GreenGreenGreenGreenGreenGreenGreenGreen
This is actually three questions; you should ask as many as you have time to. All of them use the following schema

Table: Employees
employee_id (int),
employee_name (varchar)
salary (int),
department_id (int)

Table: Projects
project_id (int),
project_name (int),
start_date (date),
end_date (date),
budget (int)

Table: Departments
department_id (int),
department_name (varchar)

Table: Employees_projects
project_id (int),
employee_id (int)

(Something you should be aware of: there are three values which are shared across tables: employee_id, department_id, and project_id. There are what we're going to join on. The first three tables have primary keys of employee_id, project_id, and department_id respective; the fourth does not and has no uniqueness guarantees. This is not particularly relevant to the problem)

Q1: Write a query to list the departments that have a total combined salary greater than $40,000. Output should be of the form: department_name, total_salary

First, we want to find the total combined salary by department. This is best done by rolling up the Employees table by department_id:

```
select department_id, sum(salary) as total_salary
from Employees
group by department_id
having sum(salary) > 40000
```

However, we want the department name and not the ID. So we need to join Departments on department_id.

```
select department_name, total_salary
from (
  select department_id, sum(salary) as total_salary
  from Employees
  group by department_id
) emp
join Departments dep
on emp.department_id = dep.department_id
```

The important thing here is that we used the table just generated. We could combine them into this query:

```
select department_name, sum(salary) as total_salary
from Employees emp
join Departments dep
on emp.department_id = dep.department_id
group by department_name
```

which is less efficient if Employees and Departments are large tables and a join is expensive but has the advantage of showing departments with no employees as having zero budget (a bit of an edge case). In the first query, a department with no employees would not appear at all! We could resolve this several ways, but the most elegant is to use a left join:

```
select department_name, case when total_salary is null then 0 else total_salary end as
total_salary
from Departments dep
left join (
  select department_id, sum(salary) as total_salary
  from Employees
  group by department_id
) emp
on dep.department_id = emp.department_id
```

Notice that I reverse the orders of the tables. You can preserve them and do a right join; my default is to only do left joins and reverse the tables as appropriate.

Q2: List the current projects and employees assigned to them.

Students should ask "what format?" What we want is names: Project Name and Employee Name. This means joining three tables: Employees, Projects, and Employee_projects. Multiple solutions exist; most will start by joining two tables

```
select employee_name, project_id
from Employees emp
join Employee_projects emp_pro
on emp.employee_id = emp_pro.employee_id
```

and then turning this into a subquery and joining the third

```
select employee_name, project_name
from (
   select employee_name, project_id
   from Employees emp
   join Employee_projects emp_pro
   on emp.employee_id = emp_pro.employee_id
   ) wob
join Projects pro
on wob.project_id = pro.project_id
```

Q3: Find the highest paid person per department

Things people should ask: We want the department name and employee name; the highest paid person is guaranteed to be unique.

This is tricky; the hard part is getting the highest salary per department_id out of Employees. There are multiple ways to do this; the most straightforward is probably a self-join

```
select emp1.department_id, employee_name
from Employees emp1
join (
  select department_id, max(salary) as highest_pay
  from Employees
  group by department_id
   ) emp2
on emp1.department_id = emp2.department_id
and emp1.salary = emp2.highest_pay
```

Notice that we must specify which table department_id is drawn from, as this field name is duplicated. Even though emp2.department_id is ALWAYS THE SAME, we have to do it. We've just been able to avoid this before by not picking duplicated fields in our joins.

Or with a partition:

```
select department_id,
        employee_name,
        max(salary) over(partition by department_id) as top_sal
from Employees
where top_sal = salary
```

Either way, we complete our nesting and finish off by joining Departments to get the name.

```
select department_name, employee_name
from (
select emp1.department_id, employee_name
from Employees emp1
join (
  select department_id, max(salary) as highest_pay
  from Employees
   ) emp2
on emp1.department_id = emp2.department_id
and emp1.salary = emp2.highest_pay
) xxx
join Departments dep
on xxx.department_id = dep.department_name
```

(Alternate: use RANK() to pick the highest salary out of each department.)

```
SELECT department_name, employee_name
  FROM Departments
  JOIN (SELECT department_id,
               employee_name,
               RANK() OVER(PARTITION BY department_id
                           ORDER BY salary DESC) rank
          FROM Employees
        HAVING rank = 1)
```