2. Задание по основам баз данных

1. Установка PostgreSQL:

docker run --name some-postgres -e postgres\_password=mysecretpassword -d postgres – это команда для установки докер контейнера с postgres

psql -U postgres - эта команда для того, чтобы зайти в postgres в докер контейнере под юзером postgres

2. Создание БД:

create database academy;

вход в БД: \c academy;

3. Создание таблиц:

create table students (s\_id serial primary key, name varchar(100) not null, start\_year int not null);

create table courses (c\_no serial primary key, title varchar(100) not null, hours int);

create table exams (s\_id int not null, c\_no int not null, score int check (score > 0 and score <= 5), foreign key (s\_id) references students(s\_id) on delete cascade, foreign key (c\_no) references courses(c\_no) on delete cascade, primary key (s\_id, c\_no));

4. Добавление записей в таблицы:

insert into students (name, start\_year) values ('Ivan Ivanov', 2019), ('Petr Petrov', 2022), ('Olga Olegova', 2020);

insert into courses (title, hours) values ('Math', 50), ('Biology', 30), ('History', 20);

insert into exams (s\_id, c\_no, score) values (2, 1, 3), (2, 2, 5), (3, 3, 4), (3, 1, 5);

5. Запрос, который возвращает всех студентов, которые еще не сдали ни одного экзамена:

select students.name from students left join exams on students.s\_id = exams.s\_id where exams.s\_id is null;

6. Запрос, который возвращает список студентов и количество сданных им экзаменов:

select students.name, count(exams.\*) as exams from students join exams on students.s\_id = exams.s\_id group by students.name;

7. Запрос, который выводит список курсов со средним баллом по экзамену и сортирует его по убыванию среднего балла:

select courses.title, avg(exams.score) as avg\_score from courses join exams on courses.c\_no = exams.c\_no group by courses.title order by avg\_score desc;

8. Скрипт, который наполняет таблицы произвольными данными (psql):

do $$

declare

i int;

random\_name text;

random\_start\_year int;

begin

for i in 1..5 loop

random\_name := 'student ' || i;

random\_start\_year := 2010 + (i % 15);

insert into students (name, start\_year)

values (random\_name, random\_start\_year);

end loop;

end $$;

do $$

declare

courses text[] := array['math', 'biology', 'history', 'programming', 'physics'];

course\_title text;

course\_hours int;

i int;

begin

for i in 1..3 loop

course\_title := courses[i];

course\_hours := 20 + (i \* 10);

insert into courses (title, hours)

values (course\_title, course\_hours);

end loop;

end $$;

do $$

declare

student\_id int;

course\_id int;

score int;

i int := 0;

begin

while i < 10 loop

student\_id := (select s\_id from students order by random() limit 1);

course\_id := (select c\_no from courses order by random() limit 1);

score := floor(random() \* 5 + 1)::int;

if not exists (

select 1 from exams where s\_id = student\_id and c\_no = course\_id)

then

insert into exams (s\_id, c\_no, score)

values (student\_id, course\_id, score);

i := i + 1;

end if;

end loop;

end $$;