

Workload XM 2.1.3

Workload XM Reference Material

Date published: 2020-12-04

Date modified: 2021-04-09

CLOUDERA

<https://docs.cloudera.com/>

Legal Notice

© Cloudera Inc. 2022. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Workload XM Reference Overview.....	4
Data Engineering Health Checks.....	4
Data Warehouse Health Checks.....	6
Data Warehouse Query Status.....	9
Data Warehouse Statement Types.....	10
Potential SQL Issues.....	12

Workload XM Reference Overview

This section provides additional information that support the features and functions in Workload XM.

The following topics provide descriptions of health checks for data engineering jobs, which involve Hive, MapReduce, and Spark, and descriptions of health checks for data warehousing workloads, which involve Impala. In addition to health check descriptions, these topics also provide recommendations for addressing the conditions that trigger health checks and information about the query statuses, types, and potential SQL issues that are identified by Workload XM.

Data Engineering Health Checks

Lists the data engineering health check tests that are performed by Workload XM at the end of a Hive, Spark, or MapReduce job. They provide job performance insights, such as the amount of data the job processed and how long the job took. You can find the data engineering health checks on the Data Engineering Jobs page in the **Health Check** list.

Table 1: Data Engineering Health Checks

Health Check	Description
Failed - Any Health Checks	Displays jobs that failed at least one health check.
Passed All Health Checks	Displays jobs that did not fail any health checks.
All Jobs	Displays all jobs, regardless of health status.
Failed to Finish	Displays jobs that failed to finish running.
<p>Baseline - The health checks for baselines use information from previous runs of the same job to measure the performance of the current run of the job. Baselines provide a way to measure the current performance of a job against the average performance of previous runs. Baselines use performance data from the 30 most recent runs of a job and require a minimum of three runs. Baseline comparisons start with the fourth run of a job. When a baseline is created, there can be drastic differences when comparing runs to the baseline. As a baseline matures and more runs of a job are added to it, you can see a more established trend of what is normal for the job.</p> <p> Important: Workload XM uses job name, job group name, and environment to correlate the job data and create the baselines. These values for subsequent runs of the job must be identical to the initial run in order for the baseline to be accurate.</p>	
Duration	Compares the completion time of the job to a baseline based on previous runs of the same job. A healthy status indicates that the difference in duration between the current job and baseline median is less than both 25% and five minutes.
Input Size	Compares the input for the current run of a job to a baseline for the job. A healthy status indicates that the difference in input data between the current job and baseline median is less than 25% and 100 MB. To calculate input size, Workload XM uses the following metrics: <ul style="list-style-type: none"> <code>org.apache.hadoop.mapreduce.FileSystemCounter:HDFS_BYTES_READ</code> <code>org.apache.hadoop.mapreduce.FileSystemCounter:S3A_BYTES_READ</code> <code>SPARK:INPUT_BYTES</code>

Health Check	Description
Output Size	<p>Compares the output for the current run of a job to the baseline for the job. A healthy status indicates that the difference in output data between the current job and baseline median is less than 25% and 100 MB. To calculate output size, Workload XM uses the following metrics:</p> <ul style="list-style-type: none"> org.apache.hadoop.mapreduce.FileSystemCounter:HDFS_BYTES_WRITTEN org.apache.hadoop.mapreduce.FileSystemCounter:S3A_BYTES_WRITTEN SPARK:OUTPUT_BYTES
Resources - The resource health checks determine whether the performance for tasks were impacted by insufficient resources.	
Task Retries	Determines whether the number of failed task attempts exceeds 10% of the total number of tasks. Failed attempts need to be repeated, leading to poor performance and resource waste.
Task GC Time	Determines whether tasks spent more than 10 minutes performing garbage collection. Long garbage collection duration contributes to task duration and slows down the application. If the status is not healthy, try giving more memory to tasks or tune the garbage collection configuration for the application as a starting point.
Disk Spillage	Determines if tasks spilled too much data to disk and ran slowly as a result of the extra disk I/O. A healthy status indicates that the total number of spilled record is less than 1000 and that the number of spilled records divided by the number of output records is less than three. If the status is not healthy, try giving more memory to tasks as a starting point.
Task Wait Time	Determines if some tasks took too long to start a successful attempt. A healthy status indicates that successful tasks took less than 15 minutes and less than 40% of total task duration time to start. Sufficient resources cut the run time of the job by lowering the maximum wait duration. If the status is not healthy, try giving more resources to the job by running it in resource pools with less contention or by adding more nodes to the cluster as a starting point.
RDD Caching	Verifies that the RDDs were cached successfully. A healthy status indicates that the RDDs were cached successfully and Workload XM did not determine that there was a redundant RDD cache. If the status is not healthy, the message will indicate whether there was a redundant cache that you can remove to save executor space.
Skew - The skew health checks compare the performance of tasks to other tasks within the same job. For optimal performance, tasks within the same job should perform the same amount of processing.	
Task Duration	Compares the amount of time tasks take to finish their processing. A healthy status indicates that successful tasks took less than two standard deviations and less than five minutes from the average for all tasks. If the status is not healthy, try to configure the job so that processing is distributed evenly across tasks as a starting point.
Data Processing Speed	Compares the data processing speed for each task. A healthy status indicates that the data processing speed for each task is less than two standard deviations from the average and less than 1 MB/s from the average. Indicates which tasks are processing data slowly.
Input Data	Compares the amount of input data that each task processed. A healthy status indicates that input data size is less than two standard deviations and 100 MB from the average amount of input data. If the status is not healthy, try to partition data so that each task processes a similar amount of input as a starting point.

Health Check	Description
Output Data	Compares the amount of output data that each task generated. A healthy status indicates that output data size is less than two standard deviations and 100 MB from the average amount of output data. If the status is not healthy, try partitioning data so that each task generates a similar amount of output as a starting point.
Shuffle Input	Compares the input size during the shuffle phase for tasks. A healthy status indicates that the shuffle phase input data size is less than two standard deviations and 100 MB from the average amount of shuffle phase input data. If the status is not healthy, try distributing input data so that tasks process similar amounts of data during the shuffle phase as a starting point.

Data Warehouse Health Checks

Lists the data warehouse health check tests that are performed by Workload XM at the end of an Apache Impala job. They provide tips on how to make your workloads run faster and point out which aspects of your queries might be causing bottlenecks on your cluster. However, their recommendations are not exhaustive and there may be additional fixes other than those described. It is important to note that query tuning is as much an art as a science. If you are currently satisfied with your cluster performance, you can use these health checks as a way to gain insights into how your query workloads are running on your cluster. That said, the suboptimal conditions identified by these health checks might cause problems as new applications are added, the system footprint is expanded, or the overall load on the system increases. Use these health checks to proactively monitor potential issues across your cluster. You can find the data warehouse health checks on the Data Warehouse Queries page in the **Health Check** list and on the Data Warehouse Summary page in the **Suboptimal Queries** graph.

Table 2: Data Warehouse Health Checks

Health Check	Description
Aggregation Spilled Partitions	<p>Indicates that data spilled to disk during the aggregation operation for these queries.</p> <p>This health check is triggered during aggregation if there is not enough memory, which causes data to spill to disk. If you are satisfied with your cluster performance despite this health check being triggered, you can disregard it.</p> <p>If performance is an issue, try the following fixes:</p> <ul style="list-style-type: none"> • Use a less complex GROUP BY clause that involves fewer columns (do not use a high cardinality GROUP BY clause). • Increase the setting for the query's MEM_LIMIT query option. See the Impala documentation. • Add more physical memory. <p>For more details, see the Impala documentation SQL Operations that Spill to Disk.</p>
Bytes Read Skew	<p>Indicates that one of the cluster nodes is reading a significantly larger amount of data than other nodes.</p> <p>To address this condition, rebalance the data or use the Impala SCHEME_RANDOM_REPLICA query option. For additional suggestions, see Avoiding CPU Hotspots for HDFS Cached Data in the Impala documentation set.</p>
Corrupt Table Statistics	<p>Indicates that these queries contain table statistics that were incorrectly computed and cannot be used.</p> <p>This condition can be caused by metastore database issues.</p> <p>Recompute table statistics. For more information, see Detecting Missing Statistics in the Impala documentation set.</p>

Health Check	Description
HashJoin Spilled Partitions	<p>Indicates that data spilled to disk during the hash join operation for these queries.</p> <p>This condition occurs when there is not enough memory during the hash join, which causes data to spill to disk.</p> <p>To address this issue:</p> <ul style="list-style-type: none"> • Reduce the cardinality of the right-hand side of the join by filtering more rows from it. • Add more physical memory. • Increase the setting for the query's MEM_LIMIT query option. See the Impala documentation. • Use a denormalized table.
Insufficient Partitioning	<p>Indicates that there is insufficient partitioning for parallel query execution to occur for these queries.</p> <p>This condition is triggered when query execution is wasting resources and time because the system is reading rows that are not required for the operation.</p> <p>To address this condition:</p> <ul style="list-style-type: none"> • Check to see if your more popular filters can become partition keys. For example, if you have many queries that use ship date as a filter, consider creating partitions using ship date as the partition key. • Add filters to your query for existing partition columns. <p>For more information, see Partitioning for Impala Tables in the Impala documentation set.</p>
Many Materialized Columns	<p>Indicates that an abnormally large number of columns were returned for these queries.</p> <p>This condition is only triggered for Parquet tables. If you are reading more than 15 columns, this health check is triggered.</p> <p>To address this condition, rewrite the query so it does not return more than 15 columns.</p>
Missing Table Statistics	<p>Indicates that no table statistics were computed for query optimization for these queries.</p> <p>To address this condition, compute table statistics. For more information, see Detecting Missing Statistics in the Impala documentation set.</p>
Slow Aggregate	<p>Indicates that the aggregation operations were slower than expected for these queries.</p> <p>Ten million rows per second is the typical throughput and if the observed throughput is less than that, this health check is triggered. Observed throughput is calculated by dividing the time spent in the aggregation operation into the number of input rows.</p> <p>Addressing this condition depends on the root cause:</p> <ul style="list-style-type: none"> • If the root cause is resource conflicts with other queries, then allocate different resource pools to reduce conflicts. • If the root cause is overly complex GROUP BY operations, then rewrite the queries to simplify the GROUP BY operations.

Health Check	Description
Slow Client	<p>Indicates that the client consumed query results slower than expected for these queries.</p> <p>The causes and fixes for this health check can vary:</p> <ul style="list-style-type: none"> • If the condition is triggered because some clients are taking too long to unregister the query, then use more appropriate clients for the workload. For example, if you are testing and building SQL queries, it might make more sense to use an interactive client over ODBC or JDBC. • If the condition is triggered because you are doing exploratory analysis and reading some rows and then waiting for some time to read the next set of rows, this uses up systems resources because the query has not closed. To remediate, consider using the Impala timeout feature. See Setting Timeout Periods for Daemons, Queries, and Sessions in the Impala documentation set. As an additional option, consider adding a <code>LIMIT</code> clause to your queries to limit the number of rows returned to 100 or less.
Slow Code Generation	<p>Indicates that compiled code was generated more slowly than expected for these queries.</p> <p>In every query plan fragment, Impala considers how much time is used to generate the code and this health check indicates that the time exceeded 20% of the overall query execution time. This might be triggered by query complexity. For example, if the query has too many predicates in its WHERE clauses, too many joins, or too many columns.</p> <p>For queries where code generation is too slow, consider using the <code>DISABLE_CODEGEN</code> query option in your session.</p>
Slow HDFS Scan	<p>Indicates that scanning data from HDFS was slower than expected for these queries.</p> <p> Note: If the workload is accessing data that is stored on Amazon S3, this is a known limitation of that storage platform.</p> <p>This condition is caused by a slow disk, extremely complex scan predicates, or the HDFS NameNode is too busy. The HDFS scan rate is based on the amount of time that the scanner took to read a specific number of rows.</p> <p>This condition can be addressed by:</p> <ul style="list-style-type: none"> • Replacing the disk if the cause is a slow disk. • Reduce complexity by simplifying the scan predicates. • If the HDFS NameNode is too busy, consider upgrading to CDH 5.15 or later. For more information, see Upgrading Cloudera Manager and CDH.
Slow Hash Join	<p>Indicates that hash join operations were slower than expected for these queries.</p> <p>This health check might be triggered when there are overly complex join predicates or the hash join is causing data to spill to disk. Five million rows per second is the typical throughput and if the observed throughput is less than that, this health check is triggered. Observed throughput is calculated by dividing the number of input rows by the time spent in the hash join operation.</p> <p>To fix this condition, simplify the join predicates or reduce the size of the right side of the join.</p>

Health Check	Description
Slow Query Planning	<p>Indicates that the query plan was generated more slowly than expected for these queries.</p> <p>This health check is triggered when the query planning time exceeds 30% of the overall query execution time. This can be caused by very complex queries or if a metadata refresh occurs while the query is executing.</p> <p>To fix this condition, consider simplifying your queries. For example, reduce the number of columns returned, reduce the number of filters, or reduce the number of joins.</p>
Slow Row Materialization	<p>Indicates that rows were returned more slowly than expected for these queries.</p> <p>This health check is triggered if it takes more than 20% of the query execution time to return rows. It can be caused by overly complex expressions in the <code>SELECT</code> list or when too many rows are requested.</p> <p>To address this condition, simplify the query by reducing the number of columns in the select list or by reducing the number of rows requested.</p>
Slow Sorting Speed	<p>Indicates that the sorting operations were slower than expected for these queries.</p> <p>Ten million rows per second is the typical throughput and if the observed throughput is less than that, this health check is triggered. Observed throughput is calculated by dividing the number of input rows by the time spent in the sorting operation.</p> <p>To fix this condition, simplify the <code>ORDER BY</code> clause in queries. If data is spilling to disk, reduce the volume of data to be sorted by adding more predicates to the <code>WHERE</code> clauses, by increasing the available memory, or by increasing the value specified for the <code>MEM_LIMIT</code> query option. See the Impala documentation.</p>
Slow Write Speed	<p>Indicates that the query write speed is slower than expected for these queries.</p> <p> Note: If the workload is accessing data that is stored on Amazon S3, this is a known limitation of that storage platform.</p> <p>If the difference between actual write time and the expected write time are more than 20% of the query execution time, this health check is triggered. This condition can be caused when overly complex expressions are used, too many columns are specified, or too many rows are requested in the <code>SELECT</code> list.</p> <p>To address this condition, simplify the query by reducing the number of columns, or by reducing the complexity of the <code>SELECT</code> list expression.</p>

Data Warehouse Query Status

Lists the query statuses for data warehousing workloads that use Apache Impala. You can find the status of your query on either the Data Warehouse Summary page in the Failed Queries graph or on the Data Warehouse Queries page in the Status list.

Table 3: Data Warehouse Query Status

Query Status	Description
Analysis Exception	These queries failed due to syntax errors or incorrect table or column names.

Query Status	Description
Authorization Exception	These queries failed because the user executing the queries does not have permission to access the data.
Cancelled	These queries were cancelled by the system or a user.
Exceeded Memory Limit	The amount of memory required to execute these queries exceed the allocated memory limit.
Failed - Any Reason	The queries that failed for any reason are listed here.
Other Failures	These queries failed for other unclassified reasons.
Rejected from Pool	These queries failed because there are too many queries already pending in the Impala resource pool.
Session Closed	The session was closed by the system or a user for this set of queries.
Succeeded	The query succeeded.

Data Warehouse Statement Types

Lists the SQL statement types for data warehousing workloads that use Apache Impala. You can find the statement types on the Data Warehouse Queries page in the Type list. For more detailed information about these types of SQL statements, see the [Impala documentation](#).

Table 4: Data Warehouse Statement Types

Statement Type	Description
ALTER TABLE	Changes the structure or properties of an existing table. For example, <code>ALTER TABLE table_name ADD PARTITION (month=1, day=1);</code>
ALTER VIEW	Changes the characteristics of a view. For example, <code>ALTER VIEW view_name AS SELECT * FROM table_name;</code>
COMPUTE STATS	Gathers information about volume and distribution data in a table and all associated columns and partitions. For example, <code>COMPUTE STATS table_name;</code>
CREATE DATABASE	Creates a new database. For example, <code>CREATE DATABASE database_name;</code>
CREATE FUNCTION	Creates a user-defined function (UDF), which you can use to implement custom logic during SELECT or INSERT operations. For example, <code>CREATE FUNCTION function_name LOCATION 'hdfs_path_to_jar' SYMBOL='class_name';</code>
CREATE ROLE	Creates a role to which privileges can be granted. After privileges are granted to roles, then the roles can be assigned to users. A user who has been assigned a role is only able to exercise the privileges of that role. For example, <code>CREATE ROLE role_name;</code>
CREATE TABLE	Creates a new table and specifies its characteristics. For example, <code>CREATE TABLE table_name (column_name data_type) PARTITIONED BY (column_name data_type) LOCATION 'hdfs_path';</code>
CREATE TABLE AS SELECT	Creates a new table with the output from a SELECT statement. For example, <code>CREATE TABLE table_name AS SELECT * FROM table_3;</code>
CREATE TABLE LIKE	Creates a new table by cloning an existing table. For example, <code>CREATE TABLE table_name_2 LIKE table_name_1;</code>

Statement Type	Description
CREATE VIEW	Creates a shorthand abbreviation (alias) for a query. A view is a purely logical construct with no physical data behind it. For example, <code>CREATE VIEW view_name AS SELECT * FROM table_name;</code>
DDL	Data Definition Language. SQL statements that define data structures. For example, <code>CREATE TABLE;</code>
DESCRIBE DB	Displays metadata about a database. For example, <code>DESCRIBE database_name;</code>
DESCRIBE TABLE	Displays metadata about a table. For example, <code>DESCRIBE table_name;</code>
DML	Data Manipulation Language. SQL statements that manipulate data structures. For example, <code>ALTER TABLE;</code>
DROP DATABASE	Removes a database from the system. For example, <code>DROP database_name;</code>
DROP FUNCTION	Removes a user-defined function (UDF) so that it is not available for execution during Impala SELECT or INSERT operations. For example, <code>DROP FUNCTION function_name;</code>
DROP STATS	Removes the specified statistics from a table or a partition. For example, <code>DROP STATS table_name;</code>
DROP TABLE	Removes a table and its underlying HDFS data files for internal tables, although not for external tables. For example, <code>DROP TABLE table_name;</code>
DROP VIEW	Removes the specified view. Because a view is purely a logical construct with no physical data behind it, DROP VIEW only involves changes to metadata in the metastore database, not any data files in HDFS. For example, <code>DROP VIEW view_name;</code>
EXPLAIN	Generates a query execution plan for a specific query. For example, <code>EXPLAIN SELECT * FROM table_1;</code>
GRANT PRIVILEGE	Grants privileges on specified objects to groups. For example, <code>GRANT privilege_name ON TABLE table_name TO role_name;</code>
GRANT ROLE	Grants roles on specified objects to groups. For example, <code>GRANT ROLE role_name TO GROUP group_name;</code>
LOAD	Loads data from an external data source into a table. For example, <code>LOAD DATA INPATH 'hdfs_file_or_directory_path' INTO TABLE tablename;</code>
N/A	These queries failed due to syntax errors and Impala is not able to identify a query type for them.
REFRESH	Reloads the metadata for a table from the metastore database and does an incremental reload of the file and block metadata from the HDFS NameNode. REFRESH is used to avoid inconsistencies between Impala and external metadata sources, specifically the Hive Metastore and the NameNode. For example, <code>REFRESH table_name;</code>
REVOKE PRIVILEGE	Revokes privileges on a specified object from groups. For example, <code>REVOKE privilege_name ON TABLE table_name;</code>
REVOKE ROLE	Revokes roles on a specified object from groups. For example, <code>REVOKE ROLE role_name FROM GROUP group_name;</code>
SELECT	Requests data from a data source. For example, <code>SELECT * FROM table_1;</code>
SET	Sets configuration properties or session parameters. For example, <code>SET compression_codec=snappy;</code>

Statement Type	Description
SHOW COLUMN STATS	Displays the column statistics for a specified table. For example, <code>SHOW COLUMN STATS table_name;</code>
SHOW CREATE TABLE	Displays the CREATE TABLE statement used to reproduce the current structure of a table. For example, <code>SHOW CREATE TABLE table_name;</code>
SHOW DATABASES	Displays all available databases. For example, <code>SHOW DATABASES;</code>
SHOW FILES	Displays the files that constitute a specified table or a partition within a partitioned table. For example, <code>SHOW FILES IN table_name;</code>
SHOW FUNCTIONS	Displays user-defined functions (UDFs) or user-defined aggregate functions (UDAFs) that are associated with a particular database. For example, <code>SHOW FUNCTIONS IN database_name;</code> or <code>SHOW AGGREGATE FUNCTIONS IN database_name;</code>
SHOW GRANT ROLE	Lists all the grants for the specified role name. For example, <code>SHOW GRANT ROLE role_name;</code>
SHOW ROLES	Displays all available roles. For example, <code>SHOW ROLES;</code>
SHOW TABLES	Displays the names of tables. For example, <code>SHOW TABLES;</code>
SHOW TABLE STATS	Displays the statistics for a table. For example, <code>SHOW TABLE STATS table_name;</code>
TRUNCATE TABLE	Removes the data from an Impala table, while leaving the table itself. For example, <code>TRUNCATE TABLE table_name;</code>
USE	Switches the current session to a specified database. For example, <code>USE database_name;</code>

Potential SQL Issues

Lists the most common mistakes made during SQL statement creation that are identified by Workload XM as potential issues. You can find the potential issues associated with a query on the Data Warehouse Queries page in the Performance Issues region of the query details page.

Table 5: Common SQL Issues

Potential SQL Issue	Impact and Recommendation
>5 table joins or > 10 join conditions found.	Possible performance impact depending on table sizes, partitioning keys, filter and join conditions that are specified in the query. To address this issue, denormalize tables to eliminate the need for joins.
>10 columns present in GROUP BY list.	Possible performance impact, depending on the number of distinct groups and the memory configuration. This issue is not raised if the source platform is Impala. To address this issue, evaluate the memory requirements for the query.
>10 Inline Views present in query.	Possible performance impact, depending on the memory configuration, especially if complex expressions are present in inline views on Impala. To address this issue, evaluate the memory requirements and materialize inline views.
>50 query blocks present in large query.	Possible performance impact, depending on the memory configuration. To address this issue, evaluate the query memory requirements, split the query into smaller queries, and materialize duplicate blocks.
>2000 expressions found in WHERE clause of a single query.	This is a hard limit enforced by Impala. The query fails if it contains >2000 expressions. To address this issue, consolidate expressions by replacing repetitive sequences with single operators like IN or BETWEEN.

Potential SQL Issue	Impact and Recommendation
Cartesian or CROSS join found.	Performance impact if tables are large. To address this issue, rewrite the query to add join conditions and eliminate Cartesian joins.
High cardinality GROUP BY column found.	Possible performance impact, depending on the number of distinct groups and the memory configuration. To address this issue, evaluate the memory requirements for the query.
Joins across large tables found.	Possible performance impact, depending on the partitioning keys, filter and join conditions that are specified in the query. To determine the cause, evaluate the EXPLAIN output on Impala. To address this issue, evaluate the filter conditions, join conditions, and the memory requirements for the query and for possible table partitioning strategies.
Join on a large table found.	Possible performance impact, depending on the partitioning keys, filter and join conditions that are specified in the query. To determine the cause, evaluate the EXPLAIN output on Hive or Impala. To address this issue, evaluate the filter conditions, the join conditions, the memory requirements for the query, and the possible table partitioning strategies.
Many single-row inserts found.	Possible performance impact because singleton inserts create too many small files instead of creating fewer large files. To address this issue, batch inserts together to prevent creating too many small data files.
Popular CASE expression across queries found.	Possible performance improvement. Consider materializing the CASE expression.
Popular filter conditions found.	Possible performance impact if the tables are large and are not partitioned. To address this issue, evaluate the possible table partitioning strategies on these filter conditions.
Popular inline views across queries found.	Possible performance impact, depending on the memory configuration, especially if complex expressions are used in inline views on Impala. To address this issue, consider materializing the inline view.
Popular subqueries across queries found.	Possible performance improvement. Consider materializing the subqueries.
Query has no filters.	Possible performance impact if the result set that is returned is very large. To address this issue, rewrite the query to add filtering conditions and to reduce the size of the result set that is returned.
Query on partitioned table is missing filters on partitioning columns.	Possible performance impact if the tables are large. To address this issue, rewrite the query to add filtering conditions.
Query with filter conditions on a large table found.	Possible performance impact if the tables are large and are not partitioned. To address this issue, evaluate the possible table partitioning strategies on these filter conditions.
Query with inline views found.	Possible performance impact, depending on the memory configuration, especially if complex expressions are used in inline views on Impala. To address this issue, if the inline view is duplicated, evaluate whether materializing the inline view is advantageous.
Table might contain too many partitions (>30K).	Can crash the Hive metastore. To address this issue, re-evaluate the partitioning key strategy. Queries that access many partitions might not complete.
Table might contain too many partitions (>50K).	Can crash the Hive metastore. To address this issue, re-evaluate the partitioning key strategy. Queries that access many partitions might not complete.
Table might contain too many partitions (>100K).	Can crash the Hive metastore. To address this issue, re-evaluate the partitioning key strategy. Queries that access many partitions might not complete.
Unsupported commands: UPDATE or DELETE.	UPDATE and DELETE are not supported on CDH. These queries fail. To address this issue, rewrite the query to use views or partitioning strategies that mimic UPDATE and DELETE. Use of partition-based INSERT with OVERWRITE or using SELECT with views over main and delta tables are common workaround strategies.