МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

# Национальный исследовательский ядерный университет «МИФИ»

Институт интеллектуальных кибернетических систем

**Кафедра кибернетики (№ 22)**

**Отчёт о работе по курсу**

**«Базы данных (теоретические основы баз данных)»**

Вариант «Stepik»

|  |  |
| --- | --- |
| Выполнил | Ковальский И.В. |
| Группа | Б22-514 |
| Вариант | Stepik |
| Преподаватель | Петровская А.В. |
| Проверяющий |  |
| Оценка |  |

**Москва 2024**

**Содержание**

1. [Формулировка задания 3](#_TOC_250012)
2. [Концептуальная модель базы данных 3](#_TOC_250011)
   1. [Конкретизация предметной области](#_TOC_250010) 4
   2. [Описание предметной области](#_TOC_250009) 4
   3. [Описание атрибутов](#_TOC_250008) 4
3. [Логическое проектирование](#_TOC_250007) 6
4. [Физическое проектирование](#_TOC_250006) 7
   1. [Создание таблиц](#_TOC_250005) 7
   2. [Заполнение базы данных 11](#_TOC_250004)
      1. [Подготовка данных 11](#_TOC_250003)
      2. [Программа заполнения базы данных 12](#_TOC_250002)
      3. [Результаты заполнения 1](#_TOC_250001)8
5. [Выполнение запросов 2](#_TOC_250000)3

# Формулировка задания

Спроектировать базу данных для образовательной онлайн-платформы, предоставляющей доступ к различным учебный курсам. База данных должна содержать информацию об авторах, студентах, курсах, дисциплинах, пройденными ими, заданиях.

# Концептуальная модель базы данных

После проведения анализа предметной области была спроектирована следующая концептуальная модель:

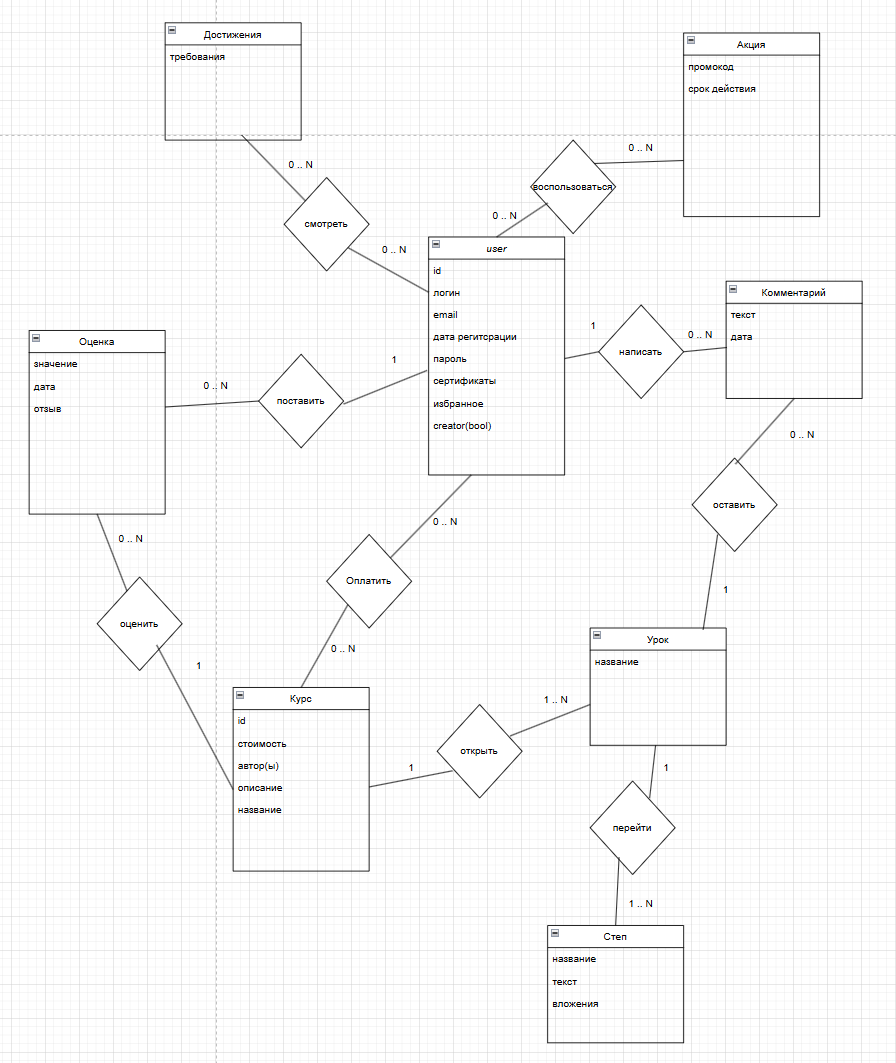


Рисунок 1 – Концептуальная модель базы данных

# Конкретизация предметной области

Необходимо создать систему, хранящую информацию о прохождении, завершении, покупке курсов пользователями. У каждого курса есть уроки, у каждого урока есть степы. Под каждым курсом можно оставлять комментарии. База данных должна отражать полное количество курсов на аккаунте у каждого пользователя, а также его результаты.

# Описание предметной области

Каждому конкретному студенту и автору в его личном кабинете доступна следующая информация: список купленных курсов, достижений, сертификатов, а также результаты успеваемости по каждому из них.

# Описание атрибутов

В процессе анализа были выделены следующие атрибуты, название и описание которых приведены в таблице ниже**:**

|  |  |
| --- | --- |
| **Имя атрибута** | **Расшифровка** |
| id | Уникальный идентификатор. Есть у каждого объекта. |

|  |  |
| --- | --- |
| логин | Учетные данные пользователя, которые используются для аутентификации. |
| email | Учетные данные пользователя, которые используются для аутентификации. |
| Дата регистрации | Дата регистрации на платформе. |
| Название курса | Название конкретного курса |
| Сертификат | Список сертификатов по успешно завершенным курса у каждого пользователя. |
| Избранное | Список курсов, добавленных пользователем в избранное. |
| Creator | Пометка, является ли пользователь создателем курсов. |
| Стоимость | Стоимость покупки курса. |
| Промокод | Код, для получения скидки при покупке курсов. |
| Срок действия | Срок действия промокода. |

# Логическое проектирование

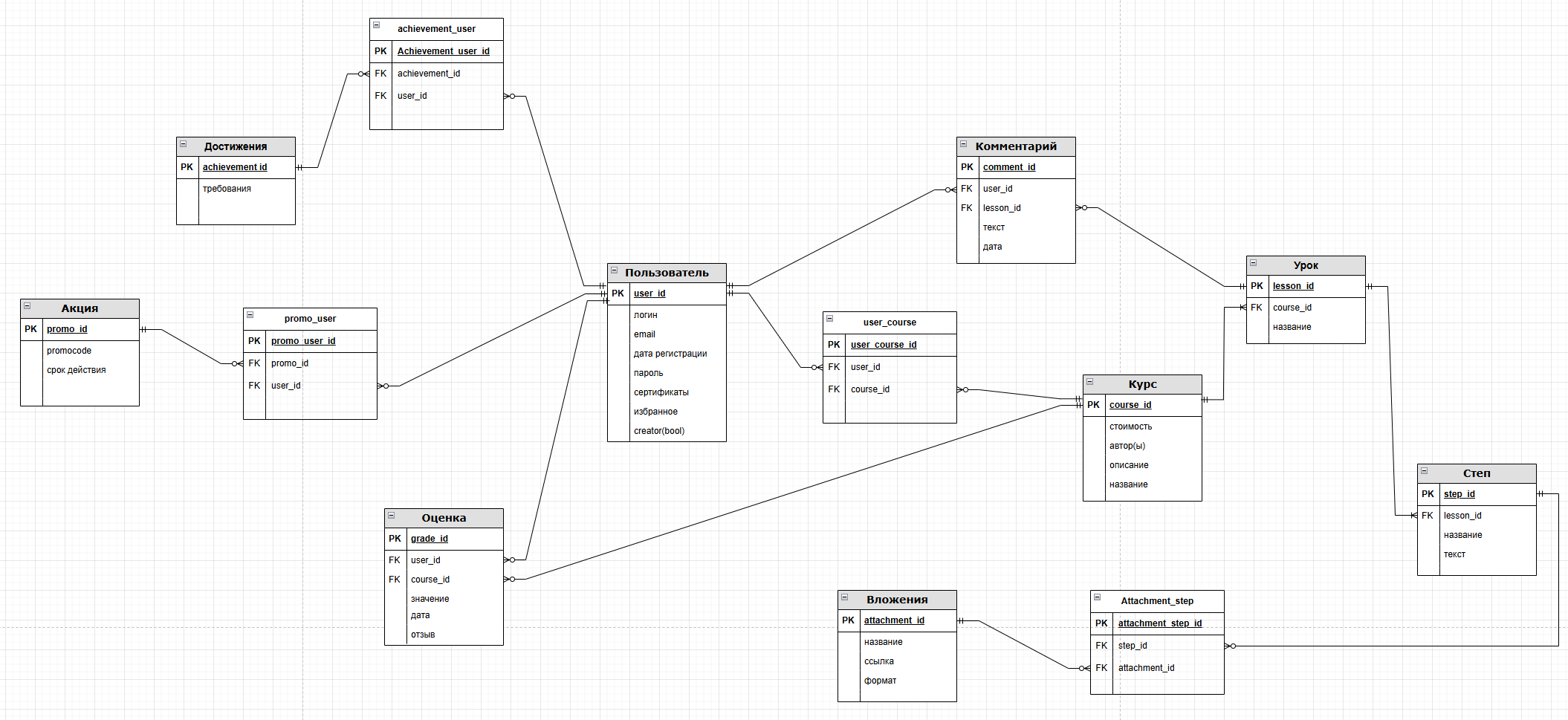
Следующим шагом на основе КМПО была разработана логическая модель базы данных, представленная ниже:  


Рисунок 2 – Логическая модель базы данных

Все таблицы находятся в 3 нормальной форме(3нф). Для связей “многие ко многим” созданы промежуточные таблицы.

# Физическое проектирование

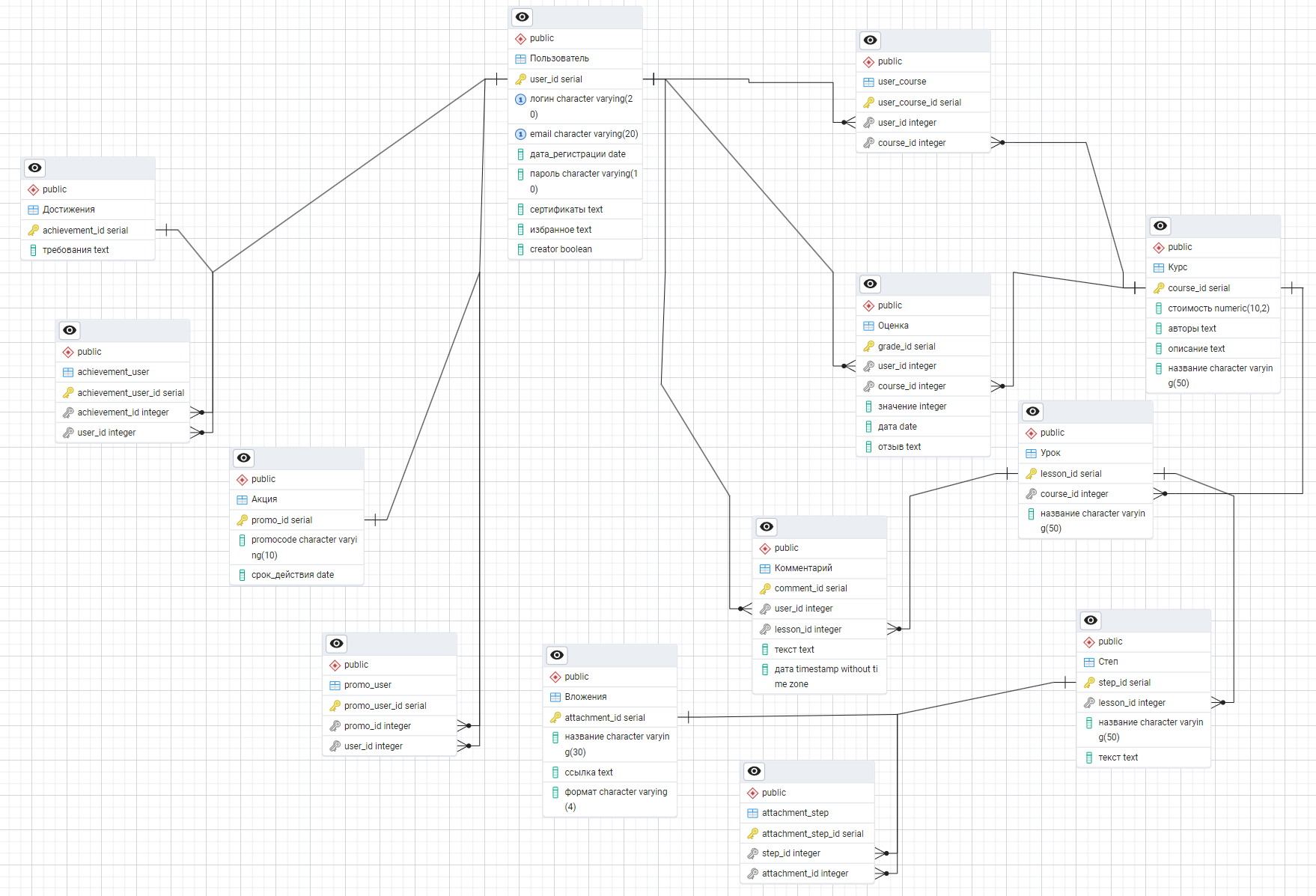
В качестве СУБД для реализации разработанной базы данных была выбрана PostgreSQL. В связи с проведённым анализом предметной области была проработана следующая физическая схема БД. Она представлена на следующем рисунке:  


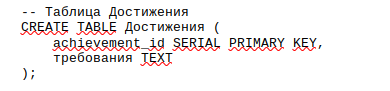
Рисунок 3 – Графическое представление базы данных

# Создание таблиц

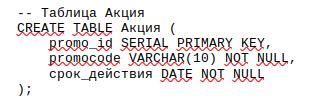
Ниже приведены запросы на языке SQL (диалект PostgreSQL) для создания таблиц, описанных выше.

Код для заполнения таблицы Пользователь:

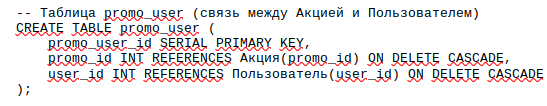
Код для заполнения таблицы Достижения



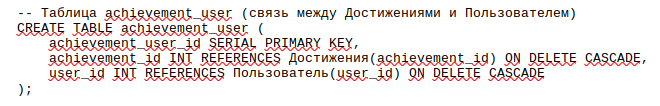
Код для заполнения таблицы Акция



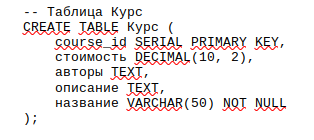
Код для заполнения таблицы promo\_user



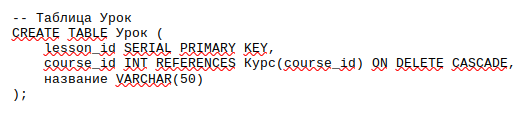
Код для заполнения таблицы achievement\_user



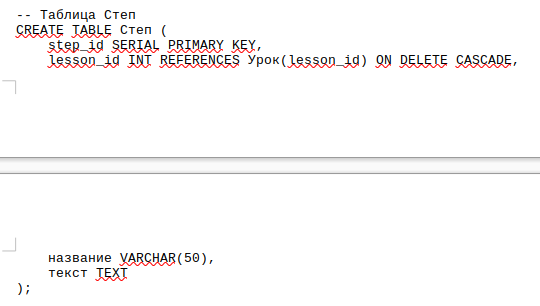
Код для заполнения таблицы Курс



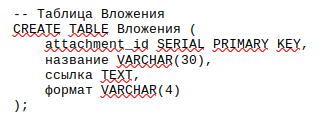
Код для заполнения таблицы Урок



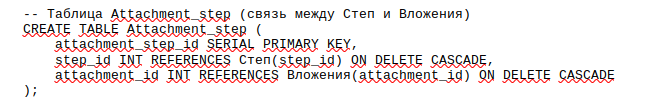
Код для заполнения таблицы Степ

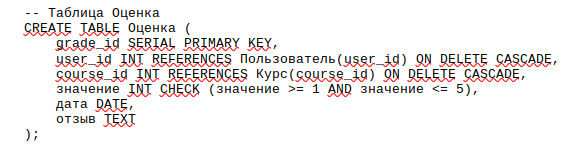


Код для заполнения таблицы Вложения

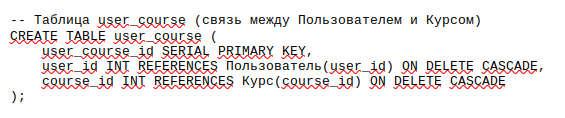


Код для заполнения таблицы Attachment\_step

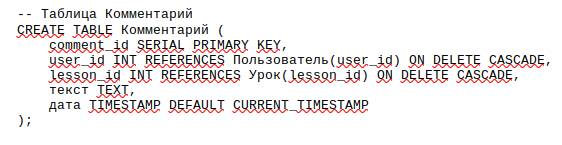


Код для заполнения таблицы Оценка

Код для заполнения таблицы user\_course



Код для заполнения таблицы Комментарий



# Заполнение базы данных

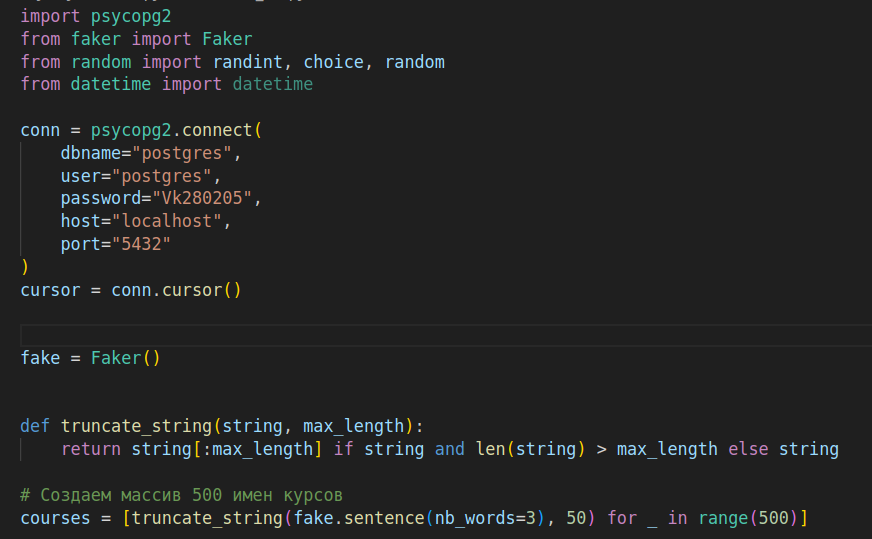
Заполнение базы данных проводилось при помощи библиотеки для работы с PostgreSQL psycopg2 языка программирования Python.

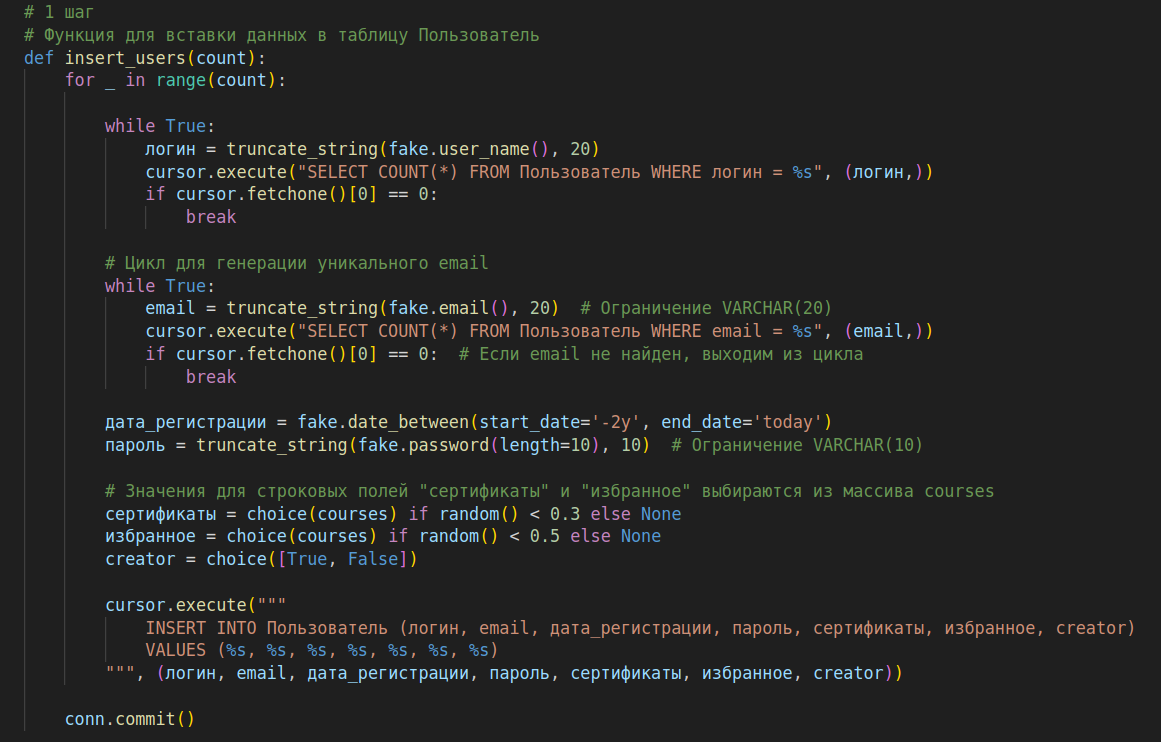
# Подготовка данных

Были подготовлены массивы имен курсов, сертификатов, достижений.

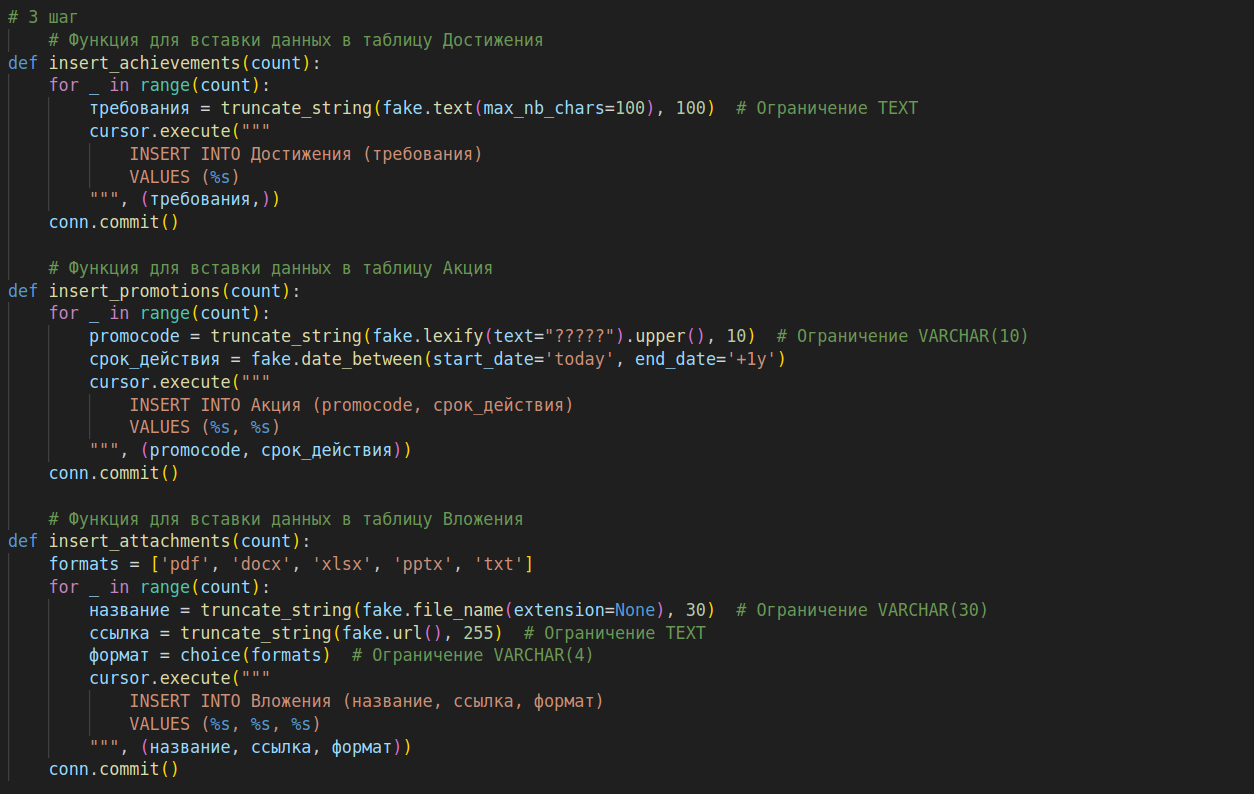
# Программа заполнения базы данных

* + - * Код, выполняющий вставку данных в таблицу “Пользователь”:

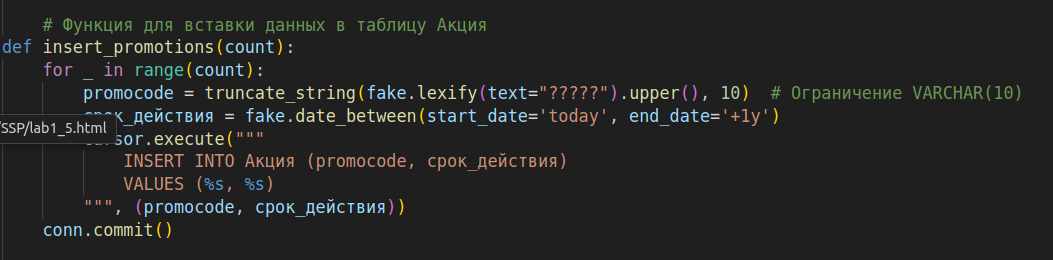




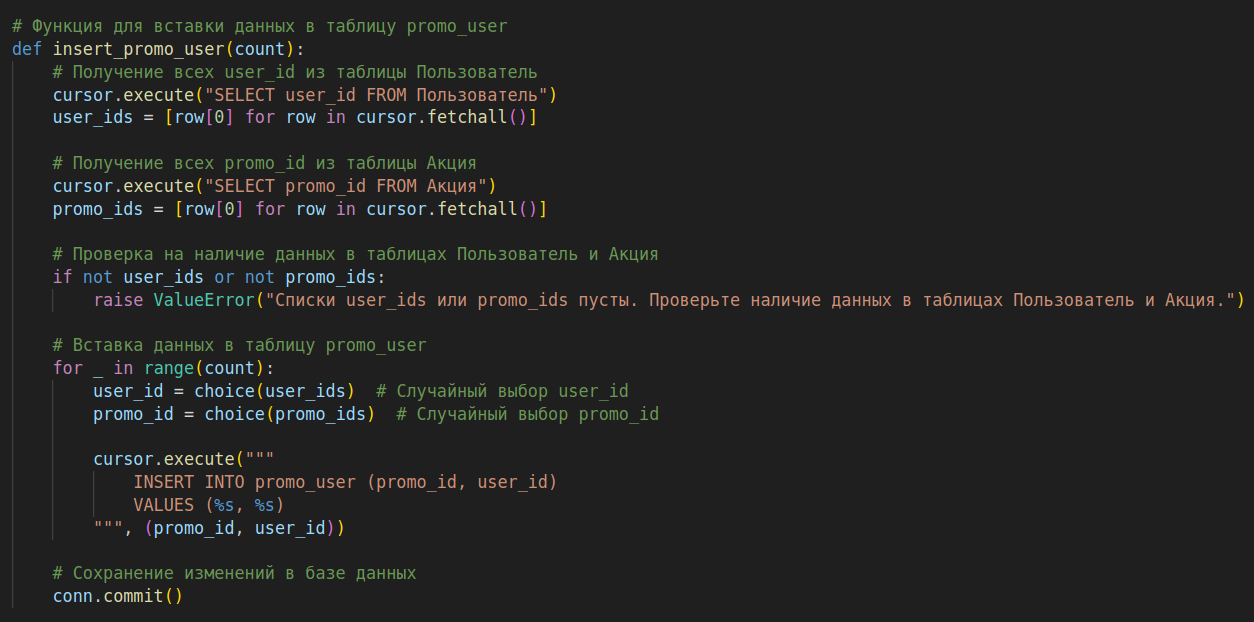
* + - * Код, выполняющий вставку данных в таблицу “Достижения”:



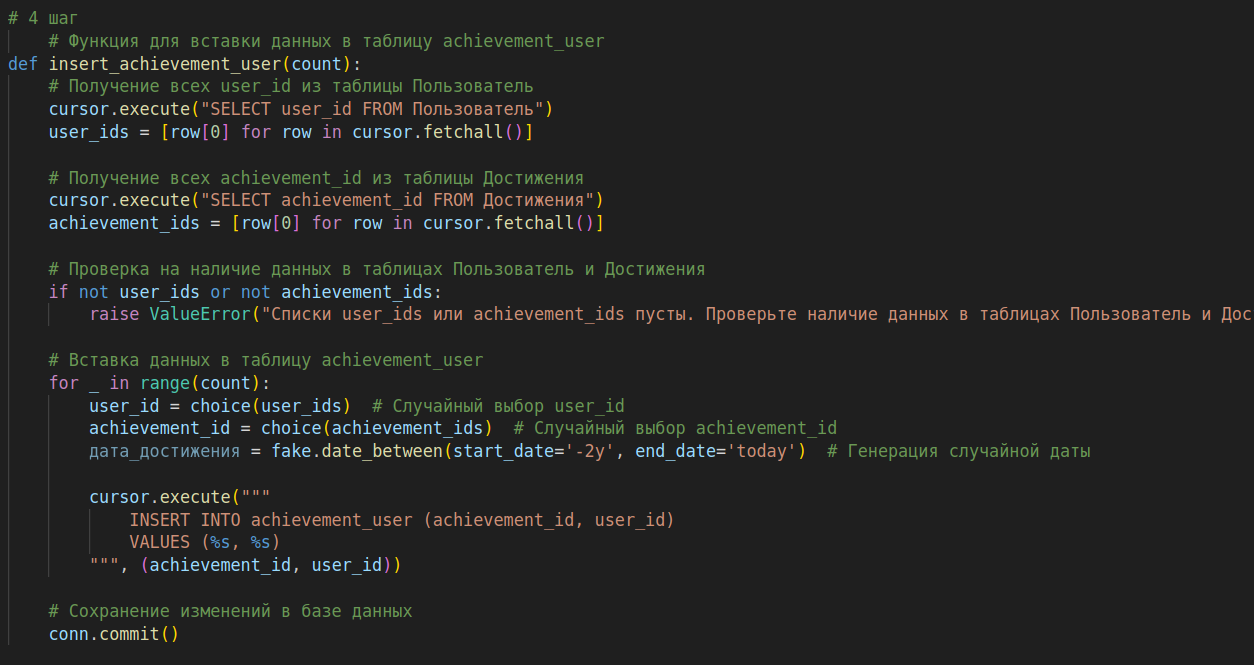
* + - * Код, выполняющий вставку данных в таблицу “Акция”:



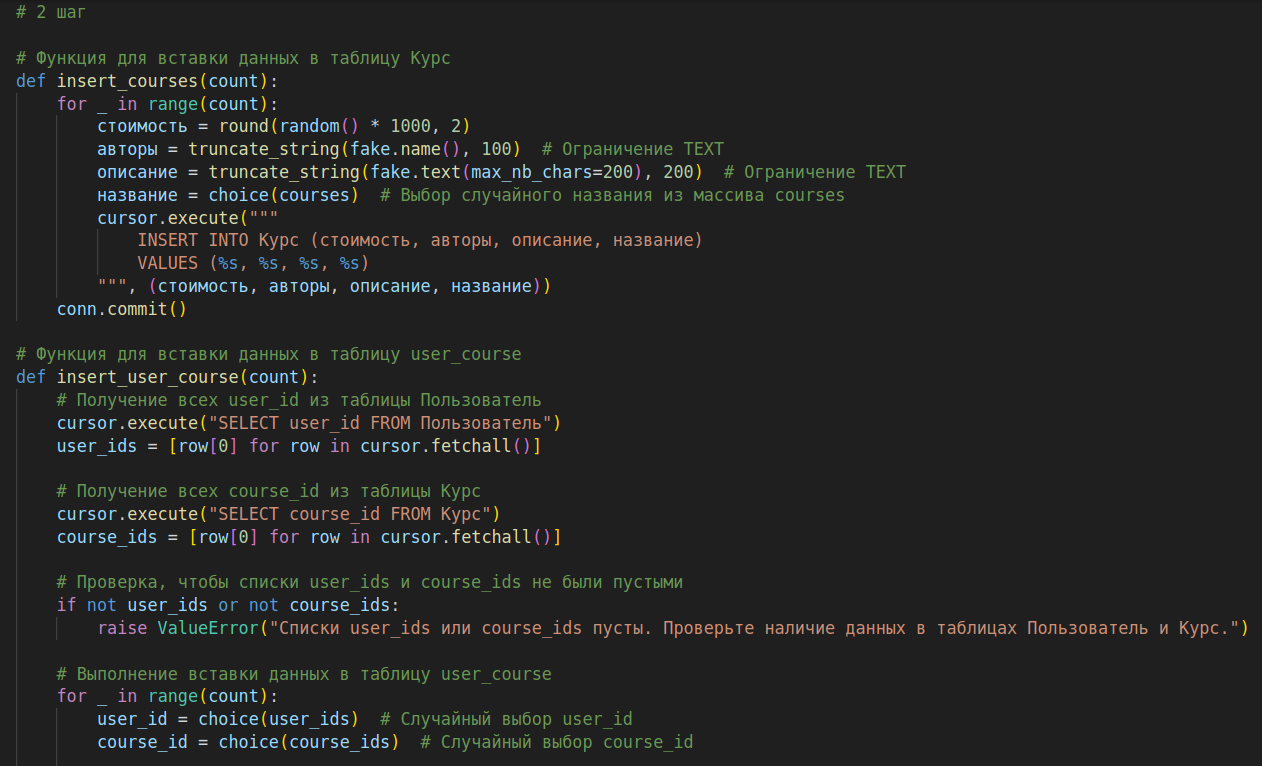
* + - * Код, выполняющий вставку данных в таблицу “promo\_user”:

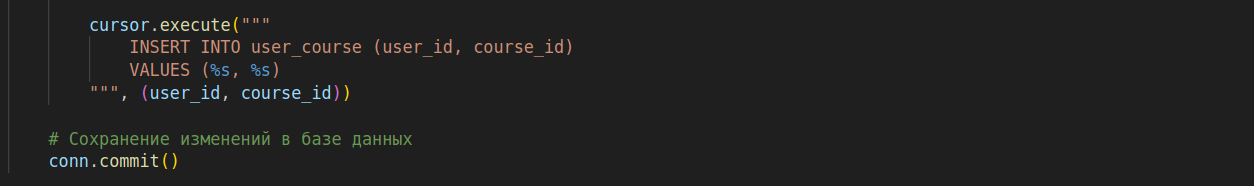


* + - * Код, выполняющий вставку данных в таблицу “achievement\_user”:

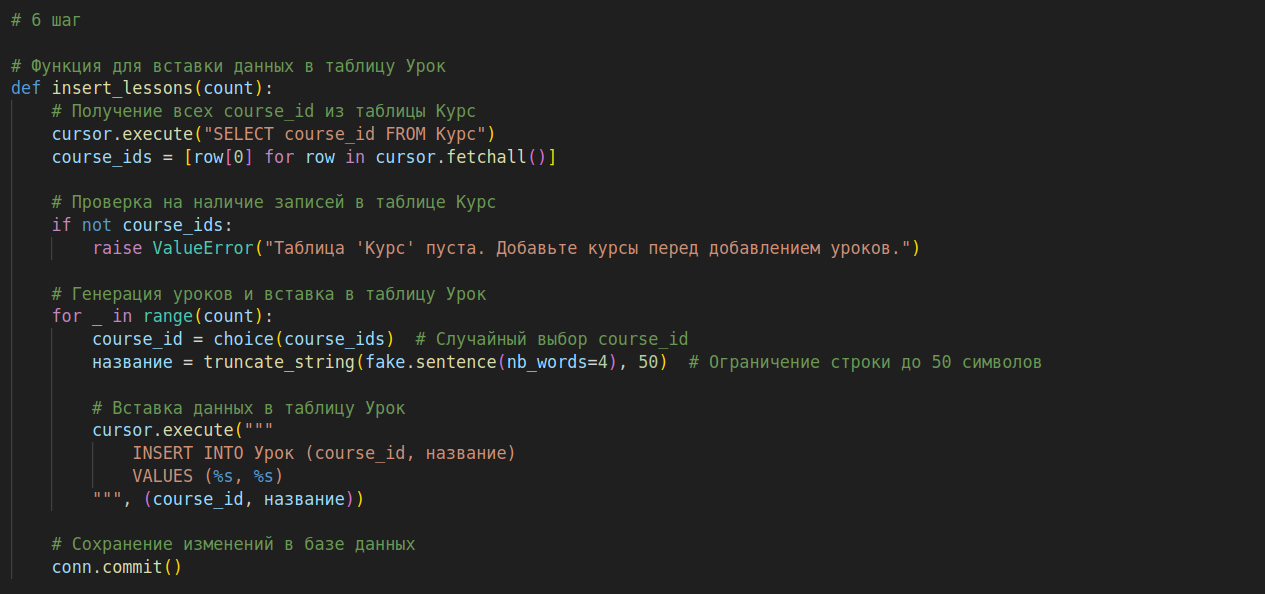


* + - * Код, выполняющий вставку данных в таблицу “Курс”:

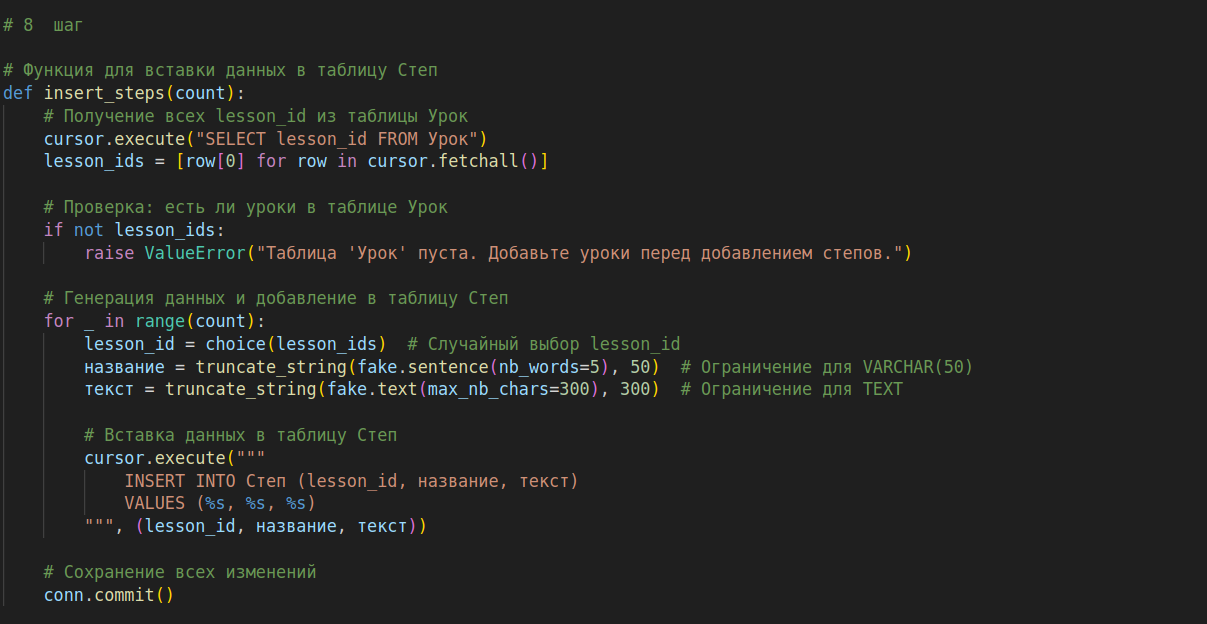




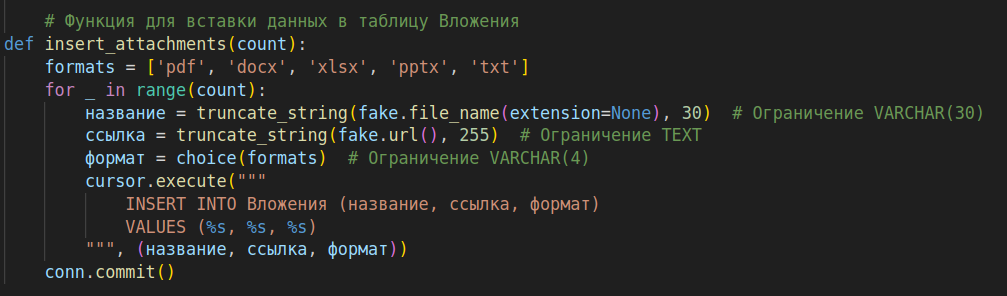
* + - * Код, выполняющий вставку данных в таблицу “Урок”:



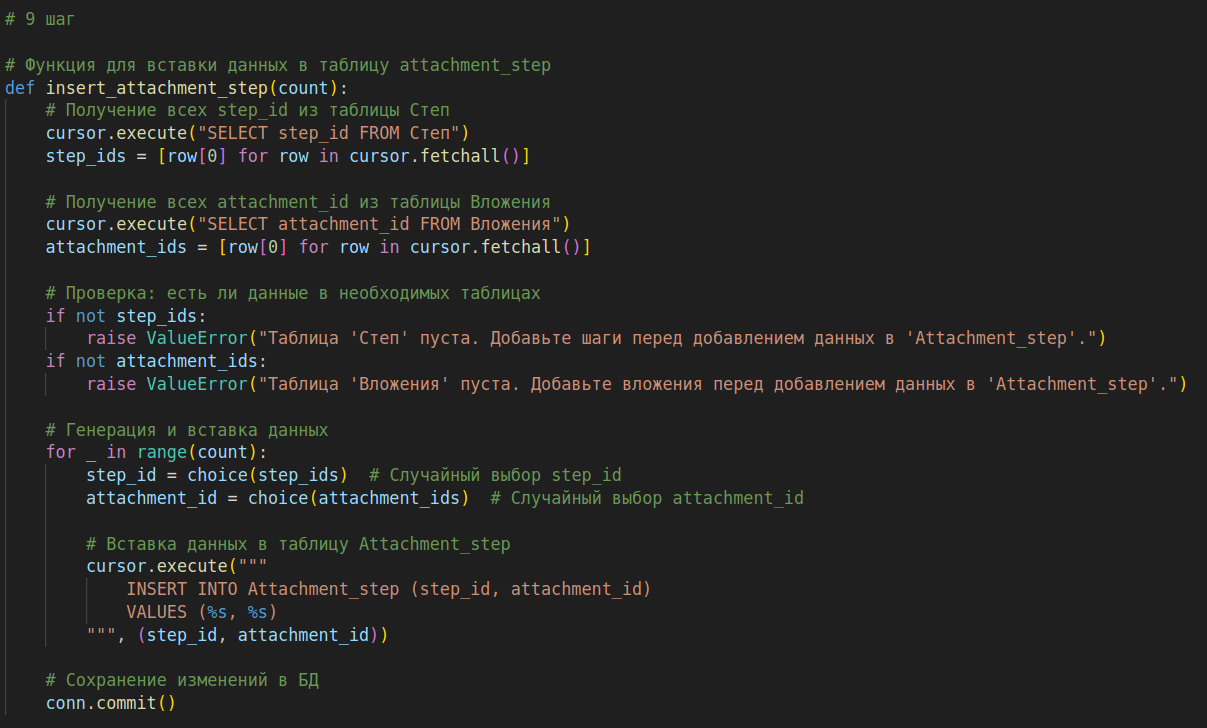
* + - * Код, выполняющий вставку данных в таблицу “Степ”:



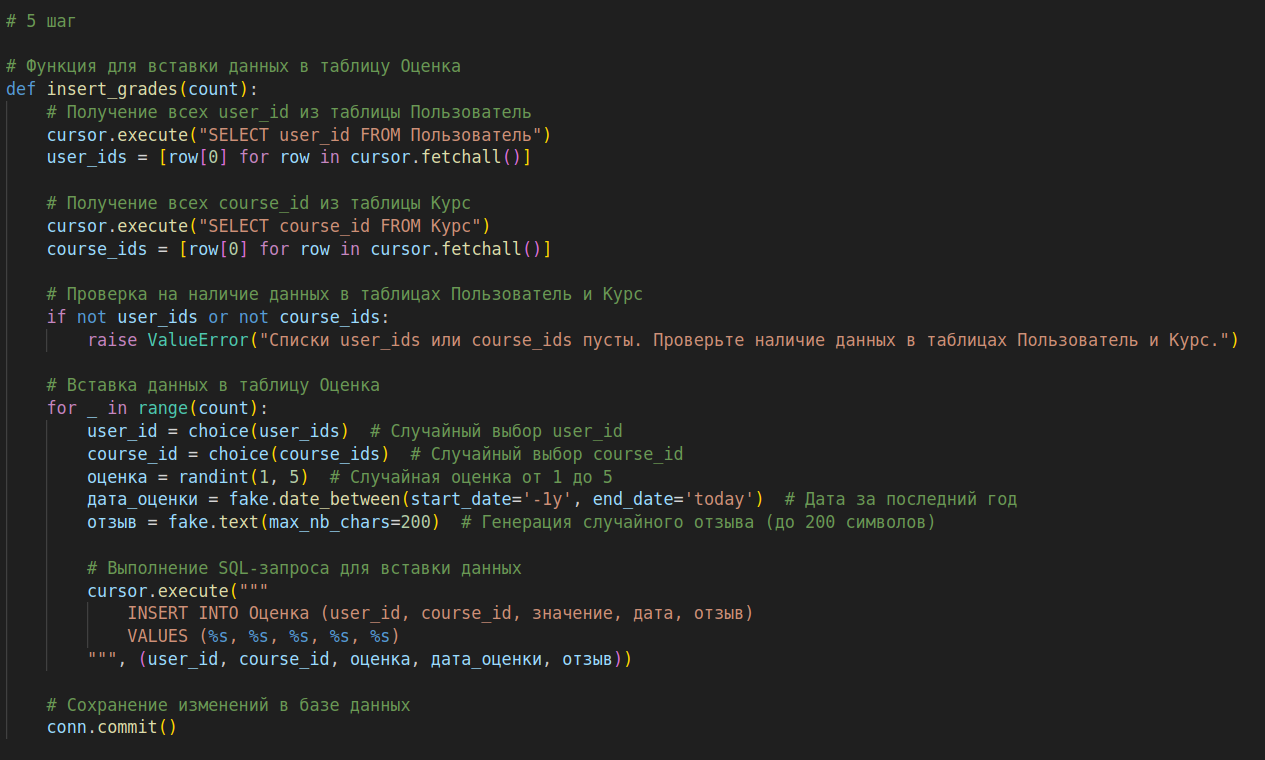
* + - * Код, выполняющий вставку данных в таблицу “Вложения”:



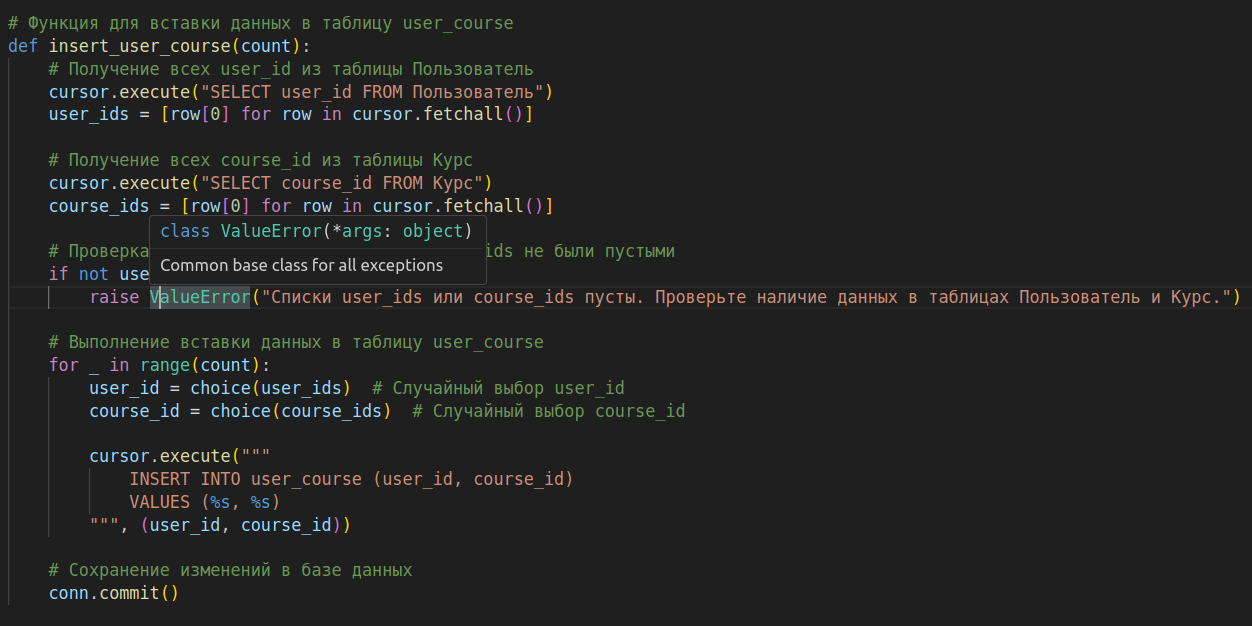
* + - * Код, выполняющий вставку данных в таблицу “Attachment\_step”:



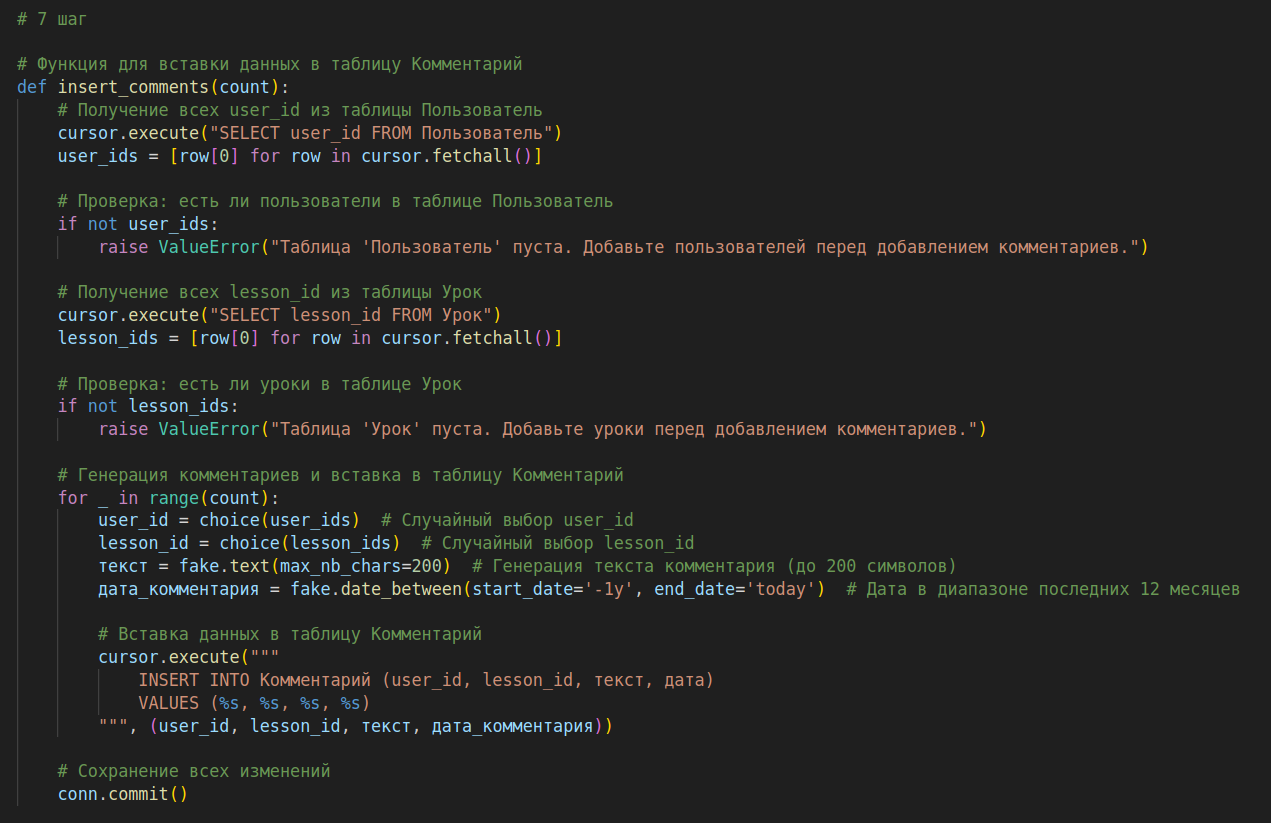
* + - * Код, выполняющий вставку данных в таблицу “Оценка”:



* + - * Код, выполняющий вставку данных в таблицу “user\_course”:



* + - * Код, выполняющий вставку данных в таблицу “Комментарий”:

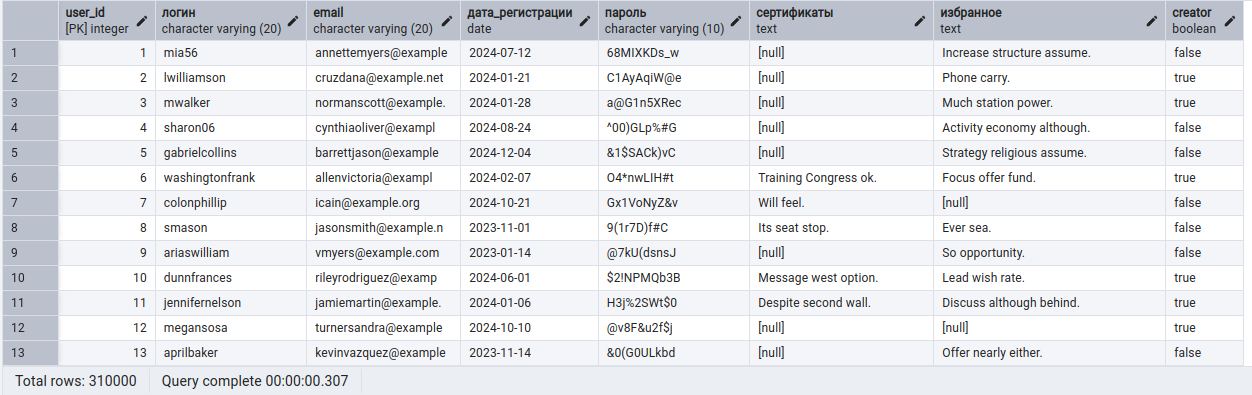


Как видно, выполняется обыкновенный INSERT-запрос, который содержит данные, выбранные случайным образом.

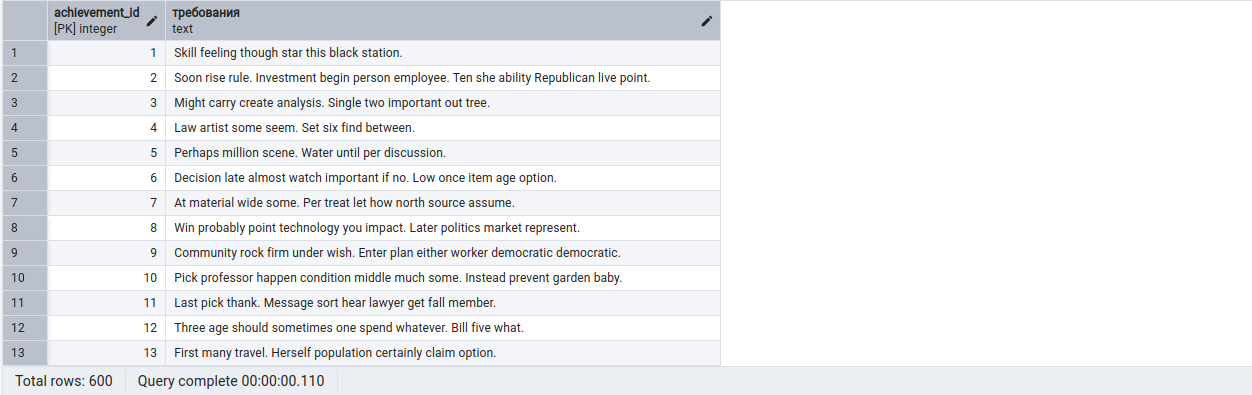
# Результаты заполнения

Далее представлены результаты работы программы на примере таблиц, соответствующих функциям, приведенным выше.

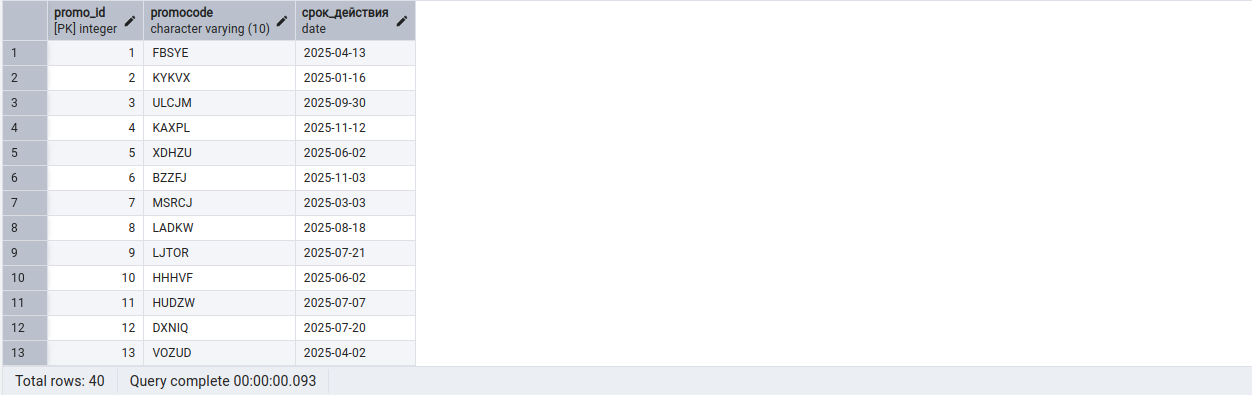
* + - * Таблица “Пользователь”:



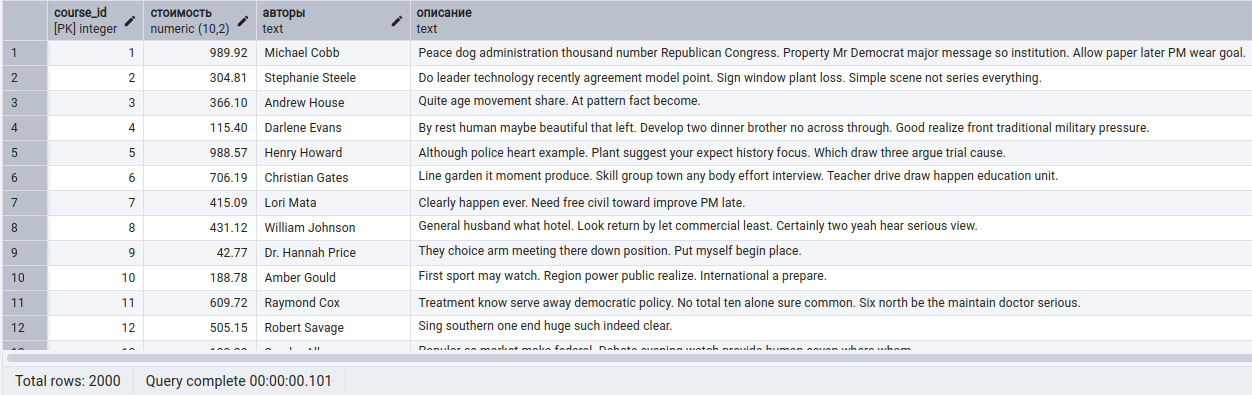
* + - * Таблица “Достижения”:



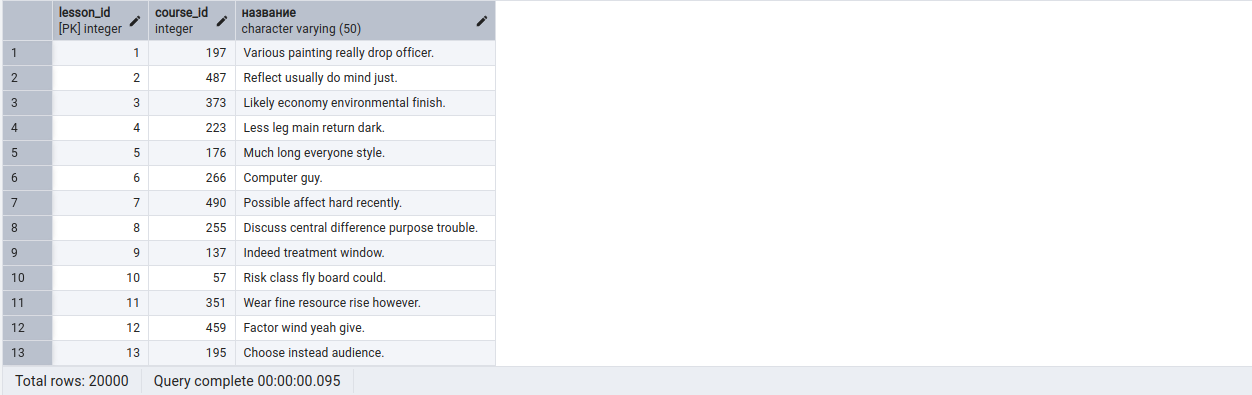
* + - * Таблица “Акция”:



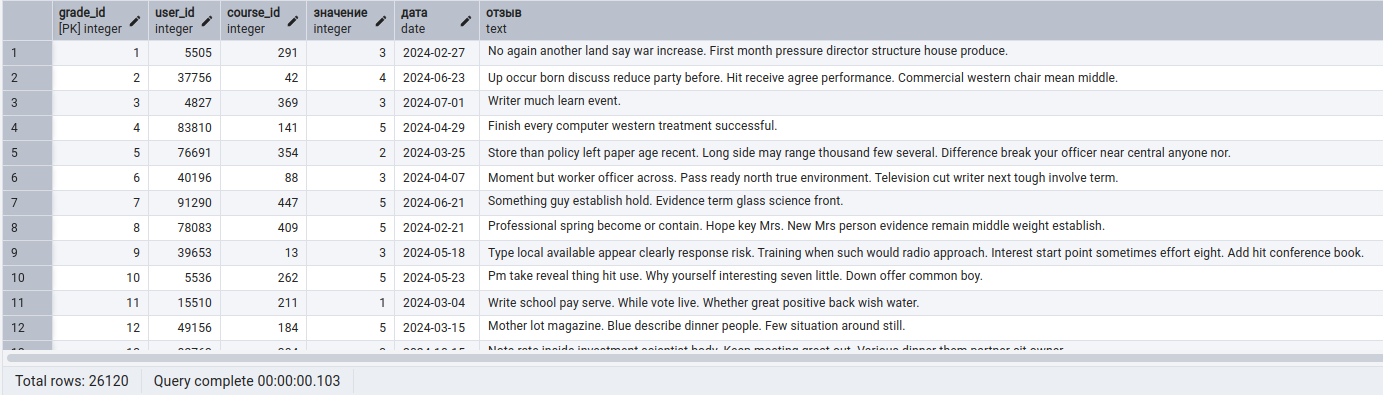
* + - * Таблица “Курс”:



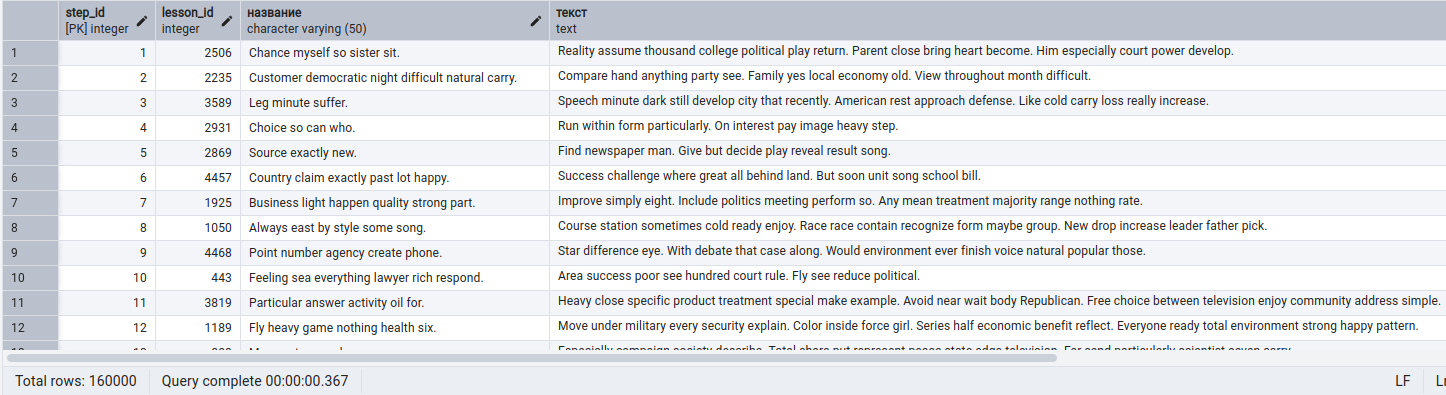
* + - * Таблица “Урок”:



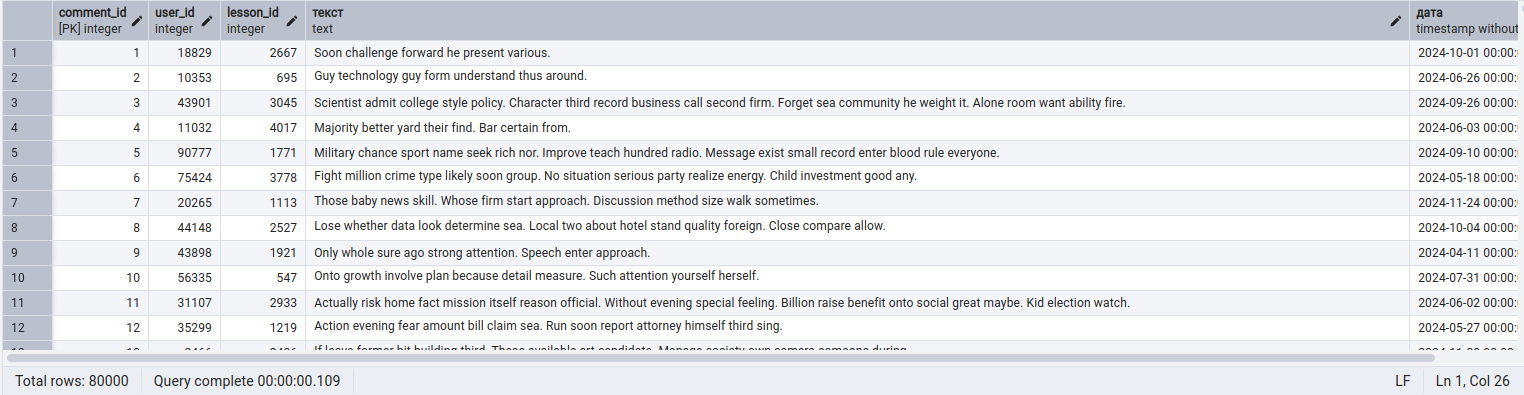
* + - * Таблица “Оценка”:



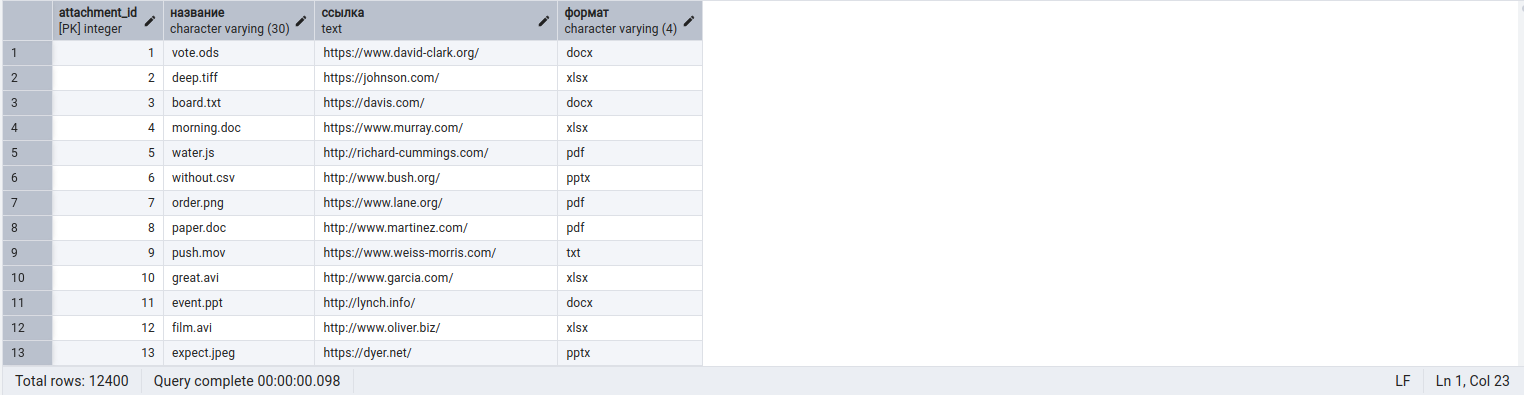
* + - * Таблица “Степ”:



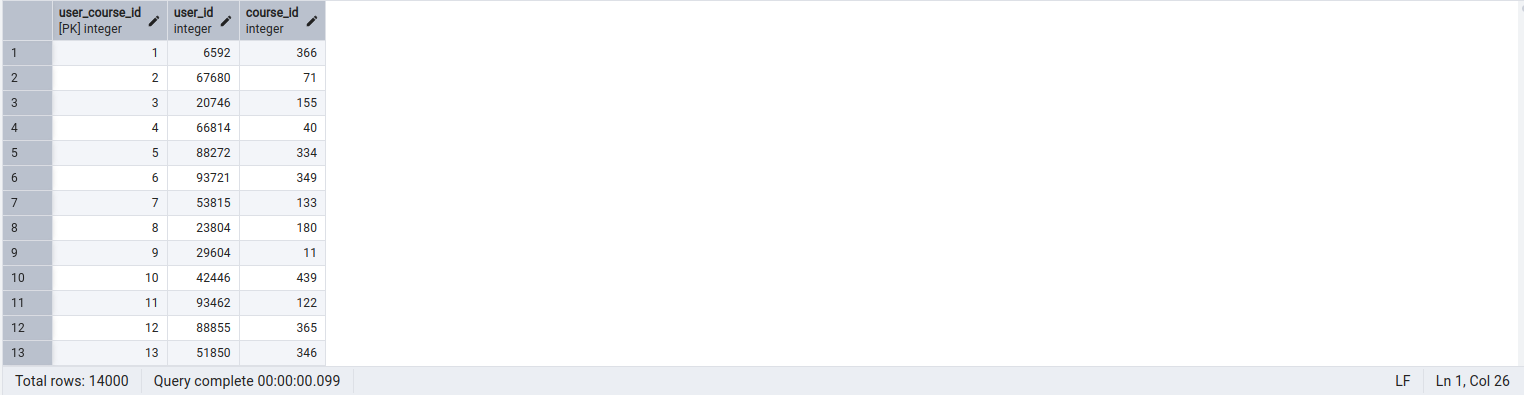
* + - * Таблица “Комментарий”:



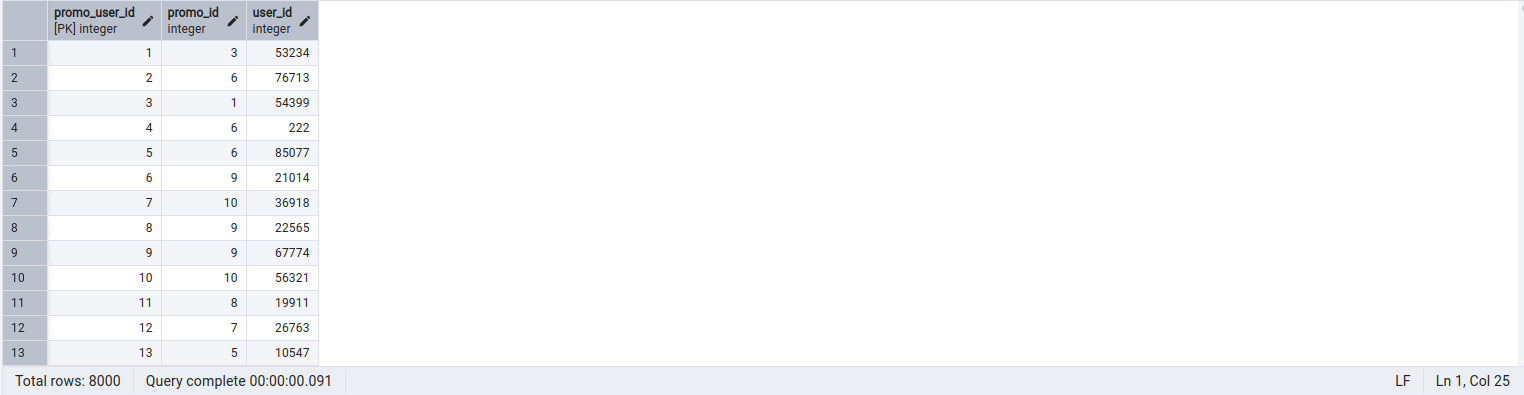
* + - * Таблица “Вложения”:



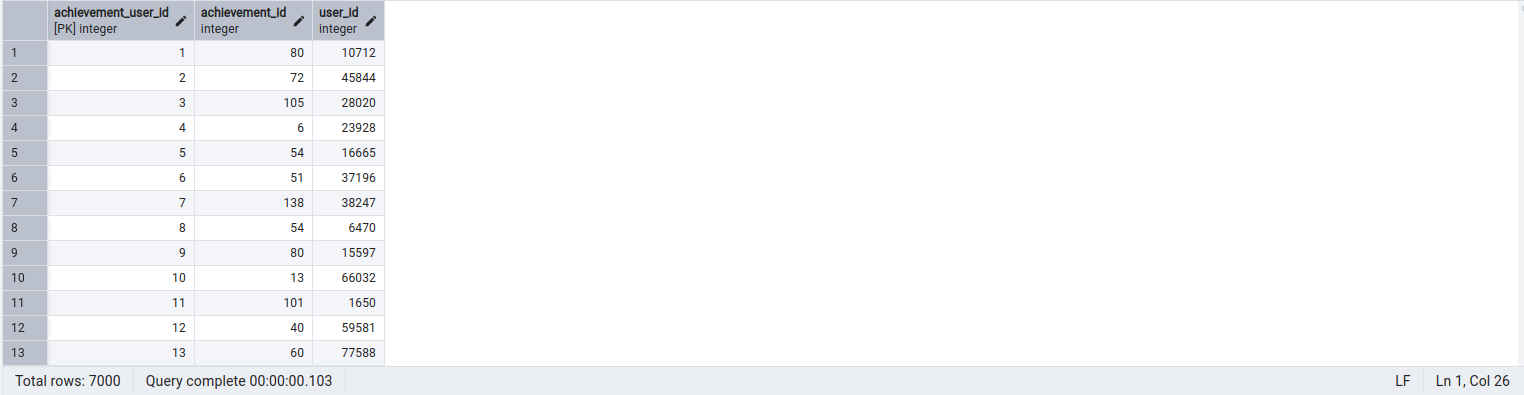
* + - * Таблица “user\_course”:



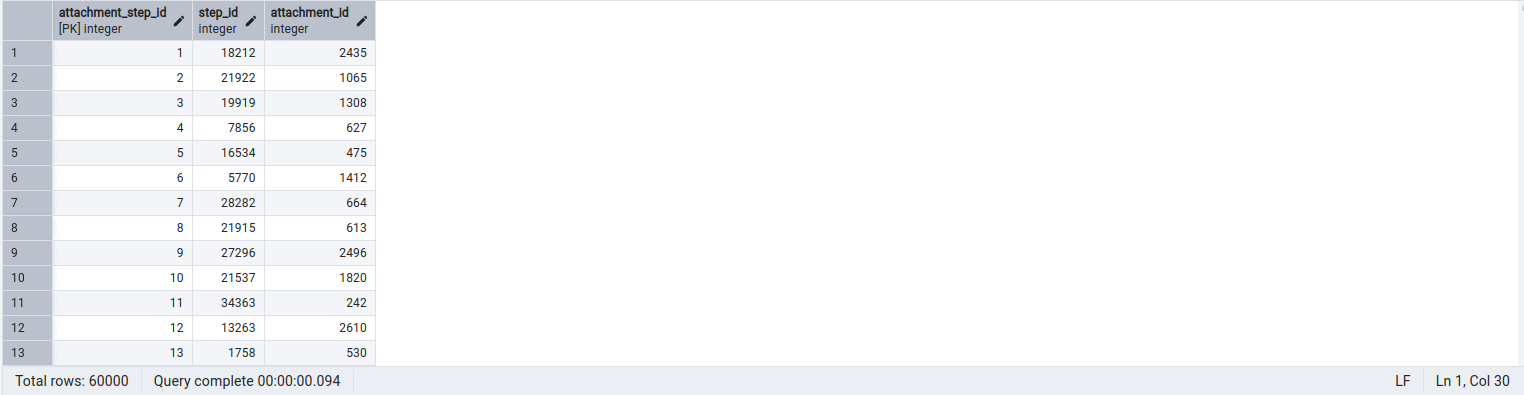
* + - * Таблица “promo\_user”:



* + - * Таблица “achievement\_user”:



* + - * Таблица “attachment\_step”:



# Выполнение запросов

В этом разделе приведены различные запросы к реализованной базе данных — их краткие описания, непосредственно запрос на языке SQL и результат выполнения.

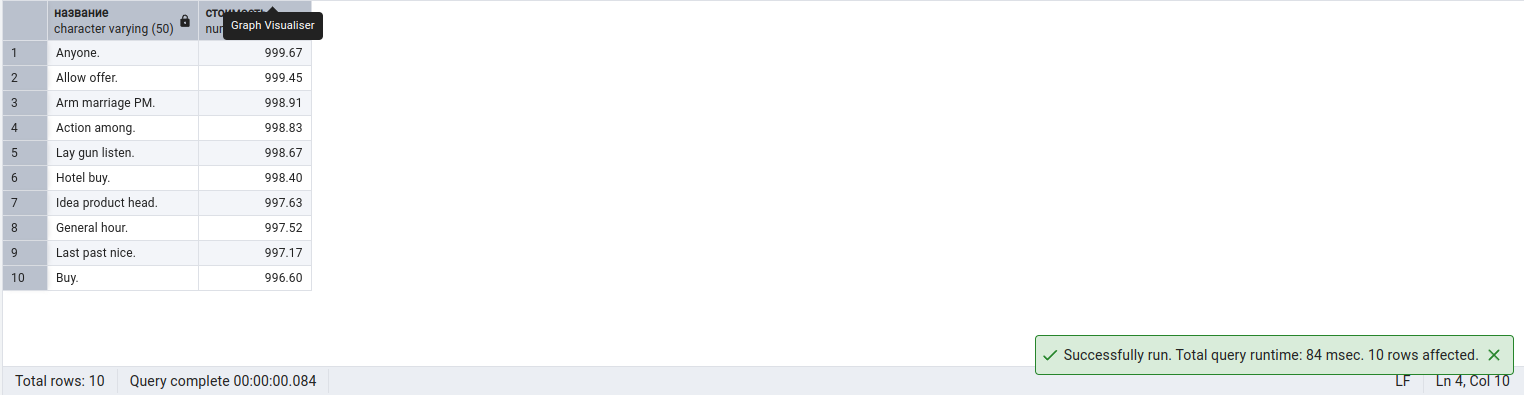
1. Топ-10 самых дорогих курсов

SELECT название, стоимость

FROM Курс

ORDER BY стоимость DESC

LIMIT 10;

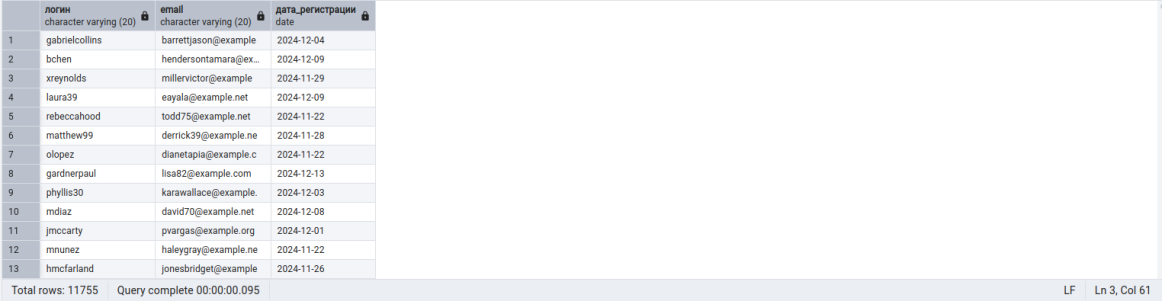


1. Пользователи, зарегистрировавшиеся в последние 30 дней

SELECT логин, email, дата\_регистрации

FROM Пользователь

WHERE дата\_регистрации >= CURRENT\_DATE - INTERVAL '30 days';

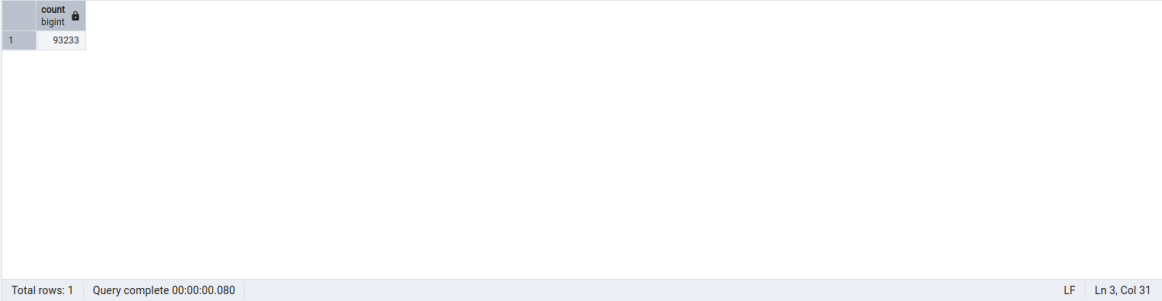


1. Количество пользователей с сертификатами

SELECT COUNT(\*)

FROM Пользователь

WHERE сертификаты IS NOT NULL;



1. Курсы и количество пользователей, записавшихся на них

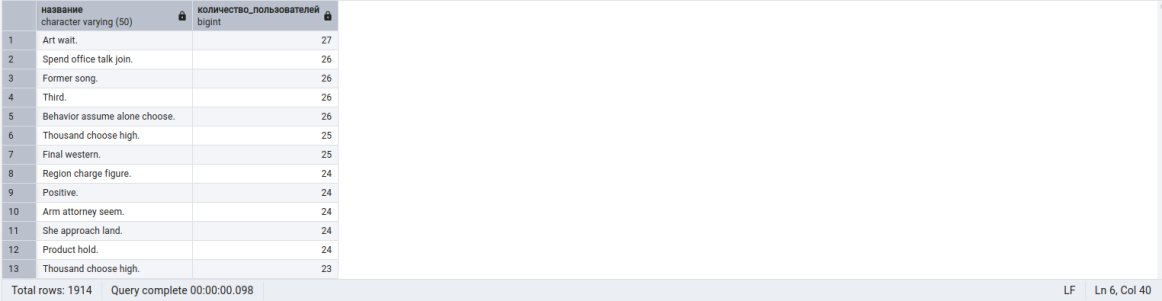
SELECT Курс.название, COUNT(user\_course.user\_id) AS количество\_пользователей

FROM user\_course

JOIN Курс ON user\_course.course\_id = Курс.course\_id

GROUP BY Курс.course\_id

ORDER BY количество\_пользователей DESC;



1. Пользователи, оставившие комментарии более чем к 5 урокам

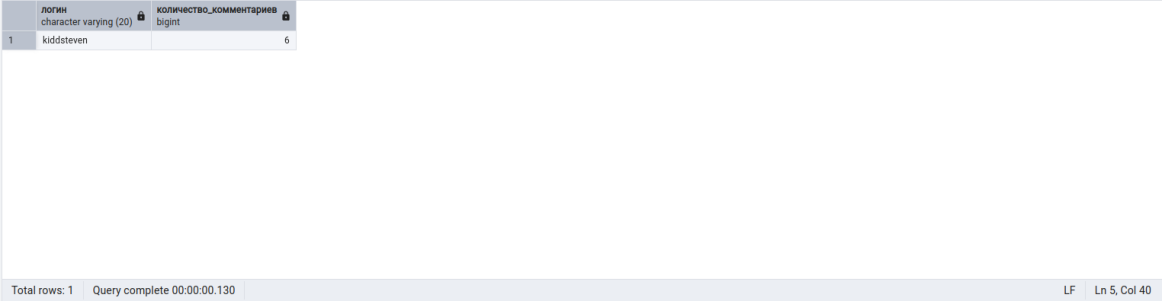
SELECT Пользователь.логин, COUNT(Комментарий.lesson\_id) AS количество\_комментариев

FROM Комментарий

JOIN Пользователь ON Комментарий.user\_id = Пользователь.user\_id

GROUP BY Пользователь.user\_id

HAVING COUNT(Комментарий.lesson\_id) > 5;



1. Средняя оценка курсов, сгруппированная по стоимости  
   SELECT

CASE

WHEN стоимость < 100 THEN 'Дешевые курсы'

WHEN стоимость BETWEEN 100 AND 500 THEN 'Средние курсы'

ELSE 'Дорогие курсы'

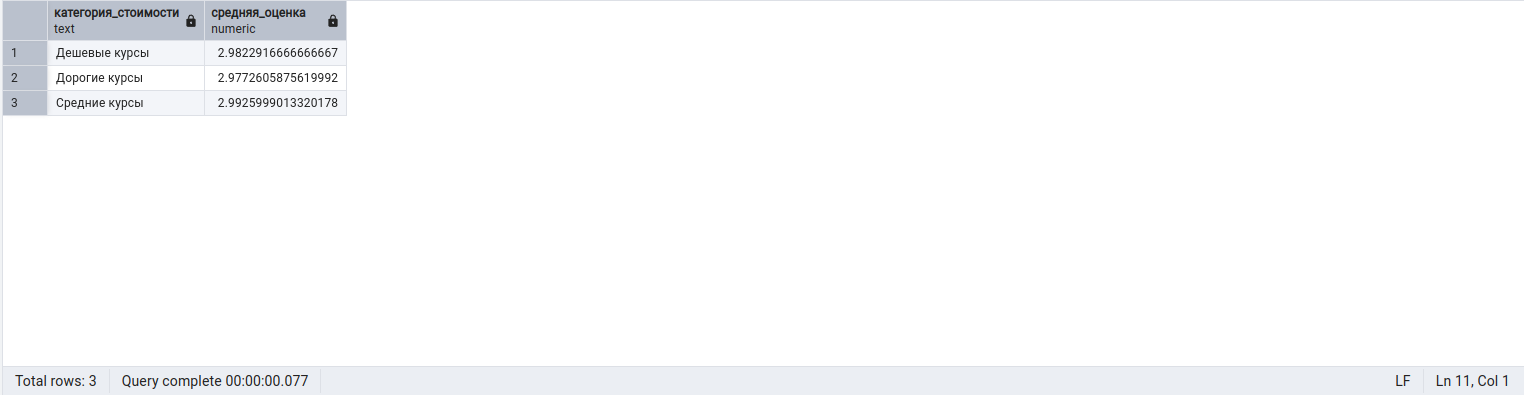
END AS категория\_стоимости,

AVG(Оценка.значение) AS средняя\_оценка

FROM Курс

JOIN Оценка ON Курс.course\_id = Оценка.course\_id

GROUP BY категория\_стоимости;



1. Топ-5 пользователей по количеству полученных достижений

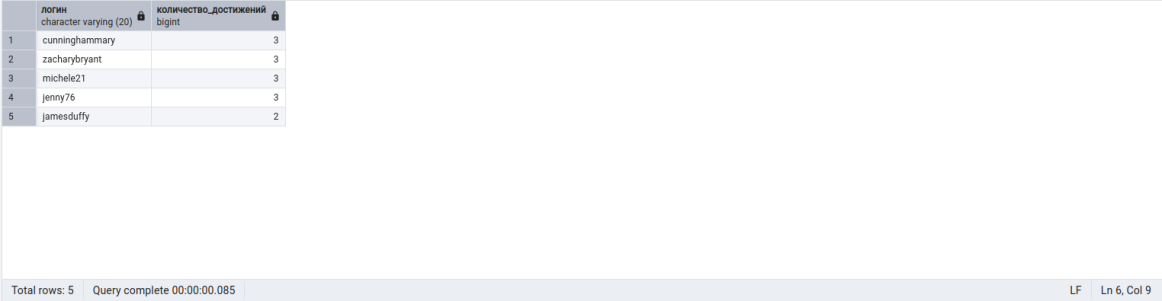
SELECT Пользователь.логин, COUNT(achievement\_user.achievement\_id) AS количество\_достижений

FROM achievement\_user

JOIN Пользователь ON achievement\_user.user\_id = Пользователь.user\_id

GROUP BY Пользователь.user\_id

ORDER BY количество\_достижений DESC

LIMIT 5;  
  


1. Среднее количество шагов на урок по курсам

SELECT Курс.название, AVG(количество\_шагов) AS среднее\_количество\_шагов

FROM (

SELECT Урок.course\_id, COUNT(Степ.step\_id) AS количество\_шагов

FROM Степ

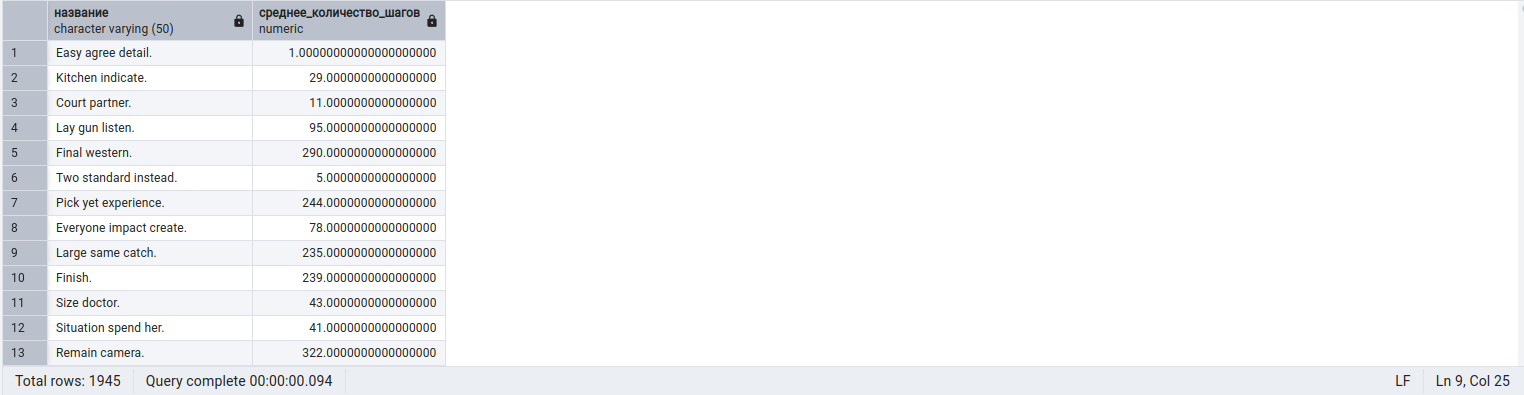
JOIN Урок ON Степ.lesson\_id = Урок.lesson\_id

GROUP BY Урок.course\_id

) AS шаги\_по\_урокам

JOIN Курс ON шаги\_по\_урокам.course\_id = Курс.course\_id

GROUP BY Курс.course\_id;



1. Retention rate (Коэффициент повторных покупок)  
     
   WITH первые\_покупки AS (

SELECT Пользователь.user\_id, MIN(Пользователь.дата\_регистрации) AS дата\_первой\_покупки

FROM user\_course

JOIN Пользователь ON user\_course.user\_id = Пользователь.user\_id

GROUP BY Пользователь.user\_id

),

повторные\_покупки AS (

SELECT user\_course.user\_id, COUNT(\*) AS количество\_покупок

FROM user\_course

WHERE user\_course.user\_id IN (SELECT первые\_покупки.user\_id FROM первые\_покупки)

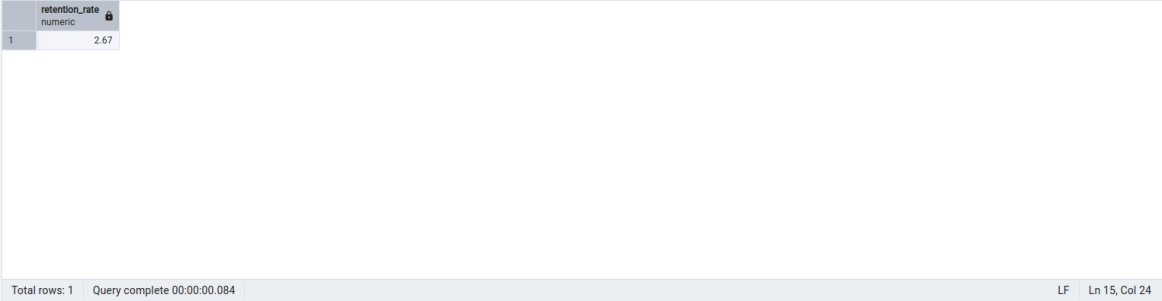
GROUP BY user\_course.user\_id

)

SELECT

ROUND(SUM(CASE WHEN повторные\_покупки.количество\_покупок > 1 THEN 1 ELSE 0 END)::DECIMAL / COUNT(\*) \* 100, 2) AS retention\_rate

FROM повторные\_покупки;



1. Анализ эффективности акций: сравнение продаж с использованием промокодов и без

SELECT

CASE WHEN promo\_user.promo\_id IS NULL THEN 'Без промокода' ELSE 'С промокодом' END AS тип\_покупки,

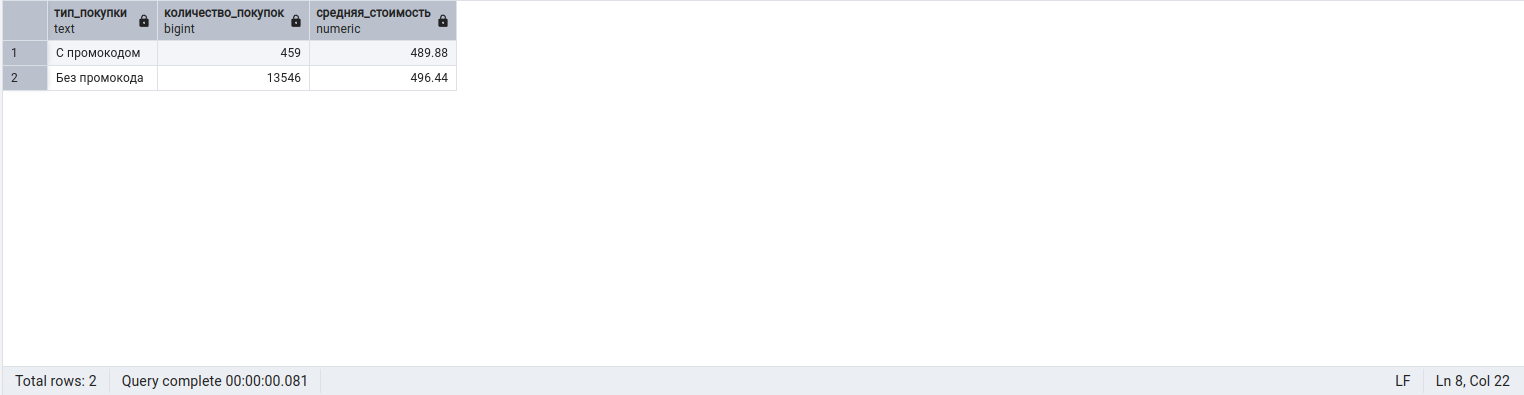
COUNT(user\_course.user\_id) AS количество\_покупок,

ROUND(AVG(Курс.стоимость), 2) AS средняя\_стоимость

FROM user\_course

LEFT JOIN promo\_user ON user\_course.user\_id = promo\_user.user\_id

JOIN Курс ON user\_course.course\_id = Курс.course\_id

GROUP BY тип\_покупки;  
  
  


1. Выявление неактивных пользователей(Запрос выявляет пользователей, которые зарегистрировались, но не проявили никакой активности (не приобрели курс и не оставили оценок). Анализ вовлеченности)  
     
   SELECT

u.логин AS username,

u.email AS email,

u.дата\_регистрации AS registration\_date

FROM

Пользователь u

LEFT JOIN

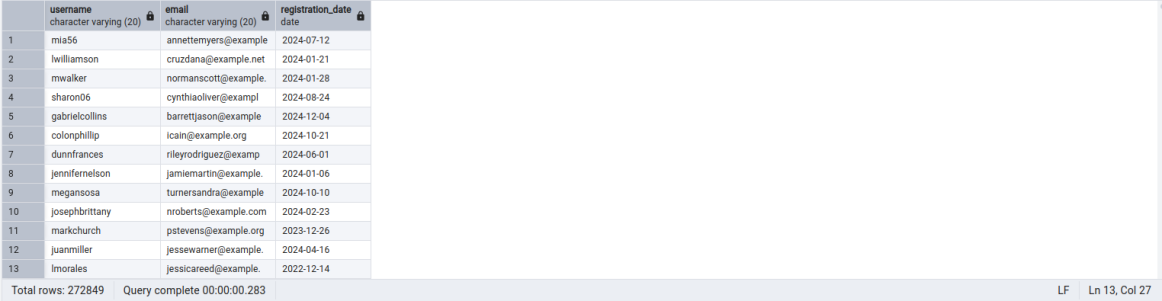
user\_course uc ON u.user\_id = uc.user\_id

LEFT JOIN

Оценка o ON u.user\_id = o.user\_id

WHERE

uc.user\_id IS NULL

AND o.user\_id IS NULL;  
  


1. курсы с наибольшим числом вложений:  
     
   SELECT

k.название AS course\_name,

COUNT(a.attachment\_id) AS attachments\_count

FROM

Курс k

JOIN

Урок l ON k.course\_id = l.course\_id

JOIN

Степ s ON l.lesson\_id = s.lesson\_id

JOIN

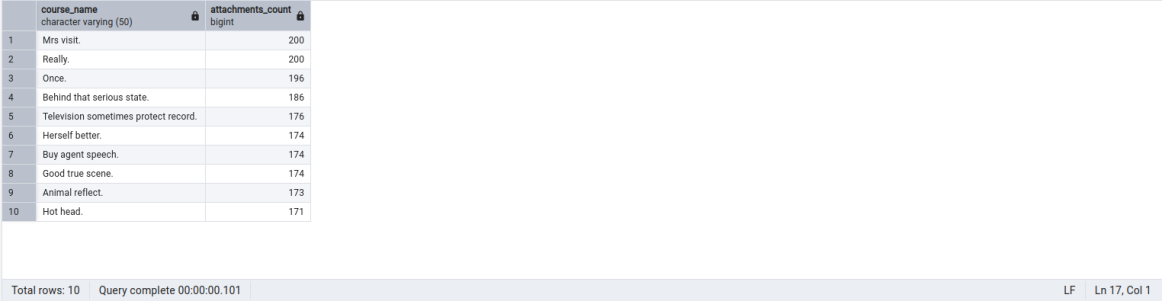
Attachment\_step a ON s.step\_id = a.step\_id

GROUP BY

k.course\_id

ORDER BY

attachments\_count DESC

LIMIT 10;