Lecture 4 Activity: Advanced SQL & Query Optimization
(Fast Learner)
Case Study: University Academic & Administrative Analytics

Scenario:

A large university maintains detailed records on students, courses, instructors, enrollment, exam results, department budgets, and faculty research projects. The university wants to perform deep analytics and optimizations on this data for strategic decisions.

Database Schema:

```
Students(StudentID, Name, DepartmentID, EnrollmentYear)
Departments(DepartmentID, DepartmentName, Budget)
Courses(CourseID, CourseName, DepartmentID, Credits)
Instructors(InstructorID, Name, DepartmentID)
Teaches(InstructorID, CourseID, Semester, Year)
Enrollments(EnrollmentID, StudentID, CourseID, Semester, Year)
Results(ResultID, EnrollmentID, Marks, Grade)
ResearchProjects(ProjectID, InstructorID, Title, StartDate, EndDate, Funding)
```

1. Students who have taken all courses offered by their department

```
SELECT s.StudentID, s.Name
FROM Students s
WHERE NOT EXISTS (
    SELECT c.CourseID
    FROM Courses c
    WHERE c.DepartmentID = s.DepartmentID
    EXCEPT
    SELECT e.CourseID
    FROM Enrollments e
    WHERE e.StudentID = s.StudentID
);
```

2. Instructors who taught more than 3 different courses in the last 2 years

```
SELECT i.Name, COUNT(DISTINCT t.CourseID) AS CourseCount
FROM Instructors i
JOIN Teaches t ON i.InstructorID = t.InstructorID
WHERE t.Year >= EXTRACT(YEAR FROM CURRENT_DATE) - 2
GROUP BY i.Name
HAVING COUNT(DISTINCT t.CourseID) > 3;
```

3. GPA per student per semester (Grade → Point mapping)

Assume grade mapping table:

```
GradePoints(Grade, Point)
    -- A=4, B=3, C=2, D=1, F=0
```

```
SELECT s.StudentID, s.Name, e.Year, e.Semester,
    ROUND(SUM(g.Point * c.Credits) / SUM(c.Credits), 2) AS GPA
FROM Students s
JOIN Enrollments e ON s.StudentID = e.StudentID
JOIN Results r ON r.EnrollmentID = e.EnrollmentID
JOIN GradePoints g ON r.Grade = g.Grade
JOIN Courses c ON e.CourseID = c.CourseID
GROUP BY s.StudentID, s.Name, e.Year, e.Semester;
```

4. Determine the most frequently taught course over all semesters.

```
SELECT c.CourseID, c.CourseName, COUNT(*) AS TimesTaught
FROM Teaches t
JOIN Courses c ON t.CourseID = c.CourseID
GROUP BY c.CourseID, c.CourseName
ORDER BY TimesTaught DESC
LIMIT 1;
```

5. Percentage of department budget spent on research

```
SELECT d.DepartmentName,
    ROUND(SUM(rp.Funding) / d.Budget * 100, 2) AS BudgetSpentPercentage
FROM Departments d
JOIN Instructors i ON d.DepartmentID = i.DepartmentID
JOIN ResearchProjects rp ON i.InstructorID = rp.InstructorID
GROUP BY d.DepartmentName, d.Budget;
```

6. Courses never enrolled in during last 3 years

```
SELECT c.CourseID, c.CourseName
FROM Courses c
WHERE NOT EXISTS (
  SELECT 1
  FROM Enrollments e
  WHERE e.CourseID = c.CourseID AND e.Year >= EXTRACT(YEAR FROM CURRENT_DATE) - 3
);
```

7. Identify instructors who have supervised projects and taught in the same semester and year.

```
SELECT DISTINCT i.Name, t.Semester, t.Year
FROM Instructors i
JOIN Teaches t ON i.InstructorID = t.InstructorID
JOIN ResearchProjects rp ON i.InstructorID = rp.InstructorID
WHERE EXTRACT(YEAR FROM rp.StartDate) <= t.Year
  AND (rp.EndDate IS NULL OR EXTRACT(YEAR FROM rp.EndDate) >= t.Year);
```

8. Count how many students from each department failed at least one course (Grade = 'F').

```
SELECT d.DepartmentName, COUNT(DISTINCT s.StudentID) AS FailedStudents
FROM Students s
JOIN Departments d ON s.DepartmentID = d.DepartmentID
JOIN Enrollments e ON s.StudentID = e.StudentID
JOIN Results r ON e.EnrollmentID = r.EnrollmentID
WHERE r.Grade = 'F'
GROUP BY d.DepartmentName;
```

9. Retrieve students who never failed any course throughout their academic history.

```
SELECT s.StudentID, s.Name
FROM Students s
WHERE NOT EXISTS (
  SELECT 1
  FROM Enrollments e
  JOIN Results r ON e.EnrollmentID = r.EnrollmentID
  WHERE e.StudentID = s.StudentID AND r.Grade = 'F'
);
```