SQL Training, Session 3

This lesson assumed that you have completed the following w3 Schools tutorials:
SQL SELF JOIN
SQL UNION
SQL GROUP BY
SQL HAVING
SQL CASE

In our last session, we covered up to left join so far. We are going to extend our SQL toolkit by learning the framework to process data.

The self-join is a powerful join that allows a user to join a table with its self. This can make certain calculations such as computing percentages or finding entries that have common values much easier. Let's dive into an example. Recall our employees sample data.

emp_id	emp_name	age	annual_salary	tenure	department
101	Sally Johnson	29	50000	1	IT
102	Tom Brooks	45	80000	4	analytics
103	Tom Felton	27	50000	1	IT
104	Jack Kelvin	55	85000	5	Sales
105	Beth George	35	70000	3	Sales
106	Rusty	65	120000	4	executive

I) Write a query that matches employees that are from the same department. Your query should return emp id, emp name, and department.

You will probably notice that even though we get our query up and running, we run into permutations of each employee name. Solving this problem requires advanced knowledge of "window" functions covered in future sessions.

Now let's use self-join to compute a percentage of something across the employees table.

II) Assume that the company only consists of the 6 employees listed on the table. Compute the percentage of employees by department for the company. Your query should only return department and percent employees.

Let's now work with UNION. UNION is an operator that allows you to combine the results from separate select statements IF AND ONLY IF the columns selected have similar data types and order of columns in each SELECT are the same.

We now have a table of incoming new hires based on the expansion needs of the company.

new_emp_id	new_emp	age	starting_salary	department
107	George Riley	54	80000	Sales
108	Jennifer Marcus	23	50000	IT
109	Jimmy Olsen	55	85000	analytics
110	Lois Lane	34	50000	IT
111	mark jacks	28	50000	Sales

III) Use a union all to make a new list of all employees including new hires. Your query should only return employee Id and employee name. Assume the new hires table is called 'new_hires'.

HAVING is an operator that allows you to filter results by a certain condition that could not be captured using a WHERE clause. This mainly applies when using aggregate functions. Let's see an example.

IV) Generate a list of departments that only have 2 or less employees from the emp_table and the new_hires table. You should add some sort of flag to identify the result from new_hires and emp_table. Your query should only return department and flag.

We conclude this lesson with CASE. CASE allows you to return a value WHEN some condition is met. If no condition is met, then it returns the value in the ELSE clause.

The format is CASE WHEN <condition> THEN <value> ELSE <other value> END

Case when is especially useful if you need to rename entries in a column.

V) Use a case when statement to identify employees from the emp_table who qualify for a raise. Identify these employees with some flag. In order to qualify for a raise, the employee must have worked 2 years or more. your query should return emp ID, emp name, annual salary, and flag. Do not exclude any employees regardless if they get a raise or not.

VI) Lets say the raise consists of a 3 percent increase of their annual salary. Compute a new annual salary for employees who qualify for a raise but do not exclude employees who did not get a raise. Your query should return employee id, employee name, and new annual salary. (leave salary unchanged for emps that did not get a raise within the same column)