CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

CASE STUDY REPORT ON PARTICULAR CASE STUDY

Program Name: BCA

Subject Name/Code: Database Management System (23CAT-251)

Submitted by:

Name: Tanupreet Singh

UID: 23BCA10468

Section: 23BCA4-B

Submitted to:

Name: Mr. Arvinder Singh

Designation:

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

ABSTRACT

- **Introduction:**
- **Technique:**
- **System Configuration:**
- **INPUT:**
- **ER DIAGRAM:**
- **TABLE REALTION:**
- **TABULAR FORMAT:**
- **TABLE CREATION:**
- **SQL QUERIES WITH OUTPUT (at least 10 to 15):**
- **SUMMARY:**
- **CONCLUSION:**

University Management System Using SQL

1. Introduction

The University Management System (UMS) is designed to manage academic, administrative, and enrolment data for a university. This case study covers the implementation of a relational database system using SQL to manage colleges, departments, courses, students, instructors, classrooms, and their inter-relationships.

2. Technique Used

- **Relational Database Design**
- **Normalization**
- **Entity-Relationship (ER) Modelling**
- **SQL: DDL (Data Definition Language), DML (Data Manipulation Language), DCL (Data Control Language)**
- **Views and Joins**
- **Aggregation Functions**

3. System Configuration

- **Database System**: MySQL
- **Version**: 8.0+
- **User Privileges**: GRANT, SELECT, UPDATE, DELETE
- **Users**: `paul1`, `constantin1`, `marius1`
- **Development Environment**: MySQL Workbench / Command Line Interface
- **OS Compatibility**: Windows/Linux/MacOS

4. Input Description

The system accepts input through SQL commands:

- College & Department details
- Course structure
- Student & Instructor information
- Classroom & Section scheduling
- Enrolment and grades

5. Tabular Representation & ER Diagram Description

Though no diagram is shown here, the ER diagram would feature:

- **Entities**: College, Department, Course, Instructor, Student, Classroom, Section, Level
- **Relationships**:
  - One-to-Many between Department and Course, Instructor, Student
  - Many-to-Many between Student and Section (via `Student_Section`)
  - Many-to-Many between College and Department (via `College_Department`)
  - One-to-Many between Course and Section
  - One-to-One between Section and Classroom

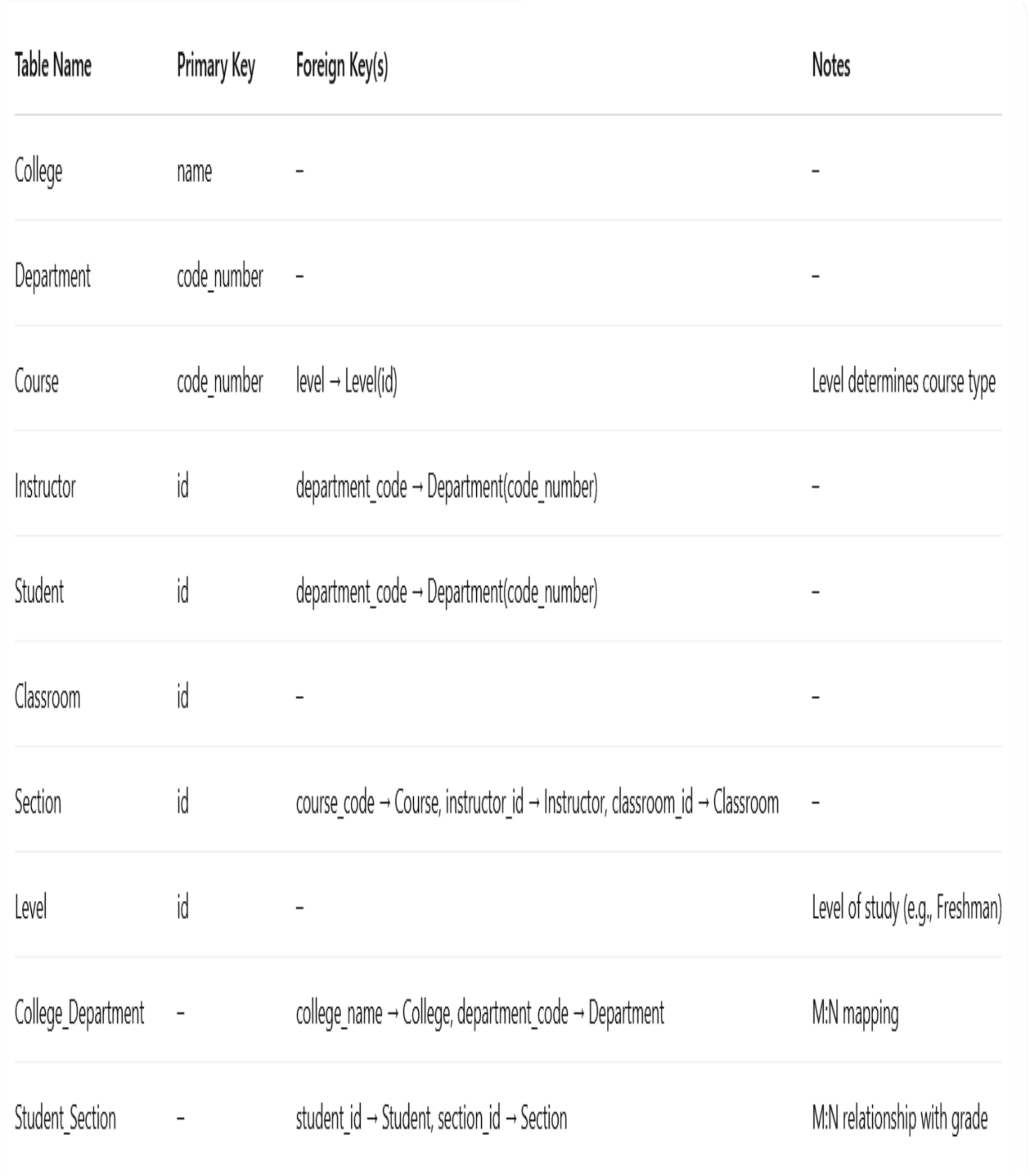
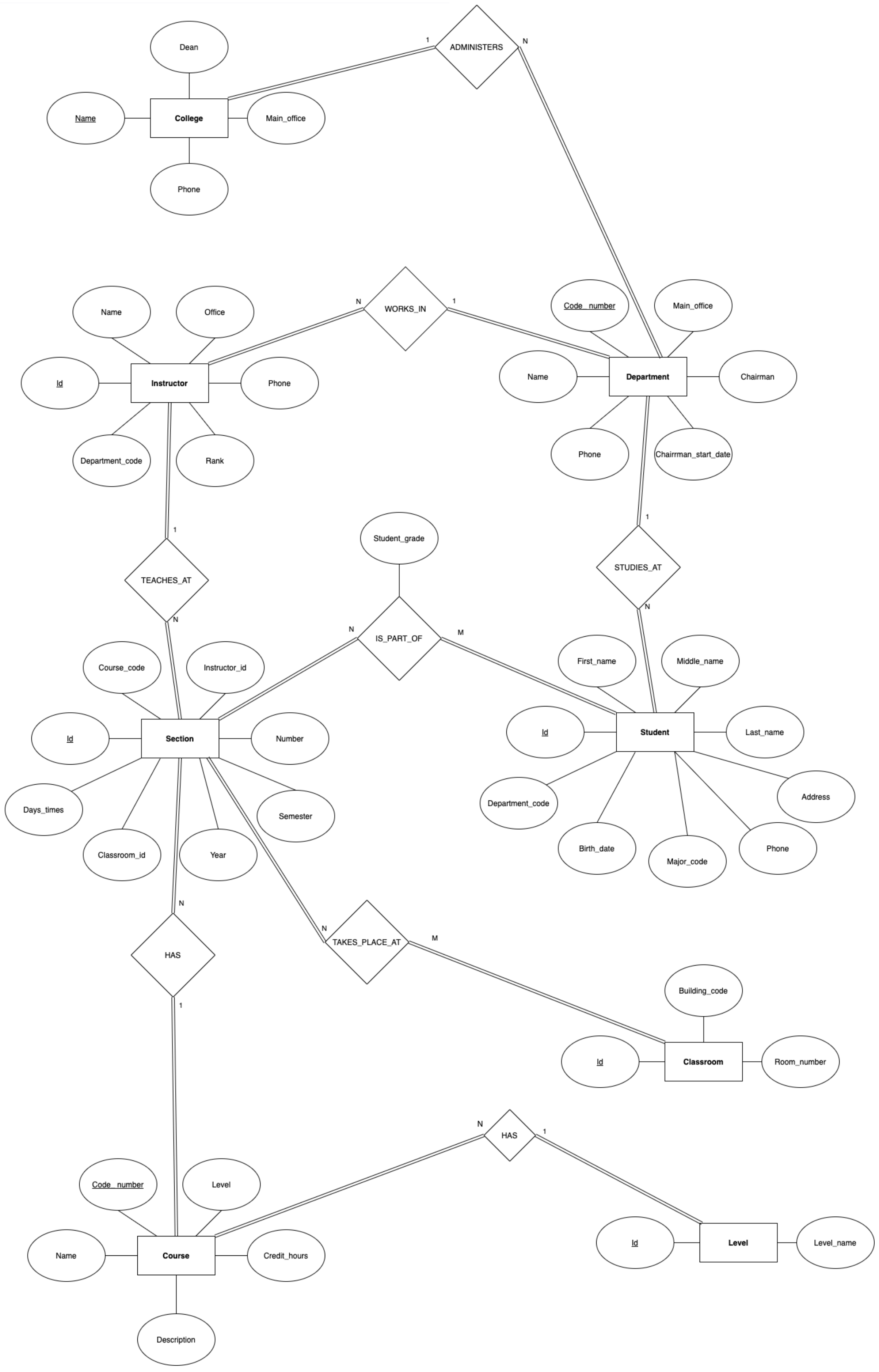


Table Name	Primary Key	Foreign Key(s)	Notes
College	name	-	-
Department	code_number	-	-
Course	code_number	level → Level(id)	Level determines course type
Instructor	id	department_code → Department(code_number)	-
Student	id	department_code → Department(code_number)	-
Classroom	id	-	-
Section	id	course_code → Course, instructor_id → Instructor, classroom_id → Classroom	-
Level	id	-	Level of study (e.g., Freshman)
College_Department	-	college_name → College, department_code → Department	M:N mapping
Student_Section	-	student_id → Student, section_id → Section	M:N relationship with grade



ADMINISTERS
1
N
Dean
Name
College
Main_office
Phone
WORKS_IN
N
1
Name
Office
Id
Instructor
Phone
Department_code
Rank
Code_number
Main_office
Name
Department
Chairman
Phone
Chairman_start_date
1
TEACHES_AT
N
Student_grade
IS_PART_OF
N
M
1
STUDIES_AT
N
Course_code
Instructor_id
Id
Section
Number
Days_times
Classroom_id
Year
Semester
First_name
Middle_name
Id
Student
Last_name
Department_code
Address
Birth_date
Major_code
Phone
N
HAS
1
N
TAKES_PLACE_AT
M
Building_code
Id
Classroom
Room_number
N
HAS
1
Code_number
Level
Name
Course
Credit_hours
Description
Id
Level
Level_name

6. Table Relationships

Table	Foreign Key Reference	Relation Type
Instructor	department_code → Department.code_number	Many-to-One
Student	department_code → Department.code_number	Many-to-One
Course	level → Level.id	Many-to-One
Section	course_code → Course.code_number	Many-to-One
Section	instructor_id → Instructor.id	Many-to-One
Section	classroom_id → Classroom.id	Many-to-One
College_Department	college_name → College.name	Many-to-One
College_Department	department_code → Department.code_number	Many-to-One
Student_Section	section_id → Section.id	Many-to-One
Student_Section	student_id → Student.id	Many-to-One

7. Tabular Data Format

Example: `Course`

name	code_number	level	credit_hours	description
Databases	1234	4	30	description 1
Testing	1235	4	15	description 2
Microservices	1236	4	10	description 3

8. Table Creation

Example of table creation:

```
sql
CopyEdit
CREATE TABLE Course (
    name CHAR(50),
    code_number INT(11),
    level INT(1),
    credit_hours INT(5),
    description CHAR(100),
    PRIMARY KEY (code_number),
    FOREIGN KEY (level) REFERENCES Level(id)
);
```

9. CODE

```
CREATE database University_1;

use university;
use university;

CREATE TABLE College (

   name CHAR(50),

   main_office CHAR(50),

   phone CHAR(12),

   dean CHAR(50),

   PRIMARY KEY (name)

);

CREATE TABLE Department (

   name CHAR(50),

         code_number INT(11),

   main_office CHAR(50),

   phone CHAR(50),

   chairman CHAR(50),

   chairman_start_date DATE,

   PRIMARY KEY (code_number)

);

CREATE TABLE Course (

   name CHAR(50),

         code_number INT(11),

   level INT(1),

   credit_hours INT(5),

   description CHAR(100),

   PRIMARY KEY (code_number)

);
```

```
CREATE TABLE Instructor (
   id INT,
   name CHAR(50),
   office CHAR(50),
   phone CHAR(50),
   inst_rank CHAR(50),
   department_code INT,
   PRIMARY KEY (id),
   FOREIGN KEY (department_code) REFERENCES Department(code_number)
);
CREATE TABLE Student (
   id INT,
          name CHAR(50),
   first_name CHAR(50),
   middle_name CHAR(50),
   last_name CHAR(50),
   address CHAR(50),
   phone CHAR(50),
   major_code CHAR(50),
   birth_date DATE,
   department_code INT,
   PRIMARY KEY (id),
   FOREIGN KEY (department_code) REFERENCES Department(code_number)
);
CREATE TABLE Classroom (
   id INT,
          building_code INT,
   room_number INT,
   PRIMARY KEY (id)
```

```
);
CREATE TABLE Section (
   id INT,
   course_code INT,
   instructor_id INT,
   number INT,
   semester CHAR(50),
   year INT,
   classroom_id INT,
   days_times CHAR(50),
   PRIMARY KEY (id),
   FOREIGN KEY (course_code) REFERENCES Course(code_number),
   FOREIGN KEY (instructor_id) REFERENCES Instructor(id),
   FOREIGN KEY (classroom_id) REFERENCES Classroom(id)
);
CREATE TABLE College_Department (
   college_name CHAR(50),
          department_code INT,
   FOREIGN KEY (college_name) REFERENCES College(name),
   FOREIGN KEY (department_code) REFERENCES Department(code_number)
);
CREATE TABLE Level (
   id INT,
          level_name CHAR(10),
   PRIMARY KEY (id)
);
CREATE TABLE Student_Section (
   section_id INT,
   student_id INT,
          student_grade CHAR(4),
   FOREIGN KEY (section_id) REFERENCES Section(id),
   FOREIGN KEY (student_id) REFERENCES Student(id)
);
ALTER TABLE College
ADD UNIQUE (name);
ALTER TABLE Department
ADD UNIQUE (name);
ALTER TABLE Course
ADD FOREIGN KEY (level) REFERENCES Level(id);
ALTER TABLE Course
ADD UNIQUE (name);
ALTER TABLE Course
          DROP FOREIGN KEY course_ibfk_1;

ALTER TABLE Course
ADD CONSTRAINT course_ibfk_1 FOREIGN KEY (level)
          REFERENCES level (id)
          ON UPDATE CASCADE
          ON DELETE SET NULL;
ALTER TABLE Instructor
          DROP FOREIGN KEY instructor_ibfk_1;

ALTER TABLE Instructor
ADD CONSTRAINT instructor_ibfk_1 FOREIGN KEY (department_code)
          REFERENCES Department (code_number)
          ON UPDATE CASCADE
          ON DELETE SET NULL;
ALTER TABLE Student
```

```
        DROP FOREIGN KEY student_ibfk_1;


ALTER TABLE Student
ADD CONSTRAINT student_ibfk_1 FOREIGN KEY (department_code)
        REFERENCES Department (code_number)
        ON UPDATE CASCADE
        ON DELETE SET NULL;
ALTER TABLE Section
        DROP FOREIGN KEY section_ibfk_1;


ALTER TABLE Section
ADD CONSTRAINT section_ibfk_1 FOREIGN KEY (course_code)
        REFERENCES Course (code_number)
        ON UPDATE CASCADE
        ON DELETE SET NULL;
ALTER TABLE Section
        DROP FOREIGN KEY section_ibfk_2;


ALTER TABLE Section
ADD CONSTRAINT section_ibfk_2 FOREIGN KEY (instructor_id)
        REFERENCES Instructor (id)
        ON UPDATE CASCADE
        ON DELETE SET NULL;
ALTER TABLE Section
        DROP FOREIGN KEY section_ibfk_3;


ALTER TABLE Section
ADD CONSTRAINT section_ibfk_3 FOREIGN KEY (classroom_id)
        REFERENCES Classroom (id)
        ON UPDATE CASCADE
        ON DELETE SET NULL;
ALTER TABLE College_Department
        DROP FOREIGN KEY college_department_ibfk_1;

ALTER TABLE College_Department
ADD CONSTRAINT college_department_ibfk_1 FOREIGN KEY (college_name)
        REFERENCES College (name)
        ON UPDATE CASCADE
        ON DELETE SET NULL;
ALTER TABLE College_Department
        DROP FOREIGN KEY college_department_ibfk_2;

ALTER TABLE College_Department
ADD CONSTRAINT college_department_ibfk_2 FOREIGN KEY (department_code)
        REFERENCES Department (code_number)
        ON UPDATE CASCADE
        ON DELETE SET NULL;
ALTER TABLE Student_Section
        DROP FOREIGN KEY student_section_ibfk_1;

ALTER TABLE Student_Section
ADD CONSTRAINT student_section_ibfk_1 FOREIGN KEY (section_id)
        REFERENCES Section (id)
        ON UPDATE CASCADE
        ON DELETE SET NULL;
ALTER TABLE Student_Section
        DROP FOREIGN KEY student_section_ibfk_2;
```

```
ALTER TABLE Student_Section
ADD CONSTRAINT student_section_ibfk_2 FOREIGN KEY (student_id)
        REFERENCES Student (id)
        ON UPDATE CASCADE
        ON DELETE SET NULL;
CREATE USER 'paul1'@'%' IDENTIFIED BY 'password';
CREATE USER 'constantin1'@'%' IDENTIFIED BY 'password';
CREATE USER 'marius1'@'%' IDENTIFIED BY 'password';
GRANT ALL ON *.* TO 'paul1'@'%' WITH GRANT OPTION;
GRANT SELECT ON *.* TO 'constantin1'@'%' WITH GRANT OPTION;
GRANT UPDATE, DELETE ON *.* TO 'marius1'@'%' WITH GRANT OPTION;
SHOW GRANTS for 'paul1'@'%';
SHOW GRANTS for 'constantin1'@'%';
SHOW GRANTS for 'marius1'@'%';
SELECT * FROM mysql.user;
CREATE VIEW User_role_information AS
        SELECT User, Select_priv, Insert_priv, Update_priv, Delete_priv, Create_priv
                FROM mysql.user
                WHERE Select_priv = 'Y' OR Insert_priv = 'Y' OR Update_priv = 'Y' OR Delete_priv =
'Y' OR Create_priv = 'Y';
INSERT INTO Classroom (id, building_code, room_number) VALUES ('1', '1001', '1');
INSERT INTO Classroom (id, building_code, room_number) VALUES ('2', '1002', '16');
INSERT INTO Classroom (id, building_code, room_number) VALUES ('3', '1003', '17');
INSERT INTO Classroom (id, building_code, room_number) VALUES ('4', '1004', '24');
INSERT INTO Classroom (id, building_code, room_number) VALUES ('5', '1005', '36');
INSERT INTO College (name, main_office, phone, dean) VALUES ('Kea1', 'L37', '123456789', 'Jesper
N');
INSERT INTO College (name, main_office, phone, dean) VALUES ('Kea2', 'L38', '123456788', 'Maria
I');
INSERT INTO College (name, main_office, phone, dean) VALUES ('Kea3', 'L16', '123456787',
'Christian L');
```

```
INSERT INTO College (name, main_office, phone, dean) VALUES ('Kea4', 'L98', '123456786', 'Christoffer K');

INSERT INTO College (name, main_office, phone, dean) VALUES ('Kea5', 'L101', '123456781', 'Peter D');

INSERT INTO Level (id, level_name) VALUES ('1', 'Freshman');

INSERT INTO Level (id, level_name) VALUES ('2', 'Sophomore');

INSERT INTO Level (id, level_name) VALUES ('3', 'Junior');

INSERT INTO Level (id, level_name) VALUES ('4', 'Senior');

INSERT INTO Level (id, level_name) VALUES ('5', 'MS');

INSERT INTO Level (id, level_name) VALUES ('6', 'PhD');

INSERT INTO Course (name, code_number, level, credit_hours, description) VALUES ('Databases', '1234', '4', '30', 'description 1');

INSERT INTO Course (name, code_number, level, credit_hours, description) VALUES ('Testing', '1235', '4', '15', 'description 2');

INSERT INTO Course (name, code_number, level, credit_hours, description) VALUES ('Microservices', '1236', '4', '10', 'description 3');

INSERT INTO Course (name, code_number, level, credit_hours, description) VALUES ('Android Game', '1237', '3', '15', 'description 4');

INSERT INTO Course (name, code_number, level, credit_hours, description) VALUES ('Python', '1238', '3', '10', 'description 5');

INSERT INTO Department (name, code_number, main_office, phone, chairman, chairman_start_date) VALUES ('Computer Science', '5678', 'Lygten', '123456789', 'Jakob P', '2015-06-11');

INSERT INTO Department (name, code_number, main_office, phone, chairman, chairman_start_date) VALUES ('Software Development', '1678', 'GBG', '123456783', 'Martin M', '2007-10-20');

INSERT INTO Department (name, code_number, main_office, phone, chairman, chairman_start_date) VALUES ('Web Development', '2678', 'GBG', '123456786', 'Daniel E', '2009-12-30');

INSERT INTO Department (name, code_number, main_office, phone, chairman, chairman_start_date) VALUES ('IT Security', '3678', 'GBG', '123456781', 'Mark T', '2008-02-08');

INSERT INTO Department (name, code_number, main_office, phone, chairman, chairman_start_date) VALUES ('Datamatiker', '4678', 'Lygten', '123456788', 'Thomas S', '2019-07-10');

INSERT INTO College_Department (college_name, department_code) VALUES ('Kea1', '1678');
```

```sql
INSERT INTO College_Department (college_name, department_code) VALUES ('Kea2', '2678');

INSERT INTO College_Department (college_name, department_code) VALUES ('Kea3', '3678');

INSERT INTO College_Department (college_name, department_code) VALUES ('Kea1', '4678');

INSERT INTO College_Department (college_name, department_code) VALUES ('Kea1', '5678');

INSERT INTO Instructor (id, name, office, phone, inst_rank, department_code) VALUES ('007',
'Andrea Corradini', 'Norrebro', '123456789', '1', '1678');

INSERT INTO Instructor (id, name, office, phone, inst_rank, department_code) VALUES ('012',
'Arturo M', 'Norrebro', '123456787', '2', '2678');

INSERT INTO Instructor (id, name, office, phone, inst_rank, department_code) VALUES ('067',
'Christian K', 'Norrebro', '123456784', '3', '4678');

INSERT INTO Instructor (id, name, office, phone, inst_rank, department_code) VALUES ('031',
'Kristoffer Mikklas', 'Norrebro', '123456782', '2', '2678');

INSERT INTO Instructor (id, name, office, phone, inst_rank, department_code) VALUES ('090', 'Jon',
'Norrebro', '123456781', '5', '1678');

INSERT INTO Section (id, course_code, instructor_id, number, semester, year, classroom_id,
days_times) VALUES ('1', '1234', '7', '10', '1', '2020', '5', 'TuWeTh 9:00 AM - 12:00 AM');

INSERT INTO Section (id, course_code, instructor_id, number, semester, year, classroom_id,
days_times) VALUES ('2', '1235', '31', '11', '2', '2018', '1', 'MoTuFr 9:00 AM - 14:00 AM');

INSERT INTO Section (id, course_code, instructor_id, number, semester, year, classroom_id,
days_times) VALUES ('3', '1236', '12', '12', '3', '2016', '3', 'TuWeTh 9:00 AM - 10:00 AM');

INSERT INTO Section (id, course_code, instructor_id, number, semester, year, classroom_id,
days_times) VALUES ('4', '1237', '67', '13', '4', '2018', '4', 'MoTuFr 9:00 AM - 14:00 AM');

INSERT INTO Section (id, course_code, instructor_id, number, semester, year, classroom_id,
days_times) VALUES ('5', '1238', '90', '14', '5', '2019', '1', 'TuWeTh 9:00 AM - 12:00 AM');

INSERT INTO Student (id, name, first_name, middle_name, last_name, address, phone,
major_code, birth_date, department_code) VALUES ('1', 'Paul Panaitescu', 'Paul', NULL,
'Panaitescu', 'Albertslund', '087654321', '1111', '1900-10-20', '1678');

INSERT INTO Student (id, name, first_name, middle_name, last_name, address, phone,
major_code, birth_date, department_code) VALUES ('2',  'Constantin Razvan Tarau', 'Constantin',
'Razvan', 'Tarau', 'Albertslund', '287654321', '1112', '1800-10-20', '1678');

INSERT INTO Student (id, name, first_name, middle_name, last_name, address, phone,
major_code, birth_date, department_code) VALUES ('3',  'Marius Daniel Munteanu', 'Marius',
'Daniel', 'Munteanu', 'Albertslund', '387654321', '1113', '2000-10-20', '2678');
```

```
INSERT INTO Student (id, name, first_name, middle_name, last_name, address, phone,
major_code, birth_date, department_code) VALUES ('4', 'Jakob M',  'Jakob', NULL, 'M', 'Norrrebro',
'587654321', '1198', '2015-10-20', '2678');

INSERT INTO Student (id, name, first_name, middle_name, last_name, address, phone,
major_code, birth_date, department_code) VALUES ('5',  'Dragos Andrei Mocanasu', 'Dragos',
'Andrei', 'Mocanasu', 'Valby', '787654321', '2113', '2100-01-20', '4678');

INSERT INTO Student_Section (section_id, student_id, student_grade) VALUES ('1', '1', '12');

INSERT INTO Student_Section (section_id, student_id, student_grade) VALUES ('1', '2', '12');

INSERT INTO Student_Section (section_id, student_id, student_grade) VALUES ('1', '3', '12');

INSERT INTO Student_Section (section_id, student_id, student_grade) VALUES ('2', '1', '4');

INSERT INTO Student_Section (section_id, student_id, student_grade) VALUES ('4', '1', '10');
-- Get all students from department 'Software Development'

SELECT * FROM Student

WHERE department_code = 1678;



SELECT * FROM Course

WHERE credit_hours > 15;



SELECT name, department_code FROM Instructor;
-- Insert a new student

INSERT INTO Student (id, name, first_name, middle_name, last_name, address, phone,
major_code, birth_date, department_code)

VALUES (6, 'Alice Johnson', 'Alice', NULL, 'Johnson', 'Roskilde', '987654321', '2222', '2001-03-14',
1678);



INSERT INTO Student_Section (section_id, student_id, student_grade)

VALUES (2, 6, '10');


UPDATE Student
```

```
SET phone = '999999999'
WHERE id = 3;



UPDATE Course
SET credit_hours = 20
WHERE name = 'Python';
DELETE FROM Student
WHERE id = 4;
DELETE FROM Section
WHERE id = 5;

SELECT s.name AS student_name, d.name AS department_name
FROM Student s
JOIN Department d ON s.department_code = d.code_number;



SELECT i.name AS instructor_name, c.name AS course_name
FROM Instructor i
JOIN Section s ON i.id = s.instructor_id
JOIN Course c ON s.course_code = c.code_number;



SELECT st.name AS student_name, co.name AS course_name, sec.semester, sec.year
FROM Student_Section ss
JOIN Student st ON ss.student_id = st.id
JOIN Section sec ON ss.section_id = sec.id
JOIN Course co ON sec.course_code = co.code_number;
SELECT department_code, COUNT(*) AS student_count
```

```
FROM Student

GROUP BY department_code;


SELECT section_id, AVG(CAST(student_grade AS UNSIGNED)) AS average_grade

FROM Student_Section

GROUP BY section_id;


SELECT name FROM Student

WHERE id IN (

    SELECT student_id

    FROM Student_Section ss

    JOIN Section s ON ss.section_id = s.id

    JOIN Course c ON s.course_code = c.code_number

    WHERE c.name = 'Databases'

);

CREATE VIEW StudentCourseView AS

SELECT s.name AS student_name, c.name AS course_name, ss.student_grade

FROM Student_Section ss

JOIN Student s ON ss.student_id = s.id

JOIN Section sec ON ss.section_id = sec.id

JOIN Course c ON sec.course_code = c.code_number;
```

10. SQL Queries with Output

✅*Get All Students from Software Development*

```
sql
CopyEdit
SELECT * FROM Student WHERE department_code = 1678;
```

Output: Returns students with department_code 1678.

✅*Get Courses with More than 15 Credit Hours*

```
sql
CopyEdit
SELECT * FROM Course WHERE credit_hours > 15;
```

Output:

- Databases (30 credit hours)

✅ *Insert a New Student*

```
sql
CopyEdit
INSERT INTO Student (...)
VALUES (6, 'Alice Johnson', 'Alice', NULL, 'Johnson', 'Roskilde',
'987654321', '2222', '2001-03-14', 1678);
```

✅ *Update Student Phone Number*

```
sql
CopyEdit
UPDATE Student SET phone = '999999999' WHERE id = 3;
```

✅ *Delete a Section*

```
sql
CopyEdit
DELETE FROM Section WHERE id = 5;
```

✅ *Get Students and Their Departments*

```
sql
CopyEdit
SELECT s.name AS student_name, d.name AS department_name
FROM Student s JOIN Department d ON s.department_code = d.code_number;
```

✅ *Instructor and Course They Teach*

```
sql
CopyEdit
SELECT i.name AS instructor_name, c.name AS course_name
FROM Instructor i
JOIN Section s ON i.id = s.instructor_id
JOIN Course c ON s.course_code = c.code_number;
```

✅ *Students and the Courses They Took*

```
sql
CopyEdit
SELECT st.name AS student_name, co.name AS course_name, sec.semester,
sec.year
FROM Student_Section ss
JOIN Student st ON ss.student_id = st.id
JOIN Section sec ON ss.section_id = sec.id
JOIN Course co ON sec.course_code = co.code_number;
```

✅ *Students Enrolled in 'Databases'*

```
sql
CopyEdit
SELECT name FROM Student
WHERE id IN (
    SELECT student_id
    FROM Student_Section ss
    JOIN Section s ON ss.section_id = s.id
    JOIN Course c ON s.course_code = c.code_number
    WHERE c.name = 'Databases'
);
```

---

11. Summary

This case study demonstrates:

- A scalable university database design
- Proper use of primary and foreign keys
- Normalization and referential integrity
- Real-world use cases: enrolling students, assigning instructors, querying enrollments
- Role-based access with SQL users and privileges

---

12. Conclusion

This University Management System showcases the essential structure and capabilities required in a relational database environment for handling academic institutions. It emphasizes proper database normalization, consistent data handling, user access control, and efficient querying for administrative tasks.