Standard Operating Procedure (SOP) for SQL Server *Query Store* covering **SQL Server 2017 through 2025**. It explains what Query Store is, how to configure it, and how to use it for performance troubleshooting and management.

Reference: https://www.sqlshack.com/sql-server-query-store-overview/

SOP: SQL Server Query Store — Configuration & Usage (SQL 2017–2025)

1. Purpose

To standardize how the **Query Store** feature is enabled, configured, and used in SQL Server databases from version **2017 through 2025**, improving performance monitoring and troubleshooting of query plan issues.

2. Scope

Applies to all SQL Server database instances (on-premises and cloud) running versions **SQL Server 2017, 2019, 2022, and 2025**, including Azure SQL Database and Managed Instance where applicable.
https://www.microsoft.com/en-us/sql-server/blog/2022/08/18/query-store-is-enabled-by-default-in-sql-server-2022/

3. Overview

- Query Store acts like a **flight recorder** for the database: it captures executed queries, associated execution plans, and runtime metrics over time.
- It stores historical query data persistently across restarts, unlike the traditional plan cache.
- SQL Server 2022 and later enable Query Store by default for new databases.
- Query Store runs at the **database level**, not at the instance level.

4. Prerequisites

Permissions

- Minimum: VIEW DATABASE STATE to view data.
- db_owner or elevated privileges required to change Query Store settings, flush data, or force plans.

Supported Versions

- Query Store introduced in **SQL Server 2016**, available and improved in later versions; fully supported and enabled by default in **SQL Server 2022/2025**.

5. Enabling Query Store

From SQL Server Management Studio (SSMS)

1. Right-click the database → *Properties*.
2. Go to the *Query Store* tab.
3. Set **Operation Mode (Requested)** to **Read Write**.
4. Click *OK* to enable.

Using T-SQL

```
ALTER DATABASE YourDatabase
SET QUERY_STORE = ON;
```

Do NOT enable on master or tempdb.

6. Configuration Options (Key Parameters)

These settings control how Query Store captures and retains data:

Setting	Purpose
OPERATION_MODE	Controls whether Query Store collects new data (READ_WRITE) or holds existing (READ_ONLY).
MAX_STORAGE_SIZE_MB	Sets max disk space for Query Store. If full, Query Store goes READ_ONLY.
DATA_FLUSH_INTERVAL_SECONDS	Time between writes from memory to disk (default often ~900s).

Setting	Purpose
INTERVAL_LENGTH_MINUTES	Aggregation interval for runtime stats. Valid values include 1,5,10,15,30,60.
QUERY_CAPTURE_MODE	Determines which queries to store (ALL/AUTO/NONE).
SIZE_BASED_CLEANUP_MODE	Controls automatic cleanup when reaching high capacity (AUTO/OFF).
STALE_QUERY_THRESHOLD_DAYS	How long data lives before automatic removal.
MAX_PLANS_PER_QUERY	Limits number of stored plans per query.

7. Use Cases and Benefits

Query Store enables DBAs to:

- **Identify high-cost/slow queries and regressions** by comparing execution plans historically.
- **Detect plan regressions** where a new plan performs worse.
- **Force "good" plans** to stabilize performance after bad plan changes.
- **Measure resource consumption** over time for CPU, I/O, memory, and execution count.
- Serve as a major performance troubleshooting tool across version migrations or compatibility level changes.

8. Built-In Reports and Analysis

Once enabled, SSMS provides multiple reports under **Query Store**:

- **Regressed Queries**
- **Top Resource Consuming Queries**
- **Tracked Queries**
- **Queries with Forced Plans**
- **Queries with High Variation**
- **Overall Resource Consumption**

These reports show query history, plan changes, and performance over time.

9. Operational Procedures

9.1 Monitoring

- Monitor Query Store state via:

```
SELECT * FROM sys.database_query_store_options;
```

to validate settings.

9.2 Plan Forcing

- Identify regressions and, if appropriate, **force a stable plan** via SSMS UI or T-SQL (sp_query_store_force_plan).

9.3 Flushing Data

- To ensure in-memory stats are written to disk, run:

```
EXEC sp_query_store_flush_db;
```

when needed.

9.4 Cleaning / Resetting

- To clear all stored data:

```
ALTER DATABASE YourDB SET QUERY_STORE CLEAR;
```

Use with caution after validation and backups.

10. Best Practices

- **Enable on all production databases** where performance monitoring and regression tracking is required.
- Ensure **reasonable size thresholds** (e.g., MAX_STORAGE_SIZE_MB) to avoid unexpected READ_ONLY state.
- Configure **capture mode to AUTO** in busy environments to reduce unnecessary overhead.
- Adjust **interval length** and **flush intervals** based on workload and acceptable overhead.
- When upgrading to **SQL Server 2022/2025**, leverage default Query Store behavior and new intelligent capabilities that use this data to improve plan selection.

11. Performance Considerations

- Query Store introduces a small performance overhead (typically ~3–5%).
- Avoid enabling on extremely high-throughput systems without reviewing capture mode and sizing settings first.

12. Troubleshooting Tips

- If Query Store goes to **READ_ONLY** unexpectedly, check MAX_STORAGE_SIZE_MB and space usage; adjust cleanup policies.
- If data appears missing, verify INTERVAL_LENGTH, capture mode, and retention policies.
- Use sp_query_store_remove_plan and sp_query_store_remove_query for fine-grained cleanup when necessary.

13. Documentation & Change Control

- Document all Query Store configuration settings per database.
- Review settings during major compatibility level upgrades or version migration.
- Include Query Store configuration checks in performance review procedures.

14. Summary

The SQL Server Query Store is a powerful performance feature available from 2016 onward, **enabled by default in 2022+**, that persists query execution metrics and plans over time, making performance analysis, regression detection, and tuning significantly easier.

T-SQL template set for **Query Store configuration, monitoring, and management** for SQL Server 2017–2025.
These scripts follow best practices and can be adapted per database.

1. Enable Query Store

```
USE [YourDatabase];
GO
-- Enable Query Store
ALTER DATABASE [YourDatabase]
SET QUERY_STORE = ON;

-- Set Operation Mode to Read Write
ALTER DATABASE [YourDatabase]
SET QUERY_STORE (OPERATION_MODE = READ_WRITE);
```

2. Configure Key Query Store Settings

```
USE [YourDatabase];
GO
ALTER DATABASE [YourDatabase]
SET QUERY_STORE (
   OPERATION_MODE = READ_WRITE,        -- Capture queries
   MAX_STORAGE_SIZE_MB = 500,          -- Adjust as needed
   INTERVAL_LENGTH_MINUTES = 15,       -- Aggregation interval
   CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 30), -- Remove old data
   QUERY_CAPTURE_MODE = AUTO,          -- Only capture relevant queries
   SIZE_BASED_CLEANUP_MODE = AUTO,     -- Auto cleanup when full
   MAX_PLANS_PER_QUERY = 10            -- Limit number of stored plans
);
```


3. Check Current Query Store Settings

```
USE [YourDatabase];
GO
SELECT *
FROM sys.database_query_store_options;
```

4. Monitor Query Store Usage

```
-- Query execution statistics
SELECT
   qsqt.query_sql_text,
   qsp.query_id,
   p.plan_id,
   rs.count_executions,
   rs.avg_duration,
   rs.avg_cpu_time,
   rs.avg_logical_io_reads
FROM sys.query_store_query_text qsqt
JOIN sys.query_store_query qsp
   ON qsqt.query_text_id = qsp.query_text_id
JOIN sys.query_store_plan p
   ON qsp.query_id = p.query_id
JOIN sys.query_store_runtime_stats rs
   ON p.plan_id = rs.plan_id
ORDER BY rs.avg_duration DESC;
```

5. Detect Regressed Queries

```
-- Identify queries where performance degraded
SELECT q.query_id, q.query_text_id, p.plan_id, rs.avg_duration, rs.last_execution_time
FROM sys.query_store_query q
JOIN sys.query_store_plan p ON q.query_id = p.query_id
JOIN sys.query_store_runtime_stats rs ON p.plan_id = rs.plan_id
WHERE rs.avg_duration > 1000 -- threshold in ms, adjust as needed
ORDER BY rs.avg_duration DESC;
```

6. Force a Good Execution Plan

```
-- Force a specific plan to stabilize performance
EXEC sp_query_store_force_plan
   @query_id = 123,  -- Replace with actual query_id
   @plan_id  = 456;  -- Replace with actual plan_id
```

7. Flush Query Store to Disk

```
-- Force Query Store to persist in-memory data to disk
EXEC sp_query_store_flush_db;
```

8. Remove Specific Query or Plan

```
-- Remove a specific plan
EXEC sp_query_store_remove_plan @plan_id = 456;

-- Remove a specific query (all plans for this query)
EXEC sp_query_store_remove_query @query_id = 123;
```


9. Reset Query Store Data

```
-- Clear all stored query data (use with caution)
ALTER DATABASE [YourDatabase] SET QUERY_STORE CLEAR;
```

10. Automate Monitoring & Alerts (Example)

```
-- Alert if Query Store reaches 90% of configured MAX_STORAGE_SIZE_MB
SELECT
   (size_in_bytes / 1024.0 / 1024) AS CurrentSizeMB,
   (max_storage_size_mb) AS MaxSizeMB,
   (size_in_bytes / 1024.0 / 1024.0) * 100.0 / max_storage_size_mb AS PercentUsed
FROM sys.database_query_store_options AS qso
JOIN (SELECT SUM(reserved_space_in_bytes) AS size_in_bytes
    FROM sys.query_store_runtime_stats) AS qs_usage ON 1=1
WHERE max_storage_size_mb IS NOT NULL;
```

These scripts provide:

1. **Standard configuration** for Query Store.
2. **Monitoring and alerting** for storage and performance.
3. **Plan forcing and regression handling**.
4. **Maintenance tasks** like flushing, cleaning, and resetting data.

SQL Server Query Store

SQL Server 2017 – 2025

Monitor, troubleshoot, and optimize queries in SQL Server 2017–2025.

Query Store is the **"Flight Recorder for queries"**

ARCHITECTURE OVERVIEW

- Tracks/runs at **database level**
- Persists query data across restarts (SSMS)

ENABLING QUERY STORE

In **SQL Server Management Studio**

1. Right-click **database** > **Properties**
2. Go to "**Query Store**" tab

```
USE MyDatabase;
ALTER DATABASE MyDatabase
    SET QUERY_STORE = ON;
```

Do NOT enable on **master** or **tempdb**.

CONFIGURE & CAPTURE DATA

#1 Set the right capture **mode:**

- **Operation Mode**
  - Sets max size byte before cleanup
- **INTERVAL_LENGTH** MINUTES
  - Sets data retention period (ms')

Configure &option options:

- **Operation Mode**
- **MAX_STORAGE_SIZE_MB**
- **INTERVAL_LENGTH** MINUTES
- **STALE_QUERY_THRESHOLD** DAYS
- **MAX_PLANS_PER_QUERY**
- **Clear/Flush Data**

MONITOR & FORCE PLANS

- Review top costly & regressed quenes
- Set appropriate storage size, % alerts
- Keep capture thresholds controlled

Alert if **Query Store** reaches **90%** of **max_storage_size_mb**

```
SELECT size_in_bytes /
    MAX(max_storage_size_mb) * 100 AS
    Percontused
FROM sys.database_query_store_options.dbo.
    query_store_runtime_stats
```

Standard Operating Procedure (SOP) for **forcing execution plans using SQL Server Query Store** aligned with **SQL Server versions 2017 through 2025**. This SOP explains the purpose, configuration, steps, and best practices for plan forcing when troubleshooting performance regressions.

Reference: https://www.sqlshack.com/force-query-execution-plan-using-sql-server-2016-query-store/

SOP: Forcing Execution Plans Using SQL Server Query Store

1. Purpose

To provide a standardized method for identifying problematic execution plans and **forcing a known good execution plan** using the SQL Server **Query Store** feature across environments running **SQL Server 2017, 2019, 2022, and 2025**.
Forcing a plan can stabilize query performance when the optimizer chooses a suboptimal plan due to changes in statistics, indexes, schema, or workload patterns.

2. Scope

This procedure applies to all database instances supporting Query Store, including on-premises SQL Server and Azure SQL Managed Instances. Query Store is **enabled by default** in SQL Server 2022 and later but must be enabled manually on earlier versions.

3. Background

- **Query Store** captures query text, execution plans, and runtime statistics persistently so that performance can be monitored over time.
- Plan forcing allows the DBA to instruct the optimizer to use a specific plan captured in Query Store instead of letting the optimizer select a new plan.
- Prior to Query Store, plan guides were required; plan forcing via Query Store simplifies performance stabilization.

4. Prerequisites

1. **Query Store must be enabled** for the target database.
2. You must have **sufficient privileges** (e.g., db_owner) on the database.
3. Ensure Query Store has collected sufficient data (such as multiple executions of the query).
4. Confirm the database compatibility level supports Query Store plan forcing (SQL Server 2016+).

5. Enable Query Store (If Not Already Enabled)

T-SQL

```
ALTER DATABASE [YourDatabase] SET QUERY_STORE = ON;
ALTER DATABASE [YourDatabase] SET QUERY_STORE (OPERATION_MODE = READ_WRITE);
```

6. Identify Candidate Queries for Forcing

1. Use built-in **Query Store reports** in SSMS (e.g., *Regressed Queries*, *Top Resource Consuming Queries*).
2. Identify the **query_id** and **plan_id** for plans associated with performance regressions.
3. Choose the plan with the best performance based on metrics such as duration, CPU, logical reads, etc.

7. Force an Execution Plan

Using SSMS Reports

1. Open the *Regressed Queries* report.
2. Select the query and its associated plan.
3. Click **Force Plan** to enforce that plan for future executions.

Using T-SQL

```
EXEC sp_query_store_force_plan @query_id = <QUERY_ID>, @plan_id = <PLAN_ID>;
```

- Replace <QUERY_ID> and <PLAN_ID> with actual IDs from Query Store.

- After forcing, SQL Server will recompile and use the forced plan going forward.

8. Monitor Forced Plans

1. Verify forced plans by inspecting sys.query_store_plan:

```
SELECT query_id, plan_id, is_forced_plan, force_failure_count
FROM sys.query_store_plan
WHERE is_forced_plan = 1;
```

- is_forced_plan = 1 identifies forced plans, and force_failure_count indicates forcing issues.

https://www.red-gate.com/simple-talk/databases/sql-server/database-administration-sql-server/determining-happens-force-execution-plan-via-query-store/?utm_source=chatgpt.com

2. Monitor runtime statistics to ensure expected performance improvements.

9. Unforce a Plan (If Needed)

If you decide to revert to the optimizer's choice:

```
EXEC sp_query_store_unforce_plan @query_id = <QUERY_ID>, @plan_id = <PLAN_ID>;
```

This removes forcing for that specific plan.

10. Maintenance and Best Practices

1. **Capture Relevant History:** Allow Query Store to collect multiple executions before forcing.
2. **Automatic Plan Correction:** Later versions (SQL Server 2019/2022+) can automatically force plans if enabled via Automatic Tuning features.
3. **Review Regularly:** Periodically review forced plans to verify continued benefit, especially after schema/statistics changes.
4. **Avoid Overuse:** Do not force plans for every query; focus on critical performance regressions.

11. Version-Specific Notes

Version	Notes
SQL Server 2017–2019	Query Store must be manually enabled; plan forcing supported for most queries.
SQL Server 2022+	Query Store is enabled by default; enhanced features like Automatic Plan Correction and optimized plan forcing are available.

12. Risks & Mitigations

- Forcing a plan may prevent the optimizer from adapting to data distribution changes.
  Mitigation: Monitor forced plan effectiveness and unforce when no longer beneficial.
- Query Store storage limits can transition store to read-only.
  Mitigation: Adjust MAX_STORAGE_SIZE_MB and cleanup policies as needed.

13. Documentation & Change Control

- Document the reason for forcing a plan (performance regression root cause).
- Log the change with plan IDs and query context.
- Include in operational runbooks for audit and review.

14. References

- SQLShack: *Force query execution plan using SQL Server Query Store* — fundamentals and steps for forcing plans.
- Microsoft SQL Server Blog: Query Store enabled by default in SQL Server 2022.

This SOP provides clear procedural steps and version-relevant details for using Query Store to force execution plans, ensuring performance stability across SQL Server versions **2017 through 2025**.
If you want, I can also provide **T-SQL templates** to automate detection of regressions and plan forcing actions.

T-SQL automation template for SQL Server 2017–2025 that helps **detect regressed queries and automatically force the best plan** using Query Store. This complements the SOP for operational efficiency.

1. Detect Regressed Queries

```
USE [YourDatabase];
GO
-- List top 10 queries with highest average duration in Query Store
SELECT TOP 10
   q.query_id,
   qt.query_sql_text,
   p.plan_id,
   rs.avg_duration,
   rs.count_executions,
   rs.last_execution_time
FROM sys.query_store_query_text qt
JOIN sys.query_store_query q ON qt.query_text_id = q.query_text_id
JOIN sys.query_store_plan p ON q.query_id = p.query_id
JOIN sys.query_store_runtime_stats rs ON p.plan_id = rs.plan_id
WHERE p.is_forced_plan = 0 -- Only consider non-forced plans
ORDER BY rs.avg_duration DESC;
```


2. Identify the Best Plan

```
-- Best plan based on lowest average duration
SELECT TOP 1
   p.plan_id,
   rs.avg_duration
FROM sys.query_store_plan p
JOIN sys.query_store_runtime_stats rs ON p.plan_id = rs.plan_id
WHERE p.query_id = @QueryID
ORDER BY rs.avg_duration ASC; -- lowest duration = best plan
```

3. Force the Best Plan

```
DECLARE @QueryID INT = 123;   -- Replace with actual query_id
DECLARE @PlanID  INT;
-- Find best plan
SELECT TOP 1 @PlanID = p.plan_id
FROM sys.query_store_plan p
JOIN sys.query_store_runtime_stats rs ON p.plan_id = rs.plan_id
WHERE p.query_id = @QueryID
ORDER BY rs.avg_duration ASC;

-- Force the plan
EXEC sp_query_store_force_plan @query_id = @QueryID, @plan_id = @PlanID;
```

4. Verify Forced Plan

```
SELECT query_id, plan_id, is_forced_plan, force_failure_count
FROM sys.query_store_plan
WHERE query_id = @QueryID;
```

- is_forced_plan = 1 confirms the plan is forced.
- force_failure_count shows any forcing issues.

5. Optional: Unforce a Plan

```
EXEC sp_query_store_unforce_plan @query_id = @QueryID, @plan_id = @PlanID;
```

6. Automation Tips

1. **Wrap in a stored procedure** to scan all critical queries regularly.
2. **Use thresholds**: only force plans if avg_duration exceeds defined SLA limits.
3. **Log changes** to a table for auditing forced plan actions.
4. **Combine with alerts**: notify DBAs when a new regression is detected or a plan is forced.
5. **Cleanup old forced plans** periodically to avoid stale plan usage.

Diagram showing the flow: detect regression → select best plan → force plan → monitor, for SQL Server 2017–2025 to make this SOP visually clear.