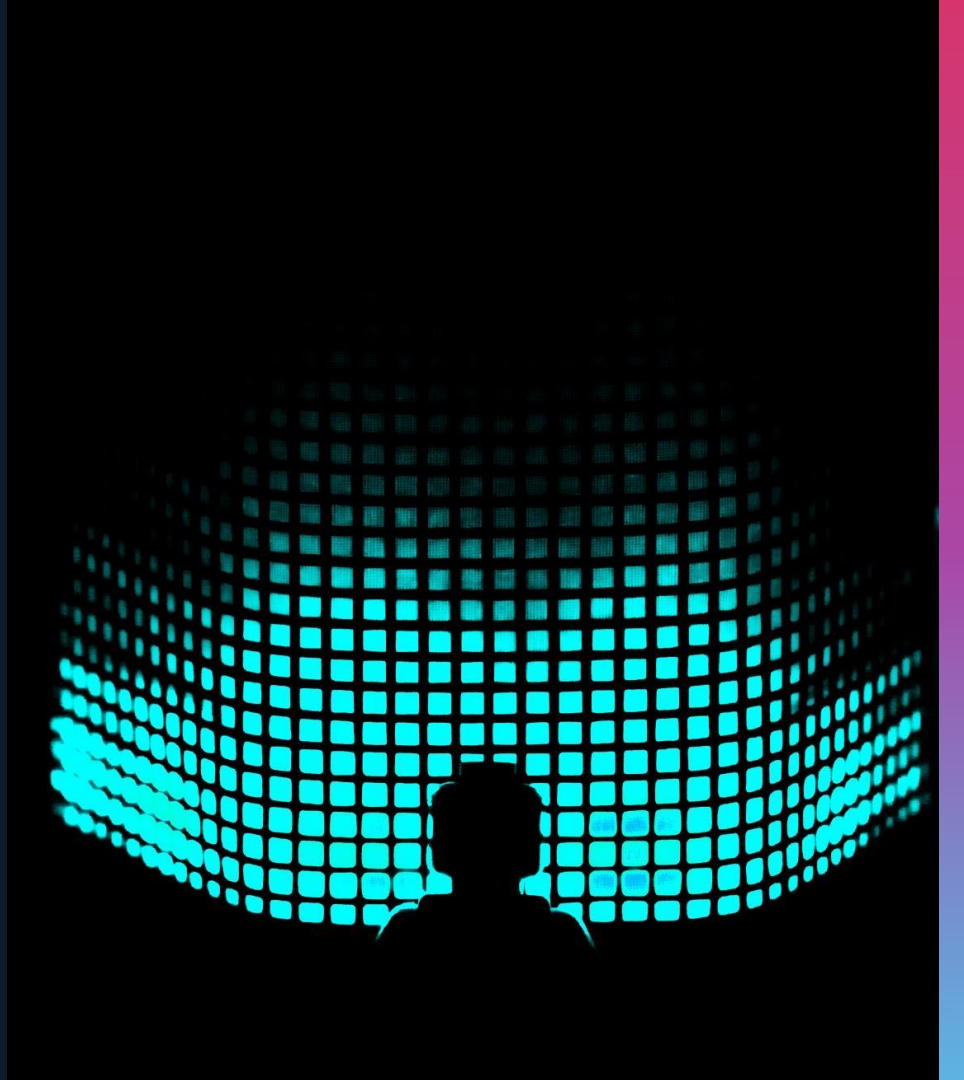

WeCloudData
SQL Fundamentals
Kick off your career in data science & analytics

Module 4:
Functions

Learning Objectives

In this module, we will continue to learn how to write queries in SQL. Specifically, we will share with you:

- SQL functions



SQL Functions

Agenda.

SQL Terminology

Functions

Here are some basic SQL **functions** to get you started:

- String functions
  - Lower/Upper
  - Trim
  - Substr
  - Like
  - Replace
- Datetime functions

String Functions: Lower/Upper

Functions

```
-- Convert CustomerName to uppercase
select CustomerName,
       upper(CustomerName)
from superstore.customer
limit 15;
```

CustomerName	upper(CustomerName)
Tamara Dahlen	TAMARA DAHLEN
Bill Donatelli	BILL DONATELLI
Greg Guthrie	GREG GUTHRIE
Trudy Brown	TRUDY BROWN
Joni Sundaresam	JONI SUNDARESAM
Jack O'Briant	JACK O'BRIANT
Jonathan Doherty	JONATHAN DOHERTY
Andrew Allen	ANDREW ALLEN

Syntax

```
SELECT lower(column1)
FROM db.table_name;

SELECT upper(column1)
FROM db.table_name;
```

NOTE:

- lower() ➜ Change the string column to lowercase
- upper() ➜ Change the string column to uppercase

String Functions: Trim

Functions

```
-- Remove leading and trailing
-- double quotes from productname

select ProductName,
       trim(both '"' from ProductName)
              as ProductName_trim
from superstore.product
where locate('"', ProductName) > 0
order by length(ProductName);
```

ProductName	ProductName_trim
"Advantus Employee of the Month Certificate Frame, 11...	Advantus Employee of the Month Certificate Fra...

Syntax

```
SELECT column1,
       trim(column2),
       trim(both ‘“’ from column3)
FROM db.table_name;
```

NOTE:

- trim() → By default, remove leading and trailing spaces

String Functions: Trim (Cont'd)

Functions

String Function: Trim (Syntax)	Explanation
trim()	• By default, remove leading and trailing spaces
trim(BOTH 'xx' FROM 'xxbarxx')	• Remove both the leading and trailing str ○ Returns 'bar'
trim(LEADING 'xx' FROM 'xxbarxx')	• Returns 'barxx'
trim(TRAILING 'xx' FROM 'xxbarxx')	• Returns 'xxbar'

String Functions: Substr

Functions

```
-- Find all orders that have
-- air shipmode

select OrderID,
       ProductID,
       ShipMode
from superstore.orders
where substr(trim(ShipMode), -3)='Air';
```

OrderID	ProductID	ShipMode
17024	778385	Regular Air
15808	284312	Regular Air
29537	681809	Express Air
38118	284312	Regular Air
69	213268	Regular Air
16768	681809	Regular Air
69	115501	Express Air

Syntax

```
SELECT column1,
       substr(column3, pos),
       column5
FROM db.table_name;

SELECT column1,
       substr(column3, pos, len),
       column5
FROM db.table_name;
```

NOTE:

- substr(string, start, length) ➔ Extracts a substring from a string
- substr('Data Science Toolbox', 14) ➔ 'Toolbox'
- substr('Data Science Toolbox', 1, 12) ➔ 'Data Science'
- substr('Data Science Toolbox', 6, 7) ➔ 'Science'

String Functions: Like

Functions

```
-- How many product in
-- the product table is from
-- the Belkin brand
select count(*)
from superstore.product
where lower(ProductName) like 'belkin%';
```

count(*)
3

```
-- Example #7
# Select all "Advantaus" related products

select *
from superstore.product
where lower(ProductName) like '%advantus%';
```

Syntax

```
SELECT *,
FROM db.table_name
WHERE column5 LIKE sql_str_pattern;
```

SQL String Pattern (sql_str_pattern):

- % ➔ Represents zero, one, or multiple characters
- _ ➔ Represents a single character

Examples:

- LIKE ‘a%’ ➔ Finds any values that start with ‘a’
- LIKE ‘%a’ ➔ Finds any values that end with ‘a’
- LIKE ‘_r%’ ➔ Finds any values that have ‘r’ in the second position

ProductID	ProductName	ProductCateg...	ProductSubCateg...	ProductContain...	ProductBaseMar...
213268	"Advantus Employee of the Month Certificate Frame, 11...	Furniture	Office Furnishings	Small Pack	0.44

String Functions: Replace

Functions

```
-- Remove the double quotes in productname
-- and create a new product_new table
drop table if exists superstore.product_new;
create table superstore.product_new as
select ProductID,
       REPLACE(ProductName, '"', '') as ProductName,
       ProductCategory,
       ProductSubCategory,
       ProductContainer,
       ProductBaseMargin
from superstore.product;

-- double check if double quotes
-- have been correctly replaced
select ProductID, ProductName
from superstore.product_new
where lower(ProductName) like '%belkin%';
```

ProductID	ProductName
213268	Advantus Employee of the Month Certificate Fra…

Syntax

```
SELECT column1,
       replace(str, from_str, to_str),
FROM db.table_name;
```

Replace Usage:

- Returns the string *str* with all occurrences of the string *from_str* replaced by the string *to_str*
- replace() performs a case-sensitive match when searching for *from_str*

Examples:

- SELECT replace('Data Science Toolbox', 'Toolbox', 'Course') → Returns 'Data Science Course'

Date and Time Functions

Functions

```
-- How many years of transactions
-- does the orders table has?

select distinct YEAR(OrderDate)
from superstore.orders;
```

YEAR(OrderDate)
2009
2011
2010
2012

Syntax

- date()
- day()
- year()
- month()
- dayofmonth()
- dayofweek()
- minute()
- hour()
- now()

MySQL Date and Time Function Reference:

- https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html

More DATE AND TIME Examples

Functions

```
-- Lab #1
#  Which order made in 2009 have the highest sales? (HINT: use order by)
select OrderID, ProductID,
       OrderDate, Sales
from superstore.orders
where year(OrderDate) = 2009
order by Sales desc;


-- Lab #2
# Which orders made in 2009 have the highest order quantity?
select OrderID, ProductID,
       OrderDate, Sales,
       OrderQuantity
from superstore.orders
where year(OrderDate) = 2009
order by OrderQuantity desc;

select * from superstore.product;


-- Lab #3
# Which day of week was the biggest sales order made?

select OrderID, ProductID,
       OrderDate,
       dayofweek(OrderDate) as dom,
       Sales,
       OrderQuantity
from superstore.orders
order by Sales desc;
```