Assignment: 1

TEAM7: SNIGDHA JOSHI & VIPRA SHAH

This assignment is divided into two parts:

1) Data Ingestion
2) Data Wrangling

For both docker images perform following steps:

1) Pull the image from the dockerhub
2) Copy the config.json and configwrangle.json file to your local machine
3) Add your AWS access and secrete keys, create a bucket called "team7pa_assigment1" then run the following codes.
4) Commit changes
5) Execute the code

For detailed description of above steps please refer below:

Data Ingestion

1. Pull image from docker hub

```
Snigdha@DESKTOP-T23DDG5 MINGW64 ~
$ docker pull joshisn/assignment1:final
final: Pulling from joshisn/assignment1
9f0706ba7422: Already exists
d3942a742d22: Already exists
62b1123c88f6: Already exists
2dac6294ef18: Already exists
a7bb658fb099: Already exists
a811de274338: Already exists
771f11f32dc9: Already exists
a16d4d6b543c: Already exists
f0b7951cc55d: Pull complete
ce2bc6ff5564: Pull complete
2784ab0b4421: Pull complete
0c2ba9465c05: Pull complete
eb42dd3fb11d: Pull complete
3c4082260b5e: Pull complete
cc2054b4362f: Pull complete
b07c492e611d: Pull complete
14130aa31fd8: Pull complete
108a648cac70: Pull complete
fb75714b7629: Pull complete
326b4a4ed0ff: Pull complete
18301873a3a8: Pull complete
cf840d1e52af: Pull complete
d6f0d1190f67: Pull complete
Digest: sha256:1efe4239fcf86f4da963abf84922835c68a33e906f5e98d0da37bfb431dfcf1a
Status: Downloaded newer image for joshisn/assignment1:final
```

2. Create the container

```
Snigdha@DESKTOP-T23DDG5 MINGW64 ~/Documents/ADS/Assignment1/Raw_Data (master)
$ docker create --name="rawdata13" joshisn/assignment1:final
1a0eeb55800aae359b25741e286f4c2f39f75727289fe941c86dff88dfc0296f
```

3. Copy config file to your local machine and edit as mentioned below:

```
Snigdha@DESKTOP-T23DDG5 MINGW64 ~/Documents/ADS/Assignment1/Raw_Data (master)
$ docker cp config.json rawdata13:/Assignment1/
```

4. Start the container

```
Snigdha@DESKTOP-T23DDG5 MINGW64 ~/Documents/ADS/Assignment1/Raw_Data (master)
$ docker start -i rawdata13
/Assignment1
1
2017-06-25
```

5. Commit the container to save the changes

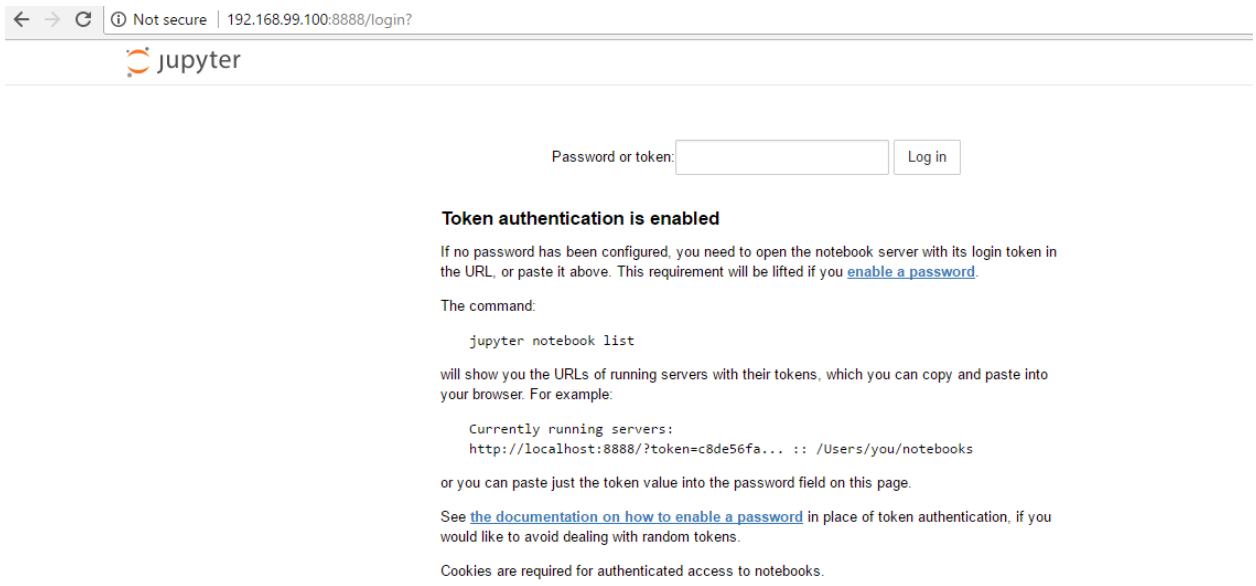
```
Snigdha@DESKTOP-T23DDG5 MINGW64 ~/Documents/ADS/Assignment1/Raw_Data (master)
$ docker commit rawdata13 joshisn/assignment1newimage
sha256:07493232db067285be1d8614ba6c626d1481d0649e05070ebe26baea9abc450e
```

6. Run jupyter notebook

```
Snigdha@DESKTOP-T23DDG5 MINGW64 ~/Documents/ADS/Assignment1/Raw_Data (master)
$ docker run -it -d --name "rawdata_2" -p 8888:8888 joshisn/assignment1newimage /bin/bash -c 'jupyter notebook --no-bro
wser --allow-root --ip=* --NotebookApp.password="$PASSWD" "$@"'
272d75e6db86843baa4373c1a8874b33585d3e78462240531d0b24b0653b8561
```

7. Connect jupyter notebook

http://<yourdocker ip address>:8888



Data Wrangling

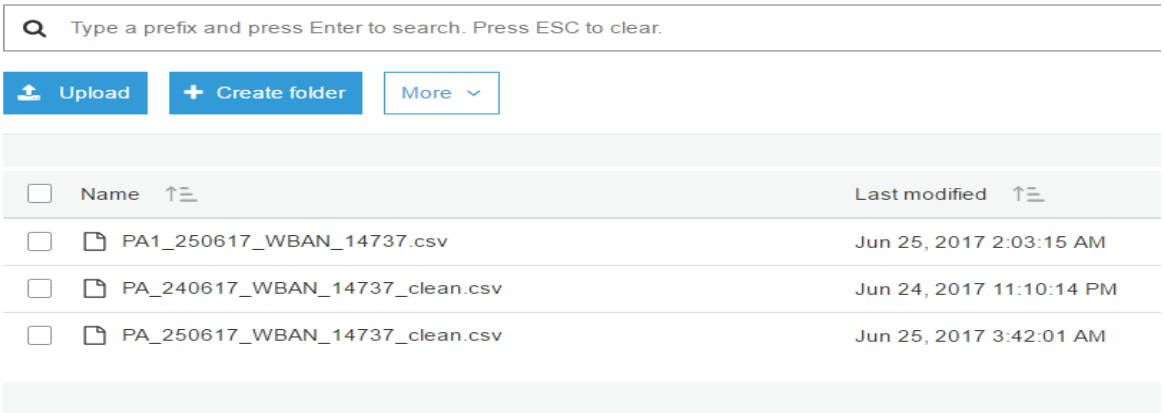
1. Run following commands to pull the image and create container and run wrangle.py file

   Note: before running Create command, copy and edit the configwrangle.json file as mentioned in Data Ingestion part.

```
Snigdha@DESKTOP-T23DDG5 MINGW64 ~/Documents/ADS/Assignment1/Clean_Data (master)
$ docker pull joshisn/assignment1:final
final: Pulling from joshisn/assignment1
Digest: sha256:71464840f073694af87742782c47e0c3087332252a9c8b9df9bff91d9b7b48ea
Status: Image is up to date for joshisn/assignment1:final
```

```
Snigdha@DESKTOP-T23DDG5 MINGW64 ~/Documents/ADS/Assignment1/Clean_Data (master)
$ docker create --name="cleandatacontainer_new" joshisn/assignment1:final
2ef4743b59e70f425a1e758f2e51a55c5c30c6df6883455671dd809a6aec5320

Snigdha@DESKTOP-T23DDG5 MINGW64 ~/Documents/ADS/Assignment1/Clean_Data (master)
$ docker start -i cleandatacontainer_new
```

After performing above command clean.csv file will be uploaded on S3 bucket.



To open jupyter notebook follow command mentioned in Data Ingestion part.

Exploratory Data Analysis

In [31]:
```
import pandas as pd
from pandas import DataFrame
import numpy as np
import matplotlib.ticker as ticker
import matplotlib.pyplot as plt
import os
from datetime import datetime

file_path = os.getcwd() + '/rawdata.csv'
data = pd.read_csv(file_path)
data.describe()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2717: DtypeWarning: Columns (10,11,14,15,20,22,23,2
5,39,40,44,45,59,63,64,67,68,69,84,85,86,87) have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[31]:

	ELEVATION	LATITUDE	LONGITUDE	HOURLYWETBULBTEMPF	HOURLYWETBULBTEMPC	HOURLYRelativeHumidity	HOURLYWin
count	1.235850e+05	1.235850e+05	1.235850e+05	119787.000000	119787.000000	119900.000000	119727.0000(
mean	1.189000e+02	4.065083e+01	-7.544920e+01	48.431115	9.127246	70.875997	6.386103
std	2.474262e-10	1.745100e-11	5.621837e-11	17.094905	9.496896	19.417163	5.084979
min	1.189000e+02	4.065083e+01	-7.544920e+01	-8.000000	-22.100000	12.000000	0.000000
25%	1.189000e+02	4.065083e+01	-7.544920e+01	34.000000	1.300000	56.000000	3.000000
50%	1.189000e+02	4.065083e+01	-7.544920e+01	50.000000	9.800000	74.000000	6.000000
75%	1.189000e+02	4.065083e+01	-7.544920e+01	63.000000	17.400000	88.000000	9.000000
max	1.189000e+02	4.065083e+01	-7.544920e+01	90.000000	32.300000	100.000000	45.000000

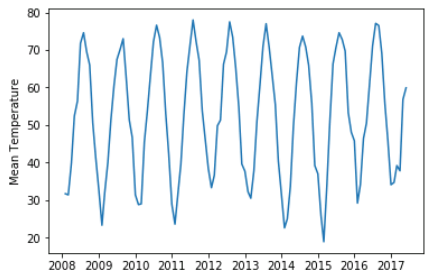
EDA is done on Local Climatologically Data (LCD) set which provides a synopsis of climatic values for a single weather station over a specific month. This EDA is done for PA station.

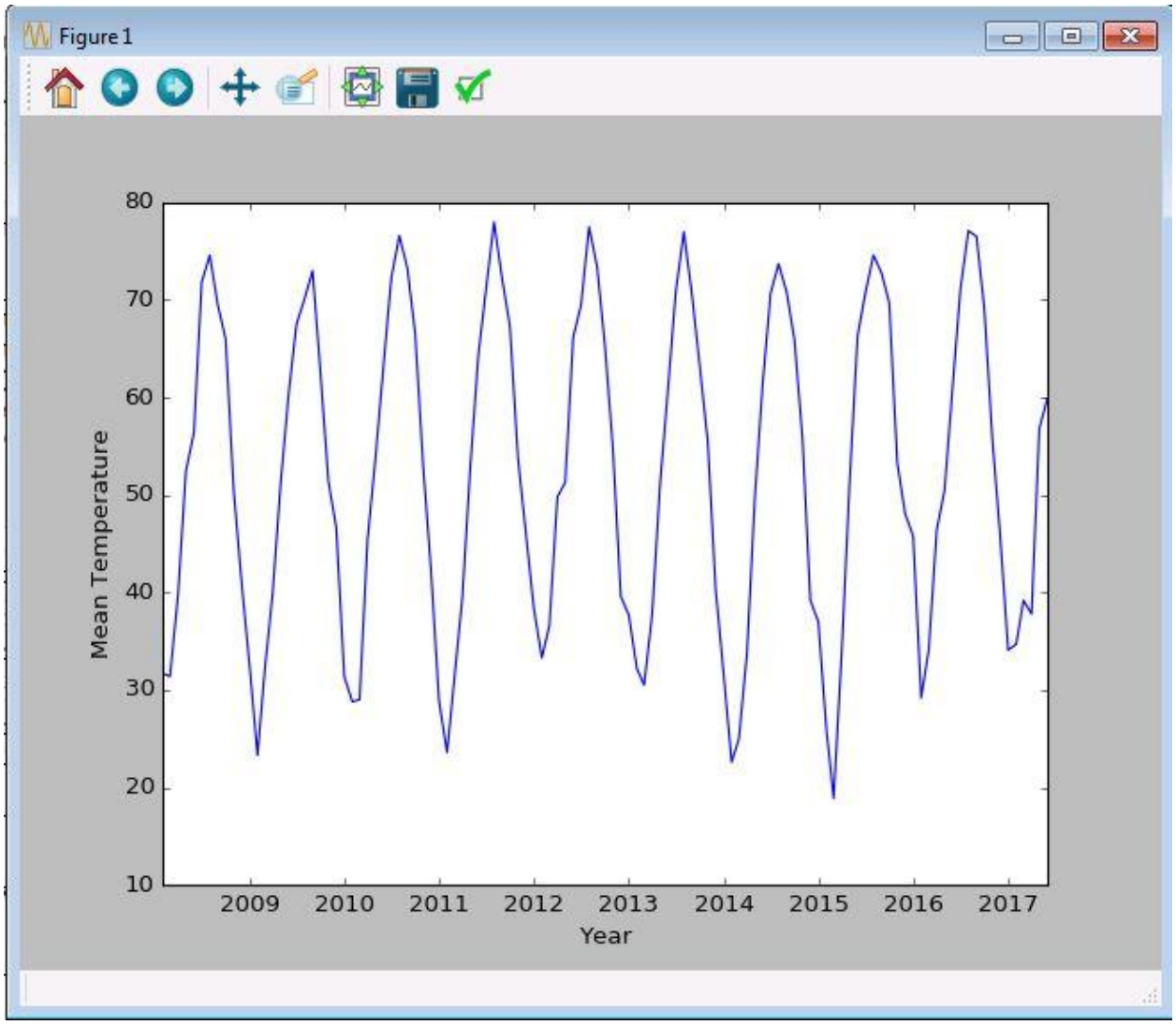
1. Perform EDA on Mean Temperature over the years.

get information on Mean Temperature over the years

In [32]:
```
def plot_mean_temp_monthly(data):

    monthly_mean_temp = data.loc[~np.isnan(data['MonthlyMeanTemp'])]
    x = monthly_mean_temp['DATE'].map(lambda x: datetime.strptime(str(x), '%Y-%m-%d %H:%M'))
    y = monthly_mean_temp['MonthlyMeanTemp']
    plt.plot(x,y)
    plt.xlabel('Year')
    plt.ylabel('Mean Temperature')
    plt.show()
```

In [33]:
```
plot_mean_temp_monthly(data)
```

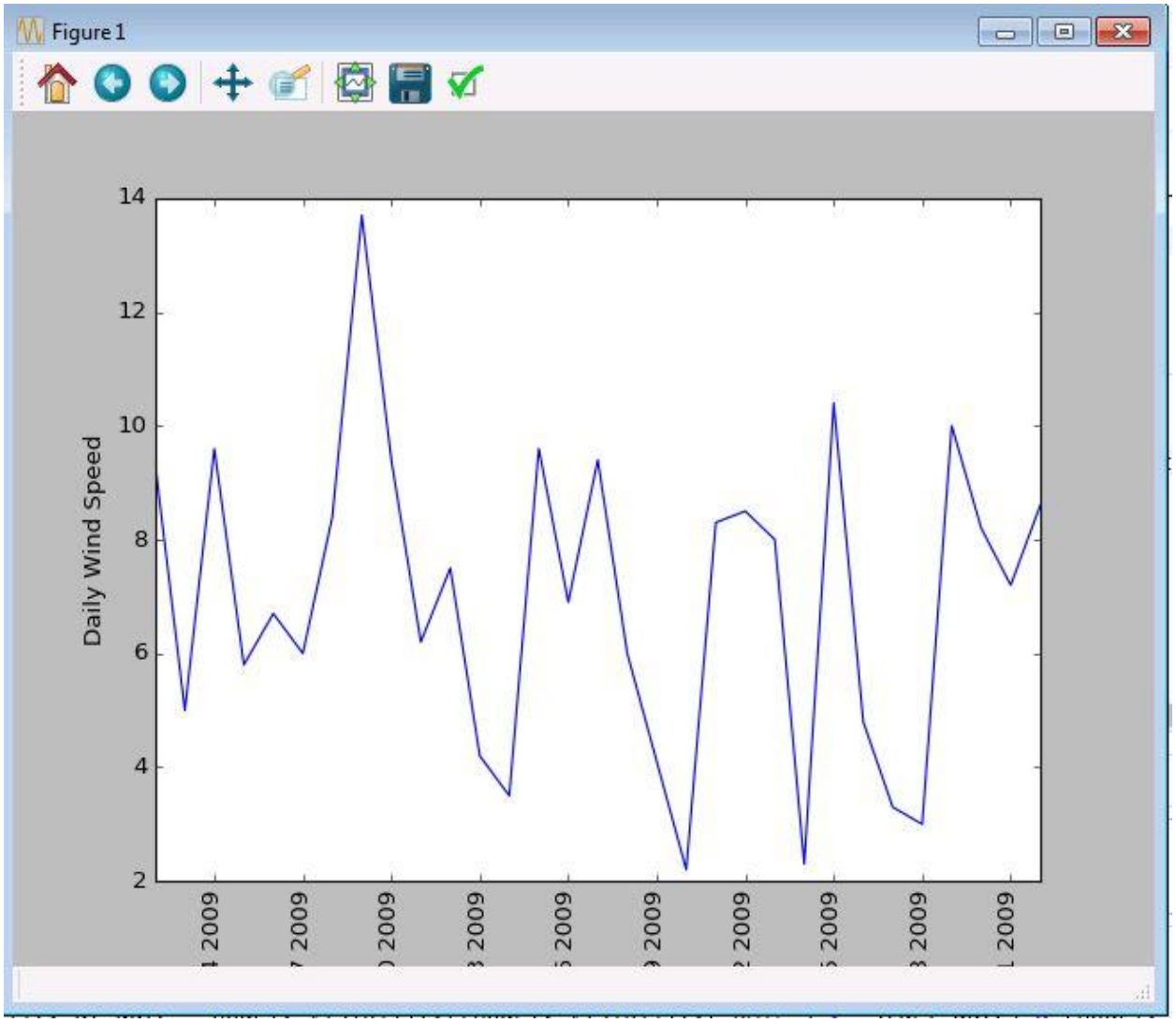




2. Perform EDA on daily wind speed on specific date ranges and plot the graph

Get information on Daily wind speed on specific date ranges

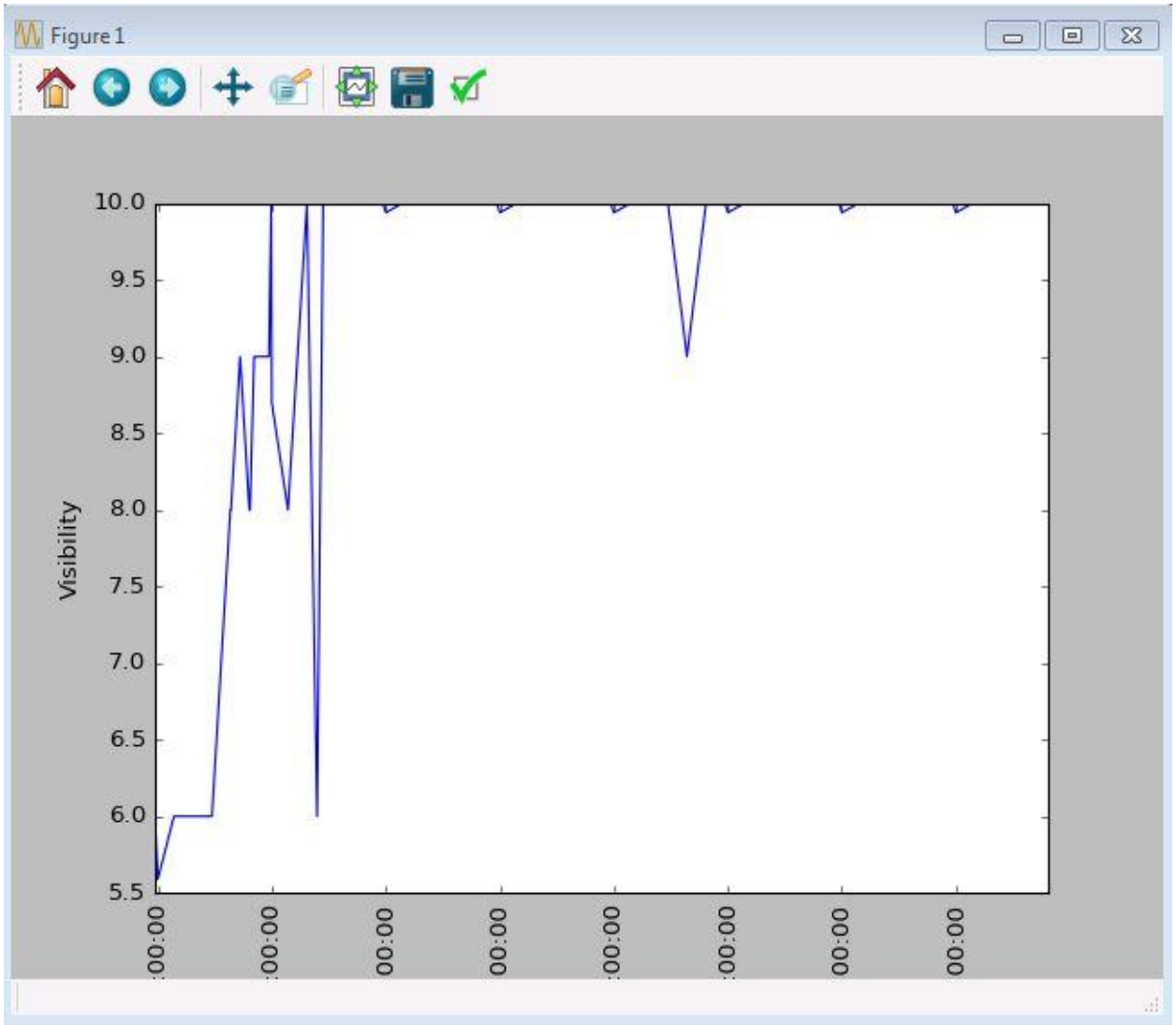
In [11]:
```
def plot_daily_wind_speed_between_ranges(data, start_time, end_time):
    daily_avg_wind_speed = data.loc[~np.isnan(data['DAILYAverageWindSpeed'])]
    daily_avg_wind_speed_between_ranges = daily_avg_wind_speed[(daily_avg_wind_speed['DATE'] > start_time) & (daily_avg_wind_spee
    x = daily_avg_wind_speed_between_ranges['DATE'].map(lambda x: datetime.strptime(str(x), '%Y-%m-%d %H:%M'))
    y = daily_avg_wind_speed_between_ranges['DAILYAverageWindSpeed']
    plt.plot(x,y)
    plt.xticks(rotation=90)
    plt.xlabel('Year')
    plt.ylabel('Daily Wind Speed')
    plt.show()
```



3. Perform EDA on hourlyvisibility column for specific date range and plot the graph

Get information on hourly visibility for specific dates

```
In [13]: def plot_hourly_visibility_for_date(data, start_date, end_date):
             hourly_visibility = data.loc[~pd.isnull(data['HOURLYVISIBILITY'])]
             hourly_visibility_at_date = hourly_visibility[(hourly_visibility['DATE'] >= start_date) & (hourly_visibility['DATE'] < end_da
             time = hourly_visibility_at_date['DATE'].map(lambda x: datetime.strptime(str(x), '%Y-%m-%d %H:%M'))
             visibility = hourly_visibility_at_date['HOURLYVISIBILITY'].astype(float)
             plt.plot(time,visibility)
             plt.xticks(rotation=90)
             plt.xlabel('Time')
             plt.ylabel('Visibility')
             plt.show()
```



4. Perform EDA on hourly dry temperature (F) on specific date range

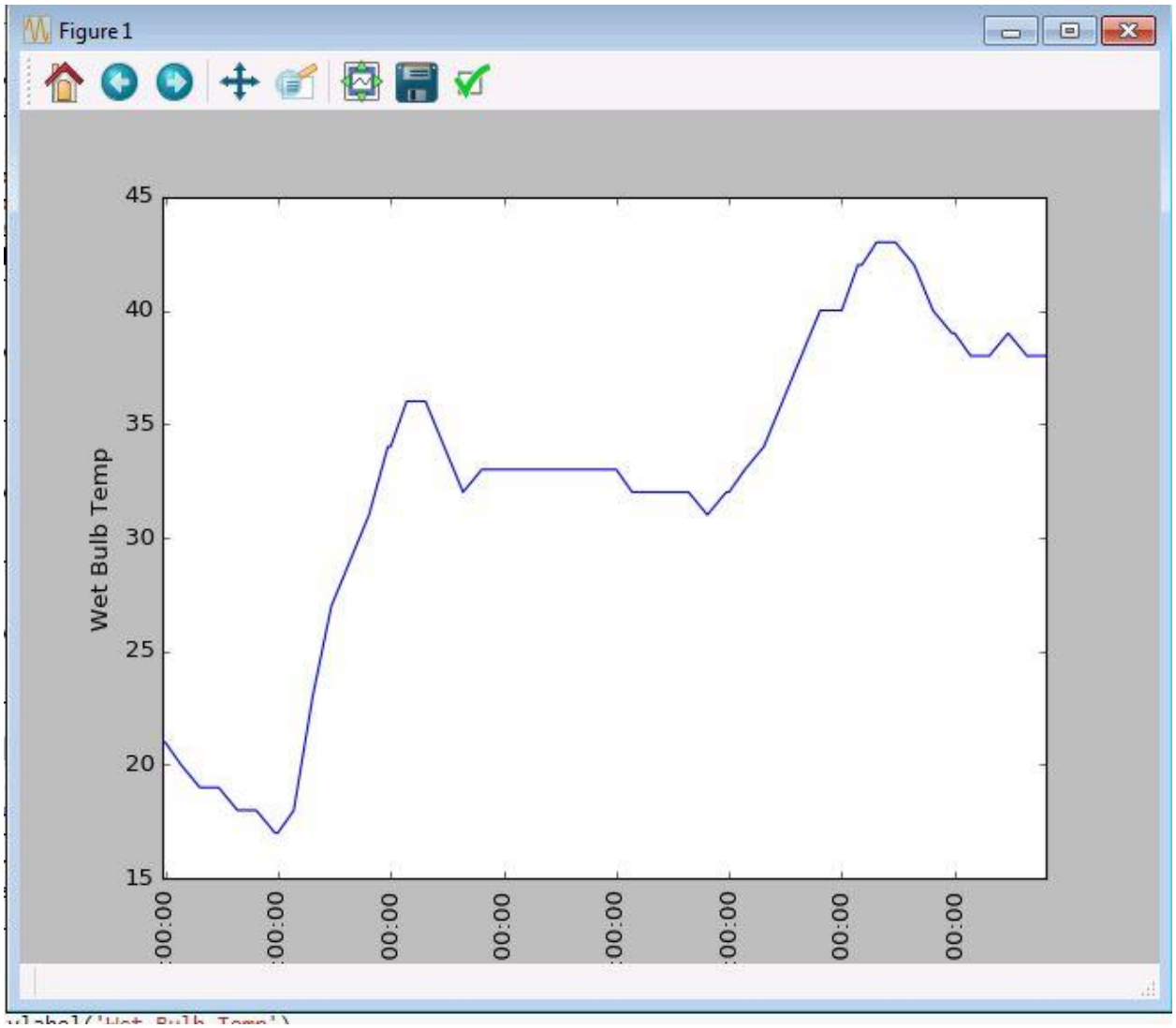
Get information on hourly dry temperature in fahrenheit on specific date range

In [57]:
```
hourly_dry_bulb_temp = data.loc[~pd.isnull(data['HOURLYDRYBULBTEMPF'])]
hourly_dry_bulb_temp[~hourly_dry_bulb_temp['HOURLYDRYBULBTEMPF'].str.contains("s", na=False)]
```

Out[57]:

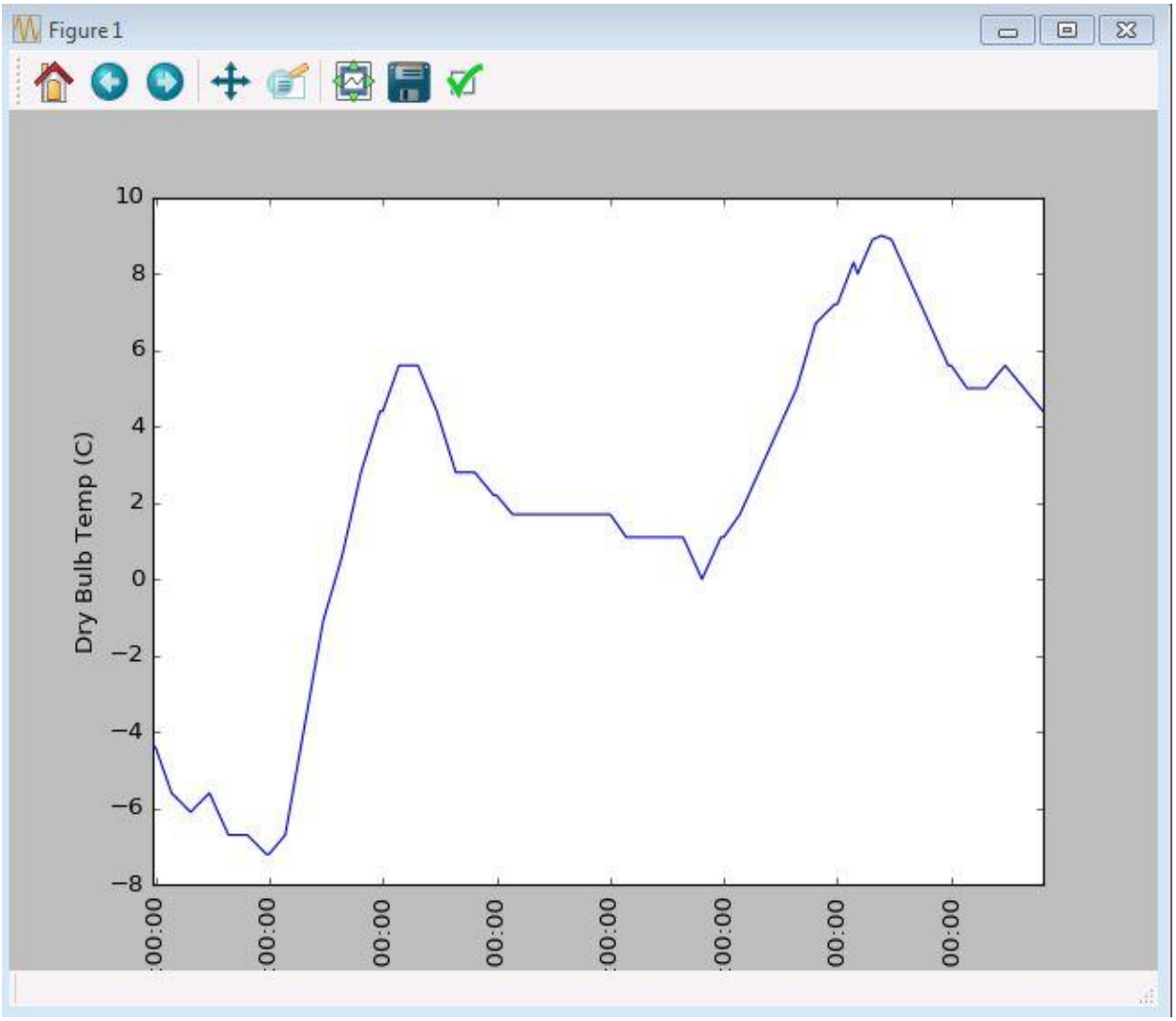
	STATION	STATION_NAME	ELEVATION	LATITUDE	LONGITUDE	DATE	REPORTTPYE	HOURLYSKYCONDITIONS	HOURLYVISIBILI
0	WBAN:14737	ALLENTOWN LEHIGH VALLEY INTERNATIONAL AIRPORT ...	118.9	40.65083	-75.4492	2008-01-01 00:51	FM-15	NaN	6.00
1	WBAN:14737	ALLENTOWN LEHIGH VALLEY INTERNATIONAL AIRPORT ...	118.9	40.65083	-75.4492	2008-01-01 01:00	FM-12	CLR:00	5.59
2	WBAN:14737	ALLENTOWN LEHIGH VALLEY INTERNATIONAL AIRPORT ...	118.9	40.65083	-75.4492	2008-01-01 01:51	FM-15	NaN	6.00
		ALLENTOWN							

In [58]:
```
def plot_hourly_wet_bulb_temp_f(data, start_date, end_date):
        hourly_wet_bulb_temp = data.loc[~pd.isnull(data['HOURLYWETBULBTEMPF'])]
        hourly_dry_bulb_temp_clean_between_dates = hourly_wet_bulb_temp[(hourly_wet_bulb_temp['DATE'] >= start_date) & (hourly_we
        time = hourly_dry_bulb_temp_clean_between_dates['DATE'].map(lambda x: datetime.strptime(str(x), '%Y-%m-%d %H:%M'))
        dry_bulb_temp = hourly_dry_bulb_temp_clean_between_dates['HOURLYWETBULBTEMPF'].astype(float)
        plt.plot(time,dry_bulb_temp)
        plt.xticks(rotation=90)
        plt.xlabel('Time')
        plt.ylabel('Wet Bulb Temp')
        plt.show()
plot_hourly_wet_bulb_temp_f(data, '2008-01-05','2008-01-07')
```



7. Perform EDA on hourly dry temperature Celsius on specific date ranges

```
In [*]: def plot_hourly_dry_bulb_temp_c(data, start_date, end_date):
            hourly_dry_bulb_temp = data.loc[~pd.isnull(data['HOURLYDRYBULBTEMPC'])]
            hourly_dry_bulb_temp_clean = hourly_dry_bulb_temp[~hourly_dry_bulb_temp['HOURLYDRYBULBTEMPC'].str.contains("s", na=False)
            hourly_dry_bulb_temp_clean_between_dates = hourly_dry_bulb_temp_clean[(hourly_dry_bulb_temp_clean['DATE'] >= start_date) &
            time = hourly_dry_bulb_temp_clean_between_dates['DATE'].map(lambda x: datetime.strptime(str(x), '%Y-%m-%d %H:%M'))
            wet_bulb_temp = hourly_dry_bulb_temp_clean_between_dates['HOURLYDRYBULBTEMPC'].astype(float)
            plt.plot(time,wet_bulb_temp)
            plt.xticks(rotation=90)
            plt.xlabel('Time')
            plt.ylabel('Dry Bulb Temp (C)')
            plt.show()
        plot_hourly_dry_bulb_temp_c(data, '2008-01-05','2008-01-07')
```



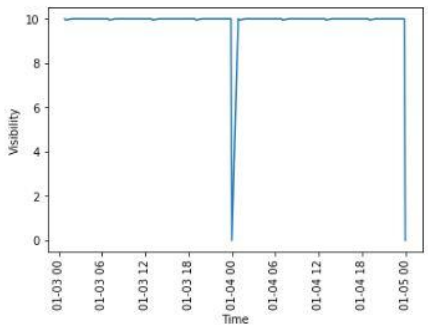
Data Wrangling

```
data.drop(['HOURLYSKYCONDITIONS','HOURLYPRSENTWEATHERTYPE', 'DAILYMaximumDryBulbTemp','DAILYMinimumDryBulbTemp',
'DAILYAverageDryBulbTemp', 'DAILYDeptFromNormalAverageTemp', 'DAILYAverageRelativeHumidity',
'DAILYAverageDewPointTemp','DAILYAverageWetBulbTemp','DAILYHeatingDegreeDays', 'DAILYCoolingDegreeDays', 'DAILYWeather', 'DAILYPrecip',
'DAILYSnowfall','DAILYSnowDepth', 'DAILYAverageStationPressure', 'DAILYAverageSeaLevelPressure', 'DAILYAverageWindSpeed',
'DAILYPeakWindSpeed' , 'PeakWindDirection','DAILYSustainedWindSpeed','DAILYSustainedWindDirection', 'MonthlyMaxSeaLevelPressureDate',
'MonthlyMaxSeaLevelPressureTime', 'MonthlyMinSeaLevelPressureDate', 'MonthlyMinSeaLevelPressureTime', 'MonthlyTotalHeatingDegreeDays',
'MonthlyTotalCoolingDegreeDays', 'MonthlyDeptFromNormalHeatingDD',
'MonthlyDeptFromNormalCoolingDD','MonthlyTotalSeasonToDateHeatingDD','MonthlyTotalSeasonToDateCoolingDD','MonthlyAverageRH','MonthlyDewpoin
tTemp','MonthlyWetBulbTemp','MonthlyAverageRH','MonthlyAvgHeatingDegreeDays','MonthlyAvgCoolingDegreeDays','MonthlyStationPressure','Monthl
ySeaLevelPressure','MonthlyAverageWindSpeed','MonthlyTotalSnowfall','MonthlyDeptFromNormalMaximumTemp','MonthlyDeptFromNormalMinimumTemp','
MonthlyDeptFromNormalAverageTemp','MonthlyDeptFromNormalPrecip','MonthlyTotalLiquidPrecip','MonthlyGreatestPrecip','MonthlyGreatestPrecipDa
te','MonthlyGreatestSnowfall','MonthlyGreatestSnowfallDate','MonthlyGreatestSnowDepth',
'MonthlyGreatestSnowDepthDate','MonthlyDaysWithGT90Temp','MonthlyDaysWithLT32Temp','MonthlyDaysWithGT32Temp','MonthlyDaysWithLT0Temp','Mont
hlyDaysWithGT001Precip','MonthlyDaysWithGT010Precip','MonthlyDaysWithGT1Snow','MonthlyMaxSeaLevelPressureValue','MonthlyMinSeaLevelPressure
Value'],inplace=True,axis=1,errors='ignore')
data.update(data[['HOURLYDRYBULBTEMPF','HOURLYDRYBULBTEMPC','HOURLYWETBULBTEMPF','HOURLYWETBULBTEMPC','HOURLYWindGustSpeed','HOURLYPressure
Tendency', 'HOURLYPressureChange', 'HOURLYVISIBILITY','MonthlyMaximumTemp','MonthlyMinimumTemp','MonthlyMeanTemp']].fillna(0))
```

Exploratory Data Analysis on Clean Data

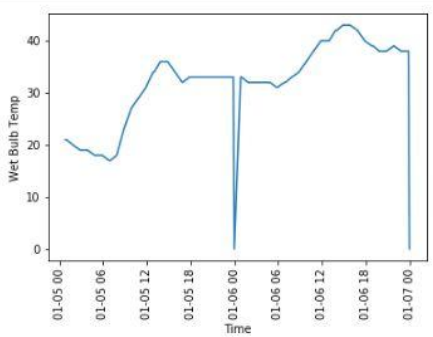
1. Perform EDA on clean data for hourly visibility for specific date range

```
In [87]: def plot_hourly_visibility_for_date(data, start_date, end_date):
             hourly_visibility = data.loc[~pd.isnull(data['HOURLYVISIBILITY'])]
             hourly_visibility_at_date = hourly_visibility[(hourly_visibility['DATE'] >= start_date) & (hourly_visibility['DATE'] < end_da
             time = hourly_visibility_at_date['DATE']
             visibility = hourly_visibility_at_date['HOURLYVISIBILITY'].astype(float)
             plt.plot(time,visibility)
             plt.xticks(rotation=90)
             plt.xlabel('Time')
             plt.ylabel('Visibility')
             plt.show()
         plot_hourly_visibility_for_date(data, '2008-01-03', '2008-01-05')
```



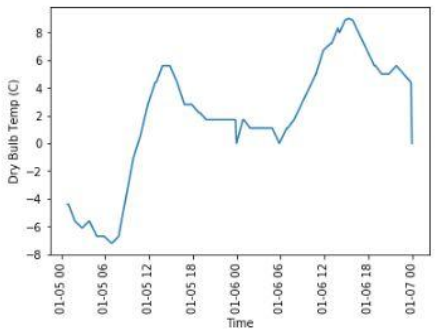
2. Perform EDA on clean data for hourly wet bulb temp for specific date ranges

```
In [89]: def plot_hourly_wet_bulb_temp_f(data, start_date, end_date):
             hourly_wet_bulb_temp = data.loc[~pd.isnull(data['HOURLYWETBULBTEMPF'])]
             hourly_dry_bulb_temp_clean_between_dates = hourly_wet_bulb_temp[(hourly_wet_bulb_temp['DATE'] >= start_date) & (hourly_we
             time = hourly_dry_bulb_temp_clean_between_dates['DATE']
             dry_bulb_temp = hourly_dry_bulb_temp_clean_between_dates['HOURLYWETBULBTEMPF'].astype(float)
             plt.plot(time,dry_bulb_temp)
             plt.xticks(rotation=90)
             plt.xlabel('Time')
             plt.ylabel('Wet Bulb Temp')
             plt.show()
         plot_hourly_wet_bulb_temp_f(data, '2008-01-05','2008-01-07')
```



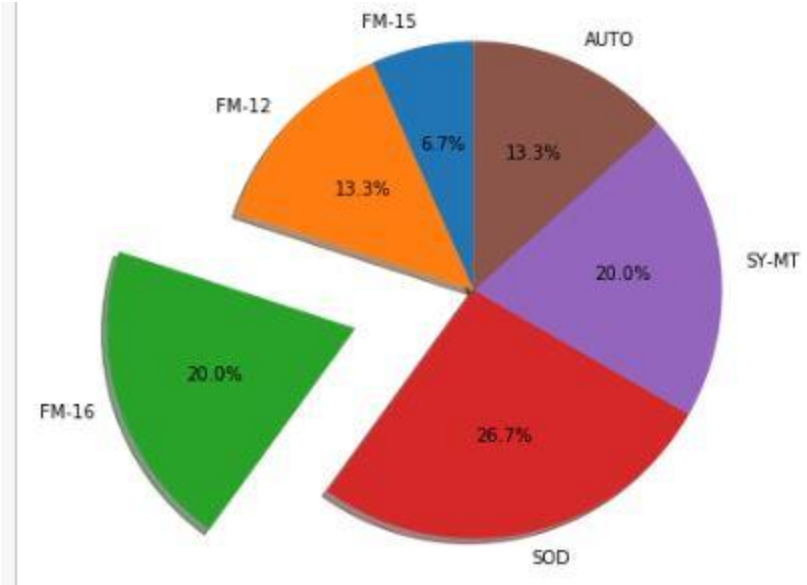
3. Perform EDA on clean data for hourly dry bulb temp celcius for specific date ranges

```
In [90]: def plot_hourly_dry_bulb_temp_c(data, start_date, end_date):
             hourly_dry_bulb_temp = data.loc[~pd.isnull(data['HOURLYDRYBULBTEMPC'])]
             #hourly_dry_bulb_temp_clean = hourly_dry_bulb_temp[~hourly_dry_bulb_temp['HOURLYDRYBULBTEMPC'].str.contains("s", na=False
             hourly_dry_bulb_temp_clean_between_dates = hourly_dry_bulb_temp[(hourly_dry_bulb_temp['DATE'] >= start_date) & (hourly_dr
             time = hourly_dry_bulb_temp_clean_between_dates['DATE']
             wet_bulb_temp = hourly_dry_bulb_temp_clean_between_dates['HOURLYDRYBULBTEMPC'].astype(float)
             plt.plot(time,wet_bulb_temp)
             plt.xticks(rotation=90)
             plt.xlabel('Time')
             plt.ylabel('Dry Bulb Temp (C)')
             plt.show()
         plot_hourly_dry_bulb_temp_c(data, '2008-01-05','2008-01-07')
```



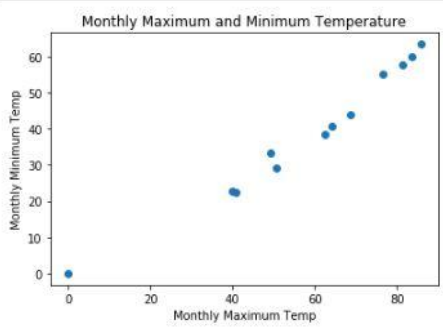
4. Plot pie chart for reporttype count

```
In [92]: figure(1, figsize=(6,6))
         ax = axes([0.1, 0.1, 0.8, 0.8])

         # The slices will be ordered and plotted counter-clockwise.
         labels = 'FM-15', 'FM-12', 'FM-16','SOD','SY-MT','AUTO'
         sizes = [10,20,30,40,30,20]
         explode=(0, 0, 0.5, 0,0,0)

         pie(sizes, explode=explode, labels=labels,
                         autopct='%1.1f%%', shadow=True, startangle=90)
         title('Reporttype', bbox={'facecolor':'0.7', 'pad':5})

         show()
```



5. Perform EDA on monthly minimum and maximum temperature and plot scatter graph

```
In [96]: plt.scatter(max[:12],min[:12])
         plt.ylabel('Monthly Minimum Temp')
         plt.xlabel('Monthly Maximum Temp')
         plt.title("Monthly Maximum and Minimum Temperature")

         plt.show()
```



6. Perform EDA on mean of monthly maximum and minimum temperature and plot box plot

In [100]:
```
fig = plt.figure(1, figsize=(9, 6))
ax = fig.add_subplot(111)
bp = ax.boxplot(data_plot)
ax.set_title('Comparison of Minimum and Maxiumum Mean Temperature')
ax.set_xlabel('Distribution')
ax.set_ylabel('Value')
plt.show()
```