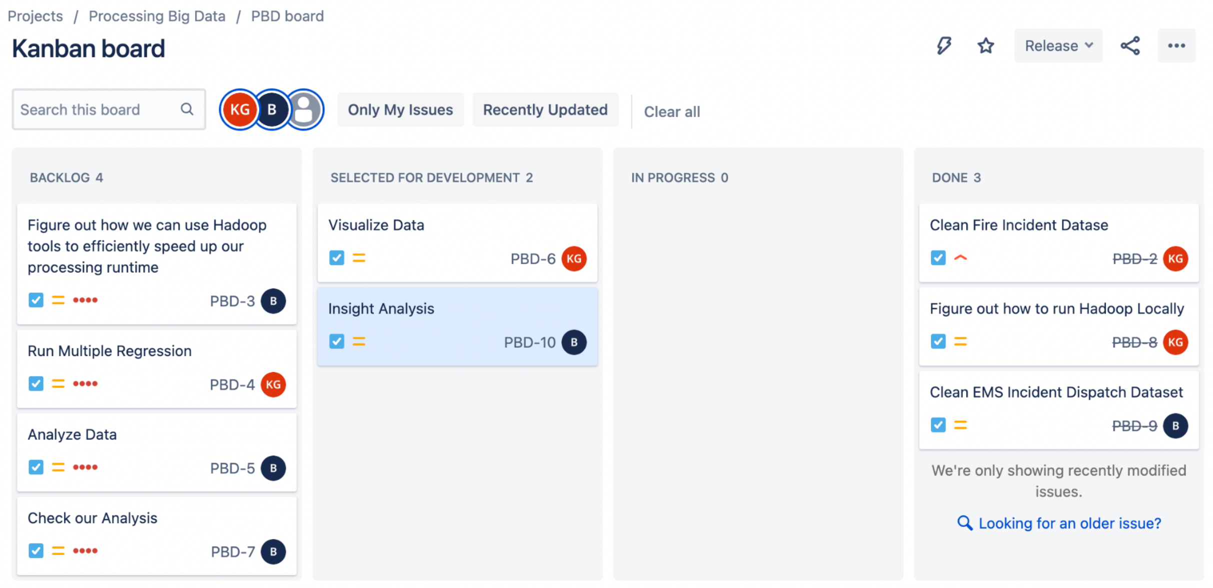
Jira


Projects / Processing Big Data / PBD board

Kanban board

BACKLOG 4
- Figure out how we can use Hadoop tools to efficiently speed up our processing runtime — PBD-3 B
- Run Multiple Regression — PBD-4 KG
- Analyze Data — PBD-5 B
- Check our Analysis — PBD-7 B

SELECTED FOR DEVELOPMENT 2
- Visualize Data — PBD-6 KG
- Insight Analysis — PBD-10 B

IN PROGRESS 0

DONE 3
- Clean Fire Incident Datase — ~~PBD-2~~ KG
- Figure out how to run Hadoop Locally — ~~PBD-8~~ KG
- Clean EMS Incident Dispatch Dataset — ~~PBD-9~~ B

We're only showing recently modified issues.

Looking for an older issue?

Cleaning

Fire Incident Dataset

We used this code to clean the dataset with the fire incident dispatch data. We dropped rows with null values and columns that we didn't think would be pertinent to our data analysis.

Driver:

```
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 Clean {
 public static void main(String[] args) throws Exception {
   Job job = new Job();
   job.setJarByClass(Clean.class);
   job.setJobName("Count Lines");
   job.setNumReduceTasks(1);
   FileInputFormat.addInputPath(job, new Path(args[0]));
   FileOutputFormat.setOutputPath(job, new Path(args[1]));
   job.setMapperClass(CleanMapper.class);
   job.setReducerClass(CleanReducer.class);
   job.setOutputKeyClass(Text.class);
   job.setOutputValueClass(Text.class);
   System.exit(job.waitForCompletion(true) ? 0 : 1);
 }
}
```

Mapper:

```
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 CleanMapper
   extends Mapper<Object, Text, Text, Text> {
   private final static IntWritable one = new IntWritable(1);
   private final static IntWritable zero = new IntWritable(0);
   private Text word = new Text();
   public void map(Object key, Text value, Context context) throws
IOException, InterruptedException {
     String input = new String(value.toString());
     String[] sentence = input.split(",");
     if (sentence.length == 29){
       String id = sentence[0];
       String date = sentence[1];
       String alarmNumber = sentence[3];
       String incidentBorough = sentence[5];
       String zip = sentence[6];
       String policePrecinct = sentence[7];
       String alarmDescription = sentence[12];
       String highestAlarm = sentence[14];
       String classification = sentence[15];
       String dispatchResponse = sentence[17];
       String assignment = sentence[18];
       String activation = sentence[19];
       String onScene = sentence[20];
       String incidentResponseTime = sentence[24];
       String incidentTravelTime = sentence[25];
       String engines = sentence[26];
       String otherUnits = sentence[28];
       context.write(new Text(id + "," + date + "," + alarmNumber + "," +
incidentBorough + "," + zip + "," + policePrecinct + "," + alarmDescription +
"," + highestAlarm + "," + classification + "," + dispatchResponse + "," +
assignment + "," + activation + "," + onScene + "," + incidentResponseTime +
"," + incidentTravelTime + "," + engines + "," + otherUnits), new Text(""));
   }
   }
}
```

Reducer:

```
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 CleanReducer
   extends Reducer<Text,Text,Text,Text> {

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

     context.write(key, values);
   }
}
```

<u>EMS Dataset</u>
We used this code to clean the dataset with EMS dispatch data. We dropped rows with null values and columns that were not relevant to our analytics.

Driver:

```
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 Clean {
       public static void main(String[] args) throws Exception {
           Configuration conf = new Configuration();
           Job job = Job.getInstance(conf, "Data Cleaning");
           job.setNumReduceTasks(1);
           job.setJarByClass(Clean.class);
           job.setMapperClass(CleanMapper.class);
           job.setCombinerClass(CleanReducer.class);
           job.setReducerClass(CleanReducer.class);
           job.setOutputKeyClass(Text.class);
           job.setOutputValueClass(Text.class);
```

```
            FileInputFormat.addInputPath(job, new Path(args[0]));
            FileOutputFormat.setOutputPath(job, new Path(args[1]));
            System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Mapper:

```
import java.io.IOException;
import java.util.StringTokenizer;
import java.util.*;
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 CleanMapper extends Mapper<Object, Text, Text, Text>{
    public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
       String line = value.toString();
       String[] values = line.split(",");
       if (values.length == 31){
           String one = values[0];
           String two = values[4];
           String three = values[5];
           String four = values[8];
           String five = values[12];
           String six = values[13];
           String seven = values[19];
           String eight = values[21];
           String nine = values[22];
           context.write(new
Text(one+","+two+","+three+","+four+","+five+","+six+","+seven+","+eight+","+nine),
new Text(""));
       }
    }
}
```

Reducer:
```
import java.io.IOException;
import java.util.StringTokenizer;
import java.util.*;
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 CleanReducer extends Reducer<Text,Text,Text,Text> {

   Text column = new Text();

   public void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException{
       context.write(key, column);
   }
}
```

Analysis

We used this to count the number of records in the dataset before and after cleaning

Driver:
```
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;
```

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

public class CountRecs {
       public static void main(String[] args) throws Exception {
           Configuration conf = new Configuration();
           Job job = Job.getInstance(conf, "Record Count");
           job.setNumReduceTasks(1);
           job.setJarByClass(CountRecs.class);
           job.setMapperClass(CountRecsMapper.class);
           job.setCombinerClass(CountRecsReducer.class);
           job.setReducerClass(CountRecsReducer.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);
   }
}
```

Mapper:

```java
import java.io.IOException;
import java.util.StringTokenizer;
import java.util.*;
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 CountRecsMapper extends Mapper<Object, Text, Text, IntWritable>{

   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text("Total number of records in EMS Incident Dispatch
file");
    public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
       context.write(word, one);
   }
}
```

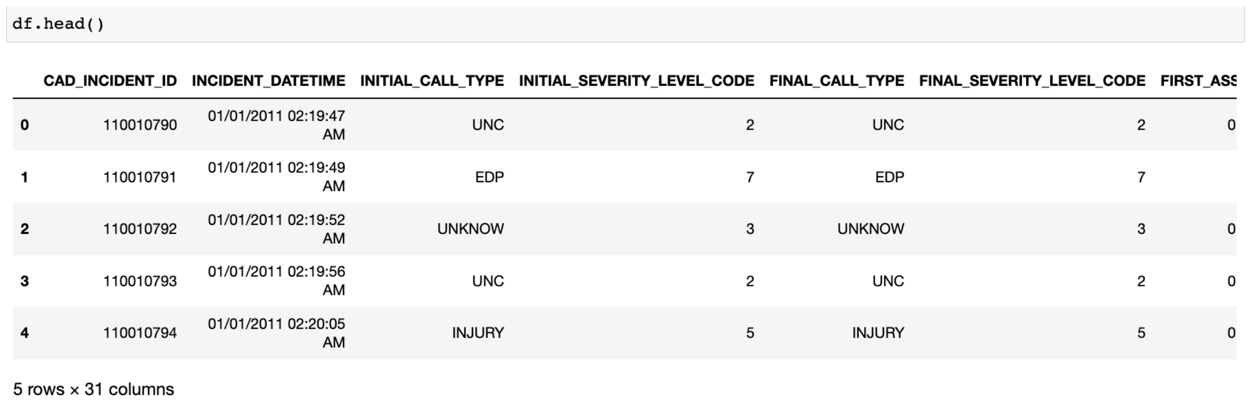
Reducer:

```
import java.io.IOException;
import java.util.StringTokenizer;
import java.util.*;
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 CountRecsReducer 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 count = 0;

       for(IntWritable value : values) {
           count += value.get();
       }
       result.set(count);
       context.write(key, result);
   }
}
```

Visualizations (In Progress)
We used pandas and python in Jupyter Notebook to conduct visualizations of our data. We only have the correlation matrix so far, but plan on doing more in the near future.

```
import numpy as np
import pandas as pd
data = pd.read_csv('EMS_Incident_Dispatch_Data.csv')
df = pd.DataFrame(data)
df.head()
cleaned =
df[['CAD_INCIDENT_ID','FINAL_CALL_TYPE','FINAL_SEVERITY_LEVEL_CODE','DISPATCH_RESPON
SE_SECONDS_QY','INCIDENT_RESPONSE_SECONDS_QY','INCIDENT_TRAVEL_TM_SECONDS_Q
Y','BOROUGH','ZIPCODE','POLICEPRECINCT']]
cleaned.dropna()
numeric = cleaned.drop(['FINAL_CALL_TYPE','BOROUGH'], axis=1)
import matplotlib.pyplot as plt
f = plt.figure(figsize=(19, 15))
plt.matshow(cleaned.corr(), fignum=f.number)
plt.xticks(range(cleaned.select_dtypes(['number']).shape[1]), cleaned.select_dtypes(['number']).columns,
fontsize=14, rotation=45)
plt.yticks(range(cleaned.select_dtypes(['number']).shape[1]), cleaned.select_dtypes(['number']).columns,
fontsize=14)
cb = plt.colorbar()
cb.ax.tick_params(labelsize=14)
plt.title('Correlation Matrix', fontsize=16);
```

Screenshots:



```
df.head()
```

	CAD_INCIDENT_ID	INCIDENT_DATETIME	INITIAL_CALL_TYPE	INITIAL_SEVERITY_LEVEL_CODE	FINAL_CALL_TYPE	FINAL_SEVERITY_LEVEL_CODE	FIRST_ASS
0	110010790	01/01/2011 02:19:47 AM	UNC	2	UNC	2	0
1	110010791	01/01/2011 02:19:49 AM	EDP	7	EDP	7	
2	110010792	01/01/2011 02:19:52 AM	UNKNOW	3	UNKNOW	3	0
3	110010793	01/01/2011 02:19:56 AM	UNC	2	UNC	2	0
4	110010794	01/01/2011 02:20:05 AM	INJURY	5	INJURY	5	0

5 rows × 31 columns

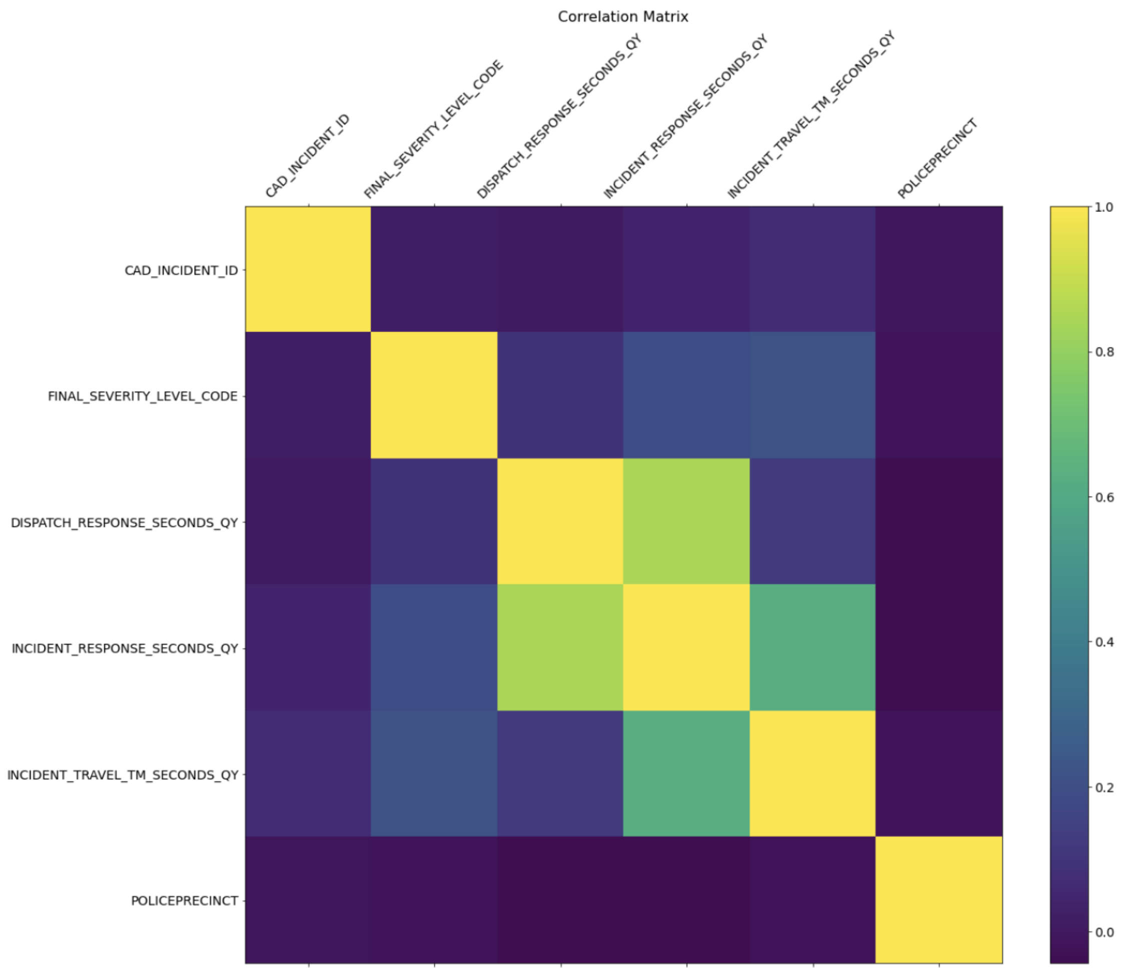
```
In [4]: cleaned = df[['CAD_INCIDENT_ID','FINAL_CALL_TYPE','FINAL_SEVERITY_LEVEL_CODE','DISPATCH_RESPONSE_SECONDS_QY','INCIDENT_
```

```
In [5]: cleaned.dropna()
```

Out[5]:

ODE	DISPATCH_RESPONSE_SECONDS_QY	INCIDENT_RESPONSE_SECONDS_QY	INCIDENT_TRAVEL_TM_SECONDS_QY	BOROUGH	ZIPCODE	POLICEPRECINCT
2	87	404.0	317.0	MANHATTAN	10030	32.0
3	313	457.0	144.0	MANHATTAN	10016	14.0
2	35	417.0	382.0	BROOKLYN	11213	77.0
5	4139	4139.0	0.0	MANHATTAN	10022	18.0
3	97	846.0	749.0	BROOKLYN	11208	75.0
...	...	...	...	...	...	...
5	34	591.0	557.0	MANHATTAN	10014	6.0
6	61	1140.0	1079.0	BROOKLYN	11212	73.0
2	33	170.0	137.0	BRONX	10470	47.0
4	11	180.0	169.0	MANHATTAN	10019	18.0
4	0	0.0	0.0	BRONX	10458	48.0

```
In [12]: import matplotlib.pyplot as plt

         f = plt.figure(figsize=(19, 15))
         plt.matshow(cleaned.corr(), fignum=f.number)
         plt.xticks(range(cleaned.select_dtypes(['number']).shape[1]), cleaned.select_dtypes(['number']).columns, fontsize=14, r
         plt.yticks(range(cleaned.select_dtypes(['number']).shape[1]), cleaned.select_dtypes(['number']).columns, fontsize=14)
         cb = plt.colorbar()
         cb.ax.tick_params(labelsize=14)
         plt.title('Correlation Matrix', fontsize=16);
```


Correlation Matrix
CAD_INCIDENT_ID
FINAL_SEVERITY_LEVEL_CODE
DISPATCH_RESPONSE_SECONDS_QY
INCIDENT_RESPONSE_SECONDS_QY
INCIDENT_TRAVEL_TM_SECONDS_QY
POLICEPRECINCT