Exercise 5: Exploring Data Types

Transform the data types in our example DataFrame to the correct types with the pandas astype function. Let’s use the same data as in previous excercises.

1. Import the required libraries, as illustrated here.

NOTE: if Activity 2 is done, Excercise 5 can be ommited but remember - you have to have data prepared (valid datatypes)

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt
```

2. Read the data.

```
df=pd.read_csv("RadNet_Laboratory_Analysis.csv")
```

3. Check the current data types using the dtypes function on the DataFrame:

```
df.dtypes
```

4. Create a list of numeric columns.

```
columns=df.columns

id_cols = ['State', 'Location', "Date Posted", 'Date Collected',
'Sample Type', 'Unit']

columns=list(set(columns)-set(id_cols))

columns
```

5. Apply lambda function (or User Defined Function) for all numeric columns to apply NaN value

```
df.loc[:,columns]=df.loc[:,columns].applymap(lambda x: np.nan if
((x=="Non-detect")|(x=="ND")) else x)

df.loc[:,columns].head()
```

6. Change datatypes to_datetime and to_numeric

```
df['Date Posted'] = pd.to_datetime(df['Date Posted'])
df['Date Collected'] = pd.to_datetime(df['Date Collected'])
for col in columns:
    df[col] = pd.to_numeric(df[col])
df.dtypes
```

7. Use the astype method to transform the columns that are not numeric to the category type:

```
df['State'] = df['State'].astype('category')
df['Location'] = df['Location'].astype('category')
df['Unit'] = df['Unit'].astype('category')
df['Sample Type'] = df['Sample Type'].astype('category')
df.dtypes
```

Excercise 6. Aggregation and Grouping

1. Group the DataFrame using the State column.

```
df.groupby('State')
```

2. Select the radionuclide Cs-134 and calculate the average value per group:

```
df.groupby('State')['Cs-134'].head()
```

3. Do the same for all columns, grouping per state and applying directly the mean function:

```
df.groupby('State').mean().head()
```

4. Now, group by more than one column, using a list of grouping columns.

```
df.groupby(['State','Location']).mean
```

5. Aggregate using several aggregation operations per column with the agg method. Use the State and Location columns:

```
df.groupby(['State', 'Location']).agg({'Cs-134':['mean',
'std'],'Te-129':['min', 'max']})
```

Excercise 7. Exporting Data in Different Formats

After applying all the changes to the dataset (change datatypes, change unwanted values), You may want to save your work.

Export our transformed DataFrame, with the right values and columns, to the CSV format with the to_csv function. Exclude the index using index=False, use a semicolon as the separator sep=";", and encode the data as UTF-8 encoding="utf-8":

```
df.to_csv('radiation_clean.csv', index=False, sep=';',
encoding='utf-8')
```

Activity 3. Plotting Data with Pandas

1. Use the RadNet DataFrame that we have been working with

2. Fix all the data type problems, as we saw before.

3. .Create a plot with a filter per Location, selecting the city of San Bernardino, and one radionuclide, with the x-axis set to the date and the y-axis with radionuclide I-131.

4. Create a plot using matplotlib.plt - matlab style - with the concentration of two related radionuclides, I-131 and I-132.

5. Create a scatter plot with the concentration of two related radionuclides, I-131 and I-132:

```
ig, ax = plt.subplots()

ax.scatter(x=df['I-131'], y=df['I-132'])

_ = ax.set(

    xlabel='I-131',

    ylabel='I-132',

    title='Comparison between concentrations of I-131 and I-132'

)
```