**Clairvoyant**

**Hive Exercise**

# **Intro**

The Apache Hive™ data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. Structure can be projected onto data already in storage. A command line tool and JDBC driver are provided to connect users to Hive.

Website: <https://hive.apache.org/>

# **Getting started with Hive - Loading Data 1**

In this exercise we will load MovieLens dataset into a table in Hive and do some data analysis.

**Starting Hive**

1. Change directories to the directory containing the sample data for workshop (sampledata.zip unzipped).
2. Copy the movies.txt from the sampledata folder to workshop/hive/movies/input folder on **HDFS**

|  |
| --- |
| $ cd {location\_of\_sample\_data}/movies/  $ hadoop fs -mkdir -p workshop/hive/movies/  $ hadoop fs -put input workshop/hive/movies/ |

1. Create a movies table on Hive with the following DDL statement.

|  |
| --- |
| # first start a hive shell:  $ hive |

|  |
| --- |
| # Run the DDL statement:  hive> CREATE TABLE movies( id int,  title String,  releaseDate string, videoReleaseDate string, IMDBurl String,  unknown String, isAction tinyint,  isAdventure tinyint,  isAnimation tinyint,  isChildren tinyint, isComedy tinyint, isCrime tinyint, isDocumentary tinyint, isDrama tinyint, isFantash tinyint, isFilmNoir tinyint,  isHorror tinyint, isMusical tinyint,  isMystery tinyint,  isRomance tinyint,  isSciFi tinyint,  isThriller tinyint, isWar tinyint,  isWestern tinyint)  row format delimited fields terminated by '|' stored as textfile; |
|  |

This should create a table with the name movies with the columns as described in the create statement.

1. Load the data from movies.txt file from **HDFS** into the table

|  |
| --- |
| hive> LOAD DATA INPATH '/user/cloudera/workshop/hive/movies/input/movies.txt' OVERWRITE INTO TABLE movies; |

1. Query the table to confirm that data was loaded

|  |
| --- |
| hive> SELECT \* FROM movies; |

1. Retrieve the total number of movies in the table

|  |
| --- |
| hive> SELECT count(1) FROM movies; |

1. Retrieve a list of all the movies that are released in 1995

|  |
| --- |
| hive> SELECT count(1) FROM movies WHERE releaseDate like '%1995'; |

1. Retrieve the names of movies that are in children genre and animation genre

|  |
| --- |
| hive> SELECT title FROM movies WHERE isChildren=1 AND isAnimation=1; |

1. Create a new Hive table from the results of query on movies. Create a new table with all the children mas;ovies

|  |
| --- |
| hive> CREATE TABLE childrenmovies AS select \* from movies where isChildren=1; |

1. Query the table to see the results of the new table created.
2. Execute all the commands through the Hive Shell and through Hive Editor on Hue
   1. Hue can be accessed while on the VM at the URL: <http://localhost:8888/about/>
      1. Note: there is also a shortcut on the Bookmark bar

# **Getting started with Hive - Loading Data 2**

In this exercise we will load Apache Log data into a table in Hive using a more advanced table creation method and do some data analysis.

**Starting Hive**

1. Change directories to the directory containing the sample data for workshop.
2. Copy the access.log from the sampledata folder to workshop/hive/logs/input folder on **HDFS**

|  |
| --- |
| $ cd {location\_of\_sample\_data}/logfiles/ $ hadoop fs -mkdir -p workshop/hive/logs/  $ hadoop fs -put input workshop/hive/logs/ |

1. Load the access.log file data as an External table into Hive using the create command below

|  |
| --- |
| hive> CREATE EXTERNAL TABLE apachelog ( host STRING, identity STRING, user STRING, time STRING, request STRING, status STRING, size STRING, referer STRING, agent STRING) ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'WITH SERDEPROPERTIES ( "input.regex" = "([^ ]\*) ([^ ]\*) ([^ ]\*) (-|\\[[^\\]]\*\\]) ([^ \"]\*|\"[^\"]\*\") (-|[0-9]\*) (-|[0-9]\*)(?: ([^ \"]\*|\"[^\"]\*\") ([^ \"]\*|\"[^\"]\*\"))?", "output.format.string" = "%1$s %2$s %3$s %4$s %5$s %6$s %7$s %8$s %9$s")STORED AS TEXTFILE LOCATION '/user/cloudera/workshop/hive/logs/input/'; |

This should create a table with the name apachelog with 9 columns as described in the create statement.

1. Retrieve the number unique response codes present in the accesslogs and their counts

|  |
| --- |
| hive> SELECT status, count(1) FROM apachelog group by status; |

1. Retrieve the total number of requests in the access.log

|  |
| --- |
| hive> SELECT count(1) FROM apachelog; |

1. Retrieve the distinct agents that made calls to website

|  |
| --- |
| hive> SELECT DISTINCT agent FROM apachelog; |

1. Retrieve the list of unique requests in the apachelog and their counts.
2. Create a new Hive table from the results of query on apachelog. Create a new table with all the successful results from apachelog table

|  |
| --- |
| hive> CREATE TABLE successfulrequests AS SELECT \* FROM apachelog2 WHERE status=200; |

1. Query the table to see the results of the new table created.
2. Execute all the commands through the Hive Shell and through Hive Editor on Hue

# **Getting Started with Hive - Submitting Hive Queries through Bash**

Executing commands from the shell is useful, but when it comes to scripting the process to regularly run the same commands in a workflow or pipeline, it is inadequate. Hive provides the ability to execute an HQL (Hive Query Language) statement or execute an HQL file from the command line interface via the HiveCLI.

Executing an HQL statement using the execute (-e) argument:

|  |
| --- |
| $ hive -e "{HQL\_STATEMENT}" |

An example of what you can do with this is:

|  |
| --- |
| $ hive -e "SELECT \* FROM apachelog2 WHERE status=200" |

Hive also provides the ability for you to to put all your HQL statements in a file and execute that file using the file (-f) argument:

|  |
| --- |
| $ hive -f {PATH\_TO\_HQL\_FILE} |

As an example can be that we have a file called some\_file.hql (the name and file extension don’t matter) with the contents:

|  |
| --- |
| SELECT \* FROM apachelog2 WHERE status=200; |

We can then execute that file with the command:

|  |
| --- |
| $ hive -f some\_file.hql |