Table of Contents

I am putting this discussion of Count in a separate document because Count has some useful features different than Max, Min, Sum, and Avg.

1. Count(*)

The expression count(*) will return the number of rows in the table expression. If the table is empty- has no rows- count(*) will return 0. If there are rows, count(*) will tell you how many rows are in the table.

Demo 01: Number of rows in the customer table

```
Select COUNT(*) As "Number of customers"
From customer_customers;
```

2. Count(expression)

Count(expression) will determine the number of rows in the table expression where that expression is not null.

Demo 02: Using the Count aggregate function with a column attribute.
At the time this query was run, we had 34 customers- all customers have a customer_id value since that is the primary key for this table; 33 of those customers have a value for first name.

```
select
  COUNT(customer_id) As "Number of customers"
, COUNT(customer_name_first) As "Number with a first name"
from customer_customers;
```

```
Number of customers Number with a first name
------------------- ------------------------
                 34                       33
```

If the table is empty, or if all of the rows have a null for the expression, count(expression) will return 0. Remember that Max, Min, Sum, and Avg returned a null in that case. If the table expression is empty, we know how many rows it has- 0.

Demo 03: Currently the largest salary value in the employee table is 120000. If I run the following query filtering for salary values equal to 200000. I get an empty set

```
select salary
from emp_employees
where salary = 200000;
```

```
mo rows selected
```

But if I do the count and sum of those salaries I get a count of 0 and a sum of null.

```
select sum(salary) as SumSalary, count(salary) as CountSalary
from emp_employees
where salary = 200000;
```

```
SumSalary CountSalary
---------- -----------
                    0
1 row selected
```

Some people think this does not make a lot of sense, but that is the way these functions work.

You will sometimes hear people suggest that you use an expression such as count(1) to count rows, or count(customer_id) with the customer table to count rows - customer_id is the pk and is never null. Those will work but there is no advantage to doing this. You could count any non null column- such as Count(customer_name_last) but that is confusing and someone might have changed that column to be a nullable column without checking all queries that use that table. Use a sensible column for count- if you are supposed to count customers, use count(customer_id); if you are supposed to count products, use count(prod_id); if you are supposed to count employees, use count(emp_id). But also see the next section of count(distinct…)

Count(1) works because 1 is a numeric literal- but you could use count (0) or count(3.14159) or count('Hi There') the same way since they are all literals. (If you are working at a job and your boss says to use count(1) instead of count(*), or if that is a company standard- go ahead. it won't hurt the query result.)

The count function requires exactly one argument which can be an expression

Demo 04: Suppose we wanted to know how many customers we have and how many have a last name starting with the letters 'Mor'.

```
select Count(customer_name_last) as CustCount
, count(case when customer_name_last like 'Mor%' then 1 end) as CustNameMorX
from customer_customers;
```

```
CUSTCOUNT CUSTNAMEMORX
--------- ------------
       34            5

1 row selected.
```

3. Count(Distinct expression)

Suppose we want to know how many products we have currently on order. It is important to realize that a simple request like this could have different interpretations. Suppose we have only these rows in our order details table.

Order_id	Line_Item_id	Prod_id	Quantity	Quoted_price
1	1	P12	2	5.00
1	2	T20	4	12.00
1	3	R67	1	25.00
2	1	P12	2	5.00
3	1	P12	8	6.50
3	2	R67	5	25.00

The question "how many products do we have currently on order ?" could be interpreted to mean "how many line items do we have ?" ; in this example we have 6 line items- we could answer this by using Count(*). The question could also be interpreted to mean "what is the total quantity of items that we have on order?"; this would be 2 + 4 + 1 + 2 + 8 +5 = 22 – for that use the function Sum (quantity). Another interpretation is "how many different products do we have on order "; this would be products P12, R67, and T20; we have three different products currently on order. This is answered using the Count (Distinct prod_id) function.

Demo 05: Using count distinct . We have 184 rows in the order headers table and 34 different products have been ordered.

```
select count(distinct prod_id) as CountDistinctProducts
, count(*) as CountProducts
from oe_orderDetails;
```

```
CountDistinctProducts CountProducts
--------------------- -------------
                   34           184
```

Note that the use of this keyword Distinct is different in syntax and meaning than the use of Distinct with a Select statement to return only unique rows.

When you add Distinct **inside** the argument list to an **aggregate** function, the function will determine the answer by using only unique values.

Demo 06: How many orders do we have? We can count the order ids in the order headers table.

```
select count(order_id) as "CountOrders"
from oe_order_headers;
```

```
CountOrders
-----------
         97
```

Demo 07: If we count the order id in the order detail tables, we count any order with multiple order lines more than once. Use distinct to count each order id only once.

```
select count( order_id) as "CountOrders"
, count(distinct order_id) as "CountDistinctOrders"
from oe_orderDetails;
```

```
CountOrders CountDistinctOrders
----------- -----------
        184          94
```

If you are wondering about the value 97 in demo 06 and the value 94 in demo 07, we have three orderheaders rows which have no detail rows.. It is a business decision if we should count those as Orders- they have an order id but there is nothing ordered.

Demo 08: How many customers do we have with orders? We can count the distinct customer ids in the order headers table. If we count customer_id, then we count customer with multiple orders multiple times. We have only 34 customers.

```
select count(distinct customer_id), count(customer_id)
from oe_order_headers;
```

```
UsingDistinct WithoutDistinct
----------- -----------
         21          97
```

Suppose we want to find out how many different combinations of customers and shipping modes we have. I am not going to count the orders where there is no info on the shipping modes.

This does not work

```
select count( distinct customer_id, shipping_mode)
from oe_order_headers
where shipping_mode is not null
;
```

```
ORA-00909: invalid number of arguments
```

Demo 09: But I can run this query to see the distinct combinations of customer ids and shipping modes

```
select distinct customer_id, shipping_mode_id
from oe_order_headers
where shipping_mode_id is not null;
```

```
    customer_id shipping_mode
----------- -------------
     400300 USPS1
     401250 FEDEX1
     401250 FEDEX2
     401890 USPS1
     402100 USPS1
     403000 FEDEX1
     403000 UPSGR
     403000 USPS1
. . .
```

Demo 10: And then I can put that into a CTE to get the count

```
With CTE as (
   select distinct customer_id, shipping_mode_id
   from oe_order_headers
   where shipping_mode_id is not null
)
select  COUNT(*)
from CTE;
```

```
  COUNT(*)
----------
        33
```

Demo 11: We have about 100 rows in the order headers table. We want to see how many different months are included in those orders. The To_Char function is use to extract just the year and month from the order date.

```
select count( distinct To_char( order_date, 'YYYY-MM')) as "number of months"
from oe_orderHeaders
;
```

```
number of months
----------------
              12
```

Demo 12: Use a CTE to make it easier to work with the aggregates. The CTE does the Distinct processing.

```
With orderdates as (
   select distinct to_char(order_date, 'YYYY-MM')as YrMnth
   from oe_orderHeaders
   )
select count (YrMnth) as "number of months"
, min(YrMnth) as "earliest month"
, max(YrMnth) as "latest month"
from orderdates;
```

```
number of months earlies latest
---------------- ------- -------
              12 2015_04 2016-05
```

You can use Distinct with any of the aggregates- but don't use it if you really don't need it. If you are looking for the price of the most expensive item we sell, Max(price) will return the same value as Max(Distinct price).

I have never seen a good business reason for calculating sum(distinct prod_list_price) or for calculating sum(prod_list_price) from the products table.

4. Count with Inner and with Outer Joins

Suppose we join the tables products and order details to see which products have been ordered. If we do this with an inner join then we get only products which have been ordered and if we use a left join (Products Left Join order_details) then we get products with orders and products without orders. This means you need to take more care with aggregate functions. The Count function is the easiest function for illustrating this point.

Demo 13: Inner join

```
select PR.prod_id, OD.prod_id, PR.catg_id
from prd_products  PR
join oe_orderDetails OD on Pr.prod_id = OD.prod_id
where pr.prod_id between 1140 and 1150
order by PR.prod_id;
```

```
    PROD_ID     PROD_ID CATG_ID
---------- ---------- -------
       1140       1140 PET
       1141       1141 PET
       1141       1141 PET
       1141       1141 PET
       1141       1141 PET
       1141       1141 PET
       1141       1141 PET
       1141       1141 PET
       1141       1141 PET
       1141       1141 PET
       1141       1141 PET
       1150       1150 PET
       1150       1150 PET
       1150       1150 PET
       1150       1150 PET
       1150       1150 PET
16 rows selected
```

Demo 14: Outer join- we get 2 more rows - products 1142, 1143 with no orders

```
select PR.prod_id, OD.prod_id, PR.catg_id
from prd_products  PR
left join oe_orderDetails OD on Pr.prod_id = OD.prod_id
where pr.prod_id between 1140 and 1150
order by PR.prod_id;
```

```
    PROD_ID     PROD_ID CATG_ID
---------- ---------- -------
      1140       1140 PET
      1141       1141 PET
      1141       1141 PET
      1141       1141 PET
      1141       1141 PET
      1141       1141 PET
      1141       1141 PET
      1141       1141 PET
      1141       1141 PET
      1141       1141 PET
      1141       1141 PET
      1141       1141 PET
      1142            PET
      1143            PET
      1150       1150 PET
      1150       1150 PET
      1150       1150 PET
      1150       1150 PET
      1150       1150 PET
18 rows selected
```

Demo 15: using the inner join and aggregates

```
select count(*) as "RowCount"
, count(distinct order_id) as "OrderCount"
, count(distinct PR.prod_id) as "PR_DistProdCount"
, count(distinct OD.prod_id) as "OD_DistProdCount"
from prd_products  PR
join oe_orderDetails OD on PR.prod_id = OD.prod_id
where pr.prod_id between 1140 and 1150
;
```

```
   RowCount  OrderCount PR_DistProdCount OD_DistProdCount
----------- ----------- ---------------- ----------------
         16          11                3                3
```

Demo 16: using the outer join and aggregates

```
select count(*) as "RowCount"
, count(distinct order_id) as "OrderCount"
, count(distinct PR.prod_id) as "PR_DistProdCount"
, count(distinct OD.prod_id) as "OD_DistProdCount"
from prd_products  PR
left join oe_orderDetails OD on PR.prod_id = OD.prod_id
where pr.prod_id between 1140 and 1150
;
```

```
   RowCount  OrderCount PR_DistProdCount OD_DistProdCount
----------- ----------- ---------------- ----------------
         18          11                5                3
```

The differences between the results for Demo 13 and 14.:

RowCount With the outer join we get two more rows because we include the rows for the two products with no orders

OrderCount These values are the same, we are counting the order_id values which occur only in the order details table. Since the rows for the products with no orders have a null in that column, they do not get counted.

PR_DistProdCount With this column we are counting the product id value from the products table. With the outer join we do count the product id for the two products with no orders.

OD_DistProdCount. With this column we are counting the product id value from the order details table. Since the rows for the products with no orders have a null in that column, they do not get counted

5. Summary

Count (*) counts rows.

Count (col) counts the values for that column, and does not count Nulls.

Count will always return a numeric integer answer- if the table is empty or there are no matching rows, it will return 0. You don't need to use coalesce with count.

Sum, Avg, Max and Min will return nulls when there are no matching rows

Sum, Avg, Max and Min ignore nulls when doing their calculations

This is a point in the semester where it makes sense to realize that SQL is like every other computer system. It is built on a system of rules and it implements those rules. The rules it has will not always agree with what you think they should be- but it makes no sense to fight a compiler. You need to know the rules the system uses and decide if those rules make sense for the processing you need to do.