Understanding SQL Missing Index Recommendations

1. What this script does:
   - SQL Server tracks queries and identifies missing indexes that could improve performance.
   - The script provides suggestions based on its analysis.
2. Caution:
   - Do not blindly create all suggested indexes.
   - Performance tuning is an art and requires experience and careful planning.
   - Implement suggested indexes gradually after review.
3. Key columns to focus on:
   - AvgUserImpact → estimated improvement for queries if the index is created.
   - UniqueCompiles → number of unique query compilations affected.
   - UserSeeks → how many times the index would be used in seeks.
   - UserScans → how many times the index would be used in scans.
4. Best practices for evaluating indexes:
   - Do not rely on a single column like AvgUserImpact.
   - Consider both impact and frequency of use:
     - An index with lower AvgUserImpact but frequent use may be more beneficial than one with high impact but rarely used.
   - Review existing indexes on the table before creating new ones to see if extending or modifying an existing index can provide the same benefit.
5. Remember:
   - Every additional index adds maintenance overhead (updates, inserts, deletes).
   - Be selective and strategic when adding indexes.

Stored Procedure: GetMissingIndexes → for a required DB

Usage

--Exec GetMissingIndexes @DBName ='Advworks'

```
Use master
go
CREATE PROCEDURE dbo.GetMissingIndexes
(
  @DBName SYSNAME
)
AS
BEGIN
  SET NOCOUNT ON;
  DECLARE @SQL NVARCHAR(MAX);
  SET @SQL = N'
  USE ' + QUOTENAME(@DBName) + ';
  SELECT
    DB_NAME() AS [DatabaseName],
    OBJECT_NAME(id.[object_id], DB_ID()) AS [ObjectName],
    id.[statement] AS [FullyQualifiedObjectName],
        id.[equality_columns] AS [EqualityColumns],
        id.[inequality_columns] AS [InEqualityColumns],
        id.[included_columns] AS [IncludedColumns],
        gs.[unique_compiles] AS [UniqueCompiles],
        gs.[user_seeks] AS [UserSeeks],
        gs.[user_scans] AS [UserScans],
        gs.[last_user_seek] AS [LastUserSeekTime],
        gs.[last_user_scan] AS [LastUserScanTime],
        gs.[avg_total_user_cost] AS [AvgTotalUserCost],
        gs.[avg_user_impact] AS [AvgUserImpact],
        gs.[user_seeks] * gs.[avg_total_user_cost] * (gs.[avg_user_impact] * 0.01) AS [IndexAdvantage],
        CONCAT(
          ''CREATE INDEX [IX_'',
          OBJECT_NAME(id.[object_id], DB_ID()),
          ''_'',
          REPLACE(REPLACE(REPLACE(ISNULL(id.[equality_columns], ''''), '', '', ''_''), ''['', ''''), '']'', ''''),
          CASE WHEN id.[equality_columns] IS NOT NULL AND id.[inequality_columns] IS NOT NULL THEN ''_'' ELSE '''' END,
          REPLACE(REPLACE(REPLACE(ISNULL(id.[inequality_columns], ''''), '', '', ''_''), ''['', ''''), '']'', ''''),
          ''_'',
          LEFT(CAST(NEWID() AS NVARCHAR(36)), 5),
          ''] ON '',
          id.[statement],
          '' ('',
          ISNULL(id.[equality_columns], ''''),
          CASE WHEN id.[equality_columns] IS NOT NULL AND id.[inequality_columns] IS NOT NULL THEN '','' ELSE '''' END,
          ISNULL(id.[inequality_columns], ''''),
          '')'',
          ISNULL(CONCAT('' INCLUDE ('', id.[included_columns], '')''), '''')
        ) AS [ProposedIndex],
        CAST(CURRENT_TIMESTAMP AS smalldatetime) AS [CollectionDate]
    FROM sys.dm_db_missing_index_group_stats gs WITH (NOLOCK)
    INNER JOIN sys.dm_db_missing_index_groups ig WITH (NOLOCK) ON gs.[group_handle] = ig.[index_group_handle]
    INNER JOIN sys.dm_db_missing_index_details id WITH (NOLOCK) ON ig.[index_handle] = id.[index_handle]
    WHERE DB_ID() = DB_ID()
    ORDER BY ObjectName, [IndexAdvantage] DESC
    OPTION (RECOMPILE);
    ';

    EXEC sp_executesql @SQL;
END;
```

Stored Procedure: GetMissingIndexes_AllDatabases

Full server-wide missing index stored procedure that scans all user databases, collects missing index recommendations, and outputs IndexAdvantage, proposed CREATE INDEX script, usage stats, and timestamp.

Key Features

1. **Scans all user databases** automatically.
2. Computes **missing index advantage** for each recommendation.
3. Generates **ready-to-run CREATE INDEX statements**.
4. Collects **usage stats**: seeks, scans, last seek/scan, average cost, and impact.
5. Stores results in a **temporary table** for easy querying or exporting.
6. Ordered by **Database → Object → IndexAdvantage descending**.

Usage

```
EXEC dbo.GetMissingIndexes_AllDatabases;
```

This will produce a **server-wide missing index report** similar to your earlier single database script but **automated across all databases**.

```
Use master
go
CREATE PROCEDURE dbo.GetMissingIndexes_AllDatabases
AS
BEGIN
   SET NOCOUNT ON;

   -- Temporary table to store results from all DBs
   IF OBJECT_ID('tempdb..#MissingIndexes') IS NOT NULL
      DROP TABLE #MissingIndexes;

   CREATE TABLE #MissingIndexes
   (
      DatabaseName SYSNAME,
      ObjectName SYSNAME,
      FullyQualifiedObjectName NVARCHAR(MAX),
      EqualityColumns NVARCHAR(MAX),
      InEqualityColumns NVARCHAR(MAX),
      IncludedColumns NVARCHAR(MAX),
      UniqueCompiles BIGINT,
      UserSeeks BIGINT,
      UserScans BIGINT,
      LastUserSeek DATETIME,
      LastUserScan DATETIME,
      AvgTotalUserCost FLOAT,
      AvgUserImpact FLOAT,
    IndexAdvantage FLOAT,
    ProposedIndex NVARCHAR(MAX),
    CollectionDate SMALLDATETIME
);

DECLARE @DB SYSNAME;
DECLARE @SQL NVARCHAR(MAX);

-- Cursor for all online user databases
DECLARE db_cursor CURSOR FOR
SELECT name FROM sys.databases
WHERE database_id > 4 AND state_desc='ONLINE';

OPEN db_cursor;
FETCH NEXT FROM db_cursor INTO @DB;

WHILE @@FETCH_STATUS = 0
BEGIN
    SET @SQL = N'
    USE ' + QUOTENAME(@DB) + ';

    INSERT INTO #MissingIndexes
    SELECT
        DB_NAME() AS [DatabaseName],
        OBJECT_NAME(id.[object_id], DB_ID()) AS [ObjectName],
        id.[statement] AS [FullyQualifiedObjectName],
        id.[equality_columns] AS [EqualityColumns],
        id.[inequality_columns] AS [InEqualityColumns],
        id.[included_columns] AS [IncludedColumns],
        gs.[unique_compiles] AS [UniqueCompiles],
        gs.[user_seeks] AS [UserSeeks],
        gs.[user_scans] AS [UserScans],
        gs.[last_user_seek] AS [LastUserSeek],
        gs.[last_user_scan] AS [LastUserScan],
        gs.[avg_total_user_cost] AS [AvgTotalUserCost],
        gs.[avg_user_impact] AS [AvgUserImpact],
        gs.[user_seeks] * gs.[avg_total_user_cost] * (gs.[avg_user_impact] * 0.01) AS [IndexAdvantage],
        CONCAT(
            ''CREATE INDEX [IX_'',
            OBJECT_NAME(id.[object_id], DB_ID()),
            ''_'',
            REPLACE(REPLACE(REPLACE(ISNULL(id.[equality_columns], ''''), '', '', ''_''), ''['', ''''), '']'', ''''),
            CASE WHEN id.[equality_columns] IS NOT NULL AND id.[inequality_columns] IS NOT NULL THEN ''_'' ELSE '''' END,
            REPLACE(REPLACE(REPLACE(ISNULL(id.[inequality_columns], ''''), '', '', ''_''), ''['', ''''), '']'', ''''),
            ''_'',
            LEFT(CAST(NEWID() AS NVARCHAR(36)),5),
            ''] ON '',
            id.[statement],
            '' ('',
            ISNULL(id.[equality_columns], ''''),
            CASE WHEN id.[equality_columns] IS NOT NULL AND id.[inequality_columns] IS NOT NULL THEN '','' ELSE '''' END,
            ISNULL(id.[inequality_columns], ''''),
            '')'',
            ISNULL(CONCAT('' INCLUDE ('', id.[included_columns], '')''), '''')
          ) AS [ProposedIndex],
          CAST(CURRENT_TIMESTAMP AS smalldatetime) AS [CollectionDate]
        FROM sys.dm_db_missing_index_group_stats gs WITH (NOLOCK)
        INNER JOIN sys.dm_db_missing_index_groups ig WITH (NOLOCK)
          ON gs.[group_handle] = ig.[index_group_handle]
        INNER JOIN sys.dm_db_missing_index_details id WITH (NOLOCK)
          ON ig.[index_handle] = id.[index_handle]
        WHERE DB_ID() = DB_ID();
        ';

        EXEC sp_executesql @SQL;

        FETCH NEXT FROM db_cursor INTO @DB;
    END

    CLOSE db_cursor;
    DEALLOCATE db_cursor;

    -- Final select of all collected missing indexes
    SELECT *
    FROM #MissingIndexes
    ORDER BY DatabaseName, ObjectName, IndexAdvantage DESC;
END;
GO
```

Fully automated missing index email report across all user databases.

This will use the GetMissingIndexes_AllDatabases procedure, store the results in a temporary table, and send them via **Database Mail** in a clean HTML format.

Step 1 — Ensure Database Mail is configured

If not already done:

-- Enable Database Mail

```
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
EXEC sp_configure 'Database Mail XPs', 1;
RECONFIGURE;
```

-- Create a Database Mail profile (example)

```
EXEC msdb.dbo.sysmail_add_account_sp
   @account_name = 'DBAAlerts',
   @description = 'DBA notifications',
   @email_address = 'dba_team@example.com',
   @display_name = 'SQL Server DBA Alerts',
   @mailserver_name = 'smtp.example.com';

EXEC msdb.dbo.sysmail_add_profile_sp
   @profile_name = 'DBAAlertsProfile',
   @description = 'Profile for DBA notifications';

EXEC msdb.dbo.sysmail_add_profileaccount_sp
   @profile_name = 'DBAAlertsProfile',
   @account_name = 'DBAAlerts',
   @sequence_number = 1;
```

Step 2 — Stored Procedure to Send Missing Index Report

```
 Use master
CREATE PROCEDURE dbo.SendMissingIndexesReport
AS
BEGIN
   SET NOCOUNT ON;

   -- Temporary table for collected results
   IF OBJECT_ID('tempdb..#AllMissingIndexes') IS NOT NULL
      DROP TABLE #AllMissingIndexes;

   CREATE TABLE #AllMissingIndexes
   (
      DatabaseName SYSNAME,
   ObjectName SYSNAME,
   FullyQualifiedObjectName NVARCHAR(MAX),
   EqualityColumns NVARCHAR(MAX),
   InEqualityColumns NVARCHAR(MAX),
   IncludedColumns NVARCHAR(MAX),
   UniqueCompiles BIGINT,
   UserSeeks BIGINT,
   UserScans BIGINT,
   LastUserSeek DATETIME,
   LastUserScan DATETIME,
   AvgTotalUserCost FLOAT,
   AvgUserImpact FLOAT,
   IndexAdvantage FLOAT,
   ProposedIndex NVARCHAR(MAX),
   CollectionDate SMALLDATETIME
);

-- Populate results from all DBs
INSERT INTO #AllMissingIndexes
EXEC dbo.GetMissingIndexes_AllDatabases;

-- Build HTML table for email
DECLARE @html NVARCHAR(MAX);

SET @html = N'<h2>Weekly Missing Index Report - ' + CONVERT(NVARCHAR, GETDATE(), 120) + '</h2>'
        + N'<table border="1" cellspacing="0" cellpadding="3">'
        + N'<tr>
             <th>Database</th>
             <th>Object</th>
             <th>User Seeks</th>
             <th>User Scans</th>
             <th>Avg Total Cost</th>
             <th>Avg User Impact</th>
             <th>Index Advantage</th>
             <th>Proposed Index</th>
           </tr>';

SELECT @html = @html +
      CONCAT(
          '<tr>',
          '<td>', DatabaseName, '</td>',
          '<td>', ObjectName, '</td>',
          '<td>', UserSeeks, '</td>',
          '<td>', UserScans, '</td>',
          '<td>', CAST(AvgTotalUserCost AS NVARCHAR), '</td>',
          '<td>', CAST(AvgUserImpact AS NVARCHAR), '</td>',
                '<td>', CAST(IndexAdvantage AS NVARCHAR), '</td>',
                '<td>', ProposedIndex, '</td>',
                '</tr>'
            )
        FROM #AllMissingIndexes;

        SET @html = @html + N'</table>';

        -- Send email
        EXEC msdb.dbo.sp_send_dbmail
            @profile_name = 'DBAAlertsProfile',
            @recipients = 'dba_team@example.com',
            @subject = 'Weekly Missing Index Report',
            @body = @html,
            @body_format = 'HTML';
    END;
    GO
```

Step 3 — Schedule SQL Agent Job

1. Open **SQL Server Agent → Jobs → New Job**
2. Name: Weekly Missing Index Report
3. Steps → New Step:
    - Type: **Transact-SQL script (T-SQL)**
    - Command:

      ```
      EXEC dbo.SendMissingIndexesReport;
      ```
4. Schedule → **Weekly** → e.g., Sunday 2 AM
5. Notifications → optional: alert if job fails

Features

- Scans **all online user databases** automatically
- Collects **missing index recommendations** with **IndexAdvantage**
- Generates **proposed CREATE INDEX T-SQL**
- Sends **HTML email report** to DBA team
- Fully automated via **SQL Agent Job**

Reference: https://www.sqlservercentral.com/scripts/missing-index-script
Author: David Waller