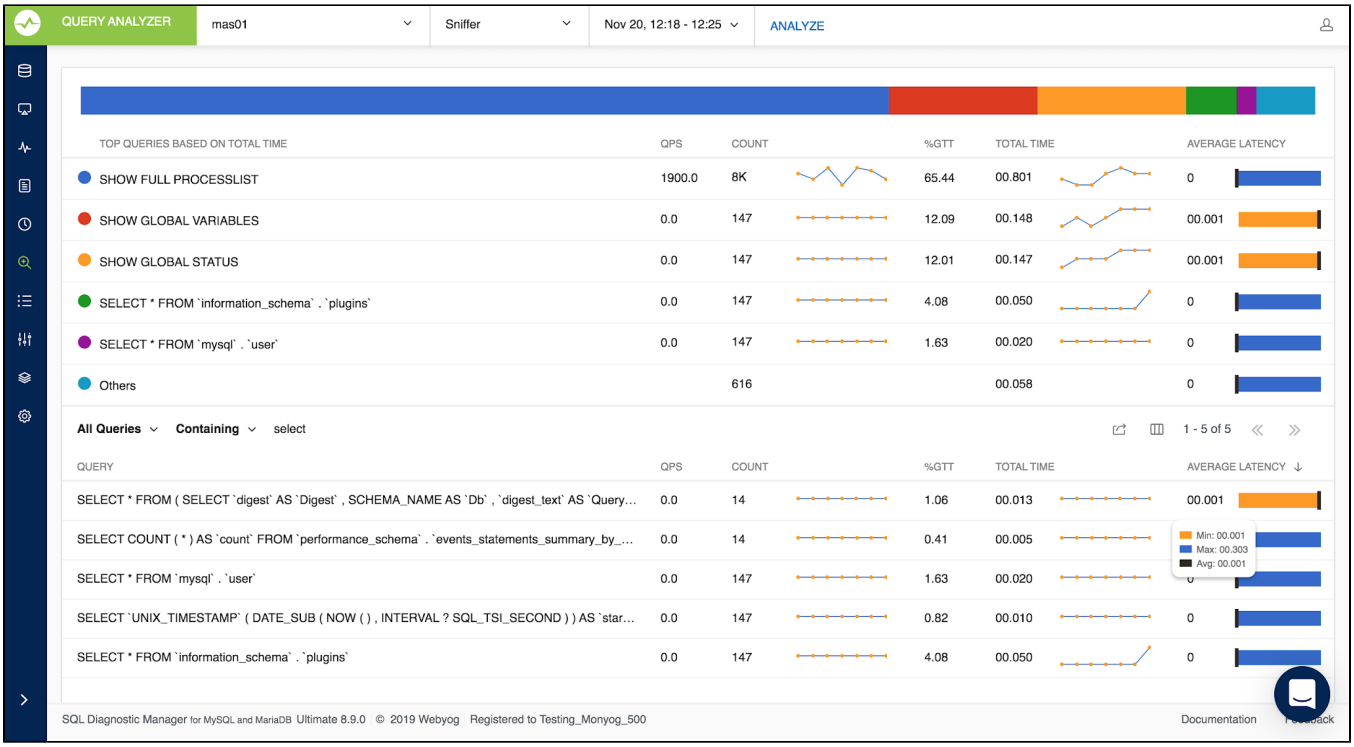
Sniffer

SQL DM for MySQL Query Sniffer is a functionality that records a pseudo server log and stores it in the SQL DM for MySQL embedded database. With **Query sniffer** enabled, SQL DM for MySQL can populate the pseudo server log in three different ways at the intervals you specify:

- By utilizing Performance Schema tables (`events_statements_summary_by_digest`, `events_statements_history_long`) and collecting snapshots at regular intervals.
- By sending the query `SHOW FULL PROCESSLIST` to the MySQL server.
- Or, by connecting to a running instance of the MySQL-Proxy program that is used by one or more clients to connect to a MySQL server.



For MySQL 5.6.14 and above you can use Performance schema, Proxy, and Processlist for query analysis. If using MySQL version less than 5.6.14 then only Proxy or Processlist can be used in SQL DM for MySQL.

Performance Schema on MySQL contains queries executed on server along with other information:

- Number of rows sent and examined
- Number of temporary tables created on disk
- Number of temporary tables created on memory
- Number of joins performed and the type of join
- Whether sorting happened and the type of sort
- Whether index used
- Whether good index used

SQL DM for MySQL uses performance schema statement digest `table(events_statements_summary_by_digest)` to get the above information and is dependent on the statements_digest in setup_consumers table. By default, this option is enabled, you can disable it by executing the following:

```
UPDATE performance_schema.setup_consumers
SET enabled = 'YES'
WHERE name = 'statements_digest';
```

Example query is available in `events_statements_history_long` table and has to be enabled and is dependent on the `events_statements_history_long` in setup_consumers table. By default, this is not enabled and should be enabled by executing the following:

```
UPDATE performance_schema.setup_consumers
SET enabled = 'YES'
WHERE name = 'events_statements_history_long';
```

The `performance_schema.events_statements_summary_by_digest` table size is dependent on `performance_schema_digests_size` `global` variable. By default, this size is set to 5000 rows. Once it reaches this limit you may lose the queries. SQL DM for MySQL provides an option to truncate the performance schema digest table when it reaches 80% of `performance_schema_digests_size`.

Performance schema based sniffer comes with different filters like: Queries with errors, Queries with warnings, Queries with missing indexes, and Queries with poor indexes.

> ⚠ `performance_schema` truncates queries after 1024 characters and always replaces literals with a wildcard (in other words: P_S contains a summary/an aggregation only). So query listing not replacing literals is not possible with this option. And finally also observe that no other tool (or user) should be writing (including deleting or truncating) to `events_statements_summary_by_digest` and `events_statements_history_long` tables if this option is used as there is only one of each table for all users (it is not a temporary table or a materialized view or similar private for the user). This is a design limitation with the tables in P_S as such and not a Monyog issue.

If using MySQL version less than 5.6.14 then only Proxy or Processlist can be used in SQL DM for MySQL. Although, configuring a Proxy instance is a little more complicated, the PROXY-based sniffer has several advantages over the PROCESSLIST-based, including the following:

1. All queries that was handled by the Proxy are recorded by SQL DM for MySQL sniffer when PROXY option is used. When PROCESSLIST option is used very fast queries may execute completely between two SHOW FULL PROCESSLIST queries andis not recorded.
2. You can choose to analyze queries from specific client(s)/application(s) only. Simply let (only) the clients that you want to focus on at the moment connect through the Proxy.
3. When using the PROXY option you can distribute part of the load generated by the sniffer on the machine that fits best in your deployment scenario (like on the one that has most free resources available) by deciding where to have the PROXY: The MySQL machine, the SQL DM for MySQL machine (if not the same), or quite another machine. The machine running MySQL have no additional load due to the sniffer if the Proxy is not running on that machine.

Also, if more SQL DM for MySQL instances use the same PROXY they use the same data collected, when the Proxy Sniffing is enabled by the first SQL DM for MySQL instance. To work with SQL DM for MySQL sniffer the MySQL Proxy instance must be started with the name of a LUA script called **MONyog.LUA** (LUA is a scripting/programming language) as argument and is distributed with SQL DM for MySQL. You find it in the Monyog program folder after installing (Windows and Linux RPM) or unpacking (Linux .tar.gz) the SQL DM for MySQL program package as downloaded from the IDERA website. The MySQL Proxy program however you need to download from MySQL website (we cannot include it for license reasons). SQL DM for MySQL works with Proxy versions from 0.61 to 0.81(latest currently) with the exception of 0.7x versions for windows and Mac due to a bug in those specific builds. For more information on Proxy MySQL Proxy.

To start a Proxy instance for use with SQL DM for MySQL use the command:

- **For Older version:**

```
Proxy installation folder>mysql-proxy --proxy-backend-addresses=192.168.x.x:3305 --proxy-address=192.168.y.y:
4045 --proxy-lua-script=SQL DM for MySQL.lua
```

- **For v0.81 and later:**

```
Proxy installation folder>mysql-proxy --proxy-backend-addresses=192.168.x.x:3305 --proxy-address=192.168.y.y:
4045 --admin-username=root --admin-password=root --admin-lua-script=MONyog.lua --proxy-lua-script=MONyog.lua
```

(It is assumed that the 'MONyog.LUA' was copied to the folder where the PROXY binary is). Also, if no port is specified the PROXY listens on port 4040. Now, you can connect to the Proxy from one or more clients/applications. The Proxy sends queries to MySQL and results back to the client. But when started with the LUA script for SQL DM for MySQL sniffer it also send information to SQL DM for MySQL that SQL DM for MySQL uses to populate the sniffer 'pseudo log'.

Once this 'pseudo log' has been recorded (in either of the two ways described: PROCESSLIST or PROXY-based) the SQL DM for MySQL log analysis functionalities can operate on the 'pseudo log' as well as the 'real logs'. The data recorded in the pseudo log is purged automatically based on the 'data retention timeframe' option set by you.

Further some filtering options are provided. This filtering happens before storing to the SQL DM for MySQL database. It prevents the sniffer database to grow out of control. The filtering options are as follows:

- **User and host**: You can choose to store queries executed only by a specific combination of users and/or hosts.
- **Minimum time taken:** For every PROCESSLIST returned to SQL DM for MySQL, the queries are recorded in the embedded database only if they have been executing for a time greater than the specified minimum execution time. Furthermore, if a query with the same structure and details (like process ID) as one already recorded is encountered, the embedded database is UPDATED, and the statement is recorded only once. This setting should be somewhat larger than the sample interval (and also consider the latency of the connection).
- **Queries starting with:** Enter any string and only queries starting with that string is recorded. Examples: `SELECT *`, `UPDATE Customer_Base`.

Also note, in the `PROCESSLIST` Sniffer there is an option 'Long Running Query Options' where you can monitor the long running queries by notifying or killing a query which takes more than a time specified by you. You can also specify users whose queries are ignored (i.e. queries) by such user are not killed by SQL DM for MySQL and never raise an alert even if they take a longer time to execute than the alert/kill setting time you specified.

Clicking the **monitor only locked queries** would only monitor those long queries that are locked.

Note that the query sniffer is never a complete General Log. Very fast statements may or may not be recorded as they may or may not finish executing between two `PROCESSLIST`'s generated. The time interval between subsequent data collections for the 'pseudo log' depends on the connection to the MySQL server.

The identical queries are only listed once and the Count column tells how many times this query was executed.