EPA Air Data Collection

EPA son las siglas en inglés de "Environmental Protection Agency", que en español significa Agencia de Protección Ambiental. La EPA es una agencia federal del gobierno de los Estados Unidos que se encarga de proteger la salud humana y el medio ambiente mediante la regulación y el control de la contaminación en el aire, agua y suelo. La EPA tiene una amplia gama de responsabilidades, incluyendo el desarrollo y la aplicación de estándares ambientales, la realización de investigaciones científicas y la educación del público sobre cuestiones ambientales.

Información sobre monitores o estaciones de medición

En este etapa utilizamos la información proporcionada en:
https://epa.maps.arcgis.com/apps/webappviewer/index.html?id=5f239fd3e72f424f98ef3d5def547eb5

Hay que seleccionar una capa (layer) en el mapa. Después, seleccionando cualquier punto desplegado(cualquier estación) se abre un menú del que se selecciona la opción de tres puntos '...' en la esquina inferior derecha. Lo que abre otro pequeño menú y del que seleccionamos la opción 'View in attribute table', que desplegará la tabla de atrbutos. En dicha tabla hay que seleccionar 'Filter by map extent'. Una vez cargado el contenido hay que seleccionar 'Options'>'Filter'. En filter selecionar 'add expression' y filatar por estado y por ciudad. Después seleccionar la opción 'export all to csv'.

Repetir el proceso para cada contaminante del que se deseé información general.

En nuestro caso se seleccionó información para los siguientes principales contaminanates: 'SO2', 'PM10', 'Ozone', 'PM2.5 - NAAQS/AQI', 'NO2', 'CO', cuya información se unió en el archivo 'nyc_air_quality_stations.csv'

In [33]:
```
import pandas as pd
```

In [34]:
```
df = pd.read_csv('nyc_air_quality_stations.csv')
```

In [35]:
```
df.columns
```

Out[35]:
```
Index(['AQS_Site_ID', 'POC', 'CBSA', 'Local_Site_Name', 'Address', 'Latitude',
       'Longitude', 'Datum', 'LatLon_Accuracy_meters', 'Elevation_meters_MSL',
       'Parameter_Name', 'Monitor_Start_Date', 'Last_Sample_Date',
       'Measurement_Scale', 'Measurement_Scale_Definition', 'Sample_Duration',
       'Sample_Collection_Frequency', 'Sample_Collection_Method',
       'Method_Reference_ID', 'FRMFEM', 'Monitor_Type', 'Reporting_Agency',
       'Sample_Analysis_Method'],
      dtype='object')
```

In [36]:
```
df = df.sort_values('Parameter_Name', ascending=False)
```

In [37]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32 entries, 0 to 31
Data columns (total 23 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   AQS_Site_ID                    32 non-null     object
 1   POC                            32 non-null     int64
 2   CBSA                           32 non-null     object
 3   Local_Site_Name                32 non-null     object
 4   Address                        32 non-null     object
 5   Latitude                       32 non-null     float64
 6   Longitude                      32 non-null     float64
 7   Datum                          32 non-null     object
 8   LatLon_Accuracy_meters         32 non-null     int64
 9   Elevation_meters_MSL           32 non-null     int64
 10  Parameter_Name                 32 non-null     object
 11  Monitor_Start_Date             32 non-null     object
 12  Last_Sample_Date               32 non-null     object
 13  Measurement_Scale              32 non-null     object
 14  Measurement_Scale_Definition   32 non-null     object
 15  Sample_Duration                32 non-null     object
 16  Sample_Collection_Frequency    13 non-null     object
 17  Sample_Collection_Method       32 non-null     object
 18  Method_Reference_ID            32 non-null     object
 19  FRMFEM                         32 non-null     object
 20  Monitor_Type                   32 non-null     object
 21  Reporting_Agency               32 non-null     object
 22  Sample_Analysis_Method         19 non-null     object
dtypes: float64(2), int64(3), object(18)
memory usage: 6.0+ KB
```

Listamos los nombres de los parámetros de los que tenemos info por estación/monitor

In [38]:
```
df['Parameter_Name'].unique()
```

Out[38]:
```
array(['SO2', 'PM2.5 - NAAQS/AQI', 'PM10', 'Ozone', 'NO2', 'CO'],
      dtype=object)
```

Generamos dataframe y exportamos archivo parquet para parámetro 'SO2'

In [39]:
```
so2_df = (df[df['Parameter_Name'] == 'SO2']).copy()
```

In [40]:
```
so2 = so2_df[['AQS_Site_ID','Local_Site_Name','Latitude',
       'Longitude','Measurement_Scale', 'Measurement_Scale_Definition']]
```

In [41]:
```
so2.to_parquet('so2.parquet', index=False)
```

Generamos dataframe y exportamos archivo parquet para parámetro 'PM10'

In [42]:
```
pm10_df = (df[df['Parameter_Name'] == 'PM10']).copy()
pm10 = pm10_df[['AQS_Site_ID','Local_Site_Name','Latitude',
       'Longitude','Measurement_Scale', 'Measurement_Scale_Definition']]
pm10.to_parquet('pm10.parquet', index=False)
```

Generamos dataframe y exportamos archivo parquet para parámetro 'Ozone'

In [43]:
```
ozone_df = (df[df['Parameter_Name'] == 'Ozone']).copy()
ozone = ozone_df[['AQS_Site_ID','Local_Site_Name','Latitude',
       'Longitude','Measurement_Scale', 'Measurement_Scale_Definition']]
ozone.to_parquet('ozone.parquet', index=False)
```

Generamos dataframe y exportamos archivo parquet para parámetro 'PM2.5 - NAAQS/AQI'

In [44]:
```
pm25_df = (df[df['Parameter_Name'] == 'PM2.5 - NAAQS/AQI']).copy()
pm25 = pm25_df[['AQS_Site_ID','Local_Site_Name','Latitude',
       'Longitude','Measurement_Scale', 'Measurement_Scale_Definition']]
pm25.to_parquet('pm25.parquet', index=False)
```

Generamos dataframe y exportamos archivo parquet para parámetro 'NO2'

In [45]:
```
no2_df = (df[df['Parameter_Name'] == 'NO2']).copy()
no2 = no2_df[['AQS_Site_ID','Local_Site_Name','Latitude',
       'Longitude','Measurement_Scale', 'Measurement_Scale_Definition']]
no2.to_parquet('no2.parquet', index=False)
```

Generamos dataframe y exportamos archivo parquet para parámetro 'CO'

In [48]:
```
co_df = (df[df['Parameter_Name'] == 'CO']).copy()
co = co_df[['AQS_Site_ID','Local_Site_Name','Latitude',
       'Longitude','Measurement_Scale', 'Measurement_Scale_Definition']]
co.to_parquet('co.parquet', index=False)
```

In [49]:
```
co
```

Out[49]:

	AQS_Site_ID	Local_Site_Name	Latitude	Longitude	Measurement_Scale	Measurement_Scale_Defi
28	36-005-0133	PFIZER LAB SITE	40.867900	-73.878090	URBAN SCALE	4 KM TO
29	36-061-0135	CCNY	40.819760	-73.948250	URBAN SCALE	4 KM TO
30	36-081-0124	QUEENS COLLEGE 2	40.736140	-73.821530	NEIGHBORHOOD	500 M T(
31	36-081-0125	Queens College Near Road	40.739264	-73.817694	MIDDLE SCALE	100 M TO

Con esta información podemos localizar a los laboratorios en un mapa e identificar a las zonas de taxis que caen dentro del alcance de la medición correspondiente.