Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

**ЗАПРОСЫ НА ВЫБОРКУ ДАННЫХ**

Отчёт о лабораторной работе № 4

по дисциплине «Базы данных»

Выполнил: студент гр. 431-3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бекиш Е.П.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Проверил: ассистент каф. АСУ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Яблонский Я. В.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Томск 2023

**1 Цель лабораторной работы**

– ознакомиться с принципами работы консольной утилиты psql;

– научиться создавать макеты таблиц с использованием команд SQL.

**2 Описание таблиц БД из индивидуального задания**

На факультете существует группа студентов, обучаемых по индивидуальным планам. Все студенты проживают в общежитии. Некоторые совмещают учёбу с работой. Каждый студент в течение семестра изучает несколько учебных дисциплин. До окончания семестра студент должен отчитаться по каждой изучаемой дисциплине. Единственный предусмотренный вид отчётности — экзамен. Один из преподавателей факультета курирует группу. В его обязанности входит текущий и итоговый контроль успеваемости студентов. Для успешного выполнения обязанностей куратору нужно всегда иметь под рукой сведения о студентах и об их успеваемости.

На рисунке 2.1 представлена структура базы данных, соответствующая индивидуальному заданию.

Рисунок 2.1 – структура базы данных

**3 Описание процесса реализации таблиц**

Первым делом для занесения данных в таблицы, необходимо их создать. Для этого используем команду CREATE DATABASE, которая позволяет создать новую базу данных.

В наших таблицах присутствуют свойства, где задаются правила проверки значений поля с помощью команды CHECK. При внесении данных СУБД отбросит все строки, где значения в столбцах не проходят проверку CHECK. Но некоторые свойства, которые имею тип данных строка, могут содержать определенные символы, которые нужно проверить, поэтому мы можем использовать регулярные выражения. Они представлены в виде шаблона … CHECK(имя\_поля SIMILAR TO ‘регулярное\_выражение’).

Примеры классов литералов:

* [A-Z] - символы верхнего регистра
* [a-z] - символы нижнего регистра
* [A-Za-z] - символы верхнего и нижнего регистра
* [0-9] – цифры (еще можно обозначать как [\d])

Так же некоторые поля требуют свою уникальность, которую можно ограничить с помощью команды UNIQUE. Указывает на то, что все значения в поле должны быть уникальными (т.е. запрещает наличие дублирующихся

значений в этом поле).

В таблице STUDENT присутствует ссылочность на первичный ключ поля из таблицы ROOM. Создается с помощью REFERENCES имя\_таблицы(имя\_поля). Связывает поле с первичным ключом другой таблицы.

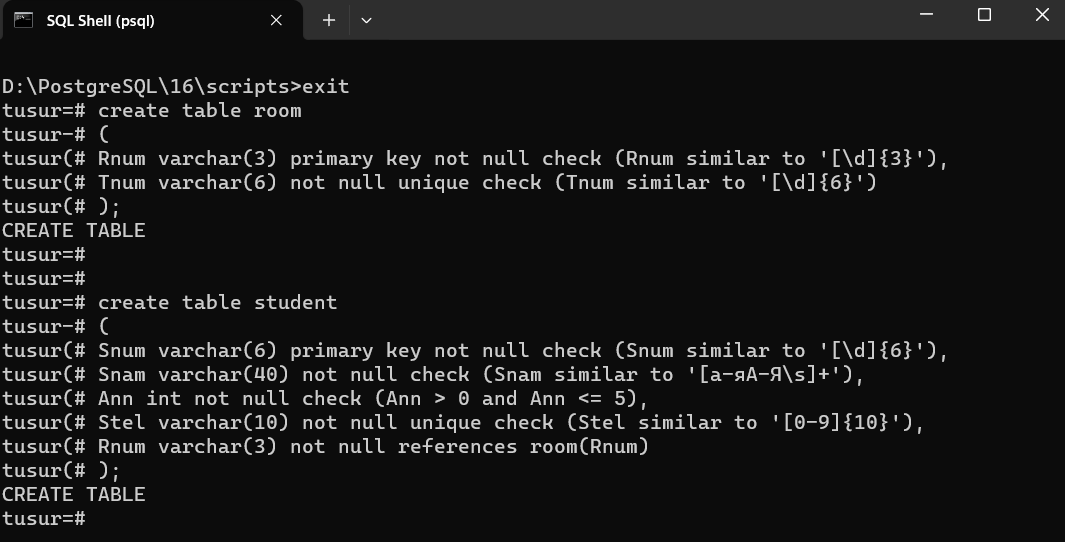
Таблицы, которые нужно создать – STUDENT, ROOM представлены на рисунке 3.1.

Рисунок 3.1 – создание таблиц STUDENT и ROOM

Теперь проверим корректность условий свойств имен для каждой таблицы.

Целостность свойств имен таблицы STUDENT:

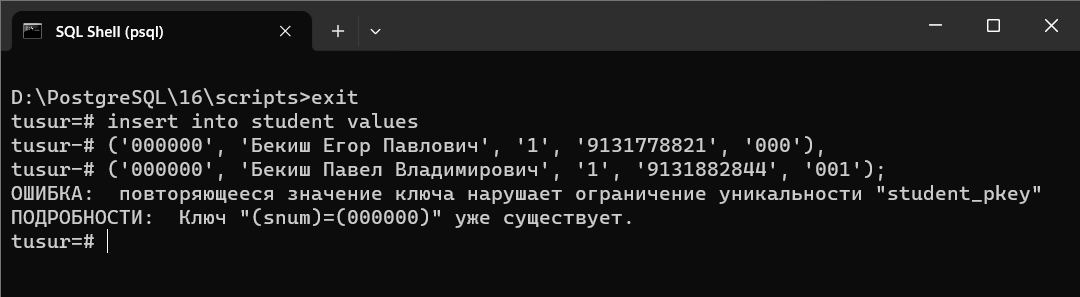
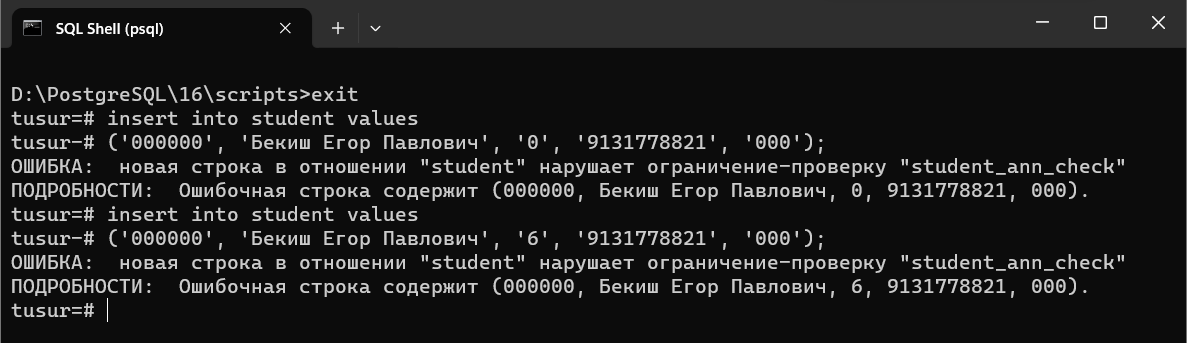
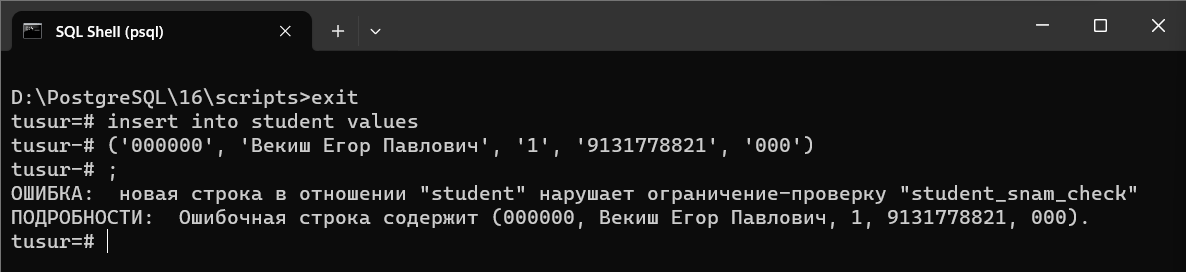
* Snum - Номер студенческого билета. (рис. 3.2)
* Snam - Фамилия, имя, отчество студента. (рис. 3.3)
* Ann - Год обучения. (рис. 3.4)
* Stel - Номер индивидуального телефона студента. (рис. 3.5)

Рисунок 3.2 – проверка Snum на первичный ключ таблицы

Рисунок 3.3 – проверка Snam на ограничение русских символов 

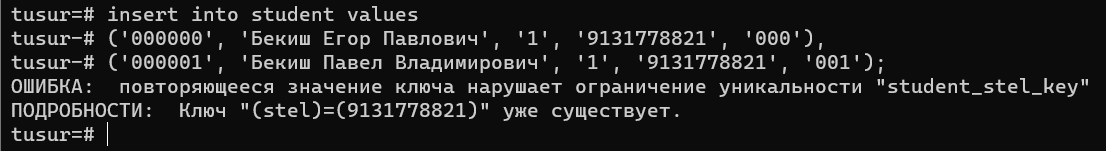
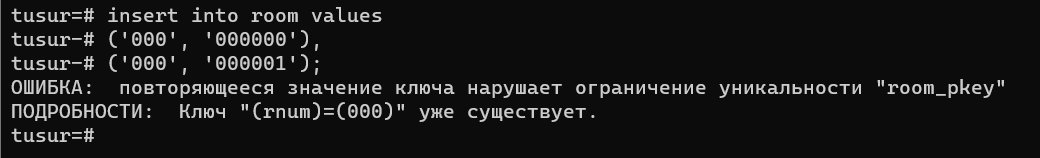
Рисунок 3.4 – проверка Ann на ограничение года от 1 до 5

Рисунок 3.5 – проверка Stel на уникальность номера телефона

Целостность свойств имен таблицы ROOM:

* Rnum - Номер комнаты. (рис. 3.6)
* Tnum - Номер телефона, установленного в комнате. (рис. 3.7)

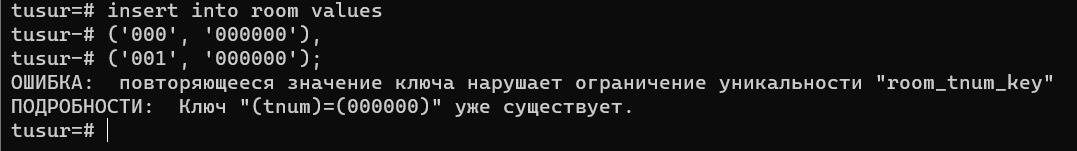
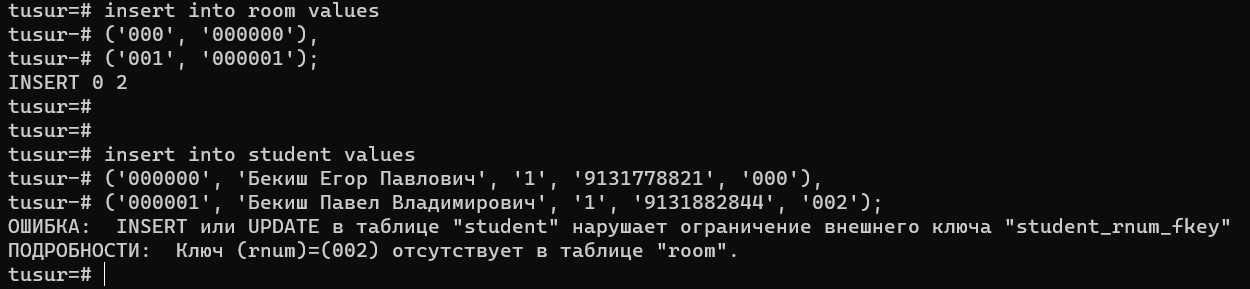
Рисунок 3.6 – проверка Rnum на первичный ключ

Рисунок 3.7 – проверка Tnum на уникальность телефона в комнате

Сcылочность свойств имен таблицы STUDENT с первичным ключом таблицы ROOM представлена на рисунке 3.8.

Рисунок 3.8 – проверка Rnum на ссылочность первичного ключа из ROOM

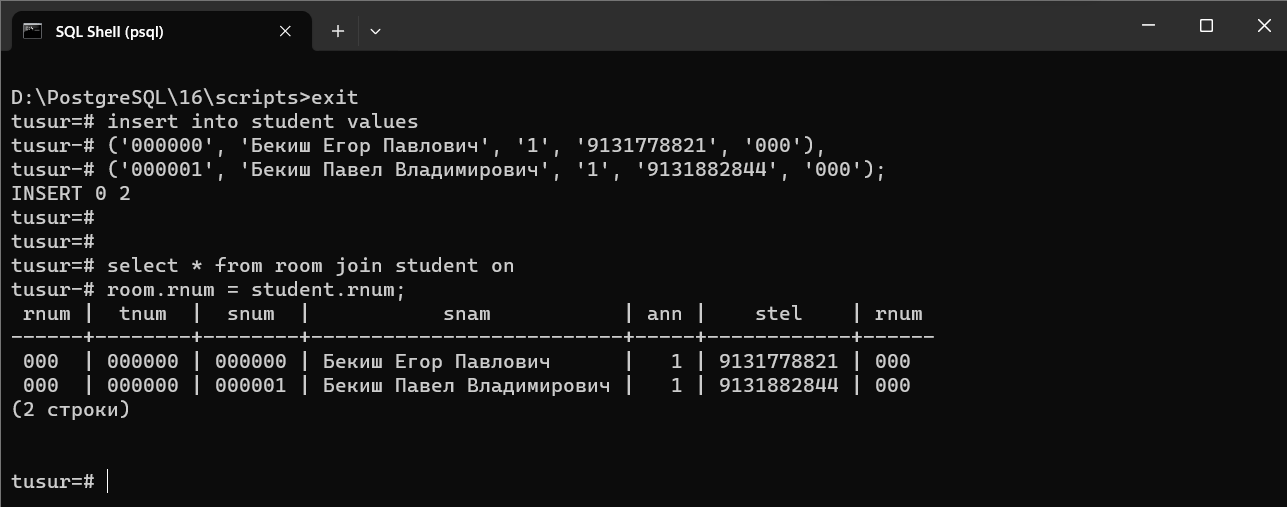
Успешное добавление данных в таблицу представлено на рисунке 3.9.

Рисунок 3.9 – успешное добавление данных

**4 Вывод**

В ходе выполнение лабораторной работы я научился пользоваться утилитой psql, создавать в ней таблицы, ограничивать поля классов требуемыми свойствами, используя в них регулярные выражение при необходимости.

**1 Цель лабораторной работы**

– познакомиться с принципами работы платформы администрирования и обслуживания сервера СУБД PostgreSQL - pgAdmin;

– научиться создавать макеты таблиц с использованием графического интерфейса pgAdmin.

**2 Описание таблиц БД из индивидуального задания**

На факультете существует группа студентов, обучаемых по индивидуальным планам. Все студенты проживают в общежитии. Некоторые совмещают учёбу с работой. Каждый студент в течение семестра изучает несколько учебных дисциплин. До окончания семестра студент должен отчитаться по каждой изучаемой дисциплине. Единственный предусмотренный вид отчётности — экзамен. Один из преподавателей факультета курирует группу. В его обязанности входит текущий и итоговый контроль успеваемости студентов. Для успешного выполнения обязанностей куратору нужно всегда иметь под рукой сведения о студентах и об их успеваемости.

На рисунке 2.1 представлена структура базы данных, соответствующая индивидуальному заданию.

Рисунок 2.1 – структура базы данных

**3 Описание процесса реализации таблиц**

Для того, чтобы создать таблицу нужно в созданной базе данных перейти во вкладку Table и нажать ПКМ. Путь до нее Schemas\public\ как показано на рисунке 3.1.

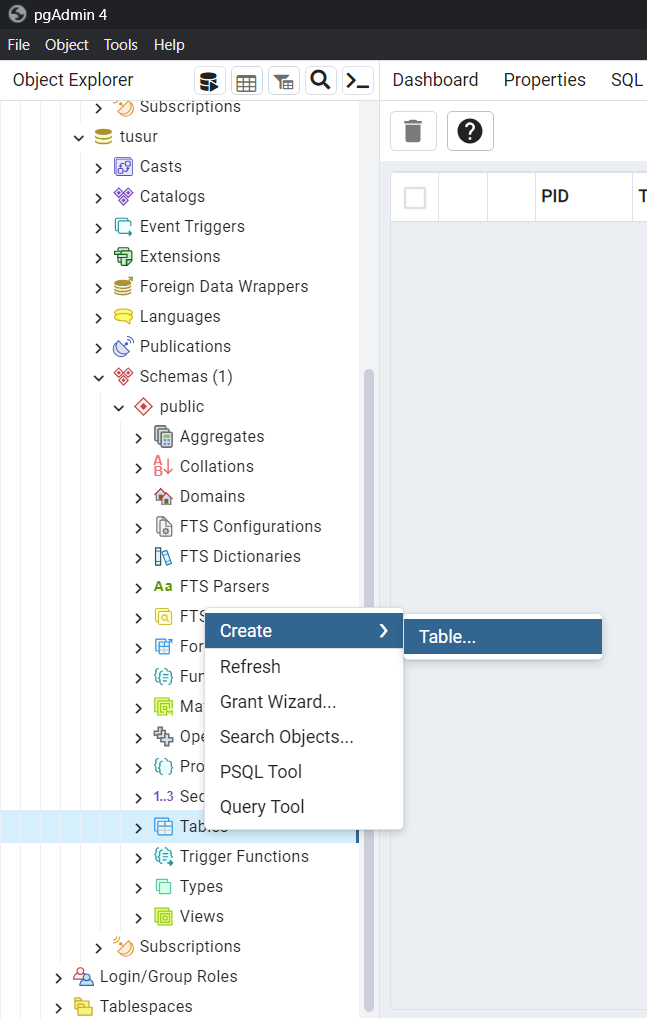


Рисунок 3.1 – путь до Table

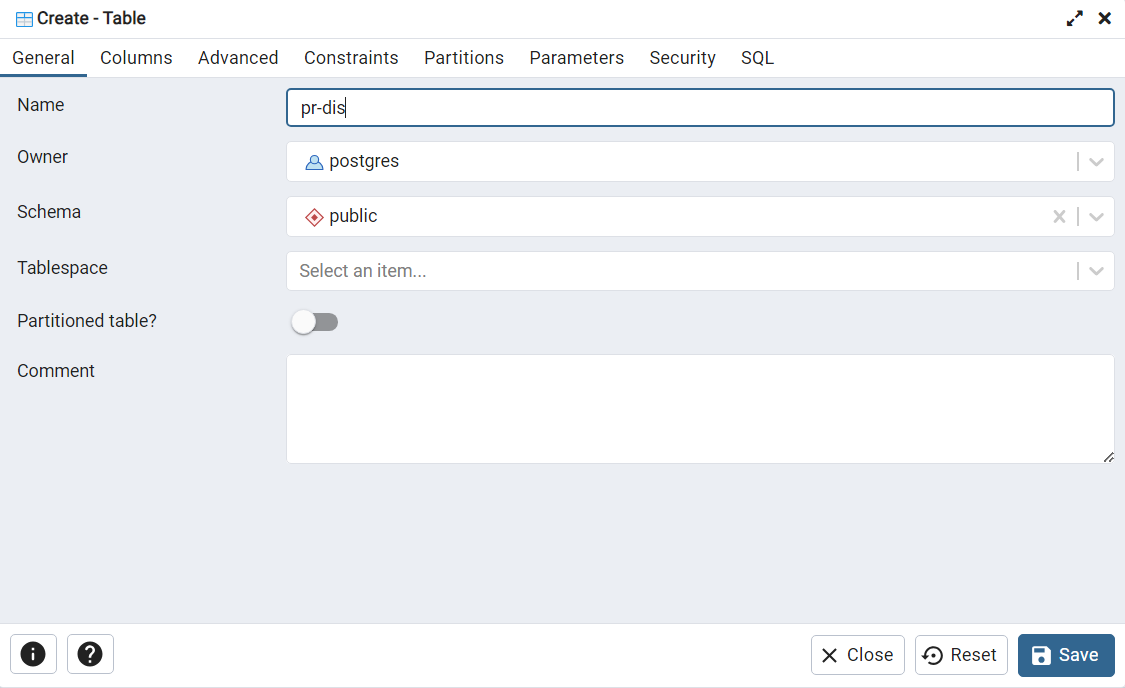
Далее, во всплывающем окне, вводим название таблицы (рис. 3.2).

Рисунок 3.2 – название таблицы

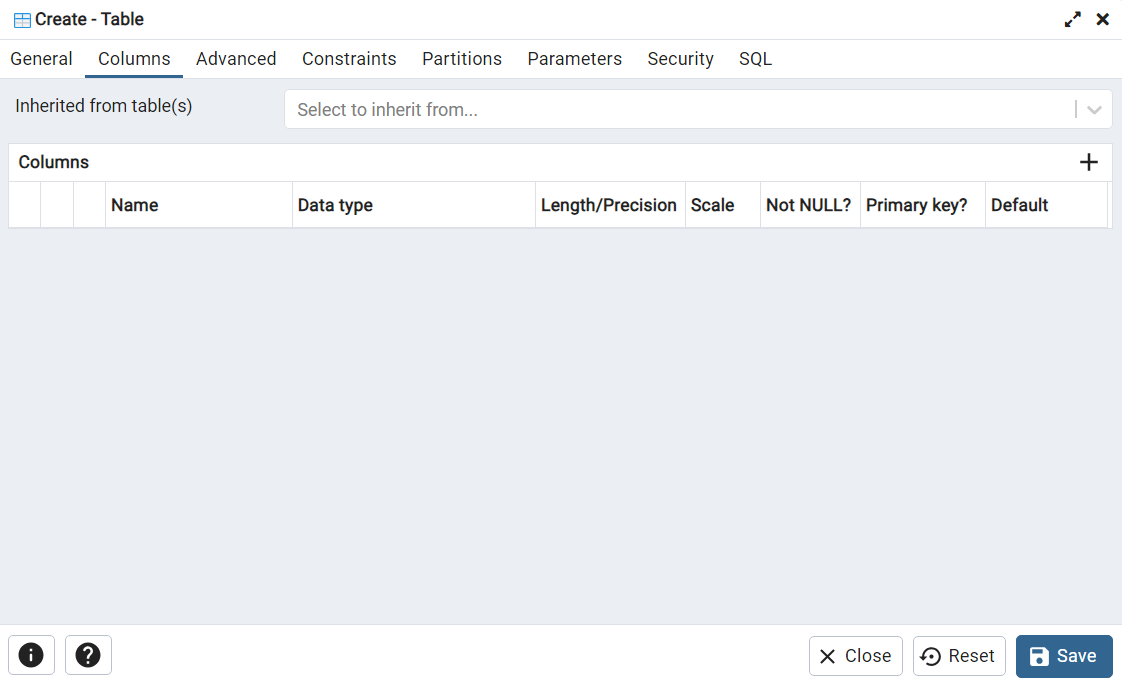
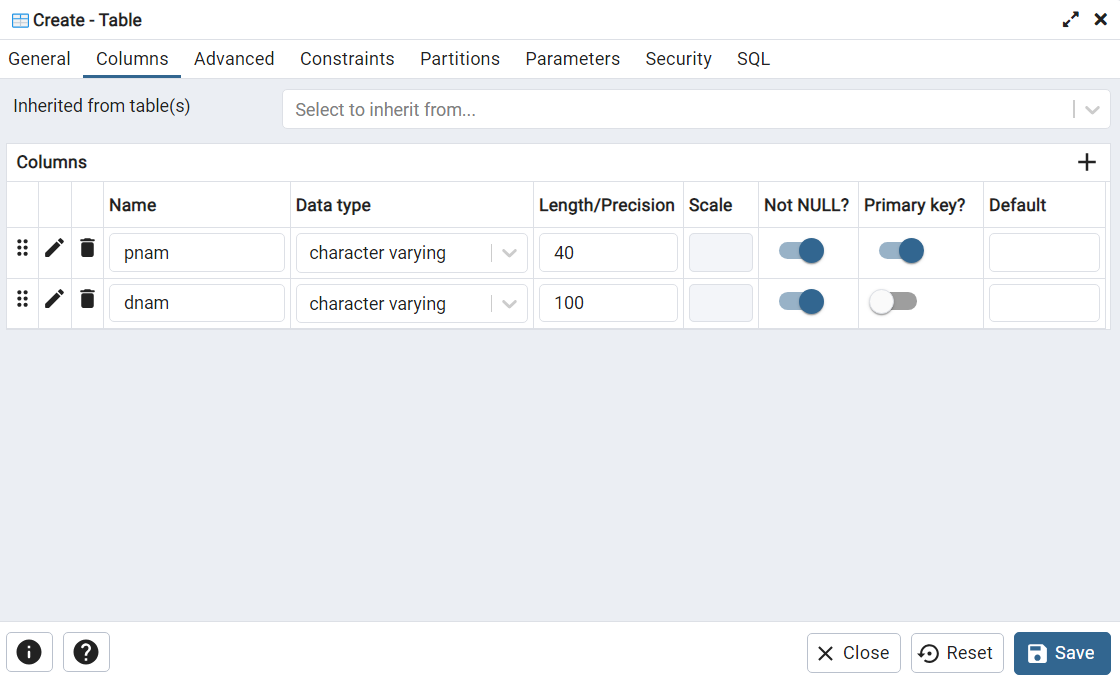
После переходим во вкладку Columns, где мы создадим поля. Результат можно увидеть на рисунке 3.3 – 3.4.

Рисунок 3.3 – создание полей таблицы

Рисунок 3.4 – заполненные поля таблицы

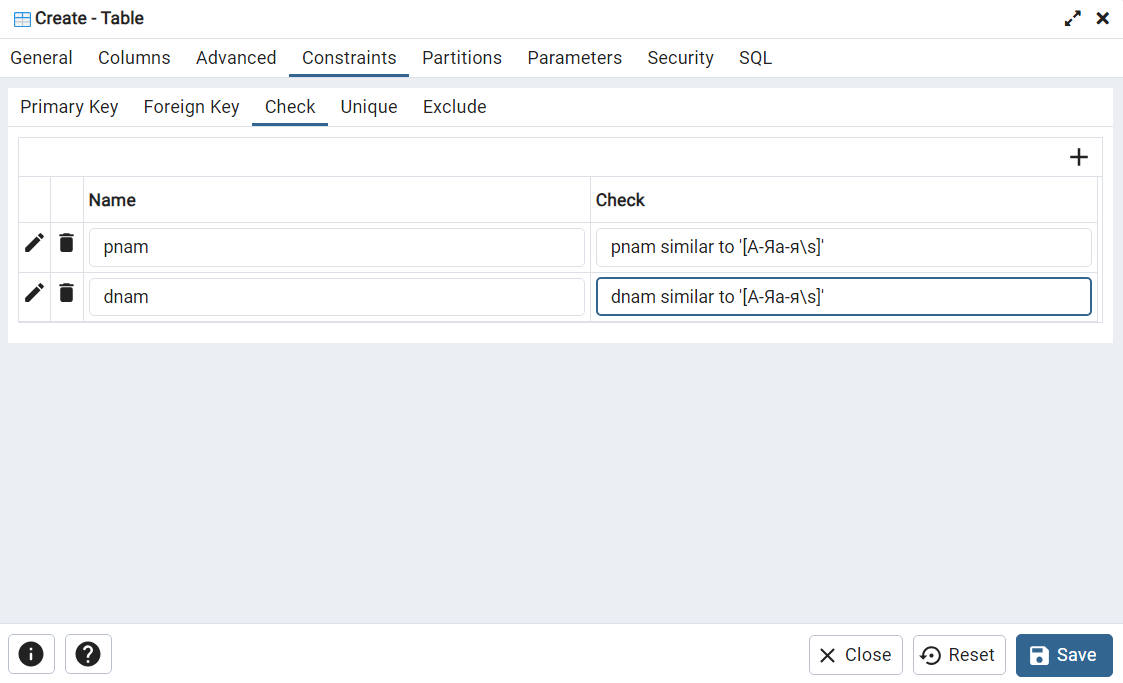
Теперь переходим во вкладку Constraints, чтобы задать ограничение созданным полям (рис. 3.5).

Рисунок 3.5 – ограничение полей

Так же при создании полей мы создали первичный ключ на поле Pnum, это можно наблюдать ранее на рисунке 3.4.

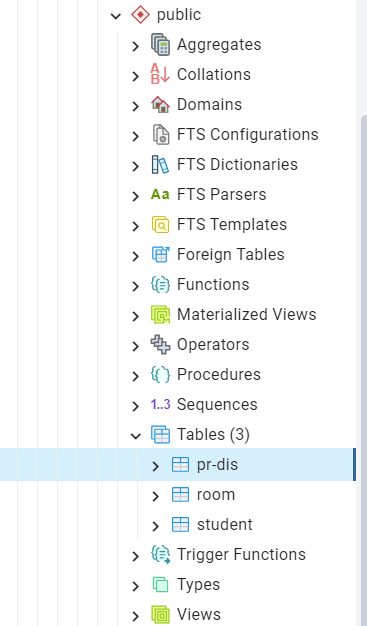
Теперь нажмем на кнопку Save для сохранения внесенных параметров и создания таблицы (рис 3.6).

Рисунок 3.6 – успешное создание таблицы

Проделаем те же действия для таблицы EXAMS с новыми свойствами. Сделаем внешний ключ для полей с именами Snum и Pnum ка показано на рисунке 3.7 – 3.8.

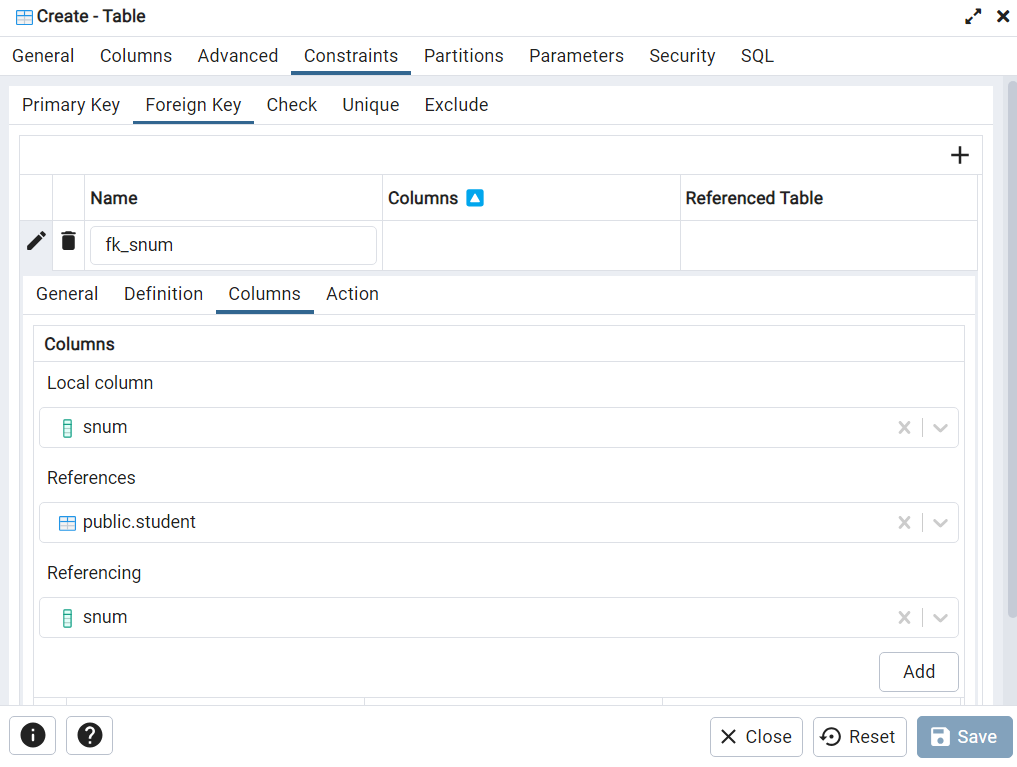
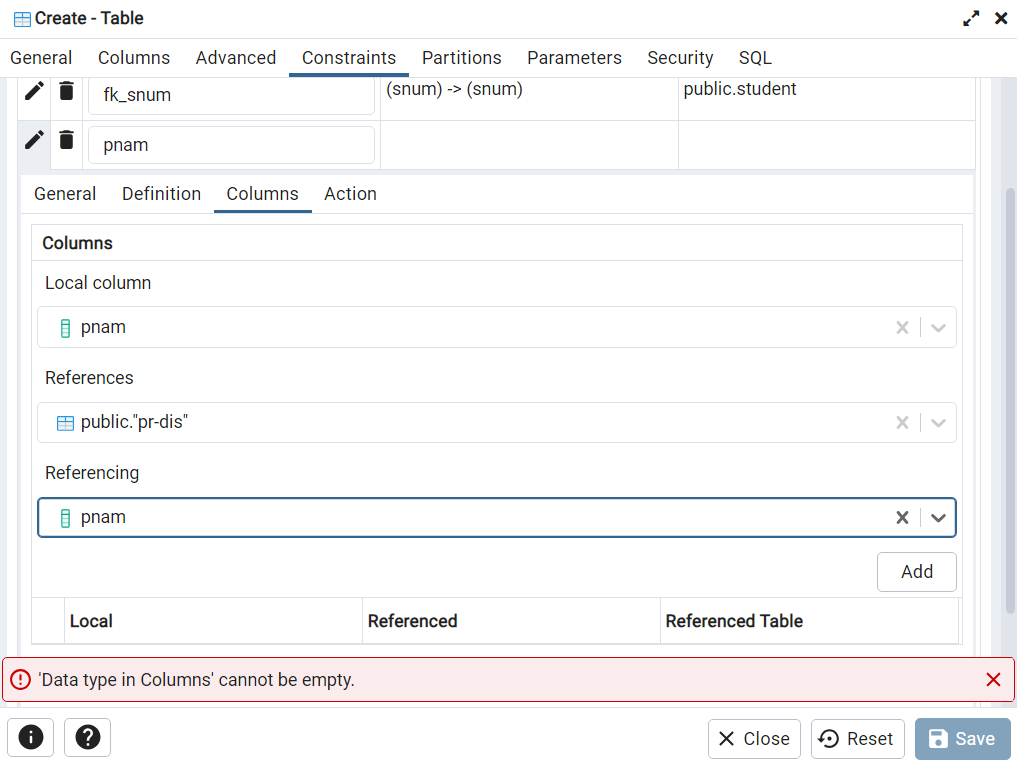
Рисунок 3.7 – внешний ключ для Snum из таблицы STUDENT

Рисунок 3.8 – внешний ключ для Pnum из таблицы PR-DIS

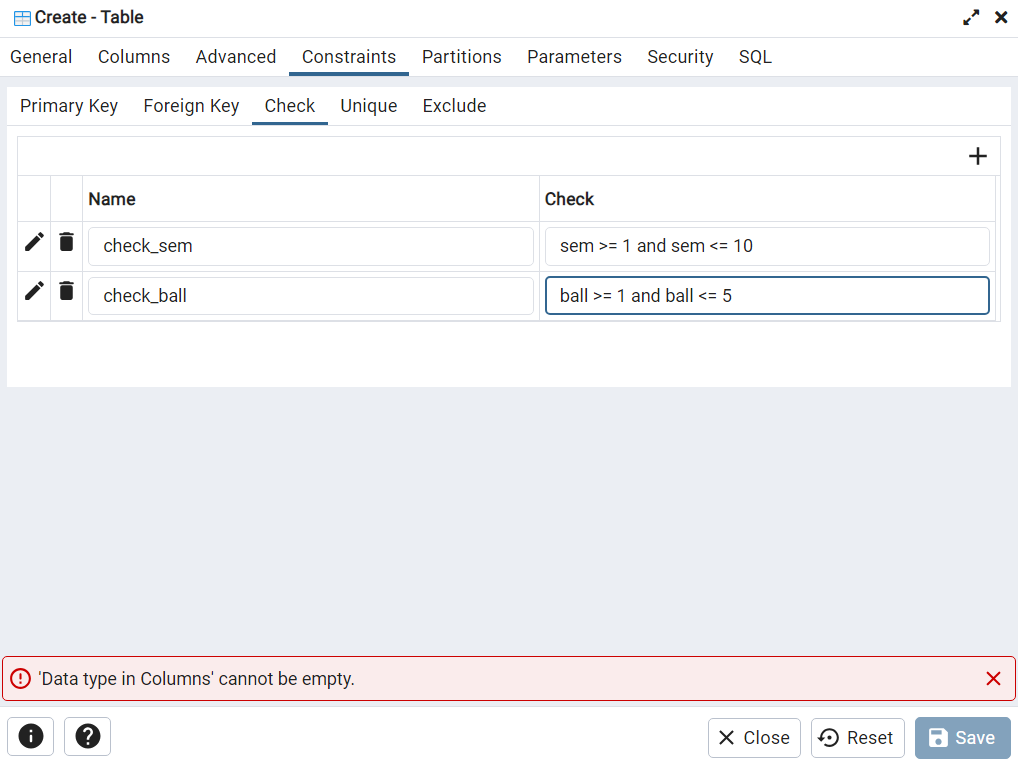
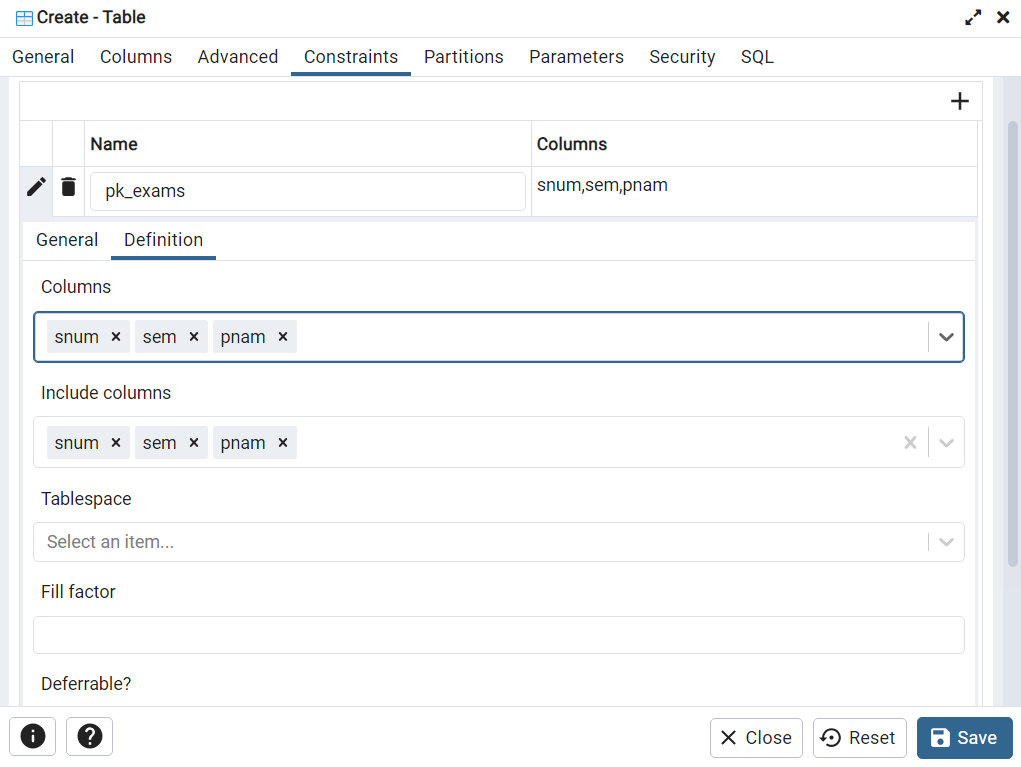
Наложим еще ограничение на имена Sem и Ball, используя CHECK. Результат можно увидеть на рисунке 3.9.

Рисунок 3.9 – ограничение на Sem и Ball

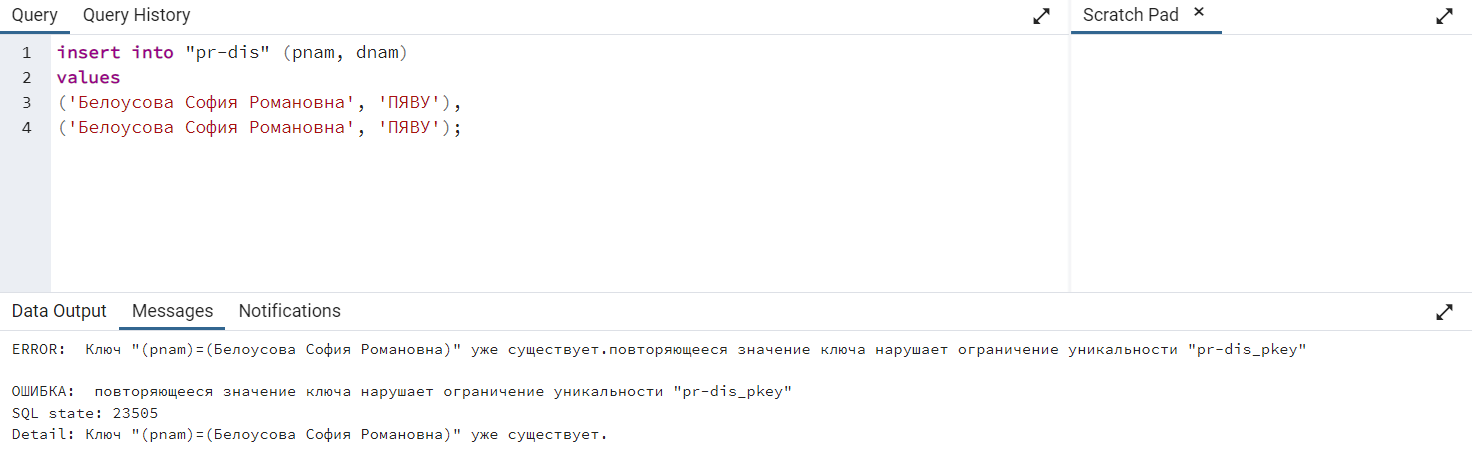
Последнее, что нам нужно сделать, объединить три поля с именами Snum, Sem и Pnam для первичного ключа таблицы. Это можно сделать, перейдя во вкладку Constration внутри формы выбрать Primary key и выбрать те колонки, которые хотим сделать первичным ключом как показано на рисунке 3.10.

Рисунок 3.10 – формирование первичного ключа из более 1 столбца

Проведем проверку наших ограничений целостности и ссылочности для таблиц PR-DIS и EXAMS.

Целостность PR-DIS:

* Pnam - Только буквы русского алфавита и пробелы. Первичный ключ таблицы (рис. 3.11 – 3.12).
* Dnam - Только буквы русского алфавита и пробелы (рис. 3.13).

Рисунок 3.11 – корректность Pnam

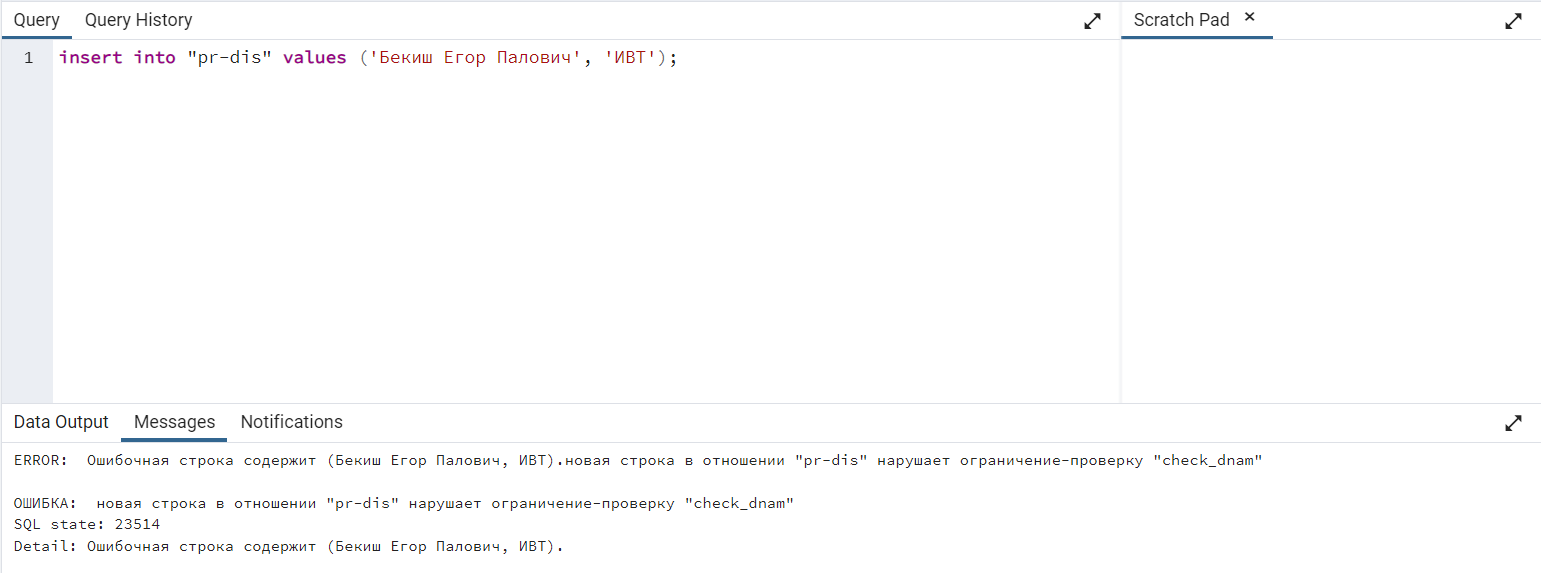
Рисунок 3.12– первичный ключ Pnam

Рисунок 3.13 – корректность Dnam

Целостность EXAMS:

* Sem - От 1 до 10 (рис 3.14).
* Ball - Целое Значения {1, 2, 3, 4, 5} (рис 3.15).

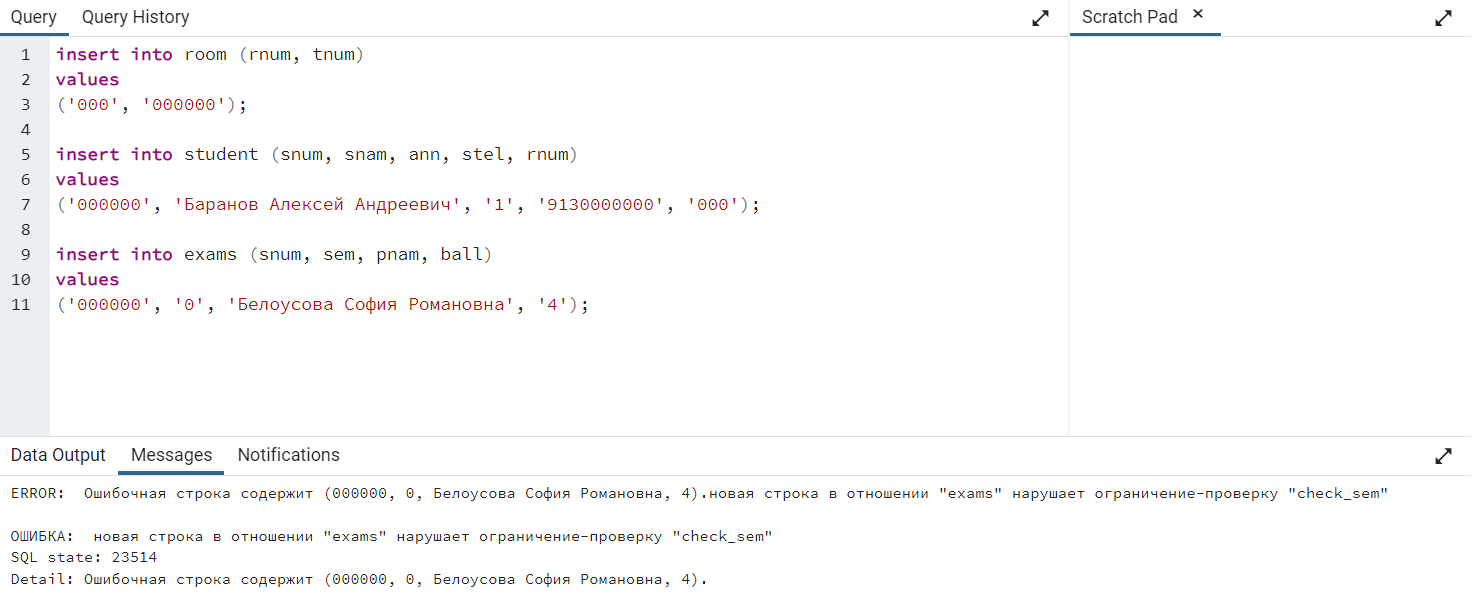


Рисунок 3.14 – корректность Sem

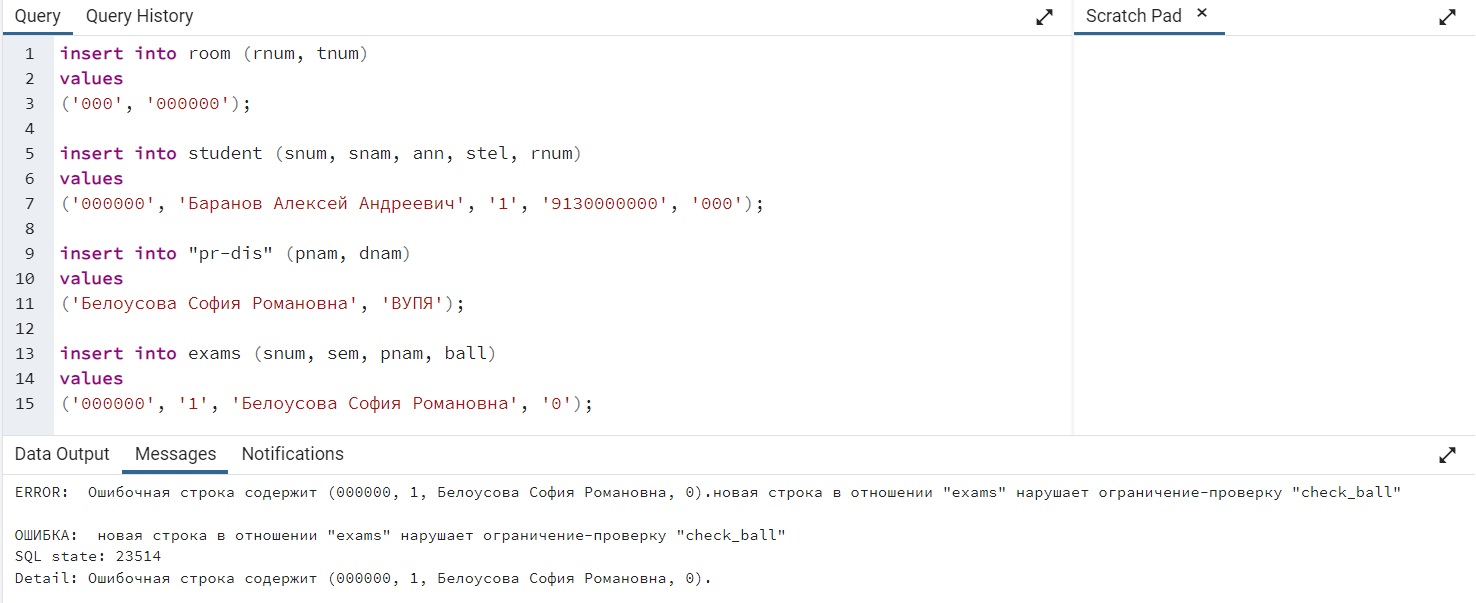


Рисунок 3.15 – корректность Ball

Ссылочность EXAMS:

* Snum - Ссылка на первичный ключ таблицы STUDENT (рис. 3.16).
* Pnam - Ссылка на первичный ключ таблицы PR-DIS (рис. 3.17).
* {Snum, Sem, Pnam} – тройка образует первичный ключ таблицы (рис 3.18).

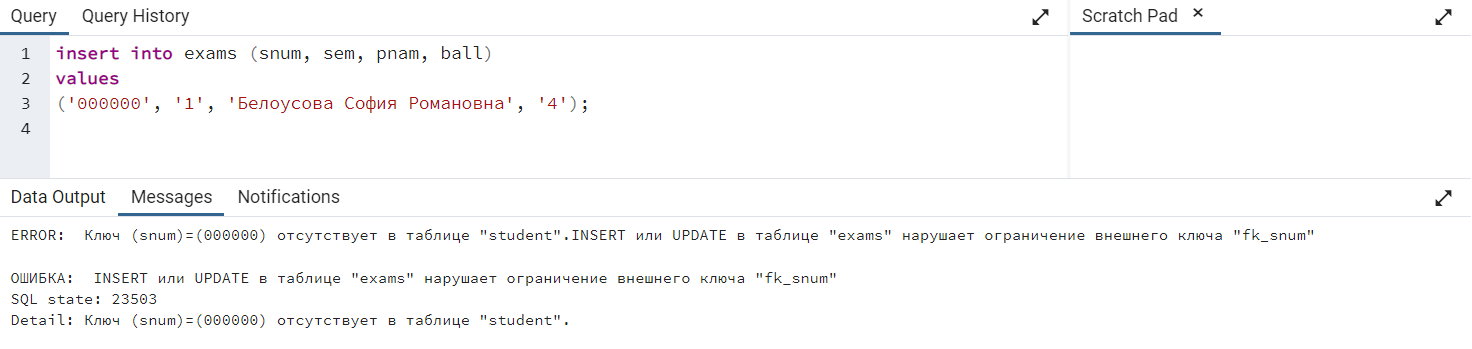
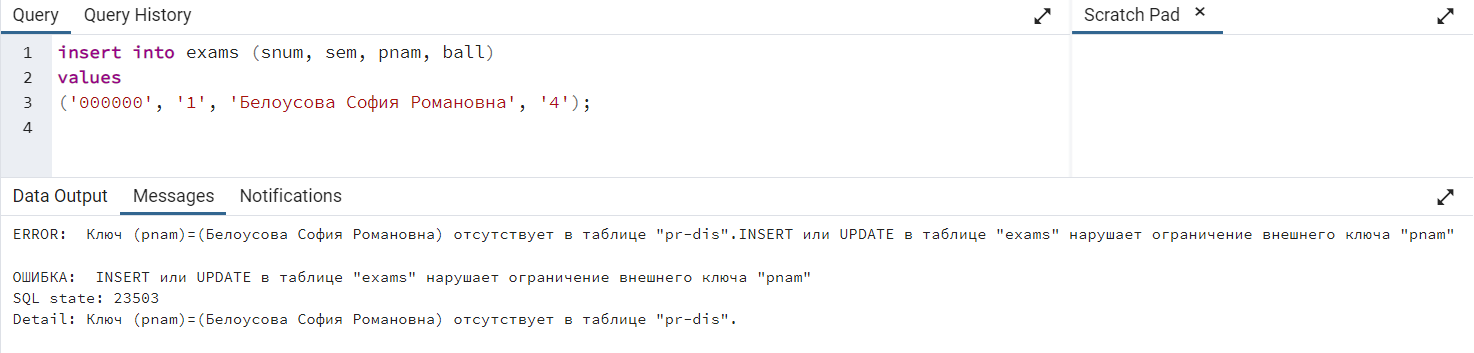
Рисунок 3.16 – ссылочность на первичный клюв из STUDENT

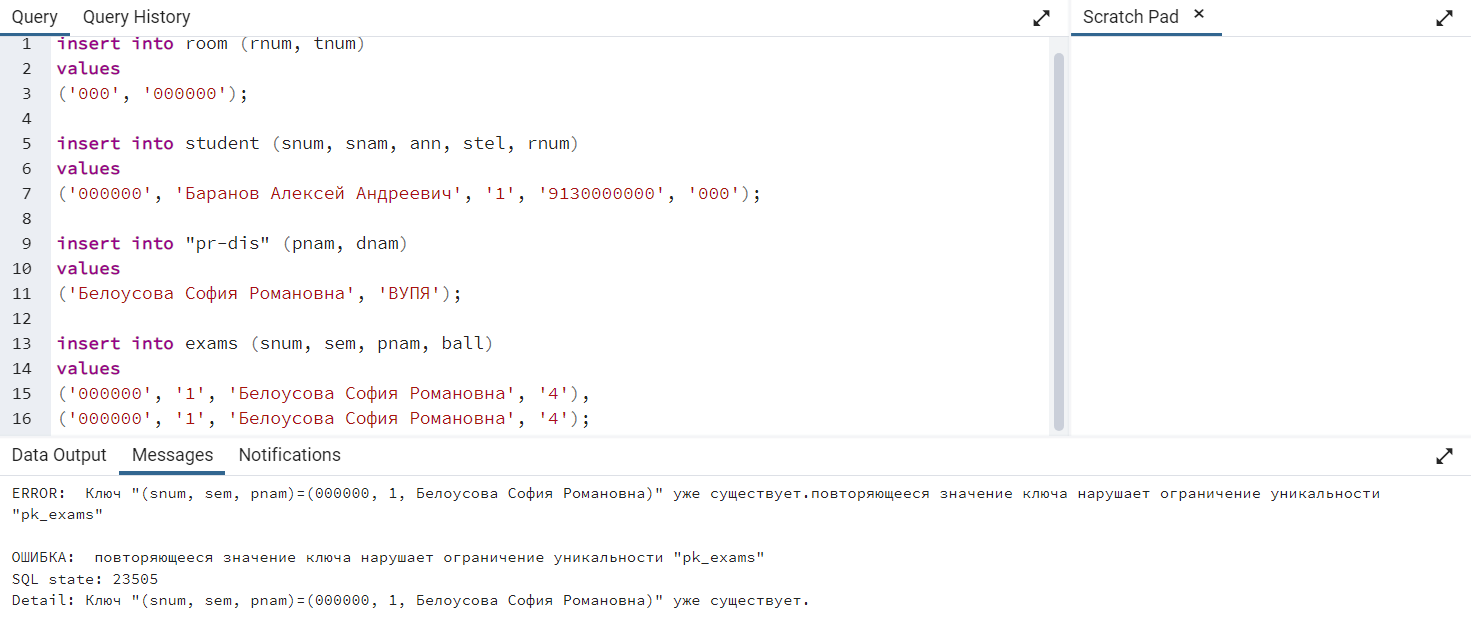
Рисунок 3.17 – ссылочность на первичный клюв из PR-DIS

Рисунок 3.18 – ограничение первичного ключа из тройки Snum, Sem, Pnam

Теперь, когда мы провели все проверки, добавим корректные данные, то есть удовлетворяющие нашим свойствам всем полям из всех созданных таблиц. Результат можно увидеть на рисунках 3.19 – 3.20.

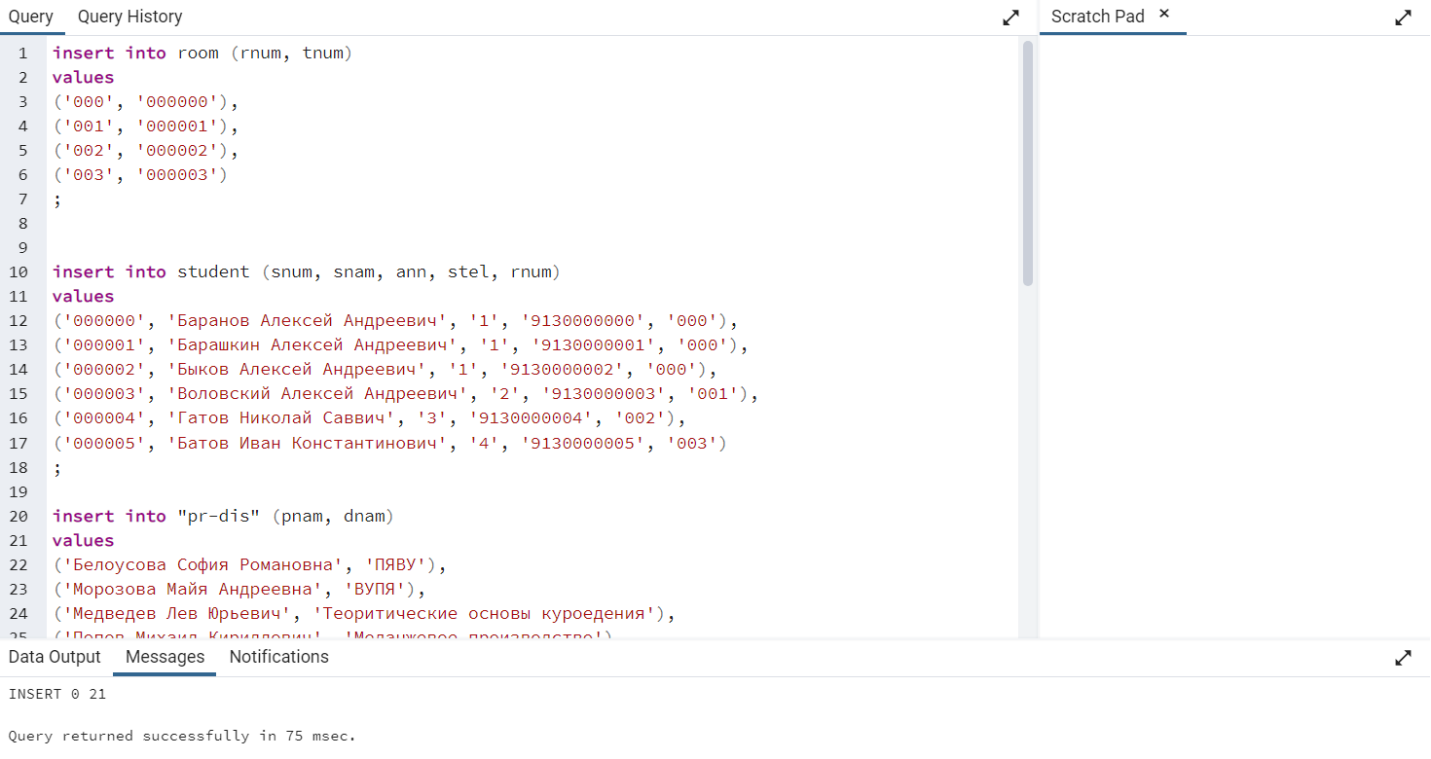
Рисунок 3.19 – успешное добавление данных во все таблицы 

Рисунок 3.20 – успешное добавление данных во все таблицы

**4 Выводы**

В ходе выполнение лабораторной работы я научился пользоваться утилитой pgAdmin, создавать в ней таблицы, ограничивать поля классов требуемыми свойствами, используя в них регулярные выражение при необходимости.

**1 Цель лабораторной работы**

– познакомится с основными командами для формирования запросов в SQL;

– сформировать запросы для индивидуального задания в SQL;

**2 Описание работы**

Используя либо psql, либо pgAdmin создать нижеперечисленные запросы.

1. Получить список студентов первого года обучения с указанием комнат, в которых они проживают. Нужно выбрать столбец с некоторыми полями из определенной таблицы, необходимо использовать SELECT имя\_поля1, имя\_поля2, ..., имя\_поляN FROM имя\_таблицы1, имя\_таблицы2, ..., имя\_таблицыN. Теперь, чтобы задать только некоторую часть таблицы, которая соответствует определенному условию, нужно использовать ключевое слово WHERE имя\_поля. Результат индивидуального задания представлен на рисунке 2.1. Данный запрос выглядит следящим образом: select snam, rnum from student where ann='1'. Выбираем из таблицы имя и комнату студента, в которой он проживает соответствующему году обучения.

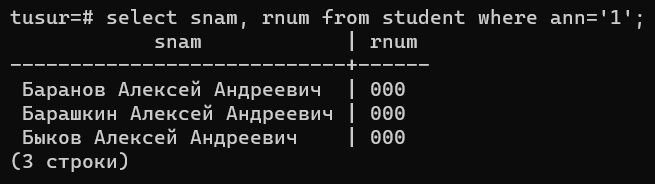
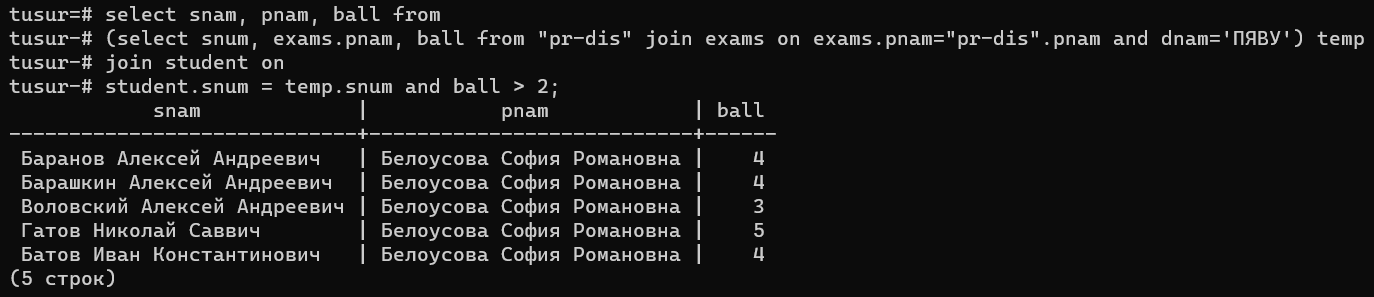


Рисунок 2.1 – успешный результат индивидуального задания

2. Для заданной дисциплины получить список сдавших её студентов с указанием экзаменатора и полученной оценки. Здесь применяем те же действия, что и в первом индивидуальном варианте, сейчас необходимо совершить внешнее соединение полей из разных таблиц. Это можно сделать с помощью оператора JOIN. Имеет следующий формальный синтаксис: SELECT имя\_поля1, имя\_поля2, ..., имя\_поляN FROM имя\_таблицы1, имя\_таблицы2, ..., имя\_таблицыN JOIN имя\_таблицы2 ON условие1 JOIN таблица2 ON условие2. Перед оператором JOIN указывается одно из ключевых слов LEFT, RIGHT или FULL, которые определяют тип соединения:

* LEFT: выборка будет содержать все строки из первой или левой таблицы.
* RIGHT: выборка будет содержать все строки из второй или правой таблицы.
* FULL: выборка будет содержать все строки из обеих таблиц.

Перед оператором JOIN может указываться ключевое слово OUTER, но его применение необязательно. После JOIN указывается присоединяемая таблица, а затем идет условие соединения после оператора ON.

Данный запрос выглядит следящим образом: select snam, pnam, ball from (select snum, exams.pnam, ball from "pr-dis" join exams on exams.pnam="pr-dis".pnam and dnam='ПЯВУ') temp join student on student.snum = temp.snum and ball > 2. Создадим таблицу select snum, exams.pnam, ball from "pr-dis" join exams on exams.pnam="pr-dis".pnam and dnam='ПЯВУ', в которой будет находится поля с номером студенческого билета, имена преподавателей и полученный балл за дисциплину. Из данной таблицы будем выбирать имена студентов, имена преподавателей, оценку в том случае, если в данной таблицы имеется студенческий билет, который присутствует в таблицы STUDENT и полученная оценка за дисциплину выше двойки. Результат индивидуального задания представлен на рисунке 2.2.

Рисунок 2.2 – успешный результат индивидуального задания

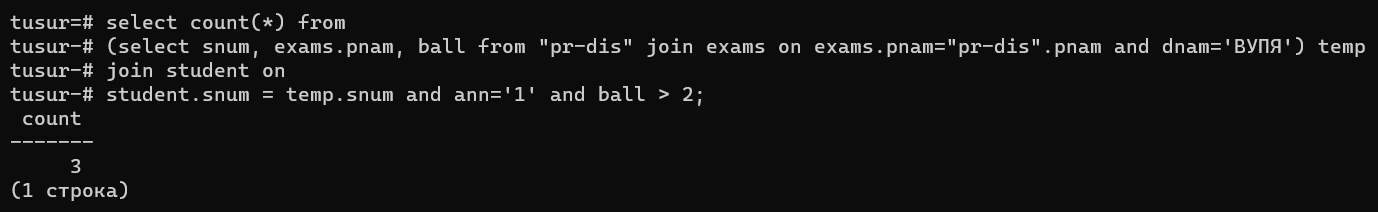
3. Подсчитать количество студентов указанного года обучения, сдавших указанную дисциплину. Теперь сформируем запрос на основе второго индивидуального задания только необходимо показать не поля определенных имен, а количество строк в полученной таблице. Это можно сделать при помощи агрегатной функции COUNT. Существует две формы использования:

* COUNT (\*): находит количество строк в запросе
* COUNT (expression): находит количество строк в запросе, для которых expression не содержит значение NULL.

Функция COUNT вычисляет количество строк в выборке. Есть две формы этой функции. Первая форма COUNT (\*) подсчитывает число строк в выборке: SELECT COUNT (\*) FROM имя\_таблицы.

Вторая форма функции вычисляет количество строк по определенному столбцу, при этом строки со значениями NULL игнорируются: SELECT COUNT (DISTINCT имя\_поля) FROM имя\_таблицы.

Данный запрос выглядит следящим образом: select count(\*) from (select snum, exams.pnam, ball from "pr-dis" join exams on exams.pnam="pr-dis".pnam and dnam='ВУПЯ') temp join student on student.snum = temp.snum and ann='1' and ball > 2. Создадим таблицу select snum, exams.pnam, ball from "pr-dis" join exams on exams.pnam="pr-dis".pnam and dnam='ВУПЯ', в которой будут находится поля с номером студенческого билета, имена преподавателей и полученный балл за дисциплину. Из данной таблицы будем выбирать имена студентов, имена преподавателей и оценку в том случае, если в этой таблицы имеется студенческий билет, который присутствует в таблице STUDENT и полученная оценка за дисциплину выше двойки с указанным годом обучения. После получаем таблицу с определенным количествам строк и считаем их. Результат индивидуального задания представлен на рисунке 2.3.

Рисунок 2.3 – успешный результат индивидуального задания

4. Для каждого студента подсчитать средний балл. Еще раз повторим построение запроса из второго и третьего индивидуального задания. И для подсчета среднего значения тоже будем использовать агрегатную функцию AVG. Входной параметр должен представлять один из следующих типов: smallint, int, bigint, real, double precision, numeric, interval. Для целочисленных параметров результатом будет значение типа numeric, для параметров, которые представляют число с плавающей точкой, - значение типа double precision. Функция AVG возвращает среднее значение на диапазоне значений столбца таблицы. Найдем среднее значение таблицы из базы данных: SELECT AVG (имя\_поля) AS Average\_Price FROM имя\_таблицы.

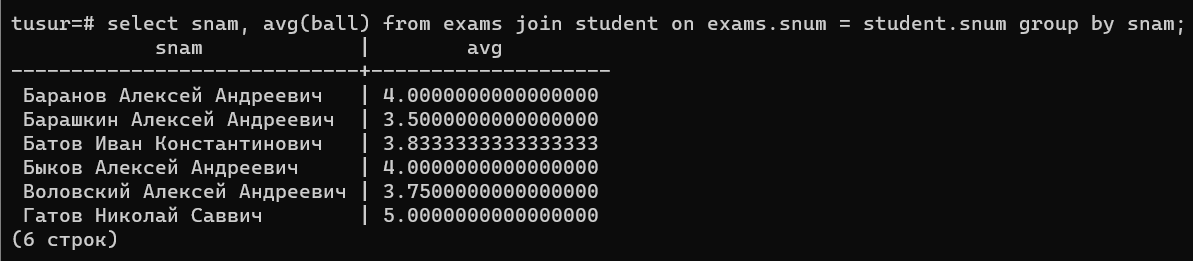
Данный запрос выглядит следящим образом: select snam, avg(ball) from exams join student on exams.snum = student.snum group by snam. Из соединенных таблиц STUDENT и EXAMS необходима выбрать поля с именем студента и его среднеарифметической оценкой за все сданные дисциплины. Результат индивидуального задания представлен на рисунке 2.4.

Рисунок 2.4 – успешный результат индивидуального задания

**3 Выводы**

В ходе выполнение лабораторной работы я научился формировать запросы из перечисленных запросов, которые были представлены в индивидуальных заданиях.

**1 Цель лабораторной работы**

– Разработать программу, которая соответствует условиям индивидуального задания на любом языке программирование.

**2 Описание требований к отчётам**

1. Выполняет подключение к серверу СУБД PostgreSQL.

2. Заносит новые данные в одну из таблиц (по выбору студента).

****3. Формирует на экране (или в файле) отчет как показано на рисунке 2.1.

Рисунок 2.1 – успешный результат индивидуального задания

**3 Листинг**

import psycopg2

originall = {

'dbname': 'tusur',

'user': 'postgres',

'password': '123',

}

connect = psycopg2.connect(\*\*originall)

cursor = connect.cursor()

connect.autocommit = True

query = """

insert into room (rnum, tnum)

values

('000', '000000'),

('001', '000001'),

('002', '000002'),

('003', '000003')

;

insert into student (snum, snam, ann, stel, rnum)

values

('000000', 'Баранов Алексей Андреевич', '1', '9130000000', '000'),

('000001', 'Барашкин Алексей Андреевич', '1', '9130000001', '000'),

('000002', 'Быков Алексей Андреевич', '1', '9130000002', '000'),

('000003', 'Воловский Алексей Андреевич', '2', '9130000003', '001'),

('000004', 'Гатов Николай Саввич', '3', '9130000004', '002'),

('000005', 'Батов Иван Константинович', '4', '9130000005', '003')

;

insert into "pr-dis" (pnam, dnam)

values

('Белоусова София Романовна', 'ПЯВУ'),

('Морозова Майя Андреевна', 'ВУПЯ'),

('Медведев Лев Юрьевич', 'Теоритические основы куроедения'),

('Попов Михаил Кириллович', 'Меланжевое производство'),

('Кулаков Роман Родионович', 'Технология собаковыгуливания'),

('Мальцев Матвей Ильич', 'Технология приготовления яичницы')

;

insert into exams (snum, sem, pnam, ball)

values

('000000', '1', 'Белоусова София Романовна', '4'),

('000000', '1', 'Морозова Майя Андреевна', '4'),

('000000', '1', 'Медведев Лев Юрьевич', '4'),

('000001', '1', 'Белоусова София Романовна', '4'),

('000001', '2', 'Морозова Майя Андреевна', '3'),

('000002', '1', 'Морозова Майя Андреевна', '5'),

('000002', '1', 'Медведев Лев Юрьевич', '3'),

('000002', '1', 'Попов Михаил Кириллович', '4'),

('000003', '1', 'Белоусова София Романовна', '3'),

('000003', '2', 'Медведев Лев Юрьевич', '5'),

('000003', '3', 'Кулаков Роман Родионович', '3'),

('000003', '4', 'Морозова Майя Андреевна', '4'),

('000004', '3', 'Белоусова София Романовна', '5'),

('000004', '3', 'Медведев Лев Юрьевич', '5'),

('000004', '3', 'Попов Михаил Кириллович', '5'),

('000005', '1', 'Белоусова София Романовна', '4'),

('000005', '2', 'Морозова Майя Андреевна', '3'),

('000005', '3', 'Кулаков Роман Родионович', '5'),

('000005', '4', 'Попов Михаил Кириллович', '3'),

('000005', '5', 'Медведев Лев Юрьевич', '5'),

('000005', '6', 'Мальцев Матвей Ильич', '3')

;"""

cursor.execute(query)

def foo(snum):

query = '''select \* from "pr-dis";'''

cursor.execute(query)

pr\_dis = cursor.fetchall()

pr\_dis\_dict = {pr\_dis[i][0]: pr\_dis[i][1] for i in range(len(pr\_dis))}

query = f'''select sem, pnam, ball from exams where snum='{snum}';'''

cursor.execute(query)

exams = cursor.fetchall()

query = f'''select avg(ball) from exams where snum='{snum}';'''

cursor.execute(query)

middle = str(cursor.fetchone()[0])

if sum(list(map(int, middle[2:4]))) == 0:

middle = int(middle[:middle.index('.')])

elif middle[3] == '0' or middle[3] != '5':

middle = middle[:3]

else:

middle = middle[:4]

s = 'Дисциплина\t\t\t\t\t\tОценка\tСеместр\n'

for info in exams:

sem, pnam, ball = info

s += f'\t\t\t\t\t\t\t\t\t {pr\_dis\_dict[pnam]:35}{ball}{sem:10}\n'

s += f'\t\t\t\t\t\t\t\t\t\t\t {"Средний балл":25}{middle}'

return s

query = '''select ann from student;'''

cursor.execute(query)

ann = set(cursor.fetchall())

file = open('laba4.txt', 'w')

for year in ann:

query = f'''select \* from student where ann={year[0]}'''

cursor.execute(query)

student\_info = cursor.fetchall()

print(f'Год обучения:{year[0]}')

for info in student\_info:

snum, snam, ann, tnum, rnum = info

print(f'{snam}\t\t\t{foo(snum)}\n')

query = """

delete from exams;

delete from student;

delete from room;

delete from "pr-dis";

"""

cursor.execute(query)

cursor.close()

connect.close()

**4 Описание процесса реализации**

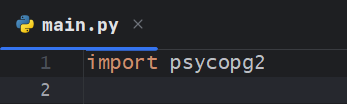
В ходе выполнения лабораторной работы будет использовать язык Python. Создаем файл с расширение .py. Теперь для использования PostgreSQL необходимо импортировать библиотеку psycopg2 как показано на рисунке 4.1.

Рисунок 4.1 – импорт библиотеки

Для того, чтобы подключиться к нашей базе данных необходимо использовать метод connect, в котором необходимо указать базу данных, если она существует (по умолчанию подключаемся к postgres), пароль и пользователя (нужно указать postgres). Для удобства создадим словарь, в котором укажем в качестве ключей необходимые название параметров и их значения. Вообще cсловарь — неупорядоченная структура данных, которая позволяет хранить пары «ключ — значение».

Пример: dictionary = {key: value}

Теперь создаем курсор — область памяти базы, предназначенная для хранения последнего оператора SQL. Иными словами, объект, отвечающий и за отправку запросов, и за получение их результатов. Реальное выполнение команд производится только при выполнении метода conn.commit(). Если же надо, чтобы выражения SQL автоматически выполнялись при каждом вызове метода cursor.execute(), то можно установить автокоммит с помощью свойства connection.autocommit. Где выполнением команды SQL автоматически создается транзакция, в процессе которой можно выполнять различные выражения SQL с помощью методов execute/executemany курсора, но для подтверждения их выполнения необходимо вызывать метод commit() объекта connection.

* execute(query, vars=None): выполняет одну SQL-инструкцию. Через второй параметр в код SQL можно передать набор параметров в виде списка или словаря
* executemany(query, vars\_list): выполняет параметризованное SQL-инструкцию. Через второй параметр принимает наборы значений, которые передаются в выполняемый код SQL.

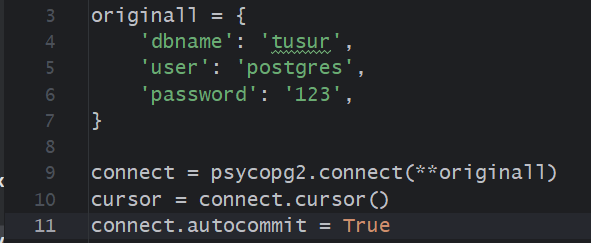
Результат представлен на рисунке 4.2.

Рисунок 4.2 – авторизация

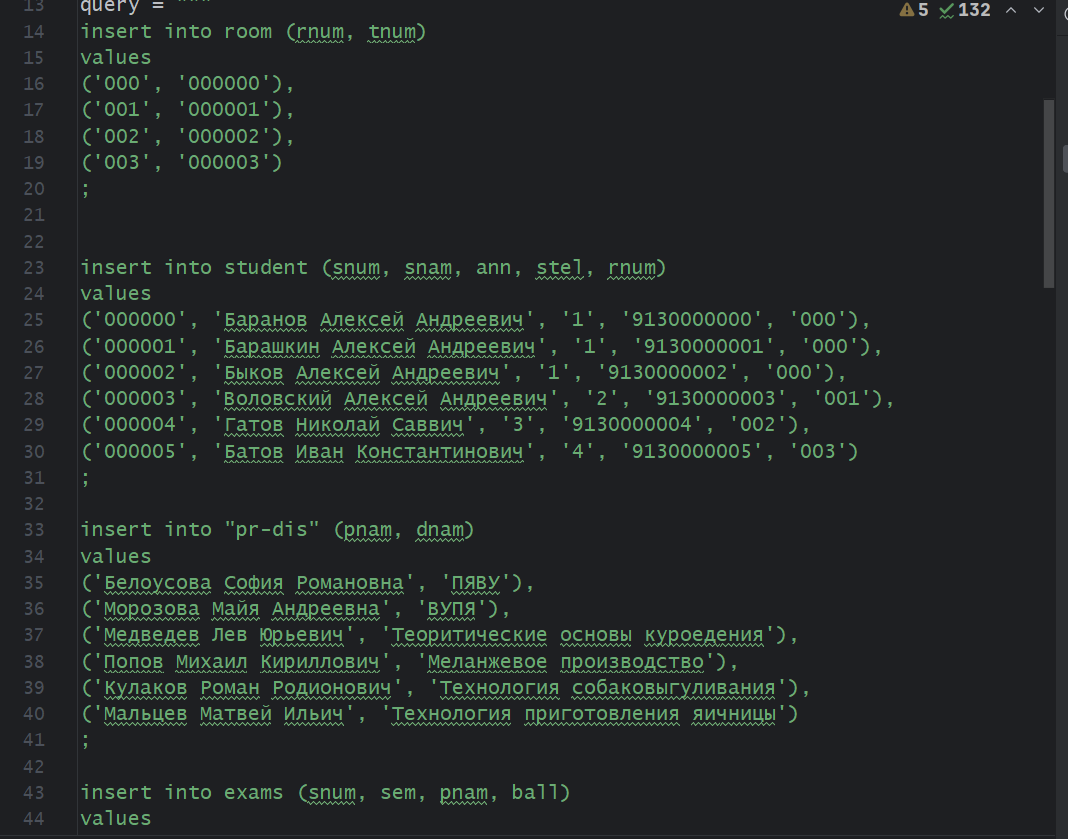
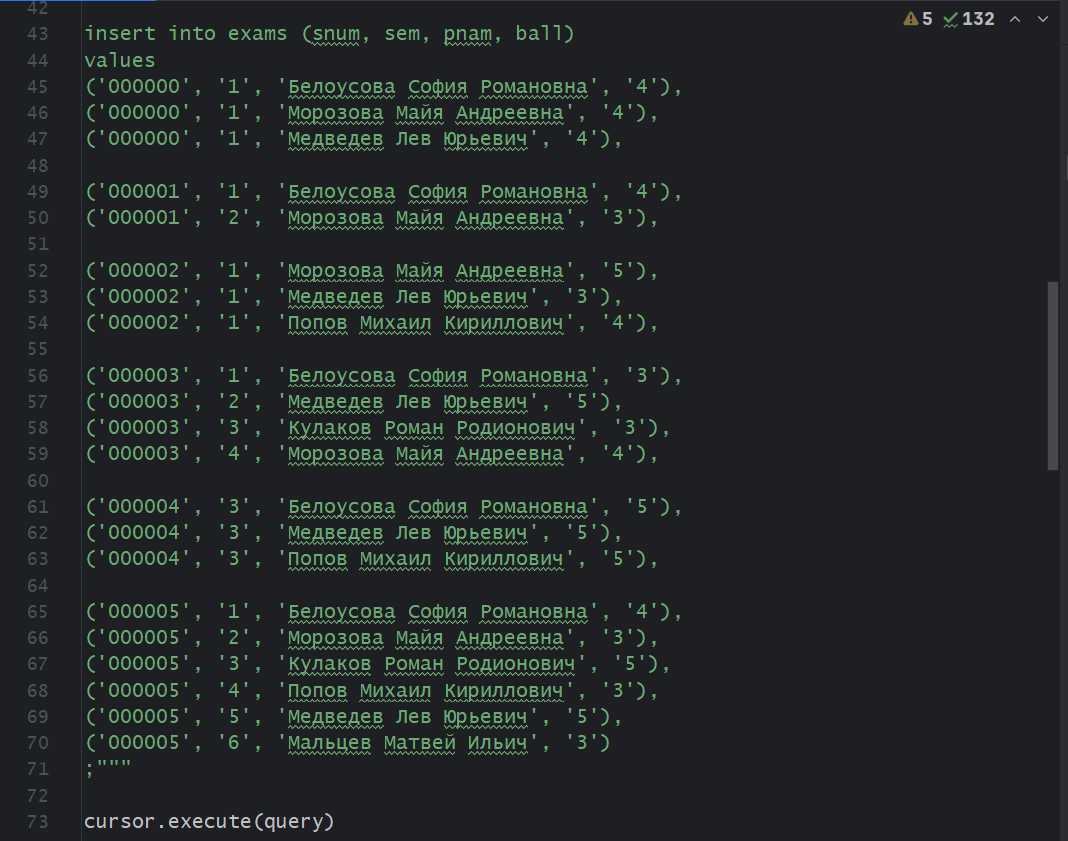
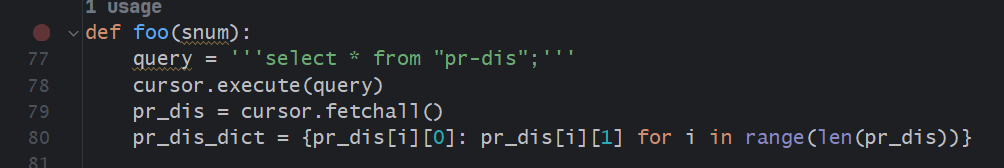
Создадим запрос по добавлению данных в наши созданные таблицы как показано на рисунках 4.3 – 4.4.

Рисунок 4.3 – добавление данных

Рисунок 4.4 – добавление данных

После добавления данных создадим функцию, которая будет выдать необходимую информацию об успеваемости. Так как таблицы STUDENT и EXAMS связаны номером студенческого билета, это будет наша переменная, по которой мы сможем брать информацию из данных таблиц. Но в таблице EXAMS хранятся имена преподавателей, поэтому необходимо сформировать словарь по именам преподавателей, которые ведут этот предмет, чтобы имя преподаватель менялось на название дисциплины. Можно это сделать с помощью кортежа словаря, где ключ будет иметь значение имя преподавателя, а значение имеет дисциплину. Результат представлен на рисунке 4.5.

Рисунок 4.5 – словарь дисциплин

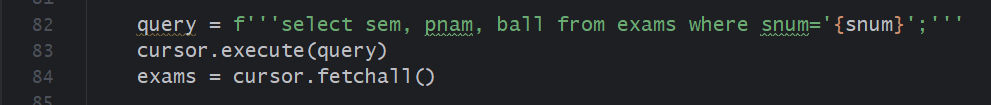
Далее нам необходимо сделать запрос в таблицу EXAMS по студенческому билету. Выбрать информацию о экзаменаторе, семестре, в котором была сдана данная дисциплина и полученная за данную дисциплину оценка (рис. 4.6).

Рисунок 4.6 – информацию о экзамене

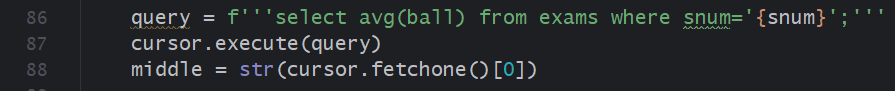
Теперь нам необходимо получить среднее значение за все сданные дисциплины. Это можно сделать с помощью агрегатной функции AVG представленной на рисунке 4.7.

Рисунок 4.7 – информацию о экзамене

После полученного нами среднего числа у нас возникает формат decimal, то есть десятичное число, которая содержит в себе погрешности c шестнадцатью нулей, и чтобы от них избавиться конвертируем decimal в строку. Проверим, что если наши два значения после запятой это два нуля, то мы конвертирует в число, если же второй знак после запятой ноль, либо же не пять, то делаем срез до него, в противном случае срезаем до второго знака включительно (рис. 4.8).

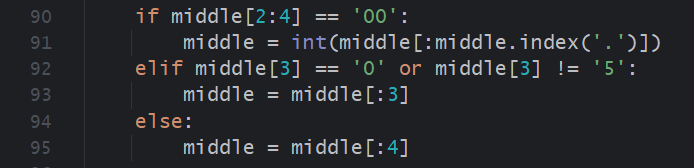


Рисунок 4.8 – средняя оценка

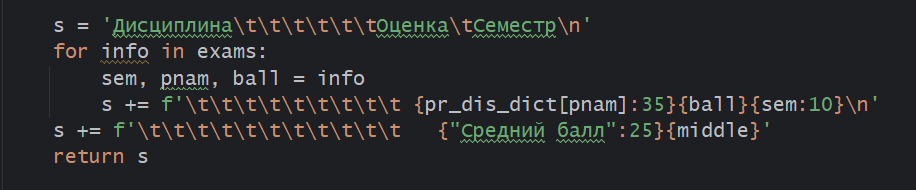
Когда мы получили все необходимые данные, нам нужно вывести всю информацию об студенте по его студенческому билету. Мы формируем строку, в которой будем хоронить всю информацию и ее возвращаем. Результат представлен на рисунке 4.9.

Рисунок 4.9 – получение информации

Теперь необходимо вывести информацию по году обучение, мы делаем запрос в таблицу STUDENT, где выбираем год обучения и формируем множество, потому что в нашей таблице могут быть несколько студентов, которые обладают одним и тем же годом обучение. Дальше создаем цикл, где проходимся по студентам данного года обучения, делаем запрос в таблицу, где выбираем номер студенческого билета, чтобы добыть информацию о данных дисциплины и его имя, после этого выводим год обучения и ту информацию, которую мы получили из нашей функции и закрываем нашу базу данных (рис. 4.10).

Рисунок 4.10 – вывод необходимой информации

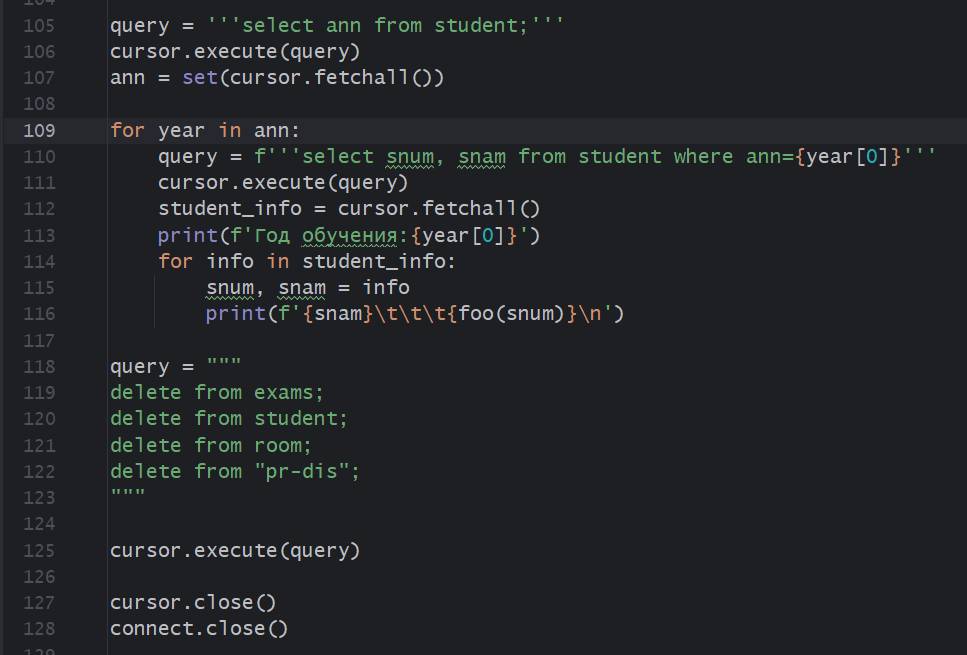
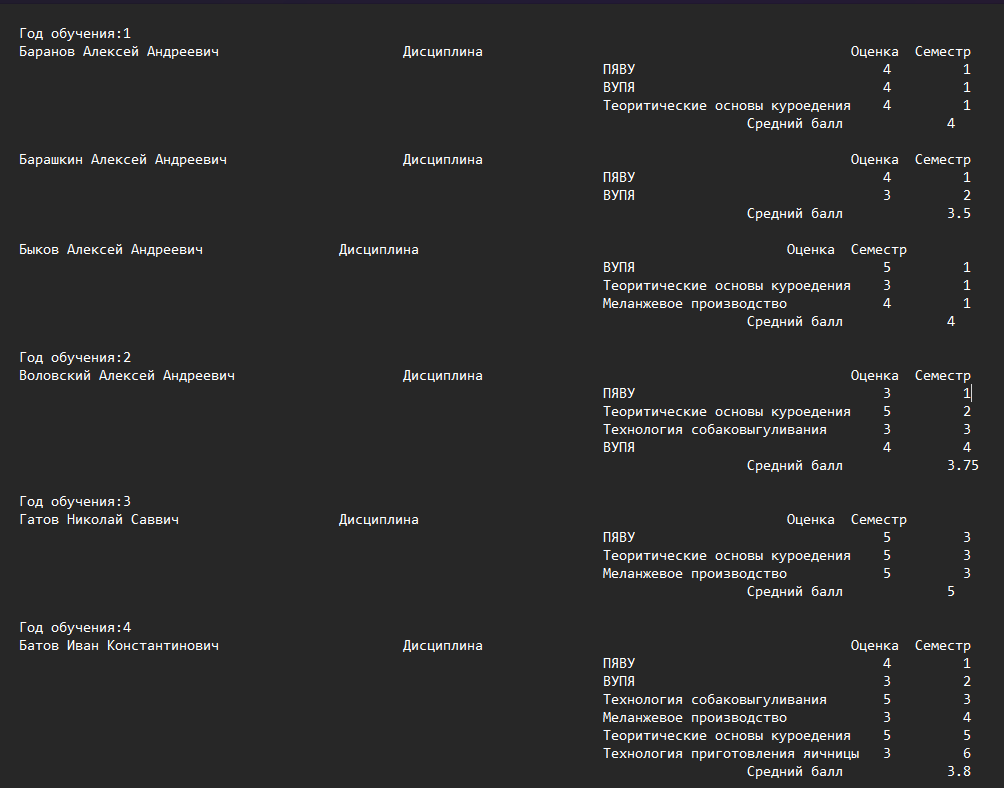
Результат работы все программы представлен на рисунке 4.11.

Рисунок 4.11 – успешное выполнение программы

**5 Выводы**

В ходе выполнение лабораторной работы я научился работать с языком Python, подключаться через библиотеку к базе данных, получать данные из таблиц, заносить данные в таблицы.