**«МІЖНАРОДНИЙ НАУКОВО-ТЕХНІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ АКАДЕМІКА ЮРІЯ БУГАЯ»**

Кафедра інженерії програмного забезпечення  
Освітній ступінь – «Бакалавр»

**Курсовий проект**

**З дисципліни:  
 «Бази даних»**

Виконав студент 3-го курсу групи І-11

Овдієнко Ілля Іванович

Перевірив:

Липко Олександр Сергійович

**Київ 2024**

**Зміст**

1. Вибір та аналіз предметної області:
   * Визначення ключових процесів та даних, які будуть оброблятися в системі.
   * Аналіз унікальних вимог до проєкту.
2. Проектування бази даних:

* Розробка концептуальної моделі БД
  + Розробка логічної моделі БД
  + Розробка фізичної моделі БД

1. Реалізація бази даних:

* Опис БД за домомогою TypeOrm
* Таблиці
* Звʼязки
* Індекси
* Створення міграції для ініціалізації БД
* Створення скриптів для наповнення БД тестовими даними
* Написання типових складних запитів для вашої предметної області, які продемонструють оптимальність роботи БД
* TypeOrm/pure SQL, TypeOrm
* Explain query

1. Розгортання бази даних:

* Вибір і налаштування середовища розгортання
* Розгортання БД і налаштування параметрів з'єднання

1. Налаштування відмовостійкості:

* Реалізація реплікації для забезпечення доступності даних.
* Розробка стратегії бекапів

**Тема курсового проекту:** Проектування та розробка бази даних для обраної предметної області

**Мета курсового проекту:**

Розробити і реалізувати комплексну базу даних для специфічної предметної області, обраної студентом або визначеної викладачем, з метою підвищення ефективності обробки, зберігання та аналізу даних відповідно до вимог цієї сфери. Проект передбачає проведення глибокого аналізу вимог до системи, розробку логічної та фізичної структури бази даних, її реалізацію з використанням обраної системи управління базами даних (СУБД), та розгортання з налаштуванням механізмів забезпечення відмовостійкості та безпеки даних. Цей проєкт спрямований на розвиток навичок проектування баз даних, програмування, адміністрування БД, а також забезпечення безпеки і надійності інформаційних систем.

**1.Вибір та аналіз предметної області**

Тема моєї БД: «Школа»

**Процеси**:

* Додавання нових учнів
* Ведення журналу успішності
* Додавання вчителів та класів
* Додавання предметів

**Дані**:

* Учні: ім'я, прізвище, клас.
* Класи: номер, учитель.
* Предмети: назва.
* Оцінки: предмет, оцінка, учень.

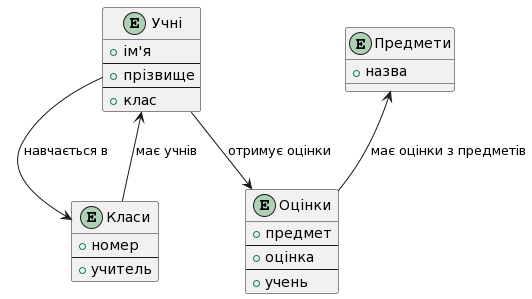
Аналіз унікальних вимог:

**Система моніторингу успішності:**

* Збір даних про успішність учнів (оцінки).

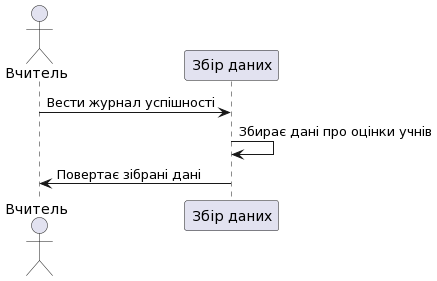
**2.Проектування бази даних**

Концептуальна модель:



ER-діаграма зв’язків між сутностями БД.

Джерело: <https://www.lucidchart.com>



ER-діаграма унікальних вимог БД.

Джерело: <https://www.lucidchart.com>

Розробка логічної моделі БД:

Для реалізації інфраструктури бази даних у проекті я використовую Google Cloud Platform (GCP) для розгортання віддаленого сервера.

Цей підхід забезпечує високу доступність і надійність інфраструктури завдяки використанню інфраструктури в хмарі. На цьому віддаленому сервері запускається база даних PostgreSQL, яка є вільною та відкритою системою управління базами даних з великою спільнотою користувачів та регулярними оновленнями.

Для взаємодії з цією базою даних я використовую TypeORM - ORM (Object-Relational Mapping) для TypeScript та JavaScript. TypeORM дозволяє працювати з базою даних, використовуючи об'єкти та зв'язки між ними, замість написання складних SQL-запитів.

Крім того, для адміністрування та керування базою даних PostgreSQL я використовую DBEAVER - універсальний інструмент, який надає зручний інтерфейс для виконання різноманітних операцій, таких як виконання SQL-запитів, перегляд структури таблиць, імпорт та експорт даних і багато іншого.

Таким чином, я створюю надійну та зручну інфраструктуру бази даних для мого проекту, яка забезпечить ефективне управління даними та забезпечить швидкий доступ до них.

Розробка фізичної моделі БД:

Опис фізичної моделі бази даних (БД):

1. Таблиця "student":

- Поля:

- id (SERIAL PRIMARY KEY): унікальний ідентифікатор студента.

- firstName (VARCHAR(255) NOT NULL): ім'я студента.

- lastName (VARCHAR(255) NOT NULL): прізвище студента.

- classId (INTEGER): ідентифікатор класу, до якого належить студент. Зовнішній ключ, що посилається на поле "id" таблиці "class".

2. Таблиця "class":

- Поля:

- id (SERIAL PRIMARY KEY): унікальний ідентифікатор класу.

- number (INTEGER NOT NULL): номер класу.

- teacher (VARCHAR(255) NOT NULL): вчитель класу.

3. Таблиця "subject":

- Поля:

- id (SERIAL PRIMARY KEY): унікальний ідентифікатор предмету.

- name (VARCHAR(255) NOT NULL): назва предмету.

- description (VARCHAR(255) NOT NULL): опис предмету.

4. Таблиця "grade":

- Поля:

- id (SERIAL PRIMARY KEY): унікальний ідентифікатор оцінки.

- subjectId (INTEGER NOT NULL): ідентифікатор предмету. Зовнішній ключ, що посилається на поле "id" таблиці "subject".

- studentId (INTEGER NOT NULL): ідентифікатор студента. Зовнішній ключ, що посилається на поле "id" таблиці "student".

- grade (INTEGER NOT NULL): оцінка студента за предмет.

Це опис структури таблиць та їх взаємозв'язків у базі даних "школа".

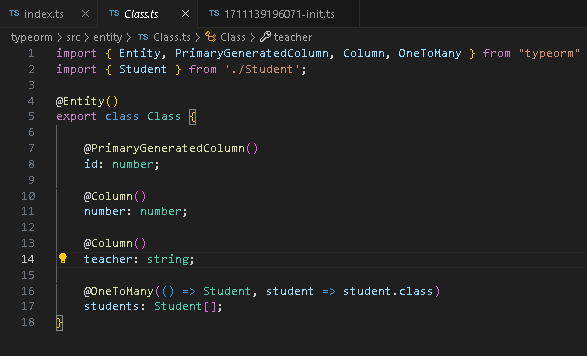
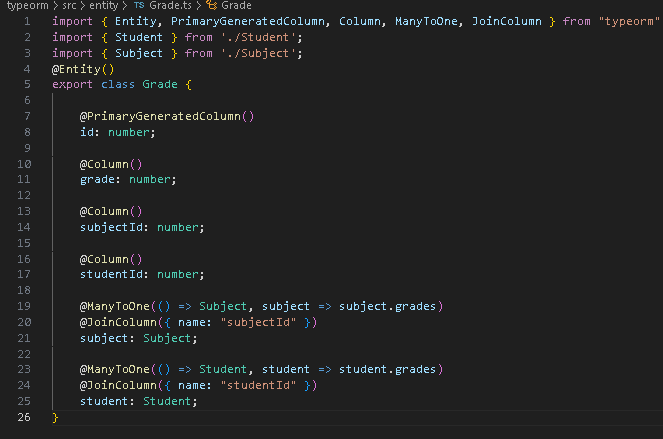
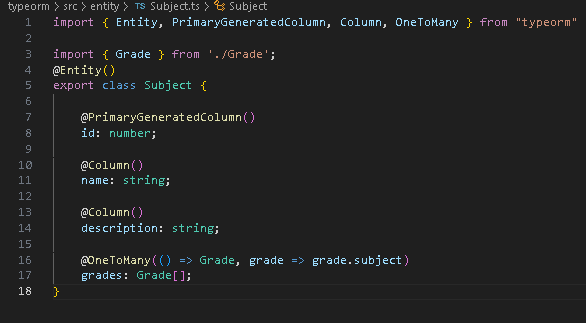
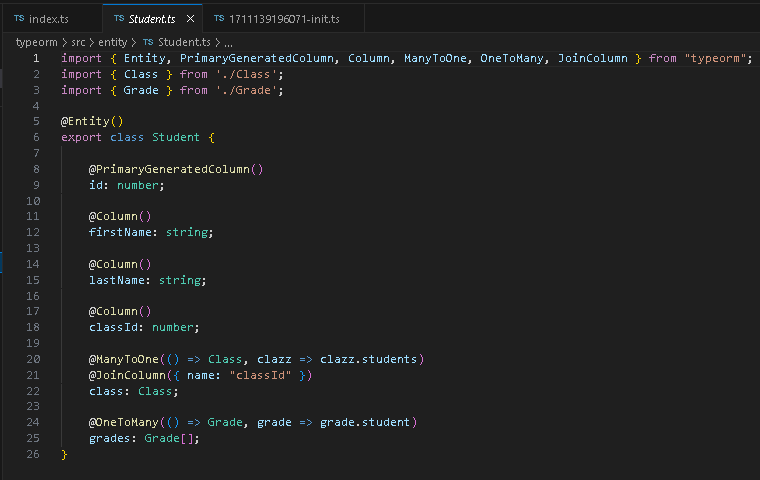
**3. Реалізація бази даних:**

**1.**Зв’язки та сутності

Сутності:

* Student
* Атрибути
* id
* First name(string)
* Last name(string)
* class(integer, references Класи.id)
* Class
* Атрибути
* id
* number(integer)
* teacher(string)
* Subject
* Атрибути
* id
* name(string)
* description(string)
* Grade
* Атрибути
* id
* Subject(integer, references Предмети.id)
* Grade(integer)
* Student(integer, references Учні.id)

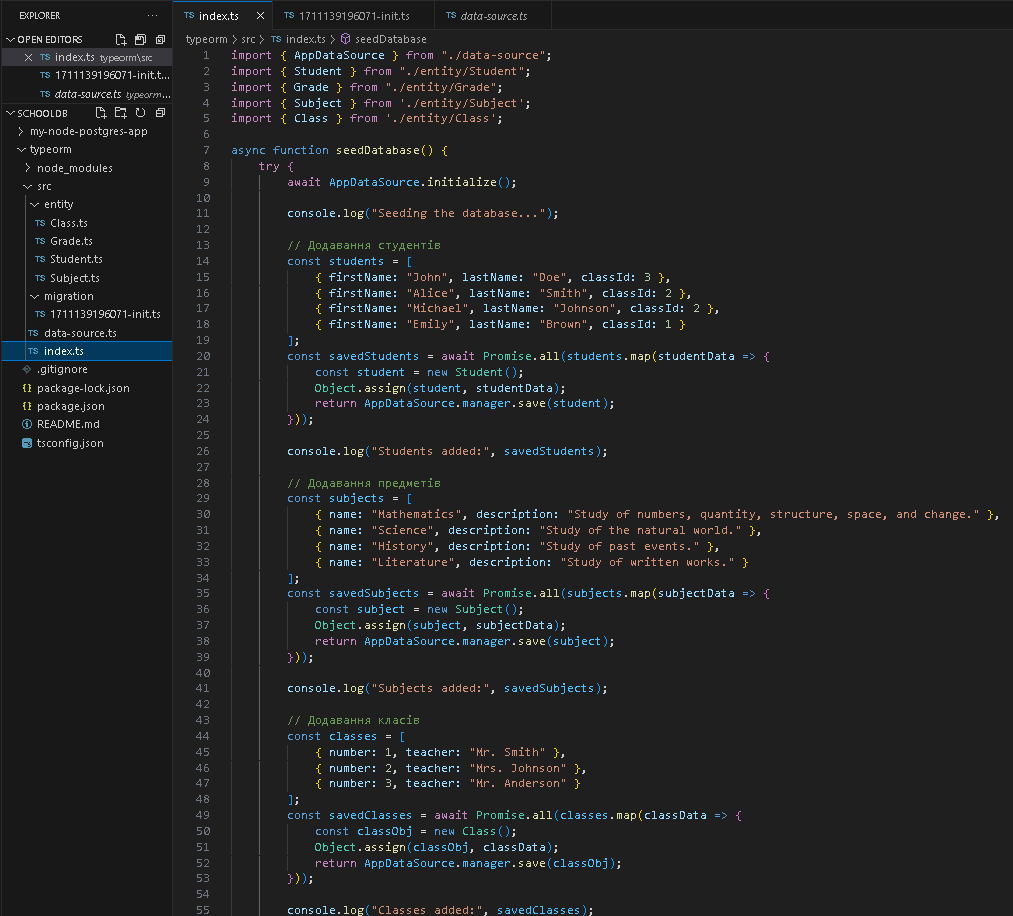
Зв’язки:

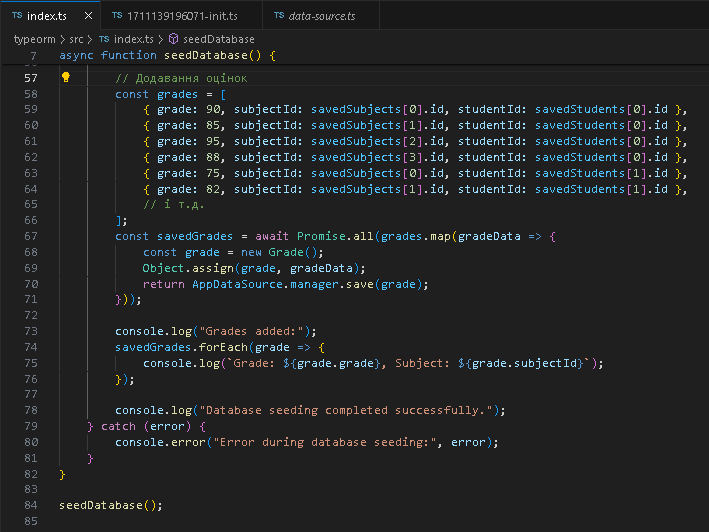
* ManyToOne:
* Class →Student
* Grade→Subject
* Student→Grade
* OneToMany
* Student→Subject(через таблицю Grade)

**2.** Індекси

* Файл Index.ts виконує ініціалізацію бази даних та наповнення її даними за допомогою ORM:

1. **Імпорт залежностей**: Код імпортує необхідні залежності, такі як **AppDataSource**, **Student**, **Grade**, **Subject**, **Class**. Ці залежності, представляють сутності (entity) та джерело даних для взаємодії з базою даних.
2. **Функція seedDatabase**: Ця асинхронна функція виконує ініціалізацію джерела даних, додає студентів, предмети, класи та оцінки до бази даних.
3. **Додавання студентів**: Створюється масив об'єктів **students**, який містить дані про студентів (ім'я, прізвище, ідентифікатор класу), після чого кожен студент додається до бази даних за допомогою методу **save** об'єкта **AppDataSource.manager**.
4. **Додавання предметів**: Аналогічно, створюється масив об'єктів **subjects**, який містить дані про предмети (назва, опис), і кожен предмет також додається до бази даних.
5. **Додавання класів**: Подібно до студентів та предметів, створюється масив об'єктів **classes**, який містить дані про класи (номер, вчитель), і кожен клас додається до бази даних.
6. **Додавання оцінок**: Створюється масив об'єктів **grades**, який містить дані про оцінки (оцінка, ідентифікатор предмету, ідентифікатор студента), і кожна оцінка додається до бази даних.
7. **Вивід результатів**: У кінці кожен етап додавання даних виводиться у консоль, щоб підтвердити успішне додавання. Крім того, якщо виникає помилка під час процесу, вона також виводиться у консоль.
8. **Завершення роботи**: У кінці функція виводить повідомлення про успішне завершення процесу заповнення бази даних.



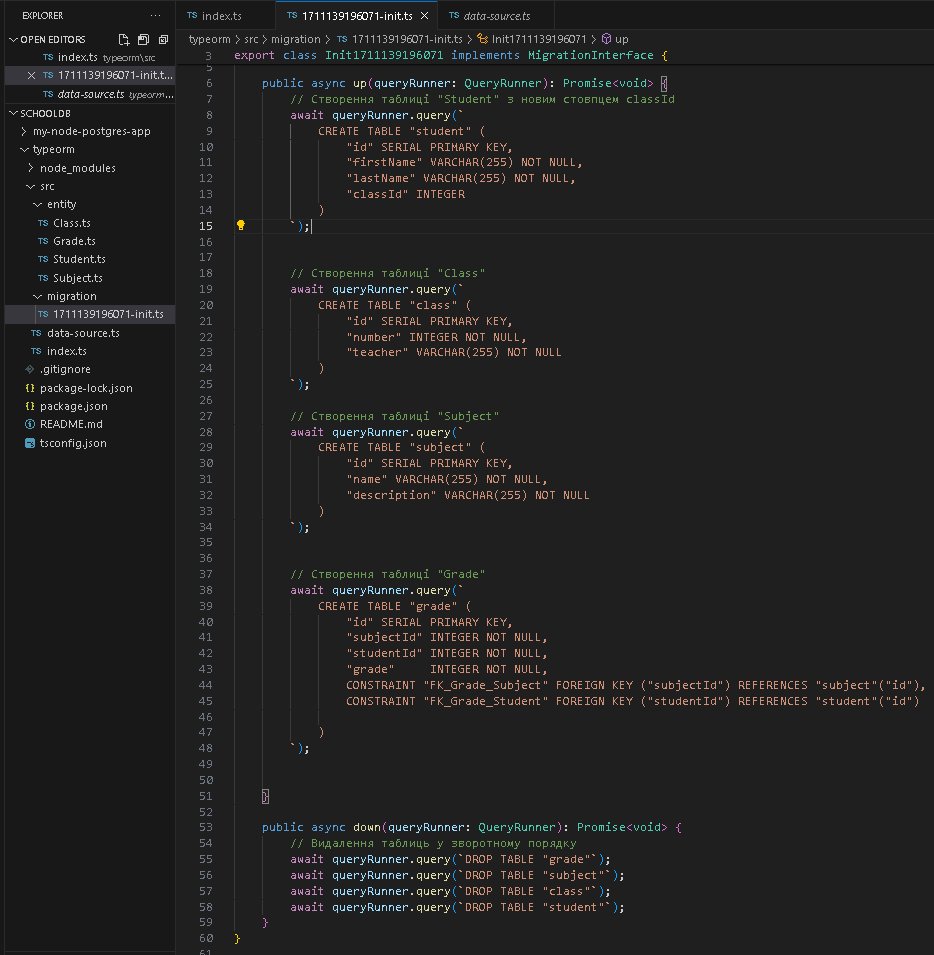


Отже файл ініціалізує базу даних та наповнює її даними про студентів, предмети, класи та оцінки, які після цього можуть бути використані для подальших операцій з даними.

**3.**Таблиці(міграції)

* Файл Init1711139196071.ts представляє міграцію бази даних, яка використовує TypeORM. Давайте розглянемо кожен крок:

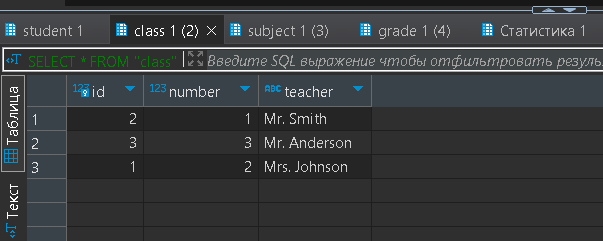
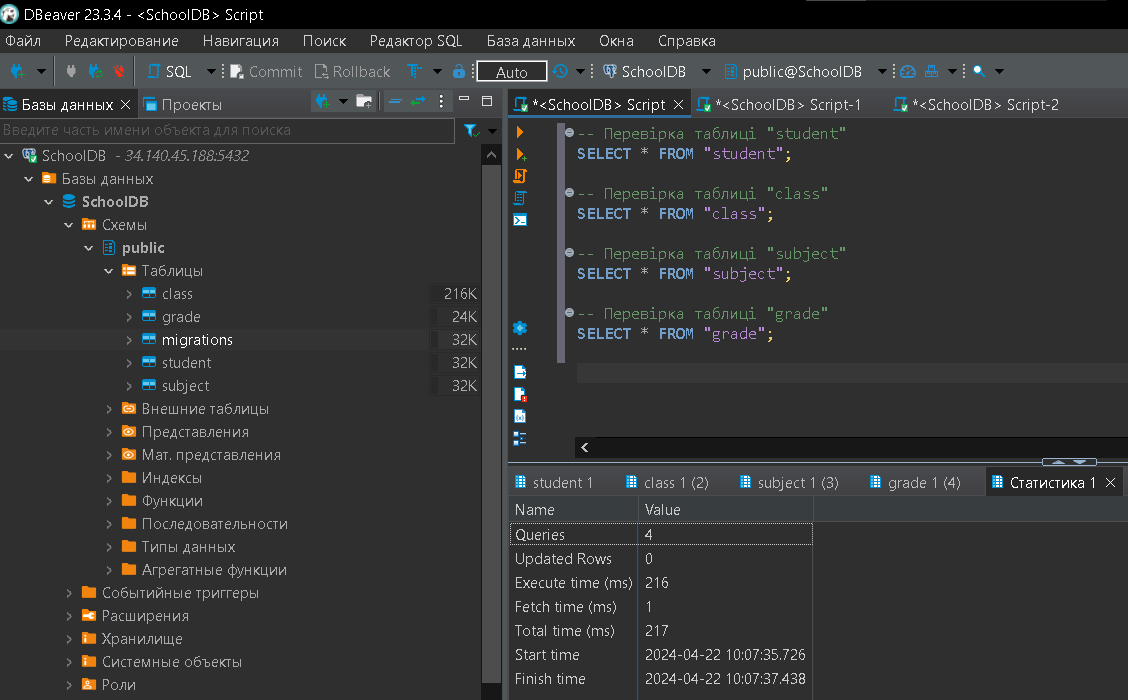
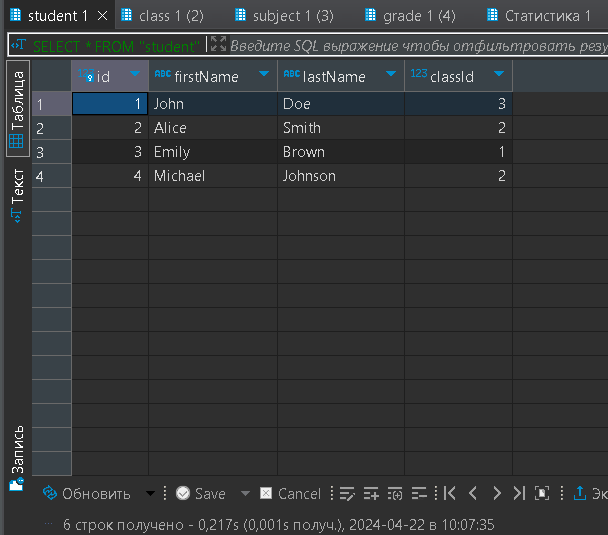
1. **Створення таблиць**: В методі **up** міграції виконується створення таблиць у базі даних. Кожен блок **await queryRunner.query()** виконує SQL-запит для створення окремої таблиці. У вашому випадку створюються таблиці "Student", "Class", "Subject" і "Grade".
2. **Додавання стовпців**: Кожна таблиця містить свої власні стовпці, які визначені у відповідних SQL-запитах. Наприклад, у таблиці "Student" створюється стовпець "classId", який вказує на клас, до якого належить студент.
3. **Зовнішні ключі (Foreign keys)**: У таблиці "Grade" визначаються зовнішні ключі (**FOREIGN KEY**) для зв'язку з таблицями "Subject" і "Student". Це дозволяє встановити зв'язок між оцінками та відповідними предметами та студентами.
4. **Метод down**: Метод **down** відповідає за відміну міграції. У даному випадку, він виконує видалення таблиць у зворотному порядку від того, як вони були створені. Це робить міграцію зворотно сумісною, тобто забезпечує можливість відновлення бази даних до попереднього стану.

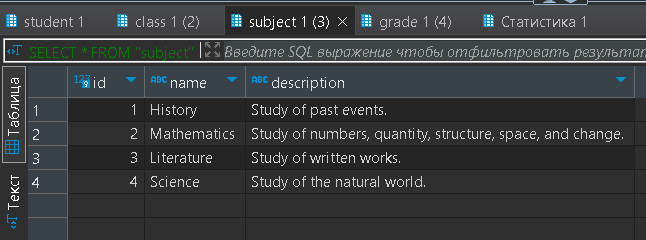
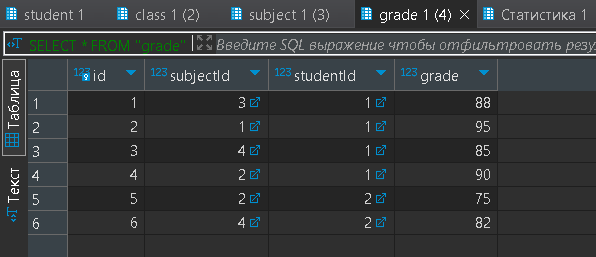


Отже файл виконує створення таблиць та відповідних зв'язків між ними у базі даних за допомогою SQL-запитів. Це дозволяє структурувати та організувати дані у відповідності до моделі даних вашого додатку.

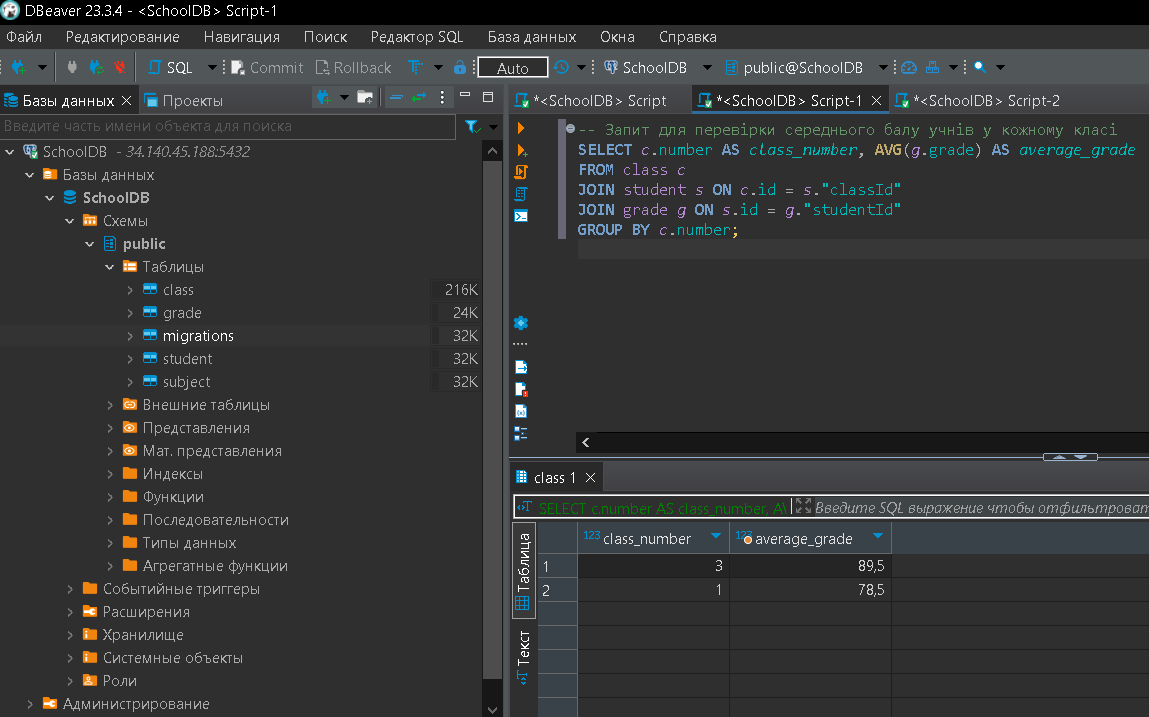
**4.** Демонстрація роботи БД:

Для демонстрації працездатності моєї БД я використаю 3 SQL запити.

****1)Цей SQL-скрипт виведе всі записи з кожної таблиці бази даних, щоб перевірити, чи правильно додані дані та чи відповідають вони очікуванням.

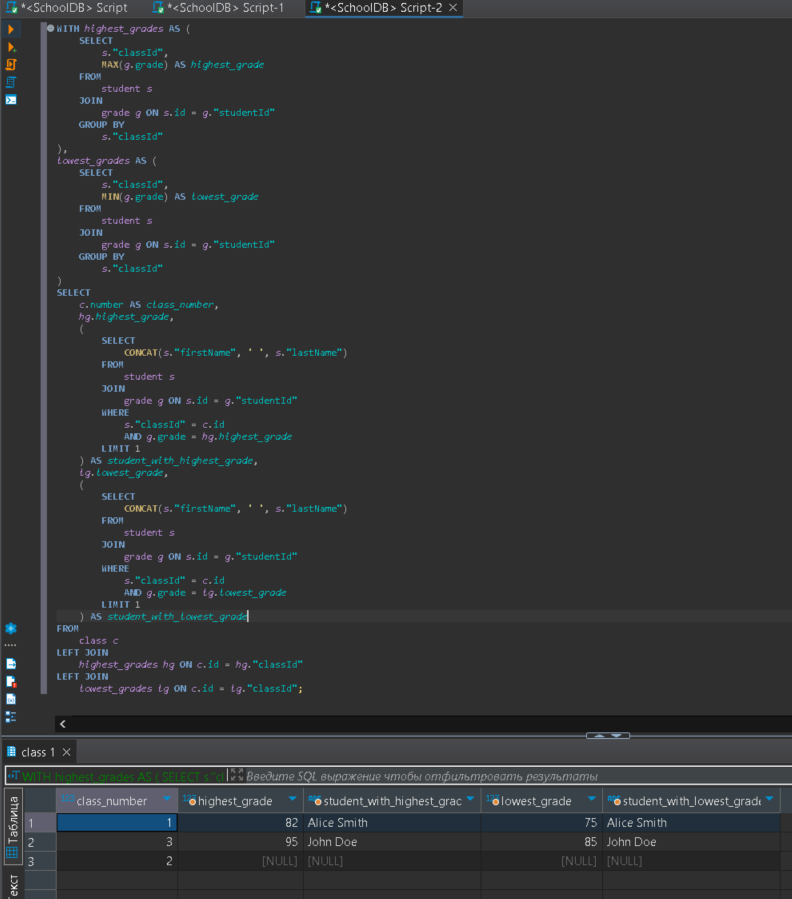
* ****Начало формы

2)Цей запит об'єднує дані з таблиць "class", "student" та "grade". Він обчислює середній бал учнів у кожному класі, групуючи дані за номером класу. Результати цього запиту допоможуть переконатися в правильності та повноті ваших даних, а також в тому, що зв'язки між таблицями працюють належним чином.



3) Цей SQL-запит виконує наступні дії:

1. Знаходить найвищий бал у кожному класі (highest\_grades) та найнижчий бал у кожному класі (lowest\_grades). Для цього використовуються спільні табличні вирази (CTE), які обчислюють максимальний та мінімальний бали за кожен клас на основі таблиць "student" та "grade". Поля "classId" використовуються для групування даних за класами.
2. Основний запит об'єднує результати CTE з таблицею "class", щоб вивести номер класу, найвищий та найнижчий бали у кожному класі, а також ім'я та прізвище учня з найвищим та найнижчим балом.
3. Для кожного класу запит використовує підзапити для знаходження учнів з найвищим та найнижчим балами. Підзапити вибирають ім'я та прізвище учня на основі таблиць "student" та "grade", де бал співпадає з максимальним чи мінімальним балом, відповідно.

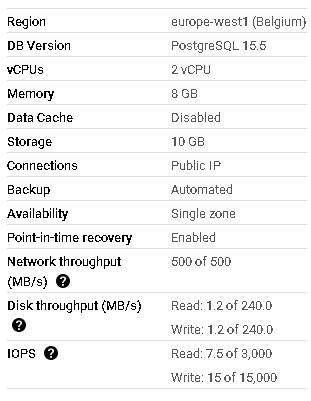


Отже, запит повертає інформацію про кожен клас, його найвищий та найнижчий бали, а також ім'я та прізвище учня з найвищим та найнижчим балами в кожному класі.

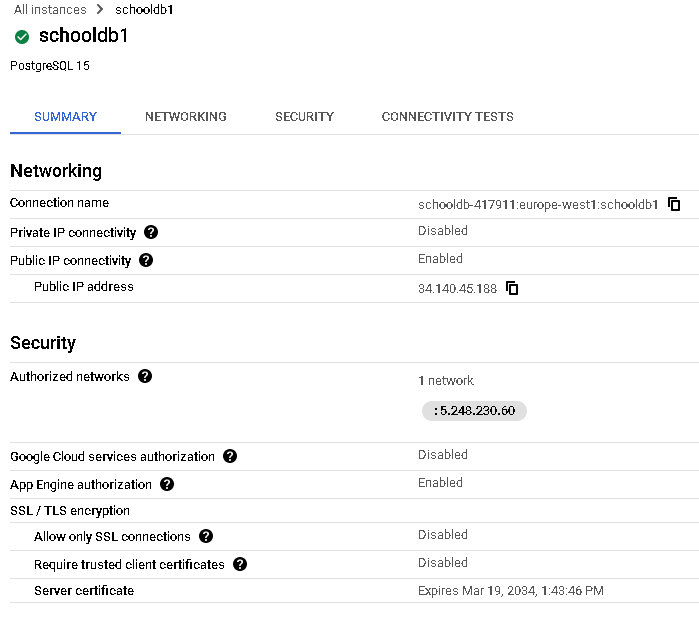
Начало формы

**4.Розгортання бази даних:**

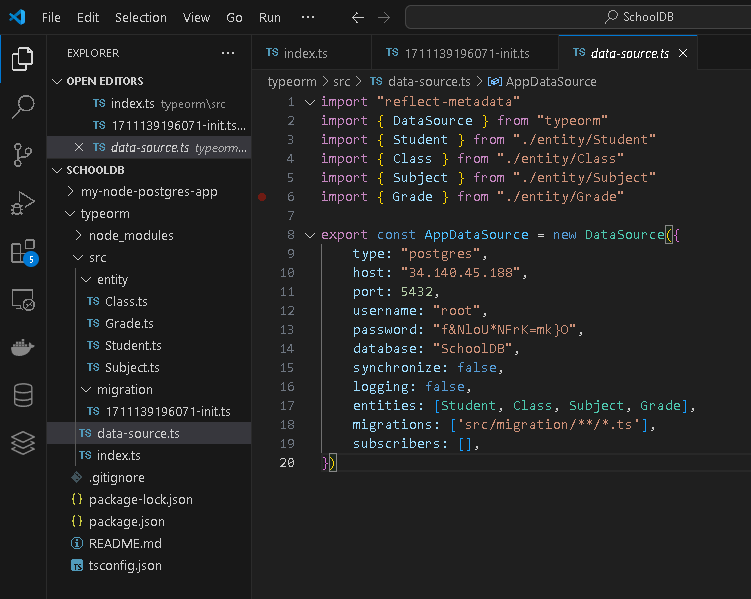
Розгортатися БД буде в Google Cloud Platform (GCP).



Налаштування серверу GCP

Налаштування підключення:

Підключення IP на GCP

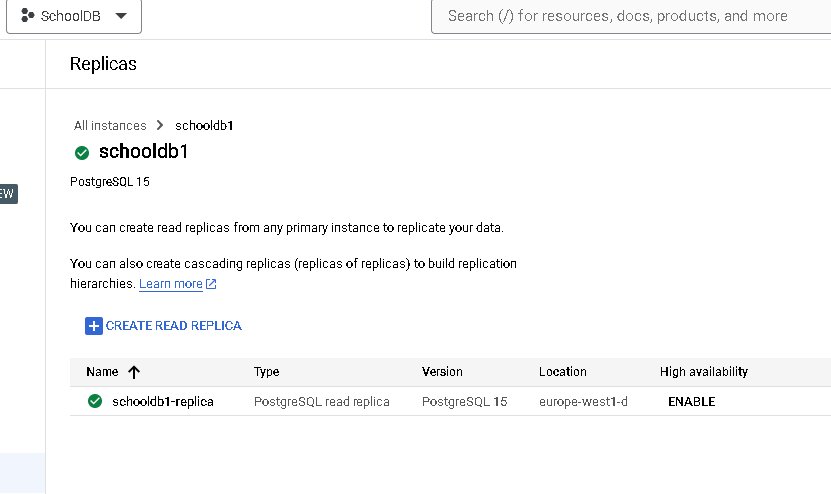


Зображення коду підключення до БД.

**5.Налаштування відмовостійкості:**

1)Створення репліки БД

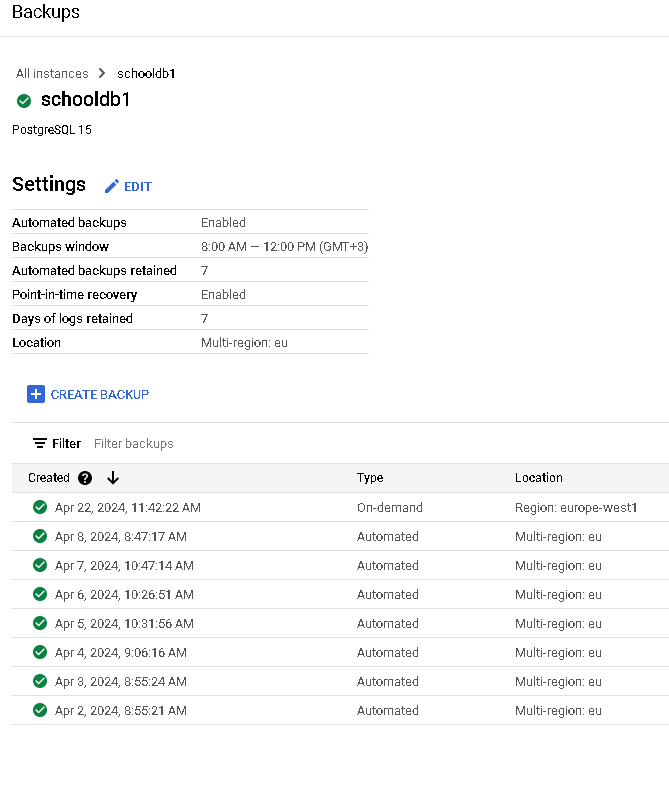
Створення репліки бази даних (DB replica) в Google Cloud Platform (GCP) є важливою практикою для забезпечення надійності та доступності бази даних. Репліка - це копія основної бази даних, яка автоматично оновлюється, коли змінюється основна база даних.



Загалом, створення репліки бази даних на GCP допомагає забезпечити високу доступність та надійність інфраструктури, зменшуючи ризик втрати даних та відмови сервісу.

2)Розробка бекапів

Створення резервних копій або бекапів в Google Cloud Platform (GCP) є важливою практикою для забезпечення безпеки та надійності даних.



**Посилання:** [**https://github.com/Woggy1/Coursework**](https://github.com/Woggy1/Coursework)