

Date Functions

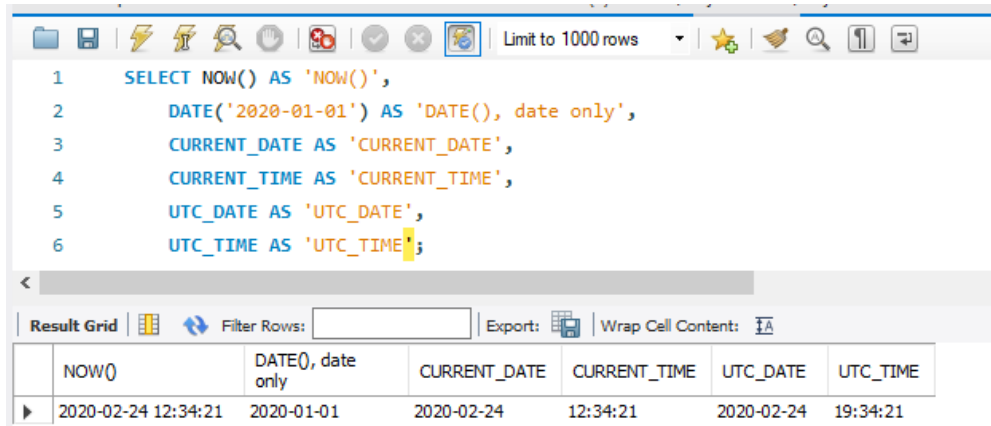
Function	Type	Example	Result
NOW() * Returns current local date and time.	date/time	NOW()	ex. '2020-02-24 09:31:31'
DATE(date)	date/time	DATE('2020-01-01 11:31:31')	'2020-02-24'
CURRENT_DATE() * Returns current local date	date	CURRENT_DATE	'2020-02-24'
CURRENT_TIME() * Returns current local time.	time	CURRENT_TIME	'11:52:10'
UTC_DATE() * Returns current UTC date.	date	UTC_DATE	'2020-02-24'
UTC_TIME() * Returns current UTC date.	time	UTC_TIME	'18:52:10'

- There are a number of functions that give the current date and time. The DATE() function is a date formatting function, but I include it in the list because it is often confused with the NOW() function.
- CURRENT_DATE, CURRENT_TIME, UTC_DATE, UTC_TIME can be used with the parentheses "()" or not. They accept no parameters

Example

```
SELECT NOW() AS 'NOW()',  
       DATE('2020-01-01') AS 'DATE()', date only',  
       CURRENT_DATE AS 'CURRENT_DATE',  
       CURRENT_TIME AS 'CURRENT_TIME',  
       UTC_DATE AS 'UTC_DATE',  
       UTC_TIME AS 'UTC_TIME';
```

Results



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 SELECT NOW() AS 'NOW()',
2     DATE('2020-01-01') AS 'DATE(), date only',
3     CURRENT_DATE AS 'CURRENT_DATE',
4     CURRENT_TIME AS 'CURRENT_TIME',
5     UTC_DATE AS 'UTC_DATE',
6     UTC_TIME AS 'UTC_TIME';
```

Below the query, the results are displayed in a table with the following columns and values:

	NOW()	DATE(), date only	CURRENT_DATE	CURRENT_TIME	UTC_DATE	UTC_TIME
▶	2020-02-24 12:34:21	2020-01-01	2020-02-24	12:34:21	2020-02-24	19:34:21

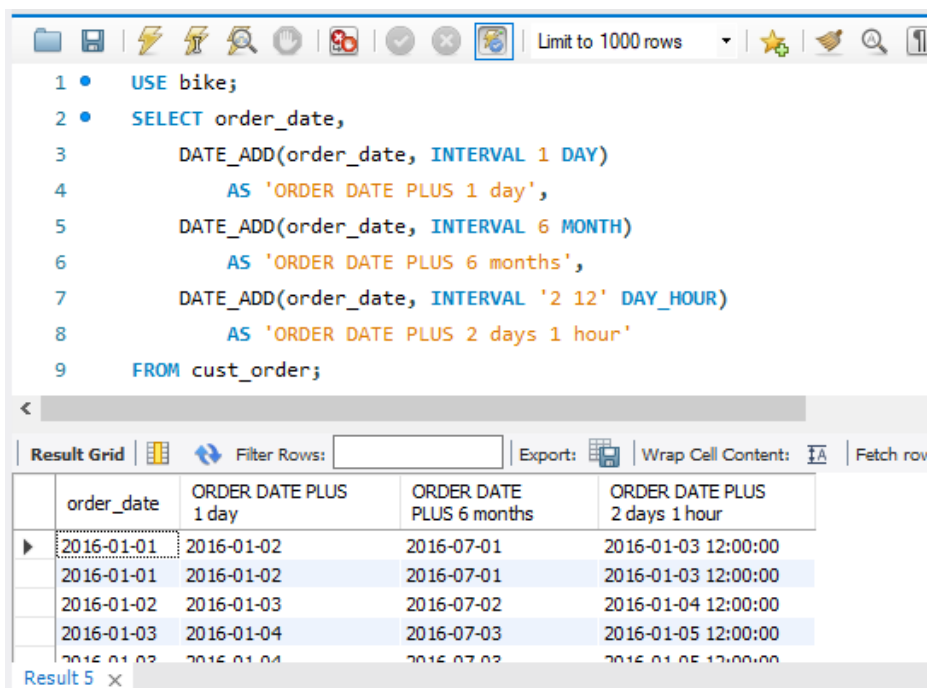
Function	Type	Example	Result
DATE_ADD(date, interval expression unit)	DATE, DATETIME	DATE_ADD('2020-01-01', INTERVAL 1 DAY)	'202-01-02'

- Returns a date with a DATE or DATETIME value equal to the original value plus the specified interval.

Example

```
USE bike;
SELECT order_date,
       DATE_ADD(order_date, INTERVAL 1 DAY) AS 'ORDER DATE PLUS 1 day',
       DATE_ADD(order_date, INTERVAL 6 MONTH) AS 'ORDER DATE PLUS 6 months',
       DATE_ADD(order_date, INTERVAL '2 12' DAY_HOUR)
       AS 'ORDER DATE PLUS 2 days 1 hour'
FROM cust_order;
```

Results



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and settings. The query editor contains the following SQL code:

```
1 • USE bike;
2 • SELECT order_date,
3       DATE_ADD(order_date, INTERVAL 1 DAY)
4       AS 'ORDER DATE PLUS 1 day',
5       DATE_ADD(order_date, INTERVAL 6 MONTH)
6       AS 'ORDER DATE PLUS 6 months',
7       DATE_ADD(order_date, INTERVAL '2 12' DAY_HOUR)
8       AS 'ORDER DATE PLUS 2 days 1 hour'
9 FROM cust_order;
```

Below the query editor, the 'Result Grid' is displayed. It shows the results of the query execution. The grid has four columns: 'order_date', 'ORDER DATE PLUS 1 day', 'ORDER DATE PLUS 6 months', and 'ORDER DATE PLUS 2 days 1 hour'. The first row of data is highlighted, showing the date '2016-01-01' and its corresponding results.

order_date	ORDER DATE PLUS 1 day	ORDER DATE PLUS 6 months	ORDER DATE PLUS 2 days 1 hour
2016-01-01	2016-01-02	2016-07-01	2016-01-03 12:00:00
2016-01-01	2016-01-02	2016-07-01	2016-01-03 12:00:00
2016-01-02	2016-01-03	2016-07-02	2016-01-04 12:00:00
2016-01-03	2016-01-04	2016-07-03	2016-01-05 12:00:00
2016-01-03	2016-01-04	2016-07-03	2016-01-05 12:00:00

The bottom of the screenshot shows the 'Result 5' tab, indicating that the query was the fifth result in the session.

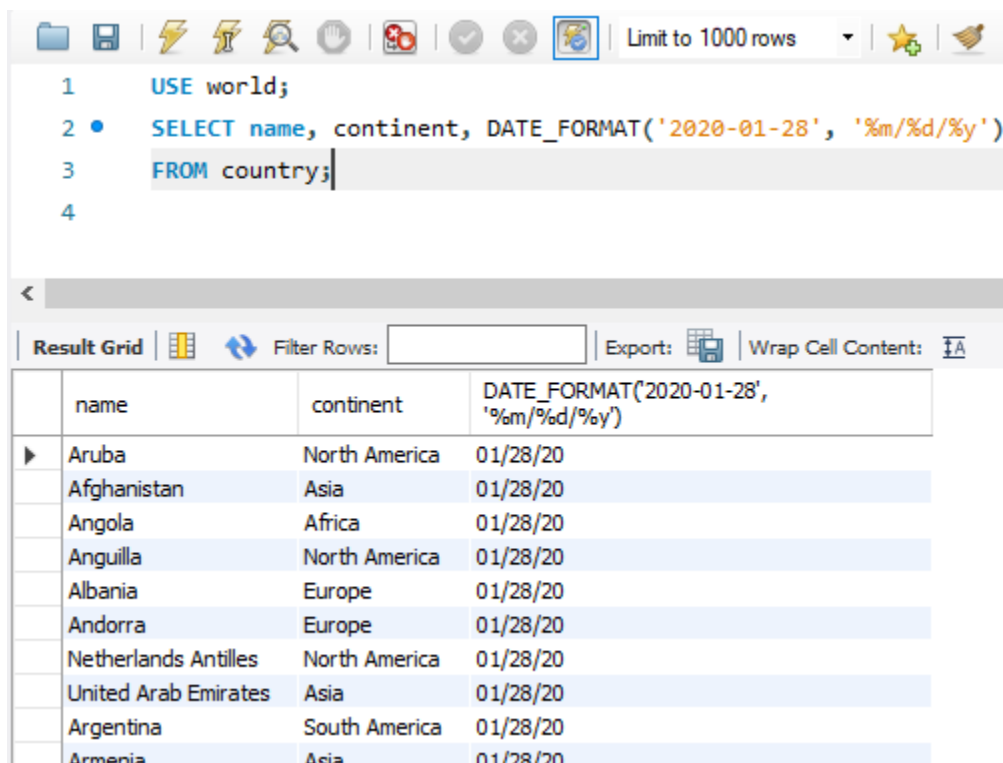
Function	Type	Example	Result
DATE_FORMAT	DATE	DATE_FORMAT('2020-09-03', '%m/%d/%y')	09/03/14

- Dates must be enclosed in quotes
- You can pass a DATE or DATETIME datatype to DATE_FORMAT

Example

```
USE world;
SELECT name, continent, DATE_FORMAT('2020-01-28', '%m/%d/%y')
FROM country;
```

Results



	name	continent	DATE_FORMAT('2020-01-28', '%m/%d/%y')
▶	Aruba	North America	01/28/20
	Afghanistan	Asia	01/28/20
	Angola	Africa	01/28/20
	Anguilla	North America	01/28/20
	Albania	Europe	01/28/20
	Andorra	Europe	01/28/20
	Netherlands Antilles	North America	01/28/20
	United Arab Emirates	Asia	01/28/20
	Argentina	South America	01/28/20
	Armenia	Asia	01/28/20

Format List

Specifier	Description
%a	Abbreviated weekday name (Sun..Sat)
%b	Abbreviated month name (Jan..Dec)
%c	Month, numeric (0..12)
%D	Day of the month with English suffix (0th, 1st, 2nd, 3rd, ...)
%d	Day of the month, numeric (00..31)
%e	Day of the month, numeric (0..31)
%f	Microseconds (000000..999999)
%H	Hour (00..23)
%h	Hour (01..12)
%I	Hour (01..12)
%i	Minutes, numeric (00..59)
%j	Day of year (001..366)
%k	Hour (0..23)
%l	Hour (1..12)
%M	Month name (January..December)
%m	Month, numeric (00..12)
%p	AM or PM
%r	Time, 12-hour (<i>hh:mm:ss</i> followed by AM or PM)
%S	Seconds (00..59)
%s	Seconds (00..59)
%T	Time, 24-hour (<i>hh:mm:ss</i>)
%U	Week (00..53), where Sunday is the first day of the week; <u>WEEK()</u> mode 0
%u	Week (00..53), where Monday is the first day of the week; <u>WEEK()</u> mode 1
%V	Week (01..53), where Sunday is the first day of the week; <u>WEEK()</u> mode 2; used with %X
%v	Week (01..53), where Monday is the first day of the week; <u>WEEK()</u> mode 3; used with %x
%W	Weekday name (Sunday..Saturday)
%w	Day of the week (0=Sunday..6=Saturday)
%X	Year for the week where Sunday is the first day of the week, numeric, four digits; used with %v
%x	Year for the week, where Monday is the first day of the week, numeric, four digits; used with %v
%Y	Year, numeric, four digits
%y	Year, numeric (two digits)
%%	A literal % character
% <i>x</i>	<i>x</i> , for any “ <i>x</i> ” not listed above

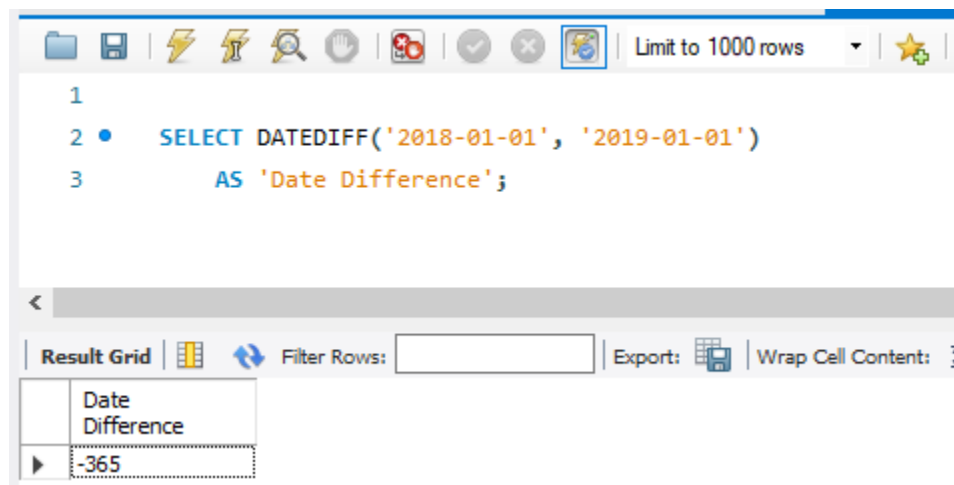
Function	Type	Example	Result
DATEDIFF(date1, date2]	DATE	DATEDIFF('2018-01-01', '2019-01-01')	-356

- The DATEDIFF function has two parameters. Both are dates.
- The value returned by the function is an integer and is the number of days between the two dates.
- If you provide the latest date, first the results will be positive. Otherwise, it will be negative.

Example

```
SELECT DATEDIFF('2018-01-01', '2019-01-01')  
AS 'Date Difference';
```

Results



Numeric Functions

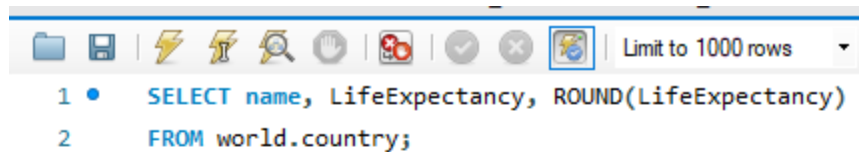
Function	Type	Example	Result
ROUND(number[, length])	Number	ROUND(13.37, 1)	13.4

- The ROUND function has two parameters. The first is a number, usually a DECIMAL or a FLOAT. The second defines the number of decimals to which the number will be rounded.
- If no length is provided, the number is rounded to a whole number.

Example

```
USE world;  
SELECT name, LifeExpectancy, ROUND(LifeExpectancy)  
FROM world.country;
```

Results



A screenshot of a SQL query results grid. The grid has columns for 'name', 'LifeExpectancy', and 'ROUND(LifeExpectancy)'. The first row is highlighted with a blue arrow pointing to it. The data is as follows:

	name	LifeExpectancy	ROUND(LifeExpectancy)
▶	Aruba	78.4	78
	Afghanistan	45.9	46
	Angola	38.3	38
	Anguilla	76.1	76
	Albania	71.6	72
	Andorra	83.5	84
	Netherlands Antilles	74.7	75
	United Arab Emirates	74.1	74

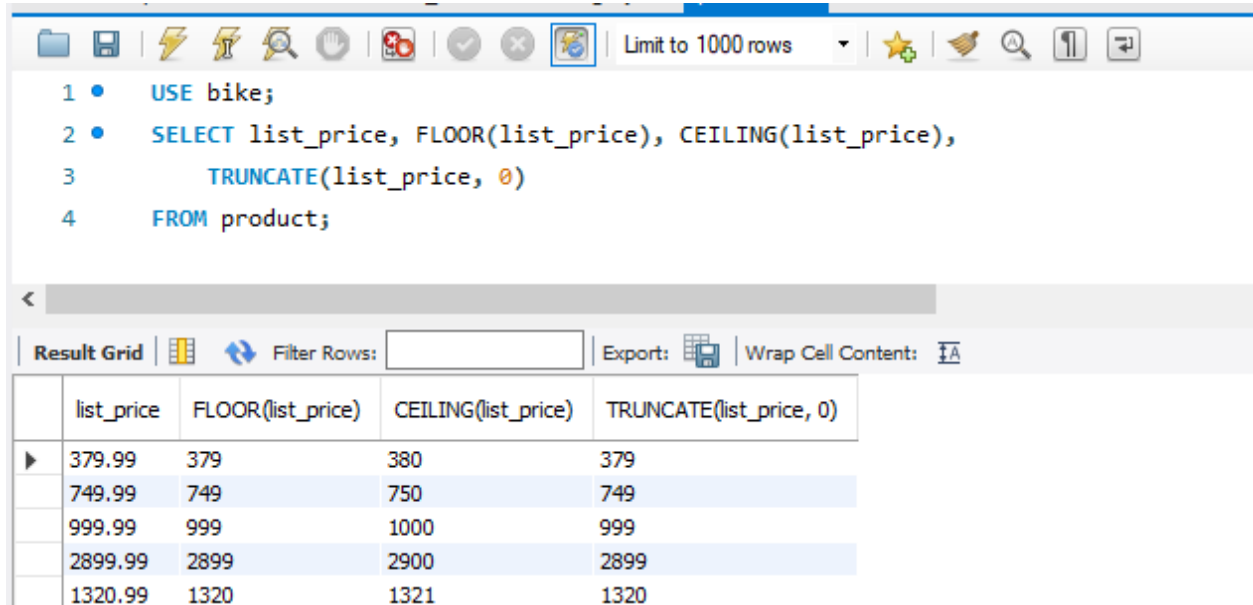
Function	Type	Example	Result
FLOOR(number)	number	FLOOR(7.7)	7
CEILING(number)	number	CEILING(6.2)	7
TRUNCATE(NUMBER, length)	number	TRUNCATE(7.9)	7

- FLOOR() will return the next lowest whole number no matter what the decimal point.
- CEILING() will return the next highest whole number no matter what the decimal point.
- TRUNCATE() will return the number truncated to the precision specified.

Example

```
USE bike;
SELECT list_price, FLOOR(list_price), CEILING(list_price),
       TRUNCATE(list_price, 0)
FROM product;
```

Results



The screenshot shows a database query tool interface. At the top, there is a toolbar with various icons for file operations, editing, and viewing. Below the toolbar, the SQL query is displayed in a text area, numbered 1 through 4. The query is:
1 • USE bike;
2 • SELECT list_price, FLOOR(list_price), CEILING(list_price),
3 TRUNCATE(list_price, 0)
4 FROM product;
Below the query, there is a 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The result grid itself is a table with 5 columns: list_price, FLOOR(list_price), CEILING(list_price), and TRUNCATE(list_price, 0). It contains 5 rows of data, with the first row highlighted in blue.

	list_price	FLOOR(list_price)	CEILING(list_price)	TRUNCATE(list_price, 0)
▶	379.99	379	380	379
	749.99	749	750	749
	999.99	999	1000	999
	2899.99	2899	2900	2899
	1320.99	1320	1321	1320

String Functions

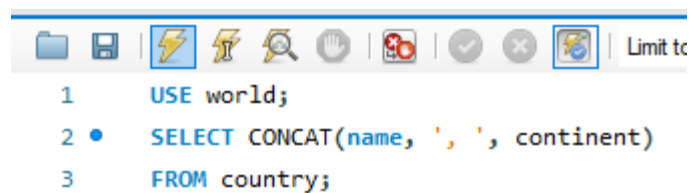
Function	Type	Example	Result
CONCAT(string1[, string2]...)	String	CONCAT("Salmon", "Fish")	Salmon Fish

- Can include column values and literal values.
- In MySQL literal values can be enclosed with either single (') or double quotes (").

Example

```
USE world;  
SELECT CONCAT(name, ', ', continent)  
FROM country;
```

Results



A screenshot of the SQL query results grid. The toolbar at the top includes a 'Result Grid' button, a 'Filter Rows' input field, and an 'Export' button. The query results are displayed in a table with the following data:

CONCAT(name, ', ', continent)
Aruba, North America
Afghanistan, Asia
Angola, Africa
Anguilla, North America
Albania, Europe
Andorra, Europe
Netherlands Antilles, North America
United Arab Emirates, Asia

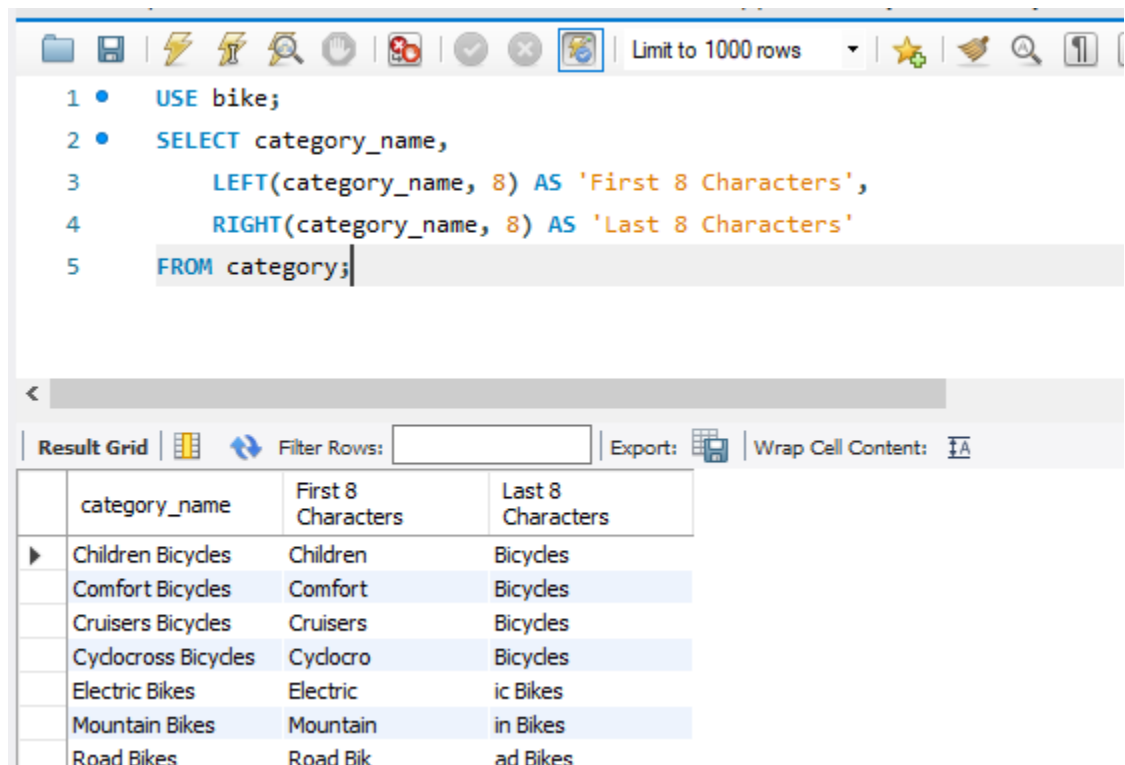
Function	Type	Example	Result
RIGHT(string, num. characters)	string	RIGHT('Salmon', 3)	mon
LEFT(string, num. characters)	string	LEFT('Salmon', 3)	Sal

- The RIGHT and LEFT functions have two parameters. The first is a string and the second is the number of characters to be returned.
- The RIGHT function starts counting from the right side of the string.
- The LEFT function starts counting from the left side of the string.

Example

```
USE bike;
SELECT category_name,
       LEFT(category_name, 8) AS 'First 8 Characters',
       RIGHT(category_name, 8) AS 'Last 8 Characters'
FROM category;
```

Results



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and search. The query editor contains the following SQL code:

```
1 • USE bike;
2 • SELECT category_name,
3       LEFT(category_name, 8) AS 'First 8 Characters',
4       RIGHT(category_name, 8) AS 'Last 8 Characters'
5 FROM category;
```

Below the query editor, the 'Result Grid' is displayed. It shows a table with four columns: 'category_name', 'First 8 Characters', and 'Last 8 Characters'. The table contains seven rows of data, representing different bicycle categories. The 'Last 8 Characters' column shows the last eight characters of each category name, such as 'Bicydes' for 'Children Bicycles' and 'ad Bikes' for 'Road Bikes'.

	category_name	First 8 Characters	Last 8 Characters
▶	Children Bicycles	Children	Bicydes
	Comfort Bicycles	Comfort	Bicydes
	Cruisers Bicycles	Cruisers	Bicydes
	Cyclocross Bicycles	Cyclocro	Bicydes
	Electric Bikes	Electric	ic Bikes
	Mountain Bikes	Mountain	in Bikes
	Road Bikes	Road Bik	ad Bikes

Function	Type	Example	Result
TRIM(string)	string	TRIM(' Salmon ')	'salmon'
LTRIM(string)	string	LEFT('Salmon ')	'salmon '
RTRIM(string)	string	RIGHT(' Salmon')	' salmon'

- The TRIM function will remove leading and trailing spaces from a string.
- The LTRIM function will remove leading spaces from a string.
- The RTRIM function will remove trailing spaces from a string.

Example

```
SELECT LTRIM(' salmon ') AS "Left Trim",
       RTRIM(' salmon ') AS "Right Trim",
       TRIM(' salmon ') AS "Trim";
```

Results

The screenshot shows a database query editor with the following SQL query:

```
1 • SELECT LTRIM(' salmon ') AS "Left Trim",
2       RTRIM(' salmon ') AS "Right Trim",
3       TRIM(' salmon ') AS "Trim";
```

Annotations with red arrows point to the results:

- "Spaces to the right remain" points to the trailing space in the "Left Trim" result.
- "Spaces to the left remain" points to the leading space in the "Right Trim" result.
- "All spaces removed" points to the result of the TRIM function.

The results are displayed in a table with columns: Left Trim, Right Trim, and Trim.

Left Trim	Right Trim	Trim
salmon	salmon	salmon

Function	Type	Example	Result
FORMAT(number, decimal)	string	FORMAT(1234.342, 2)	-356

- FORMAT() accepts a decimal but returns a comma formatted string.

Example

```
SELECT FORMAT(list_price,2)
FROM bike.product;
```

Results

The screenshot shows a database query editor interface. At the top, there is a toolbar with various icons for file operations, search, and execution. Below the toolbar, the SQL query is entered in a text area:

```
1 • SELECT list_price, FORMAT(list_price,2) AS formatted
2 FROM bike.product;
```

Below the query editor, there is a section for the results. It includes a "Result Grid" tab, a "Filter Rows" input field, and an "Export" button. The "Result Grid" displays the following data:

	list_price	formatted
▶	379.99	379.99
	749.99	749.99
	999.99	999.99
	2899.99	2,899.99
	1320.99	1,320.99
	469.99	469.99
	3000.00	3,000.00

Function	Type	Example	Result
LOWER(string)	string	LOWER('Salmon ')	'salmon'
UPPER(string)	string	UPPER('Salmon')	'SALMON'

- LOWER() converts all characters to lower case.
- UPPER() converts all characters to upper case.

Example

```
SELECT UPPER('Salmon'),
       LOWER('Salmon');
```

Results

The screenshot shows a SQL query editor with the following SQL query:

```
1 • SELECT UPPER('Salmon'),
2     LOWER('Salmon');
3
4
5
```

Below the query editor is a results grid. The grid has two columns: 'UPPER('Salmon')' and 'LOWER('Salmon')'. The first row of data shows 'SALMON' and 'salmon'.

	UPPER('Salmon')	LOWER('Salmon')
▶	SALMON	salmon

Function	Type	Example	Result
LOCATE(find,search[,start])	string	LOCATE('al','salmon',1)	2
LENGTH(str)	string	LENGTH('salmon')	6
SUBSTRING(str,start[,length])	string	SUBSTRING('salmon',3,999)	'lmon'

- LOCATE(), and LENGTH() accept a string but return an integer.
- SUBSTRING() accepts a string and returns a string.

Example

```
SELECT LOCATE('al','salmon',1),
       LENGTH('salmon'),
       SUBSTRING('salmon',3,999);
```

Results

The screenshot shows a database query editor interface. At the top, there is a toolbar with various icons for file operations, search, and execution. Below the toolbar, the SQL query is entered in a text area:

```
1 • SELECT LOCATE('al','salmon',1),
2       LENGTH('salmon'),
3       SUBSTRING('salmon',3,999);
4
5
```

Below the query editor, there is a "Result Grid" section. It includes a "Filter Rows:" input field, an "Export:" button, and a "Wrap Cell Content:" checkbox. The results are displayed in a table with three columns: LOCATE('al','salmon',1), LENGTH('salmon'), and SUBSTRING('salmon',3,999). The first row of results shows the values 2, 6, and lmon.

	LOCATE('al','salmon',1)	LENGTH('salmon')	SUBSTRING('salmon',3,999)
▶ 2	2	6	lmon