

# Aggregate arithmetic functions

FUNCTIONS FOR MANIPULATING DATA IN SQL SERVER



**Ana Voicu**  
Data Engineer

# COUNT()

- Returns the number of items found in a group.

```
COUNT([ALL] expression)  
COUNT(DISTINCT expression)  
COUNT(*)
```

# COUNT() example

```
SELECT
    COUNT(ALL country) AS total_countries,
    COUNT(country) AS total_countries,
    COUNT(DISTINCT country) AS distinct_countries,
    COUNT(*) AS all_voters
FROM voters;
```

count_countries_all	count_countries	distinct_countries	all_voters
196	196	11	196

# SUM()

- Returns the sum of all values from a group.

```
SUM([ALL] expression)
```

```
SUM(DISTINCT expression)
```

# SUM() example

```
SELECT
    first_name,
    last_name,
    total_votes
FROM voters
WHERE total_votes = 153;
```

first_name	last_name	total_votes
Isabella	Roberts	153
Chase	Ward	153
Kendra	Ortega	153
Bruce	Moreno	153

```
SELECT
    SUM(ALL total_votes) AS tot_votes1,
    SUM(total_votes) AS tot_votes2,
    SUM(DISTINCT total_votes) AS dist
FROM voters
WHERE total_votes = 153;
```

tot_votes1	tot_votes2	tot_dis_votes
612	612	153

# MAX() and MIN()

```
MAX([ALL] expression)
```

```
MAX(DISTINCT expression)
```

- Returns the maximum value in the expression.

```
MIN([ALL] expression)
```

```
MIN(DISTINCT expression)
```

- Returns the minimum value in the expression.

# MAX() and MIN() example

```
SELECT
    MIN(rating) AS min_rating,
    MAX(rating) AS max_rating
FROM ratings;
```

```
| min_rating | max_rating |
|-----|-----|
| 1.0000    | 5.0000    |
```

# AVG()

- Returns the average of the values in the group.

```
AVG([ALL] expression)
AVG(DISTINCT expression)
```

```
SELECT
    AVG(rating) AS avg_rating,
    AVG(DISTINCT rating) AS avg_dist
FROM ratings;
```

```
| avg_rating | avg_dist |
|-----|-----|
| 3.184665  | 2.788461 |
```



# Grouping data

```
SELECT company,  
       AVG(rating) AS avg_rating  
FROM ratings  
GROUP BY company;
```

```
| company      | avg_rating |  
|-----|-----|  
| A. Morin    | 3.250000  |  
| Acalli     | 3.500000  |  
| Adi        | 3.000000  | ...
```

# Let's practice!

**FUNCTIONS FOR MANIPULATING DATA IN SQL SERVER**

# Analytic functions

FUNCTIONS FOR MANIPULATING DATA IN SQL SERVER



**Ana Voicu**  
Data Engineer

# FIRST\_VALUE()

```
FIRST_VALUE(numeric_expression)
  OVER ([PARTITION BY column] ORDER BY column ROW_or_RANGE frame)
```

- Returns the first value in an ordered set.

**OVER** clause components

Component	Status	Description
PARTITION by column	optional	divide the result set into partitions
ORDER BY column	mandatory	order the result set
ROW_or_RANGE frame	optional	set the partition limits

# LAST\_VALUE()

```
LAST_VALUE(numeric_expression)  
  OVER ([PARTITION BY column] ORDER BY column ROW_or_RANGE frame)
```

- Returns the last value in an ordered set.

# Partition limits

```
RANGE BETWEEN start_boundary AND end_boundary  
ROWS BETWEEN start_boundary AND end_boundary
```

Boundary	Description
UNBOUNDED PRECEDING	first row in the partition
UNBOUNDED FOLLOWING	last row in the partition
CURRENT ROW	current row
PRECEDING	previous row
FOLLOWING	next row

# FIRST\_VALUE() and LAST\_VALUE() example

```
SELECT
  first_name + ' ' + last_name AS name,
  gender,
  total_votes AS votes,
  FIRST_VALUE(total_votes)
  OVER (PARTITION BY gender ORDER BY total_votes) AS min_votes,
  LAST_VALUE(total_votes)
  OVER (PARTITION BY gender ORDER BY total_votes
        ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS max_votes
FROM voters;
```

name	gender	votes	min_votes	max_votes
-----	-----	-----	-----	-----
Michele Suarez	F	20	20	189
...	...	...	20	189
Marcus Jenkins	M	16	16	182
Micheal Vazquez	M	18	16	182

# LAG() and LEAD()

```
LAG(numeric_expression) OVER ([PARTITION BY column] ORDER BY column)
```

- Accesses data from a previous row in the same result set.

```
LEAD(numeric_expression) OVER ([PARTITION BY column] ORDER BY column)
```

- Accesses data from a subsequent row in the same result set.



# LAG() and LEAD() example

```
SELECT
  broad_bean_origin AS bean_origin,
  rating,
  cocoa_percent,
  LAG(cocoa_percent) OVER(ORDER BY rating ) AS percent_lower_rating,
  LEAD(cocoa_percent) OVER(ORDER BY rating ) AS percent_higher_rating
FROM ratings
WHERE company = 'Felchlin'
ORDER BY rating ASC;
```

bean_origin	rating	cocoa_percent	percent_lower_rating	percent_higher_rating
Grenada	3	0.58	NULL	0.62
Dominican Republic	3.75	0.62	0.58	0.64
Madagascar	3.75	0.64	0.74	0.65
Venezuela	4	0.65	0.74	NULL

# Let's practice!

**FUNCTIONS FOR MANIPULATING DATA IN SQL SERVER**

# Mathematical functions

FUNCTIONS FOR MANIPULATING DATA IN SQL SERVER



**Ana Voicu**  
Data Engineer

# ABS(numeric\_expression)

- Returns the absolute value of an expression.
- Is the non-negative value of the expression.

SELECT

ABS(-50.4 \* 3) AS negative,

ABS(0.0) AS zero,

ABS(73.2 + 15 + 8.4) AS positive;

negative	zero	positive
151.2	0	96.6

# SIGN(numeric\_expression)

- Returns the sign of an expression, as an integer:
  - -1 (negative numbers)
  - 0
  - +1 (positive numbers)

```
SELECT
```

```
    SIGN(-50.4 * 3) AS negative,
```

```
    SIGN(0.0) AS zero,
```

```
    SIGN(73.2 + 15 + 8.4) AS positive;
```

```
| negative | zero | positive |
|-----|-----|-----|
| -1.0    | 0    | 1.0      |
```

# Rounding functions

- `CEILING(numeric_expression)`
  - Returns the smallest integer greater than or equal to the expression.
- `FLOOR(numeric_expression)`
  - Returns the largest integer less than or equal to the expression.
- `ROUND(numeric_expression, length)`
  - Returns a numeric value, rounded to the specified length.

# Rounding functions example

```
SELECT  
    CEILING(-50.49) AS ceiling_neg,  
    CEILING(73.71) AS ceiling_pos;
```

ceiling_neg	ceiling_pos
-50	74

# Rounding functions example

```
SELECT
    CEILING(-50.49) AS ceiling_neg,
    FLOOR(-50.49) AS floor_neg,
    CEILING(73.71) AS ceiling_pos,
    FLOOR(73.71) AS floor_pos;
```

ceiling_neg	floor_neg	ceiling_pos	floor_pos
-50	-51	74	73



# Rounding functions example

```
SELECT
  CEILING(-50.49) AS ceiling_neg,
  FLOOR(-50.49) AS floor_neg,
  CEILING(73.71) AS ceiling_pos,
  FLOOR(73.71) AS floor_pos,
  ROUND(-50.493, 1) AS round_neg,
  ROUND(73.715, 2) AS round_pos;
```

ceiling_neg	floor_neg	ceiling_pos	floor_pos	round_neg	round_pos
-50	-51	74	73	-50.500	73.720

# Exponential functions

- `POWER(numeric_expression, power)`
  - Returns the expression raised to the specified power.
- `SQUARE(numeric_expression)`
  - Returns the square of the expression.
- `SQRT(numeric_expression)`
  - Returns the square root of the expression.
- **Keep in mind:** the type of the expression is *float* or can be implicitly converted to *float*.

# POWER() example

```
SELECT  
POWER(2, 10) AS pos_num,  
POWER(-2, 10) AS neg_num_even_pow,  
POWER(-2, 11) AS neg_num_odd_power,  
POWER(2.5, 2) AS float_num,  
POWER(2, 2.72) AS float_pow;
```

pos_num	neg_num_even_pow	neg_num_odd_pow	float_num	float_pow
1024	1024	-2048	6.3	6

# SQUARE() example

```
SELECT  
SQUARE(2) AS pos_num,  
SQUARE(-2) AS neg_num,  
SQUARE(2.5) AS float_num;
```

pos_num	neg_num	float_num
4	4	6.25

# SQRT() example

```
SELECT  
SQRT(2) AS int_num,  
SQRT(2.76) AS float_num;
```

int_num	float_num
1.4142135623731	1.66132477258361

# Let's practice!

**FUNCTIONS FOR MANIPULATING DATA IN SQL SERVER**

# Wrapping things up

FUNCTIONS FOR MANIPULATING DATA IN SQL SERVER



**Ana Voicu**  
Data Engineer

# Chapter 1: Choosing the appropriate data type

## Data types

- Numeric
- Date and time
- Character strings

## Data conversion

- Implicit
- Explicit



# Chapter 2: Manipulating time in SQL Server

- Functions returning system date and time
  - `GETDATE()`
- Functions returning date parts
  - `YEAR()` , `MONTH()` , `DAY()`
- Arithmetic operations on dates
  - `DATEADD()` , `DATEDIFF()`
- Validating if an expression is a date
  - `ISDATE()`

# Chapter 3: Working with strings

- Functions for positions
  - `CHARINDEX()` , `PATINDEX()`
- Functions for string transformation
  - `UPPER()` , `LOWER()` , `LEFT()` , `RIGHT()`
- Functions for manipulating groups of strings
  - `STRING_AGG()` , `STRING_SPLIT()`

# Chapter 4: Recognizing numeric data properties

- Aggregate arithmetic functions
  - `SUM()` , `MIN()` , `MAX()` , `AVG()`
- Analytical functions
  - `FIRST_VALUE()` , `LAST_VALUE()` , `LAG()` , `LEAD()`
- Mathematical functions
  - `ABS()` , `POWER()`

# Congratulations!

**FUNCTIONS FOR MANIPULATING DATA IN SQL SERVER**