

# T-SQL WINDOW FUNCTION DEEP DIVE

Kathi Kellenberger

@aunkathi

Kathi.kellenberger@linchpinpeople.com

<http://aunkathisql.com>

THE EXPERT'S VOICE® IN SQL SERVER

# Expert T-SQL Window Functions in SQL Server

MASTER THE MOST USEFUL ADDITION  
TO SQL IN OVER A DECADE

Kathi Kellenberger with Clayton Groom

Apress®

What are T-SQL window functions?

2005

- Ranking functions

- Window aggregates

2012 Enhancements

- Accumulating window aggregates

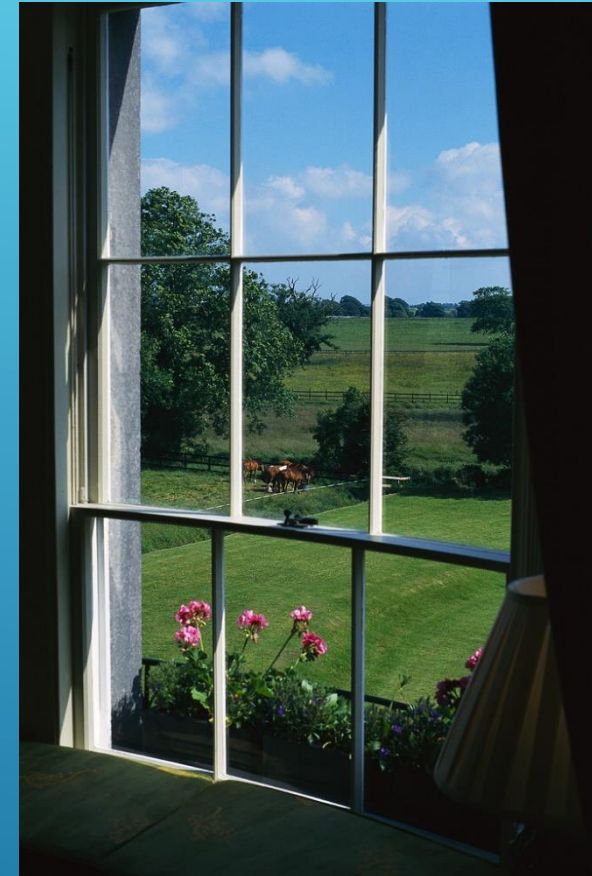
- Framing

- Offset functions

- Statistical functions

Performance considerations

# AGENDA



# WHAT ARE WINDOW FUNCTIONS?

Not OS, based on ANSI-SQL standards

Function performs over a SET (window of the results)

How to identify a window function

The OVER clause defines the window

## Where to put window functions

SELECT and ORDER BY clauses only

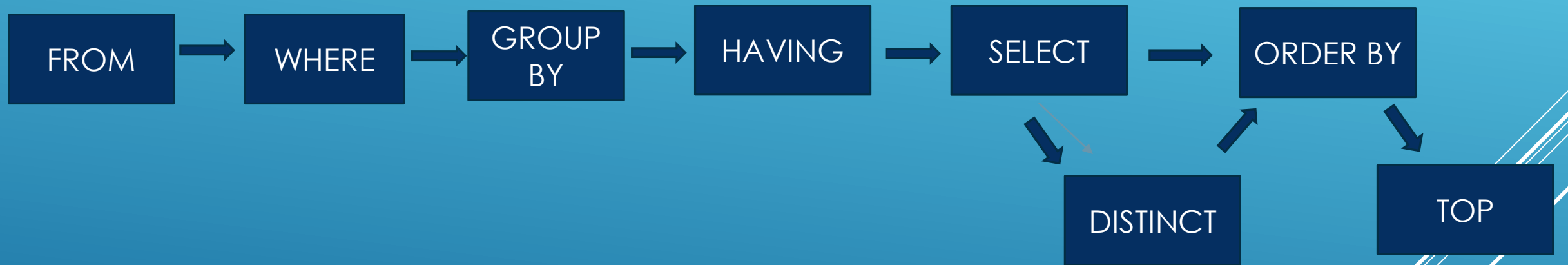
## Important!

The FROM, WHERE, GROUP BY  
and HAVING clauses operate BEFORE the window functions

## Partitions

NOT the same as GROUP BY

# ORDER OF OPERATIONS



# THE OVER CLAUSE

Always required

Defines the window

A set of the rows

ORDER BY

PARTITION BY



OVER

# PARTITION BY

Always supported  
Always optional  
Divides the window





# RANKING FUNCTIONS

Available since 2005

`ROW_NUMBER()`

A unique # over the window

`RANK()`

Deals with ties

`DENSE_RANK()`

Deals with ties

`NTILE()`

Divide rows into buckets




`ORDER BY` is required in `OVER` clause



# ROW\_NUMBER() EXAMPLE

CustomerID	OrderID	Total	ROW_NUMBER() OVER(Order by OrderID)
1	101	100	1
2	102	40	2
2	103	11	3
3	104	432	4
1	105	2000	5
1	106	300	6
4	107	674	7
5	108	76	8
4	109	234	9
4	110	889	10
5	111	234	11



# ROW\_NUMBER() EXAMPLE

CustomerID	OrderID	Total	ROW+NUMBER() OVER(PARTITION BY CustomerID Order by OrderID)
1	101	100	1
1	105	2000	2
1	106	300	3
2	102	40	1
2	103	11	2
3	104	432	1
4	107	674	1
4	109	234	2
4	110	889	3
5	108	76	1
5	111	234	2

DEMO 1

# Ranking functions



# WINDOW AGGREGATE FUNCTIONS

Your favorite aggregates with no GROUP BY

Add aggregate function to a non-aggregate query

Calculate over the window: the entire result set or partition

No ORDER BY

Partition by OR use the Empty Over Clause



# WINDOW AGGREGATE EXAMPLE

CustomerID	OrderID	TotalDue	SUM(TotalDue) OVER()
1	101	100	4990
1	102	2000	4990
1	103	300	4990
2	104	40	4990
2	105	11	4990
3	106	432	4990
4	107	674	4990
4	108	234	4990
4	109	889	4990
5	110	76	4990
5	111	234	4990

# PARTITION BY EXAMPLE

CustomerID	OrderID	TotalDue	SUM(TotalDue) OVER(Partition by CustomerID)
1	101	100	2400
1	102	2000	2400
1	103	300	2400
2	104	40	51
2	105	11	51
3	106	432	432
4	107	674	1797
4	108	234	1797
4	109	889	1797
5	110	76	310
5	111	234	310

DEMO 2

# Window aggregates

Several thin, parallel white lines are drawn diagonally across the bottom right corner of the slide, extending from the right edge towards the bottom left.



# 2012 ENHANCEMENTS

Accumulation aggregates

Even further define the  
window with FRAMING

Eight new functions

Offset

Statistical



# ACCUMULATING AGGREGATE EXAMPLE

CustomerID	OrderID	TotalDue	SUM(TotalDue) OVER(Order by OrderID)
1	101	100	100
2	102	40	140
2	103	11	151
3	104	432	583
1	105	2000	2583
1	106	300	2883
4	107	674	3557
5	108	76	3633
4	109	234	3867
4	110	889	4756
5	111	234	4990

DEMO 3

# Accumulating aggregates

Several thin, parallel white lines of varying lengths and slopes are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.

# FRAMING: ROWS AND RANGE

Further define the window  
Each row has its own window  
Supported for specific function types



# FRAMING: VOCABULARY

Term	Used for
ROWS	Positional operator used to define the frame
RANGE	Logical operator used to define the frame. Not fully supported. The default if ROWS is not specified
UNBOUNDED PRECEDING	The first row of the partition
UNBOUNDED FOLLOWING	The last row of the partition
CURRENT ROW	The row where the calculation is being performed

# FRAMING



# FRAMING





# FRAMING



# FRAMING



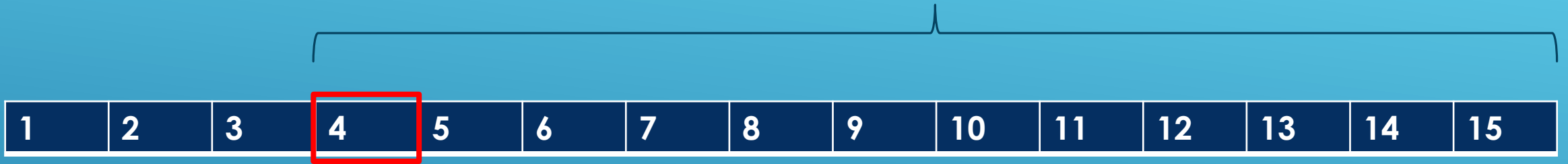
# FRAMING



ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

ROWS UNBOUNDED PRECEDING

# FRAMING



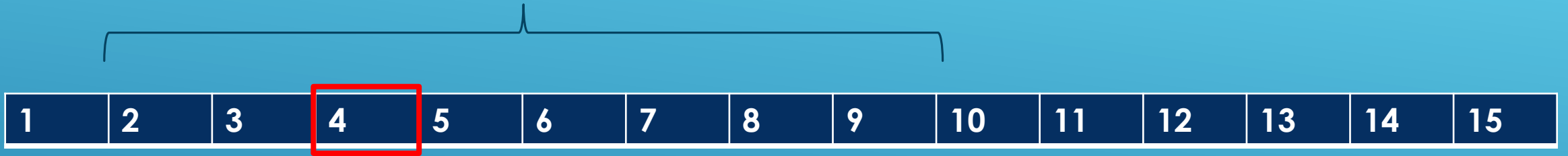
ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING

# FRAMING



ROWS BETWEEN 2 PRECEDING AND CURRENT ROW

# FRAMING



ROWS BETWEEN 2 PRECEDING AND 5 FOLLOWING

## DEMO 4

# Framing





# OFFSET FUNCTIONS

LAG()

Grab a column from a previous row

LEAD()

Grab a column from a later row

FIRST\_VALUE()

Grab a column from the first row

Supports framing

LAST\_VALUE()

Grab a column from the last row

Supports framing -- be careful!!



# LAG EXAMPLE


CustomerID	OrderID	TotalDue	LAG(TotalDue) OVER(ORDER BY OrderID)
1	101	100	NULL
2	102	40	100
2	103	11	40
3	104	432	11
1	105	2000	432
1	106	300	2000
4	107	674	300
5	108	76	674
4	109	234	76
4	110	889	234
5	111	234	889

# LEAD EXAMPLE

CustomerID	OrderID	TotalDue	LEAD(TotalDue) OVER(ORDER BY OrderID)
1	101	100	40
2	102	40	11
2	103	11	432
3	104	432	2000
1	105	2000	300
1	106	300	674
4	107	674	76
5	108	76	234
4	109	234	889
4	110	889	234
5	111	234	NULL


# FIRST\_VALUE EXAMPLE

CustomerID	OrderID	TotalDue	FIRST_VALUE (TotalDue) OVER(ORDER BY OrderID)
1	101	100	100
2	102	40	100
2	103	11	100
3	104	432	100
1	105	2000	100
1	106	300	100
4	107	674	100
5	108	76	100
4	109	234	100
4	110	889	100
5	111	234	100



# LAST\_VALUE EXAMPLE

CustomerID	OrderID	TotalDue	LAST_VALUE>TotalDue) OVER(ORDER BY OrderID)
1	101	100	234
2	102	40	234
2	103	11	234
3	104	432	234
1	105	2000	234
1	106	300	234
4	107	674	234
5	108	76	234
4	109	234	234
4	110	889	234
5	111	234	234



## DEMO 5

# Offset functions



# STATISTICAL FUNCTIONS

PERCENT\_RANK()

Relative rank

CUME\_DIST()

Cumulative distribution over a group of rows

PERCENTILE\_DISC()

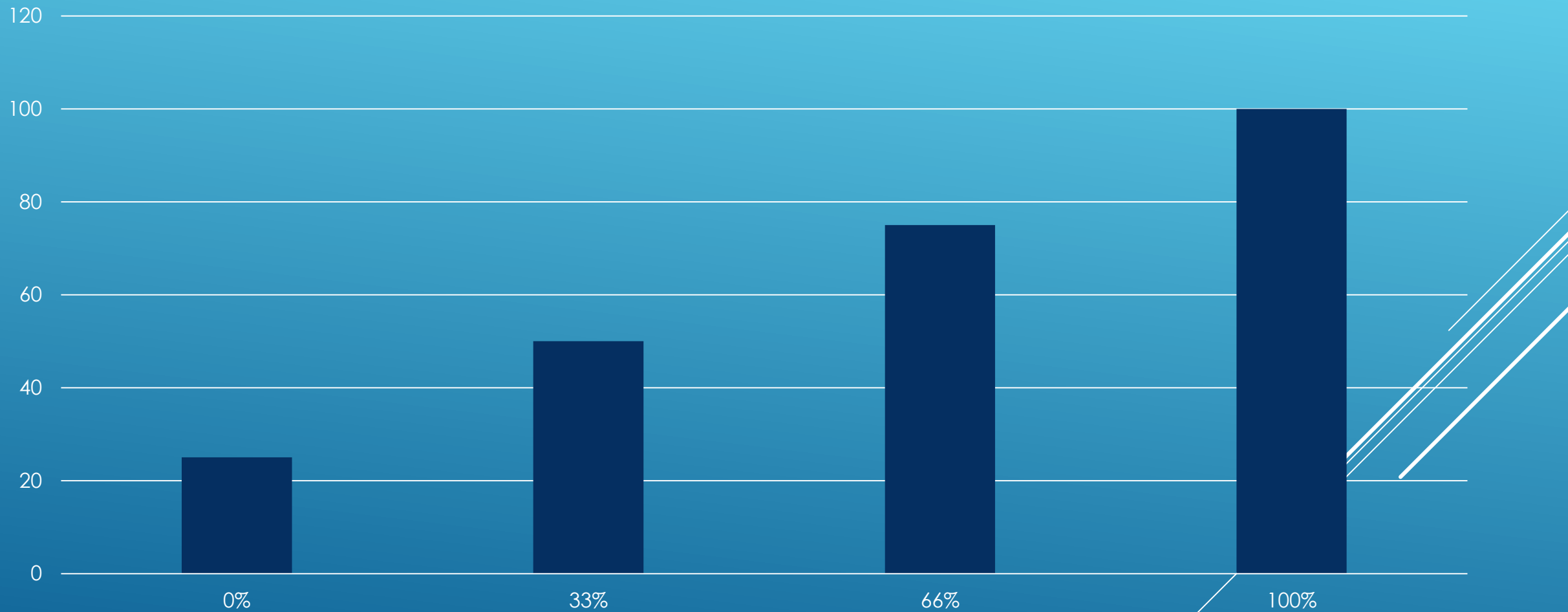
Computes a specific percentile

PERCENTILE\_CONT()

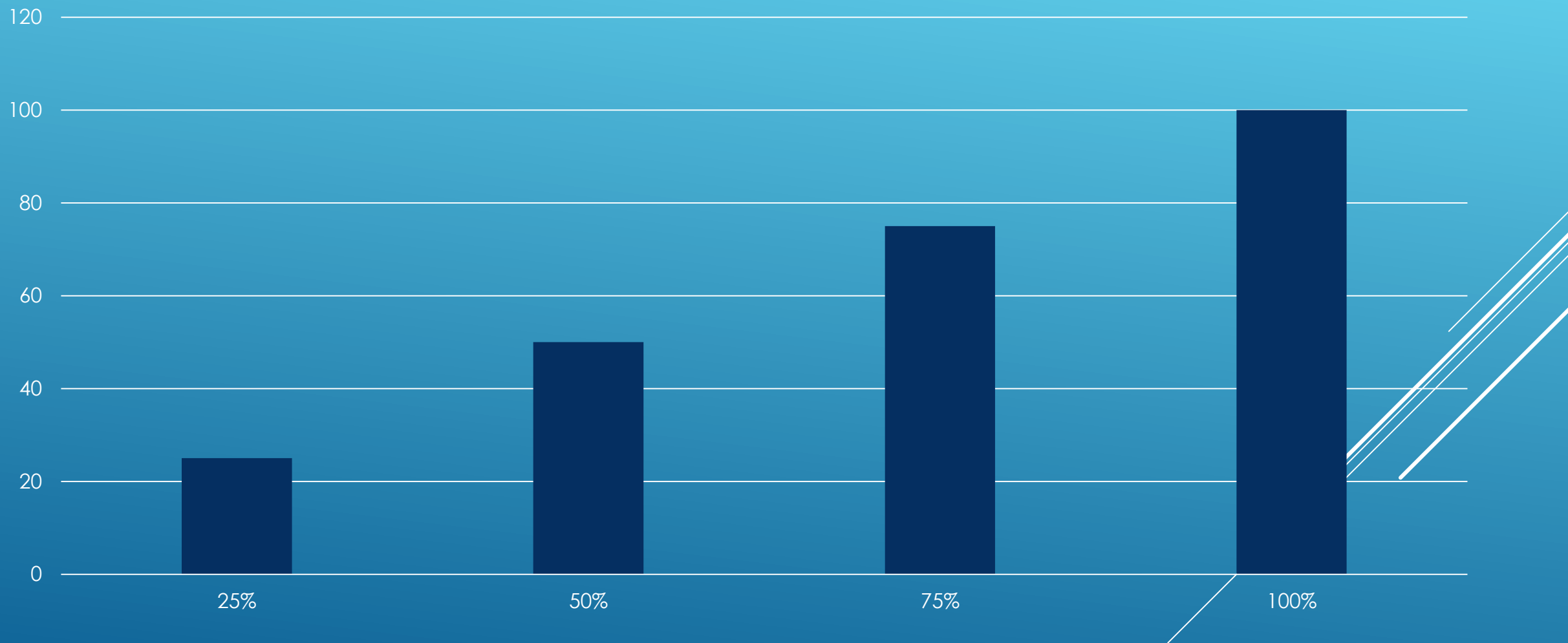
Computes an exact percentile



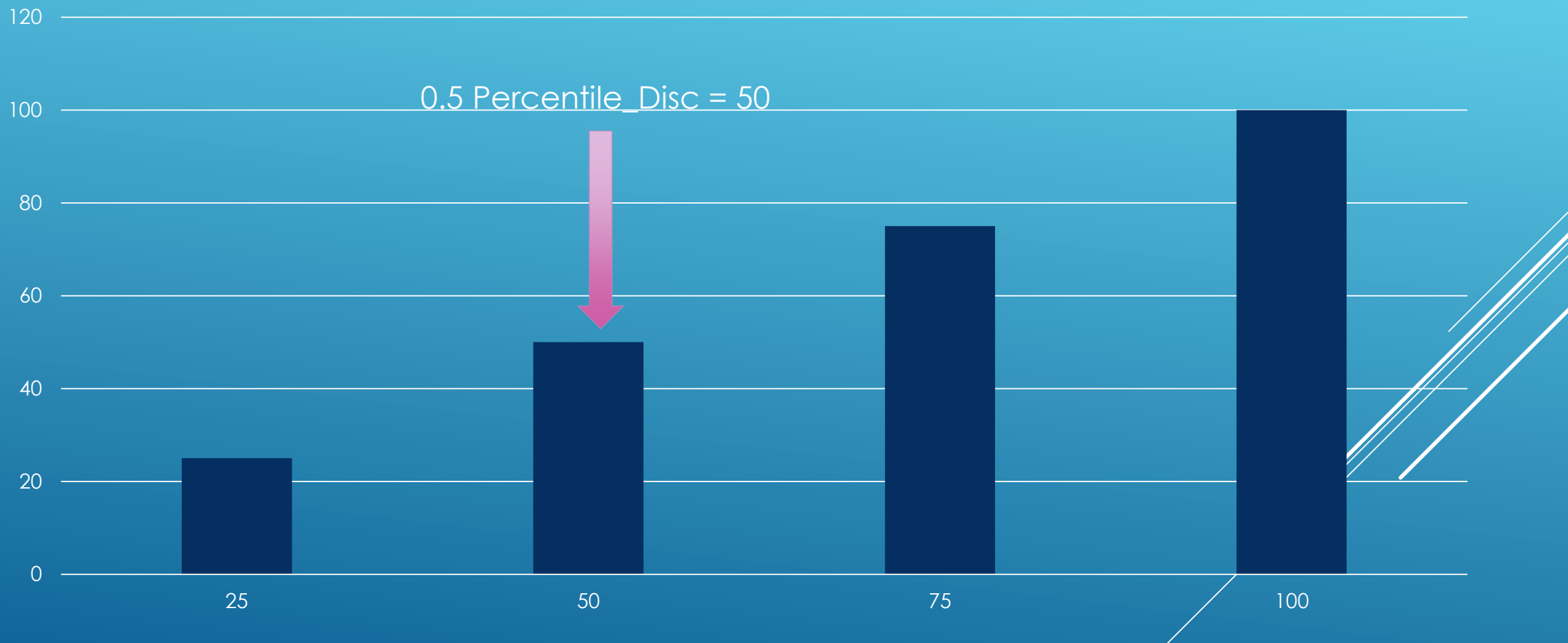
# PERCENT\_RANK EXAMPLE



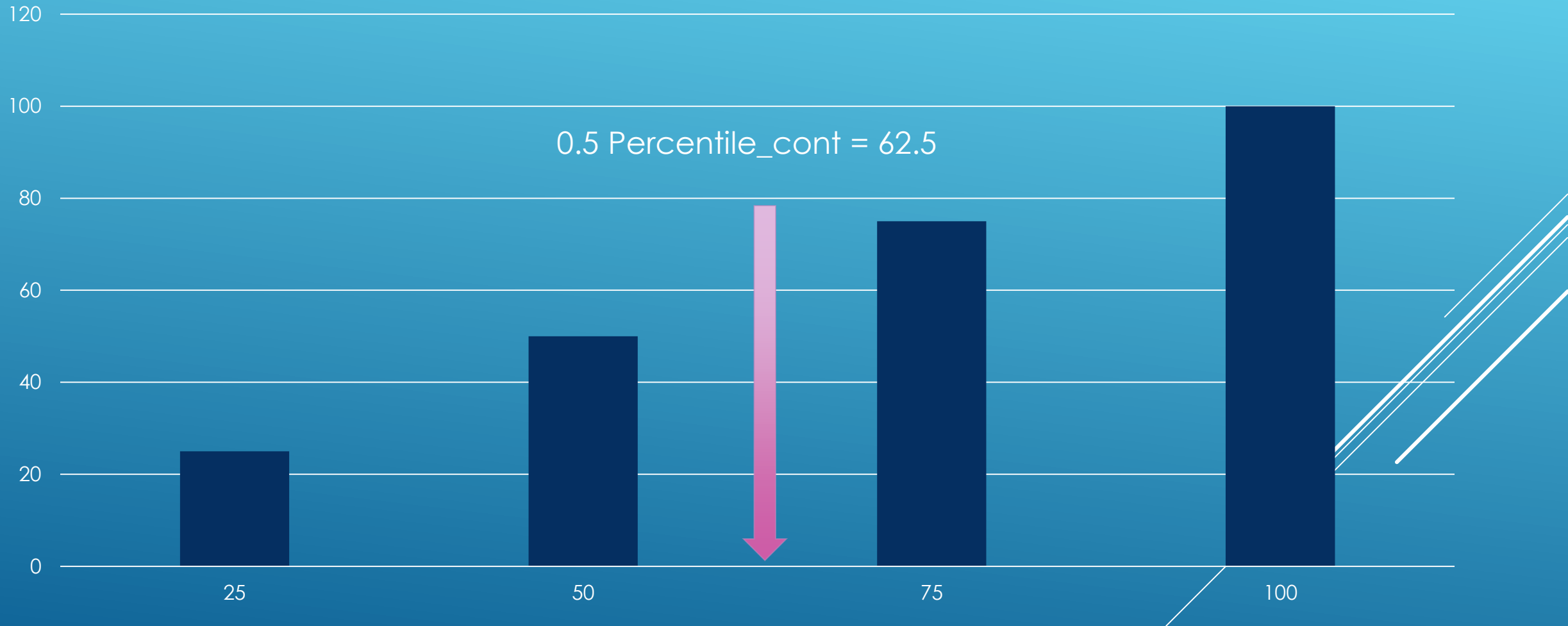
# CUME\_DIST EXAMPLE



# PERCENTILE\_DISC EXAMPLE



# PERCENTILE\_CONT EXAMPLE



DEMO 6

# Statistical functions



# PERFORMANCE AGENDA

Issues

Tools

Execution plan

Statistics IO and Time

Indexes

Framing

Measuring performance



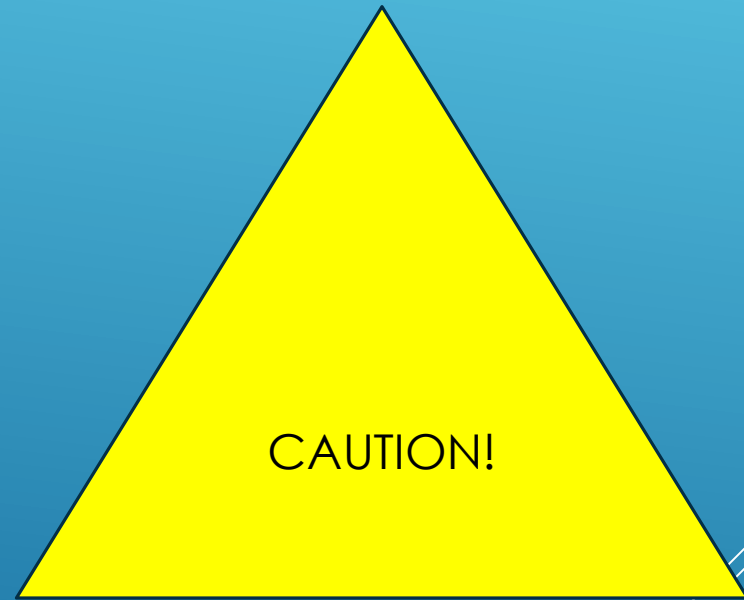
# ISSUES

Sorting

Window aggregates

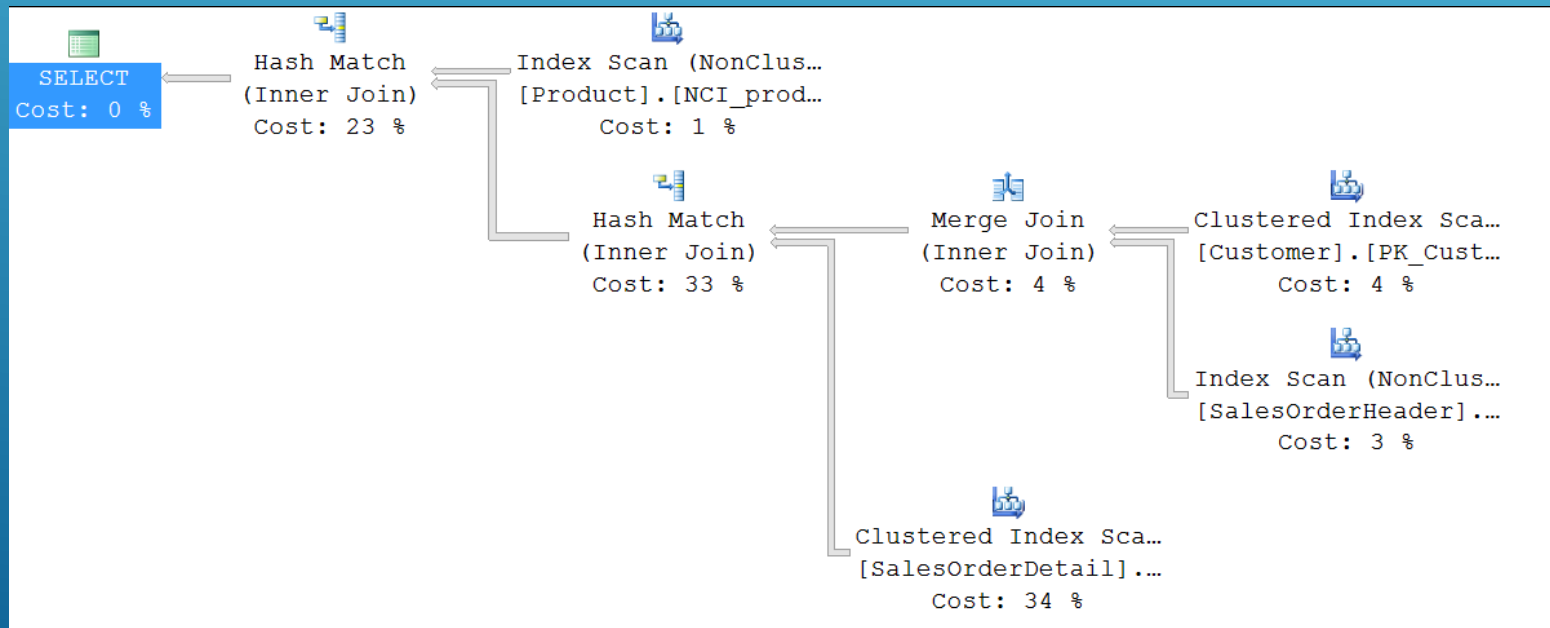
Default frame

Complex queries



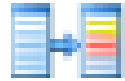
# TOOLS: EXECUTION PLANS

Text or graphical representation of how SQL Server processes a query  
CTRL+M





# EXECUTION PLAN OPERATORS



Sequence Project  
(Compute Scalar)  
Cost: 0 %



Segment  
Cost: 0 %

# EXECUTION PLAN OPERATORS



Table Spool  
(Lazy Spool)  
Cost: 0 %



Window Spool  
Cost: 3 %

# EXECUTION PLAN OPERATORS



DEMO 7

# Execution Plan



# TOOLS: STATISTICS IO AND TIME

SET STATISTICS IO ON

Number of pages that are processed

Repeatable

SET STATISTICS TIME ON

Time to run the query

Fluxuates

Several white lines of varying lengths and slopes are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.

DEMO 8

# STATISTICS IO AND TIME

The background is a blue gradient. In the bottom right corner, there are several white lines of varying lengths and slopes, creating a dynamic, abstract design.

# INDEXING

## POC Index\*

- Filtered column(s) + **P**artition column(s) + **O**rder by column(s) + **C**overing columns(s)

\*From Itzik Ben-Gan

DEMO 9

INDEXING





# FRAMING

Default frame: RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

Better performance with ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

IMPORTANT: If ORDER BY column values are not unique, will get different results

Affects Accumulating Aggregates, FIRST\_VALUE and LAST\_VALUE

Three parallel white diagonal lines are located in the bottom right corner of the slide, extending from the right edge towards the bottom left.

# FRAMING: VOCABULARY

Term	Used for
ROWS	Positional operator used to define the frame
RANGE	Logical operator used to define the frame. Not fully supported. The default if ROWS is not specified
UNBOUNDED PRECEDING	The first row of the partition
UNBOUNDED FOLLOWING	The last row of the partition
CURRENT ROW	The row where the calculation is being performed

# FRAMING



# FRAMING



# FRAMING



# FRAMING



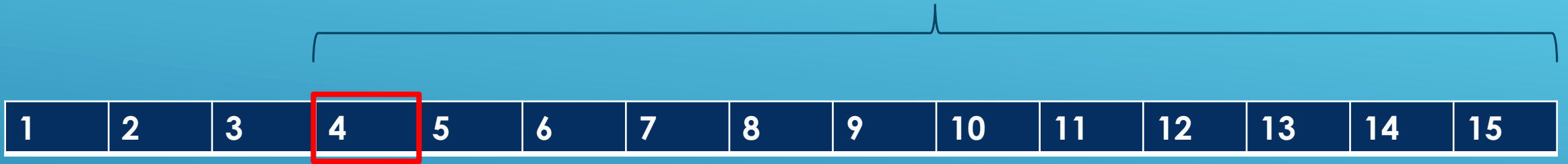
# FRAMING



ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

ROWS UNBOUNDED PRECEDING

# FRAMING



ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING

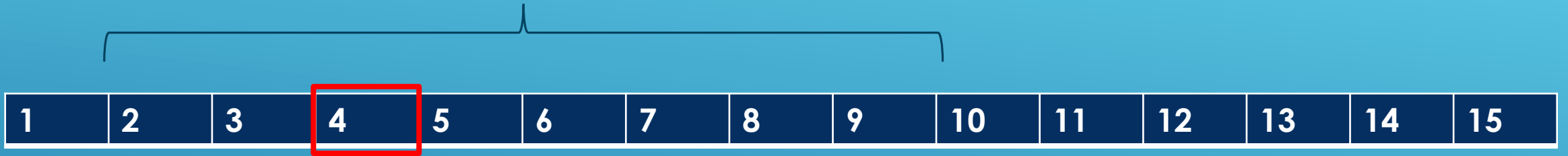


# FRAMING



ROWS BETWEEN 2 PRECEDING AND CURRENT ROW

# FRAMING



ROWS BETWEEN 2 PRECEDING AND 5 FOLLOWING

DEMO 10

FRAMING



# COMPLEX QUERIES

May not find a perfect index

Window aggregates

- Pre-aggregate if possible

Window is reusable

Weigh benefits vs. cost



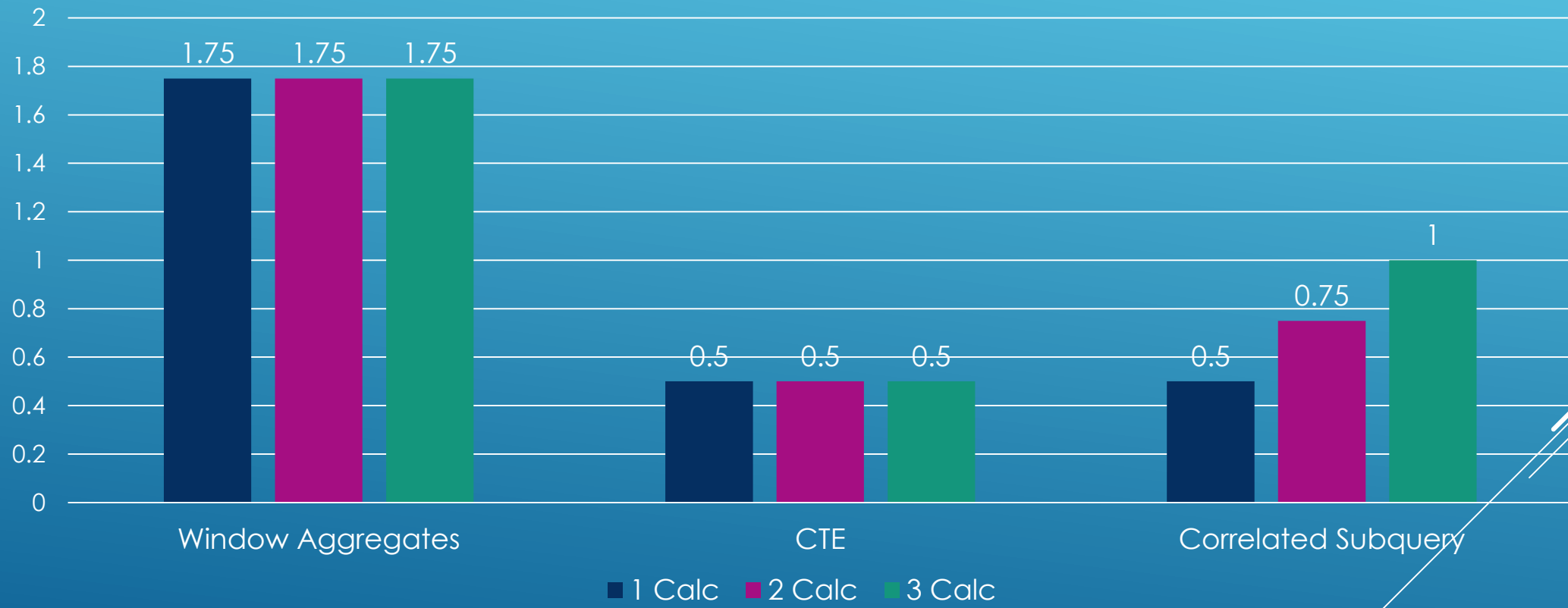
DEMO 11

# COMPLEX QUERIES

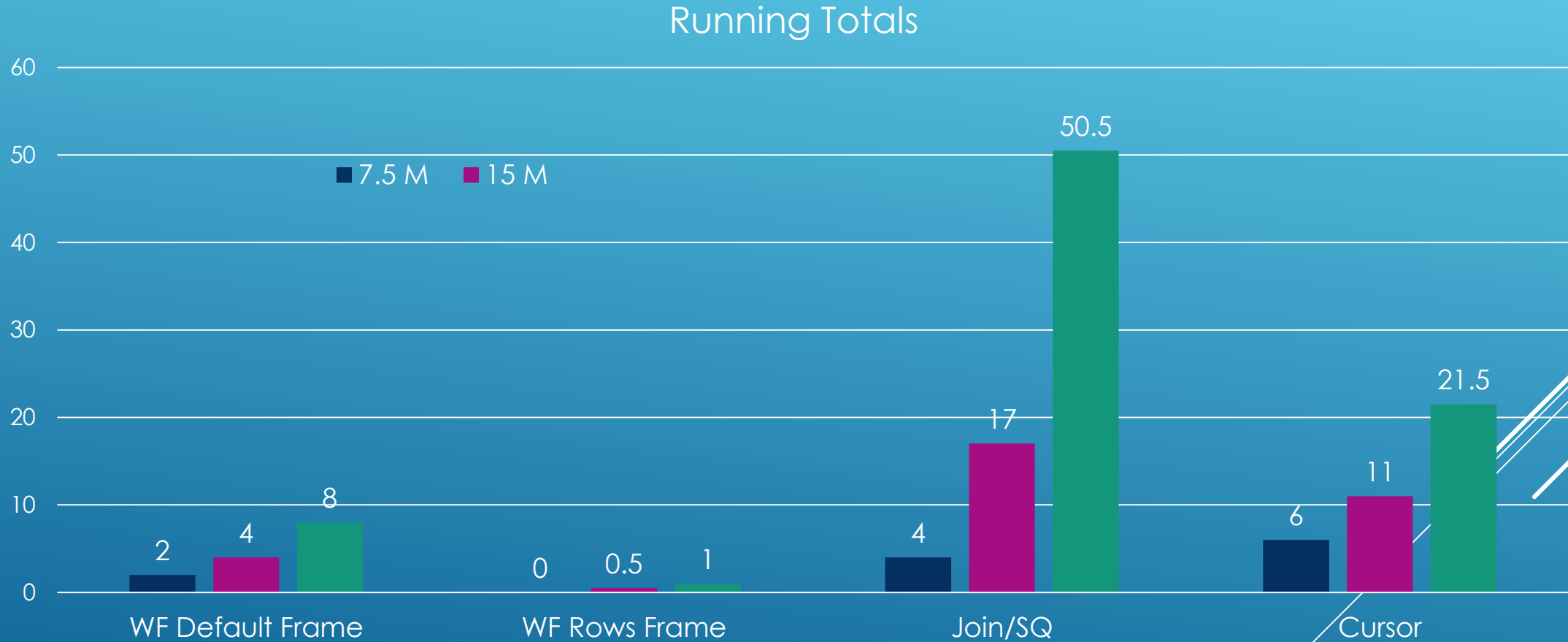


# PERFORMANCE COMPARISON

Subtotals – 30 million rows



# PERFORMANCE COMPARISON



DEMO 12

# PERFORMANCE COMPARISONS





# RESOURCES

Adam Machanic's Big Adventure Script

[http://sqlblog.com/blogs/adam\\_machanic/archive/2011/10/17/thinking-big-adventure.aspx](http://sqlblog.com/blogs/adam_machanic/archive/2011/10/17/thinking-big-adventure.aspx)

My book: Expert T-SQL Window Functions

Itzik Ben-Gan's book: Microsoft SQL Server 2012 High-Performance T-SQL Using Window Functions

<http://www.aunkathisql.com>

Pluralsight course