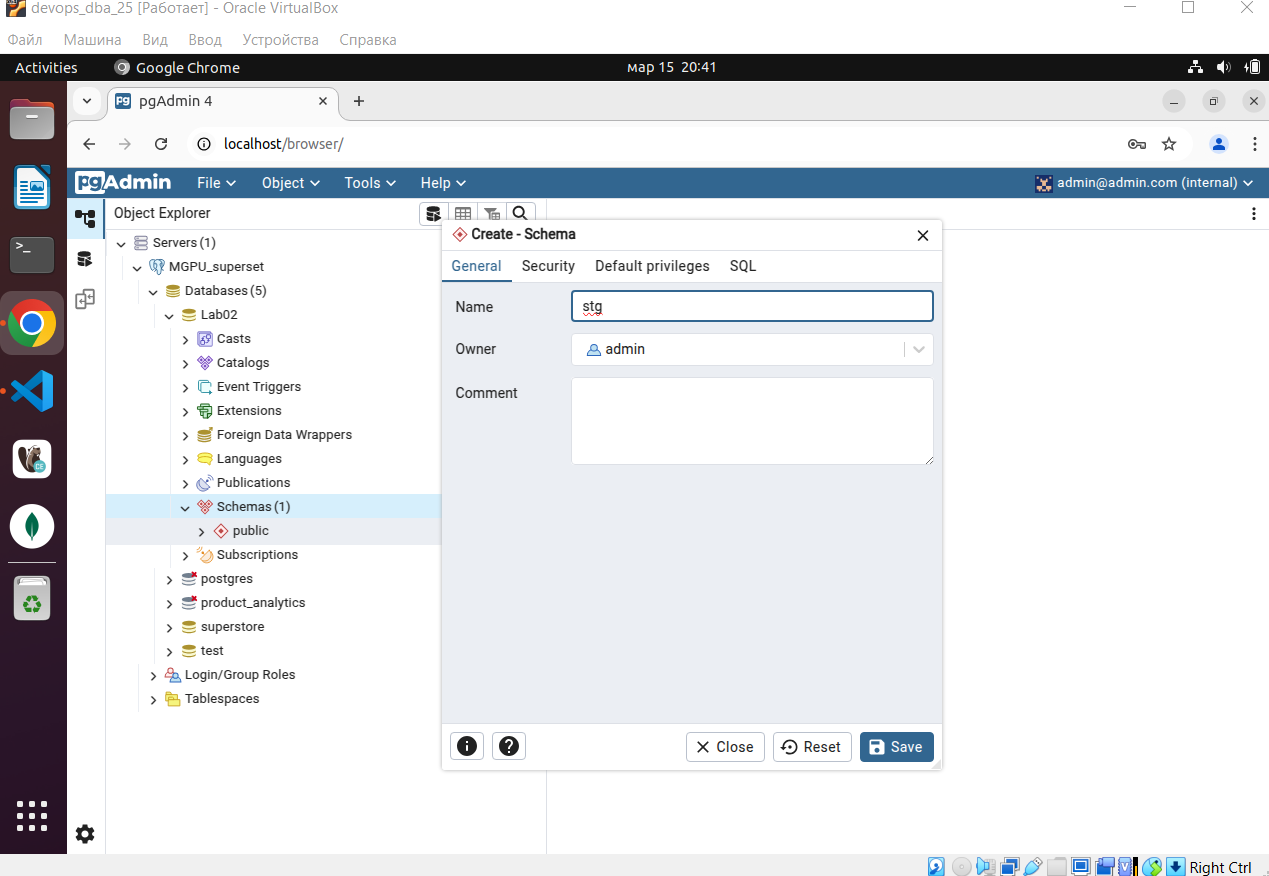
Перевела в реляционную



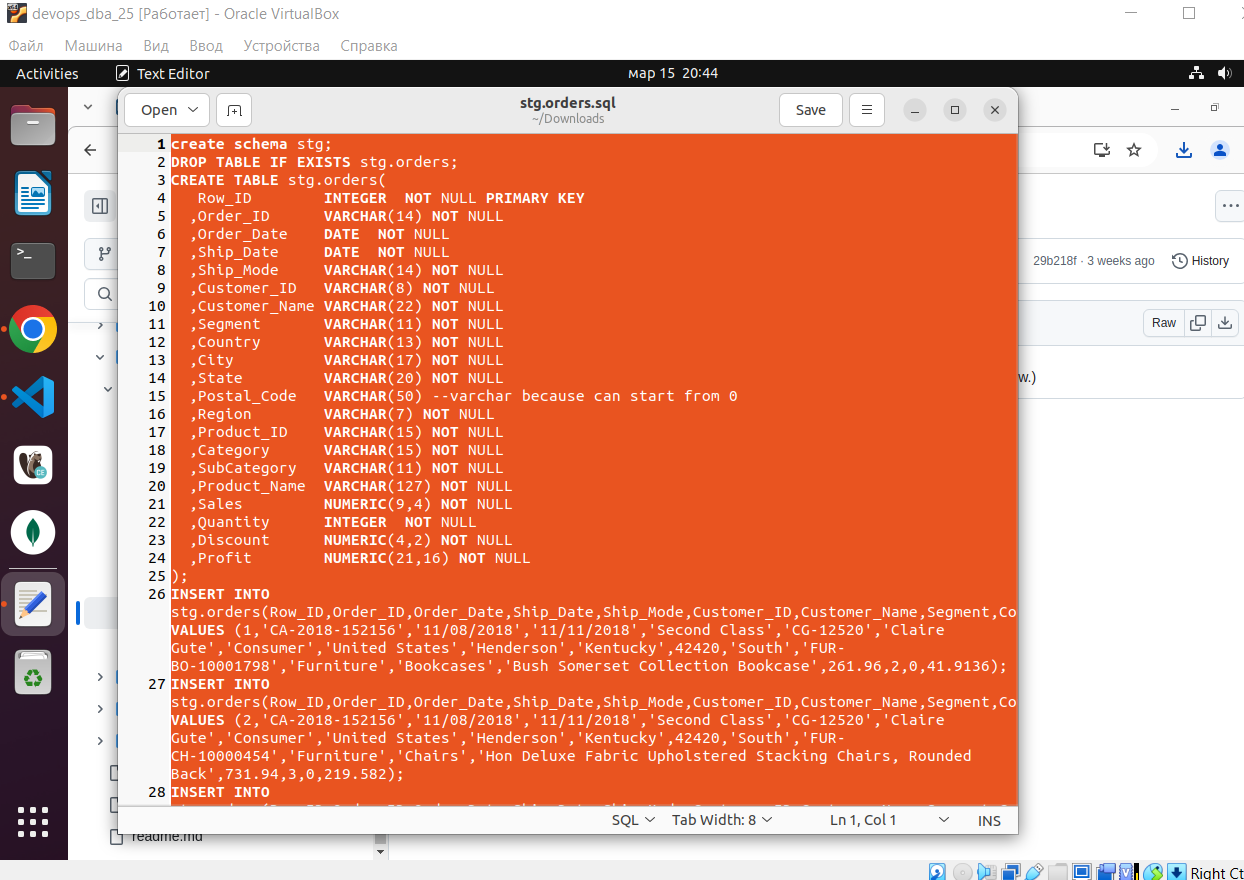
Создали ddl скрипт



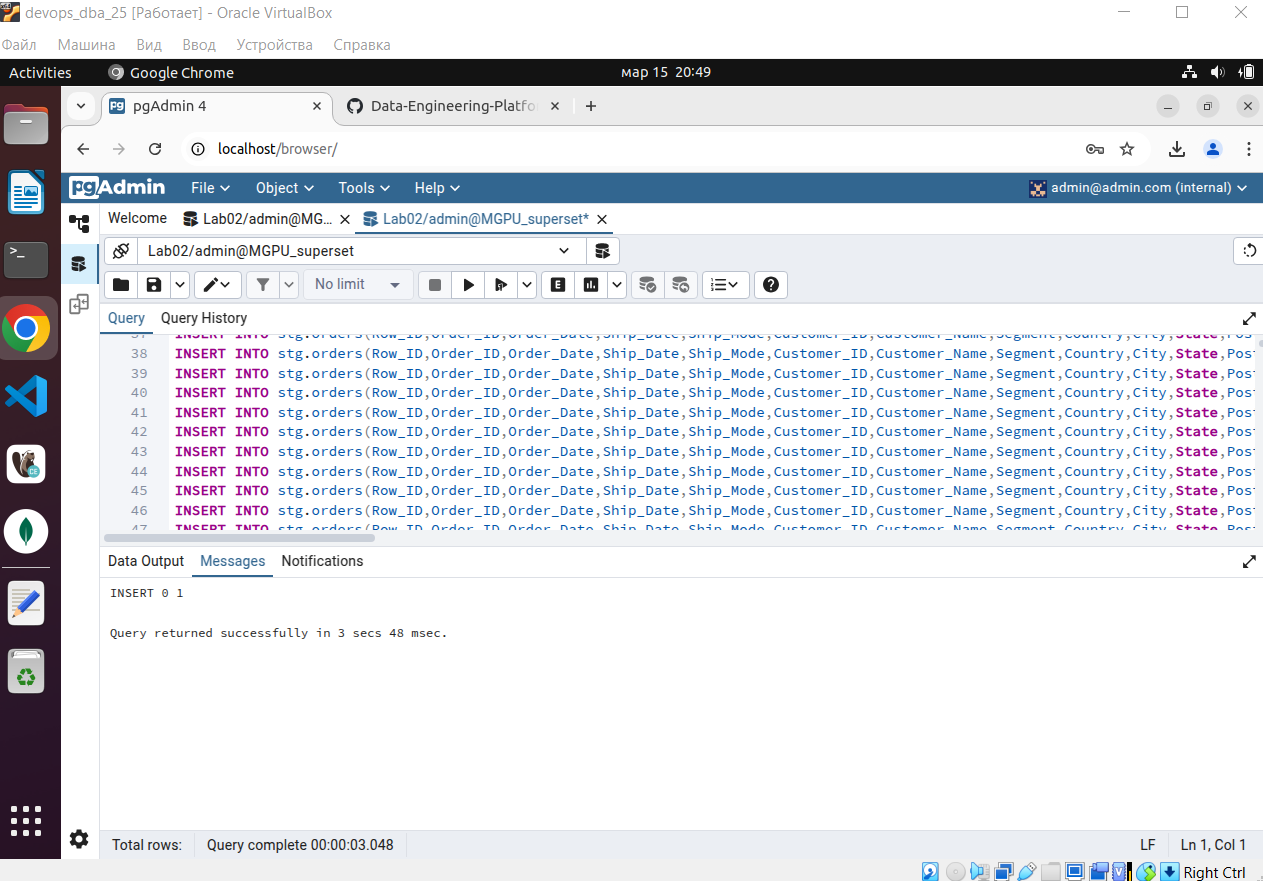
Создаем stg схему



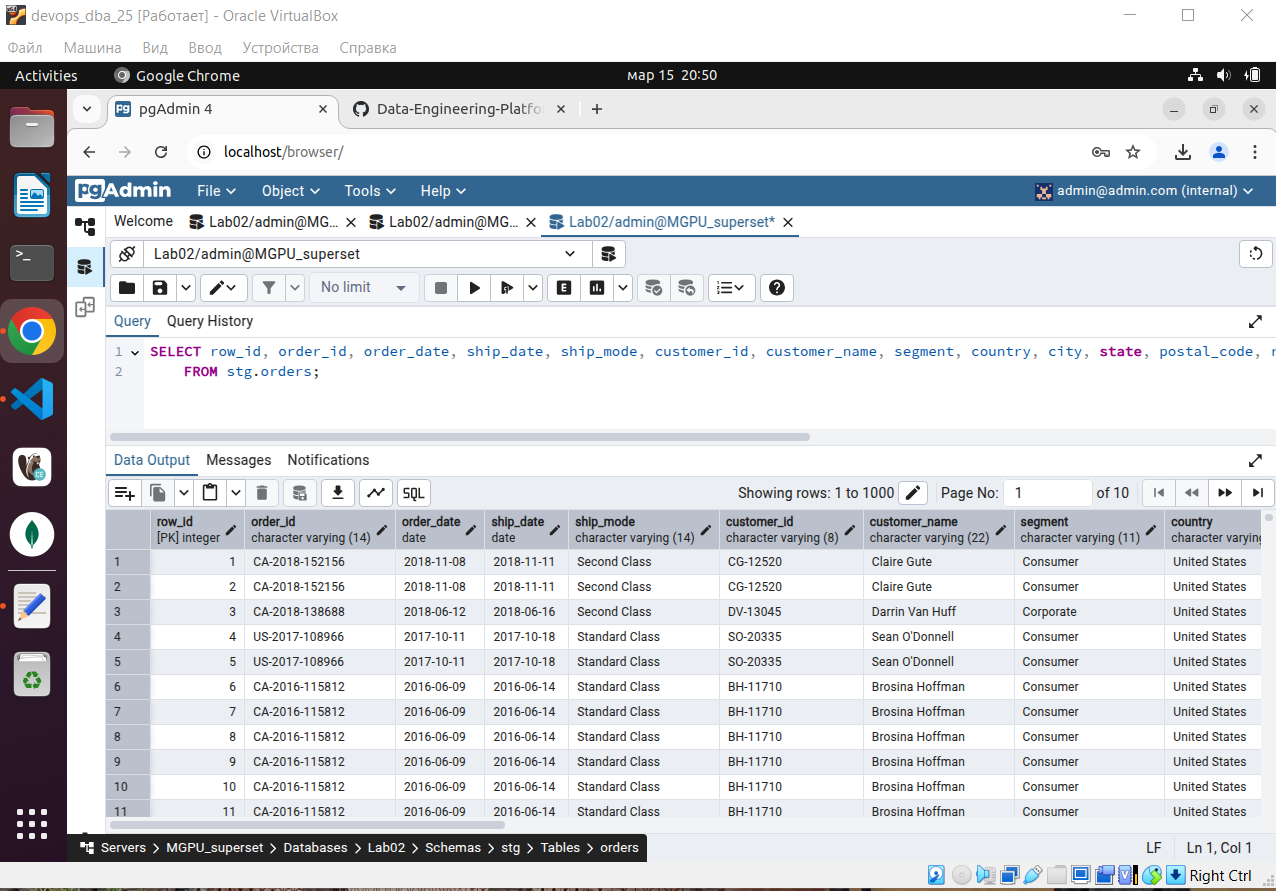
Открыли данные, закинули



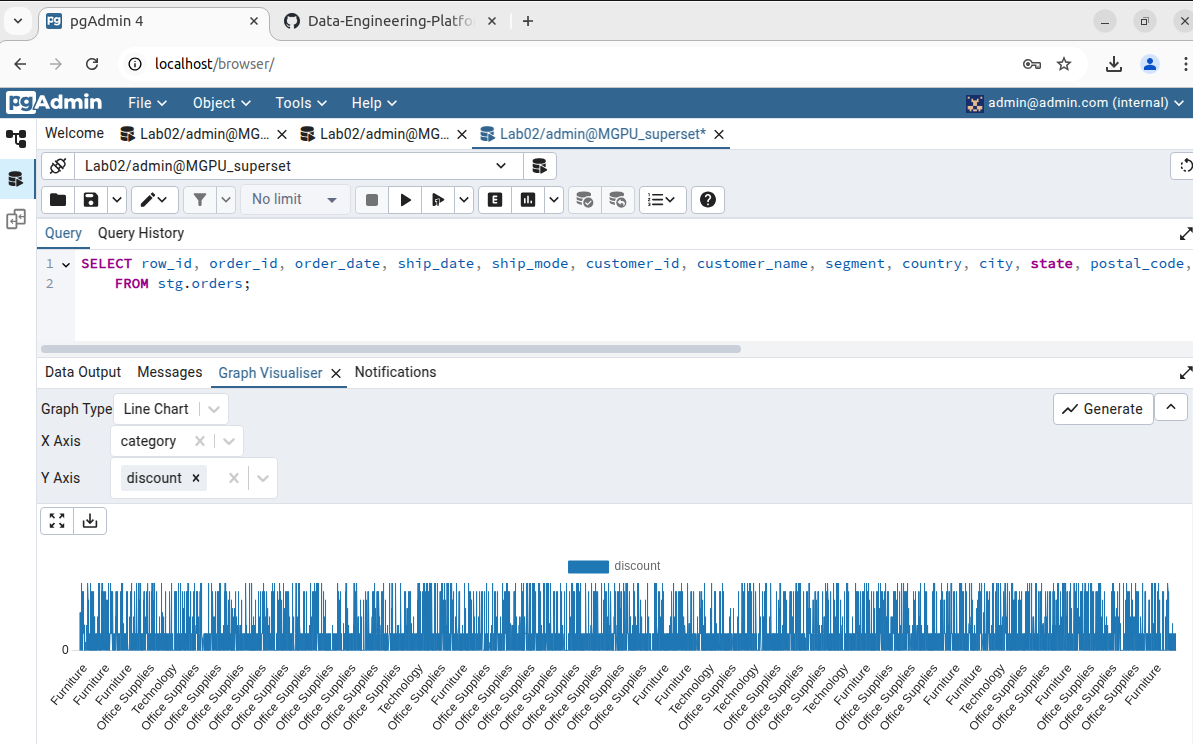
Загнали данные



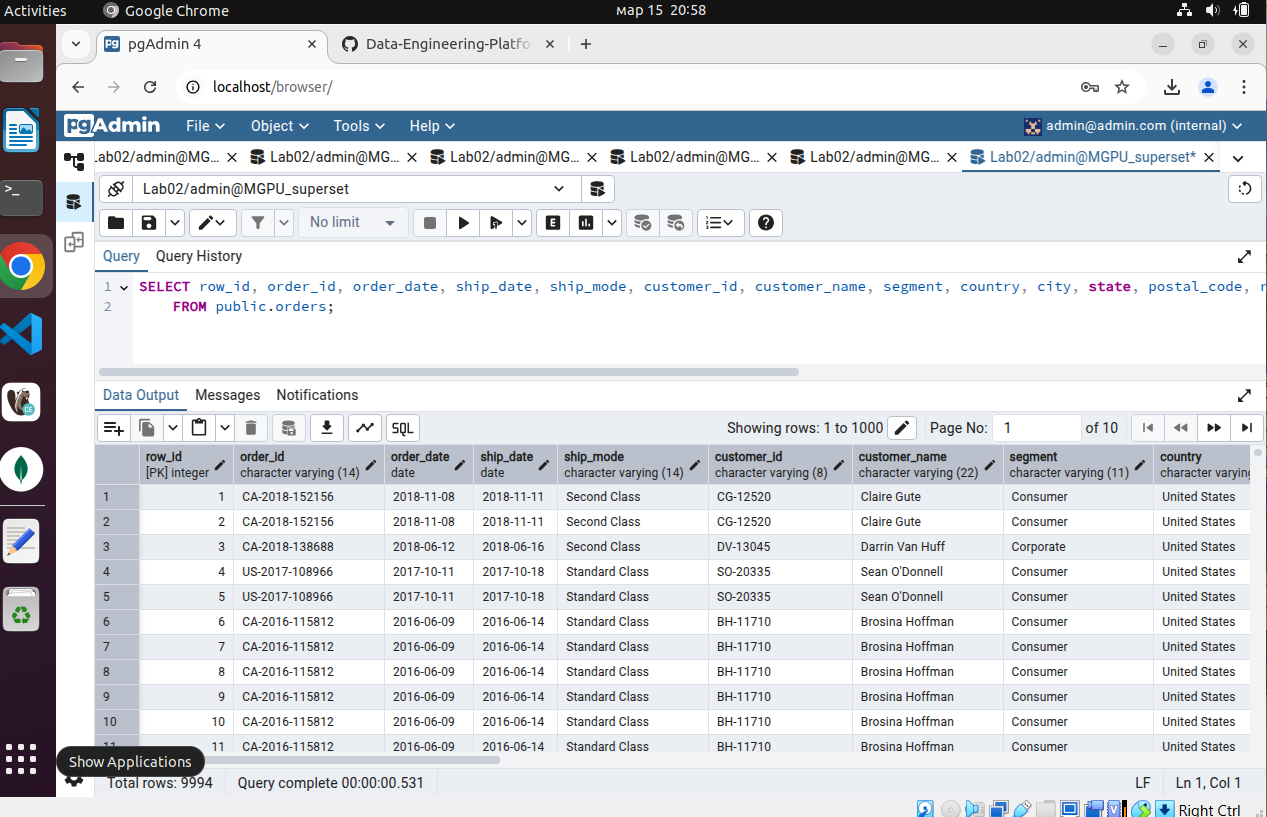
Введем запрос SELECT

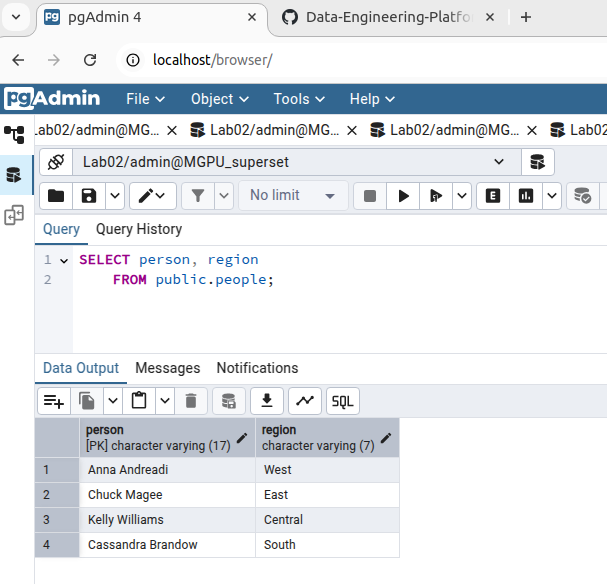


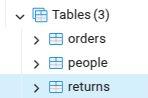
Сделали фильтрацию



Делаем так с остальными таблицами, вызываем SELECT

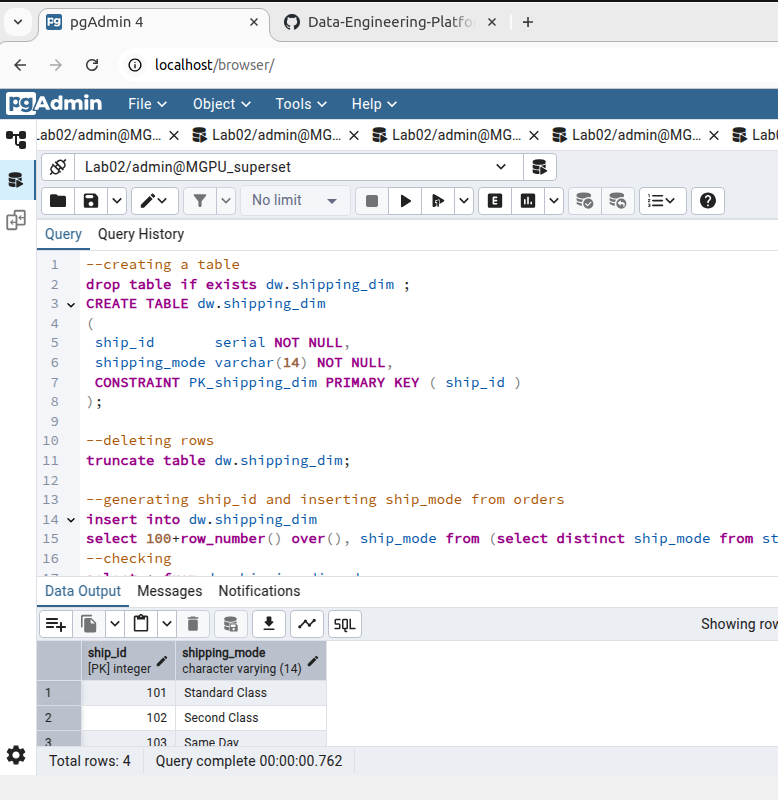




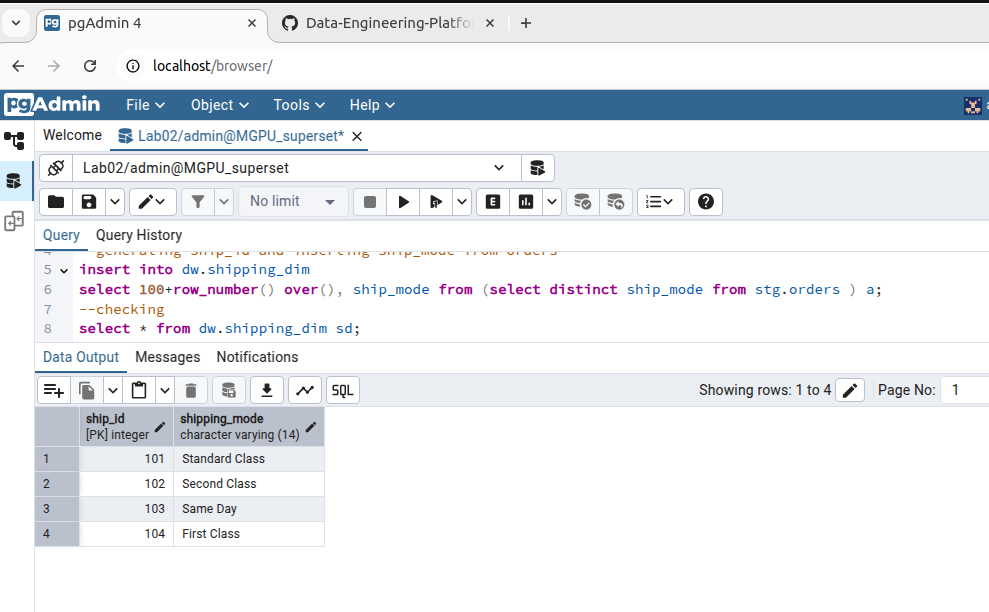


Подготовили аналитический слой, переходим к созданию слой для пользователей.

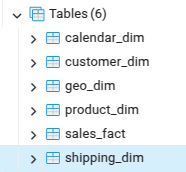
Саздаем таблицу shipping.dim

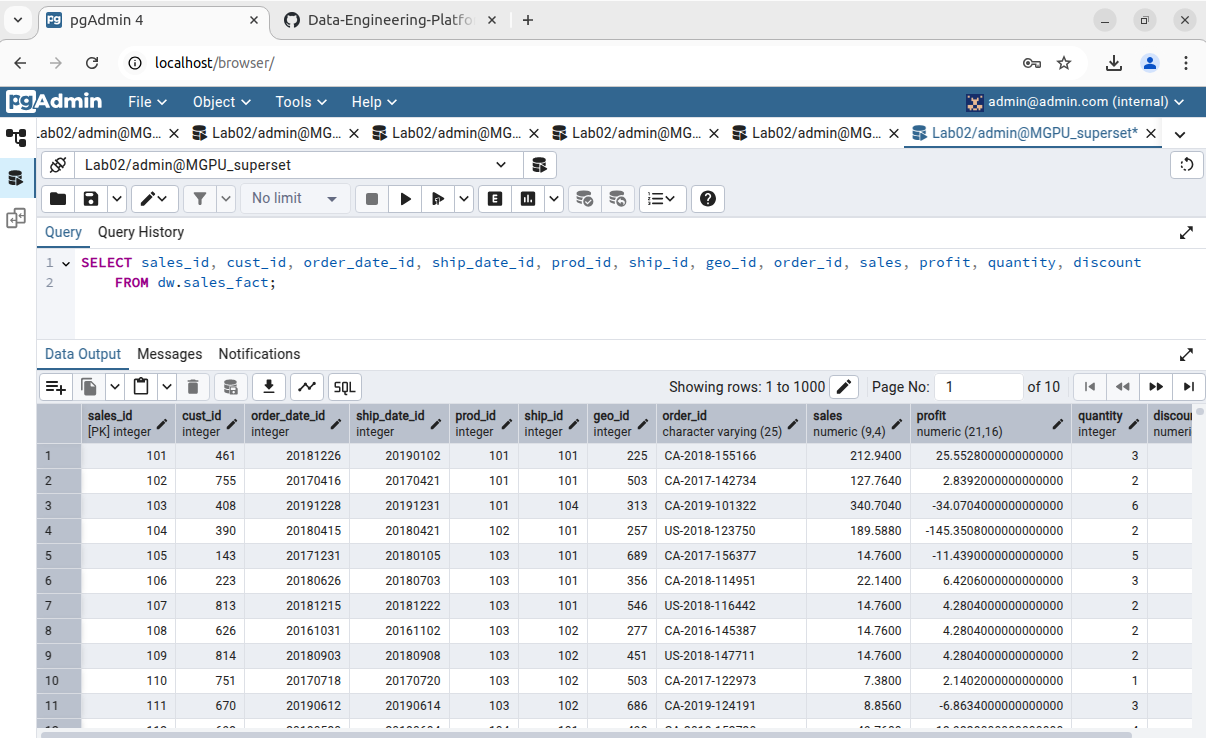


Получили результат выполнения предложенного скрипта



Создаем справочники и витрины





# Практические задания 11 вариант

##### **Задание 1: Создание представления по категориям**

Создаём представление sales\_by\_category, в котором группируем продажи по категориям товаров:

Описание логики:

- Запрос создаёт представление sales\_by\_category, которое агрегирует данные о продажах и прибыли по категориям товаров.

- Используется таблица sales\_fact для фактов продаж и таблица product\_dim для получения категорий товаров.

- Группировка выполняется по полю category, а агрегатные функции SUM вычисляют общие значения продаж и прибыли.

Обоснование выбора типов данных:

- Поле category имеет текстовый тип (`VARCHAR` или `TEXT`), так как содержит названия категорий.

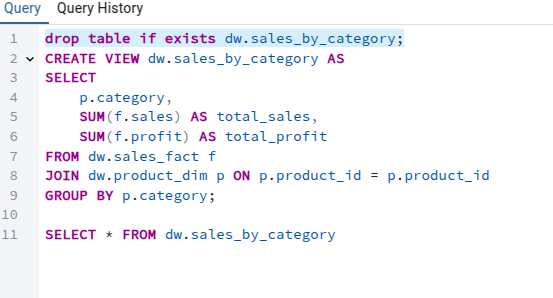
- Поля sales и profit используют числовой тип (`NUMERIC` или `DECIMAL`) для точного хранения финансовых данных.

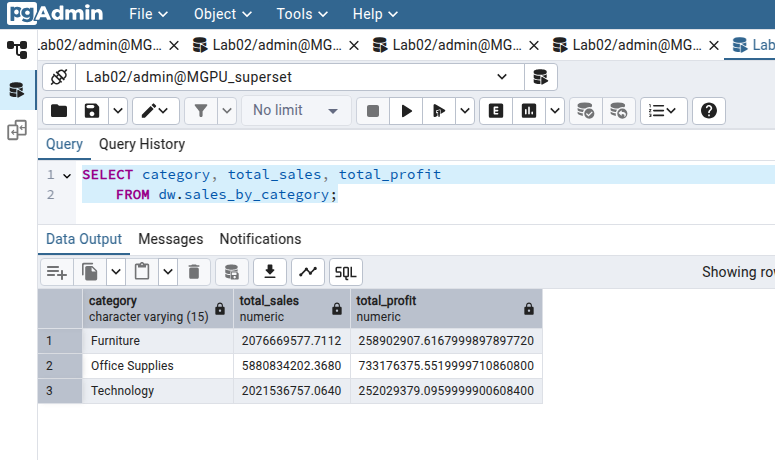
Использование индексов:

- Для ускорения выполнения запроса рекомендуется создать индекс на поле product\_id в таблицах sales\_fact и product\_dim.

Особенности реализации:

- Представление не хранит данные физически, а предоставляет актуальные данные при каждом запросе.





##### **Задание 2: Рассчитать возвраты по регионам**

Создаём таблицу returns\_by\_region, рассчитываем возвраты, связывая таблицы returns, orders и location\_dim:

Описание логики:

- Запрос создаёт таблицу returns\_by\_region, которая содержит количество возвратов по регионам.

- Данные извлекаются из таблиц returns, orders и location\_dim, связываемых по ключам order\_id и customer\_id.

- Группировка выполняется по полю region, а функция COUNT подсчитывает количество возвратов.

Обоснование выбора типов данных:

- Поле region имеет текстовый тип (`VARCHAR` или `TEXT`), так как содержит названия регионов.

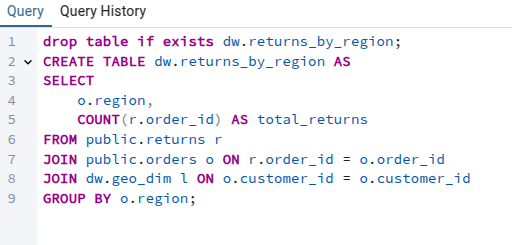
- Поле total\_returns использует целочисленный тип (`INTEGER`), так как хранит количество возвратов.

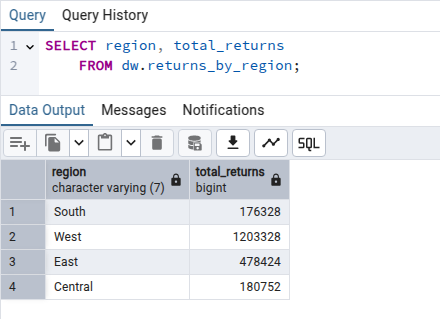
Использование индексов:

- Для повышения производительности рекомендуется создать индексы на полях order\_id (в таблицах returns и orders`) и `customer\_id (в таблицах orders и `location\_dim`).

Особенности реализации:

- Таблица returns\_by\_region создаётся один раз и хранит данные статично. Для актуализации данных потребуется повторное выполнение запроса.





Задание 3

##### **Задание 3: Определить выручку по месяцам**

Выводим выручку (sales) по месяцам (order\_date из orders):

Описание логики:

- Запрос вычисляет выручку (`sales`) по месяцам на основе даты заказа (`order\_date`).

- Функция DATE\_TRUNC округляет дату до начала месяца, что позволяет группировать данные по месяцам.

- Агрегатная функция SUM вычисляет общую выручку для каждого месяца.

Обоснование выбора типов данных:

- Поле order\_date имеет тип DATE или TIMESTAMP для хранения дат.

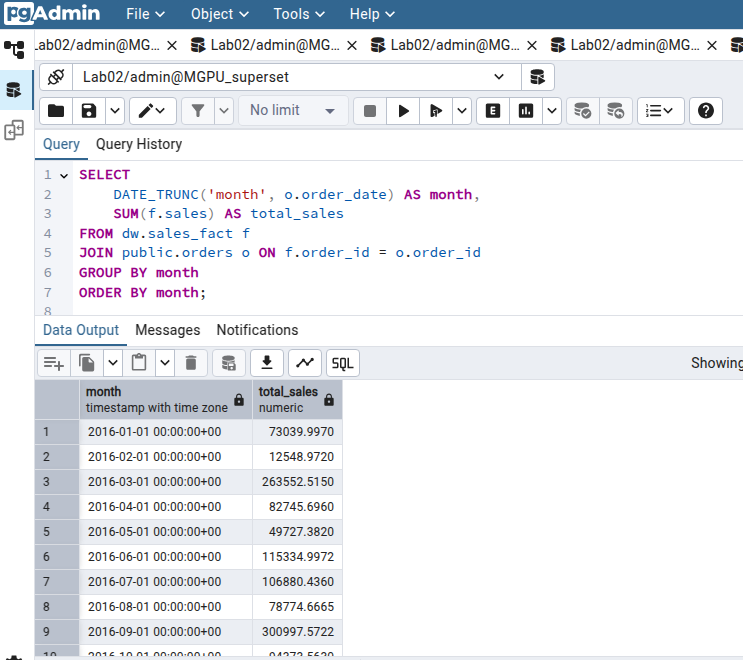
- Поле sales использует числовой тип (`NUMERIC` или `DECIMAL`) для точного хранения финансовых данных.

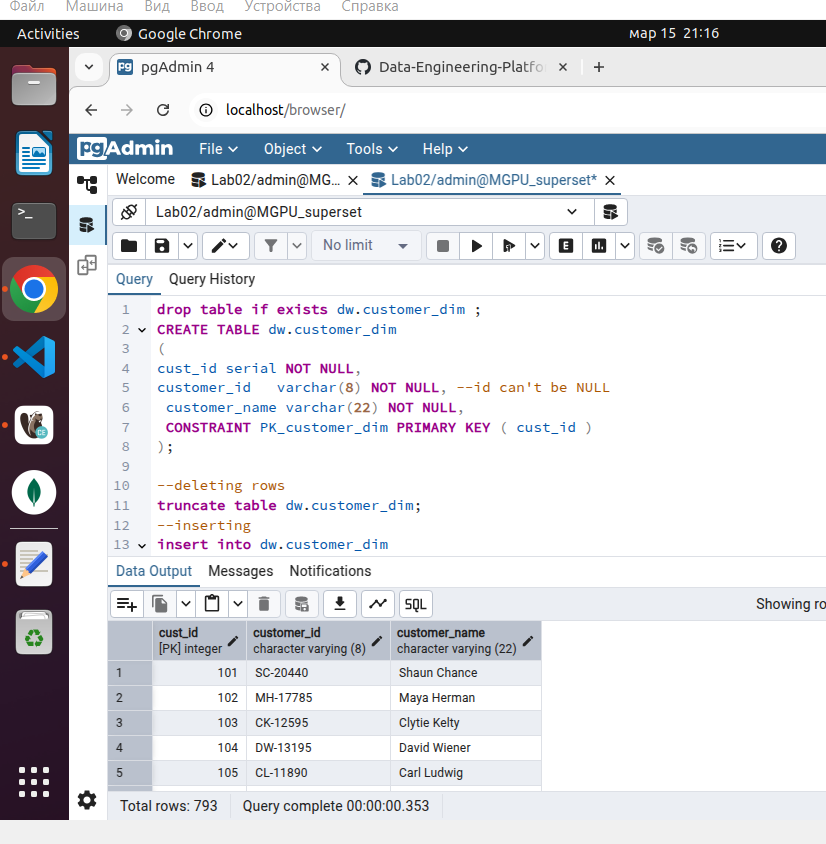
Использование индексов:

- Для ускорения выполнения запроса рекомендуется создать индекс на поле order\_date в таблице orders и на поле order\_id в таблицах sales\_fact и orders.

Особенности реализации:

- Запрос возвращает актуальные данные на момент выполнения. Для хранения результатов можно создать таблицу или представление.





# Заключение

В ходе исследования были рассмотрены фундаментальные принципы проектирования реляционных баз данных, включая концепцию многоуровневой архитектуры хранения данных, установление взаимосвязей между сущностями и выбор оптимальных типов данных для обеспечения целостности и эффективности системы.

Практическая часть работы включала реализацию этапов загрузки данных, что способствовало развитию навыков составления и выполнения SQL-запросов.

Полученные теоретические знания и практический опыт формируют основу для дальнейшего изучения расширенных функциональных возможностей систем управления базами данных, таких как оптимизация выполнения запросов, управление транзакциями и администрирование баз данных.