UNIVERSITETI®
METROPOLITAN
TIRANA

FACULTY OF COMPUTER SCIENCES AND IT

Software Engineering Program

Database Management System

Final Project Documentation

Student Management System

Worked by: Krisalda Mihali

1. Introduction

The **Student Management System (SMS)** is a software application designed to help educational institutions efficiently manage student data, courses, and academic performance. The system integrates user management features such as secure login and registration, along with comprehensive management functionalities for student records, course enrollments, and grades. Built with **JavaFX** for the graphical user interface (GUI) and **MySQL** for database, it supports essential operations like adding, updating, and deleting student and course information. The system is designed using key object-oriented programming (OOP) principles such as **abstraction**, **encapsulation**, **polymorphism**, and **inheritance** to promote maintainability and scalability.

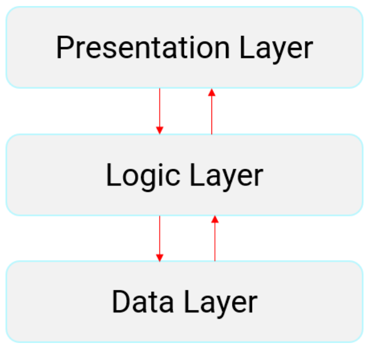


Fig.1. Key Architecture

2. Key Features

The system includes several core functionalities that serve the needs of administrators:

- **User Authentication (Login/Register):** Users can securely log in or register within the system, ensuring that only authorized users can access and manage data.
- **Dashboard:** The dashboard provides an overview and easy access to various system functionalities, including student management, course management, and grade management. Additionally, the dashboard displays student statistics retrieved directly from the database. These statistics are visualized using various types of charts.
- **Student Management:** Administrators can manage student records, including adding, editing, and deleting students. The system also supports viewing student details, such as name, gender, enrollment date, and contact information.
- **Course Management:** The system allows administrators to manage courses by adding new courses, updating course details (e.g., course name, department, instructor), and deleting courses. It also provides an overview of available courses.
- **Grade Management:** The grade management feature allows administrators to input, update or delete grades for students in specific courses, providing a means of tracking academic progress.

- **Graphical Data Visualizations:** The system includes charts and tables to visualize key data points, such as the total number of student enrollments, gender distribution, student data, courses and grades tables.
- **JavaFX Interface:** The application leverages **JavaFX** for creating a responsive and user-friendly interface, using components such as **TableView** for displaying tables of students and courses, **TextField** for input forms, and **Buttons** for executing various operations.

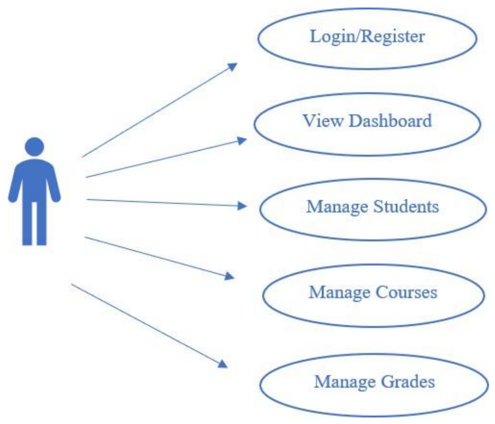


Fig.2. Usecase Diagram

3. Technical Implementation

The **Student Management System** is built using a combination of **JavaFX** for the GUI and **MySQL** for database, with the system's architecture based on the principles of **Object-Oriented Programming (OOP)**, such as abstraction, encapsulation, inheritance, and polymorphism.

- **Database Design:**
    - **MySQL Database:** The system operates with a MySQL database, facilitating the storage and management of various data components.
    - **CRUD Operations:** Each page performs basic CRUD (Create, Read, Update, Delete) operations for students, courses, and grades. For example, when adding a course, the system inserts a new record into the `courses` table using a parameterized SQL query to prevent SQL injection.
    - **Events and Actions:** Event handlers are attached to buttons (e.g., **Add**, **Clear**, **Delete**, **Update**) to trigger appropriate actions, such as adding or deleting students or courses, or clearing form fields.

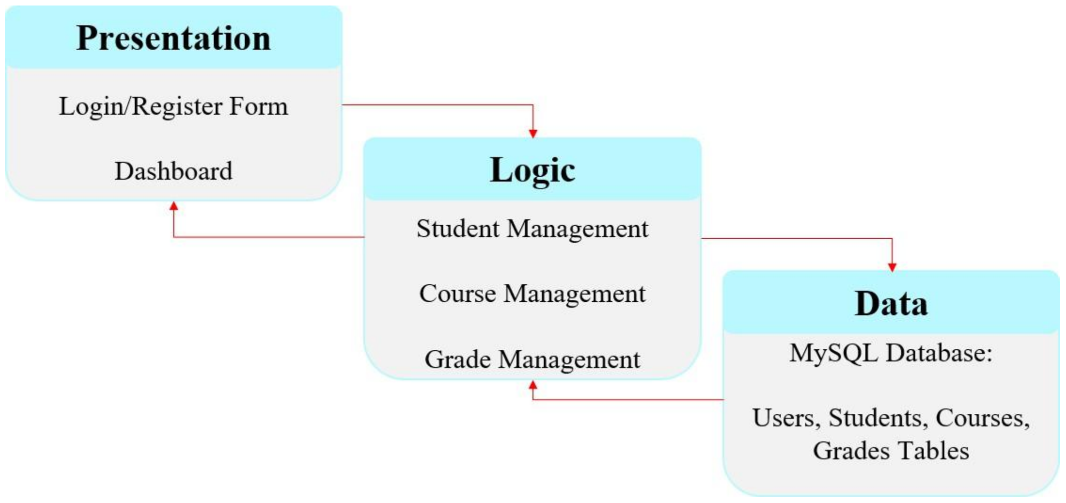


Fig.3. Architecture Details

3.1. Database conceptualization

3.1.1. Conceptual Data Model (CDM)

The database is designed to manage an academic environment where user information, student records, course details, grades, and grade-related logs are systematically stored and maintained. Below is a description of each table:

1. **Users Table**:
    - Stores login credentials and basic user information, enabling access to the system. It ensures secure authentication and authorization of users through unique identifiers such as usernames and emails.
2. **Students Table**:
    - Captures detailed information about students, including personal details (e.g., name, date of birth), academic details (e.g., study year, department), and contact information (e.g., email, phone number).
3. **Courses Table**:
    - Maintains a record of all courses available within the institution. This includes course identifiers, course names, and information about the department and instructor responsible for the course.
4. **Grades Table**:
    - Tracks the grades assigned to students for each course. This table also records the semester and academic year in which the grades were awarded.
5. **Grade Log Table**:
    - Provides an audit trail for changes to grades, such as updates or deletions. It records the action performed, the date of the action, and the type of modification made to ensure accountability.

3.1.2. Logical Data Model (LDM)

The logical data model defines the structure of the database in terms of entities, attributes, and relationships, ensuring data integrity and minimal redundancy.

Entities and Attributes

1. **Users Table**:
   - **Attributes**: `id` (Primary Key), `username` (Unique), `email` (Unique), `password`
   - **Purpose**: Manages system user credentials.
2. **Students Table**:
   - **Attributes**:
     - `student_id` (Primary Key)
     - `first_name`, `last_name`
     - `date_of_birth`
     - `gender`
     - `study_year`
     - `department`
     - `enrollment_date`
     - `email` (Unique)
     - `phone_number`
     - `status`
   - **Purpose**: Holds personal, academic, and contact information for students.
3. **Courses Table**:
   - **Attributes**:
     - `course_id` (Primary Key)
     - `course_name`
     - `department`
     - `instructor`
   - **Purpose**: Captures course details offered by the institution.
4. **Grades Table**:
   - **Attributes**:
     - `grade_id` (Primary Key)
     - `student_id` (Foreign Key referencing `students.student_id`)
     - `course_id` (Foreign Key referencing `courses.course_id`)
     - `grade`
     - `semester`
     - `year`
   - **Purpose**: Tracks student grades for specific courses in a given semester and year.
5. **Grade Log Table**:
   - **Attributes**:
     - `log_id` (Primary Key)
     - `student_id` (Foreign Key referencing `students.student_id`)
     - `course_id` (Foreign Key referencing `courses.course_id`)
     - `grade`
     - `action_date`

        - `action`
    - **Purpose**: Records grade-related actions such as updates and deletions for auditing purposes.

Relationships

1. **Students and Grades**:
    - **Relationship**: One-to-Many
    - **Description**: A student can receive grades for multiple courses.
2. **Courses and Grades**:
    - **Relationship**: One-to-Many
    - **Description**: A course can have grades assigned to multiple students.
3. **Students and Grade Log**:
    - **Relationship**: One-to-Many
    - **Description**: A student can have multiple grade-related actions logged.
4. **Courses and Grade Log**:
    - **Relationship**: One-to-Many
    - **Description**: A course can have multiple grade-related actions logged.

3.1.3. Relational Data Model (RDM)

The relational data model implements the logical data model in a relational database system. This model establishes primary and foreign key constraints to enforce relationships between tables.

Primary Keys

- `id` in `users`
- `student_id` in `students`
- `course_id` in `courses`
- `grade_id` in `grades`
- `log_id` in `grade_log`

Foreign Keys

- `student_id` in `grades` references `students.student_id`
- `course_id` in `grades` references `courses.course_id`
- `student_id` in `grade_log` references `students.student_id`
- `course_id` in `grade_log` references `courses.course_id`

Constraints

- **Unique Constraints**:
    - `username` and `email` in `users`
    - `email` in `students`
- **Referential Integrity**:

- Foreign key constraints ensure consistency across dependent tables.

3.1.4. Graphical Representation of the RDM

The attached **Entity-Relationship Diagram (ERD)** visually represents the relational data model. It highlights the following:

1. **Entities**:
   - `users`, `students`, `courses`, `grades`, and `grade_log`.
2. **Relationships**:
   - One-to-Many relationships between students and grades, students and grade logs, courses and grades, and courses and grade logs.
3. **Keys**:
   - Primary keys uniquely identify records in each table.
   - Foreign keys establish dependencies between related tables.

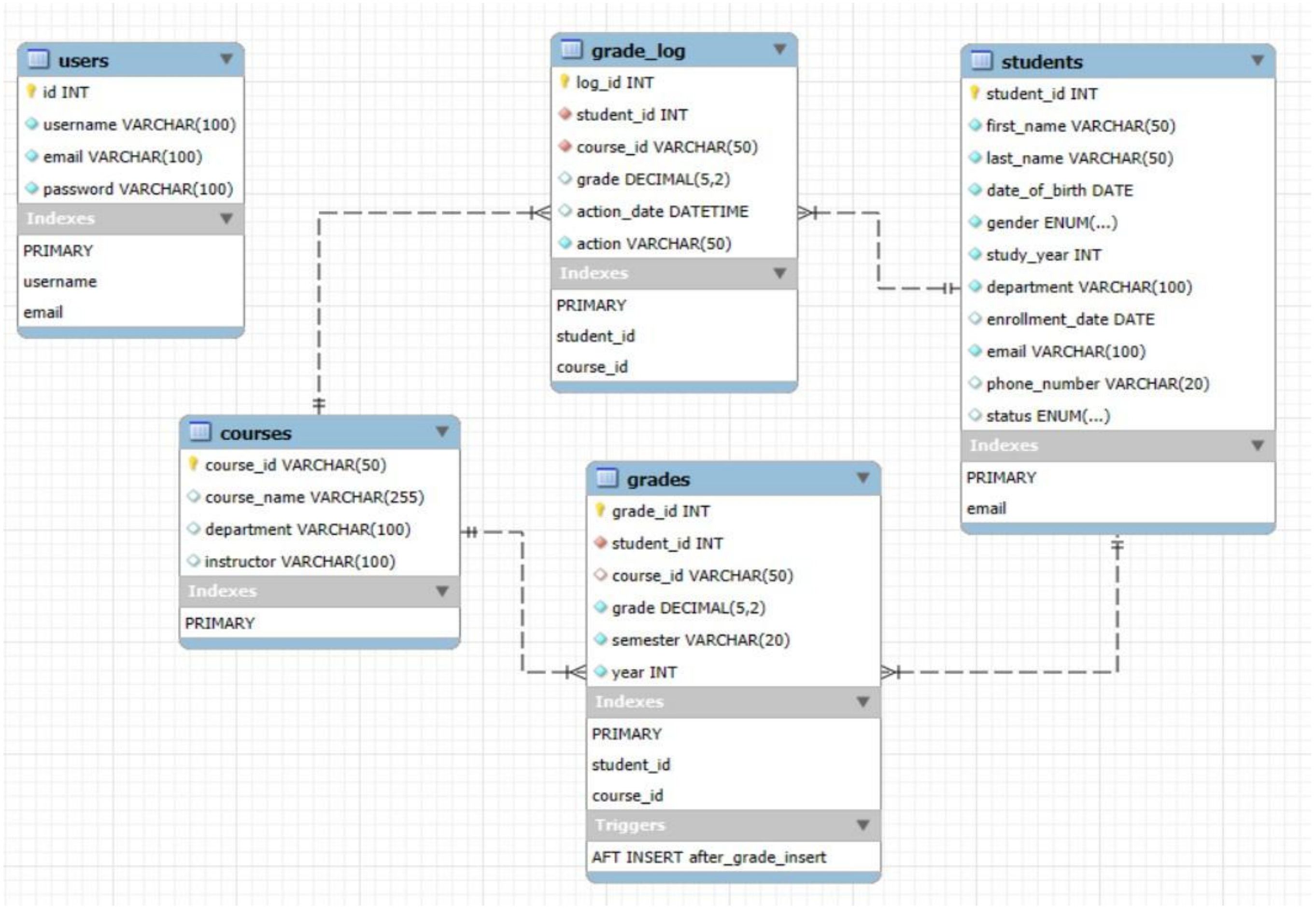
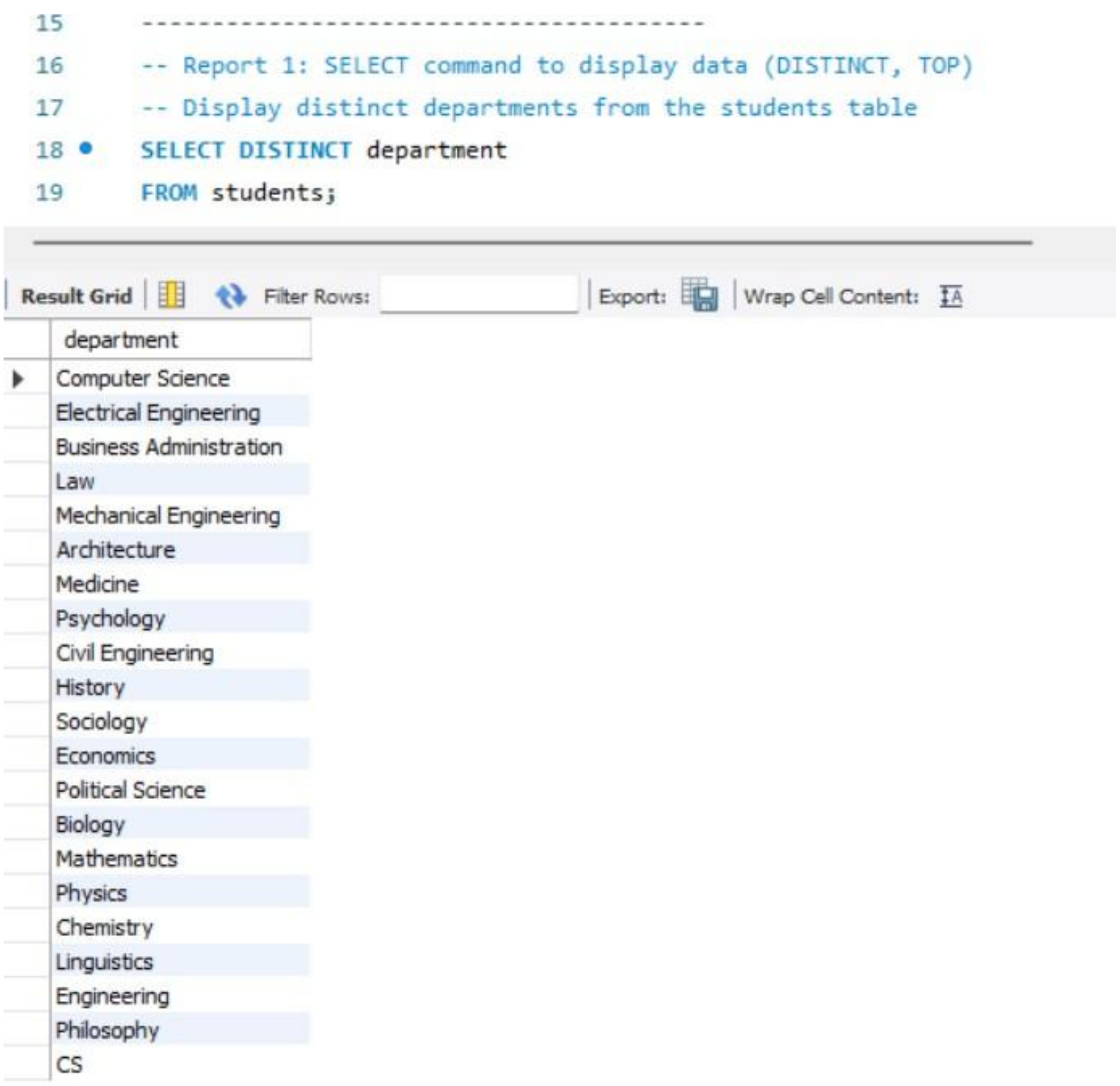


Fig.4. ER Diagrams

3.2. Data Processing

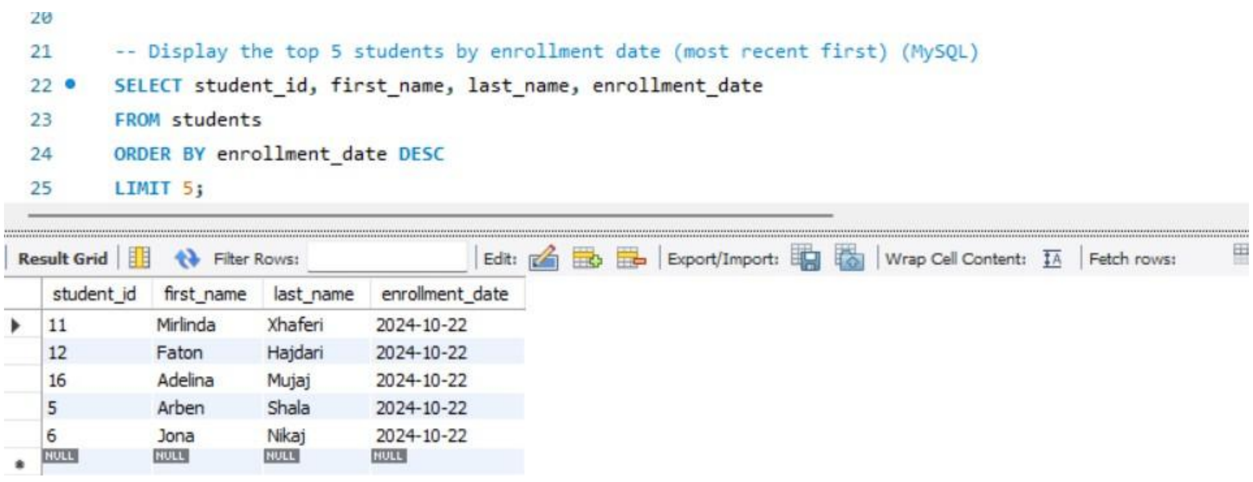
3.2.1. Query Implementation

Report 1: Use the SELECT command to display data (DISTINCT, TOP)



```
---------------------------------------
-- Report 1: SELECT command to display data (DISTINCT, TOP)
-- Display distinct departments from the students table
SELECT DISTINCT department
FROM students;
```

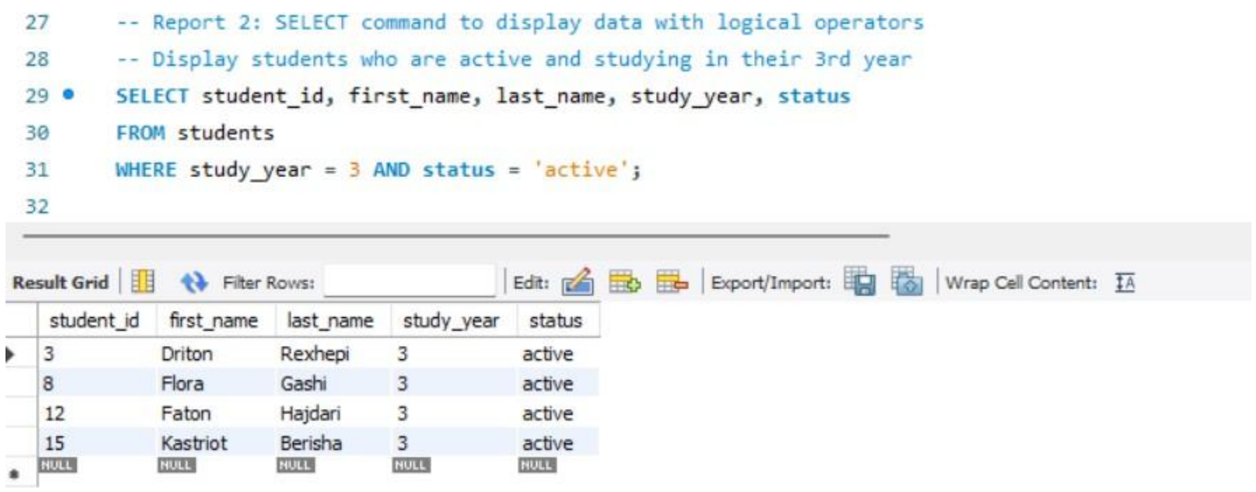
```
department
Computer Science
Electrical Engineering
Business Administration
Law
Mechanical Engineering
Architecture
Medicine
Psychology
Civil Engineering
History
Sociology
Economics
Political Science
Biology
Mathematics
Physics
Chemistry
Linguistics
Engineering
Philosophy
CS
```



```sql
-- Display the top 5 students by enrollment date (most recent first) (MySQL)
SELECT student_id, first_name, last_name, enrollment_date
FROM students
ORDER BY enrollment_date DESC
LIMIT 5;
```

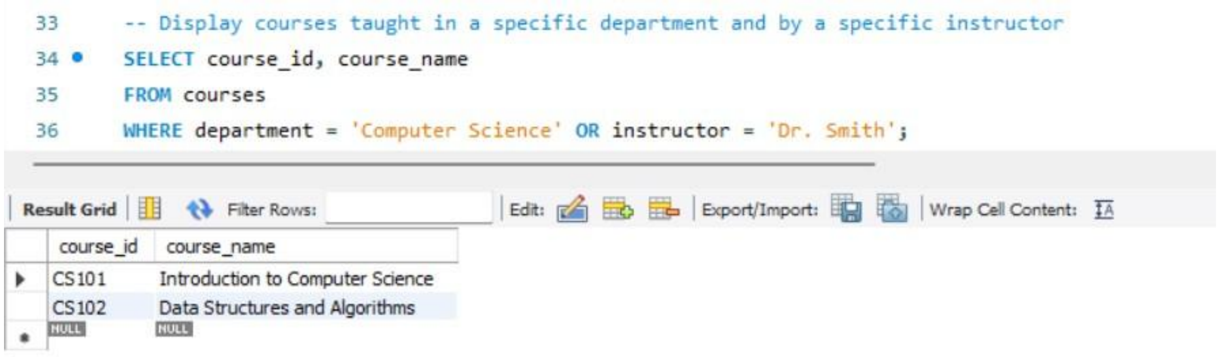
student_id	first_name	last_name	enrollment_date
11	Mirlinda	Xhaferi	2024-10-22
12	Faton	Hajdari	2024-10-22
16	Adelina	Mujaj	2024-10-22
5	Arben	Shala	2024-10-22
6	Jona	Nikaj	2024-10-22
NULL	NULL	NULL	NULL

Report 2: Use the SELECT command to display data with logical operators



```sql
-- Report 2: SELECT command to display data with logical operators
-- Display students who are active and studying in their 3rd year
SELECT student_id, first_name, last_name, study_year, status
FROM students
WHERE study_year = 3 AND status = 'active';
```

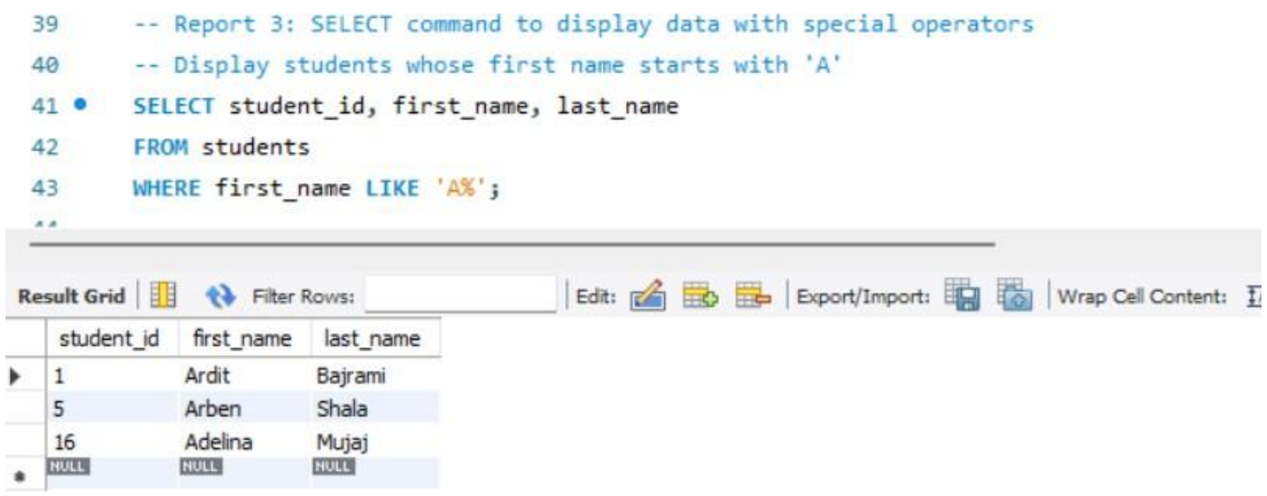
student_id	first_name	last_name	study_year	status
3	Driton	Rexhepi	3	active
8	Flora	Gashi	3	active
12	Faton	Hajdari	3	active
15	Kastriot	Berisha	3	active
NULL	NULL	NULL	NULL	NULL



```sql
-- Display courses taught in a specific department and by a specific instructor
SELECT course_id, course_name
FROM courses
WHERE department = 'Computer Science' OR instructor = 'Dr. Smith';
```

course_id	course_name
CS101	Introduction to Computer Science
CS102	Data Structures and Algorithms
NULL	NULL

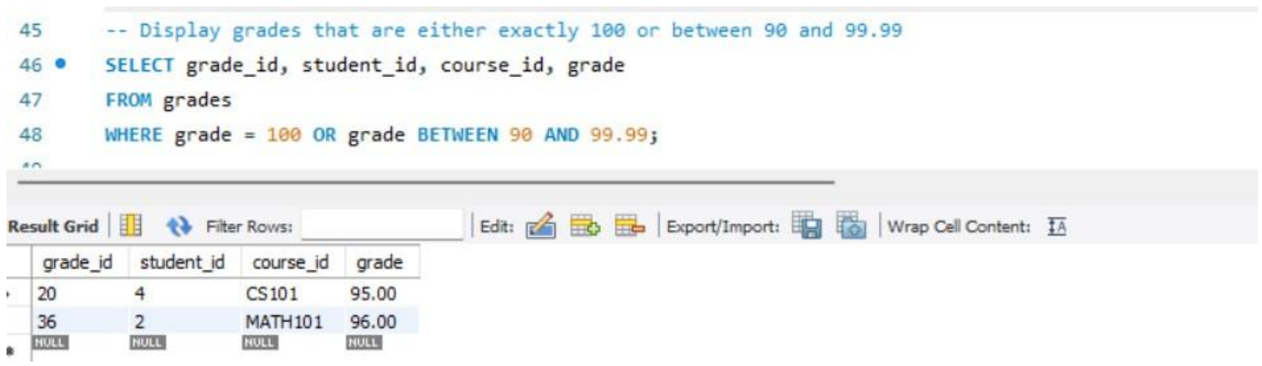
Report 3: Use the SELECT command to display data with special operators



```sql
-- Report 3: SELECT command to display data with special operators
-- Display students whose first name starts with 'A'
SELECT student_id, first_name, last_name
FROM students
WHERE first_name LIKE 'A%';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

student_id	first_name	last_name
1	Ardit	Bajrami
5	Arben	Shala
16	Adelina	Mujaj
NULL	NULL	NULL

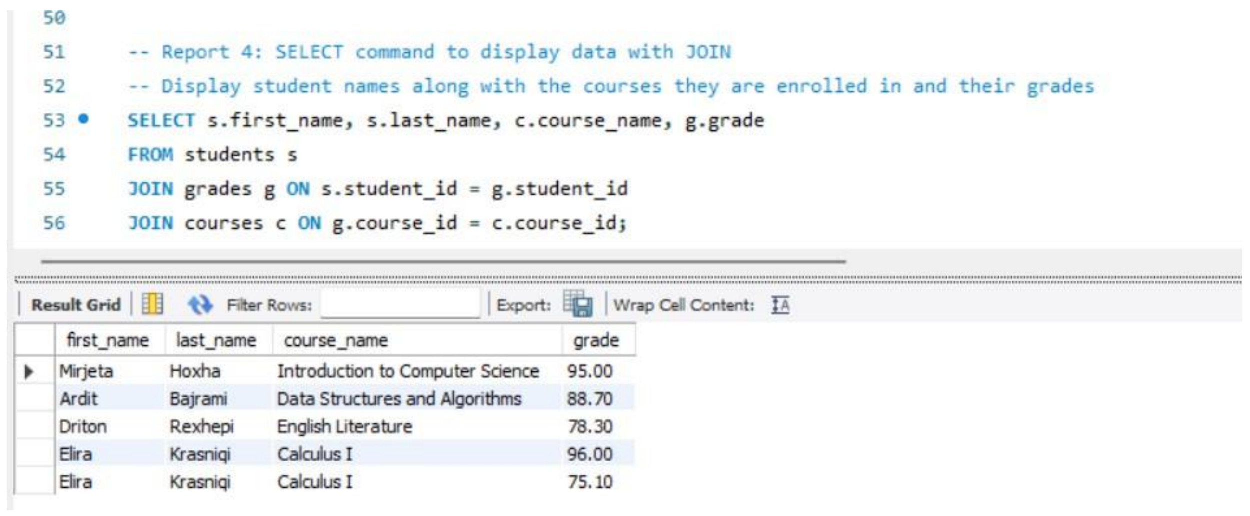


```sql
-- Display grades that are either exactly 100 or between 90 and 99.99
SELECT grade_id, student_id, course_id, grade
FROM grades
WHERE grade = 100 OR grade BETWEEN 90 AND 99.99;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

grade_id	student_id	course_id	grade
20	4	CS101	95.00
36	2	MATH101	96.00
NULL	NULL	NULL	NULL

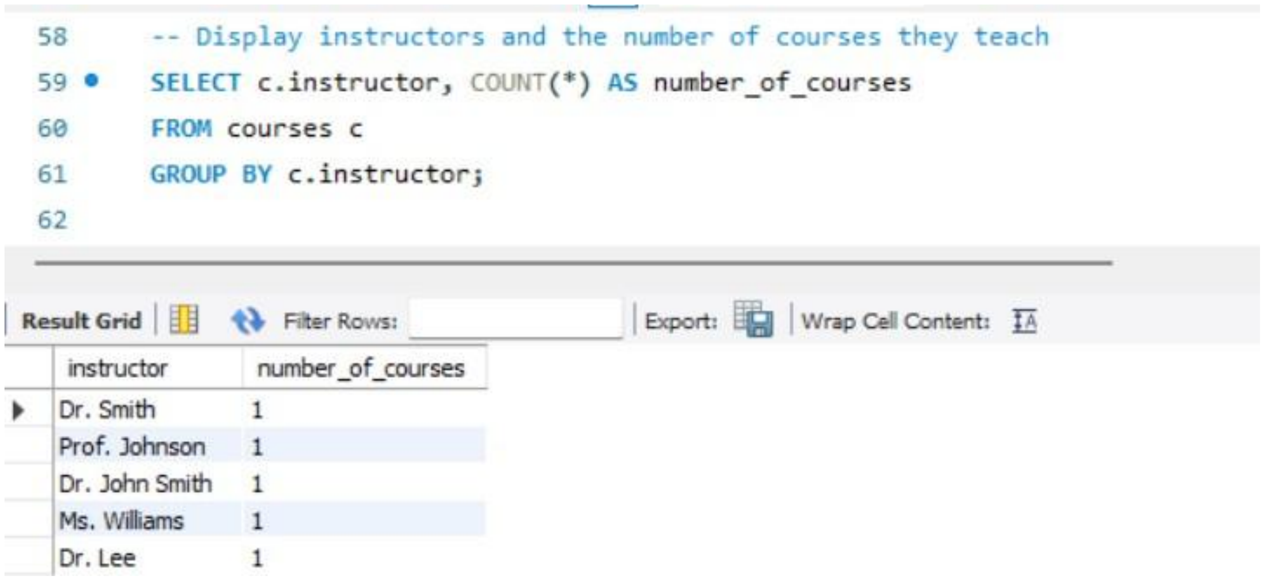
Report 4: Use the SELECT command to display data with JOIN



```sql

-- Report 4: SELECT command to display data with JOIN
-- Display student names along with the courses they are enrolled in and their grades
SELECT s.first_name, s.last_name, c.course_name, g.grade
FROM students s
JOIN grades g ON s.student_id = g.student_id
JOIN courses c ON g.course_id = c.course_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

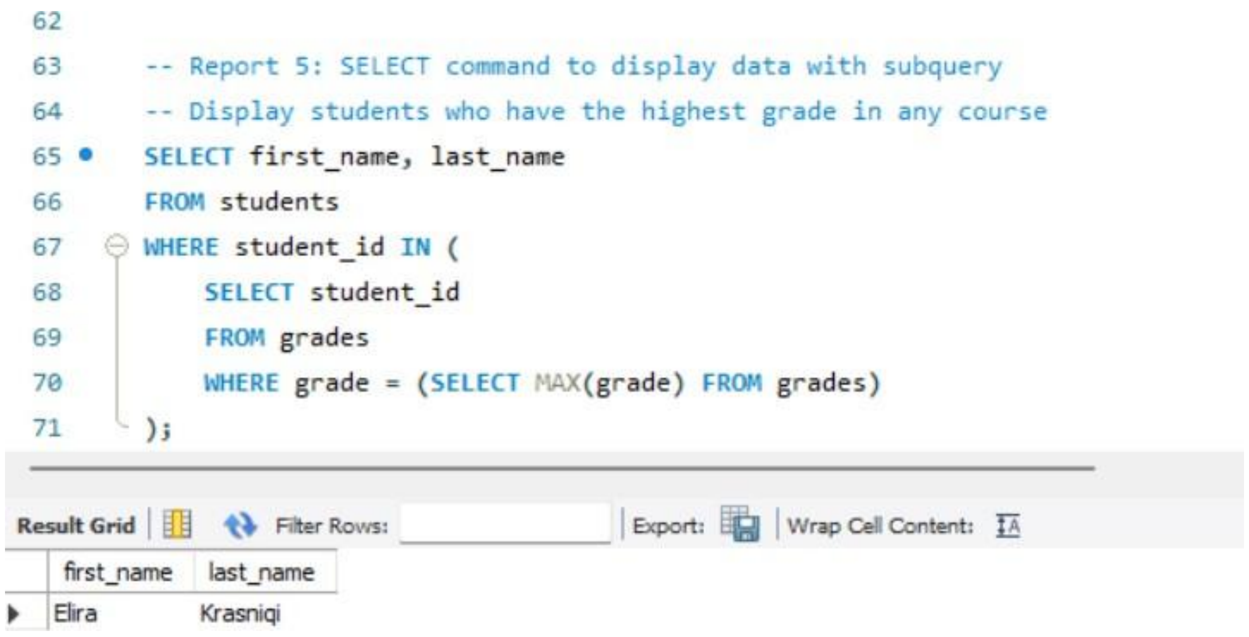
first_name	last_name	course_name	grade
Mirjeta	Hoxha	Introduction to Computer Science	95.00
Ardit	Bajrami	Data Structures and Algorithms	88.70
Driton	Rexhepi	English Literature	78.30
Elira	Krasniqi	Calculus I	96.00
Elira	Krasniqi	Calculus I	75.10



```
-- Display instructors and the number of courses they teach
SELECT c.instructor, COUNT(*) AS number_of_courses
FROM courses c
GROUP BY c.instructor;
```

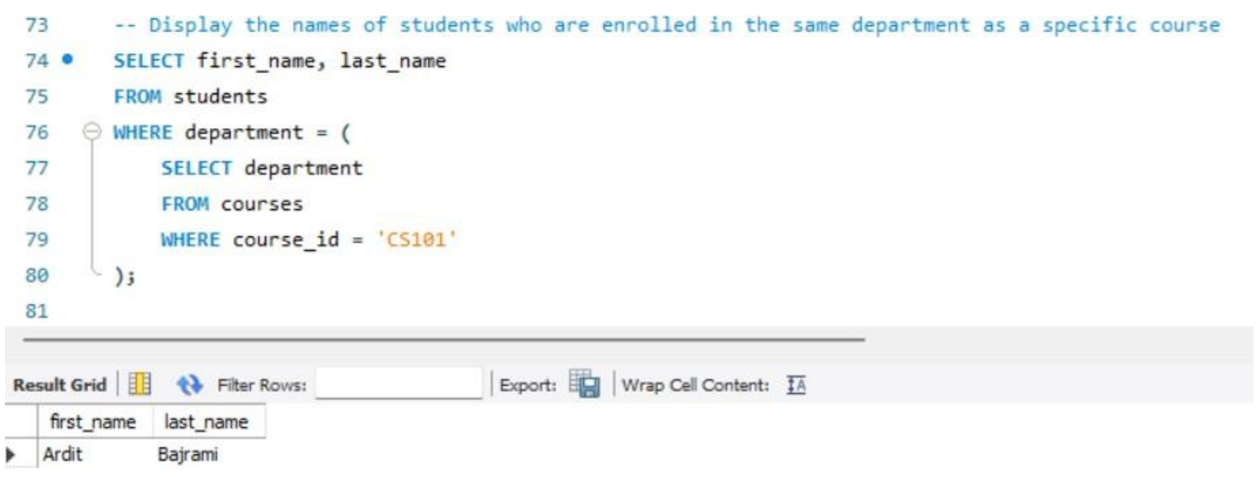
instructor	number_of_courses
Dr. Smith	1
Prof. Johnson	1
Dr. John Smith	1
Ms. Williams	1
Dr. Lee	1

Report 5: Use the SELECT command to display data with subquery



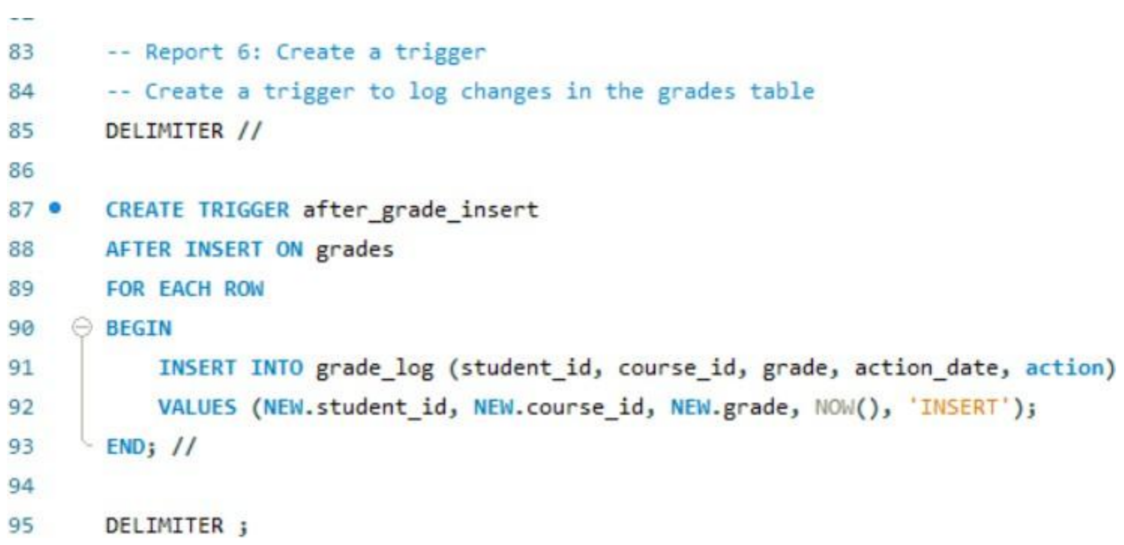
```
-- Report 5: SELECT command to display data with subquery
-- Display students who have the highest grade in any course
SELECT first_name, last_name
FROM students
WHERE student_id IN (
    SELECT student_id
    FROM grades
    WHERE grade = (SELECT MAX(grade) FROM grades)
);
```

first_name	last_name
Elira	Krasniqi

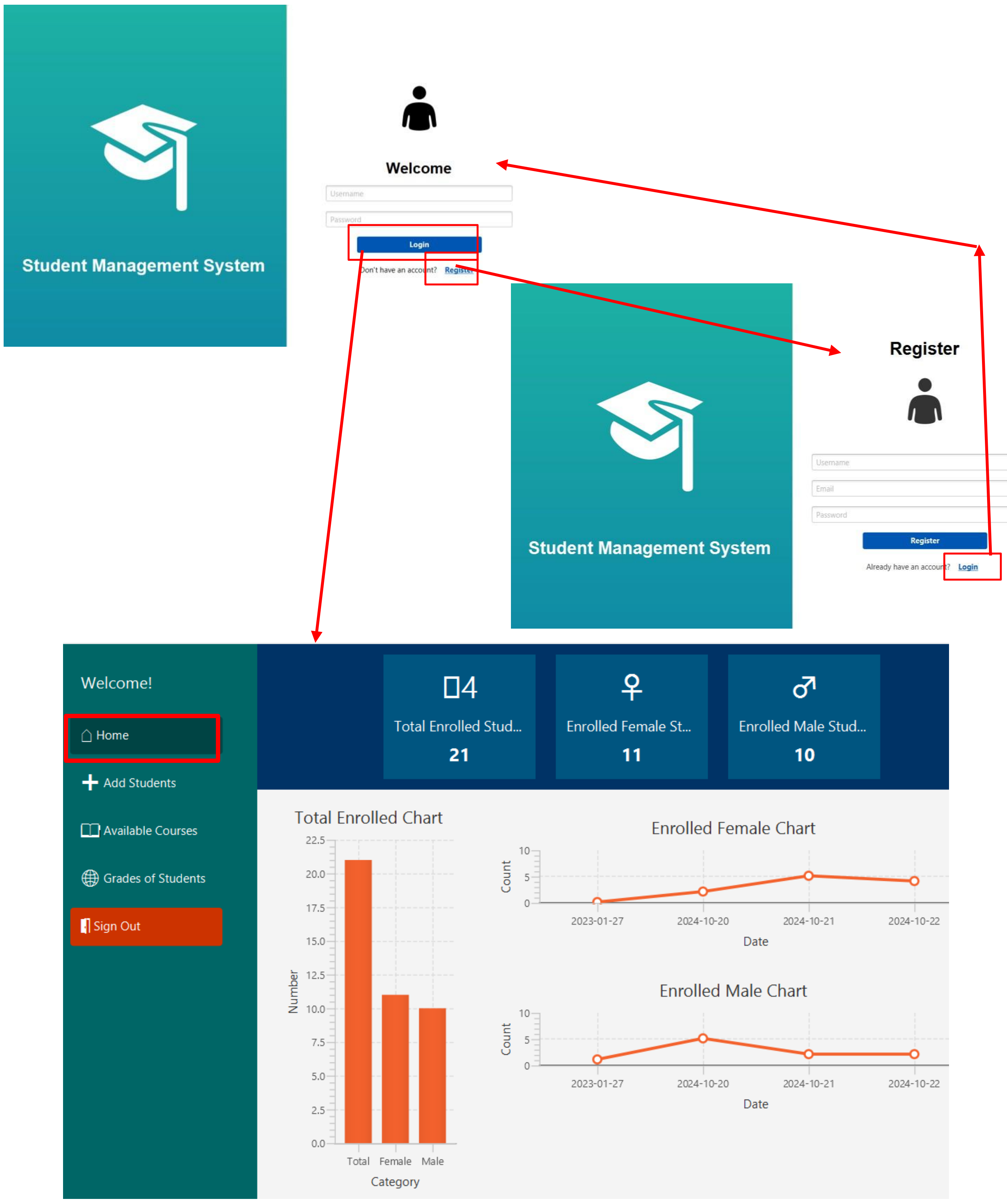

```
-- Display the names of students who are enrolled in the same department as a specific course
SELECT first_name, last_name
FROM students
WHERE department = (
    SELECT department
    FROM courses
    WHERE course_id = 'CS101'
);
```

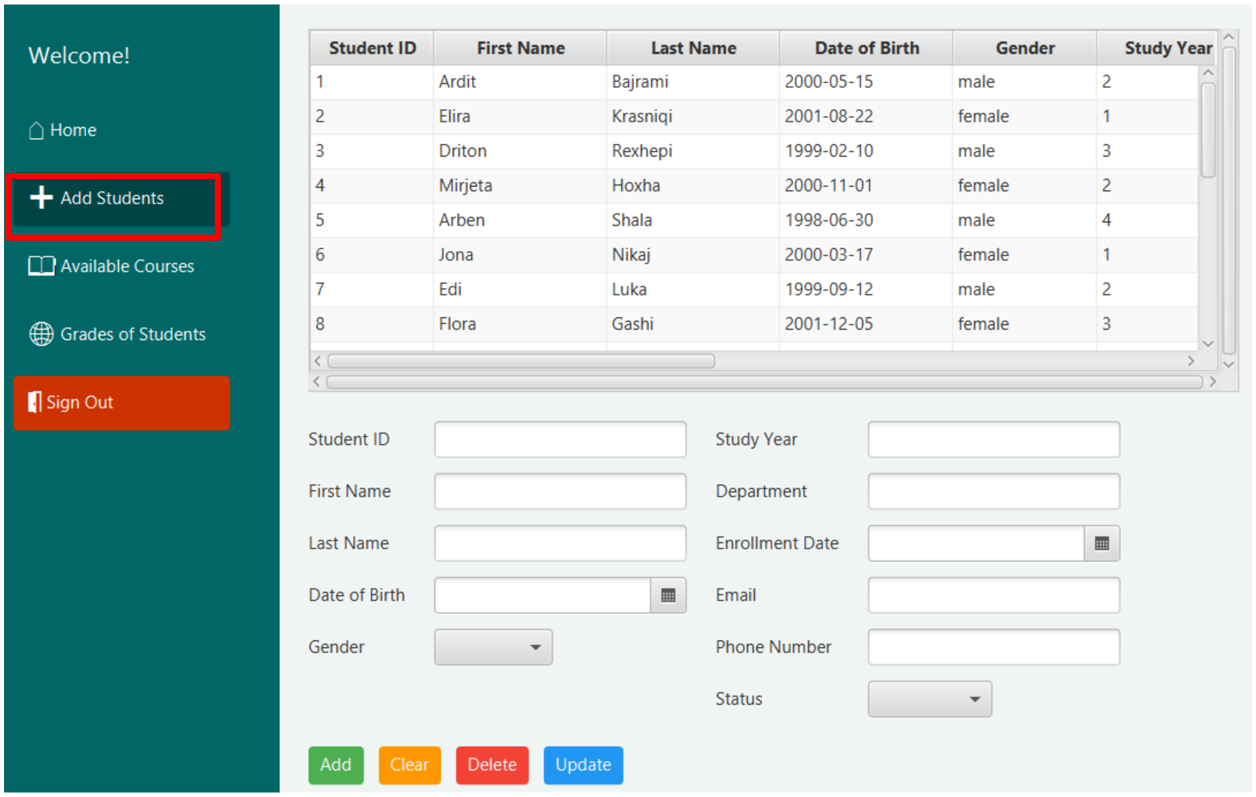
```
first_name  last_name
Ardit       Bajrami
```

Report 6: Create a trigger

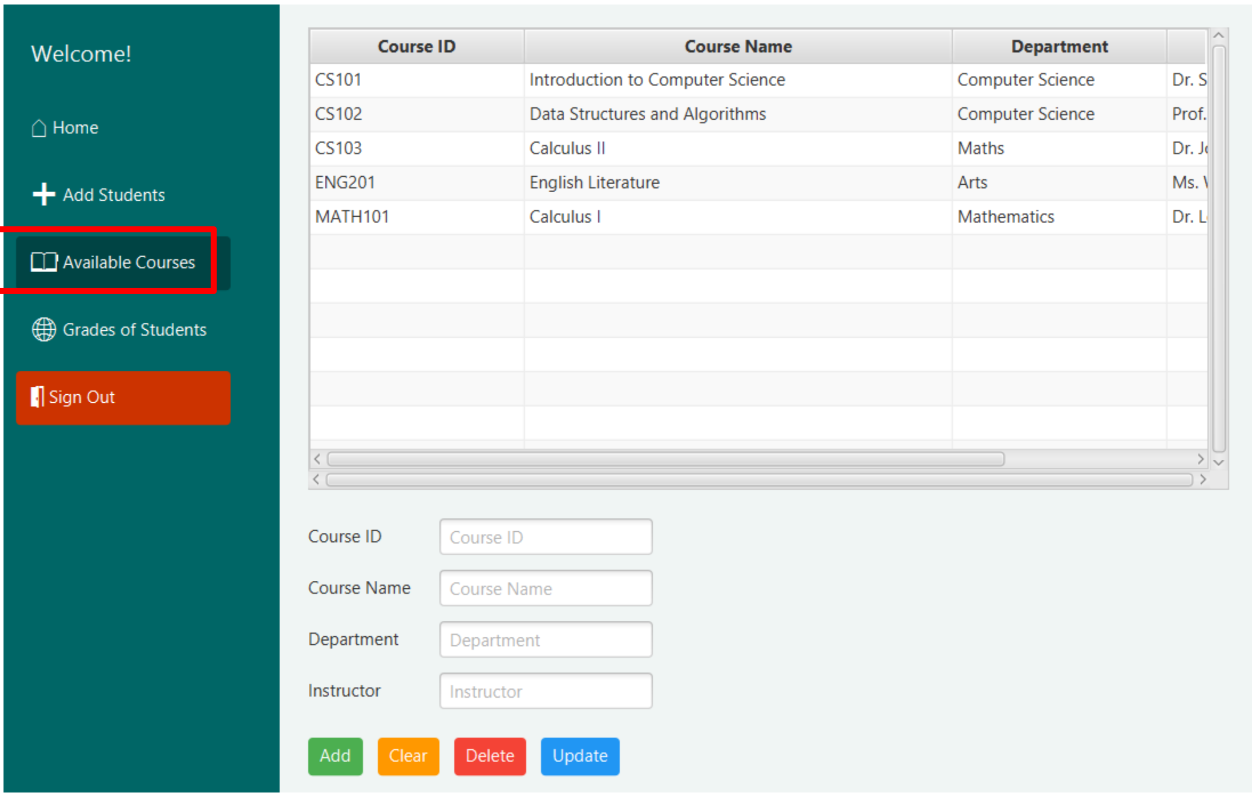

```
--
-- Report 6: Create a trigger
-- Create a trigger to log changes in the grades table
DELIMITER //

CREATE TRIGGER after_grade_insert
AFTER INSERT ON grades
FOR EACH ROW
BEGIN
    INSERT INTO grade_log (student_id, course_id, grade, action_date, action)
    VALUES (NEW.student_id, NEW.course_id, NEW.grade, NOW(), 'INSERT');
END; //

DELIMITER ;
```

3.3. Interface Development

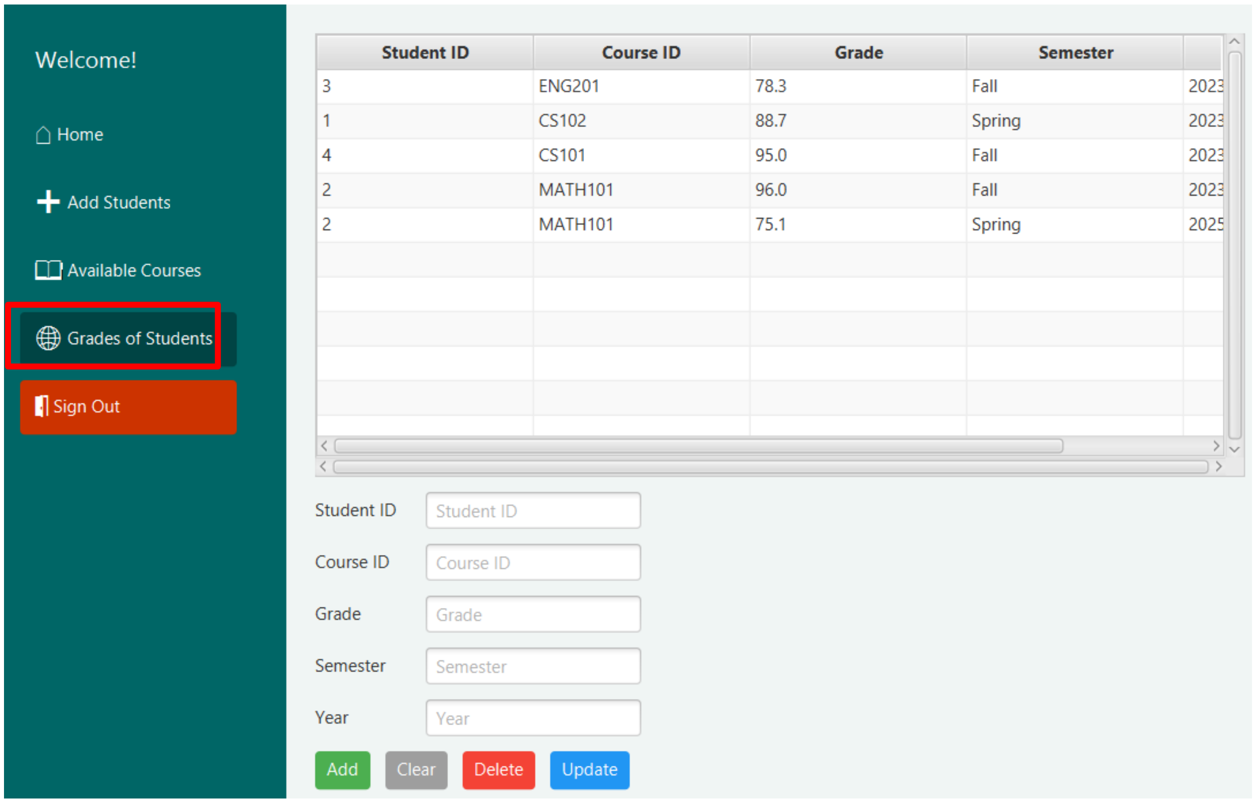




Student ID	First Name	Last Name	Date of Birth	Gender	Study Year
1	Ardit	Bajrami	2000-05-15	male	2
2	Elira	Krasniqi	2001-08-22	female	1
3	Driton	Rexhepi	1999-02-10	male	3
4	Mirjeta	Hoxha	2000-11-01	female	2
5	Arben	Shala	1998-06-30	male	4
6	Jona	Nikaj	2000-03-17	female	1
7	Edi	Luka	1999-09-12	male	2
8	Flora	Gashi	2001-12-05	female	3



Course ID	Course Name	Department	
CS101	Introduction to Computer Science	Computer Science	Dr. S
CS102	Data Structures and Algorithms	Computer Science	Prof.
CS103	Calculus II	Maths	Dr. J
ENG201	English Literature	Arts	Ms. \
MATH101	Calculus I	Mathematics	Dr. L



Student ID	Course ID	Grade	Semester	
3	ENG201	78.3	Fall	2023
1	CS102	88.7	Spring	2023
4	CS101	95.0	Fall	2023
2	MATH101	96.0	Fall	2023
2	MATH101	75.1	Spring	2025

Student ID	Student ID
Course ID	Course ID
Grade	Grade
Semester	Semester
Year	Year

5. Conclusion

The **Student Management System** is a robust solution for managing student records, courses, and grades. Its modular design, based on **Object-Oriented Programming (OOP)** principles, allows for efficient maintenance and future expansion. By integrating **JavaFX** for the front-end and **MySQL** for the back-end, the system ensures both user-friendly interactions and secure data handling.

Future improvements could include:

- **Mobile Compatibility:** Adapting the system for mobile devices to increase accessibility for students and administrators.
- **Additional Features:** Implementing advanced features such as course scheduling, notifications, and detailed reporting for better academic tracking.
- **Enhanced Security:** Strengthening security by adding features like multi-factor authentication (MFA) and encryption for sensitive data.

This system serves as a strong foundation for educational institutions seeking to streamline administrative processes and enhance student management.