Отчет по работе над тестовым заданием

на курсы ШИФТа

по направлению – разработка ЦФТ-Банк

Ссылка на тестовое задание: <https://drive.google.com/file/d/1cD1diy5BSaoFnsTaaIx5vF8FpTl_OA_m/view?usp=sharing>

При выполнении поставленных задач применялась СУБД - PostgreSQL

Ссылка на github: <https://github.com/AlexAltR/task-test-SHIFT-CFT-Bank.git>

Дата начала работы: 10.09.21г.

Дата окончания работы: 16.09.21г.

Выполнил Петрищев А.С.

Телефон: 8-962-792-20-85

**Примечания к заданию:**

1. Почему на ER-диаграмме у таблицы CLIENTS связь один к одному с таблицей ACCOUNTS? Я считаю, что связь должна быть один ко многим, так как у одного клиента может быть множество счетов.

**Задания:**

1. Подготовьте DDL-скрипты создания объектов для приведённой модели: создание таблиц, первичных, уникальных, внешних ключей и т.д.

Скрипт создания БД «bank» представлен на листинге 1.

/\*DDL-скрипт создания базы данных "Bank".\*/

CREATE DATABASE "Bank"

WITH

OWNER = postgres

ENCODING = 'UTF8'

CONNECTION LIMIT = -1;

Листинг 1 - Скрипт создания БД «bank»

Скрипт создания таблицы CLIENTS с заданными ограничениями представлен на листинге 2.

/\*DDL-скрипт создания таблицы CLIENTS с заданными ограничениями.\*/

CREATE TABLE public."CLIENTS"

(

"ID" numeric(10) NOT NULL,

"NAME" character varying(1000) NOT NULL,

"PLACE\_OF\_BIRTH" character varying(1000) NOT NULL,

"DATE\_OF\_BIRTH" date NOT NULL,

"ADDRESS" character varying(1000) NOT NULL,

"PASSPORT" character varying(100) NOT NULL,

PRIMARY KEY ("ID")

)

WITH (

OIDS = FALSE

);

ALTER TABLE public."CLIENTS"

OWNER to postgres;

Листинг 2 - Скрипт создания таблицы CLIENTS с заданными ограничениями

Скрипт создания таблицы TARIFS с заданными ограничениями представлен на листинге 3.

/\*DDL-скрипт создания таблицы TARIFS с заданными ограничениями.\*/

CREATE TABLE public."TARIFS"

(

"ID" numeric(10) NOT NULL,

"NAME" character varying(100) NOT NULL,

"COST" numeric(10, 2) NOT NULL,

PRIMARY KEY ("ID")

)

WITH (

OIDS = FALSE

);

ALTER TABLE public."TARIFS"

OWNER to postgres;

Листинг 3 - Скрипт создания таблицы TARIFS с заданными ограничениями

Скрипт создания таблицы PRODUCT\_TYPE с заданными ограничениями представлен на листинге 4.

/\*DDL-скрипт создания таблицы PRODUCT\_TYPE с заданными ограничениями\*/

CREATE TABLE public."PRODUCT\_TYPE"

(

"ID" numeric(10) NOT NULL,

"NAME" character varying(100) NOT NULL,

"BEGIN\_DATE" date NOT NULL,

"END\_DATE" date,

"TARIF\_REF" numeric(10) NOT NULL,

PRIMARY KEY ("ID"),

FOREIGN KEY ("TARIF\_REF")

REFERENCES public."TARIFS" ("ID") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

WITH (

OIDS = FALSE

);

ALTER TABLE public."PRODUCT\_TYPE"

OWNER to postgres;

Листинг 4 - Скрипт создания таблицы PRODUCT\_TYPE с заданными ограничениями

Скрипт создания таблицы PRODUCTS с заданными ограничениями представлен на листинге 5.

/\*DDL-скрипт создания таблицы PRODUCTS с заданными ограничениями\*/

CREATE TABLE public."PRODUCTS"

(

"ID" numeric(10) NOT NULL,

"PRODUCT\_TYPE\_ID" numeric(10) NOT NULL,

"NAME" character varying(100) NOT NULL,

"CLIENT\_REF" numeric(10) NOT NULL,

"OPEN\_DATE" date NOT NULL,

"CLOSE\_DATE" date,

PRIMARY KEY ("ID"),

FOREIGN KEY ("PRODUCT\_TYPE\_ID")

REFERENCES public."PRODUCT\_TYPE" ("ID") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID,

FOREIGN KEY ("CLIENT\_REF")

REFERENCES public."CLIENTS" ("ID") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

WITH (

OIDS = FALSE

);

ALTER TABLE public."PRODUCTS"

OWNER to postgres;

ALTER TABLE public."PRODUCT\_TYPE"

OWNER to postgres;

Листинг 5 - Скрипт создания таблицы PRODUCTS с заданными ограничениями

Скрипт создания таблицы ACCOUNTS с заданными ограничениями представлен на листинге 6.

/\*DDL-скрипт создания таблицы ACCOUNTS с заданными ограничениями\*/

CREATE TABLE public."ACCOUNTS"

(

"ID" numeric(10) NOT NULL,

"NAME" character varying(100) NOT NULL,

"SALDO" numeric(10, 2) NOT NULL,

"CLIENT\_REF" numeric(10) NOT NULL,

"OPEN\_DATE" date NOT NULL,

"CLOSE\_DATE" date,

"PRODUCT\_REF" numeric(10) NOT NULL,

"ACC\_NUM" character varying(25) NOT NULL,

PRIMARY KEY ("ID"),

FOREIGN KEY ("CLIENT\_REF")

REFERENCES public."CLIENTS" ("ID") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID,

FOREIGN KEY ("PRODUCT\_REF")

REFERENCES public."PRODUCTS" ("ID") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

WITH (

OIDS = FALSE

);

ALTER TABLE public."ACCOUNTS"

OWNER to postgres;

ALTER TABLE public."PRODUCT\_TYPE"

OWNER to postgres;

Листинг 6 - Скрипт создания таблицы ACCOUNTS с заданными ограничениями

Скрипт создания таблицы RECORDS с заданными ограничениями представлен на листинге 7.

/\*DDL-скрипт создания таблицы RECORDS с заданными ограничениями\*/

CREATE TABLE public."RECORDS"

(

"ID" numeric(10) NOT NULL,

"DT" numeric(1) NOT NULL,

"SUM" numeric(10, 2) NOT NULL,

"ACC\_REF" numeric(10) NOT NULL,

"OPER\_DATE" date NOT NULL,

PRIMARY KEY ("ID"),

FOREIGN KEY ("ACC\_REF")

REFERENCES public."ACCOUNTS" ("ID") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

WITH (

OIDS = FALSE

);

ALTER TABLE public."RECORDS"

OWNER to postgres;

Листинг 7 - Скрипт создания таблицы RECORDS с заданными ограничениями

1. Заполните таблицы примерами из приложенного скрипта.

Результат запроса для заполнения таблиц БД представленными данными представлен на рисунке 1.

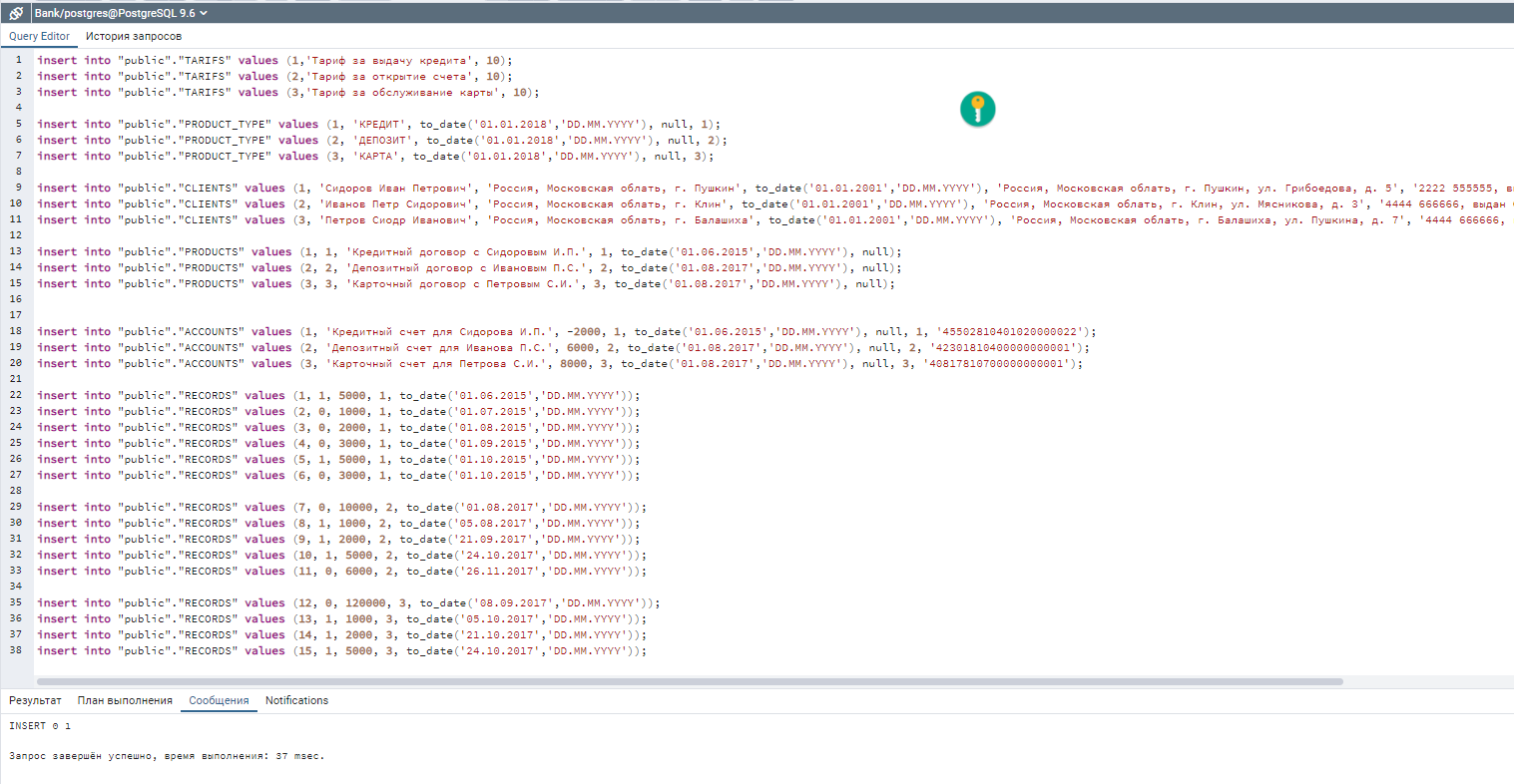


Рисунок 1 – Результат SQL-запроса для заполнения таблиц БД представленными данными

1. Подготовьте скрипты заполнения таблиц тестовыми данными, достаточными для выполнения заданий ниже.

Дополнительные скрипты для заполнения таблиц БД «bank» тестовыми данными для выполнения последующих задач представлены в листинге 8.

begin

insert into "public"."TARIFS" values (1,'Тариф за выдачу кредита', 10);

insert into "public"."TARIFS" values (2,'Тариф за открытие счета', 10);

insert into "public"."TARIFS" values (3,'Тариф за обслуживание карты', 10);

insert into "public"."PRODUCT\_TYPE" values (1, 'КРЕДИТ', to\_date('01.01.2018','DD.MM.YYYY'), null, 1);

insert into "public"."PRODUCT\_TYPE" values (2, 'ДЕПОЗИТ', to\_date('01.01.2018','DD.MM.YYYY'), null, 2);

insert into "public"."PRODUCT\_TYPE" values (3, 'КАРТА', to\_date('01.01.2018','DD.MM.YYYY'), null, 3);

insert into "public"."CLIENTS" values (1, 'Сидоров Иван Петрович', 'Россия, Московская облать, г. Пушкин', to\_date('01.01.2001','DD.MM.YYYY'), 'Россия, Московская облать, г. Пушкин, ул. Грибоедова, д. 5', '2222 555555, выдан ОВД г. Пушкин, 10.01.2015');

insert into "public"."CLIENTS" values (2, 'Иванов Петр Сидорович', 'Россия, Московская облать, г. Клин', to\_date('01.01.2001','DD.MM.YYYY'), 'Россия, Московская облать, г. Клин, ул. Мясникова, д. 3', '4444 666666, выдан ОВД г. Клин, 10.01.2015');

insert into "public"."CLIENTS" values (3, 'Петров Сиодр Иванович', 'Россия, Московская облать, г. Балашиха', to\_date('01.01.2001','DD.MM.YYYY'), 'Россия, Московская облать, г. Балашиха, ул. Пушкина, д. 7', '4444 666666, выдан ОВД г. Клин, 10.01.2015');

insert into "public"."PRODUCTS" values (1, 1, 'Кредитный договор с Сидоровым И.П.', 1, to\_date('01.06.2015','DD.MM.YYYY'), null);

insert into "public"."PRODUCTS" values (2, 2, 'Депозитный договор с Ивановым П.С.', 2, to\_date('01.08.2017','DD.MM.YYYY'), null);

insert into "public"."PRODUCTS" values (3, 3, 'Карточный договор с Петровым С.И.', 3, to\_date('01.08.2017','DD.MM.YYYY'), null);

insert into "public"."ACCOUNTS" values (1, 'Кредитный счет для Сидорова И.П.', -2000, 1, to\_date('01.06.2015','DD.MM.YYYY'), null, 1, '45502810401020000022');

insert into "public"."ACCOUNTS" values (2, 'Депозитный счет для Иванова П.С.', 6000, 2, to\_date('01.08.2017','DD.MM.YYYY'), null, 2, '42301810400000000001');

insert into "public"."ACCOUNTS" values (3, 'Карточный счет для Петрова С.И.', 8000, 3, to\_date('01.08.2017','DD.MM.YYYY'), null, 3, '40817810700000000001');

insert into "public"."RECORDS" values (1, 1, 5000, 1, to\_date('01.06.2015','DD.MM.YYYY'));

insert into "public"."RECORDS" values (2, 0, 1000, 1, to\_date('01.07.2015','DD.MM.YYYY'));

insert into "public"."RECORDS" values (3, 0, 2000, 1, to\_date('01.08.2015','DD.MM.YYYY'));

insert into "public"."RECORDS" values (4, 0, 3000, 1, to\_date('01.09.2015','DD.MM.YYYY'));

insert into "public"."RECORDS" values (5, 1, 5000, 1, to\_date('01.10.2015','DD.MM.YYYY'));

insert into "public"."RECORDS" values (6, 0, 3000, 1, to\_date('01.10.2015','DD.MM.YYYY'));

insert into "public"."RECORDS" values (7, 0, 10000, 2, to\_date('01.08.2017','DD.MM.YYYY'));

insert into "public"."RECORDS" values (8, 1, 1000, 2, to\_date('05.08.2017','DD.MM.YYYY'));

insert into "public"."RECORDS" values (9, 1, 2000, 2, to\_date('21.09.2017','DD.MM.YYYY'));

insert into "public"."RECORDS" values (10, 1, 5000, 2, to\_date('24.10.2017','DD.MM.YYYY'));

insert into "public"."RECORDS" values (11, 0, 6000, 2, to\_date('26.11.2017','DD.MM.YYYY'));

insert into "public"."RECORDS" values (12, 0, 120000, 3, to\_date('08.09.2017','DD.MM.YYYY'));

insert into "public"."RECORDS" values (13, 1, 1000, 3, to\_date('05.10.2017','DD.MM.YYYY'));

insert into "public"."RECORDS" values (14, 1, 2000, 3, to\_date('21.10.2017','DD.MM.YYYY'));

insert into "public"."RECORDS" values (15, 1, 5000, 3, to\_date('24.10.2017','DD.MM.YYYY'));

commit;

end;

Листинг 8 - Дополнительные скрипты для заполнения таблиц БД «bank» тестовыми данными

1. Сформируйте отчет, который содержит все счета, относящиеся к продуктам типа ДЕПОЗИТ, принадлежащих клиентам, у которых нет открытых продуктов типа КРЕДИТ.

Вместо отчета была сформирована выборка. SQL-запрос для задачи 4 продемонстрирован в листинге 9.

SELECT "ACCOUNTS"."NAME", "ACCOUNTS"."SALDO", "ACCOUNTS"."OPEN\_DATE", "ACCOUNTS"."CLOSE\_DATE", "ACCOUNTS"."ACC\_NUM"

FROM "public"."ACCOUNTS", "public"."PRODUCT\_TYPE", "public"."PRODUCTS", "public"."CLIENTS"

WHERE "PRODUCTS"."PRODUCT\_TYPE\_ID" = "PRODUCT\_TYPE"."ID"

AND "PRODUCTS"."CLIENT\_REF" = "CLIENTS"."ID"

AND "ACCOUNTS"."CLIENT\_REF" = "CLIENTS"."ID"

AND "PRODUCT\_TYPE"."NAME" = 'ДЕПОЗИТ'

AND "ACCOUNTS"."PRODUCT\_REF" = "PRODUCTS"."ID"

AND "PRODUCTS"."PRODUCT\_TYPE\_ID" = "PRODUCT\_TYPE"."ID"

AND "PRODUCT\_TYPE"."NAME" != 'КРЕДИТ'

Листинг 9 - SQL-запрос для задачи № 4

Результат запроса для задачи 4 представлен на рисунке 2.

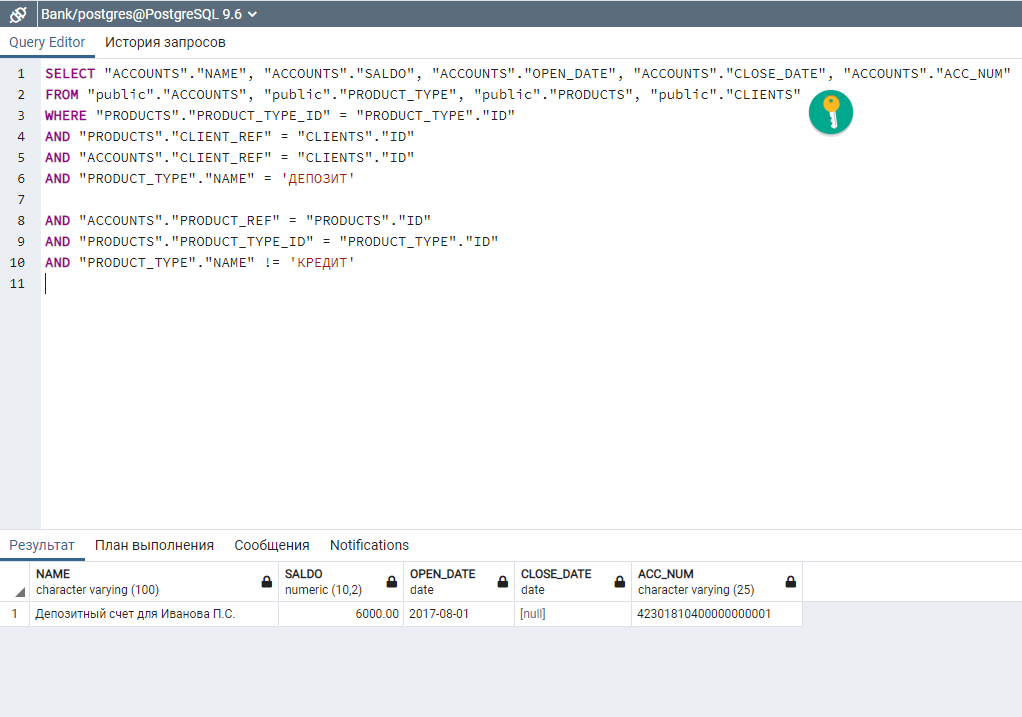


Рисунок 2 – Результат SQL-запроса для задачи № 4

1. Сформируйте выборку, который содержит средние движения по счетам в рамках одного дня, в разрезе типа продукта.

SQL-запрос выборки для задачи 5 продемонстрирован в листинге 10.

SELECT "PRODUCT\_TYPE"."NAME", ROUND(AVG("RECORDS"."SUM"), 2) AS MEDIUM\_SUMM, "RECORDS"."OPER\_DATE"

FROM "public"."RECORDS", "public"."PRODUCT\_TYPE", "public"."ACCOUNTS", "public"."PRODUCTS"

WHERE "RECORDS"."OPER\_DATE" = '2021-09-01'

AND "RECORDS"."ACC\_REF" = "ACCOUNTS"."ID"

AND "ACCOUNTS"."PRODUCT\_REF" = "PRODUCTS"."ID"

AND "PRODUCTS"."PRODUCT\_TYPE\_ID" = "PRODUCT\_TYPE"."ID"

GROUP BY "PRODUCT\_TYPE"."NAME", "RECORDS"."OPER\_DATE"

Листинг 10 - SQL-запрос для задачи № 5

Результат запроса для задачи 5 представлен на рисунке 3.

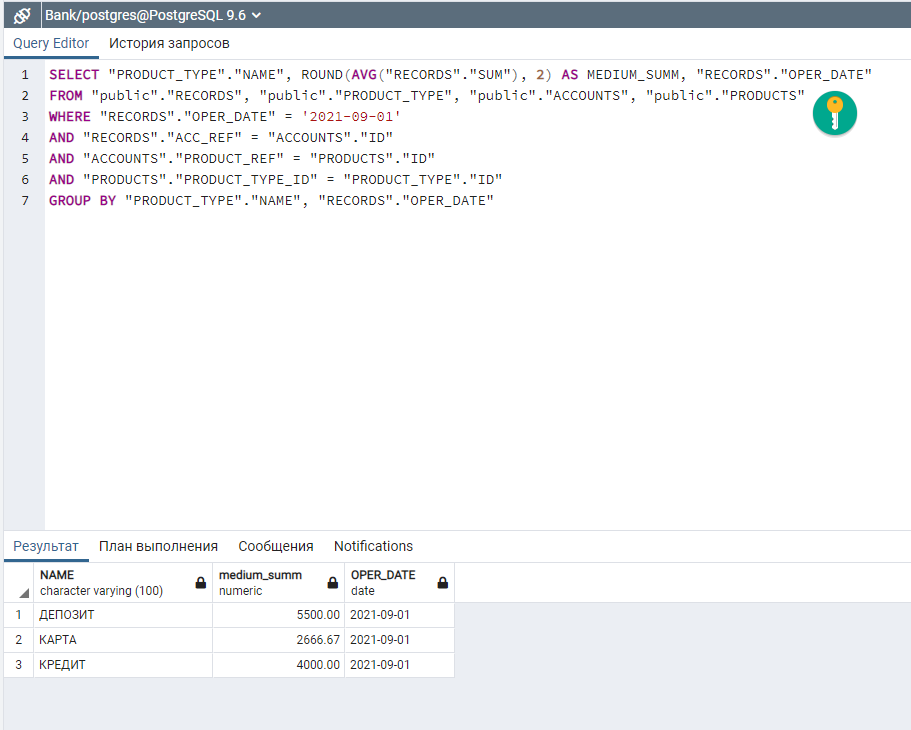


Рисунок 3 – Результат SQL-запроса для задачи № 5

1. Сформируйте выборку, в который попадут клиенты, у которых были операции по счетам за прошедший месяц от текущей даты. Выведите клиента и сумму операций за день в разрезе даты.

SQL-запрос выборки для задачи 6 продемонстрирован в листинге 11.

SELECT "RECORDS"."OPER\_DATE", SUM("RECORDS"."SUM") AS SUM\_OPERATION, "CLIENTS"."NAME"

FROM "public"."RECORDS", "public"."CLIENTS", "public"."ACCOUNTS"

WHERE "RECORDS"."ACC\_REF" = "ACCOUNTS"."ID"

AND "ACCOUNTS"."CLIENT\_REF" = "CLIENTS"."ID"

AND "RECORDS"."OPER\_DATE"::date BETWEEN now()::date-interval '1 month' AND now()::date

GROUP BY "RECORDS"."OPER\_DATE", "CLIENTS"."NAME"

Листинг 11 - SQL-запрос для задачи № 6

Результат запроса для задачи 6 представлен на рисунке 4.

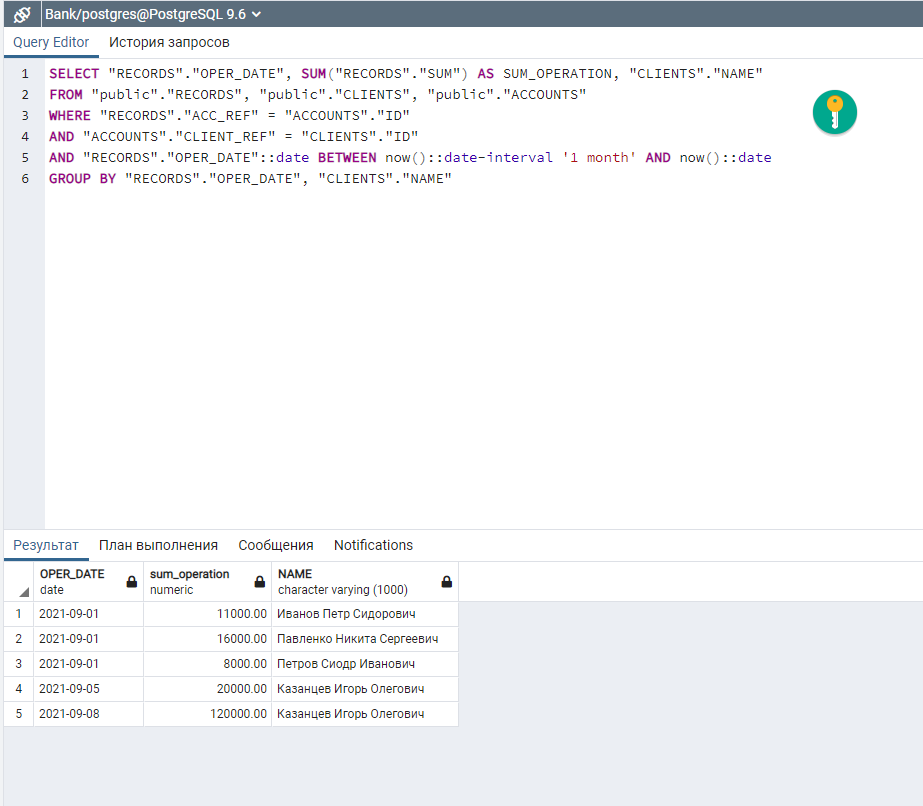


Рисунок 4 – Результат SQL-запроса для задачи № 6

1. В результате сбоя в базе данных разъехалась информация между остатками и операциями по счетам. Напишите нормализацию (процедуру выравнивающую данные), которая найдет такие счета и восстановит остатки по счету.

Код процедуры, которая предназначена для выравнивания данных (восстановления данных по счету), представлен в листинге 12.

CREATE OR REPLACE PROCEDURE Normalization\_procedure() LANGUAGE plpgsql AS $$

DECLARE

--переменные

genre\_rec RECORD;

counter integer; --счетчик

summ integer; --сумма

begin

summ := 0;

counter := 1;

for genre\_rec in (SELECT \* FROM "public"."RECORDS" order by "ID")

LOOP

--RAISE NOTICE 'Genre Id is : % , Sum is : %, ACC\_REF : %, genre\_rec : %', genre\_rec."ID",genre\_rec."SUM", genre\_rec."ACC\_REF", genre\_rec;

if(genre\_rec."ACC\_REF" = counter)

then

if(genre\_rec."DT" = 0)

then

summ := summ + genre\_rec."SUM";

end if;

if(genre\_rec."DT" = 1)

then

summ := summ - genre\_rec."SUM";

end if;

end if;

if(genre\_rec."ACC\_REF" != counter AND genre\_rec."DT" = 0)

then

--update ACCOUNTS

update "public"."ACCOUNTS"

set "SALDO" = summ

where "ACCOUNTS"."ID" = counter;

counter := counter + 1;

summ := 0 + genre\_rec."SUM";

end if;

if(genre\_rec."ACC\_REF" != counter AND genre\_rec."DT" = 1)

then

--update ACCOUNTS

update "public"."ACCOUNTS"

set "SALDO" = summ

where "ACCOUNTS"."ID" = counter;

counter := counter + 1;

summ := 0 - genre\_rec."SUM";

end if;

--update ACCOUNTS

update "public"."ACCOUNTS"

set "SALDO" = summ

where "ACCOUNTS"."ID" = counter;

--RAISE NOTICE 'Summ : %',summ;

END LOOP;

end;$$

Листинг 12 – Код процедуры выравнивания данных (восстановления данных по счету)

Результат работы данной процедуры представлен на рисунке 5.

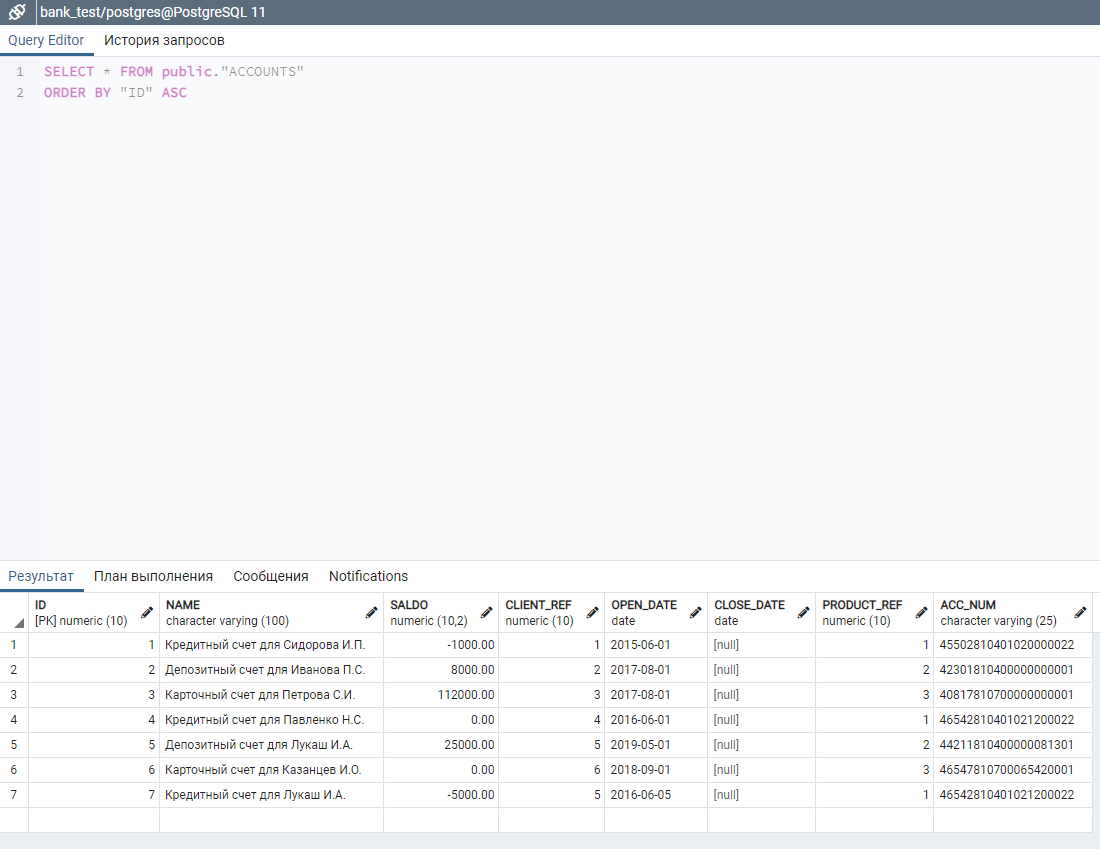


Рисунок 5 – Результат работы процедуры выравнивания данных по счетам

1. Сформируйте выборку, который содержит информацию о клиентах, которые полностью погасили кредит, но при этом не закрыли продукт.

SQL-запрос данной выборки представлен в листинге 13.

SELECT "CLIENTS"."NAME" AS "Имя клиента", "CLIENTS"."PLACE\_OF\_BIRTH" AS "Место рождения", "CLIENTS"."DATE\_OF\_BIRTH" AS "Дата клиента",

"CLIENTS"."ADDRESS" AS "Адрес клиента", "CLIENTS"."PASSPORT", "ACCOUNTS"."SALDO" AS "Остаток по счету", "PRODUCTS"."NAME" AS "Продукт клиента",

"PRODUCTS"."CLOSE\_DATE" AS "Дата закрытия продукта"

FROM "public"."CLIENTS", "public"."ACCOUNTS", "public"."PRODUCTS"

WHERE "ACCOUNTS"."CLIENT\_REF" = "CLIENTS"."ID"

AND "PRODUCTS"."CLIENT\_REF" = "CLIENTS"."ID"

AND "ACCOUNTS"."PRODUCT\_REF" = 1

AND "PRODUCTS"."CLOSE\_DATE" IS NULL

AND "ACCOUNTS"."SALDO" >= 0;

Листинг 13 - SQL-запрос для задачи № 8

Результат данного запроса представлен на рисунке 6.



Рисунок 6 – Результат SQL-запроса для задачи № 8

1. Закройте продукты (установите дату закрытия равную текущей) типа КРЕДИТ, у которых произошло полное погашение, но при этом не было повторной выдачи.

Для решения задачи №9 была написана процедура «close\_products». Код данной процедуры представлен в листинге 14.

CREATE OR REPLACE PROCEDURE close\_products() LANGUAGE plpgsql AS $$

begin

if((SELECT COUNT(\*) FROM "public"."PRODUCTS", "public"."CLIENTS" WHERE "PRODUCTS"."PRODUCT\_TYPE\_ID" = 1

AND "PRODUCTS"."CLIENT\_REF" = "CLIENTS"."ID" ) = 1)

then

UPDATE "public"."PRODUCTS"

SET "CLOSE\_DATE" = current\_date

FROM "public"."ACCOUNTS", "public"."CLIENTS"

WHERE "ACCOUNTS"."CLIENT\_REF" = "CLIENTS"."ID"

AND "PRODUCTS"."CLIENT\_REF" = "CLIENTS"."ID"

AND "PRODUCTS"."CLOSE\_DATE" IS NULL

AND "ACCOUNTS"."SALDO" >= 0;

end if;

--RAISE NOTICE 'Summ : %',summ;

end;$$

Листинг 14 – Код процедуры «close\_products» для задачи № 9

Для отображения работы данной процедуры был сделан скрин таблицы «PRODUCTS» до вызова процедуры. Данный скрин представлен на рисунке 7.

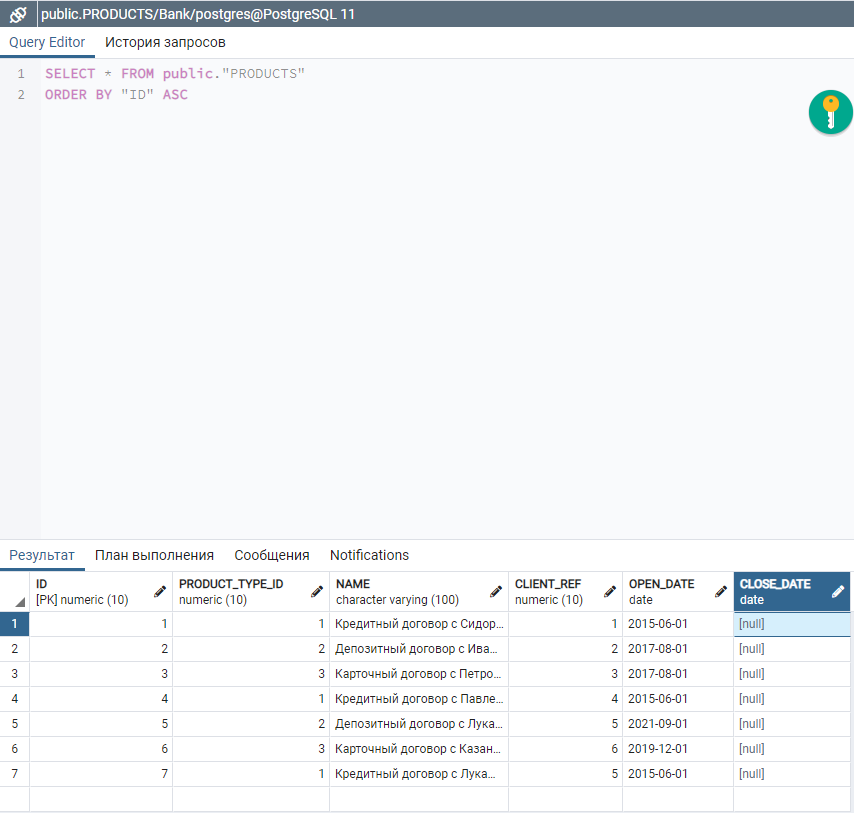


Рисунок 7 – Таблица «PRODUCTS» до вызова процедуры «close\_products»

Также был сделан скрин, после вызова процедуры «close\_products». Данный скрин представлен на рисунке 8.

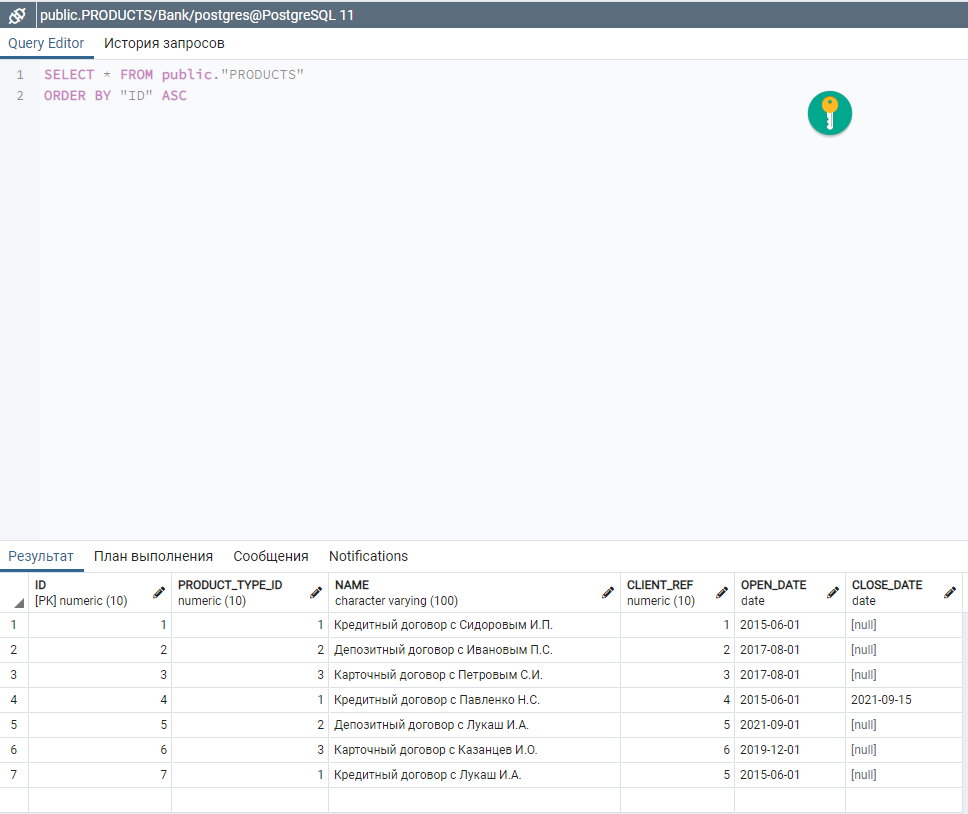


Рисунок 8 – Таблица «PRODUCTS» после вызова процедуры «close\_products»

1. Закройте возможность открытия (установите дату окончания действия) для типов продуктов, по счетам продуктов которых, не было движений более одного месяца.

Для решения задачи №10 была написана процедура «close\_products\_open». Код данной процедуры представлен в листинге 15.

CREATE OR REPLACE PROCEDURE close\_products\_open() LANGUAGE plpgsql AS $$

DECLARE

--переменные

genre\_rec numeric; --переменная цикла

begin

for genre\_rec in (SELECT DISTINCT "PRODUCT\_TYPE"."ID" FROM "public"."PRODUCT\_TYPE"

EXCEPT

SELECT DISTINCT "PRODUCT\_TYPE"."ID"--, "RECORDS"."OPER\_DATE", "CLIENTS"."NAME"

FROM "public"."RECORDS", "public"."CLIENTS", "public"."ACCOUNTS", "public"."PRODUCT\_TYPE", "public"."PRODUCTS"

WHERE "RECORDS"."ACC\_REF" = "ACCOUNTS"."ID"

AND "ACCOUNTS"."CLIENT\_REF" = "CLIENTS"."ID"

AND "ACCOUNTS"."PRODUCT\_REF" = "PRODUCTS"."ID"

AND "PRODUCTS"."PRODUCT\_TYPE\_ID" = "PRODUCT\_TYPE"."ID"

AND "RECORDS"."OPER\_DATE"::date BETWEEN now()::date-interval '1 month' AND now()::date)

LOOP

UPDATE "public"."PRODUCT\_TYPE"

SET "END\_DATE" = current\_date

FROM "public"."RECORDS", "public"."CLIENTS", "public"."ACCOUNTS", "public"."PRODUCTS"

WHERE

"PRODUCT\_TYPE"."ID" = genre\_rec;

--RAISE NOTICE 'id\_product\_type : %',genre\_rec;

END LOOP;

end;$$

Листинг 15 – Код процедуры «close\_products\_open» для задачи № 10

Для отображения работы данной процедуры был сделан скрин таблицы «PRODUCT\_TYPE» до вызова процедуры. Данный скрин представлен на рисунке 9.

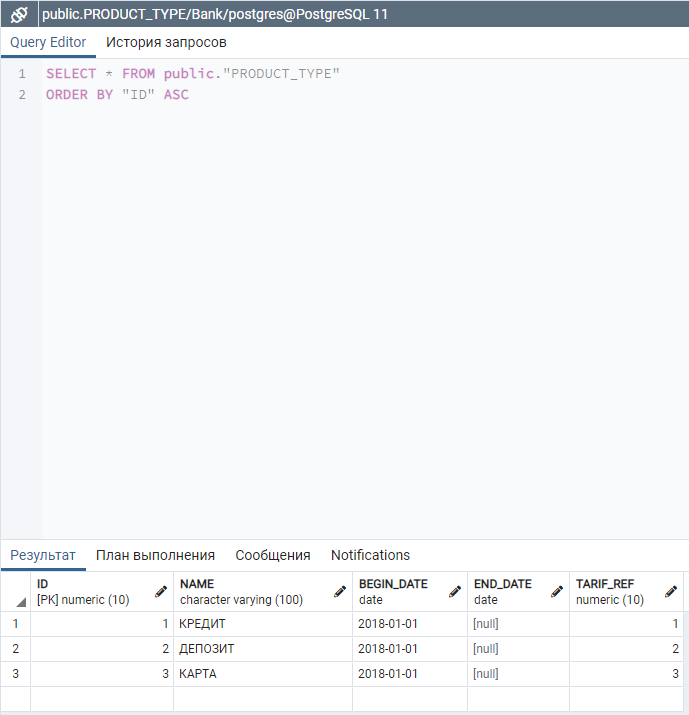


Рисунок 9 – Таблица «PRODUCT\_TYPE» до вызова процедуры «close\_products\_open»

Также был сделан скрин, после вызова процедуры «close\_products\_open». Данный скрин представлен на рисунке 10.

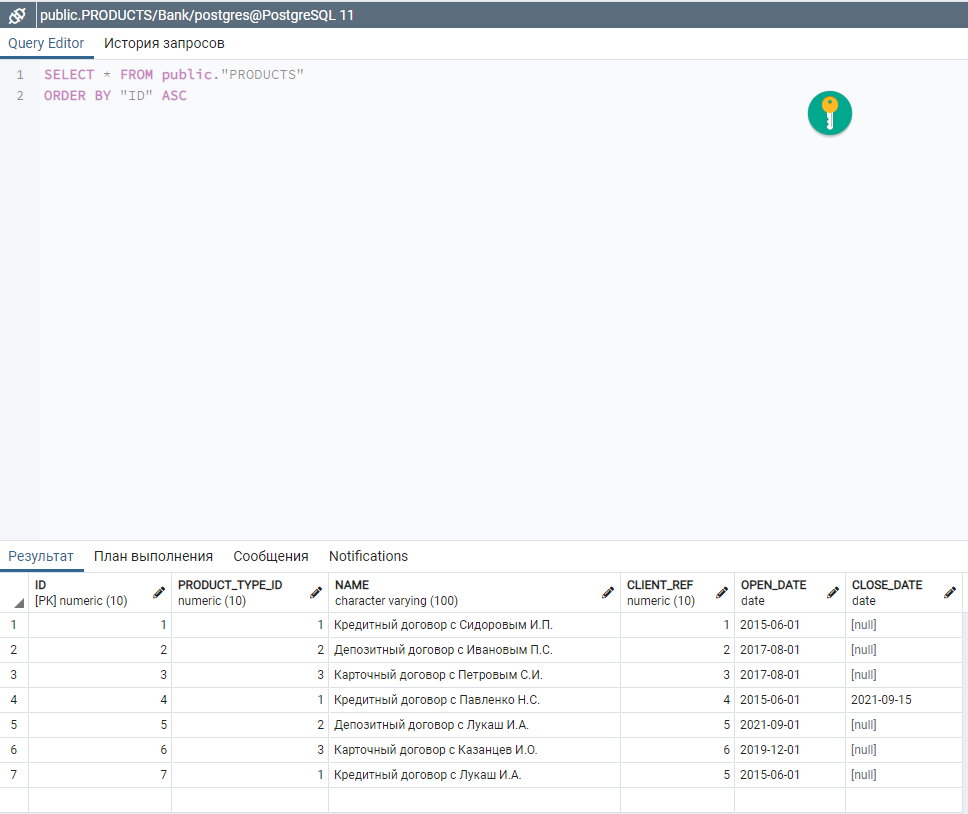


Рисунок 10 – Таблица «PRODUCT\_TYPE» после вызова процедуры «close\_products\_open»

1. В модель данных добавьте сумму договора по продукту. Заполните поле для всех продуктов суммой максимальной дебетовой операции по счету для продукта типа КРЕДИТ, и суммой максимальной кредитовой операции по счету продукта для продукта типа ДЕПОЗИТ или КАРТА.

В ходе выполнения задания номер 11 в таблицу «PRODUCT\_TYPE» был добавлен столбец «MAX\_SUM\_PRODUCT», который был заполнен данными типа numeric. Заполненные данные представляли из себя максимально возможную финансовую операцию (дебит, кредит) по продукту.

Код добавления столбца «MAX\_SUM\_PRODUCT» в таблицу «PRODUCT\_TYPE» представлен в листинге 16.

ALTER TABLE public."PRODUCT\_TYPE"

ADD COLUMN "MAX\_SUM\_PRODUCT" numeric(15, 2) NOT NULL;

Листинг 15 – Код добавления столбца «MAX\_SUM\_PRODUCT» в таблицу «PRODUCT\_TYPE»

Для отображения работы был сделан скрин таблицы «PRODUCT\_TYPE» до SQL-запроса на добавление столбца. Данный скрин представлен на рисунке 11.

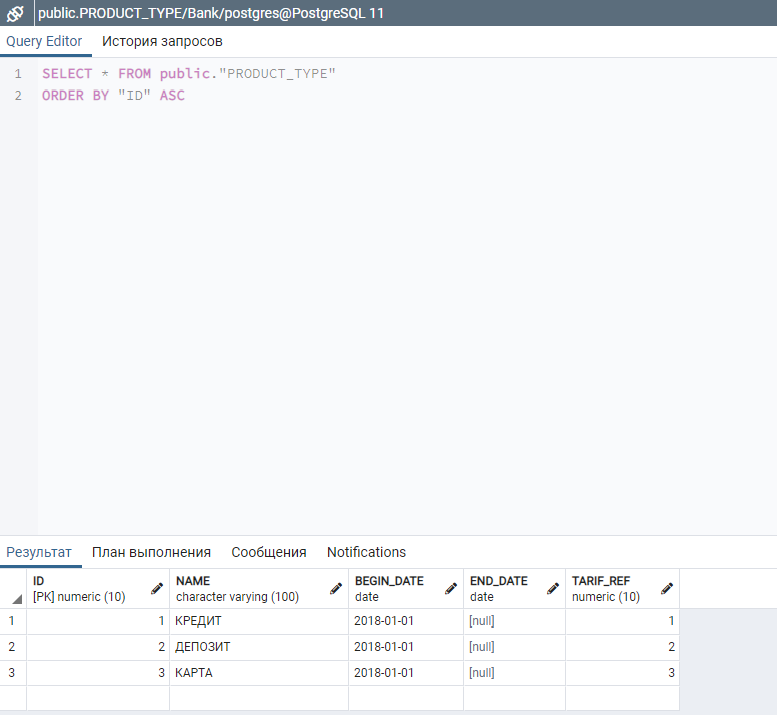


Рисунок 11 – Таблица «PRODUCT\_TYPE» до SQL-запроса на добавление столбца

Также был сделан скрин, после применения кода SQL-запроса на добавление столбца и заполнения его данными. Данный скрин представлен на рисунке 12.

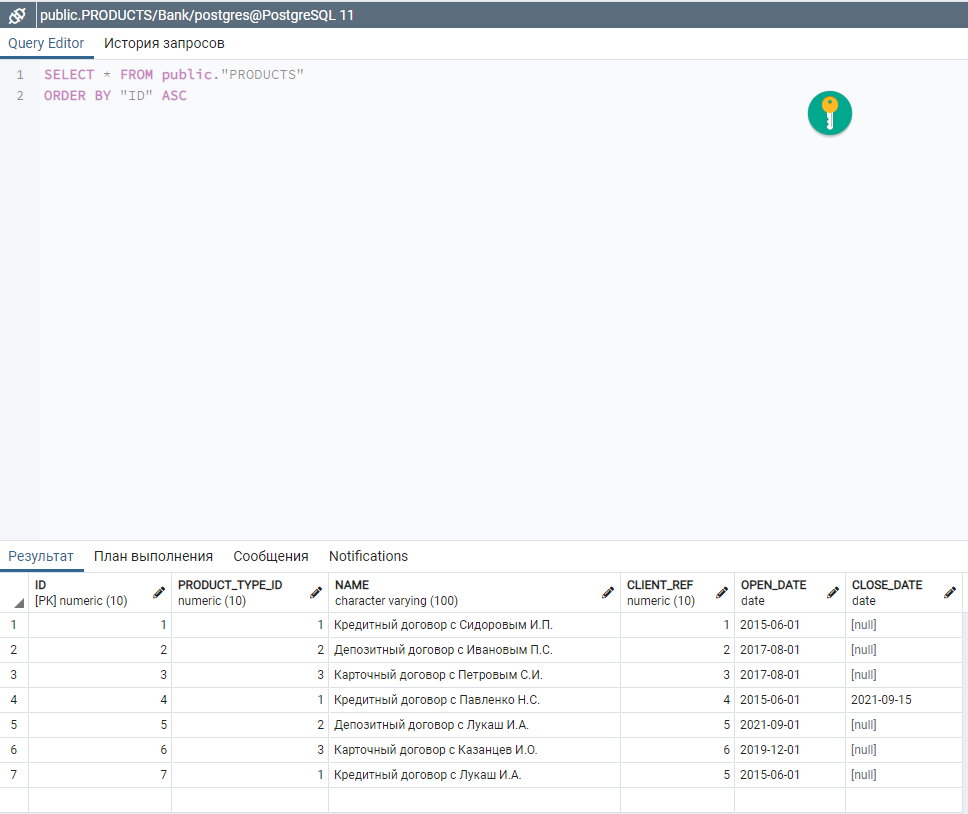


Рисунок 12 – Таблица «PRODUCT\_TYPE» после SQL-запроса на добавление столбца и заполнения его данными

**Вывод:** в ходе выполнения тестового задания от «ШИФТ» по теме «разработка ЦФТ-Банк» были закреплены навыки работы с СУБД PostgreSQL, ER-диаграммами, SQL-запросами. Был изучен принцип написания и применения процедур в PostgreSQL.