

Database of college management system

Scope ,Erd diagram, schema, normalization and implementation

Submitted by:

Aimen Nadeem (1)

Zufra Jahan (37)

Ayesha Farrukh (38)

Submitted to :

Dr.sobia Khalid

**Database Systems Project**

**Scope of College Management System:**

The scope of a College Management System encompasses a wide range of tasks and functionalities aimed at facilitating different aspects of academic and administrative operations. Some of the key tasks that this database system will perform include:

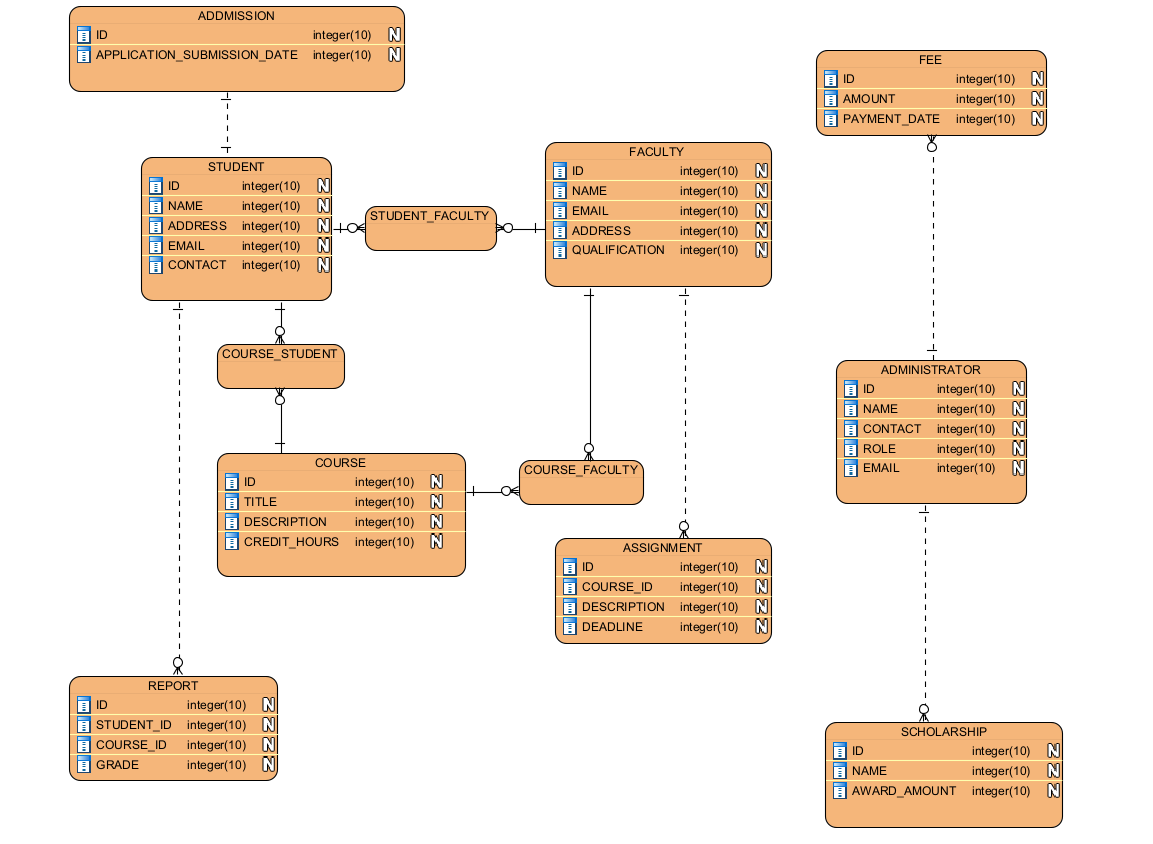
Student Management

Faculty Management

Course Management

Administrator Tasks

**ERD Diagram Of College Management System**



**SCOPE**

**Introduction to College Management System:**

A College Management System is a comprehensive software solution designed to streamline the administrative and academic processes within educational institutions. With the ever-increasing complexity of managing academic and administrative tasks, it provides a centralized platform for efficiently handling various operations, enhancing communication among people, and improving overall productivity.

**Scope of College Management System:**

The scope of a College Management System encompasses a wide range of tasks and functionalities aimed at facilitating different aspects of academic and administrative operations. Some of the key tasks that this database system will perform include:

**Student Management:**

Admission processes including application submission, document verification, and enrollment.

Student records management including personal details, academic performance, attendance, and disciplinary records.

Course registration, class scheduling, and academic advising.

Generation of student transcripts, grade reports, and academic certificates.

**Faculty Management:**

Faculty information management including personal details, qualifications, and teaching assignments.

Course allocation, class scheduling, and evaluation of student performance.

Faculty performance evaluation and professional development tracking.

Communication tools for facilitating interaction between faculty members and students.

**Course Management:**

Course catalog management including course descriptions, prerequisites, and credit hours.

Curriculum planning, development, and revision.

Assignment and assessment management including grading, feedback, and evaluation.

Integration with learning management systems for online course delivery and content management.

**Administrative Tasks:**

Fee collection, financial aid management, and scholarship administration.

Facility management including library services, hostel accommodation, and transportation.

Human resources management including payroll processing, employee attendance, and leave management.

Inventory management for educational resources, equipment, and supplies.

**Conclusion:**

In conclusion, a College Management System plays a ivotal role in enhancing the efficiency, transparency, and effectiveness of academic and administrative processes within educational institutions. By automating routine tasks, facilitating communication, and providing valuable insights through data analysis, it contributes to the overall success and growth of the institution.

**SCHEMA Of College Management System**

1. **ADMISSION**

|  |  |  |
| --- | --- | --- |
| **ID** | **APPLICATION\_SUBMISSION\_DATE** | **STUDENT\_ID** |

1. **STUDENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **NAME** | **ADDRESS** | **EMAIL** | **CONTACT** |

1. **FACULTY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **NAME** | **EMAIL** | **ADDRESS** | **QUALIFICATION** | **CONTACT** |

1. **COURSE**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **TITLE** | **DESCRIPTION** | **CREADIT\_HOURS** |

1. **COURSE\_STUDENT**

|  |  |  |
| --- | --- | --- |
| **COURSE\_ID** | **STUDENT\_ID** | **CREADIT\_HOURS** |

Here foreign keys **COURSE\_ID** and **STUDENT\_ID** both make primary composite key.

1. **REPORT**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **STUDENT\_ID** | **COURSE\_ID** | **GRADES** |

1. **ASSIGNMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **COURSE\_ID** | **DESCRIPTION** | **DEADLINE** |

1. **STUDENT\_ASSIGNMENT**

|  |  |  |
| --- | --- | --- |
| **STUDENT\_ID** | **ASSIGNMENT\_ID** | **GRADES** |

Here foreign keys **ASSIGNMENT\_ID** and **STUDENT\_ID** both make primary composite key.

1. **FEE**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **AMOUNT** | **PAYMENT\_DATE** | **ADMISSION\_ID** |

1. **ADMINITRATOR**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **NAME** | **CONTACT** | **ROLE** | **EMAIL** |

1. **SCHOLARSHIP**

|  |  |  |
| --- | --- | --- |
| **ID** | **NAME** | **AWARD\_AMOUNT** |

**NORMALIZATION Of College Management System**

|  |  |  |
| --- | --- | --- |
| **Table** | **Keys** | **Functional Dependencies** |
| Student | ID | *Email* -> Name, Address, Contact (This might not always hold true, so it's a weak dependency) |
| Faculty | ID | *Email* -> Name, Qualification, Contact (Similar to Student) |
| Course | ID | Title, Description -> None |
| Administrator | ID | *Email* -> Name, Contact, Role (Similar to Student) |
| Application | ID | Student\_ID -> Application\_Submission\_Date |
| Fee | ID | Application\_ID -> Amount, Payment\_Date |
| Course\_Student | (Course\_ID, Student\_ID) | Course\_ID -> Student\_ID, Student\_ID -> Course\_ID (Composite Key) |
| Assignment | ID | Course\_ID -> Description, Deadline |
| Student\_Assignment | (Student\_ID, Assignment\_ID) | Student\_ID -> Assignment\_ID, Assignment\_ID -> Student\_ID (Composite Key) |
| Report | ID | Student\_ID, Course\_ID -> Description (Composite Key) |
| Scholarship | ID | Name -> Award\_Amount |

**First Normal Form (1NF):**

The schema is already in 1NF because all tables have a primary key that uniquely identifies each record, and there are no repeating groups of attributes within a table.

**Second Normal Form (2NF):**

The schema currently violates 2NF in two tables:

1. Application: The **Application\_Submission\_Date** attribute depends only on the Student\_ID, not the entire Application record.
2. Fee: The **Payment\_Date** and **Amount** attributes depend only on the Application\_ID, not the entire Fee record.

**Normalized Schema:**

Here's the normalized schema with separate tables:

1. **ADMISSION**

|  |  |  |
| --- | --- | --- |
| **ID** | **APPLICATION\_SUBMISSION\_DATE** | **STUDENT\_ID** |

1. **STUDENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **NAME** | **ADDRESS** | **EMAIL** | **CONTACT** |

1. **FACULTY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **NAME** | **EMAIL** | **ADDRESS** | **QUALIFICATION** | **CONTACT** |

1. **COURSE**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **TITLE** | **DESCRIPTION** | **CREADIT\_HOURS** |

1. **COURSE\_STUDENT**

|  |  |  |
| --- | --- | --- |
| **COURSE\_ID** | **STUDENT\_ID** | **CREADIT\_HOURS** |

1. **REPORT**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **STUDENT\_ID** | **COURSE\_ID** | **GRADES** |

1. **ASSIGNMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **COURSE\_ID** | **DESCRIPTION** | **DEADLINE** |

1. **STUDENT\_ASSIGNMENT**

|  |  |  |
| --- | --- | --- |
| **STUDENT\_ID** | **ASSIGNMENT\_ID** | **GRADES** |

1. **FEE**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **AMOUNT** | **PAYMENT\_DATE** | **ADMISSION\_ID** |

1. **ADMINITRATOR**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **NAME** | **CONTACT** | **ROLE** | **EMAIL** |

1. **SCHOLARSHIP**

|  |  |  |
| --- | --- | --- |
| **ID** | **NAME** | **AWARD\_AMOUNT** |

1. **New Table: ADDMISSION\_DETAILS**

|  |  |  |
| --- | --- | --- |
| **ID** | **ADDMISSION\_ID** | **APPLICATION\_SUBMISSION\_DATE** |

**13. New Table: PAYMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **ADDMISSION\_ID** | **AMOUNT** | **PAYEMNT\_DATE** |

* We created a new table **Addmission\_Details** to store the Application\_Submission\_Date attribute. This attribute only depends on the Student\_ID, which is already captured in the Application table. Separating it avoids redundancy.
* Similarly, a new table **Payment** is created to store Amount and Payment\_Date. These depend solely on the Application\_ID and were previously redundant in the Fee table.

**Third Normal Form (3NF):**

A table is in 3NF if it satisfies both 1NF and 2NF conditions, and there are no transitive dependencies between non-key attributes. Therefore, the schema satisfies to 3NF requirements.

**BCNF (Boyce-Codd Normal Form):**

A relation is in BCNF if every determinant is a candidate key.

In our schema:

* All foreign key relationships are based on candidate keys (primary keys) of referenced tables.
* There are no determinants that are not also candidate keys.

For example, in Course\_Student, the composite key (Course\_ID, Student\_ID) is the only determinant for all attributes in the table. There is no scenario where another set of attributes determines a single attribute without being a candidate key itself.

Therefore, the schema also satisfies BCNF requirements.

**Fourth Normal Form (4NF):**

A relation is in fourth normal form if it is in BCNF and it contains no multivalued dependencies.

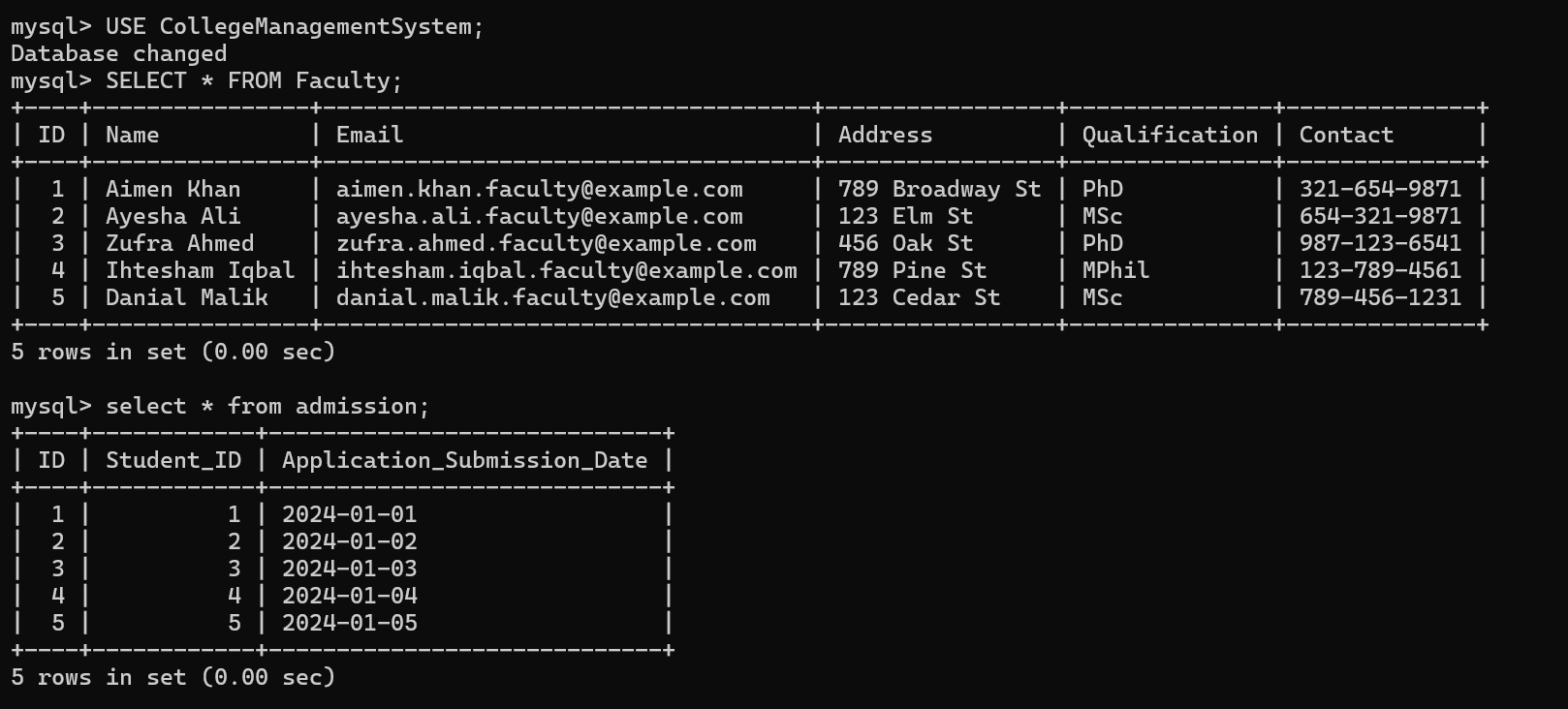
In our schema:

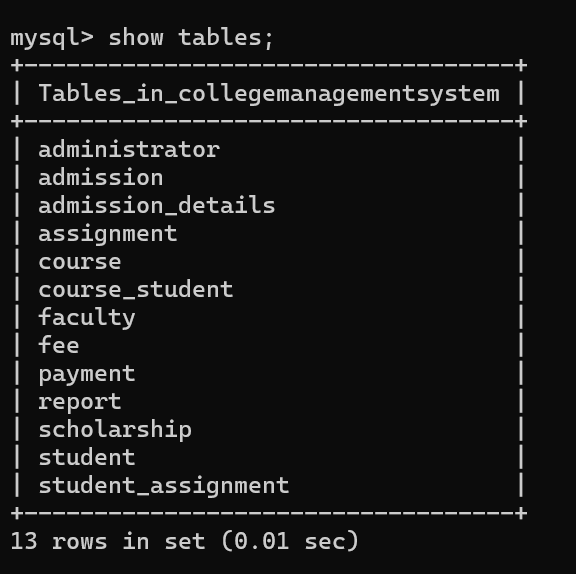
* There are no MVDs present in the relations.
* All foreign key relationships are based on candidate keys (primary keys) of the referenced tables.

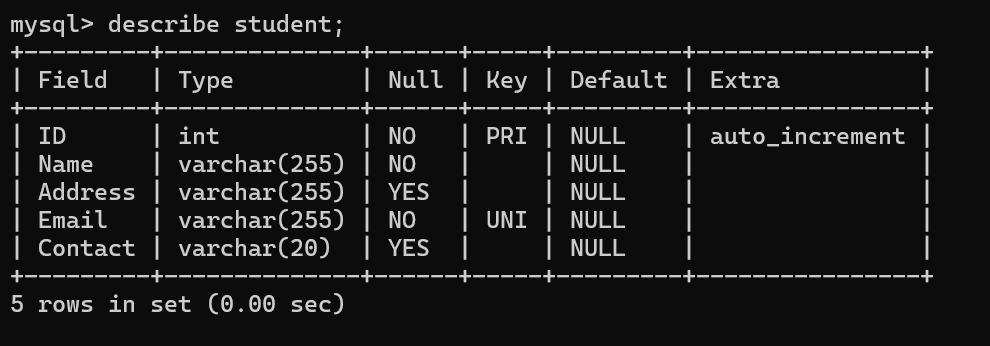
For example, in Course\_Student, the composite key (Course\_ID, Student\_ID) determines both Course\_ID for a student and Student\_ID for a course. However, this isn't an MVD because the composite key itself is the set that uniquely identifies a record in Course\_Student.

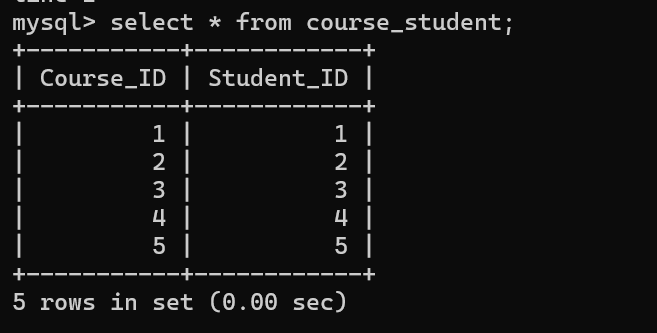
Therefore, the schema also satisfies 4NF requirements.

**Implentation in mysql**









**SCRIPT:**

CREATE DATABASE CollegeManagementSystem;

USE CollegeManagementSystem;

CREATE TABLE IF NOT EXISTS Student (

ID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(255) NOT NULL,

Address VARCHAR(255),

Email VARCHAR(255) UNIQUE NOT NULL,

Contact VARCHAR(20)

);

CREATE TABLE IF NOT EXISTS Faculty (

ID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(255) NOT NULL,

Email VARCHAR(255) UNIQUE NOT NULL,

Address VARCHAR(255),

Qualification VARCHAR(255),

Contact VARCHAR(20)

);

CREATE TABLE IF NOT EXISTS Course (

ID INT PRIMARY KEY AUTO\_INCREMENT,

Title VARCHAR(255) NOT NULL,

Description TEXT,

Credit\_Hours INT NOT NULL

);

CREATE TABLE IF NOT EXISTS Administrator (

ID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(255) NOT NULL,

Contact VARCHAR(20),

Role VARCHAR(255),

Email VARCHAR(255) UNIQUE NOT NULL

);

CREATE TABLE IF NOT EXISTS Admission (

ID INT PRIMARY KEY AUTO\_INCREMENT,

Student\_ID INT,

Application\_Submission\_Date DATE,

FOREIGN KEY (Student\_ID) REFERENCES Student(ID)

);

CREATE TABLE IF NOT EXISTS Fee (

ID INT PRIMARY KEY AUTO\_INCREMENT,

Amount DECIMAL(10, 2) NOT NULL,

Payment\_Date DATE,

Admission\_ID INT,

FOREIGN KEY (Admission\_ID) REFERENCES Admission(ID)

);

CREATE TABLE IF NOT EXISTS Course\_Student (

Course\_ID INT,

Student\_ID INT,

PRIMARY KEY (Course\_ID, Student\_ID),

FOREIGN KEY (Course\_ID) REFERENCES Course(ID),

FOREIGN KEY (Student\_ID) REFERENCES Student(ID)

);

CREATE TABLE IF NOT EXISTS Assignment (

ID INT PRIMARY KEY AUTO\_INCREMENT,

Course\_ID INT,

Description TEXT,

Deadline DATE,

FOREIGN KEY (Course\_ID) REFERENCES Course(ID)

);

CREATE TABLE IF NOT EXISTS Student\_Assignment (

Student\_ID INT,

Assignment\_ID INT,

Grades VARCHAR(10),

PRIMARY KEY (Student\_ID, Assignment\_ID),

FOREIGN KEY (Student\_ID) REFERENCES Student(ID),

FOREIGN KEY (Assignment\_ID) REFERENCES Assignment(ID)

);

CREATE TABLE IF NOT EXISTS Report (

ID INT PRIMARY KEY AUTO\_INCREMENT,

Student\_ID INT,

Course\_ID INT,

Grades VARCHAR(10),

FOREIGN KEY (Student\_ID) REFERENCES Student(ID),

FOREIGN KEY (Course\_ID) REFERENCES Course(ID)

);

CREATE TABLE IF NOT EXISTS Scholarship (

ID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(255),

Award\_Amount DECIMAL(10, 2) NOT NULL

);

CREATE TABLE IF NOT EXISTS Admission\_Details (

ID INT PRIMARY KEY AUTO\_INCREMENT,

Admission\_ID INT,

Application\_Submission\_Date DATE,

FOREIGN KEY (Admission\_ID) REFERENCES Admission(ID)

);

CREATE TABLE IF NOT EXISTS Payment (

ID INT PRIMARY KEY AUTO\_INCREMENT,

Admission\_ID INT,

Amount DECIMAL(10, 2) NOT NULL,

Payment\_Date DATE,

FOREIGN KEY (Admission\_ID) REFERENCES Admission(ID)

);

INSERT INTO Student (ID, Name, Address, Email, Contact) VALUES

(1, 'Aimen Khan', '789 Broadway St', 'aimen.khan@example.com', '321-654-9870'),

(2, 'Ayesha Ali', '123 Elm St', 'ayesha.ali@example.com', '654-321-9870'),

(3, 'Zufra Ahmed', '456 Oak St', 'zufra.ahmed@example.com', '987-123-6540'),

(4, 'Ihtesham Iqbal', '789 Pine St', 'ihtesham.iqbal@example.com', '123-789-4560'),

(5, 'Danial Malik', '123 Cedar St', 'danial.malik@example.com', '789-456-1230');

INSERT INTO Faculty (ID, Name, Email, Address, Qualification, Contact) VALUES

(1, 'Aimen Khan', 'aimen.khan.faculty@example.com', '789 Broadway St', 'PhD', '321-654-9871'),

(2, 'Ayesha Ali', 'ayesha.ali.faculty@example.com', '123 Elm St', 'MSc', '654-321-9871'),

(3, 'Zufra Ahmed', 'zufra.ahmed.faculty@example.com', '456 Oak St', 'PhD', '987-123-6541'),

(4, 'Ihtesham Iqbal', 'ihtesham.iqbal.faculty@example.com', '789 Pine St', 'MPhil', '123-789-4561'),

(5, 'Danial Malik', 'danial.malik.faculty@example.com', '123 Cedar St', 'MSc', '789-456-1231');

INSERT INTO Course (ID, Title, Description, Credit\_Hours) VALUES

(1, 'Computer Science 101', 'Introduction to Computer Science', 3),

(2, 'Mathematics 101', 'Introduction to Mathematics', 3),

(3, 'Physics 101', 'Introduction to Physics', 3),

(4, 'Chemistry 101', 'Introduction to Chemistry', 3),

(5, 'Biology 101', 'Introduction to Biology', 3);

INSERT INTO Administrator (ID, Name, Contact, Role, Email) VALUES

(1, 'Aimen Khan', '321-654-9872', 'Principal', 'aimen.khan.admin@example.com'),

(2, 'Ayesha Ali', '654-321-9872', 'Vice Principal', 'ayesha.ali.admin@example.com'),

(3, 'Zufra Ahmed', '987-123-6542', 'Head of Department', 'zufra.ahmed.admin@example.com'),

(4, 'Ihtesham Iqbal', '123-789-4562', 'Registrar', 'ihtesham.iqbal.admin@example.com'),

(5, 'Danial Malik', '789-456-1232', 'Finance Manager', 'danial.malik.admin@example.com');

INSERT INTO Admission (ID, Student\_ID, Application\_Submission\_Date) VALUES

(1, 1, '2024-01-01'),

(2, 2, '2024-01-02'),

(3, 3, '2024-01-03'),

(4, 4, '2024-01-04'),

(5, 5, '2024-01-05');

INSERT INTO Fee (ID, Amount, Payment\_Date, Admission\_ID) VALUES

(1, 1000.00, '2024-02-01', 1),

(2, 1100.00, '2024-02-02', 2),

(3, 1200.00, '2024-02-03', 3),

(4, 1300.00, '2024-02-04', 4),

(5, 1400.00, '2024-02-05', 5);

INSERT INTO Course\_Student (Course\_ID, Student\_ID) VALUES

(1, 1),

(2, 2),

(3, 3),

(4, 4),

(5, 5);

INSERT INTO Assignment (ID, Course\_ID, Description, Deadline) VALUES

(1, 1, 'Assignment 1 for Computer Science 101', '2024-03-01'),

(2, 2, 'Assignment 1 for Mathematics 101', '2024-03-02'),

(3, 3, 'Assignment 1 for Physics 101', '2024-03-03'),

(4, 4, 'Assignment 1 for Chemistry 101', '2024-03-04'),

(5, 5, 'Assignment 1 for Biology 101', '2024-03-05');

INSERT INTO Student\_Assignment (Student\_ID, Assignment\_ID, Grades) VALUES

(1, 1, 'A'),

(2, 2, 'B'),

(3, 3, 'C'),

(4, 4, 'B+'),

(5, 5, 'A+');

INSERT INTO Report (ID, Student\_ID, Course\_ID, Grades) VALUES

(1, 1, 1, 'A'),

(2, 2, 2, 'B'),

(3, 3, 3, 'C'),

(4, 4, 4, 'B+'),

(5, 5, 5, 'A+');

INSERT INTO Scholarship (ID, Name, Award\_Amount) VALUES

(1, 'Academic Excellence', 1000.00),

(2, 'Sports Scholarship', 800.00),

(3, 'Arts Scholarship', 600.00),

(4, 'Research Grant', 1500.00),

(5, 'Merit Scholarship', 1200.00);

INSERT INTO Payment (ID, Admission\_ID, Amount, Payment\_Date) VALUES

(1, 1, 1000.00, '2024-02-01'),

(2, 2, 1100.00, '2024-02-02'),

(3, 3, 1200.00, '2024-02-03'),

(4, 4, 1300.00, '2024-02-04'),

(5, 5, 1400.00, '2024-02-05');

commit;

