Lab 13 Instructions

Make students change their password

```
alter user user_name password 'new_password';
```

(it answers "alter role" ...). Syntax is slightly different with other products but it's always *alter user*.

Locking demo

Session #1:

```
start transaction;
delete from credits where movieid= id1;
```

Session #2:

```
start transaction;
delete from credits where movieid= id2; -- So far, OK
delete from credits where movieid= id1; -- Hangs
```

Session #1:

```
rollback; -- or commit, any ends the transaction
```

=> Unblocks session #2

Query attached for showing locks. The locked transactions tries to get a lock on the other transaction, this is where it waits. Note that pg_stat_activity (alias a in the query) has a column called "query" allowing to see the query.

Student tasks

1) Give the privilege to delete from YOUR credits table to somebody else

```
grant usage on schema sxxxxxxx to other_guy; -- important
grant select, delete on credits to other_guy;
```

2) Repeat with the other person what has been shown in the demo (will need to prefix the table name with the schema name)

3) What happens if, in a TX, session 1 deletes rows for credits for movie id1, at the same time session 2 deletes rows for movie id2, then session 1 tries to delete for id2 and session 2 tries to delete for id1?

4) Allow somebody else to see title and country in table movies but not the year or the id

```
create view vmovie as select distinct title, country from movies;.
```

```
grant select on vmovie to other_guy;
```

Is it updatable? (answer is no, because of distinct)

5) Take two people. Create ONE view allowing one to see only European films, and the other one only Asian films. Updatable? (answer is yes)

Cleanest way to do it:

```
create table user_scope(username varchar(30) not null,
                        continent varchar(20),
                        primary key (username, continent));
insert into user_scope(username, continent)
values(guy_1, 'EUROPE');
insert into user_scope(username, continent)
values(guy_2, 'ASIA');
create view scoped_movies
as select * from movies
where country in
(select c.country_code
 from countries c
      join user_scope s
        on s.continent = c.continent
       and s.username = current_user);
```

Note: `with check option` not implemented in PostgreSQL

It's also possible to have a

```
case current_user
  when ...
  when ...
  else ...
end
```

in the view, but the view has to be rebuilt if we change something.

Note that for the creator of the view it will return nothing.

6) Try to get into other's account

Hint: find usernames in pg_user.

Tell them:

1. Never do that in a company without having the task of testing security.
2. Some products (Oracle) lock an account after three wrong attempts, so it can mess up a lot of things.

I have written a small Java program that does it, you can give it to students, if you run this it tries the password

```
java -cp .:... PGConnect host_ip dbname user_account password
```

If you run this it looks for `password_list.txt` and tries all passwords (one per line) from it (many lists of common passwords available from the web)

```
java -cp .:... PGConnect host_ip dbname user_account
```

Of course the command can be easily generated by querying pg_user. You can even try more privileged users first.