Big Data

Worksheet 3

Name: Prakash Dahal

Student ID: 1828421

Remainder: 2

Making a separate worksheet3 folder through putty inside my directory (**home/1828421**) in the server.

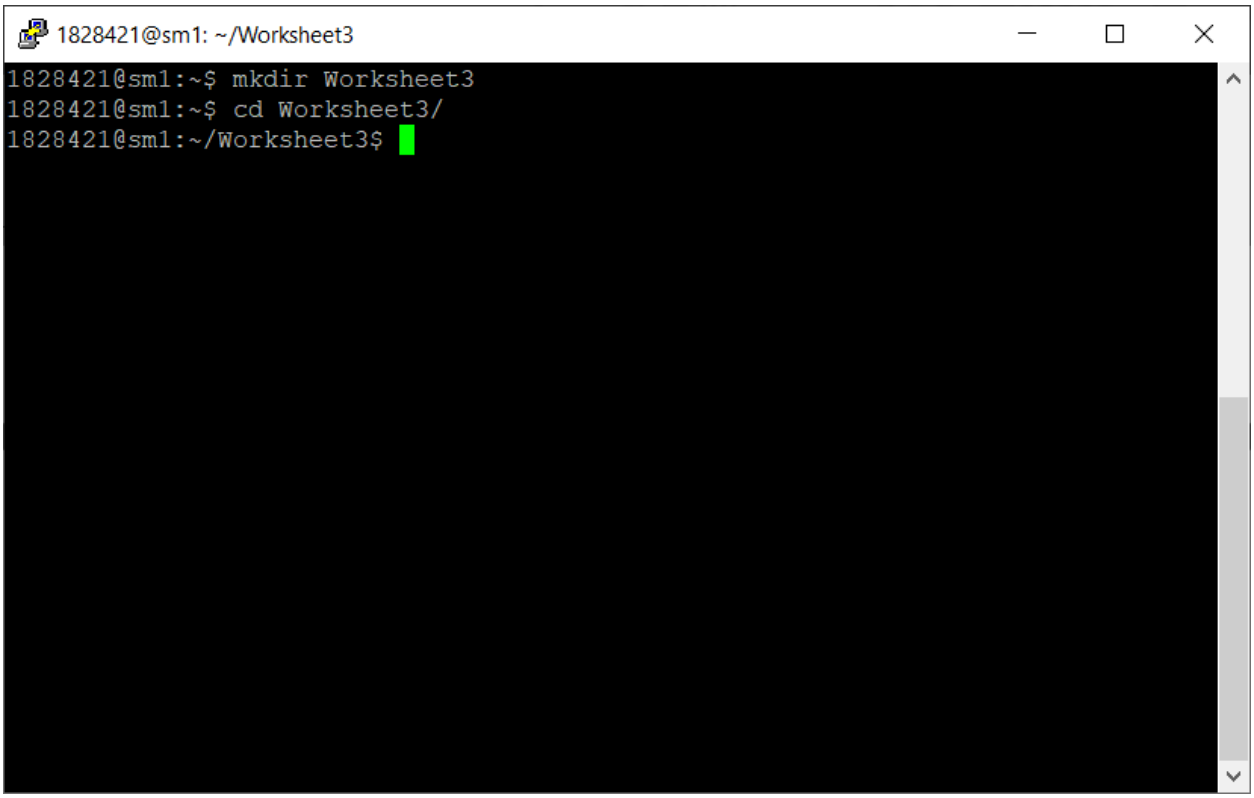


Figure 1: Worksheet folder

Copying Population.java file into this directory.

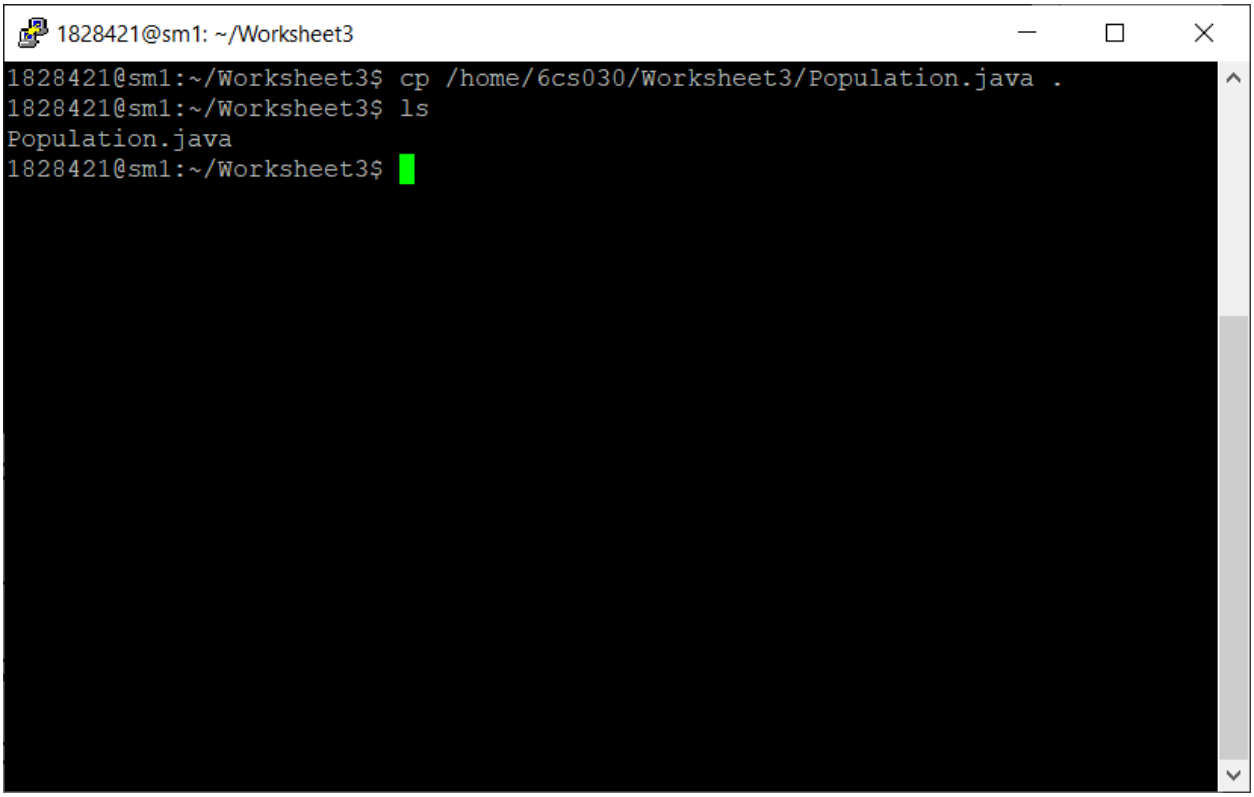


Figure 2: Copying java file and displaying

Now, making required changes in the copied file.

1. Java and Hadoop

Population.java file has to be edited. Therefore, it is downloaded using filezilla and stored in my local folder to edit.

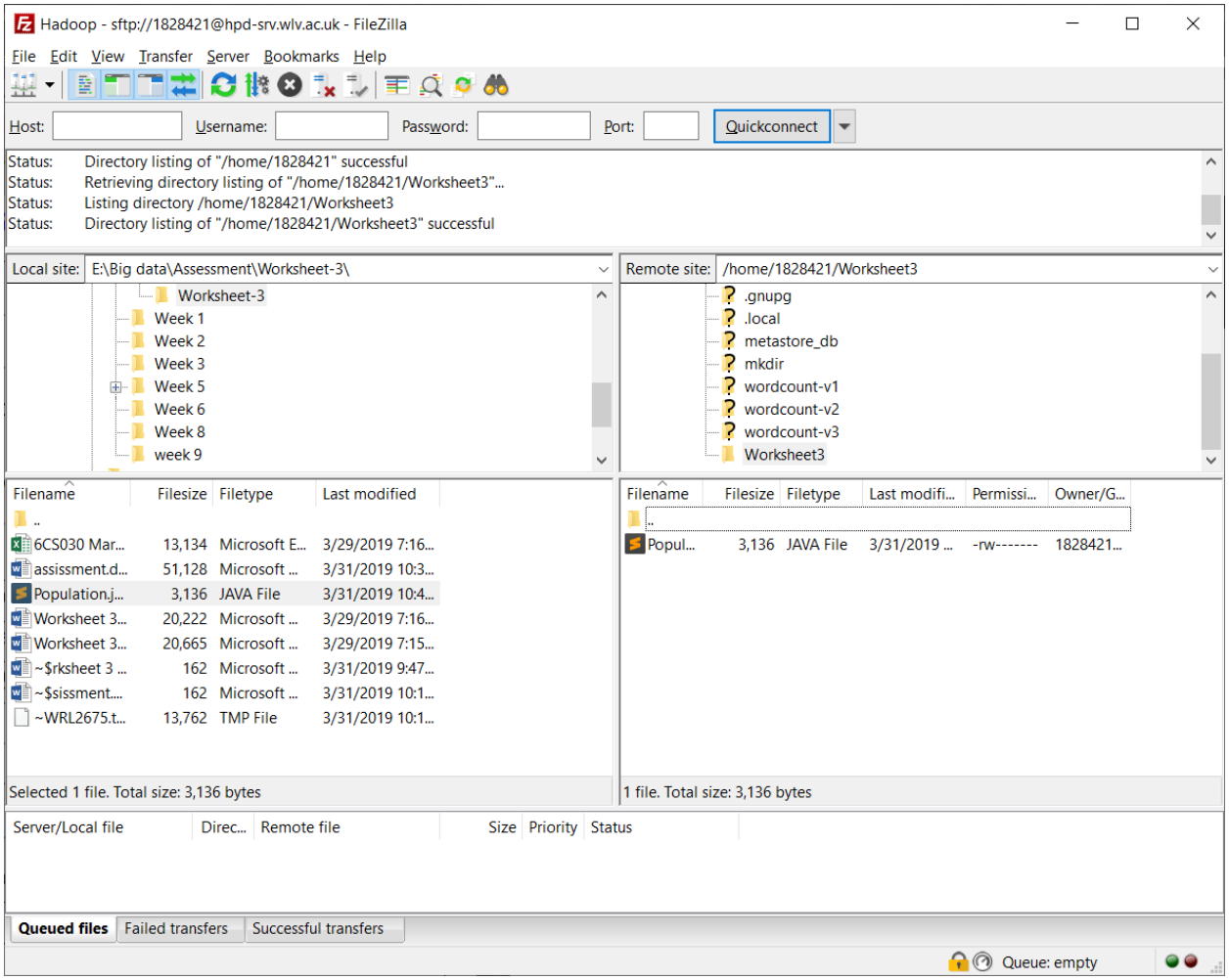


Figure 3: Saving file locally

Opening the locally saved file with sublime.

Changing the main Class name as **NoQuals** as per the given class name. Changing Mapper class as **PDMapper** and Reducer class as **PDReducer** where first two alphabets (**PD**) are my initials. Finally fixing the changed classes name in the job section.

```
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.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class NoQuals {

public static class PDMapper extends Mapper <Object, Text, Text, Text>  {
    public void map(Object key, Text value, Context context)
        throws IOException, InterruptedException {

    String record = value.toString();
    String[] parts = record.split(",");
    // 0: Key (county) 1: Year 2: figure
    // need to deal with null values - defaults to 0
    if (parts.length == 3 )
        context.write(new Text(parts[0]), new Text(parts[2]));
    else
        context.write(new Text(parts[0]), new Text("0"));
        } // map
    } // PopMapper

    public static class PDReducer extends Reducer <Text, Text, Text, Text> {
        public static boolean isInteger(String s) {
            try {
                Integer.parseInt(s);
            } catch(NumberFormatException e) {
                return false;
                } catch(NullPointerException e) {
                return false;
                }
```

Figure 4: Updated class, mapper and reducer name

```java
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    //set output delimiter to comma
    conf.set("mapreduce.output.textoutputformat.separator", ",");

    Job job = Job.getInstance(conf, "No Quals Count");
    job.setJarByClass(NoQuals.class);
    job.setMapperClass(PDMapper.class);
    job.setReducerClass(PDReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
```

Figure 5: Fixing classes names in Job

Now renaming the file name as the main class name since java class name and file name has to be same.

Name	Date modified	Type	Size
6CS030 Marksheet 3	3/29/2019 7:16 AM	Microsoft Excel W...	13 KB
assissment	4/1/2019 6:52 PM	Microsoft Word D...	499 KB
NoQuals	4/1/2019 6:57 PM	JAVA File	4 KB
Worksheet 3 Template	3/29/2019 7:16 AM	Microsoft Word D...	20 KB
Worksheet 3	3/29/2019 7:15 AM	Microsoft Word D...	21 KB

Figure 6: Renaming Java file name as NoQuals

Now uploading this updated file into the server (**Worksheet3**) folder through filezilla.

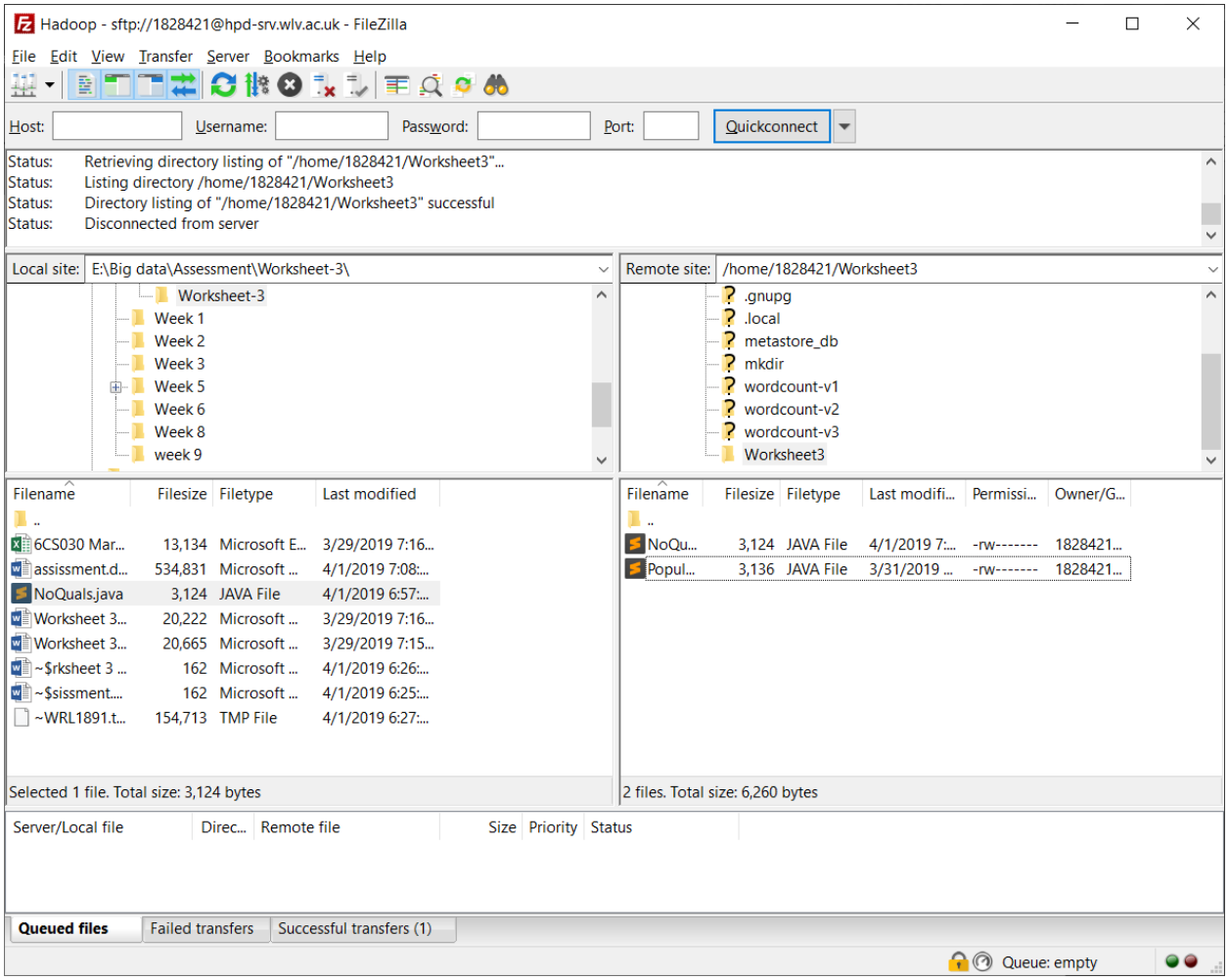


Figure 7: Uploading edited file

2. Running the code

While running the code, at first java file is compiled and jar file is created then the file is stored into input and output folders and then running the program.

a. Compiling the NoQuals.java

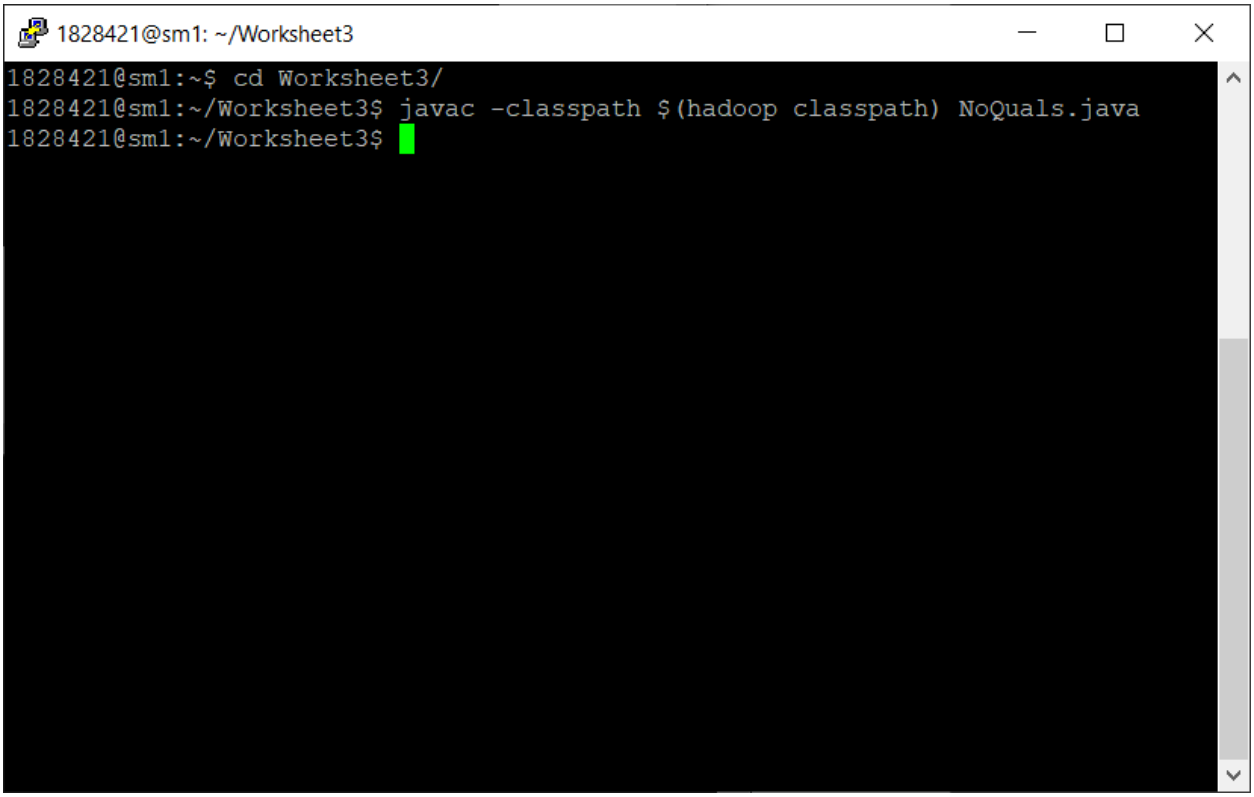


Figure 8: Compiling NoQuals.java

b. Creating jar file

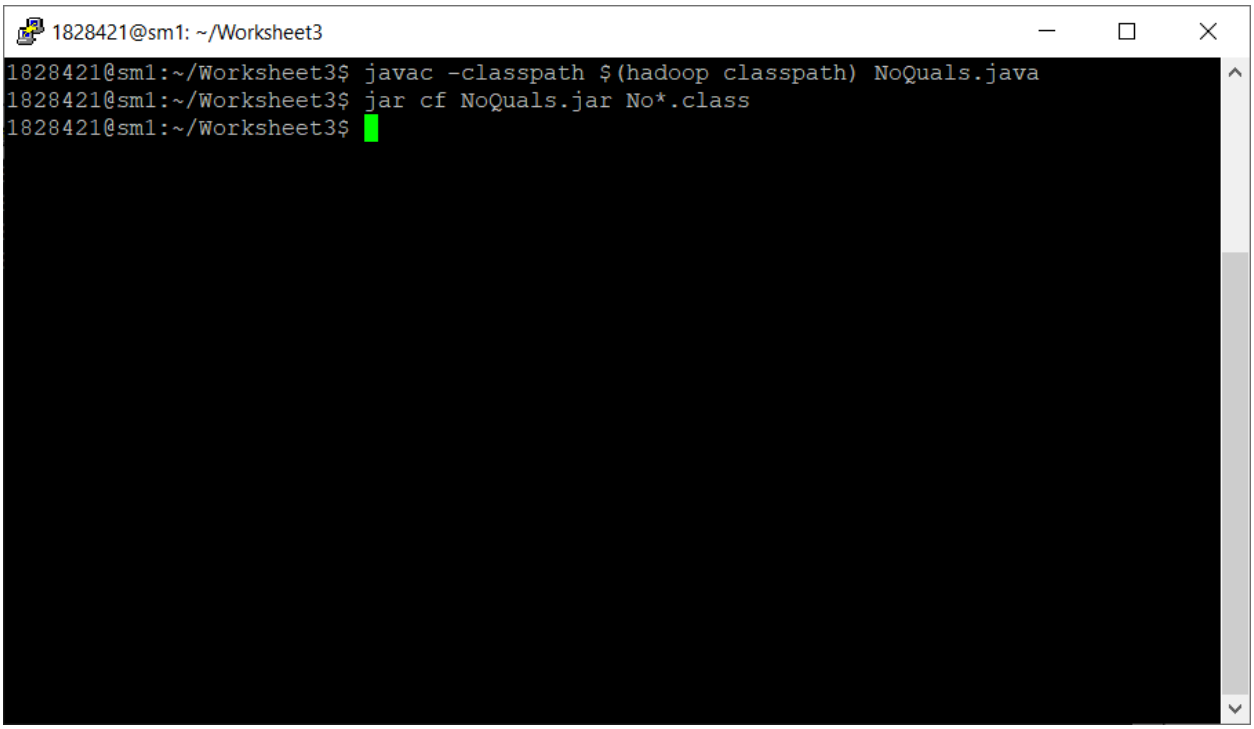


Figure 9: Creating jar file

c. Creating input and output directories and storing files

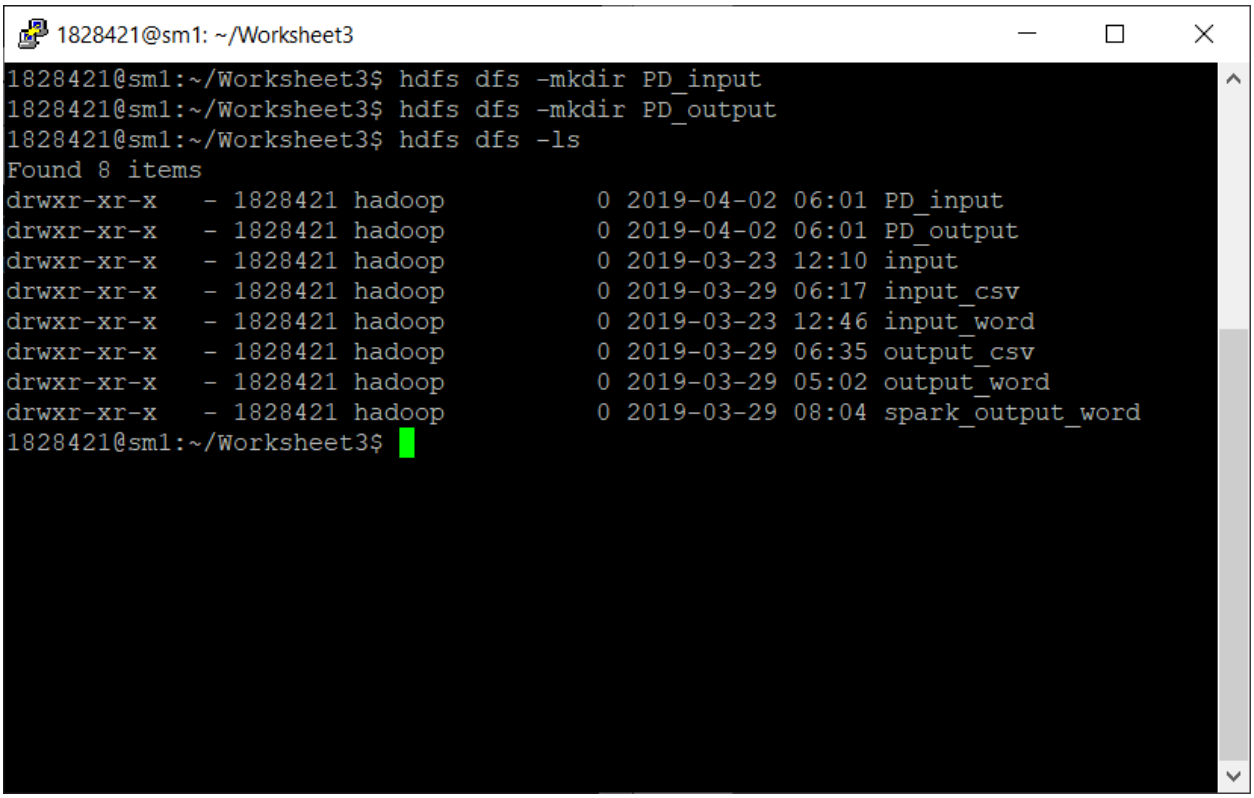
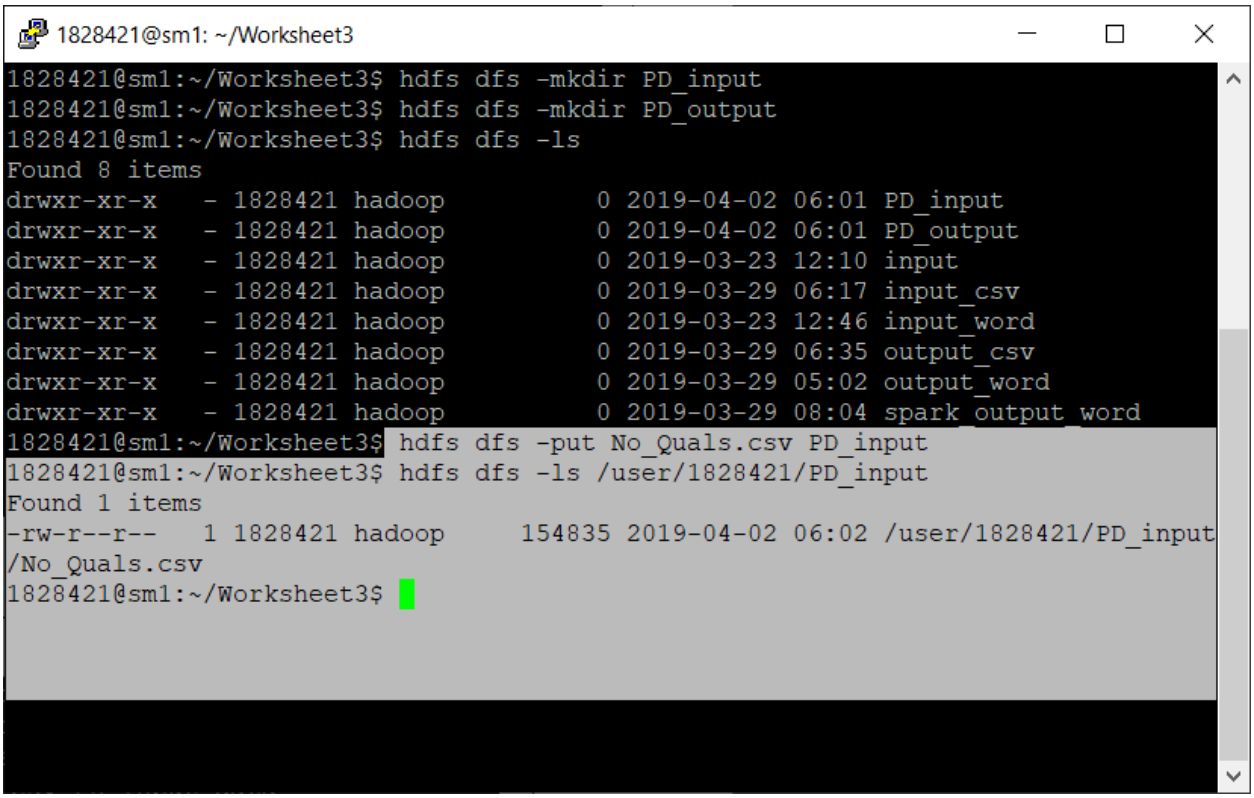


Figure 10: Making directories



```
1828421@sm1: ~/Worksheet3
1828421@sm1:~/Worksheet3$ hdfs dfs -mkdir PD_input
1828421@sm1:~/Worksheet3$ hdfs dfs -mkdir PD_output
1828421@sm1:~/Worksheet3$ hdfs dfs -ls
Found 8 items
drwxr-xr-x   - 1828421 hadoop          0 2019-04-02 06:01 PD_input
drwxr-xr-x   - 1828421 hadoop          0 2019-04-02 06:01 PD_output
drwxr-xr-x   - 1828421 hadoop          0 2019-03-23 12:10 input
drwxr-xr-x   - 1828421 hadoop          0 2019-03-29 06:17 input_csv
drwxr-xr-x   - 1828421 hadoop          0 2019-03-23 12:46 input_word
drwxr-xr-x   - 1828421 hadoop          0 2019-03-29 06:35 output_csv
drwxr-xr-x   - 1828421 hadoop          0 2019-03-29 05:02 output_word
drwxr-xr-x   - 1828421 hadoop          0 2019-03-29 08:04 spark_output_word
1828421@sm1:~/Worksheet3$ hdfs dfs -put No_Quals.csv PD_input
1828421@sm1:~/Worksheet3$ hdfs dfs -ls /user/1828421/PD_input
Found 1 items
-rw-r--r--   1 1828421 hadoop     154835 2019-04-02 06:02 /user/1828421/PD_input
/No_Quals.csv
1828421@sm1:~/Worksheet3$
```

Figure 11: Putting file into directory

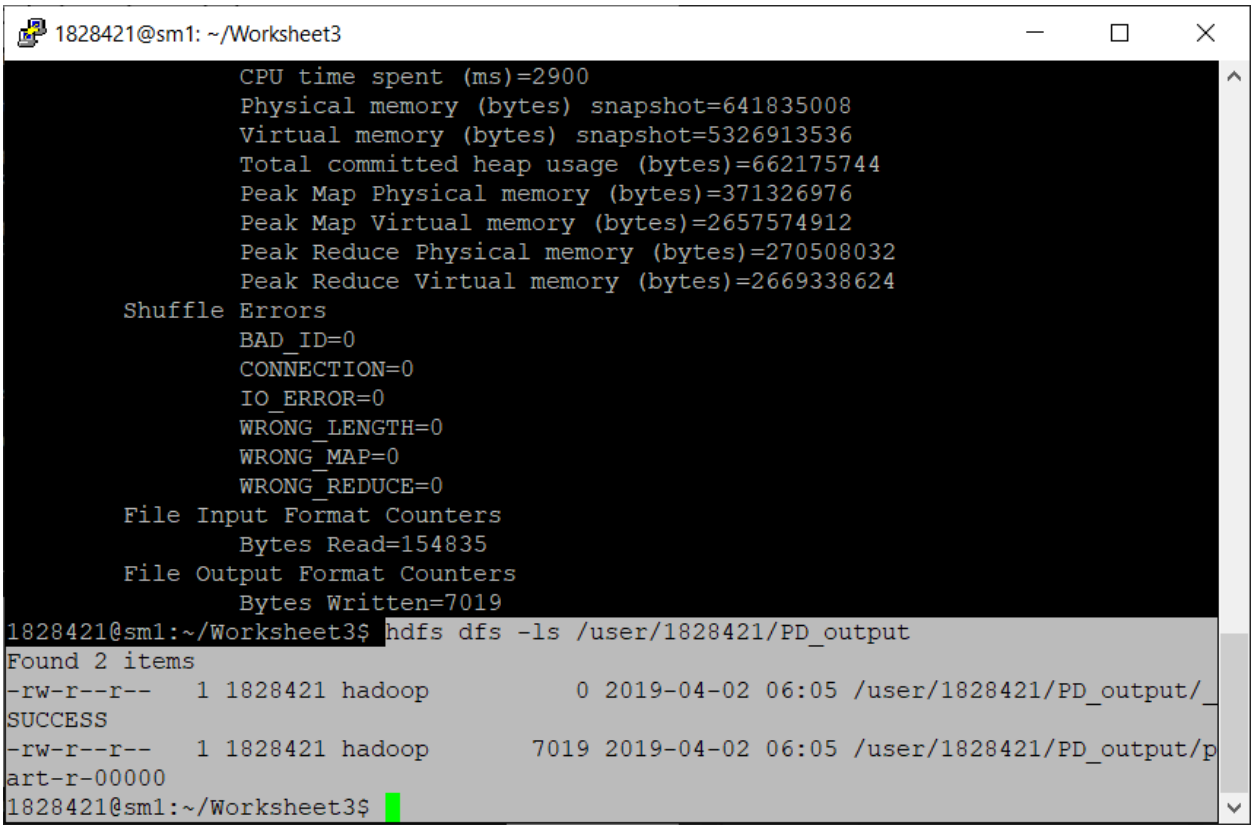
d. Running the program

```
1828421@sm1: ~/Worksheet3
1828421@sm1:~/Worksheet3$ hadoop jar NoQuals.jar NoQuals PD_input/No_Quals.csv P
D_output
2019-04-02 06:05:27,330 INFO client.RMProxy: Connecting to ResourceManager at lo
calhost/127.0.0.1:8050
2019-04-02 06:05:27,779 WARN mapreduce.JobResourceUploader: Hadoop command-line
option parsing not performed. Implement the Tool interface and execute your appl
ication with ToolRunner to remedy this.
2019-04-02 06:05:27,791 INFO mapreduce.JobResourceUploader: Disabling Erasure Co
ding for path: /tmp/hadoop-yarn/staging/1828421/.staging/job_1553250033923_0520
2019-04-02 06:05:27,979 INFO input.FileInputFormat: Total input files to process
 : 1
2019-04-02 06:05:28,021 INFO mapreduce.JobSubmitter: number of splits:1
2019-04-02 06:05:28,128 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1553250033923_0520
2019-04-02 06:05:28,129 INFO mapreduce.JobSubmitter: Executing with tokens: []
2019-04-02 06:05:28,291 INFO conf.Configuration: resource-types.xml not found
2019-04-02 06:05:28,291 INFO resource.ResourceUtils: Unable to find 'resource-ty
pes.xml'.
2019-04-02 06:05:28,347 INFO impl.YarnClientImpl: Submitted application applicat
ion_1553250033923_0520
2019-04-02 06:05:28,381 INFO mapreduce.Job: The url to track the job: http://sm1
:8088/proxy/application_1553250033923_0520/
2019-04-02 06:05:28,382 INFO mapreduce.Job: Running job: job_1553250033923_0520
2019-04-02 06:05:34,470 INFO mapreduce.Job: Job job_1553250033923_0520 running i
n uber mode : false
2019-04-02 06:05:34,471 INFO mapreduce.Job:  map 0% reduce 0%
```

Figure 12: Running jar file

e. Output directory files:

Displaying files inside the output folders.

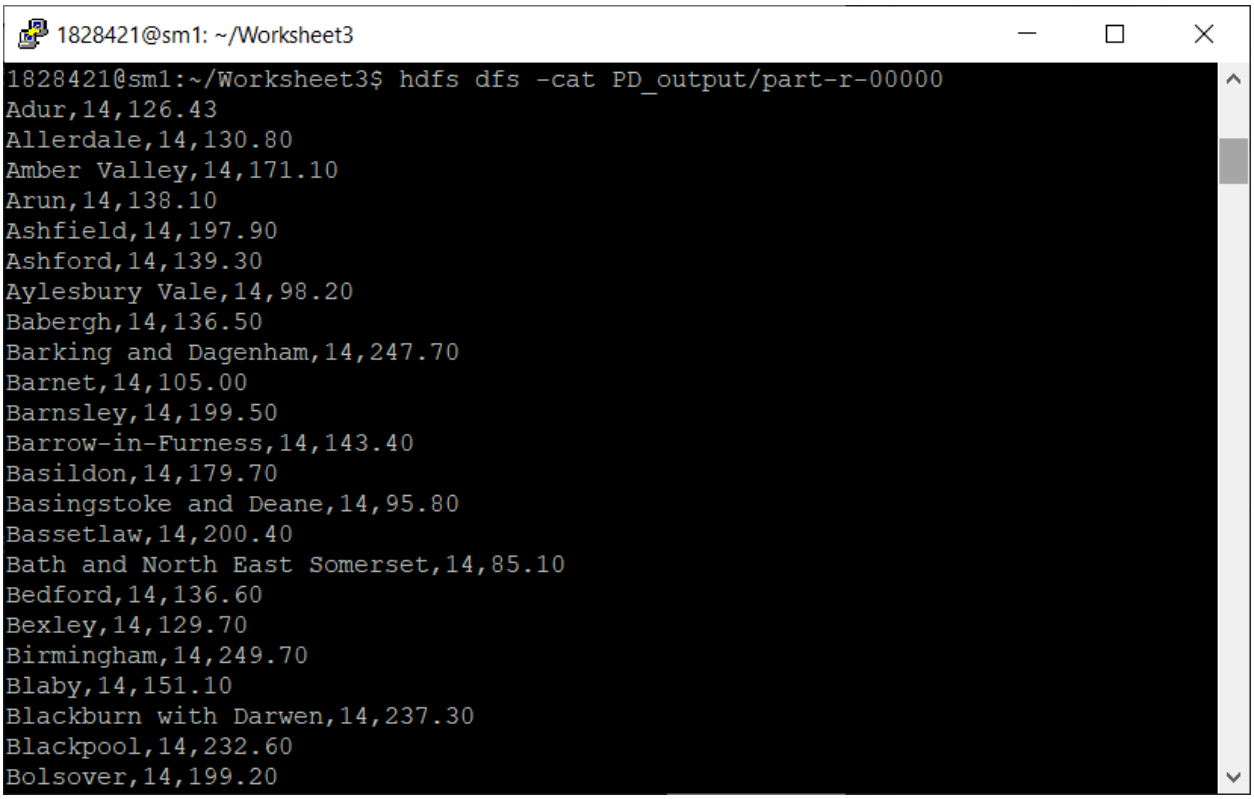


```
1828421@sm1: ~/Worksheet3
                CPU time spent (ms)=2900
                Physical memory (bytes) snapshot=641835008
                Virtual memory (bytes) snapshot=5326913536
                Total committed heap usage (bytes)=662175744
                Peak Map Physical memory (bytes)=371326976
                Peak Map Virtual memory (bytes)=2657574912
                Peak Reduce Physical memory (bytes)=270508032
                Peak Reduce Virtual memory (bytes)=2669338624
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=154835
        File Output Format Counters
                Bytes Written=7019
1828421@sm1:~/Worksheet3$ hdfs dfs -ls /user/1828421/PD_output
Found 2 items
-rw-r--r--   1 1828421 hadoop          0 2019-04-02 06:05 /user/1828421/PD_output/_
SUCCESS
-rw-r--r--   1 1828421 hadoop       7019 2019-04-02 06:05 /user/1828421/PD_output/p
art-r-00000
1828421@sm1:~/Worksheet3$
```

Figure 13: files in output folders

f. Displaying the result

To display the result more or -cat query can be used but here cat is used to display the result.



```
1828421@sm1: ~/Worksheet3
1828421@sm1:~/Worksheet3$ hdfs dfs -cat PD_output/part-r-00000
Adur,14,126.43
Allerdale,14,130.80
Amber Valley,14,171.10
Arun,14,138.10
Ashfield,14,197.90
Ashford,14,139.30
Aylesbury Vale,14,98.20
Babergh,14,136.50
Barking and Dagenham,14,247.70
Barnet,14,105.00
Barnsley,14,199.50
Barrow-in-Furness,14,143.40
Basildon,14,179.70
Basingstoke and Deane,14,95.80
Bassetlaw,14,200.40
Bath and North East Somerset,14,85.10
Bedford,14,136.60
Bexley,14,129.70
Birmingham,14,249.70
Blaby,14,151.10
Blackburn with Darwen,14,237.30
Blackpool,14,232.60
Bolsover,14,199.20
```

Figure 14: Result display

3. Apache Spark

Running apache spark. To run apache spark, **pyspark** command is used.

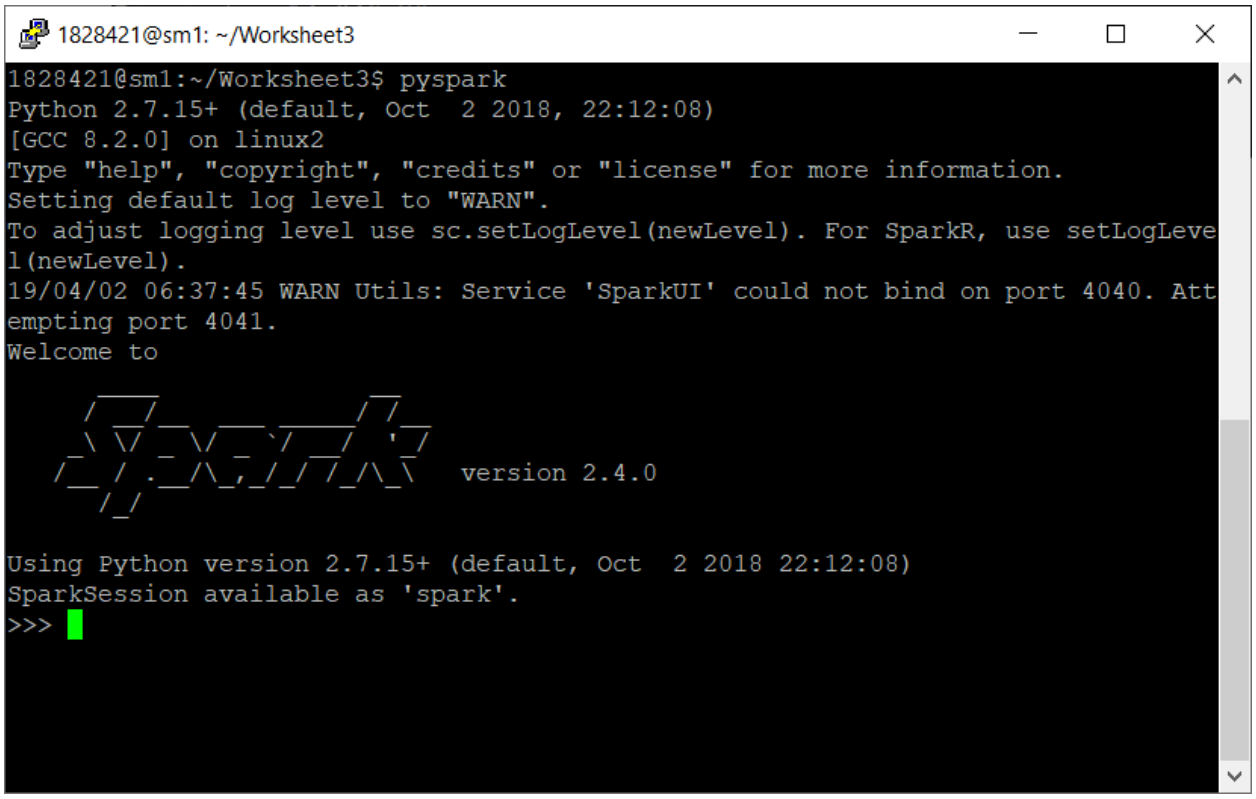


Figure 15: Running spark

a. Loading CSV file

Loading csv file from the hdfs and displaying the result.

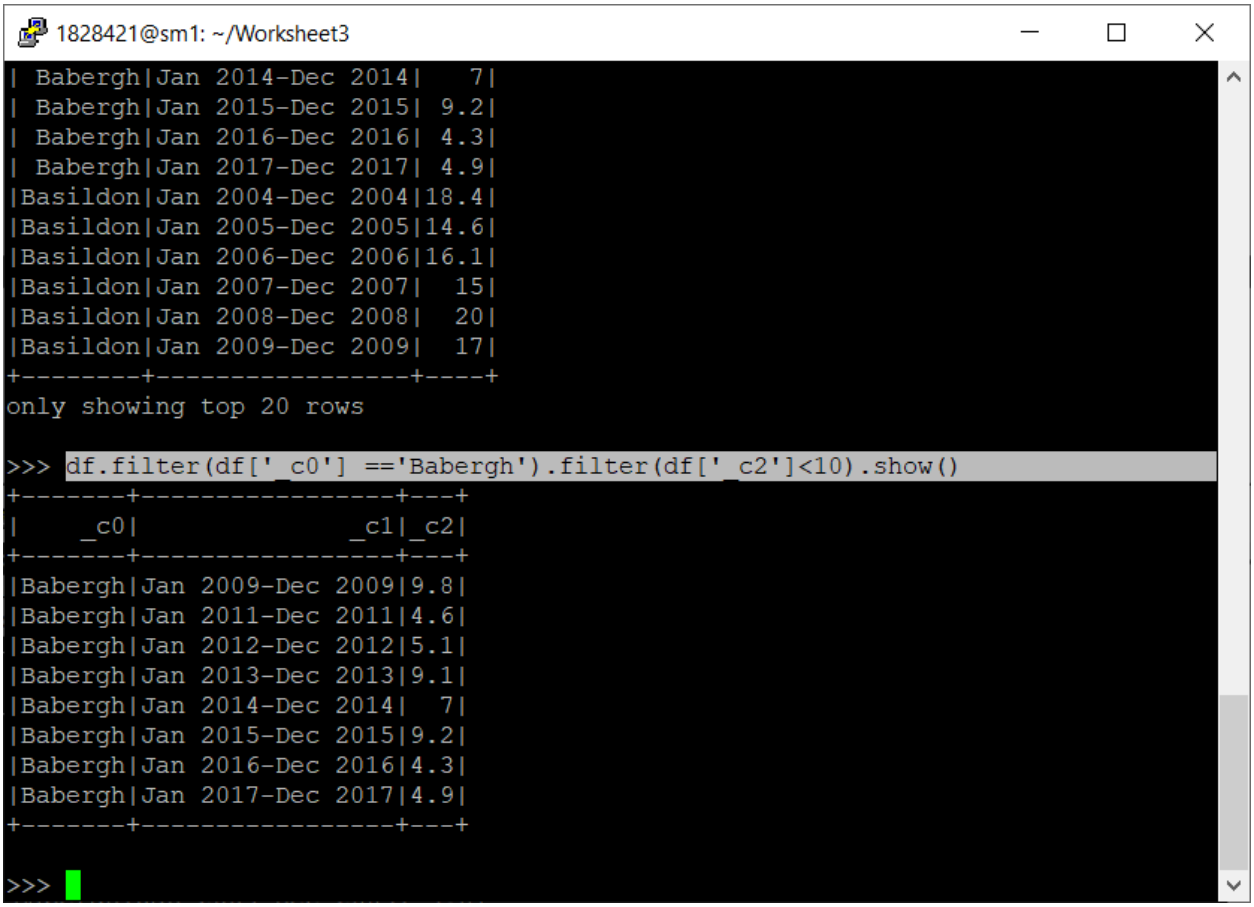
```
1828421@sm1: ~/Worksheet3
>>> df = spark.read.format("csv").load("hdfs://localhost:9000/user/1828421/PD_in
put/No_Quals.csv")
>>> df.show()
+--------+----------------+----+
|     _c0|             _c1| _c2|
+--------+----------------+----+
| Babergh|Jan 2004-Dec 2004|18.5|
| Babergh|Jan 2005-Dec 2005|12.6|
| Babergh|Jan 2006-Dec 2006|  14|
| Babergh|Jan 2007-Dec 2007|13.6|
| Babergh|Jan 2008-Dec 2008|11.8|
| Babergh|Jan 2009-Dec 2009| 9.8|
| Babergh|Jan 2010-Dec 2010|  12|
| Babergh|Jan 2011-Dec 2011| 4.6|
| Babergh|Jan 2012-Dec 2012| 5.1|
| Babergh|Jan 2013-Dec 2013| 9.1|
| Babergh|Jan 2014-Dec 2014|   7|
| Babergh|Jan 2015-Dec 2015| 9.2|
| Babergh|Jan 2016-Dec 2016| 4.3|
| Babergh|Jan 2017-Dec 2017| 4.9|
|Basildon|Jan 2004-Dec 2004|18.4|
|Basildon|Jan 2005-Dec 2005|14.6|
|Basildon|Jan 2006-Dec 2006|16.1|
|Basildon|Jan 2007-Dec 2007|  15|
|Basildon|Jan 2008-Dec 2008|  20|
|Basildon|Jan 2009-Dec 2009|  17|
+--------+----------------+----+
only showing top 20 rows
```

Figure 16: CSV load

b. Data manipulation query

Manipulation of data is shown below of csv as well as sql.

i. CSV:



```
1828421@sm1: ~/Worksheet3
| Babergh|Jan 2014-Dec 2014|   7|
| Babergh|Jan 2015-Dec 2015| 9.2|
| Babergh|Jan 2016-Dec 2016| 4.3|
| Babergh|Jan 2017-Dec 2017| 4.9|
|Basildon|Jan 2004-Dec 2004|18.4|
|Basildon|Jan 2005-Dec 2005|14.6|
|Basildon|Jan 2006-Dec 2006|16.1|
|Basildon|Jan 2007-Dec 2007|  15|
|Basildon|Jan 2008-Dec 2008|  20|
|Basildon|Jan 2009-Dec 2009|  17|
+--------+----------------+----+
only showing top 20 rows

>>> df.filter(df['_c0'] =='Babergh').filter(df['_c2']<10).show()
+-------+----------------+---+
|    _c0|              _c1|_c2|
+-------+----------------+---+
|Babergh|Jan 2009-Dec 2009|9.8|
|Babergh|Jan 2011-Dec 2011|4.6|
|Babergh|Jan 2012-Dec 2012|5.1|
|Babergh|Jan 2013-Dec 2013|9.1|
|Babergh|Jan 2014-Dec 2014|  7|
|Babergh|Jan 2015-Dec 2015|9.2|
|Babergh|Jan 2016-Dec 2016|4.3|
|Babergh|Jan 2017-Dec 2017|4.9|
+-------+----------------+---+

>>>
```

Figure 17: CSV filter

ii. SQL:

Here again fresh file is loaded from the hdfs so that the one manipulation may not affect another, it is good to have fresh and separate file. To load sql, at first it has to be converted into view. Then SQL query is implemented.

```
1828421@sm1: ~
>>> dfNew = spark.read.format("csv").load("hdfs://localhost:9000/user/1828421/PD_
input/No_Quals.csv")
>>> dfNew.createOrReplaceTempView("NoQuals")
>>> dfNew = spark.sql("SELECT _c0 AS City, count(_c1) AS Total_Year, sum(_C2) AS
Total_Value FROM NoQuals GROUP BY _c0").show()
19/04/03 04:48:29 WARN ObjectStore: Failed to get database global_temp, returning
 NoSuchObjectException
+--------------------+----------+------------------+
|                City|Total_Year|       Total_Value|
+--------------------+----------+------------------+
|           Worcester|        14|186.59999999999997|
|           Charnwood|        14|108.29999999999998|
|      North Kesteven|        14|              93.2|
|       Epping Forest|        14|             147.2|
|             Waveney|        14|             191.9|
|                Arun|        14|             138.1|
|              Stroud|        14| 88.19999999999999|
|              Maldon|        14|             208.2|
|          New Forest|        14|             103.6|
|           Sedgemoor|        14|138.79999999999998|
|           Guildford|        14| 88.69999999999999|
|            Worthing|        14|              91.9|
|             Fareham|        14|              70.7|
|   Bristol - City of|        14|             135.9|
|Central Bedfordshire|        14|             120.6|
|      North Tyneside|        14|             148.3|
|              Bolton|        14|             195.4|
|      Wellingborough|        14|194.79999999999998|
|        Surrey Heath|        14| 85.59999999999997|
|              Slough|        14|             155.0|
+--------------------+----------+------------------+
only showing top 20 rows

>>>
```

Figure 18: SQL filter

4. Advantage and Disadvantage

a. Advantage:

Fault tolerance:

Hadoop is used for processing large number of data or files. While processing sometimes some issue can arise and node may fail to process. So, in this case the whole operation will be stopped, and the task will be incomplete. Therefore, in order to maintain this, Hadoop uses technique of Fault tolerance which resolve this problem by neglecting or overcoming that module. There are many techniques for this like check-point and recovery, heartbeat message and so on but mainly **Data Replication** and or **Shredding** is used to maintain fault tolerance.

In data replication, the copy of same data is stored in multiple places where in shredding, data are split and stored in different places either row or column wise (Mugunthan, 2015).

b. Disadvantage:

Slow and not suitable for small data:

Hadoop is built for processing large data or files. So, it goes from mapper to reducer and gives the output. It is suitable for large number of data but not suitable for small data since it has to pass through mapper and reducer which make the performance very slow. On the other hand, Hadoop is slower than Apache Spark as well which is one of the major limitations of Hadoop.