

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

Мареев Георгий Александрович БД-241м

Лабораторная работа 2. Моделирование данных и SQL для Data Engineering
Вариант 13

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,
кандидат технических наук

Москва
2025

Введение

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

Основная часть

Просмотр всех запущенные контейнеры:
docker ps

```
dev@dev-vm:~/Downloads/dba/rel/postgresql$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED    STATUS    PORTS                               NAMES
a5bb7206afbd   postgres:16   "docker-entrypoint.s..." 13 days ago Up 6 minutes 0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp postgres16
ec4919b750da   dpkg/pgadmin4:latest "/entrypoint.sh"         13 days ago Up 6 minutes 0.0.0.0:80->80/tcp, [::]:80->80/tcp, 443/tcp pgadmin
```

Рисунок 1 Все запущенные контейнеры

Перезапуск контейнеров:

```
dev@dev-vm:~/Downloads/dba/rel/postgresql$ sudo docker compose stop
[sudo] password for dev:
```

[+] Stopping 2/2

✓ Container pgadmin Stopped

1.5s

✓ Container postgres16 Stopped

0.2s

```
dev@dev-vm:~/Downloads/dba/rel/postgresql$ sudo docker compose start
```

[+] Running 2/2

✓ Container postgres16 Started

0.3s

✓ Container pgadmin Started

0.3s

```
dev@dev-vm:~/Downloads/dba/rel/postgresql$ sudo docker compose stop
```

[sudo] password for dev:

[+] Stopping 2/2

✓ Container pgadmin Stopped

✓ Container postgres16 Stopped

```
dev@dev-vm:~/Downloads/dba/rel/postgresql$ sudo docker compose start
```

[+] Running 2/2

✓ Container postgres16 Started

✓ Container pgadmin Started

Рисунок 2 Перезапуск контейнеров

Проверка ip-адреса контейнера postgres

```
sudo docker inspect -f '{{range
.NetworkSettings.Networks}}{{.IPAddress}}{{end}}}' postgres16
```

172.18.0.3

```
dev@dev-vm:~/Downloads/dba/rel/postgresql$ sudo docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}}' postgres16
172.18.0.3
```

Рисунок 3 IP-адрес контейнера postgres

Запуск pgadmin в браузере на виртуальной машине:

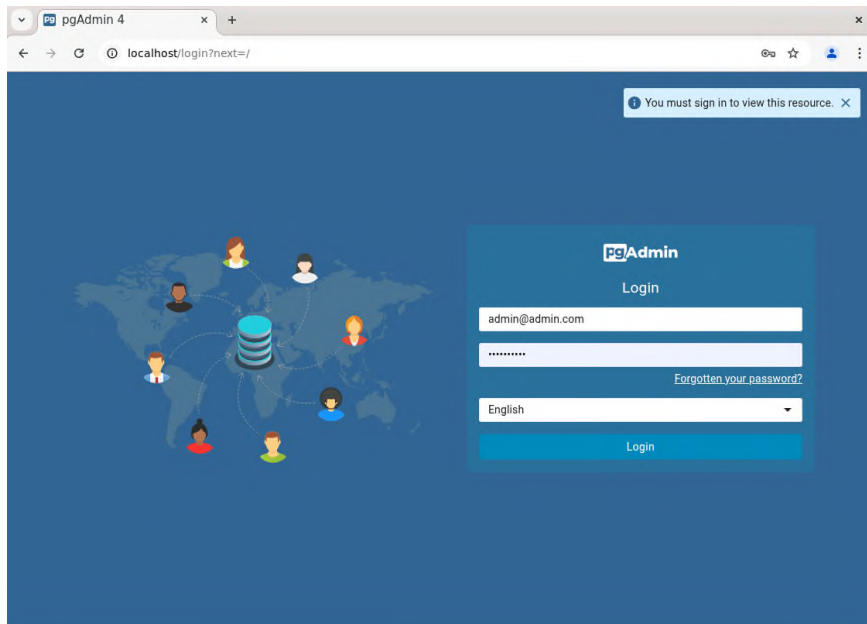


Рисунок 4 pgadmin браузер на виртуальной машине

Состояние MGPU_superset

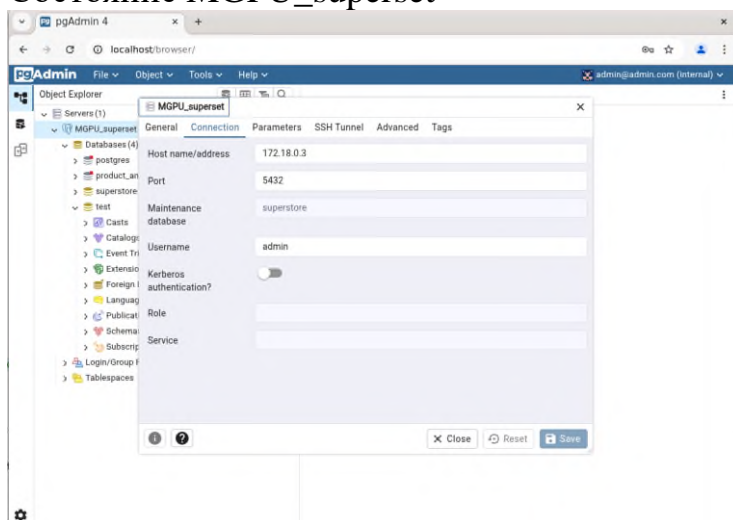


Рисунок 5 Состояние MGPU_superset

Создание бд lab_02

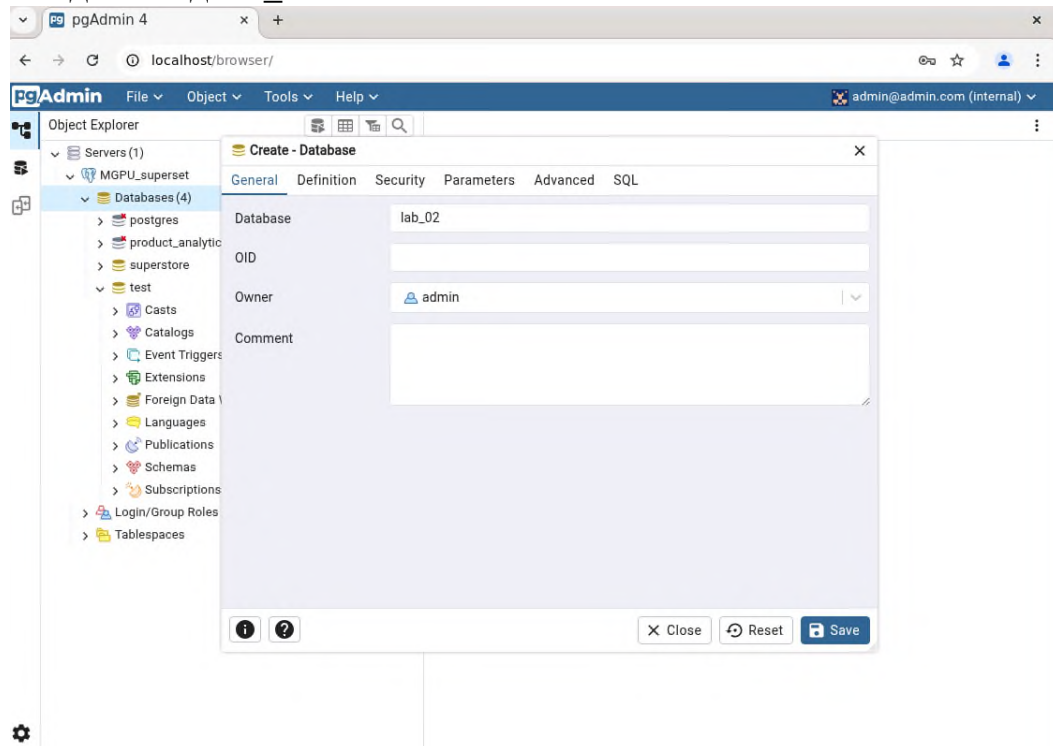


Рисунок 6 Создание бд lab_02

Запуск pgadmin в браузере на локальной машине:

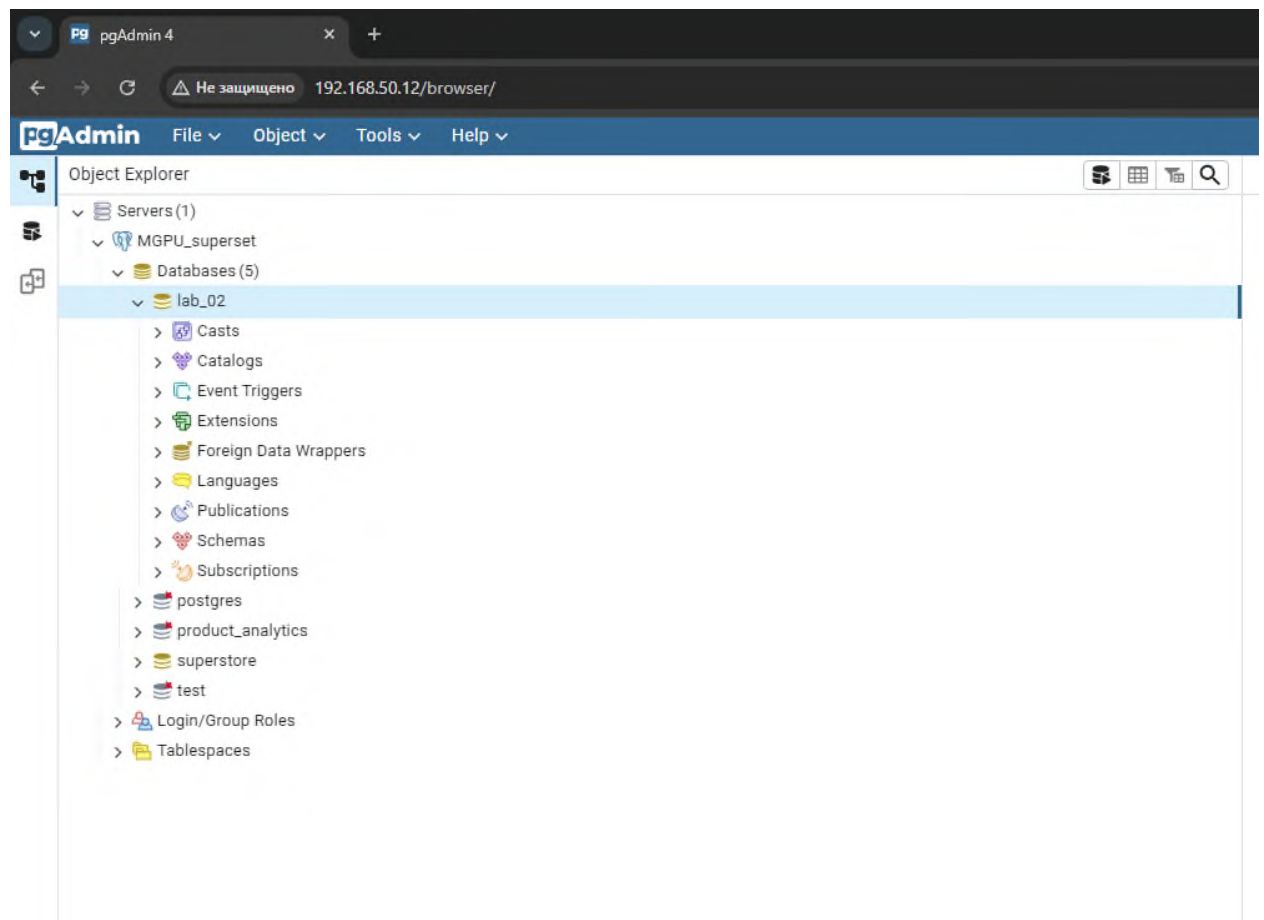
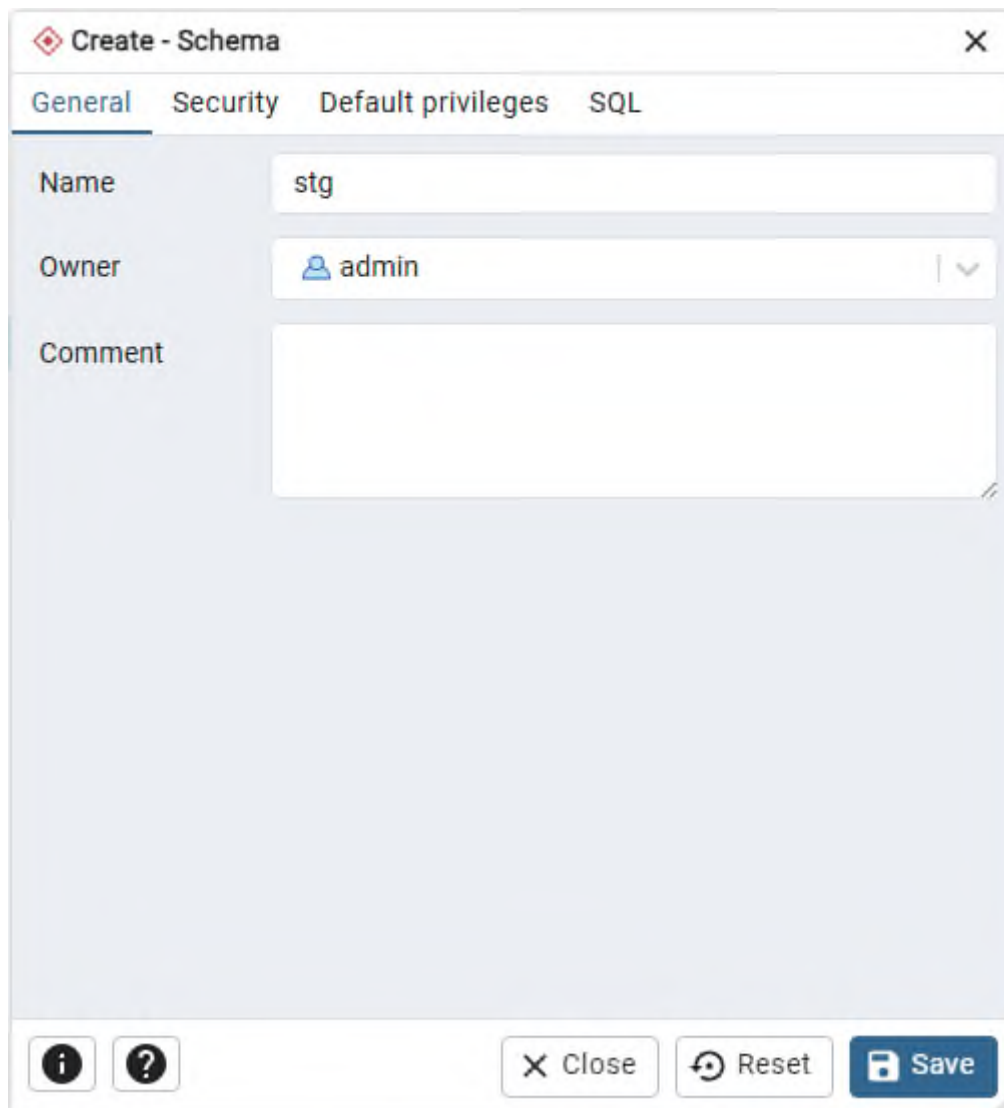


Рисунок 7 pgadmin в браузере на локальной машине

Создание Schema (stg):



Create - Schema

General Security Default privileges SQL

Name: stg

Owner: admin

Comment:

Close Reset Save

Рисунок 8 Создание stg (staging layer)

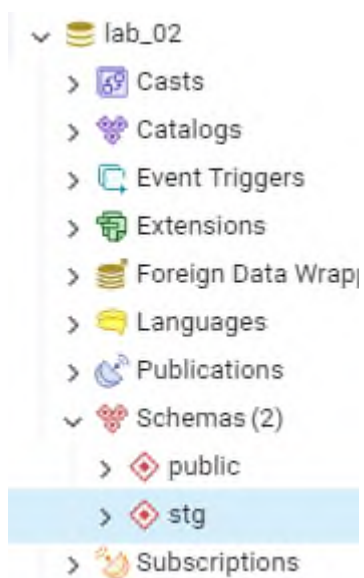


Рисунок 9 stg на общей схеме

Создание Schema (dw):

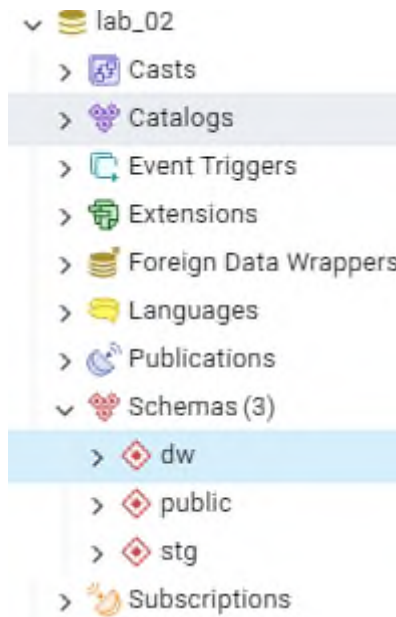


Рисунок 10 Создание dw (data warehouse layer)

Заполнение stg данными из stg.orders.sql

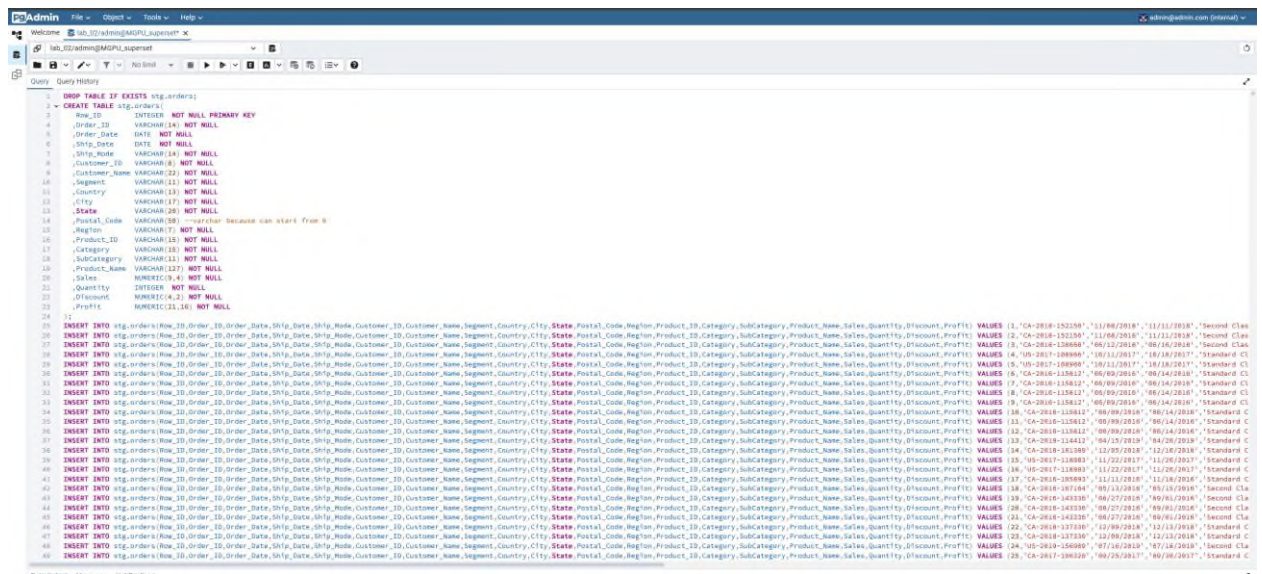


Рисунок 11 Заполнение stg данными

stg-Tables-orders-Columns:

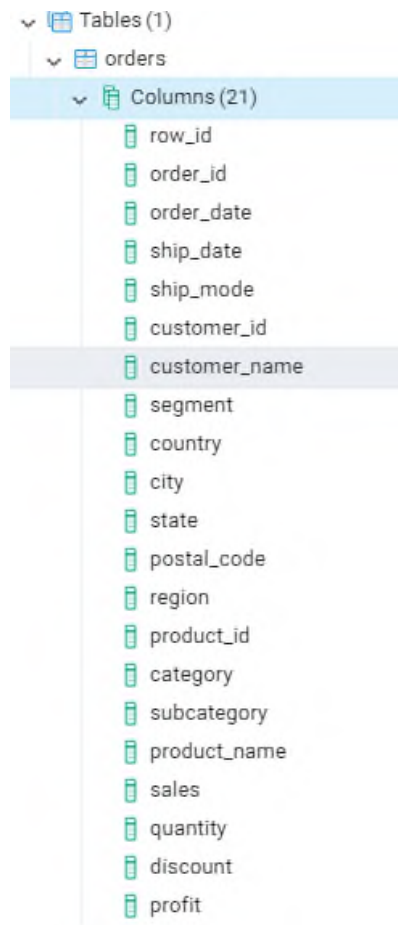


Рисунок 12 Состояние orders

Проверка количества данных:

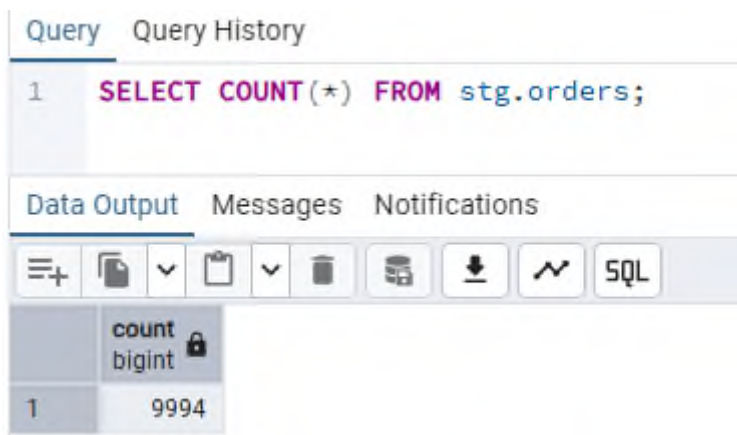


Рисунок 13 Проверка количества данных

public (raw data layer)

orders:

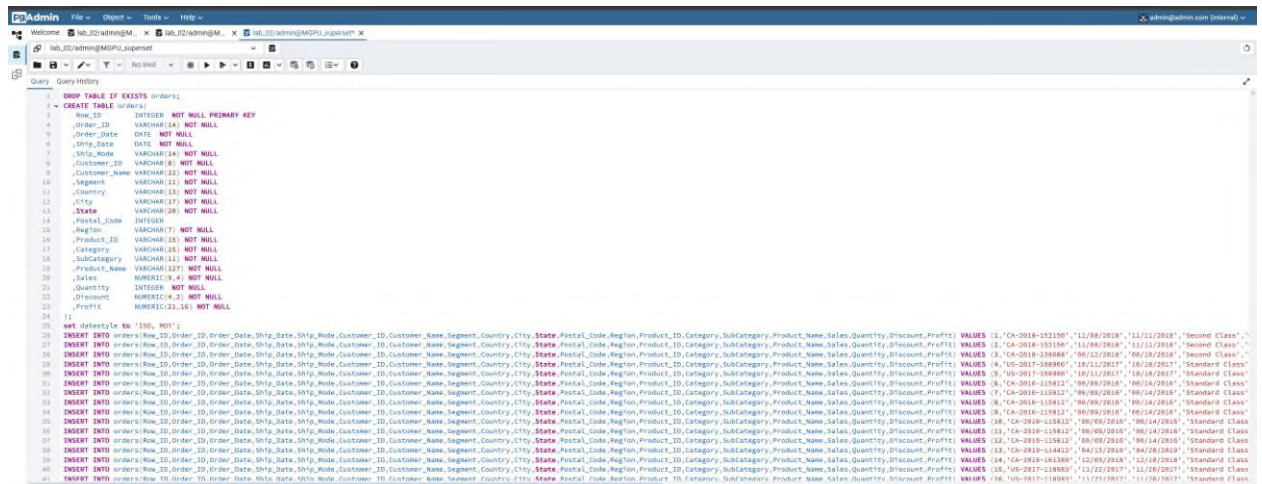


Рисунок 14 Заполнение orders

Проверка данных:

row_id	order_id	order_date	ship_date	ship_mode	customer_id	customer_name	segment	country	city	state	postal_code	region	product_id	category	subcategory	product_name	sales	quantity	discount	profit
1	CA-2018-152106	2018-11-08	2018-11-11	Second Class	DS-13208	Clara Gule	Consumer	United States	Portland	Ore	97208	South	PUR-BD-1000786	Furniture	Bedroom	Bedroom Collection Bedroom	1000000	10000	10	1000000
2	CA-2018-152106	2018-11-08	2018-11-11	Second Class	DS-13208	Clara Gule	Consumer	United States	Portland	Ore	97208	South	PUR-BD-1000786	Furniture	Bedroom	Bedroom Collection Bedroom	1000000	10000	10	1000000
3	CA-2018-152106	2018-11-08	2018-11-11	Second Class	DS-13208	Clara Gule	Consumer	United States	Portland	Ore	97208	South	PUR-BD-1000786	Furniture	Bedroom	Bedroom Collection Bedroom	1000000	10000	10	1000000
4	US-2017-108866	2017-10-11	2017-10-18	Standard Class	DS-25205	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311	South	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
5	US-2017-108866	2017-10-11	2017-10-18	Standard Class	DS-25205	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311	South	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
6	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
7	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
8	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
9	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
10	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
11	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
12	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
13	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
14	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
15	US-2017-108866	2017-10-11	2017-10-18	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
16	US-2017-108866	2017-10-11	2017-10-18	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
17	US-2017-108866	2017-10-11	2017-10-18	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
18	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
19	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
20	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
21	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
22	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
23	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
24	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
25	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000
26	CA-2018-152106	2018-11-08	2018-11-11	Standard Class	BS-11770	Bronia Hoffman	Consumer	United States	Los Angeles	California	90032	West	PUR-FA-1000077	Furniture	Tables	Bentley CH4500 Series Slim Rectangular Table	1000000	10000	10	1000000

Рисунок 15 Проверка данных в orders

Данные дублированы

people:

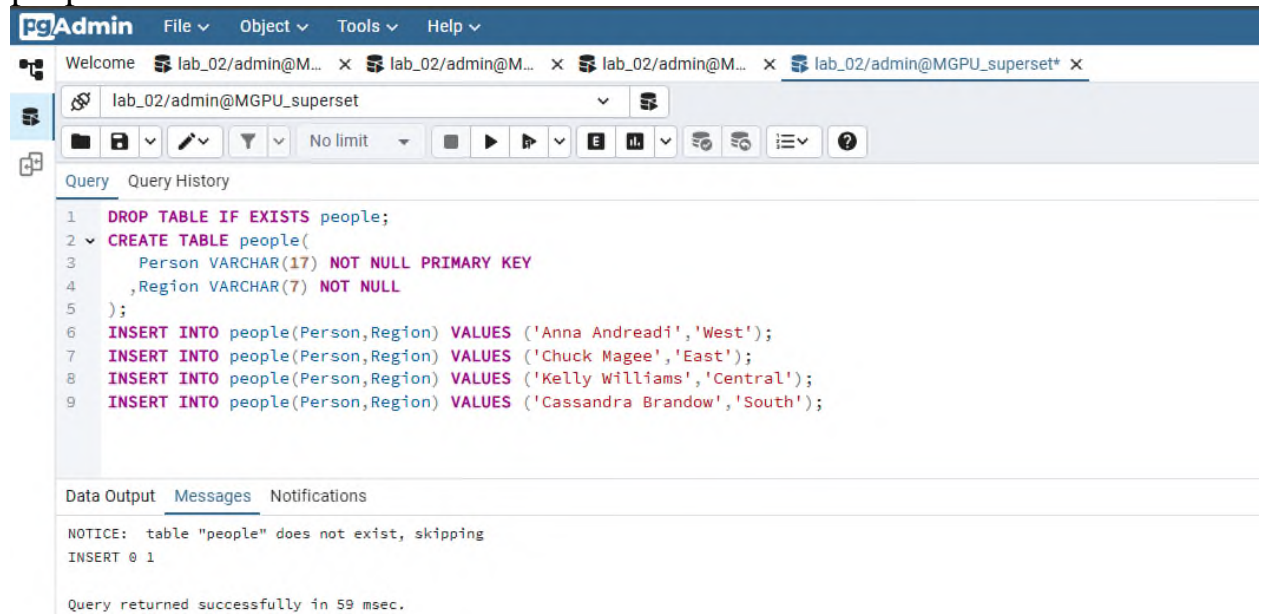


Рисунок 16 Заполнение people

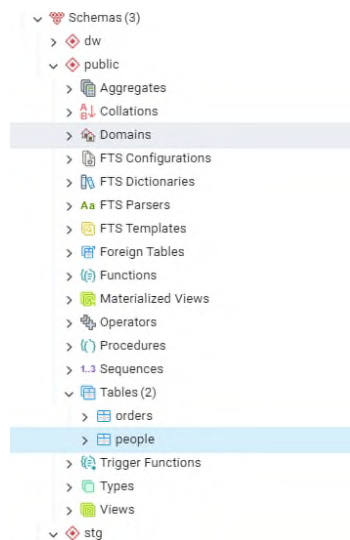


Рисунок 17 Состояние

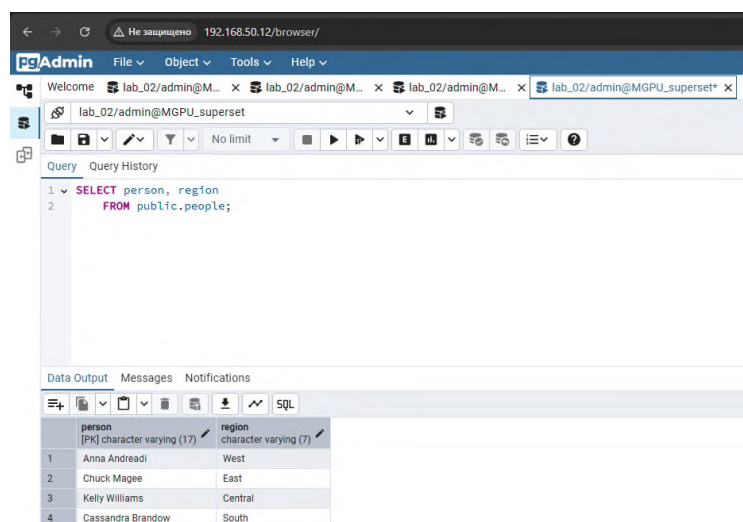


Рисунок 18 Проверка данных people

returns:

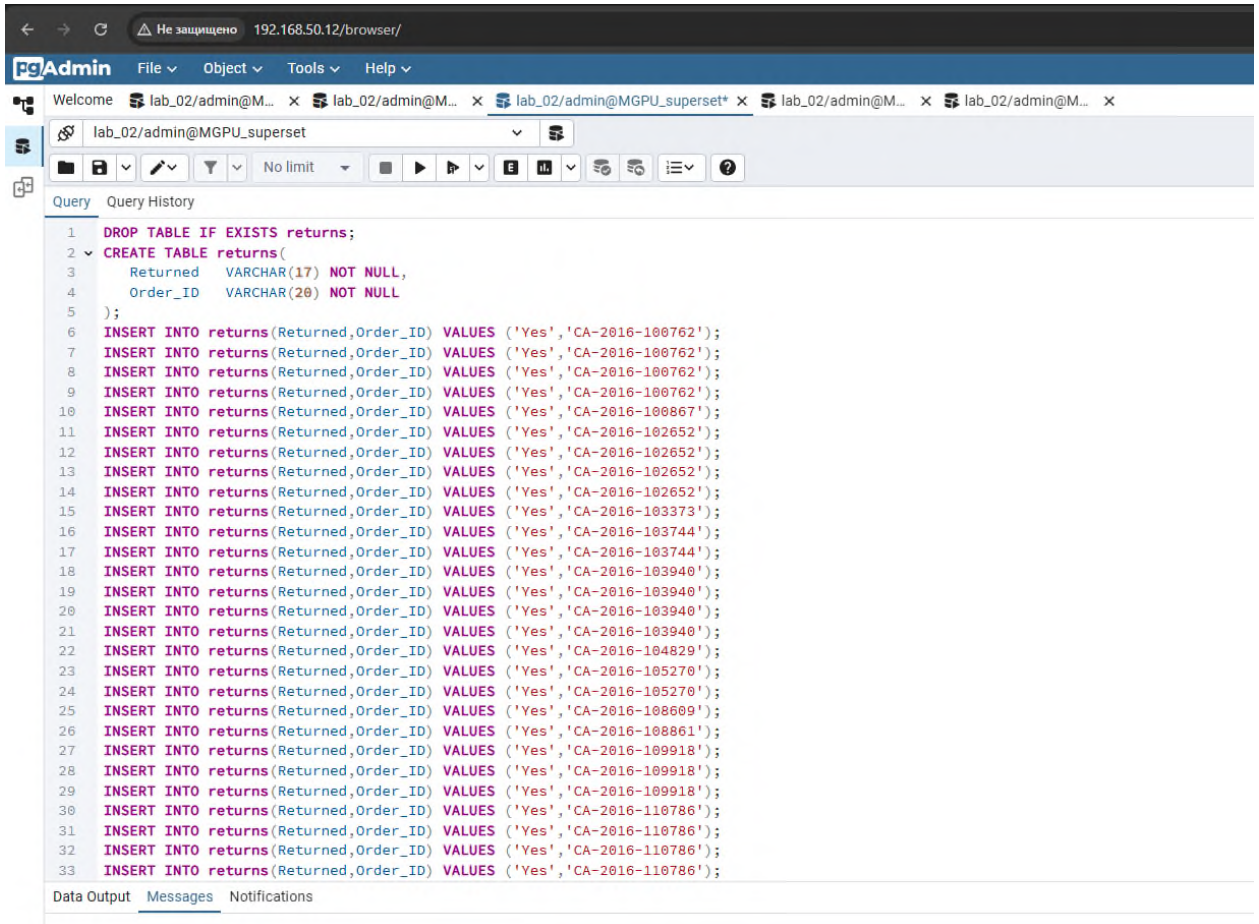


Рисунок 19 Заполнение данных returns

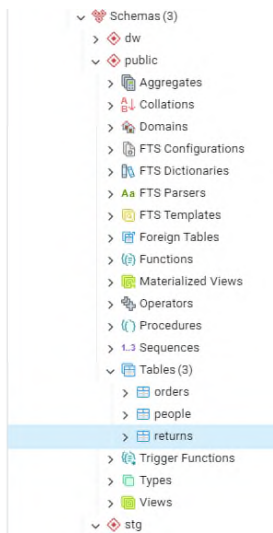


Рисунок 20 Состояние

← → ↻ Не защищено 192.168.50.12/browser/

pgAdmin File Object Tools Help

Welcome lab_02/admin@M... x lab_02/admin@M... x lab_02/admin@M... x lab_02/admin@M... x lab_02/admin@M... x lab_02/admin@MGPU_superset* x

lab_02/admin@MGPU_superset

Query Query History

```

1 SELECT returned, order_id
2 FROM public.returns;

```

Data Output Messages Notifications

	returned character varying (17)	order_id character varying (20)
1	Yes	CA-2016-100762
2	Yes	CA-2016-100762
3	Yes	CA-2016-100762
4	Yes	CA-2016-100762
5	Yes	CA-2016-100867
6	Yes	CA-2016-102652
7	Yes	CA-2016-102652
8	Yes	CA-2016-102652
9	Yes	CA-2016-102652
10	Yes	CA-2016-103373
11	Yes	CA-2016-103744
12	Yes	CA-2016-103744
13	Yes	CA-2016-103940
14	Yes	CA-2016-103940
15	Yes	CA-2016-103940
16	Yes	CA-2016-103940
17	Yes	CA-2016-104829
18	Yes	CA-2016-105270
19	Yes	CA-2016-105270
20	Yes	CA-2016-108609
21	Yes	CA-2016-108861
22	Yes	CA-2016-109918
23	Yes	CA-2016-109918
24	Yes	CA-2016-109918
25	Yes	CA-2016-110786
26	Yes	CA-2016-110786
27	Yes	CA-2016-110786
28	Yes	CA-2016-110786
29	Yes	CA-2016-110786
30	Yes	CA-2016-110786
31	Yes	CA-2016-110786
32	Yes	CA-2016-111871
33	Yes	CA-2016-116785
34	Yes	CA-2016-116785
35	Yes	CA-2016-123225
36	Yes	CA-2016-123225
37	Yes	CA-2016-123253

Total rows: 800 Query complete 00:00:00.042

Рисунок 21 Проверка данных returns

Dw (Data Warehouse layer)

скрипты из файла from_stg_to_dw.sql

Создание справочников:

- shipping_dim

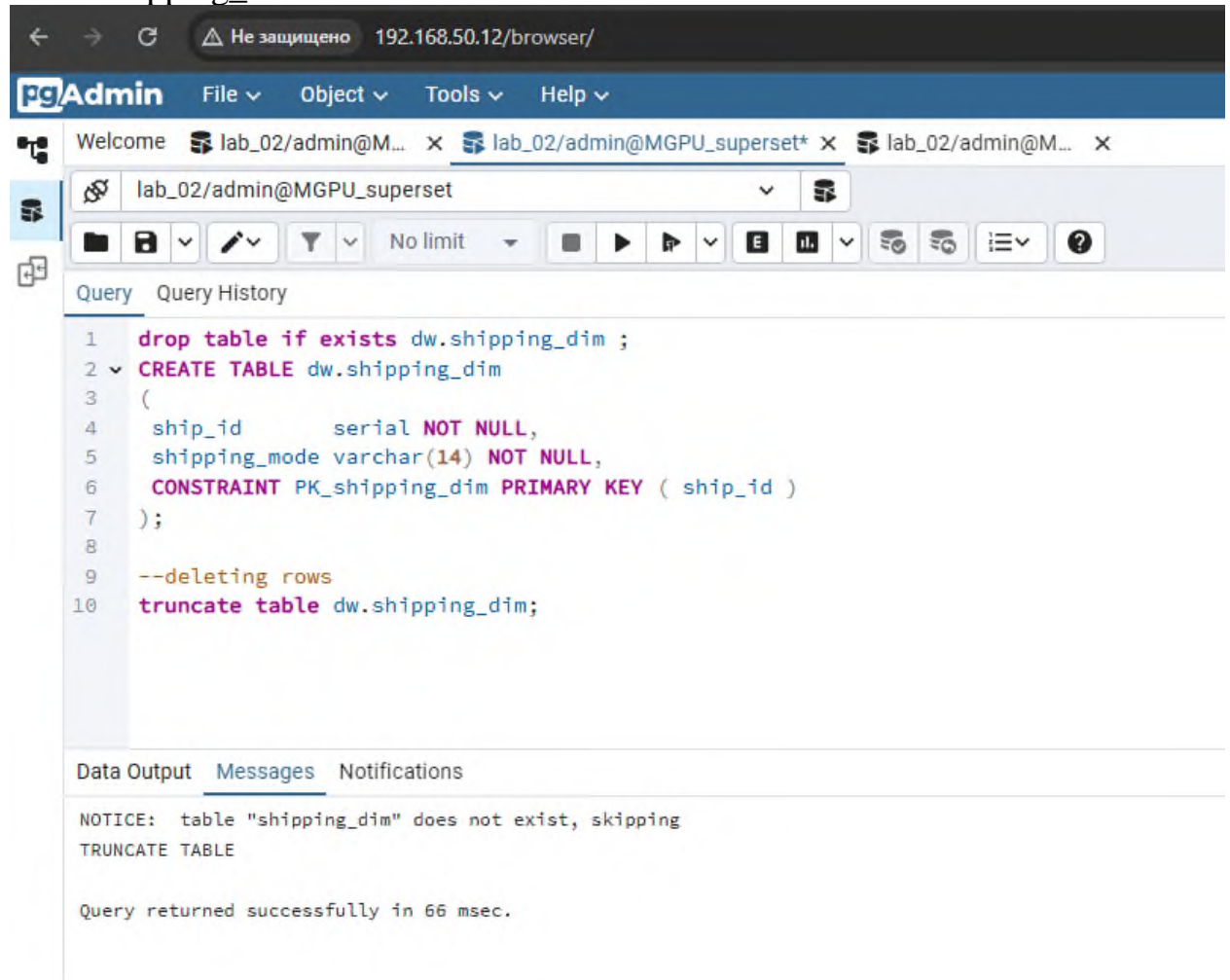


Рисунок 22 Создание справочника shipping_dim

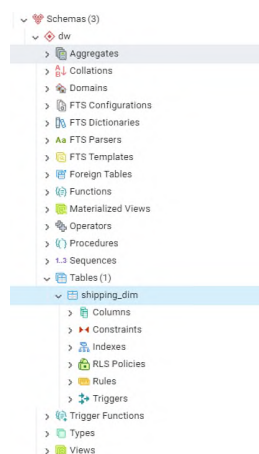


Рисунок 23 Справочник shipping_dim на схеме

Проверка и вывод

The screenshot shows the pgAdmin interface with a SQL query executed. The query is as follows:

```
1 --generating ship_id and inserting ship_mode from orders
2 insert into dw.shipping_dim
3 select 100+row_number() over(), ship_mode from (select distinct ship_mode from stg.orders ) a;
4 --checking
5 select * from dw.shipping_dim sd;
```

The results are displayed in a table with the following data:

	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

Рисунок 24 Проверка справочника shipping_dim

customer_dim

The screenshot shows the pgAdmin interface with a SQL query executed. The query is as follows:

```
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
14 select 100+row_number() over(), customer_id, customer_name from (select distinct customer_id, customer_name from stg.orders ) a;
15 --checking
16 select * from dw.customer_dim cd;
```

The results are displayed in a table with the following data:

	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 Kely
4	104	DW-13195	David Wiener
5	105	CL-11890	Carl Ludwig
6	106	VT-21700	Valerie Takahito
7	107	MH-18115	Mick Hernandez
8	108	HM-14860	Harry Marie
9	109	ME-18010	Michelle Ellison
10	110	KD-16345	Katherine Ducich

Рисунок 25 Создание и проверка справочника customer_dim

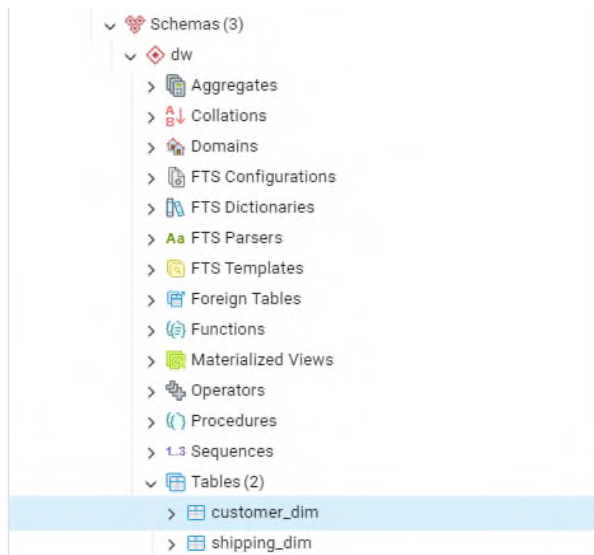


Рисунок 26 Справочник customer_dim на схеме

geo_dim

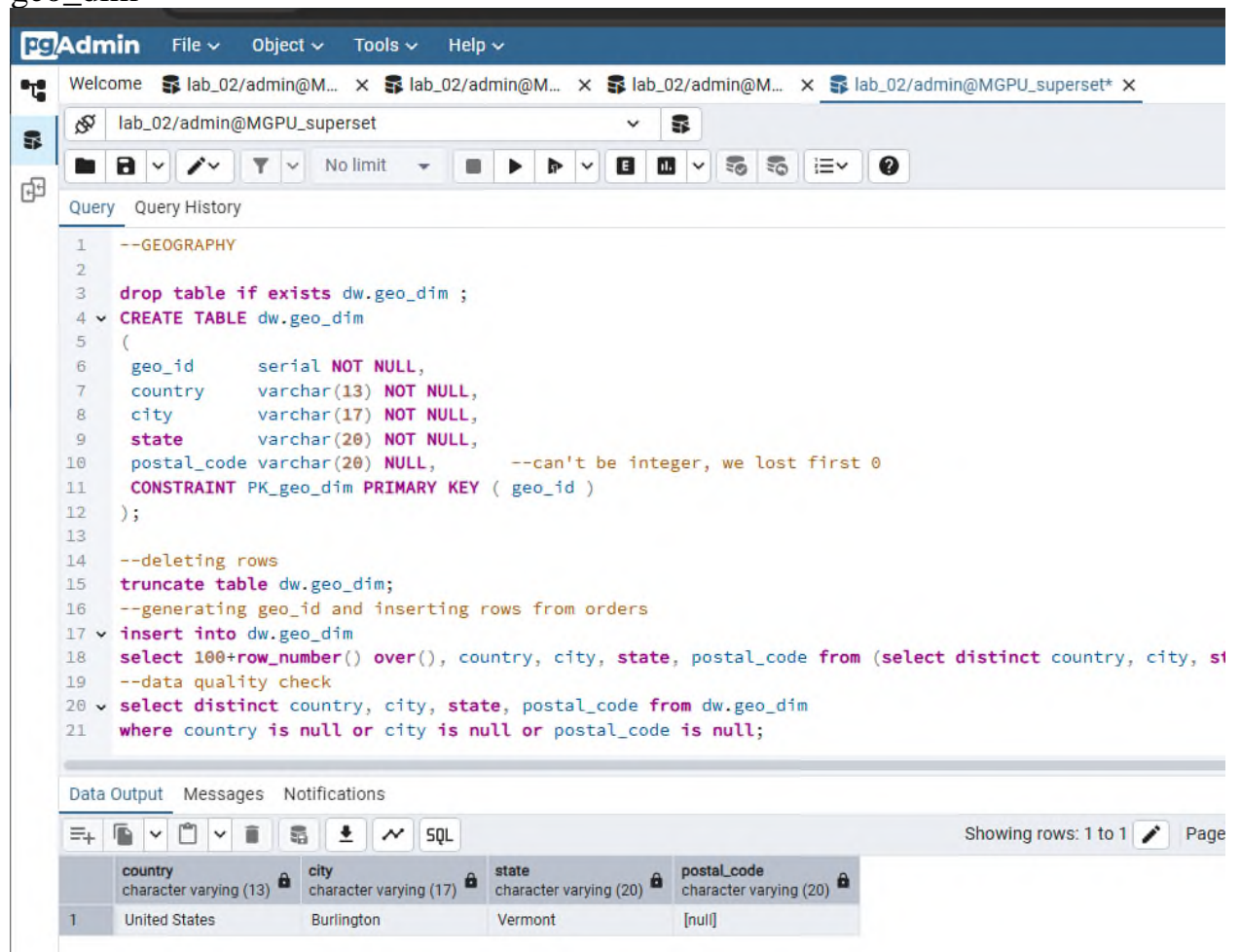


Рисунок 27 Создание и проверка справочника geo_dim

United States/Vermont/Burlington – нет почтового индекса
Обновление и проверка данных(почтового индекса) для Burlington:

pgAdmin

File Object Tools Help

Welcome lab_02/admin@M... lab_02/admin@M... lab_02/admin@M... lab_02/admin@M... lab_02/admin@MGPU_s...

lab_02/admin@MGPU_superset

No limit

Query Query History

```

1  update dw.geo_dim
2  set postal_code = '05401'
3  where city = 'Burlington' and postal_code is null;
4
5  --also update source file
6  update stg.orders
7  set postal_code = '05401'
8  where city = 'Burlington' and postal_code is null;
9
10
11 select * from dw.geo_dim
12 where city = 'Burlington'

```

Data Output Messages Notifications

Showing rows: 1 to 3 Page No:

	geo_id [PK] integer	country character varying (13)	city character varying (17)	state character varying (20)	postal_code character varying (20)
1	244	United States	Burlington	Iowa	52601
2	570	United States	Burlington	North Carolina	27217
3	264	United States	Burlington	Vermont	05401

Рисунок 28 Добавление почтового индекса для Burlington

Schemas (3)

- dw
 - Aggregates
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Operators
 - Procedures
 - Sequences
 - Tables (3)
 - customer_dim
 - geo_dim
 - shipping_dim

Рисунок 29 geo_dim на схеме

product_dim
создание и проверка:

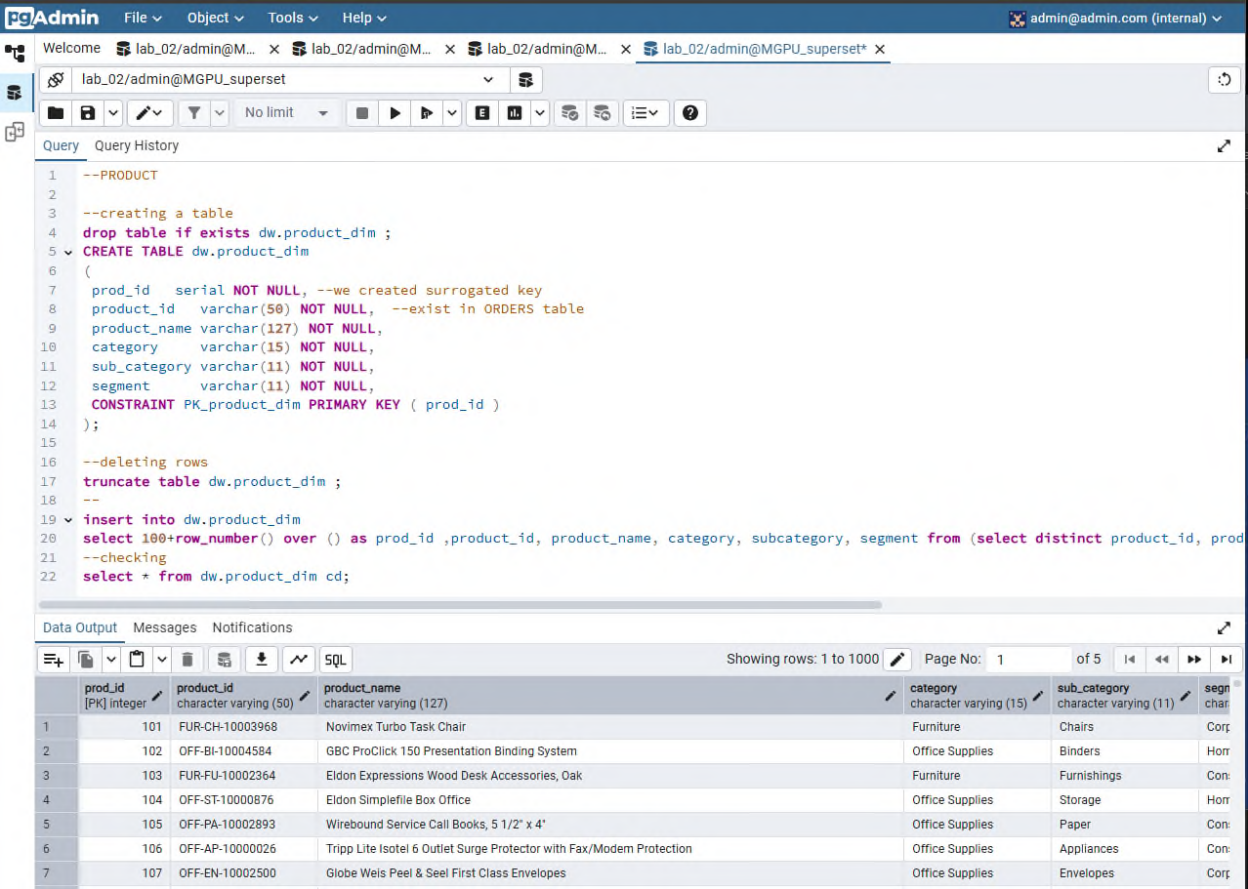


Рисунок 30 Создание и проверка справочника product_dim

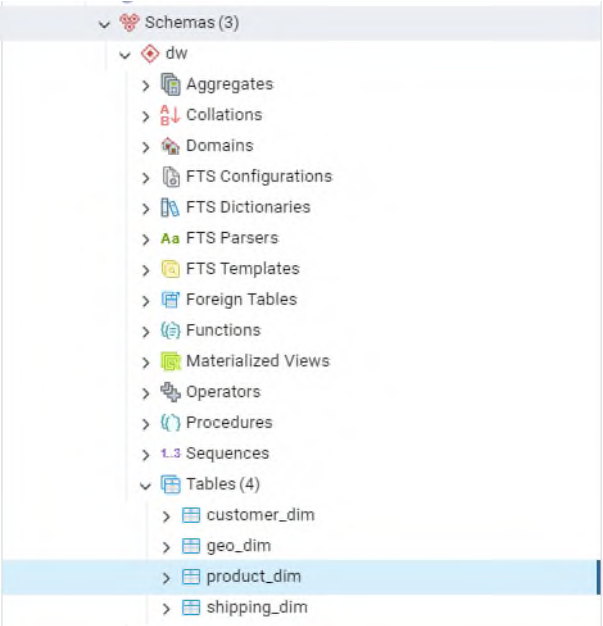


Рисунок 31 product_dim на схеме

calendar_dim

создание и проверка:

pgAdmin

FileObjectToolsHelp

Welcome lab_02/admin@MGPU_superset X

lab_02/admin@MGPU_superset

No limit

Query Query History

1

--CALENDAR

2

--creating a table

3

drop table if exists dw.calendar_dim ;

4

CREATE TABLE dw.calendar_dim

5

(

6

dateid serial NOT NULL,

7

year int NOT NULL,

8

quarter int NOT NULL,

9

month int NOT NULL,

10

week int NOT NULL,

11

date date NOT NULL,

12

week_day varchar(20) NOT NULL,

13

leap varchar(20) NOT NULL,

14

CONSTRAINT PK_calendar_dim PRIMARY KEY (dateid)

15

);

16

17

--deleting rows

18

truncate table dw.calendar_dim;

19

--

20

insert into dw.calendar_dim

21

select

22

to_char(date,'yyyymmdd')::int as date_id,

23

extract('year' from date)::int as year,

24

extract('quarter' from date)::int as quarter,

25

extract('month' from date)::int as month,

26

extract('week' from date)::int as week,

27

date::date,

28

to_char(date, 'dy') as week_day,

29

extract('day' from

30

(date + interval '2 month - 1 day')

31

) = 29

32

as leap

33

from generate_series(date '2000-01-01',

34

date '2030-01-01',

35

interval '1 day')

36

as t(date);

37

--checking

38

select * from dw.calendar_dim;

Data Output

Messages

Notifications

Showing rows: 1 to 1000

Page No: 1

	dateid [PK] integer	year integer	quarter integer	month integer	week integer	date date	week_day character varying (20)	leap character varying (20)
1	20000101	2000	1	1	52	2000-01-01	sat	true
2	20000102	2000	1	1	52	2000-01-02	sun	false
3	20000103	2000	1	1	1	2000-01-03	mon	false
4	20000104	2000	1	1	1	2000-01-04	tue	false
5	20000105	2000	1	1	1	2000-01-05	wed	false
6	20000106	2000	1	1	1	2000-01-06	thu	false
7	20000107	2000	1	1	1	2000-01-07	fri	false
8	20000108	2000	1	1	1	2000-01-08	sat	false
9	20000109	2000	1	1	1	2000-01-09	sun	false
10	20000110	2000	1	1	2	2000-01-10	mon	false
11	20000111	2000	1	1	2	2000-01-11	tue	false

Total rows: 10959 Query complete 00:00:00.101

Рисунок 32 Создание и проверка справочника calendar_dim

Schemas (3)

dw

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables (5)

calendar_dim

customer_dim

geo_dim

product_dim

shipping_dim

Рисунок 33 calendar_dim на схеме

Метрики sales_fact
Создание и проверка

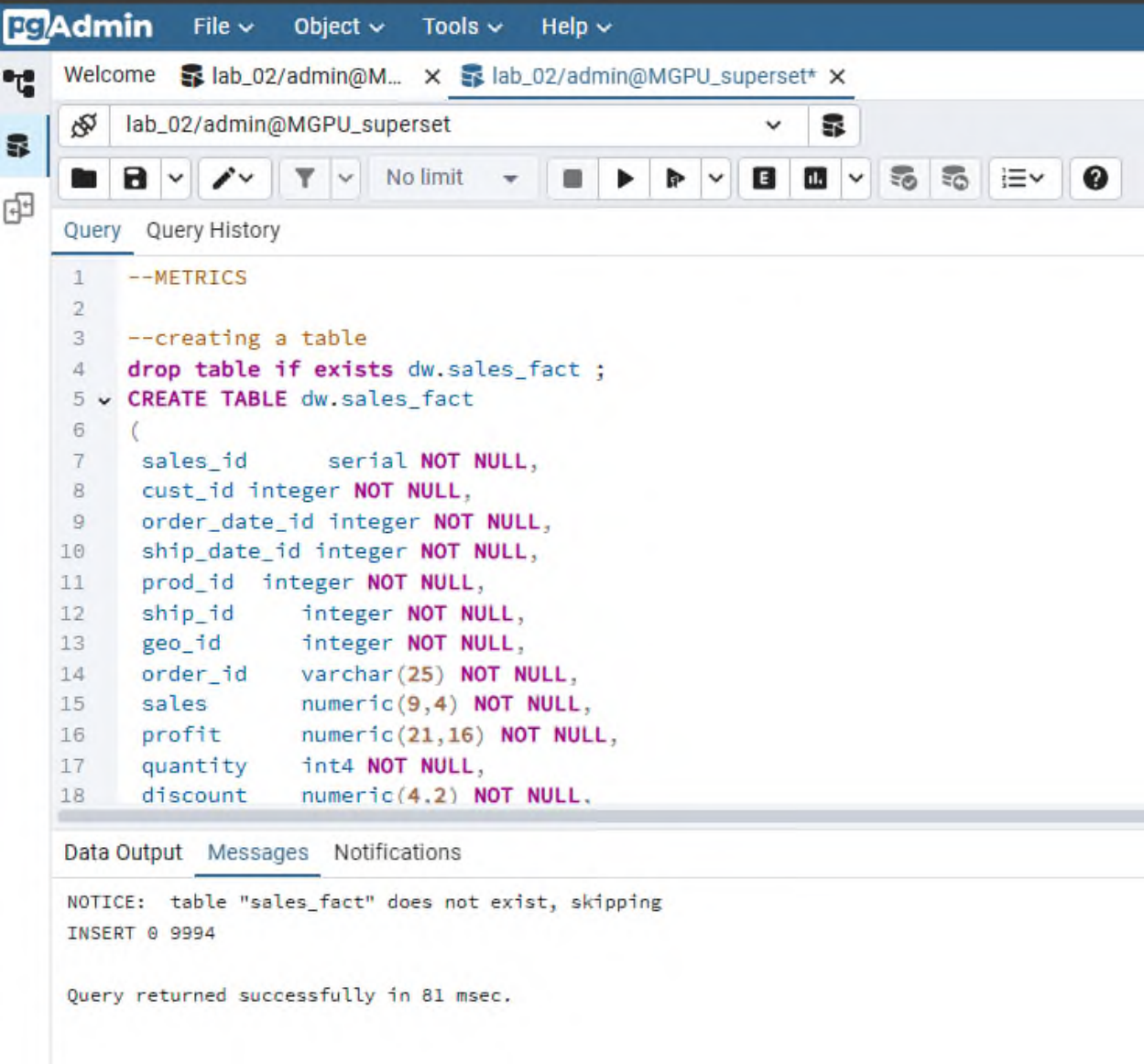


Рисунок 34 Создание метрик (sales_fact)

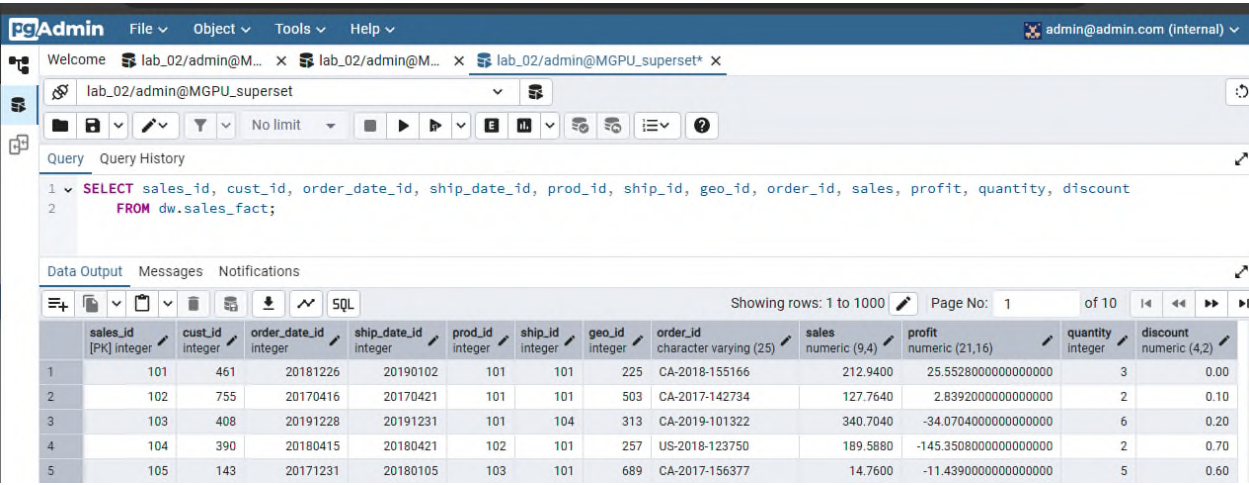


Рисунок 35 Проверка метрик (sales_fact)

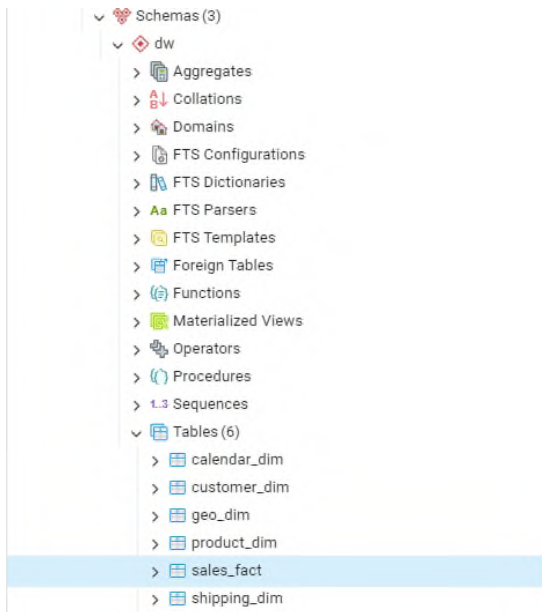


Рисунок 36 sales_fact на схеме

Проверка качества данных

Проверка количества данных:

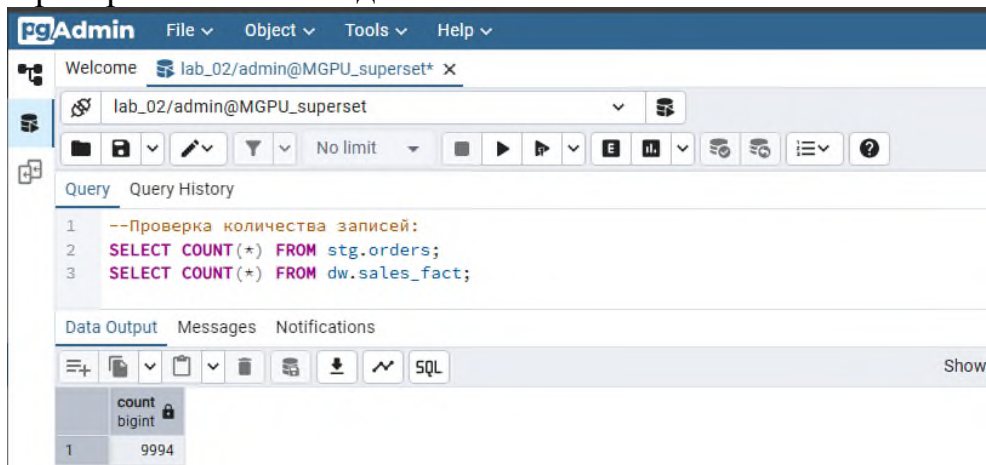


Рисунок 37 Проверка количества данных

Проверка целостности данных:

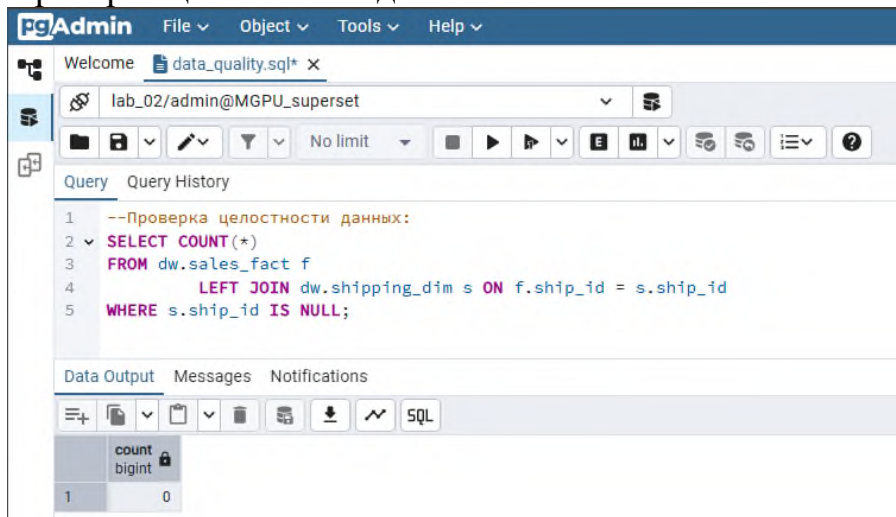


Рисунок 38 Проверка целостности данных

Проверка корректности агрегатов:

The screenshot shows the pgAdmin interface with a SQL query editor. The query is as follows:

```
1  -- Проверка корректности агрегатов
2  SELECT
3      SUM(sales) as total_sales,
4      SUM(profit) as total_profit
5  FROM stg.orders;
```

Below the query editor, the 'Data Output' tab is active, displaying the results of the query in a table format:

	total_sales numeric	total_profit numeric
1	2297200.8603	286397.0216999999887055

Рисунок 39 Проверка корректности агрегатов

The screenshot shows the pgAdmin interface with a SQL query editor. The query is as follows:

```
1  -- Проверка корректности агрегатов
2  SELECT
3      SUM(sales) as total_sales,
4      SUM(profit) as total_profit
5  FROM dw.sales_fact;
```

Below the query editor, the 'Data Output' tab is active, displaying the results of the query in a table format:

	total_sales numeric	total_profit numeric
1	2297200.8603	286397.0216999999887055

Рисунок 40 Рисунок 39 Проверка корректности агрегатов

Индивидуальное задание. Вариант 13

Задание:

1. Создать представление по клиентам.

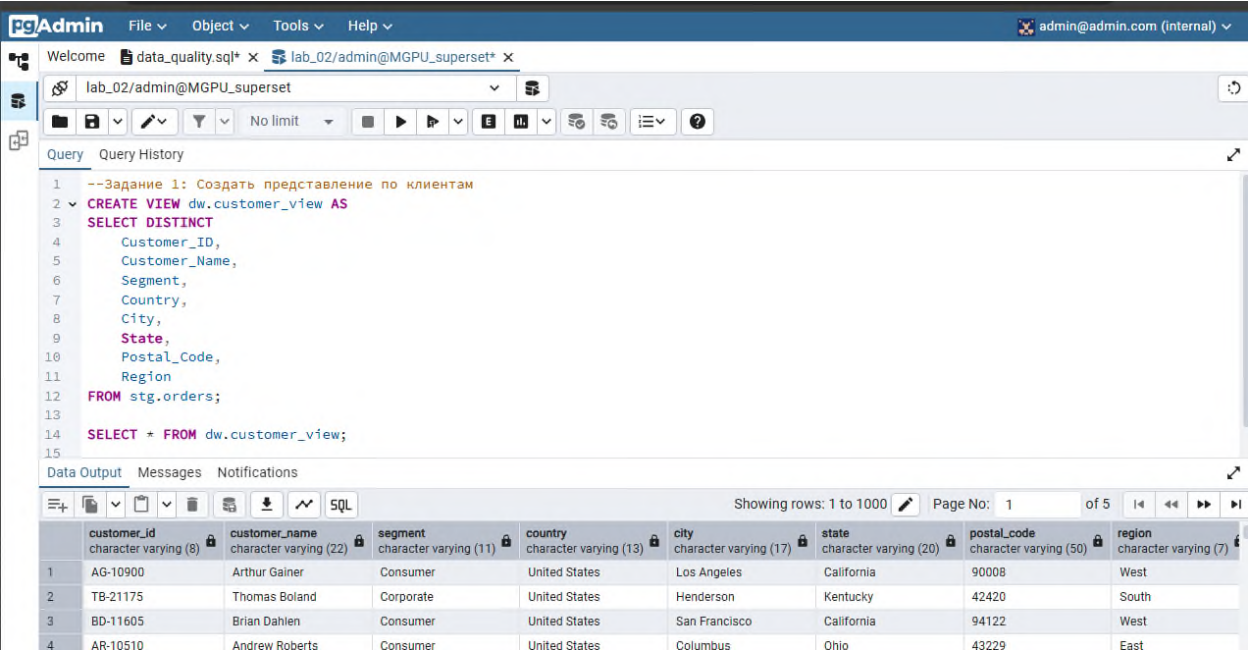


Рисунок 41 Создание представления по клиентам

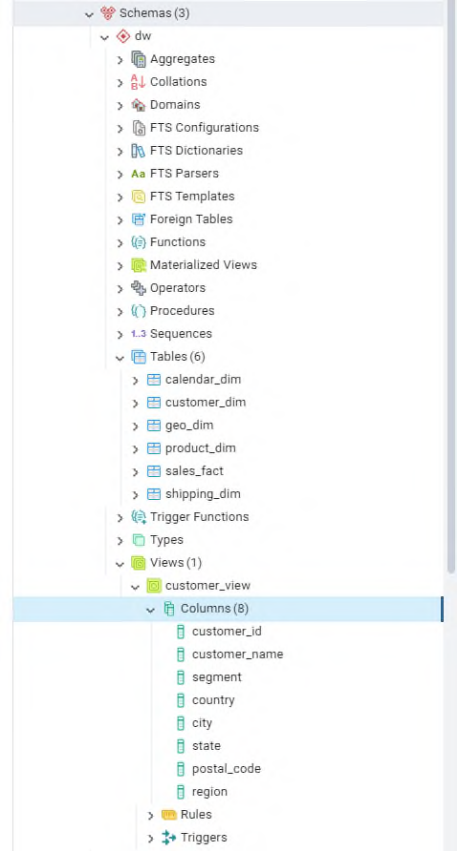


Рисунок 42 View1 на схеме

2. Определить продажи по способам доставки.

The screenshot shows the pgAdmin interface with the following components:

- Top Bar:** pgAdmin logo, File, Object, Tools, Help menus.
- Tab Bar:** Welcome, data_quality.sql*, lab_02/admin@M..., lab_02/admin@MGPU_superset*.
- Toolbar:** Icons for file operations, filters, and execution. A dropdown menu shows 'No limit'.
- Query Editor:** Contains the following SQL query:

```
1  --Задание 2: Определить продажи по способам доставки
2  CREATE VIEW dw.sales_by_delivery_method AS
3  SELECT
4      ship_mode,
5      SUM(sales) AS total_sales
6  FROM
7      stg.orders
8  GROUP BY
9      orders.ship_mode;
10
11 SELECT * FROM dw.sales_by_delivery_method;
```
- Data Output:** A table with 2 columns: ship_mode (character varying (14)) and total_sales (numeric). It contains 4 rows of data.

	ship_mode	total_sales
1	Standard Class	1358215.7430
2	Second Class	459193.5694
3	Same Day	128363.1250
4	First Class	351428.4229

Рисунок 43 Определение продажи по способам доставки

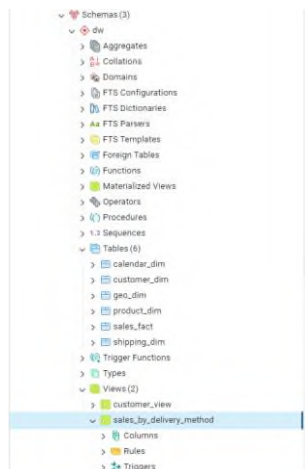


Рисунок 44 View2 на схеме

3. Рассчитать среднюю прибыль по городам.

The screenshot shows the pgAdmin interface with a SQL query executed. The query creates a view named `dw.avg_profit_by_city` and then selects all data from it. The results are displayed in a table with two columns: `city` and `average_profit`.

```
1 --Задание 3: Рассчитать среднюю прибыль по городам
2 CREATE VIEW dw.avg_profit_by_city AS
3 SELECT
4   city,
5   AVG(profit) AS average_profit
6 FROM
7   stg.orders
8 GROUP BY
9   city;
10 SELECT * FROM dw.avg_profit_by_city
11
```

	city character varying (17)	average_profit numeric
1	Norfolk	8.0178000000000000
2	New Bedford	10.0972200000000000
3	Littleton	-98.8018000000000000
4	Meriden	42.7473818181818182
5	Westland	29.9691923076923069
6	Springfield	38.0410883435582834
7	San Gabriel	96.9651000000000000
8	Redlands	28.3847250000000000
9	Glendale	7.9504260869565213

Рисунок 45 Расчёт средней прибыли по городам

The screenshot shows the pgAdmin interface with a SQL query that selects all data from the `dw.avg_profit_by_city` view, ordered by `city` in ascending order. The results are displayed in a table with two columns: `city` and `average_profit`.

```
1 SELECT * FROM dw.avg_profit_by_city
2 ORDER BY city ASC;
```

	city character varying (17)	average_profit numeric
1	Aberdeen	6.6300000000000000
2	Abilene	-3.7584000000000000
3	Akron	-8.8874095238095288
4	Albuquerque	45.2920071428571429
5	Alexandria	19.9136437500000063
6	Allen	-9.9693750000000008
7	Allentown	-32.3500571428571430
8	Altoona	-0.591750000000000000
9	Amarillo	-38.7968300000000000
10	Anaheim	45.7038370370370444
11	Andover	31.0471500000000000

Рисунок 46 Вывод городов в алфавитном порядке

Контрольные вопросы:

1. В чем разница между схемами stg и dw?

Staging – временная зона для сырых данных. Хранит неизменённые копии данных из разных источников. Если dw “сломается”, staging позволяет перезагрузить данные без обращения к исходным системам (буфет первичной обработки).

DW (data warehouse) – хранилище очищенных, преобразованных и структурированных данных для анализа (отчётов, дашбордов и тд). DW содержит справочники (измерения, dimensions), фактовые таблицы (facts).

Staging — это "песочница" для подготовки данных и страховка от потерь.

DW — слой для анализа, где справочники и факты работают вместе, обеспечивая скорость и удобство запросов.

2. Зачем нужны суррогатные ключи?

Суррогатные ключи помогают:

Изолировать DW от изменений в источниках, упрощают моделирование, обеспечивают уникальность, поддерживают историю изменений.

3. Какие преимущества дает деморализация данных в DW?

Скорость запросов, упрощение анализа, оптимизация для агрегаций, снижение нагрузки на СУБД.

Заключение

В результате выполнения данной работы были освоены ключевые принципы проектирования реляционных баз данных, включая концепцию многослойной архитектуры хранения данных, определение связей между сущностями и выбор оптимальных типов данных. Практическая реализация этапов загрузки данных позволила закрепить навыки работы с SQL-запросами. Полученные знания и опыт являются основой для дальнейшего изучения расширенных возможностей СУБД, таких как оптимизация запросов, транзакции и администрирование баз данных.