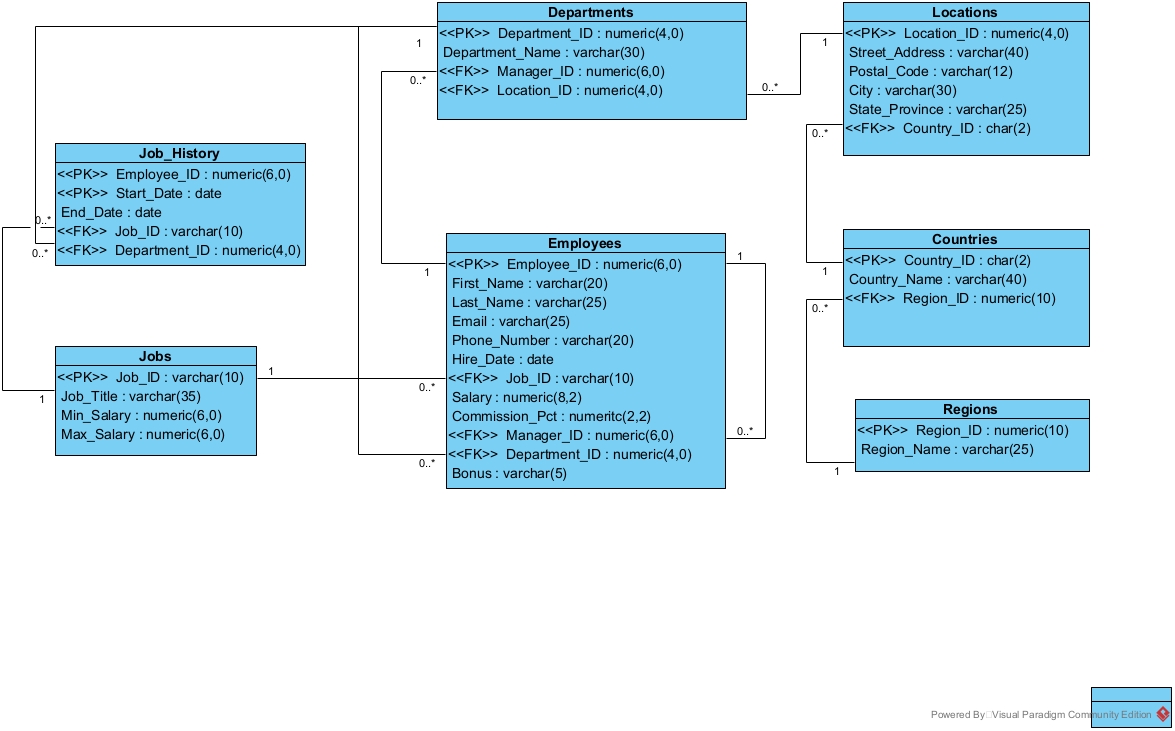
Databases 6G4Z0016 Labsheet

# Topic 4 – GROUP BY



# Worked Examples

### Q1. Give the total amount that the company pays out in wages every month.

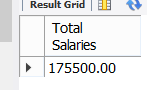
From the ERD we know that the relevant column is the salary column in the Employees table. In order to give the total amount, we will have to use a group function to combine all the salary values into one value. Specifically, we will need the SUM() function. The rest of the query is a simple SELECT, FROM query. The SQL we need is:

SELECT SUM(salary) AS ‘Total Salaries’

FROM Employees;

Note that we have provided an alias for the result because otherwise the column name would be SUM(salary) which is not very user-friendly.

The result should be:



### Q2. Give the average bonus for all employees.

This question appears similar to the previous one but this time it is asking for average instead of total and the column is bonus not salary. The function for average is simply AVG() so we might try the following SQL:

SELECT AVG(bonus) AS ‘Average Bonus’

FROM Employees;

Try this query – you should get a result with lots of decimal places. We can solve that problem by using the ROUND() function and specifying how many decimal places we want (perhaps two because it is a monetary value).

The real problem, however, is that the answer is incorrect. Run SELECT \* FROM Employees; and see how many employees receive a bonus and the amounts. You will see that only three employees receive a bonus, and their bonuses are 1500, 1700 and 1250. When you average these you will get 1483.33 which is what the query above returns.

The point here is that the values in the other rows are all NULL and NULL is a special value all of its own. When you run a group function like SUM or AVG on a column with NULLs, the NULL values are ignored. They are not treated as zeros, they are treated as if they do not exist.

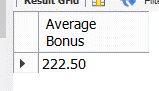
In this case, we would be better off treating the NULL values as zeros because those employees are not receiving a bonus. To do that we must use a special function IFNULL(). It’s a function that allows you to replace the value of a NULL with a value you specify. It requires two inputs – the column to test for NULLS and the value to use instead of a NULL.

Applying that to our question we have the final SQL:

SELECT ROUND(AVG(IFNULL(bonus,0)),2) AS ‘Average Bonus’

FROM Employees;

The result of this is:



### Q3. List the average salary for each job.

This question is a little more complex than the previous ones because it asks not for a single average but for an average for each job. The fact that we need to compute some value for a group of rows tells us that we need a GROUP BY clause to make sure that the average is calculated per group.

There is an additional complication in that we want to give the salary for each job. The salary column is in the Employees table, which also has a column for job\_id. But it isn’t useful to the end user to return an ID. Instead, we should return the actual job title, which is in the jobs table. We will therefore have to perform an inner join between Employees and Jobs before applying the GROUP BY.

The SQL we need is:

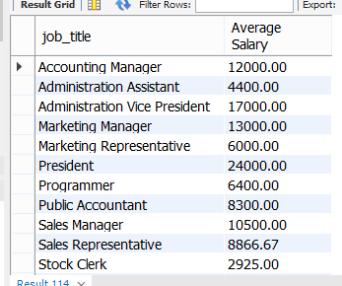
SELECT job\_title, ROUND(AVG(salary),2) AS ‘Average Salary’

FROM Employees

INNER JOIN Jobs USING (job\_id)

GROUP BY job\_title;

The result should be:



Note that the column we group by is job\_title, which is the column we include in the output. We know that there is a one-to-one mapping between job\_title and job\_id and so we might be tempted to use GROUP BY job\_id. If we try this in some systems (like Oracle) we will get an error (e.g. “not a GROUP BY expression”). This is because the database system cannot guarantee that there is only one value of job\_title for each value of job\_id. If we did want to list both job\_id and job\_title in the output, we could GROUP BY both of them, using:

SELECT job\_id, job\_title,

ROUND(AVG(salary),2) AS ‘Average Salary’

FROM Employees

INNER JOIN Jobs USING (job\_id)

GROUP BY job\_id, job\_title;

### Q4. List the number of employees in each department.

This is similar to the previous question except that instead of grouping by job title we will group by department and instead of calculating an average of something per group we instead want to count the number of rows per group. We therefore use the COUNT function not the AVG function. We will still need a GROUP BY and also an inner join with the Departments table so that we can return the department name rather than an ID because IDs are not useful for end users.

Our query should be:

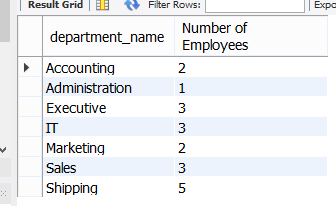
SELECT department\_name, COUNT(\*) AS ‘Number of Employees’

FROM Employees

INNER JOIN Departments USING (department\_id)

GROUP BY department\_name;

The result should be:



Notice that we use a wildcard in the COUNT function, COUNT(\*) means “count all”. This is because of the problem with NULL values being ignored by functions. To count the total number of employees we could use “SELECT COUNT(first\_name) FROM Employees;” which would return 20. But if we instead used the bonus column (SELECT COUNT(bonus) FROM Employees;) we would get the answer 3. This is because NULL values are ignored by the COUNT function.

To avoid having to ensure that the row we pick to count with has no NULL values, we can use COUNT(\*) which will count using all columns so that if one column has a NULL value in a given row but a different column does not, the row is still counted.

COUNT(\*) is the standard method of counting the rows in a table. However, sometimes we do want to focus on a specific column only. One example is when we want to count the number of unique values in a column. Clearly, using COUNT(\*) won’t help us in this situation. However, the regular COUNT(col\_name) is also not sufficient because it won’t take into account repeating values.

To count only the distinct values in a column use COUNT(DISTINCT col\_name).

### Q5. List the average salary of departments with at least three employees.

Because the question asks for the average, we know we need to use the AVG function. Because it asks for the information broken down by department, we know we need to use a GROUP BY. The only difference between this question and previous ones is that it wants us to limit the results we return to only those where the department has at least three employees.

Normally when we want to limit the output we use the WHERE clause, but that works on a row-by-row basis. In this case, we want to limit the output based on the entire group. For this we use the HAVING clause. The condition for inclusion in the output is that the department has at least three employees so we will have to use a COUNT function to count the number of employees in each group.

We also must remember the join with the Departments table, so that we can return the department’s name not its ID. And finally, remember to round off the average to two decimal places.

The final SQL is therefore:

SELECT department\_name, ROUND(AVG(salary),2) AS ‘Average Salary’

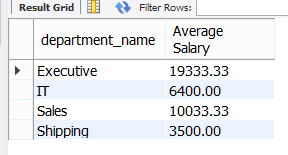
FROM Employees

INNER JOIN Departments USING (department\_id)

GROUP BY department\_name

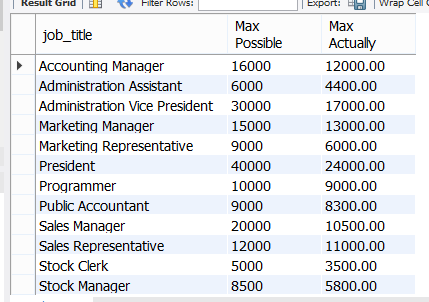
HAVING COUNT(\*) > 2;

The results should be:



# Completion Problems

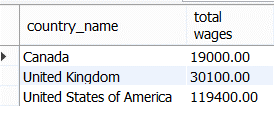
### Q1. List the job titles, the maximum salary available for that job and the maximum an employee in that job is actually earning.

SELECT job\_title, max\_salary AS ‘Max Possible’,   
 MAX(salary) AS ‘Max Actually’   
FROM Employees   
INNER JOIN Jobs USING (job\_id)   


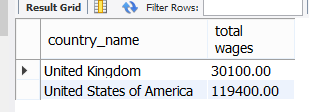
### Q2. List the names of the departments and the countries they are in. Include all countries, even if no departments are based in them yet.

SELECT department\_name, country\_name   
FROM Departments   
INNER JOIN Locations USING (location\_id)   


### Q3. List the total wages paid out to employees in each country.

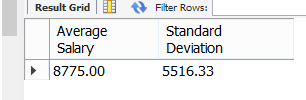
SELECT country\_name   
FROM Employees   
INNER JOIN Departments USING(department\_id)   
INNER JOIN Locations USING (location\_id)   
INNER JOIN Countries USING (country\_id)   


### Q4. List the total wages paid out to employees in each country where the total paid out is more than 20,000.

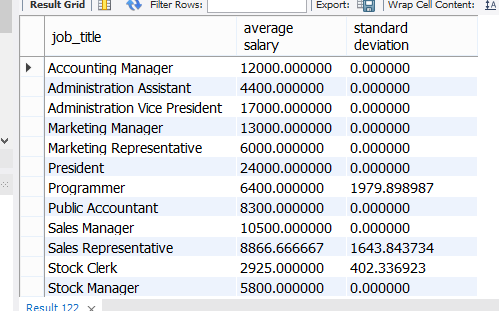
SELECT country\_name, SUM(salary) AS ‘total wages’   
INNER JOIN Departments USING(department\_id)   
INNER JOIN locations USING (location\_id)   
INNER JOIN Countries USING (country\_id)   
GROUP BY country\_name   


### Q5. Find the average salary of all employees and the standard deviation.

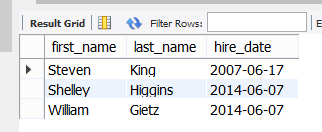
SELECT   
FROM Employees;



### Q6. Find the average salary and standard deviation for each job type.

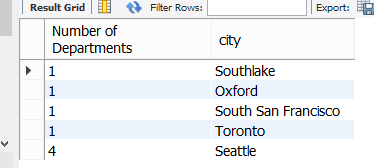
SELECT job\_title, STDDEV(salary) AS ‘standard deviation’   
FROM Employees   
INNER JOIN Jobs USING (job\_id)   


### Q7. List all employees hired during June in any year.

SELECT first\_name, last\_name, hire\_date   
FROM Employees   


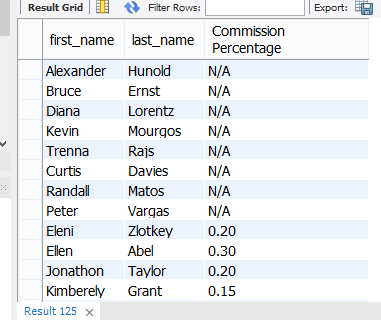
### Q8. List the number of departments in each city from smallest to largest.

SELECT COUNT(\*) AS ‘Number of Departments’, city   
FROM Departments   
ORDER BY COUNT(\*);



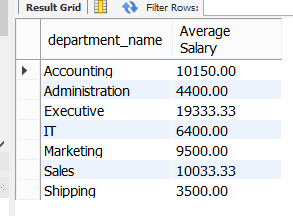
### Q9. List all the employees and their commission percentage, using ‘N/A’ for those not earning commission.

SELECT first\_name, last\_name,   
 AS ‘Commission Percentage’  
FROM Employees;

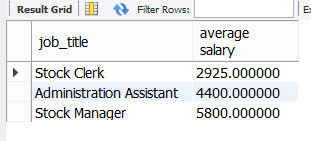


# Deliberate Practice: Write the SQL

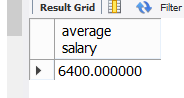
### Q1. List the average salary for each job in each department



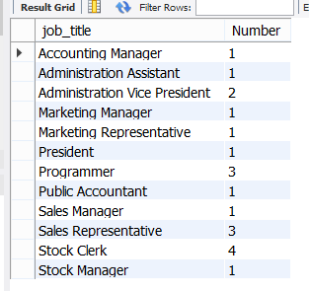
### Q2. List the three lowest paid jobs, on average, in the company



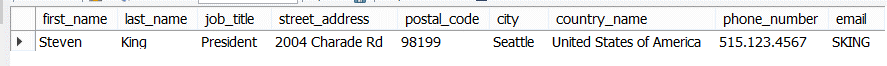
### Q3. What is the average salary in the IT departments?



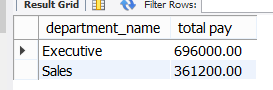
### Q4. List the number of people in each job



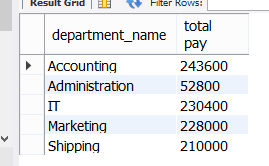
### Q5. Give the wording that will need to be put on the President’s business card: his name, position, full address (including country name), telephone number and email address.



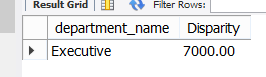
### Q6. List the departments where the total salaries (excluding bonuses) per year are more than 250,000.



### Q7. List the departments where the total salaries (including bonuses) per year are LESS than 250,000.



### Q8. Give the name of the department with the largest disparity in salaries.



### Q9. List the departments and the names of their managers, with the names combined into a single column called “Manager”. If a department does not have a manager, the name should read “NO MANAGER” instead.

