

SQL Server Query Optimization Report

Server Name: QATAR
Report Date: June 11th, 2025

Objective:

The goal of this report is to provide recommendations for optimizing SQL queries and improving overall database performance. These recommendations are based on all the collected data. While we highly believe they will enhance performance, it is essential to monitor the performance from the application side and compare the before-and-after results. Each application-database environment may have unique requirements, and further tuning may be necessary to achieve the best improvement results.

1. Expensive Queries

	Current Query	Proposed Query or Recommended Fix	Explanation
1	<pre>SELECT [_].[ID] [ID],[_].[ORDER_NUM] [Order Number],[_].[ORDER_LINE_NUM] [Order Line Number],[_].[INV_DESCRIPTION] [Invoice Description],[_].[RevenueID] [RevenueID],[_].[CustomerID] [CustomerID],[_].[GeographyID] [GeographyID],[_].[ProductID] [ProductID],[_].[LocationID] [LocationID],[_].[DemandClassID] [DemandClassID],[_].[DateID] [DateID],[_].[DateKey] [DateKey],[_].[HourID] [HourID],[_].[SELL_DOLLARS] [\$SELL_DOLLARS],[_].[ORIGINAL_LIST_DOLLARS] [\$ORIGINAL_LIST_DOLLARS],[_].[LIST_DOLLARS] [\$SALES],[_].[COGSSB] [\$COGSSB],[_].[COGSPV] [\$COGSPV],[_].[QTY] [QTY],[_].[FLOW_STATUS_CODE] [Order Status],[_].[IsOpen] [Is Open],[_].[IsInvoiced] [Is Invoiced],[_].[COOP] [\$COOP],[_].[DIF RETURNS] [\$DIF RETURNS],[_].[Freight Allowance] [\$Freight Allowance],[_].[Cash Discounts] [\$Cash Discounts],[_].[Markdown] [\$Markdown],[].[Other] [\$Other] FROM</pre>	Index on [DateKey] CREATE NONCLUSTERED INDEX IX_FactPBISalesOrders_DateKey ON dbo.FactPBISalesOrders(DateKey) INCLUDE ([QTY], [SELL_DOLLARS], [LIST_DOLLARS]); -- include commonly selected columns Final Optimized Query SELECT [_].[ID] AS [ID], [_].[ORDER_NUM] AS [Order Number], [_].[ORDER_LINE_NUM] AS [Order Line Number], [_].[INV_DESCRIPTION] AS [Invoice Description], [_].[RevenueID], [_].[CustomerID], [_].[GeographyID], [_].[ProductID], [_].[LocationID], [_].[DemandClassID], [_].[DateID], [_].[DateKey], [_].[HourID], [_].[SELL_DOLLARS] AS [\$SELL_DOLLARS], [_].[ORIGINAL_LIST_DOLLARS] AS [\$ORIGINAL_LIST_DOLLARS], [_].[LIST_DOLLARS] AS [\$SALES], [_].[COGSSB], [_].[COGSPV], [_].[QTY], [_].[FLOW_STATUS_CODE] AS [Order Status], [_].[IsOpen], [_].[IsInvoiced], [_].[COOP], [_].[DIF RETURNS], [_].[Freight Allowance],	Avoid CONVERT on Indexed Columns in WHERE Clause and Ensure @1 and @2 are passed as datetime2 already, and remove the CONVERT. WHERE [_].[DateKey] >= @1 AND [_].[DateKey] < @2 Consider selecting necessary columns only

	<pre>[dbo].[FactPBISalesOrders] [_] WHERE [_].[DateKey]>=CONVERT([datetime2],@1) AND [_].[DateKey]<CONVERT([datetime2],@2)</pre>	<pre>[_].[Cash Discounts], [_].[Markdown], [_].[Other] FROM [dbo].[FactPBISalesOrders] AS [_] WHERE [_].[DateKey] >= @1 AND [_].[DateKey] < @2;</pre>	
2	<pre>SELECT identity(int,1,1) as ID, * INTO #SalesBudget FROM Fact.SalesForecastY</pre>	Clustered Index on DateKey <pre>CREATE CLUSTERED INDEX IX_SalesBudget_DateKey ON #SalesBudget(DateKey);</pre> <pre>SELECT IDENTITY(INT, 1, 1) AS ID, [SalesPersonID], [ProductID], [ForecastAmount], [DateKey] INTO #SalesBudget FROM Fact.SalesForecastY WITH (NOLOCK) WHERE DateKey >= @StartDate AND DateKey < @EndDate;</pre>	Avoid <code>SELECT *</code> – Be Explicit. Filter Rows (if applicable). Once #SalesBudget is populated, you can improve further processing (filtering, joins, groupings, etc.) with a clustered index on key columns

	Current Query	Proposed Query or Recommended Fix	Explanation
3	SELECT ForecastSource, DateID, ProductID, DemandClassID, SUM(Quantity) AS Quantity INTO dbo.FactSalesForecastQuantites FROM Operations.Fact.SalesForecastQuantities --WHERE Quantity > 0 GROUP BY ForecastSource, DateID, ProductID, DemandClassID	CREATE CLUSTERED INDEX IX_FactSalesForecastQuantities_Date_Product ON dbo.FactSalesForecastQuantities(DateID, ProductID);	Fix table name typo: Your target table name had a typo (FactSalesForecastQuantites → FactSalesForecastQuantities). Table has 28,259 rows, please consider filtering or limiting rows (if applicable). Consider adding indexes to improve query performance using the most selective or most useful columns for filtering.
4	SELECT * INTO dbo.FactStandardPending FROM Operations.Fact.StandardPending WHERE (SELECT COGSDate FROM dbo.ForecastPeriod) BETWEEN StartDate AND ISNULL(EndDate, '9999-12-31')	DECLARE @COGSDate DATE; SELECT @COGSDate = COGSDate FROM dbo.ForecastPeriod; SELECT * INTO dbo.FactStandardPending FROM Operations.Fact.StandardPending WHERE @COGSDate BETWEEN StartDate AND ISNULL(EndDate, '9999-12-31'); SELECT sp.* INTO dbo.FactStandardPending FROM Operations.Fact.StandardPending sp WHERE EXISTS (SELECT 1 FROM dbo.ForecastPeriod fp WHERE fp.COGSDate BETWEEN sp.StartDate AND ISNULL(sp.EndDate, '9999-12-31'));	Replaced correlated subquery with EXISTS If dbo.ForecastPeriod is guaranteed to always have only one row , you can consider the first proposed fix, which avoids a join or subquery entirely and is the most efficient option if COGSDate is scalar.

	Current Query	Proposed Query or Recommended Fix	Explanation
5	<pre>SELECT * INTO dbo.FactPBISales FROM Operations.dbo.FactPBISales s WHERE s.DateKey >= DATEADD(MONTH,-36,(SELECT CustomerSplitDate FROM dbo.ForecastPeriod)) AND s.DateKey < (SELECT CustomerSplitDate FROM dbo.ForecastPeriod)</pre>	<pre>CREATE NONCLUSTERED INDEX IX_FactPBISales_DateKey ON dbo.FactPBISales(DateKey);</pre> <pre>-- Declare the scalar value once for reuse DECLARE @CustomerSplitDate DATE; SELECT @CustomerSplitDate = CustomerSplitDate FROM dbo.ForecastPeriod; -- Materialize into a temp table using range filter SELECT s.* INTO dbo.FactPBISales FROM Operations.dbo.FactPBISales s WHERE s.DateKey >= DATEADD(MONTH, -36, @CustomerSplitDate) AND s.DateKey < @CustomerSplitDate;</pre>	Ensure there's an index on FactPBISales(DateKey) — even better if it includes other needed columns. Consider selecting only the columns needed. You're calling (SELECT CustomerSplitDate FROM dbo.ForecastPeriod) twice — this is inefficient.
6	<pre>SELECT * INTO dbo.DimOrderCurveProductGroup FROM Dim.OrderCurveProductGroup</pre>	<pre>SELECT column1, column2, column3 INTO dbo.DimOrderCurveProductGroup FROM Dim.OrderCurveProductGroup;</pre>	Specify columns name instead of SELECT * Ensure indexes exist on the source table (Dim.OrderCurveProductGroup) to speed up data retrieval.

7	<pre>WITH PurgedOrders AS (SELECT ORD_LINE_ID FROM Oracle.Orders WHERE CAST(ORDER_DATE AS DATE) >= '2021- 01-01' GROUP BY ORD_LINE_ID HAVING MIN(CAST(CurrentRecord AS SMALLINT)) = MAX(CAST(CurrentRecord AS SMALLINT))) CanceledOrders AS (SELECT ORDER_NUM, ORDER_LINE_NUM FROM Oracle.Orders WHERE CAST(ORDER_DATE AS DATE) >= '2021-01-01' AND FLOW_STATUS_CODE = 'CANCELLED' GROUP BY ORDER_NUM, ORDER_LINE_NUM) SELECT ISNULL(rd.Cancel_Reason, 'No reason provided') AS Cancel_Reason , ISNULL(cr.[Cancellation Category], 'No reason provided') AS [Cancellation Category] , rd.USERNAME AS [Canceled By] , d.[LINE_ID] , d.[SELLING_PRICE] , d.[LIST_PRICE] , d.[PRICING_QUANTITY] , d.[HIST_TYPE_CODE] , d.[HIST_CREATION_DATE] , o.[ORD_LINE_ID] , o.[ORD_HEADER_ID] , o.[CUSTOMER_NAME] , o.[CUSTOMER_NUM] , o.[SALES_CHANNEL_CODE] , o.[DEMAND_CLASS] , CAST(o.[ORDER_DATE] AS DATE) AS [ORDER_DATE] , o.[ORDER_NUM] , o.[PART] , o.[ORDERED_ITEM] , o.[PART_DESC] , o.[FLOW_STATUS_CODE] , o.[QTY] , o.[SELL_DOLLARS] , o.[LIST_DOLLARS] , o.[DATE_REQUESTED] , o.[SCH_SHIP_DATE] , o.[CANCEL_DATE] , o.[PLANT] , o.[CREATE_DATE] , o.[ORD_LINE_CREATE_DATE] , o.[ORD_LINE_LST_UPDATE_DATE] , o.[SHIP_TO_ADDRESS1] , o.[SHIP_TO_ADDRESS2] , o.[SHIP_TO_ADDRESS3] , o.[SHIP_TO_ADDRESS4] , o.[SHIP_TO_CITY] , o.[SHIP_TO_STATE] , o.[SHIP_TO_POSTAL_CODE] , o.[SHIP_TO_COUNTRY] , o.[SHIP_TO_PROVINCE] , o.[Fingerprint] , o.[StartDate] , o.[EndDate] , o.[CurrentRecord] , o.[ACTUAL_SHIPMENT_DATE] , o.[ORDER_LINE_NUM] , o.[SHIPPING_METHOD_CODE] INTO Output.CancelHistory FROM Oracle.OrderLineHistory d INNER JOIN Oracle.Orders o ON o.ORD_LINE_ID = d.LINE_ID AND d.HIST_CREATION_DATE BETWEEN DATEADD(MINUTE,- 15,o.StartDate) AND StartDate --use current record if data was purged (no sales or qty available) INNER JOIN PurgedOrders po ON po.ORD_LINE_ID = o.ORD_LINE_ID INNER JOIN CanceledOrders co ON o.ORDER_NUM = co.ORDER_NUM AND o.ORDER_LINE_NUM = co.ORDER_LINE_NUM LEFT JOIN Oracle.CancelledOrdersReasonDetail rd ON o.ORDER_NUM = rd.ORDNUM AND o.ORD_LINE_ID = rd.lineID AND CAST(CANDATE AS DATE) = CAST(o.StartDate AS DATE) LEFT JOIN xref.CancelReason cr ON rd.[CANCEL REASON] = cr.[Cancellation Description] WHERE o.CurrentRecord = 1 --AND d.LINE ID = @lineID UNION SELECT</pre>	<pre>CREATE NONCLUSTERED INDEX IX_Orders_ORD_LINE_ID_OrderDate_CurrentRecord ON Oracle.Orders (ORD_LINE_ID, ORDER_DATE, CurrentRecord); CREATE NONCLUSTERED INDEX IX_Orders_OrderNum_LineNum_Status ON Oracle.Orders (ORDER_NUM, ORDER_LINE_NUM, FLOW_STATUS_CODE); CREATE NONCLUSTERED INDEX IX_OrderLineHistory_LINE_ID_HistCreationDate ON Oracle.OrderLineHistory (LINE_ID, HIST_CREATION_DATE); CREATE NONCLUSTERED INDEX IX_CancelledOrdersReasonDetail ON Oracle.CancelledOrdersReasonDetail (ORDNUM, lineID, CANDATE); CREATE NONCLUSTERED INDEX IX CancelReason ON xref.CancelReason ([Cancellation Description]);</pre>	Consider the indexes for performance improvement. Consider the changes below to the query: - Removed redundant columns repeated in the SELECT statements by using a CTE for the common columns. - Added table aliases consistently. - Replaced repeated CAST with variables where appropriate. - Used EXISTS instead of JOIN for PurgedOrders and CanceledOrders where applicable for optimization.
---	---	---	--

	<pre>ISNULL(rd.Cancel_Reason, 'No reason provided') AS Cancel_Reason , ISNULL(cr.[Cancellation Category], 'No reason provided') AS [Cancellation Category] rd.USERNAME AS [Canceled By] , d.[LINE_ID] d.[SELLING_PRICE] , d.[LIST_PRICE] d.[PRICING_QUANTITY] , d.[HIST_TYPE_CODE] d.[HIST_CREATION_DATE] , o.[ORD_LINE_ID] o.[ORD_HEADER_ID] , o.[CUSTOMER_NAME] o.[CUSTOMER_NUM] , o.[SALES_CHANNEL_CODE] o.[DEMAND_CLASS] , CAST(o.[ORDER_DATE] AS DATE) AS [ORDER_DATE] , o.[ORDER_NUM] , o.[PART] o.[ORDERED_ITEM] , o.[PART_DESC] o.[FLOW_STATUS_CODE] , o.[QTY] , o.[SELL_DOLLARS] , o.[LIST_DOLLARS] , o.[DATE_REQUESTED] o.[SCH_SHIP_DATE] , o.[CANCEL_DATE] , o.[PLANT] , o.[CREATE_DATE] , o.[ORD_LINE_CREATE_DATE] o.[ORD_LINE_LST_UPDATE_DATE] , o.[SHIP_TO_ADDRESS1] , o.[SHIP_TO_ADDRESS2] , o.[SHIP_TO_ADDRESS3] o.[SHIP_TO_ADDRESS4] , o.[SHIP_TO_CITY] o.[SHIP_TO_STATE] , o.[SHIP_TO_POSTAL_CODE] o.[SHIP_TO_COUNTRY] , o.[SHIP_TO_PROVINCE] o.[Fingerprint] , o.[StartDate] , o.[EndDate] o.[CurrentRecord] , o.[ACTUAL_SHIPMENT_DATE] o.[ORDER_LINE_NUM] , o.[SHIPPING_METHOD_CODE] FROM Oracle.OrderLineHistory d INNER JOIN Oracle.Orders o ON o.ORD_LINE_ID = d.LINE_ID AND d.HIST_CREATION_DATE BETWEEN DATEADD(MINUTE,-15,o.EndDate) AND o.EndDate --use prior record (non cancelled) if data has history LEFT JOIN PurgedOrders po ON po.ORD_LINE_ID = o.ORD_LINE_ID INNER JOIN CanceledOrders co ON o.ORDER_NUM = co.ORDER_NUM AND o.ORDER_LINE_NUM = co.ORDER_LINE_NUM LEFT JOIN Oracle.CancelledOrdersReasonDetail rd ON o.ORDER_NUM = rd.ORDNUM AND o.ORD_LINE_ID = rd.lineID AND CAST(CANDATE AS DATE) = CAST(o.EndDate AS DATE) LEFT JOIN xref.CancelReason cr ON rd.[CANCEL_REASON] = cr.[Cancellation Description] WHERE po.ORD_LINE_ID IS NULL AND o.FLOW_STATUS_CODE <> 'CANCELLED' --AND d.LINE_ID = @LineID ORDER BY StartDate</pre>	
--	---	--

8	<pre>SELECTTOP (1000001) [t0].[DateID] AS [c1],[t0].[HourID] AS [c2],[t0].[PlantID] AS [c3],[t0].[ProductAID] AS [c4],[t0].[ProductBID] AS [c5],[t0].[ShiftID] AS [c6],[t0].[MachineID] AS [c7],[t0].[ReasonID] AS [c8],[t0].[CycleCount] AS [c9],[t0].[MissedCycle] AS [c10],[t0].[EmptyCycle] AS [c11],[t0].[CycleTime] AS [c12],[t0].[IndexTime] AS [c13],[t0].[MissedTime] AS [c14],[t0].[OperatorID] AS [c15],[t0].[MadeTime] AS [c16],[t0].[TotalTime] AS [c17],[t0].[OvenTime] AS [c18],[t0].[CurrentShiftID] AS [c19],[t0].[ID] AS [c20],COUNT BIG([t0].[ID])AS [a0]FROM(((select [ID],[DateID],[HourID],[PlantID],[ProductAID],[ProductBID],[ShiftID],[MachineID],[OperatorID],[ReasonID],[CycleCount],[MissedCycle],[EmptyCycle],[CycleTime],[IndexTime],[MissedTime],[MadeTime],[OvenTime],[TotalTime],[CurrentShiftID]from [dbo].[FactMachineIndexLive] as [\$ Table]) AS [t0]INNER JOIN(select [_].[LocationID] as [LocationID],[_].[LocationKey] as [Location Key],[_].[LocationName] as [Location Name],[_].[LocationDesc] as [Location Desc],[_].[Sort] as [Sort],[_].[LocationType] as [Location Type],[_].[LocationCountry] as [LocationCountry],[_].[ShiftScalar] as [ShiftScalar],[_].[GLLocationID] as [GLLocationID],[_].[WarehouseCapacity] as [WarehouseCapacity],[_].[PhysicalLocation] as [Physical Location],[_].[PlanID] as [PlanID],[_].[COGSLocationID] as [COGSLocationID]from [dbo].[DimLocation] as [_]) AS [t1] on([t0].[PlantID] = [t1].[LocationID]))INNER JOIN(select [_].[MachineID] as [MachineID],[_].[LocationKey] as [Location Key],[_].[MachineKey] as [Machine Key],[_].[MachineName] as [Machine Name],[_].[MachineDesc] as [Machine Desc],[_].[MachineSort] as [MachineSort],[_].[MachineModel] as [Machine Model],[_].[MachineNumber] as [Machine Number],[_].[MachineCell] as [Machine Cell],[_].[CellPosition] as [Cell Position],[_].[RoundsPerShift] as [RoundsPerShift],[_].[CapacityRoundsPerShift] as [CapacityRoundsPerShift]from [dbo].[DimMachine] as [_]) AS [t2] on([t0].[MachineID] = [t2] .[MachineID]))INNER JOIN(select [_].[ShiftControlID] as [ShiftControlID],[_].[LocationID] as [LocationID],[_].[CurrentShiftID] as [CurrentShiftID],[_].[ShiftName] as [Shift Name],[_].[ShiftHourBlock1] as [ShiftHourBlock1],[_].[ShiftHourBlock2] as [ShiftHourBlock2],[_].[ShiftHourBlock3]</pre>	<pre>-- Fact table CREATE NONCLUSTERED INDEX IX_FactMachineIndexLive_JoinColumns ON dbo.FactMachineIndexLive (PlantID, MachineID, CurrentShiftID, ID); -- DimLocation CREATE NONCLUSTERED INDEX IX_DimLocation_LocationID_Key ON dbo.DimLocation (LocationID, [Location Key]); -- DimMachine CREATE NONCLUSTERED INDEX IX DimMachine MachineID Model Number ON dbo.DimMachine (MachineID, [Machine Model], [Machine Number]); -- DimShiftControl CREATE NONCLUSTERED INDEX IX_DimShiftControl_ShiftID ON Dim.ShiftControl (CurrentShiftID, ShiftControlID); CREATE NONCLUSTERED INDEX IX_FactMachineIndexLive_Recent ON dbo.FactMachineIndexLive (DateID) WHERE DateID >= '2023-01-01';</pre>	Consider the indexes. If the dataset is huge, consider partitioning FactMachineIndexLive by DateID if most queries use time filtering by creating the index: FactMachineIndexLive
---	--	--	--

	<pre>as [ShiftHourBlock3],[_].[ShiftHourBlock4] as [ShiftHourBlock4],[_].[ShiftHourBlock5] as [ShiftHourBlock5],[_].[ShiftHourBlock6] as [ShiftHourBlock6] from [Dim].[ShiftControl] as [_]) AS [t10] on([t0].[CurrentShiftID] = [t10].[CurrentShiftID]))WHERE(((([t1].[Location Key] = N'111')AND([t2].[Machine Model] = '190')) AND([t2].[Machine Number] = 'L01')AND([t10].[ShiftControlID] = 1)))GROUP BY [t0].[DateID],[t0].[HourID],[t0].[PlantID],[t0].[ProductAID],[t0].[ProductBID],[t0].[ShiftID],[t0].[MachineID],[t0].[ReasonID],[t0].[CycleCount],[t0].[MissedCycle],[t0].[EmptyCycle],[t0].[CycleTime],[t0].[IndexTime],[t0].[MissedTime],[t0].[OperatorID],[t0].[MadeTime],[t0].[TotalTime],[t0].[OvenTime],[t0].[CurrentShiftID],[t0].[ID]OPTION(LABEL = N'{ "DatasetId":"6e8664ad-20f8-4605-98a7-a8aae3d9a738", "Sources":[{"ReportId":"d636d55f-6128-4e5f-8aa7- f2c771f9faba"},"VisualId":"3b92ed862955d7416ac2"}]}')</pre>		
9	<pre>SELECT TOP (3) WITH TIES*FROM (SELECT [t8].[Repair Reason Name] AS [c97],SUM([t7].[Qty]) AS [a0]FROM ((((((select [PlantID], [ShiftID], [MachineID], [RepairID], [RepairReasonID], [DateID], [HourID], [CurrentShiftID], [Qty], [Lbs]from [dbo].[FactScrapLiveScreens] as [\$Table]) AS [t7] INNER JOIN (select [_].[LocationID] as [LocationID], [_].[LocationKey] as [Location Key], [_].[LocationName] as [Location Name], [_].[LocationDesc] as [Location Desc], [_].[Sort] as [Sort], [_].[LocationType] as [Location Type], [].[LocationCountry] as [LocationCountry], [_].[ShiftScalar] as [ShiftScalar], [_].[GLLocationID] as [GLLocationID], [_].[WarehouseCapacity] as [WarehouseCapacity], [_].[PhysicalLocation] as [Physical Location], [_].[PlanID] as [PlanID], [].[COGSLocationID] as [COGSLocationID] from [dbo].[DimLocation] as [_]) AS [t1] on ([t7].[PlantID] = [t1].[LocationID])) LEFT OUTER JOIN (select [_].[MachineID] as [MachineID], [_].[LocationKey] as [Location Key], [_].[MachineKey] as [Machine Key], [_] .[MachineName] as [Machine Name], [_].[MachineDesc] as [Machine Desc], [_].[MachineSort] as [MachineSort], [_] .[MachineModel] as [Machine Model], [_].[MachineNumber] as [Machine Number], [_].[MachineCell] as [Machine Cell], [_].[CellPosition] as [Cell Position], [_].[RoundsPerShift] as [RoundsPerShift], [_].[CapacityRoundsPerShift] as [CapacityRoundsPerShift]from [dbo].[DimMachine] as [_]) AS [t2] on ([t7].[MachineID] = [</pre>	Optimized Query (Needs to be tested) <pre>SELECT TOP (3) WITH TIES t8.RepairReasonName AS [c97], SUM(t7.Qty) AS [a0] FROM dbo.FactScrapLiveScreens t7 INNER JOIN dbo.DimLocation t1 ON t7.PlantID = t1.LocationID LEFT JOIN dbo.DimMachine t2 ON t7.MachineID = t2.MachineID LEFT JOIN dbo.DimRepairReason t8 ON t7.RepairReasonID = t8.RepairReasonID INNER JOIN dbo.DimRepair t9 ON t7.RepairID = t9.RepairID INNER JOIN Dim.ShiftControl t10 ON t7.CurrentShiftID = t10.CurrentShiftID WHERE t1.LocationKey = N'111' AND t2.MachineModel = '190' AND t2.MachineCell = 2 AND t2.CellPosition = 1 AND t9.RepairKey = 'REPAIR' AND t10.ShiftControlID = 1 GROUP BY t8.RepairReasonName HAVING SUM(t7.Qty) IS NOT NULL ORDER BY [a0] DESC OPTION (LABEL = N'{ "DatasetId":"6e8664ad-20f8-4605-98a7- a8aae3d9a738", "Sources":[{"ReportId":"d636d55f-6128-4e5f- 8aa7-f2c771f9faba", "VisualId":"89cable8a93ec32f33e9"}]}');</pre>	• Remove unnecessary subquery nesting — inline everything cleanly. • Apply filters early in JOIN conditions to reduce row counts before aggregation. • Avoid using <code>SELECT *</code> — select only needed columns. • Use efficient aliases and flatten table names. • Indexes — see suggestions at the end.

<pre>t2].[MachineID])) LEFT OUTER JOIN (select [_].[RepairReasonID] as [Repair Reason ID], [_].[OrgCode] as [Org Code], [_].[RepairReasonKey] as [Repair Reason Key], [_].[RepairReasonName] as [Repair Reason Name], [_].[RepairReasonDesc] as [Repair Reason Desc], [_].[RepairReasonSort] as [Repair Reason Sort] from [dbo].[DimRepairReason] as [_]) AS [t8] on ([t7].[RepairReasonID] = [t8].[Repair Reason ID])) INNER JOIN (select [_].[RepairID] as [RepairID], [_].[RepairKey] as [Repair Key], [].[RepairName] as [Repair Name], [].[RepairDesc] as [Repair Desc], [_].[RepairSort] as [RepairSort], [_].[Scrap] as [Scrap], [_].[Repair] as [Repair] from [dbo].[DimRepair] as [_]) AS [t9] on ([t7].[RepairID] = [t9].[RepairID])) INNER JOIN (select [_].[ShiftControlID] as [ShiftControlID], [_].[LocationID] as [LocationID], [_].[CurrentShiftID] as [CurrentShiftID], [_].[ShiftName] as [Shift Name], [_].[ShiftHourBlock1] as [ShiftHourBlock1], [_].[ShiftHourBlock2] as [ShiftHourBlock2], [_].[ShiftHourBlock3] as [ShiftHourBlock3], [_].[ShiftHourBlock4] as [ShiftHourBlock4], [_].[ShiftHourBlock5] as [ShiftHourBlock5], [_].[ShiftHourBlock6] as [ShiftHourBlock6] from [Dim].[ShiftControl] as [_]) AS [t10] on ([t7].[CurrentShiftID] = [t10].[CurrentShiftID]))WHERE (((([t1].[Location Key] = N'111') AND ([t2].[Machine Model] = '190')) AND ([t2].[Machine Cell] = 2) AND ([t2].[Cell Position] = 1))) AND ([t9].[Repair Key] = 'REPAIR') AND ([t10].[ShiftControlID] = 1)))GROUP BY [t8].[Repair Reason Name]) AS [MainTable] WHERE (NOT([a0] IS NULL))ORDER BY [a0]DESCOPTION(LABEL = N'{"DatasetId":"6e8664ad-20f8-4605-98a7-a8aae3d9a738", "Sources":[{"ReportId":"d636d55f-6128-4e5f-8aa7-f2c771f9faba","VisualId":"89cable8a93ec32f33e9"}]}')</pre>	Indexes <pre>-- Fact table CREATE NONCLUSTERED INDEX IX_FactScrapLiveScreens_Filter ON dbo.FactScrapLiveScreens (PlantID, MachineID, RepairID, RepairReasonID, CurrentShiftID, Qty) INCLUDE (DateID, HourID, ShiftID, Lbs); -- DimLocation CREATE NONCLUSTERED INDEX IX_DimLocation_LocationID_Key ON dbo.DimLocation (LocationID, LocationKey); -- DimMachine CREATE NONCLUSTERED INDEX IX_DimMachine_MachineID_Model_Cell ON dbo.DimMachine (MachineID, MachineModel, MachineCell, CellPosition); -- DimRepairReason CREATE NONCLUSTERED INDEX IX_DimRepairReason_RepairReasonID ON dbo.DimRepairReason (RepairReasonID); -- DimRepair CREATE NONCLUSTERED INDEX IX_DimRepair_RepairID_Key ON dbo.DimRepair (RepairID, RepairKey); -- ShiftControl CREATE NONCLUSTERED INDEX IX_ShiftControl_CurrentShiftID ON Dim.ShiftControl (CurrentShiftID, ShiftControlID);</pre>	
---	--	--

10	<pre>SELECT TOP (1000001) *FROM (SELECT *FROM (SELECT [t4].[Product Desc] AS [c50],SUM([t3].[\$Inventory Quantity]) AS [a0]FROM (((select [_].[ID] as [ID], [_].[PlantID] as [PlantID], [_].[ProductID] as [ProductID], [_].[InventoryCodeID] as [InventoryCodeID], [_].[InventoryTypeID] as [InventoryTypeID], [_].[InventoryStatusID] as [InventoryStatusID], [_].[LocatorID] as [LocatorID], [_].[LOT_NUMBER] as [LOT_NUMBER], [_].[LPN_ID] as [LPN_ID], [_].[AgeID] as [AgeID], [_].[Inventory Cost] as [\$Inventory Cost], [_].[Quantity] as [\$Inventory Quantity], [_].[Volume] as [\$Inventory Volume], [].[Average Age] as [Average Age]from [dbo].[FactInventoryLPNCurrent] as []) AS [t3] LEFT OUTER JOIN (select [_].[LocationID] as [LocationID], [_].[LocationKey] as [Location Key], [_].[LocationName] as [Location Name], [_].[LocationDesc] as [Location Desc], [_].[Sort] as [Sort], [_].[LocationType] as [Location Type], [_].[LocationCountry] as [LocationCountry], [_].[ShiftScalar] as [ShiftScalar], [_].[GLLocationID] as [GLLocationID], [_].[WarehouseCapacity] as [\$WarehouseCapacity], [_].[PhysicalLocation] as [Physical Location], [_].[PlanID] as [PlanID], [_].[COGSLocationID] as [COGSLocationID]from [dbo].[DimLocation] as []) AS [t1] on ([t3].[PlantID] = [t1].[LocationID])) LEFT OUTER JOIN (SELECT [t2].[LocatorID] AS [LocatorID],(COALESCE([t2].[Segment 1], '') + (N'.' + (COALESCE([t2].[Segment 2], '') + (N'.' + COALESCE([t2].[Segment 3], '')))) AS [Locator Code]FROM (select [_].[LocatorID] as [LocatorID], [_].[LocatorKey] as [Locator Key], [_].[LocationKey] as [LocationKey], [_].[LocatorDesc] as [Locator Desc], [_].[LocationType] as [Location Type], [_].[PickingOrder] as [Picking Order], [_].[SubinventoryCode] as [Subinventory Code], [_].[Segment1] as [Segment 1], [_].[Segment2] as [Segment 2], [_].[Segment3] as [Segment 3]from [dbo].[DimLocator] as [])) AS [t2]) AS [t2] on ([t3].[LocatorID] = [t2].[LocatorID])) LEFT OUTER JOIN (select [_].[ProductID] as [Product ID], [_].[ProductKey] as [Product Key], [_].[ProductName] as [Product Name], [_].[ProductDesc] as [Product Desc], [_].[ProductSort] as [Product Sort], [_].[4 Digit] as [4 Digit], [_].[UOM] as [UOM], [_].[Item Type] as [Item Type], [_].[SIOP</pre>	Optimized Query (Needs to be tested) <pre>SELECT TOP (1000001) * FROM (SELECT [t4].[ProductDesc] AS [c50], SUM([t3].[Quantity]) AS [a0] FROM [dbo].[FactInventoryLPNCurrent] AS [t3] LEFT JOIN [dbo].[DimLocation] AS [t1] ON [t3].[PlantID] = [t1].[LocationID] LEFT JOIN (SELECT [LocatorID], COALESCE([Segment1], '') + '.' + COALESCE([Segment2], '') + '.' + COALESCE([Segment3], '') AS [LocatorCode] FROM [dbo].[DimLocator]) AS [t2] ON [t3].[LocatorID] = [t2].[LocatorID] LEFT JOIN (SELECT [ProductID], [ProductDesc], [Part Class] FROM [dbo].[DimProductMaster] WHERE [HasInventory] = 1 AND [HasInventory] IS NOT NULL) AS [t4] ON [t3].[ProductID] = [t4].[ProductID] WHERE [t1].[LocationKey] = N'111' AND [t2].[LocatorCode] = N'BF.SM.03' AND [t4].[Part Class] = N'PIGMENTED RESINS' GROUP BY [t4].[ProductDesc] HAVING SUM([t3].[Quantity]) IS NOT NULL AND SUM([t3].[Quantity]) < 200) AS [MainTable] OPTION (LABEL = N '{"DatasetId":"dlc95450-dc98-4e92-bd97-8fa17cb71cc3","Sources":[{"ReportId":"f6a85ee7-3045-42cf-a309-5a610b6a8e25","VisualId":"1749bd1f6208c98b2852"}]}');</pre>	Summary of Fixes - Replaced [\$Inventory Quantity] with [Quantity], which is the actual column from FactInventoryLPNCurrent. - Moved HAVING conditions inside the GROUP BY scope. - Removed unnecessary nested subqueries to improve readability and performance. - Indexes — see suggestions at the end.
----	--	--	---

Family] as [SIOP Family], [_].[Category] as [Category], [_].[SubCategory] as [Sub Category], [_].[CategoryType] as [Category Type], [_].[Brand] as [Brand], [_].[Inventory Status Code] as [Inventory Status Code], [_].[Country of Origin] as [Country of Origin], [_].[Part Class] as [Part Class], [_].[MakeBuy] as [Make Buy], [_].[Contract Manufacturing] as [Contract Manufacturing], [_].[Channel] as [Channel], [_].[Part Type] as [Part Type], [_].[IMAP] as [IMAP], [_].[NPD] as [NPD], [_].[Product Volume] as [Product Volume], [_].[List Price] as [List Price], [_].[Royalty License Name] as [Royalty License Name], [_].[Shipping Method] as [Shipping Method], [_].[Cooler Size] as [Cooler Size], [_].[FirstProductionDate] as [First Production Date], [_].[LastProductionDate] as [Last Production Date], [_].[FirstSaleDate] as [First Sale Date], [_].[LastSaleDate] as [Last Sale Date], [_].[CreationDate] as [Creation Date], [_].[ProductGroup] as [Product Group], [_].[ProductLine] as [Product Line], [_].[Department] as [Department], [_].[Dimensions] as [Dimensions], [_].[Footprint] as [Footprint], [_].[Cube] as [Cube], [_].[ChildAdult] as [ChildAdult], [_].[Size] as [Size], [_].[Step2 Custom] as [Step2 Custom], [_].[DerivedProductKey] as [Derived Product Key], [_].[PlaceholderName] as [Placeholder Name], [_].[PlaceholderDesc] as [Placeholder Desc], [_].[PlaceholderType] as [Placeholder Type], [_].[Forecast Segment] as [Forecast Segment], [_].[UPC] as [UPC], [_].[4 Digit SKU] as [4 Digit SKU], [_].[ProductStatus] as [Product Status], [_].[VENDOR_NAME] as [Vendor Name], [_].[VENDOR_NAME_ALTERNATE] as [Vendor Name Alternate], [_].[PLANNER_CODE] as [Planner Code], [_].[BUYER_NAME] as [Buyer Name], [_].[Product Name Consolidated] as [Product Name Consolidated], [_].[Visible] as [Visible], [_].[HasSCP] as [HasSCP], [_].[HasInventory] as [HasInventory], [_].[Supercategory_NEW] as [Supercategory_NEW], [_].[Category_NEW] as [Category_NEW], [_].[Sub-Category_NEW] as [Sub-Category_NEW], [_].[ProductType NEW] as [	Indexes <pre>-- FactInventoryLPNCurrent CREATE NONCLUSTERED INDEX IX_FactInventory_Plant_Product_Locator ON dbo.FactInventoryLPNCurrent (PlantID, ProductID, LocatorID) INCLUDE (Quantity); --DimLocation CREATE NONCLUSTERED INDEX IX_DimLocation_LocationID ON dbo.DimLocation (LocationID) INCLUDE (LocationKey); --DimProductMaster CREATE NONCLUSTERED INDEX IX_DimProductMaster_ProductID ON dbo.DimProductMaster (ProductID) INCLUDE (ProductDesc, [Part Class], HasInventory);</pre>	
---	--	--

	<pre>ProductType_NEW], [_].[Brand_NEW] as [Brand_NEW], [_] .[License] as [License], [_].[Product Family] as [Product Family], [_].[US Exclusive] as [US Exclusive], [_].[ABCD Code] as [ABCD Code], [_].[Safety Stock Variability] as [Safety Stock Variability], [_].[Weight_Assembled] as [Weight_Assembled], [_].[Volume_Assembled] as [Volume_Assembled], [_].[Length_Assembled] as [Length_Assembled], [_].[Width_Assembled] as [Width_Assembled], [_].[Height_Assembled] as [Height_Assembled]from [dbo].[DimProductMaster] as [_]where [_].[HasInventory] = 1 and [_].[HasInventory] is not null) AS [t4] on ([t3].[ProductID] = [t4].[Product ID]))WHERE ((([t1].[Location Key] = N'111') AND ([t2].[Locator Code] = N'BF.SM.03')) AND ([t4].[Part Class] = N'PIGMENTED RESINS')) GROUP BY [t4].[Product Desc]) AS [MainTable]WHERE (NOT(([a0] IS NULL)))) AS [MainTable]WHERE ((COALESCE([a0], 0) < 2.E2))OPTION(LABEL = N'{"DatasetId":"d1c95450-dc98-4e92- bd97-8fa17cb71cc3","Sources":[{"ReportId":"f6a85ee7-3045- 42cf-a309- 5a610b6a8e25","VisualId":"1749bd1f6208c98b2852"}]} ')</pre>		
13			
14			

Below is not ready yet.

2. Frequent Queries

	Execution Count	Avg Elapsed Time (µs)	Current Query	Proposed Query or Recommended Fix
1				
2				
3				
4				
5				
6				
7				
8				
9				

3. Indexes that need to be rebuilt

1. The table below shows all the indexes with fragmentation greater than __%. Please see the General Fragmentation Levels as a reference:
- **Low Fragmentation (0-10%):** Generally, no action is needed.
 - **Moderate Fragmentation (10-30%):** Consider reorganizing the index.
 - **High Fragmentation (30% and above):** Consider rebuilding the index.

	Schema	Table	Index	avg_fragmentation_in_percent
1)				
2)				
3)				
4)				
5)				
6)				
7)				
8)				
9)				
10)				
11)				
12)				
13)				
14)				
15)				
16)				
17)				
18)				
19)				
20)				
21)				
22)				
23)				
24)				
25)				
26)				
27)				

28)				
29)				
30)				
31)				
32)				
33)				
34)				
35)				
36)				
37)				
38)				
39)				
40)				
41)				
42)				
43)				
44)				
45)				
46)				
47)				
48)				
49)				
50)				
51)				
52)				

To Rebuild or Reorganize the above Indexes:

- **Rebuild Index:** This operation drops and re-creates the index. It is more resource-intensive but results in a more compact and efficient index.

```
ALTER INDEX [IndexName] ON [TableName] REBUILD;
```

- **Reorganize Index:** This operation defragments the leaf level of the index. It is less resource-intensive and can be done online.

```
ALTER INDEX [IndexName] ON [TableName] REORGANIZE;
```

4. Missing Indexes

No.	Proposed Index	Table	User Seeks
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			

5. Non-beneficial Indexes

TableName	IndexName	index_id	user_seeks	user_scans	user_lookups	user_updates

Example:

```
SELECT column1, column2 FROM YourTable WHERE condition;
```

- 2. **Use Indexes in WHERE, JOIN, and ORDER BY Clauses:**
 - Write queries that leverage indexed columns in WHERE, JOIN, and ORDER BY clauses.
- 3. **Use EXISTS Over IN for Subqueries:**
 - For checking existence, use EXISTS instead of IN to improve performance, especially when dealing with large data sets.

Example:

```
SELECT column1 FROM YourTable WHERE EXISTS (SELECT 1 FROM AnotherTable WHERE condition);
```

- 4. **Minimize Cursors:**
 - Replace cursors with set-based operations where possible, as cursors are resource-intensive.

7. Update Statistics Regularly

Task: Ensure SQL Server has up-to-date statistics to optimize query plans.

- Action:** All databases were already set to auto-update.
- Recommendation:** Use full scan statistics updates for large or critical tables to ensure the optimizer has accurate data distributions.

Conclusion
