MS SQL 2022 New Functions, Syntaxes, Tips & Tricks

Damir Matešić

Damir Matešić, mag. inf.

- 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

Materijali

https://1drv.ms/u/s!Aoo7-aU7TEJblI1eHwQjkX2Q0X5Y2w?e=JGFOe9

ili

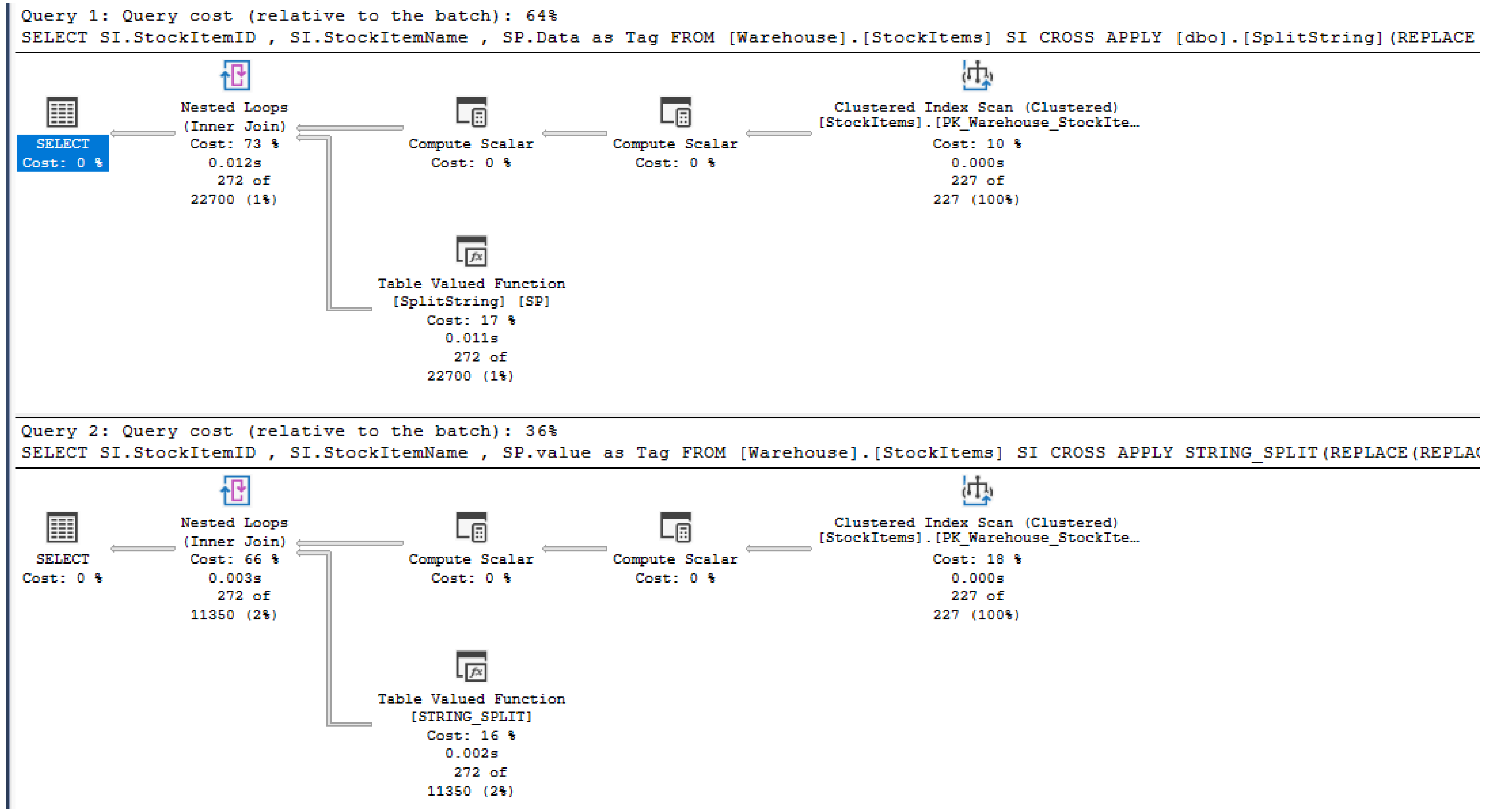
https://aka.3nf.hr/ITCommunityDay

STRING_SPLIT – SQL 2016

STRING_SPLIT (string , separator)

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

STRING_SPLIT



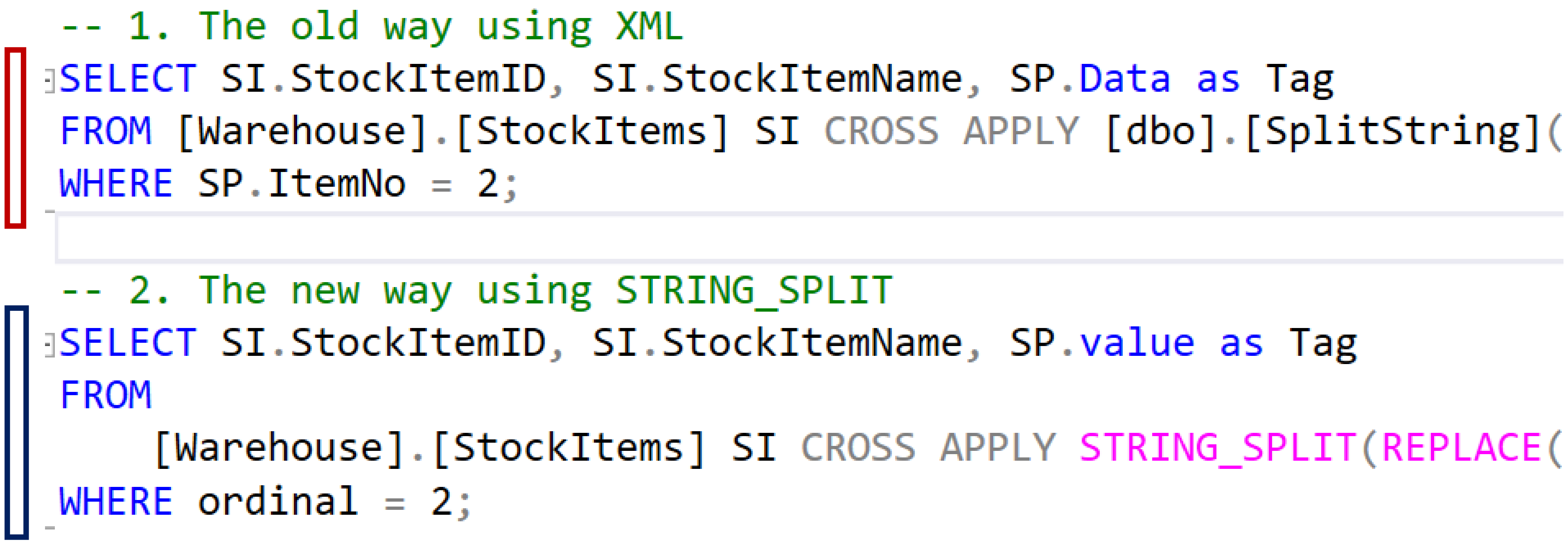
```
Query 1: Query cost (relative to the batch): 64%
SELECT SI.StockItemID , SI.StockItemName , SP.Data as Tag FROM [Warehouse].[StockItems] SI CROSS APPLY [dbo].[SplitString](REPLACE
```

```
Query 2: Query cost (relative to the batch): 36%
SELECT SI.StockItemID , SI.StockItemName , SP.value as Tag FROM [Warehouse].[StockItems] SI CROSS APPLY STRING_SPLIT(REPLACE(REPLA
```

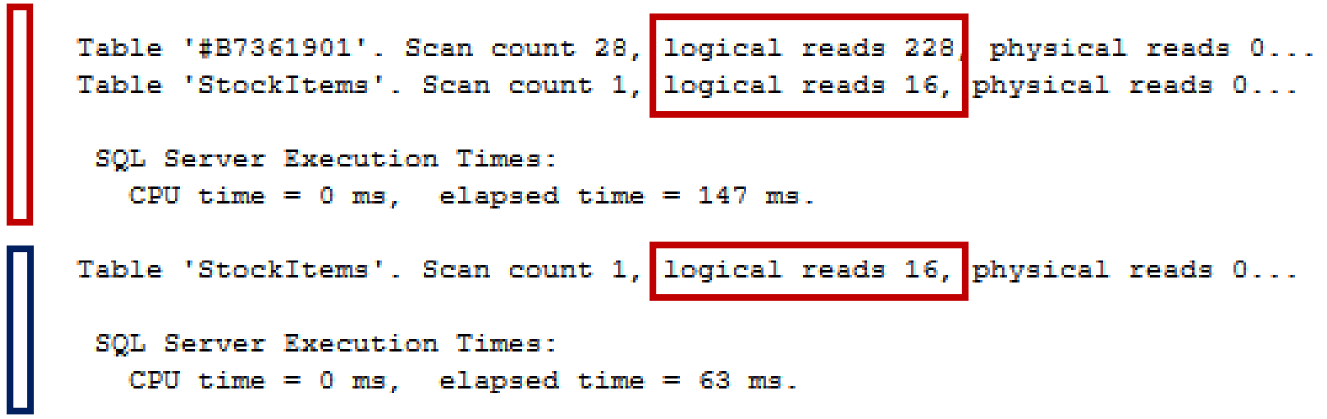
STRING_SPLIT

STRING_SPLIT (string , separator [, enable_ordinal])

STRING_SPLIT



```
-- 1. The old way using XML
SELECT SI.StockItemID, SI.StockItemName, SP.Data as Tag
FROM [Warehouse].[StockItems] SI CROSS APPLY [dbo].[SplitString](
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(REPLACE(
WHERE ordinal = 2;
```



```
Table '#B7361901'. Scan count 28, logical reads 228, physical reads 0...
Table 'StockItems'. Scan count 1, logical reads 16, physical reads 0...

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 147 ms.

Table 'StockItems'. Scan count 1, logical reads 16, physical reads 0...

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 63 ms.
```

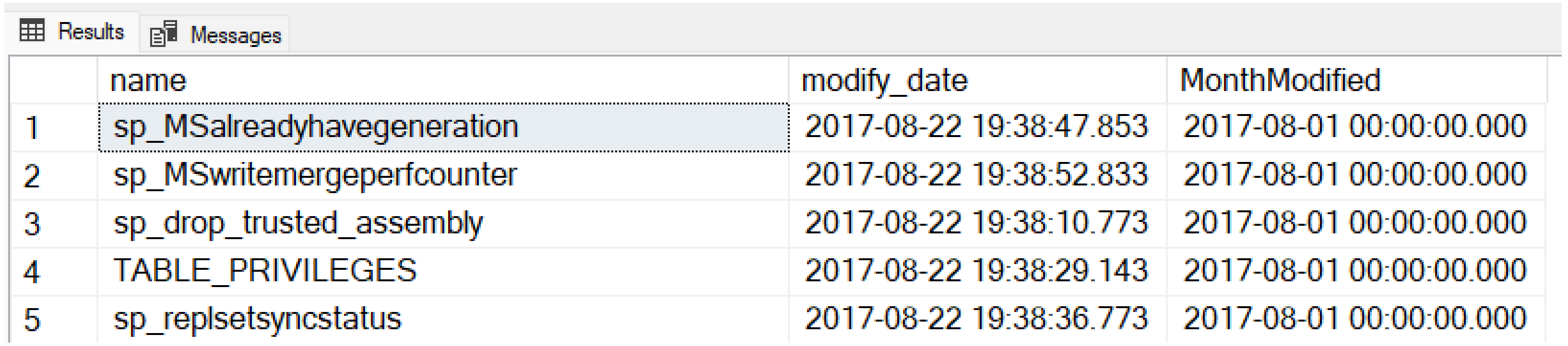
DATE_BUCKET

DATE_BUCKET(<datepart>, <bucket_width>, <input date/time> [, <origin>])

datePart	Abbreviations
day	**dd**, **d**
week	**wk**, **ww**
month	**mm, m**
quarter	**qq**, **q**
year	**yy**, **yyyy**
hour	**hh**
minute	**mi, n**
second	**ss, s**
millisecond	**ms**

DATE_BUCKET

```
-- Old way
SELECT name, modify_date, MonthModified = DATEADD(MONTH, DATEDIFF(MONTH, '19000101', modify_date), '19000101')
FROM sys.all_objects;

-- SQL DATE_BUCKET
SELECT name, modify_date, MonthModified = DATE_BUCKET(MONTH, 1, modify_date)
FROM sys.all_objects;
```

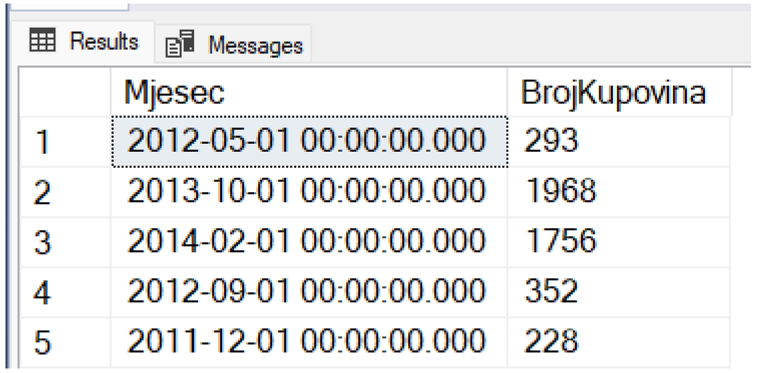


	name	modify_date	MonthModified
1	sp_MSalreadyhavegeneration	2017-08-22 19:38:47.853	2017-08-01 00:00:00.000
2	sp_MSwritemergeperfcounter	2017-08-22 19:38:52.833	2017-08-01 00:00:00.000
3	sp_drop_trusted_assembly	2017-08-22 19:38:10.773	2017-08-01 00:00:00.000
4	TABLE_PRIVILEGES	2017-08-22 19:38:29.143	2017-08-01 00:00:00.000
5	sp_replsetsyncstatus	2017-08-22 19:38:36.773	2017-08-01 00:00:00.000

DATE_BUCKET

```sql
-- Broj kupovina po mjesecima
SELECT
    Mjesec = DATEFROMPARTS(YEAR(OrderDate), MONTH(OrderDate), 1)
    , BrojKupovina = COUNT(*)
FROM
    Sales.SalesOrderHeader
GROUP BY
    DATEFROMPARTS(YEAR(OrderDate), MONTH(OrderDate), 1);
```

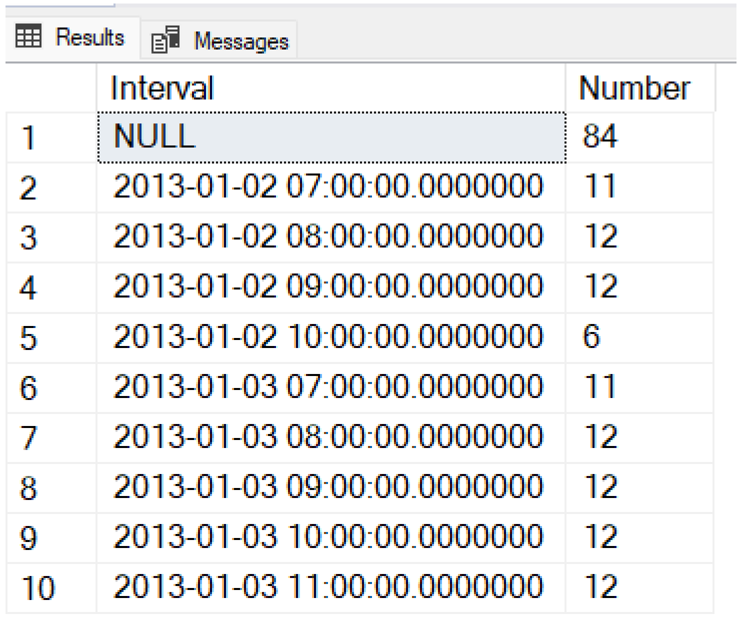
```sql
SELECT
    Mjesec = DATE_BUCKET(MONTH, 1, OrderDate)
    , BrojKupovina = COUNT(*)
FROM
    Sales.SalesOrderHeader
GROUP BY
    DATE_BUCKET(MONTH, 1, OrderDate);
```



	Mjesec	BrojKupovina
1	2012-05-01 00:00:00.000	293
2	2013-10-01 00:00:00.000	1968
3	2014-02-01 00:00:00.000	1756
4	2012-09-01 00:00:00.000	352
5	2011-12-01 00:00:00.000	228

DATE_BUCKET

```
-- Broj isporuka na satnoj razini
SELECT
    Interval = DATE_BUCKET(HOUR, 1, [ConfirmedDeliveryTime])
    , Number = COUNT(*)
FROM
    [Sales].[Invoices]
GROUP BY
    DATE_BUCKET(HOUR, 1, [ConfirmedDeliveryTime]);
```

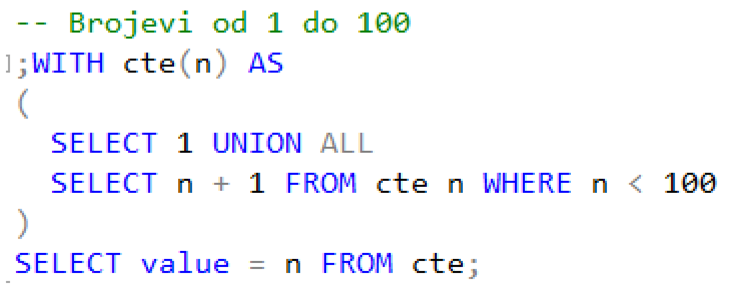


	Interval	Number
1	NULL	84
2	2013-01-02 07:00:00.0000000	11
3	2013-01-02 08:00:00.0000000	12
4	2013-01-02 09:00:00.0000000	12
5	2013-01-02 10:00:00.0000000	6
6	2013-01-03 07:00:00.0000000	11
7	2013-01-03 08:00:00.0000000	12
8	2013-01-03 09:00:00.0000000	12
9	2013-01-03 10:00:00.0000000	12
10	2013-01-03 11:00:00.0000000	12

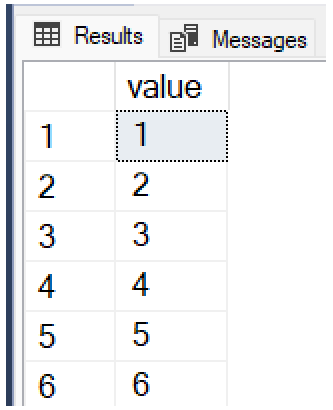
GENERATE_SERIES

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

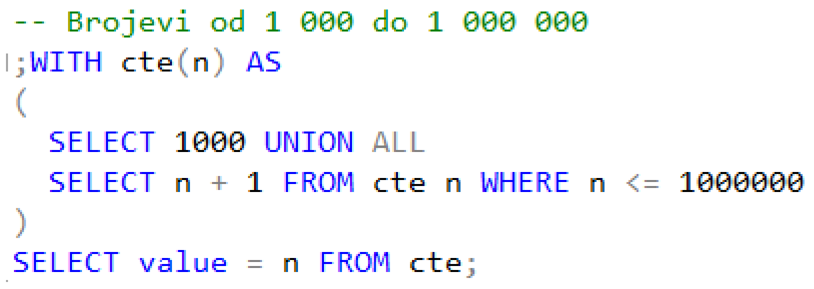
GENERATE_SERIES



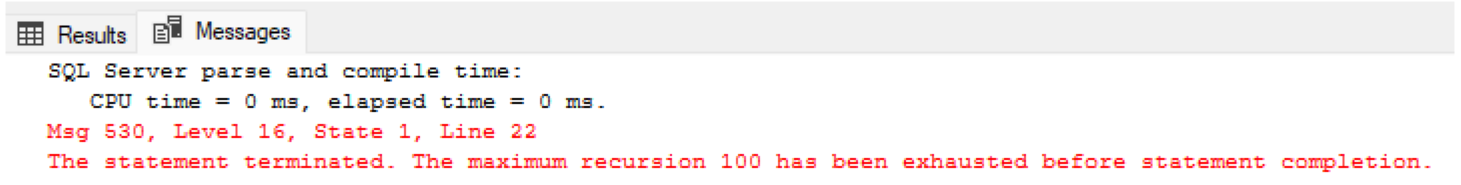
```
-- Brojevi od 1 do 100
;WITH cte(n) AS
(
  SELECT 1 UNION ALL
  SELECT n + 1 FROM cte n WHERE n < 100
)
SELECT value = n FROM cte;
```



	value
1	1
2	2
3	3
4	4
5	5
6	6



```
-- Brojevi 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;
```

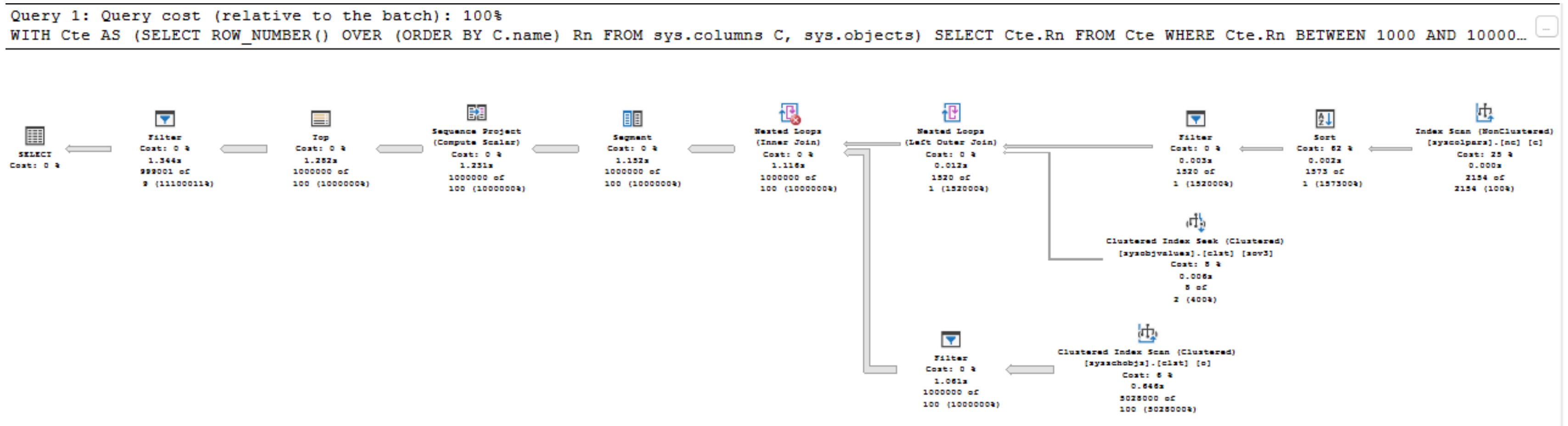


```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
Msg 530, Level 16, State 1, Line 22
The statement terminated. The maximum recursion 100 has been exhausted before statement completion.
```

GENERATE_SERIES

```sql
-- Nađimo nešto iz čega možemo dohvatiti, uzmimo podskup
;WITH Cte AS (
    SELECT ROW_NUMBER() OVER (ORDER BY C.name) Rn FROM sys.columns C, sys.objects--, sys.tables, sys.all_objects
)
SELECT Cte.Rn
FROM Cte
WHERE Cte.Rn BETWEEN 1000 AND 1000000
```

```
Table 'sysschobjs'. Scan count 1, logical reads 117037, physical reads 0...
Table 'sysobjvalues'. Scan count 1520, logical reads 3248, physical reads 0...
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0...
Table 'syscolpars'. Scan count 1, logical reads 22, physical reads 0...

SQL Server Execution Times:
  CPU time = 782 ms,  elapsed time = 3684 ms.
```



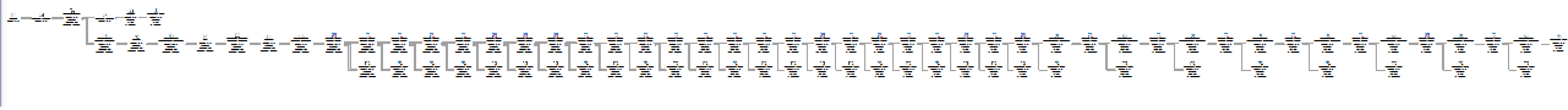
```
Query 1: Query cost (relative to the batch): 100%
WITH Cte AS (SELECT ROW_NUMBER() OVER (ORDER BY C.name) Rn FROM sys.columns C, sys.objects) SELECT Cte.Rn FROM Cte WHERE Cte.Rn BETWEEN 1000 AND 10000…
```

SELECT Cost: 0 %
Filter Cost: 0 % 1.344s 999001 of 9 (11100011%)
Top Cost: 0 % 1.282s 1000000 of 100 (1000000%)
Sequence Project (Compute Scalar) Cost: 0 % 1.231s 1000000 of 100 (1000000%)
Segment Cost: 0 % 1.152s 1000000 of 100 (1000000%)
Nested Loops (Inner Join) Cost: 0 % 1.116s 1000000 of 100 (1000000%)
Nested Loops (Left Outer Join) Cost: 0 % 0.012s 1520 of 1 (152000%)
Filter Cost: 0 % 0.003s 1520 of 1 (152000%)
Sort Cost: 62 % 0.002s 1573 of 1 (157300%)
Index Scan (NonClustered) [syscolpars].[nc] [c] Cost: 25 % 0.000s 2154 of 2154 (100%)
Clustered Index Seek (Clustered) [sysobjvalues].[clst] [sov3] Cost: 8 % 0.006s 8 of 2 (400%)
Filter Cost: 0 % 1.061s 1000000 of 100 (1000000%)
Clustered Index Scan (Clustered) [sysschobjs].[clst] [o] Cost: 6 % 0.646s 5028000 of 100 (5028000%)

GENERATE_SERIES

```sql
-- Funkcija
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
)
```

```
SQL Server Execution Times:
   CPU time = 234 ms,  elapsed time = 3655 ms.
```

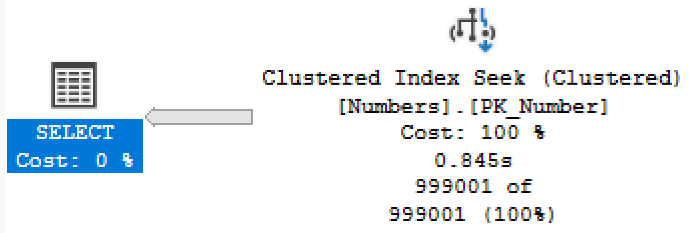


GENERATE_SERIES

```sql
-- Tablica s brojevima
DROP TABLE IF EXISTS [dbo].Numbers;

CREATE TABLE [dbo].Numbers (
Number INT NOT NULL,
 CONSTRAINT [PK_Number] PRIMARY KEY CLUSTERED
(
    [Number] ASC
)
)

INSERT INTO dbo.Numbers (Number)
SELECT Number FROM  [dbo].[NumberRange] (1, 10000000);

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

```
Table 'Numbers'. Scan count 1, logical reads 1615, physical reads 0...

 SQL Server Execution Times:
   CPU time = 32 ms,  elapsed time = 3542 ms.
```

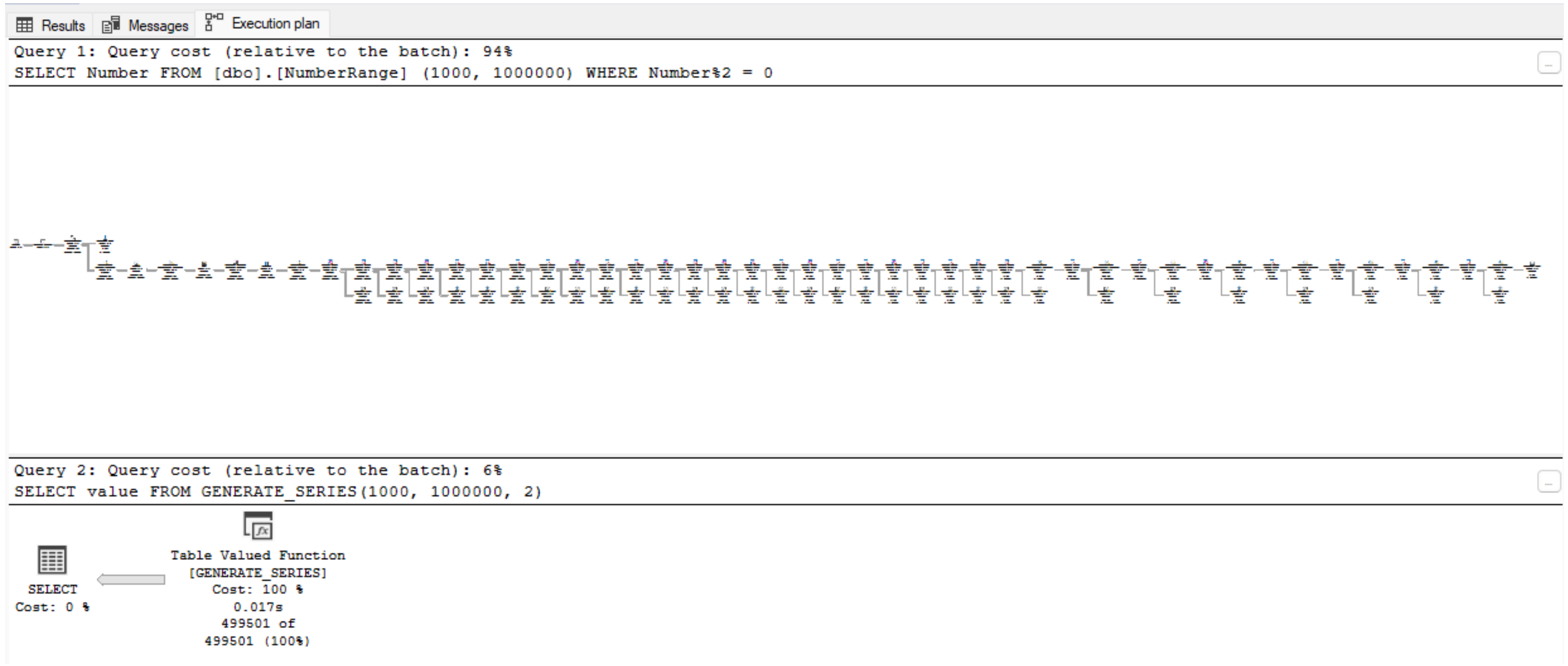
```
Query 1: Query cost (relative to the batch): 100%
SELECT [Number] FROM [dbo].[Numbers] WHERE [NUmber]>=@1 AND [NUmber]<=@2
```



GENERATE_SERIES

```
-- Nova funkcija
SELECT value FROM GENERATE_SERIES(1000, 1000000);

-- Samo parni brojevi
SELECT Number FROM  [dbo].[NumberRange] (1000, 1000000) WHERE Number%2 = 0;
SELECT value FROM GENERATE_SERIES(1000, 1000000, 2);
```

```
CPU time = 15 ms,  elapsed time = 4017 ms.
```



```
Query 1: Query cost (relative to the batch): 94%
SELECT Number FROM [dbo].[NumberRange] (1000, 1000000) WHERE Number%2 = 0
```

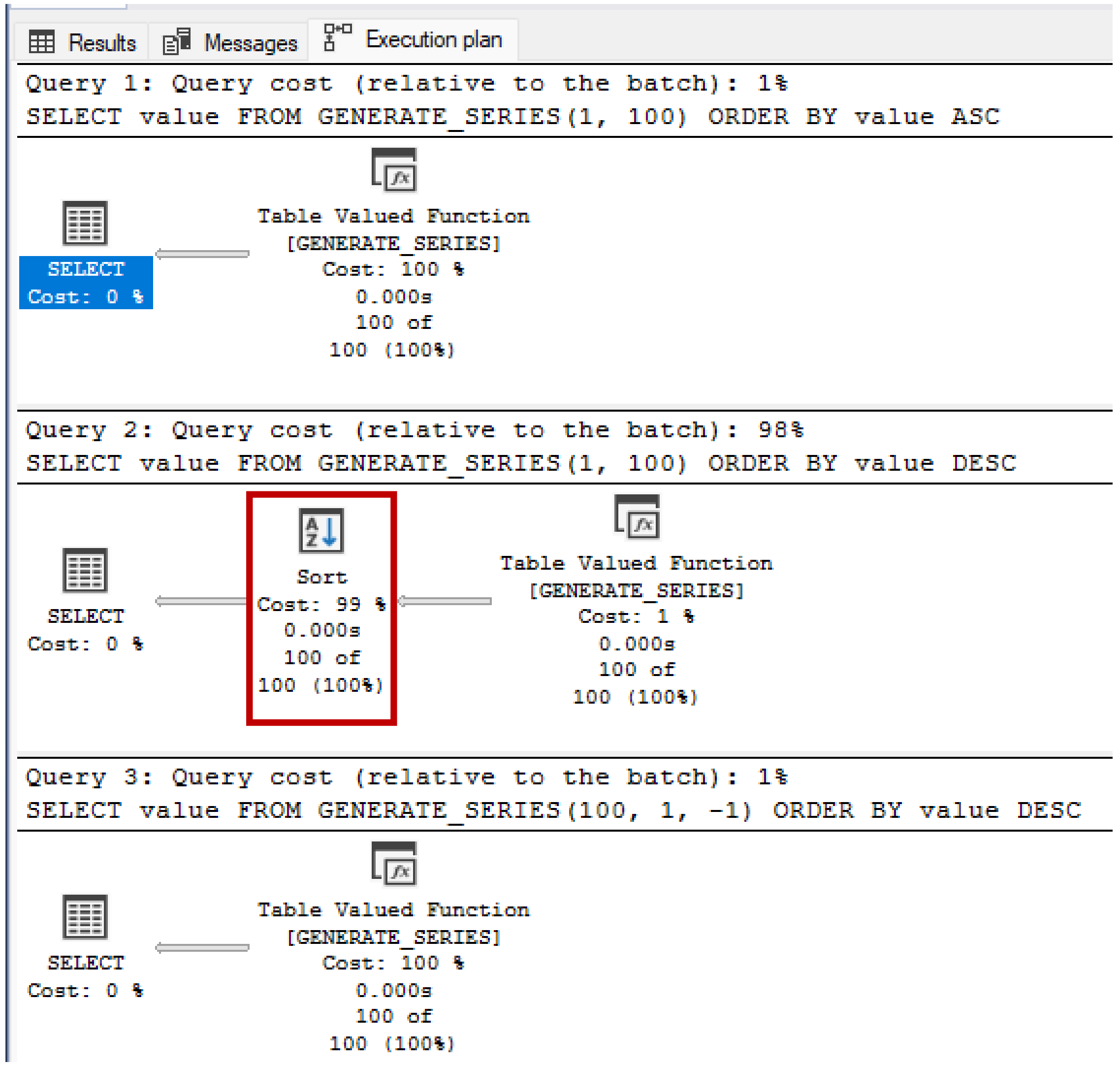
```
Query 2: Query cost (relative to the batch): 6%
SELECT value FROM GENERATE_SERIES(1000, 1000000, 2)
```

GENERATE_SERIES

```sql
-- Sort operator u execution planu
SELECT value FROM GENERATE_SERIES(1, 100)
ORDER BY value ASC;
```

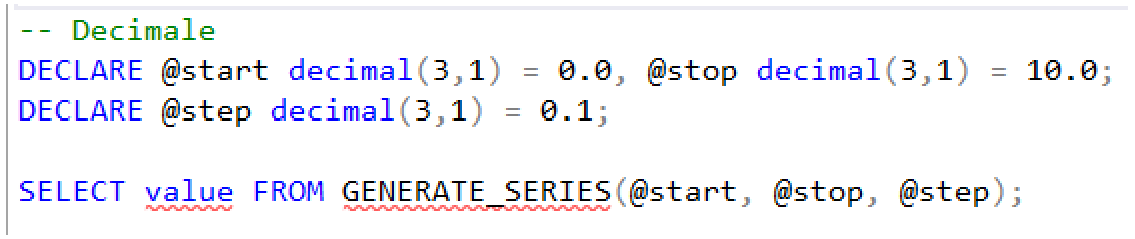
```sql
SELECT value FROM GENERATE_SERIES(1, 100)
ORDER BY value DESC;
```

```sql
-- Silazni niz
SELECT value FROM GENERATE_SERIES(100, 1, -1)
ORDER BY value DESC;
```

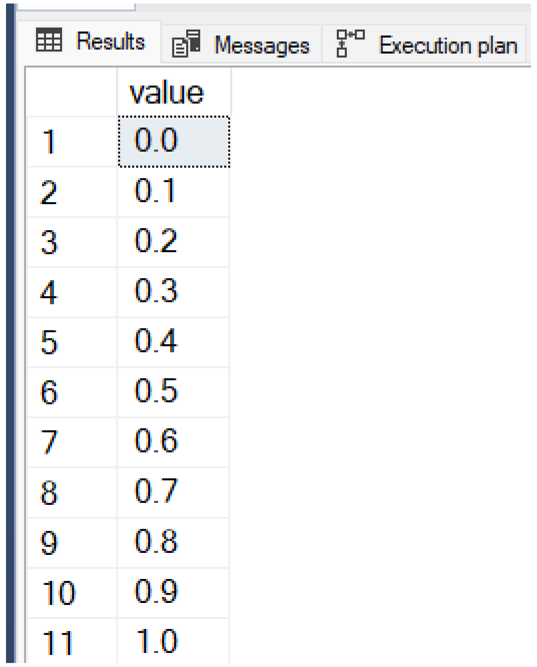


```
Query 1: Query cost (relative to the batch): 1%
SELECT value FROM GENERATE_SERIES(1, 100) ORDER BY value ASC

SELECT                 Table Valued Function
Cost: 0 %              [GENERATE_SERIES]
                       Cost: 100 %
                       0.000s
                       100 of
                       100 (100%)

Query 2: Query cost (relative to the batch): 98%
SELECT value FROM GENERATE_SERIES(1, 100) ORDER BY value DESC

SELECT        Sort          Table Valued Function
Cost: 0 %     Cost: 99 %    [GENERATE_SERIES]
              0.000s        Cost: 1 %
              100 of        0.000s
              100 (100%)    100 of
                            100 (100%)

Query 3: Query cost (relative to the batch): 1%
SELECT value FROM GENERATE_SERIES(100, 1, -1) ORDER BY value DESC

SELECT                 Table Valued Function
Cost: 0 %              [GENERATE_SERIES]
                       Cost: 100 %
                       0.000s
                       100 of
                       100 (100%)
```

GENERATE_SERIES



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

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

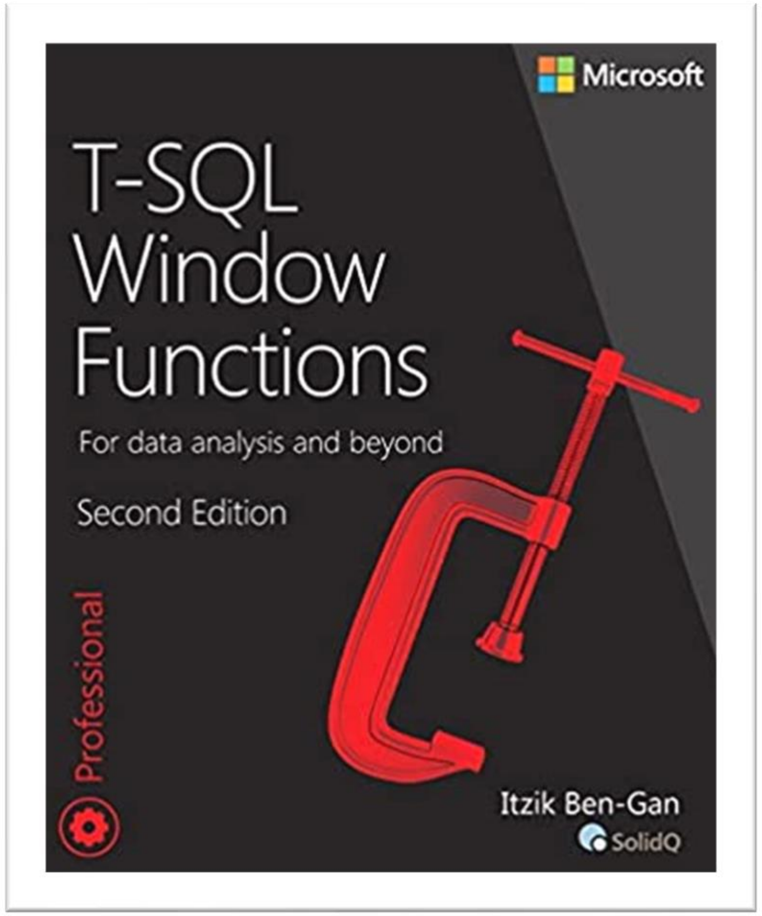


Results Messages Execution plan

	value
1	0.0
2	0.1
3	0.2
4	0.3
5	0.4
6	0.5
7	0.6
8	0.7
9	0.8
10	0.9
11	1.0

WINDOW

Itzik Ben-Gan


T-SQL Window Functions

For data analysis and beyond

Second Edition

Professional

Itzik Ben-Gan

SolidQ

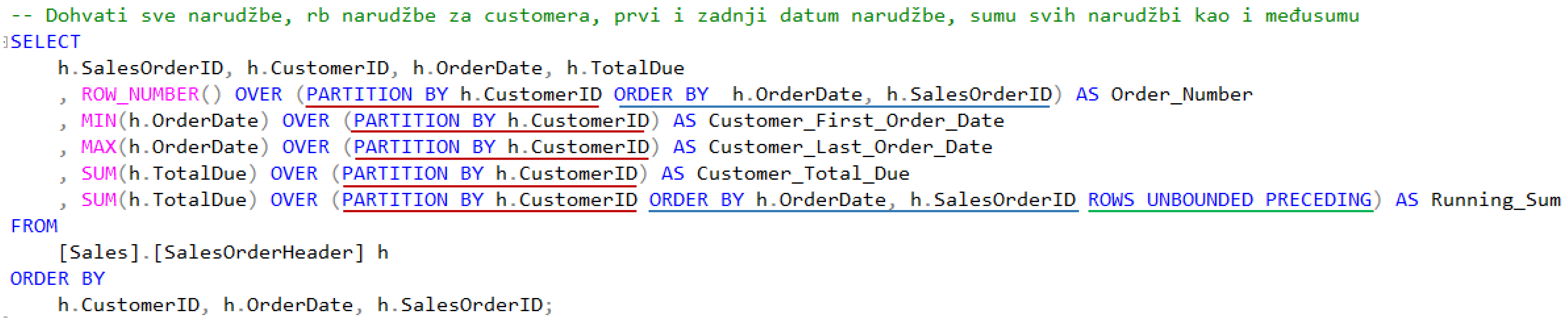
Microsoft

The WINDOW Clause

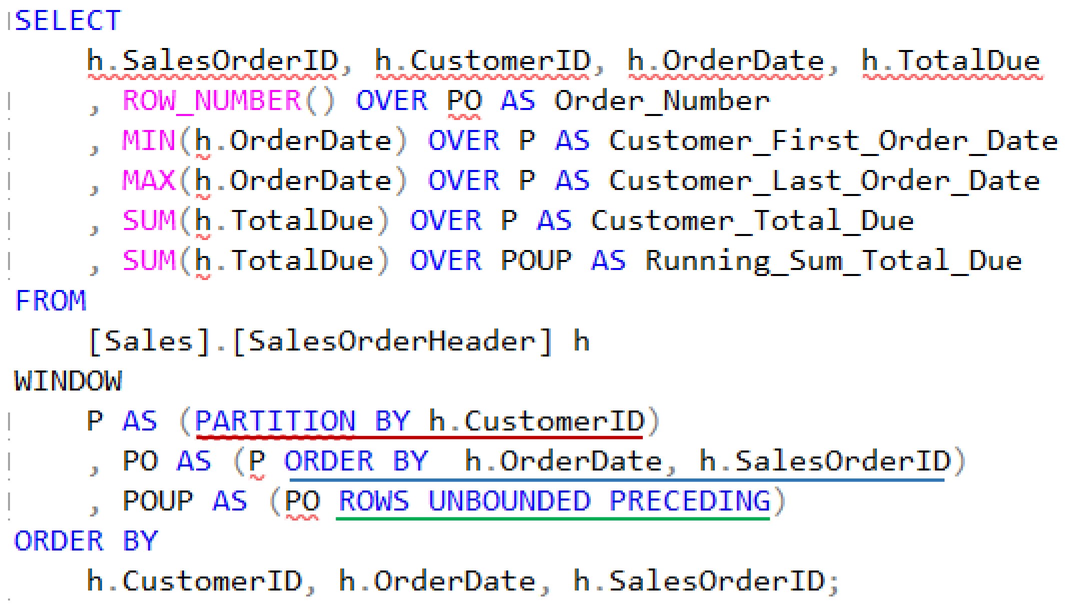
```
WINDOW window_name AS (
      [ reference_window_name ]
      [ <PARTITION BY clause> ]
      [ <ORDER BY clause> ]
      [ <ROW or RANGE clause> ]
     )
```

```
SELECT
FROM
WHERE
GROUP BY
HAVING
WINDOW
ORDER BY
```

The WINDOW Clause



```
-- Dohvati sve narudžbe, rb narudžbe za customera, prvi i zadnji datum narudžbe, sumu svih narudžbi kao i međusumu
SELECT
    h.SalesOrderID, h.CustomerID, h.OrderDate, h.TotalDue
    , ROW_NUMBER() OVER (PARTITION BY h.CustomerID ORDER BY  h.OrderDate, h.SalesOrderID) AS Order_Number
    , MIN(h.OrderDate) OVER (PARTITION BY h.CustomerID) AS Customer_First_Order_Date
    , MAX(h.OrderDate) OVER (PARTITION BY h.CustomerID) AS Customer_Last_Order_Date
    , SUM(h.TotalDue) OVER (PARTITION BY h.CustomerID) AS Customer_Total_Due
    , SUM(h.TotalDue) OVER (PARTITION BY h.CustomerID ORDER BY h.OrderDate, h.SalesOrderID ROWS UNBOUNDED PRECEDING) AS Running_Sum
FROM
    [Sales].[SalesOrderHeader] h
ORDER BY
    h.CustomerID, h.OrderDate, h.SalesOrderID;
```



```
SELECT
    h.SalesOrderID, h.CustomerID, h.OrderDate, h.TotalDue
    , ROW_NUMBER() OVER PO AS Order_Number
    , MIN(h.OrderDate) OVER P AS Customer_First_Order_Date
    , MAX(h.OrderDate) OVER P AS Customer_Last_Order_Date
    , SUM(h.TotalDue) OVER P AS Customer_Total_Due
    , SUM(h.TotalDue) OVER POUP AS Running_Sum_Total_Due
FROM
    [Sales].[SalesOrderHeader] h
WINDOW
    P AS (PARTITION BY h.CustomerID)
    , PO AS (P ORDER BY  h.OrderDate, h.SalesOrderID)
    , POUP AS (PO ROWS UNBOUNDED PRECEDING)
ORDER BY
    h.CustomerID, h.OrderDate, h.SalesOrderID;
```

FIRST_VALUE, LAST_VALUE

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

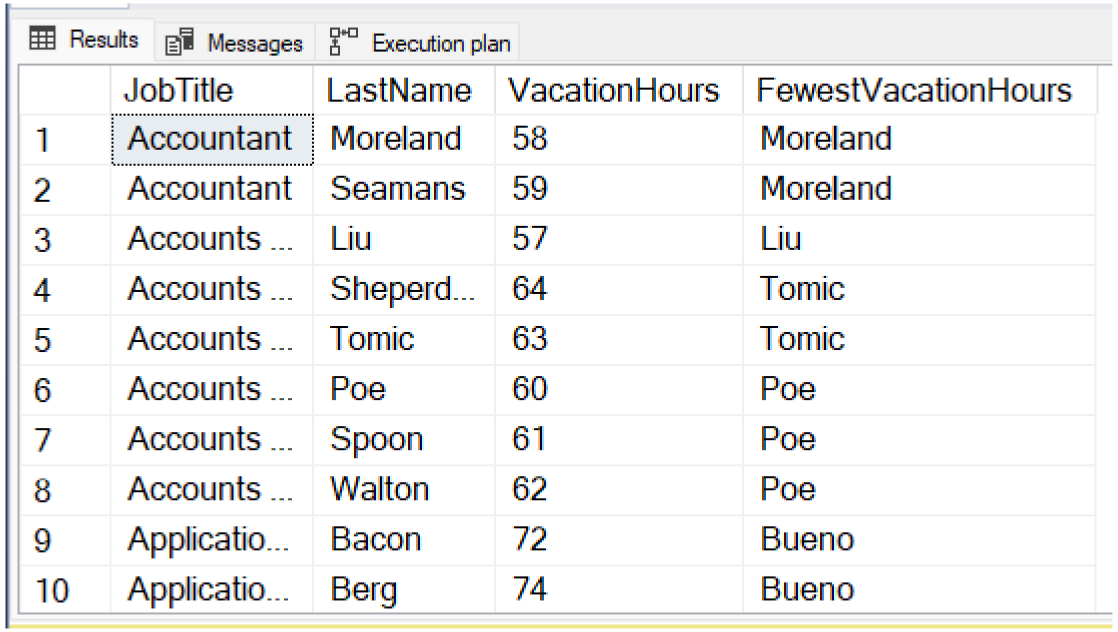
FIRST_VALUE, LAST_VALUE

```sql
-- Dohvati zaposlenike po odjelima i platnim razredima, kada su zaposleni, najmanji i posljednji datum zaposlenja u tom odjelu
;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;
```

```sql
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;
```

FIRST_VALUE, LAST_VALUE

```
-- Zaposlenik s namanje godišnjih odmora unutar jednog odjela
SELECT JobTitle, LastName, VacationHours, FIRST_VALUE(LastName) OVER W AS FewestVacationHours
FROM HumanResources.Employee AS e
INNER JOIN Person.Person AS p
    ON e.BusinessEntityID = p.BusinessEntityID
WINDOW W AS (PARTITION BY JobTitle ORDER BY VacationHours ASC ROWS UNBOUNDED PRECEDING)
ORDER BY JobTitle, LastName;
```

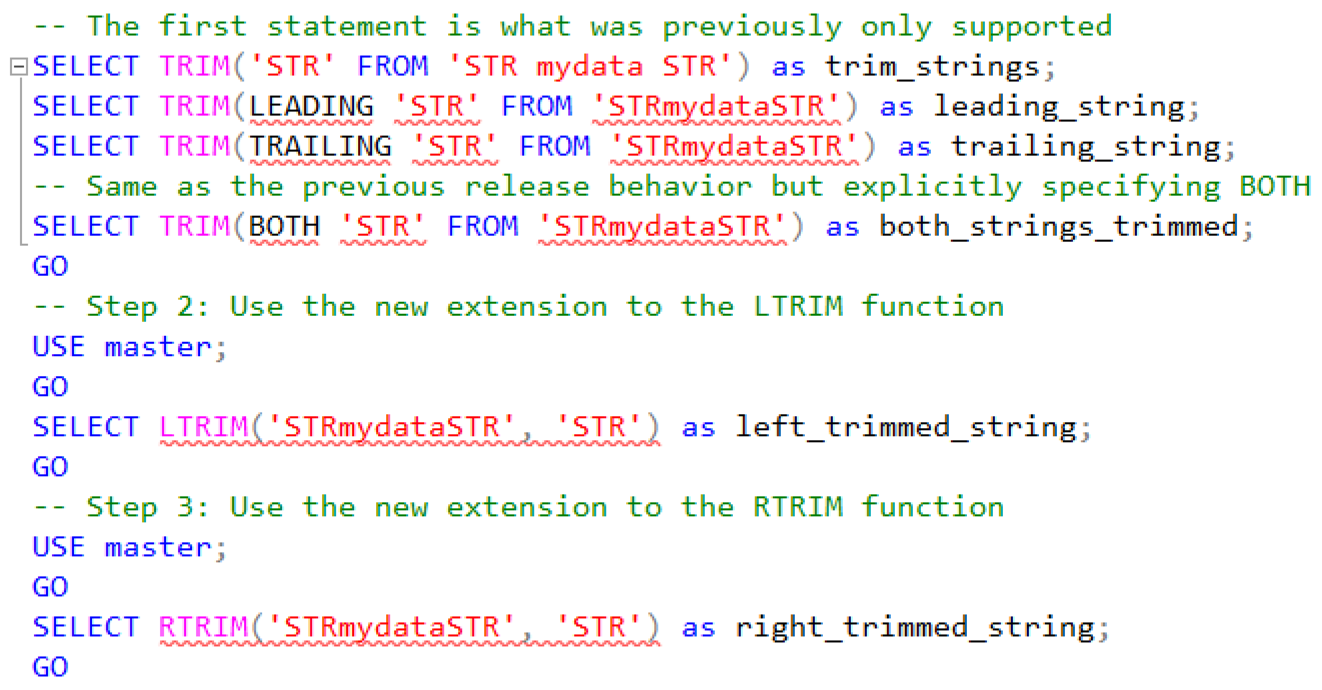


	JobTitle	LastName	VacationHours	FewestVacationHours
1	Accountant	Moreland	58	Moreland
2	Accountant	Seamans	59	Moreland
3	Accounts ...	Liu	57	Liu
4	Accounts ...	Sheperd...	64	Tomic
5	Accounts ...	Tomic	63	Tomic
6	Accounts ...	Poe	60	Poe
7	Accounts ...	Spoon	61	Poe
8	Accounts ...	Walton	62	Poe
9	Applicatio...	Bacon	72	Bueno
10	Applicatio...	Berg	74	Bueno

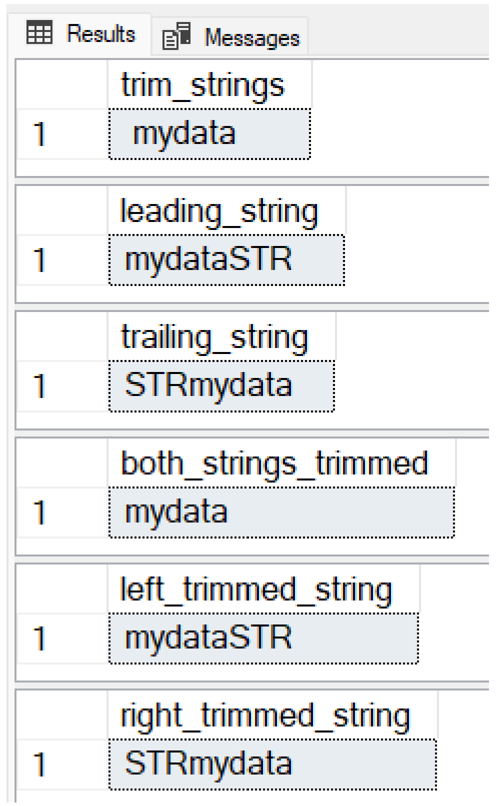
TRIM, LTRIM, RTRIM

```
TRIM ( [ LEADING | TRAILING | BOTH ] [characters FROM ] string )
LTRIM ( character_expression , [ characters ] )
RTRIM ( character_expression , [ characters ] )
```

TRIM, LTRIM, RTRIM



```sql
-- The first statement is what was previously only supported
SELECT TRIM('STR' FROM 'STR mydata STR') as trim_strings;
SELECT TRIM(LEADING 'STR' FROM 'STRmydataSTR') as leading_string;
SELECT TRIM(TRAILING 'STR' FROM 'STRmydataSTR') as trailing_string;
-- Same as the previous release behavior but explicitly specifying BOTH
SELECT TRIM(BOTH 'STR' FROM 'STRmydataSTR') as both_strings_trimmed;
GO
-- Step 2: Use the new extension to the LTRIM function
USE master;
GO
SELECT LTRIM('STRmydataSTR', 'STR') as left_trimmed_string;
GO
-- Step 3: Use the new extension to the RTRIM function
USE master;
GO
SELECT RTRIM('STRmydataSTR', 'STR') as right_trimmed_string;
GO
```



Results Messages

	trim_strings
1	mydata

	leading_string
1	mydataSTR

	trailing_string
1	STRmydata

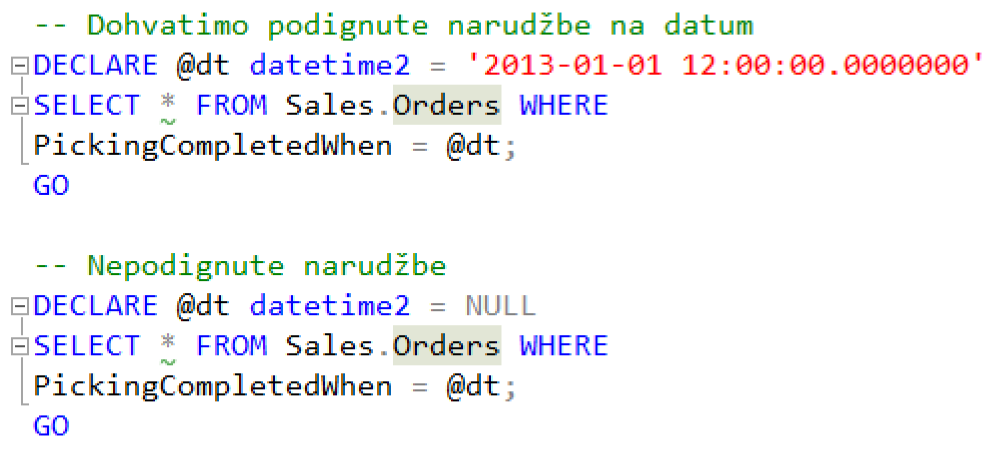
	both_strings_trimmed
1	mydata

	left_trimmed_string
1	mydataSTR

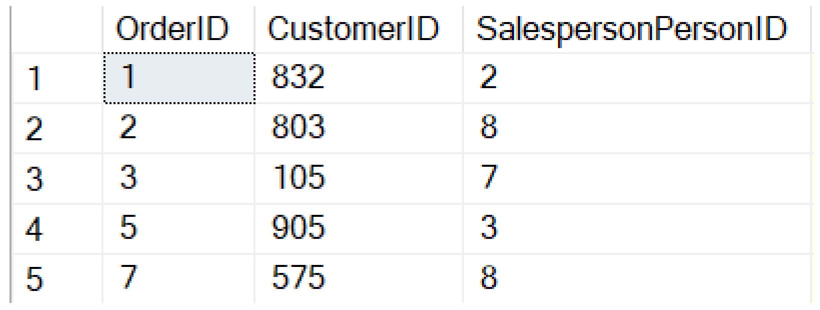
	right_trimmed_string
1	STRmydata

IS NOT DISTINCT (The Distinct Predicate)

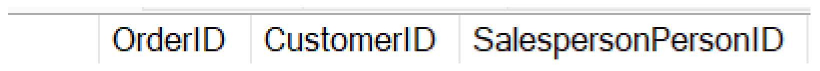
IS [NOT] DISTINCT FROM



```
-- Dohvatimo podignute narudžbe na datum
DECLARE @dt datetime2 = '2013-01-01 12:00:00.0000000'
SELECT * FROM Sales.Orders WHERE
PickingCompletedWhen = @dt;
GO

-- Nepodignute narudžbe
DECLARE @dt datetime2 = NULL
SELECT * FROM Sales.Orders WHERE
PickingCompletedWhen = @dt;
GO
```



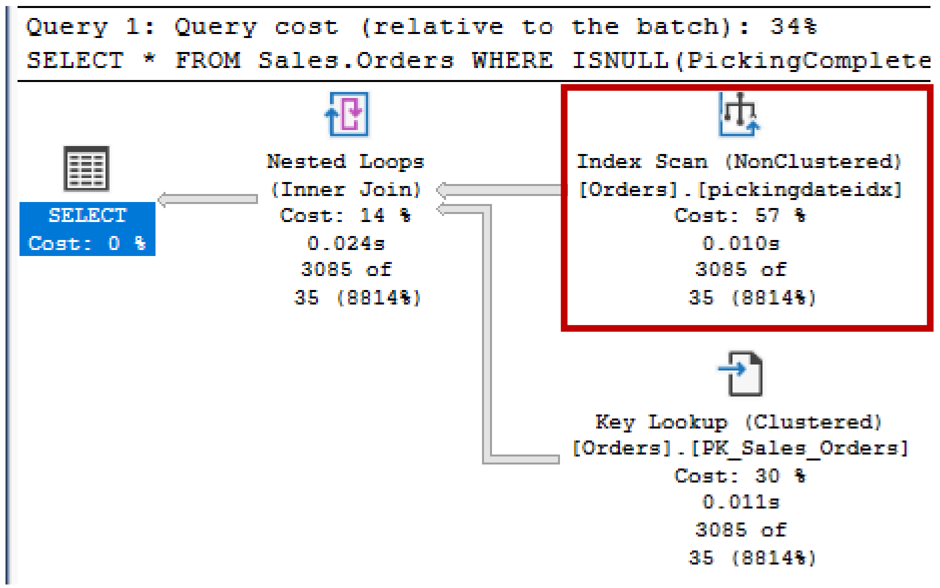
	OrderID	CustomerID	SalespersonPersonID
1	1	832	2
2	2	803	8
3	3	105	7
4	5	905	3
5	7	575	8



	OrderID	CustomerID	SalespersonPersonID

IS NOT DISTINCT (The Distinct Predicate)

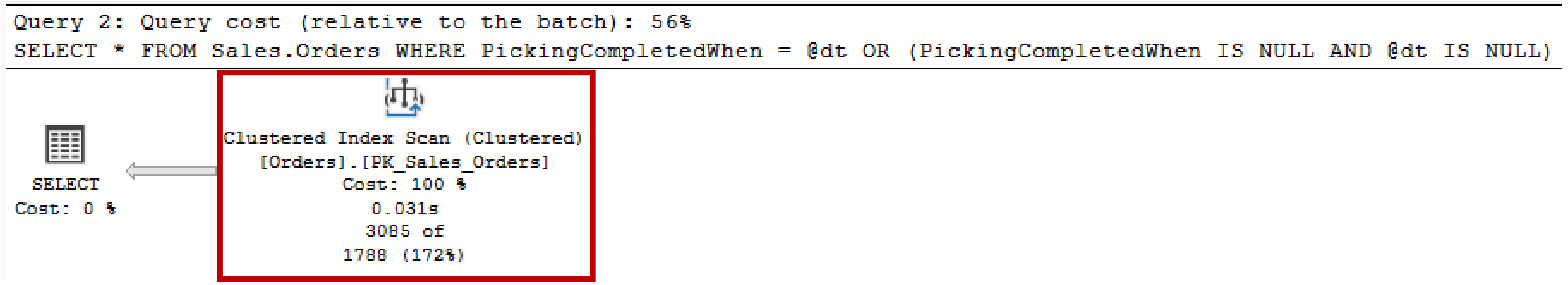
```sql
-- ISNULL - Index scan
DECLARE @dt AS DATE = NULL;
SELECT * FROM Sales.Orders
WHERE ISNULL(PickingCompletedWhen, '99991231') = ISNULL(@dt, '99991231');
```



```
Query 1: Query cost (relative to the batch): 34%
SELECT * FROM Sales.Orders WHERE ISNULL(PickingComplete
```

IS NOT DISTINCT (The Distinct Predicate)

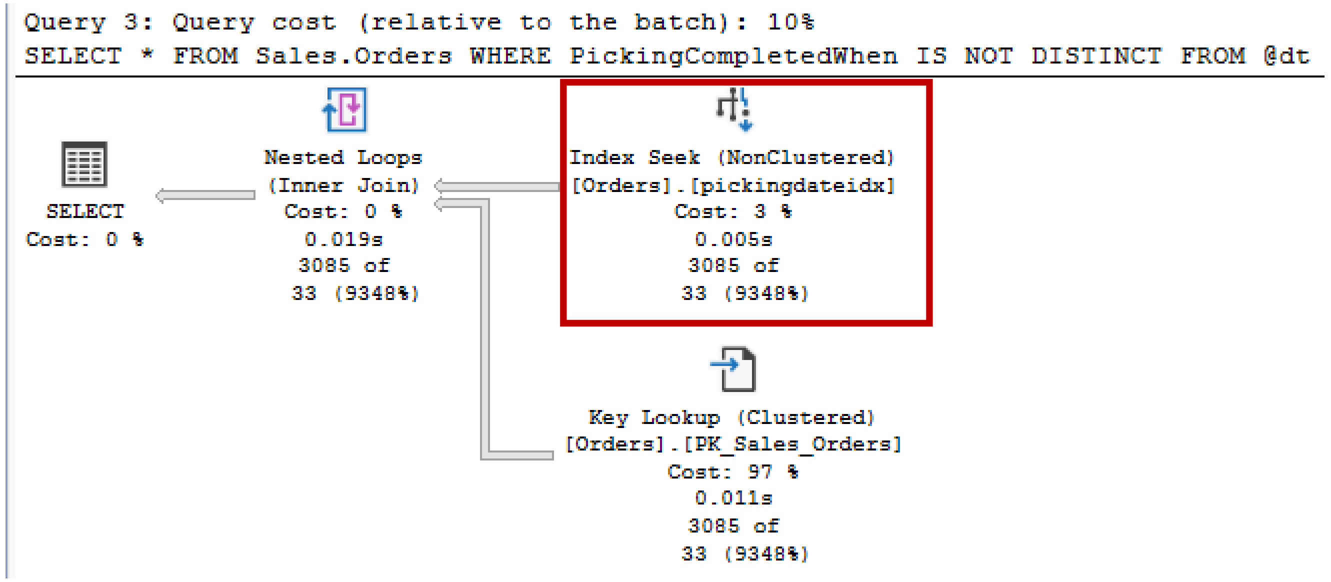
```
-- Kombinacija - Index scan
DECLARE @dt AS DATE = NULL;
SELECT * FROM Sales.Orders
WHERE PickingCompletedWhen = @dt OR (PickingCompletedWhen IS NULL AND @dt IS NULL);
```



```
Query 2: Query cost (relative to the batch): 56%
SELECT * FROM Sales.Orders WHERE PickingCompletedWhen = @dt OR (PickingCompletedWhen IS NULL AND @dt IS NULL)
```

IS NOT DISTINCT (The Distinct Predicate)

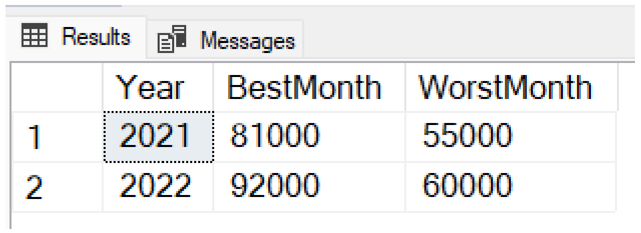
```
-- Novi operator - Index seek!
DECLARE @dt datetime2 = NULL
SELECT *
FROM Sales.Orders
WHERE PickingCompletedWhen IS NOT DISTINCT FROM @dt;
```



```
Query 3: Query cost (relative to the batch): 10%
SELECT * FROM Sales.Orders WHERE PickingCompletedWhen IS NOT DISTINCT FROM @dt
```

GREATEST() & LEAST()

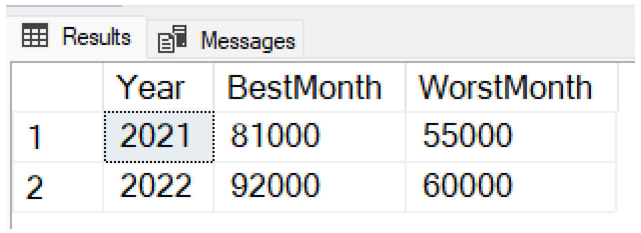
```
CREATE TABLE #SummarizedSales
(
    Year int,
    Jan  int,
    Feb  int,
    Mar  int --,...
);

INSERT #SummarizedSales(Year, Jan, Feb, Mar)
VALUES
(2021, 55000, 81000, 74000),
(2022, 60000, 92000, 86000);

-- CASE, što da imamo 12 ili 20 kolona?
SELECT Year,
  BestMonth = CASE
    WHEN Jan > Feb THEN
      CASE WHEN Jan > Mar THEN Jan ELSE Mar END
    ELSE
      CASE WHEN Mar > Feb THEN Mar ELSE Feb END
    END,
  WorstMonth = CASE
    WHEN Jan < Feb THEN
      CASE WHEN Jan < Mar THEN Jan ELSE Mar END
    ELSE
      CASE WHEN Mar < Feb THEN Mar ELSE Feb END
    END
FROM #SummarizedSales;
```



	Year	BestMonth	WorstMonth
1	2021	81000	55000
2	2022	92000	60000

GREATEST() & LEAST()

```sql
-- UNPIVOT
SELECT Year,
  BestMonth  = MAX(Months.MonthlyTotal),
  WorstMonth = MIN(Months.MonthlyTotal)
FROM #SummarizedSales AS s
UNPIVOT
(
  MonthlyTotal FOR [Month] IN ([Jan],[Feb],[Mar])
) AS Months
GROUP BY Year;

-- CROSS APPLY
SELECT Year,
  BestMonth  = MAX(MonthlyTotal),
  WorstMonth = MIN(MonthlyTotal)
FROM
(
  SELECT s.Year, Months.MonthlyTotal
  FROM #SummarizedSales AS s
  CROSS APPLY (VALUES([Jan]),([Feb]),([Mar])) AS [Months](MonthlyTotal)
) AS Sales
GROUP BY Year;

-- GREATEST, LEAST
SELECT Year,
  BestMonth  = GREATEST([Jan],[Feb],[Mar]),
  WorstMonth = LEAST   ([Jan],[Feb],[Mar])
FROM #SummarizedSales;
```



	Year	BestMonth	WorstMonth
1	2021	81000	55000
2	2022	92000	60000

Approximate Percentile Functions

PERCENTILE_CONT i PERCENTILE_DISC - SQL Server 2005 and later

APPROX_PERCENTILE_CONT i APPROX_PERCENTILE_DISC

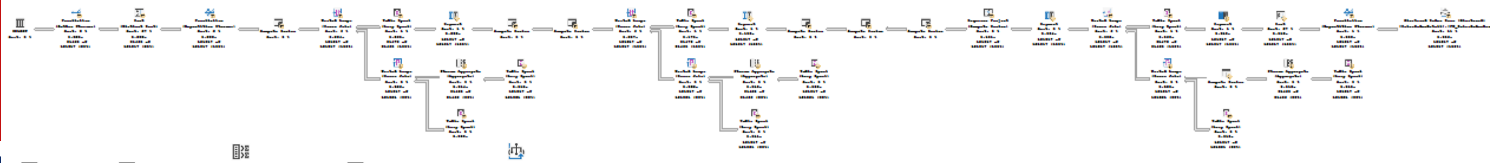
Approximate Percentile Functions

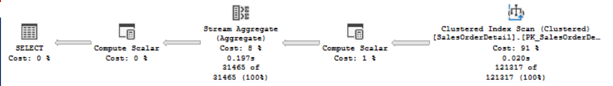
```
SELECT DISTINCT
  [SalesOrderID]
, PERCENTILE_CONT (0.5) WITHIN GROUP (ORDER BY [OrderQty]) OVER (PARTITION BY [SalesOrderID]) AS medianscore_cont
, PERCENTILE_DISC (0.5) WITHIN GROUP (ORDER BY [OrderQty]) OVER (PARTITION BY [SalesOrderID]) AS medianscore_disc
FROM [Sales].[SalesOrderDetail] ORDER BY SalesOrderID;
```

```
SELECT DISTINCT
  [SalesOrderID]
, APPROX_PERCENTILE_CONT (0.5) WITHIN GROUP (ORDER BY [OrderQty])  AS medianscore_cont
, APPROX_PERCENTILE_DISC (0.5) WITHIN GROUP (ORDER BY [OrderQty]) AS medianscore_disc
FROM [Sales].[SalesOrderDetail]
GROUP BY [SalesOrderID]
ORDER BY SalesOrderID;
```

```
Table 'SalesOrderDetail'. Scan count 9, logical reads 1313, physical reads 0...
Table 'Worktable'. Scan count 72, logical reads 1105506, physical reads 0...

 SQL Server Execution Times:
   CPU time = 2046 ms,  elapsed time = 531 ms.
```

```
Table 'SalesOrderDetail'. Scan count 1, logical reads 1248, physical reads 0...

 SQL Server Execution Times:
   CPU time = 62 ms,  elapsed time = 420 ms.
```





JSON enhancements

SQL 2016

- Funkcije - ISJSON, JSON_VALUE, JSON_QUERY, JSON_MODIFY
- Operatori - FOR JSON i OPENJSON

JSON enhancements

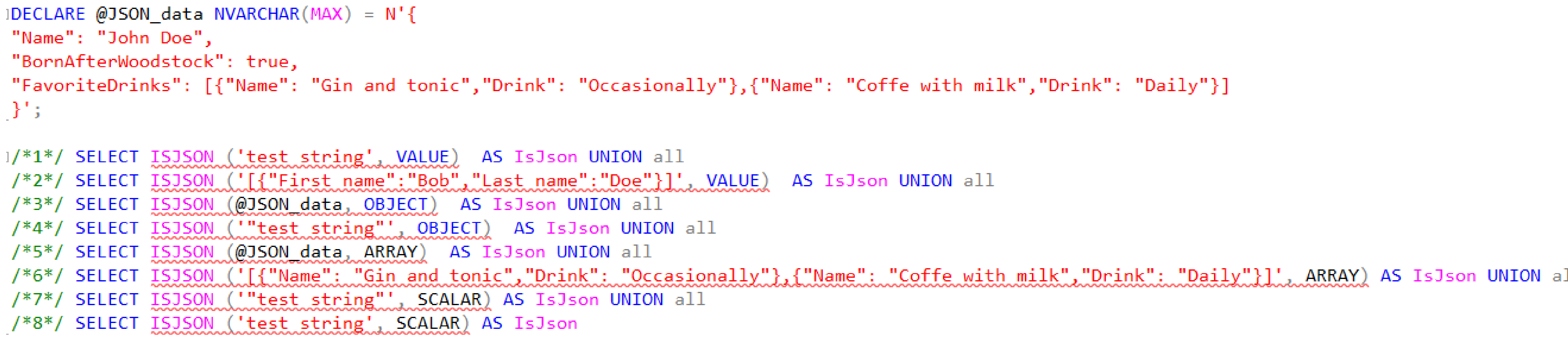
SQL 2016

- Funkcije - ISJSON, JSON_VALUE, JSON_QUERY, JSON_MODIFY
- Operatori - FOR JSON i OPENJSON

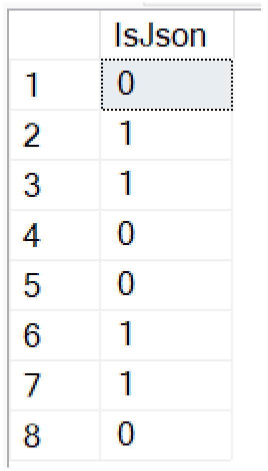
SQL 2022

- ISJSON
- JSON_PATH_EXISTS
- JSON_OBJECT
- JSON_ARRAY

JSON enhancements

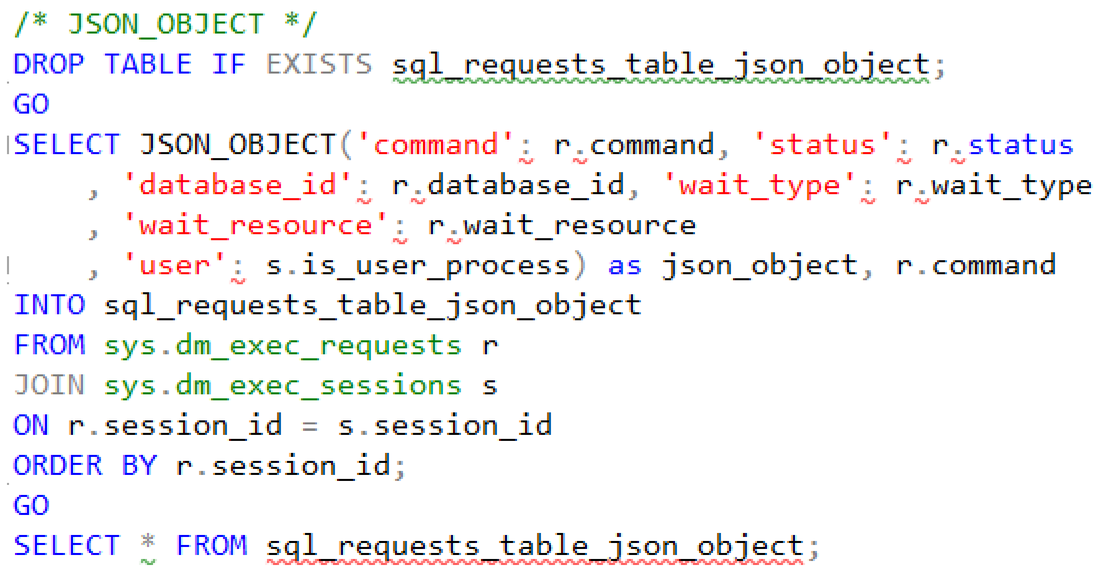


```
DECLARE @JSON_data NVARCHAR(MAX) = N'{
"Name": "John Doe",
"BornAfterWoodstock": true,
"FavoriteDrinks": [{"Name": "Gin and tonic","Drink": "Occasionally"},{"Name": "Coffe with milk","Drink": "Daily"}]
}';

/*1*/ SELECT ISJSON ('test string', VALUE)  AS IsJson UNION all
/*2*/ SELECT ISJSON ('[{"First name":"Bob","Last name":"Doe"}]', VALUE)  AS IsJson UNION all
/*3*/ SELECT ISJSON (@JSON_data, OBJECT)  AS IsJson UNION all
/*4*/ SELECT ISJSON ('"test string"', OBJECT)  AS IsJson UNION all
/*5*/ SELECT ISJSON (@JSON_data, ARRAY)  AS IsJson UNION all
/*6*/ SELECT ISJSON ('[{"Name": "Gin and tonic","Drink": "Occasionally"},{"Name": "Coffe with milk","Drink": "Daily"}]', ARRAY) AS IsJson UNION al
/*7*/ SELECT ISJSON ('"test string"', SCALAR) AS IsJson UNION all
/*8*/ SELECT ISJSON ('test string', SCALAR) AS IsJson
```

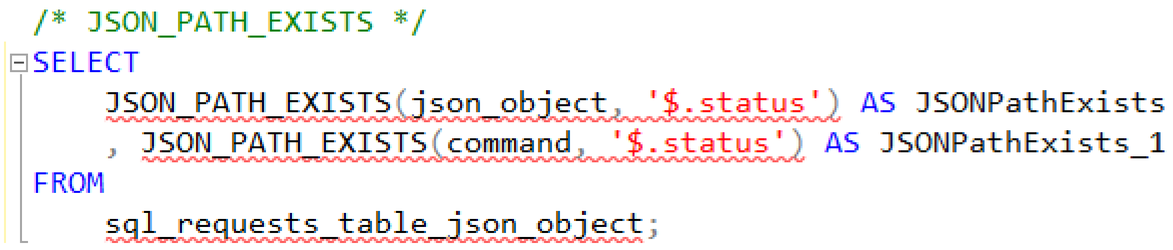


	IsJson
1	0
2	1
3	1
4	0
5	0
6	1
7	1
8	0

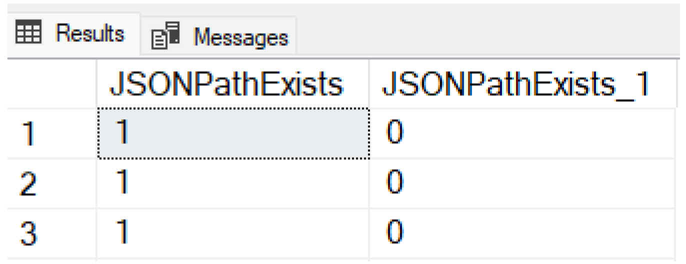
JSON enhancements



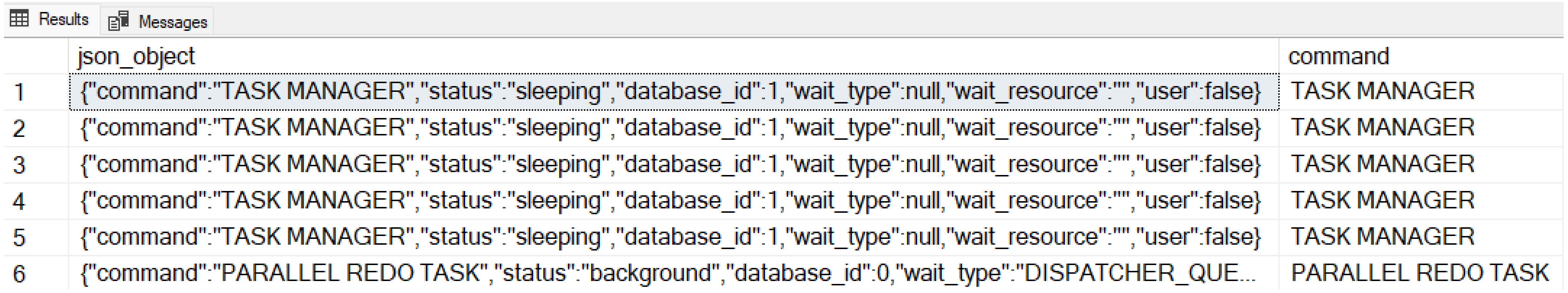
```sql
/* JSON_OBJECT */
DROP TABLE IF EXISTS sql_requests_table_json_object;
GO
SELECT JSON_OBJECT('command': r.command, 'status': r.status
     , 'database_id': r.database_id, 'wait_type': r.wait_type
     , 'wait_resource': r.wait_resource
     , 'user': s.is_user_process) as json_object, r.command
INTO sql_requests_table_json_object
FROM sys.dm_exec_requests r
JOIN sys.dm_exec_sessions s
ON r.session_id = s.session_id
ORDER BY r.session_id;
GO
SELECT * FROM sql_requests_table_json_object;
```



```sql
/* JSON_PATH_EXISTS */
SELECT
    JSON_PATH_EXISTS(json_object, '$.status') AS JSONPathExists
    , JSON_PATH_EXISTS(command, '$.status') AS JSONPathExists_1
FROM
    sql_requests_table_json_object;
```

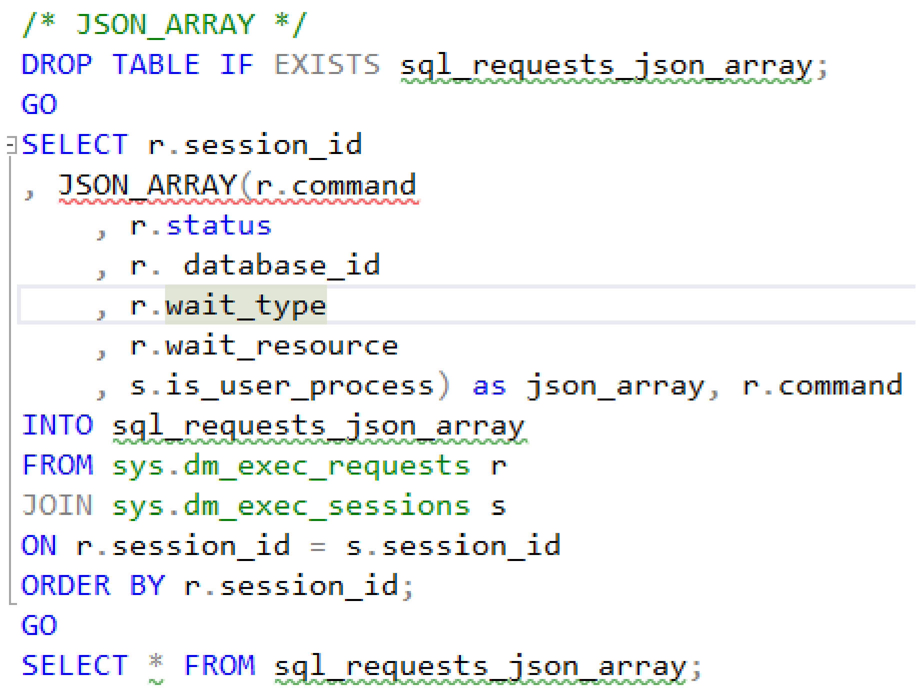


	JSONPathExists	JSONPathExists_1
1	1	0
2	1	0
3	1	0

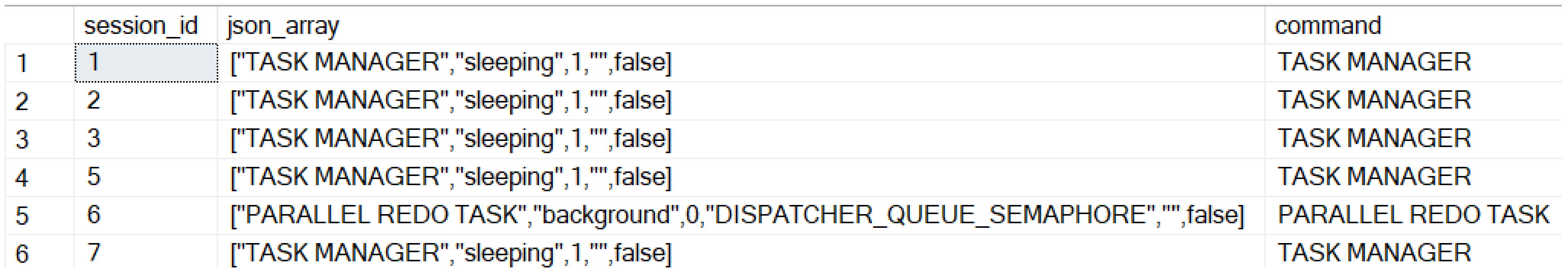


	json_object	command
1	{"command":"TASK MANAGER","status":"sleeping","database_id":1,"wait_type":null,"wait_resource":"","user":false}	TASK MANAGER
2	{"command":"TASK MANAGER","status":"sleeping","database_id":1,"wait_type":null,"wait_resource":"","user":false}	TASK MANAGER
3	{"command":"TASK MANAGER","status":"sleeping","database_id":1,"wait_type":null,"wait_resource":"","user":false}	TASK MANAGER
4	{"command":"TASK MANAGER","status":"sleeping","database_id":1,"wait_type":null,"wait_resource":"","user":false}	TASK MANAGER
5	{"command":"TASK MANAGER","status":"sleeping","database_id":1,"wait_type":null,"wait_resource":"","user":false}	TASK MANAGER
6	{"command":"PARALLEL REDO TASK","status":"background","database_id":0,"wait_type":"DISPATCHER_QUE...	PARALLEL REDO TASK

JSON enhancements



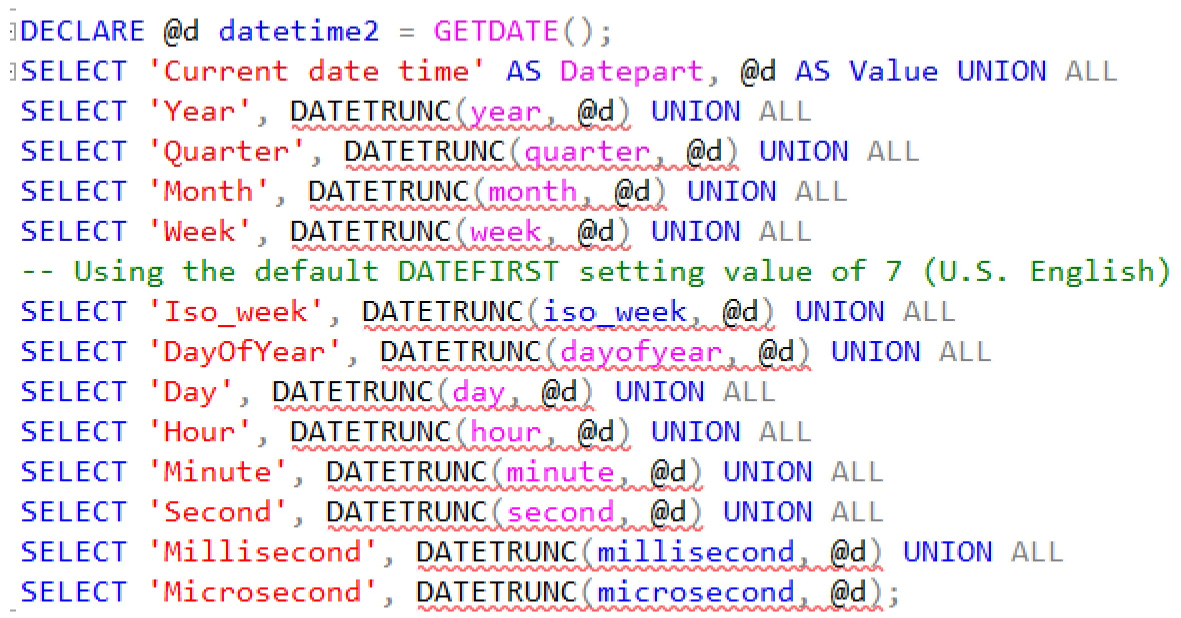
```sql
/* JSON_ARRAY */
DROP TABLE IF EXISTS sql_requests_json_array;
GO
SELECT r.session_id
, JSON_ARRAY(r.command
    , r.status
    , r. database_id
    , r.wait_type
    , r.wait_resource
    , s.is_user_process) as json_array, r.command
INTO sql_requests_json_array
FROM sys.dm_exec_requests r
JOIN sys.dm_exec_sessions s
ON r.session_id = s.session_id
ORDER BY r.session_id;
GO
SELECT * FROM sql_requests_json_array;
```



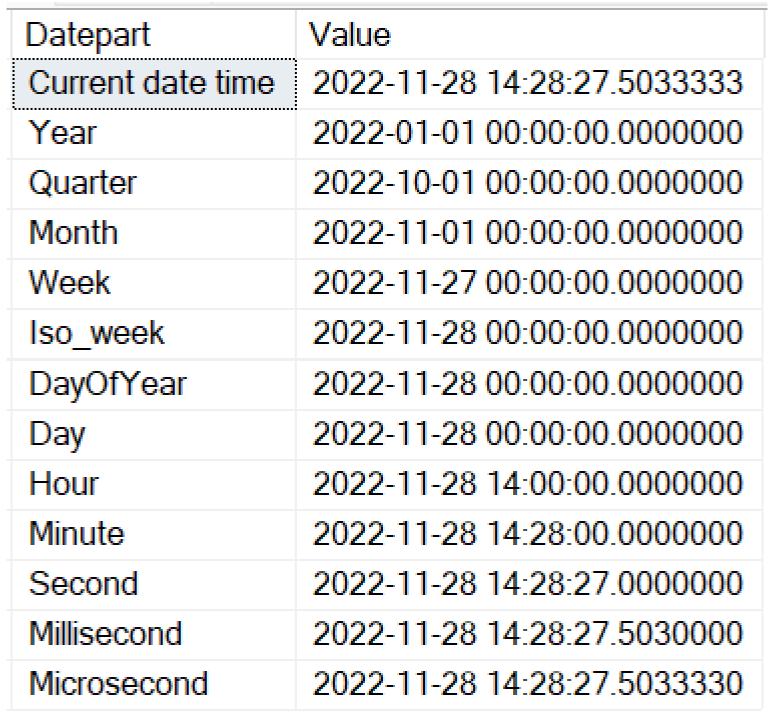
	session_id	json_array	command
1	1	["TASK MANAGER","sleeping",1,"",false]	TASK MANAGER
2	2	["TASK MANAGER","sleeping",1,"",false]	TASK MANAGER
3	3	["TASK MANAGER","sleeping",1,"",false]	TASK MANAGER
4	5	["TASK MANAGER","sleeping",1,"",false]	TASK MANAGER
5	6	["PARALLEL REDO TASK","background",0,"DISPATCHER_QUEUE_SEMAPHORE","",false]	PARALLEL REDO TASK
6	7	["TASK MANAGER","sleeping",1,"",false]	TASK MANAGER

DATETRUNC

DATETRUNC (datepart, date)



```
DECLARE @d datetime2 = GETDATE();
SELECT 'Current date time' AS Datepart, @d AS Value UNION ALL
SELECT 'Year', DATETRUNC(year, @d) UNION ALL
SELECT 'Quarter', DATETRUNC(quarter, @d) UNION ALL
SELECT 'Month', DATETRUNC(month, @d) UNION ALL
SELECT 'Week', DATETRUNC(week, @d) UNION ALL
-- Using the default DATEFIRST setting value of 7 (U.S. English)
SELECT 'Iso_week', DATETRUNC(iso_week, @d) UNION ALL
SELECT 'DayOfYear', DATETRUNC(dayofyear, @d) UNION ALL
SELECT 'Day', DATETRUNC(day, @d) UNION ALL
SELECT 'Hour', DATETRUNC(hour, @d) UNION ALL
SELECT 'Minute', DATETRUNC(minute, @d) UNION ALL
SELECT 'Second', DATETRUNC(second, @d) UNION ALL
SELECT 'Millisecond', DATETRUNC(millisecond, @d) UNION ALL
SELECT 'Microsecond', DATETRUNC(microsecond, @d);
```



Datepart	Value
Current date time	2022-11-28 14:28:27.5033333
Year	2022-01-01 00:00:00.0000000
Quarter	2022-10-01 00:00:00.0000000
Month	2022-11-01 00:00:00.0000000
Week	2022-11-27 00:00:00.0000000
Iso_week	2022-11-28 00:00:00.0000000
DayOfYear	2022-11-28 00:00:00.0000000
Day	2022-11-28 00:00:00.0000000
Hour	2022-11-28 14:00:00.0000000
Minute	2022-11-28 14:28:00.0000000
Second	2022-11-28 14:28:27.0000000
Millisecond	2022-11-28 14:28:27.5030000
Microsecond	2022-11-28 14:28:27.5033330

Hvala ;)