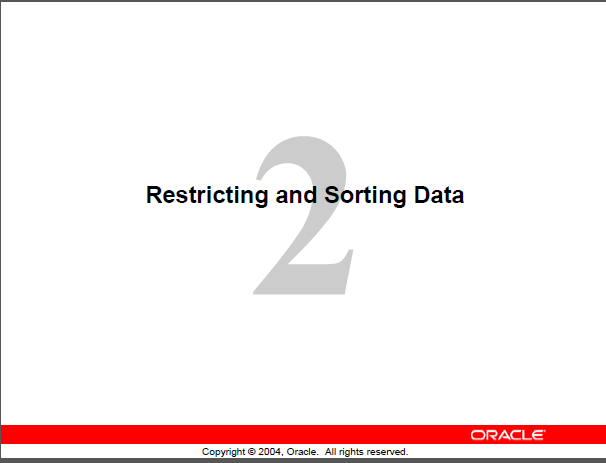
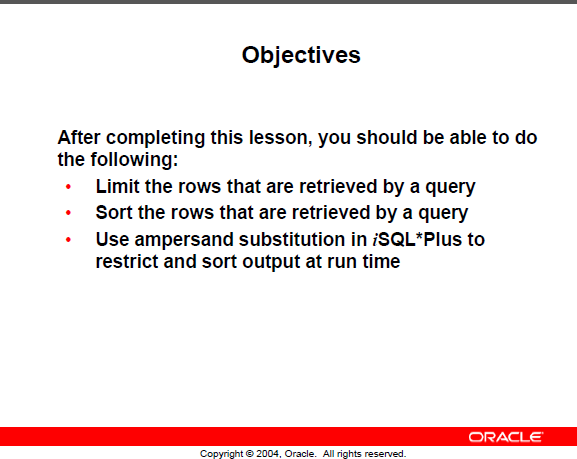
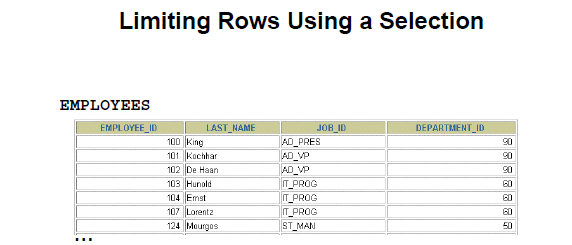
Les01B-WHERE Restricting Sorting





Limiting rows --- WHERE

Here are 7 of the 20 employees in the EMPLOYEE table



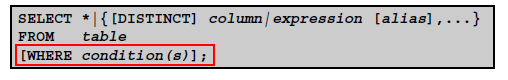
**PROBLEM: 🡺 Show those in department 90 only**

(done next page)

Result **should be**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **LAST\_NAME** | **FIRST\_NAME** | **DEPARTMENT\_ID** |
| 100 | King | Steven | 90 |
| 101 | Kochhar | Neena | 90 |
| 102 | De Haan | Lex | 90 |

General Format of WHERE



WHERE 🡺 follows the FROM

Solving the problem.

**PROBLEM: 🡺 Show those in department 90 only**

**SELECT employee\_id, last\_name, job\_id, department\_id**

**FROM employees**

**WHERE department\_id = 90;**

***Another style -- when the list is long and harder to read***

**SELECT employee\_id,**

**last\_name,**

**job\_id,**

**department\_id**

**FROM employees**

**WHERE department\_id = 90;**

Purpose: Readability

WHERE with Character strings and Dates



🡪 double quotes on alias names only





The default date format varies based on installation and Company directive.

Some prefer 01-31-2023 some refer 2023-01-31 you need to find out

SELECT sysdate FROM dual;

**SELECT last\_name, job\_id, department\_id**

**FROM employees**

**WHERE last\_name = ‘Whalen’; -- single quotes – exact case**

|  |  |  |
| --- | --- | --- |
| **LAST\_NAME** | **JOB\_ID** | **DEPARTMENT\_ID** |
| Whalen | AD\_ASST | 10 |

WHERE clause

**Compares** – values in a column,

Literal

Arithmetic expressions

Functions (more later)

Date format can be

1 customized – later chapter

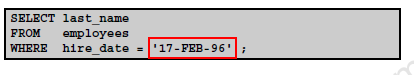
2 Default changed

**NEEDS** 1 Column name

2 Comparison condition

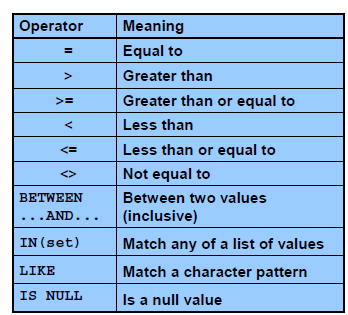
3 Column Name, constant or list of values

**Date Sample**



'02/17/1996' ----- might be the default

Comparison Operators and Conditions



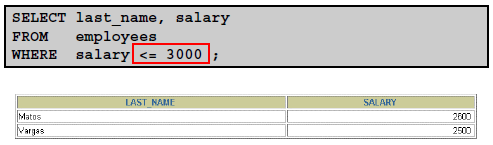
IS NULL – you cannot be equal to an unknown value

IN—very convenient. Often used in subqueries … more later

NOTE: != and ^= can also represent not equal

Samples:

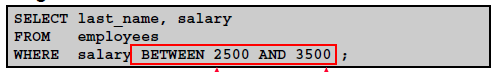
**Salary less than 3000 or a salary equal to 3000**

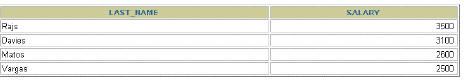


**BETWEEN**

**RANGE of SALARIES --- BETWEEN**

The values are INCLUSIVE of the start and end value limits





**RANGE of Character values such as name**

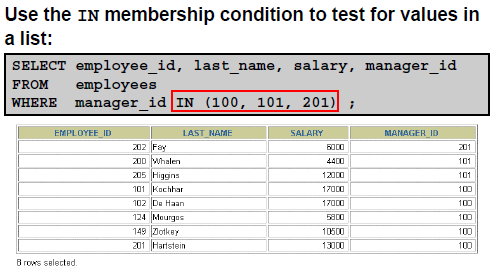
**SELECT last\_name||job\_id as "Employees"**

**FROM employees**

**WHERE last\_name BETWEEN 'King' AND 'Smith';**

**Can use NOT (BETWEEN ….. )**

**IN**



Show rows that match on any manager’s ID that are in the list shown.

Later with subqueries this becomes more useful.

**NOTE:**

Can use for 1 value, but probably should not.

It was meant as a list of values or a shortcut to wring a series of OR conditions

Also, character data

SELECT last\_name, hire\_date, job\_id, department\_id

FROM employees

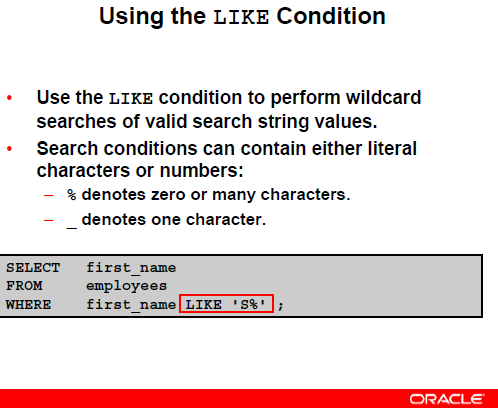
WHERE last\_name IN ('King', 'Whalen');

Later, IN may be generated by another SELECT query and not just a hard-coded list as shown above

**PROBLEM: You do not know the exact spelling of a name**

**OR**

**You are seeking a search condition**



WILD CARD values

**PROBLEM 6 for you to try**

**Sample with a year … copy and run it**

**SELECT last\_name, hire\_date**

**FROM employees**

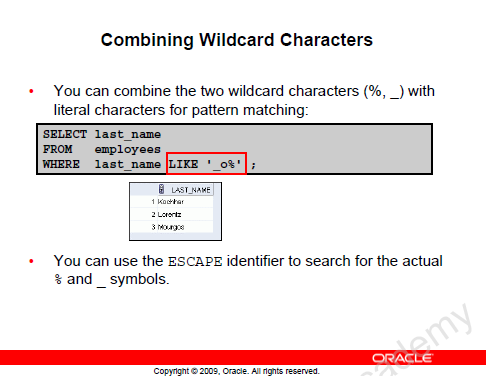
**WHERE hire\_date LIKE '%95';**

**What is the result ?**

**NOTE: Use like with wild cards**

**and not as an equal sign to an exact value**

* **not LIKE ‘King’**
* **but in conditions like this LIKE ‘%Kin%’**



**Looking for job IDs that have an underscore in the name.**

**Helps find an EXACT match**

**SELECT last\_name, job\_id**

**FROM employees**

**WHERE job\_id LIKE '%SA\\_%'**

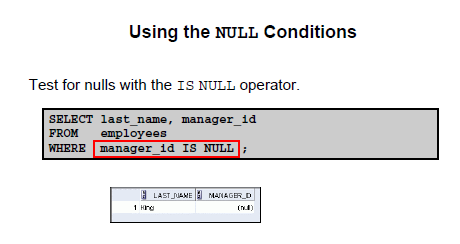
**ESCAPE '\'**

**🡺 Defined the \ as an escape identifier**

**Means looking for 1 anything or nothing in front**

**2 SA followed by \_ the underscore**

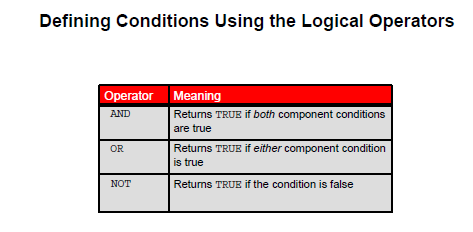
**3 Followed by anything or nothing**



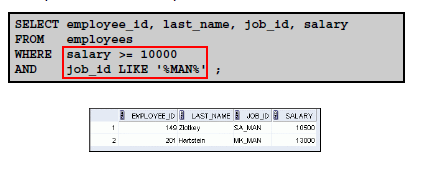
**You cannot be equal to a non-existent value**

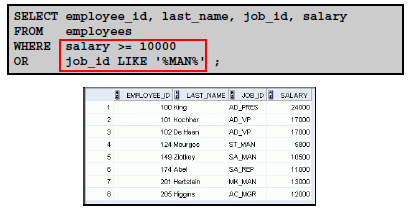
**MUST use the word ----->>>> IS**

**Try a SELECT for those who do not have a commission\_pct value**

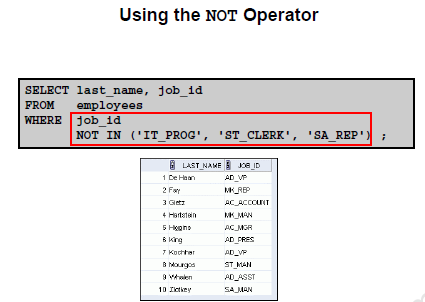


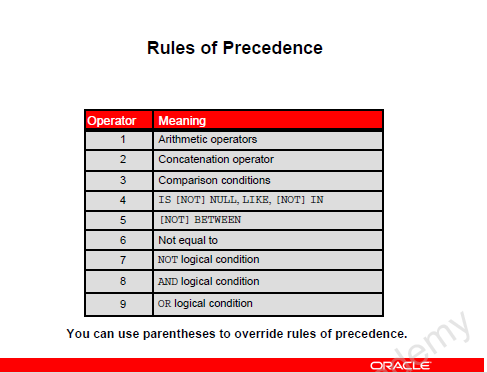
**EXAMPLES:**





**More complicated example:**





**PROBLEM 7**

**Show all employees with a job of SA\_REP or those with job of AD\_PRES and also have a salary > 15000**

**EXAMPLE:**

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

**FROM employees**

**WHERE job\_id = 'SA\_REP'**

**OR JOB\_ID = 'AD\_PRES'**

**AND salary > 15000;**

**LAST\_NAME JOB\_ID SALARY**

**------------------------- ---------- ----------**

**King AD\_PRES 24000**

**Abel SA\_REP 11000**

**O'Brian SA\_REP 8600**

**Grants SA\_REP 7000**

**de Man SA\_REP 7000**

***ALWAYS check the output.***

***Is that what the USER wanted?***

***OK or fix it.***

By adding the bracketing, it changes the resulting output.

Should use bracketing often

**SORTING**

**PROBLEM: List all employees in order of their department ID**

**Humans prefer results in some type of order**

**EX: VISA BILL – in date order of transaction**

**Class list in order by last name**

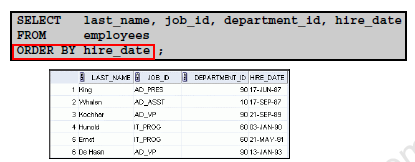
**SELECT \***

**FROM employees**

**ORDER BY department\_id;**

**ORDER BY 🡸 is the LAST statement, always**

**Another:**



**Sorting Descending order**

**SELECT last\_name, job\_id**

**FROM employees**

**ORDER BY last\_name DESC**

**LAST\_NAME JOB\_ID**

**------------------------- ----------**

**de Man SA\_REP**

**Zlotkey SA\_MAN**

**Young SA\_REP**

**Whiteduck SA\_REP**

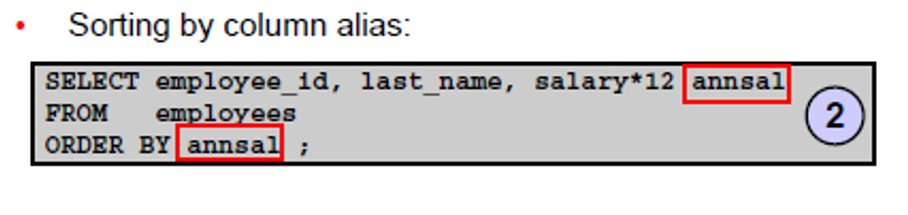
**Whalen AD\_ASST**

**Wandiko SA\_REP**

**DEFAULT is ASC**

**You can use it, but most don’t bother**

**MORE SORTING**



**NOTE: You can order by with an alias**

**You can sort on a column that is not being displayed**

**SELECT last\_name 🡸 employee name is displayed**

**FROM employees**

**ORDER BY department\_id; 🡸 but order of names depends on department**

**LAST\_NAME**

**-------------------------**

**Flertjan**

**Whalen**

**Fay**

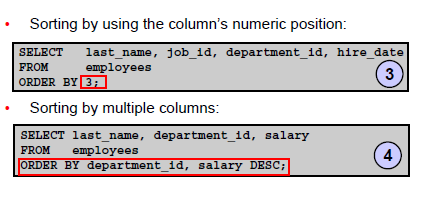
**Hartstein**

**Davies**

**Vargas**

**Usually better if also include the department id. At least when testing if it works use the department id before you put this code into production**

**ANOTHER SORTING EXAMPLE**



3 Specifies the numeric position in the SELECT🡪 - in this case the 3rd column or department\_id

4 Shows sorting on two different fields in different directions

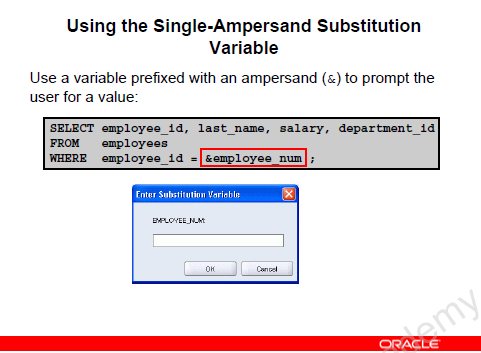
By default, the department\_id is sorted lowest value to highest (ASC)

The salary is sorted WITHIN each department in DESC order (highest to lowest)

**Substitution Variables**

**Up until now if you want to query department 90 you specified it in the code**

**What if you want to do several departments but not rewrite the code each time?**



**Allows the user to specify a value at run time and that value is not stored**

**SELECT last\_name AS LAST, department\_id AS DEPT**

**FROM employees**

**where department\_id = &deptno;**

**SELECT last\_name AS LAST, department\_id AS DEPT**

**FROM employees**

**where department\_id in (&deptno1, &deptno2);**

**Note how to use it in a character string**

**It is inside the character string quotes.**

**SELECT last\_name AS LAST, department\_id AS DEPT**

**FROM employees**

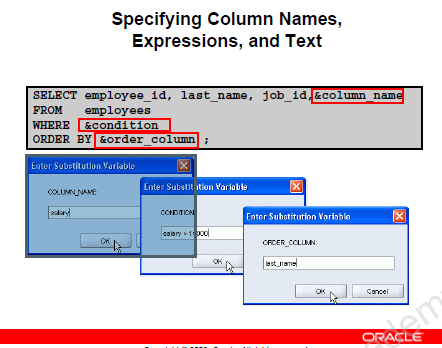
**WHERE job\_id = ‘&job\_title’;**

PROBLEM 7

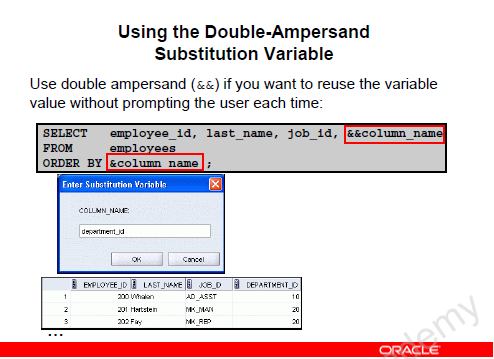
TRY AGAIN … but use ‘&job title’

**How to specify a column name to retrieve**

**More flexibility**



**Maintaining the value -- not tested for on this course**

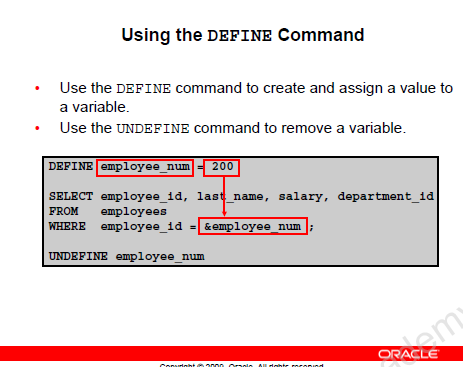


**Prompted first time**

**Value is stored**

**To remove the value from future use**

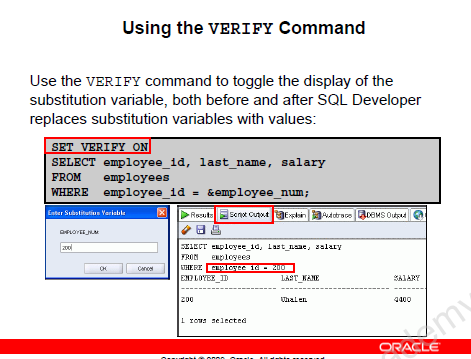
**UNDEFINE column\_name**



Release or unstore the value

User NOT prompted as the value exists already

Store 200



**Sequence of events**

**1 SQL code has no value specified**

**2 It prompts you for the value**

**3 It displays the substitution**

**(show the verify effect)**

**NOW COMPLETE LABS 1 and 2**