CSC Global DBA Lab Workbook

DAY 2 – Module 3: Index & Statistics Tuning

Database: CSC_PerfDemo

🔷 LAB OBJECTIVE

By the end of this lab, participants will be able to:

- Identify **bad plans** caused by:
  - Missing covering indexes
  - Non-optimal index design
  - Fragmentation myths
  - Poor statistics
- Fix them using:
  - Covering & filtered indexes
  - Correct rebuild vs reorganize strategy
  - Statistics & histograms
- Validate improvements using **Execution Plans + DMVs**

🧪 STEP 1 — Connect & Baseline Setup

Action

```
USE CSC_PerfDemo;
GO
SET STATISTICS IO ON;
SET STATISTICS TIME ON;
```

Trainer Notes

- Explain logical reads & CPU time
- Ask participants to **enable Actual Execution Plan** in SSMS

🧪 STEP 2 — Use Case 1: Key Lookup Problem (BAD PLAN)

Scenario

Invoice query is slow for entities with many invoices.

Run (BAD query)

```
DECLARE @EntityId INT = 1;

SELECT invoice_id, due_date, amount, currency_code
FROM dbo.Invoices
WHERE entity_id = @EntityId
  AND invoice_status = 'Open'
ORDER BY due_date DESC;
```

Expected Plan Shape (BAD)

- Index Seek on nonclustered index
- ⚠️ **Key Lookup (Clustered)** operator
- Possible Sort operator
- High logical reads

Notes

- Highlight **Key Lookup**
- Explain why lookups scale badly with row count

🧪 STEP 3 — Fix Use Case 1: Covering Index (GOOD PLAN)

Run

```
CREATE INDEX IX_Invoices_EntityStatus_DueDate_Cover
ON dbo.Invoices (entity_id, invoice_status, due_date DESC)
INCLUDE (amount, currency_code);
GO
```

Re-run Query

```
SELECT invoice_id, due_date, amount, currency_code
FROM dbo.Invoices
WHERE entity_id = @EntityId
  AND invoice_status = 'Open'
ORDER BY due_date DESC;
```

Expected Plan Shape (GOOD)

- Index Seek

- ✅ **No Key Lookup**
- ✅ No Sort
- Dramatic drop in logical reads

🧪 STEP 4 — Use Case 2: Rare Predicate Without Filtered Index (BAD)

Scenario

Compliance team frequently checks **Overdue filings** (rare).

Run

```
SELECT TOP (2000)
    filing_id, entity_id, due_date, penalty_amount
FROM dbo.ComplianceFilings
WHERE filing_status = 'Overdue'
ORDER BY due_date DESC;
```

Expected Plan Shape (BAD)

- Index Scan or wide Index Seek
- Reads many non-relevant rows
- Higher logical reads than expected

🧪 STEP 5 — Fix Use Case 2: Filtered Index

Run

```
CREATE INDEX IX_Filings_Overdue_DueDate
ON dbo.ComplianceFilings (due_date DESC)
INCLUDE (entity_id, penalty_amount)
WHERE filing_status = 'Overdue';
GO
```

Re-run Query

Expected Plan Shape (GOOD)

- Index Seek on **Filtered Index**
- Very low logical reads
- No scan on main table

Notes

- Emphasize **filtered index = smaller + faster**
- Perfect for CSC compliance-style queries

STEP 6 — Use Case 3: Fragmentation Myth (Measure First)

Scenario

DBAs blindly rebuild all indexes weekly.

Run

```
SELECT
    OBJECT_NAME(ips.object_id) AS table_name,
    ips.index_id,
    ips.avg_fragmentation_in_percent,
    ips.page_count
FROM sys.dm_db_index_physical_stats
    (DB_ID(), NULL, NULL, NULL, 'SAMPLED') ips
WHERE ips.page_count > 1000
ORDER BY ips.avg_fragmentation_in_percent DESC;
```

Notes

- Explain why:
  - Small indexes don't matter
  - Fragmentation ≠ performance always

STEP 7 — Reorganize vs Rebuild (Correct Decision)

Case A — Medium fragmentation

```
ALTER INDEX IX_Invoices_Status_Due
ON dbo.Invoices REORGANIZE;
GO
```

Case B — High fragmentation

```
ALTER INDEX IX_Invoices_Status_Due
ON dbo.Invoices REBUILD WITH (FILLFACTOR = 90);
GO
```

Expected Outcome

- Fragmentation reduced

- Explain **fill factor trade-offs**

STEP 8 — Use Case 4: Bad Estimates Due to Statistics (BAD PLAN)

Scenario

Correlated columns → wrong estimates → wrong joins.

Run

```
SELECT entity_id, entity_name
FROM dbo.Entities
WHERE country_code = 'IN'
  AND risk_tier = 5;
```

Expected Plan Shape (BAD)

- Estimated rows ≠ Actual rows
- Potential scan instead of seek
- Poor join choice in larger queries

STEP 9 — Inspect Statistics Histogram

Run

```
DBCC SHOW_STATISTICS ('dbo.Entities', 'IX_Entities_Country')
WITH HISTOGRAM;
GO
```

Trainer Notes

- Explain:
  - Density
  - Histogram buckets
  - Why independence assumption fails

🧪 STEP 10 — Fix with Multi-Column Statistics (GOOD PLAN)

Run

```
CREATE STATISTICS ST_Entities_Country_Risk
ON dbo.Entities (country_code, risk_tier);
GO
```

```
UPDATE STATISTICS dbo.Entities
ST_Entities_Country_Risk WITH FULLSCAN;
GO
```

Re-run Query

Expected Plan Shape (GOOD)

- Better estimated row count
- More stable plans
- Better join selection in complex queries

🎯 **FINAL WRAP-UP**

✔ Why **Key Lookups** hurt scalability

✔ When to use **covering vs filtered indexes**

✔ Why **fragmentation is not a blind rebuild job**

✔ How **statistics & histograms drive execution plans**
