**University Database**

**1. first create master table**

CREATE table Departments (Dept\_id int , AUTO\_INCREMENT PRIMARY key ,

Dept\_name varchar(50) not null

);

**2. Add values in Departments table**

🡺 INSERT into departments (dept\_id,dept\_name)

VALUES (1, 'Human Resources'), (2, 'Finance'), (3, 'Engineering'), (4, 'Marketing'), (5, 'Sales'), (6, 'IT Support'), (7, 'Legal'), (8, 'Operations'), (9, 'Research and Development'), (10, 'Customer Service');

**3. Create student table and department table make a foreign key**

🡺 Create table student (std\_id int primary key AUTO\_INCREMENT,

std\_first\_name varchar(50)not null ,

std\_last\_name varchar(50) not null,

email varchar(50) not null,

phone varchar (50) not null,

date\_of\_birth dateTime not null,

enrollment\_date dateTime not null,

dept\_id int ,

FOREIGN key (dept\_id) REFERENCES departments(dept\_id))

**4. Insert data in student table and give data in foreign key reference**

INSERT into student (std\_id,std\_first\_name ,std\_last\_name,email,phone ,date\_of\_birth,enrollment\_date,dept\_id)

VALUES

(1, 'Yogendra', 'Patidar', 'yogendra.patidar@example.com', '1234567890', '2000-01-15', '2023-09-01', 1),

(2, 'Sarthak', 'Sharma', 'sarthak.sharma@example.com', '1234567891', '1999-05-22', '2023-09-01', 2),

(3, 'Tanmay', 'Vishwakarma', 'tanmay.vishwakarma@example.com', '1234567892', '2001-03-11', '2023-09-01', 3),

(4, 'Mani', 'Agrawal', 'mani.agrawal@example.com', '1234567893', '2002-07-08', '2023-09-01', 1),

(5, 'Abhishek', 'Umare', 'abhishek.umare@example.com', '1234567894', '2000-10-25', '2023-09-01', 5),

(6, 'Aaditya', 'Kashyap', 'aaditya.kashyap@example.com', '1234567895', '1998-12-19', '2023-09-01', 6),

(7, 'Atishay', 'Jain', 'atishay.jain@example.com', '1234567896', '1997-11-02', '2023-09-01', 7),

(8, 'Gourav', 'Rajput', 'gourav.rajput@example.com', '1234567897', '2001-06-14', '2023-09-01', 5),

(9, 'Yash', 'Soni', 'yash.soni@example.com', '1234567898', '1999-09-09', '2023-09-01', 9),

(10, 'Shivam', 'Yadav', 'shivam.Yadav@example.com', '1234567899', '2003-04-05', '2023-09-01', 10);

**5. Create professors table**

CREATE TABLE Professors ( professor\_id INT PRIMARY KEY, first\_name VARCHAR(100) NOT NULL, last\_name VARCHAR(100) NOT NULL, email VARCHAR(100) NOT NULL UNIQUE, phone VARCHAR(20) );

6 ] insert table from professors table

INSERT INTO Professors (professor\_id, first\_name, last\_name, email, phone)

VALUES

(1, 'Udit', 'Malviya', 'udit.malviya@example.com', '1234567890'),

(2, 'Shubham', 'Sharma', 'shubham.sharma@example.com', '2345678901'),

(3, 'Hemant', 'Rathod', 'hemant.rathod@example.com', '3456789012'),

(4, 'Amit', 'Yadav', 'amit.yadav@example.com', '4567890123'),

(5, 'Niraj', 'Rajput', 'niraj.rajput@example.com', '5678901234');

**7. create course table**

CREATE TABLE Courses(course\_id INT PRIMARY KEY AUTO\_INCREMENT,

course\_name VARCHAR(100) NOT NULL,

dept\_id INT,

professor\_id INT,

credits INT NOT NULL,

FOREIGN KEY (dept\_id) REFERENCES departments(dept\_id),

FOREIGN KEY (professor\_id)REFERENCES professors(professor\_id)

);

**8. Insert data into course table**

🡺 INSERT INTO Courses (course\_name, dept\_id, professor\_id, credits)

VALUES

('Mern Stack', 1, 1, 4),

('Front End Developer', 2, 2, 3),

('DSA', 1, 3, 4),

('Python', 3, 4, 3),

('JavaScript', 2, 5, 3);

**9. Create enrollment table**

CREATE TABLE enrollments (

enrollment\_id INT PRIMARY KEY AUTO\_INCREMENT,

std\_id INT,

course\_id INT,

enrollment\_date DATE NOT NULL,

grade VARCHAR(5),

FOREIGN KEY (std\_id) REFERENCES Student(std\_id),

FOREIGN KEY (course\_id) REFERENCES Courses(course\_id)

);

**10. Insert data into enrollments table**

INSERT INTO enrollments (std\_id, course\_id, enrollment\_date, grade)

VALUES

(1, 1, '2023-09-01', 'A'),

(2, 2, '2023-09-01', 'B+'),

(3, 3, '2023-09-01', 'A-'),

(4, 4, '2023-09-01', 'B'),

(5, 5, '2023-09-01', 'C+');

SQLQueries for the Case Study

1. **Find the Total Number of Students in Each Department**

🡺 SELECT dept\_name ,count(std\_id) as total\_students from student join departments on student.dept\_id = departments.Dept\_id

GROUP by dept\_name;

1. **List All Courses Taught by a Specific Professor**

🡺 SELECT p.first\_name, p.last\_name,c.course\_name AS courses\_name

FROM professors p

JOIN courses c ON c.professor\_id = p.professor\_id

GROUP BY p.professor\_id

**3. Find the Average Grade of Students in Each Course**

**🡺** SELECT courses.course\_name, AVG( CASE WHEN enrollments.grade = "A+" THEN 10

WHEN enrollments.grade = "A" THEN 10

WHEN enrollments.grade = "B+" THEN 9

WHEN enrollments.grade = "B" THEN 8

WHEN enrollments.grade = "C" THEN 7

ELSE 0 END) as Avrage\_Grades from enrollments

JOIN courses on courses.course\_id = enrollments.course\_id

GROUP by courses.course\_id

**4. List All Students Who Have Not Enrolled in Any Courses**

🡺 SELECT s.std\_first\_name ,s.std\_last\_name FROM student s inner JOIN enrollments e on s.std\_id = e.std\_id WHERE e.enrollment\_id

**5. Find the Number of Courses Offered by Each Department.**

SELECT d.dept\_name,COUNT(c.course\_id)as total\_courses from departments d

JOIN courses c on d.Dept\_id = c.dept\_id GROUP by d.Dept\_id

**6. List All Students Who Have Taken a Specific Course (e.g., 'Database Systems').**

🡺 SELECT s.std\_first\_name ,s.std\_last\_name from student s JOIN enrollments e on s.std\_id = e.std\_id

JOIN courses c on e.course\_id = c.course\_id WHERE c.course\_name = "python"

**7. Find the Most Popular Course Based on Enrollment Numbers.**

🡺 SELECT c.course\_name,COUNT(e.std\_id) as enrollment\_count from courses c JOIN enrollments e on c.course\_id = e.course\_id GROUP by c.course\_name ORDER by enrollment\_count DESC

**9.** **List All Professors Who Teach in More Than One Department.**

🡺 SELECT p.first\_name ,p.last\_name COUNT(c.professor\_id)as count\_pro FROM professors p JOIN courses c on p.professor\_id = c.professor\_id

GROUP by p.professor\_id;

**10.** Get the Highest and Lowest Grade in a Specific Course (e.g., 'Operating Systems')

🡺 SELECT courses.course\_name,Max(enrollments.grade) as Highest\_grade FROM courses

JOIN enrollments on enrollments.course\_id = courses.course\_id

GROUP by courses.course\_id