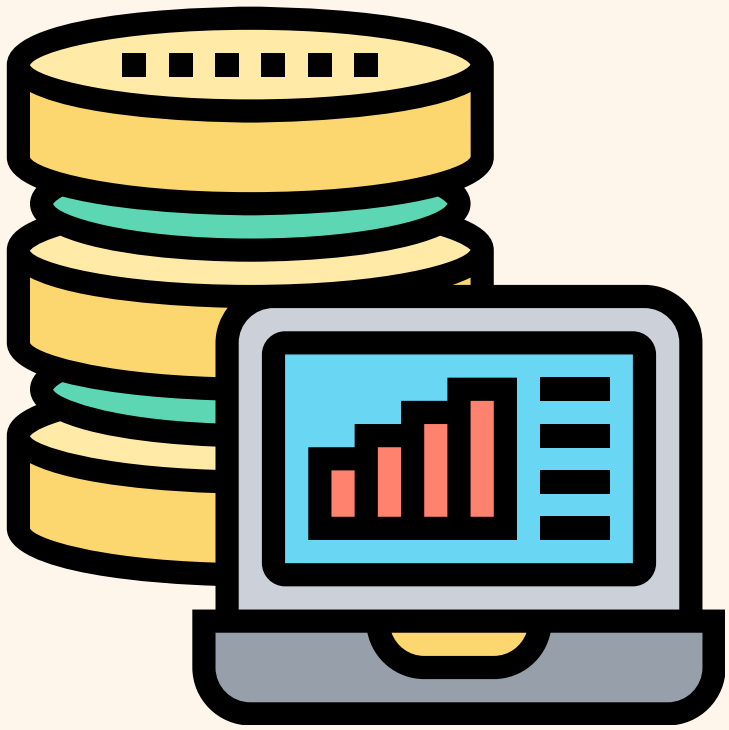
#56daysofSQL

Day 54
Window Functions
Part 4



Aditya Shenoy

#56daysofSQL

Aggregate Window Functions

- AVG()
- COUNT()
- MAX()
- MIN()
- SUM()

Aditya Shenoy

AVG()

- The **AVG()** window function returns the average value for the input expression values.
- The **AVG** function works with numeric values and ignores NULL values.

General Syntax

```
AVG( expression )
OVER ( [ PARTITION BY expr_list ]
       [ ORDER BY order_list ]  )
```

#56daysofSQL

Use Case Example of AVG() :

The following query uses the **AVG()** window function with the **PARTITION BY** clause to calculate the average sales for each car dealer in Q1.

```
SELECT dealer_id, sales,
AVG(sales) OVER(PARTITION BY dealer_id) as avgsales
FROM q1_sales;
```

Output :

dealer_id	sales	avgsales
1	19745	14357
1	19745	14357
1	8227	14357
1	9710	14357
2	16233	13925
2	16233	13925
2	9308	13925
3	15427	12368
3	12369	12368
3	9308	12368

10 rows selected (0.455 seconds)

avgsales
average of sales of each dealer

COUNT()

- The **COUNT()** window function counts the number of input rows.
- **COUNT(*)** counts all of the rows in the target table if they do or do not include nulls.
- **COUNT(expression)** computes the number of rows with non-NULL values in a specific column or expression.

General Syntax

```
COUNT( expression )
OVER ( [ PARTITION BY expr_list ]
       [ ORDER BY order_list ]  )
```

#56daysofSQL

Use Case Example of COUNT():

The following query uses the **COUNT (*)** window function to count the number of sales in Q1, ordered by dealer_id:

```
SELECT dealer_id, sales,
COUNT(*) OVER(ORDER BY dealer_id) as count
FROM q1_sales;
```

Output :

dealer_id	sales	count
1	19745	4
1	19745	4
1	8227	4
1	9710	4
2	16233	7
2	16233	7
2	9308	7
3	15427	10
3	12369	10
3	9308	10

10 rows selected (0.215 seconds)

Cumulates the total count of sales made by each dealer in Q1

MAX()

- The **MAX()** window function returns the maximum value of the expression across all input values.
- The **MAX** function works with numeric values and ignores NULL values.

General Syntax

```
MAX( expression )
OVER( [ PARTITION BY expr_list ]
      [ ORDER BY order_list ] )
```

#56daysofSQL

Use Case Example of MAX()

The following query uses the **MAX()** window function with the **PARTITION BY** clause to identify the employee with the maximum number of car sales in Q1 at each dealership:

```
SELECT emp_name, dealer_id, sales,
MAX(sales) OVER(PARTITION BY dealer_id) as max
FROM q1_sales;
```

Output :

emp_name	dealer_id	sales	max
Ferris Brown	1	19745	19745
Noel Meyer	1	19745	19745
Raphael Hull	1	8227	19745
Jack Salazar	1	9710	19745
Beverly Lang	2	16233	16233
Kameko French	2	16233	16233
Haviva Montoya	2	9308	16233
Ursa George	3	15427	15427
Abel Kim	3	12369	15427
May Stout	3	9308	15427

10 rows selected (0.402 seconds)

max shows the maximum sales made by each dealer

Aditya Shenoy

MIN()

- The **MIN()** window function returns the minimum value of the expression across all input values.
- The **MIN** function works with numeric values and ignores NULL values.

General Syntax

```
MIN( expression )
OVER( [ PARTITION BY expr_list ]
      [ ORDER BY order_list ] )
```

#56daysofSQL

Use Case Example of MIN()

The following query uses the **MIN()** window function with the **PARTITION BY** clause to identify the employee with the minimum number of car sales in Q1 at each dealership:

```
SELECT emp_name, dealer_id, sales,
MIN(sales) OVER(PARTITION BY dealer_id) as min
FROM q1_sales;
```

Output :

emp_name	dealer_id	sales	min
Ferris Brown	1	19745	8227
Noel Meyer	1	19745	8227
Raphael Hull	1	8227	8227
Jack Salazar	1	9710	8227
Beverly Lang	2	16233	9308
Kameko French	2	16233	9308
Haviva Montoya	2	9308	9308
Ursa George	3	15427	9308
Abel Kim	3	12369	9308
May Stout	3	9308	9308

10 rows selected (0.194 seconds)

min shows the minimum sales made by each dealer.

#56daysofSQL

SUM()

- The **SUM()** window function returns the sum of the expression across all input values.
- The **SUM** function works with numeric values and ignores NULL values.

General Syntax

```
SUM( expression )
OVER( [ PARTITION BY expr_list ]
      [ ORDER BY order_list ] )
```

#56daysofSQL

Use Case Example of SUM()

The following query uses the **SUM()** window function to total the amount of sales for each dealer in Q1:

```
SELECT dealer_id, emp_name, sales,
SUM(sales) OVER(PARTITION BY dealer_id) as `sum`
FROM q1_sales;
```

Output :

dealer_id	emp_name	sales	sum
1	Ferris Brown	19745	57427
1	Noel Meyer	19745	57427
1	Raphael Hull	8227	57427
1	Jack Salazar	9710	57427
2	Beverly Lang	16233	41774
2	Kameko French	16233	41774
2	Haviva Montoya	9308	41774
3	Ursa George	15427	37104
3	Abel Kim	12369	37104
3	May Stout	9308	37104

10 rows selected (0.198 seconds)

sum
sums up the sales of each dealer