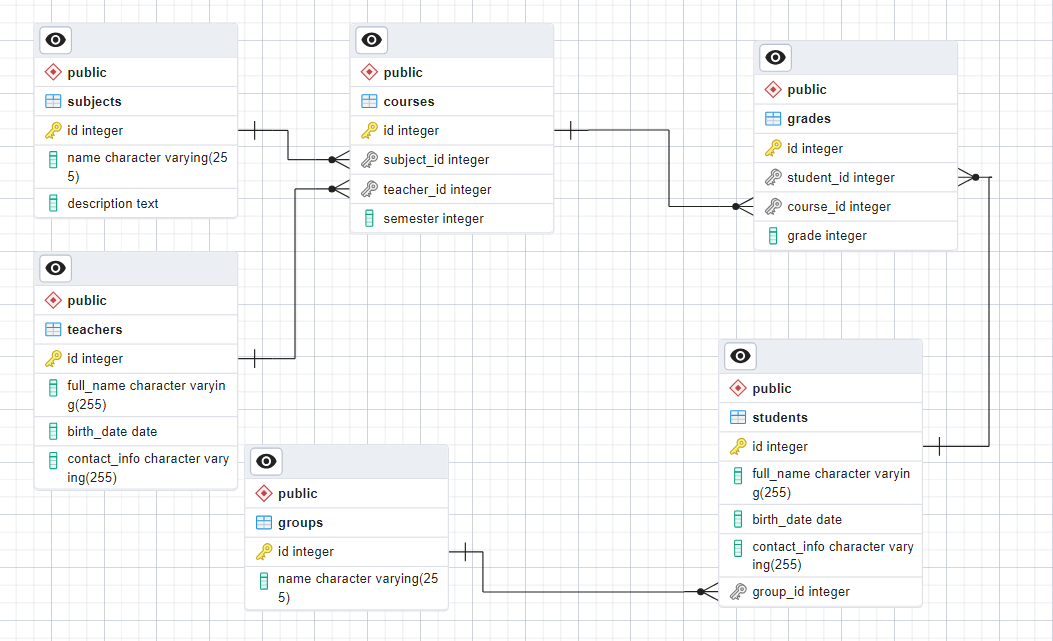
# **Документация**

# **Структура базы данных**



База данных реализована в СУБД PostgreSQL

1. **Таблица "students"**:

**id**: уникальный идентификатор студента (ключ).

**full\_name**: полное имя студента.

**birth\_date**: дата рождения студента.

**contact\_info**: контактная информация студента.

**group\_id**: идентификатор группы, к которой относится студент.

1. **Таблица "teachers"**:

**id**: уникальный идентификатор преподавателя (ключ).

**full\_name**: полное имя преподавателя.

**birth\_date**: дата рождения преподавателя.

**contact\_info**: контактная информация преподавателя.

1. **Таблица "subjects"**:

**id**: уникальный идентификатор предмета (ключ).

**name**: название предмета.

**description**: описание предмета.

1. **Таблица "courses"**:

**id**: уникальный идентификатор курса (ключ).

**subject\_id**: идентификатор предмета (связь с таблицей "subjects").

**teacher\_id**: идентификатор преподавателя (связь с таблицей "teachers").

**semester**: номер семестра, в котором проходит курс.

1. **Таблица "grades"**:

**id**: уникальный идентификатор записи об оценке (ключ).

**student\_id**: идентификатор студента (связь с таблицей "students").

**course\_id**: идентификатор курса (связь с таблицей "courses").

**grade**: оценка студента.

1. **Таблица "groups"**:

**id**: уникальный идентификатор группы (ключ).

**name**: название группы.

# **Отношения между таблицами**

Таблица "students" связана с таблицей "groups" через group\_id.

Таблицы "courses" и "grades" используют внешние ключи для связи с другими таблицами:

В "courses", subject\_id и teacher\_id связывают с "subjects" и "teachers".

В "grades", student\_id и course\_id связывают с "students" и "courses".

# **Целостность данных**

Ограничение CHECK (grade >= 1 AND grade <= 5) в таблице "grades" гарантирует, что оценки находятся в диапазоне от 1 до 5.

# **Выполнение запросов**

**4. Запросы к базе данных:**

- возможность выводить список студентов по определённому предмету — 1 балл;

SELECT students.full\_name

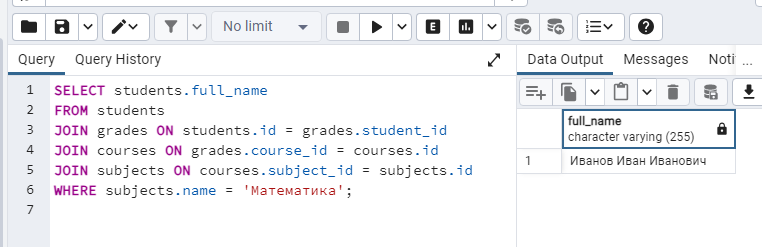
FROM students

JOIN grades ON students.id = grades.student\_id

JOIN courses ON grades.course\_id = courses.id

JOIN subjects ON courses.subject\_id = subjects.id

WHERE subjects.name = 'Математика';



- возможность выводить список предметов, которые преподает конкретный преподаватель — 1 балл;

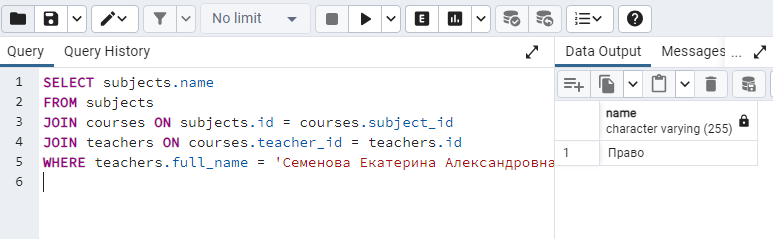
SELECT subjects.name

FROM subjects

JOIN courses ON subjects.id = courses.subject\_id

JOIN teachers ON courses.teacher\_id = teachers.id

WHERE teachers.full\_name = 'Семенова Екатерина Александровна';

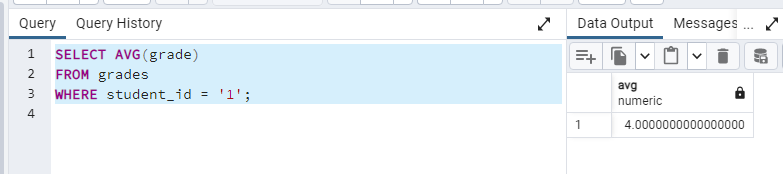


- возможность выводить средний балл студента по всем предметам — 1 балл;

SELECT AVG(grade)

FROM grades

WHERE student\_id = '1';



- возможность выводить рейтинг преподавателей по средней оценке студентов — 1 балл;

SELECT teachers.full\_name, AVG(grades.grade) as average\_grade

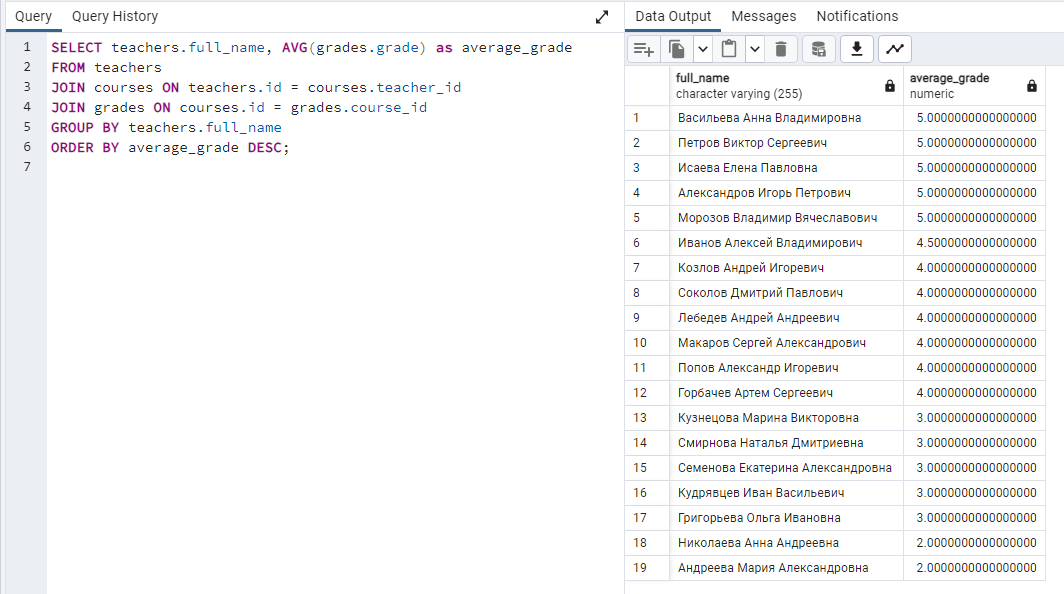
FROM teachers

JOIN courses ON teachers.id = courses.teacher\_id

JOIN grades ON courses.id = grades.course\_id

GROUP BY teachers.full\_name

ORDER BY average\_grade DESC;



- возможность выводить список преподавателей, которые преподавали более 3 предметов за последний год — 1 балл;

SELECT

t.full\_name,

COUNT(DISTINCT c.subject\_id) AS subjects\_taught

FROM

teachers t

JOIN

courses c ON t.id = c.teacher\_id

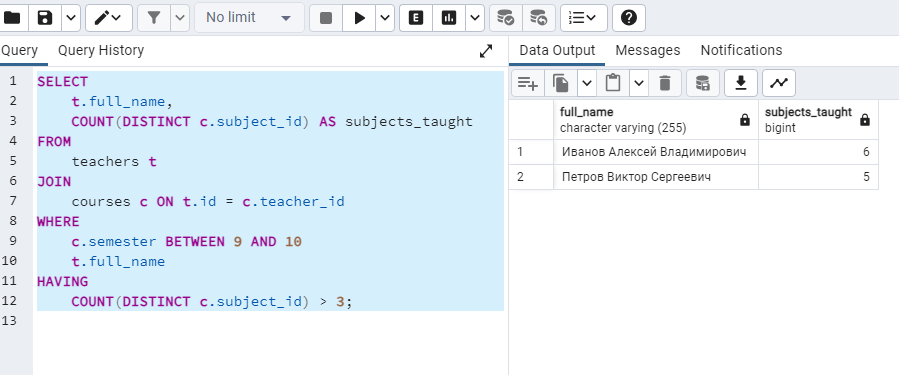
WHERE

c.semester BETWEEN 9 AND 10

t.full\_name

HAVING

COUNT(DISTINCT c.subject\_id) > 3;



- возможность выводить список студентов, которые имеют средний балл выше 4 по математическим предметам, но ниже 3 по гуманитарным — 1 балл;

SELECT

s.full\_name

FROM

students s

WHERE

(SELECT AVG(g.grade)

FROM grades g

JOIN courses c ON g.course\_id = c.id

JOIN subjects sub ON c.subject\_id = sub.id

WHERE s.id = g.student\_id AND sub.name IN ('Математика', 'Информатика')) > 4

AND

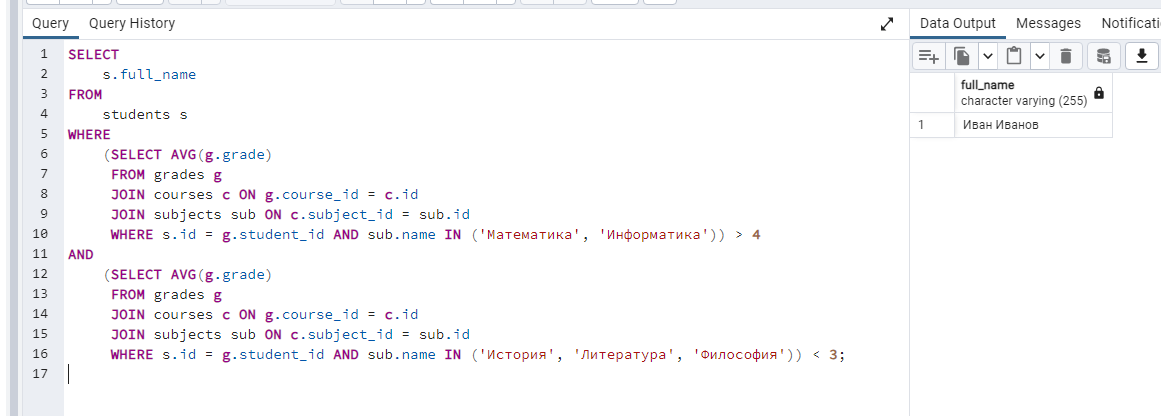
(SELECT AVG(g.grade)

FROM grades g

JOIN courses c ON g.course\_id = c.id

JOIN subjects sub ON c.subject\_id = sub.id

WHERE s.id = g.student\_id AND sub.name IN ('История', 'Литература', 'Философия')) < 3;



- возможность определить предметы, по которым больше всего двоек в текущем семестре — 1 балл;

SELECT subjects.name

FROM subjects

JOIN courses ON subjects.id = courses.subject\_id

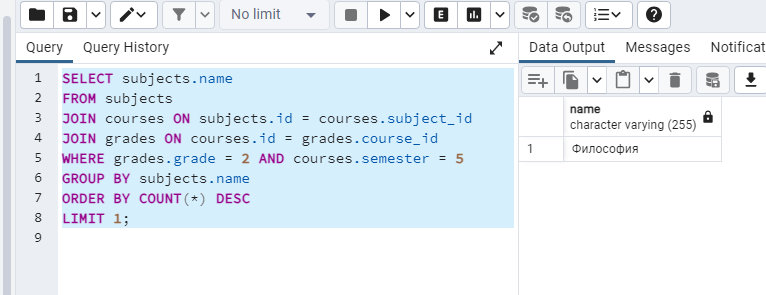
JOIN grades ON courses.id = grades.course\_id

WHERE grades.grade = 2 AND courses.semester = 5

GROUP BY subjects.name

ORDER BY COUNT(\*) DESC

LIMIT 1;



- возможность выводить студентов, которые получили высший балл по всем своим экзаменам, и преподавателей, которые вели эти предметы — 1 балл;

SELECT students.full\_name, teachers.full\_name, MAX(grades.grade)

FROM students

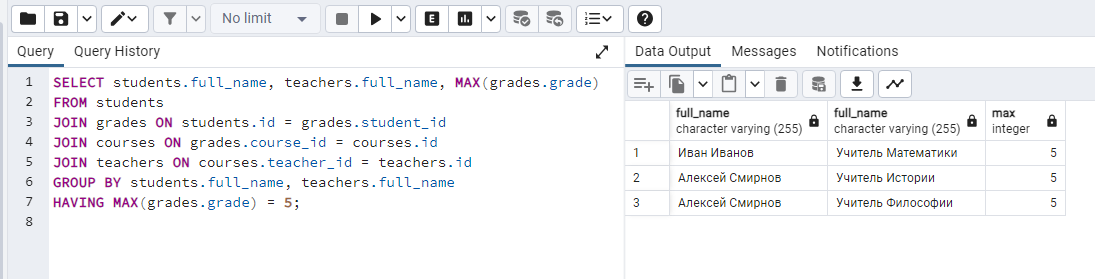
JOIN grades ON students.id = grades.student\_id

JOIN courses ON grades.course\_id = courses.id

JOIN teachers ON courses.teacher\_id = teachers.id

GROUP BY students.full\_name, teachers.full\_name

HAVING MAX(grades.grade) = 5;



- возможность просматривать изменение среднего балла студента по годам обучения — 1 балл;

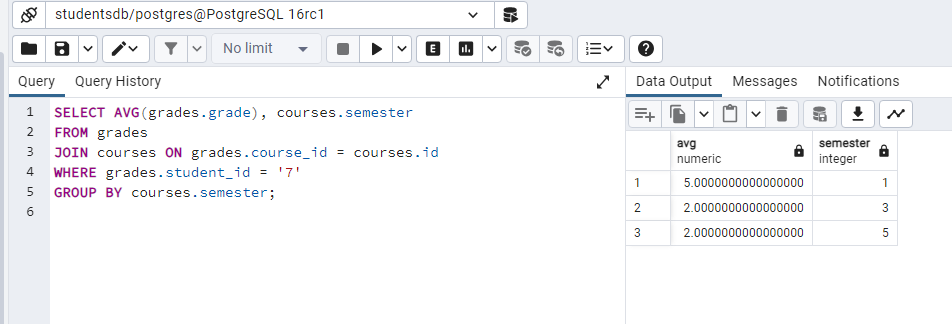
SELECT AVG(grades.grade), courses.semester

FROM grades

JOIN courses ON grades.course\_id = courses.id

WHERE grades.student\_id = '7'

GROUP BY courses.semester;



- возможность определить группы, в которых средний балл выше, чем в других, по аналогичным предметам, чтобы выявить лучшие методики преподавания или особенности состава группы — 1 балл;

SELECT groups.name, AVG(grades.grade)

FROM grades

JOIN students ON grades.student\_id = students.id

JOIN groups ON students.group\_id = groups.id

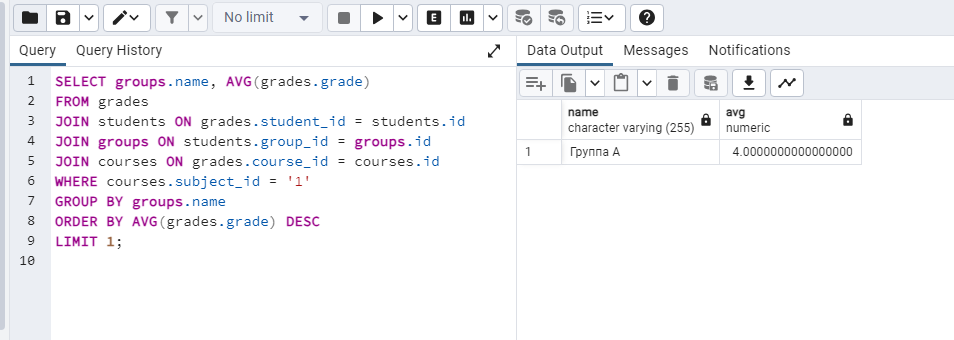
JOIN courses ON grades.course\_id = courses.id

WHERE courses.subject\_id = '1'

GROUP BY groups.name

ORDER BY AVG(grades.grade) DESC

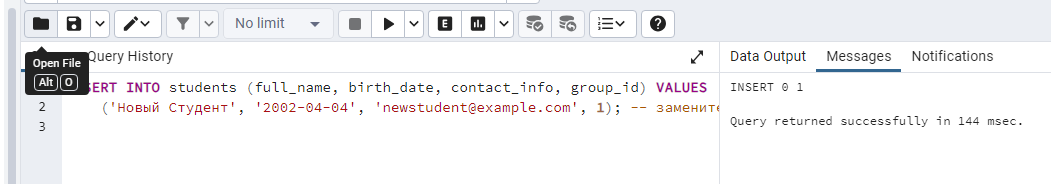
LIMIT 1;



- вставка записи о новом студенте с его личной информацией, такой как ФИО, дата рождения, контактные данные и др. — 1 балл;

INSERT INTO students (full\_name, birth\_date, contact\_info, group\_id) VALUES

('Новый Студент', '2002-04-04', 'newstudent@example.com', 1); -- замените '1' на соответствующий ID группы, если он известен



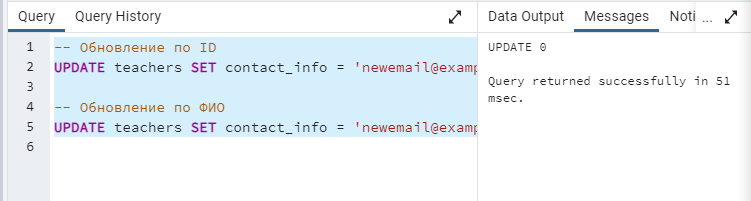
- обновление контактной информации преподавателя, например, электронной почты или номера телефона, на основе его идентификационного номера или ФИО — 1 балл;

-- Обновление по ID

UPDATE teachers SET contact\_info = 'newemail@example.com' WHERE id = 1; -- замените '1' на соответствующий ID преподавателя

-- Обновление по ФИО

UPDATE teachers SET contact\_info = 'newemail@example.com' WHERE full\_name = 'Имя Преподавателя'; -- замените 'Имя Преподавателя' на соответствующее имя



- удаление записи о предмете, который больше не преподают в учебном заведении. Требуется также учесть возможные зависимости, такие как оценки студентов по этому предмету — 1 балл;

-- Удаление оценок по предмету

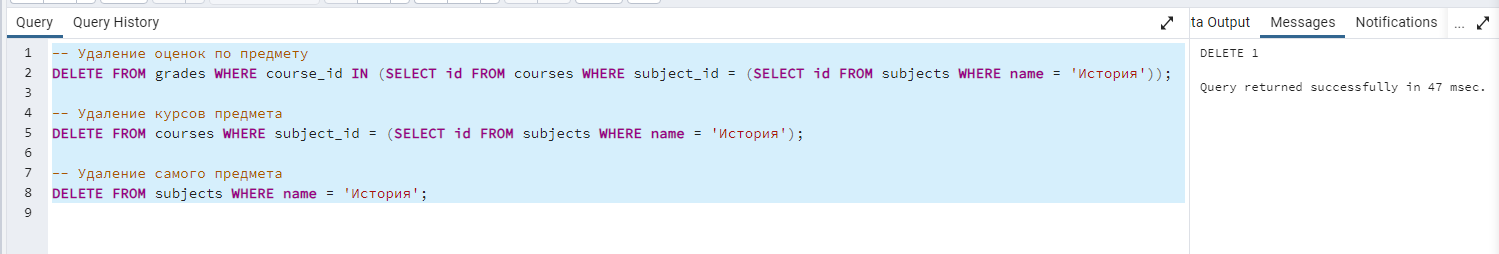
DELETE FROM grades WHERE course\_id IN (SELECT id FROM courses WHERE subject\_id = (SELECT id FROM subjects WHERE name = 'История'));

-- Удаление курсов предмета

DELETE FROM courses WHERE subject\_id = (SELECT id FROM subjects WHERE name = 'История');

-- Удаление самого предмета

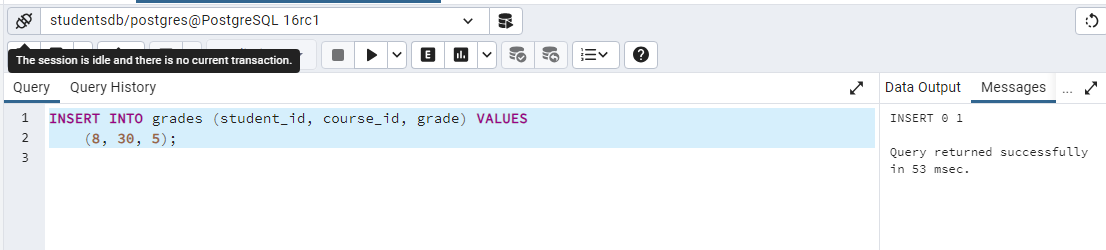
DELETE FROM subjects WHERE name = 'История';



- вставка новой записи об оценке, выставленной студенту по определённому предмету, с указанием даты, преподавателя и полученной оценки — 1 балл.

INSERT INTO grades (student\_id, course\_id, grade) VALUES

(8, 30, 5);



# **Скрипт создания базы данных**

CREATE TABLE students (

id SERIAL PRIMARY KEY,

full\_name VARCHAR(255) NOT NULL,

birth\_date DATE NOT NULL,

contact\_info VARCHAR(255)

);

CREATE TABLE teachers (

id SERIAL PRIMARY KEY,

full\_name VARCHAR(255) NOT NULL,

birth\_date DATE NOT NULL,

contact\_info VARCHAR(255)

);

CREATE TABLE subjects (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL,

description TEXT

);

CREATE TABLE courses (

id SERIAL PRIMARY KEY,

subject\_id INT NOT NULL,

teacher\_id INT NOT NULL,

semester INT NOT NULL,

FOREIGN KEY (subject\_id) REFERENCES subjects (id),

FOREIGN KEY (teacher\_id) REFERENCES teachers (id)

);

CREATE TABLE grades (

id SERIAL PRIMARY KEY,

student\_id INT NOT NULL,

course\_id INT NOT NULL,

grade INT CHECK (grade >= 1 AND grade <= 5),

FOREIGN KEY (student\_id) REFERENCES students (id),

FOREIGN KEY (course\_id) REFERENCES courses (id)

);

CREATE TABLE groups (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL

);

ALTER TABLE students

ADD COLUMN group\_id INT,

ADD FOREIGN KEY (group\_id) REFERENCES groups (id);