

# Teradata Vantage™ - Machine Learning Engine

## User Guide

---

Release ML Engine 9.02, 9.2.3, 9.2.4; ML Engine Analytic Functions 9.02




May 2022

# Copyright and Trademarks

Copyright © 2020 - 2022 by Teradata. All Rights Reserved. Portions of this document are based on materials under license from Simba Technologies Inc. (Copyright ©2020 by Simba Technologies Inc.)

All copyrights and trademarks used in Teradata documentation are the property of their respective owners. For more information, see [Trademark Information](#).

## Product Safety

Safety type	Description
 <b>NOTICE</b>	Indicates a situation which, if not avoided, could result in damage to property, such as to equipment or data, but not related to personal injury.
 <b>CAUTION</b>	Indicates a hazardous situation which, if not avoided, could result in minor or moderate personal injury.
 <b>WARNING</b>	Indicates a hazardous situation which, if not avoided, could result in death or serious personal injury.

## Third-Party Materials

Non-Teradata (i.e., third-party) sites, documents or communications ("Third-party Materials") may be accessed or accessible (e.g., linked or posted) in or in connection with a Teradata site, document or communication. Such Third-party Materials are provided for your convenience only and do not imply any endorsement of any third party by Teradata or any endorsement of Teradata by such third party. Teradata is not responsible for the accuracy of any content contained within such Third-party Materials, which are provided on an "AS IS" basis by Teradata. Such third party is solely and directly responsible for its sites, documents and communications and any harm they may cause you or others.

## Warranty Disclaimer

**Except as may be provided in a separate written agreement with Teradata or required by applicable law, the information available from the Teradata Documentation website or contained in Teradata information products is provided on an "as-is" basis, without warranty of any kind, either express or implied, including the implied warranties of merchantability, fitness for a particular purpose, or noninfringement.**

The information available from the Teradata Documentation website or contained in Teradata information products may contain references or cross-references to features, functions, products, or services that are not announced or available in your country. Such references do not imply that Teradata Corporation intends to announce such features, functions, products, or services in your country. Please consult your local Teradata Corporation representative for those features, functions, products, or services available in your country.

The information available from the Teradata Documentation website or contained in Teradata information products may be changed or updated by Teradata at any time without notice. Teradata may also make changes in the products or services described in this information at any time without notice.

## Machine-Assisted Translation

Certain materials on this website have been translated using machine-assisted translation software/tools. Machine-assisted translations of any materials into languages other than English are intended solely as a convenience to the non-English-reading users and are not legally binding. Anybody relying on such information does so at his or her own risk. No automated translation is perfect nor is it intended to replace human translators. Teradata does not make any promises, assurances, or guarantees as to the accuracy of the machine-assisted translations provided. Teradata accepts no responsibility and shall not be liable for any damage or issues that may result from using such translations. Users are reminded to use the English contents.

## Feedback

To maintain the quality of our products and services, e-mail your comments on the accuracy, clarity, organization, and value of this document to: [docs@teradata.com](mailto:docs@teradata.com).

Any comments or materials (collectively referred to as "Feedback") sent to Teradata Corporation will be deemed nonconfidential. Without any payment or other obligation of any kind and without any restriction of any kind, Teradata and its affiliates are hereby free to (1) reproduce, distribute, provide access to, publish, transmit, publicly display, publicly perform, and create derivative works of, the Feedback, (2) use any ideas, concepts, know-how, and techniques contained in such Feedback for any purpose whatsoever, including developing, manufacturing, and marketing products and services incorporating the Feedback, and (3) authorize others to do any or all of the above.

# Contents

<b>Chapter 1: Introduction to Machine Learning Engine</b>	<b>4</b>
ML Engine Feature Summary	4
ML Engine Storage	5
Teradata Vantage Execution Flow	5
Teradata QueryGrid Connector Properties	7
ML Engine Backup and Restore	10
<b>Chapter 2: Management and Monitoring</b>	<b>11</b>
User Management	11
System Monitoring	11
Query Monitoring	12
Concurrent Query Limit	21
Event Management	21
<b>Chapter 3: Analytic Functions</b>	<b>23</b>
ML Engine Analytic Functions	23
<b>Chapter 4: Data Type Mapping</b>	<b>66</b>
Data Type Mapping between Advanced SQL Engine and ML Engine	66
<b>Appendix A: Error Messages</b>	<b>80</b>
<b>Appendix B: Additional Information</b>	<b>90</b>

# Introduction to Machine Learning Engine

Teradata Vantage™ is our flagship analytic platform offering, which evolved from our industry-leading Teradata® Database. Until references in content are updated to reflect this change, the term Teradata Database is synonymous with Teradata Vantage.

Machine Learning Engine (ML Engine) provides a suite of machine learning analytic functions to perform analytics on your full dataset. It supports use cases that require training the model with full dataset, without giving up end-to-end performance of an analytic workload.

ML Engine uses SQL-MapReduce and Collaborative Planning to enable Multi-Genre Advanced Analytics™. It is designed to work with a combination of structured, multi-structured, and unstructured data (for example, text, numerical, tabular, files, CLOB, and BLOB data types) to perform Path and Pattern, Text, Machine Learning, and Graph analytics in the same workload.

ML Engine includes two engines:

Engine	Description
Machine Learning engine	Provides more than 100 pre-built analytic functions for Path, Pattern, Statistical, and Text analytics to solve a range of business problems.
Graph engine	Provides a set of functions that discover relationships between people, products, and processes within a network.

## ML Engine Feature Summary

### Containerized Metadata Persistence

The Containerized Metadata Persistence (CMDP) feature preserves metadata so that recovery from container or pod failures is possible by restarting the cluster, without the need to reinstall analytic functions on restart.

### Failure Detection Recovery

Failure Detection Recovery (FDR) detects failures and attempts to recover from those failures automatically, thereby reducing cluster downtime.

### Function Execution

ML Engine function execution feature enables Advanced SQL Engine to run functions that are remotely available on ML Engine.

## **In Memory Analytics Tables**

The In Memory Analytics Tables (IMAT) enhance performance by keeping data in memory and reducing data copies.

## **Kubernetes**

Kubernetes is a container orchestration platform. The applications that comprise ML Engine run in containers. Kubernetes launches and operates the container infrastructure that makes up the ML Engine.

## **Query Monitoring**

Query monitoring helps administrators measure compute resource utilization. Query monitoring is useful in identifying expensive phases of a query that tax certain resources.

## **Event Management Events**

Event Management (EM) events for ML Engine help administrators detect and manage situations where there is cluster degradation and down incidents.

## **System Monitoring**

Teradata Vantage can monitor ML Engine worker node (pod) resource consumption, correct any resource bottlenecks, and pinpoint long-term and cyclical utilization trends, as well as set health thresholds.

## **Workload Management**

Workload Management (WLM) manages workload performance by prioritizing workload requests, monitoring system activity, and acting when predefined limits are reached.

# **ML Engine Storage**

ML Engine storage includes local spool for temporary data.

## **Temporary Data Storage**

The local spool attached to the worker node (pod) temporarily stores the data moved to and from Advanced SQL Engine storage. Each node holds approximately three terabytes (3 TB) of data. This approach enables fast read and write access to data for analysis. The local spool is a storage array equipped with Solid State Drives (SSDs) and employs RAID1 to improve data availability through hardware redundancy.

The stored data in this temporary storage is non-persistent, unprotected, and is cleared at the end of an Advanced SQL Engine session.

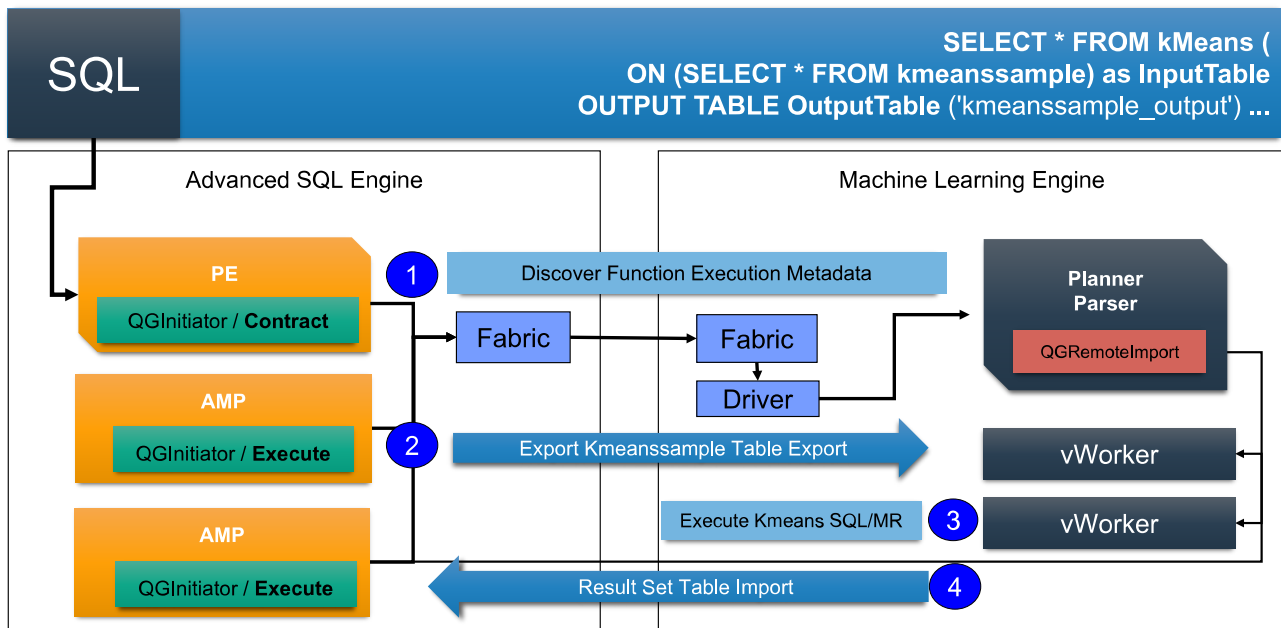
# **Teradata Vantage Execution Flow**

This section explains how SQL Engine and Machine Learning Engine work together to process a SQL request.

This diagram depicts the execution flow in Teradata Vantage, when Advanced SQL Engine receives this sample SQL request:

```
SELECT * FROM kMeans (
ON (SELECT * FROM kmeanssample) as InputTable
OUTPUT TABLE outputTable ('kmeanssample_output')
...
```

## Teradata Vantage Execution Flow



### 1. Contract Phase: Advanced SQL Engine receives the SQL request.

In Advanced SQL Engine, the Parsing Engine (PE) is a virtual processor (vproc) that parses the SQL code of the query and sends a metadata message through QueryGrid fabric to the Planner Parser in ML Engine to discover metadata for function execution.

The metadata message contains information about the function, such as the function name, version, type, and description. It also includes the function parameters that specify the input and output tables and the input columns.

**Note:**

In the KMeans function, used to perform k-means clustering on a dataset, the metadata message defines the following tables:

- Required input table "InputTable" containing the list of features for clustering data
- Optional input table "CentroidsTable" containing the initial seed means for the clusters
- Required output table "OutputTable" in which to output the centroids of the clusters
- Optional output table "ClusteredOutput" in which to store the clustered output

Depending on the function, the metadata message may contain other parameters.

In ML Engine, the Planner Parser creates InputTable based on the metadata parameters, specifically the column data types, and executes the function against them with specific options from the user. The generated output table with specific data types is then used to create the output table in Advanced SQL Engine.

The Access Module Processors (AMPs) in Advanced SQL Engine then set up connections to the vworkers in ML Engine through the QueryGrid fabric, preparing to export data for analysis.

2. Execute phase: Advanced SQL Engine exports the table to ML Engine for processing.

When Advanced SQL Engine receives the function execution metadata from ML Engine, the QueryGrid connector makes data type mapping and conversions, and initiates the export of the kmeanssample table from the AMPs through the QueryGrid fabric to the vworkers in ML Engine.

ML Engine temporarily stores the data for function execution.

3. Execute phase: ML Engine executes the KMeans function.
4. Execute phase: Advanced SQL Engine imports the result table from ML Engine.

After the function execution completes, Advanced SQL Engine pulls the result table from ML Engine through the QueryGrid fabric.

If additional tables are created as part of the execution output, ML Engine notifies Advanced SQL Engine, and exports the additional tables through separate Advanced SQL Engine sessions.

## Teradata QueryGrid Connector Properties

In Teradata Vantage, the Teradata QueryGrid™ fabric is configured for internal communication. Two Teradata QueryGrid links are created to facilitate communication between Advanced SQL Engine and ML Engine.

QueryGrid Manager shows the properties that you can configure. Vantage requires that some property defaults cannot be changed.

**ML Engine Connector Properties**

Name	Default	Description	Connector Type	Change Default
Authentication Mechanism	Trusted	Authentication mechanism used to log on to ML Engine.	Target	No
Column Name Handling	CASE-INSENSITIVE	Controls whether ML Engine will attempt to treat column names as case-insensitive.	Target	Yes
Connection Evict Frequency	1 minute	Frequency of eviction checks. It checks, closes, and removes the connection object from the pool if the idle time (current time - last time of use) of connection object is greater than connectionMaxIdleTime /sessionConnectionMaxIdleTime. It can be reduced if there are multiple concurrent users running queries and due to high demand, the connections should be freed up more frequently for new users to pick up.	Target	Yes
Connection Max Idle Time	1800 seconds	Maximum idle time for connection cache object, after which it is closed and evicted from the cache. It should be used if there are multiple concurrent users/queries running on the system that might lead to starvation of connection objects.	Target	Yes
Connection Pool Size	100	Maximum number of connection objects that can be stored in the connection pool. When trying to acquire a new connection, the connector checks for entry to be made available in the connection pool or else it fails after 5 minutes. There is one connection pool per connector config, including the username. This can be used to throttle the number of connections that can be established by concurrent queries for one user.	Target	Yes
Database Name	beehive	Database name for ML Engine.	Target	No
Default String Size	32000	VARCHAR truncation size (size at which data imported from, or exported to, string columns is truncated). Value represents maximum number of Unicode characters to import.	Initiator, Target	Yes
Enable Blob Support	TRUE	Enables support of BLOB datatype. This property is enabled by default. To disable BLOB support: set foreign server attr = 'servername=coprocessor; enableBlobSupport=false;'for session volatile;	Initiator, Target	Yes
Enable Clob Support	TRUE	Enables support of CLOB datatype. This property is enabled by default. To disable CLOB support: set foreign server attr = 'servername=coprocessor; enableClobSupport=false;'for session volatile;	Initiator, Target	Yes
Enable Logging	INFO	Logging level for connector or link properties. Applies to both initiating and target connector; however, if their levels are different, level for initiating connector takes precedence.	Initiator, Target	Yes
In-Memory Result	TRUE	Whether to use in-memory tables for primary function results imported to the Advanced SQL Engine.	Target	No
In-Memory Input	TRUE	Whether to use in-memory tables for data exported to Advanced SQL Engine for input to functions.	Target	No
Large Blob Handling	ERROR	Action to take if BLOB larger than 10485760 is exported to ML Engine: ERROR causes exception to be raised; TRUNCATE causes data to be truncated to 10485760.		Yes



Name	Default	Description	Connector Type	Change Default
		To change this behavior in Advanced SQL Engine session, include statement such as: set foreign server attr = 'servername=coprocessor; largeBlobHandling=TRUNCATE;' for session volatile;		
Large Clob Handling	ERROR	Action to take if CLOB larger than 10485760 is exported to ML Engine: ERROR causes exception to be raised; TRUNCATE causes data to be truncated to 10485760. To change this behavior in Advanced SQL Engine session, include statement such as: set foreign server attr = 'servername=coprocessor; largeClobHandling=TRUNCATE;' for session volatile;	Initiator, Target	Yes
Link Buffer Count	4	Maximum number of write buffers available on a single channel at one time.	Initiator, Target	Yes
Link Buffer Size	1048576	Maximum size of the write buffers to allocate for row handling and message exchange.	Initiator	Yes
Link Heartbeat Interval	60000	Maximum interval in milliseconds for heartbeat signal on channel between connector and fabric instance, used for health check status. This interval must be greater than Link Handshake Timeout.	Initiator, Target	Yes
Link Handshake Timeout	30000	Handshake timeout in milliseconds for link channel setup.	Initiator, Target	Yes
Max Retry Attempts	1	Maximum retries of certain errors during function execution.	Target	No
Oldest Active Session	24	No connection older than this value (in hours) will be used to begin executing a query on ML Engine.	Target	No
Password		The password for User Name.	Target	Yes
Port	30002	Port number of ML Engine server.	Target	No
Read Timeout	30000	Number of milliseconds to wait to read between data packets when importing data messages.	Initiator, Target	Yes
Regex to limit Retained Tables	.*	Regex applied to the exception toString() value using class java.util.regex.Pattern. The tables are retained only when there is a match. If there is any exception or error applying the regex, the tables will not be retained. This property is applied only when the Retain Tables On Error property is set to TRUE. Use this property only as directed by your Customer Service Representative.	Target	No
Response Timeout	1800000	Number of milliseconds to wait for final data exec response when all data has been transferred.	Initiator, Target	Yes
Retain Tables On Error	FALSE	Check the box to set to true and enable table retention on error. Use this property only as directed by your Customer Service Representative.	Target	No
Reverse Link Name	teradata_link	The name of the link that has ML Engine connector as the initiator and Advanced SQL Engine connector as the target.	Target	Yes
Schema Name	none	Default schema name.	Target	Yes
Server	Configured specifically to	Hostname of ML Engine server.	Target	No

Name	Default	Description	Connector Type	Change Default
	your environment			
Transformformatting	FALSE	Whether to transform array columns to Teradata string format. If FALSE, array columns are returned in JSON format, except those specified in RETURN clause to be returned in Teradata string format.	Initiator, Target	Yes
User Name	beehive	Value to use when initiator does not support mechanism to provide user credentials. Value is also used for connectivity diagnostic checks.	Target	No
Write Timeout	30000	Number of milliseconds to wait to write between data packets when exporting data messages.	Initiator, Target	Yes

## ML Engine Backup and Restore

Backup and restore of ML Engine is available on Teradata IntelliFlex, Vantage on AWS, and Vantage on Azure.

To initiate backup and restore of ML Engine, or to start or stop automated metadata snapshots, open a case using the Teradata Support portal.

# Management and Monitoring

## User Management

### ML Engine User Provisioning on First Access

Users and schemas on ML Engine are automatically created by the system. You do not need to take any action.

### ML Engine User Password Rules

ML Engine user password rules:

- Password length must be at least 1 character and a maximum of 128 characters.
- The password must consist of ASCII characters (foreign language or multibyte characters are not supported).
- Passwords are case sensitive.
- Passwords must not contain:
  - Whitespace (including space, tab, and newline characters).
  - Colon (:), backslash (\), or single-quote apostrophe (') characters.
  - Right parenthesis, left parenthesis, double quote ("), and back quote (`).
  - Control characters (including ASCII 0-31 and 127).

For information about passwords for service accounts, contact your Customer Support Representative.

### ML Engine User Name Rules

Follow these rules when you create a new user:

- Length must not exceed 63 bytes.
- Must start with a letter or underscore (\_) and contain only ASCII characters.
- Must not contain special characters, except for underscore and dollar sign (\$).
- Quoted user names are not supported.
- User names are not case-sensitive.

---

**Note:**

Deviating from these rules could prevent access to ML Engine functions.

---

## System Monitoring

Using Teradata Viewpoint, you can monitor ML Engine pod resource consumption, correct any resource bottlenecks, pinpoint long-term and cyclical utilization trends, and set health thresholds.

System monitoring is on by default and is managed completely from Teradata Viewpoint. To monitor your system, you must first add Machine Learning Engine in the Teradata Viewpoint Monitored Systems portlet.

To configure system health thresholds for CPU, Pod CPU Skew, and Active Sessions, define threshold values in System Health Details in the Monitored Systems portlet.

See *Teradata® Viewpoint User Guide*, B035-2206.

## Failure Detection and Recovery

Failure detection and recovery (FDR) detects failures and attempts to recover from those failures automatically, thereby significantly reducing cluster downtime.

FDR is turned on by default and cannot be disabled.

Alerts are sent to inform you about ongoing ML Engine FDR activities and cluster status. See [Event Management](#).

## Active Query Cancellation

ML Engine monitors free space in each container in all pods, checking disk capacity in the root directory every 30 seconds and canceling live queries when a disk is 85% full. ML Engine does not shut down when disk capacity is reached.

Query cancellation is automated and not configurable.

## Query Monitoring

Query monitoring enables administrators to measure computer resource utilization by individual query. Resource utilization is modeled on a per-analytic query basis. When a query takes a long time to complete, query monitoring provides insights as to why. It is useful to identify expensive phases of a query that tax certain resources or skew work distribution.

The resource utilization data that is collected is stored in tables in Advanced SQL Engine using the existing QueryGrid Remote Connector mechanism and is accessible using Teradata Studio or Database Query Logging (DBQL).

For information about Teradata Studio, see *Teradata Studio™ User Guide*, B035-2041. For information about DBQL, see *Teradata Vantage™ - Database Administration*, B035-1093.

---

### Note:

Query monitoring reports only ML Engine resource utilization while Teradata Vantage queries are running and is best suited to measuring long-running queries (several seconds to hours).

---

## Enabling or Disabling Query Monitoring

Query monitoring is enabled by default. To disable it, contact your Teradata Support Representative.

## Output Storage

Resource utilization statistics are stored in tables in Advanced SQL Engine. You can access these tables by using Teradata Studio or DBQL.

Table Name	Description
td_mle_db.QueryLog	Contains query level statistics by global Teradata Vantage ID.
td_mle_db.QueryLogSQL	Contains query level error and text summary for ML Engine.
td_mle_db.ResUsageNetwork	Contains system level statistics for ML Engine network usage.
td_mle_db.ResUsageCpu	Contains system level statistics for ML Engine CPU usage.
td_mle_db.ResUsageDisk	This table is created but currently does not contain data.
td_mle_db.ResUsageMemory	Contains system level statistics for ML Engine memory usage.

See [System Monitoring](#).

### Note:

If stats movement is disabled, these tables will be empty:

- td\_mle\_db.ResUsageNetwork
- td\_mle\_db.ResUsageCpu
- td\_mle\_db.ResUsageMemory
- td\_mle\_db.ResUsageDisk

## Table Column Mapping

The following tables detail the mapping of columns between Advanced SQL Engine and ML Engine.

### td\_mle\_db.QueryLog

This is a summary table for each ML Engine analytic query.

Advanced SQL Engine Column	Description	ML Engine Column	Source
CollectTimestamp	Time when statistics were collected for this row.	nc_system.nc_all_statement_stats.collecttimestamp	

Advanced SQL Engine Column	Description	ML Engine Column	Source
PodName	ML Engine pod name.	nc_system.nc_all_statement_stats.podname	
UUID	Advanced SQL Engine universally unique query ID.	nc_system.nc_all_statement_stats.uuid	Joined from beehive_statement table.
NumPhases	Total ML Engine phases involved in this query.	nc_system.nc_all_statement_stats.numphases	Joined from beehive_statement_phase table.
ReadIOCount	Total read I/O operations (read syscalls).	nc_system.nc_all_statement_stats.readiount	Aggregation of /proc/[pid]/io/syscr field value, the number of read I/O operations – that is, system calls such as read(2) and pread(2) value for all arc and runner processes.
WriteIOCount	Total write I/O operations (write syscalls).	nc_system.nc_all_statement_stats.writeiount	Aggregation of /proc/[pid]/io/syscw field value, the number of write I/O operations – that is, system calls such as write(2) and pwrite(2) value for all arc and runner processes.
ReadBytes	Total number of bytes fetched from storage layer (read_bytes).	nc_system.nc_all_statement_stats.readbytes	Aggregation of /proc/[pid]/io/read_bytes field value, the number of bytes which a process really did cause to be fetched from the storage layer, for all arc and runner processes.
WriteBytes	Total number of bytes sent from storage layer (write_bytes).	nc_system.nc_all_statement_stats.writebytes	Aggregation of /proc/[pid]/io/write_bytes field value, the number of bytes that a process caused to be sent to the storage layer, for all arc and runner processes.

Advanced SQL Engine Column	Description	ML Engine Column	Source
ReadRateBytesPerSec	Number of bytes that storage layer read for each second.	nc_system.nc_all_statement_stats.writeatebytespersec	Delta of ReadBytes with $\text{STATS\_UPDATE\_INTERVAL\_SEC} \times 100$ $/ \text{STATS\_UPDATE\_INTERVAL\_SEC}$ where Delta of ReadBytes = ReadBytes column value at current time - ReadBytes column value at the time before $\text{STATS\_UPDATE\_INTERVAL\_SEC}$
WriteRateBytesPerSec	Number of bytes that storage layer wrote for each second.	nc_system.nc_all_statement_stats.writeratebytespersec	Delta of WriteBytes with $\text{STATS\_UPDATE\_INTERVAL\_SEC} \times 100$ $/ \text{STATS\_UPDATE\_INTERVAL\_SEC}$ where Delta of WriteBytes = WriteBytes column value at current time - WriteBytes column value at the time before $\text{STATS\_UPDATE\_INTERVAL\_SEC}$
CpuPercent	Total CPU percentage for this query.	nc_system.nc_all_statement_stats.cpupercent	Delta of stime+utime with $\text{STATS\_UPDATE\_INTERVAL\_SEC} \times 100$ $/ (\text{STATS\_UPDATE\_INTERVAL\_SEC} \times \text{numCores})$ where Delta of stime+utime = (stime+utime) value from /proc/[pid]/stat at current time - (stime+utime) value from /proc/[pid]/stat at the time before $\text{STATS\_UPDATE\_INTERVAL\_SEC}$

Advanced SQL Engine Column	Description	ML Engine Column	Source						
CpuSecs	Total CPU seconds for vworkers and runners.	nc_system.nc_all_statement_stats.cpusecs	Aggregation of /proc/[pid]/stat/stime + utime for all arc and runner processes. /proc/[pid]/stat/stime field is the amount of time that a process has been scheduled in kernel mode, measured in clock ticks (divide by sysconf(_SC_CLK_TCK)). /proc/[pid]/stat/utime field is the amount of time that a process has been scheduled in user mode, measured in clock ticks divide by sysconf(_SC_CLK_TCK)).						
MemoryKb	Total memory taken by this query.	nc_system.nc_all_statement_stats.memorykb	Aggregation /proc/[pid]/stat/rss field value, multiplied by pageSizeKB for all arc and runner processes where pageSizeKB = getpagesize()/1024. /proc/[pid]/stat/rss field is Resident Set Size: number of pages the process has in real memory. This is just the pages that count toward text, data, or stack space. This does not include pages that have not been demand-loaded in, or that are swapped out.						
ErrorCode	Query result code: <table><tr><th>Code</th><th>Meaning</th></tr><tr><td>1</td><td>Canceled</td></tr><tr><td>2</td><td>Succeeded</td></tr></table>	Code	Meaning	1	Canceled	2	Succeeded	nc_system.nc_all_statement_stats.errorcode	Joined from beehive_statement table.
Code	Meaning								
1	Canceled								
2	Succeeded								



Advanced SQL Engine Column	Description	ML Engine Column	Source								
	<table><tr><th>Code</th><th>Meaning</th></tr><tr><td>3</td><td>Error</td></tr></table>	Code	Meaning	3	Error						
Code	Meaning										
3	Error										
Cancellable	<table><tr><th>Code</th><th>Meaning</th></tr><tr><td>1</td><td>Canceled</td></tr><tr><td>2</td><td>Succeeded</td></tr><tr><td>3</td><td>Error</td></tr></table>	Code	Meaning	1	Canceled	2	Succeeded	3	Error	nc_system.nc_all_statement_stats.cancellable	Joined from beehive_statement table.
Code	Meaning										
1	Canceled										
2	Succeeded										
3	Error										
SessionId	Unique sessionid. No correlation to Advanced SQL Engine or Teradata QueryGrid session IDs.	nc_system.nc_all_statement_stats.sessionid	Joined from beehive_statement table.								
ParentSessionId	Parent session ID. Empty for nonchild queries.	nc_system.nc_all_statement_stats.parentsessionid	Joined from beehive_statement table.								
QueryId	Unique ML Engine query ID.	nc_system.nc_all_statement_stats.queryid	Joined from beehive_statement table.								
StartTime	Timestamp when query was submitted.	nc_system.nc_all_statement_stats.starttime	Joined from beehive_statement table.								
EndTime	Timestamp when query finished.	nc_system.nc_all_statement_stats.endtime	Joined from beehive_statement table.								
ParentQueryId	The parent query id.	nc_system.nc_all_statement_stats.parentqueryid	Joined from beehive_statement table.								
WorkloadName	The workload name this query is mapped to.	nc_system.nc_all_statement_stats.workloadname	Joined from _bee_special.nc_qos_workload_transition table.								
ServiceClassName	The workload class this query is mapped to.	nc_system.nc_all_statement_stats.serviceclassname	Joined from _bee_special.nc_qos_workload_transition table.								
Priority	The workload priority.	nc_system.nc_all_statement_stats.priority	Joined from _bee_special.nc_qos_workload_transition table.								

Advanced SQL Engine Column	Description	ML Engine Column	Source
Weight	The workload weight.	nc_system.nc_all_statement_stats.weight	Joined from _bee_special.nc_qos_workload_transition table.

### **td\_mle\_db.queryLogSQL**

This table is a summary table for ML Engine query error and text.

Advanced SQL Engine Column	Description	ML Engine Column	Source
CollectTimeStamp	Time when statistics were collected for this row.	nc_system.nc_all_statement_stats.collecttimestamp	
QueryId	Unique ML Engine query ID.	nc_system.nc_all_statement_stats.queryid	Joined from beehive_statement table.
QueryText	ML Engine query text. Truncation may occur.	nc_system.nc_all_statement_stats.querytext	Joined from beehive_statement table.
ErrorText	Cumulative number of bytes received over network.	nc_system.nc_all_statement_stats.errortext	Joined from beehive_statement table.

### **td\_mle\_db.ResUsageNetwork**

This table contains a row for each time ML Engine collects the networking counts.

Advanced SQL Engine Column	Description	ML Engine Column	Source
CollectTimeStamp	Time when statistics were collected for this row.	nc_system.nc_all_resource_usage_network_stats.collecttimestamp	Prometheus metrics are collected every 60 seconds.
PodName	ML Engine pod name.	nc_system.nc_all_resource_usage_network_stats.podname	
NodeId	ML Engine node where pod is running.	nc_system.nc_all_resource_usage_network_stats.nodeid	
ReadBytes	Cumulative number of bytes received over network.	nc_system.nc_all_resource_usage_network_stats.readbytes	Prometheus

Advanced SQL Engine Column	Description	ML Engine Column	Source
WriteBytes	Cumulative number of bytes sent over network.	nc_system.nc_all_resource_usage_network_stats.writebytes	Prometheus
ReadRateBytesPerSec	Number of bytes received over network for each second.	nc_system.nc_all_resource_usage_network_stats.readratebytespersec	Prometheus
ReadErrorsRateSecs	Number of errors while receiving over network for each second.	nc_system.nc_all_resource_usage_network_stats.readerrorsratesecs	Prometheus
WriteRateBytesPerSec	Number of bytes sent over network for each second.	nc_system.nc_all_resource_usage_network_stats.writeratebytespersec	Prometheus
WriteErrorsRateSecs	Number of errors while sending over network for each second.	nc_system.nc_all_resource_usage_network_stats.writeerrorsratesecs	Prometheus

### td\_mle\_db.ResUsageCpu

This table contains a row for each time ML Engine collects the CPU counts.

Advanced Engine Column	Description	ML Engine Column	Source
CollectTimeStamp	Time when statistics were collected for this row.	nc_system.nc_all_resource_usage_cpu_stats.collecttimestamp	
PodName	ML Engine pod name.	nc_system.nc_all_resource_usage_cpu_stats.podname	
NodeId	ML Engine node identifier.	nc_system.nc_all_resource_usage_cpu_stats.nodeid	
CpuIOWaitCentiSecs	Time in centiseconds CPU is waiting for I/O completion.	nc_system.nc_all_resource_usage_cpu_stats.cpuiowaitcentisecs	<p>Aggregated value of Prometheus Node Exporter iowait time for all CPU cores for NodeId.</p> <p>Retrieved from Prometheus Node Exporter.</p> <p>Prometheus Node Exporter retrieves its values from /proc /stat/iowait field.</p> <p>The time waiting for I/O to complete. This value is not reliable for these reasons:</p>

Advanced Engine Column	Description	ML Engine Column	Source
			<ul style="list-style-type: none"> <li>The CPU does not wait for I/O to complete; iowait is the time that a task is waiting for I/O to complete. When a CPU goes into idle state for outstanding task I/O, another task will be scheduled on this CPU.</li> <li>On a multi-core CPU, the task waiting for I/O to complete is not running on any CPU, so the iowait on each CPU is difficult to calculate.</li> <li>The value in this field may decrease in certain conditions.</li> </ul>
CpuPercent	Total CPU percentage for this node	nc_system.nc_all_resource_usage_cpu_stats.cpubercent	Prometheus

### **td\_mle\_db.ResUsageDisk**

This table is created but is empty.

Advanced SQL Engine Column	Description	ML Engine Column
CollectTimeStamp	Time when statistics were collected for this row.	nc_system.nc_all_resource_usage_disk_stats.collecttimestamp
PodName	ML Engine pod name.	nc_system.nc_all_resource_usage_disk_stats.podname
NodeId	ML Engine node identifier.	nc_system.nc_all_resource_usage_disk_stats.nodeid

### **td\_mle\_db.ResUsageMemory**

This table contains a row for each time ML Engine collects memory counts.

Advanced SQL Engine Column	Description	ML Engine Column	Source
CollectTimeStamp	Time when statistics were collected for this row.	nc_system.nc_all_resource_usage_memory_stats.collecttimestamp	

Advanced SQL Engine Column	Description	ML Engine Column	Source
PodName	ML Engine pod name.	nc_system.nc_all_resource_usage_memory_stats.podname	
NodeId	ML Engine node identifier.	nc_system.nc_all_resource_usage_memory_stats.nodeid	
TotalMemoryUsage	Total memory usage.	nc_system.nc_all_resource_usage_memory_stats.totalmemoryusage	Prometheus
PageFaultsRateSec	Number of page faults for each second.	nc_system.nc_all_resource_usage_memory_stats.pagefaultsrate	Prometheus

## Concurrent Query Limit

The default maximum concurrency for ML Engine jobs is 10. For Vantage on public cloud deployment platforms, the default maximum concurrency is 5. These maximum concurrencies have been tested with a wide variety of workloads and are shown to be stable and optimal for most workloads. However, if after viewing the node utilization for the Analytic Nodes, you determine you need to increase the limit, the maximum limit in Teradata Active System Management (TASM) can be raised. For information, see *Teradata Vantage™ - Workload Management User Guide*.

## Event Management

Event Management (EM) helps administrators detect and manage situations where the system is affected by cluster degradation and down incidents.

Applications send events to Server Management using Teradata Vital Infrastructure (TVI). Depending on configuration, Server Management either logs the error or generates an EM event and sends the event to Teradata Customer Support. See *Teradata® Server Management Product Guide*, B035-6112.

### EM Events for ML Engine

MessageID	Severity	Description
TDMLEDiskFull65Percent	Warning	65% of ML Engine container disk space was consumed.
TDMLEDiskFull80Percent	Warning	80% of ML Engine container disk space was consumed.
TDMLEDiskFull90Percent	Critical	90% of ML Engine container disk space was consumed.
TDMLEFailedToUpdatedQG	Critical	ML Engine failed to update QueryGrid Manager about the Queen pod location.
TDMLEFdrReachedMaxAttempts	Critical	FDR reached maximum attempts.
TDMLEFdrTimeout	Warning	FDR sequence failed due to timeout.

MessageID	Severity	Description
TDMLEPredictDiskFull	Critical	ML Engine container disk space exhausts soon.
TDMLESystemUp	Warning	Beehive is fine. Replication factor is one and ML Engine is up and running.
TDMLERFZero	Critical	The replication factor is zero and ML Engine is down.

# Analytic Functions

## ML Engine Analytic Functions

ML Engine offers a large suite of machine learning functions for time series, graph, statistical, text, association, clustering, pattern matching, and path analysis. These analytic functions are preinstalled and ready to use on Teradata Vantage.

See *Teradata Vantage™ Machine Learning Engine Analytic Function Reference*, B700-4003.

## ML Engine Analytic Function Execution

ML Engine analytic functions run on ML Engine, accessing data that resides on Advanced SQL Engine.

To run an ML Engine analytic function, submit a SQL query that references one or more ML Engine functions. The function execution is specified in the FROM clause as a table expression. The exact syntax is specified in [ML Engine Analytic Function Execution SQL Syntax](#).

These steps performed during function execution:

1. All required data is transferred from Advanced SQL Engine to ML Engine using Teradata QueryGrid™.
2. The ML Engine analytic function is run on ML Engine.
3. Results are returned to Advanced SQL Engine.

For some analytic functions, additional secondary output tables are also returned to Advanced SQL Engine.

4. All intermediate data used in executing ML Engine analytic function is removed from ML Engine.

## ML Engine Analytic Function Execution SQL Syntax

```
SELECT select_options
FROM {
  table_name [ [ AS ] correlation_name ] |
  join |
  ( subquery ) [ AS ] derived_table_name [ ( column_name_list ) ] |
  table_function |
  table_operator |
  analytic_function [ AS ] correlation_name
} [, ...]
other_options
```

## Syntax Elements

### ***select\_options***

See the description of SELECT statements in *Teradata Vantage™ - SQL Data Manipulation Language*.

### ***table\_name***

See the description of SELECT statements in *Teradata Vantage™ - SQL Data Manipulation Language*.

### ***correlation\_name***

See the description of SELECT statements in *Teradata Vantage™ - SQL Data Manipulation Language*.

### ***subquery***

See the description of SELECT statements in *Teradata Vantage™ - SQL Data Manipulation Language*.

### ***derived\_table\_name***

See the description of SELECT statements in *Teradata Vantage™ - SQL Data Manipulation Language*.

### ***column\_name\_list***

See the description of SELECT statements in *Teradata Vantage™ - SQL Data Manipulation Language*.

### ***analytic\_function***

The name of the ML Engine function to be run. See *Teradata Vantage™ Machine Learning Engine Analytic Function Reference*, B700-4003.

```
analytic_function_name (on_clauses [ output_table ... ] [ USING
function_arguments ] )
```

### ***on\_clauses***

```
on_clause [ on_clause... ]
```

### ***on\_clause***

```
{ partition_any_input | partition_attributes_input | dimensional_input }
```



Provides the input data on which the function operates. This data is composed of one or more partitioned inputs and zero or more dimensional inputs.

#### ***partition\_any\_input***

```
table_input [ PARTITION BY ANY [ ORDER BY order_expr_list ] ]
```

Expressions that partitions the input data before the function operates on it.

#### ***partition\_attributes\_input***

```
table_input PARTITION BY expr_list [ ORDER BY order_expr_list ]
```

Expressions that partition the input data before the function operates on it.

#### ***dimensional\_input***

```
table_input DIMENSION [ ORDER BY order_expr_list ]
```

Expressions that replicates the input data to all nodes before the function operates on it.

#### ***table\_input***

```
ON table_expression [ [ AS ] alias ]
```

Includes an alias for the *table\_expression*. See the description of rules for table alias in the *Teradata Vantage™ Machine Learning Engine Analytic Function Reference*, B700-4003.

#### ***order\_expr\_list***

```
order_expr [ , ... ]
```

SQL expression on which to sort rows going into the function. The expression can specify the numeric position of the expression in the expression list with a name or constant.

#### ***order\_expr***

```
expression [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ]
```

Each *order\_expr* can be specified to sort either ascending (ASC) or descending (DESC). Ascending order is the default. Each *order\_expr* can specify whether null values appear before (NULLS FIRST) or after (NULLS LAST) non-null values. By default, null values sort

as if larger than non-null values (NULLS FIRST is default for DESC order and NULLS LAST is default for ASC order).

#### ***table\_expression***

```
{ table_name | (subquery) | analytic_function }
```

#### ***table\_name***

Name of a table, queue table, derived table, or view.

#### ***output\_table***

```
OUT TABLE alias (table_name)
```

#### ***subquery***

A SELECT expression that is nested within another SQL statement or expression.

#### ***other\_options***

See the description of SELECT statements in *Teradata Vantage™ - SQL Data Manipulation Language*.

See *Teradata Vantage™ Machine Learning Engine Analytic Function Reference*, B700-4003 for information on semantic requirements for ML Engine analytic functions.

## **Considerations**

- Column names must not be longer than 63 characters. If columns names exceed 63 characters, unpredictable errors and behavior may occur.
- A query alias name is required for analytic function invocation. In this example, dt is the query alias:

```
SELECT * FROM GLM (ON ...) AS dt;
```

- Function alias names must correspond to the coprocessor function names for the function. See *Teradata Vantage™ Machine Learning Engine Analytic Function Reference*, B700-4003 and [Function Alias](#).
- The USING keyword is required at the start of the function argument section.
- Output tables must be prefaced with OUT TABLE and the alias as required by the syntax of the specific function. See *Teradata Vantage™ Machine Learning Engine Analytic Function Reference*, B700-4003.

- Input specified with an ON clause must include any required PARTITION BY, DIMENSION, or ORDER BY clause. See *Teradata Vantage™ Machine Learning Engine Analytic Function Reference*, B700-4003.
- The SQL identifier must not be an Advanced SQL Engine reserved word. Advanced SQL Engine restricted words can be located by using the `SELECT * FROM SYSLIB.SQLRestrictedWords` command. See *Teradata Vantage™ - SQL Fundamentals*, B035-1141. If an identifier is a reserved keyword, it must be enclosed in double quotes.
- The length of a query output alias must not exceed 25 characters. For example, this alias causes an error:

```
SELECT * FROM KMeans (
  ON (SELECT * FROM kmeansample) as InputTable
  USING
  OutputTable ('kmeanssample_centroid')
  NumberK ('3')
  Threshold ('0.01')
  MaxIterNum ('10')
) AS Long_Function_Name_Alias_Example;
```

- In the `output_table` clause (see **output\_table** in [ML Engine Analytic Function Execution SQL Syntax](#)), `table_name` must not have an embedded double quote or an embedded period.
- The first column of the SELECT statement cannot be of data type CLOB or BLOB. If it is, error 5660 is generated:

```
*** Failure 5660 Cannot create index on LOB columns.
```

In some functions, the first output column is appended from the input table (through Accumulate or PartitionColumns arguments). For such cases, either choose a column with a different data type or change the order of columns in the SELECT statement. For example:

```
SELECT col2,col1 FROM func_name
```

instead of:

```
SELECT * FROM func_name
```

or

```
SELECT col1,col2 FROM func_name
```

Another option is to create an output table from the SELECT statement with NO PRIMARY INDEX or an index on a column which is not a LOB column.

## Examples

### Function with Multiple ON Clauses

This function, Attribution\_MLE, has two partitioned inputs and five dimensional inputs. When this SQL statement runs, all seven input tables are transferred to ML Engine. The Attribution\_MLE function is then called using the transferred tables as inputs and with the function arguments in the SQL statement. Finally, the result of the Attribution\_MLE function is returned to Advanced SQL Engine for use in processing the rest of the SQL statement.

```
SELECT * FROM Attribution_MLE (
  ON attribution_sample_table1 AS input1 PARTITION BY user_id
                                     ORDER BY time_stamp
  ON attribution_sample_table2 AS input2 PARTITION BY user_id
                                     ORDER BY time_stamp
  ON conversion_event_table AS conversion DIMENSION
  ON excluding_event_table AS excluding DIMENSION
  ON optional_event_table AS optional DIMENSION
  ON model1_table AS model1 DIMENSION
  ON model2_table AS model2 DIMENSION
  USING EventColumn ('event')
      TimestampColumn ('time_stamp')
      WindowSize ('rows:10&seconds:20')
) AS attrtb ORDER BY user_id, time_stamp;
```

### Function with OUT TABLE Clauses

This function, KMeans, also has multiple ON clauses, but no ON clause has a PARTITION BY or DIMENSION clause. OUT TABLE clauses specify additional (secondary) output tables (those other than the primary output table). When this SQL statement runs, the results of the ON clauses are transferred to ML Engine. The KMeans function is then called using the transferred tables as inputs and with the function arguments provided in the SQL statement. The secondary outputs of the KMeans function are returned to Advanced SQL Engine in the kmeanssample\_output and kmeanssample\_clusteredoutput tables as specified in the function arguments. Finally, the result of the KMeans function is returned to Advanced SQL Engine for use in processing the rest of the SQL statement.

```
SELECT * FROM KMeans (
  ON computers_train1 AS InputTable
  ON kmeanssample_centroid AS CentroidsTable
  OUT TABLE OutputTable (kmeanssample_output)
  OUT TABLE ClusteredOutput (kmeanssample_clusteredoutput)
  USING
```

```
MaxIterNum ('10')
) AS kmouttb2;
```

## Nested Functions

This example calls nested functions. The output of `SentimentExtractor` is used as the input to `SentimentEvaluator`. When this SQL statement is executed, the input table for `SentimentExtractor` is transferred to ML Engine. The `SentimentExtractor` function is then called using the transferred table as input and with the function arguments provided in the SQL statement. The result of the `SentimentExtractor` function is then returned to Advanced SQL Engine. This intermediate result is transferred to ML Engine. Then the `SentimentEvaluator` function is called using the transferred intermediate result as input and with the function arguments provided in the SQL statement. Finally, the result of the `SentimentEvaluator` function is returned to Advanced SQL Engine for use in processing the rest of the SQL statement. See [Nested ML Engine Analytic Functions](#). SQL identifiers that are reserved keywords must be enclosed in double quotation marks.

```
SELECT * FROM SentimentEvaluator (
  ON (
    SELECT * FROM SentimentExtractor (
      ON sentiment_extract_input as "input"
      USING TextColumn ('review')
      Accumulate ('category')
      "model" ('dictionary')
    ) AS dt1
  ) AS "input" PARTITION BY 1
  USING ObsColumn ('category')
  SentimentColumn ('out_polarity')
) AS dt;
```

## Identifiers

There are differences in the way Advanced SQL Engine and ML Engine interpret identifiers:

Identifier	Advanced SQL Engine Interpretation	ML Engine Interpretation
Defining regular identifiers. For example, <code>mixedCaseIdentifier</code> .	Case is preserved: <code>mixedCaseIdentifier</code> .	Case is not preserved. The identifier is stored as lowercase: <code>mixedcaseidentifier</code> .
Using regular identifiers. For example, defined as <code>mixedCaseIdentifier</code> .	Not case sensitive. These are all equivalent: <code>mixedCaseIdentifier</code> , <code>mixedcaseidentifier</code> , <code>MIXEDCASEIDENTIFIER</code> .	Not case sensitive. These are all equivalent: <code>mixedCaseIdentifier</code> , <code>mixedcaseidentifier</code> , <code>MIXEDCASEIDENTIFIER</code> .

Identifier	Advanced SQL Engine Interpretation	ML Engine Interpretation
Defining delimited identifiers. For example, "mixedCaseIdentifier".	Case is preserved: mixedCaseIdentifier.	Case is preserved: mixedCaseIdentifier.
Using delimited identifiers. For example, defined as "mixedCaseIdentifier".	Not case sensitive. These are all equivalent: mixedCaseIdentifier, mixedcaseidentifier, MIXEDCASEIDENTIFIER.	Case sensitive. These are all different: mixedCaseIdentifier, mixedcaseidentifier, MIXEDCASEIDENTIFIER.

When an ML Engine function execution request is sent to ML Engine, any column name of the input table is presented to ML Engine either implicitly as it is named in the input table, or explicitly in the column-list of the input subquery SELECT expression, and also in any function argument that expects a column name. A connector property named Column Name Handling controls the case-sensitivity of column names as they are presented to ML Engine. It takes one of two values, CASE-INSENSITIVE or CASE-SENSITIVE. The default value is CASE-INSENSITIVE. For information about setting connector properties, see [Teradata QueryGrid Connector Properties](#).

- CASE-INSENSITIVE:

When the Column Name Handling connector property is set to CASE-INSENSITIVE, an input column name is treated as case-insensitive (output column names are always handled as case-sensitive), unless it must be quoted to be allowed as a valid column name in ML Engine. An unquoted column name in ML Engine is required to meet these restrictions:

- The first character must be a letter or an underscore "\_".
- The remaining characters must be either a letter, a digit, an underscore, or a dollar sign.
- It must not be a keyword that is invalid as a column name.

These keywords are invalid as a column name:

all	datetime	interval	references
analyse	dec	into	result
analyze	decimal	is	right
and	default	isnull	select
any	deferrable	join	serial
as	desc	leading	serial4
asc	distinct	left	serial8
asymmetric	do	like	session_user
authorization	double	limit	similar
autopartition	else	localtime	smallint

between	end	localtimestamp	some
bigint	except	national	sql_no_cache
bigserial	false	natural	symmetric
binary	float	nchar	table
bit	float4	new	text
bool	float8	not	then
boolean	for	notnull	time
both	force	null	timestamp
case	foreign	nulls	timestamptz
cast	freeze	numeric	timetz
cdistinct	from	off	tinyint
char	full	offset	to
character	grant	old	trailing
check	group	on	true
collate	having	only	unique
column	ilike	or	user
constraint	in	order	using
create	initially	outer	varbit
cross	inner	over	varchar
current_date	int	overlaps	varying
current_role	int2	partition	verbose
current_time	int4	placing	when
current_timestamp	int8	precision	where
current_user	integer	primary	with
date	intersect	real	without

When a column name must be quoted to be valid for ML Engine, it will be treated as CASE-SENSITIVE regardless of Column Name Handling connector property, and there may be cases where the rules to avoid problems with delimited names will apply to this column name, as explained in the next section.

- CASE-SENSITIVE:

If you set the Column Name Handling property to CASE-SENSITIVE, when a function is executed, ML Engine will add quotes to delimit each column name and it will be treated as case-sensitive.

To avoid problems with identifiers, keep the default setting of the connector property as CASE-INSENSITIVE. If that cannot be, follow these rules:

- In tables to be used as input to an ML Engine function execution request, use only lowercase letters, digits, and underscores in column names.
- If you cannot observe the preceding rule (for example, when an existing table has column names that break the rule), give rule-breaking columns correlation names that have only lowercase letters, digits, and underscores. Reference these columns by their correlation names in function argument clauses.

For example, in the following query, the table `ville_climatedata` has two mixed-case column names, `temp_F` and `dewpoint_F`. They have lowercase correlation names in the `SELECT` list, and the `InputColumns` clause references them by their correlation names.

```
SELECT * FROM DWT (
  ON (SELECT city, period, temp_F AS temp_f, pressure_mbar,
        dewpoint_F AS dewpoint_f FROM ville_climatedata
    ) AS InputTable PARTITION BY 1
  OUT TABLE OutputTable (dwt_coef_table)
  OUT TABLE MetaTable (dwt_meta_table)
  USING
  InputColumns ('temp_f', 'pressure_mbar', 'dewpoint_f')
  SortColumn ('period')
  PartitionColumns ('city')
  Wavelet ('db2')
  Level (2)
) AS dwtout;
```

Alternatively, you can enclose a rule-breaking column name in double quotation marks if it meets these requirements:

- It does not appear in an `ON` clause in the SQL syntax.
- It is not passed to a function output table.

For example, in the following query, `Name` and `Period` need correlation names because they appear in the `ON` clause, but `StockPrice` needs only to be enclosed in double quotation marks, because it appears only in an argument clause and is not passed to a function output table.

```
SELECT * FROM SimpleMovAvg (
  ON (SELECT id, "Name" AS name, "Period" AS period, "StockPrice"
        FROM ibm_stock_upper
    ) PARTITION BY name ORDER BY period
  USING
  TargetColumns ('"StockPrice"')
```



```

WindowSize ('10')
IncludeFirst ('true')
) AS dt;

```

## Overriding the Column Name Handling Property

The Column Name Handling property can be overridden within a SQL session by setting a foreign server session attribute. For example:

```
set foreign server attr = 'servername=coprocessor;columnNameHandling=CASE-SENSITIVE;'for session volatile;
```

```
set foreign server attr = 'servername=coprocessor;columnNameHandling=CASE-INSENSITIVE;'for session volatile;
```

See *Teradata® QueryGrid™ Installation and User Guide*, B035-5991.

## Partition Key

The partition key of input/output tables for ML Engine functions must belong to data type group DISTRIBUTIONTYPE. The following data types belong to this group:

- BIGINT
- BLOB
- BYTE(*n*)
- CLOB
- DECIMAL(*p,s*)
- INTEGER
- NUMERIC(*p,s*)
- SMALLINT
- VARBYTE(*n*)
- VARCHAR(*n*)

This also applies to temporary tables created by some ML Engine functions for internal processing. The partitioning column of these tables is typically inherited from input tables and is exposed through a function syntax element. The data type of these columns must also be one of the data types mentioned above.

## Limitations

The following are ML Engine function execution limitations:

- The length of a query output alias must not exceed 25 characters. For example, this alias causes an error:

```
SELECT * FROM KMeans (
  ON (SELECT * FROM kmeansample) as InputTable
  USING
  OutputTable ('kmeanssample_centroid')
  NumberK ('3')
  Threshold ('0.01')
  MaxIterNum ('10')
) AS Long_Function_Name_Alias_Example;
```

- In the *output\_table* clause (see **output\_table** in [ML Engine Analytic Function Execution SQL Syntax](#)), *table\_name* must not have an embedded double quote or an embedded period.

## Nested ML Engine Analytic Functions

ML Engine analytic function calls can be nested within a single SQL statement because the input (ON clause) of a function can be a SELECT statement that can contain another ML Engine analytic function call.

For example:

```
SELECT * FROM SentimentEvaluator (
  ON (
    SELECT * FROM SentimentExtractor (
      ON sentiment_extract_input as "input"
      USING
      TextColumn ('review')
      Accumulate ('category')
      "model" ('dictionary')
    ) AS dt1
  ) AS "input" PARTITION BY 1
  USING ObsColumn ('category')
  SentimentColumn ('out_polarity')
) AS dt;
```

You can construct queries with arbitrary levels of nested function calls by nesting function calls in the subqueries of the ON clauses. When nested queries are executed, each function runs independently. For each function, inputs are transferred from Advanced SQL Engine to ML Engine. The function runs and the results are transferred from ML Engine to Advanced SQL Engine.

## Avoiding Back-and-Forth Data Transfer

Because the output of one function can be the input of another function, nesting functions can cause data transfer from ML Engine to Advanced SQL Engine and back to ML Engine.

You can write the query in a way that avoids some of the back-and-forth data transfer. For example, consider this query:

```

SELECT * FROM SentimentEvaluator (
  ON SentimentExtractor (
    ON sentiment_extractor_input as "input"
    USING
    TextColumn ('review')
    Accumulate ('category')
    "model" ('dictionary')
  ) AS "input" PARTITION BY 1
  USING
  ObsColumn ('category')
  SentimentColumn ('out_polarity')
) AS dt;

```

The preceding query does the following:

- Avoids the back and forth transfer of the intermediate result.
- Runs both functions in one request.
- Transfers only the result of the outer function from ML Engine back to Advanced SQL Engine.

However, when nesting one ML Engine analytic function within another by specifying the inner function directly in the ON clause, these restrictions apply:

- Only two levels of nesting are supported—the output of the inner function is used directly as the input to the outer function.
- The outer function can have only one ON clause, which is the inner function.

## Limitation

A driver function cannot take input from a nested function. The SQL statement must be rewritten to place the results of the nested function into a derived table and let the outer function take its input from the derived table. For example, rewrite the first query below into a form similar to the second query.

This syntax will raise an error:

```

SELECT * FROM KMeans (
  ON Sampling (
    ON kmeans_input AS data partition BY ANY
    USING SampleFraction(0.5)
  ) AS InputTable
  OUT TABLE OutputTable (kmeanssample_centroid)
  OUT TABLE ClusteredOutput (kmeanssample_clusteredoutput3)
  USING
  InitialSeeds ('2249_51_408_8_14', '2165_51_398_7_14.6', '2182_51_404_7_14.6',
    '2204_55_372_7.19_14.6', '2419_44_222_6.6_14.3', '2394_44.3_277_7.3_14.5',

```

```
'2326_43.6_301_7.11_14.3', '2288_44_325_7_14.4')
) AS kmeanstb;
```

The addition of the subquery named sdt in this example is a workaround for the problem:

```
SELECT * FROM KMeans (
  ON (
    SELECT * FROM Sampling (
      ON kmeans_input as data partition BY ANY
      USING SampleFraction(0.5)
    ) AS sdt
  ) as InputTable PARTITION BY 1
  OUT TABLE OutputTable (kmeanssample_centroid)
  OUT TABLE ClusteredOutput (kmeanssample_clusteredoutput3)
  USING
  InitialSeeds ('2249_51_408_8_14', '2165_51_398_7_14.6', '2182_51_404_7_14.6',
    '2204_55_372_7.19_14.6', '2419_44_222_6.6_14.3', '2394_44.3_277_7.3_14.5',
    '2326_43.6_301_7.11_14.3', '2288_44_325_7_14.4')
) AS kmeanstb;
```

## Function Alias

Function alias, also called function mapping, lets you give a function a name that is more descriptive or easier to use when running the function on a foreign server. You can use a function alias instead of referring to a long name, arguments, or complex function execution methods. A function alias hides the location where the function is run.

In function call query:

```
SELECT * FROM function_name (..
```

*function\_name* can refer to an installed function or a function alias. For execution, Advanced SQL Engine first searches for function alias, followed by installed function. The current database of the user is looked up first, followed by globally accessible databases syslib and td\_sysfnlib.

Use these commands when working with function alias:

Command	Purpose
CREATE FUNCTION MAPPING	Creates a function alias.
DROP FUNCTION MAPPING	Drops an existing function alias.
REPLACE FUNCTION MAPPING	Changes the function alias definition.
SHOW FUNCTION MAPPING	Displays the SQL data definition text for the function alias.

nPath and Ntree are exceptions. Function aliases are not defined for these two functions. You must append "@coprocessor" to these function names. For example:

```
SELECT * FROM nPath@coprocessor ( ...
```

See *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

For information about function mapping execution, see *Teradata Vantage™ - SQL Data Manipulation Language*, B035-1146.

## ML Engine Analytic Function Mapping

ML Engine analytic functions have well-defined function aliases. The *Teradata Vantage™ Machine Learning Engine Analytic Function Reference*, B700-4003 uses these aliases to refer to these functions.

To run these functions, use the query in this way:

```
SELECT * FROM function_name ( ...
```

where *function\_name* refers to the function alias (ML Engine analytic function name).

The following table shows the mapping between function aliases (ML Engine analytic function names) and the coprocessor analytic function names.

ML Engine Function Name	Coprocessor Function Name
AdaBoost	AdaBoost_Drive
AdaBoostPredict	AdaBoost_Predict
AllPairsShortestPath	AllPairsShortestPath
AMLGenerator	AMLGenerator
Antiselect_MLE	Antiselect
ApacheLogParser	Apache_Log_Parser
ApproxCardinalityMap	ApproxDCountMap
ApproxCardinalityReduce	ApproxDCountReduce
ApproxPercentileMap	ApproxPercentileMap
ApproxPercentileReduce	ApproxPercentileReduce
ARIMA	Arima
ARIMAPredict	ArimaPredictor
Attribution_MLE	Attribution
BasketGenerator	Basket_Generator
Betweenness	Betweenness
Burst	Burst
Canopy	Canopy
CCM	CCM
CCMPPrepare	CCMPPrepare
CFilter	CFilter

ML Engine Function Name	Coprocessor Function Name
ChangePointDetection	ChangePointDetection
ChangePointDetectionRT	RtChangePointDetection
Closeness	Closeness
ConfusionMatrix	ConfusionMatrix
ConvertToCategorical	Categorize
CorrelationMap	Corr_Map
CorrelationReduce	Corr_Reduce
CoxHazardRatio	CoxPredict
CoxPH	CoxPH
CoxSurvival	CoxSurvFit
CrossValidation	CrossValidation
CumulativeMovAvg	CMAVG
DecisionForest	Forest_Drive
DecisionForestEvaluator	Forest_Analyze
DecisionForestPredict_MLE	Forest_Predict
DecisionTree	Single_Tree_Drive
DecisionTreePredict_MLE	Single_Tree_Predict
DistributionMatchMultiInput	DistnmatchMultipleInput
DistributionMatchReduce	DistnmatchReduce
DTW	DTW
DWT	DWT
DWT2D	DWT2D
EigenVectorCentrality	EigenvectorCentrality
ExponentialMovAvg	EMAVG
FellegiSunter	FellegiSunterTrainer
FellegiSunterPredict	FellegiSunterPredict
FFT	FFT
FMeasure	FMeasure
FPGrowth	FPGrowth
FrequentPaths	FrequentPaths
GeometryOverlay	GeometryOverlay
GLM	GLM
GLML1L2	GLML1L2
GLML1L2Predict	GLML1L2Predict
GLMPredict_MLE	GLMPredict
GMM	GMMFit
GMMPredict	GMMPredict

ML Engine Function Name	Coprocessor Function Name
GMMProfile	GMMProfile
GTree	gTree
Histogram	hist
HMMDecoder	HMMDecoder
HMMEvaluator	HMMEvaluator
HMMSupervised	HMMSupervisedLearner
HMMUnsupervised	HMMUnsupervisedLearner
IdentityMatch	IdentityMatch
IDWT	IDWT
IDWT2D	IDWT2D
IFFT	IFFT
Interpolator	Interpolator
IPGeo	IPGeo
JSONParser	JSONParser
KMeans	KMeans
KMeansPredict	KMeansPlot
KModes	KModes
KModesPredict	KModesPredict
KNN	KNN
KNNRecommender	KNNRecommenderTrain
KNNRecommenderPredict	KNNRecommenderPredict
LAR	LARS
LARPredict	LARSPredict
LDA	LDATrainer
LDAInference	LDAInference
LDATopicSummary	LDATopicPrinter
LevenshteinDistance	LDist
LikelihoodRatioTest	LRTEST
LinReg	LinReg
LinRegInternal	LinRegMatrix
LinRegPredict	LinRegPredict
LocalClusteringCoefficient	LocalClusteringCoefficient
LoopyBeliefPropagation	LoopyBeliefPropagation
MinHash	Minhash
Modularity	Modularity
MultiCaseMatch	Multi_Case
MurmurHash	MurmurHash

ML Engine Function Name	Coprocessor Function Name
NaiveBayesMap	NaiveBayesMap
NaiveBayesPredict_MLE	NaiveBayesPredict
NaiveBayesReduce	NaiveBayesReduce
NaiveBayesTextClassifierPredict_MLE	NaiveBayesTextClassifierPredict
NaiveBayesTextClassifierInternal	NaiveBayesTextClassifierInternal
NaiveBayesTextClassifierTrainer	NaiveBayesTextClassifierTrainer
NamedEntityFinder	FindNamedEntity
NamedEntityFinderEvaluatorMap	EvaluateNamedEntityFinderRow
NamedEntityFinderEvaluatorReduce	EvaluateNamedEntityFinderPartition
NamedEntityFinderTrainer	TrainNamedEntityFinder
NEREvaluator	NEREvaluator
NERExtractor	NER
NERTrainer	NERTrainer
NGramSplitter_MLE	nGram
nPath@coprocessor	nPath
NTree@coprocessor	nTree
OutlierFilter	OutlierFilter
Pack_MLE	Pack
PageRank	PageRank
PathAnalyzer	Path_Analyzer
PathGenerator	Path_Generator
PathStart	Path_Starter
PathSummarizer	Path_Summarizer
PCAMap	pca_map
PCAReduce	pca_reduce
PCAScore	PCAPlot
Percentiles	Percentile
Pivoting	Pivot
PointInPolygon	PointInPolygon
POSTagger	POSTagger
PSALSA	pSALSA
RandomSample	RandomSample
RandomWalkSample	RandomWalkSample
ROC	ROC
Sampling	Sample
SAX	SAX2
Scale	Scale



ML Engine Function Name	Coprocessor Function Name
ScaleByPartition	PartitionScale
ScaleMap	ScaleMap
ScaleSummary	ScalePrinter
SentenceExtractor	Sentenizer
SentimentEvaluator	EvaluateSentimentExtractor
SentimentExtractor	ExtractSentiment
SentimentTrainer	TrainSentimentExtractor
SeriesSplitter	SeriesSplitter
Sessionize_MLE	Sessionize
ShapeletSupervised	SupervisedShapeletTrainer
ShapeletSupervisedClassifier	SupervisedShapeletClassifier
ShapeletUnsupervised	UnsupervisedShapelet
ShapleyAddOnePlayer	AddOnePlayer
ShapleyGenerateCombination	GenerateCombination
ShapleySortCombinations	SortCombination
SimpleMovAvg	SMAVG
StringSimilarity_MLE	StringSimilarity
SVM.Dense	DenseSVMTrainer
SVM.DensePredict	DenseSVMPredictor
SVM.DenseSummary	DenseSVMModelPrinter
SVM.Sparse	SparseSVMTrainer
SVM.SparsePredict_MLE	SparseSVMPredictor
SVM.SparseSummary	SVMModelPrinter
TextChunker	TextChunker
TextClassifier	TextClassifier
TextClassifierTrainer	TextClassifierTrainer
TextMorph	TextMorph
TextParser	Text_Parser
TextTagger	TextTagging
TextTokenizer	TextTokenizer
TF	TF
TFIDF	TF_IDF
TimeSeriesOrders	TimeSeriesOrders
UnivariateStatistics	UnivariateStatistics
Unpack_MLE	Unpack
Unpivoting	Unpivot
URIPack	URIPack

ML Engine Function Name	Coprocessor Function Name
URIUnpack	URIUnpack
VARMAX	VARMAX
VectorDistance	VectorDistance
VWAP	VWAP
WeightedMovAvg	WMAVG
WSRecommender	WSRecommender
WSRecommenderReduce	WSRecommenderReduce
XGBoost	XGBoost_Drive
XGBoostPredict	XGBoost_Predict
XMLParser	XMLParser
XMLRelation	XMLRelation

## Common Analytic Functions in ML Engine and Advanced SQL Engine

These analytic functions are available on both ML Engine and Advanced SQL Engine:

- Antiselect
- Attribution
- DecisionForestPredict
- DecisionTreePredict
- GLMPredict
- MovingAvg
- NaivebayesPredict
- NaiveBayesTextClassifierPredict
- NGramSplitter
- nPath
- Pack
- Sessionize
- StringSimilarity
- SVMSParsePredict
- Unpack

They have common syntax and generate similar results. The analytic functions in Advanced SQL Engine are expected to run with better performance because they are native to the database.

To run a function on Advanced SQL Engine, use this syntax:

```
SELECT * FROM function_name ( ...
```

See *Teradata Vantage™ - Advanced SQL Engine Analytic Functions*, B035-1206.

To run a function on ML Engine, you must append "\_MLE" to the ML Engine function name specified in [ML Engine Analytic Function Mapping](#):

```
SELECT * FROM function_name_MLE ( ...
```

NTree and nPath are exceptions. You must append "@coprocessor" to these function names:

```
SELECT * FROM nPath@coprocessor ( ...
```

See *Teradata Vantage™ Machine Learning Engine Analytic Function Reference*, B700-4003.

## Comments in Queries

### Queries with "@coprocessor"

Comments in queries with "@coprocessor" must have the syntax `/* comment */`. For example:

```
SELECT * FROM GLMPredict@coprocessor (
  ON admissions_test PARTITION BY ANY
  ON glm_admissions_model AS model DIMENSION
  USING
  Accumulate ('id', 'masters', 'gpa', 'stats', 'programming', 'admitted')
  Family ('LOGISTIC')      /* Same as BINOMIAL */
  LinkFunction ('LOGIT')   /* Default */
) AS dt;
```

### Queries without "@coprocessor"

Comments in queries without "@coprocessor" can have either the preceding syntax or the syntax `-- comment`. For example:

```
SELECT * FROM GLMPredict (
  ON admissions_test PARTITION BY ANY
  ON glm_admissions_model AS model DIMENSION
  USING
  Accumulate ('id', 'masters', 'gpa', 'stats', 'programming', 'admitted')
  Family ('LOGISTIC')      /* Same as BINOMIAL */
  LinkFunction ('LOGIT')   -- Default
) AS dt;
```

## EXPLAIN Foreign Function Execution

The output of the EXPLAIN foreign function command for the ML Engine connector lists query steps and shows a detailed explanation of the foreign function execution query from ML Engine. For example:

```
EXPLAIN select * from SESSIONIZE@coprocessor
(
ON sessionize_table_integer
PARTITION BY partition_id
ORDER BY clicktime
USING
TIMECOLUMN('clicktime')
TIMEOUT(60)
RAPIDFIRE(0.2)
) AS S order by S.clicktime;
```

EXPLAIN generates the following response from the ML Engine connector:

#### Explanation

- 
- 1) First, we lock ALICE.sessionize\_table\_integer in TD\_MAP1 for read on a reserved RowHash to prevent global deadlock.
  - 2) Next, we lock ALICE.sessionize\_table\_integer in TD\_MAP1 for read.
  - 3) We execute the following steps in parallel.
    - 1) We do an all-AMPs RETRIEVE step in TD\_MAP1 from ALICE.sessionize\_table\_integer by way of an all-rows scan with no residual conditions executing table operator TD\_SYSFNLIB.QGINITIATOREXPORT in TD\_MAP1 with a condition of ("(1=1)") into Spool 3 (used to materialize view, derived table, table function or table operator drvtab\_inner) (all\_amps), which is built locally on the AMPs. The size of Spool 3 is estimated with low confidence to be 112 rows ( 7,952 bytes). The estimated time for this step is 0.00 seconds.
    - 2) We do an all-AMPs RETRIEVE step in TD\_Map1 from Spool 3 (Last Use) by way of an all-rows scan executing table operator TD\_SYSFNLIB.QGINITIATORIMPORT in TD\_MAP1 with a condition of ("(1=1)") into Spool 5 (used to materialize view, derived table, table function or table operator S) (all\_amps), which is built locally on the AMPs.
- < BEGIN EXPLAIN FOR REMOTE QUERY -->
1. Temporary table Temp\_sess\_1003\_req\_3\_0\_1 is created with the local table's schema and exported to the remote system.
  2. Function SESSIONIZE is executed using the function clause 'TIMECOLUMN('clicktime') TIMEOUT(60) RAPIDFIRE(0.2)' and the result is stored in table S\_f72e961c-a5ca-494e-9af8-000000001721.
  3. Columns partition\_id, clicktime, userid, productname,

```

pagetype, referrer, productprice, sessionid, and rapidfire
are retrieved from S_f72e961c-a5ca-494e-9af8-000000001721.
4. Temporary table Temp_sess_1003_req_3_0_1 is dropped as
part of clean up.
<-- END EXPLAIN FOR REMOTE QUERY >
The size of Spool 5 is estimated with low confidence to be
112 rows (21,728 bytes). The estimated time for this step is
0.00 seconds.
4) We do an all-AMPs RETRIEVE step in TD_Map1 from Spool 5 (Last Use)
by way of an all-rows scan into Spool 6 (group_amps), which is
built locally on the AMPs. Then we do a SORT to order Spool 6 by
the sort key in spool field1. The size of Spool 6 is estimated
with low confidence to be 112 rows (20,048 bytes). The estimated
time for this step is 0.00 seconds.
5) Finally, we send out an END TRANSACTION step to all AMPs involved
in processing the request.
-> The contents of Spool 6 are sent back to the user as the result of
statement 1. The total estimated time is 0.01 seconds.

```

## Help Foreign Function Command

The output of the `HELP FOREIGN FUNCTION` command shows detailed information about the ML Engine function specified.

You must use the coprocessor function name as input to this command and append `@coprocessor` to it. See [ML Engine Analytic Function Mapping](#) to map your function with its coprocessor function name.

```
HELP FOREIGN FUNCTION kmeans@coprocessor;
```

### Function Help

#### Function Name:

kmeans

#### Function Type:

driver

#### Short Description:

Performs K-Means clustering on a data set

#### Long Description:

Finds cluster centroids and assigns data points to clusters using the k-means algorithm.

#### Usage Syntax:

```
kmeans@coprocessor(
ON(...) as InputTable
[ON(...) as CentroidsTable]
[MaxIterNum('INTEGER')]
[Seed('INTEGER')]
[UnpackColumns('BOOLEAN')]
[InitialSeeds('STRING')]
[NumClusters('INTEGER')]
[Threshold('DOUBLE')]
[ClusteredOutput('TABLE_NAME')]
OutputTable('TABLE_NAME')
)
```

**Input Columns:**

<Input Table> should be in the following format < id ,point1 real ...pointn  
real>

**Output Columns:**

<Output Table will be in the following format <clusterid, packedColumn, size,  
withinss>

**Function Owner:**

db\_superuser

**Creation Time:**

2017-08-29 16:02:06.230091

**Function Version:**

7.0\_rel\_1.7\_r65218

**Interfaces Implemented:**

Partition function

## Help Foreign Schema Command

The default output of the `HELP FOREIGN SCHEMA` command lists all tables, views, functions, and installed files for which the user has privileges.

If `listPublicFiles` was enabled during ML Engine deployment, `HELP FOREIGN SCHEMA`

"public"@coprocessor will list all tables, views, and functions for which the user has privileges and will also list all installed files on the public schema regardless of ownership.

The output of this command displays the coprocessor function names. See [ML Engine Analytic Function Mapping](#) to map your function with its coprocessor function name.

You must append @coprocessor to the schema name.

```
HELP FOREIGN SCHEMA "public"@coprocessor;
```

objectname	objecttype	owner
-----	-----	-----
addoneplayer	function	db_superuser
attribution	function	db_superuser
burst	function	db_superuser
ccm	function	db_superuser
ccmmap	function	db_superuser
ccmprepave	function	db_superuser
ccmreduce	function	db_superuser
changepointdetection	function	db_superuser
cmavg	function	db_superuser
coxph	function	db_superuser
densesvmformatconverter	function	db_superuser
densesvmmodelprinter	function	db_superuser
densesvmpredictor	function	db_superuser
densesvmtrainer	function	db_superuser
dtw	function	db_superuser
dwt	function	db_superuser
dwt2d	function	db_superuser
dwtreduce	function	db_superuser
emavg	function	db_superuser
evaluatesentimentextractor	function	db_superuser
fft	function	db_superuser
findnamedentity	function	db_superuser
glm	function	db_superuser
glmpredict	function	db_superuser
ifft	function	db_superuser
kmeans	function	db_superuser
sentinizer	function	db_superuser
trainsentimentextractor	function	db_superuser

## Installing ML Engine Analytic Functions and Files

All ML Engine analytic functions come preinstalled and ready to use.

All analytic functions take input or output argument clauses. Argument clauses specify information the function needs to run, such as table or column names, algorithm parameters, or scalar values.

Some analytic functions have argument clauses that specify input or output files. These files typically contain models, dictionaries, or rules required (input) or produced by (output) the function. The default versions of the input files are preinstalled and ready to use.

Each Advanced SQL Engine user has a private schema in ML Engine for query or function execution. For example, if user *alice* references an ML Engine analytic function, she has a private schema named *alice*. By default, all ML Engine analytic functions and files are preinstalled in the public schema. ML Engine system functions (such as *available\_memory\_on\_jvm*, *qginitiatorexport*, *qginitiatorimport*, and so on) are preinstalled in the *nc\_system* schema. Whenever a user, such as *alice*, references an ML Engine function or file, the system follows a schema search path to try to find that function or file. For example, if *alice* runs a KMeans query, the system searches schema *alice* for a KMeans function, then searches schema *public*, then *nc\_system* schema.

For a list of analytic functions that use files that are preinstalled on ML Engine and the files they use, see *Teradata Vantage™ Machine Learning Engine Analytic Function Reference*, B700-4003.

The preinstalled versions of the analytic functions and input files are adequate for most needs. However, you might want to customize the behavior (change defaults) of functions to suit your particular needs. In most cases, you can change function behavior by changing its argument clauses.

## Access Control of ML Engine Analytic Functions

A user with administrative privileges can grant or revoke access control of ML Engine analytic functions. For example:

To grant the trusted user *td\_ffe\_svc\_acct* the privilege to set role to the permanent user *username*:

```
grant ctcontrol on username to td_ffe_svc_acct;  
grant connect through td_ffe_svc_acct to permanent username without role;
```

To grant the privilege for *username* to execute functions from *syslib*, thereby giving access to the function mappings:

```
grant execute function on syslib to username with grant option;
```

The following grants allow *username* to export data, execute foreign functions, and import the results:

```
grant execute function on TD_SYSFNLIB.QGEXECUTEFOREIGNQUERY to username;  
grant execute function on TD_SYSFNLIB.QGINITIATOREXPORT to username;  
grant execute function on TD_SYSFNLIB.QGINITIATORIMPORT to username;  
grant execute function on TD_SYSFNLIB.QGREMOTEEXPORT to username;  
grant execute function on TD_SYSFNLIB.QGREMOTEIMPORT to username;  
GRANT SELECT ON TD_SERVER_DB.coprocessor TO username;  
GRANT INSERT ON TD_SERVER_DB.coprocessor TO username;  
GRANT execute function on TD_SERVER_DB.coprocessor to username;
```

The next set of grants allow *username* to access a view in which the ML Engine connector will log errors:

```
grant select on td_mle_db.user_mle_errors to username;  
grant insert on td_mle_db.user_mle_errors to username;  
grant delete on td_mle_db.user_mle_errors to username;
```



## Uninstalling ML Engine Analytic Functions and Files

Preinstalled ML Engine analytic functions and files cannot be uninstalled. Contact Teradata Services for uninstallation or upgrading.

## ML Engine User-Defined Functions

An ML Engine user-defined function (UDF) is a programming construct that accepts parameters, uses the accepted parameters, and returns a result. These functions use the syntax and APIs similar to ML Engine functions. ML Engine does not support the development of these functions. However, if you have a UDF developed on Aster Database, you can install and execute it in ML Engine. The execution of a UDF is the same as ML Engine function execution. A UDF invocation gets its data from Advanced SQL Engine, executes in ML Engine, and returns the results to Advanced SQL Engine.

When you create a user-defined function, make sure that the java class name of the function is identical to the jar/zip file name. For example, user-defined function abc.zip must have the java class name of abc.class.

## Compatibility Considerations for UDFs

To ensure the compatibility of UDFs developed on ML Engine with Advanced SQL Engine, consider the following:

- Run at least one FFE query before you call List ([List Functions and Files](#)), Status ([Status of UDFs](#)), Remove ([Uninstall UDFs](#)), or Download ([Downloading UDFs](#)) functionalities.
- For a function implementing a RowFunction interface (function with input ON clause without PARTITION BY or with PARTITION BY ANY), input data is transferred from Advanced SQL Engine to ML Engine as a keyless table (a table with no partition key). Therefore, different executions of the function may result in different data distribution on ML Engine.
- The first column in the output of a function should not be unbounded varchar because it gets mapped to CLOB on SQLE.
- The use of CLOB data type should be minimized to the columns with the CLOB data type with a maximum length greater than 32000.
- For a function implementing a PartitionFunction interface (a function with an input ON clause with PARTITION BY KEY), input data is transferred from Advanced SQL Engine to ML Engine based on the partitioning key. Therefore, different executions of the function result in the same data distribution on ML Engine. However, the order of input data in each distribution may vary. Use an ORDER BY clause with the input ON clause to ensure that the function processes input in the same order every time.
- Input tables on which a function operates are temporarily created tables in ML Engine and are cleaned up after function execution. When the function calls the getInputNames of class baseRuntimeContract.CompletedContract in ML Engine, it gets the name of the temporary table instead of the input table provided in the SQL call on Advanced SQL Engine.

- Tables output by a function are also temporarily created tables in ML Engine. ML Engine transfers these tables to Advanced SQL Engine after the execution, renamed on Advanced SQL Engine based on the names supplied in the SQL call, and removed from ML Engine afterwards.
- Output columns created by the function must begin with a lowercase letter or an underscore (\_). Subsequent characters can be lowercase letters, digits (0-9), or underscores.
- Select the data type of output produced by the functions based on the data type mapping table in [Data Type Mapping between Advanced SQL Engine and ML Engine](#). An unbounded VARCHAR data type on ML Engine maps to a CLOB data type on Advanced SQL Engine. Use VARCHAR(*n*) if the data in the output is not expected to grow beyond *n*=32000 (the maximum for a bounded VARCHAR). Similarly, use bounded numeric data types instead of unbounded numeric data types.
- Most driver functions require a JSON function descriptor file inside its archive (jar or zip). A driver function is a function that uses a JDBC driver inside its code to execute SQL statements. If a driver function has any of these characteristics, it will need a JSON function descriptor to execute correctly:
  - Its implementation takes the name of an input table from the function argument clause.
  - It creates an output table (secondary output) in addition to the SELECT statement output (primary output).
- To return syntax and usage information (using the `HELP FOREIGN FUNCTION` command), a function must have a JSON function descriptor file packaged inside its archive (jar or zip).

See [JSON Function Descriptor](#) for the specifications of a JSON file and steps to include it in the function archive. Any non-driver function or an auxiliary function (a function used internally by a driver function and does not appear in the SQL call on Advanced SQL Engine) does not need a JSON file in order to be executed.

## JSON Function Descriptor

A JSON (JavaScript Object Notation) function descriptor is a JSON file that ML Engine uses for function metadata processing.

Each of the sections in the table below is described in more details in the subsequent tables.

### Major Sections of JSON Descriptor

Section	Description
Header	Specifies function name, version, and type information.
Input_tables	Specifies function ON clause fields for non-driver functions. Input tables for non-driver functions are specified in this section.
Argument_clauses	Specifies function argument fields. Input tables for driver functions and all output tables are specified by argument clauses in this section.

### Header Section Fields

Required fields are marked with an asterisk (\*).

Field	Type	Description
*json_schema_major_version	string	Major version of JSON schema. Set to "1".
*json_schema_minor_version	string	Minor version of JSON schema. Set to "2".
*json_content_version	string	JSON content version. Set to "1".
*function_name	string	Name of function class file.
function_version	string	Version of function.
*function_type	string	Specifies function type ("driver" or "non-driver"). See <a href="#">Compatibility Considerations for UDFs</a> for an explanation of driver and non-driver functions.
short_description	string	Short description of function.
long_description	string	Long description of function.

### Input\_tables Section Fields

Field	Type	Description
*name	string	Specifies ON clause alias. If no alias, use "input" as alias.
*datatype	string	Set to "TABLE_ALIAS" for each ON clause.
requiredInputKind	list of string	Partition information for how ON clause is specified in the syntax. It can be a combination of PartitionByKey, PartitionByAny, or Dimension. If not specified, PartitionByAny is used. Examples are: ON <i>tablename</i> PARTITION BY <i>column-name</i> ON <i>tablename</i> PARTITION BY ANY ON <i>tablename</i> DIMENSION
partitionByOne	boolean	Specifies whether ON clause accepts PartitionByOne. For this to be true, requiredInputKind must be PartitionByKey. For example: ON <i>tablename</i> PARTITION BY 1
partitionByOneInclusive	boolean	Specifies whether ON clause accepts both PartitionByOne and PartitionByKey. For this to be True, PartitionByOne must also be true.
isOrdered	boolean	Specifies whether ON clause requires ORDER BY clause. If False, ORDER BY is optional.
isRequired	boolean	Specifies whether ON clause is required.
description	string	Description of ON clause.

**Argument\_clauses Section Fields**

Field	Type	Description
* name	string	Specifies argument name.
* datatype	string	Specifies data type of argument value; one of these: <ul style="list-style-type: none"> <li>• "BOOLEAN"</li> <li>• "INTEGER"</li> <li>• "DOUBLE"</li> <li>• "LONG"</li> <li>• "STRING"</li> <li>• "TABLE_NAME" (Used for names of input or output tables for driver functions. Identify output tables by setting the <code>isOutputTable</code> field to <code>True</code>.)</li> <li>• "COLUMN_NAMES" (Used for names of columns in input tables for driver functions.)</li> <li>• "COLUMNS" (Used for names of columns in input tables for non-driver functions.)</li> </ul>
datatype	string	
isRequired	boolean	Specifies whether argument clause is required or optional.
* defaultValue	Boolean, numeric, or string depending on the value of the data type.	Specifies default value of argument (value for function to use if the user omits argument). Specify only if <code>isRequired</code> is set to <code>false</code> .
permittedValues	list of string	Specifies permitted values of argument clause. For example: <pre> "permittedValues": [     "SPHERICAL",     "DIAGONAL",     "FULL",     "TIED" ] </pre>
description	string	Description of argument clause.
* isOutputTable	boolean	Specifies whether argument clause accepts database table as output. For this value to be <code>true</code> , data type must be set to <code>"TABLE_NAME"</code> .

**JSON Descriptor Example: GMMFit Function**

```

{
  "json_schema_major_version": "1",
  "json_schema_minor_version": "2",
  "json_content_version": "1",

```

```

"function_name": "GMMFit",
"function_version": "1.2",
"function_type": "driver",
"short_description": "Fits a Gaussian Mixture Model to data.",
"long_description": "Clusters data using a Gaussian Mixture Model or a
Dirichlet Process Gaussian Mixture Model.",
"input_tables":[
{
  "requiredInputKind":[
    "PartitionByKey"
  ],
  "isOrdered": false,
  "partitionByOne": true,
  "name": "init_params",
  "isRequired": true,
  "description": "Contains initial values for the cluster weights, means,
and covariances.",
  "datatype": "TABLE_ALIAS"
}
],
"argument_clauses":[
{
  "isOutputTable": false,
  "name":"InputTable",
  "isRequired": true,
  "description": "Specifies the name of the table that contains the input
data to be clustered.",
  "datatype": "TABLE_NAME"
},
{
  "isOutputTable": true,
  "name":"OutputTable",
  "isRequired": true,
  "description": "Specifies the name of the output table to which the
function outputs cluster information.",
  "datatype": "TABLE_NAME"
},
{
  "defaultValue": 20,
  "name": "MaxClusternum",
  "isRequired": false,
  "description": "Specifies the maximum number of clusters in a Dirichlet
process model.",
  "datatype": "INTEGER"
}
]

```

```

    },
    {
      "permittedValues": [
        "SPHERICAL",
        "DIAGONAL",
        "FULL",
        "TIED"
      ],
      "defaultValue": "DIAGONAL",
      "name": "CovarianceType",
      "isRequired": false,
      "description": "Specifies the type of the covariance matrices.",
      "datatype": "STRING"
    },
    {
      "defaultValue": 0.001,
      "name": "Tolerance",
      "isRequired": false,
      "description": "Specifies the minimum change in log-likelihood between
iterations that causes the function to terminate.",
      "datatype": "DOUBLE"
    },
    {
      "defaultValue": false,
      "name": "PackOutput",
      "isRequired": false,
      "description": "Specifies whether the function packs the output. The
default value is 'false'.",
      "datatype": "BOOLEAN"
    }
  ]
}

```

## Adding a JSON File to a Function Archive

To add a JSON file into a function archive, provide the JSON file as part of the build process to be included in the function archive. If you already have an archive file, you can add the JSON file into the archive zip or jar file using the Java Archive (jar) tool and zip utility. The following example shows the step-by-step process of adding a JSON file, DTW.json, into a function zip file, DTW.zip, on a Linux-based system. If your function archive is a jar file into which you want to add the JSON file, you need only steps 4 through 8.

1. Verify the contents of DTW.zip:

```
$ jar tf DTW.zip
```

```
DTW.jar
FastDTW-1.1.0.jar
sql-mr_utilities.jar
```

- Unzip DTW.zip to extract the three jar files listed in the previous step:

```
$ unzip DTW.zip
```

```
Archive:  DTW.zip
  inflating: DTW.jar
  inflating: FastDTW-1.1.0.jar
  inflating: sql-mr_utilities.jar
```

- Verify the contents of DTW.jar to which you want to add DTW.json:

```
$ jar tf DTW.jar
```

```
META-INF/
META-INF/MANIFEST.MF
com/
com/asterdata/
com/asterdata/sqlmr/
com/asterdata/sqlmr/functions/
com/asterdata/sqlmr/functions/analytics/
com/asterdata/sqlmr/functions/analytics/timeseries_analysis/
com/asterdata/sqlmr/functions/analytics/
timeseries_analysis/dynamic_time_warping/
com/asterdata/sqlmr/functions/analytics/timeseries_analysis/
dynamic_time_warping/DTW.class
com/asterdata/sqlmr/functions/analytics/timeseries_analysis/
dynamic_time_warping/DTWArgs.class
```

- Extract DTW.jar and list the directory:

```
$ jar xf DTW.jar
```

The jar file gives you compiled code in a directory structure (which in this example is under com/) and a manifest directory under META-INF/.

```
$ ls com DTW.jar DTW.json DTW.zip FastDTW-1.1.0.jar META-INF sql-
mr_utilities.jar
```

- Delete the DTW.jar and DTW.zip files (because you will create new jar and zip files):

```
$ rm DTW.jar
```

```
$ rm DTW.zip
```

- Delete the META-INF directory (because the jar tool generates it when you create a jar file):

```
$ rm -r META-INF/
```

- Copy DTW.json into the source code directory structure at the same sub-directory level where the function class resides:

```
$ cp DTW.json com/asterdata/sqlmr/functions/analytics/
timeseries_analysis/dynamic_time_warping/
```

- Create a new archive DTW.jar file from the source code:

```
$ jar cf DTW.jar com/
```

- Verify that the contents of the jar file match the contents of the old jar file, except the JSON file:

```
$ jar tf DTW.jar
```

```
META-INF/
META-INF/MANIFEST.MF
com/
com/asterdata/
com/asterdata/sqlmr/
com/asterdata/sqlmr/functions/
com/asterdata/sqlmr/functions/analytics/
com/asterdata/sqlmr/functions/analytics/timeseries_analysis/
com/asterdata/sqlmr/functions/analytics/
timeseries_analysis/dynamic_time_warping/
com/asterdata/sqlmr/functions/analytics/timeseries_analysis/
dynamic_time_warping/DTW.class
com/asterdata/sqlmr/functions/analytics/timeseries_analysis/
dynamic_time_warping/DTWArgs.class
```

- Create a new zip file from the newly created DTW.jar and other libraries in the zip file:

```
$ zip -r DTW.zip DTW.jar FastDTW-1.1.0.jar sql-mr_utilities.jar
```

```
adding: DTW.jar (deflated 11%)
adding: FastDTW-1.1.0.jar (deflated 8%)
adding: sql-mr_utilities.jar (deflated 11%)
```

- Verify the contents of DTW.zip:

```
$ jar tf DTW.zip
```

```
DTW.jar
FastDTW-1.1.0.jar
sql-mr_utilities.jar
```

The archive file DTW.zip is ready to be installed on ML Engine.

## Rules for Installing UDFs

- The user performing the installation must be authorized. A user is authorized if they own the schema or the schema is public.



- The UDF (or a file with the same name) does not exist on ML Engine.
- The UDF name is different from the destination schema name.
- The UDF file size is 238 MB or less.
- The archive file name for the UDF is the same as the main function class in the archive that must be executed.
- You cannot have executables or scripts in your UDFs.

## UDF Management

UDFs on ML Engine can be managed easily using REST APIs. REST APIs can be issued from any client machine and the response will be returned to the same client machine. This is the base URL you must use during the REST API invocation:

```
https://<CLUSTER_DOMAIN>/mlengine/udfmanager/
```

## Authentication

All API requests require a valid Authorization header. The header requires a valid JWT token. For example, Authorization: Bearer VALID\_TOKEN. For information on how to obtain a token, see [How to Generate a JWT Token](#).

### How to Generate a JWT Token

There are two ways you can generate a JWT token:

- Login to this Teradata AppCenter portal:

```
https://<CLUSTER_DOMAIN>/api/user/swagger-ui.html
```

and generate a JWT token using:

```
POST /token Authenticate A User
```

The following is a sample JSON (using sample username and password), after you click **Try it Out** and **Execute** to get the JWT token.

```
{
  "username": "alice"
  "password": "alice"
}
```

- Use this REST API to generate the JWT token:

POST https://<CLUSTER\_DOMAIN>/api/user/token

## Example Request

```
curl \
-X POST \
"https://<CLUSTER_DOMAIN>/api/user/token" \
-H "accept: application/json" \
-H "Content-Type: application/json" \
-d "{ \"password\": \"alice\" \"username\": \"alice\"}"
```

### Example Response

```
{
  "username": "alice",
  "display_name": "alice",
  "local": true,
  "admin": true,
  "access_token":
    "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwiaWUiOiA6ICJQktGOF9BUF9LdnRkNl83WTRfNH
    owNXZYMTZuWk1PUjZBNWNrNzBmWGE0In0.eyJqdGkiOiI5ZjhkZWY3MC00NWwTQWMDUtODAIYi1jY
    zAzNWYyNDY5MDQiLCJleHAiOjE1OTE3NDU1MDksIm5iZiI6MkwiaWF0IjoxNTkxNzM4MzA5LCJpc3Mi
    OiJodHRwczoVL3RkYXAxODE3bTEubGFicy50ZXJhZGF0YS5jb20vYXV0aC9yZWFSbXMvdWRhIiwiaXYX
    kIjoicmVhbG0tbWFnYWdlbWVudCIsInN1YiI6IjY3YmYxMzg4LWE2MjUtNDM0ZC05ZDZlLWZiMmEzOT
    MzMGMjkmIisInR5cCI6Ik1JLYXJlcilIsImF6cCI6InVzZXItc2VydmlljZSIIsImF1dGhfZGltdSI6Mkwic
    2Vzc2lubl9zdGF0ZSI6IjBiNmY5N2JhLTVhMTctNGI2Yy05ZGE4LWVjZjE5YzIwNWU2MSIsImFjcml6
    IjEiLCJyZWFSbV9hY2Nlc3MiOnsicm9sZXMiOlslcnktbWFnYWdlciIsInVzZXItbWFnYWdlciIsImFwc
    C1tYW5hZ2VyIiwic3lzdGVtLW1hbmFnZXIIiLCJzZWNYZXQtbnWFnYWdlciIsIm9mZmxpbmVfYXV0e
    NjZXNzIiwibm90awZpY2F0aw9uLW1hbmFnZXIIiLCJjbGllbnQtbnWFnYWdlciIsInNzb3ltYW5hZ2VyI
    iwidW1hX2F1dGhvcmcl6YXRpb24iLCJzdXB1cnVzZXIIiXX0sInJlc291cmNlX2FjY2VzcyI6eyJyZWFS
    bS1tYW5hZ2VtZW50Ijp7InJvbGVzIjpbIm1hbmFnZS1yZWFSbSIsIm1hbmFnZS1pZGVudGl0eS1wcm9
    2awRlcnMiLCJjcmVhdGUtY2xpZW50IiwibWFnYWdlLXVzZXJzIiwibWFnYWdlLWNSawVudHMiXX19LC
    JzY29wZSI6Im9wZW5pZCBwcm9maWxlIGVtYWlsIiwic3ViIjoicm9vdCIsImVtYWlsX3ZlcmllmaWVki
    jpmYXxzZSwicHJvdmlkZXJJZCI6bnVsbcWibmFtZSI6InJvb3QgdXNlciIsImFkbWluIjpb0cnVlLCJw
    cmVmZXJyZWRFdXNlcm5hbWUiOiJyb290Iiwiz2l2ZW5fbmFtZSI6InJvb3QiLCJmYW1pbHlfbmFtZSI
    6InVzZXIIiLCJsb2NhbiCI6dHJ1ZSwidXNlcm5hbWUiOiJyb290In0.JU-wPLWUR4-
    cPtOv0TF0lRmBV8Cjhe3sIGAmI9dmbDSxFxVDGYBvzFaTC17NE2dbxNLD4ejMwLWKTD6eOIZeJbXy0A
    Ba715lNc7GLSrSizGmwGULlp6lrswTwG9VcbMIPnRA2lSb--
    nwXQBdy4xgcWPvIHT7FtXQ2DZW8NymHtryQ4VUHReEyCEgJ0EEZ-
    eR989JmbK4VfzOcKoyItbf0HfSzYPh-2vUeSufijTRFU2HrKW4F2qvzKLezkoYjtdtA9ce400gmPTW1
    S3sjjW5bX5agX_lcOVxktKm8PAAHFoYy6sG70YuRT19Jk6vyNubSXPPCsrgtdrzjaQVKvpMrQ",
  "token_type": "Bearer",
  "expires_in": 7200000,
```

```
"expires_at": "2020-06-09T23:31:49Z",
"token_in":
"eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICIwODc0ZGRkOS0wMzNjLTQ1YTgtYjYyOTY0YmNhZmRhNGM5YmYifQ.eyJqdGkiOiIwYzE1ZmM2Ny01MjJzLTRkNmYtODA1ZC04YWlxYTViZjYwNGEiLCJleHAiOjE1OTE3NDU1MDksIm5iZiI6MCwiaWF0IjoxNTkxNzM4MzA5LCJpc3MiOiJodHRwcovL3RkYXAxODE3bTEubGFicy50ZXJhZGF0YS5jb20vYXV0aC9yZWFSbXMvdWRhIiwiaXVkJjoicmVhbG0tbWFnYUdlbWVudCIsInN1YiI6IjY3YmYxMzg4LWE2MjUtNDM0ZC05ZDZlLWZiMmEzOTMzMGMjKMiIsInR5cCI6IkJlYXJlciiSiImF6cCI6InVzZXItc2VydmlljZSIiImF1dGhfZGltdZSI6MCwic2Vzc2lubl9zdGF0ZSI6IjBiNmY5N2JhLTZhMTctNGI2Yy05ZGE4LWVjZjE5YzIwNWU2MSIsImFjcii6IjEiLCJyZWFSbv9hY2Nlc3MiOnsicm9sZXMiOlscXVlcnktbWFnYUdlbWVudlciIsInVzZXItbWFnYUdlbWVudlciIsImFwcC1tYW5hZ2VyIiwic3lzZGVtLW1hbmFnZXIIiLCJzZWNYZXQtbWFnYUdlbWVudlciIsIm9mZmxpbmVfYWVudlciIsInNzby1tYW5hZ2VyIiwidW1hX2F1dGhvcm16YXRpb24iLCJzdXB1cnVzZXIIiXX0sInJlc291cmNlX2FjY2VzcyI6eyJyZWFSbS1tYW5hZ2VtZW50Ijp7InJvbGVzIjpbIm1hbmFnZS1yZWFSbSIsIm1hbmFnZS1pZGVudGl0eS1wcm92aWR1cnMiLCJjcmVhdGUTy2xpZW50IiwibWFnYUdlLXVzZXJzIiwibWFnYUdlLWNsaWVudHMIXX19LCJzY29wZSI6Im9wZW5pZCBwcm9maWxlIGVtYWlsIiwic3ViIjoicm9vdCIsImVtYWlsX3Zlcm1maWVkJjpmYXxzZSwicHJvdmlkZXJJZCI6bnVsbcwibmFtZSI6InJvb3QgdXNlciIsImFkbWluIjpb0cnV1LCJwcmVmZXJyZWRFdXNlcm5hbWUiOiJyb290Iiwiz212ZW5fbmFtZSI6InJvb3QiLCJmYW1pbHlfbmFtZSI6InVzZXIIiLCJsb2NhbCI6dHJ1ZSwidXNlcm5hbWUiOiJyb290In0.JU-wPLWUR4-cPtOv0TF01RmBV8Cjhe3sIGAmI9dmbDSxFxVDGYBvzFaTC17NE2dbxNLD4ejMwlWKTd6eOIzeJbXy0ABa715lNc7GLSrSizGmwGUlPlp6lrswTwG9VcbMIPnRA2lSb--nwXQBdy4xgcWPvIHT7FtXQ2DZW8NymHtryQ4VUHreEyCEgJ0EEZ-eR989JmbK4Vfz0ckoyItbf0HfSzYPH-2vUesufijTRFU2HrKW4F2qzvKLekoYjtdtA9ce400gmPTW1S3sjjw5bX5agX_lcovxktKm8PAAHFoYy6sG70YuRT19Jk6vyNubSXPPCsrgtdrzjaQVKvpMrQ",
"refresh_token":
"eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICIwODc0ZGRkOS0wMzNjLTQ1YTgtYjYyOTY0YmNhZmRhNGM5YmYifQ.eyJqdGkiOiIwYzE1ZmM2Ny01MjJzLTRkNmYtODA1ZC04YWlxYTViZjYwNGEiLCJleHAiOjE1OTE3NDU1MDksIm5iZiI6MCwiaWF0IjoxNTkxNzM4MzA5LCJpc3MiOiJodHRwcovL3RkYXAxODE3bTEubGFicy50ZXJhZGF0YS5jb20vYXV0aC9yZWFSbXMvdWRhIiwiaXVkJjoiaHR0cHM6Ly90ZGFwMTgxN20xLmhhYnMudGVyYWRhdGEuY29tL2F1dGdvcmVhbG1zL3VkYSIsInN1YiI6IjY3YmYxMzg4LWE2MjUtNDM0ZC05ZDZlLWZiMmEzOTMzMGMjKMiIsInR5cCI6IkJlL1JlZnJlc2giLCJhenAiOiJ1c2VyLXNlcnZpY2UiLCJhdXRoX3RpbWUiOiJAsInNlc3Npb25fc3RhdGUiOiIwYjZmOTdiYS01YTE3LTRiNmMtOWRhOC1lY2YxOWMyMDVlNjEiLCJyZWFSbv9hY2Nlc3MiOnsicm9sZXMiOlscXVlcnktbWFnYUdlbWVudlciIsInVzZXItbWFnYUdlbWVudlciIsImFwcC1tYW5hZ2VyIiwic3lzZGVtLW1hbmFnZXIIiLCJzZWNYZXQtbWFnYUdlbWVudlciIsIm9mZmxpbmVfYWVudlciIsInNzby1tYW5hZ2VyIiwidW1hX2F1dGhvcm16YXRpb24iLCJzdXB1cnVzZXIIiXX0sInJlc291cmNlX2FjY2VzcyI6eyJyZWFSbS1tYW5hZ2VtZW50Ijp7InJvbGVzIjpbIm1hbmFnZS1yZWFSbSIsIm1hbmFnZS1pZGVudGl0eS1wcm92aWR1cnMiLCJjcmVhdGUTy2xpZW50IiwibWFnYUdlLXVzZXJzIiwibWFnYUdlLWNsaWVudHMIXX19LCJzY29wZSI6Im9wZW5pZCBwcm9maWxlIGVtYWlsIn0.HQNmg-l0nDrrXHaoLtzyedKUHOYvAe2k_Qp1kt4fy4w"
```

Get the JWT token (token1). In this case, "eyJ....LKG".

A token is valid for two hours from the time of issue. When the token expires, delete the cache or stored copy and obtain a new token.

## Install UDFs

You can upload dtw.zip (the zip file containing the function or file) from any client machine. Specify the schema to which the installation should occur. You can only install in your private schema or public schema.

A unique uuid or udfid will be generated for each of the UDFs installed. The uuid will be available in the response.

### REST API

```
POST https://<CLUSTER_DOMAIN>/mlengine/udfmanager/api/v1/udfs
```

### Example Request

```
curl \
  -X POST \
  -H "Accept: application/json" \
  -H "Content-Type: multipart/form-data" \
  -H "Authorization: Bearer <JWT token>" \
  -F 'file=@/home/user/dtw.zip' \
  -F 'schema=alice' (OR) 'schema=public' \
  "https://<CLUSTER_DOMAIN>/mlengine/udfmanager/api/v1/udfs"
```

### Example Response

```
HTTP/1.1 201 OK
Content-Type: "application/json"
```

```
{
  "filename": "dtw.zip",
  "result": "200",
  "owner": "alice",
  "message": "function dtw.zip installed successfully",
  "id": "e4349670-c7e3-4214-b133-66c5708a1787",
  "schema": "alice"
}
```

You can optionally create a function alias for your UDF. See [Function Alias](#).

## Invoking UDFs

### Prerequisite:

You must install your UDF before you can invoke it.

- Invoke your UDF as a function reference in a SQL query:

```
SELECT * FROM { udf@coprocessor | udf_alias } (
  ON (SELECT * FROM MLE_function AS input_table_alias)
  USING MLE_function_arguments
);
```

For example:

```
SELECT * FROM new_kmeans@coprocessor (
  ON (SELECT * FROM KmeansSample as InputTable)
  USING
  OutputTable ('kmeanssample_centroid')
  NumberK ('3')
  Threshold ('0.01')
  MxIterNum ('10')
);
```

## List Functions and Files

You can list the functions and files in your private schema or in the public schema. All the files you own will be returned.

### REST API

```
GET https://<CLUSTER_DOMAIN>/mlengine/udfmanager/api/v1/udfs
```

### Example Request

```
curl \
  -H "Authorization: Bearer <JWT token>" \
  -X GET \
  -d '{"schema"="alice"} (OR) '{"schema"="public"}' \
  "https://<CLUSTER_DOMAIN>/mlengine/udfmanager/api/v1/udfs"
```

**Example Response**

```
HTTP/1.1 200 OK
Content-Type: "application/json"
```

```
{
  [
    {
      name:<name1>,
      user:<user>,
      schema:<schema>,
      descriptor:true or false,
      version:<version>,
      id:<functionID>,
    },
  ]
}
```

**Status of UDFs**

You can get the status of a function or file in a private or public schemas by passing the uuid or udfid.

**REST API**

```
GET https://<CLUSTER_DOMAIN>/mlengine/udfmanager/api/v1/udfs/{UDFID}
```

**Example Request**

```
curl \
  -H "Accept: application/json" \
  -H "Authorization: Bearer <JWT token>" \
  -X GET \
  "https://<CLUSTER_DOMAIN>/mlengine/udfmanager/api/v1/udfs/e4349670-c7e3-4214-b133-66c5708a1787"
```

**Example Response**

```
HTTP/1.1 201 OK
Content-Type: "application/json"
```

```
{
  "name"."dtw.zip",
  "owner"."alice",
}
```

```
"schema"."alice",
"id":"e4349670-c7e3-4214-b133-66c5708a1787",
"status":"Installed"
}
```

## Downloading UDFs

You usually do not need to download a UDF because you already have access to the archive file that you used to install the UDF. If you do not have access to the archive file and want to download the installed UDF, see [Downloading Custom Files](#).

## Custom Files

Some ML Engine functions take files as input instead of tables, as mentioned in [Installing ML Engine Analytic Functions and Files](#). The default files that functions use are preinstalled. Custom files can be installed the same way as custom functions. For information, see [Install UDFs](#).

## Access to Custom Files

Custom files, after being installed, can be used as input by ML Engine functions or custom UDFs. For details on accessing custom files in a specific ML Engine function, see *Teradata Vantage™ Machine Learning Engine Analytic Function Reference*, B700-4003.

## Downloading Custom Files

A custom file can be downloaded to any location in your client machine. The downloaded files can be opened, read, updated, or transferred to other systems like a regular file.

You can download a file from your private schema or from the public schema. Before you download, you must know your udfid.

### REST API

```
GET https://<CLUSTER_DOMAIN>/mlengine/udfmanager/api/v1/udfs/
{UDFID}?download=true
```

### Example Request

The file will be downloaded to standard output and to users and can be redirected to a zip file in any location.

```
curl \
-D header.txt \
```

```
-kb cookies.txt \
-H "Accept: application/json" \
-H "Authorization: Bearer <JWT token>" \
-X GET \
"https://<CLUSTER_DOMAIN>/mlengine/udfmanager/api/v1/udfs/e4349670-c7e3-4214-b133-66c5708a1787?download=true" > /home/tmp/output.zip
```

Or

```
curl
-H "Accept: application/json" \
-D header.txt \
-o mydownload.zip \
-H "Authorization: Bearer <JWT token>" \
-X GET \
"https://<CLUSTER_DOMAIN>/mlengine/udfmanager/api/v1/udfs/e4349670-c7e3-4214-b133-66c5708a1787?download=true"
```

### Example Response

```
Content-Type: "application/octet/stream"
Content-Disposition: 'attachment:filename=output.zip'
```

```
<Binary content>
```

## Uninstall UDFs

An authorized user can uninstall a UDF from their own schema or a public schema.

### REST API

```
DELETE https://<CLUSTER_DOMAIN>/mlengine/udfmanager/api/v1/udfs/{UDFID}
```

### Example Request

```
curl \
-H "Accept: application/json" \
-H "Authorization: Bearer <JWT token>" \
-X DELETE \
"https://<CLUSTER_DOMAIN>/mlengine/udfmanager/api/v1/udfs/e4349670-c7e3-4214-b133-66c5708a1787"
```



**Example Response**

```
HTTP/1.1 204 OK
Content-Type: "application/json"
```

```
{
  "name"."dtw.zip",
  "owner"."alice",
  "schema"."alice",
  "id":"e4349670-c7e3-4214-b133-66c5708a1787",
  "status":"Removed"
}
```

## Data Type Mapping

## Data Type Mapping between Advanced SQL Engine and ML Engine

Advanced SQL Engine to ML Engine

Advanced SQL Engine Type	Advanced SQL Engine Range	Default ML Engine Type	ML Engine Range	Expected Data Loss in Conversion
BYTEINT	-128 to 127	SMALLINT / INT2	-32,768 to +32,767	No
SMALLINT	-32,768 to 32,767	SMALLINT / INT2	-32,768 to +32,767	No
INTEGER	-2,147,483,648 to 2,147,483,647	INTEGER / INT4	-2,147,483,648 to +2,147,483,647	No
BIGINT	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	BIGINT / INT8	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	No
DECIMAL( <i>p</i> , <i>s</i> ) / DEC / NUMERIC <i>p</i> : total number of digits, <i>s</i> : number of digits to the right of the decimal point $1 \leq p \leq 38$ $0 \leq s \leq p$ DECIMAL === DECIMAL(5, 0) DECIMAL( <i>p</i> ) === DECIMAL( <i>p</i> , 0).	<ul style="list-style-type: none"> <li>The maximum value for DECIMAL(<i>p</i>, <i>s</i>) is a value consisting of <i>p</i> 9s, with the decimal point <i>s</i> digits from the right.</li> <li>The minimum value is the negative of the maximum value.</li> </ul>	NUMERIC( <i>p</i> , <i>s</i> ) / NUMERIC / NUMERIC( <i>p</i> ) $1 \leq p \leq 1000$ $0 \leq s \leq p$ NUMERIC( <i>p</i> ) === NUMERIC( <i>p</i> , 0)	<ul style="list-style-type: none"> <li>The maximum value for NUMERIC(<i>p</i>, <i>s</i>) is a value consisting of <i>p</i> 9s, with the decimal point <i>s</i> digits from the right.</li> <li>The minimum value is the negative of the maximum value.</li> </ul>	No
FLOAT / REAL / DOUBLE PRECISION	Values in sign/magnitude form ranging from $2.226 * 10^{-308}$ to $1.797 * 10^{+308}$	DOUBLE PRECISION / FLOAT4 / FLOAT( <i>p</i> ) $1 \leq p \leq 24$	The double precision type typically has a range of approximately $1.0 * 10^{-307}$ to $1.0 * 10^{+308}$ with a precision of at least 15 digits. Infinity -Infinity NaN	No
NUMBER( <i>p</i> , <i>s</i> ) $1 \leq p \leq 38$ $0 \leq s \leq p$	<ul style="list-style-type: none"> <li>The maximum value for DECIMAL(<i>p</i>, <i>s</i>) is a value consisting of <i>p</i> 9s, with the decimal point <i>s</i> digits from the right.</li> </ul>	NUMERIC( <i>p</i> , <i>s</i> ) / NUMERIC / NUMERIC( <i>p</i> ) $1 \leq p \leq 1000$	<ul style="list-style-type: none"> <li>The maximum value for NUMERIC(<i>p</i>, <i>s</i>) is a value consisting of <i>p</i> 9s, with the</li> </ul>	No

Advanced SQL Engine Type	Advanced SQL Engine Range	Default ML Engine Type	ML Engine Range	Expected Data Loss in Conversion
<b>Note:</b> Unbounded number defined as NUMBER(*) / NUMBER is not supported.	<ul style="list-style-type: none"> <li>The minimum value is the negative of the maximum value.</li> </ul>	$0 \leq s \leq p$ NUMERIC( $p$ ) === NUMERIC( $p$ , 0)	decimal point $s$ digits from the right. <ul style="list-style-type: none"> <li>The minimum value is the negative of the maximum value.</li> </ul>	
VARCHAR( $n$ )/CHARACTER VARYING( $n$ )/CHAR VARYING( $n$ )/LONG VARCHAR	<ul style="list-style-type: none"> <li>For the LATIN server character set, the maximum value for <math>n</math> is 64,000 characters.</li> <li>For the UNICODE and GRAPHIC server character sets, the maximum value for <math>n</math> is 32,000 characters.</li> <li>For the KANJISJIS server character set, the maximum value for <math>n</math> is 32,000 bytes.</li> <li>For LONG VARCHAR, the size is the maximum value of the <math>n</math> in the above description. That is, 64,000 characters for LATIN, 32,000 characters for UNICODE and GRAPHIC, and 32,000 bytes for KANJISJIS.</li> </ul>	CHARACTER VARYING( $n$ ) / VARCHAR( $n$ )	Variable length with limit: the maximum length is 10 MB.	Strings will be truncated if they are larger than the value of <i>Default String Size</i> property of the ML Engine Connector in QueryGrid.
CHARACTER / CHAR( $n$ ) default $n::1$	<ul style="list-style-type: none"> <li>For the LATIN server character set, the maximum value for <math>n</math> is 64,000 characters.</li> <li>For the UNICODE and GRAPHIC server character sets, the maximum value for <math>n</math> is 32,000 characters.</li> <li>For the KANJISJIS server character set, the maximum value for <math>n</math> is 32,000 bytes.</li> </ul>	CHARACTER( $n$ ) / CHAR( $n$ )	Fixed length with blank padded: the maximum length is 10 MB.	Strings will be truncated if they are larger than the value of <i>Default String Size</i> property of the ML Engine Connector in QueryGrid.
CHARACTER LARGE OBJECT/ CLOB ( $n$ K/M/G) default $n::$ maximum value allowed	<ul style="list-style-type: none"> <li>For LATIN:               <ul style="list-style-type: none"> <li><math>n \leq 2,097,088,000</math>.</li> <li><math>nK = n * 1024</math>, for the LATIN server character set, <math>n</math> cannot exceed 2,047,937.</li> <li><math>nM = n * 1024K</math>, for the LATIN server character set, <math>n</math> cannot exceed 2,047,937.</li> <li><math>nG = n * 1024M</math>, <math>n</math> must be 1 and the server character set must be LATIN.</li> </ul> </li> <li>For UNICODE:               <ul style="list-style-type: none"> <li><math>n \leq 1,048,544,000</math>.</li> </ul> </li> </ul>	Based on the size: <ul style="list-style-type: none"> <li>If <math>n &lt; 10</math> MB: VARCHAR(<math>n</math>)</li> <li>If <math>n &gt; 10</math> MB: VARCHAR/ CHARACTER VARYING</li> </ul>	The maximum length is 10 MB.	Strings will be truncated if they are larger than the value of <i>Default String Size</i> property of the ML Engine Connector in QueryGrid.

Advanced SQL Engine Type	Advanced SQL Engine Range	Default ML Engine Type	ML Engine Range	Expected Data Loss in Conversion
	$nK = n * 1024$ , for the UNICODE server character set, $n$ cannot exceed 1,023,968. $nM = n * 1024k$ , for the UNICODE server character set, $n$ cannot exceed 999. There is no G for the Unicode.			
DATE	YEAR-MONTH-DAY $1 \leq \text{YEAR} \leq 9999$ $1 \leq \text{MONTH} \leq 12$ $1 \leq \text{DAY} \leq 31$	DATE	4713 BC to 5,874, 897 AD	No
TIME[(p) $0 \leq p \leq 6$  <b>Note:</b> Advanced SQL Engine supports SECOND > 60 (leap second), which is not supported by ML Engine.	$00 \leq \text{HOUR} \leq 23$ $00 \leq \text{MINUTE} \leq 59$ $00.000000 \leq \text{SECOND} \leq 61.999999$	TIME[(p)] WITHOUT TIME ZONE $0 \leq p \leq 6$	00:00:00 24:00:00	No
TIME[(p)] WITH TIME ZONE $0 \leq p \leq 6$  <b>Note:</b> Advanced SQL Engine supports SECOND > 60 (leap second), which is not supported by ML Engine.	$00 \leq \text{HOUR} \leq 23$ $00 \leq \text{MINUTE} \leq 59$ $00.000000 \leq \text{SECONDS} \leq 61.999999$ $-12.59 \leq \text{TIMEZONE\_HOUR} \leq +14.00$ $-12.59 \leq \text{TIMEZONE\_MINUTE} \leq +14.00$	TIME[(p)] WITH TIME ZONE, TIMEZ $0 \leq p \leq 10$	00:00:00+1459 24:00:00-1459	Timezone is converted to GMT/UTC. For example, 13:17:59+04:00 will be processed as 09:17:59+00:00.
TIMESTAMP[(p)] $0 \leq p \leq 6$  <b>Note:</b> Advanced SQL Engine supports SECOND > 60 (leap second), which is not supported by ML Engine.	$00 \leq \text{HOUR} \leq 23$ $00 \leq \text{MINUTE} \leq 59$ $00.000000 \leq \text{SECOND} \leq 61.999999$ $1 \leq \text{YEAR} \leq 9999$ $1 \leq \text{MONTH} \leq 12$ $1 \leq \text{DAY} \leq 31$	TIMESTAMP[(p)] WITHOUT TIME ZONE $0 \leq p \leq 6$	4713 BC to 5,874, 897 AD	No
TIMESTAMP[(p)] WITH TIMEZONE $0 \leq p \leq 6$	$00 \leq \text{HOUR} \leq 23$ $00 \leq \text{MINUTE} \leq 59$	TIMESTAMP[(p)] WITH TIME ZONE, TIMESTAMPZ	4713 BC to 5,874, 897 AD	Timezone is converted to GMT/UTC. For

Advanced SQL Engine Type	Advanced SQL Engine Range	Default ML Engine Type	ML Engine Range	Expected Data Loss in Conversion
<b>Note:</b> Advanced SQL Engine supports SECOND > 60 (leap second), which is not supported by ML Engine.	00.000000<=SECOND<=61.999999 1<=YEAR<=9999 1<=MONTH<=12 1<=DAY<=31 -12.59<=TIMEZONE_HOUR<=+14.00 -12.59<=TIMEZONE_MINUTE<=+14.00	0<=p<=6		example, 13:17:59+04:00 will be processed as 09:17:59+00:00.
BYTE( <i>n</i> )	The maximum value for <i>n</i> is 64,000.	BYTEA	Variable length with limit: the maximum length is 10 MB.	No
VARBYTE( <i>n</i> )	The maximum value for <i>n</i> is 64,000.	BYTEA	Variable length with limit: the maximum length is 10 MB.	No
BLOB( <i>n</i> K/M/G) *	The maximum number of bytes is 2,097,088,000, which is the default if <i>n</i> is not specified. K - <i>n</i> is specified in kilobytes (KB). When K is specified, <i>n</i> cannot exceed 2,047,937. M - <i>n</i> is specified in megabytes (Mb). When M is specified, <i>n</i> cannot exceed 1,999. G - <i>n</i> is specified in gigabytes (GB). When G is specified, <i>n</i> must be 1.			
INTERVAL YEAR[( <i>p</i> )] TO MONTH 1<=p<=4 *	Range for YEAR depends on value of <i>p</i> . MONTH=11			
INTERVAL MONTH[( <i>p</i> )] 1<=p<=4 *	Range depends on value of <i>p</i> .			
INTERVAL DAY[( <i>p</i> )] 1<=p<=4 *	Range depends on value of <i>p</i> .			

Advanced SQL Engine Type	Advanced SQL Engine Range	Default ML Engine Type	ML Engine Range	Expected Data Loss in Conversion
INTERVAL DAY[(p)] TO HOUR 1<=p<=4 *	Range for DAY depends on value of p. HOUR=23			
INTERVAL DAY[(p)] TO MINUTE 1<=p<=4 *	Range for DAY depends on value of p. HOUR=23 MINUTE=59			
INTERVAL DAY[(p)] TO SECOND[(n)] 1<=p<=4 1<=n<=6 *	Range for DAY depends on value of p. HOUR=23 MINUTE=59 SECOND=59 Value of n=9			
INTERVAL HOUR[(p)] 1<=p<=4 *	Range depends on value of p.			
INTERVAL HOUR[(p)] TO MINUTE 1<=p<=4 *	Range for HOUR depends on value of p. MINUTE=59			
INTERVAL HOUR[(p)] TO SECOND[(n)] 1<=p<=4 1<=n<=6 *	Range for HOUR depends on value of p. MINUTE=59 SECOND=59 Value of n=9			
INTERVAL MINUTE[(p)] 1<=p<=4 *	Range depends on value of p.			
INTERVAL MINUTE[(p)] TO SECOND[(n)] 1<=p<=4 1<=n<=6	Range for MINUTE depends on value of p. SECOND=59 Value of n=9			

Advanced SQL Engine Type	Advanced SQL Engine Range	Default ML Engine Type	ML Engine Range	Expected Data Loss in Conversion
*				
INTERVAL SECOND[(p, s)] 1<=p<=4 1<=s<=6 *	Range depends on value of p. Value of s=9			
PERIOD(DATE) *	User specify the beginning(inclusive) and ending(exclusive) bound.			
PERIOD(TIME[(p)]) 0<=p<=6 *	User specify the beginning(inclusive) and ending(exclusive) bound.			
PERIOD(TIME[(p)] WITH TIME ZONE) 0<=p<=6 *	User specify the beginning(inclusive) and ending(exclusive) bound.			
PERIOD(TIMESTAMP[(p)]) 0<=p<=6 *	User specify the beginning(inclusive) and ending(exclusive) bound.			
PERIOD(TIMESTAMP[(p)] WITH TIME ZONE) 0<=p<=6 *	User specify the beginning(inclusive) and ending(exclusive) bound.			
UDT(User Defined Type) *				
ARRAY / VARRAY *	1 DIMENSION, n DIMENSIONS 2<=n<=5			
TD_ANYTYPE *				



Advanced SQL Engine Type	Advanced SQL Engine Range	Default ML Engine Type	ML Engine Range	Expected Data Loss in Conversion
VARIANT_TYPE *				
JSON *	<p>The absolute maximum length is 16,776,192 bytes.</p> <ul style="list-style-type: none"> <li>For LATIN, the maximum length is 16,776,192 characters.</li> <li>For UNICODE, the maximum length is 8,388,096 characters.</li> </ul>			
XML / XMLTYPE *				
ST_Geometry *				
DATASET *				
GRAPHIC(n) / CHARACTER(n) CHARACTER SET GRAPHIC *	Maximum value for n is 32,000.			
VARGRAPHIC(n) / VARCHAR(n) CHARACTER SET GRAPHIC *	Maximum value for n is 32,000.			

**Note:**

\* For Advanced SQL Engine Type without corresponding Default ML Engine Type, the conversion raises an error.

## ML Engine to Advanced SQL Engine

ML Engine Type	ML Engine Range	Default Advanced SQL Engine Type	Advanced SQL Engine Range	Expected Data Loss in Conversion
SMALLINT / INT2	-32,768 to +32,767	SMALLINT	-32,768 to 32,767	No
INTEGER / INT4	-2,147,483,648 to +2,147,483,647	INTEGER	-2147483648 to 2147483647	No
BIGINT / INT8	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	BIGINT	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	No
DECIMAL( $p$ , $s$ ) / NUMERIC( $p$ , $s$ ) / NUMERIC / NUMERIC( $p$ ) $1 \leq p \leq 1000$ $0 \leq s \leq p$ DECIMAL === NUMERIC === DECIMAL(1000, $s$ ) for all possible values of $s$ DECIMAL( $p$ ) === NUMERIC( $p$ ) === NUMERIC( $p$ , 0)  <b>Note:</b> <ul style="list-style-type: none"> <li><math>p &gt; 38</math> is not supported.</li> <li>Unbounded decimal / numeric defined as DECIMAL / NUMERIC is not supported.</li> </ul>	<ul style="list-style-type: none"> <li>The maximum value for DECIMAL(<math>p</math>, <math>s</math>) is a value consisting of <math>p</math> 9s, with the decimal point <math>s</math> digits from the right.</li> <li>The minimum value is the negative of the maximum value.</li> </ul>	NUMERIC( $P$ ) / NUMERIC( $p$ , $s$ ) $p$ : total number of digits $s$ : number of digits to the right of the decimal point $1 \leq p \leq 38$ $0 \leq s \leq p$	<ul style="list-style-type: none"> <li>The maximum value for NUMERIC(<math>p</math>, <math>s</math>) is a value consisting of <math>p</math> 9s, with the decimal point <math>s</math> digits from the right.</li> <li>The minimum value is the negative of the maximum value.</li> </ul>	No

ML Engine Type	ML Engine Range	Default Advanced SQL Engine Type	Advanced SQL Engine Range	Expected Data Loss in Conversion
REAL / FLOAT4 / FLOAT( <i>p</i> ) 1<= <i>p</i> <=24	1.0 * 10 <sup>-37</sup> to 1.0 * 10 <sup>+37</sup> with a precision of at least 6 decimal digits. Infinity -Infinity NaN	FLOAT	Values in sign/magnitude form ranging from 2.226 * 10 <sup>-308</sup> to 1.797 * 10 <sup>+308</sup>	No
DOUBLE PRECISION / FLOAT8 / FLOAT( <i>p</i> ) 25<= <i>p</i> <=53	1.0 * 10 <sup>-307</sup> to 1.0 * 10 <sup>+308</sup> with a precision of at least 15 decimal digits. Infinity -Infinity NaN	FLOAT	Values in sign/magnitude form ranging from 2.226 * 10 <sup>-308</sup> to 1.797 * 10 <sup>+308</sup>	No
CHARACTER VARYING( <i>n</i> ) / VARCHAR( <i>n</i> )	Variable length with limit: the maximum length is 10 MB.	<ul style="list-style-type: none"> <li>If <i>n</i> &lt; 32k characters for UNICODE (64k characters for LATIN): VARCHAR(<i>n</i>)</li> <li>If <i>n</i> &gt; 32k characters for UNICODE (64k characters for LATIN): CLOB</li> </ul>	<ul style="list-style-type: none"> <li>LATIN: <i>n</i> &lt;= 2,097,088,000.</li> <li>UNICODE: <i>n</i> &lt;= 1,048,544,000.</li> </ul>	Strings will be truncated if they are larger than the value of <i>Default String Size</i> property of the ML Engine Connector in QueryGrid.
VARCHAR	Variable length with limit: the maximum length is 10 MB.	CLOB	<ul style="list-style-type: none"> <li>LATIN: <i>n</i> &lt;= 2,097,088,000.</li> <li>UNICODE: <i>n</i> &lt;= 1,048,544,000.</li> </ul>	Strings will be truncated if they are larger than the value of <i>Default String Size</i> property of the ML Engine Connector in QueryGrid.

ML Engine Type	ML Engine Range	Default Advanced SQL Engine Type	Advanced SQL Engine Range	Expected Data Loss in Conversion
CHARACTER( <i>n</i> ) / CHAR( <i>n</i> ) / CHARACTER / CHAR	Fixed length with blank padded: the maximum length is 10 MB. default: <i>n</i> ::1	<ul style="list-style-type: none"> <li>If <i>n</i> &lt; 32k characters for UNICODE (64k characters for LATIN): CHARACTER / CHAR(<i>n</i>)</li> <li>If <i>n</i> &gt; 32k characters for UNICODE (64k characters for LATIN): CLOB</li> </ul> default: <i>n</i> ::1	<ul style="list-style-type: none"> <li>If <i>n</i> &lt; 32k characters for UNICODE (64k characters for LATIN):               <ul style="list-style-type: none"> <li>For the LATIN server character set, the maximum value for <i>n</i> is 64,000 characters.</li> <li>For the UNICODE and GRAPHIC server character sets, the maximum value for <i>n</i> is 32,000 characters.</li> <li>For the KANJISJIS server character set, the maximum value for <i>n</i> is 32,000 bytes.</li> </ul> </li> <li>If <i>n</i> &gt; 32k characters for UNICODE (64k characters for LATIN):               <ul style="list-style-type: none"> <li>LATIN: <i>n</i> &lt;= 2,097,088,000.</li> <li>UNICODE: <i>n</i> &lt;= 1,048,544,000.</li> </ul> </li> </ul>	Strings will be truncated if they are larger than the value of <i>Default String Size</i> property of the ML Engine Connector in QueryGrid.
DATE <b>Note:</b> ML Engine can have YEAR > 9999 which is not supported by Advanced SQL Engine.	4713 BC to 5,874,897 AD	DATE	YEAR-MONTH-DAY 1<=YEAR<=9999 1<=MONTH<=12 1<=DAY<=31	No
TIME[( <i>p</i> )] WITHOUT TIME ZONE 0<= <i>p</i> <=6	00:00:00 24:00:00	TIME[( <i>p</i> )] 0<= <i>p</i> <=6	00<=HOUR<=23 00<=MINUTE<=59 00.000000<=SECOND<=61.999999	No
TIME[( <i>p</i> )] WITH TIME ZONE, TIMEZ 0<= <i>p</i> <=10	00:00:00+1459 24: 00:00-1459	TIME[( <i>p</i> )] WITH TIME ZONE 0<= <i>p</i> <=6	000<=HOUR<=23 00<=MINUTE<=59 00.000000<=SECOND<=61.999999 -12.59<=TIMEZONE_HOUR<=+14.00	No

ML Engine Type	ML Engine Range	Default Advanced SQL Engine Type	Advanced SQL Engine Range	Expected Data Loss in Conversion
<b>Note:</b> TIMEZONE range is different in ML Engine and Advanced SQL Engine.			-12.59<=TIMEZONE_MINUTE<=+14.00	
TIMESTAMP[(p)] WITHOUT TIME ZONE 0<=p<=6  <b>Note:</b> ML Engine can have YEAR > 9999 which is not supported by Advanced SQL Engine.	4713 BC to 5,874,897 AD	TIMESTAMP[(p)] 0<=p<=6	00<=HOUR<=23 00<=MINUTE<=59 00.000000<=SECOND<=61.999999 1<=YEAR<=9999 1<=MONTH<=12 1<=DAY<=31	No
TIMESTAMP[(p)] WITH TIME ZONE, TIMESTAMPZ 0<=p<=6  <b>Note:</b> ML Engine can have YEAR > 9999 which is not supported by Advanced SQL Engine.  <b>Note:</b> TIMEZONE range is different in ML Engine and Advanced SQL Engine.	4713 BC to 5,874,897 AD	TIMESTAMP[(p)] WITH TIMEZONE 0<=p<=6	00<=HOUR<=23 00<=MINUTE<=59 00.000000<=SECOND<=61.999999 1<=YEAR<=9999 1<=MONTH<=12 1<=DAY<=31 -12.59<=TIMEZONE_HOUR<=+14.00 -12.59<=TIMEZONE_MINUTE<=+14.00	No
BOOLEAN	TRUE/FALSE/NULL	BYTEINT	-128 to 127	No

ML Engine Type	ML Engine Range	Default Advanced SQL Engine Type	Advanced SQL Engine Range	Expected Data Loss in Conversion
TEXT	Variable length with limit: the maximum length is 32 MB.	CLOB	<ul style="list-style-type: none"> <li>LATIN: <math>n \leq 2,097,088,000</math>.</li> <li>UNICODE: <math>n \leq 1,048,544,000</math>.</li> </ul>	Strings will be truncated if they are larger than the value of <i>Default String Size</i> property of the ML Engine Connector in QueryGrid.
BYTEA	Variable length with limit: the maximum length is 10 MB.	VARBYTE(32,000) <b>Note:</b> 32,000 is the default value of <i>Default Binary Size</i> property of the ML Engine Connector in QueryGrid.		Strings will be truncated if they are larger than the value of <i>Default Binary Size</i> property of the ML Engine Connector in QueryGrid.
SERIAL / SERIAL4	1 to 2147483647	INTEGER / INT4	-2,147,483,648 to +2,147,483,647	No
BIGSERIAL / SERIAL8	1 to 9,223,372,036,854,775,807	BIGINT / INT8	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	No
INTERVAL	-178,956,970 years to 178,956,970 years	VARCHAR(49)	49	No
BIT( <i>n</i> )	Value must match the length specified by <i>n</i> . The maximum length of bit is 10MB (83,866,080 bits).	CHAR( <i>n</i> )	<ul style="list-style-type: none"> <li>If <math>n &lt; 32k</math> characters for UNICODE (64k characters for LATIN):               <ul style="list-style-type: none"> <li>For the LATIN server character set, the maximum value for <i>n</i> is 64,000 characters.</li> <li>For the UNICODE and GRAPHIC server character sets, the maximum value for <i>n</i> is 32,000 characters.</li> </ul> </li> </ul>	No

ML Engine Type	ML Engine Range	Default Advanced SQL Engine Type	Advanced SQL Engine Range	Expected Data Loss in Conversion
			<ul style="list-style-type: none"> <li>For the KANJISJIS server character set, the maximum value for <math>n</math> is 32,000 bytes.</li> <li>If <math>n &gt; 32k</math> characters for UNICODE (64k characters for LATIN):               <ul style="list-style-type: none"> <li>LATIN: <math>n \leq 2,097,088,000</math>.</li> <li>UNICODE: <math>n \leq 1,048,544,000</math>.</li> </ul> </li> </ul>	
BIT VARYING( $n$ )	Value must match the length specified by $n$ . The maximum length of bit is 10MB (83,866,080 bits).	BLOB	2,097,088,000	No
UUID	Unique string with 32 digits	CLOB(10485760)	10485760	No
IP4 *				
IP4RANGE *				

**Note:**

\* For ML Engine Type without corresponding Default Advanced SQL Engine Engine Type, the conversion raises an error.

# Error Messages

## ML Engine Error Messages

Some ML Engine function names have changed. To map an error message function name with its new function name, see [ML Engine Analytic Function Mapping](#).

Error message text contains an error number, universally unique identifier (UUID), error tag, and error text.

- UUID is a unique identifier. As such, the UUID shown in your error message will not be the same as what appears in the table below.
- Error tag MLE\_ERR indicates the error was raised from ML Engine.
- Error tag MLE\_ERR\_FFE indicates that the error was raised by the ML Engine connector.
- Errors raised in the various functions executing in ML Engine are tagged with the function name.
- The presence of the UUID with no error tags indicates that the error was raised in Advanced SQL Engine.
- QueryID is a unique identifier, which is present in the error message if the ML Engine connector was able to log the error message and make it available in the view `td_mle_db.user_mle_errors`. It will be delimited by square brackets.

Error Message Text	Description and Solution
*** Error 9134 Error: 9101 (63e825e9-3d89-4327-b1a1-00000000001f) [307190494990685772] MLE_ERR_FFE: G_IntervalHour is not supported	A function attempted to return an unsupported datatype. Solution: It may be possible to correct the function to use a decimal with defined precision, or use CAST in the SQL syntax to cast the datatype to a supported datatype. Otherwise, contact Teradata Services.
Error 9134: QGIEC: 60104: (5c1e36df-bf96-4137-8ac7-000000000013f) [307190494990685772] FUNCTION_NAME: The column <i>column-name</i> in <i>function-argument</i> cannot be found. ().	The function argument expects a valid column name from an input table, but that column was not present. This issue can also be caused if the column name is longer than 63 characters. Solution: Ensure the column name is present in the input table and no longer than 63 characters.
Error 9134: QGIEC: 60104: (5c1e36df-bf96-4137-8ac7-000000000013f) [307190494990685772] MLE_ERR: column <i>column-name</i> already exists. ().	This error is caused by using a column name longer than 63 characters in one of the function's input tables. Solution: Ensure the column name in the input table is no longer than 63 characters.



Error Message Text	Description and Solution
Error 9134: QGIEC: 60104: (5c1e36df-bf96-4137-8ac7-0000000013f) [307190494990685772] MLE_ERR: column reference <i>column-name</i> is ambiguous. ().	This error is caused by using a column name longer than 63 characters in one of the function's input tables. Solution: Ensure the column name in the input table is no longer than 63 characters.
*** Error 9134 QGIEC: 60104: (f8b385b9-8f12-4da2-9d63-0000000010e) [307190494990685772] asterdart0595 : FFMeta : MLE_ERR_FFE: Column user_id in sequenceInputBy argument clause does not exist in the table with Alias <i>inputtable</i>	In the sequenceInputBy or UniqueId argument, the column name does not exist in the table given in the <i>inputtable</i> alias. Solution: Check the column name to make sure it is in the table.
*** Error 9134 QGIEC: 60104: (ab64f100-dfd9-4dde-8f74-00000000044) [307190494990685772] MLE_ERR_FFE: Column col1 in sequenceInputBy argument clause does not exist in the given table	In the sequenceInputBy or UniqueId argument, the column name does not exist in the table given. Solution: Check the column name to make sure it is in the table.
*** Failure 9134 QGIE: 24582: (d30ab077-53b9-4c24-928a-00000000020) [307190494990685772] MLE_ERR: An internal error has occurred. Please contact support at Teradata. ()	An internal error may be encountered during query interactions. Solution: Resubmit the query. If it fails again, contact Teradata Services.
*** Failure 9134 QGIE: 24582: (4ec0fb9f-4cfe-4cbe-8f52-000000000661) [307190494990685772] tdap1513t2 : ExpDE : MLE_ERR: The supplied statement could not be completed due to an internal error at a database worker node. Please contact support at Teradata. ()	An internal error may be encountered during query interactions. Solution: Resubmit the query. If it fails again, contact Teradata Services.
*** Failure 9134 QGIE: 24582: (07955e5e-9b89-4f47-b91b-00000000aabc) [307190494990685772] MLE_ERR: The supplied statement could not be completed due to an internal error at the database coordinator node. Please contact support at Teradata. ()	An internal error may be encountered during query interactions. Solution: Resubmit the query. If it fails again, contact Teradata Services.
*** Error 9134 QGIE: 60104: (beea1fb3-4202-4e81-896c-0000000009db) [307190494990685772] tdap656t2 : ForeignFunctionMeta : MLE_ERR_FFE: Function <i>function name</i> cannot take input from a nested function.	<i>function name</i> is the name of the function that cannot take input from a nested function. Solution: Rewrite the SQL statement to place the results of the nested function into a derived table and let the outer function take its input from the derived table. See <a href="#">Limitation</a> for an example.
*** Error 9134 QGInitiatorExportContract: 9101 (829113ff-e7ac-4179-b694-00000000018) [307190494990685772] MLE_ERR_FFE: Alias <i>table alias</i> appears more than once	The same table alias was used more than once in a function invocation. Solution: Resubmit the query with a different name for each table alias.

Error Message Text	Description and Solution
Failure 9134 QGII: 24582: (8d520b5f-d2c4-4e9d-b9c0-00000000124b) [307190494990685772] MLE_ERR_FFE: No help available; JSON function descriptor was not found	The JSON function descriptor of the function was not found. If the function came from Teradata, contact Teradata Services. Otherwise, for a user defined function, package a JSON function descriptor file with the function archive and reinstall the function.
*** Error 9134 QGInitiatorImport: 6 (f743340f-d4ae-49d4-8bce-00000000001b) [307190494990685772] MLE_ERR_FFE: JsonElement <i>element-name</i> not found in function descriptor.	Incomplete information in the function descriptor file. Solution: Correct the function descriptor to supply the required information, or if the function descriptor came from Teradata, contact Teradata Services.
*** Error 9134 QGInitiatorExportContract: 9101 (f743340f-d4ae-49d4-8bce-000000000019) [307190494990685772] MLE_ERR_FFE:Exception in processing the function descriptor: JsonElement <i>element-name</i> not found in function descriptor.	Incomplete information in the function descriptor file. Solution: Correct the function descriptor to supply the required information, or if the function descriptor came from Teradata, contact Teradata Services.
*** Failure 9134 Error: 6 (d8264ce0-43a5-4dbd-a8a6-00000000009dd) [307190494990685772] MLE_ERR: Schema doesn't exist.	HELP FOREIGN SCHEMA statement referenced a non-existent schema. Solution: Correct the schema name and resubmit the query.
*** Failure 9134 Error: 6 (56567b0106-3fff-47a1-acbf-0000000000168) [307190494990685772] MLE_ERR_FFE: Function ' <i>FunctionName</i> ' not found. Please verify that the function is installed.	HELP FOREIGN FUNCTION was invoked for a function that is not installed. Solution: Verify the function is installed.
*** Failure 9134 Error: 6 (b41e68d0-0305-4f4d-8911-0000000000df3) [307190494990685772] MLE_ERR_FFE: Function Alias name (' <i>alias</i> ') exceeds the maximum length of 25 characters.	A statement using a coprocessor function was written to refer to the function invocation with an alias name longer than the maximum supported length. Solution: Resubmit the statement with the function invocation alias name no longer than 25 characters. A function invocation alias is the name that follows the closing parenthesis that follows the name of the function.
*** Failure 9134 Error 6: (d8264ce0-43a5-4dbd-a8a6-00000000009bc) [307190494990685772] QGEXECUTEFOREIGNQUERY: Cannot read hostname.	There is a problem reading hostname from a file in /tmp/hostname in the Docker container. Solution: Contact Teradata Services.

Error Message Text	Description and Solution
<p>*** Failure 9134 Error 6: (d8264ce0-43a5-4dbd-a8a6-000000009ab) [307190494990685772] QGREMOTEEEXPORT: Column <i>name</i> is of unsupported type <i>type</i></p>	<p>The datatype of a column is not supported. Solution: It may be possible to redefine the column with a supported datatype, or use a CAST in the SQL syntax to cast the datatype to a supported datatype. Otherwise, contact Teradata Services.</p>
<p>*** Failure 9134 Error: 9101 (d8567b0106-3fff-47a1-acbf-00000000168) [307190494990685772] MLE_ERR_FFE: Function <i>FunctionName</i> is missing a ON clause.</p>	<p>The function call is missing required input tables. Solution: Correct the syntax.</p>
<p>*** Failure 9134 Error: 60104 (63e825e9-3d89-4327-b1a1-0000000001d) [307190494990685772] MLE_ERR: relation "dsdfs" does not exist</p>	<p>If the error was encountered when executing the HELP FOREIGN TABLE statement, it referenced a non-existent table. Solution: Correct the table name and resubmit the query. If the error was encountered during a function execution, resubmit the query. If the failure is seen again, contact Teradata Services</p>
<p>*** Failure 9134 Error: 60104 (eb6397be-a020-4277-a1be-000000000457) [307190494990685772] MLE_ERR_FFE: Column: <i>name</i> of <i>type</i> datatype is not supported</p>	<p>An attempt was made to export an unsupported datatype. Solution: It may be possible to use CAST in the SQL syntax to cast the datatype to a supported datatype. Otherwise, contact Teradata Services.</p>
<p>*** Failure 9134 Error: 60104 (d8264ce0-43a5-4dbd-a8a6-0000000009dc) [307190494990685772] MLE_ERR_FFE: Precision range of column <i>name</i> cannot be more than 38.</p>	<p>A function attempted to return a datatype with a precision greater than 38. Solution: It may be possible to correct the function to use a datatype with less precision, or use CAST in the SQL syntax to cast the datatype to a supported datatype. Otherwise, contact Teradata Services.</p>
<p>*** Failure 9134 Error: 60104 (65b09610-8cc4-4f40-9e1b-000000000b4b) [307190494990685772] MLE_ERR_FFE:Unbounded G_Decimal is not supported</p>	<p>A function attempted to return an unbounded decimal type. Solution: It may be possible to correct the function to use a decimal with defined precision, or use CAST in the SQL syntax to cast the datatype to a supported datatype. Otherwise, contact Teradata Services.</p>
<p>*** Failure 9134 Error: 60109 (56f921d4-bc08-49e1-baba-000000000004) [307190494990685772] MLE_ERR_FFE: Error occurred while validating the connection.:</p>	<p>This error occurred while validating the connection. Solution: Contact Teradata Services.</p>

Error Message Text	Description and Solution
Unable to create JDBC connection. Role "proxyuser" does not have connect privileges on this database. ()	
*** Failure 9134 Error: 60129 (94255d29-3ab7-4bc4-b533-000000000110) [307190494990685772] Error occurred while executing the Foreign Function Execution[Teradata Database][TeraJDBC 15.10.00.07] [Error 3803][SQL State 42S01] Table <i>table_name</i> already exists.()	This error occurred while executing the Foreign Function Execution. Solution: Drop the table <i>table_name</i> and resubmit the query.

## Viewing Errors in `td_mle_db.user_mle_errors`

When the ML Engine connector is able to log the error message and make it available in the view `td_mle_db.user_mle_errors`, you can retrieve the error and information about its context with a SQL query. This is especially helpful in the case of long error messages that are truncated by the Advanced SQL Engine, as shown in this example:

```

** Failure 9134 QGIE: 24582: (c4df9fdf-0f79-48bd-a7f3-0000000039c0)
[307190312363806345] MLE_ERR_FFE: java.sql.SQLException: [Error 9134]
QGII: 24582: (c4df9fdf-0f79-48bd-a7f3-0000000039c2) QGREMOTEEEXPORT:
com.asterdata.ncluster.sqlmr.ClientVisibleException: com.asterdat

```

To retrieve the full error message, you can write a query using the queryID shown in the message:

```

BTEQ -- Enter your SQL request or BTEQ command:
.set width 320;

```

```

BTEQ -- Enter your SQL request or BTEQ command:
select errorText from td_mle_db.user_mle_errors where queryId =
307190312363806345;

```

```

*** Query completed. One row found. One column returned.
*** Total elapsed time was 1 second.

```

```
errorText
```

```
-----
```

```

MLE_ERR_FFE: java.sql.SQLException: [Error 9134] QGII: 24582: (c4df9fdf-0f79-48bd-
a7f3-0000000039c2) QGREMOTEEEXPORT:
com.asterdata.ncluster.sqlmr.ClientVisibleException:
com.asterdata.ncluster.sqlmr.ClientVisibleException:
java.lang.UnsupportedOperationException: Column b is of unsupported type ip4 ()

```

**Note:**

The table underlying this view is defined with a trigger to retain only the 300 most recent errors for a user.

**User-Defined Functions Error Messages**

User-defined error messages occur in the context of invoking a pm.stored\_procedure. For example, this call installs a user-defined function in ML Engine:

```
call pm.install_afunction('kmeans_uif');
```

Use this table to map substitutions of the variable shown in the error messages:

Variable	Variable value can be replaced by:
<i>pm.stored_procedure</i>	pm.download_afile pm.download_afile_from_public pm.install_afile pm.install_afile_to_public pm.install.afunction pm.install_afunction_to_public pm.remove_afile pm.remove_afile_from_public pm.remove_afunction pm.remove_afunction_from_public
<i>pm.install_procedure</i>	pm.install_afile pm.install_afile_to_public pm.install_afunction pm.install_afunction_to_public
<i>pm.install_function</i>	pm.install_afunction pm.install_afunction_to_public
<i>pm.install_file</i>	pm.install_afile pm.install_afile_to_public
<i>pm.remove_procedure</i>	pm.remove_afile pm.remove_afile_from_public pm.remove_afunction pm.remove_afunction_from_public
<i>pm.download_procedure</i>	pm.download_afile pm.download_afile_from_public

Error Message Text	Description and Solution
Executed as Single statement. Failed [7825 : 38000]   EXECUTEFOREIGNSQL: The user does not have SELECT access to TD_(SERVER_DB.coprocessor_ddL) Elapsed time = ... STATEMENT 1: Unknown failed.	User tried to install, uninstall, or download UDF/file without appropriate privilege. Solution: Ask privileged user to grant the user the appropriate access privileges.
*** Failure 3523 The user does not have EXECUTE PROCEDURE access to <i>pm.stored-procedure</i> .	The user does not have the proper access rights to execute this request. Solution: Obtain the necessary access rights.
*** Failure 5660 Cannot create index on LOB columns.	The first column of the SELECT statement cannot be of data type CLOB or BLOB. In some functions, first output column is appended from input table (through Accumulate or PartitionColumns arguments). For such cases, either choose a column with a different data type or change the order of columns in the SELECT statement (for example, SELECT col2,col1 FROM <func_name instead of SELECT * FROM func_name or SELECT col1,col2 FROM func_name). Another option is to create an output table from the SELECT statement with NO PRIMARY INDEX or an index on a column which is not a LOB column.
*** Failure 7825 in UDF/XSP/UDM PM.remove_afile_from_public: SQLSTATE 38U78: Either the file <i>file_name</i> does not exist in ML Engine or the user does not have permission to uninstall the file.	An attempt was made to remove a file that does not exist or the user does not have the proper permission. Solution: Check to see if the file exists or if you have the required permission to delete the file.
*** Failure 7825 in UDF/XSP/UDM <i>pm.stored-procedure</i> : SQLSTATE 38U79: empty result from: <i>internal query</i>	Internal operation produced an empty result after a retry. Solution: Try the <i>pm.stored_procedure</i> again. If the problem persists, contact Teradata Services. Give them your Teradata QueryGrid Manager node hostname to help diagnose the problem.
*** Failure 7825 in UDF/XSP/UDM <i>pm.remove-procedure</i> : SQLSTATE 38U81: uninstall <i>function_name</i> failed	Failure to remove a function or file caused an internal error. Solution: Contact Teradata Services. Give them your Teradata QueryGrid Manager node hostname to help diagnose the problem.
*** Failure 7825 in UDF/XSP/UDM PM.remove_afunction_from_public: SQLSTATE 38U82: Either the function <i>function_name</i> does not exist in ML	An attempt was made to remove a function that does not exist or the user does not have the proper permission.

Error Message Text	Description and Solution
Engine or the user does not have permission to uninstall the function.	Solution: Check to see if the function exists or if you have the required permission to delete the function.
*** Failure 7825 in UDF/XSP/UDM <i>pm.download_procedure</i> : SQLSTATE 38U83: File <i>file_name</i> is not found in <i>schema_name</i> schema.	An attempt was made to download a file that does not exist on a named ML Engine schema. Solution: Replace the reference to nonexistent file with the correct one.
*** Failure 7825 in UDF/XSP/UDM <i>pm.stored_procedure</i> : SQLSTATE 38U84: User <i>public</i> is not allowed.	"public" was used as a user name. Solution: "public" cannot be used as a user name. Use a different user name.
*** Failure 7825 in UDF/XSP/UDM <i>pm.stored_procedure</i> : SQLSTATE 38U87: <i>uif_name</i> must not have '#'.	There is an invalid character in the name of the user installed file (uif). Solution: Correct the file name.
*** Failure 7825 in UDF/XSP/UDM <i>pm.install_file</i> : SQLSTATE 38U88: File <i>file_name</i> already exists in Machine Learning Engine.	An attempt was made to install a file that already exists. Solution: Remove the file, then install its replacement.
*** Failure 7825 in UDF/XSP/UDM <i>pm.install_function</i> : SQLSTATE 38U89: Function <i>function_name</i> already exists in Machine Learning Engine.	An attempt was made to install a function that already exists. Solution: Remove the function, then install its replacement.
*** Failure 7825 in UDF/XSP/UDM <i>pm.install_function</i> : SQLSTATE 38U90: <i>Linux command</i> failed: <i>error message</i>	An internal Linux command IOException occurred during a UDF installation. Solution: Contact Teradata Services. Give them your Teradata QueryGrid Manager node hostname to help diagnose the problem.
*** Failure 7825 in UDF/XSP/UDM <i>pm.install_function</i> : SQLSTATE 38U91: file IOException not found: <i>error message</i>	Internal operation to read a json file failed during a UDF installation. Solution: Contact Teradata Services. Give them your Teradata QueryGrid Manager node hostname to help diagnose the problem.
*** Failure 7825 in UDF/XSP/UDM <i>pm.stored_procedure</i> : SQLSTATE 38U92: <i>ssh tdqg@sshtdmle mkdir -p /opt/teradata/tdqg/connector/tdqg-tdmle-connector/UDF/fromTD/alice</i> failed with return code 255.	Internal operation " <i>ssh tdqg@sshtdmle...</i> " failed because passwordless <i>ssh/scp</i> between Advanced SQL Engine internal user and ML Engine is not working. Solution: Have your Customer Support Representative set up a passwordless <i>ssh/scp</i> between Advanced SQL Engine and ML Engine.



Error Message Text	Description and Solution
*** Failure 7825 in UDF/XSP/UDM <i>pm.install_function</i> : SQLSTATE 38U93: unzip <i>path</i> failed: <i>error message</i>	An internal unzip IOException occurred during a UDF installation. Solution: Contact Teradata Services. Give them your Teradata QueryGrid Manager node hostname to help diagnose the problem.
*** Failure 7825 in UDF/XSP/UDM <i>pm.install_function</i> : SQLSTATE 38U94: cp <i>./path/tmp/_udfInstallInput..</i> <i>.</i> failed with return code 1	An internal Linux command got a non zero return code during a UDF installation. Solution: Retry the operation. If the error persists, contact Teradata Services.
*** Failure 7825 in UDF/XSP/UDM <i>pm.stored_procedure</i> : SQLSTATE 38U95: Please CALL SYSUIF. INSTALL_FILE ( <i>file_name</i> ) first.	An attempt was made to install, remove, or download a function without installing its input user installed file (uif) first. Or, the name of the input uif is over 30 characters. Solution: If input uif does not exist, install the input uif of the function first. For example: <pre>call sysuif.install_afile ( 'kmeans_uif', 'kmeans.zip', 'cb! /tmp/kmeans.zip' );</pre> If the input uif name is over 30 characters long, use a shorter name.
*** Failure 7825 in UDF/XSP/UDM <i>pm.install_function</i> : SQLSTATE 38U96: walkFileTree <i>path</i> failed: <i>error message</i>	An internal walkFileTree IOException occurred during a UDF installation. Solution: Contact Teradata Services. Give them your Teradata QueryGrid Manager node hostname to help diagnose the problem.
*** Failure 7825 in UDF/XSP/UDM <i>pm.install_function</i> : SQLSTATE 38U97: SearchAndUnzip <i>path</i> failed: <i>error message</i>	An internal SearchAndUnzip IOException occurred during a UDF installation. Solution: Contact Teradata Services. Give them your Teradata QueryGrid Manager node hostname to help diagnose the problem.
*** Failure 7825 in UDF/XSP/UDM <i>pm.install_procedure</i> : SQLSTATE 38U98: call syslib. ExecuteForeignSQL ( 'install file <i>file_name</i> failed: file <i>file_name</i> already exists in schema "public" ).	An attempt was made to install a public function that already exists. Solution: Remove the public function or file, then install its replacement.
*** Failure 7825 in UDF/XSP/UDM <i>pm.install_function</i> : SQLSTATE 38U99: create temp dir <i>path</i> gets <i>error message</i>	An internal create temp dir IOException occurred during a UDF installation. Solution: Contact Teradata Services. Give them your Teradata QueryGrid Manager node hostname to help diagnose the problem.



Error Message Text	Description and Solution
*** Failure 7827 Java SQL Exception SQLSTATE 39001: Invalid SQL State(HY000[Error 9134] : [Teradata Database] [TeraJDBC 16.20.00.02] [Error 9134] [SQLState HY000] Error: 60104 (bf774d8a-a2c5-424d-9d68-000000000030) invalid user name and/or password ()).	An internal operation failed because proxyuser has not been set up on ML Engine. Solution: Have your Customer Support Representative set up proxyuser on ML Engine.

## Additional Information

### Changes and Additions

Date	Release	Description
May 2022	ML Engine 9.2.4	Added support for ML Engine 9.2.4
December 2021	<ul style="list-style-type: none"> <li>ML Engine 9.23</li> <li>ML Engine Analytic Functions 9.02</li> </ul>	<ul style="list-style-type: none"> <li>Added support for ML Engine 9.23 and ML Engine Analytic Functions 9.02</li> <li>Updated the Data Type Mapping section.</li> </ul>
October 2020	<ul style="list-style-type: none"> <li>ML Engine 9.02</li> <li>ML Engine Analytic Functions 9.01</li> </ul>	Updated <a href="#">Analytic Functions</a> to include changes for the new UDF Manager feature.
August 2020	<ul style="list-style-type: none"> <li>ML Engine 9.01</li> <li>ML Engine Analytic Functions 9.01</li> </ul>	Initial release

### Teradata Links

Link	Description
<a href="https://docs.teradata.com/">https://docs.teradata.com/</a>	Search Teradata Documentation, customize content to your needs, and download PDFs. Customers: Log in to access Orange Books.
<a href="https://support.teradata.com">https://support.teradata.com</a>	Helpful resources in one place: <ul style="list-style-type: none"> <li>Support requests</li> <li>Account management and software downloads</li> <li>Knowledge base, community, and support policies</li> <li>Product documentation</li> <li>Learning resources, including Teradata University</li> </ul>
<a href="https://www.teradata.com/University/Overview">https://www.teradata.com/University/Overview</a>	Teradata education network
<a href="https://support.teradata.com/community">https://support.teradata.com/community</a>	Link to Teradata community

### Related Documentation

Title	Publication ID
<i>Teradata® QueryGrid™ Installation and User Guide</i>	B035-5991
<i>Teradata® Server Management Product Guide</i>	B035-6112

Title	Publication ID
<i>Teradata Studio™ User Guide</i>	B035-2041
<i>Teradata Vantage™ - Advanced SQL Engine Analytic Functions</i>	B035-1206
<i>Teradata Vantage™ Machine Learning Engine Analytic Function Reference</i>	B700-4003
<i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>	B035-1144
<i>Teradata Vantage™ - SQL External Routine Programming</i>	B035-1147
<i>Teradata Vantage™ - SQL Data Manipulation Language</i>	B035-1146
<i>Teradata Vantage™ - SQL Functions, Expressions, and Predicates</i>	B035-1145
<i>Teradata Vantage™ User Guide</i>	B700-4002
<i>Teradata® Viewpoint User Guide</i>	B035-2206