1. Java and Hadoop

Renaming java class, mapper and reducer as NoQuals, Mapper_2050214 and Reducer_2050214

```
public class NoQuals {

public static class Mapper_2050214 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
       } // Mapper_2050214

       public static class Reducer_2050214 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;
                      }
                      // only get here if a number
                return true;
              } // isInteger

              public void reduce(Text key, Iterable<Text> values, Context context)
```

2. Java and Hadoop

Starting Hadoop

```
yogesh@yogesh-Inspiron-5570:~$ sudo su - hdoop
hdoop@yogesh-Inspiron-5570:~$ cd hadoop-3.2.2/
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2$ cd sbin
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2/sbin$ ./st
start-all.cmd         start-yarn.cmd        stop-dfs.sh
start-all.sh          start-yarn.sh         stop-secure-dns.sh
start-balancer.sh     stop-all.cmd          stop-yarn.cmd
start-dfs.cmd         stop-all.sh           stop-yarn.sh
start-dfs.sh          stop-balancer.sh
start-secure-dns.sh   stop-dfs.cmd
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2/sbin$ ./st
start-all.cmd         start-yarn.cmd        stop-dfs.sh
start-all.sh          start-yarn.sh         stop-secure-dns.sh
start-balancer.sh     stop-all.cmd          stop-yarn.cmd
start-dfs.cmd         stop-all.sh           stop-yarn.sh
start-dfs.sh          stop-balancer.sh
start-secure-dns.sh   stop-dfs.cmd
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2/sbin$ ./st
start-all.cmd         start-yarn.cmd        stop-dfs.sh
start-all.sh          start-yarn.sh         stop-secure-dns.sh
start-balancer.sh     stop-all.cmd          stop-yarn.cmd
start-dfs.cmd         stop-all.sh           stop-yarn.sh
start-dfs.sh          stop-balancer.sh
start-secure-dns.sh   stop-dfs.cmd
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2/sbin$ ./start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [yogesh-Inspiron-5570]
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2/sbin$ ./start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2/sbin$ jps
6500 NameNode
7156 ResourceManager
7318 NodeManager
7689 Jps
6874 SecondaryNameNode
6668 DataNode
```

Copying Csv file and java file to Hadoop user's folder

```
yogesh@yogesh-Inspiron-5570:~/Documents/IA2$ sudo cp ~yogesh/Documents/IA2/NoQuals.java ~hdoop/hadoop-3.2.2/No
Quals/
[sudo] password for yogesh:
yogesh@yogesh-Inspiron-5570:~/Documents/IA2$ sudo cp ~yogesh/Documents/IA2/No_Quals.csv ~hdoop/hadoop-3.2.2/No
Quals/
```

Compiling java program

```
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2$ mkdir NoQuals
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2$ cd NoQuals/
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2/NoQuals$ javac -classpath $(hadoop classpath) NoQuals.java
```

Creating NoQuals jar file

```
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2/NoQuals$ jar cf NoQuals.jar NoQuals*.class
```

Creating input_2050214 directory to store csv file

```
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2/NoQuals$ hdfs dfs -mkdir input_2050214
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2/NoQuals$ hdfs dfs -put No_Quals.csv in
put_2050214
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2/NoQuals$ hadoop jar NoQuals.jar NoQual
s input_2050214/No_Quals.csv output_2050214
2022-04-12 07:37:51,874 INFO client.RMProxy: Connecting to ResourceManager at /1
27.0.0.1:8032
2022-04-12 07:37:52,100 WARN mapreduce.JobResourceUploader: Hadoop command-line
option parsing not performed. Implement the Tool interface and execute your appl
ication with ToolRunner to remedy this.
2022-04-12 07:37:52,139 INFO mapreduce.JobResourceUploader: Disabling Erasure Co
ding for path: /tmp/hadoop-yarn/staging/hdoop/.staging/job_1649763240228_0001
2022-04-12 07:37:52,457 INFO input.FileInputFormat: Total input files to process
 : 1
2022-04-12 07:37:52,747 INFO mapreduce.JobSubmitter: number of splits:1
2022-04-12 07:37:53,172 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1649763240228_0001
```

Running the file

Check if the output_2050214 is created.

```
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2/NoQuals$ hdfs dfs -ls output_2050214
Found 2 items
-rw-r--r--   1 hdoop supergroup          0 2022-04-12 07:38 output_2050214/_SUCC
ESS
-rw-r--r--   1 hdoop supergroup       7019 2022-04-12 07:38 output_2050214/part-
r-00000
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2/NoQuals$ hdfs dfs -cat output_2050214/
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
```

Retrieving the result_2050214 file and viewing the result.

```
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2/NoQuals$ hdfs dfs -get output_2050214/
part-r-00000 results_2050214.csv
hdoop@yogesh-Inspiron-5570:~/hadoop-3.2.2/NoQuals$ more results_2050214.csv
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
```

3. Apache Spark

Starting spark using “pyspark” command.

```
yogesh@yogesh-Inspiron-5570:~/Documents/IA2$ pyspark
Python 3.8.10 (default, Mar 15 2022, 12:22:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
df = spark.read.csv(22/04/12 10:41:22 WARN Utils: Your hostname, yogesh-Inspiron
-5570 resolves to a loopback address: 127.0.1.1; using 192.168.0.100 instead (on
 interface enp2s0)
22/04/12 10:41:22 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/ho
me/yogesh/spark/jars/spark-unsafe_2.12-3.2.1.jar) to constructor java.nio.Direct
ByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.u
nsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflect
ive access operations
WARNING: All illegal access operations will be denied in a future release
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
```

Loading CSV data in apache spark

```
SparkSession available as 'spark'.
>>> df = spark.read.csv("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|
```

Using groupby query to display data.

```
>>> df.groupBy("_c1").count().show()
+-----------------+-----+
|              _c1|count|
+-----------------+-----+
|Jan 2011-Dec 2011|  324|
|Jan 2015-Dec 2015|  324|
|Jan 2010-Dec 2010|  324|
|Jan 2004-Dec 2004|  324|
|Jan 2008-Dec 2008|  324|
|Jan 2017-Dec 2017|  324|
|Jan 2016-Dec 2016|  324|
|Jan 2005-Dec 2005|  324|
|Jan 2007-Dec 2007|  324|
|Jan 2012-Dec 2012|  324|
|Jan 2009-Dec 2009|  324|
|Jan 2006-Dec 2006|  324|
|Jan 2013-Dec 2013|  324|
|Jan 2014-Dec 2014|  324|
+-----------------+-----+
```

Converting to dataframes using SQL

```
>>> df.createOrReplaceTempView("No_Quals")
```

Using filter method to display data

```
>>> dfNoQuals.filter(dfNoQuals['Babergh'] == '7').show()
+-------+-----------------+----+
|Babergh|Jan 2004-Dec 2004|18.5|
+-------+-----------------+----+
+-------+-----------------+----+
```

Displaying total number of data using count.

```
>>> df.createOrReplaceTempView("NoQuals")
>>> sqlDf = spark.sql('SELECT _c0.Basildon FROM NoQuals _c0.Basildon = 11)
  File "<stdin>", line 1
    sqlDf = spark.sql('SELECT _c0.Basildon FROM NoQuals _c0.Basildon = 11)
                                                                       ^
SyntaxError: EOL while scanning string literal
>>> sqlDF.show()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sqlDF' is not defined
>>> sqlDF = spark.sql("SELECT count(*) AS NoQuals_count FROM NoQUals").show()
+--------------+
```

4. Advantage of using Hadoop for handling Big Data

Scalability

Hadoop is a model that is extremely scalable. A significant volume of data is split across several low-cost devices in a clustered and analyzed in simultaneously. According to the needs of the business, the quantity of such computers or nodes could be raised or lowered. Conventional RDBMS (Relational DataBase Management System) systems are incapable of handling enormous volumes of data.

5. Disadvantage of using Hadoop for handling Big Data

Problems with Small files

Hadoop can handle a huge quantity of data in a brief amount of time. Hadoop data is stored in file blocks ranging in size from 128Megabytes (by default) to 256Megabytes. Hadoop malfunctions when a huge number of tiny files must be accessed. The Namenode is overburdened by so many little files, making it difficult to handle with.