Making the Most of Query Store

Jeff Iannucci

Download all slides and scripts

https://github.com/desertdba/Presentations

StraightPath Solutions

A word of thanks to these folks

Conor Cunningham

Erik Darling

Grant Fritchey

Tara Kizer

Kendra Little

Brent Ozar

Paul Randal

Erin Stellato

StraightPath Solutions

Has this been your experience?

Query Store made performance worse

Query Store unexpectedly went READ_ONLY

Forced query plans didn’t always get used

Can we solve any REAL problems with this thing?

YES, WE CAN!!!

“Real world” solutions

Ways to use Query Store beyond forcing plans

Considerations for defaults and “best practices”

Queries and free, community-supported tools

Common problems you can solve

StraightPath Solutions

Setting up Query Store

What defaults to change…maybe

MAX_STORAGE_SIZE_MB

MAX_PLANS_PER_QUERY

STALE_QUERY_THRESHOLD_DAYS

QUERY_CAPTURE_MODE

CUSTOM QUERY_CAPTURE_MODE

Under **QUERY_CAPTURE_POLICY**…

- EXECUTION_COUNT
- TOTAL_EXECUTION_CPU_TIME_MS

Start with high values

What defaults to leave alone

DATA_FLUSH_INTERVAL_SECONDS

INTERVAL_LENGTH_MINUTES

SIZE_BASED_CLEANUP_MODE

WAIT_STATS_CAPTURE_MODE

Trace flags

TF 7752 – allows queries to execute while QS loads into memory (for SQL Server 2016 & 2017 only)

TF 7745 – bypasses writing to disk at shutdown, losing unflushed data (all versions)

dbatools (PowerShell)

Get-DbaDbQueryStoreOption

Set-DbaDbQueryStoreOption

Copy-DbaDbQueryStoreOption

Test-DbaDbQueryStore

You should get a job (for monitoring)

```
/*
Query Store - current status
*/
SELECT
    DB_NAME() as DatabaseName
    , actual_state_desc
    , readonly_reason
    , max_storage_size_mb
    , (max_storage_size_mb - current_storage_size_mb) AS query_store_free_space_mb
    , flush_interval_seconds
    , interval_length_minutes
    , stale_query_threshold_days
    , max_plans_per_query
    , query_capture_mode_desc
    , size_based_cleanup_mode_desc
FROM sys.database_query_store_options
```

StraightPath Solutions

Tools you can use

SSMS built-in reports

Top Resource Consuming Queries

Queries With Forced Plans

Tracked Queries

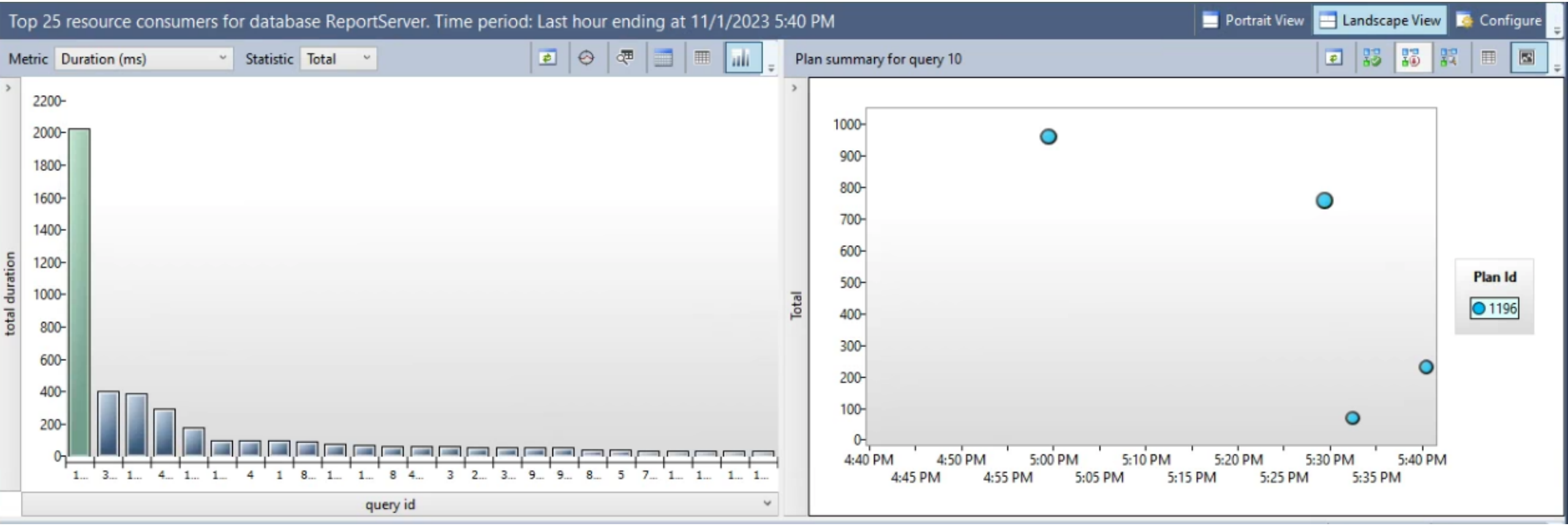
SSMS built-in reports

Top Resource Consuming Queries

Queries With Forced Plans

Tracked Queries

Top Resource Consuming Queries



StraightPath Solutions

Defaults are…

Top 25

Last hour

Total and Duration (2 separate defaults)

Cute and colorful circles (squares and triangles too)

What is the real problem?

CPU usage

Memory consumption

I/O (reads and/or writes)

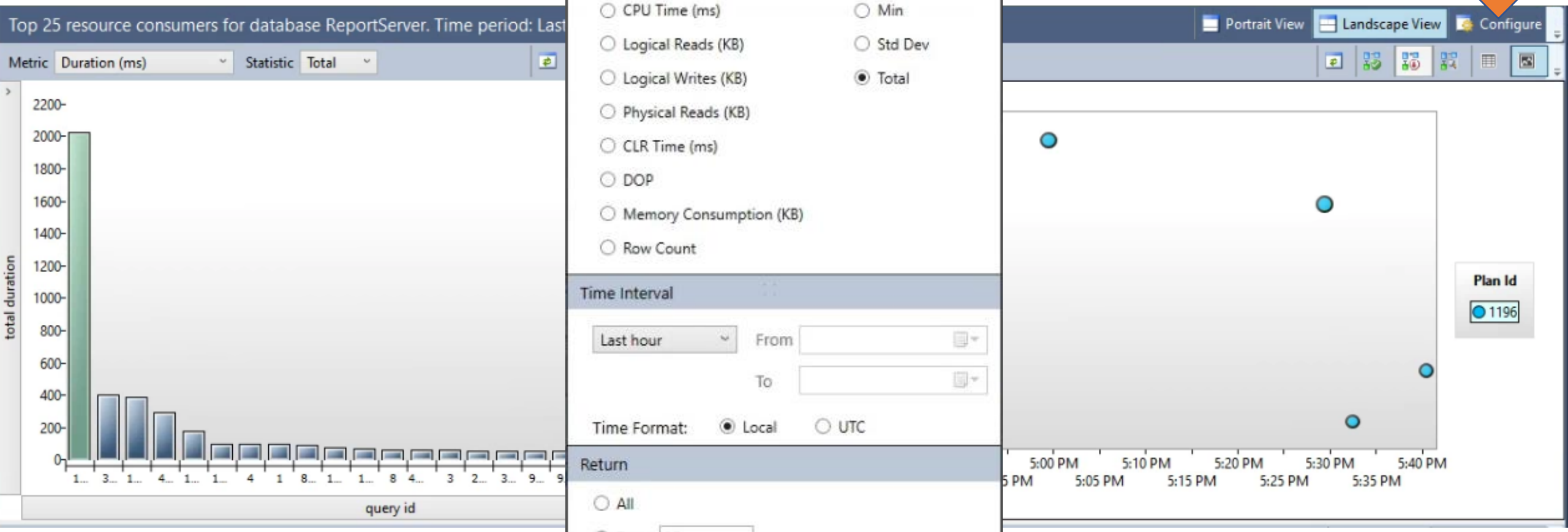
Total, Average, Maximum

Paul Randal's Wait Stats

https://www.sqlskills.com/blogs/paul/wait-statistics-or-please-tell-me-where-it-hurts/

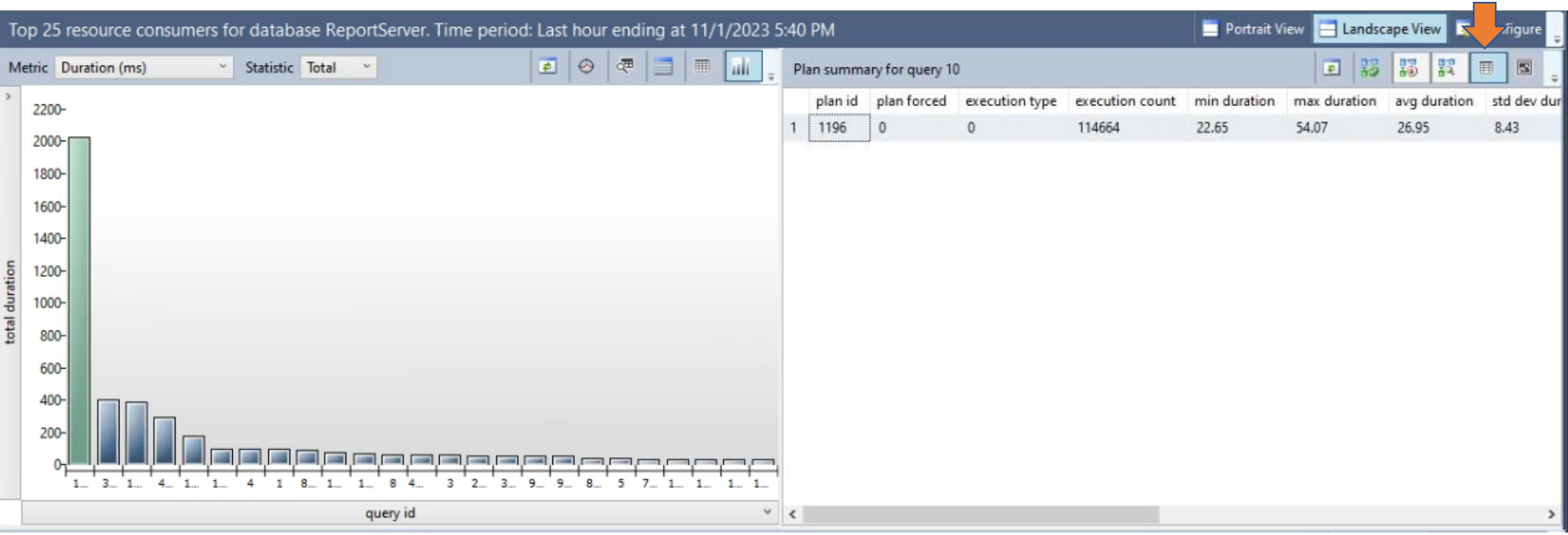
WaitType	Wait_S	Resource_S	Signal_S	WaitCount	Percentage	AvgWait_S	AvgRes_S	AvgSig_S	Help/Info URL
CXPACKET	7668440.60	6793057.13	875383.47	2362131893	65.03	0.0032	0.0029	0.0004	https://www.sqlskills.com/help/waits/CXPACKET
SOS_SCHEDULER_YIELD	2127990.30	1678.10	2126312.20	2777370420	18.05	0.0008	0.0000	0.0008	https://www.sqlskills.com/help/waits/SOS_SCHEDULE...
PAGEIOLATCH_SH	449569.26	421665.76	27903.51	460398999	3.81	0.0010	0.0009	0.0001	https://www.sqlskills.com/help/waits/PAGEIOLATCH_SH
ASYNC_NETWORK_IO	351253.40	328880.93	22372.48	160290986	2.98	0.0022	0.0021	0.0001	https://www.sqlskills.com/help/waits/ASYNC_NETWO...
MSQL_XP	180808.78	180808.78	0.00	4249985	1.53	0.0425	0.0425	0.0000	https://www.sqlskills.com/help/waits/MSQL_XP
WRITELOG	180553.20	130858.81	49694.39	229807177	1.53	0.0008	0.0006	0.0002	https://www.sqlskills.com/help/waits/WRITELOG
OLEDB	137839.73	137839.73	0.00	10707789161	1.17	0.0000	0.0000	0.0000	https://www.sqlskills.com/help/waits/OLEDB
LATCH_EX	94723.08	68904.50	25818.58	316396002	0.80	0.0003	0.0002	0.0001	https://www.sqlskills.com/help/waits/LATCH_EX
PAGEIOLATCH_EX	85873.68	84007.76	1865.92	137039849	0.73	0.0006	0.0006	0.0000	https://www.sqlskills.com/help/waits/PAGEIOLATCH_EX

Top Resource C ies



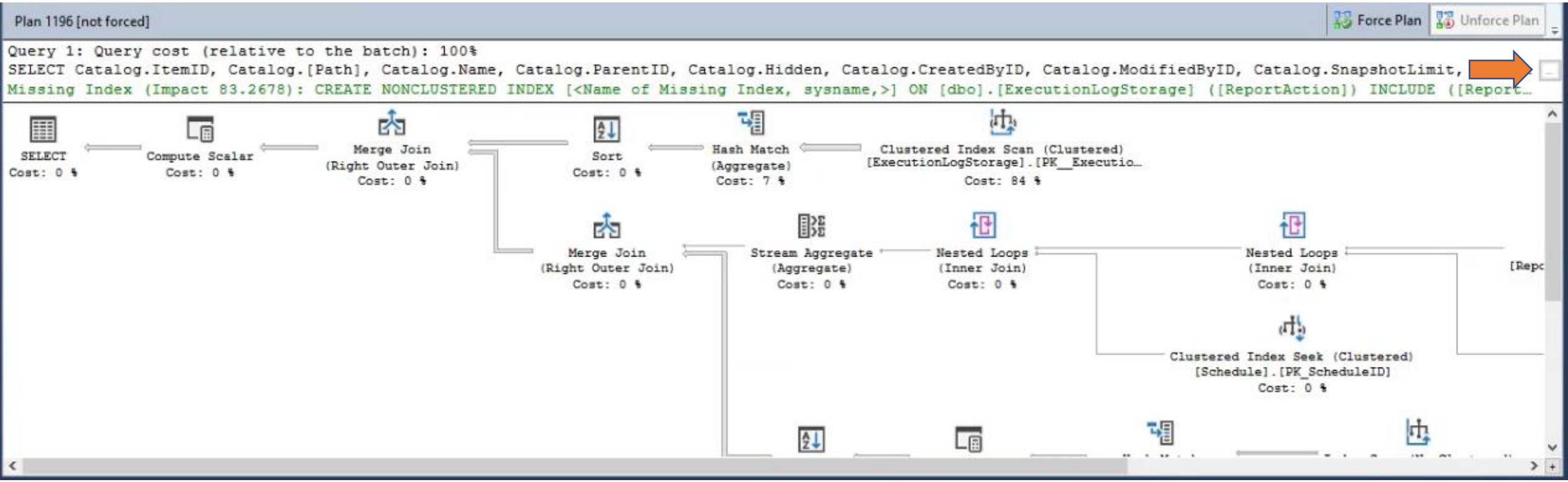
StraightPath Solutions

Top Resource Consuming Queries



StraightPath Solutions

Top Resource Consuming Queries



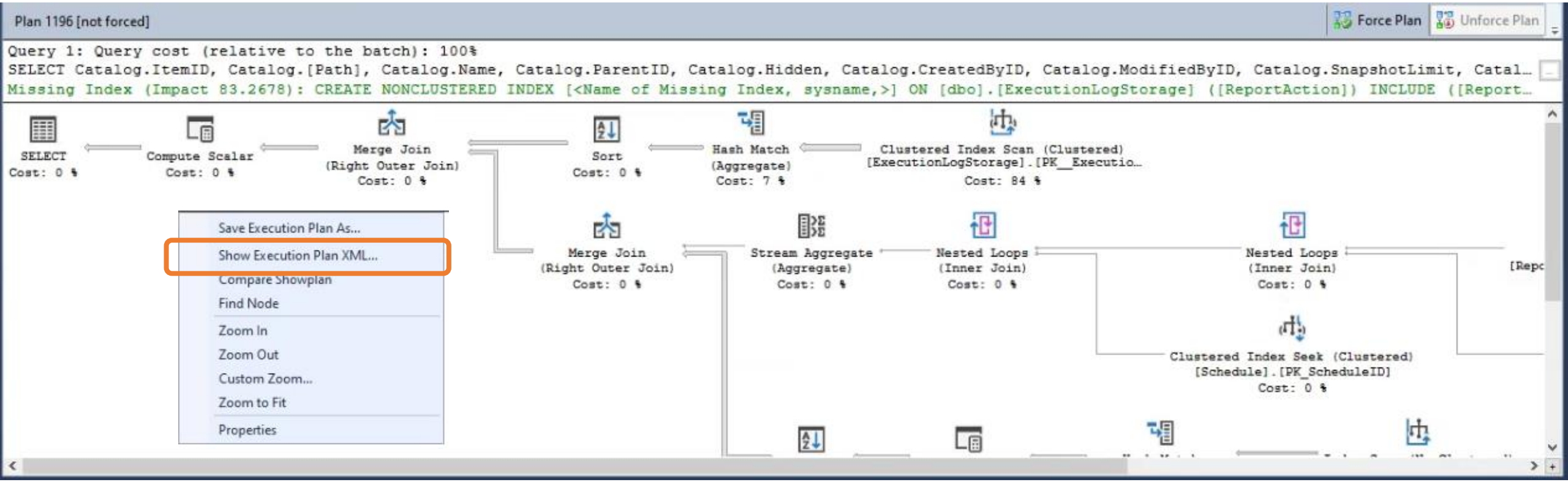
StraightPath Solutions

Top Resource Consuming Queries

```
/*
This query text was retrieved from showplan XML, and may be truncated.
*/

SELECT
        Catalog.ItemID,
        Catalog.[Path],
        Catalog.Name,
        Catalog.ParentID,
        Catalog.Hidden,
        Catalog.CreatedByID,
        Catalog.ModifiedByID,
        Catalog.SnapshotLimit,
        Catalog.PolicyID,
        Catalog.PolicyRoot,
        Catalog.ExecutionFlag,
        Catalog.ExecutionTime,
        Catalog.CreationDate,
        Catalog.ModifiedDate,
        Catalog.[Description],
        (SELECT UserName FROM Users WHERE UserID = Catalog.CreatedByID) CreatedByName,
        (SELECT UserName FROM Users WHERE UserID = Catalog.ModifiedByID) ModifiedByName,
        (SELECT MAX(NextRunTime) FROM  Schedule INNER JOIN ReportSchedule ON
        Schedule.ScheduleID = ReportSchedule.ScheduleID
        WHERE ReportSchedule.ReportID = Catalog.ItemID) NextRunTime,
        (SELECT MAX(TimeStart) FROM ExecutionLog WHERE ExecutionLog.ReportID = Catalog.ItemID) LastExecutionTime
        FROM Catalog Catalog WITH (NOLOCK) WHERE Catalog.Type = 4 OR Catalog.Type = 2
```

StraightPath Solutions

Top Resource Consuming Queries



StraightPath Solutions

Top Resource Consuming Queries

```
                    <Identifier>
                      <ColumnReference Column="@AuthType" />
                    </Identifier>
                  </ScalarOperator>
                </RangeExpressions>
              </Prefix>
            </SeekKeys>
          </SeekPredicateNew>
        </SeekPredicates>
      </IndexScan>
    </RelOp>
  </NestedLoops>
</RelOp>
</ComputeScalar>
</RelOp>
<ParameterList>
  <ColumnReference Column="@AuthType" ParameterDataType="int" ParameterCompiledValue="(1)" />
  <ColumnReference Column="@EditSessionID" ParameterDataType="varchar(32)" ParameterCompiledValue="NULL" />
  <ColumnReference Column="@Path" ParameterDataType="nvarchar(425)" ParameterCompiledValue="N'/CustomerReports/Medallion/ElkayDailyOrderListing'" />
</ParameterList>
</QueryPlan>
</StmtSimple>
</Statements>
</Batch>
</BatchSequence>
</ShowPlanXML>
```

WARNING: This could contain PII

StraightPath Solutions

SSMS built-in reports

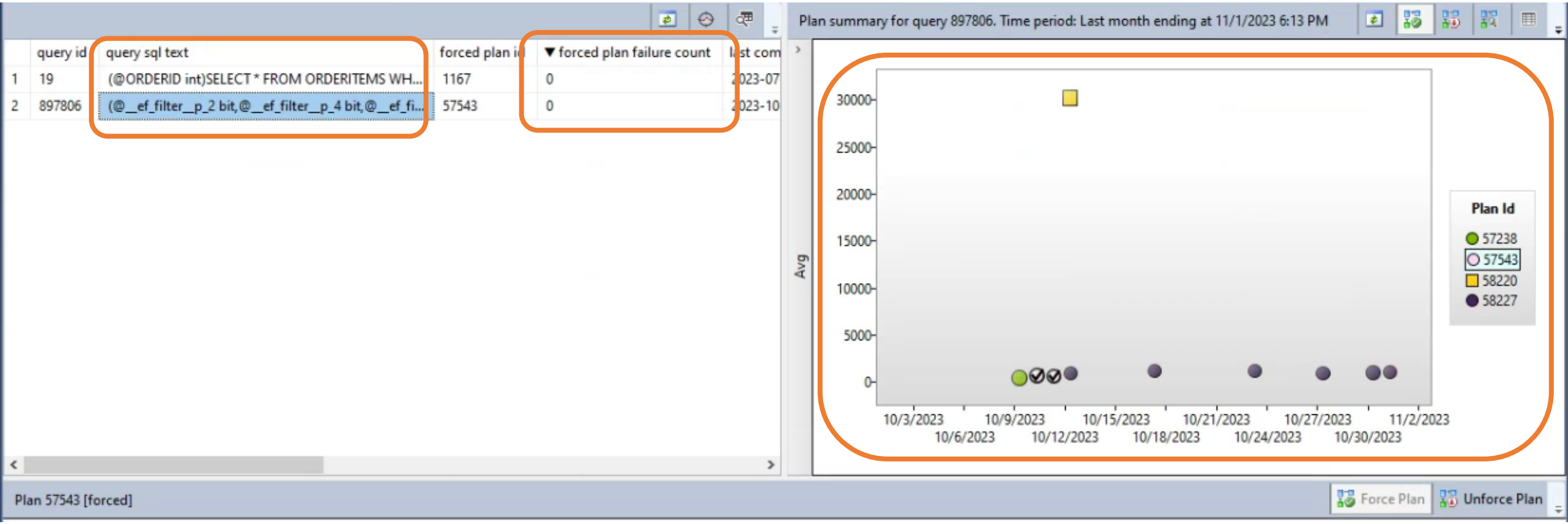
Top Resource Consuming Queries

Queries With Forced Plans

Tracked Queries

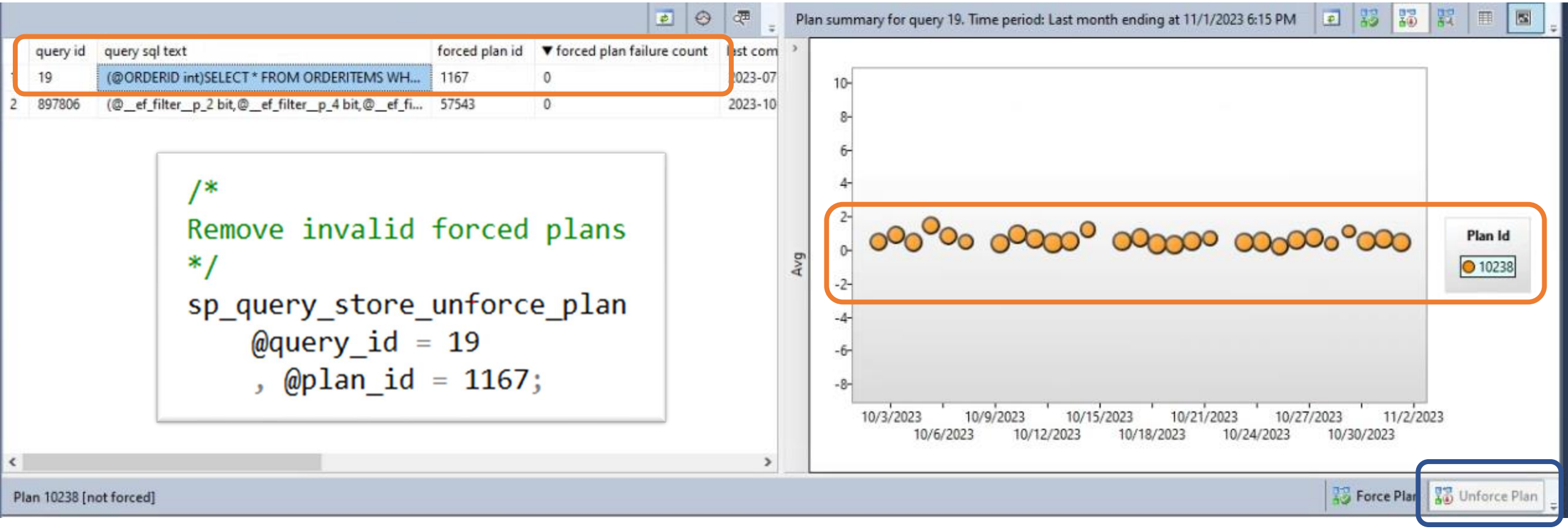
Forced plans are not a long-term solution

Queries With Forced Plans



StraightPath Solutions

Queries With Forced Plans



StraightPath Solutions

Failed forced plans

```
/*
Find failed forced plans
*/
SELECT
    p.query_id
    , p.plan_id
    , q.object_id as containing_object_id
    , p.force_failure_count
    , p.last_force_failure_reason
    , p.last_force_failure_reason_desc
    , p.last_execution_time
FROM sys.query_store_plan AS p
JOIN sys.query_store_query AS q
    ON p.query_id = q.query_id
WHERE p.is_forced_plan = 1
    AND p.force_failure_count > 0;
```

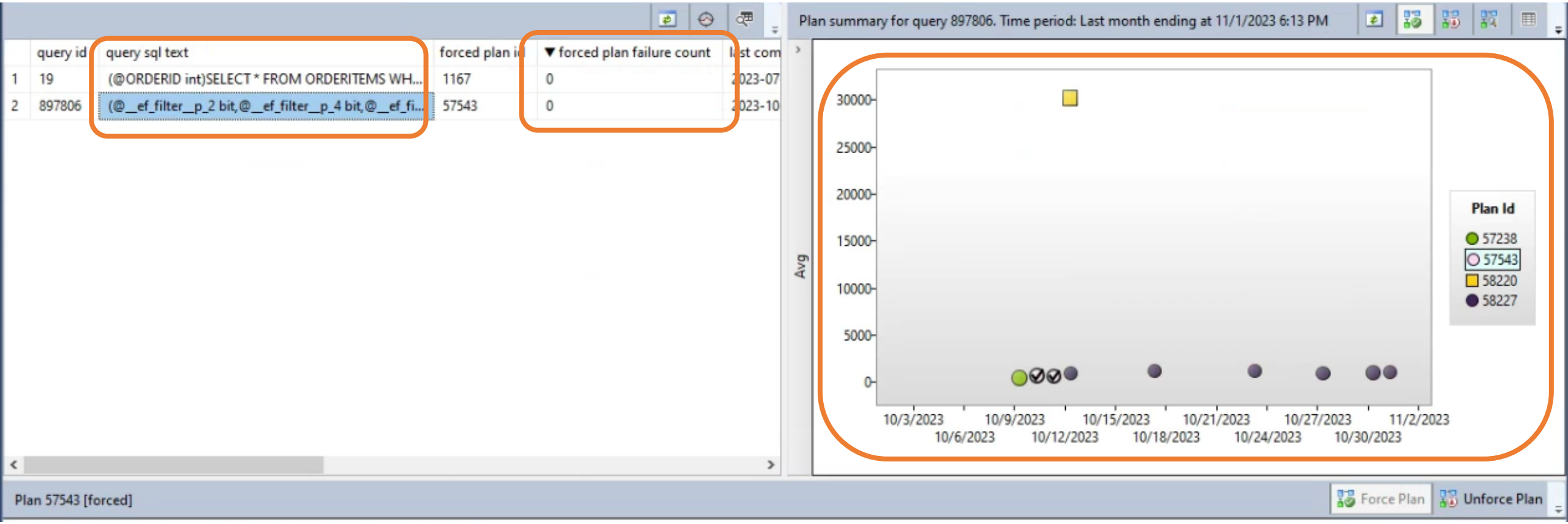
StraightPath Solutions

Failed forced plans

Results | Messages

	query_id	plan_id	containing_object_id	force_failure_count	last_force_failure_reason	last_force_failure_reason_desc	last_execution_time
1	94515	41042	0	1746	8698	NO_PLAN	2023-01-06 21:10:21.9730000 +00:00
2	91452	55877	0	204	8712	NO_INDEX	2023-02-09 20:11:52.1170000 +00:00
3	5149	57907	0	1349	8712	NO_INDEX	2023-02-21 16:07:42.7870000 +00:00
4	5410	91409	0	593	8698	NO_PLAN	2023-02-16 19:22:34.4100000 +00:00
5	12	9117995	1216540659	10	8695	GENERAL_FAILURE	2023-07-27 15:57:14.7330000 +00:00
6	17	9118009	1216540659	6	8695	GENERAL_FAILURE	2023-07-27 15:52:17.2800000 +00:00
7	23908674	9236540	1216540659	2184	8695	GENERAL_FAILURE	2023-10-28 16:52:10.7700000 +00:00
8	23908673	9534016	1216540659	287	8695	GENERAL_FAILURE	2023-10-30 12:07:16.6700000 +00:00
9	23908676	9534028	1216540659	277	8695	GENERAL_FAILURE	2023-10-30 12:07:15.9100000 +00:00

But…what's the deal here?



StraightPath Solutions

“Morally equivalent plans”

Query Store has some flexibility

Not a bug, but a feature

Don’t show as failures

Forced plans are not a long-term solution

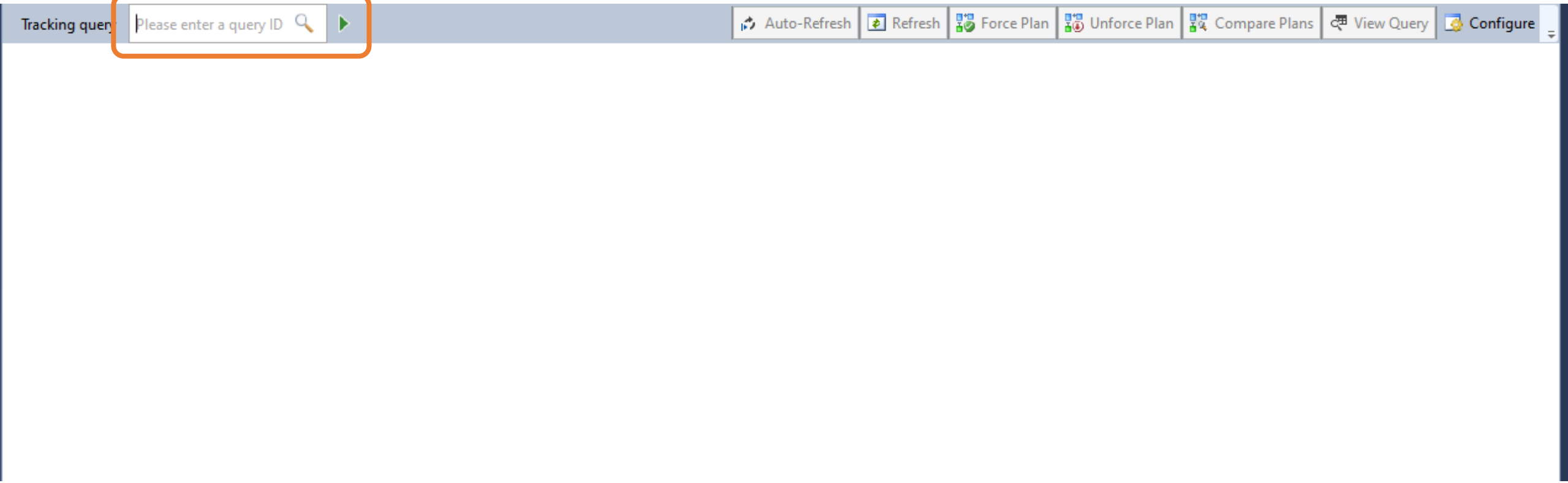
SSMS built-in reports

Top Resource Consuming Queries

Queries With Forced Plans

Tracked Queries

Tracked Queries



StraightPath Solutions

Tracked Queries

Shows performance of one specific **query_id**

Can compare plans

Use “Auto-Update” to track in real time

Find queries (query_id) by…

Object name (stored procedure/function/trigger)

String of characters

Time of execution

…by object name

```
/*
Find query_id by object name (stored procedure, function, trigger, etc.)
*/
SELECT q.query_id
    , t.query_sql_text
FROM sys.query_store_query AS q
JOIN sys.query_store_query_text AS t
    ON q.query_text_id = t.query_text_id
WHERE q.object_id = OBJECT_ID('zzz');
```

StraightPath Solutions

…by string

```
/*
Find query_id by string
*/
SELECT q.query_id
    , t.query_sql_text
FROM sys.query_store_query AS q
JOIN sys.query_store_query_text AS t
    ON q.query_text_id = t.query_text_id
WHERE t.query_sql_text LIKE '%zzz%';
```

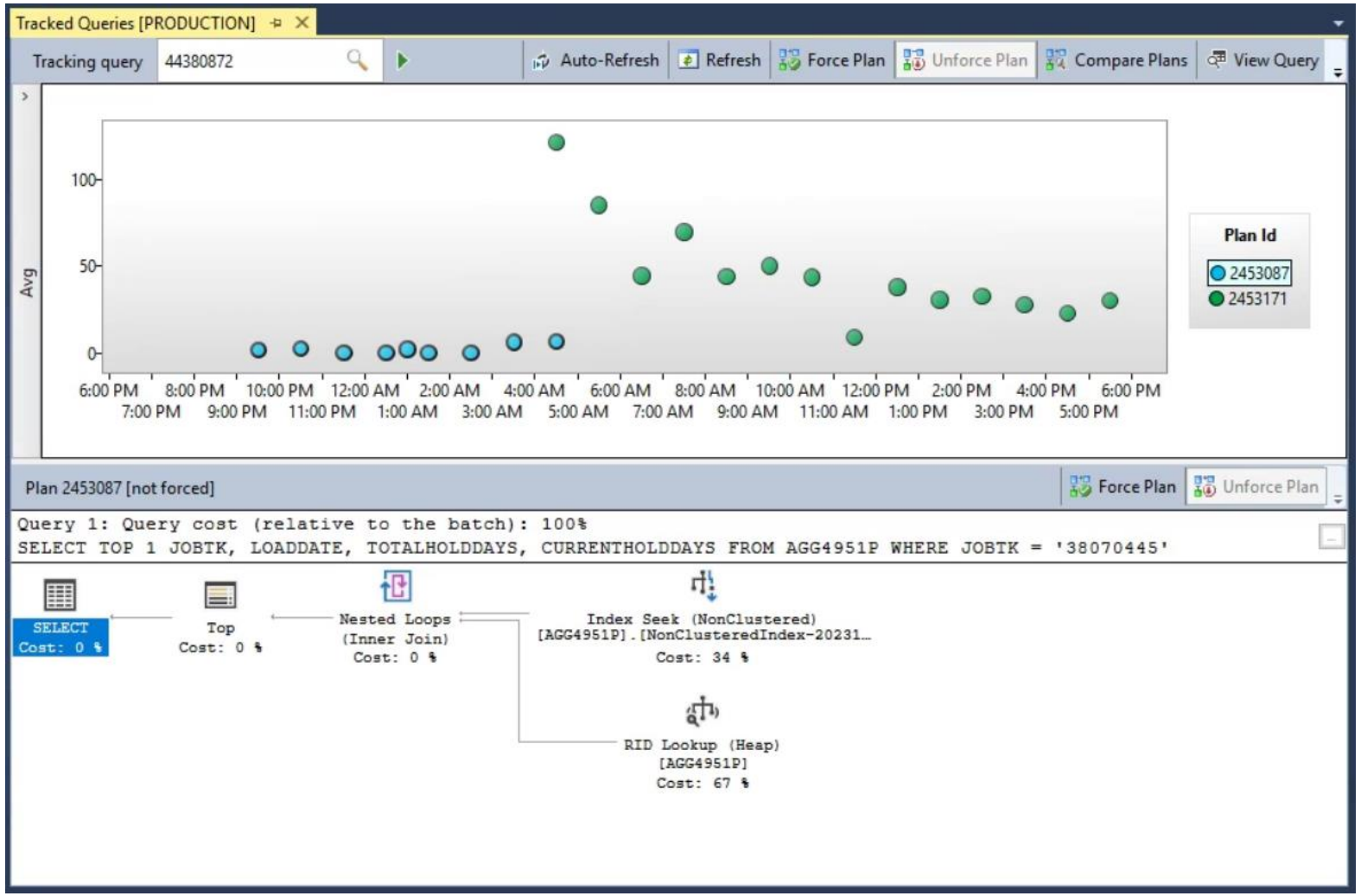
StraightPath Solutions

…by time of execution

```
/*
Find query_id by time of execution
*/
SELECT q.query_id
    , t.query_sql_text
FROM sys.query_store_query AS q
JOIN sys.query_store_query_text AS t
    ON q.query_text_id = t.query_text_id
WHERE q.last_execution_time BETWEEN '1900/01/30 23:00:00' AND '1900/01/31 01:00:00';
```

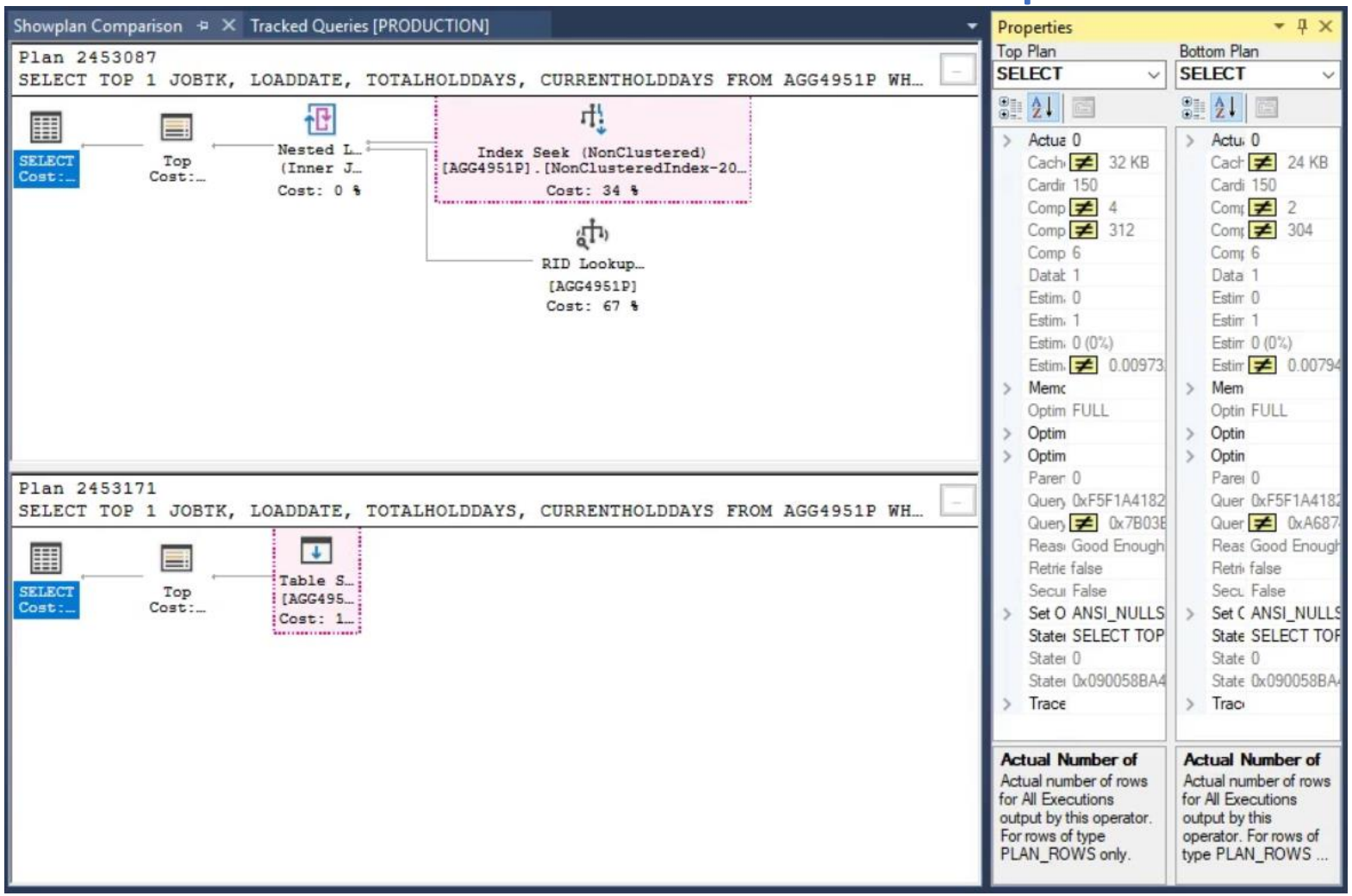
StraightPath Solutions

Tracked Queries



StraightPath Solutions

Tracked Queries - Comparison



StraightPath Solutions

sp_BlitzQueryStore

Prioritized listing of potential issues

Grouped by findings (“Worst Avg CPU”, etc..)

Better if you don’t know where to start

https://github.com/BrentOzarULTD/SQL-Server-First-Responder-Kit

sp_QuickieStore

Choose a metric (CPU, reads, writes, memory…)

Returns query text by default

Faster if you know what you're looking for

https://github.com/erikdarlingdata/DarlingData/tree/main/sp_QuickieStore

Common problems you can solve

Testing index changes

Missing index requests shown in plans

Review existing indexes before adding

Unforce plans if adding/modifying indexes

Reducing upgrade pains

Upgrade to higher version of SQL Server

…but don’t raise the Compatibility Level

If performance is stable, try raising the compatibility level

Use Query Store for analysis and regression fixes

Deadlocks

You can capture deadlocks with extended events

Query Store can give you more info

Use sys.query_store_runtime_stats

WHERE execution_type = 4

What do those shapes mean?

Successful query execution (0)

Aborted query (3)

Error during execution (4)

StraightPath Solutions

Deadlocks

```
/*
Find deadlocks (and other aborted/cancelled queries)
*/
SELECT
    q.query_id
    , t.query_sql_text
    , r.execution_type
    , r.execution_type_desc
    , x.query_plan_xml
    , r.count_executions
    , r.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_query_text t
    ON q.query_text_id=t.query_text_id
OUTER APPLY (SELECT TRY_CONVERT(XML, p.query_plan) AS query_plan_xml) x
JOIN sys.query_store_runtime_stats r
    ON p.plan_id = r.plan_id
WHERE r.execution_type = 4 /* Exception aborted execution */
    AND q.last_execution_time > GETDATE() - 1;
```

Deadlocks

WARNING: this data is still aggregated

Compare deadlocks using **sql_text**

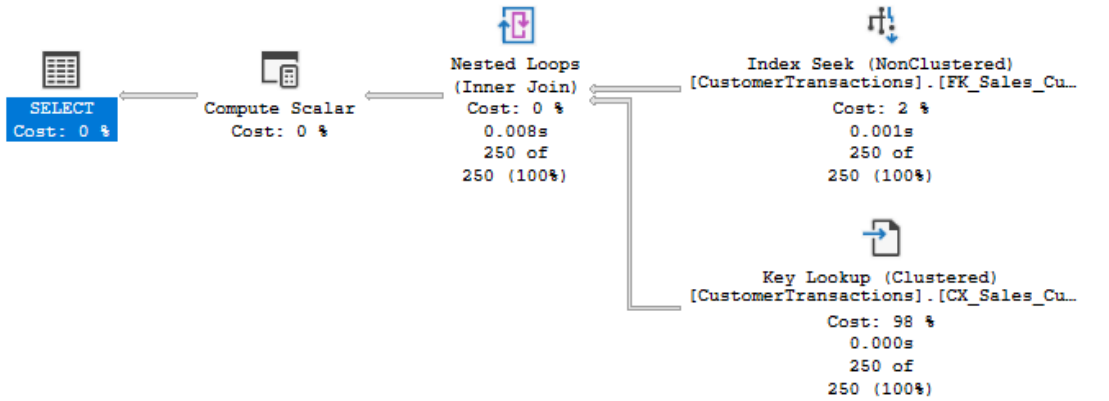
Review execution plans for objects locked in deadlocks

Use **Tracked Queries** report

Parameter Sniffing

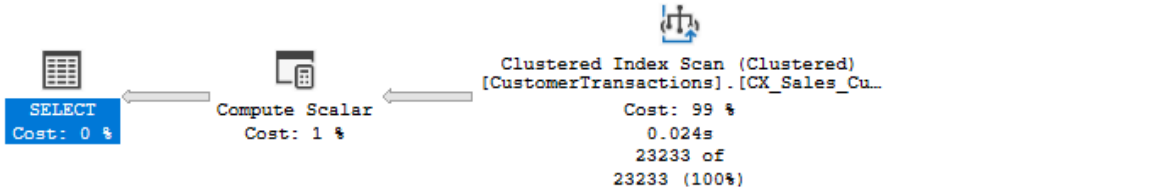
```
EXEC GetTransactionByCustomer
@CustomerID = 401
```

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM [Sales].[CustomerTransactions] WHERE CustomerID = @CustomerID
```




```
EXEC GetTransactionByCustomer
@CustomerID = 976
```

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM [Sales].[CustomerTransactions] WHERE CustomerID = @CustomerID
Missing Index (Impact 99.3199): CREATE NONCLUSTERED INDEX [<Name of Missing Index,
```



Different plans, different performance

StraightPath Solutions

What about FORCE_LAST_GOOD_PLAN?

Automatically force a better plan when found

Can still choose a “bad” plan

Can take hours to automatically revert

Parameter sniffing

Query sys.dm_db_tuning_recommendations*

Insert JSON data from Details column into a table

Create a job that queries the table

Set job to “fail” when regressions occur

*Enterprise Edition only

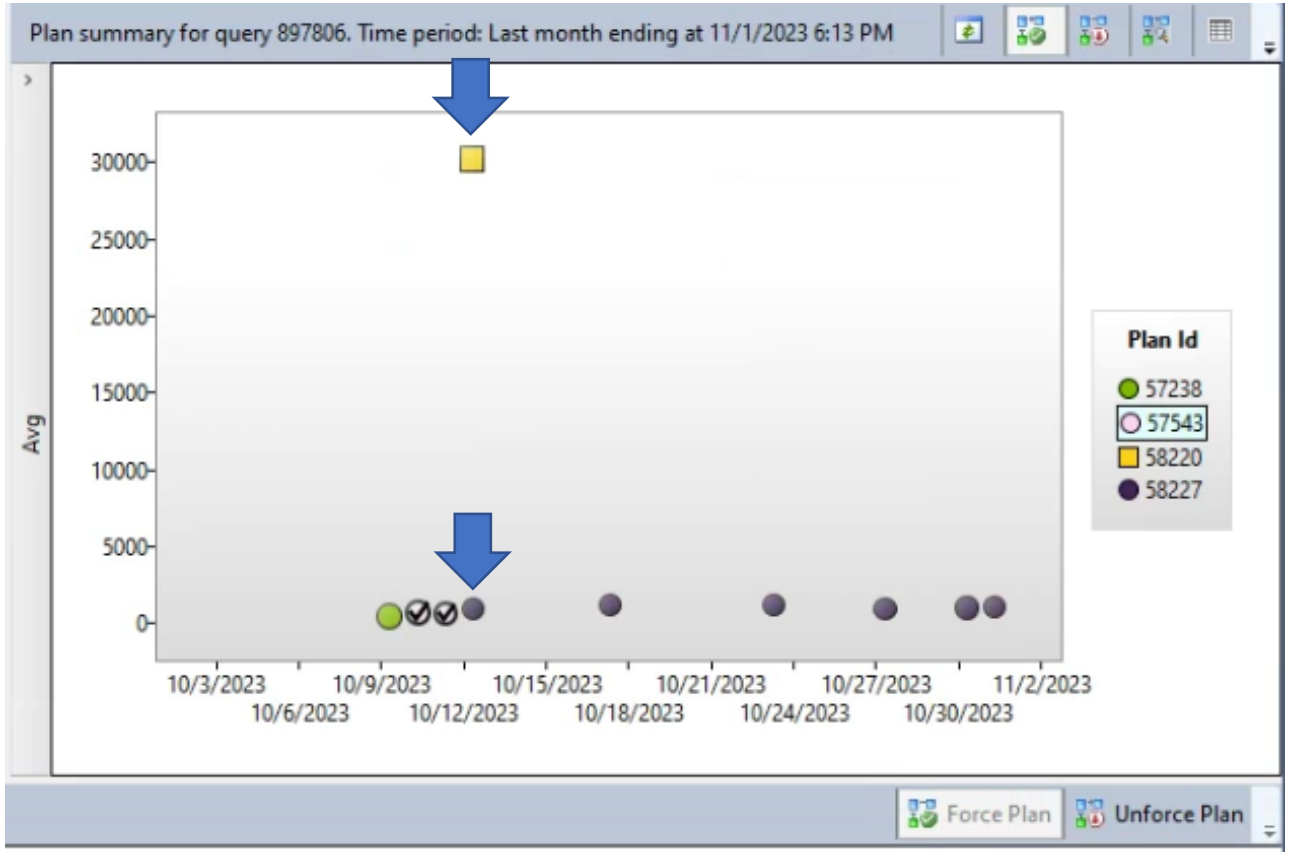
StraightPath Solutions

```
 DECLARE @current_date datetime = GETDATE(), @cmd nvarchar(max), @query nvarchar(1000), @database sysname;

 SET @database = 'DB_Administration'
 SET @cmd = '

 INSERT INTO ' + @database + '.dbo.AutomaticTuning
 SELECT *
 FROM (
     SELECT ''' + CONVERT(varchar(25), @current_date, 121) + ''' AS created_date, DB_NAME() AS database_name, planForceDetails.query_id,
         (planForceDetails.regressedPlanExecutionCount + planForceDetails.recommendedPlanExecutionCount)
                     * (planForceDetails.regressedPlanCpuTimeAverage - planForceDetails.recommendedPlanCpuTimeAverage)/1000000 as estimated_gain,
         TR.reason, TR.score, JSON_VALUE(details, ''$.implementationDetails.script'') as scipt,
         planForceDetails.regressedPlanId, planForceDetails.recommendedPlanId,
         planForceDetails.regressedPlanCpuTimeAverage, planForceDetails.recommendedPlanCpuTimeAverage,
         planForceDetails.regressedPlanExecutionCount, planForceDetails.recommendedPlanExecutionCount
     FROM sys.dm_db_tuning_recommendations AS TR
     CROSS APPLY OPENJSON (Details, ''$.planForceDetails'')
         WITH (  [query_id] int ''$.queryId'',
                 regressedPlanId int ''$.regressedPlanId'',
                 recommendedPlanId int ''$.recommendedPlanId'',
                 regressedPlanErrorCount int,
                 recommendedPlanErrorCount int,
                 regressedPlanExecutionCount int,
                 regressedPlanCpuTimeAverage float,
                 recommendedPlanExecutionCount int,
                 recommendedPlanCpuTimeAverage float
                 ) AS planForceDetails
     LEFT JOIN sys.query_store_query AS Q ON planForceDetails.query_id = Q.query_id
     JOIN sys.query_store_query_text as QT ON Q.query_text_id = QT.query_text_id
     LEFT JOIN sys.query_store_plan AS regressedQP ON planForceDetails.regressedPlanId = regressedQP.plan_id
     LEFT JOIN sys.query_store_plan AS recQP ON planForceDetails.recommendedPlanId =  recQP.plan_id
     WHERE JSON_VALUE(state, ''$.currentValue'') = ''Active''
         AND planForceDetails.regressedPlanCpuTimeAverage/1000.0 >= 500 --bad plan uses at least 500 milliseconds for CPU time
         AND planForceDetails.regressedPlanCpuTimeAverage/planForceDetails.recommendedPlanCpuTimeAverage >= 10 --bad plan is at least 10 times slower than the good plan
         AND planForceDetails.recommendedPlanExecutionCount >= 500 --good plan has at least 500 executions
         --AND planForceDetails.regressedPlanExecutionCount >= 200 --bad plan has at least 200 executions
         AND TR.reason <> ''Number of errors in the regressed plan is greater than in the recommended plan''
 ) t
 WHERE estimated_gain >= 1000
 ORDER BY estimated_gain desc, score desc;
 ';

 EXEC sp_ineachdb @cmd, @user_only = 1, @exclude_list = @database;
```

```
39.      LEFT JOIN sys.query_store_query AS Q ON planForceDetails.query_id = Q.query_id
40.      JOIN sys.query_store_query_text as QT ON Q.query_text_id = QT.query_text_id
41.      LEFT JOIN sys.query_store_plan AS regressedQP ON planForceDetails.regressedPlanId =
regressedQP.plan_id
42.      LEFT JOIN sys.query_store_plan AS recQP ON planForceDetails.recommendedPlanId =  recQP.plan_id
43.      WHERE JSON_VALUE(state, ''$.currentValue'') = ''Active''
44.          AND planForceDetails.regressedPlanCpuTimeAverage/1000.0 >= 500 --bad plan uses at least 500
milliseconds for CPU time
45.          AND
planForceDetails.regressedPlanCpuTimeAverage/planForceDetails.recommendedPlanCpuTimeAverage >= 10 --
bad plan is at least 10 times slower than the good plan
46.          AND planForceDetails.recommendedPlanExecutionCount >= 500 --good plan has at least 500
executions
47.          --AND planForceDetails.regressedPlanExecutionCount >= 200 --bad plan has at least 200
executions
48.          AND TR.reason <> ''Number of errors in the regressed plan is greater than in the recommended
plan''
49.   ) t
50.   WHERE estimated_gain >= 1000
51.   ORDER BY estimated_gain desc, score desc;
52.   ';
```

StraightPath Solutions

What happens next?



https://rb.gy/p1bme2

StraightPath Solutions

Let’s review what we have learned

Summary – Setting up Query Store

Which defaults to change (maybe)

…and which to leave alone

Trace flags

Set up a monitoring job

Summary – Tools you can use

SSMS Built-in Reports

Forced plans…which are not a long-term solution

T-SQL queries to find Query IDs

sp_BlitzQueryStore, sp_QuickieStore

Summary – Problems you can solve

Testing index changes

Reducing pain from SQL Server upgrades

Deadlocks

Parameter sniffing

Thank you!

github.com/desertdba/Presentations

desertdba.com

jeff@desertdba.com

@desertdba

www.linkedin.com/in/jeff-iannucci

StraightPath Solutions