Query Analyzer User Guide

March 3, 2022

Revision History

Version	Date	Author	Comments
1.0	02/24/2020	Mark Hoerth	Initial release
1.1	03/19/2020	Deane Harding	Updated document format.
1.2	03/27/2020	Deane Harding	Added further VDS definitions
1.3	05/22/2020	Deane Harding	Minor update to queries.json scrubbing process.
1.4	07/21/2020	Deane Harding	Addition of functionality to retrieve error messages related to failed queries
1.5	10/08/2020	Deane Harding	Removed reliance on Java. Added more analysis VDSs.
1.6	03/03/2022	Deane Harding	Introduced splitting of error messages into 16k chunks

Related Documents

Name	Version
semantic_layer_best_practices.pdf	1.0, September 2019

Supported Versions

Name	Version
Dremio	3.2.8 or later
Python	3.5+

Table of Contents

Introduction

Purpose

Dremio provides a log of completed queries called `queries.json` that provides a rich and extensive view of queries on the system. The `queries.json` logs can be queried by Dremio itself or another tool for monitoring and analytics about the queries on the system. Such analytics can reveal new insight into query acceleration opportunities, most frequently run queries, or highest cost queries.

Dremio Query Analyzer is a tool that automates the retrieval, preparation and copying of current and historic `queries.json` files into S3/ADLS/HDFS file storage ready for analysis in Dremio. The tool also provides a set of VDS definitions that can be loaded into Dremio in order to enable immediate analysis of the query data.

Setup and Configuration

Install Dremio Query Analyzer

Dremio Query Analyzer is delivered as a compressed file called `dremio-query-analyzer.tar.gz`.

- Copy the file to the Dremio Coordinator node
- Unpack the file into a location of your choice, here we choose the `dremio` user's home directory

```
tar xvzf dremio-query-analyzer.tar.gz -C /home/dremio
```

- The tool also requires python in order to successfully execute. Therefore, ensure python is present on the Dremio Coordinator; first issue the following command to see if python is already installed.

```
Python -V
```

- If you are not presented with details of the currently installed python version, then you will need to install Python. Version 3.5 or above is recommended. The following site provides an explanation if required:

  https://docs.python-guide.org/starting/installation/

- **(Optional)** If you are going to be using `dremio-query-analyzer` to copy the test results files to Amazon S3, then you may need to install and configure the AWS CLI on the Dremio Coordinator. First issue the following command to see if the AWS CLI is already installed:

```
aws –version
```

  If you are not presented with details of the currently installed AWS CLI version, then use the following links as guidance.

  - To install the AWS CLI, follow these instructions: https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html
  - To configure the AWS CLI, follow these instructions:

https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html#cli-quick-configuration

- **(Optional)** If you are going to be using `dremio-query-analyzer` to copy the test results files to Azure Storage, then you may need to install and configure the Azure CLI on the Dremio Coordinator. First issue the following command to see if the Azure CLI is already installed:

```
az –version
```

  If you are not presented with details of the currently installed Azure CLI version, then use the following links as guidance:
  https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest

- **(Optional)** If you are going to be using `dremio-query-analyzer` to copy the test results files to Hadoop Storage, then you need to make sure the following command successfully executes on the Dremio Coordinator:

```
hdfs version
```

This `dremio-query-analyzer` package includes scripts for the one-time or continuous preparation of `queries.json` files from Dremio.

The `queries.json` log is located on the cluster coordinator, at the location `/var/log/dremio` and `/var/log/dremio/archive` on most clusters. See the Dremio documentation for additional detail.

`Dremio-query-analyzer` is responsible for cleaning up ("scrubbing") the `queries.json` file prior to its consumption in Dremio, ensuring rules regarding lengths of data fields when they are imported into Dremio are adhered to. The `dremio-query-analyzer` tool is also responsible for copying the scrubbed `queries.json` files into a user-defined cloud storage container (S3, ADLS, HDFS) that is accessible to Dremio in order to analyze the data, the script assumes that there will be two folders available in the designated storage location to store the scrubbed `queries.json` files, one called "`results`" and another called "`chunks`".

The main scripts of this package are defined below:

Script	Description
`gather_queries.sh`	This is the primary driver script for the dremio-query-analyzer. The script can be optionally installed in cron to regularly copy

	`queries.json` from the Dremio cluster to a location on S3\HDFS\ADLS. Requires customization with your configuration. Once a day is usually suitable for most users.
`scrub-queries-json.py`	Scrubs a set of `queries.json` files in a specified directory to truncate the queryText field into a number of equally sized chunks. The first chunk is written back into a `header.queries*.json` file, all other chunks are written into a `chunks.queries*.json` file. The scrubbed output files are written to a specified output directory. This is primarily because Dremio VARCHAR fields can be no greater than 32000 characters in length.
`get-error-messages.py`	Reads each `queries.json` file, identifies queries that failed and for each failed queries makes an API call to Dremio to retrieve a more descriptive message for why the query failed. Records get written to a new file called `errormessages.queries.*.json`.
`refresh-pds.py`	Refreshes metadata for a specified PDS in Dremio. Used in `gather_queries.sh` to refresh the metadata for results, chunks and errormessages PDSs in Dremio after new files have been uploaded to the data lake storage.

Queries.json analysis scripts provided in the dremio-query-analyzer package from Dremio Professional Services

Configure gather_queries.sh

The driving script of the `dremio-query-analyzer`, called `gather_queries.sh` requires the following variables inside the script to be specified correctly prior to use:

Variable	Description
`DREMIO_LOG_DIR`	The directory on the Dremio Coordinator where all the Dremio log files are stored. Default value is `"/var/log/dremio"`
`dremio_url`	The base URL for accessing the Dremio REST API, e.g. http://localhost:9047. Used when gather_queries.sh calls get-error-messages.py and refresh-pds.py
`user_name`	Name of an administrative user in Dremio that can access the REST API. Used when gather_queries.sh calls get-error-messages.py
`pwd`	Password associated with an administrative user in Dremio that can access the REST API. Used when gather_queries.sh calls get-error-messages.py. The password can be entered directly or stored in a file only accessible by the user running gather_queries.sh.

Variables in gather_queries.sh configured prior to use

Execution

At execution time, `gather_queries.sh`, requires the following three parameters as inputs on the command line:

Parameter	Description
`storage_type`	The type of storage where we want to write the scrubbed `queries.json` file to. Valid values are currently **s3**, **adls**, **hdfs**. If no valid value is supplied, then the scrubbed files will remain on the Dremio Coordinator and will need to be manually copied to a storage location.
`storage_path`	The path on the storage to a "results" folder e.g. s3://mybucket
`num_archive_days`	The number of days of archived `queries.json` files to also scrub and copy into storage. If no value is supplied, then only `queries.json` files for the current day will be scrubbed and copied.

Runtime parameters in gather_queries.sh

To make the results available for analysis, execute `gather_queries.sh` on the Dremio coordinator with the desired input parameter values for your chosen storage type, storage location and number of historical `queries.json` files that you wish to process. This script reads the desired `queries.json` and `queries.YYYY-MM-DD.N.json.gz` files and prepares them for analysis. e.g.:

```
cd /home/dremio/dremio-query-analyzer/scripts/
./gather_queries.sh s3 s3://mybucket/dremio 2
```

`gather_queries.sh` copies the `queries.json` and `queries.YYYY-MM-DD.N.json.gz` files locally to the `dremio-query-analyzer` in `/home/dremio/dremio-query-analyzer/scripts/dremio_queries`, it then generates one or more files called `header.<filename>.json`, `chunks.<filename>.json`, `errorheader.<filename>.json` and `errorchunks.<filename>.json` in the `/home/dremio/dremio-query-analyzer/scripts/dremio_queries/scrubbed/` directory.

The scrubbed files then get automatically copied into the desired cloud storage location. By default, the `header.<filename>.json` files will get copied into a `results` sub-directory, the `chunks.<filename>.json` files will get copied into a `chunks` sub-directory and the `errorheader.<filename>.json` files will get copied into an `errormessages` sub-directory and the `errorchunks.<filename>.json` files will get copied into an `errorchunks` sub-directory in the cloud storage.

If a desired cloud storage location is not specified or if the storage type is invalid then the files will remain in the `/home/dremio/dremio-query-analyzer/scripts/dremio_queries/scrubbed/` directory and they will require manually copying to some other storage location that will be accessible to Dremio as a data source.

Query Analysis

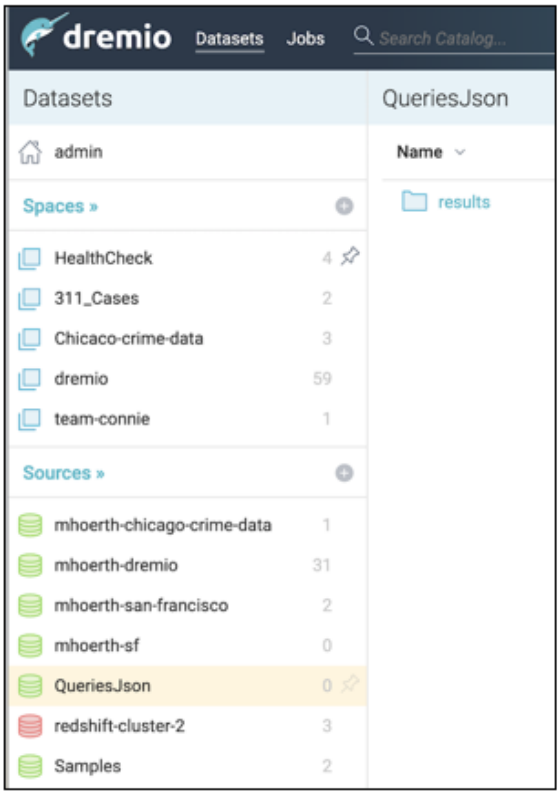
Before analysis begins there are several one-off tasks to perform in order to enable the analysis. Firstly, a data source and Physical Data Set (PDS) needs to be created in Dremio that will connect directly to the `queries.json` files. Secondly, a set of Virtual Data Sets (VDS) need to be created to query and make sense of the data in the PDS.

Create the PDSs

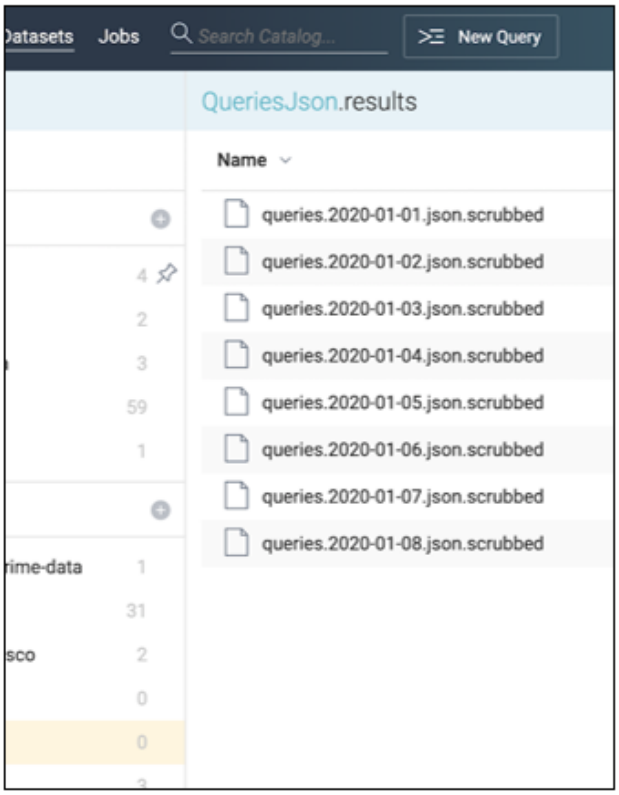
The assumption in this document is that the `queries.json` files have been copied into either S3, ADLS or HDFS storage, although equally they could have been placed on a user's local filesystem and then uploaded into an individual user's home space.

While not required, for the most straightforward integration with the pre-created scripts supplied in `dremio-query-analyzer` Dremio Professional Services recommends the `queries.json` files are placed into a folder called **`results`** in the chosen storage device.

- Create a data source in Dremio applicable to the storage device where the `queries.json` files are stored, e.g. Amazon S3, Azure Storage, HDFS.
- For the chosen data source, enter the relevant connection credentials.
- Navigate to the Advanced Options panel of the data source and enter the Root Path to be the **parent path** of the `results` folder on the storage device.
- Set up the path of the data source to point to the directory immediately above the results directory. The following screenshots show an Amazon S3 data source called QueriesJson configured with access to the results folder in Dremio (left) and also the contents of the results folder (right):

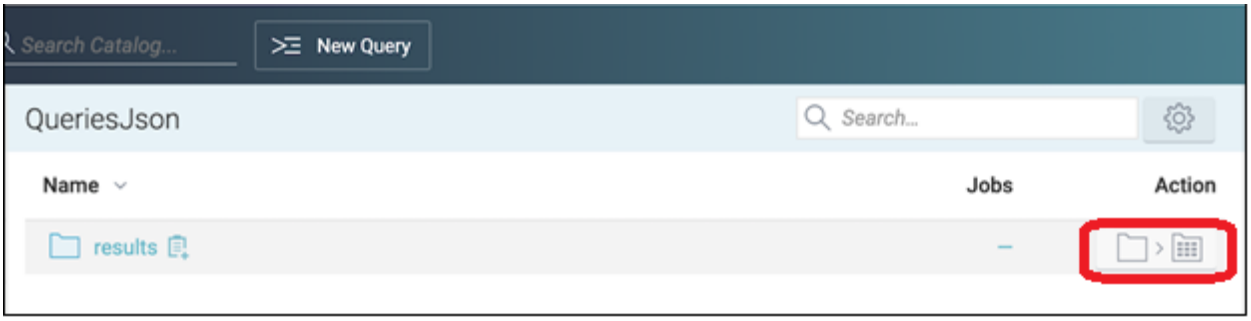


Example integration of queries.json files into Dremio for analysis. QueriesJson is an AWS S3 source that includes a directory called ***results****.*

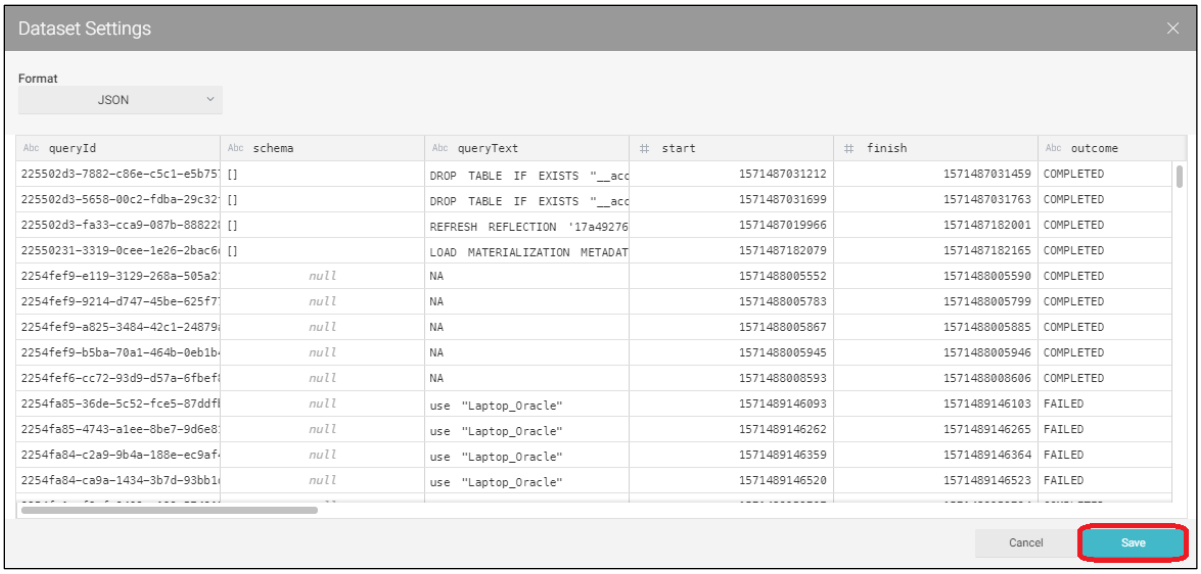


After scrubbing queries.json from several days of cluster operation, the files were uploaded to the ***results*** *directory on S3.*

- Use the Folder Format button to convert the results directory to a PDS



- In the resulting dialog, notice that Dremio automatically recognizes the format of the file inside this folder as JSON and has formatted the data table appropriately. Click Save.



- The same steps can be followed to convert the `chunks`, `errormessages` and `errorchunks` folders into PDSs in the same data source.

Create the VDSs

Create a set of VDSs that will be used to analyze the data in the `queries.json` files. There are two approaches to this, automated VDS creation, or manual. Both options are described in the sections below. Dremio recommends the automated approach.

Automated VDS creation

The primary file that is used to automatically create the relevant spaces, folders and VDSs in Dremio is called `run_vdscreator.sh`. This file is located at `/home/dremio/dremio-query-analyzer/scripts/vdscreator/run_vdscreator.sh` and has several parameters that need to be specified before VDSs can be created.

- Specify the following in `run_vdscreator.sh`. All other parameters in the file must not be edited.

Variable	Description
`dremio_host`	The hostname or IP address of the Dremio Coordinator
`dremio_port`	The HTTP port used by Dremio, default is 9047.
`user_name`	The name of an administrative user in Dremio that can create spaces, folders and VDSs.
`pwd`	The password for the user user_name above, read from `local.pwd` or embedded directly. This password will be used to log into Dremio via the REST API and issue calls to create the relevant objects.

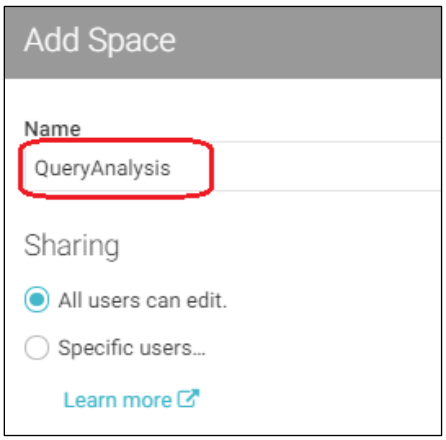
Variables in run_vdscreator.sh

- Execute `run_vdscreator.sh` to generate the required space, folders and VDSs in Dremio

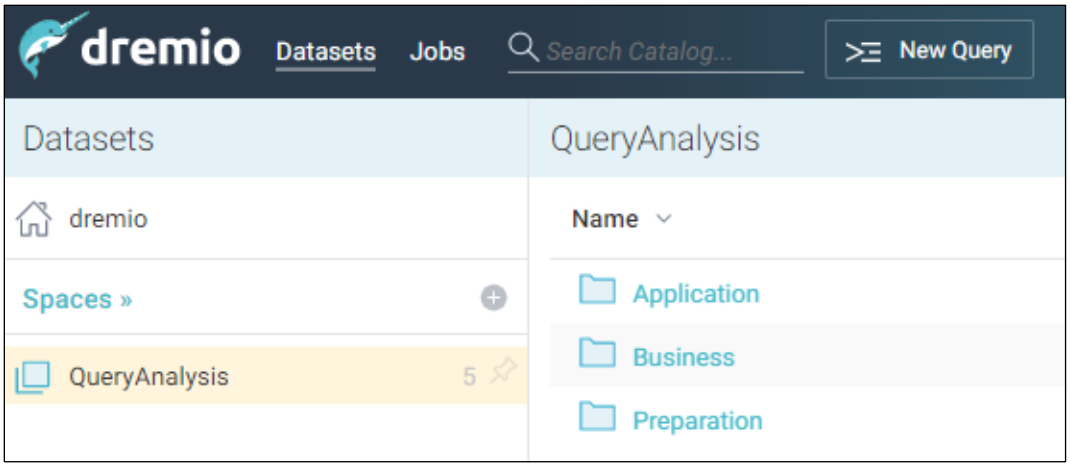
```
# Change user if necessary
su - dremio
cd /home/dremio/dremio-query-analyzer/scripts/vdscreator/
./run_vdscreator.sh
```

Manual VDS creation

- Create a new Space in Dremio called `QueryAnalysis`



- Inside the `QueryAnalysis` space create three new sub-folders: `Application, Business, and Preparation`



- On the Dremio Coordinator, navigate to the `vdsdefinition` directory:

```
cd /home/dremio/dremio-query-analyzer/vdsdefinition
```

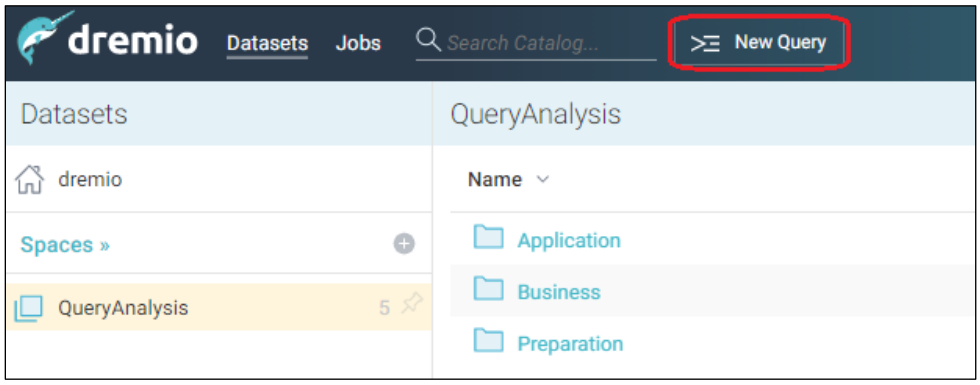
This directory contains the following `sql` files:

```
01_QueryAnalysis.Preparation.chunks.sql
01_QueryAnalysis.Preparation.errormessages.sql
01_QueryAnalysis.Preparation.errorchunks.sql
01_QueryAnalysis.Preparation.results.sql
02_QueryAnalysis.Business.QueryErrorMessages.sql
02_QueryAnalysis.Business.QueryTextChunks.sql
02_QueryAnalysis.Business.SelectQueryData.sql
03_QueryAnalysis.Application.HighCostSelects.sql
04_QueryAnalysis.Application.AcceleratedSelects.sql
05_QueryAnalysis.Application.NonAcceleratedSelects.sql
06_QueryAnalysis.Application.QueriesNotCompleted.sql
07_QueryAnalysis.Application.QueryCostVsExecutionTime.sql
08_QueryAnalysis.Application.Top20DataSetsQueried.sql
09_QueryAnalysis.Application.Top20ExecutionTimes.sql
10_QueryAnalysis.Application.Top20PlanningTimes.sql
11_QueryAnalysis.Application.Top20TotalDuration.sql
12_QueryAnalysis.Application.TotalQueriesPerMinute.sql
13_QueryAnalysis.Application.TotalQueriesPerMinutePerUser.sql
14_QueryAnalysis.Application.TotalQueriesPerUser.sql
15_QueryAnalysis.Application.PctAccelerated.sql
16_QueryAnalysis.Application.PctQueueUsageAllStartedQueries.sql
17_QueryAnalysis.Application.TotalQueriesPerMinutePerQueue.sql
18_QueryAnalysis.Application_TotalQueriesStartedPerSecond.sql
19_QueryAnalysis.Application.ActiveUsers.sql
20_QueryAnalysis.Application.FailedQueriesPerUser.sql
21_QueryAnalysis.Application.SelectWorkloadDistribution.sql
22_QueryAnalysis.Application.OverallWorkloadDistribution.sql
23_QueryAnalysis.Application.MostUsedDatasetsPerDay.sql
24_QueryAnalysis.Application.QueryConcurrency.sql
25_QueryAnalysis.Application.QueryConcurrencyCount.sql
26_QueryAnalysis.Application.QueryThresholds_2Queues.sql
27_QueryAnalysis.Application.QueryThresholds_3Queues.sql
```

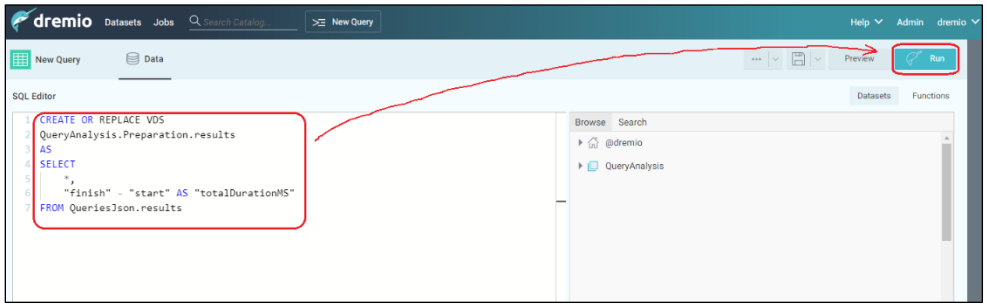
```
28_QueryAnalysis.Application.QueueMaxQueriesPerSecond.sql
29_QueryAnalysis.Application.Range_ExecutionTime.sql
30_QueryAnalysis.Application.Range_EnQueuedTime.sql
31_QueryAnalysis.Application.SummaryQueueExecTimeByQueue.sql
32_QueryAnalysis.Application.NtileQueryCost.sql
33_QueryAnalysis.Application.NtileQueryDuration.sql
```

For each of the files above in numeric order do the following:

- Copy the SQL out of the file
- In the Dremio UI, click on New Query



- Paste the SQL into the SQL Editor and click Run. This will create the VDS.



Analyze Results

Dremio is now available to query the VDSs in order to start analyzing the queries that have been flowing through your Dremio cluster. The VDSs supplied in this package are common examples of the sorts of questions a Dremio DBA might ask about the platform usage. Dremio encourages you to create your own VDSs as well in order to gain further insights.

Details of each of the VDSs in this package are shown below:

Create Order	VDS in QueryAnalysis space	Description
1a	Preparation.results	Directly queries the QueriesJson.results PDS. *Important: You may need to alter the FROM clause to point to the PDS on your system containing your scrubbed query data if the name is different than* ***QueriesJson.results***
1b	Preparation.chunks	Directly queries the QueriesJson.chunks PDS. *Important: You may need to alter the FROM clause to point to the PDS on your system containing your chunks of query text data if the name is different than* ***QueriesJson.chunks***
1c	Preparation.errormessages	Directly queries the QueriesJson.errormessages PDS. *Important: You may need to alter the FROM clause to point to the PDS on your system containing your query error message data if the name is different than* ***QueriesJson.errormessages***
1d	Preparation.errorchunks	Directly queries the QueriesJson.errorchunks PDS. *Important: You may need to alter the FROM clause to point to the PDS on your system containing your query error message data if the name is different than* ***QueriesJson.errorchunks***
2a	Business.SelectQueryData	Builds upon the results VDS, it filters out all except SELECT queries. *Important: Replace <DREMIO_HOST> in the query with the DNS name or IP address of the cluster. This is to identify this cluster in queries*
2b	Business.QueryTextChunks	Builds upon the chunks VDS and selects all fields from it. Can be joined with other VDSs via the queryId field in order to present the entire queryText for a query if it is longer than 4000 characters.
2c	Business.QueryErrorMessages	Builds upon the errormessages VDS and selects all fields from it. Can be joined with other VDSs via the queryId field.
3	Application.HighCostSelects	All high-cost queries
4	Application.Accelerated Selects	All accelerated queries
5	Application.NonAccelerated Selects	All non-accelerated queries. These may be opportunities for new or modified reflections
6	Application.QueriesNot Completed	All queries that did not complete, i.e. failed or canceled queries
7	Application.QueryCostVs ExecutionTime	Comparison of the query cost and execution time for each query
8	Application.Top20Data SetsQueried	List of the top 20 most frequently queried data sets

Create Order	VDS in QueryAnalysis space	Description
9	Application.Top20Execution Times	List of the top 20 queries with the highest execution times
10	Application.Top20 PlanningTimes	List of the top 20 queries with the highest planning times
11	Application.Top20 TotalDuration	List of the top 20 queries with the highest overall query duration (planning + enqueued + execution time)
12	Application.TotalQueries PerMinute	Count of how many queries started (but not necessarily finished) in any one-minute period
13	Application.TotalQueries PerMinutePerUser	Count of how many queries started per user (but not necessarily finished) in any one-minute period
14	Application.TotalQueries PerUser	Count of the total number of queries each user has executed, broken down by query outcome (completed, failed, canceled).
15	Application.PctAccelerated	Summary of the percentage of completed queries that were accelerated by reflections
16	Application.PctQueue UsageAllStartedQueries	Summary of the percentage of all started queries allocated to each queue
17	Application.TotalQueries PerMinutePerQueue	Total number of new queries being allocated to each queue, broken down by minute
18	Application.TotalQueries StartedPerSecond	Total number of new queries started in Dremio each second
19	Application.ActiveUsers	Summary per day of the total number of queries executed by each user and the total wall clock time spent running those queries.
20	Application.FailedQueries PerUser	Summary per day of the total number of failed queries issued by each user.
21	Application.SelectWorkload Distribution	Summary for each day and each hour, how many SELECT queries were executed in that hour as well as the peak number of SELECT queries ran per second.
22	Application.OverallWorkload Distribution	Summary for each day and each hour, how many total queries were executed in that hour as well as the peak number of queries ran per second.
23	Application.MostUsed DatasetsPerDay	Summary of the total number of queries and total number of users hitting each data set per day
24	Application.Query Concurrency	Summary for each individual query, what other queries were overlapping with this query during its execution and how did they overlap.
25	Application.Query ConcurrencyCount	Summary for each individual query, count of how many queries overlapped with this query during its execution.

Create Order	VDS in QueryAnalysis space	Description
26	Application.Query Thresholds_2Queues	75% percentile markers of queryCost which can be used to configure WLM for 2 queue model
27	Application.Query Thresholds_3Queues	75% & 90% percentile markers of queryCost which can be used to configure WLM for 3 queue model
28	Application.Queue MaxQueriesPerSecond	Summary of the maximum number of queries processed in a second for each queue
29	Application.Range_ ExecutionTime	Summary of range of execution time, with count and percentage of total (e.g., <1 sec, <10 sec, < 1 min....)
30	Application.Range_ EnQueuedTime	Summary of range of enqueued time, with count and percentage of total (e.g., <1 sec, <10 sec, < 20 secthese....)
31	Application.Summary QueueExecTimeByQueue	Summary of count of queries, their execution time, queue time for each queue
32	Application.NtileQueryCost	100 percentile buckets of all queries by QueryCost
33	Application.NtileQueryDuration	100 percentile buckets of all queries by QueryDuration

VDS definitions in this package

These VDS are designed to be examples. Consider modifications or additional datasets you could create:

- QueryAnalysis.Business.SelectQueryData is an example of how to filter out INSERTS, UPDATES, and DELETES. Use other fields of the Preparation.results VDS to filter on selected query types, durations, user names, or other characteristics.
- The application layer VDSs are typical of the datasets for end-users and could be analyzed with Tableau or other front-end tools.
- Please share your new VDS and query analysis insights with others by contacting Dremio Professional Services.

Getting Support

Dremio Query Analyzer is provided by Dremio Professional Services. Issues should be directed to Dremio Professional Services, who provide support during the engagement on a best-effort basis.