**Федеральное государственное автономное образовательное**

**учреждение высшего образования**

**«Национальный исследовательский университет «Высшая школа экономики»**

**Факультет Высшая Школа Бизнеса**

**КОНТРОЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ**

**Проектирование базы данных для онлайн-курсов и учебного контента**

**по направлению подготовки 38.03.05**

**образовательная программа**

**«Бизнес-информатика»**

**Выполнили:**

**Икромов Навруз Рустамович, ББИ 223**

**Мухлаев Алан Арслангович, ББИ 222**

**Горяев Даян Мергенович, ББИ 225**

**Чункуров Тимур Юрьевич, ББИ 223**

[**1. Общие сведения о предметной области 3**](#_iz1c3yfhvs3j)

[**2. Цель создания базы данных 3**](#_urb3w89i1e1g)

[**3. Возможные пользователи базы данных 3**](#_a1eemr4i4vk7)

[**4. Сценарии взаимодействия пользователей и БД 4**](#_b8kre86g2v1k)

[**5. Список основных сущностей и их атрибутов и типов данных 4**](#_a7yjkml3xccr)

[**6. Типы и описание связей между сущностями 6**](#_sdhbpnmo0rc8)

[**7. Инфологическая модель базы данных 6**](#_issrhiohu87w)

[**8. Модель в нотации IDEF1X 12**](#_8t4k49ucm5c)

[9. Заполнение базы данных тестовыми данными 12](#_78ad6170uw96)

[**10. Настройка индексов 13**](#_k9w1tdio0og6)

[**11. Разработка запросов 14**](#_r12q2erqyw1m)

[**12. Создание хранимых процедур 24**](#_8u1mo8mii2yj)

[**13. Создание триггера 27**](#_kby48fnco6um)

[**14. Создание представлений 28**](#_1li0ulkzi87e)

[**15. Создание функций 30**](#_r2nksolfjet7)

## 

## **1. Общие сведения о предметной области**

Предметная область включает разработку базы данных для управления онлайн-курсами и учебным контентом. Основная цель системы — организация и управление информацией о курсах, пользователях и их взаимодействии с учебным контентом. В систему входят различные функции для студентов, инструкторов и администраторов, что позволяет реализовать процесс обучения и взаимодействия с учебными материалами.

## **2. Цель создания базы данных**

Цель создания базы данных — предоставление централизованной системы для управления учебным контентом. Эта система должна поддерживать основные функции:

* Регистрация и управление пользователями
* Управление курсами и учебными материалами
* Проведение тестов и заданий
* Выдача сертификатов и отчетов
* Оценка курсов и обратная связь
* Отслеживание прогресса студентов

## **3. Возможные пользователи базы данных**

Возможными пользователями нашей базы данных могут стать любые образовательные учреждения, онлайн-платформы, компании и организации, занимающиеся разработкой и распространением учебного контента. С помощью нашей базы данных они смогут эффективно организовывать, анализировать и хранить информацию о курсах, образовательных программах, студентах и их успеваемости. Это позволит им получить более структурированное и подробное представление обо всех образовательных процессах, связанных с онлайн-обучением, а также использовать данные для улучшения учебных программ, разработки индивидуальных учебных траекторий и оценки различных показателей эффективности.

## **4. Сценарии взаимодействия пользователей и БД**

**Студенты**:

* Регистрация и создание профиля.
* Поиск и просмотр курсов.
* Запись на курс и доступ к материалам.
* Участие в тестах и выполнении заданий.
* Получение сертификатов.
* Отправление обратной связи.

**Инструкторы**:

* Создание и редактирование курсов.
* Загрузка видеолекций и учебных материалов.
* Создание тестов и контрольных заданий.
* Управление обратной связью от студентов.
* Отслеживание прогресса студентов.

**Администраторы**:

* Управление пользователями (добавление, редактирование, удаление).
* Модерация курсов и контента.
* Генерация отчетов по активности пользователей и успеваемости.

## **5. Список основных сущностей и их атрибутов и типов данных**

**Пользователи**

* ID пользователя (первичный ключ), тип данных:INT
* Имя, тип данных:VARCHAR(100)
* Email, тип данных:VARCHAR(100)
* Пароль, тип данных:VARCHAR(255)
* Номер телефона, тип данных:VARCHAR(16)
* Роль (студент, инструктор, администратор), тип данных:ENUM('Student', 'Instructor', 'Administrator')

**Курсы**

* ID курса (первичный ключ), тип данных:INT
* Название, тип данных:VARCHAR(100)
* Описание, тип данных:TEXT
* Категория, тип данных:VARCHAR(50)
* Инструктор (внешний ключ на пользователей), тип данных:INT

**Модули курсов**

* ID модуля (первичный ключ), тип данных:INT
* Название, тип данных:VARCHAR(100)
* Описание, тип данных:TEXT
* ID курса (внешний ключ), тип данных:INT

**Видеолекции**

* ID видеолекции (первичный ключ), тип данных:INT
* Название, тип данных:VARCHAR(100)
* URL, тип данных:VARCHAR(255)
* ID модуля (внешний ключ), тип данных:INT

**Тесты и задания**

* ID теста (первичный ключ), тип данных:INT
* Название, тип данных:VARCHAR(100)
* Вопросы, тип данных:TEXT
* ID модуля (внешний ключ), тип данных:INT

**Сертификаты**

* ID сертификата (первичный ключ), тип данных:INT
* ID пользователя (внешний ключ), тип данных:INT
* ID курса (внешний ключ), тип данных:INT
* Дата выдачи, тип данных:DATE

**Обратная связь**

* ID обратной связи (первичный ключ), тип данных:INT
* ID пользователя (внешний ключ), тип данных:INT
* ID курса (внешний ключ), тип данных:INT
* Оценка, тип данных:TINYINT (1-5)
* Комментарий, тип данных:TEXT
* Дата, тип данных:DATE

**Прогресс студентов**

* ID прогресса (первичный ключ), тип данных:INT
* ID пользователя (внешний ключ), тип данных:INT
* ID курса (внешний ключ), тип данных:INT
* ID модуля(внешний ключ), тип данных:INT
* Статус завершения, тип данных:ENUM('завершено', 'в процессе', 'не начато')
* Дата завершения, тип данных:DATE

## **6. Типы и описание связей между сущностями**

* Один студент может быть записан на несколько курсов, и один курс может иметь много студентов. (Многие ко многим)
* Один курс может содержать несколько модулей. (Один ко многим)
* Один модуль может включать несколько видеолекций. (Один ко многим)
* Один модуль может включать несколько тестов и заданий. (Один ко многим)
* Один студент может получить несколько сертификатов по завершении курсов. (Один ко многим)
* Один студент может оставить несколько отзывов. (Один ко многим)
* Один курс может включать несколько отзывов. (Один ко многим)
* Один студент имеет прогресс по нескольким курсам и модулям. (Один ко многим)
* Один курс имеет прогресс у многих студентов. (Один ко многим)
* Один модуль имеет прогресс у многих студентов. (Один ко многим)

## **7. Инфологическая модель базы данных**

**Сущности и атрибуты:**

Пользователи (Users)

* **ID пользователя (User\_ID)** - первичный ключ, тип данных: INT
* **Имя (Name)** - тип данных: VARCHAR(100)
* **Email (Email)** - тип данных: VARCHAR(100)
* **Пароль (Password)** - тип данных: VARCHAR(255)
* **Номер телефона (PhoneNumber)** - тип данных: VARCHAR(16)
* **Роль (Role)** - студент, инструктор, администратор, тип данных: ENUM('студент', 'инструктор', 'администратор')

Курсы (Courses)

1. **ID курса (Course\_ID)** - первичный ключ, тип данных: INT
2. **Название (Title)** - тип данных: VARCHAR(100)
3. **Описание (Description)** - тип данных: TEXT
4. **Категория (Category)** - тип данных: VARCHAR(50)
5. **Инструктор (Instructor\_ID)** - внешний ключ на Users, тип данных: INT

Модули курсов (Course\_Modules)

1. **ID модуля (Module\_ID)** - первичный ключ, тип данных: INT
2. **Название (Title)** - тип данных: VARCHAR(100)
3. **Описание (Description)** - тип данных: TEXT
4. **ID курса (Course\_ID)** - внешний ключ на Courses, тип данных: INT

Видеолекции (Video\_Lectures)

1. **ID видеолекции (Video\_ID)** - первичный ключ, тип данных: INT
2. **Название (Title)** - тип данных: VARCHAR(100)
3. **URL (URL)** - тип данных: VARCHAR(255)
4. **ID модуля (Module\_ID)** - внешний ключ на Course\_Modules, тип данных: INT

Тесты и задания (Tests)

1. **ID теста (Test\_ID)** - первичный ключ, тип данных: INT
2. **Название (Title)** - тип данных: VARCHAR(100)
3. **Вопросы (Questions)** - тип данных: TEXT
4. **ID модуля (Module\_ID)** - внешний ключ на Course\_Modules, тип данных: INT

Сертификаты (Certificates)

1. **ID сертификата (Certificate\_ID)** - первичный ключ, тип данных: INT
2. **ID пользователя (User\_ID)** - внешний ключ на Users, тип данных: INT
3. **ID курса (Course\_ID)** - внешний ключ на Courses, тип данных: INT
4. **Дата выдачи (Issued\_Date)** - тип данных: DATE

Обратная связь (Feedback)

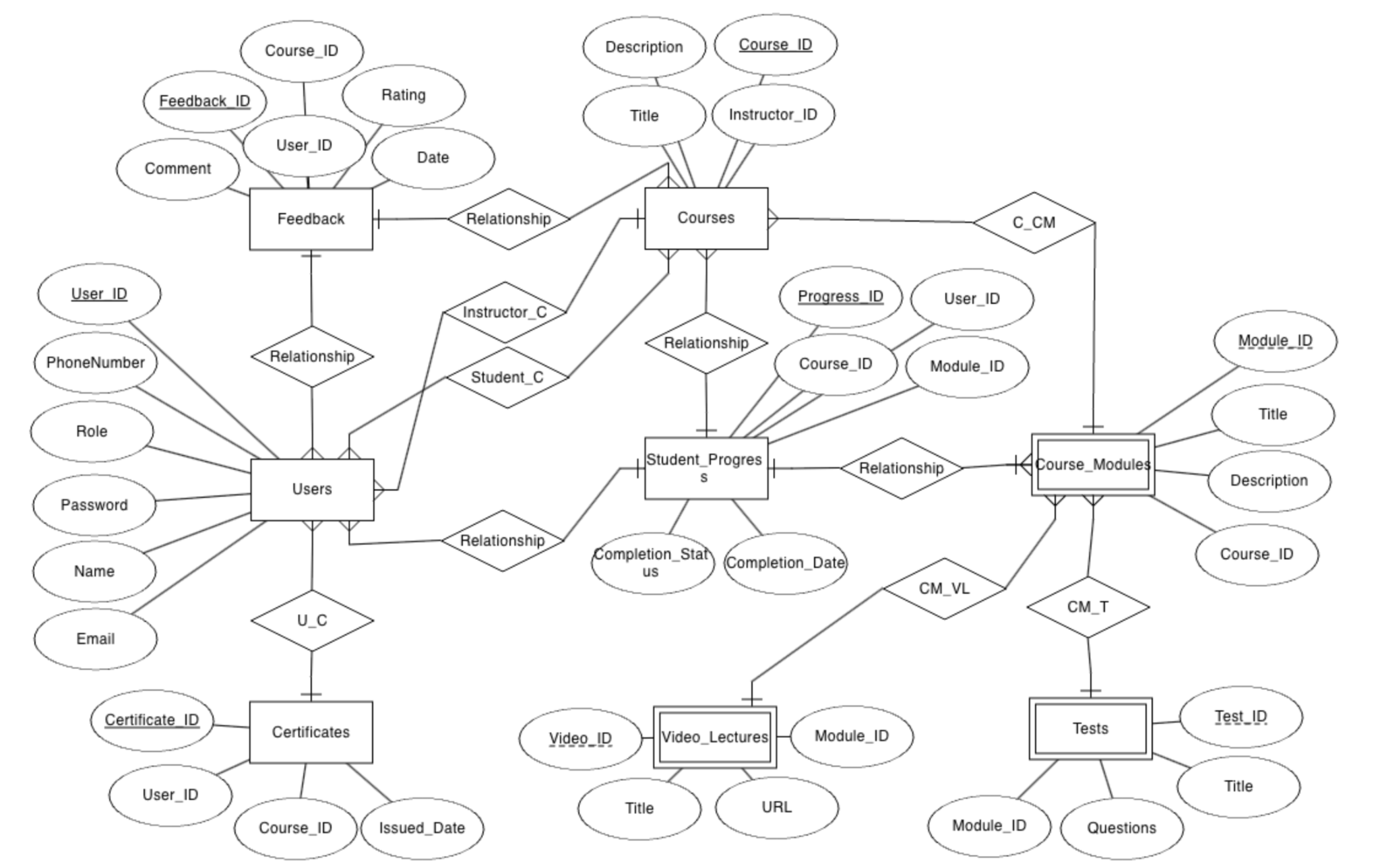
1. **ID обратной связи (Feedback\_ID)** - первичный ключ, тип данных: INT
2. **ID пользователя (User\_ID)** - внешний ключ на Users, тип данных: INT
3. **ID курса (Course\_ID)** - внешний ключ на Courses, тип данных: INT
4. **Оценка (Rating)** - тип данных: TINYINT (1-5)
5. **Комментарий (Comment)** - тип данных: TEXT
6. **Дата (Date)** - тип данных: DATE

Прогресс студентов (Student\_Progress)

1. **ID прогресса (Progress\_ID)** - первичный ключ, тип данных: INT
2. **ID пользователя (User\_ID)** - внешний ключ на Users, тип данных: INT
3. **ID курса (Course\_ID)** - внешний ключ на Courses, тип данных: INT
4. **ID модуля (Module\_ID)** - внешний ключ на Course\_Modules, тип данных: INT
5. **Статус завершения (Completion\_Status)** - тип данных: ENUM('завершено', 'в процессе', 'не начато')
6. **Дата завершения (Completion\_Date)** - тип данных: DATE

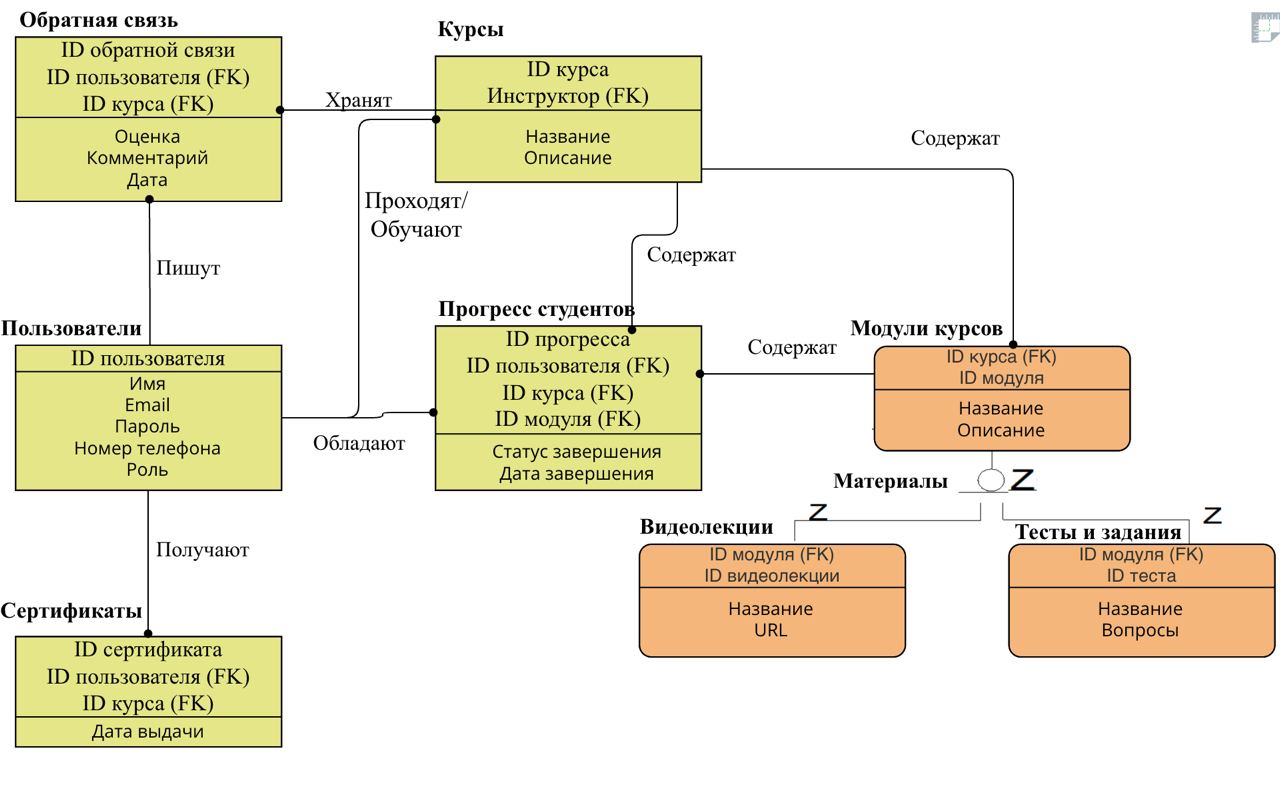
**Типы связей между сущностями:**

1. **Пользователи и Курсы**:
   * Один инструктор может создавать несколько курсов (Один ко многим).
   * Один студент может быть записан на несколько курсов, и один курс может иметь много студентов (Многие ко многим).
2. **Курсы и Модули**:
   * Один курс может содержать несколько модулей (Один ко многим).
3. **Модули и Видеолекции**:
   * Один модуль может включать несколько видеолекций (Один ко многим).
4. **Модули и Тесты**:
   * Один модуль может включать несколько тестов (Один ко многим).
5. **Пользователи и Сертификаты**:
   * Один студент может получить несколько сертификатов по завершении курсов (Один ко многим).
6. **Пользователи и Обратная связь:**
   * Один студент может оставить несколько отзывов (Один ко многим).
7. **Курсы и Обратная связь:**
   * Один курс может включать несколько отзывов (Один ко многим).
8. **Пользователи и Прогресс студентов:**
   * Один студент имеет прогресс по нескольким курсам и модулям (Один ко многим).
9. **Курсы и Прогресс студентов:**
   * Один курс имеет прогресс у многих студентов (Один ко многим).
10. **Модули курсов и Прогресс студентов:**
    * Один модуль имеет прогресс у многих студентов (Один ко многим).

**Проверка диаграммы:**

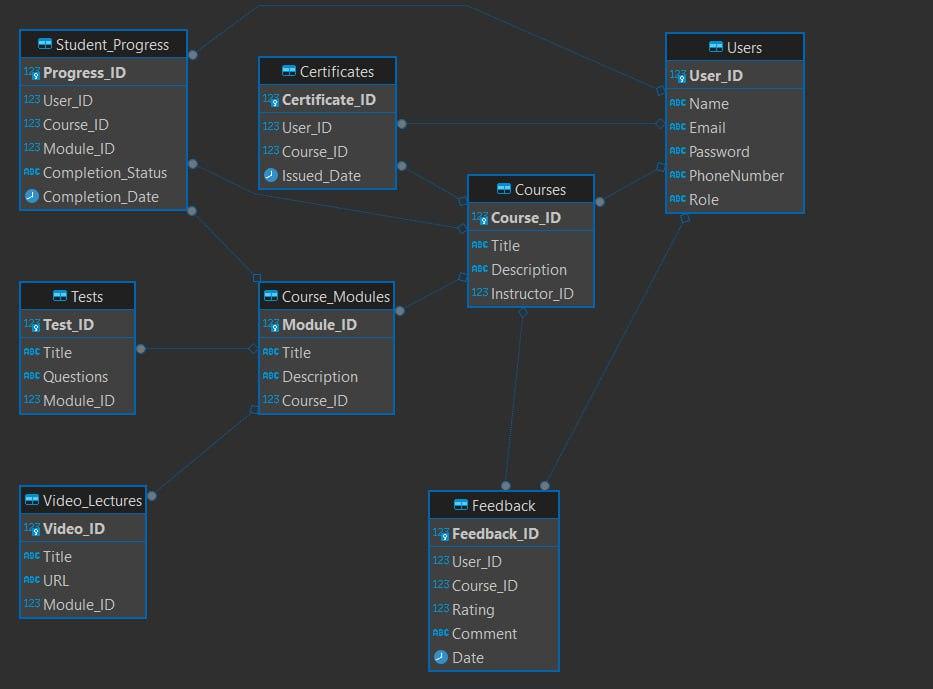
1. **Первая нормальная форма (1NF):**
   * Все атрибуты на диаграмме атомарны. Например, Name, Email, Password и т.д. являются неделимыми значениями.
   * Нет множественных значений в одном атрибуте.
2. **Вторая нормальная форма (2NF):**
   * Все сущности имеют первичные ключи.
   * Все неключевые атрибуты зависят от всего первичного ключа.
3. **Третья нормальная форма (3NF):**
   * Проверка на транзитивные зависимости.
   * Рассмотрим каждую сущность:
     1. **Users:** Все атрибуты напрямую зависят от User\_ID.
     2. **Certificates:** Все атрибуты напрямую зависят от составного ключа (Certificate\_ID, User\_ID).
     3. **Feedback:** Все атрибуты напрямую зависят от составного ключа (Feedback\_ID, User\_ID, Course\_ID).
     4. **Courses:** Все атрибуты напрямую зависят от Course\_ID.
     5. **Student\_Progress:** Все атрибуты напрямую зависят от составного ключа (Progress\_ID, User\_ID, Course\_ID, Module\_ID).
     6. **Course\_Modules:** Все атрибуты напрямую зависят от Module\_ID.
     7. **Video\_Lectures:** Все атрибуты напрямую зависят от Video\_ID.
     8. **Tests:** Все атрибуты напрямую зависят от Test\_ID.
4. **Диаграмма соответствует 1NF, 2NF, 3NF.**

## **8. Модель в нотации IDEF1X**

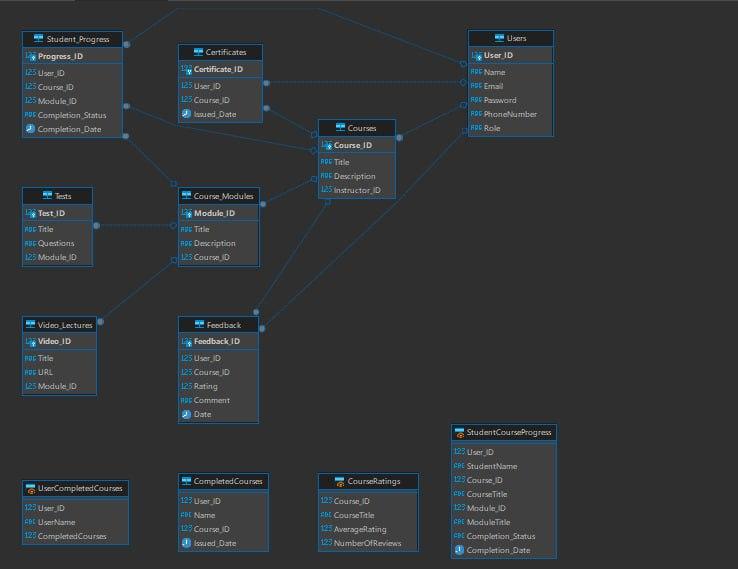


## **9. Физическая модель**

Первоначальная физическая модель

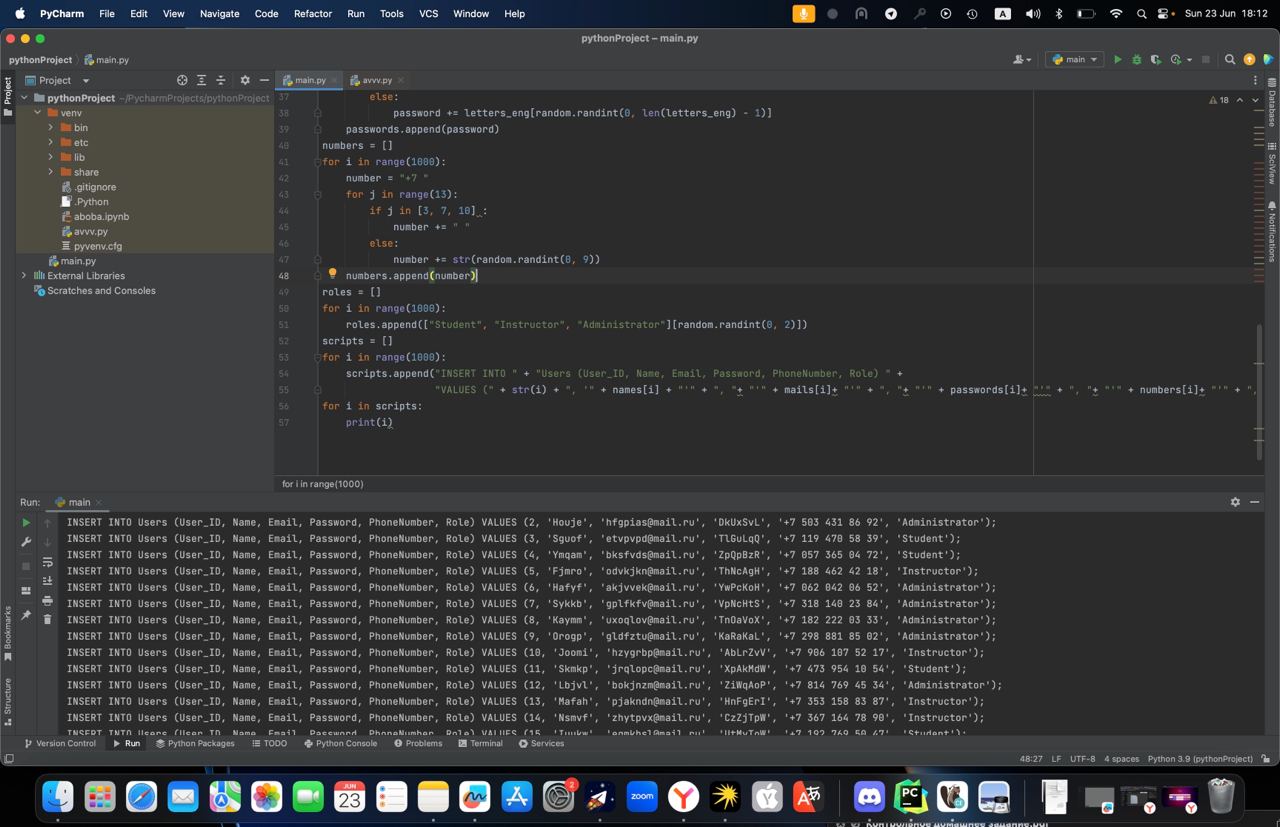


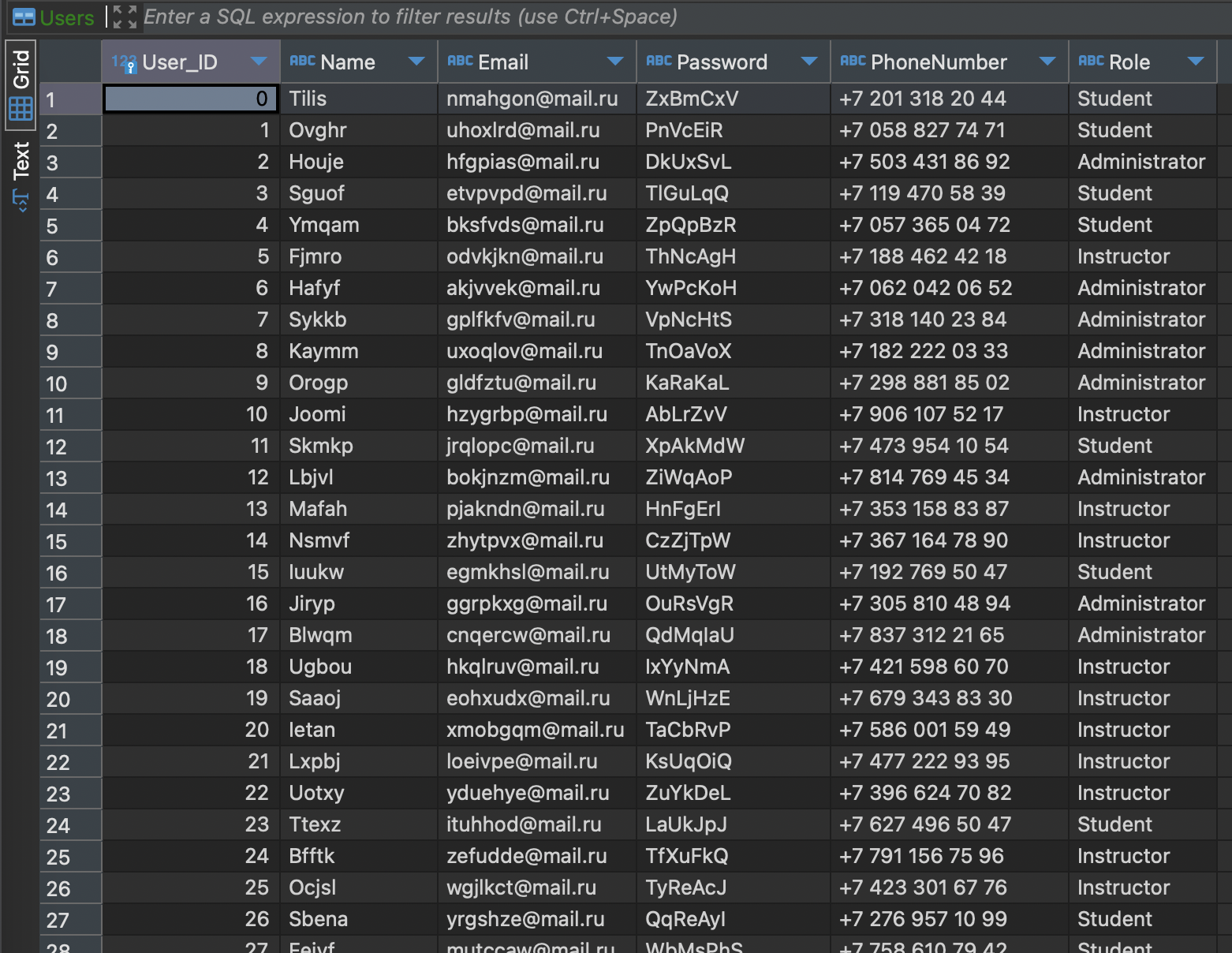
Конечная модель



## **10. Заполнение базы данных тестовыми данными**

1. **Описание данных**:
   * Пользователи (Users): 1000 записей
   * Курсы (Courses): 4 записи
   * Модули курсов (Course\_Modules): 4 записи
   * Видеолекции (Video\_Lectures): 4 записи
   * Тесты и задания (Tests): 4 записи
   * Сертификаты (Certificates): 4 записи
   * Обратная связь (Feedback): 4 записи
   * Прогресс студентов (Student\_Progress): 4 записи
2. **Методика заполнения**:
   * Каждую таблицу мы заполнили случайными данными. Пример заполнения таблиц:



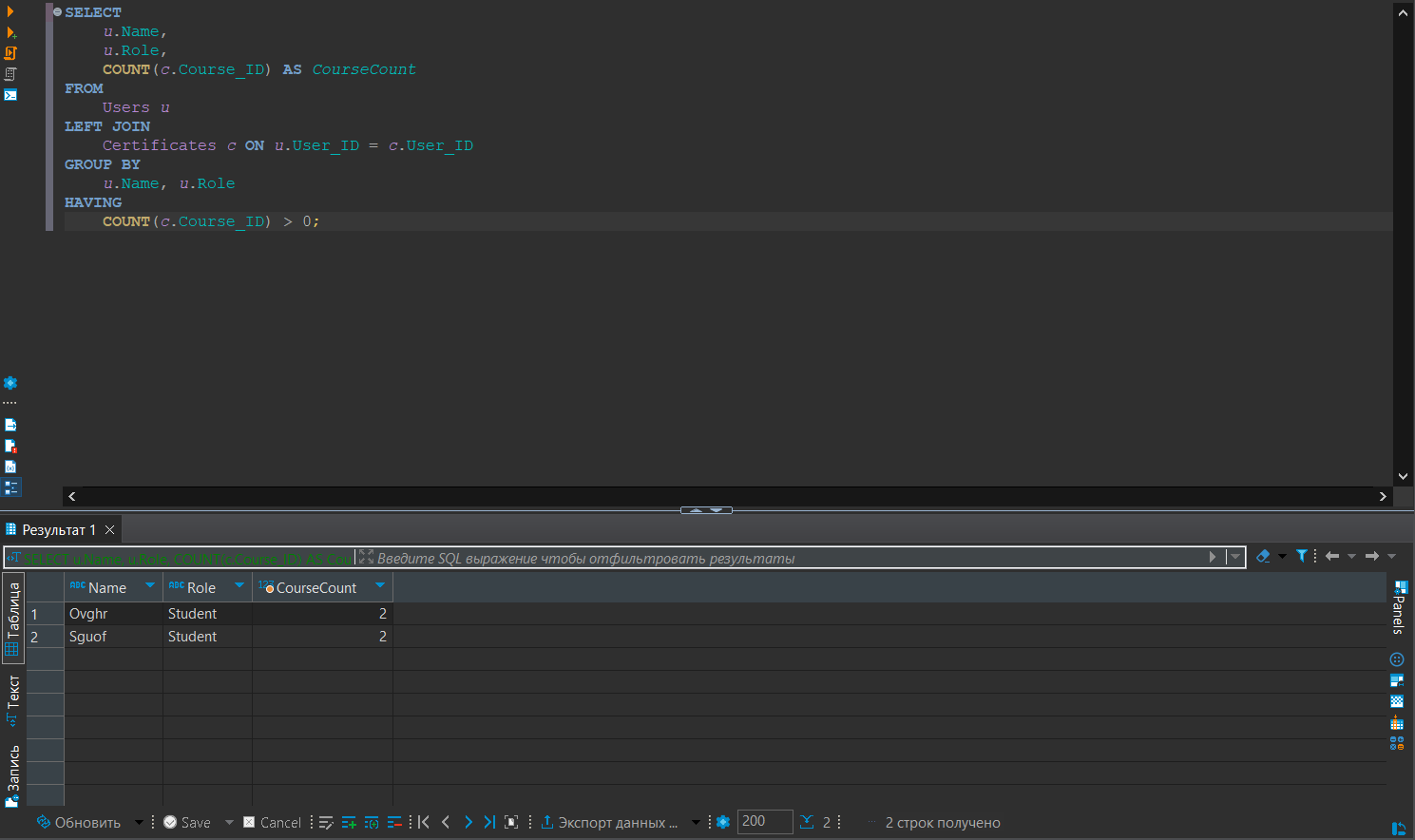


## **11. Настройка индексов**

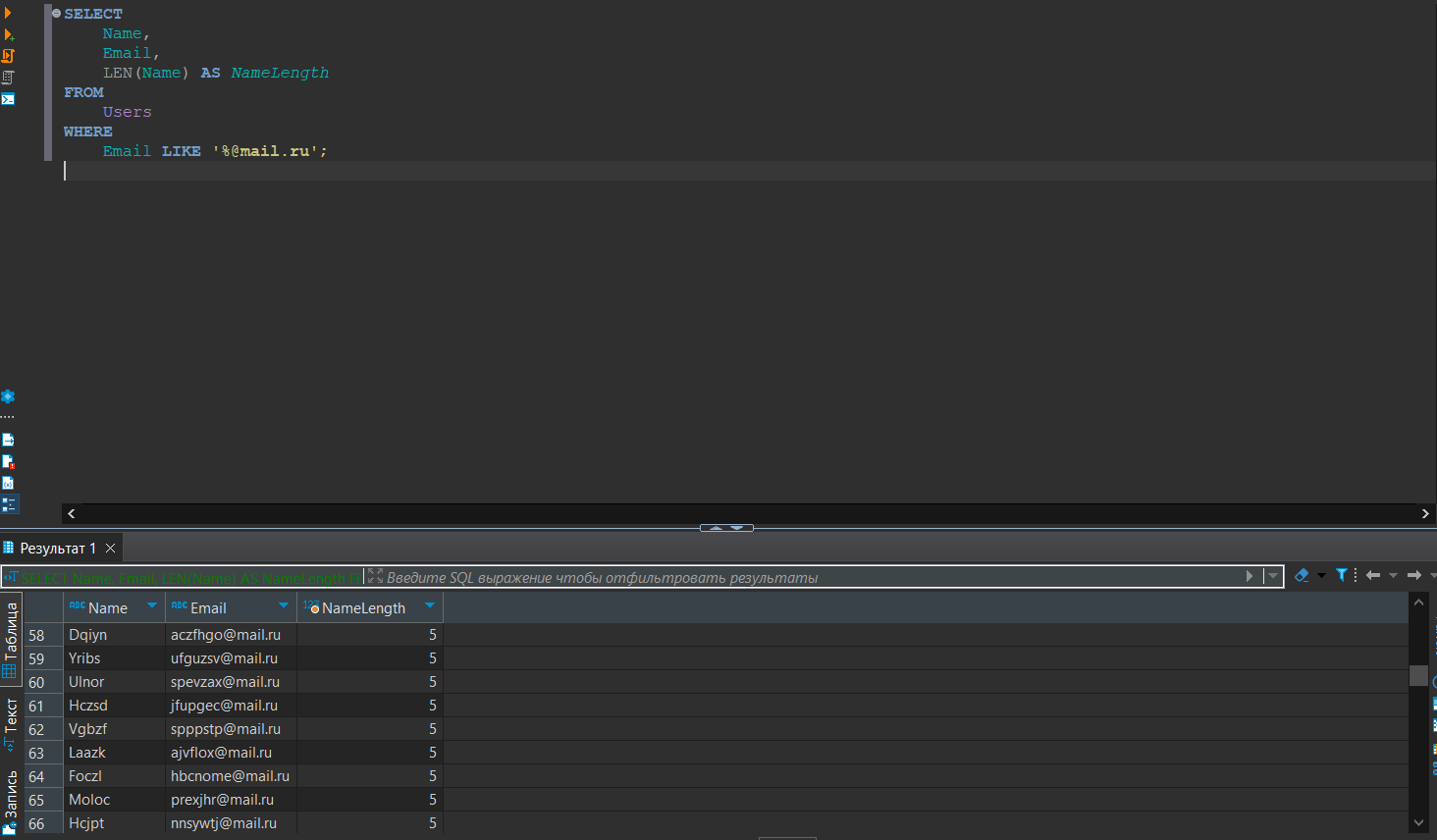
Индексы были настроены на уровне кода

## **12. Разработка запросов**

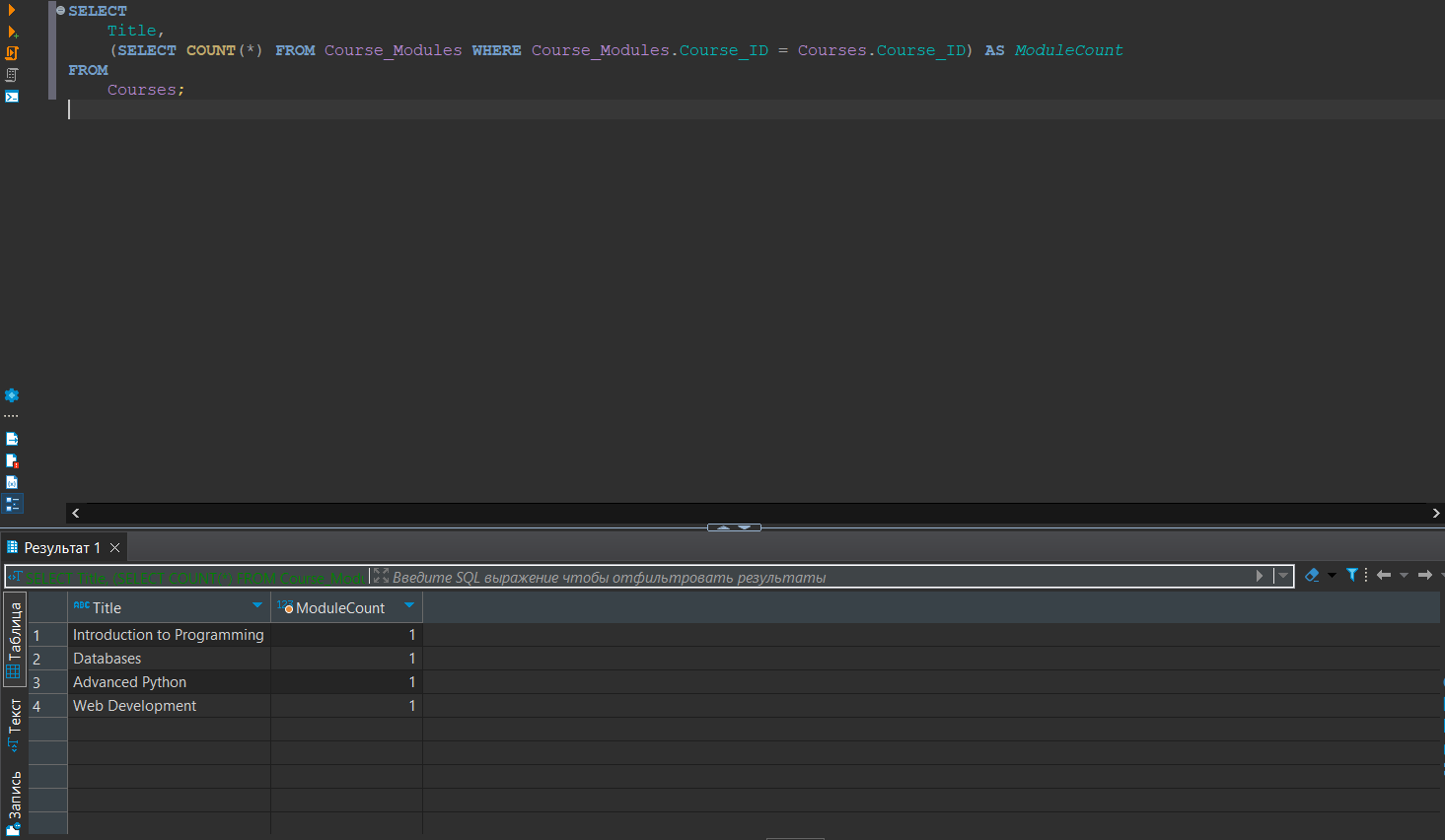
1. Запрос для получения имен пользователей, их ролей и количество курсов, на которые они записаны (Простой запрос с условием и формулами в SELECT):



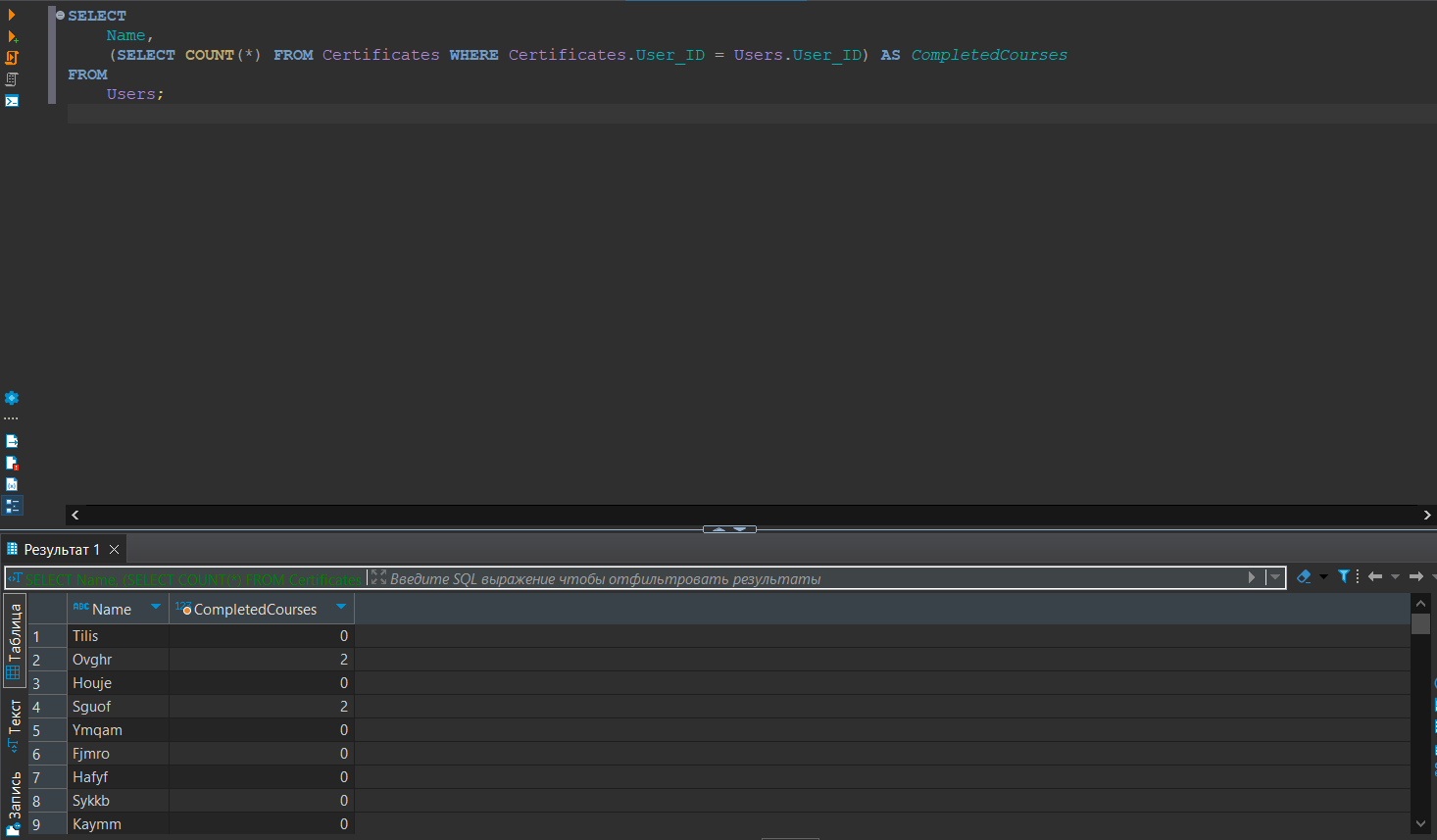
1. Вывести пользователей, у которых в email указан домен "mail.ru", и посчитать длину их имени (Простой запрос с условием и формулами в SELECT):



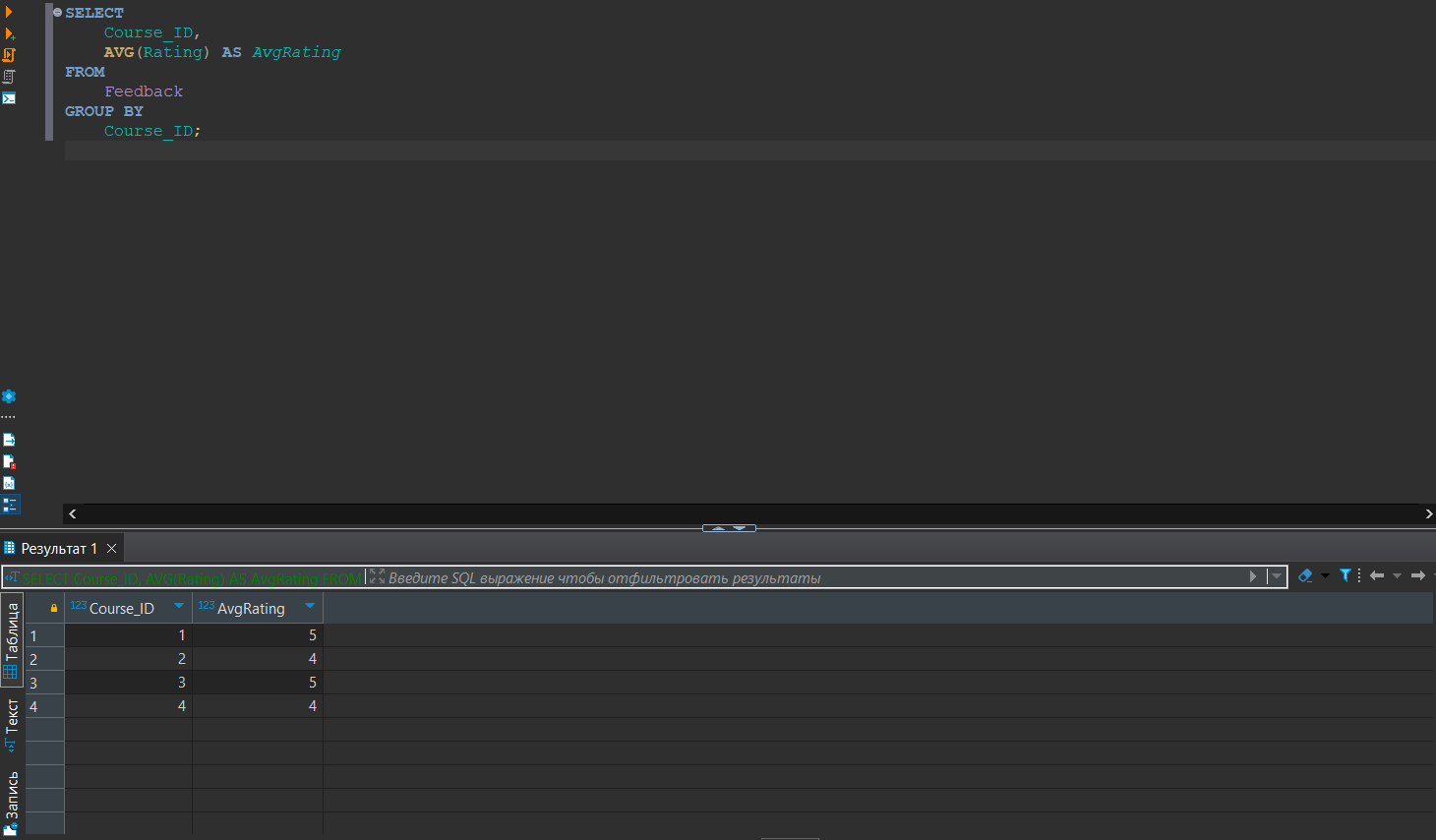
1. Вывести название курсов и количество модулей в каждом курсе (Запрос с коррелированным подзапросом в SELECT):



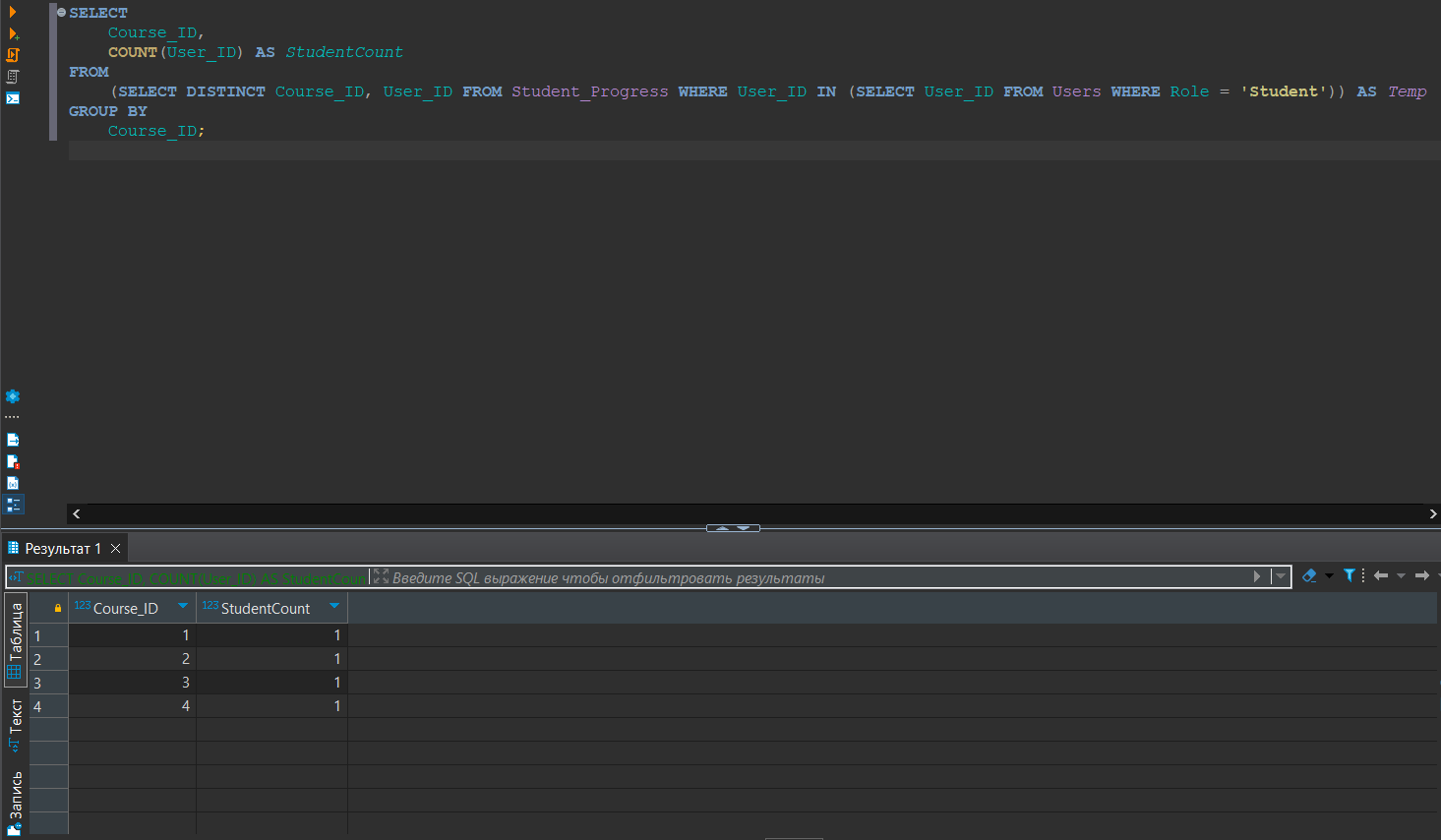
1. Вывести имя пользователей и количество курсов, которые они закончили (Запрос с коррелированным подзапросом в SELECT):



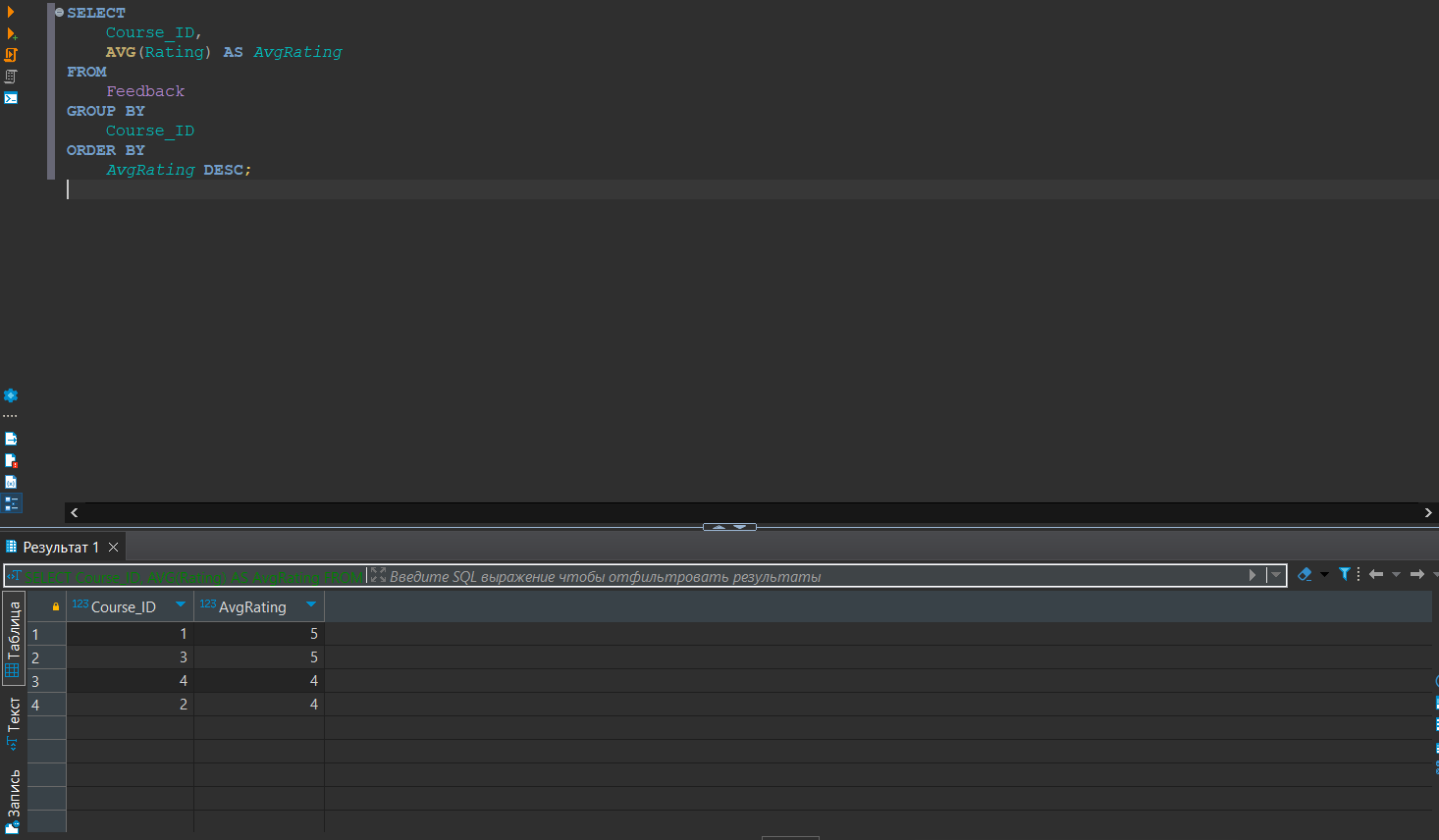
1. Вывести средний рейтинг курсов (Запрос с подзапросом в FROM):



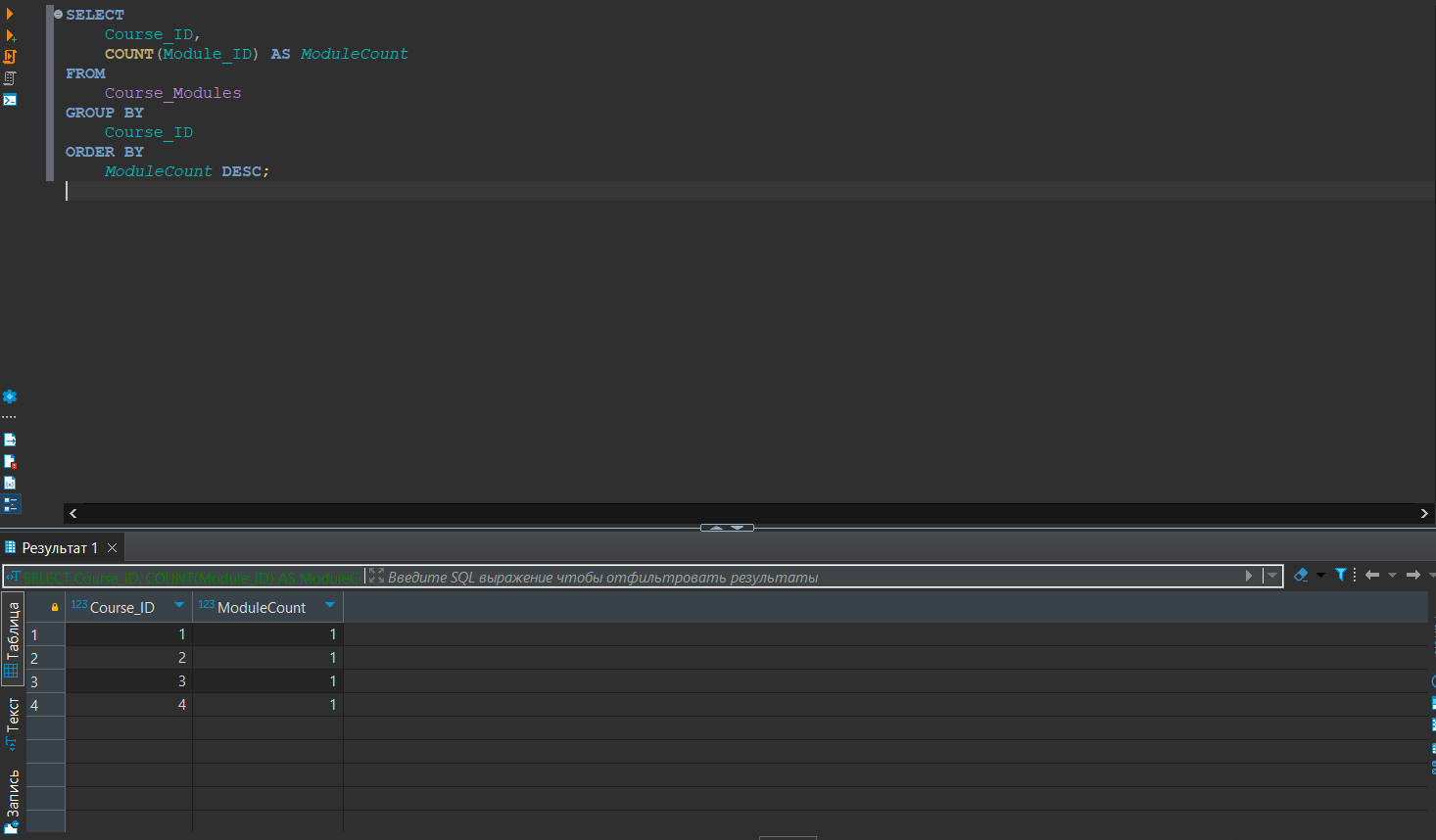
1. Вывести количество пользователей с ролью "Student" в каждом курсе (Запрос с подзапросом в FROM):



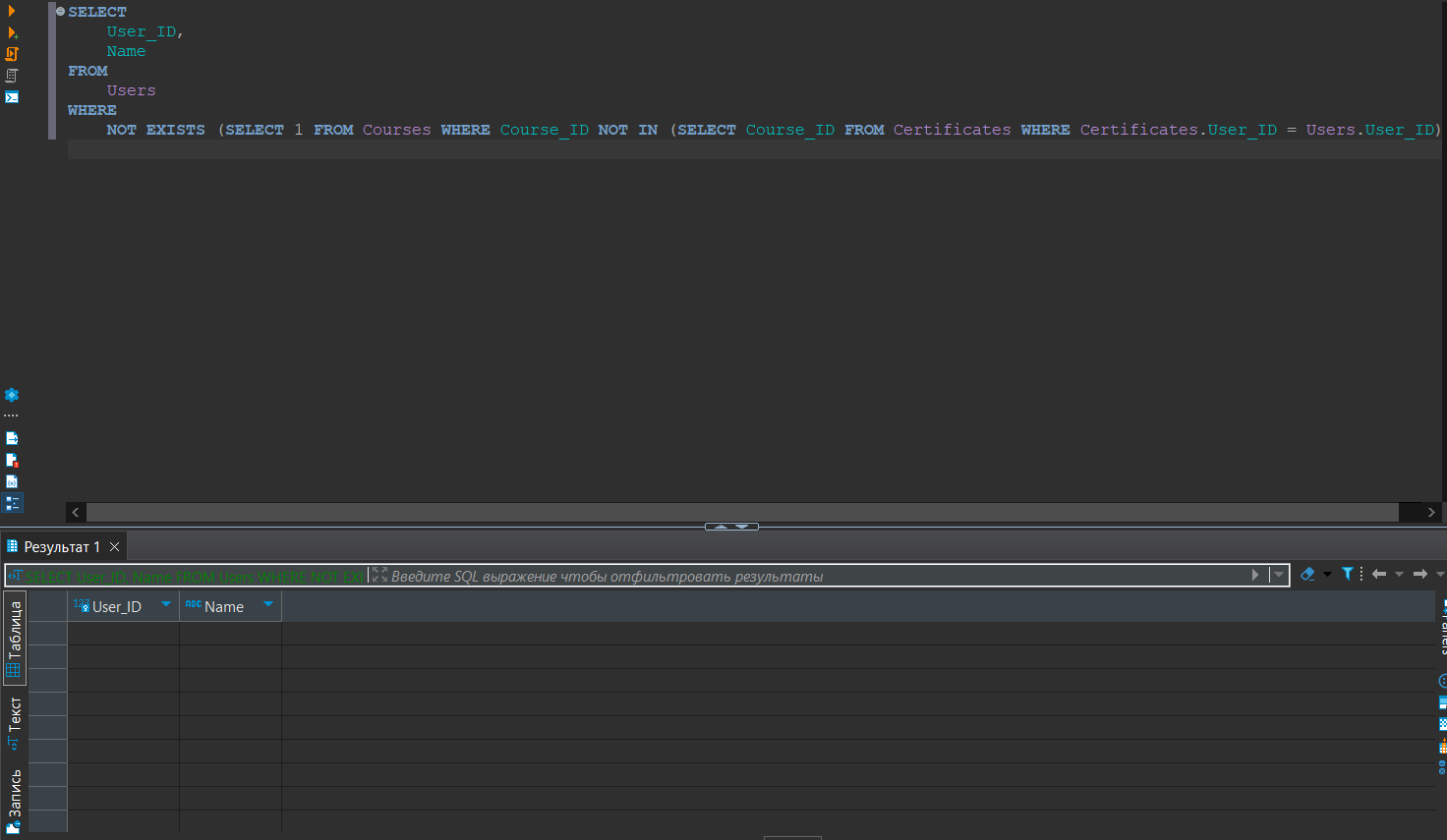
1. Вывести средний рейтинг курсов и отсортировать их по убыванию рейтинга (Запрос с подзапросом в FROM, агрегированием, группировкой и сортировкой):



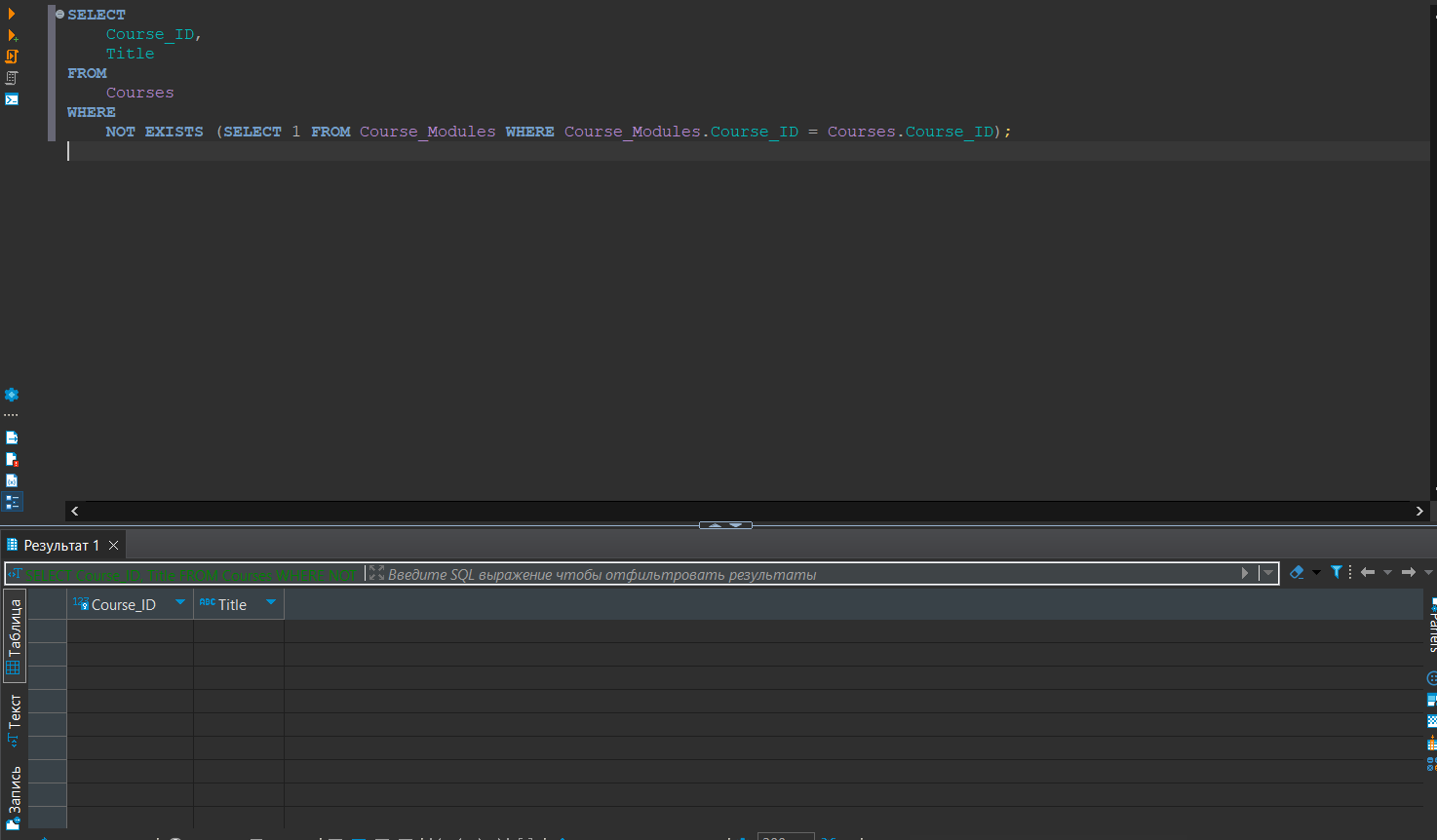
1. Вывести количество модулей в каждом курсе и отсортировать их по убыванию количества модулей (Запрос с подзапросом в FROM, агрегированием, группировкой и сортировкой):



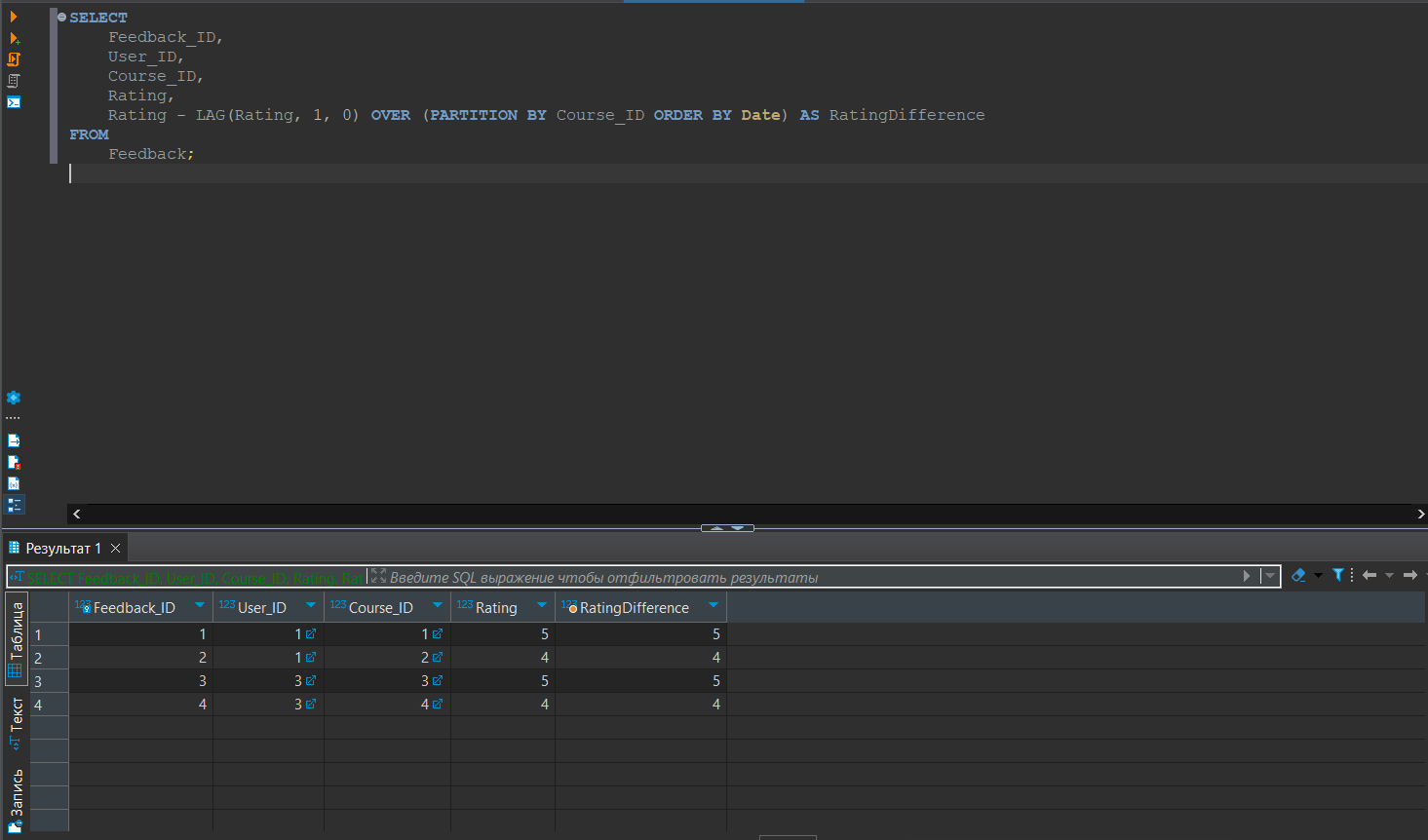
1. Вывести пользователей, которые завершили все курсы (Запрос с коррелированным подзапросом в WHERE):



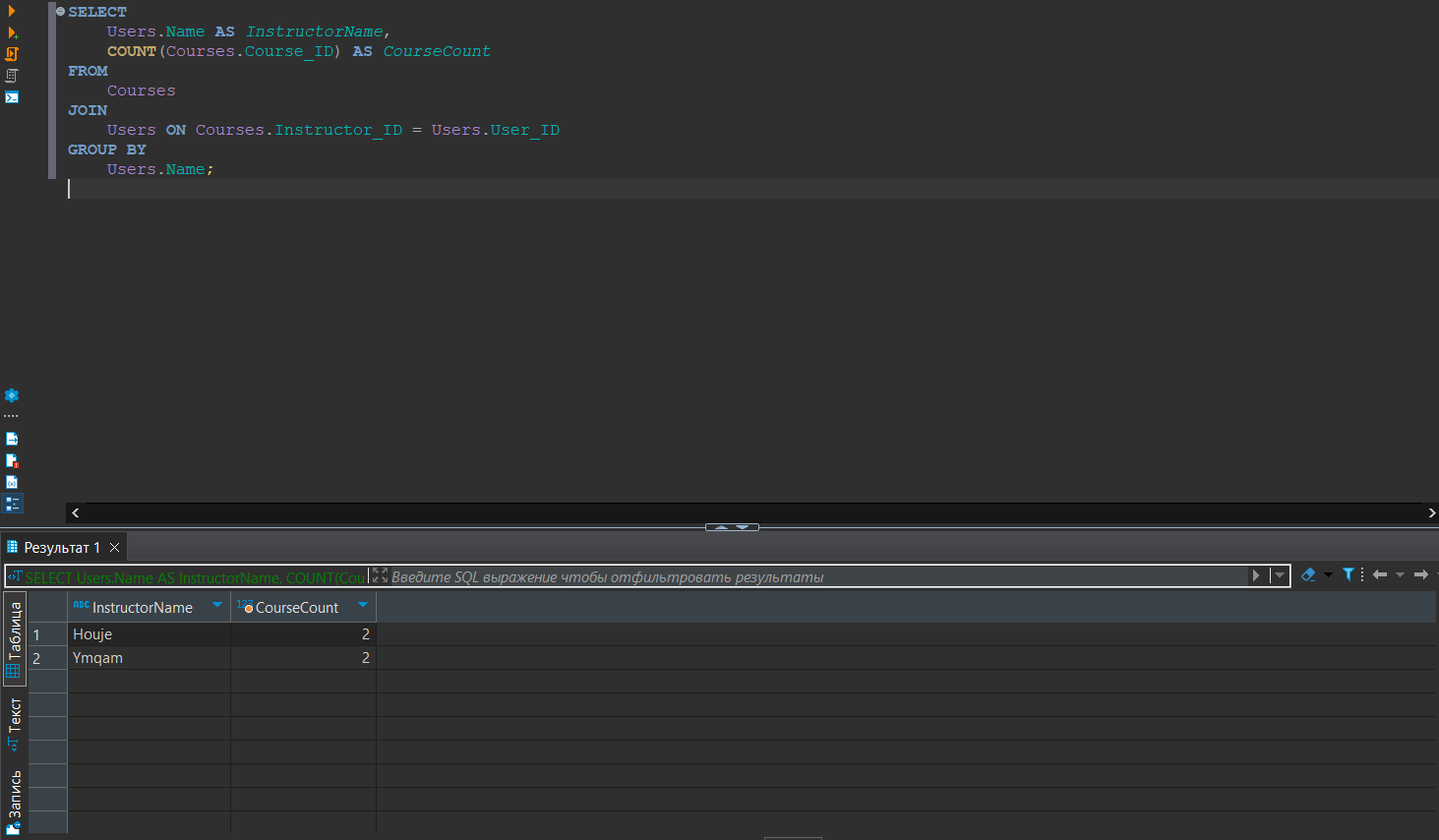
1. Вывести курсы, которые не имеют модулей (Запрос с коррелированным подзапросом в WHERE):



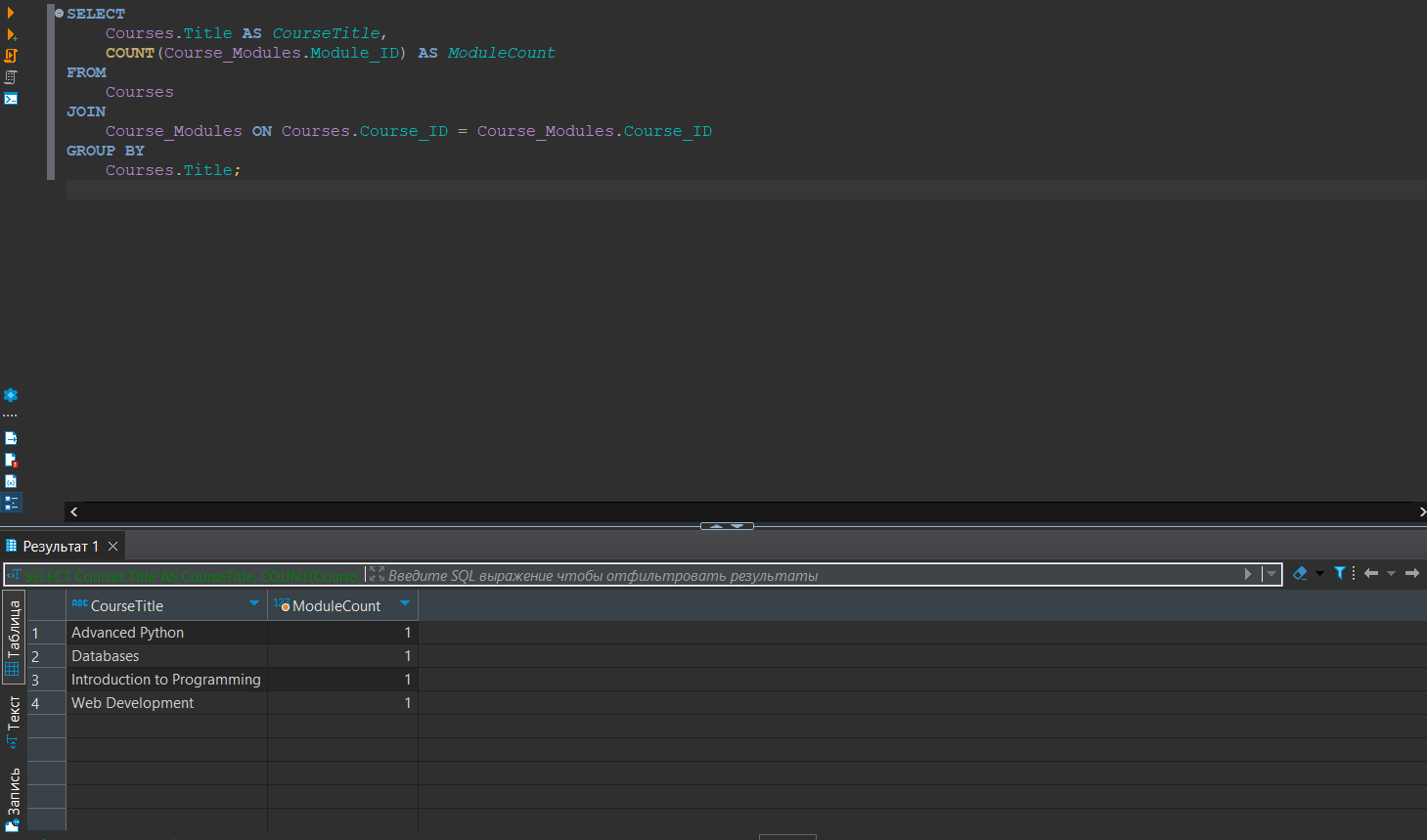
1. Вывести разницу в рейтинге курсов между текущим и предыдущим отзывом (Запрос, использующий оконную функцию LAG или LEAD для выполнения сравнения данных в разных периодах ):



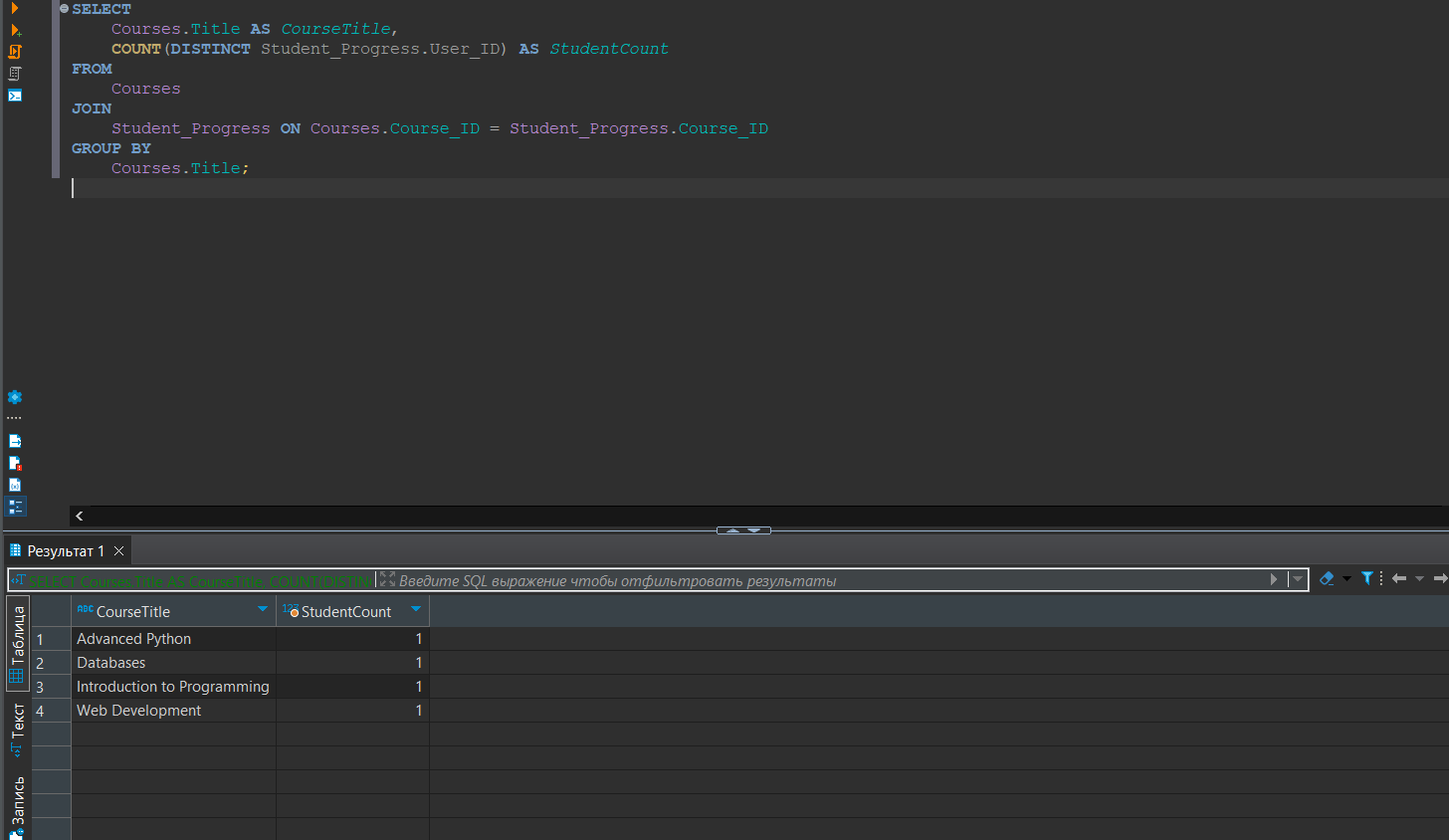
1. Вывести количество курсов, которые ведет каждый инструктор (Запрос с агрегированием и выражением JOIN):



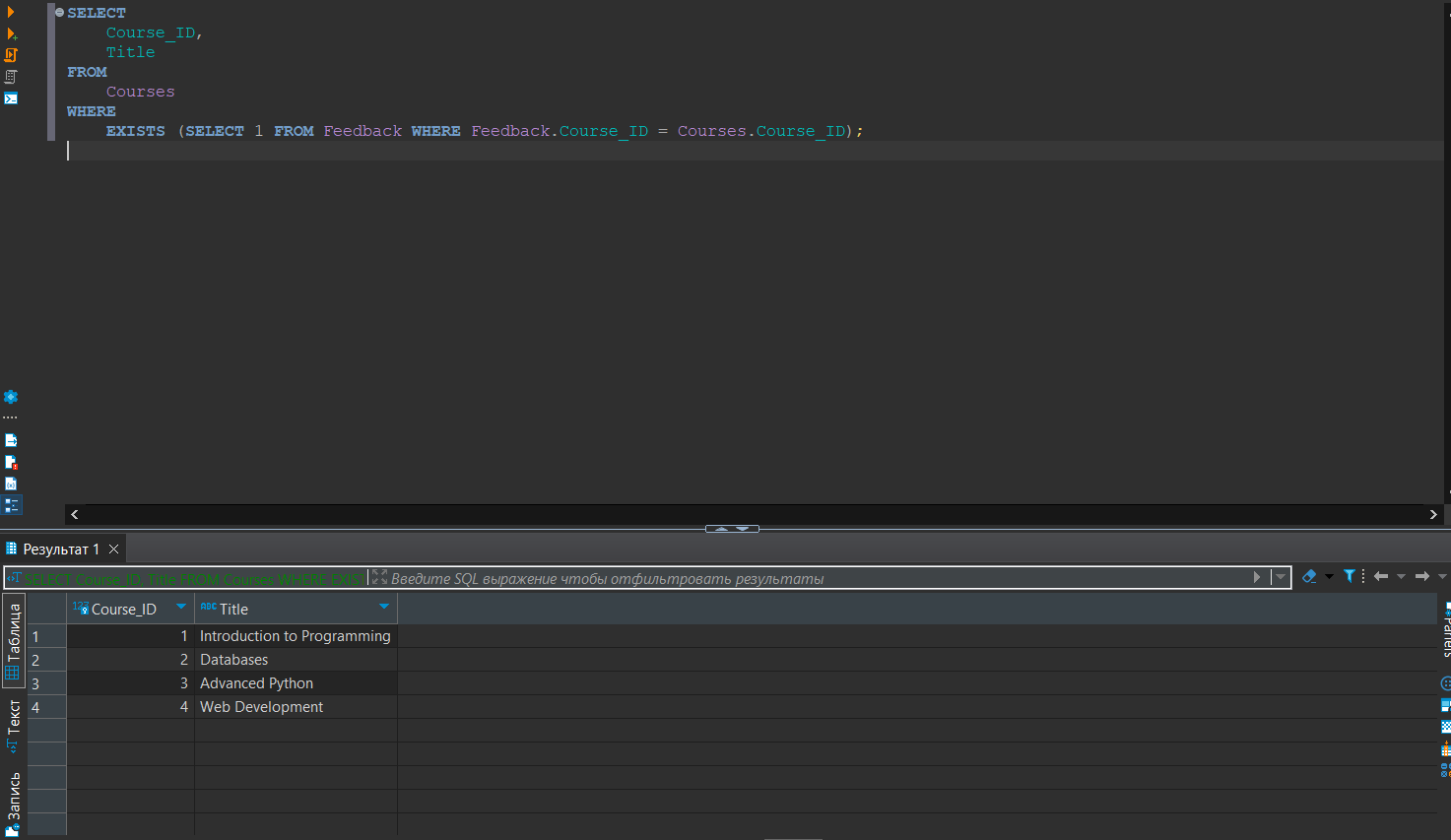
1. Вывести количество модулей в каждом курсе (Запрос с агрегированием и выражением JOIN):



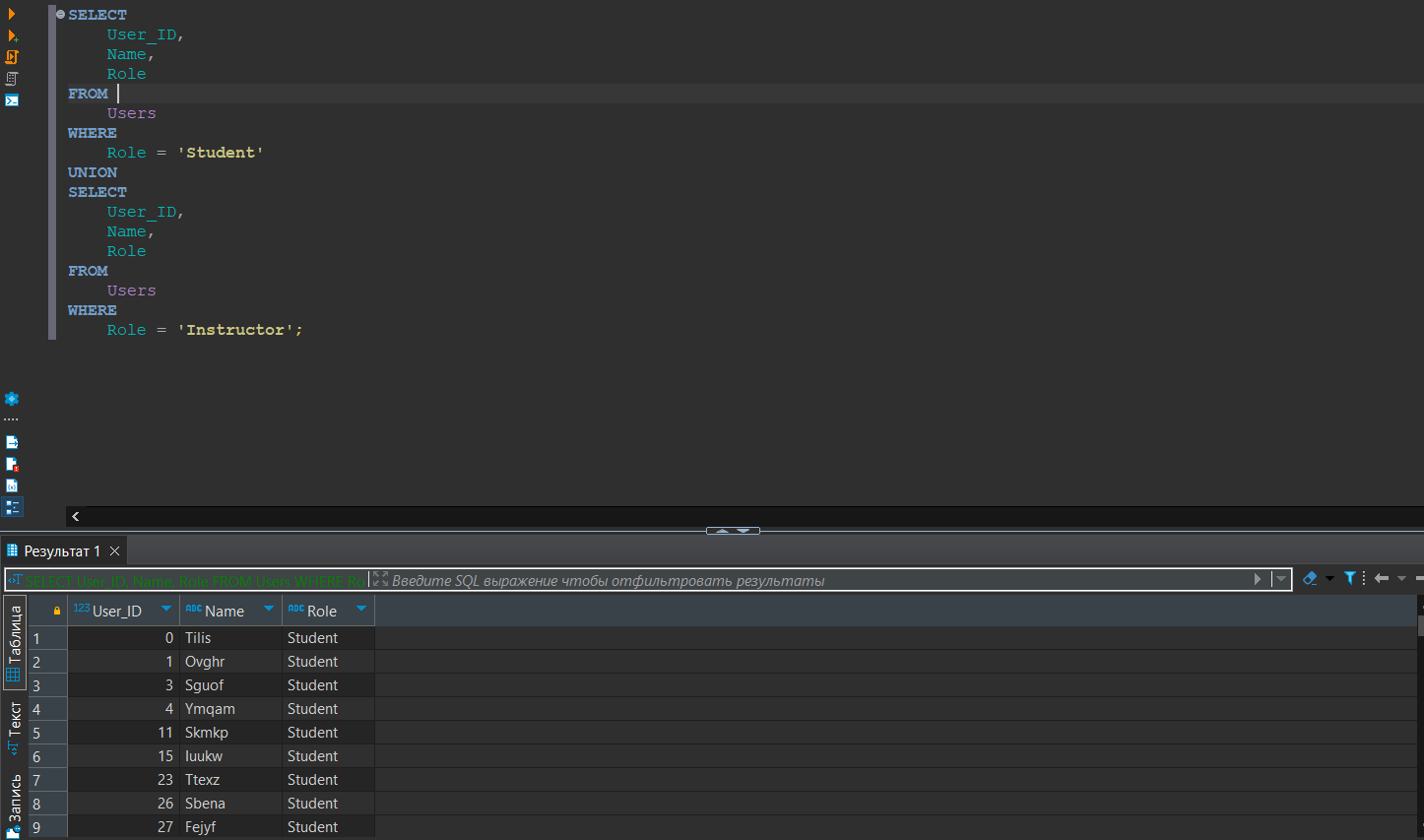
1. Вывести количество студентов, зарегистрированных на каждый курс (Запрос с агрегированием и выражением JOIN):



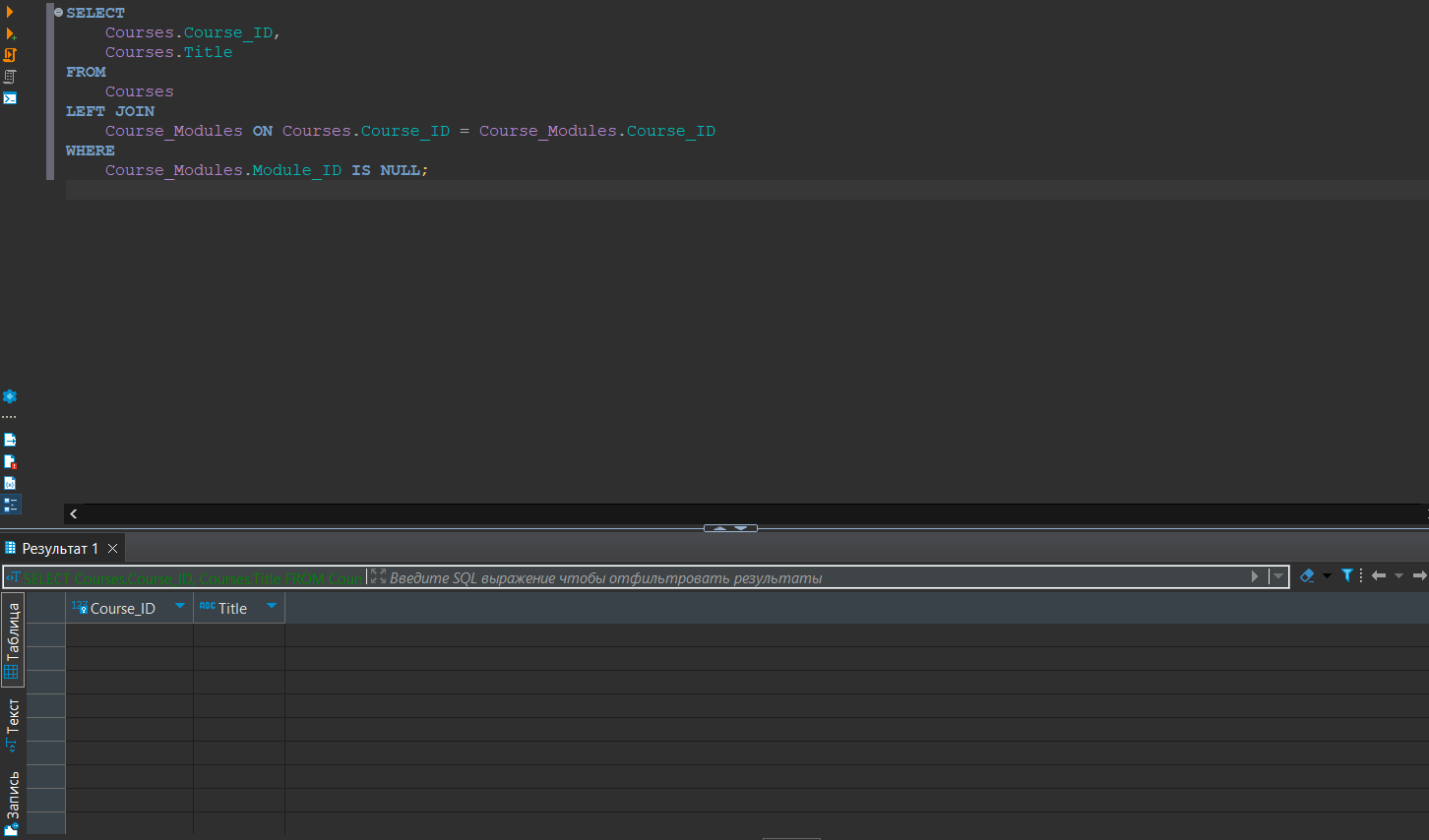
1. Вывести курсы, на которые есть отзывы (Запрос с EXISTS):



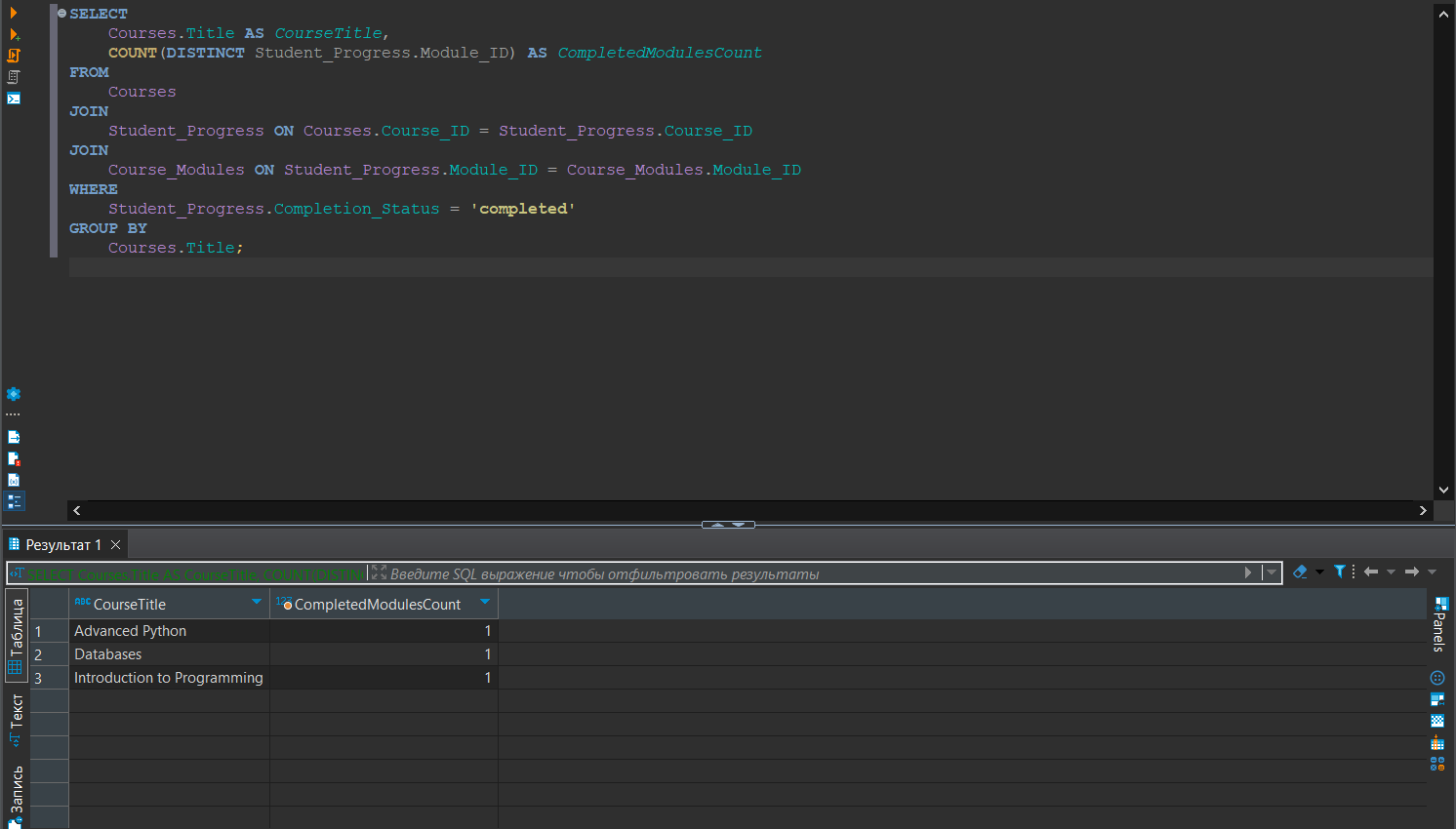
1. Вывести всех пользователей, которые являются либо студентами, либо инструкторами (Запрос, использующий манипуляции с множествами):



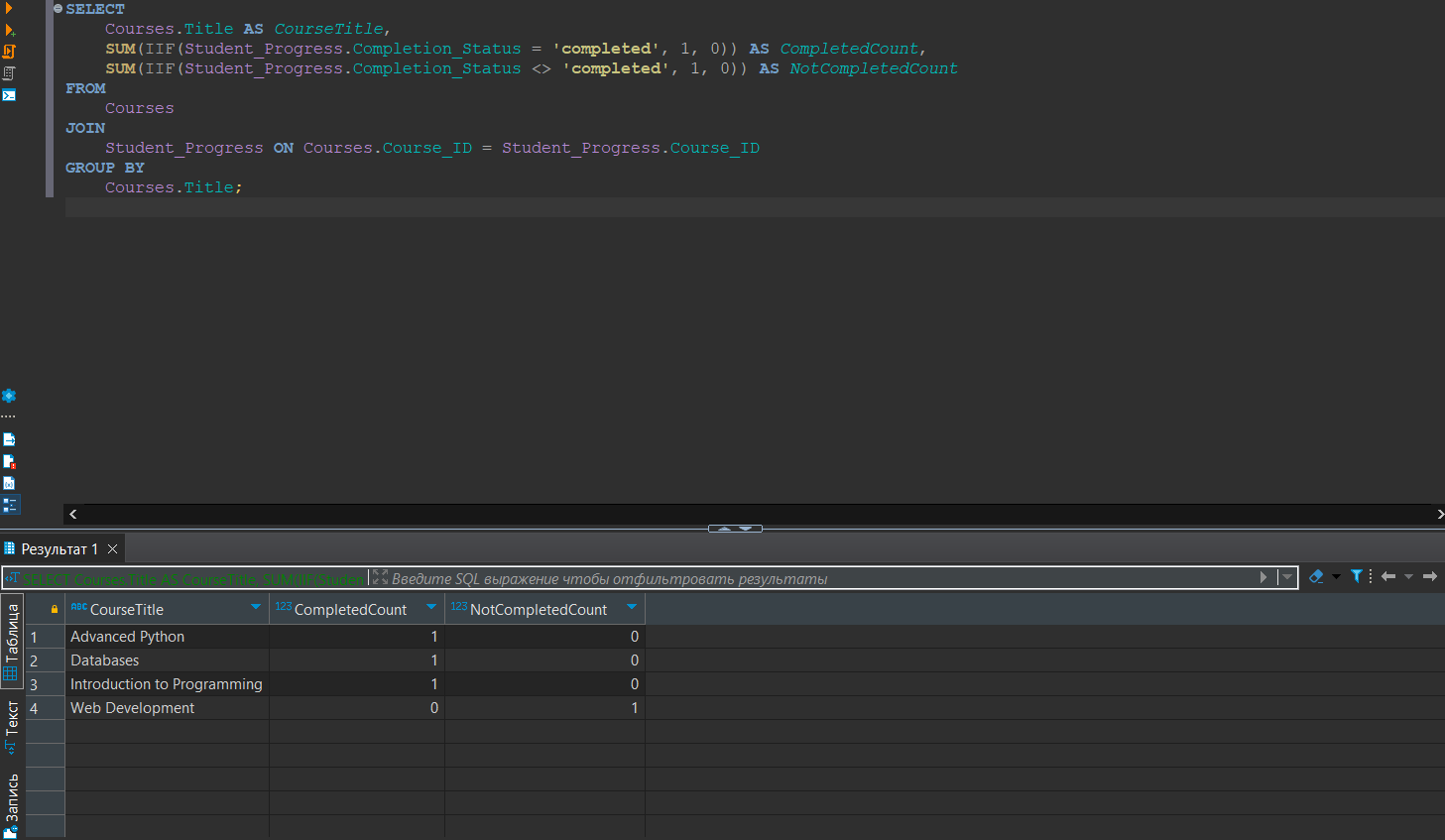
1. Вывести курсы, которые не имеют модулей (Запрос с внешним соединением и проверкой на наличие NULL):



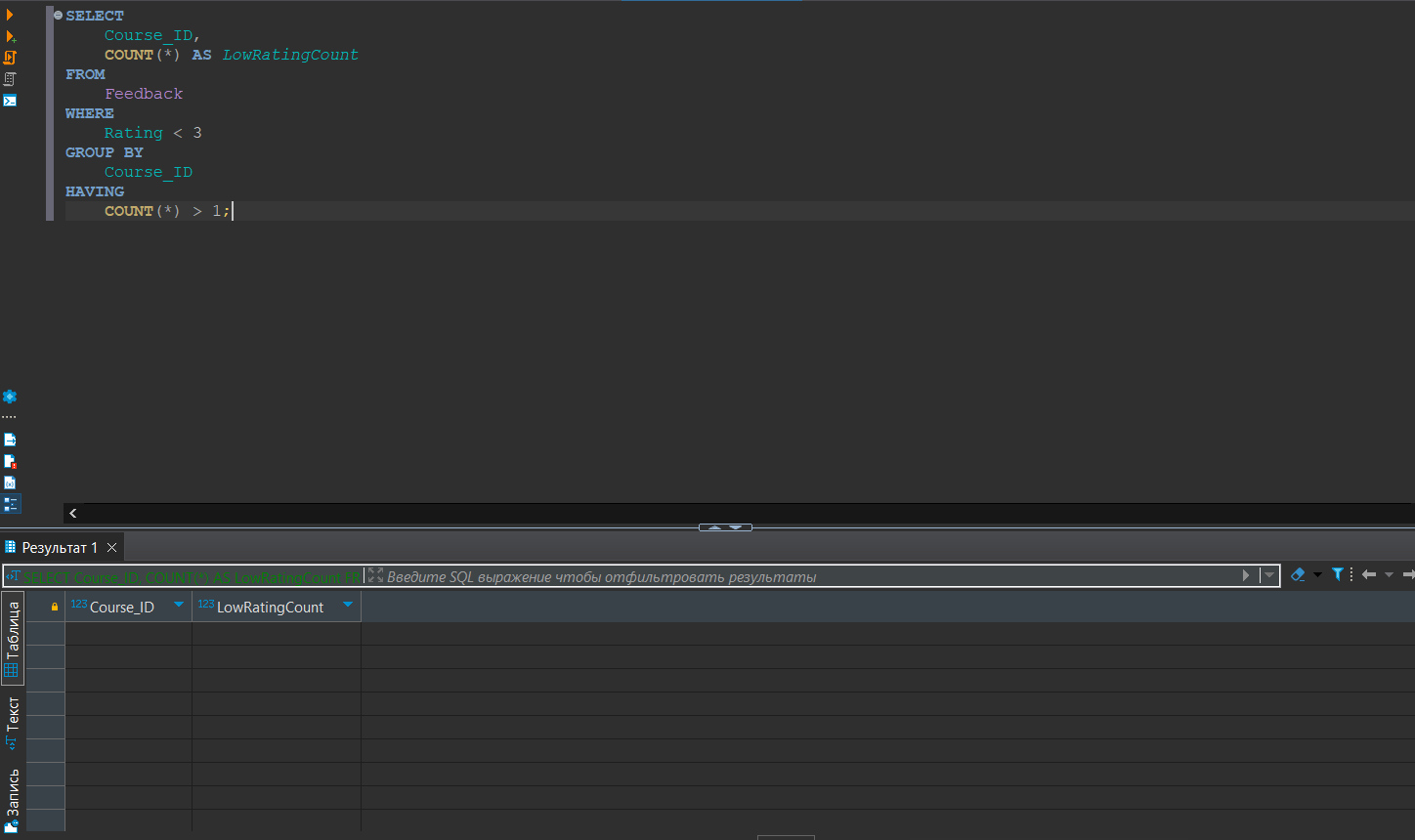
1. Вывести количество завершенных модулей для каждого курса (Запрос с агрегированием и выражением JOIN):



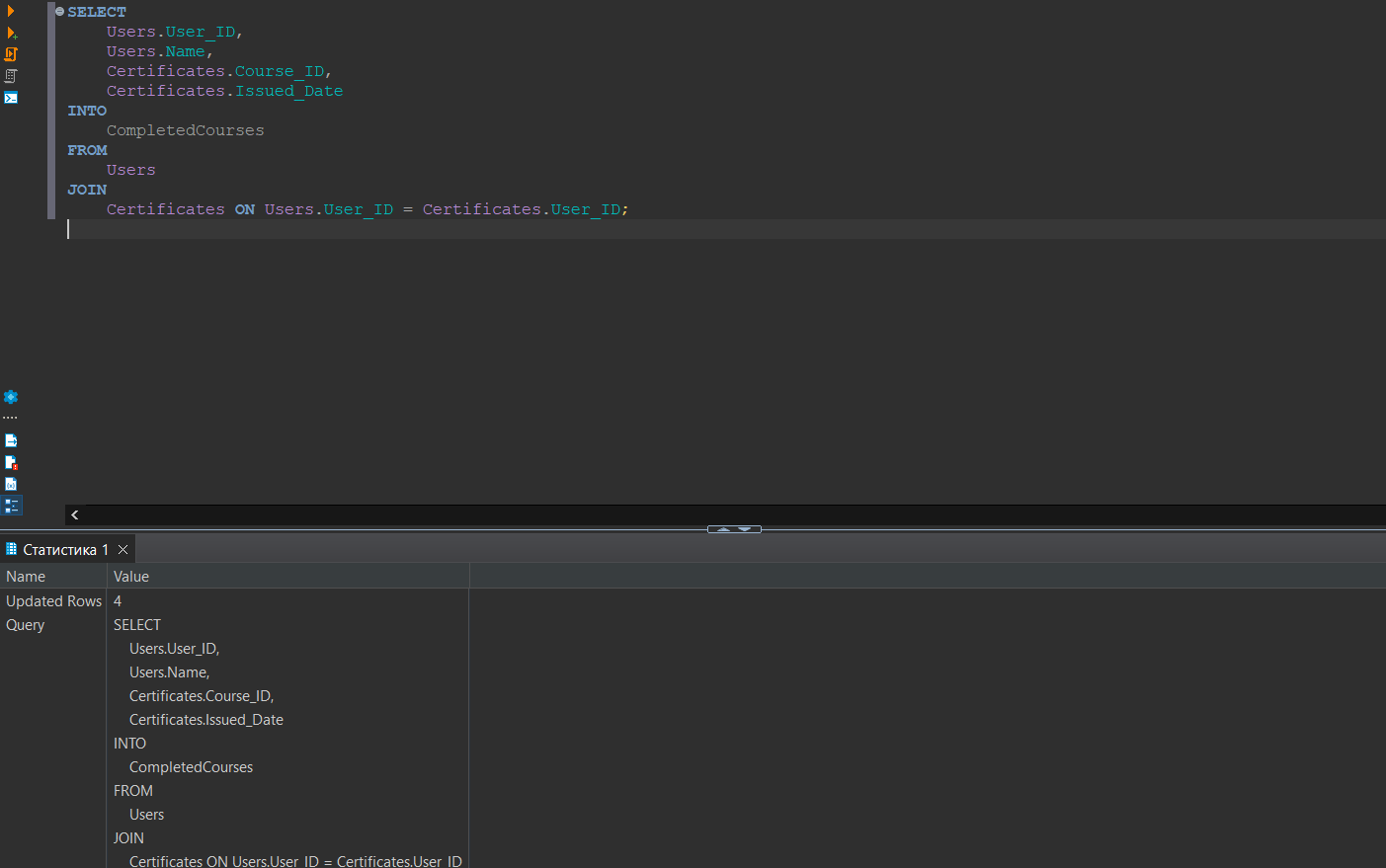
1. Вывести количество студентов по курсам с разбивкой на завершенные и незавершенные курсы (Запрос с CASE (IIF) и агрегированием):



1. Вывести курсы, которые имеют более одного отзыва с рейтингом ниже 3 (Запрос с HAVING и агрегированием):



1. Создать таблицу с информацией о пользователях, которые закончили курсы (Запрос SELECT INTO для подготовки выгрузки):

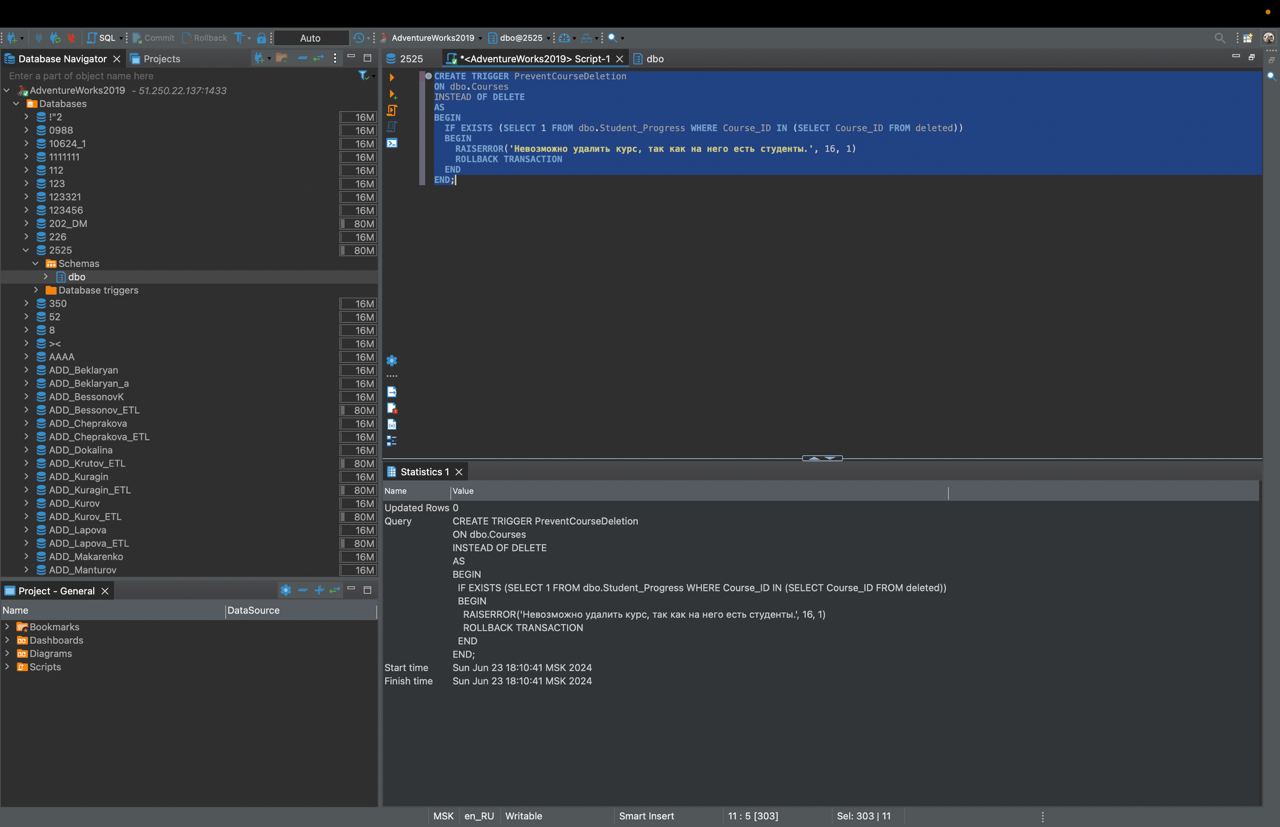


## **13. Создание хранимых процедур**

1. **Процедура 1: RegisterUser**
   * Эта процедура регистрирует нового пользователя в системе. Она принимает параметры для имени пользователя, электронной почты, пароля, номера телефона и роли. Если электронная почта уже существует, процедура возвращает ошибку.
   * CREATE PROCEDURE RegisterUser
   * @Name VARCHAR(100),
   * @Email VARCHAR(100),
   * @Password VARCHAR(255),
   * @PhoneNumber VARCHAR(16),
   * @Role VARCHAR(50)
   * AS
   * BEGIN
   * IF EXISTS (SELECT 1 FROM Users WHERE Email = @Email)
   * BEGIN
   * RAISERROR('Email already exists', 16, 1);
   * RETURN;
   * END
   * INSERT INTO Users (Name, Email, Password, PhoneNumber, Role)
   * VALUES (@Name, @Email, @Password, @PhoneNumber, @Role);
   * END;
2. **Процедура 2: UpdateCourse**
   * Эта процедура обновляет информацию о курсе. Она принимает параметры для ID курса, названия, описания и ID инструктора.
   * CREATE PROCEDURE UpdateCourse
   * @Course\_ID INT,
   * @Title VARCHAR(100),
   * @Description TEXT,
   * @Instructor\_ID INT
   * AS
   * BEGIN
   * UPDATE Courses
   * SET Title = @Title, Description = @Description, Instructor\_ID = @Instructor\_ID
   * WHERE Course\_ID = @Course\_ID;
   * END;
3. **Процедура 3: AddFeedback**
   * Эта процедура добавляет или обновляет обратную связь пользователя по курсу. Она принимает параметры для ID пользователя, ID курса, рейтинга и комментария. Если отзыв уже существует, он обновляется; если нет – создается новый.
   * CREATE PROCEDURE AddFeedback
   * @User\_ID INT,
   * @Course\_ID INT,
   * @Rating TINYINT,
   * @Comment TEXT
   * AS
   * BEGIN
   * DECLARE @ExistingFeedbackID INT;
   * SELECT @ExistingFeedbackID = Feedback\_ID
   * FROM Feedback
   * WHERE User\_ID = @User\_ID AND Course\_ID = @Course\_ID;
   * IF @ExistingFeedbackID IS NOT NULL
   * BEGIN
   * UPDATE Feedback
   * SET Rating = @Rating, Comment = @Comment, Date = GETDATE()
   * WHERE Feedback\_ID = @ExistingFeedbackID;
   * END
   * ELSE
   * BEGIN
   * INSERT INTO Feedback (User\_ID, Course\_ID, Rating, Comment, Date)
   * VALUES (@User\_ID, @Course\_ID, @Rating, @Comment, GETDATE());
   * END
   * UPDATE Courses
   * SET Description = Description -- Example of another update
   * WHERE Course\_ID = @Course\_ID;
   * END;
4. **Процедура 4: UpdateStudentProgress**
   * Эта процедура обновляет прогресс студента по курсу. Она принимает параметры для ID пользователя, ID курса, ID модуля и статуса завершения. Процедура включает обработку ошибок и транзакции для обеспечения целостности данных.
   * CREATE PROCEDURE UpdateStudentProgress
   * @User\_ID INT,
   * @Course\_ID INT,
   * @Module\_ID INT,
   * @Completion\_Status VARCHAR(50)
   * AS
   * BEGIN
   * BEGIN TRANSACTION;
   * BEGIN TRY
   * DECLARE @Completion\_Date DATE = NULL;
   * IF @Completion\_Status = 'Completed'
   * BEGIN
   * SET @Completion\_Date = GETDATE();
   * END
   * UPDATE Student\_Progress
   * SET Completion\_Status = @Completion\_Status, Completion\_Date = @Completion\_Date
   * WHERE User\_ID = @User\_ID AND Course\_ID = @Course\_ID AND Module\_ID = @Module\_ID;
   * IF @@ROWCOUNT = 0
   * BEGIN
   * INSERT INTO Student\_Progress (User\_ID, Course\_ID, Module\_ID, Completion\_Status, Completion\_Date)
   * VALUES (@User\_ID, @Course\_ID, @Module\_ID, @Completion\_Status, @Completion\_Date);
   * END
   * COMMIT TRANSACTION;
   * END TRY
   * BEGIN CATCH
   * ROLLBACK TRANSACTION;
   * DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();
   * RAISERROR (@ErrorMessage, 16, 1);
   * END CATCH
   * END;
5. **Процедура 5: DeleteCourse**
   * Эта процедура удаляет курс и все связанные с ним модули и видео-лекции. Процедура включает обработку ошибок и транзакции для обеспечения целостности данных.
   * CREATE PROCEDURE DeleteCourse
   * @Course\_ID INT
   * AS
   * BEGIN
   * BEGIN TRANSACTION;
   * BEGIN TRY
   * DELETE FROM Video\_Lectures
   * WHERE Module\_ID IN (SELECT Module\_ID FROM Course\_Modules WHERE Course\_ID = @Course\_ID);
   * DELETE FROM Course\_Modules
   * WHERE Course\_ID = @Course\_ID;
   * DELETE FROM Courses
   * WHERE Course\_ID = @Course\_ID;
   * COMMIT TRANSACTION;
   * END TRY
   * BEGIN CATCH
   * ROLLBACK TRANSACTION;
   * DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();
   * RAISERROR (@ErrorMessage, 16, 1);
   * END CATCH
   * END;

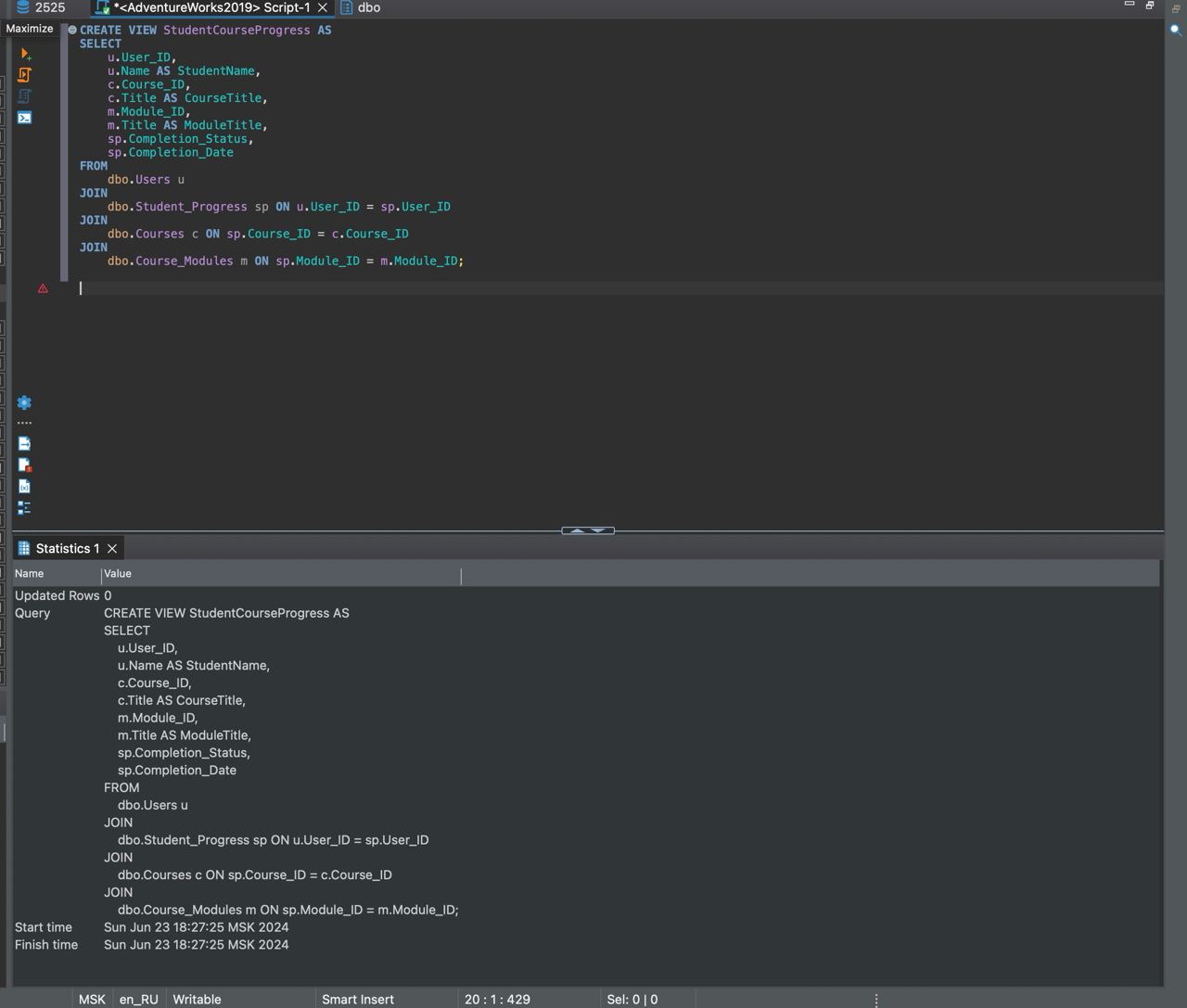
## **14. Создание триггера**

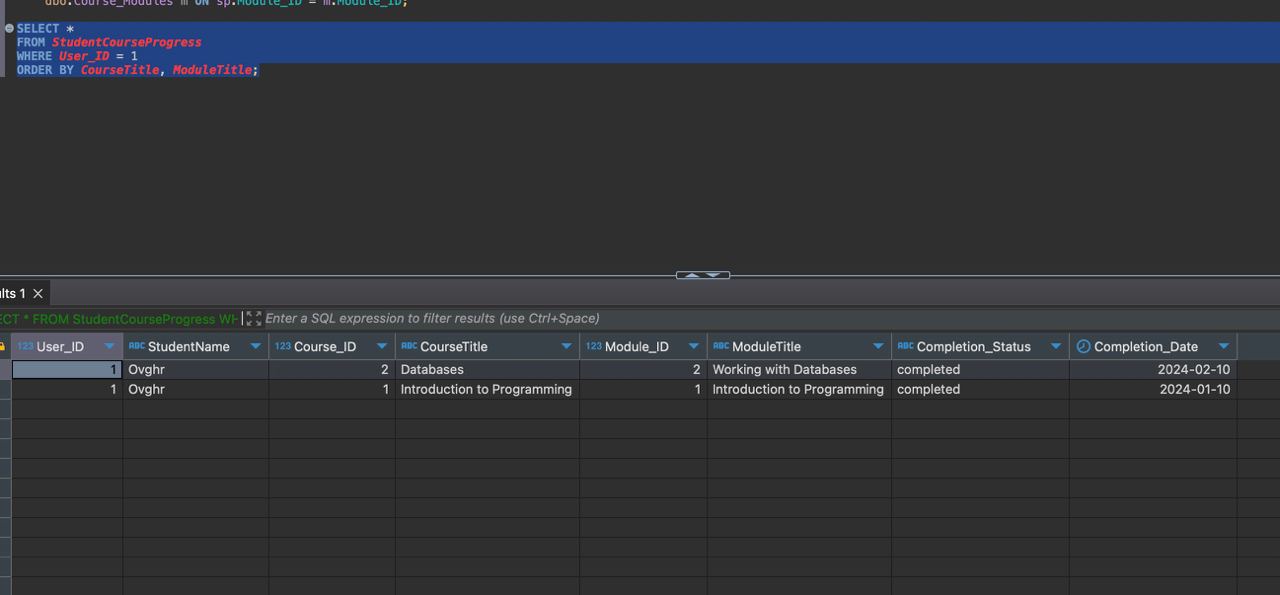
1. **Триггер**
   * Запрещает удаление курса, если на нем есть студенты

****

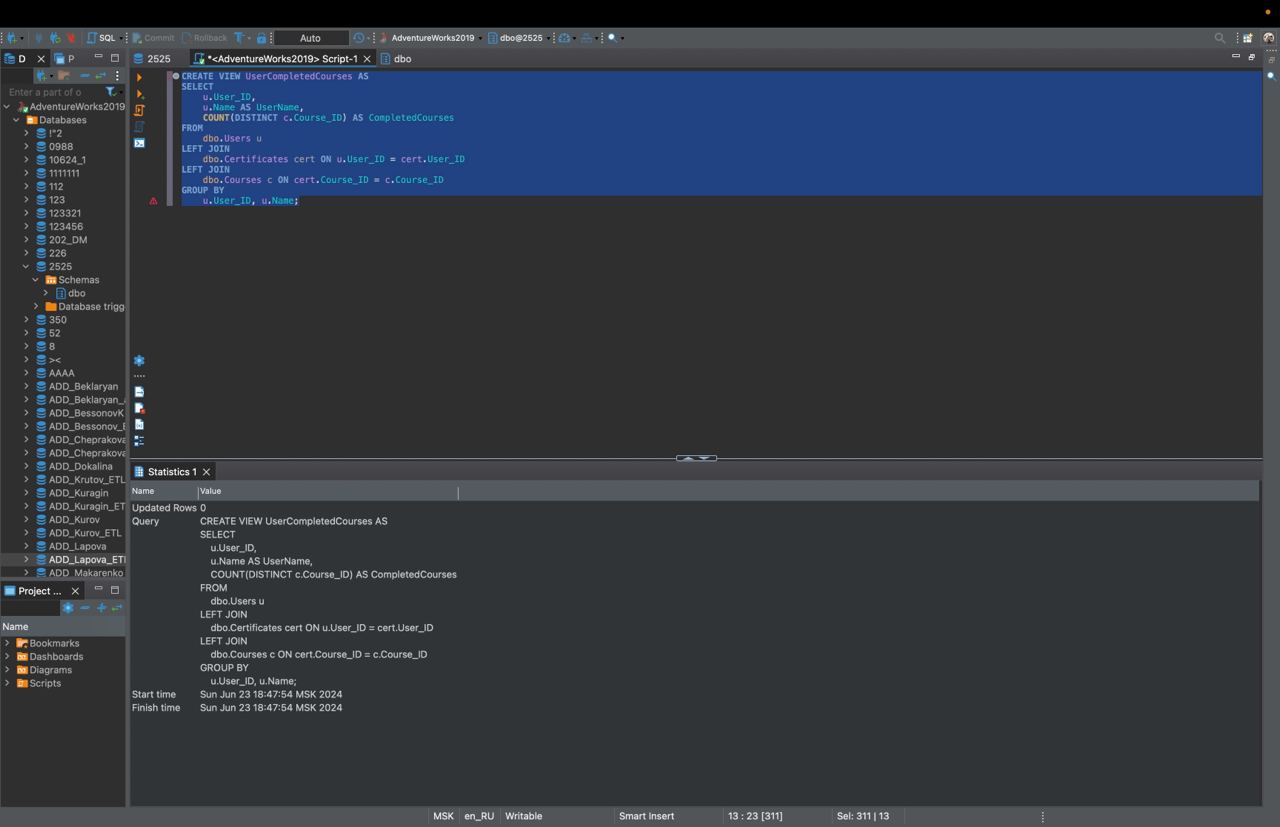
## **15. Создание представлений**

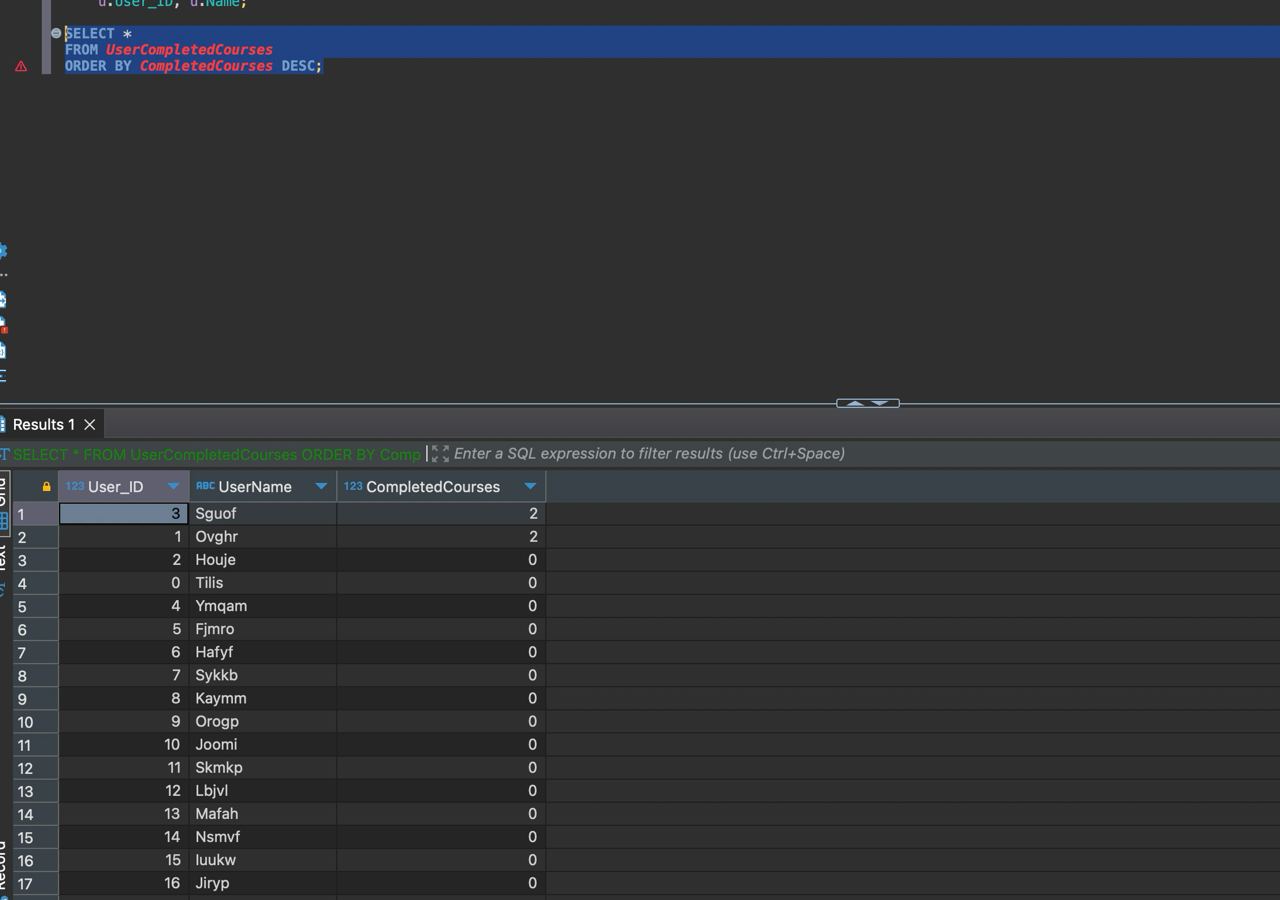
1. **Представление 1:**
   * Это представление показывает прогресс каждого студента в каждом модуле каждого курса





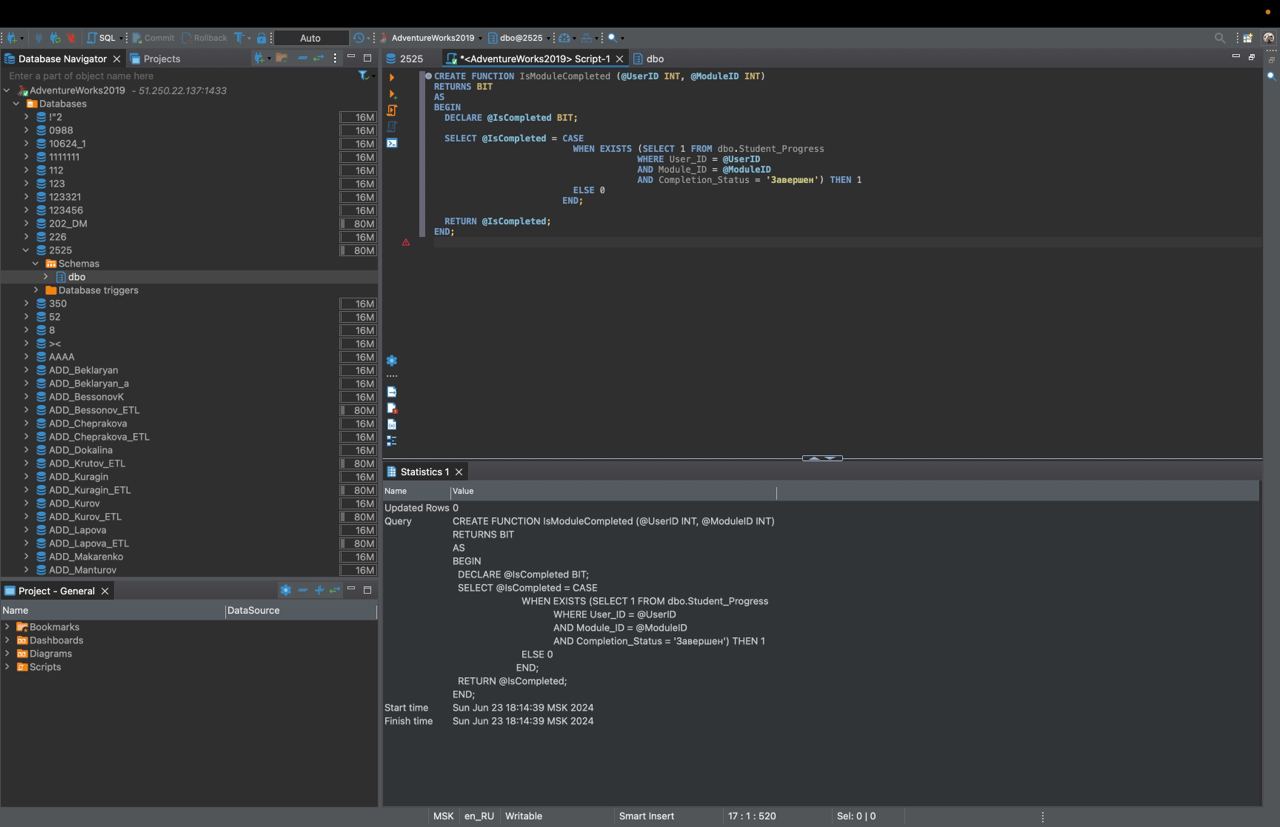
1. **Представление 2:**
   * Это представление показывает список пользователей и количество курсов, которые они успешно завершили

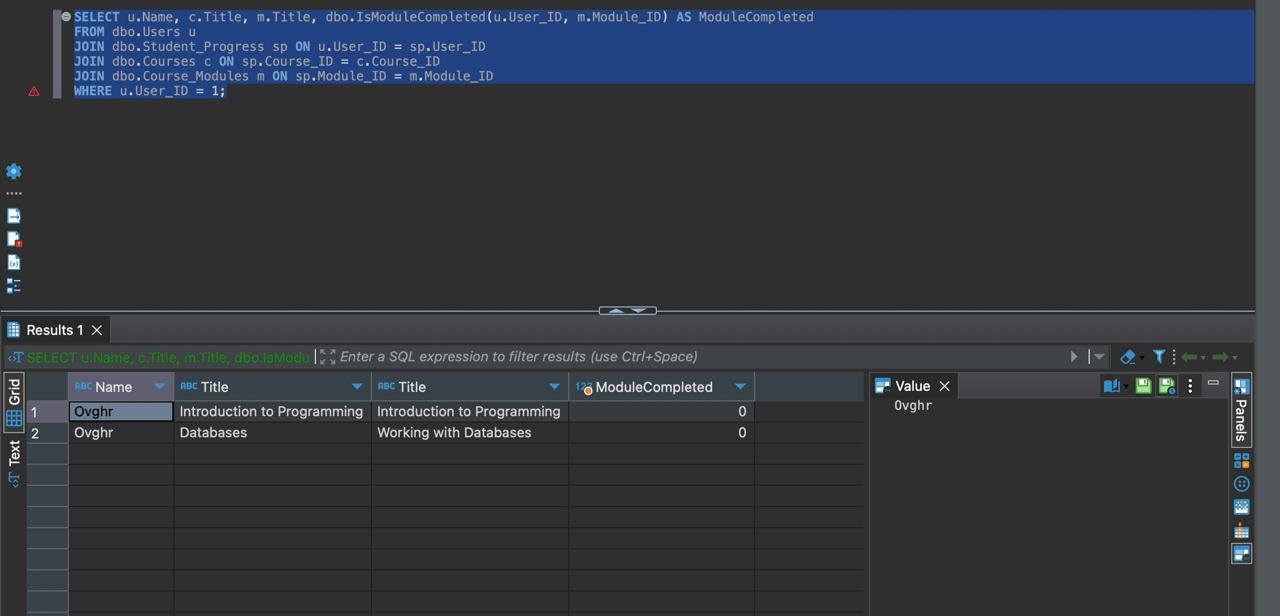
****

****

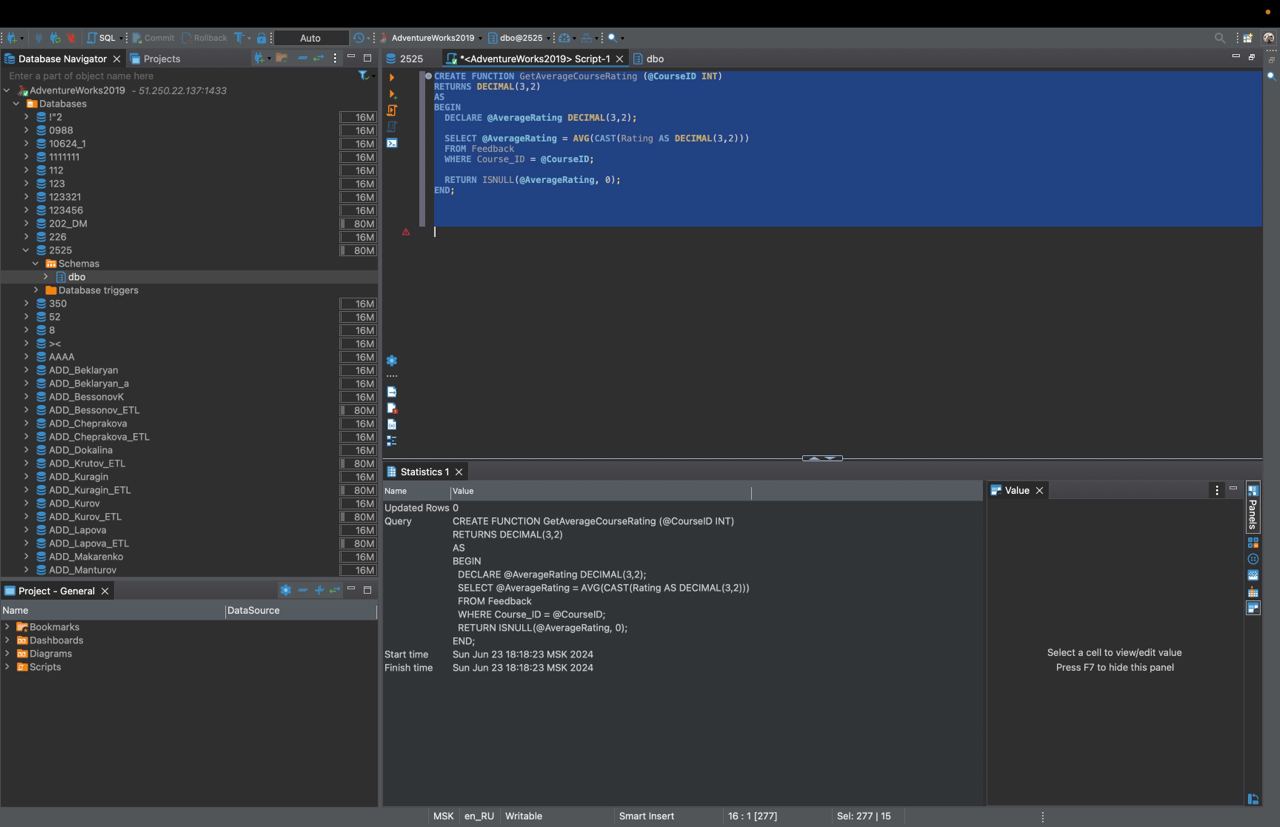
## **16. Создание функций**

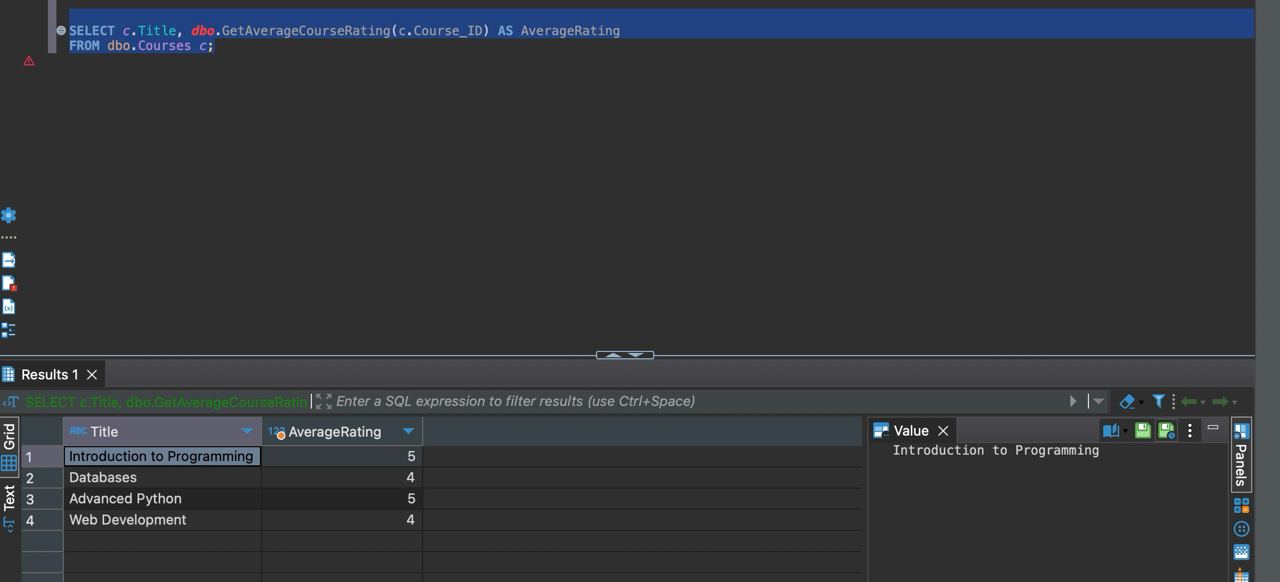
1. **Функция 1 - Проверка прохождения модуля пользователем**





1. **Функция 2 - Получение средней оценки курса**

****

****