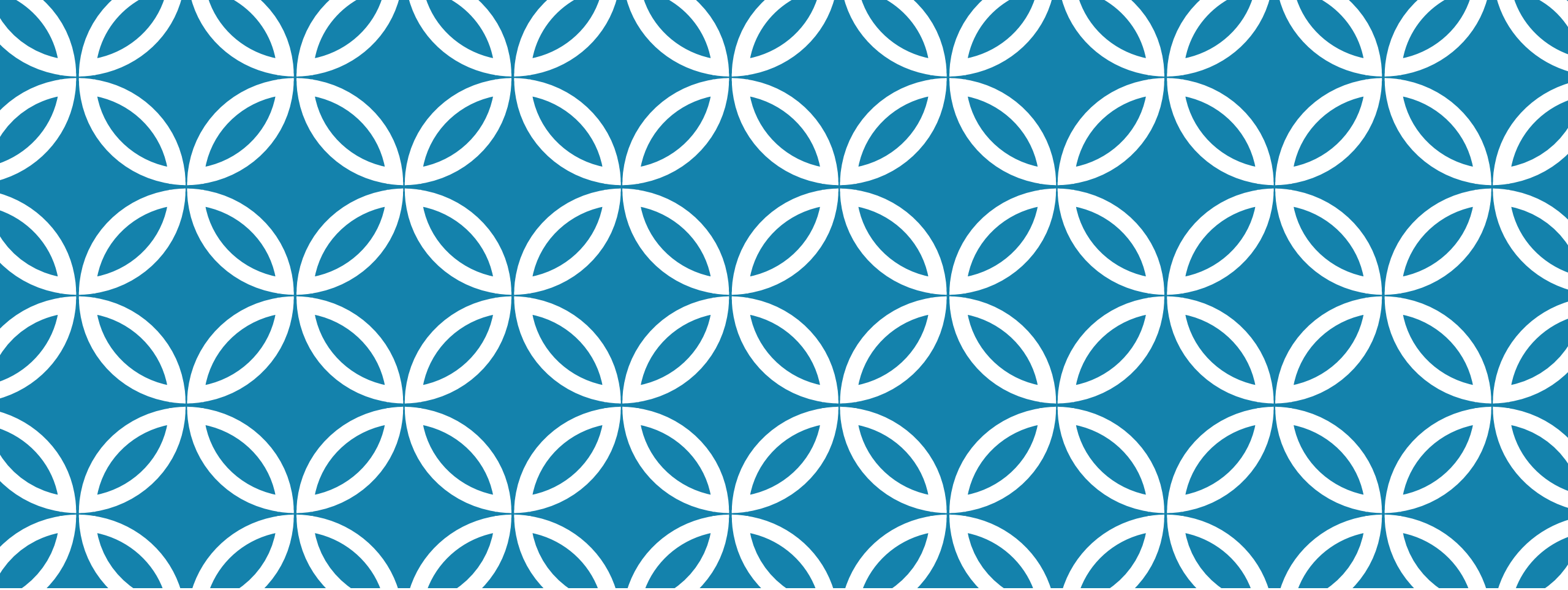


BEYOND THE BASIC SELECT

Deborah Melkin
Calgary Data Platform UG
April 28, 2021

BEYOND THE BASICS: ME

- 20+ years as a DBA
- Mainly work with SQL Server
- Mainly work with OLTP but have worked with some data marts.
- NESQL Board Member
- Data Platform Community Speaker
- IDERA ACE Class of 2020
- Speaker Idol Winner 2019
- Microsoft Data Platform MVP (2020-)

Random facts:

- I'm the alto section leader in my choir.
- I go to bluegrass jams regularly.
- I've been learning guitar and now mandolin.
- I am a bit of a musical theater geek.

WHO DID NOT EXECUTE A SELECT STATEMENT TODAY?

```
SELECT sod.SalesOrderID
FROM Sales.SalesOrderDetail as sod
```

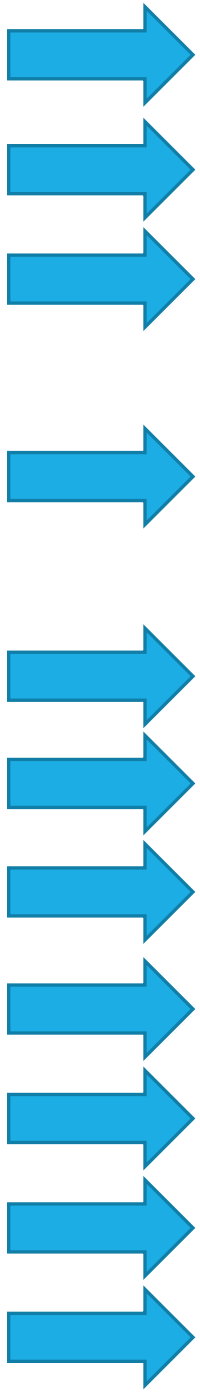
```
;
WITH employee_cte AS
(
SELECT e.BusinessEntityID, e.JobTitle,
e.OrganizationNode.ToString() as OrgChart,
ManagerBusinessEntityID,
ISNULL(e.OrganizationNode.ToString(), '0') as
ManagerOrgChart
FROM HumanResources.Employee e
WHERE e.ManagerBusinessEntityID IS NULL
UNION ALL
SELECT hre.BusinessEntityID, hre.JobTitle,
hre.OrganizationNode.ToString() as OrgChart,
hre.ManagerBusinessEntityID,
ISNULL(cte.ManagerOrgChart, '') + ' | ' +
REPLACE(hre.OrganizationNode.ToString(), '/', '')
FROM HumanResources.Employee hre
JOIN employee_cte cte ON
hre.ManagerBusinessEntityID = cte.BusinessEntityID
)
SELECT
cte.BusinessEntityID, p.FirstName, p.LastName,
cte.JobTitle, sp.TerritoryName as SalesTerritory,
sp.TerritoryGroup,
mp.FirstName as ManagerFirstName, mp.LastName as
ManagerLastName, cte.ManagerOrgChart,
orders.CustomerID, orders.AccountNumber,
orders.StoreID, orders.TerritoryID,
orders.TotalSalesOrdersPerSalesPerson,
orders.TotalDue, orders.TotalDueForCustomer,
orders.AvgSalesOrderAmt,
orders.TotalNumberOfShipments,
orders.TotalSalesOrderQuanity,
orders.WeightedAvgPricePerQty,
orders.AvgNumberOfProducts,
orders.MaxNumberOfProducts,
orders.MinNumberOfProducts
FROM employee_cte cte
JOIN Person.Person p ON cte.BusinessEntityID =
p.BusinessEntityID
JOIN Sales.vSalesPerson sp ON cte.BusinessEntityID
= sp.BusinessEntityID
JOIN (
SELECT
c.CustomerID, c.AccountNumber, c.StoreID,
c.TerritoryID,
soh.SalesPersonID,
COUNT(soh.SalesOrderID) OVER(PARTITION BY
c.CustomerID, soh.SalesPersonID) as
TotalSalesOrdersPerSalesPerson,
SUM(soh.TotalDue) OVER(PARTITION BY c.CustomerID,
soh.SalesPersonID) as TotalDue,
SUM(soh.TotalDue) OVER(PARTITION BY c.CustomerID)
as TotalDueForCustomer,
SUM(soh.TotalDue) OVER(PARTITION BY c.CustomerID,
soh.SalesPersonID)
/Count(soh.SalesOrderID) OVER(PARTITION BY
c.CustomerID, soh.SalesPersonID) as
AvgSalesOrderAmt,
SUM(details.NumberOfShipments) OVER(PARTITION BY
c.CustomerID, soh.SalesPersonID) as
TotalNumberOfShipments,
SUM(details.TotalQuantity) OVER(PARTITION BY
c.CustomerID, soh.SalesPersonID) as
TotalSalesOrderQuanity,
SUM(details.WeightedAvgPrice) OVER(PARTITION BY
c.CustomerID, soh.SalesPersonID)
/COUNT(soh.SalesOrderID) OVER(PARTITION BY
c.CustomerID, soh.SalesPersonID) as
WeightedAvgPricePerQty,
AVG(details.AvgOrderQty) OVER(PARTITION BY
c.CustomerID) as AvgNumberOfProducts,
MAX(details.MaxOrderQty) OVER(PARTITION BY
c.CustomerID) as MaxNumberOfProducts,
MIN(details.MinOrderQty) OVER(PARTITION BY
c.CustomerID) as MinNumberOfProducts
FROM Sales.Customer c
JOIN Sales.SalesOrderHeader soh ON soh.CustomerID =
c.CustomerID
JOIN (
SELECT sod.SalesOrderID,
COUNT(distinct CarrierTrackingNumber) as
NumberofShipments,
COUNT(distinct ProductID) as NumberofProducts,
SUM(sod.OrderQty) as TotalQuantity,
AVG(sod.OrderQty) as AvgQuantityPerProduct,
SUM(sod.OrderQty * sod.UnitPrice) as SubTotal,
SUM(sod.UnitPriceDiscount * sod.OrderQty) as
TotalUnitDiscount,
SUM([dbo].[ufnGetProductListPrice](sod.ProductID,
sod.ModifiedDate) * sod.OrderQty)
as TotalUsingListPrice,
SUM(UnitPrice * sod.OrderQty)/SUM(sod.OrderQty) as
WeightedAvgPrice,
AVG(UnitPrice) as AvgUnitPrice,
MIN(UnitPrice) as MinUnitPrice,
MAX(UnitPrice) as MaxUnitPrice,
AVG(OrderQty) as AvgOrderQty,
MIN(OrderQty) as MinOrderQty,
MAX(OrderQty) as MaxOrderQty
FROM Sales.SalesOrderDetail sod
GROUP BY sod.SalesOrderID
) details ON soh.SalesOrderID =
details.SalesOrderID
) orders ON orders.SalesPersonID =
sp.BusinessEntityID
LEFT JOIN Person.Person mp ON
cte.ManagerBusinessEntityID = mp.BusinessEntityID
ORDER BY cte.ManagerOrgChart
```

AGENDA

- SELECT FROM “fancy rowsets”
- SELECT “fancy column list” FROM rowsets

LOGICAL ORDER OF OPERATION

1. FROM
2. ON
3. OUTER
4. WHERE
5. GROUP BY
6. HAVING
7. SELECT
8. DISTINCT
9. ORDER BY
10. TOP



SELECT

[distinct]

[top (@number)]

<column list>

FROM <table> as A

JOIN <table> as B

ON A.id = B.ID

LEFT OUTER JOIN <table> as C

ON A.ID = C.ID

WHERE <condition>

GROUP BY <column list>

HAVING <condition>

ORDER BY <column list>

TIME TO GET "FANCY"...

DERIVED TABLES

```
SELECT <column list>
FROM (
    SELECT <column list>
    FROM <table(s)>
    ) as DerivedTable
```

CTE (COMMON TABLE EXPRESSIONS)

```
;
WITH cte AS (
     SELECT <column list>
     FROM <table(s)>
)
SELECT <column list>
FROM cte
```

VIEWS

```
SELECT <column list>
FROM vw_View
```

USER DEFINED FUNCTIONS

```
SELECT <column list>
FROM udf_userdefinedfunction ()

SELECT <column list>,
    udf_userdefinedfunction ()
FROM <rowset>
```

WINDOWED FUNCTIONS

```
SELECT ROW_NUMBER() OVER(),
       SUM() OVER(),
       COUNT() OVER(),
       PERCENTILE_RANK() OVER()
FROM <rowset>
```

LET'S TAKE A LOOK...

DERIVED TABLES: PROS & CONS

Pros:

- Commonly used
- Easy to use

Cons:

- Not “reusable” from one query to another
- Query plan may not directly reflect the derived table

CTE: PROS & CONS

Pros:

- Great for hierarchical data
- Could be used to avoid temporary tables

Cons:

- Not “reusable” from one query to another
- Can create performance problems in certain scenarios
- Not necessarily faster

VIEWS: PROS & CONS

Pros:

- Reusable code
- Great for prepackaging reports for end users
- Indexed Views

Cons:

- May cause execution plans to involve tables that aren't needed by the final query
- Hard to troubleshoot nested views
- Indexed views may not use the indexes or have the underlying data change too often
- May not improve performance

USER DEFINED FUNCTIONS: PROS & CONS

Pros:

- Reusable code
- Can be used to implement Security Policy functionality
- SQL Server is working to improve performance: MTVFs with SQL 2017, scalar functions with SQL 2019

Cons:

- Depends on your SQL Server version
  - Actual Execution Plans may not show underlying tables depending on the type of UDF
  - Inaccurate query plans caused by SQL Server cardinality estimates

WINDOWED FUNCTIONS: PROS & CONS

Pros:

- Flexibility for levels of aggregation on row levels
- Performance

Cons:

- Cannot use in WHERE clause
- Some performance issues may occur

ADDITIONAL "FANCY RABBIT HOLES"

- SELECT XML(), JSON_VALUE()
- FROM OPENROWSET()\OPENQUERY()\OPENXML()
- FROM linkedserver.schema.table
- CROSS APPLY\OUTER APPLY
- UNION (ALL)\INTERSECT\EXCEPT
- etc...

FINAL THOUGHTS

- No Silver Bullet when trying to troubleshoot
- Performance, Performance, Performance
- Test, Test, Test
- Your Mileage May Vary
- Keep It Simple
- Even if your code doesn't change, the way SQL Server processes it may.

ADDITIONAL RESOURCES

- Logical Query Processing:
  - http://www.itprotoday.com/sql-server/logical-query-processing-what-it-and-what-it-means-you
- CTE
  - https://www.erikdarlingdata.com/sql-server/more-cte-myths-persistent-expressions/
- Schemabinding with Views:
  - http://www.sqlhammer.com/sql-server-schemabinding/
- Views and Performance:
  - https://www.scarydba.com/2018/05/14/a-view-will-not-make-your-query-faster/

ADDITIONAL RESOURCES (CONT'D)

- User Defined Functions:
  - https://www.red-gate.com/simple-talk/sql/t-sql-programming/sql-server-user-defined-functions/
- Interleaved Execution with UDFs:
  - https://blogs.msdn.microsoft.com/sqlserverstorageengine/2017/04/19/introducing-interleaved-execution-for-multi-statement-table-valued-functions/
  - https://blogs.msdn.microsoft.com/sqlserverstorageengine/2018/11/07/introducing-scalar-udf-inlining/

ADDITIONAL RESOURCES (CONT'D)

- Window Functions:
  - https://sqlperformance.com/2013/03/t-sql-queries/the-problem-with-window-functions-and-views
  - https://sqlperformance.com/2019/08/sql-performance/t-sql-bugs-pitfalls-and-best-practices-window-function

BEYOND THE SESSION: ADDITIONAL QUESTIONS? LET ME KNOW!

Email: dgmelkin@gmail.com

Twitter: @dgmelkin

Blog: DebtheDBA.wordpress.com

Session and Demo Scripts:

- https://tinyurl.com/y2vh9j6s

Thanks for coming!