Shrey Shah - Data Viz Project - Part 1

December 9, 2022

[498]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cartopy.crs as crs
import cartopy
import seaborn as sns

%matplotlib inline
```

[569]:
```
import plotly
import plotly.express as px
```

[1466]:
```
from IPython.display import display, HTML
```

0.1 Importing dataset

[1404]:
```
earth_quake = pd.read_csv("Earthquakes_In_NorthAmerica.tsv", sep='\t')
```

[1405]:
```
earth_quake
```

[1405]:

	Search Parameters	Year	Mo	Dy	Hr	Mn	Sec	Country	Location Name	Latitude	…
0	["Region = North America and Hawaii"]	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	…
1		NaN	1475.0	NaN	NaN	NaN	NaN	MEXICO	MEXICO: MEXICO CITY	NaN	…
2		NaN	1500.0	NaN	NaN	NaN	NaN	USA	HAWAII	NaN	…
3		NaN	1523.0	NaN	NaN	NaN					
4		NaN	1537.0	NaN	NaN	NaN					
..		…	…	…	…	…					
494		NaN	2021.0	7.0	29.0	6.0	15.0				
495		NaN	2021.0	9.0	8.0	1.0	47.0				
496		NaN	2022.0	5.0	25.0	21.0	43.0				
497		NaN	2022.0	9.0	19.0	18.0	5.0				
498		NaN	2022.0	9.0	22.0	6.0	16.0				

```
3     NaN  MEXICO                    MEXICO:  VERACRUZ    19.200  …
4     NaN  MEXICO                    MEXICO:  SOUTHERN       NaN  …
..    …     …                                    …       …  …
494  47.0     USA                     ALASKA PENINSULA    55.325  …
495  47.0  MEXICO                    MEXICO:  GUERRERO    16.982  …
496   2.0  MEXICO                      MEXICO:  OAXACA    16.325  …
497   6.0  MEXICO  MEXICO:  MICHOACAN, COLIMA, JALISCO    18.367  …
498   9.0  MEXICO       MEXICO: MEXICO CITY, MICHOACAN    18.308  …

     Injuries  Damage ($Mil)  Houses Destroyed  Houses Damaged  Total Deaths  \
0         NaN            NaN               NaN             NaN           NaN
1         NaN            NaN               NaN             NaN           NaN
2         NaN            NaN               NaN             NaN           NaN
3         NaN            NaN               NaN             NaN           NaN
4         NaN            NaN               NaN             NaN           NaN
..        …             …                 …               …           …
494       NaN            NaN               NaN             NaN           NaN
495       NaN          200.0               NaN          7317.0           3.0
496       NaN            NaN               NaN             NaN           NaN
497       NaN            NaN             800.0          6084.0           2.0
498       3.0            NaN               NaN             NaN           2.0

     Total Missing  Total Injuries  Total Damage ($Mil)  \
0              NaN             NaN                  NaN
1              NaN             NaN                  NaN
2              NaN             NaN                  NaN
3              NaN             NaN                  NaN
4              NaN             NaN                  NaN
..             …              …                   …
494            NaN             NaN                  NaN
495            NaN             NaN                200.0
496            NaN             NaN                  NaN
497            NaN             NaN                  NaN
498            NaN             3.0                  NaN

     Total Houses Destroyed  Total Houses Damaged
0                       NaN                   NaN
1                       NaN                   NaN
2                       NaN                   NaN
3                       NaN                   NaN
4                       NaN                   NaN
..                      …                    …
494                     NaN                   NaN
495                     NaN                7317.0
496                     NaN                   NaN
497                   800.0                6084.0
498                     NaN                   NaN
```

```
[499 rows x 26 columns]
```

0.2 General dataset properties

[1406]:
```
# Original dimensions of data

earth_quake.shape
```

[1406]:
```
(499, 26)
```

[1407]:
```
# Summary statistics of the numerical columns

earth_quake.describe()
```

[1407]:
```
              Year          Mo          Dy          Hr          Mn  \
count   498.000000  485.000000  481.000000  374.000000  366.000000
mean   1913.411647    6.274227   16.565489   11.596257   28.718579
std      97.481801    3.413188    9.120740    6.767695   17.414528
min    1475.000000    1.000000    1.000000    0.000000    0.000000
25%    1870.000000    3.000000    9.000000    6.000000   14.000000
50%    1930.500000    6.000000   17.000000   12.000000   29.000000
75%    1987.000000    9.000000   24.000000   17.000000   43.750000
max    2022.000000   12.000000   31.000000   23.000000   59.000000

              Sec    Latitude   Longitude  Focal Depth (km)         Mag  …  \
count  372.000000  489.000000  489.000000        261.000000  357.000000  …
mean    19.400269   32.930575 -106.910184         28.383142    6.726050  …
std     19.741008   14.540648   50.768601         27.559285    1.111189  …
min      0.000000   14.680000 -179.971000          0.000000    1.600000  …
25%      0.000000   18.200000 -122.400000         10.000000    6.100000  …
50%     13.300000   34.100000 -105.500000         20.000000    7.000000  …
75%     37.250000   42.375000  -97.782000         35.000000    7.600000  …
max     59.900000   73.122000  179.690000        170.000000    9.200000  …

          Injuries  Damage ($Mil)  Houses Destroyed  Houses Damaged  \
count     77.00000      87.000000         28.000000       26.000000
mean     710.00000     865.237931       4344.607143    12247.500000
std     3564.65645    4442.709878      12360.655117    39503.599036
min        1.00000       0.150000          1.000000       20.000000
25%        7.00000       3.000000         15.500000      106.750000
50%       30.00000      12.700000         95.500000      660.000000
75%      200.00000      83.500000        425.000000     2000.000000
max    30000.00000   40000.000000      47468.000000   184000.000000

       Total Deaths  Total Missing  Total Injuries  Total Damage ($Mil)  \
count    112.000000            1.0       77.000000            85.000000
```

```
mean     158.714286         1755.0      706.610390          909.888600
std      931.264029            NaN     3565.260222         4493.720014
min        1.000000         1755.0        1.000000            0.010000
25%        2.000000         1755.0        6.000000            3.000000
50%        7.000000         1755.0       29.000000           17.000000
75%       34.250000         1755.0      167.000000          100.000000
max     9500.000000         1755.0    30000.000000        40000.000000

       Total Houses Destroyed  Total Houses Damaged
count               25.000000             25.000000
mean              3763.320000          12728.200000
std              12085.829639          40240.506801
min                  1.000000             20.000000
25%                 16.000000            100.000000
50%                105.000000            720.000000
75%                500.000000           2000.000000
max              47468.000000         184000.000000

[8 rows x 23 columns]
```

[1408]:
```
earth_quake.columns
```

[1408]:
```
Index(['Search Parameters', 'Year', 'Mo', 'Dy', 'Hr', 'Mn', 'Sec', 'Country',
       'Location Name', 'Latitude', 'Longitude', 'Focal Depth (km)', 'Mag',
       'MMI Int', 'Deaths', 'Missing', 'Injuries', 'Damage ($Mil)',
       'Houses Destroyed', 'Houses Damaged', 'Total Deaths', 'Total Missing',
       'Total Injuries', 'Total Damage ($Mil)', 'Total Houses Destroyed',
       'Total Houses Damaged'],
      dtype='object')
```

[1409]:
```
earth_quake.dtypes
```

[1409]:
```
Search Parameters          object
Year                      float64
Mo                        float64
Dy                        float64
Hr                        float64
Mn                        float64
Sec                       float64
Country                    object
Location Name              object
Latitude                  float64
Longitude                 float64
Focal Depth (km)          float64
Mag                       float64
MMI Int                   float64
Deaths                    float64
```

```
Missing                   float64
Injuries                  float64
Damage ($Mil)             float64
Houses Destroyed          float64
Houses Damaged            float64
Total Deaths              float64
Total Missing             float64
Total Injuries            float64
Total Damage ($Mil)       float64
Total Houses Destroyed    float64
Total Houses Damaged      float64
dtype: object
```

[1410]:
```
# Infor of the dataset

earth_quake.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 499 entries, 0 to 498
Data columns (total 26 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Search Parameters       1 non-null      object
 1   Year                    498 non-null    float64
 2   Mo                      485 non-null    float64
 3   Dy                      481 non-null    float64
 4   Hr                      374 non-null    float64
 5   Mn                      366 non-null    float64
 6   Sec                     372 non-null    float64
 7   Country                 498 non-null    object
 8   Location Name           498 non-null    object
 9   Latitude                489 non-null    float64
 10  Longitude               489 non-null    float64
 11  Focal Depth (km)        261 non-null    float64
 12  Mag                     357 non-null    float64
 13  MMI Int                 222 non-null    float64
 14  Deaths                  104 non-null    float64
 15  Missing                 0 non-null      float64
 16  Injuries                77 non-null     float64
 17  Damage ($Mil)           87 non-null     float64
 18  Houses Destroyed        28 non-null     float64
 19  Houses Damaged          26 non-null     float64
 20  Total Deaths            112 non-null    float64
 21  Total Missing           1 non-null      float64
 22  Total Injuries          77 non-null     float64
 23  Total Damage ($Mil)     85 non-null     float64
 24  Total Houses Destroyed  25 non-null     float64
```

```
 25  Total Houses Damaged    25 non-null     float64
dtypes: float64(23), object(3)
memory usage: 101.5+ KB
```

0.3 Dataset Cleaning

[1411]:
```
# Removing the Search Parameters column
# Also deleting the row associated with Search Parameters

earth_quake = earth_quake.drop(columns='Search Parameters')
earth_quake = earth_quake.drop(labels=0, axis=0)
```

[1412]:
```
earth_quake
```

[1412]:

	Year	Mo	Dy	Hr	Mn	Sec	Country	\
1	1475.0	NaN	NaN	NaN	NaN	NaN	MEXICO	
2	1500.0	NaN	NaN	NaN	NaN	NaN	USA	
3	1523.0	NaN	NaN	NaN	NaN	NaN	MEXICO	
4	1537.0	NaN	NaN	NaN	NaN	NaN	MEXICO	
5	1538.0	NaN	NaN	NaN	NaN	NaN	MEXICO	
..	…	…	…	…	…	…	…	
494	2021.0	7.0	29.0	6.0	15.0	47.0	USA	
495	2021.0	9.0	8.0	1.0	47.0	47.0	MEXICO	
496	2022.0	5.0	25.0	21.0	43.0	2.0	MEXICO	
497	2022.0	9.0	19.0	18.0	5.0	6.0	MEXICO	
498	2022.0	9.0	22.0	6.0	16.0	9.0	MEXICO	

	Location Name	Latitude	Longitude	…	Injuries	\
1	MEXICO: MEXICO CITY	NaN	NaN	…	NaN	
2	HAWAII	NaN	NaN	…	NaN	
3	MEXICO: VERACRUZ	19.200	-96.400	…	NaN	
4	MEXICO: SOUTHERN	NaN	NaN	…	NaN	
5	MEXICO: MEXICO CITY	19.200	-99.100	…	NaN	
..	…	…	…	…	…	
494	ALASKA PENINSULA	55.325	-157.841	…	NaN	
495	MEXICO: GUERRERO	16.982	-99.773	…	NaN	
496	MEXICO: OAXACA	16.325	-95.856	…	NaN	
497	MEXICO: MICHOACAN, COLIMA, JALISCO	18.367	-103.252	…	NaN	
498	MEXICO: MEXICO CITY, MICHOACAN	18.308	-102.923	…	3.0	

	Damage ($Mil)	Houses Destroyed	Houses Damaged	Total Deaths	\
1	NaN	NaN	NaN	NaN	
2	NaN	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	
4	NaN	NaN	NaN	NaN	
5	NaN	NaN	NaN	NaN	
..	…	…	…	…	

```
494            NaN               NaN             NaN           NaN
495          200.0               NaN          7317.0           3.0
496            NaN               NaN             NaN           NaN
497            NaN             800.0          6084.0           2.0
498            NaN               NaN             NaN           2.0

     Total Missing  Total Injuries  Total Damage ($Mil)  \
1              NaN             NaN                  NaN
2              NaN             NaN                  NaN
3              NaN             NaN                  NaN
4              NaN             NaN                  NaN
5              NaN             NaN                  NaN
..             …               …                    …
494            NaN             NaN                  NaN
495            NaN             NaN                200.0
496            NaN             NaN                  NaN
497            NaN             NaN                  NaN
498            NaN             3.0                  NaN

     Total Houses Destroyed  Total Houses Damaged
1                       NaN                   NaN
2                       NaN                   NaN
3                       NaN                   NaN
4                       NaN                   NaN
5                       NaN                   NaN
..                      …                     …
494                     NaN                   NaN
495                     NaN                7317.0
496                     NaN                   NaN
497                   800.0                6084.0
498                     NaN                   NaN

[498 rows x 25 columns]
```

[1413]:
```
# Count of null values in all columns

for i in earth_quake.columns:
    print(i, sum(earth_quake[i].isna()), sep = "\t")
```

```
Year    0
Mo      13
Dy      17
Hr      124
Mn      132
Sec     126
Country 0
Location Name   0
```

```
Latitude        9
Longitude       9
Focal Depth (km)        237
Mag     141
MMI Int 276
Deaths  394
Missing 498
Injuries        421
Damage ($Mil)   411
Houses Destroyed        470
Houses Damaged  472
Total Deaths    386
Total Missing   497
Total Injuries  421
Total Damage ($Mil)     413
Total Houses Destroyed  473
Total Houses Damaged    473
```

[1414]:
```
# Converting the data types of Year, Mo, Dy, Hr, Mn, Sec to Float64

earth_quake = earth_quake.astype({"Year":"Float64","Mo":"Float64", "Dy":
↪"Float64", "Hr":"Float64",
                                  "Mn":"Float64", "Sec":"Float64"})
```

[1415]:
```
earth_quake
```

[1415]:

	Year	Mo	Dy	Hr	Mn	Sec	Country	\
1	1475.0	NaN	NaN	NaN	NaN	NaN	MEXICO	
2	1500.0	NaN	NaN	NaN	NaN	NaN	USA	
3	1523.0	NaN	NaN	NaN	NaN	NaN	MEXICO	
4	1537.0	NaN	NaN	NaN	NaN	NaN	MEXICO	
5	1538.0	NaN	NaN	NaN	NaN	NaN	MEXICO	
..	…	…	…	…	…	…	…	
494	2021.0	7.0	29.0	6.0	15.0	47.0	USA	
495	2021.0	9.0	8.0	1.0	47.0	47.0	MEXICO	
496	2022.0	5.0	25.0	21.0	43.0	2.0	MEXICO	
497	2022.0	9.0	19.0	18.0	5.0	6.0	MEXICO	
498	2022.0	9.0	22.0	6.0	16.0	9.0	MEXICO	

	Location Name	Latitude	Longitude	…	Injuries	\
1	MEXICO: MEXICO CITY	NaN	NaN	…	NaN	
2	HAWAII	NaN	NaN	…	NaN	
3	MEXICO: VERACRUZ	19.200	-96.400	…	NaN	
4	MEXICO: SOUTHERN	NaN	NaN	…	NaN	
5	MEXICO: MEXICO CITY	19.200	-99.100	…	NaN	
..	…	…	…	…	…	
494	ALASKA PENINSULA	55.325	-157.841	…	NaN	

```
495                    MEXICO:  GUERRERO    16.982    -99.773  …       NaN
496                      MEXICO:  OAXACA    16.325    -95.856  …       NaN
497  MEXICO:  MICHOACAN, COLIMA, JALISCO    18.367   -103.252  …       NaN
498       MEXICO: MEXICO CITY, MICHOACAN    18.308   -102.923  …       3.0

     Damage ($Mil)  Houses Destroyed  Houses Damaged  Total Deaths  \
1              NaN               NaN             NaN           NaN
2              NaN               NaN             NaN           NaN
3              NaN               NaN             NaN           NaN
4              NaN               NaN             NaN           NaN
5              NaN               NaN             NaN           NaN
..             …                 …               …             …
494            NaN               NaN             NaN           NaN
495          200.0               NaN          7317.0           3.0
496            NaN               NaN             NaN           NaN
497            NaN             800.0          6084.0           2.0
498            NaN               NaN             NaN           2.0

     Total Missing  Total Injuries  Total Damage ($Mil)  \
1              NaN             NaN                  NaN
2              NaN             NaN                  NaN
3              NaN             NaN                  NaN
4              NaN             NaN                  NaN
5              NaN             NaN                  NaN
..             …               …                    …
494            NaN             NaN                  NaN
495            NaN             NaN                200.0
496            NaN             NaN                  NaN
497            NaN             NaN                  NaN
498            NaN             3.0                  NaN

     Total Houses Destroyed  Total Houses Damaged
1                       NaN                   NaN
2                       NaN                   NaN
3                       NaN                   NaN
4                       NaN                   NaN
5                       NaN                   NaN
..                      …                     …
494                     NaN                   NaN
495                     NaN                7317.0
496                     NaN                   NaN
497                   800.0                6084.0
498                     NaN                   NaN

[498 rows x 25 columns]
```

[1416]:
```
# Converting the data types of Year, Mo, Dy, Hr, Mn to Int64

earth_quake = earth_quake.astype({"Year":"Int64","Mo":"Int64", "Dy":"Int64",␣
↪"Hr":"Int64",
                                  "Mn":"Int64", "Sec":"Float64"})
```

[1417]:
```
earth_quake
```

[1417]:

	Year	Mo	Dy	Hr	Mn	Sec	Country	\
1	1475	<NA>	<NA>	<NA>	<NA>	NaN	MEXICO	
2	1500	<NA>	<NA>	<NA>	<NA>	NaN	USA	
3	1523	<NA>	<NA>	<NA>	<NA>	NaN	MEXICO	
4	1537	<NA>	<NA>	<NA>	<NA>	NaN	MEXICO	
5	1538	<NA>	<NA>	<NA>	<NA>	NaN	MEXICO	
..	…	…	…	…	…	…	…	
494	2021	7	29	6	15	47.0	USA	
495	2021	9	8	1	47	47.0	MEXICO	
496	2022	5	25	21	43	2.0	MEXICO	
497	2022	9	19	18	5	6.0	MEXICO	
498	2022	9	22	6	16	9.0	MEXICO	

	Location Name	Latitude	Longitude	…	Injuries	\
1	MEXICO: MEXICO CITY	NaN	NaN	…	NaN	
2	HAWAII	NaN	NaN	…	NaN	
3	MEXICO: VERACRUZ	19.200	-96.400	…	NaN	
4	MEXICO: SOUTHERN	NaN	NaN	…	NaN	
5	MEXICO: MEXICO CITY	19.200	-99.100	…	NaN	
..	…	…	…	…	…	
494	ALASKA PENINSULA	55.325	-157.841	…	NaN	
495	MEXICO: GUERRERO	16.982	-99.773	…	NaN	
496	MEXICO: OAXACA	16.325	-95.856	…	NaN	
497	MEXICO: MICHOACAN, COLIMA, JALISCO	18.367	-103.252	…	NaN	
498	MEXICO: MEXICO CITY, MICHOACAN	18.308	-102.923	…	3.0	

	Damage ($Mil)	Houses Destroyed	Houses Damaged	Total Deaths	\
1	NaN	NaN	NaN	NaN	
2	NaN	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	
4	NaN	NaN	NaN	NaN	
5	NaN	NaN	NaN	NaN	
..	…	…	…	…	
494	NaN	NaN	NaN	NaN	
495	200.0	NaN	7317.0	3.0	
496	NaN	NaN	NaN	NaN	
497	NaN	800.0	6084.0	2.0	
498	NaN	NaN	NaN	2.0	

```
     Total Missing  Total Injuries  Total Damage ($Mil)  \
1              NaN             NaN                  NaN
2              NaN             NaN                  NaN
3              NaN             NaN                  NaN
4              NaN             NaN                  NaN
5              NaN             NaN                  NaN
..             …               …                    …
494            NaN             NaN                  NaN
495            NaN             NaN                200.0
496            NaN             NaN                  NaN
497            NaN             NaN                  NaN
498            NaN             3.0                  NaN

     Total Houses Destroyed  Total Houses Damaged
1                       NaN                   NaN
2                       NaN                   NaN
3                       NaN                   NaN
4                       NaN                   NaN
5                       NaN                   NaN
..                      …                     …
494                     NaN                   NaN
495                     NaN                7317.0
496                     NaN                   NaN
497                   800.0                6084.0
498                     NaN                   NaN

[498 rows x 25 columns]
```

[1418]:
```
# Converting the data types of columns

earth_quake = earth_quake.astype({"Focal Depth (km)":"Int64","Injuries":
↪"Int64", "Houses Destroyed":"Int64",
                                  "Deaths":"Int64", "Missing":"Int64","Houses␣
↪Damaged":"Int64",
                                  "Total Deaths":"Int64", "Total Missing":
↪"Int64",
                                  "Total Injuries":"Int64","Total Houses␣
↪Destroyed":"Int64",
                                  "Total Houses Damaged":"Int64"})
```

[1419]:
```
earth_quake.drop(columns=['Missing', 'Total Missing'], inplace=True)
```

[1420]:
```
earth_quake
```

[1420]:
```
     Year    Mo    Dy    Hr    Mn   Sec Country  \
1    1475  <NA>  <NA>  <NA>  <NA>   NaN  MEXICO
2    1500  <NA>  <NA>  <NA>  <NA>   NaN     USA
```

```
3    1523  <NA>  <NA>  <NA>  <NA>   NaN  MEXICO
4    1537  <NA>  <NA>  <NA>  <NA>   NaN  MEXICO
5    1538  <NA>  <NA>  <NA>  <NA>   NaN  MEXICO
..    …    …   …   …   …   …   …      …
494  2021     7    29     6    15  47.0     USA
495  2021     9     8     1    47  47.0  MEXICO
496  2022     5    25    21    43   2.0  MEXICO
497  2022     9    19    18     5   6.0  MEXICO
498  2022     9    22     6    16   9.0  MEXICO

                          Location Name  Latitude  Longitude  …  Deaths  \
1                   MEXICO:  MEXICO CITY       NaN        NaN  …    <NA>
2                                 HAWAII       NaN        NaN  …    <NA>
3                     MEXICO:  VERACRUZ    19.200    -96.400  …    <NA>
4                     MEXICO:  SOUTHERN       NaN        NaN  …    <NA>
5                   MEXICO:  MEXICO CITY    19.200    -99.100  …    <NA>
..                                   …       …        …  …     …
494                     ALASKA PENINSULA    55.325   -157.841  …    <NA>
495                   MEXICO:  GUERRERO    16.982    -99.773  …       3
496                     MEXICO:  OAXACA    16.325    -95.856  …    <NA>
497  MEXICO:  MICHOACAN, COLIMA, JALISCO    18.367   -103.252  …       2
498       MEXICO: MEXICO CITY, MICHOACAN    18.308   -102.923  …       2

     Injuries  Damage ($Mil)  Houses Destroyed  Houses Damaged  Total Deaths  \
1        <NA>            NaN              <NA>            <NA>          <NA>
2        <NA>            NaN              <NA>            <NA>          <NA>
3        <NA>            NaN              <NA>            <NA>          <NA>
4        <NA>            NaN              <NA>            <NA>          <NA>
5        <NA>            NaN              <NA>            <NA>          <NA>
..        …              …                …               …             …
494      <NA>            NaN              <NA>            <NA>          <NA>
495      <NA>          200.0              <NA>            7317             3
496      <NA>            NaN              <NA>            <NA>          <NA>
497      <NA>            NaN               800            6084             2
498         3            NaN              <NA>            <NA>             2

     Total Injuries  Total Damage ($Mil)  Total Houses Destroyed  \
1              <NA>                  NaN                    <NA>
2              <NA>                  NaN                    <NA>
3              <NA>                  NaN                    <NA>
4              <NA>                  NaN                    <NA>
5              <NA>                  NaN                    <NA>
..              …                    …                      …
494            <NA>                  NaN                    <NA>
495            <NA>                200.0                    <NA>
496            <NA>                  NaN                    <NA>
497            <NA>                  NaN                     800
```

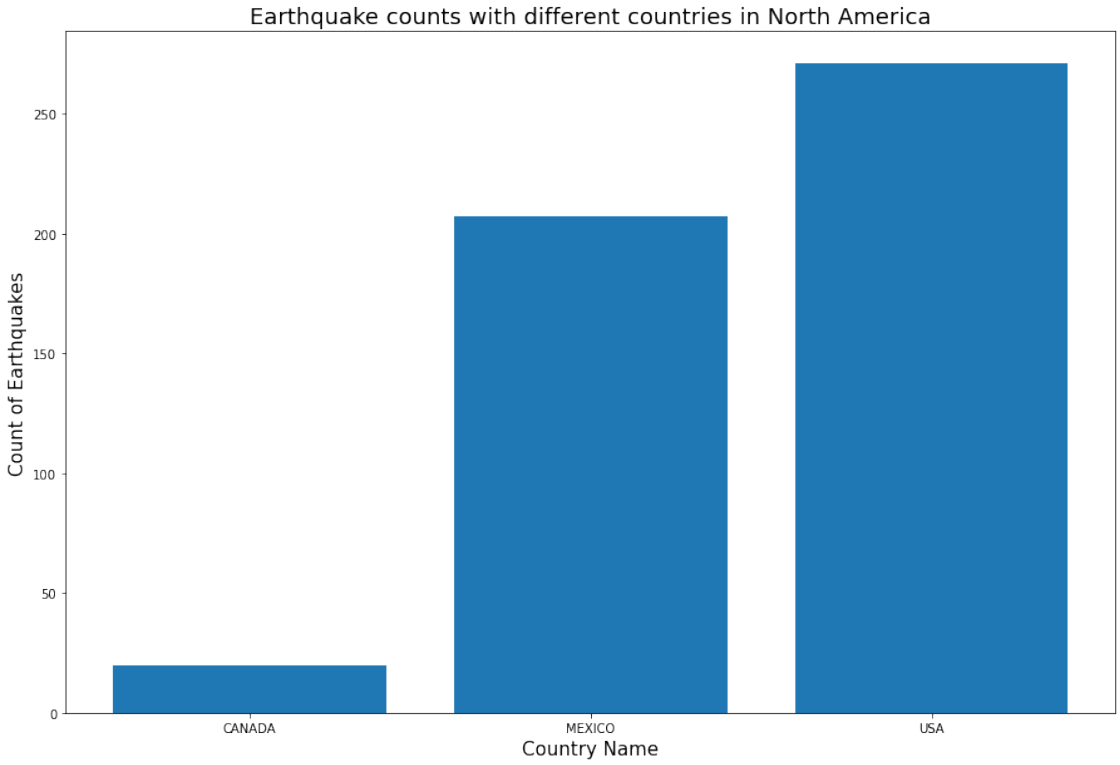
```
498               3                   NaN                      <NA>

     Total Houses Damaged
1                    <NA>
2                    <NA>
3                    <NA>
4                    <NA>
5                    <NA>
..                    …
494                  <NA>
495                  7317
496                  <NA>
497                  6084
498                  <NA>

[498 rows x 23 columns]
```

0.3.1 Earthquakes by different countries in North America

[1421]:
```
# Plotting the count of earthquakes by different countries of North America
# and seeing which country has the highest number of earthquakes

x = earth_quake.groupby(['Country'])[['Year']].count()
earthquake_by_country = x.reset_index()
earthquake_by_country
```

[1421]:
```
  Country  Year
0  CANADA    20
1  MEXICO   207
2     USA   271
```

[1422]:
```
fig, ax = plt.subplots(figsize = (15,10))
ax = plt.bar(earthquake_by_country['Country'], earthquake_by_country['Year'])
plt.xlabel("Country Name", fontsize=15)
plt.ylabel("Count of Earthquakes", fontsize=15)
plt.title("Earthquake counts with different countries in North America",␣
 ↪fontsize=18)

plt.show()
```



[1423]:
```
fig = px.bar(earthquake_by_country, 'Country', 'Year', width=1000, height=1200,
             title='Earthquake counts with different grouped locations in North␣
 ↪America',
             labels=dict(Place="Country Name", Year="Count of Earthquakes"))
fig.show('notebook')
```

Approach:

- In this case, I have firstly obtained a count of the earthquakes and then grouped them by Country Name.
- There are 3 countries - Canada, USA and Mexico
- I have used a simple bar chart to show the counts of earthquake with Country names.

0.3.2 Earthquakes by different regions in North America

[1424]:
```
earth_quake['Location Name'].value_counts()
```

[1424]:
```
MEXICO:  OAXACA                                28
MEXICO:  GUERRERO                              12
CALIFORNIA:  NORTHERN                          10
MEXICO:  ACAPULCO                               9
ALASKA:  ALEUTIAN ISLANDS:  FOX ISLANDS         9
                                               ..
```

```
CALIFORNIA:  OROVILLE                                1
NEW YORK: ROCKAWAY BEACH, NEAR NEW YORK CITY     1
CALIFORNIA:  NORTH:  HONEYDEW                        1
CALIFORNIA:  WINTERS                                 1
MEXICO:  SOUTHERN                                    1
Name: Location Name, Length: 315, dtype: int64
```

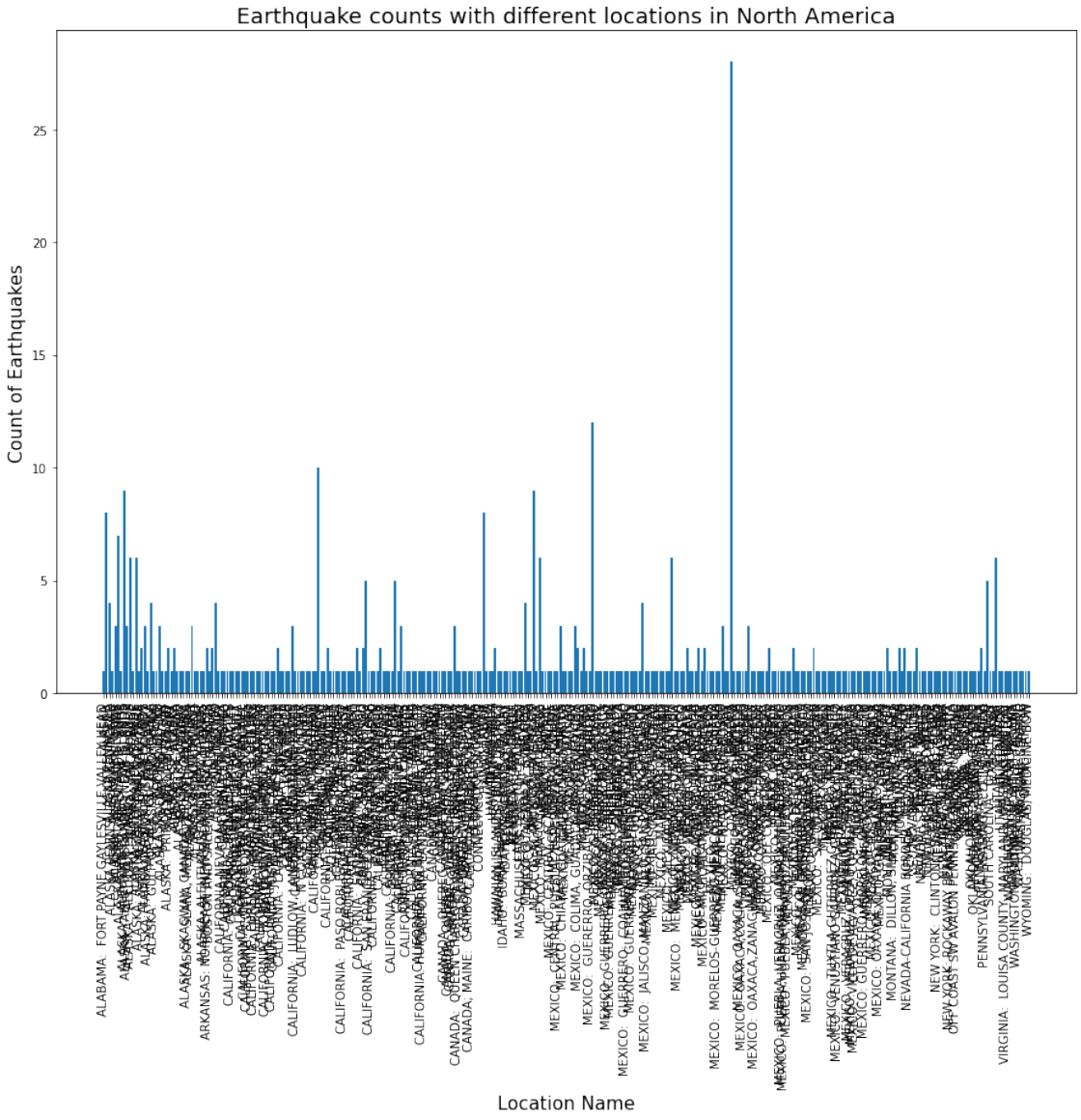
[1425]:
```
# Plotting the earthquake counts with respect to the region

x = earth_quake.groupby(['Location Name'])[['Year']].count()
earthquake_by_location = x.reset_index()
earthquake_by_location
```

[1425]:

	Location Name	Year
0	ALABAMA: FORT PAYNE,GAYLESVILLE,VALLEY HEAD	1
1	ALASKA	8
2	ALASKA PENINSULA	4
3	ALASKA PENINSULA: UNGA ISLAND	1
4	ALASKA: ALASKA PENINSULA	3
..	…	…
310	WASHINGTON: OLYMPIA, SEATTLE, TACOMA	1
311	WASHINGTON: PUGET SOUND	1
312	WASHINGTON: SEATTLE	1
313	WYOMING: AFTON	1
314	WYOMING: DOUGLAS, MEDICINE BOW	1

[315 rows x 2 columns]

[1426]:
```
fig, ax = plt.subplots(figsize = (15,10))
ax = plt.bar(earthquake_by_location['Location Name'],␣
 ↪earthquake_by_location['Year'])
plt.xticks(rotation=90)
plt.xlabel("Location Name", fontsize=15)
plt.ylabel("Count of Earthquakes", fontsize=15)
plt.title("Earthquake counts with different locations in North America",␣
 ↪fontsize=18)

plt.show()
```



Approach:

- I have made use of the Location Name column and then counted the number of earthquakes for that location using the Year column as it has no missing values.
- Then, I have used a bar chart for displaying the Location Name on the x-axis and the respective counts on the y-axis.

Problems:

- The above visualization is very messy because we have 315 values on the x-axis which makes it very difficult to visualise the content.
- Also, the bars are very thin and they are not clearly distinguishable.

0.3.3 Earthquakes by different regions split for better visualization

[1427]:
```
c = list(earth_quake['Location Name'].unique())
c
```

[1427]:
```
['MEXICO:  MEXICO CITY',
 'HAWAII',
 'MEXICO:  VERACRUZ',
 'MEXICO:  SOUTHERN',
 'MEXICO:  COCULA,JALISCO',
 'MEXICO:  COLIMA',
 'MEXICO:  OAXACA, MIXTECA',
 'MEXICO:  OAXACA',
 'MEXICO:  MEXICO CITY,JALISCO',
 'MEXICO',
 'MEXICO:  ZACATECAS',
 'CANADA:  ST LAWRENCE VALLEY',
 'MEXICO:  PUEBLA',
 'BOSTON AND SALEM, MASSACHUSETTS',
 'MEXICO:  OAXACA,MEXICO CITY',
 'MEXICO:  MEXICO CITY,ACAPULCO',
 'CASCADIA SUBDUCTION ZONE',
 'MEXICO:  MEXICO CITY, COLIMA, GUADALAJARA',
 'MEXICO:  BAJA CALIFORNIA',
 'MEXICO:  MEXICO CITY,OAXACA',
 'S. MEXICO',
 'MEXICO:  COLIMA, GUADALAJARA, MEXICO CITY',
 'MEXICO:  ACAPULCO',
 'MASSACHUSETTS:  EAST OF CAPE ANN',
 'MEXICO:  JORULLO',
 'MEXICO:  JALISCO,ZACATECAS',
 'CANADA:  NEWFOUNDLAND',
 'MEXICO: GUERRERO, MORELOS, OAXACA, MEXICO CITY',
 'MEXICO:  JALISCO',
 'MEXICO:  GUERRERO, ACAPULCO',
 'MEXICO:  SAN MARCOS, OAXACA',
 'ALASKA PENINSULA: UNGA ISLAND',
 'ALASKA PENINSULA',
 'ALASKA:  KODIAK ISLAND',
 'CALIFORNIA:  SANTA BARBARA',
 'MEXICO:  JALISCO,OAXACA,MICHOACAN',
 'MEXICO:  BAJA CALIFORNIA: LORETO',
 'ARKANSAS: NORTHEAST (NEW MADRID EARTHQUAKES)',
 'MISSOURI:  NEW MADRID',
 'CALIFORNIA:   SAN JUAN CAPISTRANO',
 'CALIFORNIA:   PURISIMA',
 'MEXICO:  OAXACA,TAMAZULAPAN',
```

```
'PENNSYLVANIA:  PHILADELPHIA',
'MEXICO:  MORELOS-GUERRERO,MEXICO CITY,GUADALAJARA',
'ALASKA:  EAST ALEUTIAN ISLANDS',
'MEXICO:  GUERRERO, OAXACA, PUEBLA, VERACRUZ',
'LAKE ERIE (GREAT LAKES)',
'CALIFORNIA',
'MEXICO: ACAPULCO',
'MEXICO: MEXICO CITY,  ACAPULCO',
'MEXICO:  GUADALAJARA',
'CALIFORNIA:  SAN FRANCISCO,SAN JOSE,SANTA CLARA',
'CANADA:  MONTREAL',
'MEXICO:  OAXACA,GUERRERO',
'CALIFORNIA:  NORTHERN',
'N. MEXICO',
'CALIFORNIA:  SAN SIMEON',
'MEXICO:  DURANGO',
'MEXICO:  TLAPUJAHUA',
'ALASKA:  GULF OF ALASKA',
'MEXICO:  VERACRUZ,OAXACA',
'CALIFORNIA:  SOUTHERN',
'MEXICO:  GUERRERO,OAXACA',
'CALIFORNIA:  LOS ANGELES',
'CALIFORNIA:  SAN FRANCISCO',
'CALIFORNIA:  SAN JOSE,SAN FRANCISCO',
'CALIFORNIA:  HALF MOON BAY',
'CALIFORNIA:  CONTRA COSTA,ALAMEDA COUNTIES',
'CALIFORNIA:  SAN DIEGO',
'OFF COAST SW AVALON PENINSULA, NEWFOUNDLAND',
'MEXICO:  PUEBLA,VERACRUZ,ACULTZINGO,ACATZINGO',
'CALIFORNIA:  SONOMA COUNTY:  E CENTRAL',
'CALIFORNIA:  FORT HUMBOLDT, EUREKA',
'CALIFORNIA:  SANTA CRUZ',
'MEXICO:  PUEBLA, VERACRUZ',
'ALASKA:  KODIAK ISLAND, AK',
'HAWAII:  SE OF',
'MEXICO:  GUANAJUATO',
'MEXICO:  SAN JOSE DE ITURBIDE,PUEBLA,VERACRUZ',
'CALIFORNIA:  HAYWARD,SAN FRANCISCO',
'MEXICO:  OAXACA,ZANAGUIA,SAN FRANCISCO OZOLOTEPEC',
'CANADA:  QUEBEC: QUEBEC CITY, CHARLEVOIX',
'MEXICO:  MINATITLAN',
'HAWAIIAN ISLANDS',
'CALIFORNIA:  HUMBOLDT COUNTY',
'NEW YORK:  LONG ISLAND',
'CALIFORNIA:  OWENS VALLEY',
'ALASKA:  ALEUTIAN ISLANDS:  FOX ISLANDS',
'NEW HAMPSHIRE: CONCORD',
```

```
'MEXICO:  MORELOS,GUERRERO',
'MEXICO:  SAN CRISTOBAL DE LA BARRANCA',
'ALASKA:  ALEUTIAN ISLANDS',
'CALIFORNIA:  SAN FERNANDO',
'ALASKA:  SOUTHEASTERN',
'MEXICO:  PUEBLA,HUAHUAPAN,HUAMUXTITLAN,XALPATLAHUAC',
'NEW YORK: ROCKAWAY BEACH, NEAR NEW YORK CITY',
'CONNECTICUT: NEW HAVEN',
'MEXICO:  JALISCO:  SAN CRISTOBAL',
'SOUTH CAROLINA:  CHARLESTON',
'MEXICO:  BAVISPE',
'MEXICO:  GUERRERO,MORELOS',
'MEXICO:  GUERRERO',
'MEXICO:  MORELOS',
'WASHINGTON:  PUGET SOUND',
'CALIFORNIA:  VACAVILLE,WINTERS',
'CALIFORNIA:  WINTERS',
'NEW JERSEY: HIGH BRIDGE',
'MISSOURI:  CHARLESTON',
'MEXICO:  OAXACA:  TEHUANTEPEC',
'MEXICO:  TEHUANTEPEC',
'CALIFORNIA:  SONOMA COUNTY',
'CALIFORNIA:  N COAST,MENDOCINO COUNTY',
'ALASKA:  ALEUTIAN ISLANDS:  NEAR ISLANDS',
'MEXICO:  GUERRERO-OAXACA',
'ALASKA:  CAPE YAKATAGA',
'ALASKA:  SE ALASKA',
'MEXICO:  NEAR COAST OF JALISCO',
'ALASKA:  SE. ALASKA',
'MEXICO:  VENUSTIANO CARRANZA,CHIAPAS,CHIS,TABASCO',
'UTAH',
'MEXICO:  OFF COAST OF GUERRERO',
'ALASKA:  SOUTHWEST',
'ALASKA:  RAMPART',
'ALASKA:  ANDREANOF ISLANDS',
'MEXICO:  REVILLA GIGEDO ISLANDS',
'ALASKA:  ALEUTIAN ISLANDS:  RAT ISLANDS',
'ALASKA:  SKAGWAY',
'MEXICO:  GULF OF CALIFORNIA',
'MEXICO:  GUERRERO,MICHOACAN',
'MEXICO:  MICHOACAN',
'ALASKA:  PRINCE WILLIAM SOUND',
'MEXICO:  NEAR COAST OF GUERRERO',
'MEXICO:  ACAMBAY-TIXMADEJE',
'ALASKA:  ALASKA PENINSULA',
'MEXICO:  CENTRAL, ACAMBAY, TIXMADEJE',
'MEXICO:  CHIAPAS',
```

```
'CALIFORNIA:  EL CENTRO',
'NEVADA:  PLEASANT VALLEY',
'MEXICO:  NEAR OAXACA COAST',
'MEXICO:  VERACRUZ: COZAUTLAN, PUEBLA: PATLANALA',
'UTAH:  ELSINORE',
'MONTANA:  CLARKSTON VALLEY',
'CALIFORNIA, MEXICO',
'CALIFORNIA:  S:  OFF COAST',
'BRITISH COLUMBIA',
'MEXICO:  OAXACA,COLIMA,PUEBLA,GUERRERO,MORELOS',
'CANADA: QUEEN CHARLOTTE ISLANDS',
'NEW YORK:  ATTICA',
'CANADA:  GRAND BANKS',
'MEXICO:  CENTRAL, COLIMA',
'MEXICO:  CENTRAL',
'NEW YORK:  WILLETTS POINT',
'NEVADA:  CEDAR MOUNTAIN',
'CALIFORNIA:  LONG BEACH',
'CANADA:  BAFFIN BAY',
'CALIFORNIA:  BAJA,IMPERIAL VALLEY',
'MONTANA:  HELENA',
'MEXICO:  TUXTLA GUTIERREZ,GUERRERO,MEXICO CITY',
'MEXICO:  CENTRAL, PUEBLA:  ESPERANZA, VERACRUZ',
'MEXICO:  CHILPANCINGO,TIXTLA',
'MEXICO:  GUERRERO:  OMETEPEC',
'MEXICO:  OMETEPEC',
'CANADA:  BRITISH COLUMBIA',
'MEXICO:  TELOLOAPAN',
'ALASKA',
'CALIFORNIA; MEXICO',
'MEXICO:  MICHOACAN, COLIMA, JALISCO',
'MEXICO:  JUCHITAN',
'MEXICO:  GUERRERO: PARICUTIN VOLCANO FORMS',
'NEW YORK:  MASSENA',
'ALASKA:  UNIMAK ISLAND',
'MEXICO:  MARIA MADRE ISLAND',
'WASHINGTON',
'CANADA:  QUEEN CHARLOTTE ISLANDS',
'MEXICO:  OAXACA:  MIAHIATLAN',
'CALIFORNIA:  TERMINAL ISLAND',
'CALIFORNIA:  KERN COUNTY',
'NEVADA:  FALLON',
'NEVADA:  STILLWATER RANGE',
'NEVADA:  DIXIE VALLEY',
'MEXICO:  ACAPULCO,MEXICO CITY',
'ALASKA:  LITUYA BAY',
'MEXICO:  OAXACA;',
```

```
'MONTANA:  HEBGEN LAKE',
'MEXICO:  GULF OF CAMPECHE',
'MEXICO:  S',
'WASHINGTON:  SEATTLE',
'MEXICO:  ME\\XICO CITY, OAXACA',
'MEXICO:  OAXACA, GUERRERO',
'MEXICO-GUATEMALA:  S CHIAPAS',
'CALIFORNIA:  SANTA ROSA',
'CANADA:  QUEEN CHARLOTTE ISLANDS, BRITISH COLOMBIA',
'ALASKA:  SITKA, JUNEAU',
'MEXICO:  S, FARIAS, TECOMAN',
'CALIFORNIA:  OXNARD',
'HAWAII: HILO',
'MEXICO:  VERACRUZ, MEXICO CITY',
'IDAHO:  POCATELLO VALLEY',
'CALIFORNIA:  OROVILLE',
'CALIFORNIA:  IMPERIAL VALLEY; MEXICO: MEXICALI',
'CALIFORNIA:  LIVERMORE',
'WASHINGTON:  MT ST HELENS',
'CALIFORNIA:  MAMMOTH LAKES',
'MEXICO:  NW',
'KENTUCKY:  MAYSVILLE',
'MEXICO:  S, HUAJAPAN DE LEON, OAXACA',
'CALIFORNIA:  NORTH COAST',
'CALIFORNIA:  WESTMORLAND,CALIPATRIA',
'MEXICO:  MICHOACAN: LAZARO CARDENAS',
'CANADA; MAINE:  CARIBOU, HAYNESVILLE, PRESQUE ISLE',
'MEXICO:  GUERERRO, ORZIBA, OAXACA, GUADALUPE',
'MEXICO:  OAXACA:   SALINA CRUZ',
'CALIFORNIA:  CENTRAL, COALINGA',
'IDAHO:  BORAH PEAK, CHALLIS, MACKAY',
'HAWAII: KAPAPALA',
'CALIFORNIA:  CENTRAL:  MORGAN HILL',
'WYOMING:  DOUGLAS, MEDICINE BOW',
'MEXICO:  MICHOACAN:  MEXICO CITY',
'MEXICO:  SW COAST:  MEXICO CITY',
'ALASKA:  ALEUTIAN ISLANDS:  ADAK',
'CALIFORNIA:  PALM SPRINGS',
'CALIFORNIA:  SAN DIEGO, NEWPORT BEACH',
'CALIFORNIA-NEVADA:  CHALFANT VALLEY',
'CALIFORNIA:  WHITTIER',
'CALIFORNIA:  WHITTIER, PASADENA',
'CALIFORNIA:  SUPERSTITION HILLS',
'ALASKA:  YAKUTAT',
'ALASKA:  GULF OF ALASKA:  ANCHORAGE',
'CANADA:  QUEBEC:  SAGUENAY, QUEBEC CITY',
'MEXICO:  MEXICO CITY, ACAPULCO',
```

```
'HAWAIIAN ISLANDS:  PUNA DISTRICT',
'CALIFORNIA:  LOMA PRIETA',
'CALIFORNIA:  S,  CLAREMONT,  COVINA',
'CALIFORNIA:  ARCADIA, GLENDALE, LOS ANGELES',
'CALIFORNIA:  HONEYDEW, WHITETHORN, PETROLIA',
'CALIFORNIA:  JOSHUA TREE, ANGELUS OAKS',
'CALIFORNIA: HUMBOLDT COUNTY: FERNDALE, PETROLIA',
'CALIFORNIA: LANDERS, YUCCA  VALLEY',
'CALIFORNIA: BIG BEAR LAKE, BIG BEAR CITY',
'NEVADA-CALIFORNIA BORDER:  NEVADA TEST SITE',
'WASHINGTON-OREGON BORDER',
'OREGON:  KLAMATH FALLS',
'PENNSYLVANIA: READING, FELT TO CANADA',
'CALIFORNIA:  NORTHRIDGE',
'WYOMING:  AFTON',
'CALIFORNIA:  NORTH:  HONEYDEW',
'CALIFORNIA:  EUREKA, SAMOA, ARCATA, BLUE LAKE',
'MEXICO:  GUERRERO, OAXACA, PUEBLA, MEXICO CITY',
'ALASKA:  FAIRBANKS NORTH STAR COUNTY',
'MEXICO:  JALISCO, MANZANILLO, SAN PATRICIO MELAQUE',
'MEXICO:  MICHOACAN, ARTEAGA',
'MEXICO:  OAXACA, SAN AGUSTIN, SAN FRANCISCO',
'MEXICO:  PUEBLA, VERACRUZ, OAXACA, MORELOS, GUERRERO',
'MEXICO:  GUERRERO:  COAHUAYUTLA; MICHOACAN: CUITZEO',
'CALIFORNIA:  LUDLOW, LANDERS, TWENTYNINE PALMS',
'CALIFORNIA:  NAPA',
'WASHINGTON:  OLYMPIA, SEATTLE, TACOMA',
'MEXICO:  VERACRUZ:  SAN ANDRES TUXTLA, TUXTEPEC',
'MEXICO:  MEXICALI, BAJA CALIFORNIA',
'NEW YORK:  CLINTON, ESSEX, AU SABLE FORKS',
'ALASKA:  CANTWELL, DENALI NATL PARK',
'ALASKA:  SLANA, MENTASTA LAKE, FAIRBANKS',
'MEXICO:  VILLA DE ALVAREZ, COLIMA, TECOMAN, JALISCO',
'CALIFORNIA:  BIG BEAR CITY',
'ALABAMA:  FORT PAYNE,GAYLESVILLE,VALLEY HEAD',
'KENTUCKY:  BARDWELL',
'CALIFORNIA:  PASO ROBLES,TEMPLETON,ATASCADERO',
'MEXICO:  GUERRERO, MEXICO CITY',
'CALIFORNIA:  CENTRAL: PARKFIELD, SAN MIGUEL',
'CANADA:  VANCOUVER ISLAND',
'CALIFORNIA:  OFF COAST NORTHERN',
'MONTANA:  DILLON, SILVER STAR, TWIN BRIDGES',
'MEXICO:  GUERRERO, ATOYAC',
'MONTANA:  SHERIDAN',
'CALIFORNIA:  MONTCLAIR',
'UTAH:  HUNTINGTON',
'CALIFORNIA:  SAN JOSE',
```

```
 'NEVADA:  WELLS',
 'ILLINOIS:  WEST SALEM',
 'MEXICO:  MEXICALI',
 'CALIFORNIA:  OCOTILLO',
 'CANADA:  QUEBEC:  VAL-DES-BOIS, GRACEFIELD',
 'MEXICO:  SAN ANDRES HUAXPALTEPEC',
 'OKLAHOMA:  LUTHER',
 'COLORADO:  PAONIA',
 'ALASKA: ALEUTIAN ISLANDS: FOX ISLANDS',
 'COLORADO:  SEGUNDO',
 'VIRGINIA:  LOUISA COUNTY, MARYLAND, WASHINGTON D.C.',
 'OKLAHOMA:  SPARKS',
 'OKLAHOMA:  SPARKS, PRAGUE',
 'MEXICO:  GUERRERO, OAXACA',
 'TEXAS:  WEST',
 'MEXICO:  SAN MARCOS, ACAPULCO',
 'OKLAHOMA:  GUTHRIE',
 'CALIFORNIA:  LA HABRA, BREA, FULLERTON',
 'MEXICO:  GUERRERO; MEXICO CITY',
 'MEXICO:  TECPAN',
 'MEXICO; GUATEMALA:  SAN MARCOS',
 'OKLAHOMA:  HARRAH',
 'CALIFORNIA:  NAPA, VALLEJO',
 'MEXICO:  COCOTITLAN',
 'ALASKA:  KENAI',
 'OKLAHOMA:  CUSHING',
 'ALASKA:  SKAGWAY; CANADA:  BRITISH COLUMBIA',
 'MEXICO: OAXACA, CHIAPAS, TABASCO; GUATEMALA',
 'MEXICO: MEXICO CITY, MORELOS, PUEBLA',
 'OKLAHOMA: BRECKENRIDGE, ENID',
 'HAWAIIAN ISLANDS: PUNA DISTRICT',
 'ALASKA:  ANCHORAGE',
 'MEXICO:  CHIAPAS; GUATEMALA:  SAN MARCOS',
 'CALIFORNIA:  RIDGECREST;  NEVADA',
 'CALIFORNIA:  RIDGECREST',
 'MEXICO: OAXACA',
 'NORTH CAROLINA:  SPARTA',
 'MEXICO: MEXICO CITY, MICHOACAN']
```

[1428]:
```
earth_quake['Location Name'] = earth_quake['Location Name'].
↪replace(['CALIFORNIA, MEXICO', 'CALIFORNIA-NEVADA', 'MEXICO-GUATEMALA'],
↪['CALIFORNIA; MEXICO', 'NEVADA-CALIFORNIA BORDER', 'MEXICO; GUATEMALA'])

earth_quake
```

[1428]:

	Year	Mo	Dy	Hr	Mn	Sec	Country	\
1	1475	<NA>	<NA>	<NA>	<NA>	NaN	MEXICO	
2	1500	<NA>	<NA>	<NA>	<NA>	NaN	USA	
3	1523	<NA>	<NA>	<NA>	<NA>	NaN	MEXICO	
4	1537	<NA>	<NA>	<NA>	<NA>	NaN	MEXICO	
5	1538	<NA>	<NA>	<NA>	<NA>	NaN	MEXICO	
..	…	…	…	…	…	…	…	
494	2021	7	29	6	15	47.0	USA	
495	2021	9	8	1	47	47.0	MEXICO	
496	2022	5	25	21	43	2.0	MEXICO	
497	2022	9	19	18	5	6.0	MEXICO	
498	2022	9	22	6	16	9.0	MEXICO	

	Location Name	Latitude	Longitude	…	Deaths	\
1	MEXICO: MEXICO CITY	NaN	NaN	…	<NA>	
2	HAWAII	NaN	NaN	…	<NA>	
3	MEXICO: VERACRUZ	19.200	-96.400	…	<NA>	
4	MEXICO: SOUTHERN	NaN	NaN	…	<NA>	
5	MEXICO: MEXICO CITY	19.200	-99.100	…	<NA>	
..	…	…	…	…	…	
494	ALASKA PENINSULA	55.325	-157.841	…	<NA>	
495	MEXICO: GUERRERO	16.982	-99.773	…	3	
496	MEXICO: OAXACA	16.325	-95.856	…	<NA>	
497	MEXICO: MICHOACAN, COLIMA, JALISCO	18.367	-103.252	…	2	
498	MEXICO: MEXICO CITY, MICHOACAN	18.308	-102.923	…	2	

	Injuries	Damage ($Mil)	Houses Destroyed	Houses Damaged	Total Deaths	\
1	<NA>	NaN	<NA>	<NA>	<NA>	
2	<NA>	NaN	<NA>	<NA>	<NA>	
3	<NA>	NaN	<NA>	<NA>	<NA>	
4	<NA>	NaN	<NA>	<NA>	<NA>	
5	<NA>	NaN	<NA>	<NA>	<NA>	
..	…	…	…	…	…	
494	<NA>	NaN	<NA>	<NA>	<NA>	
495	<NA>	200.0	<NA>	7317	3	
496	<NA>	NaN	<NA>	<NA>	<NA>	
497	<NA>	NaN	800	6084	2	
498	3	NaN	<NA>	<NA>	2	

	Total Injuries	Total Damage ($Mil)	Total Houses Destroyed	\
1	<NA>	NaN	<NA>	
2	<NA>	NaN	<NA>	
3	<NA>	NaN	<NA>	
4	<NA>	NaN	<NA>	
5	<NA>	NaN	<NA>	
..	…	…	…	
494	<NA>	NaN	<NA>	

```
495            <NA>                200.0                    <NA>
496            <NA>                  NaN                    <NA>
497            <NA>                  NaN                     800
498               3                  NaN                    <NA>

     Total Houses Damaged
1                    <NA>
2                    <NA>
3                    <NA>
4                    <NA>
5                    <NA>
..                    …
494                  <NA>
495                  7317
496                  <NA>
497                  6084
498                  <NA>

[498 rows x 23 columns]
```

[1429]:
```
# The above plot is a messy one with many columns
# Let's extract the first word from the Location Name and then group according␣
 ↪to it

earth_quake['Place'] = earth_quake.apply(lambda x: x['Location Name'].split(":
 ↪")[0], axis=1)
```

[1430]:
```
earth_quake[['Place']]
```

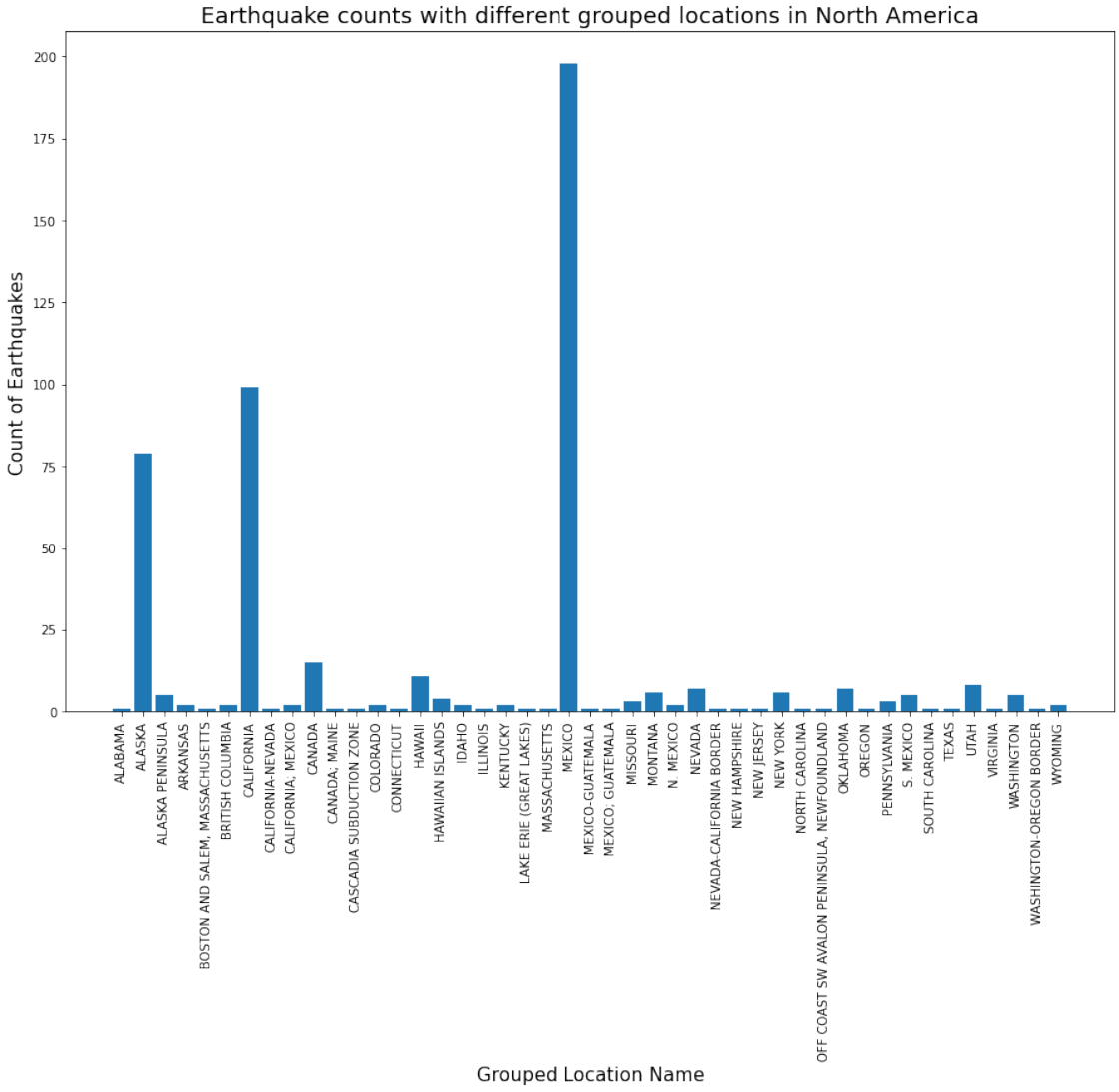
[1430]:
```
                Place
1              MEXICO
2              HAWAII
3              MEXICO
4              MEXICO
5              MEXICO
..                …
494  ALASKA PENINSULA
495            MEXICO
496            MEXICO
497            MEXICO
498            MEXICO

[498 rows x 1 columns]
```

[1431]:
```
x = earth_quake.groupby(['Place'])[['Year']].count()
earthquake_by_place = x.reset_index()
earthquake_by_place
```

[1431]:

	Place	Year
0	ALABAMA	1
1	ALASKA	79
2	ALASKA PENINSULA	5
3	ARKANSAS	2
4	BOSTON AND SALEM, MASSACHUSETTS	1
5	BRITISH COLUMBIA	2
6	CALIFORNIA	99
7	CALIFORNIA-NEVADA	1
8	CALIFORNIA; MEXICO	2
9	CANADA	15
10	CANADA; MAINE	1
11	CASCADIA SUBDUCTION ZONE	1
12	COLORADO	2
13	CONNECTICUT	1
14	HAWAII	11
15	HAWAIIAN ISLANDS	4
16	IDAHO	2
17	ILLINOIS	1
18	KENTUCKY	2
19	LAKE ERIE (GREAT LAKES)	1
20	MASSACHUSETTS	1
21	MEXICO	198
22	MEXICO-GUATEMALA	1
23	MEXICO; GUATEMALA	1
24	MISSOURI	3
25	MONTANA	6
26	N. MEXICO	2
27	NEVADA	7
28	NEVADA-CALIFORNIA BORDER	1
29	NEW HAMPSHIRE	1
30	NEW JERSEY	1
31	NEW YORK	6
32	NORTH CAROLINA	1
33	OFF COAST SW AVALON PENINSULA, NEWFOUNDLAND	1
34	OKLAHOMA	7
35	OREGON	1
36	PENNSYLVANIA	3
37	S. MEXICO	5
38	SOUTH CAROLINA	1
39	TEXAS	1
40	UTAH	8
41	VIRGINIA	1
42	WASHINGTON	5
43	WASHINGTON-OREGON BORDER	1
44	WYOMING	2

[1432]:
```
fig, ax = plt.subplots(figsize = (15,10))
ax = plt.bar(earthquake_by_place['Place'], earthquake_by_place['Year'], )
plt.xticks(rotation=90)
plt.xlabel("Grouped Location Name", fontsize=15)
plt.ylabel("Count of Earthquakes", fontsize=15)
plt.title("Earthquake counts with different grouped locations in North␣
 ↪America", fontsize=18)

plt.show()
```



[1433]:
```
fig = px.bar(earthquake_by_place, 'Place', 'Year', width=1000, height=1200,
             title='Earthquake counts with different grouped locations in North␣
 ↪America',
```

```
            labels=dict(Place="Grouped Location Name", Year="Count of␣
 ↪Earthquakes"))

fig.show('notebook')
```

Approach:

- In this visualization, firstly we have obtained the data by performing certain manipulations.
- I have made use of the Location Name column and then observed that it has a ':' in it which can be used to separate the values based on the first value.
- There are a few values which are same but in different format. So, I have changed them in the original dataset.
- This will give us a reduced set of values by using which we can group the count of earthquakes.
- I have then counted the number of earthquakes for the new location using the Year column as it has no missing values and grouped according to the new location.
- Then, I have used a bar chart for displaying the Location Name on the x-axis and the respective counts on the y-axis.

Pros:

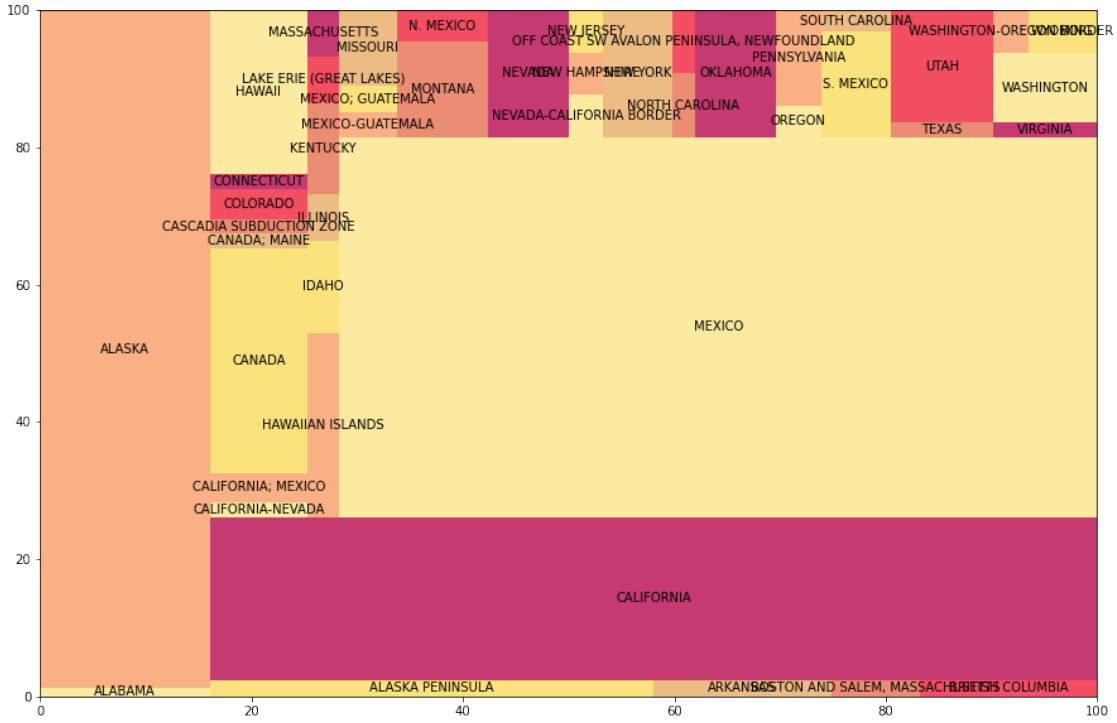
- This visualization is better than the previus one, because here we are able to make out the proportion of earthquakes by the split regions.
- The visualization is legible to the eyes and it is easy to distinguish between the bars.

[978]:
```
!pip install squarify
```

```
Requirement already satisfied: squarify in /opt/conda/lib/python3.8/site-
packages (0.4.3)
```

[979]:
```
import squarify
```

[1434]:
```
fig, ax = plt.subplots(figsize=(15,10))
colors=['#fae588','#f79d65','#f9dc5c','#e8ac65','#e76f51','#ef233c','#b7094c']
squarify.plot(earthquake_by_place['Year'], label=earthquake_by_place['Place'],␣
 ↪alpha=0.8, color=colors)

plt.show()
```



Approach:

- Another way to visualise the same data, is to use a Treemap for it.
- It gives us the proportion of the total volcanoes that has occurred in the region out of the total.
- In terms of working, it is similar to a pie chart and here we can see that the blocks with the maximum area are the ones which have the highest proportion of volcanoes among them.

Problems:

- Here, the visualization is not aesthetically appealing because we are not able to make out the different squares and its content.
- Treemaps are basically used when we want to visualize data which is in hierarchical format and in our case there is no hierarchy.
- So this visualization does not fit.

0.3.4 Variation of Earthquake count over the years

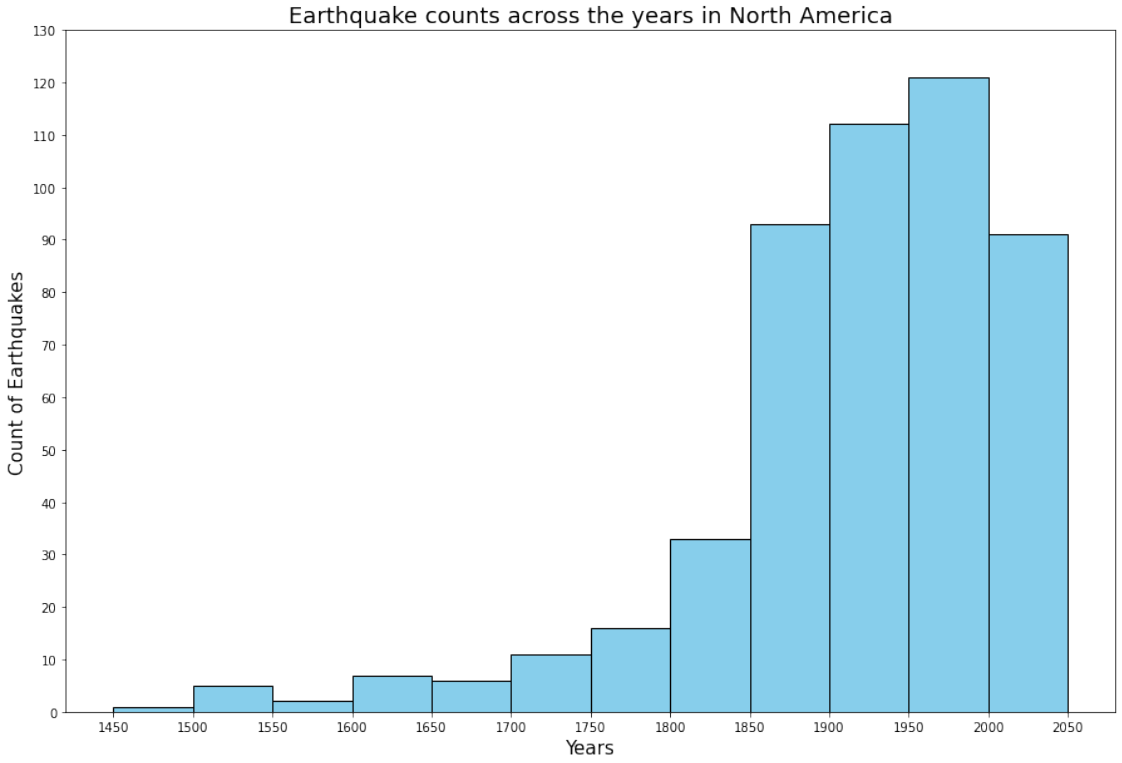
[981]:
```
min(earth_quake['Year'])
```

[981]: 1475

[982]:
```
max(earth_quake['Year'])
```

[982]: 2022

[1435]:
```
fig, ax = plt.subplots(figsize = (15,10))
ax = plt.hist(earth_quake['Year'], bins=[i for i in range(1450, 2100, 50)],
              edgecolor='black', color='skyblue')
plt.xlabel("Years", fontsize=15)
plt.xticks([i for i in range(1450, 2100, 50)])
plt.yticks([i for i in range(0, 140, 10)])
plt.ylabel("Count of Earthquakes", fontsize=15)
plt.title("Earthquake counts across the years in North America", fontsize=18)

plt.show()
```



Approach:

- In this plot, I have made use of a histogram to show a variation of the count of earthquakes across North America over the years ranging from 1430 till date.
- We can see that during the early ages, the number of eathquakes was very low ~ 1-2 earthquakes in gap of 50 years.
- But as time progressed, there has been a exponential increase in the number of earthquakes.
- Just as the Industrial revolution begain in 1760 ther has been a drastic increase.

Concerns:

- Finally, the number of earthquakes has reduced after 2000. But we cannot comment because we have not reached the year 2050.

0.3.5 Earthquakes based on different magnitudes

[1436]:
```
earth_quake['Mag'].unique()
```

[1436]:
```
array([nan, 9. , 7. , 8.3, 7.3, 8. , 7.5, 7.4, 6.9, 7.6, 6.5, 5.2, 6. ,
       5.5, 5.9, 6.3, 7.9, 6.8, 7.8, 6.7, 4.3, 6.2, 8.4, 8.2, 6.4, 7.7,
       8.1, 4.7, 7.2, 5.8, 7.1, 6.6, 5.6, 8.6, 6.1, 9.2, 8.7, 4.8, 5.7,
       5.1, 4.5, 3.5, 5.4, 4.6, 5. , 4.2, 5.3, 4. , 1.6, 3.7, 3.1, 2.1,
       4.1, 3.9])
```

[1437]:
```
earth_quake['Mag'] = earth_quake['Mag'].fillna(0.0)
earth_quake['Mag']
```

[1437]:
```
1      0.0
2      0.0
3      0.0
4      0.0
5      0.0
      ...
494    8.2
495    7.0
496    5.5
497    7.6
498    6.8
Name: Mag, Length: 498, dtype: float64
```

[1438]:
```
sum(earth_quake['Mag'].isna())
```

[1438]:
```
0
```

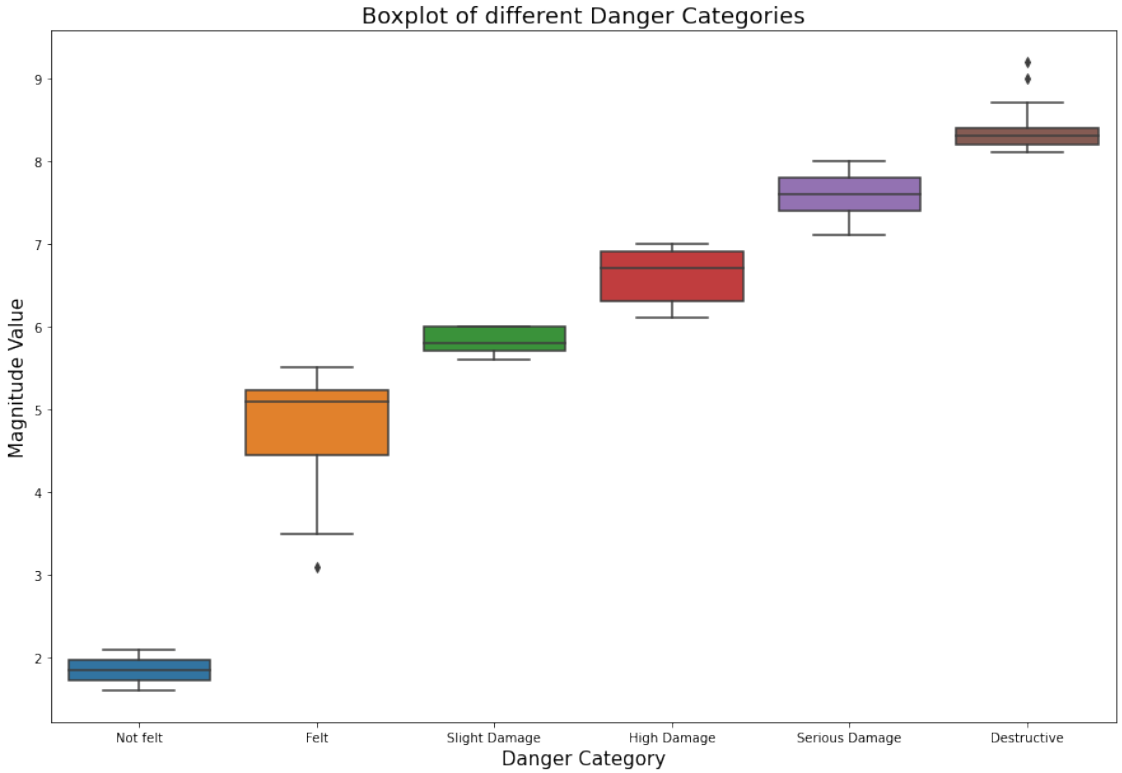
[1439]:
```
earth_quake['Danger Category'] = pd.cut(earth_quake['Mag'],
                     bins=[0.0, 2.5, 5.5, 6.0, 7.0, 8.0, float('Inf')],
                     labels=['Not felt','Felt','Slight Damage','High␣
 ↪Damage','Serious Damage','Destructive'])
```

[1440]:
```
earth_quake['Danger Category'].unique()
```

[1440]:
```
[NaN, 'Destructive', 'High Damage', 'Serious Damage', 'Felt', 'Slight Damage',
'Not felt']
Categories (6, object): ['Not felt' < 'Felt' < 'Slight Damage' < 'High Damage' <
'Serious Damage' < 'Destructive']
```

[1441]:
```
fig, ax = plt.subplots(figsize = (15,10))
ax = sns.boxplot(x = 'Danger Category', y='Mag', data=earth_quake)
plt.xlabel("Danger Category", fontsize=15)
plt.ylabel("Magnitude Value", fontsize=15)
plt.title("Boxplot of different Danger Categories", fontsize=18)
```

```
plt.show()
```



Approach:

- For this particular visualization, I have made use of the Earthquake Magnitude column for determining the type of danger category it belongs to.
- I have used the range of values for identifying the type of damage based on the details in the link: https://www.mtu.edu/geo/community/seismology/learn/earthquake-measure/magnitude/
- I have converted the numeric column to a categorical column using the cut function into 6 different categories of dangers based on the numbers in the link and gave them a suitable naming.
- Then, I have used boxplots to visualize each of the 6 categories to check for outliers and median values.
- We have 1 outlier for the 'Felt' category and 2 for the 'Destructive'.

[1182]:
```
!pip install geoplot
```

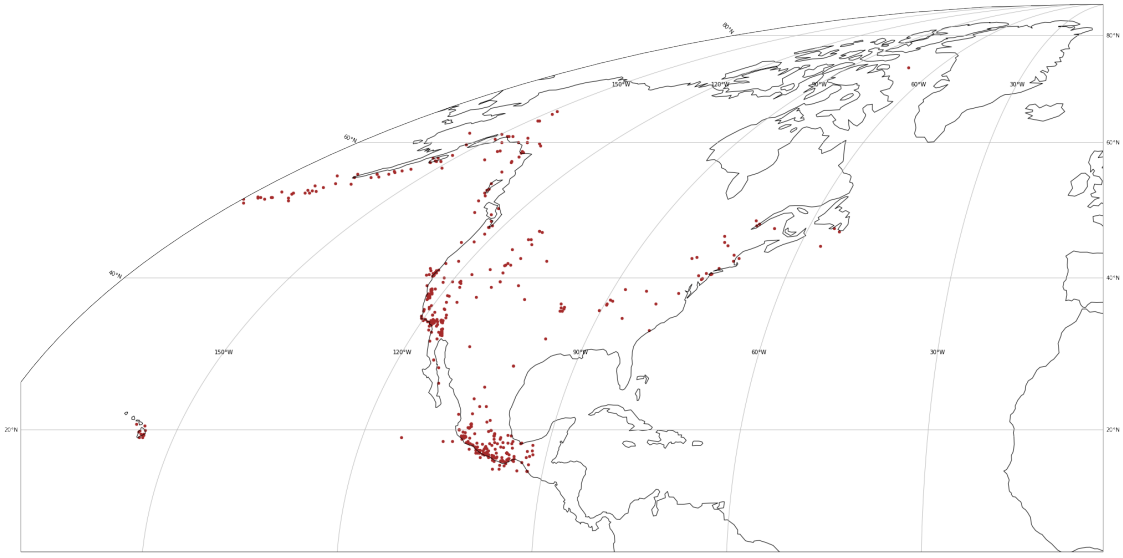
```
Requirement already satisfied: geoplot in /opt/conda/lib/python3.8/site-packages
(0.5.1)
Requirement already satisfied: seaborn in /opt/conda/lib/python3.8/site-packages
(from geoplot) (0.11.0)
```

```
Requirement already satisfied: pandas in /opt/conda/lib/python3.8/site-packages
(from geoplot) (1.1.2)
Requirement already satisfied: geopandas>=0.9.0 in
/opt/conda/lib/python3.8/site-packages (from geoplot) (0.12.1)
Requirement already satisfied: mapclassify>=2.1 in
/opt/conda/lib/python3.8/site-packages (from geoplot) (2.4.3)
Requirement already satisfied: contextily>=1.0.0 in
/opt/conda/lib/python3.8/site-packages (from geoplot) (1.2.0)
Requirement already satisfied: matplotlib>=3.1.2 in
/opt/conda/lib/python3.8/site-packages (from geoplot) (3.3.2)
Requirement already satisfied: cartopy in /opt/conda/lib/python3.8/site-packages
(from geoplot) (0.18.0)
Requirement already satisfied: scipy>=1.0 in /opt/conda/lib/python3.8/site-
packages (from seaborn->geoplot) (1.5.2)
Requirement already satisfied: numpy>=1.15 in /opt/conda/lib/python3.8/site-
packages (from seaborn->geoplot) (1.19.1)
Requirement already satisfied: pytz>=2017.2 in /opt/conda/lib/python3.8/site-
packages (from pandas->geoplot) (2020.1)
Requirement already satisfied: python-dateutil>=2.7.3 in
/opt/conda/lib/python3.8/site-packages (from pandas->geoplot) (2.8.1)
Requirement already satisfied: shapely>=1.7 in /opt/conda/lib/python3.8/site-
packages (from geopandas>=0.9.0->geoplot) (1.7.1)
Requirement already satisfied: packaging in /opt/conda/lib/python3.8/site-
packages (from geopandas>=0.9.0->geoplot) (20.4)
Requirement already satisfied: pyproj>=2.6.1.post1 in
/opt/conda/lib/python3.8/site-packages (from geopandas>=0.9.0->geoplot)
(2.6.1.post1)
Requirement already satisfied: fiona>=1.8 in /opt/conda/lib/python3.8/site-
packages (from geopandas>=0.9.0->geoplot) (1.8.18)
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.8/site-
packages (from mapclassify>=2.1->geoplot) (0.23.2)
Requirement already satisfied: networkx in /opt/conda/lib/python3.8/site-
packages (from mapclassify>=2.1->geoplot) (2.5)
Requirement already satisfied: xyzservices in /opt/conda/lib/python3.8/site-
packages (from contextily>=1.0.0->geoplot) (2022.9.0)
Requirement already satisfied: mercantile in /opt/conda/lib/python3.8/site-
packages (from contextily>=1.0.0->geoplot) (1.2.1)
Requirement already satisfied: pillow in /opt/conda/lib/python3.8/site-packages
(from contextily>=1.0.0->geoplot) (7.2.0)
Requirement already satisfied: requests in /opt/conda/lib/python3.8/site-
packages (from contextily>=1.0.0->geoplot) (2.28.1)
Requirement already satisfied: rasterio in /opt/conda/lib/python3.8/site-
packages (from contextily>=1.0.0->geoplot) (1.2.1)
Requirement already satisfied: joblib in /opt/conda/lib/python3.8/site-packages
(from contextily>=1.0.0->geoplot) (0.17.0)
Requirement already satisfied: geopy in /opt/conda/lib/python3.8/site-packages
(from contextily>=1.0.0->geoplot) (2.3.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
```

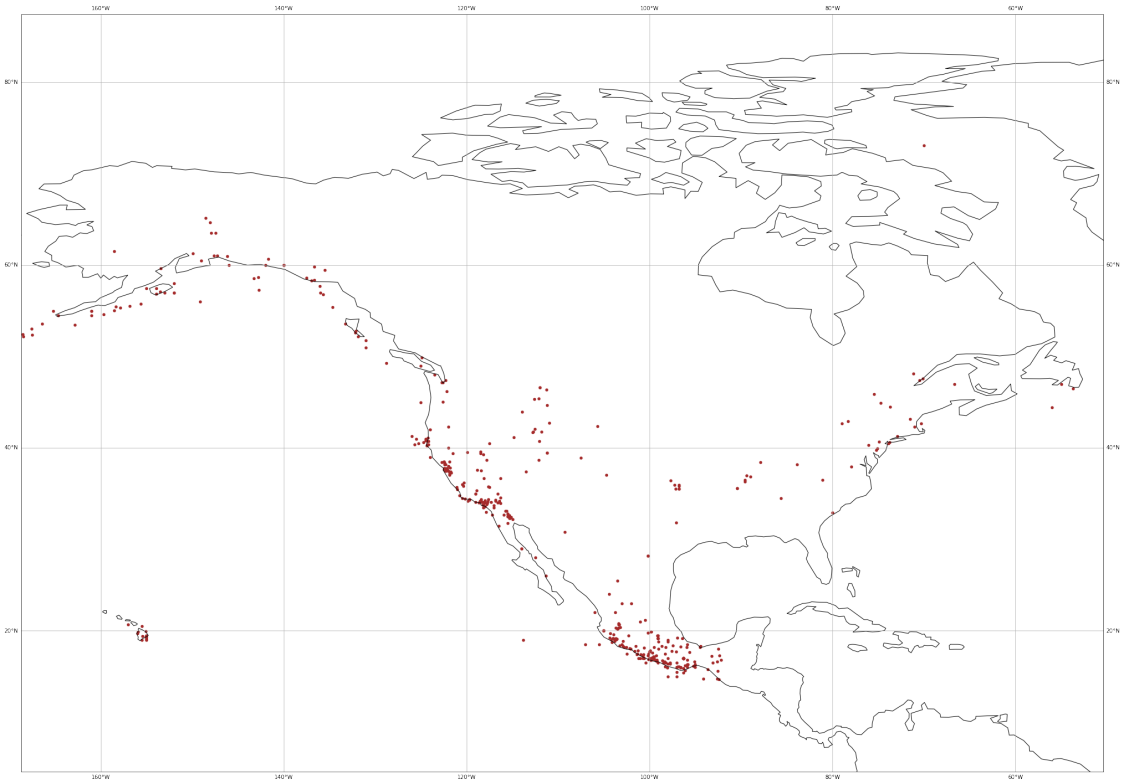
```
/opt/conda/lib/python3.8/site-packages (from matplotlib>=3.1.2->geoplot) (1.2.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in
/opt/conda/lib/python3.8/site-packages (from matplotlib>=3.1.2->geoplot) (2.4.7)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.8/site-
packages (from matplotlib>=3.1.2->geoplot) (0.10.0)
Requirement already satisfied: certifi>=2020.06.20 in
/opt/conda/lib/python3.8/site-packages (from matplotlib>=3.1.2->geoplot)
(2022.9.24)
Requirement already satisfied: setuptools>=0.7.2 in
/opt/conda/lib/python3.8/site-packages (from cartopy->geoplot)
(49.6.0.post20200917)
Requirement already satisfied: pyshp>=1.1.4 in /opt/conda/lib/python3.8/site-
packages (from cartopy->geoplot) (2.3.1)
Requirement already satisfied: six>=1.3.0 in /opt/conda/lib/python3.8/site-
packages (from cartopy->geoplot) (1.15.0)
Requirement already satisfied: attrs>=17 in /opt/conda/lib/python3.8/site-
packages (from fiona>=1.8->geopandas>=0.9.0->geoplot) (20.2.0)
Requirement already satisfied: click<8,>=4.0 in /opt/conda/lib/python3.8/site-
packages (from fiona>=1.8->geopandas>=0.9.0->geoplot) (7.1.2)
Requirement already satisfied: cligj>=0.5 in /opt/conda/lib/python3.8/site-
packages (from fiona>=1.8->geopandas>=0.9.0->geoplot) (0.7.2)
Requirement already satisfied: click-plugins>=1.0 in
/opt/conda/lib/python3.8/site-packages (from
fiona>=1.8->geopandas>=0.9.0->geoplot) (1.1.1)
Requirement already satisfied: munch in /opt/conda/lib/python3.8/site-packages
(from fiona>=1.8->geopandas>=0.9.0->geoplot) (2.5.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/conda/lib/python3.8/site-packages (from scikit-
learn->mapclassify>=2.1->geoplot) (2.1.0)
Requirement already satisfied: decorator>=4.3.0 in
/opt/conda/lib/python3.8/site-packages (from
networkx->mapclassify>=2.1->geoplot) (4.4.2)
Requirement already satisfied: charset-normalizer<3,>=2 in
/opt/conda/lib/python3.8/site-packages (from
requests->contextily>=1.0.0->geoplot) (2.1.1)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.8/site-
packages (from requests->contextily>=1.0.0->geoplot) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/opt/conda/lib/python3.8/site-packages (from
requests->contextily>=1.0.0->geoplot) (1.25.10)
Requirement already satisfied: snuggs>=1.4.1 in /opt/conda/lib/python3.8/site-
packages (from rasterio->contextily>=1.0.0->geoplot) (1.4.7)
Requirement already satisfied: affine in /opt/conda/lib/python3.8/site-packages
(from rasterio->contextily>=1.0.0->geoplot) (2.3.1)
Requirement already satisfied: geographiclib<3,>=1.52 in
/opt/conda/lib/python3.8/site-packages (from geopy->contextily>=1.0.0->geoplot)
(1.52)
```

0.3.6 Plotting the different locations of earthquakes on the map of North America

[1442]:
```
fig, ax = plt.subplots(figsize=(35,35))
ax = plt.axes(projection=crs.Mollweide())
ax.coastlines()
ax.set_extent([-168.739, -10.395, 4.583, 85.633])
ax.gridlines(draw_labels=True)

plt.scatter(x=earth_quake.Longitude, y=earth_quake.Latitude,
            color="brown", s=20, transform=crs.PlateCarree())


plt.show()
```



[1443]:
```
fig, ax = plt.subplots(figsize=(35,35))
ax = plt.axes(projection=crs.PlateCarree())
ax.coastlines()
ax.set_extent([-168.739, -50.395, 4.583, 85.633])
ax.gridlines(draw_labels=True)

plt.scatter(x=earth_quake.Longitude, y=earth_quake.Latitude,
            color="brown", s=20, transform=crs.PlateCarree())


plt.show()
```



Approach:

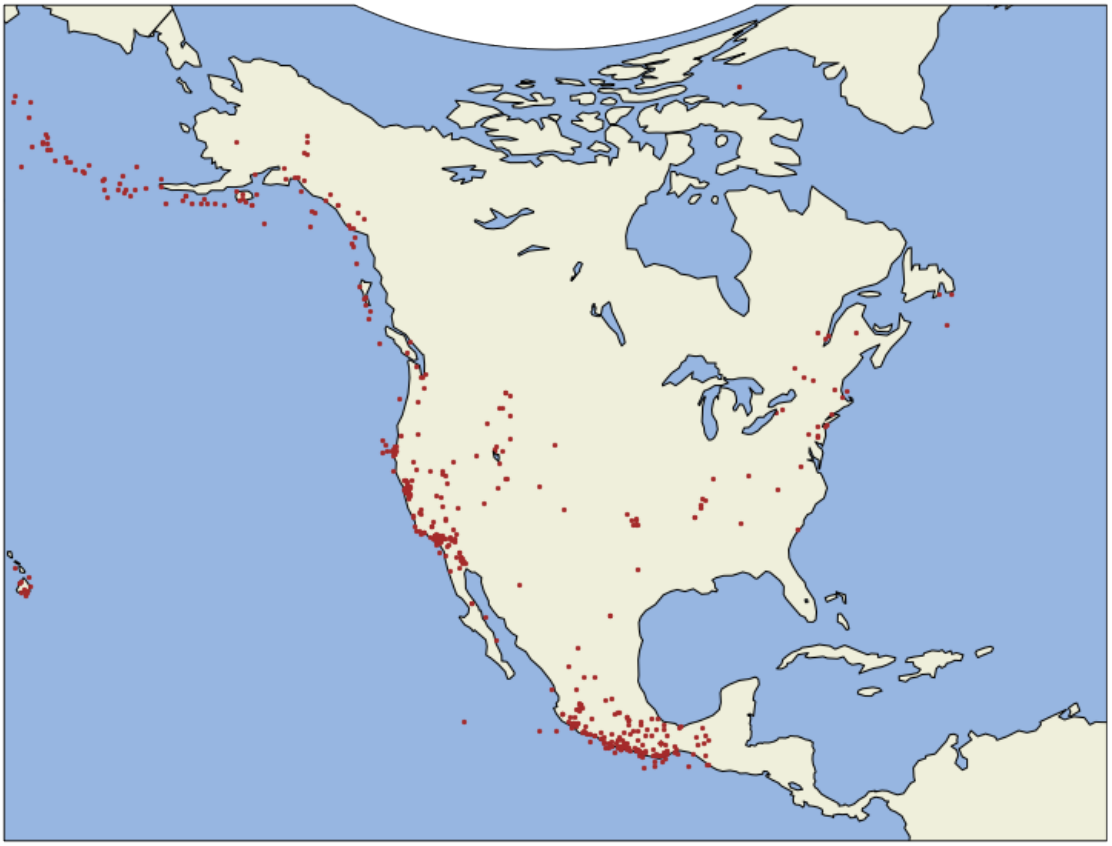
- Since the data contains the Latitude and Longitude columns, I have made use of the cartopy for plotting the data on a map.
- I have provided the extreme geographical coordinates of the North America continent to visualize North America only.
- I have used the PlateCarree projection which is a cylindrical projection which contains of parallel lines making perfect squares.
- I have then made use of the Latitute and Longitude variables from the data to plot the epicenters of the locations where earthquake had occurred.

Problems:

- I tried plotting the same data using the Mollweide projection.
- Mollweide projection is an equal-area pseudocylindrical map projection which displays the area in a form of an ellipse.
- When I provided the coordinates for North America, it gave me a ellipse projection and I plotted the points on it using the PlateCaree projection.
- The ellipse projection makes the geography of North America distorted and difficult to visualize points close to the poles.

0.3.7 Plotting the different locations of earthquakes on the map of North America

[1444]:
```
extent = [-150.74, -60.395, 5.583, 88.633]
central_lon = np.mean(extent[:2])
central_lat = np.mean(extent[2:])

plt.figure(figsize=(15, 10))
ax = plt.axes(projection=crs.AlbersEqualArea(central_lon, central_lat))
ax.set_extent(extent)

plt.scatter(x=earth_quake.Longitude, y=earth_quake.Latitude,
            color="brown", s=5, transform=crs.PlateCarree())

ax.add_feature(cartopy.feature.OCEAN)
ax.add_feature(cartopy.feature.LAND, edgecolor='black')
ax.add_feature(cartopy.feature.LAKES, edgecolor='black')

plt.show()
```



Approach:

- Since the data contains the Latitude and Longitude columns, I have made use of the cartopy for plotting the data on a map.
- I have provided the extreme geographical coordinates of the North America continent to visualize North America only.
- I have used the AlbersEqualArea projection which is an equal area conic projection which contains of 2 standard parallel lines.
- I have added some geographical features including the lakes, surrounding oceans and landmass to make it aesthetically appealing.
- I have then made use of the Latitute and Longitude variables from the data to plot the epicenters of the locations where earthquake had occurred.

[1187]:
```
!pip install plotly
!pip install streamlit
```

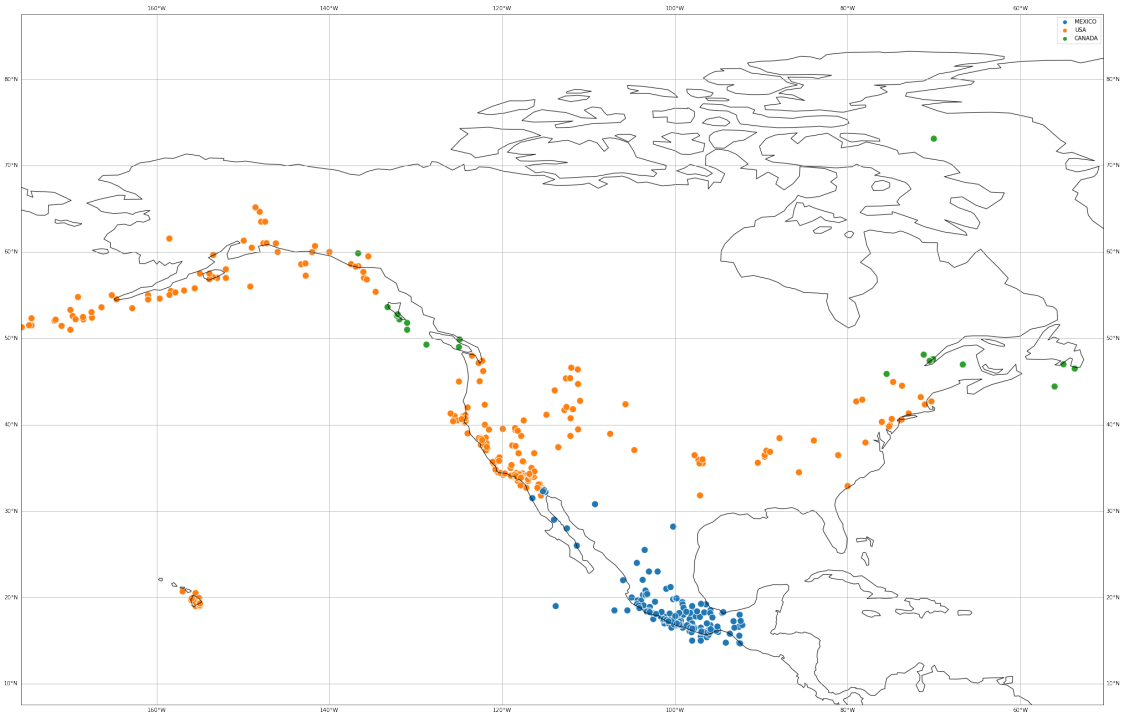
```
Requirement already satisfied: plotly in /opt/conda/lib/python3.8/site-packages
(5.11.0)
Requirement already satisfied: tenacity>=6.2.0 in /opt/conda/lib/python3.8/site-
packages (from plotly) (8.1.0)
Requirement already satisfied: streamlit in /opt/conda/lib/python3.8/site-
packages (1.15.1)
Requirement already satisfied: typing-extensions>=3.10.0.0 in
/opt/conda/lib/python3.8/site-packages (from streamlit) (4.4.0)
Requirement already satisfied: rich>=10.11.0 in /opt/conda/lib/python3.8/site-
packages (from streamlit) (12.6.0)
Requirement already satisfied: numpy in /opt/conda/lib/python3.8/site-packages
(from streamlit) (1.19.1)
Requirement already satisfied: cachetools>=4.0 in /opt/conda/lib/python3.8/site-
packages (from streamlit) (5.2.0)
Requirement already satisfied: click>=7.0 in /opt/conda/lib/python3.8/site-
packages (from streamlit) (7.1.2)
Requirement already satisfied: tornado>=5.0 in /opt/conda/lib/python3.8/site-
packages (from streamlit) (6.0.4)
Requirement already satisfied: gitpython!=3.1.19 in
/opt/conda/lib/python3.8/site-packages (from streamlit) (3.1.29)
Requirement already satisfied: toml in /opt/conda/lib/python3.8/site-packages
(from streamlit) (0.10.2)
Requirement already satisfied: watchdog; platform_system != "Darwin" in
/opt/conda/lib/python3.8/site-packages (from streamlit) (2.1.9)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.8/site-
packages (from streamlit) (7.2.0)
Requirement already satisfied: protobuf<4,>=3.12 in
/opt/conda/lib/python3.8/site-packages (from streamlit) (3.12.4)
Requirement already satisfied: validators>=0.2 in /opt/conda/lib/python3.8/site-
packages (from streamlit) (0.20.0)
Requirement already satisfied: python-dateutil in /opt/conda/lib/python3.8/site-
packages (from streamlit) (2.8.1)
Requirement already satisfied: pandas>=0.21.0 in /opt/conda/lib/python3.8/site-
packages (from streamlit) (1.1.2)
```

```
Requirement already satisfied: importlib-metadata>=1.4 in
/opt/conda/lib/python3.8/site-packages (from streamlit) (2.0.0)
Requirement already satisfied: tzlocal>=1.1 in /opt/conda/lib/python3.8/site-
packages (from streamlit) (4.2)
Requirement already satisfied: pydeck>=0.1.dev5 in
/opt/conda/lib/python3.8/site-packages (from streamlit) (0.8.0)
Requirement already satisfied: altair>=3.2.0 in /opt/conda/lib/python3.8/site-
packages (from streamlit) (4.2.0)
Requirement already satisfied: blinker>=1.0.0 in /opt/conda/lib/python3.8/site-
packages (from streamlit) (1.4)
Requirement already satisfied: packaging>=14.1 in /opt/conda/lib/python3.8/site-
packages (from streamlit) (20.4)
Requirement already satisfied: pyarrow>=4.0 in /opt/conda/lib/python3.8/site-
packages (from streamlit) (10.0.1)
Requirement already satisfied: pympler>=0.9 in /opt/conda/lib/python3.8/site-
packages (from streamlit) (1.0.1)
Requirement already satisfied: semver in /opt/conda/lib/python3.8/site-packages
(from streamlit) (2.13.0)
Requirement already satisfied: requests>=2.4 in /opt/conda/lib/python3.8/site-
packages (from streamlit) (2.28.1)
Requirement already satisfied: pygments<3.0.0,>=2.6.0 in
/opt/conda/lib/python3.8/site-packages (from rich>=10.11.0->streamlit) (2.7.1)
Requirement already satisfied: commonmark<0.10.0,>=0.9.0 in
/opt/conda/lib/python3.8/site-packages (from rich>=10.11.0->streamlit) (0.9.1)
Requirement already satisfied: gitdb<5,>=4.0.1 in /opt/conda/lib/python3.8/site-
packages (from gitpython!=3.1.19->streamlit) (4.0.9)
Requirement already satisfied: six>=1.9 in /opt/conda/lib/python3.8/site-
packages (from protobuf<4,>=3.12->streamlit) (1.15.0)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.8/site-
packages (from protobuf<4,>=3.12->streamlit) (49.6.0.post20200917)
Requirement already satisfied: decorator>=3.4.0 in
/opt/conda/lib/python3.8/site-packages (from validators>=0.2->streamlit) (4.4.2)
Requirement already satisfied: pytz>=2017.2 in /opt/conda/lib/python3.8/site-
packages (from pandas>=0.21.0->streamlit) (2020.1)
Requirement already satisfied: zipp>=0.5 in /opt/conda/lib/python3.8/site-
packages (from importlib-metadata>=1.4->streamlit) (3.3.0)
Requirement already satisfied: backports.zoneinfo; python_version < "3.9" in
/opt/conda/lib/python3.8/site-packages (from tzlocal>=1.1->streamlit) (0.2.1)
Requirement already satisfied: pytz-deprecation-shim in
/opt/conda/lib/python3.8/site-packages (from tzlocal>=1.1->streamlit)
(0.1.0.post0)
Requirement already satisfied: jinja2>=2.10.1 in /opt/conda/lib/python3.8/site-
packages (from pydeck>=0.1.dev5->streamlit) (2.11.2)
Requirement already satisfied: jsonschema>=3.0 in /opt/conda/lib/python3.8/site-
packages (from altair>=3.2.0->streamlit) (3.2.0)
Requirement already satisfied: entrypoints in /opt/conda/lib/python3.8/site-
packages (from altair>=3.2.0->streamlit) (0.3)
Requirement already satisfied: toolz in /opt/conda/lib/python3.8/site-packages
```

```
(from altair>=3.2.0->streamlit) (0.11.1)
Requirement already satisfied: pyparsing>=2.0.2 in
/opt/conda/lib/python3.8/site-packages (from packaging>=14.1->streamlit) (2.4.7)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.8/site-packages (from requests>=2.4->streamlit)
(2022.9.24)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.8/site-
packages (from requests>=2.4->streamlit) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/opt/conda/lib/python3.8/site-packages (from requests>=2.4->streamlit) (1.25.10)
Requirement already satisfied: charset-normalizer<3,>=2 in
/opt/conda/lib/python3.8/site-packages (from requests>=2.4->streamlit) (2.1.1)
Requirement already satisfied: smmap<6,>=3.0.1 in /opt/conda/lib/python3.8/site-
packages (from gitdb<5,>=4.0.1->gitpython!=3.1.19->streamlit) (5.0.0)
Requirement already satisfied: tzdata; python_version >= "3.6" in
/opt/conda/lib/python3.8/site-packages (from pytz-deprecation-
shim->tzlocal>=1.1->streamlit) (2022.6)
Requirement already satisfied: MarkupSafe>=0.23 in
/opt/conda/lib/python3.8/site-packages (from
jinja2>=2.10.1->pydeck>=0.1.dev5->streamlit) (1.1.1)
Requirement already satisfied: pyrsistent>=0.14.0 in
/opt/conda/lib/python3.8/site-packages (from
jsonschema>=3.0->altair>=3.2.0->streamlit) (0.17.3)
Requirement already satisfied: attrs>=17.4.0 in /opt/conda/lib/python3.8/site-
packages (from jsonschema>=3.0->altair>=3.2.0->streamlit) (20.2.0)
```

0.3.8 Plotting the locations of earthquakes hotspots on the map of North America based on Country

[1445]:
```
fig, ax = plt.subplots(figsize=(35,35))
ax = plt.axes(projection=crs.PlateCarree())
ax.coastlines()
ax.set_extent([-175.739, -50.395, 7.583, 85.633])
ax.gridlines(draw_labels=True)

sns.scatterplot(x=earth_quake.Longitude, y=earth_quake.Latitude, s=150,
                hue=earth_quake[['Country']].apply(lambda row: f"{row.
 ↪Country}", axis=1),
                transform=crs.PlateCarree(), legend=True)


plt.show()
```



Approach:

- This visualization combines 2 different plots.
- We have superimposed a scatter plot onto a map of North America to display the hotspots of the earthquakes.
- Firstly, I have used the PlateCarree projection to plot the map of North America by passing in the rough coordinates of the continent.
- Onto it, I have plotted the scatter plot of the hotspot locations using their latitude and longitude values.
- To differentiate, the earthquakes based on country, I have passed the country to the hue parameter.

0.3.9 Plotting the MMI Intensity on maps using sizes

[1446]:
```
earth_quake['MMI Int'].unique()
```

[1446]:
```
array([nan, 10.,  4.,  8.,  9.,  7.,  6., 11.,  5., 12.])
```

[1447]:
```
earth_quake['MMI Int'] = earth_quake['MMI Int'].fillna(1)
earth_quake['MMI Int']
```

[1447]:
```
1      1.0
2      1.0
3      1.0
4      1.0
```

```
5      1.0
      …
494    8.0
495    7.0
496    5.0
497    9.0
498    8.0
Name: MMI Int, Length: 498, dtype: float64
```

[1448]:
```
earth_quake['MMI Int'].astype(np.int64)
```

[1448]:
```
1      1
2      1
3      1
4      1
5      1
      ..
494    8
495    7
496    5
497    9
498    8
Name: MMI Int, Length: 498, dtype: int64
```

[1449]:
```
earth_quake['MMI Int'].unique()
```

[1449]:
```
array([ 1., 10.,  4.,  8.,  9.,  7.,  6., 11.,  5., 12.])
```

[1450]:
```
earth_quake[['Location Name', 'Place', 'MMI Int']]
```

[1450]:

	Location Name	Place	MMI Int
1	MEXICO: MEXICO CITY	MEXICO	1.0
2	HAWAII	HAWAII	1.0
3	MEXICO: VERACRUZ	MEXICO	1.0
4	MEXICO: SOUTHERN	MEXICO	1.0
5	MEXICO: MEXICO CITY	MEXICO	1.0
..	…	…	…
494	ALASKA PENINSULA	ALASKA PENINSULA	8.0
495	MEXICO: GUERRERO	MEXICO	7.0
496	MEXICO: OAXACA	MEXICO	5.0
497	MEXICO: MICHOACAN, COLIMA, JALISCO	MEXICO	9.0
498	MEXICO: MEXICO CITY, MICHOACAN	MEXICO	8.0

[498 rows x 3 columns]

[1451]:
```
fig = px.scatter_geo(earth_quake, 'Latitude', 'Longitude', color='MMI Int',␣
↪size='MMI Int',
```

```
                    title='Earthquake hotspots with the MMI Intensity Levels',
                    height=1000, width=1000)
fig.update_layout(geo_scope = 'north america')
fig.show('notebook')
```

Approach:

- I wanted to plot the earthquake hotspots on the map of North America based on the MMI Intensity.
- I used the scatter_geo function of plotly and I have passed the latitude and longitude of the earthquake hotspots as parameters.
- To include the MMI Intensity information, I have passed it to both parameters the size and color.
- Size parameter will create bubbles based on the order of intensity. Intensity values range from 1 to 12 with 1 being the least and 12 being the highest. So the bubbles will be sized accordingly.
- Color parameter will color again according to the MMI Intensity now treating it as a continuous variable from 1 to 12.

Problems:

- In this plot passing the MMI Intensity to both Size and Color parameters treats it as categorical and numerical variable respectively.
- This is not correct as it is a categorical variable only.
- Also, the bubble sizes convey the same information as the color and hence it is redundant.

[1452]:
```
fig = px.scatter_geo(earth_quake, 'Latitude', 'Longitude', size='MMI Int',␣
↪color='Country',
                    title='Earthquake hotspots with the MMI Intensity Levels',
                    height=1000, width=1000)
fig.update_layout(geo_scope = 'north america')
fig.show('notebook')
```

Approach:

- I wanted to plot the earthquake hotspots on the map of North America based on the MMI Intensity.
- I used the scatter_geo function of plotly and I have passed the latitude and longitude of the earthquake hotspots as parameters.
- To include the MMI Intensity information, I have passed it to the size parameter.
- I have passed the Country value to the color parameter to add extra information.
- Size parameter will create bubbles based on the order of intensity. Intensity values range from 1 to 12 with 1 being the least and 12 being the highest. So the bubbles will be sized accordingly.
- Color parameter will color the hotspots based on their Country.

Improvements:

- In this plot passing the MMI Intensity to Size parameter, the information is just represented only once.
- The color parameter of Country includes another dimension of information.

- The bubble sizes only convey details about the MMI Intensity and the color represents the Country.

0.3.10 Average Focal Depth in kms for earthquakes over the years

[1453]:
```
earth_quake['Focal Depth (km)'] = earth_quake['Focal Depth (km)'].fillna(0)
```

[1454]:
```
x = earth_quake.groupby(['Year'])[['Focal Depth (km)']].mean()
focal_depth_by_year = x.reset_index()
focal_depth_by_year
```

[1454]:

	Year	Focal Depth (km)
0	1475	0.000000
1	1500	0.000000
2	1523	0.000000
3	1537	0.000000
4	1538	0.000000
..	…	…
210	2018	19.400000
211	2019	28.333333
212	2020	21.400000
213	2021	26.000000
214	2022	24.333333

[215 rows x 2 columns]

[1455]:
```
fig = px.bar(x=focal_depth_by_year['Year'], y=focal_depth_by_year['Focal Depth␣
↪(km)']*-1,
             title='Average Focal Depth of Earthquakes over the years',
             labels=dict(x="Year", y="Average Focal Depth in km"))
fig.show('notebook')
```

Approach:

- I wanted to plot the Focal Depth but in a unique way.
- I used the simple bar chart concept but this time instead of the traditional bar chart with bars going from bottom to top, I have reversed their direction.
- I have made 0 as the reference level and the bar values are multiplied by -1 to convert them to negative values.
- Since it is the Focal Depth we are plotting, I thought of visualizing it as actual depth.
- I have considered the 0 as the refernce sea level and the depths of the earthquake epicentres are below the earth surface. So, converting them to negative values allows me to plot them just as a normal bar chart but in the opposite direction.
- It gives us the feeling of earthquake points which are below the mean sea level.

Pros:

- I have considered the 0 as the refernce sea level and the depths of the earthquake epicentres are below the earth surface. So, converting them to negative values allows me to plot them

just as a normal bar chart but in the opposite direction.
- It gives us the feeling of earthquake points which are below the mean sea level.

Cons:

- The filtered dataset consists of 215 rows and it is very difficult to visualize them.
- Many of the values for Focal Depth are 0 and hence majority of the graph area is blank with the top right corner populated with the data.
- This makes it difficult to understand the graph.

[1456]:
```
(focal_depth_by_year['Focal Depth (km)']==0).sum()
```

[1456]:
```
120
```

[1457]:
```
focal_depth_by_year_new = focal_depth_by_year[focal_depth_by_year['Focal Depth␣
↪(km)'] != 0]
focal_depth_by_year_new
```

[1457]:

	Year	Focal Depth (km)
96	1887	26.666667
104	1899	34.000000
105	1900	17.500000
107	1902	26.666667
108	1903	11.000000
..	…	…
210	2018	19.400000
211	2019	28.333333
212	2020	21.400000
213	2021	26.000000
214	2022	24.333333

[95 rows x 2 columns]

[1458]:
```
fig = px.bar(x=focal_depth_by_year_new['Year'],␣
↪y=focal_depth_by_year_new['Focal Depth (km)']*-1,
             title='Average Focal Depth of Earthquakes over the years',
             labels=dict(x="Year", y="Average Focal Depth in km"))
fig.show('notebook')
```

Approach:

- I wanted to plot the Focal Depth but in a unique way.
- I used the simple bar chart concept but this time instead of the traditional bar chart with bars going from bottom to top, I have reversed their direction.
- I have made 0 as the reference level and the bar values are multiplied by -1 to convert them to negative values.
- Since it is the Focal Depth we are plotting, I thought of visualizing it as actual depth.
- I have considered the 0 as the refernce sea level and the depths of the earthquake epicentres are below the earth surface. So, converting them to negative values allows me to plot them

just as a normal bar chart but in the opposite direction.
- It gives us the feeling of earthquake points which are below the mean sea level.

Pros:

- I have considered the 0 as the refernce sea level and the depths of the earthquake epicentres are below the earth surface. So, converting them to negative values allows me to plot them just as a normal bar chart but in the opposite direction.
- It gives us the feeling of earthquake points which are below the mean sea level.

Improvements:

- In this case, I have created a subset wherein I have removed the rows where the Focal Depth is 0 km.
- This significantly reduces the number of records in our dataset to 95 from 215.
- The above visualization is now legible and clearly the bars are distinguishable.

[1459]:
```
earth_quake[['Deaths']].isna().sum()
```

[1459]:
```
Deaths    394
dtype: int64
```

[1460]:
```
earth_quake[['Total Deaths']].isna().sum()
```

[1460]:
```
Total Deaths    386
dtype: int64
```

0.3.11 Plotting interactive correlations of all numeric columns

[1461]:
```
def plot_correlations(x, y):
    fig = px.scatter(earth_quake, earth_quake[x], earth_quake[y],
                         title="Plot of {0} vs {1}".format(y,x),
                         height=1000, width=1000)
    fig.show('notebook')

widgets.interact(plot_correlations, x=list(earth_quake.columns),␣
 ↪y=list(earth_quake.columns));
```

```
interactive(children=(Dropdown(description='x', options=('Year', 'Mo', 'Dy',␣
 ↪'Hr', 'Mn', 'Sec', 'Country', 'Lo…
```

Approach:

- I wanted to plot the correlations between the numeric columns of the data.
- I have created 2 drop down lists for choosing the variables on the x and y axis.
- Basied on the parameters, the scatter plot is visualized.

Cons:

- It is a raw approach and it cannot be used to analyse the correlations between the columns.
- Some of the columns can be analysed using this and the correlations can be understood.
- But some columns like Year, Hr, Sec vs other columns do not make sense.

0.3.12 Interactive Map to show hotspots based on the Danger category

[1462]:
```
danger_category = list(earth_quake['Danger Category'].unique())
danger_category
```

[1462]:
```
[nan,
 'Destructive',
 'High Damage',
 'Serious Damage',
 'Felt',
 'Slight Damage',
 'Not felt']
```

[1463]:
```
earth_quake['Danger Category'] == 'High Damage'
```

[1463]:
```
1      False
2      False
3      False
4      False
5      False
       …
494    False
495     True
496    False
497    False
498     True
Name: Danger Category, Length: 498, dtype: bool
```

[1467]:
```
def plot_by_danger_category(Danger):
    temp = earth_quake.loc[earth_quake['Danger Category'] == Danger]
    fig = px.scatter_geo(temp, 'Latitude', 'Longitude',
                         title='Earthquake hotspots based on Danger Category',
                         height=1000, width=1000)
    fig.update_layout(geo_scope = 'north america')
    fig.show('notebook')

widgets.interact(plot_by_danger_category, Danger=['Destructive', 'Serious␣
 ↪Damage', 'High Damage',
                                                   'Slight Damage',␣
 ↪'Felt', 'Not Felt']);
```

```
interactive(children=(Dropdown(description='Danger', options=('Destructive',␣
 ↪'Serious Damage', 'High Damage', …
```

[1467]:
```
<IPython.core.display.HTML object>
```

Approach:

- I wanted to plot the hotspots based on the Danger Category.
- I used the map of North America as the base and imposed a scatter plot on it which marks the hotspots.
- For filtering, I had initially created a categorical column of Danger based on the Magnitude and have used it to filter.
- I have created a dropdown list which proviides the option of all the danger categories.
- I have defined a plotting function in which the Danger Category is passed as a parameter.
- I subset the dataset rows based on the paramter and use this temporary dataset to plot the points on the map.

Pros:

- It add an element of interactivity wherin the user can choose the parameter value and see only those values.
- It improves the interpretability as there will be fewer data points due to filtering facility.

Improvements:

- I will try to add another filter which gives the option of choosing the country as well.

[1465]:
```
def plot_by_danger_category_country(Danger, Country):
    temp = earth_quake.loc[(earth_quake['Danger Category'] == Danger) &␣
 ↪(earth_quake['Country'] == Country)]
    fig = px.scatter_geo(temp, 'Latitude', 'Longitude',
                         title='Earthquake hotspots based on Danger Category␣
 ↪and Country',
                         height=1000, width=1000)
    fig.update_layout(geo_scope = 'north america')
    fig.show('notebook')

widgets.interact(plot_by_danger_category_country,
                 Danger = ['Destructive', 'Serious Damage', 'High Damage',␣
 ↪'Slight Damage', 'Felt', 'Not Felt'],
                 Country = ['USA', 'MEXICO', 'CANADA'])
```

```
interactive(children=(Dropdown(description='Danger', options=('Destructive',␣
 ↪'Serious Damage', 'High Damage', …
```

[1465]:
```
<function __main__.plot_by_danger_category_country(Danger, Country)>
```

Approach:

- I wanted to plot the hotspots based on the Danger Category.
- I used the map of North America as the base and imposed a scatter plot on it which marks the hotspots.
- For filtering, I had initially created a categorical column of Danger based on the Magnitude and have used it to filter.
- I have created 2 dropdown lists - one which provides the choice for Danger Category and the second provides the choice for the Country.

- I have defined a plotting function in which the Danger Category and Country are passed as parameters.
- I subset the dataset rows based on the paramter and use this temporary dataset to plot the points on the map.

Pros:

- It add an element of interactivity wherin the user can choose the parameter value and see only those values.
- It improves the interpretability as there will be fewer data points due to filtering facility.

References:

https://towardsdatascience.com/an-introduction-to-geographical-data-visualization-3486959cd4b8

https://pro.arcgis.com/en/pro-app/latest/help/mapping/properties/plate-carree.htm

https://pro.arcgis.com/en/pro-app/latest/help/mapping/properties/albers.htm

https://www.statology.org/matplotlib-boxplot-by-group/

https://coderzcolumn.com/tutorials/python/interactive-widgets-in-jupyter-notebook-using-ipywidgets

https://plotly.com/python/scatter-plots-on-maps/