# Analyzing Police activity and weather conditions using Pandas

March 29, 2020

### 0.0.1 Analyzing Police activity and weather conditions using Pandas

This is one of my pieces of work towards my endeavour to learn Pandas. It encompases a wide range of functions used in Pandas.

Below analysis gives an in depth information on Police activity in different states of the US. The results can be used to find out the range of road violations taking place in different places at different times and during different weather.

The results can be used by police develop strategies to reduce violations.

Data is studied from various angles, taking into consideration whole dataset as one unit, analyzing variables individually and relationship between different variables.

Data is distributed row wise, column wise to find interesting insights and to make inferences.

Different data frames are also merged to conduct a more comprehensive analysis.

Pandas is a great tool for data pre-processing which is the most important aspect of data analysis.

Among other the Pandas skills I am learning will definitely help me to analyze data for different sectors.

[184]:
```
import pandas as pd
```

[60]:
```
ri=pd.read_csv('Traffic stops in Rhode Island.csv')
print(ri.head())
```

```
  state   stop_date stop_time  county_name driver_gender driver_race  \
0    RI  2005-01-04     12:55          NaN             M       White
1    RI  2005-01-23     23:15          NaN             M       White
2    RI  2005-02-17      4:15          NaN             M       White
3    RI  2005-02-20     17:15          NaN             M       White
4    RI  2005-02-24      1:20          NaN             F       White

                    violation_raw  violation  search_conducted search_type  \
0  Equipment/Inspection Violation  Equipment             False         NaN
1                        Speeding   Speeding             False         NaN
2                        Speeding   Speeding             False         NaN
3                Call for Service      Other             False         NaN
4                        Speeding   Speeding             False         NaN
```

```
    stop_outcome is_arrested stop_duration drugs_related_stop district
0       Citation       False      0-15 Min              False  Zone X4
1       Citation       False      0-15 Min              False  Zone K3
2       Citation       False      0-15 Min              False  Zone X4
3  Arrest Driver        True     16-30 Min              False  Zone X1
4       Citation       False      0-15 Min              False  Zone X3
```

[61]:
```
# Counting the number of missing values in each column
print(ri.isnull().sum())
```

```
state                   0
stop_date               0
stop_time               0
county_name         67027
driver_gender        4069
driver_race          4067
violation_raw        4067
violation            4067
search_conducted        0
search_type         64430
stop_outcome         4067
is_arrested          4067
stop_duration        4068
drugs_related_stop      1
district                1
dtype: int64
```

[62]:
```
# Examining the shape of the DataFrame
print(ri.shape)
```

```
(67027, 15)
```

The data contains 67027 rows and 15 columns.

[63]:
```
# Dropping the 'county_name' and 'state' columns
ri.drop(['county_name', 'state'], axis='columns', inplace=True)
```

[64]:
```
# Examining the shape of the DataFrame (again)
print(ri.shape)
```

```
(67027, 13)
```

[65]:
```
# Counting the number of missing values in each column
print(ri.isnull().sum())
```

```
stop_date               0
stop_time               0
driver_gender        4069
driver_race          4067
```

```
violation_raw          4067
violation              4067
search_conducted          0
search_type           64430
stop_outcome           4067
is_arrested            4067
stop_duration          4068
drugs_related_stop        1
district                  1
dtype: int64
```

[66]:
```
# Dropping all rows that are missing 'driver_gender'
ri.dropna(subset=['driver_gender'], inplace=True)
```

Driver gender cannot be infered from the data, hence deleted the missing values.

[67]:
```
# Counting the number of missing values in each column (again)
print(ri.isnull().sum())
```

```
stop_date                 0
stop_time                 0
driver_gender             0
driver_race               0
violation_raw             0
violation                 0
search_conducted          0
search_type           60361
stop_outcome              0
is_arrested               0
stop_duration             1
drugs_related_stop        1
district                  1
dtype: int64
```

[68]:
```
# Examine the shape of the DataFrame
print(ri.shape)
```

```
(62958, 13)
```

[69]:
```
# Examining the head of the 'is_arrested' column
print(ri.is_arrested.head())
```

```
0    False
1    False
2    False
3     True
4    False
Name: is_arrested, dtype: object
```

[70]:
```
# Change the data type of 'is_arrested' to 'bool'
ri['is_arrested'] = ri.is_arrested.astype(bool)
```

[71]:
```
# Check the data type of 'is_arrested'
print(ri.is_arrested.dtype)
```

```
bool
```

[72]:
```
# Concatenating 'stop_date' and 'stop_time' (separated by a space)
combined = ri.stop_date.str.cat(ri.stop_time, sep=' ')
```

[73]:
```
# Converting 'combined' to datetime format
ri['stop_datetime'] = pd.to_datetime(combined)
```

[74]:
```
# Examine the data types of the DataFrame
print(ri.dtypes)
```

```
stop_date                     object
stop_time                     object
driver_gender                 object
driver_race                   object
violation_raw                 object
violation                     object
search_conducted                bool
search_type                   object
stop_outcome                  object
is_arrested                     bool
stop_duration                 object
drugs_related_stop            object
district                      object
stop_datetime         datetime64[ns]
dtype: object
```

[75]:
```
# Setting 'stop_datetime' as the index
ri.set_index('stop_datetime', inplace=True)

# Examining the index
print(ri.index)
```

```
DatetimeIndex(['2005-01-04 12:55:00', '2005-01-23 23:15:00',
               '2005-02-17 04:15:00', '2005-02-20 17:15:00',
               '2005-02-24 01:20:00', '2005-03-14 10:00:00',
               '2005-03-29 21:55:00', '2005-04-04 21:25:00',
               '2005-07-14 11:20:00', '2005-07-14 19:55:00',
               …
               '2013-02-01 15:02:00', '2013-02-01 20:48:00',
               '2013-02-01 21:18:00', '2013-02-01 22:39:00',
               '2013-02-01 22:50:00', '2013-02-02 01:59:00',
```

```
               '2013-02-02 02:22:00', '2013-02-02 06:18:00',
               '2013-02-02 07:01:00', '2013-02-02 08:28:00'],
              dtype='datetime64[ns]', name='stop_datetime', length=62958,
freq=None)
```

The records pertaining to stop time can be used by Police to find out the range of time and violations. Police can use the data to be more vigilant during the hours of more violations, eventually bringing more discipline in the people.

[76]:
```
# Examining the columns
print(ri.columns)
```

```
Index(['stop_date', 'stop_time', 'driver_gender', 'driver_race',
       'violation_raw', 'violation', 'search_conducted', 'search_type',
       'stop_outcome', 'is_arrested', 'stop_duration', 'drugs_related_stop',
       'district'],
      dtype='object')
```

[77]:
```
# Counting the unique values in 'violation'
print(ri.violation.value_counts())
```

```
Speeding               38015
Moving violation       12120
Equipment               6697
Other                   3675
Registration/plates     2392
Seat belt                 59
Name: violation, dtype: int64
```

[78]:
```
# Express the counts as proportions
print(ri.violation.value_counts(normalize=True))
```

```
Speeding               0.603815
Moving violation       0.192509
Equipment              0.106373
Other                  0.058372
Registration/plates    0.037994
Seat belt              0.000937
Name: violation, dtype: float64
```

Number and percentage of violations.

[79]:
```
# Creating a DataFrame of female drivers
female = ri[ri.driver_gender == 'F']
```

[80]:
```
# Create a DataFrame of male drivers
male = ri[ri.driver_gender == 'M']
```

[81]:
```
# Computing the violations by female drivers (as proportions)
print(female.violation.value_counts(normalize=True))
```

```
Speeding               0.712089
Moving violation       0.133665
Equipment              0.086172
Registration/plates    0.038680
Other                  0.028685
Seat belt              0.000710
Name: violation, dtype: float64
```

[82]:
```
# Compute the violations by male drivers (as proportions)
print(male.violation.value_counts(normalize = True))
```

```
Speeding               0.564061
Moving violation       0.214115
Equipment              0.113789
Other                  0.069273
Registration/plates    0.037742
Seat belt              0.001021
Name: violation, dtype: float64
```

Interestingly, higher percentage of women are stopped for speeding than men.

[83]:
```
# Creating a DataFrame of female drivers stopped for speeding
female_and_speeding = ri[(ri.driver_gender == 'F') & (ri.violation ==␣
↪'Speeding')]
print(female_and_speeding)
```

```
                     stop_date stop_time driver_gender driver_race  \
stop_datetime
2005-02-24 01:20:00  2005-02-24      1:20             F       White
2005-03-14 10:00:00  2005-03-14     10:00             F       White
2005-07-14 11:20:00  2005-07-14     11:20             F       White
2005-07-18 19:30:00  2005-07-18     19:30             F       White
2005-07-24 20:10:00  2005-07-24     20:10             F       White
…                           …         …             …         …
2013-02-01 06:59:00  2013-02-01      6:59             F       White
2013-02-01 09:20:00  2013-02-01      9:20             F       White
2013-02-01 09:21:00  2013-02-01      9:21             F       White
2013-02-01 09:50:00  2013-02-01      9:50             F       White
2013-02-01 10:32:00  2013-02-01     10:32             F       White

                    violation_raw violation  search_conducted search_type  \
stop_datetime
2005-02-24 01:20:00      Speeding  Speeding             False         NaN
2005-03-14 10:00:00      Speeding  Speeding             False         NaN
2005-07-14 11:20:00      Speeding  Speeding             False         NaN
```

```
2005-07-18 19:30:00      Speeding  Speeding             False         NaN
2005-07-24 20:10:00      Speeding  Speeding             False         NaN
…                           …       …                 …          …
2013-02-01 06:59:00      Speeding  Speeding             False         NaN
2013-02-01 09:20:00      Speeding  Speeding             False         NaN
2013-02-01 09:21:00      Speeding  Speeding             False         NaN
2013-02-01 09:50:00      Speeding  Speeding             False         NaN
2013-02-01 10:32:00      Speeding  Speeding             False         NaN

                    stop_outcome  is_arrested stop_duration  \
stop_datetime
2005-02-24 01:20:00     Citation        False      0-15 Min
2005-03-14 10:00:00     Citation        False      0-15 Min
2005-07-14 11:20:00     Citation        False      0-15 Min
2005-07-18 19:30:00     Citation        False      0-15 Min
2005-07-24 20:10:00     Citation        False      0-15 Min
…                          …          …           …
2013-02-01 06:59:00     Citation         True      0-15 Min
2013-02-01 09:20:00     Citation         True      0-15 Min
2013-02-01 09:21:00     Citation         True     16-30 Min
2013-02-01 09:50:00     Citation         True      0-15 Min
2013-02-01 10:32:00     Citation         True      0-15 Min

                    drugs_related_stop district
stop_datetime
2005-02-24 01:20:00              False  Zone X3
2005-03-14 10:00:00              False  Zone K3
2005-07-14 11:20:00              False  Zone X4
2005-07-18 19:30:00              False  Zone K3
2005-07-24 20:10:00              False  Zone K3
…                                …      …
2013-02-01 06:59:00              False  Zone X4
2013-02-01 09:20:00              False  Zone X4
2013-02-01 09:21:00              False  Zone X4
2013-02-01 09:50:00              False  Zone X4
2013-02-01 10:32:00              False  Zone K3

[12040 rows x 13 columns]
```

[84]:
```
# Creating a DataFrame of male drivers stopped for speeding
male_and_speeding = ri[(ri.driver_gender == 'M') & (ri.violation == 'Speeding')]
print(male_and_speeding)
```

```
                     stop_date stop_time driver_gender driver_race  \
stop_datetime
2005-01-23 23:15:00  2005-01-23     23:15             M       White
2005-02-17 04:15:00  2005-02-17      4:15             M       White
2005-03-29 21:55:00  2005-03-29     21:55             M       White
```

```
2005-04-04 21:25:00  2005-04-04     21:25             M       White
2005-07-14 19:55:00  2005-07-14     19:55             M       White
…                          …       …           …         …
2013-02-01 09:49:00  2013-02-01      9:49             M       Black
2013-02-01 10:05:00  2013-02-01     10:05             M       White
2013-02-01 21:18:00  2013-02-01     21:18             M       Asian
2013-02-02 06:18:00  2013-02-02      6:18             M       White
2013-02-02 08:28:00  2013-02-02      8:28             M       Black

                    violation_raw violation  search_conducted search_type  \
stop_datetime
2005-01-23 23:15:00      Speeding  Speeding             False         NaN
2005-02-17 04:15:00      Speeding  Speeding             False         NaN
2005-03-29 21:55:00      Speeding  Speeding             False         NaN
2005-04-04 21:25:00      Speeding  Speeding             False         NaN
2005-07-14 19:55:00      Speeding  Speeding             False         NaN
…                           …       …                …         …
2013-02-01 09:49:00      Speeding  Speeding             False         NaN
2013-02-01 10:05:00      Speeding  Speeding             False         NaN
2013-02-01 21:18:00      Speeding  Speeding             False         NaN
2013-02-02 06:18:00      Speeding  Speeding             False         NaN
2013-02-02 08:28:00      Speeding  Speeding             False         NaN

                    stop_outcome  is_arrested stop_duration  \
stop_datetime
2005-01-23 23:15:00     Citation        False      0-15 Min
2005-02-17 04:15:00     Citation        False      0-15 Min
2005-03-29 21:55:00     Citation        False      0-15 Min
2005-04-04 21:25:00     Citation        False      0-15 Min
2005-07-14 19:55:00     Citation        False      0-15 Min
…                          …          …            …
2013-02-01 09:49:00     Citation         True      0-15 Min
2013-02-01 10:05:00     Citation         True      0-15 Min
2013-02-01 21:18:00     Citation         True      0-15 Min
2013-02-02 06:18:00     Citation         True      0-15 Min
2013-02-02 08:28:00      Warning         True           NaN

                    drugs_related_stop district
stop_datetime
2005-01-23 23:15:00              False  Zone K3
2005-02-17 04:15:00              False  Zone X4
2005-03-29 21:55:00              False  Zone K3
2005-04-04 21:25:00              False  Zone K1
2005-07-14 19:55:00              False  Zone X4
…                                …      …
2013-02-01 09:49:00              False  Zone X4
2013-02-01 10:05:00              False  Zone K2
2013-02-01 21:18:00              False  Zone X3
```

```
2013-02-02 06:18:00              False  Zone X3
2013-02-02 08:28:00                NaN      NaN

[25975 rows x 13 columns]
```

[85]:
```
# Computing the stop outcomes for female drivers (as proportions)
print(female_and_speeding.stop_outcome.value_counts(normalize=True))
```

```
Citation            0.971096
Warning             0.020349
Arrest Driver       0.006561
N/D                 0.001163
Arrest Passenger    0.000581
No Action           0.000249
Name: stop_outcome, dtype: float64
```

[86]:
```
# Computing the stop outcomes for male drivers (as proportions)
print(male_and_speeding.stop_outcome.value_counts(normalize=True))
```

```
Citation            0.958961
Warning             0.019711
Arrest Driver       0.017979
Arrest Passenger    0.001270
N/D                 0.001078
No Action           0.001001
Name: stop_outcome, dtype: float64
```

Arrest rate among women is much less than that of men.

[87]:
```
# Calculating the search rate for female drivers
print(ri[(ri.driver_gender == 'F')].search_conducted.mean())
```

```
0.020995978235154956
```

[88]:
```
# Calculating the search rate for male drivers
print(ri[ri.driver_gender == 'M'].search_conducted.mean())
```

```
0.04868621064060803
```

[89]:
```
# Calculating the search rate for both groups simultaneously
print(ri.groupby('driver_gender').search_conducted.mean())
```

```
driver_gender
F    0.020996
M    0.048686
Name: search_conducted, dtype: float64
```

Male drivers are searched more than twice as often as female drivers.

[91]:
```
# Calculating the search rate for each combination of gender and violation
print(ri.groupby(['driver_gender', 'violation']).search_conducted.mean())
```

```
driver_gender  violation
F              Equipment              0.056966
               Moving violation       0.039823
               Other                  0.047423
               Registration/plates    0.073394
               Seat belt              0.083333
               Speeding               0.009136
M              Equipment              0.089695
               Moving violation       0.061663
               Other                  0.044828
               Registration/plates    0.138090
               Seat belt              0.021277
               Speeding               0.030029
Name: search_conducted, dtype: float64
```

[92]:
```
# Reversing the ordering to group by violation before gender
print(ri.groupby(['violation', 'driver_gender']).search_conducted.mean())
```

```
violation            driver_gender
Equipment            F                0.056966
                     M                0.089695
Moving violation     F                0.039823
                     M                0.061663
Other                F                0.047423
                     M                0.044828
Registration/plates  F                0.073394
                     M                0.138090
Seat belt            F                0.083333
                     M                0.021277
Speeding             F                0.009136
                     M                0.030029
Name: search_conducted, dtype: float64
```

Search rate on different violations with respect to men and women can be compared side by side.

For all types of violations, the search rate is higher for males than for females.

[94]:
```
# Total number of searches conducted
ri.search_conducted.value_counts()
```

[94]:
```
False    60361
True      2597
Name: search_conducted, dtype: int64
```

[95]:
```
# Count the 'search_type' values
print(ri.search_type.value_counts())
```

```
Incident to Arrest                                          1106
Probable Cause                                               623
Reasonable Suspicion                                         177
Inventory                                                    165
Protective Frisk                                             136
Incident to Arrest,Inventory                                 102
Incident to Arrest,Probable Cause                             76
Incident to Arrest,Protective Frisk                           32
Probable Cause,Reasonable Suspicion                           28
Probable Cause,Protective Frisk                               26
Incident to Arrest,Inventory,Probable Cause                   26
Incident to Arrest,Inventory,Protective Frisk                 18
Protective Frisk,Reasonable Suspicion                         17
Inventory,Probable Cause                                      15
Inventory,Protective Frisk                                    12
Incident to Arrest,Probable Cause,Protective Frisk            11
Incident to Arrest,Reasonable Suspicion                        8
Probable Cause,Protective Frisk,Reasonable Suspicion           5
Incident to Arrest,Probable Cause,Reasonable Suspicion         4
Incident to Arrest,Inventory,Reasonable Suspicion              3
Inventory,Reasonable Suspicion                                 2
Incident to Arrest,Protective Frisk,Reasonable Suspicion       2
Inventory,Probable Cause,Protective Frisk                      1
Inventory,Protective Frisk,Reasonable Suspicion                1
Inventory,Probable Cause,Reasonable Suspicion                  1
Name: search_type, dtype: int64
```

[96]:
```
# Checking if 'search_type' contains the string 'Protective Frisk'
ri['frisk'] = ri.search_type.str.contains('Protective Frisk', na=False)
```

[97]:
```
# Checking the data type of 'frisk'
print(ri.frisk.dtype)
```

```
bool
```

[98]:
```
# Taking the sum of 'frisk'
print(ri.frisk.sum())
```

```
261
```

It looks like there were 261 drivers who were frisked.

[186]:
```
# Creating a DataFrame of stops in which a search was conducted
searched = ri[ri.search_conducted == True]
print(searched.head())
```

```
searched.shape
```

```
     index state   stop_date stop_time  county_name driver_gender driver_race  \
46      46    RI  2005-10-01     22:00          NaN             M       White
61      61    RI  2005-10-02      9:30          NaN             M       White
94      94    RI  2005-10-03     15:40          NaN             M       Black
120    120    RI  2005-10-05      3:00          NaN             F       White
137    137    RI  2005-10-05     22:50          NaN             M       Black

                     violation_raw         violation  search_conducted  \
46          Other Traffic Violation  Moving violation              True
61                         Speeding          Speeding              True
94   Equipment/Inspection Violation         Equipment              True
120  Equipment/Inspection Violation         Equipment              True
137                        Speeding          Speeding              True

                          search_type   stop_outcome  is_arrested  \
46                     Probable Cause       Citation        False
61                   Incident to Arrest  Arrest Driver         True
94                   Incident to Arrest  Arrest Driver         True
120                  Incident to Arrest  Arrest Driver         True
137  Incident to Arrest,Probable Cause       Citation        False

    stop_duration drugs_related_stop district
46        30+ Min              False  Zone K3
61        30+ Min              False  Zone K1
94        30+ Min               True  Zone X4
120     16-30 Min               True  Zone K2
137       30+ Min              False  Zone K1
```

[186]:
```
(2597, 16)
```

There are 2597 stop incidents when search was conducted.

[101]:
```
# Calculating the overall frisk rate by taking the mean of 'frisk'
print(searched.frisk.mean())
```

```
0.10050057758952638
```

Frisk rate is 10 percent.

[103]:
```
# Calculating the frisk rate for each gender
print(searched.groupby('driver_gender').frisk.mean())
```

```
driver_gender
F    0.078873
M    0.103925
Name: frisk, dtype: float64
```

The frisk rate is higher for males than for females, though it can't be conclude that this difference is caused by the driver's gender.

[105]:
```
# Calculating the overall arrest rate
print(ri.is_arrested.mean())
```

```
0.05916642841259252
```

[106]:
```
# Calculating the hourly arrest rate
print(ri.groupby(ri.index.hour).is_arrested.mean())
```

```
stop_datetime
0     0.071266
1     0.092196
2     0.086732
3     0.077510
4     0.050228
5     0.032258
6     0.040595
7     0.035854
8     0.042223
9     0.049490
10    0.052892
11    0.052969
12    0.061991
13    0.058392
14    0.058701
15    0.054746
16    0.055534
17    0.055408
18    0.055797
19    0.053672
20    0.058902
21    0.086432
22    0.081869
23    0.070566
Name: is_arrested, dtype: float64
```

[107]:
```
# Saving the hourly arrest rate
hourly_arrest_rate = ri.groupby(ri.index.hour).is_arrested.mean()
```

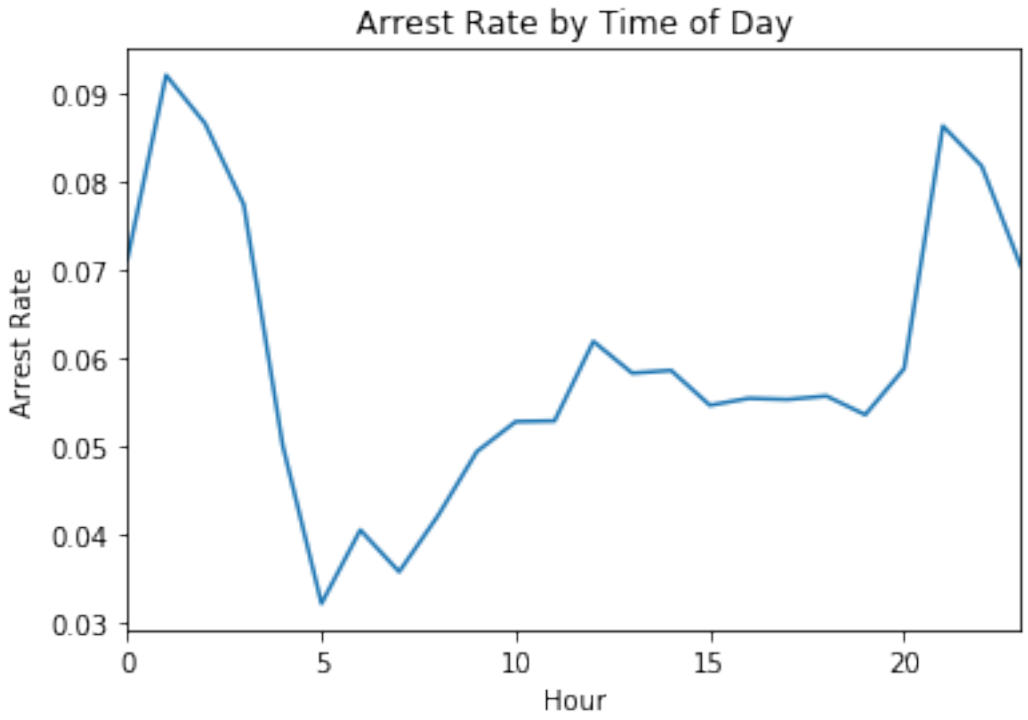
[110]:
```
# Importing matplotlib.pyplot as plt
import matplotlib.pyplot as plt

# Creating a line plot of 'hourly_arrest_rate'
hourly_arrest_rate.plot()

# Adding the xlabel, ylabel, and title
```

```
plt.xlabel('Hour')
plt.ylabel('Arrest Rate')
plt.title('Arrest Rate by Time of Day')

# Displaying the plot
plt.show()
```




The arrest rate has a significant spike overnight, and then dips in the early morning hours.

[112]:
```
# Changing the data type to bool
ri['drugs_related_stop'] = ri.drugs_related_stop.astype(bool)
ri.drugs_related_stop.dtype
```

[112]:
```
dtype('bool')
```

[113]:
```
# Calculating the annual rate of drug-related stops
print(ri.drugs_related_stop.resample('A').mean())
```

```
stop_datetime
2005-12-31    0.006501
2006-12-31    0.007258
2007-12-31    0.007970
2008-12-31    0.007505
2009-12-31    0.009889
```

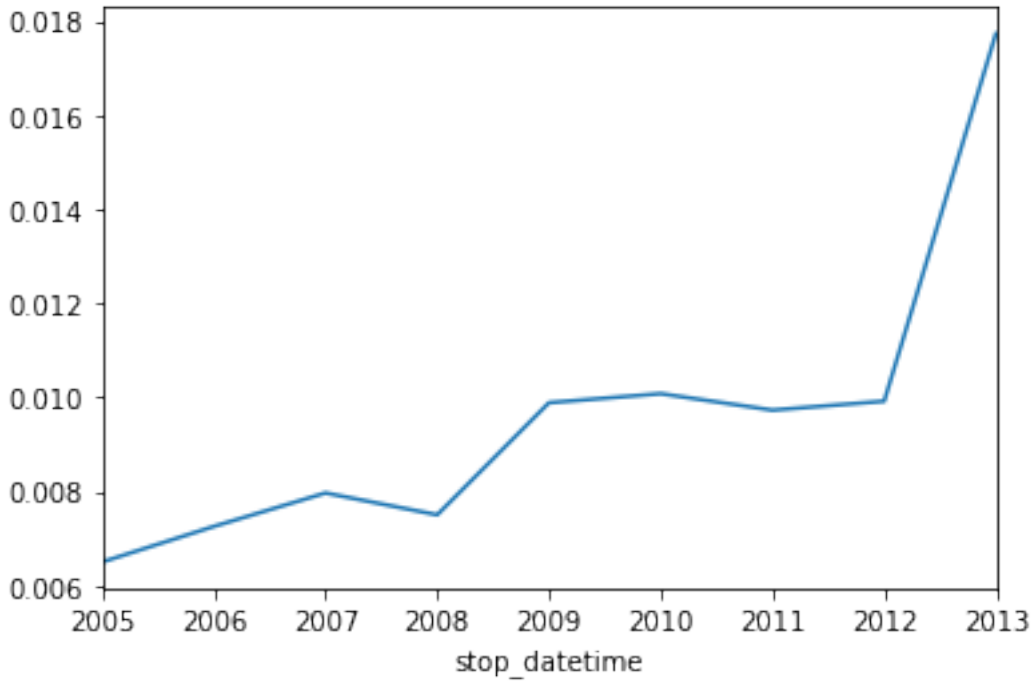
```
2010-12-31    0.010081
2011-12-31    0.009731
2012-12-31    0.009921
2013-12-31    0.017735
Freq: A-DEC, Name: drugs_related_stop, dtype: float64
```

[117]:
```
# Saving the annual rate of drug-related stops
annual_drug_rate = ri.drugs_related_stop.resample('A').mean()
print(annual_drug_rate.head())
```

```
stop_datetime
2005-12-31    0.006501
2006-12-31    0.007258
2007-12-31    0.007970
2008-12-31    0.007505
2009-12-31    0.009889
Freq: A-DEC, Name: drugs_related_stop, dtype: float64
```

[116]:
```
# Creating a line plot of 'annual_drug_rate'
annual_drug_rate.plot()

# Displaying the plot
plt.show()
```




The rate of drug-related stops increased more than double over the course of 9 years.

[121]:
```
# Calculating and saving the annual search rate
annual_search_rate = ri.search_conducted.resample('A').mean()
print(annual_search_rate)
```

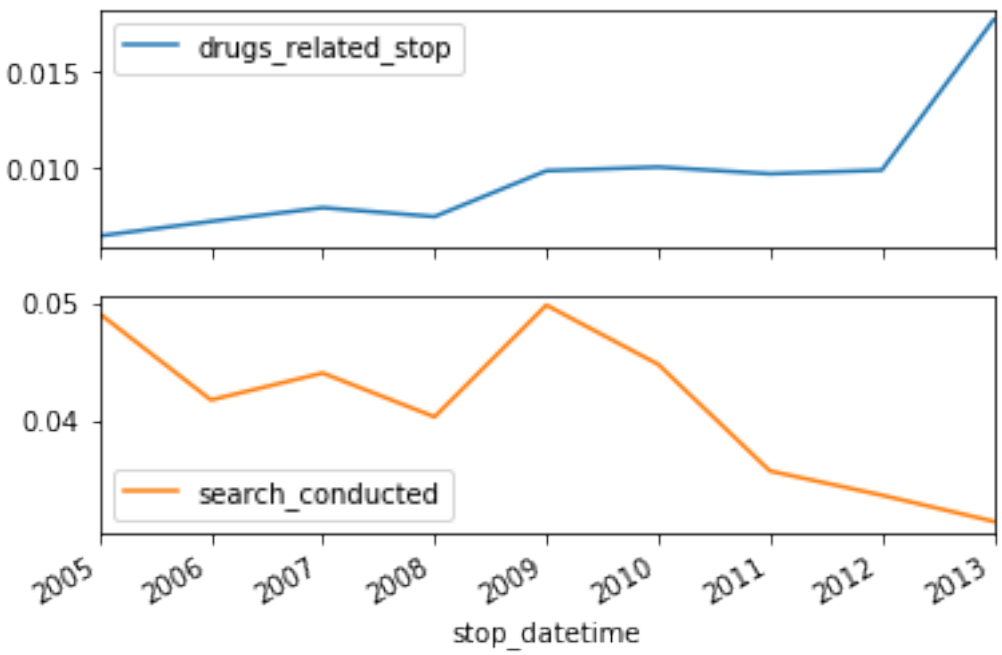
```
stop_datetime
2005-12-31    0.049167
2006-12-31    0.041758
2007-12-31    0.044056
2008-12-31    0.040310
2009-12-31    0.049861
2010-12-31    0.044805
2011-12-31    0.035682
2012-12-31    0.033616
2013-12-31    0.031378
Freq: A-DEC, Name: search_conducted, dtype: float64
```

[123]:
```
# Concatenating 'annual_drug_rate' and 'annual_search_rate'
annual = pd.concat([annual_drug_rate, annual_search_rate], axis='columns')
print(annual.head())
```

```
              drugs_related_stop  search_conducted
stop_datetime
2005-12-31              0.006501          0.049167
2006-12-31              0.007258          0.041758
2007-12-31              0.007970          0.044056
2008-12-31              0.007505          0.040310
2009-12-31              0.009889          0.049861
```

[124]:
```
# Creating subplots from 'annual'
annual.plot(subplots=True)

# Displaying the subplots
plt.show()
```




The rate of drug-related stops increased even though the search rate decreased.

[125]:
```
# Creating a frequency table of districts and violations
print(pd.crosstab(ri.district, ri.violation))
```

```
violation  Equipment  Moving violation  Other  Registration/plates  Seat belt  \
district
Zone K1          672              1254    290                  120          0
Zone K2         1041              2051    794                  441          6
Zone K3         1349              2074    588                  391         20
Zone X1          171               470     95                   16          1
Zone X3         1121              2105    619                  386         12
Zone X4         2343              4166   1289                 1038         20

violation  Speeding
district
Zone K1        5960
Zone K2        7492
Zone K3        9477
Zone X1         865
Zone X3        6840
Zone X4        7380
```

As analyzed above number of speeding violations exceeds other violations.

[129]:
```
# Saving the frequency table as 'all_zones'
all_zones = pd.crosstab(ri.district, ri.violation)
```

[130]:
```
# Selecting rows 'Zone K1' through 'Zone K3'
print(all_zones.loc['Zone K1':'Zone K3'])
```

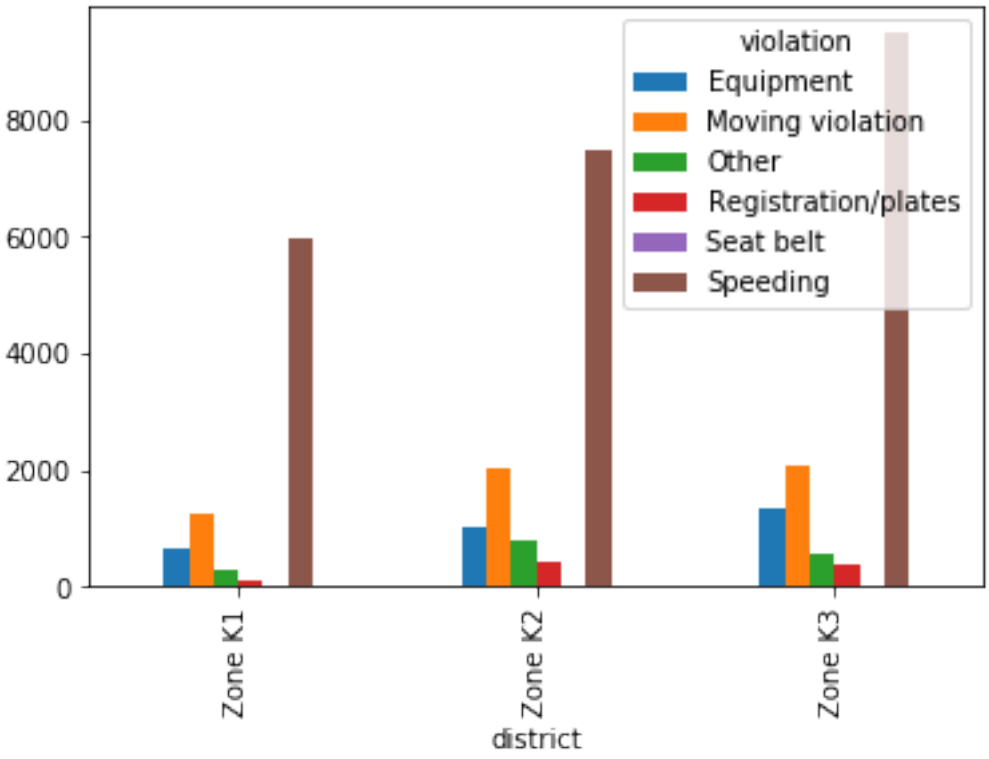
```
violation  Equipment  Moving violation  Other  Registration/plates  Seat belt  \
district
Zone K1          672              1254    290                  120          0
Zone K2         1041              2051    794                  441          6
Zone K3         1349              2074    588                  391         20

violation  Speeding
district
Zone K1        5960
Zone K2        7492
Zone K3        9477
```

[132]:
```
# Saving the smaller table as 'k_zones'
k_zones = all_zones.loc['Zone K1' : 'Zone K3']
```

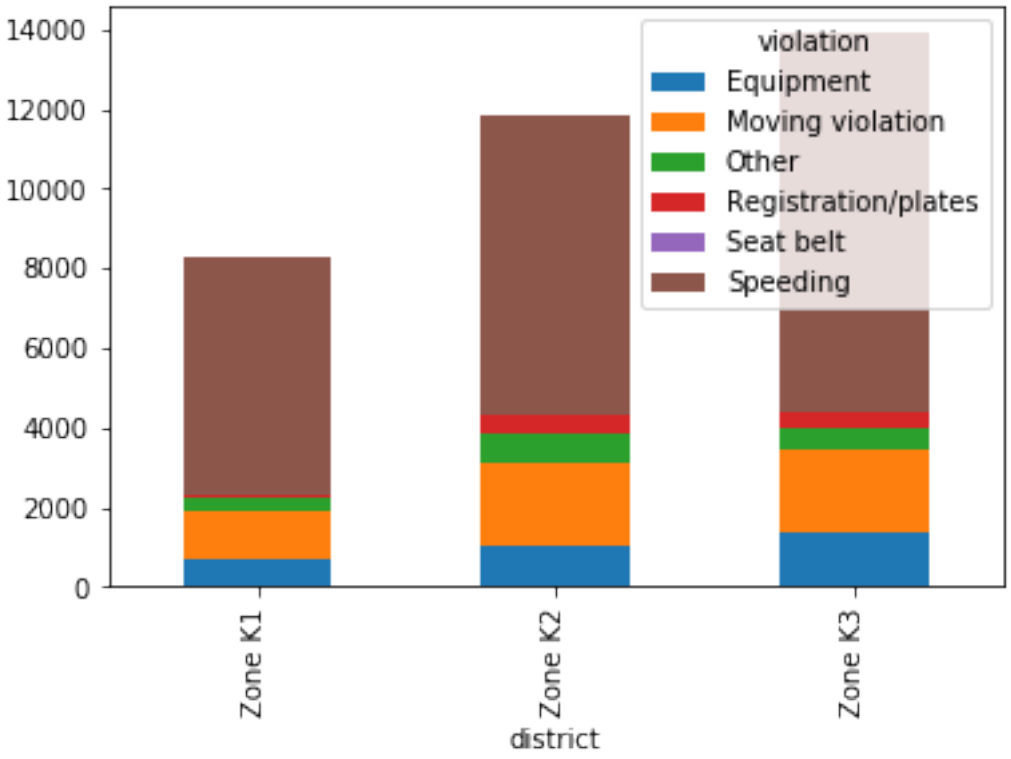
[133]:
```
# Creating a bar plot of 'k_zones'
k_zones.plot(kind = 'bar')

# Displaying the plot
plt.show()
```




[134]:
```
# Creating a stacked bar plot of 'k_zones'
k_zones.plot(kind='bar', stacked=True)

# Displaying the plot
plt.show()
```




The vast majority of traffic stops in Zone K1 are for speeding, and Zones K2 and K3 are remarkably similar to one another in terms of violations.

[136]:
```
# Dropping the stop_duration
ri.dropna(subset=['stop_duration'], inplace=True)
```

[137]:
```
# Printing the unique values in 'stop_duration'
print(ri.stop_duration.unique())
```

```
['0-15 Min' '16-30 Min' '30+ Min']
```

[139]:
```
# Creating a dictionary that maps strings to integers
mapping = {'0-15 Min':8, '16-30 Min': 23, '30+ Min': 45}
```

[140]:
```
# Converting the 'stop_duration' strings to integers using the 'mapping'
ri['stop_minutes'] = ri.stop_duration.map(mapping)
```

[141]:
```
# Printing the unique values in 'stop_minutes'
print(ri.stop_minutes.unique())
```

```
[ 8 23 45]
```

[142]:
```
# Calculating the mean 'stop_minutes' for each value in 'violation_raw'
print(ri.groupby('violation_raw').stop_minutes.mean())
```

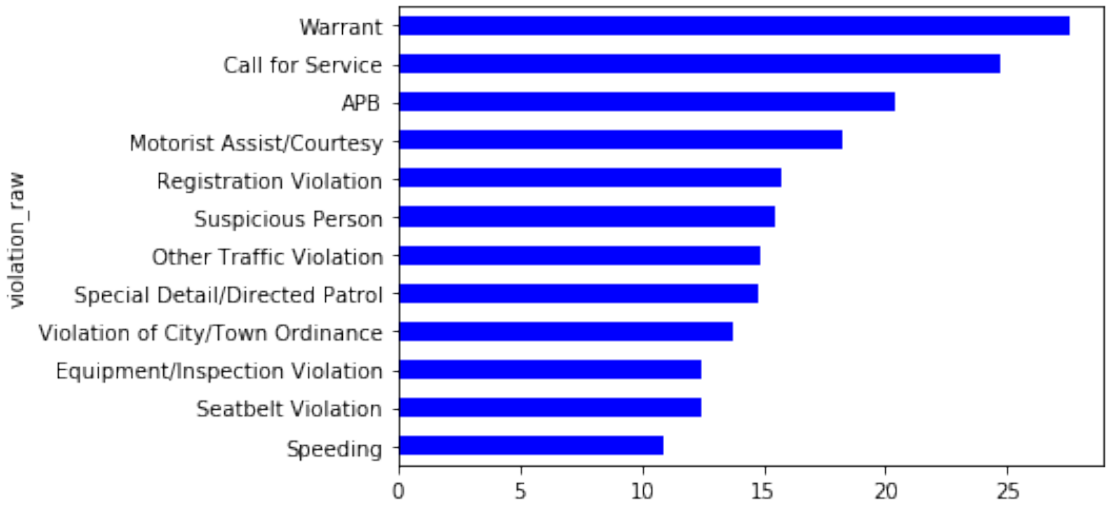
```
violation_raw
APB                                  20.378788
Call for Service                     24.702273
Equipment/Inspection Violation       12.478572
Motorist Assist/Courtesy             18.264286
Other Traffic Violation              14.843977
Registration Violation               15.761706
Seatbelt Violation                   12.423729
Special Detail/Directed Patrol       14.794538
Speeding                             10.850713
Suspicious Person                    15.447368
Violation of City/Town Ordinance     13.756250
Warrant                              27.545455
Name: stop_minutes, dtype: float64
```

Mean stop minutes are highest in cases of warrants and call for service is depicted in below graph.

[143]:
```
# Saving the resulting Series as 'stop_length'
stop_length = ri.groupby('violation_raw').stop_minutes.mean()
```

[144]:
```
# Sorting 'stop_length' by its values and creating a horizontal bar plot
stop_length.sort_values().plot(kind = 'barh', color = 'blue')
```

[144]: <matplotlib.axes._subplots.AxesSubplot at 0x119c1ef50>




Time to incorporate weather data.

[145]:
```
# Reading 'weather.csv' into a DataFrame named 'weather'
weather = pd.read_csv('weather.csv')
```

[188]:
```
weather.head()
```

[188]:

| | STATION | DATE | TAVG | TMIN | TMAX | AWND | WSF2 | WT01 | WT02 | WT03 | … | WT15 | WT16 | WT17 | WT18 | WT19 | WT21 | WT22 | TDIFF | bad_conditions | rating |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | USW00014765 | 2005-01-01 | 44.0 | 35 | 53 | 8.95 | 25.1 | 1.0 | NaN | NaN | … | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 18 | 2 | bad |
| 1 | USW00014765 | 2005-01-02 | 36.0 | 28 | 44 | 9.40 | 14.1 | NaN | NaN | NaN | … | NaN | 1.0 | NaN | 1.0 | NaN | NaN | NaN | 16 | 2 | bad |
| 2 | USW00014765 | 2005-01-03 | 49.0 | 44 | 53 | 6.93 | 17.0 | 1.0 | NaN | NaN | … | NaN | 1.0 | NaN | NaN | NaN | NaN | NaN | 9 | 3 | bad |
| 3 | USW00014765 | 2005-01-04 | 42.0 | 39 | 45 | 6.93 | 16.1 | 1.0 | NaN | NaN | … | NaN | 1.0 | NaN | NaN | NaN | NaN | NaN | 6 | 4 | bad |
| 4 | USW00014765 | 2005-01-05 | 36.0 | 28 | 43 | 7.83 | 17.0 | 1.0 | NaN | NaN | … | NaN | 1.0 | NaN | 1.0 | NaN | NaN | NaN | 15 | 4 | bad |

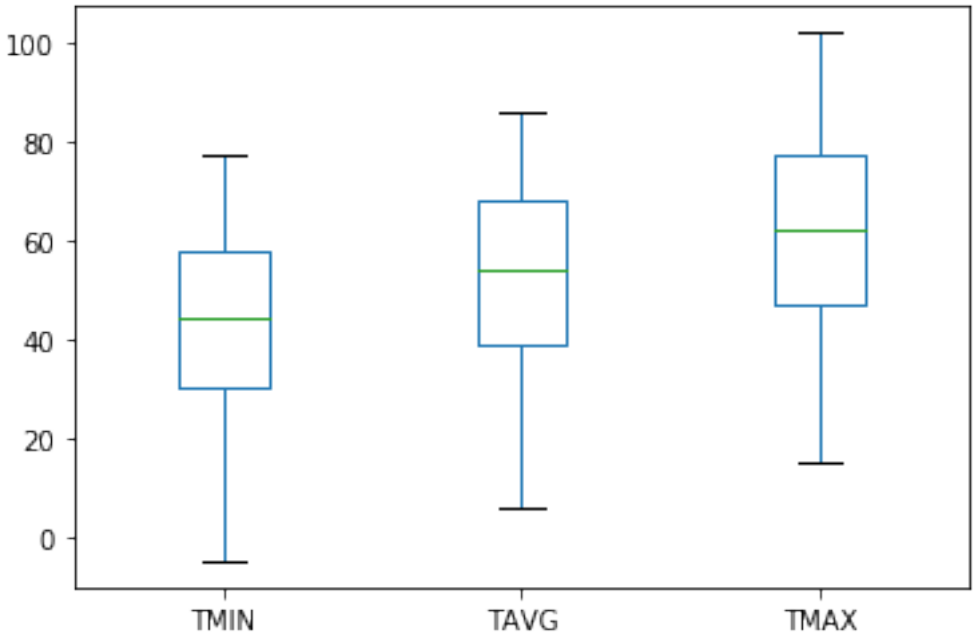
[5 rows x 30 columns]

[146]:
```
# Describing the temperature columns
print(weather[['TMIN', 'TAVG', 'TMAX']].describe())
```

```
              TMIN         TAVG         TMAX
count  4017.000000  1217.000000  4017.000000
mean     43.484441    52.493016    61.268608
std      17.020298    17.830714    18.199517
min      -5.000000     6.000000    15.000000
25%      30.000000    39.000000    47.000000
50%      44.000000    54.000000    62.000000
75%      58.000000    68.000000    77.000000
max      77.000000    86.000000   102.000000
```

[147]:
```
# Creating a box plot of the temperature columns
weather[['TMIN', 'TAVG', 'TMAX']].plot(kind='box')
```

[147]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11b68f850>
```




The temperature data looks good so far: the TAVG values are in between TMIN and TMAX, and the measurements and ranges seem reasonable.

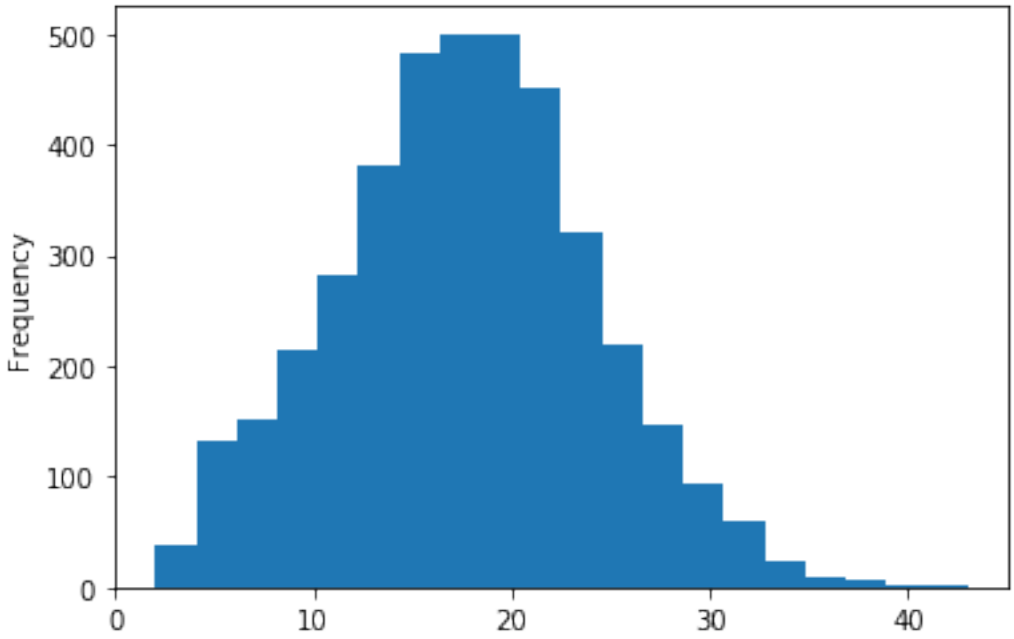
[148]:
```
# Creating a 'TDIFF' column that represents temperature difference
weather['TDIFF'] = weather.TMAX - weather.TMIN
```

[150]:
```
# Describing the 'TDIFF' column
print(weather.TDIFF.describe())
```

```
count    4017.000000
mean       17.784167
std         6.350720
min         2.000000
25%        14.000000
50%        18.000000
75%        22.000000
max        43.000000
Name: TDIFF, dtype: float64
```

[151]:
```
# Creating a histogram with 20 bins to visualize 'TDIFF'
weather.TDIFF.plot(kind='hist', bins=20)
```

[151]: `<matplotlib.axes._subplots.AxesSubplot at 0x11b68ca10>`




The TDIFF column has no negative values and its distribution is approximately normal, both of which are signs that the data is trustworthy.

[152]:
```
weather.shape
```

[152]:
```
(4017, 28)
```

[153]:
```
# Copying 'WT01' through 'WT22' to a new DataFrame
WT = weather.loc[:, 'WT01':'WT22']
```

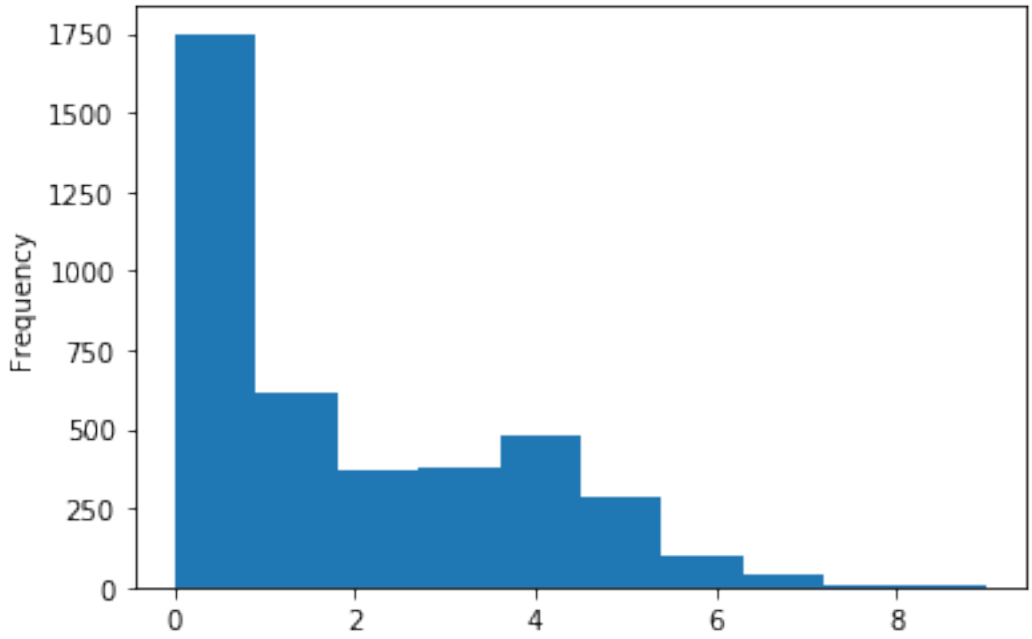
[154]:
```
# Calculating the sum of each row in 'WT'
weather['bad_conditions'] = WT.sum(axis='columns')
```

[156]:
```
# Replacing missing values in 'bad_conditions' with '0'
weather['bad_conditions'] = weather.bad_conditions.fillna(0).astype('int')
```

[157]:
```
# Creating a histogram to visualize 'bad_conditions'

weather.bad_conditions.plot(kind='hist')

# Displaying the plot
plt.show()
```




It looks like many days didn't have any bad weather conditions, and only a small portion of days had more than four bad weather conditions

[158]:
```
# Counting the unique values in 'bad_conditions' and sorting the index
print(weather.bad_conditions.value_counts().sort_index())
```

```
0    1749
1     613
2     367
3     380
4     476
5     282
6     101
7      41
8       4
9       4
Name: bad_conditions, dtype: int64
```

[159]:
```
# Creating a dictionary that maps integers to strings
mapping = {0:'good', 1:'bad', 2:'bad', 3:'bad', 4:'bad', 5:'worse', 6:'worse',␣
 ↪7:'worse', 8:'worse', 9:'worse'}
```

[160]:
```
# Converting the 'bad_conditions' integers to strings using the 'mapping'
weather['rating'] = weather.bad_conditions.map(mapping)
```

[162]:
```
# Counting the unique values in 'rating'
print(weather.rating.value_counts())
```

```
bad      1836
good     1749
worse     432
Name: rating, dtype: int64
```

[165]:
```
# Creating a list of weather ratings in logical order
from pandas.api.types import CategoricalDtype
cats = CategoricalDtype(['good', 'bad', 'worse'])
```

[168]:
```
# Changing the data type of 'rating' to category
weather['rating'] = weather.rating.astype('category', order=True, category=cats)
```

[169]:
```
print(weather.head())
```

```
       STATION        DATE  TAVG  TMIN  TMAX  AWND  WSF2  WT01  WT02  WT03  \
0  USW00014765  2005-01-01  44.0    35    53  8.95  25.1   1.0   NaN   NaN
1  USW00014765  2005-01-02  36.0    28    44  9.40  14.1   NaN   NaN   NaN
2  USW00014765  2005-01-03  49.0    44    53  6.93  17.0   1.0   NaN   NaN
3  USW00014765  2005-01-04  42.0    39    45  6.93  16.1   1.0   NaN   NaN
4  USW00014765  2005-01-05  36.0    28    43  7.83  17.0   1.0   NaN   NaN

   …  WT15  WT16  WT17  WT18  WT19  WT21  WT22  TDIFF  bad_conditions  \
0  …   NaN   NaN   NaN   NaN   NaN   NaN   NaN     18               2
1  …   NaN   1.0   NaN   1.0   NaN   NaN   NaN     16               2
2  …   NaN   1.0   NaN   NaN   NaN   NaN   NaN      9               3
3  …   NaN   1.0   NaN   NaN   NaN   NaN   NaN      6               4
4  …   NaN   1.0   NaN   1.0   NaN   NaN   NaN     15               4

   rating
0     bad
1     bad
2     bad
3     bad
4     bad

[5 rows x 30 columns]
```

[170]:
```
weather['rating'] = weather.rating.astype(CategoricalDtype(['good', 'bad',␣
↪'worse'], ordered=True))
```

[172]:
```
# Resetting the index of 'ri'
ri = pd.read_csv('Traffic stops in Rhode Island.csv')
ri.reset_index(inplace=True)
ri.dtypes
```

```
ri['is_arrested'] = ri.is_arrested.astype(bool)
```

```
/opt/anaconda3/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:3058: DtypeWarning: Columns (13) have
mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

[173]:
```
# Examining the head of 'ri'
print(ri.head())
```

```
   index state   stop_date stop_time  county_name driver_gender driver_race  \
0      0    RI  2005-01-04     12:55          NaN             M       White
1      1    RI  2005-01-23     23:15          NaN             M       White
2      2    RI  2005-02-17      4:15          NaN             M       White
3      3    RI  2005-02-20     17:15          NaN             M       White
4      4    RI  2005-02-24      1:20          NaN             F       White

                    violation_raw  violation  search_conducted search_type  \
0  Equipment/Inspection Violation  Equipment             False         NaN
1                        Speeding   Speeding             False         NaN
2                        Speeding   Speeding             False         NaN
3               Call for Service      Other             False         NaN
4                        Speeding   Speeding             False         NaN

    stop_outcome  is_arrested stop_duration drugs_related_stop district
0       Citation        False      0-15 Min              False  Zone X4
1       Citation        False      0-15 Min              False  Zone K3
2       Citation        False      0-15 Min              False  Zone X4
3  Arrest Driver         True     16-30 Min              False  Zone X1
4       Citation        False      0-15 Min              False  Zone X3
```

[174]:
```
# Creating a DataFrame from the 'DATE' and 'rating' columns
weather_rating = weather[['DATE', 'rating']]
```

[175]:
```
# Examining the head of 'weather_rating'
print(weather_rating.head())
```

```
         DATE rating
0  2005-01-01    bad
1  2005-01-02    bad
2  2005-01-03    bad
3  2005-01-04    bad
4  2005-01-05    bad
```

The ri and weather_rating DataFrames are now ready to be merged.

[176]:
```
# Examining the shape of 'ri'
print(ri.shape)
```

```
(67027, 16)
```

[177]:
```
# Merging 'ri' and 'weather_rating' using a left join
ri_weather = pd.merge(left=ri, right=weather_rating, left_on='stop_date',␣
 ↪right_on='DATE', how='left')
```

[178]:
```
ri_weather.head()
```

[178]:

| | index | state | stop_date | stop_time | county_name | driver_gender | driver_race | violation_raw | violation | search_conducted | search_type | stop_outcome | is_arrested | stop_duration | drugs_related_stop | district | DATE | rating |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | RI | 2005-01-04 | 12:55 | NaN | M | White | Equipment/Inspection Violation | Equipment | False | NaN | Citation | False | 0-15 Min | False | Zone X4 | 2005-01-04 | bad |
| 1 | 1 | RI | 2005-01-23 | 23:15 | NaN | M | White | Speeding | Speeding | False | NaN | Citation | False | 0-15 Min | False | Zone K3 | 2005-01-23 | worse |
| 2 | 2 | RI | 2005-02-17 | 4:15 | NaN | M | White | Speeding | Speeding | False | NaN | Citation | False | 0-15 Min | False | Zone X4 | 2005-02-17 | good |
| 3 | 3 | RI | 2005-02-20 | 17:15 | NaN | M | White | Call for Service | Other | False | NaN | Arrest Driver | True | 16-30 Min | False | Zone X1 | 2005-02-20 | bad |
| 4 | 4 | RI | 2005-02-24 | 1:20 | NaN | F | White | Speeding | Speeding | False | NaN | Citation | False | 0-15 Min | False | Zone X3 | 2005-02-24 | bad |

[179]:
```
# Examining the shape of 'ri_weather'
print(ri_weather.shape)
```

```
(67027, 18)
```

[180]:
```
# Setting 'stop_datetime' as the index of 'ri_weather'
ri_weather.set_index('stop_date', inplace=True)
```

[181]:
```
# Calculating the overall arrest rate
print(ri_weather.is_arrested.mean())
```

```
0.11625165977889507
```

[182]:
```
# Calculating arrest rates during different weathers
print(ri_weather.groupby('rating').is_arrested.mean())
```

```
rating
good     0.111335
bad      0.122750
worse    0.106995
Name: is_arrested, dtype: float64
```

[183]:
```
# Calculating the arrest rate for each 'violation' and 'rating'
print(ri_weather.groupby(['violation', 'rating']).is_arrested.mean())
```

```
violation            rating
Equipment            good      0.102384
                     bad       0.123762
                     worse     0.097872
Moving violation     good      0.075846
                     bad       0.091887
                     worse     0.066261
Other                good      0.072359
                     bad       0.100724
                     worse     0.062992
Registration/plates  good      0.142237
                     bad       0.153979
                     worse     0.117460
Seat belt            good      1.000000
                     bad       1.000000
                     worse          NaN
Speeding             good      0.030534
                     bad       0.039206
                     worse     0.017610
Name: is_arrested, dtype: float64
```

Number of violations is highest in the bad weather.