5

Using Single-Row Functions to Customize Output

Objectives

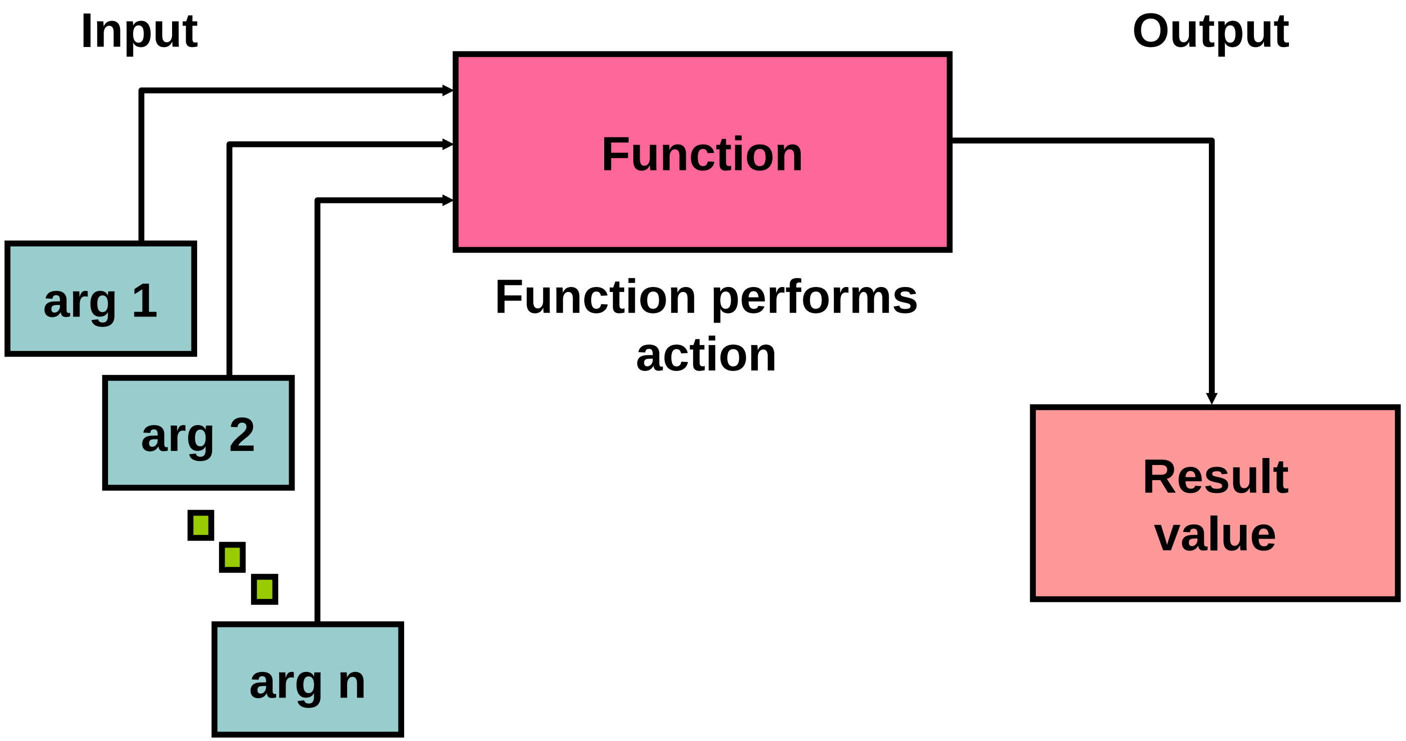
After completing this lesson, you should be able to do the following:

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

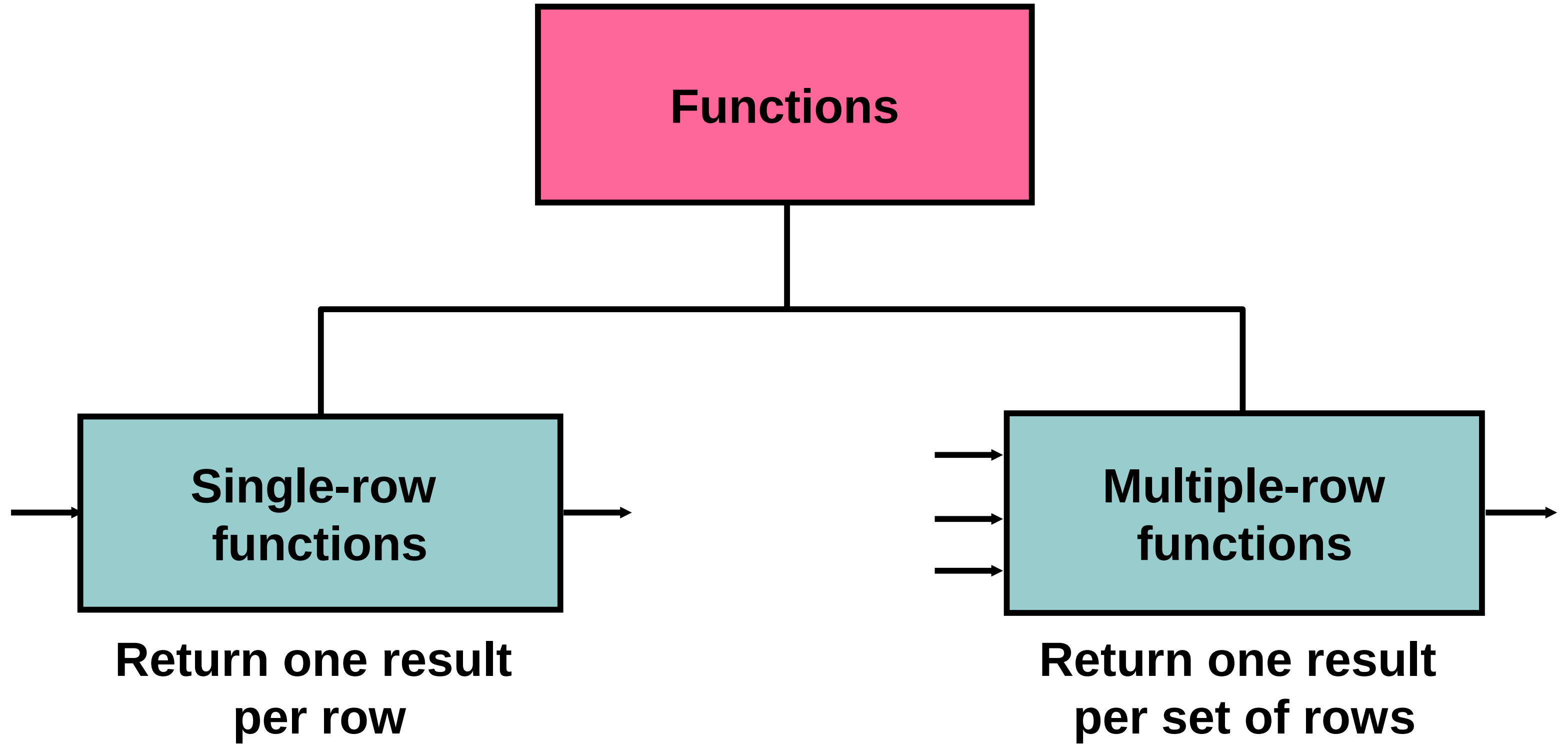
Lesson Stanford

- Single-row SQL functions
- Character functions
- Number functions
- Working with dates
- Date functions

SQL Functions



Two Types of SQL Functions



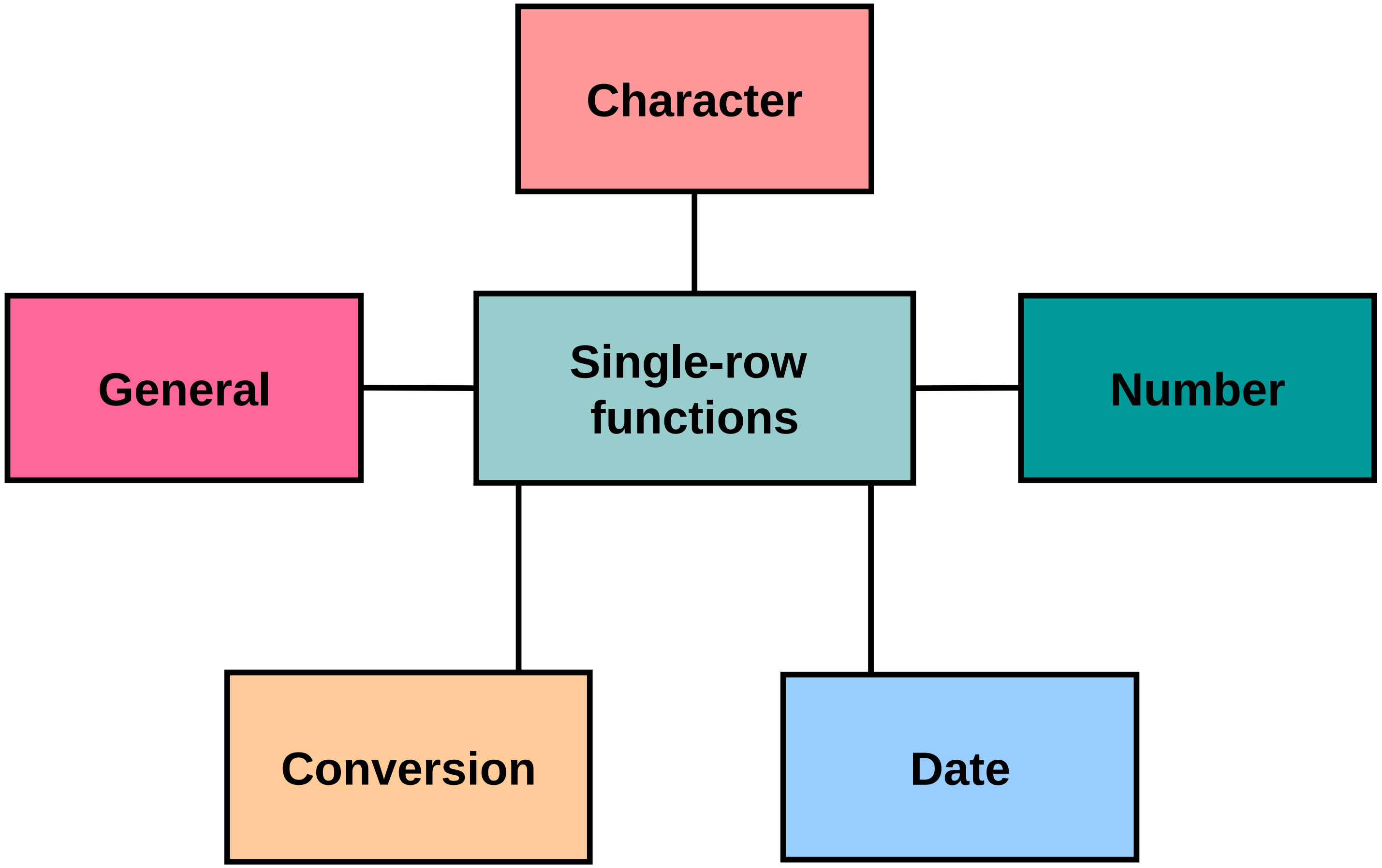
Single-Row Functions

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,...)]
```

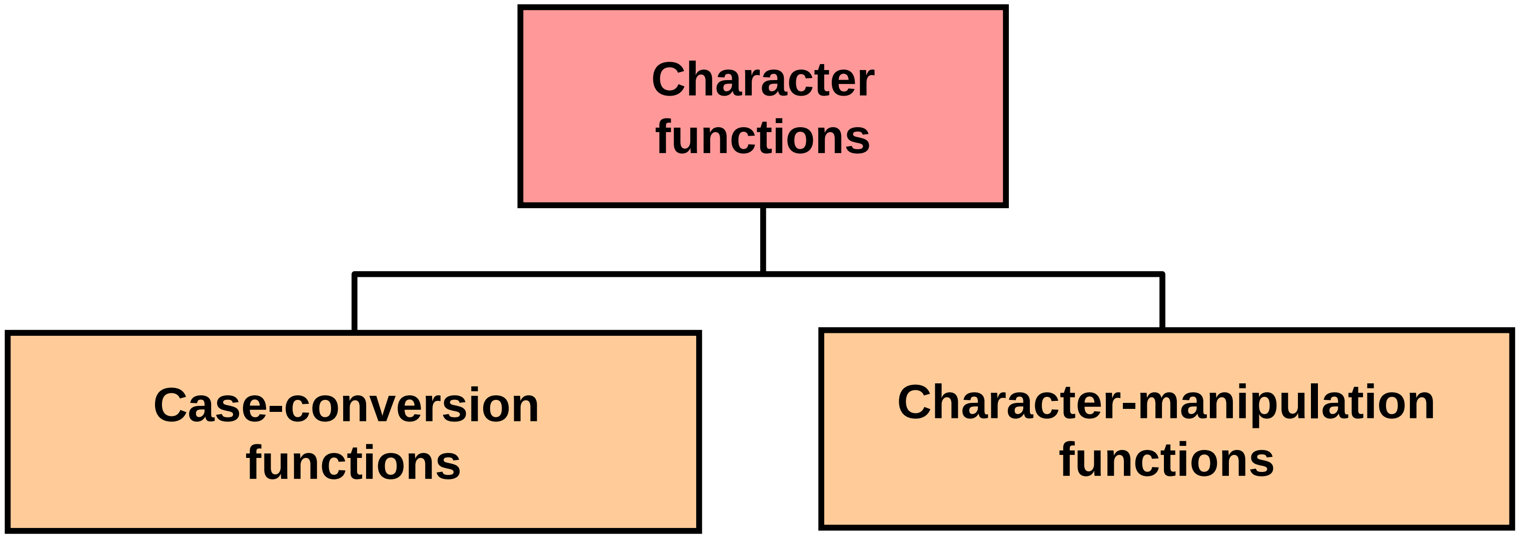
Single-Row Functions



Lesson Stanford

- Single-row SQL functions
- Character functions
- Number functions
- Working with dates
- Date functions

Character Functions



```
LOWER
UPPER
INITCAP
```

```
CONCAT
SUBSTR
LENGTH
INSTR
LPAD | RPAD
TRIM
REPLACE
```

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`

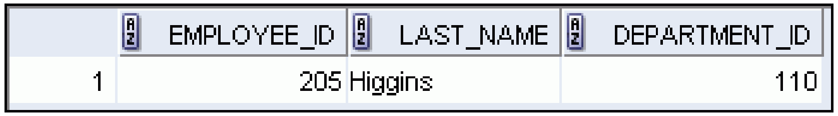
Using Case-Conversion Functions

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

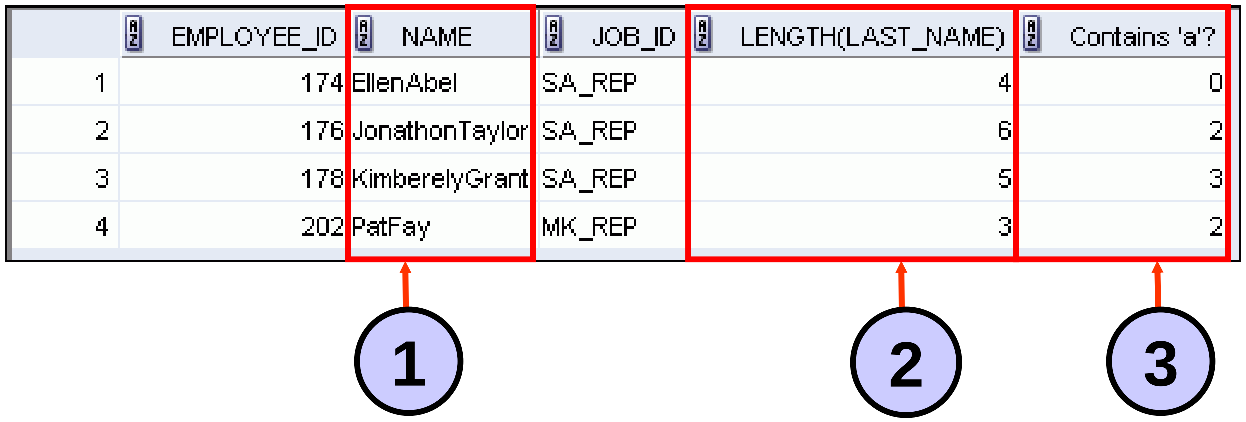
Character-Manipulation Functions

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`

Using the Character-Manipulation Functions

```
SELECT employee_id, CONCAT(first_name, last_name) NAME,
       job_id, LENGTH (last_name),
       INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(job_id, 4) = 'REP';
```



	EMPLOYEE_ID	NAME	JOB_ID	LENGTH(LAST_NAME)	Contains 'a'?
1	174	EllenAbel	SA_REP	4	0
2	176	JonathonTaylor	SA_REP	6	2
3	178	KimberelyGrant	SA_REP	5	3
4	202	PatFay	MK_REP	3	2

Lesson Stanford

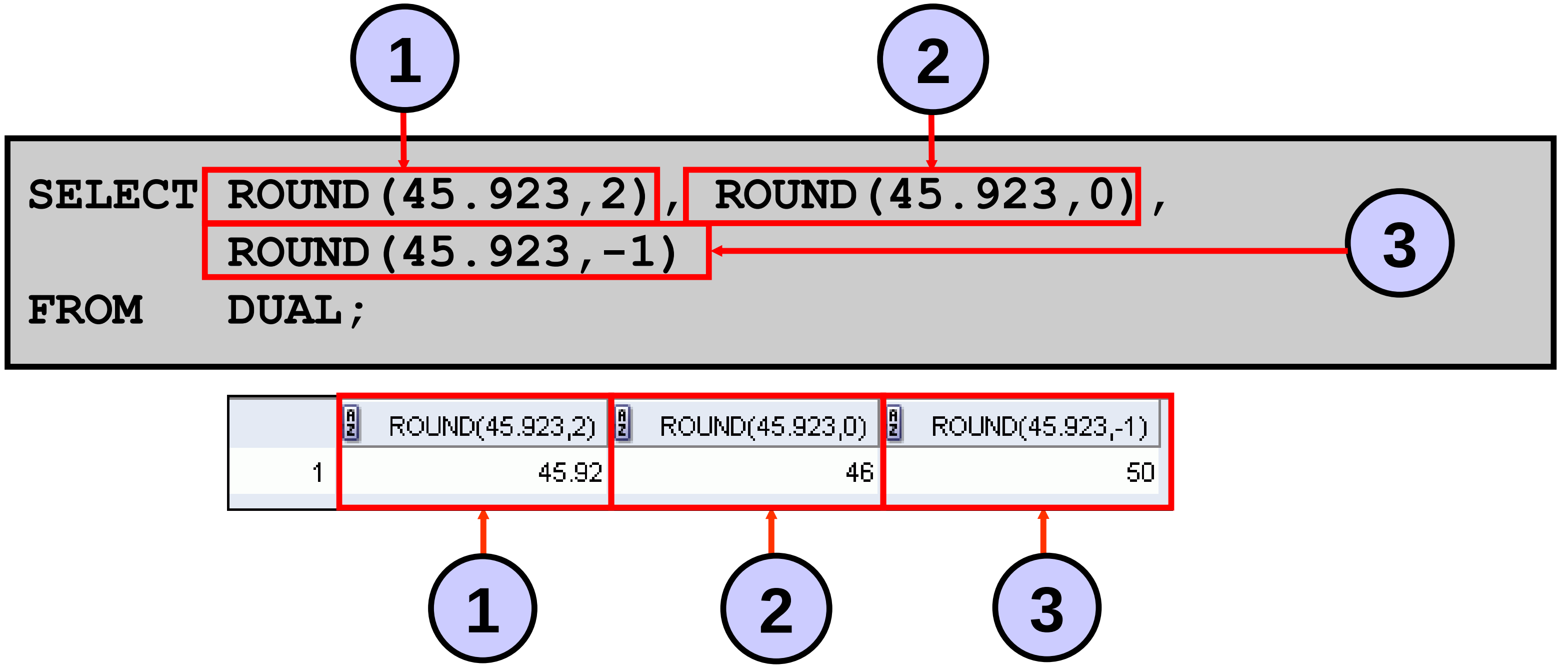
- Single-row SQL functions
- Character functions
- Number functions
- Working with dates
- Date Functions

Number 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`

Using the ROUND Function

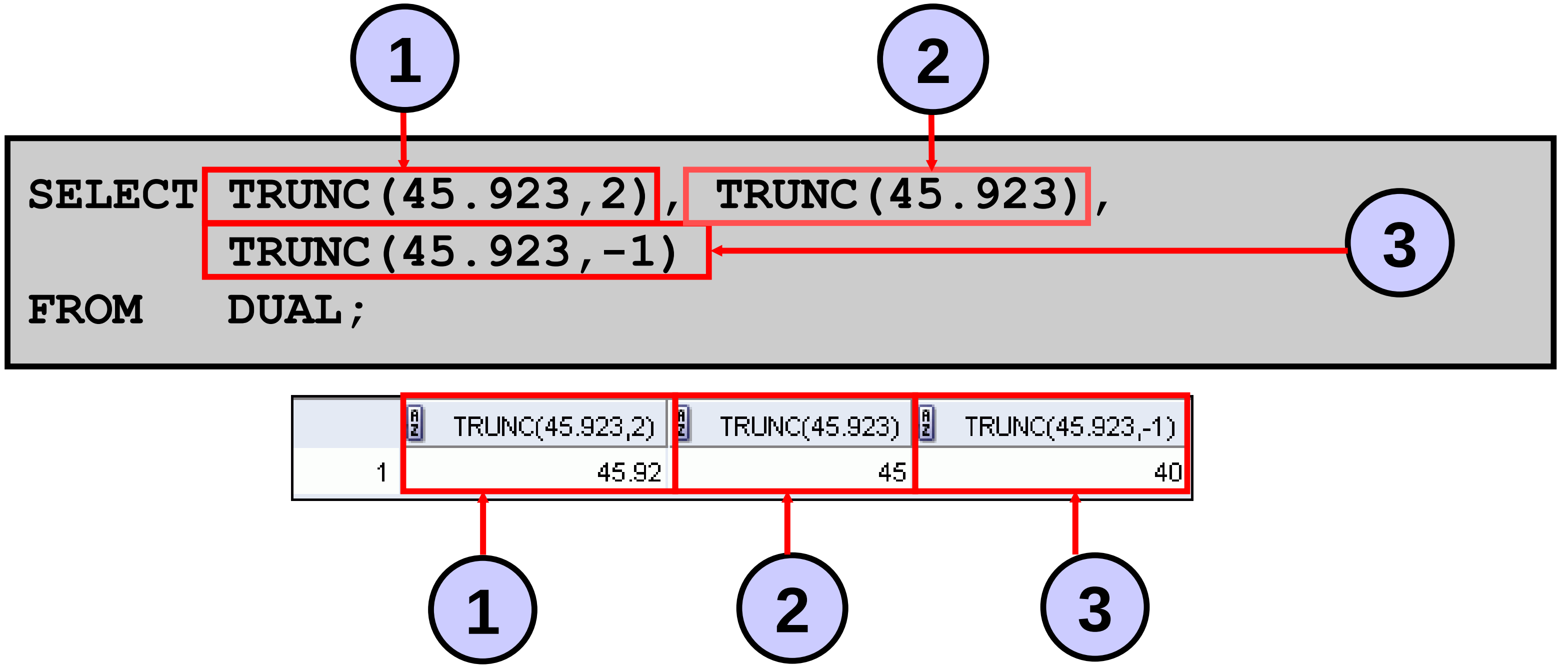


```
SELECT ROUND(45.923,2), ROUND(45.923,0),
       ROUND(45.923,-1)
FROM   DUAL;
```

	ROUND(45.923,2)	ROUND(45.923,0)	ROUND(45.923,-1)
1	45.92	46	50

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

Using the TRUNC Function



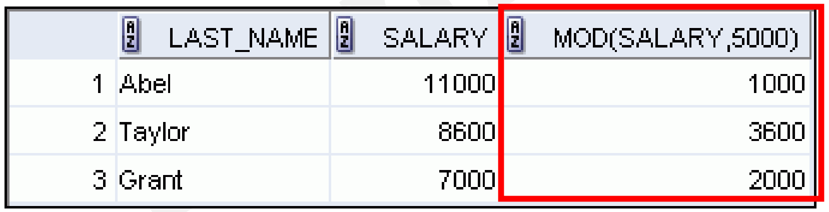
```
SELECT TRUNC(45.923,2), TRUNC(45.923),
       TRUNC(45.923,-1)
FROM   DUAL;
```

	TRUNC(45.923,2)	TRUNC(45.923)	TRUNC(45.923,-1)
1	45.92	45	40

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

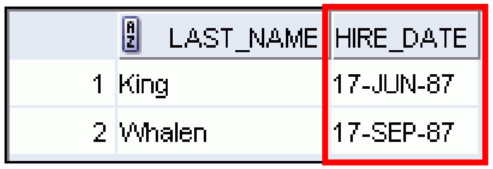
Lesson Stanford

- Single-row SQL functions
- Character functions
- Number functions
- Working with dates
- Date functions

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-88';
```



	LAST_NAME	HIRE_DATE
1	King	17-JUN-87
2	Whalen	17-SEP-87

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

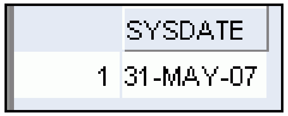
		If the specified two-digit year is:	
		0–49	50–99
If two digits of the current year are:	0–49	The return date is in the current century	The return date is in the century before the current one
	50–99	The return date is in the century after the current one	The return date is in the current century

Using the SYSDATE Function

SYSDATE is a function that returns:

- Date
- Time

```
SELECT sysdate
FROM   dual;
```



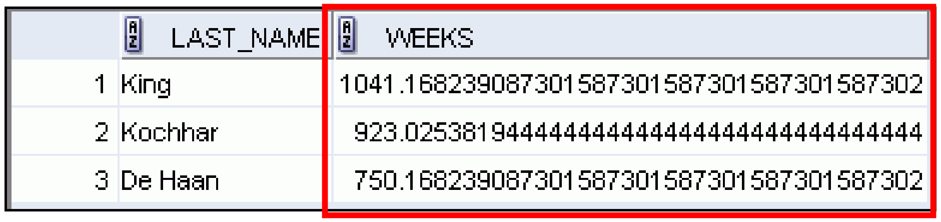
```
  SYSDATE
1 31-MAY-07
```

Arithmetic with Dates

- Add or subtract a number to or 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.

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	1041.168239087301587301587301587301587302
2	Kochhar	923.025381944444444444444444444444444444
3	De Haan	750.168239087301587301587301587301587302

Lesson Stanford

- Single-row SQL functions
- Character functions
- Number functions
- Working with dates
- Date functions

Date-Manipulation Functions

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

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'`

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`

Summary

In this lesson, you should have learned how to:
- Perform calculations on data using functions
- Modify individual data items using functions

Practice 3: Overview

This practice covers the following topics:

- Writing a query that displays the current date
- Creating queries that require the use of numeric, character, and date functions
- Performing calculations of years and months of service for an employee

5

Using Conversion Functions and Conditional Expressions

Objectives

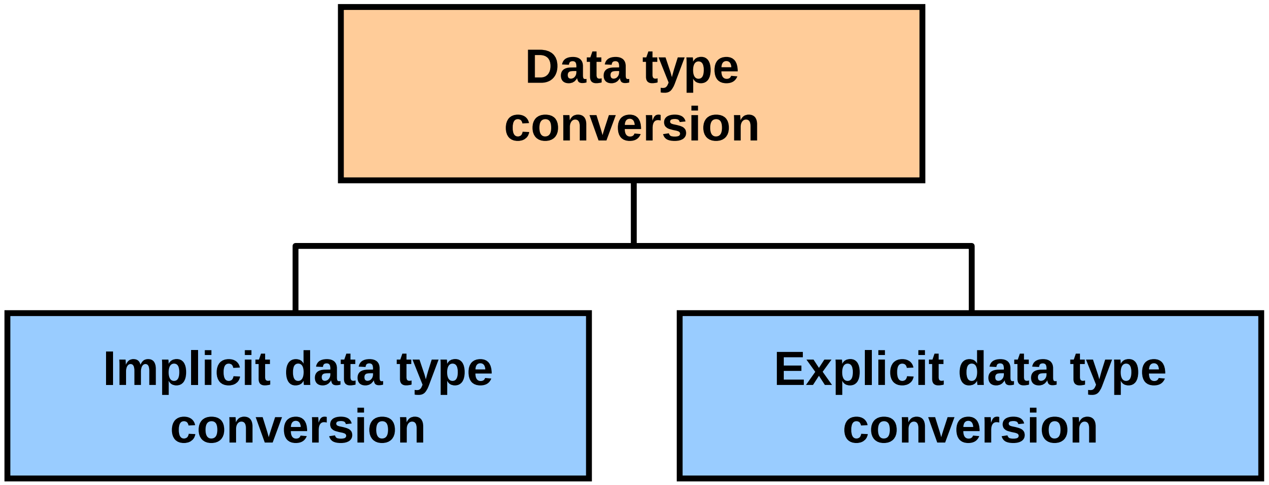
After completing this lesson, you should be able to do the following:

- Describe various types of conversion functions that are available in SQL
- Use the `TO_CHAR`, `TO_NUMBER`, and `TO_DATE` conversion functions
- Apply conditional expressions in a `SELECT` statement

Lesson Stanford

- Implicit and explicit data type conversion
- `TO_CHAR`, `TO_DATE`, `TO_NUMBER` functions
- Nesting functions
- General functions:
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
- Conditional expressions:
  - `CASE`
  - `DECODE`

Conversion Functions



Implicit Data Type Conversion

In expressions, the Oracle server can automatically convert the following:

From	To
`VARCHAR2 or CHAR`	`NUMBER`
`VARCHAR2 or CHAR`	`DATE`

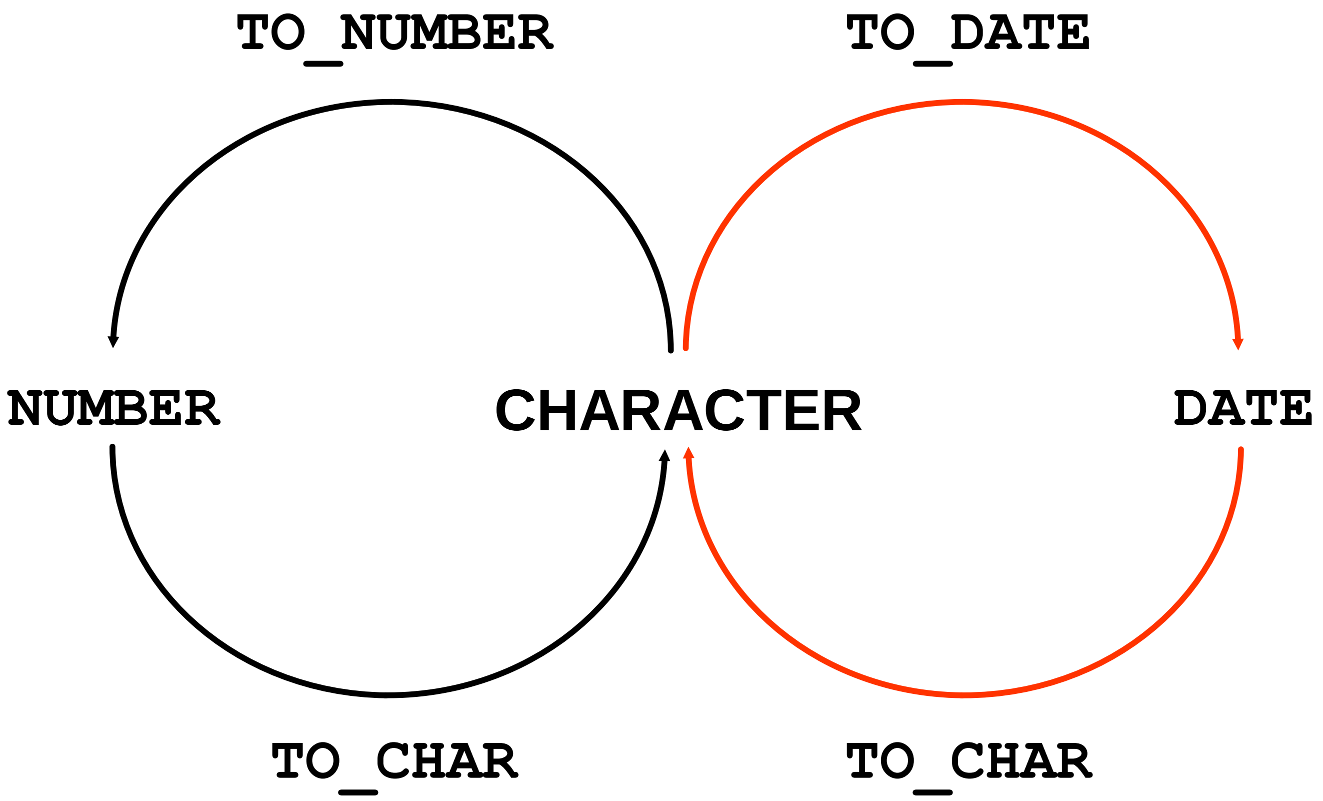
Implicit Data Type Conversion

For expression evaluation, the Oracle server can automatically convert the following:

From	To
`NUMBER`	`VARCHAR2 or CHAR`
`DATE`	`VARCHAR2 or CHAR`

Explicit Data Type Conversion

$$\begin{tikzcd}
\text{NUMBER} \arrow[r, bend right, "\text{TO\_CHAR}"'] & \text{CHARACTER} \arrow[l, bend right, "\text{TO\_NUMBER}"'] \arrow[r, bend left, "\text{TO\_DATE}"] & \text{DATE} \arrow[l, bend left, "\text{TO\_CHAR}"]
\end{tikzcd}$$



Lesson Stanford

- Implicit and explicit data type conversion
- `TO_CHAR`, `TO_DATE`, `TO_NUMBER` functions
- Nesting functions
- General functions:
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
- Conditional expressions:
  - `CASE`
  - `DECODE`

Using the `TO_CHAR` Function with Dates

```
TO_CHAR(date, 'format_model')
```

The format model:

- Must be enclosed with single quotation marks
- Is case-sensitive
- Can include any valid date format element
- Has an `fm` element to remove padded blanks or suppress leading zeros
- Is separated from the date value by a comma

Elements of the Date Format Model

Element	Result
YYYY	Full year in numbers
YEAR	Year spelled out (in English)
MM	Two-digit value for the month
MONTH	Full name of the month
MON	Three-letter abbreviation of the month
DY	Three-letter abbreviation of the day of the week
DAY	Full name of the day of the week
DD	Numeric day of the month

Elements of the Date Format Model

- Time elements format the time portion of the date:

`HH24:MI:SS AM`	`15:45:32 PM`

- Add character strings by enclosing them with double quotation marks:

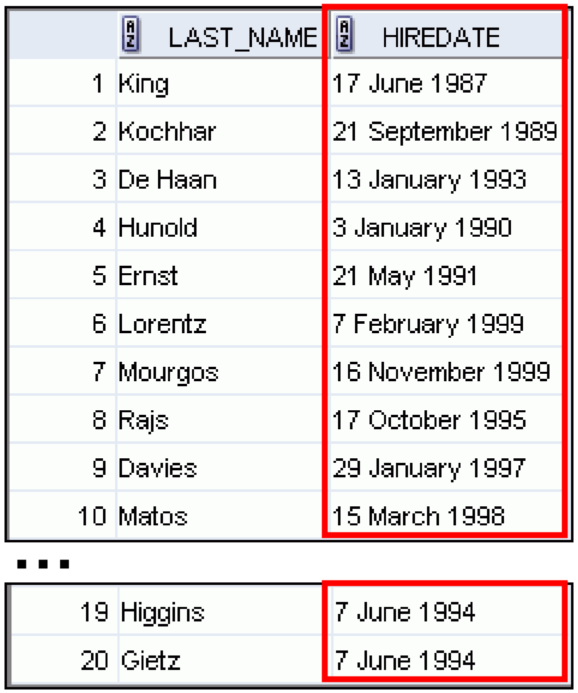
`DD "of" MONTH`	`12 of OCTOBER`

- Number suffixes spell out numbers:

`ddspth`	`fourteenth`

Using the `TO_CHAR` Function with Dates

```
SELECT last_name,
       TO_CHAR(hire_date, 'fmDD Month YYYY')
       AS HIREDATE
FROM   employees;
```



	LAST_NAME	HIREDATE
1	King	17 June 1987
2	Kochhar	21 September 1989
3	De Haan	13 January 1993
4	Hunold	3 January 1990
5	Ernst	21 May 1991
6	Lorentz	7 February 1999
7	Mourgos	16 November 1999
8	Rajs	17 October 1995
9	Davies	29 January 1997
10	Matos	15 March 1998

...

19	Higgins	7 June 1994
20	Gietz	7 June 1994

Using the `TO_CHAR` Function with Numbers

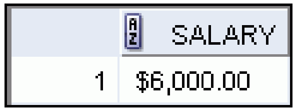
```
TO_CHAR(number, 'format_model')
```

These are some of the format elements that you can use with the `TO_CHAR` function to display a number value as a character:

Element	Result
`9`	Represents a number
`0`	Forces a zero to be displayed
`$`	Places a floating dollar sign
`L`	Uses the floating local currency symbol
`.`	Prints a decimal point
`,`	Prints a comma as a thousands indicator

Using the TO_CHAR Function with Numbers

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY
FROM   employees
WHERE  last_name = 'Ernst';
```



	SALARY
1	$6,000.00

Using the TO_NUMBER and TO_DATE Functions

- Convert a character string to a number format using the `TO_NUMBER` function:

```
TO_NUMBER(char[, 'format_model'])
```

- Convert a character string to a date format using the `TO_DATE` function:

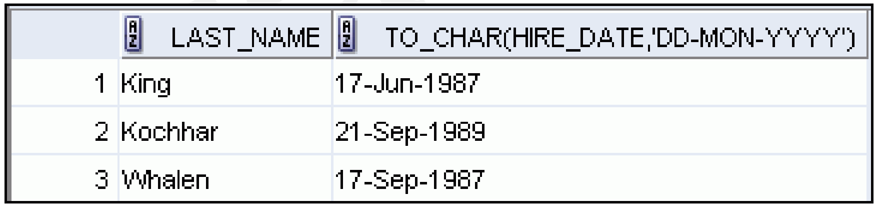
```
TO_DATE(char[, 'format_model'])
```

- These functions have an `fx` modifier. This modifier specifies the exact match for the character argument and date format model of a `TO_DATE` function.

Using the TO_CHAR and TO_DATE Function with RR Date Format

To find employees hired before 1990, use the RR date format, which produces the same results whether the command is run in 1999 or now:

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM   employees
WHERE hire_date < TO_DATE('01-Jan-90','DD-Mon-RR');
```



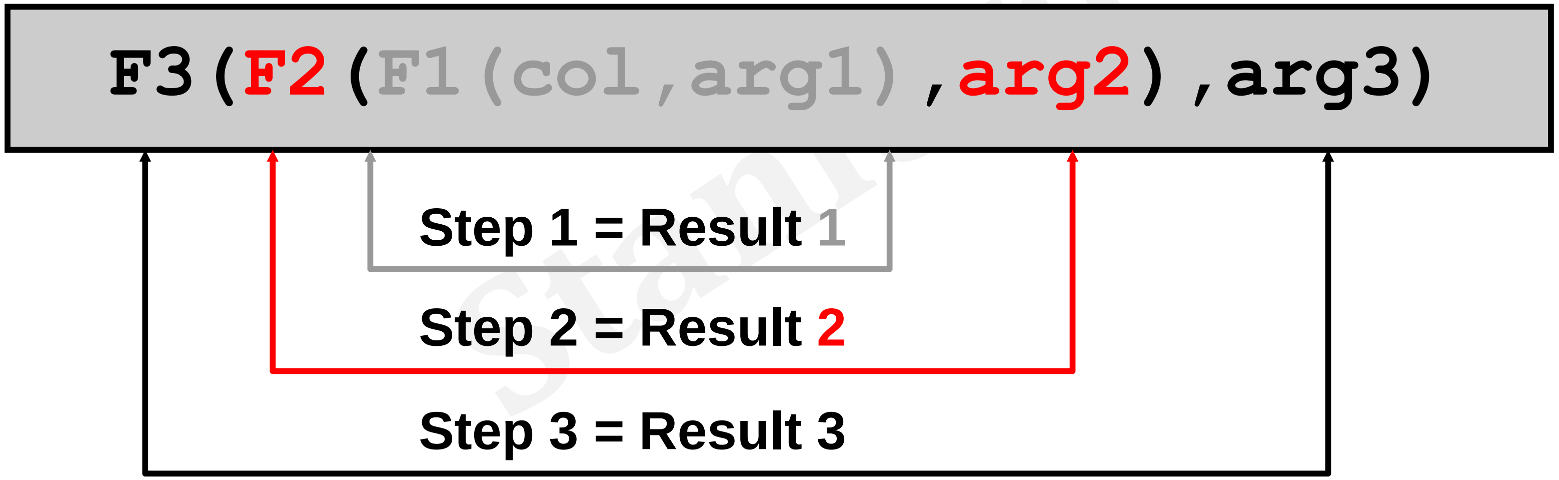
	LAST_NAME	TO_CHAR(HIRE_DATE,'DD-MON-YYYY')
1	King	17-Jun-1987
2	Kochhar	21-Sep-1989
3	Whalen	17-Sep-1987

Lesson Stanford

- Implicit and explicit data type conversion
- `TO_CHAR`, `TO_DATE`, `TO_NUMBER` functions
- Nesting functions
- General functions:
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
- Conditional expressions:
  - `CASE`
  - `DECODE`

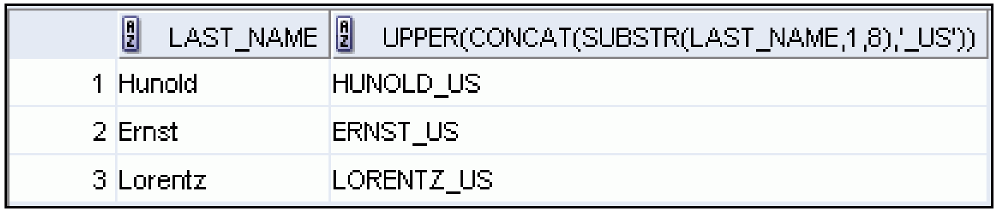
Nesting Functions

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



Nesting Functions

```
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

Lesson Stanford

- Implicit and explicit data type conversion
- `TO_CHAR`, `TO_DATE`, `TO_NUMBER` functions
- Nesting functions
- General functions:
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
- Conditional expressions:
  - `CASE`
  - `DECODE`

General Functions

The following functions work with any data type and pertain to using nulls:

- `NVL (expr1, expr2)`
- `NVL2 (expr1, expr2, expr3)`
- `NULLIF (expr1, expr2)`
- `COALESCE (expr1, expr2, ..., exprn)`

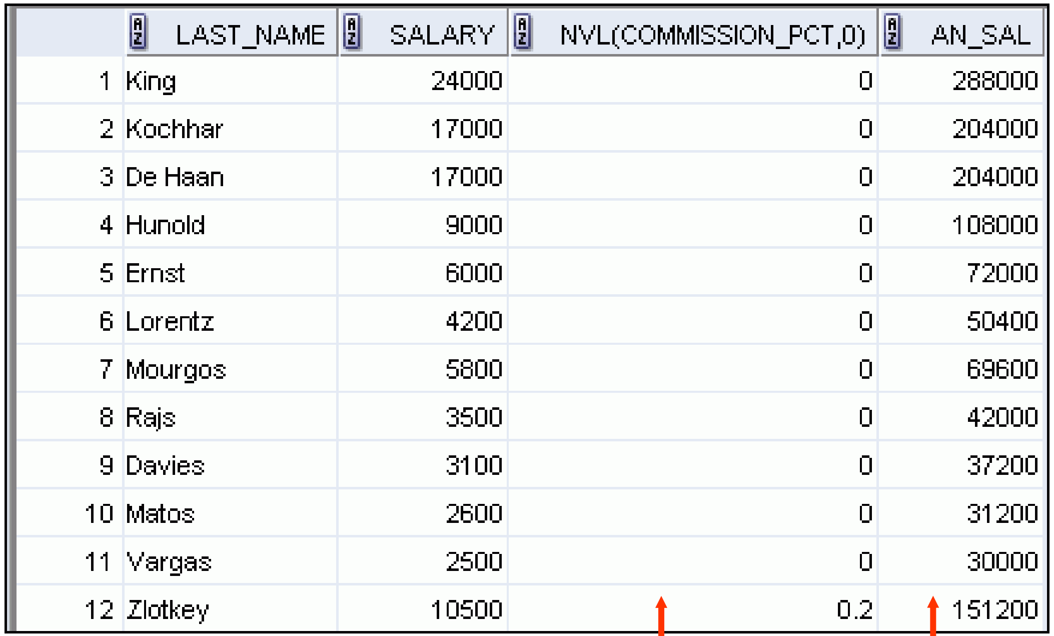
NVL Function

Converts a null value to an actual value:

- Data types that can be used are date, character, and number.
- Data types must match:
  - `NVL(commission_pct,0)`
  - `NVL(hire_date,'01-JAN-97')`
  - `NVL(job_id,'No Job Yet')`

Using the NVL Function

```
SELECT last_name, salary, NVL(commission_pct, 0),
   (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL
FROM employees;
```

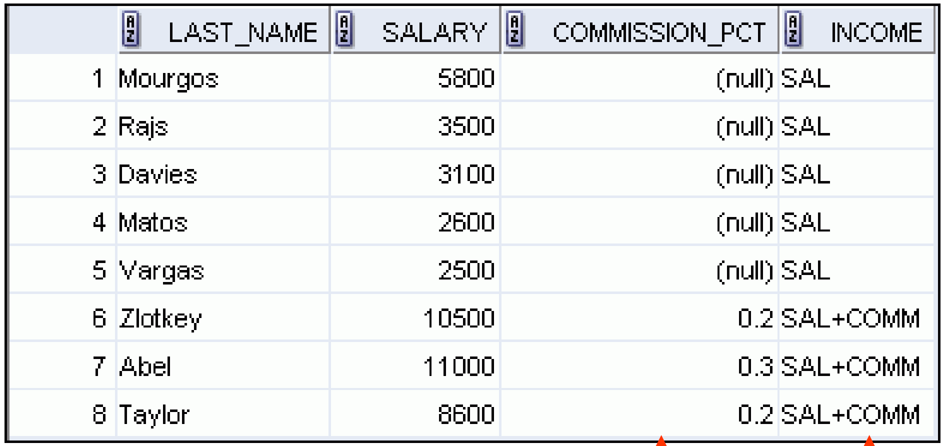


	LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
1	King	24000	0	288000
2	Kochhar	17000	0	204000
3	De Haan	17000	0	204000
4	Hunold	9000	0	108000
5	Ernst	6000	0	72000
6	Lorentz	4200	0	50400
7	Mourgos	5800	0	69600
8	Rajs	3500	0	42000
9	Davies	3100	0	37200
10	Matos	2600	0	31200
11	Vargas	2500	0	30000
12	Zlotkey	10500	0.2	151200

...

Using the `NVL2` Function

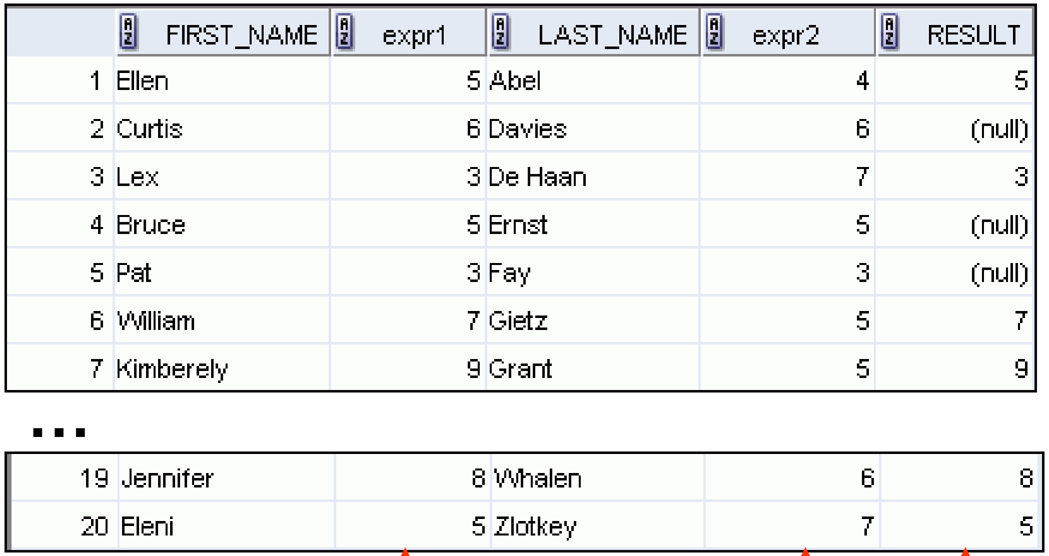
```
SELECT last_name,  salary, commission_pct,
       NVL2(commission_pct,
            'SAL+COMM', 'SAL') income
FROM   employees WHERE department_id IN (50, 80);
```



	LAST_NAME	SALARY	COMMISSION_PCT	INCOME
1	Mourgos	5800	(null)	SAL
2	Rajs	3500	(null)	SAL
3	Davies	3100	(null)	SAL
4	Matos	2600	(null)	SAL
5	Vargas	2500	(null)	SAL
6	Zlotkey	10500	0.2	SAL+COMM
7	Abel	11000	0.3	SAL+COMM
8	Taylor	8600	0.2	SAL+COMM

Using the NULLIF Function

```
SELECT first_name, LENGTH(first_name) "expr1",
       last_name,  LENGTH(last_name)  "expr2",
       NULLIF(LENGTH(first_name), LENGTH(last_name)) result
FROM   employees;
```



	FIRST_NAME	expr1	LAST_NAME	expr2	RESULT
1	Ellen	5	Abel	4	5
2	Curtis	6	Davies	6	(null)
3	Lex	3	De Haan	7	3
4	Bruce	5	Ernst	5	(null)
5	Pat	3	Fay	3	(null)
6	William	7	Gietz	5	7
7	Kimberely	9	Grant	5	9

…

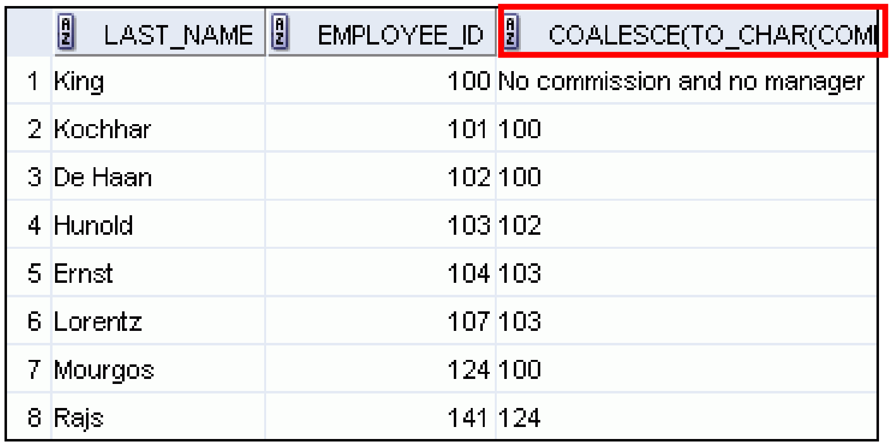
19	Jennifer	8	Whalen	6	8
20	Eleni	5	Zlotkey	7	5

Using the `COALESCE` Function

- The advantage of the `COALESCE` function over the `NVL` function is that the `COALESCE` function can take multiple alternate values.
- If the first expression is not null, the `COALESCE` function returns that expression; otherwise, it does a `COALESCE` of the remaining expressions.

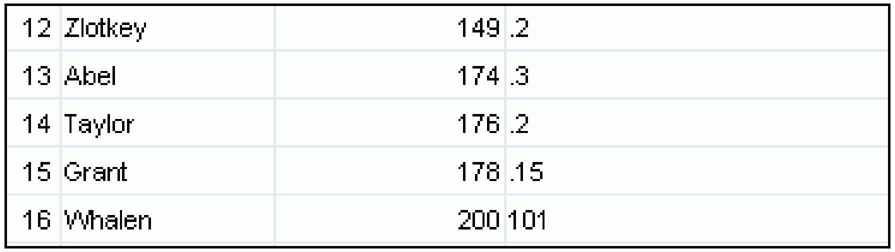
Using the COALESCE Function

```
SELECT last_name, employee_id,
COALESCE(TO_CHAR(commission_pct),TO_CHAR(manager_id),
         'No commission and no manager')
FROM employees;
```



	LAST_NAME	EMPLOYEE_ID	COALESCE(TO_CHAR(COM
1	King	100	No commission and no manager
2	Kochhar	101	100
3	De Haan	102	100
4	Hunold	103	102
5	Ernst	104	103
6	Lorentz	107	103
7	Mourgos	124	100
8	Rajs	141	124

…



12	Zlotkey	149	.2
13	Abel	174	.3
14	Taylor	176	.2
15	Grant	178	.15
16	Whalen	200	101

…

Lesson Stanford

- Implicit and explicit data type conversion
- `TO_CHAR`, `TO_DATE`, `TO_NUMBER` functions
- Nesting functions
- General functions:
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
- Conditional expressions:
  - `CASE`
  - `DECODE`

Conditional Expressions

- Provide the use of the `IF-THEN-ELSE` logic within a SQL statement
- Use two methods:
  - `CASE` expression
  - `DECODE` function

CASE Expression

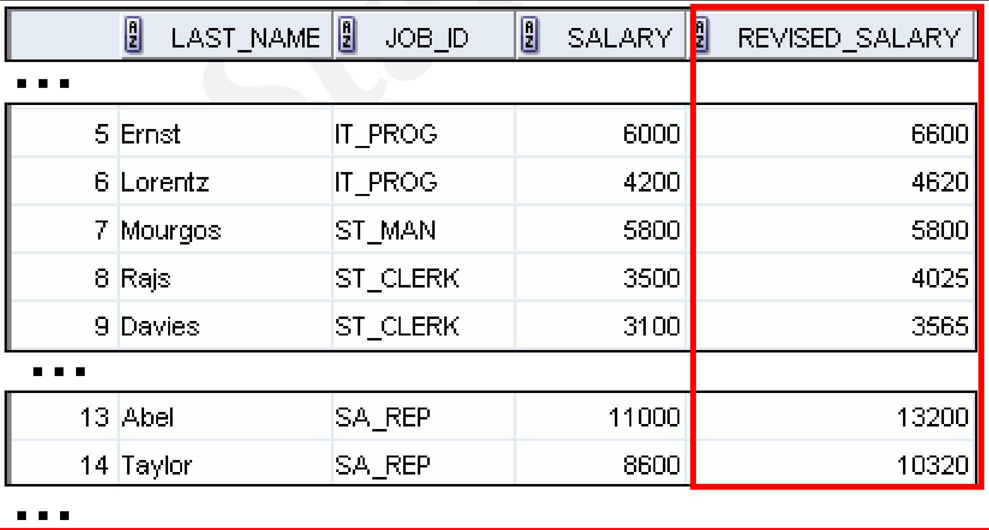
Facilitates conditional inquiries by doing the work of an `IF-THEN-ELSE` statement:

```
CASE expr WHEN comparison_expr1 THEN return_expr1
         [WHEN comparison_expr2 THEN return_expr2
          WHEN comparison_exprn THEN return_exprn
          ELSE else_expr]
END
```

Using the CASE Expression

Facilitates conditional inquiries by doing the work of an `IF-THEN-ELSE` statement:

```
SELECT last_name, job_id, salary,
       CASE job_id WHEN 'IT_PROG'  THEN  1.10*salary
                   WHEN 'ST_CLERK' THEN  1.15*salary
                   WHEN 'SA_REP'   THEN  1.20*salary
       ELSE      salary END     "REVISED_SALARY"
FROM   employees;
```



	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...				
5	Ernst	IT_PROG	6000	6600
6	Lorentz	IT_PROG	4200	4620
7	Mourgos	ST_MAN	5800	5800
8	Rajs	ST_CLERK	3500	4025
9	Davies	ST_CLERK	3100	3565
...				
13	Abel	SA_REP	11000	13200
14	Taylor	SA_REP	8600	10320
...				

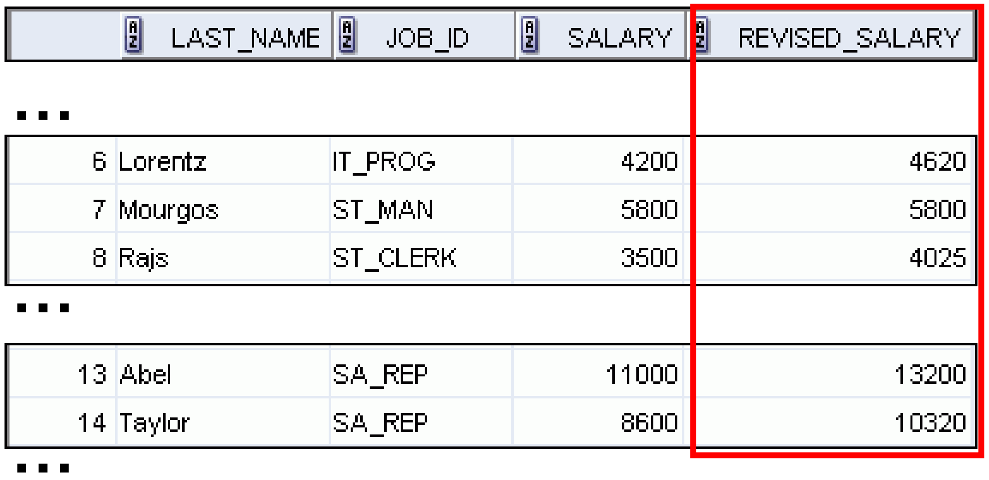
DECODE Function

Facilitates conditional inquiries by doing the work of a CASE expression or an IF-THEN-ELSE statement:

```
DECODE(col|expression, search1, result1
                       [, search2, result2,...,]
                       [, default])
```

Using the DECODE Function

```
SELECT last name, job id, salary,
       DECODE(job_id, 'IT_PROG',  1.10*salary,
                      'ST_CLERK', 1.15*salary,
                      'SA_REP',   1.20*salary,
              salary)
       REVISED_SALARY
FROM   employees;
```



	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...				
6	Lorentz	IT_PROG	4200	4620
7	Mourgos	ST_MAN	5800	5800
8	Rajs	ST_CLERK	3500	4025
...				
13	Abel	SA_REP	11000	13200
14	Taylor	SA_REP	8600	10320
...				

Using the DECODE Function

Display the applicable tax rate for each employee in department 80:

```
SELECT last_name, salary,
       DECODE (TRUNC(salary/2000, 0),
                         0, 0.00,
                         1, 0.09,
                         2, 0.20,
                         3, 0.30,
                         4, 0.40,
                         5, 0.42,
                         6, 0.44,
                            0.45) TAX_RATE
FROM   employees
WHERE  department_id = 80;
```

Summary

In this lesson, you should have learned how to:

- Alter date formats for display using functions
- Convert column data types using functions
- Use `NVL` functions
- Use `IF-THEN-ELSE` logic and other conditional expressions in a `SELECT` statement

Practice 4: Overview

This practice covers the following topics:

- Creating queries that use `TO_CHAR`, `TO_DATE`, and other `DATE` functions
- Creating queries that use conditional expressions such as `DECODE` and `CASE`