Restricting and Sorting Data

- **Restrict the rows returned by using the `WHERE` clause.**

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

- **The `WHERE` clause follows the `FROM` clause.**

Limiting the Rows Selected

You can restrict the rows returned from the query by using the `WHERE` clause. A `WHERE` clause contains a condition that must be met, and it directly follows the `FROM` clause. If the condition is true, the row meeting the condition is returned.

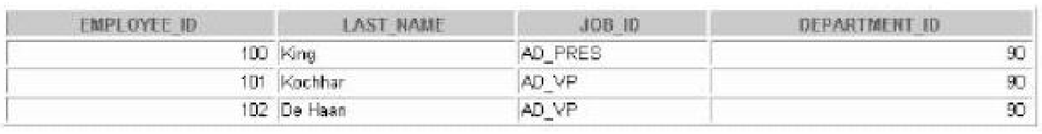
In the syntax:

`WHERE`	restricts the query to rows that meet a condition
`condition`	is composed of column names, expressions, constants, and a comparison operator

The `WHERE` clause can compare values in columns, literal values, arithmetic expressions, or functions. It consists of three elements:

- Column name
- Comparison condition
- Column name, constant, or list of values

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



EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90

In the example, the `SELECT` statement retrieves the name, job ID, and department number of all employees whose job ID is `SA_REP`.

Note that the job title `SA_REP` has been specified in uppercase to ensure that it matches the job ID column in the `EMPLOYEES` table. Character strings are case sensitive.

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 = 'Goyal';
```

Character strings and dates in the WHERE clause must be enclosed in single quotation marks (''). Number constants, however, should not be enclosed in single quotation marks.
Oracle databases store dates in an internal numeric format, representing the century, year, month, day, hours, minutes, and seconds. The default date display is DD-MON-RR.

Note: Changing the default date format is covered in a subsequent lesson.

Comparison Conditions

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

Example

```
    ... WHERE hire_date='01-JAN-95'
    ... WHERE salary>=6000
    ... WHERE last_name='Smith'
```

An alias cannot be used in the WHERE clause.

Note: The symbol != and ^= can also represent the *not equal to* condition.

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

Other Comparison Conditions

Operator	Meaning
`BETWEEN ...AND...`	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 the `BETWEEN` Condition

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

```
SELECT last_name, salary
FROM   employees
WHERE  salary BETWEEN  2500 AND 3500;
```

Lower limit   Upper limit

LAST_NAME	SALARY
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500

You can display rows based on a range of values using the `BETWEEN` range condition. The range that you specify contains a lower limit and an upper limit.
Values specified with the `BETWEEN` condition are inclusive. You must specify the lower limit first.

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);
```

EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
202	Fay	6000	201
200	Whalen	4400	101
205	Higgins	12000	101
101	Kochhar	17000	100
102	De Haan	17000	100
124	Mourgos	5800	100
149	Zlotkey	10500	100
201	Hartstein	13000	100

8 rows selected.

To test for values in a specified set of values, use the IN condition. The IN condition is also known as the membership condition.

The example displays employee numbers, last names, salaries, and manager's employee numbers for all the employees whose manager's employee number is 100, 101, or 201.

The IN condition can be used with any data type. The following example returns a row from the EMPLOYEES table for any employee whose last name is included in the list of names in the WHERE clause:

```
SELECT employee_id, manager_id, department_id
FROM   employees
WHERE  last_name IN ('Hartstein', 'Vargas');
```

If characters or dates are used in the list, they must be enclosed in single quotation marks (' ').

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:**
  - **% denotes zero or many characters.**
  - **_ denotes one character.**

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

You may not always know the exact value to search for. You can select rows that match a character pattern by using the LIKE condition. The character pattern-matching operation is referred to as a wildcard search. Two symbols can be used to construct the search string.

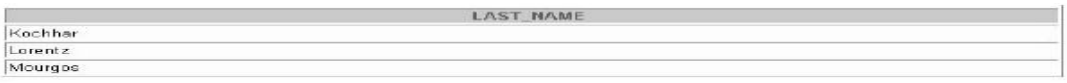
Symbol	Description
%	Represents any sequence of zero or more characters
_	Represents any single character

The SELECT statement on the slide returns the employee first name from the EMPLOYEES table for any employee whose first name begins with an *S*. Note the uppercase *S*. Names beginning with an *s* are not returned.
The LIKE condition can be used as a shortcut for some BETWEEN comparisons. The following example displays the last names and hire dates of all employees who joined between January 1995 and December 1995:

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date LIKE '%95';
```

- **You can combine pattern-matching characters.**

```
SELECT  last_name
FROM    employees
WHERE   last_name LIKE '_o%';
```



```
LAST_NAME
Kochhar
Lorentz
Mourgos
```

- **You can use the ESCAPE identifier to search for the actual % and _ symbols.**

Combining Wildcard Characters

The % and _ symbols can be used in any combination with literal characters. The example on the slide displays the names of all employees whose last name has an *o* as the second character.

The ESCAPE Option

When you need to have an exact match for the actual % and _ characters, use the ESCAPE option. This option specifies what the escape character is. If you want to search for strings that contain SA_, you can search for it using the following SQL statement:

```
SELECT employee_id, last_name, job_id
FROM   employees
WHERE  job_id LIKE '%SA\_%' ESCAPE '\';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID
149	Zlotkey	SA_MAN
174	Abel	SA_REP
176	Taylor	SA_REP
178	Grant	SA_REP

The ESCAPE option identifies the backslash (\) as the escape character. In the pattern, the escape character precedes the underscore (_). This causes the Oracle Server to interpret the underscore literally.

Using the NULL Conditions

Test for nulls with the IS NULL operator.

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

LAST_NAME	MANAGER_ID
King	

The NULL conditions include the IS NULL condition and the IS NOT NULL condition.

The IS NULL condition tests for nulls. A null value means the value is unavailable, unassigned, unknown, or inapplicable. Therefore, you cannot test with = because a null cannot be equal or unequal to any value. The slide example retrieves the last names and managers of all employees who do not have a manager.

For another example, to display last name, job ID, and commission for all employees who are *not* entitled to get a commission, use the following SQL statement:

```
SELECT last_name, job_id, commission_pct
FROM   employees
WHERE  commission_pct IS NULL;
```

LAST_NAME	JOB_ID	COMMISSION_PCT
King	AD_PRES	
Kochhar	AD_VP	
De Haan	AD_VP	
Gietz	AC_ACCOUNT	

Logical Conditions

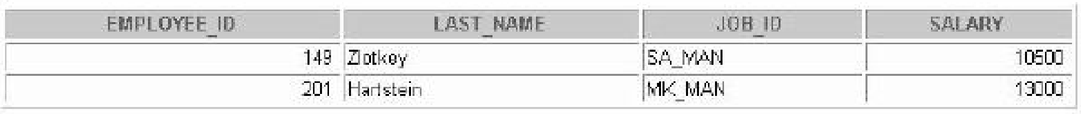
Operator	Meaning
AND	Returns TRUE if *both* component conditions are true
OR	Returns TRUE if *either* component condition is true
NOT	Returns TRUE if the following condition is false

All the examples so far have specified only one condition in the WHERE clause. You can use several conditions in one WHERE clause using the AND and OR operators.

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%';
```



EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
149	Zlotkey	SA_MAN	10500
201	Hartstein	MK_MAN	13000

In the example, both conditions must be true for any record to be selected. Therefore, only employees who have a job title that contains the string MAN *and* earn more than $10,000 are selected.

All character searches are case sensitive. No rows are returned if MAN is not in uppercase. Character strings must be enclosed in quotation marks.

AND Truth Table

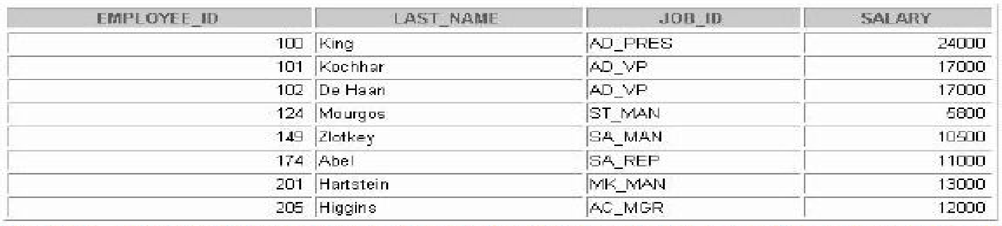
The following table shows the results of combining two expressions with AND:

AND	**TRUE**	**FALSE**	**NULL**
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

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%';
```



EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
100	King	AD_PRES	24000
101	Kochhar	AD_VP	17000
102	De Haan	AD_VP	17000
124	Mourgos	ST_MAN	5800
149	Zlotkey	SA_MAN	10500
174	Abel	SA_REP	11000
201	Hartstein	MK_MAN	13000
205	Higgins	AC_MGR	12000

In the example, either condition can be true for any record to be selected. Therefore, any employee who has a job ID containing MAN *or* earns more than $10,000 is selected.

The OR Truth Table

The following table shows the results of combining two expressions with OR:

OR	**TRUE**	**FALSE**	**NULL**
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

Using the NOT Operator

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

LAST_NAME	JOB_ID
King	AD_PRES
Kochhar	AD_VP
De Haan	AD_VP
Mourgos	ST_MAN
Zlotkey	SA_MAN
Whalen	AD_ASST
Hartstein	MK_MAN
Fay	MK_REP
Higgins	AC_MGR
Gietz	AC_ACCOUNT

The slide example displays the last name and job ID of all employees whose job ID *is not* `IT_PROG`, `ST_CLERK`, or `SA_REP`.

The NOT Truth Table

The following table shows the result of applying the `NOT` operator to a condition:

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

Note: The `NOT` operator can also be used with other SQL operators, such as `BETWEEN`, `LIKE`, and `NULL`.

```
... WHERE  job_id    NOT  IN ('AC_ACCOUNT', 'AD_VP')
... WHERE  salary    NOT  BETWEEN  10000 AND  15000
... WHERE  last_name NOT  LIKE '%A%'
... WHERE  commission_pct  IS   NOT  NULL
```

Rules of Precedence

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

Override rules of precedence by using parentheses.

The rules of precedence determine the order in which expressions are evaluated and calculated. The table lists the default order of precedence. You can override the default order by using parentheses around the expressions you want to calculate first.

```
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
Taylor	SA_REP	8600
Grant	SA_REP	7000

Example of the Precedence of the AND Operator

In the slide example, there are two conditions:

- The first condition is that the job ID is `AD_PRES` *and* the salary is greater than $15,000.
- The second condition is that the job ID is `SA_REP`.

Therefore, the `SELECT` statement reads as follows:

"Select the row if an employee is a president *and* earns more than $15,000, *or* if the employee is a sales representative."

Use parentheses to force priority.

```
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

In the example, there are two conditions:

- The first condition is that the job ID is `AD_PRES` *or* `SA_REP`.
- The second condition is that salary is greater than $15,000.

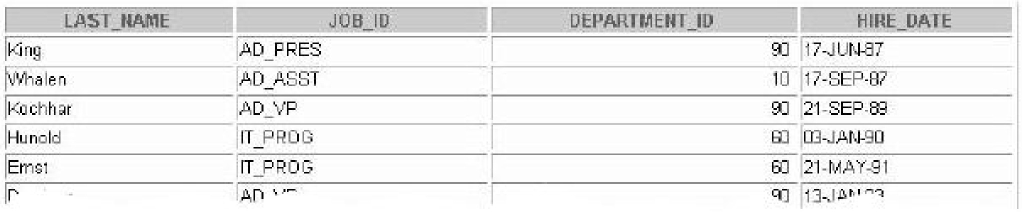
Therefore, the `SELECT` statement reads as follows:

"Select the row if an employee is a president *or* a sales representative, *and* if the employee earns more than $15,000."

ORDER BY Clause

- Sort rows with the `ORDER BY` clause
  - `ASC`: ascending order (the default order)
  - `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	JOB_ID	DEPARTMENT_ID	HIRE_DATE
King	AD_PRES	90	17-JUN-87
Whalen	AD_ASST	10	17-SEP-87
Kochhar	AD_VP	90	21-SEP-89
Hunold	IT_PROG	60	03-JAN-90
Ernst	IT_PROG	60	21-MAY-91
D	AD_V	90	13-JAN

The order of rows returned in a query result is undefined. The `ORDER BY` clause can be used to sort the rows. If you use the `ORDER BY` clause, it must be the last clause of the SQL statement. You can specify an expression, or an alias, or column position as the sort condition.

Syntax

```
SELECT          expr
FROM            table
[WHERE          condition(s)]
[ORDER BY {column, expr} [ASC|DESC]];
```

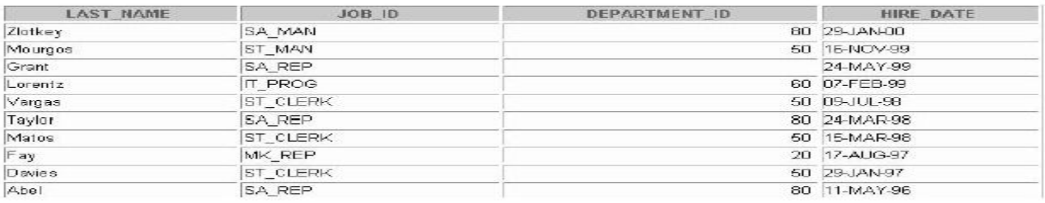
In the syntax:

`ORDER BY`	specifies the order in which the retrieved rows are displayed
`ASC`	orders the rows in ascending order (this is the default order)
`DESC`	orders the rows in descending order

If the `ORDER BY` clause is not used, the sort order is undefined, and the Oracle server may not fetch rows in the same order for the same query twice. Use the `ORDER BY` clause to display the rows in a specific order.

Sorting in Descending Order

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



LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
Zlotkey	SA_MAN	80	29-JAN-00
Mourgos	ST_MAN	50	16-NOV-99
Grant	SA_REP		24-MAY-99
Lorentz	IT_PROG	60	07-FEB-99
Vargas	ST_CLERK	50	09-JUL-98
Taylor	SA_REP	80	24-MAR-98
Matos	ST_CLERK	50	15-MAR-98
Fay	MK_REP	20	17-AUG-97
Davies	ST_CLERK	50	29-JAN-97
Abel	SA_REP	80	11-MAY-96

Default Ordering of Data

The default sort order is ascending:

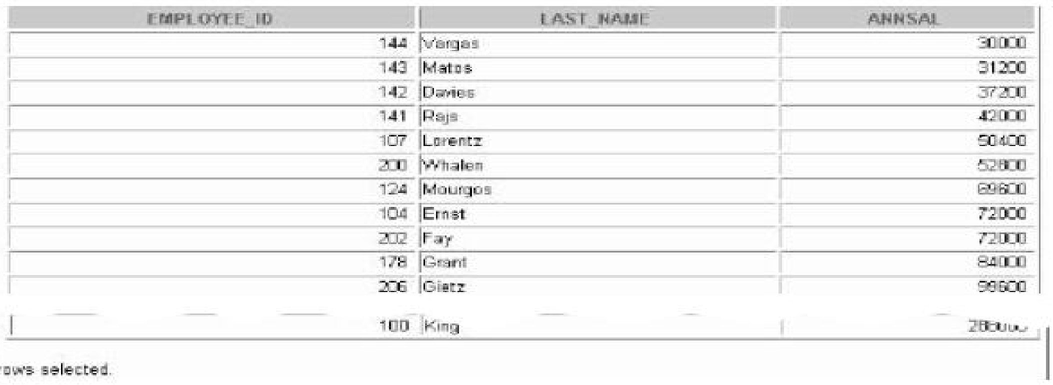
- Numeric values are displayed with the lowest values first: for example, 1–999.
- Date values are displayed with the earliest value first: for example, 01-JAN-92 before 01-JAN-95.
- Character values are displayed in alphabetical order: for example, *A* first and *Z* last.
- Null values are displayed last for ascending sequences and first for descending sequences.

Reversing the Default Order

To reverse the order in which rows are displayed, specify the `DESC` keyword after the column name in the `ORDER BY` clause. The slide example sorts the result by the most recently hired employee.

Sorting by Column Alias

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



EMPLOYEE_ID	LAST_NAME	ANNSAL
144	Vargas	30000
143	Matos	31200
142	Davies	37200
141	Rajs	42000
107	Lorentz	50400
200	Whalen	52800
124	Mourgos	69600
104	Ernst	72000
202	Fay	72000
178	Grant	84000
206	Gietz	99600
100	King	288000

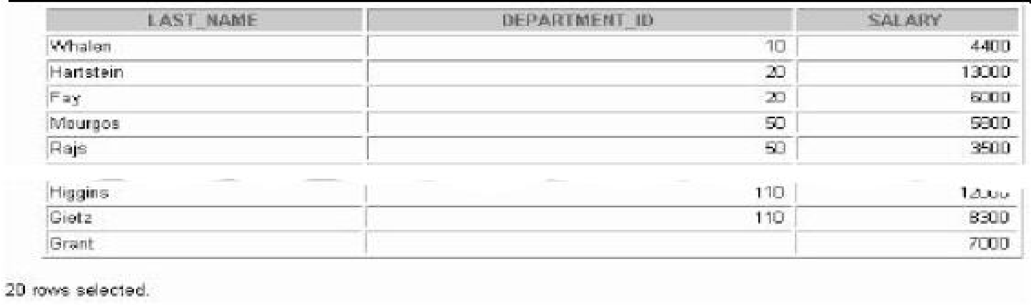
20 rows selected.

You can use a column alias in the `ORDER BY` clause. The slide example sorts the data by annual salary.

Sorting by Multiple Columns

- **The order of `ORDER BY` list is the order of sort.**

```
SELECT last_name, department_id, salary
FROM   employees
ORDER BY department_id, salary DESC;
```



LAST_NAME	DEPARTMENT_ID	SALARY
Whalen	10	4400
Hartstein	20	13000
Fay	20	6000
Mourgos	50	5800
Rajs	50	3500
Higgins	110	12000
Gietz	110	8300
Grant		7000

20 rows selected.

- **You can sort by a column that is not in the `SELECT` list.**

You can sort query results by more than one column. The sort limit is the number of columns in the given table.

In the ORDER BY clause, specify the columns, and separate the column names using commas. If you want to reverse the order of a column, specify DESC after its name. You can also order by columns that are not included in the SELECT clause.

Example

Display the last names and salaries of all employees. Order the result by department number, and then in descending order by salary.

```
SELECT   last_name, salary
FROM     employees
ORDER BY department_id, salary DESC;
```