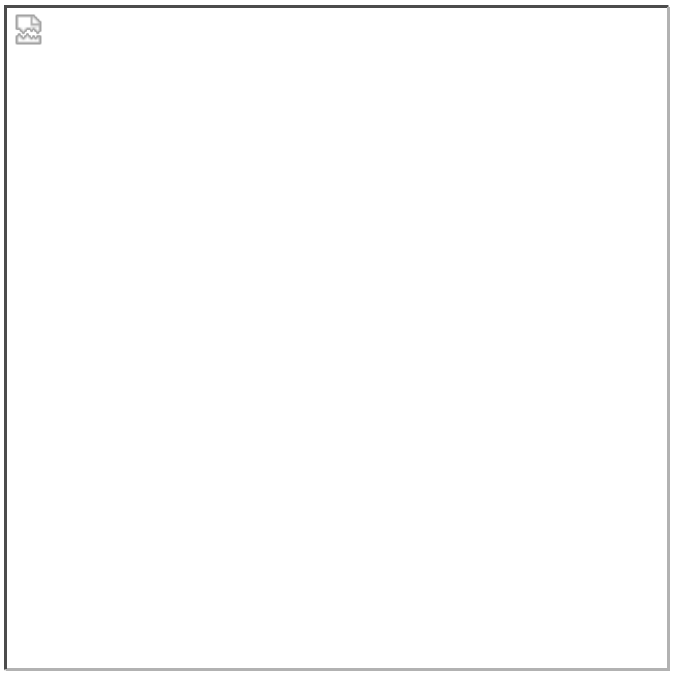
mlcourse.ai – Open Machine Learning Course

Author: Yury Kashnitskiy. Translated and edited by Maxim Keremet, Artem Trunov, and Aditya Soni.

Assignment #2. Fall 2018

Exploratory Data Analysis (EDA) of US flights

(using Pandas, Matplotlib & Seaborn)



Prior to working on the assignment, you'd better check out the corresponding course material:

- Visualization: from Simple Distributions to Dimensionality Reduction
- Overview of Seaborn, Matplotlib and Plotly libraries
- first lectures in this YouTube playlist

Your task is to:

- write code and perform computations in the cells below
- choose answers in the webform
- submit answers with **the very same email and name** as in assignment 1. This is a part of the assignment, if you don't manage to do so, you won't get credits. If in doubt, you can re-submit A1 form till the deadline for A1, no problem

Deadline for A2: 2018 October 21, 20:59 CET

```
import numpy as np
import pandas as pd
# pip install seaborn
import seaborn as sns
import matplotlib.pyplot as plt
```

- Download the data archive (Archived ~ 114 Mb, unzipped - ~ 690 Mb). No need to unzip - pandas can unbzip on the fly.
- Place it in the "../../data" folder, or change the path below according to your location.
- The dataset has information about carriers and flights between US airports during the year 2008.
- Column description is available here. Visit this site to find ex. meaning of flight cancellation codes.

Reading data into memory and creating a Pandas *DataFrame* object

(This may take a while, be patient)

We are not going to read in the whole dataset. In order to reduce memory footprint, we instead load only needed columns and cast them suitable data types.

```
dtype = {'DayOfWeek': np.uint8, 'DayofMonth': np.uint8, 'Month': np.uint8 , 'Cancelled':
         'Year': np.uint16, 'FlightNum': np.uint16 , 'Distance': np.uint16,
         'UniqueCarrier': str, 'CancellationCode': str, 'Origin': str, 'Dest': str,
         'ArrDelay': np.float16, 'DepDelay': np.float16, 'CarrierDelay': np.float16,
         'WeatherDelay': np.float16, 'NASDelay': np.float16, 'SecurityDelay': np.float16
         'LateAircraftDelay': np.float16, 'DepTime': np.float16}
```

```
!wget http://stat-computing.org/dataexpo/2009/2008.csv.bz2
```

```
--2018-10-18 17:35:35--  http://stat-computing.org/dataexpo/2009/2008.csv.bz2
Resolving stat-computing.org (stat-computing.org)... 52.218.240.83
Connecting to stat-computing.org (stat-computing.org)|52.218.240.83|:80... connect
HTTP request sent, awaiting response... 200 OK
Length: 113753229 (108M) [application/x-bzip2]
Saving to: ‘2008.csv.bz2’

2008.csv.bz2        100%[===================>] 108.48M  33.4MB/s    in 3.6s

2018-10-18 17:35:39 (30.5 MB/s) - ‘2008.csv.bz2’ saved [113753229/113753229]
```

```
%%time
# change the path if needed
path = './2008.csv.bz2'
flights_df = pd.read_csv(path, usecols=dtype.keys(), dtype=dtype)
```

```
CPU times: user 46.6 s, sys: 1.22 s, total: 47.8 s
Wall time: 48.1 s
```

Check the number of rows and columns and print column names.

```
print(flights_df.shape)
print(flights_df.columns)
```

```
(7009728, 19)
Index(['Year', 'Month', 'DayofMonth', 'DayOfWeek', 'DepTime', 'UniqueCarrier',
       'FlightNum', 'ArrDelay', 'DepDelay', 'Origin', 'Dest', 'Distance',
       'Cancelled', 'CancellationCode', 'CarrierDelay', 'WeatherDelay',
       'NASDelay', 'SecurityDelay', 'LateAircraftDelay'],
      dtype='object')
```

Print first 5 rows of the dataset.

```
flights_df.head()
```

	Year	Month	DayofMonth	DayOfWeek	DepTime	UniqueCarrier	FlightNum	ArrDelay
0	2008	1	3	4	2003.0	WN	335	-14.0
1	2008	1	3	4	754.0	WN	3231	2.0
2	2008	1	3	4	628.0	WN	448	14.0
3	2008	1	3	4	926.0	WN	1746	-6.0
4	2008	1	3	4	1829.0	WN	3920	34.0

Transpose the frame to see all features at once.

```
flights_df.head().T
```

	0	1	2	3	4
Year	2008	2008	2008	2008	2008
Month	1	1	1	1	1
DayofMonth	3	3	3	3	3
DayOfWeek	4	4	4	4	4
DepTime	2003	754	628	926	1829
UniqueCarrier	WN	WN	WN	WN	WN
FlightNum	335	3231	448	1746	3920
ArrDelay	-14	2	14	-6	34
DepDelay	8	19	8	-4	34

DepDelay	8	19	8	-4	34
Origin	IAD	IAD	IND	IND	IND
Dest	TPA	TPA	BWI	BWI	BWI
Distance	810	810	515	515	515
Cancelled	0	0	0	0	0
CancellationCode	NaN	NaN	NaN	NaN	NaN
CarrierDelay	NaN	NaN	NaN	NaN	2
WeatherDelay	NaN	NaN	NaN	NaN	0
NASDelay	NaN	NaN	NaN	NaN	0
SecurityDelay	NaN	NaN	NaN	NaN	0
LateAircraftDelay	NaN	NaN	NaN	NaN	32

Examine data types of all features and total dataframe size in memory.

```
flights_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7009728 entries, 0 to 7009727
Data columns (total 19 columns):
Year                 uint16
Month                uint8
DayofMonth           uint8
DayOfWeek            uint8
DepTime              float16
UniqueCarrier        object
FlightNum            uint16
ArrDelay             float16
DepDelay             float16
Origin               object
Dest                 object
Distance             uint16
Cancelled            uint8
CancellationCode     object
CarrierDelay         float16
WeatherDelay         float16
NASDelay             float16
SecurityDelay        float16
LateAircraftDelay    float16
dtypes: float16(8), object(4), uint16(3), uint8(4)
memory usage: 387.7+ MB
```

Get basic statistics of each feature.

```
flights_df.describe().T
```

	count	mean	std	min	25%	50%	75%

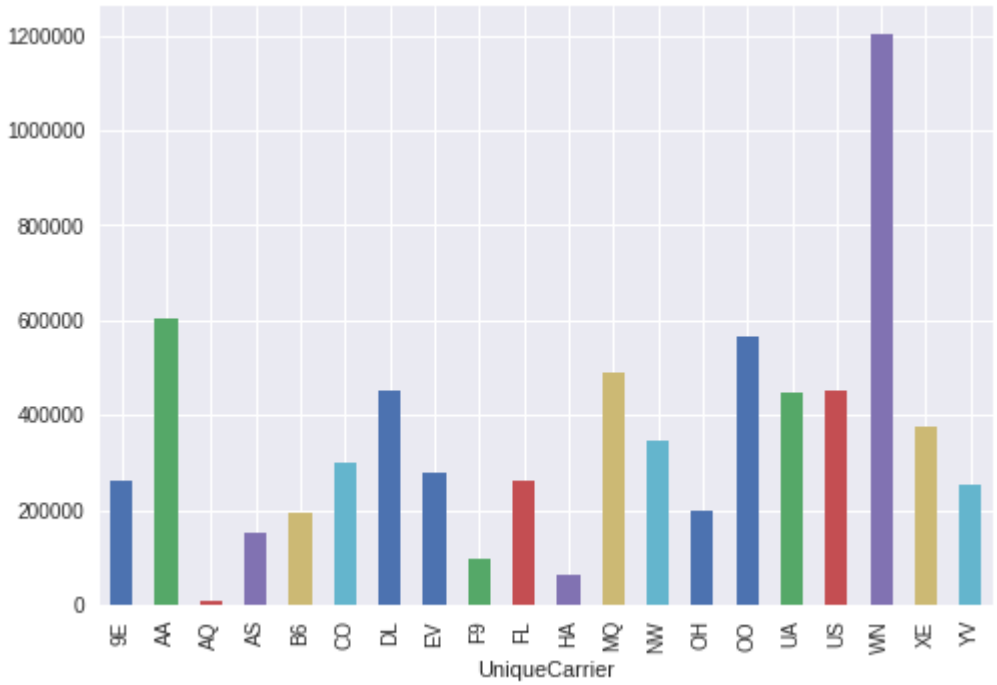
Year	7009728.0	2008.000000	0.000000	2008.0	2008.0	2008.0	2008.0
Month	7009728.0	6.375130	3.406737	1.0	3.0	6.0	9.0
DayofMonth	7009728.0	15.728015	8.797068	1.0	8.0	16.0	23.0
DayOfWeek	7009728.0	3.924182	1.988259	1.0	2.0	4.0	6.0
DepTime	6873482.0	NaN	NaN	1.0	928.0	1325.0	1728.0
FlightNum	7009728.0	2224.200105	1961.715999	1.0	622.0	1571.0	3518.0
ArrDelay	6855029.0	NaN	NaN	-519.0	-10.0	-2.0	12.0
DepDelay	6873482.0	NaN	NaN	-534.0	-4.0	-1.0	8.0
Distance	7009728.0	726.387029	562.101803	11.0	325.0	581.0	954.0
Cancelled	7009728.0	0.019606	0.138643	0.0	0.0	0.0	0.0
CarrierDelay	1524735.0	NaN	NaN	0.0	0.0	0.0	16.0
WeatherDelay	1524735.0	NaN	NaN	0.0	0.0	0.0	0.0
NASDelay	1524735.0	NaN	NaN	0.0	0.0	6.0	21.0
SecurityDelay	1524735.0	NaN	NaN	0.0	0.0	0.0	0.0
LateAircraftDelay	1524735.0	NaN	NaN	0.0	0.0	0.0	26.0

Count unique Carriers and plot their relative share of flights:

```
flights_df['UniqueCarrier'].nunique()
```

```
20
```

```
flights_df.groupby('UniqueCarrier').size().plot(kind='bar');
```



We can also *group by* category/categories in order to calculate different aggregated statistics.

For example, finding top-3 flight codes, that have the largest total distance travelled in year 2008.

```
flights_df.groupby(['UniqueCarrier','FlightNum'])['Distance'].sum().sort_values(ascendir
```

```
UniqueCarrier  FlightNum
CO             15           1796244.0
               14           1796244.0
UA             52           1789722.0
Name: Distance, dtype: float64
```

Another way:

```
flights_df.groupby(['UniqueCarrier','FlightNum'])\
  .agg({'Distance': [np.mean, np.sum, 'count'],
        'Cancelled': np.sum})\
  .sort_values(('Distance', 'sum'), ascending=False)\
  .iloc[0:3]
```

		Distance			Cancelled
		mean	sum	count	sum
UniqueCarrier	FlightNum				
CO	15	4962.000000	1796244.0	362	0
	14	4962.000000	1796244.0	362	0
UA	52	2465.181818	1789722.0	726	8

Number of flights by days of week and months:

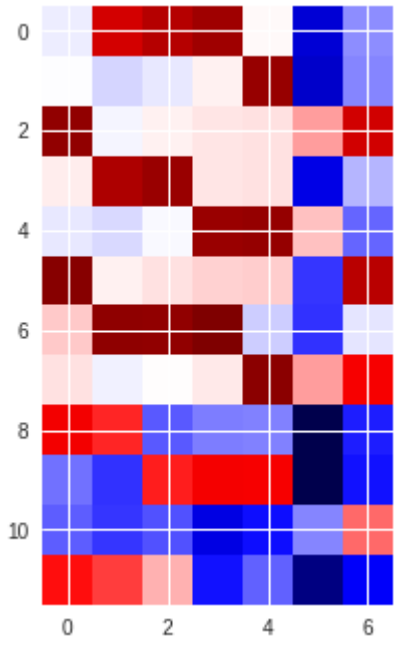
```
pd.crosstab(flights_df.Month, flights_df.DayOfWeek)
```

DayOfWeek	1	2	3	4	5	6	7
Month							
1	80807	97298	100080	102043	81940	67178	76419
2	81504	79700	80587	82158	102726	66462	76099
3	103210	81159	82307	82831	82936	86153	97494
4	82463	100785	102586	82799	82964	68304	78225
5	80626	79884	81264	102572	102878	84493	74576

6	104168	82160	82902	83617	83930	72322	99566
7	84095	103429	103315	105035	79349	72219	80489
8	82983	80895	81773	82625	103878	86155	93970
9	94300	91533	74057	75589	75881	58343	71205
10	75131	72195	91900	94123	93894	58168	70794
11	74214	72443	73653	68071	70484	76031	88376
12	92700	90568	85241	70761	74306	61708	69674

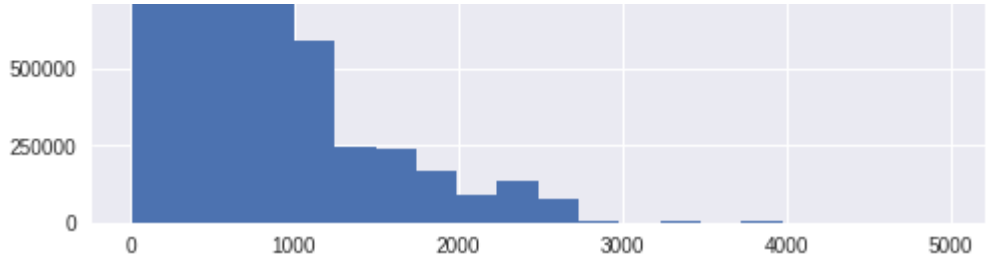
It can also be handy to color such tables in order to easily notice outliers:

```
plt.imshow(pd.crosstab(flights_df.Month, flights_df.DayOfWeek),
           cmap='seismic', interpolation='none');
```



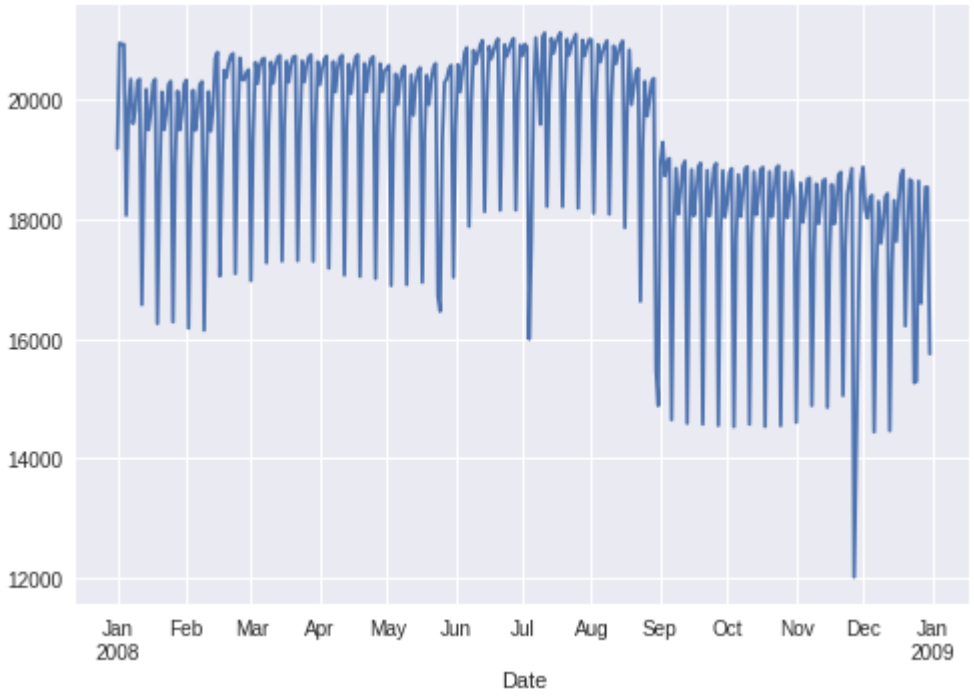
Flight distance histogram:

```
flights_df.hist('Distance', bins=20);
```



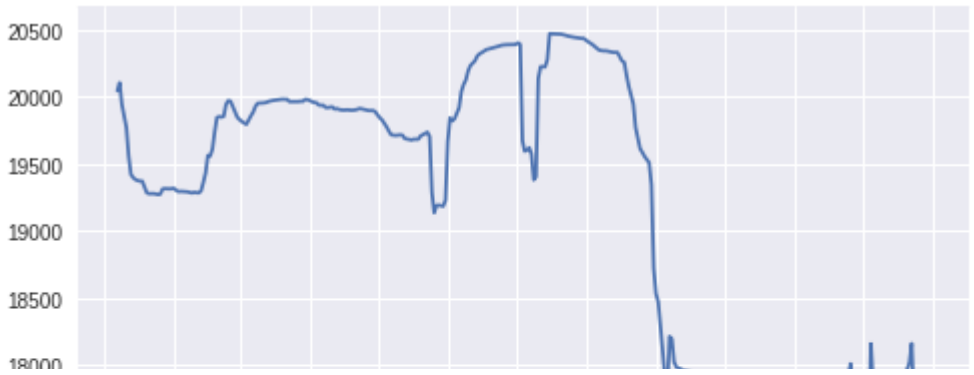
Making a histogram of flight frequency by date.

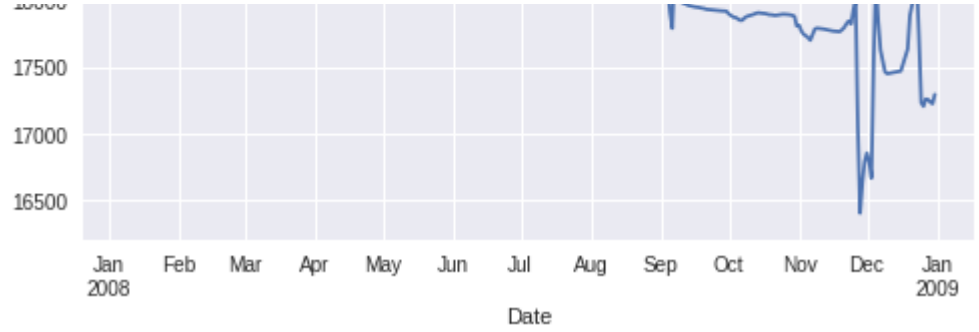
```
flights_df['Date'] = pd.to_datetime(flights_df.rename(columns={'DayofMonth': 'Day'})[['Y

num_flights_by_date = flights_df.groupby('Date').size()

num_flights_by_date.plot();
```



Do you see a weekly pattern above? And below?

```
num_flights_by_date.rolling(window=7).mean().plot();
```





1. Find top-10 carriers in terms of the number of completed flights (*UniqueCarrier* column)?

Which of the listed below is *not* in your top-10 list?

- DL
- AA
- OO
- ***EV***

```
# You code here
flights_df.groupby('UniqueCarrier')['FlightNum'].count().sort_values(ascending=False).il
```

```
UniqueCarrier
WN    1201754
AA     604885
OO     567159
MQ     490693
US     453589
DL     451931
UA     449515
XE     374510
NW     347652
CO     298455
Name: FlightNum, dtype: int64
```

2. Plot distributions of flight cancellation reasons (*CancellationCode*).

What is the most frequent reason for flight cancellation? (Use this link to translate codes into reasons)

- carrier
- ***weather conditions ***
- National Air System
- security reasons

```
# You code here
flights_df.groupby('CancellationCode')['CancellationCode'].count().sort_values(ascending
```

```
CancellationCode
B    54904
A    54330
```

```
..    5455
C    28188
D       12
Name: CancellationCode, dtype: int64
```

3. Which route is the most frequent, in terms of the number of flights?

(Take a look at *'Origin'* and *'Dest'* features. Consider *A->B* and *B->A* directions as *different* routes)

- New-York – Washington
- ***San-Francisco – Los-Angeles***
- San-Jose – Dallas
- New-York – San-Francisco

```
# You code here
pd.crosstab(flights_df.Origin, flights_df.Dest)
```

Dest	ABE	ABI	ABQ	ABY	ACK	ACT	ACV	ACY	ADK	ADQ	...	TYR	TYS	VLD	VI
Origin															
ABE	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
ABI	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
ABQ	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
ABY	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
ACK	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
ACT	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
ACV	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
ACY	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
ADK	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
ADQ	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
AEX	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
AGS	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
AKN	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
ALB	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
ALO	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
AMA	0	0	366	0	0	0	0	0	0	0	...	0	0	0	
ANC	0	0	0	0	0	0	0	0	102	706	...	0	0	0	
ASE	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
ATL	852	0	1064	1095	0	0	0	113	0	0	...	0	2428	938	30

											...			
ATW	0	0	0	0	0	0	0	0	0	0	...	0	0	0
AUS	0	0	435	0	0	0	0	0	0	0	...	0	0	0
AVL	0	0	0	0	0	0	0	0	0	0	...	0	0	0
AVP	0	0	0	0	0	0	0	0	0	0	...	0	0	0
AZO	0	0	0	0	0	0	0	0	0	0	...	0	0	0
BDL	0	0	0	0	0	0	0	0	0	0	...	0	0	0
BET	0	0	0	0	0	0	0	0	0	0	...	0	0	0
BFL	0	0	0	0	0	0	0	0	0	0	...	0	0	0
BGM	0	0	0	0	0	0	0	0	0	0	...	0	0	0
BGR	0	0	0	0	0	0	0	0	0	0	...	0	0	0
BHM	0	0	0	0	0	0	0	0	0	0	...	0	0	0

```
flight_freq = flights_df.groupby(['Origin', 'Dest'])['FlightNum'].count()#.max()
```

```
type(flight_freq)
```

```
pandas.core.series.Series
```

```
flight_freq.idxmax()
```

```
('SFO', 'LAX')
```

4. Find top-5 delayed routes (count how many times they were delayed on departure). From all flights on these 5 routes, count all flights with weather conditions contributing to a delay.

- 449
- 539
- 549
- ***668***

This didn't work :(

[ ] ↳ *10 cells hidden*

Let's break this problem down

First, let's find top 5 delayed routes

```
#delayed_routes = flights_df.groupby(['Origin', 'Dest'])['DepDelay']#.filter(lambda x: x
```

```
flight_delay_df = pd.DataFrame({'count' : flights_df.loc[(flights_df['DepDelay'] > 0)].g
```

```
delayed_routes_top5 = flight_delay_df.sort_values(by='count', ascending=False).iloc[:5]
```

```
delayed_routes_top5
```

	Origin	Dest	count
2722	LAX	SFO	6253
1179	DAL	HOU	5742
4636	SFO	LAX	5322
3775	ORD	LGA	5311
2080	HOU	DAL	5288

Making sure everything looks okay till now

[ ] ↳ *2 cells hidden*

Get all flights between these routes

```
# flights_delayed_filtered = flights_df.loc((flights_df['Origin'] == delayed_routes_top5
```

```
# flights_df.loc((flights_df['Origin'].isin(delayed_routes_top5['Origin'])))
```

```
keys = list(['Origin', 'Dest'])
i1 = flights_df.set_index(keys).index
i2 = delayed_routes_top5.set_index(keys).index
```

```
flights_delayed_filtered = flights_df[i1.isin(i2)]
```

```
flights_delayed_filtered
```

	Year	Month	DayofMonth	DayOfWeek	DepTime	UniqueCarrier	FlightNum	Ar
398	2008	1	3	4	708.0	WN	457	
399	2008	1	3	4	NaN	WN	469	
400	2008	1	3	4	2320.0	WN	593	
401	2008	1	3	4	NaN	WN	618	
402	2008	1	3	4	2008.0	WN	646	
403	2008	1	3	4	1625.0	WN	656	

404	2008	1	3	4	1305.0	WN	680
405	2008	1	3	4	1558.0	WN	776
1896	2008	1	3	4	1800.0	WN	442
1897	2008	1	3	4	852.0	WN	475
1898	2008	1	3	4	1053.0	WN	553
1899	2008	1	3	4	1801.0	WN	578
1900	2008	1	3	4	1447.0	WN	750
1901	2008	1	3	4	2156.0	WN	777
1902	2008	1	3	4	NaN	WN	797
1903	2008	1	3	4	NaN	WN	801
3000	2008	1	4	5	701.0	WN	1
3001	2008	1	4	5	737.0	WN	3
3002	2008	1	4	5	758.0	WN	5
3003	2008	1	4	5	826.0	WN	7
3004	2008	1	4	5	858.0	WN	9
3005	2008	1	4	5	930.0	WN	11
3006	2008	1	4	5	NaN	WN	15
3007	2008	1	4	5	1105.0	WN	19
3008	2008	1	4	5	1133.0	WN	21
3009	2008	1	4	5	1210.0	WN	23
3010	2008	1	4	5	1231.0	WN	25
3011	2008	1	4	5	1300.0	WN	27
3012	2008	1	4	5	1338.0	WN	29
3013	2008	1	4	5	1414.0	WN	31
...	...	...	...	...	...	...	...

```
flights_delayed_filtered['Origin'].unique()
```

```
array(['LAX', 'SFO', 'DAL', 'HOU', 'ORD'], dtype=object)
```

```
flights_delayed_filtered.groupby(['Origin', 'Dest'])['DepDelay'].count()
```

```
Origin  Dest
DAL     HOU      9350
HOU     DAL      9259
LAX     SFO     12935
ORD     LGA      9948
SFO     LAX     13299
Name: DepDelay, dtype: int64
```

```
flights_delayed_filtered = pd.DataFrame({'count' : flights_delayed_filtered.loc[(flights
```

```
flights_delayed_filtered['count'].sum()
```

```
668
```

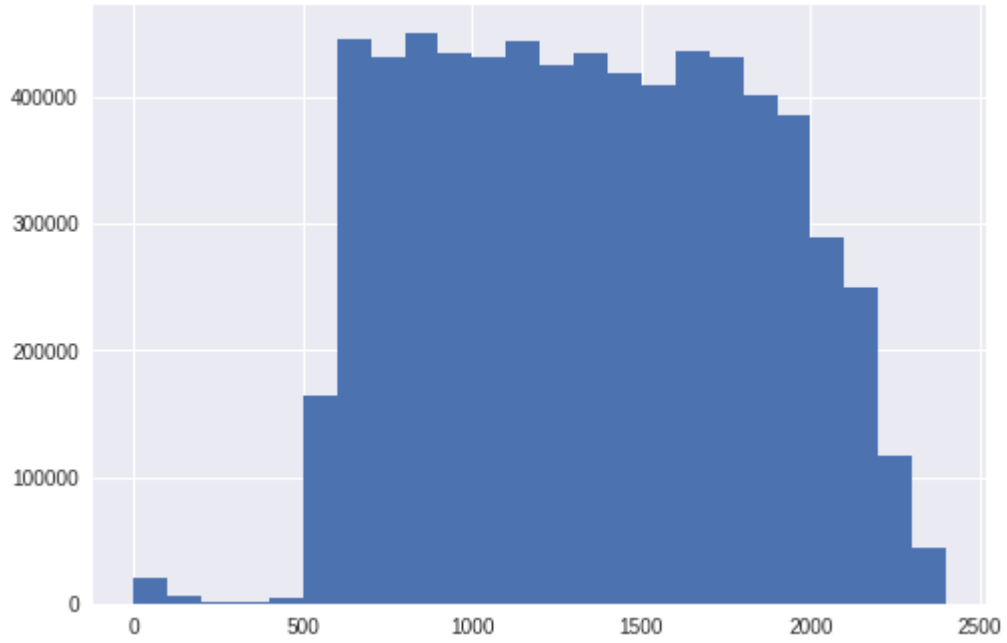
5. Examine the hourly distribution of departure times. For that, create a new series from DepTime, removing missing values.

Choose all correct statements:

- ***Flights are normally distributed within time interval [0-23] (Search for: Normal distribution, bell curve).***
- Flights are uniformly distributed within time interval [0-23].
- ***In the period from 0 am to 4 am there are considerably less flights than from 7 pm to 8 pm.***

```
# You code here
flights_df['DepTime'].hist(bins=24)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4d3e946f98>
```



6. Show how the number of flights changes through time (on the daily/weekly/monthly basis) and interpret the findings.

Choose all correct statements:

- The number of flights during weekends is less than during weekdays (working days).
- The lowest number of flights is on Sunday.
- There are less flights during winter than during summer.

```
# You code here
```

7. Examine the distribution of cancellation reasons with time. Make a bar plot of cancellation reasons aggregated by months.

Choose all correct statements:

- December has the highest rate of cancellations due to weather.
- The highest rate of cancellations in September is due to Security reasons.
- April's top cancellation reason is carriers.
- Flights cancellations due to National Air System are more frequent than those due to carriers.

```
# You code here
```

8. Which month has the greatest number of cancellations due to Carrier?

- May
- January
- September
- April

```
# You code here
```

9. Identify the carrier with the greatest number of cancellations due to carrier in the corresponding month from the previous question.

- 9E
- EV
- HA
- AA

```
# You code here
```

10. Examine median arrival and departure delays (in time) by carrier. Which carrier has the lowest median delay time for both arrivals and departures? Leave only non-negative values of delay times ('ArrDelay', 'DepDelay'). Boxplots can be helpful in this exercise, as well as it might be a good idea to remove outliers in order to build nice graphs. You can exclude delay time values higher than a corresponding .95 percentile.

- EV
- OO
- AA

- AQ

```
# You code here
```