

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет

«Дніпровська політехніка»



ЗВІТ

з лабораторної роботи №5

з дисципліни

**“Поглиблене програмування в середовищі Java”**

Виконала

студентка групи

122-21-2

Мартиненко Юлія Володимирівна

Дніпро

2025

## Лабораторна робота №5

### Тема: Jdbc

Створити базу даних в будь-якому сервері баз даних. Створити таблицю з переліком студентів вказати їх прізвище, ім'я, по батькові, день народження, номер залікової книжки та ID.

Створити програму, що буде дозволяти виводити на екран інформацію про студентів, які народилися в тому чи іншому місяці року. Програма повинна завдяки системі jdbc під'єднатися до вашої бази даних та робити до неї запити. Вимог до розробки бази даних немає. Програма ж має бути написана за усіма стандартами ООП. Та може бути спроектована за двох принципів:

- при будь-якій ситуації буде забиратися весь перелік студентів, а вже на стороні java буде зроблено пошук необхідного

- SQL запит буде сформований згідно запиту, який зробив користувач і вже сервер управління баз даних буде вирішувати, які самі студенти народилися в тому чи іншому місяці.

У висновку обов'язково пояснити чому вибрали той чи інший принцип, які в нього переваги та недоліки. Оцінка не залежить від того, який сервер управління баз даних вибрали. Перелік студентів зробити не менше 20 людей. Місяць червень зробити місяцем, коли в жодного зі студентів немає дня народження.

SQL код створення бази даних розмістити в проєкті 6 лабораторної роботи в файлі database в пакеті resources. Для використання цієї лабораторної роботи рекомендується активно використовувати знання, отримані на дисципліні, що стосуються розробки баз даних.

До паперового звіту обов'язково додати принтскрин з програми, в якій ви дивитесь інформацію вашого сервера управління баз даних, де показати створену таблицю, її ім'я та загальні відомості бази даних, наприклад назва, ім'я, назва користувача адміністратора, пароль тощо. Для роботи з сервером управління баз даних рекомендуємо використовувати програмне забезпечення

компанії jetbrains datagrip. Або вбудовану панель користування базами даних, що міститься у середовищі intelliJ Idea, яка на сьогоднішній день підтримує майже всі сервери управління баз даних.

## Виконання

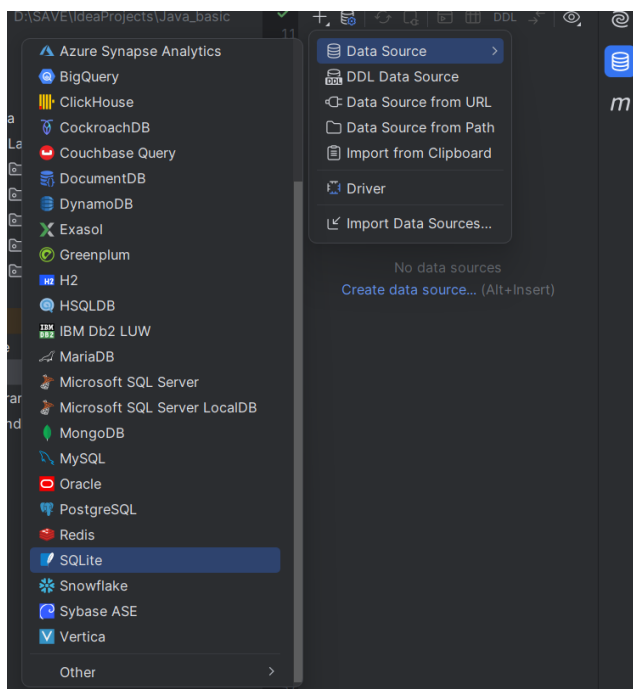
1. Додаю нову залежність у файл pom.xml:

```
<dependency>
  <groupId>org.xerial</groupId>
  <artifactId>sqlite-jdbc</artifactId>
  <version>3.42.0.0</version>
</dependency>
```

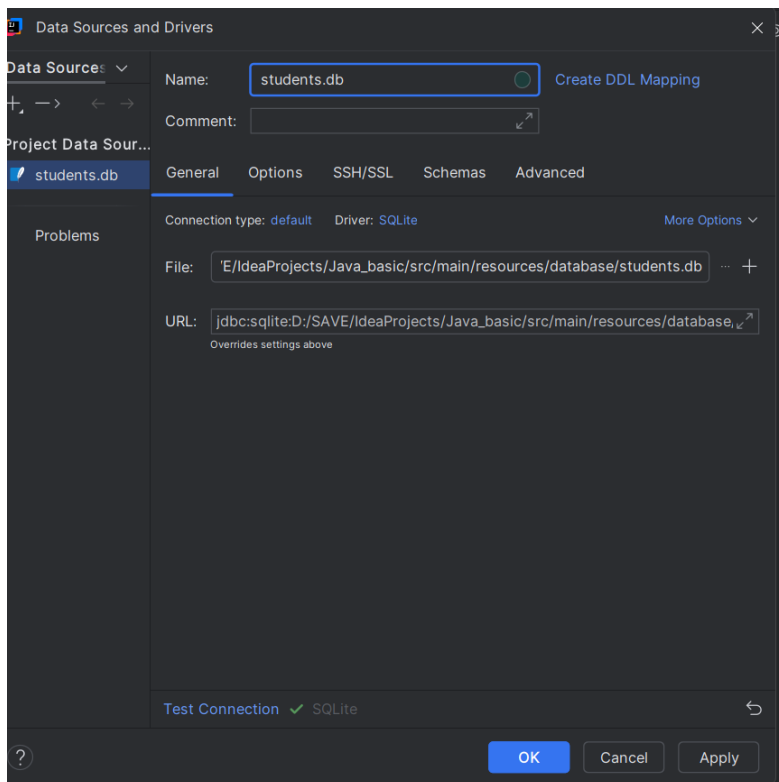
2. Я використовую вбудовану панель користування базами даних, що міститься у середовищі intelliJ Idea, яка на сьогоднішній день підтримує майже всі сервери управління баз даних.

У IntelliJ IDEA відкриваю вкладку Database (View → Tool Windows → Database).

Обираю:



Налаштовую файл та проводжу test connection:



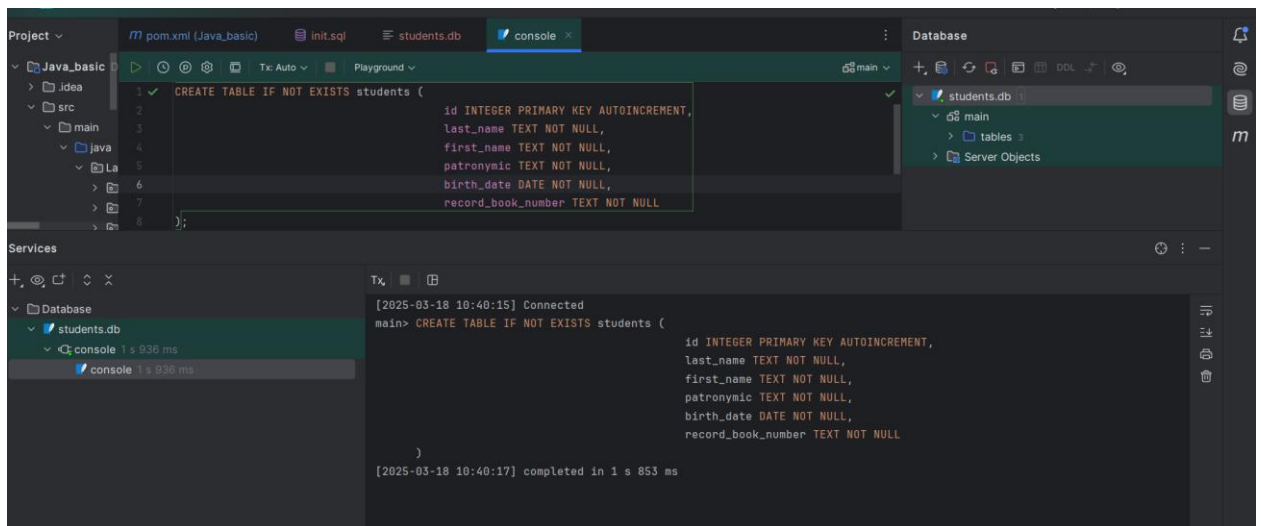
### 3. Написання SQL-скрипта для створення таблиці студентів

Додаю скрипт:

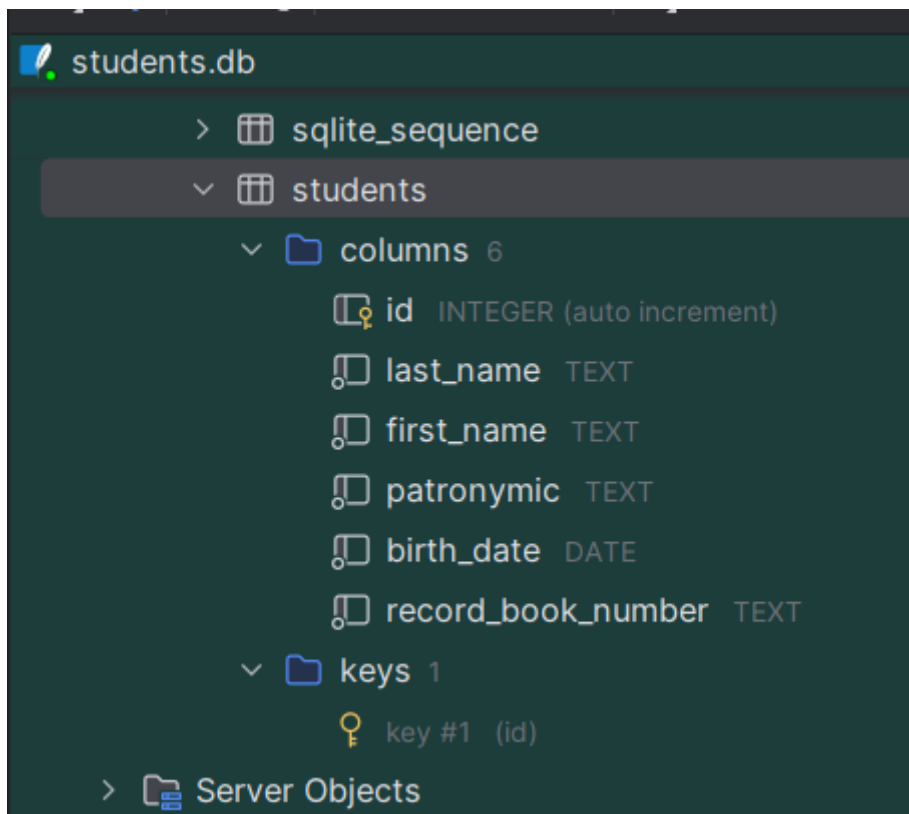
The screenshot shows the IntelliJ IDEA interface with the 'init.sql' file open in the 'console' tab. The SQL script is as follows:

```
1 CREATE TABLE IF NOT EXISTS students (  
2     id INTEGER PRIMARY KEY AUTOINCREMENT,  
3     last_name TEXT NOT NULL,  
4     first_name TEXT NOT NULL,  
5     patronymic TEXT NOT NULL,  
6     birth_date DATE NOT NULL,  
7     record_book_number TEXT NOT NULL  
8 );  
9
```

Потім відкриваю у Database students.db, та обираю Open Query Console, додаю цей SQL-код та натискаю Run:



Результат створення:



4. Створюємо клас, який забезпечить підключення до бази даних SQLite:

```
Java_basic)  init.sql  students  console  DatabaseManager.java x Student.java  StudentService.java  Main.java  v
1  package Lab.LR_5.db;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.SQLException;
6
7  public class DatabaseManager { 5 usages
8      private static final String URL = "jdbc:sqlite:src/main/resources/database/students.db"; // Ensure the path is correct 1 usage
9
10     // Method to connect to the database
11     public static Connection connect() throws SQLException { 3 usages
12         return DriverManager.getConnection(URL);
13     }
14 }
15
```

## Student.java:

```
m pom.xml (Java_basic)  init.sql  console  DatabaseManager.java  Student.java x StudentService.java  Main.java  5
1  package Lab.LR_5.model;
2
3  import java.time.LocalDate;
4
5  public class Student {
6      private int id; 4 usages
7      private String lastName; 5 usages
8      private String firstName; 5 usages
9      private String patronymic; 5 usages
10     private LocalDate birthDate; 5 usages
11     private String recordBookNumber; 5 usages
12
13     // with student ID
14     public Student(int id, String lastName, String firstName, String patronymic, LocalDate birthDate, String recordBookNumber) {
15         this.id = id;
16         this.lastName = lastName;
17         this.firstName = firstName;
18         this.patronymic = patronymic;
19         this.birthDate = birthDate;
20         this.recordBookNumber = recordBookNumber;
21     }
22
23     // without ID
24     public Student(String lastName, String firstName, String patronymic, LocalDate birthDate, String recordBookNumber) {
25         this.lastName = lastName;
26         this.firstName = firstName;
27         this.patronymic = patronymic;
28         this.birthDate = birthDate;
29         this.recordBookNumber = recordBookNumber;
30     }
31 }
```

```

5      public class Student {
32          // get
33          public int getId() { return id; } no usages
34          public String getLastName() { return lastName; }
35          public String getFirstName() { return firstName; }
36          public String getPatronymic() { return patronymic; } 1 usage
37          public LocalDate getBirthDate() { return birthDate; } 1 usage
38          public String getRecordBookNumber() { return recordBookNumber; } 1 usage
39
40          // set
41          public void setId(int id) { this.id = id; } no usages
42          public void setLastName(String lastName) { this.lastName = lastName; }
43          public void setFirstName(String firstName) { this.firstName = firstName; }
44          public void setPatronymic(String patronymic) { this.patronymic = patronymic; } no usages
45          public void setBirthDate(LocalDate birthDate) { this.birthDate = birthDate; } no usages
46          public void setRecordBookNumber(String recordBookNumber) { this.recordBookNumber = recordBookNumber; }
47
48          @Override
49          public String toString() {
50              return "Student{" +
51                  "id=" + id +
52                  ", lastName='" + lastName + '\'' +
53                  ", firstName='" + firstName + '\'' +
54                  ", patronymic='" + patronymic + '\'' +
55                  ", birthDate=" + birthDate +
56                  ", recordBookNumber='" + recordBookNumber + '\'' +
57                  '}';
58          }
59      }
60

```

Цей клас представляє студента як об'єкт.

Клас для взаємодії з базою даних (StudentService):

```
m pom.xml (Java_basic)  init.sql  console  DatabaseManager.java  Student.java  StudentService.java x Main.java

11 public class StudentService { no usages
29
30     public List<Student> getStudentsByMonth(int month) { no usages
31         List<Student> students = new ArrayList<>();
32         String sql = "SELECT * FROM students WHERE strftime('%m', birth_date) = ?";
33
34         try (Connection conn = DatabaseManager.connect();
35              PreparedStatement stmt = conn.prepareStatement(sql)) {
36
37             stmt.setString(1, String.format("%02d", month));
38             ResultSet rs = stmt.executeQuery();
39
40             while (rs.next()) {
41                 students.add(new Student(
42                     rs.getInt(columnLabel: "id"),
43                     rs.getString(columnLabel: "last_name"),
44                     rs.getString(columnLabel: "first_name"),
45                     rs.getString(columnLabel: "patronymic"),
46                     LocalDate.parse(rs.getString(columnLabel: "birth_date")),
47                     rs.getString(columnLabel: "record_book_number")
48                 ));
49             }
50         } catch (SQLException e) {
51             e.printStackTrace();
52         }
53         return students;
54     }
55 }
56

m pom.xml (Java_basic)  init.sql  console  DatabaseManager.java  Student.java  StudentService.java x Main.java

1 package Lab.LR_5.service;
2
3 import Lab.LR_5.db.DatabaseManager;
4 import Lab.LR_5.model.Student;
5
6 import java.sql.*;
7 import java.time.LocalDate;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 public class StudentService { no usages
12     @
13     public void addStudent(Student student) { no usages
14         String sql = "INSERT INTO students (last_name, first_name, patronymic, birth_date, record_book_number) VALUES (?, ?, ?, ?, ?)";
15
16         try (Connection conn = DatabaseManager.connect();
17              PreparedStatement stmt = conn.prepareStatement(sql)) {
18
19             stmt.setString(1, student.getLastName());
20             stmt.setString(2, student.getFirstName());
21             stmt.setString(3, student.getPatronymic());
22             stmt.setString(4, student.getBirthDate().toString());
23             stmt.setString(5, student.getRecordBookNumber());
24             stmt.executeUpdate();
25         } catch (SQLException e) {
26             e.printStackTrace();
27         }
28     }
29
30     public List<Student> getStudentsByMonth(int month) { no usages
31         List<Student> students = new ArrayList<>();
32     }
33 }
```

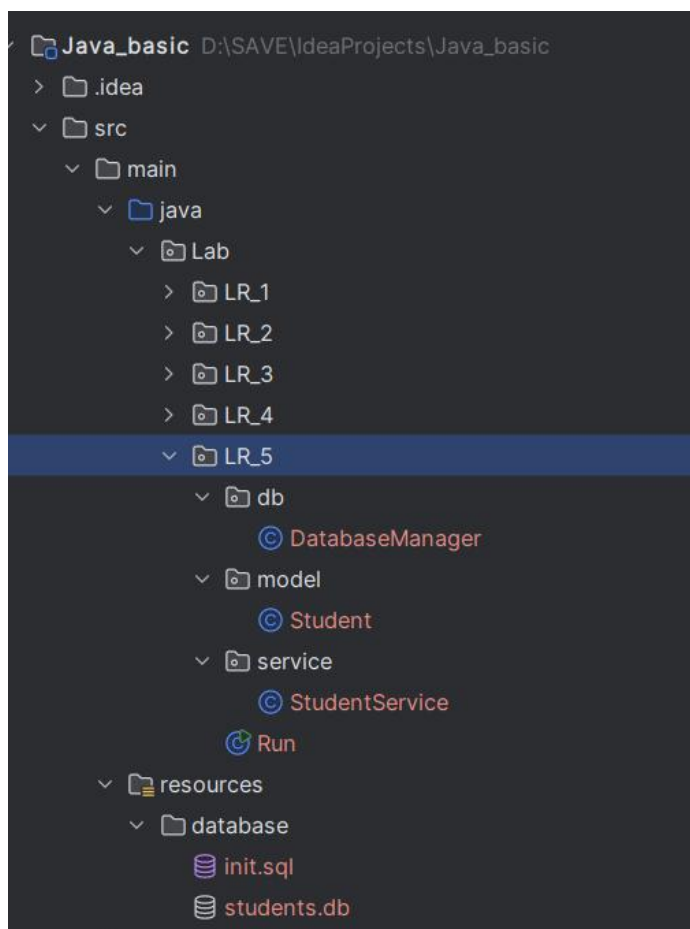
Цей клас містить методи додавання студента та отримання студентів за місяцем народження.

Run.java:

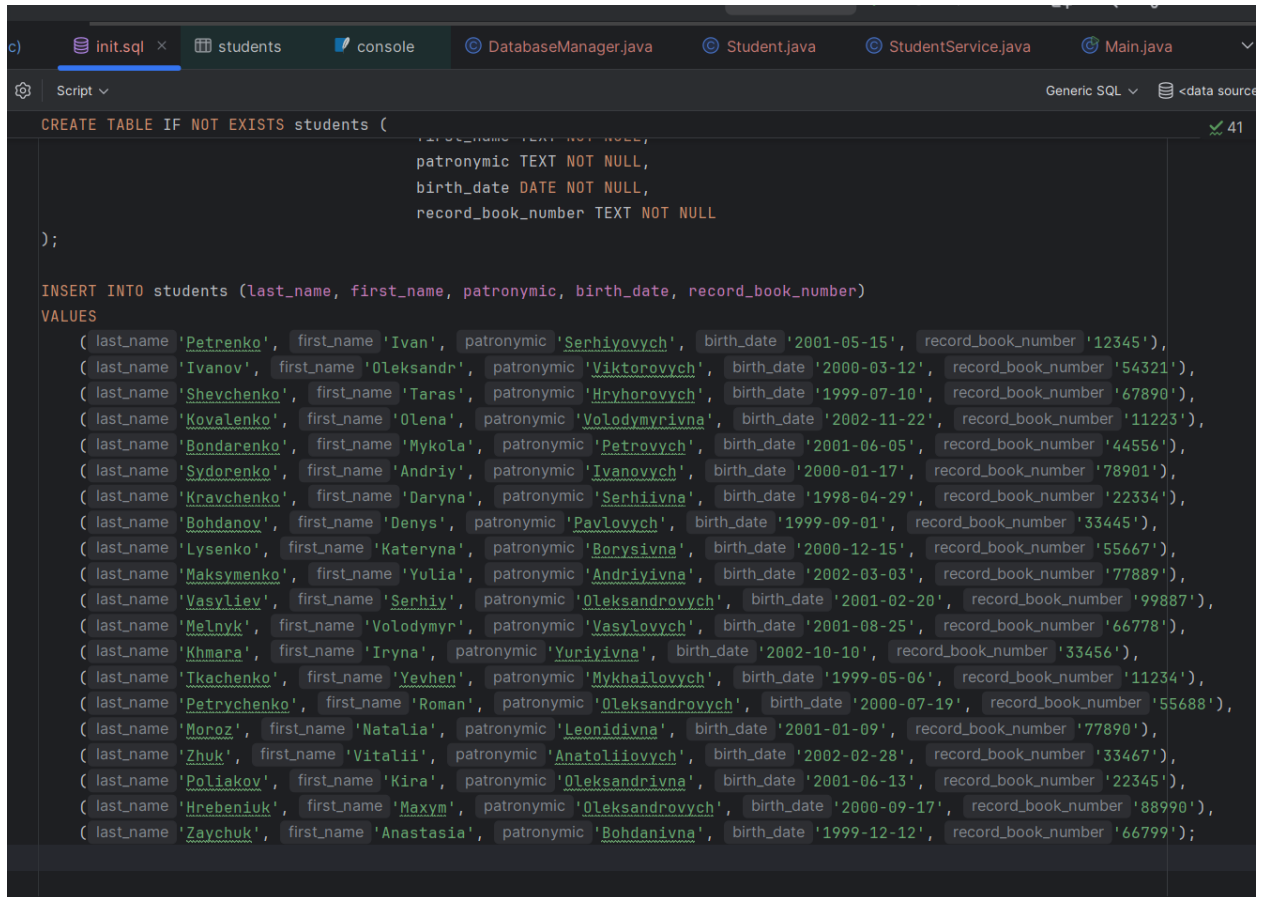


```
students console controller\Run.java DatabaseManager.java Student.java StudentService.java LR_5\Run.java x
1 package Lab.LR_5;
2
3 import Lab.LR_5.db.DatabaseManager;
4 import Lab.LR_5.model.Student;
5 import Lab.LR_5.service.StudentService;
6
7 import java.sql.Connection;
8 import java.sql.DatabaseMetaData;
9 import java.sql.SQLException;
10 import java.util.List;
11 import java.util.Scanner;
12
13 public class Run {
14     public static void main(String[] args) {
15         // Check if the table exists
16         if (!isTableExist()) {
17             System.out.println("Table not found. Please initialize the database manually.");
18         }
19
20         StudentService studentService = new StudentService();
21         Scanner scanner = new Scanner(System.in);
22
23         // Month input
24         System.out.print("Enter the month number (1-12): ");
25         int month = scanner.nextInt();
26
27         List<Student> students = studentService.getStudentsByMonth(month);
28         if (students.isEmpty()) {
29             System.out.println("No students born in this month.");
30         } else {
31             System.out.println("Students born in the " + month + "-th month:");
32             students.forEach(System.out::println);
33         }
34     }
35
36     private static boolean isTableExist() {
37         DatabaseManager dbManager = new DatabaseManager();
38         Connection connection = dbManager.getConnection();
39         DatabaseMetaData metaData = connection.getMetaData();
40         try {
41             String sql = "SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = 'students'";
42             java.sql.ResultSet resultSet = connection.createStatement().executeQuery(sql);
43             return resultSet.next();
44         } catch (SQLException e) {
45             e.printStackTrace();
46             return false;
47         }
48     }
49 }
```

Структура проекта:

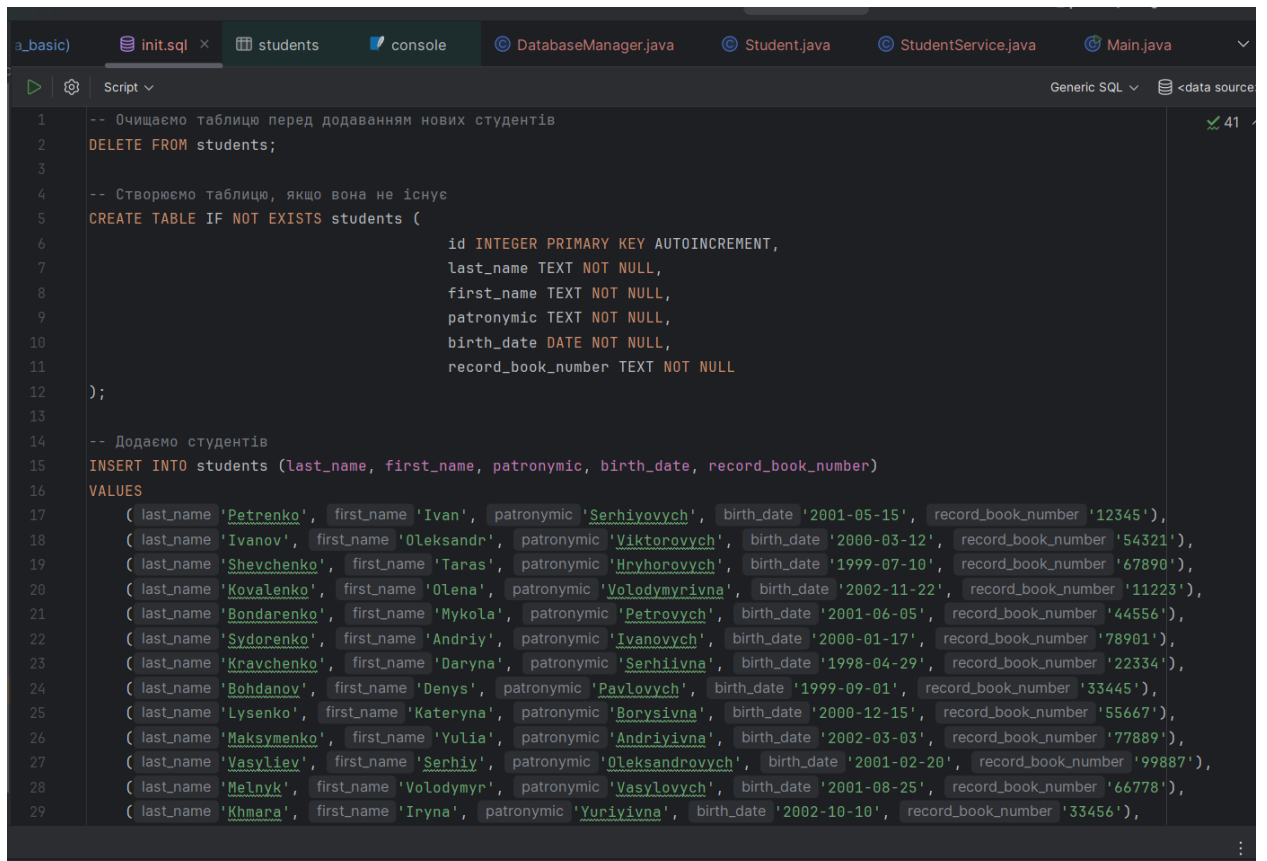


## Оптимізуємо додавання даних студентів через init.sql:



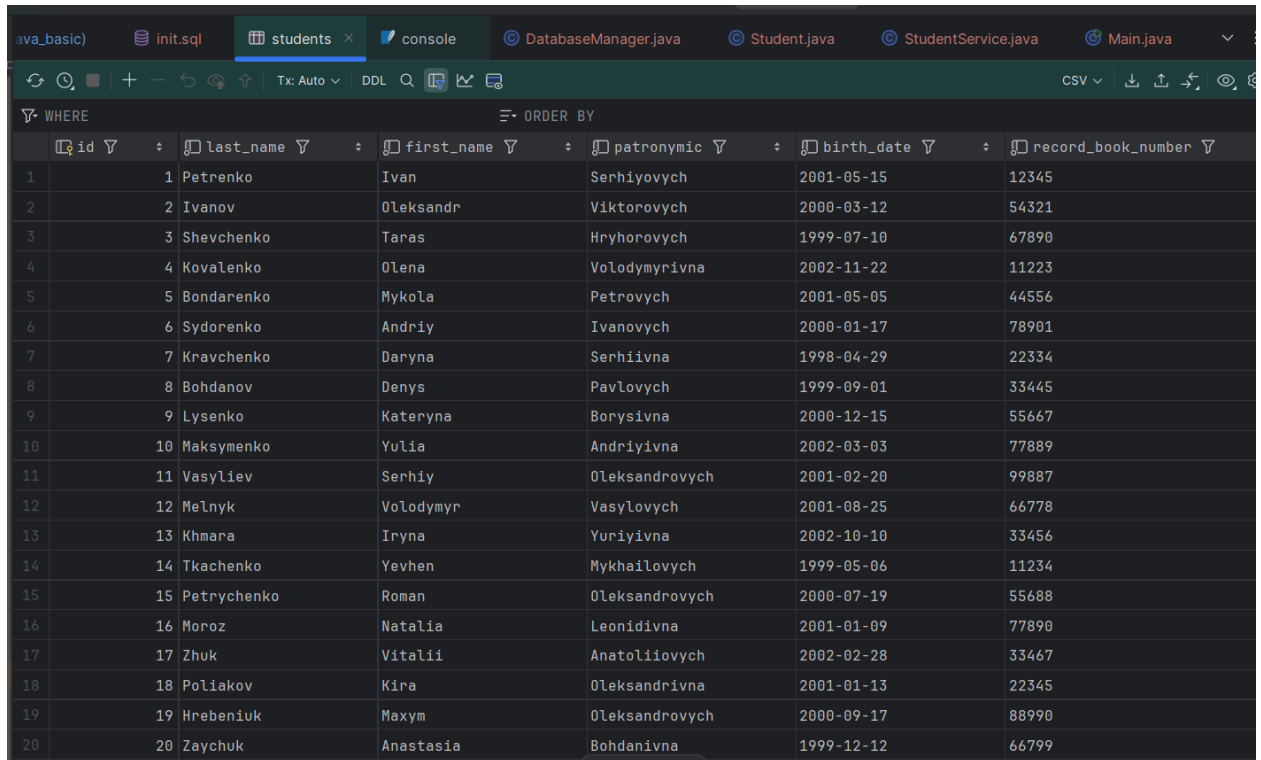
```
CREATE TABLE IF NOT EXISTS students (  
    last_name TEXT NOT NULL,  
    first_name TEXT NOT NULL,  
    patronymic TEXT NOT NULL,  
    birth_date DATE NOT NULL,  
    record_book_number TEXT NOT NULL  
);  
  
INSERT INTO students (last_name, first_name, patronymic, birth_date, record_book_number)  
VALUES  
(last_name 'Petrenko', first_name 'Ivan', patronymic 'Serhiyovych', birth_date '2001-05-15', record_book_number '12345'),  
(last_name 'Ivanov', first_name 'Oleksandr', patronymic 'Viktorovych', birth_date '2000-03-12', record_book_number '54321'),  
(last_name 'Shevchenko', first_name 'Taras', patronymic 'Hryhorovych', birth_date '1999-07-10', record_book_number '67890'),  
(last_name 'Kovalenko', first_name 'Olena', patronymic 'Volodymyrivna', birth_date '2002-11-22', record_book_number '11223'),  
(last_name 'Bondarenko', first_name 'Mykola', patronymic 'Petrovych', birth_date '2001-06-05', record_book_number '44556'),  
(last_name 'Sydorenko', first_name 'Andriy', patronymic 'Ivanovych', birth_date '2000-01-17', record_book_number '78901'),  
(last_name 'Kravchenko', first_name 'Daryna', patronymic 'Serhiivna', birth_date '1998-04-29', record_book_number '22334'),  
(last_name 'Bohdanov', first_name 'Denys', patronymic 'Pavlovych', birth_date '1999-09-01', record_book_number '33445'),  
(last_name 'Lysenko', first_name 'Kateryna', patronymic 'Borysivna', birth_date '2000-12-15', record_book_number '55667'),  
(last_name 'Maksymenko', first_name 'Yulia', patronymic 'Andriyivna', birth_date '2002-03-03', record_book_number '77889'),  
(last_name 'Vasyliiev', first_name 'Serhiy', patronymic 'Oleksandrovych', birth_date '2001-02-20', record_book_number '99887'),  
(last_name 'Melnyk', first_name 'Volodymyr', patronymic 'Vasylovych', birth_date '2001-08-25', record_book_number '66778'),  
(last_name 'Khmara', first_name 'Iryna', patronymic 'Yuriyivna', birth_date '2002-10-10', record_book_number '33456'),  
(last_name 'Tkachenko', first_name 'Yevhen', patronymic 'Mykhailovych', birth_date '1999-05-06', record_book_number '11234'),  
(last_name 'Petrychenko', first_name 'Roman', patronymic 'Oleksandrovych', birth_date '2000-07-19', record_book_number '55688'),  
(last_name 'Moroz', first_name 'Natalia', patronymic 'Leonidivna', birth_date '2001-01-09', record_book_number '77890'),  
(last_name 'Zhuk', first_name 'Vitalii', patronymic 'Anatoliiovych', birth_date '2002-02-28', record_book_number '33467'),  
(last_name 'Poliaikov', first_name 'Kira', patronymic 'Oleksandrivna', birth_date '2001-06-13', record_book_number '22345'),  
(last_name 'Hrebeniuk', first_name 'Maxym', patronymic 'Oleksandrovych', birth_date '2000-09-17', record_book_number '88990'),  
(last_name 'Zaychuk', first_name 'Anastasia', patronymic 'Bohdanivna', birth_date '1999-12-12', record_book_number '66799');
```

## Init.sql:



```
-- Очищаємо таблицю перед додаванням нових студентів  
DELETE FROM students;  
  
-- Створюємо таблицю, якщо вона не існує  
CREATE TABLE IF NOT EXISTS students (  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    last_name TEXT NOT NULL,  
    first_name TEXT NOT NULL,  
    patronymic TEXT NOT NULL,  
    birth_date DATE NOT NULL,  
    record_book_number TEXT NOT NULL  
);  
  
-- Додаємо студентів  
INSERT INTO students (last_name, first_name, patronymic, birth_date, record_book_number)  
VALUES  
(last_name 'Petrenko', first_name 'Ivan', patronymic 'Serhiyovych', birth_date '2001-05-15', record_book_number '12345'),  
(last_name 'Ivanov', first_name 'Oleksandr', patronymic 'Viktorovych', birth_date '2000-03-12', record_book_number '54321'),  
(last_name 'Shevchenko', first_name 'Taras', patronymic 'Hryhorovych', birth_date '1999-07-10', record_book_number '67890'),  
(last_name 'Kovalenko', first_name 'Olena', patronymic 'Volodymyrivna', birth_date '2002-11-22', record_book_number '11223'),  
(last_name 'Bondarenko', first_name 'Mykola', patronymic 'Petrovych', birth_date '2001-06-05', record_book_number '44556'),  
(last_name 'Sydorenko', first_name 'Andriy', patronymic 'Ivanovych', birth_date '2000-01-17', record_book_number '78901'),  
(last_name 'Kravchenko', first_name 'Daryna', patronymic 'Serhiivna', birth_date '1998-04-29', record_book_number '22334'),  
(last_name 'Bohdanov', first_name 'Denys', patronymic 'Pavlovych', birth_date '1999-09-01', record_book_number '33445'),  
(last_name 'Lysenko', first_name 'Kateryna', patronymic 'Borysivna', birth_date '2000-12-15', record_book_number '55667'),  
(last_name 'Maksymenko', first_name 'Yulia', patronymic 'Andriyivna', birth_date '2002-03-03', record_book_number '77889'),  
(last_name 'Vasyliiev', first_name 'Serhiy', patronymic 'Oleksandrovych', birth_date '2001-02-20', record_book_number '99887'),  
(last_name 'Melnyk', first_name 'Volodymyr', patronymic 'Vasylovych', birth_date '2001-08-25', record_book_number '66778'),  
(last_name 'Khmara', first_name 'Iryna', patronymic 'Yuriyivna', birth_date '2002-10-10', record_book_number '33456'),
```

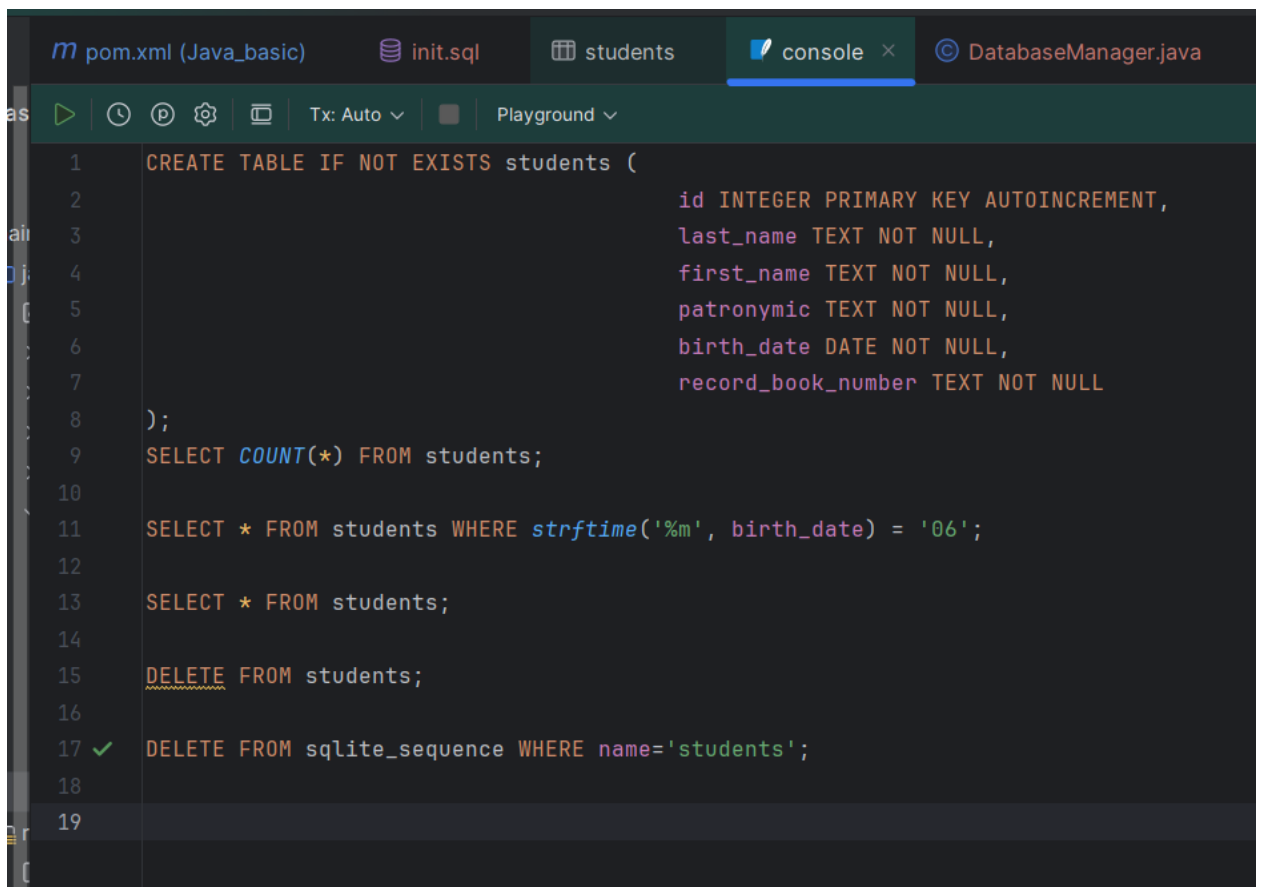
Запускаємо init.sql та отримуємо таблицю:



The screenshot shows a web application interface with a table of students. The table has columns for id, last\_name, first\_name, patronymic, birth\_date, and record\_book\_number. The data is sorted by id in ascending order.

	id	last_name	first_name	patronymic	birth_date	record_book_number
1	1	Petrenko	Ivan	Serhiyovych	2001-05-15	12345
2	2	Ivanov	Oleksandr	Viktorovych	2000-03-12	54321
3	3	Shevchenko	Taras	Hryhorovych	1999-07-10	67890
4	4	Kovalenko	Olena	Volodymyrivna	2002-11-22	11223
5	5	Bondarenko	Mykola	Petrovych	2001-05-05	44556
6	6	Sydorenko	Andriy	Ivanovych	2000-01-17	78901
7	7	Kravchenko	Daryna	Serhiivna	1998-04-29	22334
8	8	Bohdanov	Denys	Pavlovych	1999-09-01	33445
9	9	Lysenko	Kateryna	Borysivna	2000-12-15	55667
10	10	Maksymenko	Yulia	Andriyivna	2002-03-03	77889
11	11	Vasyliiev	Serhiy	Oleksandrovych	2001-02-20	99887
12	12	Melnyk	Volodymyr	Vasylovych	2001-08-25	66778
13	13	Khmara	Iryna	Yuriyivna	2002-10-10	33456
14	14	Tkachenko	Yevhen	Mykhailovych	1999-05-06	11234
15	15	Petrychenko	Roman	Oleksandrovych	2000-07-19	55688
16	16	Moroz	Natalia	Leonidivna	2001-01-09	77890
17	17	Zhuk	Vitalii	Anatoliiovych	2002-02-28	33467
18	18	Poliaikov	Kira	Oleksandrivna	2001-01-13	22345
19	19	Hrebeniuk	Maxym	Oleksandrovych	2000-09-17	88990
20	20	Zaychuk	Anastasia	Bohdanivna	1999-12-12	66799

Через консоль розбираюсь з різними командами:



```
1 CREATE TABLE IF NOT EXISTS students (  
2     id INTEGER PRIMARY KEY AUTOINCREMENT,  
3     last_name TEXT NOT NULL,  
4     first_name TEXT NOT NULL,  
5     patronymic TEXT NOT NULL,  
6     birth_date DATE NOT NULL,  
7     record_book_number TEXT NOT NULL  
8 );  
9 SELECT COUNT(*) FROM students;  
10  
11 SELECT * FROM students WHERE strftime('%m', birth_date) = '06';  
12  
13 SELECT * FROM students;  
14  
15 DELETE FROM students;  
16  
17 DELETE FROM sqlite_sequence WHERE name='students';  
18  
19
```

5. Запускаємо програму:

```
"C:\Program Files\Java\jdk-21\bin\java
Enter the month number (1-12): 6
No students born in this month.

Process finished with exit code 0
|
```

```
Enter the month number (1-12): 5
Students born in the 5-th month:
Student{id=1, lastName='Petrenko', firstName='Ivan'}
Student{id=5, lastName='Bondarenko', firstName='Mikhail'}
Student{id=14, lastName='Tkachenko', firstName='Yuriy'}

Process finished with exit code 0
|
```

У процесі виконання лабораторної роботи ми створили базу даних, що містить таблицю студентів, і розробили програму на мові Java, яка підключається до цієї бази даних через JDBC, щоб виконувати запити на отримання студентів, що народилися в конкретному місяці.

Кроки виконання:

Створення бази даних:

Ми вибрали SQLite як сервер управління базами даних, оскільки це легкий, вбудований у застосунки сервер, що не вимагає окремого налаштування.

Таблиця студентів була створена за допомогою SQL-скрипта init.sql, який містить дані про студентів, включаючи їх прізвище, ім'я, по батькові, день народження, номер залікової книжки та унікальний ID.

Розробка програми на Java:

Програма підключається до бази даних через JDBC, що дозволяє виконувати SQL запити до бази даних.

Програма отримує вхідний місяць від користувача та використовує SQL запит для вибору студентів, що народилися в цей місяць.

Ми обрали другий підхід, коли SQL-запит виконується безпосередньо в базі даних, що забезпечує кращу ефективність та зменшує навантаження на сервер Java, оскільки база даних буде виконувати фільтрацію даних. Це дозволяє уникнути витрат на обробку великої кількості даних на стороні Java.

Вибір принципу реалізації:

Ми вирішили використовувати другий підхід, де фільтрація даних здійснюється через SQL-запит (не на стороні Java). Цей підхід має кілька переваг:

Переваги:

Ефективність: SQL-сервер спеціалізується на обробці запитів, що дозволяє виконувати фільтрацію та пошук значно швидше.

Масштабованість: Якщо кількість студентів у базі даних збільшиться, цей підхід залишатиметься ефективним.

Недоліки:

Потрібно правильно формулювати запити, що може бути складно при великих або складних запитах.

Підключення та взаємодія з базою даних:

Підключення здійснюється через JDBC, що є стандартним способом взаємодії Java з різними базами даних.

За допомогою SQL-скрипту, який створює таблицю студентів, ми забезпечили правильну ініціалізацію бази даних та її наповнення.

Рекомендації щодо подальшого розвитку:

Можна додати функціональність для зміни даних про студентів або видалення їх, а також створити інші запити для додаткової фільтрації.

Використовувати більш складні SQL запити для реалізації складніших вимог (наприклад, пошук студентів за кількома критеріями).

Загальний висновок:

В результаті виконання лабораторної роботи було досягнуто поставленої мети: створено базу даних, написано програму для пошуку студентів за місяцем народження за допомогою SQL запитів через JDBC.

Вибраний підхід із фільтрацією даних безпосередньо в базі даних є оптимальним для цієї задачі.

Висновок. Під час виконання лабораторної роботи №5 з дисципліни «Поглиблене програмування в середовищі Java» було вивчено Jdbc та створено програму й базу даних згідно завданню.