***Listing 2.1 – Suggested Index***

/\*

Missing Index Details from chapter 02 - TSQL code - 09-08-2022.sql - DESKTOP-CEBK38L\GRUMPY2019I1.APSales (DESKTOP-CEBK38L\Angelo (63))

The Query Processor estimates that implementing the following index could improve the query cost by 96.4491%.

\*/

/\*

USE [APSales]

GO

CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]

ON [StagingTable].[SalesTransaction] ([ProductNo])

INCLUDE ([CustomerNo],[StoreNo],[CalendarDate])

GO

\*/

***Listing 2.2 – Basic Sales Profile Report***

SELECT YEAR(CalendarDate) AS PurchaseYear,

MONTH(CalendarDate) AS PurchaseMonth,

StoreNo,

ProductNo,

ProductName,

COUNT(\*) AS NumTransactions,

MIN(TransactionQuantity) AS MinQuantity,

MAX(TransactionQuantity) AS MaxQuantity,

AVG(TransactionQuantity) AS AvgQuantity,

SUM(TransactionQuantity) AS SumQuantity

FROM SalesReports.YearlySalesReport

WHERE StoreNo = 'S00001'

AND ProductNo = 'P0000001112'

AND YEAR(CalendarDate) = 2010

GROUP BY YEAR(CalendarDate),

MONTH(CalendarDate),

StoreNo,

ProductNo,

ProductName

ORDER BY YEAR(CalendarDate),

MONTH(CalendarDate),

StoreNo,

ProductNo,

ProductName

GO

***Listing 2.3 – Part 1 the CTE***

WITH ProductPurchaseAnaysis (

PurchaseYear,PurchaseMonth,CalendarDate,StoreNo,CustomerFullName,ProductNo,ItemsPurchased,NumTransactions

)

AS (

SELECT YEAR(CalendarDate) AS PurchaseYear,

MONTH(CalendarDate) AS PurchaseMonth,

CalendarDate,

StoreNo,

CustomerFullName,

ProductNo,

TransactionQuantity AS ItemsPurchased,

COUNT(\*) AS NumTransactions

FROM SalesReports.YearlySalesReport

GROUP BY YEAR(CalendarDate) ,

MONTH(CalendarDate),

CalendarDate,

StoreNo,

CustomerFullName,

ProductNo,

ProductName,

TransactionQuantity

)

***Listing 2.4 – Part 2 – Using Window Functions***

SELECT PurchaseYear,PurchaseMonth,CalendarDate,StoreNo,

CustomerFullName,ProductNo,NumTransactions,

SUM(NumTransactions) OVER (

PARTITION BY PurchaseYear,CustomerFullName

ORDER BY CustomerFullName,PurchaseMonth

) AS SumTransactions,ItemsPurchased,

SUM(ItemsPurchased) OVER (

PARTITION BY PurchaseYear,CustomerFullName

ORDER BY CustomerFullName,PurchaseMonth

) AS TotalItems,

AVG(CONVERT(DECIMAL(10,2),ItemsPurchased)) OVER (

PARTITION BY PurchaseYear,CustomerFullName

ORDER BY CustomerFullName,PurchaseMonth

) AS AvgPurchases,

MIN(ItemsPurchased) OVER (

PARTITION BY PurchaseYear,CustomerFullName

ORDER BY CustomerFullName,PurchaseMonth

) AS MinPurchases,

MAX(ItemsPurchased) OVER (

PARTITION BY PurchaseYear,CustomerFullName

ORDER BY CustomerFullName,PurchaseMonth

) AS MaxPurchases

FROM ProductPurchaseAnalysis

WHERE StoreNo = 'S00001'

AND ProductNo = 'P0000001112'

AND PurchaseYear = 2010

AND PurchaseMonth = 1

AND ItemsPurchased > 0

GROUP BY PurchaseYear,PurchaseMonth,CalendarDate,StoreNo,

CustomerFullName,ProductNo,NumTransactions,ItemsPurchased

ORDER BY CustomerFullName,PurchaseYear,PurchaseMonth,CalendarDate,StoreNo,

ProductNo,ItemsPurchased

GO

***Listing 2.5 – Generating a Rollup Report***

WITH StoreProductSalesAnalysis

(TransYear,TransQuarter,TransMonth,TransDate,StoreNo,ProductNo,MonthlySales)

AS

(

SELECT

YEAR(CalendarDate) AS TransYear,

DATEPART(qq,CalendarDate) AS TransQuarter,

MONTH(CalendarDate) AS TransMonth,

CalendarDate AS TransDate,

StoreNo,

ProductNo,

SUM(TotalSalesAmount) AS MonthlySales

FROM FactTable.YearlySalesReport

GROUP BY

CalendarDate,

StoreNo,

ProductNo

)

SELECT TransYear,

TransQuarter,

TransMonth,

StoreNo,

ProductNo,

MonthlySales,

SUM(MonthlySales) AS SumMonthlySales,

GROUPING(MonthlySales) AS RollupFlag

FROM StoreProductSalesAnalysis

WHERE TransYear = 2011

AND ProductNo = 'P0000001103'

AND StoreNo = 'S00001'

GROUP BY TransYear,

TransQuarter,

TransMonth,

StoreNo,

ProductNo,

MonthlySales WITH ROLLUP

ORDER BY TransYear,

TransQuarter,

TransMonth,

StoreNo,

ProductNo,

(

CASE

WHEN MonthlySales IS NULL THEN 0

END

) DESC,

GROUPING(MonthlySales) DESC

GO

***Listing 2.6 – suggested new index***

/\*

Missing Index Details from chapter 02 - TSQL code - 09-13-2022.sql - DESKTOP-CEBK38L\GRUMPY2019I1.APSales (DESKTOP-CEBK38L\Angelo (55))

The Query Processor estimates that implementing the following index could improve the query cost by 98.1615%.

\*/

/\*

USE [APSales]

GO

CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]

ON [SalesReports].[YearlySalesReport] ([ProductNo],[StoreNo])

INCLUDE ([CalendarDate],[TotalSalesAmount])

GO

\*/

DROP INDEX IF EXISTS [ieProductNoStoreNoDateTotalSalesAmt]

ON [SalesReports].[YearlySalesReport]

GO

CREATE NONCLUSTERED INDEX [ieProductNoStoreNoDateTotalSalesAmt]

ON [SalesReports].[YearlySalesReport] ([ProductNo],[StoreNo])

INCLUDE ([CalendarDate],[TotalSalesAmount])

GO

***Listing 2.7 – Product Report using STRING\_AGG()***

WITH CustomerPurchaseAnalysis(PurchaseYear,PurchaseMonth,CustomerNo,ProductNo,PurchaseCount)

AS

(

SELECT DISTINCT

YEAR(CalendarDate) AS PurchaseYear,

MONTH(CalendarDate) AS PurchaseMonth,

CustomerNo,

ProductNo,

COUNT(\*) AS PurchaseCount

FROM StagingTable.SalesTransaction

GROUP BY YEAR(CalendarDate),

MONTH(CalendarDate),

CustomerNo,

ProductNo

)

SELECT

PurchaseYear,

PurchaseMonth,

CustomerNo,

STRING\_AGG(ProductNo,',') AS ItemsPurchased,

COUNT(PurchaseCount) AS PurchaseCount

FROM CustomerPurchaseAnalysis

WHERE CustomerNo = 'C00000008'

GROUP BY

PurchaseYear,

PurchaseMonth,

CustomerNo

ORDER BY CustomerNo,

PurchaseYear,

PurchaseMonth

GO

***Listing 2.8 – Standard Deviation Sales Analysis***

WITH CustomerPurchaseAnalysis

(PurchaseYear,PurchaseMonth,StoreNo,ProductNo,CustomerNo,TotalSalesAmount)

AS

(

SELECT

YEAR(CalendarDate) AS PurchaseYear,

MONTH(CalendarDate) AS PurchaseMonth,

StoreNo,

ProductNo,

CustomerNo,

SUM(TransactionQuantity \* UnitRetailPrice) AS TotalSalesAmount

FROM StagingTable.SalesTransaction

GROUP BY YEAR(CalendarDate),MONTH(CalendarDate),ProductNo,CustomerNo,StoreNo

)

SELECT

cpa.PurchaseYear,

cpa.PurchaseMonth,

cpa.StoreNo,

cpa.ProductNo,

c.CustomerNo,

CONVERT(DECIMAL(10,2),cpa.TotalSalesAmount) AS TotalSalesAmount,

AVG(CONVERT(DECIMAL(10,2),cpa.TotalSalesAmount)) OVER(

--PARTITION BY cpa.PurchaseYear,c.CustomerNo

ORDER BY cpa.PurchaseYear,c.CustomerNo

) AS AvgPurchaseCount,

STDEV(CONVERT(DECIMAL(10,2),cpa.TotalSalesAmount)) OVER(

ORDER BY cpa.PurchaseMonth

) AS StdevTotalSales,

STDEVP(CONVERT(DECIMAL(10,2),cpa.TotalSalesAmount)) OVER(

ORDER BY cpa.PurchaseMonth

) AS StdevpTotalSales,

STDEV(CONVERT(DECIMAL(10,2),cpa.TotalSalesAmount)) OVER(

) AS StdevTotalSales,

STDEVP(CONVERT(DECIMAL(10,2),cpa.TotalSalesAmount)) OVER(

) AS StdevpYearTotalSales

FROM CustomerPurchaseAnalysis cpa

JOIN DimTable.Customer c

ON cpa.CustomerNo = c.CustomerNo

WHERE cpa.CustomerNo = 'C00000008'

AND PurchaseYear = 2011

AND ProductNo = 'P00000038114';

GO

***Listing 2.9 – Estimated Query Plan - Suggested Index***

/\*

Missing Index Details from chapter 02 - TSQL code - 09-13-2022.sql - DESKTOP-CEBK38L\GRUMPY2019I1.APSales (DESKTOP-CEBK38L\Angelo (65))

The Query Processor estimates that implementing the following index could improve the query cost by 80.174%.

\*/

/\*

USE [APSales]

GO

CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]

ON [StagingTable].[SalesTransaction] ([CustomerNo],[ProductNo])

INCLUDE ([StoreNo],[CalendarDate],[TransactionQuantity],[UnitRetailPrice])

GO

\*/

/\* Copy code from above and paste and supply name \*/

DROP INDEX IF EXISTS [CustNoProdNoStoreNoDateQtyPrice]

ON [StagingTable].[SalesTransaction]

GO

CREATE NONCLUSTERED INDEX [CustNoProdNoStoreNoDateQtyPrice]

ON [StagingTable].[SalesTransaction] ([CustomerNo],[ProductNo])

INCLUDE ([StoreNo],[CalendarDate],[TransactionQuantity],[UnitRetailPrice])

GO

***Listing 2.10 – Calculating Sales Variance***

WITH CustomerPurchaseAnalysis

(PurchaseYear,PurchaseMonth,StoreNo,ProductNo,CustomerNo,TotalSalesAmount)

AS

(

SELECT

YEAR(CalendarDate) AS PurchaseYear,

MONTH(CalendarDate) AS PurchaseMonth,

StoreNo,

ProductNo,

CustomerNo,

SUM(TransactionQuantity \* UnitRetailPrice) AS TotalSalesAmount

FROM StagingTable.SalesTransaction

GROUP BY YEAR(CalendarDate),MONTH(CalendarDate),

ProductNo,CustomerNo,StoreNo

)

SELECT

cpa.PurchaseYear,

cpa.PurchaseMonth,

cpa.StoreNo,

cpa.ProductNo,

c.CustomerNo,

c.CustomerFullName,

CONVERT(DECIMAL(10,2),cpa.TotalSalesAmount) AS TotalSalesAmount,

AVG(CONVERT(DECIMAL(10,2),cpa.TotalSalesAmount)) OVER(

ORDER BY cpa.PurchaseMonth) AS AvgPurchaseCount,

VAR(CONVERT(DECIMAL(10,2),cpa.TotalSalesAmount)) OVER(

ORDER BY cpa.PurchaseMonth

) AS VarTotalSales,

VARP(CONVERT(DECIMAL(10,2),cpa.TotalSalesAmount)) OVER(

ORDER BY cpa.PurchaseMonth

) AS VarpTotalSales,

VAR(CONVERT(DECIMAL(10,2),cpa.TotalSalesAmount)) OVER(

) AS VarTotalSales,

VARP(CONVERT(DECIMAL(10,2),cpa.TotalSalesAmount)) OVER(

) AS VarpYearTotalSales

FROM CustomerPurchaseAnalysis cpa

JOIN DimTable.Customer c

ON cpa.CustomerNo = c.CustomerNo

WHERE cpa.CustomerNo = 'C00000008'

AND PurchaseYear = 2011

AND ProductNo = 'P00000038114';

GO

***Listing 2.11 – Average by Year,Month and Customer***

WITH CustomerPurchaseAnalysis

(PurchaseYear,PurchaseMonth,CustomerNo,TotalSalesAmount)

AS

(

SELECT

YEAR(CalendarDate) AS PurchaseYear,

MONTH(CalendarDate) AS PurchaseMonth,

CustomerNo,

SUM(TransactionQuantity \* UnitRetailPrice) AS TotalSalesAmount

FROM StagingTable.SalesTransaction

GROUP BY YEAR(CalendarDate),MONTH(CalendarDate),CustomerNo

)

SELECT

cpa.PurchaseYear,

cpa.PurchaseMonth,

c.CustomerNo,

c.CustomerFullName,

cpa.TotalSalesAmount,

AVG(cpa.TotalSalesAmount) OVER SalesWindow AS AvgTotalSales

FROM CustomerPurchaseAnalysis cpa

JOIN DimTable.Customer c

ON cpa.CustomerNo = c.CustomerNo

WHERE cpa.CustomerNo = 'C00000008'

WINDOW SalesWindow AS (

PARTITION BY cpa.PurchaseYear

ORDER BY cpa.PurchaseYear ASC,cpa.PurchaseMonth ASC

)

GO

***Listing 2.12 – Suggested Index***

/\*

Missing Index Details from SQLQuery2.sql - DESKTOP-CEBK38L.APSales (DESKTOP-CEBK38L\Angelo (66))

The Query Processor estimates that implementing the following index could improve the query cost by 99.0667%.

\*/

/\*

USE [APSales]

GO

CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]

ON [StagingTable].[SalesTransaction] ([CustomerNo])

INCLUDE ([CalendarDate],[TransactionQuantity],[UnitRetailPrice])

GO

\*/

DROP INDEX IF EXISTS [CustomerNoieDateQuantityRetailPrice]

ON [StagingTable].[SalesTransaction]

GO

CREATE NONCLUSTERED INDEX [CustomerNoieDateQuantityRetailPrice]

ON [StagingTable].[SalesTransaction] ([CustomerNo])

INCLUDE ([CalendarDate],[TransactionQuantity],[UnitRetailPrice])

GO

***Listing 2.13 – Defining Multiple Windows***

SELECT

cpa.PurchaseYear,

cpa.PurchaseMonth,

c.CustomerNo,

c.CustomerFullName,

cpa.TotalSalesAmount,

AVG(cpa.TotalSalesAmount) OVER AvgSalesWindow AS AvgTotalSales,

STDEV(cpa.TotalSalesAmount) OVER StdevSalesWindow AS StdevTotalSales,

SUM(cpa.TotalSalesAmount) OVER SumSalesWindow AS SumTotalSales

FROM CustomerPurchaseAnalysis cpa

JOIN DimTable.Customer c

ON cpa.CustomerNo = c.CustomerNo

WHERE cpa.CustomerNo = 'C00000008'

WINDOW

StdevSalesWindow AS (AvgSalesWindow),

AvgSalesWindow AS (

PARTITION BY cpa.PurchaseYear

ORDER BY cpa.PurchaseYear ASC,cpa.PurchaseMonth ASC

),

SumSalesWindow AS (

);

GO