Data Cleaning Examples

Sample csv data is used for data cleaning and analysis

- The data is a randomnly generated data

Outline

```
- Load data and check columns
- Rename columns
- Add underscore in all columns
- Replace a character or empty space in column names
- Convert to ppercase/lowercase columns
- Select all columns except one
- Select columns of a particular order or phrase(df.filter)
- Select a group of column name
- User matplotlib to create a world map
- Overlay the data using log and lat on the worldmap
```

This is just a sample code

Used geopandas to map the data and overlay with world map.

Partically working but I am out of time for submission.

In [526]:
```
# Load Dataset
#!pip install geopandas
#!pip install descartes

import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import geopandas as gpd
```

In [527]:
```
# Load Dataset
df = pd.read_csv("rawdata3.csv")
```

In [528]:
```
# Firt Rows
df.head()
```

Out[528]:

	First Name	Last Name	Age	Salary	Email	City
0	Basil	Doris	39	68073	bibendum.sed@at.co.uk	High Level
1	Odysseus	Deirdre	60	76432	magna.nec@pedemalesuada.ca	Deschambault
2	Jesse	Amanda	45	57757	Quisque.imperdiet.erat@ullamcorpereueuismod.ca	Machilipatnam
3	Kareem	Charlotte	69	54158	scelerisque@dolorFuscemi.com	Snellegem
4	Mannix	Isadora	6	56973	velit@liberoestcongue.edu	Goes

List columns

In [529]:
```
# get a list of all Columns within the dataset
df.columns
```

Out[529]:
```
Index(['First Name', 'Last Name', 'Age', 'Salary', 'Email', 'City', 'Co
untry',
       'Long', 'Lat', 'Street Address0', 'Street Address1'],
      dtype='object')
```

In [530]:
```
## Features of Columns
# We use the dir() inbuilt function in Python3, which returns list of th
e attributes and methods of any object, in this case column
#dir(df.columns)
```

In [531]:
```
# Get a an array of the columns
df.columns.values
```

Out[531]:
```
array(['First Name', 'Last Name', 'Age', 'Salary', 'Email', 'City',
       'Country', 'Long', 'Lat', 'Street Address0', 'Street Address1'],
      dtype=object)
```

In [532]:
```
# Get The Columns As a List
df.columns.tolist()
```

Out[532]:
```
['First Name',
 'Last Name',
 'Age',
 'Salary',
 'Email',
 'City',
 'Country',
 'Long',
 'Lat',
 'Street Address0',
 'Street Address1']
```

In [533]:
```
### To View Columns Names
df.columns.view()
```

Out[533]:
```
Index(['First Name', 'Last Name', 'Age', 'Salary', 'Email', 'City', 'Co
untry',
       'Long', 'Lat', 'Street Address0', 'Street Address1'],
      dtype='object')
```

In [534]:
```
# Convert the Column Names To Series/ DataFrame
df.columns.to_series()
```

Out[534]:
```
First Name              First Name
Last Name                Last Name
Age                            Age
Salary                      Salary
Email                        Email
City                          City
Country                    Country
Long                          Long
Lat                            Lat
Street Address0    Street Address0
Street Address1    Street Address1
dtype: object
```

Convert the Column Names To DataFrame

In [535]:
```
# Convert the Column Names To DataFrame
df.columns.to_frame()
```

Out[535]:

	0
First Name	First Name
Last Name	Last Name
Age	Age
Salary	Salary
Email	Email
City	City
Country	Country
Long	Long
Lat	Lat
Street Address0	Street Address0
Street Address1	Street Address1

In [536]:
```
# Check to see if column names contains a phrase
df.columns.str.contains('Last Name')
```

Out[536]:
```
array([False,  True, False, False, False, False, False, False, False,
       False, False])
```

In [537]:
```
df['First Name']
```

Out[537]:
```
0        Basil
1     Odysseus
2        Jesse
3       Kareem
4       Mannix
        ...
95       Plato
96       Jacob
97      Walker
98     Wallace
99     Harding
Name: First Name, Length: 100, dtype: object
```

Check for duplicate column names

In [538]:
```
# Check for duplicate column names
df.columns.duplicated()
```

Out[538]:
```
array([False, False, False, False, False, False, False, False, False,
       False, False])
```

In [539]:
```
# Making Column Name Lower Case
df.columns.str.lower()
```

Out[539]:
```
Index(['first name', 'last name', 'age', 'salary', 'email', 'city', 'co
untry',
       'long', 'lat', 'street address0', 'street address1'],
      dtype='object')
```

In [540]:
```
# Making Column Name Upper Case
df.columns.str.upper()
```

Out[540]:
```
Index(['FIRST NAME', 'LAST NAME', 'AGE', 'SALARY', 'EMAIL', 'CITY', 'CO
UNTRY',
       'LONG', 'LAT', 'STREET ADDRESS0', 'STREET ADDRESS1'],
      dtype='object')
```

In [541]:
```
# Convert Column Name to Title Case
df.columns.str.title()
```

Out[541]:
```
Index(['First Name', 'Last Name', 'Age', 'Salary', 'Email', 'City', 'Co
untry',
       'Long', 'Lat', 'Street Address0', 'Street Address1'],
      dtype='object')
```

Replacing Empty spaces with underscore

In [542]:
```
# Replacing Empty spaces with underscore
df.columns.str.replace(' ','_')
```

Out[542]:
```
Index(['First_Name', 'Last_Name', 'Age', 'Salary', 'Email', 'City', 'Co
untry',
       'Long', 'Lat', 'Street_Address0', 'Street_Address1'],
      dtype='object')
```

In [543]:
```
# Renaming Column Name
```

In [544]:
```
# Renaming Column Name
df.rename(columns={'Age':'Date of Birth'}).head()
```

Out[544]:

	First Name	Last Name	Date of Birth	Salary	Email	City
0	Basil	Doris	39	68073	bibendum.sed@at.co.uk	High Level
1	Odysseus	Deirdre	60	76432	magna.nec@pedemalesuada.ca	Deschambault
2	Jesse	Amanda	45	57757	Quisque.imperdiet.erat@ullamcorpereueuismod.ca	Machilipatnam
3	Kareem	Charlotte	69	54158	scelerisque@dolorFuscemi.com	Snellegem
4	Mannix	Isadora	6	56973	velit@liberoestcongue.edu	Goes

In [545]:
```
# Renaming Column Name /Inplace
df.rename(columns={'Age':'Date of Birth'},inplace=True)
```

In [546]:
```
df.columns
```

Out[546]:
```
Index(['First Name', 'Last Name', 'Date of Birth', 'Salary', 'Email',
'City',
       'Country', 'Long', 'Lat', 'Street Address0', 'Street Address1'],
      dtype='object')
```

In [547]:
```
len(df.columns.values)
```

Out[547]: 11

Renaming Column Names using select values

In [548]:
```
# Renaming Column Names using select values
df.columns.values[5] = 'CITY'
```

In [549]:
```
df.columns
```

Out[549]:
```
Index(['First Name', 'Last Name', 'Date of Birth', 'Salary', 'Email',
'CITY',
       'Country', 'Long', 'Lat', 'Street Address0', 'Street Address1'],
      dtype='object')
```

Selecting All Columns Except One

In [550]:
```
# Selecting All Columns Except One
df.columns[df.columns != 'SALARY']
```

Out[550]:
```
Index(['First Name', 'Last Name', 'Date of Birth', 'Salary', 'Email',
'CITY',
       'Country', 'Long', 'Lat', 'Street Address0', 'Street Address1'],
      dtype='object')
```

In [551]:
```
# Selecting All Columns Except One
df.loc[:, df.columns != 'SALARY'].columns
```

Out[551]:
```
Index(['First Name', 'Last Name', 'Date of Birth', 'Salary', 'Email',
'CITY',
       'Country', 'Long', 'Lat', 'Street Address0', 'Street Address1'],
      dtype='object')
```

In [552]:
```
# Select Column Names Except One Using Difference
df.columns.difference(['SALARY'])
```

Out[552]:
```
Index(['CITY', 'Country', 'Date of Birth', 'Email', 'First Name', 'Last
Name',
       'Lat', 'Long', 'Salary', 'Street Address0', 'Street Address1'],
      dtype='object')
```

In [553]:
```
# Select Column Names Except One Using Negation of isin
df.loc[:,~df.columns.isin(['SALARY'])].columns
```

Out[553]:
```
Index(['First Name', 'Last Name', 'Date of Birth', 'Salary', 'Email',
'CITY',
       'Country', 'Long', 'Lat', 'Street Address0', 'Street Address1'],
      dtype='object')
```

Select Column Names that Begins with a Word or Character

In [554]:
```
# Select Column Names that Begins with a Word or Character
df.filter(like='STREET').columns
```

Out[554]:
```
Index([], dtype='object')
```

In [555]:
```
### Select Column Names that Begins with a Word or Character
df.loc[:,df.columns.str.startswith('STREET')].columns
```

Out[555]:
```
Index([], dtype='object')
```

In [556]:
```
### Select Column Names that ENDS with a Word or Character
df.loc[:,df.columns.str.endswith('ame')].columns
```

Out[556]:
```
Index(['First Name', 'Last Name'], dtype='object')
```

In [557]:
```
### Select Column Names that ENDS with a Word or Character Using Filter
 and Regex name$
df.filter(regex='ame$',axis=1).columns
```

Out[557]:
```
Index(['First Name', 'Last Name'], dtype='object')
```

In [558]:
```
### Select A Group of Column Names
df.columns.values[0:4]
```

Out[558]:
```
array(['First Name', 'Last Name', 'Date of Birth', 'Salary'], dtype=obj
ect)
```

In [559]:
```
### Select A Group of Column Names
df.columns[0:4]
```

Out[559]:
```
Index(['First Name', 'Last Name', 'Date of Birth', 'Salary'], dtype='ob
ject')
```

In [560]:
```
df.shape
```

Out[560]:
```
(100, 11)
```

In [561]:
```
df.head()
```

Out[561]:

	First Name	Last Name	Date of Birth	Salary	Email	CITY
0	Basil	Doris	39	68073	bibendum.sed@at.co.uk	High Level
1	Odysseus	Deirdre	60	76432	magna.nec@pedemalesuada.ca	Deschambault
2	Jesse	Amanda	45	57757	Quisque.imperdiet.erat@ullamcorpereueuismod.ca	Machilipatnam
3	Kareem	Charlotte	69	54158	scelerisque@dolorFuscemi.com	Snellegem
4	Mannix	Isadora	6	56973	velit@liberoestcongue.edu	Goes

In [562]:
```
df.columns
```

Out[562]:
```
Index(['First Name', 'Last Name', 'Date of Birth', 'Salary', 'Email',
'CITY',
       'Country', 'Long', 'Lat', 'Street Address0', 'Street Address1'],
      dtype='object')
```

In [563]:
```
df.melt(id_vars = ['First Name', 'Last Name', 'Date of Birth', 'Salary',
'Email', 'CITY','Country', 'Long', 'Lat', ]).tail()
```

Out[563]:

	First Name	Last Name	Date of Birth	Salary	Email	CITY	Country	Lo
195	Plato	Inga	45	84944	luctus@non.co.uk	Puerto López	Denmark	56.26
196	Jacob	Sage	95	48942	vel.venenatis.vel@euodioPhasellus.net	Erciş	Estonia	58.59
197	Walker	Hayfa	6	69620	ligula.Nullam.enim@Uttincidunt.ca	Hines Creek	Denmark	61.89
198	Wallace	Tara	10	20109	pede.Nunc.sed@Nunc.co.uk	San Massimo	Finland	64.00
199	Harding	Melodie	84	39535	imperdiet.nec.leo@lectusNullam.ca	Malbaie	Denmark	71.70

In [564]:
```
df_data1_melted = df.melt(id_vars = ['First Name', 'Last Name', 'Date of
Birth', 'Salary', 'Email', 'CITY','Country', 'Long', 'Lat', ])
```

In [565]:
```
df_data1_melted
```

Out[565]:

	First Name	Last Name	Date of Birth	Salary	Email	CIT
0	Basil	Doris	39	68073	bibendum.sed@at.co.uk	High Lev
1	Odysseus	Deirdre	60	76432	magna.nec@pedemalesuada.ca	Deschambau
2	Jesse	Amanda	45	57757	Quisque.imperdiet.erat@ullamcorpereueuismod.ca	Machilipatna
3	Kareem	Charlotte	69	54158	scelerisque@dolorFuscemi.com	Snellege
4	Mannix	Isadora	6	56973	velit@liberoestcongue.edu	Goe
...	...	...	...	...	...	
195	Plato	Inga	45	84944	luctus@non.co.uk	Puerto Lópe
196	Jacob	Sage	95	48942	vel.venenatis.vel@euodioPhasellus.net	Erc
197	Walker	Hayfa	6	69620	ligula.Nullam.enim@Uttincidunt.ca	Hines Cree
198	Wallace	Tara	10	20109	pede.Nunc.sed@Nunc.co.uk	San Massim
199	Harding	Melodie	84	39535	imperdiet.nec.leo@lectusNullam.ca	Malba

200 rows × 11 columns

In [566]:
```
# Renaming the last 2 columns.
# It is not 100% clean but it gives you the idea.
df_data1_melted.rename(columns={"variable":"Address", "value":"PO.Box"})
```

Out[566]:

	First Name	Last Name	Date of Birth	Salary	Email	CIT
0	Basil	Doris	39	68073	bibendum.sed@at.co.uk	High Lev
1	Odysseus	Deirdre	60	76432	magna.nec@pedemalesuada.ca	Deschambau
2	Jesse	Amanda	45	57757	Quisque.imperdiet.erat@ullamcorpereueuismod.ca	Machilipatna
3	Kareem	Charlotte	69	54158	scelerisque@dolorFuscemi.com	Snellege
4	Mannix	Isadora	6	56973	velit@liberoestcongue.edu	Goe
...	...	...	...	...	...	
195	Plato	Inga	45	84944	luctus@non.co.uk	Puerto Lópe
196	Jacob	Sage	95	48942	vel.venenatis.vel@euodioPhasellus.net	Erc
197	Walker	Hayfa	6	69620	ligula.Nullam.enim@Uttincidunt.ca	Hines Cree
198	Wallace	Tara	10	20109	pede.Nunc.sed@Nunc.co.uk	San Massim
199	Harding	Melodie	84	39535	imperdiet.nec.leo@lectusNullam.ca	Malba

200 rows × 11 columns

In [567]:
```
df_data1_melted.tail()
```

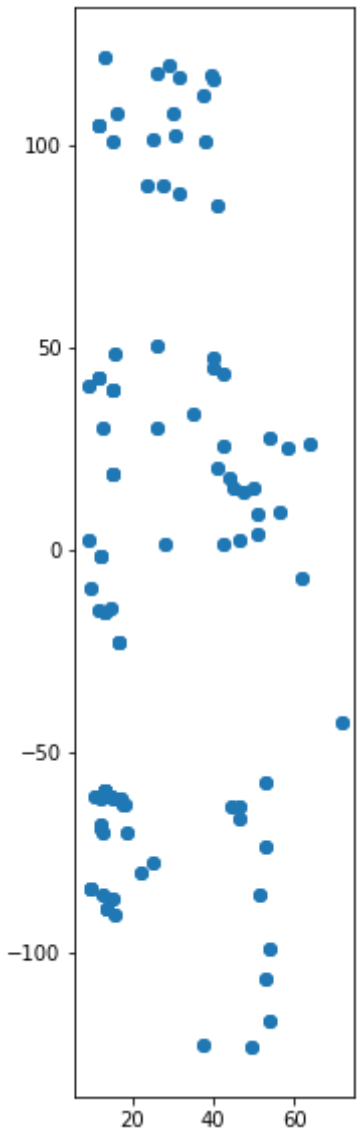
Out[567]:

	First Name	Last Name	Date of Birth	Salary	Email	CITY	Country	Lo
195	Plato	Inga	45	84944	luctus@non.co.uk	Puerto López	Denmark	56.26
196	Jacob	Sage	95	48942	vel.venenatis.vel@euodioPhasellus.net	Erciş	Estonia	58.59
197	Walker	Hayfa	6	69620	ligula.Nullam.enim@Uttincidunt.ca	Hines Creek	Denmark	61.89
198	Wallace	Tara	10	20109	pede.Nunc.sed@Nunc.co.uk	San Massimo	Finland	64.00
199	Harding	Melodie	84	39535	imperdiet.nec.leo@lectusNullam.ca	Malbaie	Denmark	71.70

In [568]:
```
gpd_df1 = gpd.GeoDataFrame(df_data1_melted, geometry=gpd.points_from_xy(
df_data1_melted['Long'], df_data1_melted['Lat']))
```

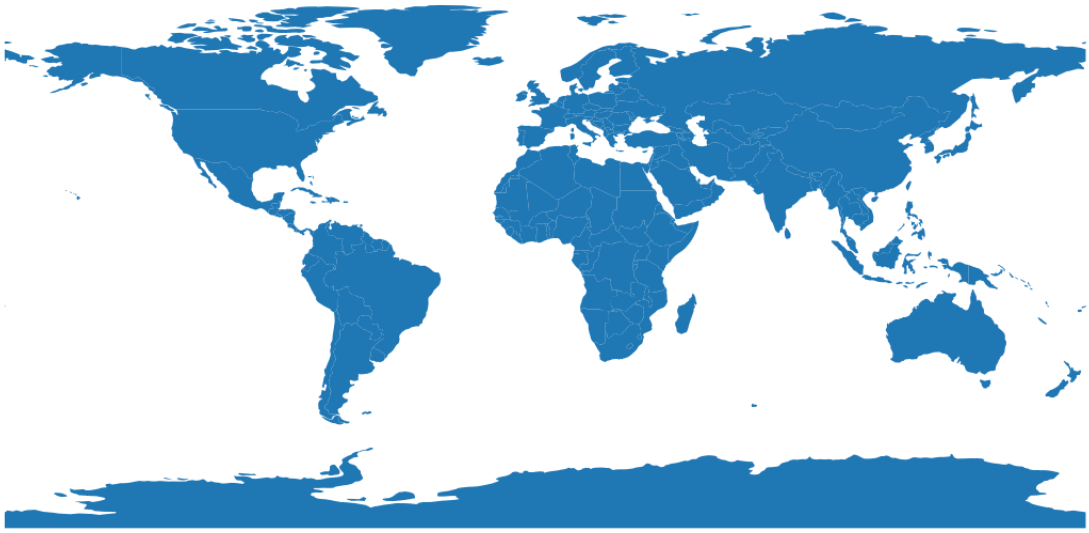
In [573]:
```
gpd_df1.plot(figsize=(20,10))
```

Out[573]: <matplotlib.axes._subplots.AxesSubplot at 0x12efa0518>



In [574]:
```
# Display the world map
world =gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
ax = world.plot(figsize=(20,10))
ax.axis('off')
```

Out[574]:
```
(-198.0, 198.00000000000006, -98.6822565, 92.32738650000002)
```

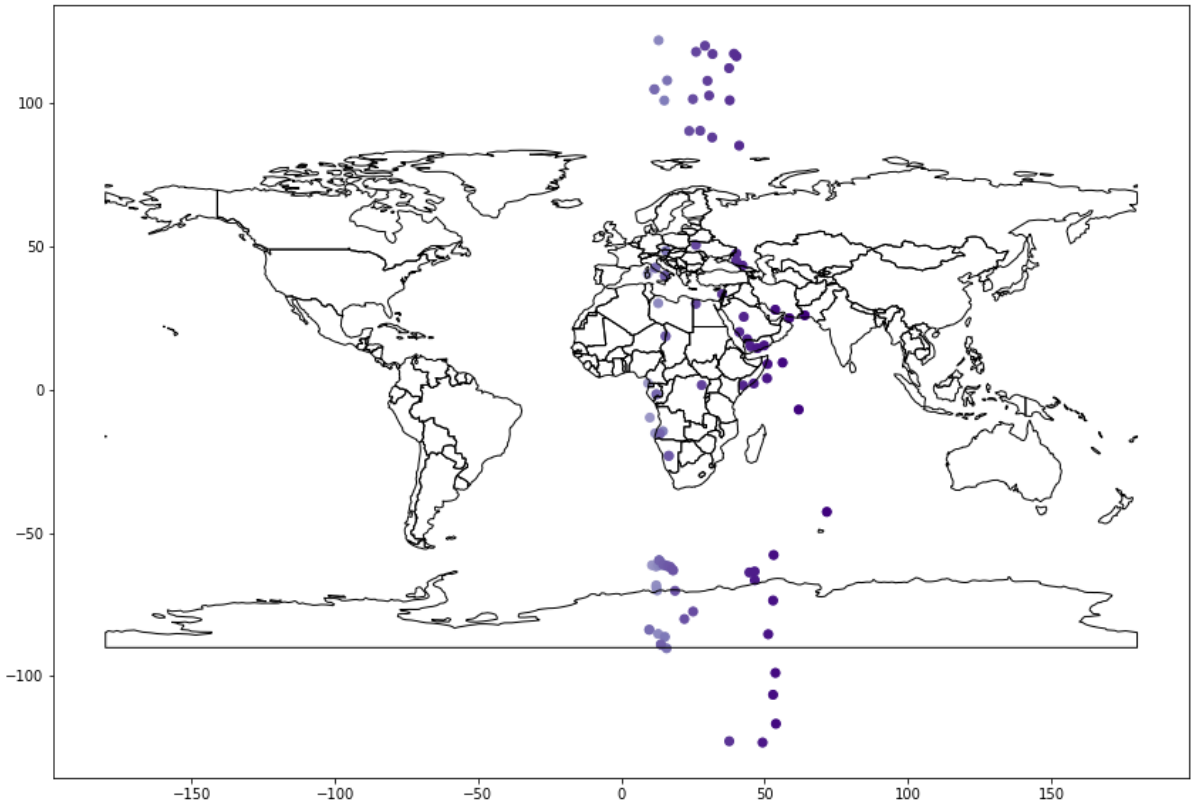


In [575]:
```
# Would like to debug and fix the overlay but I am running out of time.

# Over lap with our data

fig,ax = plt.subplots(figsize=(20,10))
gpd_df1.plot(cmap ='Purples', ax = ax)
world.geometry.boundary.plot(color = None, edgecolor = 'k', linewidth=1,
ax =ax)
```

Out[575]: <matplotlib.axes._subplots.AxesSubplot at 0x12ecfa4a8>



In [ ]:

In [ ]:

In [ ]: