

Northeastern University Assignment 02

ALY6110 Data Management and Big Data

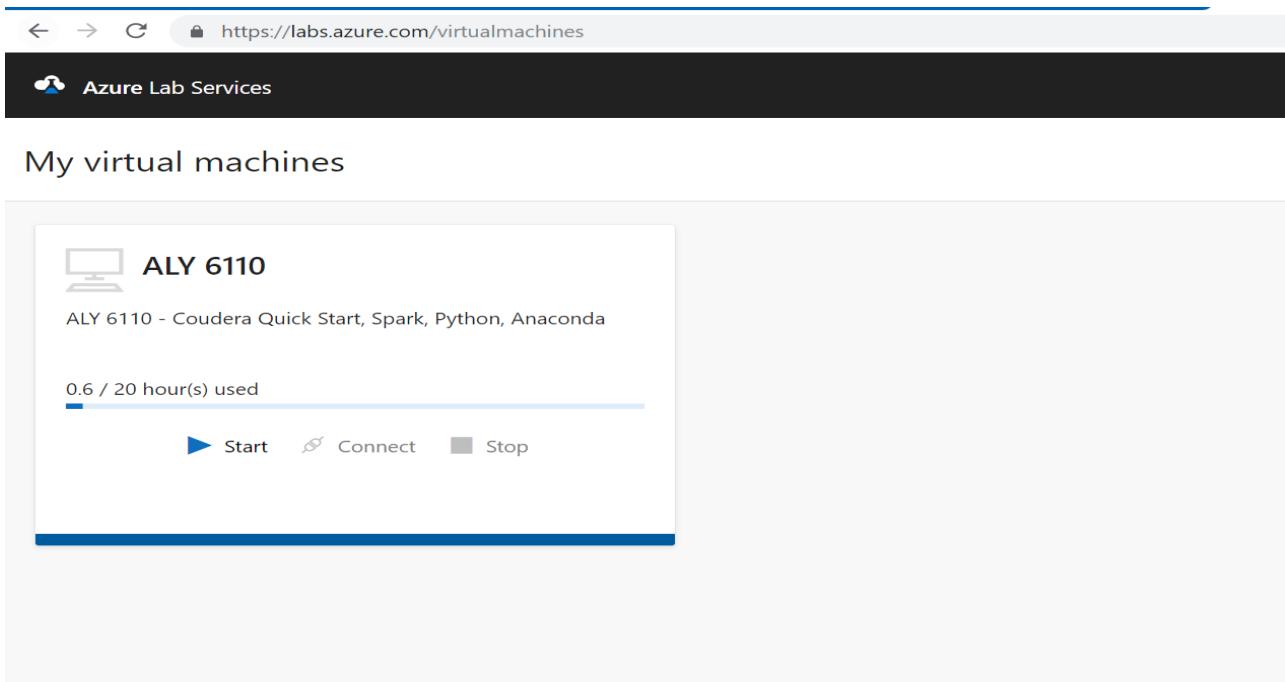
<https://northeastern.blackboard.com>

Problem 1:

Start up the Microsoft Azure Lab – Virtual Machine and get all the Hadoop Services running.

Note: I am using the Microsoft Azure Lab for Class ALY 6110 for running Cloudera and Hadoop Services which already has the Hyper-V Manager and the Cloudera installed in it.

1. As I am using the Microsoft Azure Lab Machine ALY 6110 for running Cloudera and Hadoop Services, we first visit the URL
<https://labs.azure.com/virtualmachines>.

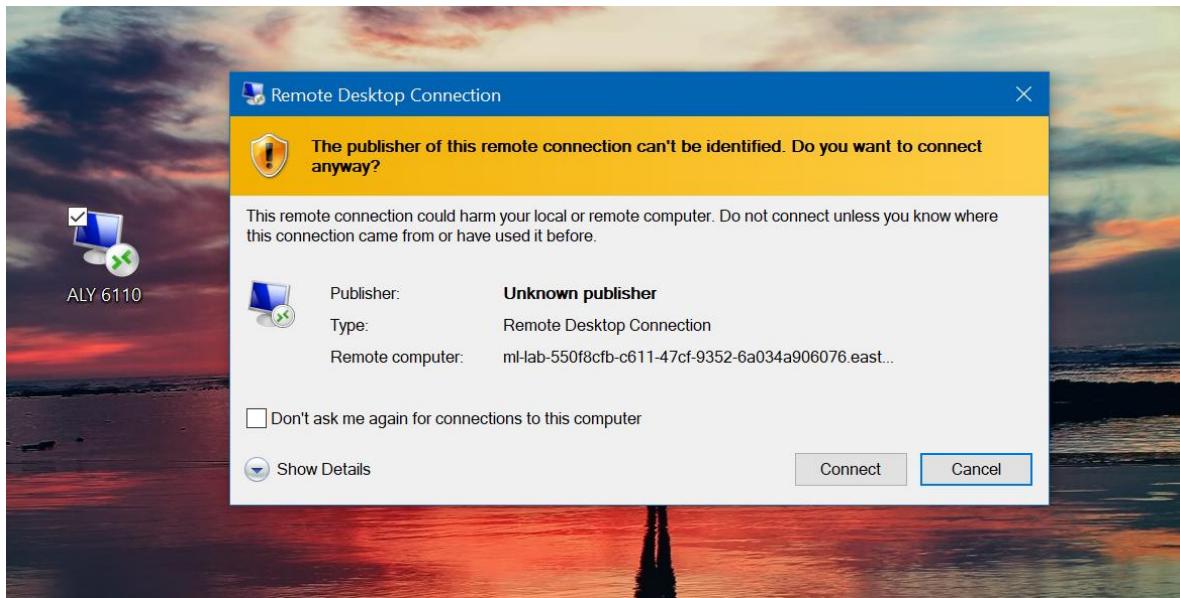


2. Now click on "Start" to start the virtual machine. After we click on Start, a remote desktop client will get installed. I have downloaded it on the Desktop. Now double click on the Remote Desktop Client named as "ALY 6110". You will see the window as shown below. Click on "Connect".

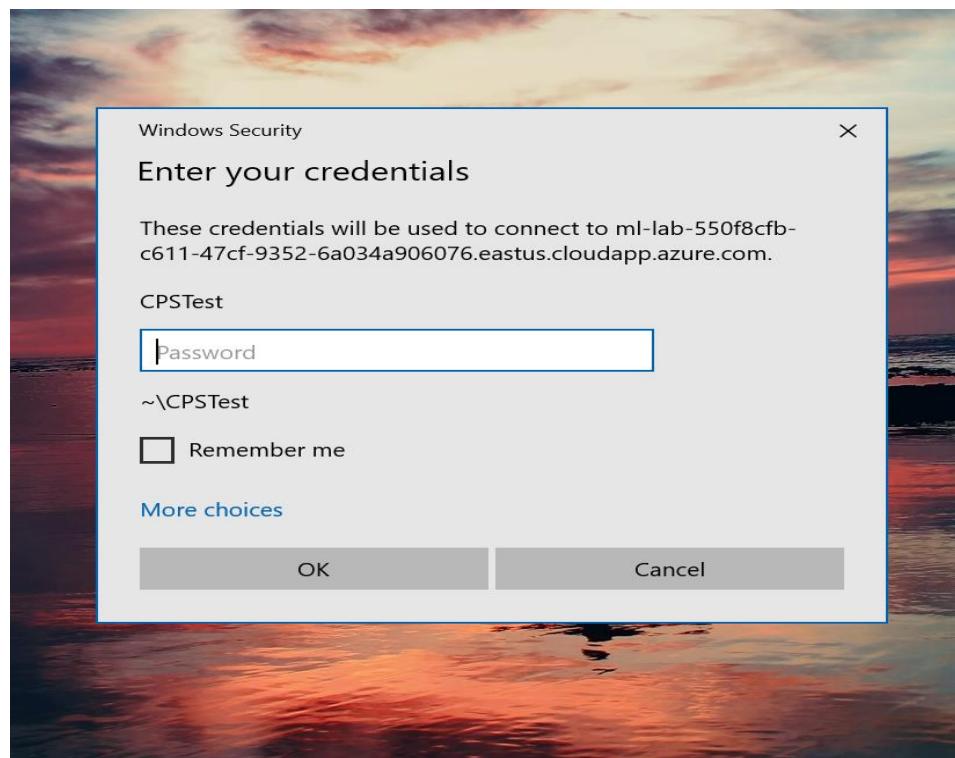
Northeastern University Assignment 02

ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>



3. It will then ask for the password for starting the virtual machine. Enter the password and click "OK".

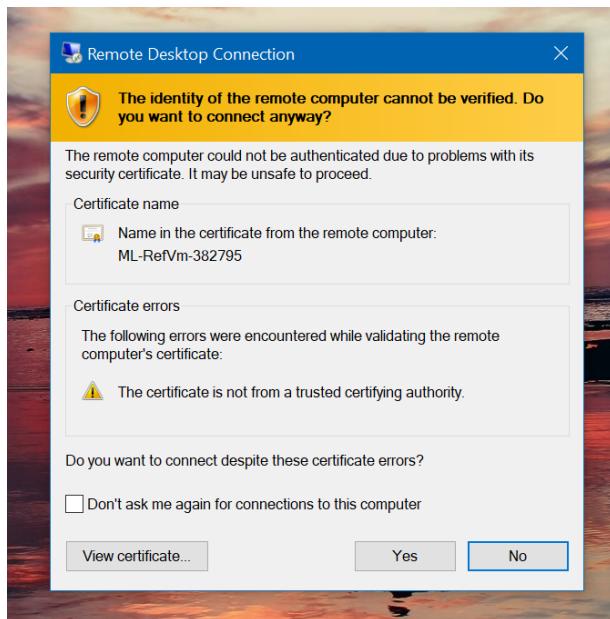


Northeastern University Assignment 02

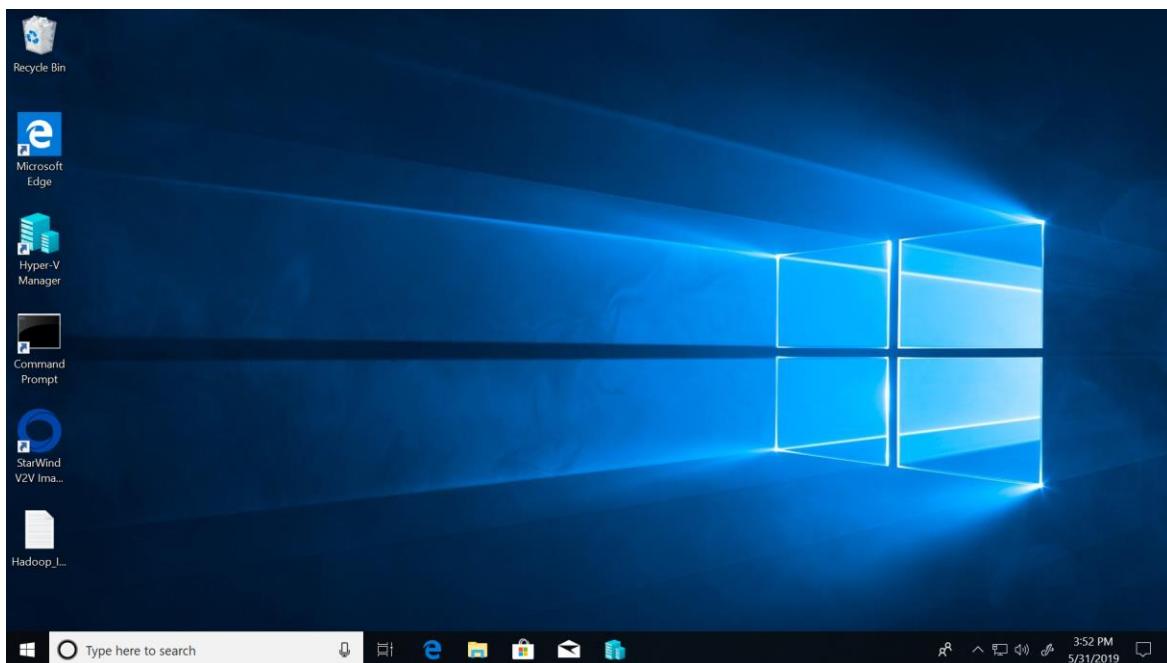
ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

4. It will then give a prompt as shown below because of the certificate errors. But we will be connecting despite these certificate errors and click on “Yes”.



5. Our Virtual Machine will get started and looks like the one below in the screenshot.

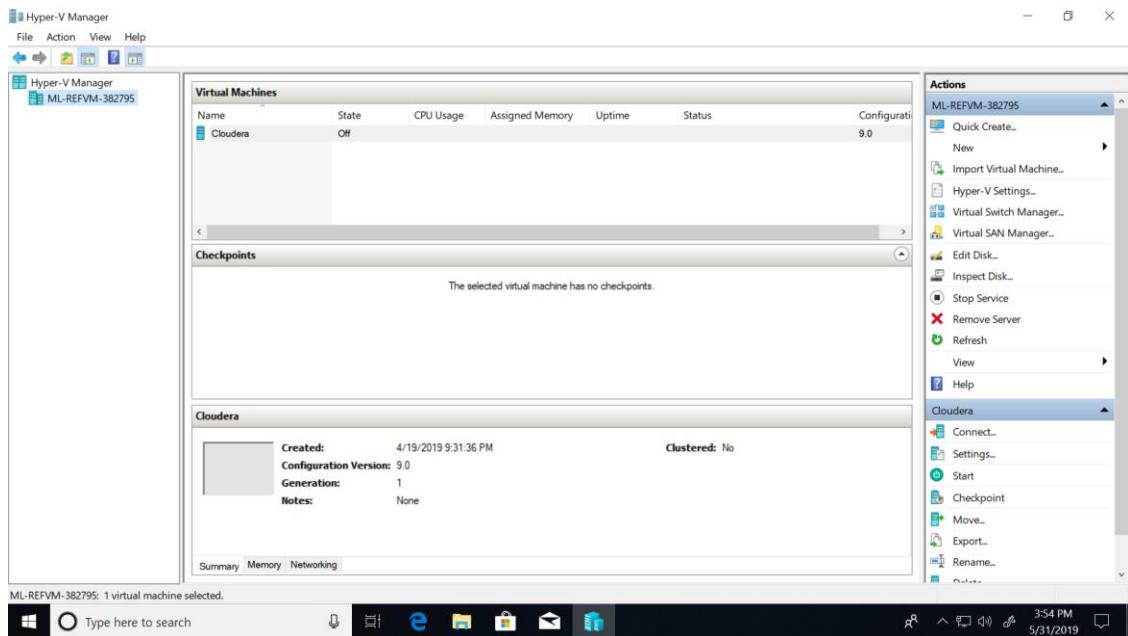


Northeastern University Assignment 02

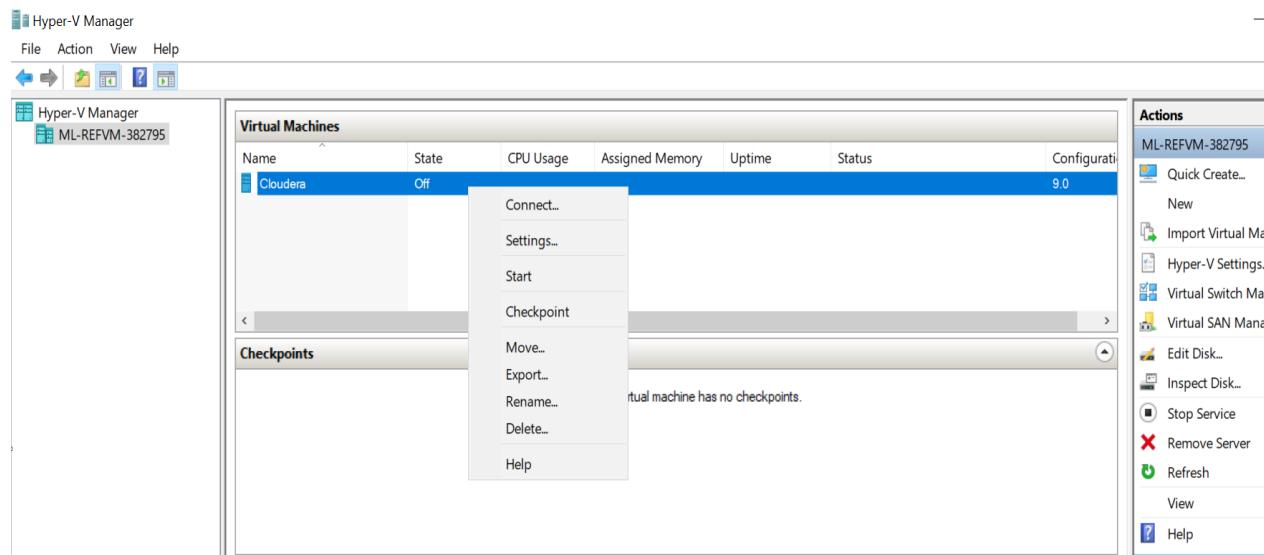
ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

6. Open Hyper-V Manager present on the desktop. Refer the below screenshot. We can see that under Virtual Machines, status of Cloudera is “Off”.



7. Right click on Cloudera and select “Connect” as shown below.

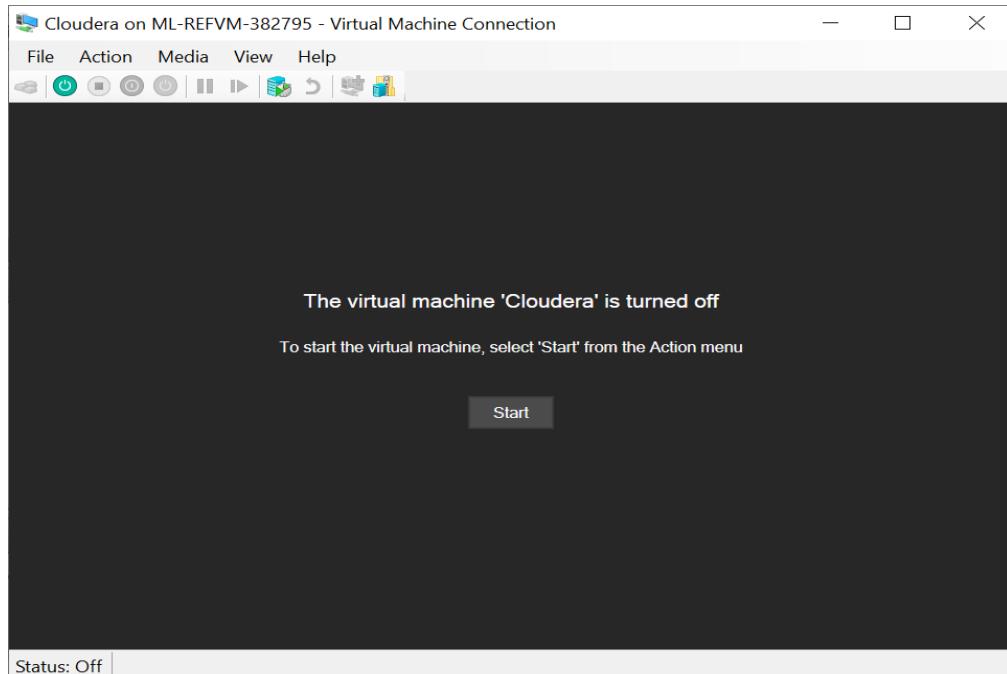


Northeastern University Assignment 02

ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

8. We can see that a new window with Cloudera VM comes up but it can be seen at the bottom left that the status is still Off.



9. Click on "Start". The Cloudera VM will now get started as shown below.

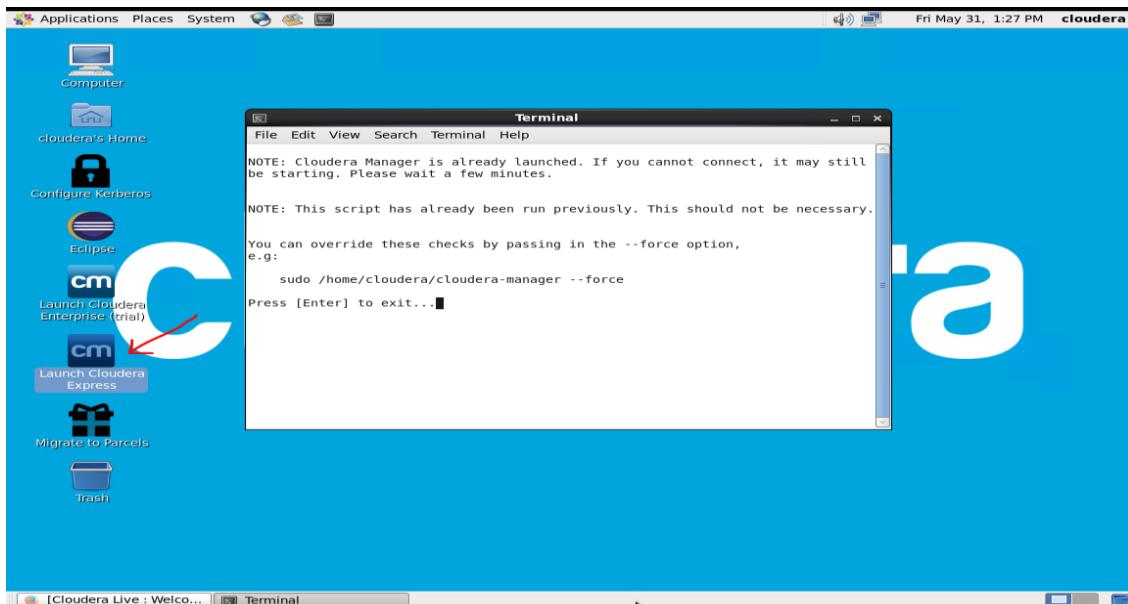


Northeastern University Assignment 02

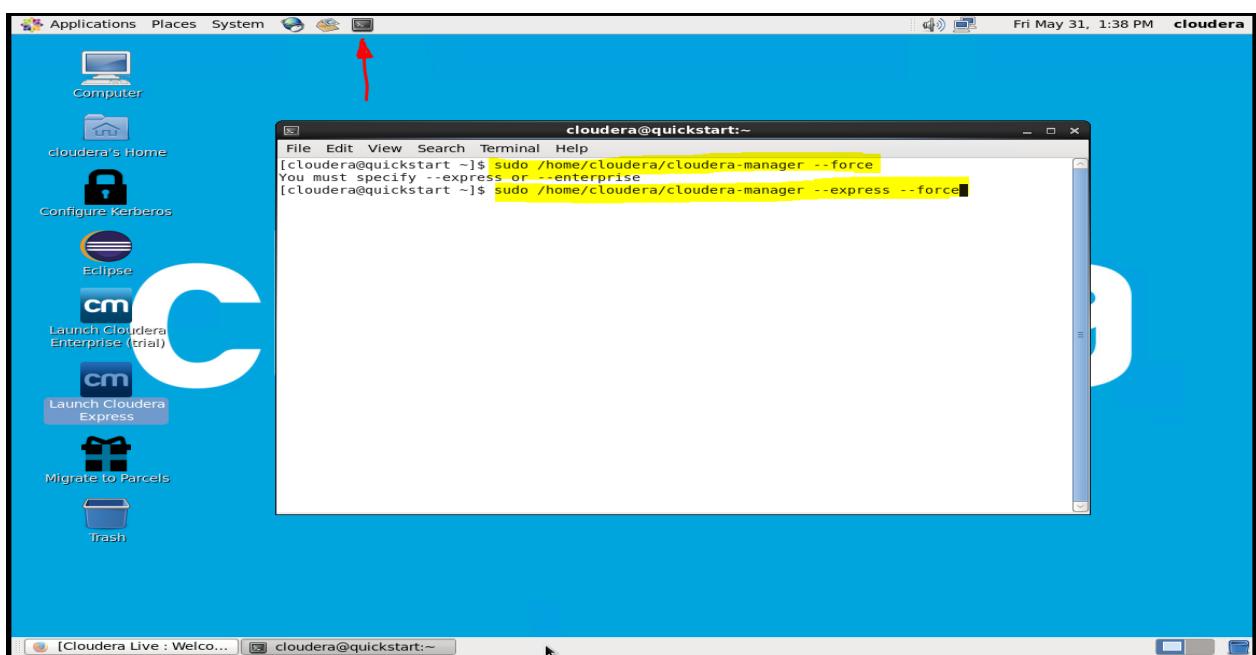
ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

10. After the Cloudera gets started, open the “Launch Cloudera Express” as shown in the below screenshot. A new terminal window gets started.



11. Copy the path “***sudo /home/cloudera/cloudera-manager --force***” from the above terminal and paste it in the new terminal. Now we must specify “***--express***” and hit enter. Refer the below screenshot.

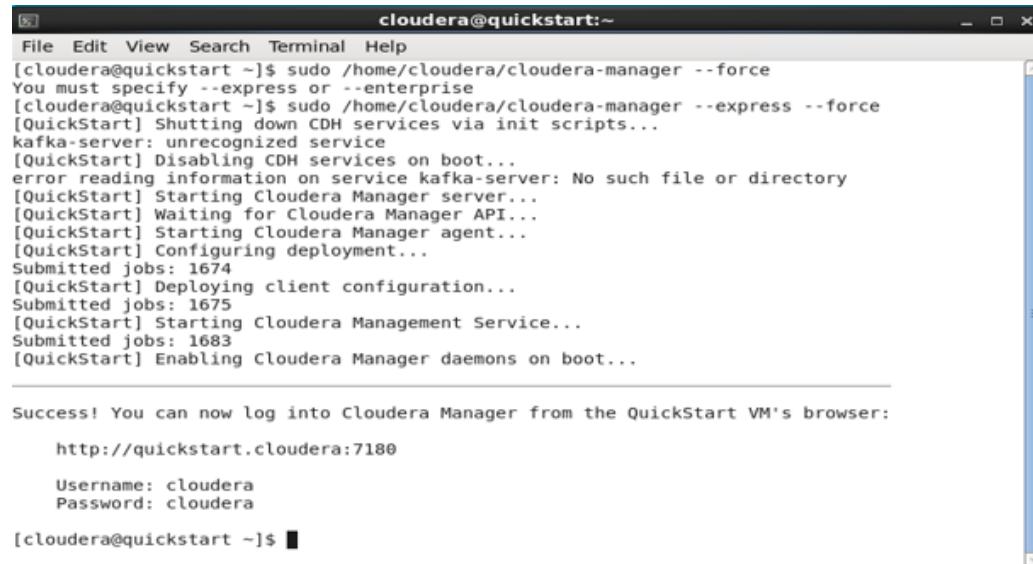


Northeastern University Assignment 02

ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

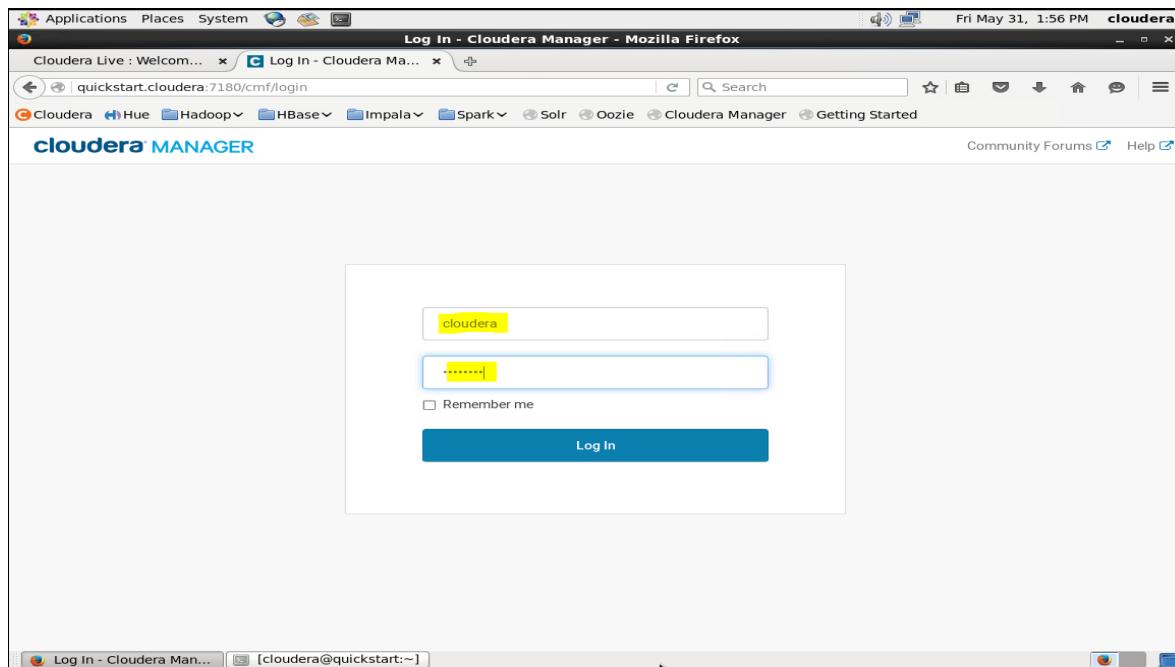
12. The terminal will now provide us with the username and password for the Cloudera Manager as shown below.



```
cloudera@quickstart:~$ sudo /home/cloudera/cloudera-manager --force
You must specify --express or --enterprise
[cloudera@quickstart ~]$ sudo /home/cloudera/cloudera-manager --express --force
[QuickStart] Shutting down CDH services via init scripts...
kafka-server: unrecognized service
[QuickStart] Disabling CDH services on boot...
error reading information on service kafka-server: No such file or directory
[QuickStart] Starting Cloudera Manager server...
[QuickStart] Waiting for Cloudera Manager API...
[QuickStart] Starting Cloudera Manager agent...
[QuickStart] Configuring deployment...
Submitted jobs: 1674
[QuickStart] Deploying client configuration...
Submitted jobs: 1675
[QuickStart] Starting Cloudera Management Service...
Submitted jobs: 1683
[QuickStart] Enabling Cloudera Manager daemons on boot...

Success! You can now log into Cloudera Manager from the QuickStart VM's browser:
http://quickstart.cloudera:7180
Username: cloudera
Password: cloudera
[cloudera@quickstart ~]$
```

13. Open the browser in the Cloudera and go to Cloudera Manager. Use the credentials provided by the terminal to login to Cloudera Manager.

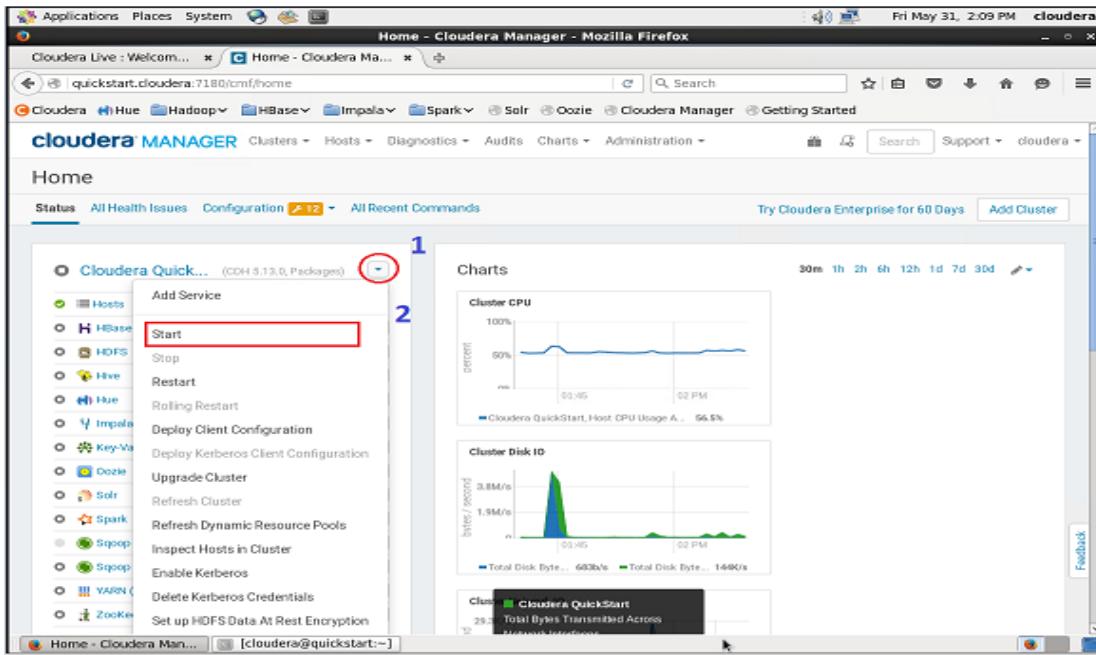


Northeastern University Assignment 02

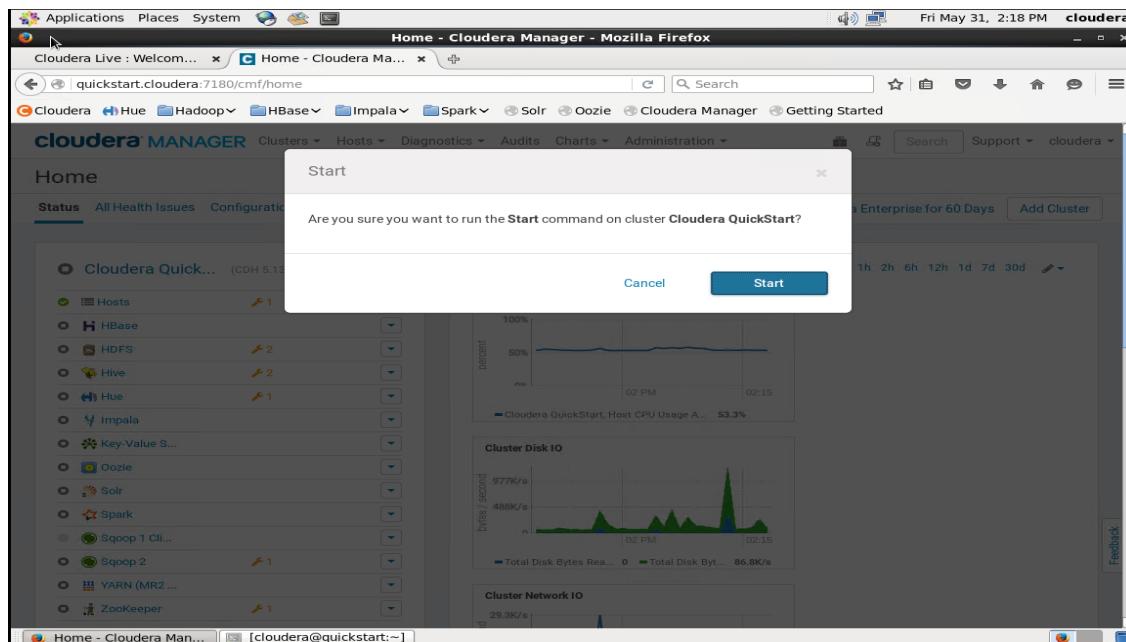
ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

14. Go to the down arrow besides “Cloudera Quickstart” and click on “Start”.



15. We can then again see a start prompt. Click on “Start” as shown below. It makes all the services to run.

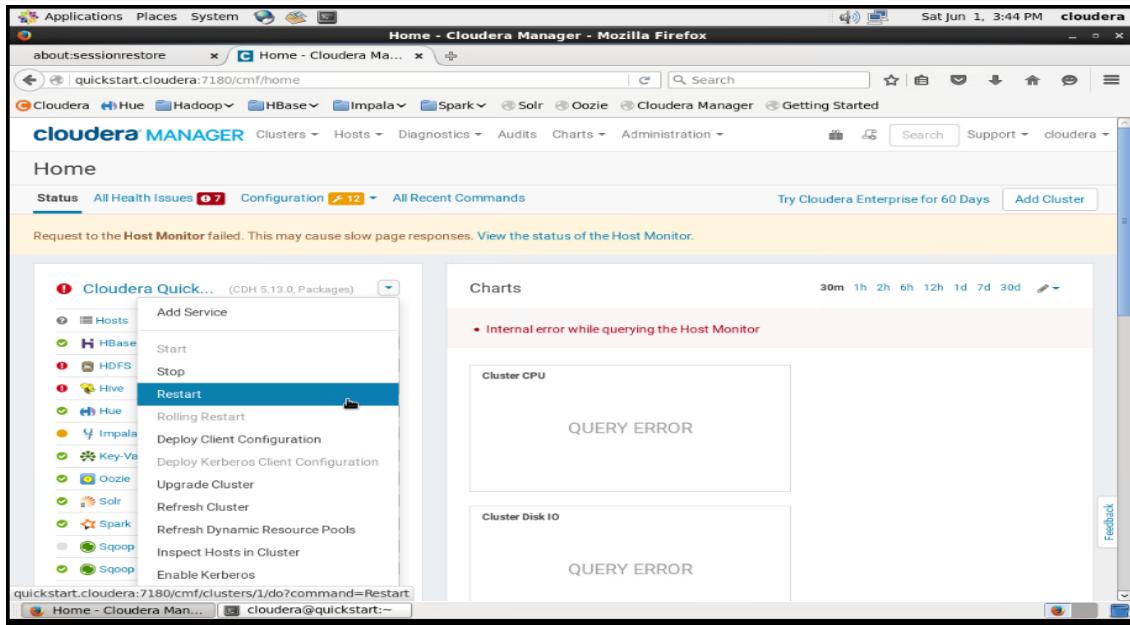


Northeastern University Assignment 02

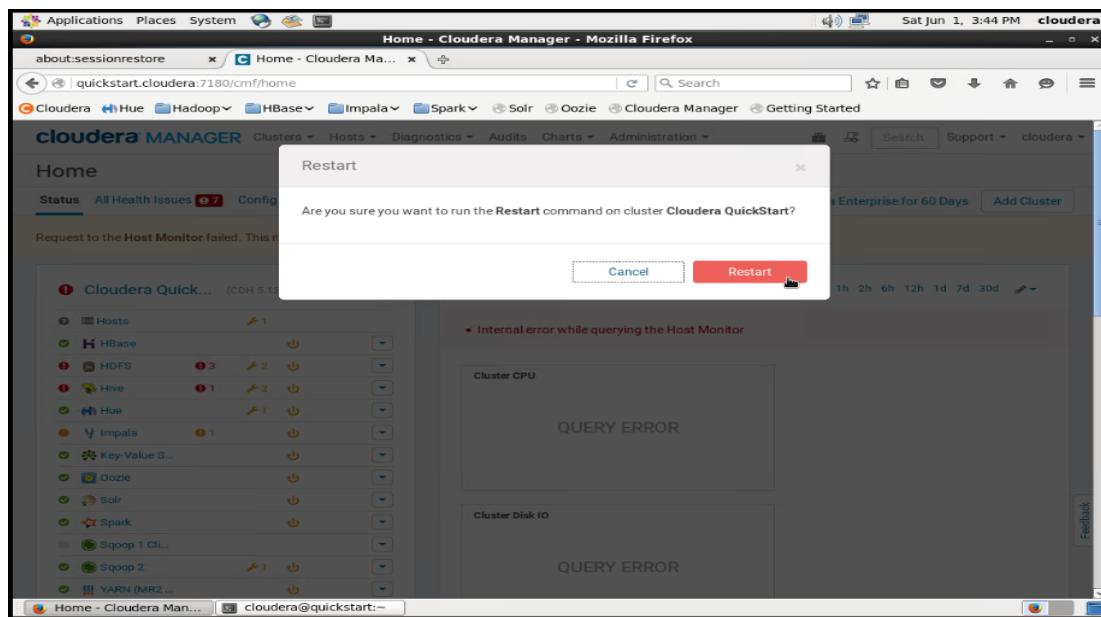
ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

16. Due to inactivity, I had been logged out of my Azure Lab, so I again followed the same steps and logged on to Cloudera Manager. Now, instead of Start, I can see "Restart" under Cloudera Quickstart. Refer the below screenshot.



17. Confirm that you want to restart all the services for Cloudera Quickstart.

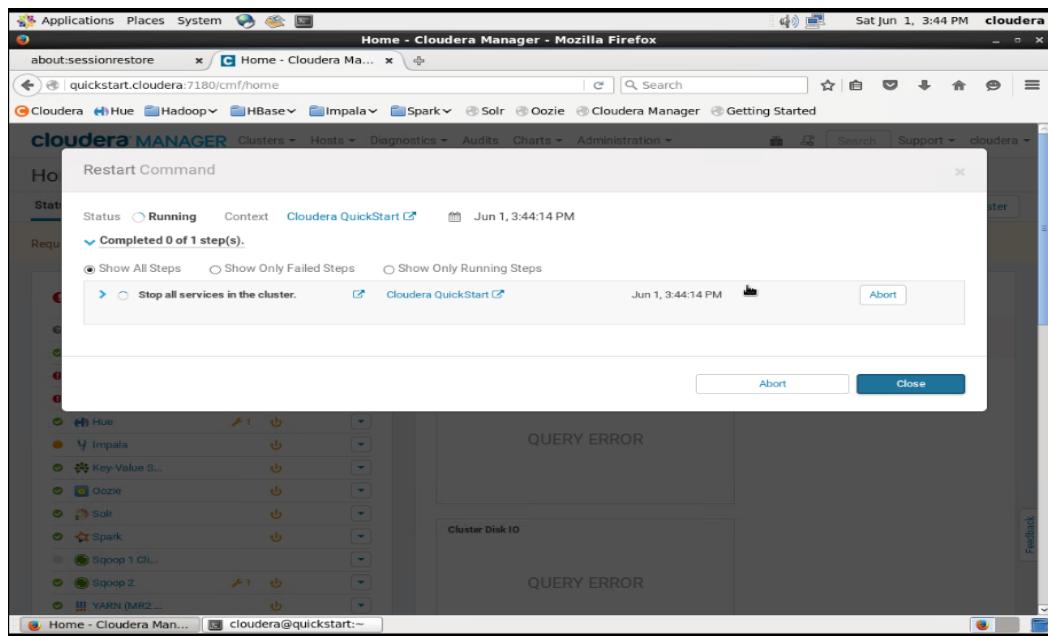


Northeastern University Assignment 02

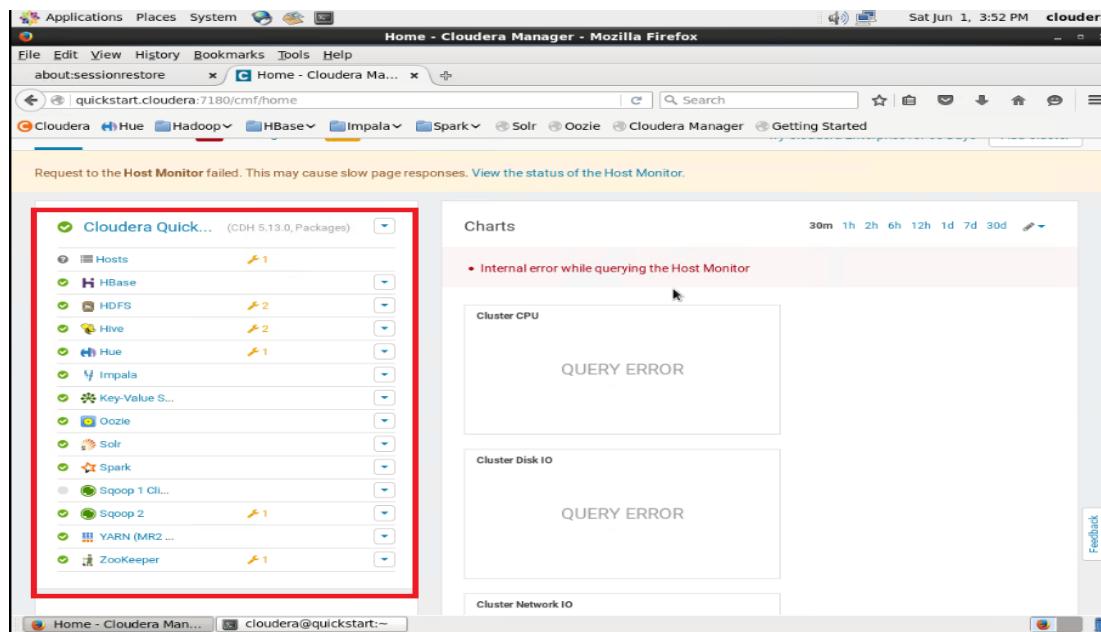
ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

18. See the below screenshot to see how all the services are getting started.



19. As you can see below, there is a green tick against the service that is successfully running.

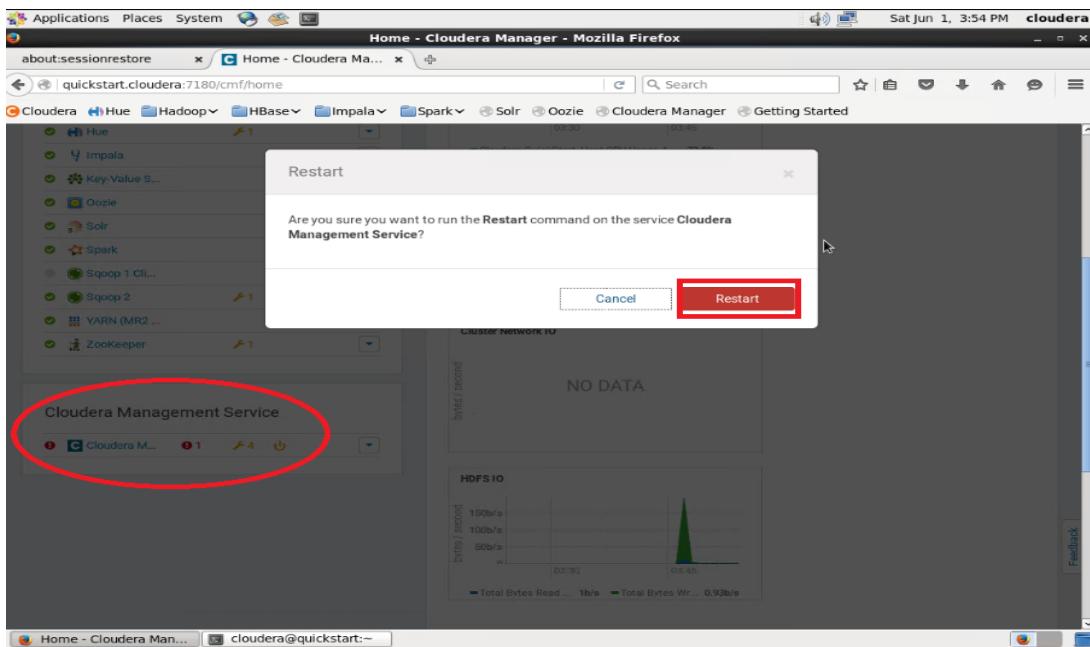


Northeastern University Assignment 02

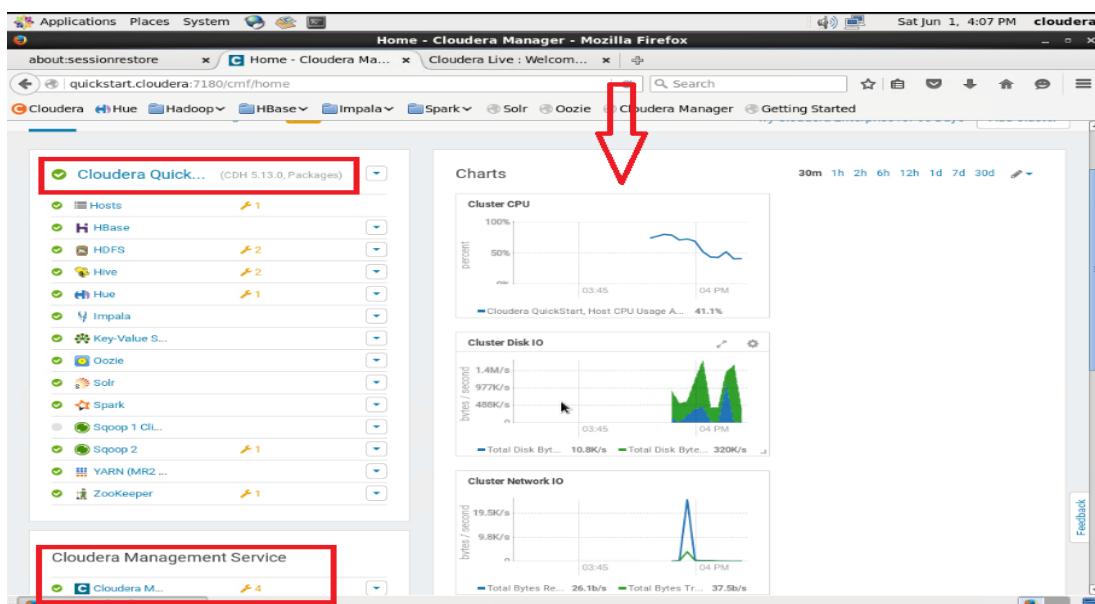
ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

20. Similarly, we can “Restart” the services under Cloudera Management Service as shown below.



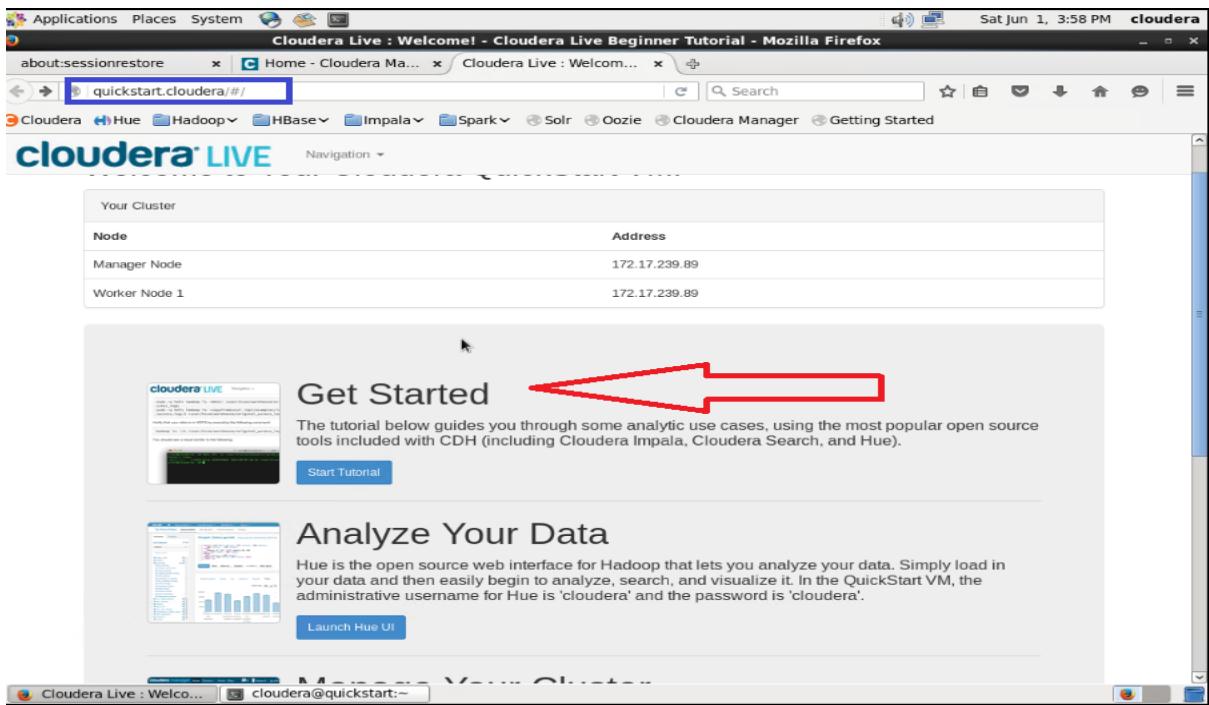
21. The below screenshot shows that all our services are now running. We can also see the charts showing the Cluster CPU, Disk IO & Network IO.



Problem 2:

- a. Use this page as a guide: <https://www.cloudera.com/developers/get-started-with-hadoop-tutorial/setup.html>
- b. Complete the "The "Getting Started with Hadoop" Tutorial at <https://www.cloudera.com/developers/get-started-with-hadoop-tutorial/setup.html>.

Go to Cloudera Live through the URL “quickstart.cloudera/#/” to start the tutorials “Get Started” by clicking on “Start Tutorial”.



Below are the steps which were followed to complete the above tutorial.

The “Getting Started with Hadoop” Tutorial

Defining our Business Question: The first step in data analysis is to define a business question. Below screenshot gives a detailed description of the scenario for Tutorial 1.

Northeastern University Assignment 02

ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

The screenshot shows a Mozilla Firefox browser window with the title 'Cloudera Live : Welcome! - Cloudera Live Beginner Tutorial - Mozilla Firefox'. The address bar shows 'quickstart.cloudera/#/tutorial/getting_started'. The page content is titled 'cloudera LIVE' with a 'Navigation' dropdown. Under 'Getting Started', there is a red box around the 'Define a Business Question' link. Below it, a paragraph of text is present. To the right, a sidebar titled 'Tutorial Exercise 1 >' contains sections like 'Good to Know' and 'Tutorial Exercise 1 >'. A red box highlights the 'Tutorial Exercise 1 >' link.

Tutorial Exercise 1: Ingest and Query Relational Data

Business Question for this Scenario: ***What products do the customers like to buy?***

The screenshot shows the 'Tutorial Exercise 1' section of the Cloudera Live tutorial. The top navigation bar includes 'Getting Started' and 'Tutorial Exercise 1 >'. The main content area is titled 'Tutorial Exercise 1' and 'Ingest and Query Relational Data'. It discusses the business question and how to answer it using Cloudera's platform. To the right, a sidebar titled 'Showing Big Data Value >' contains a section about Apache Sqoop, its purpose, and its two versions. A red box highlights the 'Showing Big Data Value >' link.

Ingest and Query Relational Data

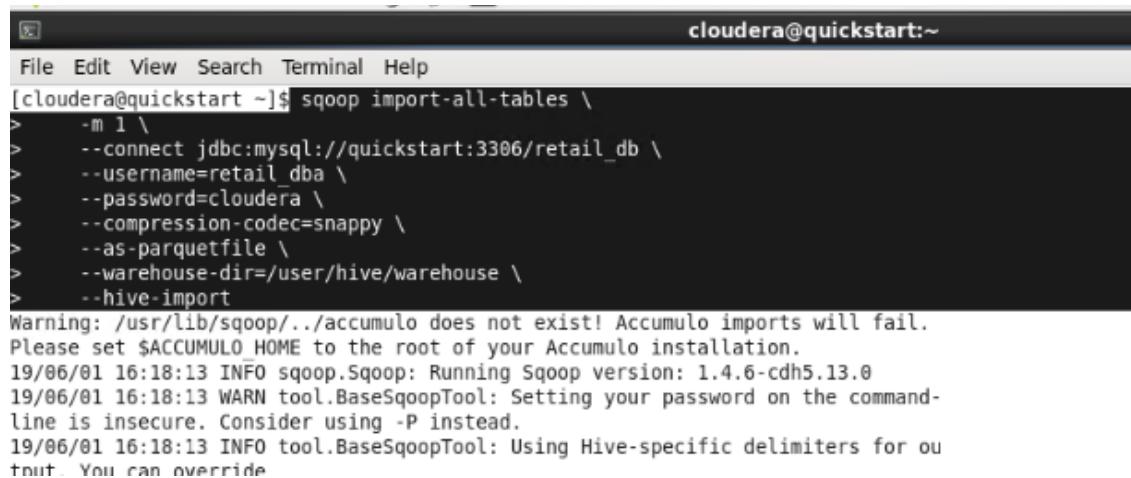
Northeastern University Assignment 02

ALY6110 Data Management and Big Data

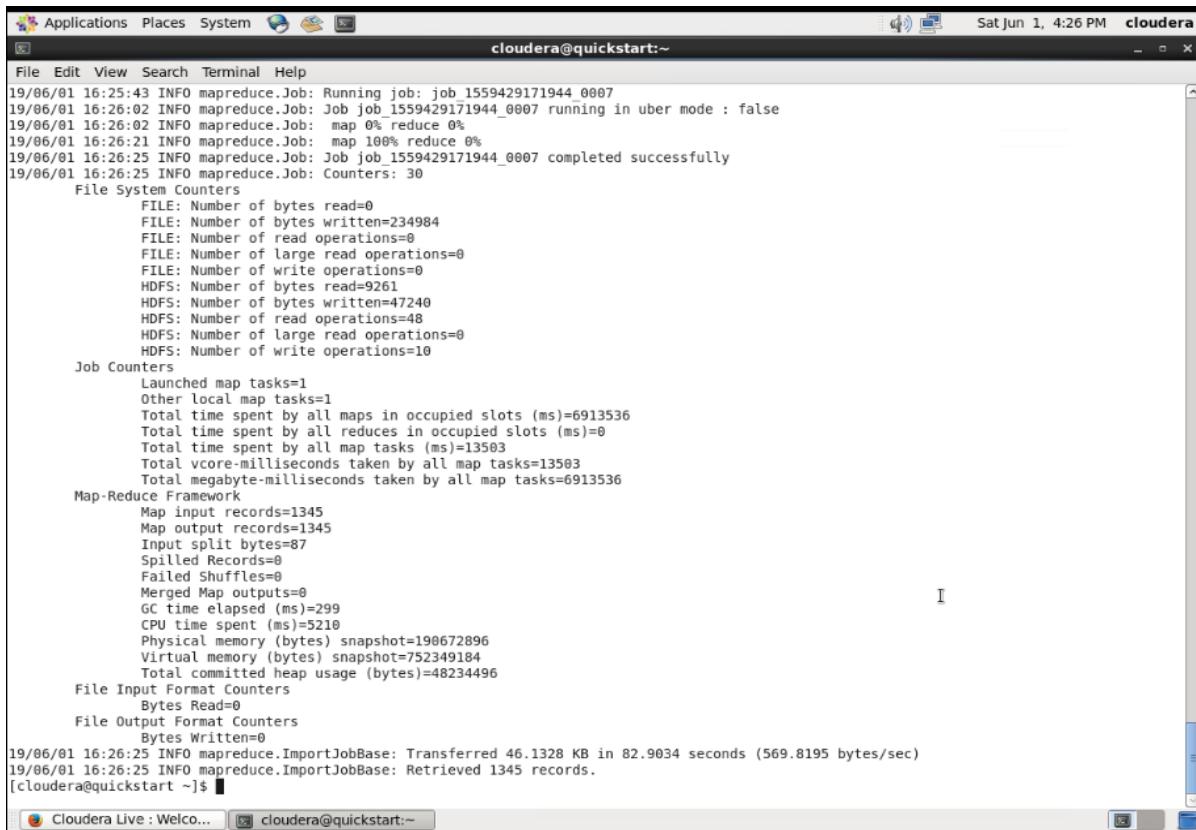
<https://northeastern.blackboard.com>

In this tutorial, we would be using Apache Sqoop to transfer data between Hadoop clusters and relational database. It works on parallel processing.

We have opened the terminal and written the command as shown in the below image.



```
cloudera@quickstart:~$ sqoop import-all-tables \
>   -m 1 \
>   --connect jdbc:mysql://quickstart:3306/retail_db \
>   --username=retail_dba \
>   --password=cloudera \
>   --compression-codec=snappy \
>   --as-parquetfile \
>   --warehouse-dir=/user/hive/warehouse \
>   --hive-import
Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
19/06/01 16:18:13 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
19/06/01 16:18:13 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
19/06/01 16:18:13 INFO tool.BaseSqoopTool: Using Hive-specific delimiters for output. You can override
```



```
cloudera@quickstart:~$ 
File Edit View Search Terminal Help
19/06/01 16:25:43 INFO mapreduce.Job: Running job: job_1559429171944_0007
19/06/01 16:26:02 INFO mapreduce.Job: Job job_1559429171944_0007 running in uber mode : false
19/06/01 16:26:02 INFO mapreduce.Job: map 0% reduce 0%
19/06/01 16:26:21 INFO mapreduce.Job: map 100% reduce 0%
19/06/01 16:26:25 INFO mapreduce.Job: Job job_1559429171944_0007 completed successfully
19/06/01 16:26:25 INFO mapreduce.Job: Counters: 30
  File System Counters
    FILE: Number of bytes read=0
    FILE: Number of bytes written=234984
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=9261
    HDFS: Number of bytes written=47240
    HDFS: Number of read operations=48
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=10
  Job Counters
    Launched map tasks=1
    Other local map tasks=1
    Total time spent by all maps in occupied slots (ms)=6913536
    Total time spent by all reduces in occupied slots (ms)=0
    Total time spent by all map tasks (ms)=13503
    Total vcore-milliseconds taken by all map tasks=13503
    Total megabyte-milliseconds taken by all map tasks=6913536
  Map-Reduce Framework
    Map input records=1345
    Map output records=1345
    Input split bytes=87
    Spilled Records=0
    Failed Shuffles=0
    Merged Map outputs=0
    GC time elapsed (ms)=299
    CPU time spent (ms)=5218
    Physical memory (bytes) snapshot=190672896
    Virtual memory (bytes) snapshot=752349184
    Total committed heap usage (bytes)=48234496
  File Input Format Counters
    Bytes Read=0
  File Output Format Counters
    Bytes Written=0
19/06/01 16:26:25 INFO mapreduce.ImportJobBase: Transferred 46.1328 KB in 82.9034 seconds (569.8195 bytes/sec)
19/06/01 16:26:25 INFO mapreduce.ImportJobBase: Retrieved 1345 records.
[cloudera@quickstart ~]$ 
```

Northeastern University Assignment 02

ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

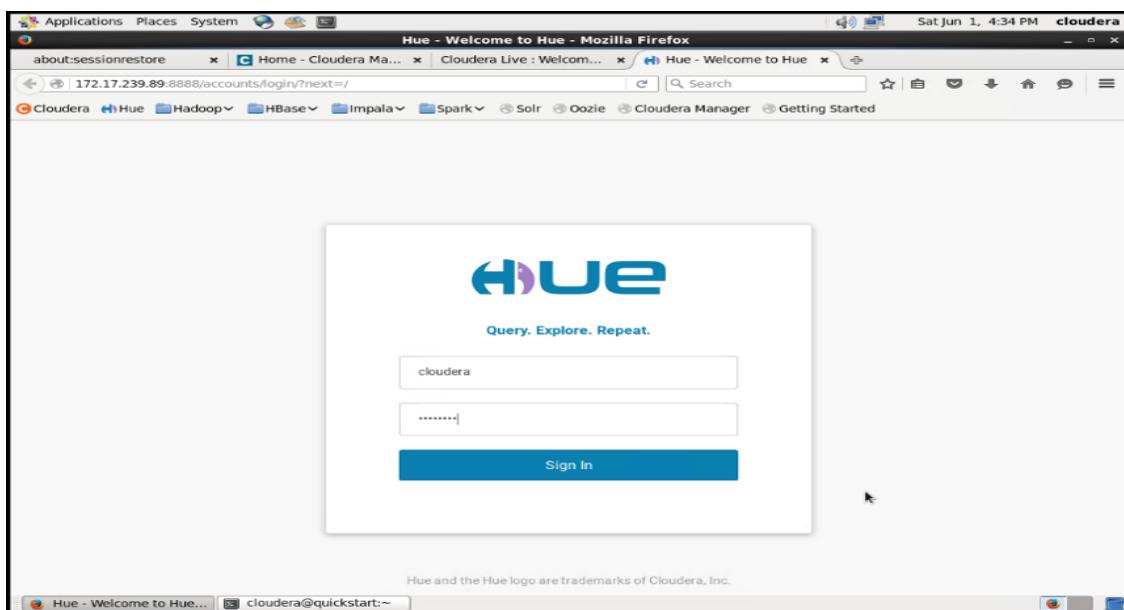
In the above image, we can see how Sqoop uses MapReduce jobs to pull the data from our MySQL database and write the data to HDFS in parallel, distributed across the cluster in Apache Parquet format.

Verification: We will now confirm if our data files exist in HDFS using the commands as shown in the screenshot. We can see the directories and files inside HDFS.

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse
Found 6 items
drwxrwxrwx - cloudera supergroup      0 2019-06-01 16:19 /user/hive/wareho
use/categories
drwxrwxrwx - cloudera supergroup      0 2019-06-01 16:20 /user/hive/wareho
use/customers
drwxrwxrwx - cloudera supergroup      0 2019-06-01 16:22 /user/hive/wareho
use/departments
drwxrwxrwx - cloudera supergroup      0 2019-06-01 16:23 /user/hive/wareho
use/order_items
drwxrwxrwx - cloudera supergroup      0 2019-06-01 16:24 /user/hive/wareho
use/orders
drwxrwxrwx - cloudera supergroup      0 2019-06-01 16:26 /user/hive/wareho
use/products
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/categories/
Found 4 items
drwxr-xr-x - cloudera supergroup      0 2019-05-07 06:48 /user/hive/warehouse/categories/.metadata
drwxr-xr-x - cloudera supergroup      0 2019-06-01 16:19 /user/hive/warehouse/categories/.signals
-rw-r--r--  1 cloudera supergroup    1957 2019-06-01 16:19 /user/hive/warehouse/categories/0d892943-41b5-470e-88fd-78563740c433.parquet
-rw-r--r--  1 cloudera supergroup    1957 2019-05-07 06:41 /user/hive/warehouse/categories/5a6f1112-ea16-4844-86f3-1f271f217418.parquet
[cloudera@quickstart ~]$
```

Cloudera Live : Welco... cloudera@quickstart:~

Using Hue: Hue is a web-based interface for many of the tools in CDH and we would be using Impala query editor in Hue to query out tables. Open the Hue webpage from the browser in Cloudera. Enter the Username and Password for Hue in VM is the “*Cloudera*”.

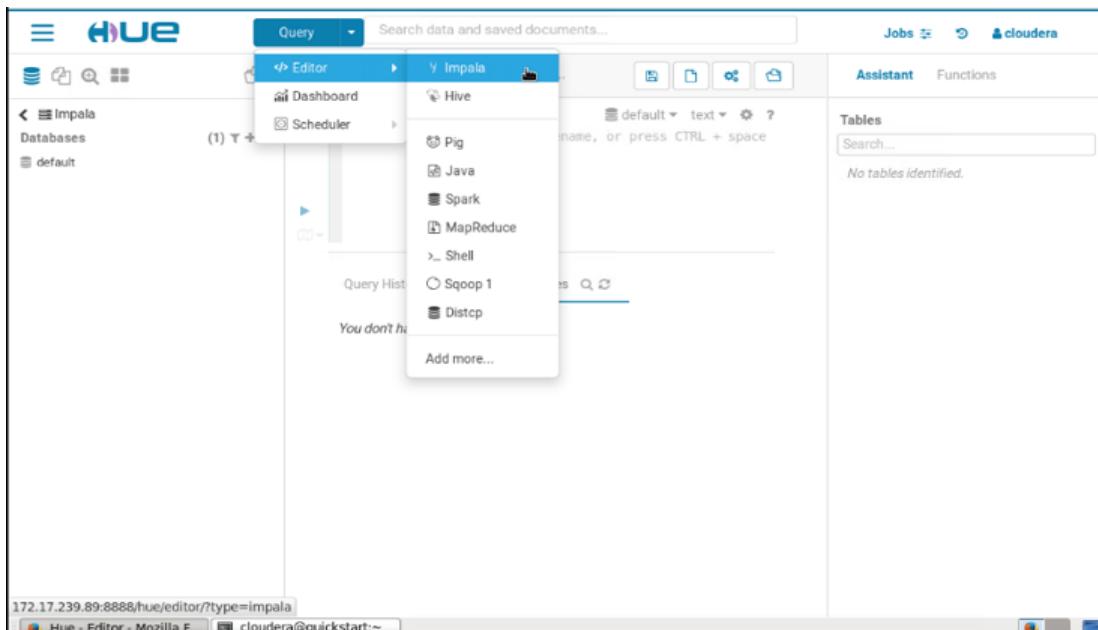


Northeastern University Assignment 02

ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

Once, we go inside Hue, go to **Query > Editor > Impala**. This will open the Impala Query Editor for us. Refer the below image for better understanding.



Impala Query Editor in Hue

We first need to inform Impala that its metadata is out of date and it is done by executing the query "*invalidate metadata*". Then we use the query "*show tables*" to have a look at the tables as shown below.

```
1 invalidate metadata;
2 show tables;
```

name
1 categories
2 customers
3 departments
4 order_items
5 orders
6 products

Northeastern University Assignment 02

ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

The query executes successfully to give us the list of six tables.

Now, we will focus on our business question. We have written the SQL queries to calculate total revenue per product and for displaying the top 10 revenue generating products as shown below.

-- Most popular product categories

```
select c.category_name, count(order_item_quantity) as count
from order_items oi
inner join products p on oi.order_item_product_id = p.product_id
inner join categories c on c.category_id = p.product_category_id
group by c.category_name
order by count desc
limit 10;
```

The screenshot shows the Hue interface with the following details:

- Query Editor:** Displays the SQL query for finding the most popular product categories. The code is:

```
4 -- Most popular product categories
5 select c.category_name, count(order_item_quantity) as count
6 from order_items oi
7 inner join products p on oi.order_item_product_id = p.product_id
8 inner join categories c on c.category_id = p.product_category_id
9 group by c.category_name
10 order by count desc
11 limit 10;|
```
- Results:** Shows the output of the query as a table titled "Results (10)".

	category_name	count
1	Cleats	196408
2	Men's Footwear	177968
3	Women's Apparel	168280
4	Indoor/Outdoor Games	154384
5	Fishing	138600
6	Water Sports	124320
7	Camping & Hiking	109832
8	Cardio Equipment	99896
9	Shop By Sport	87872
10	Electronics	25248
- Tables:** A sidebar showing three tables: default.categories, default.order_items, and default.products.

Most popular product categories

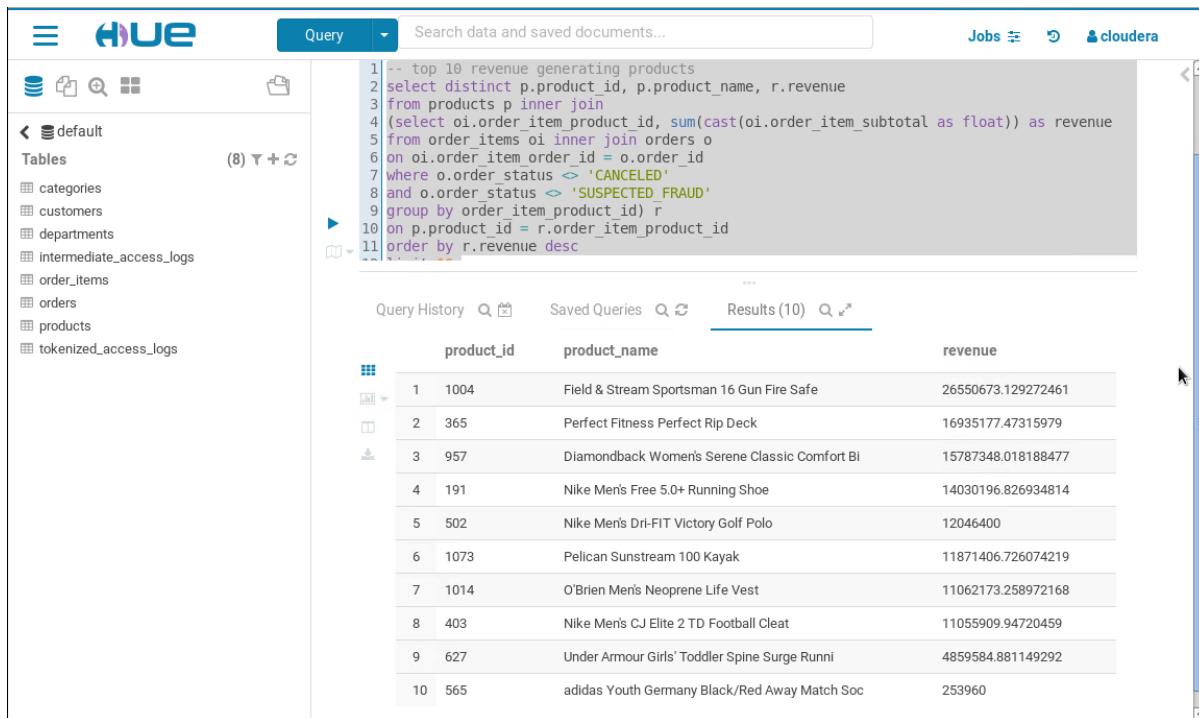
Northeastern University Assignment 02

ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

-- top 10 revenue generating products

```
select distinct p.product_id, p.product_name, r.revenue
from products p inner join
(select oi.order_item_product_id, sum(cast(oi.order_item_subtotal as float)) as revenue
from order_items oi inner join orders o
on oi.order_item_order_id = o.order_id
where o.order_status <> 'CANCELED'
and o.order_status <> 'SUSPECTED_FRAUD'
group by order_item_product_id) r
on p.product_id = r.order_item_product_id
order by r.revenue desc
limit 10;
```



The screenshot shows the Hue interface forApache Hadoop. The left sidebar lists tables: default, categories, customers, departments, intermediate_access_logs, order_items, orders, products, and tokenized_access_logs. The main area displays the query code and its results.

```
1 -- top 10 revenue generating products
2 select distinct p.product_id, p.product_name, r.revenue
3 from products p inner join
4 (select oi.order_item_product_id, sum(cast(oi.order_item_subtotal as float)) as revenue
5 from order_items oi inner join orders o
6 on oi.order_item_order_id = o.order_id
7 where o.order_status <> 'CANCELED'
8 and o.order_status <> 'SUSPECTED_FRAUD'
9 group by order_item_product_id) r
10 on p.product_id = r.order_item_product_id
11 order by r.revenue desc
```

Query History: 10 queries saved. Saved Queries: 0. Results (10):

	product_id	product_name	revenue
1	1004	Field & Stream Sportsman 16 Gun Fire Safe	26550673.129272461
2	365	Perfect Fitness Perfect Rip Deck	16935177.47315979
3	957	Diamondback Women's Serene Classic Comfort BI	15787348.018188477
4	191	Nike Men's Free 5.0+ Running Shoe	14030196.826934814
5	502	Nike Men's Dri-FIT Victory Golf Polo	12046400
6	1073	Pelican Sunstream 100 Kayak	11871406.726074219
7	1014	O'Brien Men's Neoprene Life Vest	11062173.258972168
8	403	Nike Men's CJ Elite 2 TD Football Cleat	11055909.94720459
9	627	Under Armour Girls' Toddler Spine Surge Runni	4859584.881149292
10	565	adidas Youth Germany Black/Red Away Match Soc	253960

Top 10 revenue generating products

From the results, we can easily identify the most popular products as well as the top 10 revenue generating products. For e.g., Field & Stream Sportsman 16 Gun Fire Safe is the product that is generating the maximum revenue for the company.

Conclusion

Through this tutorial we learned how to Sqoop structured data into HDFS, transforming of the data into Parquet file format and creating hive tables. We also learned how to use SQL like queries in Hadoop environment with the help of Impala query editor.

Tutorial Exercise 2: Correlate Structured Data with Unstructured Data

In this tutorial, we are using *Apache Flume* to ingest web clickstream data. Flume is a scalable real-time ingest framework that allows us to route, filter, aggregate, and do "mini-operations" on data.

The screenshot shows the Cloudera LIVE interface. At the top, there's a navigation bar with 'cloudera LIVE', 'Navigation ▾', and 'Tutorial Exercise 2 ▾'. Below the navigation, there are two links: '< Showing Big Data Value' on the left and 'Advanced Analytics in the Same Platform >' on the right. The main content area is titled 'Tutorial Exercise 2' and contains the following text:

Tutorial Exercise 2

Correlate Structured Data with Unstructured Data

Since you are a pretty smart data person, you realize another interesting business question would be: *are the most viewed products also the most sold?* (or for other scenarios, the most searched for, the most chatted about...). Since Hadoop can store unstructured and semi-structured data alongside structured data without remodelling an entire database, you can just as well ingest, store and process web log events. Let's find out what site visitors have actually viewed the most.

For this, you need the web clickstream data. The most common way to ingest web clickstream is to use Flume. Flume is a scalable real-time ingest framework that allows you to route, filter, aggregate, and do "mini-operations" on data on its way in to the scalable processing platform.

In Exercise 4, later in this tutorial, you can explore a Flume configuration example, to use for real-time ingest and transformation of our sample web clickstream data. However, for the sake of tutorial-time, in this step, we will not have the patience to wait for three days of data to be ingested. Instead, we prepared a web clickstream data set (just pretend you fast forwarded three days) that you can bulk upload into HDFS directly.

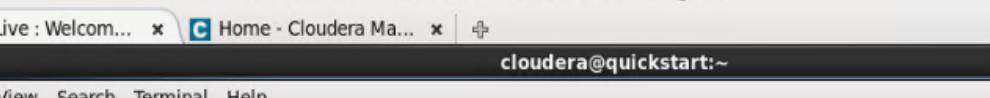
Bulk Upload Data

There are some pre-loaded sample access log data into `/opt/examples/log_data/access.log.2`. We have moved data from local file system to HDFS with the help of the commands as shown in the below image.

Northeastern University Assignment 02

ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>



The screenshot shows a terminal window titled "cloudera@quickstart:~". The user has run several commands to manage files in HDFS:

- [cloudera@quickstart ~]\$ sudo -u hdfs hadoop fs -mkdir /user/hive/warehouse/original_access_logs
- [cloudera@quickstart ~]\$ sudo -u hdfs hadoop fs -copyFromLocal /opt/examples/log_files/access.log.2 /user/hive/warehouse/original_access_logs
- [cloudera@quickstart ~]\$ hadoop fs -ls /user/hive/warehouse/original_access_logs

The last command's output is shown:

-rw-r--r-- 1 hdfs supergroup 39593868 2019-06-02 14:05 /user/hive/warehouse/original_access_logs/access.log.2

Now we must build a table in Hive and query the data using Apache Impala and Hue. The first step in this would be to parse the logs into individual fields using a regular expression using SerDes. The second step would be transferring the data from this table to the one which doesn't need any SerDe.

Let's go to Hive query editor in Hue to execute the queries shown in below image.

cloudera® LIVE Tutorial Exercise 2 ▾

```
CREATE EXTERNAL TABLE intermediate_access_logs (
    ip STRING,
    date STRING,
    method STRING,
    url STRING,
    http_version STRING,
    code1 STRING,
    code2 STRING,
    dash STRING,
    user_agent STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
    'input.regex' = '([^\"]* - - \[[^\]]*\] *)[\"][^\"]*([^\"]*) ([^\"]*) ([^\"]*) ([^\"]*) ([^\"]*) ([^\"]*)',
    'output.format.string' = "%1$S %2$S %3$S %4$S %5$S %7$S %8$S %9$S")
LOCATION '/user/hive/warehouse/original_access_logs';

CREATE EXTERNAL TABLE tokenized_access_logs (
    ip STRING,
    date STRING,
    method STRING,
    url STRING,
    http_version STRING,
    code1 STRING,
    code2 STRING,
    dash STRING,
    user_agent STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/user/hive/warehouse/tokenized_access_logs';

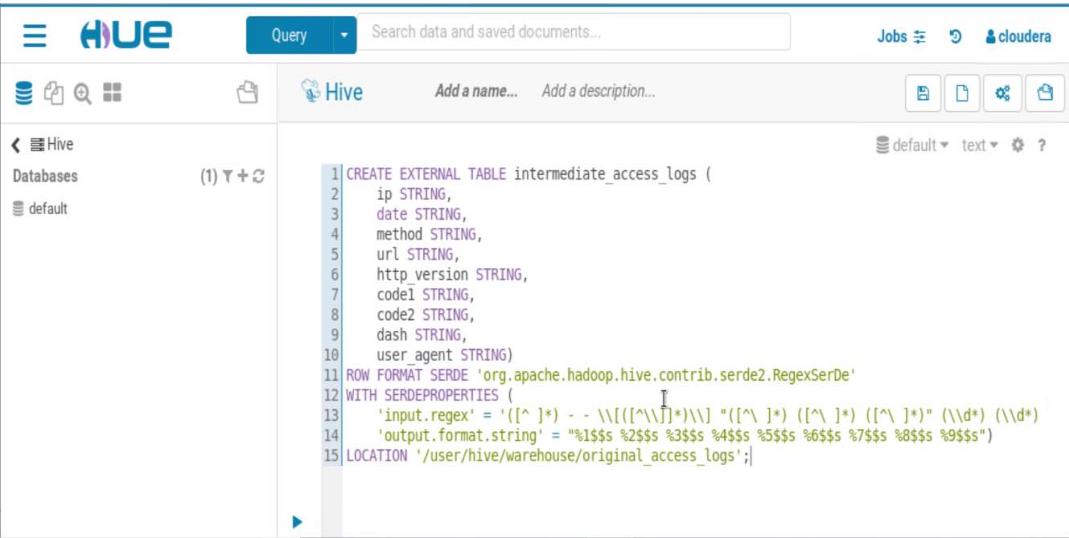
ADD JAR /usr/lib/hive/lib/hive-contrib.jar;

INSERT OVERWRITE TABLE tokenized_access_logs SELECT * FROM intermediate_access_logs;
```

Northeastern University Assignment 02

ALY6110 Data Management and Big Data

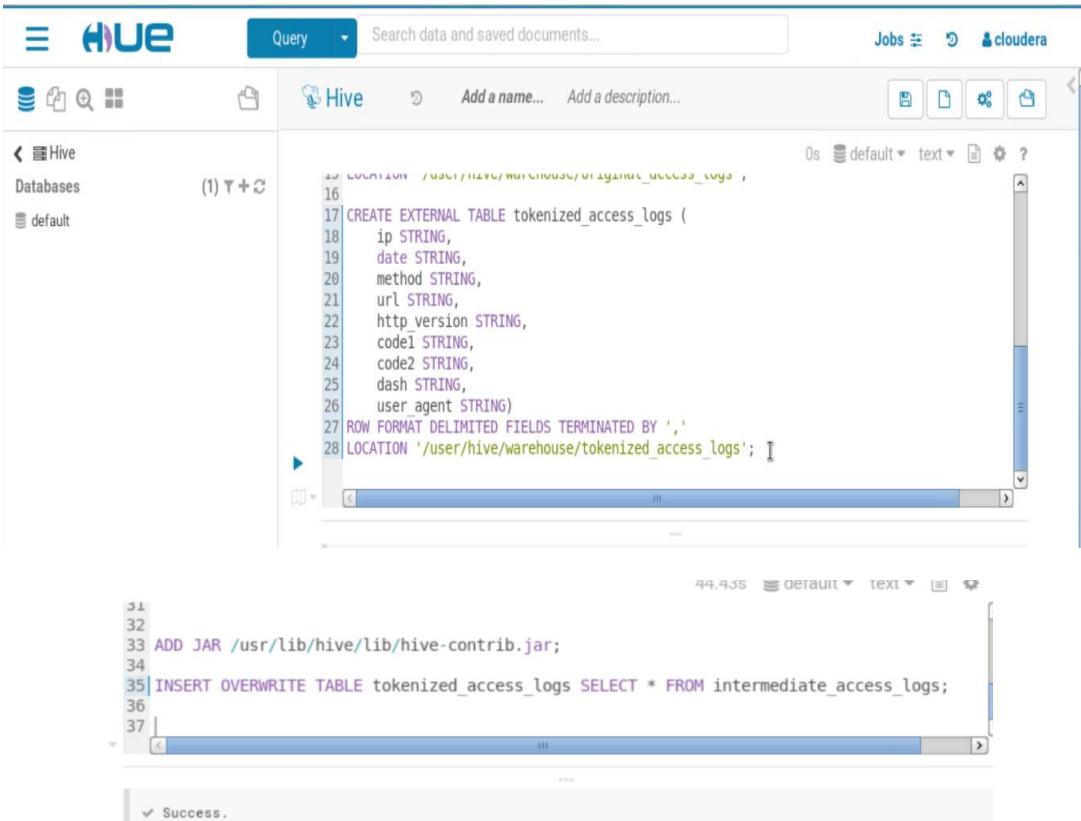
<https://northeastern.blackboard.com>



The screenshot shows the Hue interface for Apache Hadoop. In the top navigation bar, 'HUE' is selected. The main area is a 'Query' editor titled 'Hive'. On the left sidebar, under 'Databases', 'default' is selected. The central pane displays the following Hive query:

```
1 CREATE EXTERNAL TABLE intermediate_access_logs (
2   ip STRING,
3   date STRING,
4   method STRING,
5   url STRING,
6   http_version STRING,
7   code1 STRING,
8   code2 STRING,
9   dash STRING,
10  user_agent STRING)
11 ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'
12 WITH SERDEPROPERTIES (
13   'input.regex' = '([^\s]+) - - ([^\[\]]*) [^"] "([^\s"]*) ([^\s"]*) ([^\s"]*) (\d*) (\d*)'
14   'output.format.string' = "%1$s %2$$ %3$$ %4$$ %5$$ %6$$ %7$$ %8$$ %9$$";
15 LOCATION '/user/hive/warehouse/original_access_logs';
```

The second query which is shown in the below image takes some time to run. It works in the same way as Sqoop import works using a MapReduce job.



The screenshot shows the Hue interface for Apache Hadoop. In the top navigation bar, 'HUE' is selected. The main area is a 'Query' editor titled 'Hive'. On the left sidebar, under 'Databases', 'default' is selected. The central pane displays two queries:

```
16 LOCATION '/user/hive/warehouse/intermediate_access_logs';
17 CREATE EXTERNAL TABLE tokenized_access_logs (
18   ip STRING,
19   date STRING,
20   method STRING,
21   url STRING,
22   http_version STRING,
23   code1 STRING,
24   code2 STRING,
25   dash STRING,
26   user_agent STRING)
27 ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
28 LOCATION '/user/hive/warehouse/tokenized_access_logs';
```

Below this, another query is partially visible:

```
31
32
33 ADD JAR /usr/lib/hive/lib/hive-contrib.jar;
34
35 INSERT OVERWRITE TABLE tokenized_access_logs SELECT * FROM intermediate_access_logs;
36
37 |
```

In the bottom status bar, there is a message: 'Success.'

Northeastern University Assignment 02

ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

Now, we need to go to Impala query editor and enter the queries as shown in the below image.

The screenshot displays two windows of the Hue interface. The top window is the 'Hive' editor, showing an Impala query:

```
41 invalidate metadata;
42
43
44 select count(*),url from tokenized_access_logs
45 where url like '%/product/%'
46 group by url order by count(*) desc;  I
47
```

The status bar indicates 'Success.' and '44.43s'. The bottom window is the 'Results' view, showing the output of the query:

	count(*)	url
1	1926	/department/apparel/category/cleats/product/Perfect%20Fitness%20Perfect%20Rip%20Deck
2	1793	/department/apparel/category/featured%20shops/product/adidas%20Kids%20RG%20III%20Mid%20Football%20Cleat
3	1780	/department/golf/category/women's%20apparel/product/Nike%20Men's%20Dri-FIT%20Victory%20Golf%20Polo
4	1757	/department/apparel/category/men's%20footwear/product/Nike%20Men's%20CJ%20Elite%20%20TD%20Football%20Cleat
5	1104	/department/fan%20shop/category/water%20sports/product/Pelican%20Sunstream%20100%20Kayak
6	1084	/department/fan%20shop/category/indoor%20outdoor%20games/product/O'Brien%20Men's%20Neoprene%20Life%20Vest
7	1059	/department/fan%20shop/category/camping%20%20hiking/product/Diamondback%20Women's%20Serene%20Classic%20Comfort%20Bl
8	1028	/department/fan%20shop/category/fishing/product/Field%20%20Stream%20Sportsman%2016%20Gun%20Fire%20Safe
9	1004	/department/footwear/category/cardio%20equipment/product/Nike%20Men's%20Free%205.0%20Running%20Shoe
10	939	/department/footwear/category/fitness%20accessories/product/Under%20Armour%20Hustle%20Storm%20Medium%20Duffle%20Bag

Output: 152 rows returned

After carefully observing the results, we found that the list of most sold products is displayed from the previous tutorial steps, but there is one product which was missing in the previous list.

In this example of DataCo from Tutorial Exercise 1, the reason for that missing product could be assumed as, on the view page where a lot of visitors stopped and checked that product but because of a typo in the price of that item. And after the typo was fixed and the displayed price was corrected, the sales for that product increased rapidly.

Refer the below screenshot from the tutorial link for this missing product details.

Northeastern University Assignment 02

ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

product_id	product_name	revenue
1	1004	Field & Stream Sportsman 16 Gun Fire Safe
2	365	Perfect Fitness Perfect Rip Deck
3	957	Diamondback Women's Serene Classic Comfort BI
4	191	Nike Men's Free 5.0+ Running Shoe
5	502	Nike Men's Dri-FIT Victory Golf Polo
6	1073	Pelican Sunstream 100 Kayak
7	1014	O'Brien Men's Neoprene Life Vest
8	403	Nike Men's CJ Elite 2 TD Football Cleat
9	627	Under Armour Girls' Toddler Spine Surge Runni
10	565	adidas Youth Germany Black/Red Away Match Soc

count(*)	url
1	/department/apparel/category/cleats/product/Perfect%20Fitness%20Perfect%20Rip%20Deck
2	/department/apparel/category/featured%20shops/product/adidas%20kids%20ORG%20III%20Mid%20Football%20Cleat
3	/department/golf/category/women's%20apparel/product/Nike%20Men's%20Dri-FIT%20Victory%20Golf%20Polo
4	/department/apparel/category/men's%20footwear/product/Nike%20Men's%20CJ%20Elite%20%20TD%20Football%20Cleat
5	/department/fan%20shop/category/water%20sports/product/Pelican%20Sunstream%20100%20Kayak
6	/department/fan%20shop/category/indoor%20outdoor%20games/product/O'Brien%20Men's%20Neoprene%20Life%20Vest
7	/department/fan%20shop/category/camping%20%20hiking/product/Diamondback%20Women's%20Serene%20Classic%20Comfort%20BI
8	/department/fan%20shop/category/fishing/product/Field%20%20Stream%20Sportsman%2016%20Gun%20Fire%20Safe
9	/department/footwear/category/cardio%20equipment/product/Nike%20Men's%20Free%205.0%20Running%20Shoe
10	/department/footwear/category/fitness%20accessories/product/Under%20Armour%20Hustle%20Storm%20Medium%20Duffle%20Bag

Conclusion

Through this tutorial we got to know that if we correlate two datasets for the same business questions adds value to any business and as the analysis was being on the same platform, it makes life easy for both the big data analyst as well as for the business.

Tutorial Exercise 3: Relationship Strength Analytics using Spark

In this tutorial, we are using Apache Spark which uses the similar concepts of MapReduce operations, but it has the ability to perform in-memory operations and thus gives concise results very fast.

Northeastern University Assignment 02

ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

The screenshot shows a Firefox browser window with the title "Cloudera Live : Welcome! - Cloudera Live Beginner Tutorial - Mozilla Firefox". The address bar shows "quickstart.cloudera/#/tutorial/relationship_strength_using_spark". The main content area is titled "Tutorial Exercise 3" and "Relationship strength analytics using Spark". It contains text about product correlations and a terminal window showing the command "[cloudera@quickstart ~]\$ spark-shell --master yarn-client". To the right, a sidebar titled "About Spark" compares it to MapReduce, noting its conciseness and iterative nature. The bottom of the browser window shows the status bar with "Cloudera Live : Welco..." and "[cloudera@quickstart:/...]".

To install Spark on Cloudera, we write the command shown in the below screenshot.

The screenshot shows a terminal window with the title "cloudera@quickstart:~". The command "[cloudera@quickstart ~]\$ spark-shell --master yarn-client" is entered. The output includes the Scala logo, the message "version 1.6.0", and the Java version information "Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_67)".

Northeastern University Assignment 02

ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

The below screenshot shows that the Spark has been successfully installed as it shows Scala at the terminal prompt.

```
[cloudera@quickstart ~]$ spark-shell --master yarn-client
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
Welcome to

    / \ \ / \ / \ \ / \ / \
   / \ \ / \ / \ \ / \ / \ \ / \
  / \ \ / \ / \ \ / \ / \ \ / \
 / \ \ / \ / \ \ / \ / \ \ / \
version 1.6.0

Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_67)
Type in expressions to have them evaluated.
Type :help for more information.
Spark context available as sc (master = yarn-client, app id = application_1559445471030_0002).
19/06/01 21:23:17 WARN metastore.ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not enabled so recording the schema version 1.1.0-cdh5.13.0
19/06/01 21:23:17 WARN metastore.ObjectStore: Failed to get database default, returning NoSuchObjectException
SQL context available as sqlContext.

scala> // First we're going to import the classes we need

scala> import org.apache.hadoop.mapreduce.Job
import org.apache.hadoop.mapreduce.Job

scala> import org.apache.hadoop.mapreduce.lib.input.FileInputFormat
import org.apache.avro.generic.GenericRecord
import parquet.hadoop.ParquetInputFormat
import parquet.avro.AvroReadSupport
import org.apache.spark.rdd.RDD
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat
import org.apache.avro.generic.GenericRecord
import parquet.hadoop.ParquetInputFormat
import parquet.avro.AvroReadSupport
import org.apache.spark.rdd.RDD

scala> [
```

As the Scala prompt appeared, we will write the code as shown in the below image to import the classes we need.

```
scala> // First we're going to import the classes we need  
  
scala> import org.apache.hadoop.mapreduce.Job  
import org.apache.hadoop.mapreduce.Job  
  
scala> import org.apache.hadoop.mapreduce.lib.input.FileInputFormat  
import org.apache.avro.generic.GenericRecord  
import parquet.hadoop.ParquetInputFormat  
import parquet.avro.AvroReadSupport  
import org.apache.spark.rdd.RDD  
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat  
import org.apache.avro.generic.GenericRecord  
import parquet.hadoop.ParquetInputFormat  
import parquet.avro.AvroReadSupport  
import org.apache.spark.rdd.RDD
```

Northeastern University Assignment 02

ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

After this, we will be creating RDD's for the files we imported from MySQL with Sqoop. RDD's are the low-level API for Spark's data structures for working with distributed datasets.

```
scala> // Then we create RDD's for 2 of the files we imported from MySQL with Sqoop
// RDD's are Spark's data structures for working with distributed datasets

scala> def rddFromParquetHdfsFile(path: String): RDD[GenericRecord] = {
    val job = new Job()
    FileInputFormat.setInputPaths(job, path)
    ParquetInputFormat.setReadSupportClass(job,
        classOf[AvroReadSupport[GenericRecord]])
    return sc.newAPIHadoopRDD(job.getConfiguration,
        classOf[ParquetInputFormat[GenericRecord]],
        classOf[Void],
        classOf[GenericRecord]).map(x => x._2)
}
warning: there were 1 deprecation warning(s); re-run with -deprecation for details
rddFromParquetHdfsFile: (path: String)org.apache.spark.rdd.org.apache.avro.generic.GenericRecord
```

Now, we need to extract the fields of interest from the tables *order_items* and *products*. After that, we try to get the list of every product, its name, and quantity grouped by order.

```
scala> // Next, we extract the fields from order_items and products that we care about
// and get a list of every product, its name and quantity, grouped by order

scala> val orders = order_items.map { x =>
    x.get("order_item_product_id"),
    (x.get("order_item_order_id"), x.get("order_item_quantity"))
}.join(
    products.map { x =>
        (x.get("product_id"),
        (x.get("product_name")))
    }
).map(x => (
    scala.Int.unbox(x._2._1._1), // order_id
    (
        scala.Int.unbox(x._2._1._2), // quantity
        x._2._2.toString // product_name
    )
)).groupByKey()
orders: org.apache.spark.rdd.RDD[(Int, Iterable[(Int, String)])] = ShuffledRDD[10] at groupByKey at <console>:55
```

Now, we will count how many times each combination of products appears together in an order and then we sort them and only take 10 most frequent combinations.

Northeastern University Assignment 02

ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

```
scala> val orders = order_items.map { x =>
  x.get("order_item_product_id"),
  (x.get("order_item_order_id"), x.get("order_item_quantity")))
}.join(
  products.map { x =>
    x.get("product_id"),
    (x.get("product_name"))
  }
).map(x => (
  scala.Int.unbox(x._2._1._1), // order_id
  (
    scala.Int.unbox(x._2._1._2), // quantity
    x._2._2.toString // product_name
  )
)).groupByKey()
orders: org.apache.spark.rdd.RDD[(Int, Iterable[(Int, String)])] = ShuffledRDD[10] at groupByKey at <console>:55

scala> val cooccurrences = orders.map(order =>
  (
    order._1,
    order._2.toList.combinations(2).map(order_pair =>
      (
        if (order_pair(0)._2 < order_pair(1)._2)
          (order_pair(0)._2, order_pair(1)._2)
        else
          (order_pair(1)._2, order_pair(0)._2),
        order_pair(0)._1 * order_pair(1)._1
      )
    )
  )
)
val combos = cooccurrences.flatMap(x => x._2).reduceByKey((a, b) => a + b)
val mostCommon = combos.map(x => (x._2, x._1)).sortByKey(false).take(10)
cooccurrences: org.apache.spark.rdd.RDD[(Int, Iterator[((String, String), Int)))] = MapPartitionsRDD[11] at map at <console>:43
combos: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[13] at reduceByKey at <console>:57
mostCommon: Array[(Int, (String, String))] = Array((290678,(Perfect Fitness Perfect Rip Deck,Perfect Fitness Perfect Rip Deck)), (246633,(Nike Men's Dri-FIT Victory Golf Polo,Nike Men's Dri-FIT Victory Golf Polo)), (225067,(O'Brien Men's Neoprene Life Vest,O'Brien Men's Neoprene Life Vest)), (140254,(Nike Men's Free 5.0+ Running Shoe,Nike Men's Free 5.0+ Running Shoe)), (119904,(Under Armour Girls' Toddler Spine Surge Runni,Under Armour Girls' Toddler Spine Surge Runni)), (67876,(Nike Men's Dri-FIT Victory Golf Polo,Perfect Fitness Perfect Rip Deck)), (62924,(O'Brien Men's Neoprene Life Vest,Perfect Fitness Perfect Rip Deck))
```

Now, we will print results, each output in one line and then exit the Spark shell.

```
- scala> // We print our results, 1 per line, and exit the Spark shell

scala> println(mostCommon.deep.mkString("\n"))
(290678,(Perfect Fitness Perfect Rip Deck,Perfect Fitness Perfect Rip Deck))
(246633,(Nike Men's Dri-FIT Victory Golf Polo,Nike Men's Dri-FIT Victory Golf Polo))
(225067,(O'Brien Men's Neoprene Life Vest,O'Brien Men's Neoprene Life Vest))
(140254,(Nike Men's Free 5.0+ Running Shoe,Nike Men's Free 5.0+ Running Shoe))
(119904,(Under Armour Girls' Toddler Spine Surge Runni,Under Armour Girls' Toddler Spine Surge Runni))
(67876,(Nike Men's Dri-FIT Victory Golf Polo,Perfect Fitness Perfect Rip Deck))
(62924,(O'Brien Men's Neoprene Life Vest,Perfect Fitness Perfect Rip Deck))
(54399,(Nike Men's Dri-FIT Victory Golf Polo,O'Brien Men's Neoprene Life Vest))
(39656,(Nike Men's Free 5.0+ Running Shoe,Perfect Fitness Perfect Rip Deck))
(35092,(Perfect Fitness Perfect Rip Deck,Under Armour Girls' Toddler Spine Surge Runni))

scala> exit
warning: there were 1 deprecation warning(s); re-run with -deprecation for details
Jun 1, 2019 9:44:18 PM INFO: parquet.hadoop.ParquetInputFormat: Total input paths to process : 2
Jun 1, 2019 9:44:18 PM INFO: parquet.hadoop.ParquetInputFormat: Total input paths to process : 2
[clooudera@quickstart ~]$
```

Problem 3:

Answer these questions:

A. What is the 5th most revenue generating product?

The screenshot shows the Impala Query Editor interface. At the top, there are tabs for 'Add a name...' and 'Add a description...'. Below the tabs, the execution time is shown as 15.57s, and there are icons for default, text, and other settings. The main area contains a SQL query:

```
1 select p.product_id, p.product_name, r.revenue
2 from products p inner join
3 (select oi.order_item_product_id, sum(cast(oi.order_item_subtotal as float)) as revenue
4 from order_items oi inner join orders o
5 on oi.order_item_order_id = o.order_id
6 where o.order_status <> 'CANCELED'
7 and o.order_status <> 'SUSPECTED_FRAUD'
8 group by order_item_product_id) r
9 on p.product_id = r.order_item_product_id
10 order by r.revenue desc
11 limit 10;
```

Below the query, the results are displayed in a table with three columns: product_id, product_name, and revenue. The results are ordered by revenue in descending order. The 5th row, which corresponds to the Nike Men's Dri-FIT Victory Golf Polo, is highlighted with a red border.

	product_id	product_name	revenue
1	1004	Field & Stream Sportsman 16 Gun Fire Safe	26550673.129272461
2	365	Perfect Fitness Perfect Rip Deck	16935177.47315979
3	957	Diamondback Women's Serene Classic Comfort Bi	15787348.018188477
4	191	Nike Men's Free 5.0+ Running Shoe	14030196.826934814
5	502	Nike Men's Dri-FIT Victory Golf Polo	12046400
6	1073	Pelican Sunstream 100 Kayak	11871406.726074219
7	1014	O'Brien Men's Neoprene Life Vest	11062173.258972168
8	403	Nike Men's CJ Elite 2 TD Football Cleat	11055909.94720459
9	627	Under Armour Girls' Toddler Spine Surge Runni	4859584.881149292
10	565	adidas Youth Germany Black/Red Away Match Soc	253960

After running the above query in Impala Query Editor, we got the top 10 most revenue generating products. From the output, we can see that **5th most Revenue Generating Product is Nike Men's Dri-Fit Victory Golf Polo.**

B. How much revenue does the Nike men's dry fit polo earn?

From the above output screenshot, we can see that **Nike Men's Dry-Fit Polo earns the revenue of \$12046400**

C. There is one product that did not show up in the previous result. It seems to be viewed a lot, but never purchased. Why?

The image shows two screenshots of Big Data query results. The top screenshot displays a table with columns: product_id, product_name, and revenue. The bottom screenshot displays a table with columns: count(*) and url.

Top Screenshot (Query Results):

product_id	product_name	revenue
1004	Field & Stream Sportsman 16 Gun Fire Safe	26550673.129272461
365	Perfect Fitness Perfect Rip Deck	16935177.47315979
957	Diamondback Women's Serene Classic Comfort Bi	15787348.018188477
191	Nike Men's Free 5.0+ Running Shoe	14030196.826934814
502	Nike Meris Dri-FIT Victory Golf Polo	12046400
1073	Pelican Sunstream 100 Kayak	11871406.726074219
1014	O'Brien Meris Neoprene Life Vest	11062173.258972168
403	Nike Men's CJ Elite 2 TD Football Cleat	11055909.94720459
627	Under Armour Girls' Toddler Spine Surge Runni	4859584.881149292
565	adidas Youth Germany Black/Red Away Match Soc	253960

Bottom Screenshot (Query Results):

count(*)	url
1926	/department/apparel/category/cleats/product/Perfect%20Fitness%20Perfect%20Rip%20Deck
1793	/department/apparel/category/featured%20shops/product/adidas%20Kids%20RG%20III%20Mid%20Football%20Cleat
1780	/department/golf/category/women's%20apparel/product/Nike%20Men's%20Dri-FIT%20Victory%20Golf%20Polo
1757	/department/apparel/category/men's%20footwear/product/Nike%20Men's%20CJ%20Elite%20TD%20Football%20Cleat
1104	/department/fan%20shop/category/water%20sports/product/Pelican%20unstream%20100%20Kayak
1084	/department/fan%20shop/category/indoor%20outdoor%20games/product/O'Brien%20Mens%20Neoprene%20Life%20Vest
1059	/department/fan%20shop/category/camping%20&%20hiking/product/Diamondback%20Women's%20Serene%20Classic%20Comfort%20Bi
1028	/department/fan%20shop/category/fishing/product/Field%20%20Stream%20Sportsman%2016%20Gun%20Fire%20Safe
1004	/department/footwear/category/cardio%20equipment/product/Nike%20Men's%20Free%205.0%20Running%20Shoe
939	/department/footwear/category/fitness%20accessories/product/Under%20Armour%20Hustle%20Storm%20Medium%20Duffle%20Bag

Missing Value ?

After carefully observing the results, we found that the list of most sold products is displayed from the previous tutorial steps, but there is one product which was missing in the previous list and that product is **Adidas Kids III Mid Football Cleat**. In this example of DataCo from

Aakash Sarap

Northeastern University Assignment 02

ALY6110 Data Management and Big Data

<https://northeastern.blackboard.com>

Tutorial Exercise 1, the reason for that missing product could be assumed as, on the view page where a lot of visitors stopped and checked that product but because of a typo in the price of that item. And after the typo was fixed and the displayed price was corrected, the sales for that product increased rapidly.

References

Cloudera. (2019). *The "Getting Started With Hadoop" tutorial, setup / Cloudera*. [online] Available at <https://www.cloudera.com/developers/get-started-with-hadoop-tutorial/setup.html> [Accessed 3 Jun. 2019].