
ORACLE
ACADEMY

Database Programming with SQL

3-2
Sorting Rows

ORACLE ACADEMY

Objectives

This lesson covers the following objectives:

- Construct a query to sort a result set in ascending or descending order
- State the order in which expressions are evaluated and calculated based on the rules of precedence
- Construct a query to order a result set using a column alias
- Construct a query to order a result set for single or multiple columns

Purpose

- By nature, most of us need order in our lives.
- Imagine if each time you had dinner, you had to look in every kitchen drawer or cabinet to find a knife and a fork?
- Ordering, grouping, and sorting makes finding things easier.
- Biologists group animals in phyla, astronomers order brightness of stars by magnitude, and Java programmers organize code in classes.

Purpose

- Our everyday lives are ordered in many situations:
  - Library books in library
  - Grocery-store shelves
  - Documents stored in file cabinets
- Being able to sort results is a convenient feature in SQL and enables programmers to display information in many different ways.
- For database design, business functions are ordered by entities and attributes; in database information, SQL uses the ORDER BY clause.

ORDER BY Clause

- Information sorted in ascending order is familiar to most of us.
- It's what makes looking up a number in a phone book, finding a word in the dictionary, or locating a house by its street address relatively easy.
- SQL uses the ORDER BY clause to order data.
- The ORDER BY clause can specify several ways in which to order rows returned in a query.

ORDER BY Clause

- The default sort order is ascending.
- Numeric values are displayed lowest to highest.
- Date values are displayed with the earliest value first.
- Character values are displayed in alphabetical order.
- Null values are displayed last in ascending order and first in descending order.
- NULLS FIRST Specifies that NULL values should be returned before non-NULL values.
- NULLS LAST Specifies that NULL values should be returned after non-NULL values.

ORDER BY Clause

- The following employees example uses the ORDER BY clause to order hire_date in ascending (default) order.
- Note: The ORDER BY clause must be the last clause of the SQL statement.

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

LAST_NAME	HIRE_DATE
King	17-Jun-1987
Whalen	17-Sep-1987
Kochhar	21-Sep-1989
Hunold	03-Jan-1990
Ernst	21-May-1991
De Haan	13-Jan-1993
Gietz	07-Jun-1994
Higgins	07-Jun-1994
Rajs	17-Oct-1995
Hartstein	17-Feb-1996

Sorting in Descending Order

- You can reverse the default order in the ORDER BY clause to descending order by specifying the DESC keyword after the column name in the ORDER BY clause.

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

LAST_NAME	HIRE_DATE
Zlotkey	29-Jan-2000
Mourgos	16-Nov-1999
Grant	24-May-1999
Lorentz	07-Feb-1999
Vargas	09-Jul-1998
Taylor	24-Mar-1998
Matos	15-Mar-1998
Fay	17-Aug-1997
Davies	29-Jan-1997
Abel	11-May-1996

Using Column Aliases

- You can order data by using a column alias.
- The alias used in the SELECT statement is referenced in the ORDER BY clause.

```
SELECT last_name, hire_date AS "Date
  Started"
FROM employees
ORDER BY "Date Started";
```

LAST_NAME	Date Started
King	17-Jun-1987
Whalen	17-Sep-1987
Kochhar	21-Sep-1989
Hunold	03-Jan-1990
Ernst	21-May-1991
De Haan	13-Jan-1993
Gietz	07-Jun-1994
Higgins	07-Jun-1994
Rajs	17-Oct-1995
Hartstein	17-Feb-1996

Sorting with Other Columns

- It is also possible to use the ORDER BY clause to order output by a column that is not listed in the SELECT clause.
- In the following example, the data is sorted by the last_name column even though this column is not listed in the SELECT statement.

```
SELECT employee_id, first_name
FROM employees
WHERE employee_id < 105
ORDER BY last_name;
```

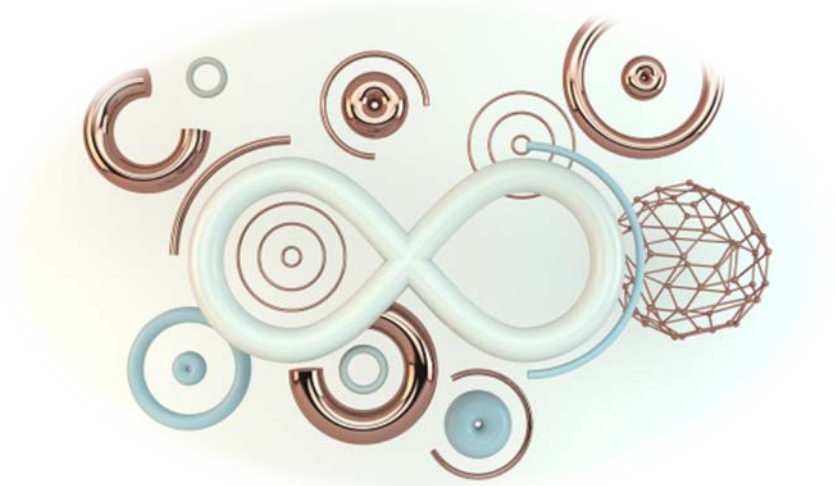
EMPLOYEE_ID	FIRST_NAME
102	Lex
104	Bruce
103	Alexander
100	Steven
101	Neena

Order of Execution

- The order of execution of a SELECT statement is as follows:
  - FROM clause: locates the table that contains the data
  - WHERE clause: restricts the rows to be returned
  - SELECT clause: selects from the reduced data set the columns requested
  - ORDER BY clause: orders the result set

Sorting with Multiple Columns

- It is also possible to sort query results by more than one column.
- In fact, there is no limit on how many columns you can add to the ORDER BY clause.



Sorting with Multiple Columns

- An example of sorting with multiple columns is shown below.
- Employees are first ordered by department number (from lowest to highest), then for each department, the last names are displayed in alphabetical order (A to Z).

```
SELECT department_id, last_name
FROM employees
WHERE department_id <= 50
ORDER BY department_id, last_name;
```

DEPARTMENT_ID	LAST_NAME
10	Whalen
20	Fay
20	Hartstein
50	Davies
50	Matos
50	Mourgos
50	Rajs
50	Vargas

Sorting with Multiple Columns

- To create an ORDER BY clause to sort by multiple columns, specify the columns to be returned and separate the column names using commas.
- If you want to reverse the sort order of a column, add DESC after its name.

```
SELECT department_id, last_name
FROM employees
WHERE department_id <= 50
ORDER BY department_id DESC, last_name;
```

DEPARTMENT_ID	LAST_NAME
50	Davies
50	Matos
50	Mourgos
50	Rajs
50	Vargas
20	Fay
20	Hartstein
10	Whalen

Terminology

Key terms used in this lesson included:

- ORDER BY Clause
- ASCENDING
- DESCENDING
- Order of Execution

Summary

In this lesson, you should have learned how to:

- Construct a query to sort a result set in ascending or descending order
- Construct a query to order a result set using a column alias
- Construct a query to order a result set for single or multiple columns


ORACLE
ACADEMY