

Задание 1.

Примените пожалуйста [скрипт](#) для базы данных PostgreSQL. Если есть необходимость, то вы можете применить его заново, предварительно удалив таблицы командой

DROP TABLE *имя_таблицы*;

Применить скрипт - это значит выполнить его через ваш используемый IDE (DataGrip, DBeaver , pgAdmin)

Результат работы:

```
INSERT 0 1
```

```
Query returned successfully in 63 msec.
```



Задание 2

- а. Напишите SQL запрос, который возвращает имена студентов и название курсов университетов в одном списке. Результат отсортируйте в убывающем порядке. Пример части результата представлен ниже

Код запроса:

```
SELECT s.name, c.name  
FROM student s  
INNER JOIN course c ON s.college_id = c.college_id  
ORDER BY s.name DESC;
```

Результат работы:

| | name character varying  | name character varying  |
|---|---|---|
| 1 | Сергей Петров | Актерское мастерство |
| 2 | Ильяс Мухаметшин | Цифровая трансформация |
| 3 | Иван Иванов | Введение в РСУБД |
| 4 | Екатерина Андреева | Data Mining |
| 5 | Анна Потапова | Нейронные сети |

- b. Напишите SQL запрос который возвращает имена университетов и название курсов в одном списке, но с типом что запись является или “университет” или “курс”. Результат отсортируйте в убывающем порядке по типу записи и потом по имени. Пример части результата представлен ниже

Код запроса:

```
SELECT name, 'университет' AS type
FROM college
UNION ALL
SELECT name, 'курс' AS type
FROM course
ORDER BY type DESC, name DESC;
```

Результат работы:



| | name character varying | type text |
|----|---------------------------|--------------|
| 1 | Сколково | университет |
| 2 | МФТИ | университет |
| 3 | МГУ | университет |
| 4 | КФУ | университет |
| 5 | Иннополис | университет |
| 6 | Цифровая трансформация | курс |
| 7 | Нейронные сети | курс |
| 8 | Введение в РСУБД | курс |
| 9 | Актерское мастерство | курс |
| 10 | Data Mining | курс |

- с. Напишите SQL запрос который возвращает название курса и количество заявленных студентов в отсортированном списке по количеству слушателей в возрастающем порядке, **НО** запись с количеством слушателей равным 300 должна быть на первом месте. Ограничьте вывод данных до 3 строк. Пример результата представлен ниже

Код запроса:

```
SELECT name, amount_of_students
FROM course
ORDER BY CASE WHEN amount_of_students = 300 THEN 0 ELSE 1
          END, amount_of_students
LIMIT 3;
```

Результат работы:

| | name character varying  | amount_of_students integer  |
|---|---|---|
| 1 | Введение в РСУБД | 300 |
| 2 | Data Mining | 10 |
| 3 | Актерское мастерство | 15 |

d. Напишите DML запрос который создает новый **offline** курс со следующими характеристиками:

- id = 60
- название курса = Machine Learning
- количество студентов = 17
- курс проводится в том же университете что и курс Data Mining

Предоставьте **INSERT** выражение, которое заполняет необходимую таблицу данными (Не используйте вложенные подзапросы!).

Приложите скрин результата запроса к данным курсов после выполнения команды **INSERT** к таблице, которая была изменена.

Код запроса:

```
INSERT INTO course (id, name, is_online, amount_of_students, college_id)
VALUES (60, 'Machine Learning', false, 17, 20)
```

Результат работы:

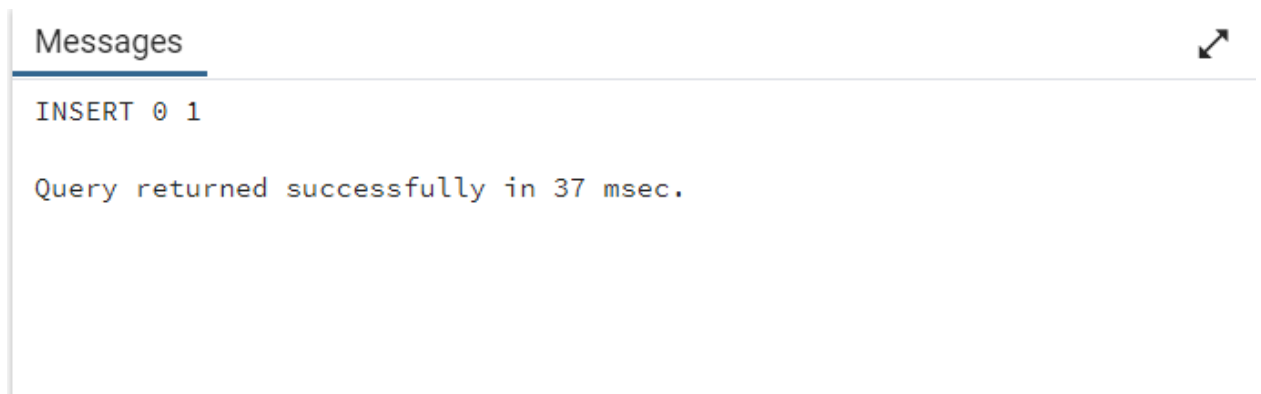


Рис. 1. Результат работы кода вставки

| | id [PK] bigint | name character varying | is_online boolean | amount_of_students integer | college_id bigint |
|---|-------------------|---------------------------|----------------------|-------------------------------|----------------------|
| 1 | 10 | Введение в РСУБД | true | 300 | 10 |
| 2 | 20 | Data Mining | true | 10 | 20 |
| 3 | 30 | Нейронные сети | true | 25 | 30 |
| 4 | 40 | Цифровая трансформация | true | 50 | 40 |
| 5 | 50 | Актерское мастерство | false | 15 | 50 |
| 6 | 60 | Machine Learning | false | 17 | 20 |

Рис. 2. Результат работы кода **SELECT ***

- е. Напишите SQL скрипт который подсчитывает симметрическую разницу множеств A и B.

$$(A \setminus B) \cup (B \setminus A)$$

где A - таблица **course**, B - таблица **student_on_course**, “\” - это разница множеств, “∪” - объединение множеств. Необходимо подсчитать на основании атрибута **id** из обеих таблиц. Результат отсортируйте по 1 столбцу. Пример результата представлен ниже.

| id |
|-----|
| 70 |
| 80 |
| 90 |
| 100 |
| ... |


Код запроса:

```
SELECT id
FROM (
    -- (A \ B)
    SELECT id FROM course
    EXCEPT
    SELECT id FROM student_on_course

    UNION

    -- (B \ A)
    SELECT id FROM student_on_course
    EXCEPT
    SELECT id FROM course
) AS id
ORDER BY 1;
```

Результат работы:

| | id bigint  |
|---|--|
| 1 | 70 |
| 2 | 80 |
| 3 | 90 |
| 4 | 100 |
| 5 | 110 |
| 6 | 120 |
| 7 | 130 |

- f. (** - задача опциональная) Напишите SQL запрос который подсчитывает коэффициент Жаккара (близости множеств) между двумя таблицами А (**course**) и В (**student_on_course**) по атрибуту id на основании формулы:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

где А - таблица **course**, В - таблица **student_on_course**, “ \cup ” - объединение множеств, “ \cap ” - пересечение множеств, $|A \cap B|$ - мощность множества (количество записей) по пересечению, $|A \cup B|$ - мощность множества (количество записей) по объединению

Подсказка: чтобы подсчитать количество элементов используйте конструкцию **SELECT count(*) FROM ...**


Пример результата представлен ниже

| |
|------------------------|
| val |
| 0.46153846153846153846 |

Код программы:

```
SELECT (CAST(COUNT(*) AS FLOAT) / (  
    SELECT COUNT(*)  
    FROM (  
        SELECT id  
        FROM course  
        UNION  
        SELECT id  
        FROM student_on_course  
    ) AS combined_ids  
)) AS коэффициент_Жаккара  
FROM (  
    SELECT id  
    FROM course  
    INTERSECT  
    SELECT id  
    FROM student_on_course  
    ) AS intersection_ids;
```

Результат работы программы:

| | коэффициент_Жаккара double precision  |
|---|--|
| 1 | 0.46153846153846156 |