МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФГБОУ ВО «ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ»

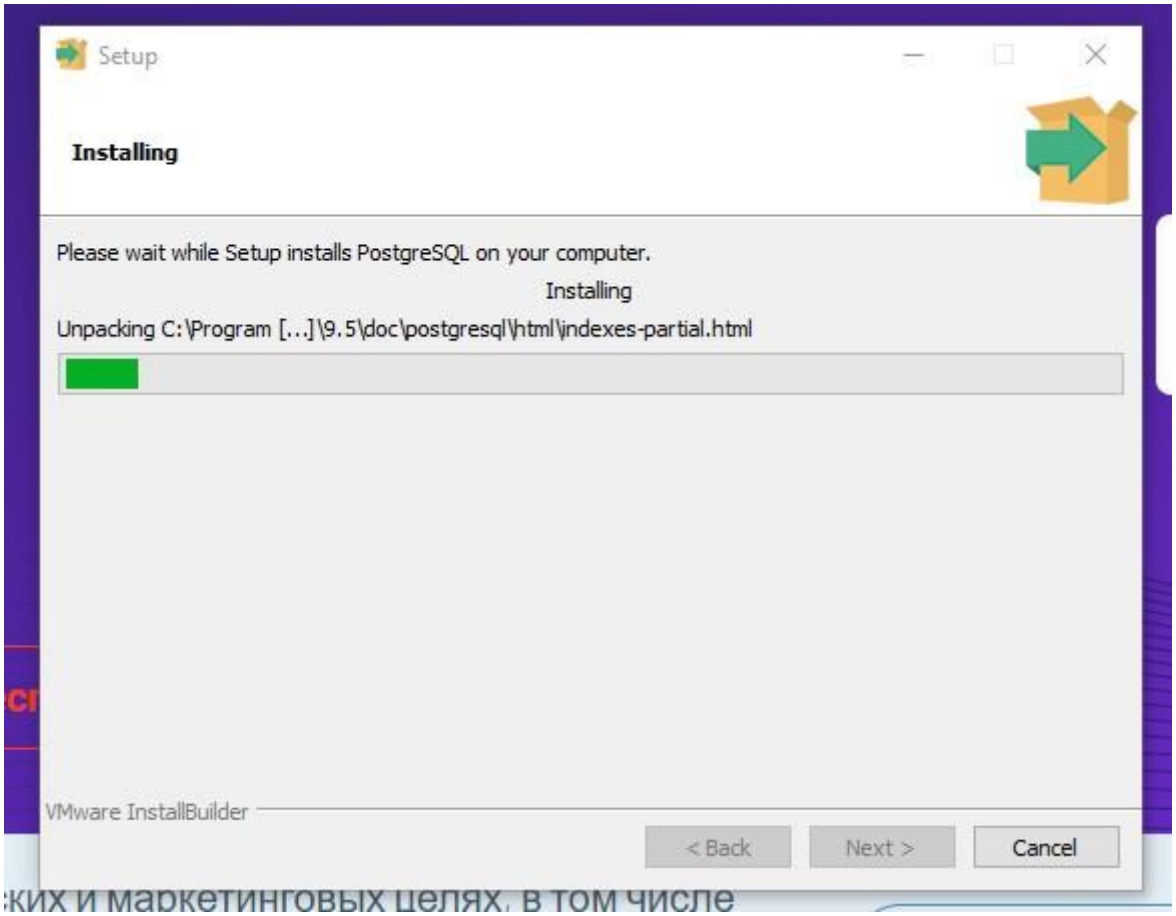
БКЗОПАСНОСТЬ СЕТЕЙ БАЗ ДАННЫХ

СОЗДАНИЕ БАЗЫ ДАННЫХ В POSTGRESQL

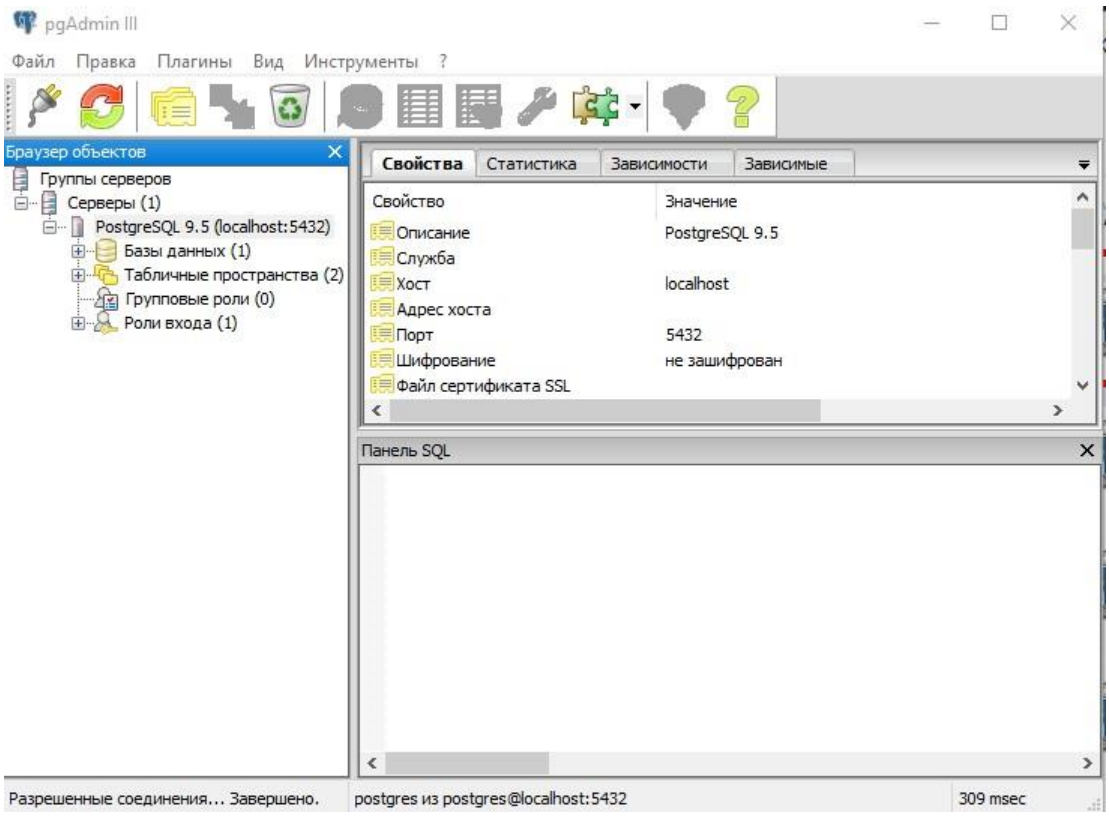
Выполнил: Петрашов Никита Андреевич
Группа: УБ-02
Номер зачетной книжки: 207416

ВОРОНЕЖ
2023

1)Устанавливаем PostgreSQL



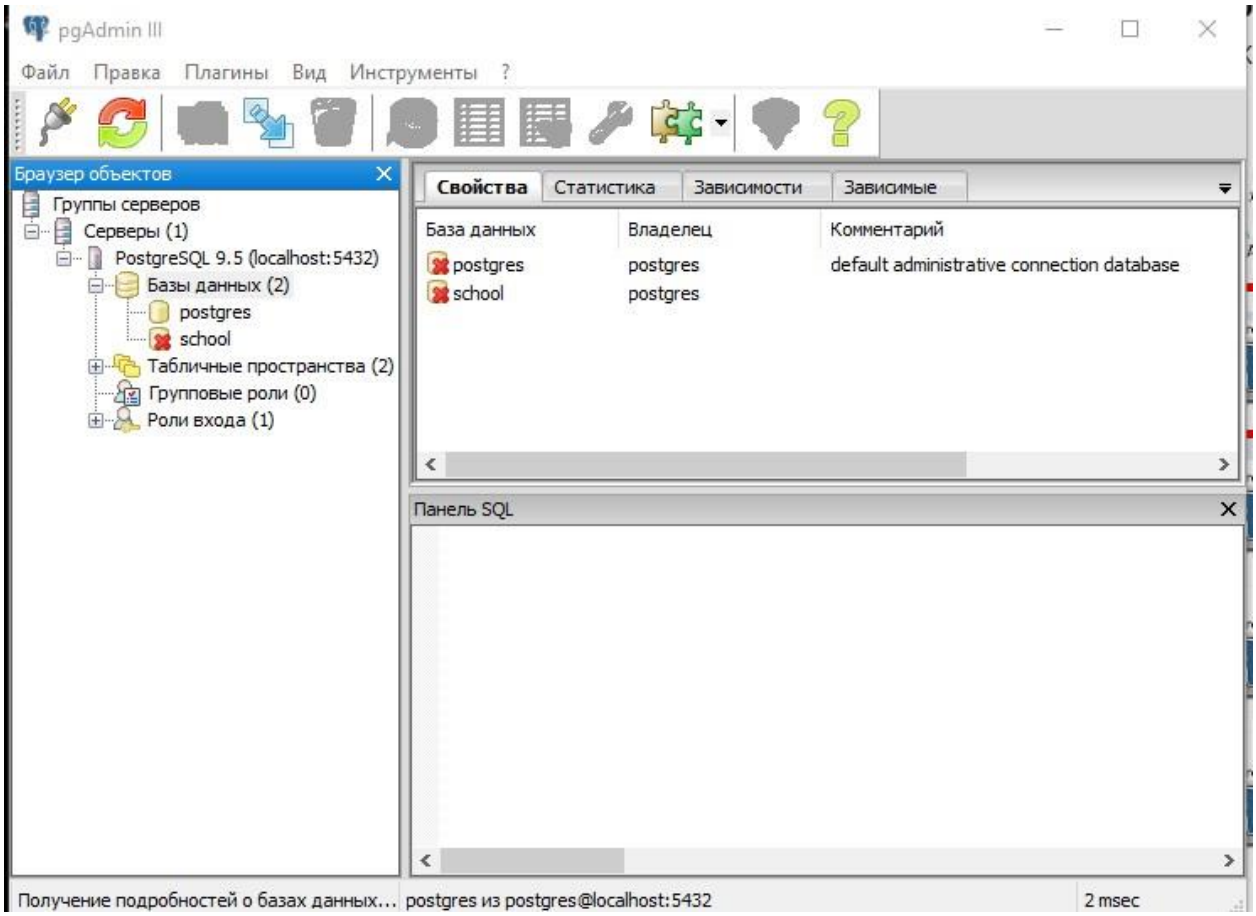
2)Заходим в PgAdmin и проверяем корректность установки, помимо этого авторизовываемся (вводим пароль, который указали при установке).



3)Для создания новой базы данных используется оператор CREATE DATABASE. Создадим CREATE DATABASE school. Для того, чтобы подключиться к ней, используем команду \c school (\connect school).Проверяем корректность создание базы данных через PgAdmin.

```
postgres=#
postgres=# CREATE DATABASE school;
CREATE DATABASE
postgres=# \l
                                        ╥яшёюъ срч фрээ√ї
    ╚ь      | ┬ырфхыхЎ | ╩юфшЁютър |     LC_COLLATE      |      LC_CTYPE       |     ╧Ёртр фюёЄєяр
------------+----------+-----------+---------------------+---------------------+-----------------------
 postgres   | postgres | UTF8      | Russian_Russia.1251 | Russian_Russia.1251 |
 school     | postgres | UTF8      | Russian_Russia.1251 | Russian_Russia.1251 |
 template0  | postgres | UTF8      | Russian_Russia.1251 | Russian_Russia.1251 | =c/postgres          +
            |          |           |                     |                     | postgres=CTc/postgres
 template1  | postgres | UTF8      | Russian_Russia.1251 | Russian_Russia.1251 | =c/postgres          +
            |          |           |                     |                     | postgres=CTc/postgres
```

```
postgres=# CREATE DATABASE school;
ОШИБКА:  база данных "school" уже существует
postgres=# clear
postgres-# \c school
Вы подключены к базе данных "school" как пользователь "postgres".
school-# /conninfo
school-# \conninfo
Вы подключены к базе данных "school" как пользователь "postgres" (сервер "localhost", порт "5432").
school-#
```



4)Для создания таблиц базы данных используется оператор CREATE TABLE. Создадим таблицы: Lesson (Урок\Предмет), Teacher (Учитель), Student (Ученик) и Gradebook (Журнал успеваемости).

При создании таблицы Lesson укажем первичный ключ – id_lesson SERIAL PRIMARY KEY, название предмета с типом данных – name_lesson varchar(30), ФИО учителя, который ведет предмет – fuull_name varchar(45), номер класса, у которого идет предмет – id_class varchar(3), начало предмета – start_l_time TIME, конец предмета end_l_time TIME.

При создании таблицы Teacher укажем первичный ключ – id_teacher SERIAL (тип данных, используемый для создания уникального идентификатора для каждого нового объекта) PRIMARY KEY, ФИО – full_name varchar(60), телефон – phone varchar(15),преподавательский стаж – experience integer , преподаваемый урок\предмет – id_lesson int REFERENCES (используется для установки связи между таблицами) Lesson (id_lesson).

При создании таблицы Student укажем первичный ключ – id_student SERIAL PRIMARY KEY, ФИО – fio_student varchar(60), класс – class varchar(3), телефон – phone varchar(15), дата рождения student_l_date DATE.

При создании таблицы Gradebook укажем первичный ключ – id_gradebook SERIAL PRIMARY KEY, ФИО ученика – fio_student int REFERENCES Student (id_student), предмет – name_lesson int REFERENCES Lesson (id_lesson), ФИО преподавателя full_name int REFERENCES Teacher(id_teacher), последняя оценка–grade integer , средний балл – grade_medium integer.

```
school=# CREATE TABLE Lesson(
school(# id_lesson SERIAL PRIMARY KEY,
school(# name_lesson varchar(30) NULL,
school(# fuull_name varchar(45) NULL,
school(# id_class varchar(3) NULL,
school(# start_L_time TIME,
school(# end_L_time TIME
school(# );
CREATE TABLE
```

```
school=# CREATE TABLE Teacher(
school(# id_teacher SERIAL PRIMARY KEY,
school(# full_name varchar(60) NULL,
school(# phone varchar(15) NULL,
school(# experience integer NULL,
school(# id_lesson int REFERENCES Lesson (id_lesson)
school(# );
CREATE TABLE
```

```
school=# CREATE TABLE Student(
school(# id_student SERIAL PRIMARY KEY,
school(# fio_student varchar(60) NULL,
school(# class varchar(3) NULL,
school(# phone varchar(15) NULL,
school(# student_l_date DATE
school(# );
CREATE TABLE
```

```
school=# CREATE TABLE Gradebook(
school(# id_gradebook SERIAL PRIMARY KEY,
school(# fio_student int REFERENCES Student (id_student),
school(# name_lesson int REFERENCES Lesson (id_lesson),
school(# full_name int REFERENCES Teacher(id_teacher),
school(# grade integer NULL,
school(# grade_medium integer NULL
school(# );
CREATE TABLE
school=#
```

После выполнения данного запроса можно получить список созданных таблиц и связей с помощью команды \d. Результат работы команды имеет вид:

```
school=# \d
                    ╤яшёюъ ю€эю°хэшщ
 ╤їхьр  |            ╚ь             |        ╥шя         | ┬ырфхыхЎ
--------+---------------------------+--------------------+----------
 public | gradebook                 | ЄрсышЎр            | postgres
 public | gradebook_id_gradebook_seq | яюёыхфютрЄхы№эюёЄ№ | postgres
 public | lesson                    | ЄрсышЎр            | postgres
 public | lesson_id_lesson_seq      | яюёыхфютрЄхы№эюёЄ№ | postgres
 public | student                   | ЄрсышЎр            | postgres
 public | student_id_student_seq    | яюёыхфютрЄхы№эюёЄ№ | postgres
 public | teacher                   | ЄрсышЎр            | postgres
 public | teacher_id_teacher_seq    | яюёыхфютрЄхы№эюёЄ№ | postgres
(8 ёЄЁюъ)
```

Список созданных индексов можно получить с помощью команды \di. Результат представлен в таблице:

```
school=# \di
                     ╤яшёюъ ю€эю°хэшщ
 ╤їхьр  |       ╚ь        |  ╥шя   | ┬ырфхыхЎ |  ╥рсышЎр
--------+-----------------+--------+----------+-----------
 public | gradebook_pkey  | шэфхъё | postgres | gradebook
 public | lesson_pkey     | шэфхъё | postgres | lesson
 public | student_pkey    | шэфхъё | postgres | student
 public | teacher_pkey    | шэфхъё | postgres | teacher
(4 ёЄЁюъш)
```

Для вставки данных в таблицы служит оператор INSERT. Для того, чтобы вывести заполненную таблицу используем оператор SELECT – SELECT * FROM **название_таблицы**.

```
school=# INSERT INTO Lesson(
school(# name_lesson, fuull_name, id_class, start_l_time, end_l_time)
school-# VALUES ('Математика','Петров Н.А.','11А','08:00:00','08:45:00');
INSERT 0 1
school=# INSERT INTO Lesson(
school(# name_lesson, fuull_name, id_class, start_l_time, end_l_time)
school-# VALUES ('Физка','Климов Н.А.','11Б','08:00:00','08:45:00');
INSERT 0 1
school=# INSERT INTO Lesson(
school(# name_lesson, fuull_name, id_class, start_l_time, end_l_time)
school-# VALUES ('Химия','Карасев Д.Д.','10Б','08:00:00','08:45:00');
INSERT 0 1
school=# INSERT INTO Lesson(
school(# name_lesson, fuull_name, id_class, start_l_time, end_l_time)
school-# VALUES ('ИЗО','Гагарин Д.Д.','10А','08:45:00','09:40:00');
INSERT 0 1
school=# INSERT INTO Lesson(
school(# name_lesson, fuull_name, id_class, start_l_time, end_l_time)
school-# VALUES ('ПСО','Гаврилов В.С.','9А','08:45:00','09:40:00');
INSERT 0 1
school=# SELECT * FROM Lesson;
 id_lesson | name_lesson |  fuull_name   | id_class | start_l_time | end_l_time
-----------+-------------+---------------+----------+--------------+------------
         1 | Математика  | Петров Н.А.   | 11А      | 08:00:00     | 08:45:00
         2 | Физка       | Климов Н.А.   | 11Б      | 08:00:00     | 08:45:00
         3 | Химия       | Карасев Д.Д.  | 10Б      | 08:00:00     | 08:45:00
         4 | ИЗО         | Гагарин Д.Д.  | 10А      | 08:45:00     | 09:40:00
         5 | ПСО         | Гаврилов В.С. | 9А       | 08:45:00     | 09:40:00
(5 ёёЁюъ)
```

```
school=# INSERT INTO Teacher(
school(# full_name, phone, experience, id_lessonn)
school-# VALUES ('Петров Н.А.',' +79548478474 ','12','Математика');
INSERT 0 1
school=# INSERT INTO Teacher(
school(# full_name, phone, experience, id_lessonn)
school-# ('Климов Н.А.',' +79646478474 ','17','Физика');
ОШИБКА:  ошибка синтаксиса (примерное положение: "'?<Ё┐Rў ?.?.'")
СТРОКА 3: ('?<Ё┐Rў ?.?.',' +79646478474 ','17','"Ё§ЁЄ ');
          ^
school=# INSERT INTO Teacher(
school(# full_name, phone, experience, id_lessonn)
school-# ('Климов Н.А.',' +79646478474 ','17','Физика');
ОШИБКА:  ошибка синтаксиса (примерное положение: "'?<Ё┐Rў ?.?.'")
СТРОКА 3: ('?<Ё┐Rў ?.?.',' +79646478474 ','17','"Ё§ЁЄ ');
          ^
school=# INSERT INTO Teacher(
school(# full_name, phone, experience, id_lessonn)
school-#  VALUES ('Климов Н.А.',' +79646478474 ','17','Физика');
INSERT 0 1
school=# INSERT INTO Teacher(
school(# full_name, phone, experience, id_lessonn)
school-# VALUES ('Карасев Д.Д.',' +75454554554 ','21','Химия');
INSERT 0 1
school=# INSERT INTO Teacher(
school(# full_name, phone, experience, id_lessonn)
school-# VALUES ('Гагарин Д.Д',' +75454524455 ','17','ИЗО');
INSERT 0 1
school=# INSERT INTO Teacher(
school(# full_name, phone, experience, id_lessonn)
school-# VALUES ('Гаврилов В.С.',' +78478478395 ','7','ПСО');
INSERT 0 1
school=# SELECT * FROM Teacher;
 id_teacher |   full_name   |     phone      | experience | id_lessonn
------------+---------------+----------------+------------+------------
          1 | Петров Н.А.   |  +79548478474  |         12 | Математика
          2 | Климов Н.А.   |  +79646478474  |         17 | Физика
          3 | Карасев Д.Д.  |  +75454554554  |         21 | Химия
          4 | Гагарин Д.Д   |  +75454524455  |         17 | ИЗО
          5 | Гаврилов В.С. |  +78478478395  |          7 | ПСО
(5 ёЄЁюъ)
```

```
school=# INSERT INTO Student(
school(# fio_student, class, phone, student_l_date)
school-#  VALUES ('Петрашов Н.А','11А',' +79435878877 ','2001-11-26');
INSERT 0 1
school=# INSERT INTO Student(
school(# fio_student, class, phone, student_l_date)
school-# VALUES ('Стутко В.Д.','11А',' +79435874356 ','2001-06-17');
INSERT 0 1
school=# INSERT INTO Student(
school(# fio_student, class, phone, student_l_date)
school-#  VALUES ('Сароян С.А','11А',' +79764564356 ','2001-06-24');
INSERT 0 1
school=# INSERT INTO Student(
school(# fio_student, class, phone, student_l_date)
school-# VALUES ('Аничкин Д.Ю','10А',' +79564567788 ','2002-01-26');
INSERT 0 1
school=# INSERT INTO Student(
school(# fio_student, class, phone, student_l_date)
school-# VALUES ('Бульвар Д.П.','9А',' +79764545555 ','2003-12-26');
INSERT 0 1
school=#  SELECT * FROM Student;
 id_student | fio_student  | class |     phone      | student_l_date
------------+--------------+-------+----------------+----------------
          1 | Петрашов Н.А | 11А   |  +79435878877  | 2001-11-26
          2 | Стутко В.Д.  | 11А   |  +79435874356  | 2001-06-17
          3 | Сароян С.А   | 11А   |  +79764564356  | 2001-06-24
          4 | Аничкин Д.Ю  | 10А   |  +79564567788  | 2002-01-26
          5 | Бульвар Д.П. | 9А    |  +79764545555  | 2003-12-26
(5 ёЄЁюъ)


school=#
```

```
school=#  INSERT INTO Gradebook(
school(#  fio_studentt, name_lessonn, full_namee,grade, grade_medium)
school-#  VALUES ('Петрашов Н.А','Математика','Петров Н.А.',4,4);
INSERT 0 1
school=#  INSERT INTO Gradebook(
school(#  fio_studentt, name_lessonn, full_namee,grade, grade_medium)
school-#  VALUES ('Стутко В.Д','Физика','Климов Н.А.',4,4);
INSERT 0 1
school=#  INSERT INTO Gradebook(
school(#  fio_studentt, name_lessonn, full_namee,grade, grade_medium)
school-#  VALUES ('Сароян С.А.','Химия','Карасев Д.Д.',3,3);
INSERT 0 1
school=#  INSERT INTO Gradebook(
school(#  fio_studentt, name_lessonn, full_namee,grade, grade_medium)
school-# VALUES ('Аничкин Д.Ю.','ПСО','Гаврилов В.С.',3,3);
INSERT 0 1
school=#  INSERT INTO Gradebook(
school(#  fio_studentt, name_lessonn, full_namee,grade, grade_medium)
school-# VALUES ('Бульвар Д.П.','ИЗО','Гагарин Д.Д.',5,5);
INSERT 0 1
school=# SELECT * FROM Gradebook;
 id_gradebook | fio_studentt | name_lessonn |  full_namee   | grade | grade_medium
--------------+--------------+--------------+---------------+-------+--------------
            1 | Петрашов Н.А | Математика   | Петров Н.А.   |     4 |            4
            2 | Стутко В.Д   | Физика       | Климов Н.А.   |     4 |            4
            3 | Сароян С.А.  | Химия        | Карасев Д.Д.  |     3 |            3
            4 | Аничкин Д.Ю. | ПСО          | Гаврилов В.С. |     3 |            3
            5 | Бульвар Д.П. | ИЗО          | Гагарин Д.Д.  |     5 |            5
(5 ёЄЁюъ)
```

Например, чтобы вывести название предмета, ФИО учителя и время начала занятия из таблицы Lesson и отсортировать их по времени начала занятия, нужно написать запрос следующего вида:

SELECT name_lesson, full_name, start_l_time FROM Lesson ORDER BY start_l_time;

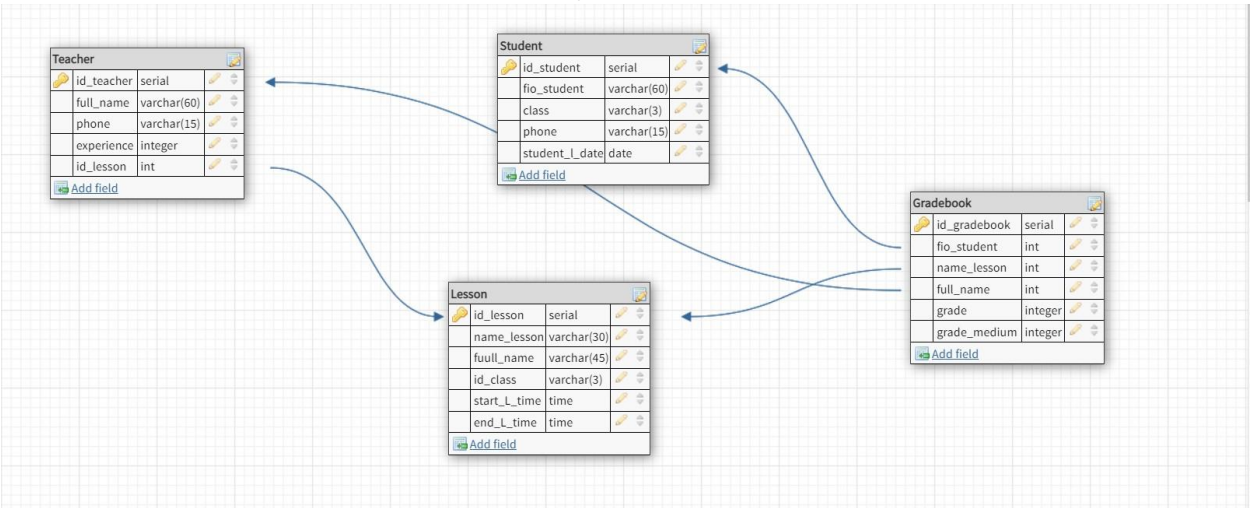
При этом результаты будут отсортированы в порядке возрастания времени начала занятия. Если нужно отсортировать их в порядке убывания, нужно добавить ключевое слово DESC после названия столбца:

SELECT name_lesson, full_name, start_l_time FROM Lesson ORDER BY start_l_time DESC;

Таким образом, результаты будут отсортированы в порядке убывания времени начала занятия.

```
school=# SELECT name_lesson, fuull_name, start_l_time FROM Lesson ORDER BY start_l_time DESC;
 name_lesson |  fuull_name   | start_l_time
-------------+---------------+--------------
 ИЗО         | Гагарин Д.Д.  | 08:45:00
 ПСО         | Гаврилов В.С. | 08:45:00
 Математика  | Петров Н.А.   | 08:00:00
 Физка       | Климов Н.А.   | 08:00:00
 Химия       | Карасев Д.Д.  | 08:00:00
(5 ёЄЁюъ)
```

Создаем схему



SQL представления

```
school=# CREATE VIEW FIOY AS
school-# SELECT full_name  AS FIOYY
school-# FROM Teacher
school-# ORDER BY full_name;
CREATE VIEW
school=# SELECT *FROM FIOY;
     fioyy
---------------
 Гагарин Д.Д
 Гаврилов В.С.
 Карасев Д.Д.
 Климов Н.А.
 Петров Н.А.
(5 ёЄЁюъ)
```

Создает представление (VIEW) с именем FIOY которое содержит одну колонку FIOYY. Эта колонка является результатом выборки столбца full_name из таблицы Teacher.

Представление FIOY будет отображать имена всех учителей, отсортированных по алфавиту по возрастанию.
Результаты при выводе таблицы будут получены, если бы происходил вывод только одного столбцы SELECT full_name FROM Teacher;

Использование представлений для скрытия столбцов и строк

```
school=# CREATE VIEW class AS
school-# SELECT name_lesson, fuull_name, id_class
school-# FROM Lesson;
CREATE VIEW
school=# SELECT * FROM class;
 name_lesson |  fuull_name   | id_class
-------------+---------------+----------
 Математика  | Петров Н.А.   | 11A
 Физка       | Климов Н.А.   | 115
 Химия       | Карасев Д.Д.  | 105
 ИЗО         | Гагарин Д.Д.  | 10A
 ПСО         | Гаврилов В.С. | 9A
(5 ёЄЁюъ)
```

код создает представление (view) в базе данных, которое называется "class". Представление содержит данные о клиентах, взятых из таблицы "Lesson".

Представление возвращает три столбца: "Name_lesson" (название предмета), "fuull_name" (имя учителя) и "id_class" (номер класса), которые выбираются из таблицы "Lesson".

Использование представлений для отображения вычисляемых столбцов

```
school=# CREATE VIEW STUD AS
school-# SELECT class,
school-# (’(’ || class || ’)’) || Phone AS Phone
school-# FROM STUDENT;
CREATE VIEW
school=# SELECT FROM STUD;
--
(5 ёЄЁюъ)


school=# SELECT * FROM STUD;
 class |         phone
-------+----------------------
 11A   | (11A) +79435878877
 11A   | (11A) +79435874356
 11A   | (11A) +79764564356
 10A   | (10A) +79564567788
 9A    | (9A) +79764545555
(5 ёЄЁюъ)
```

код создает представление (view) в базе данных, которое называется "Stud". Представление отображает два столбца из таблицы "Student" - "class" и "Phone".

Использование представлений для скрытия сложного синтаксиса

```
school=# CREATE VIEW GradebookData AS
school-# SELECT s.fio_student AS StudentName,
school-#  l.name_lesson AS LessonName,
school-# t.full_name AS TeacherName,
school-# g.grade AS Grade,
school-# g.grade_medium AS GradeMedium
school-# FROM Gradebook g
school-# JOIN Lesson l ON g.name_lessonn = l.name_lesson
school-# JOIN Teacher t ON g.full_namee = t.full_name
school-# JOIN Student s ON g.fio_studentt = s.fio_student;
CREATE VIEW
school=# SELECT * FROM GradebookData;
 studentname | lessonname | teachername | grade | grademedium
-------------+------------+-------------+-------+-------------
(0 ёЄЁюъ)


school=# SELECT  FROM GradebookData;
--
(0 ёЄЁюъ)


school=# SELECT * FROM GradebookData;
 studentname | lessonname | teachername | grade | grademedium
-------------+------------+-------------+-------+-------------
(0 ёЄЁюъ)


school=# SELECT StudentName, LessonName, TeacherName, Grade FROM GradebookData;
 studentname | lessonname | teachername | grade
-------------+------------+-------------+-------
(0 ёЄЁюъ)
```

Отображает данные из таблицы "Gradebook" и связанных таблиц "Lesson", "Teacher" и "Student". Он выбирает имена столбцов для отображения данных, чтобы сделать их более понятными. При использовании этого представления можно получить информацию о студентах, учителях, уроках и оценках, которые они получили, с помощью одного запроса.

Для получения количества строк в таблице "Lesson" можно использовать следующий код:

```
school=# SELECT COUNT(*) INTO rowcount
school-# FROM Lesson;
SELECT 1
```

Данный запрос поможет выделить данные из таблиц "Lesson", "Teacher" и "Gradebook" и получить общую информацию о количестве оценок, средних оценках и т.д. для каждой уникальной комбинации учителя, названия урока, оценки и среднего балла.

```
school=# SELECT
school-# Teacher.full_name AS TeacherName,
school-# Lesson.name_lesson AS LessonName,
school-# Gradebook.grade AS Grade,
school-# Gradebook.grade_medium AS GradeMedium,
school-# COUNT(Gradebook.id_gradebook) AS TotalGrades
school-# FROM
school-# Gradebook
school-# JOIN Lesson ON Gradebook.name_lessonn = Lesson.name_lesson
school-# JOIN Teacher ON Teacher.id_teacher = Lesson.id_lesson
school-# GROUP BY
school-# Teacher.full_name,
school-# Lesson.name_lesson,
school-# Gradebook.grade,
school-# Gradebook.grade_medium;
  teachername   | lessonname | grade | grademedium | totalgrades
----------------+------------+-------+-------------+-------------
 Карасев Д.Д.   | Химия      |     3 |           3 |           1
 Гагарин Д.Д    | ИЗО        |     5 |           5 |           1
 Гаврилов В.С.  | ПСО        |     3 |           3 |           1
 Петров Н.А.    | Математика |     4 |           4 |           1
(4 ёЄЁюъш)


school=#
```

Хранимые процедуры

```
school=# CREATE OR REPLACE FUNCTION insert_gradebook(
school(#     fio_studentt VARCHAR(15),
school(#     name_lessonn VARCHAR(15),
school(#     full_namee VARCHAR(15),
school(#     grade INTEGER,
school(#     grade_medium INTEGER
school(# )
school-# RETURNS VOID AS $$
school$# BEGIN
school$#     INSERT INTO Gradebook (fio_studentt, name_lessonn, full_namee, grade, grade_medium)
school$#     VALUES (fio_studentt, name_lessonn, full_namee, grade, grade_medium);
school$# END;
school$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
school=# SELECT insert_gradebook('Иванов Иван', 'Математика', 'Петров Петр', 95, 90);
 insert_gradebook
------------------

(1 ёЄЁюър)


school=#
```

Эта процедура принимает значения для всех столбцов таблицы Gradebook и вставляет их в соответствующие столбцы. Можно использовать эту хранимую память, передавая значения времени.
Он позволяет вставить данные в таблицу Gradebook с помощью вызова хранимой процедуры. Можно модифицировать или создать другие хранимые процедуры, в зависимости от потребностей и операций.

Триггеры (все создаю под таблицу Lesson)
Использование триггеров для проверки допустимости вводимых данных

```
school=# CREATE OR REPLACE FUNCTION lesson_check_data() RETURNS TRIGGER AS $$
school$# BEGIN
school$#     IF NEW.start_l_time >= NEW.end_l_time THEN
school$#         RAISE EXCEPTION 'Start time must be earlier than end time';
school$#     END IF;
school$#
school$#     RETURN NEW;
school$# END;
school$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
school=#
school=# CREATE TRIGGER lesson_check_data_trigger
school-# BEFORE INSERT OR UPDATE ON Lesson
school-# FOR EACH ROW
school-# EXECUTE FUNCTION lesson_check_data();
CREATE TRIGGER
school=#
```

триггеры проверяют определенные условия для каждой таблицы перед вставкой или обновлением данных. Если условие не выполняется, будет сгенерировано исключение и операция будет отклонена.

Использование триггеров для присвоения значений по умолчанию

```
school=# CREATE OR REPLACE FUNCTION lesson_check_data() RETURNS TRIGGER AS $$
school$# BEGIN
school$#     IF NEW.start_l_time >= NEW.end_l_time THEN
school$#         RAISE EXCEPTION 'Start time must be earlier than end time';
school$#     END IF;
school$#
school$#     RETURN NEW;
school$# END;
school$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
school=#
school=# CREATE TRIGGER lesson_check_data_trigger
school-# BEFORE INSERT OR UPDATE ON Lesson
school-# FOR EACH ROW
school-# EXECUTE FUNCTION lesson_check_data();
CREATE TRIGGER
school=# CREATE OR REPLACE FUNCTION lesson_default_values()
school-# RETURNS TRIGGER AS $$
school$# BEGIN
school$#   IF NEW.name_lesson IS NULL THEN
school$#     NEW.name_lesson := 'Default Lesson Name';
school$#   END IF;
school$#
school$#   IF NEW.full_name IS NULL THEN
school$#     NEW.full_name := 'Default Full Lesson Name';
school$#   END IF;
school$#
school$#   IF NEW.id_class IS NULL THEN
school$#     NEW.id_class := 'Default Class ID';
school$#   END IF;
school$#
school$#   IF NEW.start_l_time IS NULL THEN
school$#     NEW.start_l_time := '00:00:00';
school$#   END IF;
school$#
school$#   IF NEW.end_l_time IS NULL THEN
school$#     NEW.end_l_time := '00:00:00';
school$#   END IF;
school$#
school$#   RETURN NEW;
school$# END;
school$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
school=#
school=# CREATE TRIGGER lesson_default_trigger
school-# BEFORE INSERT ON Lesson
school-# FOR EACH ROW
school-# EXECUTE FUNCTION lesson_default_values();
CREATE TRIGGER
school=#
```

Триггер, обновляющий представление

```
chool=# CREATE TRIGGER update_gradebook_trigger
chool-# AFTER INSERT OR UPDATE OR DELETE ON Gradebook
chool-# FOR EACH STATEMENT
chool-# EXECUTE FUNCTION update_gradebook_view();
REATE TRIGGER
chool=# CREATE OR REPLACE FUNCTION update_gradebook_view()
chool-# RETURNS TRIGGER AS
chool-# $BODY$
chool$# BEGIN
chool$# REFRESH MATERIALIZED VIEW gradebook_view;
chool$# RETURN NULL;
chool$# END;
chool$# $BODY$
chool-# LANGUAGE plpgsql;
REATE FUNCTION
chool=#
```

В представленном коде реализована функция update_gradebook_view, которая выполняет обновление представленияgradebook_view. Затем следует триггер update_gradebook_trigger, который вызывает функцию update_gradebook_view после каждой операции добавления (INSERT), обновления (UPDATE) или удаления (DELETE) в таблице Gradebook.
Создаем представление:

```
school=# CREATE MATERIALIZED VIEW gradebook_view AS
school-# SELECT *
school-# FROM Gradebook;
SELECT 6
school=#
```

Триггер, обеспечивающий ссылочную целостность

```
COMMIT
school-# RETURNS TRIGGER AS $$
school$# BEGIN
school$#     IF NEW.id_class IS NOT NULL AND NOT EXISTS (SELECT 1 FROM Class WHERE id_class = NEW.id_class) THEN
school$#         RAISE EXCEPTION 'Неверное значение id_class';
school$#     END IF;
school$#
school$#     IF NEW.id_lessonn IS NOT NULL AND NOT EXISTS (SELECT 1 FROM Teacher WHERE id_lessonn = NEW.id_lessonn) THEN
school$#         RAISE EXCEPTION 'Неверное значение id_lessonn';
school$#     END IF;
school$#
school$#     RETURN NEW;
school$# END;
school$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
school=#
school=# CREATE TRIGGER lesson_reference_trigger
school-# BEFORE INSERT OR UPDATE ON Lesson
school-# FOR EACH ROW
school-# EXECUTE FUNCTION lesson_reference_integrity();
CREATE TRIGGER
school=#
```

проверяет ссылочную целостность при вставке или обновлении данных в таблице "Lesson"

Модуль TABLEFUNC

```
school=# CREATE OR REPLACE FUNCTION calculate_average_grades_by_lesson()
school-#   RETURNS TABLE (
school(#     name_lesson varchar(30),
school(#     average_grade numeric
school(#   )
school-# AS $$
school$# BEGIN
school$#   RETURN QUERY
school$#   SELECT l.name_lesson, AVG(g.grade) AS average_grade
school$#   FROM Lesson l
school$#   INNER JOIN Gradebook g ON l.name_lesson = g.name_lessonn
school$#   GROUP BY l.name_lesson;
school$# END;
school$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
school=# -- Вызов метода и получение результатов
school=# SELECT *
school-# FROM calculate_average_grades_by_lesson();
 name_lesson |    average_grade
-------------+---------------------
 ИЗО         |  5.0000000000000000
 Математика  | 49.5000000000000000
 Химия       |  3.0000000000000000
 ПСО         |  3.0000000000000000
(4 ёЄЁюъш)


school=# SELECT *
school-# FROM calculate_average_grades_by_lesson()
school-# WHERE name_lesson = 'Математика';
 name_lesson |    average_grade
-------------+---------------------
 Математика  | 49.5000000000000000
(1 ёЄЁюър)
```

Теперь есть метод, который может быть использован для формирования сводных таблиц по предметам.

Функция ROW_NUMBER

```
school=# SELECT ROW_NUMBER() OVER (ORDER BY name_lesson) AS num, name_lesson
school-# FROM lesson;
 num | name_lesson
-----+-------------
   1 | Физка
   2 | ИЗО
   3 | Химия
   4 | Математика
   5 | ПСО
(5 ёЄЁюъ)
```

функция ROW_NUMBER() используется для нумерации строк таблицы "Lesson" в порядке сортировки по столбцу "name_lesson".

```
school=# SELECT *
school-# FROM (
school(#   SELECT ROW_NUMBER() OVER (ORDER BY name_lesson) AS num, name_lesson
school(#   FROM lesson
school(# ) subquery
school-# WHERE num <= 5;
 num | name_lesson
-----+-------------
   1 | Физка
   2 | ИЗО
   3 | Химия
   4 | Математика
   5 | ПСО
(5 ёЄЁюъ)
```

создаем подзапрос, который вычисляет числовую строку с помощью функции ROW_NUMBER(), а затем внешний запрос выбирает только первые пять строк (где число <= 5).

```
school=# SELECT ROW_NUMBER() OVER (PARTITION BY id_class ORDER BY name_lesson) AS num, name_lesson, id_class
school-# FROM lesson;
 num | name_lesson | id_class
-----+-------------+----------
   1 | Химия       | 105
   1 | ИЗО         | 10A
   1 | Физка       | 115
   1 | Математика  | 11A
   1 | ПСО         | 9A
(5 ёЄЁюъ)


school=#
```

здесь строки разбиваются на группы по группе "id_class", и каждая группа столбцов нумеруется отдельно внутри группы в порядке сортировки по столбцу "name_lesson".

Функция COALESCE

```
school=# SELECT id_lesson, COALESCE(name_lesson, '') AS name_lesson, COALESCE(fuull_name, '') AS fuull_name,
school-#        COALESCE(id_class, '') AS id_class, start_l_time, end_l_time
school-# FROM Lesson;
 id_lesson | name_lesson |  fuull_name   | id_class | start_l_time | end_l_time
-----------+-------------+---------------+----------+--------------+------------
         1 | Математика  | Петров Н.А.   | 11A      | 08:00:00     | 08:45:00
         2 | Физка       | Климов Н.А.   | 115      | 08:00:00     | 08:45:00
         3 | Химия       | Карасев Д.Д.  | 105      | 08:00:00     | 08:45:00
         4 | ИЗО         | Гагарин Д.Д.  | 10A      | 08:45:00     | 09:40:00
         5 | ПСО         | Гаврилов В.С. | 9A       | 08:45:00     | 09:40:00
(5 ёЄЁюъ)
```

функция COALESCE для замены значений NULL в соответствующих столбцах на пустую строку

Числовые функции

```
school=# SELECT ABS(100) X1, ABS(-100) X2, ABS(-100.2) X3;
 x1  | x2  |  x3
-----+-----+-------
 100 | 100 | 100.2
(1 ёЄËюър)


school=# SELECT ABS(100) X1, ABS(-100) X2, ABS(-100.2) X3;
 x1  | x2  |  x3
-----+-----+-------
 100 | 100 | 100.2
(1 ёЄËюър)


school=# SELECT CEIL(100) X1, CEIL(-100) X2,
school-# CEIL(100.2) X3, CEIL(-100.2) X4;
 x1  |  x2  | x3  |  x4
-----+------+-----+------
 100 | -100 | 101 | -100
(1 ёЄËюър)


school=# SELECT FLOOR(100.22) X1, FLOOR(-100.22) X2,
school-# FLOOR(100.99) X3, FLOOR(100.01) X4;
 x1  |  x2  | x3  | x4
-----+------+-----+-----
 100 | -101 | 100 | 100
(1 ёЄËюър)


school=# SELECT TRUNC(100.25678) X1, TRUNC(-100.25678) X2,
school-# TRUNC(100.99) X3, TRUNC(100.25678, 2) X4;
 x1  |  x2  | x3  |   x4
-----+------+-----+--------
 100 | -100 | 100 | 100.25
(1 ёЄËюър)


school=# SELECT ROUND(100.25678) X1, ROUND(100.5) X2,
school-# ROUND(100.99) X3, ROUND(100.25678, 2) X4;
 x1  | x2  | x3  |   x4
-----+-----+-----+--------
 100 | 101 | 101 | 100.26
(1 ёЄËюър)


school=# SELECT SIGN(100.22) X1, SIGN(-100.22) X2, SIGN(0) X3;
 x1 | x2 | x3
----+----+----
  1 | -1 |  0
(1 ёЄËюър)


school=# SELECT MOD(10, 3) X1, MOD(10, 2) X2, MOD(100, 98) X3;
 x1 | x2 | x3
----+----+----
  1 |  0 |  2
(1 ёЄËюър)
```

Тригонометрические функци

```
school=# SELECT SIN(0) X1, COS(0) X2, TAN(1) X3, COT(1);
 x1 | x2 |         x3         |        cot
----+----+--------------------+--------------------
  0 |  1 | 1.5574077246549023 | 0.6420926159343306
(1 ёЄЁюър)
```

Строковые и символьные функции

```
postgres=# SELECT
postgres-#     REPLACE(' попа была собака', 'собака', 'кошка') AS X1,
postgres-#     REPLACE(' попа была злая собака', 'злая', '') AS X2,
postgres-#     REPLACE(' попа была собака', 'Собака', 'Кошка') AS X3;
        x1         |         x2          |        x3
-------------------+---------------------+-------------------
  попа была кошка |  попа была  собака |  попа была собака
(1 ёЄЁюър)


postgres=# SELECT LOWER('TeXt DATA') X;
     x
-----------
 text data
(1 ёЄЁюър)


postgres=# SELECT UPPER('TeXt DATA') X;
     x
-----------
 TEXT DATA
(1 ёЄЁюър)
```

```
ostgres=# SELECT REPLACEУ(’ попа была собака’, собака’’, кошка’’) X1,
ostgres-# REPLACEУ(’ попа была злая собака’, злая’’, ’’) X2,
ostgres-# REPLACEУ(’ попа была собака’, Собака’’, Кошка’’) X3;
ШИБКА:  тип "бRЎ Є " не существует
ТРОКА 1: SELECT REPLACE"(' ÏRÏ  Ўл<  бRЎ Є ', бRЎ Є '', ЄRиЄ '') X1,
                                          ^
ostgres=# SELECT TRANSLATE(’Test 12345’, ’e2’, ’E!’) X1,
ostgres-# TRANSLATE(’Test 12345’, ’e234’, ’E’) X2;
    x1     |   x2
-----------+---------
TEst 1!345 | TEst 15
1 ёЄЁюър)


ostgres=# SELECT LTRIM(’ TeXt DATA’) X1,
ostgres-# LTRIM(’ _ # TeXt DATA’, ’ #_’) X2,
ostgres-# LTRIM(’ 1234567890 TeXt DATA’, ’ 1234567890’) X3
ostgres-# UNION ALL
ostgres-# SELECT RTRIM(’TeXt DATA ’) X1,
ostgres-# RTRIM(’TeXt DATA _ # ’, ’ #_’) X2,
ostgres-# RTRIM(’TeXt DATA 1234567890 ’, ’ 1234567890’) X3;
   x1     |    x2     |    x3
----------+-----------+-----------
TeXt DATA | TeXt DATA | TeXt DATA
TeXt DATA | TeXt DATA | TeXt DATA
2 ёЄЁюъш)


ostgres=# CREATE OR REPLACE FUNCTION is_password_correct(
ostgres(# password IN char)
ostgres-# RETURNS int AS $is_password_correct$
ostgres$# BEGIN
ostgres$# IF TRANSLATE(password, ’0123456789’, ’*’) = password THEN
ostgres$# RAISE WARNING
ostgres$# Пароль’ должен содержать хотя бы одну цифру!’;
ostgres$# RETURN 0;
ostgres$# END IF;
ostgres$# RAISE INFO Корректный’ пароль!’;
ostgres$# RETURN 1;
ostgres$# END;
ostgres$# $is_password_correct$ LANGUAGE plpgsql;
ШИБКА:  нераспознанное условие исключения "? aR<м"
ОНТЕКСТ:  компиляция функции PL/pgSQL "is_password_correct" в районе строки 4
ostgres=# SELECT TRANSLATE(’123 455,23’, ’., ’, ’..’) X1,
ostgres-# TRANSLATE(’-123 455.23’, ’., ’, ’..’) X2;
   x1     |     x2
----------+------------
123455.23 | -123455.23
```

Функции работы с датой и временем

```
postgres=# SELECT NOW() D1,
postgres-# NOW() + JUSTIFY_INTERVAL('30 DAYS 1 HOUR 2 MINUTE') D2,
postgres-# NOW() - JUSTIFY_INTERVAL('30 DAYS 1 HOUR 2 MINUTE') D3;
              d1               |              d2               |              d3
-------------------------------+-------------------------------+-------------------------------
 2023-05-25 20:43:38.400353+03 | 2023-06-25 21:45:38.400353+03 | 2023-04-25 19:41:38.400353+03
(1 ёЄЁюър)


postgres=# SELECT
postgres-# DATE_TRUNC('HOUR', NOW()) D1,
postgres-# DATE_TRUNC('DAY', NOW()) D2,
postgres-# DATE_TRUNC('MONTH', NOW()) D3;
          d1           |           d2           |          d3
-----------------------+------------------------+------------------------
 2023-05-25 20:00:00+03 | 2023-05-25 00:00:00+03 | 2023-05-01 00:00:00+03
(1 ёЄЁюър)


postgres=# SELECT
postgres-# DATE_TRUNC('MONTH', NOW()) D1,
postgres-# DATE_TRUNC('MONTH', NOW())
postgres-# + JUSTIFY_INTERVAL('1 MONTH - 1 DAY') D2;
          d1           |          d2
-----------------------+------------------------
 2023-05-01 00:00:00+03 | 2023-05-30 00:00:00+03
(1 ёЄЁюър)


postgres=# SELECT
postgres-# CURRENT_DATE D1,
postgres-# AGE(MAKE_TIMESTAMP(2013, 7, 15, 8, 15, 23.5)) D2,
postgres-# AGE(MAKE_DATE(2016, 3, 3),
postgres(# MAKE_TIMESTAMP(2013, 7, 15, 8, 15, 23.5)) D3;
     d1     |                d2                |                d3
------------+----------------------------------+----------------------------------
 2023-05-25 | 9 years 10 mons 9 days 15:44:36.5 | 2 years 7 mons 18 days 15:44:36.5
(1 ёЄЁюър)


postgres=# SELECT
postgres-# NOW() D1,
postgres-# EXTRACT(MONTH FROM NOW()) D2,
postgres-# EXTRACT(YEAR FROM NOW()) D3,
postgres-# EXTRACT(MINUTE FROM NOW()) D4;
              d1               | d2 |  d3  | d4
-------------------------------+----+------+----
 2023-05-25 20:44:12.602209+03 |  5 | 2023 | 44
(1 ёЄЁюър)


postgres=# SELECT NOW() D1,
postgres-# TO_CHAR(NOW(), 'DD.MM.YY HH24:MI') D2;
              d1               |       d2
-------------------------------+----------------
 2023-05-25 20:44:36.828089+03 | 25.05.23 20:44
(1 ёЄЁюър)


postgres=# SELECT
postgres-# TO_DATE('05 Dec 2000', 'DD Mon YYY') D1,
postgres-# TO_DATE('15.12.2000', 'dd.mm.yy') D2;
     d1     |     d2
------------+------------
 2000-12-05 | 2000-12-15
(1 ёЄЁюър)
```

Вывод: я научился работать в psql и освежил+ обновил свои знания языка sql.