Chapter 9

How to use functions

Objectives

Applied

- Code queries that format numeric or date/time data.
- Code queries that require any of the scalar functions presented in this chapter.

Knowledge

- Describe how the use of functions can solve the problems associated with (1) sorting string data that contains numeric values, and (2) doing date or time searches.

Some of the string functions

```
CONCAT(str1[,str2]...)
CONCAT_WS(sep,str1[,str2]...)
LTRIM(str)
RTRIM(str)
TRIM([[BOTH|LEADING|TRAILING] [remove] FROM] str)
LENGTH(str)
LOCATE(find,search[,start])
LEFT(str,length)
RIGHT(str,length)
SUBSTRING_INDEX(str,delimiter, count)
SUBSTRING(str,start[,length])
```

Some of the string functions (continued)

```
REPLACE(search,find,replace)
INSERT(str,start,length,insert)
REVERSE(str)
LOWER(str)
UPPER(str)
LPAD(str,length,pad)
RPAD(str,length,pad)
SPACE(count)
REPEAT(str,count)
```

String function examples

Function	Result
`CONCAT('Last', 'First')`	`'LastFirst'`
`CONCAT_WS(', ', 'Last', 'First')`	`'Last, First'`
`LTRIM('  MySQL  ')`	`'MySQL  '`
`RTRIM('  MySQL  ')`	`'  MySQL'`
`TRIM('  MySQL  ')`	`'MySQL'`
`TRIM(BOTH '*' FROM '****MySQL****')`	`'MySQL'`
`LOWER('MySQL')`	`'mysql'`
`UPPER('ca')`	`'CA'`
`LEFT('MySQL', 3)`	`'MyS'`
`RIGHT('MySQL', 3)`	`'SQL'`

String function examples (continued)

Function	Result
`SUBSTRING('(559) 555-1212', 7, 8)`	`'555-1212'`
`SUBSTRING_INDEX('http://www.murach.com', '.', -2)`	`'murach.com'`
`LENGTH('MySQL')`	`5`
`LENGTH('  MySQL  ')`	`9`
`LOCATE('SQL', '  MySQL')`	`5`
`LOCATE('-', '(559) 555-1212')`	`10`
`REPLACE(RIGHT('(559) 555-1212', 13),') ', '-')`	`'559-555-1212'`
`INSERT("MySQL", 1, 0, "Murach's ")`	`"Murach's MySQL"`
`INSERT('MySQL', 1, 0, 'Murach''s ')`	`"Murach's MySQL"`

A SELECT statement that uses three functions

```
SELECT vendor_name,
       CONCAT_WS(', ', vendor_contact_last_name,
                 vendor_contact_first_name) AS contact_name,
       RIGHT(vendor_phone, 8) AS phone
FROM vendors
WHERE LEFT(vendor_phone, 4) = '(559'
ORDER BY contact_name
```

vendor_name	contact_name	phone
Dristas Groom & McCormick	Aaronsen, Thom	555-8484
Yale Industrial Trucks-Fresno	Alexis, Alexandro	555-2993
Lou Gentile's Flower Basket	Anum, Trisha	555-6643
Pollstar	Aranovitch, Robert	555-2631

How to sort by a string column that contains numbers

Sorted by the emp_id column

```
SELECT *
FROM string_sample
ORDER BY emp_id
```

emp_id	emp_name
1	Lizbeth Darien
17	Lance Pinos-Potter
2	Darnell O'Sullivan
20	Jean Paul Renard
3	Alisha von Strump

How to sort by a string column that contains numbers (continued)

Sorted by the emp_id column explicitly cast as an integer

```
SELECT *
FROM string_sample
ORDER BY CAST(emp_id AS SIGNED)
```

emp_id	emp_name
1	Lizbeth Darien
2	Darnell O'Sullivan
3	Alisha von Strump
17	Lance Pinos-Potter
20	Jean Paul Renard

How to sort by a string column that contains numbers (continued)

Sorted by the emp_id column implicitly cast as an integer

```
SELECT *
FROM string_sample
ORDER BY emp_id + 0
```

emp_id	emp_name
1	Lizbeth Darien
2	Darnell O'Sullivan
3	Alisha von Strump
17	Lance Pinos-Potter
20	Jean Paul Renard

How to sort by a string column that contains numbers (continued)

Sorted by the emp_id column padded with leading zeros

```
SELECT LPAD(emp_id, 2, '0') AS emp_id, emp_name
FROM string_sample
ORDER BY emp_id
```

emp_id	emp_name
01	Lizbeth Darien
02	Darnell O'Sullivan
03	Alisha von Strump
17	Lance Pinos-Potter
20	Jean Paul Renard

How to use the SUBSTRING_INDEX function to parse a string

```
SELECT emp_name,
    SUBSTRING_INDEX(emp_name, ' ', 1) AS first_name,
    SUBSTRING_INDEX(emp_name, ' ', -1) AS last_name
FROM string_sample
```

emp_name	first_name	last_name
Lizbeth Darien	Lizbeth	Darien
Darnell O'Sullivan	Darnell	O'Sullivan
Lance Pinos-Potter	Lance	Pinos-Potter
Jean Paul Renard	Jean	Renard
Alisha von Strump	Alisha	Strump

How to use the LOCATE function to find a character in a string

```
SELECT emp_name,
    LOCATE(' ', emp_name) AS first_space,
    LOCATE(' ', emp_name, LOCATE(' ', emp_name) + 1)
    AS second_space
FROM string_sample
```

emp_name	first_space	second_space
Lizbeth Darien	8	0
Darnell O'Sullivan	8	0
Lance Pinos-Potter	6	0
Jean Paul Renard	5	10
Alisha von Strump	7	11

How to use the SUBSTRING function to parse a string

```
SELECT emp_name,
    SUBSTRING(emp_name, 1, LOCATE(' ', emp_name) - 1)
    AS first_name,
    SUBSTRING(emp_name, LOCATE(' ', emp_name) + 1)
    AS last_name
FROM string_sample
```

emp_name	first_name	last_name
Lizbeth Darien	Lizbeth	Darien
Darnell O'Sullivan	Darnell	O'Sullivan
Lance Pinos-Potter	Lance	Pinos-Potter
Jean Paul Renard	Jean	Paul Renard
Alisha von Strump	Alisha	von Strump

Some of the numeric functions

```
ROUND(number[,length])
TRUNCATE(number,length)
CEILING(number)
FLOOR(number)
ABS(number)
SIGN(number)
SQRT(number)
POWER(number,power)
RAND([integer])
```

Examples that use the numeric functions

Function	Result
`ROUND(12.49,0)`	12
`ROUND(12.50,0)`	13
`ROUND(12.49,1)`	12.5
`TRUNCATE(12.51,0)`	12
`TRUNCATE(12.49,1)`	12.4

Examples that use the numeric functions (continued)

Function	Result
`CEILING(12.5)`	`13`
`CEILING(-12.5)`	`-12`
`FLOOR(-12.5)`	`-13`
`FLOOR(12.5)`	`12`
`ABS(-1.25)`	`1.25`
`ABS(1.25)`	`1.25`
`SIGN(-1.25)`	`-1`
`SIGN(1.25)`	`1`
`SQRT(125.43)`	`11.199553562530964`
`POWER(9,2)`	`81`
`RAND()`	`0.2444132019248`

The Float_Sample table

	float_id	float_value
▶	1	0.999999999999999
	2	1
	3	1.000000000000001
	4	1234.56789012345
	5	999.04440209348
	6	24.04849

A search for an exact value that doesn't include two approximate values

```
SELECT *
FROM float_sample
WHERE float_value = 1
```

	float_id	float_value
▶	2	1

How to search for approximate values

Search for a range of values

```
SELECT *
FROM float_sample
WHERE float_value BETWEEN 0.99 AND 1.01
```

float_id	float_value
1	0.999999999999999
2	1
3	1.000000000000001

Search for rounded values

```
SELECT *
FROM float_sample
WHERE ROUND(float_value, 2) = 1.00
```

float_id	float_value
1	0.999999999999999
2	1
3	1.000000000000001

Functions that get the current date and time

```
NOW()
SYSDATE()
CURRENT_TIMESTAMP()

CURDATE()
CURRENT_DATE()

CURTIME()
CURRENT_TIME()

UTC_DATE()

UTC_TIME()
```

Examples that get the current date and time

Function	Result
`NOW()`	`2014-09-29 14:12:04`
`SYSDATE()`	`2014-09-29 14:12:04`
`CURDATE()`	`2014-09-29`
`CURTIME()`	`14:12:04`
`UTC_DATE()`	`2014-09-29`
`UTC_TIME()`	`21:12:04`
`CURRENT_TIMESTAMP()`	`2014-09-29 14:12:04`
`CURRENT_DATE()`	`2014-09-29`
`CURRENT_TIME()`	`14:12:04`

Some of the date/time parsing functions

```
DAYOFMONTH(date)
MONTH(date)
YEAR(date)
HOUR(time)
MINUTE(time)
SECOND(time)
DAYOFWEEK(date)

QUARTER(date)
DAYOFYEAR(date)
WEEK(date[,first])

LAST_DAY(date)

DAYNAME(date)
MONTHNAME(date)
```

Examples that use the date/time parsing functions

Function	Result
`DAYOFMONTH('2014-09-03')`	3
`MONTH('2014-09-03')`	9
`YEAR('2014-09-03')`	2011
`HOUR('11:35:00')`	11
`MINUTE('11:35:00')`	35
`SECOND('11:35:00')`	0
`DAYOFWEEK('2014-09-03')`	7
`QUARTER('2014-09-03')`	3
`DAYOFYEAR('2014-09-03')`	246
`WEEK('2014-09-03')`	35
`LAST_DAY('2014-09-03')`	30
`DAYNAME('2014-09-03')`	Saturday
`MONTHNAME('2014-09-03')`	September

The EXTRACT function

```
EXTRACT(unit FROM date)
```

Date/time units

Unit	Description
`SECOND`	Seconds
`MINUTE`	Minutes
`HOUR`	Hours
`DAY`	Day
`MONTH`	Month
`YEAR`	Year

Date/time units (continued)

Unit	Description
`MINUTE_SECOND`	Minutes and seconds
`HOUR_MINUTE`	Hour and minutes
`DAY_HOUR`	Day and hours
`YEAR_MONTH`	Year and month
`HOUR_SECOND`	Hours, minutes, and seconds
`DAY_MINUTE`	Day, hours, and minutes
`DAY_SECOND`	Day, hours, minutes, and seconds

Examples that use the EXTRACT function

Function	Result
`EXTRACT(SECOND FROM '2014-09-03 11:35:00')`	0
`EXTRACT(MINUTE FROM '2014-09-03 11:35:00')`	35
`EXTRACT(HOUR FROM '2014-09-03 11:35:00')`	11
`EXTRACT(DAY FROM '2014-09-03 11:35:00')`	3
`EXTRACT(MONTH FROM '2014-09-03 11:35:00')`	9
`EXTRACT(YEAR FROM '2014-09-03 11:35:00')`	2011
`EXTRACT(MINUTE_SECOND FROM '2014-09-03 11:35:00')`	3500
`EXTRACT(HOUR_MINUTE FROM '2014-09-03 11:35:00')`	1135
`EXTRACT(DAY_HOUR FROM '2014-09-03 11:35:00')`	311
`EXTRACT(YEAR_MONTH FROM '2014-09-03 11:35:00')`	201109
`EXTRACT(HOUR_SECOND FROM '2014-09-03 11:35:00')`	113500
`EXTRACT(DAY_MINUTE FROM '2014-09-03 11:35:00')`	31135
`EXTRACT(DAY_SECOND FROM '2014-09-03 11:35:00')`	3113500

Two functions for formatting dates and times

```
DATE_FORMAT(date,format)
TIME_FORMAT(time,format)
```

Common codes for date/time format strings

Code	Description
`%m`	Month, numeric (01…12)
`%c`	Month, numeric (1…12)
`%M`	Month name (January…December)
`%b`	Abbreviated month name (Jan…Dec)
`%d`	Day of the month, numeric (00…31)
`%e`	Day of the month, numeric (0…31)
`%D`	Day of the month with suffix (1st, 2nd, 3rd, etc.)
`%y`	Year, numeric, 2 digits
`%Y`	Year, numeric, 4 digits

Common codes for date/time format strings (continued)

Code	Description
`%W`	Weekday name (Sunday…Saturday)
`%a`	Abbreviated weekday name (Sun…Sat)
`%H`	Hour (00…23)
`%k`	Hour (0…23)
`%h`	Hour (01…12)
`%l`	Hour (1…12)
`%i`	Minutes (00…59)
`%r`	Time, 12-hour (hh:mm:ss AM or PM)
`%T`	Time, 24-hour (hh:mm:ss)
`%S`	Seconds (00…59)
`%p`	AM or PM

Examples that use the date/time formatting functions

Function	Result
`DATE_FORMAT('2014-09-03', '%m/%d/%y')`	`09/03/14`
`DATE_FORMAT('2014-09-03', '%W, %M %D, %Y')`	`Saturday, September 3rd, 2014`
`DATE_FORMAT('2014-09-03', '%e-%b-%y')`	`3-Sep-14`
`DATE_FORMAT('2014-09-03 16:45', '%r')`	`04:45:00 PM`
`TIME_FORMAT('16:45', '%r')`	`04:45:00 PM`
`TIME_FORMAT('16:45', '%l:%i %p')`	`4:45 PM`

Some of the functions for calculating dates and times

```
DATE_ADD(date,INTERVAL expression unit)
DATE_SUB(date,INTERVAL expression unit)
DATEDIFF(date1, date2)
TO_DAYS(date)
TIME_TO_SEC(time)
```

Examples of the functions for calculating dates and times

Function	Result
`DATE_ADD('2014-12-31', INTERVAL 1 DAY)`	2015-01-01
`DATE_ADD('2014-12-31', INTERVAL 3 MONTH)`	2015-03-31
`DATE_ADD('2014-12-31 23:59:59', INTERVAL 1 SECOND)`	2015-01-01 00:00:00
`DATE_ADD('2015-01-01', INTERVAL -1 DAY)`	2014-12-31
`DATE_SUB('2015-01-01', INTERVAL 1 DAY)`	2014-12-31
`DATE_ADD('2011-02-29', INTERVAL 1 YEAR)`	2012-02-28
`DATE_ADD('2012-02-29', INTERVAL 1 YEAR)`	NULL
`DATE_ADD('2014-12-31 12:00', INTERVAL '2 12' DAY_HOUR)`	2015-01-03 00:00:00

Examples of the functions for calculating dates and times (continued)

Function	Result
`DATEDIFF('2014-09-30', '2014-09-03')`	27
`DATEDIFF('2014-09-30 23:59:59', '2014-09-03')`	27
`DATEDIFF('2014-09-03', '2014-09-30')`	-27
`TO_DAYS('2014-09-30') - TO_DAYS('2014-09-03')`	27
`TIME_TO_SEC('10:00') - TIME_TO_SEC('09:59')`	60

The contents of the Date_Sample table

date_id	start_date
1	1982-03-01 00:00:00
2	2002-02-28 00:00:00
3	2006-10-31 00:00:00
4	2014-02-28 10:00:00
5	2015-02-28 13:58:32
6	2015-03-01 09:02:25

A SELECT statement that fails to return a row

```
SELECT *
FROM date_sample
WHERE start_date = '2014-02-28'
```

date_id	start_date

Three techniques for ignoring time values

Search for a range of dates

```
SELECT *
FROM date_sample
WHERE start_date >= '2014-02-28'
  AND start_date < '2014-03-01'
```

date_id	start_date
4	2014-02-28 10:00:00

Search for month, day, and year integers

```
SELECT *
FROM date_sample
WHERE MONTH(start_date) = 2 AND
      DAYOFMONTH(start_date) = 28 AND
      YEAR(start_date) = 2014
```

date_id	start_date
4	2014-02-28 10:00:00

Three techniques for ignoring time values (continued)

Search for a formatted date

```
SELECT *
FROM date_sample
WHERE DATE_FORMAT(start_date, '%m-%d-%Y') = '02-28-2014'
```

date_id	start_date
4	2014-02-28 10:00:00

The contents of the Date_Sample table

date_id	start_date
1	1982-03-01 00:00:00
2	2002-02-28 00:00:00
3	2006-10-31 00:00:00
4	2014-02-28 10:00:00
5	2015-02-28 13:58:32
6	2015-03-01 09:02:25

A SELECT statement that fails to return a row

```
SELECT * FROM date_sample
WHERE start_date = '10:00:00'
```

date_id	start_date

Examples that ignore date values

Search for a time that has been formatted

```
SELECT * FROM date_sample
WHERE DATE_FORMAT(start_date, '%T') = '10:00:00'
```

date_id	start_date
4	2014-02-28 10:00:00

Search for a time that hasn't been formatted

```
SELECT * FROM date_sample
WHERE EXTRACT(HOUR_SECOND FROM start_date) = 100000
```

date_id	start_date
4	2014-02-28 10:00:00

Examples that ignore date values (continued)

Search for an hour of the day

```
SELECT * FROM date_sample
WHERE HOUR(start_date) = 9
```

date_id	start_date
6	2015-03-01 09:02:25

Search for a range of times

```
SELECT * FROM date_sample
WHERE EXTRACT(HOUR_MINUTE FROM start_date)
          BETWEEN 900 AND 1200
```

date_id	start_date
4	2014-02-28 10:00:00
6	2015-03-01 09:02:25

The syntax of the simple CASE function

```
CASE input_expression
     WHEN when_expression_1 THEN result_expression_1
    [WHEN when_expression_2 THEN result_expression_2]...
    [ELSE else_result_expression]
END
```

A statement that uses a simple CASE function

```
SELECT invoice_number, terms_id,
    CASE terms_id
        WHEN 1 THEN 'Net due 10 days'
        WHEN 2 THEN 'Net due 20 days'
        WHEN 3 THEN 'Net due 30 days'
        WHEN 4 THEN 'Net due 60 days'
        WHEN 5 THEN 'Net due 90 days'
    END AS terms
FROM invoices
```

invoice_number	terms_id	terms
111-92R-10096	2	Net due 20 days
25022117	4	Net due 60 days
P02-88D77S7	3	Net due 30 days

The syntax of the searched CASE function

```
CASE
    WHEN conditional_expression_1
        THEN result_expression_1
   [WHEN conditional_expression_2
        THEN result_expression_2]...
   [ELSE else_result_expression]
END
```

A statement that uses a searched CASE function

```
SELECT invoice_number, invoice_total, invoice_date,
       invoice_due_date,
    CASE
      WHEN DATEDIFF(NOW(), invoice_due_date) > 30
        THEN 'Over 30 days past due'
      WHEN DATEDIFF(NOW(), invoice_due_date) > 0
        THEN '1 to 30 days past due'
      ELSE 'Current'
    END AS invoice_status
FROM invoices
WHERE invoice_total - payment_total - credit_total > 0
```

invoice_number	invoice_total	invoice_date	invoice_due_date	invoice_status
39104	85.31	2014-07-10	2014-08-09	Over 30 days past due
963253264	52.25	2014-07-18	2014-08-17	Over 30 days past due
31361833	579.42	2014-07-21	2014-08-10	Over 30 days past due

The syntax of the IF function

```
IF(test_expression, if_true_expression, else_expression)
```

A SELECT statement that uses the IF function

```
SELECT vendor_name,
       IF(vendor_city = 'Fresno', 'Yes', 'No')
       AS is_city_fresno
FROM vendors
```

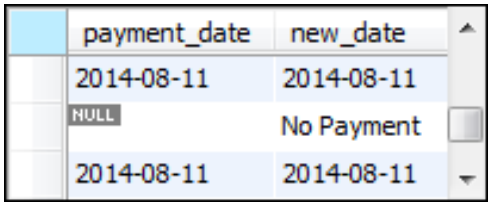
vendor_name	is_city_fresno
Towne Advertiser's Mailing Svcs	No
BFI Industries	Yes
Pacific Gas & Electric	No
Robbins Mobile Lock And Key	Yes
Bill Marvin Electric Inc	Yes

The syntax of the IFNULL function

```
IFNULL(test_expression, replacement_value)
```

A SELECT statement that uses the IFNULL function

```
SELECT payment_date,
       IFNULL(payment_date, 'No Payment') AS new_date
FROM invoices
```



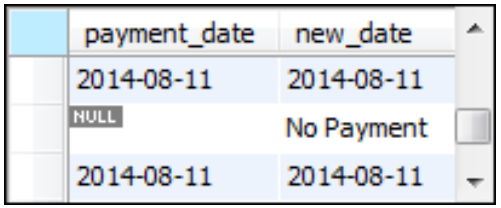
payment_date	new_date
2014-08-11	2014-08-11
NULL	No Payment
2014-08-11	2014-08-11

The syntax of the COALESCE function

```
COALESCE(expression_1[, expression_2]...)
```

A SELECT statement that uses the COALESCE function

```
SELECT payment_date,
       COALESCE(payment_date, 'No Payment') AS new_date
FROM invoices
```



payment_date	new_date
2014-08-11	2014-08-11
NULL	No Payment
2014-08-11	2014-08-11