Задание 1. Создание таблицы и настройка RLS

Step 1: Создать тестовую таблицу, назовите её **skillbox_test**. Таблица должна содержать поля: «Идентификатор», «Логин», «Размер заработной платы», «Контактный номер телефона».

Step 2: Создать тестовые учётные записи пользователей БД.
Для сотрудников отдела сбыта:

Семёнов Вадим (login SEMIONOV)

Данилов Иван (login DANILOV)

Для бухгалтерии:

Романова Светлана (login ROMANOVA)

Step 3: Заполнить таблицу следующими данными:
1, SEMIONOV, 50000, 791101234567
2, DANILOV, 90000, 795301234567
3, ROMANOVA, 70000, 790401234567

```
postgres=# create database test_db
postgres-# ;
CREATE DATABASE
postgres=# create table test_skillbox;
ERROR:  syntax error at or near ";"
LINE 1: create table test_skillbox;
                                  ^
postgres=# create table test_skillbox (identifier integer, username text, salary integer, phone number bigint);
ERROR:  syntax error at or near "bigint"
LINE 1: ...nteger, username text, salary integer, phone number bigint);
                                                              ^
postgres=# create table test_skillbox (identifier integer, username text, salary integer, phone_number bigint);
CREATE TABLE
postgres=# insert into test_skillbox values ('1', 'Semionov', '50000', '791101234567');
INSERT 0 1
postgres=# insert into test_skillbox values ('2', 'Danilov', '90000', '795301234567');
INSERT 0 1
postgres=# insert into test_skillbox values ('3', 'Romanova', '70000', '790401234567');
INSERT 0 1
postgres=# select * from test_skillbox;
 identifier | username | salary | phone_number
------------+----------+--------+--------------
          1 | Semionov |  50000 | 791101234567
          2 | Danilov  |  90000 | 795301234567
          3 | Romanova |  70000 | 790401234567
(3 rows)

postgres=# grant all on table test_skillbox to semionov, danilov, romanova;
ERROR:  role "semionov" does not exist
postgres=# create user semionol nosuperuser, login;
ERROR:  syntax error at or near ","
LINE 1: create user semionol nosuperuser, login;
                                        ^
postgres=# create user semionol nosuperuser login;
CREATE ROLE
postgres=# create user danilov nosuperuser login;
CREATE ROLE
postgres=# create user romanova nosuperuser login;
CREATE ROLE
postgres=# grant all on table test_skillbox to semionov, danilov, romanova;
ERROR:  role "semionov" does not exist
postgres=# alter role semionol
BYPASSRLS           NOCREATEDB          REPLICATION
CONNECTION LIMIT    NOCREATEROLE        RESET
CREATEDB            NOINHERIT           SET
CREATEROLE          NOLOGIN             SUPERUSER
ENCRYPTED PASSWORD  NOREPLICATION       VALID UNTIL
INHERIT             NOSUPERUSER         WITH
```

Тут 3 в 1 — создание пользователей, таблицы и её заполнение.

Step 4: Настроить правила RLS.
Для сотрудников отдела сбыта: видят только свои личные оклады (одну запись со своим логином).
Для сотрудника бухгалтерии: видит все данные.

```
test_db=# CREATE FUNCTION read_me(text) RETURNS boolean as
test_db-# $$
test_db$# BEGIN
test_db$#     RAISE NOICE 'called as sessio_user=%, current_user=% for "%" ',
test_db$#         session_user, current_user, $1;
test_db$#     RETURN true;
test_db$# END;
test_db$# $$ LANGUAGE 'plpgsql';
ERROR:  unrecognized exception condition "noice"
CONTEXT:  compilation of PL/pgSQL function "read_me" near line 3
test_db=# CREATE FUNCTION read_me(text) RETURNS boolean as
$$
BEGIN
    RAISE NOTICE 'called as sessio_user=%, current_user=% for "%" ',
        session_user, current_user, $1;
    RETURN true;
END;
$$ LANGUAGE 'plpgsql';
CREATE FUNCTION
test_db=# GRANT ALL ON FUNCTION read_me TO romanova, semionov, danilov;
GRANT
test_db=#
```

Не знаю, зачем добавил данную функцию.

```
test_db=# CREATE POLICY sem_info on test_skillbox
    for select
    to semionov
    using (read_me(Username) and status = 'Employee 1');
CREATE POLICY
test_db=# CREATE POLICY dan_info on test_skillbox
    for select
    to danilov
    using (read_me(Username) and status = 'Employee 2');
CREATE POLICY
test_db=#
```

Тут я создал 2 политики, где 2 данных пользователя могут видеть только их соответствующие ряды.

```
test_db=# CREATE POLICY admin_all on test_skillbox to romanova using (true) with check (true);
CREATE POLICY
test_db=#
```

Тут политика, где данный пользователь может видеть всю таблицу.

```
test_db=# alter table test_skillbox enable row level security;
ALTER TABLE
test_db=#
```

И самое главное, включить безопасность на уровне ряда.

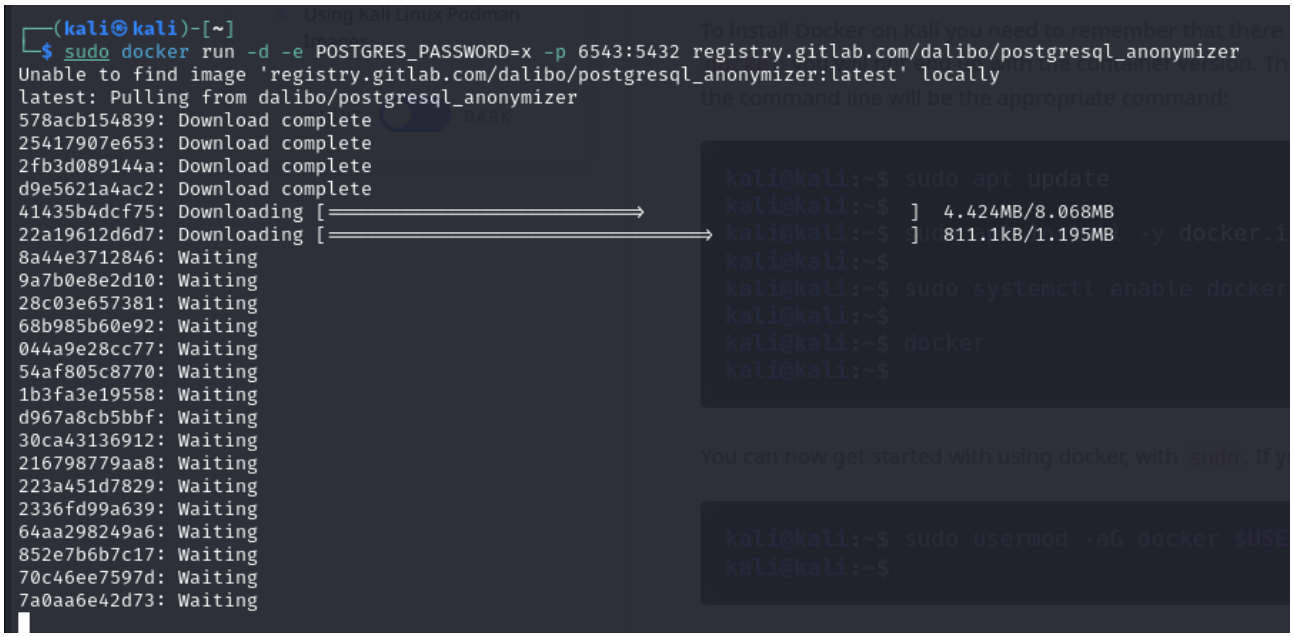
```
ERROR:  must be owner of table test_skillbox
test_db=> \c test_db postgres
You are now connected to database "test_db" as user "postgres".
test_db=# alter table test_skillbox enable row level security;
ALTER TABLE
test_db=# \c test_db romanova
You are now connected to database "test_db" as user "romanova".
test_db=> select * from test_skillbox;
   status   | username | salary | phone_number
------------+----------+--------+--------------
 Employee 1 | semionov |  50000 | 791101234567
 Employee 2 | danilov  |  90000 | 795301234567
 Accounter  | romanova |  70000 | 790401234567
(3 rows)

test_db=> \c test_db semionov
You are now connected to database "test_db" as user "semionov".
test_db=> select * from test_skillbox;
NOTICE:  called as sessio_user=semionov, current_user=semionov for "semionov"
   status   | username | salary | phone_number
------------+----------+--------+--------------
 Employee 1 | semionov |  50000 | 791101234567
(1 row)

test_db=> \c test_db danilov
You are now connected to database "test_db" as user "danilov".
test_db=> select * from test_skillbox;
NOTICE:  called as sessio_user=danilov, current_user=danilov for "danilov"
   status   | username | salary | phone_number
------------+----------+--------+--------------
 Employee 2 | danilov  |  90000 | 795301234567
(1 row)

test_db=>
```

Итоговый результат.

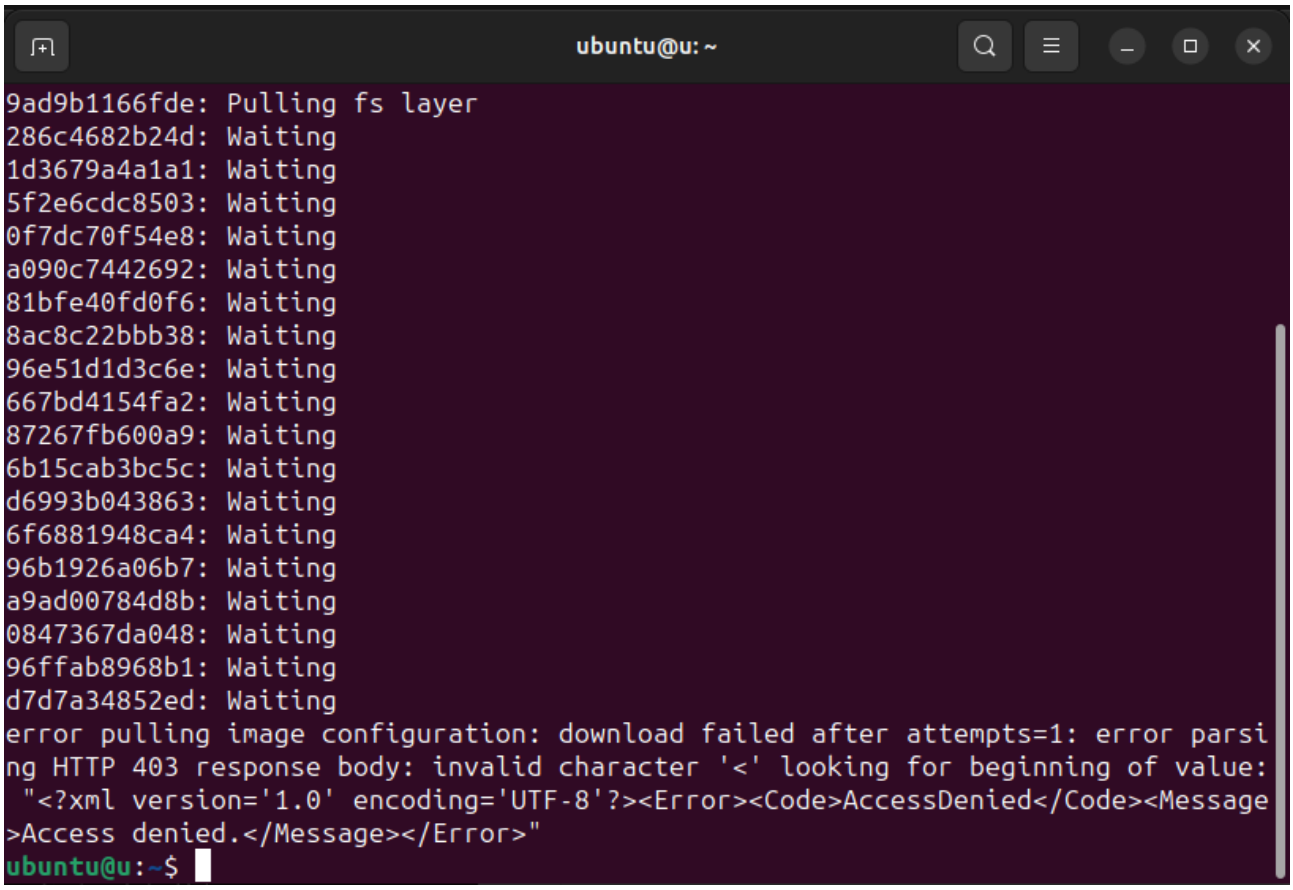
P.S. Хотя в задании было указано так - «Создать SQL-скрипт, выполнив описанные ниже действия». Но я решил сделать по своему.

Задание 2. Настройка маскирования

Step 1: Подключить расширение PostgreSQL Anonymizer.

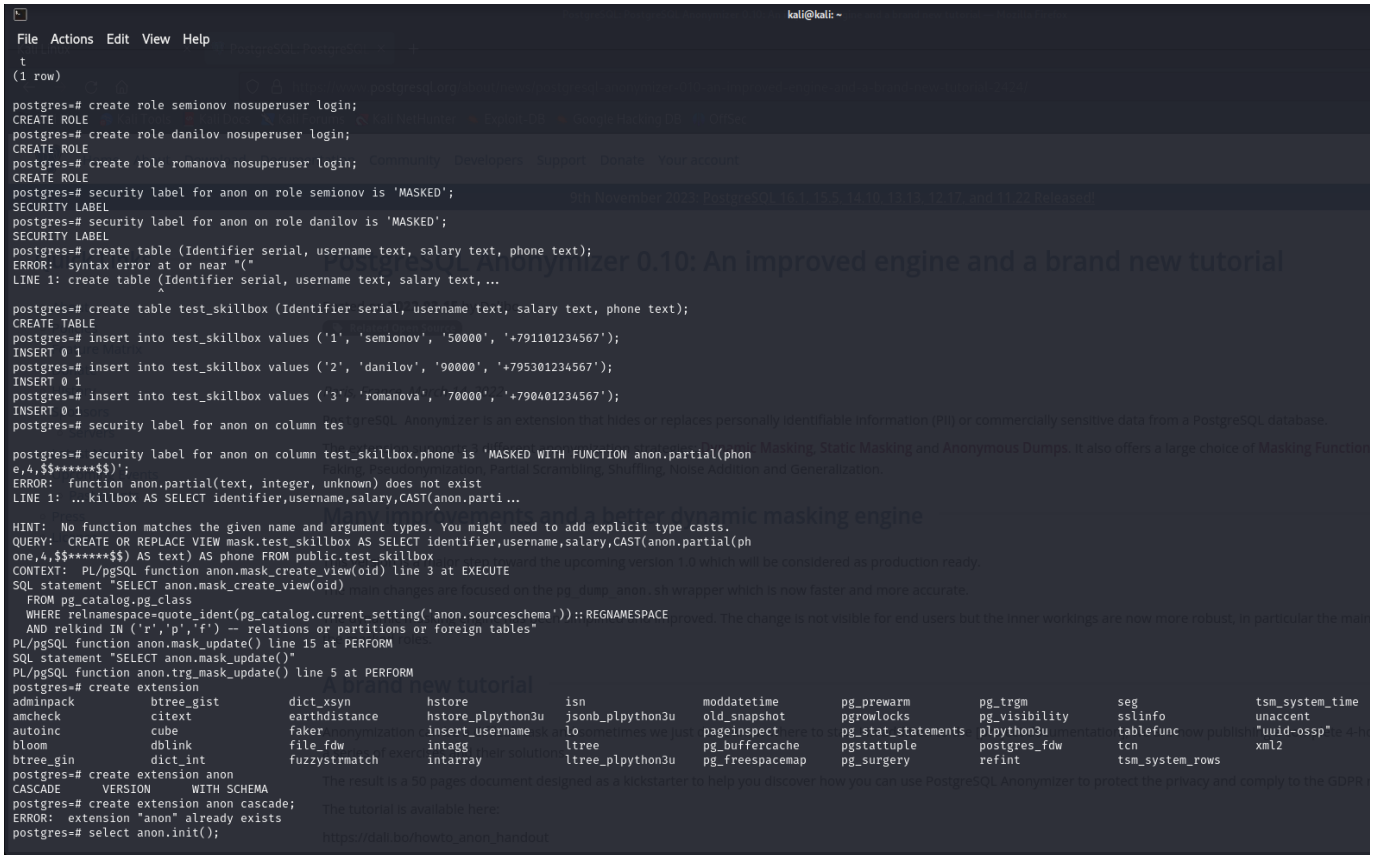


Установка данной утилиты (вот почему я это делаю на Kali Linux, а не на Ubuntu).

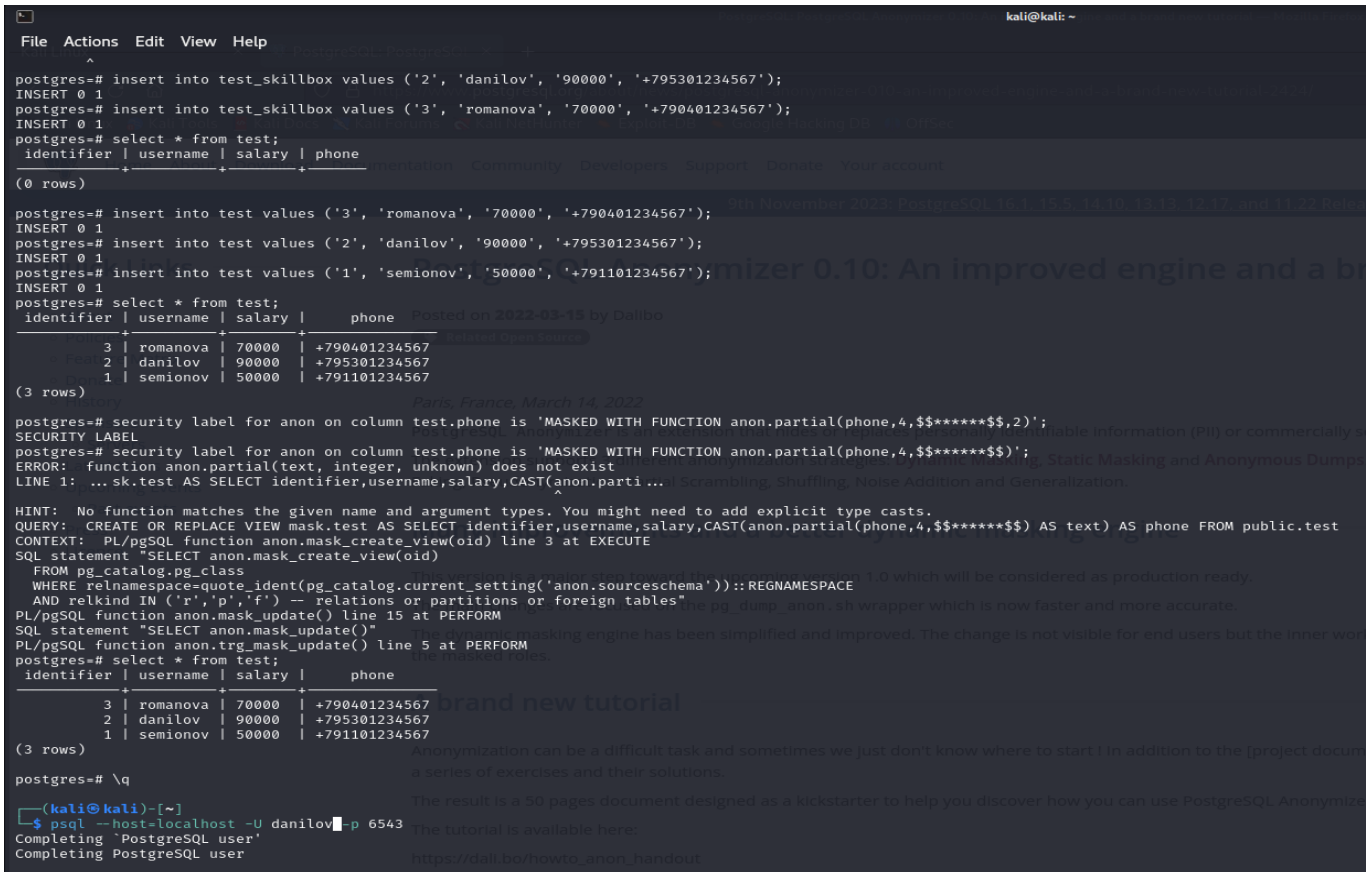


Ошибка доступа (или статус 403). Сделав то же самое на Kali Linux, ошибки не было.

Step 2: Настроить правила маскирования на таблицу **skillbox_test** (из задания 1) так, чтобы сотрудникам отдела сбыта в поле «Телефон» показывались бы только четыре символа слева, остальные символы выводились бы символом «*», сотрудники бухгалтерии должны видеть данные без маскирования (без назначения роли superuser).



Мне пришлось создавать ту ДБ заново, поскольку фактически это другой инстанс (ту ДБ делал в самом PostgreSQL, а утилита была установлена и использована через Docker).



Фактически повторение первых 3 шагов из задания 1 — самая первая команда была «select anon.start_dynamic_masking();». Там была не одна попытка написать данную функцию про номер телефона, но вылезала ошибка — получилось пока так (phone,4,$$****$$,2), хотя в задании указано только 4 символа слева.

Пользователь Romanova видит все данные, а Semionov и Danilov видят только часть своих номеров телефона (если что, RLS там не включён).

P.S. Хотя в задании было указано так - «Создать SQL-скрипт, выполнив описанные ниже действия». Но я решил сделать по своему.