CSA0593

DATABASE MANAGEMENT SYSTEM.

NAME : S.DEEPIKA

REG NO. : 192324269

ASSIGNMENT - 1

1.University Academic Performance Analytics:

a) Design a database to analyze and report on student academic performance, attendance, and course completion.

Requirements: Model the relationships between students, courses, faculty, and attendance records.

b) Write SQL queries to generate reports for academic performance, identifying students at risk and tracking GPA trends.

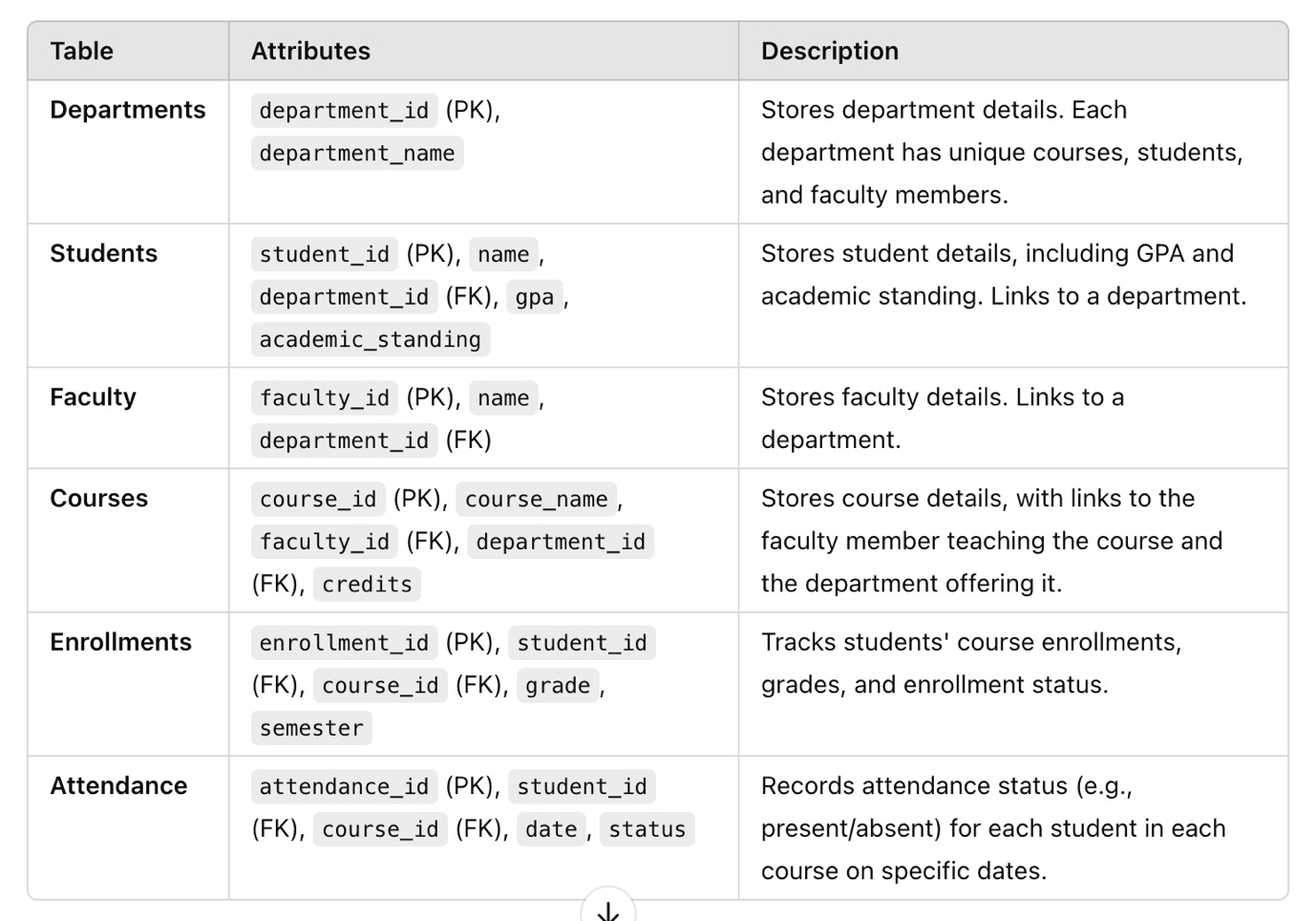
c) Create views for department heads to view summary reports on student performance and faculty teaching loads.

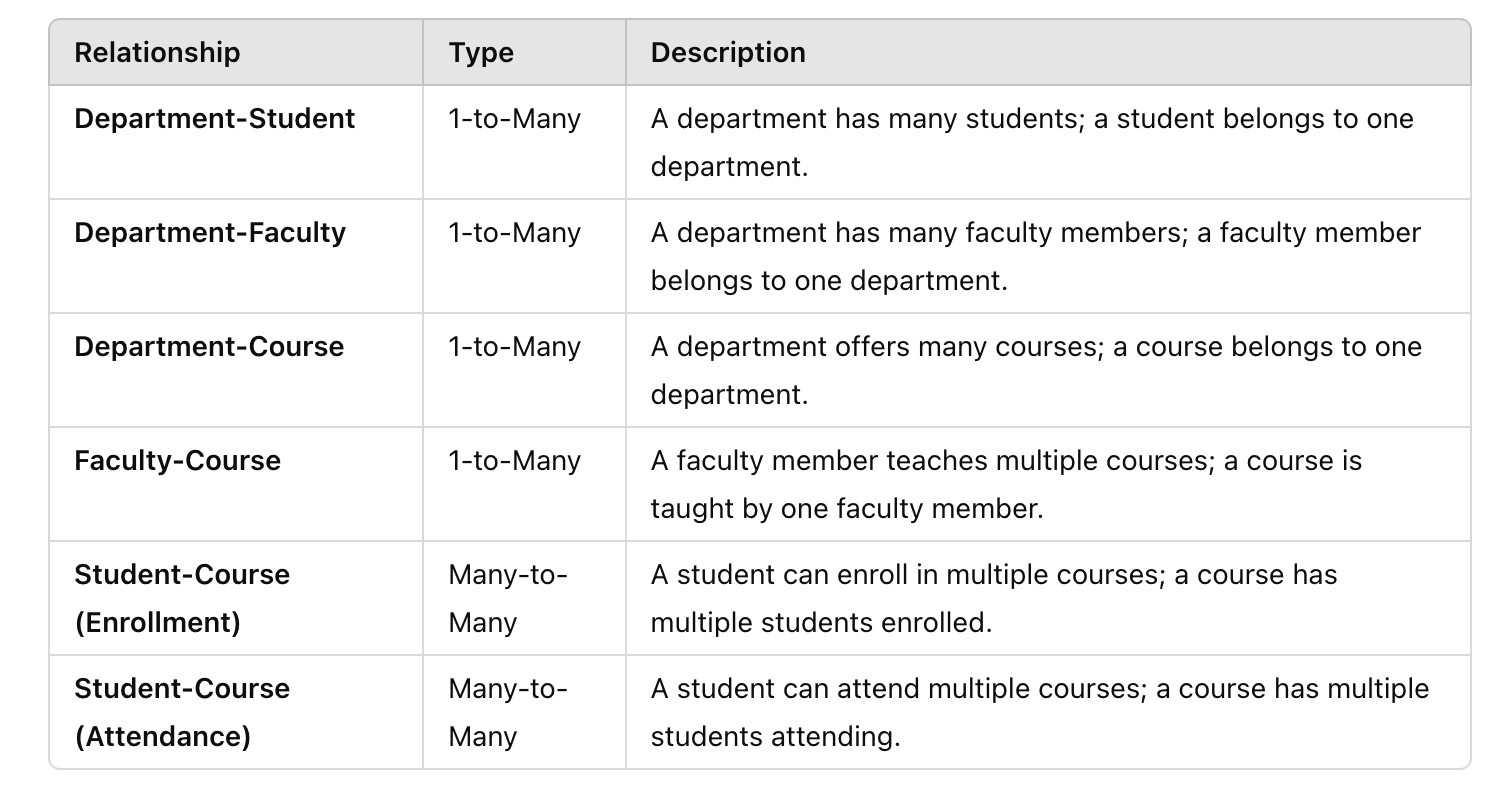
d) Use triggers to update GPA and academic standing when new grades are entered, ensuring data consistency

a).**DATABASE DESIGN:-**

To design a database for analysing and reporting on student academic performance, attendance, and course completion, here’s an outline that covers:

1. **Entity Definitions**: Defining students, courses, faculty, and attendance as entities.
2. **Relationships**: Modelling relationships between these entities.
3. **Database Tables**: Outlining the structure of the tables with attributes, primary keys (PK), and foreign keys (FK).





b). **SQL queries to generate reports for:**

1. **Academic Performance:**

SELECT s.student\_id, s.name, e.course\_id, c.course\_name, e.grade

FROM Students s

JOIN Enrollments e ON s.student\_id = e.student\_id

JOIN Courses c ON e.course\_id = c.course\_id

ORDER BY s.student\_id, e.course\_id;

1. **Identifying Students at Risk:**

SELECT student\_id, name, gpa

FROM Students

WHERE gpa < 2.0;

1. **Tracking GPA Trends:**

**SELECT s.student\_id, s.name, e.semester, AVG(e.grade) AS semester\_gpa**

**FROM Students s**

**JOIN Enrollments e ON s.student\_id = e.student\_id**

**GROUP BY s.student\_id, e.semester**

**ORDER BY s.student\_id, e.semester;**

**c).** summary reports on student performance:

* **Department Student Performance**: This view provides each department with a summary of student performance, including the average grade for each student in their enrolled courses and their overall GPA.

CREATE VIEW DepartmentStudentPerformance AS

SELECT d.department\_name, c.course\_name, s.student\_id, s.name, AVG(e.grade) AS avg\_grade, s.gpa

FROM Departments d

JOIN Students s ON d.department\_id = s.department\_id

JOIN Enrollments e ON s.student\_id = e.student\_id

JOIN Courses c ON e.course\_id = c.course\_id

GROUP BY d.department\_name, c.course\_name, s.student\_id, s.name;

summary reports on faculty teaching loads:

* **Faculty Teaching Load**: This view shows the number of courses each faculty member teaches within their department, which helps department heads monitor teaching loads.

CREATE VIEW FacultyTeachingLoad AS

SELECT d.department\_name, f.faculty\_id, f.name AS faculty\_name, COUNT(c.course\_id) AS number\_of\_courses

FROM Departments d

JOIN Faculty f ON d.department\_id = f.department\_id

JOIN Courses c ON f.faculty\_id = c.faculty\_id

GROUP BY d.department\_name, f.faculty\_id, f.name;

d). Trigger for GPA and Academic Standing Update

DELIMITER $$

CREATE TRIGGER update\_gpa\_and\_standing

AFTER INSERT OR UPDATE ON Enrollments

FOR EACH ROW

BEGIN

DECLARE total\_credits INT DEFAULT 0;

DECLARE total\_points DECIMAL(5,2) DEFAULT 0.0;

DECLARE calculated\_gpa DECIMAL(3,2);

DECLARE new\_academic\_standing VARCHAR(20);

-- Calculate total credits and grade points for the student

SELECT SUM(c.credits) INTO total\_credits

FROM Enrollments e

JOIN Courses c ON e.course\_id = c.course\_id

WHERE e.student\_id = NEW.student\_id;

SELECT SUM(e.grade \* c.credits) INTO total\_points

FROM Enrollments e

JOIN Courses c ON e.course\_id = c.course\_id

WHERE e.student\_id = NEW.student\_id;

-- Calculate GPA (grade points divided by total credits)

SET calculated\_gpa = total\_points / total\_credits;

-- Determine academic standing based on the calculated GPA

IF calculated\_gpa >= 3.0 THEN

SET new\_academic\_standing = 'Good Standing';

ELSEIF calculated\_gpa >= 2.0 THEN

SET new\_academic\_standing = 'Probation';

ELSE

SET new\_academic\_standing = 'At Risk';

END IF;

-- Update the Students table with the new GPA and academic standing

UPDATE Students

SET gpa = calculated\_gpa,

academic\_standing = new\_academic\_standing

WHERE student\_id = NEW.student\_id;

END$$

DELIMITER ;

 This trigger will ensure that each time a grade is inserted or updated, the GPA and academic standing remain accurate.

 Be cautious with triggers that have complex calculations, as they may impact database performance if the **Enrollments** table has many records.

This trigger provides data consistency and ensures that the student's performance is accurately reflected after every grade update

**Conclusion:-**

Overall, this database design supports comprehensive analysis, ensures accurate data maintenance, and automates key processes, providing university stakeholders with reliable tools to monitor and improve academic performance and resource management. This system serves as a valuable foundation for further enhancements, such as integrating predictive analytics or more advanced reporting features.