# T-SQL Performance Tuning Dashboard Project

This project will create a **Performance Tuning Dashboard** using SQL Server **views** that track long-running queries, index fragmentation, and deadlocks. The dashboard will leverage **Dynamic Management Views (DMVs)** to provide real-time insights for performance optimization.

**1. Directory Structure**

Organize your SQL scripts into a structured format:

PerformanceTuningDashboard/

│

├── Views/

│   ├── vw\_LongRunningQueries.sql

│   ├── vw\_IndexFragmentation.sql

│   ├── vw\_Deadlocks.sql

│

├── Procedures/

│   ├── sp\_RebuildIndexes.sql

│

├── Scripts/

│   ├── Create\_Performance\_Dashboard.sql

│

└── README.md

**2. SQL Components**

**A. View for Long-Running Queries**

This view tracks queries that have been running for a long time (e.g., more than 5 seconds):

CREATE VIEW vw\_LongRunningQueries AS

SELECT

    s.session\_id,

    r.status,

    r.start\_time,

    r.total\_elapsed\_time / 1000 AS ElapsedTime\_Sec,

    s.login\_name,

    r.command,

    q.text AS QueryText,

    s.host\_name,

    s.program\_name

FROM sys.dm\_exec\_requests r

JOIN sys.dm\_exec\_sessions s ON r.session\_id = s.session\_id

CROSS APPLY sys.dm\_exec\_sql\_text(r.sql\_handle) q

WHERE r.total\_elapsed\_time > 5000;  -- Queries running longer than 5 sec

**Usage:**

SELECT \* FROM vw\_LongRunningQueries;

**B. View for Index Fragmentation**

This view identifies indexes with fragmentation above 30% that may need optimization.

CREATE VIEW vw\_IndexFragmentation AS

SELECT

    dbschemas.[name] AS SchemaName,

    dbtables.[name] AS TableName,

    dbindexes.[name] AS IndexName,

    indexstats.index\_id,

    indexstats.avg\_fragmentation\_in\_percent

FROM sys.dm\_db\_index\_physical\_stats(DB\_ID(), NULL, NULL, NULL, 'LIMITED') AS indexstats

JOIN sys.tables dbtables ON dbtables.[object\_id] = indexstats.[object\_id]

JOIN sys.schemas dbschemas ON dbtables.[schema\_id] = dbschemas.[schema\_id]

JOIN sys.indexes dbindexes ON dbindexes.[object\_id] = indexstats.[object\_id]

AND indexstats.index\_id = dbindexes.index\_id

WHERE indexstats.avg\_fragmentation\_in\_percent > 30

ORDER BY indexstats.avg\_fragmentation\_in\_percent DESC;

**Usage:**

SELECT \* FROM vw\_IndexFragmentation;

**C. View for Deadlocks**

This view helps track recent deadlocks by querying the system health extended events.

CREATE VIEW vw\_Deadlocks AS

SELECT

    XEvent.query('(event/data/value/deadlock/process-list)[1]') AS ProcessList,

    XEvent.query('(event/data/value/deadlock/resource-list)[1]') AS ResourceList,

    XEvent.query('(event/data/value/deadlock)[1]') AS DeadlockGraph,

    EventData.value('@timestamp', 'datetime') AS DeadlockTime

FROM (SELECT CAST(target\_data AS XML) AS TargetData

      FROM sys.dm\_xe\_session\_targets AS t

      JOIN sys.dm\_xe\_sessions AS s

      ON t.event\_session\_address = s.address

      WHERE s.name = 'system\_health'

      AND t.target\_name = 'ring\_buffer') AS X

CROSS APPLY TargetData.nodes('//RingBufferTarget/event') AS XEvent(EventData)

WHERE EventData.value('@name', 'nvarchar(4000)') = 'xml\_deadlock\_report';

**Usage:**

SELECT \* FROM vw\_Deadlocks;

**3. Stored Procedure for Index Optimization**

This stored procedure **rebuilds fragmented indexes** if fragmentation is above 30%.

CREATE PROCEDURE sp\_RebuildIndexes AS

BEGIN

    DECLARE @TableName NVARCHAR(128);

    DECLARE @IndexName NVARCHAR(128);

    DECLARE @SQL NVARCHAR(MAX);

    DECLARE IndexCursor CURSOR FOR

    SELECT

        dbschemas.[name] AS SchemaName,

        dbtables.[name] AS TableName,

        dbindexes.[name] AS IndexName

    FROM sys.dm\_db\_index\_physical\_stats(DB\_ID(), NULL, NULL, NULL, 'LIMITED') AS indexstats

    JOIN sys.tables dbtables ON dbtables.[object\_id] = indexstats.[object\_id]

    JOIN sys.schemas dbschemas ON dbtables.[schema\_id] = dbschemas.[schema\_id]

    JOIN sys.indexes dbindexes ON dbindexes.[object\_id] = indexstats.[object\_id]

    AND indexstats.index\_id = dbindexes.index\_id

    WHERE indexstats.avg\_fragmentation\_in\_percent > 30;

    OPEN IndexCursor;

    FETCH NEXT FROM IndexCursor INTO @TableName, @IndexName;

    WHILE @@FETCH\_STATUS = 0

    BEGIN

        SET @SQL = 'ALTER INDEX [' + @IndexName + '] ON [' + @TableName + '] REBUILD;';

        EXEC sp\_executesql @SQL;

        FETCH NEXT FROM IndexCursor INTO @TableName, @IndexName;

    END;

    CLOSE IndexCursor;

    DEALLOCATE IndexCursor;

END;

**Usage:**

EXEC sp\_RebuildIndexes;

**4. Master Script to Create the Dashboard**

This script **creates all views and procedures** in one execution.

-- Create Views

PRINT 'Creating Views...';

GO

:r Views/vw\_LongRunningQueries.sql

GO

:r Views/vw\_IndexFragmentation.sql

GO

:r Views/vw\_Deadlocks.sql

GO

-- Create Stored Procedures

PRINT 'Creating Stored Procedures...';

GO

:r Procedures/sp\_RebuildIndexes.sql

GO

PRINT 'Performance Tuning Dashboard Created Successfully!';

**Execution:**

:r Scripts/Create\_Performance\_Dashboard.sql

**5. Usage and Monitoring**

1. **Check long-running queries:**

SELECT \* FROM vw\_LongRunningQueries;

1. **Identify fragmented indexes:**

SELECT \* FROM vw\_IndexFragmentation;

1. **Monitor deadlocks:**

SELECT \* FROM vw\_Deadlocks;

1. **Optimize indexes:**

EXEC sp\_RebuildIndexes;

**6. Benefits of This Approach**

**Real-time monitoring** using DMV views.  
**Easy performance tuning** with a structured dashboard.  
**Automated index maintenance** for optimized performance.  
**Deadlock tracking** to identify and resolve contention issues.