Hello guys!

As data analyst sometimes you will be given data by a company or someone to do analysis on but the data might not be organized and contains some impurities, unormalies,missing values, and so many junks, so you need to make everything right by cleaning the the whole of the data set.Before any kind analysis the data given to do analysis on must be cleaned for proper results, and the following bellow are 11 simple steps to follow in data cleaning in pandas with python;

1. Renaming of Columns
2. Re-arranging Column Order
3. Checking data types of specific columns.
4. Checking and removal of outliners
5. Removing Text from column
6. Dealing with Missing Data
7. Changing Data Types
8. Replacing Text within a column
9. String operations of column data
10. Removing Columns
11. Dropping Rows

These are not the ony steps in data cleaning but it depends on how dirty your dataset is. Add yours...

Pandas - Data Cleaning

Let's load a new dataset on the number of fires in the Amazon rainforest

Let's load a new dataset on the number of fires in the Amazon rainforest

import pandas as pd from IPython.core.display import display,HTML display(HTML(""))#this will change the with of the notebook

file_name = "https://raw.githubusercontent.com/rajeevratan84/datascienceforbusiness/master/amazon_fires.csv" df = pd.read_csv(file_name, encoding = "ISO-8859-1")

df.tail()

In [29]:
```
# How many regions are in the dataset?

df['estado'].value_counts()
```

Out[29]:
```
Rio                 717
Paraiba             478
Mato Grosso         478
alagoas             240
Acre                239
Sergipe             239
Sao Paulo           239
Santa Catarina      239
Roraima             239
rondonia            239
Piau                239
Pernambuco          239
Minas Gerais        239
pará                239
Maranhao            239
Goias               239
Espirito Santo      239
Distrito Federal    239
Ceara               239
Bahia               239
Amazonas            239
Amapa               239
Tocantins           239
Name: estado, dtype: int64
```

Renaming Columns

In [30]:
```
new_columns = {'ano' : 'year',
               'estado': 'state',
               'mes': 'month',
               'numero': 'number_of_fires',
               'encontro': 'date'}

df.rename(columns = new_columns, inplace=True)
```

In [31]:
```
df.head()
```

Out[31]:

	year	month	state	number_of_fires	date
0	1998	Janeiro	Acre	0 Fires	1/1/1998
1	1999	Janeiro	Acre	0 Fires	1/1/1999
2	2000	Janeiro	Acre	0 Fires	1/1/2000
3	2001	Janeiro	Acre	0 Fires	1/1/2001
4	2002	Janeiro	Acre	0 Fires	1/1/2002

In [32]:
```
# How many years of data do we have?
df['year'].unique()
```

Out[32]:
```
array([1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008,
       2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017], dtype=int64)
```

In [33]:
```
# Let's explore our datetypes, we should expect number_of_types to be an integer or float datatype

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6454 entries, 0 to 6453
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   year             6454 non-null   int64
 1   month            6454 non-null   object
 2   state            6454 non-null   object
 3   number_of_fires  6322 non-null   object
 4   date             6454 non-null   object
dtypes: int64(1), object(4)
memory usage: 252.2+ KB
```

In [34]:
```
df.head()
```

Out[34]:

	year	month	state	number_of_fires	date
0	1998	Janeiro	Acre	0 Fires	1/1/1998
1	1999	Janeiro	Acre	0 Fires	1/1/1999
2	2000	Janeiro	Acre	0 Fires	1/1/2000
3	2001	Janeiro	Acre	0 Fires	1/1/2001
4	2002	Janeiro	Acre	0 Fires	1/1/2002

In [8]:
```
Ranamin of columnn
```

```
  Input In [8]
    Ranamin of columnn
            ^
SyntaxError: invalid syntax
```

In [35]:
```
cd=df.rename(columns={'state':'Hamza'})
cd
```

Out[35]:

	year	month	Hamza	number_of_fires	date
0	1998	Janeiro	Acre	0 Fires	1/1/1998
1	1999	Janeiro	Acre	0 Fires	1/1/1999
2	2000	Janeiro	Acre	0 Fires	1/1/2000
3	2001	Janeiro	Acre	0 Fires	1/1/2001
4	2002	Janeiro	Acre	0 Fires	1/1/2002
...	...	...	...	...	...
6449	2012	Dezembro	Tocantins	128	1/1/2012
6450	2013	Dezembro	Tocantins	85	1/1/2013
6451	2014	Dezembro	Tocantins	223	1/1/2014
6452	2015	Dezembro	Tocantins	373	1/1/2015
6453	2016	Dezembro	Tocantins	119	1/1/2016

6454 rows × 5 columns

In [ ]:
```
CHANGING THE ARRANGEMENT OF THE COL
```

In [36]:
```
df[df.columns[[4,1,0,2,3]]].head()
```

Out[36]:

	date	month	year	state	number_of_fires
0	1/1/1998	Janeiro	1998	Acre	0 Fires
1	1/1/1999	Janeiro	1999	Acre	0 Fires
2	1/1/2000	Janeiro	2000	Acre	0 Fires
3	1/1/2001	Janeiro	2001	Acre	0 Fires
4	1/1/2002	Janeiro	2002	Acre	0 Fires

Re-arranging columns

In [37]:
```
# Columns are numbered from 0, left to right
# Let's put date first, month second and year 3rd

new_order = [4,1,0,2,3,]
df = df[df.columns[new_order]]
df.head()
```

Out[37]:

	date	month	year	state	number_of_fires
0	1/1/1998	Janeiro	1998	Acre	0 Fires
1	1/1/1999	Janeiro	1999	Acre	0 Fires
2	1/1/2000	Janeiro	2000	Acre	0 Fires
3	1/1/2001	Janeiro	2001	Acre	0 Fires
4	1/1/2002	Janeiro	2002	Acre	0 Fires

In [38]:
```
df.head(25)
```

Out[38]:

	date	month	year	state	number_of_fires
0	1/1/1998	Janeiro	1998	Acre	0 Fires
1	1/1/1999	Janeiro	1999	Acre	0 Fires
2	1/1/2000	Janeiro	2000	Acre	0 Fires
3	1/1/2001	Janeiro	2001	Acre	0 Fires
4	1/1/2002	Janeiro	2002	Acre	0 Fires
5	1/1/2003	Janeiro	2003	Acre	10 Fires
6	1/1/2004	Janeiro	2004	Acre	0 Fires
7	1/1/2005	Janeiro	2005	Acre	12 Fires
8	1/1/2006	Janeiro	2006	Acre	4 Fires
9	1/1/2007	Janeiro	2007	Acre	0 Fires
10	1/1/2008	Janeiro	2008	Acre	0 Fires
11	1/1/2009	Janeiro	2009	Acre	0 Fires
12	1/1/2010	Janeiro	2010	Acre	1 Fires
13	1/1/2011	Janeiro	2011	Acre	0 Fires
14	1/1/2012	Janeiro	2012	Acre	0 Fires
15	1/1/2013	Janeiro	2013	Acre	0 Fires
16	1/1/2014	Janeiro	2014	Acre	0 Fires
17	1/1/2015	Janeiro	2015	Acre	1 Fires
18	1/1/2016	Janeiro	2016	Acre	12 Fires
19	1/1/2017	Janeiro	2017	Acre	0 Fires
20	1/1/1998	Fevereiro	1998	Acre	0 Fires
21	1/1/1999	Fevereiro	1999	Acre	0 Fires
22	1/1/2000	Fevereiro	2000	Acre	0 Fires
23	1/1/2001	Fevereiro	2001	Acre	0 Fires
24	1/1/2002	Fevereiro	2002	Acre	1 Fires

In [39]:
```
df.tail()
```

Out[39]:

	date	month	year	state	number_of_fires
6449	1/1/2012	Dezembro	2012	Tocantins	128
6450	1/1/2013	Dezembro	2013	Tocantins	85
6451	1/1/2014	Dezembro	2014	Tocantins	223
6452	1/1/2015	Dezembro	2015	Tocantins	373
6453	1/1/2016	Dezembro	2016	Tocantins	119

Determing if a column contains numeric data

In [40]:
```
# It isn't, let's find our why

df['number_of_fires'].str.isnumeric()
```

Out[40]:
```
0       False
1       False
2       False
3       False
4       False
        ...
6449     True
6450     True
6451     True
6452     True
6453     True
Name: number_of_fires, Length: 6454, dtype: object
```

In [41]:
```
# We get the above error because our isdigit() returns Nan for blank or missing values
# To fix this we need to convert our column datatype from non-null objects to a String

df['number_of_fires'].astype(str).str.isdigit()
```

Out[41]:
```
0       False
1       False
2       False
3       False
4       False
        ...
6449     True
6450     True
6451     True
6452     True
6453     True
Name: number_of_fires, Length: 6454, dtype: bool
```

- Bascially, `str.isdigit` only returns True for strings containing solely the digits 0-9.
- By contrast, `str.isnumeric` returns True if it contains any numeric characters. e.g. '½'

Removing unnecessary text from columns

In [42]:
```
df['number_of_fires'].str.strip(" Fires")
```

Out[42]:
```
0         0
1         0
2         0
3         0
4         0
       ...
6449    128
6450     85
6451    223
6452    373
6453    119
Name: number_of_fires, Length: 6454, dtype: object
```

Strip - Return a copy of the string with leading and trailing characters removed. If chars is omitted or None, whitespace characters are removed. If given and not None, chars must be a string; the characters in the string will be stripped from the both ends of the string this method is called on.

In [48]:
```
# To replace column with cleaned column

df['number_of_fires'] = df['number_of_fires'].str.strip('Fires')
df.head()
```

Out[48]:

	date	month	year	state	number_of_fires
0	1/1/1998	Janeiro	1998	Acre	0
1	1/1/1999	Janeiro	1999	Acre	0
2	1/1/2000	Janeiro	2000	Acre	0
3	1/1/2001	Janeiro	2001	Acre	0
4	1/1/2002	Janeiro	2002	Acre	0

In [44]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6454 entries, 0 to 6453
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   date             6454 non-null   object
 1   month            6454 non-null   object
 2   year             6454 non-null   int64
 3   state            6454 non-null   object
 4   number_of_fires  6322 non-null   object
dtypes: int64(1), object(4)
memory usage: 252.2+ KB
```

In [53]:
```
# We need to convert our number_of_fires column to a float data type
# Also, here's an alternative string manipulation technique we can use

df["number_of_fires"] = df["number_of_fires"].astype(float)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6454 entries, 0 to 6453
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   date             6454 non-null   object
 1   month            6454 non-null   object
 2   year             6454 non-null   int64
 3   state            6454 non-null   object
 4   number_of_fires  6322 non-null   float64
dtypes: float64(1), int64(1), object(3)
memory usage: 252.2+ KB
```

In [ ]:

In [ ]:
```
# That was one way we could have handled blank data
```

Handling missing data

In [54]:
```
# Let's reload our dataframe
file_name = "https://raw.githubusercontent.com/rajeevratan84/datascienceforbusiness/master/amazon_fires.csv"
df = pd.read_csv(file_name, encoding = "ISO-8859-1")
new_columns = {'ano' : 'year',
               'estado': 'state',
               'mes': 'month',
               'numero': 'number_of_fires',
               'encontro': 'date'}
df.rename(columns = new_columns, inplace=True)
df['number_of_fires'] = df['number_of_fires'].str.strip(" Fires")
# Creating a true copy of our dataframe
df_copy = df.copy()
df.head()
```

Out[54]:

	year	month	state	number_of_fires	date
0	1998	Janeiro	Acre	0	1/1/1998
1	1999	Janeiro	Acre	0	1/1/1999
2	2000	Janeiro	Acre	0	1/1/2000
3	2001	Janeiro	Acre	0	1/1/2001
4	2002	Janeiro	Acre	0	1/1/2002

In [56]:
```
# Viewing the sum of missing values in each column

df.isnull().sum()
```

Out[56]:
```
index              0
year               0
month              0
state              0
number_of_fires    0
date               0
dtype: int64
```

In [55]:
```
# We can easily remove Null or NaN (not a number) values

# Drop rows with NaN values
df = df.dropna()
df = df.reset_index() # reset's row indexes in case any rows were dropped
df.head()
```

Out[55]:

	index	year	month	state	number_of_fires	date
0	0	1998	Janeiro	Acre	0	1/1/1998
1	1	1999	Janeiro	Acre	0	1/1/1999
2	2	2000	Janeiro	Acre	0	1/1/2000
3	3	2001	Janeiro	Acre	0	1/1/2001
4	4	2002	Janeiro	Acre	0	1/1/2002

In [57]:
```
# Let's check and see it worked

df.isnull().sum()
```

Out[57]:
```
index              0
year               0
month              0
state              0
number_of_fires    0
date               0
dtype: int64
```

In [58]:
```
# Alright so it worked, now let's reload the data and look at a few other methods of dealing with NaN or Null values

# Let's reload our dataframe
file_name = "https://raw.githubusercontent.com/rajeevratan84/datascienceforbusiness/master/amazon_fires.csv"
df = pd.read_csv(file_name, encoding = "ISO-8859-1")
new_columns = {'ano' : 'year',
               'estado': 'state',
               'mes': 'month',
               'numero': 'number_of_fires',
               'encontro': 'date'}
df.rename(columns = new_columns, inplace=True)
df['number_of_fires'] = df['number_of_fires'].str.strip(" Fires")
df_copy = df.copy()
df.head()
```

Out[58]:

	year	month	state	number_of_fires	date
0	1998	Janeiro	Acre	0	1/1/1998
1	1999	Janeiro	Acre	0	1/1/1999
2	2000	Janeiro	Acre	0	1/1/2000
3	2001	Janeiro	Acre	0	1/1/2001
4	2002	Janeiro	Acre	0	1/1/2002

In [59]:
```
# Create a boolean index for all null values
d=df['number_of_fires'].isnull()
d
```

Out[59]:
```
0       False
1       False
2       False
3       False
4       False
        ...
6449    False
6450    False
6451    False
6452    False
6453    False
Name: number_of_fires, Length: 6454, dtype: bool
```

In [60]:
```
df[df['number_of_fires'].isnull()].head(10)
```

Out[60]:

	year	month	state	number_of_fires	date
68	2006	Abril	Acre	NaN	1/1/2006
110	2008	Junho	Acre	NaN	1/1/2008
127	2005	Julho	Acre	NaN	1/1/2005
206	2004	Novembro	Acre	NaN	1/1/2004
217	2015	Novembro	Acre	NaN	1/1/2015
444	2002	Novembro	alagoas	NaN	1/1/2002
522	2001	Março	Amapa	NaN	1/1/2001
550	2009	Abril	Amapa	NaN	1/1/2009
614	2013	Julho	Amapa	NaN	1/1/2013
642	2001	Setembro	Amapa	NaN	1/1/2001

What do to with missing data?

- Remove them via .dropna(axis=0)
- Replace them with some arbitary number (e.g. an average)
- Replace them zeros, or Forward Fill (ffill) or Back Fill (backfill)

In [61]:
```
# Using fillna with zeros

df['number_of_fires'].fillna(0).head()
```

Out[61]:
```
0    0
1    0
2    0
3    0
4    0
Name: number_of_fires, dtype: object
```

In [62]:
```
# Let's try back filling
df['number_of_fires'].fillna(method='ffill').head(70)
```

Out[62]:
```
0     0
1     0
2     0
3     0
4     0
     ..
65    1
66    2
67    1
68    1
69    0
Name: number_of_fires, Length: 70, dtype: object
```

In [ ]:

In [63]:
```
# View index 444 to see how it changes
# Homework, change 444 using ffill and backfill to see how it changes
bfill=df['number_of_fires'].fillna(method='backfill')
bfill.iloc[444]
```

Out[63]:
```
'17'
```

In [ ]:

In [ ]:
```
df.iloc[445]
```

In [ ]:
```
# let's make the assumption that blank values are 0 fires

# let's get back our copy of our original pre-processed datafrmae
df = df_copy

# replace all missing or NaN values with 0
df['number_of_fires'] = df['number_of_fires'].fillna(0)
```

In [ ]:
```
# Let's check to see if we did change our Nans to 0s
df.iloc[444]
```

Assigning data types to our columns

In [ ]:
```
df.info()
```

In [ ]:
```
df["number_of_fires"] = df["number_of_fires"].str.replace('','0').astype(float)
```

In [ ]:
```
df.head()
```

In [ ]:
```
df.info()
```

In [ ]:
```
df['month'].unique()
```

Replacing text in columns

In [ ]:
```
# Let's convert our Portuguese month names to English

month_translations = {'Janeiro': 'January',
'Fevereiro': 'February',
'Março': 'March',
'Abril': 'April',
'Maio': 'May',
'Junho': 'June',
'Julho': 'July',
'Agosto': 'August',
'Setembro': 'September',
'Outubro': 'October',
'Novembro': 'November',
'Dezembro': 'December'}

df["month"] = df["month"].map(month_translations)
df.head()
```

In [ ]:
```
base={
    'Acre':'HAMZA',
    'Alagoa':'JAARAH',
    'Amapa':'PANDO',
    'Amazonas':'TAKWA',
    'Bahia':'SUNYANI',
    'Ceara':'TAMALE',
    'Distrito Federal':'MALLAM',
    'Espirito Santo':'JACK',
    'Goias':'HFJJ',
    'Maranhao':'SANTA',
       'Mato Grosso':'GOOSE',
    'Minas Gerais':'GHANA',
    'Pará':'GDFJFJ',
    'Paraiba':'NIMA',
    'Pernambuco':'HAMRAAZ',
       'Piau':'KASOA',
    'Rio':'sass',
    'OPE':'CIRCLE',
    'Rondonia':'RONALDO',
    'Roraima':'SALAGA',
    'Santa Catarina':'hello',
       'Sao Paulo':'GEJ',
    'Sergipe':'SAB',
    'Tocantins':'MESSI'
}

df["state"] = df["state"].map(base)
df.head()
```

In [ ]:
```
df['state'].unique()
```

In [ ]:
```
df.isnull().sum()
```

Further string functions on columns

In [ ]:
```
df['state'] = df['state'].str.title()
df['state'].unique()
```

Removing columns

In [ ]:
```
df.head()
```

In [ ]:
```
# Dropping multiple columns
df = df.drop("date", axis=1) # axis = 1 so that it works across our columns
df.head()
```

In [ ]:
```
# Let's reload the data

# Let's reload our dataframe
file_name = "https://raw.githubusercontent.com/rajeevratan84/datascienceforbusiness/master/amazon_fires.csv"
df = pd.read_csv(file_name, encoding = "ISO-8859-1")
new_columns = {'ano' : 'year',
               'estado': 'state',
               'mes': 'month',
               'numero': 'number_of_fires',
               'encontro': 'date'}
df.rename(columns = new_columns, inplace=True)
df['number_of_fires'] = df['number_of_fires'].str.strip(" Fires")
df_copy = df.copy()
df.head()
```

In [ ]:
```
# Drop multiple columns
df = df.drop(["year", "date"], axis=1)
df.head()
```

Dropping Rows

Using the df.index function

In [ ]:
```
# Let's drop the first row
df = df.drop(df.index[0])
df = df.reset_index()
df.head()
```

In [ ]:
```
# Drop multiple rows

df = df.drop(df.index[[2,3]])
df.head()
```

In [ ]:
```
# Drop a range of rows

df = df.drop(df.index[1:4])
df.head()
```

In [ ]: