**POC on Log the History/Present Query I/O and CPU Cost of Database**

We are going to implement the inbuilt mechanism of Microsoft SQL Server feature (Monitoring performance by using the Query Store) as Oracle Automatic Workload Repository (AWR) which helps to collect statistic/snapshot in specific interval of Database.

**Introduction:**

The Monitoring performance by using the Query Store is stands for Query Store automatically captures a history of queries, plans, and runtime statistics, I/O and CPU Cost to collect statistic/snapshot in specific interval of Database.

**Objectives:**

* Database performance tuning based on history query I/O and CPU Cost
* Database activity monitoring
* Used for reporting purpose
* Track the used Query in Database

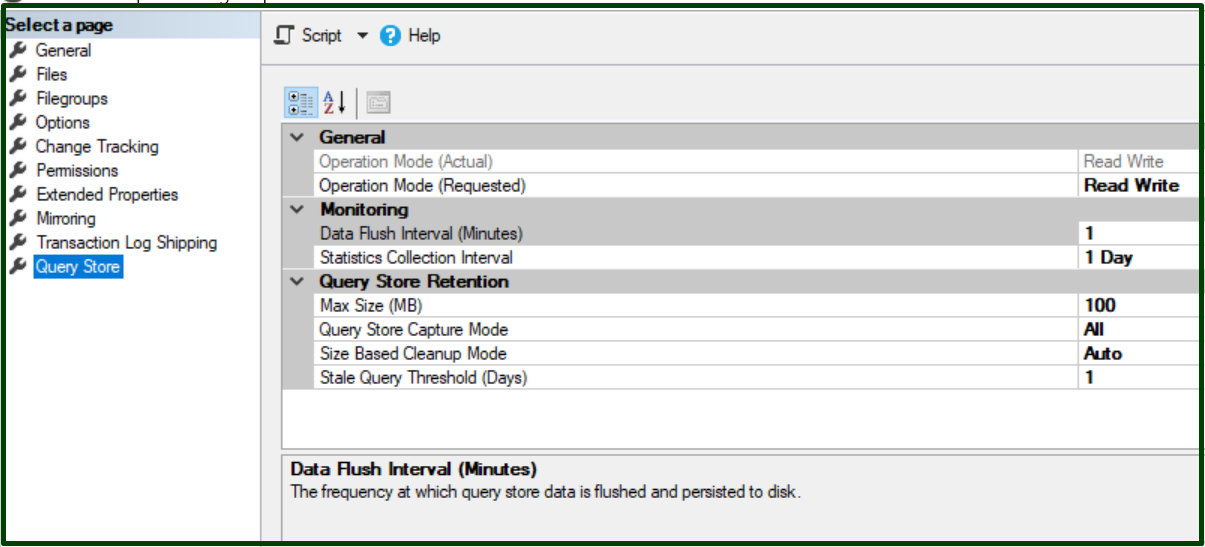
**Database Configurations:**

1. **Enabling the Query Store**

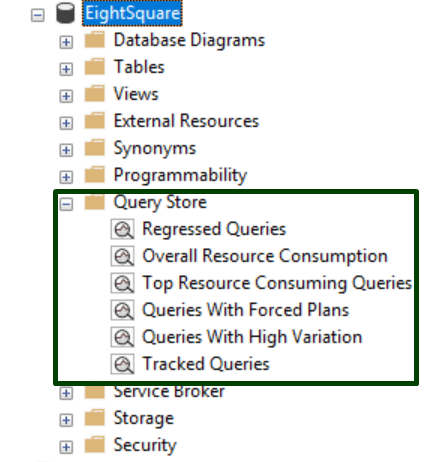
Query Store is not active for new databases by default.

**Use the Query Store Page in SQL Server Management Studio**

* In Object Explorer, right-click a database <<DatabaseName>>, and then click **Properties**.
* In the **Database Properties** dialog box, select the **Query Store** page.
* In the **Operation Mode (Requested)** box, select **Read Write**.
* In the **Data Flush Interval (Minutes)** put **1**
* In the **Statics Collection Interval** put **1 Day**
* In the **Max Size (MB) 100**
* In the **Stale Query Threshold (Days)** put **1**



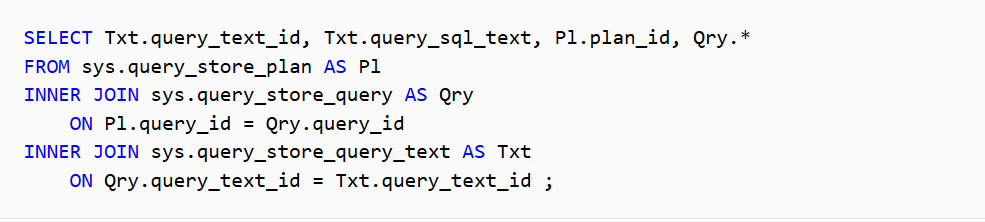
* To verify in Database <<DatabaseName>>



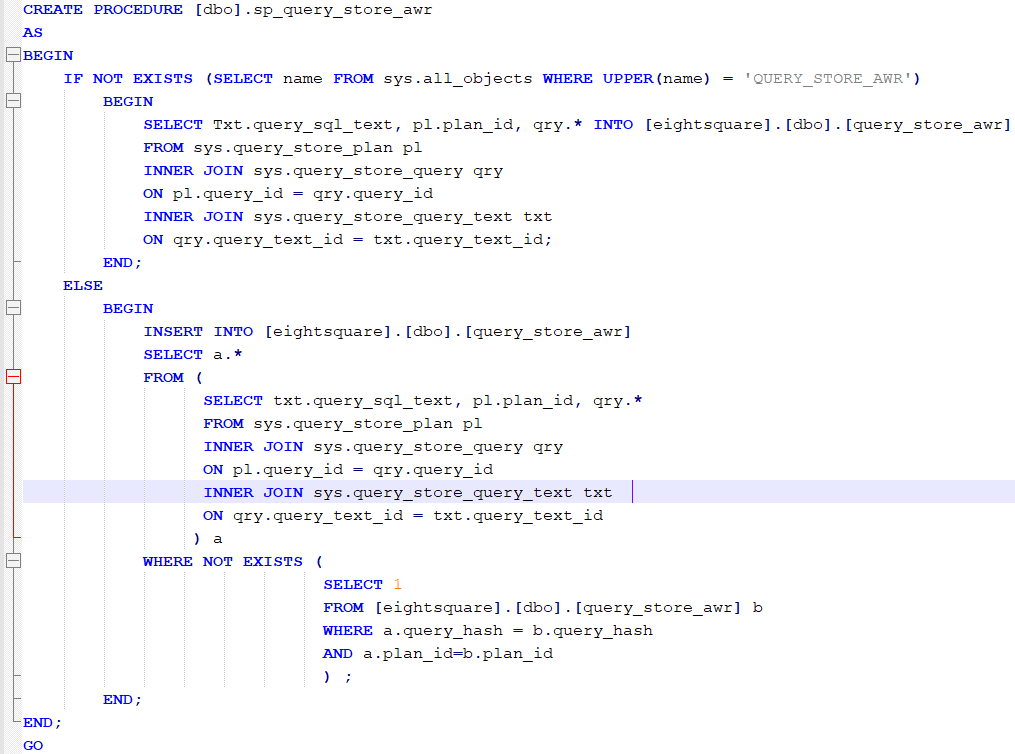
1. **Write a Procedure** to put thescript output (captures a history of queries, plans, and runtime statistics, I/O and CPU Cost to collect statistic/snapshot in specific interval of Database) to store into another database table.

The table data has been truncate it after every request to save the server storage.

* The Script puts inside the Procedure to fetch the outcomes:



* The Stored Procedure Script

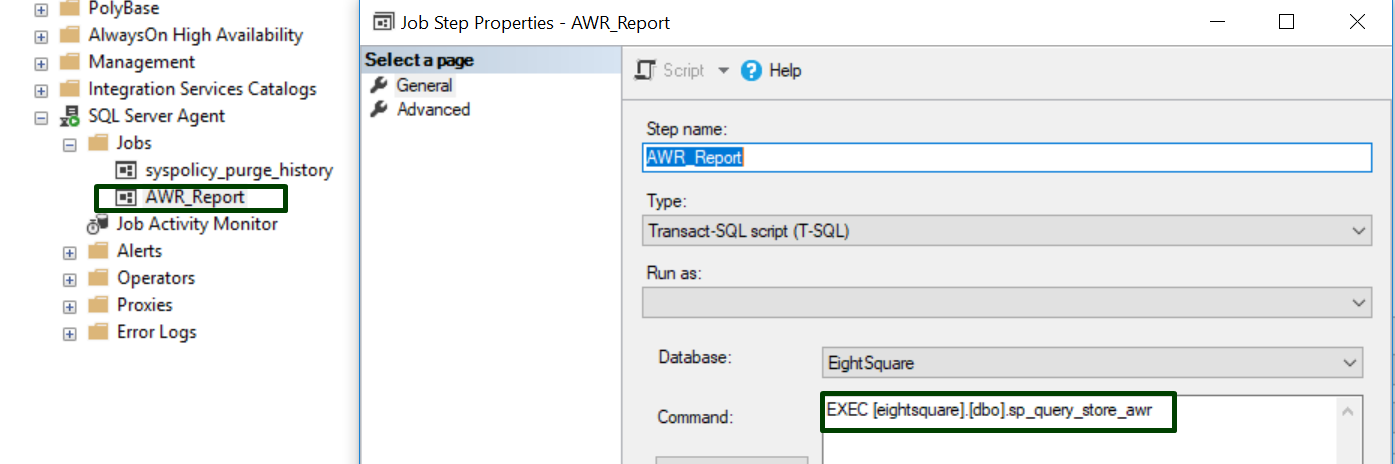


1. **Procedure** should be scheduled before daily backup starts.

If we place scheduled before daily backup then backup doesn’t have to store the query result from **Query Storage Page.**

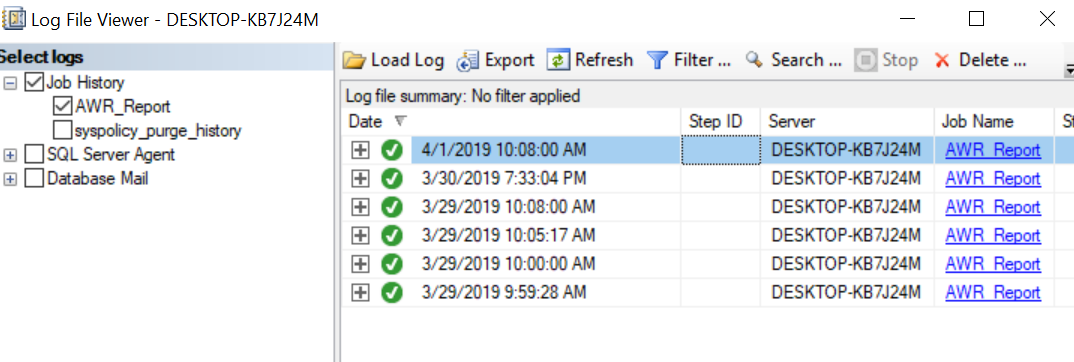
* The script put inside the scheduler





**Monitoring:**

Check the status/log of Scheduler job

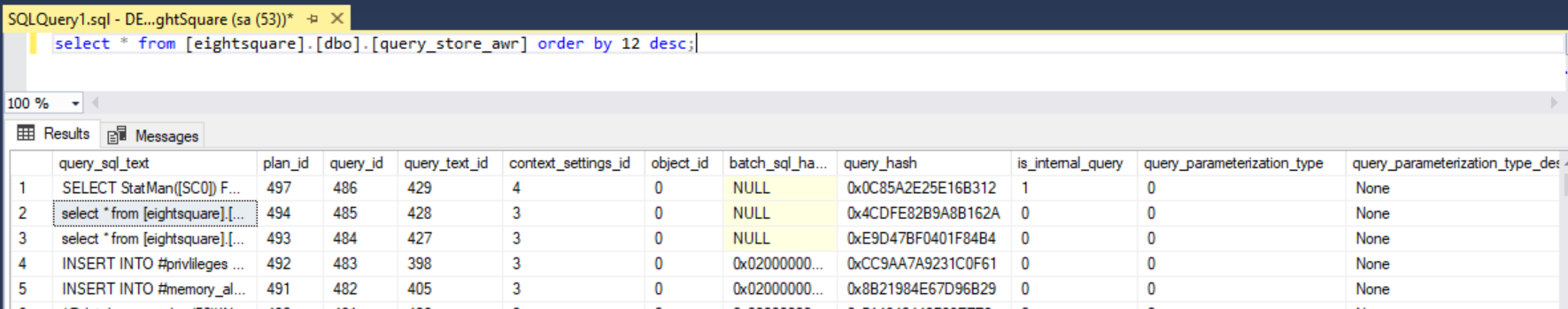


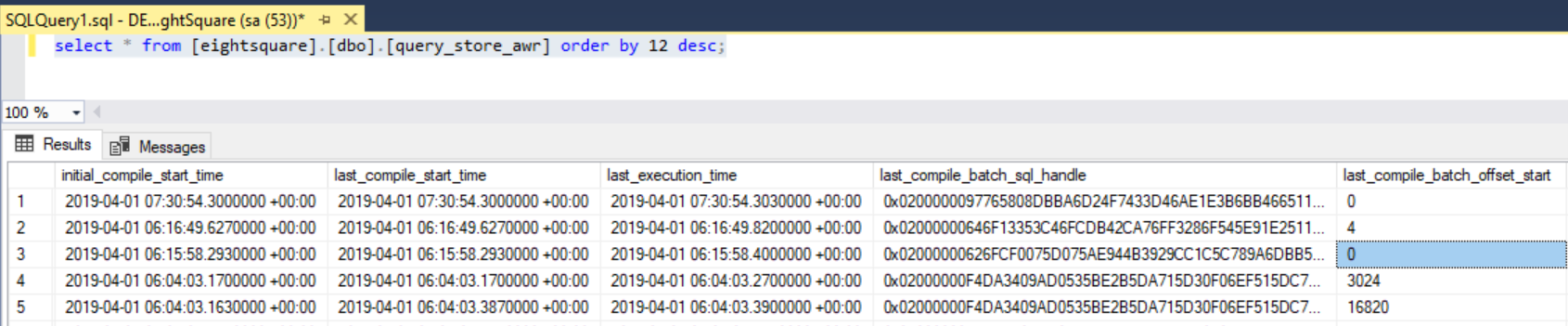
**Outcomes:**

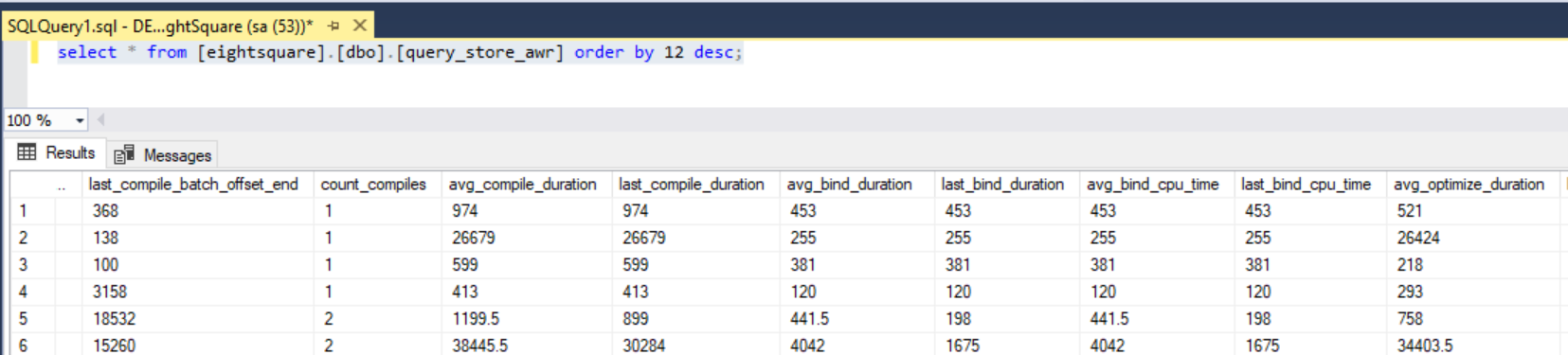
The data from table <<[eightsquare].[dbo].[query\_store\_awr]>> produced by scheduler job.

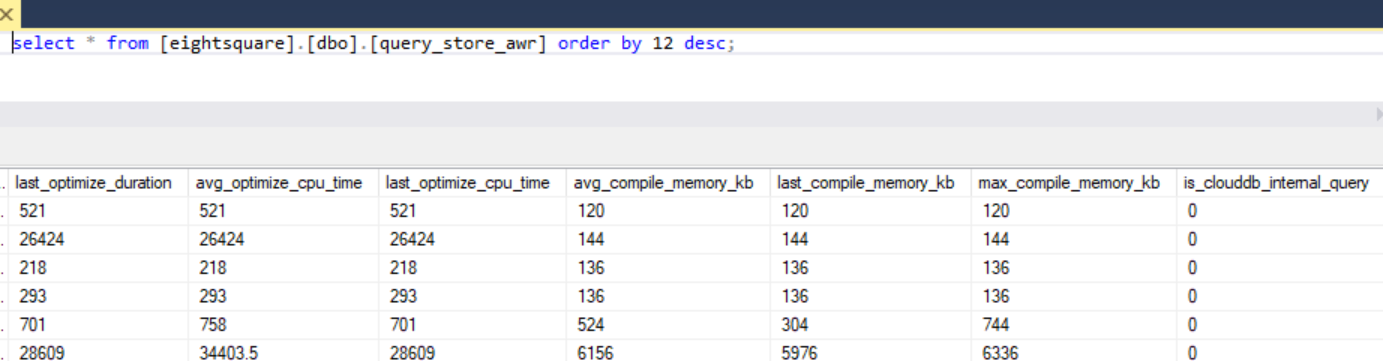
The table wounded the column to store the information:

|  |  |
| --- | --- |
| **Column Name** | **Remarks** |
| query\_sql\_text | Executed Query (DML) |
| plan\_id | Executed Query Plan ID |
| query\_id | Primary key. |
| query\_text\_id | Foreign key. |
| context\_settings\_id | Primary key. This value is exposed in Showplan XML for queries |
| object\_id | ID of the database object that the query is part of (stored procedure, trigger,etc.) |
| batch\_sql\_handle | ID of the statement batch the query is part of. Populated only if query references temporary tables or table variables. |
| query\_hash | MD5 hash of the individual query, based on the logical query tree. Includes optimizer hints. |
| is\_internal\_query | The query was generated internally. |
| query\_parameterization\_type | Kind of parameterization: 0 – None/1 – User/2 – Simple/3 - Forced |
| query\_parameterization\_type\_desc | Textual description for the parameterization type. |
| initial\_compile\_start\_time | Compile start time. |
| last\_compile\_start\_time | Compile start time. |
| last\_execution\_time | Last execution time refers to the last end time of the query/plan. |
| last\_compile\_batch\_sql\_handle | Handle of the last SQL batch in which query was used last time |
| last\_compile\_batch\_offset\_start | Information that can be provided along with last\_compile\_batch\_sql\_handle. |
| last\_compile\_batch\_offset\_end | Information that can be provided along with last\_compile\_batch\_sql\_handle. |
| count\_compiles | Compilation statistics. |
| avg\_compile\_duration | Compilation statistics in microseconds. |
| last\_compile\_duration | Compilation statistics in microseconds. |
| avg\_bind\_duration | Binding statistics in microseconds. |
| last\_bind\_duration | Binding statistics. |
| avg\_bind\_cpu\_time | Binding statistics. |
| last\_bind\_cpu\_time | Binding statistics. |
| avg\_optimize\_duration | Optimization statistics in microseconds. |
| last\_optimize\_duration | Optimization statistics. |
| avg\_optimize\_cpu\_time | Optimization statistics in microseconds. |
| last\_optimize\_cpu\_time | Optimization statistics. |
| avg\_compile\_memory\_kb | Compile memory statistics. |
| last\_compile\_memory\_kb | Compile memory statistics. |
| max\_compile\_memory\_kb | Compile memory statistics. |
| is\_clouddb\_internal\_query | Always 0 in SQL Server on-premises. |

****







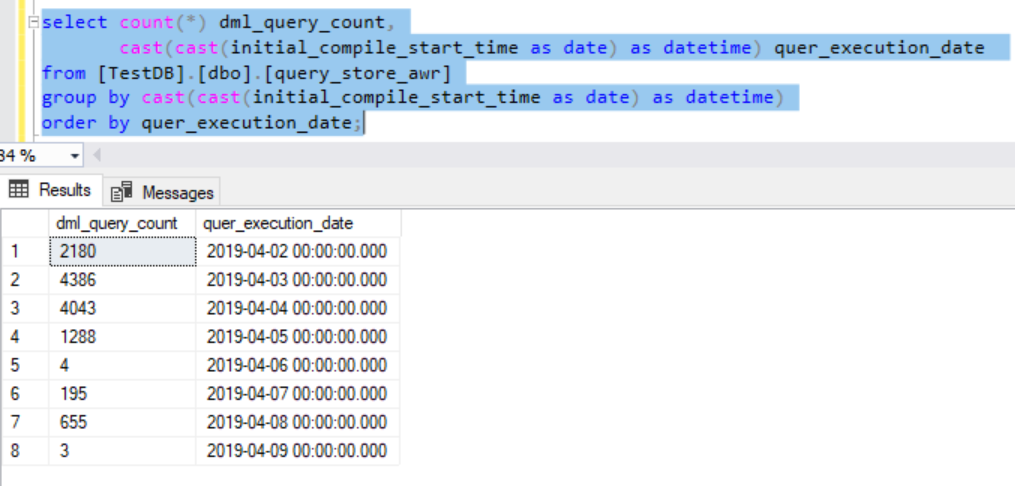
**Conclusion:**

We can store per days queries hit in the database with their CPU cost and IO cost. Which will definitely helps us to find the long running queries in the database and max resources using queries. For which we can point out the long queries and work on them for the best optimization plan. Moreover, we can have a report of everyday DML QUERIES (SELECT, INSERT, UPDATE, DELETE, MERGE) on database which is not possible to store without Query Store mechanism of SQL Server.

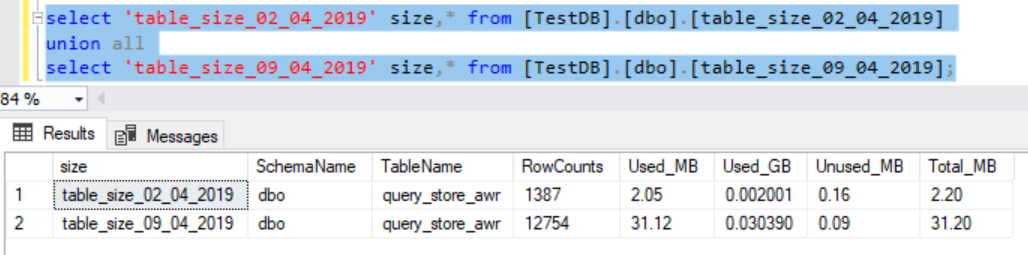
**For testing Purpose Implemented at UAT (*MerchantradeMoneyUAT*)**

Log from 02-04-2019 to 09-04-2019

* Store Query Count(Date wise):



* Table Size(Date wise):



* Database Size(Log Table Created Database):

