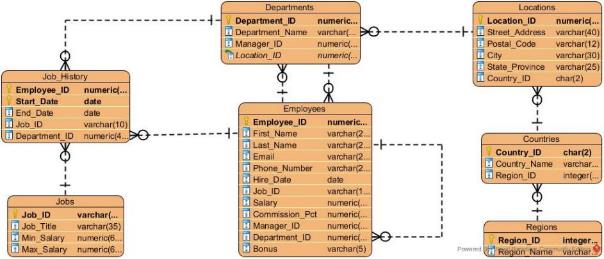
# Topic 3 – The WHERE Clause and Ordering

## Familiarise yourself with the Human Resource (HR) schema from Oracle

Whereas in the lectures we use a small database consisting of only three table, for the labs we will be using the demo tables (schema) supplied by Oracle. The schema describes a human resources database used by a large multi-national company.

Use the Entity Relationship Diagram (ERD) below to familiarise yourself with the tables, relationships and attributes.



# Worked Examples

### Q1. List the employees earning more than 10,000.

This question is asking us to list employees so we clearly need to use the employees table. It asks for those earning more than 10,000 and so we immediately know we need to use a WHERE clause because we only want to return some of the rows and not all of them. The columns we need are not fully defined but we can reasonably suppose that we need to return first\_name, last\_name and salary. We need salary because it is useful to have the column we use in the WHERE clause returned in the output, though it is not actually required. The filter condition is that the salary must be more than 10,000 so we use the simple greater than comparison operator.

The final SQL query is therefore:

SELECT first\_name, last\_name, salary

FROM employees

WHERE salary > 10000;

The output is:



### Q2. List the employees earning more than 10,000 in order of their salary.

This is essentially the same as the previous question plus some ordering. Notice that in the output from the previous question Eleni Zlotkey was returned before Ellen Abel even though Eleni’s salary is lower than Ellen’s. In this question we need to impose an order on the result using the ORDER BY clause.

The SQL is therefore:

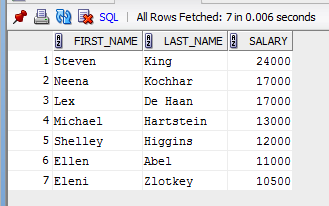
SELECT first\_name, last\_name, salary

FROM employees

WHERE salary > 10000

ORDER BY salary DESC;

The output is:



Try the query again but remove the DESC keyword – notice how the default behaviour is to order the results in ascending order.

### Q3. List the job titles with “Manager” in the title

This question refers us to the jobs table which contains the job\_title column. It is asking for only some of the rows (those containing the word “Manager”) so we know we need a WHERE clause. Also, because it is asking us to perform some pattern matching we know we will need a LIKE clause.

We might try:

SELECT job\_title

FROM jobs

WHERE job\_title LIKE “Manager”;

But if you try this (and please do) you will see that nothing is returned. The problem is that the word “Manager” appears in the middle of the job title. In order to match it we have to use a wildcard to create a pattern that can match any string with the word “Manager” in it. Since we want to match any string, that implies any length as well. So of the two wildcards (\_ and %) we have to use % which matches any string of any length, including zero-length. We therefore create a pattern of “%Manager%” which will match any string containing the word “Manager” anywhere inside it.

The final SQL is therefore:

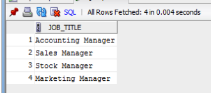
SELECT job\_title

FROM jobs

WHERE job\_title

LIKE '%Manager';

And the results are:



Remember that pattern matching is case-sensitive. Try the following query and notice that this time nothing is returned:

SELECT job\_title

FROM jobs

WHERE job\_title

LIKE '%manager';

Q4. List the employees, their jobs and salaries and make sure the employees with the same jobs are in consecutive rows ordered by their salary

This question refers to the employees table and requires the columns first\_name, last\_name, job\_id and salary. The tricky part here is to make sure that the employees with the same jobs appear in consecutive rows. Fortunately, we can enforce this by ordering the results by job\_id as this will make sure that rows with the same job\_id are next to each other.

The question also asks us to order by their salary so that if two employees in the same job have different salaries they are ordered by their salary. So although it doesn’t say so explicitly, the question requires us to order based on two columns. This will order by the first one first and then within each ordered group (i.e. where job\_id is the same) it will order again based on the second column.

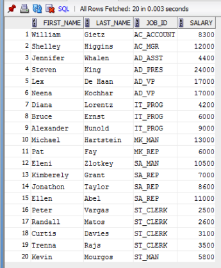
The final SQL is therefore:

SELECT first\_name, last\_name, job\_id, salary

FROM employees

ORDER BY job\_id, salary;

The results are:



### Q5. List all employees who are Managers

This question refers us to the employees table. We know if an employee is a manager from their job\_id. If the database had been constructed with some consistency then all manager positions would have a job\_id with either “\_MGR” or “\_MAN” in it but that hasn’t happened. Instead we have a mix of both.

We therefore need to provide a range of options in the WHERE clause. One option is to use LIKE and OR to give:

SELECT first\_name, last\_name, job\_id

FROM employees

WHERE job\_id LIKE ‘%MGR’ OR job\_id LIKE ‘%MAN’;

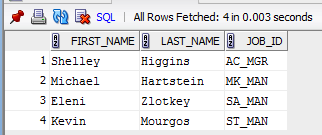
Another option is to use the IN keyword and have:

SELECT first\_name, last\_name, job\_id

FROM employees

WHERE job\_id IN (‘AC\_MGR’, ’SA\_MAN’, ’ST\_MAN’, MK\_MAN’);

Both should give the same results:



Since both give the same result they are both correct, and this is a feature of SQL – that you can write more than one query that does the same thing. My preference would be for the second option because it avoids using lots of LIKE statements which can be confusing. Instead it is a bit more verbose but that brings clarity.

### Q6. List all locations in the US

This is a straightforward SELECT and WHERE query. The locations are all in the locations table and that table contains a column called country\_id which we can use to limit the results to only those locations in the US. Don’t be confused by the use of the word “IN” in the question. Here we do not want to use the IN keyword in the WHERE clause. Instead, we simply say that the country\_id must be equal to “US”.

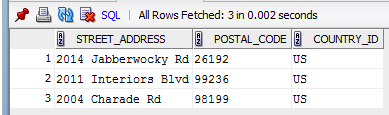
The final SQL is therefore:

SELECT street\_address, city, country\_id

FROM locations

WHERE country\_id = ‘US’;

The results are:



### Q7. List the employees whose commission percentage is not 0.2

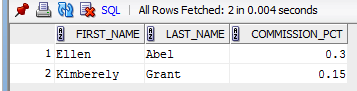
This is again a straightforward SELECT and WHERE. The SQL we need is:

SELECT first\_name, last\_name, commission\_pct

FROM employees

WHERE commission\_pct != 0.2;

Pay careful attention to the results however:



**NEW MATERIAL**

Notice that only two rows are returned. If you run SELECT \* FROM employees; you will see that the majority of employees have (null) as their commission\_pct. Clearly (null) is not 0.2 so why were they not returned?

The answer is that (null) or NULL is a special value in relational databases. It is not the same as a zero, or a blank string. It is its own special value that means NULL. When you run a query like the one above and compare NULL to something, the result is also NULL. So when you try “NULL != 0.2” the answer is not true or false, it is NULL and therefore the row is not returned. NULL values are so special that they are not even equal to each other. If you try the following query you will get nothing returned because even the rows containing NULL values are not equal to NULL:

SELECT first\_name, last\_name, commission\_pct

FROM employees

WHERE commission\_pct = NULL;

The correct way to retrieve all the rows containing NULL values is to use the special condition “IS NULL”. Try the following query:

SELECT first\_name, last\_name, commission\_pct

FROM employees

WHERE commission\_pct IS NULL;

Since NULL values are not equal or unequal to anything, all filter conditions will remove them except “IS NULL”. However, if you only want to remove rows that contain NULL values and have no other filter conditions, then the “IS NOT NULL” condition will remove any rows containing NULL values in the specified column.

### Q8. List all jobs where the maximum salary is between 10000 and 2000 inclusive

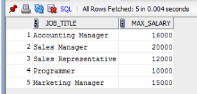
This question refers to the jobs table where the maximum salary for a job is given. Because we only want some of the rows we obviously need to use a WHERE clause. We have at least two ways to construct the clause though. One is to say that we want rows where the maximum salary is greater than or equal to 10000 and the maximum salary is also less than or equal to 20000. A simpler way is to use the BETWEEN keyword as follows:

SELECT job\_title, max\_salary

FROM jobs

WHERE max\_salary BETWEEN 10000 AND 20000;

The results are:



Check for yourself that the other method gives the same results.

### Q9. List the five highest paid employees

**NEW MATERIAL (ADVANCED)**

This question is related to the first two in that it asks us to list employees in order of their salaries. The difference is that now we must limit the results to only the top five. Different database systems have slightly different ways of specifying this, but here we present the syntax that appears in the SQL standard and is also used by Oracle (as of 12c at least).

The syntax is FETCH FIRST x ROWS ONLY (where x is the number of rows you want) and this clause comes at the end of the query. So in our case the SQL we need is:

SELECT first\_name, last\_name, salary

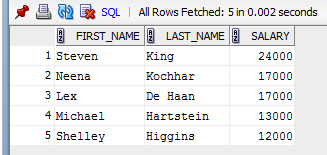
FROM employees

WHERE salary > 10000

ORDER BY salary DESC

FETCH FIRST 5 ROWS ONLY;

The results should be:



When ordering and limiting the number of rows, we can run into a problem with ties. For example, suppose we only wanted the top two employees we would have a tie for second place between Neena Kochhar and Lex De Haan, both of whom earn 17,000. If we only want two employees, which do we choose?

The “ONLY” keyword in “FETCH FIRST 2 ROWS ONLY” tells the database system that we only want two rows, even if there’s a tie. It is then up to the database system to break the tie however it wants to. We can, however, use “WITH TIES” instead of “ONLY” and then even if we specify only 2 rows, we may get more if there are ties. Try the following query and see how many rows are returned.

SELECT first\_name, last\_name, salary

FROM employees

WHERE salary > 10000

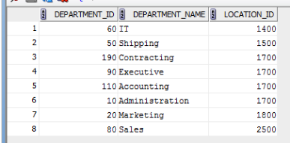
ORDER BY salary DESC

FETCH FIRST 2 ROWS WITH TIES;

# Completion Problems

### Q1. List the departments ordered by their location ID in ascending order.

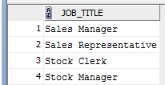
A: SELECT department\_id, department\_name, location\_id   
FROM departments

ORDER BY location\_ID ASC;  


### Q2. List all the job titles starting with the letter S.

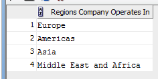
A: SELECT job\_title   
FROM jobs   
ORDER BY job\_title ASC

WHERE job\_titles LIKE ‘S%’;



### Q3. List the regions the company operates in, giving the column name “Regions Company Operates In”.

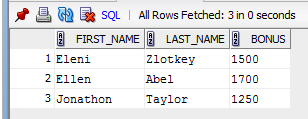
A: SELECT regions\_id AS “Regions Company Operates In”  
FROM regions;



### Q4. List all the employees who earn a bonus

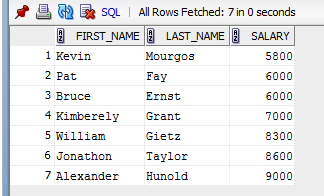
A: SELECT first\_name, last\_name, bonus   
FROM employees

WHERE bonus IS NOT NULL;



### Q5. List all employees earning between 5000 and 9000 inclusive, ordered by their salary in ascending order.

A: SELECT first\_name, last\_name, salary   
FROM employees   
WHERE salary BETWEEN 5000 AND 9000  
ORDER BY salary ASC;



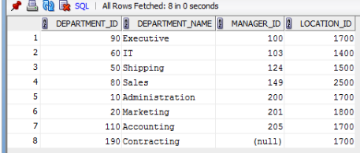
### Q6. List all the employees who are managed by someone with manager ID of 100, 149 or 124.

A: SELECT first\_name, last\_name, manager\_id   
FROM employees

WHERE manager\_id IN (‘100’, ‘149’,’124’);  


### Q7. List all the department details ordered by the manager ID.

A: SELECT \*  
FROM departments

ORDER BY manager\_id;  


### Q8. List the employees whose last names begin with K. Put everything in lower case.

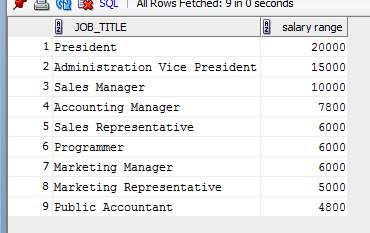
A: SELECT LOWER(last\_name), LOWER(first\_name)  
FROM employees   
WHERE last\_name LIKE 'K%';

### 

### Q9. List all the jobs where the range of possible salaries is greater than 4000. Order them by the range of salaries from largest to smallest.

A: SELECT job\_title, max\_salary - min\_salary AS "salary range"   
FROM jobs

WHERE salary > 4000

ORDER BY max\_salary – min\_salary DESC  


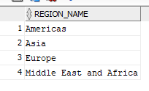
### 

# Deliberate Practice: Write the SQL

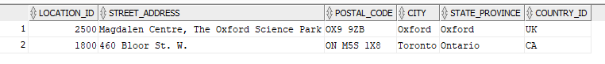
### Q1. List the employees whose phone numbers start with “515”



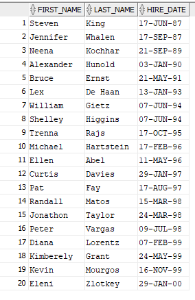
### Q2. List the regions in in alphabetical order



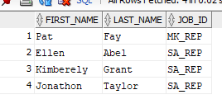
### Q3. List the locations of the company outside the US, ordered alphabetically by city name



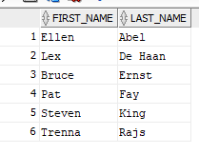
### Q4. List the employees by how long they have worked at the company



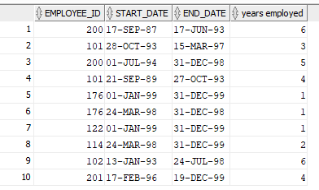
### Q5. List the reps in order of their surname, but keeping those in the same job together.



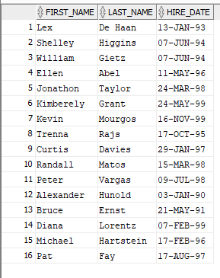
### Q6. List the employees whose full name (first name and surname including the space between them) contains fewer than 12 characters.



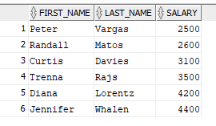
### Q7. List the employee IDs of workers who are no longer with the company and how many years they worked at the company.



### Q8. List the names of the employees hired in the 1990s.



### Q9. List the names of the six lowest-paid employees.



### Q10. List the names of the employees receiving a bonus as well as what percentage of their salary their bonus is. Give the percentage to two decimal places and include the percentage sign. Order the results by the percentage from lowest to highest.

