Table of Contents

There are functions that let you change data from one type to another and functions that let you format data. Functions that relate specifically to date formatting are included in the document on temporal functions.

1. Binary

We have mentioned binary before. The Binary operator casts a string to a binary string. This makes various operations case sensitive. The demo technique here uses a test on the select statement; if the test has a value true then it returns a 1; if the test is false it returns a 0.

Demo 01: **Binary**

```
select  'DOG' = 'dog', binary 'DOG' = 'dog';
+---------------+----------------------+
| 'DOG' = 'dog' | binary 'DOG' = 'dog' |
+---------------+----------------------+
|             1 |                    0 |
+---------------+----------------------+
```

2. Cast and Convert

Cast and Convert are used to change a data value from one type to another.

Demo 02: **CAST()**

```
select cast(123.567 as char);
+-----------------------+
| cast(123.567 as char) |
+-----------------------+
| 123.567               |
+-----------------------+
```

You will get a warning for some casts. Use the \W command to see warnings automatically or use show warnings to display the most recent warning message. This cast is destroying the numeric value.

```
select cast(123.567 as char(2));
+--------------------------+
| cast(123.567 as char(2)) |
+--------------------------+
| 12                       |
+--------------------------+
1 row in set, 1 warning (0.00 sec)
Warning (Code 1292): Truncated incorrect CHAR(2) value: '123.567'
```

Demo 03: **CAST()**

```
select cast('abcsdefg' as char(2));
+-----------------------------+
| cast('abcsdefg' as char(2))  |
+-----------------------------+
| ab                          |
+-----------------------------+
1 row in set, 1 warning (0.00 sec)
Warning (Code 1292): Truncated incorrect CHAR(2) value: 'abcsdefg'
```

Demo 04: Some casts that might surprise you - particularly when you use them in the Where clause in a query.

```
select cast('1004,1009,1010' as signed integer), cast('1004,abc' as signed integer);
select cast('123,456' as signed integer), cast('123xyz' as signed integer);
select cast(234.98 as signed integer), cast('234.98' as signed integer);
```

CONVERT()

```
select convert(123.567 , char);
+------------------------+
| convert(123.567 , char) |
+------------------------+
| 123.567                |
+------------------------+

select convert(123.567 , char(2));
+---------------------------+
| convert(123.567 , char(2)) |
+---------------------------+
| 12                        |
+---------------------------+
1 row in set, 1 warning (0.00 sec)
Warning (Code 1292): Truncated incorrect CHAR(2) value: '123.567'
```

The types you can cast/convert to include:

- `CHAR, CHAR(n)    ( not varchar)`
- `Decimal,  Decimal(m),  Decimal(m, n)`
- `Signed integer, Signed  int`
- `Unsigned integer, Unsigned int`
- `Date`
- `Datetime`
- `Time`
- `BINARY[(N)]`

3. Format

FORMAT takes a number and formats it. The result is a string. The second argument is the number of digits to display after the decimal.

Demo 05: **select Format(1234.5678, 3)**

```
+----------------------+
| Format(1234.5678, 3) |
+----------------------+
| 1,234.568            |
+----------------------+
```

Demo 06: **select Format(1234.5678, 0);**

```
+----------------------+
| Format(1234.5678, 0) |
+----------------------+
| 1,235                |
+----------------------+
```

Demo 07: **select Format(1234.5678, 8);**

```
+----------------------+
| Format(1234.5678, 8) |
+----------------------+
| 1,234.56780000       |
+----------------------+
```

Demo 08: Combining functions

```
select  lpad(Format(1234.5678, 8), 20, ' ') as result ;
select  lpad(Format(1.5678,    8), 20, ' ') as result ;
select  lpad(Format(0.5678,    8), 20, ' ') as result ;
select  lpad(Format(-4.5678,   8), 20, ' ') as result ;
```

Each query gets its own result- I have skipped the feedback lines here. The Format function builds the string with the requested number of digits after the decimal and the Lpad sets the column width

```
+----------------------+
| result               |
+----------------------+
|       1,234.56780000 |
+----------------------+

+----------------------+
| result               |
+----------------------+
|           1.56780000 |
+----------------------+

+----------------------+
| result               |
+----------------------+
|           0.56780000 |
+----------------------+

+----------------------+
| result               |
+----------------------+
|          -4.56780000 |
+----------------------+
```

The default display for the command line client includes a space at each edge of the cell and sets the cell widths to match the widest data value - and the column alias. Note the wider column here since I did not supply a more appropriate column alias.

```
select  lpad(Format(1234.5678, 8), 20, ' ');
+-------------------------------------+
| lpad(Format(1234.5678, 8), 20, ' ') |
+-------------------------------------+
|       1,234.56780000                |
+-------------------------------------+
```

Demo 09: You can create all kinds of problems with this. The numeric value is 1234.5678. But Format gave me a string and the lpad did string processing.

```
select lpad(Format(1234.5678, 8), 2, ' ') as R ;
+------+
| R    |
+------+
| 1,   |
+------+
```