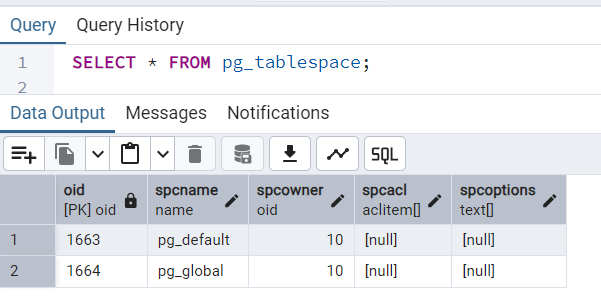
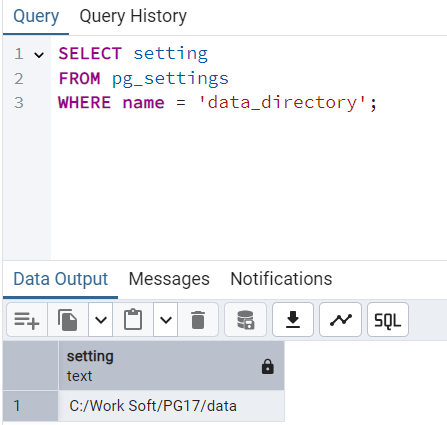
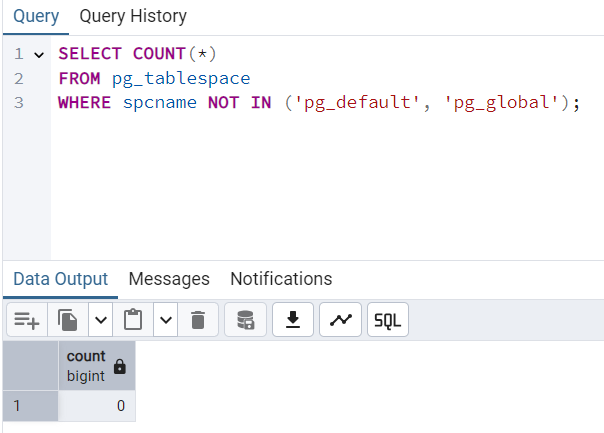
Задание 9.1. Ознакомьтесь с данными о файлах и файловых группах, выполнив запрос и просмотрев содержимое соответствующих системных представлений.



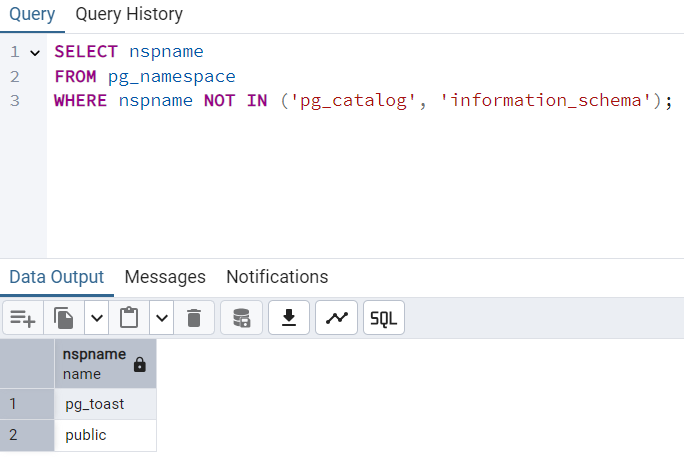
Задание 9.2. Напишите запрос, выводящий расположение файла (файлов) журнала на диске. Чтобы отобрать нужные записи, используйте условие на столбец type.

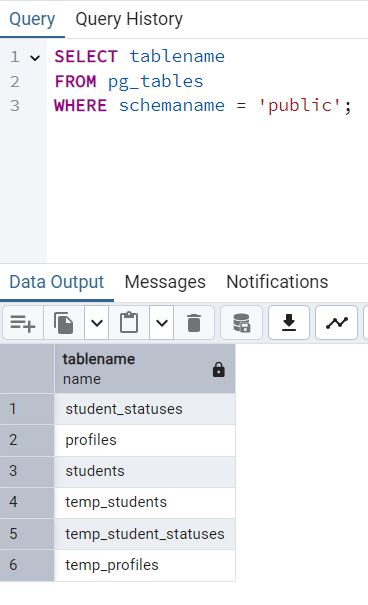


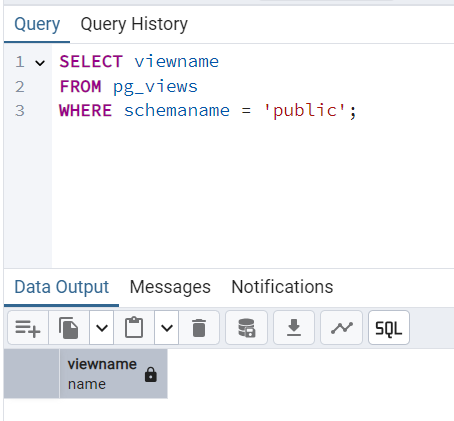
Задание 9.3. Напишите запрос, подсчитывающий количество файловых групп базы данных, доступных на чтение и запись.



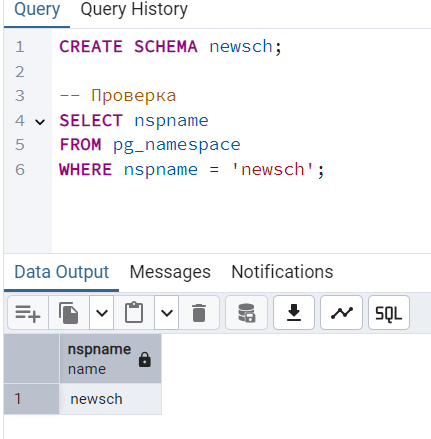
Задание 9.4. Используя системные представления sys.schemas, sys.tables, sys.views, получите сведения о схемах, таблицах и представлениях, определенных в базе данных STUDENTS.



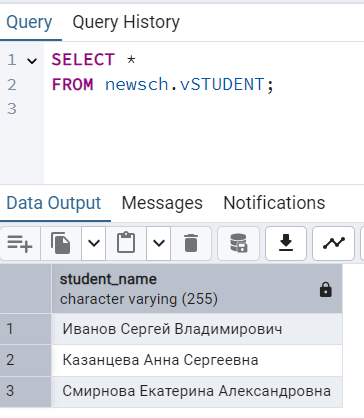




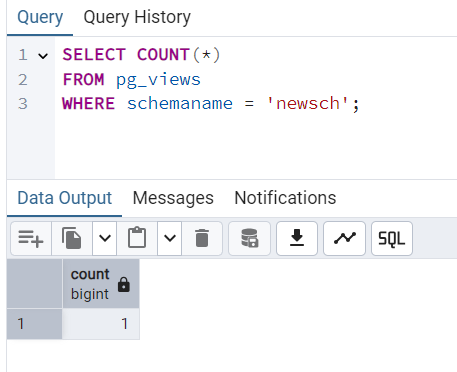
Задание 9.5. Создайте новую схему. Проверьте, что данные о созданной схеме отображаются в представлении sys.schemas.



Задание 9.6. В схеме newsch создайте представление vSTUDENT, содержащее один столбец, в котором перечислены ФИО студентов из таблицы STUDENT. Посмотрите данные из представления. Не забудьте, что newsch не является схемой по умолчанию, поэтому надо указывать имя представления вместе с схемой.

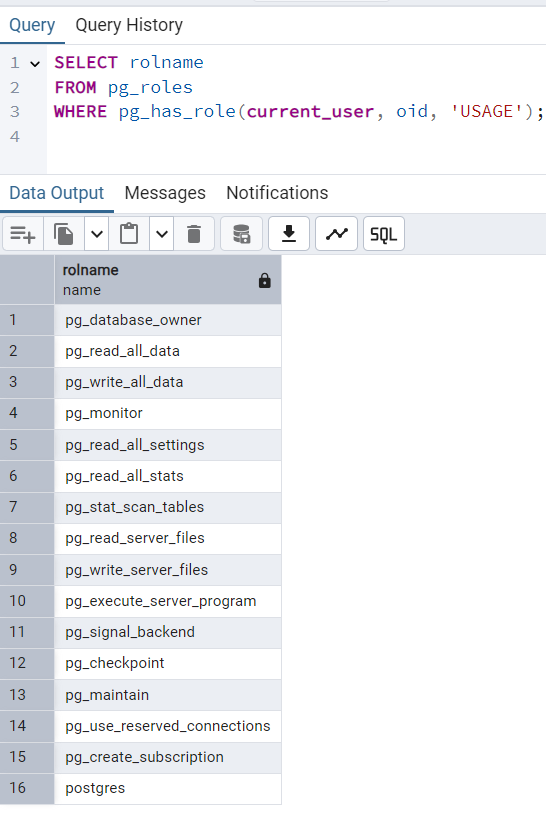


Задание 9.7. Используя рассмотренные ранее системные представления, посчитайте количество представлений в схеме newsch.

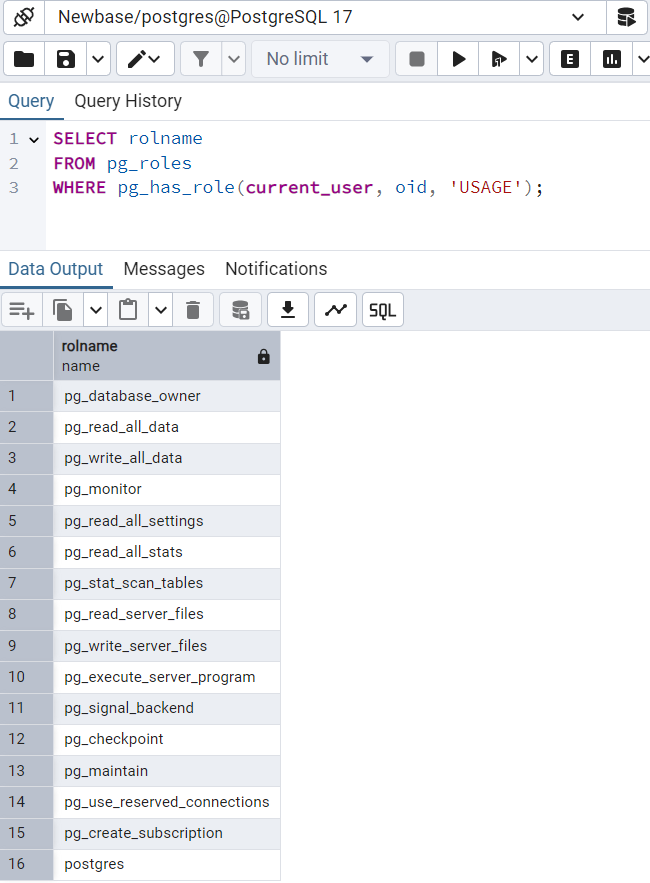


Задание 9.8 -

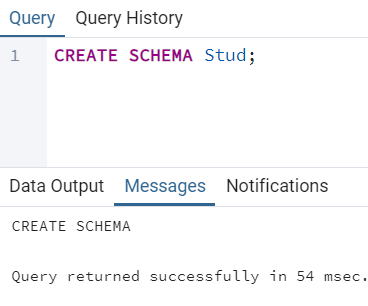
Задание 9.9. В окне Object Explorer (Обозреватель объектов) откройте список учетных записей (logins). На выполнение каких серверных ролей авторизована 81 используемая вами учетная запись? В каких базах данных сервера вашей учетной записи сопоставлены пользователи? На выполнение каких ролей они авторизованы?

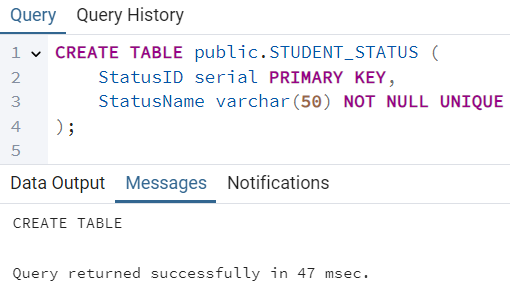


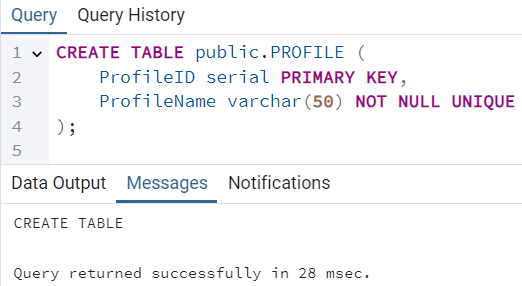
Задание 9.10. Создайте новую базу данных Newbase. Откройте список пользователей и ролей. Убедитесь, что учетная запись, под которой вы работаете, сопоставлена пользователю dbo, авторизованному на роль db\_owner.

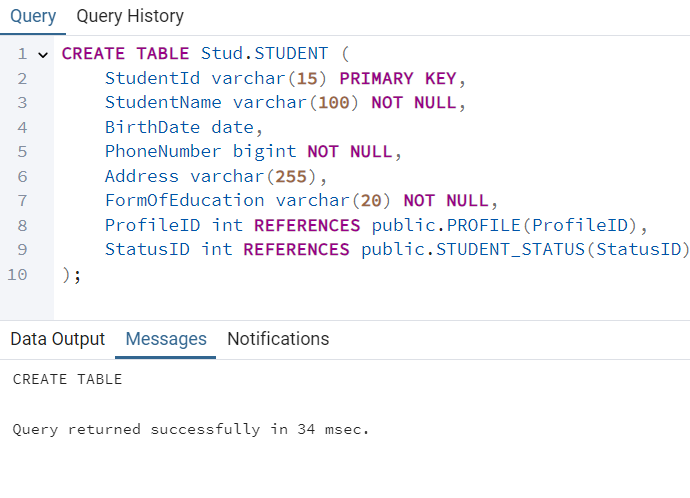


Задание 9.11. Используя приведенный ниже скрипт, создайте в базе данных таблицы. Обратите внимание, что приведенный скрипт создает не только три таблицы, но и схему Stud с помощью оператора CREATE SCHEME.

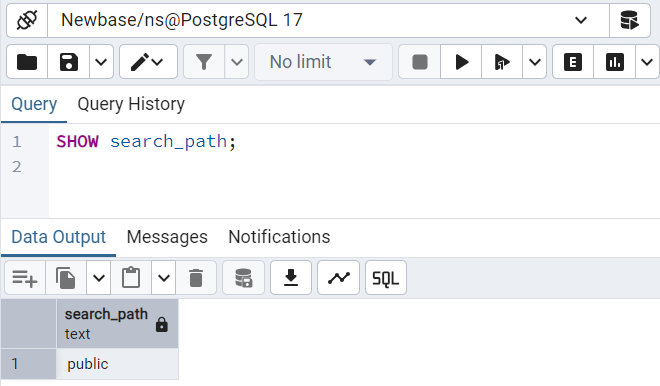








Задание 9.13. Создайте пользователя в вашей базе данных (в примерах ниже имя пользователя – «ns»), в качестве схемы по умолчанию выберите dbo (public postgres).



Задание 9.14. Добавьте этого пользователя в роль db\_datareader. Это можно сделать через графический интерфейс или с помощью системной хранимой процедуры sp\_addrolemember, первым параметром которой будет имя роли, а вторым – имя пользователя

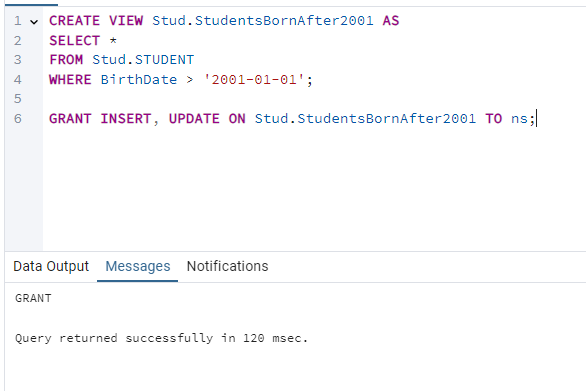


*Задание 9.15. Создайте новую роль уровня базы данных и добавьте ей разрешение на удаление (DELETE), изменение (UPDATE) и добавление данных (INSERT) в объекты схемы Stud. Добавьте нашего пользователя к этой роли.*

CREATE ROLE db\_datawriter;

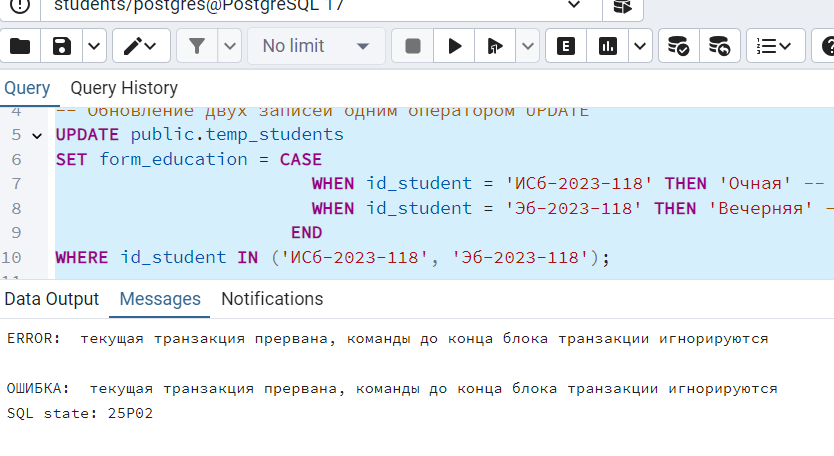
GRANT DELETE, UPDATE, INSERT ON ALL TABLES IN SCHEMA Stud TO db\_datawriter;

GRANT db\_datawriter TO ns;





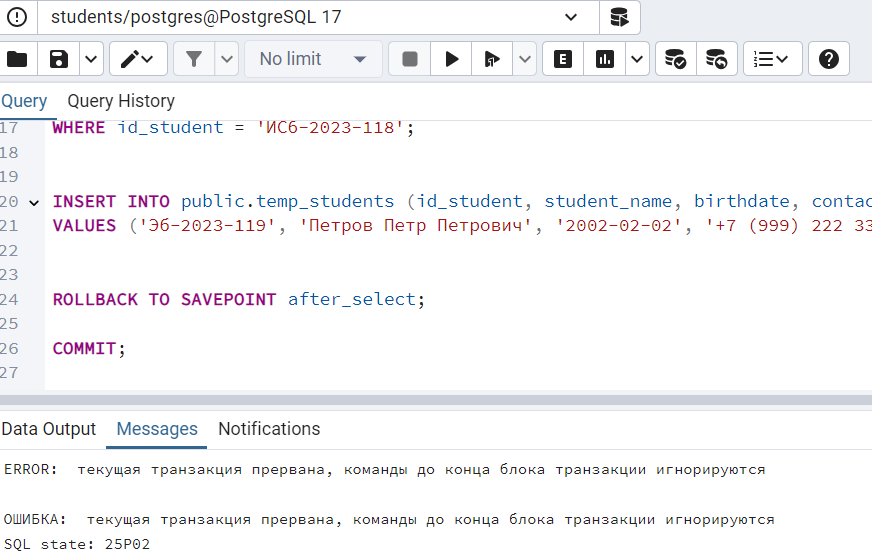
Задание 9.17. Обновите две записи о студентах в таблице STUDENT одним оператором UPDATE так, чтобы одно из изменений не могло быть произведено (транзакция должна быть целиком откачена).



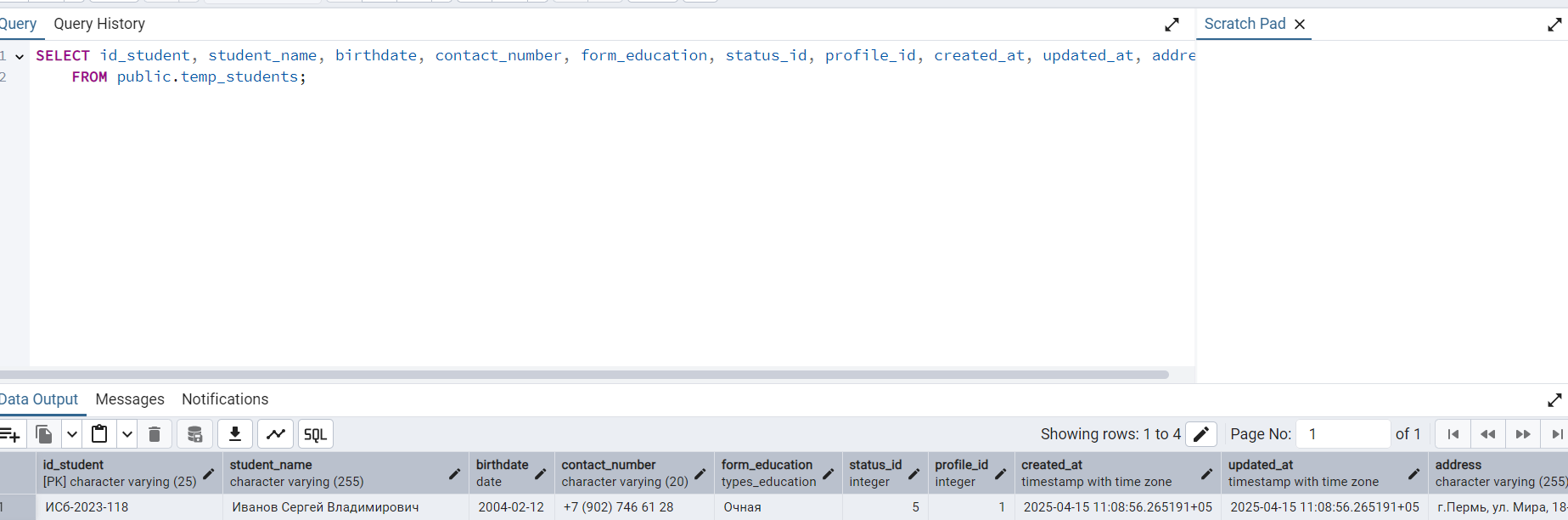
9.18 – СКИП

Задание 9.20. Между операторами UPDATE и INSERT кода транзакции, созданной на предыдущем шаге, поставьте оператор SELECT, выбирающий все записи из соответствующей таблицы. После отката транзакции выполните аналогичный SELECT. Убедитесь, что внутренний SELECT «видит» производимые изменения, т.е. внутри транзакции эти данные доступны.

Задание 9.21. Дополните транзакцию точкой сохранения, установленной после внутреннего SELECT; выполните откат до точки сохранения (в ROLLBACK TRAN указываете имя точки сохранения, а не транзакции), ваш код должен выглядеть примерно так:



Задание 9.22. Обратите внимание, что на момент окончания задания 9.21 транзакция не завершена. Откройте новое окно и выполните в нем следующий код:

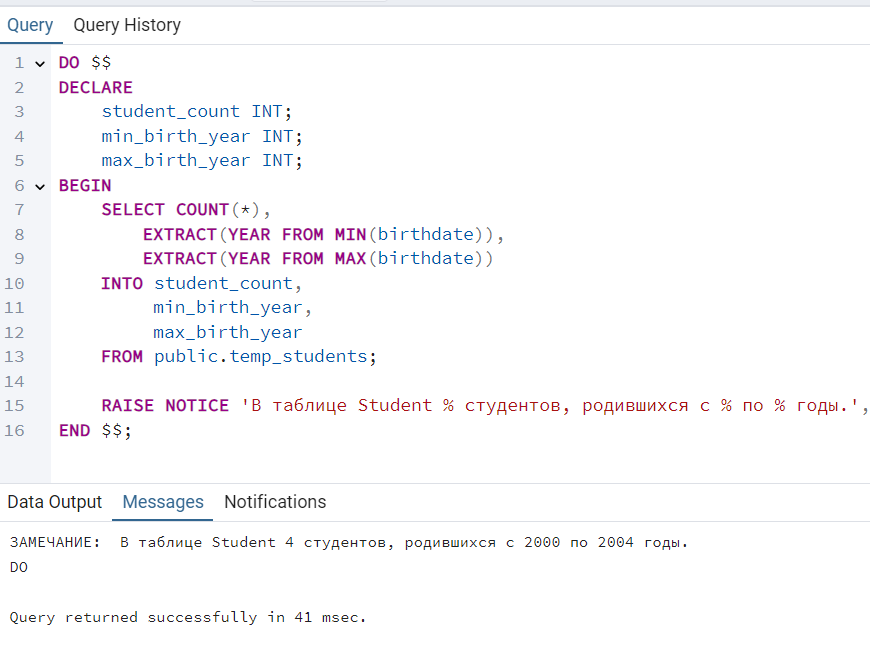


9.23 – скип

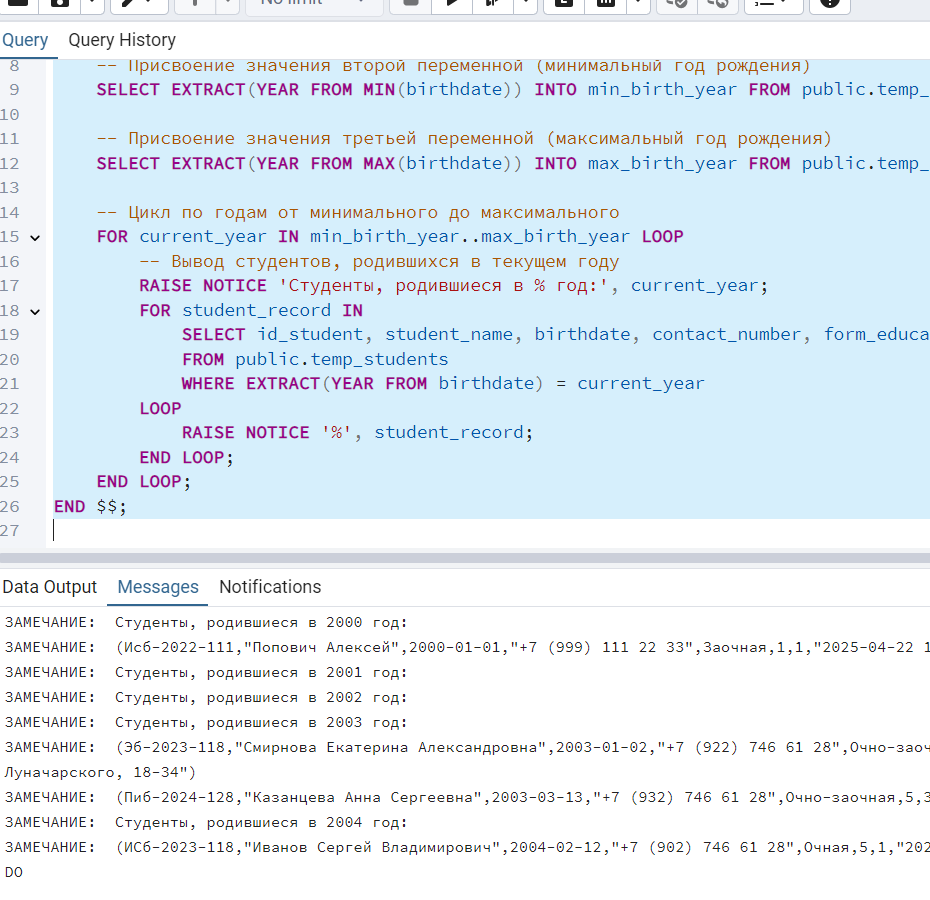
Задание 9.24. Объявите три целочисленные переменные. Первой из них присвойте значение, равное количеству студентов в таблице STUDENT, второй – минимальный года рождения (самые старшие студенты); третьей – максимальный (самые младшие студенты).



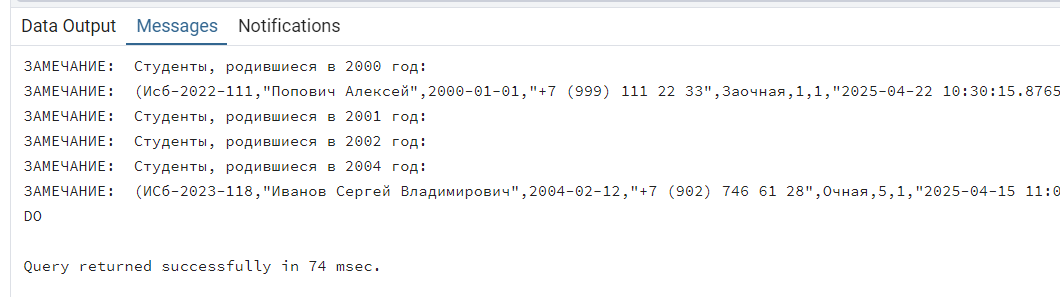
Задание 9.25. Перепишите результат выполнения задания 9.24 таким образом, чтобы все три переменные получали значения в результате выполнения одного оператора SELECT.



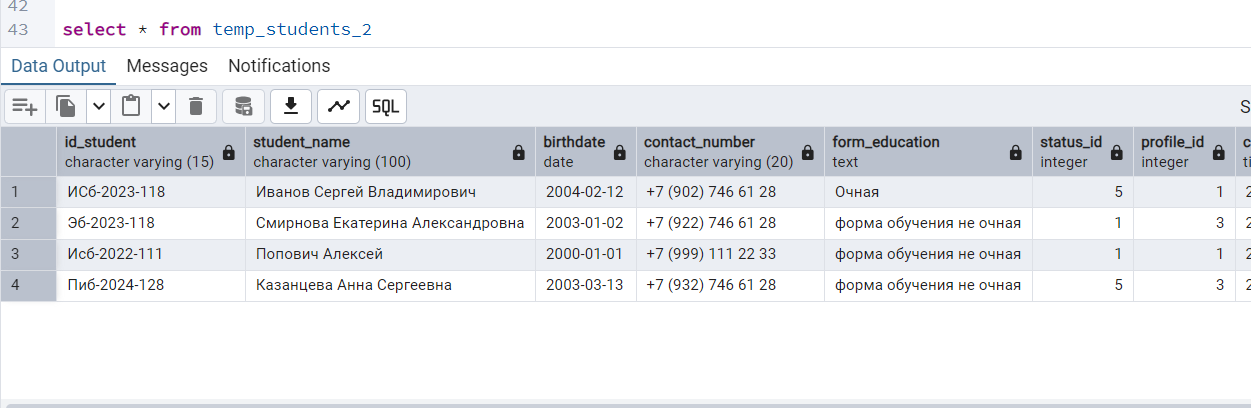
Задание 9.26. Основываясь на полученном при выполнении заданий 9.24 и 9.25 результате, напишите код, выводящий в цикле по годам от минимального к максимальному студентов, родившихся в соответствующий 91 год (на каждый год выполняется отдельный SELECT, возвращающий только студентов, родившихся в этом году)



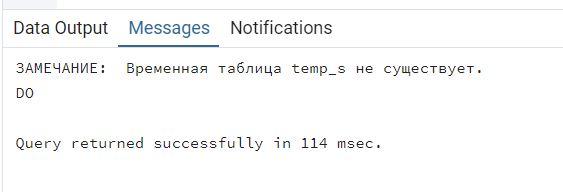
Задание 9.27. Измените полученный при выполнении прошлого задания фрагмент кода так, чтобы для какого-то заданного года из диапазона от минимального до максимального SELECT не выполнялся (используйте проверку условия и оператор CONTINUE; возможно вам понадобится внести какие-то еще изменения). Проверьте работу вашего кода. Замените CONTINUE на BREAK. Что поменяется в порядке выполнения?



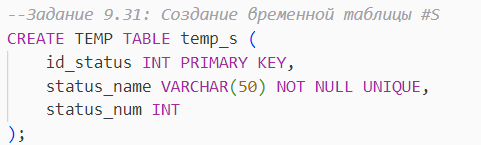
Задание 9.28. Создайте табличную переменную @STUDENTS, по структуре соответствующую таблице STUDENT. Заполните ее данными из таблицы STUDENT, меняя значение поля FormOfEducation так, что: – если форма обучения очная, значение не меняем; – если в поле FormOfEducation в таблице STUDENT стоит NULL, в @STUDENTS в соответствующее поле записывается «формы обучения нет»; – во всех других случаях в поле FormOfEducation в @STUDENTS записывается «форма обучения не очная»



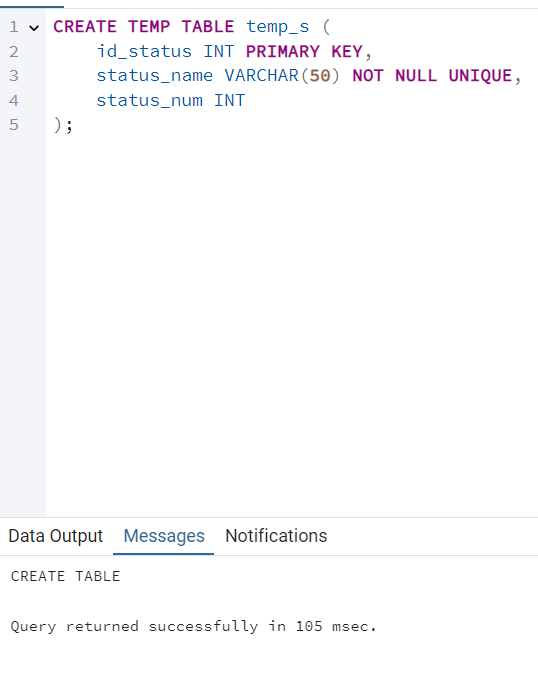
Задание 9.30. В Management Studio напишите код, проверяющий наличие временной таблицы #S и удаляющий ее, если она есть в базе данных



Задание 9.31. Создайте временную таблицу #S со столбцами, аналогичными столбцам STUDENT\_STATUS и дополнительным целочисленным столбцом StatusNum, допускающим значение NULL. Первичный ключ StatusID – целочисленный, без автоматической генерации.

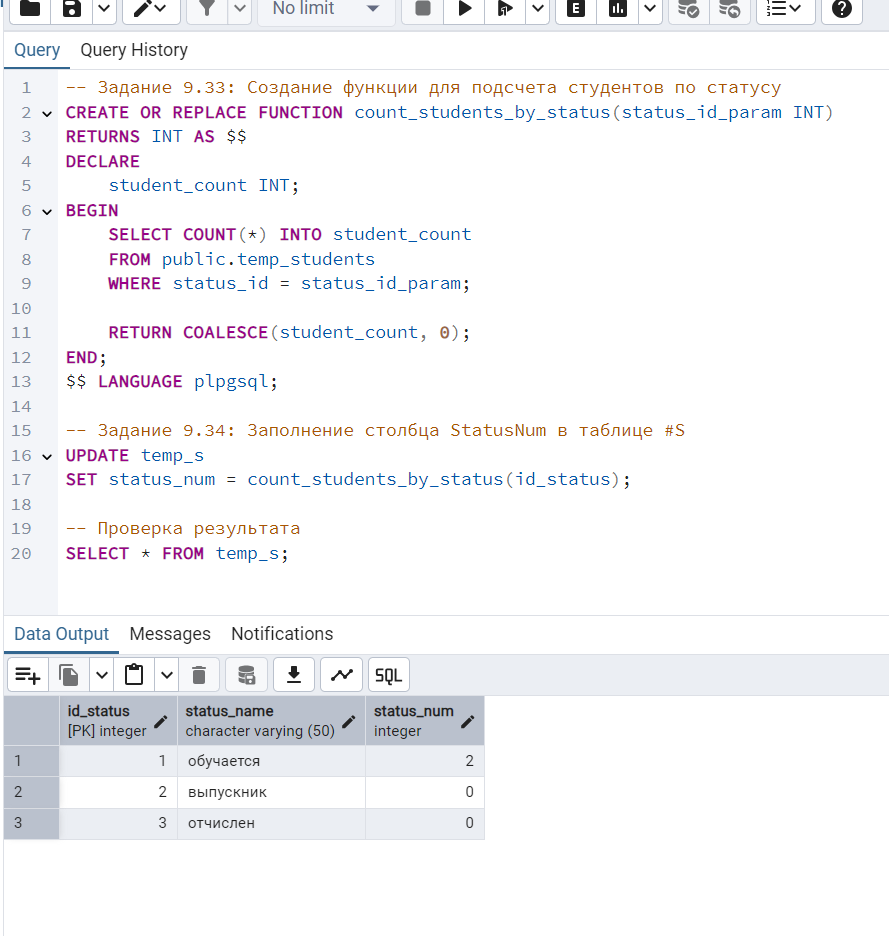


Задание 9.32. Заполните таблицу #S данными из таблицы STUDENT\_STATUS.



Задание 9.33. Напишите функцию, принимающую в качестве входного параметра значение идентификатора StatusID и возвращающую количество студентов этого статуса, упомянутых в таблице STUDENT. Обратите внимание, что, если соответствующих записей в Student не обнаружено, функция должна вернуть 0, а не NULL.

Задание 9.34. Используя написанную функцию, заполните значениями столбец StatusNum в таблице #S.



Задание 9.35. Создайте целочисленную переменную @MaxStatusNum и присвойте ей максимальное значение, встречающееся в столбце StatusNum таблицы #S. Выведите значение этой переменной с помощью команды PRINT.

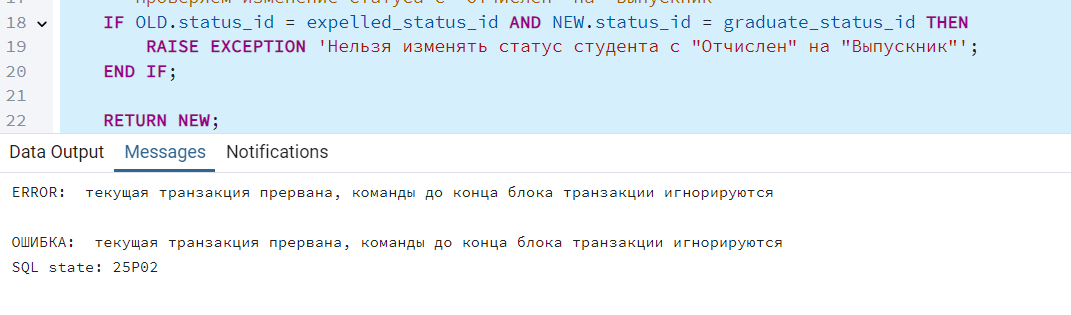
Задание 9.36. Создайте хранимую процедуру, которая получает на вход название профиля подготовки студента и: – если входной параметр (название профиля) имеет значение NULL, процедура заканчивает работу с кодом возврата -1; – если такой профиль есть в таблице PROFILE, процедура считает количество студентов с таким профилем в таблице STUDENT и возвращает его через выходной параметр; процедура завершается с кодом возврата 0; – если такого профиля нет в таблице PROFILE, процедура добавляет такой профиль в таблицу, возвращает 0 через выходной параметр и завершается с кодом возврата 1.



Задание 9.37. Проверьте работы процедуры при разных значениях входного параметра.



Задание 9.38. Для таблицы STUDENT создайте триггер на изменение данных, не позволяющий изменять статус студента «отчислен» на «выпускник». Протестируйте работу триггера на разных комбинациях вводимых данных. Проверьте правильность работы триггера в случае, когда одним оператором INSERT вносится изменения в несколько записей (не только с «отчислен» на «выпускник»). Триггер должен генерировать ошибку и откатывать транзакцию.



Задание 9.39. Создайте DDL-триггер на базу данных, не позволяющий удалять и изменять существующие таблицы и представления базы данных. Проверьте его работу. Убедитесь, что триггер блокирует только указанные выше действия. Попытайтесь удалить временную таблицу #S. Объясните полученный результат.

