ADVANCED TECHNOLOGY DAYS

29. i 30. studeni 2023.
Cinestar Arena

SPONZORI KONFERENCIJE

SREBRNI SPONZOR

infobip M-Files NEPHOS unitfly eleks

BRONČANI SPONZOR

ALSO THE TECHNOLOGY PROVIDER 4PP enna energia naturalis

CONNECTIVITY PARTNER

telemach

SPONZOR COFFEE BREAKA

NESCAFÉ

PARTNERI

Syskit Visibility. Governance. Security. POPCORN FACTORY ZAGREB POPUP ARIEL PODS+

POWERED BY

WeAreDevelopers

ORGANIZATORI

!MPG INSPIRED MOMENTS NEPHOS unitfly

Great MS SQL functions for developers

Damir Matešić, MVP

Senior Database Developer @Span.eu

AD 2018 - Leading Data Events in Croatia

AD 2019 - Introduced SQL Saturday in Croatia

AD 2020 - Co-founder & organizer of #Dataweekender...

W: blog.matesic.info

@: dmatesic@gmail.com

in: linkedin.com/in/dmatesic

Slides & Demos

https://github.com/matesic-damir/presentations

COMPRESS AND DECOMPRESS

- 2016+
- ROW, PAGE…
- Syntax:

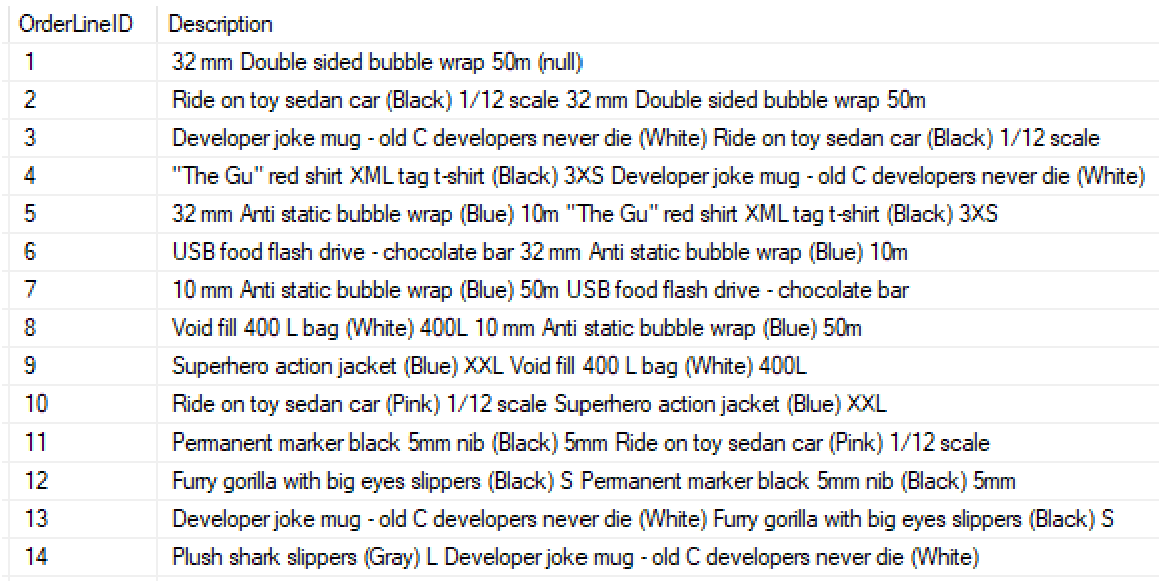
COMPRESS (expression)

- nvarchar(n), nvarchar(max), varchar(n), varchar(max), varbinary(n), varbinary(max), char(n), nchar(n), or binary(n) expression.
- GZIP
- INDEKS !?!
- XML, Log-s, Rarely used data

COMPRESS AND DECOMPRESS

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam mollis maximus quam, quis malesuada felis sollicitudin eget. Nunc feugiat nisi et elit blandit, eget vulputate quam faucibus. Nullam vitae commodo nisi. Cras consequat sapien et urna malesuada rhoncus. Sed feugiat ornare ultricies. Nulla neque velit, tristique pretium erat ut, fermentum consequat nulla. Fusce pellentesque ornare lacus, tempor molestie libero tincidunt nec. Pellentesque ac purus mattis, semper sapien id, rhoncus elit. Morbi sagittis sapien sit amet condimentum mollis. Maecenas in mollis eros.	Compression rate -> 62,24%
Damir like beer!	Compression rate -> -46,88%

COMPRESS AND DECOMPRESS



OrderLineID	Description
1	32 mm Double sided bubble wrap 50m (null)
2	Ride on toy sedan car (Black) 1/12 scale 32 mm Double sided bubble wrap 50m
3	Developer joke mug - old C developers never die (White) Ride on toy sedan car (Black) 1/12 scale
4	"The Gu" red shirt XML tag t-shirt (Black) 3XS Developer joke mug - old C developers never die (White)
5	32 mm Anti static bubble wrap (Blue) 10m "The Gu" red shirt XML tag t-shirt (Black) 3XS
6	USB food flash drive - chocolate bar 32 mm Anti static bubble wrap (Blue) 10m
7	10 mm Anti static bubble wrap (Blue) 50m USB food flash drive - chocolate bar
8	Void fill 400 L bag (White) 400L 10 mm Anti static bubble wrap (Blue) 50m
9	Superhero action jacket (Blue) XXL Void fill 400 L bag (White) 400L
10	Ride on toy sedan car (Pink) 1/12 scale Superhero action jacket (Blue) XXL
11	Permanent marker black 5mm nib (Black) 5mm Ride on toy sedan car (Pink) 1/12 scale
12	Furry gorilla with big eyes slippers (Black) S Permanent marker black 5mm nib (Black) 5mm
13	Developer joke mug - old C developers never die (White) Furry gorilla with big eyes slippers (Black) S
14	Plush shark slippers (Gray) L Developer joke mug - old C developers never die (White)

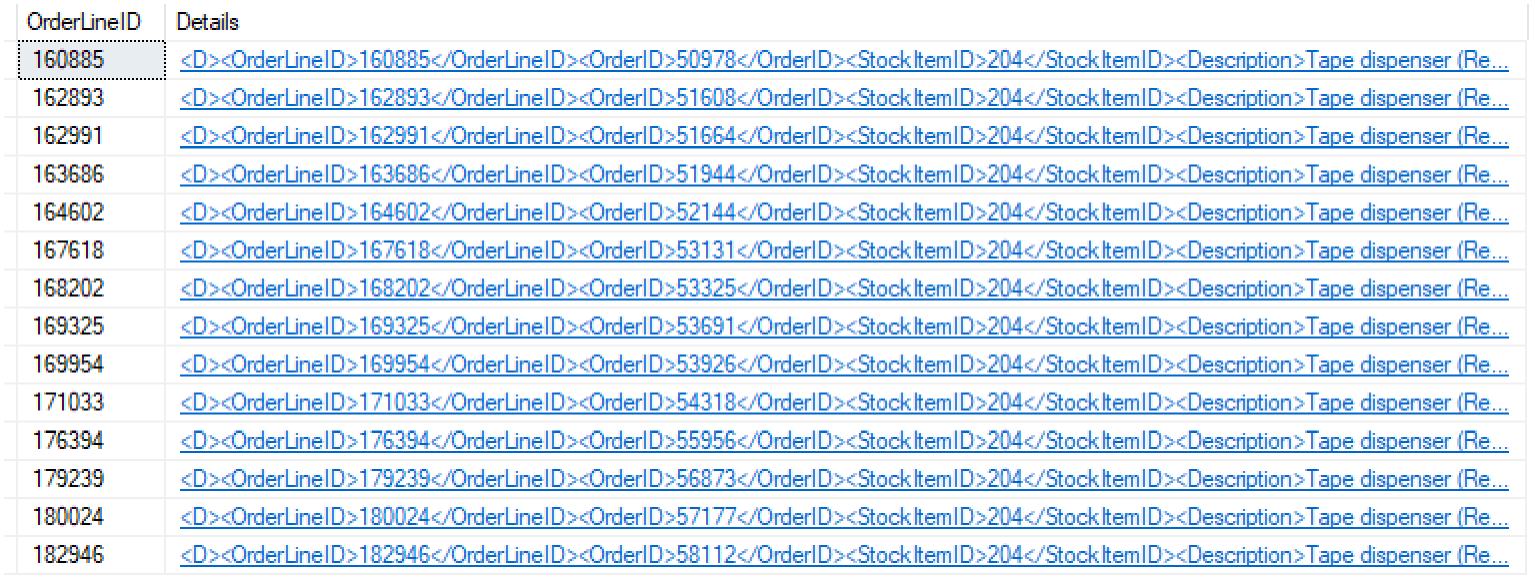
Rows: 231.412

Original size: 43.200 KB

PAGE: 22.600 KB
ROW: 22.248 KB

COMPRESS: 32.656 KB

COMPRESS AND DECOMPRESS



OrderLineID	Details
160885	<D><OrderLineID>160885</OrderLineID><OrderID>50978</OrderID><StockItemID>204</StockItemID><Description>Tape dispenser (Re...
162893	<D><OrderLineID>162893</OrderLineID><OrderID>51608</OrderID><StockItemID>204</StockItemID><Description>Tape dispenser (Re...
162991	<D><OrderLineID>162991</OrderLineID><OrderID>51664</OrderID><StockItemID>204</StockItemID><Description>Tape dispenser (Re...
163686	<D><OrderLineID>163686</OrderLineID><OrderID>51944</OrderID><StockItemID>204</StockItemID><Description>Tape dispenser (Re...
164602	<D><OrderLineID>164602</OrderLineID><OrderID>52144</OrderID><StockItemID>204</StockItemID><Description>Tape dispenser (Re...
167618	<D><OrderLineID>167618</OrderLineID><OrderID>53131</OrderID><StockItemID>204</StockItemID><Description>Tape dispenser (Re...
168202	<D><OrderLineID>168202</OrderLineID><OrderID>53325</OrderID><StockItemID>204</StockItemID><Description>Tape dispenser (Re...
169325	<D><OrderLineID>169325</OrderLineID><OrderID>53691</OrderID><StockItemID>204</StockItemID><Description>Tape dispenser (Re...
169954	<D><OrderLineID>169954</OrderLineID><OrderID>53926</OrderID><StockItemID>204</StockItemID><Description>Tape dispenser (Re...
171033	<D><OrderLineID>171033</OrderLineID><OrderID>54318</OrderID><StockItemID>204</StockItemID><Description>Tape dispenser (Re...
176394	<D><OrderLineID>176394</OrderLineID><OrderID>55956</OrderID><StockItemID>204</StockItemID><Description>Tape dispenser (Re...
179239	<D><OrderLineID>179239</OrderLineID><OrderID>56873</OrderID><StockItemID>204</StockItemID><Description>Tape dispenser (Re...
180024	<D><OrderLineID>180024</OrderLineID><OrderID>57177</OrderID><StockItemID>204</StockItemID><Description>Tape dispenser (Re...
182946	<D><OrderLineID>182946</OrderLineID><OrderID>58112</OrderID><StockItemID>204</StockItemID><Description>Tape dispenser (Re...

Rows: 231.412

Original size: 156.544 KB

COMPRESS: 79.808 KB

COMPRESS AND **DECOMPRESS**

- Opposite of COMPRESS?
- Syntax:

- DECOMPRESS (expression)
- varbinary(n), varbinary(max), or binary(n)
- **Return** -> data in varbinary(max)

- Casting!!!

COMPRESS AND **DECOMPRESS**

```
DECLARE @Input NVARCHAR(MAX) = N'Damir like Gin and tonic!'
SELECT DECOMPRESS(COMPRESS(@Input)) AS "Decompressed value"
```

```
0x440061006D00690072002000 6C0069006B00650020004700 69006E00200061006E006400200074006F0…
```

```
SELECT CAST(DECOMPRESS(COMPRESS(@Input)) AS NVARCHAR(max)) AS "Decompressed value"
```

```
Damir like Gin and tonic!
```

```
SELECT CAST(DECOMPRESS(COMPRESS(@Input)) AS VARCHAR(max)) AS "Decompressed value"
```

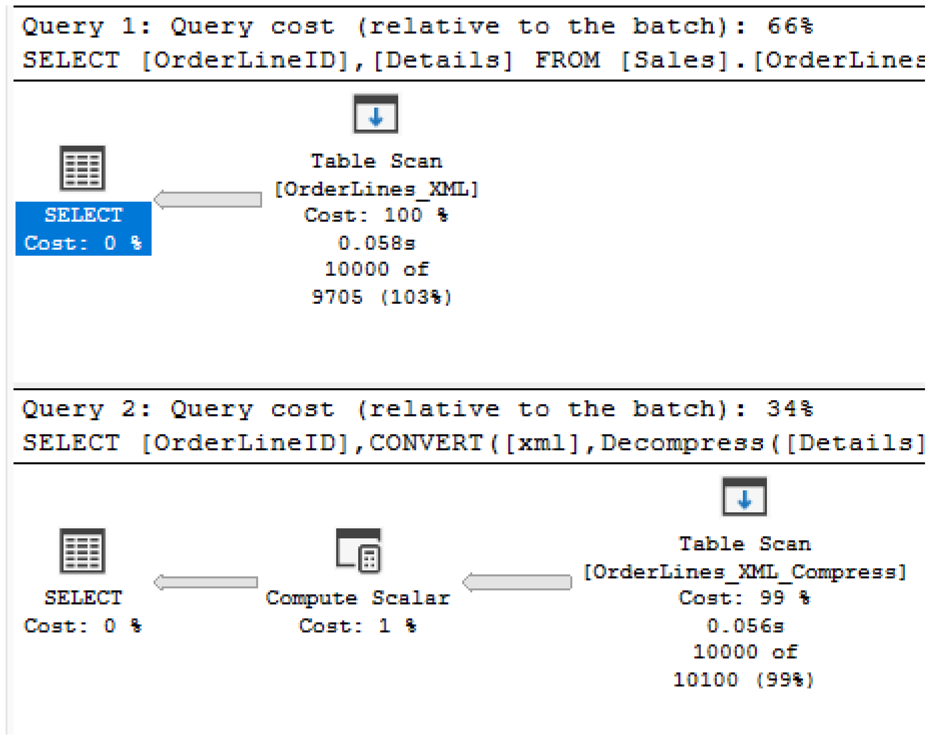
```
D
```

```
DECLARE @Input VARCHAR(MAX) = N'Damir like Gin and tonic!'
SELECT CAST(DECOMPRESS(COMPRESS(@Input)) AS NVARCHAR(max)) AS "Decompressed value"
```

```
慄業□楬敫讀湩愠摮琠湯捩!
```

COMPRESS AND **DECOMPRESS**

```
SELECT OrderLineID, Details FROM [Sales].[OrderLines_XML] WHERE [OrderLineID] <= 10000;

SELECT OrderLineID, CAST(DECOMPRESS(Details) AS XML) AS Details
FROM [Sales].[OrderLines_XML_Compress] WHERE [OrderLineID] <= 10000;
```



```
Query 1: Query cost (relative to the batch): 66%
SELECT [OrderLineID],[Details] FROM [Sales].[OrderLines
```

```
Query 2: Query cost (relative to the batch): 34%
SELECT [OrderLineID],CONVERT([xml],Decompress([Details]
```

```
Table 'OrderLines_XML'. Scan count 1, logical reads 19566

SQL Server Execution Times: CPU time = 31 ms,  elapsed time = 461 ms.

Table 'OrderLines_XML_Compress'. Scan count 1, logical reads 9974

SQL Server Execution Times: CPU time = 375 ms,  elapsed time = 1251 ms.
```

STRING_SPLIT – SQL 2016

- Syntax:

STRING_SPLIT (string , separator)

- table-valued function
- splitting string values by a separator

STRING_SPLIT – SQL 2022

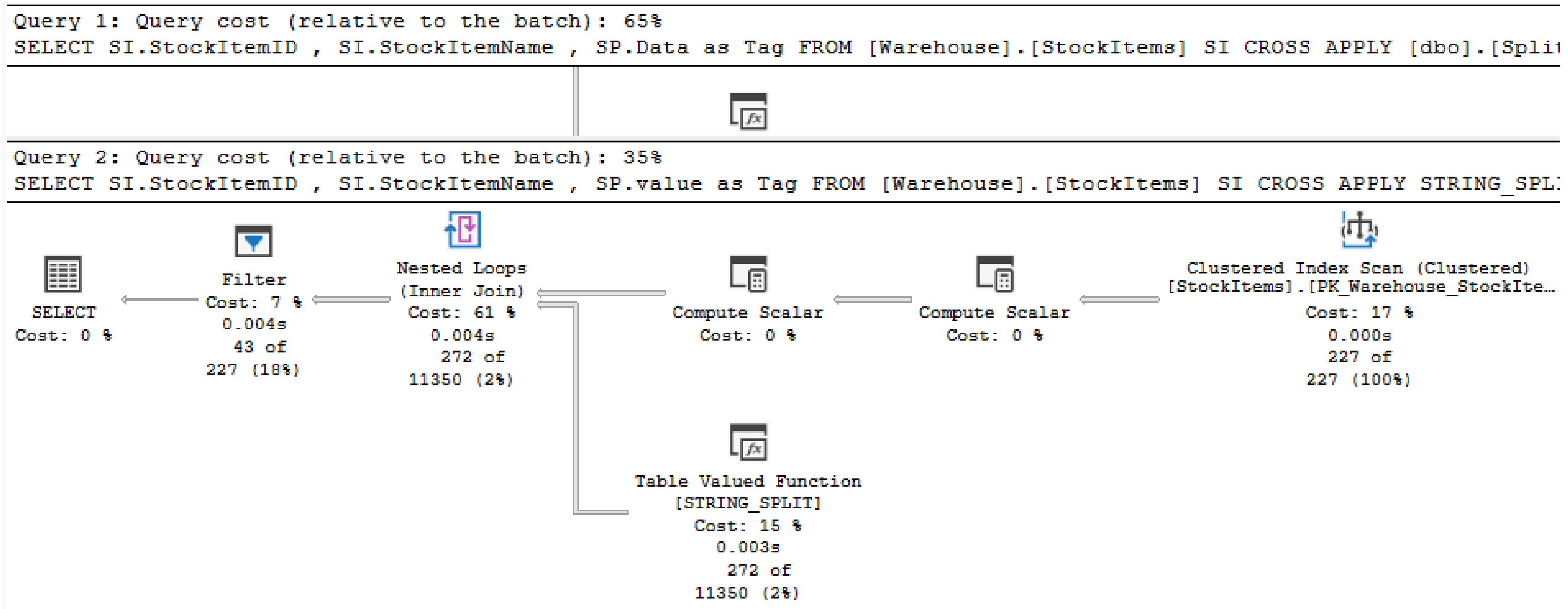
- Syntax:

STRING_SPLIT (string , separator [, **enable_ordinal**])

STRING_SPLIT – SQL 2022

```sql
SELECT SI.StockItemID, SI.StockItemName, SP.Data as Tag
FROM [Warehouse].[StockItems] SI
CROSS APPLY [dbo].[SplitString](Tags, ',') SP
WHERE SP.ItemNo = 2;

-- 2. The new way using STRING_SPLIT
SELECT SI.StockItemID, SI.StockItemName, SP.value as Tag
FROM [Warehouse].[StockItems] SI
CROSS APPLY STRING_SPLIT(Tags, ‘,’, 1) SP
WHERE ordinal = 2;
```

```
1)

Table '#B12D4DB7'. Scan count 28, logical reads 227
Table 'StockItems'. Scan count 1, logical reads 16

2)

Table 'StockItems'. Scan count 1, logical reads 16
```



```
Query 1: Query cost (relative to the batch): 65%
SELECT SI.StockItemID , SI.StockItemName , SP.Data as Tag FROM [Warehouse].[StockItems] SI CROSS APPLY [dbo].[Splil

Query 2: Query cost (relative to the batch): 35%
SELECT SI.StockItemID , SI.StockItemName , SP.value as Tag FROM [Warehouse].[StockItems] SI CROSS APPLY STRING_SPL:
```

STRING_AGG – SQL 2017

- Syntax:

```
STRING_AGG (expression, separator ) [ <order_clause> ]
<order_clause> ::=
    WITHIN GROUP ( ORDER BY <order_by_expression_list> [ ASC | DESC ] )
```

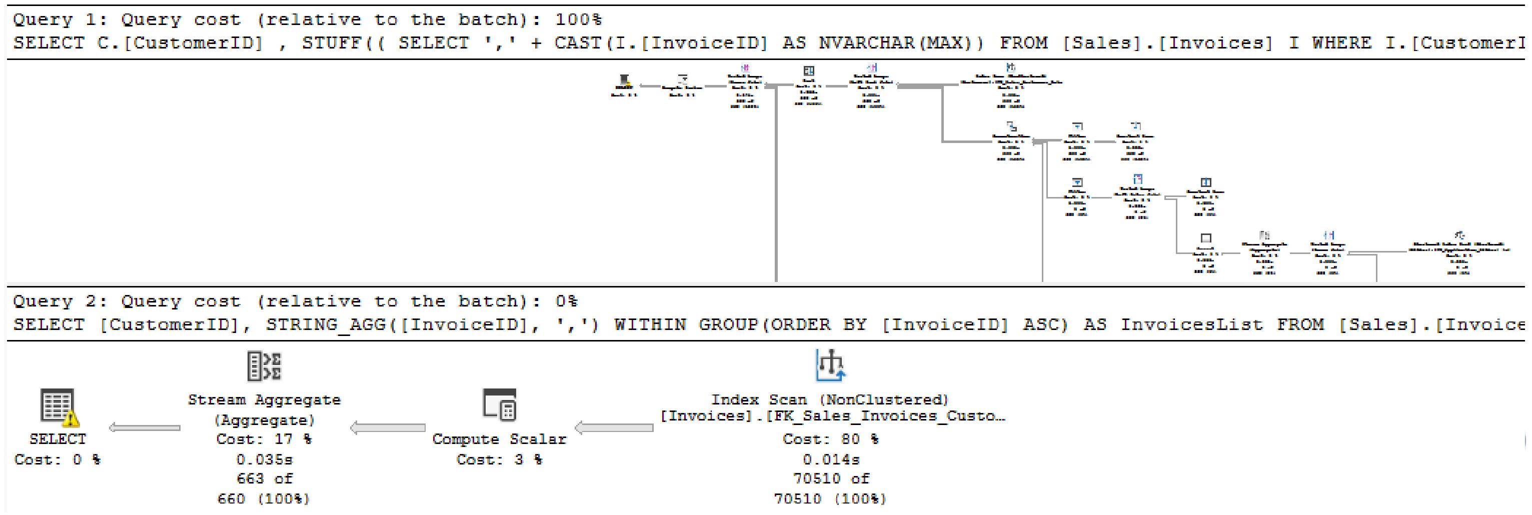
- string aggregation using a separator

STRING_AGG – SQL 2017

```
SELECT C.[CustomerID]
, STUFF((
    SELECT ',' + CAST(I.[InvoiceID] AS NVARCHAR(MAX))
    FROM [Sales].[Invoices] I
    WHERE I.[CustomerID] = C.[CustomerID]
    ORDER BY I.[InvoiceID] ASC
    FOR XML PATH(''), TYPE).value('.', 'varchar(max)'),1,1,‘’)
    AS InvoicesList
FROM [Sales].[Customers] AS C
ORDER BY C.[CustomerID] ASC;
```

```
1)

Table 'Invoices'. Scan count 663, logical reads 1514
Table 'Worktable'. Scan count 0, logical reads 0
Table 'Worktable'. Scan count 0, logical reads 0
Table 'Customers'. Scan count 1, logical reads 4

SQL Server Execution Times: CPU time = 78 ms,  elapsed time = 262 ms.

2)

Table 'Invoices'. Scan count 1, logical reads 166

SQL Server Execution Times: CPU time = 0 ms,  elapsed time = 345 ms.
```

```
SELECT [CustomerID],
STRING_AGG([InvoiceID], ',') WITHIN GROUP(ORDER BY [InvoiceID] ASC) AS InvoicesList
FROM [Sales].[Invoices] I
GROUP BY I.CustomerID
ORDER BY I.[CustomerID] ASC;
```



```
Query 1: Query cost (relative to the batch): 100%
SELECT C.[CustomerID] , STUFF(( SELECT ',' + CAST(I.[InvoiceID] AS NVARCHAR(MAX)) FROM [Sales].[Invoices] I WHERE I.[CustomerI

Query 2: Query cost (relative to the batch): 0%
SELECT [CustomerID], STRING_AGG([InvoiceID], ',') WITHIN GROUP(ORDER BY [InvoiceID] ASC) AS InvoicesList FROM [Sales].[Invoice
```

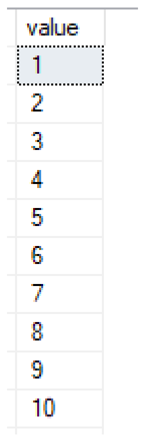
GENERATE_SERIES

- Syntax:

```
GENERATE_SERIES(<start>, <stop> [, STEP = <step>])
```

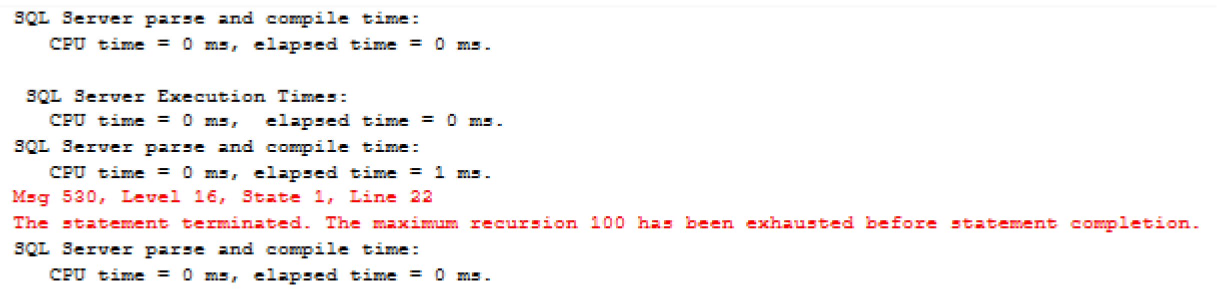
GENERATE_SERIES

```
Od 1 do 100
;WITH cte(n) AS
(
  SELECT 1 UNION ALL
  SELECT n + 1 FROM cte n WHERE n < 100
)
SELECT value = n FROM cte;
```



```
value
1
2
3
4
5
6
7
8
9
10
```

```
Od 1.000 do 1.000.000
;WITH cte(n) AS
(
  SELECT 1000 UNION ALL
  SELECT n + 1 FROM cte n WHERE n <= 1000000
)
SELECT value = n FROM cte
WHERE cte.n BETWEEN 1000 AND 1000000;
```



```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 1 ms.
Msg 530, Level 16, State 1, Line 22
The statement terminated. The maximum recursion 100 has been exhausted before statement completion.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
```

GENERATE_SERIES

```sql
-- Some object in DB -> 328 ms
;WITH Cte AS (SELECT ROW_NUMBER() OVER (ORDER BY C.name) Rn
FROM sys.columns C, sys.objects)
SELECT Cte.Rn
FROM Cte
WHERE Cte.Rn BETWEEN 1000 AND 1000000
```

GENERATE_SERIES

```
-- Function -> 217 ms
CREATE OR ALTER FUNCTION [dbo].[NumberRange](@start BIGINT, @end BIGINT)
RETURNS TABLE
AS
RETURN
( WITH CTE(n) AS(
SELECT 1 AS Number UNION ALL SELECT 1
),
CTE2(n) AS (SELECT 1 AS Number  FROM CTE x, CTE y), CTE3(n) AS (SELECT 1 AS
Number  FROM CTE2 x, CTE2 y), CTE4(n) AS (SELECT 1 AS Number  FROM CTE3 x, CTE3
y), CTE5(n) AS (SELECT 1 AS Number  FROM CTE4 x, CTE4 y), CTE6(n) AS (SELECT 0 AS
Number  UNION ALL SELECT TOP (@end-@start) ROW_NUMBER() OVER (ORDER BY (SELECT
NULL))  AS Number FROM CTE5 x, CTE5 y) SELECT @start+n  AS Number FROM CTE6
WHERE @start+n <= @end)

SELECT Number FROM  [dbo].[NumberRange] (1000, 1000000) ORDER BY Number;
```

GENERATE_SERIES

```
-- Table -> 78 ms
CREATE TABLE [dbo].Numbers (
Number INT NOT NULL,
 CONSTRAINT [PK_Number] PRIMARY KEY CLUSTERED
(
[Number] ASC
)
)

INSERT INTO dbo.Numbers (Number)
SELECT value …;


SELECT Number FROM  [dbo].[Numbers] WHERE Number BETWEEN 1000 AND 1000000;
```

GENERATE_SERIES

```sql
-- New function -> 0 ms
SELECT value FROM GENERATE_SERIES(1000, 1000000, 1);

-- Only even -> 250 ms vs 0 ms
SELECT Number FROM  [dbo].[NumberRange] (1000, 1000000) WHERE Number%2 = 0;
SELECT value FROM GENERATE_SERIES(1000, 1000000, 2);

-- Decimal step !!!
DECLARE @start decimal(3,1) = 0.0;
DECLARE @stop decimal(3,1) = 10.0;
DECLARE @step decimal(3,1) = 0.1;

SELECT value FROM GENERATE_SERIES(@start, @stop, @step);

-- Negative step
SELECT value FROM GENERATE_SERIES(1000000, 1000, -1);
```

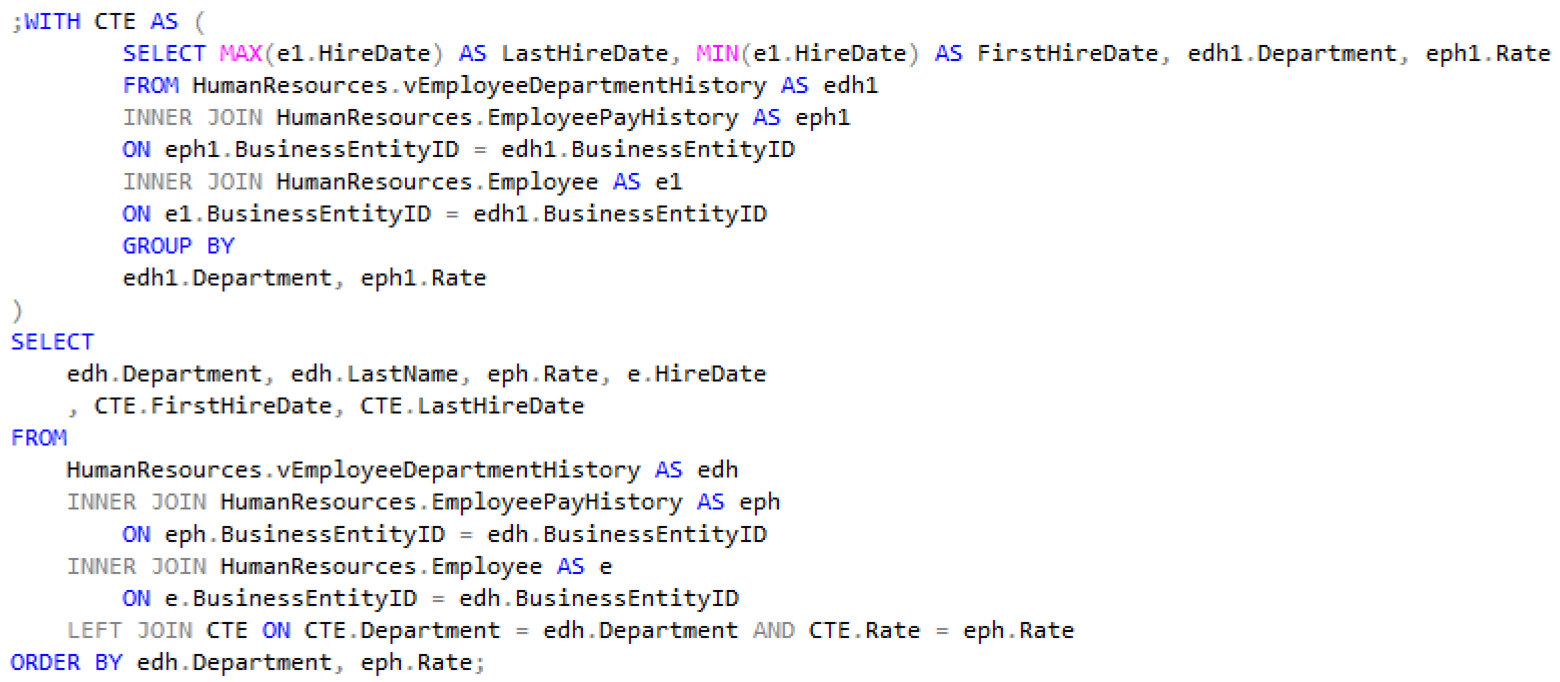
FIRST_VALUE, LAST_VALUE

- Syntax:

```
FIRST/LAST_VALUE ( [ scalar_expression ] )
  OVER ( [ partition_by_clause ] order_by_clause [ rows_range_clause ] )
```

FIRST_VALUE, LAST_VALUE

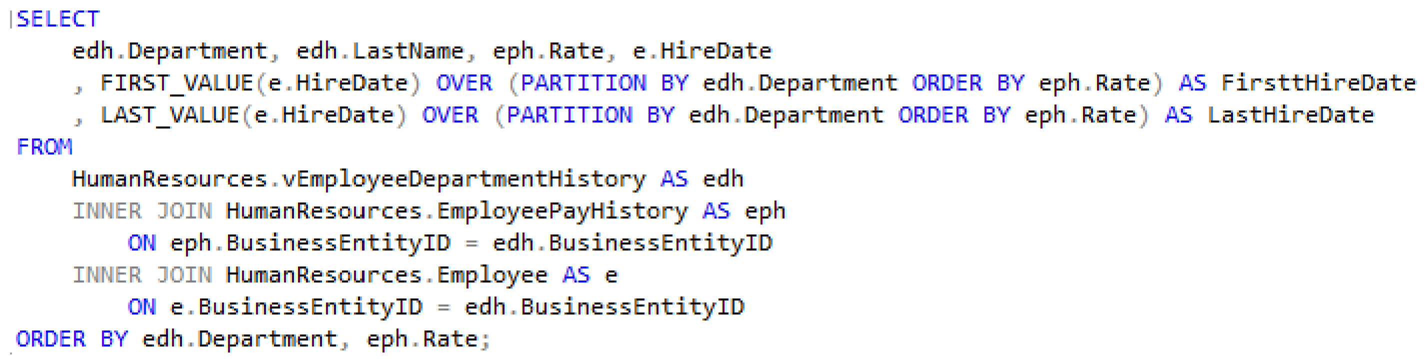
Retrieve employees by department and pay grade, when they were hired, the minimum and last date of employment in that department



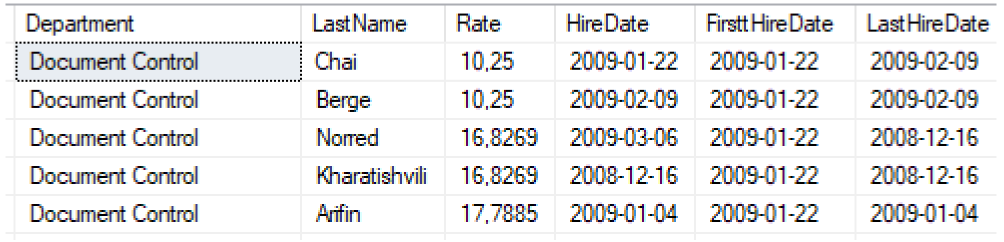
```sql
;WITH CTE AS (
        SELECT MAX(e1.HireDate) AS LastHireDate, MIN(e1.HireDate) AS FirstHireDate, edh1.Department, eph1.Rate
        FROM HumanResources.vEmployeeDepartmentHistory AS edh1
        INNER JOIN HumanResources.EmployeePayHistory AS eph1
        ON eph1.BusinessEntityID = edh1.BusinessEntityID
        INNER JOIN HumanResources.Employee AS e1
        ON e1.BusinessEntityID = edh1.BusinessEntityID
        GROUP BY
        edh1.Department, eph1.Rate
)
SELECT
    edh.Department, edh.LastName, eph.Rate, e.HireDate
    , CTE.FirstHireDate, CTE.LastHireDate
FROM
    HumanResources.vEmployeeDepartmentHistory AS edh
    INNER JOIN HumanResources.EmployeePayHistory AS eph
        ON eph.BusinessEntityID = edh.BusinessEntityID
    INNER JOIN HumanResources.Employee AS e
        ON e.BusinessEntityID = edh.BusinessEntityID
    LEFT JOIN CTE ON CTE.Department = edh.Department AND CTE.Rate = eph.Rate
ORDER BY edh.Department, eph.Rate;
```

FIRST_VALUE, LAST_VALUE

Retrieve employees by department and pay grade, when they were hired, the minimum and last date of employment in that department



```
SELECT
    edh.Department, edh.LastName, eph.Rate, e.HireDate
    , FIRST_VALUE(e.HireDate) OVER (PARTITION BY edh.Department ORDER BY eph.Rate) AS FirsttHireDate
    , LAST_VALUE(e.HireDate) OVER (PARTITION BY edh.Department ORDER BY eph.Rate) AS LastHireDate
FROM
    HumanResources.vEmployeeDepartmentHistory AS edh
    INNER JOIN HumanResources.EmployeePayHistory AS eph
        ON eph.BusinessEntityID = edh.BusinessEntityID
    INNER JOIN HumanResources.Employee AS e
        ON e.BusinessEntityID = edh.BusinessEntityID
ORDER BY edh.Department, eph.Rate;
```



Department	LastName	Rate	HireDate	FirsttHireDate	LastHireDate
Document Control	Chai	10,25	2009-01-22	2009-01-22	2009-02-09
Document Control	Berge	10,25	2009-02-09	2009-01-22	2009-02-09
Document Control	Norred	16,8269	2009-03-06	2009-01-22	2008-12-16
Document Control	Kharatishvili	16,8269	2008-12-16	2009-01-22	2008-12-16
Document Control	Arifin	17,7885	2009-01-04	2009-01-22	2009-01-04

IS [NOT] DISTINCT FROM

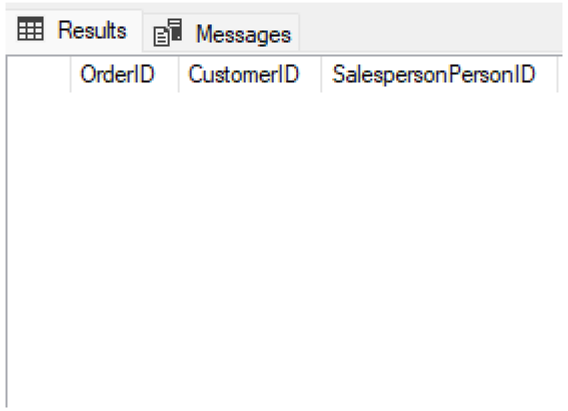
(The Distinct Predicate)

```
DECLARE @dt AS DATE = '20220212';
SELECT orderid, shippeddate
FROM Sales.Orders
WHERE shippeddate = @dt;

DECLARE @dt AS DATE = NULL; ??
```

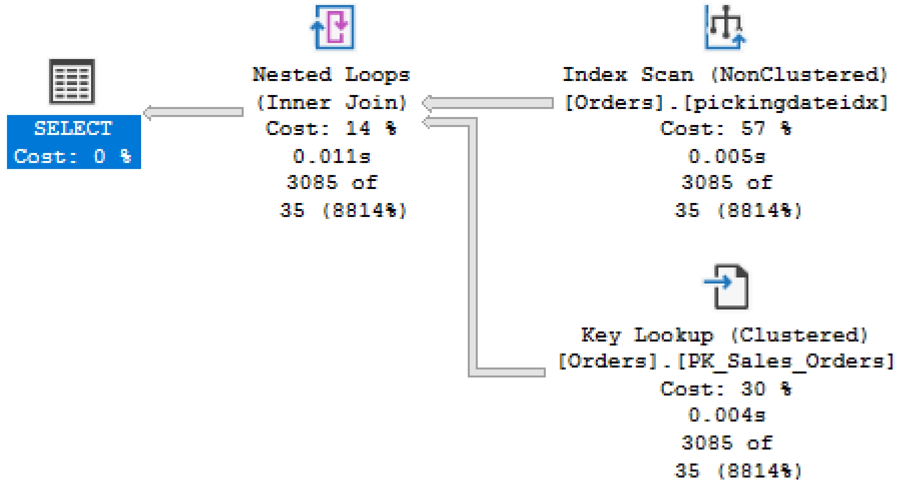
IS [NOT] DISTINCT FROM

(The Distinct Predicate)

```
-- Non picked up orders -> 3.085 items
DECLARE @dt datetime2 = NULL
SELECT * FROM Sales.Orders WHERE
PickingCompletedWhen = @dt;

-- ISNULL - Index scan
DECLARE @dt AS DATE = NULL;
SELECT * FROM Sales.Orders
WHERE ISNULL(PickingCompletedWhen, '99991231')
= ISNULL(@dt, '99991231’);

-- Combination - Index Scan
DECLARE @dt AS DATE = NULL;
SELECT * FROM Sales.Orders
WHERE PickingCompletedWhen = @dt OR
(PickingCompletedWhen IS NULL AND @dt IS NULL);
```


Results | Messages

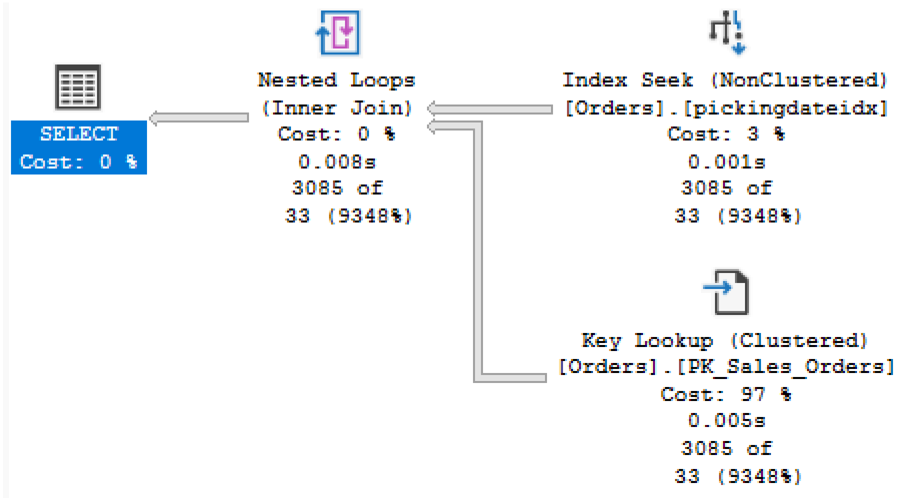
OrderID	CustomerID	SalespersonPersonID



IS [NOT] DISTINCT FROM

(The Distinct Predicate)

```
DECLARE @dt datetime2 = NULL;
SELECT *
FROM Sales.Orders
WHERE PickingCompletedWhen IS NOT DISTINCT FROM @dt;
```

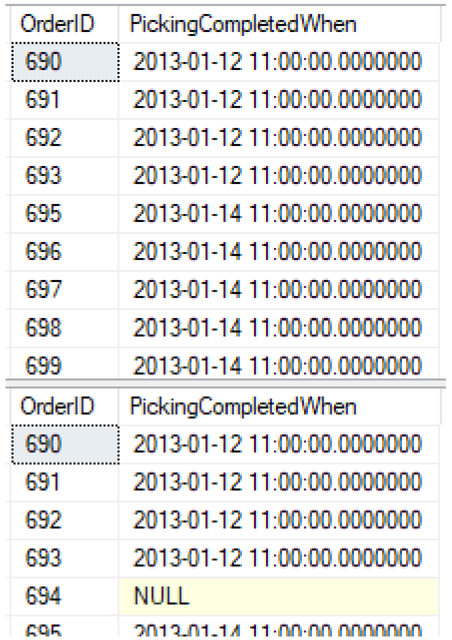


```
SELECT
Cost: 0 %

Nested Loops
(Inner Join)
Cost: 0 %
0.008s
3085 of
33 (9348%)

Index Seek (NonClustered)
[Orders].[pickingdateidx]
Cost: 3 %
0.001s
3085 of
33 (9348%)

Key Lookup (Clustered)
[Orders].[PK_Sales_Orders]
Cost: 97 %
0.005s
3085 of
33 (9348%)
```

IS [NOT] DISTINCT FROM

(The Distinct Predicate)

```
-- IS DISTINCT FROM
SELECT OrderID, PickingCompletedWhen FROM Sales.Orders
WHERE PickingCompletedWhen <> '2013-01-01 12:00:00.0000000'
ORDER BY OrderID;

SELECT OrderID, PickingCompletedWhen FROM Sales.Orders
WHERE
PickingCompletedWhen IS DISTINCT FROM '2013-01-01 12:00:00.0000000'
ORDER BY OrderID;

-- IS NOT DISTINCT FROM - Orders on a date
SELECT OrderID, PickingCompletedWhen FROM Sales.Orders
WHERE PickingCompletedWhen = '2013-01-01 12:00:00.0000000';

SELECT OrderID, PickingCompletedWhen FROM Sales.Orders
WHERE PickingCompletedWhen
IS NOT DISTINCT FROM '2013-01-01 12:00:00.0000000';
```



OrderID	PickingCompletedWhen
690	2013-01-12 11:00:00.0000000
691	2013-01-12 11:00:00.0000000
692	2013-01-12 11:00:00.0000000
693	2013-01-12 11:00:00.0000000
695	2013-01-14 11:00:00.0000000
696	2013-01-14 11:00:00.0000000
697	2013-01-14 11:00:00.0000000
698	2013-01-14 11:00:00.0000000
699	2013-01-14 11:00:00.0000000

OrderID	PickingCompletedWhen
690	2013-01-12 11:00:00.0000000
691	2013-01-12 11:00:00.0000000
692	2013-01-12 11:00:00.0000000
693	2013-01-12 11:00:00.0000000
694	NULL
695	2013-01-14 11:00:00.0000000

GREATEST() & LEAST()

- Syntax:

GREATEST/LEAST (expression1 [,...expressionN])

- same data type or implicitly convert
- NULL
- Types not supported for comparison: varchar(max), varbinary(max) or nvarchar(max) exceeding 8,000 bytes, cursor, geometry, geography, image, non-byte-ordered user-defined types, ntext, table, text, and xml.

GREATEST() & LEAST()

```
-- Example -- returns 5
SELECT GREATEST(1, 5, 3);

-- Widouth function?
SELECT CASE
  WHEN 1 > 5 THEN
    CASE WHEN 1 > 3 THEN 1 ELSE 3 END
  ELSE
    CASE WHEN 5 > 3 THEN 5 ELSE 3 END
  END;

-- Simple example
SELECT GREATEST(6.5, 3.5, 7) as greatest_of_numbers;
-- Does it work even if datatypes are not the same?
SELECT GREATEST(6.5, 3.5, N'7') as greatest_of_values;
-- What about strings?
SELECT GREATEST('Buffalo Bills', 'Cleveland Browns', 'Dallas Cowboys') as the_best_team
```

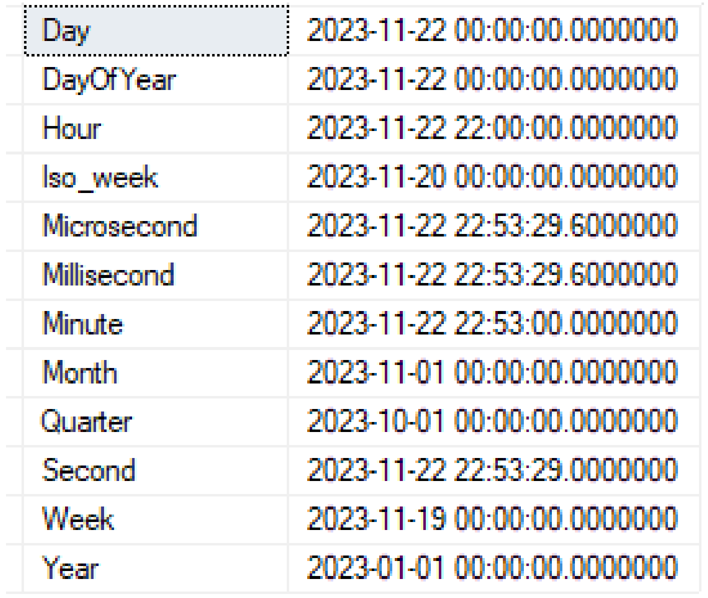
DATETRUNC

- Syntax:

DATETRUNC (datepart, date)

DATETRUNC

```sql
DECLARE @d datetime2 = GETDATE();
SELECT 'Year', DATETRUNC(year, @d) UNION
SELECT 'Quarter', DATETRUNC(quarter, @d) UNION
SELECT 'Month', DATETRUNC(month, @d) UNION
SELECT 'Week', DATETRUNC(week, @d) UNION
SELECT 'Iso_week', DATETRUNC(iso_week, @d) UNION
SELECT 'DayOfYear', DATETRUNC(dayofyear, @d) UNION
SELECT 'Day', DATETRUNC(day, @d) UNION
SELECT 'Hour', DATETRUNC(hour, @d) UNION
SELECT 'Minute', DATETRUNC(minute, @d) UNION
SELECT 'Second', DATETRUNC(second, @d) UNION
SELECT 'Millisecond', DATETRUNC(millisecond, @d) UNION
SELECT 'Microsecond', DATETRUNC(microsecond, @d);
```



Day	2023-11-22 00:00:00.0000000
DayOfYear	2023-11-22 00:00:00.0000000
Hour	2023-11-22 22:00:00.0000000
Iso_week	2023-11-20 00:00:00.0000000
Microsecond	2023-11-22 22:53:29.6000000
Millisecond	2023-11-22 22:53:29.6000000
Minute	2023-11-22 22:53:00.0000000
Month	2023-11-01 00:00:00.0000000
Quarter	2023-10-01 00:00:00.0000000
Second	2023-11-22 22:53:29.0000000
Week	2023-11-19 00:00:00.0000000
Year	2023-01-01 00:00:00.0000000

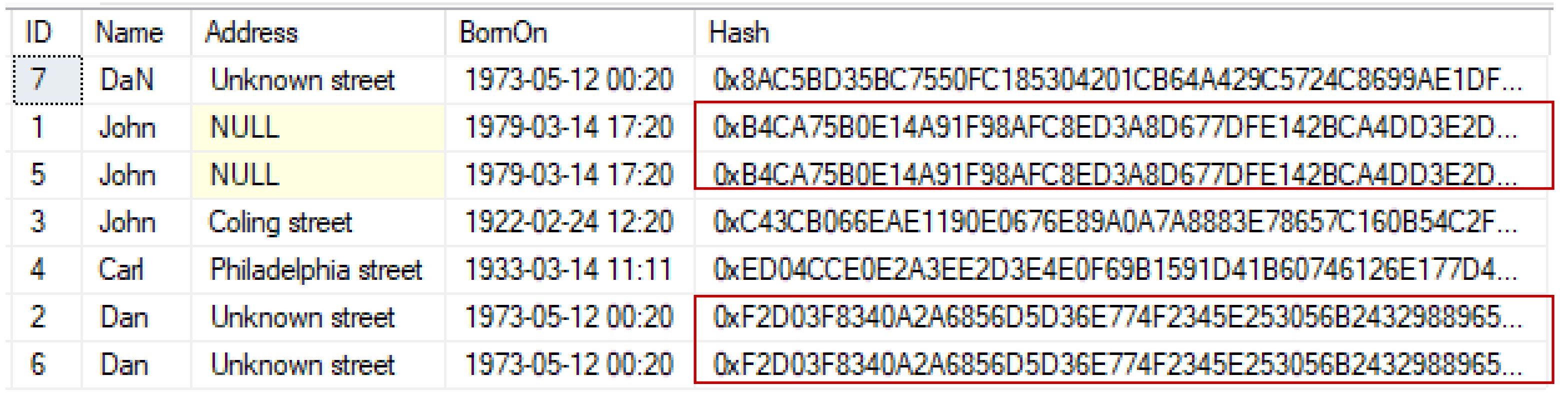
HASHBYTES

- SQL 2005 - ~~MD2, MD4, MD5, SHA, SHA1~~
- SQL 2012 - SHA2_256, SHA2_512

- SQL 2016 - Input: ~~8 000 bytes~~

HASHBYTES

```sql
;WITH CTE AS(
SELECT 1 AS ID, 'John' AS Name, NULL AS Address, '1979-03-14 17:20' AS BornOn UNION ALL
SELECT 2 AS ID, 'Dan' AS Name, 'Unknown street' AS Address, '1973-05-12 00:20' AS BornOn UNION ALL
SELECT 3 AS ID, 'John' AS Name, 'Coling street' AS Address, '1922-02-24 12:20' AS BornOn UNION ALL
SELECT 4 AS ID, 'Carl' AS Name, 'Philadelphia street' AS Address, '1933-03-14 11:11' AS BornOn UNION ALL
SELECT 5 AS ID, 'John' AS Name, NULL AS Address, '1979-03-14 17:20' AS BornOn UNION ALL
SELECT 6 AS ID, 'Dan' AS Name, 'Unknown street' AS Address, '1973-05-12 00:20' AS BornOn UNION ALL
SELECT 7 AS ID, 'DaN' AS Name, 'Unknown street' AS Address, '1973-05-12 00:20' AS BornOn
)
, CTEHash AS (
SELECT ID, Name, Address, BornOn
, HASHBYTES ('SHA2_512', (SELECT C.Name, C.Address, C.BornOn FROM CTE C WHERE C.ID = D.ID FOR JSON AUTO,
INCLUDE_NULL_VALUES)) AS Hash
FROM CTE D
) SELECT * FROM CTEHash ORDER BY Hash, ID;
```



ID	Name	Address	BornOn	Hash
7	DaN	Unknown street	1973-05-12 00:20	0x8AC5BD35BC7550FC185304201CB64A429C5724C8699AE1DF...
1	John	NULL	1979-03-14 17:20	0xB4CA75B0E14A91F98AFC8ED3A8D677DFE142BCA4DD3E2D...
5	John	NULL	1979-03-14 17:20	0xB4CA75B0E14A91F98AFC8ED3A8D677DFE142BCA4DD3E2D...
3	John	Coling street	1922-02-24 12:20	0xC43CB066EAE1190E0676E89A0A7A8883E78657C160B54C2F...
4	Carl	Philadelphia street	1933-03-14 11:11	0xED04CCE0E2A3EE2D3E4E0F69B1591D41B60746126E177D4...
2	Dan	Unknown street	1973-05-12 00:20	0xF2D03F8340A2A6856D5D36E774F2345E253056B2432988965...
6	Dan	Unknown street	1973-05-12 00:20	0xF2D03F8340A2A6856D5D36E774F2345E253056B2432988965...

sp_invoke_external_rest_endpoint

- HTTPS REST
- Allowed endpoints - *.windows.net, *.azure.net, …

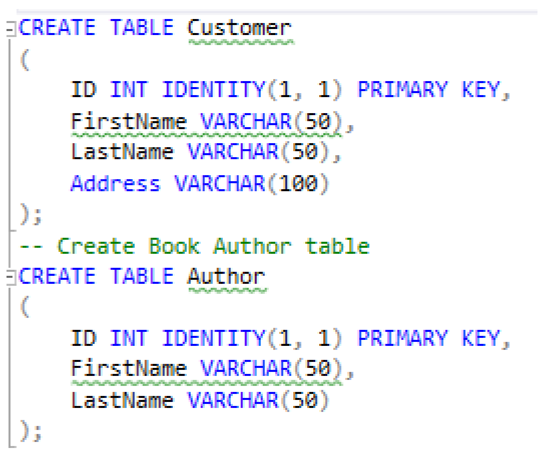
```
DECLARE @url NVARCHAR(4000) = N'https://Uri/openai/deployments/test/chat/completions?api-
version=2023-08-01-preview';
DECLARE @headers NVARCHAR(102) = N'{"api-key":"ApiKey"}'
DECLARE @payload NVARCHAR(max) = N'{"messages":[{"role":"system","content":"Question"}]}'
DECLARE @response NVARCHAR(max);

EXEC  sp_invoke_external_rest_endpoint
@url = @url,
@method = 'POST',
@headers = @headers,
@payload = @payload,
@timeout = 230,
@response = @response OUTPUT;
```

sp_invoke_external_rest_endpoint

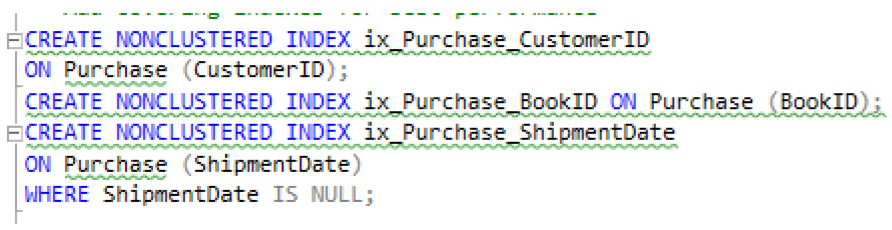
Q: Šta ima?
A: Ništa posebno, ja sam samo AI i nemam emocije niti sposobnost da imam loš ili dobar dan. Kako mogu da pomognem?

sp_invoke_external_rest_endpoint

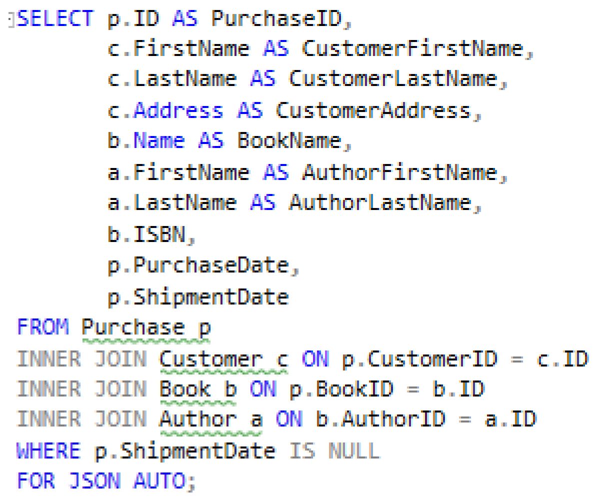
Q: Can you write a code for a MS SQL tables containing
1. Customer table: ID (auto generated), First name, Last Name, Address information
2. Book author table: ID (auto generated), First name, Last Name
3. Book table: ID (auto generated), Book name, Author (foregien key to Authors table), ISBN
4. Purchase table: ID (auto generated), Customer (foregien key to Customer table), Book (foregien key to Book table), Date of purchase, Shipment date (nullable)
Please add covering indexes for best performance when querying purchases, especially not shipped.
Insert some test data. Also, create a MS SQL Query to get all not shipped purchases, with all related data and the result should be a JSON document. And, finally, drop all tables.



```sql
CREATE TABLE Customer
(
    ID INT IDENTITY(1, 1) PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Address VARCHAR(100)
);
-- Create Book Author table
CREATE TABLE Author
(
    ID INT IDENTITY(1, 1) PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50)
);
```



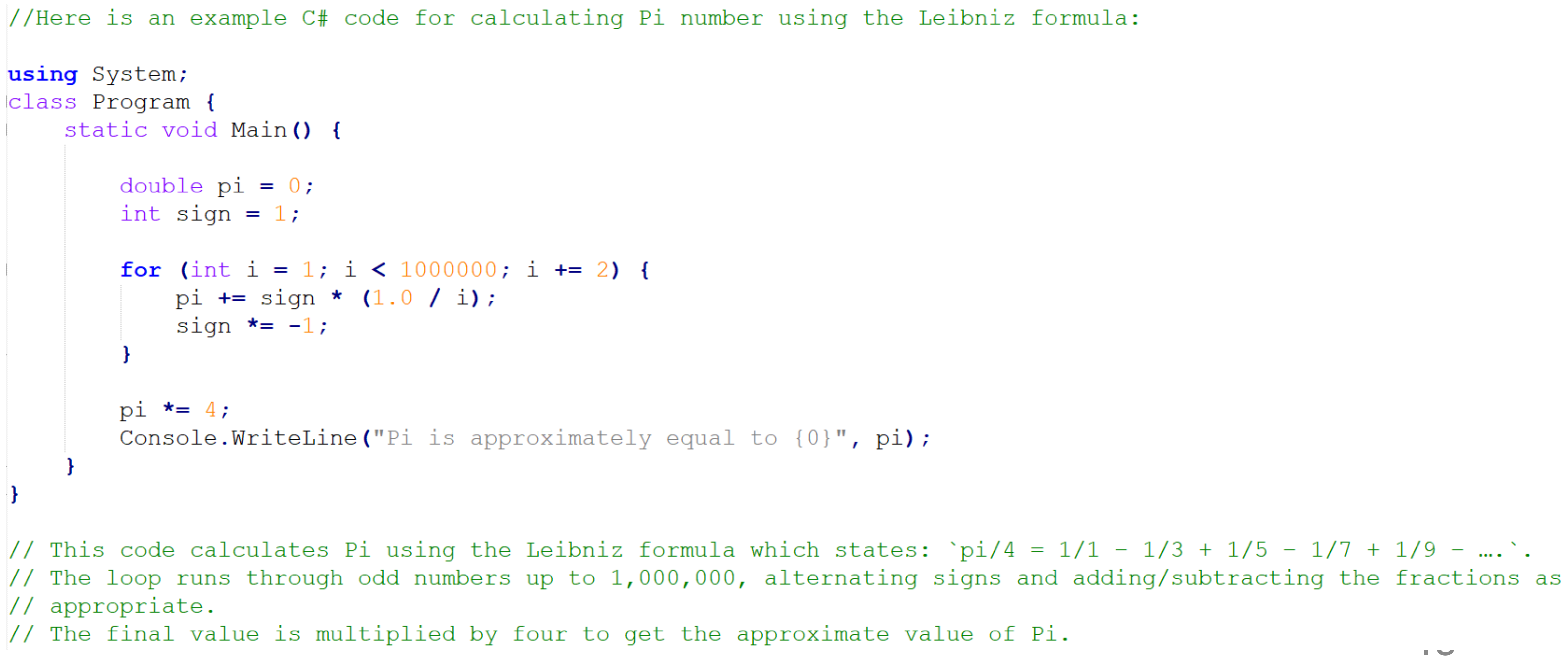
```sql
CREATE NONCLUSTERED INDEX ix_Purchase_CustomerID
ON Purchase (CustomerID);
CREATE NONCLUSTERED INDEX ix_Purchase_BookID ON Purchase (BookID);
CREATE NONCLUSTERED INDEX ix_Purchase_ShipmentDate
ON Purchase (ShipmentDate)
WHERE ShipmentDate IS NULL;
```



```sql
SELECT p.ID AS PurchaseID,
       c.FirstName AS CustomerFirstName,
       c.LastName AS CustomerLastName,
       c.Address AS CustomerAddress,
       b.Name AS BookName,
       a.FirstName AS AuthorFirstName,
       a.LastName AS AuthorLastName,
       b.ISBN,
       p.PurchaseDate,
       p.ShipmentDate
FROM Purchase p
INNER JOIN Customer c ON p.CustomerID = c.ID
INNER JOIN Book b ON p.BookID = b.ID
INNER JOIN Author a ON b.AuthorID = a.ID
WHERE p.ShipmentDate IS NULL
FOR JSON AUTO;
```

sp_invoke_external_rest_endpoint

Q: Write a C# code for calculating Pi number



```
//Here is an example C# code for calculating Pi number using the Leibniz formula:

using System;
class Program {
    static void Main() {

        double pi = 0;
        int sign = 1;

        for (int i = 1; i < 1000000; i += 2) {
            pi += sign * (1.0 / i);
            sign *= -1;
        }

        pi *= 4;
        Console.WriteLine("Pi is approximately equal to {0}", pi);
    }
}

// This code calculates Pi using the Leibniz formula which states: `pi/4 = 1/1 - 1/3 + 1/5 - 1/7 + 1/9 - ….`.
// The loop runs through odd numbers up to 1,000,000, alternating signs and adding/subtracting the fractions as
// appropriate.
// The final value is multiplied by four to get the approximate value of Pi.
```

Hvala.

ADVANCED TECHNOLOGY DAYS

29. i 30. studeni 2023.
Cinestar Arena