Big Data And Hadoop

Project 2.2 - USA Consumer Forum Data Analysis

1. Executive Summary

1.1. Project Overview

To develop the System to analyze the USA Consumer Forum Data.

1.2.Purpose And Scope of specification

The purpose of this project is to capture the data for analyzing the complaints registered by the consumers at the forum.

In Scope:

The following requirement will be addressed:

a. Developing system to handle the data of the complaints registered at the forum and store the information in Hadoop Cluster (Flume).
b. Analyze the data that could be used to get the insight of various businesses.
c. Store the results in RDBMS (MySQL).

2. Product/Service Description

2.1.Assumptions

The data is accessible to the system to download it. The data is present in a CSV file.

2.2.Constraints

Describe any item that will constrain the design options, including

a. This system may not be used for searching for now. But it will be used for analysis and saving the relevant information as of now.
b. System will be using MySQL as the database.

3. Problem Statement:

The dataset is in csv format and contains the attributes pertaining to resolution of the consumer complaints.

You need to copy the dataset into HDFS using Flume and the results of the problem statements should be exported into RDBMS(Mysql) using sqoop.

The aim of this project is to analyze performance of various companies on aspects like:

i. Write a pig script to find no of complaints which got timely response.
ii. Write a pig script to find no of complaints where consumer forum forwarded the complaint same day they received to respective company.
iii. Write a pig script to find list of companies toping in complaint chart (companies with maximum number of complaints).

iv. Write a pig script to find no of complaints filed with product type has “Debt collection” for the year 2015.

Submit the screenshots of all the solutions with the source code.

4. **Input dataset download link**

   Associated Data File is placed at the following location
   https://drive.google.com/file/d/0B1QaXx7tpw3SQTlnQ0MzVW5HajA/view?usp=sharing

5. **Dataset description**

   Below is the description of the data set

Column heading	**Index**	**Description**
Date received	0	date on which consumer filed the complaint
Product	1	Type of the product
Sub-product	2	Sub product type
Issue	3	Issue faced by the consumer
Sub-issue	4	Any sub issues if exists
Consumer complaint narrative	5	Detailed description of complaint
Company public response	6	Company’s public response to the complaint
Company	7	Name of the company
State	8	State from which consumer filed the complaint
ZIP code	9	Zip code
Submitted via	10	Channel from which complaint was submitted
Date sent to company	11	Date on which consumer forum forwarded the complaint to company
Company response to consumer	12	Company’s response to the consumer
Timely response?	13	
Consumer disputed?	14	
Complaint ID	15	Unique complaint id

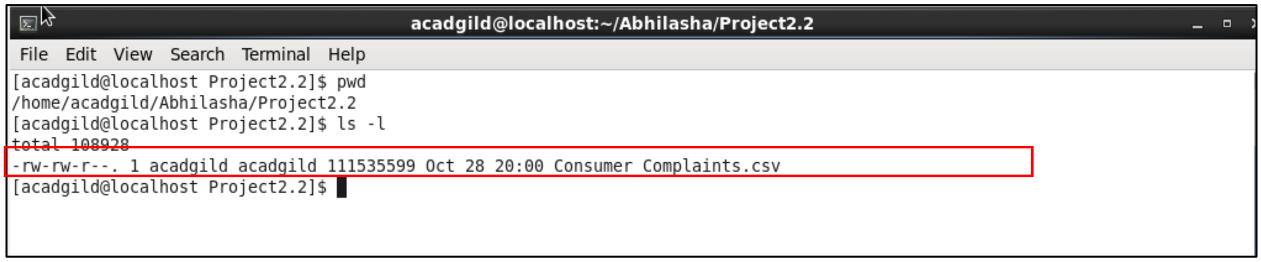
   This data is comma delimited.

6. **Solution**

   a. **Download the data into local file system**

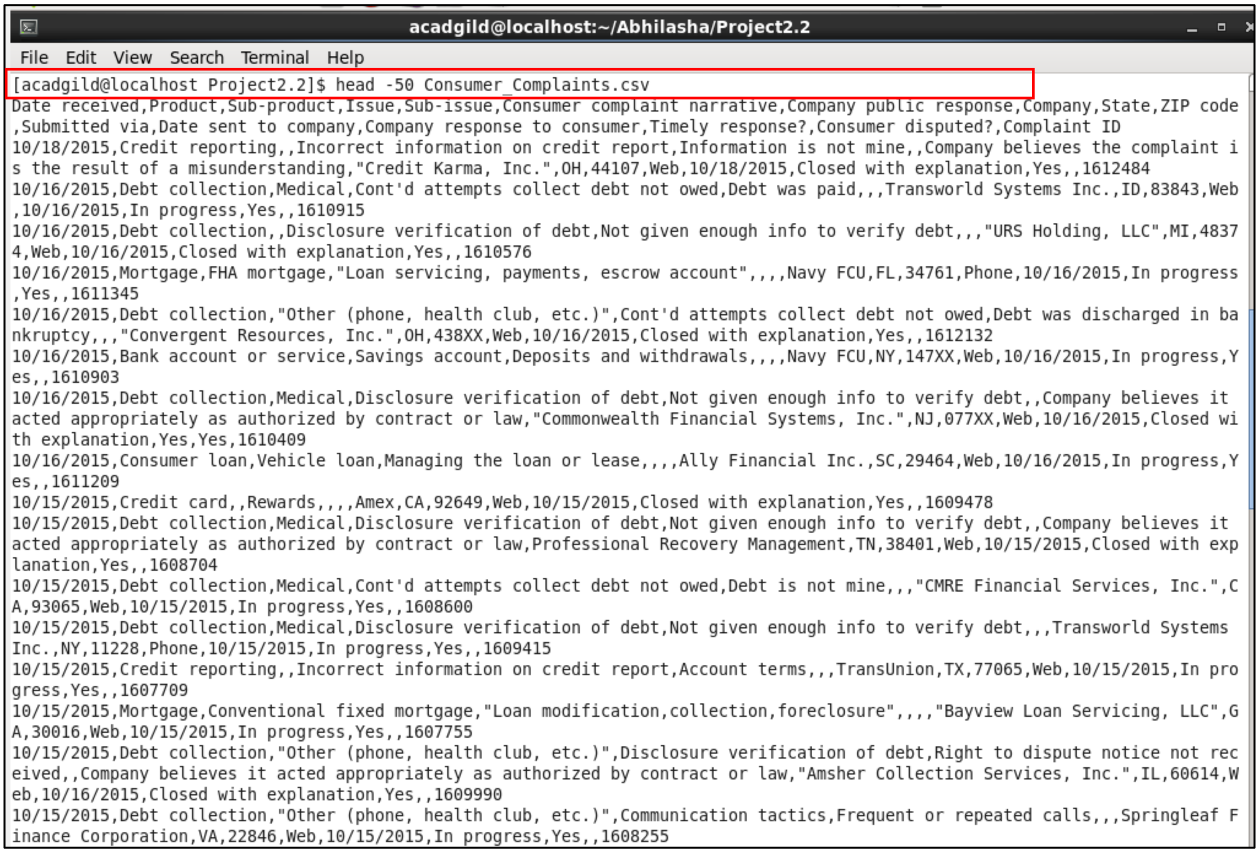
      The data file is downloaded from the link mentioned in section 4 and placed in the local file system at **/home/acadgild/Project2.2**. Name of the file is **Consumer_Complaints.csv.**

This file can be seen on the local system using **ls** command as follows:


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
[acadgild@localhost Project2.2]$ pwd
/home/acadgild/Abhilasha/Project2.2
[acadgild@localhost Project2.2]$ ls -l
total 108928
-rw-rw-r--. 1 acadgild acadgild 111535599 Oct 28 20:00 Consumer Complaints.csv
[acadgild@localhost Project2.2]$
```

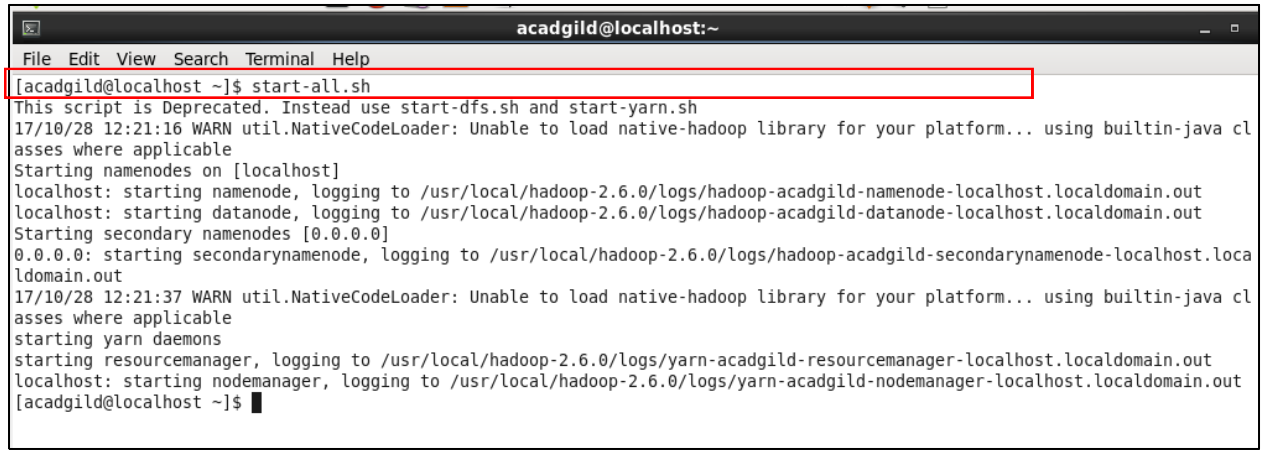
Using **head** command, we get can view the content of this file as follows. Here we have displayed only first 50 lines of the file.


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
[acadgild@localhost Project2.2]$ head -50 Consumer_Complaints.csv
Date received,Product,Sub-product,Issue,Sub-issue,Consumer complaint narrative,Company public response,Company,State,ZIP code
,Submitted via,Date sent to company,Company response to consumer,Timely response?,Consumer disputed?,Complaint ID
10/18/2015,Credit reporting,,Incorrect information on credit report,Information is not mine,,Company believes the complaint i
s the result of a misunderstanding,"Credit Karma, Inc.",OH,44107,Web,10/18/2015,Closed with explanation,Yes,,1612484
10/16/2015,Debt collection,Medical,Cont'd attempts collect debt not owed,Debt was paid,,,Transworld Systems Inc.,ID,83843,Web
,10/16/2015,In progress,Yes,,1610915
10/16/2015,Debt collection,,Disclosure verification of debt,Not given enough info to verify debt,,,"URS Holding, LLC",MI,4837
4,Web,10/16/2015,Closed with explanation,Yes,,1610576
10/16/2015,Mortgage,FHA mortgage,"Loan servicing, payments, escrow account",,,,Navy FCU,FL,34761,Phone,10/16/2015,In progress
,Yes,,1611345
10/16/2015,Debt collection,"Other (phone, health club, etc.)",Cont'd attempts collect debt not owed,Debt was discharged in ba
nkruptcy,,,"Convergent Resources, Inc.",OH,438XX,Web,10/16/2015,Closed with explanation,Yes,,1612132
10/16/2015,Bank account or service,Savings account,Deposits and withdrawals,,,,Navy FCU,NY,147XX,Web,10/16/2015,In progress,Y
es,,1610903
10/16/2015,Debt collection,Medical,Disclosure verification of debt,Not given enough info to verify debt,,Company believes it
acted appropriately as authorized by contract or law,"Commonwealth Financial Systems, Inc.",NJ,077XX,Web,10/16/2015,Closed wi
th explanation,Yes,Yes,1610409
10/16/2015,Consumer loan,Vehicle loan,Managing the loan or lease,,,,Ally Financial Inc.,SC,29464,Web,10/16/2015,In progress,Y
es,,1611209
10/15/2015,Credit card,,Rewards,,,,Amex,CA,92649,Web,10/15/2015,Closed with explanation,Yes,,1609478
10/15/2015,Debt collection,Medical,Disclosure verification of debt,Not given enough info to verify debt,,Company believes it
acted appropriately as authorized by contract or law,Professional Recovery Management,TN,38401,Web,10/15/2015,Closed with exp
lanation,Yes,,1608704
10/15/2015,Debt collection,Medical,Cont'd attempts collect debt not owed,Debt is not mine,,,"CMRE Financial Services, Inc.",C
A,93065,Web,10/15/2015,In progress,Yes,,1608600
10/15/2015,Debt collection,Medical,Disclosure verification of debt,Not given enough info to verify debt,,,Transworld Systems
Inc.,NY,11228,Phone,10/15/2015,In progress,Yes,,1609415
10/15/2015,Credit reporting,,Incorrect information on credit report,Account terms,,,TransUnion,TX,77065,Web,10/15/2015,In pro
gress,Yes,,1607709
10/15/2015,Mortgage,Conventional fixed mortgage,"Loan modification,collection,foreclosure",,,,"Bayview Loan Servicing, LLC",G
A,30016,Web,10/15/2015,In progress,Yes,,1607755
10/15/2015,Debt collection,"Other (phone, health club, etc.)",Disclosure verification of debt,Right to dispute notice not rec
eived,,Company believes it acted appropriately as authorized by contract or law,"Amsher Collection Services, Inc.",IL,60614,W
eb,10/16/2015,Closed with explanation,Yes,,1609990
10/15/2015,Debt collection,"Other (phone, health club, etc.)",Communication tactics,Frequent or repeated calls,,,Springleaf F
inance Corporation,VA,22846,Web,10/15/2015,In progress,Yes,,1608255
```

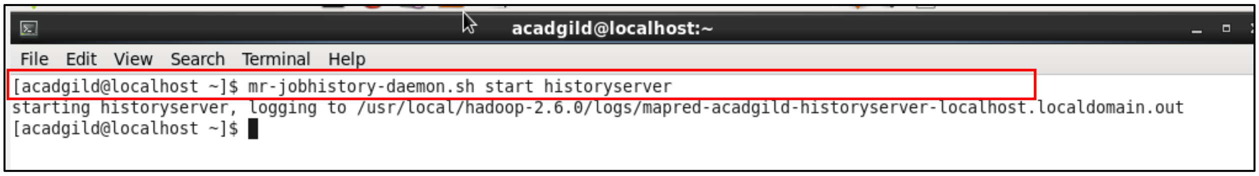
b. Start Hadoop cluster and history server

We first start Hadoop cluster using the command **start-all.sh** as follows:



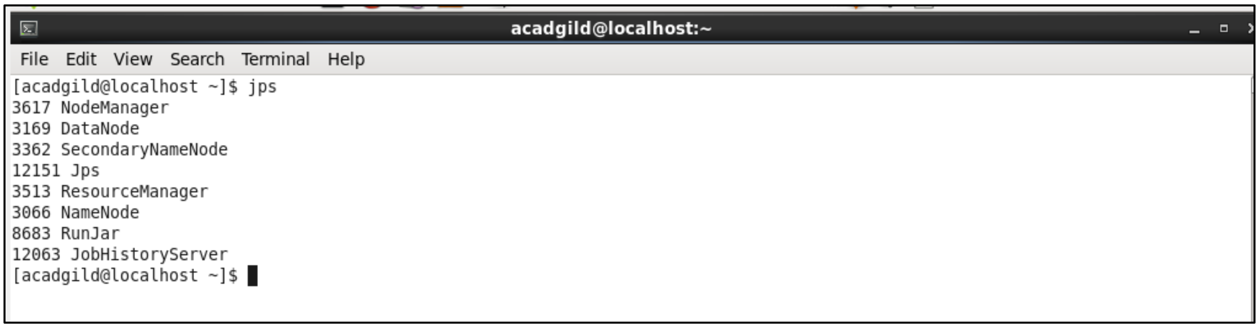
```
[acadgild@localhost ~]$ start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
17/10/28 12:21:16 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop-2.6.0/logs/hadoop-acadgild-namenode-localhost.localdomain.out
localhost: starting datanode, logging to /usr/local/hadoop-2.6.0/logs/hadoop-acadgild-datanode-localhost.localdomain.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop-2.6.0/logs/hadoop-acadgild-secondarynamenode-localhost.loca
ldomain.out
17/10/28 12:21:37 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop-2.6.0/logs/yarn-acadgild-resourcemanager-localhost.localdomain.out
localhost: starting nodemanager, logging to /usr/local/hadoop-2.6.0/logs/yarn-acadgild-nodemanager-localhost.localdomain.out
[acadgild@localhost ~]$
```

Now, we start the history server using the command **mr-jobhistory-daemon.sh start historyserver** as follows:



```
[acadgild@localhost ~]$ mr-jobhistory-daemon.sh start historyserver
starting historyserver, logging to /usr/local/hadoop-2.6.0/logs/mapred-acadgild-historyserver-localhost.localdomain.out
[acadgild@localhost ~]$
```

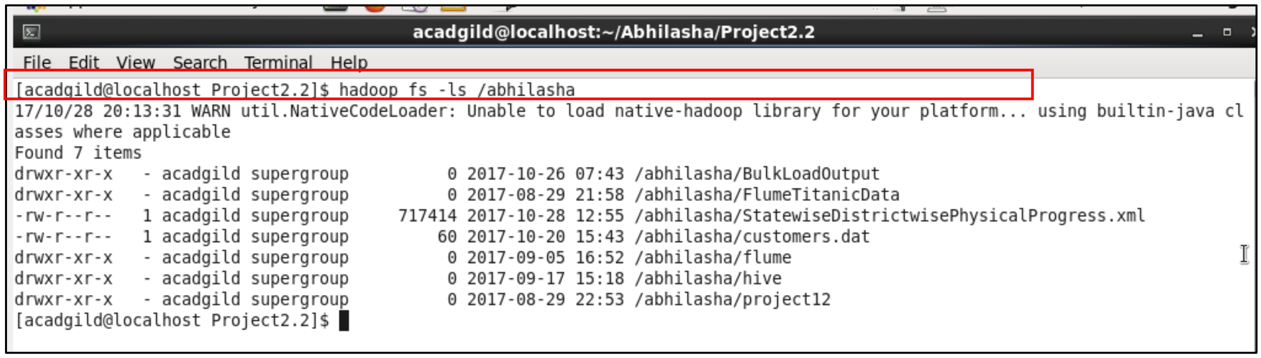
The threads running after the start up are seen using **jps** command as follows:



```
[acadgild@localhost ~]$ jps
3617 NodeManager
3169 DataNode
3362 SecondaryNameNode
12151 Jps
3513 ResourceManager
3066 NameNode
8683 RunJar
12063 JobHistoryServer
[acadgild@localhost ~]$
```

c. Load data into HDFS using Flume

Now, we need to place this file from local file system to HDFS.
We use the **ls** command to check if the input file already existed in HDFS:


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
[acadgild@localhost Project2.2]$ hadoop fs -ls /abhilasha
17/10/28 20:13:31 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 7 items
drwxr-xr-x   - acadgild supergroup          0 2017-10-26 07:43 /abhilasha/BulkLoadOutput
drwxr-xr-x   - acadgild supergroup          0 2017-08-29 21:58 /abhilasha/FlumeTitanicData
-rw-r--r--   1 acadgild supergroup     717414 2017-10-28 12:55 /abhilasha/StatewiseDistrictwisePhysicalProgress.xml
-rw-r--r--   1 acadgild supergroup         60 2017-10-20 15:43 /abhilasha/customers.dat
drwxr-xr-x   - acadgild supergroup          0 2017-09-05 16:52 /abhilasha/flume
drwxr-xr-x   - acadgild supergroup          0 2017-09-17 15:18 /abhilasha/hive
drwxr-xr-x   - acadgild supergroup          0 2017-08-29 22:53 /abhilasha/project12
[acadgild@localhost Project2.2]$
```

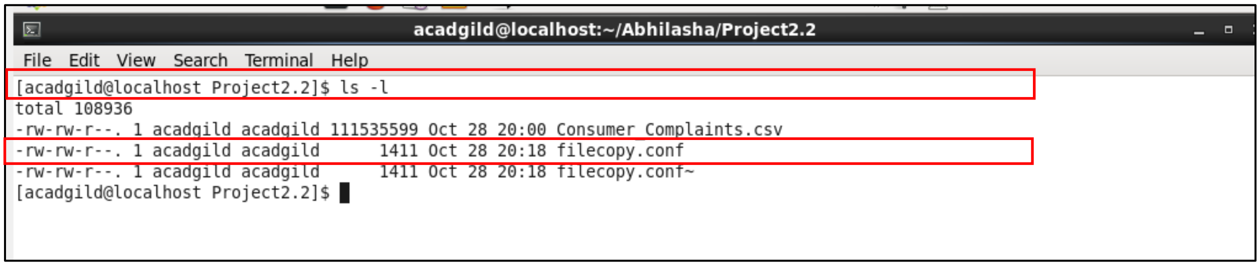
In the above screenshot, the file named **Consumer_Complaints.csv** is not listed. So, this file is not already present.

Flume agent is made of three parts

a. Source
b. Channel
c. Sink

In our use-case, source is the exec, sink in HDFS and the channel is memory channel.

The location of the configuration file to be used in flume is **/home/acadgild/Abhilasha/Project2.2**. Its name is **filecopy.conf**. This file can be seen using **ls** command as follows:


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
[acadgild@localhost Project2.2]$ ls -l
total 108936
-rw-rw-r--. 1 acadgild acadgild 111535599 Oct 28 20:00 Consumer_Complaints.csv
-rw-rw-r--. 1 acadgild acadgild      1411 Oct 28 20:18 filecopy.conf
-rw-rw-r--. 1 acadgild acadgild      1411 Oct 28 20:18 filecopy.conf~
[acadgild@localhost Project2.2]$
```

P**roperties of source** defined in it are as follows:

i. type = exec
ii. command = hadoop dfs -put /home/acadgild/Abhilasha/Project2.2/Consumer_Complaints.csv /abhilasha/

P**roperties of the channel** defined in it are as follows:

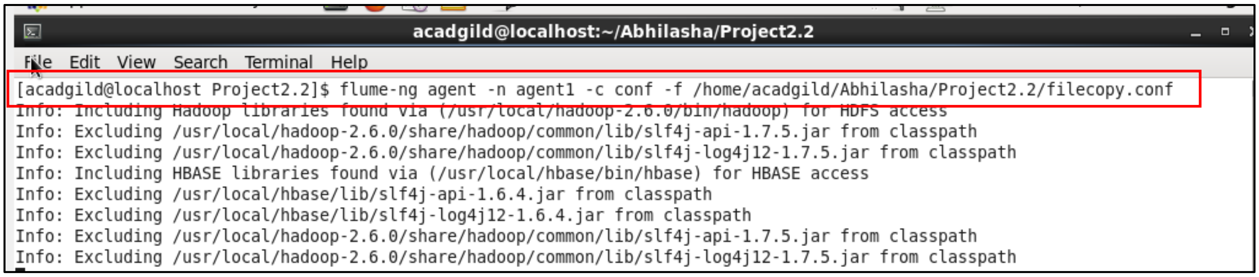
i. type = memory

P**roperties of the sink** defined in it are as follows:

i. type = hdfs
ii. path = hdfs://localhost:9000/abhilasha/

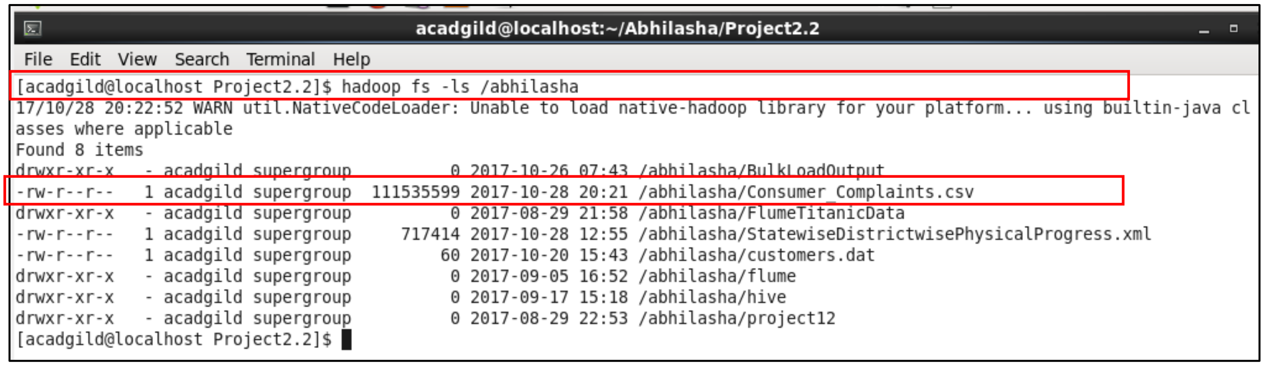
The command used to put data into HDFS using flume is
flume-ng agent –n agent1 –c conf –f /home/acadgild/Abhilasha/Project2.2/filecopy.conf

In this command, we have mentioned the configuration file to be used in the execution of flume job.


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

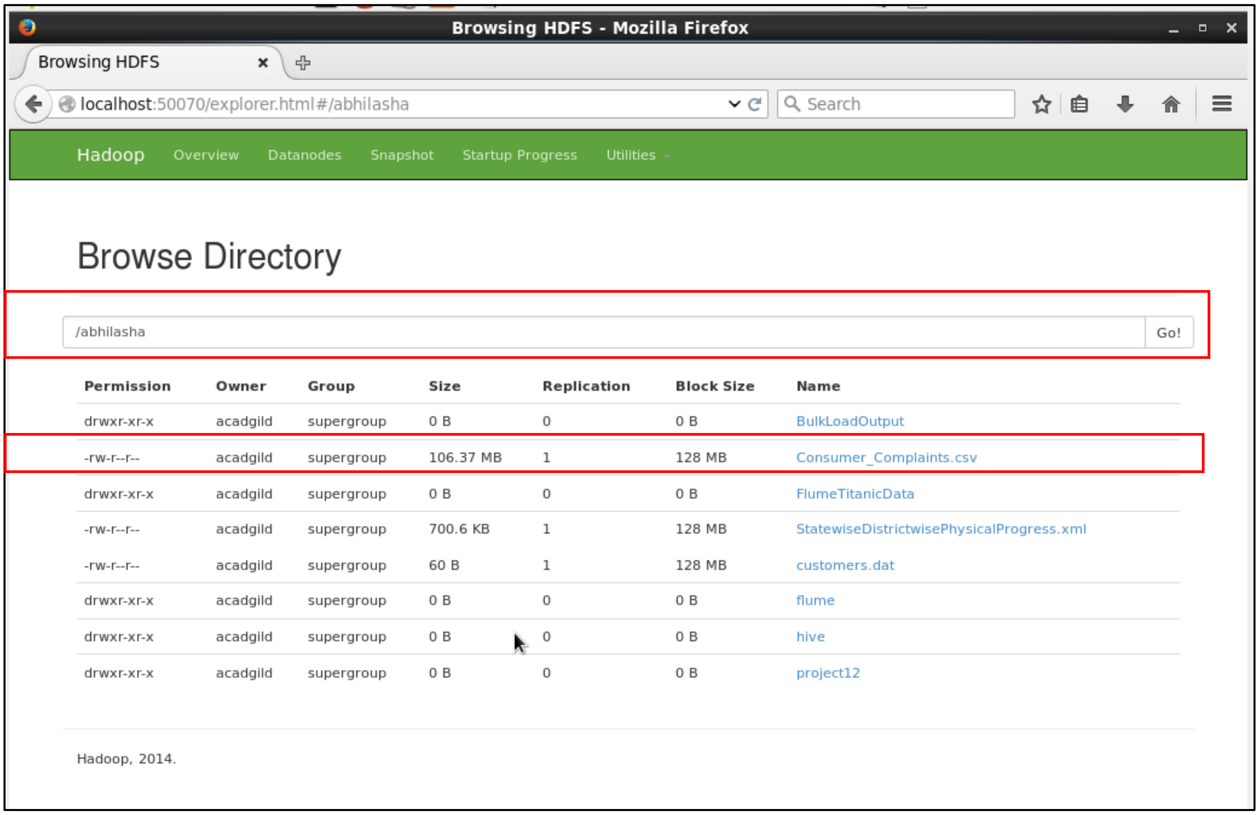
```
[acadgild@localhost Project2.2]$ flume-ng agent -n agent1 -c conf -f /home/acadgild/Abhilasha/Project2.2/filecopy.conf
Info: Including Hadoop libraries found via (/usr/local/hadoop-2.6.0/bin/hadoop) for HDFS access
Info: Excluding /usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-api-1.7.5.jar from classpath
Info: Excluding /usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar from classpath
Info: Including HBASE libraries found via (/usr/local/hbase/bin/hbase) for HBASE access
Info: Excluding /usr/local/hbase/lib/slf4j-api-1.6.4.jar from classpath
Info: Excluding /usr/local/hbase/lib/slf4j-log4j12-1.6.4.jar from classpath
Info: Excluding /usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-api-1.7.5.jar from classpath
Info: Excluding /usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar from classpath
```

After the successful execution of this command, we see the input file placed in HDFS using **ls** command.


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
[acadgild@localhost Project2.2]$ hadoop fs -ls /abhilasha
17/10/28 20:22:52 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 8 items
drwxr-xr-x   - acadgild supergroup          0 2017-10-26 07:43 /abhilasha/BulkLoadOutput
-rw-r--r--   1 acadgild supergroup  111535599 2017-10-28 20:21 /abhilasha/Consumer_Complaints.csv
drwxr-xr-x   - acadgild supergroup          0 2017-08-29 21:58 /abhilasha/FlumeTitanicData
-rw-r--r--   1 acadgild supergroup     717414 2017-10-28 12:55 /abhilasha/StatewiseDistrictwisePhysicalProgress.xml
-rw-r--r--   1 acadgild supergroup         60 2017-10-20 15:43 /abhilasha/customers.dat
drwxr-xr-x   - acadgild supergroup          0 2017-09-05 16:52 /abhilasha/flume
drwxr-xr-x   - acadgild supergroup          0 2017-09-17 15:18 /abhilasha/hive
drwxr-xr-x   - acadgild supergroup          0 2017-08-29 22:53 /abhilasha/project12
[acadgild@localhost Project2.2]$ 
```

We can also see it through the HDFS UI as follows:



Browse Directory

/abhilasha

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	acadgild	supergroup	0 B	0	0 B	BulkLoadOutput
-rw-r--r--	acadgild	supergroup	106.37 MB	1	128 MB	Consumer_Complaints.csv
drwxr-xr-x	acadgild	supergroup	0 B	0	0 B	FlumeTitanicData
-rw-r--r--	acadgild	supergroup	700.6 KB	1	128 MB	StatewiseDistrictwisePhysicalProgress.xml
-rw-r--r--	acadgild	supergroup	60 B	1	128 MB	customers.dat
drwxr-xr-x	acadgild	supergroup	0 B	0	0 B	flume
drwxr-xr-x	acadgild	supergroup	0 B	0	0 B	hive
drwxr-xr-x	acadgild	supergroup	0 B	0	0 B	project12

Hadoop, 2014.

Task 1: Write a pig script to find no of complaints which got timely response

Answer:

The fields from the data that we are going to use are *Complaint ID* and *Timely Response* flag.

The pig script that we are using to solve the problem statement has the following steps:

Step 1: *REGISTER '/usr/local/pig/lib/piggybank.jar';*

We are going to use *CSVExcelStorage* to read data from the csv file. In order to use this, which is present in *piggybank.jar,* we register this jar.

We mention the full qualified local path of the jar, which is */usr/local/pig/lib/piggybank.jar* in this case.

Step 2*: complaintDetails = LOAD '/abhilasha/Consumer_Complaints.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER') AS (dateRec:chararray, product:chararray, subProduct:chararray, issue:chararray, subIssue:chararray, complaintNarrative:chararray, companyPublicResponse:chararray, company:chararray, state:chararray, zipCode:chararray, submittedVia:chararray, dateSentToCompany:chararray, companyResponseToConsumer:chararray, timelyResponse:chararray, consumerDisputed:chararray, complaintId:chararray);*

This command is to specify the details of the input file, schema of the data if known, to be used to load the data.

Here, the file path given is */abhilasha/Consumer_Complaints.csv*. This is the location in HDFS. Rest is the schema of the input, as we have the schema beforehand.

Step 3: *complaintsWithTimelyResponse = FILTER complaintDetails BY timelyResponse == 'Yes';*

As we need records that were given timely response, we use *FILTER* to do so. The clause used to identify complaints with timely response is *timelyResponse == 'Yes'.*

Step 4: *distinctComplaints = GROUP complaintsWithTimelyResponse BY complaintId;*

This command is used to get rid of the redundant records. It is found that the data in the input file had a pinch of duplicity. Hence, this is the measure taken to avoid duplicate complaint IDs in the data. So, we group the data by complaint ID.

Step 5: *complaintIds = FOREACH distinctComplaints GENERATE group AS complaintID;*

Of all the fields in the data, now only complaint ID remains of use to us. Hence, we extract only the complaint ID from the records in the previous step.

Step 6: *groupForCount = GROUP complaintIds ALL;*

This step is to aid us in getting the total count of complaints. This step will get all the complaint IDs in a single group.

Step 7: *count = FOREACH groupForCount GENERATE COUNT(complaintIds);*

This is used to get the count of complaints grouped in the previous step.

Step 8: *STORE count INTO '/abhilasha/Project2.2.Task1';*

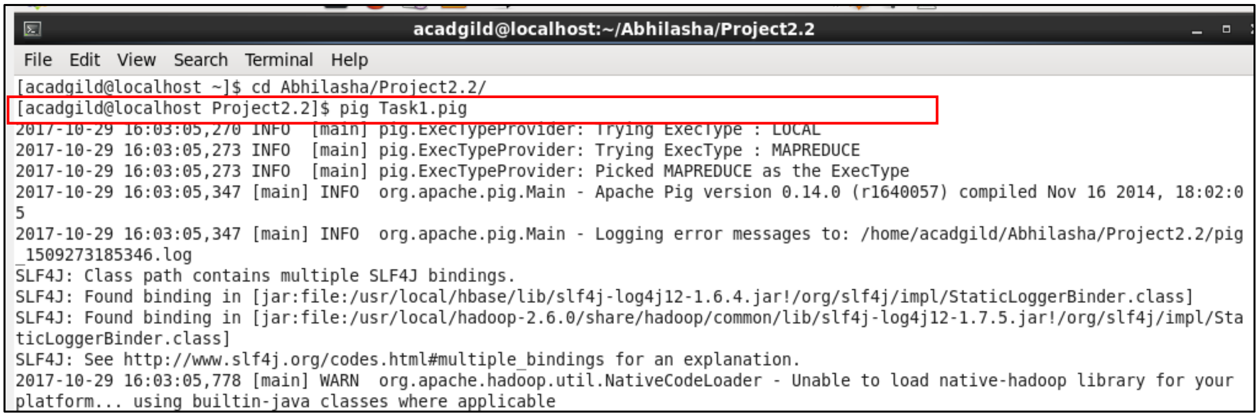
This is to store the data back into HDFS. The destination path given is */abhilasha/Project2.2.Task1*.

All these commands are put together in a file named *Task1* stored at */home/acadgild/Abhilasha/Project2.2*.

This script file is executed as follows:

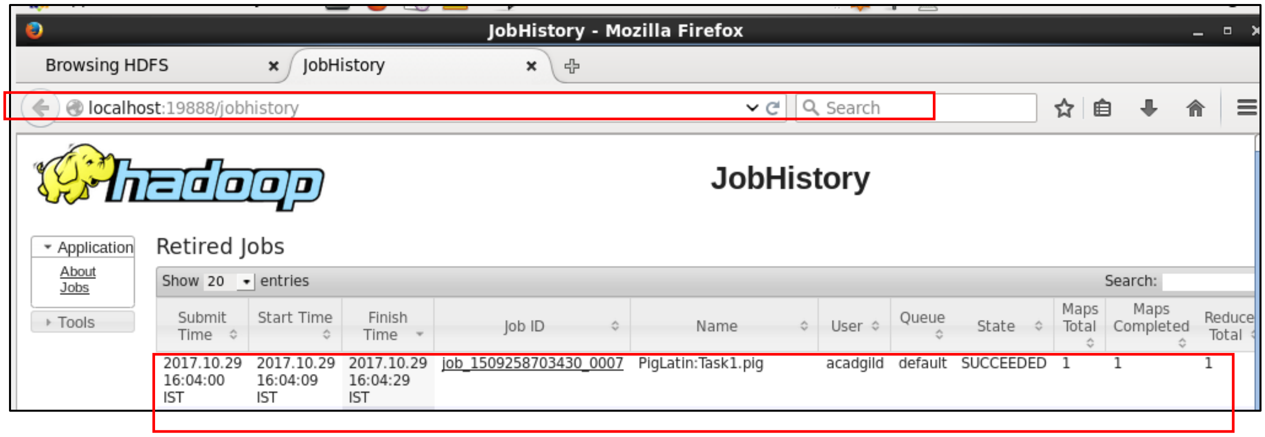
The command used is *pig Task1.pig*

The command indicates that the script will be run not locally but will use HDFS to read and write data.


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
[acadgild@localhost ~]$ cd Abhilasha/Project2.2/
[acadgild@localhost Project2.2]$ pig Task1.pig
2017-10-29 16:03:05,270 INFO  [main] pig.ExecTypeProvider: Trying ExecType : LOCAL
2017-10-29 16:03:05,273 INFO  [main] pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2017-10-29 16:03:05,273 INFO  [main] pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2017-10-29 16:03:05,347 [main] INFO  org.apache.pig.Main - Apache Pig version 0.14.0 (r1640057) compiled Nov 16 2014, 18:02:0
5
2017-10-29 16:03:05,347 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/acadgild/Abhilasha/Project2.2/pig
_1509273185346.log
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase/lib/slf4j-log4j12-1.6.4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Sta
ticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
2017-10-29 16:03:05,778 [main] WARN  org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your
platform... using builtin-java classes where applicable
```

The underlying map-reduce job can be seen successfully completed in the job history server as follows:

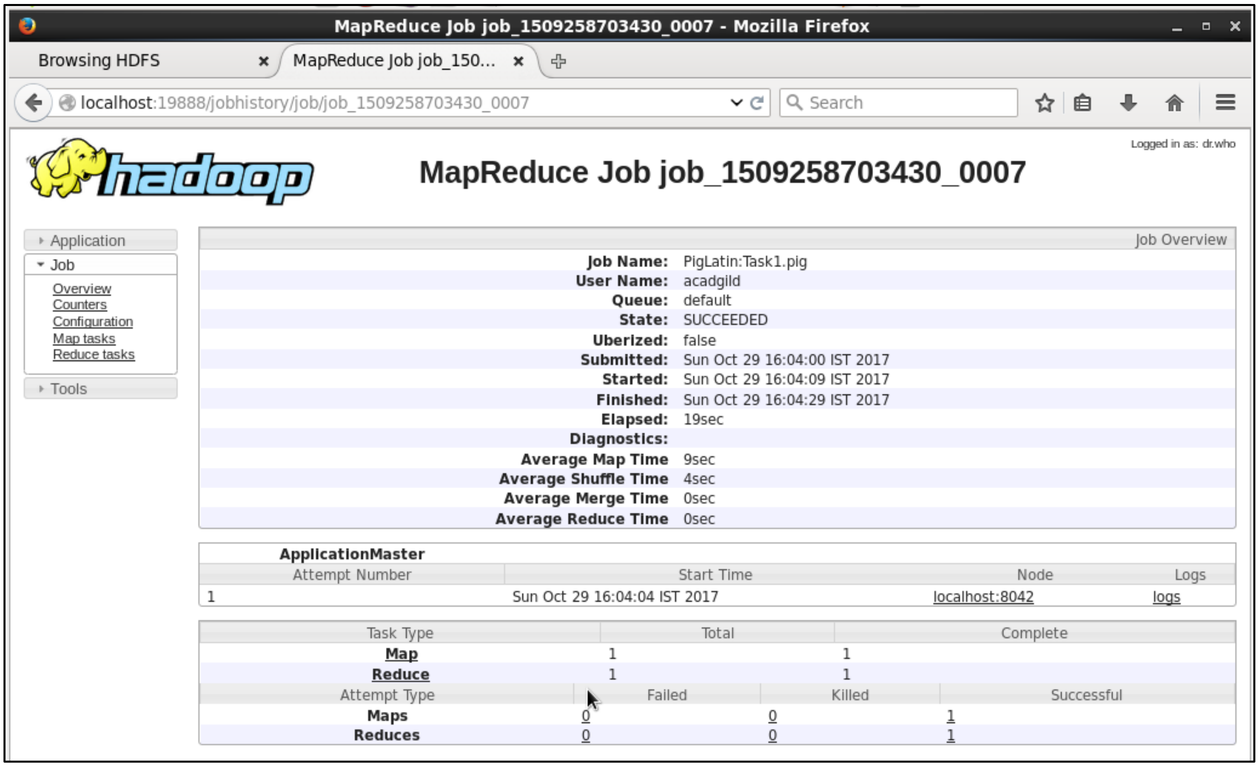


JobHistory

Retired Jobs

Submit Time	Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduce Total
2017.10.29 16:04:00 IST	2017.10.29 16:04:09 IST	2017.10.29 16:04:29 IST	job_1509258703430_0007	PigLatin:Task1.pig	acadgild	default	SUCCEEDED	1	1	1

Details of job execution can be seen below:



MapReduce Job job_1509258703430_0007

Job Overview

Job Name:	PigLatin:Task1.pig
User Name:	acadgild
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Sun Oct 29 16:04:00 IST 2017
Started:	Sun Oct 29 16:04:09 IST 2017
Finished:	Sun Oct 29 16:04:29 IST 2017
Elapsed:	19sec
Diagnostics:	
Average Map Time	9sec
Average Shuffle Time	4sec
Average Merge Time	0sec
Average Reduce Time	0sec

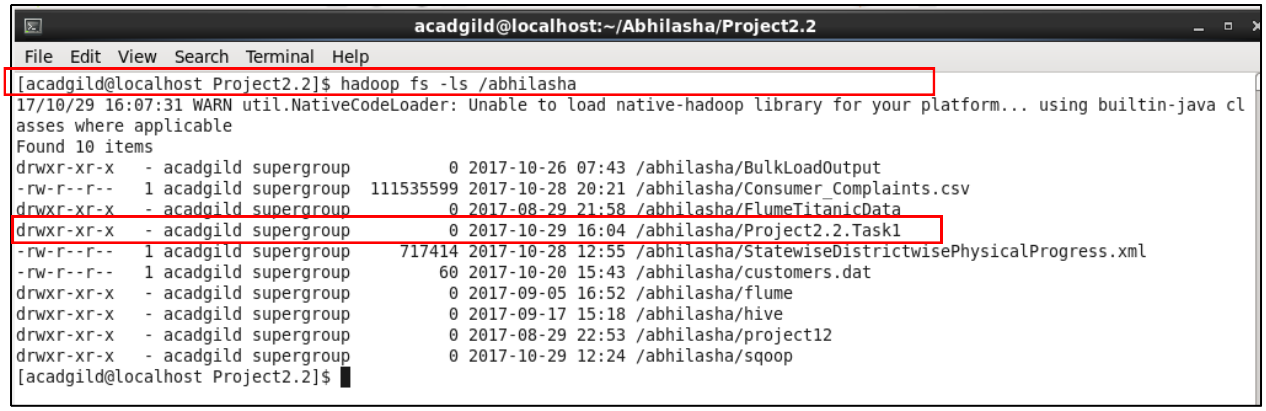
ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Sun Oct 29 16:04:04 IST 2017	localhost:8042	logs

Task Type	Total	Complete
Map	1	1
Reduce	1	1

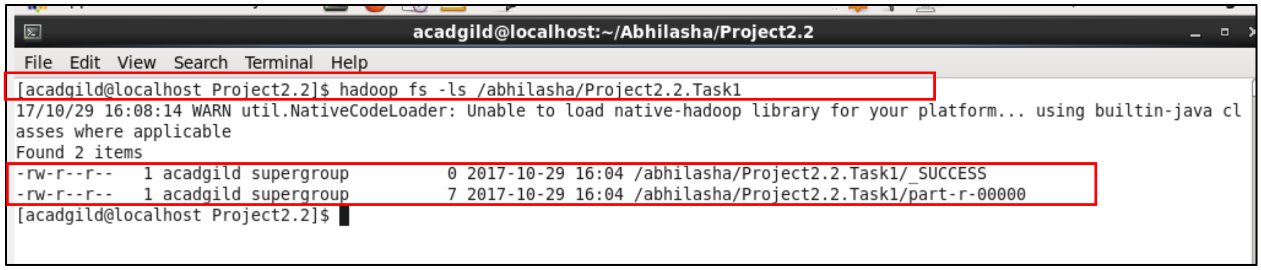
Attempt Type	Failed	Killed	Successful
Maps	0	0	1
Reduces	0	0	1

The output path mentioned in the script was */abhilasha/Project2.2.Task1* in HDFS. We can see the output folder that got created as a result of the job execution as follows:


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

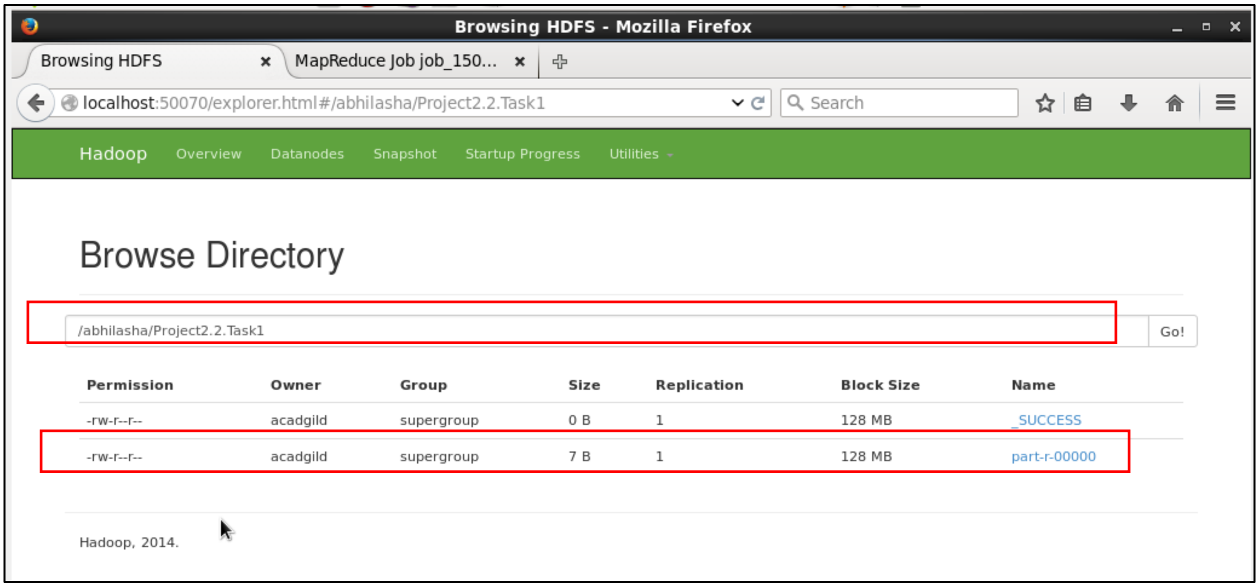
```
[acadgild@localhost Project2.2]$ hadoop fs -ls /abhilasha
17/10/29 16:07:31 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 10 items
drwxr-xr-x   - acadgild supergroup          0 2017-10-26 07:43 /abhilasha/BulkLoadOutput
-rw-r--r--   1 acadgild supergroup  111535599 2017-10-28 20:21 /abhilasha/Consumer_Complaints.csv
drwxr-xr-x   - acadgild supergroup          0 2017-08-29 21:58 /abhilasha/FlumeTitanicData
drwxr-xr-x   - acadgild supergroup          0 2017-10-29 16:04 /abhilasha/Project2.2.Task1
-rw-r--r--   1 acadgild supergroup     717414 2017-10-28 12:55 /abhilasha/StatewiseDistrictwisePhysicalProgress.xml
-rw-r--r--   1 acadgild supergroup         60 2017-10-20 15:43 /abhilasha/customers.dat
drwxr-xr-x   - acadgild supergroup          0 2017-09-05 16:52 /abhilasha/flume
drwxr-xr-x   - acadgild supergroup          0 2017-09-17 15:18 /abhilasha/hive
drwxr-xr-x   - acadgild supergroup          0 2017-08-29 22:53 /abhilasha/project12
drwxr-xr-x   - acadgild supergroup          0 2017-10-29 12:24 /abhilasha/sqoop
[acadgild@localhost Project2.2]$
```

This folder contains the files mentioned in the screen shot below. Of these *part-r-00000* file contains the output of the pig script.


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
[acadgild@localhost Project2.2]$ hadoop fs -ls /abhilasha/Project2.2.Task1
17/10/29 16:08:14 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 2 items
-rw-r--r--   1 acadgild supergroup          0 2017-10-29 16:04 /abhilasha/Project2.2.Task1/_SUCCESS
-rw-r--r--   1 acadgild supergroup          7 2017-10-29 16:04 /abhilasha/Project2.2.Task1/part-r-00000
[acadgild@localhost Project2.2]$
```

These files can also be viewed from HDFS UI as shown below:

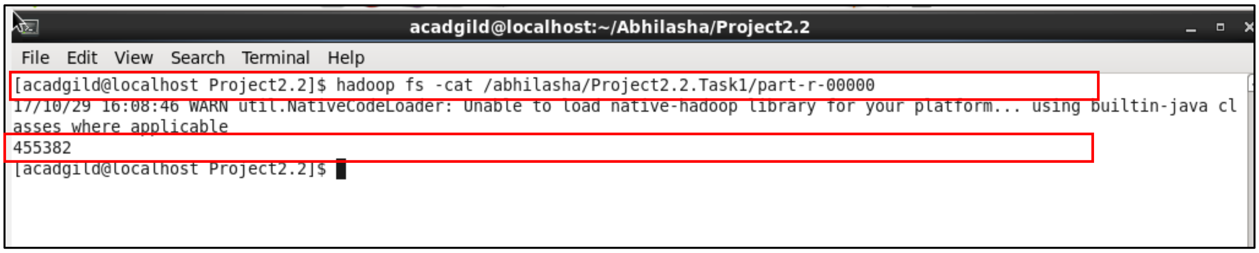


Browse Directory

/abhilasha/Project2.2.Task1

Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	acadgild	supergroup	0 B	1	128 MB	_SUCCESS
-rw-r--r--	acadgild	supergroup	7 B	1	128 MB	part-r-00000

The content of this output file can be seen using the **cat** command as follows:

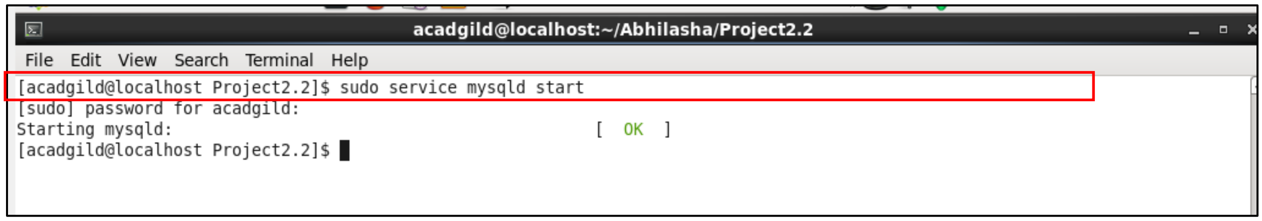

acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
[acadgild@localhost Project2.2]$ hadoop fs -cat /abhilasha/Project2.2.Task1/part-r-00000
17/10/29 16:08:46 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
455382
[acadgild@localhost Project2.2]$ 
```

Note: To upload the output file in GitHub, we have renamed the file to *part-r-000001.*

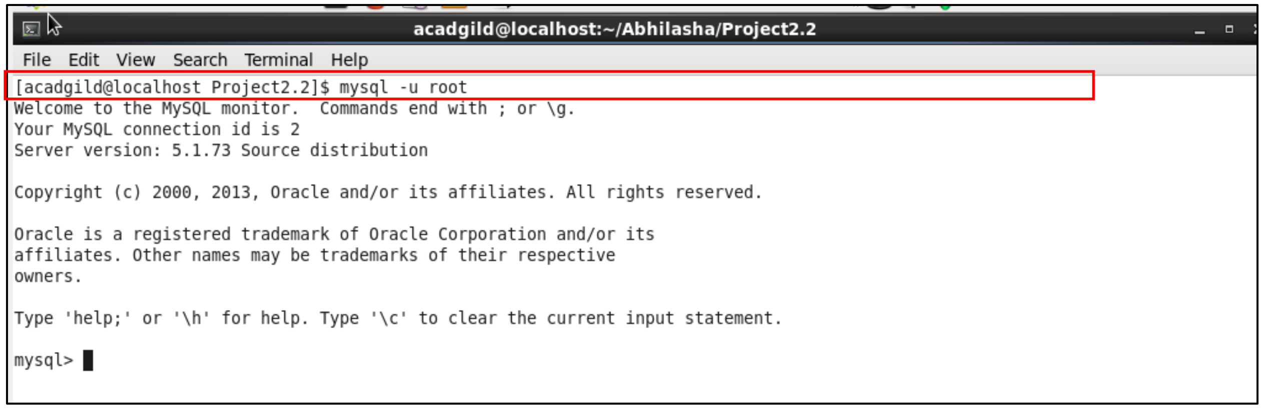
Now that the output is placed in HDFS, we need to store the output in MySQL. To do so, we perform the following steps

Step 1: Start MySQL using the command
sudo service mysqld start


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

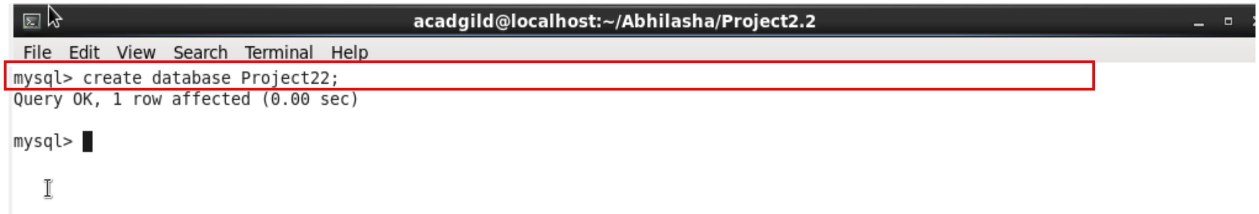
```
[acadgild@localhost Project2.2]$ sudo service mysqld start
[sudo] password for acadgild:
Starting mysqld:                                           [  OK  ]
[acadgild@localhost Project2.2]$ 
```

Step 2: To start the command line interface of MySQL, we use the command
mysql -u root


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
[acadgild@localhost Project2.2]$ mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
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> 
```

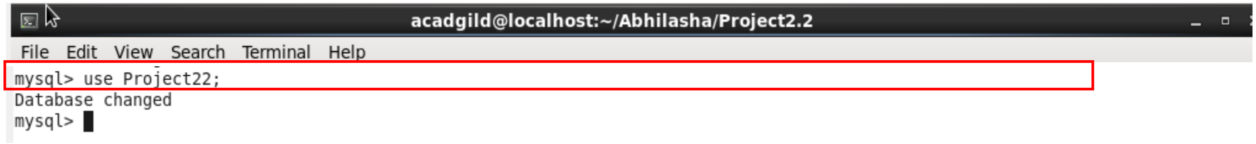
Step 3: We create a new database named Project22 using the command
create database Project22;


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
mysql> create database Project22;
Query OK, 1 row affected (0.00 sec)

mysql> 
```

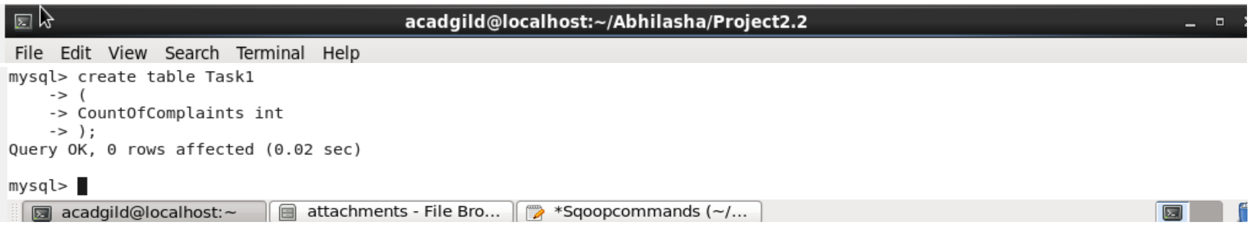
Step 4: We change the database from default to Project22, we use the command

use Project22;


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
mysql> use Project22;
Database changed
mysql> 
```

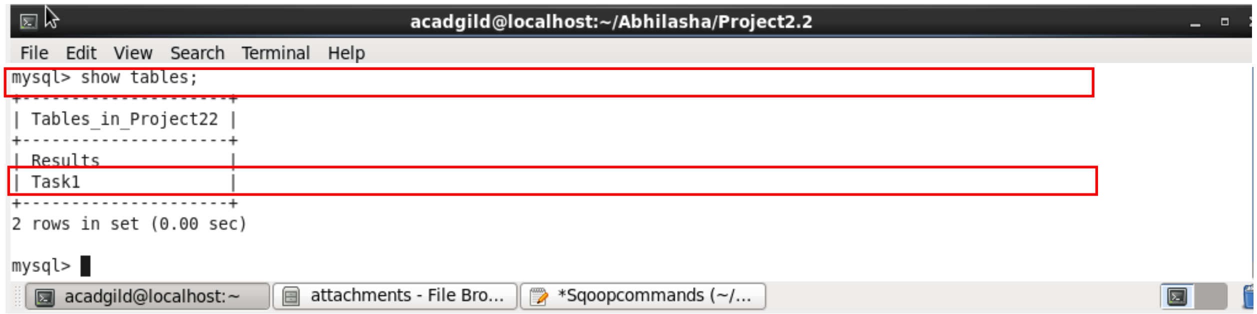
Step 5: We now create table named *Task1* to store the output of Task 1 in it using the command
create table Task1
(
CountOfComplaints int
);
Here, the column of the table is *CountOfComplaints.*


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help
acadgild@localhost:~
attachments - File Bro...
*Sqoopcommands (~/...

```
mysql> create table Task1
    -> (
    -> CountOfComplaints int
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> 
```

Step 6: The created table can be listed using the command

Show tables;



```
mysql> show tables;
+---------------------+
| Tables_in_Project22 |
+---------------------+
| Results             |
| Task1               |
+---------------------+
2 rows in set (0.00 sec)

mysql> 
```

Step 6: Next step is to export the data from HDFS and store in into the table created in previous step.

sqoop export --connect jdbc:mysql://localhost/Project22 --username 'root' -P --table 'Task1' --export-dir '/abhilasha/Project2.2.Task1/part-r-00000' -m 1 --columns CountOfComplaints;

Here, the parameters mentioned in the above command are:

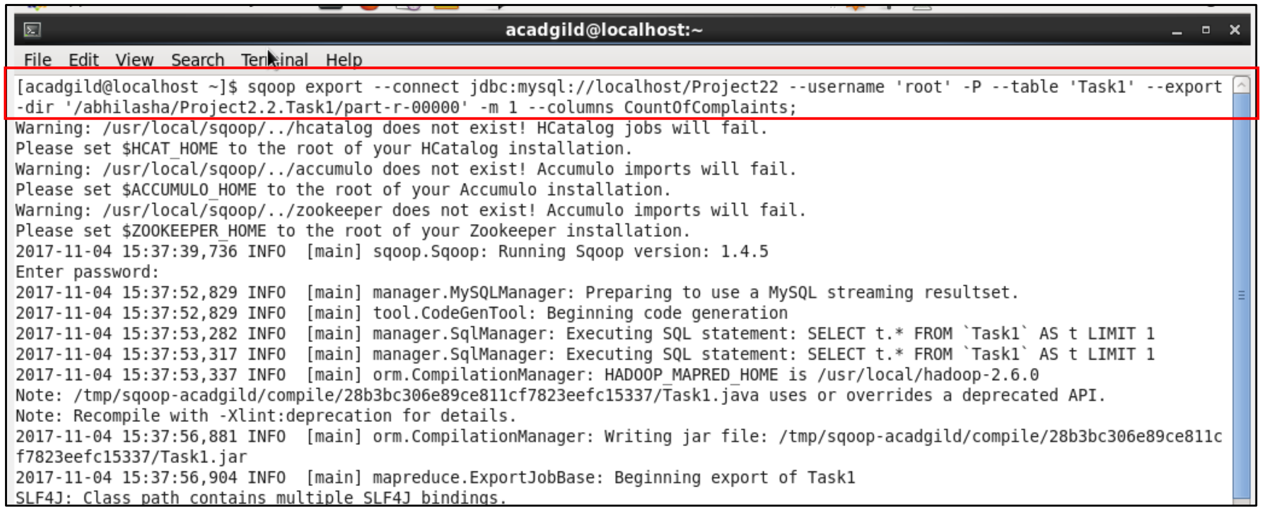
i. Database : jdbc:mysql://localhost/Project22

ii. Table Name : Task1

iii. Path where exported file is placed : /abhilasha/Project2.2.Task1/part-r-00000

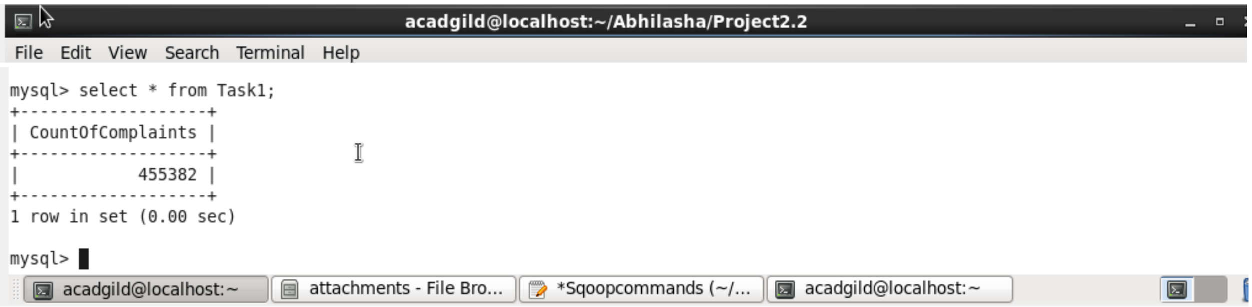
iv. Column to populate : CountOfComplaints

The command execution is shown below:



```
[acadgild@localhost ~]$ sqoop export --connect jdbc:mysql://localhost/Project22 --username 'root' -P --table 'Task1' --export
-dir '/abhilasha/Project2.2.Task1/part-r-00000' -m 1 --columns CountOfComplaints;
Warning: /usr/local/sqoop/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /usr/local/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /usr/local/sqoop/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
2017-11-04 15:37:39,736 INFO  [main] sqoop.Sqoop: Running Sqoop version: 1.4.5
Enter password:
2017-11-04 15:37:52,829 INFO  [main] manager.MySQLManager: Preparing to use a MySQL streaming resultset.
2017-11-04 15:37:52,829 INFO  [main] tool.CodeGenTool: Beginning code generation
2017-11-04 15:37:53,282 INFO  [main] manager.SqlManager: Executing SQL statement: SELECT t.* FROM `Task1` AS t LIMIT 1
2017-11-04 15:37:53,317 INFO  [main] manager.SqlManager: Executing SQL statement: SELECT t.* FROM `Task1` AS t LIMIT 1
2017-11-04 15:37:53,337 INFO  [main] orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/local/hadoop-2.6.0
Note: /tmp/sqoop-acadgild/compile/28b3bc306e89ce811cf7823eefc15337/Task1.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
2017-11-04 15:37:56,881 INFO  [main] orm.CompilationManager: Writing jar file: /tmp/sqoop-acadgild/compile/28b3bc306e89ce811c
f7823eefc15337/Task1.jar
2017-11-04 15:37:56,904 INFO  [main] mapreduce.ExportJobBase: Beginning export of Task1
SLF4J: Class path contains multiple SLF4J bindings.
```

Step 7: After the successful placement of data in MySQL, we can see the content of the table in the database populated, using *select * from Task1;* as follows:


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help
acadgild@localhost:~
attachments - File Bro...
*Sqoopcommands (~/...
acadgild@localhost:~

```
mysql> select * from Task1;
+-------------------+
| CountOfComplaints |
+-------------------+
|            455382 |
+-------------------+
1 row in set (0.00 sec)

mysql> 
```

This shows that the data is placed successfully in database.

Task 2: Write a pig script to find no of complaints where consumer forum forwarded the complaint same day they received to respective company

Answer:

The fields from the data that we are going to use are *Date Received*, *Date sent to company* and *Complaint ID*.

The pig script that we are using to solve the problem statement has the following steps:

Step 1: *REGISTER '/usr/local/pig/lib/piggybank.jar';*

We are going to use *CSVExcelStorage* to read data from the csv file. In order to use this, which is present in *piggybank.jar*, we register this jar.

We mention the full qualified local path of the jar, which is */usr/local/pig/lib/piggybank.jar* in this case.

Step 2*: complaintDetails = LOAD '/abhilasha/Consumer_Complaints.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER') AS (dateRec:chararray, product:chararray, subProduct:chararray, issue:chararray, subIssue:chararray, complaintNarrative:chararray, companyPublicResponse:chararray, company:chararray, state:chararray, zipCode:chararray, submittedVia:chararray, dateSentToCompany:chararray, companyResponseToConsumer:chararray, timelyResponse:chararray, consumerDisputed:chararray, complaintId:chararray);*

This command is to specify the details of the input file, schema of the data if known, to be used to load the data.

Here, the file path given is */abhilasha/Consumer_Complaints.csv*. This is the location in HDFS. Rest is the schema of the input, as we have the schema beforehand.

Step 3: *complaintsRequired = FILTER complaintDetails BY dateRec == dateSentToCompany;*

As we need records where consumer forum forwarded the complaint same day they received to respective company, we use the clause *dateRec == dateSentToCompany* to filter the records.

Step 4: *distinctComplaints = GROUP complaintsRequired BY complaintId;*

This command is used to get rid of the redundant records. It is found that the data in the input file had a pinch of duplicity. Hence, this is the measure taken to avoid duplicate complaint IDs in the data. So, we group the data by complaint ID.

Step 5: *complaintIds = FOREACH distinctComplaints GENERATE group AS complaintID;*

Of all the fields in the data, now only complaint ID remains of use to us. Hence, we extract only the complaint ID from the records in the previous step.

Step 6: *groupForCount = GROUP complaintIds ALL;*

This step is to aid us in getting the total count of complaints. This step will get all the complaint IDs in a single group.

Step 7: *count = FOREACH groupForCount GENERATE COUNT(complaintIds);*

This is used to get the count of complaints grouped in the previous step.

Step 8: *STORE count INTO '/abhilasha/Project2.2.Task2';*

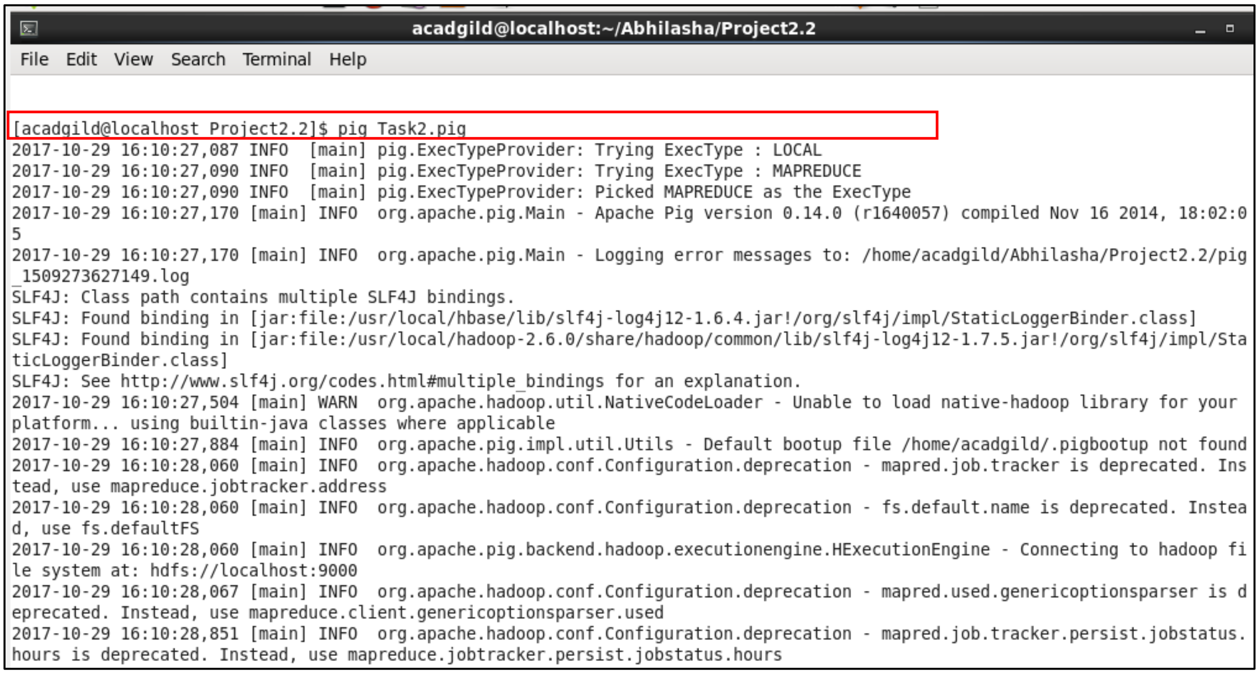
This is to store the data back into HDFS. The destination path given is */abhilasha/Project2.2.Task2.*

All these commands are put together in a file named *Task2* stored at */home/acadgild/Abhilasha/Project2.2*.

This script file is executed as follows:

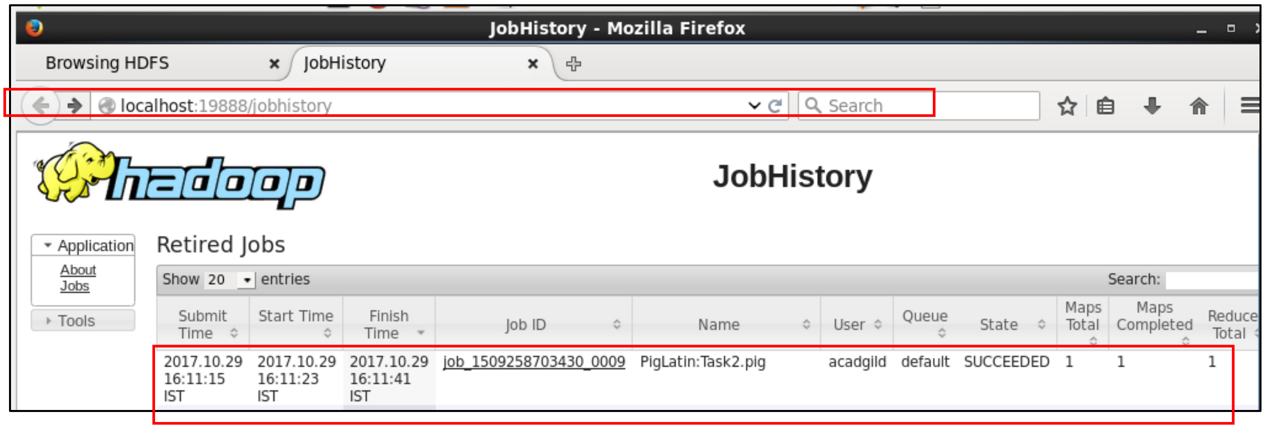
The command used is *pig Task2.pig*

The command indicates that the script will be run not locally but will use HDFS to read and write data.


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
[acadgild@localhost Project2.2]$ pig Task2.pig
2017-10-29 16:10:27,087 INFO  [main] pig.ExecTypeProvider: Trying ExecType : LOCAL
2017-10-29 16:10:27,090 INFO  [main] pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2017-10-29 16:10:27,090 INFO  [main] pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2017-10-29 16:10:27,170 [main] INFO  org.apache.pig.Main - Apache Pig version 0.14.0 (r1640057) compiled Nov 16 2014, 18:02:0
5
2017-10-29 16:10:27,170 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/acadgild/Abhilasha/Project2.2/pig
_1509273627149.log
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase/lib/slf4j-log4j12-1.6.4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Sta
ticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
2017-10-29 16:10:27,504 [main] WARN  org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your
platform... using builtin-java classes where applicable
2017-10-29 16:10:27,884 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file /home/acadgild/.pigbootup not found
2017-10-29 16:10:28,060 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Ins
tead, use mapreduce.jobtracker.address
2017-10-29 16:10:28,060 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instea
d, use fs.defaultFS
2017-10-29 16:10:28,060 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop fi
le system at: hdfs://localhost:9000
2017-10-29 16:10:28,067 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.used.genericoptionsparser is d
eprecated. Instead, use mapreduce.client.genericoptionsparser.used
2017-10-29 16:10:28,851 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker.persist.jobstatus.
hours is deprecated. Instead, use mapreduce.jobtracker.persist.jobstatus.hours
```

The underlying map-reduce job can be seen successfully completed in the job history server as follows:

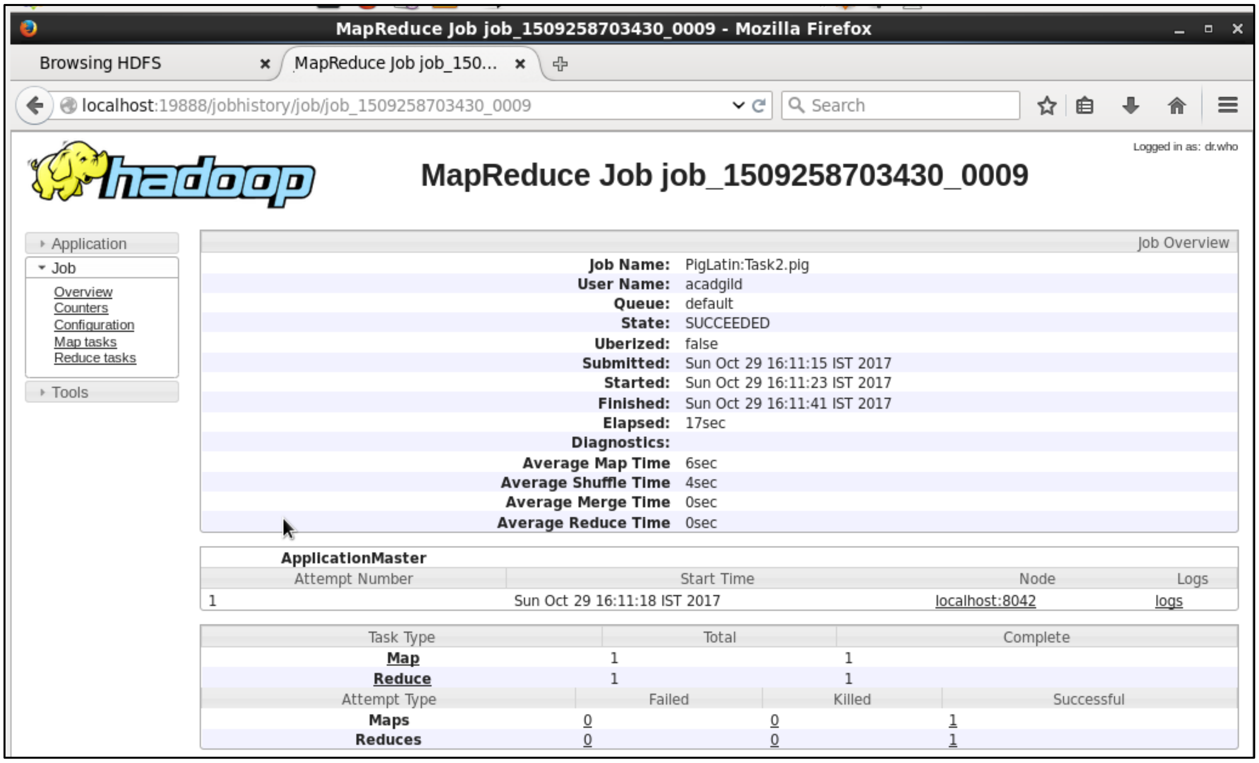


Retired Jobs

Show 20 entries Search:

Submit Time	Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduce Total
2017.10.29 16:11:15 IST	2017.10.29 16:11:23 IST	2017.10.29 16:11:41 IST	job_1509258703430_0009	PigLatin:Task2.pig	acadgild	default	SUCCEEDED	1	1	1

Details of job execution can be seen below:



MapReduce Job job_1509258703430_0009

Job Overview

Job Name:	PigLatin:Task2.pig
User Name:	acadgild
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Sun Oct 29 16:11:15 IST 2017
Started:	Sun Oct 29 16:11:23 IST 2017
Finished:	Sun Oct 29 16:11:41 IST 2017
Elapsed:	17sec
Diagnostics:	
Average Map Time	6sec
Average Shuffle Time	4sec
Average Merge Time	0sec
Average Reduce Time	0sec

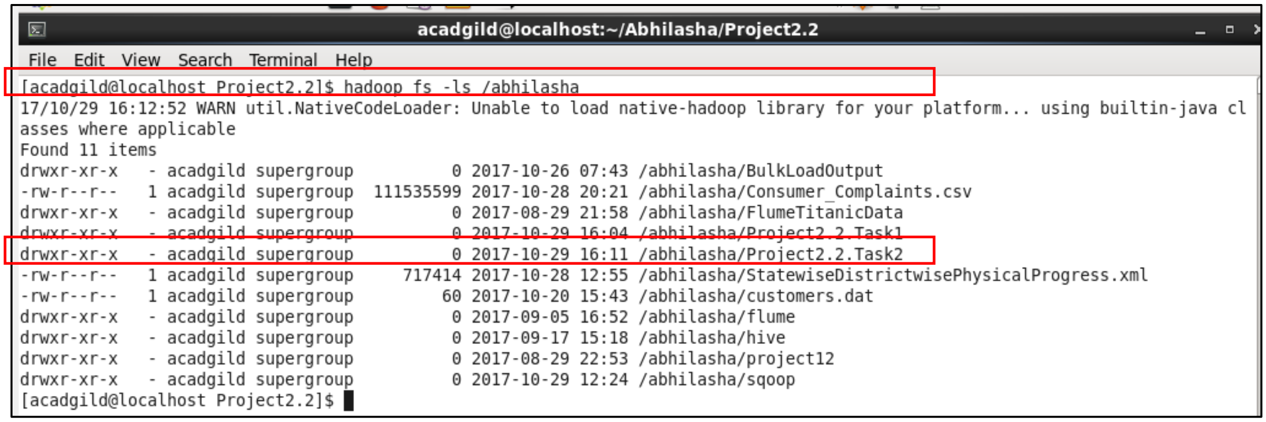
ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Sun Oct 29 16:11:18 IST 2017	localhost:8042	logs

Task Type	Total	Complete
Map	1	1
Reduce	1	1

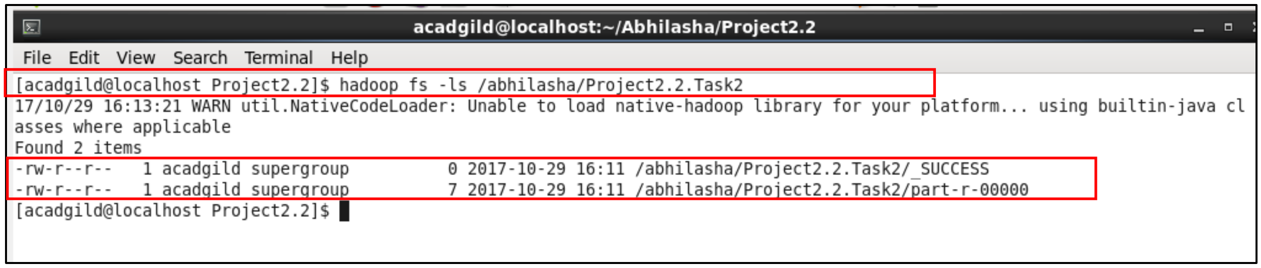
Attempt Type	Failed	Killed	Successful
Maps	0	0	1
Reduces	0	0	1

The output path mentioned in the script was */abhilasha/Project2.2.Task2* in HDFS. We can see the output folder that got created as a result of the job execution as follows:


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

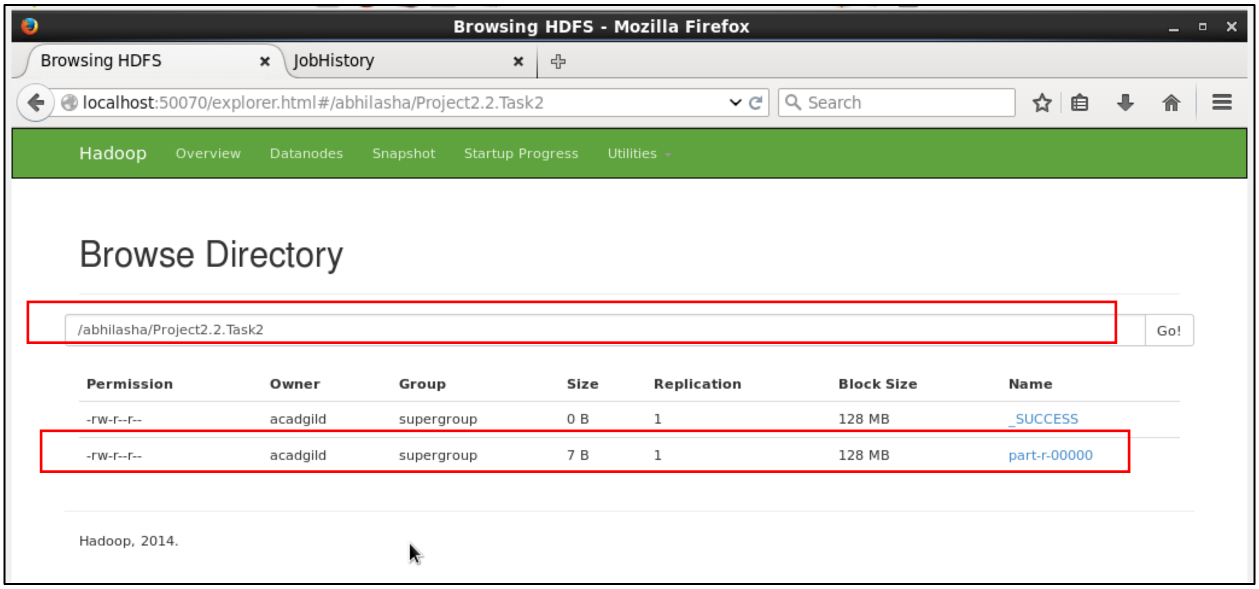
```
[acadgild@localhost Project2.2]$ hadoop fs -ls /abhilasha
17/10/29 16:12:52 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 11 items
drwxr-xr-x   - acadgild supergroup          0 2017-10-26 07:43 /abhilasha/BulkLoadOutput
-rw-r--r--   1 acadgild supergroup  111535599 2017-10-28 20:21 /abhilasha/Consumer_Complaints.csv
drwxr-xr-x   - acadgild supergroup          0 2017-08-29 21:58 /abhilasha/FlumeTitanicData
drwxr-xr-x   - acadgild supergroup          0 2017-10-29 16:04 /abhilasha/Project2.2.Task1
drwxr-xr-x   - acadgild supergroup          0 2017-10-29 16:11 /abhilasha/Project2.2.Task2
-rw-r--r--   1 acadgild supergroup     717414 2017-10-28 12:55 /abhilasha/StatewiseDistrictwisePhysicalProgress.xml
-rw-r--r--   1 acadgild supergroup         60 2017-10-20 15:43 /abhilasha/customers.dat
drwxr-xr-x   - acadgild supergroup          0 2017-09-05 16:52 /abhilasha/flume
drwxr-xr-x   - acadgild supergroup          0 2017-09-17 15:18 /abhilasha/hive
drwxr-xr-x   - acadgild supergroup          0 2017-08-29 22:53 /abhilasha/project12
drwxr-xr-x   - acadgild supergroup          0 2017-10-29 12:24 /abhilasha/sqoop
[acadgild@localhost Project2.2]$
```

This folder contains the files mentioned in the screen shot below. Of these *part-r-00000* file contains the output of the pig script.


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
[acadgild@localhost Project2.2]$ hadoop fs -ls /abhilasha/Project2.2.Task2
17/10/29 16:13:21 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 2 items
-rw-r--r--   1 acadgild supergroup          0 2017-10-29 16:11 /abhilasha/Project2.2.Task2/_SUCCESS
-rw-r--r--   1 acadgild supergroup          7 2017-10-29 16:11 /abhilasha/Project2.2.Task2/part-r-00000
[acadgild@localhost Project2.2]$
```

These files can also be viewed from HDFS UI as shown below:


Browsing HDFS - Mozilla Firefox
Browsing HDFS
JobHistory
localhost:50070/explorer.html#/abhilasha/Project2.2.Task2
Search
Hadoop
Overview
Datanodes
Snapshot
Startup Progress
Utilities
Go!

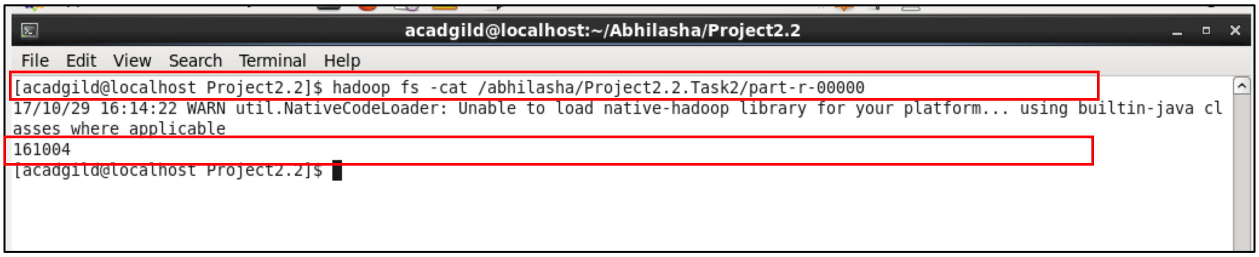
Browse Directory

/abhilasha/Project2.2.Task2

Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	acadgild	supergroup	0 B	1	128 MB	_SUCCESS
-rw-r--r--	acadgild	supergroup	7 B	1	128 MB	part-r-00000

Hadoop, 2014.

The content of this output file can be seen using the **cat** command as follows:


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
[acadgild@localhost Project2.2]$ hadoop fs -cat /abhilasha/Project2.2.Task2/part-r-00000
17/10/29 16:14:22 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
161004
[acadgild@localhost Project2.2]$ 
```

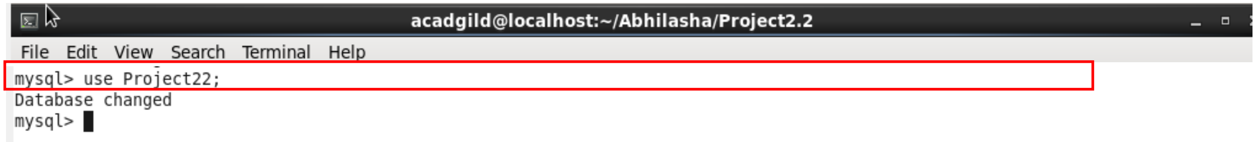
Note: To upload the output file in GitHub, we have renamed the file to *part-r-000002.*

Now that the output is placed in HDFS, we need to store the output in MySQL. To do so, we perform the following steps:

We have already seen steps to start MySQL and create database *Project22.* So not mentioning those steps again.

Step 1: As seen previously, we have created the database named *Project22.* We change the database from default to Project22, we use the command

use Project22;

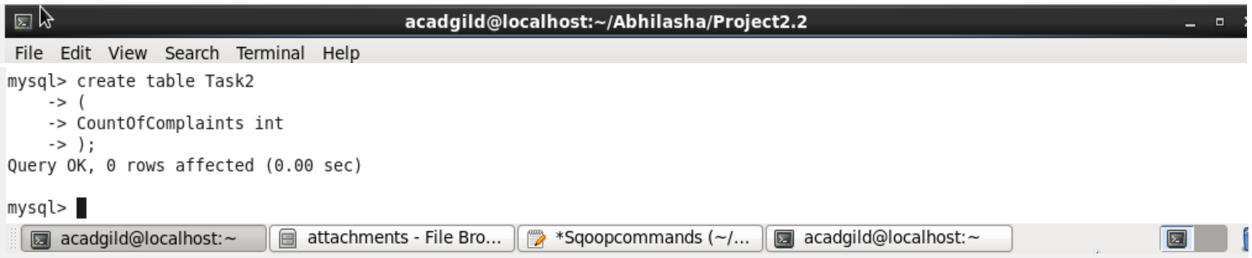

acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
mysql> use Project22;
Database changed
mysql> 
```

Step 5: We now create table named *Task2* to store the output of Task 2 in it using the command

create table Task2
(
    CountOfComplaints int
);

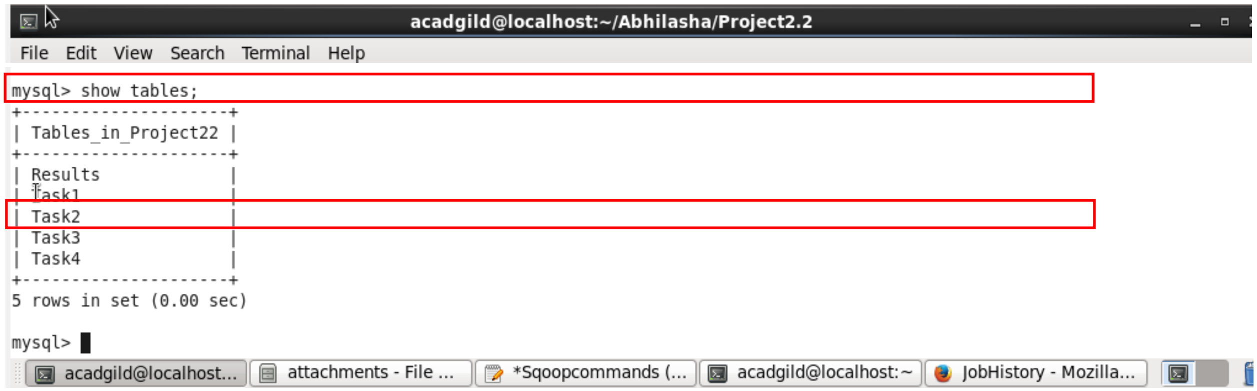
Here, the column of the table is *CountOfComplaints.*


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help
acadgild@localhost:~
attachments - File Bro...
*Sqoopcommands (~/...
acadgild@localhost:~

```
mysql> create table Task2
    -> (
    -> CountOfComplaints int
    -> );
Query OK, 0 rows affected (0.00 sec)

mysql> 
```

Step 6: The created table can be listed using the command

Show tables;



```
mysql> show tables;
+---------------------+
| Tables_in_Project22 |
+---------------------+
| Results             |
| Task1               |
| Task2               |
| Task3               |
| Task4               |
+---------------------+
5 rows in set (0.00 sec)

mysql> 
```

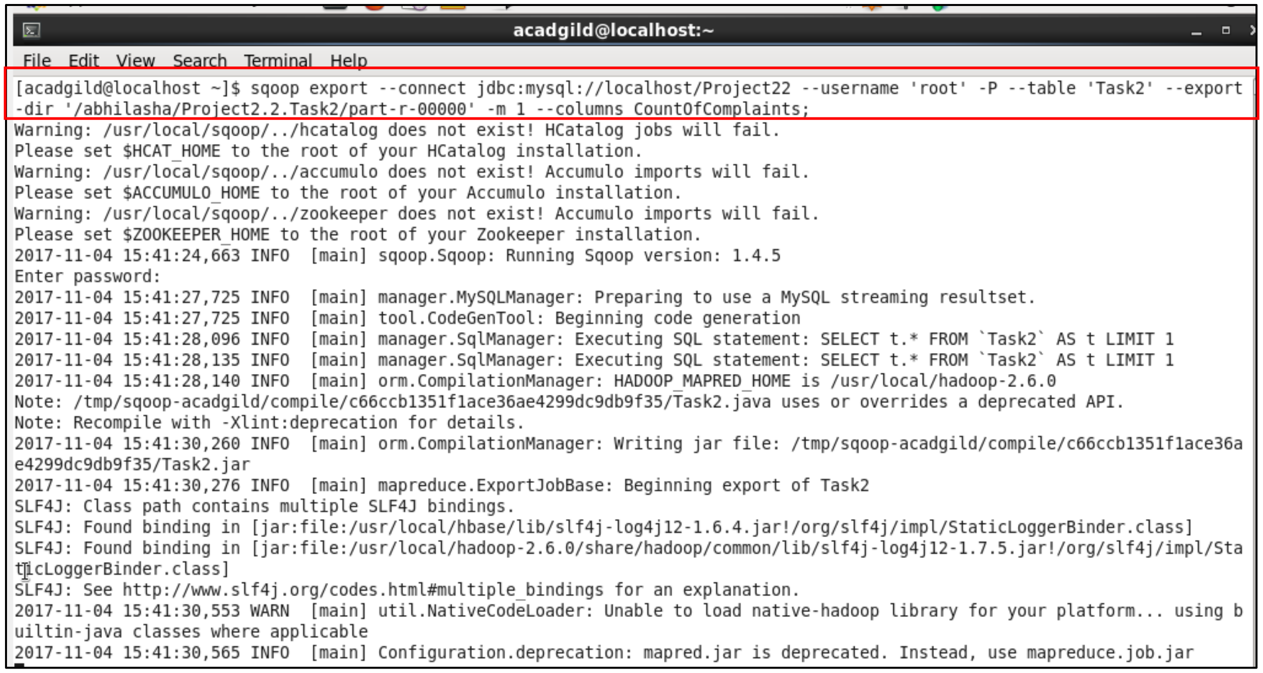
Step 6: Next step is to export the data from HDFS and store in into the table created in previous step.

sqoop export --connect jdbc:mysql://localhost/Project22 --username 'root' -P --table 'Task2' --export-dir '/abhilasha/Project2.2.Task2/part-r-00000' -m 1 --columns CountOfComplaints;

Here, the parameters mentioned in the above command are:

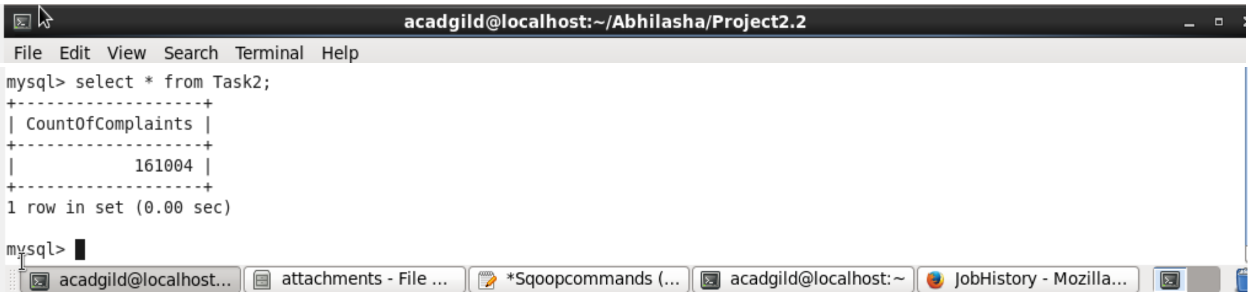
i. Database : jdbc:mysql://localhost/Project22
ii. Table Name : Task2
iii. Path where exported file is placed : /abhilasha/Project2.2.Task2/part-r-00000
iv. Column to populate : CountOfComplaints

The command execution is shown below:



```
[acadgild@localhost ~]$ sqoop export --connect jdbc:mysql://localhost/Project22 --username 'root' -P --table 'Task2' --export
-dir '/abhilasha/Project2.2.Task2/part-r-00000' -m 1 --columns CountOfComplaints;
Warning: /usr/local/sqoop/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /usr/local/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /usr/local/sqoop/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
2017-11-04 15:41:24,663 INFO  [main] sqoop.Sqoop: Running Sqoop version: 1.4.5
Enter password:
2017-11-04 15:41:27,725 INFO  [main] manager.MySQLManager: Preparing to use a MySQL streaming resultset.
2017-11-04 15:41:27,725 INFO  [main] tool.CodeGenTool: Beginning code generation
2017-11-04 15:41:28,096 INFO  [main] manager.SqlManager: Executing SQL statement: SELECT t.* FROM `Task2` AS t LIMIT 1
2017-11-04 15:41:28,135 INFO  [main] manager.SqlManager: Executing SQL statement: SELECT t.* FROM `Task2` AS t LIMIT 1
2017-11-04 15:41:28,140 INFO  [main] orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/local/hadoop-2.6.0
Note: /tmp/sqoop-acadgild/compile/c66ccb1351f1ace36ae4299dc9db9f35/Task2.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
2017-11-04 15:41:30,260 INFO  [main] orm.CompilationManager: Writing jar file: /tmp/sqoop-acadgild/compile/c66ccb1351f1ace36a
e4299dc9db9f35/Task2.jar
2017-11-04 15:41:30,276 INFO  [main] mapreduce.ExportJobBase: Beginning export of Task2
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase/lib/slf4j-log4j12-1.6.4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Sta
ticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
2017-11-04 15:41:30,553 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using b
uiltin-java classes where applicable
2017-11-04 15:41:30,565 INFO  [main] Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar
```

Step 7: After the successful placement of data in MySQL, we can see the content of the table in the database populated, using *select * from Task2;* as follows:



```
mysql> select * from Task2;
+-------------------+
| CountOfComplaints |
+-------------------+
|            161004 |
+-------------------+
1 row in set (0.00 sec)

mysql> 
```

This shows that the data is placed successfully in database.

Task 3: Write a pig script to find list of companies toping in complaint chart (companies with maximum number of complaints).

Answer:

The fields from the data that we are going to use are *company* and *Complaint ID.*

The pig script that we are using to solve the problem statement has the following steps:

Step 1: *REGISTER '/usr/local/pig/lib/piggybank.jar';*

We are going to use *CSVExcelStorage* to read data from the csv file. In order to use this, which is present in *piggybank.jar*, we register this jar.

We mention the full qualified local path of the jar, which is */usr/local/pig/lib/piggybank.jar* in this case.

Step 2*: complaintDetails = LOAD '/abhilasha/Consumer_Complaints.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER') AS (dateRec:chararray, product:chararray, subProduct:chararray, issue:chararray, subIssue:chararray, complaintNarrative:chararray, companyPublicResponse:chararray, company:chararray, state:chararray, zipCode:chararray, submittedVia:chararray, dateSentToCompany:chararray, companyResponseToConsumer:chararray, timelyResponse:chararray, consumerDisputed:chararray, complaintId:chararray);*

This command is to specify the details of the input file, schema of the data if known, to be used to load the data.

Here, the file path given is */abhilasha/Consumer_Complaints.csv*. This is the location in HDFS. Rest is the schema of the input, as we have the schema beforehand.

Step 3: *companywiseComplaints = GROUP complaintDetails BY company;*

Here, we group the records by company.

Step 4: *companywiseCount = FOREACH companywiseComplaints GENERATE group AS companyName, COUNT(complaintDetails.complaintId) AS countOfComplaints;*

After the grouping of records is done by company, we find the count of records per group. We have used the alias *countOfComplaints* to store the count. This is done for every company in *companywiseComplaints* and hence, used *FOREACH.*

Step 5: *orderByCount = ORDER companywiseCount BY countOfComplaints DESC;*

We now order the records in descending order of the count of complaints.

Step 6: *companyWithMaxComplaints = LIMIT orderByCount 1;*

We need the company with highest count of complaints only. Hence, we limit the data to 1 record.

Step 7: *STORE companyWithMaxComplaints INTO '/abhilasha/Project2.2.Task3' USING PigStorage('|');*

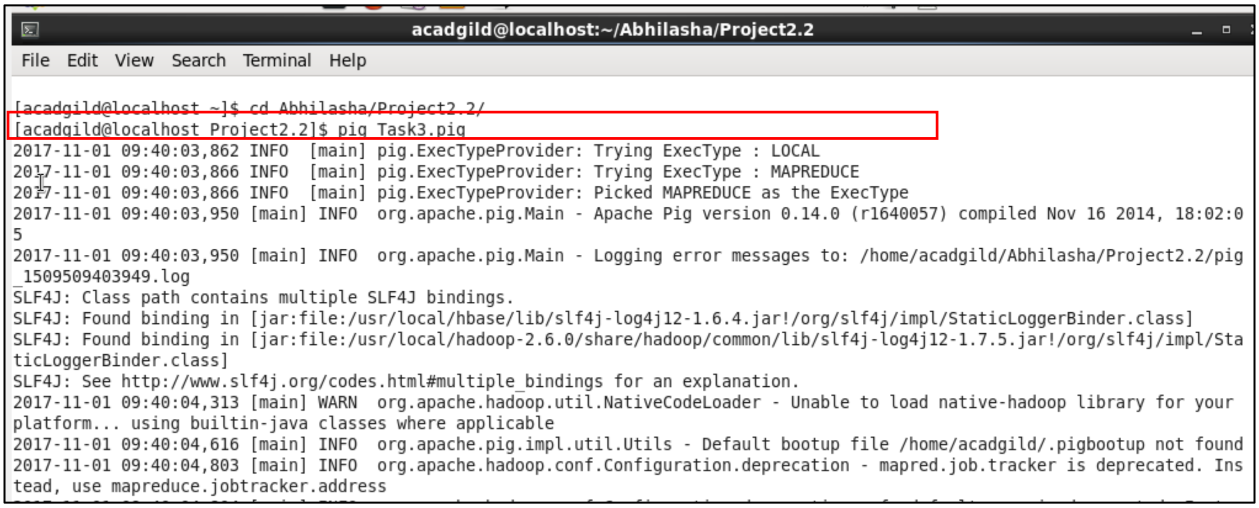
This is to store the data back into HDFS. The destination path given is */abhilasha/Project2.2.Task3.* Also, the delimiter used to separate the fields in the output file is ‘|’.

All these commands are put together in a file named *Task3* stored at */home/acadgild/Abhilasha/Project2.2.*

This script file is executed as follows:

The command used is *pig Task3.pig*

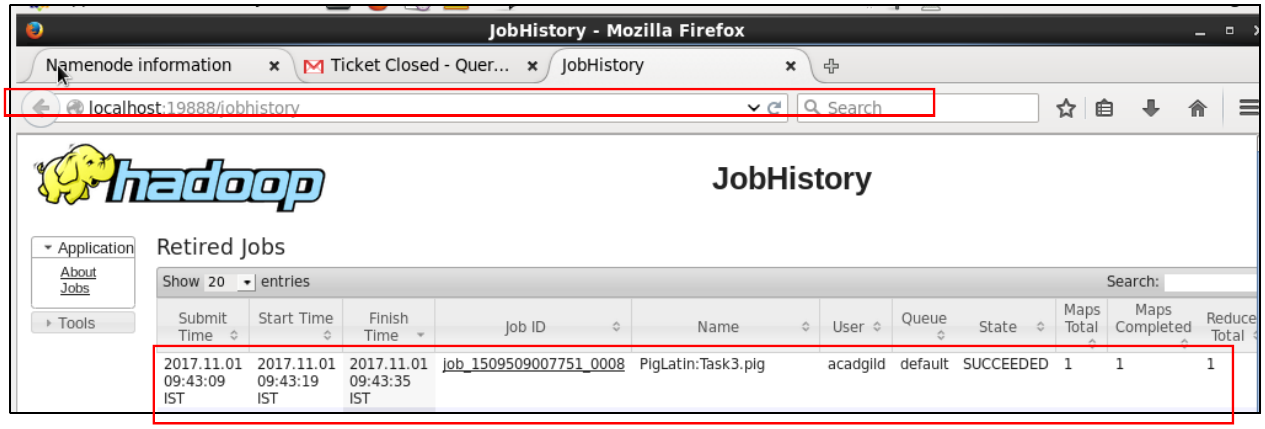
The command indicates that the script will be run not locally but will use HDFS to read and write data.



acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
[acadgild@localhost ~]$ cd Abhilasha/Project2.2/
[acadgild@localhost Project2.2]$ pig Task3.pig
2017-11-01 09:40:03,862 INFO  [main] pig.ExecTypeProvider: Trying ExecType : LOCAL
2017-11-01 09:40:03,866 INFO  [main] pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2017-11-01 09:40:03,866 INFO  [main] pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2017-11-01 09:40:03,950 [main] INFO  org.apache.pig.Main - Apache Pig version 0.14.0 (r1640057) compiled Nov 16 2014, 18:02:0
5
2017-11-01 09:40:03,950 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/acadgild/Abhilasha/Project2.2/pig
_1509509403949.log
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase/lib/slf4j-log4j12-1.6.4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Sta
ticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
2017-11-01 09:40:04,313 [main] WARN  org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your
platform... using builtin-java classes where applicable
2017-11-01 09:40:04,616 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file /home/acadgild/.pigbootup not found
2017-11-01 09:40:04,803 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Ins
tead, use mapreduce.jobtracker.address
```

The underlying map-reduce job can be seen successfully completed in the job history server as follows:



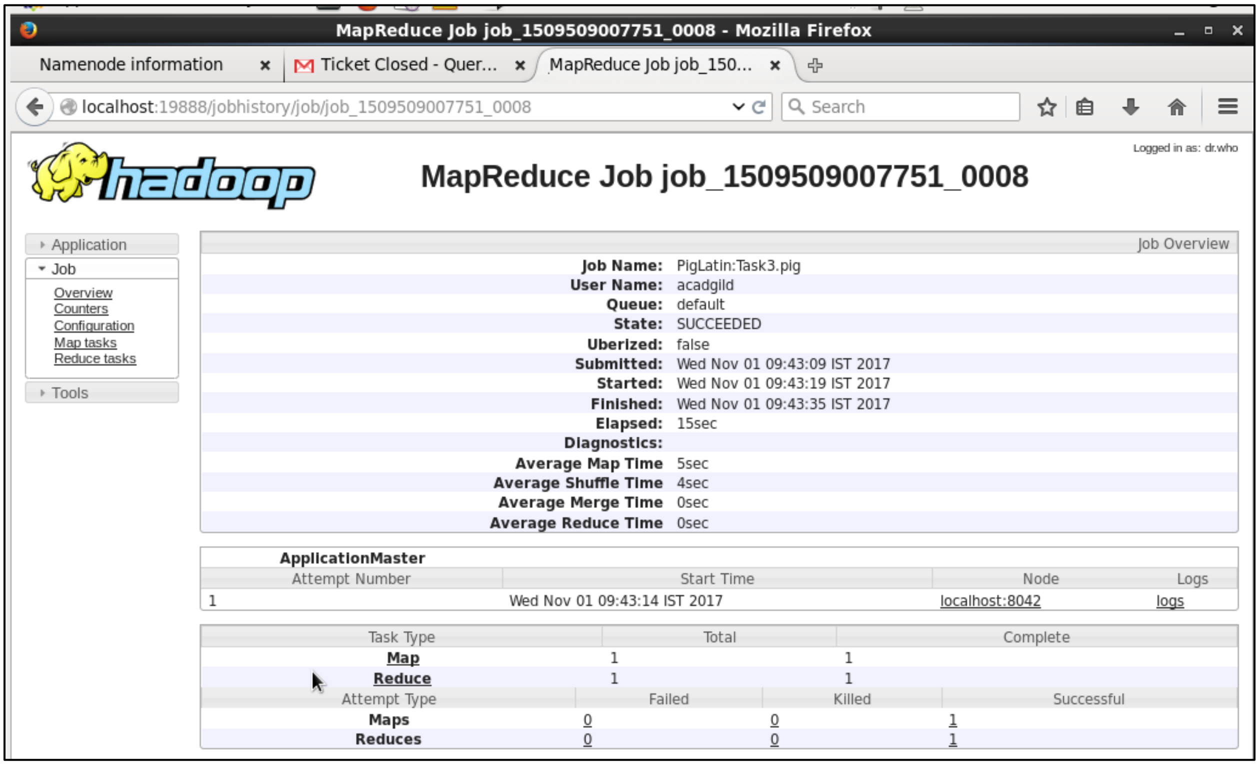
Retired Jobs

Show 20 entries

Search:

Submit Time	Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduce Total
2017.11.01 09:43:09 IST	2017.11.01 09:43:19 IST	2017.11.01 09:43:35 IST	job_1509509007751_0008	PigLatin:Task3.pig	acadgild	default	SUCCEEDED	1	1	1

Details of job execution can be seen below:



MapReduce Job job_1509509007751_0008

Job Overview

Job Name:	PigLatin:Task3.pig
User Name:	acadgild
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Wed Nov 01 09:43:09 IST 2017
Started:	Wed Nov 01 09:43:19 IST 2017
Finished:	Wed Nov 01 09:43:35 IST 2017
Elapsed:	15sec
Diagnostics:	
Average Map Time	5sec
Average Shuffle Time	4sec
Average Merge Time	0sec
Average Reduce Time	0sec

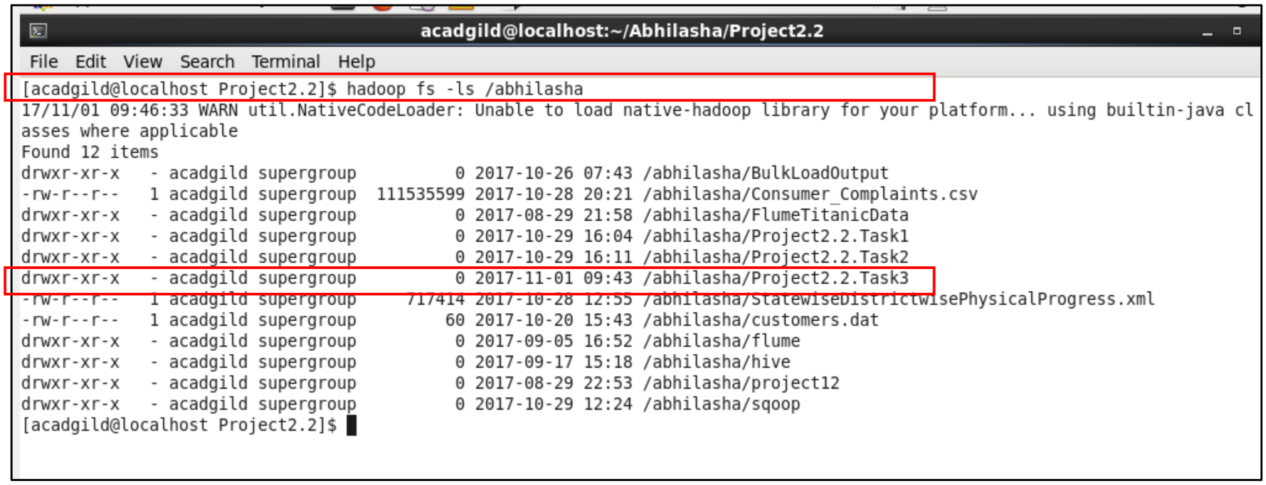
ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Wed Nov 01 09:43:14 IST 2017	localhost:8042	logs

Task Type	Total	Complete
Map	1	1
Reduce	1	1

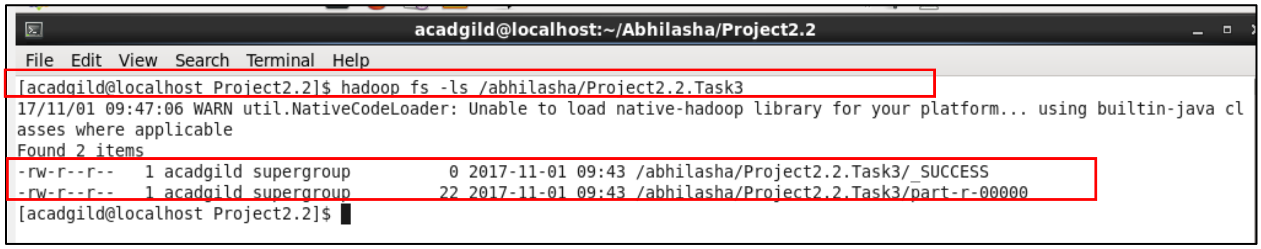
Attempt Type	Failed	Killed	Successful
Maps	0	0	1
Reduces	0	0	1

The output path mentioned in the script was */abhilasha/Project2.2.Task3* in HDFS. We can see the output folder that got created as a result of the job execution as follows:


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

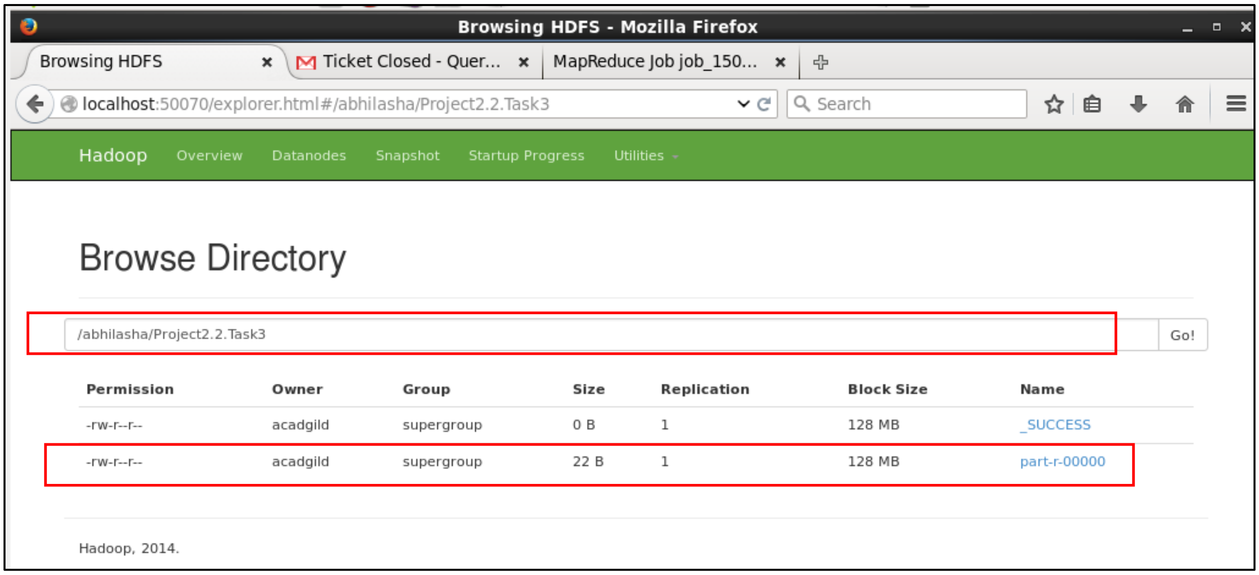
```
[acadgild@localhost Project2.2]$ hadoop fs -ls /abhilasha
17/11/01 09:46:33 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 12 items
drwxr-xr-x   - acadgild supergroup          0 2017-10-26 07:43 /abhilasha/BulkLoadOutput
-rw-r--r--   1 acadgild supergroup  111535599 2017-10-28 20:21 /abhilasha/Consumer_Complaints.csv
drwxr-xr-x   - acadgild supergroup          0 2017-08-29 21:58 /abhilasha/FlumeTitanicData
drwxr-xr-x   - acadgild supergroup          0 2017-10-29 16:04 /abhilasha/Project2.2.Task1
drwxr-xr-x   - acadgild supergroup          0 2017-10-29 16:11 /abhilasha/Project2.2.Task2
drwxr-xr-x   - acadgild supergroup          0 2017-11-01 09:43 /abhilasha/Project2.2.Task3
-rw-r--r--   1 acadgild supergroup     717414 2017-10-28 12:55 /abhilasha/StatewiseDistrictwisePhysicalProgress.xml
-rw-r--r--   1 acadgild supergroup         60 2017-10-20 15:43 /abhilasha/customers.dat
drwxr-xr-x   - acadgild supergroup          0 2017-09-05 16:52 /abhilasha/flume
drwxr-xr-x   - acadgild supergroup          0 2017-09-17 15:18 /abhilasha/hive
drwxr-xr-x   - acadgild supergroup          0 2017-08-29 22:53 /abhilasha/project12
drwxr-xr-x   - acadgild supergroup          0 2017-10-29 12:24 /abhilasha/sqoop
[acadgild@localhost Project2.2]$
```

This folder contains the files mentioned in the screen shot below. Of these *part-r-00000* file contains the output of the pig script.


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
[acadgild@localhost Project2.2]$ hadoop fs -ls /abhilasha/Project2.2.Task3
17/11/01 09:47:06 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 2 items
-rw-r--r--   1 acadgild supergroup          0 2017-11-01 09:43 /abhilasha/Project2.2.Task3/_SUCCESS
-rw-r--r--   1 acadgild supergroup         22 2017-11-01 09:43 /abhilasha/Project2.2.Task3/part-r-00000
[acadgild@localhost Project2.2]$ 
```

These files can also be viewed from HDFS UI as shown below:

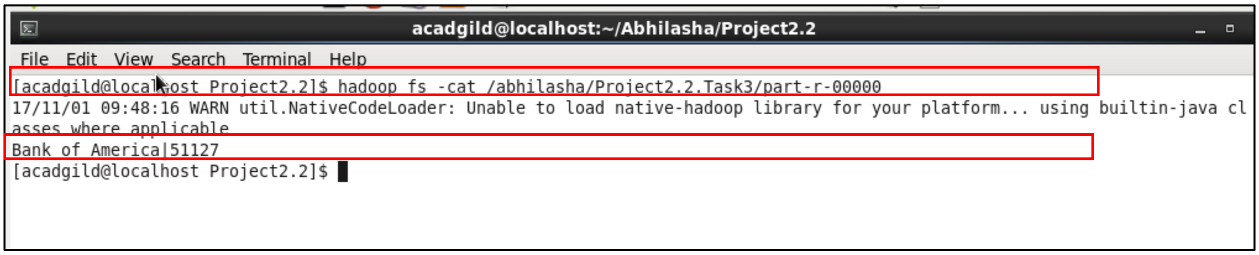


Browse Directory

/abhilasha/Project2.2.Task3

Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	acadgild	supergroup	0 B	1	128 MB	_SUCCESS
-rw-r--r--	acadgild	supergroup	22 B	1	128 MB	part-r-00000

The content of this output file can be seen using the **cat** command as follows:


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
[acadgild@localhost Project2.2]$ hadoop fs -cat /abhilasha/Project2.2.Task3/part-r-00000
17/11/01 09:48:16 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Bank of America|51127
[acadgild@localhost Project2.2]$ 
```

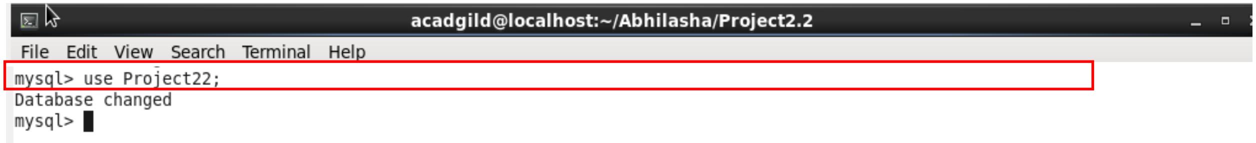
Note: To upload the output file in GitHub, we have renamed the file to *part-r-000003.*

Now that the output is placed in HDFS, we need to store the output in MySQL. To do so, we perform the following steps:

We have already seen steps to start MySQL and create database *Project22.* So not mentioning those steps again.

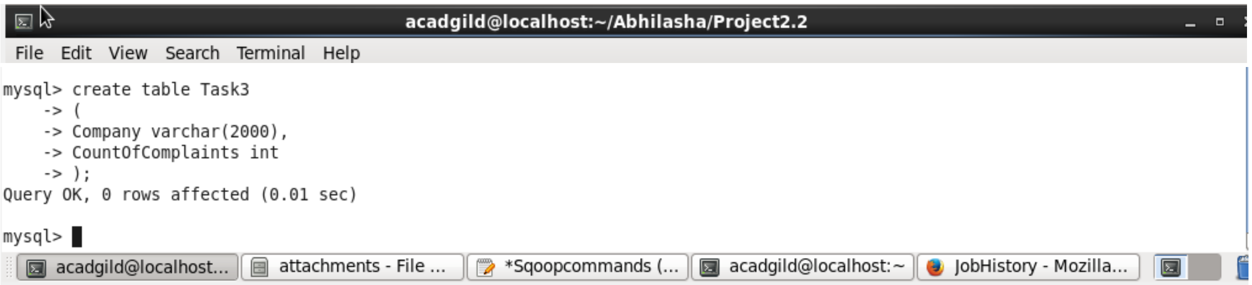
Step 1: As seen previously, we have created the database named *Project22*. We change the database from default to Project22, we use the command

use Project22;


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
mysql> use Project22;
Database changed
mysql> 
```

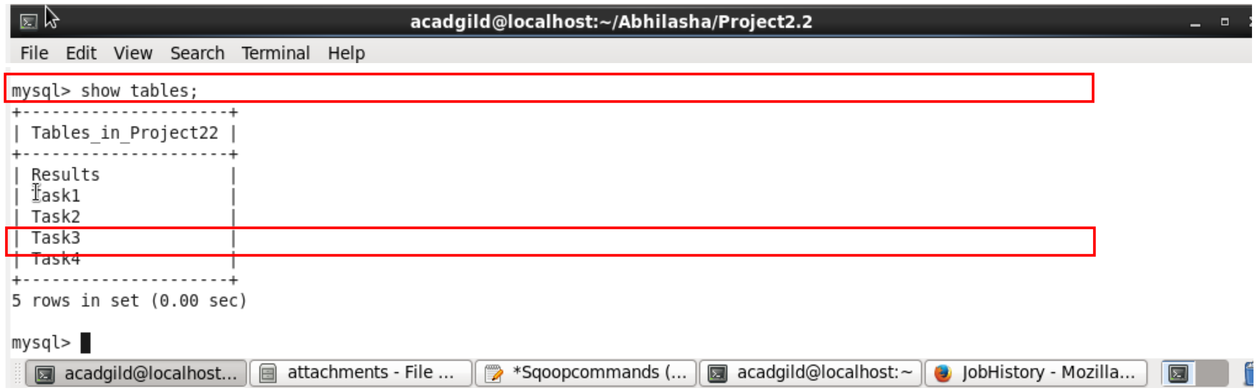
Step 5: We now create table named *Task2* to store the output of Task 2 in it using the command
create table Task3
(
Company varchar(2000),
CountOfComplaints int
);
Here, the columns of the table are *Company* and *CountOfComplaints*.



```
mysql> create table Task3
    -> (
    -> Company varchar(2000),
    -> CountOfComplaints int
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> 
```

Step 6: The created table can be listed using the command

Show tables;



```
mysql> show tables;
+---------------------+
| Tables_in_Project22 |
+---------------------+
| Results             |
| Task1               |
| Task2               |
| Task3               |
| Task4               |
+---------------------+
5 rows in set (0.00 sec)

mysql> 
```

Step 6: Next step is to export the data from HDFS and store in into the table created in previous step.

sqoop export --connect jdbc:mysql://localhost/Project22 --username 'root' -P --table 'Task3' --export-dir '/abhilasha/Project2.2.Task3/part-r-00000' –input-field-terminated-by '|' -m 1 --columns Columns,CountOfComplaints;

Here, the parameters mentioned in the above command are:

i. Database : jdbc:mysql://localhost/Project22

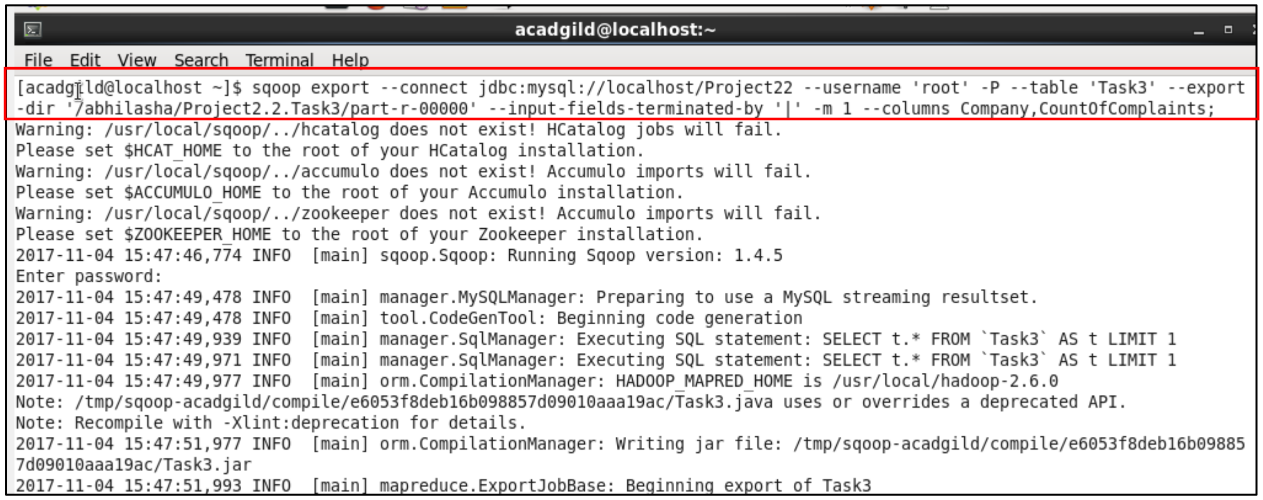
ii. Table Name : Task2

iii. Path where exported file is placed : /abhilasha/Project2.2.Task2/part-r-00000

iv. Field separator : |

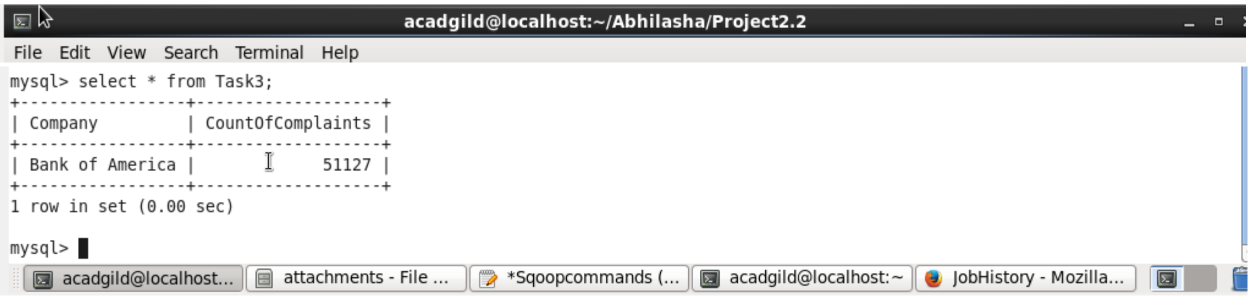
v. Columns to populate : Company, CountOfComplaints

The command execution is shown below:



```
[acadgild@localhost ~]$ sqoop export --connect jdbc:mysql://localhost/Project22 --username 'root' -P --table 'Task3' --export
-dir '/abhilasha/Project2.2.Task3/part-r-00000' --input-fields-terminated-by '|' -m 1 --columns Company,CountOfComplaints;
Warning: /usr/local/sqoop/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /usr/local/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /usr/local/sqoop/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
2017-11-04 15:47:46,774 INFO  [main] sqoop.Sqoop: Running Sqoop version: 1.4.5
Enter password:
2017-11-04 15:47:49,478 INFO  [main] manager.MySQLManager: Preparing to use a MySQL streaming resultset.
2017-11-04 15:47:49,478 INFO  [main] tool.CodeGenTool: Beginning code generation
2017-11-04 15:47:49,939 INFO  [main] manager.SqlManager: Executing SQL statement: SELECT t.* FROM `Task3` AS t LIMIT 1
2017-11-04 15:47:49,971 INFO  [main] manager.SqlManager: Executing SQL statement: SELECT t.* FROM `Task3` AS t LIMIT 1
2017-11-04 15:47:49,977 INFO  [main] orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/local/hadoop-2.6.0
Note: /tmp/sqoop-acadgild/compile/e6053f8deb16b098857d09010aaa19ac/Task3.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
2017-11-04 15:47:51,977 INFO  [main] orm.CompilationManager: Writing jar file: /tmp/sqoop-acadgild/compile/e6053f8deb16b09885
7d09010aaa19ac/Task3.jar
2017-11-04 15:47:51,993 INFO  [main] mapreduce.ExportJobBase: Beginning export of Task3
```

Step 7: After the successful placement of data in MySQL, we can see the content of the table in the database populated, using *select * from Task3;* as follows:



```
mysql> select * from Task3;
+-----------------+-------------------+
| Company         | CountOfComplaints |
+-----------------+-------------------+
| Bank of America |             51127 |
+-----------------+-------------------+
1 row in set (0.00 sec)

mysql> 
```

This shows that the data is placed successfully in database.

Task 4: Write a pig script to find no of complaints filed with product type has "Debt collection" for the year 2015.

Answer:

The fields from the data that we are going to use are *Date Received*, *Product to company* and *Complaint ID*.

The pig script that we are using to solve the problem statement has the following steps:

Step 1: *REGISTER '/usr/local/pig/lib/piggybank.jar';*

We are going to use *CSVExcelStorage* to read data from the csv file. In order to use this, which is present in *piggybank.jar*, we register this jar.

We mention the full qualified local path of the jar, which is */usr/local/pig/lib/piggybank.jar* in this case.

Step 2*: complaintDetails = LOAD '/abhilasha/Consumer_Complaints.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',','NO_MULTILINE','UNIX','SKIP_INPUT_HEADER') AS (dateRec:chararray, product:chararray, subProduct:chararray, issue:chararray, subIssue:chararray, complaintNarrative:chararray, companyPublicResponse:chararray, company:chararray, state:chararray, zipCode:chararray, submittedVia:chararray, dateSentToCompany:chararray, companyResponseToConsumer:chararray, timelyResponse:chararray, consumerDisputed:chararray, complaintId:chararray);*

This command is to specify the details of the input file, schema of the data if known, to be used to load the data.

Here, the file path given is */abhilasha/Consumer_Complaints.csv*. This is the location in HDFS. Rest is the schema of the input, as we have the schema beforehand.

Step 3: *requiredRecords = FILTER complaintDetails BY (product == 'Debt collection' AND dateRec MATCHES '.*2015.*');*

As we need records where complaints filed with product type has " Debt collection" for the year 2015, we use the clause *product == 'Debt collection' AND dateRec MATCHES '.*2015.*'*, to filter the records. Here, *MATCHES* is used to perform pattern matching to get records where date of registration of complaint is 2015.

Step 4: *distinctComplaints = GROUP requiredRecords BY complaintId;*

This command is used to get rid of the redundant records. It is found that the data in the input file had a pinch of duplicity. Hence, this is the measure taken to avoid duplicate complaint IDs in the data. So, we group the data by complaint ID.

Step 5: *complaintIds = FOREACH distinctComplaints GENERATE group AS complaintID;*

Of all the fields in the data, now only complaint ID remains of use to us. Hence, we extract only the complaint ID from the records in the previous step.

Step 6: *groupForCount = GROUP complaintIds ALL;*

This step is to aid us in getting the total count of complaints. This step will get all the complaint IDs in a single group.

Step 7: *count = FOREACH groupForCount GENERATE COUNT(complaintIds);*

This is used to get the count of complaints grouped in the previous step.

Step 8: *STORE count INTO '/abhilasha/Project2.2.Task4';*

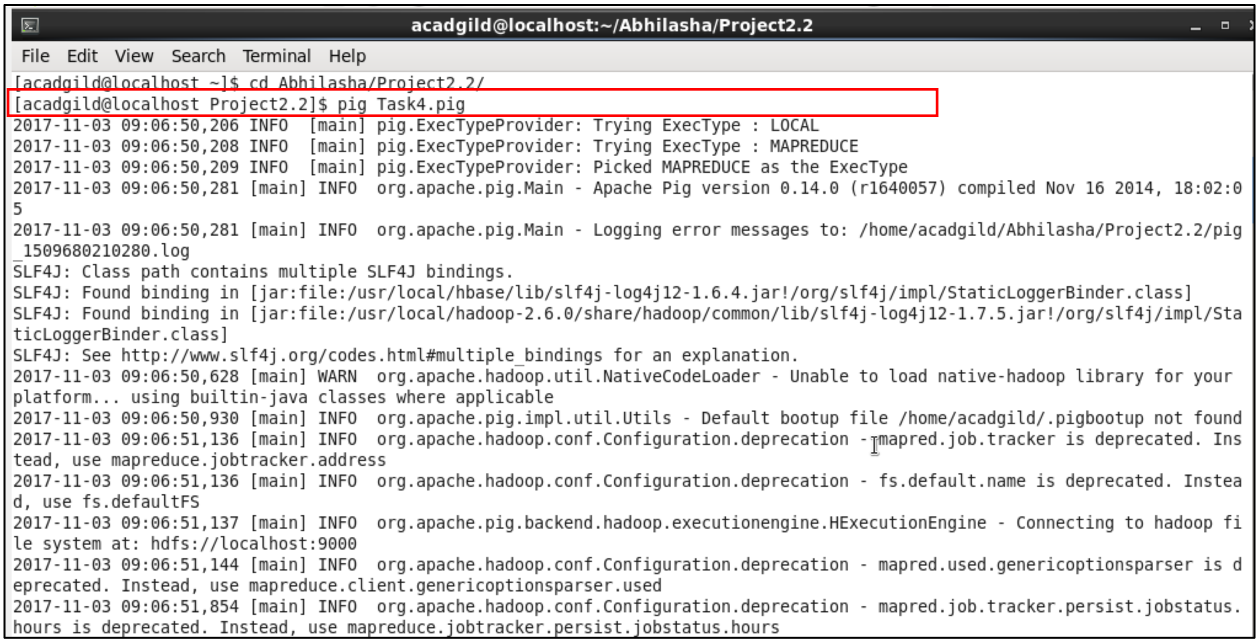
This is to store the data back into HDFS. The destination path given is */abhilasha/Project2.2.Task4.*

All these commands are put together in a file named *Task4* stored at */home/acadgild/Abhilasha/Project2.2*.

This script file is executed as follows:

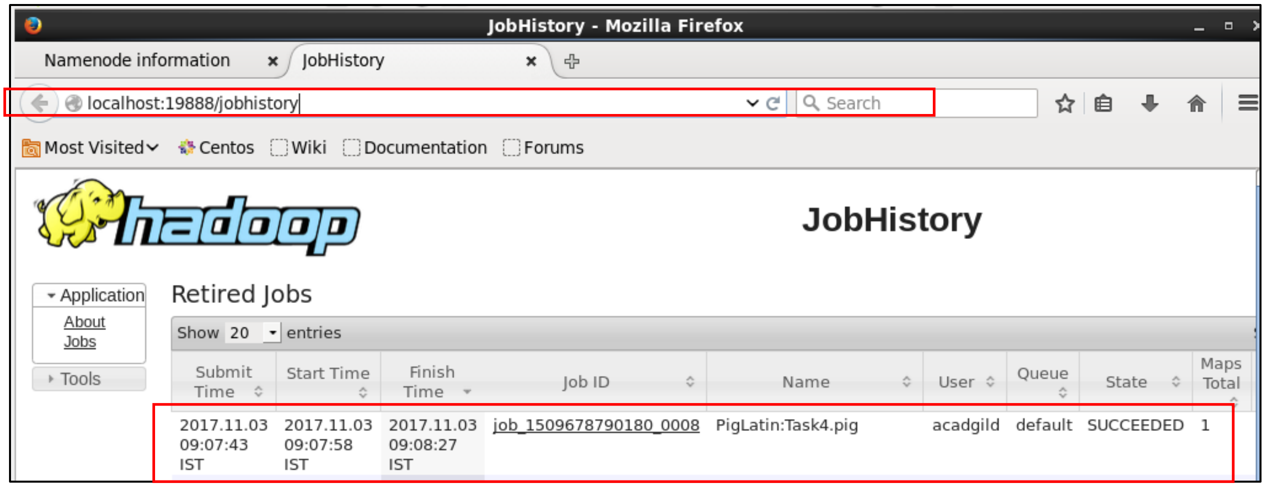
The command used is *pig Task4.pig*

The command indicates that the script will be run not locally but will use HDFS to read and write data.


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

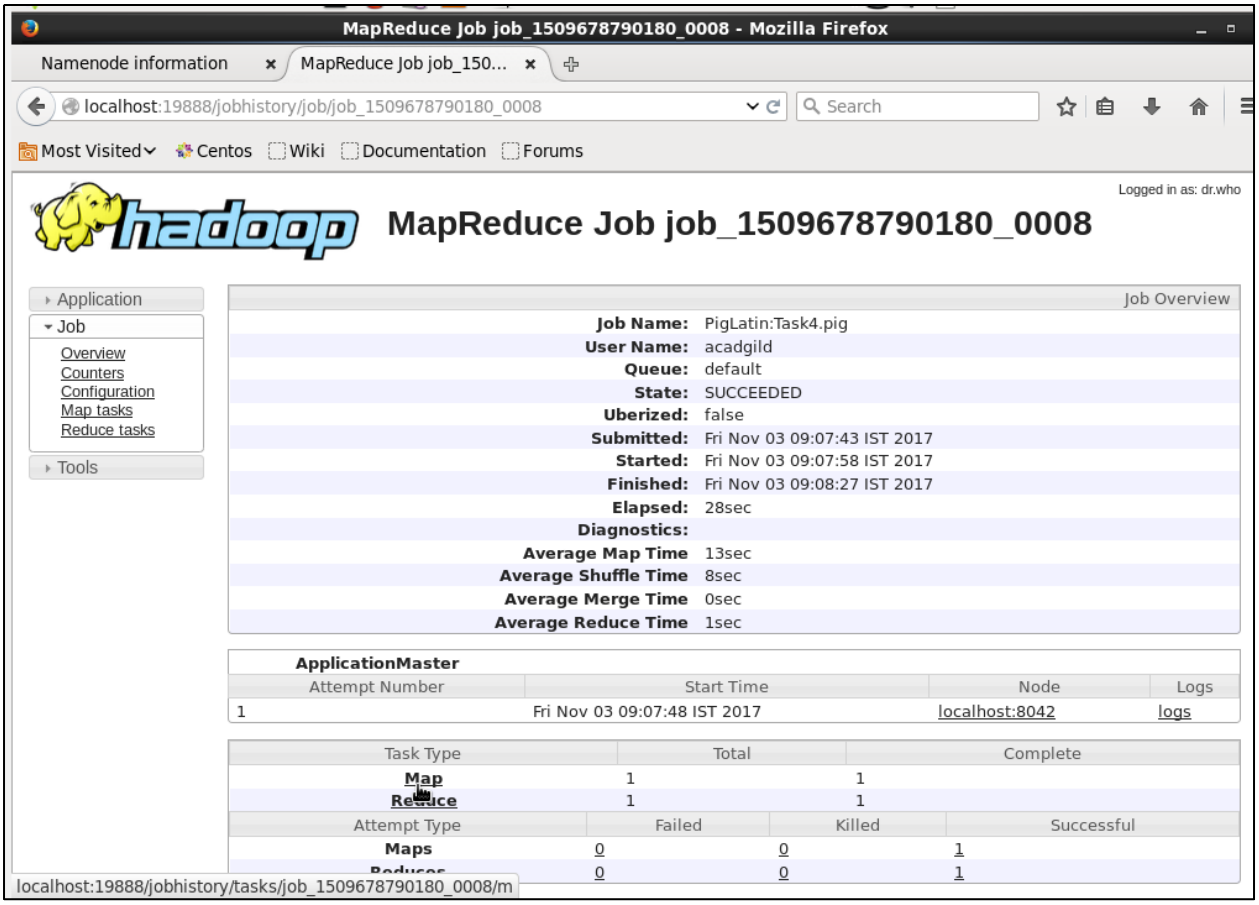
```
[acadgild@localhost ~]$ cd Abhilasha/Project2.2/
[acadgild@localhost Project2.2]$ pig Task4.pig
2017-11-03 09:06:50,206 INFO  [main] pig.ExecTypeProvider: Trying ExecType : LOCAL
2017-11-03 09:06:50,208 INFO  [main] pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2017-11-03 09:06:50,209 INFO  [main] pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2017-11-03 09:06:50,281 [main] INFO  org.apache.pig.Main - Apache Pig version 0.14.0 (r1640057) compiled Nov 16 2014, 18:02:0
5
2017-11-03 09:06:50,281 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/acadgild/Abhilasha/Project2.2/pig
_1509680210280.log
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase/lib/slf4j-log4j12-1.6.4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Sta
ticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
2017-11-03 09:06:50,628 [main] WARN  org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your
platform... using builtin-java classes where applicable
2017-11-03 09:06:50,930 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file /home/acadgild/.pigbootup not found
2017-11-03 09:06:51,136 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Ins
tead, use mapreduce.jobtracker.address
2017-11-03 09:06:51,136 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instea
d, use fs.defaultFS
2017-11-03 09:06:51,137 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop fi
le system at: hdfs://localhost:9000
2017-11-03 09:06:51,144 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.used.genericoptionsparser is d
eprecated. Instead, use mapreduce.client.genericoptionsparser.used
2017-11-03 09:06:51,854 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker.persist.jobstatus.
hours is deprecated. Instead, use mapreduce.jobtracker.persist.jobstatus.hours
```

The underlying map-reduce job can be seen successfully completed in the job history server as follows:



Submit Time	Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total
2017.11.03 09:07:43 IST	2017.11.03 09:07:58 IST	2017.11.03 09:08:27 IST	job_1509678790180_0008	PigLatin:Task4.pig	acadgild	default	SUCCEEDED	1

Details of job execution can be seen below:



MapReduce Job job_1509678790180_0008

Job Overview	
Job Name:	PigLatin:Task4.pig
User Name:	acadgild
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Fri Nov 03 09:07:43 IST 2017
Started:	Fri Nov 03 09:07:58 IST 2017
Finished:	Fri Nov 03 09:08:27 IST 2017
Elapsed:	28sec
Diagnostics:	
Average Map Time	13sec
Average Shuffle Time	8sec
Average Merge Time	0sec
Average Reduce Time	1sec

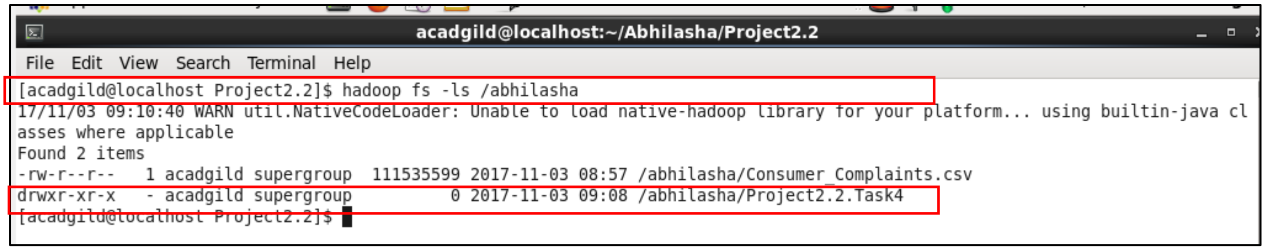
ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Fri Nov 03 09:07:48 IST 2017	localhost:8042	logs

Task Type	Total	Complete
Map	1	1
Reduce	1	1

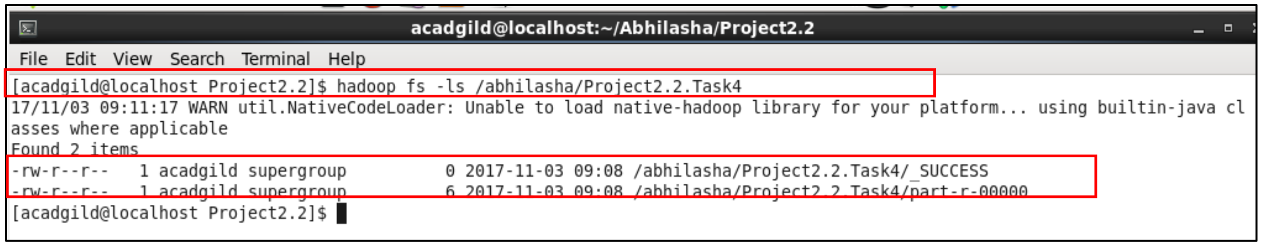
Attempt Type	Failed	Killed	Successful
Maps	0	0	1
Reduces	0	0	1

The output path mentioned in the script was */abhilasha/Project2.2.Task4* in HDFS. We can see the output folder that got created as a result of the job execution as follows:


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

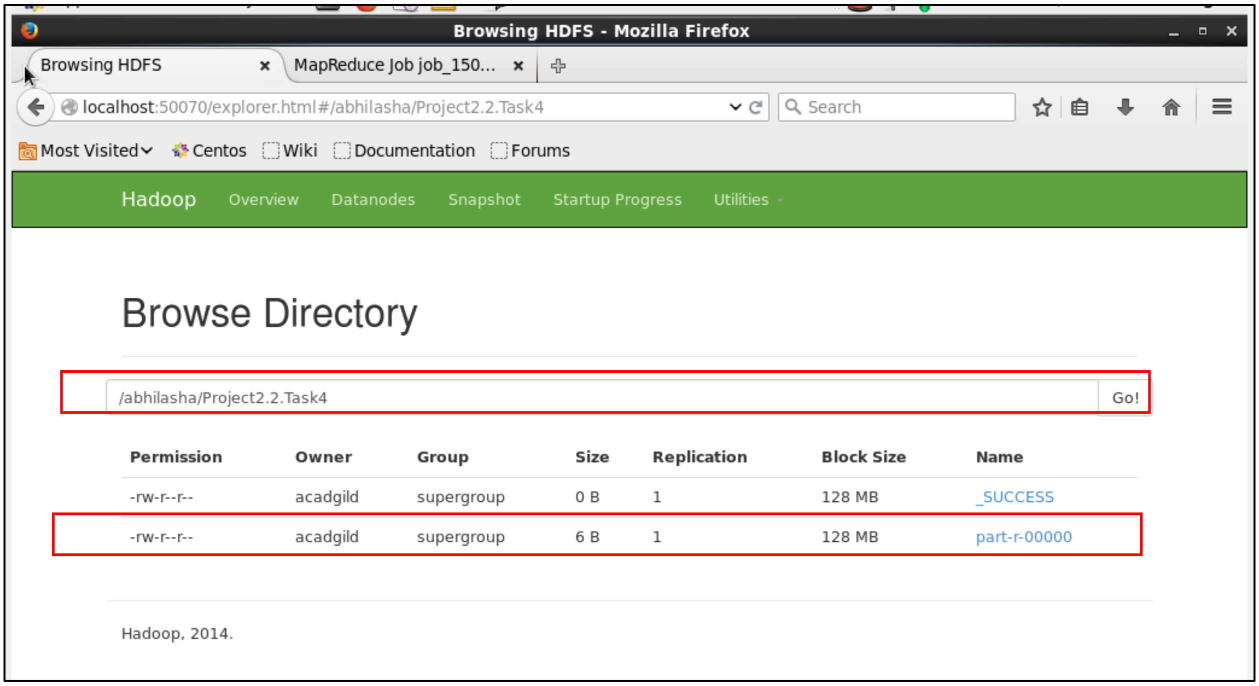
```
[acadgild@localhost Project2.2]$ hadoop fs -ls /abhilasha
17/11/03 09:10:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 2 items
-rw-r--r--   1 acadgild supergroup  111535599 2017-11-03 08:57 /abhilasha/Consumer_Complaints.csv
drwxr-xr-x   - acadgild supergroup          0 2017-11-03 09:08 /abhilasha/Project2.2.Task4
[acadgild@localhost Project2.2]$
```

This folder contains the files mentioned in the screen shot below. Of these *part-r-00000* file contains the output of the pig script.


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
[acadgild@localhost Project2.2]$ hadoop fs -ls /abhilasha/Project2.2.Task4
17/11/03 09:11:17 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Found 2 items
-rw-r--r--   1 acadgild supergroup          0 2017-11-03 09:08 /abhilasha/Project2.2.Task4/_SUCCESS
-rw-r--r--   1 acadgild supergroup          6 2017-11-03 09:08 /abhilasha/Project2.2.Task4/part-r-00000
[acadgild@localhost Project2.2]$
```

These files can also be viewed from HDFS UI as shown below:



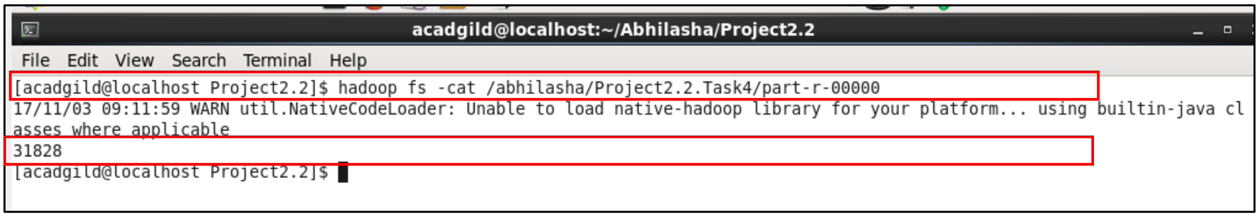
Browse Directory

/abhilasha/Project2.2.Task4 Go!

Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	acadgild	supergroup	0 B	1	128 MB	_SUCCESS
-rw-r--r--	acadgild	supergroup	6 B	1	128 MB	part-r-00000

Hadoop, 2014.

The content of this output file can be seen using the **cat** command as follows:


acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
[acadgild@localhost Project2.2]$ hadoop fs -cat /abhilasha/Project2.2.Task4/part-r-00000
17/11/03 09:11:59 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
31828
[acadgild@localhost Project2.2]$
```

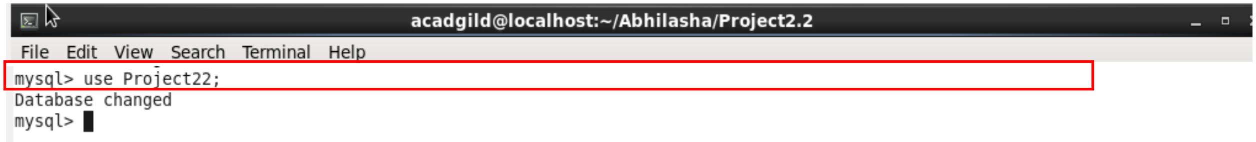
Note: To upload the output file in GitHub, we have renamed the file to *part-r-000004.*

Now that the output is placed in HDFS, we need to store the output in MySQL. To do so, we perform the following steps:

We have already seen steps to start MySQL and create database *Project22.* So not mentioning those steps again.

Step 1: As seen previously, we have created the database named *Project22.* We change the database from default to Project22, we use the command

use Project22;

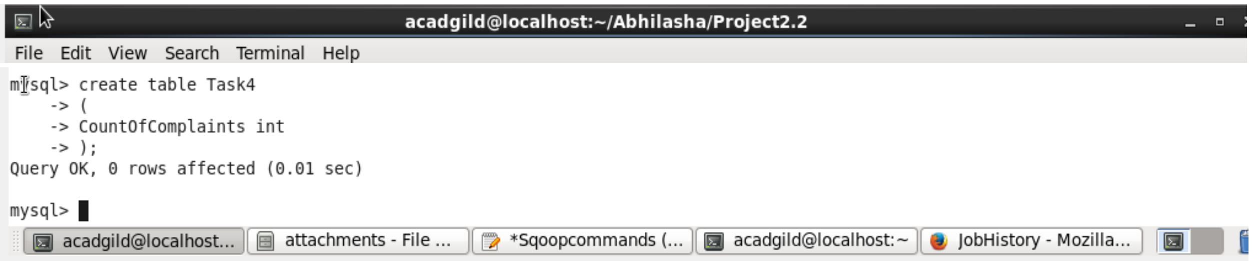

acadgild@localhost:~/Abhilasha/Project2.2
File Edit View Search Terminal Help

```
mysql> use Project22;
Database changed
mysql>
```

Step 5: We now create table named *Task4* to store the output of Task 4 in it using the command

create table Task4
(
            CountOfComplaints int
);

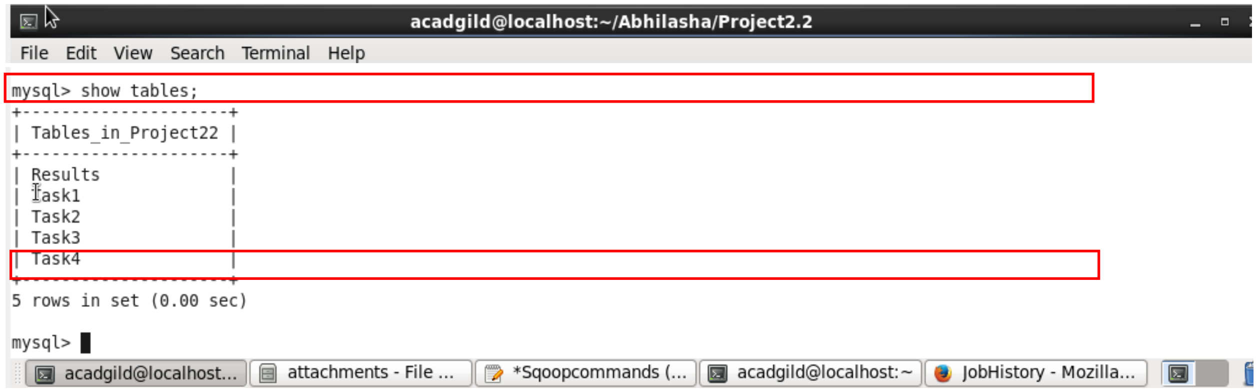
Here, the column of the table is *CountOfComplaints.*



```
mysql> create table Task4
    -> (
    -> CountOfComplaints int
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> 
```

Step 6: The created table can be listed using the command

Show tables;



```
mysql> show tables;
+---------------------+
| Tables_in_Project22 |
+---------------------+
| Results             |
| Task1               |
| Task2               |
| Task3               |
| Task4               |
+---------------------+
5 rows in set (0.00 sec)

mysql> 
```

Step 6: Next step is to export the data from HDFS and store in into the table created in previous step.

sqoop export --connect jdbc:mysql://localhost/Project22 --username 'root' -P --table 'Task4' --export-dir '/abhilasha/Project2.2.Task4/part-r-00000' -m 1 --columns CountOfComplaints;

Here, the parameters mentioned in the above command are:

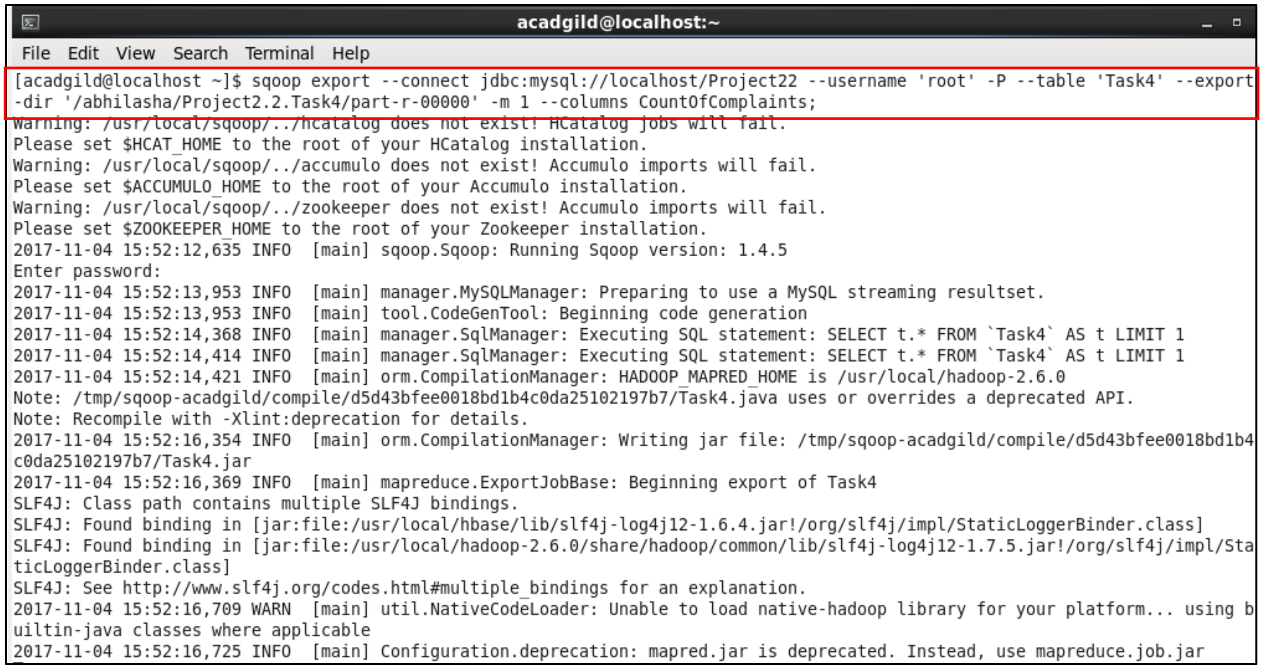
v. Database : jdbc:mysql://localhost/Project22

vi. Table Name : Task4

vii. Path where exported file is placed : /abhilasha/Project2.2.Task4/part-r-00000

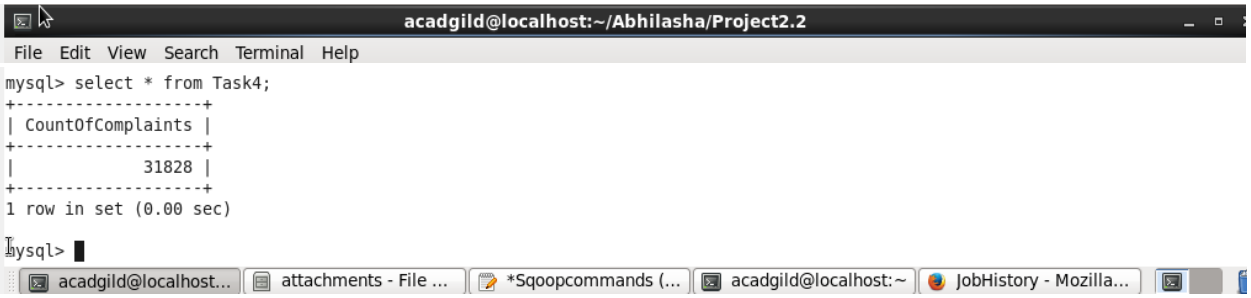
viii. Column to populate : CountOfComplaints

The command execution is shown below:



```
[acadgild@localhost ~]$ sqoop export --connect jdbc:mysql://localhost/Project22 --username 'root' -P --table 'Task4' --export
-dir '/abhilasha/Project2.2.Task4/part-r-00000' -m 1 --columns CountOfComplaints;
Warning: /usr/local/sqoop/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /usr/local/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /usr/local/sqoop/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
2017-11-04 15:52:12,635 INFO  [main] sqoop.Sqoop: Running Sqoop version: 1.4.5
Enter password:
2017-11-04 15:52:13,953 INFO  [main] manager.MySQLManager: Preparing to use a MySQL streaming resultset.
2017-11-04 15:52:13,953 INFO  [main] tool.CodeGenTool: Beginning code generation
2017-11-04 15:52:14,368 INFO  [main] manager.SqlManager: Executing SQL statement: SELECT t.* FROM `Task4` AS t LIMIT 1
2017-11-04 15:52:14,414 INFO  [main] manager.SqlManager: Executing SQL statement: SELECT t.* FROM `Task4` AS t LIMIT 1
2017-11-04 15:52:14,421 INFO  [main] orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/local/hadoop-2.6.0
Note: /tmp/sqoop-acadgild/compile/d5d43bfee0018bd1b4c0da25102197b7/Task4.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
2017-11-04 15:52:16,354 INFO  [main] orm.CompilationManager: Writing jar file: /tmp/sqoop-acadgild/compile/d5d43bfee0018bd1b4
c0da25102197b7/Task4.jar
2017-11-04 15:52:16,369 INFO  [main] mapreduce.ExportJobBase: Beginning export of Task4
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase/lib/slf4j-log4j12-1.6.4.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/Sta
ticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
2017-11-04 15:52:16,709 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using b
uiltin-java classes where applicable
2017-11-04 15:52:16,725 INFO  [main] Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar
```

Step 7: After the successful placement of data in MySQL, we can see the content of the table in the database populated, using *select * from Task4;* as follows:



```
mysql> select * from Task4;
+-------------------+
| CountOfComplaints |
+-------------------+
|             31828 |
+-------------------+
1 row in set (0.00 sec)

mysql> 
```

This shows that the data is placed successfully in database.