Database Foundations

6-6
Retrieving Data Using SELECT
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 JOIN

You are here
Objectives

This lesson covers the following objectives:

• List the capabilities of SQL `SELECT` statements

• Write and execute a `SELECT` statement that:
  – Returns all rows and columns from a table
  – Returns specific columns from a table
  – Uses column aliases to display descriptive column headings
  – Uses arithmetic and concatenation operators
  – Uses literal character strings
  – Eliminates duplicate rows

• Describe the structure of a table
Basic SELECT Statement

- **SELECT** identifies the columns to be displayed.
- **FROM** identifies the table that contains those columns.

```
SELECT {*, [DISTINCT] column|expression [alias], ...} 
FROM table;
```
## Selecting All Columns

```
SELECT *
FROM   departments;
```

<table>
<thead>
<tr>
<th>DEPARTMENT_ID</th>
<th>DEPARTMENT_NAME</th>
<th>MANAGER_ID</th>
<th>LOCATION_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Administration</td>
<td>200</td>
<td>1700</td>
</tr>
<tr>
<td>20</td>
<td>Marketing</td>
<td>201</td>
<td>1800</td>
</tr>
<tr>
<td>30</td>
<td>Purchasing</td>
<td>114</td>
<td>1700</td>
</tr>
<tr>
<td>40</td>
<td>Human Resources</td>
<td>203</td>
<td>2400</td>
</tr>
<tr>
<td>50</td>
<td>Shipping</td>
<td>121</td>
<td>1500</td>
</tr>
<tr>
<td>60</td>
<td>IT</td>
<td>103</td>
<td>1400</td>
</tr>
<tr>
<td>70</td>
<td>Public Relations</td>
<td>204</td>
<td>2700</td>
</tr>
</tbody>
</table>
Selecting Specific Columns

```sql
SELECT department_id, location_id
FROM departments;
```

<table>
<thead>
<tr>
<th>DEPARTMENT_ID</th>
<th>LOCATION_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1700</td>
</tr>
<tr>
<td>20</td>
<td>1800</td>
</tr>
<tr>
<td>30</td>
<td>1700</td>
</tr>
<tr>
<td>40</td>
<td>2400</td>
</tr>
<tr>
<td>50</td>
<td>1500</td>
</tr>
<tr>
<td>60</td>
<td>1400</td>
</tr>
<tr>
<td>70</td>
<td>2700</td>
</tr>
</tbody>
</table>
Writing SQL Statements

- SQL statements are not case-sensitive.
- SQL statements can be entered on one or more lines.
- Keywords cannot be abbreviated or split across lines and are typically spelled with uppercase letters.
- Clauses are usually placed on separate lines.
- Indents are used to enhance readability.
- In SQL Developer, SQL statements can be terminated by a semicolon (;). Semicolons are required when you execute multiple SQL statements.
- In SQL*Plus, you are required to end each SQL statement with a semicolon (;).
Case Scenario: Retrieving Data

Sean, I would like to retrieve the data from the AUTHOR and BOOKS tables. Is that possible?

Sure. Let me retrieve the data and show it to you.
Case Scenario: Retrieving Data

Here is the information

SELECT AUTHOR_ID, AUTHOR_NAME
FROM AUTHOR;

AN0001 Oliver Goldsmith
AN0002 Oscar Wilde
AN0003 George Bernard Shaw
AN0004 Leo Tolstoy
AN0005 Percy Shelley
AN0006 Lord Byron
AN0007 John Keats
AN0008 Rudyard Kipling
AN0009 P. G. Wodehouse

9 rows selected

SELECT * FROM BOOKS;

BN0001 Florentine Tragedy
BN0002 A Vision
BN0003 Citizen of the World
BN0004 The Complete Poetical Works of Oliver Goldsmith
BN0005 Androcles and the Lion
BN0006 An Unsocial Socialist
BN0007 A Thing of Beauty is a Joy Forever
BN0008 Beyond the Pale
BN0009 The Clicking of Cuthbert
BN0010 Bride of Frankenstein
BN0011 Shelley Poetry and Prose
BN0012 War and Peace
Column Heading Defaults

• SQL Developer:
  – Default heading alignment: Left-aligned
  – Default heading display: Uppercase

• SQL*Plus:
  – Character and Date column headings: Left-aligned
  – Number column headings: Right-aligned
  – Default heading display: Uppercase
Arithmetic Expressions

Create expressions with number and date data by using arithmetic operators.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Add</td>
</tr>
<tr>
<td>-</td>
<td>Subtract</td>
</tr>
<tr>
<td>*</td>
<td>Multiply</td>
</tr>
<tr>
<td>/</td>
<td>Divide</td>
</tr>
</tbody>
</table>
Using Arithmetic Operators

```
SELECT last_name, salary, salary + 300
FROM employees;
```

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>SALARY</th>
<th>SALARY+300</th>
</tr>
</thead>
<tbody>
<tr>
<td>King</td>
<td>24000</td>
<td>24300</td>
</tr>
<tr>
<td>Kochhar</td>
<td>17000</td>
<td>17300</td>
</tr>
<tr>
<td>De Haan</td>
<td>17000</td>
<td>17300</td>
</tr>
<tr>
<td>Hunold</td>
<td>9000</td>
<td>9300</td>
</tr>
<tr>
<td>Ernst</td>
<td>6000</td>
<td>6300</td>
</tr>
<tr>
<td>Austin</td>
<td>4800</td>
<td>5100</td>
</tr>
<tr>
<td>Pataballa</td>
<td>4800</td>
<td>5100</td>
</tr>
</tbody>
</table>
Operator Precedence

1. SELECT last_name, salary, 12*salary+100
   FROM employees;

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>SALARY</th>
<th>12*SALARY+100</th>
</tr>
</thead>
<tbody>
<tr>
<td>King</td>
<td>24000</td>
<td>288100</td>
</tr>
<tr>
<td>Kochhar</td>
<td>17000</td>
<td>204100</td>
</tr>
<tr>
<td>De Haan</td>
<td>17000</td>
<td>204100</td>
</tr>
</tbody>
</table>

2. SELECT last_name, salary, 12*(salary+100)
   FROM employees;

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>SALARY</th>
<th>12*(SALARY+100)</th>
</tr>
</thead>
<tbody>
<tr>
<td>King</td>
<td>24000</td>
<td>289200</td>
</tr>
<tr>
<td>Kochhar</td>
<td>17000</td>
<td>205200</td>
</tr>
<tr>
<td>De Haan</td>
<td>17000</td>
<td>205200</td>
</tr>
<tr>
<td>Hunold</td>
<td>9000</td>
<td>109200</td>
</tr>
</tbody>
</table>
Defining a Null Value

- Null is a value that is unavailable, unassigned, unknown, or inapplicable.
- Null is not the same as zero or a blank space.

```
SELECT last_name, job_id, salary, commission_pct
FROM employees;
```
Null Values in Arithmetic Expressions

Arithmetic expressions containing a null value to evaluate to null.

```sql
SELECT last_name, 12*salary*commission_pct
FROM employees;
```

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>12<em>Salary</em>Commission_Pct</th>
</tr>
</thead>
<tbody>
<tr>
<td>King</td>
<td>-</td>
</tr>
<tr>
<td>Kochhar</td>
<td>-</td>
</tr>
<tr>
<td>De Haan</td>
<td>-</td>
</tr>
<tr>
<td>Hunold</td>
<td>-</td>
</tr>
</tbody>
</table>

...  

<table>
<thead>
<tr>
<th>Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Russell</td>
<td>67200</td>
</tr>
<tr>
<td>Partners</td>
<td>48600</td>
</tr>
<tr>
<td>Errazuriz</td>
<td>43200</td>
</tr>
<tr>
<td>Cambrault</td>
<td>39600</td>
</tr>
</tbody>
</table>
Defining a Column Alias

A column alias:

• Renames a column heading
• Is useful with calculations
• Immediately follows the column name (There can also be the optional \texttt{AS} keyword between the column name and the alias.)
• Requires double quotation marks if it contains spaces or special characters or if it is case-sensitive
Using Column Aliases

SELECT last_name AS name, commission_pct comm
FROM employees;

```
NAME   COMM
King   -
Kochhar -
De Haan -
Hunold -
...  
```

SELECT last_name "Name", salary*12 "Annual Salary"
FROM employees;

```
Name     Annual Salary
King     288000
Kochhar  204000
De Haan  204000
Hunold   108000
Ernst    72000
```
Concatenation Operator

- Links columns or character strings to other columns
- Is represented by two vertical bars (||)
- Creates a column that is a character expression

```
SELECT last_name||job_id AS "Employees"
FROM employees;
```
Literal Character Strings

• A literal is a character, a number, or a date that is included in the `SELECT` statement.

• Date and character literal values must be enclosed within single quotation marks.

• Each character string is output once for each row returned.
Using Literal Character Strings

SELECT last_name || ' is a ' || job_id AS "Employee Details"
FROM employees;

<table>
<thead>
<tr>
<th>Employee Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abel is a SA REP</td>
</tr>
<tr>
<td>Ande is a SA REP</td>
</tr>
<tr>
<td>Atkinson is a ST_CLERK</td>
</tr>
<tr>
<td>Austin is a IT_PROG</td>
</tr>
<tr>
<td>Baer is a PR REP</td>
</tr>
<tr>
<td>Baida is a PU_CLERK</td>
</tr>
<tr>
<td>Banda is a SA REP</td>
</tr>
</tbody>
</table>
Alternative Quote (q) Operator

```
SELECT department_name || q'[ Department's Manager Id: ]' || manager_id
    AS "Department and Manager"
FROM departments;
```

```
Department and Manager
Administration Department's Manager Id: 200
Marketing Department's Manager Id: 201
Purchasing Department's Manager Id: 114
Human Resources Department's Manager Id: 203
Shipping Department's Manager Id: 121
IT Department's Manager Id: 103
```
Case Scenario: Using the Column Alias

Faculty

I can create a simple query using the SELECT statement and display that information.

Student

Sean, I would like to see the different locations where the members are located.
Case Scenario: Using the SELECT Statement

Here the concatenation operator as well as the column alias has been used.

Successful retrieval of data

```
SELECT FIRST_NAME || ' IS LOCATED IN ' || CITY
AS "MEMBER LOCATION"
FROM MEMBERS;
```

MEMBER LOCATION
Velasquez Carmen IS LOCATED IN Seattle
Ngao LaDoris IS LOCATED IN Bratislava
Nagayama Midori IS LOCATED IN Sao Paolo
Quick-To-See Mark IS LOCATED IN Lagos
Ropeburn Audry IS LOCATED IN Hong Kong
Urguhart Molly IS LOCATED IN Quebec
Menchu Roberta IS LOCATED IN Brussels
Biri Ben IS LOCATED IN Columbus

8 rows selected
Duplicate Rows

The default display of queries is all rows, including duplicate rows.

1. SELECT department_id FROM employees;

2. SELECT DISTINCT department_id FROM employees;

---

<table>
<thead>
<tr>
<th>DEPARTMENT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
</tr>
<tr>
<td>90</td>
</tr>
<tr>
<td>90</td>
</tr>
<tr>
<td>60</td>
</tr>
<tr>
<td>60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEPARTMENT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td>-</td>
</tr>
<tr>
<td>90</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>70</td>
</tr>
</tbody>
</table>
Displaying the Table Structure

• Use the `DESCRIBE` command to display the structure of a table.

• Or, select the table in the Connections tree and use the Columns tab to view the table structure.

```
DESC[RIBE] tablename
```
Using the DESCRIBE Command

```
DESCRIBE employees
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Null</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE_ID</td>
<td>NOT NULL</td>
<td>NUMBER(6)</td>
</tr>
<tr>
<td>FIRST_NAME</td>
<td></td>
<td>VARCHAR2(20)</td>
</tr>
<tr>
<td>LAST_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(25)</td>
</tr>
<tr>
<td>EMAIL</td>
<td>NOT NULL</td>
<td>VARCHAR2(25)</td>
</tr>
<tr>
<td>PHONE_NUMBER</td>
<td>NOT NULL</td>
<td>VARCHAR2(20)</td>
</tr>
<tr>
<td>HIRE_DATE</td>
<td>NOT NULL</td>
<td>DATE</td>
</tr>
<tr>
<td>JOB_ID</td>
<td>NOT NULL</td>
<td>VARCHAR2(10)</td>
</tr>
<tr>
<td>SALARY</td>
<td></td>
<td>NUMBER(8,2)</td>
</tr>
<tr>
<td>COMMISSION_PCT</td>
<td></td>
<td>NUMBER(2,2)</td>
</tr>
<tr>
<td>MANAGER_ID</td>
<td></td>
<td>NUMBER(6)</td>
</tr>
<tr>
<td>DEPARTMENT_ID</td>
<td></td>
<td>NUMBER(4)</td>
</tr>
</tbody>
</table>
Summary

In this lesson, you should have learned how to:

• List the capabilities of SQL \texttt{SELECT} statements

• Write and execute a \texttt{SELECT} statement that:
  – Returns all rows and columns from a table
  – Returns specific columns from a table
  – Uses column aliases to display descriptive column headings
  – Uses arithmetic and concatenation operators
  – Uses literal character strings
  – Eliminates duplicate rows

• Describe the structure of a table