Oracle 11g - SQL

Restricting and Sorting Data



Objectives

After completing this lesson, you should be able to do the following:

- WHERE clause to limit the output retrieved
- Use character string literals with WHERE clause
- ☐ Use of Comparison and Logical Operators in WHERE clause
- Rules of precedence for Comparison and Logical operators
- ☐ Sort data using ORDER BY clause in SELECT statement
- Sort output in descending and ascending order



Limiting Rows Using a Selection

EMPLOYEES

	A	EMPLOYEE_ID	LAST_NAME	∄ JOB_ID	DEPARTMENT_ID
1		200	Whalen	AD_ASST	10
2		201	Hartstein	MK_MAN	20
3		202	Fay	MK_REP	20
4		205	Higgins	AC_MGR	110
5		206	Gietz	AC_ACCOUNT	110
6		100	King	AD_PRES	90
7		101	Kochhar	AD_VP	90

. . .

20	178 Grant	SA_REP	(null)
----	-----------	--------	--------

"retrieve all employees in department 90"

	A	EMPLOYEE_ID	A	LAST_	NAME	A	JOB_ID	A	DEPARTMENT_ID
1		100	Kin	g		AD.	_PRES		90
2		101	Ko	chhar		AD.	_VP		90
3		102	De	Haan		AD.	_VP		90



Limiting the Rows That Are Selected

Restrict the rows that are returned by using the WHERE clause:

```
SELECT *|{[DISTINCT] column|expression [alias],...}
FROM table
[WHERE condition(s)];
```

The WHERE clause follows the FROM clause.



Using the WHERE Clause

```
SELECT employee_id, last_name, job_id, department_id
FROM employees
WHERE department_id = 90;
```

	A	EMPLOYEE_ID	LAST_NAME	g Job_ID g	DEPARTMENT_ID
1		100	King	AD_PRES	90
2		101	Kochhar	AD_VP	90
3		102	De Haan	AD_VP	90



Character Strings and Dates

- Character strings and date values are enclosed in single quotation marks.
- Character values are case sensitive, and date values are format sensitive.
- The default date format is DD-MON-RR.

```
SELECT last_name, job_id, department_id
FROM employees
WHERE last_name = 'Whalen';
```





Comparison Conditions

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
BETWEENAND	Between two values (inclusive)
IN(set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value



Using Comparison Conditions

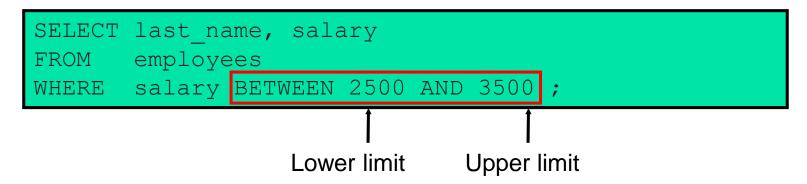
```
SELECT last_name, salary
FROM employees
WHERE salary <= 3000;
```

	LAST_NAME	A	SALARY
1	Matos		2600
2	Vargas		2500



Using the Between Condition

Use the BETWEEN condition to display rows based on a range of values:



	LAST_NAME	A	SALARY
1	Rajs		3500
2	Davies		3100
3	Matos		2600
4	Vargas		2500



Using the IN Condition

Use the IN membership condition to test for values in a list:

```
SELECT employee_id, last_name, salary, manager_id FROM employees
WHERE manager_id IN (100, 101, 201);
```

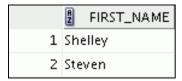
	A	EMPLOYEE_ID	B L	_AST_NAME	A	SALARY	A	MANAGER_ID
1		201	Hart	stein		13000		100
2		101	Kock	nhar		17000		100
3		102	De H	aan		17000		100
4		124	Mou	rgos		5800		100
5		149	Zlotk	key		10500		100
6		200	Whal	en		4400		101
7		205	Higg	ins		12000		101
8		202	Fay			6000		201



Using the LIKE Condition

- Use the LIKE condition to perform wildcard searches of valid search string values.
- Search conditions can contain either literal characters or numbers:
 - o % denotes zero or many characters.
 - o denotes one character.

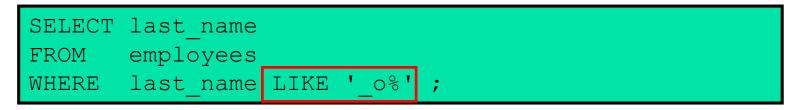
```
SELECT first_name
FROM employees
WHERE first_name LIKE 'S%';
```





Using the LIKE Condition

You can combine pattern-matching characters:





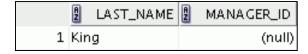
■ You can use the ESCAPE identifier to search for the actual % and symbols.



Using the NULL Conditions

Test for nulls with the IS NULL operator.

```
SELECT last_name, manager_id
FROM employees
WHERE manager_id IS NULL;
```





Logical Conditions

Operator	Meaning
AND	Returns TRUE if both component conditions are true
OR	Returns TRUE if either component condition is true
TON	Returns TRUE if the following condition is false



Using the AND Operator

AND requires both conditions to be true:

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary >=10000
AND job_id LIKE '%MAN%';
```

	A	EMPLOYEE_ID	A	LAST_NAME	A	JOB_ID	A	SALARY
1		201	Hai	rtstein	MK	_MAN		13000
2		149	ZIo	tkey	SA.	_MAN		10500



Using the OR Operator

OR requires either condition to be true:

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary >= 10000
OR job_id LIKE '%MAN%';
```

	A	EMPLOYEE_ID	LAST_NAME	∄ JOB_ID	2 SALARY
1		201	Hartstein	MK_MAN	13000
2		205	Higgins	AC_MGR	12000
3		100	King	AD_PRES	24000
4		101	Kochhar	AD_VP	17000
5		102	De Haan	AD_VP	17000
6		124	Mourgos	ST_MAN	5800
7		149	Zlotkey	SA_MAN	10500
8		174	Abel	SA_REP	11000



Using the NOT Operator

```
SELECT last_name, job_id
FROM employees
WHERE job_id
NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP');
```

	0	
	LAST_NAME	₿ JOB_ID
1	De Haan	AD_VP
2	Fay	MK_REP
3	Gietz	AC_ACCOUNT
4	Hartstein	MK_MAN
5	Higgins	AC_MGR
6	King	AD_PRES
7	Kochhar	AD_VP
8	Mourgos	ST_MAN
9	Whalen	AD_ASST
10	Zlotkey	SA_MAN



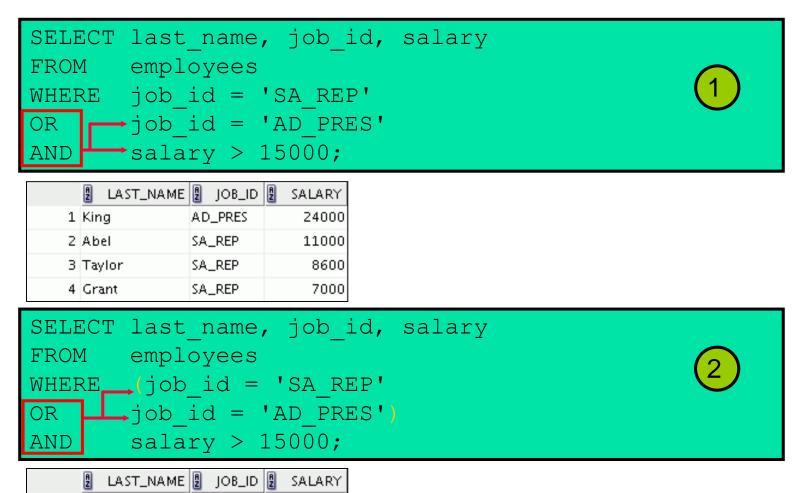
Rules of Precedence

Operator	Meaning			
1	Arithmetic operators			
2	Concatenation operator			
3	Comparison conditions			
4	IS [NOT] NULL, LIKE, [NOT] IN			
5	[NOT] BETWEEN			
6	Not equal to			
7	NOT logical condition			
8	AND logical condition			
9	OR logical condition			

You can use parentheses to override rules of precedence.



Rules of Precedence



1 King

24000

AD_PRES

Using the ORDER BY Clause

- Sort retrieved rows with the ORDER BY clause:
 - o ASC: ascending order, default
 - o DESC: descending order
- The ORDER BY clause comes last in the SELECT statement:

```
SELECT last_name, job_id, department_id, hire_date FROM employees
ORDER BY hire_date;
```

	LAST_NAME		DEPARTMENT_ID	HIRE_DATE
1	King	AD_PRES	90	17-JUN-87
2	Whalen	AD_ASST	10	17-SEP-87
3	Kochhar	AD_VP	90	21-SEP-89
4	Hunold	IT_PROG	60	03-JAN-90
20	Zlotkey	SA_MAN	80	29-JAN-00



Sorting

Sorting in descending order:

```
SELECT last_name, job_id, department_id, hire_date FROM employees
ORDER BY hire_date DESC;
```

Sorting by column alias:

```
SELECT employee_id, last_name, salary*12 annsal FROM employees
ORDER BY annsal;
```

Sorting by multiple columns:

```
SELECT last_name, department_id, salary
FROM employees

ORDER BY department_id, salary DESC;
```

