**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Воронежский государственный технический университет»

(ФГБОУ ВО «ВГТУ», ВГТУ)

Строительно-политехнический колледж

**КУРСОВОЙ ПРОЕКТ**

по дисциплине Технология разработки и защиты баз данных

тема: Проектирование и разработка базы данных по индивидуальной предметной области

**Расчетно-пояснительная записка**

Разработала ИСП-201о, А.А. Клочкова

*(Подпись, дата) (группа, И.О., Фамилия)*

Руководитель М.А. Попов

*(Подпись, дата) (И.О., Фамилия)*

Оценка

**Министерство науки и высшего образования российской федерации**

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Воронежский государственный технический университет»

(ФГБОУ ВО «ВГТУ», ВГТУ)

Строительно-политехнический колледж

ЗАДАНИЕ

На курсовой проект

Дисциплина (МДК) Технология разработки и защиты баз данных

Тема: Проектирование и разработка базы данных по индивидуальной предметной области

Студент группы ИСП-201о Клочкова Анастасия Александровна

*(индекс группы) (фамилия, имя, отчество)*

Технические условия Предметная область: Деканат, ПО построения ER-диаграмм: Software Ideas Modeller. СУБД: SQLite.

Содержание (разделы, графические работы, расчеты и проч.)

Сроки выполнения этапов: с по

Руководитель М.А. Попов

*(Подпись, дата) (И.О., Фамилия)*

Студент А.А. Клочкова

*(Подпись, дата) ( И.О., Фамилия)*

**Содержание**

[Введение 4](#_Toc93506198)

[1. Анализ предметной области 6](#_Toc93506199)

[1.1 Описание предметной области 6](#_Toc93506200)

[1.2 Подробный анализ предметной области 8](#_Toc93506201)

[1.3 Разработка информационно-логической модели предметной области 9](#_Toc93506202)

[1.4 Разработка даталогической модели предметной области 9](#_Toc93506203)

[2. Реализация базы данных 14](#_Toc93506204)

[2.1 Создание отчётов 29](#_Toc93506205)

[2.2 Хранимые процедуры 33](#_Toc93506206)

[3 Защита базы данных 38](#_Toc93506207)

[3.1 Шифрование данных в БД 38](#_Toc93506208)

[3.2 Комбинирование методов защиты БД 41](#_Toc93506209)

[Заключение 44](#_Toc93506210)

[Список использованных источников 45](#_Toc93506211)

# **Введение**

Проектирование баз данных — процесс создания схемы базы данных и определения необходимых ограничений целостность.

Основные задачи:

* Обеспечение хранения в БД всей необходимой информации.
* Обеспечение возможности получения данных по всем необходимым запросам.
* Сокращение избыточности и дублирования данных.
* Обеспечение целостности базы данных.

Концептуальное (инфологическое) проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных.

Чаще всего концептуальная модель базы данных включает в себя:

* описание информационных объектов или понятий предметной области и связей между ними.
* описание ограничений целостности, то есть требований к допустимым значениям данных и к связям между ними.

Логическое (даталогическое) проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной. Для реляционной модели данных даталогическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи.

Преобразование концептуальной модели в логическую модель, как правило, осуществляется по формальным правилам. Этот этап может быть в значительной степени автоматизирован.

На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Физическое проектирование — создание схемы базы данных для конкретной СУБД. Специфика конкретной СУБД может включать в себя ограничения на именование объектов базы данных, ограничения на поддерживаемые типы данных и т. п. Кроме того, специфика конкретной СУБД при физическом проектировании включает выбор решений, связанных с физической средой хранения данных (выбор методов управления дисковой памятью, разделение БД по файлам и устройствам, методов доступа к данным), создание индексов и т. д.

# **Анализ предметной области**

# **Описание предметной области**

Для успешного и эффективного функционирования ВУЗа необходимо комплексное внедрение информационно-коммуникационных технологий (ИКТ), охватывающее все без исключения функциональные подсистемы учебного заведения (такие как деканаты, библиотека, приёмная комиссия, бухгалтерия, кафедры и др.)

Одним из структурных подразделений ВУЗа, наиболее очевидно требующих автоматизации, является деканат.

Деканат ­ подразделение ВУЗа, осуществляющее контроль и организацию деятельности факультета.

В деканате проводится организация учебного процесса в рамках факультета, ведение документации по учету, анализу состава и движения контингента студентов факультета, формирование оперативных данных о деятельности факультета и кафедр.

Представим предметную область «Учебный процесс» как взаимодействие следующих сущностей: каждый «Студент» сдает экзамен или зачет по некоторому «Предмету» согласно учебному плану. В учебном процессе участвует «Преподаватель», который осуществляет чтение учебного курса и контроль знаний «Студента». В учебном процессе также участвует «Деканат», который организовывает работу «Преподавателя». Обучение «Студента» ведется в «Группе» совместно с его одногруппниками.

Следует отметить, что для каждой сущности устанавливается свой код - ключевой атрибут, однозначно характеризующий сущность. Например, обычный номер студента в группе не может выполнять роль ключа, поскольку для каждой группы эти номера могут повторяться. Для преподавателя атрибут Табельный номер нежелательно брать в качестве ключевого, поскольку все-таки возможно изменение табельного номера.

Для реализации дополнительных функций базы может потребоваться введение дополнительных атрибутов, например, номера зачетной книжки и домашнего телефона студента, домашнего адреса и домашнего телефона преподавателя, должности преподавателя, рабочей программы, даты сдачи экзамена (зачета) и т.д.

О преподавателях есть следующая информация:

* Название кафедры;
* Должность;
* Стаж работы;
* ФИО;
* Оклад;
* Специальность;
* Табельный номер;
* Расписание.

Они обеспечивают проведение занятий по некоторым предметам.

Также курируют группу, о которой известно:

* Курс;
* Факультет;
* Направление;
* Количество студентов;
* Название;
* Номера телефонов студентов и куратора.

О студентах будет известна следующая информация:

* ФИО;
* Номер телефона;
* Группа.

Они ходят на пары по разным предметам:

* Название;
* Вид пар (Лекция, семинар или лабораторная);
* День недели;
* Время пары;
* Номер аудитории;
* Группа;
* Преподаватель.

# **Подробный анализ предметной области**

С этой системой могут работать следующие группы пользователей:

- Сотрудники деканата.

Сущность – Дисциплины

Атрибуты:

* название;
* тип занятия;
* количество часов;
* вид контроля;

Сущность ­ Расписание

Атрибуты:

* название дисциплины;
* вид пар;
* день недели;
* время пары;
* номер аудитории;
* преподаватель;
* группа.

# **Разработка информационно-логической модели предметной области**

Связи:

* Сущности Преподаватели и Расписание связаны между собой связью «один ко многим».
* Сущности Группа и Студенты связаны связью «один ко многим».
* Сущности Группа и Расписание связаны связью «один ко многим».
* Сущности Дисциплины и Расписание связаны связью «один ко многим».

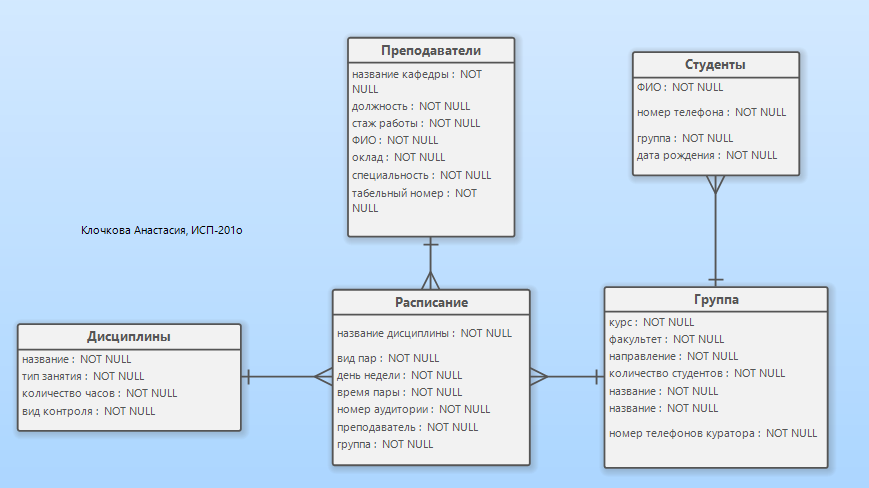


Рисунок 1 - Разработка информационно-логической модели

# **Разработка даталогической модели предметной области**

Сущность «Продукт» из информационно-логической модели, становится отношением «Product» с атрибутами, представленными в Таблице 1:

В сущностях Преподаватели, Студент, Расписание, Группа, Дисциплины выделили следующие типы данных, они представлены в таблицах 1, 2, 3, 4, 5.

Таблица 1 – типы данных в сущности Teacher

| Атрибут | Тип данных | Пояснение |
| --- | --- | --- |
| ID\_Teacher | INTEGER | Поле содержит только целочисленные значения |
| Department\_name | VARCHAR (30) | Поле может быть любого размера, не превышающего 30 символов, а также быть пустым |
| Work\_experience | VARCHAR (30) | Поле может быть любого размера, не превышающего 30 символов, а также быть пустым |
| Salary | INTEGER | Поле содержит только целочисленные значения |
| Specialty | VARCHAR (30) | Поле может быть любого размера, не превышающего 30 символов, а также быть пустым |
| Personal\_number | INTEGER | Поле содержит только целочисленные значения |
| FIO | VARCHAR (30) | Поле может быть любого размера, не превышающего 30 символов, а также быть пустым |
| Position | VARCHAR (30) | Поле может быть любого размера, не превышающего 30 символов, а также быть пустым |

Таблица 2 ­ типы данных в сущности Students

| Атрибут | Тип данных | Пояснение |
| --- | --- | --- |
| ID Students | INTEGER | Поле содержит только целочисленные значения |
| FIO | VARCHAR (30) | Поле может быть любого размера, не превышающего 30 символов, а также быть пустым |
| Phone\_number | VARCHAR (12) | Поле может быть любого размера, не превышающего 12 символов. |
| Group | VARCHAR (30) | Поле может быть любого размера, не превышающего 30 символов, а также быть пустым |
| Date\_of\_Birth | DateTime | Используется для хранения значений, содержащих как дату, так и время. |

Таблица 3 ­ типы данных в сущности Timetable

| Атрибут | Тип данных | Пояснение |
| --- | --- | --- |
| ID\_Subject | INTEGER | Поле содержит только целочисленные значения |
| Name | VARCHAR (30) | Поле может быть любого размера, не превышающего 30 символов |
| Day\_of\_week | VARCHAR (15) | Поле может быть любого размера, не превышающего 15 символов. |
| Audience\_number | VARCHAR (4) | Поле может быть любого размера, не превышающего 4 символов. |
| Kind\_of\_couples | VARCHAR (30) | Поле может быть любого размера, не превышающего 30 символов, а также быть пустым |

Таблица 4 – типы данных в сущности Group

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Пояснение |
| ID\_Group | INTEGER | Поле содержит только целочисленные значения |
| Course | INTEGER | Поле содержит только целочисленные значения |
| Faculty | VARCHAR (30) | Поле может быть любого размера, не превышающего 30 символов, а также быть пустым |
| Direction | VARCHAR (30) | Поле может быть любого размера, не превышающего 30 символов. |
| Number\_of\_students | INTEGER | Поле содержит только целочисленные значения |
| Name | VARCHAR (15) | Поле может быть любого размера, не превышающего 15 символов. |
| Phone \_number\_of\_the curator | VARCHAR (12) | Поле может быть любого размера, не превышающего 12 символов. |

Таблица 5 – типы данных в сущности Discipline

| Атрибут | Тип данных | Пояснение |
| --- | --- | --- |
| ID\_Discipline | INTEGER | Поле содержит только целочисленные значения |
| Name | VARCHAR (15) | Поле может быть любого размера, не превышающего 15 символов. |
| Class type | VARCHAR (30) | Поле может быть любого размера, не превышающего 30 символов, а также быть пустым |
| Number\_of\_hours | INTEGER | Поле содержит только целочисленные значения |
| Type\_of\_control | VARCHAR (15) | Поле может быть любого размера, не превышающего 15 символов. |

Получившаяся схема даталогической модели представлена на рисунке 2.

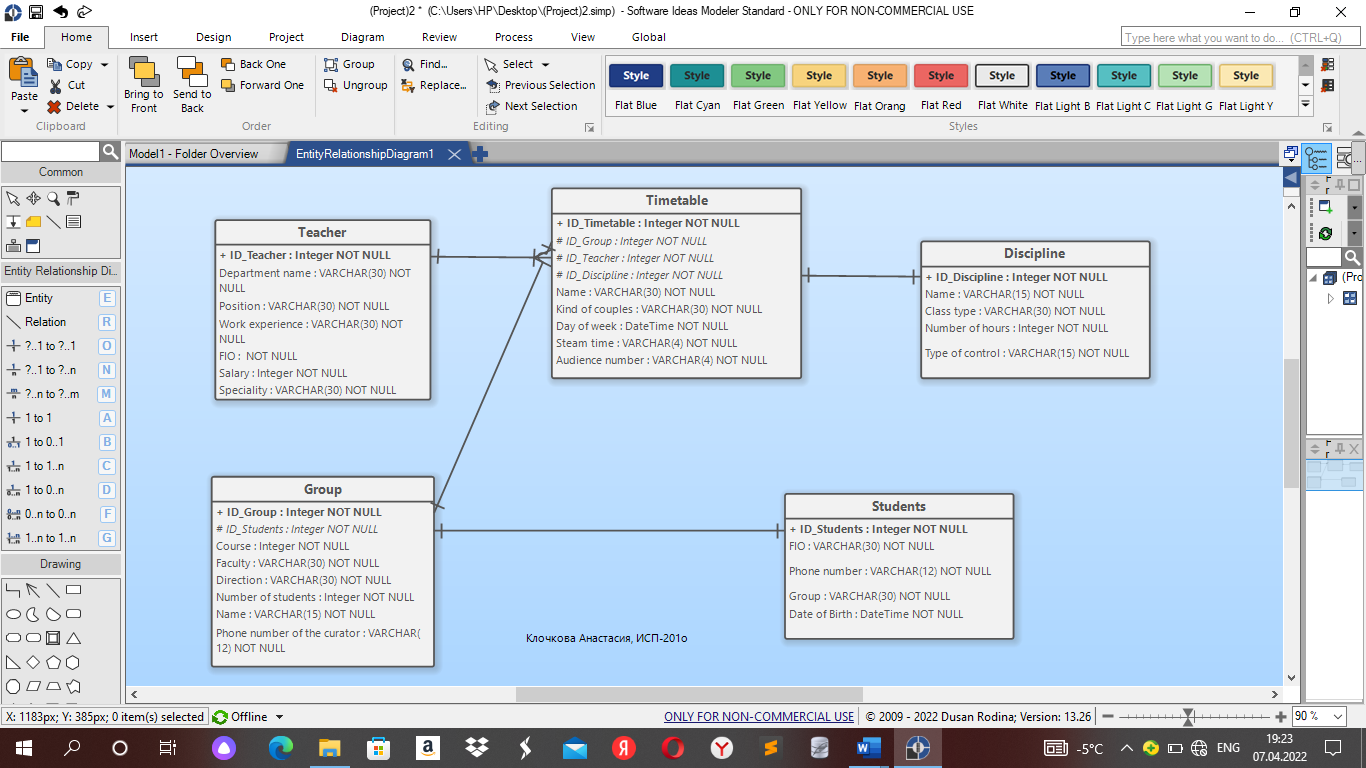


Рисунок 2 - Разработка даталогической модели

Для устранения из базы избыточности, которая потенциально может привести к логически ошибочным результатам выборки или изменения данных, мы создали две сущности – Specialist и Faculties, в которые поместили атрибуты. В первую – Department name, Position, Speciality. Во вторую – Faculty, Direction. Также установили связи с внешними ключами. Получившаяся схема даталогической модели представлена на рисунке 3.

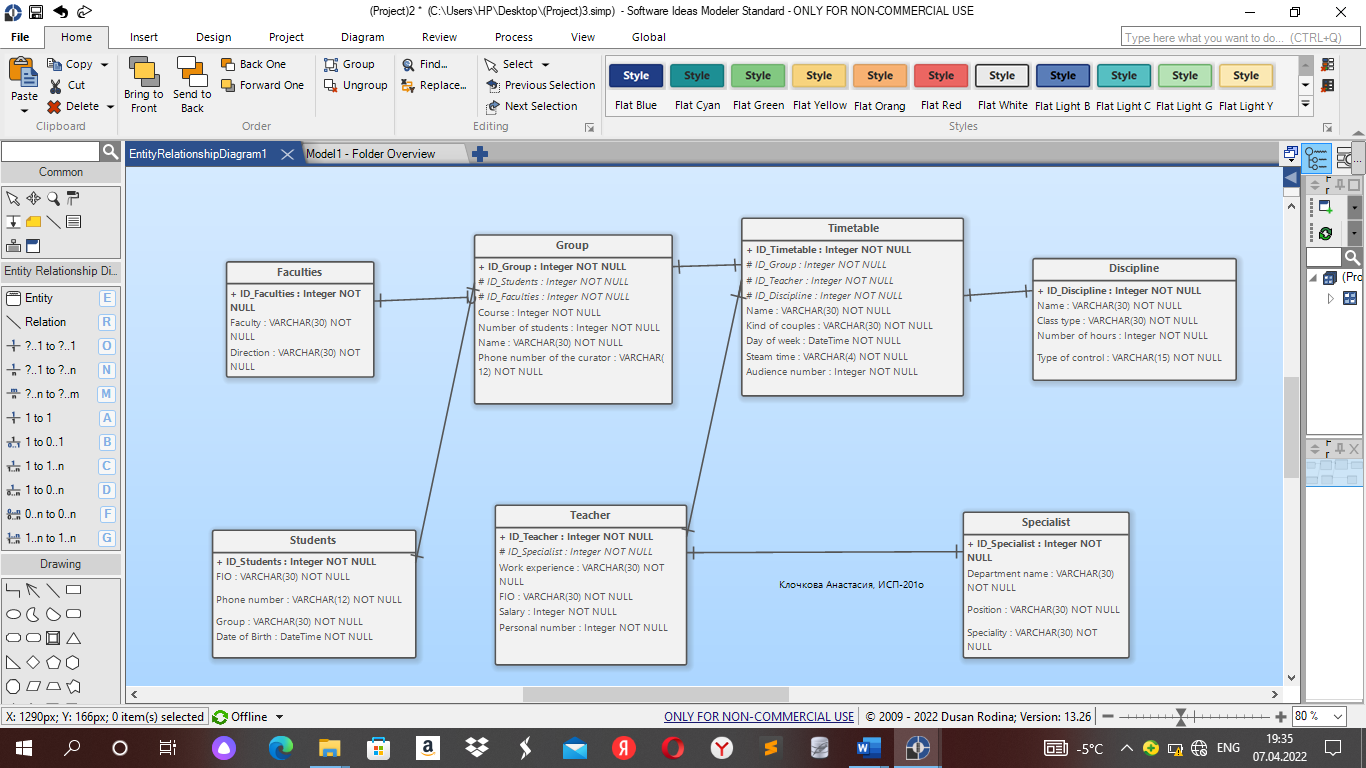


Рисунок 3 – даталогическая модель ПО, приведённая к 3НФ

# **Реализация базы данных**

Таблицу Specialist создали с помощью SQL кода

CREATE TABLE IF NOT EXISTS Specialist(ID INT PRIMARY KEY,

Department\_name VARCHAR(30) NOT NULL,

Position VARCHAR(30) NOT NULL,

Speciality VARCHAR(30) NOT NULL);

На основании следующих данных, заполнили таблицу Specialist.

Таблица 6 – таблица Specialist

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Department\_name | Position | Speciality |
| 1 | Кафедра физики | Преподаватель | Юрист |
| 2 | Кафедра истории | Преподаватель | Юрист |
| 3 | Кафедра философии | Доцент | Юрист |
| 4 | Кафедра литературы | Преподаватель | Юрист |
| 5 | Кафедра математики | Преподаватель | Юрист |
| 6 | Кафедра физики | Доцент | Юрист |
| 7 | Кафедра изо | Преподаватель | Юрист |
| 8 | Кафедра филологии | Преподаватель | Юрист |
| 9 | Кафедра высоких технологий | Доцент | Юрист |
| 10 | Кафедра физкультуры | Преподаватель | Юрист |

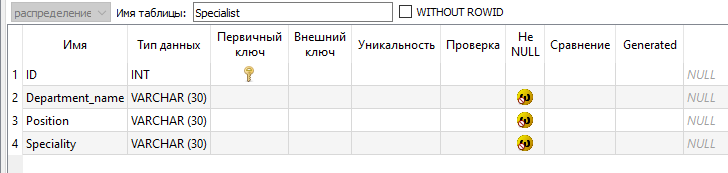


Рисунок 4 ¬ Структура таблицы Specialist

Заполнили с помощью кода

INSERT INTO Specialist VALUES (1,'Кафедра Физкульту-ры','Преподаватель', 'Юрист');

INSERT INTO Specialist VALUES (2,'Кафедра истории','Преподаватель', 'Юрист');

INSERT INTO Specialist VALUES (3,'Кафедра философии','Доцент', 'Юрист');

INSERT INTO Specialist VALUES (4,'Кафедра литературы','Преподаватель', 'Юрист');

INSERT INTO Specialist VALUES (5,'Кафедра математики','Преподаватель', 'Юрист');

INSERT INTO Specialist VALUES (6,'Кафедра физики','Доцент', 'Юрист');

INSERT INTO Specialist VALUES (7,'Кафедра изо','Преподаватель', 'Юрист');

INSERT INTO Specialist VALUES (8,'Кафедра филологии','Преподаватель', 'Юрист');

INSERT INTO Specialist VALUES (9,'Кафедра высоких технологий','Доцент', 'Юрист');

INSERT INTO Specialist VALUES (10,'Кафедра Физкультуры','Преподаватель', 'Юрист');

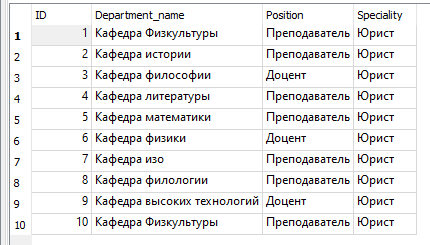


Рисунок 5 ­ Заполненная таблица Specialist

Создали таблицу Teacher c помощью кода SQL.

CREATE TABLE IF NOT EXISTS Teacher(ID INT PRIMARY KEY,

ID\_Specialist INT REFERENCES

Specialist(ID) ON DELETE CASCADE ON

UPDATE CASCADE,

Work\_experience VARCHAR(3) NOT NULL,

FIO VARCHAR(35) NOT NULL,

Salary INT NOT NULL,

Personal\_number INT NOT NULL);

На основании следующих данных, заполнили таблицу Teacher.

Таблица 7 – таблица Teacher

| ID | ID\_Speciality | Work\_experience | FIO | Salary | Personal\_number |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | 3 года | Иванова Наталья Владимировна | 13000 | 3456 |
| 2 | 2 | 12 лет | Петрова Жанна Адольфовна | 35000 | 4765 |
| 3 | 3 | 4 года | Крысова Иннеса Валентиновна | 11000 | 8726 |
| 4 | 4 | 20 лет | Красномызник Янина Станиславовна | 8000 | 8916 |
| 5 | 5 | 4 года | Стрельникова Нателла Наумовна | 7000 | 5456 |
| 6 | 6 | 9 лет | Шершанский Яков Осипович | 65000 | 9376 |
| 7 | 7 | 8 лет | Вишневский Гвидон Соломонович | 5000 | 3936 |
| 8 | 8 | 3 года | Восьмиглазова Татьяна Всеволодовна | 33000 | 2746 |
| 9 | 9 | 5 лет | Сапогов Ричард Генрихович | 65000 | 7346 |
| 10 | 10 | 11 лет | Катамаранов Игорь Анатольевич | 13000 | 7036 |

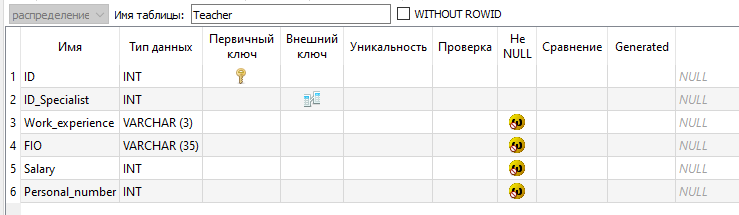


Рисунок 6 – Структура таблицы Teacher

Заполнили таблицу с помощью кода

INSERT INTO Teacher Values (1, 1,'3 года','Иванова Наталья Владимиров-на',13000,3456);

INSERT INTO Teacher Values (2,2,'12 лет','Петрова Жанна Адольфов-на',35000,4765);

INSERT INTO Teacher Values (3,3,'4 года','Крысова Иннеса Валентиновна',11000,8726);

INSERT INTO Teacher Values (4,4,'20 лет','Красномызник Янина Станисла-вовна',8000,8916);

INSERT INTO Teacher Values (5,5,'4 года','Стрельникова Нателла Наумовна',7000,5456);

INSERT INTO Teacher Values (6,6,'9 лет','Шершанский Яков Осипович',65000,9376);

INSERT INTO Teacher Values (7,7,'8 лет','Вишневский Гвидон Соломонович',5000,3936);

INSERT INTO Teacher Values (8,8,'3 года','Восьмиглазова Татьяна Всево-лодовна',33000,2746);

INSERT INTO Teacher Values (9,9,'5 лет','Сапогов Ричард Генрихович',65000,7346);

INSERT INTO Teacher Values (10,10,'11 лет','Катамаранов Игорь Анатолье-вич',13000,7036);

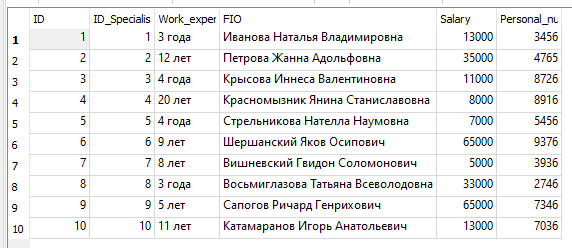


Рисунок 7 – Заполненная таблица Teacher

Таблицу Students создали с помощью SQL кода

CREATE TABLE IF NOT EXISTS Students(ID INT PRIMARY KEY,

FIO VARCHAR(30) NOT NULL,

Phone\_Number VARCHAR(12) NOT NULL,

Groupss VARCHAR(30) NOT NULL,

Date\_of\_birth DateTime NOT NULL);

На основании следующих данных, заполнили таблицу Students.

Таблица 8 – таблица Students

| ID | FIO | Phone\_Number | Groups | Date\_of\_birth |
| --- | --- | --- | --- | --- |
| 1 | Абрамов Александр Николаевич | +78956142155 | ПБ-201 | 20.02.2001 |
| 2 | Сатанитский Константин Глебович | +79456458922 | ПБ-202 | 15.08.2002 |
| 3 | Кишмишева Алёна Константиновна | +79456458256 | ПБ-201 | 12.04.2004 |
| 4 | Тазиков Джамал Шимонович | +79457988922 | ПБ-203 | 28.09.2003 |
| 5 | Цукерник Кондрат Софронович | +79456458312 | ПБ-201 | 27.01.2000 |
| 6 | Клёнова Мелисса Курбатовна | +79456458356 | ПБ-204 | 25.07.2005 |
| 7 | Брежнев Саймон Владимирович | +79456461822 | ПБ-202 | 21.08.1999 |
| 8 | Сильвесторов Павел Геннадиевич | +7945645894 | ПБ-204 | 14.05.2002 |
| 9 | Шизоид Сергей Алексеевич | +7946548912 | ПБ-203 | 31.02.2004 |
| 10 | Попловский Кшиштоф Борисович | +7954565241 | ПБ-201 | 12.05.2003 |



Рисунок 8 ¬ Структура таблицы Students

Заполнили таблицу с помощью кода

INSERT INTO Students VALUES

(1,'Абрамов Александр Николаевич','+78956142155', 'ПБ-201', '20.02.2001');

INSERT INTO Students VALUES

(2,'Сатанитский Константин Глебович','+79456458922','ПБ-202', '15.08.2002');

INSERT INTO Students VALUES

(3,'Кишмишева Алёна Константиновна','+79456458256','ПБ-201', '12.04.2004');

INSERT INTO Students VALUES

(4,'Тазиков Джамал Шимонович','+79457988922','ПБ-203', '28.09.2003');

INSERT INTO Students VALUES

(5,'Цукерник Кондрат Софронович','+79456458312', 'ПБ-201', '27.01.2000');

INSERT INTO Students VALUES

(6,'Клёнова Мелисса Курбатовна','+79468558922', 'ПБ-204', '25.07.2005');

INSERT INTO Students VALUES

(7,'Брежнев Саймон Владимирович','+79456458356', 'ПБ-202', '21.08.1999');

INSERT INTO Students VALUES

(8,'Сильвесторов Павел Геннадиевич','+79456461822','ПБ-204', '14.05.2002' );

INSERT INTO Students VALUES

(9,'Шизоид Сергей Алексеевич','+79456458874', 'ПБ-203', '31.02.2004');

INSERT INTO Students VALUES

(10,'Попловский Кшиштоф Борисович ','+79456458922', 'ПБ-201', '12.06.2003');

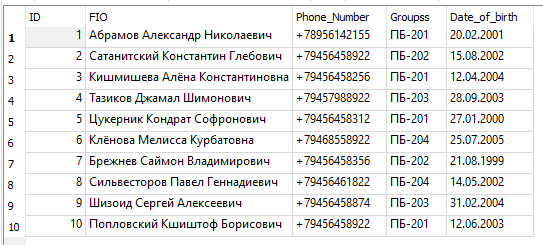


Рисунок 9 ¬ Заполненная таблица Students

Таблицу Faculties создали с помощью SQL кода

CREATE TABLE IF NOT EXISTS Faculties(ID INT PRIMARY KEY,

Faculty VARCHAR(30) NOT NULL,

Direction VARCHAR(30) NOT NULL);

На основании следующих данных, заполнили таблицу Faculties.

Таблица 9 – таблица Faculties

| ID | Faculty | Direction |
| --- | --- | --- |
| 1 | СПК | ИТ |
| 22 | ИСП | ИТ |
| 33 | ИСП | ПБ |
| 44 | СПУ | ПБ |
| 55 | ИСП | ИТ |
| 66 | СПК | ИТ |
| 77 | СПК | ИТ |
| 88 | СПК | ИТ |
| 99 | СПК | ИТ |
| 11 | ИСП | ПБ |

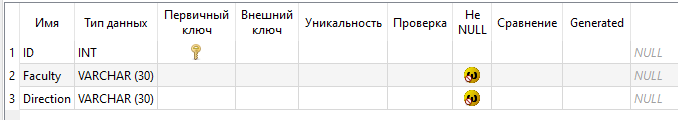


Рисунок 10 ¬ Структура таблицы Faculties

Заполнили с помощью кода

INSERT INTO Faculties VALUES (1,'спк','ИТ');

INSERT INTO Faculties VALUES (22,'ИСП','ИТ');

INSERT INTO Faculties VALUES (33,'ИСП','ИТ');

INSERT INTO Faculties VALUES (44,'спу','ИТ');

INSERT INTO Faculties VALUES (55,'ИСП','ИТ');

INSERT INTO Faculties VALUES (66,'Ипа','ИТ');

INSERT INTO Faculties VALUES (77,'Исп','ИТ');

INSERT INTO Faculties VALUES (88,'ИСП','ИТ');

INSERT INTO Faculties VALUES (99,'ИСП','ИТ');

INSERT INTO Faculties VALUES (11,'ИСП','ИТ');

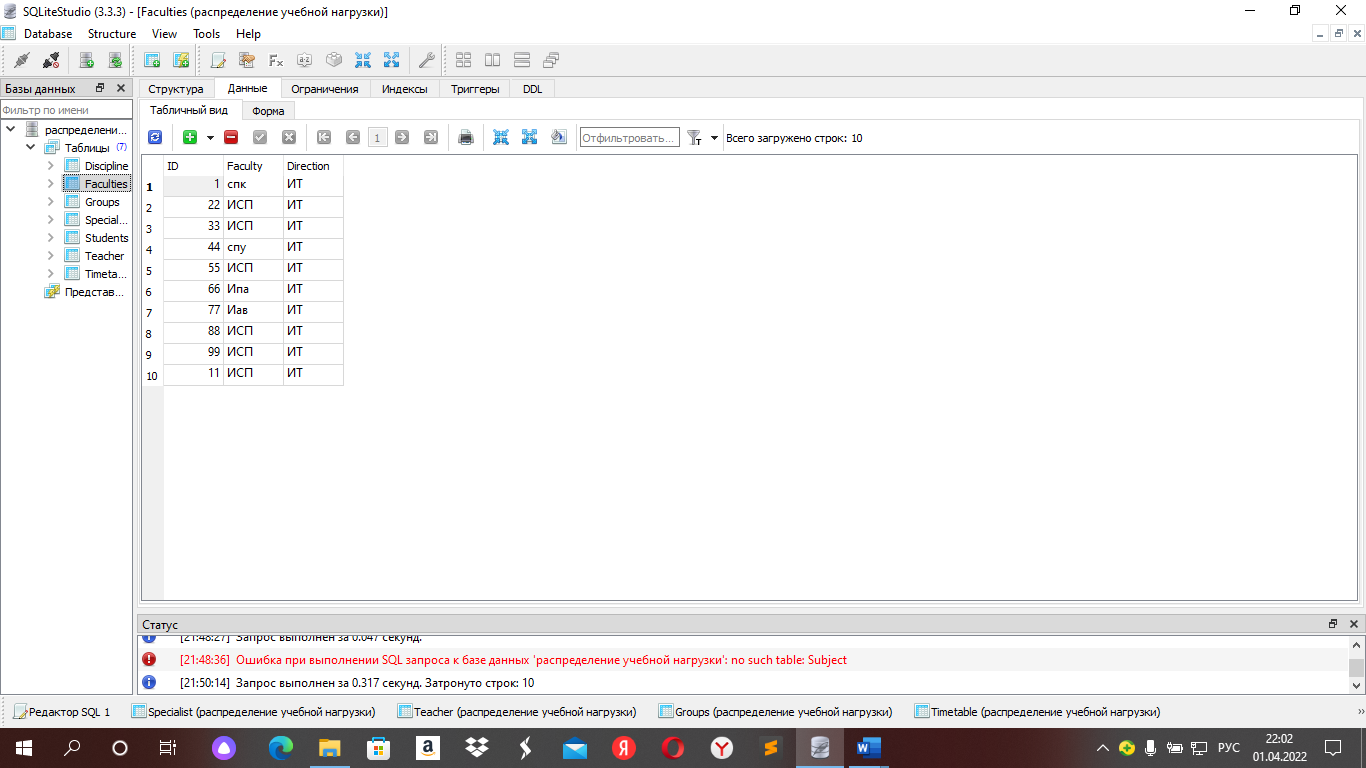


Рисунок 11 ¬ Заполненная таблица Faculties

Создали таблицу Groups

CREATE TABLE IF NOT EXISTS Groups(ID INT PRIMARY KEY,

ID\_Students INT REFERENCES

Students(ID) ON DELETE CASCADE ON

UPDATE CASCADE,

ID\_Faculties INT REFERENCES

Faculties(ID) ON DELETE CASCADE ON

UPDATE CASCADE,

Course INT NOT NULL,

Number\_Of\_Students INT NOT NULL,

Name VARCHAR(15) NOT NULL,

Phone\_Number\_Of\_The\_Curator VARCHAR(12) NOT NULL);

На основании следующих данных, заполнили таблицу Groups

Таблица 10 – таблица Groups

| ID | ID\_Students | ID\_Faculties | Course | Number\_of\_students | Name | Phone\_Number |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 4 | 21 | ПБ-201 | +79214566596 |
| 2 | 2 | 22 | 1 | 15 | ПБ-201с | +79854555455 |
| 3 | 3 | 33 | 2 | 29 | бГРАД-182 | +79555515415 |
| 4 | 4 | 44 | 1 | 30 | бГРАД-183 | +79135455585 |
| 5 | 5 | 55 | 2 | 20 | ПБ-201 | +79545558325 |
| 6 | 6 | 66 | 3 | 25 | ПБ-201с | +79134682464 |
| 7 | 7 | 77 | 4 | 31 | ПБ-205 | +79134645265 |
| 8 | 8 | 88 | 4 | 21 | ПБ-201 | +79813465354 |
| 9 | 9 | 99 | 1 | 25 | ПБ-201с | +79234581365 |
| 10 | 10 | 11 | 1 | 19 | бГРАД-182 | +79461325864 |



Рисунок 12 ­ Структура таблицы Groups

Заполнили с помощью код

INSERT INTO Groups VALUES

(1,1,1,4,21,'ПБ-201','+79214654213');

INSERT INTO Groups VALUES

(2,2,22,1,15,'ПБ-202с','+79214654278');

INSERT INTO Groups VALUES

(3,3,33,2,29,'бГРАД-182','+79214978213');

INSERT INTO Groups VALUES

(4,4,44,1,30,'бГРАД-183','+79621978213');

INSERT INTO Groups VALUES

(5,5,55,2,20,'ПБ-201','+79214981213');

INSERT INTO Groups VALUES

(6,6,66,3,25,'ПБ-201с','+79214655618');

INSERT INTO Groups VALUES

(7,7,77,4,31,'ПБ-205','+79289054213');

INSERT INTO Groups VALUES

(8,8,88,4,21,'ПБ-201','+79212354213');

INSERT INTO Groups VALUES

(9,9,99,1,25,'ПБ-201с','+79219785618');

INSERT INTO Groups VALUES

(10,10,11,1,19,'бГРАД-182','+79346978213');

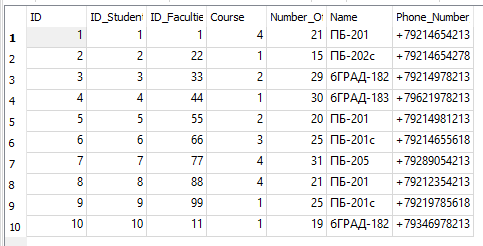


Рисунок 13 ¬ Заполненная таблица Groups

Таблицу Discipline создали с помощью SQL кода

CREATE TABLE IF NOT EXISTS Discipline(ID INT PRIMARY KEY,

Name VARCHAR(30) NOT NULL,

Class\_type VARCHAR(30) NOT NULL,

Number\_Of\_Hours VARCHAR(30) NOT NULL,

Type\_Of\_Control VARCHAR(30) NOT NULL);

На основании следующих данных, заполнили таблицу Discipline.

Таблица 11 – таблица Discipline

| ID | Name | Class\_type | Number\_Of\_Hours | Type\_Of\_Control |
| --- | --- | --- | --- | --- |
| 1 | ПБ-193 | Практика | 36 | Экзамен |
| 2 | ПБ-201 | Практика | 40 | Зачёт с оценкой |
| 3 | ПБ-202 | Практика | 36 | Экзамен |
| 4 | ПБ-192 | Лекция | 36 | Курсовая работа |
| 5 | ПБ-191 | Практика | 36 | Экзамен |
| 6 | ПБ-202 | Лекция | 36 | Зачёт с оценкой |
| 7 | ПБ-203 | Практика | 36 | Экзамен |
| 8 | ПБ-204 | Практика | 36 | Курсовая работа |
| 9 | ПБ-193 | Лекция | 36 | Экзамен |
| 10 | ПБ-192 | Практика | 36 | Экзамен |

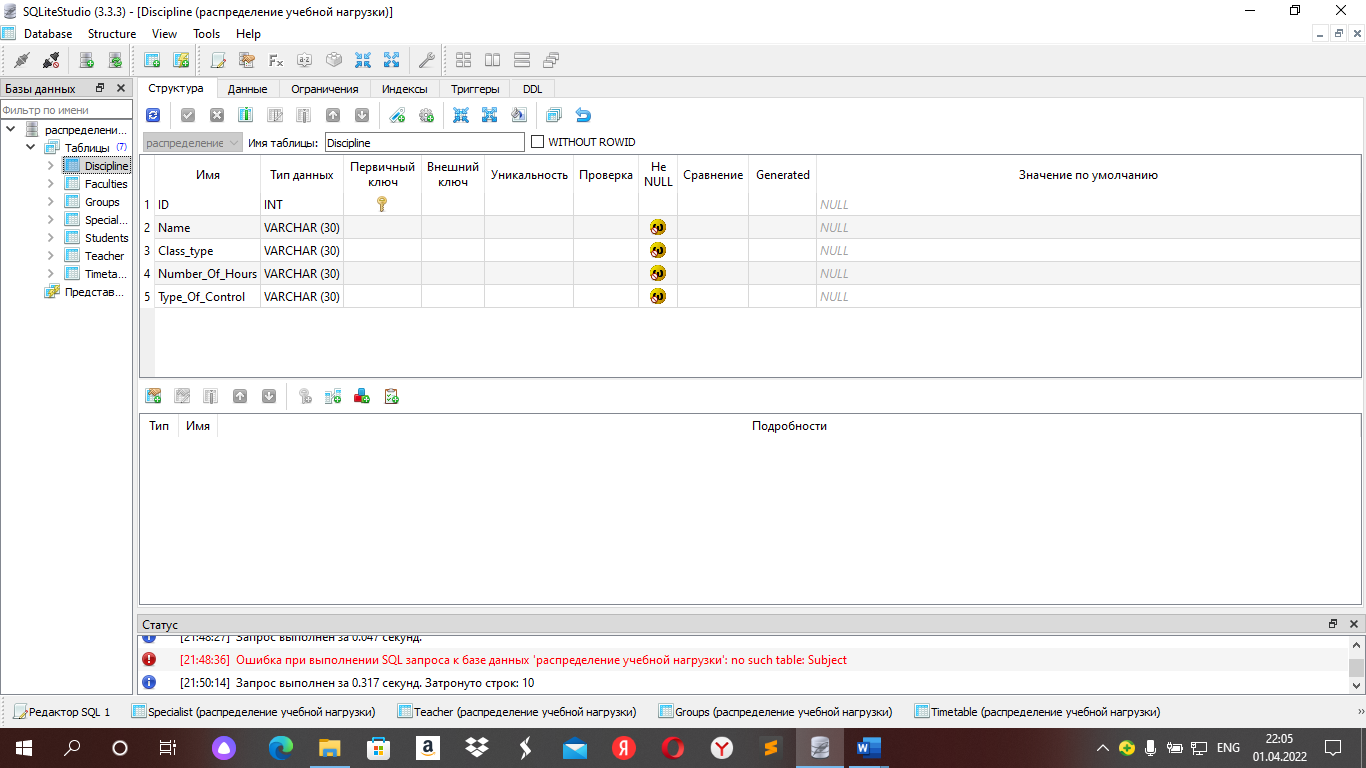


Рисунок 14 ¬ Структура таблицы Discipline

Заполнили с помощью кода

INSERT INTO Discipline VALUES (1,'ПБ-193','Практика',36,'экзамен');

INSERT INTO Discipline VALUES (2,'ПБ-201','Практика',40,'зачёт с оценкой');

INSERT INTO Discipline VALUES (3,'ПБ-202','Практика',36,'экзамен');

INSERT INTO Discipline VALUES (4,'ПБ-192','Лекция',36,'курсовая работа');

INSERT INTO Discipline VALUES (5,'ПБ-191','Практика',36,'экзамен');

INSERT INTO Discipline VALUES (6,'ПБ-202','Лекция',36,'зачёт с оценкой');

INSERT INTO Discipline VALUES (7,'ПБ-203','Практика',36,'экзамен');

INSERT INTO Discipline VALUES (8,'ПБ-204','Лекция',36,'экзамен');

INSERT INTO Discipline VALUES (9,'ПБ-193','Лекция',36,'курсовая работа');

INSERT INTO Discipline VALUES (10,'ПБ-192','Практика',36,'экзамен');

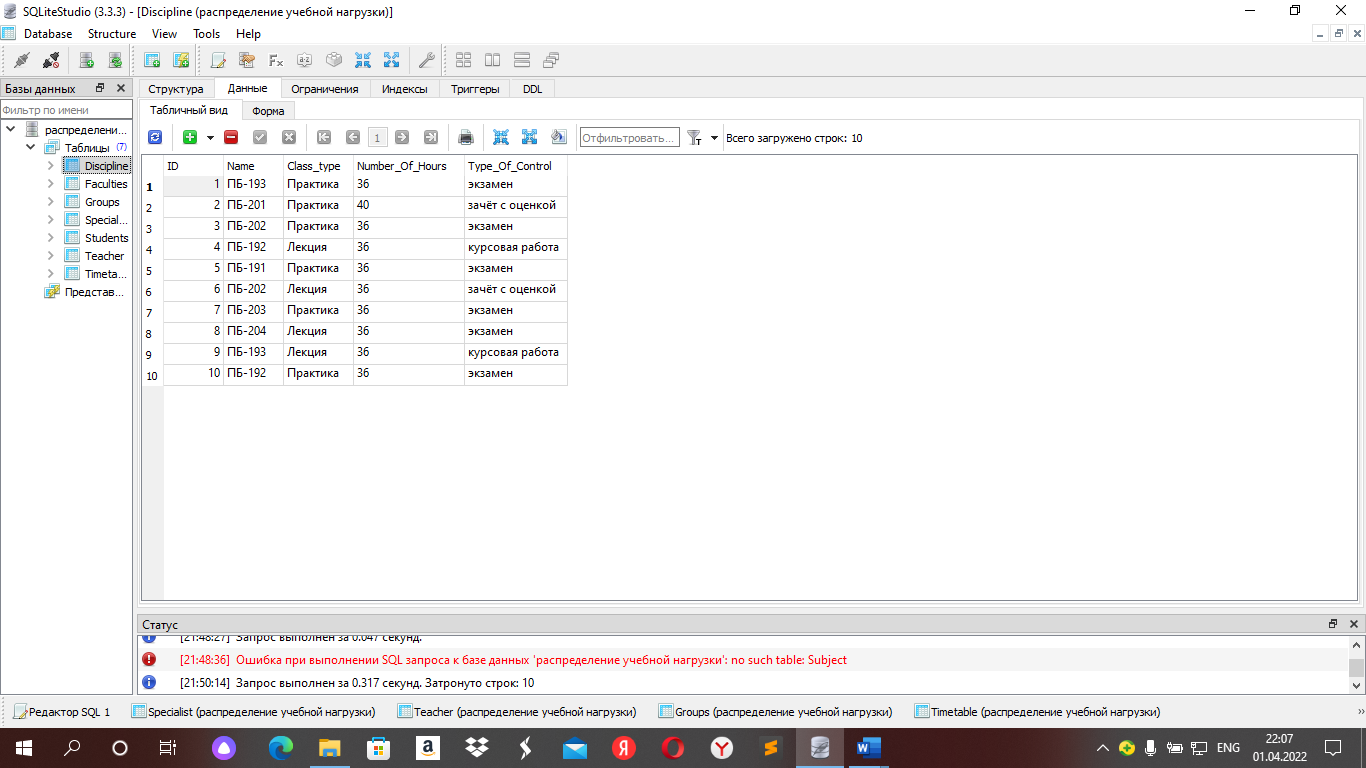


Рисунок 15 ¬ Заполненная таблица Discipline

Создали таблицу Timetable

CREATE TABLE IF NOT EXISTS Timetable(ID INT PRIMARY KEY NOT NULL,

ID\_Groups INT REFERENCES Groups(ID)

ON DELETE CASCADE ON UPDATE

CASCADE,

ID\_Teacher INT REFERENCES Teacher(ID)

ON DELETE CASCADE ON UPDATE

CASCADE,

ID\_Discipline INT REFERENCES Discipline(ID)

ON DELETE CASCADE ON UPDATE

CASCADE,

NAME VARCHAR(30) NOT NULL,

Kind\_Of\_Couples VARCHAR(50) NOT NULL,

Day\_Of\_Week VARCHAR(15) NOT NULL,

Steam\_Time DATETIME NOT NULL,

Audience\_Number VARCHAR(4) NOT NULL);

На основании следующих данных, заполнили таблицу Timetable.

Таблица 12 – таблица Timetable

| ID | ID\_Groups | ID\_Teacher | ID\_Discipline | NAME | Kind\_of\_couples | Day\_of\_Week | Steam\_time | Audience\_number |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | ПБ-193 | Практика | Вторник | 14:20 | 6270 |
| 2 | 2 | 2 | 2 | ИСП-202с | Практика | Среда | 8:30 | 3270 |
| 3 | 3 | 3 | 3 | бГРАД-181 | Лекция | Понедельник | 16:05 | 6252 |
| 4 | 4 | 4 | 4 | ПБ-192 | Практика | Четверг | 17:20 | 5270 |
| 5 | 5 | 5 | 5 | ИСП-203 | Практика | Понедельник | 12:50 | 7270 |
| 6 | 6 | 6 | 6 | бГРАД-193 | Практика | Пятница | 11:25 | 2317 |
| 7 | 7 | 7 | 7 | ПБ-193 | Лекция | Суббота | 19:20 | 7561 |
| 8 | 8 | 8 | 8 | ИСП-202с | Лекция | Вторник | 14:20 | 6421 |
| 9 | 9 | 9 | 9 | бГРАД-181 | Практика | Вторник | 14:20 | 6270 |
| 10 | 10 | 10 | 10 | ПБ-193 | Практика | Вторник | 19:20 | 6271 |

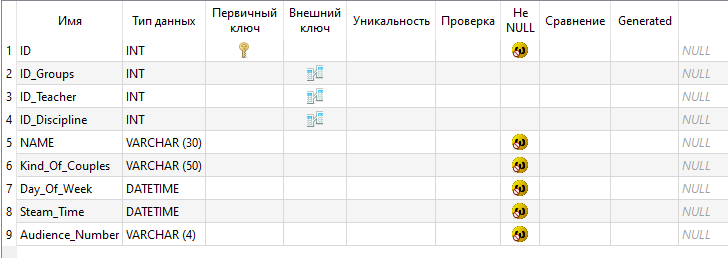


Рисунок 16 ­ Структура таблицы Timetable

Заполнили с помощью кода

INSERT INTO Timetable VALUES (1,1,1,1,'ПБ-193','Практика','Вторник','14:20','6270');

INSERT INTO Timetable VALUES (2,2,2,2,'ИСП-202с','Практика','Среда','8:30','3270');

INSERT INTO Timetable VALUES (3,3,3,3,'бГРАД-181','Лекция','Понедельник','16:05','6252');

INSERT INTO Timetable VALUES (4,4,4,4,'ПБ-192','Практика','Четверг','17:20','5270');

INSERT INTO Timetable VALUES (5,5,5,5,'ИСП-203','Практика','Понедельник','12:50','7270');

INSERT INTO Timetable VALUES (6,6,6,6,'бГРАД-193','Практика','Пятница','11:25','2317');

INSERT INTO Timetable VALUES (7,7,7,7,'ПБ-193','Лекция','Суббота','19:20','7561');

INSERT INTO Timetable VALUES (8,8,8,8,'ИСП-202с','Лекция','Вторник','14:20','6421');

INSERT INTO Timetable VALUES (9,9,9,9,'бГРАД-181','Практика','Вторник','14:20','6270');

INSERT INTO Timetable VALUES (10,10,10,10,'ПБ-193','Практика','Вторник','19:20','6271');

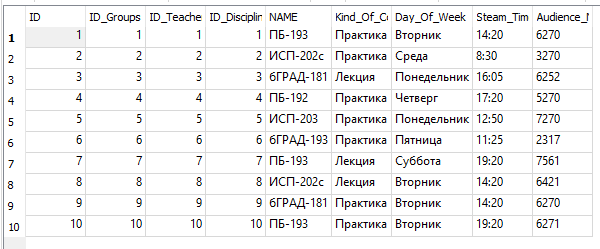


Рисунок 17 ­ Заполненная таблица Timetable

# **Создание отчётов**

В предметной области создаётся некоторое количество отчётов. Сформулируем их в данной лабораторной работе. Некоторые из них будут сделаны на основе задач из лабораторной №1.

Необходимо вывести ФИО у студентов. Для этого мы используем следующий код:

SELECT FIO FROM Students

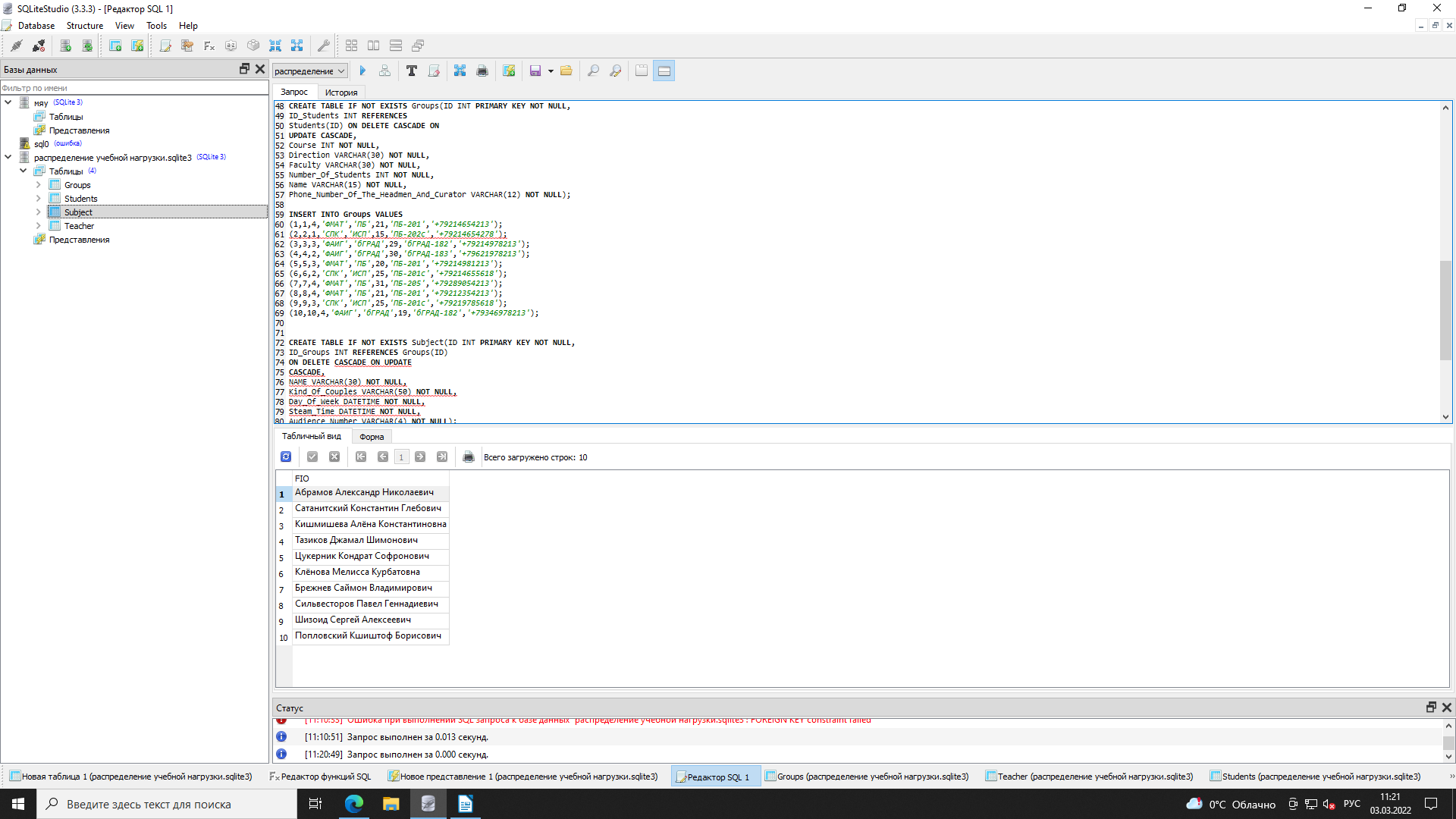


Рисунок 18 ­ ФИО студентов

Необходимо вывести группировку количества преподавателей по стажу работы. Для этого мы используем следующий код:

SELECT FIO, Work\_experience, COUNT(ID) FROM Teacher

GROUP BY Work\_experience

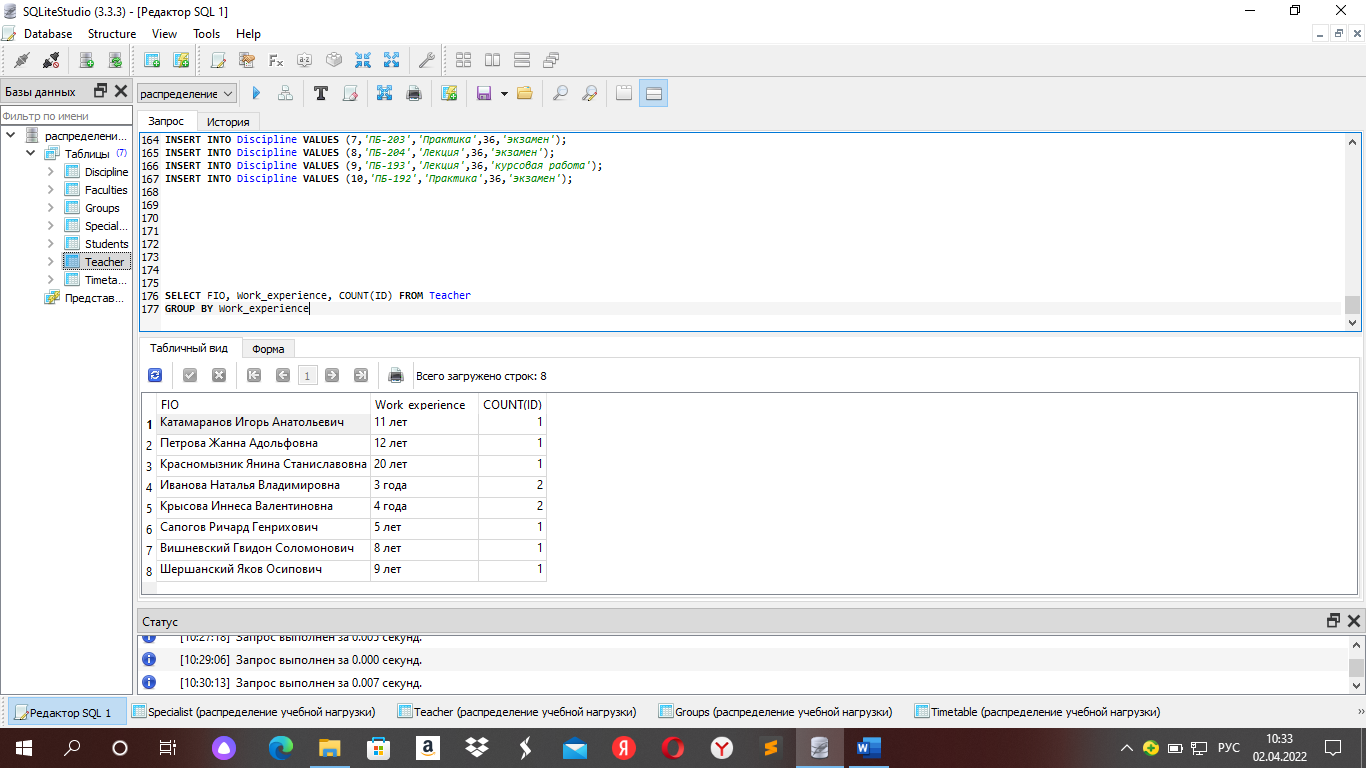
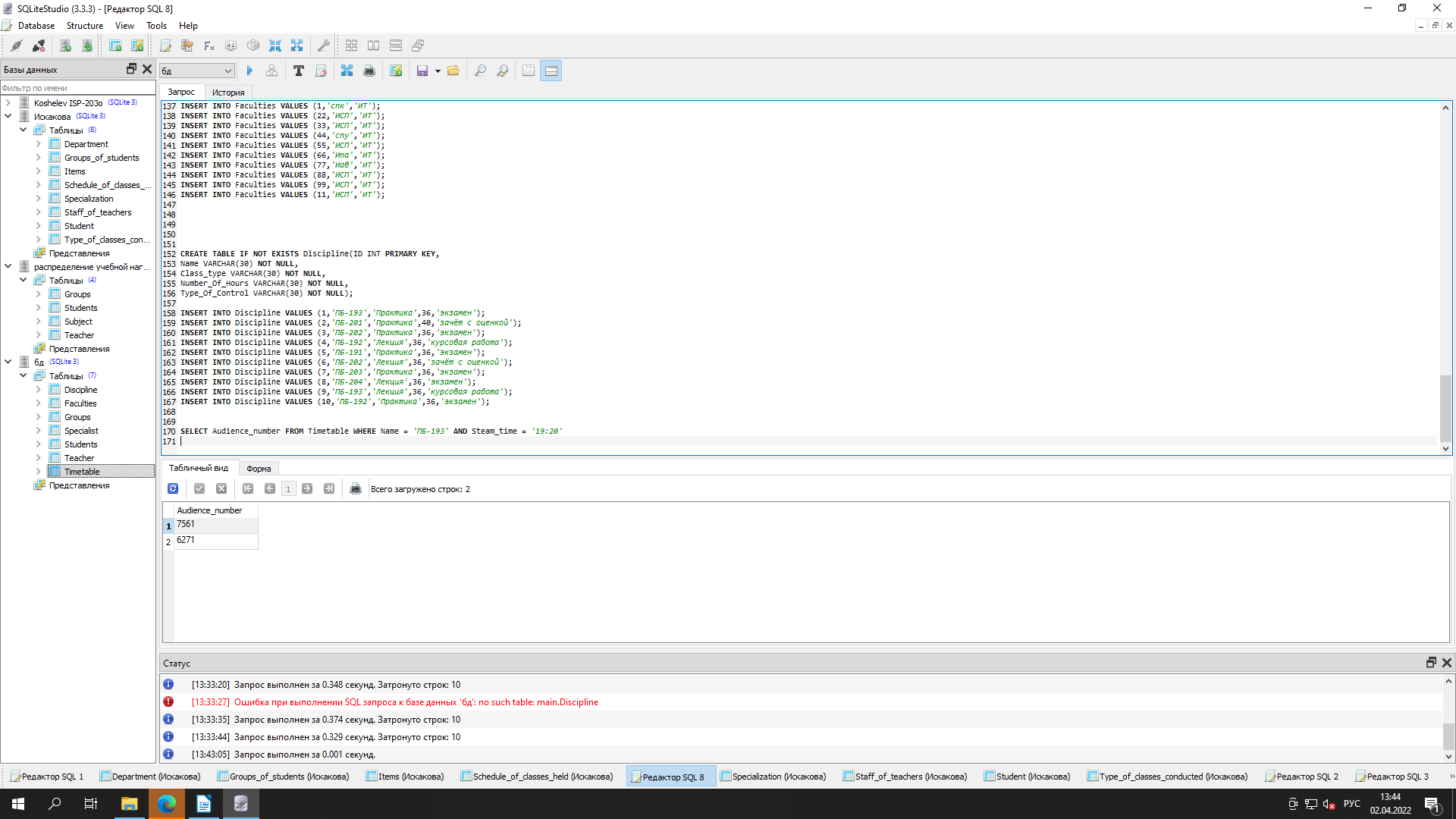


Рисунок 19 – Группировка количества преподавателей по стажу работы.

Необходимо вывести аудиторию группы ПБ-193 в 19:20. Для этого мы используем следующий код:

SELECT Audience\_number FROM Timetable WHERE Name = 'ПБ-193' AND Steam\_time = '19:20'

Рисунок 20 ­ аудиторию у объекта, где его имя равно ПБ-193

Необходимо вывести ФИО учителей, у которых оклад больше 17000. Для этого мы используем следующий код:

SELECT FIO FROM Teacher WHERE Salary > '17,000'

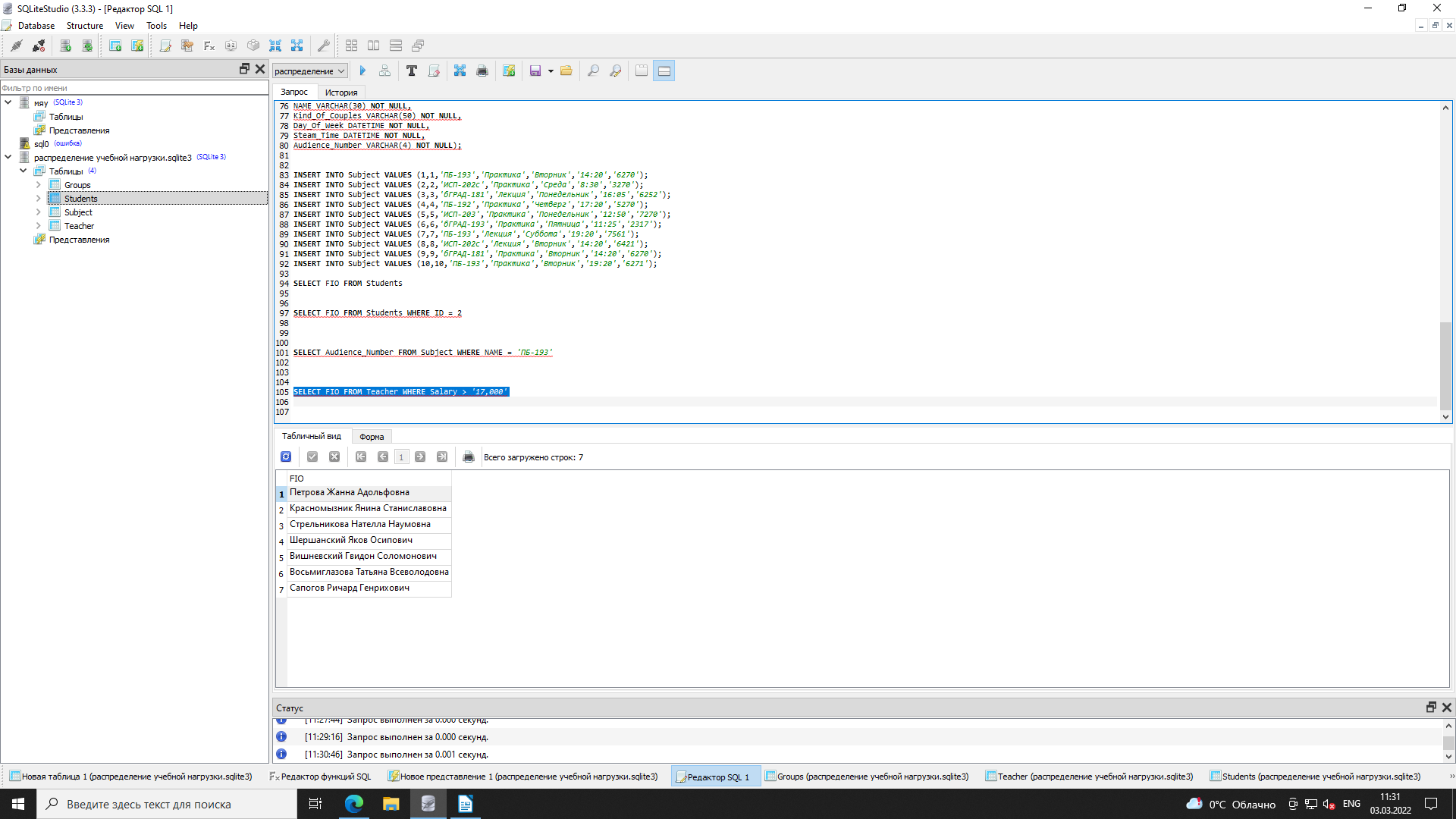


Рисунок 21 ­ ФИО учителей, у которых оклад больше 17000

Необходимо вывести факультет группы, 4 курса. Для этого мы используем следующий код:

SELECT Faculty FROM Groups WHERE Course = 4

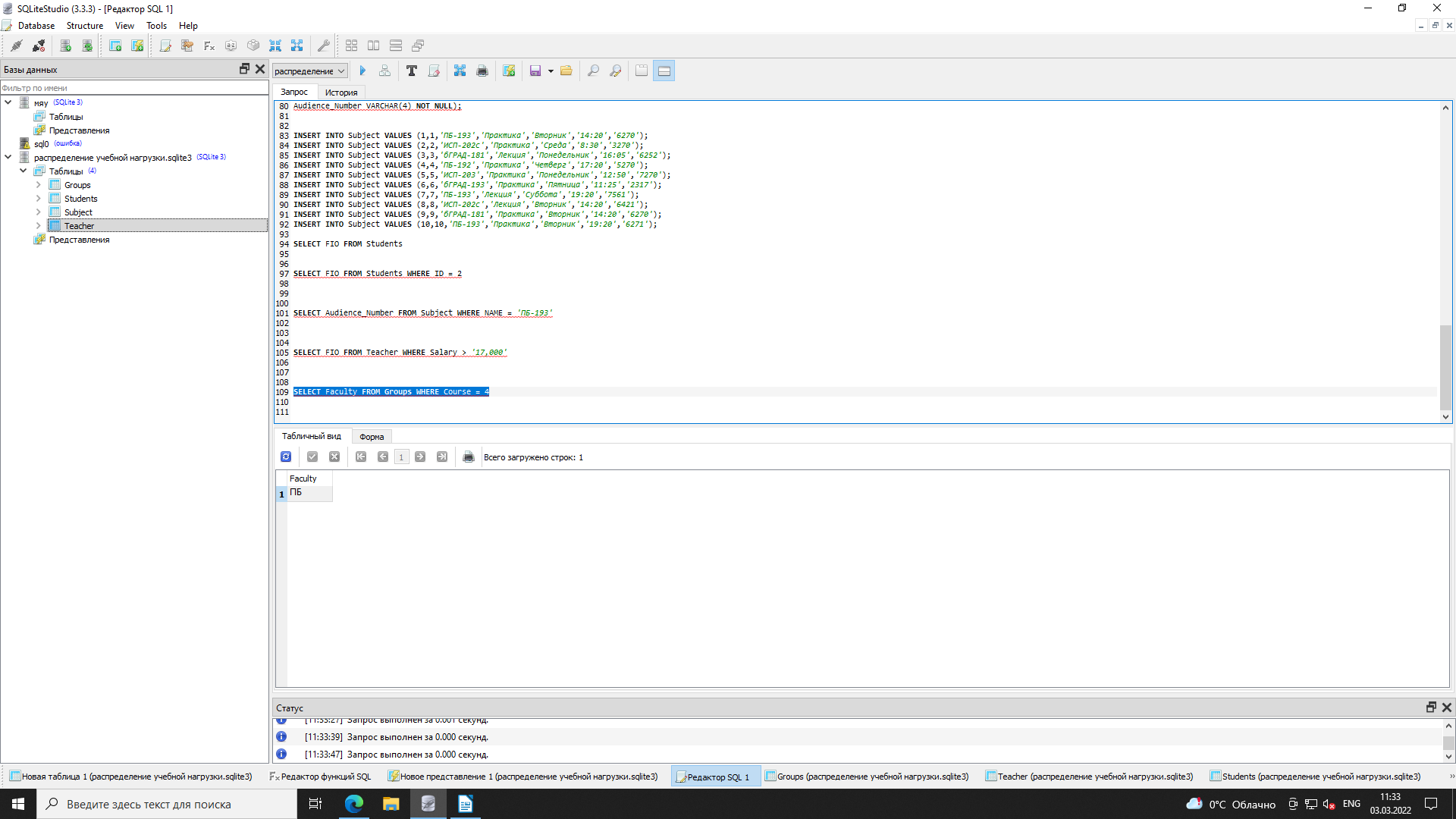


Рисунок 22 ­ факультет группы, у которого курс равен 4

Необходимо вывести дату рождения и номер телефона студента по имени Шизоид Сергей Алексеевич. Для этого мы используем следующий код:

SELECT Date\_of\_birth, Phone\_Number FROM Students WHERE FIO = 'Шизоид Сергей Алексеевич'

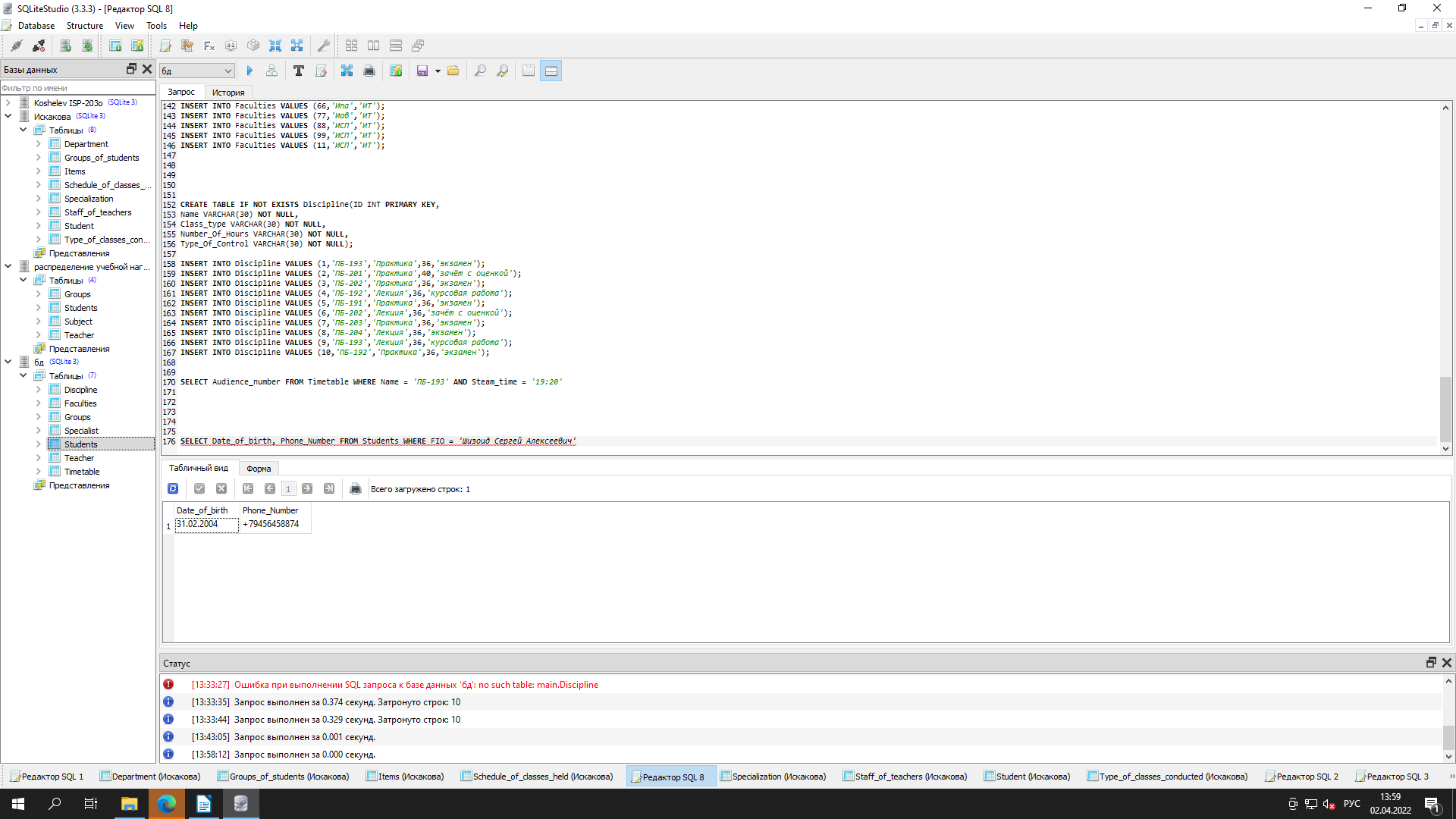
****

Рисунок 23 ­ Дата рождения и номер телефона студента по имени Шизоид Сергей Алексеевич

Необходимо удалить студента по имени Абрамов Александр Николаевич. Для этого мы используем следующий код:

DELETE FROM Students WHERE FIO = 'Абрамов Александр Николаевич'

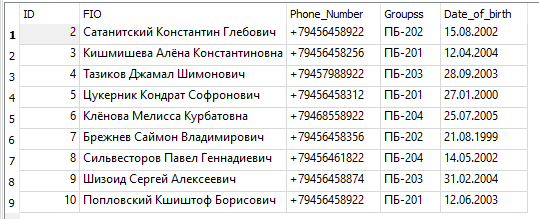


Рисунок 24 ­ Удаление студента по имени Абрамов Александр Николаевич

Необходимо обновить данные о студенте с ID равным 2. Для этого мы используем следующий код:

UPDATE Students SET Phone\_Number = '+79042114699', Groupss = 'ПБ-205' WHERE ID = 2

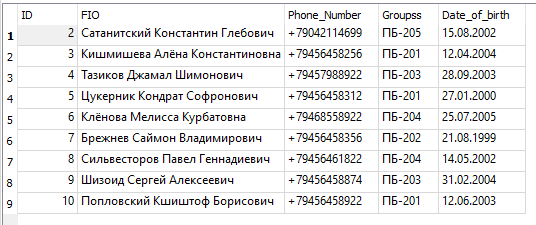


Рисунок 25 ­ Обновление данных о студенте с ID равным 2

# **Хранимые процедуры**

Создаем скалярную функцию «Teacher\_add» с аргументами: ID, Departament\_Name, Position, Work\_experience, FIO, Salary, Speciality, Personnel\_Number. Реализация на языке SQL:

INSERT INTO Teacher VALUES (@ID, @Departament\_Name, @Position, @Work\_experience, @FIO, @Salary, @Speciality, @Personal\_Number);

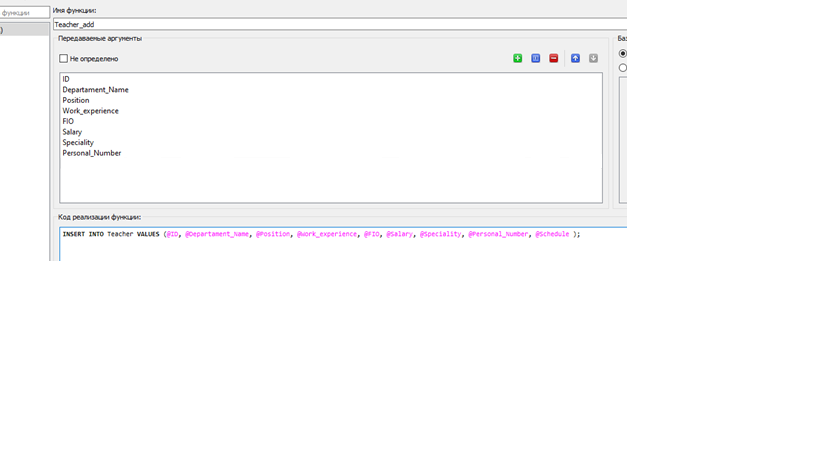


Рисунок 24 – Скалярная функция «Teacher\_add»

Использование функции «Teacher\_add»:

SELECT Teacher\_add (34,'Кафедра Физики','Преподаватель','4 года','Крысова Иннеса Валентиновна',11000,'Юрист','8726');

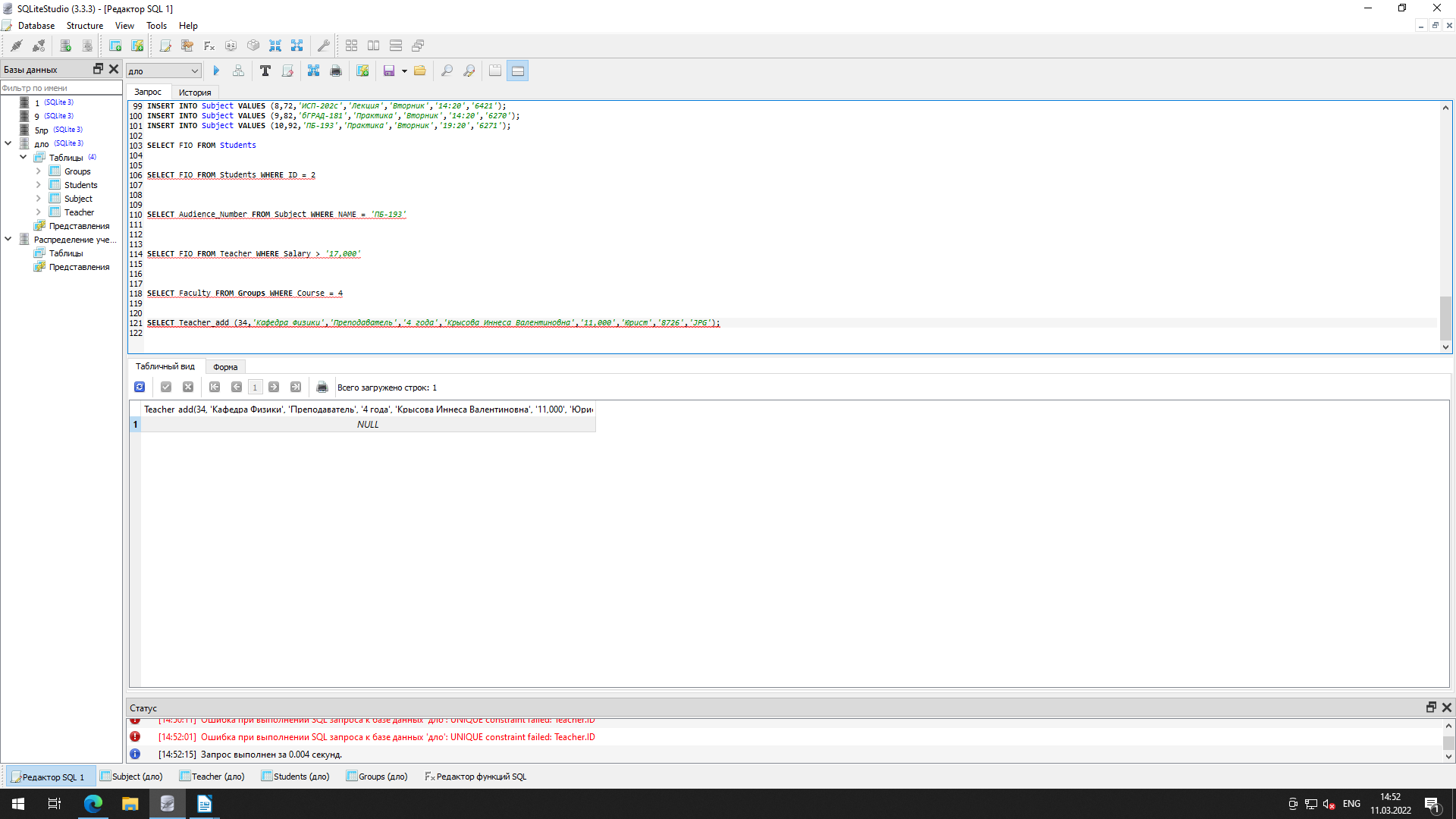


Рисунок 25 – Использование функции«Teacher\_add»

Данные, внесенные в таблицу, представлены на рисунке 3.

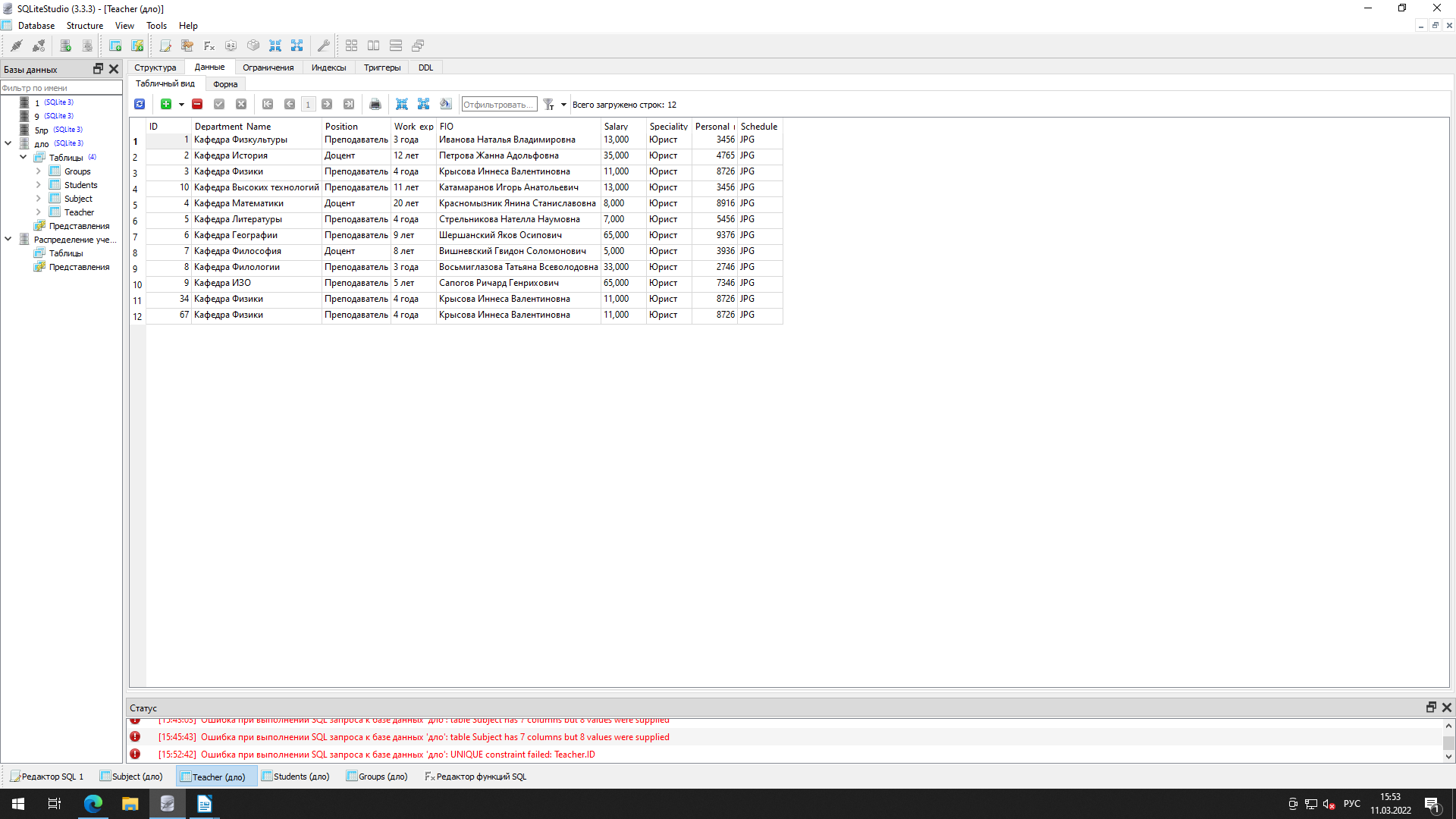


Рисунок 26 – Результат использования функции «Teacher\_add»

Создаем скалярную функцию «Students\_add» с аргументами: ID, FIO, Phone\_Number. Реализация на языке SQL:

INSERT INTO Students VALUES (@ID, @FIO, @Phone\_Number );

Создание функции представлено на рисунке 4.

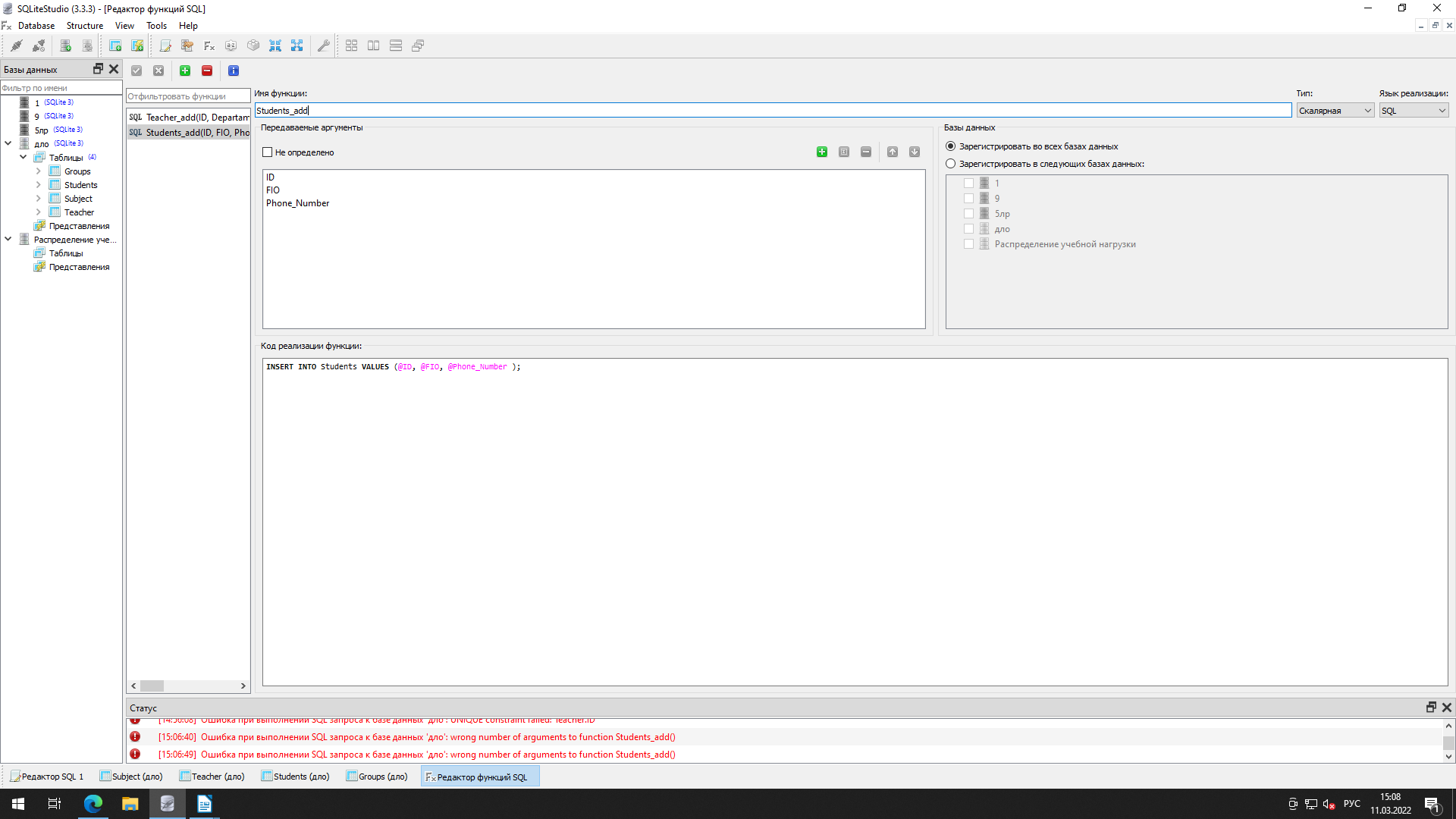


Рисунок 27 – Скалярная функция «Students\_add»

Использование функции «Students\_add»

SELECT Students\_add (24,'Сатанитский Константин Глебович','+79456458922');

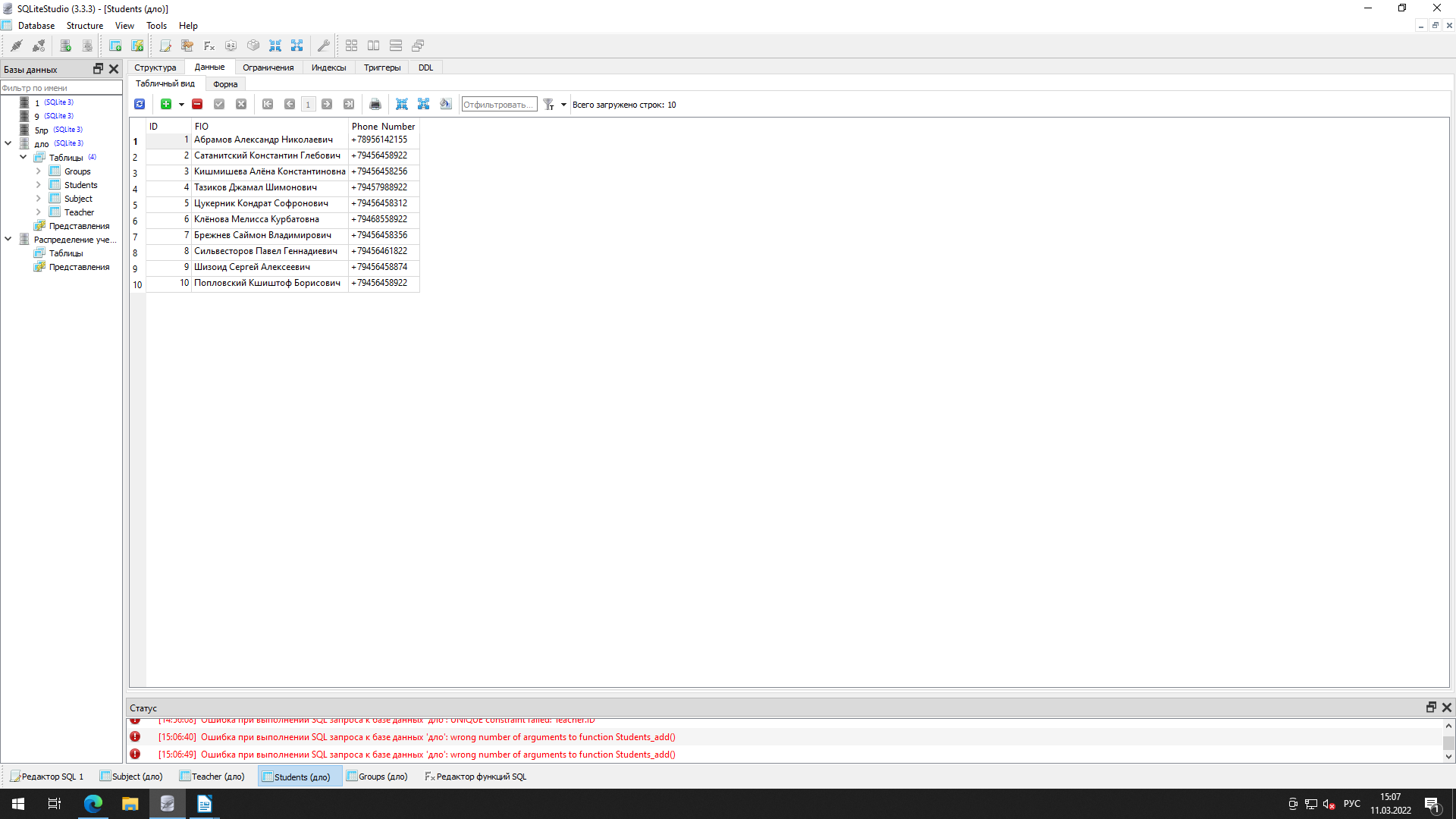


Рисунок 28 – Результат использования функции «Students\_add»

Создаем скалярную функцию «Groups\_add» с аргументами: ID, ID\_Students, Course, Faculty, Direction, Number\_of\_Students, Name, Phone\_Number\_Of\_The\_Headmen\_And\_Curator.

Реализация на языке SQL:

INSERT INTO Groups VALUES (@ID, @ID\_Students, @Course, @Faculty, @Direction, @Number\_Of\_Students, @Name, @Phone\_Number\_Of\_The\_Headmen\_And\_Curator);

Создание функции представлено на рисунке 6.

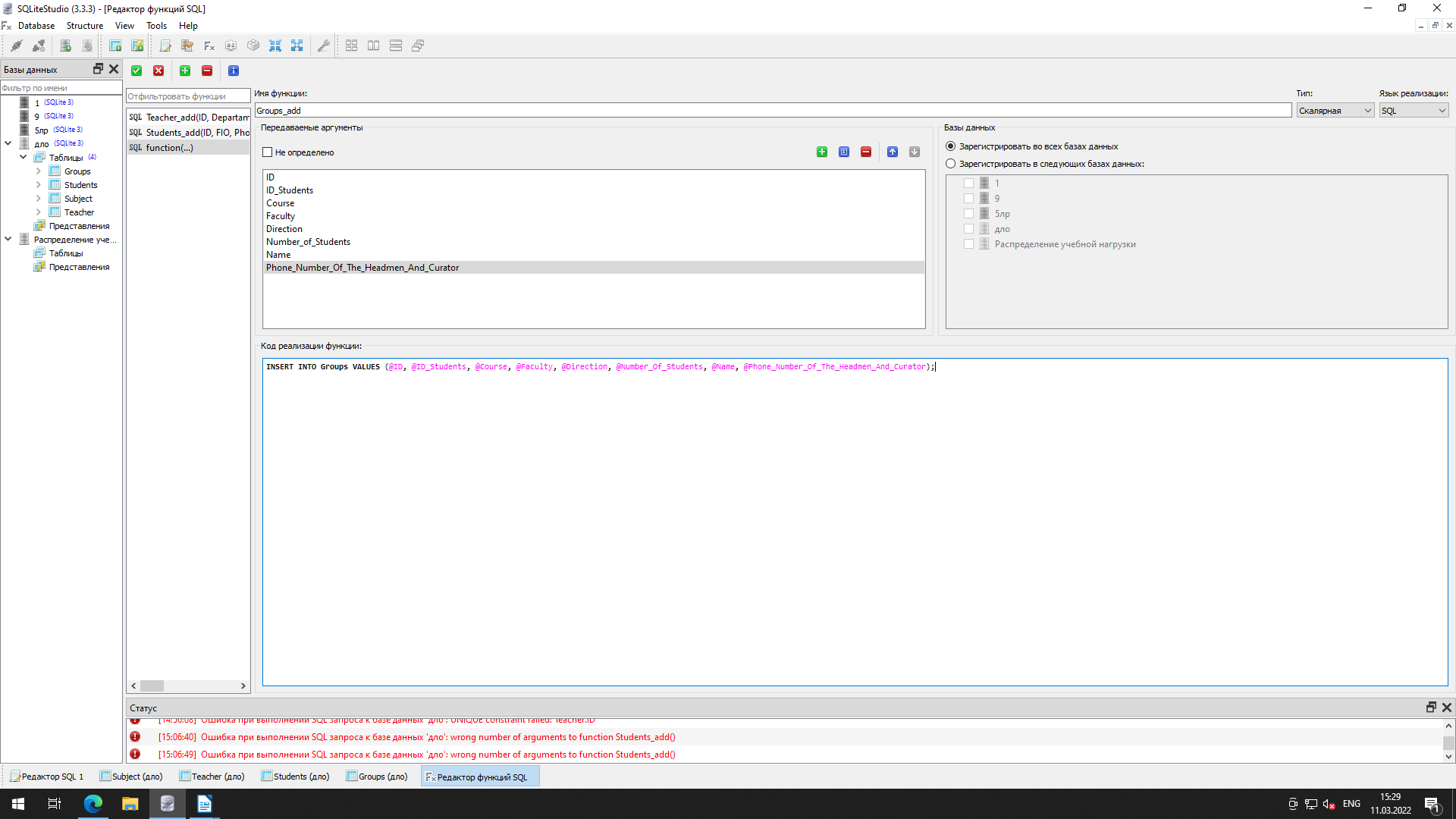


Рисунок 29 – Скалярная функция «Groups\_add»

Использование функции «Groups\_add»

SELECT Groups\_add (22,2,1,'СПК','ИСП',15,'ПБ-202с','+79214654278');

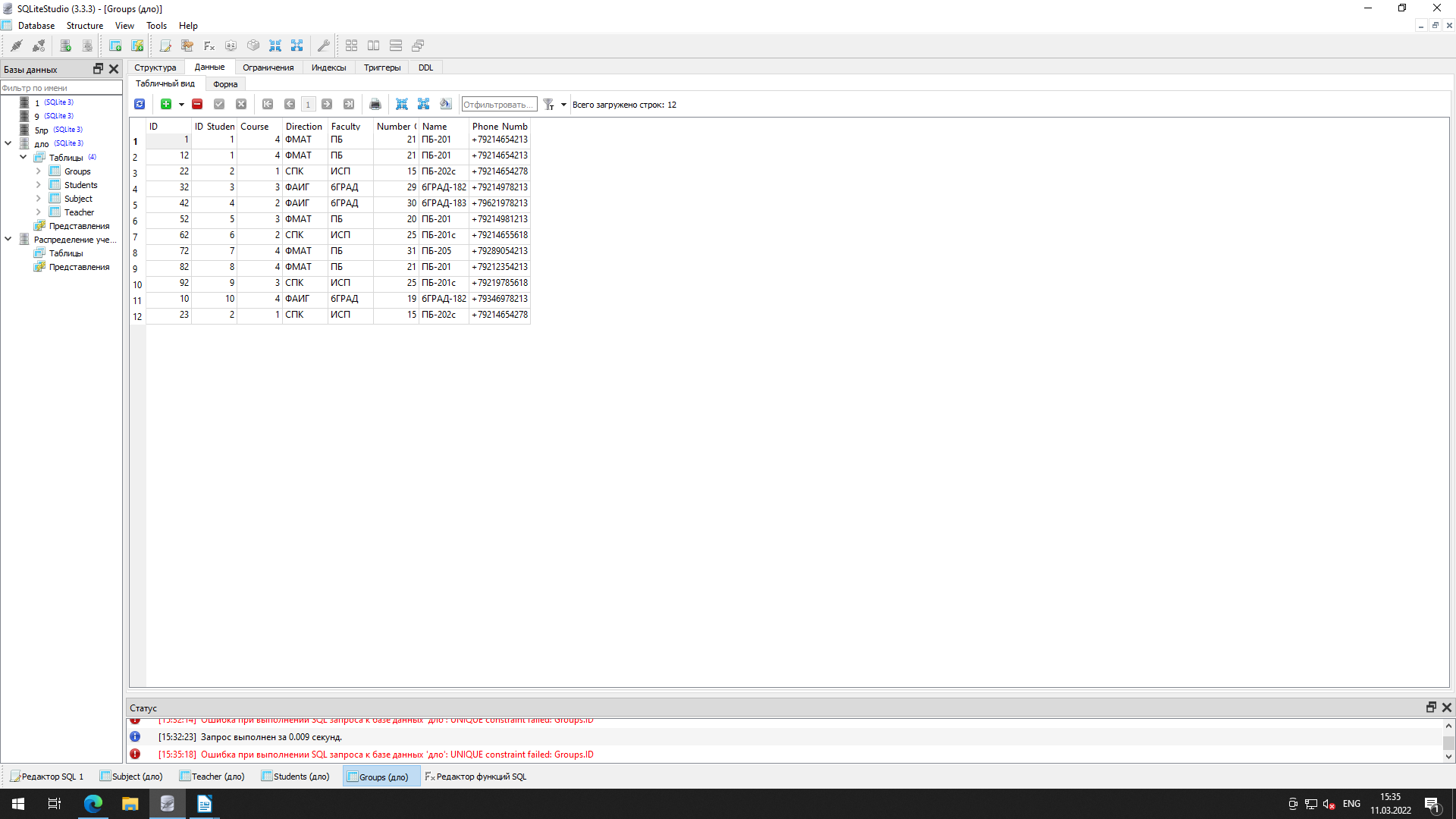


Рисунок 30 – Результат использования функции «Groups\_add»

Создаем скалярную функцию «Subject\_add» с аргументами: ID, ID\_Groups, ID\_Teacher, Name,Kind\_Of\_Couples, Day\_Of\_Week, Steam\_Time. Реализация на языке SQL:

INSERT INTO Subject VALUES (@ID, @ID\_Groups, @ID\_Teacher, @NAME, @Kind\_Of\_Couples, @Day\_Of\_Week, @Steam\_Time, @Audience\_Number);

Создание функции представлено на рисунке 8.

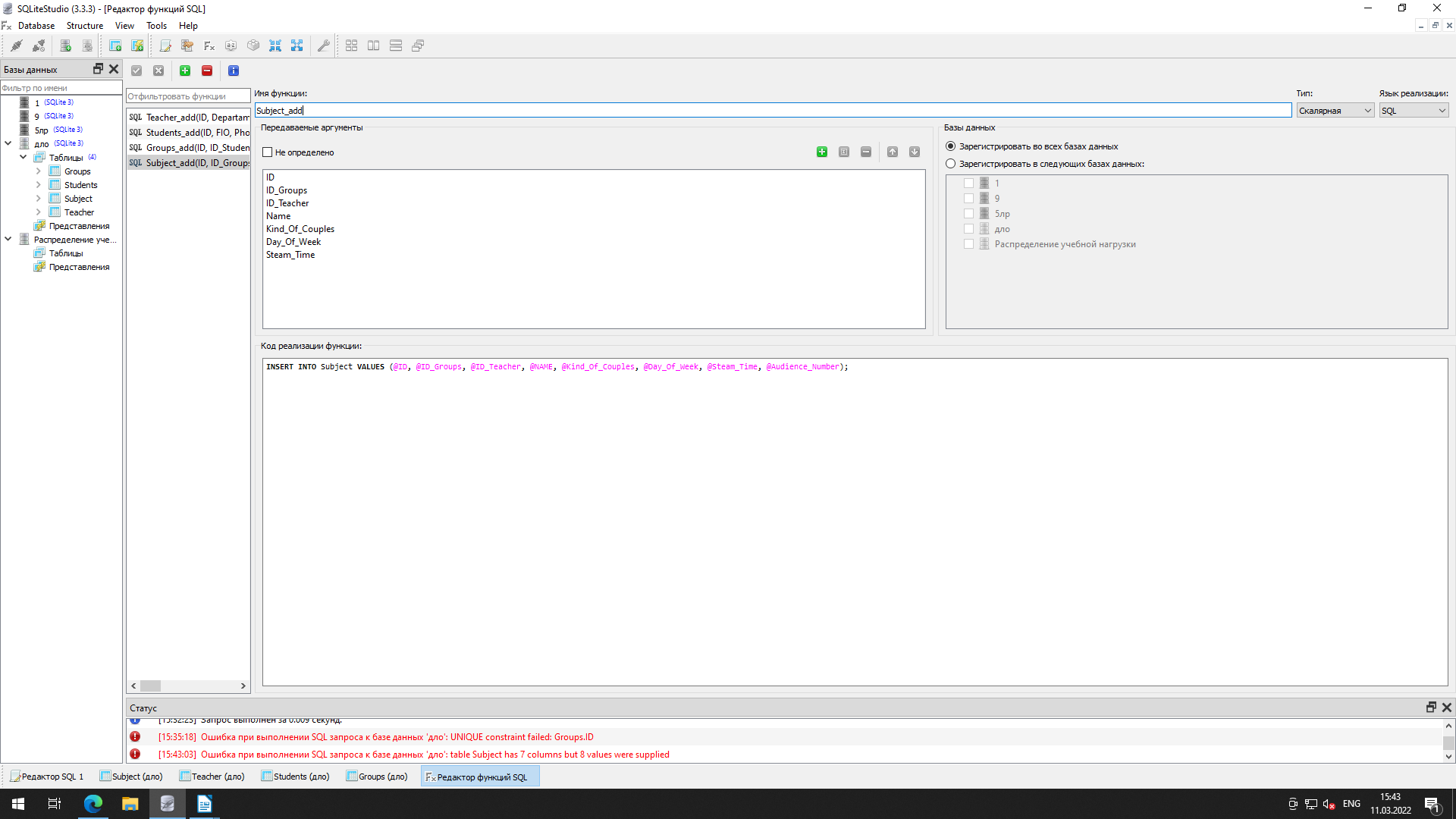


Рисунок 31 – Скалярная функция «Subject\_add»

Использование функции «Subject\_add»

SELECT Subject\_add (2,12,'ИСП-202с','Практика','Среда','8:30','3270');

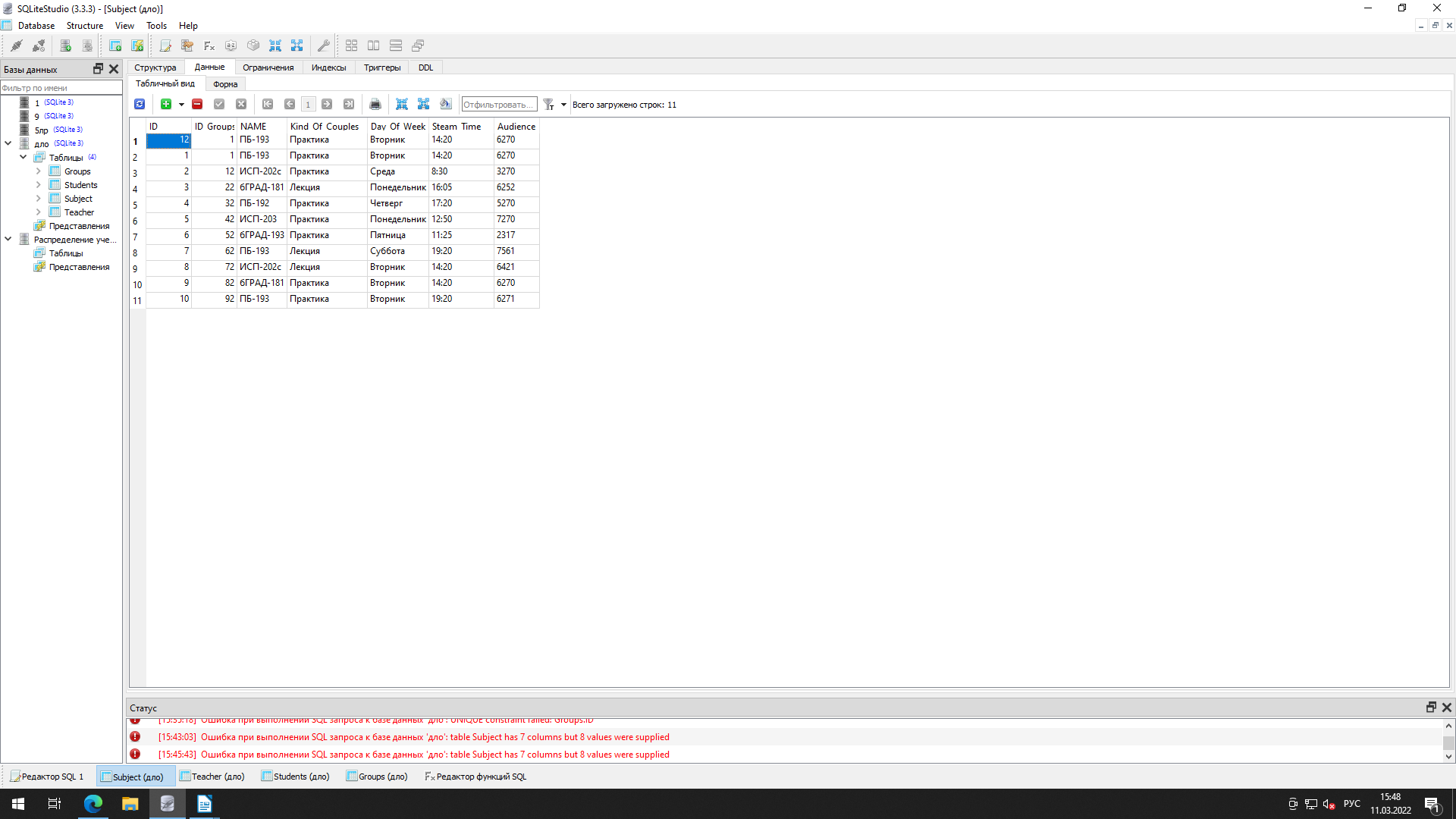


Рисунок 32 – Результат использования функции «Subject\_add»

# **Защита базы данных**

# **3.1 Шифрование данных в БД**

Для обеспечения доступа к БД, необходимо настроить права групп пользователей, выделенных лабораторной работе №1.

Для приведения к нормальной форме созданы таблицы Group и Tables. Код создания и заполнения таблиц представлен ниже:

CREATE TABLE 'Group' (

ID INTEGER PRIMARY KEY ,

Group\_Name VARCHAR(22) NOT NULL

);

Созданная таблица «Rules» представлена на рисунке 33

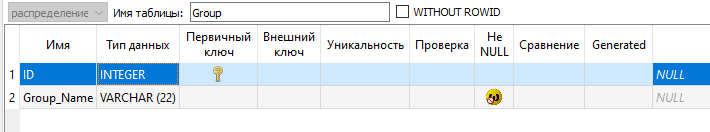


Рисунок 33 ­ Структура таблицы Group

Таблица 13 ­ таблица Group

|  |  |
| --- | --- |
| ID | Group\_Name |
| 1 | Сотрудник |
| 2 | Администратор |
| 3 | Клиент |
| 4 | Администратор БД |

INSERT INTO 'Group' VALUES (1, 'Сотрудник');

INSERT INTO 'Group' VALUES (2, 'Администратор');

INSERT INTO 'Group' VALUES (3, 'Клиент');

INSERT INTO 'Group' VALUES (4, 'Администратор БД');

Заполненная таблица «Group» представлена на рисунке 34.

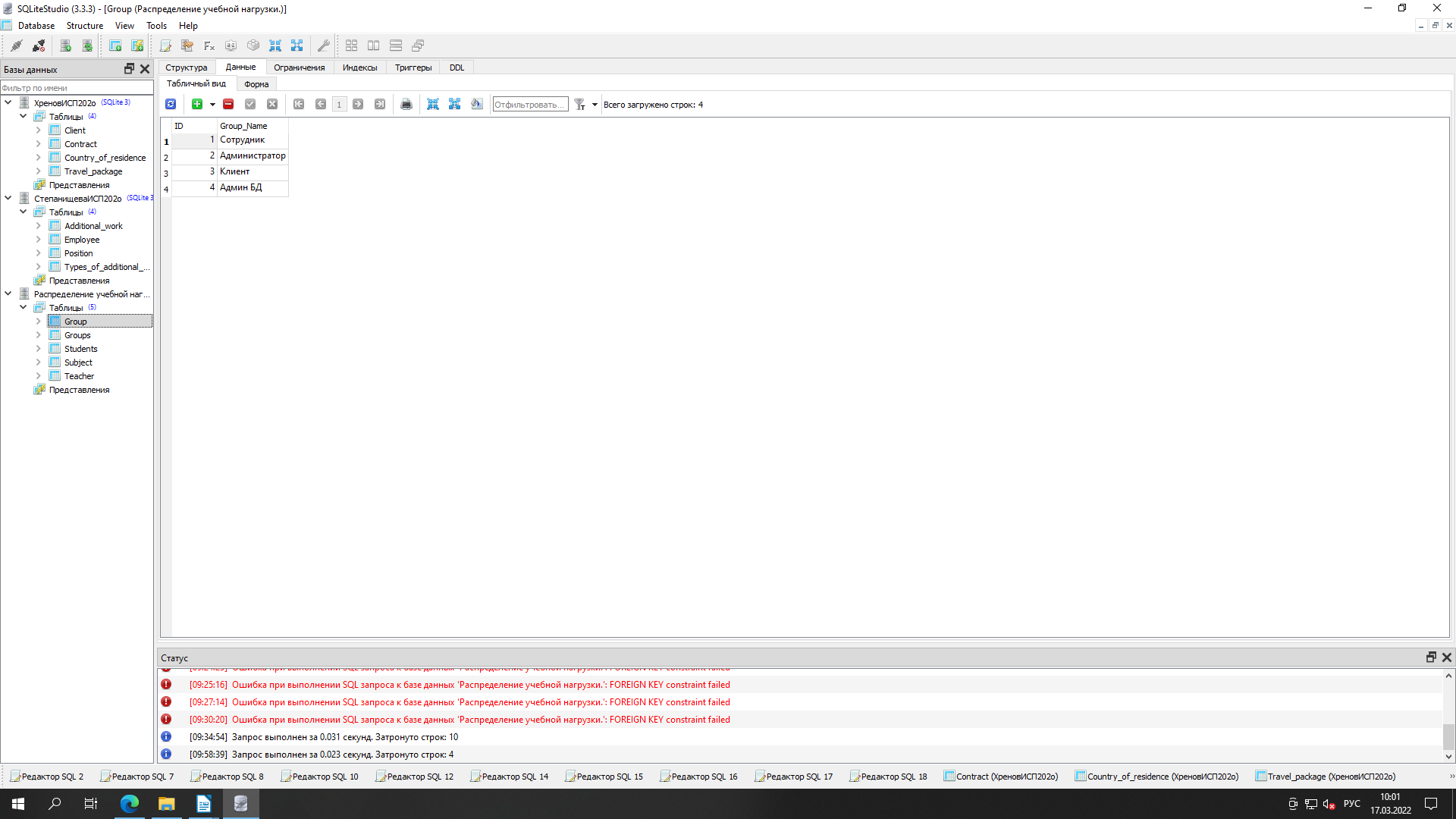


Рисунок 34 – Заполненная таблица «Group»

Создаем таблицу «Rules» для хранения прав доступа. Реализация на языке SQL:

CREATE TABLE Rules (

ID INTEGER PRIMARY KEY,

Group\_ID INTEGER NOT NULL,

[SELECT] INTEGER NOT NULL,

[UPDATE] INTEGER NOT NULL,

[INSERT] INTEGER NOT NULL,

[DELETE] INTEGER NOT NULL

)

Созданная таблица «Rules» представлена на рисунке 2.

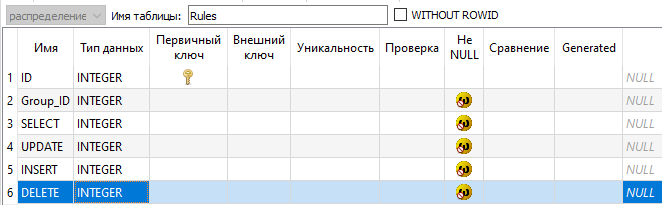


Рисунок 34 – Таблица «Rules»

Сотрудник имеет право смотреть информацию и менять свои личные данные.

Администратор не имеет право смотреть удалять и создавать данные.

Клиент не имеет право смотреть информацию, например маршрут.

Администратор базы данных и хозяин холдинга ограничений не имеют.

На основании следующих данных, заполнили таблицу Rules

Таблица 13 –Rules

| ID | Group\_ID | SELECT | UPDATE | INSERT | DELETE |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 2 | 1 | 1 | 0 | 1 | 1 |
| 3 | 1 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 0 | 1 | 1 |
| 5 | 1 | 1 | 0 | 1 | 1 |
| 6 | 1 | 1 | 0 | 1 | 1 |
| 7 | 1 | 1 | 0 | 1 | 1 |
| 8 | 2 | 0 | 0 | 0 | 0 |
| 9 | 2 | 0 | 0 | 0 | 0 |
| 10 | 2 | 0 | 0 | 0 | 0 |
| 11 | 2 | 1 | 0 | 0 | 0 |
| 12 | 2 | 0 | 0 | 0 | 0 |
| 13 | 2 | 0 | 0 | 0 | 0 |
| 14 | 2 | 0 | 0 | 0 | 0 |
| 15 | 3 | 0 | 0 | 0 | 0 |
| 16 | 3 | 0 | 0 | 0 | 0 |
| 17 | 3 | 0 | 0 | 0 | 0 |
| 18 | 3 | 1 | 0 | 0 | 0 |
| 19 | 3 | 0 | 0 | 0 | 0 |
| 20 | 3 | 0 | 0 | 0 | 0 |
| 21 | 3 | 0 | 0 | 0 | 0 |
| 22 | 4 | 1 | 1 | 1 | 1 |
| 23 | 4 | 1 | 1 | 1 | 1 |
| 24 | 4 | 1 | 1 | 1 | 1 |
| 25 | 4 | 1 | 1 | 1 | 1 |
| 26 | 4 | 1 | 1 | 1 | 1 |
| 27 | 4 | 1 | 1 | 1 | 1 |
| 28 | 4 | 1 | 1 | 1 | 1 |

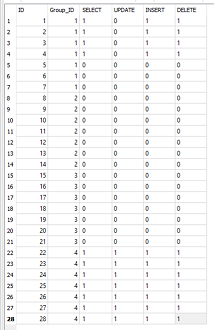


Рисунок 35 ­ заполненная таблица Rules

# **Комбинирование методов защиты БД**

Создадим таблицу с помощью, которой возможна идентификация пользователей.

Реализация таблицы на языке SQL:

CREATE TABLE Users (

ID INT PRIMARY KEY,

Group\_ID INT NOT NULL,

Name VARCHAR(14) NOT NULL,

Login VARCHAR(14) NOT NULL,

Password VARCHAR(16) NOT NULL,

);

На основании следующих данных, заполнили таблицу Users.

Таблица 14 ­ Users

| ID | Group\_ID | Name | Login | Password |
| --- | --- | --- | --- | --- |
| 1 | 1 | Иванов Иван Петрович | 1234 | Asdf12341 |
| 2 | 3 | Петров Петр Иванович | 9874 | Asdf12342 |
| 3 | 3 | Васильева Мария Антоновна | 6541 | Asdf12343 |
| 4 | 4 | Хрячков Семён Вольфович | 8912 | Asdf12344 |
| 5 | 1 | Паромонов Никита Паромонович | 7412 | Asdf12345 |
| 6 | 2 | Мартов Ярослав Мартович | 8963 | Asdf12346 |
| 7 | 2 | Курочкин Денис Денисович | 5632 | Asdf12347 |
| 8 | 3 | Дубинин Павел Павлович | 9872 | Asdf12348 |
| 9 | 1 | Павлов Павел Павлович | 9812 | Asdf12349 |
| 10 | 2 | Зимин Олег Юрьевич | 9124 | Asdf12310 |

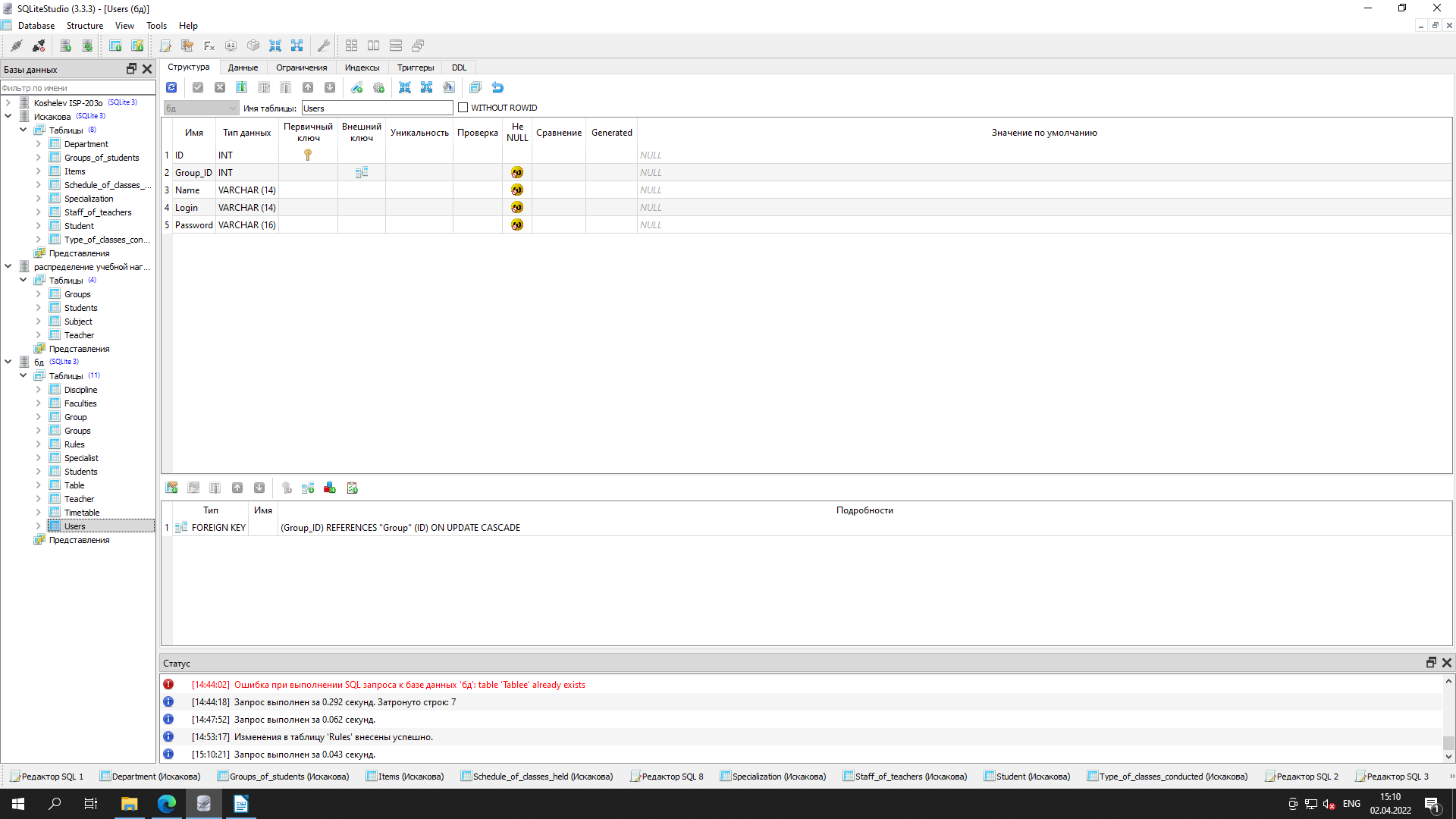


Рисунок 36 – Структура таблицы Users

Внесем данные в таблицу «Users».

Заполненная таблица представлена на рисунке 37.

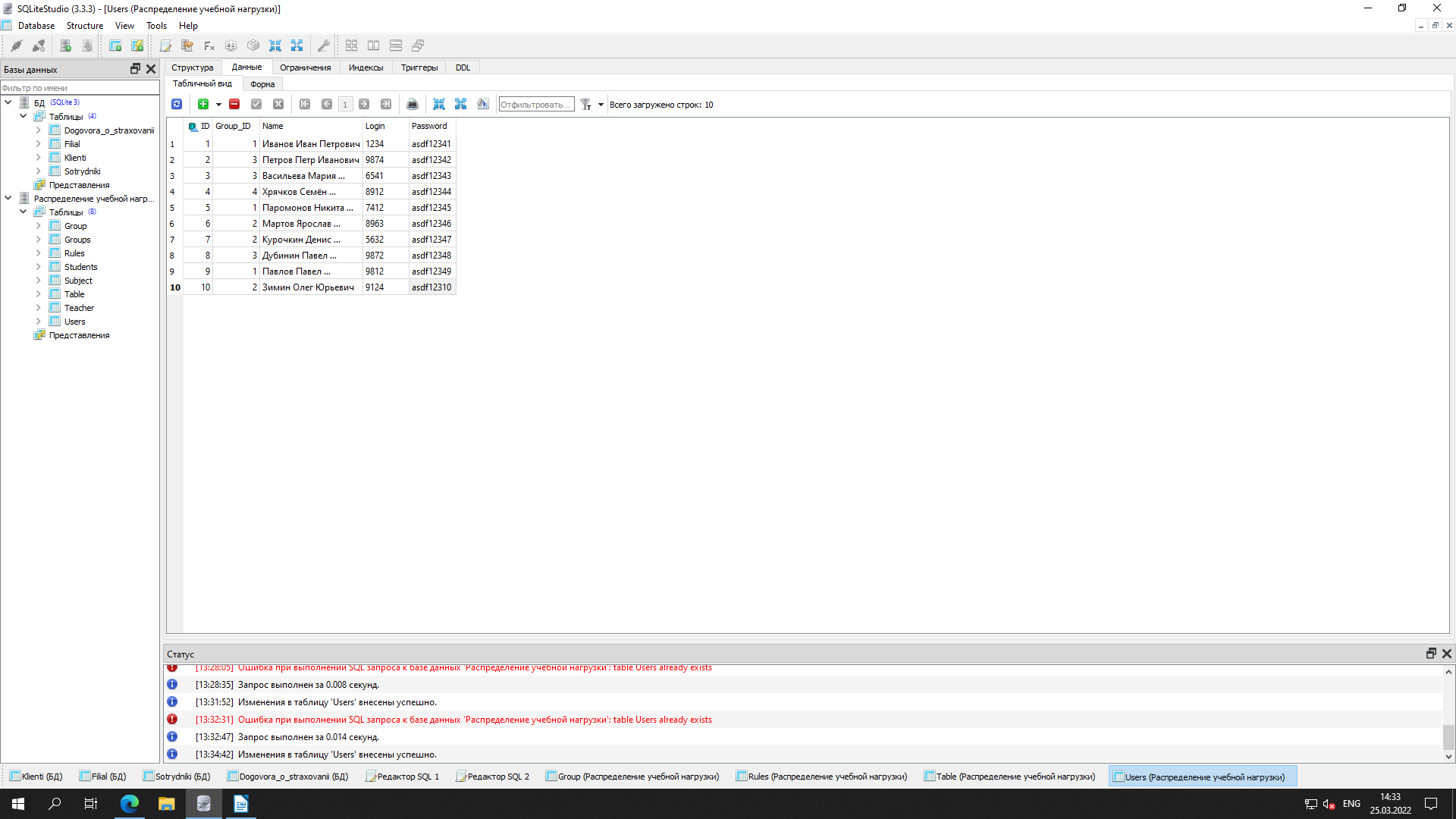


Рисунок 37 – Заполненная таблица «Users»

# **Заключение**

На основе БД сформировался новый пласт информационных технологий, которые эффективно используются во многих областях деятельности человека. Организации также нуждаются в специально разработанных БД. Они способствуют наиболее эффективной работе.

В ходе данного курсового проекта разработана база данных «Деканат». СУБД обеспечит быстрое получение информации, необходимой для своевременного реагирования на ту или иную ситуацию, быстрое выявление ошибок и легкость их исправления. БД деканат может существенно уменьшить влияние человеческого фактора на общее функционирование работы. Кроме того БД исключит избыточность информации и выполнения одной и той же работы дважды.

Все эти факторы положительно скажутся на общей работе деканата, экономии временных затрат и повышению производительности работы деканата, а, следовательно, всего ВУЗа.

Таким образом, автоматизация деканатов необходима для поддержки принятия решений руководством деканата при управлении учебным процессом на основе обработки интегрированных данных. В данном проекте была проанализирована предметная область и на этой основе были реализованы поставленные задачи. Данная курсовая работа позволит облегчить работу сотрудников деканата, что значительно повысит скорость и качество работы.

# **Список использованных источников**

1. Королева О.Н. Базы данных [Электронный ресурс]: курс лекций/ Королева О.Н., Мажукин А.В., Королева Т.В. - Электрон. текстовые данные. - М.: Московский гуманитарный университет, 2012. - 66 c.
2. Балалаев М.А. Проектирование баз данных: Методические указания по выполнению курсового проекта по дисциплине «Базы данных» / М.А. Балалаев; ДВГУПС., 2015. – 87с.
3. Кучма В.Н. Стандарты проектирования автоматизированных систем обработки данных: Учебное пособие для ВУЗов / В.Н. Кучма; ДВГУПС. Каф. «Информационные технологии и системы». - Хабаровск: Изд-во ДВГУПС, 2006. - 80 с.
4. Кандзюба СП. Ое1рЫ 5. Базы данных и приложения. Лекции и упражнения. /, СП. Кандзюба, В.Н. Громов - К.: Диасофт, 2010. - 592 с.
5. Дейт К.Дж. Введение в системы баз данных. Изд. 7 - М. - СПб. - Киев: Вильяме, 2011. - 1072 с.
6. Хансен Г., Хансен Дж. Базы данных. Разработка и управление. - М.: Бином, 2012. - 700 с.