

Домашнее задание 1. Создание и нормализация базы данных

Дана база данных **customer_and_transaction.xlsx**

В ней содержатся две таблицы:

1. **transaction**
2. **customer**

Таблица **transaction** содержит следующие колонки (слева укажем предполагаемый тип данных в синтаксисе dbdiagram.io / не PostgreSQL):

1. transaction_id – integer [primary key]
2. product_id – integer [not null]
3. customer_id – integer [not null]
4. transaction_date – date [not null]
5. online_order – boolean
6. order_status – varchar(50) [not null]
7. brand – varchar(100) [not null]
8. product_line – varchar(50)
9. product_class – varchar(50)
10. product_size – varchar(50)
11. list_price – decimal(10,2) [not null]
12. standard_cost – decimal(10,2)

Таблица **customer** содержит следующие колонки:

1. customer_id – integer [primary key]
2. first_name – varchar(100) [not null]
3. last_name – varchar(100) [not null]
4. gender – varchar(10)
5. DOB – date
6. job_title – varchar(100)
7. job_industry_category – varchar(100)
8. wealth_segment – varchar(50)
9. deceased_indicator – varchar(10) [not null]
10. owns_car – varchar(10)
11. address – text
12. postcode – varchar(20)
13. state – varchar(50)
14. country – varchar(50)
15. property_valuation – integer

Анализ таблиц

Таблица **transaction**:

transaction_id – первичный ключ (уникальный идентификатор транзакции).

Функциональные зависимости от transaction_id:

transaction_id → (product_id,
customer_id,

transaction_date,
online_order,
order_status,
brand,
product_line,
product_class,
product_size,
list_price,
standard_cost)

Частичные зависимости:

product_id → (brand,
product_line,
product_class,
product_size,
list_price,
standard_cost)

customer_id → (все атрибуты из таблицы customer)

Таблица **customer**:

customer_id – первичный ключ (уникальный идентификатор покупателя).

Функциональные зависимости:

customer_id → (first_name,
last_name,
gender,
DOB,
job_title,
job_industry_category,
wealth_segment,
deceased_indicator,
owns_car,
address,
postcode)

Эта база данных имеет 1НФ (первую нормальную форму) потому что:

1. Все атрибуты атомарны (каждое поле содержит только одно значение).
2. Нет повторяющихся групп (в таблицах нет столбцов, которые повторяют одну и ту же информацию в разных колонках).
3. Есть первичный ключ.

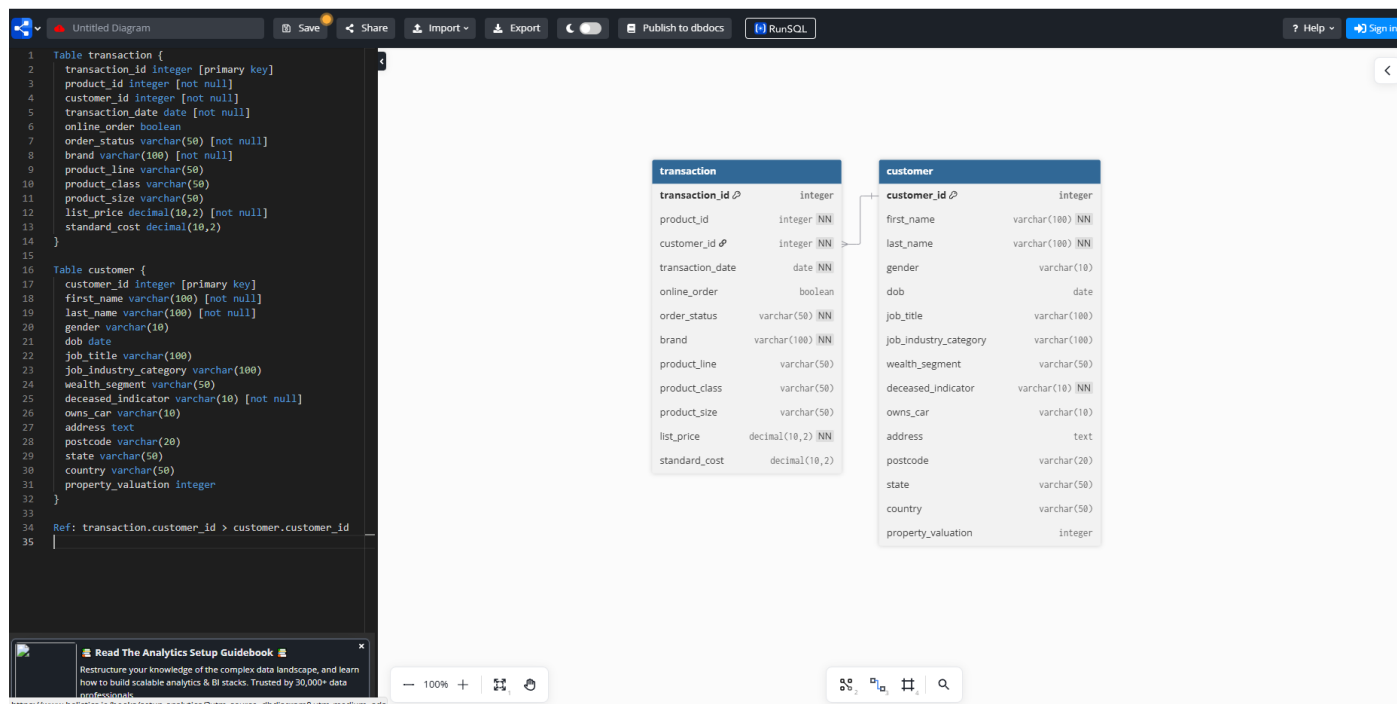


Рисунок 1 – 1НФ (исходный вариант)

Приведем данные к 2НФ (второй нормальной форме).

Мы выделили в таблице **transaction** зависимость:

product_id → (brand,
product_line,
product_class,
product_size,
list_price,
standard_cost)

Преобразуем таблицу **transaction** в две таблицы **transaction** и **product**.

Таблицу **customer** оставим как есть.

Таблица **transaction** содержит следующие колонки:

1. transaction_id – integer [primary key]
2. product_id – integer [not null]
3. customer_id – integer [not null]
4. transaction_date – date [not null]
5. online_order – boolean
6. order_status – varchar(50) [not null]

Таблица **product** содержит следующие колонки:

1. product_id – integer [primary key]
2. brand – varchar(100) [not null]
3. product_line – varchar(50)
4. product_class – varchar(50)
5. product_size – varchar(50)
6. list_price – decimal(10,2) [not null]
7. standard_cost – decimal(10,2)

Таблица **customer** содержит следующие колонки:

1. customer_id – integer [primary key]
2. first_name – varchar(100) [not null]

3. last_name – varchar(100) [not null]
4. gender – varchar(10)
5. DOB – date
6. job_title – varchar(100)
7. job_industry_category – varchar(100)
8. wealth_segment – varchar(50)
9. deceased_indicator – varchar(10) [not null]
10. owns_car – varchar(10)
11. address – text
12. postcode – varchar(20)
13. state – varchar(50)
14. country – varchar(50)
15. property_valuation – integer

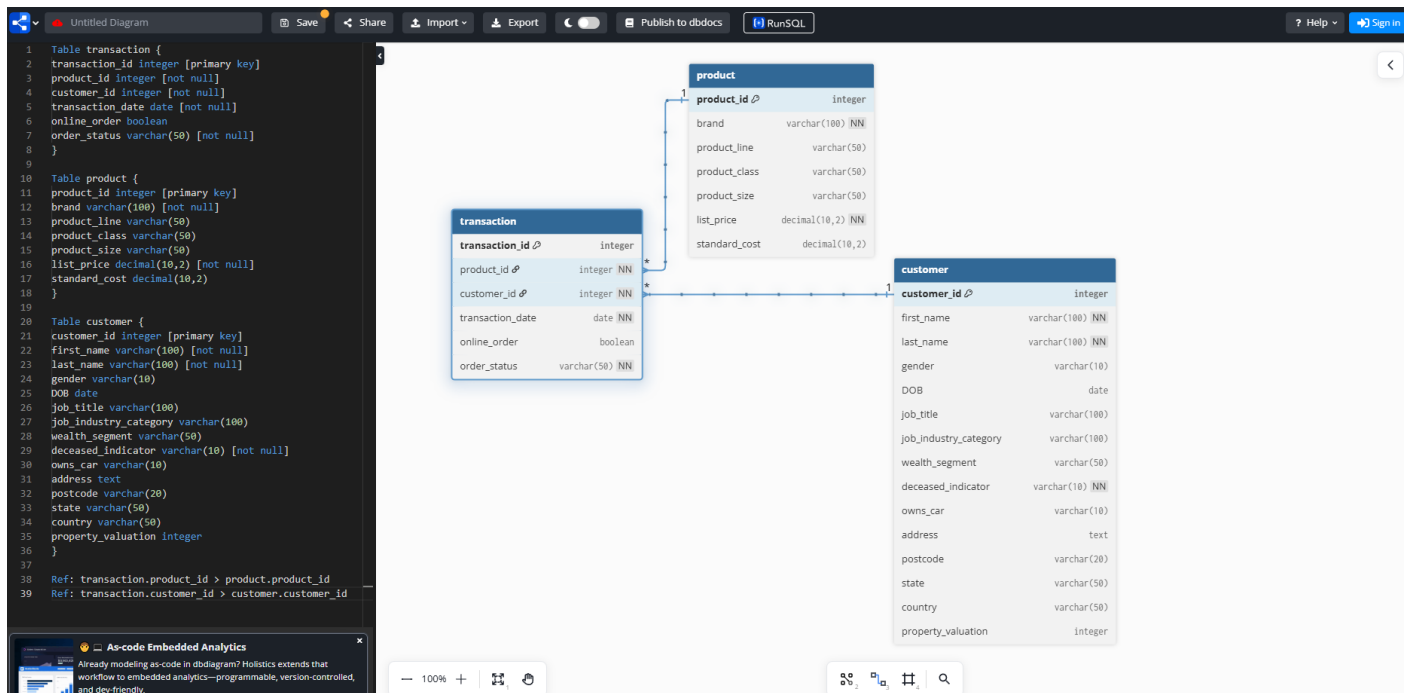


Рисунок 2 – 2НФ

Эта база данных имеет 2НФ (вторую нормальную форму) потому что:

1. Соответствует всем требованиям 1НФ (все атрибуты атомарны, нет повторяющихся групп, есть первичный ключ)
2. Отсутствуют частичные зависимости:
 $product_id \rightarrow (brand, product_line, product_class, product_size, list_price, standard_cost)$
3. Каждый неключевой атрибут зависит от первичного ключа:
 В **transaction**: все атрибуты зависят от transaction_id
 В **customer**: все атрибуты зависят от customer_id
 В **product**: все атрибуты зависят от product_id

Приведем данные к 3НФ (третьей нормальной форме).

Для этого нужно установить транзитивные зависимости.

В таблице **customer** можно установить следующие зависимости:

- job_title – job_industry_category (относятся к работе покупателя)
- wealth_segment – property_valuation – owns_car (относятся к благосостоянию покупателя)
- address – postcode (относятся к адресу проживания покупателя)
- state – country (относятся к месту проживания покупателя)

Преобразуем таблицу customer в четыре таблицы: **customer**, **job_industry**, **wealth_segment** и **address**.

Таблицы **transaction** и **product** не меняем.

Таблица **transaction** содержит следующие колонки:

1. transaction_id – integer [primary key]
2. product_id – integer [not null]
3. customer_id – integer [not null]
4. transaction_date – date [not null]
5. online_order – boolean
6. order_status – varchar(50) [not null]

Таблица **product** содержит следующие колонки:

1. product_id – integer [primary key]
2. brand – varchar(100) [not null]
3. product_line – varchar(50)
4. product_class – varchar(50)
5. product_size – varchar(50)
6. list_price – decimal(10,2) [not null]
7. standard_cost – decimal(10,2)

Таблица **customer** содержит следующие колонки:

1. customer_id – integer [primary key]
2. first_name – varchar(100) [not null]
3. last_name – varchar(100) [not null]
4. gender – varchar(10)
5. DOB – date
6. job_industry_category_id – integer [not null]
7. wealth_segment_id – integer [not null]
8. deceased_indicator – varchar(10) [not null]
9. address_id – integer [not null]
10. property_valuation – integer

Таблица **job_industry** содержит следующие колонки:

1. job_industry_category_id – serial [primary key]
2. industry_name – varchar(100)
3. job_title – varchar(100)

Таблица **wealth_segment** содержит следующие колонки:

1. wealth_segment_id – serial [primary key]
2. segment_name – varchar(50) [not null]
3. owns_car – varchar(10)
4. mean_property_valuation – integer [not null]

Таблица **address** содержит следующие колонки:

1. address_id – serial [primary key]
2. address – text [not null]
3. postcode – varchar(20) [not null]
4. state – varchar(50) [not null]
5. country – varchar(50) [not null]

Эта база данных имеет 3НФ (третью нормальную форму) потому что:

1. Соответствует всем требованиям 1НФ и 2НФ (все атрибуты атомарны, нет повторяющихся групп, есть первичный ключ, нет частичных зависимостей)

2. Отсутствуют транзитивные зависимости:

job_title - job_industry_category (относятся к работе покупателя);

wealth_segment - property_valuation (относятся к благосостоянию покупателя);

address - postcode (относятся к адресу проживания покупателя);

state - country (относятся к месту проживанию покупателя).

3. Каждый неключевой атрибут зависит только от первичного ключа:

В **transaction**: все атрибуты зависят только от transaction_id

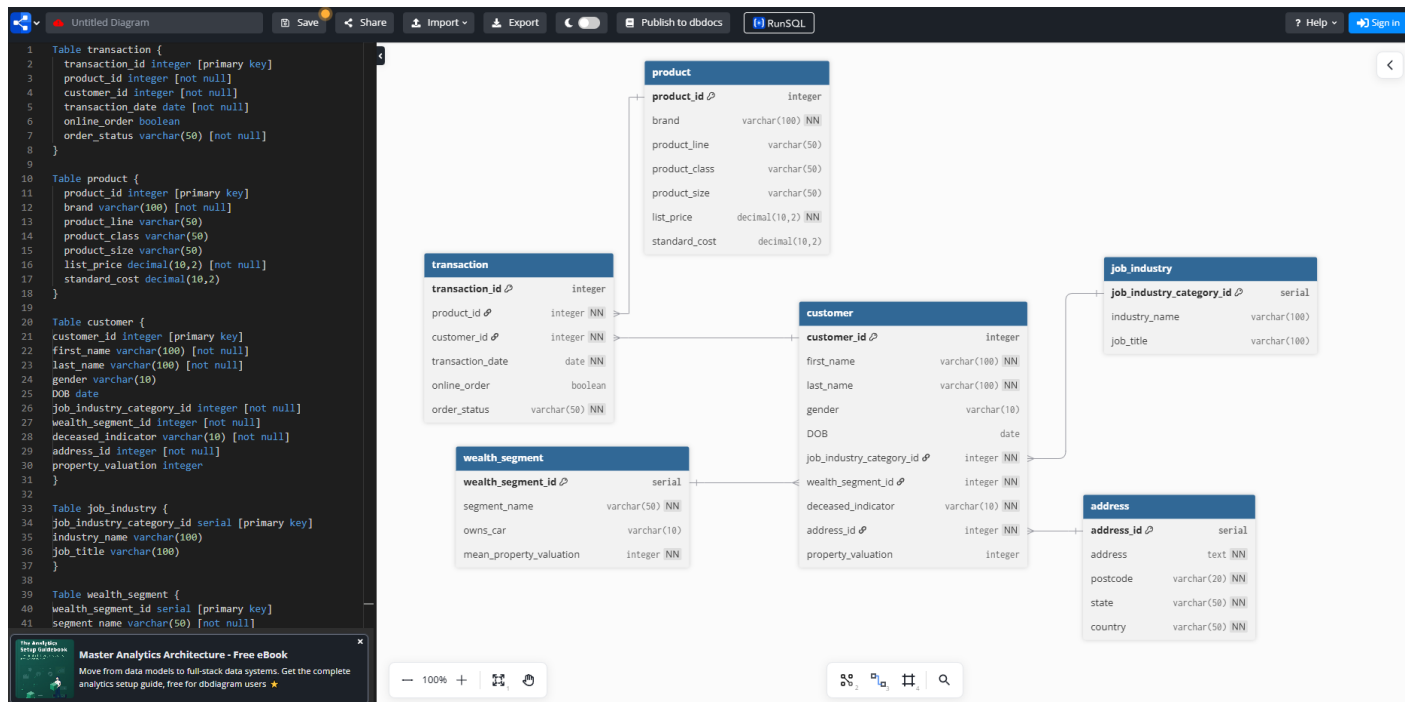
В **customer**: все атрибуты зависят только от customer_id

В **product**: все атрибуты зависят только от product_id

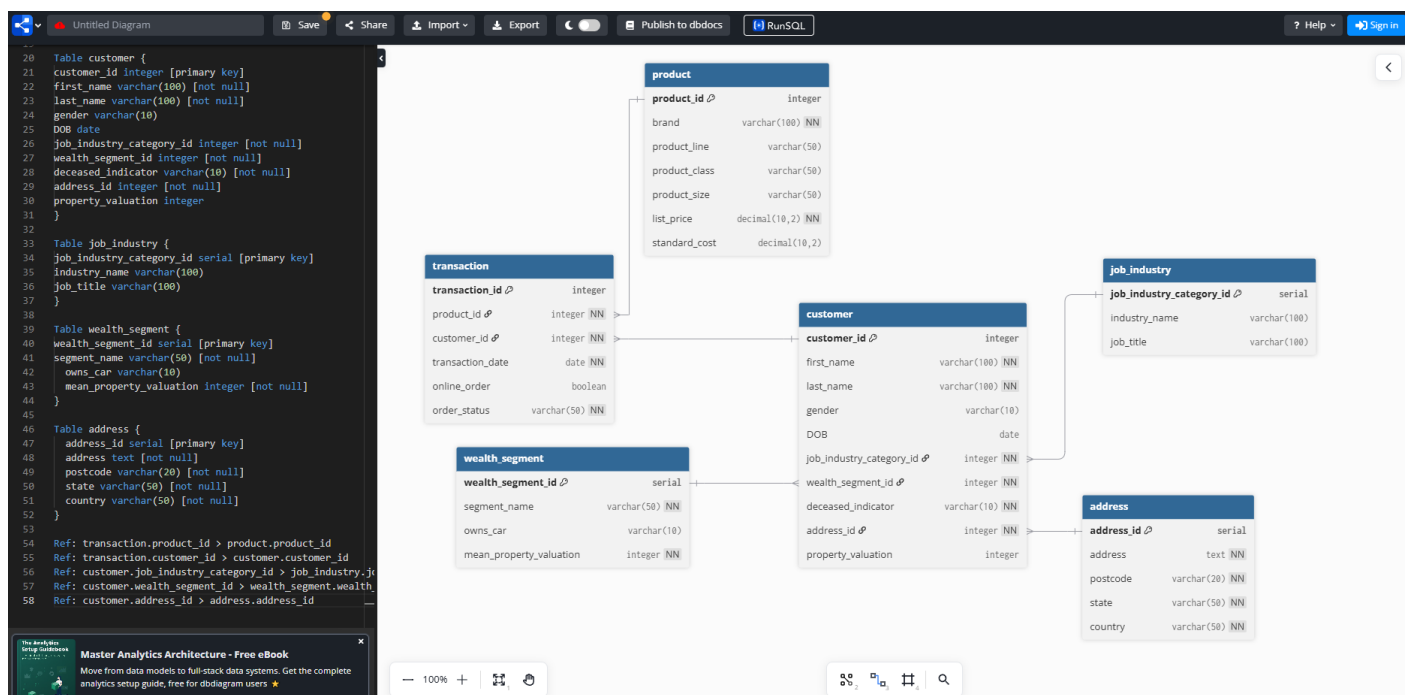
В **job_industry**: все атрибуты зависят только от job_industry_category_id

В **wealth_segment**: все атрибуты зависят только от wealth_segment_id

В **address**: все атрибуты зависят только от address_id



а) верхняя часть левого окна



б) нижняя часть левого окна

Рисунок 3 – ЗНФ

Далее работаем с базой данных в программе DBeaver 25.2.4.

Для начала сохраняем базу данных **customer_and_transaction.xlsx** в виде двух файлов: **customer.csv** и **transaction.csv**.

Создаем две таблицы **transaction_old** и **customer_old** с колонками, соответствующим ИФ.

Эти таблицы нужны только для того, чтобы импортировать данные.

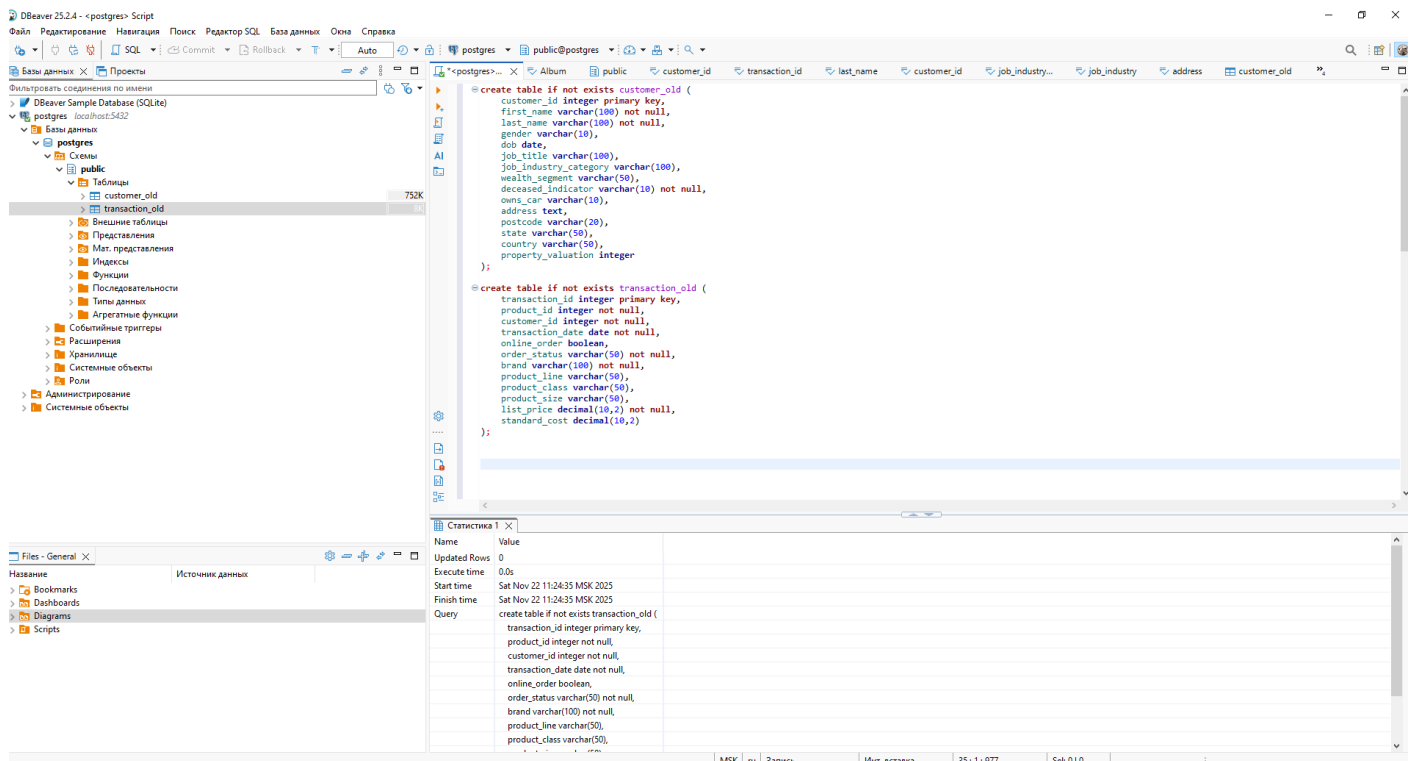


Рисунок 4 – Создание таблиц **transaction_old** и **customer_old**

Далее импортируем данные из **customer.csv** и **transaction.csv**.

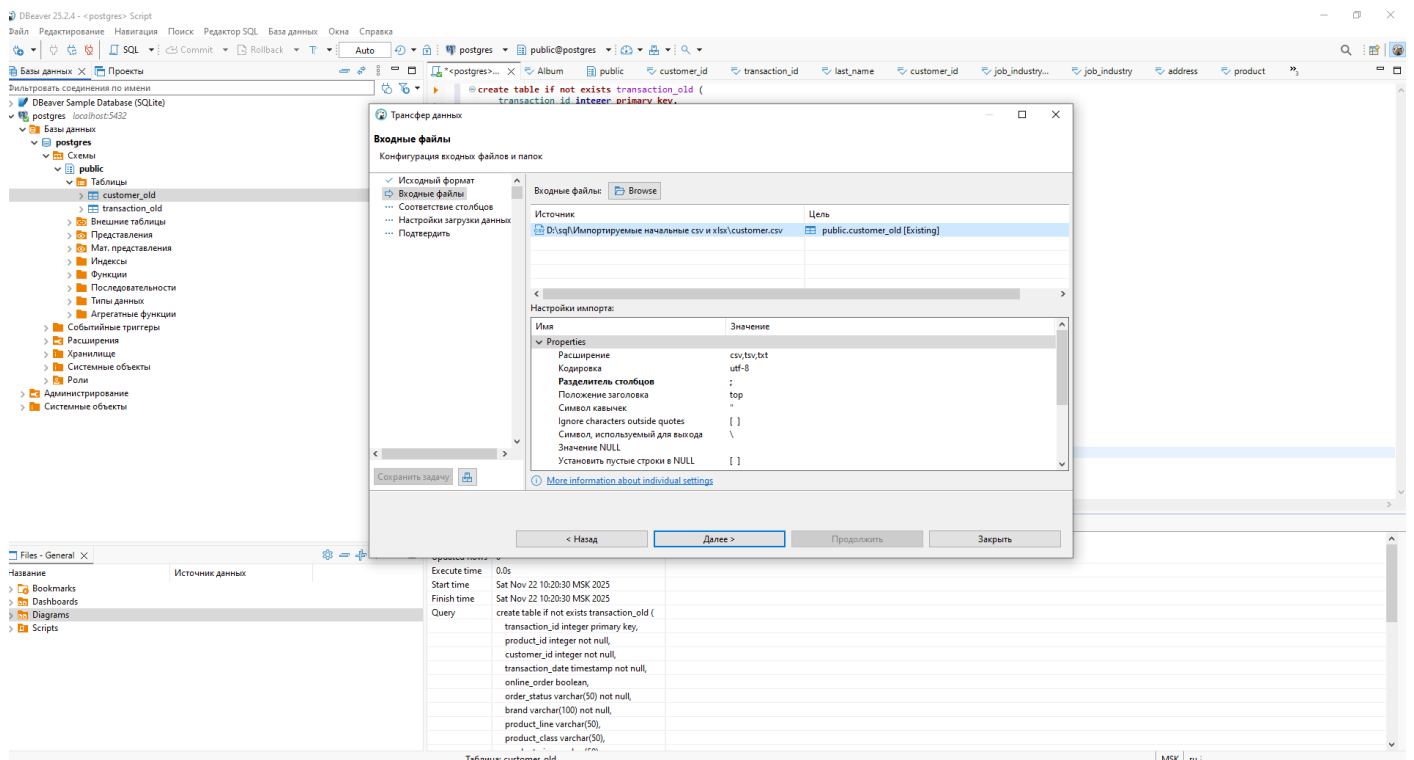


Рисунок 5 – Импорт данных в таблицу **customer_old**

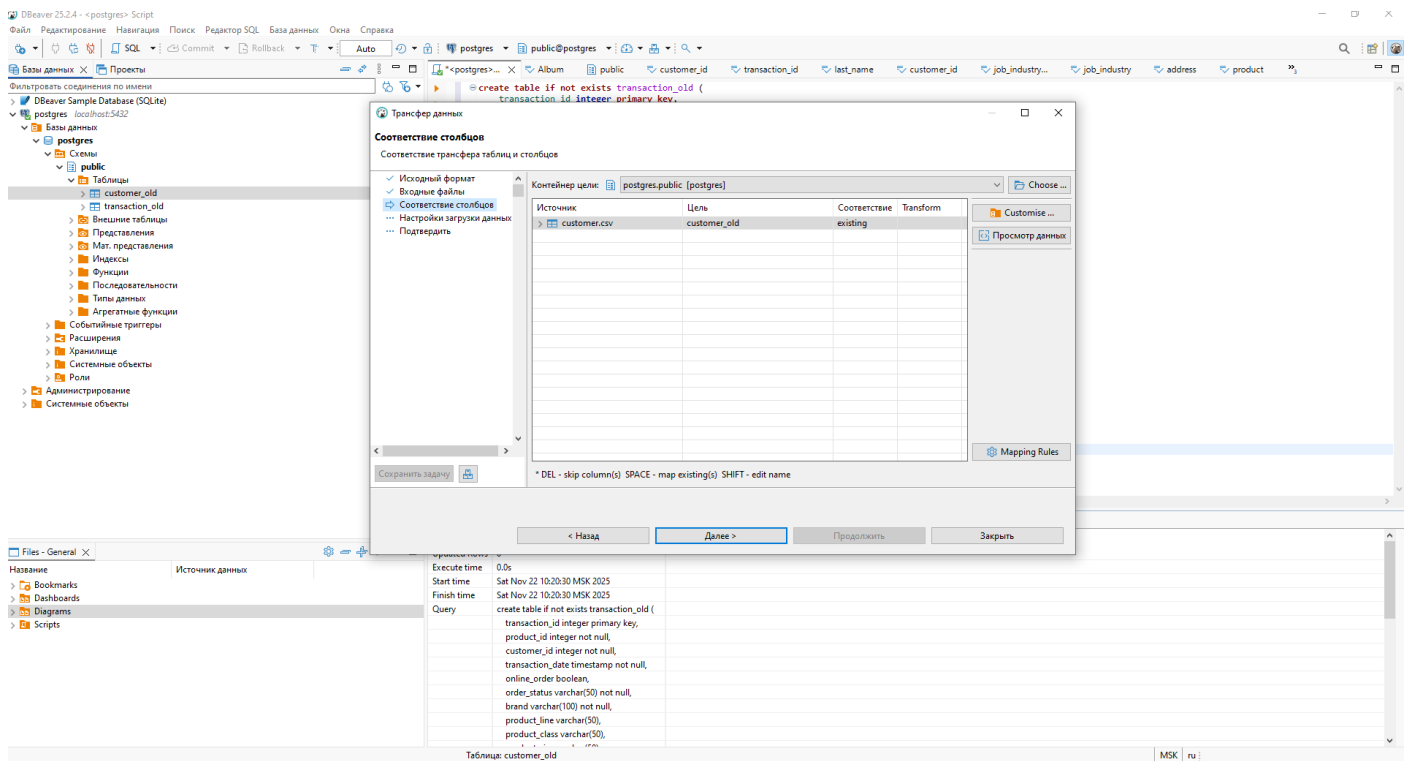


Рисунок 6 – Импорт данных в таблицу **customer_old**

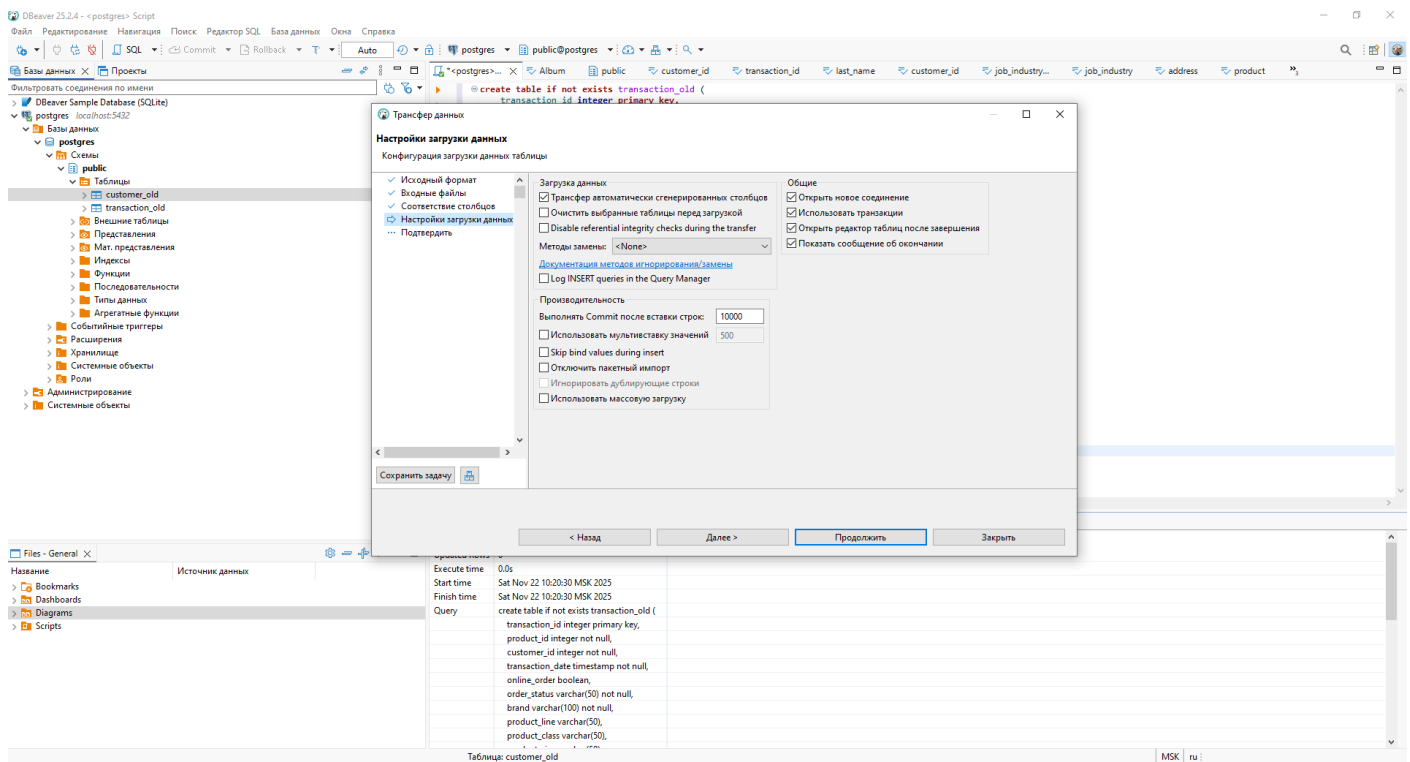


Рисунок 7 – Импорт данных в таблицу **customer_old**

ДБЕАВЕР 25.2.4 - customer_old

Файл Редактирование Навигация Поиск Редактор SQL База данных Окоя Справка

Базы данных X Проекты

Фильтровать соединения по имени

postgres localhost:5432

Базы данных

postgres

Схемы

public

Таблицы

customer_old 2,5M

transaction_old

Внешние таблицы

Представления

Мат. представления

Индексы

Функции

Последовательности

Типы данных

Агрегатные функции

Событийные триггеры

Расширения

Хранилище

Системные объекты

Роли

Администрирование

Системные объекты

Files - General X

Источники данных

Bookmarks

Dashboards

Diagrams

Scripts

Показать SQL

Введите SQL выражение чтобы отфильтровать результаты

customer_id	first_name	last_name	gender	dob	job_title	job_industry_category	wealth_segment	decase
1	Leraime	Medendorp	F	1953-10-12	Executive Secretary	Health	Mass Customer	N
2	Eli	Bockman	Male	1980-12-16	Administrative Officer	Financial Services	Mass Customer	N
3	Arlin	Dearle	Male	1984-01-20	Recruiting Manager	Property	Mass Customer	N
4	Talbot		Male	1961-10-20		IT	Mass Customer	N
5	Sheila-kathryn	Calton	Female	1977-05-13	Senior Editor	n/a	Affluent Customer	N
6	Curr	Duckhouse	Male	1966-09-16		Retail	High Net Worth	N
7	Fina	Merali	Female	1976-02-23		Financial Services	Affluent Customer	N
8	Rod	Inder	Male	1962-03-30	Media Manager I	n/a	Mass Customer	N
9	Mala	Lind	Female	1973-03-10	Business Systems Development Analyst	Agriculture	Affluent Customer	N
10	Florence	Birdall	Female	1988-10-11	Senior Quality Engineer	Financial Services	Mass Customer	N
11	Uriah	Bisett	Male	1954-04-30		Property	Mass Customer	N
12	Sawyer	Flattman	Male	1904-07-21	Nuclear Power Engineer	Manufacturing	Mass Customer	N
13	Gabriele	Norcross	Male	1955-02-15	Developer I	Financial Services	High Net Worth	N
14	Rayshell	Kitterman	Female	1983-03-25	Account Executive	Financial Services	Affluent Customer	N
15	Erroll	Radage	Male	2000-07-13	Junior Executive	Manufacturing	Mass Customer	N
16	Harlin	Parr	Male	1977-02-27	Media Manager IV	n/a	Mass Customer	N
17	Heath	Faraday	Male	1962-03-19	Sales Associate	n/a	Affluent Customer	N
18	Marje	Neasham	Female	1967-07-06	Professor	n/a	Affluent Customer	N
19	Keyson		Female	2001-08-15	Geological Engineer	Manufacturing	High Net Worth	N
20	Basile	Firth	Male	1980-08-13	Project Manager	Manufacturing	Mass Customer	N
21	Mile	Cammocke	Male	1980-09-20	Safety Technician I	Manufacturing	Affluent Customer	N
22	Deanne	Durtnell	Female	1962-12-10		IT	Mass Customer	N
23	Olav	Polak	Male	1995-02-10		n/a	High Net Worth	N
24	Kim	Skipsy	Female	1977-12-03	Research Assistant I	Agriculture	Mass Customer	N
25	Geoff	Assaf	Male	1976-12-02	Accounting Assistant III	Financial Services	Mass Customer	N
26	Tina	Ginnelly	Female	1978-09-10	Editor	Financial Services	Mass Customer	N
27	Garvin	Klies	Male	1978-09-25	Research Nurse	Health	Mass Customer	N
28	Fee	Zellmer	Male	1973-09-30	Senior Quality Engineer	Health	Affluent Customer	N
29	Mona	Sancraft	Female	1968-06-22	Safety Technician III	Manufacturing	Affluent Customer	N
30	Darrick	Helleckas	Male	1961-10-18		IT	Affluent Customer	N
31	Star	Praton	Female	1962-11-24	Staff Accountant III	Telecommunications	High Net Worth	N
32	Marion	Vanichkin	Female	1995-04-20	Legal Assistant	Manufacturing	Affluent Customer	N
33	Ernst		Male	1957-06-25	Product Engineer	n/a	Affluent Customer	N
34	Jephthah	Bachmann	U	1843-12-21	Legal Assistant	IT	Affluent Customer	N
35	Margaretha	Strettle	Female	1963-09-28	Information Systems Manager	Health	High Net Worth	N
36	Lurette	Stonnell	Female	1977-11-09	VP Quality Control	n/a	Affluent Customer	N
37	Laurie	Dveryhouse	Female	1985-12-22	Social Worker	Health	High Net Worth	N
38	Cordi	Merman	Female	1955-10-29	Senior Cost Accountant	Financial Services	Affluent Customer	N
39	Hunfredo	Smalley	Male	1979-04-16	Assistant Media Planner	Entertainment	Mass Customer	N
40	Tomasine	Jerche	Female	1981-10-27	Payment Adjustment Coordinator	Manufacturing	Affluent Customer	N
41	Radluc	Czarna	Male	1976-04-14	Fond Chemist	Health	Mass Customer	N

Обновить Save Cancel Экспорт данных... 200 200+ 200 строк получено - 0.0s (0.0s получ.) 2025-11-22 в 10:24:49

postgres postgres public customer old

Рисунок 8 – Просмотр данных таблицы **customer_old**

ДБЕАВЕР 25.2.4 - transaction_old

Файл Редактирование Навигация Поиск Редактор SQL База данных Окоя Справка

Базы данных X Проекты

Фильтровать соединения по имени

postgres localhost:5432

Базы данных

postgres

Схемы

public

Таблицы

customer_old

transaction_old

Внешние таблицы

Представления

Мат. представления

Индексы

Функции

Последовательности

Типы данных

Агрегатные функции

Событийные триггеры

Расширения

Хранилище

Системные объекты

Роли

Администрирование

Системные объекты

Files - General X

Источники данных

Bookmarks

Dashboards

Diagrams

Scripts

Трансфер данных

Конфигурация входных файлов и папок

Исходный формат

Входные файлы

Соответствие столбцов

Настройки загрузки данных

Подтвердить

Входные файлы: Browse

Исходник: D:\sql\Импортируемые начальные csv и xls\transaction.csv

Цель: public.transaction_old [Existing]

Настройки импорта:

Имя	Значение
Расширение	csv,txt,txt
Кодировка	utf-8
Разделитель столбцов	;
Положение заголовка	top
Символ кавычки	"
Ignore characters outside quotes	[]
Символ, используемый для выхода	\
Значение NULL	[]
Установить пустые строки в NULL	[]

More information about individual settings

Сохранить задачу

Назад Далее Продолжить Закрыть

Значение X

Select a cell to view/edit value

Press F7 to hide this panel

Обновить Save Cancel Экспорт данных... 200 200+ Нет данных - 0.0s, 2025-11-22 в 10:25:37

postgres postgres public transaction old

Таблица: transaction_old

Рисунок 9 – Импорт данных в таблицу **transaction_old**

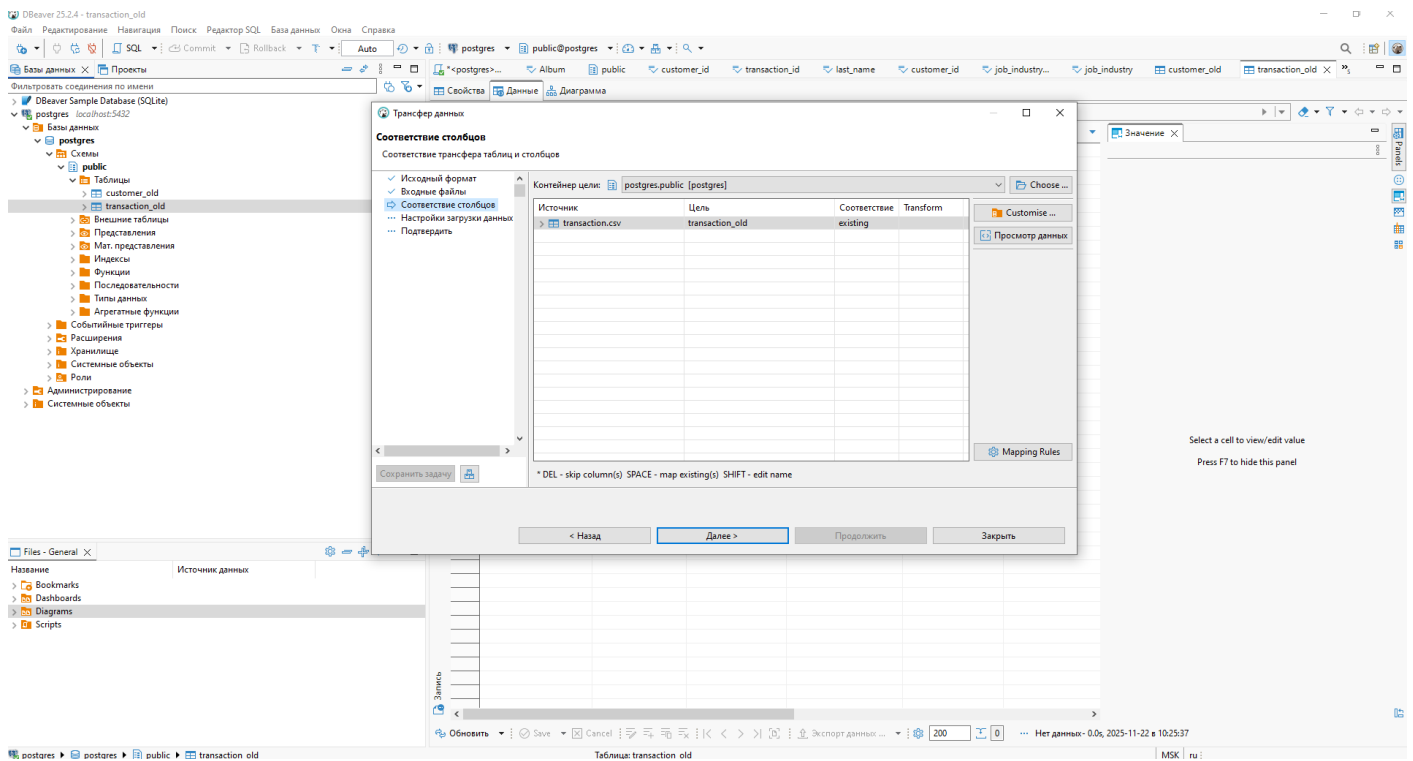


Рисунок 10 – Импорт данных в таблицу **transaction_old**

transaction_id	product_id	customer_id	transaction_date	online_order	AZ order_status	AZ brand	AZ product_line	AZ product_class
1	2	2	2017-02-25	[]	Approved	Soles	Standard	medium
2	2	3	2017-05-21	[v]	Approved	Trek Bicycles	Standard	medium
3	3	37	2017-10-16	[]	Approved	OHM Cycles	Standard	low
4	4	88	2017-08-31	[]	Approved	Norco Bicycles	Standard	medium
5	5	78	2017-04-21	[v]	Approved	Giant Bicycles	Standard	medium
6	6	25	2017-03-08	[v]	Approved	Giant Bicycles	Road	medium
7	7	22	2017-04-21	[v]	Approved	WearA2B	Standard	medium
8	8	15	2017-07-15	[]	Approved	WearA2B	Standard	medium
9	9	67	2017-08-10	[]	Approved	Soles	Standard	medium
10	10	12	2017-08-30	[v]	Approved	WearA2B	Standard	medium
11	11	5	2017-01-17	[]	Approved	Trek Bicycles	Mountain	low
12	12	61	2017-01-05	[v]	Approved	OHM Cycles	Standard	low
13	13	35	2017-02-26	[v]	Approved	Trek Bicycles	Standard	low
14	14	16	2017-09-10	[]	Approved	Norco Bicycles	Standard	high
15	15	12	2017-06-11	[]	Approved	Giant Bicycles	Standard	medium
16	16	3	2017-10-10	[]	Approved	Trek Bicycles	Standard	medium
17	17	79	2017-04-03	[]	Approved	Norco Bicycles	Standard	medium
18	18	33	2017-06-02	[]	Approved	Giant Bicycles	Standard	medium
19	19	54	2017-04-06	[v]	Approved	WearA2B	Standard	medium
20	20	25	2017-01-28	[v]	Approved	Giant Bicycles	Road	medium
21	21	27	2017-10-09	[]	Approved	Trek Bicycles	Standard	medium
22	22	37	2017-06-29	[v]	Approved	OHM Cycles	Standard	low
23	23	37	2017-04-08	[]	Approved	OHM Cycles	Standard	low
24	24	82	2017-10-18	[]	Approved	Giant Bicycles	Road	medium
25	25	89	2017-06-11	[]	Approved	WearA2B	Touring	medium
26	26	64	2017-01-10	[]	Approved	Trek Bicycles	Standard	medium
27	27	64	2017-04-11	[v]	Approved	Trek Bicycles	Standard	medium
28	28	19	2017-12-23	[]	Approved	Trek Bicycles	Mountain	low
29	29	72	2017-10-13	[v]	Approved	Norco Bicycles	Standard	medium
30	30	91	2017-03-15	[]	Approved	WearA2B	Standard	low
31	31	88	2017-08-05	[v]	Approved	Norco Bicycles	Standard	medium
32	32	1	2017-02-18	[]	Approved	Giant Bicycles	Standard	medium
33	33	25	2017-02-20	[]	Approved	Giant Bicycles	Road	medium
34	34	99	2017-02-28	[v]	Approved	Trek Bicycles	Road	low
35	35	0	2017-08-20	[]	Approved	Norco Bicycles	Road	medium
36	36	92	2017-07-07	[]	Approved	WearA2B	Standard	medium
37	37	14	2017-01-09	[]	Approved	Soles	Standard	high
38	38	2	2017-12-06	[]	Approved	Soles	Standard	medium
39	39	12	2017-09-12	[]	Approved	WearA2B	Standard	medium
40	40	0	2017-11-28	[v]	Approved	Norco Bicycles	Road	medium
41	41	44	2017-05-08	[v]	Announc	WearA2B	Standard	medium

Рисунок 11 – Просмотр данных таблицы **transaction_old**

Создаем таблицы: **job_industry**, **wealth_segment**, **address**, **product**, **customer**, **transaction** с колонками и типами данных, соответствующими ЗНФ.

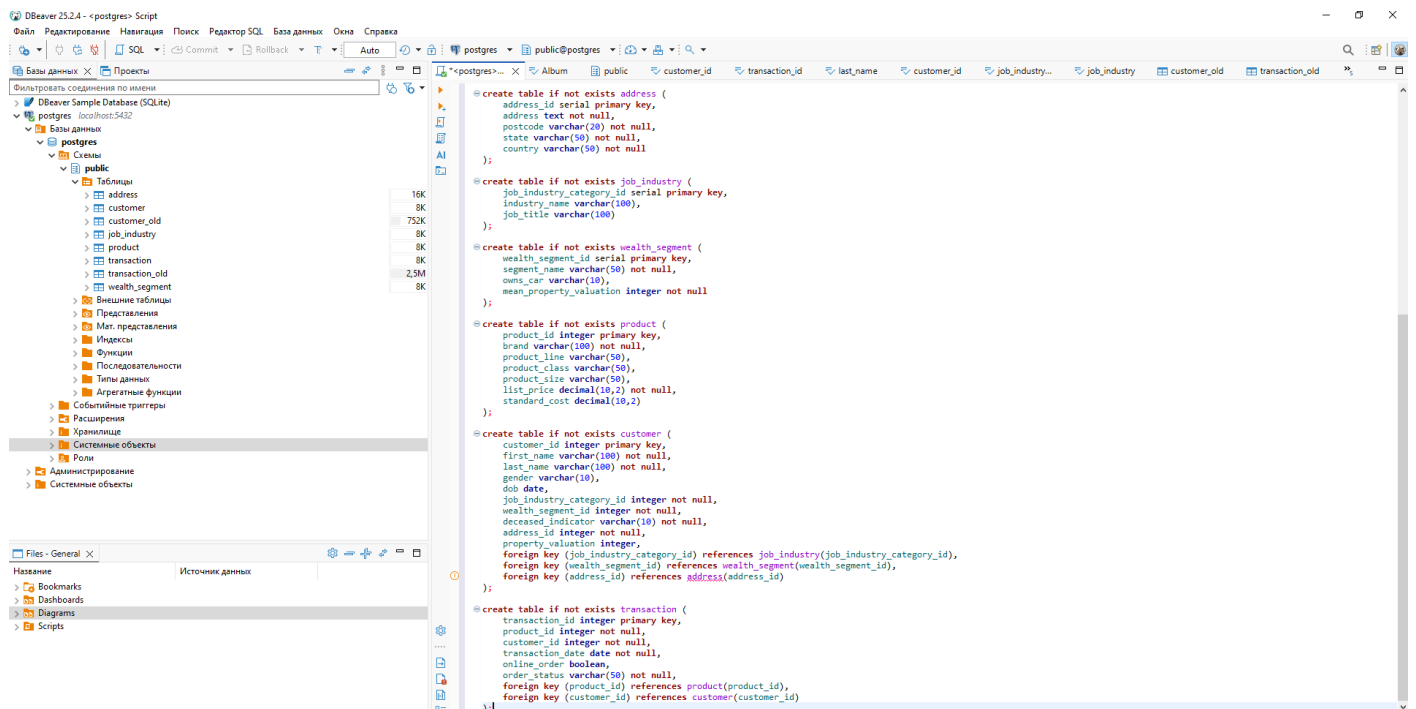


Рисунок 12 – Создание таблиц **address**, **job_industry**, **wealth_segment**, **product**, **customer**, **transaction**

Заполняем таблицу **address**, используя данные из **customer_old**.

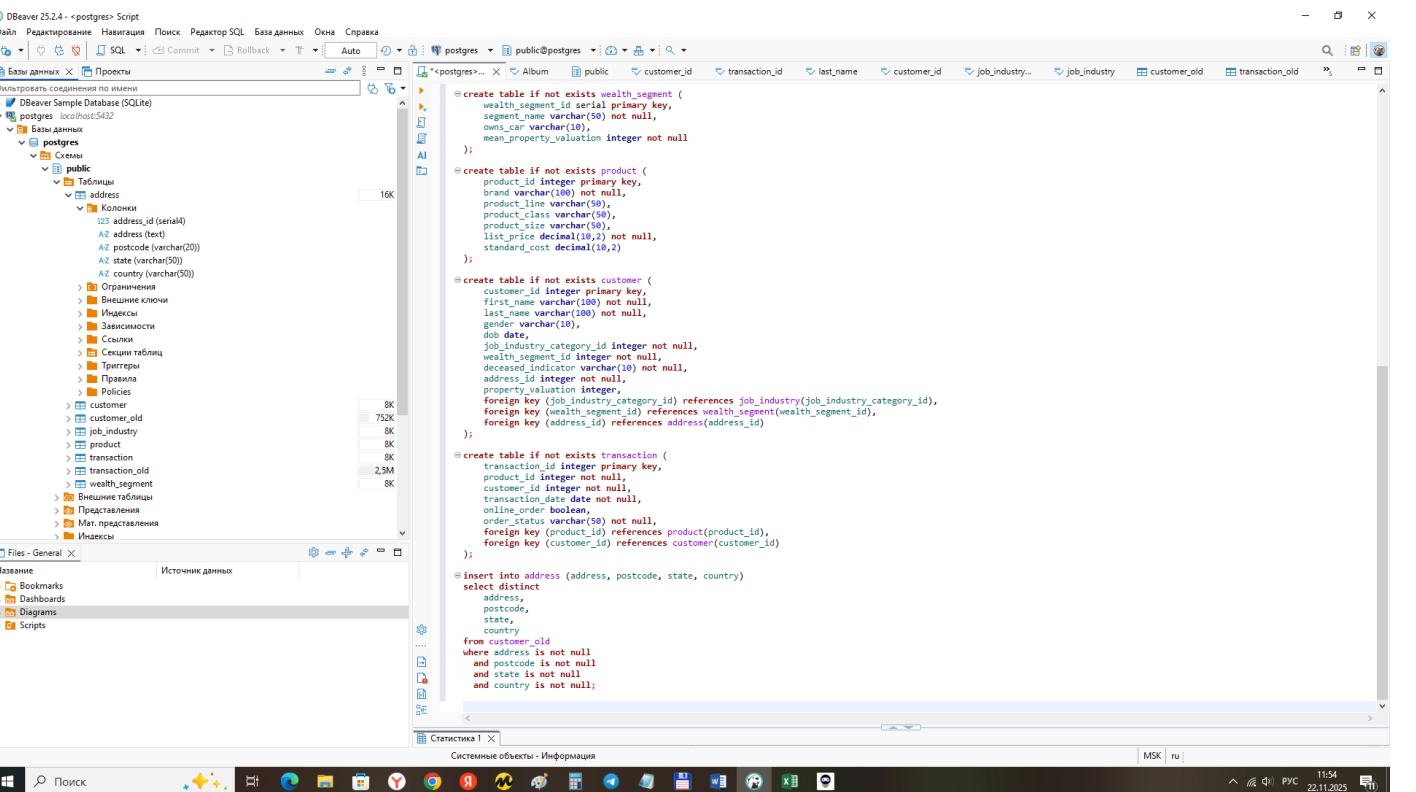


Рисунок 13 – Заполнение таблицы **address**

address_id	address	postcode	state	country
1	709 Rieder Terrace	3995	VIC	Australia
2	8462 Emmet Trail	4878	QLD	Australia
3	24198 Almo Crossing	2021	NSW	Australia
4	2388 Thackeray Place	3130	VIC	Australia
5	7 Summerdown Parkway	3073	VIC	Australia
6	31186 Hoard Junction	2015	NSW	Australia
7	2 Green Ridge Park	2478	NSW	Australia
8	1 Dayton Park	2767	NSW	Australia
9	434 Eggendart Circle	2260	NSW	Australia
10	5 Hauk Lane	2650	NSW	Australia
11	624 Harper Way	2094	NSW	Australia
12	3 Gale Street	4507	QLD	Australia
13	11 Dunnington Pass	2219	NSW	Australia
14	9317 Mendota Parkway	3064	VIC	Australia
15	81856 Express Lane	2299	NSW	Australia
16	13 Colorado Lane	2099	NSW	Australia
17	54295 Dorton Hill	2219	NSW	Australia
18	707 Spaight Place	2830	NSW	Australia
19	5015 Pawling Park	2525	NSW	Australia
20	2390 Blackbird Point	2035	NSW	Australia
21	37 Darwin Circle	2282	NSW	Australia
22	5 Reindahl Point	3400	VIC	Australia
23	29 South Point	3111	Victoria	Australia
24	53 Dakota Court	2027	NSW	Australia
25	6674 Russell Center	2197	NSW	Australia
26	1 Parkside Avenue	3109	VIC	Australia
27	22 Dells Point	2233	NSW	Australia
28	2 Monterey Terrace	2120	NSW	Australia
29	7 Heavey Point	4005	QLD	Australia
30	869 Forster Circle	2112	New South Wales	Australia
31	3 Bashford Plaza	2267	NSW	Australia
32	098 Veith Hill	4510	QLD	Australia
33	77729 Kim Plaza	2560	NSW	Australia
34	6 Golf Center	2042	NSW	Australia
35	1807 Falkview Plaza	4209	QLD	Australia
36	7049 Sutteridge Lane	2251	NSW	Australia
37	69278 High Crossing Place	2795	NSW	Australia
38	75717 Bartillon Road	2880	NSW	Australia
39	4368 Dayton Street	2753	NSW	Australia
40	486 Mariners Cove Plaza	3046	VIC	Australia
41	4386 Spaight Court	3752	VIC	Australia
42	8 Cambridge Drive	7193	NSW	Australia

Рисунок 14 – Просмотр данных таблицы **address**

Заполняем таблицу **job_industry**, используя данные из **customer_old**.

```

-- create table if not exists product (
--   product_id integer primary key,
--   brand varchar(100) not null,
--   product_line varchar(50),
--   product_class varchar(50),
--   product_size varchar(50),
--   list_price decimal(10,2) not null,
--   standard_cost decimal(10,2)
-- );

-- create table if not exists customer (
--   customer_id integer primary key,
--   first_name varchar(100) not null,
--   last_name varchar(100) not null,
--   gender varchar(10),
--   dob date,
--   job_industry_category_id integer not null,
--   wealth_segment_id integer not null,
--   deceased_indicator varchar(10) not null,
--   address_id integer not null,
--   property_valuation integer,
--   foreign key (job_industry_category_id) references job_industry(job_industry_category_id),
--   foreign key (wealth_segment_id) references wealth_segment(wealth_segment_id),
--   foreign key (address_id) references address(address_id)
-- );

-- create table if not exists transaction (
--   transaction_id integer primary key,
--   product_id integer not null,
--   customer_id integer not null,
--   transaction_date date not null,
--   online_order boolean,
--   order_status varchar(50) not null,
--   foreign key (product_id) references product(product_id),
--   foreign key (customer_id) references customer(customer_id)
-- );

-- insert into address (address, postcode, state, country)
-- select distinct
--   address,
--   postcode,
--   state,
--   country
-- from customer_old
-- where address is not null
-- and postcode is not null
-- and state is not null
-- and country is not null;

-- insert into job_industry (industry_name, job_title)
-- select distinct
--   job_industry_category,
--   job_title
-- from customer_old
-- where job_industry_category is not null;

```

Рисунок 15 – Заполнение таблицы **job_industry**

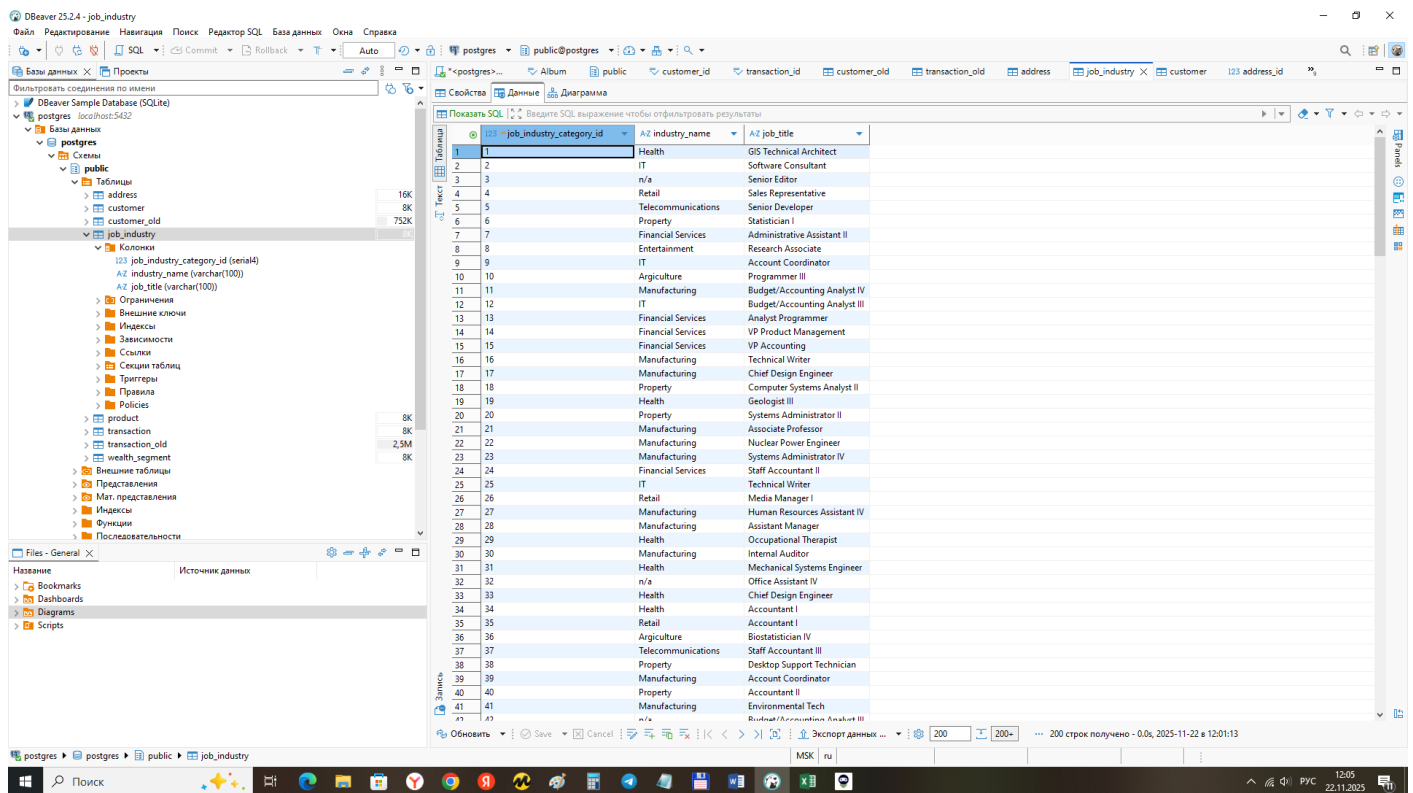


Рисунок 16 – Просмотр данных таблицы **job_industry**

Заполняем таблицу **wealth_segment**, используя данные из **customer_old**.

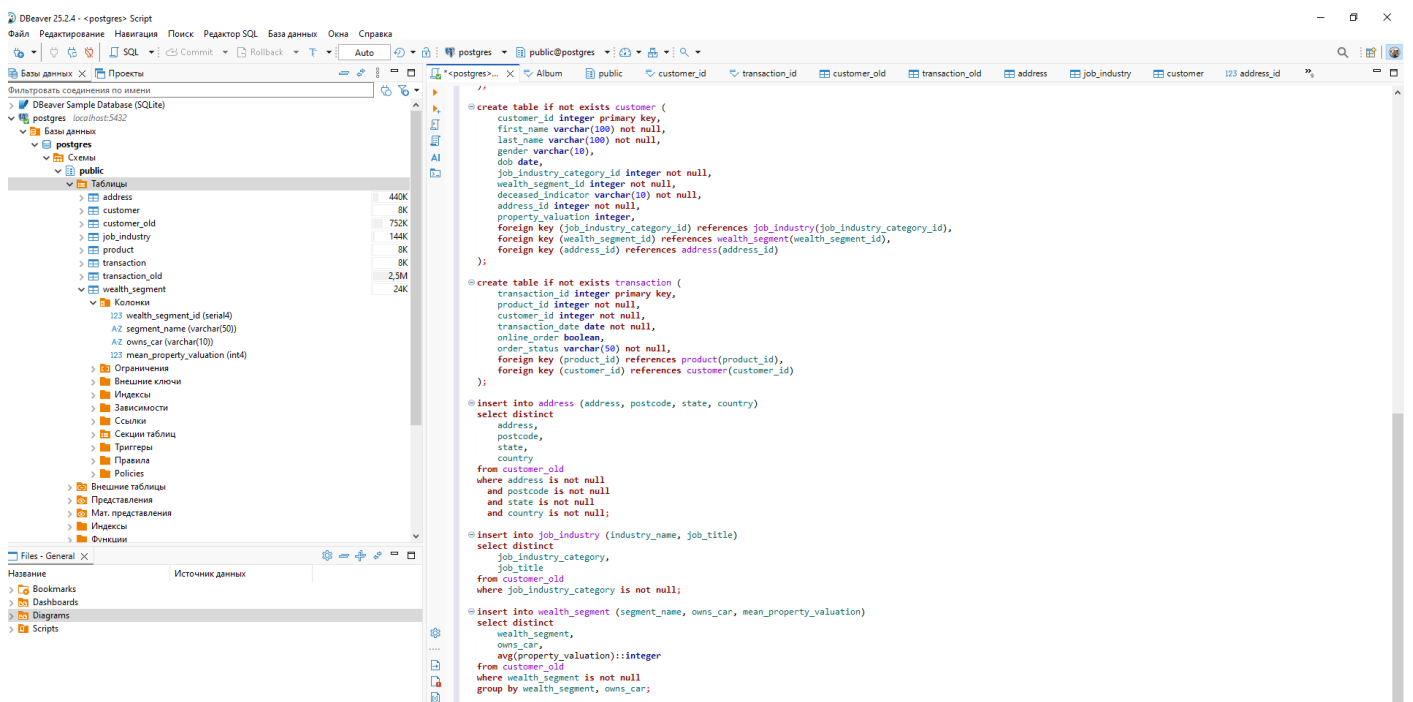


Рисунок 17 – Заполнение таблицы **wealth_segment**

The screenshot shows the DBeaver 25.2.4 interface. On the left, the 'public' schema is expanded, showing the 'wealth_segment' table. The table has columns: wealth_segment_id (serial4), segment_name (varchar(50)), owns_car (varchar(10)), and mean_property_valuation (int4). The table size is 2.5M. The main pane displays the table data with 6 rows.

wealth_segment_id	segment_name	owns_car	mean_property_valuation
1	Affluent Customer	No	7
2	Affluent Customer	Yes	8
3	High Net Worth	No	7
4	High Net Worth	Yes	8
5	Mass Customer	No	7
6	Mass Customer	Yes	8

Рисунок 18 – Просмотр данных таблицы **wealth_segment**

Заполняем таблицу **product**, используя данные из **transaction_old**.

При попытке заполнения оказалось, что есть дубликаты product_id. Выполнена проверка га дубликаты.

The screenshot shows the DBeaver 25.2.4 interface with a SQL script editor. The script contains SQL commands to create the 'product' table, insert data from 'transaction_old', and check for duplicates.

```

-- Create table if not exists transaction (
transaction_id integer primary key,
product_id integer not null,
customer_id integer not null,
transaction_date date not null,
online_order boolean,
order_status varchar(50) not null,
foreign key (product_id) references product(product_id),
foreign key (customer_id) references customer(customer_id)
);

-- Create table if not exists transaction (
transaction_id integer primary key,
product_id integer not null,
customer_id integer not null,
transaction_date date not null,
online_order boolean,
order_status varchar(50) not null,
foreign key (product_id) references product(product_id),
foreign key (customer_id) references customer(customer_id)
);

-- Insert into address (address, postcode, state, country)
select distinct
address,
postcode,
state,
country
from customer_old
where address is not null
and postcode is not null
and state is not null
and country is not null;

-- Insert into job_industry (industry_name, job_title)
select distinct
job_industry_category,
job_title
from customer_old
where job_industry_category is not null;

-- Insert into wealth_segment (segment_name, owns_car, mean_property_valuation)
select distinct
wealth_segment,
owns_car,
avg(mean_property_valuation)::integer
from customer_old
where wealth_segment is not null
group by wealth_segment, owns_car;

-- Select product_id, count(*)
from transaction_old
where product_id is not null
group by product_id
having count(*) > 1;

```

Рисунок 19 – Проверка на дубликаты product_id

product_id	count
1	5
2	18
3	64
4	55
5	27
6	23
7	56
8	58
9	91
10	8
11	87
12	74
13	29
14	54
15	71
16	68
17	4
18	34
19	51
20	0
21	96
22	52
23	70
24	80
25	83
26	67
27	63
28	90
29	10
30	35
31	45
32	6
33	84
34	86
35	39
36	92
37	93
38	89
39	69
40	36
41	31
42	50
43	204

Рисунок 20 – Просмотр дубликатов product_id

Тогда необходимо изменить таблицы **product**, **transaction** и добават новую таблицу связей “многие ко многим” **transaction_product**.

```

drop table if exists product cascade

create table if not exists product (
    product_version_id serial primary key,
    product_id integer not null,
    brand varchar(100) not null,
    product_line varchar(50),
    product_class varchar(50),
    product_size varchar(50),
    list_price decimal(10,2) not null,
    standard_cost decimal(10,2)
);

drop table if exists transaction cascade

create table if not exists transaction (
    transaction_id integer primary key,
    customer_id integer not null,
    transaction_date date not null,
    online_order boolean,
    order_status varchar(50) not null,
    foreign key (customer_id) references customer(customer_id)
);

create table if not exists transaction_product (
    transaction_product_id serial primary key,
    transaction_id integer not null,
    product_version_id integer not null,
    foreign key (transaction_id) references transaction(transaction_id),
    foreign key (product_version_id) references product(product_version_id)
);

```

Рисунок 21 – Создание таблицы **transaction_product** и удаление и повторное создание таблицы **transaction**

Заполняем таблицу **product**, используя данные из **transaction_old**.

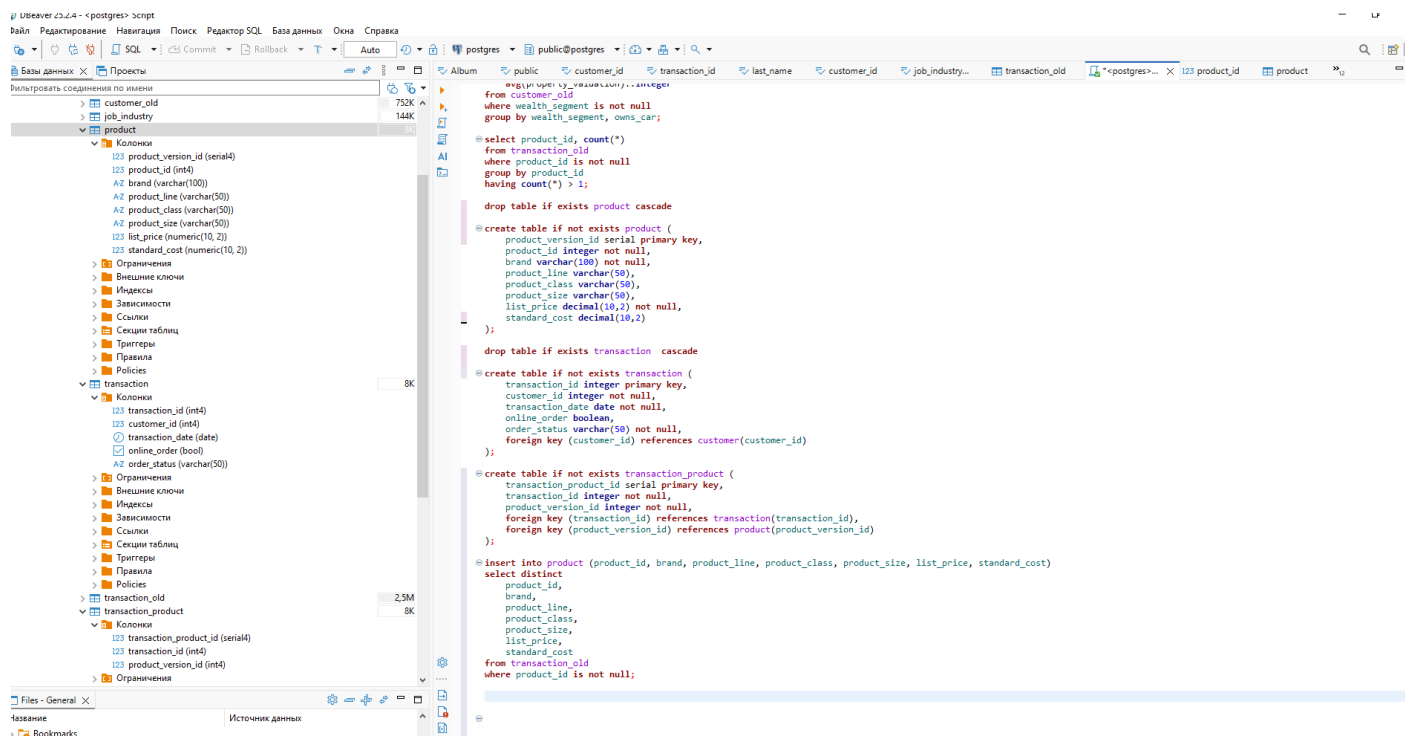


Рисунок 22 – Заполнение таблицы **product**

product_version_id	product_id	brand	product_line	product_class	product_size	list_price	standard_cost
1	1					1 678,51	[NULL]
2	2	81	Solex	Standard	medium	1 151,96	649,49
3	3	45	Trek Bicycles	Road	low	980,37	234,43
4	4	58	OHM Cycles	Standard	medium	912,52	141,4
5	5	0				2 061,38	[NULL]
6	6	0				330,36	[NULL]
7	7	0				890,28	[NULL]
8	8	73	Solex	Standard	medium	1 945,43	323,19
9	9	0				1 562,88	[NULL]
10	10	0				1 202,34	[NULL]
11	11	41	Norco Bicycles	Standard	low	958,74	748,9
12	12	74	WearA2B	Standard	medium	1 228,07	400,91
13	13	0	OHM Cycles	Road	medium	742,54	667,4
14	14	5	Giant Bicycles	Standard	high	1 129,13	677,48
15	15	0				1 999,34	[NULL]
16	16	41	Solex	Road	medium	416,98	312,74
17	17	0				294,35	[NULL]
18	18	0				149,3	[NULL]
19	19	0				205,94	[NULL]
20	20	0				483,12	[NULL]
21	21	66	Solex	Standard	medium	1 163,89	589,27
22	22	79	Solex	Touring	medium	2 083,94	675,03
23	23	0				2 028,26	[NULL]
24	24	0				1 438,9	[NULL]
25	25	53	OHM Cycles	Standard	medium	795,34	101,58
26	26	0				1 776,08	[NULL]
27	27	37	OHM Cycles	Standard	low	1 793,43	248,82
28	28	0				1 224,41	[NULL]
29	29	0				162,87	[NULL]
30	30	0				226,94	[NULL]
31	31	53	Giant Bicycles	Standard	high	1 274,93	764,96
32	32	0				753,76	[NULL]
33	33	0	Solex	Standard	medium	478,16	298,72
34	34	34	WearA2B	Standard	medium	1 231,15	161,6
35	35	81	Norco Bicycles	Standard	medium	586,45	521,94
36	36	61	Norco Bicycles	Standard	medium	586,45	521,94
37	37	0				1 535,84	[NULL]
38	38	0				1 281,6	[NULL]
39	39	0				1 473,09	[NULL]
40	40	0				813,6	[NULL]
41	41	93	OHM Cycles	Standard	high	1 458,17	874,9
42	42	0				1 788,02	[NULL]

Рисунок 23 – Просмотр данных таблицы **product**

Заполняем таблицу **customer**, используя данные из **customer_old**.

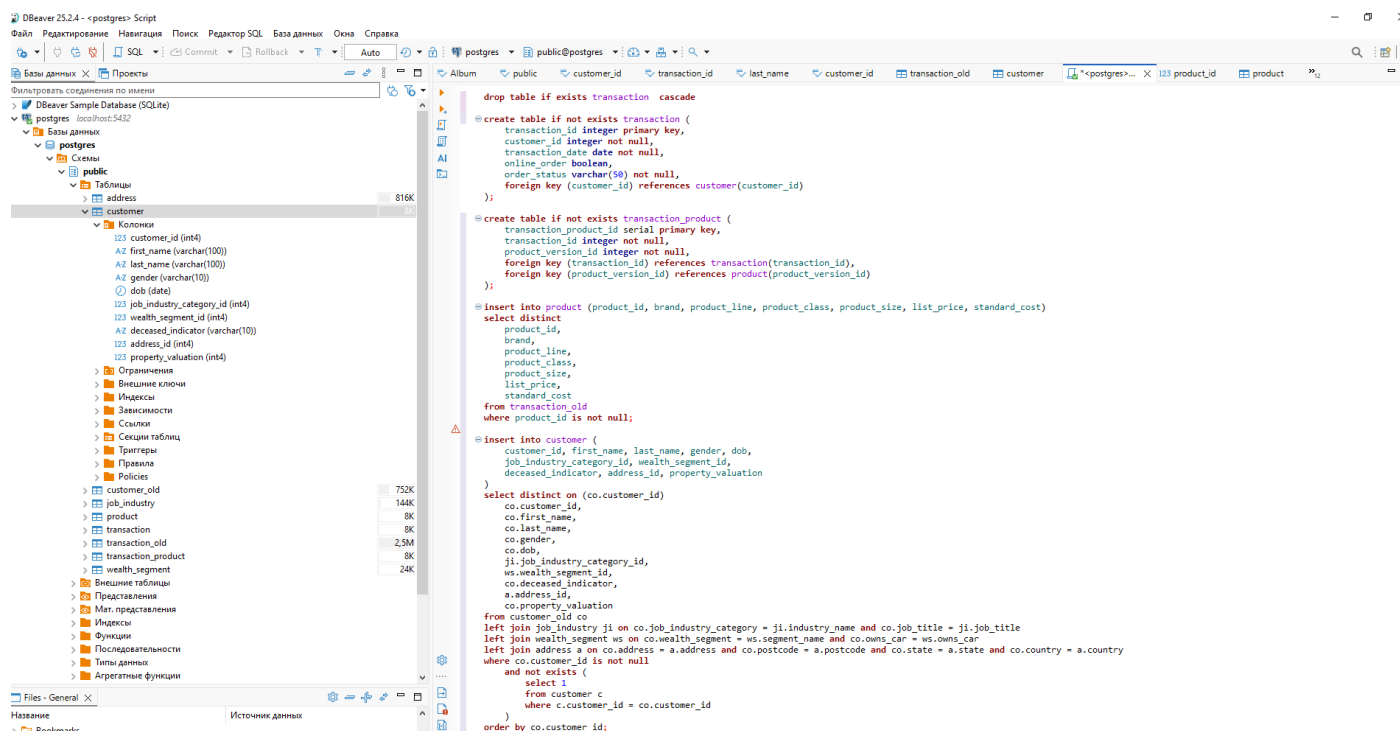


Рисунок 24 – Заполнение таблицы **customer**

The screenshot shows the DBeaver 25.2.4 interface with the 'customer' table data displayed. The table has the following columns: customer_id, first_name, last_name, gender, dob, job_industry_category_id, wealth_segment_id, deceased_indicator, and address_id. The data is as follows:

customer_id	first_name	last_name	gender	dob	job_industry_category_id	wealth_segment_id	deceased_indicator	address_id
1	Laraine	Medendorp	F	1953-10-12	648	6	N	
2	Ell	Bockman	Male	1900-12-16	270	6	N	
3	Arlin	Dearle	Male	1954-01-20	892	6	N	
4	Talbot		Male	1961-10-03	397	5	N	
5	Sheila-kathryn	Calton	Female	1977-05-13	3	2	N	
6	Curr	Duckhouse	Male	1966-09-16	585	4	N	
7	Fina	Merali	Female	1976-02-23	908	2	N	
8	Rod	Inder	Male	1962-03-30	324	5	N	
9	Mala	Lind	Female	1973-03-10	142	2	N	
10	Finanze	Bridall	Female	1988-10-11	222	6	N	
11	Ulrich	Bisatt	Male	1954-04-30	1043	5	N	
12	Sawyer	Flattman	Male	1904-07-21	22	5	N	
13	Gabriele	Norcross	Male	1955-02-15	1018	4	N	
14	Rayshell	Kitterman	Female	1983-03-25	982	1	N	
15	Erroll	Radage	Male	2000-07-13	1032	5	N	
16	Harlin	Parr	Male	1977-02-27	274	6	N	
17	Heath	Faraday	Male	1962-03-19	628	2	N	
18	Marje	Nasham	Female	1967-07-06	96	1	N	
19	Sorcha	Keypson	Female	2001-04-15	942	3	N	
20	Basile	Firth	Male	1980-08-13	901	5	N	
21	Mike	Cammocke	Male	1980-09-20	301	2	N	
22	Deanne	Durtnell	Female	1962-12-10	397	5	N	
23	Olav	Polak	Male	1995-02-10	568	4	N	
24	Kim	Skpsey	Female	1977-12-03	421	6	N	
25	Geoff	Assaf	Male	1976-12-02	953	6	N	
26	Toxi	Ginnelly	Female	1978-06-10	698	6	N	
27	Gavin	Kloes	Male	1978-09-25	474	6	N	
28	Fee	Zellmer	Male	1973-09-30	179	2	N	
29	Mona	Sancraft	Female	1968-06-22	196	5	N	
30	Derrick	Helleskas	Male	1961-10-18	397	2	N	
31	Star	Praton	Female	1962-11-24	37	4	N	
32	Maion	Vanichkin	Female	1995-04-20	774	1	N	
33	Ernst	Hacon	Male	1957-06-25	262	2	N	
34	Jephthah	Bachmann	U	1843-12-21	759	1	N	
35	Margaretha	Strettle	Female	1963-09-28	314	4	N	
36	Lurette	Stonell	Female	1977-11-09	432	1	N	
37	Laurie	Diversityhouse	Female	1985-12-22	652	3	N	
38	Cordi	Merman	Female	1955-10-29	193	1	N	
39	Hunfredo	Smalley	Male	1979-04-16	848	5	N	
40	Tomatine	Jerche	Female	1981-10-27	696	1	N	
41	Racilue	Cnipe	Male	1976-04-14	604	5	N	

Рисунок 25 – Просмотр данных таблицы **customer**

Заполняем таблицу **transaction**, используя данные из **transaction_old**.

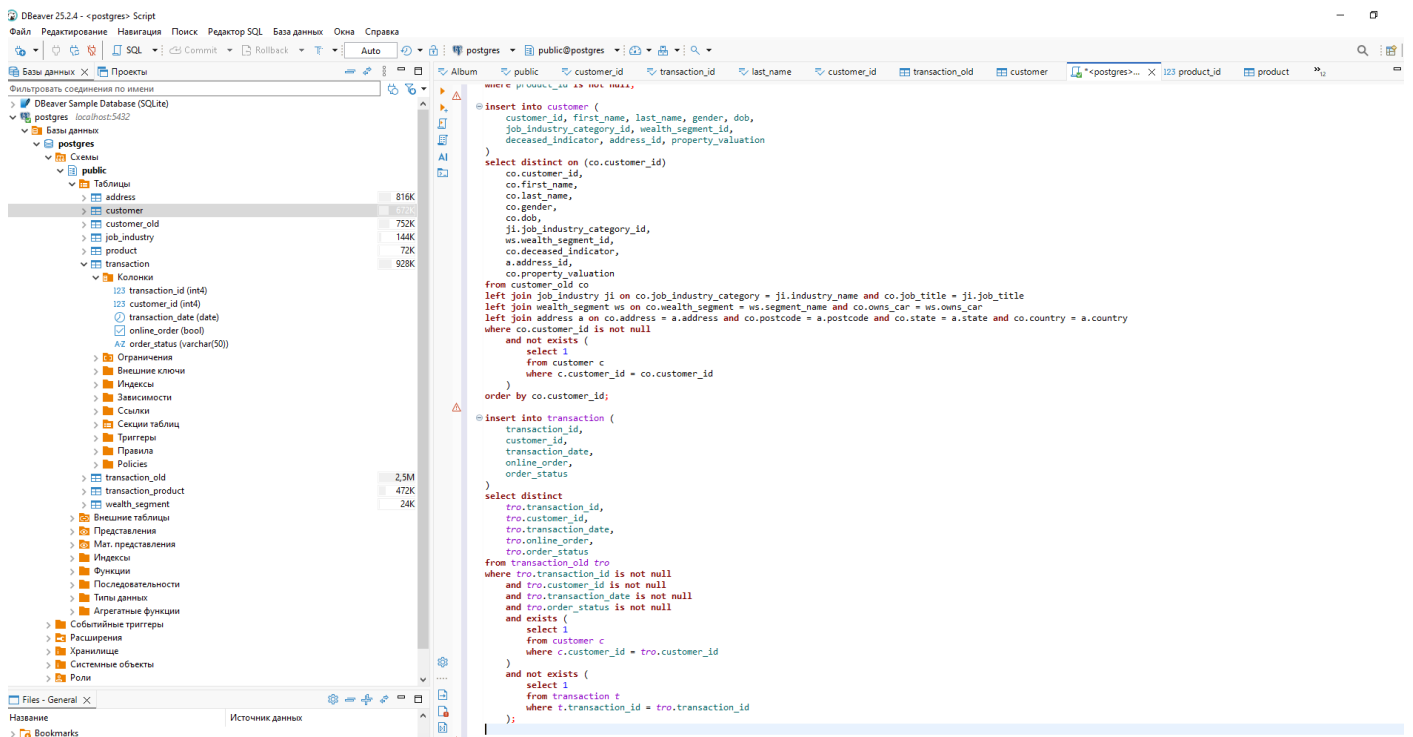


Рисунок 26 – Заполнение таблицы **transaction**

transaction_id	customer_id	transaction_date	online_order	A/Z order_status
1	12 142	2 326		Approved
2	15 186	1 287	[v]	Approved
3	19 489	927	[v]	Approved
4	12 104	323	[]	Approved
5	13 008	2 976		Approved
6	12 425	1 611	[]	Approved
7	6 751	2 533	[]	Approved
8	4 940	2 502	[]	Approved
9	3 530	2 537	[v]	Approved
10	19 228	3 307	[]	Approved
11	9 449	350	[]	Approved
12	3 413	197	[]	Approved
13	13 336	2 833	[v]	Approved
14	18 334	917	[v]	Approved
15	11 296	37	[]	Approved
16	1 705	712	[v]	Approved
17	18 067	3 107	[v]	Approved
18	12 073	1 207	[]	Approved
19	19 693	1 532	[]	Approved
20	13 298	2 182	[v]	Approved
21	2 027	2 106	[]	Approved
22	466	505	[v]	Approved
23	19 278	159	[v]	Approved
24	8 663	3 201	[]	Approved
25	5 346	2 017	[]	Approved
26	6 272	1 707	[]	Approved
27	1 534	3 004	[v]	Approved
28	331	217	[]	Approved
29	18 959	860	[]	Approved
30	8 320	2 439	[]	Approved
31	19 729	3 177	[]	Approved
32	1 301	1 893	[v]	Approved
33	19 119	776	[v]	Approved
34	12 864	1 988	[v]	Approved
35	3 486	2 012	[v]	Approved
36	1 286	2 856	[v]	Approved
37	16 042	1 689	[v]	Approved
38	17 304	3 322	[]	Approved
39	865	1 211	[]	Approved
40	8 896	1 736	[v]	Approved
41	10 253	441	[]	Approved
A 001	3 307	2017-10-31	[]	Approved

Рисунок 27 – Просмотр данных таблицы **transaction**

Заполняем таблицу **transaction_product**, используя данные из **transaction_old**.

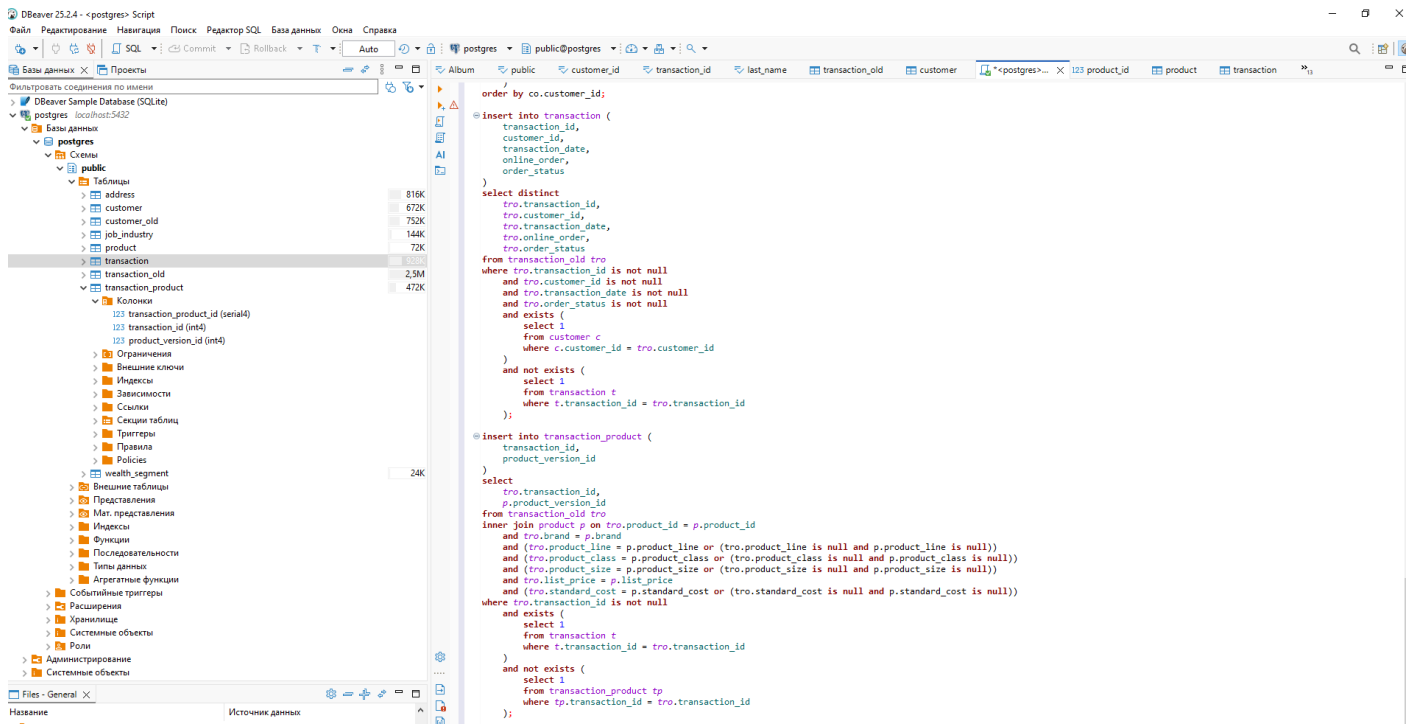


Рисунок 26 – Заполнение таблицы **transaction_product**

transaction_id	product_version_id	transaction_date
40 001	1	120
40 002	2	351
40 003	3	27
40 004	4	341
40 005	5	168
40 006	6	93
40 007	7	190
40 008	8	186
40 009	9	295
40 010	10	72
40 011	11	395
40 012	12	262
40 013	13	176
40 014	14	187
40 015	15	88
40 016	16	351
40 017	17	265
40 018	18	300
40 019	19	161
40 020	20	93
40 021	21	46
40 022	22	27
40 023	23	27
40 024	24	337
40 025	25	238
40 026	26	111
40 027	27	111
40 028	28	75
40 029	29	338
40 030	30	157
40 031	31	341
40 032	32	144
40 033	33	93
40 034	34	282
40 035	35	293
40 036	36	87
40 037	37	367
40 038	38	120
40 039	39	72
40 040	40	293
40 041	41	210

Рисунок 27 – Просмотр данных таблицы **transaction_product**

