**Project Report:**

1. **Dataset Information:**

**US Consumer Finance Complaints**

CFPB sends thousands of consumers’ complaints about financial products and services to companies for response. Those complaints are published here after the company responds or after 15 days, whichever comes first.

By submitting a complaint, consumers can be heard by financial companies, get help with their own issues, and help others avoid similar ones. These complaint provides insight into problems that people are experiencing, helping us identify inappropriate practices and allowing us to stop them before they become major issues. Idea behind this is better outcomes for consumers, and a better financial marketplace for everyone.

The Consumer Complaint Database is a collection of 669,721 complaints, on a range of consumer financial products and services, sent to nearly 3,000 companies for response.

**Link:** https://www.kaggle.com/cfpb/us-consumer-finance-complaints

**Reporting analysis:** The reporting analysis is performed from potential consumer and company’s performance analysis point of view. MapReduce job has been run on Hadoop as well as AWS.

Big data Ecosystem:

Tableau

(Visualization)

AWS RedShift

(Dataware House)

AWS S3

AWS EMR Cluster

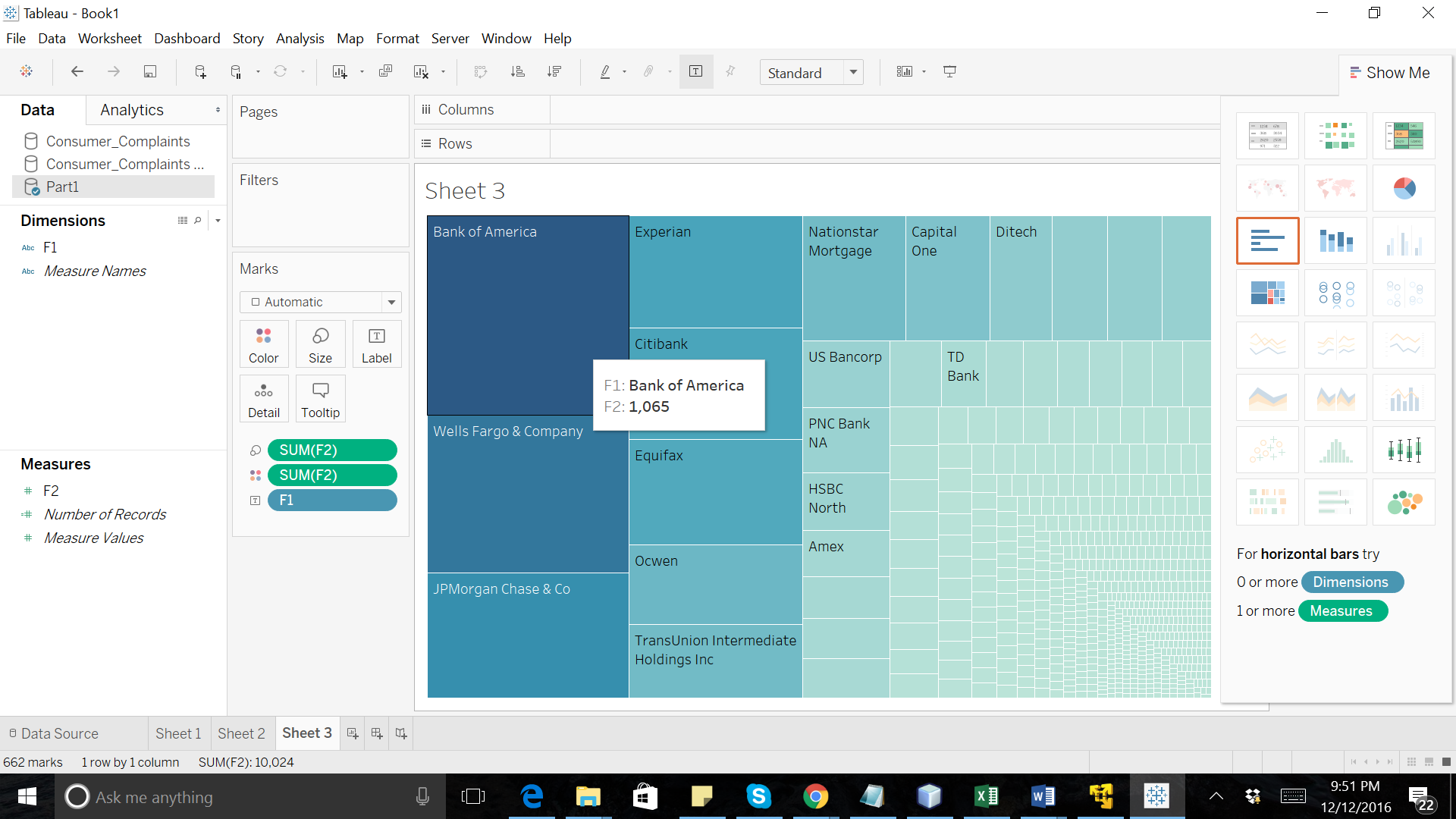
* Pig
* Hive
* Spark
* MapReduce

**Consumer point of view**:

1. **Company review analysis:**

While investing for a product of a company, consumer will first investigate how many complaints are associated with the company. (company review). Company - No of complaints (In this pattern, company is the key and no of complaints will be output from reducer)

Visualization on tableau:

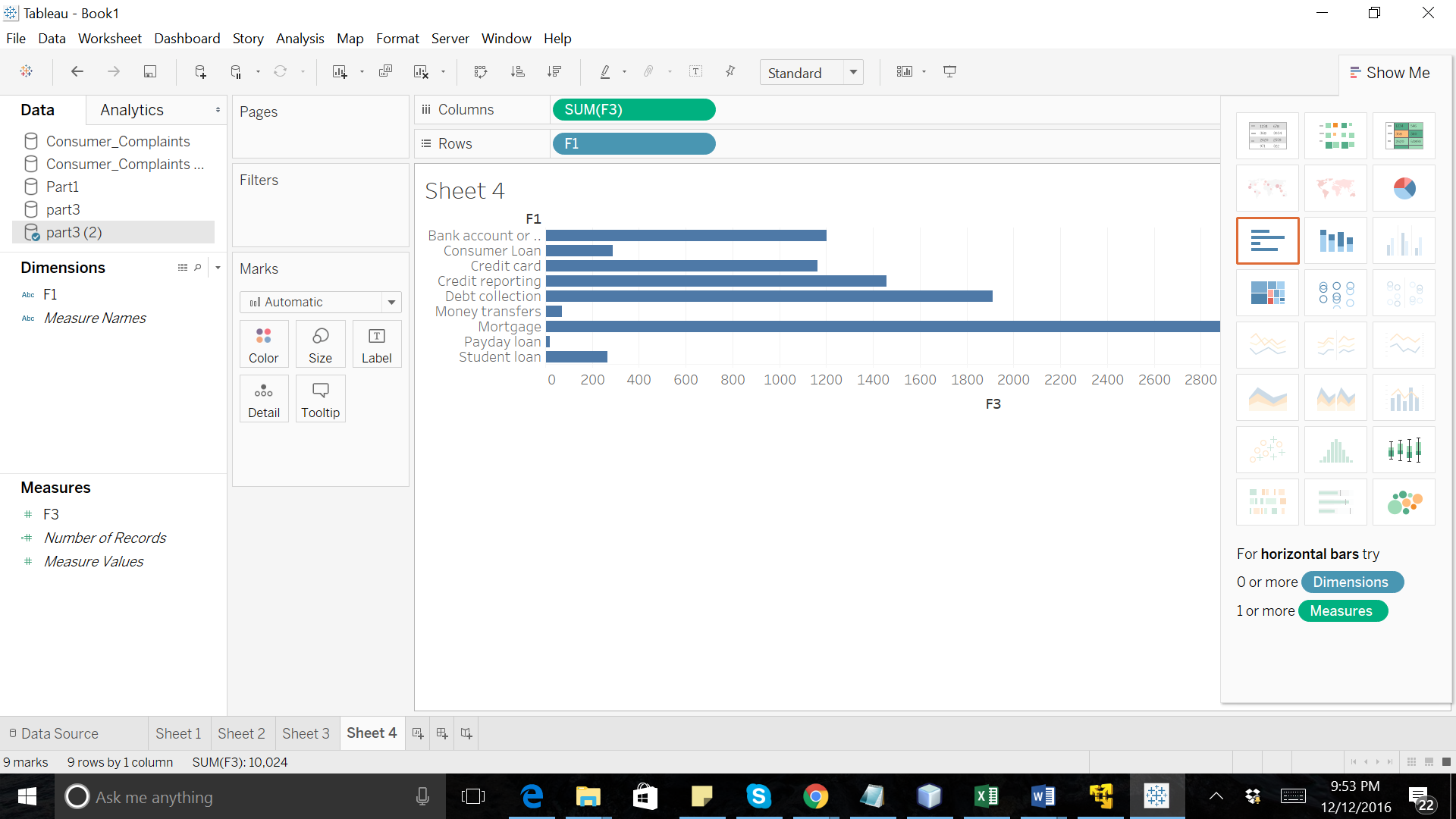


1. **Top 10 products with maximum number of complaints:**

Consumer would like to see products with maximum number of complaints against them.

this pattern I have implementing secondary sorting to fetch top 10 products which have maximum number of complaints against them. Chaining of two mapreduce jobs has been implemented.

Visualization:

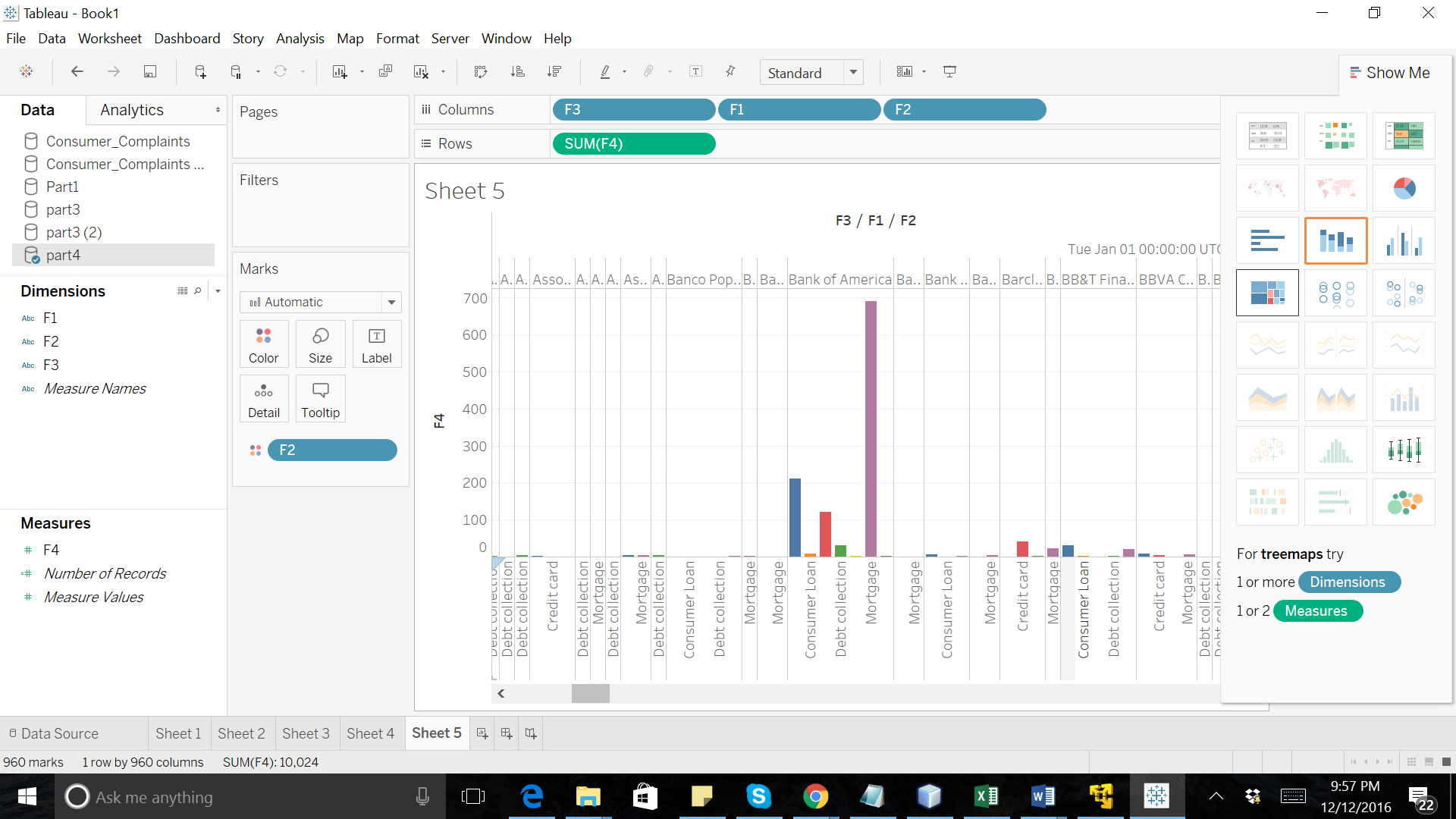


1. **Yearly analysis of company’s products**

Consumer can perform analysis for particular company and it’s particular product, what is the review till now? How many complaints are associated with company's each product in a year

For this pattern, company and product as composite key, tuple having attributes i.e year, no of complaints as value of reducer.

Visualization:



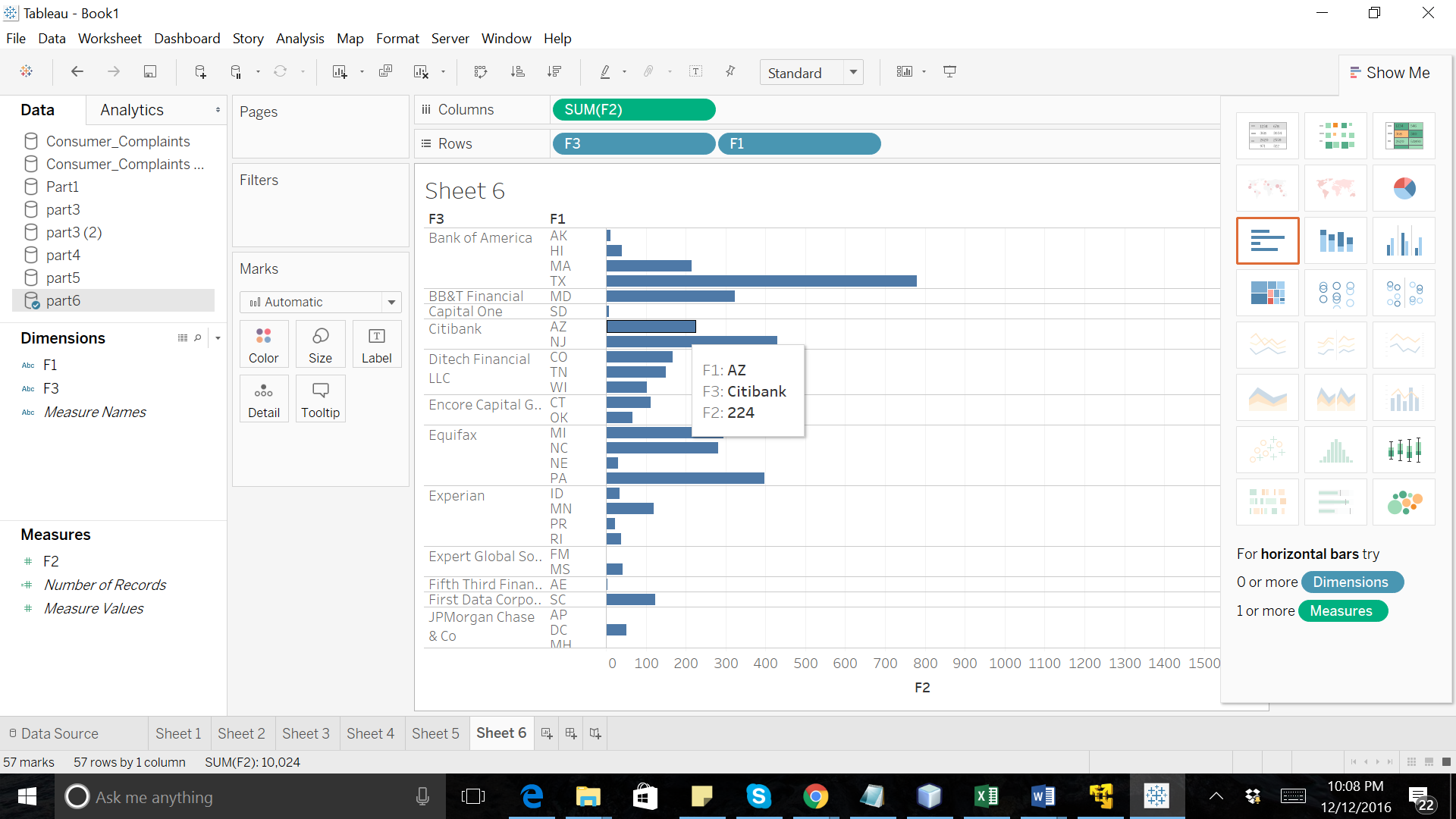
1. **Bloom filter to find products with specific issues:-**

Consumer would like to know the products which will be associated with specific issues. Using bloom filtering pattern to fetch products which will be having specific issues associated with it. Data will be compared against the product defined with

1. **State wise company analysis-**

How many complaints are associated with the company in specific state. In this pattern, key is state name and value will be tuple having company and number of complaints as attributes.

Visualization:



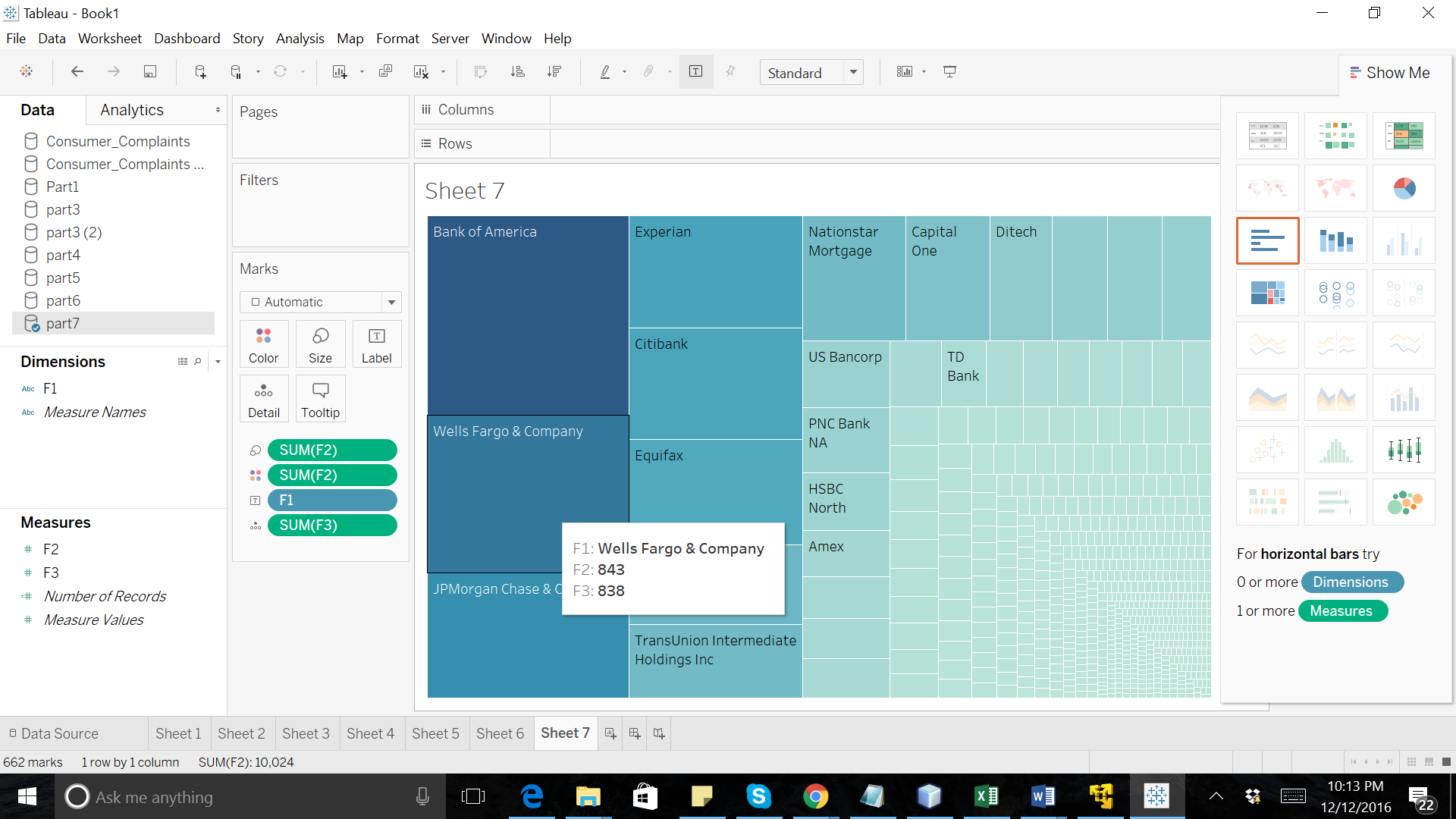
**Company owner point of view:**

**6.** **Timely response analysis:**

Company -No of complaints associated with company among which how many got timely response from company.

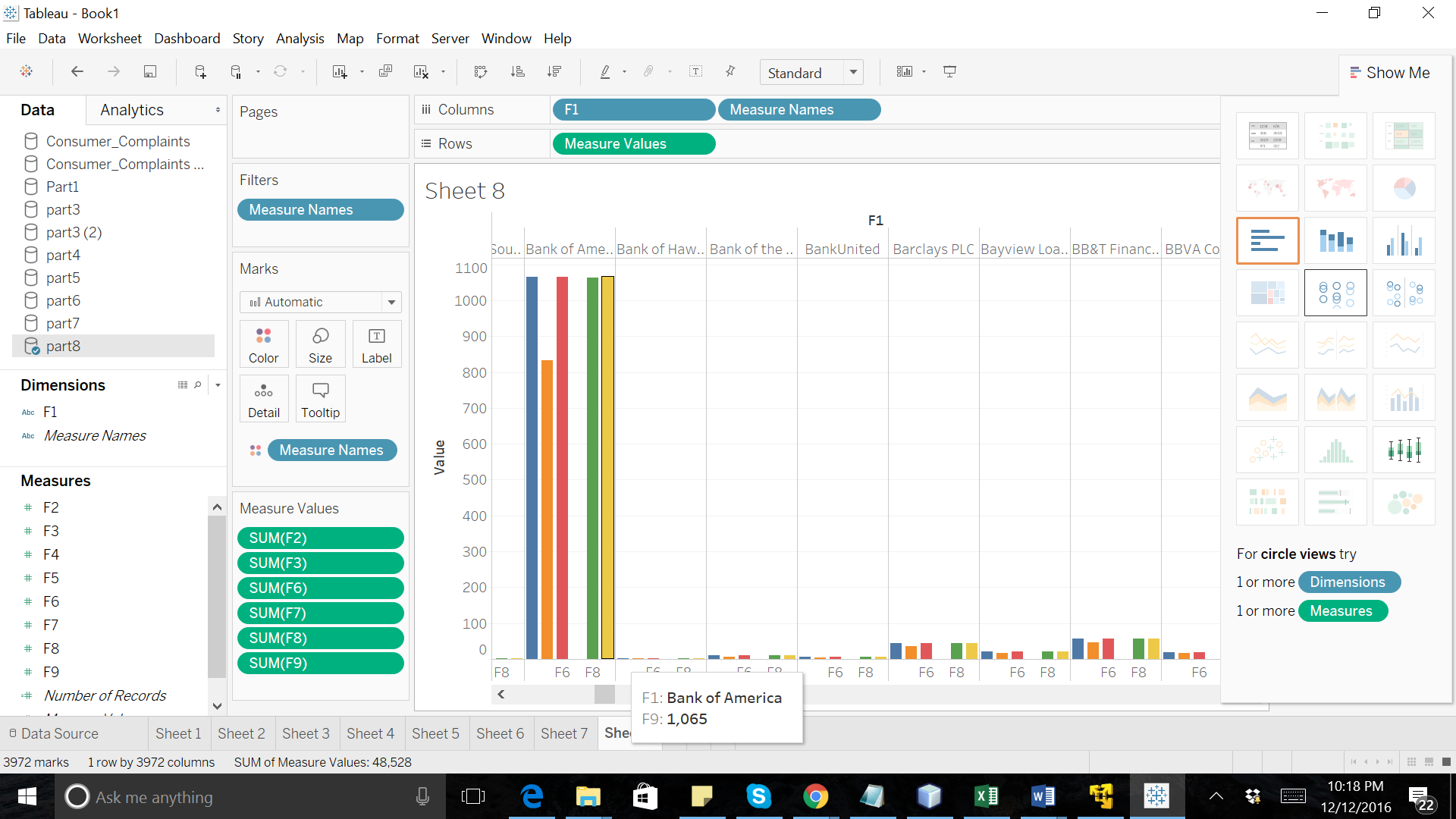
Company head can see how many complaints were received and among them which were addressed in timely manner. If the complaint is addressed within 15 days from the day it is posted, it will be timely response.

Visualization:



8. **Complaints submission mode analysis:**

For each company, count of complaints received via each mode. Company department can see which mode of submission is efficient for consumer to submit the complaints. They can improve on other modes by the statistical analysis.



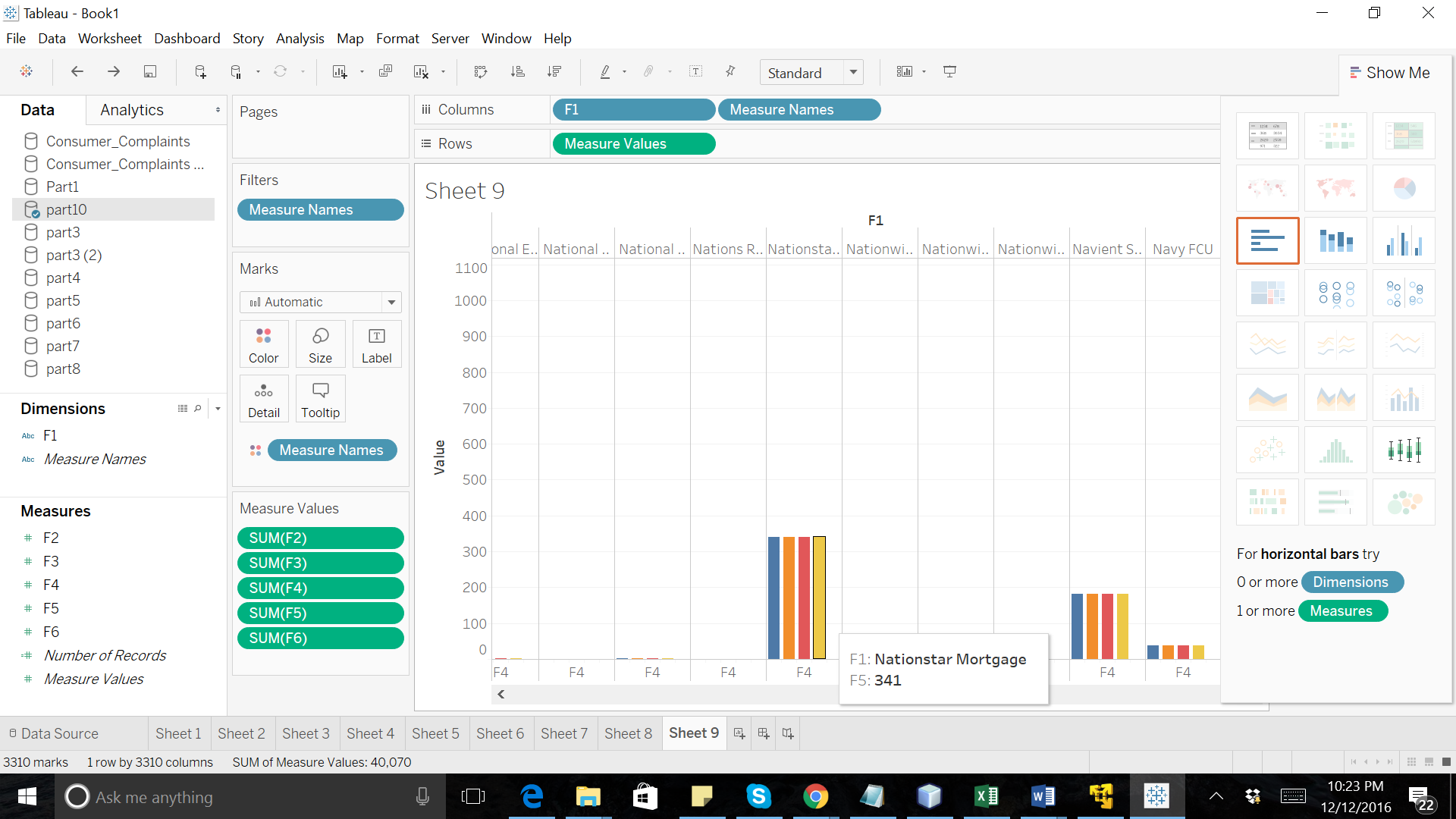
**9. Clustering by state:**

Portioning the complaints by state so that data will be organized according to state.

If a consumer belongs to particular state, he would refer to data particular to the state rather than the whole dataset. Dataset is being clustered by state.

**10. Complaint –status analysis:**

Company owner can see the reports of complaints with their statuses (closed with explanation, closed with monetary, closed with non-monetary, untimely).In this pattern, company is key and tuple having counts for different status will be value from reducer.



**12. Inverted index:**

If employer searches with issue, he will get all complaint ids associated with that issue. In this pattern, issue will be the key and all related complaint ids will be the values.

**Pig script:**

queryData3 = LOAD 's3://trafficdatacollection123/consumer2.csv' USING PigStorage(',') AS (Datereceived:chararray,company:chararray,state:chararray,product:chararray,subproduct:chararray,issue:chararray,subissue:chararray,timelyresponse:chararray,submittedvia:chararray,zip:int,datesent:chararray,companyresponse:chararray,consumerdispute:chararray,complaintid:int);

hdata = FILTER queryData3 BY (timelyresponse == No);

gdata = Group hdata BY company;

countData = FOREACH gdata GENERATE group, COUNT(hdata) AS topDestination;

orderedData = ORDER countData by topDestination DESC;

finalData = LIMIT orderedData 10;

STORE finalData INTO 's3://trafficdatacollection123/final\_output3';

**Hive script:**

CREATE EXTERNAL TABLE IF NOT EXISTS compliants\_fin(

DateObject Date,

Company STRING,

State STRING,

Product STRING,

Subproduct STRING,

Issue STRING,

Subissue STRING,

Timelyresponse STRING,

Submittedvia String,

Consumercomplaintnarrative STRING,

Companypublicresponse STRING,

zip STRING,

tags STRING,

consumerconsent string,

DateSent Date,

Companyresponse STRING,

ConsumerDispute STRING,

complaintId STRING

)ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

STORED AS TEXTFILE

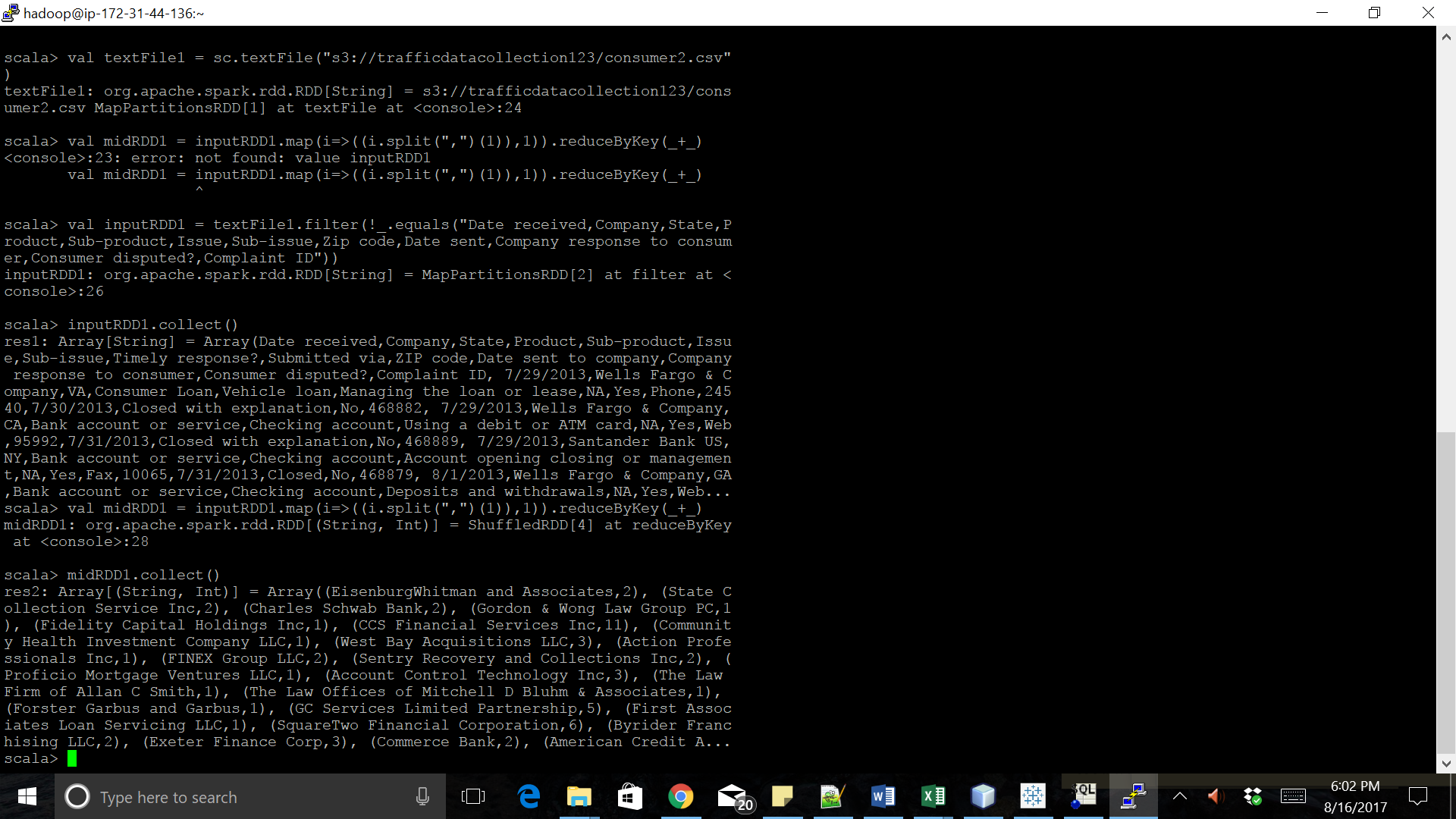
LOCATION '${INPUT}/data1/';

INSERT OVERWRITE DIRECTORY '${OUTPUT}/data1/'

SELECT company, count(\*) count FROM compliants\_fin

GROUP BY company;

Spark:



2.Number ofcomplaints of company per state:

Val textFile1 = sc.textFile(“s3://trafficdatacollection123/consumer2.csv”)

Val midRDD1 = inputRDD1.map(i=>i.split(“,”)).map(c=>(c(1),c(2)))

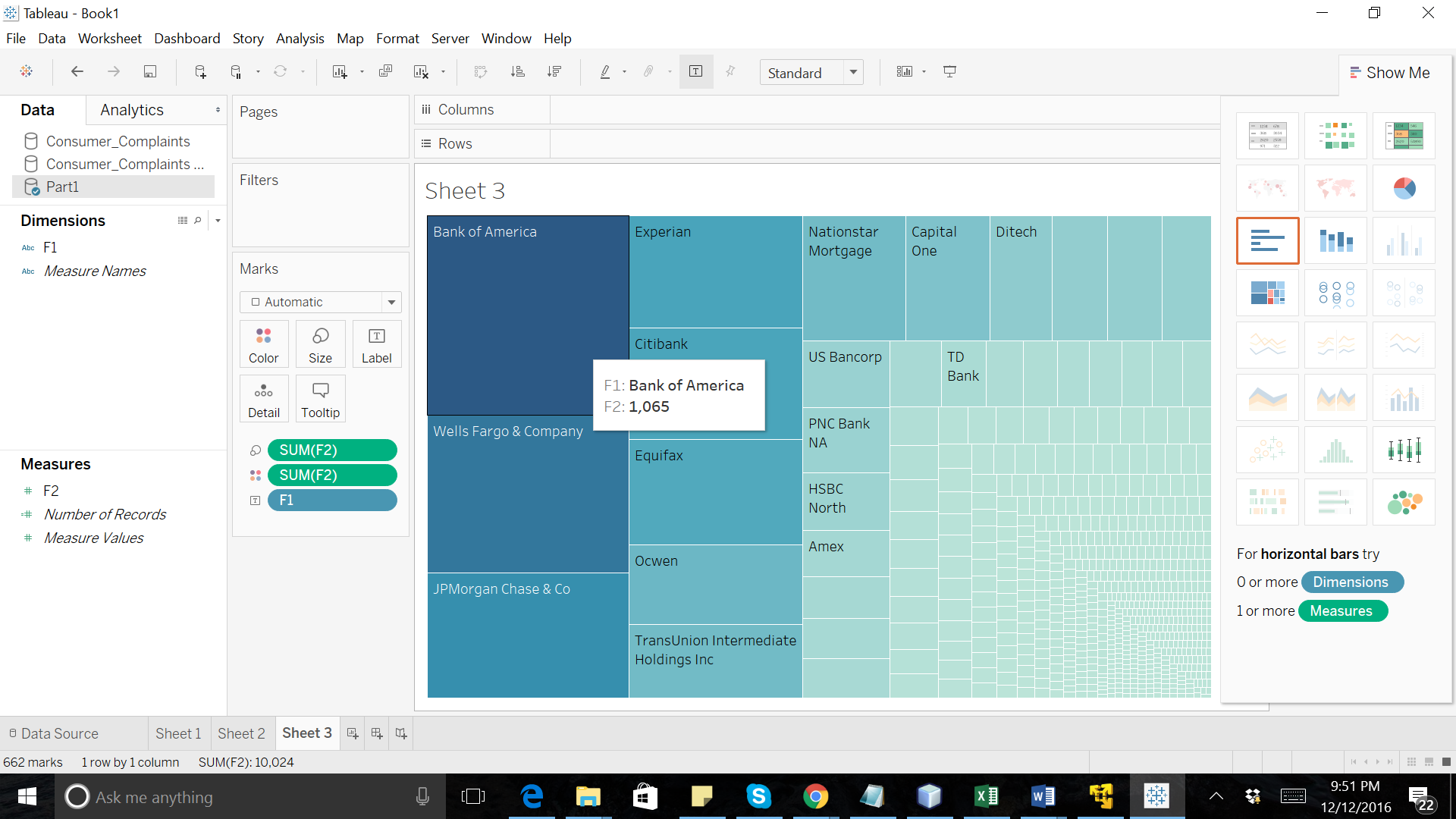
Val out2 = midRDD1.countByValue()

Sc.parallelize(out4.toSeq).saveAsObjectFile(“s3://trafficdatacollection123/sparkou3”)

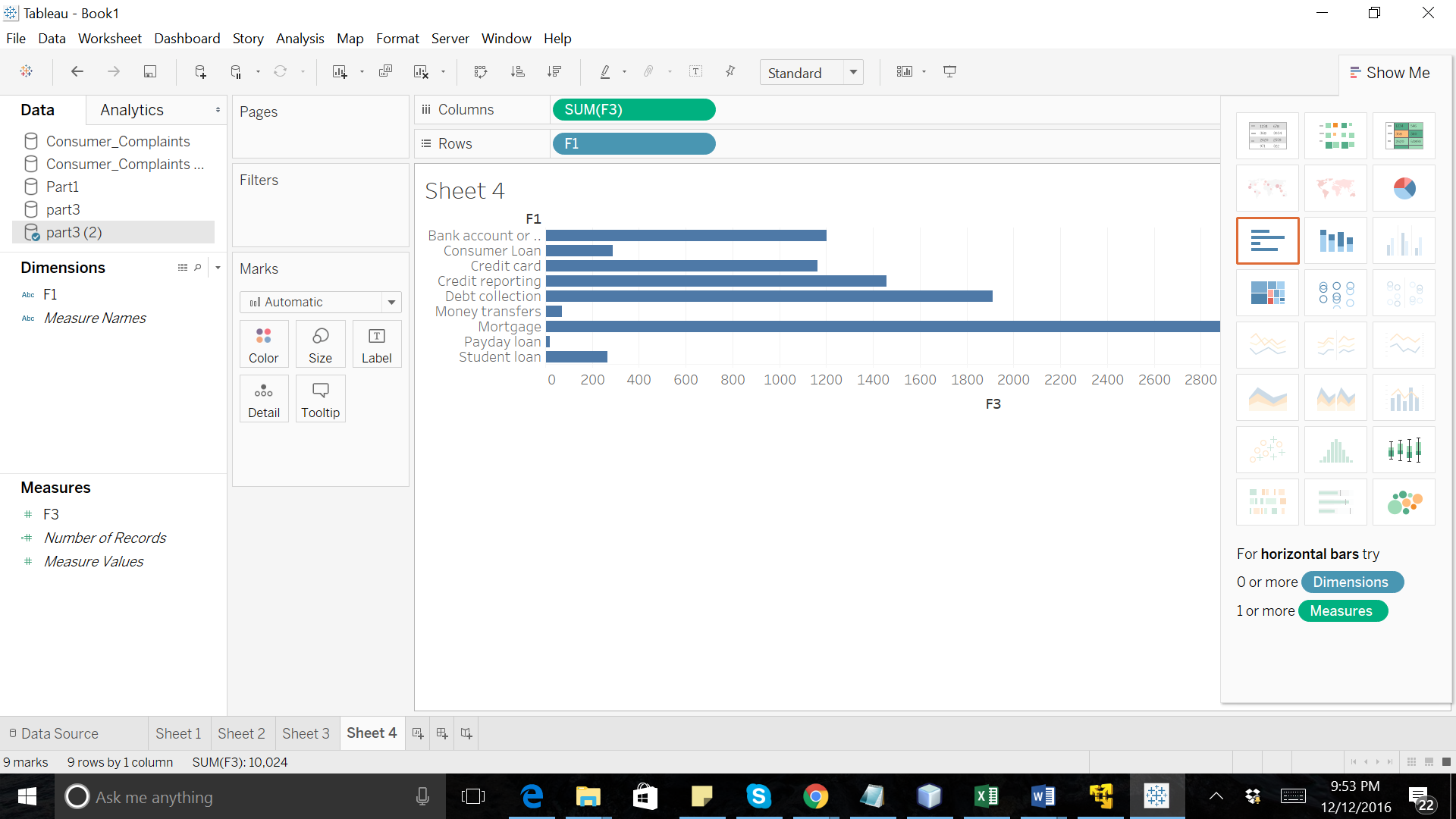
Machine learning algorithm: Using multiclass classification aws machine learning service.

**Visualization in tablau:**

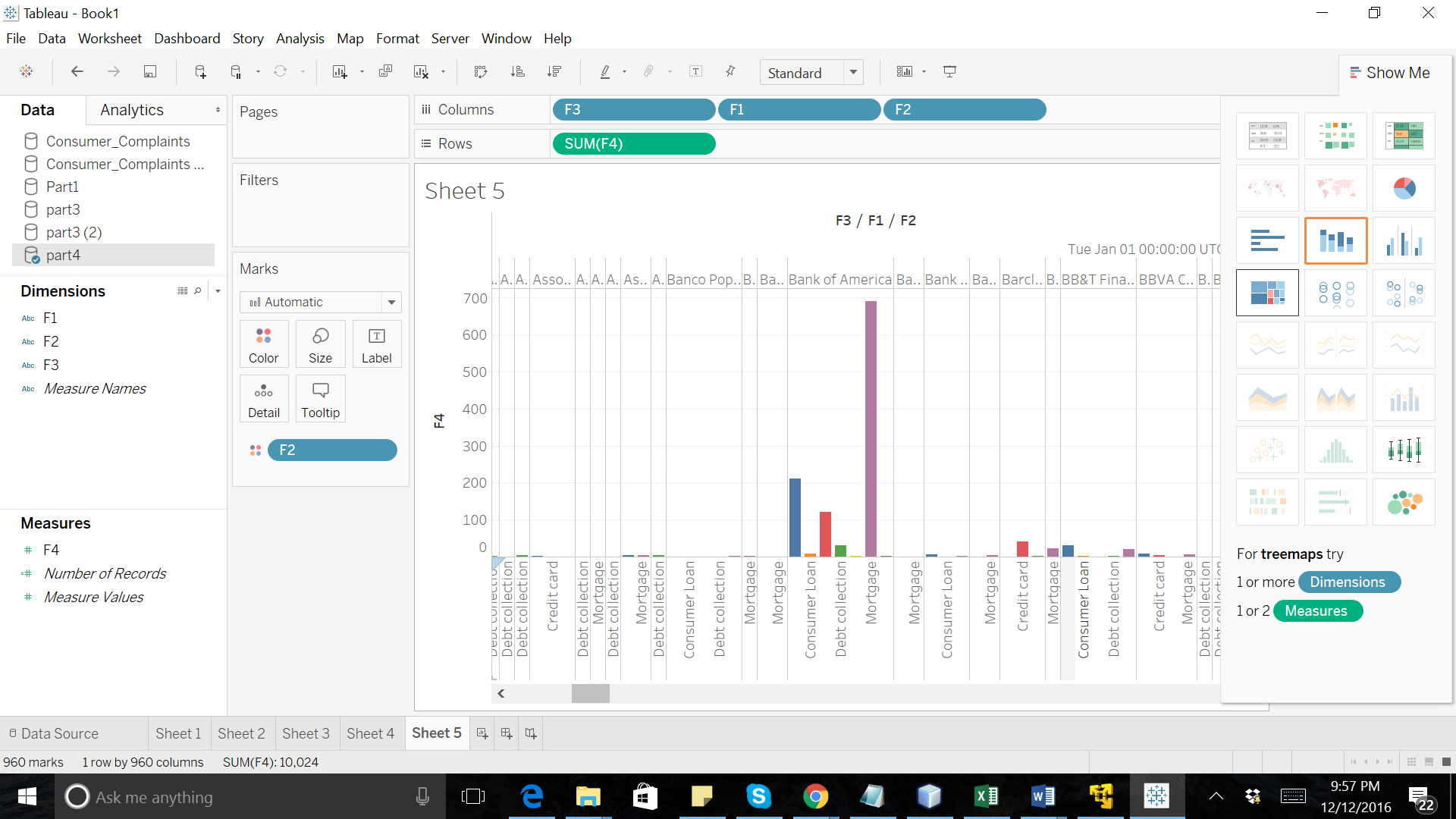
1.Company and number of complaints:



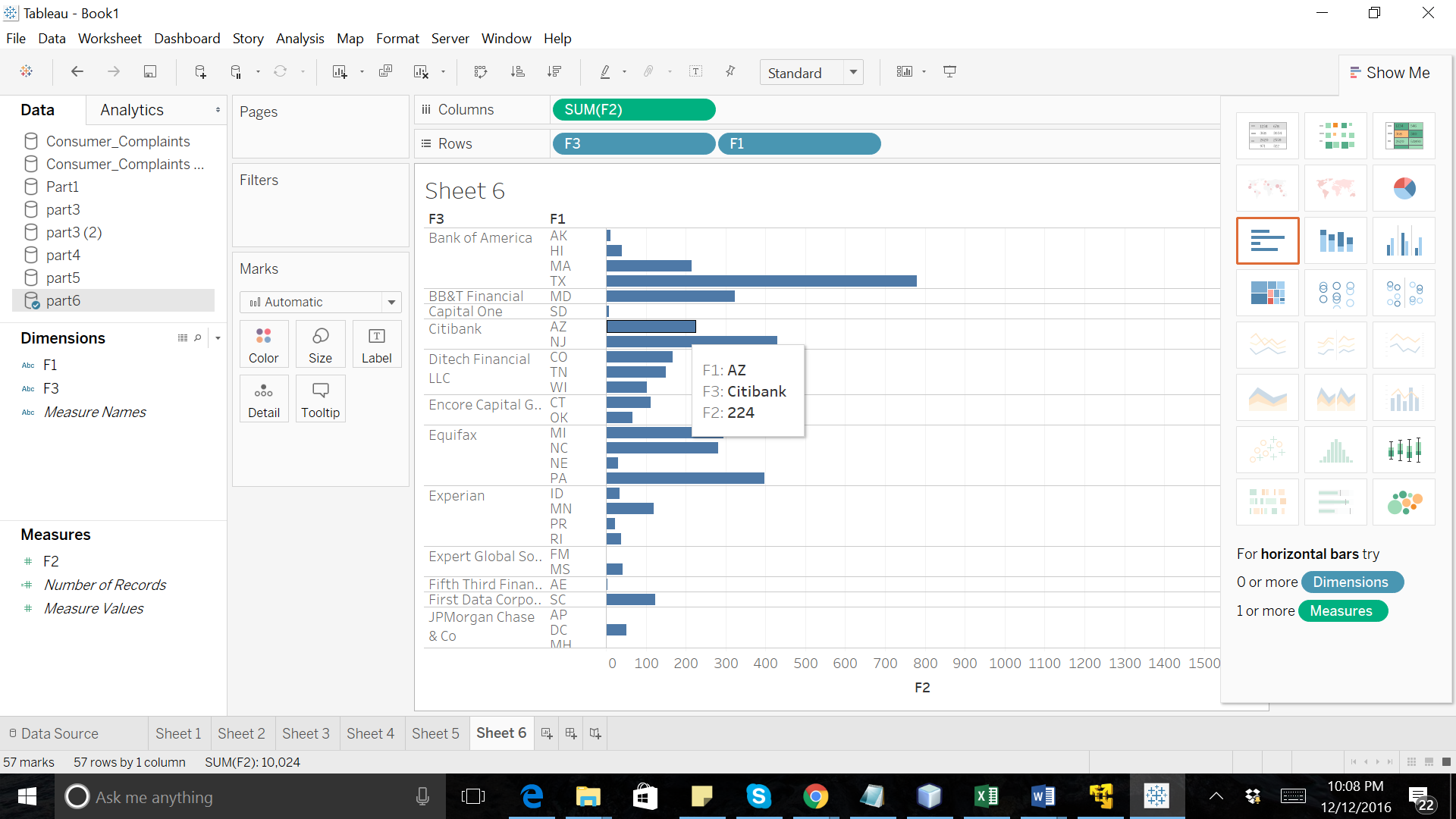
2.Top 10 products with maximum number of complaints(poor performance):



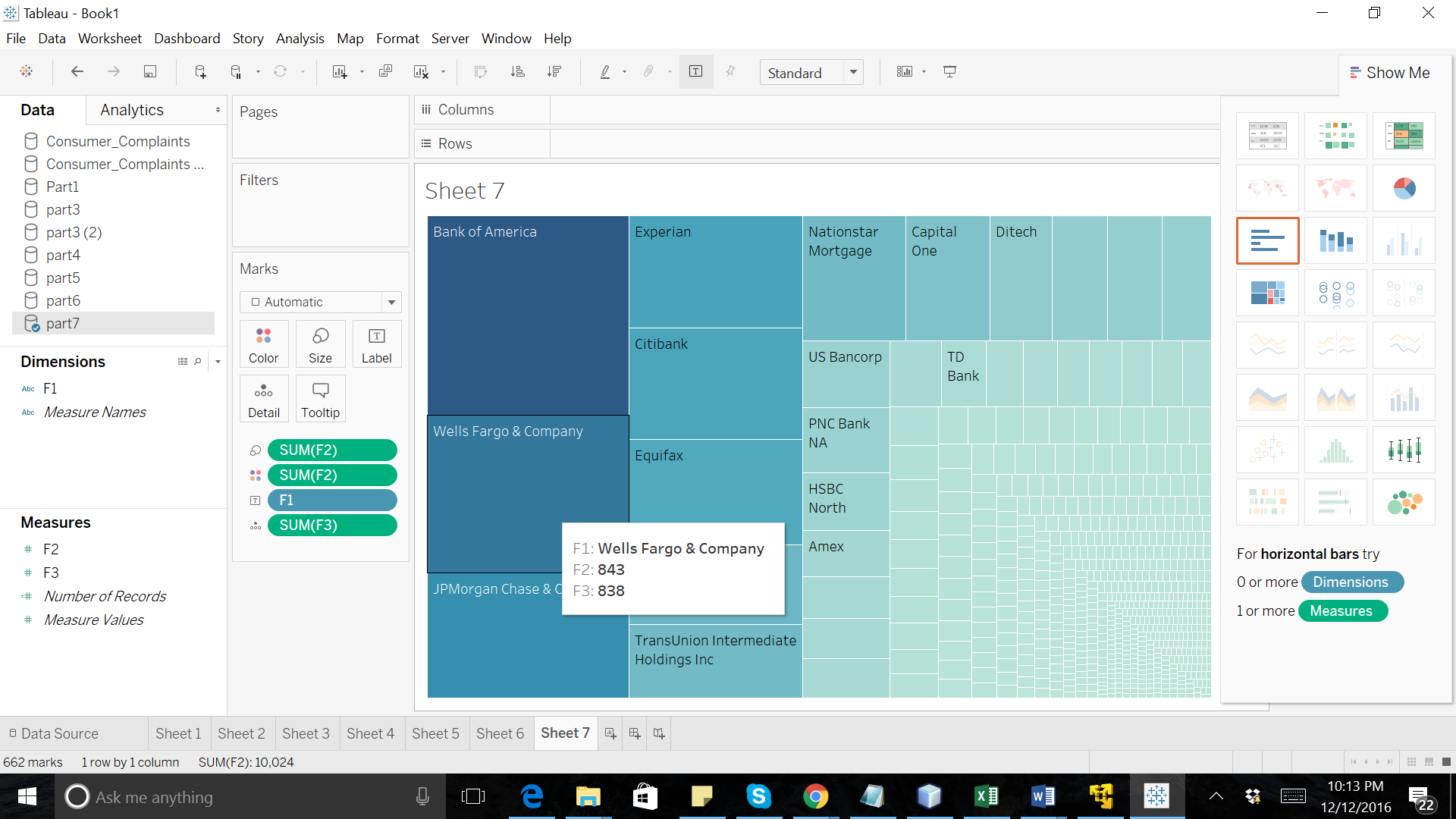
3.Yearly analysis of each company’s each product:



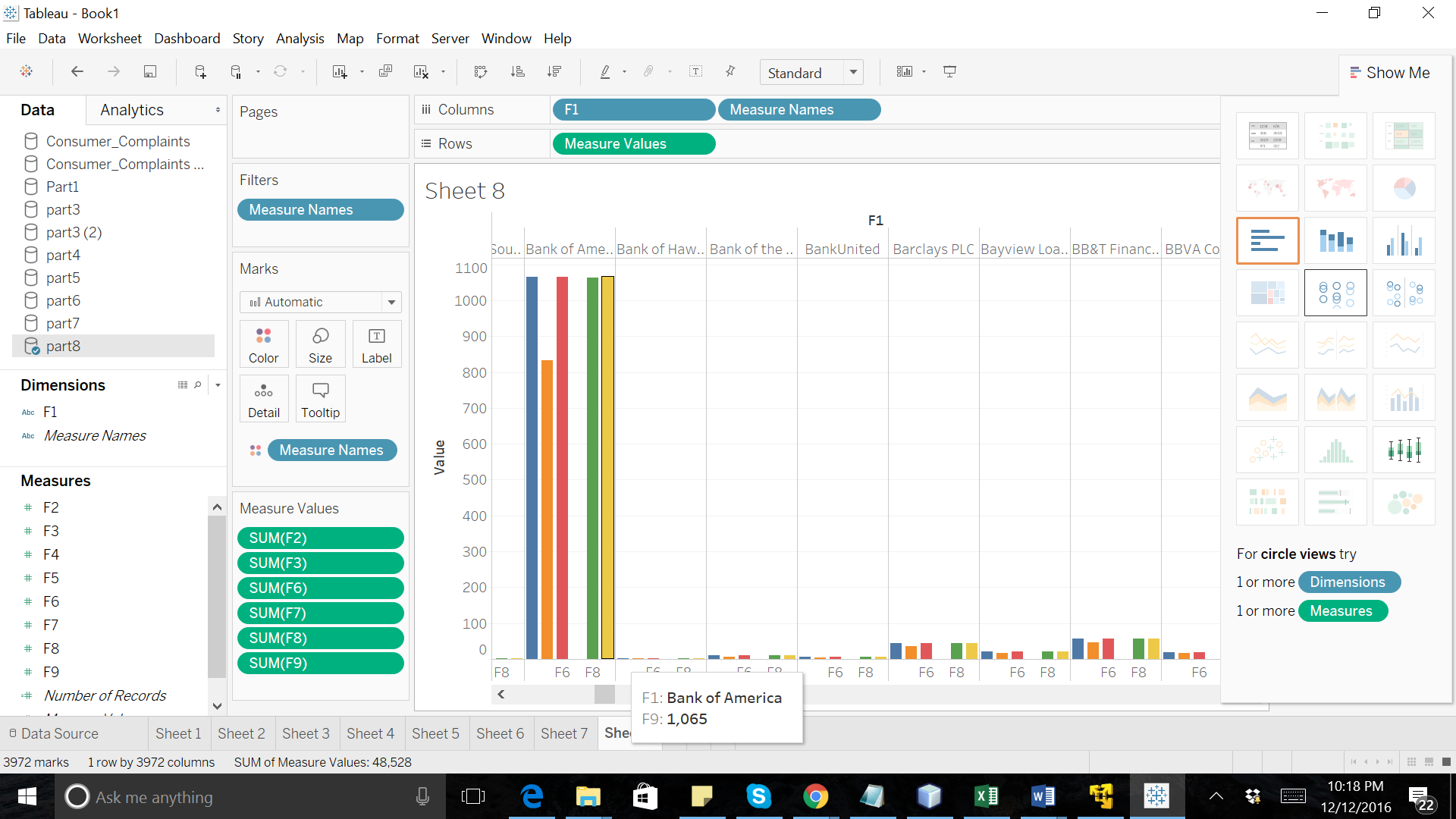
1. State wise company’s analysis:



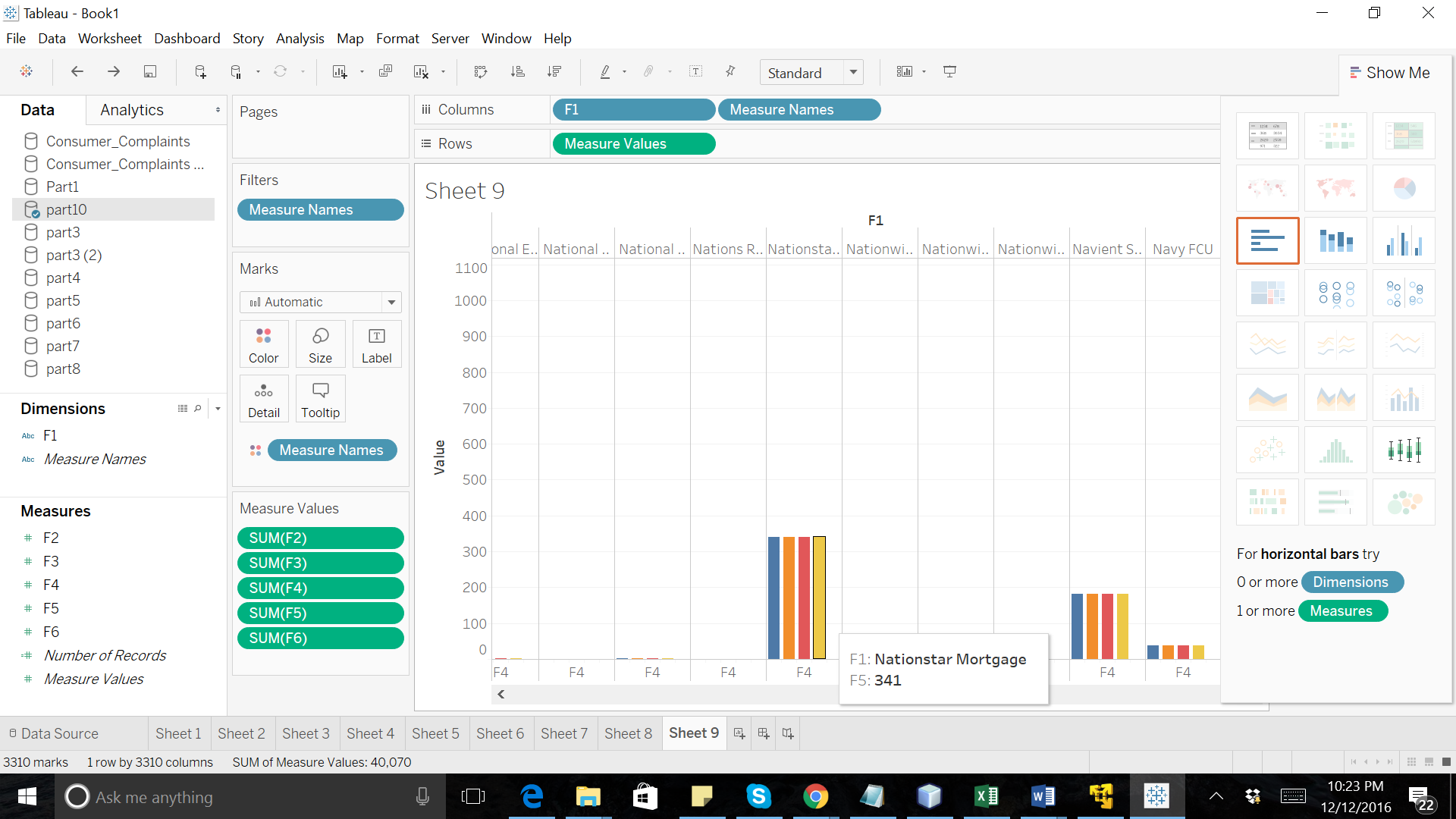
4.Timely response analysis:



5.Submission mode analysis:



6.Complaint- status analysis:



**Appendix:**

**Python Script to upload data to s3:**

**import** boto3  
**import** boto3.s3  
  
**import** os.path  
**import** sys  
  
aws\_access\_key\_id=**'AKIAJD2B6VZWCDJANLDQ'**aws\_secret\_access\_key=**'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'**s3 = boto3.resource(**'s3'**,aws\_access\_key\_id=aws\_access\_key\_id,aws\_secret\_access\_key=aws\_secret\_access\_key)  
**for** bucket **in** s3.buckets.all():  
 **print**(bucket.name)  
  
data = open(**'C:\\bigdata\\consumer10000.csv'**, **'rb'**)  
s3.Bucket(**'trafficdatacollection123'**).put\_object(Key=**'complaints\_dataset.csv'**, Body=data)

**Pattern1:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart1;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author shubhangi

\*/

public class FinalPart1 {

/\*\*

\* @param args the command line arguments

\*/

public static class TokenizerMapper

extends Mapper<Object, Text, Text, IntWritable>{

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

private Text word = new Text();

public void map(Object key, Text value, Context context

) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString(),",");

//while (itr.hasMoreTokens()) {

String first=itr.nextToken();

String sec=itr.nextToken();

System.out.println(sec);

if(first.equalsIgnoreCase("Date received"))

{

}

else{

word.set(sec);

context.write(word, one);

}

}

}

public static class IntSumReducer

extends Reducer<Text,IntWritable,Text,IntWritable> {

private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values,

Context context

) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "part1");

job.setJarByClass(FinalPart1.class);

job.setMapperClass(TokenizerMapper.class);

//job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**Pattern2:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart3;

import java.io.IOException;

import java.nio.ByteBuffer;

import java.util.Iterator;

import java.util.StringTokenizer;

import java.io.IOException;

import java.nio.ByteBuffer;

import java.util.\*;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapred.\*;

import org.apache.hadoop.util.\*;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import static javax.management.Query.value;

/\*\*

\*

\* @author shubhangi

\*/

public class FinalPart3 {

/\*\*

\* @param args the command line arguments

\*/

public static class IpMapper extends MapReduceBase

implements Mapper<LongWritable, Text, Text, FloatWritable>

{

// Reusable IntWritable for the count

private final static FloatWritable one = new FloatWritable(1);

private Text product = new Text();

public void map(LongWritable fileOffset, Text lineContents,

OutputCollector<Text, FloatWritable> output, Reporter reporter)

throws IOException {

// apply the regex to the line of the access log

String line = lineContents.toString();

String se[] = line.split(",");

if (se[0].equals("Date received") ) {

} else {

System.out.println(se[3]);

// System.out.println(se[2]);

product = new Text(se[3]);

output.collect(product, one);

}

}

}

public static class IpReducer extends MapReduceBase implements Reducer<Text, FloatWritable, Text, FloatWritable>

{

public void reduce(Text ip, Iterator<FloatWritable> counts,

OutputCollector<Text, FloatWritable> output, Reporter reporter)

throws IOException {

int totalCount = 0;

// loop over the count and tally it up

while (counts.hasNext())

{

FloatWritable count = counts.next();

totalCount += count.get();

}

output.collect(ip, new FloatWritable(totalCount));

}

}

public static class IpMapper1 extends MapReduceBase

implements Mapper<LongWritable, Text, FloatWritable, Text>

{

public void map(LongWritable fileOffset, Text lineContents,

OutputCollector<FloatWritable, Text> output, Reporter reporter)

throws IOException {

String[] lineData = lineContents.toString().split("\\s+");

String prod="";

prod = lineContents.toString().replaceAll("\\d+.\*", "");

//for(int i=0;i<lineData.length-1;i++)

//{

// prod=prod+lineData[i];

//}

Text ip = new Text(prod);

FloatWritable count = new FloatWritable(Float.parseFloat(lineData[lineData.length-1].trim()));

output.collect(count,ip);

}

}

public static class IpReducer1 extends MapReduceBase implements Reducer<FloatWritable, Text, Text, FloatWritable>

{

private static int count1=0;

public void reduce(FloatWritable count, Iterator<Text> ip, OutputCollector<Text, FloatWritable> output, Reporter reporter) throws IOException

{

while(ip.hasNext())

{

if(count1++ >=10)

break;

output.collect(ip.next(),count);

}

}

}

public static class MyComparator extends WritableComparator {

public MyComparator() {

super(IntWritable.class);

}

@Override

public int compare(byte[] b1, int s1, int l1, byte[] b2, int s2, int l2) {

Integer v1 = ByteBuffer.wrap(b1, s1, l1).getInt();

Integer v2 = ByteBuffer.wrap(b2, s2, l2).getInt();

return (v1.compareTo(v2)) \* -1;

}

}

/\*\*

\* @param args

\*/

public static void main(String[] args) throws Exception

{

JobConf conf = new JobConf(FinalPart3.class);

JobConf conf1 = new JobConf(FinalPart3.class);

conf.setJobName("top10");

conf1.setJobName("top10");

conf.setMapperClass(IpMapper.class);

conf1.setMapperClass(IpMapper1.class);

conf.setMapOutputKeyClass(Text.class);

conf.setMapOutputValueClass(FloatWritable.class);

conf1.setMapOutputKeyClass(IntWritable.class);

conf1.setMapOutputValueClass(Text.class);

conf.setReducerClass(IpReducer.class);

conf1.setReducerClass(IpReducer1.class);

FileInputFormat.setInputPaths(conf, new Path(args[0]));

FileOutputFormat.setOutputPath(conf, new Path("temp"));

JobClient.runJob(conf);

JobConf conf2 = new JobConf(FinalPart3.class);

conf2.setJobName("ip-sort2");

conf2.setMapperClass(IpMapper1.class);

conf2.setMapOutputKeyClass(FloatWritable.class);

conf2.setMapOutputValueClass(Text.class);

conf2.setOutputKeyComparatorClass(MyComparator.class);

conf2.setNumReduceTasks(1);

conf2.setReducerClass(IpReducer1.class);

FileInputFormat.setInputPaths(conf2, new Path("temp"));

FileOutputFormat.setOutputPath(conf2, new Path(args[1]));

JobClient.runJob(conf2);

}

}

**Pattern3:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart4;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableUtils;

/\*\*

\*

\* @author shubhangi

\*/

public class CompositeKeyClass implements Writable,

WritableComparable<CompositeKeyClass> {

private String company;

private String product;

public CompositeKeyClass() {

}

public CompositeKeyClass(String company, String product) {

this.company = company;

this.product = product;

}

@Override

public String toString() {

return (new StringBuilder().append(company).append("\t")

.append(product)).toString();

}

public void readFields(DataInput dataInput) throws IOException {

company = WritableUtils.readString(dataInput);

product = WritableUtils.readString(dataInput);

}

public void write(DataOutput dataOutput) throws IOException {

WritableUtils.writeString(dataOutput, company);

WritableUtils.writeString(dataOutput, product);

}

public int compareTo(CompositeKeyClass objKeyPair) {

// TODO:

/\*

\* Note: This code will work as it stands; but when CompositeKeyWritable

\* is used as key in a map-reduce program, it is de-serialized into an

\* object for comapareTo() method to be invoked;

\*

\* To do: To optimize for speed, implement a raw comparator - will

\* support comparison of serialized representations

\*/

int result = company.compareTo(objKeyPair.company);

if (0 == result) {

result = product.compareTo(objKeyPair.product);

}

return result;

}

public String getCompany() {

return company;

}

public void setCompany(String company) {

this.company = company;

}

public String getProduct() {

return product;

}

public void setProduct(String product) {

this.product = product;

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart4;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import java.util.Date;

import org.apache.hadoop.io.Writable;

/\*\*

\*

\* @author shubhangi

\*/

public class MinTuple implements Writable{

/\*\*

\* @param args the command line arguments

\*/

private Date year = new Date();

private long count=0;

//private final static SimpleDateFormat format = new SimpleDateFormat("yyyy-mm-dd");

public Date getYear() {

return year;

}

public void setYear(Date year) {

this.year = year;

}

public long getCount() {

return count;

}

public void setCount(long count) {

this.count = count;

}

@Override

public void write(DataOutput d) throws IOException {

d.writeLong(year.getTime());

d.writeLong(count);

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public void readFields(DataInput di) throws IOException {

year = new Date(di.readLong());

count = di.readLong();

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public String toString(){

return year + "\t" + count;

//return stockVolume+ " "+stockAdjClose;

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart4;

import java.io.IOException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author shubhangi

\*/

public class FinalPart4 {

/\*\*

\* @param args the command line arguments

\*/

public static class MinMaxMapper extends Mapper<Object, Text, CompositeKeyClass, MinTuple> {

private MinTuple result = new MinTuple();

private final static SimpleDateFormat format = new SimpleDateFormat("yyyy-");

Date strDate = new Date();

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

String line = value.toString();

String se[] = line.split(",");

if (se[0].equals("Date received") ) {

} else {

try {

// System.out.println(se[2]);

strDate = format.parse(se[0]);

System.out.println(se[0]);

System.out.println(se[3]);

result.setYear(strDate);

result.setCount(1);

context.write(new CompositeKeyClass(

se[1].toString(),

se[3].toString()), result);

} catch (ParseException ex) {

Logger.getLogger(FinalPart4.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

}

public static class MinMaxReducer extends Reducer<CompositeKeyClass, MinTuple, CompositeKeyClass, MinTuple> {

private MinTuple result = new MinTuple();

public void reduce(CompositeKeyClass key, Iterable<MinTuple> values, Context context) throws IOException, InterruptedException {

result.setYear(null);

result.setCount(0);

long sum=0;

//result.setStockAdjClose(0);

for (MinTuple value : values) {

// if (value.getStockVolume() < result.getStockVolume()) {

// result.setMin(value.getMin());

// result.setStockVolume(value.getStockVolume());

// //result.setStockAdjClose(value.getStockAdjClose());

// }

// if (value.getStockVolume() > result.getStockVolume()) {

// result.setMax(value.getMax());

// result.setStockVolume(value.getStockVolume());

//

// }

// if(value.getStockAdjClose() > result.getStockAdjClose()){

// result.setStockAdjClose(value.getStockAdjClose());

// }

result.setYear(value.getYear());

sum =sum + value.getCount();

}

result.setCount(sum);

context.write(key, result);

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "StockVolume Min Max date");

job.setJarByClass(FinalPart4.class);

job.setMapperClass(MinMaxMapper.class);

job.setMapOutputKeyClass(CompositeKeyClass.class);

job.setMapOutputValueClass(MinTuple.class);

//job.setCombinerClass(MinMaxReducer.class);

job.setReducerClass(MinMaxReducer.class);

job.setOutputKeyClass(CompositeKeyClass.class);

job.setOutputValueClass(MinTuple.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**Pattern4:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart5;

/\*\*

\*

\* @author shubhangi

\*/

public class Product {

final String name;

final String issue;

Product(final String name, String issue) {

this.name = name;

this.issue = issue;

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart5;

import com.google.common.base.Charsets;

import com.google.common.hash.BloomFilter;

import com.google.common.hash.Funnel;

import com.google.common.hash.Sink;

import java.io.IOException;

import java.util.ArrayList;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

/\*\*

\*

\* @author shubhangi

\*/

public class FinalPart5 extends Configured implements Tool {

public static class BloomFilterMapper extends Mapper<Object, Text, Text, NullWritable> {

Funnel<Product> p = new Funnel<Product>() {

public void funnel(Product product, Sink into) {

// TODO Auto-generated method stub

into.putString(product.name, Charsets.UTF\_8)

.putString(product.issue, Charsets.UTF\_8);

}

};

private BloomFilter<Product> friends = BloomFilter.create(p, 4, 0.1);

@Override

public void setup(Context context) throws IOException, InterruptedException {

Product p1 = new Product( "Bank account or service", "Deposits and withdrawals");

ArrayList<Product> friendsList = new ArrayList<Product>();

friendsList.add(p1);

for (Product pr : friendsList) {

friends.put(pr);

}

}

@Override

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

String values[] = value.toString().split(",");

Product p = new Product(values[3], values[5]);

if (friends.mightContain(p)) {

context.write(value, NullWritable.get());

}

}

}

public static void main(String[] args) throws Exception {

int res = ToolRunner.run(new Configuration(), new FinalPart5(), args);

System.exit(res);

}

public int run(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Bloom Filter");

job.setJarByClass(FinalPart5.class);

job.setMapperClass(BloomFilterMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(NullWritable.class);

job.setNumReduceTasks(0);

FileInputFormat.addInputPath(job, new Path(

args[0]));

FileOutputFormat.setOutputPath(job,

new Path(args[1]));

boolean success = job.waitForCompletion(true);

System.out.println(success);

return success ? 0 : 1;

}

}

**Part5:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart6;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import java.util.Date;

import org.apache.hadoop.io.Writable;

/\*\*

\*

\* @author shubhangi

\*/

public class CompanyStateTuple implements Writable{

/\*\*

\* @param args the command line arguments

\*/

private long count=0;

private String company;

//private final static SimpleDateFormat format = new SimpleDateFormat("yyyy-mm-dd");

public long getCount() {

return count;

}

public void setCount(long count) {

this.count = count;

}

public String getCompany() {

return company;

}

public void setCompany(String company) {

this.company = company;

}

@Override

public void write(DataOutput d) throws IOException {

d.writeLong(count);

d.writeBytes(company);

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public void readFields(DataInput di) throws IOException {

count = di.readLong();

company = di.readLine();

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public String toString(){

return count + "\t" + company;

//return stockVolume+ " "+stockAdjClose;

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart6;

import java.io.IOException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author shubhangi

\*/

public class FinalPart6 {

/\*\*

\* @param args the command line arguments

\*/

public static class MinMaxMapper extends Mapper<Object, Text, Text, CompanyStateTuple> {

private Text state = new Text();

private CompanyStateTuple result = new CompanyStateTuple();

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

String line = value.toString();

String se[] = line.split(",");

if (se[0].equals("Date received") ) {

} else {

System.out.println(se[1]);

System.out.println(se[2]);

state = new Text(se[2]);

result.setCompany(se[1]);

result.setCount(1);

context.write(state, result);

}

}

}

public static class MinMaxReducer extends Reducer<Text, CompanyStateTuple, Text, CompanyStateTuple> {

private CompanyStateTuple result = new CompanyStateTuple();

public void reduce(Text key, Iterable<CompanyStateTuple> values, Context context) throws IOException, InterruptedException {

result.setCompany(null);

result.setCount(0);

long sum=0;

//result.setStockAdjClose(0);

for (CompanyStateTuple value : values) {

// if (value.getStockVolume() < result.getStockVolume()) {

// result.setMin(value.getMin());

// result.setStockVolume(value.getStockVolume());

// //result.setStockAdjClose(value.getStockAdjClose());

// }

// if (value.getStockVolume() > result.getStockVolume()) {

// result.setMax(value.getMax());

// result.setStockVolume(value.getStockVolume());

//

// }

// if(value.getStockAdjClose() > result.getStockAdjClose()){

// result.setStockAdjClose(value.getStockAdjClose());

// }

result.setCompany(value.getCompany());

sum =sum + value.getCount();

}

result.setCount(sum);

context.write(key, result);

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "StockVolume Min Max date");

job.setJarByClass(FinalPart6.class);

job.setMapperClass(MinMaxMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(CompanyStateTuple.class);

//job.setCombinerClass(MinMaxReducer.class);

job.setReducerClass(MinMaxReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(CompanyStateTuple.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**Part 6:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart7;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

/\*\*

\*

\* @author shubhangi

\*/

public class ComplaintResponseTuple implements Writable{

/\*\*

\* @param args the command line arguments

\*/

private long count=0;

private long timelyCount=0;

//private final static SimpleDateFormat format = new SimpleDateFormat("yyyy-mm-dd");

public long getCount() {

return count;

}

public long getTimelyCount() {

return timelyCount;

}

public void setTimelyCount(long timelyCount) {

this.timelyCount = timelyCount;

}

public void setCount(long count) {

this.count = count;

}

@Override

public void write(DataOutput d) throws IOException {

d.writeLong(count);

d.writeLong(timelyCount);

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public void readFields(DataInput di) throws IOException {

count = di.readLong();

timelyCount=di.readLong();

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public String toString(){

return count + "\t" + timelyCount ;

//return stockVolume+ " "+stockAdjClose;

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart7;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author shubhangi

\*/

public class FinalPart7{

public static class MinMaxMapper extends Mapper<Object, Text, Text, ComplaintResponseTuple> {

private Text company = new Text();

private ComplaintResponseTuple result = new ComplaintResponseTuple();

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

String line = value.toString();

String se[] = line.split(",");

if (se[0].equals("Date received") ) {

} else {

System.out.println(se[1]);

System.out.println(se[7]);

company.set(se[1]);

if(se[7].equalsIgnoreCase("Yes"))

{

result.setTimelyCount(1);

}

else

{

result.setTimelyCount(0);

}

result.setCount(1);

context.write(company, result);

}

}

}

public static class MinMaxReducer extends Reducer<Text, ComplaintResponseTuple, Text, ComplaintResponseTuple> {

private ComplaintResponseTuple result = new ComplaintResponseTuple();

public void reduce(Text key, Iterable<ComplaintResponseTuple> values, Context context) throws IOException, InterruptedException {

result.setTimelyCount(0);

result.setCount(0);

long sum=0;

long timelySum=0;

//result.setStockAdjClose(0);

for (ComplaintResponseTuple value : values) {

// if (value.getStockVolume() < result.getStockVolume()) {

// result.setMin(value.getMin());

// result.setStockVolume(value.getStockVolume());

// //result.setStockAdjClose(value.getStockAdjClose());

// }

// if (value.getStockVolume() > result.getStockVolume()) {

// result.setMax(value.getMax());

// result.setStockVolume(value.getStockVolume());

//

// }

// if(value.getStockAdjClose() > result.getStockAdjClose()){

// result.setStockAdjClose(value.getStockAdjClose());

// }

sum =sum + value.getCount();

timelySum=timelySum + value.getTimelyCount();

}

result.setCount(sum);

result.setTimelyCount(timelySum);

context.write(key, result);

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "StockVolume Min Max date");

job.setJarByClass(FinalPart7.class);

job.setMapperClass(MinMaxMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(ComplaintResponseTuple.class);

//job.setCombinerClass(MinMaxReducer.class);

job.setReducerClass(MinMaxReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(ComplaintResponseTuple.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**Part7:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart8;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

/\*\*

\*

\* @author shubhangi

\*/

public class ModeSubmissionTuple implements Writable{

/\*\*

\* @param args the command line arguments

\*/

private long count=0;

private long emailCount=0;

private long faxCount=0;

private long phoneCount=0;

private long postalCount=0;

private long mailCount=0;

private long referralCount=0;

private long web=0;

//private final static SimpleDateFormat format = new SimpleDateFormat("yyyy-mm-dd");

public long getCount() {

return count;

}

public long getEmailCount() {

return emailCount;

}

public void setEmailCount(long emailCount) {

this.emailCount = emailCount;

}

public long getFaxCount() {

return faxCount;

}

public void setFaxCount(long faxCount) {

this.faxCount = faxCount;

}

public long getPhoneCount() {

return phoneCount;

}

public void setPhoneCount(long phoneCount) {

this.phoneCount = phoneCount;

}

public long getPostalCount() {

return postalCount;

}

public void setPostalCount(long postalCount) {

this.postalCount = postalCount;

}

public long getMailCount() {

return mailCount;

}

public void setMailCount(long mailCount) {

this.mailCount = mailCount;

}

public long getReferralCount() {

return referralCount;

}

public void setReferralCount(long referralCount) {

this.referralCount = referralCount;

}

public long getWeb() {

return web;

}

public void setWeb(long web) {

this.web = web;

}

public void setCount(long count) {

this.count = count;

}

@Override

public void write(DataOutput d) throws IOException {

d.writeLong(count);

d.writeLong(emailCount);

d.writeLong(faxCount);

d.writeLong(web);

d.writeLong(mailCount);

d.writeLong(phoneCount);

d.writeLong(postalCount);

d.writeLong(referralCount);

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public void readFields(DataInput di) throws IOException {

count = di.readLong();

emailCount=di.readLong();

faxCount=di.readLong();

mailCount=di.readLong();

referralCount=di.readLong();

postalCount=di.readLong();

phoneCount=di.readLong();

web=di.readLong();

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public String toString(){

return count + "\t" + emailCount + "\t" + mailCount + "\t" + phoneCount + "\t" + postalCount + "\t" + referralCount + "\t" + web + "\t" + faxCount ;

//return stockVolume+ " "+stockAdjClose;

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart8;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author shubhangi

\*/

public class FinalPart8{

public static class MinMaxMapper extends Mapper<Object, Text, Text, ModeSubmissionTuple> {

private Text company = new Text();

private ModeSubmissionTuple result = new ModeSubmissionTuple();

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

String line = value.toString();

String se[] = line.split(",");

if (se[0].equals("Date received") ) {

} else {

System.out.println(se[1]);

System.out.println(se[8]);

company.set(se[1]);

if(se[8].equalsIgnoreCase("Phone"))

{

result.setPhoneCount(1);

}

else if (se[8].equalsIgnoreCase("Referral"))

{

result.setReferralCount(1);

}

else if (se[8].equalsIgnoreCase("Web"))

{

result.setWeb(1);

}

else if (se[8].equalsIgnoreCase("Postal"))

{

result.setPostalCount(1);

}

else if (se[8].equalsIgnoreCase("Mail"))

{

result.setMailCount(1);

}

else if (se[8].equalsIgnoreCase("Fax"))

{

result.setFaxCount(1);

}

else if (se[8].equalsIgnoreCase("Email"))

{

result.setEmailCount(1);

}

result.setCount(1);

context.write(company, result);

}

}

}

public static class MinMaxReducer extends Reducer<Text, ModeSubmissionTuple, Text, ModeSubmissionTuple> {

private ModeSubmissionTuple result = new ModeSubmissionTuple();

public void reduce(Text key, Iterable<ModeSubmissionTuple> values, Context context) throws IOException, InterruptedException {

result.setPhoneCount(0);

result.setWeb(0);

result.setPostalCount(0);

result.setReferralCount(0);

result.setFaxCount(0);

result.setMailCount(0);

result.setEmailCount(0);

result.setCount(0);

long sum=0;

long phoneSum=0;

long faxSum=0;

long emailSum=0;

long mailSum=0;

long referralSum=0;

long postalSum=0;

long webSum=0;

//result.setStockAdjClose(0);

for (ModeSubmissionTuple value : values) {

// if (value.getStockVolume() < result.getStockVolume()) {

// result.setMin(value.getMin());

// result.setStockVolume(value.getStockVolume());

// //result.setStockAdjClose(value.getStockAdjClose());

// }

// if (value.getStockVolume() > result.getStockVolume()) {

// result.setMax(value.getMax());

// result.setStockVolume(value.getStockVolume());

//

// }

// if(value.getStockAdjClose() > result.getStockAdjClose()){

// result.setStockAdjClose(value.getStockAdjClose());

// }

sum =sum + value.getCount();

phoneSum=phoneSum + value.getPhoneCount();

postalSum=postalSum + value.getPostalCount();

faxSum=faxSum + value.getFaxCount();

emailSum=emailSum + value.getEmailCount();

mailSum=mailSum + value.getMailCount();

referralSum=referralSum + value.getReferralCount();

webSum=webSum + value.getWeb();

}

result.setCount(sum);

result.setPhoneCount(phoneSum);

result.setPostalCount(postalSum);

result.setWeb(webSum);

result.setReferralCount(referralSum);

result.setEmailCount(emailSum);

result.setMailCount(mailSum);

result.setFaxCount(faxSum);

context.write(key, result);

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "StockVolume Min Max date");

job.setJarByClass(FinalPart8.class);

job.setMapperClass(MinMaxMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(ModeSubmissionTuple.class);

//job.setCombinerClass(MinMaxReducer.class);

job.setReducerClass(MinMaxReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(ModeSubmissionTuple.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**Part8:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart9;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Partitioner;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author shubhangi

\*/

public class FinalPart9 {

/\*\*

\* @param args the command line arguments

\*/

public static class StateMapper extends Mapper<Object,Text,Text, Text>{

private Text outKey=new Text();

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

String line = value.toString();

String se[] = line.split(",");

if (se[0].equals("Date received") ) {

} else {

System.out.println(se[1]);

System.out.println(se[7]);

outKey.set(se[2]);

context.write(outKey, value);

}

}

}

public static class StatePartioner extends Partitioner<Text, Text>

{

@Override

public int getPartition(Text key, Text value, int numReduceTasks) {

String word = key.toString();

// char letter = word.toLowerCase().charAt(0);

// return (int) letter - 97;

//

if(word.contains("VA")){

return 1 ;

}else if(word.contains("CA")){

return 2;

}else if(word.contains("NY")){

return 3 ;

}else if(word.contains("GA")){

return 4 ;

}else if(word.contains("CT")){

return 5 ;

}else if(word.contains("TX")){

return 6 ;

}

return 0;

}

}

public static class StateReducer extends Reducer<Text, Text, Text, NullWritable>

{

public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

for(Text t:values)

{

context.write(t, NullWritable.get());

}

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "state partition");

job.setJarByClass(FinalPart9.class);

job.setMapperClass(StateMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setPartitionerClass(StatePartioner.class);

job.setNumReduceTasks(7);

//job.setCombinerClass(MinMaxReducer.class);

job.setReducerClass(StateReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**Part9:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart10;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

/\*\*

\*

\* @author shubhangi

\*/

public class StatusTupleClass implements Writable{

/\*\*

\* @param args the command line arguments

\*/

private long count=0;

private long closedCount=0;

private long monCount=0;

private long nonMonCount=0;

private long untimelyCount=0;

//private final static SimpleDateFormat format = new SimpleDateFormat("yyyy-mm-dd");

public long getCount() {

return count;

}

public long getClosedCount() {

return closedCount;

}

public void setClosedCount(long closedCount) {

this.closedCount = closedCount;

}

public long getMonCount() {

return monCount;

}

public void setMonCount(long monCount) {

this.monCount = monCount;

}

public long getNonMonCount() {

return nonMonCount;

}

public void setNonMonCount(long nonMonCount) {

this.nonMonCount = nonMonCount;

}

public long getUntimelyCount() {

return untimelyCount;

}

public void setUntimelyCount(long untimelyCount) {

this.untimelyCount = untimelyCount;

}

public void setCount(long count) {

this.count = count;

}

@Override

public void write(DataOutput d) throws IOException {

d.writeLong(count);

d.writeLong(closedCount);

d.writeLong(monCount);

d.writeLong(nonMonCount);

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public void readFields(DataInput di) throws IOException {

count = di.readLong();

closedCount=di.readLong();

monCount=di.readLong();

nonMonCount=di.readLong();

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public String toString(){

return count + "\t" + closedCount +"\t" + monCount +"\t" + nonMonCount

+"\t" + untimelyCount;

//return stockVolume+ " "+stockAdjClose;

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart10;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author shubhangi

\*/

public class FinalPart10 {

/\*\*

\* @param args the command line arguments

\*/

public static class MinMaxMapper extends Mapper<Object, Text, Text, StatusTupleClass> {

private Text company = new Text();

private StatusTupleClass result = new StatusTupleClass();

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

String line = value.toString();

String se[] = line.split(",");

if (se[0].equals("Date received") ) {

} else {

System.out.println(se[1]);

System.out.println(se[7]);

company.set(se[1]);

if(se[15].equalsIgnoreCase("Closed") || se[15].equalsIgnoreCase("Closed with explanation"))

{

result.setClosedCount(1);

}

else if(se[15].equalsIgnoreCase("Closed with monetary relief"))

{

result.setMonCount(1);

}

else if(se[15].equalsIgnoreCase("Closed with non-monetary relief"))

{

result.setNonMonCount(1);

}

else if(se[15].equalsIgnoreCase("Untimely response"))

{

result.setUntimelyCount(1);

}

result.setCount(1);

context.write(company, result);

}

}

}

public static class MinMaxReducer extends Reducer<Text, StatusTupleClass, Text, StatusTupleClass> {

private StatusTupleClass result = new StatusTupleClass();

public void reduce(Text key, Iterable<StatusTupleClass> values, Context context) throws IOException, InterruptedException {

result.setUntimelyCount(0);

result.setClosedCount(0);

result.setMonCount(0);

result.setNonMonCount(0);

result.setCount(0);

long sum=0;

long closed=0;

long mon=0;

long nonMon=0;

long untimely=0;

//result.setStockAdjClose(0);

for (StatusTupleClass value : values) {

// if (value.getStockVolume() < result.getStockVolume()) {

// result.setMin(value.getMin());

// result.setStockVolume(value.getStockVolume());

// //result.setStockAdjClose(value.getStockAdjClose());

// }

// if (value.getStockVolume() > result.getStockVolume()) {

// result.setMax(value.getMax());

// result.setStockVolume(value.getStockVolume());

//

// }

// if(value.getStockAdjClose() > result.getStockAdjClose()){

// result.setStockAdjClose(value.getStockAdjClose());

// }

sum =sum + value.getCount();

closed=closed + value.getClosedCount();

mon=mon + value.getMonCount();

nonMon= nonMon + value.getNonMonCount();

untimely=untimely+value.getUntimelyCount();

}

result.setCount(sum);

result.setClosedCount(closed);

result.setMonCount(mon);

result.setNonMonCount(nonMon);

result.setUntimelyCount(untimely);

context.write(key, result);

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "StockVolume Min Max date");

job.setJarByClass(FinalPart10.class);

job.setMapperClass(MinMaxMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(StatusTupleClass.class);

//job.setCombinerClass(MinMaxReducer.class);

job.setReducerClass(MinMaxReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(StatusTupleClass.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**Part10:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart11;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import java.util.Date;

import org.apache.hadoop.io.Writable;

/\*\*

\*

\* @author shubhangi

\*/

public class Tuple implements Writable{

/\*\*

\* @param args the command line arguments

\*/

private long count=0;

private long timelyCount=0;

//private final static SimpleDateFormat format = new SimpleDateFormat("yyyy-mm-dd");

public long getCount() {

return count;

}

public long getTimelyCount() {

return timelyCount;

}

public void setTimelyCount(long timelyCount) {

this.timelyCount = timelyCount;

}

public void setCount(long count) {

this.count = count;

}

@Override

public void write(DataOutput d) throws IOException {

d.writeLong(count);

d.writeLong(timelyCount);

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public void readFields(DataInput di) throws IOException {

count = di.readLong();

timelyCount=di.readLong();

//throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public String toString(){

return count + "\t" + timelyCount ;

//return stockVolume+ " "+stockAdjClose;

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart11;

import java.io.IOException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Map;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.SortedMapWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Partitioner;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author shubhangi

\*/

public class FinalPart11 {

public static class MinMaxMapper extends Mapper<Object, Text, Text, Tuple> {

private Text company = new Text();

private Tuple result = new Tuple();

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

String line = value.toString();

String se[] = line.split(",");

if (se[0].equals("Date received") ) {

} else {

System.out.println(se[1]);

System.out.println(se[7]);

company.set(se[1]);

if(se[7].equalsIgnoreCase("Yes"))

{

result.setTimelyCount(1);

}

else

{

result.setTimelyCount(0);

}

result.setCount(1);

context.write(company, result);

}

}

}

public static class MinMaxReducer extends Reducer<Text, Tuple, Text, Tuple> {

private Tuple result = new Tuple();

public void reduce(Text key, Iterable<Tuple> values, Context context) throws IOException, InterruptedException {

result.setTimelyCount(0);

result.setCount(0);

long sum=0;

long timelySum=0;

//result.setStockAdjClose(0);

for (Tuple value : values) {

sum =sum + value.getCount();

timelySum = timelySum + value.getTimelyCount();

}

result.setCount(sum);

result.setTimelyCount(timelySum);

context.write(key, result);

}

}

public static class SimplePartitioner extends Partitioner<Text, Tuple>{

@Override

public int getPartition(Text key, Tuple value, int numReducerTask) {

String s= key.toString();

return (Character.toString(s.charAt(0)).hashCode() % numReducerTask);

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "StockVolume Min Max date");

job.setJarByClass(FinalPart11.class);

job.setMapperClass(MinMaxMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Tuple.class);

job.setCombinerClass(MinMaxReducer.class);

job.setPartitionerClass(SimplePartitioner.class);

job.setNumReduceTasks(10);

job.setReducerClass(MinMaxReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Tuple.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**Part11:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package finalpart13;

/\*\*

\*

\* @author shubhangi

\*/

import java.io.IOException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class FinalPart13 {

/\*\*

\* @param args the command line arguments

\*/

public static class WordcountMapper extends

Mapper<Object, Text, Text, Text> {

private Text issue = new Text();

private Text id = new Text();

private boolean caseSensitive = false;

@Override

public void map(Object key, Text value, Context context)

throws IOException, InterruptedException {

//String filenameStr = ((FileSplit) context.getInputSplit()).getPath().getName();

//filename = new Text(filenameStr);

String line = value.toString();

if (!caseSensitive) {

line = line.toLowerCase();

}

//String line = value.toString();

String se[] = line.split(",");

if (se[0].equals("Date received") ) {

} else {

//System.out.println(se[1]);

// System.out.println(se[2]);

issue = new Text(se[5]);

id = new Text(se[17]);

context.write(issue, id);

}

}

@Override

protected void setup(Context context) throws IOException, InterruptedException {

Configuration conf = context.getConfiguration();

this.caseSensitive = conf.getBoolean("wordcount.case.sensitive",false);

}

}

public static class WordcountReducer extends Reducer<Text, Text, Text, Text> {

@Override

public void reduce(final Text key, final Iterable<Text> values,

final Context context) throws IOException, InterruptedException {

StringBuilder stringBuilder = new StringBuilder();

for (Text value : values) {

stringBuilder.append(value.toString());

if (values.iterator().hasNext()) {

stringBuilder.append(" -> ");

}

}

context.write(key, new Text(stringBuilder.toString()));

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "StockVolume Min Max date");

job.setJarByClass(FinalPart13.class);

job.setMapperClass(WordcountMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

//job.setCombinerClass(MinMaxReducer.class);

job.setReducerClass(WordcountReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}