“Your all-in-one campus enrollment hub”

Project: CAMPUSCLOUD

Date: 4th December, 2025

Product Title: **CampusCloud – Student Productivity & Campus Service Hub**

Authors:
Himanshu Sandhu
Iknoordeep Kaur
Jastan Singh
Mykhailo Halushko

Client: INFO 2312 – KPU

Document Ref: Your all-in-one campus enrollment hub

Version No: 6.7

1. PRODUCT DESCRIPTION HISTORY

1.1. Revision History

Revision date	Previous revision date	Summary of Changes	Changes marked
23/11/2025	-	Changes made to Product Description and Identifier Section	N/A
24/11/2025	23/11/2025	Changes made to Composition, Relationships and Relational Schema	N/A
26/11/2025	24/11/2025	Changes made to Composition, Relationships and Relational Schema	N/A
29/11/2025	26/11/2025	Fixing Grammatical Errors and Instructions to use	N/A
01/12/2025	29/11/2025	Changes made to Composition, Relationships and Relational Schema	N/A
02/12/2025	01/12/2025	Changes made to Table of Contents	N/A
04/12/2025	02/12/2025	Changes made to Composition, Relationships and Relational Schema	N/A

1.2. Distribution

This document has been distributed to:

Name	Title	Date of Issue	Version
Himanshu Sandhu	Manager	23 /11/ 2025	6.1
Iknoordeep Kaur	Project Designer	23 /11 /2025	6.1
Jastan Singh	Communication Officer, Project Analyst	24 / 11/ 2025	6.2
Mykhailo Halushko	Project Analyst	26 / 11/ 2025	6.3

Table of Contents

2. OVERVIEW

2.1. Identifier

INFO2312 - Section 10 - Team 6

2.2. Title

CampusCloud

2.3. Purpose

Picture a university where thousands of students and professors rely on a single platform to manage their academic lives. CampusCloud was designed with this vision in mind—a product whose purpose is to centralize academic data, streamline communication, and ensure accuracy in every record. Its function goes beyond storing information; it enables real-time updates, secure access, and personalized dashboards for different roles. Students use it to track grades and schedules, professors to manage courses and share announcements, and administrators to maintain integrity across the institution. To meet these expectations, the system demands a high level of quality—robust enough to handle thousands of records, scalable for future growth, and secure against unauthorized access.

Every feature, from messaging to grade validation, reflects careful consideration of size, complexity, and reliability, ensuring that CampusCloud is not just functional but dependable in the dynamic academic environment. The messaging system, for instance, isn't just a chat tool—it's a real-time communication channel designed to handle thousands of simultaneous interactions without lag, ensuring professors can share announcements instantly and students never miss critical updates. Grade validation goes beyond simple input checks; it enforces strict constraints at both the interface and database level, preventing errors that could compromise academic integrity. The scheduling module accounts for overlapping classes and professor availability, while the notification engine is optimized for speed and scalability, delivering alerts across devices in seconds. Each component was tested for robustness under heavy loads, ensuring that even during peak enrollment periods or exam seasons, CampusCloud remains stable and responsive. This careful engineering transforms the system from a basic database into a dependable academic ecosystem—one that thrives under complexity and guarantees accuracy, security, and performance at every level.

Developing CampusCloud as a conceptual academic database required a structured approach to ensure clarity, accuracy, and compliance with project goals. The process was divided into three key stages: Production, Review, and Approval, each with specific activities and quality benchmarks.

Production: Tasks Required

- **Requirements Analysis**
  Define the scope, goals, risks, and functional capabilities based on academic

workflows.

- **Design**
  Create the **E-R diagram**, establish entity relationships, and normalize the schema to minimize redundancy.
- **Schema Documentation**
  Prepare **DDL scripts** with primary keys, foreign keys, and constraints to enforce data integrity.
- **Sample Queries**
  Formulate SQL queries for common use cases (e.g., retrieving grades, listing enrolled courses, generating reports).
- **User Instructions**
  Document query purposes, syntax, and expected results for clarity and usability.

Review: Quality Checks

- Validate the E-R diagram for correct cardinalities and relationships.
- Ensure schema normalization (at least 3NF) and proper constraints for integrity.
- Test sample queries conceptually to confirm they return accurate and meaningful results.
- Review documentation for completeness, clarity, and alignment with functional goals.

Approval: Criteria for Sign-Off

- **Quality**
  Schema is normalized, relationships are correct, and constraints enforce integrity.
- **Functionality**
  Queries cover essential academic workflows (enrollments, grading, assignments, reporting).
- **Compliance**
  Design reflects role-based access, data sensitivity considerations, and scalability plans.
- **Documentation**
  Instructions for use are clear, with query explanations and expected outputs.

This structured approach ensured that CampusCloud's conceptual design met academic standards and was ready for presentation as a robust, future-ready solution.

CampusCloud was not just an idea—it was a collaborative effort shaped by a team where each role carried unique responsibilities and skills. At the center was our **Project Manager**, the strategist who ensured that timelines were realistic, tasks were distributed fairly, and progress stayed aligned with the project charter. Their organizational skills and ability to manage competing priorities were crucial, especially since all team members balanced coursework and part-time jobs alongside this project.

The **Designer** brought creativity and technical precision, transforming abstract concepts into tangible models. They crafted the **E-R diagram**, defined relationships, and ensured the database structure adhered to normalization principles. Their understanding of database design and visual representation made the foundation strong. Supporting communication and coordination was the **Communication Officer**, who acted as the glue

between team members—documenting decisions, scheduling meetings, and ensuring clarity in every discussion. Finally, the **Project Analyst** played the role of the critical thinker, analyzing requirements, identifying risks, and validating that the functional capabilities matched institutional needs.

While we didn't build the system physically, these roles collectively produced the conceptual design, reviewed its accuracy, and approved its readiness for presentation.MySQL was selected as the DBMS for CampusCloud because it meets the core requirements of our project: **data integrity, scalability, security, and cloud compatibility**. Its support for **ACID transactions** ensures accurate and consistent academic records, which is critical for our goal of maintaining integrity in student details, courses, and grades. MySQL's ability to handle large datasets efficiently through **indexing and query optimization** aligns with our scalability objective, allowing the system to manage thousands of records over multiple academic years without performance issues.

Additionally, MySQL integrates seamlessly with **cloud platforms**, enabling automated backups and disaster recovery—key deliverables for CampusCloud. Its **role-based access control** supports personalized dashboards for students and professors, ensuring secure and relevant data access. Compared to alternatives like Oracle or SQL Server, MySQL offers a cost-effective, widely adopted solution with strong community support, making it ideal for academic environments. In short, MySQL provides the reliability, flexibility, cost-effectiveness and future-readiness needed to achieve CampusCloud's vision of a centralized, secure, and scalable academic database.

2.4. Composition

The ER diagram for Campus Cloud defines the core structure of the academic database. THe entities and their relationships were selected to accurately model the key actors, objects and processes within a university environment.

Entities

- **Faculty:** It represents various academic departments or schools within the university. It is a central entity that oversees and employs Professors. A faculty may offer multiple Programs.
- **Program:** Represents the degree or certificate programs offered by the university.
- **Course:** Represents individual classes offered within a Program.
- **Professor:** Represents the teaching staff. Each Professor may be associated with a single or multiple faculties.
- **Class:** Represents a specific section of a Course in a given semester, including the schedule, location and the specific Professor teaching it.
- **Student:** Represents a student enrolled at the institution and stores their information.

Relationships:

- Faculty -> Program: A Faculty can offer many Programs but a Program can only belong to one Faculty.
- Program -> Course: A program includes several courses and the same course may belong to multiple Programs. Some courses do not belong to any program.
- Course -> Class: A course may have multiple or no class in a given semester but only one course can be taught in a given class.
- Professor -> Class: A professor may teach single or multiple classes or no class in a semester but a class can only be taught by one professor.
- Program -> Professor: A program may be assigned one or more than one professor, however, a professor can be part of only one program.
- Student -> Program: A student enrolls in one program for a given semester but a program may include one or many students.
- Student -> Class: A student can take many classes and classes can have multiple students.

Relational Schema

Table 1: *Faculty Table*: Stores Information about academic faculties or departments within the university. Each faculty represents a major academic division and is managed by a dean.

Attribute	Datatype	Primary Key	Foreign Key	Referential Integrity				Additional Comments
				Parent		On Update	On Delete	
				Table	Attribute			
facultyID	INT	YES	NO	-	-	-	-	Increments as new faculty is employed
deanName	VARCHAR (100)	NO	NO	-	-	-	-	Name of the Dean
facultyName	VARCHAR (50)	NO	NO	-	-	-	-	Name of the Faculty
office	VARCHAR (20)	NO	NO	-	-	-	-	Office Location

Table 2: *Program Table*: Contains details about academic programs offered by the university. Each program belongs to a specific faculty and has a specific duration.

Attribute	Datatype	Primary Key	Foreign Key	Referential Integrity				Additional Comments
				Parent		On Update	On Delete	
				Table	Attribute			
programID	INT	YES	NO	-	-	-	-	Increments as more programs are introduced
facultyID	INT	NO	YES	Faculty	facultyID	CASCADE	SET NULL	If Faculty is deleted, program stays but facultyID becomes null
credentialType	VARCHAR (50)	NO	NO	-	-	-	-	Eg: Diploma, Bachelors, PhD
durationYears	INT	NO	NO	-	-	-	-	Duration of the Program
programName	VARCHAR (50)	NO	NO	-	-	-	-	Name of the Program

Table 3: *Student Table*: Stores personal and academic information for all enrolled students. Each student is registered in one academic program and contains demographic and contact details.

Attribute	Datatype	Primary Key	Foreign Key	Referential Integrity				Additional Comments
				Parent		On Update	On Delete	
				Table	Attribute			
studentID	INT	YES	NO	-	-	-	-	Increments as new students enroll
programID	INT	NO	YES	Program	programID	CASCADE	SET NULL	If program is deleted, student stays but programID becomes null
address	VARCHAR (200)	NO	NO	-	-	-	-	Student's address
DOB	DATETIME	NO	NO	-	-	-	-	Date of birth
email	VARCHAR (100)	NO	NO	-	-	-	-	Email address
enrollmenDetail	DATETIME	NO	NO	-	-	-	-	Date of enrollment
fNAME	VARCHAR (50)	NO	NO	-	-	-	-	First name
lNAME	VARCHAR (50)	NO	NO	-	-	-	-	Last Name
Phone	VARCHAR (50)	NO	NO	-	-	-	-	Contact number

Table 4: *Professor Table*: Stores information about teaching faculty members. Each professor is associated with a faculty and includes their academic and contact details.

Attribute	Datatype	Primary Key	Foreign Key	Referential Integrity				Additional Comments
				Parent		On Update	On Delete	
				Table	Attribute			
profID	INT	YES	NO	-	-	-	-	Increments as new professors are employed
programID	INT	NO	YES	Program	ProgramID	CASCADE	SET NULL	If Program is deleted, professor stays and proramID becomes null
address	VARCHAR (200)	NO	NO	-	-	-	-	Professor's address
DOB	DATETIME	NO	NO	-	-	-	-	Date of birth
email	VARCHAR (100)	NO	NO	-	-	-	-	Email address
fNAME	VARCHAR (50)	NO	NO	-	-	-	-	First name
lNAME	VARCHAR (50)	NO	NO	-	-	-	-	Last Name
officeHours	VARCHAR (100)	NO	NO	-	-	-	-	Office hours schedule

Table 5: *Course Table*: Stores catalog information for all courses offered by the university. Each course belongs to a program and a faculty, defining its academic properties.

Attribute	Datatype	Primary Key	Foreign Key	Referential Integrity				Additional Comments
				Parent		On Update	On Delete	
				Table	Attribute			
courseID	INT	YES	NO	-	-	-	-	Increments as new courses are added
programID	INT	NO	YES	Program	programID	CASCADE	CASCADE	If program is deleted, course is also deleted
courseCode	VARCHAR (20)	NO	NO	-	-	-	-	Course Code(e.g. INFO 2312)
courseNAME	VARCHAR (100)	NO	NO	-	-	-	-	Name of the Course
credits	INT	NO	NO	-	-	-	-	Credit Hours
description	TEXT	NO	NO	-	-	-	-	Course Description

Table 6: *Class Table*: Represents scheduled instances of course offered in a specific term/semester. Each class is an offering of a course taught by a professor at a particular location or time.

Attribute	Datatype	Primary Key	Foreign Key	Referential Integrity				Additional Comments
				Parent		On Update	On Delete	
				Table	Attribute			
classID	INT	YES	NO	-	-	-	-	Increments as new classes are added
courseID	INT	NO	YES	Course	courseID	CASCADE	CASCADE	If Course is deleted, class is also deleted
profID	INT	NO	YES	Professor	profID	CASCADE	SEt NULL	If Professor is deleted, class stays but profID becomes null
capacity	INT	NO	NO	-	-	-	-	Maximum number of students
location	VARCHAR (50)	NO	NO	-	-	-	-	Classroom Location
scheduleDay	VARCHAR (20)	NO	NO	-	-	-	-	Day(s) of the week
scheduleTime	TIME	NO	NO	-	-	-	-	Class Time
lNAME	VARCHAR (50)	NO	NO	-	-	-	-	Last Name
Phone	VARCHAR (20)	NO	NO	-	-	-	-	Contact number

Table 7: *Class Evaluation Table*: Records student evolutions and grades for specific class offerings. This table captures student performance and feedback for each class they attend.

Attribute	Datatype	Primary Key	Foreign Key	Referential Integrity				Additional Comments
				Parent		On Update	On Delete	
				Table	Attribute			
studentID	INT	YES	YES	Student	studentID	CASCADE	CASCADE	If student is deleted, evaluation is also deleted
classID	INT	YES	YES	Class	classID	CASCADE	CASCADE	If class is deleted, evaluation is also deleted
feedback	TEXT	NO	NO	-	-	-	-	Student feedback
grade	DECIMAL(5,2)	NO	NO	-	-	-	-	Grade Received(0-100)

3. USING THE PRODUCT

3.1. Special considerations

CampusCloud is designed to manage sensitive academic data and streamline communication, but its handling requires careful attention to several critical factors. First and foremost is **data sensitivity**. Student records, grades, and faculty details are confidential, so strict **role-based access control** and encryption must be enforced to prevent unauthorized access. Mismanagement here could compromise academic integrity and privacy.

Equally important is **role-based access**. The system is built to provide personalized dashboards for students, professors, and administrators, meaning permissions must be configured accurately. A single error in access settings could expose sensitive data or restrict essential functionality.

Another consideration is **data integrity during updates**. CampusCloud supports real-time updates for grades, schedules, and announcements, which requires careful transaction handling. Improper execution could lead to inconsistencies, such as incorrect grades or duplicate enrollments.

Backup and disaster recovery is another critical area. Academic data is irreplaceable, so automated cloud backups and tested restoration procedures must be in place to safeguard against outages, accidental deletions, or system failures.

As the system grows, **scalability and performance optimization** become vital. Poor indexing or normalization could lead to slow queries, especially during peak enrollment periods. Regular performance checks and database tuning are essential to maintain efficiency.

Integration with existing campus tools introduces **compatibility challenges**. Data mapping and synchronization must be handled carefully to avoid duplication or conflicts between systems.

Finally, **thorough testing before deployment** cannot be overlooked. Usability testing ensures a smooth experience for students and professors, while security testing prevents vulnerabilities that could compromise sensitive information.

These considerations are not just technical—they are fundamental to ensuring CampusCloud remains secure, reliable, and efficient throughout its lifecycle.

3.2. Instructions for use

CampusCloud’s database is designed to provide quick access to academic information through structured SQL queries. Below are step-by-step examples of how users can retrieve meaningful data using the schema represented in the E-R diagram.

3.2.1. Top Five Students in Class 401

Purpose:

To identify the top-performing students in a specific class (Class ID = 401) based on grades.

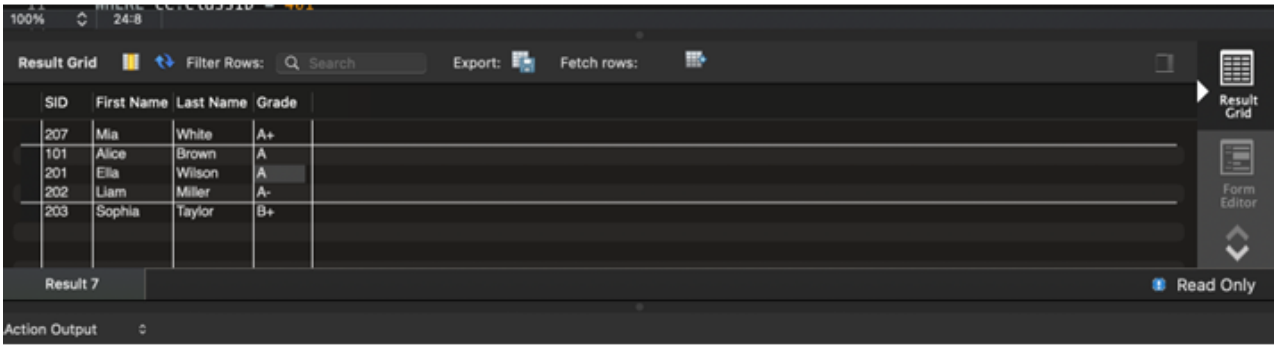
```sql
-- Top five students in class 401

SELECT
  s.studentID AS `SID`,
    s.fName AS `First Name`,
    s.lName AS `Last Name`,
    ce.grade AS `Grade`
FROM ClassEvaluation ce
JOIN Student s
    ON ce.studentID = s.studentID
WHERE ce.classID = 401
ORDER BY
    CASE ce.grade
        WHEN 'A+' THEN 12
        WHEN 'A'  THEN 11
        WHEN 'A-' THEN 10
        WHEN 'B+' THEN 9
        WHEN 'B'  THEN 8
        WHEN 'B-' THEN 7
        WHEN 'C+' THEN 6
        WHEN 'C'  THEN 5
        WHEN 'C-' THEN 4
        WHEN 'D+' THEN 3
        WHEN 'D'  THEN 2
        WHEN 'D-' THEN 1
        WHEN 'F'  THEN 0
        ELSE -1
    END DESC
LIMIT 5;
```

Explanation:

- JOIN links Student and Class entities to access student names and grades.
- WHERE ce.classID = 401 filters for the specific class.
- ORDER BY ce.grade DESC sorts students by grade in descending order.
- LIMIT 5 returns only the top five students.

Result:

A table listing the names and grades of the top five students in Class 401.



3.2.2. List Faculties with the Number of Professors in Them

Purpose:

To generate a summary showing each faculty and the count of professors assigned to it.

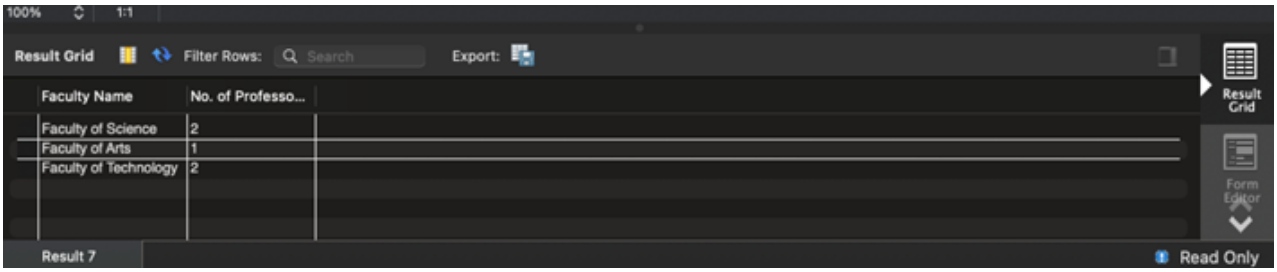
```sql
-- List faculties with the number of professors in them

SELECT
  f.facultyName AS `Faculty Name`,
  COUNT(p.profID) AS `No. of Professors`
FROM Professor p
JOIN Faculty f ON p.facultyID = f.facultyID
GROUP BY p.facultyID;
```

Explanation:

- JOIN connects Faculty and Professor entities.
- COUNT(p.profID) calculates the number of professors per faculty.
- GROUP BY p.facultyID organizes results by faculty name.

Result:

A table showing each faculty and the total number of professors in that faculty.



3.2.3. Select All Classes the Student is Taking

Purpose:

To display all classes a specific student is enrolled in.

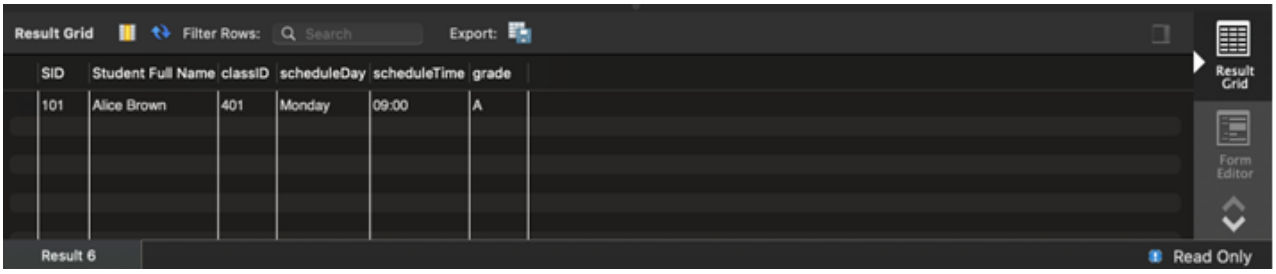
```sql
-- Select all classes the student is taking

SELECT
  ce.studentID AS `SID`,
  CONCAT(s.fName, " ", s.lName) AS `Student Full Name`,
  c.classID, c.scheduleDay, c.scheduleTime, ce.grade
FROM Class c
JOIN ClassEvaluation ce ON c.classID = ce.classID
JOIN Student s ON ce.studentID = s.studentID
WHERE ce.studentID = 101;
```

Explanation:

- Links Student and Class entities to retrieve class details for a given student.
- Filters by ce.studentID = 101 to show only that student's classes.

Result:

A list of class IDs, schedule days, and times for the student.



3.2.4. GPA Calculator for Student

Purpose:

To calculate the GPA for a student based on grades and course credits.

```sql
-- GPA Calculator for students

SELECT
    s.studentID AS `SID`,
    CONCAT(s.fName, " ", s.lName) AS `Student Name`,
    AVG(g.gpaValue) AS GPA
FROM (
    SELECT
        studentID,
        CASE grade
            WHEN 'A+' THEN 4.33
            WHEN 'A'  THEN 4.00
            WHEN 'A-' THEN 3.67
            WHEN 'B+' THEN 3.33
            WHEN 'B'  THEN 3.00
            WHEN 'B-' THEN 2.67
            WHEN 'C+' THEN 2.33
            WHEN 'C'  THEN 2.00
            WHEN 'C-' THEN 1.67
            WHEN 'D+' THEN 1.33
            WHEN 'D'  THEN 1.00
            WHEN 'D-' THEN 0.67
            WHEN 'F'  THEN 0.00
            ELSE NULL
        END AS gpaValue
    FROM ClassEvaluation
) g
JOIN Student s
    ON s.studentID = g.studentID
GROUP BY `SID`;
```

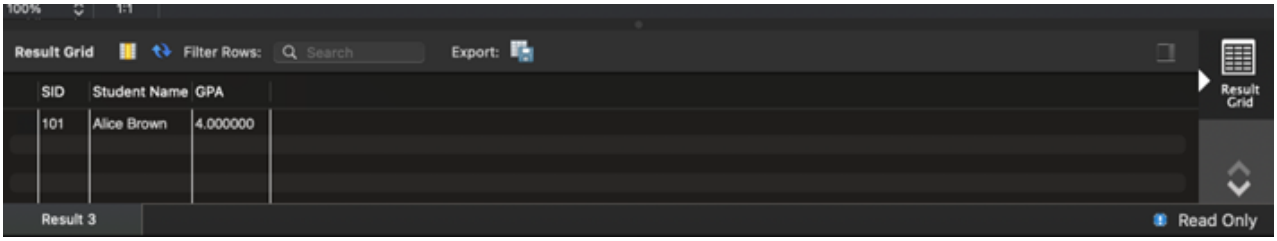
Explanation:

- s.studentID AS SID: Displays the student's unique ID.
- CONCAT(s.fName, " ", s.lName): Combines first and last name into a single column called Student Name.
- AVG(g.gpaValue): Calculates the average GPA value for each student.
- JOIN Student s: Links the GPA values to the student's personal details (name and ID).

- GROUP BY SID: Groups all GPA values for each student so the AVG() function can compute the overall GPA.

Result:

A single value representing the student's GPA.



Conclusion:

The queries demonstrated in this section highlight how CampusCloud's database design supports essential academic operations through structured SQL commands. From identifying the top five students in a class to generating faculty-level summaries, retrieving all classes for a student, and calculating GPA, each query reflects the system's ability to deliver accurate, meaningful insights for different user roles. These examples show that CampusCloud is not only capable of storing academic data but also of transforming it into actionable information for students, professors, and administrators. By combining clear syntax with logical relationships from the E-R diagram, the system ensures data integrity, usability, and scalability—core principles of our project vision. Ultimately, these queries illustrate how CampusCloud can streamline academic processes and enhance decision-making in a modern educational environment.