```sql
-- ================================================
-- FILE: employee-database-analysis/01_salary_trends.sql
-- ================================================

-- Business Question: How have average salaries changed over time?
-- Analysis: Historical salary trends to understand compensation evolution

SELECT
    YEAR(from_date) AS year,
    ROUND(AVG(salary)) AS average_salary
FROM salaries
WHERE from_date <= '2005-01-01'
GROUP BY year
ORDER BY year DESC;

-- Business Insight:
-- This query helps identify salary inflation trends and can inform
-- future compensation planning and budget forecasting.

-- ================================================
-- FILE: employee-database-analysis/02_department_analysis.sql
-- ================================================

-- Business Question: Which departments offer the highest current compensation?
-- Analysis: Current average salary by department for active employees

SELECT
    d.dept_name AS department,
    ROUND(AVG(s.salary)) AS average_salary
FROM departments d
JOIN dept_emp de ON de.dept_no = d.dept_no
JOIN salaries s ON s.emp_no = de.emp_no
WHERE s.to_date >= CURDATE()
  AND de.to_date >= CURDATE()
GROUP BY d.dept_name
ORDER BY average_salary DESC;

-- Business Insight:
-- Identifies high-value departments for talent attraction and
-- helps ensure competitive compensation across the organization.

-- ================================================
-- FILE: employee-database-analysis/03_salary_by_year_dept.sql
-- ================================================

-- Business Question: How do salary trends vary by department over time?
-- Analysis: Year-over-year salary trends broken down by department
```

```
SELECT
    YEAR(s.from_date) AS year,
    d.dept_name AS department,
    ROUND(AVG(s.salary)) AS average_salary
FROM departments d
JOIN dept_emp de ON de.dept_no = d.dept_no
JOIN salaries s ON s.emp_no = de.emp_no
GROUP BY year, department
ORDER BY year, department;

-- Business Insight:
-- Reveals department-specific salary evolution patterns
-- and helps identify departments with different growth trajectories.

-- ================================================
-- FILE: employee-database-analysis/04_large_departments.sql
-- ================================================

-- Business Question: Which departments have the largest workforce?
-- Analysis: Identify departments with more than 15,000 active employees

SELECT
    d.dept_no,
    d.dept_name,
    COUNT(de.emp_no) AS employee_count
FROM departments d
JOIN dept_emp de ON d.dept_no = de.dept_no
WHERE de.to_date >= CURDATE()
GROUP BY d.dept_no, d.dept_name
HAVING COUNT(de.emp_no) > 15000
ORDER BY employee_count DESC;

-- Business Insight:
-- Large departments may require specialized management structures
-- and represent significant operational and budget responsibility.

-- ================================================
-- FILE: employee-database-analysis/05_senior_manager.sql
-- ================================================

-- Business Question: Who is the longest-serving active manager?
-- Analysis: Find the manager with earliest hire date among current managers

SELECT
    CONCAT(e.first_name, ' ', e.last_name) AS full_name,
    e.emp_no,
    d.dept_name,
    t.title,
```

```
    e.hire_date
FROM employees e
JOIN dept_emp de ON de.emp_no = e.emp_no
JOIN departments d ON d.dept_no = de.dept_no
JOIN titles t ON t.emp_no = e.emp_no
WHERE t.title = 'Manager'
  AND de.to_date >= CURDATE()
  AND t.to_date >= CURDATE()
ORDER BY hire_date ASC
LIMIT 1;

-- Business Insight:
-- Senior managers represent institutional knowledge and
-- can be valuable mentors for management development programs.

-- ================================================
-- FILE: employee-database-analysis/06_salary_comparison.sql
-- ================================================

-- Business Question: Which employees earn significantly above/below their department
average?
-- Analysis: Compare individual salaries to department averages using window functions

WITH salary_analysis AS (
    SELECT
        e.emp_no,
        CONCAT(e.first_name, ' ', e.last_name) AS full_name,
        s.salary,
        d.dept_name,
        ROUND(AVG(s.salary) OVER (PARTITION BY d.dept_name), 1) AS avg_dept_salary
    FROM employees e
    JOIN salaries s ON s.emp_no = e.emp_no
    JOIN dept_emp de ON de.emp_no = e.emp_no
    JOIN departments d ON de.dept_no = d.dept_no
    WHERE s.to_date >= CURDATE()
      AND de.to_date >= CURDATE()
),
salary_differences AS (
    SELECT
        full_name,
        salary,
        dept_name,
        avg_dept_salary,
        ROUND(salary - avg_dept_salary, 1) AS salary_difference
    FROM salary_analysis
)
SELECT *
FROM salary_differences
```

```
ORDER BY salary_difference DESC
LIMIT 10;

-- Business Insight:
-- Identifies high performers who may deserve recognition or retention efforts,
-- and helps ensure equitable compensation within departments.

-- ================================================
-- FILE: employee-database-analysis/07_second_managers.sql
-- ================================================

-- Business Question: Who was the second manager hired in each department?
-- Analysis: Historical management succession using window functions

WITH manager_history AS (
    SELECT
        d.dept_name,
        CONCAT(e.first_name, ' ', e.last_name) AS full_name,
        e.hire_date,
        dm.from_date,
        ROW_NUMBER() OVER (
            PARTITION BY d.dept_name
            ORDER BY dm.from_date
        ) AS manager_sequence
    FROM employees e
    JOIN dept_manager dm ON dm.emp_no = e.emp_no
    JOIN departments d ON dm.dept_no = d.dept_no
)
SELECT
    dept_name,
    full_name,
    hire_date,
    from_date AS management_start_date
FROM manager_history
WHERE manager_sequence = 2
ORDER BY dept_name;

-- Business Insight:
-- Understanding management succession patterns helps with
-- leadership development and succession planning strategies.

-- ================================================
-- FILE: course-management-system/01_create_database.sql
-- ================================================

-- Course Management System Database Design
-- Purpose: Educational institution management system
```

```
DROP DATABASE IF EXISTS CourseManagement;
CREATE DATABASE IF NOT EXISTS CourseManagement;
USE CourseManagement;

-- Teachers table: Store instructor information
CREATE TABLE IF NOT EXISTS teachers (
    teacher_no INT AUTO_INCREMENT,
    teacher_name VARCHAR(50) NOT NULL,
    phone_no VARCHAR(20),
    PRIMARY KEY (teacher_no)
);

-- Courses table: Course catalog and scheduling
CREATE TABLE IF NOT EXISTS courses (
    course_no INT AUTO_INCREMENT,
    course_name VARCHAR(100) NOT NULL,
    start_date DATE,
    end_date DATE,
    PRIMARY KEY (course_no)
);

-- Students table: Student enrollment and assignment
CREATE TABLE IF NOT EXISTS students (
    student_no INT AUTO_INCREMENT,
    teacher_no INT,
    course_no INT,
    student_name VARCHAR(50) NOT NULL,
    email VARCHAR(100),
    birth_date DATE,
    PRIMARY KEY (student_no),
    FOREIGN KEY (teacher_no) REFERENCES teachers(teacher_no),
    FOREIGN KEY (course_no) REFERENCES courses(course_no)
);

-- ================================================
-- FILE: course-management-system/02_insert_data.sql
-- ================================================

-- Sample Data Insertion with Transaction Management
-- Ensures data consistency across related tables

START TRANSACTION;

-- Insert teacher data
INSERT INTO teachers (teacher_name, phone_no)
VALUES
    ('John Smith', '050-123-4567'),
    ('Sarah Johnson', '067-890-1234'),
```

```
    ('Michael Brown', '093-456-7890');

-- Insert course catalog
INSERT INTO courses (course_name, start_date, end_date)
VALUES
    ('Mathematics', '2022-01-01', '2022-05-31'),
    ('English Language', '2022-02-01', '2022-06-30'),
    ('Physics', '2022-03-01', '2022-07-31');

-- Insert student enrollments
INSERT INTO students (teacher_no, course_no, student_name, email, birth_date)
VALUES
    (1, 1, 'Alexander Petrov', 'petrov@example.com', '1990-01-01'),
    (1, 1, 'Olga Ivanova', 'ivanova@example.com', '1992-02-02'),
    (2, 2, 'Andrew Kovalenko', 'kovalenko@example.com', '1995-03-03'),
    (2, 2, 'Catherine Petrenko', 'petrenko@example.com', '1998-04-04'),
    (3, 3, 'Michael Shevchenko', 'shevchenko@example.com', '2000-05-05'),
    (3, 3, 'Jane Kovalenko', 'jane.kovalenko@example.com', '2002-06-06'),
    (1, 1, 'Victoria Petrova', 'petrova@example.com', '2005-07-07'),
    (2, 2, 'Maxim Kovalyov', 'kovalyov@example.com', '2008-08-08'),
    (3, 3, 'Elena Petrenko', 'elena.petrenko@example.com', '2010-09-09'),
    (1, 1, 'Anna Shevchenko', 'anna.shevchenko@example.com', '2012-10-10');

COMMIT;

-- ================================================
-- FILE: course-management-system/03_analysis_queries.sql
-- ================================================

-- Business Analytics for Course Management System

-- Query 1: Teacher workload analysis
-- Business Question: How many students is each teacher responsible for?
SELECT
    t.teacher_name,
    COUNT(s.student_no) AS student_count
FROM teachers t
JOIN students s ON t.teacher_no = s.teacher_no
GROUP BY t.teacher_name, t.teacher_no
ORDER BY student_count DESC;

-- Query 2: Course enrollment summary
-- Business Question: Which courses have the highest enrollment?
SELECT
    c.course_name,
    COUNT(s.student_no) AS enrollment_count,
    c.start_date,
    c.end_date
```

```
FROM courses c
LEFT JOIN students s ON c.course_no = s.course_no
GROUP BY c.course_no, c.course_name, c.start_date, c.end_date
ORDER BY enrollment_count DESC;

-- Query 3: Student age demographics
-- Business Question: What is the age distribution of our students?
SELECT
    CASE
        WHEN TIMESTAMPDIFF(YEAR, birth_date, CURDATE()) < 18 THEN 'Under 18'
        WHEN TIMESTAMPDIFF(YEAR, birth_date, CURDATE()) BETWEEN 18 AND 25
THEN '18-25'
        WHEN TIMESTAMPDIFF(YEAR, birth_date, CURDATE()) BETWEEN 26 AND 35
THEN '26-35'
        ELSE 'Over 35'
    END AS age_group,
    COUNT(*) AS student_count
FROM students
GROUP BY age_group
ORDER BY student_count DESC;

-- Query 4: Teacher-Course assignments
-- Business Question: Which teachers are assigned to which courses?
SELECT
    t.teacher_name,
    c.course_name,
    COUNT(s.student_no) AS students_enrolled
FROM teachers t
JOIN students s ON t.teacher_no = s.teacher_no
JOIN courses c ON s.course_no = c.course_no
GROUP BY t.teacher_name, c.course_name
ORDER BY t.teacher_name, students_enrolled DESC;
```