

Visa

Big Data 101

Student Lab 1



Web Age Solutions Inc.
USA: 1-877-517-6540
Canada: 1-866-206-4644
Web: <http://www.webagesolutions.com>

The following terms are trademarks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

IBM, WebSphere, DB2 and Tivoli are trademarks of the International Business Machines

Corporation in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

For customizations of this book or other sales inquiries, please contact us
at:

USA: 1-877-517-6540, email: getinfousa@webagesolutions.com

Canada: 1-866-206-4644 toll free, email: getinfo@webagesolutions.com

Copyright © 2022 Web Age Solutions Inc.

This publication is protected by the copyright laws of Canada, United States and any other country where this book is sold. Unauthorized use of this material, including but not limited to, reproduction of the whole or part of the content, re-sale or transmission through fax, photocopy or e-mail is prohibited. To obtain authorization for any such activities, please write to:

Web Age Solutions Inc.
439 University Ave
Suite 820
Toronto
Ontario, M5G 1Y8

Contents

Lab 1 – The Class Environment and tables - 1 Hour	4
Step 1. Connect to our Lab Environment	4
Step2. Working with Hive	8
Step 3. Working with MapReduce	13
Step 4. HiveQL Projection	17
Step 5. Working with Hadoop HDFS	19
Step 6. Hive Metastore and Hadoop HDFS.....	24
Step 7. How to set variables in HIVE scripts	26
Review	30

Lab 1 – The Class Environment and tables - 1 Hour

Overview

The Apache Hadoop and Hive software

In this lab we will work with Apache Hive through the Command Line Interface (CLI) using the HiveQL Language which is based upon Structured Query Language (SQL). Much of the lab data is already stored within the Hadoop Distributed File System (HDFS) in the /data directory. We will be discussing the various folders and files in the hierarchy as we continue throughout the class.

Step 1. Connect to our Lab Environment

Please connect to the assigned Nuvepro Virtual Machine where the labs will be run from.

On the Linux desktop you will see the HDP VM has already been started and is showing a couple of ways to login to the system.

Open a web browser on the Nuvepro Virtual Machine. Enter this address into the URL to open the SSH connection to HortonWorks HDP Sandbox.

<http://localhost:4200>



Login to the console and provide the password when prompted.

Login username: **wasadmin**
Password: **wasadmin**

We will now list the directory information using the Linux ll command (lowercase letter L) for long listing. You should see LabFiles folder.

11

Use the following command to change to that directory.

cd LabFiles

Hive has several features that can be used without running in it's CLI directly. Let's see the options available here. To get help, run

hive -help

Output like the following should be shown. Read through the various options.

Q: How would you use option -e or -f for instance?

Beeline Command Line Shell Options	Description
-u <database url>	JDBC URL to connect to Hive database
-r	Reconnect to last saved JDBC URL connection
-n <username>	Username
-p <password>	Password
-d <driver class>	Driver class to be used if any
-i <init file>	Script file for initialization of variables

<code>-e <query></code>	Query to be executed
<code>-f <exec file></code>	Execute script file
<code>-w (or) -password-file <password file></code>	Password file. Should provide full path of the file.
<code>-hiveconf property=value</code>	Set value for given property
<code>-hivevar name=value</code>	Hive variable name and value. You can use this variable inside HiveQL queries.
<code>-color=[true/false]</code>	Set whether color is used for display
<code>-showHeader=[true/false]</code>	Display header in the query output
<code>-headerInterval=ROWS;</code>	Interval between which headers are displayed
<code>-fastConnect=[true/false]</code>	Skip building table/column list for tab-completion
<code>-autoCommit=[true/false]</code>	Enable or disable automatic transaction commit
<code>-verbose=[true/false]</code>	Execute query in verbose mode
<code>-showWarnings=[true/false]</code>	Display connection warnings
<code>-showNestedErrs=[true/false]</code>	Display nested errors
<code>-numberFormat=[pattern]</code>	Format numbers using DecimalFormat pattern
<code>-force=[true/false]</code>	Continue script even after errors
<code>-maxWidth=MAXWIDTH</code>	Maximum width of the terminal
<code>-maxColumnWidth=MAXCOLWIDTH</code>	Maximum width to use when displaying columns
<code>-silent=[true/false]</code>	Execute in silent mode
<code>-autosave=[true/false]</code>	Automatically save preferences

<code>-outputformat=[table/vertical/csv2/tsv2/dsv/csv/tsv]</code>	Format mode for result display
<code>-incremental=[true/false]</code>	Defaults to false. When set to false, the entire result set is fetched and buffered before being displayed, yielding optimal display column sizing. When set to true, result rows are displayed immediately as they are fetched, yielding lower latency and memory usage at the price of extra display column padding. Setting <code>-incremental=true</code> is recommended if you encounter an OutOfMemory on the client side (due to the fetched result set size being large).
<code>-truncateTable=[true/false]</code>	Truncate table column when it exceeds column length
<code>-delimiterForDSV=DELIMITER</code>	Set delimiter for output format
<code>-isolation=LEVEL</code>	Set the transaction isolation level
<code>-nullemptystring=[true/false]</code>	Insert NULL for empty strings
<code>-addlocaldriverjar=DRIVERJARNAME</code>	Add driver jar file in the beeline client side
<code>-addlocaldrivername=DRIVERNAME</code>	Add driver name needs to be supported in the beeline client side
<code>-help</code>	Command help option
<code>-delimiter=</code>	Set output delimiter for data extraction

A: hive -e 'select count(*) from wasadmin_db.accounts'

- Executes the query directly passing it to Hive and uses MapReduce to count number of records in the accounts table.

A: hive -f count_account.q

- Uses MapReduce to count number of records in the accounts table by running identical query 'select count(*) from wasadmin_db.accounts' stored within the count_account.q file.

For further helpers Hortonworks has a handy cheat sheet, open and download it if you like from the following URL:

<http://hortonworks.com/wp-content/uploads/2016/05/Hortonworks.CheatSheet.SQLtoHive.pdf>

You will also find it on the VM under your user account **LabFiles/resources** folder as [Hortonworks.CheatSheet.SQLtoHive.pdf](#)

Don't forget to use the course reference book at the O'Reilly site:



[Programming Hive Chapter 1 Introduction to Apache Hive](#)

Step2. Working with Hive

We will now launch Hive from the LabFiles folder. Type the hive command to run it on the VM SSH.

hive

The output will be like what is shown below.

```
[wasadmin@sandbox-hdp LabFiles]$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/hdp/3.0.1.0-187/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/hdp/3.0.1.0-187/hadoop/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://sandbox-hdp.hortonworks.com:2181/default;password=hive;serviceDiscoveryMode=zooKeeper;user=hive;zooKeeperNamespace=hiveserver2
22/01/16 09:08:58 [main]: INFO jdbc.HiveConnection: Connected to sandbox-hdp.hortonworks.com:10000
Connected to: Apache Hive (version 3.1.0.3.0.1.0-187)
Driver: Hive JDBC (version 3.1.0.3.0.1.0-187)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 3.1.0.3.0.1.0-187 by Apache Hive
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> □
```

Specifically, the version of Hive Beeline 3.1.0.3.0.1.0-187 on this HDP sandbox is shown with the command prompt >


```
Beeline version 3.1.0.3.0.1.0-187 by Apache Hive  
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> □
```

Next let's review what databases are setup on the Hive Server.

show databases;

There will be some INFO lines shown and then the result.

```
+-----+  
| database_name |  
+-----+  
| default       |  
| foodmart      |  
| information_schema |  
| sys           |  
| wasadmin_db   |  
+-----+  
5 rows selected (0.355 seconds)
```

We will utilize the wasadmin_db database for this lab. To set it type:

use wasadmin_db;

The database and tables were created earlier. First the various data file types were loaded into Hadoop HDFS, then commands for Hive were run to load that HDFS data into a schema. (Covered later in another lab).

Now we can review the tables that are already loaded into Hive. Type the command to show the tables.

show tables;

```

+-----+
|  tab_name  |
+-----+
| accounts   |
| offers     |
| transactions |
| vendors    |
| x_clickstream |
| x_domains  |
+-----+
6 rows selected (0.068 seconds)

```

Notice the range of INFO messages shown. This has been set for the Lab environment to show more details. By default, in a production Hive environment, it's normally set to WARN.

```

0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> show tables;
INFO : Compiling command(queryId=hive_20220120014231_f6c14868-167a-4ac1-8c07-b6fbd6308322): show tables
INFO : Semantic Analysis Completed (retrial = false)
INFO : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:tab_name, type:string, comment:from deserializer)], properties:null)
INFO : Completed compiling command(queryId=hive_20220120014231_f6c14868-167a-4ac1-8c07-b6fbd6308322); Time taken: 0.036 seconds
INFO : Executing command(queryId=hive_20220120014231_f6c14868-167a-4ac1-8c07-b6fbd6308322): show tables
INFO : Starting task [Stage-0:DDL] in serial mode
INFO : Completed executing command(queryId=hive_20220120014231_f6c14868-167a-4ac1-8c07-b6fbd6308322); Time taken: 0.012 seconds
INFO : OK

+-----+
|  tab_name  |
+-----+
| accounts   |
| offers     |
| transactions |
| vendors    |
| x_clickstream |
| x_domains  |
+-----+
6 rows selected (0.068 seconds)

```

This is generally setup as part of the server configuration. It can be changed at runtime as can many other options with the Hive set command. This is done within Hive Server: For Example:

```
set hive.server2.logging.operation.level=WARN;
```

More information is available on logging and shell settings here:

<https://cwiki.apache.org/confluence/display/hive/languagemanual+cli>

Table Schema

Understanding the schema of a table you will work with is very important for the selection of columns and rows. Hive can show the structure by using the commands: *describe* *or* *desc*

describe accounts;

or

desc accounts;

col_name	data_type	comment
id	int	
accountno	int	
bankid	int	
first_name	string	
last_name	string	
email	string	
gender	string	
address	string	
city	string	
state	string	
zip	string	

11 rows selected (0.076 seconds)

Here you can see the output is 11 column names, their data types, and no comments are found. (Comments are optional at table creation time).

Take a moment to study the data types in the columns, while they seem straight forward, you'll find many types are not exactly the same as Structured Query Language (SQL) data types. Use the following link to study the types in detail:

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types#LanguageManualTypes-HiveDataTypes>

Q: How many rows of data are in the accounts table.

A: Use the select count(*) command with the table name.

select count(*) from accounts;

This will kick off a MapReduce job, concluding with the total count. Notice immediately Hive returns INFO regarding its operations like the number of jobs, schema in use, task(s) used and status.

```
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> select count(*) from accounts;
INFO : Compiling command(queryId=hive_20220116171906_46913050-a417-4fc5-8d9a-91a139407ec7): select count(*) from accounts
INFO : Semantic Analysis Completed (retrial = false)
INFO : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name: c0, type:bigint, comment:null)], properties:null)
INFO : Completed compiling command(queryId=hive_20220116171906_46913050-a417-4fc5-8d9a-91a139407ec7); Time taken: 2.751 seconds
INFO : Executing command(queryId=hive_20220116171906_46913050-a417-4fc5-8d9a-91a139407ec7): select count(*) from accounts
INFO : Query ID = hive_20220116171906_46913050-a417-4fc5-8d9a-91a139407ec7
INFO : Total jobs = 1
INFO : Launching Job 1 out of 1
INFO : Starting task [Stage-1:MAPRED] in serial mode
INFO : Subscribed to counters: [] for queryId: hive_20220116171906_46913050-a417-4fc5-8d9a-91a139407ec7
INFO : Tez session hasn't been created yet. Opening session
INFO : Dag name: select count(*) from accounts (Stage-1)
INFO : Status: Running (Executing on YARN cluster with App id application_1642291388844_0001)
```

VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	RUNNING	1	0	1	0	0	0
Reducer 2	container	INITED	1	0	0	1	0	0

Q: How many mappers and how many reducers are used here?

A: 1 map and 1 reducer were utilized to perform the operation

Look for the Total field from Hive showing the details. Initially Hive may report using more, once it loads the data the final number is shown.

```
-----  
VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED  
-----  
Map 1 ..... container  SUCCEEDED    1         1         0         0         0         0  
Reducer 2 ..... container  SUCCEEDED    1         1         0         0         0         0  
-----  
VERTICES: 02/02 [=====>>>] 100% ELAPSED TIME: 81.32 s
```

Here we can also see the time it took to process the query.

Q: What is MapReduce?

A: The term "MapReduce" refers to two separate and distinct tasks that Hadoop programs perform.

- **Map - Map jobs filter and organizes the data in sorted order.**
- **Reduce - Reduce jobs apply summary / aggregate operations across the data.**

Q: What is select count(*) doing in Hive?

A: In this case Map set a key/value for each row and the Reducer applied the aggregate count of all rows for the data source.

Additional INFO was also provided as the operation was running including the execution summary, task execution summary and success of the operations. Look for this section in the output of the statement.

```

INFO : Status: DAG finished successfully in 61.18 seconds
INFO :
INFO : Query Execution Summary
INFO : -----
INFO : OPERATION                                DURATION
INFO : -----
INFO : Compile Query                             2.75s
INFO : Prepare Plan                             65.99s
INFO : Get Query Coordinator (AM)                1.24s
INFO : Submit Plan                               7.09s
INFO : Start DAG                                7.79s
INFO : Run DAG                                  61.27s
INFO : -----
INFO :
INFO : Task Execution Summary
INFO : -----
INFO : VERTICES      DURATION(ms)    CPU_TIME(ms)    GC_TIME(ms)    INPUT_RECORDS    OUTPUT_RECORDS
INFO : -----
INFO : Map 1          45397.00        10,080          3,238          1,000            1
INFO : Reducer 2      1415.00          890             100            1                0
INFO : -----
INFO :
INFO : org.apache.tez.common.counters.DAGCounter:
INFO :   NUM_SUCCEEDED_TASKS: 2
INFO :   TOTAL_LAUNCHED_TASKS: 2
INFO :   AM_CPU_MILLISECONDS: 8930
INFO :   AM_GC_TIME_MILLIS: 227

```

Finally, the output of the Reducer task shows the result.

```

INFO : OK
+-----+
|  _c0  |
+-----+
| 1000  |
+-----+
1 row selected (44.561 seconds)

```

There are 1000 rows in the accounts table.

Step 3. Working with MapReduce

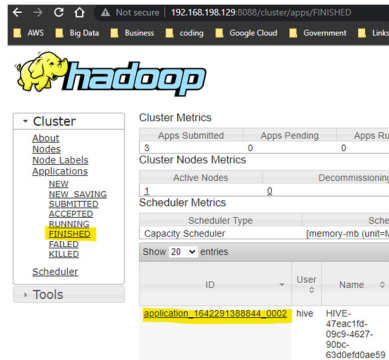
It is quite common in large MapReduce jobs for tasks to take a long time and sometimes jobs must be aborted. Hadoop today features another tool for diagnosing and monitoring jobs and their tasks.

We will use the YARN Resource Manager UI to look at this further. At the top of the web browser click the + symbol to open a new browser tab. *Don't close the current SSH tab, we will come back to it.*

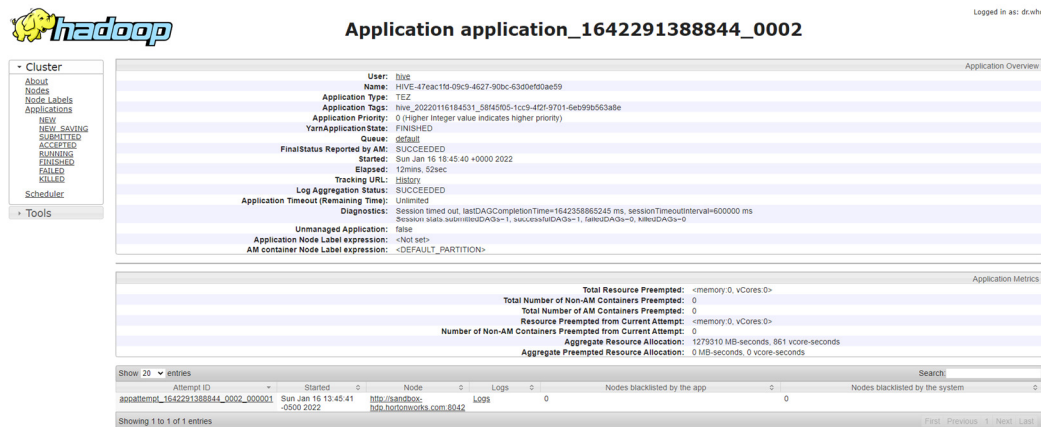
To work with YARN we will launch the Resource Manager Web Interface which is attached to port 8088 on this HDP system.

<http://localhost:8088/cluster>

On the left-hand side menu click on the FINISHED item and then in the table look at the application ID for the latest completed job.



Click the *application_1642291388844_0002* ID of your job to see more details. This opens a new detailed view containing information like the type of application, when it was started, how long it ran, and of course whether it was successful. The output below should be be what is seen.

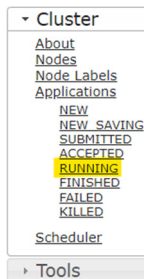


Without closing this browser window, return to the SSH window where you ran the Hive query. If you closed the SSH window, reopen it and login again: refer to Step 1.

Now rerun the previous query. Using the up arrow on the keyboard is the simplest way to bring back a previous command, then press Enter.

select count(*) from accounts;

Switch to the YARN Resource Manager browser window, and this time choose the RUNNING menu item on the left-hand side.



Q: Can you spot your new query? Look for *your* application ID.



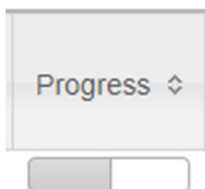
All Applications

Cluster	Cluster Metrics																			
	Apps Submitted		Apps Pending		Apps Running		Apps Completed		Containers Running		Memory Used		Memory Total		Memory Reserved		VCores Used		VCores Total	
	3		0		1		2		1		1.50 GB		4 GB		0 B		1		4	
	Cluster Nodes Metrics																			
	Active Nodes			Decommissioning Nodes			Decommissioned Nodes			Lost Nodes			Unhealthy Nodes			Rebooted Nodes				
	1			0			0			0			0			0				
	Scheduler Metrics																			
	Scheduler Type			Scheduling Resource Type			Minimum Allocation			Maximum Allocation						Maximum Cluster App				
	Capacity Scheduler			[memory-mb (unit=mb), vcores]			<memory:256, vCores:1>			<memory:4096, vCores:4>			0							
	Show 20 ▾ entries																			
Scheduler																				
Tools																				

Look at the details closer, you should see something happening here.

application_1642291388844_0002	hive	HIVE-47eac11d-09c9-4627-90bc-	TEZ	default	0	Sun Jan 16 13:45:40 -0500	N/A	RUNNING	UNDEFINED	3	3	3584	0	0	87.5	87.5	ApplicationMaster
--------------------------------	------	-------------------------------	-----	---------	---	---------------------------	-----	---------	-----------	---	---	------	---	---	------	------	-------------------

Q: Do you see the progress bar updating? Your job is running.



Again, the resource manager is very handy for following jobs. Wait until the job completes and then find it using the FINISHED menu.

Q: How long did it run? Did it succeed?

A: Time will vary on every query as will success.

Q: Where would you find failed jobs? What about aborted jobs?

A: The menu items on the left allow for discovering these and additional information.

Take a moment to familiarize yourself with some of the other menu items. When you are finished return to the SSH browser window. The result again for the number of rows should be 1000.

Run the next additional commands to find out how many rows exist in each of the following other tables:

select count(*) from offers;

Should be 8 rows

select count(*) from vendors;

Should be 10 rows

select count(*) from transactions;

Caution: Don't do a select * from transactions; It's a large table!

Should be 1000010. Yes, one million and 10 rows

To see the actual data in tables we can run a DQL command, try:

select * from vendors;


```

0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> select * from vendors;
INFO : Compiling command(queryId=hive_20220120020953_eebaf721-d846-44f4-bfcc-cbd49bed495c): select * from vendor
INFO : Semantic Analysis Completed (retrial = false)
INFO : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name=vendors.id, type=int, comment:null), FieldSc
ring, comment:null), FieldSchema(name=vendors.state, type:string, comment:null), FieldSchema(name=vendors.categor
ull)], properties:null)
INFO : Completed compiling command(queryId=hive_20220120020953_eebaf721-d846-44f4-bfcc-cbd49bed495c); Time taker
INFO : Executing command(queryId=hive_20220120020953_eebaf721-d846-44f4-bfcc-cbd49bed495c): select * from vendor
INFO : Completed executing command(queryId=hive_20220120020953_eebaf721-d846-44f4-bfcc-cbd49bed495c); Time taker
INFO : OK

```

vendors.id	vendors.name	vendors.city	vendors.state	vendors.category	vendors.swipe_rate
1	Walmart	Bentonville	AR	General	2.0
2	Kroger	Cincinnati	OH	Grocery	2.4
3	Starbucks	Seattle	WA	Coffee	1.5
4	Apple	Cupertino	CA	Technology	2.5
5	Best Buy	Sacramento	CA	Technology	3.0
6	Kohls	Minneapolis	MN	Fashion	2.4
7	Barnes and Noble	New York	NY	Books	3.5
8	Hilton	Washington	DC	Travel	3.2
9	American Airlines	Dallas	TX	Travel	2.7
10	REI	Seattle	WA	Sporting Goods	3.0

```

10 rows selected (0.445 seconds)

```

This command returned all the rows and columns from the table.

Step 4. HiveQL Projection

We could also use Projection, to select specific columns in the result. Just like SQL, HiveQL allows you to return any combination of columns by adding those to the select statement. Return the name and city columns from the vendor's table.

Remember to separate table column names with commas. Like this:

```
select table_column_name, table_column_name2 from table_name;
```

```
select name, city from vendors;
```

```

0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> select name, city from vendors;
INFO : Compiling command(queryId=hive_20220120021546_3016deeb-d4f8-4d6c-9a26-b49522389cc8):
INFO : Semantic Analysis Completed (retrial = false)
INFO : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name=name, type:string, comm
INFO : Completed compiling command(queryId=hive_20220120021546_3016deeb-d4f8-4d6c-9a26-b495
INFO : Executing command(queryId=hive_20220120021546_3016deeb-d4f8-4d6c-9a26-b49522389cc8):
INFO : Completed executing command(queryId=hive_20220120021546_3016deeb-d4f8-4d6c-9a26-b495
INFO : OK

```

name	city
Walmart	Bentonville
Kroger	Cincinnati
Starbucks	Seattle
Apple	Cupertino
Best Buy	Sacramento
Kohls	Minneapolis
Barnes and Noble	New York
Hilton	Washington
American Airlines	Dallas
REI	Seattle

```

10 rows selected (0.311 seconds)

```

Here the ‘resultset’ or ‘recordset’ now only has the rows for the columns name and city as we expected with our SQL projection script.

Try something instead to return a subset of the data, like limit clause. The limit clause can be used to constrain the number of rows returned by the select statement. Use the transactions table to return some of the rows. Try 10 for now.

select * from transactions limit 10;

Use limit, otherwise you’ll wait a long time for all 1000010 rows!

```
0: jdbc:hive2://sandbox-hdp.hortonworks.com:2> select * from transactions limit 10;
INFO : Compiling command(queryId=hive_20220116173948_d97ee6b8-552b-4473-92f4-781768a14dc0): select * from transactions limit 10
INFO : Semantic Analysis Completed (retry = false)
INFO : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:transactions.id, type:int, comment:null), FieldSchema(name:transactions.account_id, type:int, comment:null), FieldSchema(name:transactions.vendor_id, type:int, comment:null), FieldSchema(name:transactions.times, type:string, comment:null), FieldSchema(name:transactions.city, type:string, comment:null), FieldSchema(name:transactions.state, type:string, comment:null), FieldSchema(name:transactions.amount, type:double, comment:null)], properties:null)
INFO : Completed compiling command(queryId=hive_20220116173948_d97ee6b8-552b-4473-92f4-781768a14dc0); Time taken: 0.42 seconds
INFO : Executing command(queryId=hive_20220116173948_d97ee6b8-552b-4473-92f4-781768a14dc0): select * from transactions limit 10
INFO : Completed executing command(queryId=hive_20220116173948_d97ee6b8-552b-4473-92f4-781768a14dc0); Time taken: 0.004 seconds
INFO : OK
```

transactions.id	transactions.account_id	transactions.vendor_id	transactions.times	transactions.city	transactions.state	transactions.amount
1	5715	5	2015-01-01 00:00:00	El Paso	TX	62.81
2	8236	7	2015-01-01 00:00:00	Pittsburgh	PA	157.33
3	2500	10	2015-01-01 00:00:01	Austin	TX	14.06
4	6832	4	2015-01-01 00:00:02	Savannah	GA	203.46
5	6859	7	2015-01-01 00:00:03	Indianapolis	IN	231.98
6	9694	7	2015-01-01 00:00:04	Gainesville	GA	2.13
7	9411	4	2015-01-01 00:00:05	Harrisburg	PA	270.31
8	5490	5	2015-01-01 00:00:06	Boise	ID	176.84
9	1565	3	2015-01-01 00:00:06	Bronx	NY	12.5
10	8180	7	2015-01-01 00:00:07	El Paso	TX	257.15

10 rows selected (0.713 seconds)

The output should look like above and be relatively quick compared to count(*) we used earlier...

Q: Why do you think that is?

A: The query fetches only a limited number of rows from the table, without using a MapReduce job.

We will now exit the Hive CLI. Use one of the following two commands to exit the Hive CLI at the command prompt.

!q or !quit

Step 5. Working with Hadoop HDFS

Hive integrates with Hadoop's HDFS and allows for the usage of MapReduce to speed up queries on large datasets by running these queries in parallel against a dataset when properly configured. It takes advantage of advanced clustering features like partitioning and sharding to deal with these very large datasets overall. Programmatically we can use languages like Java, C#, and Python to perform these queries directly but that requires additional libraries and often significant experience coding.

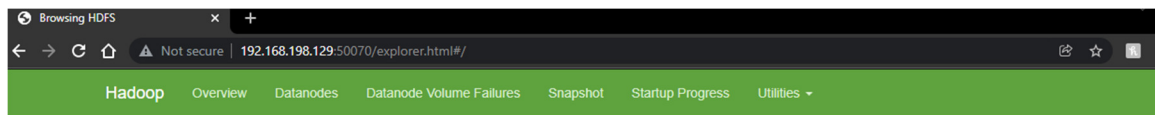
In this course Hive's Metastore will manage the data structures and we will take advantage of our SQL knowledge allowing Hive and Hadoop to translate our HiveQL language code into runnable MapReduce jobs. The expectation in this course is you know how to work with Hadoop and SQL already. If you do not, please find materials to familiarize yourself before going further in the labs.

First, we need to understand the how the data structures in both Hadoop and Hive can help us store and retrieve the data.

Hadoop has a GUI we can utilize to explore the data files loaded for the system. Open the following URL.

<http://localhost:50070/explorer.html#/>




You will see the Hadoop directory folder and file structure for the lab files that will be used in this class as shown below.



Browse Directory

/

Go!



















Show

25

entries

Search:

<input type="checkbox"/>	 Permission	 Owner	 Group	 Size	 Last Modified	 Replication	 Block Size	 Name	
<input type="checkbox"/>	drwxrwxrwt	yarn	hadoop	0 B	Nov 29 2018	0	0 B	app-logs	
<input type="checkbox"/>	drwxr-xr-x	hdfs	hdfs	0 B	Nov 29 2018	0	0 B	apps	
<input type="checkbox"/>	drwxr-xr-x	yarn	hadoop	0 B	Nov 29 2018	0	0 B	ats	
<input type="checkbox"/>	drwxr-xr-x	hdfs	hdfs	0 B	Nov 29 2018	0	0 B	atsv2	
<input type="checkbox"/>	drwxr-xr-x	root	hdfs	0 B	Jan 15 23:31	0	0 B	data	
<input type="checkbox"/>	drwxr-xr-x	root	hdfs	0 B	Jan 15 23:51	0	0 B	data2	
<input type="checkbox"/>	drwxr-xr-x	hdfs	hdfs	0 B	Nov 29 2018	0	0 B	hdp	

On the right-hand side of the browser window, you will see a directory named “data” shown circled here. This is where some of the files for this class can be found.

Click the [data](#) link. This will display the ten folders with their corresponding data files underneath. See /data is shown as path.

Browse Directory

/data

Go!

Show

25

entries

Search:

<div><input type="checkbox"/></div>	<div>Permission</div>	<div>Owner</div>	<div>Group</div>	<div>Size</div>	<div>Last Modified</div>	<div>Replication</div>	<div>Block Size</div>	<div>Name</div>	<div></div>
<div><input type="checkbox"/></div>	<div>drwxr-xr-x</div>	<div>root</div>	<div>hdfs</div>	<div>0 B</div>	<div>Jan 15 23:25</div>	<div>0</div>	<div>0 B</div>	<div>accounts</div>	<div></div>
<div><input type="checkbox"/></div>	<div>drwxr-xr-x</div>	<div>root</div>	<div>hdfs</div>	<div>0 B</div>	<div>Jan 15 23:28</div>	<div>0</div>	<div>0 B</div>	<div>clickstream</div>	<div></div>
<div><input type="checkbox"/></div>	<div>drwxr-xr-x</div>	<div>root</div>	<div>hdfs</div>	<div>0 B</div>	<div>Jan 15 23:31</div>	<div>0</div>	<div>0 B</div>	<div>domains</div>	<div></div>
<div><input type="checkbox"/></div>	<div>drwxr-xr-x</div>	<div>root</div>	<div>hdfs</div>	<div>0 B</div>	<div>Jan 15 23:25</div>	<div>0</div>	<div>0 B</div>	<div>offers</div>	<div></div>
<div><input type="checkbox"/></div>	<div>drwxr-xr-x</div>	<div>root</div>	<div>hdfs</div>	<div>0 B</div>	<div>Jan 15 23:25</div>	<div>0</div>	<div>0 B</div>	<div>text_fomc</div>	<div></div>
<div><input type="checkbox"/></div>	<div>drwxr-xr-x</div>	<div>root</div>	<div>hdfs</div>	<div>0 B</div>	<div>Jan 15 23:25</div>	<div>0</div>	<div>0 B</div>	<div>text_moby</div>	<div></div>
<div><input type="checkbox"/></div>	<div>drwxr-xr-x</div>	<div>root</div>	<div>hdfs</div>	<div>0 B</div>	<div>Jan 15 23:25</div>	<div>0</div>	<div>0 B</div>	<div>text_sotu</div>	<div></div>
<div><input type="checkbox"/></div>	<div>drwxr-xr-x</div>	<div>root</div>	<div>hdfs</div>	<div>0 B</div>	<div>Jan 15 23:25</div>	<div>0</div>	<div>0 B</div>	<div>transactions</div>	<div></div>
<div><input type="checkbox"/></div>	<div>drwxr-xr-x</div>	<div>root</div>	<div>hdfs</div>	<div>0 B</div>	<div>Jan 15 23:28</div>	<div>0</div>	<div>0 B</div>	<div>twinkle</div>	<div></div>
<div><input type="checkbox"/></div>	<div>drwxr-xr-x</div>	<div>root</div>	<div>hdfs</div>	<div>0 B</div>	<div>Jan 15 23:25</div>	<div>0</div>	<div>0 B</div>	<div>vendors</div>	<div></div>

Showing 1 to 10 of 10 entries

Previous

1

Next

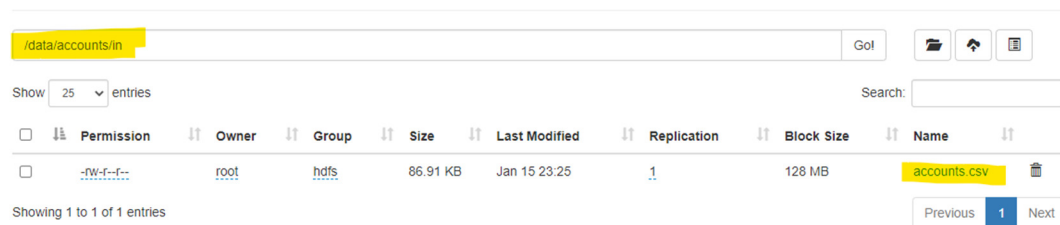
Click on a few of the folders above and review the files.

Q: What file extensions do you see?

A: There should be a mixture of csv, json, txt and data files.

In the labs we will specify these data files through the usage of data table structures within Hive like we have seen, but the actual data in Hadoop is textual based flat files generally. An example is shown below of the `/data/accounts/in` folder and it's `accounts.csv` file we counted.

Browse Directory



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	hdfs	86.91 KB	Jan 15 23:25	1	128 MB	accounts.csv

Return to the SSH browser window at <http://localhost:4200>

You may need to login as *wasadmin* again.

If you are still in the command prompt for hive use **!q** or **!quit** to exit. At the command prompt, type the following to see the root folder:

hdfs dfs -ls /

The output should show 14 folders with HDFS like below.

```
[wasadmin@sandbox-hdp LabFiles]$ hdfs dfs -ls /
Found 14 items
drwxrwxrwt - yarn hadoop 8236 0 2018-11-29 17:56 /app-logs
drwxr-xr-x - hdfs hdfs 2500 0 2018-11-29 19:01 /apps
drwxr-xr-x - yarn hadoop 6032 0 2018-11-29 17:25 /ats
drwxr-xr-x - hdfs hdfs 6859 0 2018-11-29 17:26 /atsv2
drwxrwxrwx - wasadmin hdfs 9694 0 2022-01-20 00:46 /data
drwxr-xr-x - hdfs hdfs 9411 0 2018-11-29 17:26 /hdp
drwx----- - livy hdfs 5490 0 2018-11-29 17:55 /livy2-recovery
drwxr-xr-x - mapred hdfs 1565 0 2018-11-29 17:26 /mapred
drwxrwxrwx - mapred hadoop 8180 0 2018-11-29 17:26 /mr-history
drwxr-xr-x - hdfs hdfs 0 0 2018-11-29 18:54 /ranger
drwxrwxrwx - spark hadoop 0 0 2022-01-20 02:27 /spark2-history
drwxrwxrwx - hdfs hdfs 0 0 2018-11-29 19:01 /tmp
drwxr-xr-x - hdfs hdfs 0 0 2022-01-20 00:38 /user
drwxr-xr-x - hdfs hdfs 0 0 2018-11-29 17:51 /warehouse
```

Now let's add the data folder to the previous command.

hdfs dfs -ls /data

Did you see the ten folders we saw earlier in the Hadoop File Browser?
If not check the command syntax used above.

```
[wasadmin@sandbox-hdp LabFiles]$ hdfs dfs -ls /data
Found 10 items
drwxrwxrwx - wasadmin hdfs      0 2022-01-20 00:38 /data/accounts
drwxrwxrwx - wasadmin hdfs      0 2022-01-20 00:42 /data/clickstream
drwxrwxrwx - wasadmin hdfs      0 2022-01-20 00:47 /data/domains
drwxrwxrwx - wasadmin hdfs      0 2022-01-20 00:38 /data/offers
drwxrwxrwx - wasadmin hdfs      0 2022-01-20 00:39 /data/text_fomc
drwxrwxrwx - wasadmin hdfs      0 2022-01-20 00:39 /data/text_moby
drwxrwxrwx - wasadmin hdfs      0 2022-01-20 00:39 /data/text_sotu
drwxrwxrwx - wasadmin hdfs      0 2022-01-20 00:38 /data/transactions
drwxrwxrwx - wasadmin hdfs      0 2022-01-20 00:42 /data/twinkle
drwxrwxrwx - wasadmin hdfs      0 2022-01-20 00:38 /data/vendors
```

Let's look within a specific folder from our *user/wasadmin* account.
Notice the removal of / before the name of the data folder here.

hdfs dfs -ls wasadmin/billing/in

This will return output from the hadoop folder /billing/in as follows:

```
[wasadmin@sandbox-hdp LabFiles]$ hdfs dfs -ls wasadmin/billing/in
Found 10 items
-rw-r--r-- 1 wasadmin hdfs      2816114 2022-01-20 01:14 wasadmin/billing/in/billing-2012-01-01.log
-rw-r--r-- 1 wasadmin hdfs      2816178 2022-01-20 01:14 wasadmin/billing/in/billing-2012-01-02.log
-rw-r--r-- 1 wasadmin hdfs      2816068 2022-01-20 01:14 wasadmin/billing/in/billing-2012-01-03.log
-rw-r--r-- 1 wasadmin hdfs      2816721 2022-01-20 01:14 wasadmin/billing/in/billing-2012-01-04.log
-rw-r--r-- 1 wasadmin hdfs      2815984 2022-01-20 01:14 wasadmin/billing/in/billing-2012-01-05.log
-rw-r--r-- 1 wasadmin hdfs      2816024 2022-01-20 01:14 wasadmin/billing/in/billing-2012-01-06.log
-rw-r--r-- 1 wasadmin hdfs      2815959 2022-01-20 01:14 wasadmin/billing/in/billing-2012-01-07.log
-rw-r--r-- 1 wasadmin hdfs      2815733 2022-01-20 01:14 wasadmin/billing/in/billing-2012-01-08.log
-rw-r--r-- 1 wasadmin hdfs      2815905 2022-01-20 01:14 wasadmin/billing/in/billing-2012-01-09.log
-rw-r--r-- 1 wasadmin hdfs      2816103 2022-01-20 01:14 wasadmin/billing/in/billing-2012-01-10.log
```

This is a specific folder owned by the user *wasadmin* but found under a different path */user/wasadmin/wasadmin/billing/in*

Let's preview one of the files and look at the data using Hadoop commands. Type:

hadoop fs -cat wasadmin/billing/in/billing-2012-01-01.log | tail -5

The last five lines from the billing-2012-01-01.log file are:

```
[wasadmin@sandbox-hdp LabFiles]$ hadoop fs -cat wasadmin/billing/in/billing-2012-01-01.log | tail -5
1325462395680,85648,5,31,82
1325462396544,44139,4,74,54
1325462397408,99017,7,22,168
1325462398272,51897,8,70,46
1325462399136,89086,5,74,86
```

This corresponds to the file header for the billing-2012-01-01.log which would be timestamp, customer_id, resource_id, quantity, and cost.

Q: Can you return the billing information above in HiveQL?

A: There are a number of ways to accomplish this generally when there is an existing table and database schema for the billing information stored in the Hive Metastore.

First check to see if there is a billing database

hive -e "show databases like 'b*' ";

Q: Were you able to find a database starting like 'b*' or 'B*'?

A: No there is no database in this form.

```
INFO : OK
+-----+
| database_name |
+-----+
+-----+
```

Q: What else could be checked? Perhaps change the command use show tables? Maybe this: hive -e "show tables like 'b*' ";

A: That will result in an empty tab_name as shown.

```
INFO : OK
+-----+
| tab_name |
+-----+
+-----+
```


Maybe we need to look in a specific Database like wasadmin_db we used earlier. What command could be used for that? Try:

hive -e 'show tables in wasadmin_db'

```
INFO : OK
+-----+
| tab_name |
+-----+
| accounts |
| offers   |
| transactions |
| vendors  |
| x_clickstream |
| x_domains |
+-----+
```

Currently there is no table in the existing databases named 'billings' table.

Step 6. Hive Metastore and Hadoop HDFS

Let's create a new table in hive under the wasadmin_db database for one of the log files we have explored, billings. First start the Hive CLI so we can run the SQL DDL command to create a table. In this case using the external data source from Hadoop HDFS.

Hive

Now set the database to store the table within Hive.

use wasadmin_db;

Next use the create statement to form the billings table schema.

**CREATE EXTERNAL TABLE IF NOT EXISTS billings (datetime
STRING, customer_id INT, resource_id INT, qty INT, cost
INT) ROW FORMAT DELIMITED FIELDS TERMINATED
BY ',' ;**

This will create the new schema for the billings data and allow us to select data from the table called billings. We used EXTERNAL TABLE as the files existed already in our own folder wasadmin.

Describe the details.

desc billings;

```
INFO : OK
+-----+-----+-----+
| col_name | data_type | comment |
+-----+-----+-----+
| datetime | string    |         |
| customer_id | int      |         |
| resource_id | int      |         |
| qty       | int      |         |
| cost      | int      |         |
+-----+-----+-----+
5 rows selected (0.131 seconds)
```

The five columns have been created with the proper data types. Now select some rows to view the data. Make sure you *use limit*.

select * from billings limit 10;

```
INFO : OK
+-----+-----+-----+-----+-----+
| billings.datetime | billings.customer_id | billings.resource_id | billings.qty | billings.cost |
+-----+-----+-----+-----+-----+
| 1325376000000    | 54499                | 3                    | 49           | 29            |
| 1325376000864    | 77358                | 1                    | 91           | 67            |
| 1325376001728    | 60728                | 10                   | 21           | 174           |
| 1325376002592    | 1572                 | 3                    | 97           | 62            |
| 1325376003456    | 5476                 | 6                    | 90           | 140           |
| 1325376004320    | 65299                | 10                   | 35           | 77            |
| 1325376005184    | 42120                | 2                    | 46           | 140           |
| 1325376006048    | 77556                | 7                    | 8            | 35            |
| 1325376006912    | 71208                | 8                    | 42           | 164           |
| 1325376007776    | 557                  | 6                    | 94           | 62            |
+-----+-----+-----+-----+-----+
```

Q: Why use limit?

A: We have loaded all the log files, there are 100000 rows in billings table. It would take a long time to retrieve all 100000 rows.

select count(*) from billings;

```
INFO : OK
+-----+
|      _c0      |
+-----+
|      100000    |
+-----+
```

We will finish this Lab by investigating further the tables we have seen and the two other tables in the wasadmin_db.

Use commands we have learned to detail all the tables including the two other tables **x_domains** and **x_clickstream** which we have not seen yet.

Specifically find these last tables schemas, table sizes and review some of the data present in the tables. Remember having a good understanding of the data you are querying is important to writing successful queries.

Step 7. How to set variables in HIVE scripts

Hive variables are key-value pairs that can be set using the set command and they can be used in scripts and Hive SQL. The values of the variables in Hive scripts are substituted during the query construct.

When working with Hive QL and scripts we often required to use specific values for each environment, unfortunately the values change for each environment. So, when you set values to variables, they are local to the active Hive session and these values are not visible to other sessions. You can set these variables on Hive CLI (older version), Beeline, and Hive scripts.

Hive stores variables in four different namespaces, namespace is a way to separate variables.

- hiveconf
- hivevar
- system

- env

hiveconf

hiveconf is the default namespace, if you don't provide a namespace at the time of setting a variable, it will store your variable in hiveconf namespace by default. hiveconf namespace also contains several Hive default configuration variables.

In order to retrieve the variable from hiveconf, you have to explicitly specify the namespace. Not specifying namespace returns an error.

Notice that hiveconf can be used for variable substitution.

```
hiveconf deptno=20
```

```
select * from emp_tab where col6=${hiveconf:deptno};  
select col1,${name} from emp_tab where col6 =${hiveconf:deptno};
```

```
hive --hiveconf deptno=20 -e 'select * from emp_tab where col6  
=${hiveconf:deptno};'
```

hivevar

Hive version 0.8.0 introduced a new namespace hivevar to set the custom variables (JIRA HIVE-2020), this separates custom variables from Hive default config variables. Hive recommends using hivevar explicitly for custom variables.

To retrieve values from hivevar namespace, you can either specify hivevar namespace or ignore it as hivevar is a default namespace for retrieval.

```
hive --hivevar deptno=10 -e 'select * from emp_tab where col6
=${deptno};'
```

```
hive --hivevar deptno=10 --hiveconf tablename=emp_tab -e 'select *
from ${hiveconf:tablename} where col6 =${deptno};'
```

```
hive --hivevar empid=col1 --hiveconf tablename=emp_tab --
hivevar deptno=10 -f /home/wasadmin/script.hql
```

Hive System Variables (system)

Hive default provides certain system variables and all system variables can be accessed in Hive using system namespace. For example.

system:java.version

system:user.timezone

system:user.home

Hive Environment Variables (env)

Hive also default provides certain environment variables and all environment variables can be accessed in Hive using env namespace.

For example

env:PWD

env:USER

To see all the available variables, from the command line, run `hive -e 'set;'` Inside the Hive CLI just use `'set;'` command at the prompt

set;

There are over 1000 items that may be configured in Hive.

Let's try a few commands for the set variables and see what is returned.

set system:java.version;

You should see the current version of Java installed on the system.

set hive.metastore.warehouse.dir;

You should see the hive metastore directory listing for Hadoop
hive.metastore.warehouse.dir=/user/hive/warehouse

Which execution engine is currently being used in hive?

set hive.execution.engine;

You should see Tez as the engine in use. There are three choice available for Hortonworks: mr, tez and spark engines. mr has been deprecated as of Hortonworks version 3 and spark is considered experimental still, so we have the tez engine.

Use !q to exit the Hive CLI.

!q

Set Variables in Hive Scripts

Hive scripts supports using all variables explained above, you can use any of these along with their namespace. Let's create test.hql file with the below content. Here we will refer to the table name from hivevar namespace. You can use vi to create the file. Type:

vi test.hql

Add this text by pressing the 'i' key on the keyboard for *INSERT* mode.

i

select * from \${hivevar:table} limit 10;

Press the ESC (escape key) to exit INSERT mode and then use the Shift Key to enter :wq to save the file.

:wq

Execute the test.hql script by running the below command to see the 10 rows returned from the billings table.

hive --hivevar table='wasadmin_db.billings' -f test.hql

Variable substitution resolves the database and table names for query

Review

We have run some Hive and Hadoop commands in this lab to better understand these tools and command usage.

One of the keys to being successful when working with data is to understand the data schemas, data types and commands available to use for retrieving the resultsets. We will explore these further in the next lab.