Practical No 1

Aim: HDFS: List of Commands (mkdir, touchz, copy from local/put, copy to local/get, move from local, cp, rmr, du, dus, stat).

A) -mkdir: To create a directory.

```
hadoop fs -mkdir /user/cloudera/thursday
```

```
[cloudera@quickstart ~]$ hadoop fs -ls
Found 2 items
drwxr-xr-x   - cloudera cloudera          0 2021-09-20 08:49 sunio
drwxr-xr-x   - cloudera cloudera          0 2021-09-20 21:51 thursday
```

B) -touchz: It creates an empty file.

```
hadoop fs -touchz /user/cloudera/thursday/sample.txt
```

```
[cloudera@quickstart ~]$ hadoop fs -touchz /user/cloudera/thursday/sample.txt
[cloudera@quickstart ~]$ hadoop fs -ls  /user/cloudera/thursday
Found 1 items
-rw-r--r--   1 cloudera cloudera          0 2021-09-20 22:11 /user/cloudera/thur
sday/sample.txt
```

C) CopyFromLocal: To copy files/folders from local file system to hdfs store.

```
[cloudera@quickstart ~]$ ls
cloudera-manager  enterprise-deployment.json  parcels      Templates
cm_api.py         express-deployment.json     Pictures     Videos
Desktop           kerberos                    Public       WordCount.jar
Documents         lib                         putfile.txt  word.txt
Downloads         local.txt                   q            workspace
eclipse           Music                       sunil
```

```
hadoop fs -copyFromLocal  local.txt  /user/cloudera/thursday
```

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/cloudera/thursday
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2021-09-21 23:19 /user/cloudera/thur
sday/local.txt
-rw-r--r--   1 cloudera cloudera          0 2021-09-20 22:11 /user/cloudera/thur
sday/sample.txt
```

D) put

```
[cloudera@quickstart ~]$ ls
cloudera-manager  enterprise-deployment.json  parcels      Templates
cm_api.py         express-deployment.json     Pictures     Videos
Desktop           kerberos                    Public       WordCount.jar
Documents         lib                         putfile.txt  word.txt
Downloads         local.txt                   q            workspace
eclipse           Music                       sunil
```

```
hadoop fs -put /home/cloudera/putfile.txt  /user/cloudera/thursday
```

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/cloudera/sunio
Found 1 items
-rw-r--r--   1 cloudera cloudera         24 2021-09-23 02:25 /user/cloudera/suni
o/putfile.txt
```

E) CopyToLocal: To Copy files/folders from hdfs store to local file system.

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/cloudera/sunio
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2021-09-23 04:38 /user/cloudera/suni
o/cplocal.txt
-rw-r--r--   1 cloudera cloudera         24 2021-09-23 02:25 /user/cloudera/suni
o/putfile.txt
```

```
hadoop fs -copyToLocal /user/cloudera/sunio/cplocal.txt /home/cloudera/sunil
```

```
[cloudera@quickstart ~]$ hadoop fs -copyToLocal /user/cloudera/sunio/cplocal.txt
 /home/cloudera/sunil
[cloudera@quickstart ~]$ ls /home/cloudera/sunil
cplocal.txt
```

F) get

```
[cloudera@quickstart ~]$ hadoop fs -touchz /user/cloudera/sunio/getfile.txt
[cloudera@quickstart ~]$ hadoop fs -ls /user/cloudera/sunio
Found 3 items
-rw-r--r--   1 cloudera cloudera          0 2021-09-23 04:38 /user/cloudera/suni
o/cplocal.txt
-rw-r--r--   1 cloudera cloudera          0 2021-09-23 04:44 /user/cloudera/suni
o/getfile.txt
-rw-r--r--   1 cloudera cloudera         24 2021-09-23 02:25 /user/cloudera/suni
o/putfile.txt
```

```
hadoop fs -get /user/cloudera/sunio/getfile.txt /home/cloudera/sunil
```

```
[cloudera@quickstart ~]$ hadoop fs -get /user/cloudera/sunio/getfile.txt /home/c
loudera/sunil
[cloudera@quickstart ~]$ ls /home/cloudera/sunil
cplocal.txt  getfile.txt
```

G) moveFromLocal: This command will move file from local to hdfs.

```
[cloudera@quickstart ~]$ ls /home/cloudera/sunil
cplocal.txt  getfile.txt  mvfromlocal.txt
```

```
hadoop fs -moveFromLocal /home/cloudera/sunil/mvfromlocal.txt /user/cloudera/sunio
```

```
[cloudera@quickstart ~]$ hadoop fs -moveFromLocal /home/cloudera/sunil/mvfromloc
al.txt /user/cloudera/sunio
[cloudera@quickstart ~]$ hadoop fs -ls /user/cloudera/sunio
Found 4 items
-rw-r--r--   1 cloudera cloudera          0 2021-09-23 04:38 /user/cloudera/suni
o/cplocal.txt
-rw-r--r--   1 cloudera cloudera          0 2021-09-23 04:44 /user/cloudera/suni
o/getfile.txt
-rw-r--r--   1 cloudera cloudera          0 2021-09-23 05:00 /user/cloudera/suni
o/mvfromlocal.txt
-rw-r--r--   1 cloudera cloudera         24 2021-09-23 02:25 /user/cloudera/suni

[cloudera@quickstart ~]$ ls /home/cloudera/sunil
cplocal.txt  getfile.txt
```

H) cp: This command is used to copy files within hdfs.

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/cloudera/thursday
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2021-09-21 23:19 /user/cloudera/thur
sday/local.txt
-rw-r--r--   1 cloudera cloudera          0 2021-09-20 22:11 /user/cloudera/thur
sday/sample.txt
[cloudera@quickstart ~]$ hadoop fs -ls /user/cloudera/sunio
Found 4 items
-rw-r--r--   1 cloudera cloudera          0 2021-09-23 04:38 /user/cloudera/suni
o/cplocal.txt
-rw-r--r--   1 cloudera cloudera          0 2021-09-23 04:44 /user/cloudera/suni
o/getfile.txt
-rw-r--r--   1 cloudera cloudera          0 2021-09-23 05:00 /user/cloudera/suni
o/mvfromlocal.txt
-rw-r--r--   1 cloudera cloudera         24 2021-09-23 02:25 /user/cloudera/suni
o/putfile.txt
```

```
hadoop fs -cp /user/cloudera/sunio/cplocal.txt /user/cloudera/sunio
```

```
[cloudera@quickstart ~]$ hadoop fs -cp /user/cloudera/sunio/cplocal.txt /user/cl
oudera/thursday
[cloudera@quickstart ~]$ hadoop fs -ls /user/cloudera/thursday
Found 3 items
-rw-r--r--   1 cloudera cloudera          0 2021-09-23 06:50 /user/cloudera/thur
sday/cplocal.txt
-rw-r--r--   1 cloudera cloudera          0 2021-09-21 23:19 /user/cloudera/thur
sday/local.txt
-rw-r--r--   1 cloudera cloudera          0 2021-09-20 22:11 /user/cloudera/thur
sday/sample.txt
```

I) rmr: This Command delete a file from HDFS recursively.

```
hadoop fs -rmr /user/cloudera/thursday
```

```
[cloudera@quickstart ~]$ hadoop fs -rmr /user/cloudera/thursday
rmr: DEPRECATED: Please use 'rm -r' instead.
Deleted /user/cloudera/thursday
```

J) du: It will give the size of each file in directory.

```
hadoop fs -du /user/cloudera/sunio
```

```
[cloudera@quickstart ~]$ hadoop fs -du /user/cloudera/sunio
0   0   /user/cloudera/sunio/cplocal.txt
0   0   /user/cloudera/sunio/getfile.txt
0   0   /user/cloudera/sunio/mvfromlocal.txt
24  24  /user/cloudera/sunio/putfile.txt
```

K) dus: This command will give the total size of the directory/file.

```
hadoop fs -dus /user/cloudera/sunio
```

```
[cloudera@quickstart ~]$ hadoop fs -dus /user/cloudera/sunio
dus: DEPRECATED: Please use 'du -s' instead.
24  24  /user/cloudera/sunio
```

L) stat: It will give the last modified time of directory or path.

```
hadoop fs -stat /user/cloudera/sunio
```

```
[cloudera@quickstart ~]$ hadoop fs -stat /user/cloudera/sunio
2021-09-23 12:00:50
```

Practical No 2

Aim: Write a program in Map Reduce for Word Count operation.

Code:

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


public class WordCount
{
	public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable>
	{
		private final static IntWritable one=new IntWritable(1);
		private Text word=new Text();

		public void map(Object key,Text Value,Context context)throws IOException,InterruptedException
		{
			StringTokenizer itr=new StringTokenizer(Value.toString());
			while(itr.hasMoreTokens())
			{
				word.set(itr.nextToken());
```

```
                    context.write(word, one);
                }
            }
        }
        public static class IntSumReducer extends Reducer<Text,
IntWritable,Text,IntWritable>
        {
            private IntWritable result=new IntWritable();

            public void reduce(Text key,Iterable<IntWritable>values,Context
context)throws IOException,InterruptedException
            {
                int sum=0;
                for(IntWritable val:values)
                {
                    sum=sum+val.get();
                }
                result.set(sum);
                context.write(key, result);
            }
        }
        public static void main(String[] args)throws Exception
        {
            Configuration conf=new Configuration();
            Job job=Job.getInstance(conf,"word count");
            job.setJarByClass(WordCount.class);
            job.setMapperClass(TokenizerMapper.class);
            job.setReducerClass(IntSumReducer.class);
            job.setCombinerClass(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);
    }
}
```

Input File:

```
[cloudera@quickstart ~]$ cat word.txt
This is word count file
This is Good file
```

Output:

hadoop jar WordCounr.jar WordCount /idata/word.txt/output1.txt

```
[cloudera@quickstart ~]$ hadoop jar WordCount.jar WordCount /idata/word.txt /out
put1.txt
21/09/23 08:16:42 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
21/09/23 08:16:47 WARN mapreduce.JobResourceUploader: Hadoop command-line option
 parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
21/09/23 08:16:48 INFO input.FileInputFormat: Total input paths to process : 1
21/09/23 08:16:49 INFO mapreduce.JobSubmitter: number of splits:1
21/09/23 08:16:51 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_16
32404512059_0001
21/09/23 08:16:52 INFO impl.YarnClientImpl: Submitted application application_16
32404512059_0001
21/09/23 08:16:52 INFO mapreduce.Job: The url to track the job: http://quickstar
t.cloudera:8088/proxy/application_1632404512059_0001/
21/09/23 08:16:52 INFO mapreduce.Job: Running job: job_1632404512059_0001
21/09/23 08:17:33 INFO mapreduce.Job: Job job_1632404512059_0001 running in uber
 mode : false
21/09/23 08:17:33 INFO mapreduce.Job:  map 0% reduce 0%
21/09/23 08:17:58 INFO mapreduce.Job:  map 100% reduce 0%
21/09/23 08:18:14 INFO mapreduce.Job:  map 100% reduce 100%
21/09/23 08:18:16 INFO mapreduce.Job: Job job_1632404512059_0001 completed succe
ssfully
```

hadoop fs -cat /output1/part-r-00000

```
[cloudera@quickstart ~]$ hadoop fs -cat /output1/part-r-00000
Good    1
This    2
count   1
file    2
is      2
word    1
```

Aim: Write a program in Map Reduce for Union operation.

Code:

```
import java.io.*;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.io.LongWritable;
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.MultipleInputs;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class Union {
        public static class MultipleMaps extends Mapper<LongWritable,Text,Text,IntWritable>
        {
                private final static IntWritable one=new IntWritable(1);
                private Text keyEmit=new Text();

                public void map(LongWritable k,Text value,Context context)throws IOException,
InterruptedException
                {
                        StringTokenizer itr=new StringTokenizer(value.toString());
                        while(itr.hasMoreElements())
                        {
```

```
                keyEmit.set(itr.nextToken());
                context.write(keyEmit, one);
            }
        }
    }
    public static class MultipleReducer 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
    {
        if(args.length!=3)
        {
            System.err.println("No of arguments should be 3");
            System.exit(0);
        }
        Configuration c=new Configuration();
        String[] files=new GenericOptionsParser(c,args).getRemainingArgs();
        Path p1=new Path(files[0]);
```

```
		Path p2=new Path(files[1]);
		Path p3=new Path(files[2]);
		FileSystem fs=FileSystem.get(c);

		if(fs.exists(p3)){
			fs.delete(p3,true);
		}

		Job job=Job.getInstance(c,"Union");
		job.setJarByClass(Union.class);

		MultipleInputs.addInputPath(job,p1,TextInputFormat.class,MultipleMaps.class);
		MultipleInputs.addInputPath(job,p2,TextInputFormat.class,MultipleMaps.class);

		job.setReducerClass(MultipleReducer.class);
		job.setCombinerClass(MultipleReducer.class);
		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(IntWritable.class);

		FileOutputFormat.setOutputPath(job, p3);
		boolean success=job.waitForCompletion(true);
		System.exit(success?0:1);

	}
}
```

Input File:

```
[cloudera@quickstart ~]$ cat setA.txt
aa bb cc dd
[cloudera@quickstart ~]$ cat setB.txt
bb cc dd ee
```

Output:

hadoop jar Union.jar Union /user/cloudera/inputfile/SetA.txt /user/cloudera/inputfile/SetB.txt /output_union

```
[cloudera@quickstart ~]$ hadoop jar Union.jar Union /user/cloudera/inputfile/set
A.txt /user/cloudera/inputfile/setB.txt /output_union
21/10/03 09:21:34 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
21/10/03 09:24:01 INFO input.FileInputFormat: Total input paths to process : 2
21/10/03 09:24:02 INFO mapreduce.JobSubmitter: number of splits:2
21/10/03 09:24:04 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_16
33273879556_0001
21/10/03 09:24:09 INFO impl.YarnClientImpl: Submitted application application_16
33273879556_0001
21/10/03 09:24:10 INFO mapreduce.Job: The url to track the job: http://quickstar
t.cloudera:8088/proxy/application_1633273879556_0001/
21/10/03 09:24:10 INFO mapreduce.Job: Running job: job_1633273879556_0001
21/10/03 09:27:14 INFO mapreduce.Job: Job job_1633273879556_0001 running in uber
 mode : false
21/10/03 09:27:14 INFO mapreduce.Job:  map 0% reduce 0%
21/10/03 09:28:12 INFO mapreduce.Job:  map 100% reduce 0%
21/10/03 09:28:33 INFO mapreduce.Job:  map 100% reduce 100%
21/10/03 09:28:34 INFO mapreduce.Job: Job job_1633273879556_0001 completed succe
ssfully
```

hadoop fs -cat /output_union/part-r-00000

```
[cloudera@quickstart ~]$ hadoop fs -cat /output_union/part-r-00000
aa      1
bb      1
cc      2
dd      2
ee      1
ff      1
```

Aim: Write a program in Map Reduce for Intersection operation.

```
import java.io.*;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.TextInputFormat;
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.MultipleInputs;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class Difference1 {
        public static class MultipleMapA extends Mapper<LongWritable,Text,Text,Text>
        {
                private Text keyEmit=new Text();
                private Text valEmit=new Text("A");
                public void map(LongWritable k,Text value,Context context)throws
IOException, InterruptedException
                {
                        StringTokenizer itr=new StringTokenizer(value.toString());
                        while(itr.hasMoreTokens())
```

```
                {
                        keyEmit.set(itr.nextToken());
                        context.write(keyEmit, valEmit);
                }
        }
}
public static class MultipleMapB extends Mapper<LongWritable,Text,Text,Text>
{
        private Text keyEmit=new Text();
        private Text valEmit=new Text("B");
        public void map(LongWritable k,Text value,Context context)throws
IOException, InterruptedException
        {
                StringTokenizer itr=new StringTokenizer(value.toString());
                while(itr.hasMoreTokens())
                {
                        keyEmit.set(itr.nextToken());
                        context.write(keyEmit, valEmit);
                }
        }
}
public static class MultipleReducer extends Reducer<Text,Text,Text,Text>
{
        private Text valEmit=new Text("");
        public void Reduce(Text key,Iterable<Text> values,Context context)throws
IOException, InterruptedException
        {
                boolean flag=true;
                for(Text val:values)
                {
                        if(val.toString().contains("B")){
```

```
                                flag=false;
                                break;
                        }
                }
                if(flag){
                        context.write(key, valEmit);
                }
        }
}
public static void main(String[] args)throws Exception
{
        if(args.length!=3){
                System.err.println("No of arguments should be 3");
                System.exit(0);
        }
        Configuration c=new Configuration();
        String[] files=new GenericOptionsParser(c,args).getRemainingArgs();
        Path p1=new Path(files[0]);
        Path p2=new Path(files[1]);
        Path p3=new Path(files[2]);
        FileSystem fs=FileSystem.get(c);
        if(fs.exists(p3)){
                fs.delete(p3,true);
        }
        Job job=Job.getInstance(c,"Difference1");
        job.setJarByClass(Difference1.class);
        MultipleInputs.addInputPath(job, p1,
TextInputFormat.class,MultipleMapA.class);
        MultipleInputs.addInputPath(job, p2,
TextInputFormat.class,MultipleMapB.class);
        job.setReducerClass(MultipleReducer.class);
```

```
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        FileOutputFormat.setOutputPath(job, p3);
        if(!job.waitForCompletion(true))
        {
            System.exit(1);
        }
    }
}
```

Input File:

```
[cloudera@quickstart ~]$ cat setA.txt
aa bb cc dd
[cloudera@quickstart ~]$ cat setB.txt
bb cc dd ee
```

Output:

```
hadoop jar Difference1.jar Difference1 /user/cloudera/inputfile/setA.txt /user/cloudera/inputfile/setB.txt /output_diff1
```

```
[cloudera@quickstart ~]$ hadoop jar Difference1.jar Difference1 /user/cloudera/inputfile/setA.txt /user/cloudera/inputfile/setB.txt /output_diff1
21/10/03 10:01:28 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
21/10/03 10:01:28 INFO input.FileInputFormat: Total input paths to process : 1
21/10/03 10:01:29 INFO input.FileInputFormat: Total input paths to process : 1
21/10/03 10:01:29 INFO mapreduce.JobSubmitter: number of splits:2
21/10/03 10:01:29 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1633273879556_0003
21/10/03 10:01:30 INFO impl.YarnClientImpl: Submitted application application_1633273879556_0003
21/10/03 10:01:30 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1633273879556_0003/
21/10/03 10:01:30 INFO mapreduce.Job: Running job: job_1633273879556_0003
21/10/03 10:01:36 INFO mapreduce.Job: Job job_1633273879556_0003 running in uber mode : false
21/10/03 10:01:36 INFO mapreduce.Job:  map 0% reduce 0%
21/10/03 10:02:07 INFO mapreduce.Job:  map 100% reduce 0%
21/10/03 10:02:43 INFO mapreduce.Job:  map 100% reduce 100%
21/10/03 10:02:44 INFO mapreduce.Job: Job job_1633273879556_0003 completed successfully
```

```
hadoop fs -cat /output_diff1/part-r-00000
```

```
[cloudera@quickstart ~]$ hadoop fs -cat /output_diff1/part-r-00000
aa
bb
```

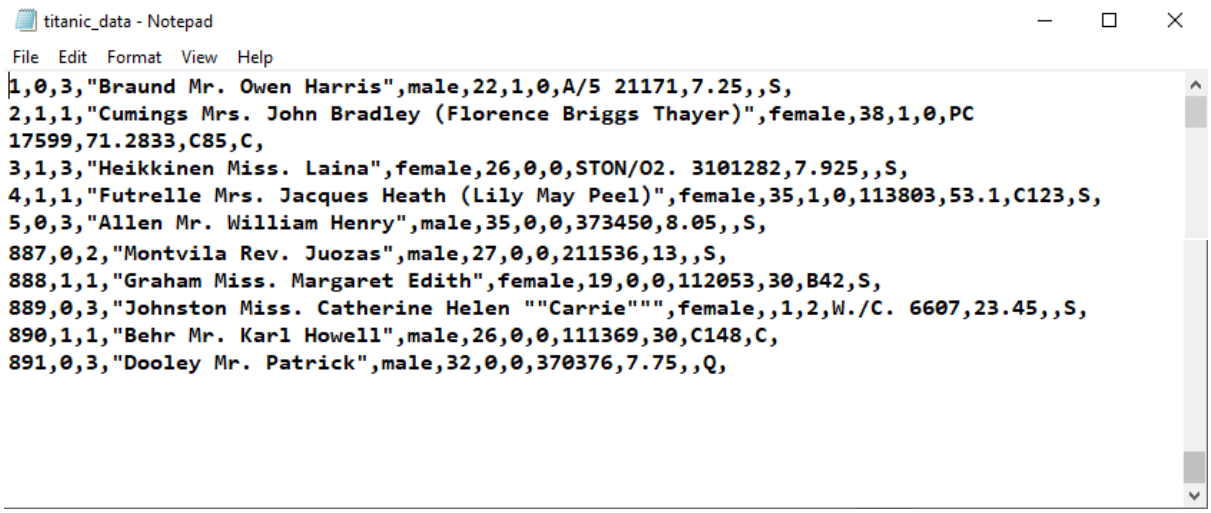
Aim: Write a program in Map Reduce for Grouping and Aggregation.

```
import java.io.IOException;
import java.io.IOException.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class AvgGroup {

        public static class Mapavg extends Mapper<LongWritable, Text, Text, IntWritable>
        {
                private Text gender=new Text();
                private IntWritable age=new IntWritable();

                public void map(LongWritable key,Text value,Context context)throws
IOException, InterruptedException
                {
                        String line=value.toString();
                        String str[]=line.split(",");
                        if(str.length>6)
                        {
                                gender.set(str[4]);
                                if(str[1].equals("0"))
                                {
```

```
                        if(str[5].matches("\\d+"))
                        {
                            int i=Integer.parseInt(str[5]);
                            age.set(i);
                        }
                    }
                }
                context.write(gender,age);
            }
        }
        public static class Reduceavg extends Reducer<Text, IntWritable, Text, IntWritable>
        {
            public void reduce(Text key, Iterable<IntWritable>values, Context
context)throws IOException, InterruptedException
            {
                int sum=0;
                int l=0;
                for(IntWritable val:values)
                {
                    l+=1;
                    sum+=val.get();
                }
                sum=sum/l;
                context.write(key, new IntWritable(sum));
            }
        }
        public static void main(String[] args)throws Exception
        {
            Configuration conf=new Configuration();
            Job job=Job.getInstance(conf,"AvgGroup");
            job.setJarByClass(AvgGroup.class);
```

```
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        job.setMapperClass(Mapavg.class);
        job.setReducerClass(Reduceavg.class);
        job.setCombinerClass(Reduceavg.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        FileInputFormat.addInputPath(job,new Path(args[0]));
        FileOutputFormat.setOutputPath(job,new Path(args[1]));
        Path out=new Path(args[1]);
        out.getFileSystem(conf).delete(out);
        job.waitForCompletion(true);
    }
}
```

Input File:



Output:

hadoop fs jar AVG.jar AvgGroup /user/cloudera/inputfile/Titanic.txt /outputAvg

```
[cloudera@quickstart ~]$ hadoop jar AVG.jar AvgGroup /user/cloudera/inputfile/Ti
tanic.txt /outputAvg
21/10/07 21:00:44 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
21/10/07 21:00:47 WARN mapreduce.JobResourceUploader: Hadoop command-line option
 parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
21/10/07 21:00:54 INFO input.FileInputFormat: Total input paths to process : 1
21/10/07 21:00:55 INFO mapreduce.JobSubmitter: number of splits:1
21/10/07 21:00:57 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_16
33663593785_0001
21/10/07 21:01:18 INFO impl.YarnClientImpl: Submitted application application_16
33663593785_0001
21/10/07 21:01:18 INFO mapreduce.Job: The url to track the job: http://quickstar
t.cloudera:8088/proxy/application_1633663593785_0001/
21/10/07 21:01:18 INFO mapreduce.Job: Running job: job_1633663593785_0001
21/10/07 21:02:15 INFO mapreduce.Job: Job job_1633663593785_0001 running in uber
 mode : false
21/10/07 21:02:15 INFO mapreduce.Job:  map 0% reduce 0%
21/10/07 21:02:46 INFO mapreduce.Job:  map 100% reduce 0%
21/10/07 21:03:08 INFO mapreduce.Job:  map 100% reduce 100%
21/10/07 21:03:08 INFO mapreduce.Job: Job job_1633663593785_0001 completed succe
ssfully
```

hadoop fs -cat /outputAvg/part-r-00000

```
[cloudera@quickstart ~]$ hadoop fs -ls /outputAvg
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2021-10-07 21:03 /outputAvg/_SUCCE
SS
-rw-r--r--   1 cloudera supergroup         18 2021-10-07 21:03 /outputAvg/part-r
-00000
[cloudera@quickstart ~]$ hadoop fs -cat /outputAvg/part-r-00000
female  28
male    30
```

Aim: Write a program in Map Reduce for Matrix Multiplication

Code:

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

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.Reducer.Context;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class Matrix {
	public static class MapM extends Mapper<LongWritable, Text, Text, Text> {

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

			Configuration conf = context.getConfiguration();
			int m = Integer.parseInt(conf.get("m"));
			int p = Integer.parseInt(conf.get("p"));

			String line = value.toString();

			String[] indicesAndValue = line.split(",");
			Text outputKey = new Text();
			Text outputValue = new Text();
```

```
                if(indicesAndValue[0].equals("M")){
                    for(int k = 0 ; k< p ; k++)
                    {
                        outputKey.set(indicesAndValue[1] + "," +  k);
                        outputValue.set(indicesAndValue[0] + "," + indicesAndValue[2] +
"," + indicesAndValue[3] );
                        context.write(outputKey, outputValue);

                    }
                }
                else
                {
                    for(int i=0 ; i<m; i++)
                    {
                        outputKey.set(i + "," + indicesAndValue[2]);
                        outputValue.set("N," + indicesAndValue[1] + "," +
indicesAndValue[3]);
                        context.write(outputKey, outputValue);
                    }
                }
            }
        }
        public static class ReduceM extends Reducer<Text, Text, Text, Text> {

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

                String[] value;
                HashMap<Integer, Float> hashA = new HashMap<Integer, Float>();
                HashMap<Integer, Float> hashB = new HashMap<Integer, Float>();
```

```
            for(Text val : values){
                value =val.toString().split(",");
                if(value[0].equals("M"))
                {
                    hashA.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));
                }
                else
                {
                    hashB.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));
                }

            }
            int n = Integer.parseInt(context.getConfiguration().get("n"));
            float result = 0.0f;
            float m_ij;
            float n_jk;
            for(int j=0; j<n; j++){
                m_ij = hashA.containsKey(j) ? hashA.get(j) : 0.0f;
                n_jk = hashB.containsKey(j) ? hashB.get(j) : 0.0f;

                result += m_ij * n_jk;

            }
            if(result != 0.0f){
                context.write(null, new Text( key.toString() + " " +
Float.toString(result)));
            }
        }
    }
```

```
    public static void main(String[] args) throws Exception{
        if(args.length != 2){
            System.err.println("use 2 arguments");
            System.exit(2);
        }
        Configuration conf = new Configuration();
        conf.set("m", "2");
        conf.set("n", "3");
        conf.set("p", "2");
        @SuppressWarnings("depcretion")

         Job job = new Job(conf,"Matrix");
        job.setJarByClass(Matrix.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        job.setMapperClass(MapM.class);
        job.setReducerClass(ReduceM.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

         FileInputFormat.addInputPath(job, new Path(args[0]));
       FileOutputFormat.setOutputPath(job, new Path(args[1]));
       job.waitForCompletion(true);

    }

}
```

Input File:

```
[cloudera@quickstart ~]$ cat final.txt
M,0,0,3
M,0,1,4
M,0,2,5
M,1,0,2
M,1,1,3
M,1,2,1
N,0,0,3
N,0,1,2
N,1,0,1
N,1,1,3
N,2,0,4
N,2,1,2
```

Output:

hadoop jar Matrix.jar Matrix /user/cloudera/inputfile/final.txt /output_59

```
[cloudera@quickstart ~]$ hadoop jar Matrix.jar Matrix /user/cloudera/inputfile/f
inal.txt /output_59
21/10/18 00:44:35 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
21/10/18 00:44:37 WARN mapreduce.JobResourceUploader: Hadoop command-line option
 parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
21/10/18 00:44:38 INFO input.FileInputFormat: Total input paths to process : 1
21/10/18 00:44:38 INFO mapreduce.JobSubmitter: number of splits:1
21/10/18 00:44:39 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_16
34534649142_0009
21/10/18 00:44:39 INFO impl.YarnClientImpl: Submitted application application_16
34534649142_0009
21/10/18 00:44:40 INFO mapreduce.Job: The url to track the job: http://quickstar
t.cloudera:8088/proxy/application_1634534649142_0009/
21/10/18 00:44:40 INFO mapreduce.Job: Running job: job_1634534649142_0009
21/10/18 00:45:28 INFO mapreduce.Job: Job job_1634534649142_0009 running in uber
 mode : false
21/10/18 00:45:28 INFO mapreduce.Job:  map 0% reduce 0%
21/10/18 00:46:26 INFO mapreduce.Job:  map 100% reduce 0%
21/10/18 00:46:49 INFO mapreduce.Job:  map 100% reduce 100%
21/10/18 00:46:50 INFO mapreduce.Job: Job job_1634534649142_0009 completed succe
ssfully
```

hadoop fs -cat /output_59/part-r-00000

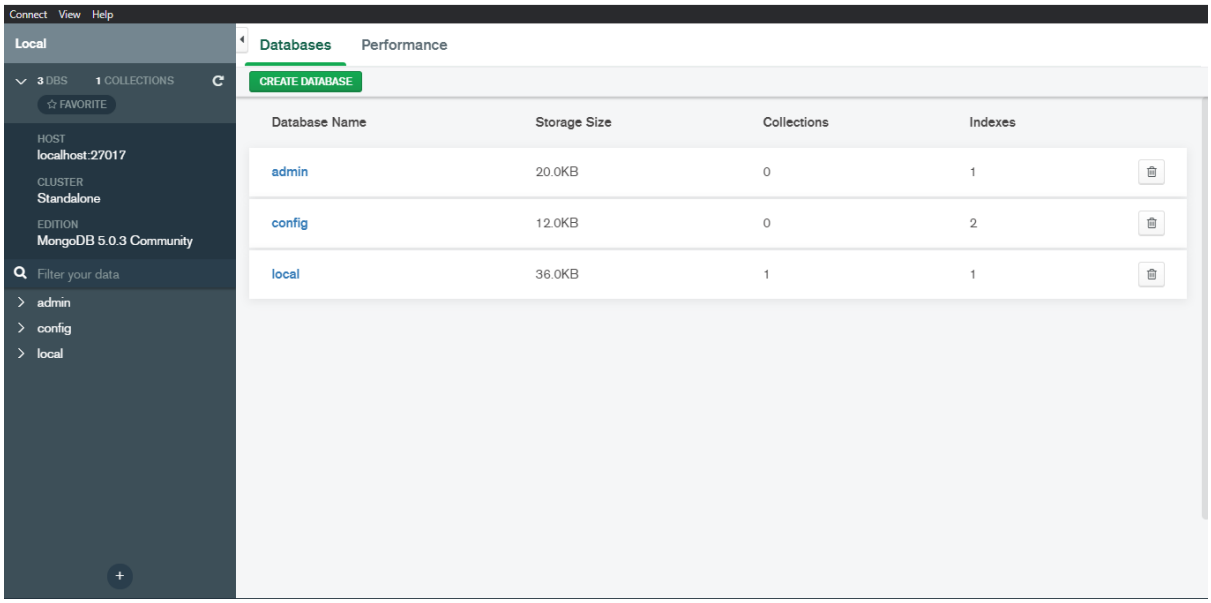
```
[cloudera@quickstart ~]$ hadoop fs -cat /output_59/part-r-00000
0,0 33.0
0,1 28.0
1,0 13.0
1,1 15.0
```

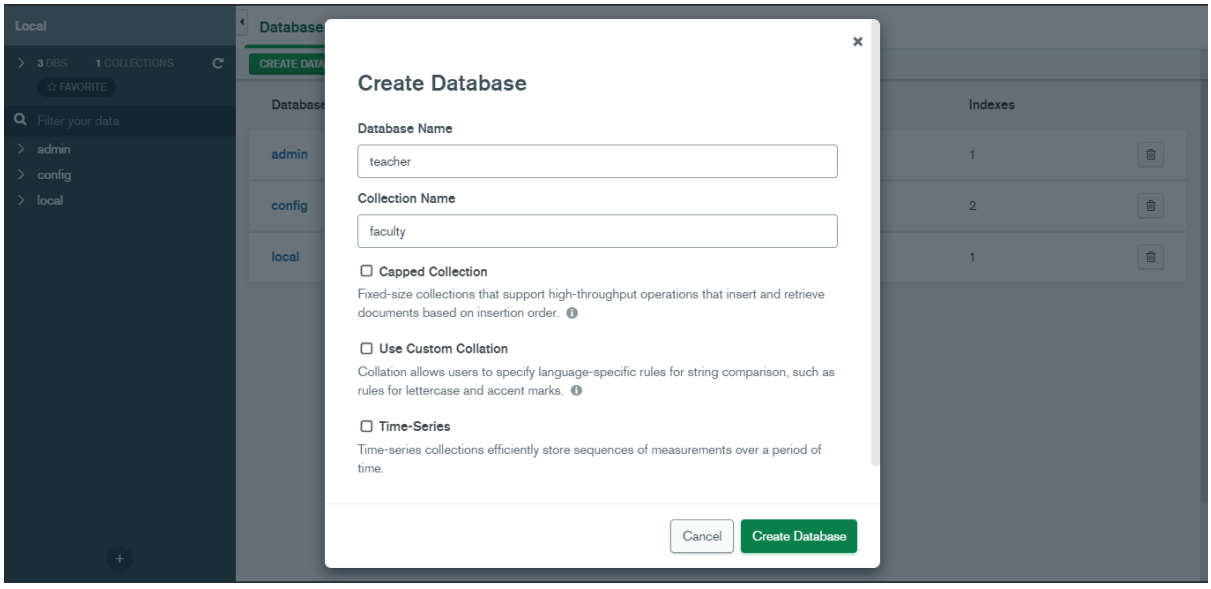
Practical No 3

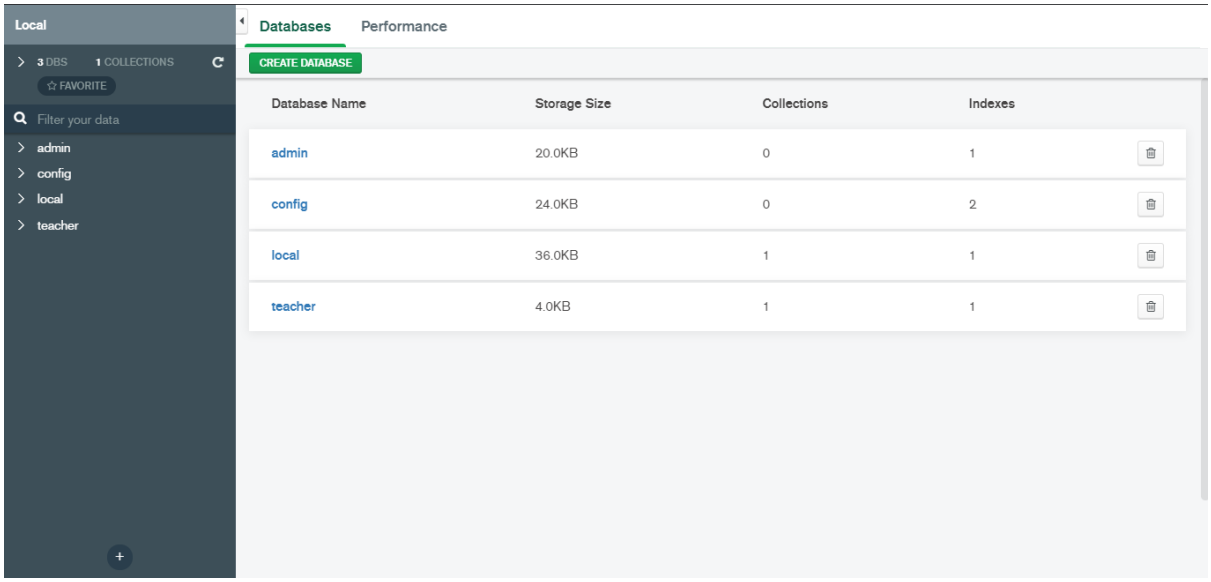
MongoDB

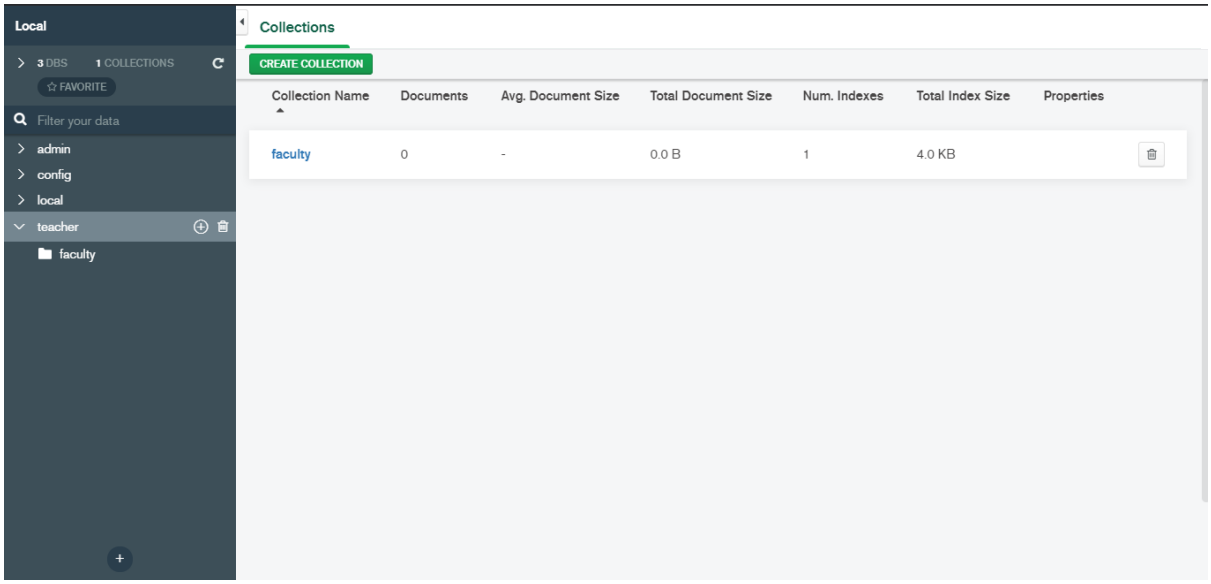
A) Installation

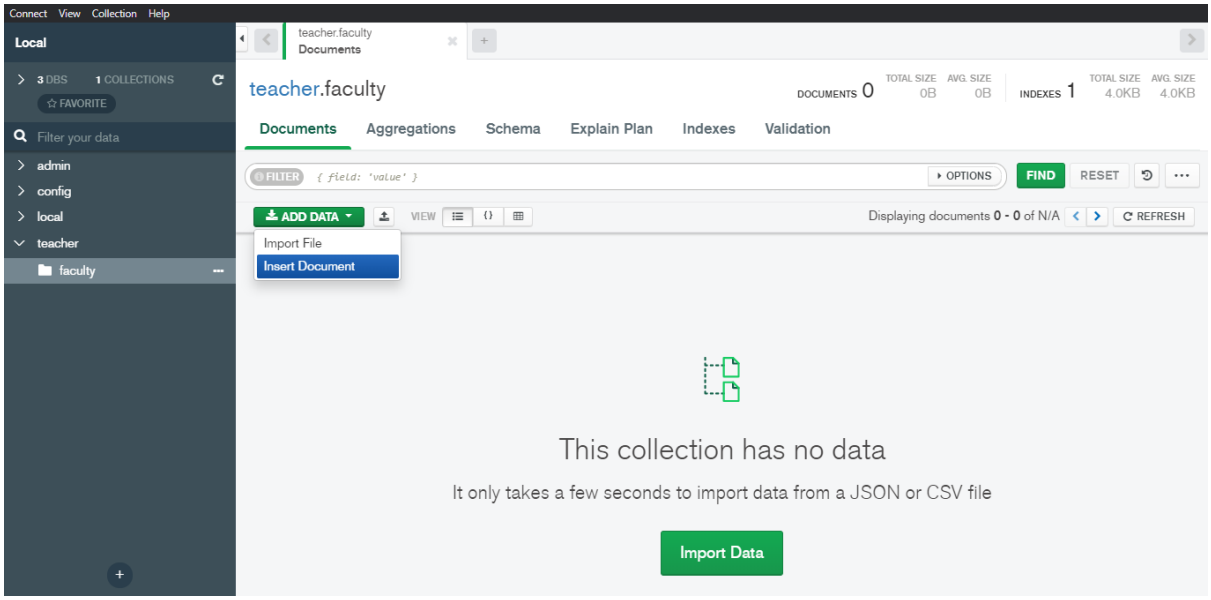
B) Sample Database Creation

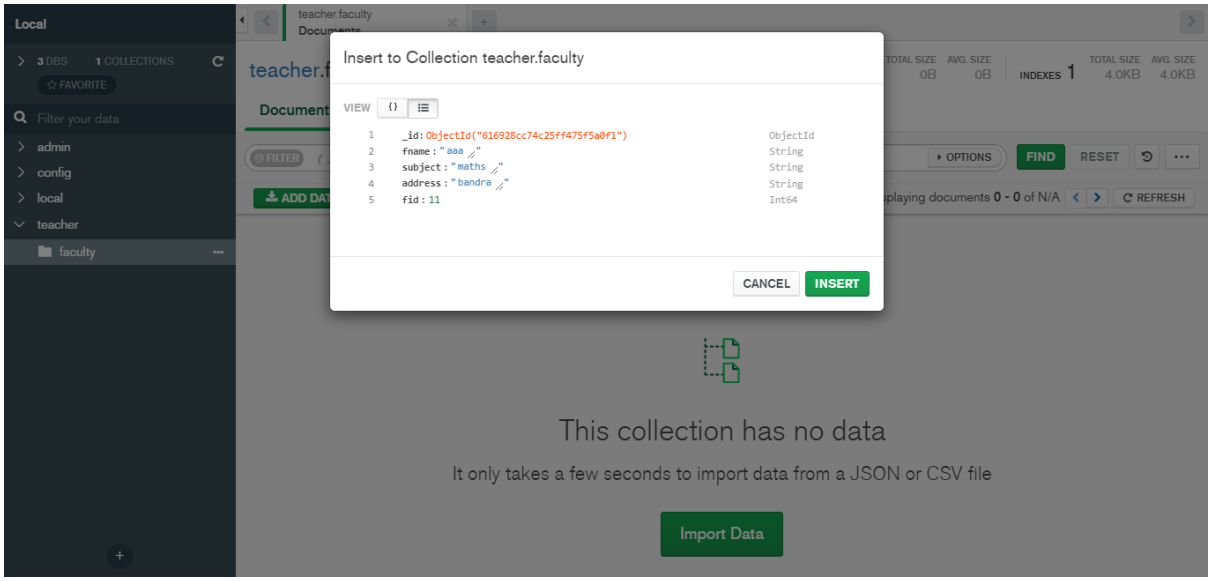


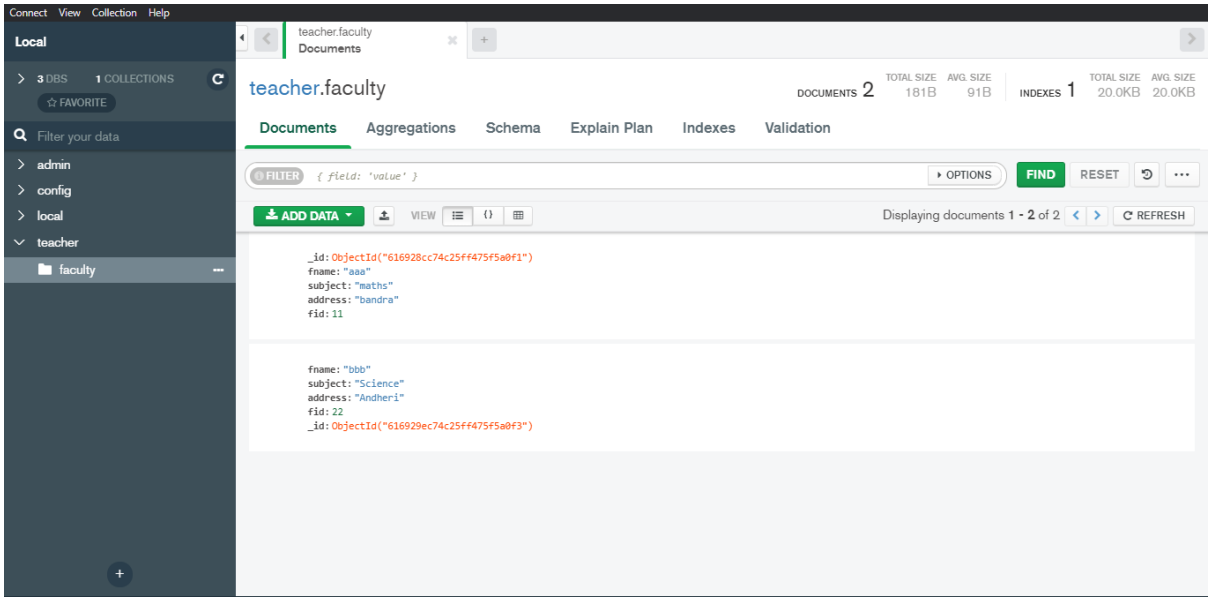



Local
3 DBS 1 COLLECTIONS
FAVORITE
Filter your data
admin
config
local
teacher
Databases
Performance
CREATE DATABASE
Database Name
Storage Size
Collections
Indexes
admin 20.0KB 0 1
config 24.0KB 0 2
local 36.0KB 1 1
teacher 4.0KB 1 1


Local
3 DBS 1 COLLECTIONS
FAVORITE
Filter your data
admin
config
local
teacher
faculty
Collections
CREATE COLLECTION
Collection Name
Documents
Avg. Document Size
Total Document Size
Num. Indexes
Total Index Size
Properties
faculty 0 - 0.0 B 1 4.0 KB


Connect View Collection Help
Local
3 DBS 1 COLLECTIONS
FAVORITE
Filter your data
admin
config
local
teacher
faculty
teacher.faculty
Documents
teacher.faculty
DOCUMENTS 0
TOTAL SIZE 0B
AVG. SIZE 0B
INDEXES 1
TOTAL SIZE 4.0KB
AVG. SIZE 4.0KB
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
FILTER { field: 'value' }
OPTIONS
FIND
RESET
ADD DATA
VIEW
Displaying documents 0 - 0 of N/A
REFRESH
Import File
Insert Document
This collection has no data
It only takes a few seconds to import data from a JSON or CSV file
Import Data





C) Query the Sample Database using MongoDB querying commands

a) Create Collection

In MongoDB, db.createCollection(name, options) is used to create collection. But usually you don’t need to create collection. MongoDB creates collection automatically when you insert some documents.

Syntax:

db.createCollection(**name**, options)

Example:

```
show dbs
```

```
> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
teacher  0.000GB
```

use teacher

```
> use teacher
switched to db teacher
```

show collections

```
> show collections
faculty
```

db.faculty.find()

```
> db.faculty.find()
{ "_id" : ObjectId("616928cc74c25ff475f5a0f1"), "fname" : "aaa", "subject" : "maths", "address" : "bandra", "fid" : Numb
erLong(11) }
{ "_id" : ObjectId("616929ec74c25ff475f5a0f3"), "fname" : "bbb", "subject" : "Science", "address" : "Andheri", "fid" : N
umberLong(22) }
```

db.createCollection("book")

```
> db.createCollection("book")
{ "ok" : 1 }
```

show collections

```
> show collections
book
faculty
```

b) Insert Document

In MongoDB, the **db.collection.insert()** method is used to add or insert new documents into a collection in your database.

Syntax: db.COLLECTION_NAME.**insert**(document)

Example:

db.book.insert({"b_id":33, "b_language":"English", "author":"xyz", "edition":"first"})

```
> db.book.insert({"b_id":33, "b_language":"English", "author":"xyz", "edition":"first"})
WriteResult({ "nInserted" : 1 })
```

db.book.find()

```
> db.book.find()
{ "_id" : ObjectId("61692cb20ef07ac0671e83a5"), "b_id" : 33, "b_language" : "English", "author" : "xyz", "edition" : "fi
rst" }
```

db.book.insert({"b_id":11, "b_language":"Marathi", "author":"abc", "edition":"first"})

db.book.insert({"b_id":22, "b_language":"Hindi", "author":"pqr", "edition":"second"})

```
> db.book.insert({"b_id":11, "b_language":"Marathi", "author":"abc", "edition":"first"})
WriteResult({ "nInserted" : 1 })
> db.book.insert({"b_id":22, "b_language":"Hindi", "author":"pqr", "edition":"second"})
WriteResult({ "nInserted" : 1 })
```

db.book.find()

```
> db.book.find()
{ "_id" : ObjectId("61692cb20ef07ac0671e83a5"), "b_id" : 33, "b_language" : "English", "author" : "xyz", "edition" : "fi
rst" }
{ "_id" : ObjectId("61692dd50ef07ac0671e83a6"), "b_id" : 11, "b_language" : "Marathi", "author" : "abc", "edition" : "fi
rst" }
{ "_id" : ObjectId("61692e020ef07ac0671e83a7"), "b_id" : 22, "b_language" : "Hindi", "author" : "pqr", "edition" : "seco
nd" }
```

c) Query Document

In MongoDB, the **db.collection.find()** method is used to retrieve documents from a collection. This method returns a cursor to the retrieved documents.

```
db.employee.insert([
    {"id":45,"firstname":"Tom", "lastname":"Cruise"},
    {"id":50,"firstname":"Maria", "lastname":"Sharapova"},
    {"id":37,"firstname":"James", "lastname":"Bond"},
    {"id":65,"firstname":"Eva", "lastname":"Bond"},
    {"id":59,"firstname":"Ginger", "lastname":"Bond"}])
```

```
> db.employee.insert([
... {"id":45,"firstname":"Tom", "lastname":"Cruise"},
... {"id":50,"firstname":"Maria", "lastname":"Sharapova"},
... {"id":37,"firstname":"James", "lastname":"Bond"},
... {"id":65,"firstname":"Eva", "lastname":"Bond"},
... {"id":59,"firstname":"Ginger", "lastname":"Bond"}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 5,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

```
db.employee.find()
```

```
> db.employee.find()
{ "_id" : ObjectId("616935d50ef07ac0671e83a9"), "id" : 45, "firstname" : "Tom", "lastname" : "Cruise" }
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
```

```
db.employee.find({"id":{$lt:50}}).pretty()
```

```
> db.employee.find({"id":{$lt:50}}).pretty()
{
        "_id" : ObjectId("616935d50ef07ac0671e83a9"),
        "id" : 45,
        "firstname" : "Tom",
        "lastname" : "Cruise"
}
{
        "_id" : ObjectId("616935d50ef07ac0671e83ab"),
        "id" : 37,
        "firstname" : "James",
        "lastname" : "Bond"
}
```

```
db.employee.find({"id":{$gt:50}}).pretty()
```

```
> db.employee.find({"id":{$gt:50}}).pretty()
{
        "_id" : ObjectId("616935d50ef07ac0671e83ac"),
        "id" : 65,
        "firstname" : "Eva",
        "lastname" : "Bond"
}
{
        "_id" : ObjectId("616935d50ef07ac0671e83ad"),
        "id" : 59,
        "firstname" : "Ginger",
        "lastname" : "Bond"
}
```

db.employee.find({"firstname":{$in:["Tom","Eva","Ginger"]}}).pretty()

```
> db.employee.find({"firstname":{$in:["Tom","Eva","Ginger"]}}).pretty()
{
        "_id" : ObjectId("616935d50ef07ac0671e83a9"),
        "id" : 45,
        "firstname" : "Tom",
        "lastname" : "Cruise"
}
{
        "_id" : ObjectId("616935d50ef07ac0671e83ac"),
        "id" : 65,
        "firstname" : "Eva",
        "lastname" : "Bond"
}
{
        "_id" : ObjectId("616935d50ef07ac0671e83ad"),
        "id" : 59,
        "firstname" : "Ginger",
        "lastname" : "Bond"
}
```

db.employee.find({"firstname":{$nin:["Tom","Eva","Ginger"]}}).pretty()

```
> db.employee.find({"firstname":{$nin:["Tom","Eva","Ginger"]}}).pretty()
{
        "_id" : ObjectId("616935d50ef07ac0671e83aa"),
        "id" : 50,
        "firstname" : "Maria",
        "lastname" : "Sharapova"
}
{
        "_id" : ObjectId("616935d50ef07ac0671e83ab"),
        "id" : 37,
        "firstname" : "James",
        "lastname" : "Bond"
}
```

db.employee.find({"id":{$ne:50}}).pretty()

```
> db.employee.find({"id":{$ne:50}}).pretty()
{
        "_id" : ObjectId("616935d50ef07ac0671e83a9"),
        "id" : 45,
        "firstname" : "Tom",
        "lastname" : "Cruise"
}
{
        "_id" : ObjectId("616935d50ef07ac0671e83ab"),
        "id" : 37,
        "firstname" : "James",
        "lastname" : "Bond"
}
{
        "_id" : ObjectId("616935d50ef07ac0671e83ac"),
        "id" : 65,
        "firstname" : "Eva",
        "lastname" : "Bond"
}
{
        "_id" : ObjectId("616935d50ef07ac0671e83ad"),
        "id" : 59,
        "firstname" : "Ginger",
        "lastname" : "Bond"
}
```

db.employee.find()

```
> db.employee.find()
{ "_id" : ObjectId("616935d50ef07ac0671e83a9"), "id" : 45, "firstname" : "Tom", "lastname" : "Cruise" }
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
```

```
db.employee.update({"firstname": "Tom"},{$set:{"firstname": "Jerry"}})
```

```
> db.employee.update({"firstname": "Tom"},{$set:{"firstname": "Jerry"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```
db.employee.find()
```

```
> db.employee.find()
{ "_id" : ObjectId("616935d50ef07ac0671e83a9"), "id" : 45, "firstname" : "Jerry", "lastname" : "Cruise" }
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
```

```
db.employee.insert([

        {"id": 45, "firstname": "Tom","lastname": "Cruise"},

        {"id": 45,"firstname": "Tom","lastname": "Cruise"}

])
```

```
> db.employee.insert([
... {
... "id": 45,
... "firstname": "Tom",
... "lastname": "Cruise"
... },
... {
... "id": 45,
... "firstname": "Tom",
... "lastname": "Cruise"
... }
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

```
db.employee.find()
```

```
> db.employee.find()
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b2"), "id" : 45, "firstname" : "Tom", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b3"), "id" : 45, "firstname" : "Tom", "lastname" : "Cruise" }
```

d) Delete Document

In MongoDB, the db.colloction.remove() method is used to delete documents from a collection. The remove() method works on two parameters.

Syntax: db.collection_name.remove (DELETION_CRITERIA)

```
show dbs
```

```
> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
student  0.000GB
teacher  0.000GB
```

use student

```
> use student
switched to db student
```

db.dropDatabase()

```
> db.dropDatabase()
{ "ok" : 1 }
```

show dbs

```
> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
teacher  0.000GB
```

```
db.employee.insert([
        {"id":78,"firstname": "Emma","lastname": "Cruise"},
        {"id": 79,"firstname": "Emma","lastname": "Cruise"}
])
```

```
> db.employee.insert([
... {"id":78,"firstname": "Emma","lastname": "Cruise"},
... {"id": 79,"firstname": "Emma","lastname": "Cruise"}
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

db.employee.find()

```
> db.employee.find()
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b2"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b3"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693dc80ef07ac0671e83b4"), "id" : 78, "firstname" : "Emma", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693dc80ef07ac0671e83b5"), "id" : 79, "firstname" : "Emma", "lastname" : "Cruise" }
```

db.employee.remove({"id":78})

```
> db.employee.remove({"id":78})
WriteResult({ "nRemoved" : 1 })
```

db.employee.find()

```
> db.employee.find()
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b2"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b3"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693dc80ef07ac0671e83b5"), "id" : 79, "firstname" : "Emma", "lastname" : "Cruise" }
```

e) Indexing

```
db.book.findOne({"edition":"first"})
```

```
> db.book.findOne({"edition":"first"})
{
        "_id" : ObjectId("61692cb20ef07ac0671e83a5"),
        "b_id" : 33,
        "b_language" : "English",
        "author" : "xyz",
        "edition" : "first"
}
```

```
show collections
```

```
> show collections
book
faculty
```

```
db.library.insert({"library_id":56, "lib_name":"hiray", "lib_address":"bandra"})
```

```
> db.library.insert({"library_id":56, "lib_name":"hiray", "lib_address":"bandra"})
WriteResult({ "nInserted" : 1 })
```

```
show collections
```

```
> show collections
book
faculty
library
```

```
db.employee.update({"firstname":"Tom"},{$set:{"firstname": "Scooby"}},{multi:true})
```

```
> db.employee.update({"firstname":"Tom"},{$set:{"firstname": "Scooby"}},{multi:true})
WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2 })
```

```
db.employee.find()
```

```
> db.employee.find()
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b2"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b3"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
```

```
db.employee.find().sort({"id":1})
```

```
> db.employee.find().sort({"id":1})
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b2"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b3"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("61693dc80ef07ac0671e83b5"), "id" : 79, "firstname" : "Emma", "lastname" : "Cruise" }
```

```
db.employee.find().sort({"id":-1})
```

```
> db.employee.find().sort({"id":-1})
{ "_id" : ObjectId("61693dc80ef07ac0671e83b5"), "id" : 79, "firstname" : "Emma", "lastname" : "Cruise" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b2"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b3"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
```

```
db.employee.drop()
```

```
> db.employee.drop()
true
```

```
db.employee.insert([
{
"userId":"rirani", "jobTitleName":"Developer", "firstName":"Romin","lastName":"Irani",
"preferredFullName":"Romin Irani","employeeCode":"E1","region":"CA",
"phoneNumber":"408-1234567","emailAddress":"romin.k.irani@gmail.com","age":42
},
{
"userId":"nirani","jobTitleName":"Developer","firstName":"Neil","lastName":"Irani",
"preferredFullName":"Neil Irani","employeeCode":"E2","region":"CA",
"phoneNumber":"408-1111111","emailAddress":"neilrirani@gmail.com","age":36
},
{
"userId":"thanks","jobTitleName":"Program Directory","firstName":"Tom","lastName":"Hanks",
"preferredFullName":"Tom Hanks","employeeCode":"E3","region":"CA",
"phoneNumber":"408-2222222","emailAddress":"tomhanks@gmail.com","age":34
},
{
"userId":"rirani","jobTitleName":"Developer","firstName":"Romin","lastName":"Irani",
"preferredFullName":"Romin Irani","employeeCode":"E1","region":"CB",
"phoneNumber":"408-1234567","emailAddress":"romin.k.irani@gmail.com","age":45
},
{
"userId":"nirani","jobTitleName":"Developer","firstName":"Neil","lastName":"Irani",
"preferredFullName":"Neil Irani","employeeCode":"E2","region":"CB",
"phoneNumber":"408-1111111","emailAddress":"neilrirani@gmail.com","age":32
},
{
"userId":"thanks","jobTitleName":"Program Directory","firstName":"Tom","lastName":"Hanks",
"preferredFullName":"Tom Hanks","employeeCode":"E3","region":"CB",
"phoneNumber":"408-2222222","emailAddress":"tomhanks@gmail.com","age":36
}])
```

```
> db.employee.find()
{ "_id" : ObjectId("61705a49309ea63c54635fe0"), "userId" : "rirani", "jobTitleName" : "Developer", "firstName" : "Romin"
, "lastName" : "Irani", "preferredFullName" : "Romin Irani", "employeeCode" : "E1", "region" : "CA", "phoneNumber" : "40
8-1234567", "emailAddress" : "romin.k.irani@gmail.com", "age" : 43 }
{ "_id" : ObjectId("61705a49309ea63c54635fe1"), "userId" : "nirani", "jobTitleName" : "Developer", "firstName" : "Neil",
 "lastName" : "Irani", "preferredFullName" : "Neil Irani", "employeeCode" : "E2", "region" : "CA", "phoneNumber" : "408-
1111111", "emailAddress" : "neilrirani@gmail.com", "age" : 36 }
{ "_id" : ObjectId("61705a49309ea63c54635fe2"), "userId" : "thanks", "jobTitleName" : "Program Directory", "firstName" :
 "Tom", "lastName" : "Hanks", "preferredFullName" : "Tom Hanks", "employeeCode" : "E3", "region" : "CA", "phoneNumber" :
 "408-2222222", "emailAddress" : "tomhanks@gmail.com", "age" : 34 }
{ "_id" : ObjectId("61705d73309ea63c54635fe3"), "userId" : "rirani", "jobTitleName" : "Developer", "firstName" : "Romin"
, "lastName" : "Irani", "preferredFullName" : "Romin Irani", "employeeCode" : "E1", "region" : "CB", "phoneNumber" : "40
8-1234567", "emailAddress" : "romin.k.irani@gmail.com", "age" : 45 }
{ "_id" : ObjectId("61705d73309ea63c54635fe4"), "userId" : "nirani", "jobTitleName" : "Developer", "firstName" : "Neil",
 "lastName" : "Irani", "preferredFullName" : "Neil Irani", "employeeCode" : "E2", "region" : "CB", "phoneNumber" : "408-
1111111", "emailAddress" : "neilrirani@gmail.com", "age" : 32 }
{ "_id" : ObjectId("61705d73309ea63c54635fe5"), "userId" : "thanks", "jobTitleName" : "Program Directory", "firstName" :
 "Tom", "lastName" : "Hanks", "preferredFullName" : "Tom Hanks", "employeeCode" : "E3", "region" : "CB", "phoneNumber" :
 "408-2222222", "emailAddress" : "tomhanks@gmail.com", "age" : 36 }
```

db.employee.aggregate([{$group{_id:"$region",age_s:{$sum:"$age"}}}])

```
> db.employee.aggregate([{$group:{_id:"$region",age_s:{$sum:"$age"}}}])
{ "_id" : "CB", "age_s" : 113 }
{ "_id" : "CA", "age_s" : 113 }
```

db.employee.find({"userId":"rirani"}).limit(2)

```
> db.employee.find({"userId":"rirani"}).limit(2).pretty()
{
        "_id" : ObjectId("61705a49309ea63c54635fe0"),
        "userId" : "rirani",
        "jobTitleName" : "Developer",
        "firstName" : "Romin",
        "lastName" : "Irani",
        "preferredFullName" : "Romin Irani",
        "employeeCode" : "E1",
        "region" : "CA",
        "phoneNumber" : "408-1234567",
        "emailAddress" : "romin.k.irani@gmail.com",
        "age" : 43
}
{
        "_id" : ObjectId("61705d73309ea63c54635fe3"),
        "userId" : "rirani",
        "jobTitleName" : "Developer",
        "firstName" : "Romin",
        "lastName" : "Irani",
        "preferredFullName" : "Romin Irani",
        "employeeCode" : "E1",
        "region" : "CB",
        "phoneNumber" : "408-1234567",
        "emailAddress" : "romin.k.irani@gmail.com",
        "age" : 45
}
```

Hive

```
CREATE TABLE empy(eid int, name String, Salary String)

COMMENT 'Employee Details'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

STORED AS TEXTFILE;
```

```
hive> CREATE TABLE empy(E_id int, Name String, Salary String)
    > COMMENT 'Employee Details'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LINES TERMINATED BY '\n'
    > STORED AS TEXTFILE;
OK
Time taken: 40.901 seconds
```

Input File:

```
[cloudera@quickstart ~]$ cat data.txt
1001,aaa,10000
1002,bbb,20000
1003,ccc,30000
1004,ddd,40000
1005,eee,50000
1006,fff,60000
```

```
load data local inpath '/home/cloudera/data.txt' overwrite into table empy;
```

```
hive> load data local inpath '/home/cloudera/data.txt' overwrite into table empy
;
Loading data to table default.empy
Table default.empy stats: [numFiles=1, numRows=0, totalSize=90, rawDataSize=0]
OK
Time taken: 10.542 seconds
```

```
Select * from empy;
```

```
hive> select * from empy;
OK
1001    aaa     10000
1002    bbb     20000
1003    ccc     30000
1004    ddd     40000
1005    eee     50000
1006    fff     60000
Time taken: 8.504 seconds, Fetched: 6 row(s)
```

```
Alter table empy rename to Employee;
```

```
hive> alter table empy rename to Employee;
OK
Time taken: 0.216 seconds
hive> select * from Employee;
OK
1001    aaa     10000
1002    bbb     20000
1003    ccc     30000
1004    ddd     40000
1005    eee     50000
1006    fff     60000
Time taken: 0.076 seconds, Fetched: 6 row(s)
```

Alter table Employee change Salary Salary double;

```
hive> alter table Employee change Salary Salary double;
OK
Time taken: 0.202 seconds
hive> select * from Employee;
OK
1001    aaa     10000.0
1002    bbb     20000.0
1003    ccc     30000.0
1004    ddd     40000.0
1005    eee     50000.0
1006    fff     60000.0
Time taken: 0.101 seconds, Fetched: 6 row(s)
```

Select * from Employee where Salary > 10000;

```
hive> select * from Employee where Salary > 10000;
OK
1002    bbb     20000.0
1003    ccc     30000.0
1004    ddd     40000.0
1005    eee     50000.0
1006    fff     60000.0
Time taken: 0.58 seconds, Fetched: 5 row(s)
```

Select * from Employee order by Salary;

```
Total MapReduce CPU Time Spent: 3 seconds 430 msec
OK
1001    aaa     10000.0
1002    bbb     20000.0
1003    ccc     30000.0
1004    ddd     40000.0
1005    eee     50000.0
1006    fff     60000.0
Time taken: 110.567 seconds, Fetched: 6 row(s)
```

Select Salary, count(*) from Employee group by Salary;

```
Total MapReduce CPU Time Spent: 12 seconds 240 msec
OK
10000.0 1
20000.0 1
30000.0 1
40000.0 1
50000.0 1
60000.0 1
Time taken: 234.217 seconds, Fetched: 6 row(s)
```

Input File:

```
[cloudera@quickstart ~]$ cat customer.txt
11,Sunil,32,15000
12,Sumeet,24,12000
13,Sunio,22,25000
14,Kumar,24,30000
15,Shubham,25,25000
```

Select * from customer;

```
hive> select * from customer;
OK
11      Sunil   32      15000
12      Sumeet  24      12000
13      Sunio   22      25000
14      Kumar   24      30000
15      Shubham 25      25000
Time taken: 3.523 seconds, Fetched: 5 row(s)
```

Select * from order;

```
hive> select * from order;
OK
101     11      2345    Cash
102     13      1150    Cash
103     14      2410    Card
104     15      1180    Card
105     16      2309    Cash
106     17      2900    Card
Time taken: 0.207 seconds, Fetched: 6 row(s)
```

Select c.cust_id, c.cust_name, c.age, c.salary, o.amount, o.payment from customer c LEFT OUTER JOIN order o on(c.cust_id=o.customer_id);

```
Total MapReduce CPU Time Spent: 5 seconds 210 msec
OK
11      Sunil   32      15000   2345    Cash
12      Sumeet  24      12000   NULL    NULL
13      Sunio   22      25000   1150    Cash
14      Kumar   24      30000   2410    Card
15      Shubham 25      25000   1180    Card
Time taken: 118.222 seconds, Fetched: 5 row(s)
```

Select c.cust_id, c.cust_name, c.age, c.salary, o.amount, o.payment from customer c RIGHT OUTER JOIN order o on(c.cust_id=o.customer_id);

```
Total MapReduce CPU Time Spent: 6 seconds 280 msec
OK
11      Sunil   32      15000   2345    Cash
13      Sunio   22      25000   1150    Cash
14      Kumar   24      30000   2410    Card
15      Shubham 25      25000   1180    Card
NULL    NULL    NULL    NULL    2309    Cash
NULL    NULL    NULL    NULL    2900    Card
Time taken: 103.656 seconds, Fetched: 6 row(s)
```

Select c.cust_id, c.cust_name, c.age, c.salary, o.amount, o.payment from customer c FULL OUTER JOIN order o on(c.cust_id=o.customer_id);

```
Total MapReduce CPU Time Spent: 49 seconds 300 msec
OK
11      Sunil   32      15000   2345    Cash
12      Sumeet  24      12000   NULL    NULL
13      Sunio   22      25000   1150    Cash
14      Kumar   24      30000   2410    Card
15      Shubham 25      25000   1180    Card
NULL    NULL    NULL    NULL    2309    Cash
NULL    NULL    NULL    NULL    2900    Card
Time taken: 133.087 seconds, Fetched: 7 row(s)
```

Select * from customer;

```
hive> select * from customer;
OK
11      Sunil   32      15000
12      Sumeet  24      12000
13      Sunio   22      25000
14      Kumar   24      30000
15      Shubham 25      25000
Time taken: 0.384 seconds, Fetched: 5 row(s)
```

Select salary, count(*) from customer group by salary;

```
Total MapReduce CPU Time Spent: 6 seconds 220 msec
OK
12000   1
15000   1
25000   2
30000   1
Time taken: 49.098 seconds, Fetched: 4 row(s)
```

Select min(salary) from customer;

```
Total MapReduce CPU Time Spent: 4 seconds 950 msec
OK
12000
Time taken: 32.656 seconds, Fetched: 1 row(s)
```

Select avg(salary) from customer;

```
Total MapReduce CPU Time Spent: 15 seconds 120 msec
OK
21400.0
Time taken: 55.328 seconds, Fetched: 1 row(s)
```

Select max(salary) from customer;

```
Total MapReduce CPU Time Spent: 21 seconds 920 msec
OK
30000
Time taken: 127.657 seconds, Fetched: 1 row(s)
```

Select count(*) from customer;

```
Total MapReduce CPU Time Spent: 12 seconds 520 msec
OK
5
Time taken: 240.625 seconds, Fetched: 1 row(s)
```

Pig

Input File:

```
[cloudera@quickstart ~]$ cat pigfile.txt
1,2,5
2,1,4
1,2,5
5,6,7
4,8,9
3,9,4
```

```
data = LOAD '/user/cloudera/piginput/pigfile.txt' USING PigStorage (',') AS (A1:int, A2:int, A3:int);

dump data;
```

```
(1,2,5)
(2,1,4)
(1,2,5)
(5,6,7)
(4,8,9)
(3,9,4)
```

```
result = DISTINCT data;

dump result;
```

```
(1,2,5)
(2,1,4)
(3,9,4)
(4,8,9)
(5,6,7)
```

```
fltresult= FILTER data BY A3==4;

dump fltresult;
```

```
(2,1,4)
(3,9,4)
```

```
feresult= FOREACH data generate A1,A2;

dump feresult;
```

```
(1,2)
(2,1)
(1,2)
(5,6)
(4,8)
(3,9)
```

Input File:

```
[cloudera@quickstart ~]$ cat employee.txt
Jhon,32,87
Emma,33,88
Emma,30,90
Aliya,31,89
Tom,31,90
Pooja,32,93
Kajal,34,95
Tom,30,89
Tom,31,93
Emma,29,88
```

```
student = LOAD '/user/cloudera/piginput/employee.txt' USING PigStorage(',') AS
(name:charArray, Age:int, Grade:int);

dump student;
```

```
(Jhon,32,87)
(Emma,33,88)
(Emma,30,90)
(Aliya,31,89)
(Tom,31,90)
(Pooja,32,93)
(Kajal,34,95)
(Tom,30,89)
(Tom,31,93)
(Emma,29,88)
```

```
grpname= GROUP student BY name;

dump grpname;
```

```
(Tom,{(Tom,31,93),(Tom,30,89),(Tom,31,90)})
(Emma,{(Emma,29,88),(Emma,30,90),(Emma,33,88)})
(Jhon,{(Jhon,32,87)})
(Aliya,{(Aliya,31,89)})
(Kajal,{(Kajal,34,95)})
(Pooja,{(Pooja,32,93)})
```

```
lmtresult = LIMIT student 2;

dump lmtresult;
```

```
(Emma,33,88)
(Jhon,32,87)
```

```
ascresult = ORDER student BY name;

dump ascresult;
```

```
(Aliya,31,89)
(Emma,29,88)
(Emma,30,90)
(Emma,33,88)
(Jhon,32,87)
(Kajal,34,95)
(Pooja,32,93)
(Tom,31,93)
(Tom,30,89)
(Tom,31,90)
```

```
descresult = ORDER student BY name DESC;

dump descresult;
```

```
(Tom,31,93)
(Tom,30,89)
(Tom,31,90)
(Pooja,32,93)
(Kajal,34,95)
(Jhon,32,87)
(Emma,29,88)
(Emma,30,90)
(Emma,33,88)
(Aliya,31,89)
```

```
groupname = GROUP student BY  name;

avgres= FOREACH groupname generate student.name, AVG(student.Grade);

dump avgres;
```

```
({(Tom),(Tom),(Tom)},90.66666666666667)
({(Emma),(Emma),(Emma)},88.66666666666667)
({(Jhon)},87.0)
({(Aliya)},89.0)
({(Kajal)},95.0)
({(Pooja)},93.0)
```

```
groupname = GROUP student BY name;

minres= FOREACH groupname generate student.name, MIN(student.Grade);

dump minres;
```

```
({(Tom),(Tom),(Tom)},89)
({(Emma),(Emma),(Emma)},88)
({(Jhon)},87)
({(Aliya)},89)
({(Kajal)},95)
({(Pooja)},93)
```

```
groupname = GROUP student BY name;

maxres= FOREACH groupname generate student.name, MAX(student.Grade);
```

```
dump maxres;
```

```
({(Tom),(Tom),(Tom)},93)
({(Emma),(Emma),(Emma)},90)
({(Jhon)},87)
({(Aliya)},89)
({(Kajal)},95)
({(Pooja)},93)
```

```
sizeres = FOREACH student generate SIZE(name);

dump sizeres;
```

```
(4)
(4)
(4)
(5)
(3)
(5)
(5)
(3)
(3)
(4)
```

```
groupname = GROUP student BY name;

countres= FOREACH groupname generate COUNT(student);

dump countres;
```

```
(3)
(3)
(1)
(1)
(1)
(1)
```

Input File:

```
[cloudera@quickstart ~]$ cat matrixA.txt
4,5
3,6
[cloudera@quickstart ~]$ cat matrixB.txt
3,3,5
2,6,9
```

```
matrixA= LOAD '/user/cloudera/piginput/matrixA.txt' USING PigStorage(',') AS(a1:int, a2:int);

dump matrixA
```

```
(4,5)
(3,6)
```

```
matrixB= LOAD '/user/cloudera/piginput/matrixB.txt' USING PigStorage(',') AS(a1:int, a2:int,
a3:int);

dump matrixB
```

```
(3,3,5)
(2,6,9)
```

```
Crossresult= CROSS matrixA,matrixB;

dump crossresult;
```

```
(3,6,2,6,9)
(3,6,3,3,5)
(4,5,2,6,9)
(4,5,3,3,5)
```

Input File:

```
[cloudera@quickstart ~]$ cat ab.txt
1.8
2.7
9.6
-1.7
3.2
5.9
-2.7
```

```
AB = LOAD '/user/cloudera/piginput/ab.txt' AS(ab1:float);

dump AB;
```

```
(1.8)
(2.7)
(9.6)
(-1.7)
(3.2)
(5.9)
(-2.7)
```

```
abresult= FOREACH AB GENERATE ABS(ab1);

dump abresult;
```

```
(1.8)
(2.7)
(9.6)
(1.7)
(3.2)
(5.9)
(2.7)
```

Input File:

```
[cloudera@quickstart ~]$ cat cube.txt
8
27
64
125
216
343
```

```
CB = LOAD '/user/cloudera/piginput/cube.txt' AS(c1:int);

dump CB;
```

```
(8)
(27)
(64)
(125)
(216)
(343)
```

```
cbresult= FOREACH CB GENERATE CBRT(c1);

dump cbresult;
```

```
(2.0)
(3.0)
(4.0)
(5.0)
(6.0)
(7.0)
```

Input File:

```
[cloudera@quickstart ~]$ cat floatcube.txt
5.0
6.5
8.2
5.9
3.1
```

```
FLCB = LOAD '/user/cloudera/piginput/floatcube.txt' AS(c1:float);

dump FLCB;
```

```
(5.0)
(6.5)
(8.2)
(5.9)
(3.1)
```

```
flcbresult= FOREACH FLCB GENERATE CBRT(c1);

dump flcbresult;
```

```
(1.709975946676697)
(1.866255578408624)
(2.0165296595830426)
(1.8069688790571206)
(1.4580997208745365)
```

Input File:

```
[cloudera@quickstart ~]$ cat cell.txt
1.2
2.4
1.1
3.4
2.7
7.2
```

```
cell = LOAD '/user/cloudera/piginput/cell.txt' AS(ce:float);

dump cell;
```

```
(1.2)
(2.4)
(1.1)
(3.4)
(2.7)
(7.2)
```

```
cellresult= FOREACH cell GENERATE (ce), CEIL(ce);

dump cellresult;
```

```
(1.2,2.0)
(2.4,3.0)
(1.1,2.0)
(3.4,4.0)
(2.7,3.0)
(7.2,8.0)
```

```
floorresult= FOREACH cell GENERATE (ce), FLOOR(ce);

dump floorresult;
```

```
(1.2,1.0)
(2.4,2.0)
(1.1,1.0)
(3.4,3.0)
(2.7,2.0)
(7.2,7.0)
```

```
roundresult= FOREACH cell GENERATE (ce), ROUND(ce);

dump roundresult;
```

```
(1.2,1)
(2.4,2)
(1.1,1)
(3.4,3)
(2.7,3)
(7.2,7)
```

Input File:

```
[cloudera@quickstart ~]$ cat log.txt
2
3
7
8
1
6
```

```
logdata= LOAD '/user/cloudera/piginput/log.txt' AS(l1:int);

dump logdata;
```

```
(2)
(3)
(7)
(8)
(1)
(6)
```

```
logresult= FOREACH logdata GENERATE (l1), LOG(l1);

dump logresult;
```

```
(2,0.6931471805599453)
(3,1.0986122886681098)
(7,1.9459101490553132)
(8,2.0794415416798357)
(1,0.0)
(6,1.791759469228055)
```

```
logtresult=FOREACH logdata GENERATE (l1), LOG10(l1);

dump logtresult;
```

```
(2,0.3010299956639812)
(3,0.47712125471966244)
(7,0.8450980400142568)
(8,0.9030899869919435)
(1,0.0)
(6,0.7781512503836436)
```

Scala: Spark

```
var a=10;
```

a: Int = 10

```
a;
```

res0: Int = 10

```
var b=20;
```

b: Int = 20

```
b;
```

res1: Int = 20

```
var s="abc";
```

s: String = abc

```
s;
```

res2: String = abc

```
var c=a+b;
```

c: Int = 30

```
c;
```

res3: Int = 30

```
val a=Array(1,2,3,4,5,6);
```

a: Array[Int] = Array(1, 2, 3, 4, 5, 6)

```
val b=sc.parallelize(a);
```

b: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:29

```
b;
```

res4: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:29

```
[cloudera@quickstart ~]$ cat number.txt
1
2
3
4
5
6
```

```
val n1=sc.textFile("file:/home/cloudera/number.txt");
```

n1: org.apache.spark.rdd.RDD[String] = file:/home/cloudera/number.txt MapPartitionsRDD[2] at textFile at <console>:27

```
n1;
```

res5: org.apache.spark.rdd.RDD[String] = file:/home/cloudera/number.txt MapPartitionsRDD[2] at textFile at <console>:27

```
n1.collect();
```

res6: Array[String] = Array(1, 2, 3, 4, 5, 6)

```
n1.count();
```

res7: Long = 6

```
n1.first();
```

res8: String = 1

```
n1.take(5);
```

res9: Array[String] = Array(1, 2, 3, 4, 5)

```
val y=sc.parallelize(List(10,20,30));
```

y: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[3] at parallelize at <console>:27

```
y.collect();
```

res10: Array[Int] = Array(10, 20, 30)

```
val mapt=y.map(x=>x+5);
```

mapt: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[4] at map at <console>:29

```
mapt.collect();
```

res11: Array[Int] = Array(15, 25, 35)

```
val mapm=y.map(x=>x-10);
```

mapm: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[5] at map at <console>:29

```
mapm.collect();
```

res12: Array[Int] = Array(0, 10, 20)

```
val mapmu=y.map(x=>x*2);
```

mapmu: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[6] at map at <console>:29

```
mapmu.collect();
```

res13: Array[Int] = Array(20, 40, 60)

```
val data=sc.parallelize(List(12,30,30,45,56,78,90,78,45));
```

data: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[7] at parallelize at <console>:27

```
val dt=data.distinct();
```

dt: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[10] at distinct at <console>:29

```
dt.collect();
```

res14: Array[Int] = Array(56, 12, 45, 30, 90, 78)

```
val data1=sc.parallelize(List(11,56,76,34,91,24));
```

data1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[11] at parallelize at <console>:27

```
val undata=data.union(data1);
```

undata: org.apache.spark.rdd.RDD[Int] = UnionRDD[12] at union at <console>:31

```
undata.collect();
```

res15: Array[Int] = Array(12, 30, 30, 45, 56, 78, 90, 78, 45, 11, 56, 76, 34, 91, 24)

```
val interdata=data.intersection(data1);
```

interdata: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[18] at intersection at <console>:31

```
interdata.collect();
```

res16: Array[Int] = Array(56)

```
val cartdata=data.cartesian(data1);
```

cartdata: org.apache.spark.rdd.RDD[(Int, Int)] = CartesianRDD[19] at cartesian at <console>:31

```
cartdata.collect();
```

res17: Array[(Int, Int)] = Array((12,11), (30,11), (12,56), (12,76), (30,56), (30,76), (12,34), (30,34), (12,91), (12,24), (30,91), (30,24), (30,11), (45,11), (30,56), (30,76), (45,56), (45,76), (30,34), (45,34), (30,91), (30,24), (45,91), (45,24), (56,11), (78,11), (56,56), (56,76), (78,56), (78,76), (56,34), (78,34),

(56,91), (56,24), (78,91), (78,24), (90,11), (78,11), (45,11), (90,56), (90,76), (78,56), (78,76), (45,56), (45,76), (90,34), (78,34), (45,34), (90,91), (90,24), (78,91), (78,24), (45,91), (45,24))

```
val sdata=sc.parallelize(Seq(("z",38),("a",45),("j",30),("c",23),("k",65)));
```

sdata: org.apache.spark.rdd.RDD[(String, Int)] = ParallelCollectionRDD[20] at parallelize at <console>:27

```
sdata.collect();
```

res18: Array[(String, Int)] = Array((z,38), (a,45), (j,30), (c,23), (k,65))

```
val sortdata=sdata.sortByKey();
```

sortdata: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[23] at sortByKey at <console>:29

```
sortdata.collect();
```

res19: Array[(String, Int)] = Array((a,45), (c,23), (j,30), (k,65), (z,38))

```
val sortdata=sdata.sortByKey(false);
```

sortdata: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[26] at sortByKey at <console>:29

```
sortdata.collect();
```

res20: Array[(String, Int)] = Array((z,38), (k,65), (j,30), (c,23), (a,45))

```
val grpdata=sdata.groupByKey();
```

grpdata: org.apache.spark.rdd.RDD[(String, Iterable[Int])] = ShuffledRDD[27] at groupByKey at <console>:29

```
grpdata.collect();
```

res21: Array[(String, Iterable[Int])] = Array((a,CompactBuffer(45)), (z,CompactBuffer(38)), (j,CompactBuffer(30)), (k,CompactBuffer(65)), (c,CompactBuffer(23)))

```
val gdata=sc.parallelize(Seq(("z",38),("a",45),("j",30),("c",23),("k",65),("a",54),("j",20),("c",30)));
```

gdata: org.apache.spark.rdd.RDD[(String, Int)] = ParallelCollectionRDD[28] at parallelize at <console>:27

```
val newgrp=gdata.groupByKey();
```

newgrp: org.apache.spark.rdd.RDD[(String, Iterable[Int])] = ShuffledRDD[29] at groupByKey at <console>:29

```
newgrp.collect();
```

res22: Array[(String, Iterable[Int])] = Array((a,CompactBuffer(45, 54)), (z,CompactBuffer(38)), (j,CompactBuffer(30, 20)), (k,CompactBuffer(65)), (c,CompactBuffer(23, 30)))

Input File:

```
[cloudera@quickstart ~]$ cat name.txt
Sunil Sumeet Sunio Piyush Lekha Sunil Lekha Sunio Sumeet Sunil
```

```
val cfile=sc.textFile("file:/home/cloudera/name.txt");
```

cfile: org.apache.spark.rdd.RDD[String] = file:/home/cloudera/name.txt MapPartitionsRDD[1] at textFile at <console>:27

```
cfile.collect();
```

res0: Array[String] = Array(Sunil Sumeet Sunio Piyush Lekha)

```
var charfile=cfile.flatMap(line=>line.split(""));
```

charfile: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <console>:29

```
charfile.collect();
```

res1: Array[String] = Array("", S, u, n, i, l, " ", S, u, m, e, e, t, " ", S, u, n, i, o, " ", P, i, y, u, s, h, " ", L, e, k, h, a)

```
val fchar=charfile.map(fchar=>(fchar,1));
```

fchar: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[3] at map at <console>:31

```
fchar.collect();
```

res2: Array[(String, Int)] = Array(("",1), (S,1), (u,1), (n,1), (i,1), (l,1), (" ",1), (S,1), (u,1), (m,1), (e,1), (e,1), (t,1), (" ",1), (S,1), (u,1), (n,1), (i,1), (o,1), (" ",1), (P,1), (i,1), (y,1), (u,1), (s,1), (h,1), (" ",1), (L,1), (e,1), (k,1), (h,1), (a,1))

```
val finalchar=fchar.reduceByKey(_+_);
```

finalchar: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:33

```
finalchar.collect();
```

res3: Array[(String, Int)] = Array((L,1), (P,1), (t,1), (h,2), (" ",4), ("",1), (n,2), (l,1), (s,1), (e,3), (a,1), (i,3), (k,1), (y,1), (u,4), (o,1), (S,3), (m,1))

```
val namefile=sc.textFile("file:/home/cloudera/name.txt");
```

namefile: org.apache.spark.rdd.RDD[String] = file:/home/cloudera/name.txt MapPartitionsRDD[1] at textFile at <console>:27

```
namefile.collect();
```

res0: Array[String] = Array(Sunil Sumeet Sunio Piyush Lekha Sunil Lekha Sunio Sumeet Sunil)

```
val splitdata=namefile.flatMap(line=>line.split(" "));
```

splitdata: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <console>:29

```
splitdata.collect();
```

res1: Array[String] = Array(Sunil, Sumeet, Sunio, Piyush, Lekha, Sunil, Lekha, Sunio, Sumeet, Sunil)

```
val cword=splitdata.map(cword=>(cword,1));
```

cword: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[3] at map at <console>:31

```
cword.collect();
```

res2: Array[(String, Int)] = Array((Sunil,1), (Sumeet,1), (Sunio,1), (Piyush,1), (Lekha,1), (Sunil,1), (Lekha,1), (Sunio,1), (Sumeet,1), (Sunil,1))

```
val result=cword.reduceByKey(_+_);
```

result: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:33

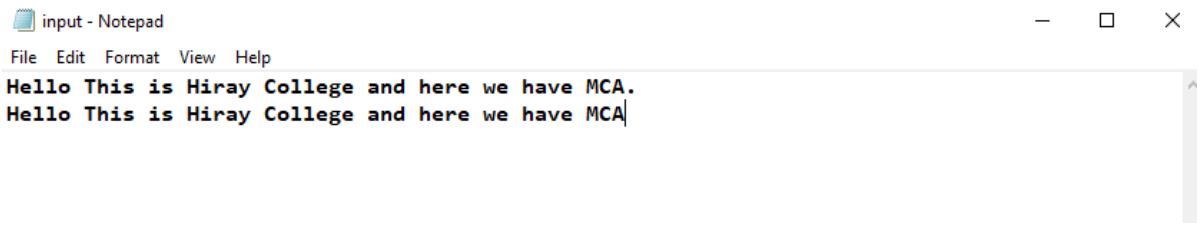
```
result.collect();
```

res3: Array[(String, Int)] = Array((Sunio,2), (Piyush,1), (Sunil,3), (Sumeet,2), (Lekha,2))

Spark Python Mode

Word Count

Input File:




```
import findspark

findspark.init()

import pyspark

findspark.find()
```

```
'C:\\apps\\spark-3.0.3-bin-hadoop2.7'
```

```
from pyspark import SparkContext, SparkConf

sc=SparkContext("local","Counting Words")

nwords=sc.textFile("D:\MCA-SY\BDAnV\Spark-word-count\input.txt").flatMap(lambda line:
line.split(" "))

wordCo=nwords.map(lambda nwords:(nwords,1))

wordresult=wordCo.reduceByKey(lambda wordsCount, b: (wordsCount + b))

wordresult.collect()
```

```
[('Hello', 2),
 ('This', 2),
 ('is', 2),
 ('Hiray', 2),
 ('College', 2),
 ('and', 2),
 ('here', 2),
 ('we', 2),
 ('have', 2),
 ('MCA.', 1),
 ('MCA', 1)]
```

KMean

```
import findspark

findspark.init()

findspark.find()
```

```
'C:\\apps\\spark-3.0.3-bin-hadoop2.7'
```

```
import pyspark

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("K Means Cluster Model").getOrCreate()

spark
```

SparkSession - in-memory

SparkContext

Spark UI

Version	`v3.0.3`
Master	`local[*]`
AppName	`K Means Cluster Model`

```
from pyspark.ml.clustering import KMeans

from pyspark.ml.feature import VectorAssembler, VectorIndexer

from pyspark.ml.feature import StandardScaler

from pyspark.ml.evaluation import ClusteringEvaluator

data=spark.read.csv("seeds_dataset.csv", header=True, inferSchema=True)

type(data)
```

```
pyspark.sql.dataframe.DataFrame
```

```
data.head()
```

```
Row(area=15.26, perimeter=14.84, compactness=0.871, length_of_kernel=5.763, width_of_kernel=3.312, asy
mmetry_coefficient=2.221, length_of_groove=5.22)
```

```
data.printSchema()
```

```
root
 |-- area: double (nullable = true)
 |-- perimeter: double (nullable = true)
 |-- compactness: double (nullable = true)
 |-- length_of_kernel: double (nullable = true)
 |-- width_of_kernel: double (nullable = true)
 |-- asymmetry_coefficient: double (nullable = true)
 |-- length_of_groove: double (nullable = true)
```

```
data.describe().show()
```

```
+-------+------------------+------------------+--------------------+------------------+------------------+
|summary|              area|         perimeter|         compactness|   length_of_kernel|    width_of_kernel|asymmetry_coefficient|  length_of_groove|
+-------+------------------+------------------+--------------------+------------------+------------------+
|  count|               210|               210|                 210|               210|               210|
|   mean|14.847523809523816|14.559285714285718|  0.8709985714285714|  5.628533333333335|  3.2586047619047621|   3.7001999999999997| 5.408071428571429|
| stddev|2.9096994306873647|1.3059587265640225|0.023629416583846364|0.44306347772645016|0.37771444490658673|    1.503558970254739|0.4914804991024053|
|    min|             10.59|             12.41|              0.8081|              4.899|              2.63|                0.765|             4.519|
|    max|             21.18|             17.25|              0.9183|              6.675|             4.033|                8.456|              6.55|
+-------+------------------+------------------+--------------------+------------------+------------------+
```

```
#Data Processing

assembler=VectorAssembler(inputCols=data.columns,outputCol='features')

final_data=assembler.transform(data)

#Scaler

scaler=StandardScaler(inputCol='features',outputCol='scaledfeatures')

final_data=scaler.fit(final_data).transform(final_data)

final_data.head()
```

```
Row(area=15.26, perimeter=14.84, compactness=0.871, length_of_kernel=5.763, width_of_kernel=3.312, asymmetry_coefficient=2.221, length_of_groove=5.22, features=DenseVector([15.26, 14.84, 0.871, 5.763, 3.312, 2.221, 5.22]), scaledfeatures=DenseVector([5.2445, 11.3633, 36.8608, 13.0072, 8.7685, 1.4772, 10.621]))
```

```
# train and test

train_data,test_data=final_data.randomSplit([0.7,0.3])

train_data.show(2)
```

```
+-----+---------+-----------+---------------+--------------+--------------------+----------------+-
------------------+--------------------+
| area|perimeter|compactness|length_of_kernel|width_of_kernel|asymmetry_coefficient|length_of_groove|
features|      scaledfeatures|
+-----+---------+-----------+---------------+--------------+--------------------+----------------+-
------------------+--------------------+
|10.74|    12.73|     0.8329|          5.145|         2.642|               4.702|           4.963|
[10.74,12.73,0.83...|[3.69110289768413...|
| 10.8|    12.57|      0.859|          4.981|         2.821|               4.773|           5.063|
[10.8,12.57,0.859...|[3.71172358426337...|
+-----+---------+-----------+---------------+--------------+--------------------+----------------+-
------------------+--------------------+
only showing top 2 rows
```

```
# Build and Evaluate

# K-Means

classifier=KMeans(k=2,featuresCol='scaledfeatures')

model=classifier.fit(train_data)

# Predictions

predictions=model.transform(test_data)

#Evaluate clustring by computing Silhoutte score

score=ClusteringEvaluator().evaluate(predictions)

print(score)
```

```
0.7227070129257039
```

```
# Printing clusters centers

centers=model.clusterCenters()

print("Cluster Centers:")

for center in centers:

  print(center)
```

```
Cluster Centers:
[ 4.46157311 10.50131634 36.57279791 12.05836797  8.03273921  2.5990782
 10.33786112]
[ 6.26387723 12.32253895 37.3628276  13.88054747  9.6847573   2.35261202
 12.20660657]
```

```
predictions.show(100)
```

```
+-----+---------+-----------+---------------+--------------+--------------------+---------------+-
------------------+--------------------+----------+
| area|perimeter|compactness|length_of_kernel|width_of_kernel|asymmetry_coefficient|length_of_groove|
features|      scaledfeatures|prediction|
+-----+---------+-----------+---------------+--------------+--------------------+---------------+-
------------------+--------------------+----------+
|10.59|    12.41|     0.8648|          4.899|         2.787|               4.975|          4.794|
[10.59,12.41,0.86...|[3.63955118123602...|         0|
|10.79|    12.93|     0.8107|          5.317|         2.648|               5.462|          5.194|
[10.79,12.93,0.81...|[3.70828680316683...|         0|
|11.21|    13.13|     0.8167|          5.279|         2.687|               6.169|          5.275|
[11.21,13.13,0.81...|[3.85263160922151...|         0|
|11.23|    12.63|      0.884|          4.902|         2.879|               2.269|          4.703|
[11.23,12.63,0.88...|[3.85950517141459...|         0|
|11.34|    12.87|     0.8596|          5.053|         2.849|               3.347|          5.003|
[11.34,12.87,0.85...|[3.89730976347654...|         0|
|11.36|    13.05|     0.8382|          5.175|         2.755|               4.048|          5.263|
[11.36,13.05,0.83...|[3.90418332566962...|         0|
| 11.4|    13.08|     0.8375|          5.136|         2.763|               5.588|          5.089|
[11.4,13.08,0.837...|[3.91793045005578...|         0|
|11.48|    13.05|     0.8473|           5.18|         2.758|               5.876|          5.002|
[11.48,13.05,0.84...|[3.94542469882810...|         0|
|11.56|    13.31|     0.8198|          5.363|         2.683|               4.062|          5.182|
[11.56,13.31,0.81...|[3.97291894760042...|         0|
|11.75|    13.52|     0.8082|          5.444|         2.678|               4.378|           5.31|
[11.75,13.52,0.80...|[4.03821778843468...|         0|
```

Page Ranking

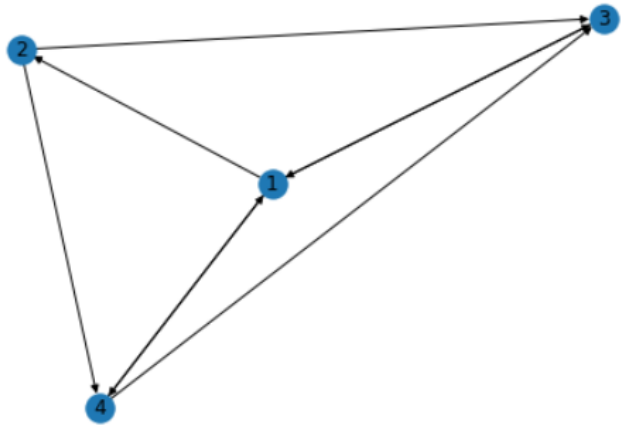
```
import numpy as np

import matplotlib.pyplot as plt

import networkx as nx

# To create an empty graph or initializing a graph

G = nx.DiGraph()

# creating pages as nodes

pages = ["1","2","3","4"]

G.add_nodes_from(pages)

# print nodes of graph

print(G.nodes())
```

```
['1', '2', '3', '4']
```

```
G.add_edges_from([('1','2'),('1','3'),('1','4'),('2','3'),('2','4'),('3','1'),('4','1'),('4','3')])

# printing outward links for each node   ['Page 1 = 3']  ['Page 2 = 2']

print("Number of outward links for each node:")

for page in pages:

   print(["Page %s = %s" %(page,str(len(G.out_edges(page))))])
```

```
Number of outward links for each node:
['Page 1 = 3']
['Page 2 = 2']
['Page 3 = 1']
['Page 4 = 2']
```

```
# we will print graph using matplotlib

nx.draw(G, with_labels=True)

plt.show()
```



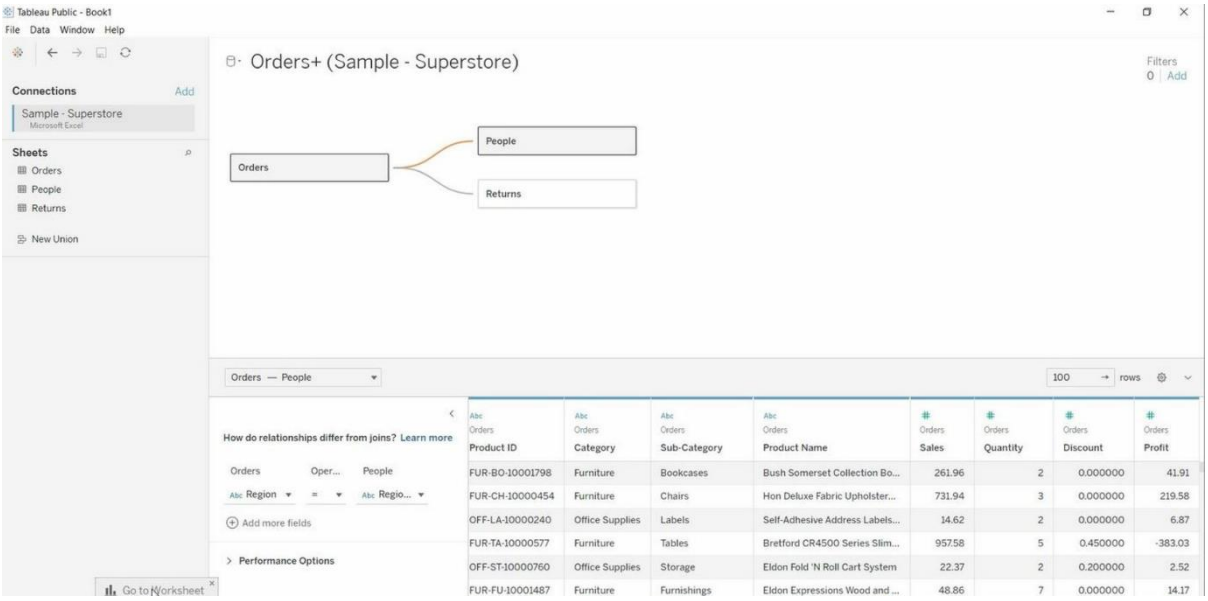

```
linkmatrix = np.matrix([[0,0,1,1/2],
                        [1/3,0,0,0],
                        [1/3,1/2,0,1/2],
                        [1/3,1/2,0,0]])
def findPageRank(linkmatrix,pages):
    eigval,eigvector = np.linalg.eig(linkmatrix)
    final_eigval = np.abs(eigval).max()
    PageRank = np.where(eigval==final_eigval)
    print("The most important page is %s" %(str(pages[PageRank[0][0]])))
findPageRank(linkmatrix,pages)
```

```
The most important page is 1
```

Tableau

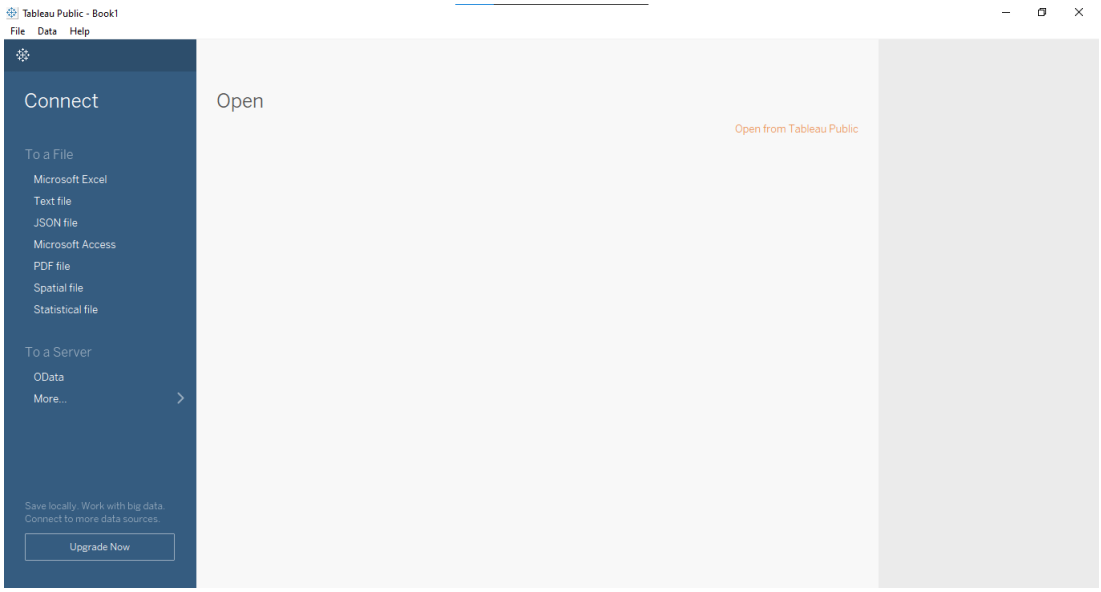
- Open Tableau public and connect to the data source.

JOINS:



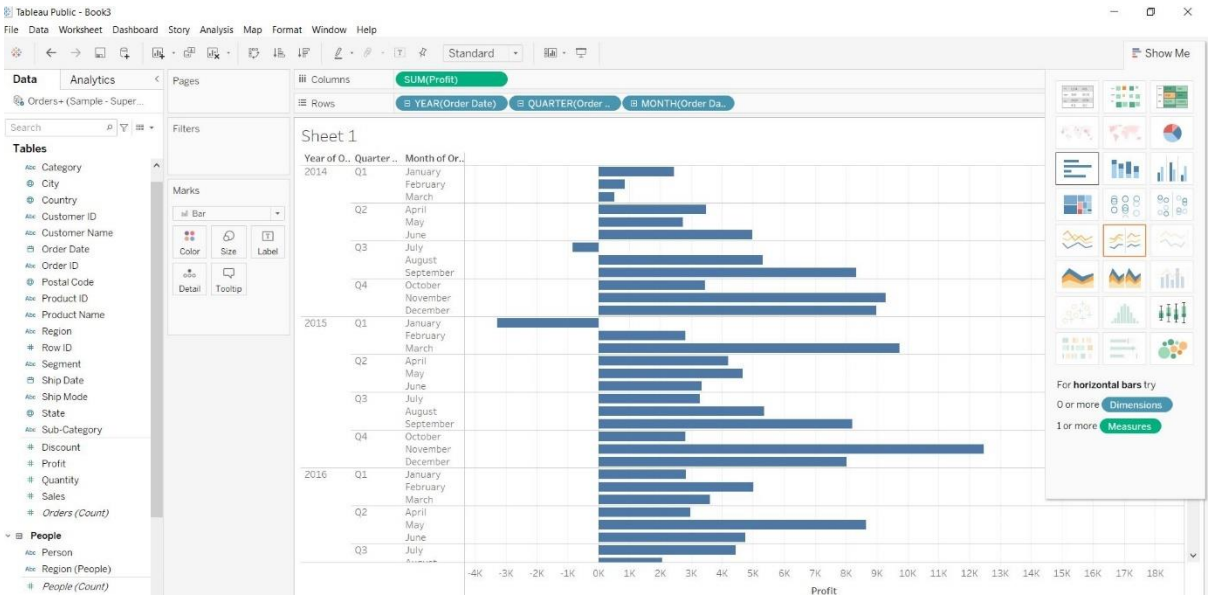
Bar Graph:

- Go to the new worksheet.

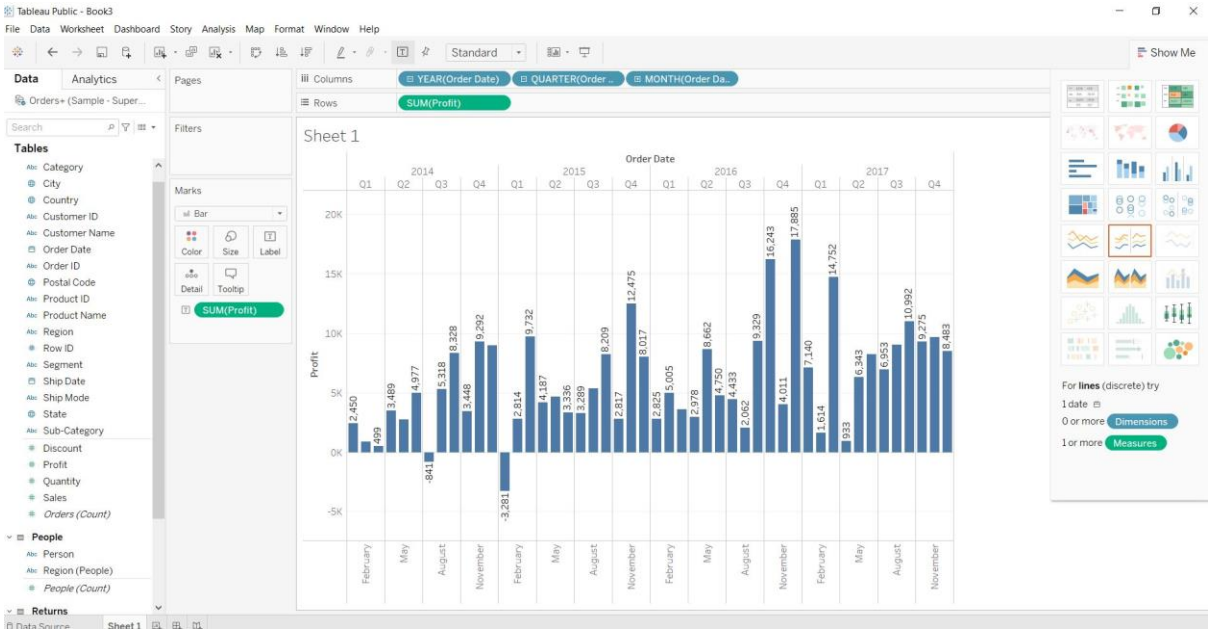


- Under Dimension select Order Date
- Go to the Show Me (user can see chart) ->select chart.

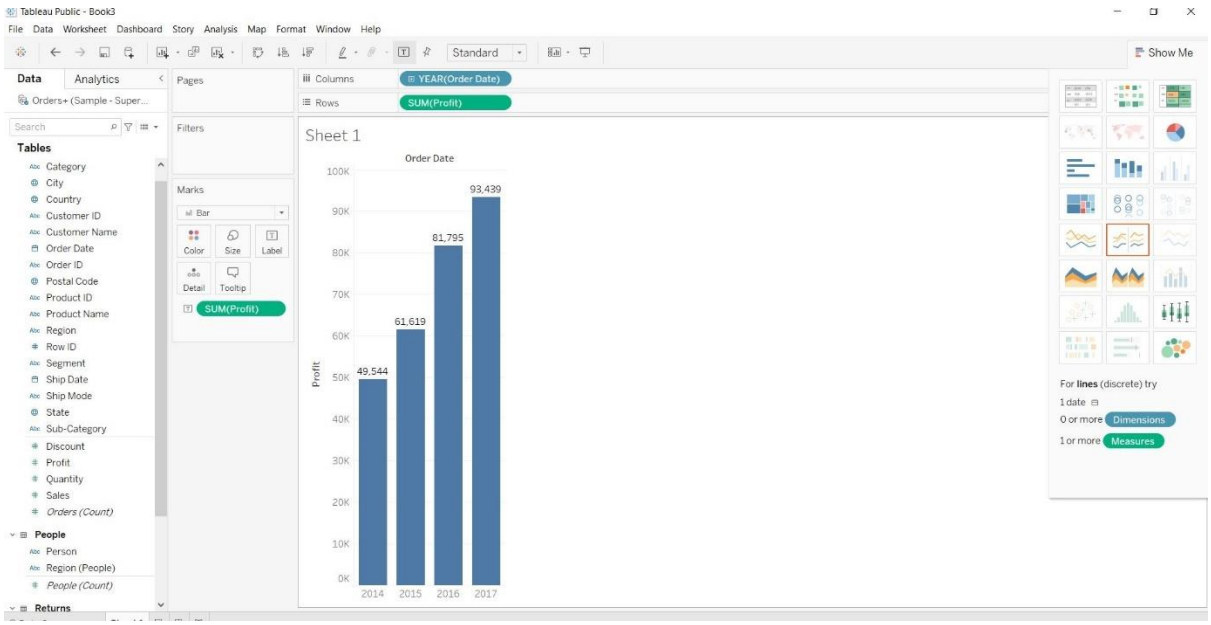
- Go to the Rows > select Year (Order Date) -> drop-down -> select Month.



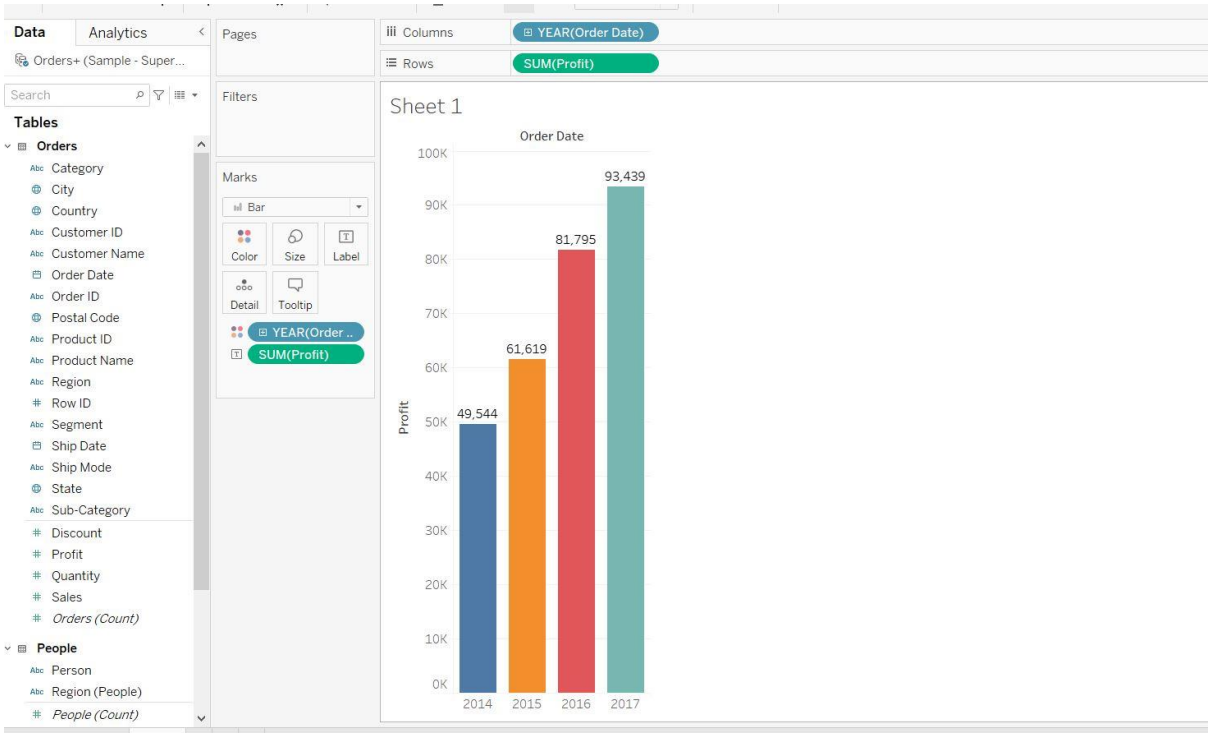
Under Dimension select SUM(Profit) ->Drag to Label

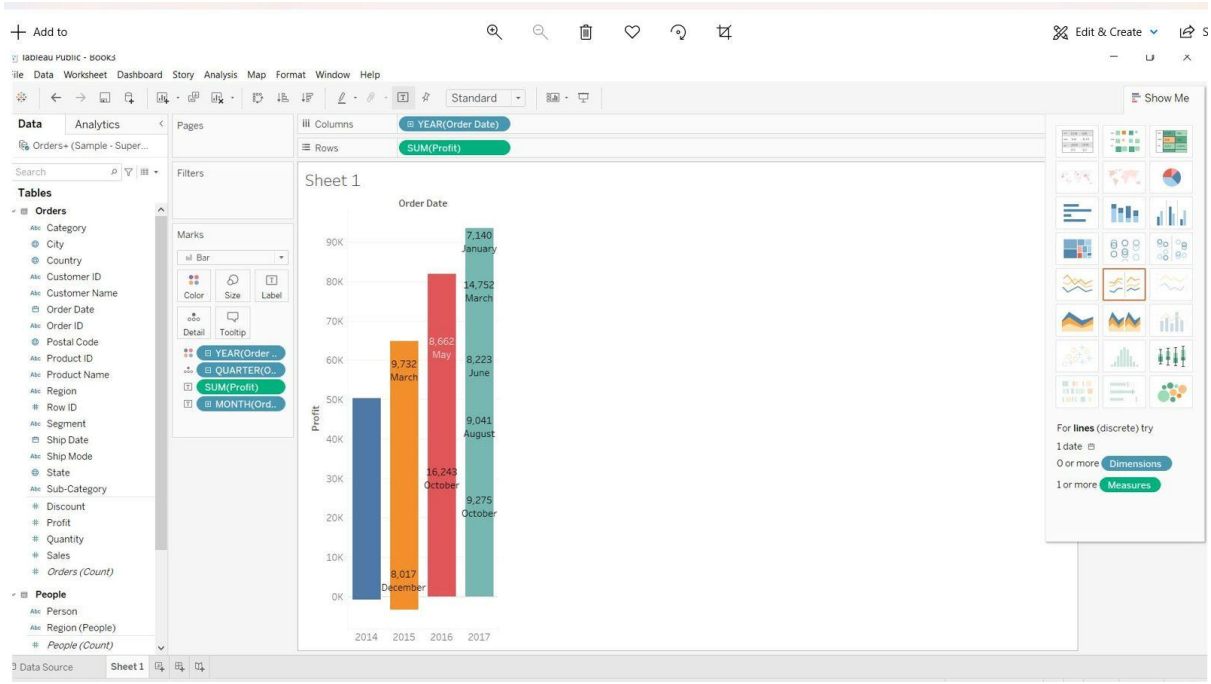


Remove Month & Quarter



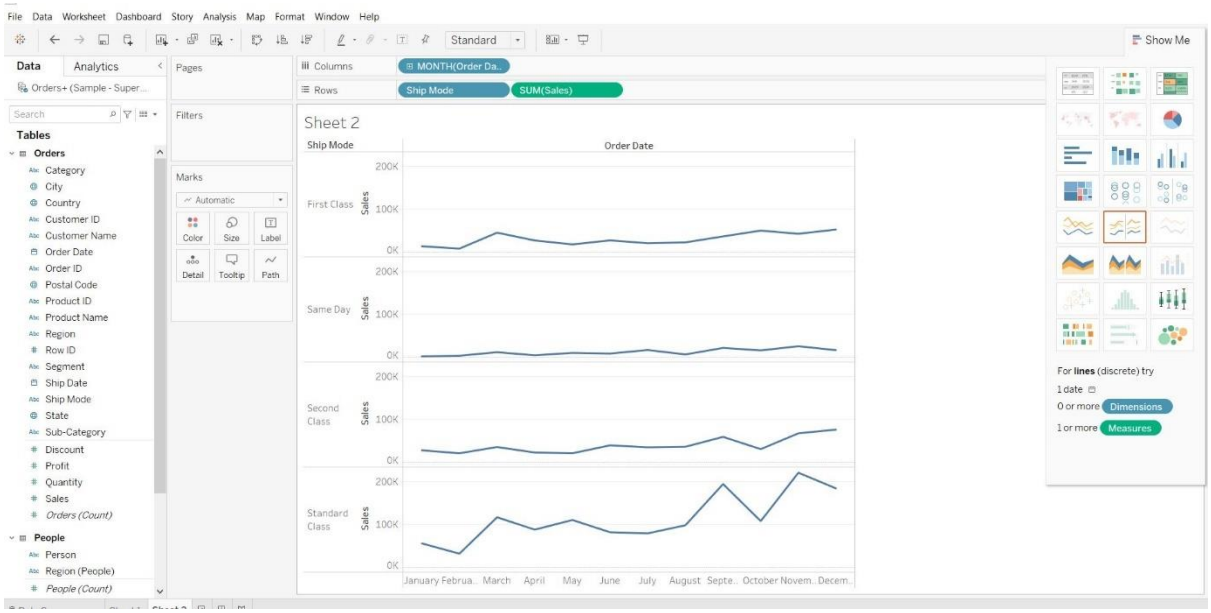
Under Dimension select Year (Order Date) ->Drag to Colour





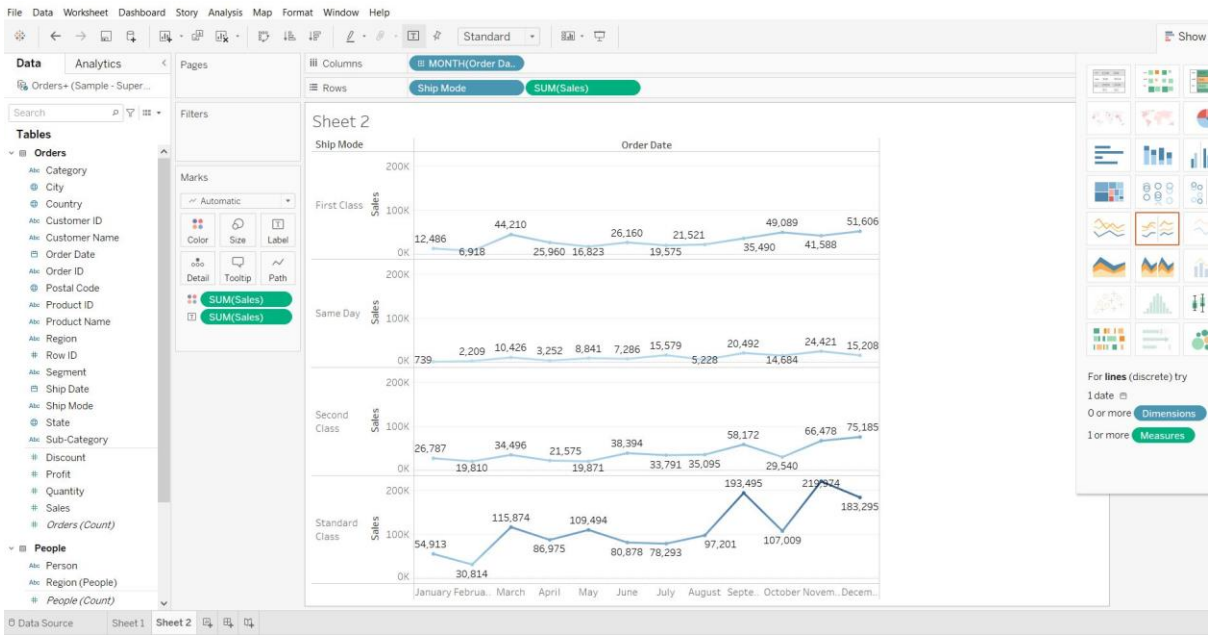
2. Multilevel Chart:

- Open Tableau public and connect to the data source.
- Go to the new worksheet.
- Drag **Ship Mode** & **Sales** into rows and **Month** into columns.

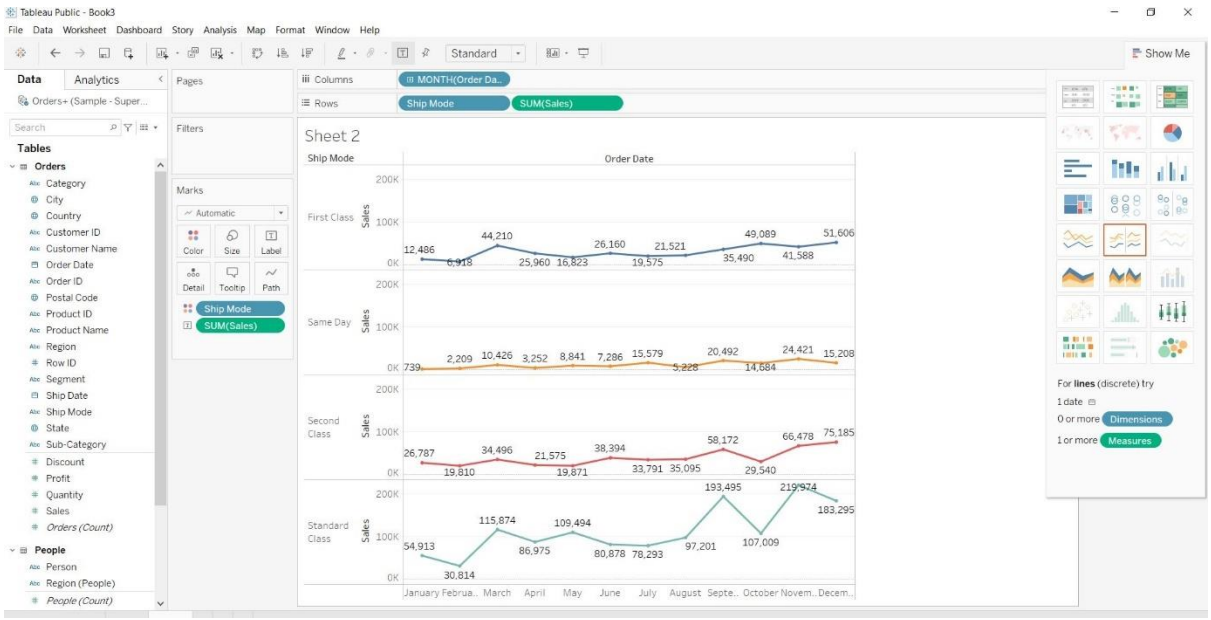


Drag "**Sales**" into Label (Marks card).

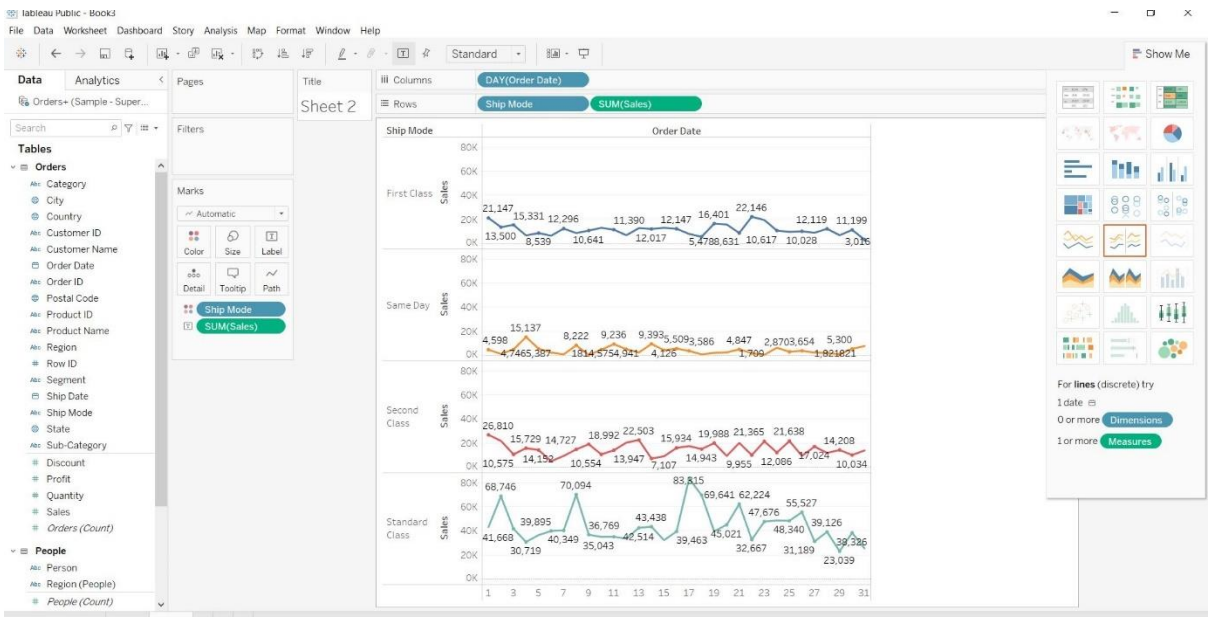
Drag "**Ship Mode**" into Colour (Marks card).



Click on “**show me**”, select chart.



Change Month into Day in Columns



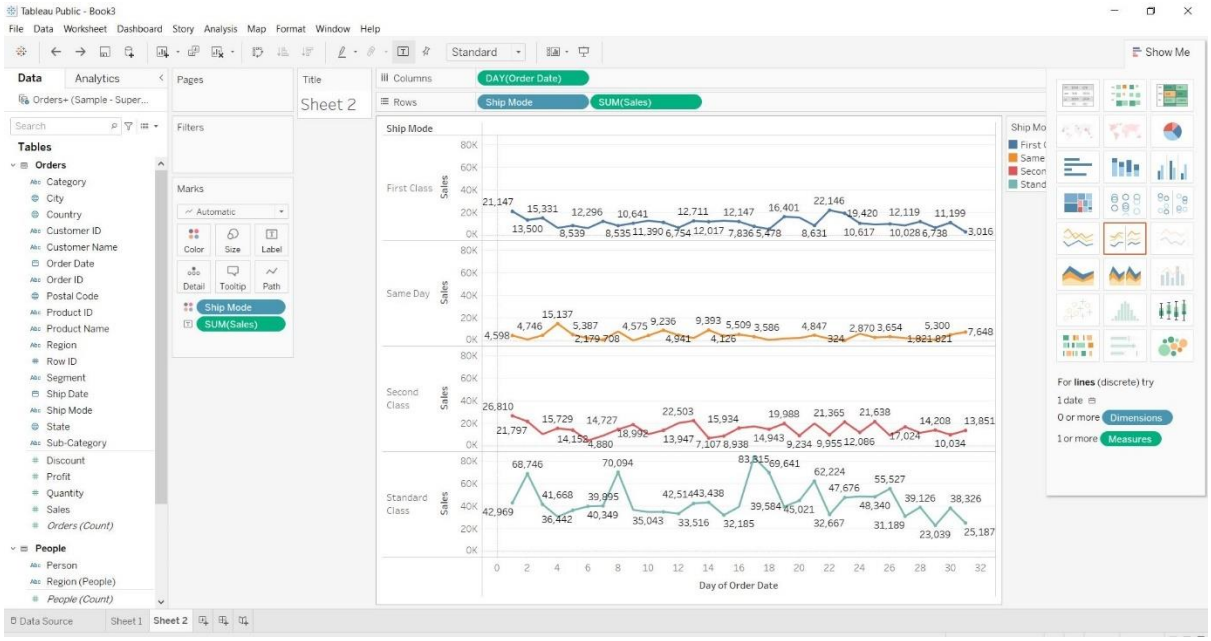


Tableau Public - Book3
File Data Worksheet Dashboard Story Analysis Map Format Window Help
Standard
Show Me
Data
Analytics
Orders+ (Sample - Super...
Search
Tables
Orders
Category
City
Country
Customer ID
Customer Name
Order Date
Order ID
Postal Code
Product ID
Product Name
Region
Row ID
Segment
Ship Date
Ship Mode
State
Sub-Category
Discount
Profit
Quantity
Sales
Orders (Count)
People
Person
Region (People)
People (Count)
Pages
Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Path
Ship Mode
SUM(Sales)
Title
Sheet 2
Columns
DAY(Order Date)
Rows
Ship Mode
SUM(Sales)
Ship Mode
First Class
Same Day
Second Class
Standard Class
Sales
Day of Order Date
For lines (discrete) try
1 date
0 or more
Dimensions
1 or more
Measures
Data Source
Sheet 1
Sheet 2

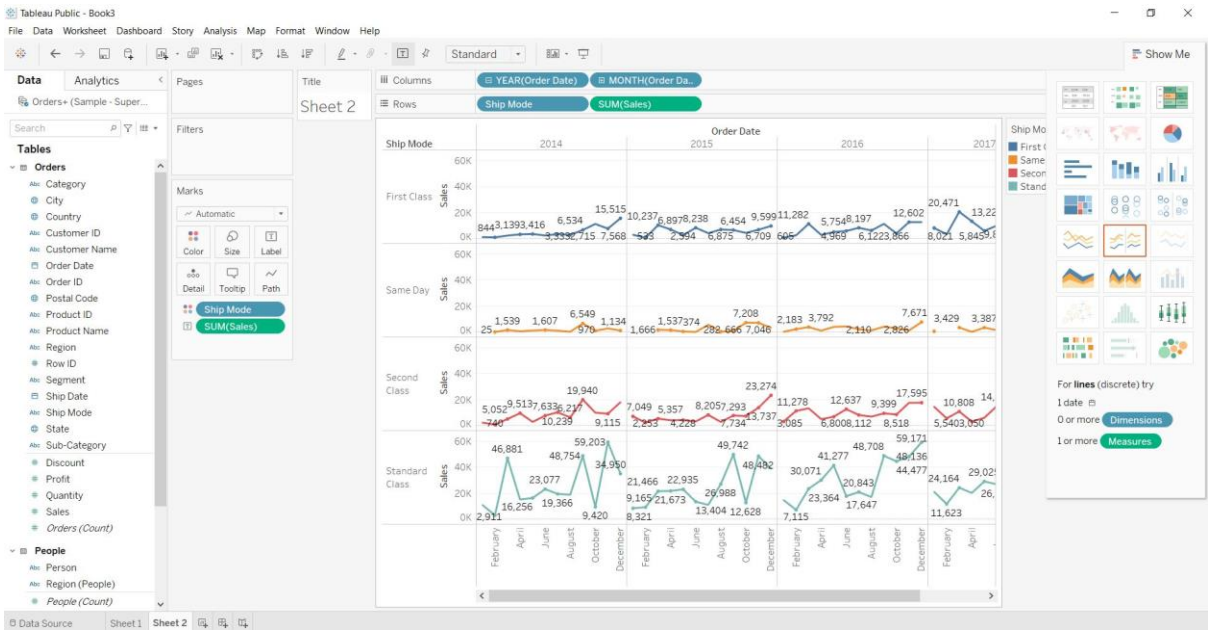
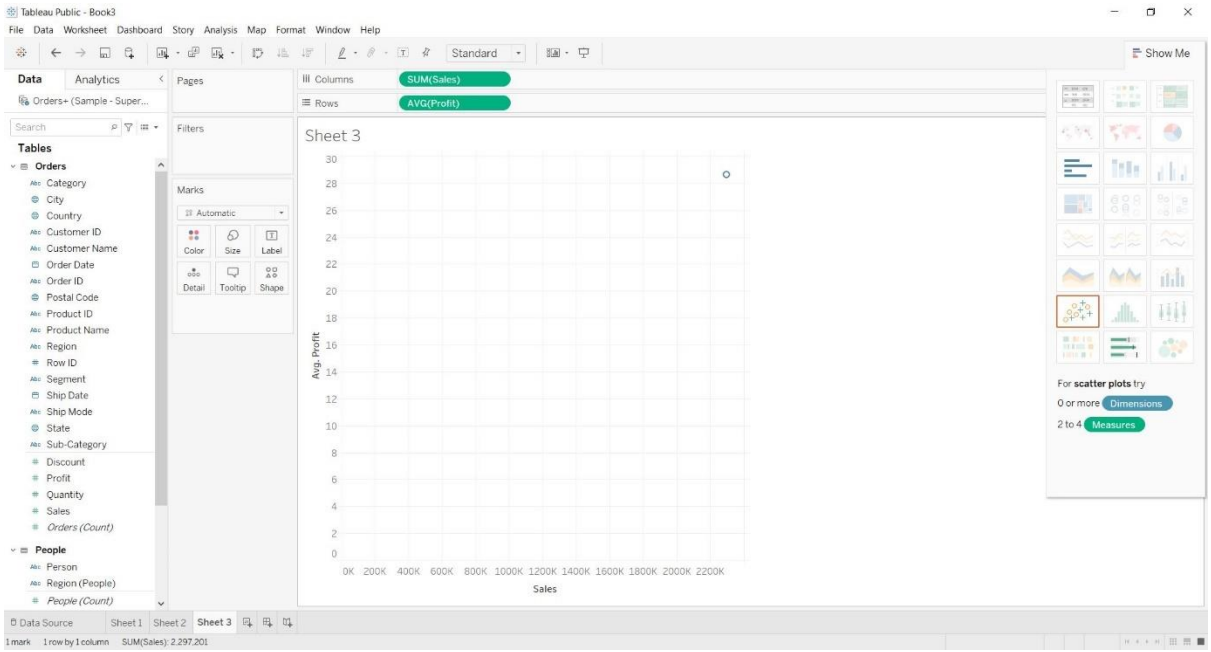


Tableau Public - Book3
File Data Worksheet Dashboard Story Analysis Map Format Window Help
Standard
Show Me
Data
Analytics
Orders+ (Sample - Super...
Search
Tables
Orders
Category
City
Country
Customer ID
Customer Name
Order Date
Order ID
Postal Code
Product ID
Product Name
Region
Row ID
Segment
Ship Date
Ship Mode
State
Sub-Category
Discount
Profit
Quantity
Sales
Orders (Count)
People
Person
Region (People)
People (Count)
Pages
Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Path
Ship Mode
SUM(Sales)
Title
Sheet 2
Columns
YEAR(Order Date)
MONTH(Order Da...
Rows
Ship Mode
SUM(Sales)
Order Date
2014
2015
2016
2017
Ship Mode
First Class
Same Day
Second Class
Standard Class
Sales
February
April
June
August
October
December
For lines (discrete) try
1 date
0 or more
Dimensions
1 or more
Measures
Data Source
Sheet 1
Sheet 2

3. Mark Plot Chart:

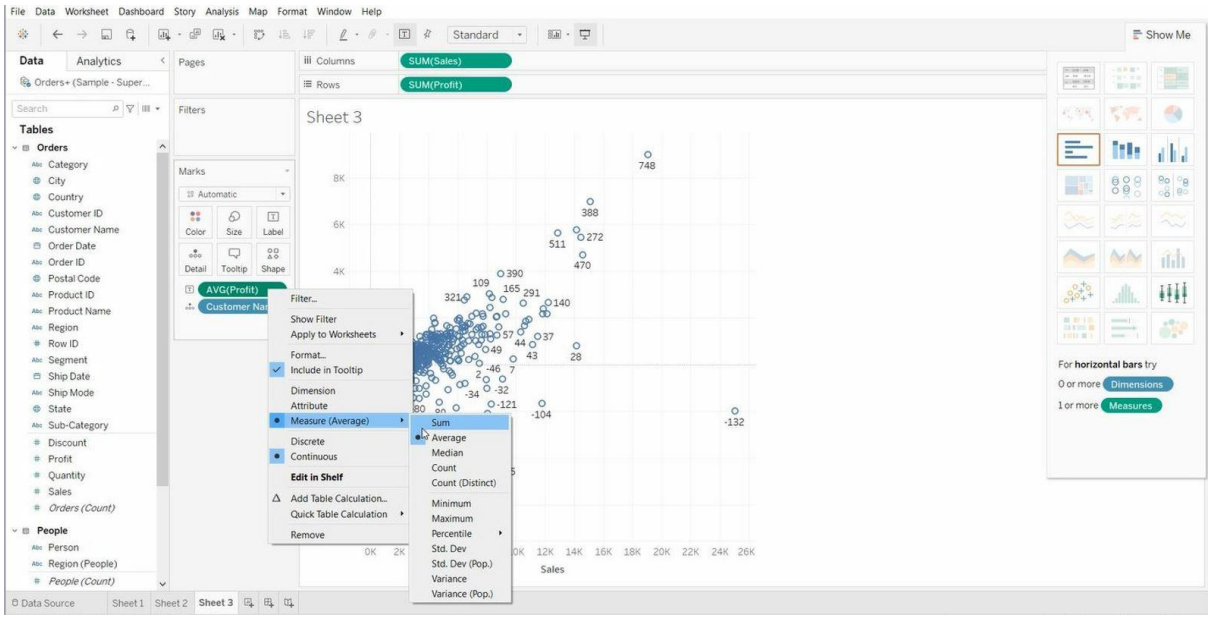
- Open Tableau public and connect to the data source.
- Go to the new worksheet.
- Drag Profit into rows and Sales into columns. This will create a scatter plot by default.



Drag "**Customer Name**" into Detail (Marks card).

Drag “**AVG(Profit)**” into Label (Marks card).

Right Click on “**AVG(Profit)” > Measure (Average) > Sum**

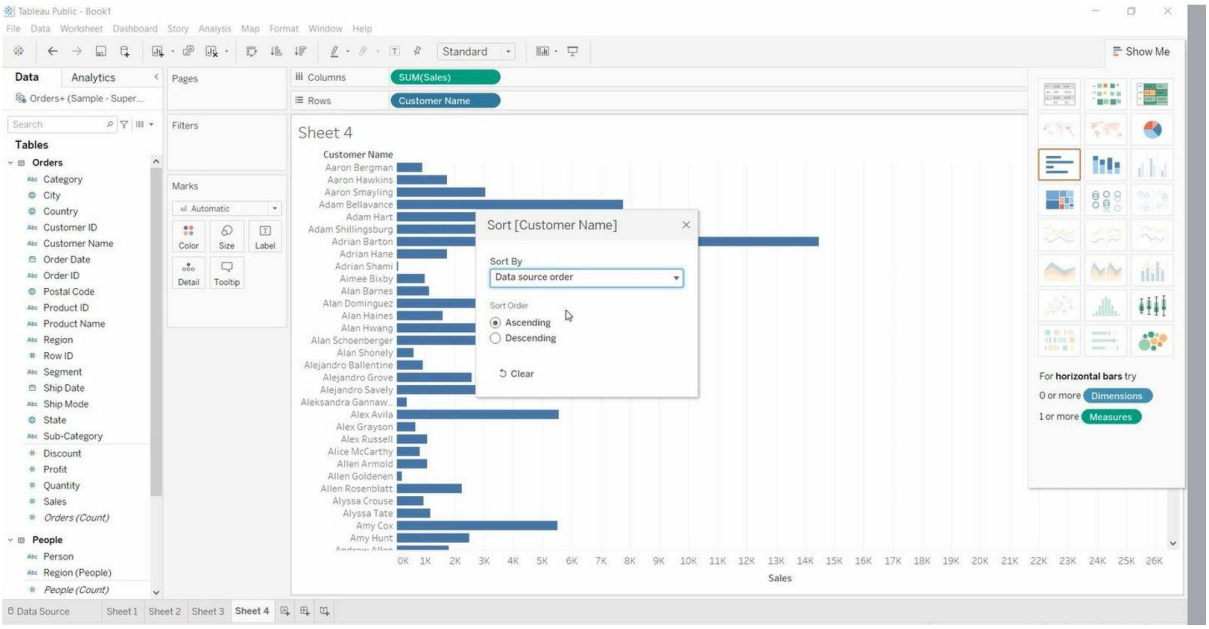


Drag “**Segment**” into Colour (Marks card).

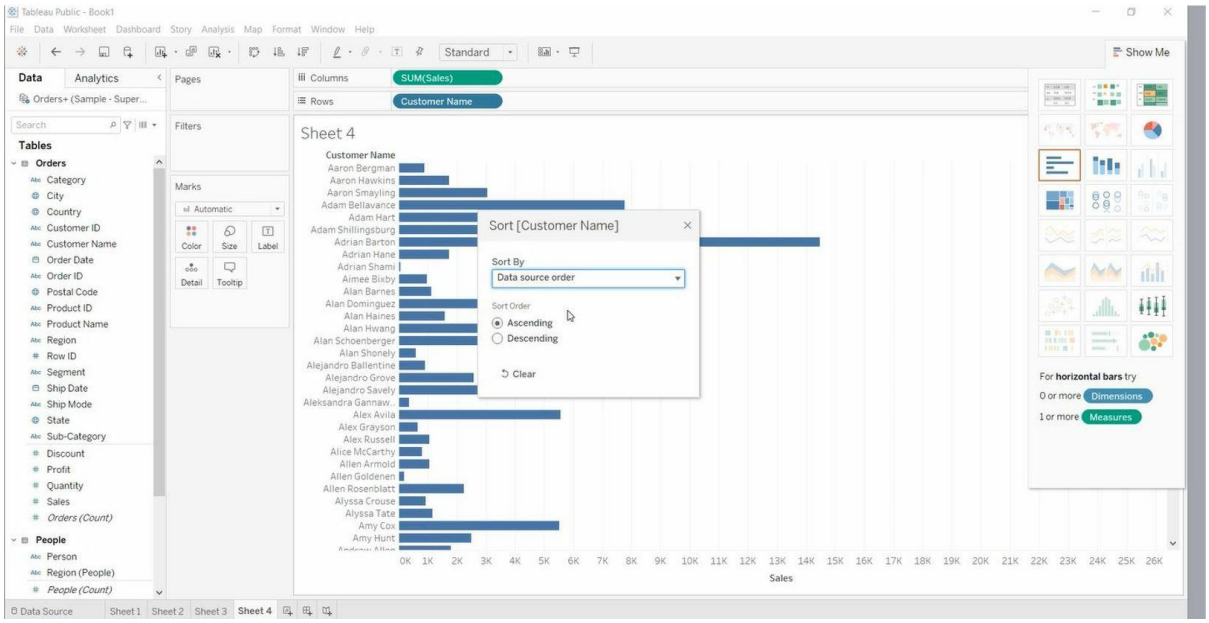
Drag "**Category**" into Shape (Marks card).

Working on filters:

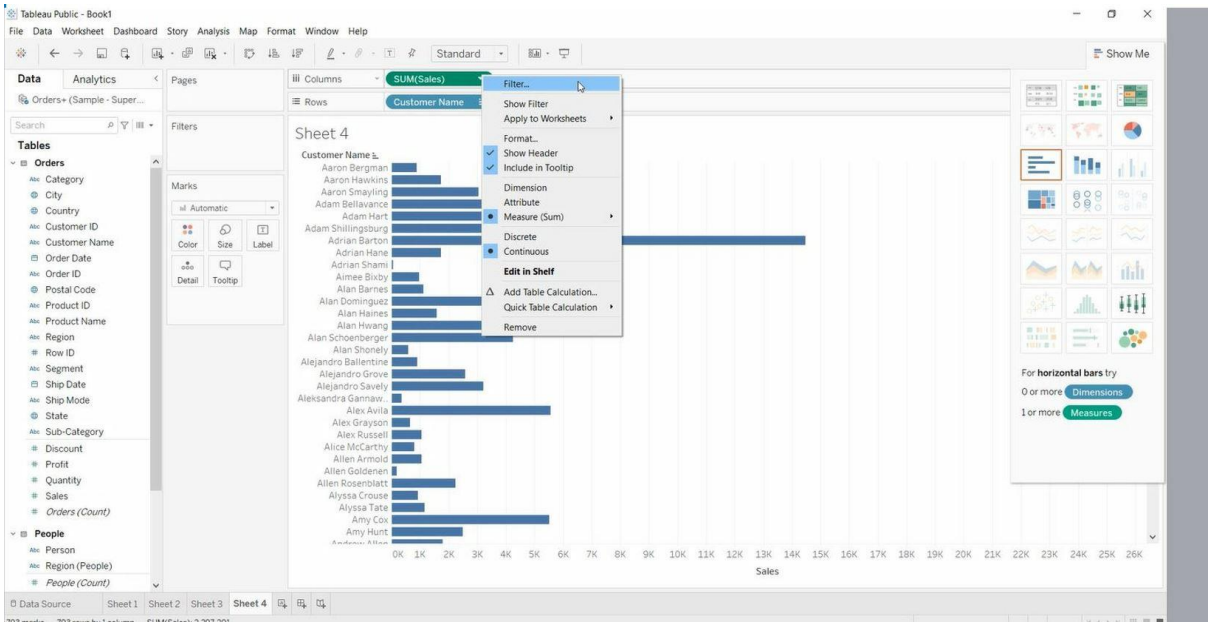
- Open Tableau public and connect to the data source.
- Go to the new worksheet.
- Drag **Customer Name** into rows and **Sales** into columns.



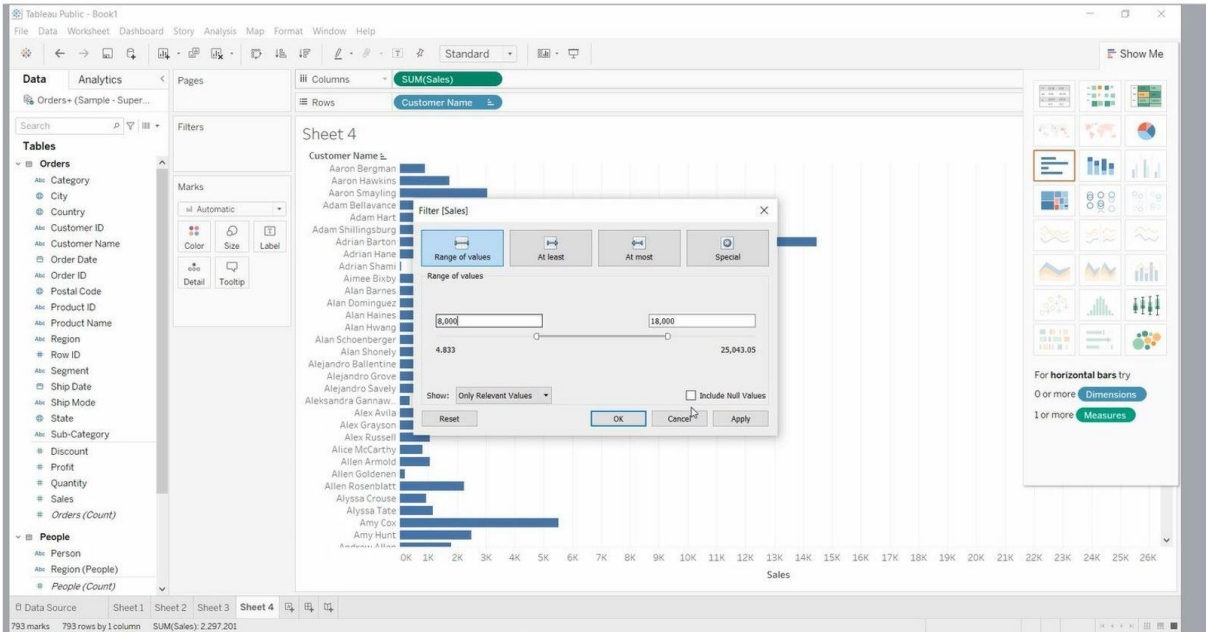
Right Click on **Customer Name** in Graph > Select **Ascending**



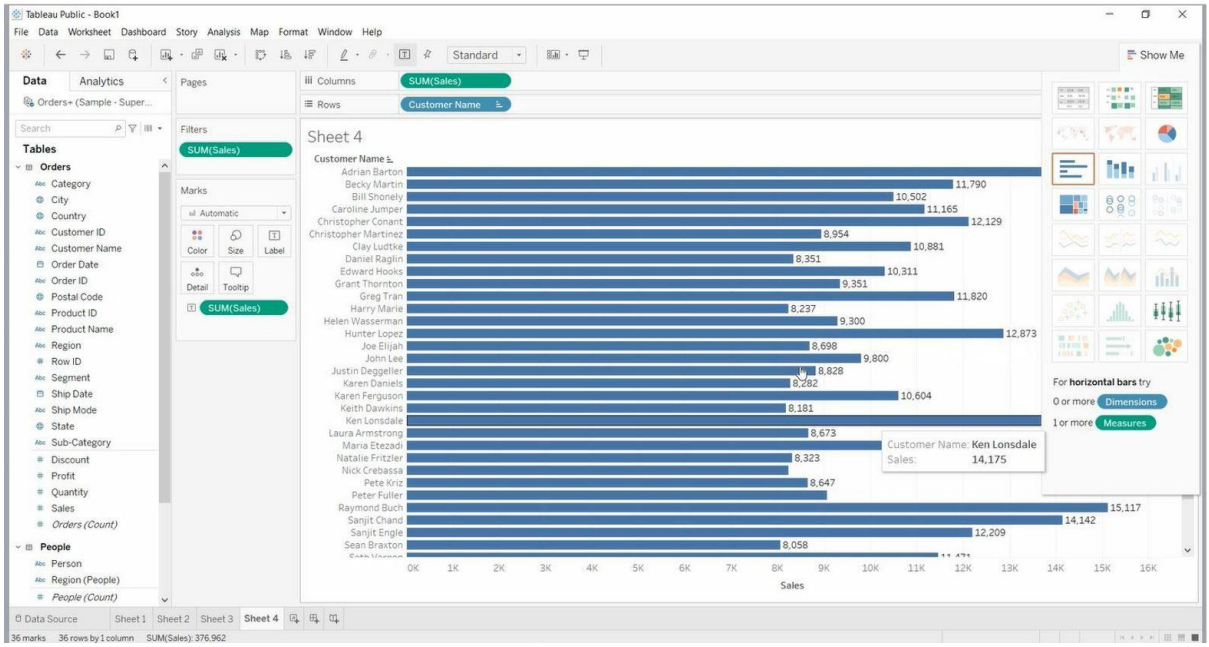
Right Click on **Sales** in Columns > Click on **Filter**



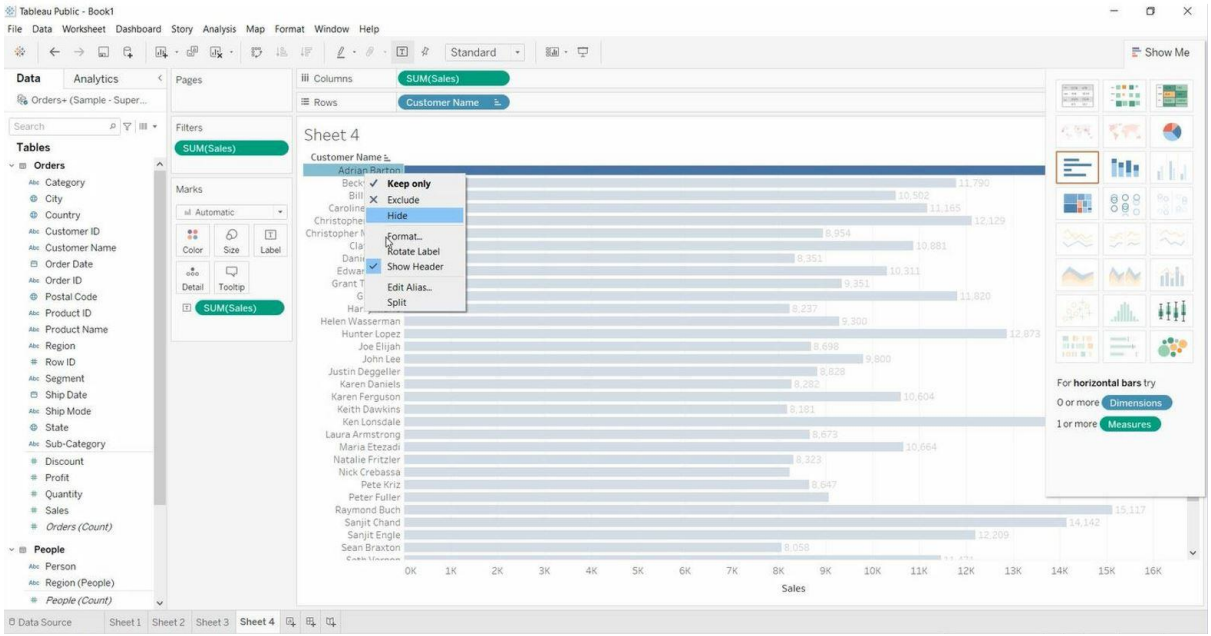
Select **Range of Values,** Set Range > Click on OK



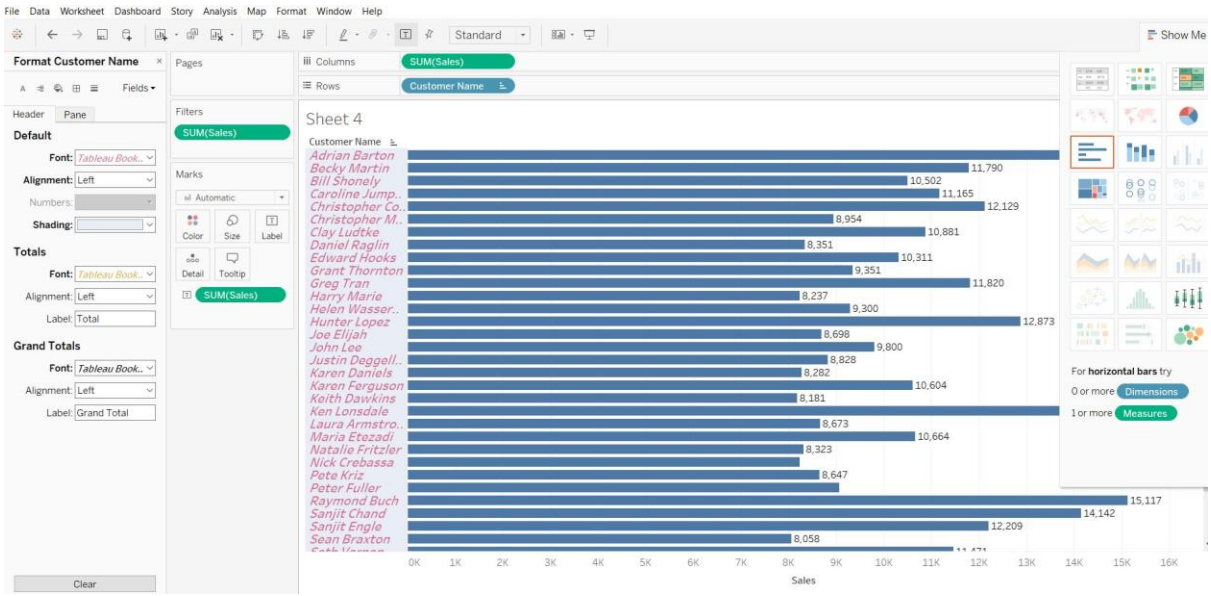
Drag "**Sales**" into Label (Marks card).



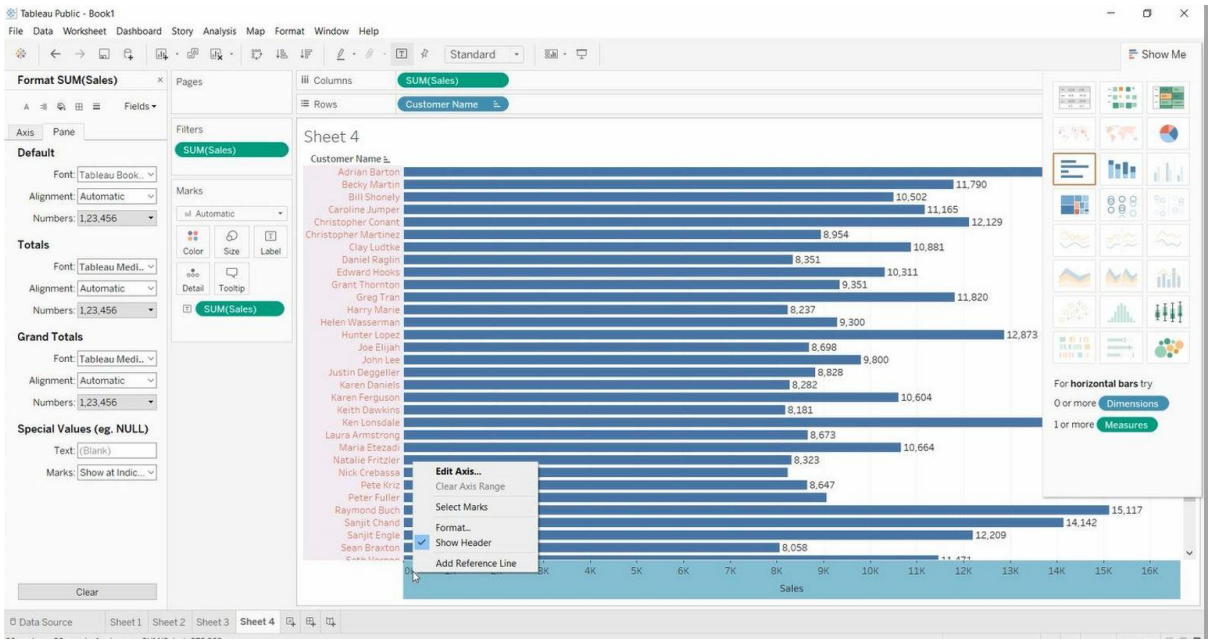
Right Click on any customer name > Format

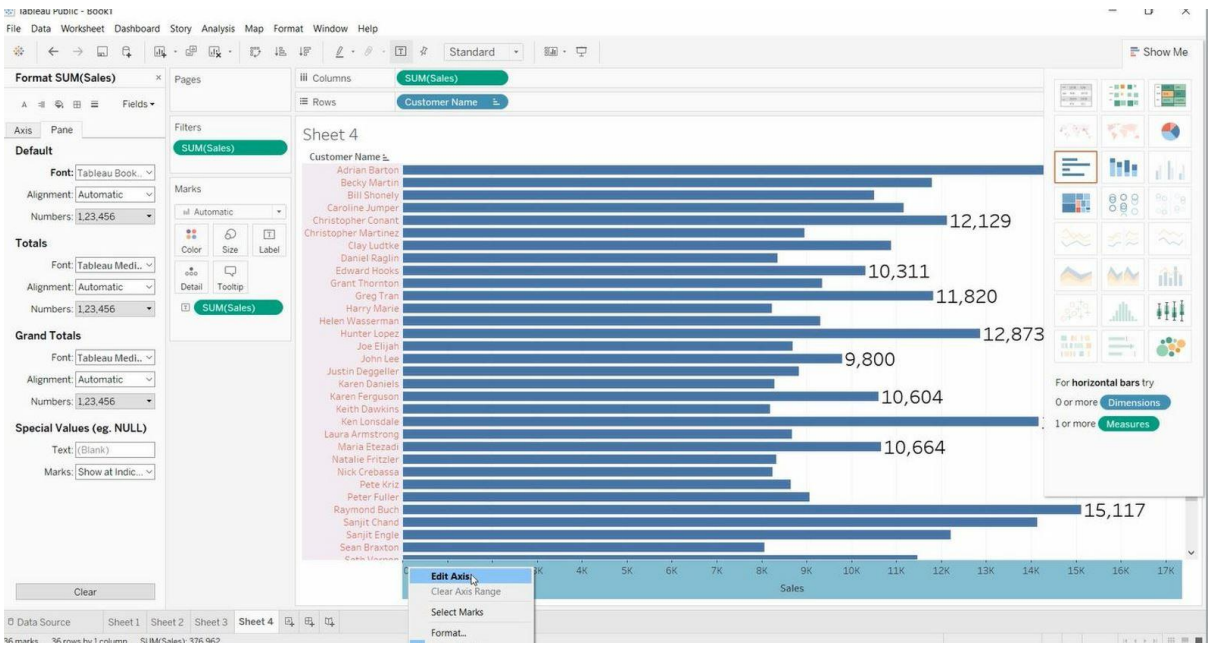


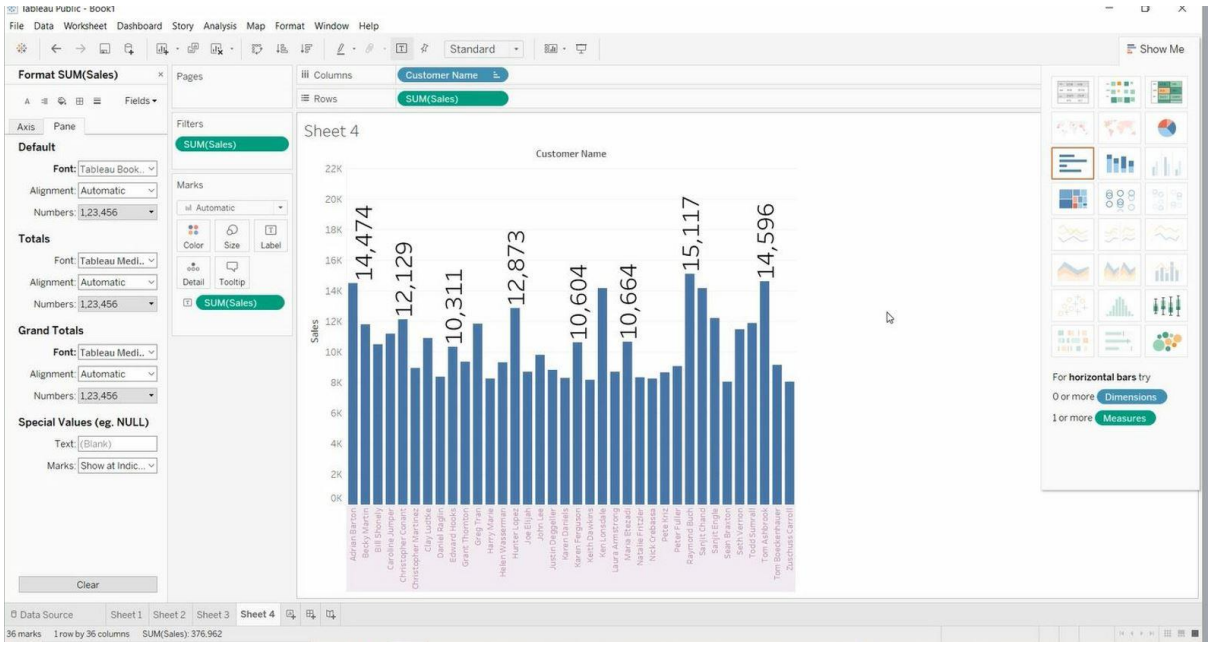
Make Changes



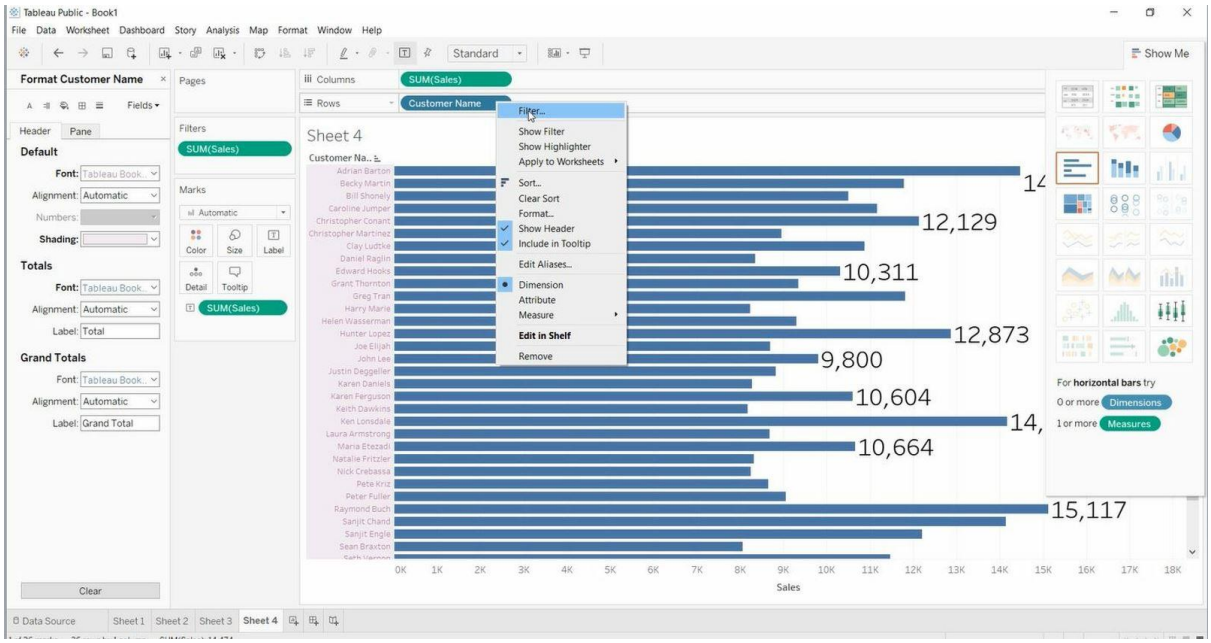
Right Click on **“x-axis”.**



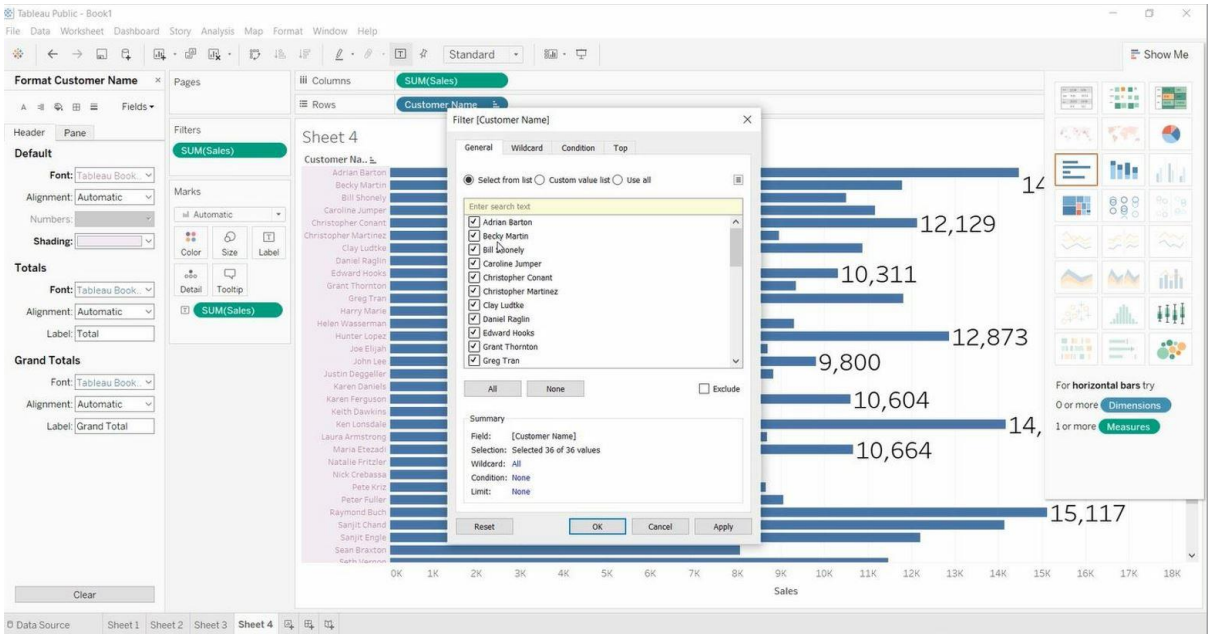




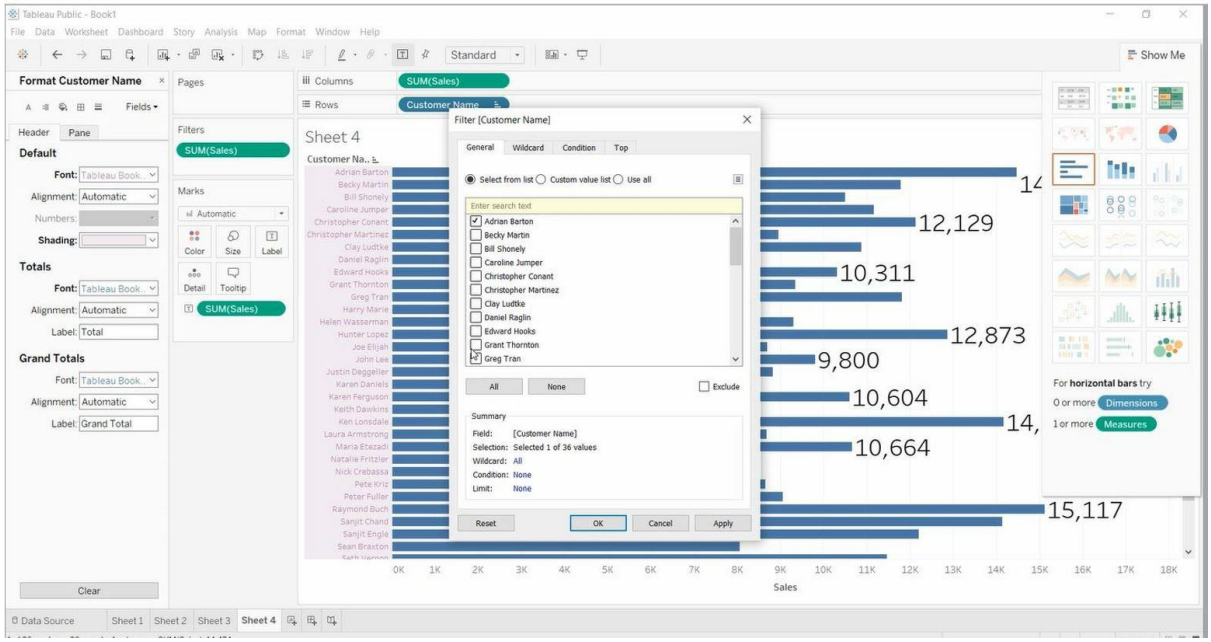
Right Click on **Customer Name** in Rows > Click on **Filter.**



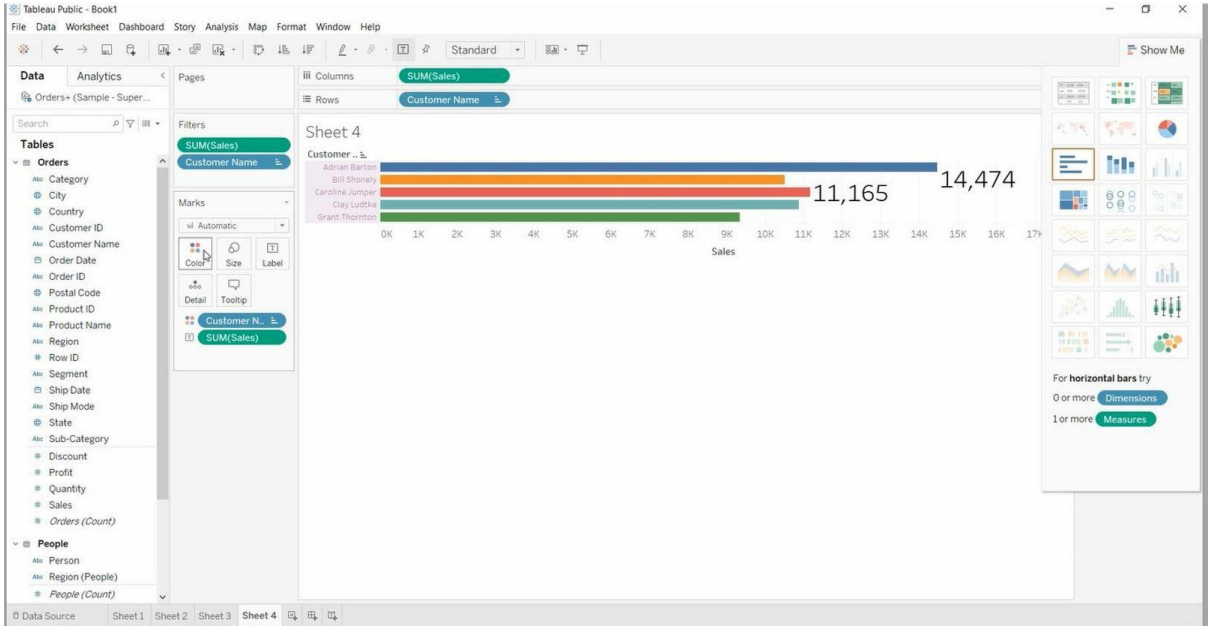
Select **Selected from list**



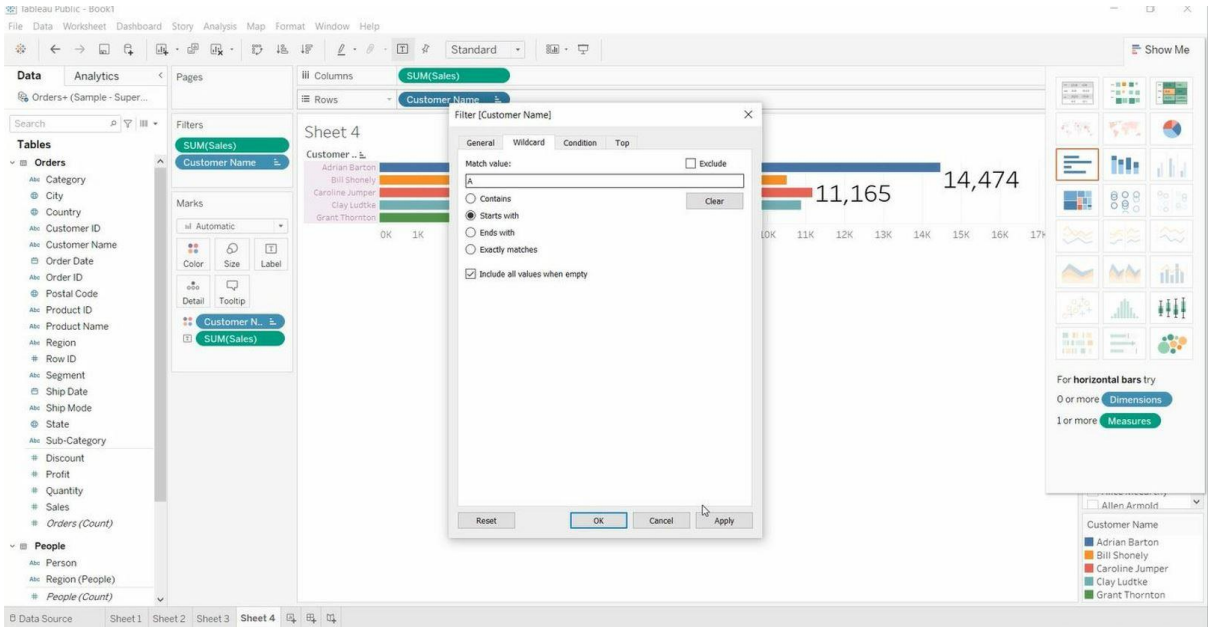
Select 5 or 10 customer from list > Click on **Ok**



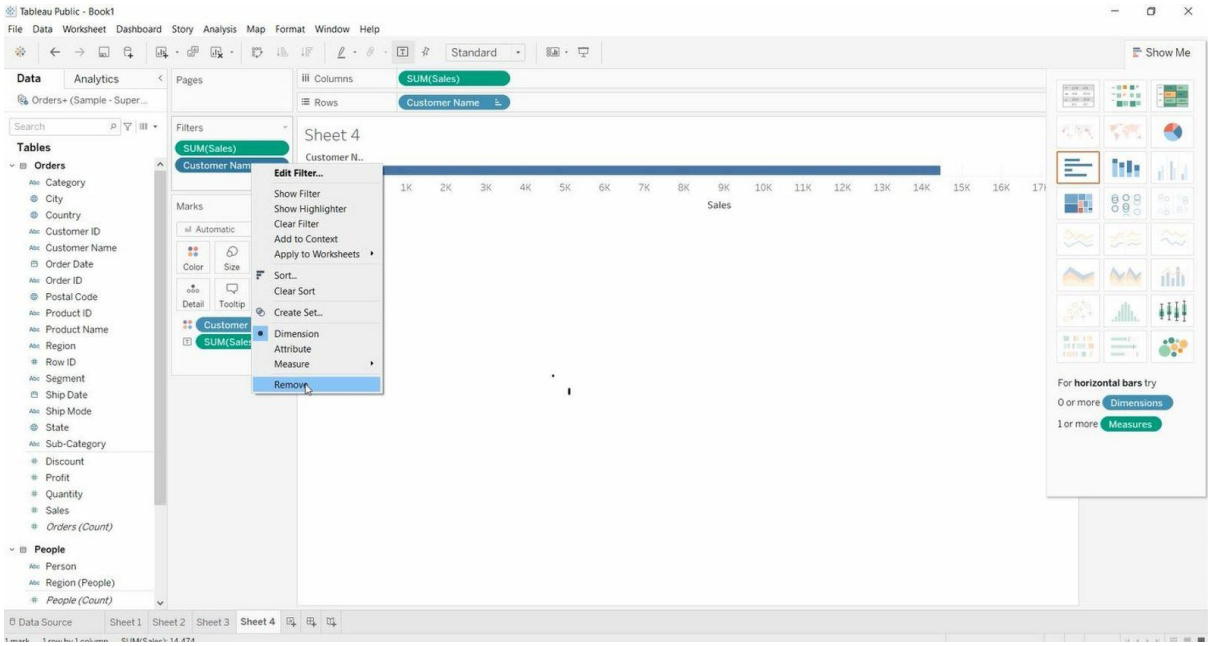
Drag “**Customer Name**” into Colour (Marks card).



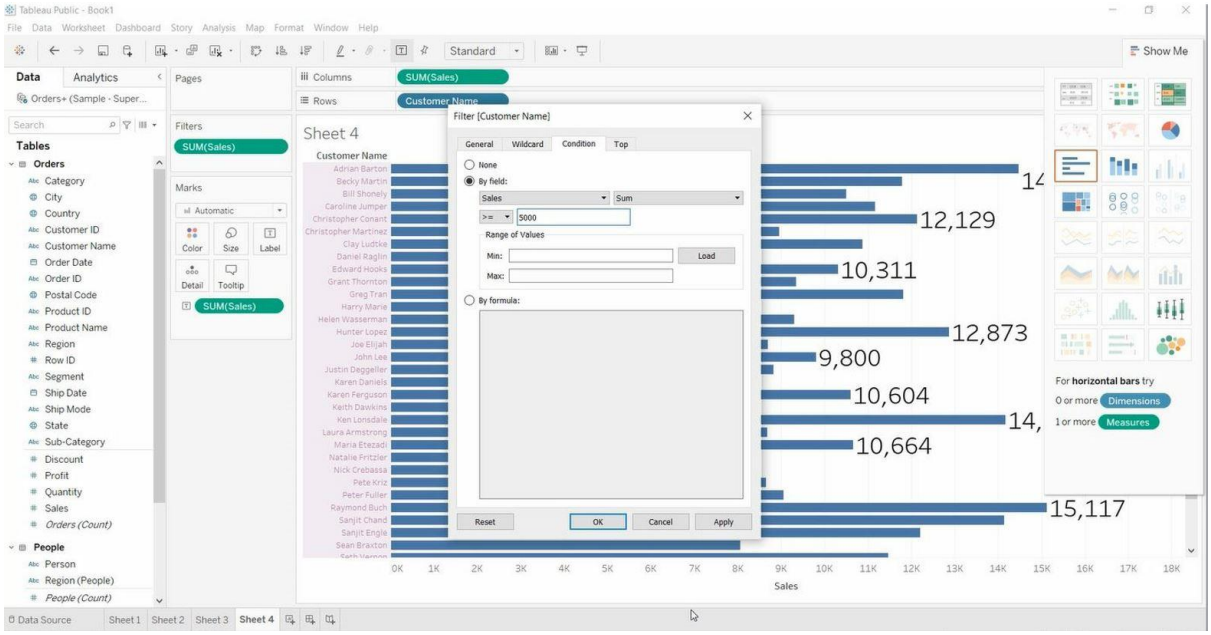
Right Click on **Customer Name** in Rows > Select Wildcard tab > Set Match Value > Ok

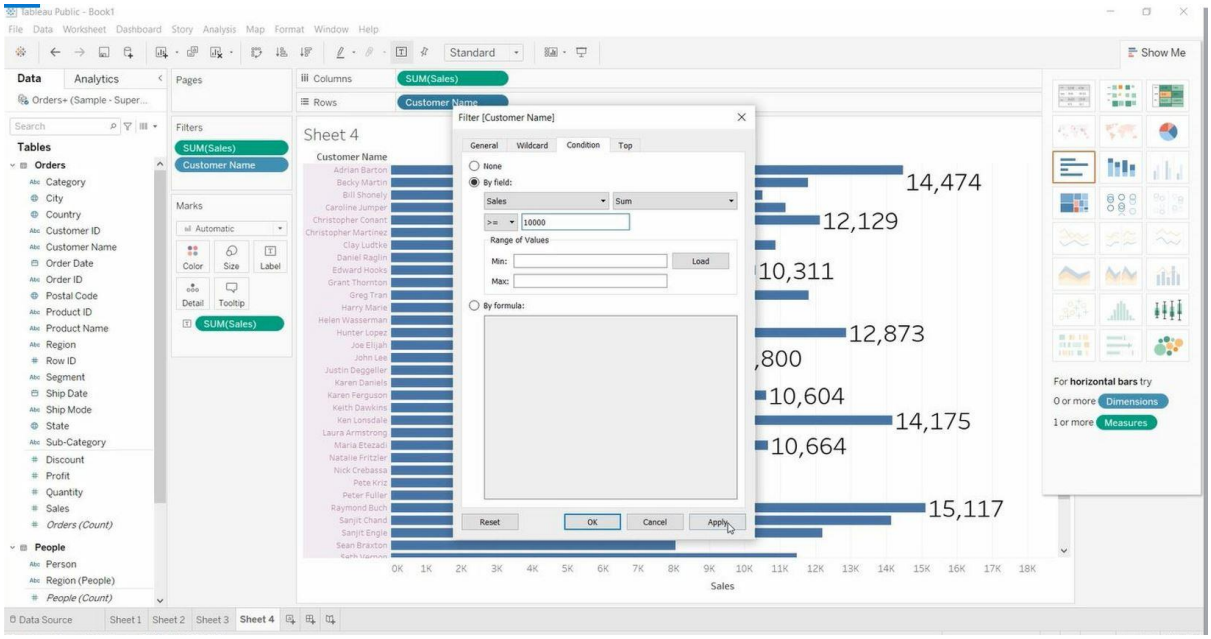


Remove Filter from **Customer Name**

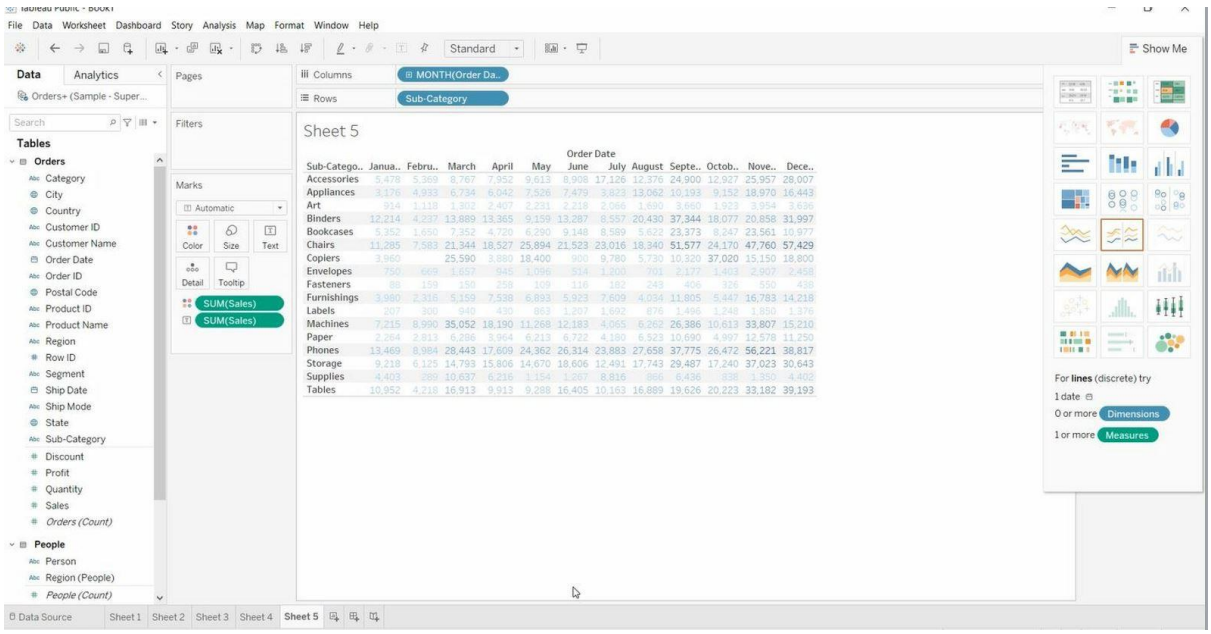


Right Click on **Customer Name** in Rows > Condition tab > Enter Condition Value > Ok

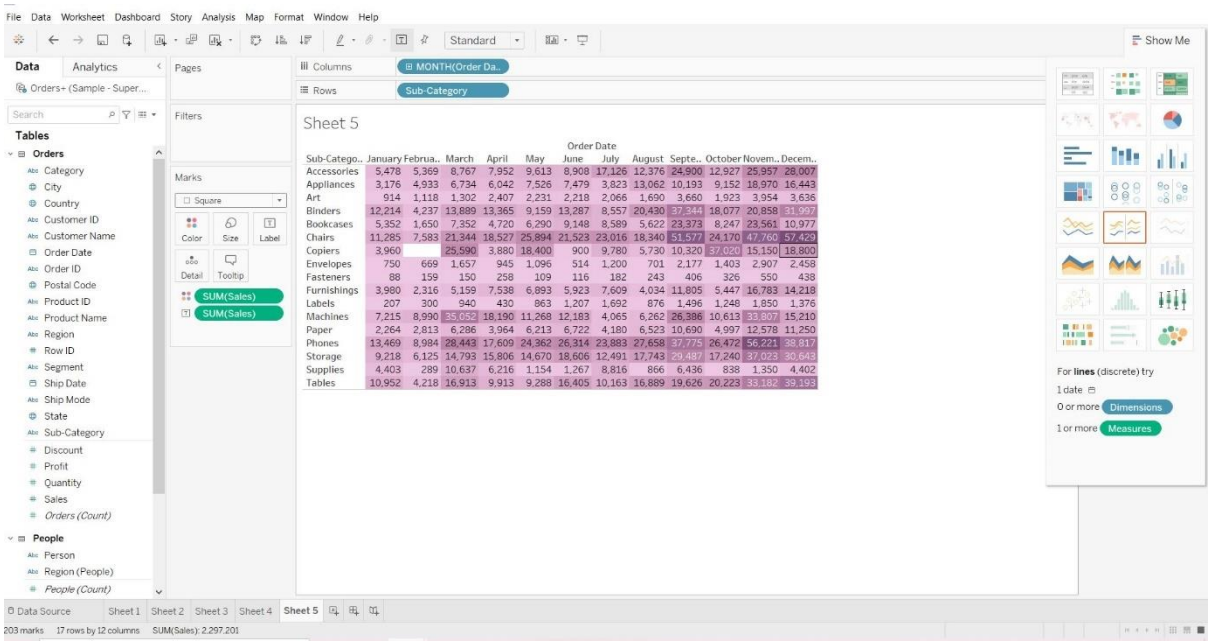




- Open Tableau public and connect to the data source.
- Go to the new worksheet.
- Drag **Sub-Category** into rows and **Order Date** into columns.



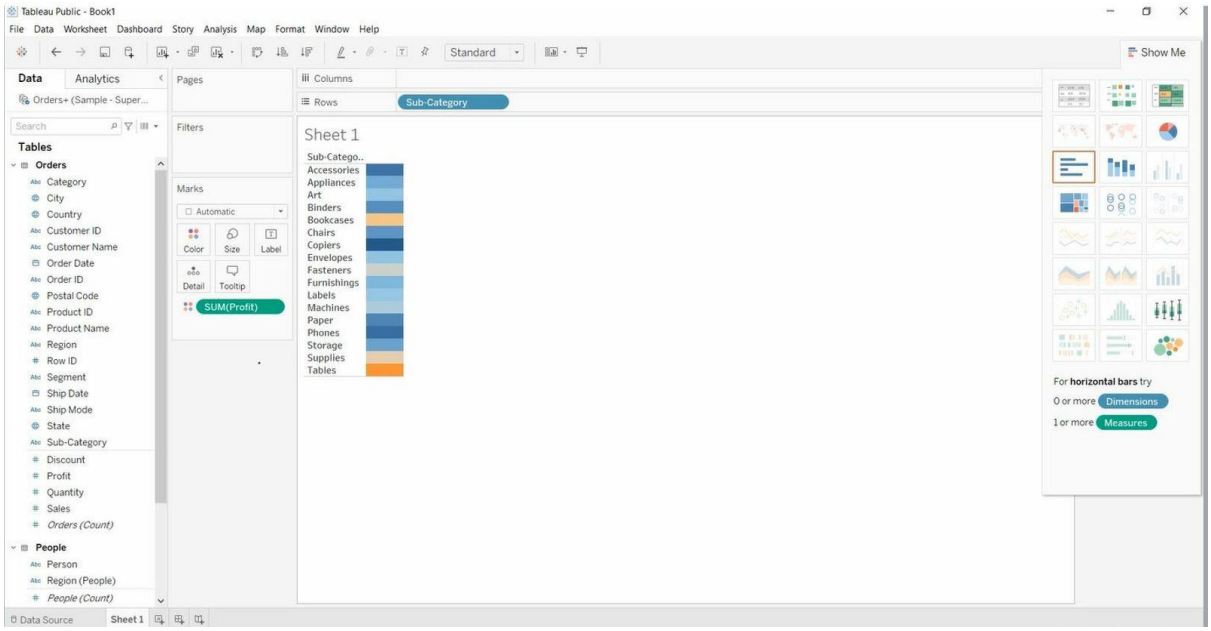
Drag “**Sales**” into Colour & Label (Marks card).



Remove **Order Date** from Columns and **Sales** from Marks Card.

4. <u>Tree Map:</u>

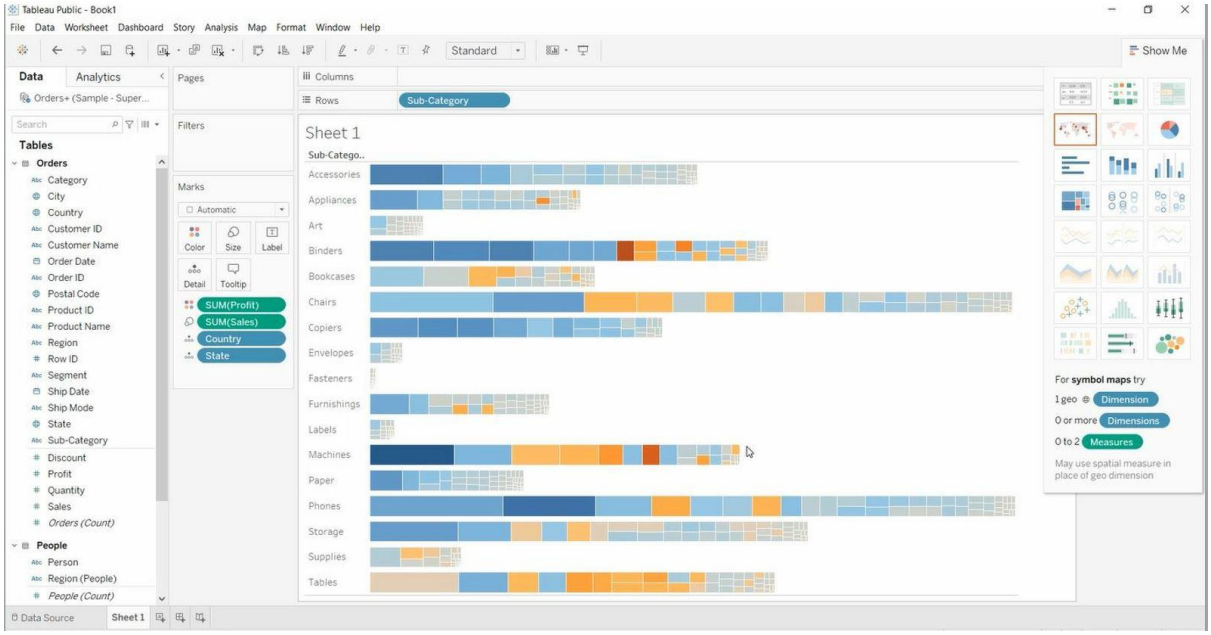
Drag “**Profit**” into Colour (Marks card).



Drag “**Sales**” into Size (Marks card).

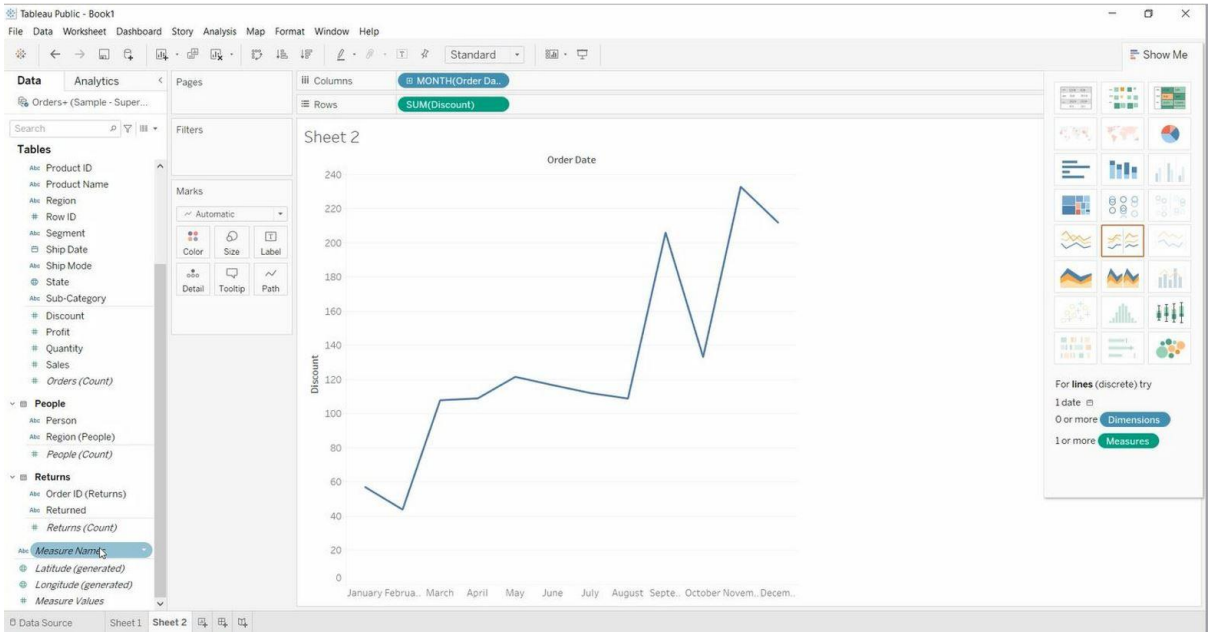
Drag “**Country**” into Detail (Marks card).

Drag “**State**” into Detail (Marks card).

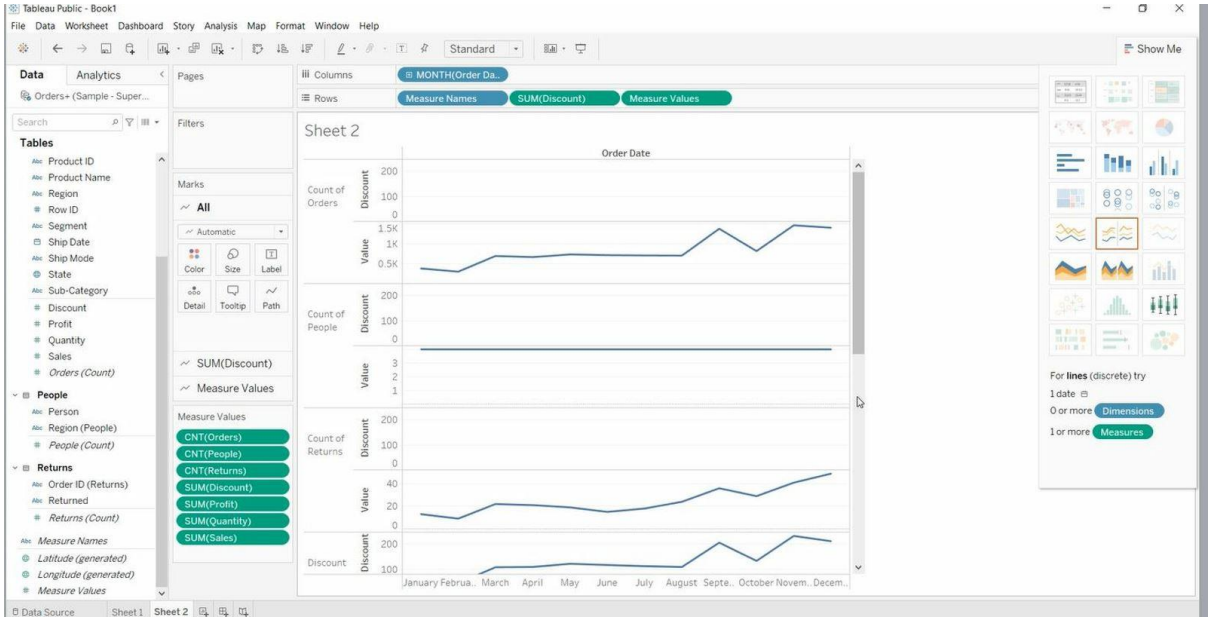


5. <u>Spike Chart:</u>

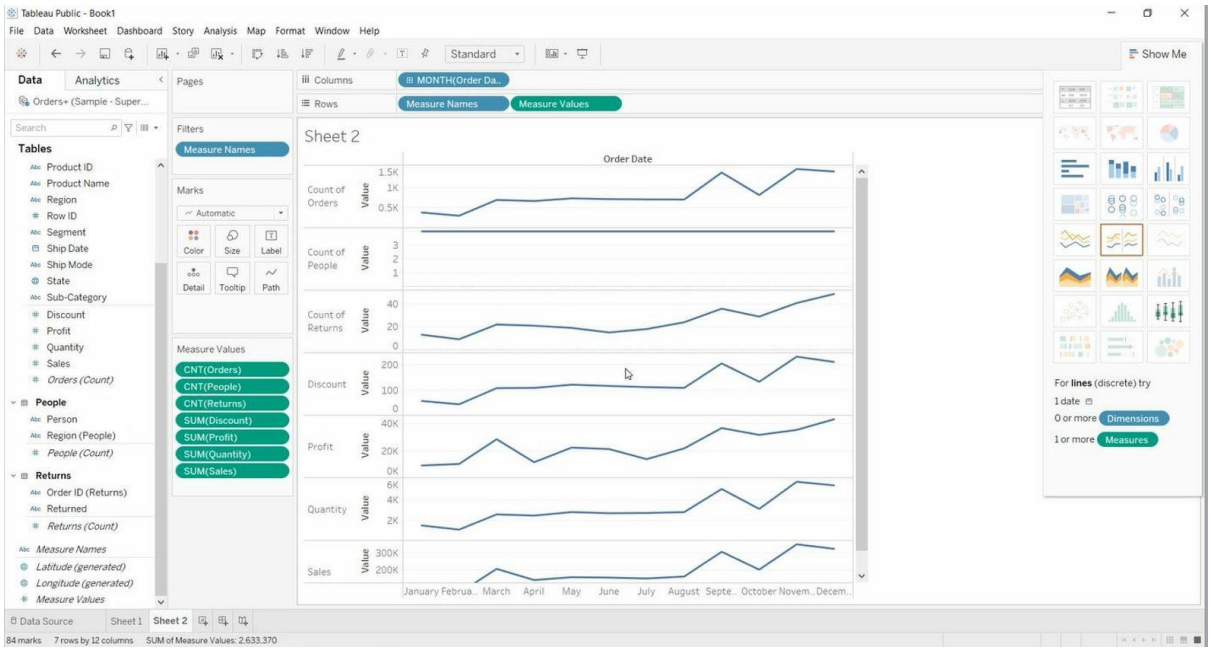
- Open Tableau public and connect to the data source.
- Go to the new worksheet.
- Drag **Discount** into rows and **Order Date** into columns.



Drag “**Measure Names**” & “**Measure Values**” into Rows.

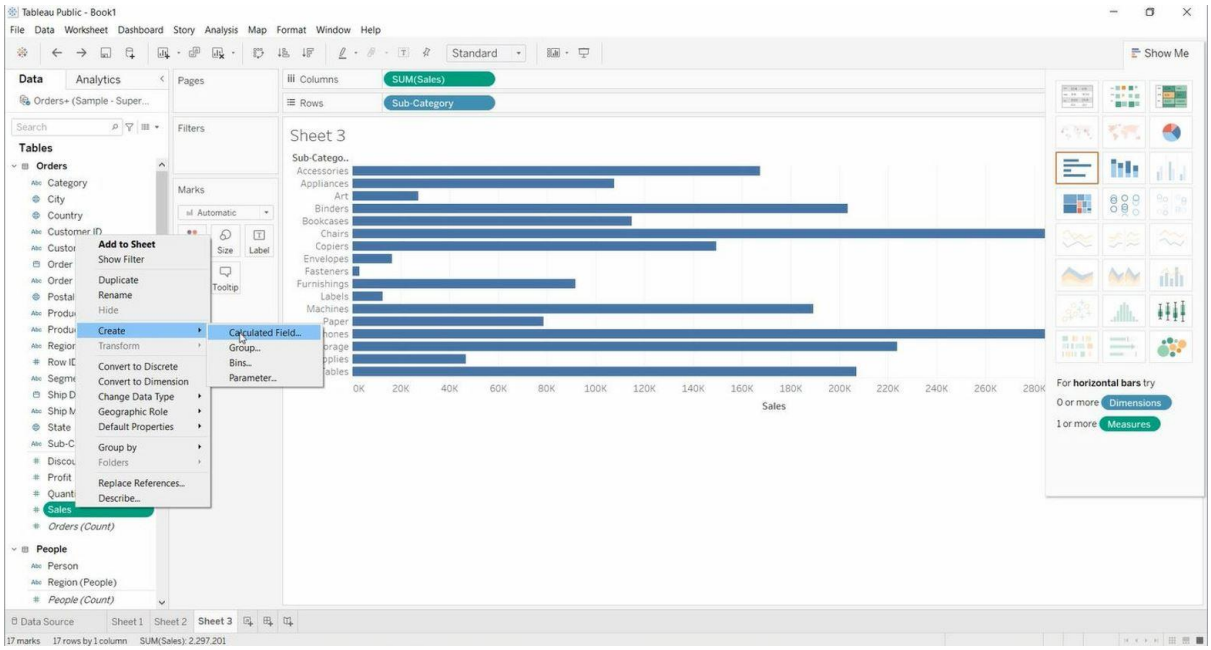


Remove “**Discount**” from Rows.

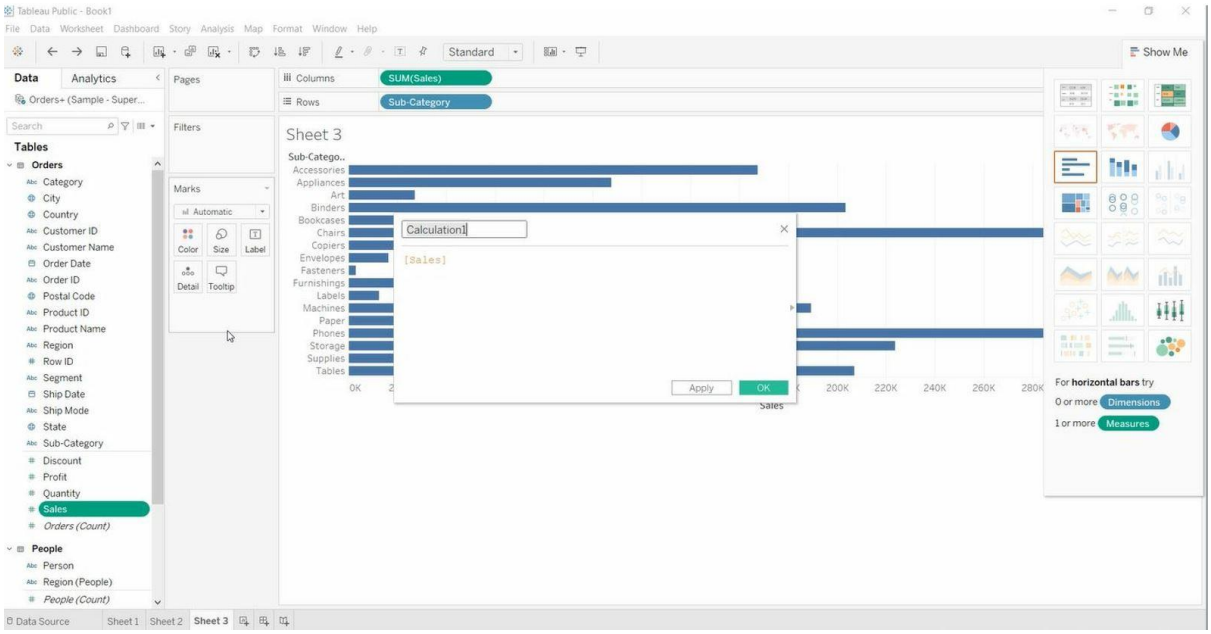


6. Bullet Chart:

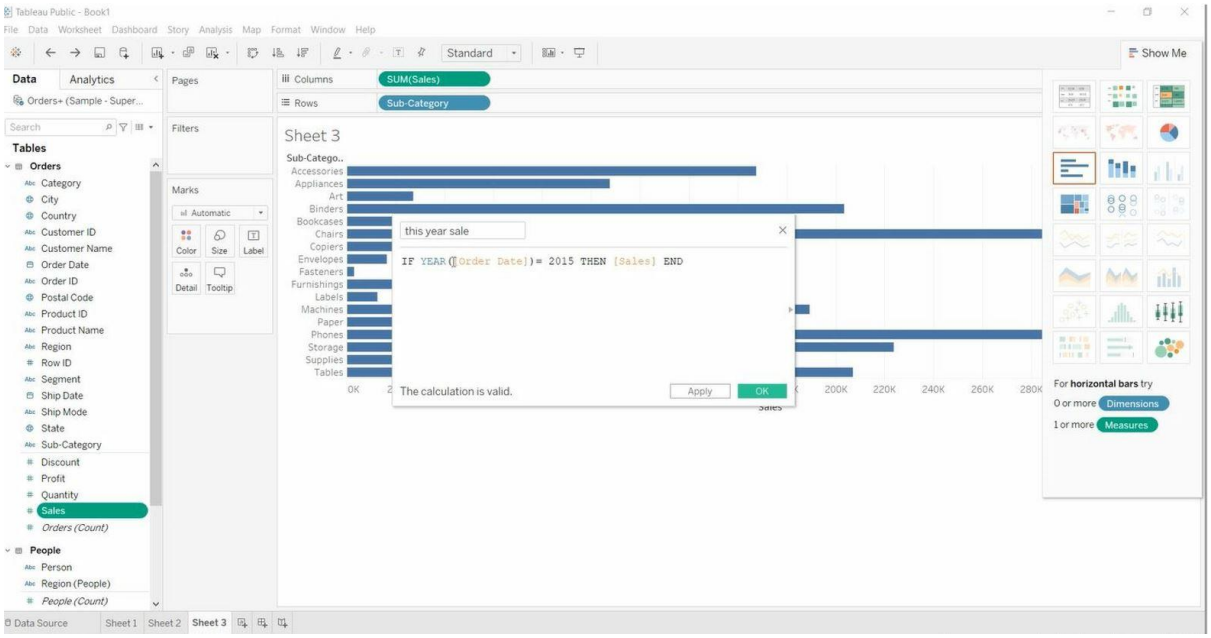
- Open Tableau public and connect to the data source.
- Go to the new worksheet.
- Drag **Sub-Category** into rows and **Sales** into columns.
- Right Click on "**Sales**" > Create > Calculation Field.



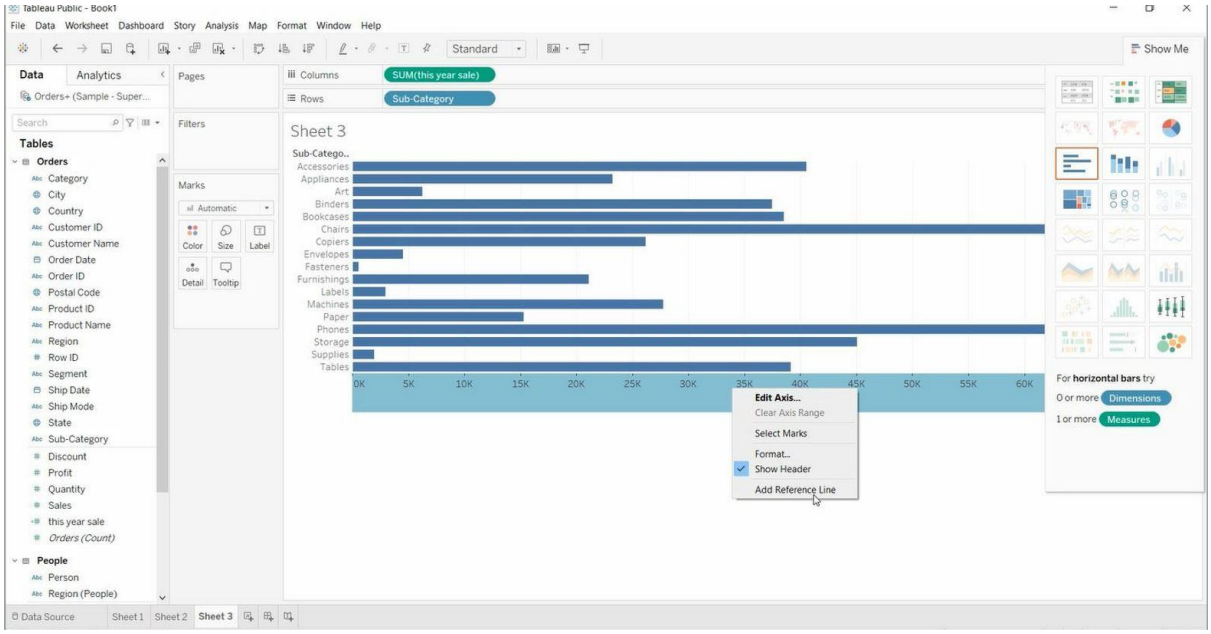
Give any name



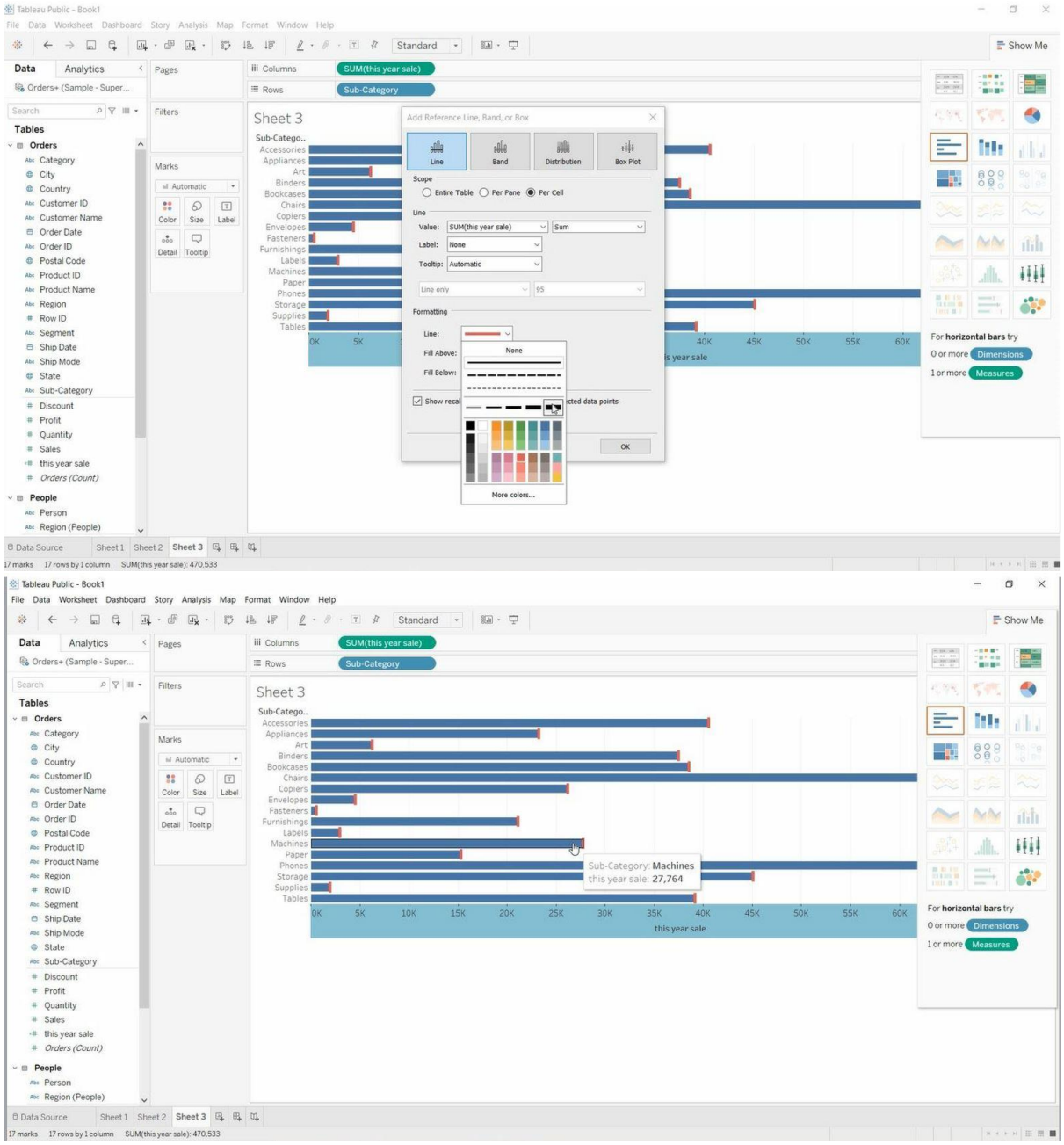
Write the Condition



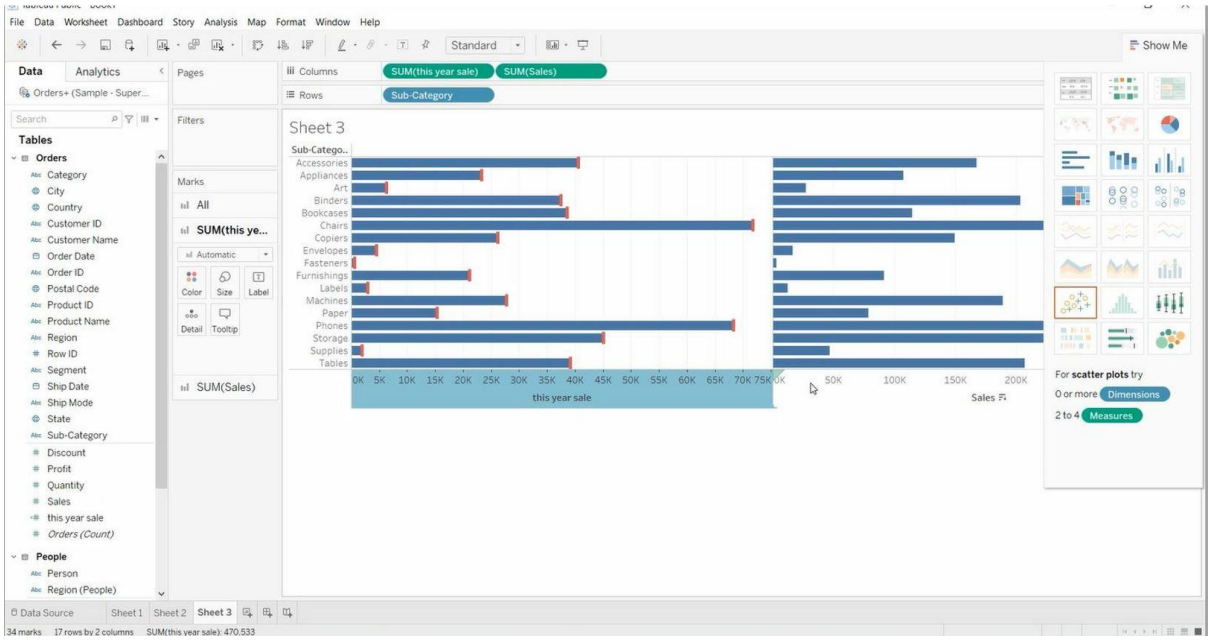
Right Click on "**X-axis**" > Add Reference Line



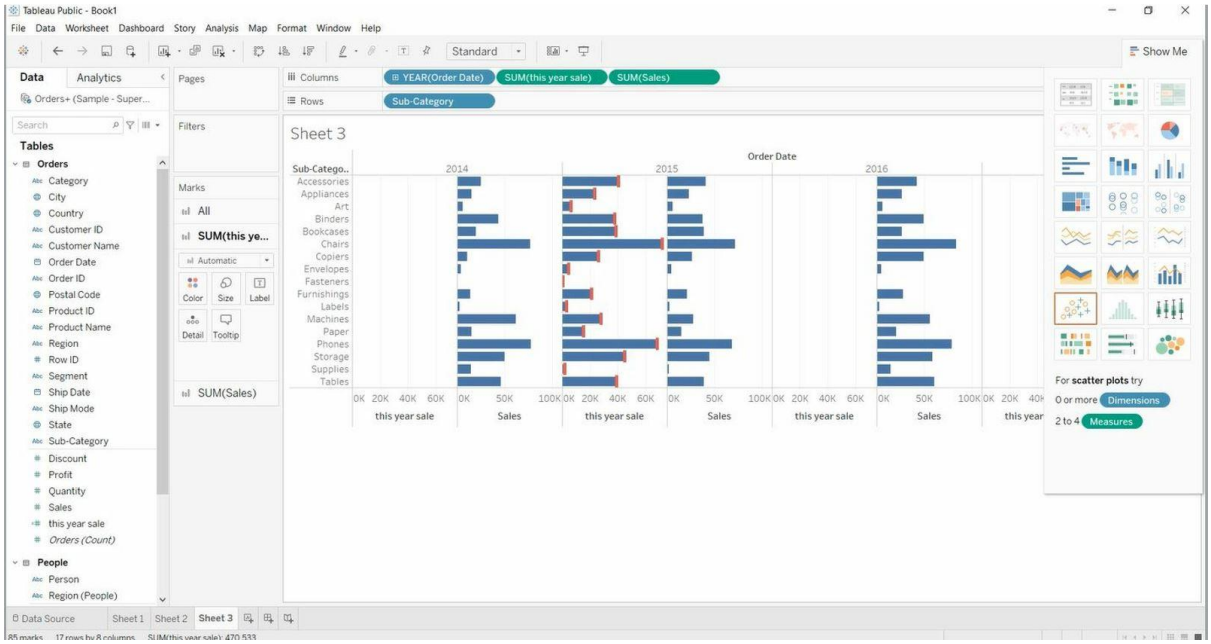
Select “**Line**” > Per Cell > Change Line Colour and Width.



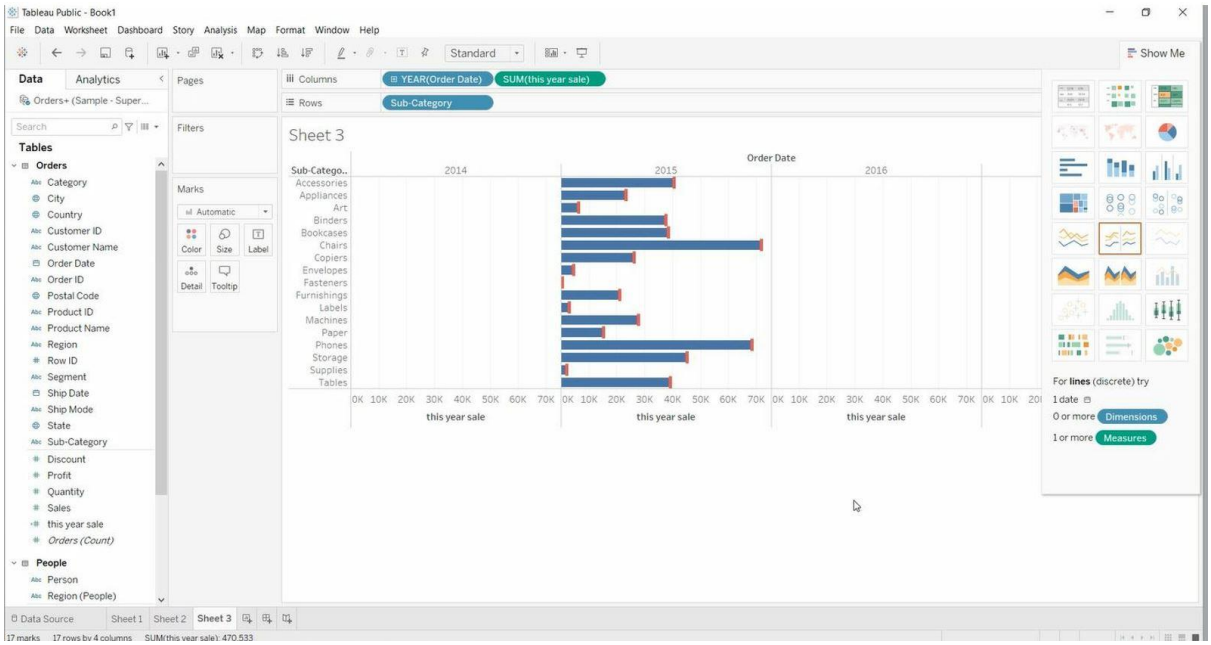
Drag “**Sales**” into Columns.



Drag “**Order Date**” into Columns.

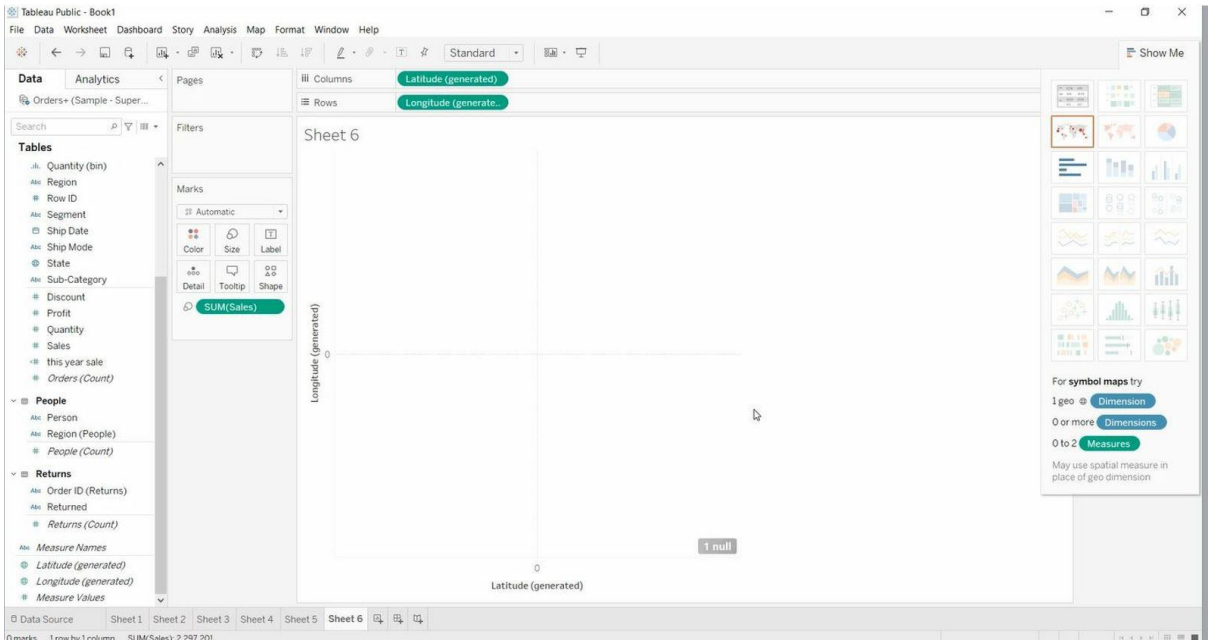


Remove “**Sales”** from Columns.

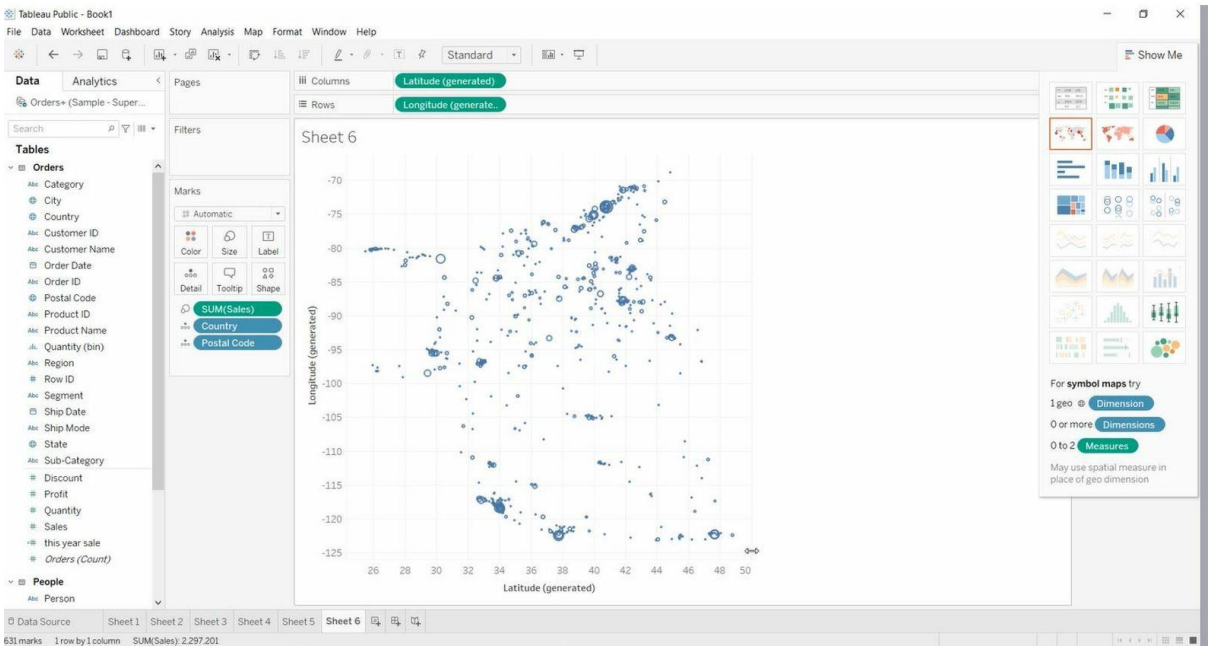


7. Map Chart:

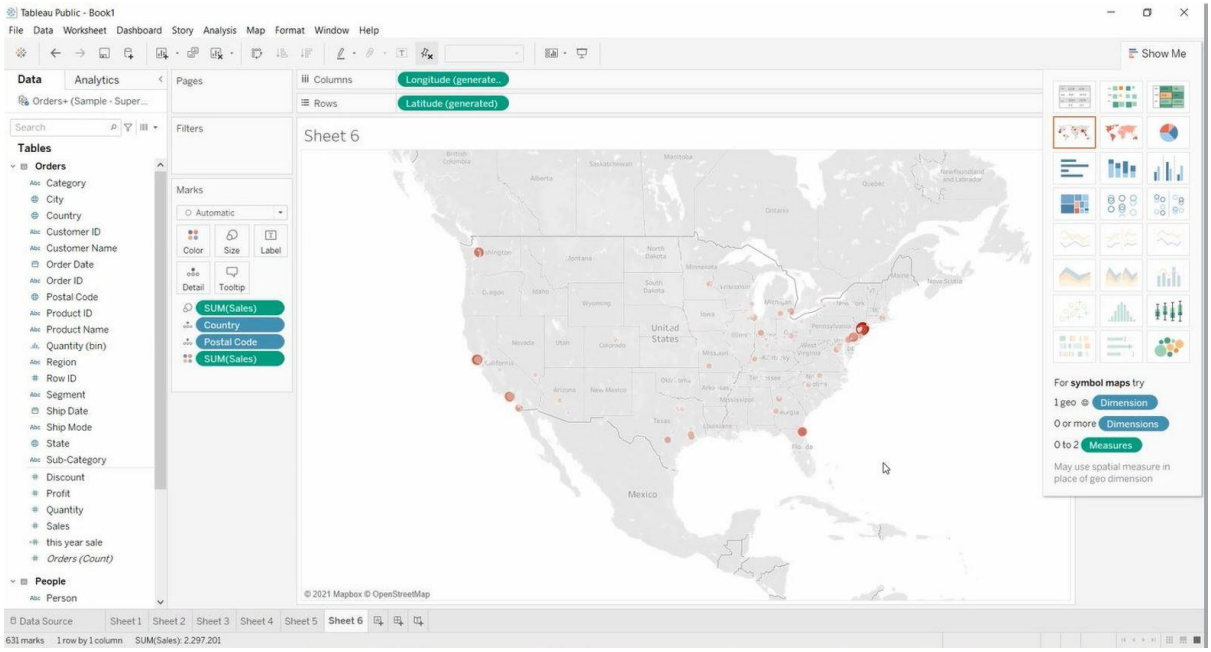
- Open Tableau public and connect to the data source.
- Go to the new worksheet.
- Drag **Longitude** into rows and **Latitude** into columns.
- Drag “**Sales**” into Size (Marks card).



Drag “**Country**” & “**Postal Code**” into Detail (Marks card).

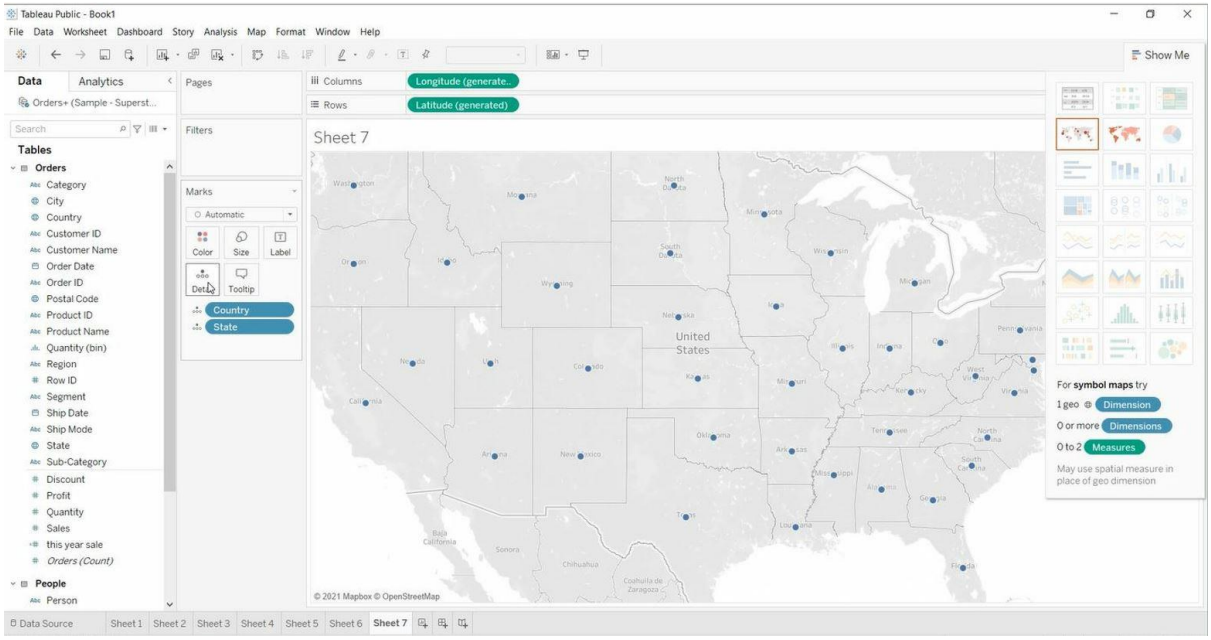


Drag “**Sales**” into Colour (Marks card).

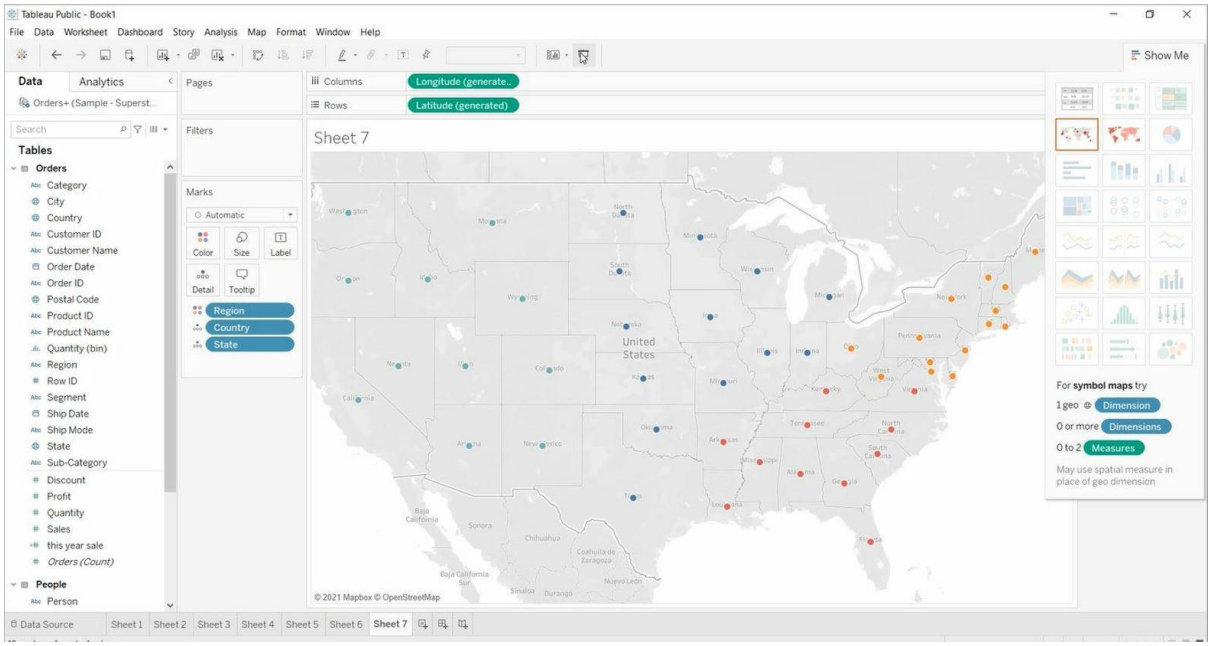


Remove **Postal Code** & **Sales** from Marks Cards.

Drag "**State**" into Detail (Marks card).

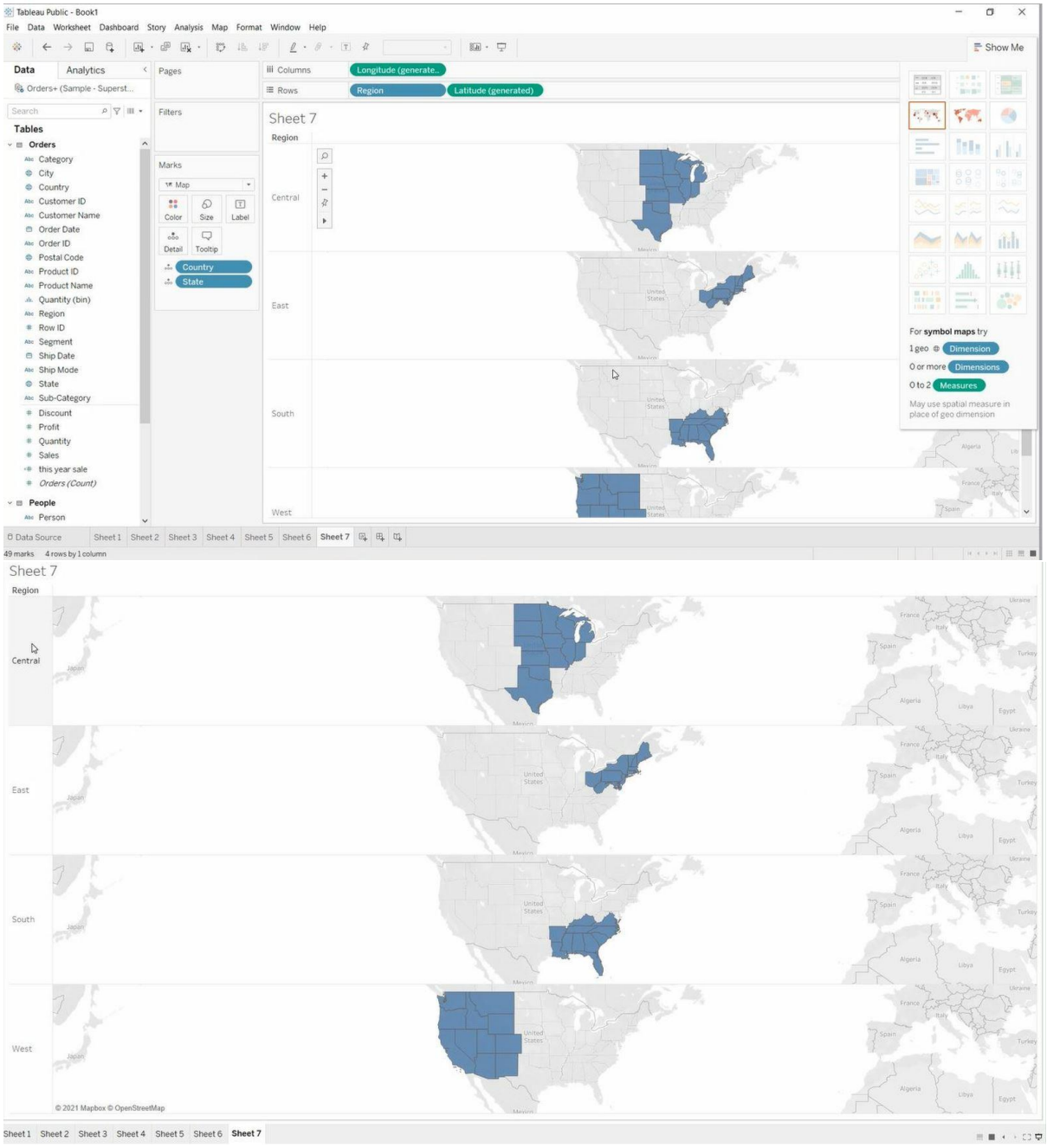


Drag "**Region**" into Colour (Marks card).

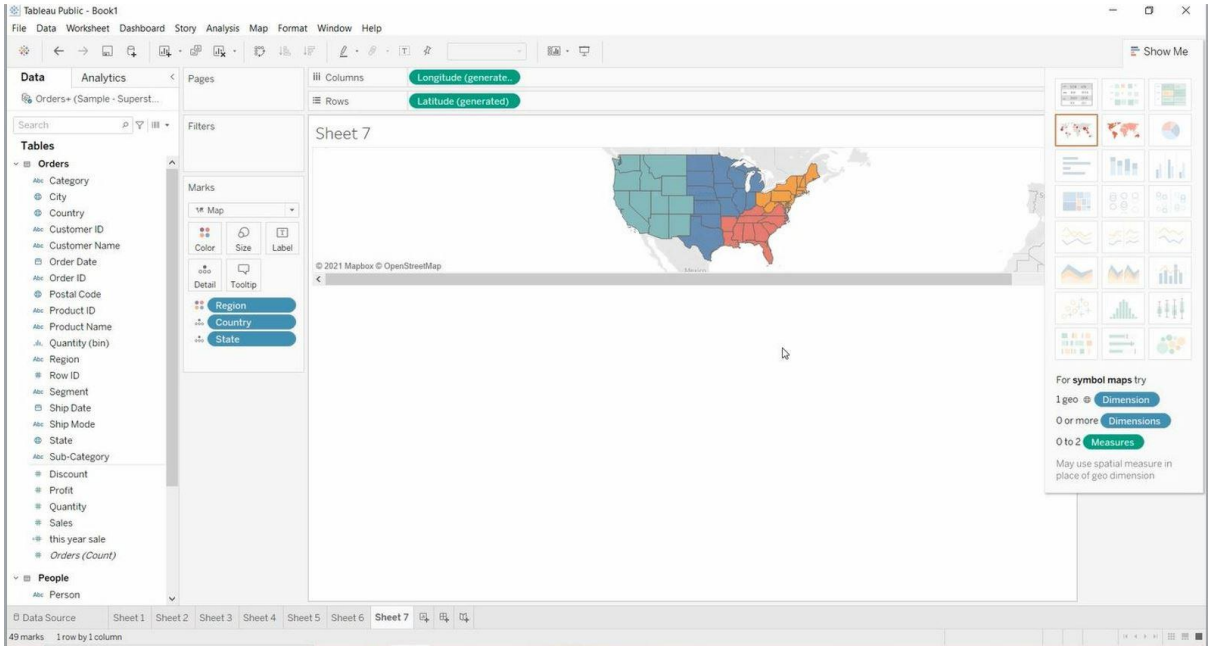


Remove **“Region”** from Marks Card.

Drag “**Region**” into Rows.

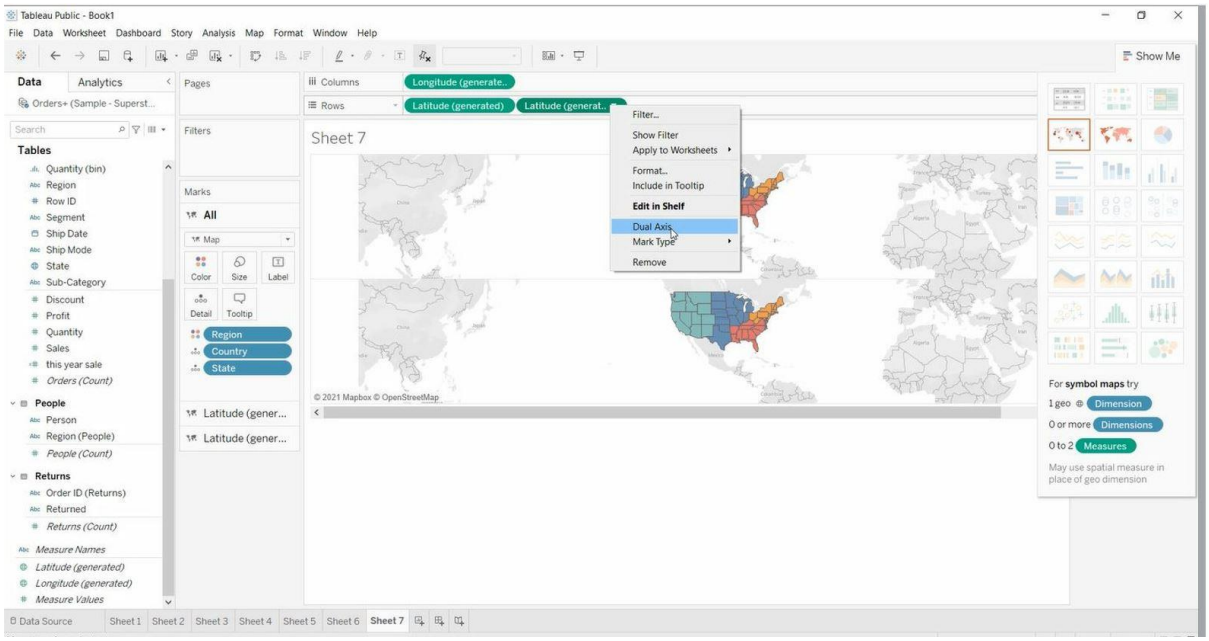


Drag “**Region**” into Colour (Marks Card).

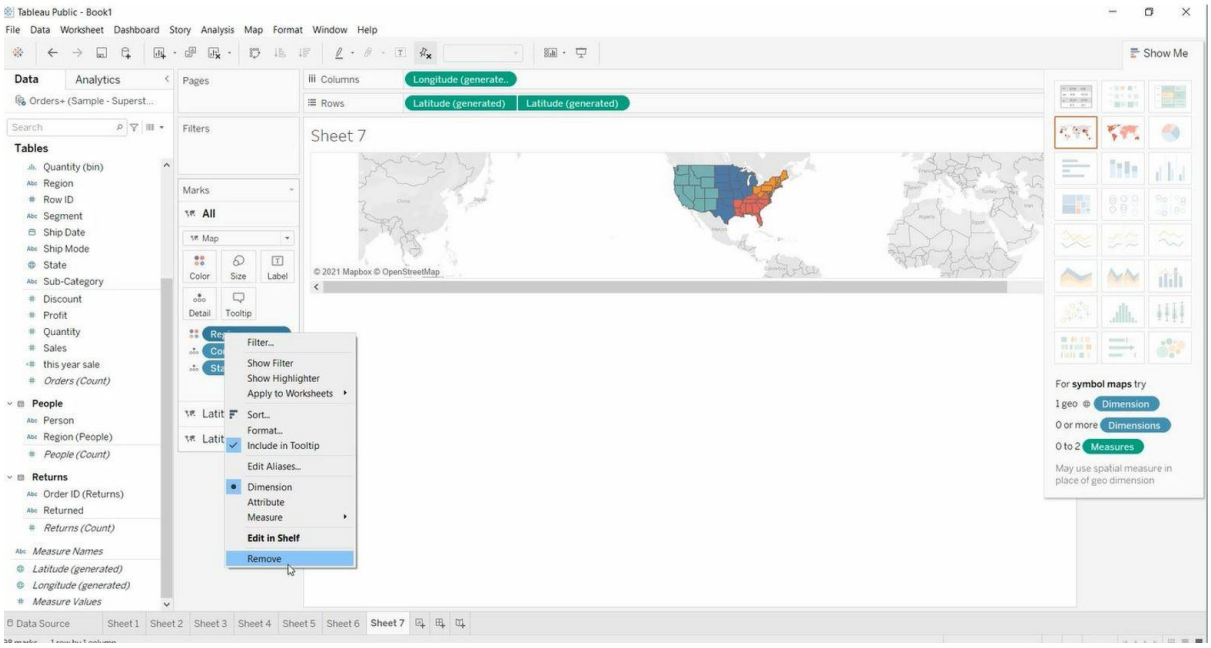


Drag “**Latitude**” into Rows.

Right Click on **Latitude >** Click on **Dual Axis**.



Remove **“Region”** from Marks Card.



Drag “**State**” into Colour (Marks Card).

Drag “**Sales**” into Size (Marks Card).

Drag “**City**” into Detail (Marks Card).