Exploring Pandas with NYC Flights Dataset

R users will recognize this famous dataset as a favorite of Hadley Wickham, author of dplyr. No matter, we can use it with pandas as well.

This dataset will give us flight delay data from all flights departing from the three NYC airports (JFK, LGA, EWR) in the year 2013.

In [1]:
```
%matplotlib inline

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
df = pd.read_csv('../../data/nycflights13/flights.csv.gz')
```

In [3]:
```
df
```

336771	2013	9	30	NaN	1455	NaN	NaN	1634	Na
336772	2013	9	30	NaN	2200	NaN	NaN	2312	Na
336773	2013	9	30	NaN	1210	NaN	NaN	1330	Na
336774	2013	9	30	NaN	1159	NaN	NaN	1344	Na
336775	2013	9	30	NaN	840	NaN	NaN	1020	Na

336776 rows × 19 columns

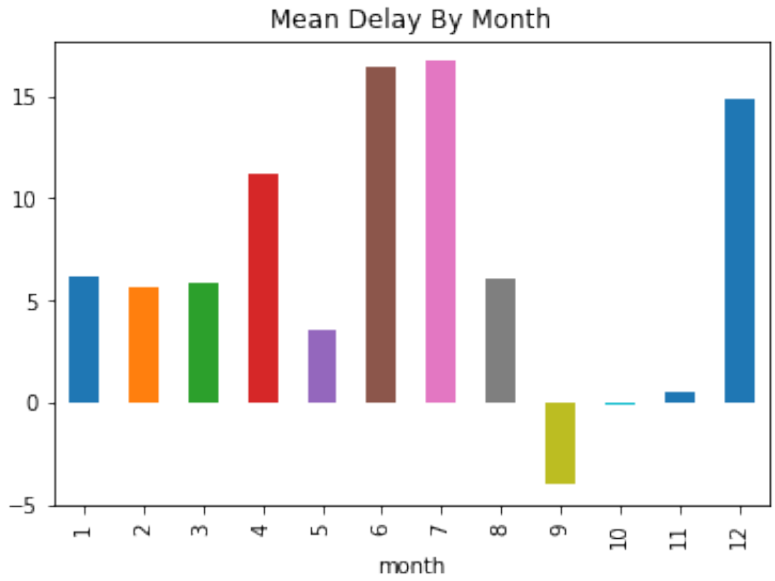
Let us focus on some summary and visualizations that we can do here. Let's start out by finding the average delay by month. We'll also do a matlibplot on it.

In [4]:
```
mean_delay_by_month = df.groupby(['month'])['arr_delay'].mean()
mean_delay_by_month
```

Out[4]:
```
month
1       6.129972
2       5.613019
3       5.807577
4      11.176063
5       3.521509
6      16.481330
7      16.711307
8       6.040652
9      -4.018364
10     -0.167063
11      0.461347
12     14.870355
Name: arr_delay, dtype: float64
```

In [5]:
```
mean_month_plt = mean_delay_by_month.plot(kind='bar', title="Mean Delay B
mean_month_plt
```

Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x103edfac8>
```



Notice that flights in two months actually have negative delay. No doubt airlines "pad" flight times to achieve better results.

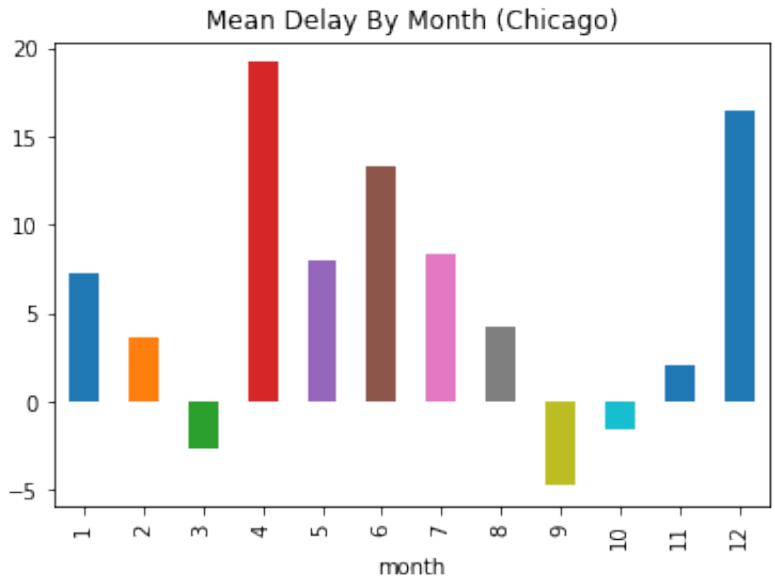
One would guess that flights to certain airports may have different results. For example, flights to Chicago (ORD) would no doubt be affected by winter weather. Let's take a look at that.

In [6]:
```
mean_delay_by_month_ord = df[(df.dest == 'ORD')].groupby(['month'])['arr_
print("Flights to Chicago (ORD)")
print(mean_delay_by_month_ord)

mean_month_plt_ord = mean_delay_by_month_ord.plot(kind='bar', title="Mean
mean_month_plt_ord
```

```
Flights to Chicago (ORD)
month
1       7.287694
2       3.680794
3      -2.702473
4      19.179352
5       7.938280
6      13.299376
7       8.405514
8       4.256851
9      -4.745370
10     -1.597090
11      2.071058
12     16.462817
Name: arr_delay, dtype: float64
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x104c38da0>

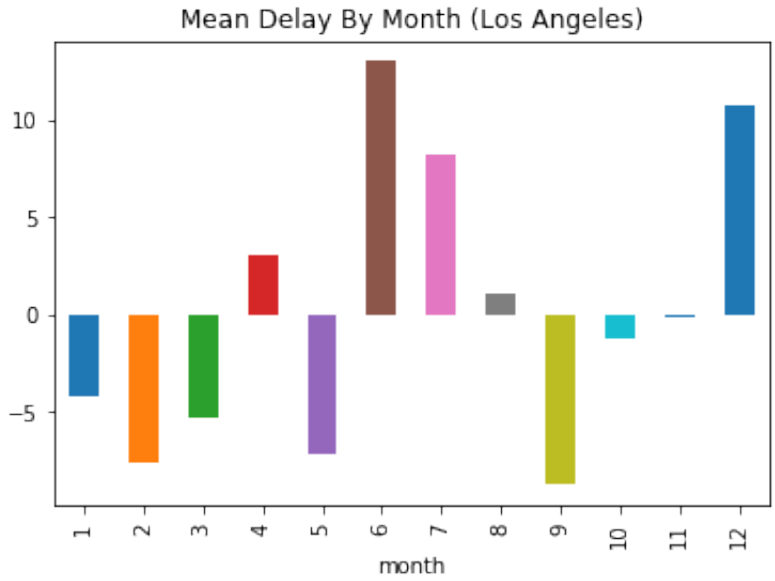


In [7]:
```
# Let's try Los Angeles and compare that

mean_delay_by_month_lax = df[(df.dest == 'LAX')].groupby(['month'])['arr_
print("Flights to Los Angeles (LAX)")
print(mean_delay_by_month_lax)

mean_month_plt_lax = mean_delay_by_month_lax.plot(kind='bar', title="Mean
mean_month_plt_lax
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x104c4e5c0>
```



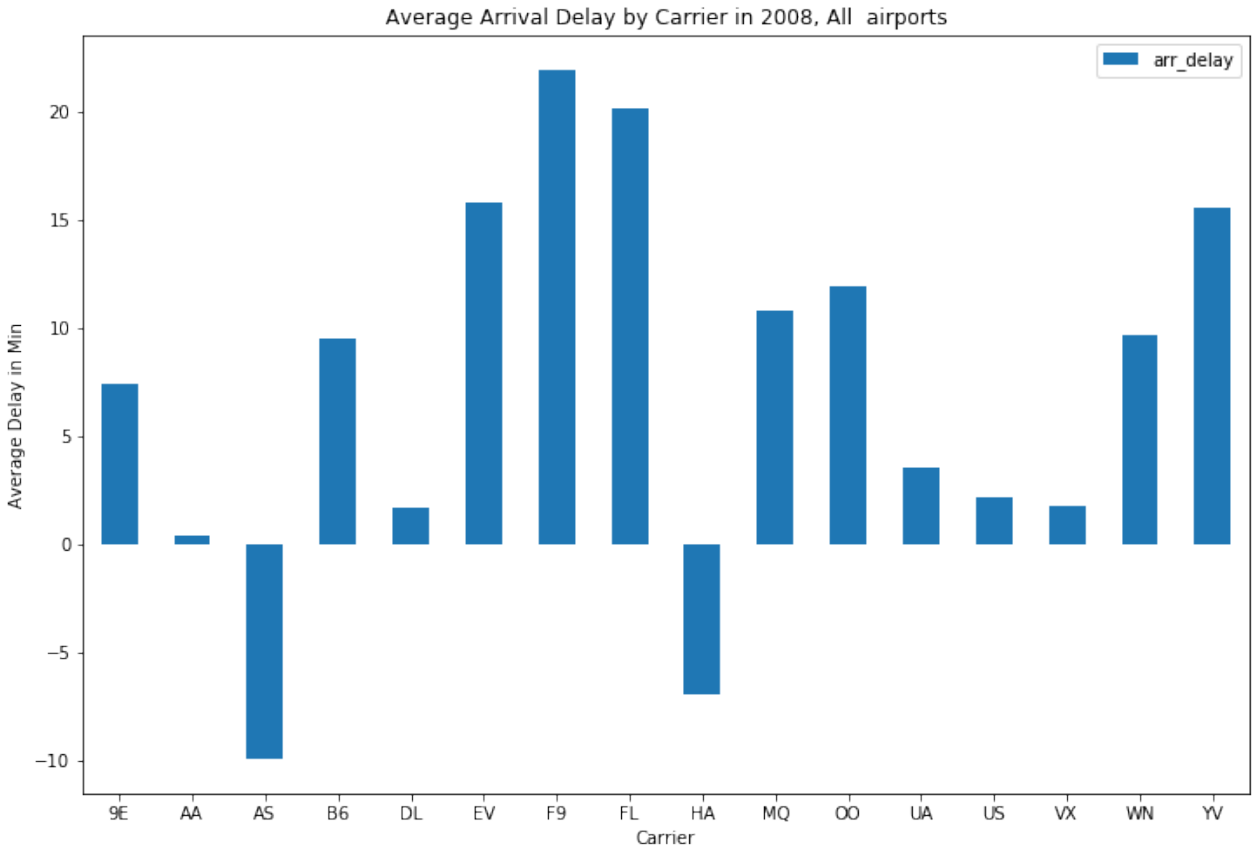
Perhaps we would like to see which Airline carrier is the best for delays. Here we have some plots to do that.

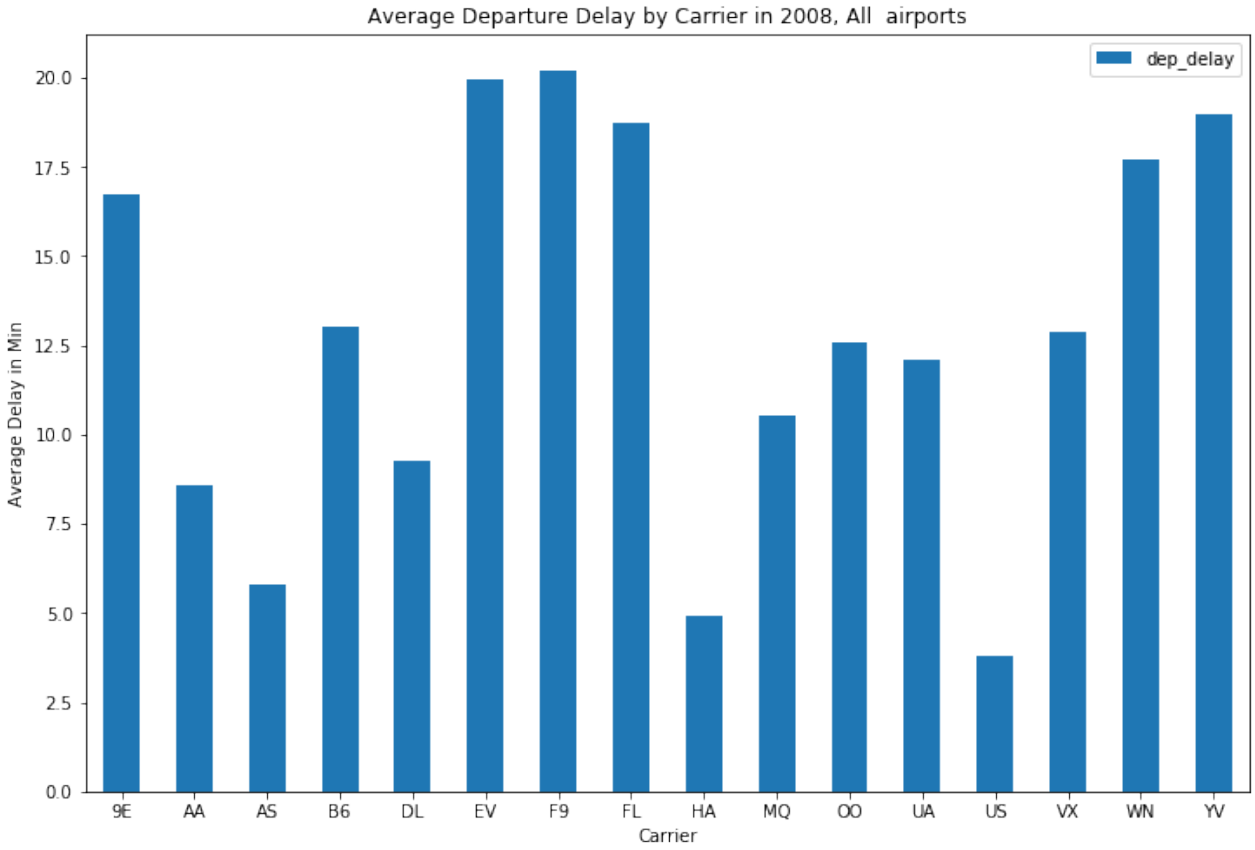
In [8]:
```
# Examine if specific carrier will create different delay impact
df[['carrier','arr_delay']].groupby('carrier').mean().plot(kind='bar', fi
plt.xticks(rotation=0)
plt.xlabel('Carrier')
plt.ylabel('Average Delay in Min')
plt.title('Average Arrival Delay by Carrier in 2008, All  airports')


df[['carrier','dep_delay']].groupby('carrier').mean().plot(kind='bar', fi
plt.xticks(rotation=0)
plt.xlabel('Carrier')
plt.ylabel('Average Delay in Min')
plt.title('Average Departure Delay by Carrier in 2008, All  airports')
```

Out[8]:
```
Text(0.5,1,'Average Departure Delay by Carrier in 2008, All  airports'
```

)


Average Arrival Delay by Carrier in 2008, All airports
arr_delay
Average Delay in Min
20
15
10
5
0
−5
−10
9E
AA
AS
B6
DL
EV
F9
FL
HA
MQ
OO
UA
US
VX
WN
YV
Carrier


Average Departure Delay by Carrier in 2008, All airports
dep_delay
Average Delay in Min
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
9E
AA
AS
B6
DL
EV
F9
FL
HA
MQ
OO
UA
US
VX
WN
YV
Carrier

We see that F9 (Frontier Airlines) is among the worst in terms of delays, while Hawaiian (HA) is among the best.

Joins

We have a couple of other tables in our dataset: Weather, and airports. Let's look at these two tables and see if we can perform joins.

In [9]:
```
weather = pd.read_csv('../../data/nycflights13/weather.csv.gz')
weather
```

5	EWR	2013	1	1	6	39.02	26.06	59.37	270.0	10.35702	11.918651	(
6	EWR	2013	1	1	7	39.02	26.96	61.63	250.0	8.05546	9.270062	(
7	EWR	2013	1	1	8	39.02	28.04	64.43	240.0	11.50780	13.242946	(
8	EWR	2013	1	1	9	39.92	28.04	62.21	250.0	12.65858	14.567241	(
9	EWR	2013	1	1	10	39.02	28.04	64.43	260.0	12.65858	14.567241	(
10	EWR	2013	1	1	11	37.94	28.04	67.21	240.0	11.50780	13.242946	(

In [10]:
```
df_withweather = pd.merge(df, weather, how='left', on=['year','month', 'd
df_withweather
```

1007584	2013	9	30	NaN	1210	NaN	NaN	1330	
1007585	2013	9	30	NaN	1159	NaN	NaN	1344	
1007586	2013	9	30	NaN	1159	NaN	NaN	1344	
1007587	2013	9	30	NaN	1159	NaN	NaN	1344	
1007588	2013	9	30	NaN	840	NaN	NaN	1020	
1007589	2013	9	30	NaN	840	NaN	NaN	1020	
1007590	2013	9	30	NaN	840	NaN	NaN	1020	

1007591 rows × 30 columns

In [11]:
```
airports = pd.read_csv('../../data/nycflights13/airports.csv.gz')
airports
```

Out[11]:

	faa	name	lat	lon	alt	tz	dst	tzone
0	04G	Lansdowne Airport	41.130472	-80.619583	1044	-5	A	America/New_York
1	06A	Moton Field Municipal Airport	32.460572	-85.680028	264	-6	A	America/Chicago
2	06C	Schaumburg Regional	41.989341	-88.101243	801	-6	A	America/Chicago
3	06N	Randall Airport	41.431912	-74.391561	523	-5	A	America/New_York
4	09J	Jekyll Island Airport	31.074472	-81.427778	11	-5	A	America/New_York
5	0A9	Elizabethton Municipal Airport	36.371222	-82.173417	1593	-5	A	America/New_York
6	0G6	Williams County Airport	41.467306	-84.506778	730	-5	A	America/New_York
7	0G7	Finger Lakes Regional Airport	42.883565	-76.781232	492	-5	A	America/New_York
8	0P2	Shoestring Aviation Airfield	39.794824	-76.647191	1000	-5	U	America/New_York

In [12]:
```
df_withairport = pd.merge(df_withweather, airports, how='left', left_on='
df_withairport
```

Out[12]:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_d
0	2013	1	1	517.0	515	2.0	830.0	819	
1	2013	1	1	533.0	529	4.0	850.0	830	
2	2013	1	1	542.0	540	2.0	923.0	850	
3	2013	1	1	544.0	545	-1.0	1004.0	1022	
4	2013	1	1	554.0	600	-6.0	812.0	837	
5	2013	1	1	554.0	600	-6.0	812.0	837	
6	2013	1	1	554.0	600	-6.0	812.0	837	

In [ ]:
```
```