Database Foundations

6-8
Sorting Data Using ORDER BY
Roadmap

- Introduction to Oracle Application Express
- Structured Query Language (SQL)
- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Transaction Control Language (TCL)

- Retrieving Data Using SELECT
- Restricting Data Using WHERE
- Sorting Data Using ORDER BY
- Joining Tables Using JOINS

You are here
Objectives

This lesson covers the following objectives:

- **Use the `ORDER BY` clause to sort SQL query results**
- **Identify the correct placement of the `ORDER BY` clause within a `SELECT` statement**
- **Order data and limit row output by using the SQL `row_limiting_clause`**
- **Use substitution variables in the `ORDER BY` clause**
Using the ORDER BY Clause

• Sort the retrieved rows with the ORDER BY clause:
  – **ASC**: Ascending order (default)
  – **DESC**: Descending order

• The ORDER BY clause comes last in the SELECT statement:

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

• Sorting in descending order:

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

• Sorting by column alias:

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

• Sorting by using the column's numeric position:

```sql
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY 3;
```

• Sorting by multiple columns:

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

• The `row_limiting_clause` allows you to limit the rows that are returned by the query.

• Queries that order data and then limit row output are widely used and are often referred to as Top-N queries.

• You can specify the number of rows or percentage of rows to return with the `FETCH_FIRST` keywords.
SQL `row_limiting_clause`

- You can use the `OFFSET` keyword to specify that the returned rows begin with a row after the first row of the full result set.

- The `WITH TIES` keyword includes additional rows with the same ordering keys as the last row of the row-limited result set (you must specify `ORDER BY` in the query).

- You can specify the `ORDER BY` clause to ensure a deterministic sort order.
Using the SQL `row_limiting_clause` in a Query

You can specify the `row_limiting_clause` in the SQL `SELECT` statement by placing it after the `ORDER BY` clause.

```
subquery ::= 
  { query_block
    | subquery { UNION [ALL] | INTERSECT | MINUS } subquery 
  [ { UNION [ALL] | INTERSECT | MINUS } subquery ]...
    | ( subquery )
  }
[ order_by_clause ]
[OFFSET offset { ROW | ROWS }]
[FETCH { FIRST | NEXT } [{ row_count | percent PERCENT }]
  { ROW | ROWS }
  { ONLY | WITH TIES }]
```
SQL `row_lmiting_clause`: Example

```
SELECT employee_id, first_name
FROM employees
ORDER BY employee_id
FETCH FIRST 5 ROWS ONLY;
```

```
SELECT employee_id, first_name
FROM employees
ORDER BY employee_id
OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;
```
Substitution Variables

... salary = ? ...
... department_id = ? ...
... last_name = ? ...

I want to query different values.
Substitution Variables

- Use substitution variables to temporarily store values with single-ampersand (\&) and double-ampersand (\&\&) substitutions.

- Use substitution variables to supplement the following:
  - WHERE conditions
  - ORDER BY clauses
  - Column expressions
  - Table names
  - Entire SELECT statements
Using the Single-Ampersand Substitution Variable

Use a variable prefixed with an ampersand (&) to prompt the user for a value:

```
SELECT employee_id, last_name, salary, department_id
FROM employees
WHERE employee_id = &employee_num;
```
Using the Single-Ampersand Substitution Variable

![Enter Substitution Variable dialog box]

- **EMPLOYEE_NUM:**
  - 101

![Table with Employee Data]

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>SALARY</th>
<th>DEPARTMENT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Kochhar</td>
<td>17000</td>
<td>90</td>
</tr>
</tbody>
</table>
Character and Date Values with Substitution Variables

Use single quotation marks for date and character values:

```sql
SELECT last_name, department_id, salary*12
FROM employees
WHERE job_id = '&job_title';
```

[Table]

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>DEPARTMENT_ID</th>
<th>SALARY*12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hunold</td>
<td>60</td>
<td>108000</td>
</tr>
<tr>
<td>Ernst</td>
<td>60</td>
<td>72000</td>
</tr>
<tr>
<td>Austin</td>
<td>60</td>
<td>57600</td>
</tr>
<tr>
<td>Pataballa</td>
<td>60</td>
<td>57600</td>
</tr>
<tr>
<td>Lorentz</td>
<td>60</td>
<td>50400</td>
</tr>
</tbody>
</table>
Specifying Column Names, Expressions, and Text

```
SELECT employee_id, last_name, job_id, &column_name
FROM   employees
WHERE  &condition
ORDER BY &order_column;
```
Using the Double-Ampersand Substitution Variable

Use double ampersands (&&) if you want to reuse the variable value without prompting the user each time:

```
SELECT employee_id, last_name, job_id, &&column_name
FROM employees
ORDER BY &column_name;
```
Using the `DEFINE` Command

• Use the `DEFINE` command to create and assign a value to a variable.

• Use the `UNDEFINE` command to remove a variable.

```sql
DEFINE employee_num = 200

SELECT employee_id, last_name, salary, department_id
FROM employees
WHERE employee_id = &employee_num;

UNDEFINE employee_num
```
Using the `VERIFY` Command

Use the `VERIFY` command to toggle the display of the substitution variable before and after SQL Developer replaces substitution variables with values:

```sql
SET VERIFY ON

SELECT employee_id, last_name, salary
FROM   employees
WHERE  employee_id = &employee_num;
```

![Image of SQL query execution with substitution variable](image-url)
Summary

In this lesson, you should have learned how to:

• Use the `ORDER BY` clause to sort SQL query results

• Identify the correct placement of the `ORDER BY` clause within a `SELECT` statement

• Order data and limit row output by using the SQL `row_limiting_clause`

• Use substitution variables in the `ORDER BY` clause