-- 1. Select all employees from the 'Engineering' department.

π\_emp\_id, first\_name, last\_name, salary, department\_id (σ\_department\_name='Engineering'(employees))

-- 2. Perform a projection to display only the first names and salaries of all employees.

π\_first\_name, salary(employees)

-- 3. Find employees who are managers.

σ\_manager\_id IS NOT NULL(employees)

-- 4. Retrieve employees earning a salary greater than $60000.

σ\_salary > 60000(employees)

-- 5. Join employees with their respective departments.

employees ⨝ department\_id = department\_id(departments)

-- 6. Perform a Cartesian product between employees and projects.

employees × projects

-- 7. Find employees who are not managers.

employees - (σ\_manager\_id IS NOT NULL(employees))

-- 8. Perform a natural join between departments and projects.

departments ⋈ projects

-- 9. Project the department names and locations from departments table.

π\_department\_name, location(departments)

-- 10. Retrieve projects with budgets greater than $100000.

σ\_budget > 100000(projects)

-- 11. Find employees who are managers in the 'Sales' department.

π\_emp\_id, first\_name, last\_name, salary, department\_id (σ\_department\_name='Sales' ^ manager\_id IS NOT NULL(employees))

-- 12. Perform a union operation between two sets of employees from the 'Engineering' and 'Finance' departments.

π\_emp\_id, first\_name, last\_name, salary, department\_id (σ\_department\_name='Engineering'(employees)) ∪ π\_emp\_id, first\_name, last\_name, salary, department\_id (σ\_department\_name='Finance'(employees))

-- 13. Find employees who are not assigned to any projects.

employees - (employees ⨝ employees.emp\_id = projects.emp\_id(projects))

-- 14. Join operation to display employees along with their project assignments.

employees ⨝ employees.emp\_id = projects.emp\_id(projects)

-- 15. Find employees whose salaries are not within the range $50000 to $70000.

σ\_salary < 50000 OR salary > 70000(employees)