

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
Основная часть.....	3
Практические задания 11 вариант	15
Заключение	19

Введение

Цель

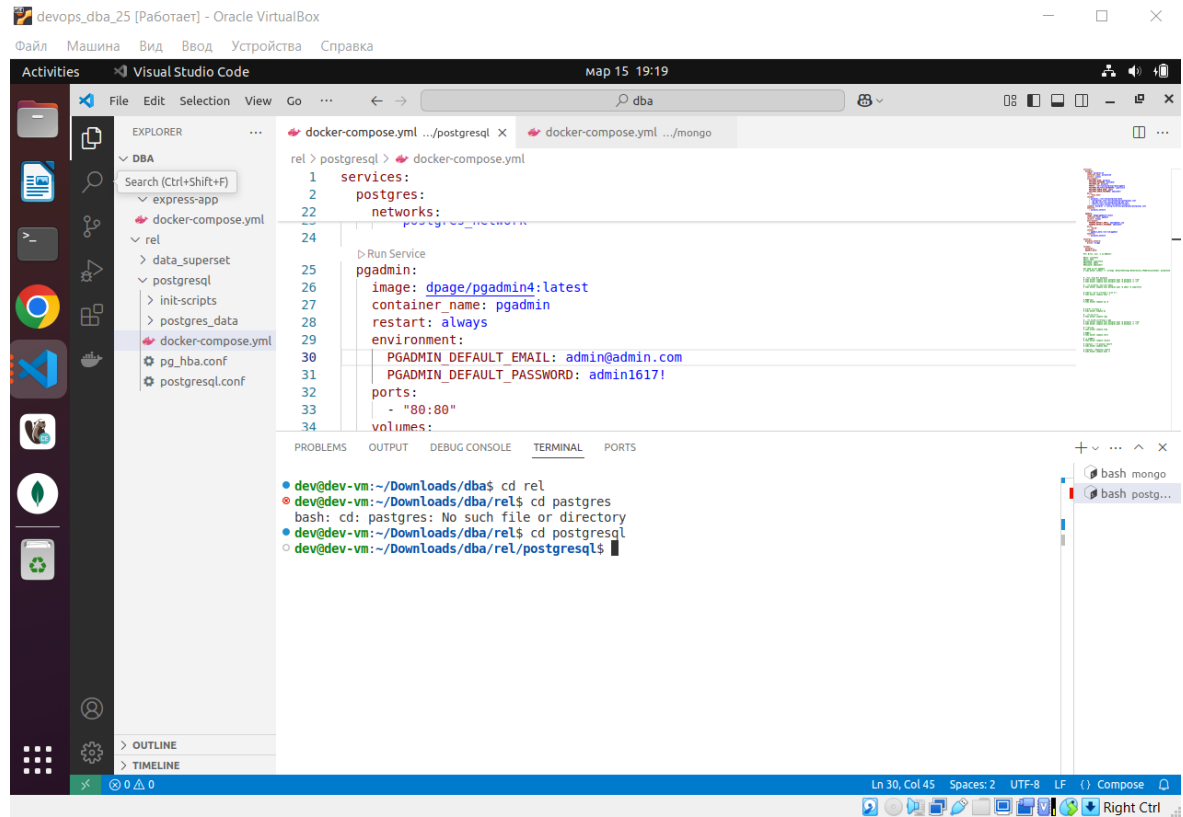
Цель: освоить принципы проектирования баз данных, создания структуры таблиц и загрузки данных в PostgreSQL.

Задачи

- Установить Postgres базу данных на компьютер. Инструкции по установке Postgres.
- Установить клиент SQL для подключения базы данных. Инструкции по установке DBeaver . Так же вы можете использовать любой другой клиент для подключения к вашей 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



```
devops_dba_25 [Работает] - Oracle VirtualBox
Файл Машина Вид Ввод Устройства Справка

Visual Studio Code map 15 19:19

EXPLORER
  DBA
    express-app
    docker-compose.yml
    rel
      data_superset
      postgresql
      init-scripts
      postgres_data
      docker-compose.yml
      pg_hba.conf
      postgresql.conf

  docker-compose.yml .../postgresql X docker-compose.yml .../mongo

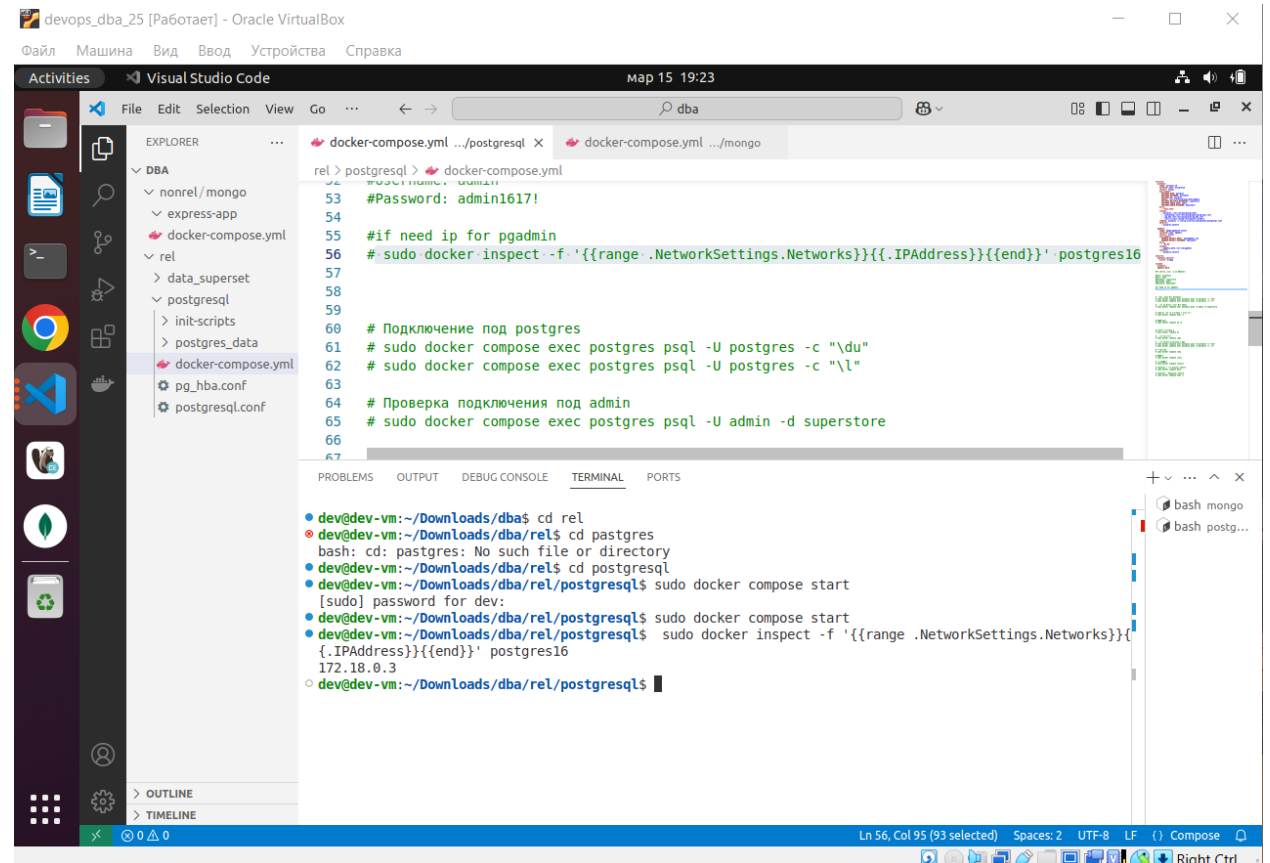
rel > postgresql > docker-compose.yml
1 services:
2   postgres:
22     networks:
24       postgres_network

> Run Service
25 pgadmin:
26   image: dpage/pgadmin4:latest
27   container_name: pgadmin
28   restart: always
29   environment:
30     PGADMIN_DEFAULT_EMAIL: admin@admin.com
31     PGADMIN_DEFAULT_PASSWORD: admin1617!
32   ports:
33     - "80:80"
34   volumes:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

dev@dev-vm:~/Downloads/dba$ cd rel
dev@dev-vm:~/Downloads/dba/rel$ cd postgres
bash: cd: postgres: No such file or directory
dev@dev-vm:~/Downloads/dba/rel$ cd postgresql
dev@dev-vm:~/Downloads/dba/rel/postgresql$
```

Запустила докер композер, проверила ip базы



```
devops_dba_25 [Работает] - Oracle VirtualBox
Файл Машина Вид Ввод Устройства Справка

Visual Studio Code map 15 19:23

EXPLORER
  DBA
    nonrel/mongo
    express-app
    docker-compose.yml
    rel
      data_superset
      postgresql
      init-scripts
      postgres_data
      docker-compose.yml
      pg_hba.conf
      postgresql.conf

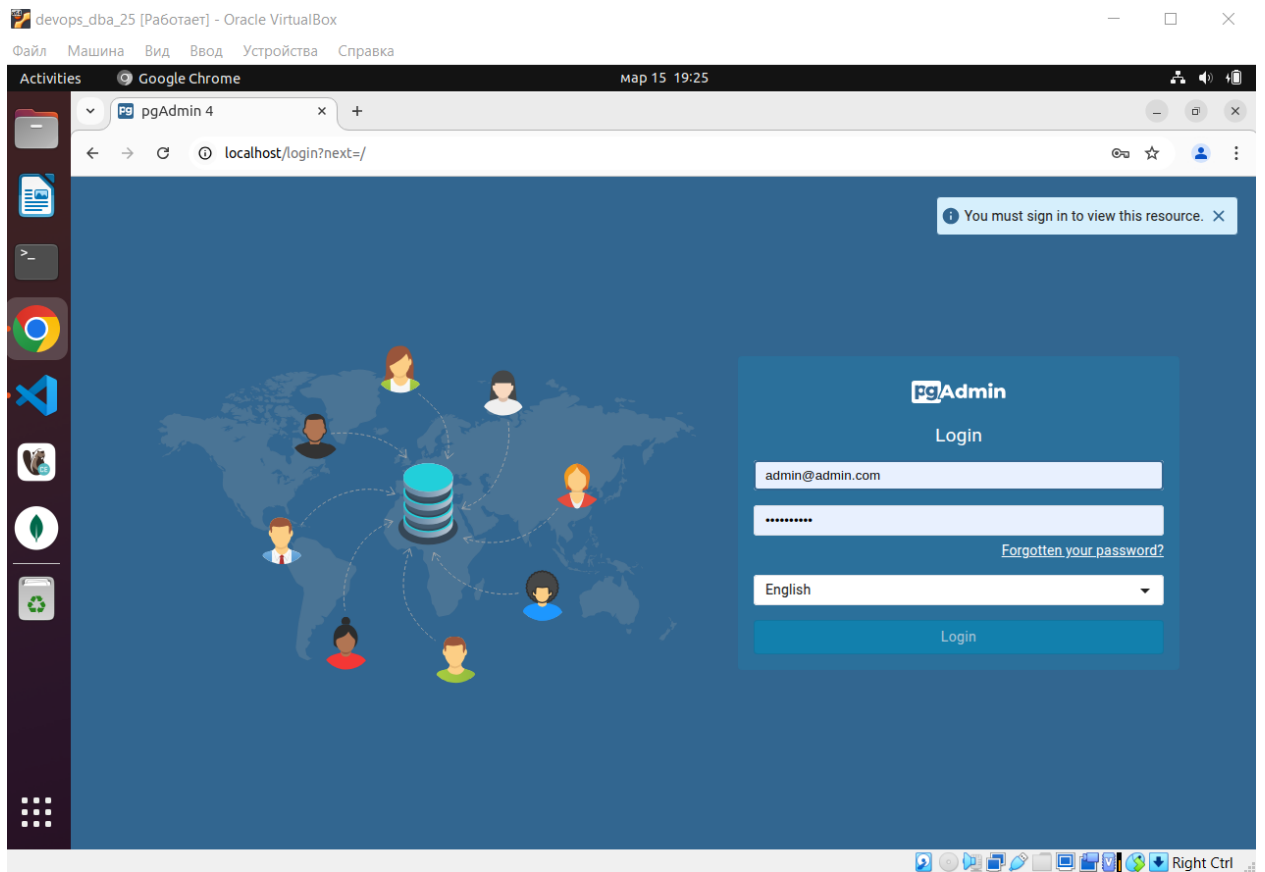
  docker-compose.yml .../postgresql X docker-compose.yml .../mongo

rel > postgresql > docker-compose.yml
53 #Password: admin1617!
54
55 #if need ip for pgadmin
56 #sudo docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' postgres16
57
58
59
60 # Подключение под postgres
61 # sudo docker compose exec postgres psql -U postgres -c "\du"
62 # sudo docker compose exec postgres psql -U postgres -c "\l"
63
64 # Проверка подключения под admin
65 # sudo docker compose exec postgres psql -U admin -d superstore
66
67

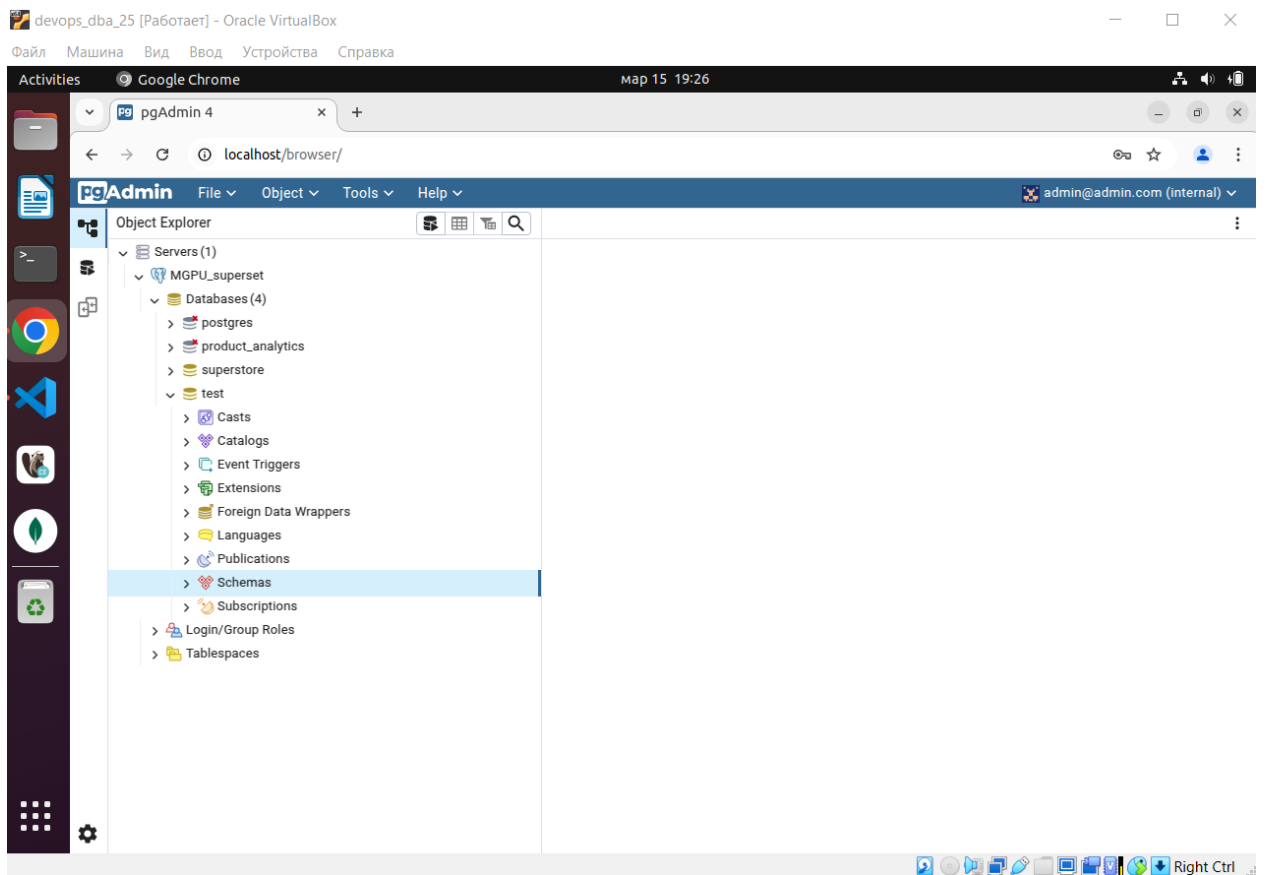
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

dev@dev-vm:~/Downloads/dba$ cd rel
dev@dev-vm:~/Downloads/dba/rel$ cd postgres
bash: cd: postgres: No such file or directory
dev@dev-vm:~/Downloads/dba/rel$ cd postgresql
dev@dev-vm:~/Downloads/dba/rel/postgresql$ sudo docker compose start
[sudo] password for dev:
dev@dev-vm:~/Downloads/dba/rel/postgresql$ sudo docker compose start
dev@dev-vm:~/Downloads/dba/rel/postgresql$ sudo docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' postgres16
172.18.0.3
dev@dev-vm:~/Downloads/dba/rel/postgresql$
```

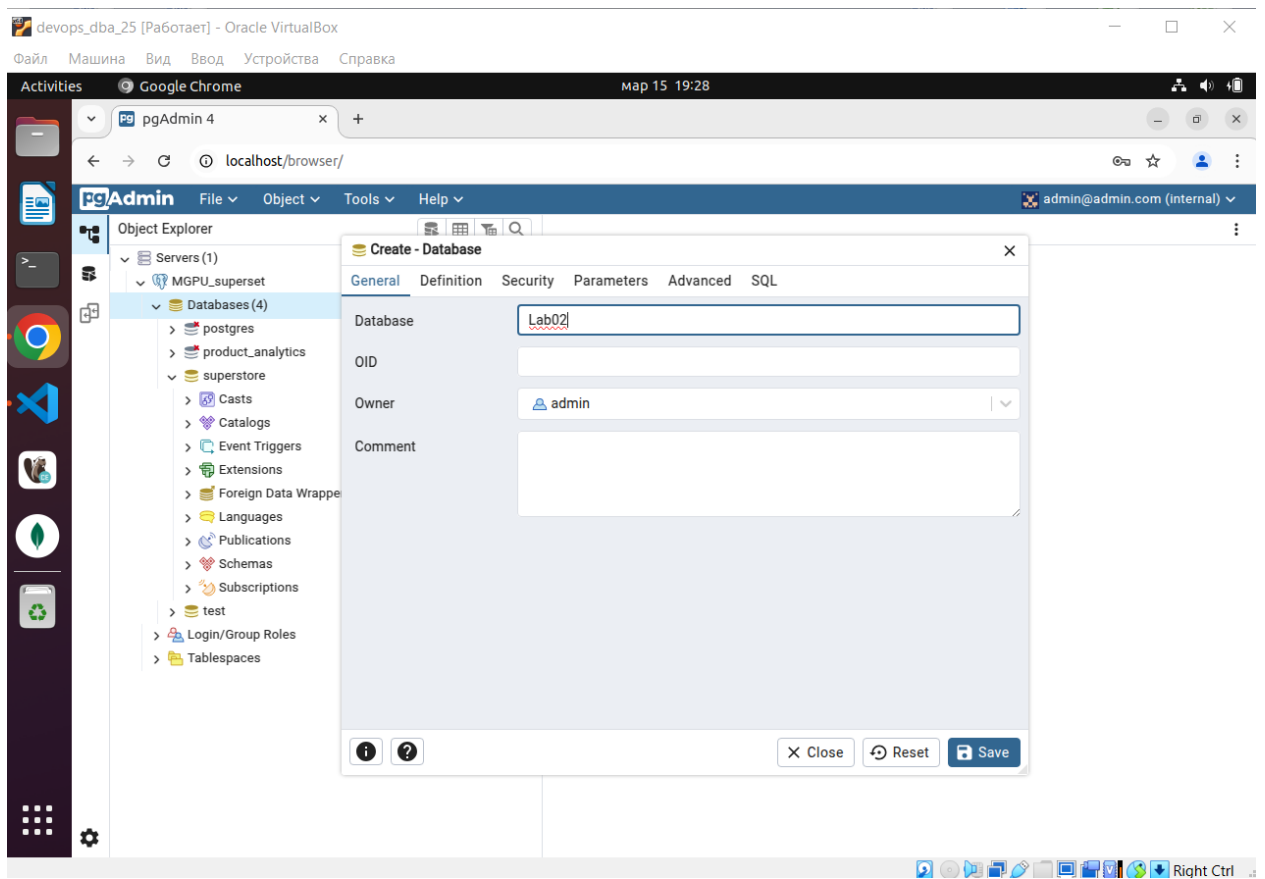
Вхожу в PGAdmin



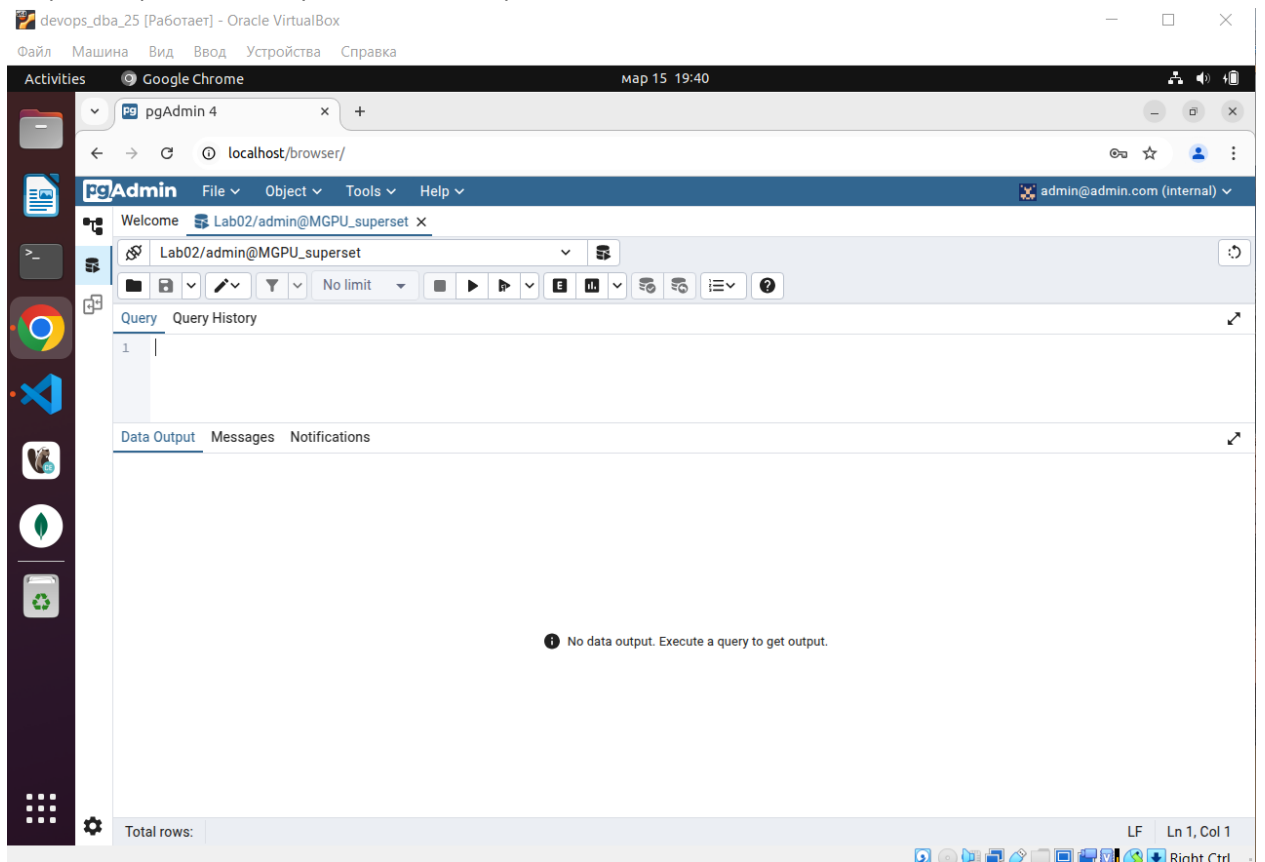
Подключилась к СУБД



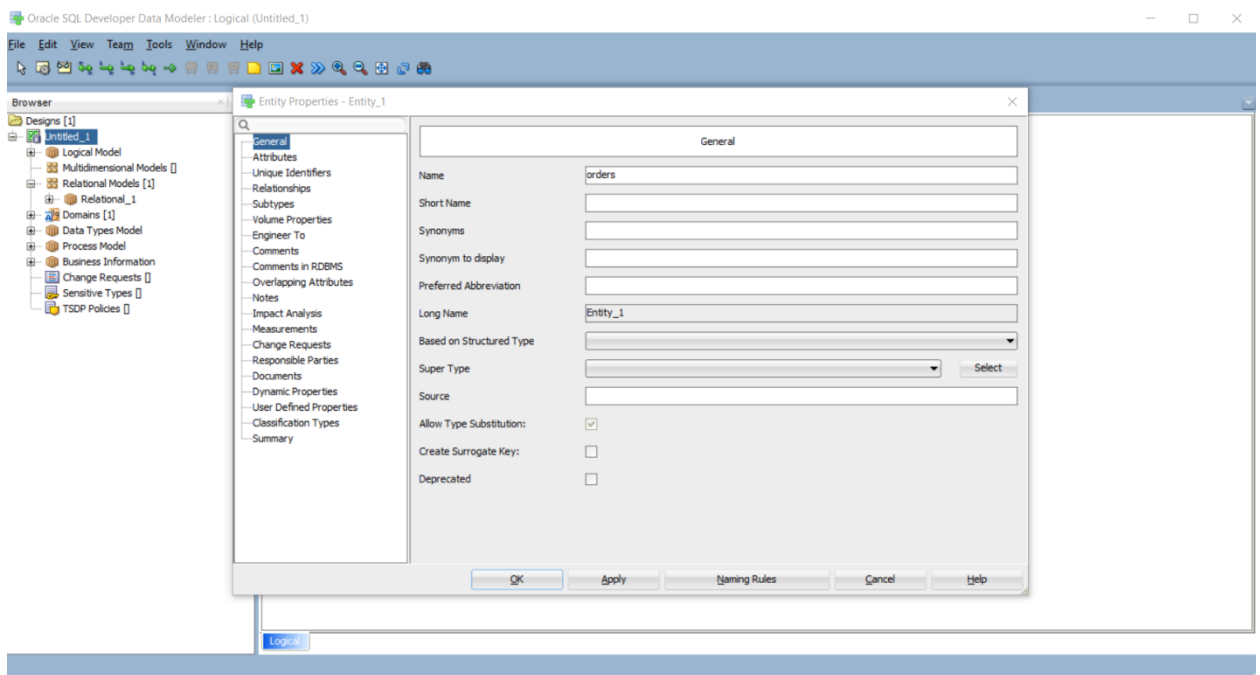
Создаем свою базу данных для работы



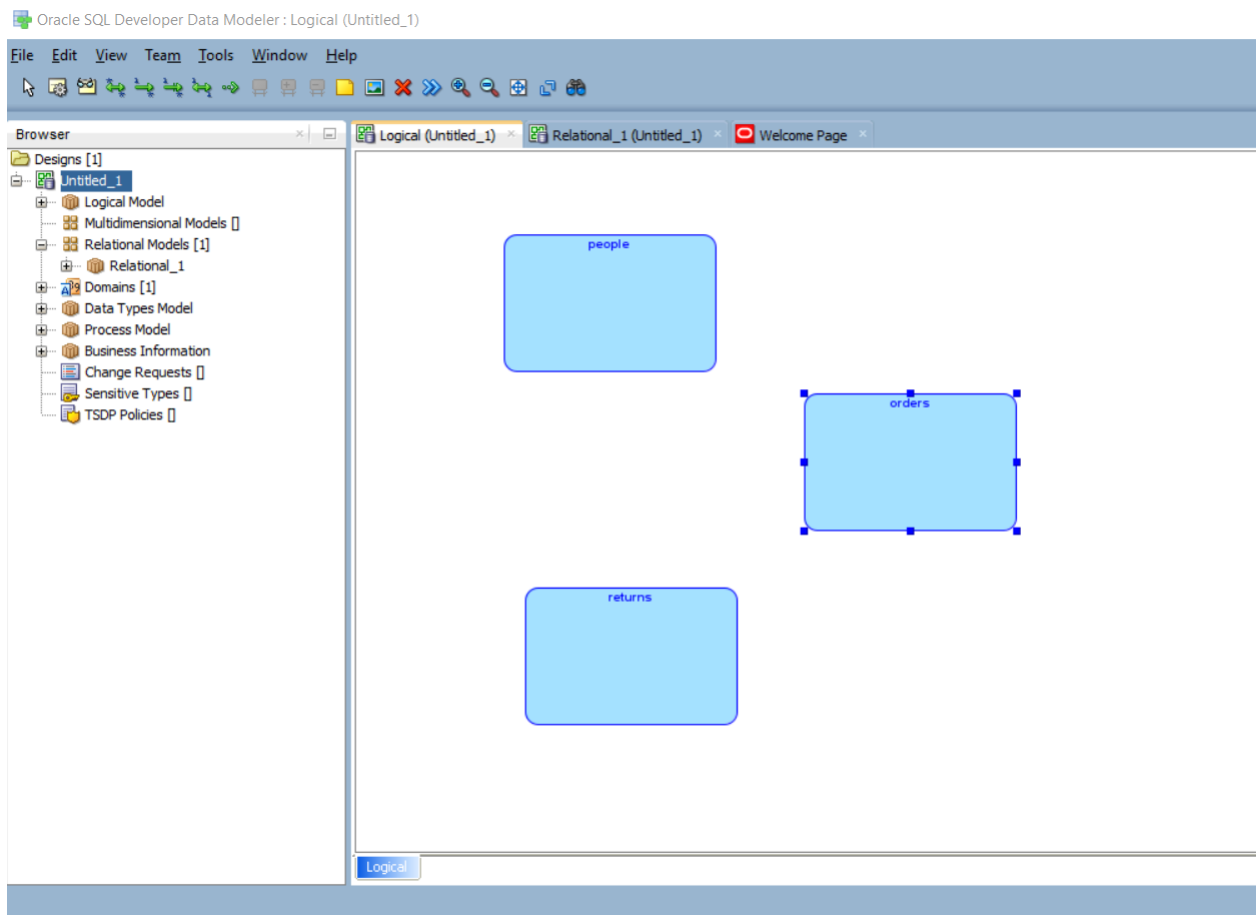
Открыла sql-окно, где будем делать запросы к базе данных



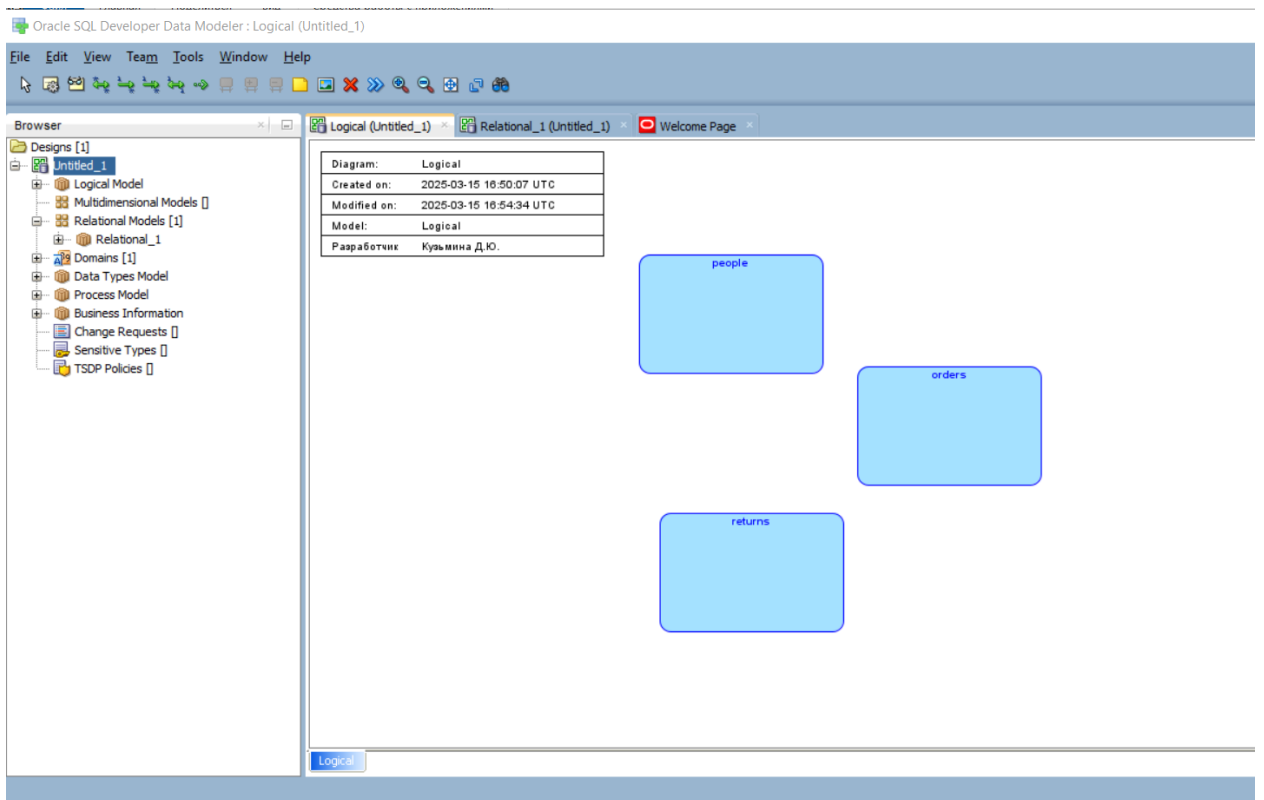
Создаем модель



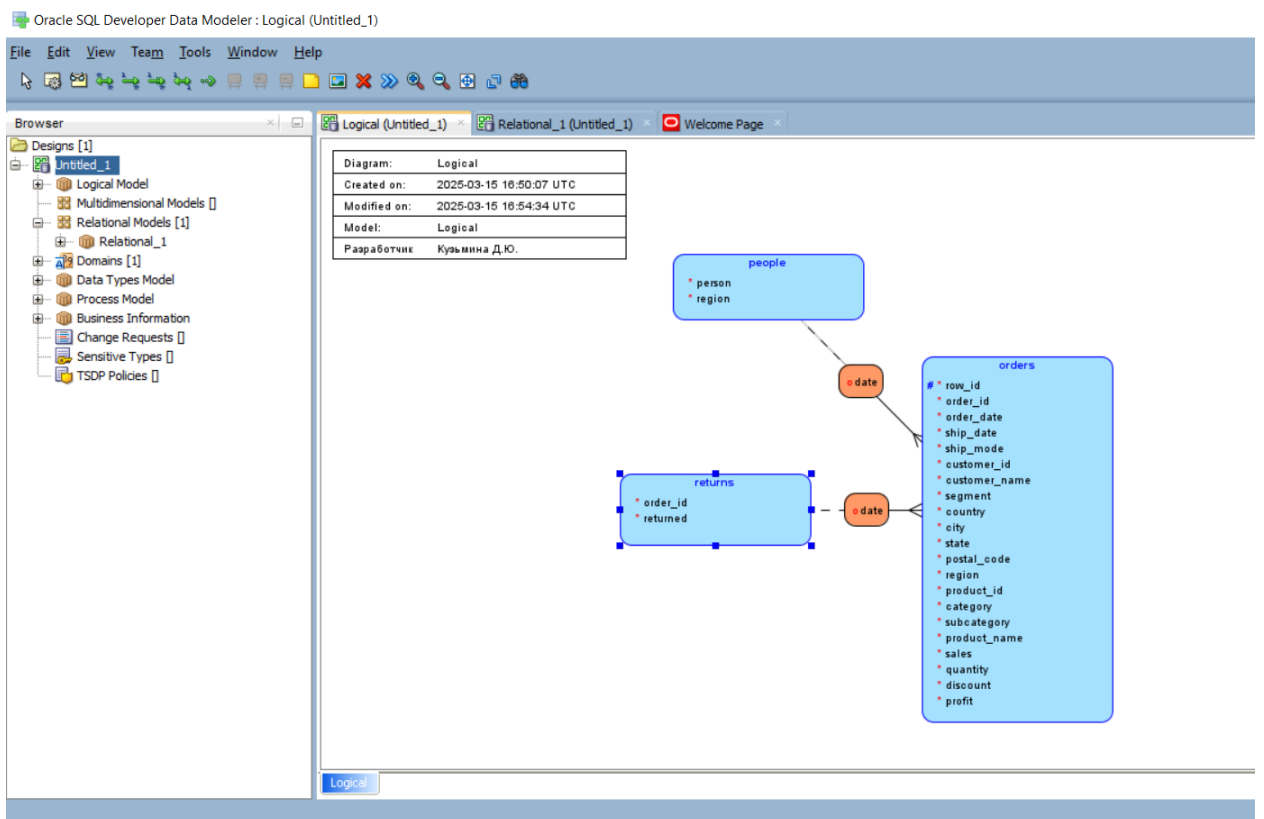
Создаем 3 сущности на основе внешнего источника данных superset excel



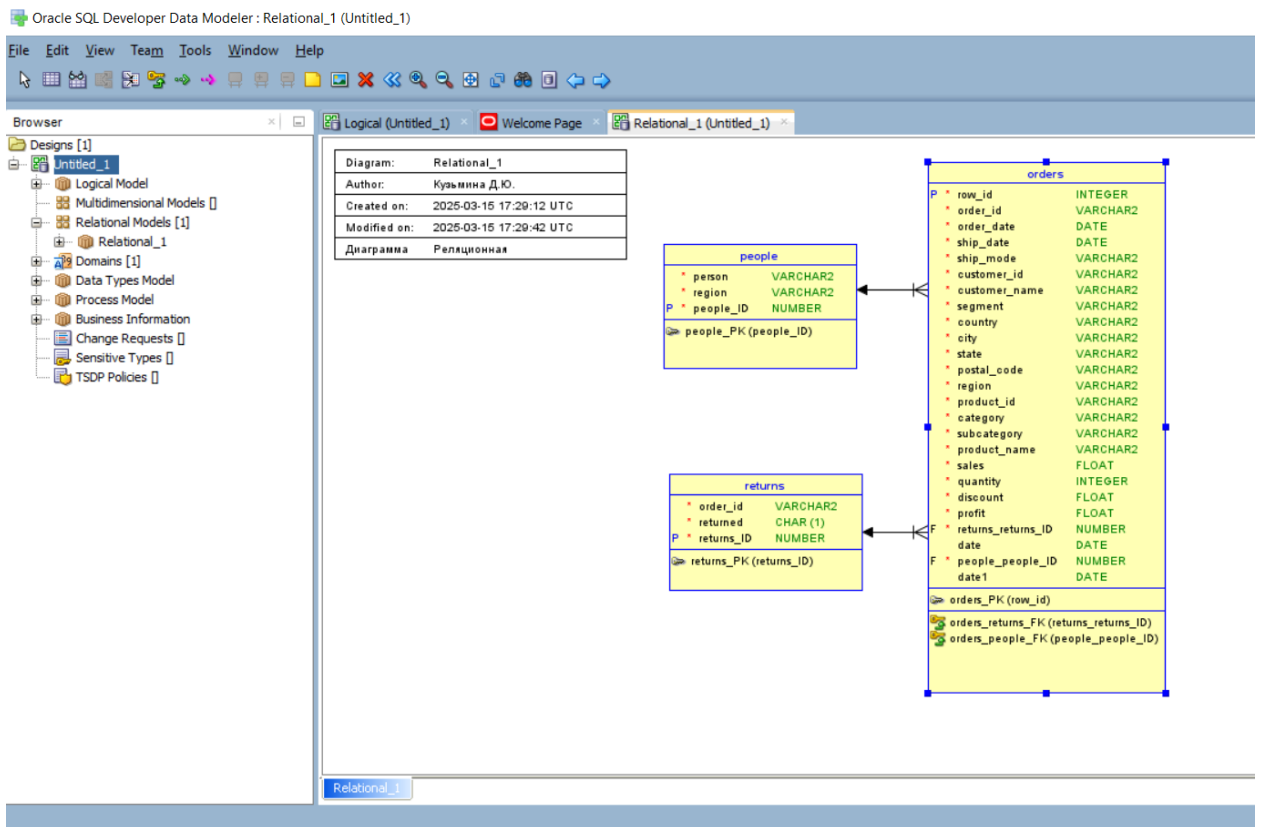
Настроим легенду



Добавим все поля в концептуальном проектировании



Перевела в реляционную



Создали ddl скрипт

Oracle SQL Developer Data Modeler : Relational_1 (Untitled_1)

File Edit View Team Tools Window Help

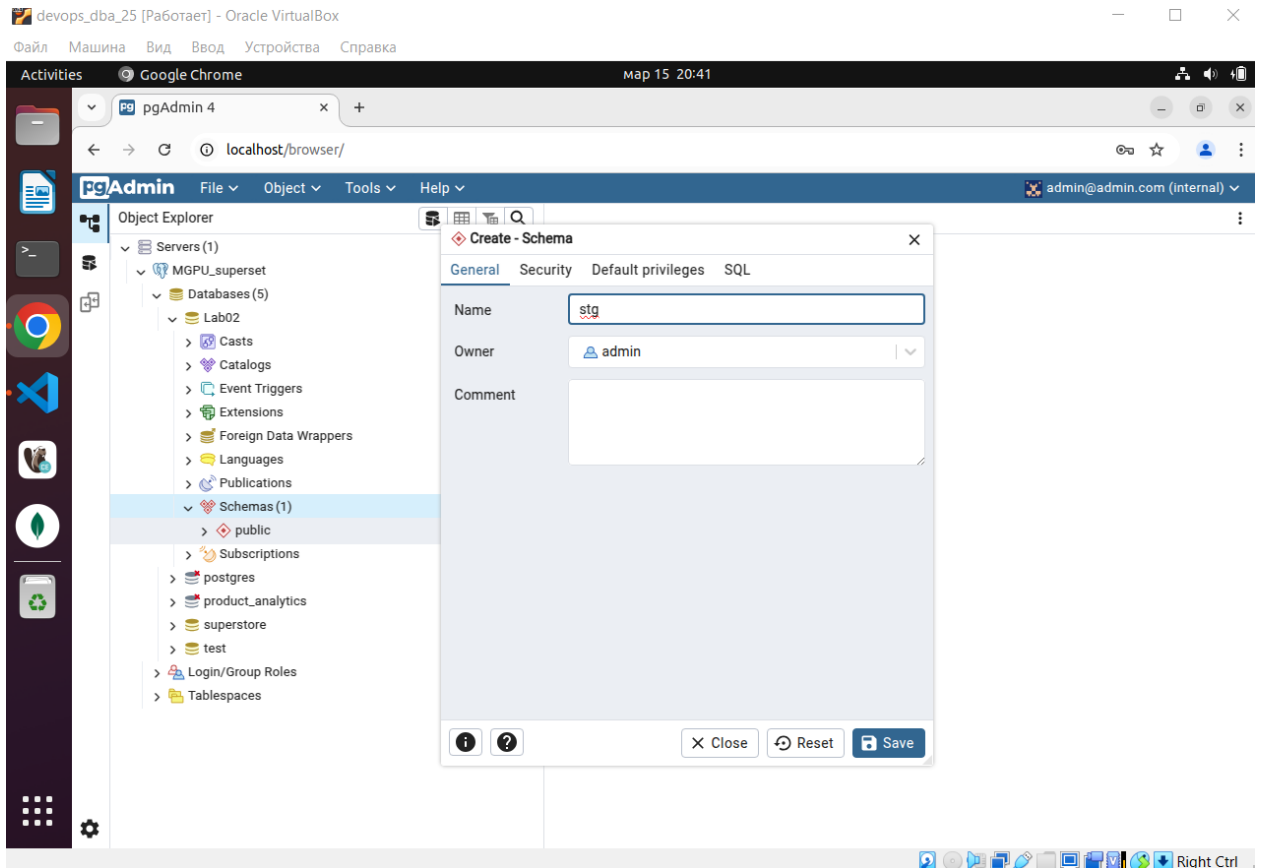
DDL File Editor - Oracle Database 11g

Oracle Database 11g Relational_1 Generate Clear

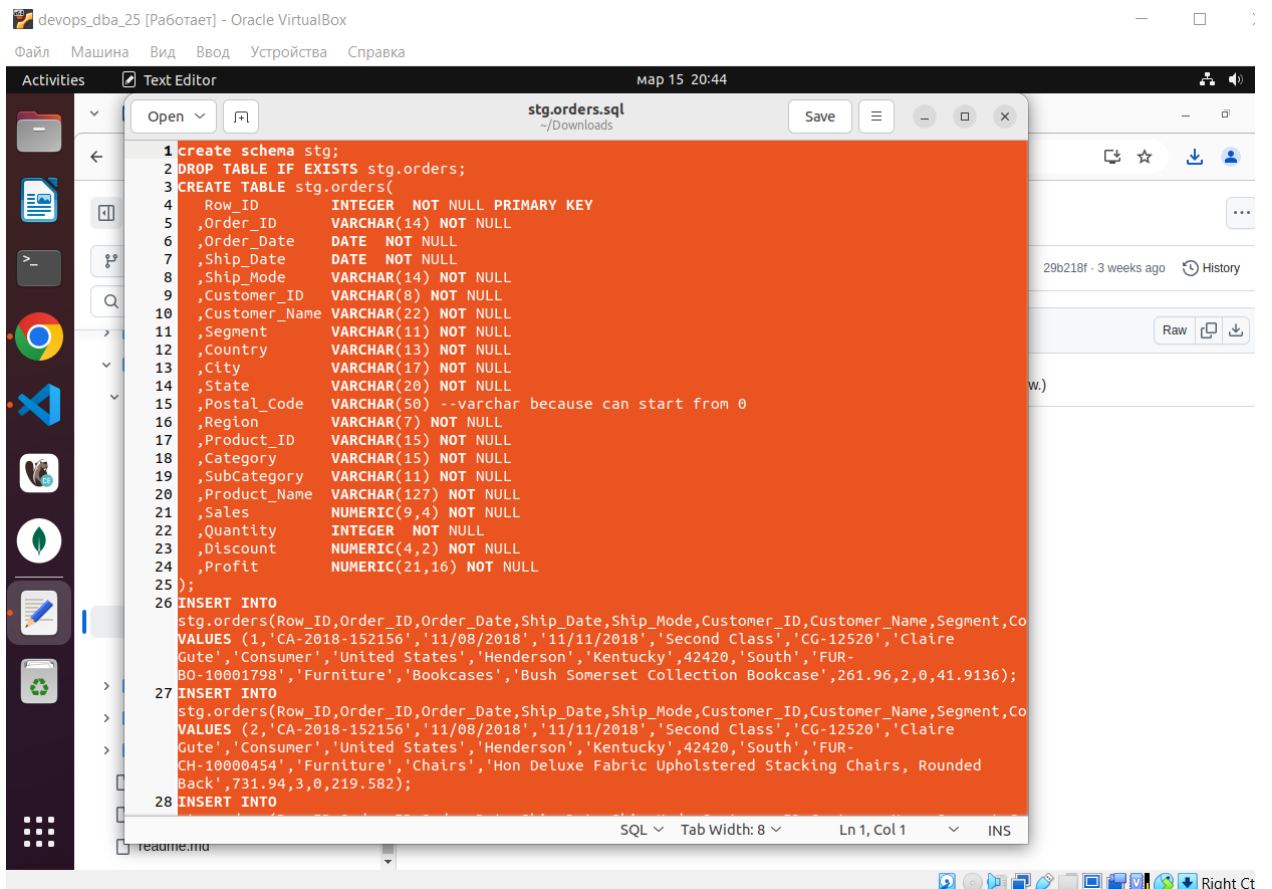
```
10 -- predefined type, no DDL - XMLTYPE
11
12 CREATE TABLE orders
13 (
14     row_id          INTEGER NOT NULL ,
15     order_id        VARCHAR2
16     -- ERROR: VARCHAR2 size not specified
17     NOT NULL ,
18     order_date      DATE NOT NULL ,
19     ship_date       DATE NOT NULL ,
20     ship_mode       VARCHAR2
21     -- ERROR: VARCHAR2 size not specified
22     NOT NULL ,
23     customer_id     VARCHAR2
24     -- ERROR: VARCHAR2 size not specified
25     NOT NULL ,
26     customer_name   VARCHAR2
27     -- ERROR: VARCHAR2 size not specified
28     NOT NULL ,
29     segment         VARCHAR2
30     -- ERROR: VARCHAR2 size not specified
31     NOT NULL ,
32     country         VARCHAR2
33     -- ERROR: VARCHAR2 size not specified
34     NOT NULL ,
35     city            VARCHAR2
36     -- ERROR: VARCHAR2 size not specified
37     NOT NULL ,
38     state           VARCHAR2
```

Save Find Close Help

Создаем stg схему



Открыли данные, закинули



Загнали данные

devops_dba_25 [Работает] - Oracle VirtualBox

Файл Машина Вид Ввод Устройства Справка

Activities Google Chrome map 15 20:49

pgAdmin 4 Data-Engineering-Platform localhost/browser/ admin@admin.com (internal)

Welcome Lab02/admin@MGPU_superset

Query Query History

```
38 INSERT INTO stg.orders (Row_ID, Order_ID, Order_Date, Ship_Date, Ship_Mode, Customer_ID, Customer_Name, Segment, Country, City, State, Postal_Code)
39 INSERT INTO stg.orders (Row_ID, Order_ID, Order_Date, Ship_Date, Ship_Mode, Customer_ID, Customer_Name, Segment, Country, City, State, Postal_Code)
40 INSERT INTO stg.orders (Row_ID, Order_ID, Order_Date, Ship_Date, Ship_Mode, Customer_ID, Customer_Name, Segment, Country, City, State, Postal_Code)
41 INSERT INTO stg.orders (Row_ID, Order_ID, Order_Date, Ship_Date, Ship_Mode, Customer_ID, Customer_Name, Segment, Country, City, State, Postal_Code)
42 INSERT INTO stg.orders (Row_ID, Order_ID, Order_Date, Ship_Date, Ship_Mode, Customer_ID, Customer_Name, Segment, Country, City, State, Postal_Code)
43 INSERT INTO stg.orders (Row_ID, Order_ID, Order_Date, Ship_Date, Ship_Mode, Customer_ID, Customer_Name, Segment, Country, City, State, Postal_Code)
44 INSERT INTO stg.orders (Row_ID, Order_ID, Order_Date, Ship_Date, Ship_Mode, Customer_ID, Customer_Name, Segment, Country, City, State, Postal_Code)
45 INSERT INTO stg.orders (Row_ID, Order_ID, Order_Date, Ship_Date, Ship_Mode, Customer_ID, Customer_Name, Segment, Country, City, State, Postal_Code)
46 INSERT INTO stg.orders (Row_ID, Order_ID, Order_Date, Ship_Date, Ship_Mode, Customer_ID, Customer_Name, Segment, Country, City, State, Postal_Code)
47 TRUNCATE TABLE stg.orders;
```

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 3 secs 48 msec.

Total rows: Query complete 00:00:03.048

LF Ln 1, Col 1

Введем запрос SELECT

devops_dba_25 [Работает] - Oracle VirtualBox

Файл Машина Вид Ввод Устройства Справка

Activities Google Chrome map 15 20:50

pgAdmin 4 Data-Engineering-Platform localhost/browser/ admin@admin.com (internal)

Welcome Lab02/admin@MGPU_superset

Query Query History

```
1 SELECT row_id, order_id, order_date, ship_date, ship_mode, customer_id, customer_name, segment, country, city, state, postal_code,
2 FROM stg.orders;
```

Data Output Messages Notifications

Showing rows: 1 to 1000 Page No: 1 of 10

row_id [PK] integer	order_id character varying (14)	order_date date	ship_date date	ship_mode character varying (14)	customer_id character varying (8)	customer_name character varying (22)	segment character varying (11)	country character varying (3)	
1	1	CA-2018-152156	2018-11-08	2018-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States
2	2	CA-2018-152156	2018-11-08	2018-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States
3	3	CA-2018-138688	2018-06-12	2018-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States
4	4	US-2017-108966	2017-10-11	2017-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States
5	5	US-2017-108966	2017-10-11	2017-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States
6	6	CA-2016-115812	2016-06-09	2016-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States
7	7	CA-2016-115812	2016-06-09	2016-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States
8	8	CA-2016-115812	2016-06-09	2016-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States
9	9	CA-2016-115812	2016-06-09	2016-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States
10	10	CA-2016-115812	2016-06-09	2016-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States
11	11	CA-2016-115812	2016-06-09	2016-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States

Servers > MGPU_superset > Databases > Lab02 > Schemas > stg > Tables > orders

LF Ln 1, Col 1

Сделали фильтрацию

pgAdmin 4 Data-Engineering-Platform x +

localhost/browser/

pgAdmin File Object Tools Help

Lab02/admin@MG... x Lab02/admin@MG... x Lab02/admin@MG... x Lab02

Lab02/admin@MGPU_superset

Query Query History

```
1 SELECT person, region
2 FROM public.people;
```

Data Output Messages Notifications

	person [PK] character varying (17)	region character varying (7)
1	Anna Andreadi	West
2	Chuck Magee	East
3	Kelly Williams	Central
4	Cassandra Brandow	South

Tables (3)

- orders
- people
- returns

Подготовили аналитический слой, переходим к созданию слой для пользователей.
Создаем таблицу shipping.dim

pgAdmin 4 x Data-Engineering-Platform x +

localhost/browser/

pgAdmin File Object Tools Help

Lab02/admin@MG... x Lab02/admin@MG... x Lab02/admin@MG... x Lab02/admin@MG... x Lab02/admin@MG...

Lab02/admin@MGPU_superset

No limit

Query Query History

```
1 --creating a table
2 drop table if exists dw.shipping_dim ;
3 CREATE TABLE dw.shipping_dim
4 (
5     ship_id          serial NOT NULL,
6     shipping_mode     varchar(14) NOT NULL,
7     CONSTRAINT PK_shipping_dim PRIMARY KEY ( ship_id )
8 );
9
10 --deleting rows
11 truncate table dw.shipping_dim;
12
13 --generating ship_id and inserting ship_mode from orders
14 insert into dw.shipping_dim
15 select 100+row_number() over(), ship_mode from (select distinct ship_mode from st
16 --checking
```

Data Output Messages Notifications

Showing rows

	ship_id [PK] integer	shipping_mode character varying (14)
1	101	Standard Class
2	102	Second Class
3	103	Same Day

Total rows: 4 Query complete 00:00:00.762

Получили результат выполнения предложенного скрипта

pgAdmin 4 | Data-Engineering-Platform | localhost/browser/

Welcome | Lab02/admin@MGPU_superset*

Lab02/admin@MGPU_superset

Query | Query History

```

5 insert into dw.shipping_dim
6 select 100+row_number() over(), ship_mode from (select distinct ship_mode from stg.orders ) a;
7 --checking
8 select * from dw.shipping_dim sd;

```

Data Output | Messages | Notifications

Showing rows: 1 to 4 | Page No: 1

	ship_id [PK] integer	shipping_mode character varying (14)
1	101	Standard Class
2	102	Second Class
3	103	Same Day
4	104	First Class

Создаем справочники и витрины

pgAdmin 4 | Data-Engineering-Platform | localhost/browser/

admin@admin.com (internal)

Lab02/admin@MG... | Lab02/admin@MG... | Lab02/admin@MG... | Lab02/admin@MG... | Lab02/admin@MG... | Lab02/admin@MGPU_superset*

Lab02/admin@MGPU_superset

Query | Query History

```

1 SELECT sales_id, cust_id, order_date_id, ship_date_id, prod_id, ship_id, geo_id, order_id, sales, profit, quantity, discount
2 FROM dw.sales_fact;

```

Data Output | Messages | Notifications

Showing rows: 1 to 1000 | Page No: 1 of 10

	sales_id [PK] integer	cust_id integer	order_date_id integer	ship_date_id integer	prod_id integer	ship_id integer	geo_id integer	order_id character varying (25)	sales numeric (9,4)	profit numeric (21,16)	quantity integer	discount numeric
1	101	461	20181226	20190102	101	101	225	CA-2018-155166	212.9400	25.552800000000000000	3	
2	102	755	20170416	20170421	101	101	503	CA-2017-142734	127.7640	2.839200000000000000	2	
3	103	408	20191228	20191231	101	104	313	CA-2019-101322	340.7040	-34.070400000000000000	6	
4	104	390	20180415	20180421	102	101	257	US-2018-123750	189.5880	-145.350800000000000000	2	
5	105	143	20171231	20180105	103	101	689	CA-2017-156377	14.7600	-11.439000000000000000	5	
6	106	223	20180626	20180703	103	101	356	CA-2018-114951	22.1400	6.420600000000000000	3	
7	107	813	20181215	20181222	103	101	546	US-2018-116442	14.7600	4.280400000000000000	2	
8	108	626	20161031	20161102	103	102	277	CA-2016-145387	14.7600	4.280400000000000000	2	
9	109	814	20180903	20180908	103	102	451	US-2018-147711	14.7600	4.280400000000000000	2	
10	110	751	20170718	20170720	103	102	503	CA-2017-122973	7.3800	2.140200000000000000	1	
11	111	670	20190612	20190614	103	102	686	CA-2019-124191	8.8560	-6.863400000000000000	3	

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.

Query Query History

```
1 drop table if exists dw.sales_by_category;
2 CREATE VIEW dw.sales_by_category AS
3 SELECT
4     p.category,
5     SUM(f.sales) AS total_sales,
6     SUM(f.profit) AS total_profit
7 FROM dw.sales_fact f
8 JOIN dw.product_dim p ON p.product_id = f.product_id
9 GROUP BY p.category;
10
11 SELECT * FROM dw.sales_by_category
```


pgAdmin File Object Tools Help

Lab02/admin@MG... x Lab02/admin@MG... x Lab02/admin@MG... x Lab02/admin@MG... x Lab

Lab02/admin@MGPU_superset

Query Query History

```
1 SELECT category, total_sales, total_profit
2 FROM dw.sales_by_category;
```

Data Output Messages Notifications

Showing rows

	category character varying (15)	total_sales numeric	total_profit numeric
1	Furniture	2076669577.7112	258902907.6167999897897720
2	Office Supplies	5880834202.3680	733176375.5519999710860800
3	Technology	2021536757.0640	252029379.0959999900608400

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.

Query Query History

```
1 drop table if exists dw.returns_by_region;
2 CREATE TABLE dw.returns_by_region AS
3 SELECT
4     o.region,
5     COUNT(r.order_id) AS total_returns
6 FROM public.returns r
7 JOIN public.orders o ON r.order_id = o.order_id
8 JOIN dw.geo_dim l ON o.customer_id = l.customer_id
9 GROUP BY o.region;
```


Query

Query History

1

▼

SELECT region, total_returns

2

FROM dw.returns_by_region;

Data Output

Messages

Notifications

☰+

📄

▼

📋

▼

🗑️

📦

⬇️

📈

SQL

	region character varying (7) 🔒	total_returns bigint 🔒
1	South	176328
2	West	1203328
3	East	478424
4	Central	180752

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.

pgAdmin File Object Tools Help

Lab02/admin@MG... x Lab02/admin@MG... x Lab02/admin@MG... x Lab02/admin@MG... x

Lab02/admin@MGPU_superset

Query Query History

```

1 SELECT
2     DATE_TRUNC('month', o.order_date) AS month,
3     SUM(f.sales) AS total_sales
4 FROM dw.sales_fact f
5 JOIN public.orders o ON f.order_id = o.order_id
6 GROUP BY month
7 ORDER BY month;
8

```

Data Output Messages Notifications

	month timestamp with time zone	total_sales numeric
1	2016-01-01 00:00:00+00	73039.9970
2	2016-02-01 00:00:00+00	12548.9720
3	2016-03-01 00:00:00+00	263552.5150
4	2016-04-01 00:00:00+00	82745.6960
5	2016-05-01 00:00:00+00	49727.3820
6	2016-06-01 00:00:00+00	115334.9972
7	2016-07-01 00:00:00+00	106880.4360
8	2016-08-01 00:00:00+00	78774.6665
9	2016-09-01 00:00:00+00	300997.5722
10	2016-10-01 00:00:00+00	84373.5520

Файл Машина Вид Ввод Устройства Справка

Activities Google Chrome map 15 21:16

pgAdmin 4 Data-Engineering-Platform

localhost/browser/

pgAdmin File Object Tools Help

Welcome Lab02/admin@MG... x Lab02/admin@MGPU_superset* x

Lab02/admin@MGPU_superset

Query Query History

```

1 drop table if exists dw.customer_dim ;
2 CREATE TABLE dw.customer_dim
3 (
4     cust_id serial NOT NULL,
5     customer_id varchar(8) NOT NULL, --id can't be NULL
6     customer_name varchar(22) NOT NULL,
7     CONSTRAINT PK_customer_dim PRIMARY KEY ( cust_id )
8 );
9
10 --deleting rows
11 truncate table dw.customer_dim;
12 --inserting
13 insert into dw.customer_dim

```

Data Output Messages Notifications

	cust_id [PK] integer	customer_id character varying (8)	customer_name character varying (22)
1	101	SC-20440	Shaun Chance
2	102	MH-17785	Maya Herman
3	103	CK-12595	Clytie Kelty
4	104	DW-13195	David Wiener
5	105	CL-11890	Carl Ludwig

Total rows: 793 Query complete 00:00:00.353

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.