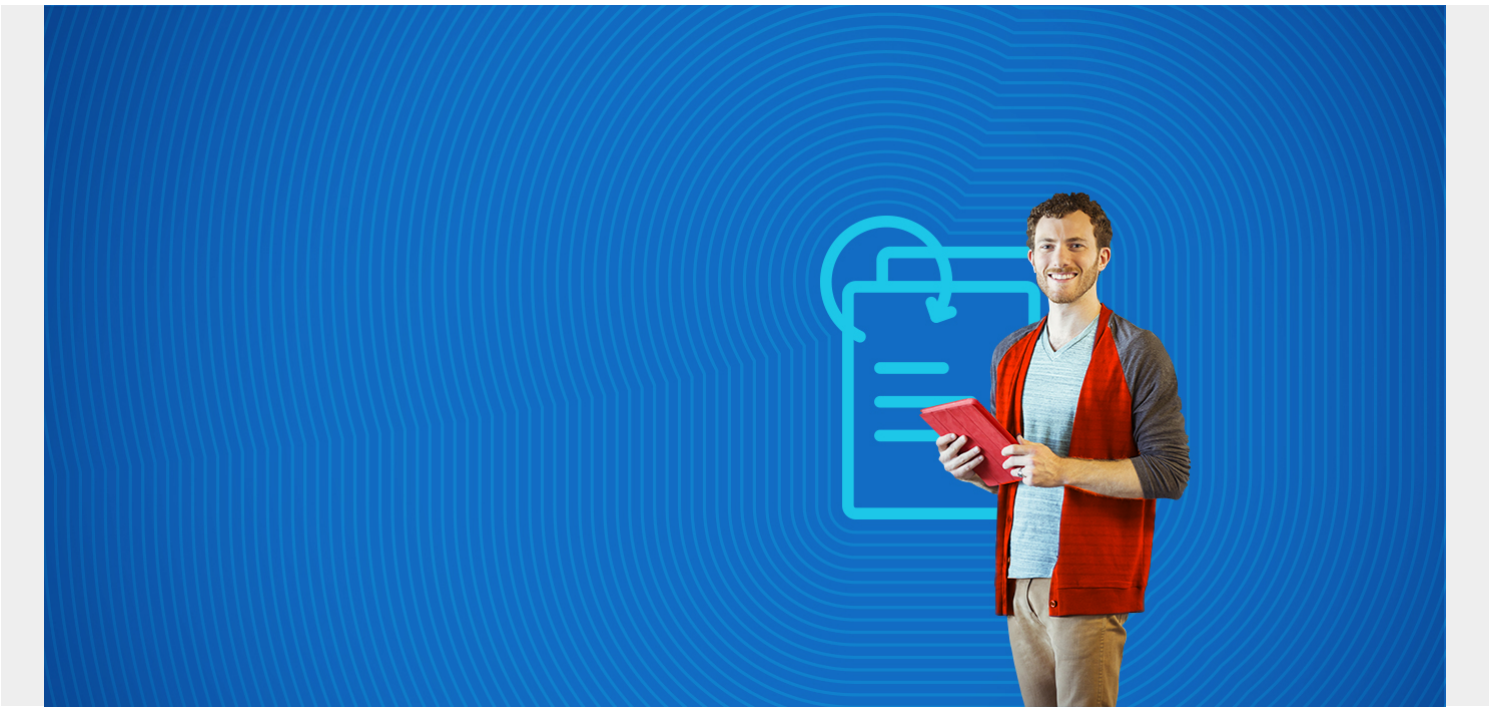
SNOWFLAKE SQL AGGREGATE FUNCTIONS & TABLE JOINS



In this article, we explain how to use aggregate functions with Snowflake.

(This article is part of our Snowflake Guide. Use the right-hand menu to navigate.)

What are aggregate functions?

Aggregate functions are those that perform some calculation over all the rows or subsets of rows in a table.

For example, the simplest aggregate function is **count()**. You could count all the customers in a table using count(*) with no group or where clause. The * tells Snowflake to look at all columns, but you could have put just one column as it means the same thing.

```
select count(*) from orders
```

But if you want to count orders over some subset you could, for example, count customers by order type:

```
select ordertype, count(*) from orders
group by ordertype;
```

Create some sample data

Let's create some sample data in order to explore some of these functions. Log into Snowflake and click the **Create Database** button to create a database called **inventory**. Next, open the worksheet editor and paste in these two SQL commands:

```
CREATE TABLE customers
  (
     customernumber     varchar(100) PRIMARY KEY,
    customername varchar(50),
    phonenumber varchar(50),
    postalcode varchar(50),
    locale varchar(10),
    datecreated date,
    email varchar(50)
  );


CREATE TABLE orders
  (
     customernumber     varchar(100) PRIMARY KEY,
    ordernumber varchar(100),
    comments varchar(200),
    orderdate date,
    ordertype varchar(10),
    shipdate date,
discount number,
quantity int,
    productnumber varchar(50)
)
```

Then paste in this data. The data looks like this:

```
insert into customers
(customernumber,customername,phonenumber,postalcode,locale,datecreated,email)
values ('ee56d97a-fcaa-11ea-
ab7a-0ec120e133fc','zopvxqhwocrtsonemrcf','3119110','vqlx','','2020-09-22','m
nst@yoaq.com');

insert into orders
(customernumber,ordernumber,comments,orderdate,ordertype,shipdate,discount,qu
antity,productnumber) values ('ee56d97a-fcaa-11ea-
ab7a-0ec120e133fc','ee56d97b-fcaa-11ea-
ab7a-0ec120e133fc','shsyuaraxxftdzooafbg','2020-09-22','sale','2020-10-01','0
.7751890540939359','40','ee56d97c-fcaa-11ea-ab7a-0ec120e133fc');
```

Joining tables

The customer and orders tables are related by order number. Obviously you would need to bring them together in one-set when you need both customer and order data together. You do this with a **join**, which creates that set temporarily

You join the two tables on the column element customer number. Note that:

- We use **as** to create an alias to abbreviate the table names to make it easier to type.

- We use **join** instead of **inner join**. (Other tutorials often add **inner join** but it just confuses things when they are the same thing. They often write this, too, to contrast that with a left-hand, right-hand, or outer join which are like cartesian products, i.e. tack each of n orders onto each of m customers thus creating a set of n*m rows.)

```
select c.customernumber, c.customername, o.ordernumber,  c.datecreated,
o.orderdate,  o.shipdate from customers as c
join orders as o on c.customernumber = o.customernumber;
```

Standard deviation

Let's calculate the standard deviation in shipping times. We do this in three steps:

1. Join customer and order tables
2. Use the **datediff()** function to calculate the shipping time, meaning how long the customer must wait.
3. Each outer query refers to an inner query by wrapping it in parentheses . So, the query is built up in stages.

Here is the complete query. See below to see how it is broken down.

```
select
  avg(shiptime),
  stddev_pop(shiptime)
from
  (
    select
      customernumber,
      customername,
      orderdate,
      shipdate,
      datediff(days, orderdate, shipdate) as shiptime
    from
      (
        select
          c.customernumber,
          c.customername,
          o.ordernumber,
          c.datecreated,
          o.orderdate,
          o.shipdate
        from
          customers as c
          join orders as o on c.customernumber = o.customernumber
      )
    order by
      shiptime desc
  )
```

We build up the query in stages. Start at bottom (aka innermost) query and work upwards:

1. Join the customer and orders table so that we can have the customer and order details in one set so we can list both. (We could have skipped this step since we are only using the orders table.)

```
(
  select
    c.customernumber,
    c.customername,
    o.ordernumber,
    c.datecreated,
    o.orderdate,
    o.shipdate
  from
    customers as c
    join orders as o on c.customernumber = o.customernumber
)
```

2. Calculate the shipping time using the datediff() function

```
select
  count(*),
  datediff(days, orderdate, shipdate) as shiptime
from
  orders
group by
  shiptime
order by
  shiptime
```

3. Calculate the standard deviation over the shipping times:

```
select avg(shiptime), stddev_pop(shiptime) from (step b)
```

Here are the results:

```
AVG(SHIPTIME)   STDDEV_POP(SHIPTIME)
8.539063        3.512038155
```

Discrete percentile

The 95th percentile means to show 95% of the population. That's a common statistic as data outside that range is generally considered outliers.

Here we show how to calculate the 25th percentile:

```
select customernumber ,  PERCENTILE_disc( 0.25 ) within group (order by
quantity)
```

```
from orders
  where customernumber = '5d2b742e-fcaa-11ea-ab7a-0ec120e133fc'
  group by customernumber
  order by customernumber
```

Results in:

```
CUSTOMERNUMBER  PERCENTILE_DISC( 0.25 ) WITHIN GROUP (ORDER BY QUANTITY)
5d2b742e-fcaa-11ea-ab7a-0ec120e133fc    9
```

Do a check and you can see that order quantities are 92, 55, and 9. So the only one in the bottom 25% percentile is 9.

```
select quantity from orders
where customernumber = '5d2b742e-fcaa-11ea-ab7a-0ec120e133fc'
order by quantity desc;
```

Here are the results:

```
QUANTITY
92
55
9
```

listagg

The **listagg** function lists orders by customer in an array, putting them into another format that you could use into a where clause that calls for a list of elements.

```
select listagg(ordernumber, '|')
from orders
where customernumber = '5d2b742e-fcaa-11ea-ab7a-0ec120e133fc'
```

Here are the results:

```
LISTAGG(ORDERNUMBER, '|')
5d2b742f-fcaa-11ea-ab7a-0ec120e133fc|5d2b7431-fcaa-11ea-
ab7a-0ec120e133fc|5d2b7433-fcaa-11ea-ab7a-0ec120e133fc
```

When you run queries, you should cross check them with other queries to double check your work. Here we list customer numbers straight up and down in rows.

```
select orderumber from orders
where where customernumber = '5d2b742e-fcaa-11ea-ab7a-0ec120e133fc'
```

mode

The **mode()** function shows the most frequent values:

```
select mode(quantity)
from orders
```

Results in:

```
MODE(QUANTITY)
13
```

Additional resources

For more tutorials like this, explore these resources:

- BMC Machine Learning & Big Data Blog
- How To Import Amazon S3 Data to Snowflake
- Snowflake Window Functions: Partition By and Order By
- AWS Guide, with 15 articles and tutorials
- Amazon Braket Quantum Computing: How To Get Started