KIRK DAHL
McKesson **Assignment 12** Deep Azure

Handed out: 01/13/2018 Due by 11:59 PM, midnight (CST) on Saturday, 01/20/2018

Problem 01. Create your own Hadoop cluster using HDInsight service. Demonstrate that you can establish an ssh session with the master node of that cluster. Tell us where is the `hadoop` executable by issuing command $ which Hadoop. Tell us which version of Python and Java are installed. Demonstrate that you have wiped out all components of that cluster.
(25%)

Create resource group

```
Kirks-MacBook:~ el5vgxz$ az group create -g rg-kirkdahl --location eastus
```

Create Storage account

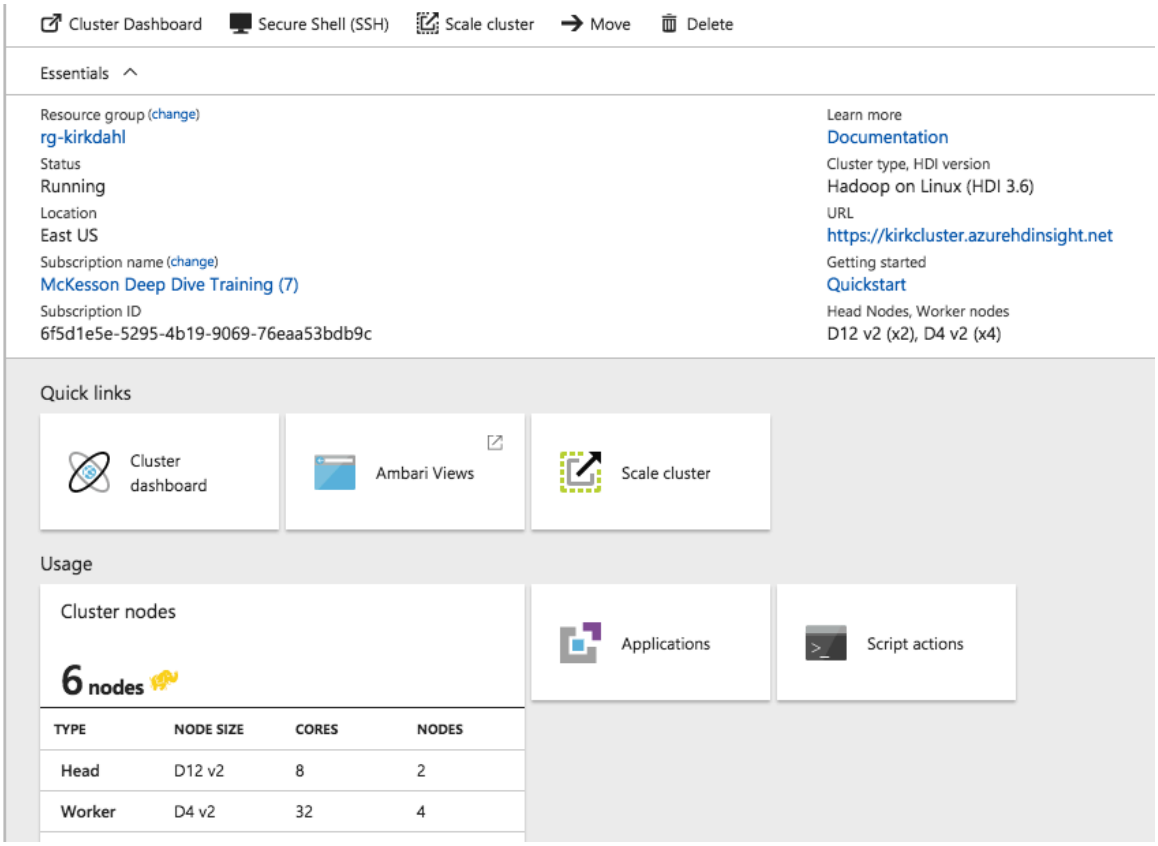
```
Kirks-MacBook:~ el5vgxz$ az storage account create -g rg-kirkdahl --sku Standard_LRS
-l eastus --kind Storage -n sakirkdahl
```

Retrieve storage keys

```
Kirks-MacBook:~ el5vgxz$ az storage account keys list -g rg-kirkdahl -n sakirkdahl
[
  {
    "keyName": "key1",
    "permissions": "Full",
    "value":
"wZpGmhiLbOYv8ZG2/4SYSxwCh4WknQjcw9XY9/+r7/AQ8GiXJld/ntqB91m0wpIftJD+uC+J7LDVuOSBKHl
qnw=="
  },
  {
    "keyName": "key2",
    "permissions": "Full",
    "value":
"B+bMouuLBYfxv7zZHeOD86pBsPDvL5LQUfnKSQ4rrU0aufobHn//mfizIzmyE2Qq2+tIaWJcg5QFZFDcoB/
I3Q=="
  }
]
```

As I have Azure CLI v2 – HDinsight is not supported in the CLI. I will use Portal.

CREATE HADOOP CLUSTER



SSH INTO CLUSTER

```
Kirks-MacBook:~ el5vgxz$ ssh sshuser@kirkcluster-ssh.azurehdinsight.net
Authorized uses only. All activity may be monitored and reported.
sshuser@kirkcluster-ssh.azurehdinsight.net's password:
```

LOCATION OF HADOOP

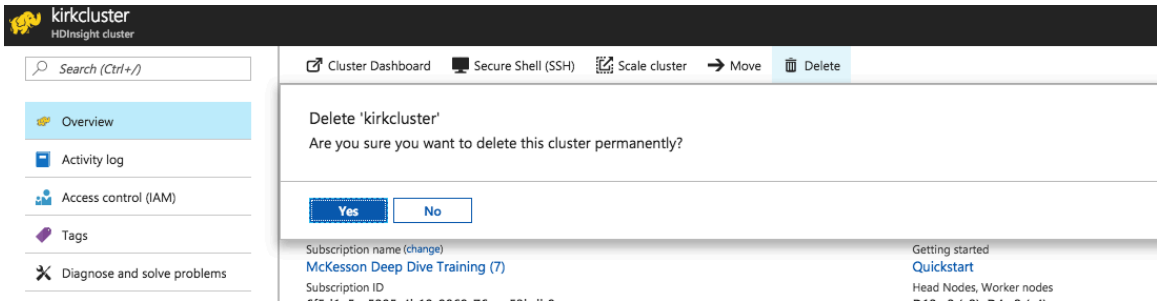
```
sshuser@hn0-kirkcl:~$ which hadoop
/usr/bin/hadoop
sshuser@hn0-kirkcl:~$
```

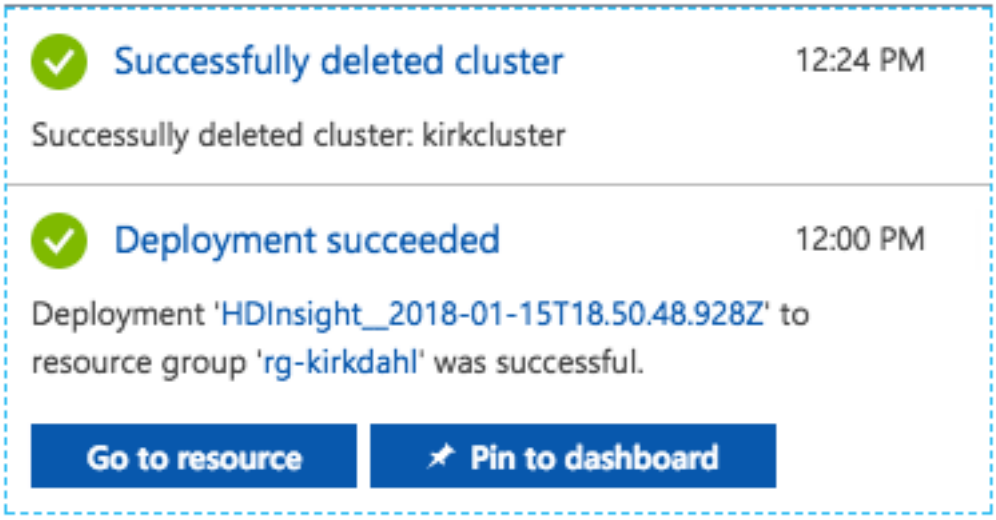
VERSIONS OF PYTHON AND JAVA

```
sshuser@hn0-kirkcl:~$ python -V
Python 2.7.12
sshuser@hn0-kirkcl:~$

sshuser@hn0-kirkcl:~$ java -version
openjdk version "1.8.0_151"
OpenJDK Runtime Environment (build 1.8.0_151-8u151-b12-0ubuntu0.16.04.2-b12)
OpenJDK 64-Bit Server VM (build 25.151-b12, mixed mode)
sshuser@hn0-kirkcl:~$
```

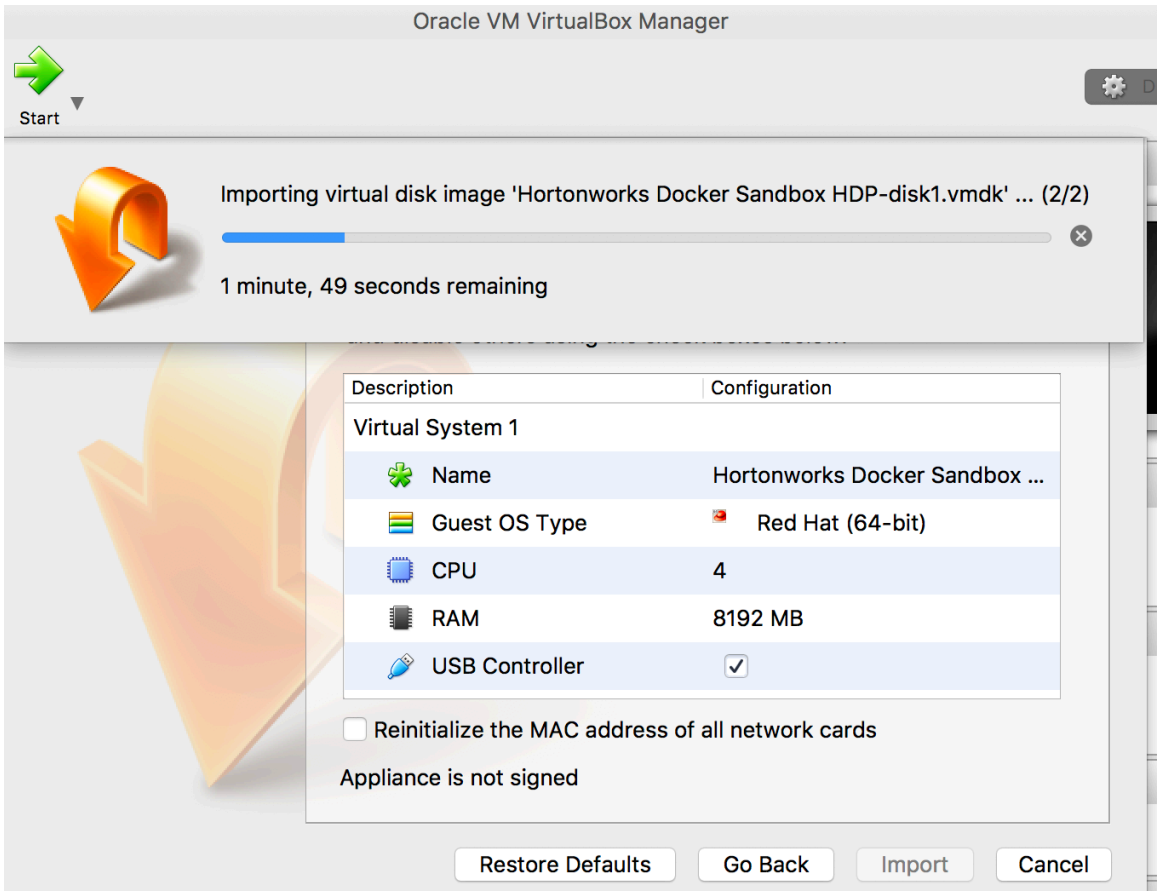
DELETE CLUSTER (will re-use resource group next problem instead of deleting)





Problem 02 Download and import Hortonworks HDP 2.6.3 sandbox VM. **Create a new** Linux user centos. Create HDFS home directory for that user. Move attached 4300-0.txt file to a subdirectory in centos' HDFS home directory. Run Hadoop MapReduce grep program on that file. Copy the result of that analysis to your local file system and examine top 10 and bottom 10 lines.
(25%)

IMPORT SANDBOX



Connected to machine and changed root password

```
Kirks-MacBook:Downloads el5vgxz$ ssh root@localhost -p 2222
root@localhost's password:
You are required to change your password immediately (root enforced)
Changing password for root.
```

SCP the 4300.txt file to machine

```
Kirks-MacBook:Downloads el5vgxz$ scp -P 2222 4300-0.txt root@localhost:/tmp
root@localhost's password:
4300-
0.txt
                  100% 4583KB  42.6MB/s   00:00
Kirks-MacBook:Downloads el5vgxz$
```

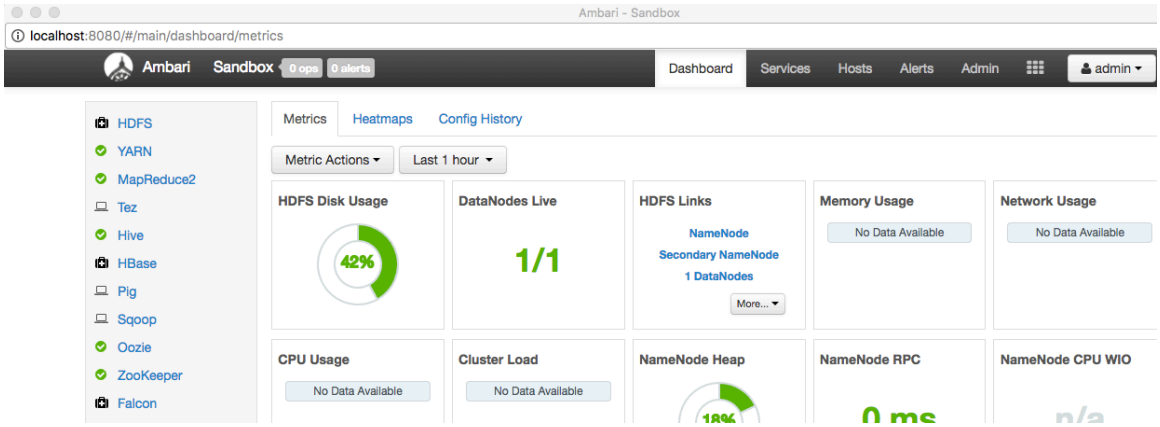
```
[root@sandbox-hdp ~]# ambari-admin-password-reset
Please set the password for admin:
Please retype the password for admin:

The admin password has been set.
Restarting ambari-server to make the password change effective...

Using python  /usr/bin/python
Restarting ambari-server
Waiting for server stop...
```

```
Ambari Server stopped
Ambari Server running with administrator privileges.
Organizing resource files at /var/lib/ambari-server/resources...
Ambari database consistency check started...
Server PID at: /var/run/ambari-server/ambari-server.pid
Server out at: /var/log/ambari-server/ambari-server.out
Server log at: /var/log/ambari-server/ambari-server.log
Waiting for server start.....................................
Server started listening on 8080
```

https://localhost:8888 with admin and new password



MAKE CENTOS USER HOME DIRECTORY

```
[root@sandbox-hdp ~]# sudo -u hdfs hdfs dfs -mkdir /user/centos
[root@sandbox-hdp ~]#
```

Copy 4300-0.txt into input

```
[centos@sandbox-hdp ~]$ hadoop fs -mkdir input
[centos@sandbox-hdp ~]$ hadoop fs -put 4300-0.txt input
[centos@sandbox-hdp ~]$ hadoop fs -ls input
Found 1 items
-rw-r--r--   1 centos mapred    4692498 2018-01-15 21:31 input/4300-0.txt
[centos@sandbox-hdp ~]$
```

RUNNING MAPREDUCE GREP JOB

```
[centos@sandbox-hdp ~]$ hadoop jar /usr/hdp/2.6.3.0-235/hadoop-mapreduce/hadoop-mapreduce-examples.jar grep input/4300-0.txt ulysses_freq '\w+'
18/01/15 21:33:52 INFO client.RMProxy: Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.17.0.2:8032
18/01/15 21:33:52 INFO client.AHSProxy: Connecting to Application History server at sandbox-hdp.hortonworks.com/172.17.0.2:10200
18/01/15 21:33:53 INFO input.FileInputFormat: Total input paths to process : 1
18/01/15 21:33:53 INFO mapreduce.JobSubmitter: number of splits:1
18/01/15 21:33:53 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1516050262624_0001
18/01/15 21:33:54 INFO impl.YarnClientImpl: Submitted application application_1516050262624_0001
18/01/15 21:33:54 INFO mapreduce.Job: The url to track the job: http://sandbox-hdp.hortonworks.com:8088/proxy/application_1516050262624_0001/
18/01/15 21:33:54 INFO mapreduce.Job: Running job: job_1516050262624_0001
```

TOP 10 and BOTTOM 10 LINES OF OUTPUT

```
18/01/15 21:33:52 INFO client.RMProxy: Connecting to ResourceManager at sandbox-
hdp.hortonworks.com/172.17.0.2:8032
18/01/15 21:33:52 INFO client.AHSProxy: Connecting to Application History server at
sandbox-hdp.hortonworks.com/172.17.0.2:10200
18/01/15 21:33:53 INFO input.FileInputFormat: Total input paths to process : 1
18/01/15 21:33:53 INFO mapreduce.JobSubmitter: number of splits:1
18/01/15 21:33:53 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1516050262624_0001
18/01/15 21:33:54 INFO impl.YarnClientImpl: Submitted application
application_1516050262624_0001
18/01/15 21:33:54 INFO mapreduce.Job: The url to track the job: http://sandbox-
hdp.hortonworks.com:8088/proxy/application_1516050262624_0001/
18/01/15 21:33:54 INFO mapreduce.Job: Running job: job_1516050262624_0001
18/01/15 21:34:11 INFO mapreduce.Job: Job job_1516050262624_0001 running in uber
mode : false
18/01/15 21:34:11 INFO mapreduce.Job:  map 0% reduce 0%
18/01/15 21:34:20 INFO mapreduce.Job:  map 100% reduce 0%
18/01/15 21:34:27 INFO mapreduce.Job:  map 100% reduce 100%
18/01/15 21:34:28 INFO mapreduce.Job: Job job_1516050262624_0001 completed
successfully
18/01/15 21:34:28 INFO mapreduce.Job: Counters: 49


        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=924094
        File Output Format Counters
                Bytes Written=393354
```

Problem 03. Use Spark 1.6.3 to find out the number of lines in 4300-0.txt which contain word “future”. Repeat the exercise with Spark 2.2.
(25%)

SPARKS1

```
scala> val file = sc.textFile("file:///home/centos/4300-0.txt")
18/01/16 23:01:48 INFO storage.MemoryStore: Block broadcast_2 stored as values in memory
(estimated size 356.0 KB, free 510.3 MB)
18/01/16 23:01:48 INFO storage.MemoryStore: Block broadcast_2_piece0 stored as bytes in memory
(estimated size 30.9 KB, free 510.2 MB)
18/01/16 23:01:48 INFO storage.BlockManagerInfo: Added broadcast_2_piece0 in memory on
localhost:44995 (size: 30.9 KB, free: 511.0 MB)
18/01/16 23:01:48 INFO spark.SparkContext: Created broadcast 2 from textFile at <console>:27
file: org.apache.spark.rdd.RDD[String] = file:///home/centos/4300-0.txt MapPartitionsRDD[7] at
textFile at <console>:27

scala> val counts = file.flatMap(line => line.split(" ")).map(word => (word,
1)).reduceByKey(_+_)
```

```
18/01/16 23:01:51 INFO mapred.FileInputFormat: Total input paths to process : 1
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[10] at reduceByKey at
<console>:29

scala>
```

Save output

```
scala> counts.saveAsTextFile("/tmp/wordcount")
```

Hadoop grep for future (using regex)

```
hadoop org.apache.hadoop.examples.Grep /tmp/wordcount /tmp/output  .*future.*
```

The output contains 65 versions of “future”

```
[centos@sandbox-hdp tmp]$ hadoop fs -cat /tmp/output/part-r-00000
1       (futures,3)
1       (future:,3)
1       (future.,9)
1       (future,47)
1       (future,,3)
```

SPARKS2 – 65 occurences

```
>>> dset = spark.read.text("file:///home/centos/4300-0.txt")
>>> future = dset.filter(dset.value.contains('future'))
>>> future.count()
65
>>>
```

Problem 04. Use hive command line client to demonstrate that you can import bible word frequencies and Shakespeare word frequencies into respective tables bible and shakespeare. Convince yourself that you can run a select statement joining those two tables.
(25%)

FIRST CREATE THE HADOOP FILES

```
[centos@sandbox-hdp ~]$ hadoop jar /usr/hdp/2.6.3.0-235/hadoop-mapreduce/hadoop-mapreduce-
examples.jar grep input_bible bible_freq '\w+'
18/01/16 22:26:07 INFO client.RMProxy: Connecting to ResourceManager at sandbox-
hdp.hortonworks.com/172.17.0.2:8032
18/01/16 22:26:07 INFO client.AHSProxy: Connecting to Application History server at sandbox-
hdp.hortonworks.com/172.17.0.2:10200
18/01/16 22:26:08 INFO input.FileInputFormat: Total input paths to process : 1
18/01/16 22:26:08 INFO mapreduce.JobSubmitter: number of splits:1
```

```
[centos@sandbox-hdp ~]$ hadoop jar /usr/hdp/2.6.3.0-235/hadoop-mapreduce/hadoop-mapreduce-
examples.jar grep input_shake shake_freq '\w+'
18/01/16 22:28:47 INFO client.RMProxy: Connecting to ResourceManager at sandbox-
hdp.hortonworks.com/172.17.0.2:8032
18/01/16 22:28:47 INFO client.AHSProxy: Connecting to Application History server at sandbox-
hdp.hortonworks.com/172.17.0.2:10200
18/01/16 22:28:48 INFO input.FileInputFormat: Total input paths to process : 1
```

LOAD DATA INTO TABLES

Bible

```
hive> LOAD DATA INPATH "/user/centos/bible_freq" INTO TABLE
    > Bible;
Loading data to table default.bible
Table default.bible stats: [numFiles=2, numRows=0, totalSize=294816, rawDataSize=0]
OK
Time taken: 1.972 seconds
```

Shakespeare

```
hive> LOAD DATA INPATH "/user/centos/shake_freq" INTO TABLE
    > shakespeare;
Loading data to table default.shakespeare
Table default.shakespeare stats: [numFiles=2, numRows=0, totalSize=598758, rawDataSize=0]
OK
Time taken: 1.143 seconds
```

CREATE MERGED TABLE

```
hive> CREATE TABLE merged
    > (word STRING, shake_f INT, bible_f INT);
OK
Time taken: 0.401 seconds
```

INSERT “merged” DATA INTO TABLE AND PERFORM SELECT

```
hive> INSERT OVERWRITE TABLE merged
    > SELECT s.word, s.freq, k.freq FROM
    > shakespeare s JOIN bible k ON
    > (s.word = k.word)
    > WHERE s.freq >= 1 AND k.freq >=1;
Query ID = centos_20180116223419_a0910c5e-bd45-4128-b55f-3dfef12d122d
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1516123096941_0016)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Map 2 ..........   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 9.14 s
--------------------------------------------------------------------------------
Loading data to table default.merged
Table default.merged stats: [numFiles=1, numRows=31020, totalSize=393820, rawDataSize=362800]
OK
Time taken: 13.031 seconds
hive> select * from merged limit 10;
OK
the     25578   62394
the     25578   62394
I       23027   8854
I       23027   8854
and     19654   38985
and     19654   38985
```

```
to      17462   13526
to      17462   13526
of      16444   34654
of      16444   34654
Time taken: 0.152 seconds, Fetched: 10 row(s)
```

SUBMISSION INSTRUCTIONS:

Your main submission should be a MS Word or PDF document containing descriptions of your action while configuring Azure services. If your MS Word document is larger than 1 MB, save it as a MINIMIZED PDF. Please be merciful and capture small JPGs. Describe the purpose of every action and the significance of the results. Start with the text of this homework assignment as the template. Please add the entire text of your JAVA, C# or Python programs to the end of your MS Word/PDF document. Please write your solution as if you are writing a tutorial for your colleagues. Please make your text readable. Make sure that your fonts, especially in captured images are not unreadable. Please do not provide ZIP or RAR or any other archives. Canvas cannot open those archives and they turn into a nuisance for us.