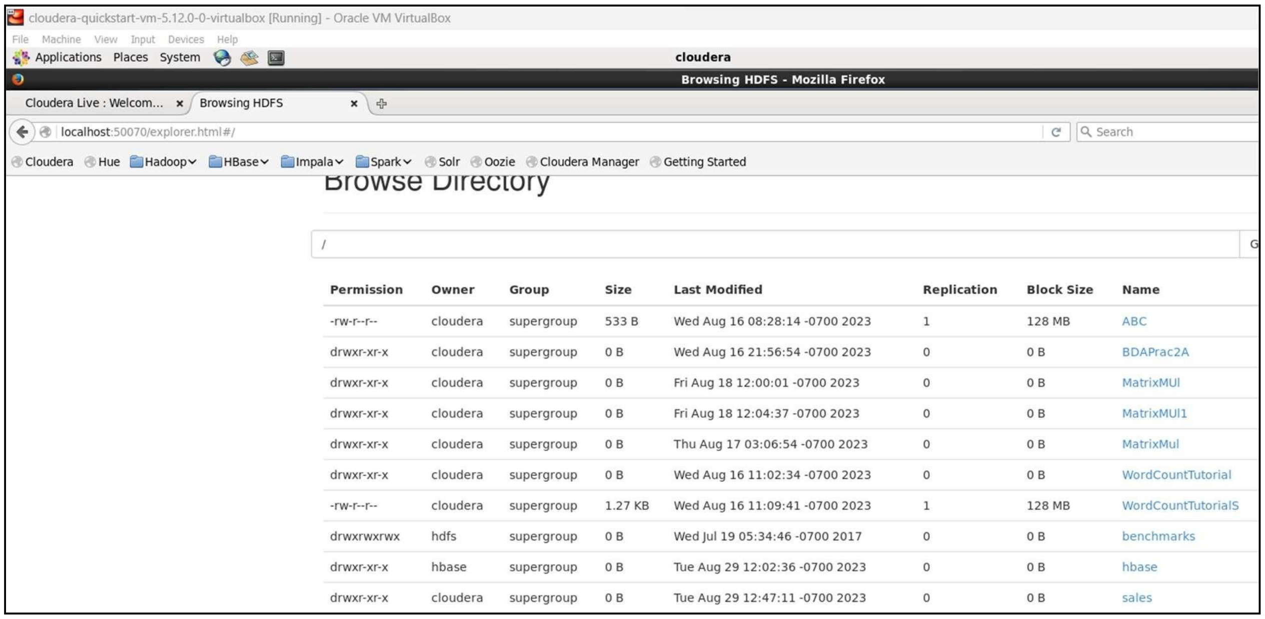
Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

Artificial Intelligence and Data Science Department
Big Data Analytics/Odd Sem 2023-23/Experiment 3

Importing tables from RDMS to HDFS using Sqoop:

```
[cloudera@quickstart ~]$ sqoop import --connect jdbc:mysql://localhost/sales --username=root --
password="cloudera" --table=sales1 --target-dir=/sales/sales -incremental append --check-column
 month_number --fields-terminated-by='\t';
```


cloudera-quickstart-vm-5.12.0-0-virtualbox [Running] - Oracle VM VirtualBox

Browsing HDFS - Mozilla Firefox

localhost:50070/explorer.html#/

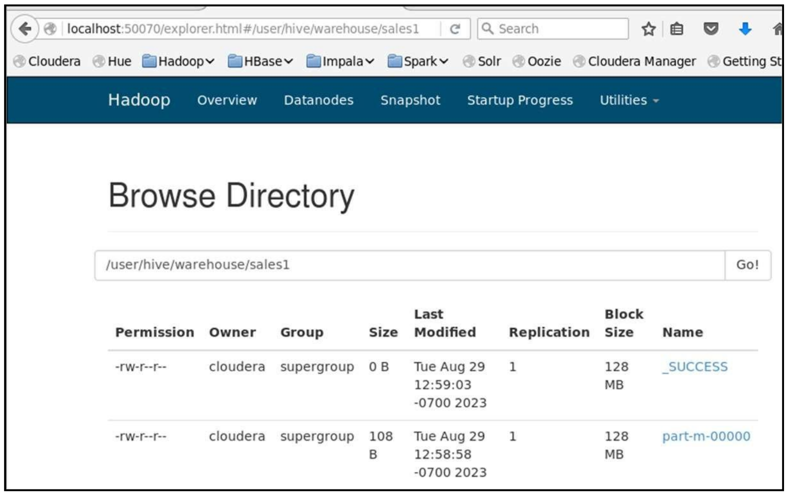
Browse Directory

/

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	cloudera	supergroup	533 B	Wed Aug 16 08:28:14 -0700 2023	1	128 MB	ABC
drwxr-xr-x	cloudera	supergroup	0 B	Wed Aug 16 21:56:54 -0700 2023	0	0 B	BDAPrac2A
drwxr-xr-x	cloudera	supergroup	0 B	Fri Aug 18 12:00:01 -0700 2023	0	0 B	MatrixMUl
drwxr-xr-x	cloudera	supergroup	0 B	Fri Aug 18 12:04:37 -0700 2023	0	0 B	MatrixMUl1
drwxr-xr-x	cloudera	supergroup	0 B	Thu Aug 17 03:06:54 -0700 2023	0	0 B	MatrixMul
drwxr-xr-x	cloudera	supergroup	0 B	Wed Aug 16 11:02:34 -0700 2023	0	0 B	WordCountTutorial
-rw-r--r--	cloudera	supergroup	1.27 KB	Wed Aug 16 11:09:41 -0700 2023	1	128 MB	WordCountTutorialS
drwxrwxrwx	hdfs	supergroup	0 B	Wed Jul 19 05:34:46 -0700 2017	0	0 B	benchmarks
drwxr-xr-x	hbase	supergroup	0 B	Tue Aug 29 12:02:36 -0700 2023	0	0 B	hbase
drwxr-xr-x	cloudera	supergroup	0 B	Tue Aug 29 12:47:11 -0700 2023	0	0 B	sales

Importing Table From HDFS to HIVE:

```
[cloudera@quickstart ~]$ sqoop import-all-tables --connect jdbc:mysql://localhost/sales --username
root --password "cloudera" --warehouse-dir /user/hive/warehouse
```

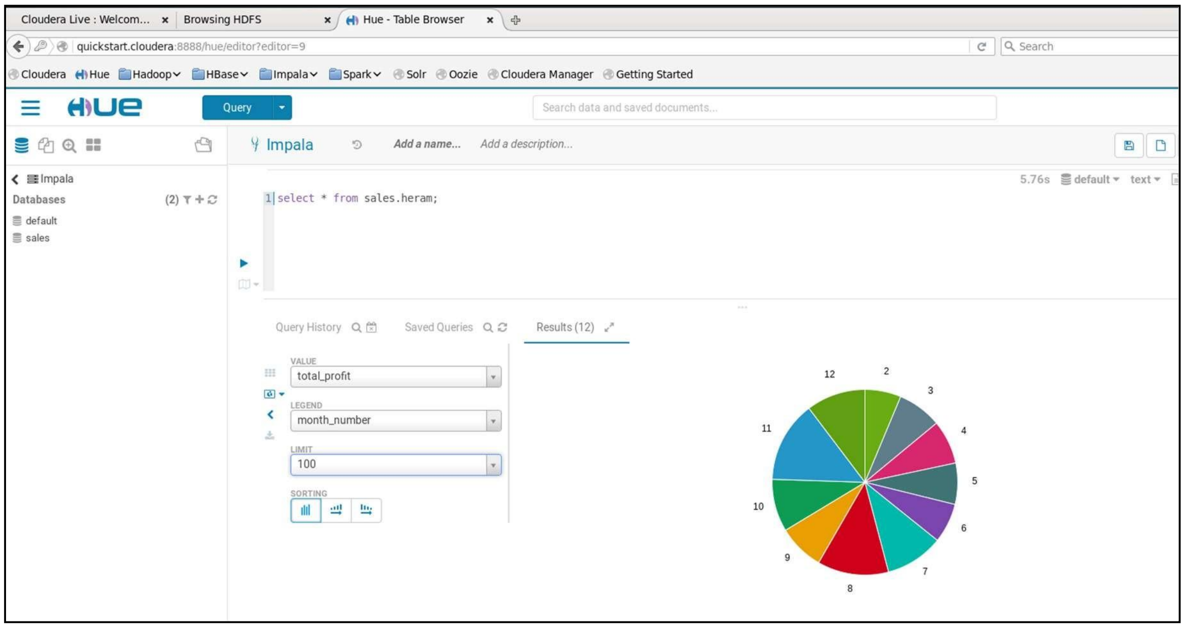

localhost:50070/explorer.html#/user/hive/warehouse/sales1

Browse Directory

/user/hive/warehouse/sales1

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	cloudera	supergroup	0 B	Tue Aug 29 12:59:03 -0700 2023	1	128 MB	_SUCCESS
-rw-r--r--	cloudera	supergroup	108 B	Tue Aug 29 12:58:58 -0700 2023	1	128 MB	part-m-00000

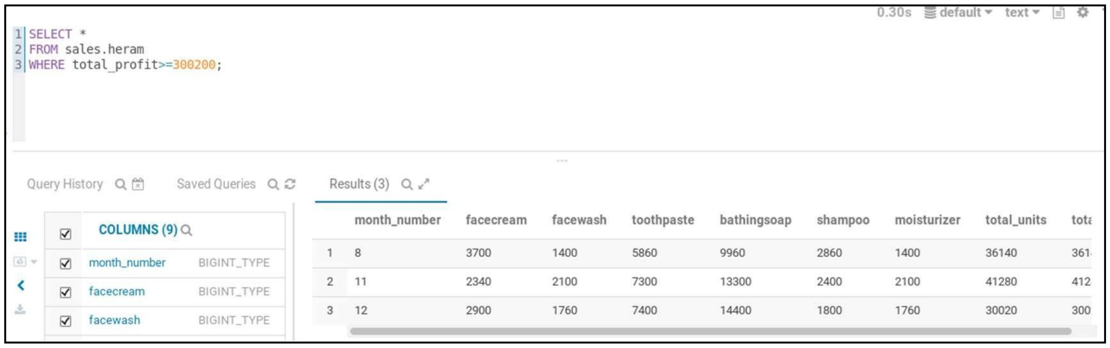
Going to Hue Editor, Importing table, Writing Query And Doing Visualization.



```
select * from sales.heram;
```

VALUE: total_profit
LEGEND: month_number
LIMIT: 100

Running Some Queries:

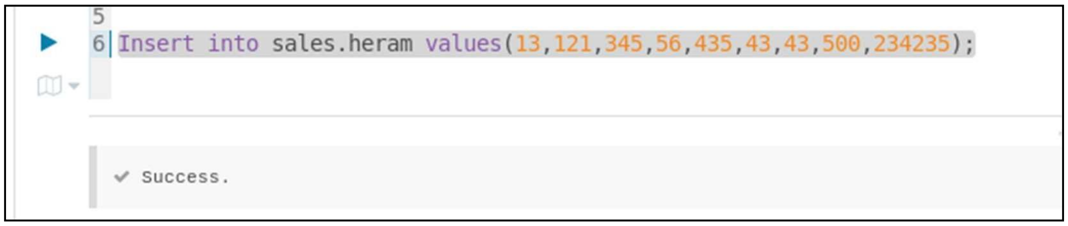


```
SELECT *
FROM sales.heram
WHERE total_profit>=300200;
```

COLUMNS (9)

	month_number	BIGINT_TYPE
✓	facecream	BIGINT_TYPE
✓	facewash	BIGINT_TYPE

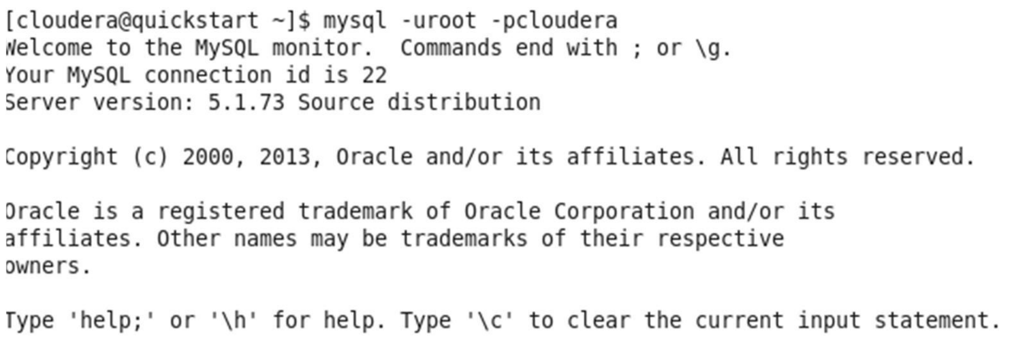
	month_number	facecream	facewash	toothpaste	bathingsoap	shampoo	moisturizer	total_units	tota
1	8	3700	1400	5860	9960	2860	1400	36140	361
2	11	2340	2100	7300	13300	2400	2100	41280	412
3	12	2900	1760	7400	14400	1800	1760	30020	300



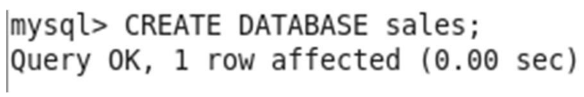
```
Insert into sales.heram values(13,121,345,56,435,43,43,500,234235);
```

```
✔ Success.
```

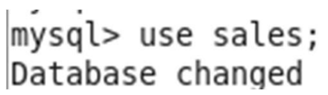
Output:



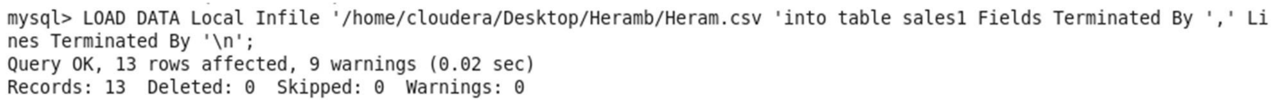
```
[cloudera@quickstart ~]$ mysql -uroot -pcloudera
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```



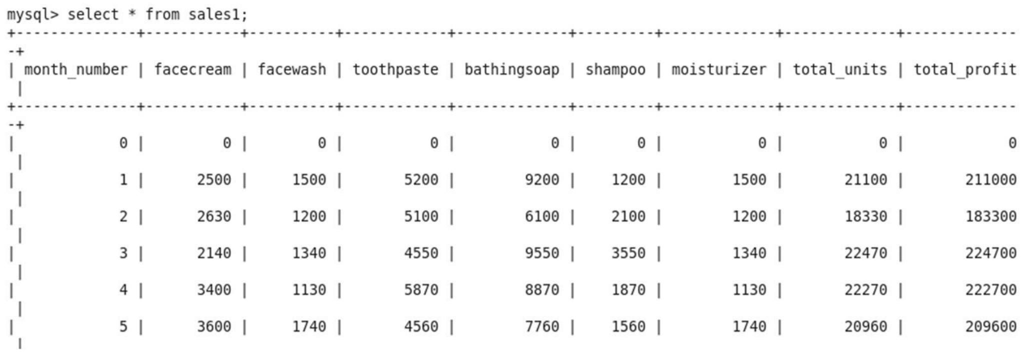
```
mysql> CREATE DATABASE sales;
Query OK, 1 row affected (0.00 sec)
```



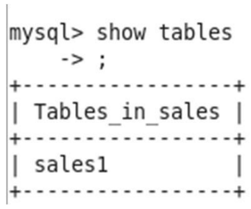
```
mysql> use sales;
Database changed
```



```
mysql> LOAD DATA Local Infile '/home/cloudera/Desktop/Heramb/Heram.csv 'into table sales1 Fields Terminated By ',' Li
nes Terminated By '\n';
Query OK, 13 rows affected, 9 warnings (0.02 sec)
Records: 13  Deleted: 0  Skipped: 0  Warnings: 0
```



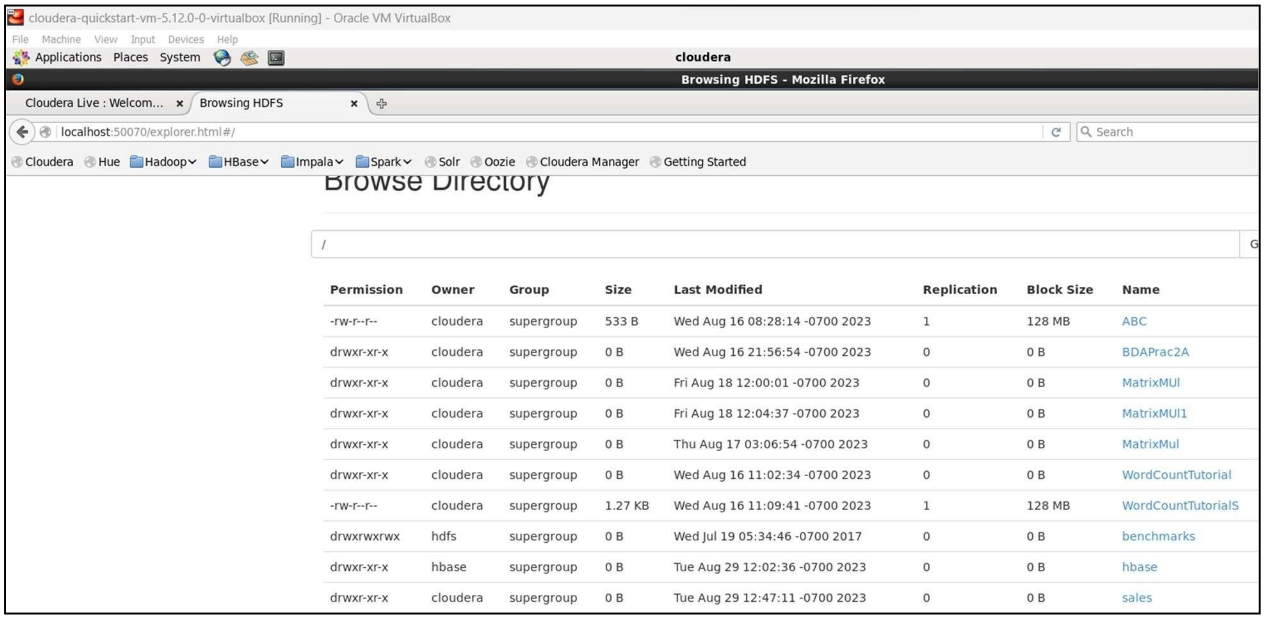
```
mysql> select * from sales1;
+--------------+-----------+----------+------------+-------------+---------+-------------+-------------+------------
-+
| month_number | facecream | facewash | toothpaste | bathingsoap | shampoo | moisturizer | total_units | total_profit
 |
+--------------+-----------+----------+------------+-------------+---------+-------------+-------------+------------
-+
|            0 |         0 |        0 |          0 |           0 |       0 |           0 |           0 |            0
 |
|            1 |      2500 |     1500 |       5200 |        9200 |    1200 |        1500 |       21100 |       211000
 |
|            2 |      2630 |     1200 |       5100 |        6100 |    2100 |        1200 |       18330 |       183300
 |
|            3 |      2140 |     1340 |       4550 |        9550 |    3550 |        1340 |       22470 |       224700
 |
|            4 |      3400 |     1130 |       5870 |        8870 |    1870 |        1130 |       22270 |       222700
 |
|            5 |      3600 |     1740 |       4560 |        7760 |    1560 |        1740 |       20960 |       209600
 |
```



```
mysql> show tables
    -> ;
+----------------+
| Tables_in_sales |
+----------------+
| sales1          |
+----------------+
```

Importing tables from RDMS to HDFS using Sqoop:

```
[cloudera@quickstart ~]$ sqoop import --connect jdbc:mysql://localhost/sales --username=root --
password="cloudera" --table=sales1 --target-dir=/sales/sales -incremental append --check-column
 month_number --fields-terminated-by='\t';
```



cloudera-quickstart-vm-5.12.0-0-virtualbox [Running] - Oracle VM VirtualBox

Browsing HDFS - Mozilla Firefox

localhost:50070/explorer.html#/

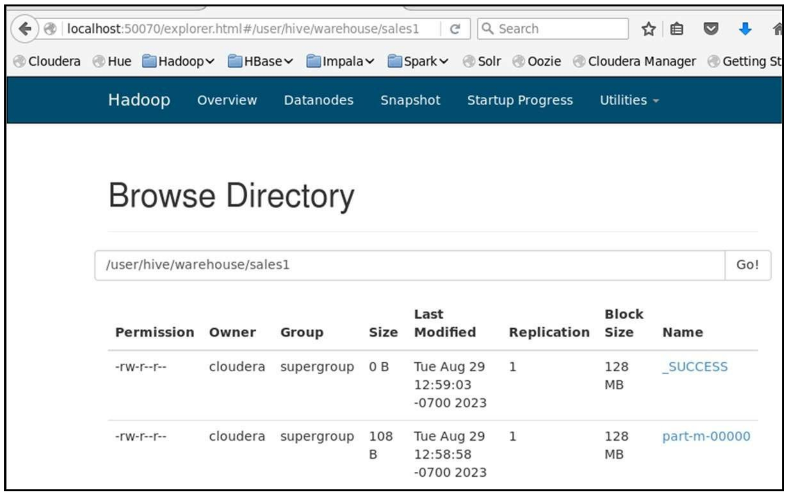
Browse Directory

/

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	cloudera	supergroup	533 B	Wed Aug 16 08:28:14 -0700 2023	1	128 MB	ABC
drwxr-xr-x	cloudera	supergroup	0 B	Wed Aug 16 21:56:54 -0700 2023	0	0 B	BDAPrac2A
drwxr-xr-x	cloudera	supergroup	0 B	Fri Aug 18 12:00:01 -0700 2023	0	0 B	MatrixMUl
drwxr-xr-x	cloudera	supergroup	0 B	Fri Aug 18 12:04:37 -0700 2023	0	0 B	MatrixMUl1
drwxr-xr-x	cloudera	supergroup	0 B	Thu Aug 17 03:06:54 -0700 2023	0	0 B	MatrixMul
drwxr-xr-x	cloudera	supergroup	0 B	Wed Aug 16 11:02:34 -0700 2023	0	0 B	WordCountTutorial
-rw-r--r--	cloudera	supergroup	1.27 KB	Wed Aug 16 11:09:41 -0700 2023	1	128 MB	WordCountTutorialS
drwxrwxrwx	hdfs	supergroup	0 B	Wed Jul 19 05:34:46 -0700 2017	0	0 B	benchmarks
drwxr-xr-x	hbase	supergroup	0 B	Tue Aug 29 12:02:36 -0700 2023	0	0 B	hbase
drwxr-xr-x	cloudera	supergroup	0 B	Tue Aug 29 12:47:11 -0700 2023	0	0 B	sales

Importing Table From HDFS to HIVE:

```
[cloudera@quickstart ~]$ sqoop import-all-tables --connect jdbc:mysql://localhost/sales --username
root --password "cloudera" --warehouse-dir /user/hive/warehouse
```



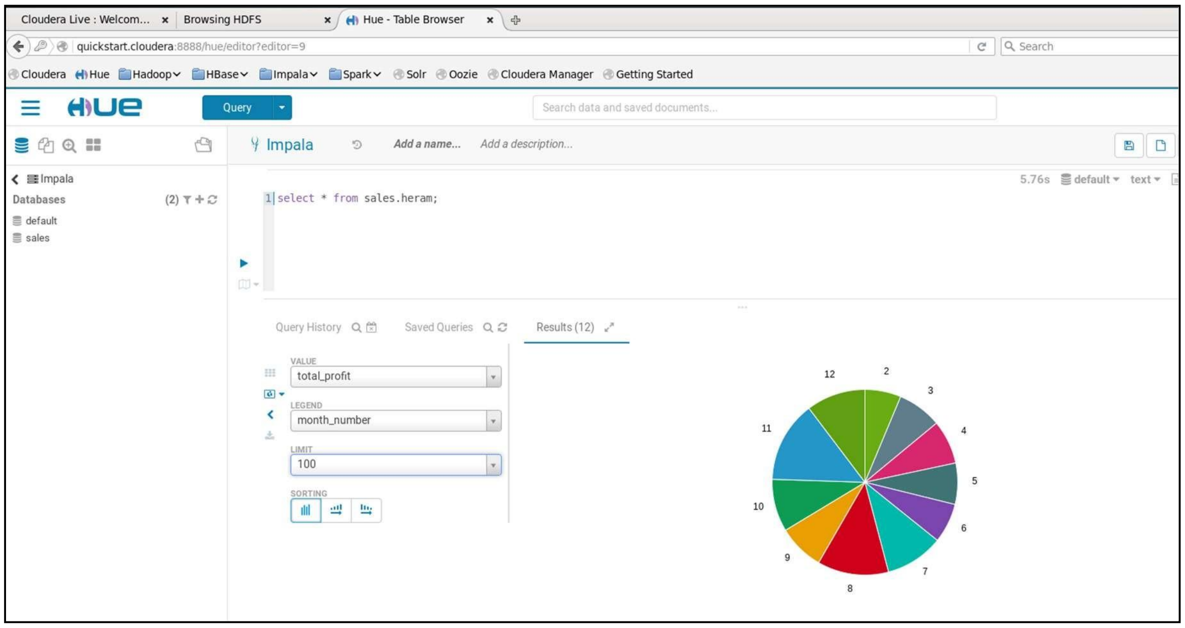
localhost:50070/explorer.html#/user/hive/warehouse/sales1

Browse Directory

/user/hive/warehouse/sales1

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	cloudera	supergroup	0 B	Tue Aug 29 12:59:03 -0700 2023	1	128 MB	_SUCCESS
-rw-r--r--	cloudera	supergroup	108 B	Tue Aug 29 12:58:58 -0700 2023	1	128 MB	part-m-00000

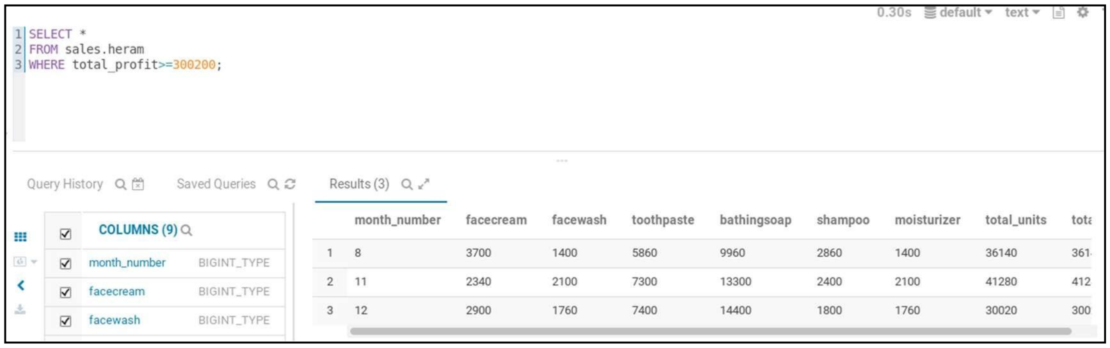
Going to Hue Editor, Importing table, Writing Query And Doing Visualization.


Cloudera Live : Welcom...
Browsing HDFS
Hue - Table Browser
quickstart.cloudera:8888/hue/editor?editor=9
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
Query
Impala
Databases (2)
default
sales
5.76s default text
Query History
Saved Queries
Results (12)
12
2
3
4
5
6
7
8
9
10
11

```
select * from sales.heram;
```

VALUE: total_profit
LEGEND: month_number
LIMIT: 100
SORTING

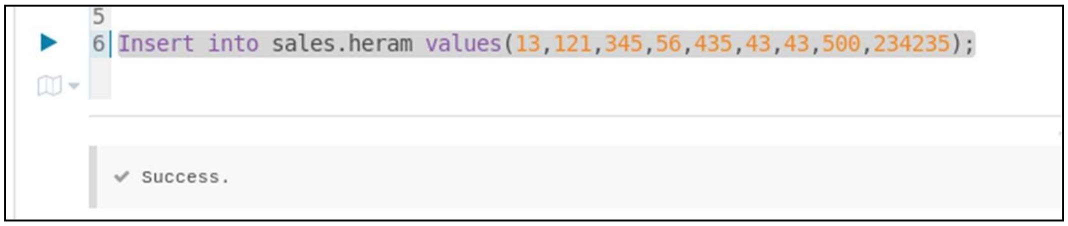
Running Some Queries:



```
SELECT *
FROM sales.heram
WHERE total_profit>=300200;
```

	COLUMNS (9)	
✓	month_number	BIGINT_TYPE
✓	facecream	BIGINT_TYPE
✓	facewash	BIGINT_TYPE

	month_number	facecream	facewash	toothpaste	bathingsoap	shampoo	moisturizer	total_units	tota
1	8	3700	1400	5860	9960	2860	1400	36140	361
2	11	2340	2100	7300	13300	2400	2100	41280	412
3	12	2900	1760	7400	14400	1800	1760	30020	300



```
Insert into sales.heram values(13,121,345,56,435,43,43,500,234235);
```

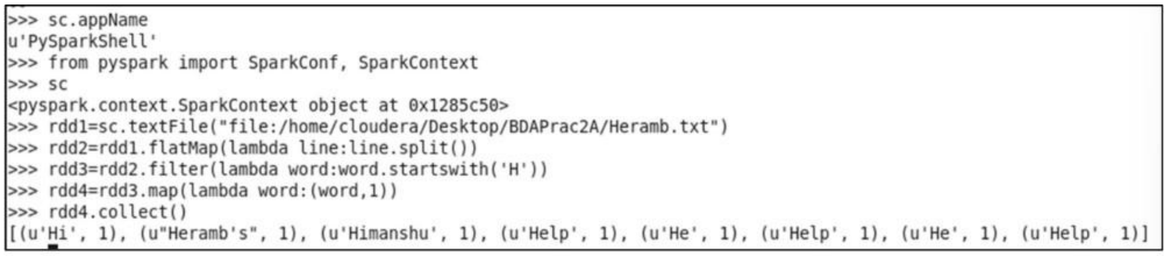
```
✔ Success.
```

Program:

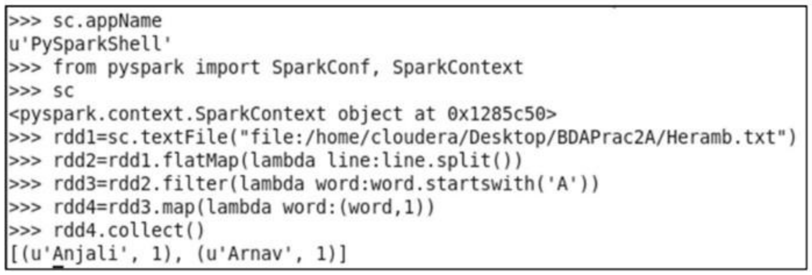
First Type 'pyspark' in the terminal then type the below commands.

```
>>> sc.appName

u'PySparkShell'

>>>from pyspark import SparkConf, SparkContext

>>> sc

<pyspark.context.SparkContext object at 0x2918c50>

>>> rdd1=sc.textFile("file:/home/cloudera/RT/data1.txt")

>>> rdd2=rdd1.flatMap(lambda line:line.split())

>>> rdd3=rdd2.filter(lambda word:word.startswith('h'))

>>> rdd4=rdd3.map(lambda word:(word,1))

>>> rdd4.collect
```

Output:



```
>>> sc.appName
u'PySparkShell'
>>> from pyspark import SparkConf, SparkContext
>>> sc
<pyspark.context.SparkContext object at 0x1285c50>
>>> rdd1=sc.textFile("file:/home/cloudera/Desktop/BDAPrac2A/Heramb.txt")
>>> rdd2=rdd1.flatMap(lambda line:line.split())
>>> rdd3=rdd2.filter(lambda word:word.startswith('H'))
>>> rdd4=rdd3.map(lambda word:(word,1))
>>> rdd4.collect()
[(u'Hi', 1), (u"Heramb's", 1), (u'Himanshu', 1), (u'Help', 1), (u'He', 1), (u'Help', 1), (u'He', 1), (u'Help', 1)]
```

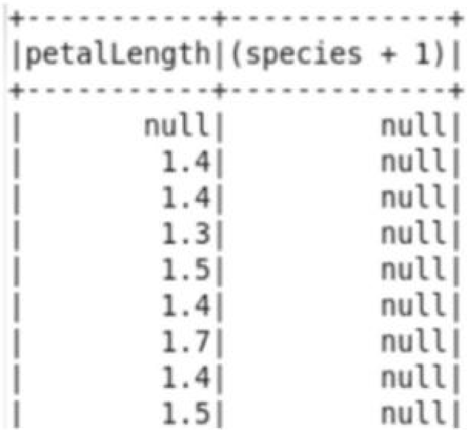


```
>>> sc.appName
u'PySparkShell'
>>> from pyspark import SparkConf, SparkContext
>>> sc
<pyspark.context.SparkContext object at 0x1285c50>
>>> rdd1=sc.textFile("file:/home/cloudera/Desktop/BDAPrac2A/Heramb.txt")
>>> rdd2=rdd1.flatMap(lambda line:line.split())
>>> rdd3=rdd2.filter(lambda word:word.startswith('A'))
>>> rdd4=rdd3.map(lambda word:(word,1))
>>> rdd4.collect()
[(u'Anjali', 1), (u'Arnav', 1)]
```

Program + Output RDD Programs

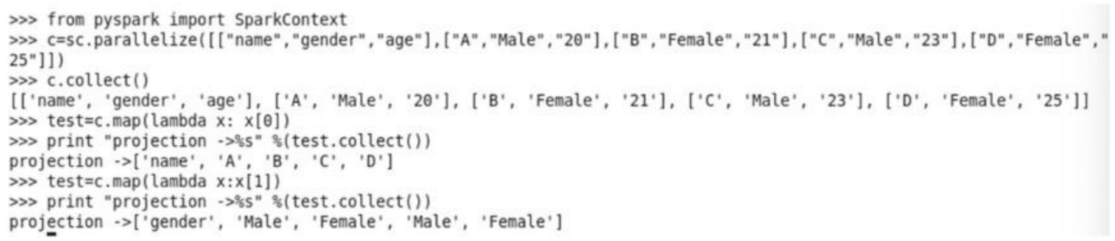
A. Selection

```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
df = sqlContext.read.json("/user/cloudera/iris.json")
df.show()
df.select("species").show()
df.select(df['petalLength'], df['species'] + 1).show()
```



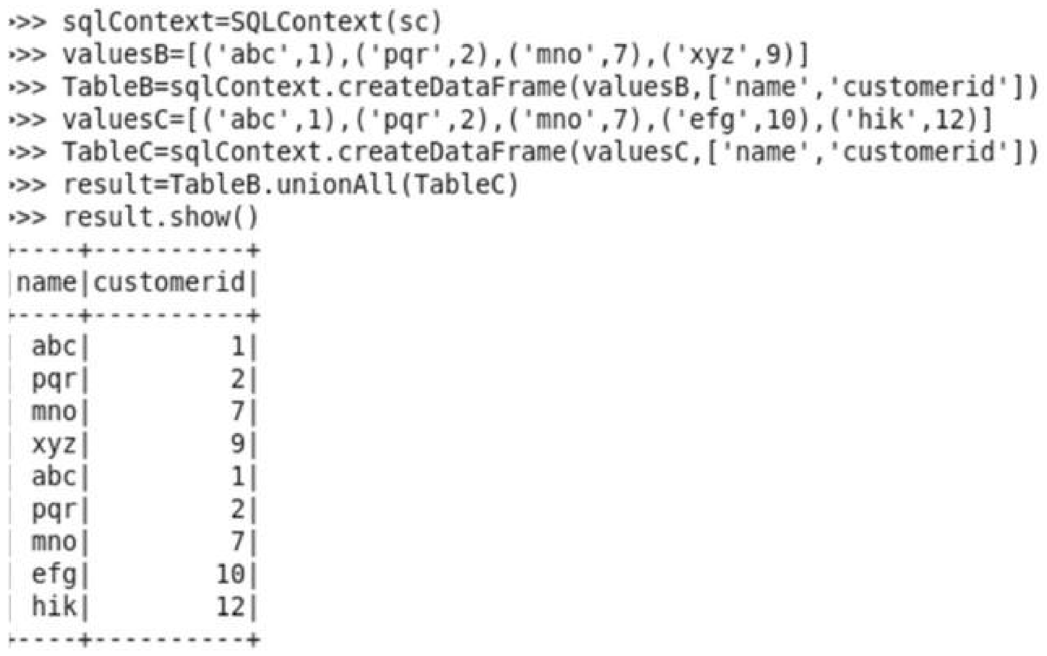
```
+-----------+-------------+
|petalLength|(species + 1)|
+-----------+-------------+
|       null|         null|
|        1.4|         null|
|        1.4|         null|
|        1.3|         null|
|        1.5|         null|
|        1.4|         null|
|        1.7|         null|
|        1.4|         null|
|        1.5|         null|
```

B. Projection



```
>>> from pyspark import SparkContext
>>> c=sc.parallelize([["name","gender","age"],["A","Male","20"],["B","Female","21"],["C","Male","23"],["D","Female","
25"]])
>>> c.collect()
[['name', 'gender', 'age'], ['A', 'Male', '20'], ['B', 'Female', '21'], ['C', 'Male', '23'], ['D', 'Female', '25']]
>>> test=c.map(lambda x: x[0])
>>> print "projection ->%s" %(test.collect())
projection ->['name', 'A', 'B', 'C', 'D']
>>> test=c.map(lambda x:x[1])
>>> print "projection ->%s" %(test.collect())
projection ->['gender', 'Male', 'Female', 'Male', 'Female']
```

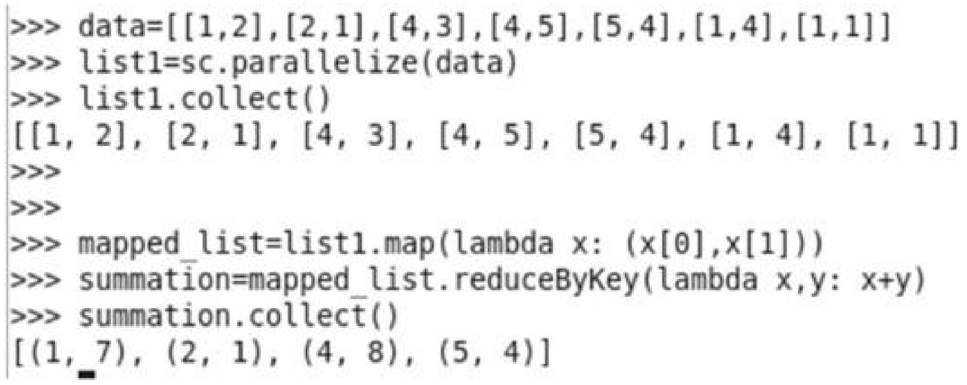
C. Union



```
>>> sqlContext=SQLContext(sc)
>>> valuesB=[('abc',1),('pqr',2),('mno',7),('xyz',9)]
>>> TableB=sqlContext.createDataFrame(valuesB,['name','customerid'])
>>> valuesC=[('abc',1),('pqr',2),('mno',7),('efg',10),('hik',12)]
>>> TableC=sqlContext.createDataFrame(valuesC,['name','customerid'])
>>> result=TableB.unionAll(TableC)
>>> result.show()
+----+----------+
|name|customerid|
+----+----------+
| abc|         1|
| pqr|         2|
| mno|         7|
| xyz|         9|
| abc|         1|
| pqr|         2|
| mno|         7|
| efg|        10|
| hik|        12|
+----+----------+
```

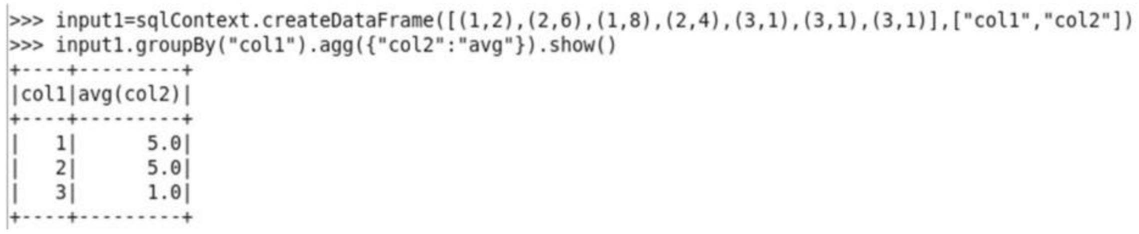
D. Aggregate and Grouping

Sum:

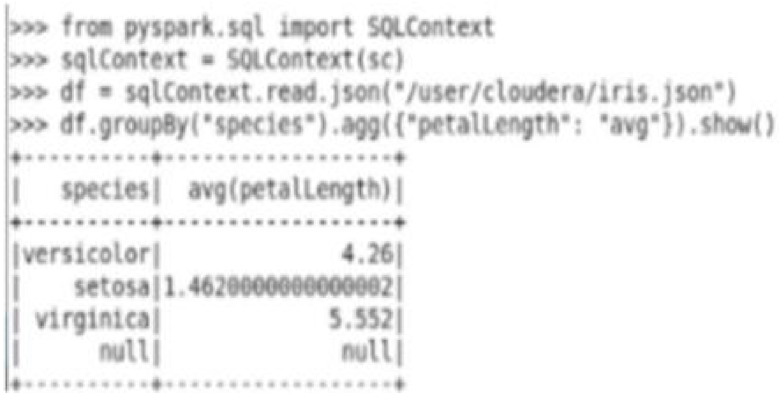


```
>>> data=[[1,2],[2,1],[4,3],[4,5],[5,4],[1,4],[1,1]]
>>> list1=sc.parallelize(data)
>>> list1.collect()
[[1, 2], [2, 1], [4, 3], [4, 5], [5, 4], [1, 4], [1, 1]]
>>>
>>>
>>> mapped_list=list1.map(lambda x: (x[0],x[1]))
>>> summation=mapped_list.reduceByKey(lambda x,y: x+y)
>>> summation.collect()
[(1, 7), (2, 1), (4, 8), (5, 4)]
```

Average:

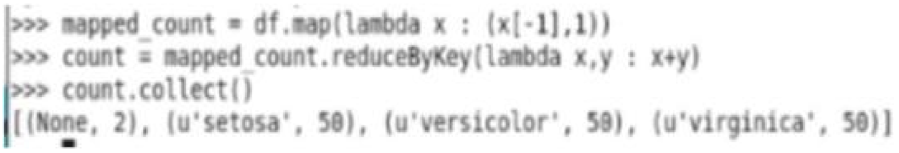


```
>>> input1=sqlContext.createDataFrame([(1,2),(2,6),(1,8),(2,4),(3,1),(3,1),(3,1)],["col1","col2"])
>>> input1.groupBy("col1").agg({"col2":"avg"}).show()
+----+---------+
|col1|avg(col2)|
+----+---------+
|   1|      5.0|
|   2|      5.0|
|   3|      1.0|
+----+---------+
```



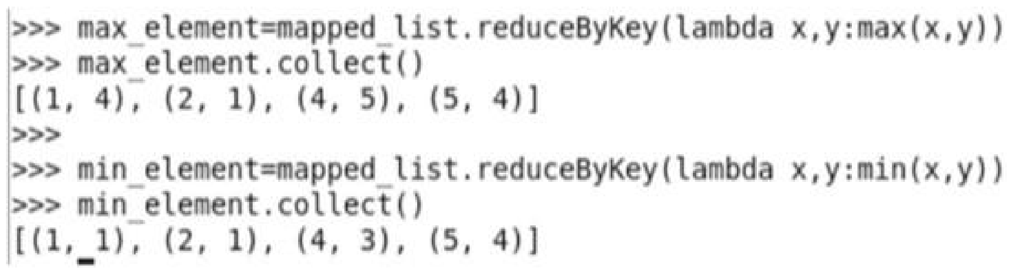
```
>>> from pyspark.sql import SQLContext
>>> sqlContext = SQLContext(sc)
>>> df = sqlContext.read.json("/user/cloudera/iris.json")
>>> df.groupBy("species").agg({"petalLength": "avg"}).show()
+----------+------------------+
|   species|  avg(petalLength)|
+----------+------------------+
|versicolor|              4.26|
|    setosa|1.4620000000000002|
| virginica|             5.552|
|      null|              null|
+----------+------------------+
```

Count:



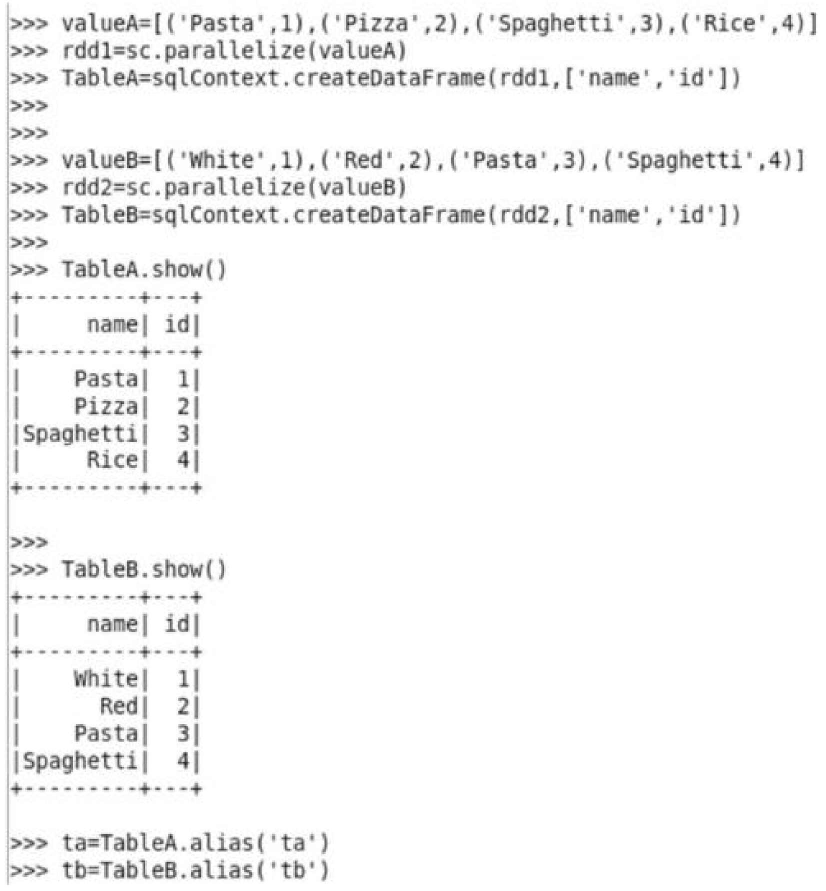
```
>>> mapped_count = df.map(lambda x : (x[-1],1))
>>> count = mapped_count.reduceByKey(lambda x,y : x+y)
>>> count.collect()
[(None, 2), (u'setosa', 50), (u'versicolor', 50), (u'virginica', 50)]
```

Max & Min element

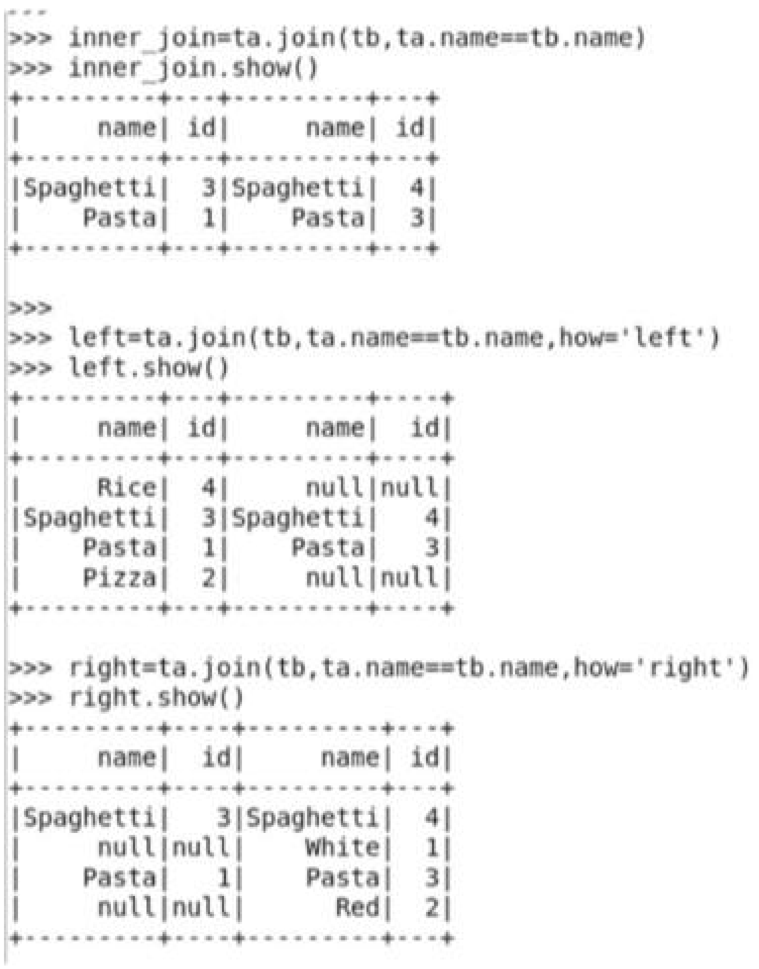


```
>>> max_element=mapped_list.reduceByKey(lambda x,y:max(x,y))
>>> max_element.collect()
[(1, 4), (2, 1), (4, 5), (5, 4)]
>>>
>>> min_element=mapped_list.reduceByKey(lambda x,y:min(x,y))
>>> min_element.collect()
[(1, 1), (2, 1), (4, 3), (5, 4)]
```

E. Join

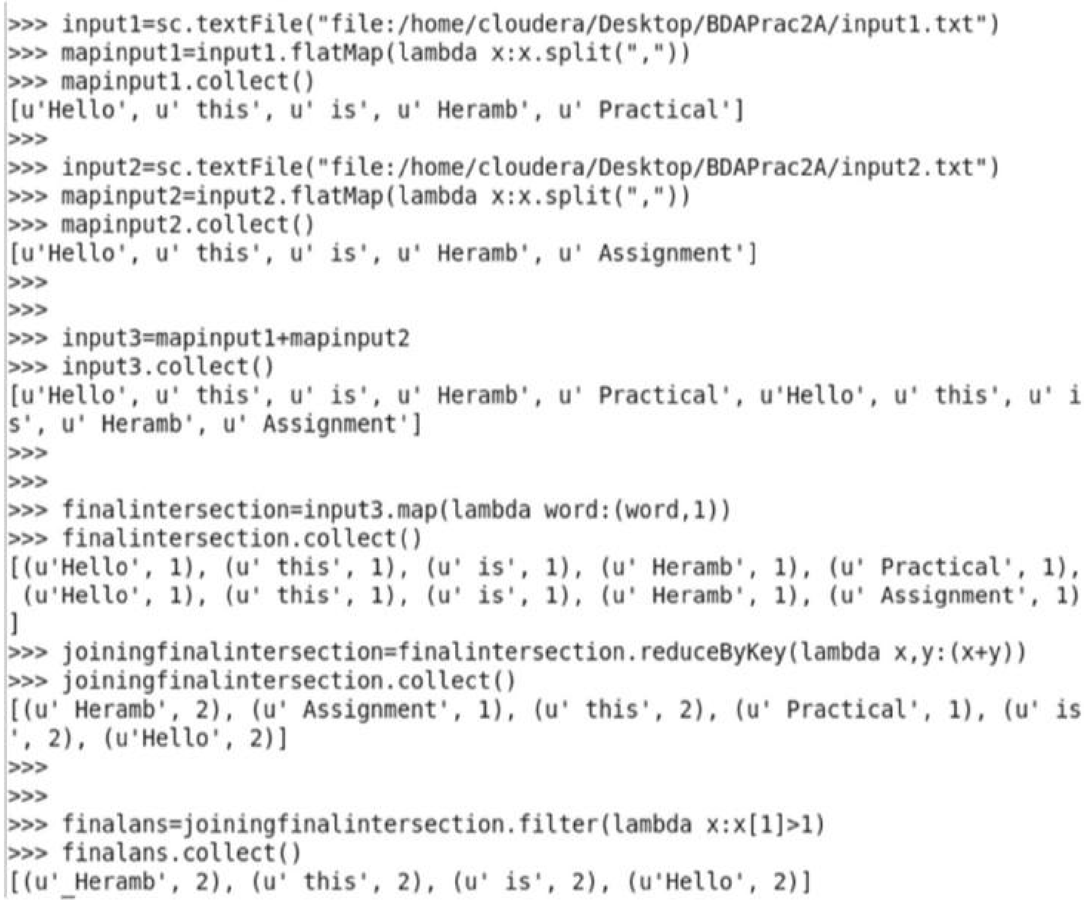


```
>>> valueA=[('Pasta',1),('Pizza',2),('Spaghetti',3),('Rice',4)]
>>> rdd1=sc.parallelize(valueA)
>>> TableA=sqlContext.createDataFrame(rdd1,['name','id'])
>>>
>>>
>>> valueB=[('White',1),('Red',2),('Pasta',3),('Spaghetti',4)]
>>> rdd2=sc.parallelize(valueB)
>>> TableB=sqlContext.createDataFrame(rdd2,['name','id'])
>>>
>>> TableA.show()
+---------+---+
|     name| id|
+---------+---+
|    Pasta|  1|
|    Pizza|  2|
|Spaghetti|  3|
|     Rice|  4|
+---------+---+

>>>
>>> TableB.show()
+---------+---+
|     name| id|
+---------+---+
|    White|  1|
|      Red|  2|
|    Pasta|  3|
|Spaghetti|  4|
+---------+---+

>>> ta=TableA.alias('ta')
>>> tb=TableB.alias('tb')
```



```
...
>>> inner_join=ta.join(tb,ta.name==tb.name)
>>> inner_join.show()
+---------+---+---------+---+
|     name| id|     name| id|
+---------+---+---------+---+
|Spaghetti|  3|Spaghetti|  4|
|    Pasta|  1|    Pasta|  3|
+---------+---+---------+---+

>>>
>>> left=ta.join(tb,ta.name==tb.name,how='left')
>>> left.show()
+---------+---+---------+----+
|     name| id|     name|  id|
+---------+---+---------+----+
|     Rice|  4|     null|null|
|Spaghetti|  3|Spaghetti|   4|
|    Pasta|  1|    Pasta|   3|
|    Pizza|  2|     null|null|
+---------+---+---------+----+

>>> right=ta.join(tb,ta.name==tb.name,how='right')
>>> right.show()
+---------+----+---------+---+
|     name|  id|     name| id|
+---------+----+---------+---+
|Spaghetti|   3|Spaghetti|  4|
|     null|null|    White|  1|
|    Pasta|   1|    Pasta|  3|
|     null|null|      Red|  2|
+---------+----+---------+---+
```

F. Intersection



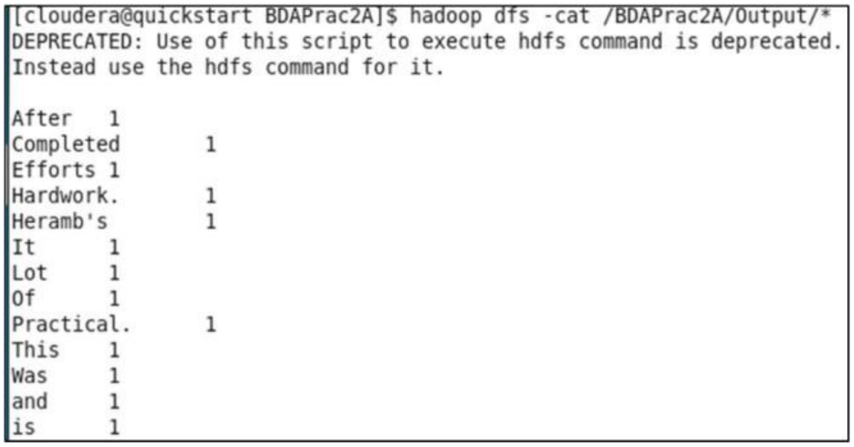
```
>>> input1=sc.textFile("file:/home/cloudera/Desktop/BDAPrac2A/input1.txt")
>>> mapinput1=input1.flatMap(lambda x:x.split(","))
>>> mapinput1.collect()
[u'Hello', u' this', u' is', u' Heramb', u' Practical']
>>>
>>> input2=sc.textFile("file:/home/cloudera/Desktop/BDAPrac2A/input2.txt")
>>> mapinput2=input2.flatMap(lambda x:x.split(","))
>>> mapinput2.collect()
[u'Hello', u' this', u' is', u' Heramb', u' Assignment']
>>>
>>>
>>> input3=mapinput1+mapinput2
>>> input3.collect()
[u'Hello', u' this', u' is', u' Heramb', u' Practical', u'Hello', u' this', u' i
s', u' Heramb', u' Assignment']
>>>
>>>
>>> finalintersection=input3.map(lambda word:(word,1))
>>> finalintersection.collect()
[(u'Hello', 1), (u' this', 1), (u' is', 1), (u' Heramb', 1), (u' Practical', 1),
 (u'Hello', 1), (u' this', 1), (u' is', 1), (u' Heramb', 1), (u' Assignment', 1)
]
>>> joiningfinalintersection=finalintersection.reduceByKey(lambda x,y:(x+y))
>>> joiningfinalintersection.collect()
[(u' Heramb', 2), (u' Assignment', 1), (u' this', 2), (u' Practical', 1), (u' is
', 2), (u'Hello', 2)]
>>>
>>>
>>> finalans=joiningfinalintersection.filter(lambda x:x[1]>1)
>>> finalans.collect()
[(u'_Heramb', 2), (u' this', 2), (u' is', 2), (u'Hello', 2)]
```

Program:

A. To implement the word Count

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

 public static class TokenizerMapper
    extends Mapper<Object, Text, Text, IntWritable>{

  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(Object key, Text value, Context context
          ) throws IOException, InterruptedException {
   StringTokenizer itr = new StringTokenizer(value.toString());
   while (itr.hasMoreTokens()) {
    word.set(itr.nextToken());
    context.write(word, one);
   }
  }
 }
public static class IntSumReducer
     extends Reducer<Text,IntWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();

  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
 }

 public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();
  Job job = Job.getInstance(conf, "word count");
  job.setJarByClass(WordCount.class);
  job.setMapperClass(TokenizerMapper.class);
  job.setCombinerClass(IntSumReducer.class);
  job.setReducerClass(IntSumReducer.class);
  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(IntWritable.class);
  FileInputFormat.addInputPath(job, new Path(args[0]));
  FileOutputFormat.setOutputPath(job, new Path(args[1]));
  System.exit(job.waitForCompletion(true) ? 0 : 1);
 }
}
```

Output

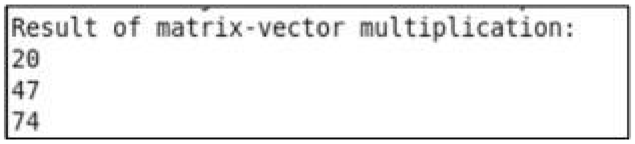

```
[cloudera@quickstart BDAPrac2A]$ hadoop dfs -cat /BDAPrac2A/Output/*
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

After   1
Completed       1
Efforts 1
Hardwork.       1
Heramb's        1
It      1
Lot     1
Of      1
Practical.      1
This    1
Was     1
and     1
is      1
```

B. Matrix-Vector multiplication

Programs:

```
public class MatrixVectorMultiplication {
    public static void main(String[] args) {
        // Define the matrix and vector
        int[][] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
        int[] vector = {2, 3, 4};

        // Check if matrix and vector dimensions are compatible
        int matrixRows = matrix.length;
        int matrixCols = matrix[0].length;
        int vectorSize = vector.length;

        if (matrixCols != vectorSize) {
            System.out.println("Matrix and vector dimensions are not compatible for
multiplication.");
            return;
        }

        // Perform matrix-vector multiplication
        int[] result = new int[matrixRows];
        for (int i = 0; i < matrixRows; i++) {
            for (int j = 0; j < matrixCols; j++) {
            result[i] += matrix[i][j] * vector[j];
         }
      }

      // Display the result
      System.out.println("Result of matrix-vector multiplication:");
      for (int i = 0; i < matrixRows; i++) {
         System.out.println(result[i]);
      }
   }
}
```

Output:



```
Result of matrix-vector multiplication:
20
47
74
```

Results and Discussions:

MapReduce is a parallel processing model for large scale data**:**

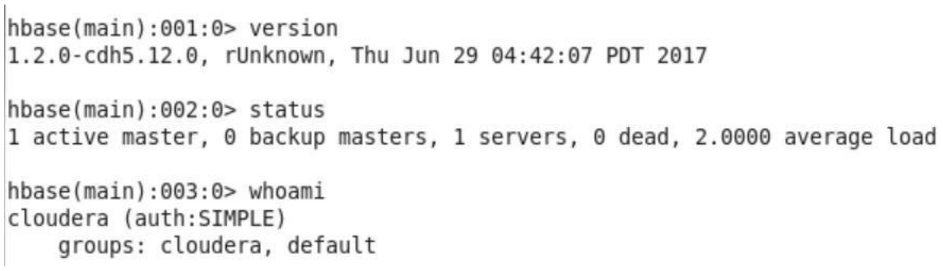
Word Count:

Result: Efficiently count words occurrences in texts.
Discussions: Map phase splits text, emits . Reduce phase aggregate counts. Scales well for basic tasks.

Matrix-Vector Multiplication:

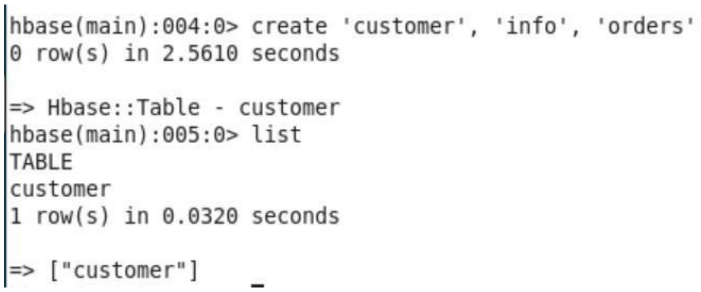
Output/Procedure:

1. General Commands:



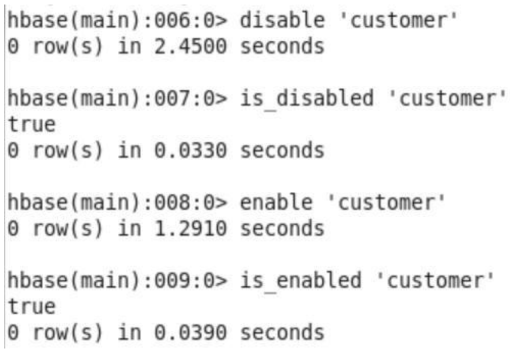
```
hbase(main):001:0> version
1.2.0-cdh5.12.0, rUnknown, Thu Jun 29 04:42:07 PDT 2017

hbase(main):002:0> status
1 active master, 0 backup masters, 1 servers, 0 dead, 2.0000 average load

hbase(main):003:0> whoami
cloudera (auth:SIMPLE)
    groups: cloudera, default
```

2. Create Table:



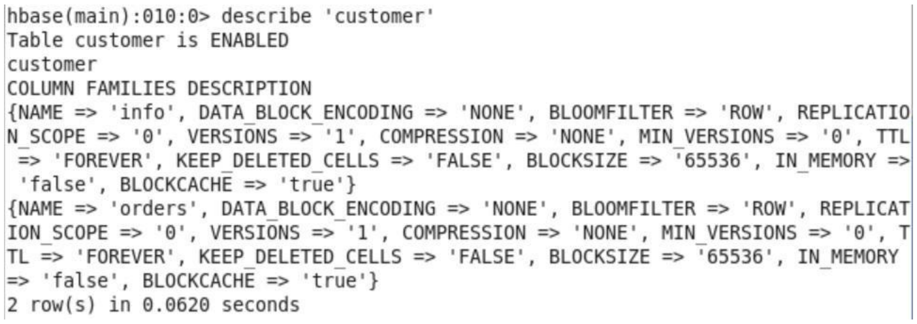
```
hbase(main):004:0> create 'customer', 'info', 'orders'
0 row(s) in 2.5610 seconds

=> Hbase::Table - customer
hbase(main):005:0> list
TABLE
customer
1 row(s) in 0.0320 seconds

=> ["customer"]
```

3. Disable and Enable Table:



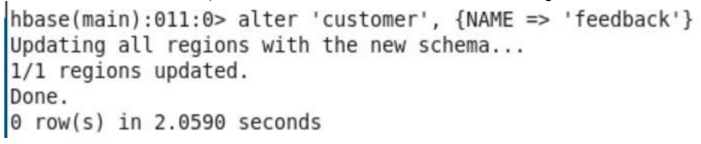
```
hbase(main):006:0> disable 'customer'
0 row(s) in 2.4500 seconds

hbase(main):007:0> is_disabled 'customer'
true
0 row(s) in 0.0330 seconds

hbase(main):008:0> enable 'customer'
0 row(s) in 1.2910 seconds

hbase(main):009:0> is_enabled 'customer'
true
0 row(s) in 0.0390 seconds
```

4. Describe Table:



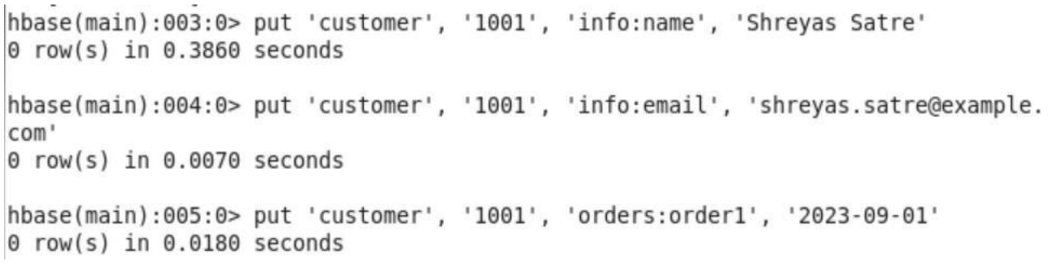
```
hbase(main):010:0> describe 'customer'
Table customer is ENABLED
customer
COLUMN FAMILIES DESCRIPTION
{NAME => 'info', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATIO
N_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL
 => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY =>
 'false', BLOCKCACHE => 'true'}
{NAME => 'orders', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICAT
ION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS => '0', T
TL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY
=> 'false', BLOCKCACHE => 'true'}
2 row(s) in 0.0620 seconds
```

5. Alter Table (Add a new column family 'feedback'):



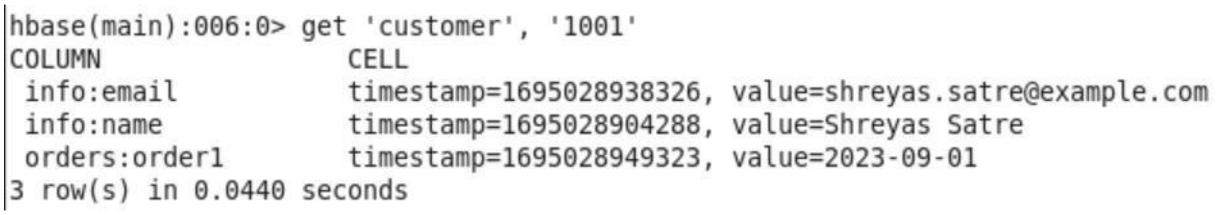
```
hbase(main):011:0> alter 'customer', {NAME => 'feedback'}
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 2.0590 seconds
```

6. Insert values in Table:



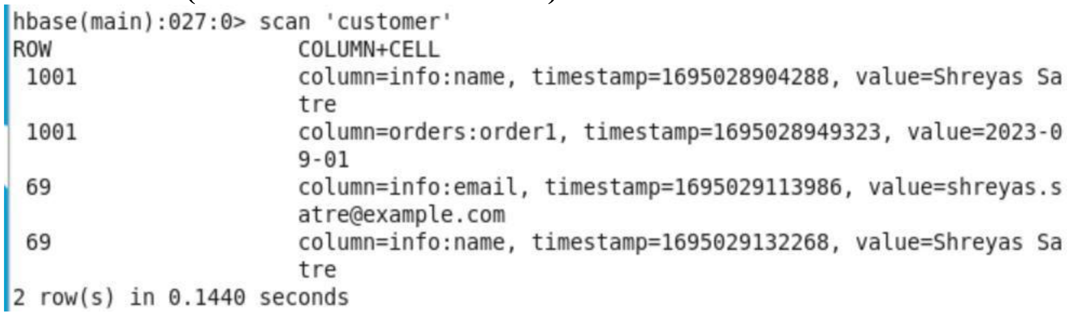
```
hbase(main):003:0> put 'customer', '1001', 'info:name', 'Shreyas Satre'
0 row(s) in 0.3860 seconds

hbase(main):004:0> put 'customer', '1001', 'info:email', 'shreyas.satre@example.
com'
0 row(s) in 0.0070 seconds

hbase(main):005:0> put 'customer', '1001', 'orders:order1', '2023-09-01'
0 row(s) in 0.0180 seconds
```

7. Retrive table data:



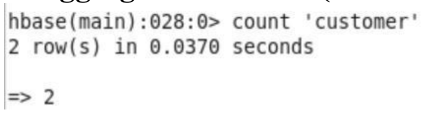
```
hbase(main):006:0> get 'customer', '1001'
COLUMN                     CELL
 info:email                timestamp=1695028938326, value=shreyas.satre@example.com
 info:name                 timestamp=1695028904288, value=Shreyas Satre
 orders:order1             timestamp=1695028949323, value=2023-09-01
3 row(s) in 0.0440 seconds
```

8. Scan table (List all customers/records):



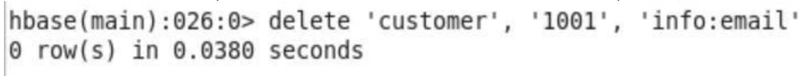
```
hbase(main):027:0> scan 'customer'
ROW                    COLUMN+CELL
 1001                  column=info:name, timestamp=1695028904288, value=Shreyas Sa
                       tre
 1001                  column=orders:order1, timestamp=1695028949323, value=2023-0
                       9-01
 69                    column=info:email, timestamp=1695029113986, value=shreyas.s
                       atre@example.com
 69                    column=info:name, timestamp=1695029132268, value=Shreyas Sa
                       tre
2 row(s) in 0.1440 seconds
```

9. Aggregate Functions(Count Records):



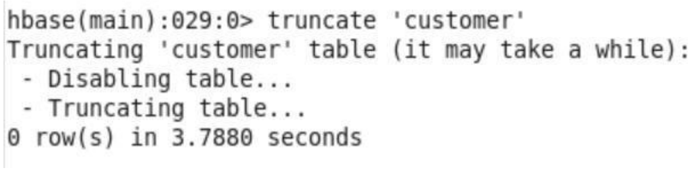
```
hbase(main):028:0> count 'customer'
2 row(s) in 0.0370 seconds

=> 2
```

10. Delete Cell (Remove email for a customer):



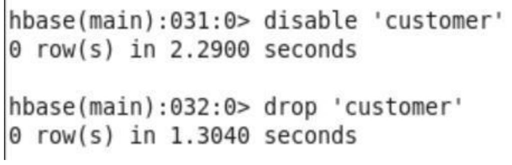
```
hbase(main):026:0> delete 'customer', '1001', 'info:email'
0 row(s) in 0.0380 seconds
```

11. Truncate Table:



```
hbase(main):029:0> truncate 'customer'
Truncating 'customer' table (it may take a while):
 - Disabling table...
 - Truncating table...
0 row(s) in 3.7880 seconds
```

12. Drop Table:

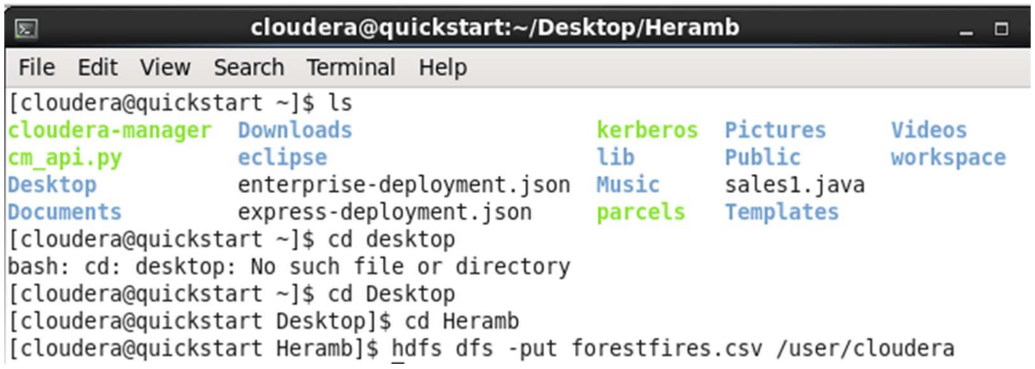


```
hbase(main):031:0> disable 'customer'
0 row(s) in 2.2900 seconds

hbase(main):032:0> drop 'customer'
0 row(s) in 1.3040 seconds
```

Program & Output:

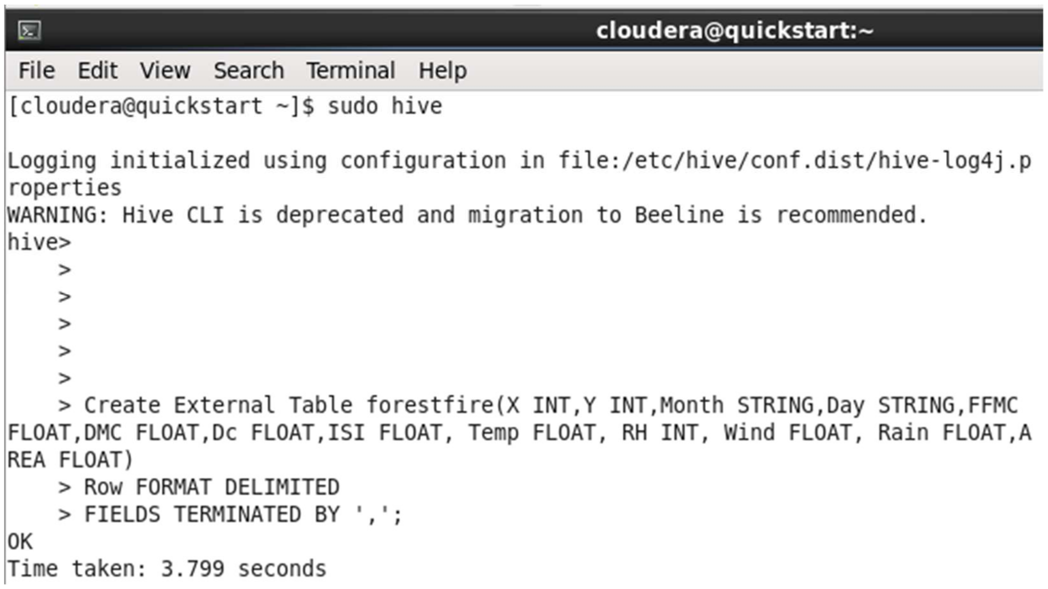
Download Dataset from : https://archive.ics.uci.edu/ml/datasets/forest+fires

Upload Dataset into Cloudera.



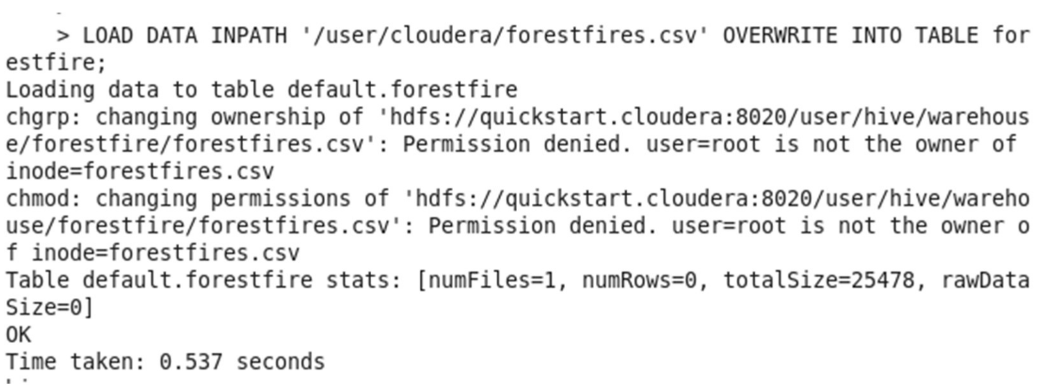
```
cloudera@quickstart:~/Desktop/Heramb
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ ls
cloudera-manager  Downloads                  kerberos  Pictures   Videos
cm_api.py         eclipse                    lib       Public     workspace
Desktop           enterprise-deployment.json Music     sales1.java
Documents         express-deployment.json    parcels   Templates
[cloudera@quickstart ~]$ cd desktop
bash: cd: desktop: No such file or directory
[cloudera@quickstart ~]$ cd Desktop
[cloudera@quickstart Desktop]$ cd Heramb
[cloudera@quickstart Heramb]$ hdfs dfs -put forestfires.csv /user/cloudera
```

Opening Hive Shell & Creating ForestFire Table:



```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ sudo hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.p
roperties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive>
    >
    >
    >
    >
    >
    > Create External Table forestfire(X INT,Y INT,Month STRING,Day STRING,FFMC
FLOAT,DMC FLOAT,Dc FLOAT,ISI FLOAT, Temp FLOAT, RH INT, Wind FLOAT, Rain FLOAT,A
REA FLOAT)
    > Row FORMAT DELIMITED
    > FIELDS TERMINATED BY ',';
OK
Time taken: 3.799 seconds
```

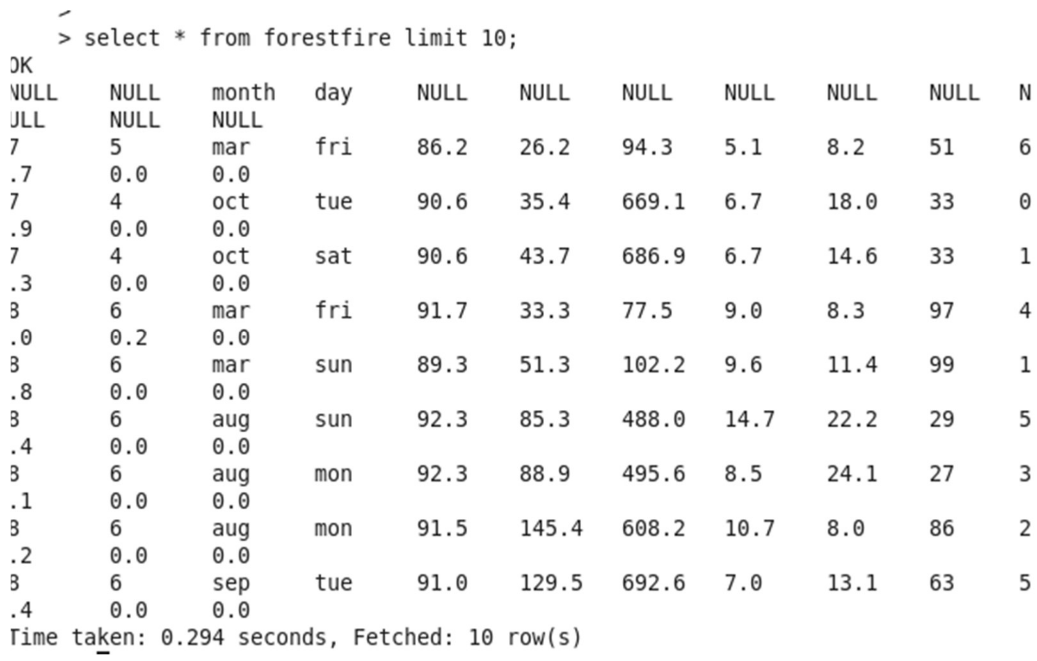
Loading Data From Dataset Into ForestFire Table:



```
    > LOAD DATA INPATH '/user/cloudera/forestfires.csv' OVERWRITE INTO TABLE for
estfire;
Loading data to table default.forestfire
chgrp: changing ownership of 'hdfs://quickstart.cloudera:8020/user/hive/warehous
e/forestfire/forestfires.csv': Permission denied. user=root is not the owner of
inode=forestfires.csv
chmod: changing permissions of 'hdfs://quickstart.cloudera:8020/user/hive/wareho
use/forestfire/forestfires.csv': Permission denied. user=root is not the owner o
f inode=forestfires.csv
Table default.forestfire stats: [numFiles=1, numRows=0, totalSize=25478, rawData
Size=0]
OK
Time taken: 0.537 seconds
```

Executing Queries:

Query 1 : select * from forestfire limit 10;



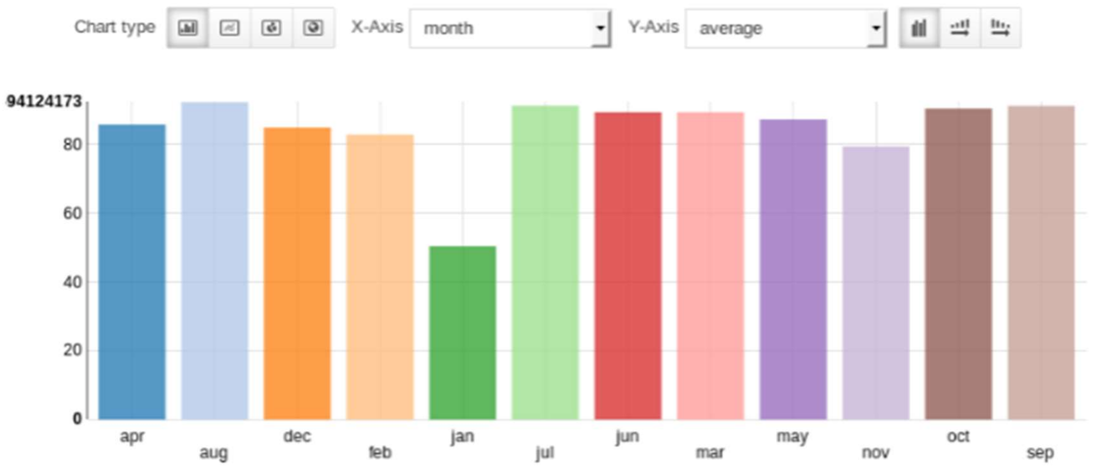
```
    > select * from forestfire limit 10;
OK
NULL    NULL    month   day     NULL    NULL    NULL    NULL    NULL    NULL    N
ULL     NULL    NULL
7       5       mar     fri     86.2    26.2    94.3    5.1     8.2     51      6
.7      0.0     0.0
7       4       oct     tue     90.6    35.4    669.1   6.7     18.0    33      0
.9      0.0     0.0
7       4       oct     sat     90.6    43.7    686.9   6.7     14.6    33      1
.3      0.0     0.0
8       6       mar     fri     91.7    33.3    77.5    9.0     8.3     97      4
.0      0.2     0.0
8       6       mar     sun     89.3    51.3    102.2   9.6     11.4    99      1
.8      0.0     0.0
8       6       aug     sun     92.3    85.3    488.0   14.7    22.2    29      5
.4      0.0     0.0
8       6       aug     mon     92.3    88.9    495.6   8.5     24.1    27      3
.1      0.0     0.0
8       6       aug     mon     91.5    145.4   608.2   10.7    8.0     86      2
.2      0.0     0.0
8       6       sep     tue     91.0    129.5   692.6   7.0     13.1    63      5
.4      0.0     0.0
Time taken: 0.294 seconds, Fetched: 10 row(s)
```

Query 2: : select * from forestfire where x=7 and y=4 limit 10;

```
    > select * from forestfire where X=7 and Y=4 limit 10;
OK
7       4       oct     tue     90.6    35.4    669.1   6.7     18.0    33      0.9     0.0     0.0
7       4       oct     sat     90.6    43.7    686.9   6.7     14.6    33      1.3     0.0     0.0
7       4       jun     sun     94.3    96.3    200.0   56.1    21.0    44      4.5     0.0     0.0
7       4       aug     sat     90.2    110.9   537.4   6.2     19.5    43      5.8     0.0     0.0
7       4       aug     sat     93.5    139.4   594.2   20.3    23.7    32      5.8     0.0     0.0
7       4       aug     sun     91.4    142.4   601.4   10.6    16.3    60      5.4     0.0     0.0
7       4       sep     fri     92.4    117.9   668.0   12.2    19.0    34      5.8     0.0     0.0
7       4       sep     mon     90.9    126.5   686.5   7.0     19.4    48      1.3     0.0     0.0
7       4       oct     fri     90.0    41.5    682.6   8.7     11.3    60      5.4     0.0     0.0
7       4       aug     sun     94.8    108.3   647.1   17.0    16.4    47      1.3     0.0     1.56
Time taken: 0.2 seconds, Fetched: 10 row(s)
```

Query 3: select MONTH, avg(FFMC) as Average from forestfire group by MONTH;

```
apr     85.7888895670573
aug     92.33695594124173
dec     84.96666717529297
feb     82.90499916076661
jan     50.39999961853027
jul     91.32812428474426
jun     89.42941194422104
mar     89.44444345544886
may     87.3499984741211
month   NULL
nov     79.5
oct     90.45333251953124
sep     91.24302336227062
Time taken: 29.623 seconds, Fetched: 13 row(s)
```

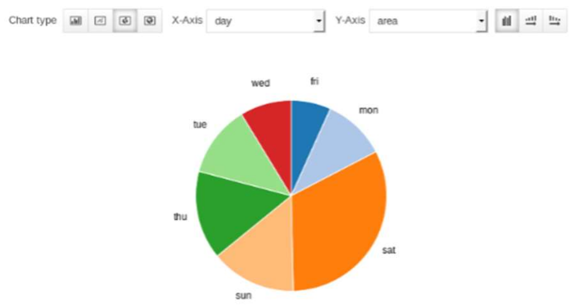


Query 4: SELECT MONTH , MAX(RH) AS MAXIMUM FROM forestfire GROUP BY MONTH HAVING MONTH ='sep';

```
OK
sep     86
Time taken: 26.654 seconds, Fetched: 1 row(s)
```

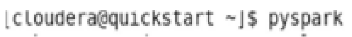
Query 5: select DAY, SUM(AREA) AS AREA from forestfire group by DAY ORDER BY DAY;

```
day     NULL
fri     447.24000039696693
mon     706.5299995839596
sat     2144.8599796295166
sun     959.9299972057343
thu     997.1000298261642
tue     807.79000864923
wed     578.5999903082848
Time taken: 45.033 seconds, Fetched: 8 row(s)
```

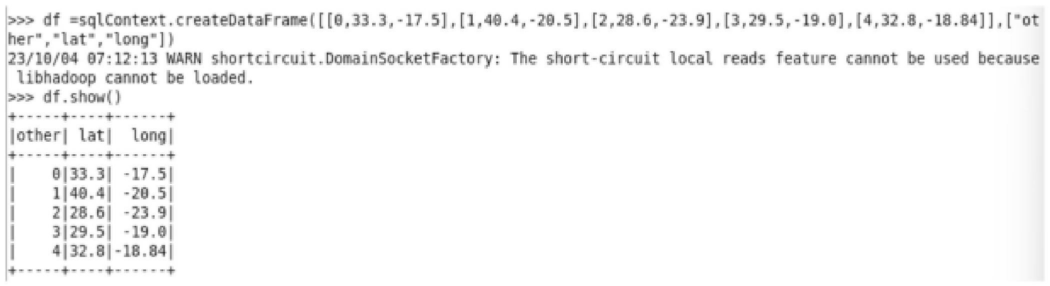


Query 6: SELECT MONTH, MAX(DC) AS MAXIMUM FROM forestfire GROUP BY MONTH ORDER BY MONTH;

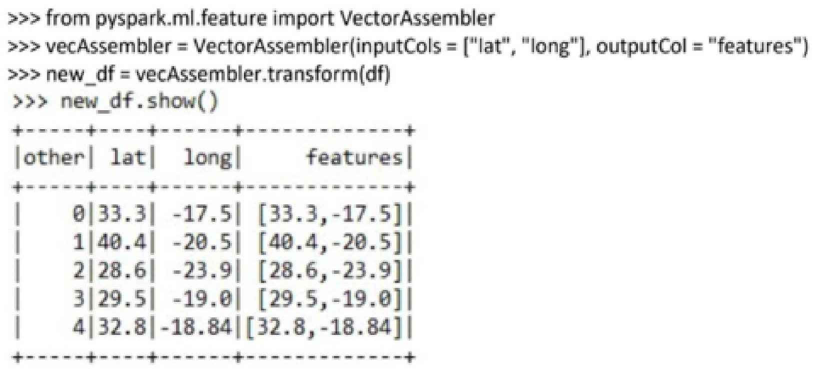
```
apr     97.1
aug     819.1
dec     354.6
feb     353.5
jan     171.4
jul     795.9
jun     433.3
mar     103.8
may     113.8
month   NULL
nov     106.7
oct     696.1
sep     860.6
Time taken: 50.182 seconds, Fetched: 13 row(s)
```



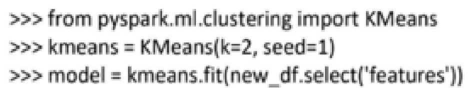
```
[cloudera@quickstart ~]$ pyspark
```



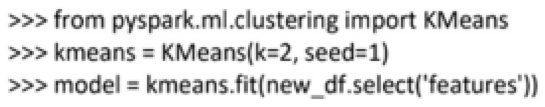
```
>>> df =sqlContext.createDataFrame([[0,33.3,-17.5],[1,40.4,-20.5],[2,28.6,-23.9],[3,29.5,-19.0],[4,32.8,-18.84]],["ot
her","lat","long"])
23/10/04 07:12:13 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature cannot be used because
 libhadoop cannot be loaded.
>>> df.show()
+-----+----+------+
|other| lat|  long|
+-----+----+------+
|    0|33.3| -17.5|
|    1|40.4| -20.5|
|    2|28.6| -23.9|
|    3|29.5| -19.0|
|    4|32.8|-18.84|
+-----+----+------+
```



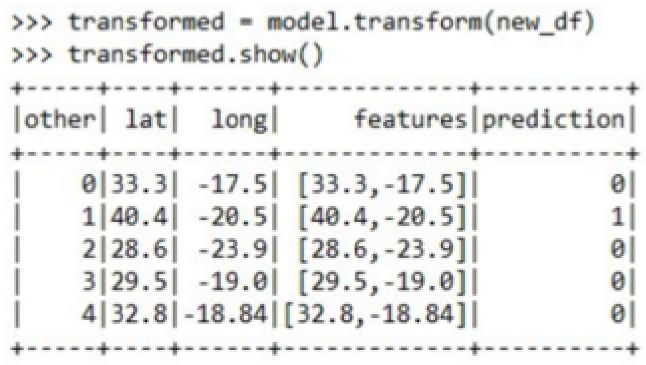
```
>>> from pyspark.ml.feature import VectorAssembler
>>> vecAssembler = VectorAssembler(inputCols = ["lat", "long"], outputCol = "features")
>>> new_df = vecAssembler.transform(df)
>>> new_df.show()
+-----+----+------+-------------+
|other| lat|  long|     features|
+-----+----+------+-------------+
|    0|33.3| -17.5| [33.3,-17.5]|
|    1|40.4| -20.5| [40.4,-20.5]|
|    2|28.6| -23.9| [28.6,-23.9]|
|    3|29.5| -19.0| [29.5,-19.0]|
|    4|32.8|-18.84|[32.8,-18.84]|
+-----+----+------+-------------+
```



```
>>> from pyspark.ml.clustering import KMeans
>>> kmeans = KMeans(k=2, seed=1)
>>> model = kmeans.fit(new_df.select('features'))
```



```
>>> from pyspark.ml.clustering import KMeans
>>> kmeans = KMeans(k=2, seed=1)
>>> model = kmeans.fit(new_df.select('features'))
```



```
>>> transformed = model.transform(new_df)
>>> transformed.show()
+-----+----+------+-------------+----------+
|other| lat|  long|     features|prediction|
+-----+----+------+-------------+----------+
|    0|33.3| -17.5| [33.3,-17.5]|         0|
|    1|40.4| -20.5| [40.4,-20.5]|         1|
|    2|28.6| -23.9| [28.6,-23.9]|         0|
|    3|29.5| -19.0| [29.5,-19.0]|         0|
|    4|32.8|-18.84|[32.8,-18.84]|         0|
+-----+----+------+-------------+----------+
```

Results and Discussions: