2

Retrieving Data Using the SQL `SELECT` Statement

ORACLE

Objectives

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

- List the capabilities of SQL SELECT statements
- Execute a basic SELECT statement

To extract data from the database, you need to use the SQL SELECT statement. However, you may need to restrict the columns that are displayed. This lesson describes the SELECT statement that is needed to perform these actions. Further, you may want to create SELECT statements that can be used more than once.

Lesson Agenda

- **Capabilities of SQL SELECT statements**
- Arithmetic expressions and NULL values in the SELECT statement
- Column aliases
- Use of concatenation operator, literal character strings, alternative quote operator, and the DISTINCT keyword
- DESCRIBE command

Basic SELECT Statement

```
SELECT   *|{[DISTINCT] column [alias],...}
FROM     table;
```

- SELECT identifies the columns to be displayed.
- FROM identifies the table containing those columns.

In its simplest form, a SELECT statement must include the following:
- A SELECT clause, which specifies the columns to be displayed
- A FROM clause, which identifies the table containing the columns that are listed in the SELECT clause

In the syntax:

SELECT	Is a list of one or more columns
*	Selects all columns
DISTINCT	Suppresses duplicates
column|expression	Selects the named column or the expression
alias	Gives different headings to the selected columns
FROM table	Specifies the table containing the columns

Note: Throughout this course, the words *keyword*, *clause*, and *statement* are used as follows:
- A *keyword* refers to an individual SQL element—for example, SELECT and FROM are keywords.
- A *clause* is a part of a SQL statement—for example, SELECT employee_id, last_name, and so on.
- A *statement* is a combination of two or more clauses—for example, SELECT * FROM employees.

Selecting All Columns

```
SELECT *
FROM   departments;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

You can display all columns of data in a table by following the SELECT keyword with an asterisk (*). In the example in the slide, the DEPARTMENTS table contains four columns: DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID, and LOCATION_ID. The table contains eight rows, one for each department.

You can also display all columns in the table by listing them after the SELECT keyword. For example, the following SQL statement (like the example in the slide) displays all columns and all rows of the DEPARTMENTS table:

```
SELECT  department_id, department_name, manager_id, location_id
FROM    departments;
```

Note: In SQL Developer, you can enter your SQL statement in a SQL Worksheet and click the "Execute Statement" icon or press [F9] to execute the statement. The output displayed on the Results tabbed page appears as shown in the slide.

Selecting Specific Columns

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

	DEPARTMENT_ID	LOCATION_ID
1	10	1700
2	20	1800
3	50	1500
4	60	1400
5	80	2500
6	90	1700
7	110	1700
8	190	1700

You can use the SELECT statement to display specific columns of the table by specifying the column names, separated by commas. The example in the slide displays all the department numbers and location numbers from the DEPARTMENTS table.

In the SELECT clause, specify the columns that you want in the order in which you want them to appear in the output. For example, to display location before department number (from left to right), you use the following statement:

```
SELECT location_id, department_id
FROM   departments 6
```

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.
- Clauses are usually placed on separate lines.
- Indents are used to enhance readability.
- In SQL Developer, SQL statements can be optionally 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 (;).

Writing SQL Statements

By using the following simple rules and guidelines, you can construct valid statements that are both easy to read and edit:

- SQL statements are not case-sensitive (unless indicated).
- SQL statements can be entered on one or many lines.
- Keywords cannot be split across lines or abbreviated.
- Clauses are usually placed on separate lines for readability and ease of editing.
- Indents should be used to make code more readable.
- Keywords typically are entered in uppercase; all other words, such as table names and columns names, are entered in lowercase.

Executing SQL Statements

In SQL Developer, click the Run Script icon or press [F5] to run the command or commands in the SQL Worksheet. You can also click the Execute Statement icon or press [F9] to run a SQL statement in the SQL Worksheet. The Execute Statement icon executes the statement at the cursor in the Enter SQL Statement box while the Run Script icon executes all the statements in the Enter SQL Statement box. The Execute Statement icon displays the output of the query on the Results tabbed page, whereas the Run Script icon emulates the SQL*Plus display and shows the output on the Script Output tabbed page.

In SQL*Plus, terminate the SQL statement with a semicolon, and then press [Enter] to run the command.

Column Heading Defaults

- SQL Developer:
  - Default heading alignment: Left-aligned
  - Default heading display: Uppercase
- SQL*Plus:
  - Character and Date column headings are left-aligned.
  - Number column headings are right-aligned.
  - Default heading display: Uppercase

In SQL Developer, column headings are displayed in uppercase and are left-aligned.

```
SELECT last_name, hire_date, salary
FROM   employees;
```

You can override the column heading display with an alias. Column aliases are covered later in this lesson.

Lesson Agenda

- Capabilities of SQL SELECT statements
- **Arithmetic expressions and NULL values in the SELECT statement**
- Column aliases
- Use of concatenation operator, literal character strings, alternative quote operator, and the DISTINCT keyword
- DESCRIBE command

ORACLE

Arithmetic Expressions

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

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

You may need to modify the way in which data is displayed, or you may want to perform calculations, or look at what-if scenarios. All these are possible using arithmetic expressions. An arithmetic expression can contain column names, constant numeric values, and the arithmetic operators.

Arithmetic Operators

The slide lists the arithmetic operators that are available in SQL. You can use arithmetic operators in any clause of a SQL statement (except the FROM clause).

Note: With the DATE and TIMESTAMP data types, you can use the addition and subtraction operators only.

Using Arithmetic Operators

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

	LAST_NAME	SALARY	SALARY+300
1	King	24000	24300
2	Kochhar	17000	17300
3	De Haan	17000	17300
4	Hunold	9000	9300
5	Ernst	6000	6300
6	Lorentz	4200	4500
7	Mourgos	5800	6100
8	Rajs	3500	3800
9	Davies	3100	3400
10	Matos	2600	2900

...

ORACLE

The example in the slide uses the addition operator to calculate a salary increase of $300 for all employees. The slide also displays a SALARY+300 column in the output.

Note that the resultant calculated column, SALARY+300, is not a new column in the EMPLOYEES table; it is for display only. By default, the name of a new column comes from the calculation that generated it—in this case, salary+300.

Note: The Oracle server ignores blank spaces before and after the arithmetic operator.

Rules of Precedence

- Multiplication and division occur before addition and subtraction.
- Operators of the same priority are evaluated from left to right.
- Parentheses are used to override the default precedence or to clarify the statement.

Operator Precedence

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

	LAST_NAME	SALARY	12*SALARY+100
1	King	24000	288100
2	Kochhar	17000	204100
3	De Haan	17000	204100
4	Hunold	9000	108100

...

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

	LAST_NAME	SALARY	12*(SALARY+100)
1	King	24000	289200
2	Kochhar	17000	205200
3	De Haan	17000	205200
4	Hunold	9000	109200

...

ORACLE

The first example in the slide displays the last name, salary, and annual compensation of employees. It calculates the annual compensation by multiplying the monthly salary with 12, plus a one-time bonus of $100. Note that multiplication is performed before addition.

Note: Use parentheses to reinforce the standard order of precedence and to improve clarity. For example, the expression in the slide can be written as `(12*salary)+100` with no change in the result.

Using Parentheses

You can override the rules of precedence by using parentheses to specify the desired order in which the operators are to be executed.

The second example in the slide displays the last name, salary, and annual compensation of employees. It calculates the annual compensation as follows: adding a monthly bonus of $100 to the monthly salary, and then multiplying that subtotal with 12. Because of the parentheses, addition takes priority over multiplication.

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;
```

	LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
1	King	AD_PRES	24000	(null)
2	Kochhar	AD_VP	17000	(null)
3	De Haan	AD_VP	17000	(null)

...

12	Zlotkey	SA_MAN	10500	0.2
13	Abel	SA_REP	11000	0.3
14	Taylor	SA_REP	8600	0.2
15	Grant	SA_REP	7000	0.15

...

18	Fay	MK_REP	6000	(null)
19	Higgins	AC_MGR	12008	(null)
20	Gietz	AC_ACCOUNT	8300	(null)

ORACLE

If a row lacks a data value for a particular column, that value is said to be NULL or to contain a null.

Columns with NULL value can be selected in a SELECT query and can be the part of an arithmetic expression. Any arithmetic expression using NULL values results into NULL.

Columns of any data type can contain nulls. However, some constraints (NOT NULL and PRIMARY KEY) prevent nulls from being used in the column.

In the slide example, notice that only a sales manager or sales representative can earn a commission in the COMMISSION_PCT column of the EMPLOYEES table. Other employees are not entitled to earn commissions. A null represents that fact.

Note: By default, SQL Developer uses the literal, (null), to identify null values. However, you can set it to something more relevant to you. To do so, select Preferences from the Tools menu. In the Preferences dialog box, expand the Database node. Click Advanced Parameters and on the right pane, for the "Display Null value As," enter the appropriate value.

Null Values in Arithmetic Expressions

Arithmetic expressions containing a null value evaluate to null.

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

	LAST_NAME	12*SALARY*COMMISSION_PCT
1	King	(null)
2	Kochhar	(null)
3	De Haan	(null)

...

12	Zlotkey	25200
13	Abel	39600
14	Taylor	20640
15	Grant	12600

...

17	Hartstein	(null)
18	Fay	(null)
19	Higgins	(null)
20	Gietz	(null)

If any column value in an arithmetic expression is null, the result is null. For example, if you attempt to perform division by zero, you get an error. However, if you divide a number by null, the result is a null or unknown.

In the example in the slide, employee King does not get any commission. Because the COMMISSION_PCT column in the arithmetic expression is null, the result is null.

For more information, see the section on "Basic Elements of Oracle SQL" in *Oracle Database SQL Language Reference* for 12*c* database.

Lesson Agenda

- Capabilities of SQL `SELECT` statements
- Arithmetic expressions and `NULL` values in the `SELECT` statement
- **Column aliases**
- Use of concatenation operator, literal character strings, alternative quote operator, and the `DISTINCT` keyword
- `DESCRIBE` command

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 `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

When displaying the result of a query, SQL Developer normally uses the name of the selected column as the column heading. This heading may not be descriptive and, therefore, may be difficult to understand. You can change a column heading by using a column alias.

Specify the alias after the column in the `SELECT` list using blank space as a separator. By default, alias headings appear in uppercase. If the alias contains spaces or special characters (such as `-`, `!`, `_`), or if it is case-sensitive, enclose the alias in double quotation marks (" ").

Using Column Aliases

```
SELECT last_name AS name, commission_pct comm
FROM   employees;
```

	NAME	COMM
1	King	(null)
2	Kochhar	(null)
3	De Haan	(null)
4	Hunold	(null)

...

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

	Name	Annual Salary
1	King	288000
2	Kochhar	204000
3	De Haan	204000
4	Hunold	108000

...

The first example displays the names and the commission percentages of all the employees. Note that the optional AS keyword has been used before the column alias name. The result of the query is the same whether the AS keyword is used or not. Also, note that the SQL statement has the column aliases, name and comm, in lowercase, whereas the result of the query displays the column headings in uppercase. As mentioned in the preceding slide, column headings appear in uppercase by default.

The second example displays the last names and annual salaries of all the employees. Because Annual Salary contains a space, it has been enclosed in double quotation marks. Note that the column heading in the output is exactly the same as the column alias.

Lesson Agenda

- Capabilities of SQL SELECT statements
- Arithmetic Expressions and NULL values in SELECT statement
- Column aliases
- **Use of concatenation operator, literal character strings, alternative quote operator, and the DISTINCT keyword**
- DESCRIBE command

Concatenation Operator

A concatenation operator:

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

```
SELECT    last_name||job_id AS "Employees"
FROM      employees;
```

📄	Employees
1	AbelSA_REP
2	DaviesST_CLERK
3	De HaanAD_VP
4	ErnstIT_PROG
5	FayMK_REP
6	GietzAC_ACCOUNT
7	GrantSA_REP
8	HartsteinMK_MAN

...

You can link columns to other columns, arithmetic expressions, or constant values to create a character expression by using the concatenation operator (||). Columns on either side of the operator are combined to make a single output column.

In the example, LAST_NAME and JOB_ID are concatenated, and given the alias Employees. Note that the last name of the employee and the job code are combined to make a single output column.

The AS keyword before the alias name makes the SELECT clause easier to read.

Null Values with the Concatenation Operator

If you concatenate a null value with a character string, the result is a character string. LAST_NAME || NULL results in LAST_NAME.

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.

A literal is a character, a number, or a date that is included in the `SELECT` list. It is not a column name or a column alias. It is printed for each row returned. Literal strings of free-format text can be included in the query result and are treated the same as a column in the `SELECT` list.

The date and character literals *must* be enclosed within single quotation marks (`' '`); number literals need not be enclosed in a similar manner.

Using Literal Character Strings

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

	Employee Details
1	Abel is a SA_REP
2	Davies is a ST_CLERK
3	De Haan is a AD_VP
4	Ernst is a IT_PROG
5	Fay is a MK_REP
6	Gietz is a AC_ACCOUNT
7	Grant is a SA_REP
8	Hartstein is a MK_MAN
9	Higgins is a AC_MGR
10	Hunold is a IT_PROG
11	King is a AD_PRES

...

The example in the slide displays the last names and job codes of all employees. The column has the heading Employee Details. Note the spaces between the single quotation marks in the SELECT statement. The spaces improve the readability of the output.

In the following example, the last name and salary for each employee are concatenated with a literal, to give the returned rows more meaning:

```
SELECT last_name ||': 1 Month salary = '||salary Monthly
FROM   employees;
```

Alternative Quote (q) Operator

- Specify your own quotation mark delimiter.
- Select any delimiter.
- Increase readability and usability.

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

	Department and Manager
1	Administration Department's Manager Id: 200
2	Marketing Department's Manager Id: 201
3	Shipping Department's Manager Id: 124
4	IT Department's Manager Id: 103
5	Sales Department's Manager Id: 149
6	Executive Department's Manager Id: 100
7	Accounting Department's Manager Id: 205
8	Contracting Department's Manager Id:

ORACLE

Many SQL statements use character literals in expressions or conditions. If the literal itself contains a single quotation mark, you can use the quote (q) operator and select your own quotation mark delimiter.

You can choose any convenient delimiter, single-byte or multibyte, or any of the following character pairs: [ ], { }, ( ), or < >.

In the example shown, the string contains a single quotation mark, which is normally interpreted as a delimiter of a character string. By using the q operator, however, brackets [ ] are used as the quotation mark delimiters. The string between the brackets delimiters is interpreted as a literal character string.

Duplicate Rows

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

1
```
SELECT department_id
FROM    employees;
```

	DEPARTMENT_ID
1	90
2	90
3	90
4	60
5	60
6	60
7	50
8	50

…

2
```
SELECT DISTINCT department_id
FROM    employees;
```

	DEPARTMENT_ID
1	(null)
2	90
3	20
4	110
5	50
6	80
7	60
8	10

Unless you indicate otherwise, SQL displays the results of a query without eliminating the duplicate rows. The first example in the slide displays all the department numbers from the `EMPLOYEES` table. Note that the department numbers are repeated.

To eliminate duplicate rows in the result, include the `DISTINCT` keyword in the `SELECT` clause immediately after the `SELECT` keyword. In the second example in the slide, the `EMPLOYEES` table actually contains 20 rows, but there are only seven unique department numbers in the table.

You can specify multiple columns after the `DISTINCT` qualifier. The `DISTINCT` qualifier affects all the selected columns, and the result is every distinct combination of the columns.

```
SELECT  DISTINCT department_id, job_id
FROM    employees;
```

Lesson Agenda

- Capabilities of SQL SELECT statements
- Arithmetic expressions and NULL values in the SELECT statement
- Column aliases
- Use of concatenation operator, literal character strings, alternative quote operator, and the DISTINCT keyword
- DESCRIBE command

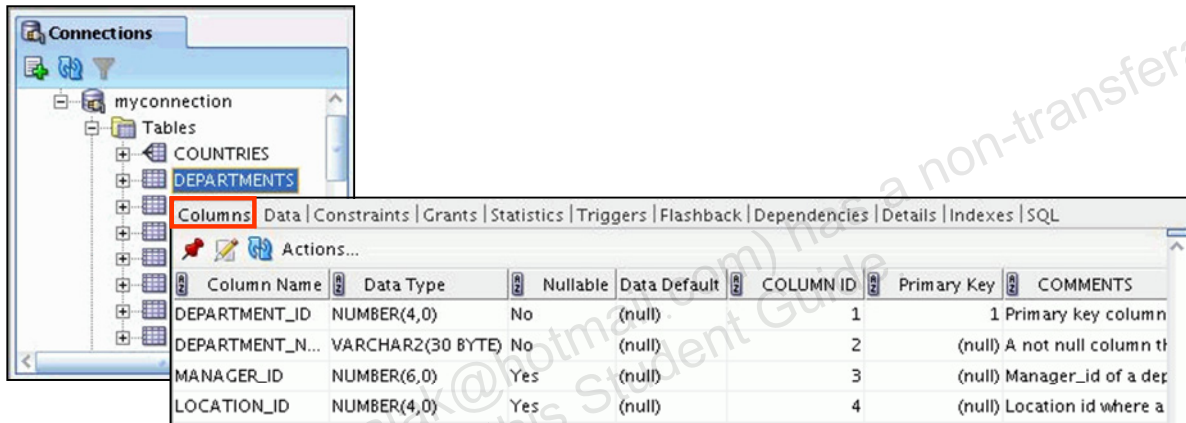# Displaying Table Structure

- 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
```

You can display the structure of a table by using the `DESCRIBE` command. The command displays the column names and the data types, and it shows you whether a column *must* contain data (that is, whether the column has a `NOT NULL` constraint).

In the syntax, `table name` is the name of any existing table, view, or synonym that is accessible to the user.

Using the SQL Developer GUI interface, you can select the table in the Connections tree and use the Columns tab to view the table structure.

Note: `DESCRIBE` is a SQL*Plus command supported by SQL Developer. It is abbreviated as `DESC`.

Using the DESCRIBE Command

```
DESCRIBE employees
```

```
DESCRIBE Employees
Name              Null      Type
---------------   --------  ------------
EMPLOYEE_ID       NOT NULL  NUMBER(6)
FIRST_NAME                  VARCHAR2(20)
LAST_NAME         NOT NULL  VARCHAR2(25)
EMAIL             NOT NULL  VARCHAR2(25)
PHONE_NUMBER                VARCHAR2(20)
HIRE_DATE         NOT NULL  DATE
JOB_ID            NOT NULL  VARCHAR2(10)
SALARY                      NUMBER(8,2)
COMMISSION_PCT              NUMBER(2,2)
MANAGER_ID                  NUMBER(6)
DEPARTMENT_ID               NUMBER(4)
```

The example in the slide displays information about the structure of the EMPLOYEES table using the DESCRIBE command.

In the resulting display, *Null* indicates that the values for this column may be unknown. NOT NULL indicates that a column must contain data. *Type* displays the data type for a column.

The data types are described in the following table:

Data Type	Description
NUMBER(p,s)	Number value having a maximum number of digits p, with s digits to the right of the decimal point
VARCHAR2(s)	Variable-length character value of maximum size s
DATE	Date and time value between January 1, 4712 B.C. and December 31, A.D. 9999

Quiz

Identify the `SELECT` statements that execute successfully.

a.
```
SELECT first_name, last_name, job_id, salary*12
  AS Yearly Sal
FROM    employees;
```

b.
```
SELECT first_name, last_name, job_id, salary*12
  "yearly sal"
FROM    employees;
```

c.
```
SELECT first_name, last_name, job_id, salary AS
  "yearly sal"
FROM    employees;
```

d.
```
SELECT first_name+last_name AS name, job_Id,
  salary*12 yearly sal
FROM    employees;
```

ORACLE

Answer: b, c

Summary

In this lesson, you should have learned how to write a `SELECT` statement that:

- Returns all rows and columns from a table
- Returns specified columns from a table
- Uses column aliases to display more descriptive column headings

In this lesson, you should have learned how to retrieve data from a database table with the `SELECT` statement.

```
SELECT   *|{[DISTINCT] column [alias],...}
FROM     table;
```

In the syntax:

SELECT	Is a list of one or more columns
*	Selects all columns
DISTINCT	Suppresses duplicates
column|expression	Selects the named column or the expression
alias	Gives different headings to the selected columns
FROM table	Specifies the table containing the columns

Practice 2: Overview

This practice covers the following topics:

- Selecting all data from different tables
- Describing the structure of tables
- Performing arithmetic calculations and specifying column names

In this practice, you write simple `SELECT` queries. The queries cover most of the `SELECT` clauses and operations that you learned in this lesson.