Ртищева Алена Сергеевна
Группа М08-501НД
Науки о данных

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.

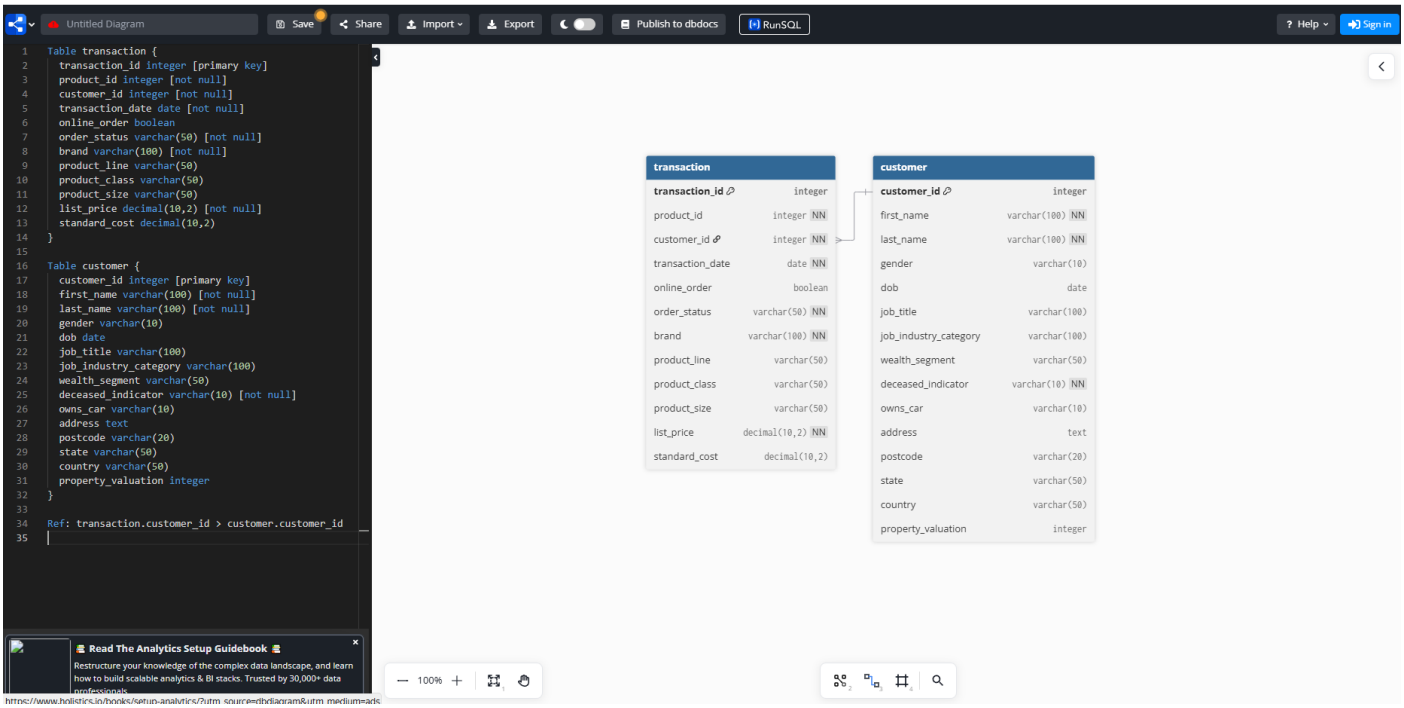


Рисунок 1 – 1НФ (исходный вариант)

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

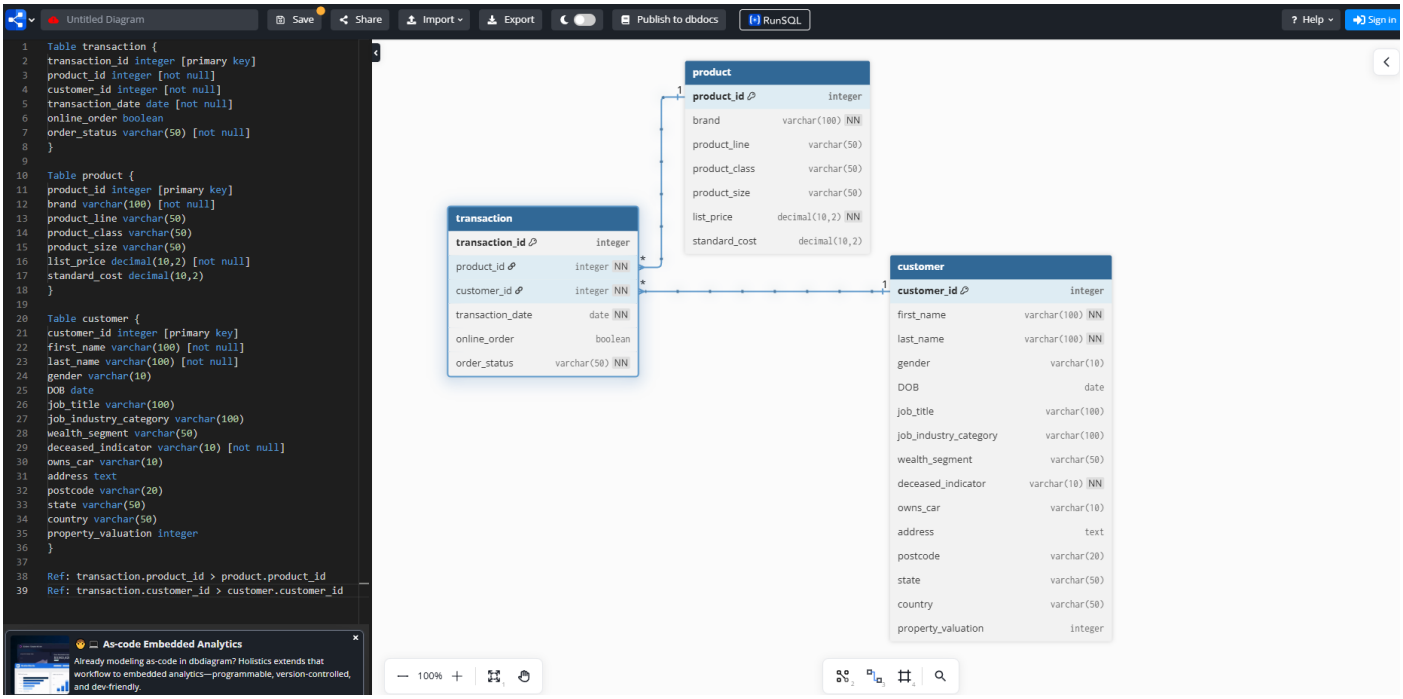


Рисунок 2 – 2НФ

<u>Эта база данных имеет 2НФ (вторую нормальную форму) потому что:</u>

1. Соотвествует всем требованиям 1НФ (все атрибуты атомарны, нет повторяющихся групп, есть первичный ключ)
2. Отсутствуют частичные зависимости:
   product_id → (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]

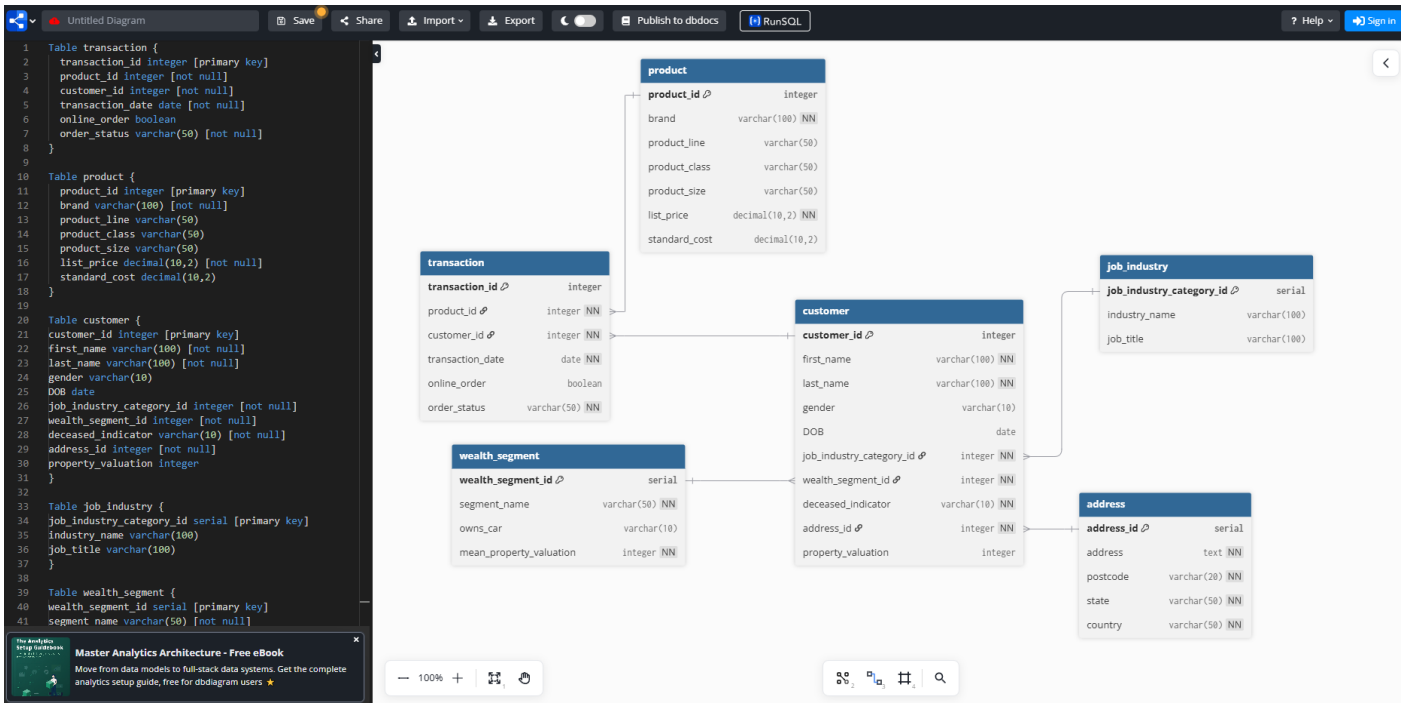
Таблица **address** содержит следующие колонки:

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]

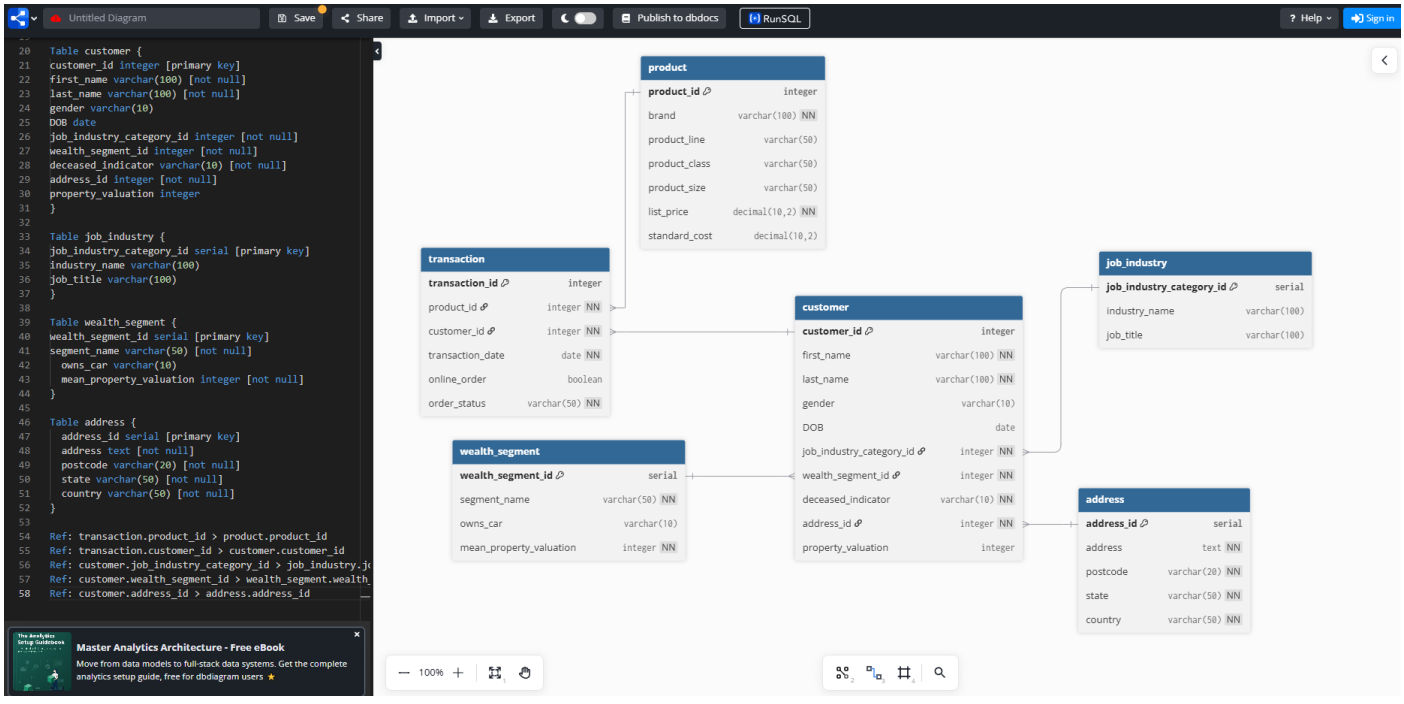
Эта база данных имеет 3НФ (третью нормальную форму) потому что:

1. Соотвествует 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 – 3НФ

Далее работаем с базой данных в программе DBeaver 25.2.4.

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

Создаем две таблицы **transaction_old** и **customer_old** с колонками, соответствующим 1НФ.

Эти таблицы нужны только для того, чтобы импортировать данные.

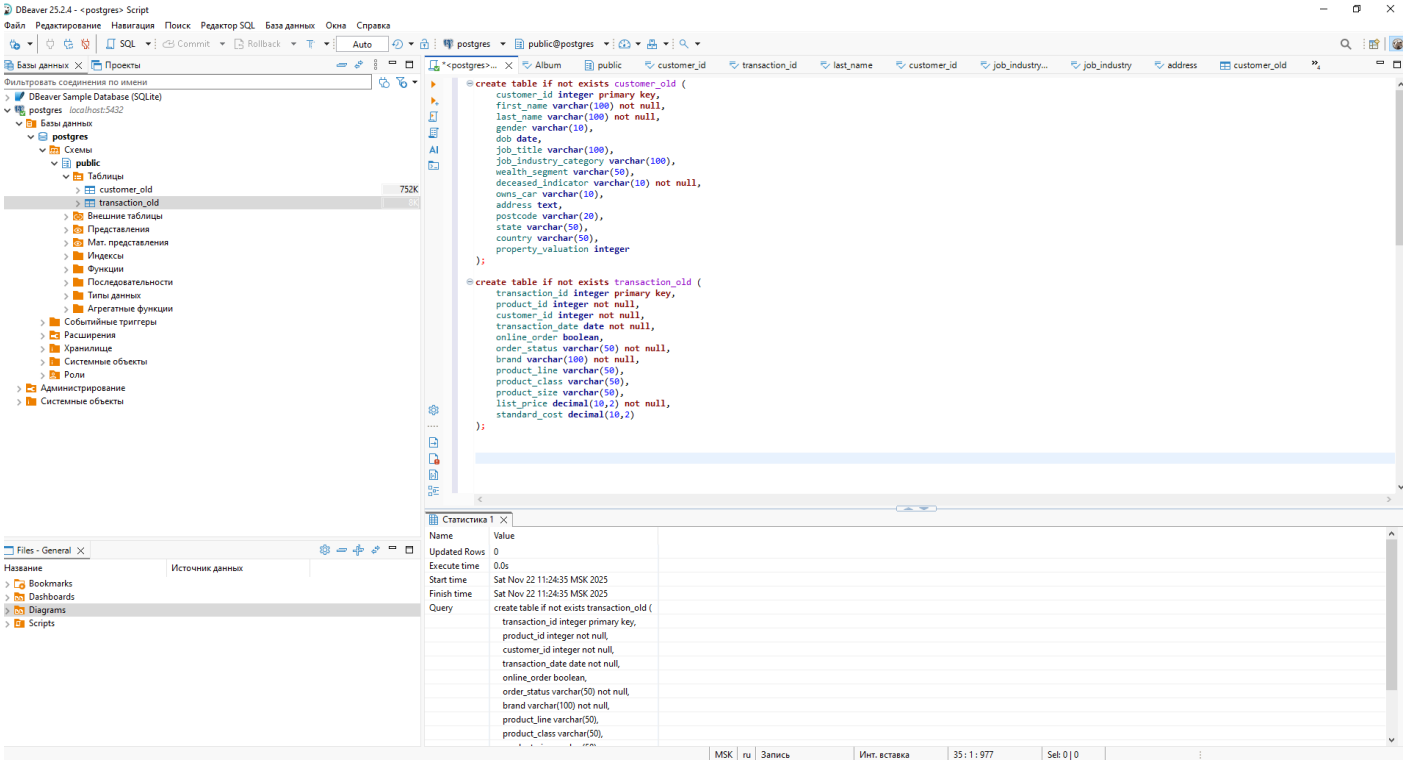


Рисунок 4 – Создание таблиц **transaction_old** и **customer_old**

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

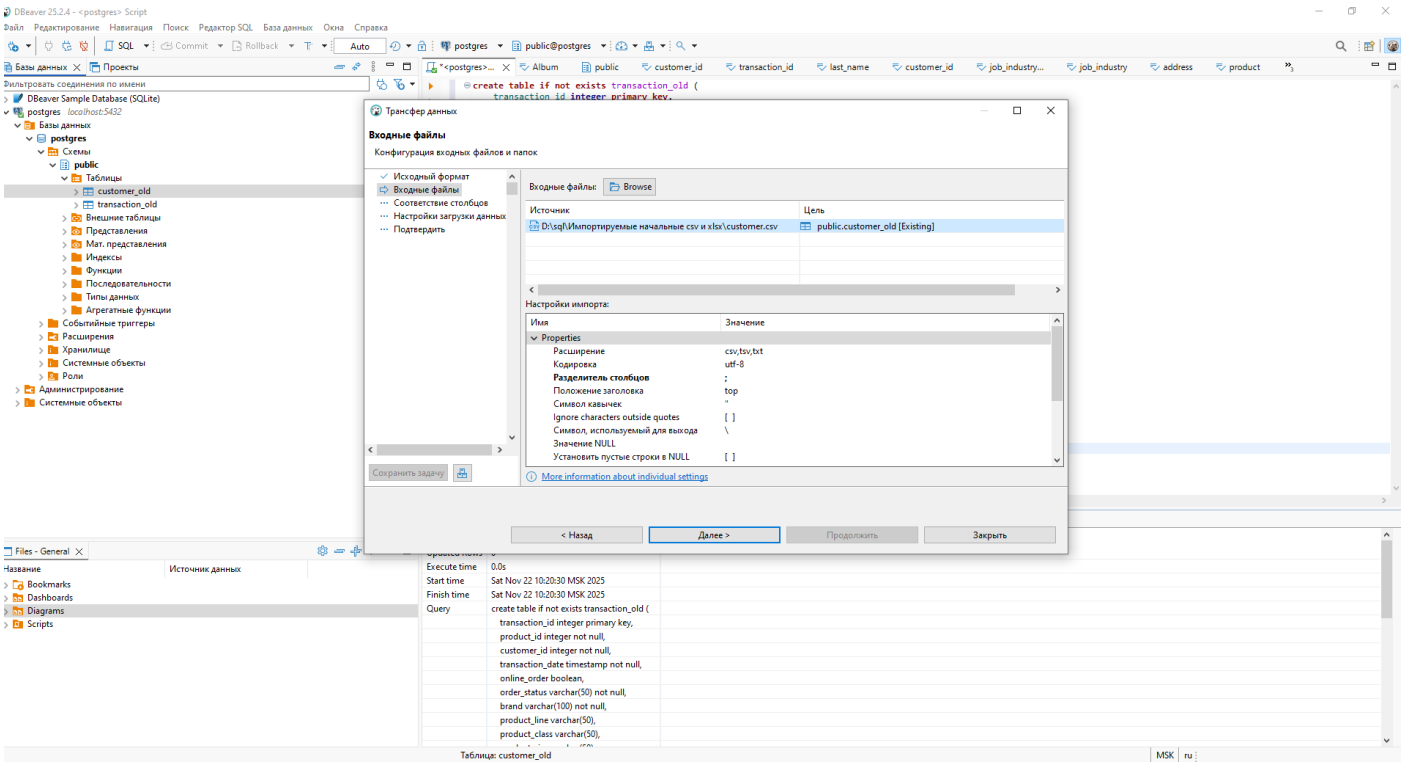


Рисунок 5 – Импорт данных в таблицу **customer_old**

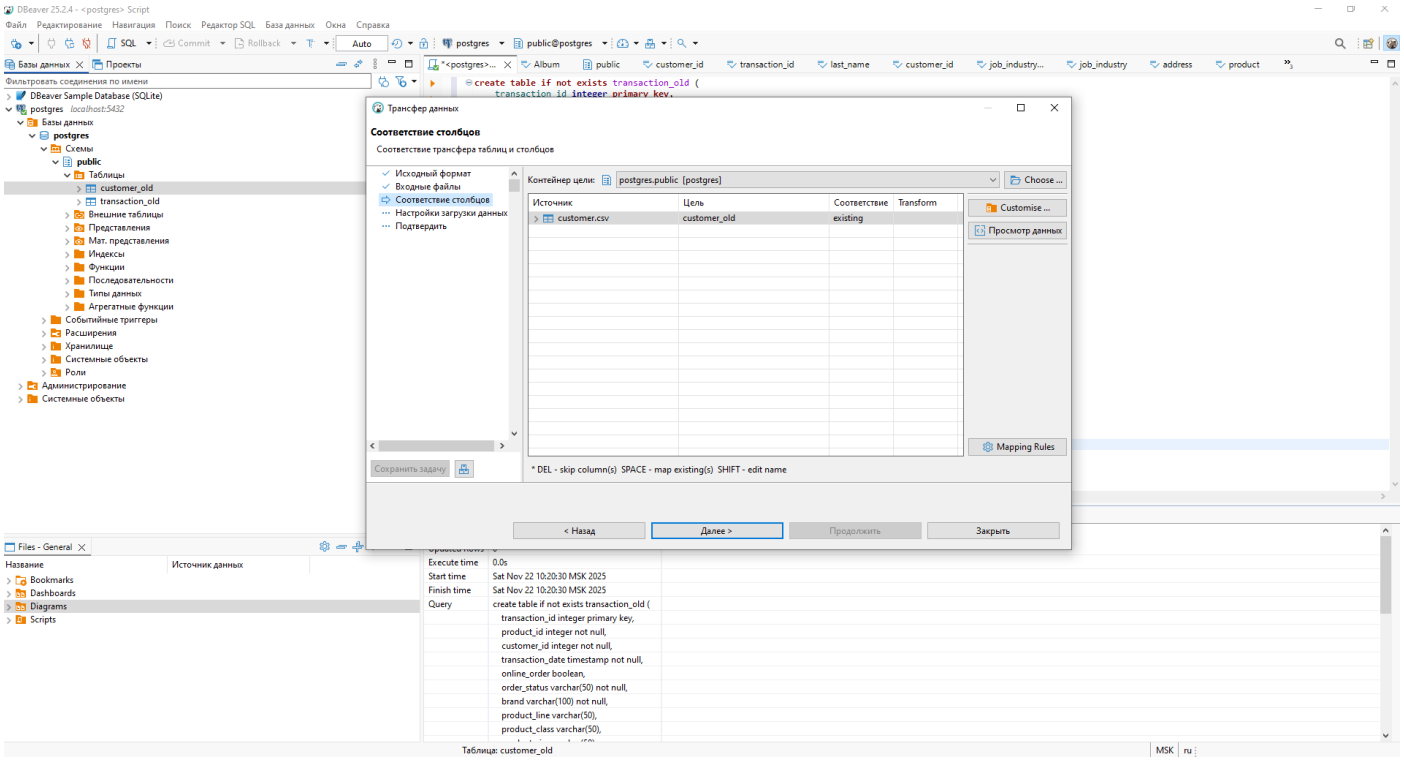


Рисунок 6 – Импорт данных в таблицу **customer_old**

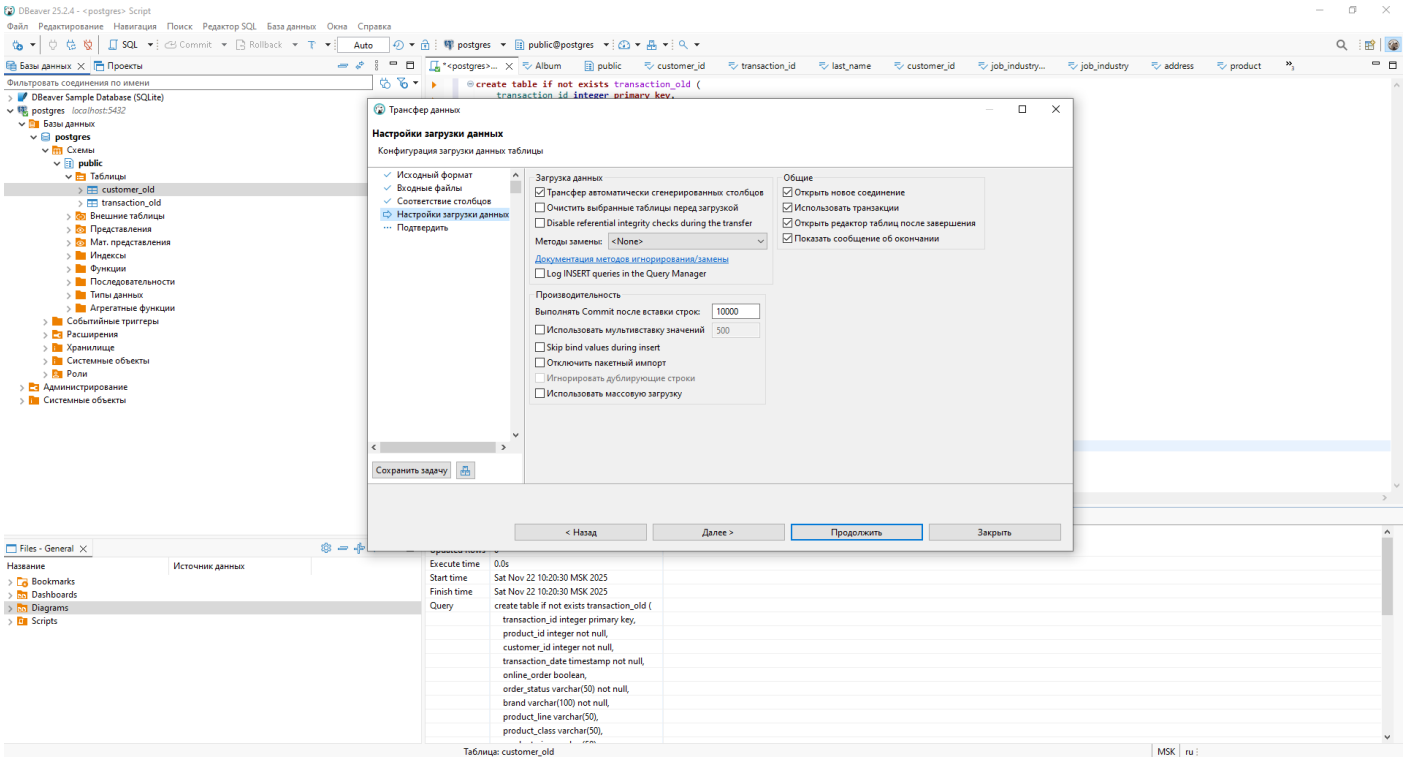


Рисунок 7 – Импорт данных в таблицу **customer_old**

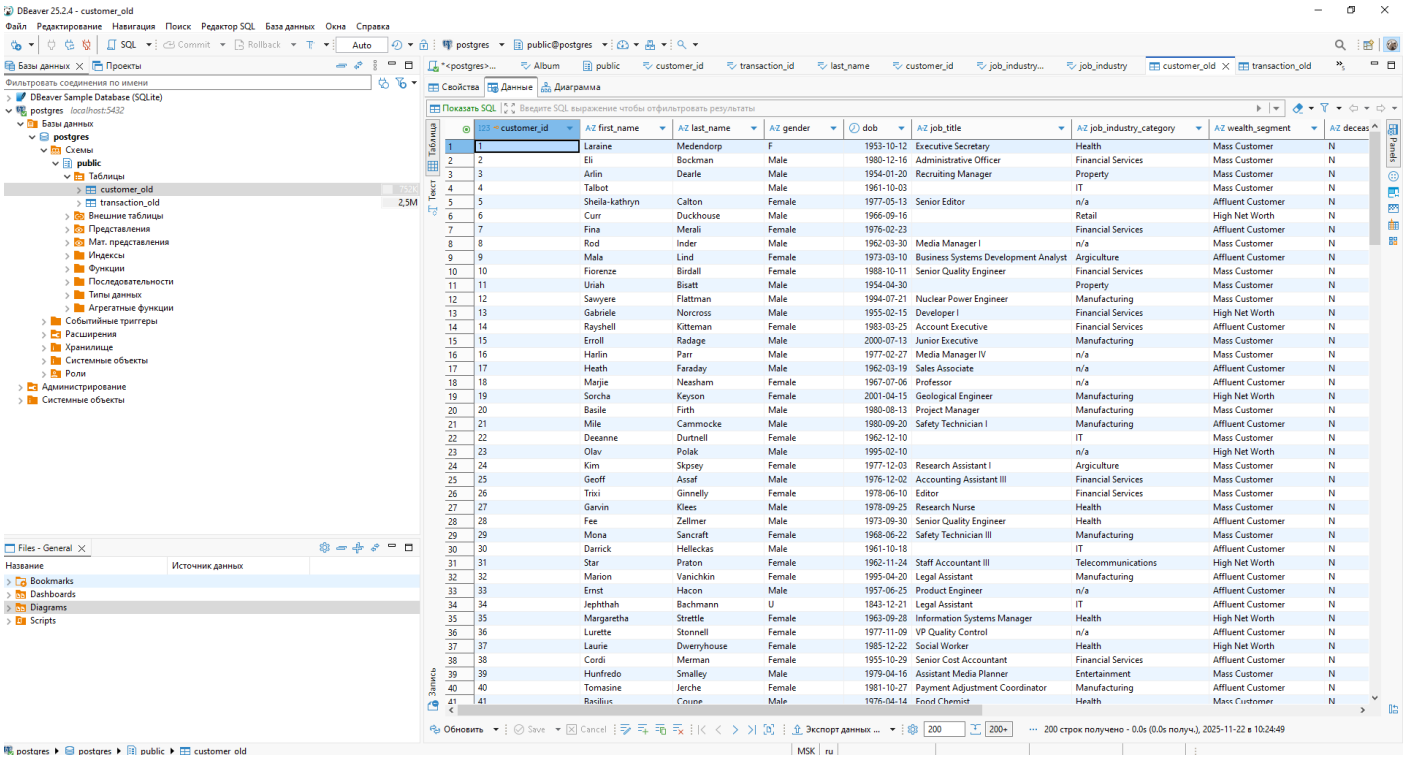


Рисунок 8 – Просмотр данных таблицы **customer_old**

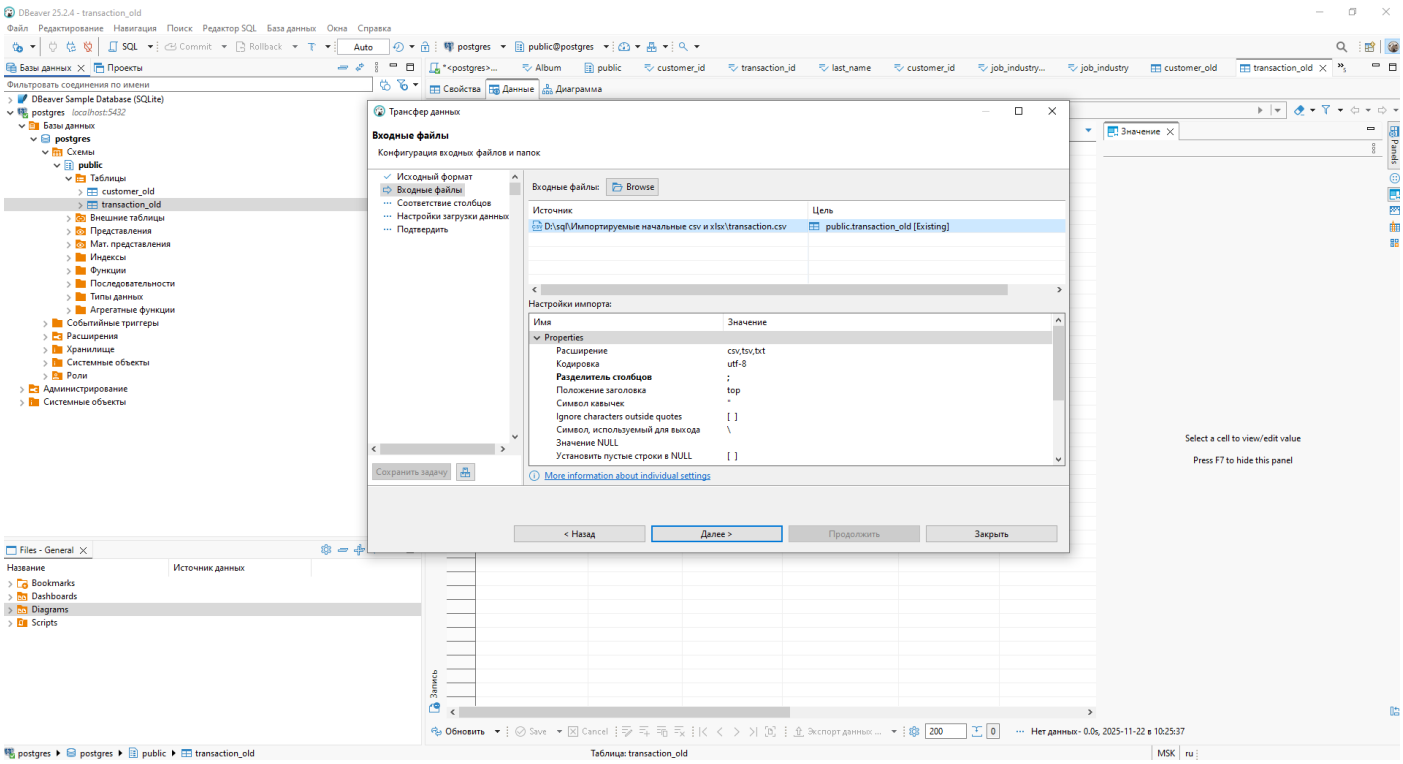


Рисунок 9 – Импорт данных в таблицу **transaction_old**

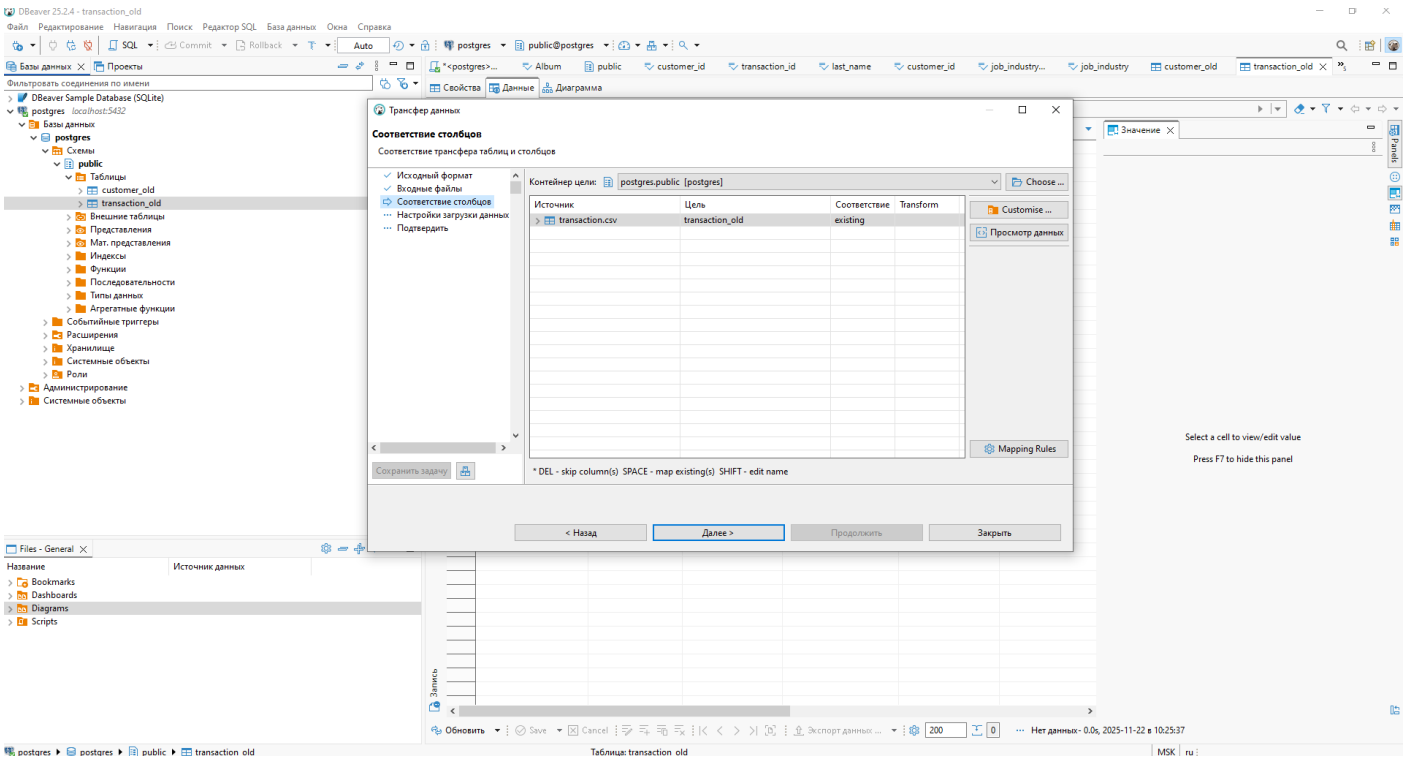


Рисунок 10 – Импорт данных в таблицу **transaction_old**

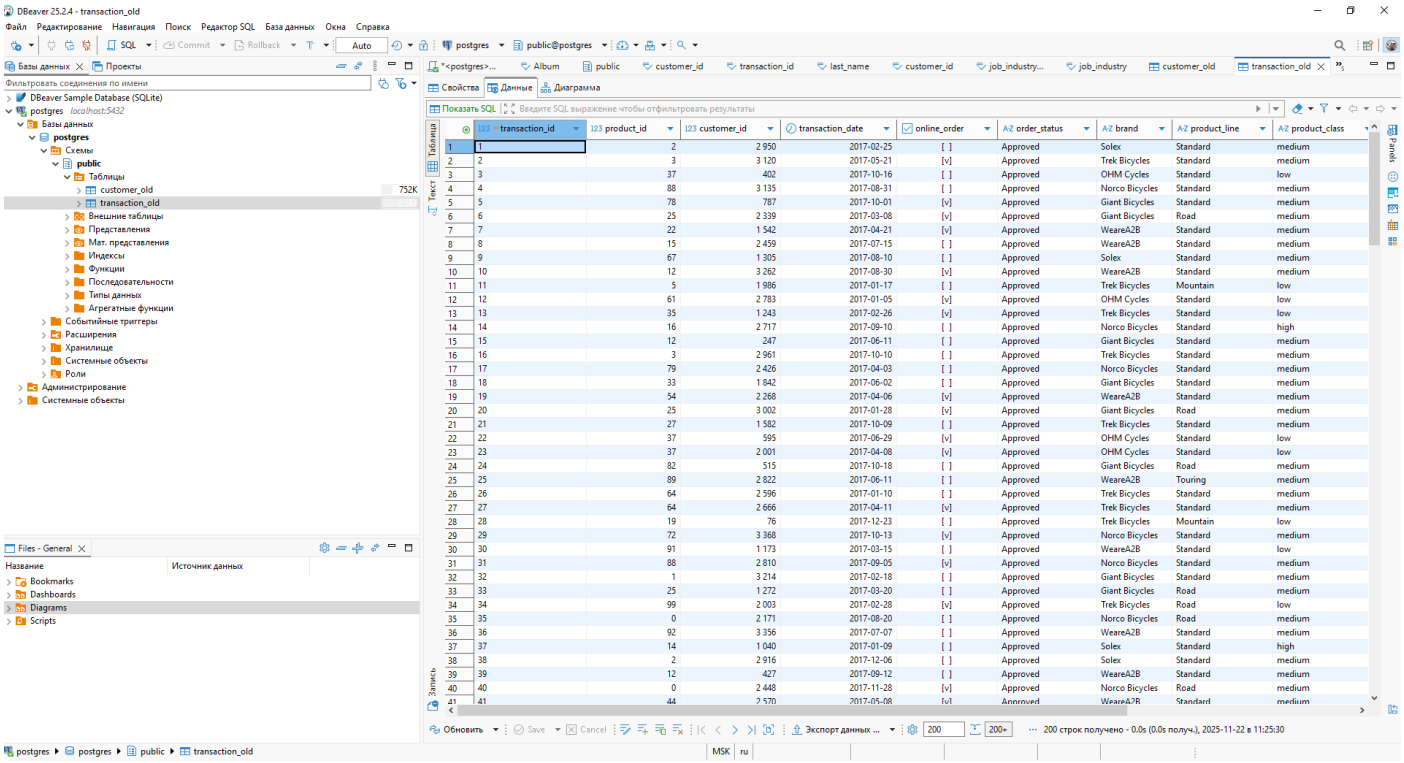
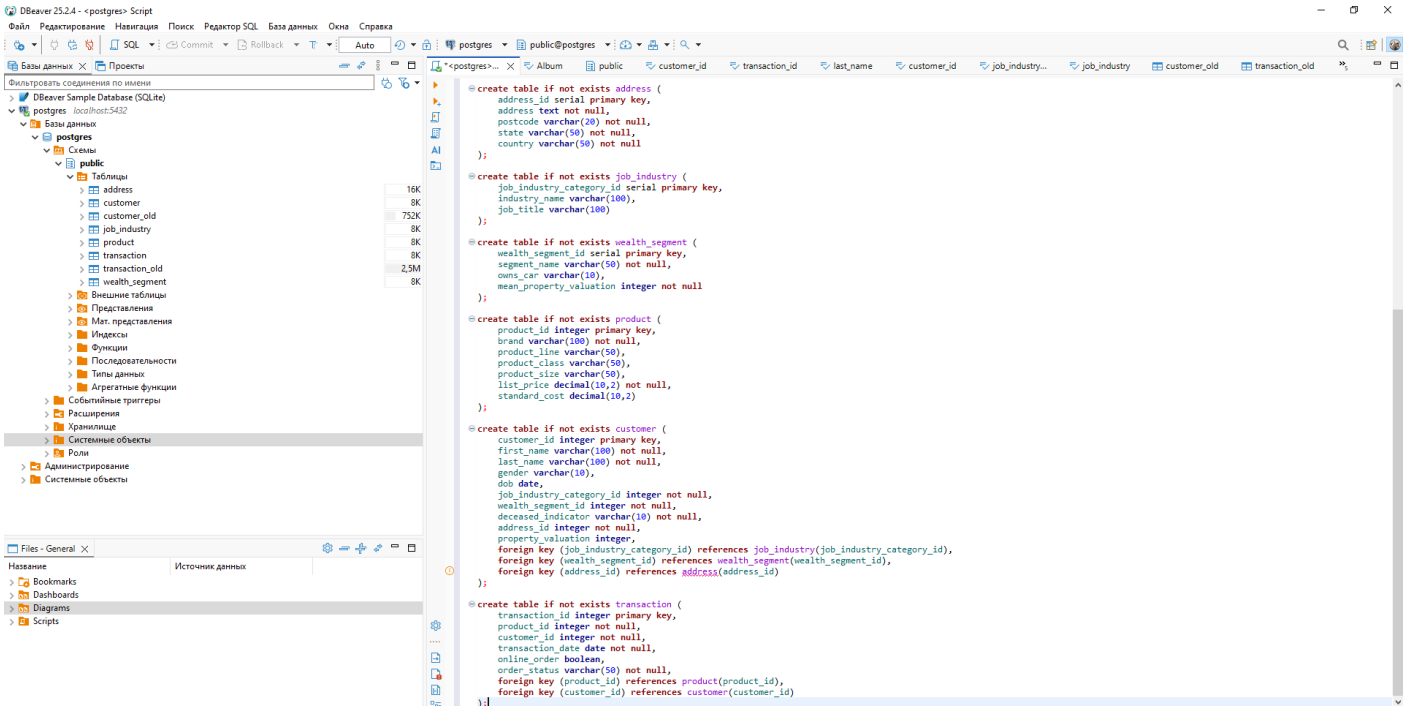


Рисунок 11 – Просмотр данных таблицы **transaction_old**

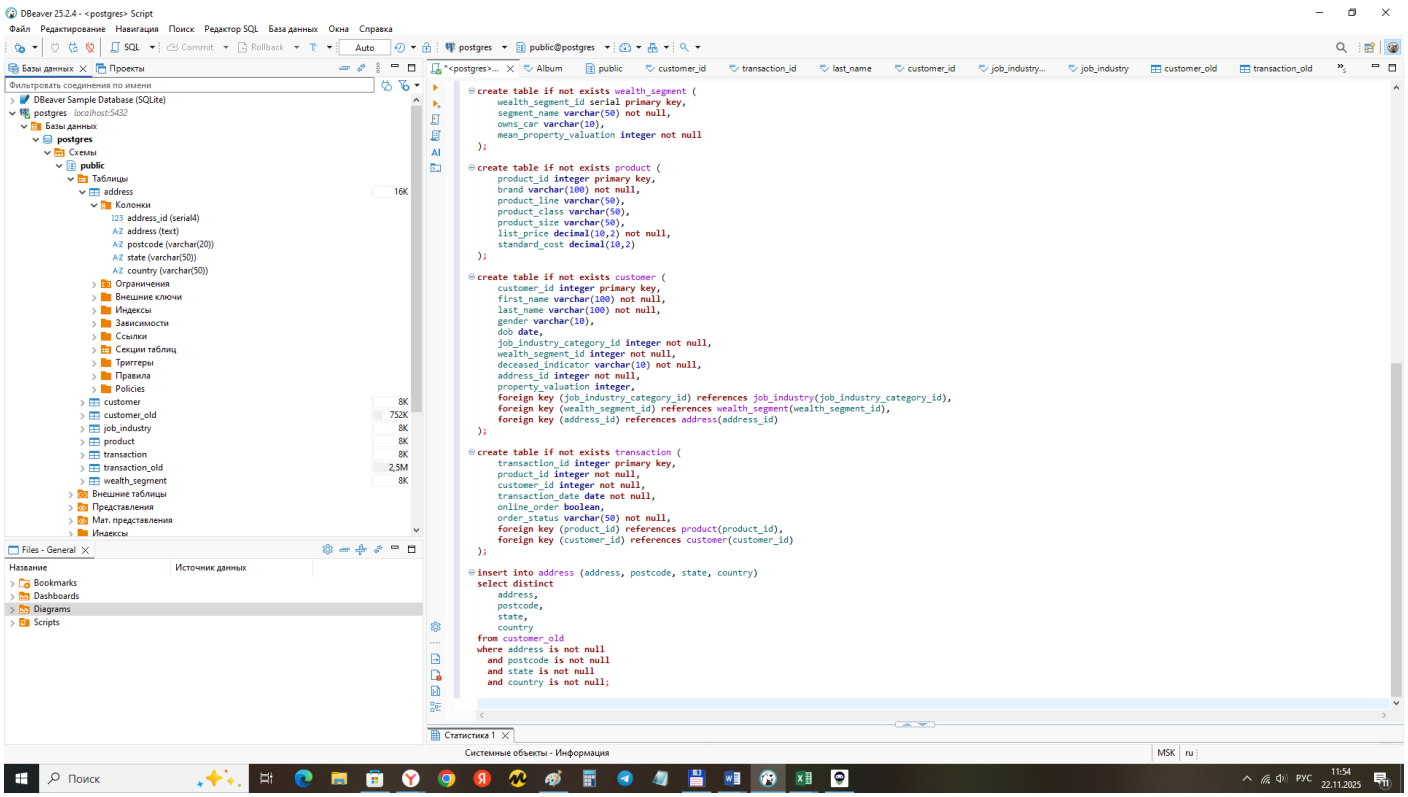
Создаем таблицы: **job_industry**, **wealth_segment**, **address**, **product**, **customer**, **transaction** с колонками и типами данных, соответствующими 3НФ.



```
create table if not exists address (
    address_id serial primary key,
    address text not null,
    postcode varchar(20) not null,
    state varchar(50) not null,
    country varchar(50) not null
);

create table if not exists job_industry (
    job_industry_category_id serial primary key,
    industry_name varchar(100),
    job_title varchar(100)
);

create table if not exists wealth_segment (
    wealth_segment_id serial primary key,
    segment_name varchar(50) not null,
    owns_car varchar(10),
    mean_property_valuation integer not null
);

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)
);
```

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

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



```
create table if not exists wealth_segment (
    wealth_segment_id serial primary key,
    segment_name varchar(50) not null,
    owns_car varchar(10),
    mean_property_valuation integer not null
);

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;
```

Рисунок 13 – Заполнение таблицы **address**

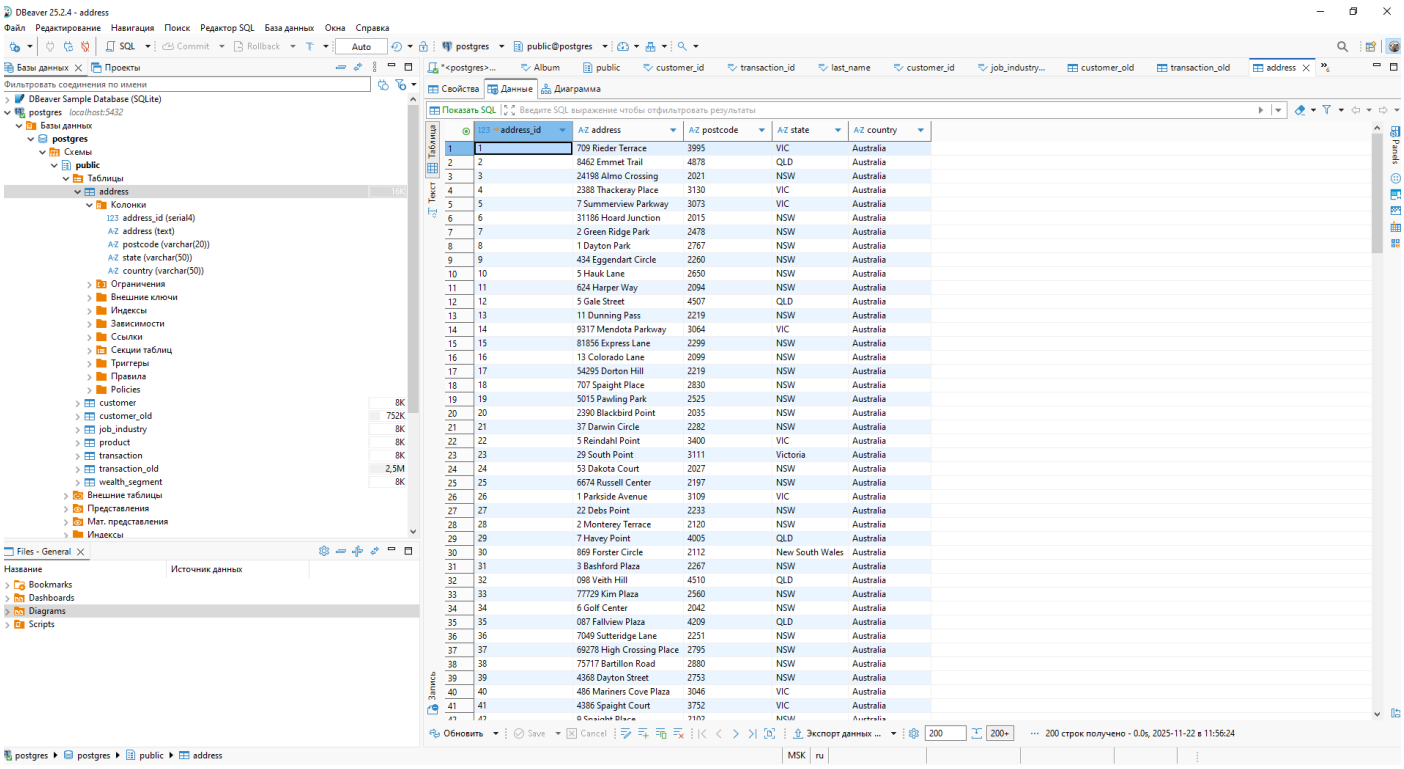
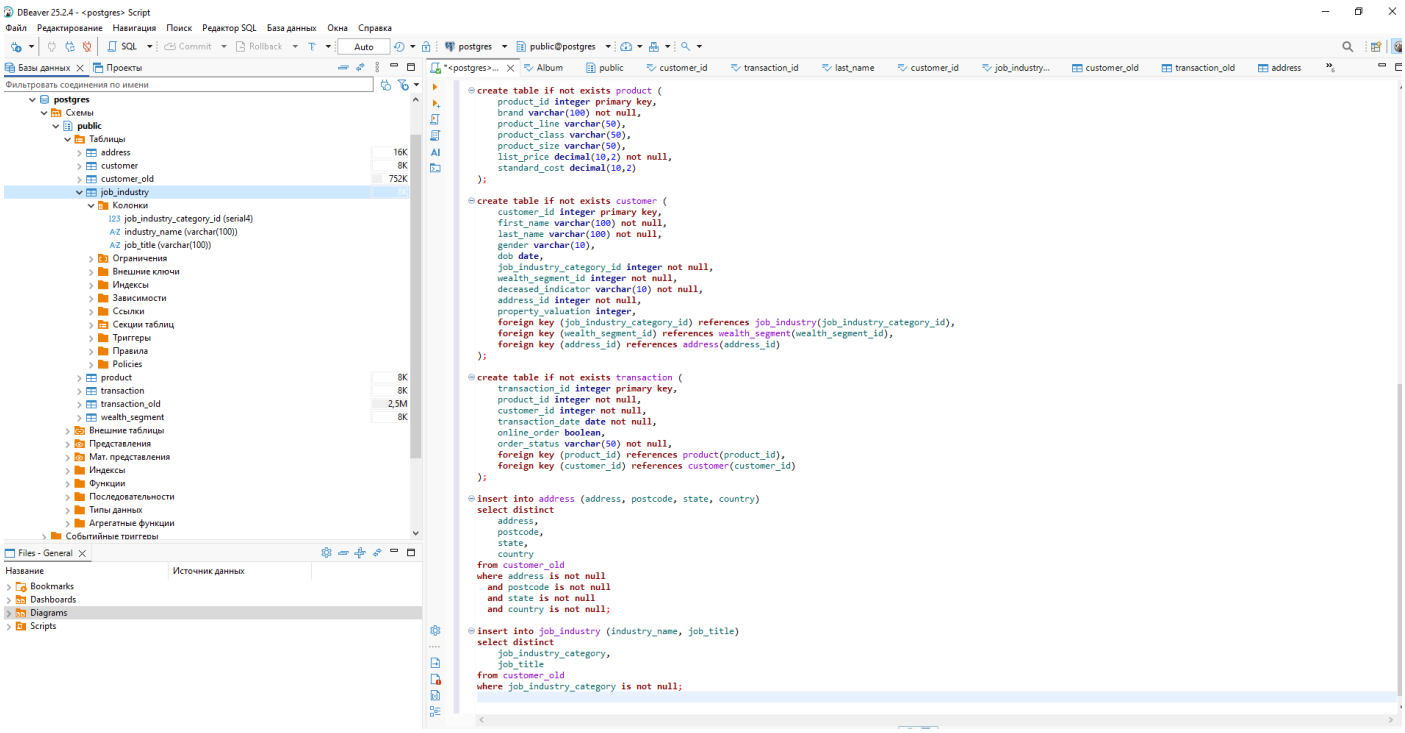


Рисунок 14 – Просмотр данных таблицы **address**

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**

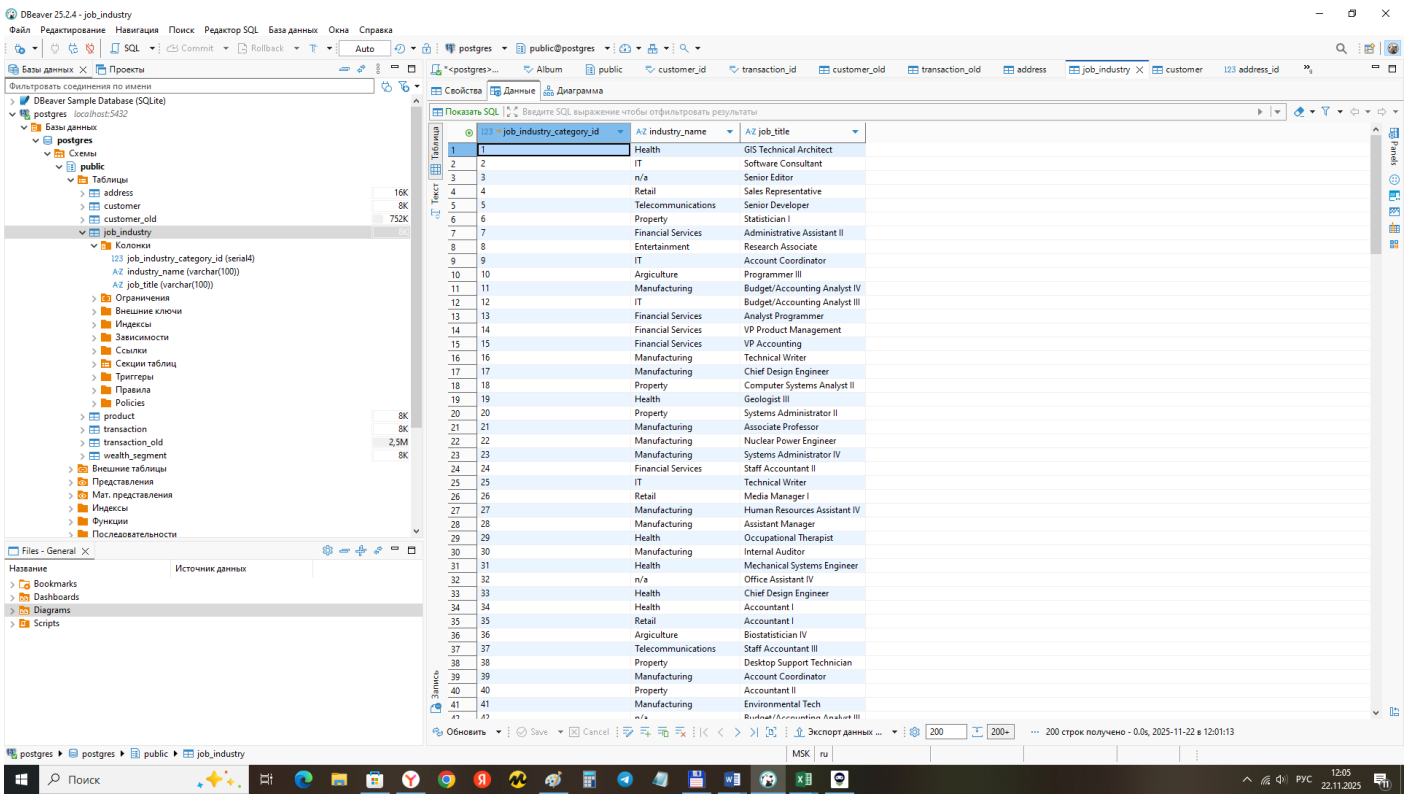
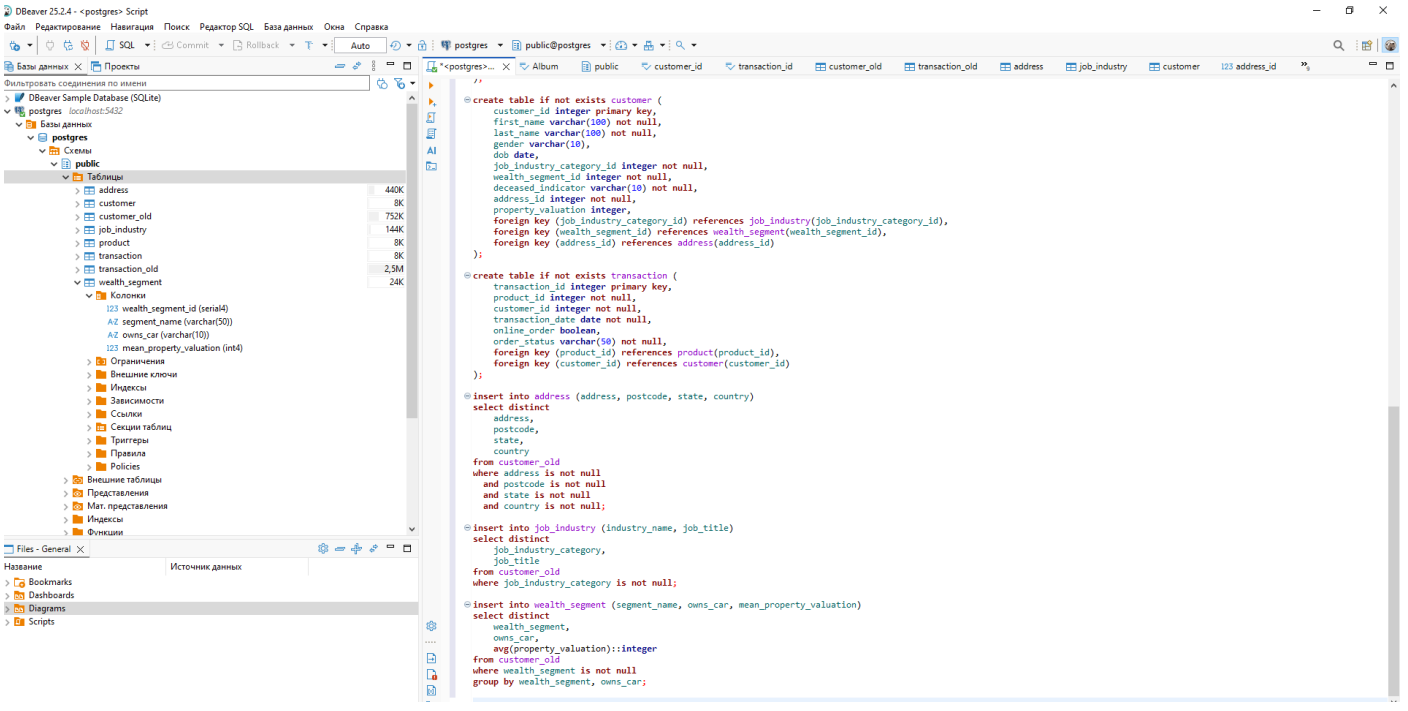


Рисунок 16 – Просмотр данных таблицы **job_industry**

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



```
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;

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

Рисунок 17 – Заполнение таблицы **wealth_segment**

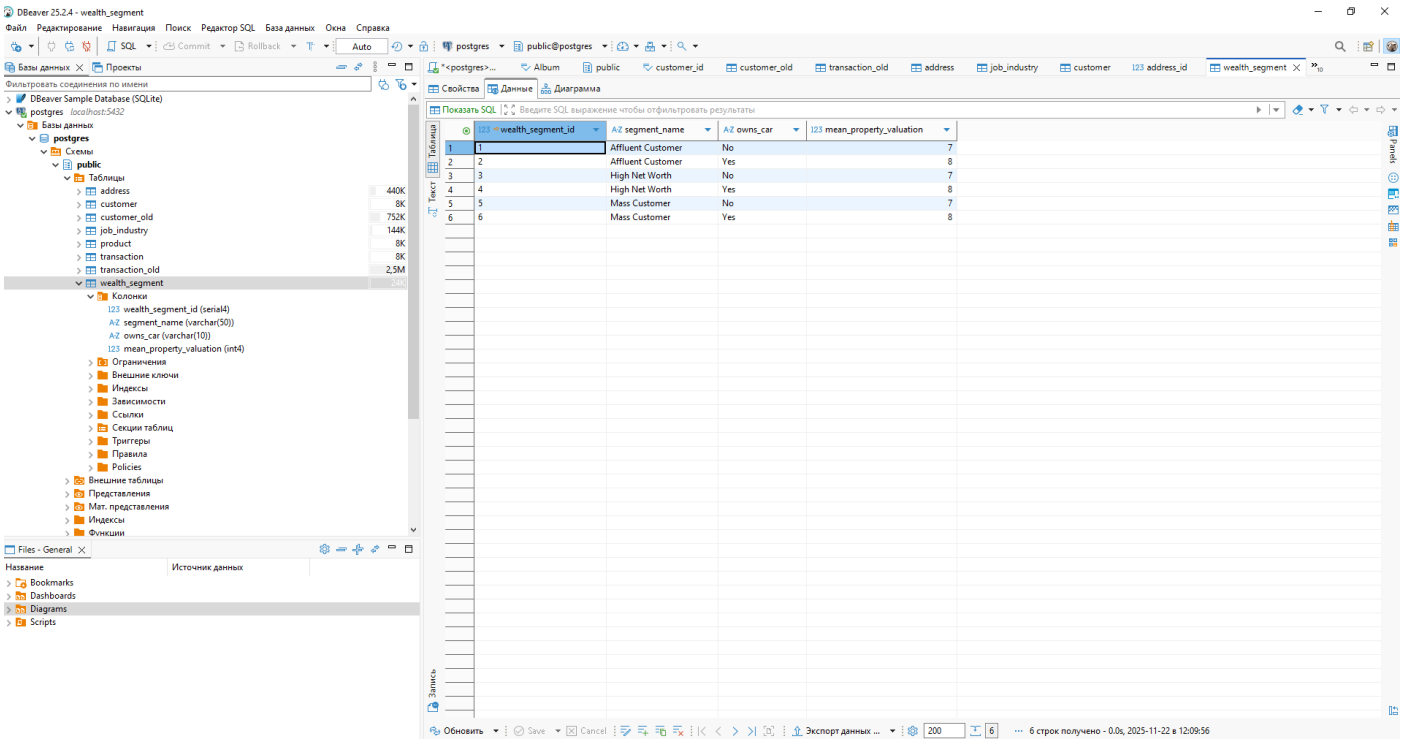


Рисунок 18 – Просмотр данных таблицы **wealth_segment**

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

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

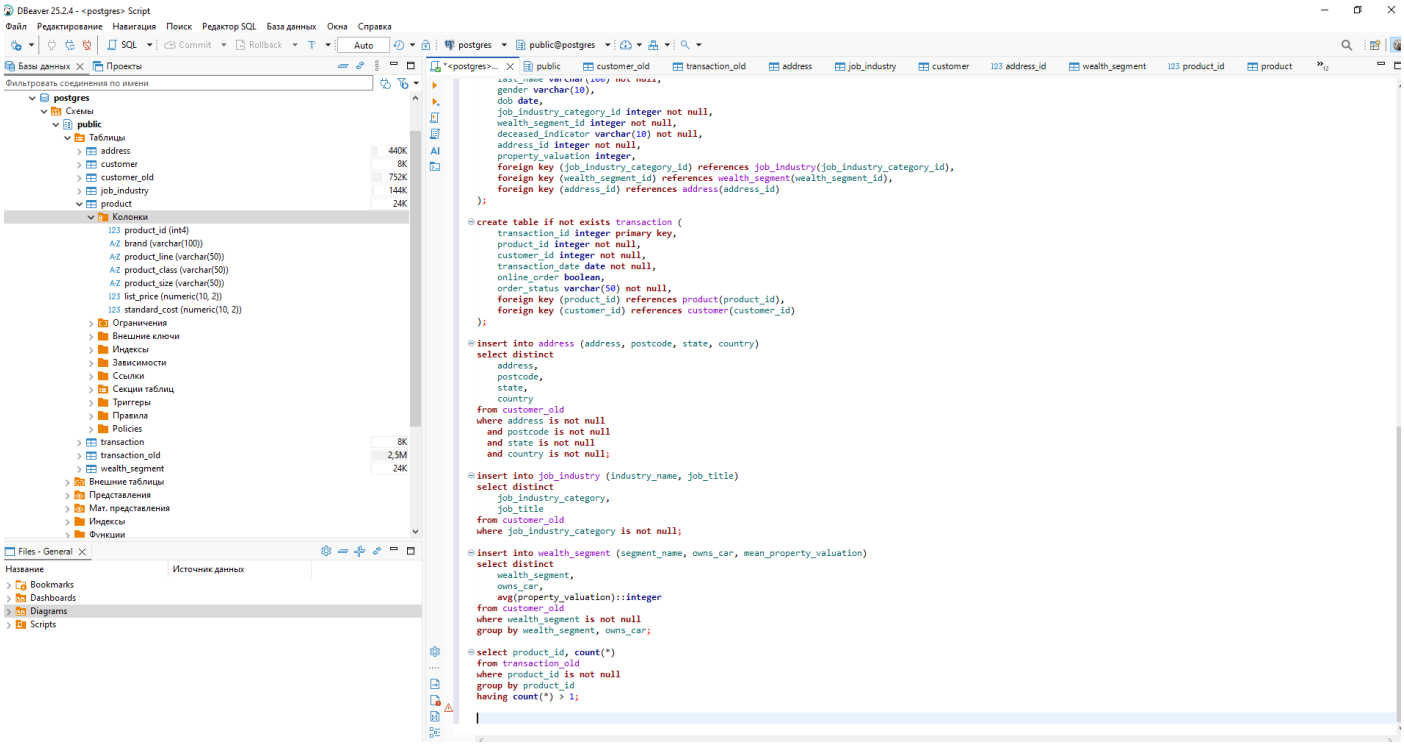


Рисунок 19 – Проверка на дубликаты product_id

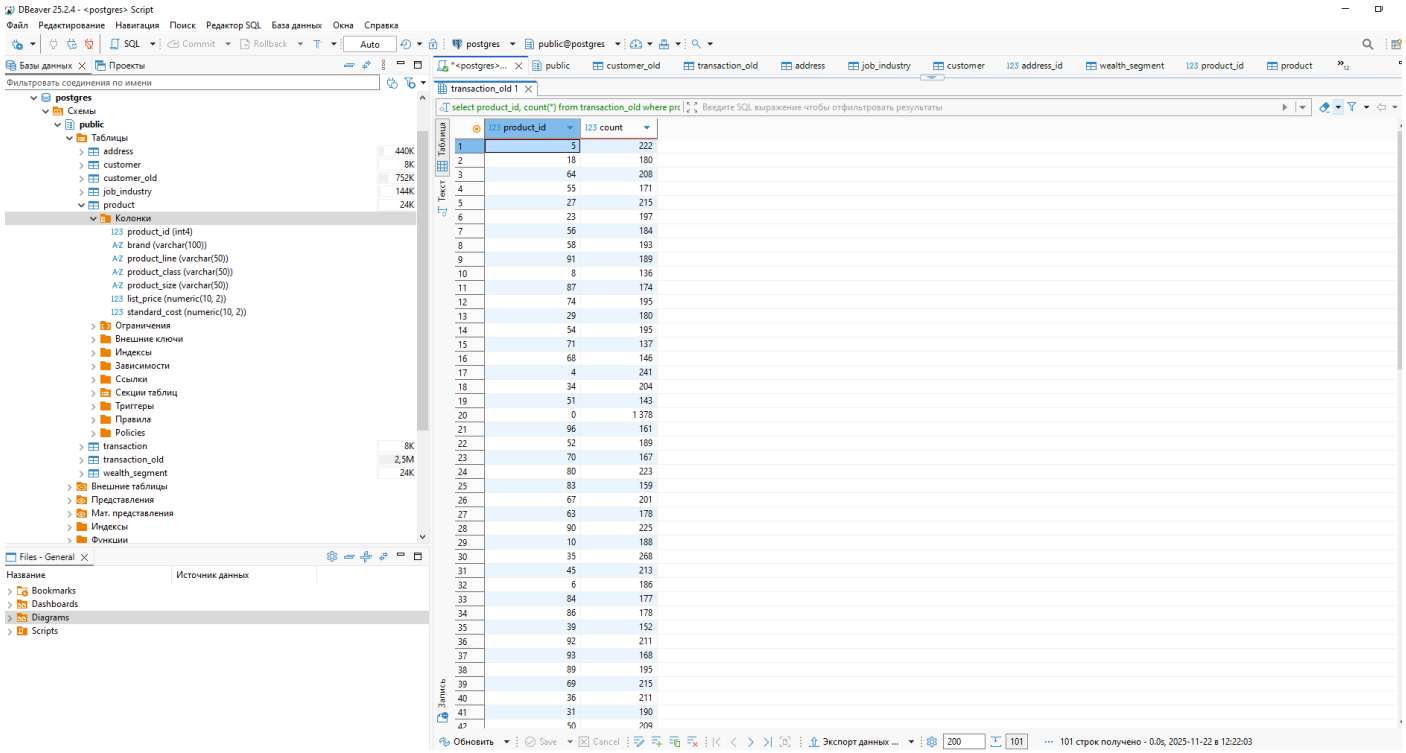
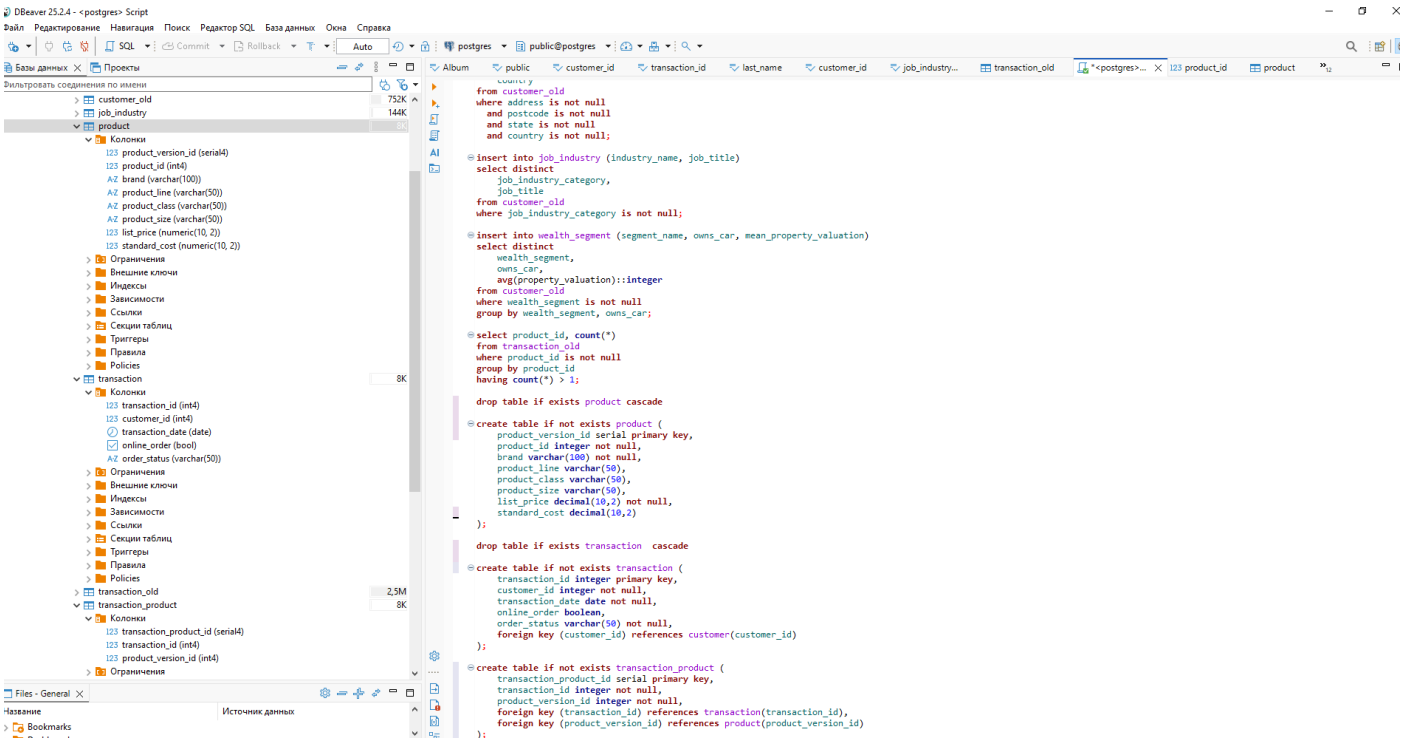


Рисунок 20 – Просмотр дубликатов product_id

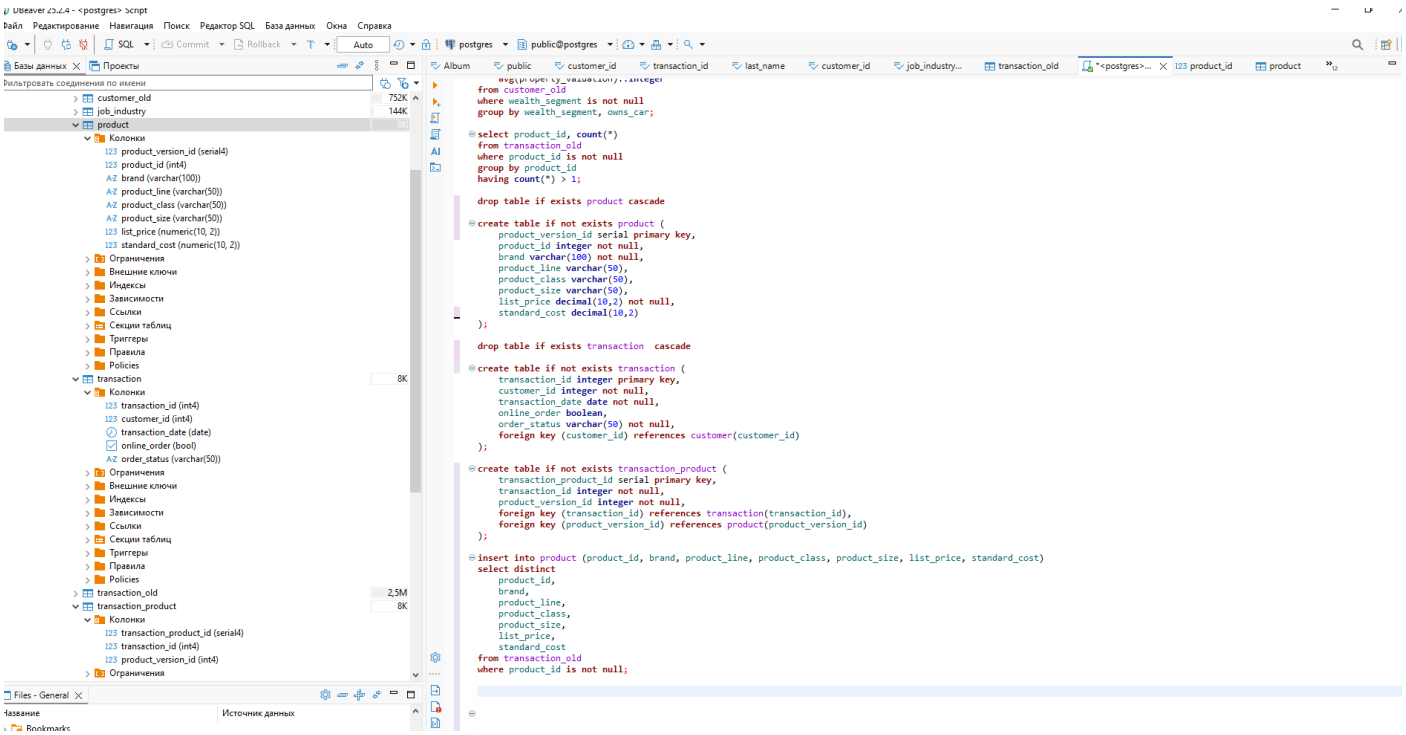
Тогда необходимо изменить таблицы **product**, **transaction** и добавать новую таблицу связей “многие ко многим” **transaction_product**.



```
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(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;

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)
);
```

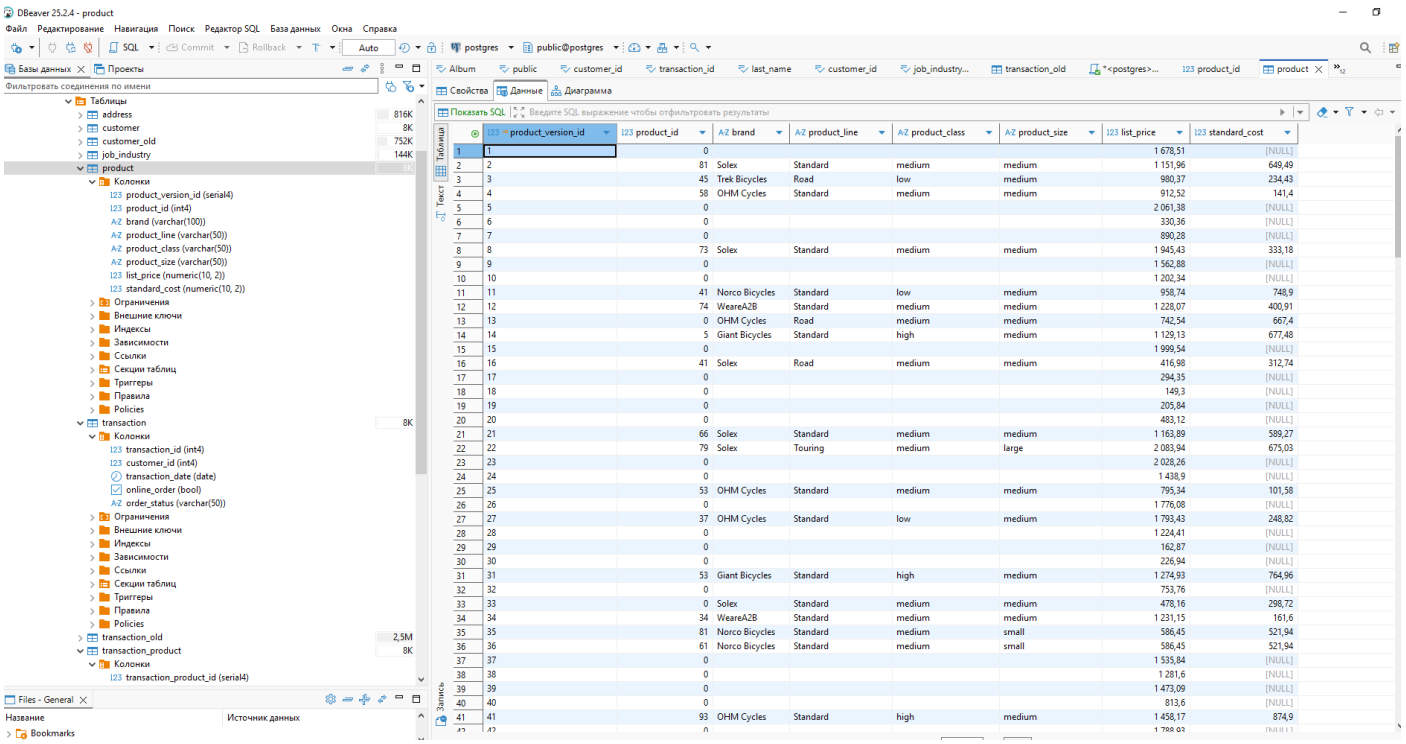
Рисунок 21 – Создание таблицы **transaction_product**
и удаление и повторное создание таблицы **transaction**

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



```
    avg(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;

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)
);

insert into product (product_id, brand, product_line, product_class, product_size, list_price, standard_cost)
select distinct
    product_id,
    brand,
    product_line,
    product_class,
    product_size,
    list_price,
    standard_cost
from transaction_old
where product_id is not null;
```

Рисунок 22 – Заполнение таблицы **product**



	product_version_id	product_id	brand	product_line	product_class	product_size	list_price	standard_cost
1	1	0					1 678,51	[NULL]
2	2	81	Solex	Standard	medium	medium	1 151,96	649,49
3	3	45	Trek Bicycles	Road	low	medium	980,37	234,43
4	4	58	OHM Cycles	Standard	medium	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	medium	1 945,43	333,18
9	9	0					1 562,88	[NULL]
10	10	0					1 202,34	[NULL]
11	11	41	Norco Bicycles	Standard	low	medium	958,74	748,9
12	12	74	WeareA2B	Standard	medium	medium	1 228,07	400,91
13	13	0	OHM Cycles	Road	medium	medium	742,54	667,4
14	14	5	Giant Bicycles	Standard	high	medium	1 129,13	677,48
15	15	0					1 999,54	[NULL]
16	16	41	Solex	Road	medium	medium	416,98	312,74
17	17	0					294,35	[NULL]
18	18	0					149,3	[NULL]
19	19	0					205,84	[NULL]
20	20	0					483,12	[NULL]
21	21	66	Solex	Standard	medium	medium	1 163,89	589,27
22	22	79	Solex	Touring	medium	large	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	medium	795,34	101,58
26	26	0					1 776,08	[NULL]
27	27	37	OHM Cycles	Standard	low	medium	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	medium	1 274,93	764,96
32	32	0					753,76	[NULL]
33	33	0	Solex	Standard	medium	medium	478,16	298,72
34	34	34	WeareA2B	Standard	medium	medium	1 231,15	161,6
35	35	81	Norco Bicycles	Standard	medium	small	586,45	521,94
36	36	61	Norco Bicycles	Standard	medium	small	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	medium	1 458,17	874,9

Рисунок 23 – Просмотр данных таблицы **product**

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

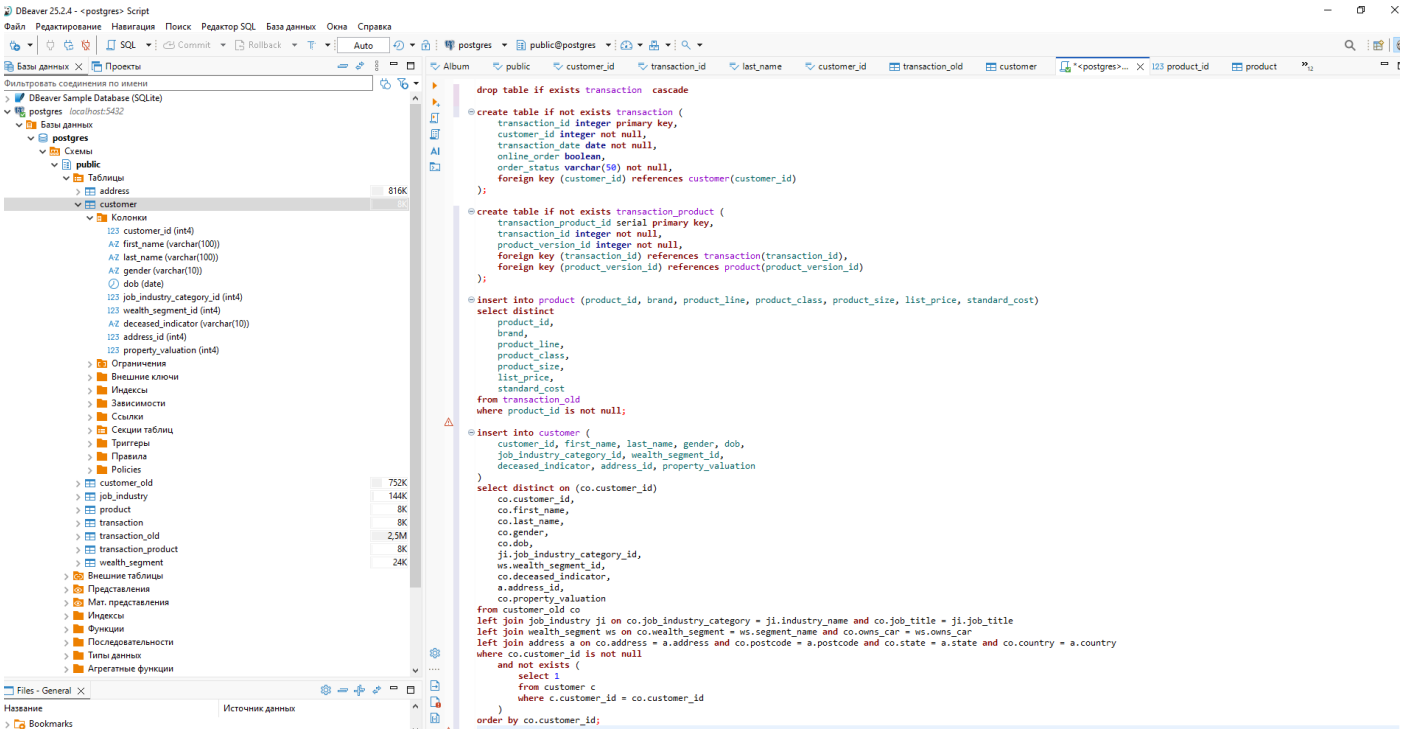


Рисунок 24 – Заполнение таблицы **customer**

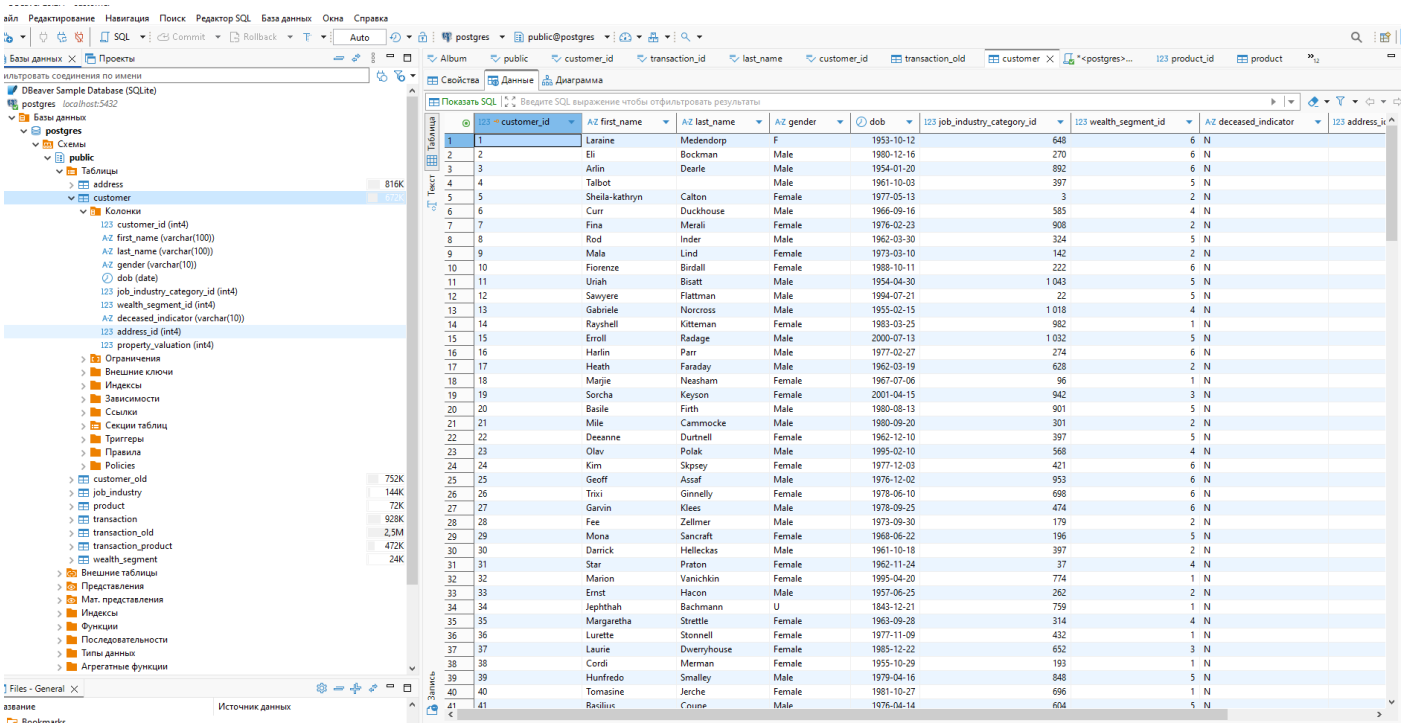


Рисунок 25 – Просмотр данных таблицы **customer**

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

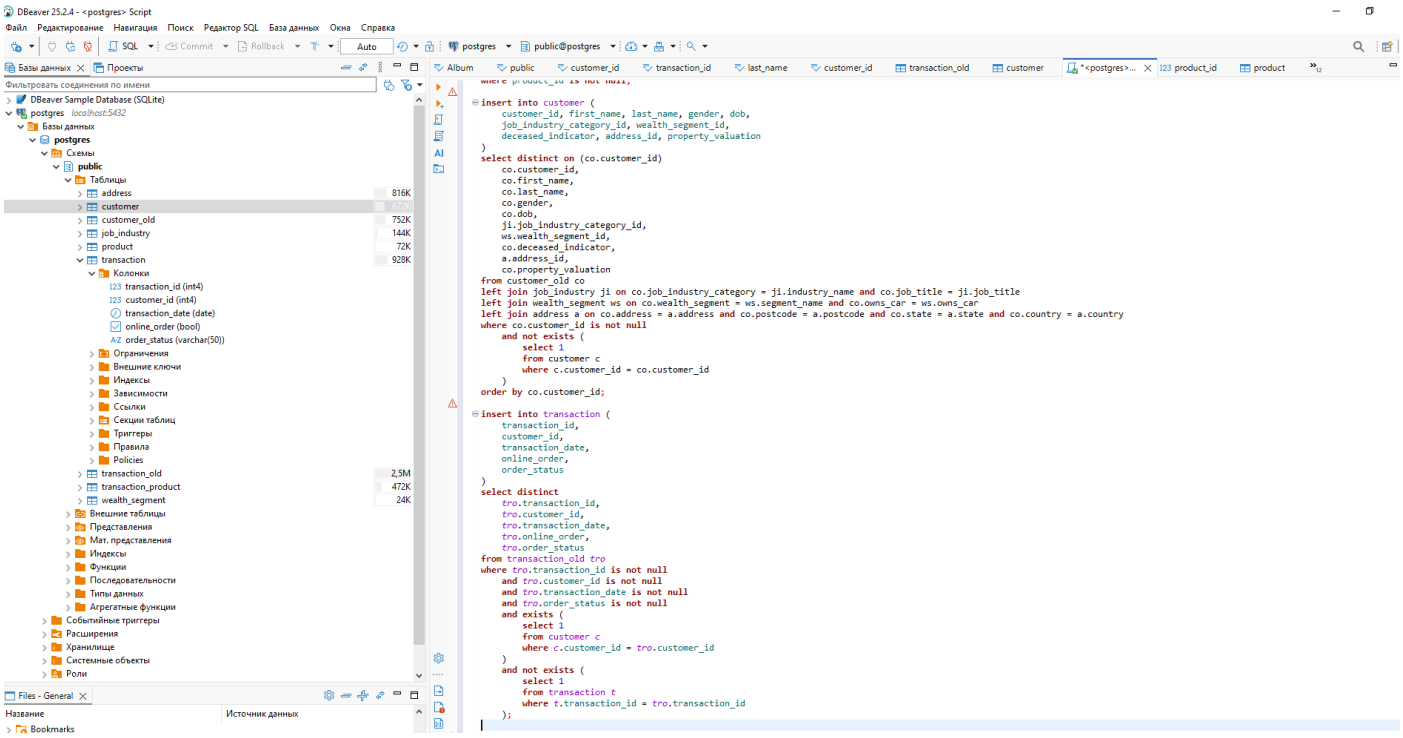


Рисунок 26 – Заполнение таблицы **transaction**

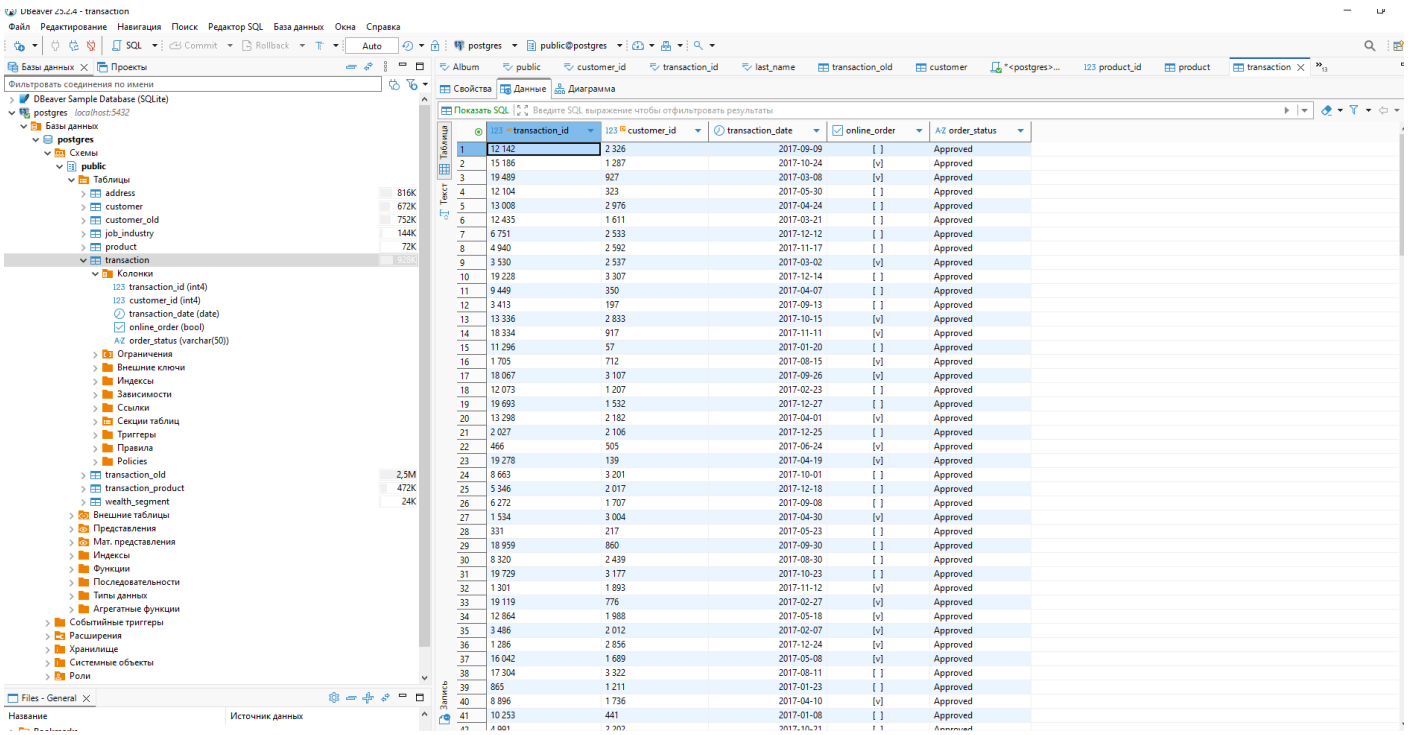


Рисунок 27 – Просмотр данных таблицы **transaction**

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

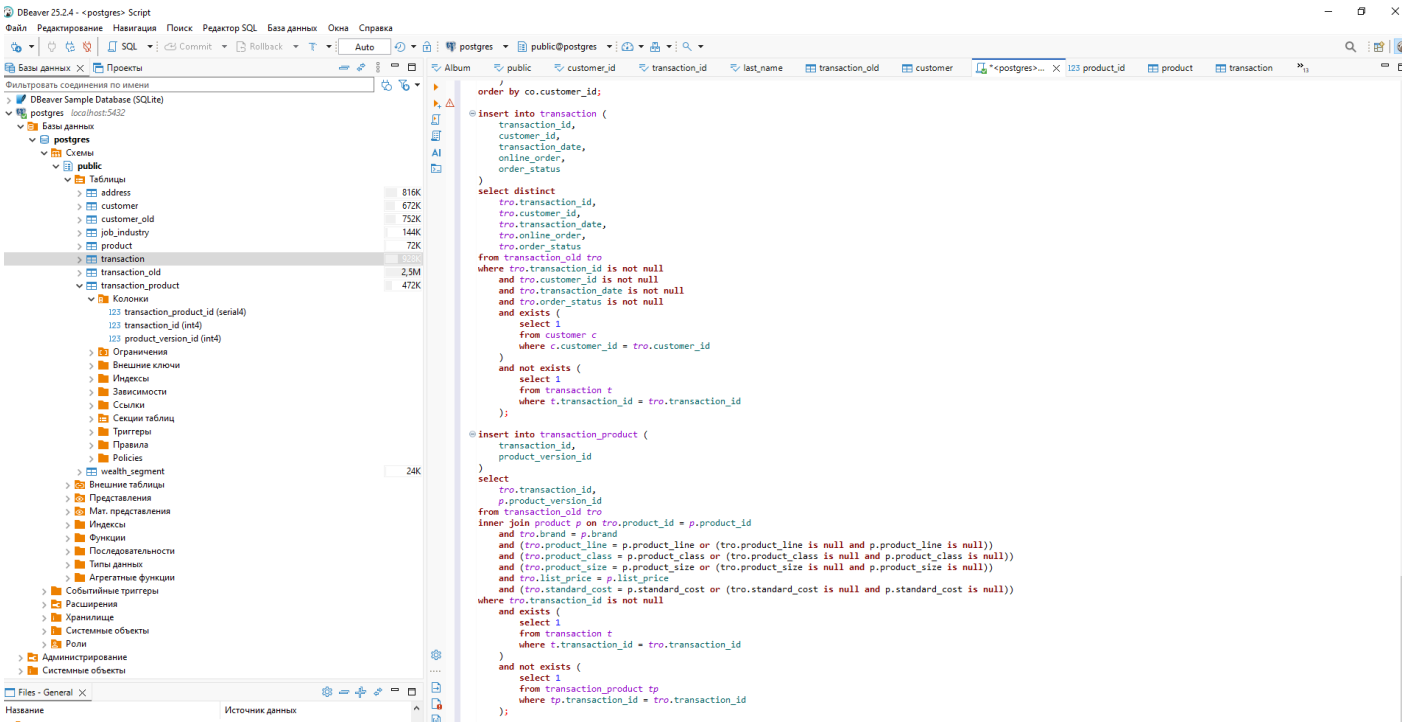


Рисунок 26 – Заполнение таблицы **transaction_product**

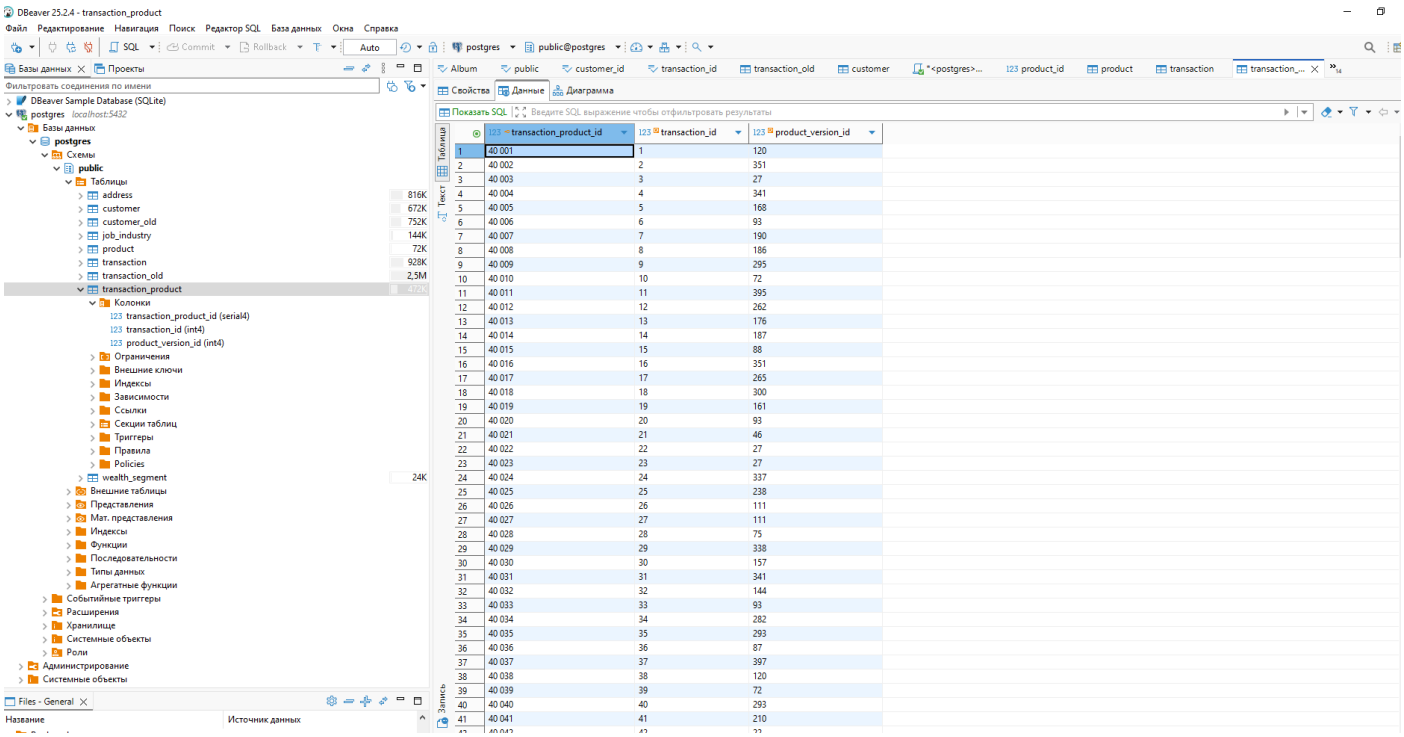


Рисунок 27 – Просмотр данных таблицы **transaction_product**

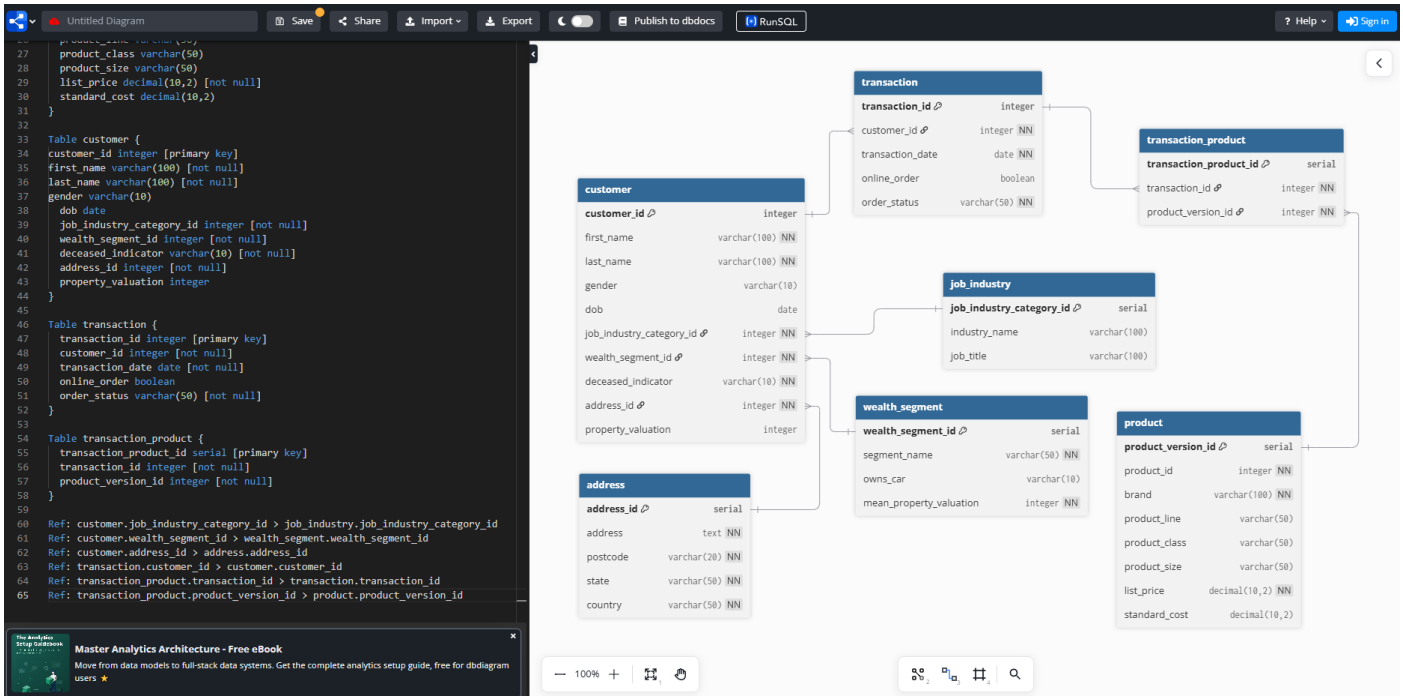


Рисунок 28 – Итоговая структура 3НФ

Преобразованную до 3НФ базу данных в формате .csv экспортированную из таблиц: **job_industry**, **wealth_segment**, **address**, **product**, **customer**, **transaction, transaction_product** можно найти в папке “ Экспортируемые итоговые csv”.