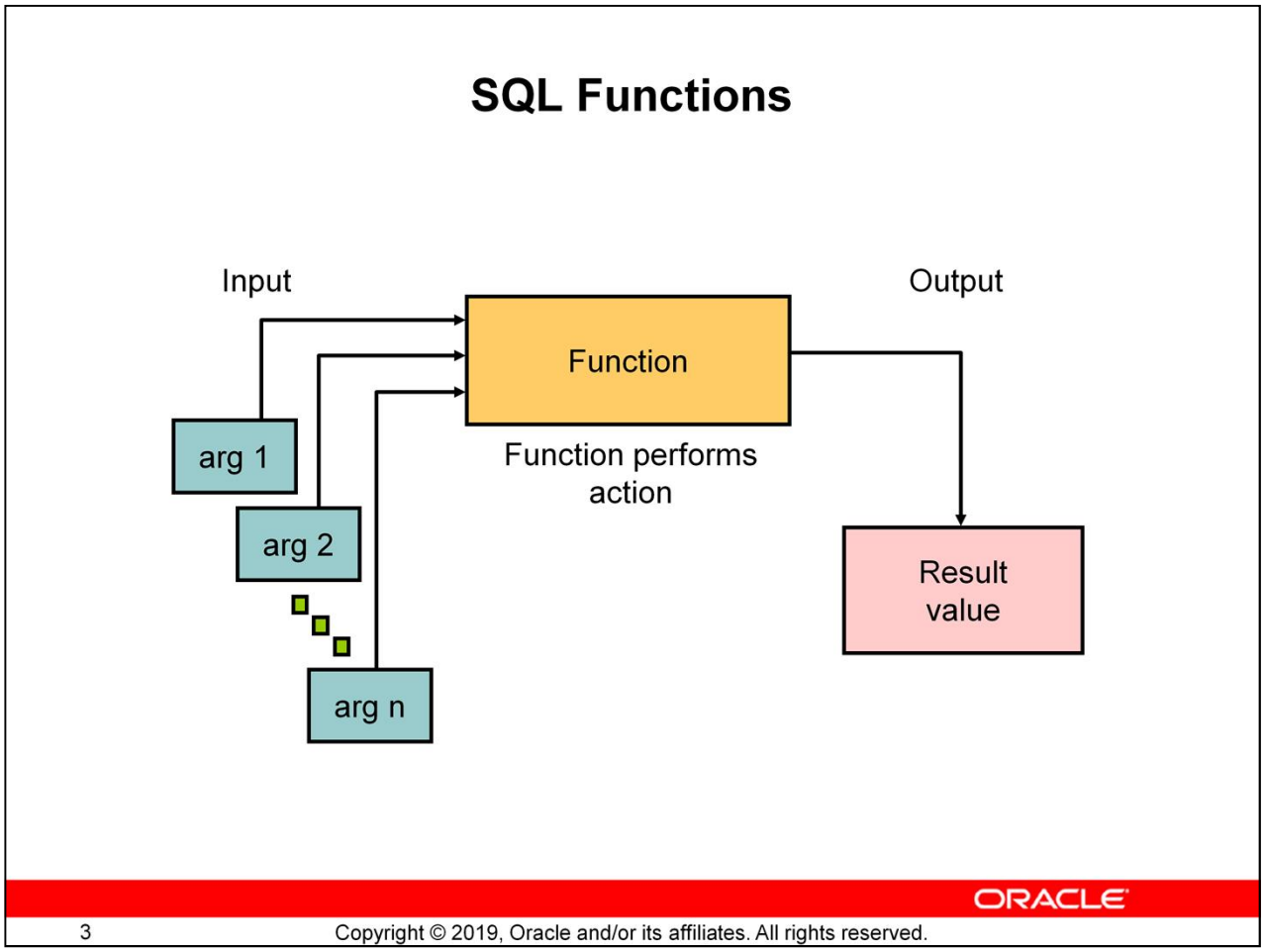
Using Single-Row Functions to Customize Output

Objectives

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

- Describe the various types of functions available in SQL
- Use the character, number, and date functions in `SELECT` statements

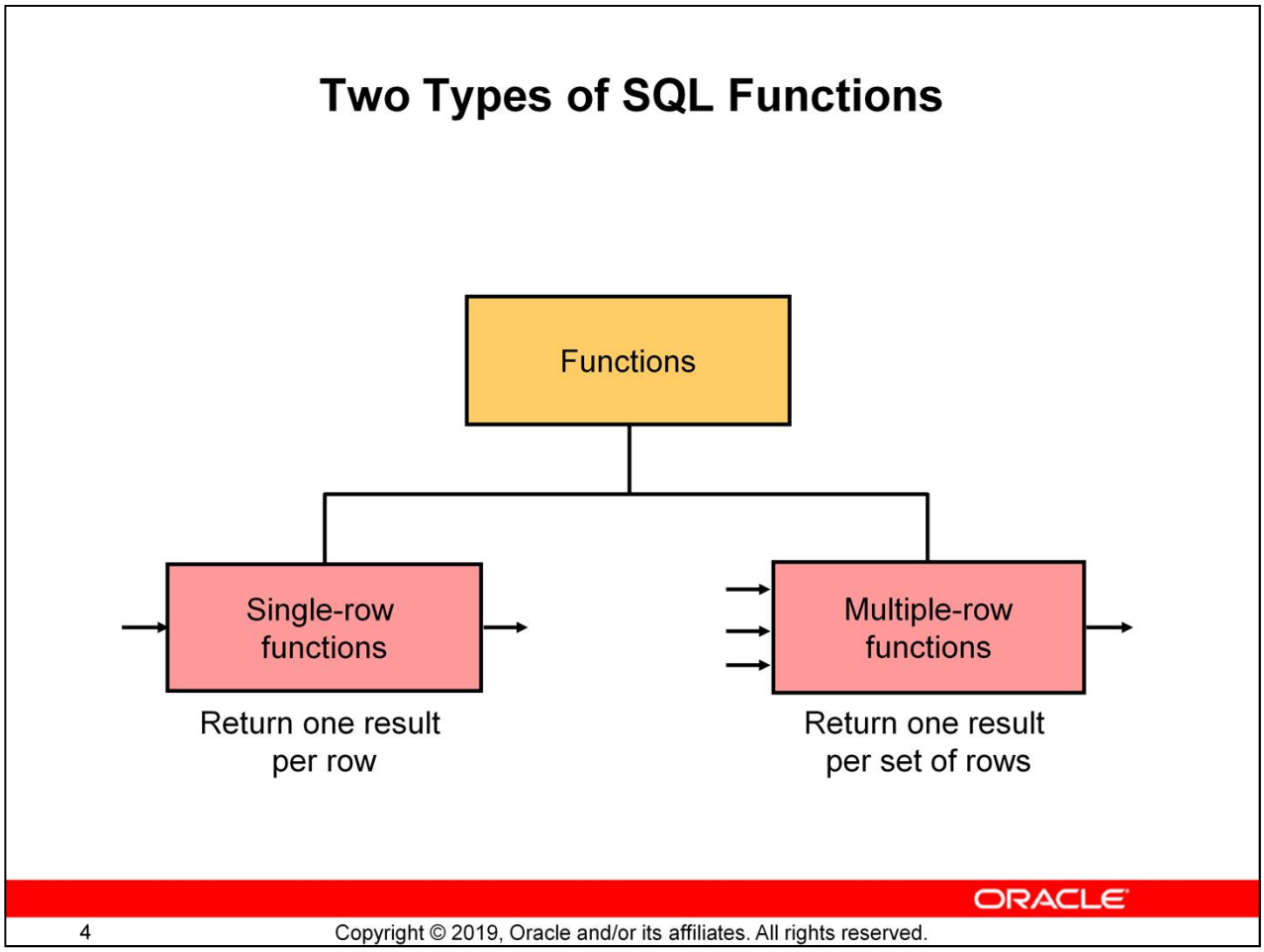
Functions make the basic query block more powerful, and they are used to manipulate data values. This is the first of two lessons that explore functions. It focuses on single-row character, number, and date functions.



Functions are a very powerful feature of SQL. They can be used to do the following:

- Perform calculations on data
- Modify individual data items
- Manipulate output for groups of rows
- Format dates and numbers for display
- Convert column data types

SQL functions sometimes take arguments and always return a value.



There are two types of functions:

- Single-row functions
- Multiple-row functions

Single-Row Functions

These functions operate on single rows only and return one result per row. There are different types of single-row functions. This lesson covers the following functions:

- Character
- Number
- Date
- Conversion
- General

Multiple-Row Functions

Functions can manipulate groups of rows to give one result per group of rows. These functions are also known as *group functions* (covered in the lesson titled “Reporting Aggregated Data Using the Group Functions”).

Note: For more information and a complete list of available functions and their syntax, see the “Functions” section in *Oracle Database SQL Language Reference* for 19c database.

Single-Row Functions

Single-row functions:

- Manipulate data items
- Accept arguments and return one value
- Act on each row that is returned
- Return one result per row
- May modify the data type
- Can be nested
- Accept arguments that can be a column or an expression

```
function_name [(arg1, arg2,...)]
```


Single-row functions are used to manipulate data items. They accept one or more arguments and return one value for each row that is returned by the query. An argument can be one of the following:

- User-supplied constant
- Variable value
- Column name
- Expression

Features of single-row functions include:

- Acting on each row that is returned in the query
- Returning one result per row
- Possibly returning a data value of a different type than the one that is referenced
- Possibly expecting one or more arguments
- Can be used in `SELECT`, `WHERE`, and `ORDER BY` clauses; can be nested.

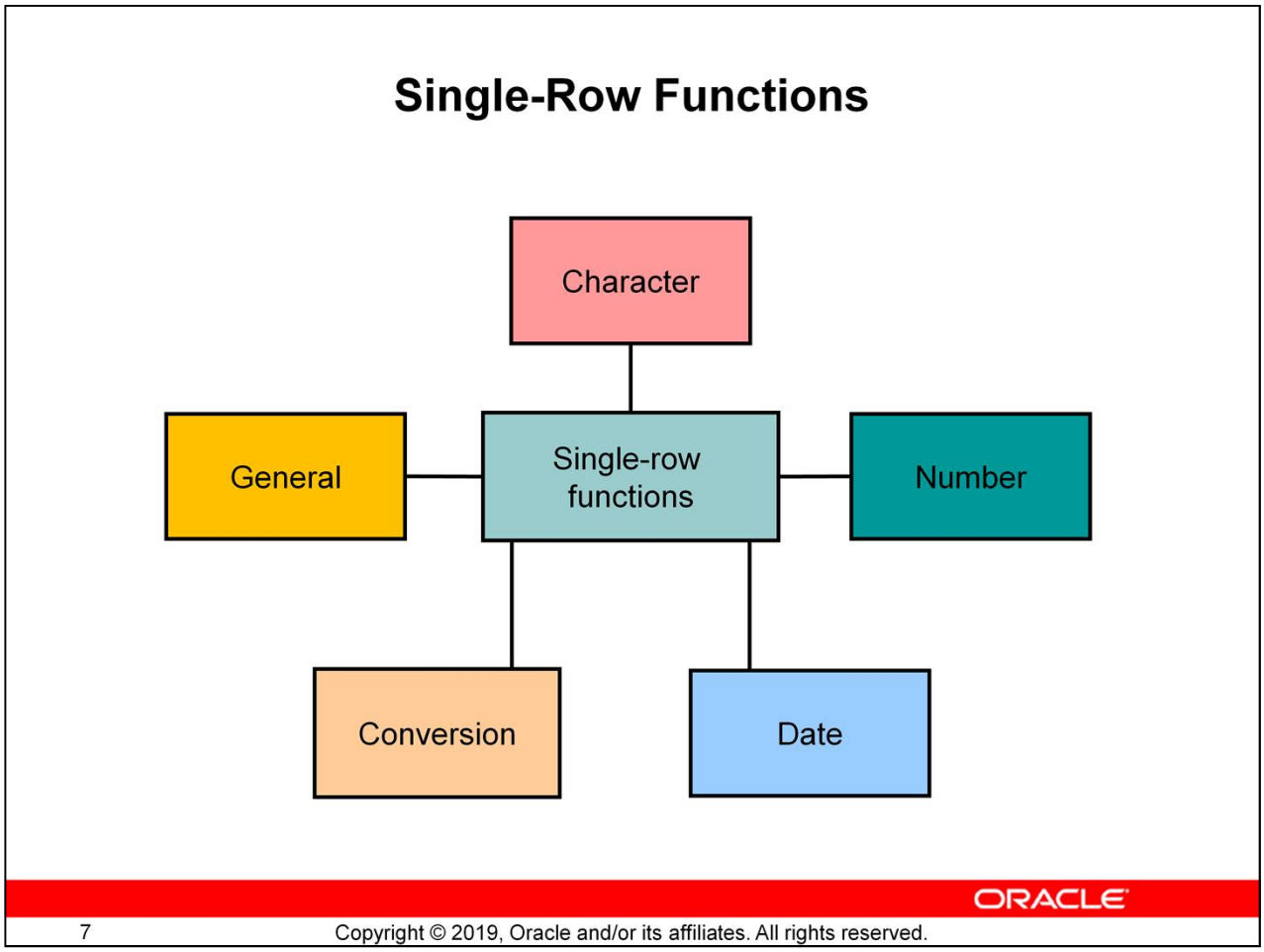
In the syntax:

function_name Is the name of the function

arg1, arg2	Is any argument to be used by the function. This can be represented by a column name or expression.

Single-Row Functions

- **Character functions:** Accept character input and can return both character and number values
- **Number functions:** Accept numeric input and return numeric values
- **Date functions:** Operate on values of the `DATE` data type (All date functions return a value of the `DATE` data type except the `MONTHS_BETWEEN` function, which returns a number.)



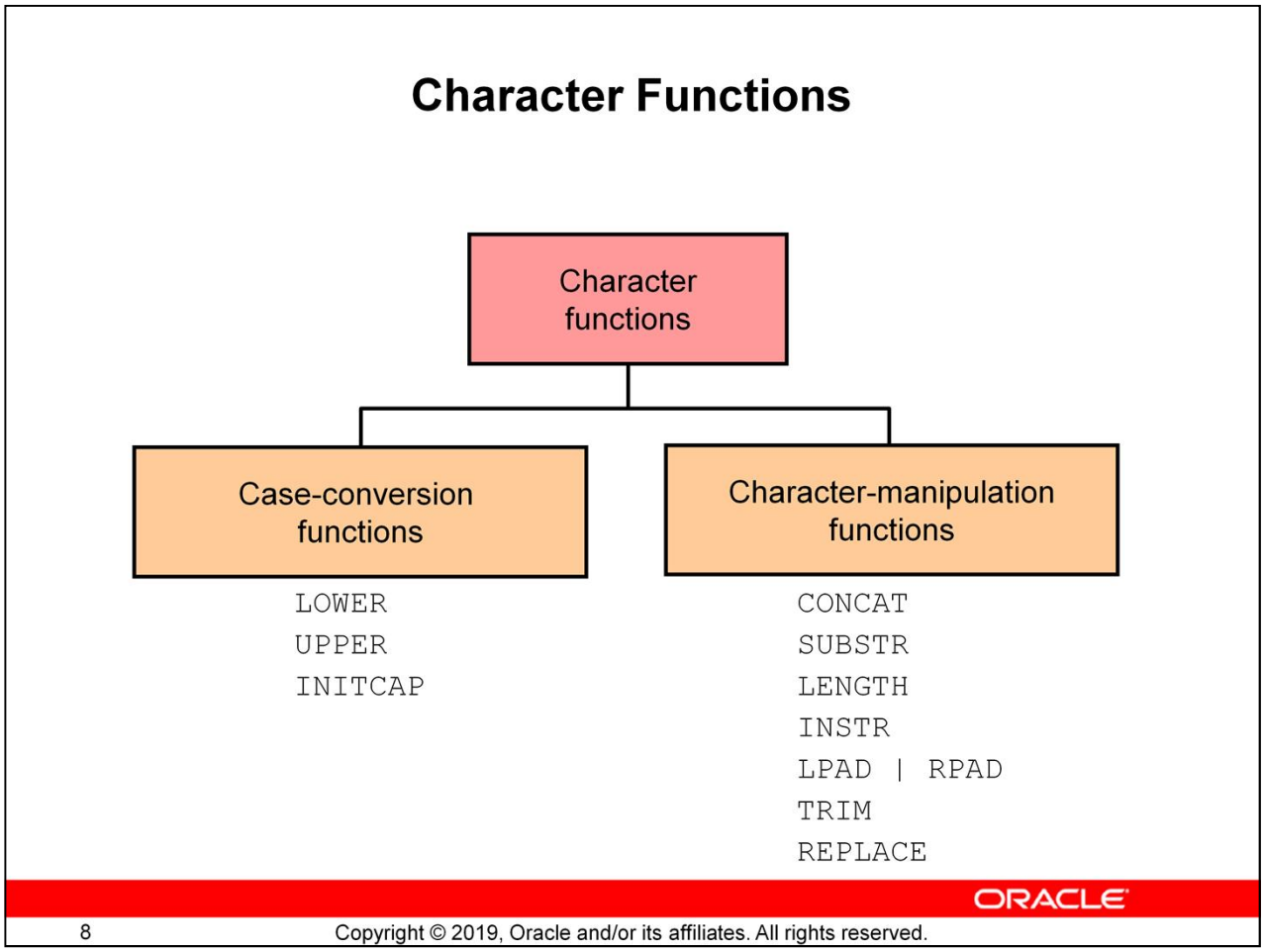
This lesson covers the following single-row functions:

- **Character functions:** Accept character input and can return both character and number values
- **Number functions:** Accept numeric input and return numeric values
- **Date functions:** Operate on values of the `DATE` data type (All date functions return a value of the `DATE` data type except the `MONTHS_BETWEEN` function, which returns a number.)

The following single-row functions are discussed in the lesson titled “Using Conversion Functions and Conditional Expressions”:

- **Conversion functions:** Convert a value from one data type to another
- **General functions:**
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
  - `CASE`

- DECODE



Single-row character functions accept character data as input and can return both character and numeric values. Character functions can be divided into the following:

- Case-conversion functions
- Character-manipulation functions

Function	Purpose
`LOWER(column|expression)`	Converts alpha character values to lowercase
`UPPER(column|expression)`	Converts alpha character values to uppercase
`INITCAP(column|expression)`	Converts alpha character values to uppercase for the first letter of each word; all other letters in lowercase
`CONCAT(column1|expression1, column2|expression2)`	Concatenates the first character value to the second character value; equivalent to concatenation operator (||)
`SUBSTR(column|expression,m[ ,n])`	Returns specified characters from character value starting at character position *m, n* characters long (If *m* is negative, the count starts from the end of the character value. If *n* is omitted, all characters to the end of the string are returned.)

Note: The functions discussed in this lesson are only some of the available functions.

Case-Conversion Functions

These functions convert the case for character strings:

Function	Result
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL Course')	Sql Course

- **LOWER:** Converts mixed-case or uppercase character strings to lowercase
- **UPPER:** Converts mixed-case or lowercase character strings to uppercase
- **INITCAP:** Converts the first letter of each word to uppercase and the remaining letters to lowercase

ORACLE

LOWER, UPPER, and INITCAP are the three case-conversion functions.

- **LOWER:** Converts mixed-case or uppercase character strings to lowercase
- **UPPER:** Converts mixed-case or lowercase character strings to uppercase
- **INITCAP:** Converts the first letter of each word to uppercase and the remaining letters to lowercase

```
SELECT 'The job id for '||UPPER(last_name)||' is '
 ||LOWER(job_id) AS "EMPLOYEE DETAILS"
FROM   employees;
```

	EMPLOYEE DETAILS
1	The job id for ABEL is sa_rep
2	The job id for DAVIES is st_clerk
3	The job id for DE HAAN is ad_vp
4	The job id for ERNST is it_prog
5	The job id for FAY is mk_rep
6	The job id for GIETZ is ac_account
7	The job id for GRANT is sa_rep
8	The job id for HARTSTEIN is mk_man

. . .

Using Case-Conversion Functions

Display the employee number, name, and department number for employee Higgins:

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  last_name = 'higgins';
```

```
0 rows selected
```

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  LOWER(last_name) = 'higgins';
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	205	Higgins	110

The slide example displays the employee number, name, and department number of employee Higgins.

The `WHERE` clause of the first SQL statement specifies the employee name as `higgins`. Because all the data in the `EMPLOYEES` table is stored in proper case, the name `higgins` does not find a match in the table, and no rows are selected.

The `WHERE` clause of the second SQL statement specifies that the employee name in the `EMPLOYEES` table is compared to `higgins`, converting the `LAST_NAME` column to lowercase for comparison purposes. Because both names are now lowercase, a match is found and one row is selected. The `WHERE` clause can be rewritten in the following manner to produce the same result:

```
...WHERE last_name = 'Higgins'
```

The name in the output appears as it was stored in the database. To display the name in uppercase, use the `UPPER` function in the `SELECT` statement.

```
SELECT employee_id, UPPER(last_name), department_id
FROM   employees
WHERE  INITCAP(last_name) = 'Higgins
```

	EMPLOYEE_ID	UPPER(LAST_NAME)	DEPARTMENT_ID
1	205	HIGGINS	110

Character-Manipulation Functions

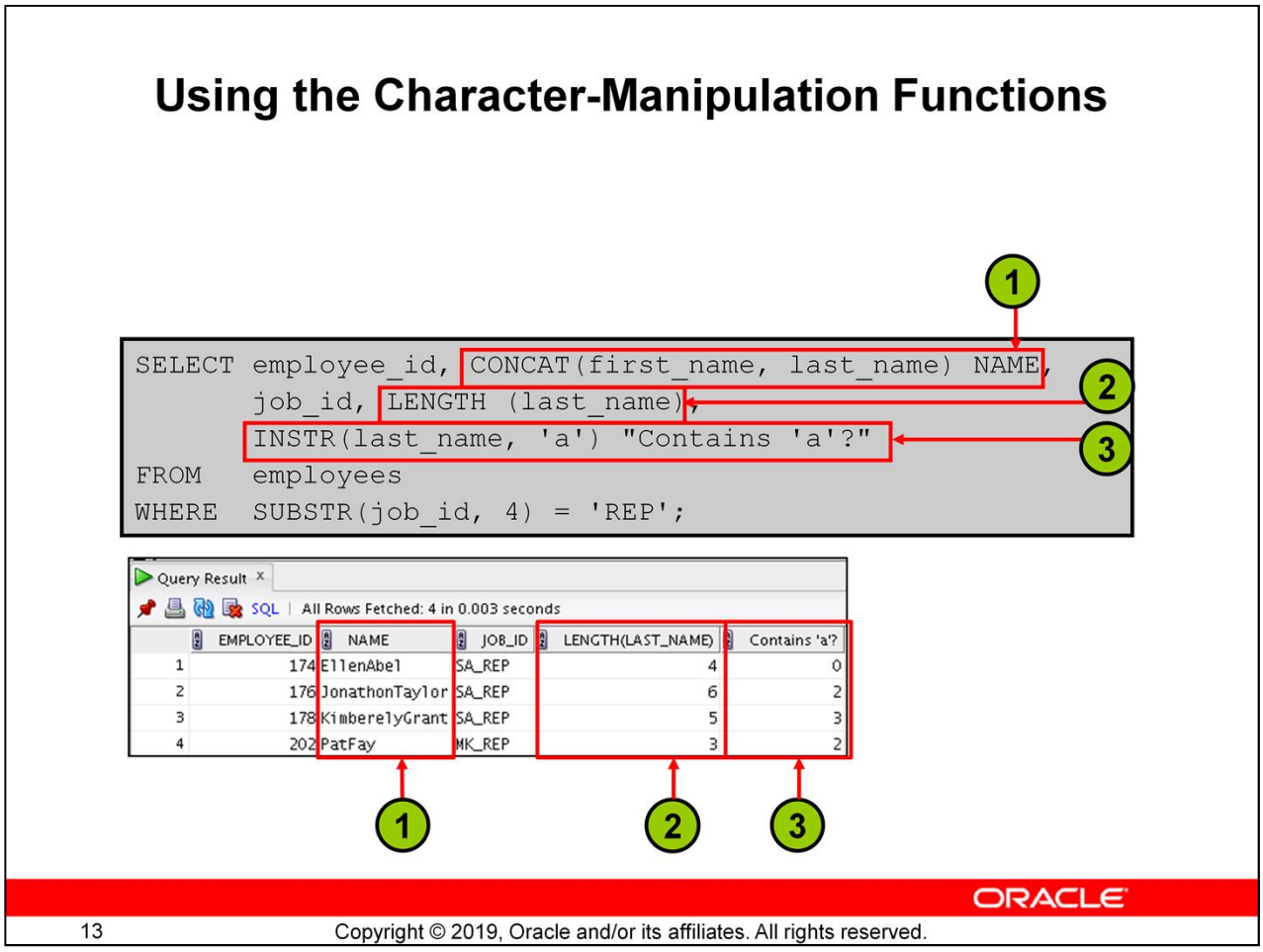
- These functions manipulate character strings:

- **CONCAT:** Joins values together (You are limited to using two parameters with `CONCAT`.)
- **SUBSTR:** Extracts a string of determined length
- **LENGTH:** Shows the length of a string as a numeric value
- **INSTR:** Finds the numeric position of a named character
- **LPAD:** Returns an expression left-padded to the length of *n* characters with a character expression
- **RPAD:** Returns an expression right-padded to the length of *n* characters with a character expression
- **TRIM:** Trims leading or trailing characters (or both) from a character string

Character-Manipulation Functions

These functions manipulate character strings:

Function	Result
`CONCAT('Hello', 'World')`	`HelloWorld`
`SUBSTR('HelloWorld',1,5)`	`Hello`
`LENGTH('HelloWorld')`	`10`
`INSTR('HelloWorld', 'W')`	`6`
`LPAD(salary,10,'*')`	`*****24000`
`RPAD(salary, 10, '*')`	`24000*****`
`REPLACE ('JACK and JUE','J','BL')`	`BLACK and BLUE`
`TRIM('H' FROM 'HelloWorld')`	`elloWorld`



The example in the slide displays employee first names and last names joined together, the length of the employee last name, and the numeric position of the letter “a” in the employee last name for all employees who have the string, `REP`, contained in the job `ID` starting at the fourth position of the job `ID`.

Example

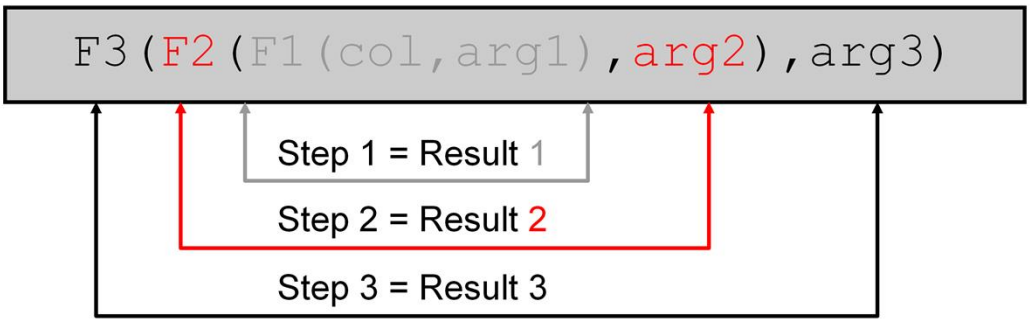
Modify the SQL statement in the slide to display the data for those employees whose last names end with the letter “n.”

```
SELECT employee_id, CONCAT(first_name, last_name) NAME,
LENGTH (last_name), INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(last_name, -1, 1) = 'n';
```

	EMPLOYEE_ID	NAME	LENGTH(LAST_NAME)	Contains 'a'?
1	102	LexDe Haan	7	5
2	200	JenniferWhalen	6	3
3	201	MichaelHartstein	9	2

Nesting Functions

- Single-row functions can be nested to any level.
- Nested functions are evaluated from the deepest level to the least deep level.



Single-row functions can be nested to any depth. Nested functions are evaluated from the innermost level to the outermost level. Some examples follow to show you the flexibility of these functions.

Nesting Functions: Example

```
SELECT last_name,
  UPPER(CONCAT(SUBSTR (LAST_NAME, 1, 8), '_US'))
FROM   employees
WHERE  department_id = 60;
```

	LAST_NAME	UPPER(CONCAT(SUBSTR(LAST_NAME,1,8),'_US'))
1	Hunold	HUNOLD_US
2	Ernst	ERNST_US
3	Lorentz	LORENTZ_US

The example in the slide displays the last names of employees in department 60. The evaluation of the SQL statement involves three steps:

1. The inner function retrieves the first eight characters of the last name.
   ```
   Result1 = SUBSTR (LAST_NAME, 1, 8)
   ```
2. The outer function concatenates the result with `_US`.
   ```
   Result2 = CONCAT(Result1, '_US')
   ```
3. The outermost function converts the results to uppercase.

The entire expression becomes the column heading because no column alias was given.

Example

Display the date of the next Friday that is six months from the hire date. The resulting date should appear as Friday, July 20th, 2001. Order the results by hire date.

Numeric Functions

- ROUND: Rounds value to a specified decimal
- TRUNC: Truncates value to a specified decimal
- MOD: Returns remainder of division

Function	Result
ROUND(45.926, 2)	45.93
TRUNC(45.926, 2)	45.92
MOD(1600, 300)	100

ORACLE

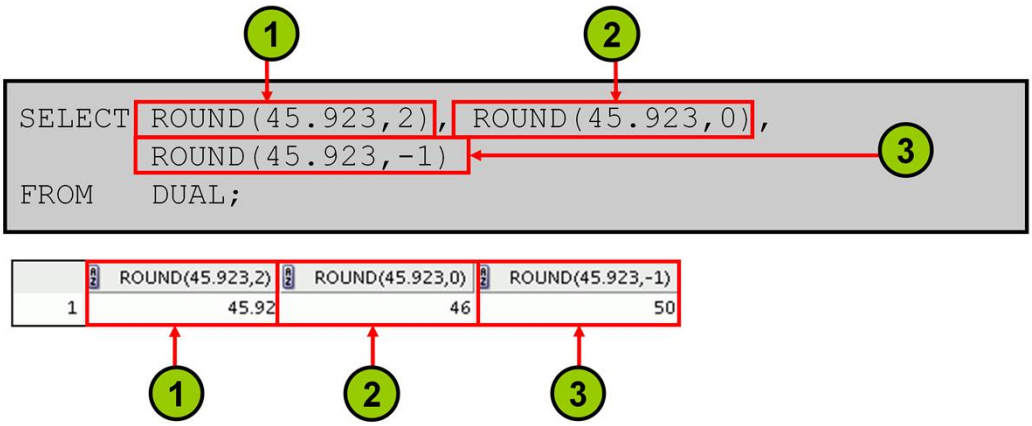
Numeric functions accept numeric input and return numeric values. This section describes some of the numeric functions.

Function	**Purpose**
ROUND(*column*|*expression*, *n*)	Rounds the column, expression, or value to *n* decimal places or, if *n* is omitted, no decimal places (If *n* is negative, numbers to the left of decimal point are rounded.
TRUNC(*column*|*expression*, *n*)	Truncates the column, expression, or value to *n* decimal places or, if *n* is omitted, *n* defaults to zero
MOD(*m*,*n*)	Returns the remainder of *m* divided by *n*

Note: This list contains only some of the available numeric functions.

For more information, see the “Numeric Functions” section in *Oracle Database SQL Language Reference* for 19c database.

Using the ROUND Function



DUAL is a public table that you can use to view results from functions and calculations.

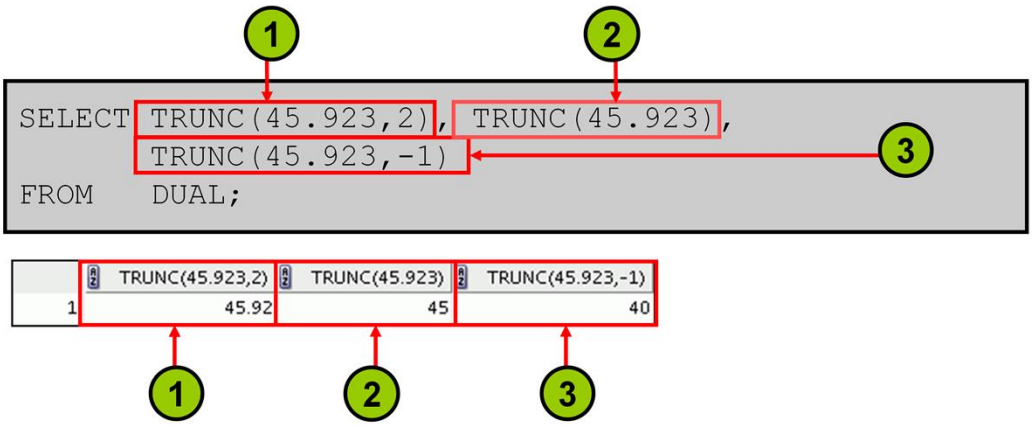
The ROUND function rounds the column, expression, or value to *n* decimal places. If the second argument is 0 or is missing, the value is rounded to zero decimal places. If the second argument is 2, the value is rounded to two decimal places. Conversely, if the second argument is –2, the value is rounded to two decimal places to the left (rounded to the nearest unit of 100).

The ROUND function can also be used with date functions. You will see examples later in this lesson.

DUAL Table

The DUAL table is owned by the user SYS and can be accessed by all users. It contains one column, DUMMY, and one row with the value X. The DUAL table is useful when you want to return a value only once (for example, the value of a constant, pseudocolumn, or expression that is not derived from a table with user data). The DUAL table is generally used for completeness of the SELECT clause syntax, because both SELECT and FROM clauses are mandatory, and several calculations do not need to select from the actual tables.

Using the TRUNC Function



The TRUNC function truncates the column, expression, or value to *n* decimal places.

The TRUNC function works with arguments similar to those of the ROUND function. If the second argument is 0 or is missing, the value is truncated to zero decimal places. If the second argument is 2, the value is truncated to two decimal places. Conversely, if the second argument is –2, the value is truncated to two decimal places to the left. If the second argument is –1, the value is truncated to one decimal place to the left.

Like the ROUND function, the TRUNC function can be used with date functions.

Using the MOD Function

For all employees with the job title of Sales Representative, calculate the remainder of the salary after it is divided by 5,000.

```
SELECT last_name, salary, MOD(salary, 5000)
FROM   employees
WHERE  job_id = 'SA_REP';
```

	LAST_NAME	SALARY	MOD(SALARY,5000)
1	Abel	11000	1000
2	Taylor	8600	3600
3	Grant	7000	2000

The `MOD` function finds the remainder of the first argument divided by the second argument. The slide example calculates the remainder of the salary after dividing it by 5,000 for all employees whose job `ID` is `SA_REP`.

Note: The `MOD` function is often used to determine whether a value is odd or even. The `MOD` function is also the Oracle hash function.

Working with Dates

- The Oracle Database stores dates in an internal numeric format: century, year, month, day, hours, minutes, and seconds.
- The default date display format is `DD-MON-RR`.
  - Enables you to store 21st-century dates in the 20th century by specifying only the last two digits of the year
  - Enables you to store 20th-century dates in the 21st century in the same way

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date < '01-FEB-08';
```

	LAST_NAME	HIRE_DATE
1	King	17-JUN-03
2	Kochhar	21-SEP-05

...

The Oracle Database stores dates in an internal numeric format, representing the century, year, month, day, hours, minutes, and seconds.

The default display and input format for any date is `DD-MON-RR`. Valid Oracle dates are between January 1, 4712 B.C., and December 31, 9999 A.D.

In the example in the slide, the `HIRE_DATE` column output is displayed in the default format `DD-MON-RR`. However, dates are not stored in the database in this format. All the components of the date and time are stored. So, although a `HIRE_DATE` such as 17-JUN-03 is displayed as day, month, and year, there is also *time* and *century* information associated with the date. The complete data might be June 17, 2003, 5:10:43 PM.

RR Date Format

Current Year	Specified Date	RR Format	YY Format
1995	27-OCT-95	1995	1995
1995	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2001	27-OCT-95	1995	2095

		If the specified two-digit year is:	
		0–49	50–99
If two digits of the current year are:	0–49	The return date is in the current century	The return date is in the century before the current one
	50–99	The return date is in the century after the current one	The return date is in the current century

ORACLE

The RR date format is similar to the YY element, but you can use it to specify different centuries. Use the RR date format element instead of YY so that the century of the return value varies according to the specified two-digit year and the last two digits of the current year. The table in the slide summarizes the behavior of the RR element.

Current Year	Given Date	Interpreted (RR)	Interpreted (YY)
1994	27-OCT-95	1995	1995
1994	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2048	27-OCT-52	1952	2052
2051	27-OCT-47	2147	2047

Note the values shown in the last two rows of the above table. As we approach the middle of the century, then the RR behavior is probably not what you want.

This data is stored internally as follows:

CENTURY	YEAR	MONTH	DAY	HOUR	MINUTE	SECOND
19	87	06	17	10	43	

Centuries and the Year 2000

When a record with a date column is inserted into a table, the *century* information is picked up from the SYSDATE function. However, when the date column is displayed on the screen, the century component is not displayed (by default).

The `DATE` data type uses 2 bytes for the year information, one for century and one for year. The century value is always included, whether or not it is specified or displayed. In this case, `RR` determines the default value for century on `INSERT`.

Using the SYSDATE Function

- SYSDATE is a date function that returns the current database server date and time.
- You can use SYSDATE just as you would use any other column name.
- For example, you can display the current date by selecting SYSDATE from a table.
- It is customary to select SYSDATE from a public table called DUAL.

```
SELECT sysdate
FROM   dual;
```

SYSDATE
1 24-AUG-12

ORACLE

Note: SYSDATE returns the current date and time set for the operating system on which the database resides. Therefore, if you are in a place in Australia and connected to a remote database in a location in the United States (U.S.), the sysdate function will return the U.S. date and time. In that case, you can use the CURRENT_DATE function that returns the current date in the session time zone.

The CURRENT_DATE function and other related time zone functions are discussed in detail in *Oracle Database: SQL Workshop II*.

Arithmetic with Dates

- Because the database stores dates as numbers, you can perform calculations using arithmetic operators such as addition and subtraction.
- Add to or subtract a number from a date for a resultant date value.
- Subtract two dates to find the number of days between those dates.
- Add hours to a date by dividing the number of hours by 24.

Because the database stores dates as numbers, you can perform calculations using arithmetic operators such as addition and subtraction. You can add and subtract number constants as well as dates.

You can perform the following operations:

Operation	Result	Description
date + number	Date	Adds a number of days to a date
date – number	Date	Subtracts a number of days from a date
date – date	Number of days	Subtracts one date from another
date + number/24	Date	Adds a number of hours to a date

Using Arithmetic Operators with Dates

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS
FROM   employees
WHERE  department_id = 90;
```

	LAST_NAME	WEEKS
1	King	478.871917989417989417989417989417989418
2	Kochhar	360.729060846560846560846560846560846561
3	De Haan	605.300489417989417989417989417989417989

The example in the slide displays the last name and the number of weeks employed for all employees in department 90. It subtracts the date on which the employee was hired from the current date (`SYSDATE`) and divides the result by 7 to calculate the number of weeks that a worker has been employed.

Note: `SYSDATE` is a SQL function that returns the current date and time. Your results may differ depending on the date and time set for the operating system of your local database when you run the SQL query.

If a more current date is subtracted from an older date, the difference is a negative number.

Date-Manipulation Functions

Function	Result
MONTHS_BETWEEN	Number of months between two dates
ADD_MONTHS	Add calendar months to date
NEXT_DAY	Week day of the date specified
LAST_DAY	Last day of the month
ROUND	Round date
TRUNC	Truncate date

Date functions operate on Oracle dates. All date functions return a value of the DATE data type except MONTHS_BETWEEN, which returns a numeric value.

- **MONTHS_BETWEEN(*date1, date2*):** Finds the number of months between *date1* and *date2*. The result can be positive or negative. If *date1* is later than *date2*, the result is positive; if *date1* is earlier than *date2*, the result is negative. The noninteger part of the result represents a portion of the month.
- ADD_MONTHS(*date, n*): Adds *n* number of calendar months to *date*. The value of *n* must be an integer and can be negative.
- **NEXT_DAY(*date, 'char'*):** Finds the date of the next specified day of the week ('*char*') following *date*. The value of *char* may be a number representing a day or a character string.
- **LAST_DAY(*date*):** Finds the date of the last day of the month that contains *date*

The above list is a subset of the available date functions. ROUND and TRUNC number functions can also be used to manipulate the date values as shown below:

- **ROUND(*date*[,'*fmt*']):** Returns *date* rounded to the unit that is specified by the format model *fmt*. If the format model *fmt* is omitted, *date* is rounded to the nearest day.
- **TRUNC(*date*[, '*fmt*']):** Returns *date* with the time portion of the day truncated to the unit that is specified by the format model *fmt*. If the format model *fmt* is omitted, *date* is truncated to the nearest day.

The format models are covered in detail in the lesson titled “Using Conversion Functions and Conditional Expressions.”

Using Date Functions

Function	Result
`MONTHS_BETWEEN ('01-SEP-95','11-JAN-94')`	`19.6774194`
`ADD_MONTHS ('31-JAN-96',1)`	`'29-FEB-96'`
`NEXT_DAY ('01-SEP-95','FRIDAY')`	`'08-SEP-95'`
`LAST_DAY ('01-FEB-95')`	`'28-FEB-95'`

ORACLE

In the example in the slide, the `ADD_MONTHS` function adds one month to the supplied date value “31-JAN-96” and returns “29-FEB-96.” The function recognizes the year 1996 as the leap year and, therefore, returns the last day of the February month. If you change the input date value to “31-JAN-95,” the function returns “28-FEB-95.”

For example, display the employee number, hire date, number of months employed, six-month review date, first Friday after hire date, and the last day of the hire month for all employees who have been employed for fewer than 150 months.

```
SELECT employee_id, hire_date, MONTHS_BETWEEN (SYSDATE, hire_date)
TENURE, ADD_MONTHS (hire_date, 6) REVIEW,  NEXT_DAY (hire_date,
'FRIDAY'), LAST_DAY(hire_date)
FROM employees WHERE  MONTHS_BETWEEN (SYSDATE, hire_date) < 150;
```

Query Result

SQL | All Rows Fetched: 20 in 0.016 seconds

	EMPLOYEE_ID	HIRE_DATE	TENURE	REVIEW	NEXT_DAY(HIRE_DATE,'FRIDAY')	LAST_DAY(HIRE_DATE)
1	100	17-JUN-03	110.100155316606929510155316606929510155	17-DEC-03	20-JUN-03	30-JUN-03
2	101	21-SEP-05	82.971123058542413381123058542413381123 06	21-MAR-06	23-SEP-05	30-SEP-05
3	102	13-JAN-01	139.229187574671445639187574671445639188	13-JUL-01	19-JAN-01	31-JAN-01
4	103	03-JAN-06	79.551768219832735961768219832735961768 22	03-JUL-06	06-JAN-06	31-JAN-06
5	104	21-MAY-07	62.971123058542413381123058542413381123 06	21-NOV-07	25-MAY-07	31-MAY-07
6	107	07-FEB-07	66.422735961768219832735961768219832735 96	07-AUG-07	09-FEB-07	28-FEB-07
7	124	16-NOV-07	57.132413381123058542413381123058542413 38	16-MAY-08	23-NOV-07	30-NOV-07

...

Using ROUND and TRUNC Functions with Dates

Assume SYSDATE = '25-JUL-03':

Function	Result
ROUND(SYSDATE,'MONTH')	01-AUG-03
ROUND(SYSDATE ,'YEAR')	01-JAN-04
TRUNC(SYSDATE ,'MONTH')	01-JUL-03
TRUNC(SYSDATE ,'YEAR')	01-JAN-03

The ROUND and TRUNC functions can be used for number and date values. When used with dates, these functions round or truncate to the specified format model. Therefore, you can round dates to the nearest year or month. If the format model is month, dates 1-15 result in the first day of the current month. Dates 16-31 result in the first day of the next month. If the format model is year, months 1-6 result in January 1 of the current year. Months 7-12 result in January 1 of the next year.

Example

Compare the hire dates for all employees who started in 2004. Display the employee number, hire date, and starting month using the ROUND and TRUNC functions.

```
SELECT employee_id, hire_date,
 ROUND(hire_date, 'MONTH'), TRUNC(hire_date, 'MONTH')
FROM   employees
WHERE  hire_date LIKE '%04
```

	EMPLOYEE_ID	HIRE_DATE	ROUND(HIRE_DATE,'MONTH')	TRUNC(HIRE_DATE,'MONTH')
1	174	11-MAY-04	01-MAY-04	01-MAY-04
2	201	17-FEB-04	01-MAR-04	01-FEB-04

Quiz

Which four of the following statements are true about single-row functions?

a. Manipulate data items
b. Accept arguments and return one value per argument
c. Act on each row that is returned
d. Return one result per set of rows
e. May not modify the data type
f. Can be nested
g. Accept arguments that can be a column or an expression

Answer: a, c, f, g

Summary

In this lesson, you should have learned how to:

- Perform calculations on data using functions
- Modify individual data items using functions

Single-row functions can be nested to any level. Single-row functions can manipulate the following:

- **Character data:** `LOWER`, `UPPER`, `INITCAP`, `CONCAT`, `SUBSTR`, `INSTR`, `LENGTH`
- **Number data:** `ROUND`, `TRUNC`, `MOD`
- **Date values:** `SYSDATE`, `MONTHS_BETWEEN`, `ADD_MONTHS`, `NEXT_DAY`, `LAST_DAY`

Remember the following:

- Date values can also use arithmetic operators.
- `ROUND` and `TRUNC` functions can also be used with date values.

`SYSDATE` and `DUAL`

`SYSDATE` is a date function that returns the current date and time. It is customary to select `SYSDATE` from a single-row public table called `DUAL`