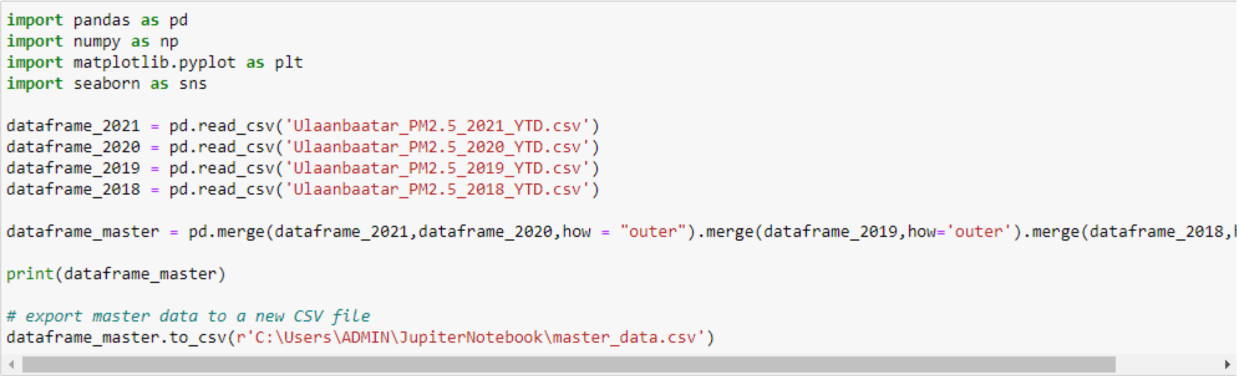
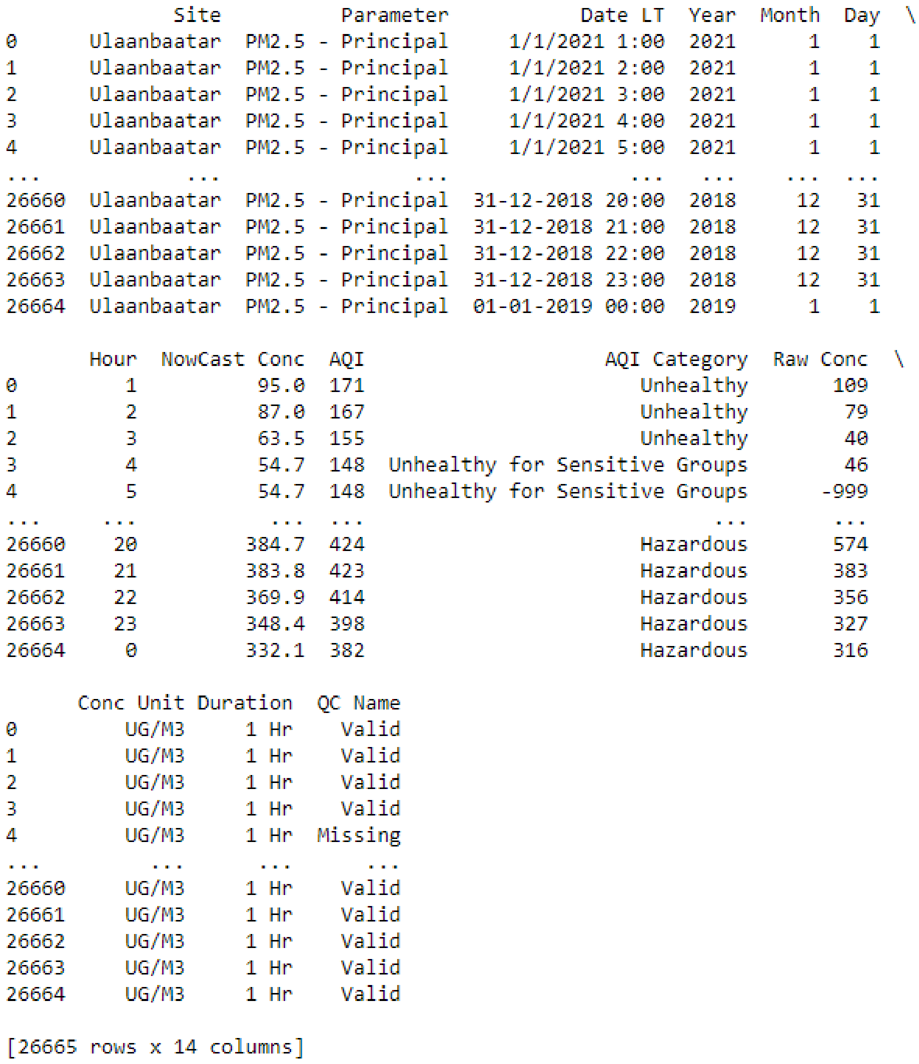
1. Merge the monthly data into a master dataset and categorize on types of variables (categorical and numerical) – provide an explanation the categories that you outlined.



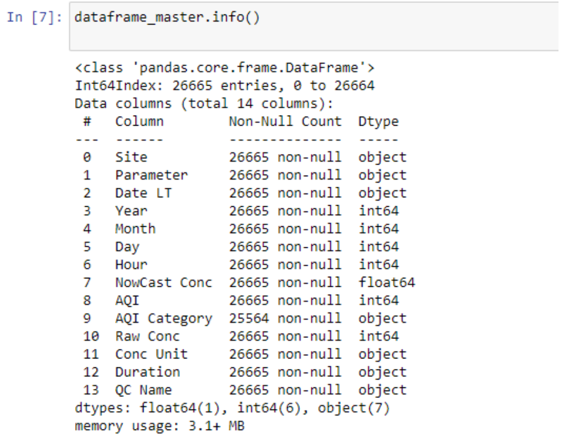
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

dataframe_2021 = pd.read_csv('Ulaanbaatar_PM2.5_2021_YTD.csv')
dataframe_2020 = pd.read_csv('Ulaanbaatar_PM2.5_2020_YTD.csv')
dataframe_2019 = pd.read_csv('Ulaanbaatar_PM2.5_2019_YTD.csv')
dataframe_2018 = pd.read_csv('Ulaanbaatar_PM2.5_2018_YTD.csv')

dataframe_master = pd.merge(dataframe_2021,dataframe_2020,how = "outer").merge(dataframe_2019,how='outer').merge(dataframe_2018,

print(dataframe_master)

# export master data to a new CSV file
dataframe_master.to_csv(r'C:\Users\ADMIN\JupiterNotebook\master_data.csv')
```

print the merged dataset, it has 26665 rows with 14 columns



```
              Site           Parameter           Date LT  Year  Month  Day  \
0      Ulaanbaatar  PM2.5 - Principal     1/1/2021 1:00  2021      1    1
1      Ulaanbaatar  PM2.5 - Principal     1/1/2021 2:00  2021      1    1
2      Ulaanbaatar  PM2.5 - Principal     1/1/2021 3:00  2021      1    1
3      Ulaanbaatar  PM2.5 - Principal     1/1/2021 4:00  2021      1    1
4      Ulaanbaatar  PM2.5 - Principal     1/1/2021 5:00  2021      1    1
...            ...                ...               ...   ...    ...  ...
26660  Ulaanbaatar  PM2.5 - Principal  31-12-2018 20:00  2018     12   31
26661  Ulaanbaatar  PM2.5 - Principal  31-12-2018 21:00  2018     12   31
26662  Ulaanbaatar  PM2.5 - Principal  31-12-2018 22:00  2018     12   31
26663  Ulaanbaatar  PM2.5 - Principal  31-12-2018 23:00  2018     12   31
26664  Ulaanbaatar  PM2.5 - Principal  01-01-2019 00:00  2019      1    1

       Hour  NowCast Conc  AQI                    AQI Category  Raw Conc  \
0         1          95.0  171                       Unhealthy       109
1         2          87.0  167                       Unhealthy        79
2         3          63.5  155                       Unhealthy        40
3         4          54.7  148  Unhealthy for Sensitive Groups        46
4         5          54.7  148  Unhealthy for Sensitive Groups      -999
...     ...           ...  ...                             ...       ...
26660    20         384.7  424                       Hazardous       574
26661    21         383.8  423                       Hazardous       383
26662    22         369.9  414                       Hazardous       356
26663    23         348.4  398                       Hazardous       327
26664     0         332.1  382                       Hazardous       316

      Conc Unit Duration  QC Name
0         UG/M3     1 Hr    Valid
1         UG/M3     1 Hr    Valid
2         UG/M3     1 Hr    Valid
3         UG/M3     1 Hr    Valid
4         UG/M3     1 Hr  Missing
...         ...      ...      ...
26660     UG/M3     1 Hr    Valid
26661     UG/M3     1 Hr    Valid
26662     UG/M3     1 Hr    Valid
26663     UG/M3     1 Hr    Valid
26664     UG/M3     1 Hr    Valid

[26665 rows x 14 columns]
```

list all variable name, data type of master dataset.



```
In [7]: dataframe_master.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 26665 entries, 0 to 26664
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Site          26665 non-null  object
 1   Parameter     26665 non-null  object
 2   Date LT       26665 non-null  object
 3   Year          26665 non-null  int64
 4   Month         26665 non-null  int64
 5   Day           26665 non-null  int64
 6   Hour          26665 non-null  int64
 7   NowCast Conc  26665 non-null  float64
 8   AQI           26665 non-null  int64
 9   AQI Category  25564 non-null  object
 10  Raw Conc      26665 non-null  int64
 11  Conc Unit     26665 non-null  object
 12  Duration      26665 non-null  object
 13  QC Name       26665 non-null  object
dtypes: float64(1), int64(6), object(7)
memory usage: 3.1+ MB
```

There are 3 data type of variables in the master dataset (object, int64 and float64)

- Object: string, text with numeric values (e.q name of sites, duration: 1 Hr)
- Int64: Describe integer numbers. (e.q: -1, 0, 1,2,3…)
- Float64: Describe decimal numbers. (e.q 2.0 , 2.1…)

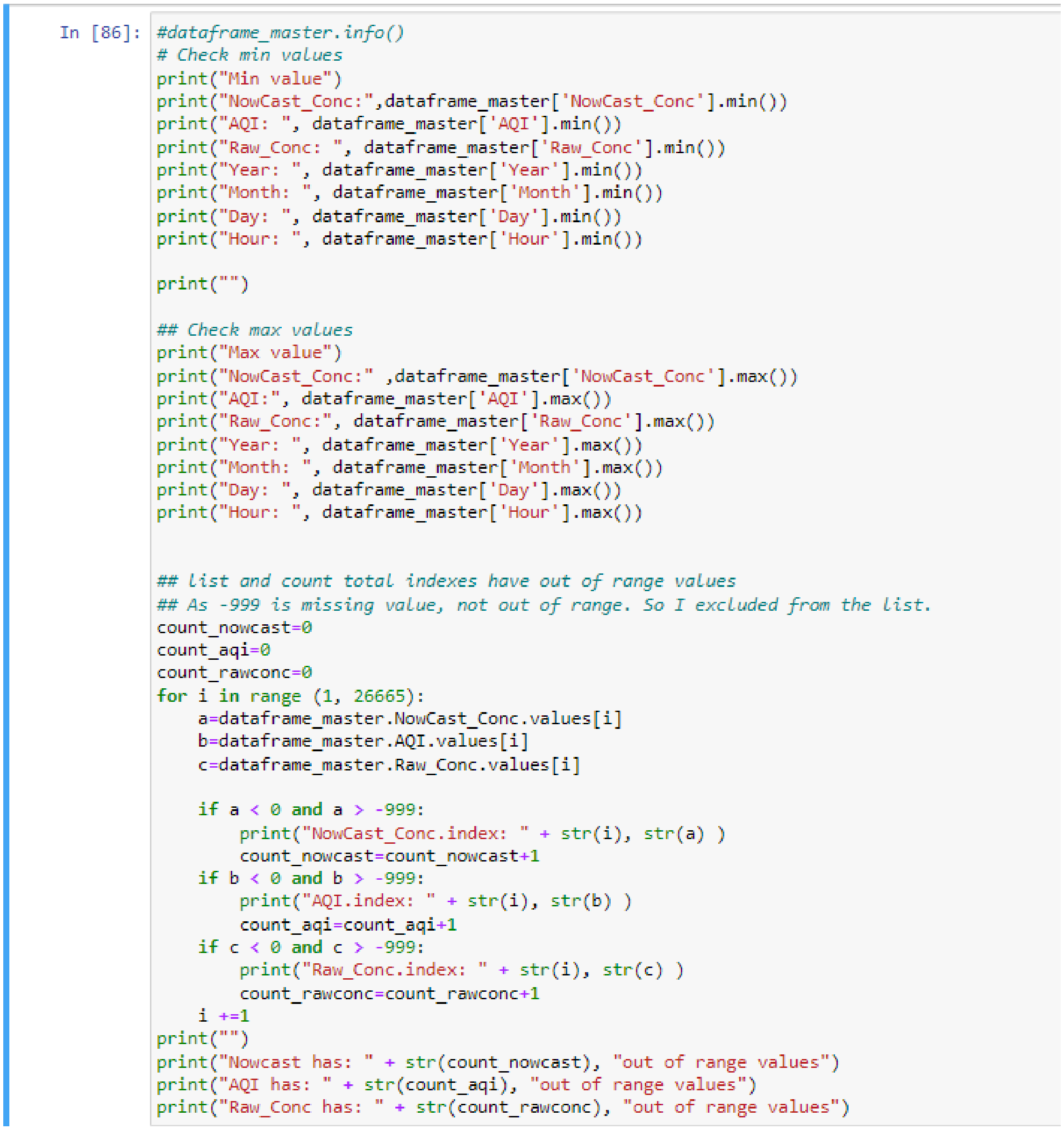
Object is categorical, and int64/float64 are numerical type.

#	**Column**	**Dtype**	**Type of variable**	**Note**
0	Site	object	Categorical	Only one value
1	Parameter	object	Categorical	Only one value
2	Date_LT	object	Categorical (ordinal)	can be ranked by date
3	Year	int64	Numerical (interval	it does not allow 0 value for the year
4	Month	int64	Numerical (Interval	it does not allow 0 value for the month
5	Day	int64	Numerical (Interval	it does not allow 0 value for the day
6	Hour	int64	Numerical (Ratio)	
7	NowCast_Conc	float64	Numerical (interval)	Zero value it means
8	AQI	int64	Numerical (interval)	
9	AQI_Category	object	Categorical (ordinal)	String and can be ranked by level of the air
10	Raw_Conc	int64	Numerical (Ratio)	Allow true zero
11	Conc_Unit	object	Categorical	Only 1 unit, no ranking
12	Duration	object	Categorical	Measure by 1 hour, no ranking
13	QC_Name	object	Categorical (ordinal)	It can be ordered by 4 values

2. Accuracy

a. Check the data for out of range scores. Include the codes and its outputs to show any out range.

Checking all variables which have been assign as int64, float64



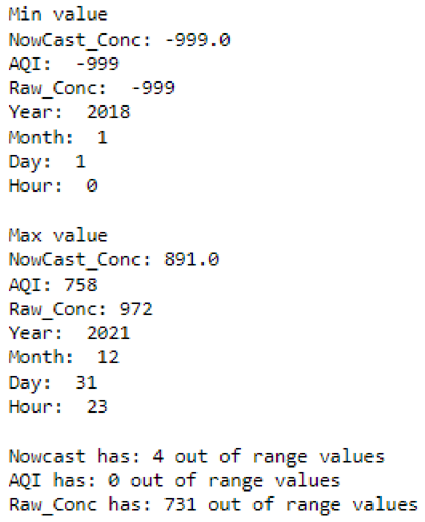
```
In [86]: #dataframe_master.info()
         # Check min values
         print("Min value")
         print("NowCast_Conc:",dataframe_master['NowCast_Conc'].min())
         print("AQI: ", dataframe_master['AQI'].min())
         print("Raw_Conc: ", dataframe_master['Raw_Conc'].min())
         print("Year: ", dataframe_master['Year'].min())
         print("Month: ", dataframe_master['Month'].min())
         print("Day: ", dataframe_master['Day'].min())
         print("Hour: ", dataframe_master['Hour'].min())

         print("")

         ## Check max values
         print("Max value")
         print("NowCast_Conc:" ,dataframe_master['NowCast_Conc'].max())
         print("AQI:", dataframe_master['AQI'].max())
         print("Raw_Conc:", dataframe_master['Raw_Conc'].max())
         print("Year: ", dataframe_master['Year'].max())
         print("Month: ", dataframe_master['Month'].max())
         print("Day: ", dataframe_master['Day'].max())
         print("Hour: ", dataframe_master['Hour'].max())


         ## List and count total indexes have out of range values
         ## As -999 is missing value, not out of range. So I excluded from the list.
         count_nowcast=0
         count_aqi=0
         count_rawconc=0
         for i in range (1, 26665):
             a=dataframe_master.NowCast_Conc.values[i]
             b=dataframe_master.AQI.values[i]
             c=dataframe_master.Raw_Conc.values[i]

             if a < 0 and a > -999:
                 print("NowCast_Conc.index: " + str(i), str(a) )
                 count_nowcast=count_nowcast+1
             if b < 0 and b > -999:
                 print("AQI.index: " + str(i), str(b) )
                 count_aqi=count_aqi+1
             if c < 0 and c > -999:
                 print("Raw_Conc.index: " + str(i), str(c) )
                 count_rawconc=count_rawconc+1
             i +=1
         print("")
         print("Nowcast has: " + str(count_nowcast), "out of range values")
         print("AQI has: " + str(count_aqi), "out of range values")
         print("Raw_Conc has: " + str(count_rawconc), "out of range values")
```

Checking out of range by min/max function and get the output below:

Nowcast_Conc, Raw_Conc has out-of-range values because it defines from 0 to above 500.

According to AIRNOW, the air quality index does not include a higher 500 for PM2.5, but it treats as an "extremely hazardous" level. (Ref: https://www.airnow.gov/aqi/aqi-basics/extremely-high-levels-of-pm25/)

However, it has some values lower than 0.



```
Min value
NowCast_Conc: -999.0
AQI:  -999
Raw_Conc:  -999
Year:  2018
Month:  1
Day:  1
Hour:  0

Max value
NowCast_Conc: 891.0
AQI: 758
Raw_Conc: 972
Year:  2021
Month:  12
Day:  31
Hour:  23

Nowcast has: 4 out of range values
AQI has: 0 out of range values
Raw_Conc has: 731 out of range values
```

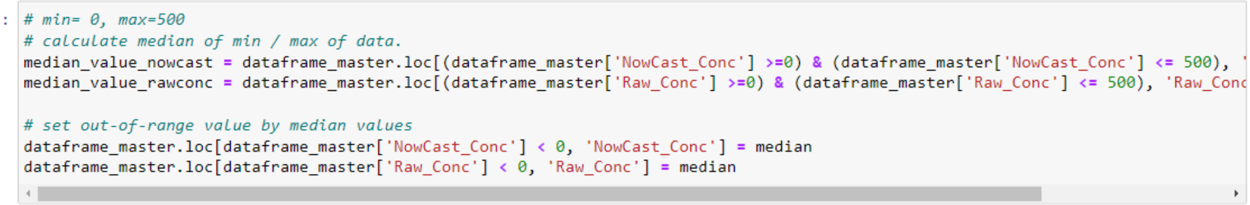
For further details, we can re-execute the code to acknowledge which indexes are invalid range.

b. If necessary, fix the out-of-range scores.

i. Describe how you fixed them.

valid range is defined from 0 – 500, according to the AQI monitoring values. Therefore, I set min=0 and max=500 and calculated by median function. Then I checked any of Nowcast_Conc and Raw_Conc values, which are lower than the min value (0), and replaced by the median result.

ii. Include a R/Python codes and its outputs showing that you fixed the accuracy issues.



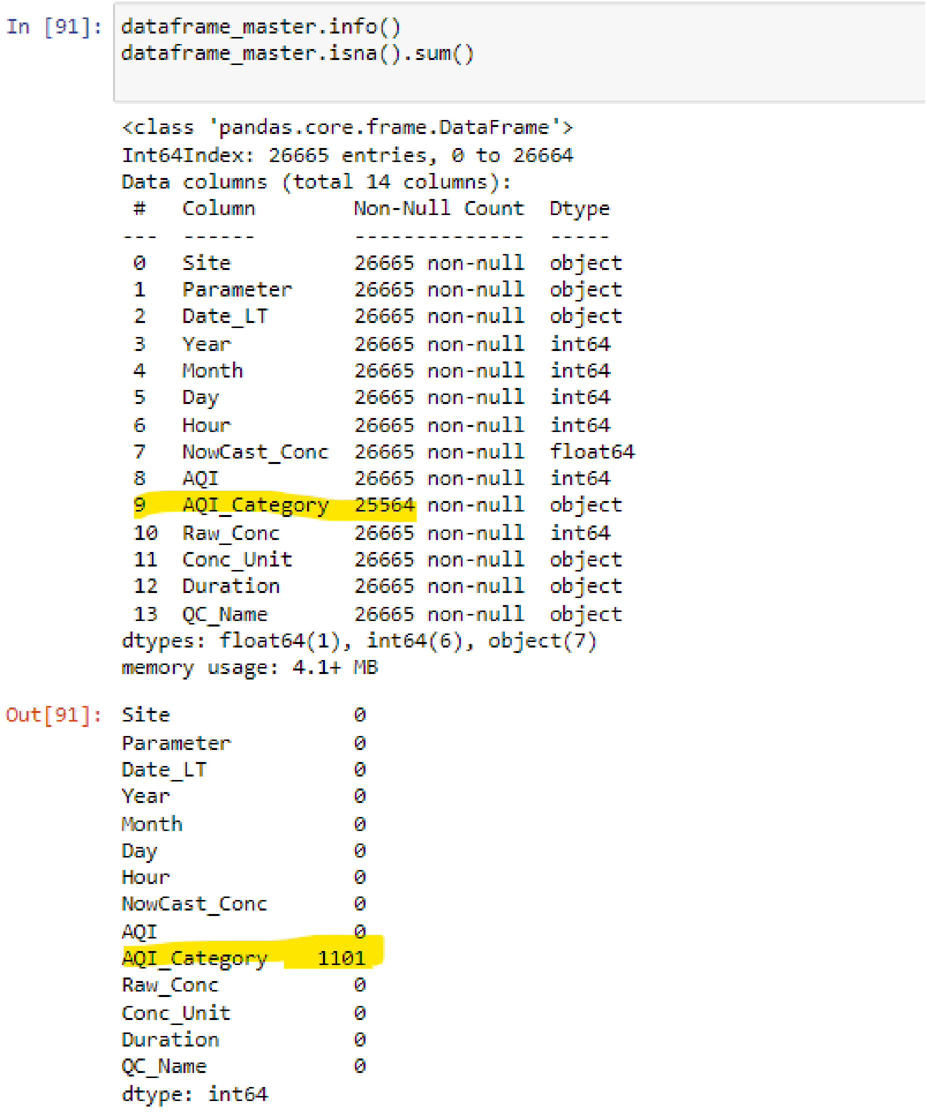
```
: # min= 0, max=500
  # calculate median of min / max of data.
  median_value_nowcast = dataframe_master.loc[(dataframe_master['NowCast_Conc'] >=0) & (dataframe_master['NowCast_Conc'] <= 500), '
  median_value_rawconc = dataframe_master.loc[(dataframe_master['Raw_Conc'] >=0) & (dataframe_master['Raw_Conc'] <= 500), 'Raw_Conc

  # set out-of-range value by median values
  dataframe_master.loc[dataframe_master['NowCast_Conc'] < 0, 'NowCast_Conc'] = median
  dataframe_master.loc[dataframe_master['Raw_Conc'] < 0, 'Raw_Conc'] = median
```

c. Other accuracy issues that you might detect

3. Missing data

a. Include a R/Python output that shows that there is not missing data.

Using isna() to identify blank values in the dataset. All the variables have value except some rows of AQI_Category.

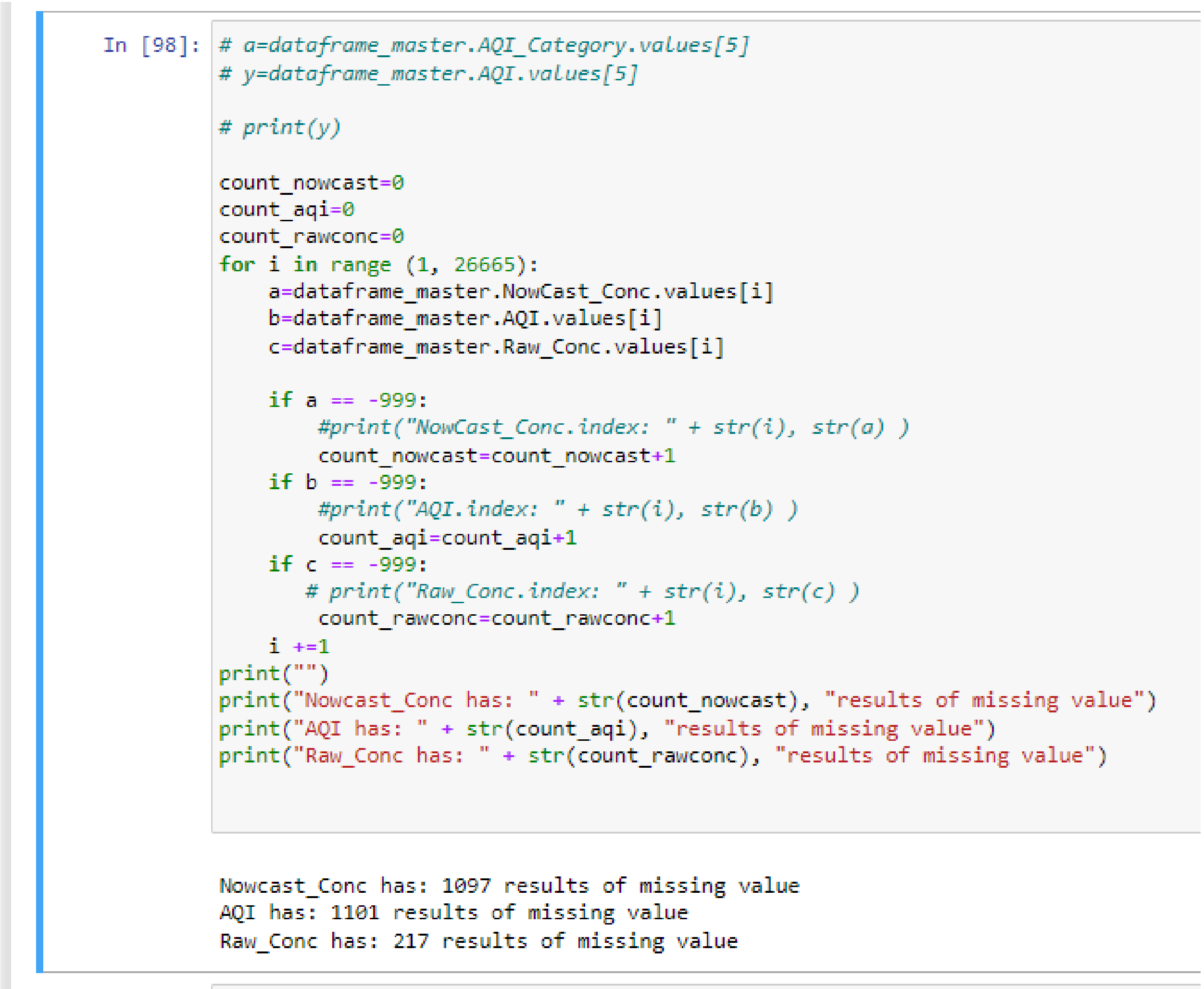


```
In [91]: dataframe_master.info()
         dataframe_master.isna().sum()

         <class 'pandas.core.frame.DataFrame'>
         Int64Index: 26665 entries, 0 to 26664
         Data columns (total 14 columns):
          #   Column         Non-Null Count  Dtype
         ---  ------         --------------  -----
          0   Site           26665 non-null  object
          1   Parameter      26665 non-null  object
          2   Date_LT        26665 non-null  object
          3   Year           26665 non-null  int64
          4   Month          26665 non-null  int64
          5   Day            26665 non-null  int64
          6   Hour           26665 non-null  int64
          7   NowCast_Conc   26665 non-null  float64
          8   AQI            26665 non-null  int64
          9   AQI_Category   25564 non-null  object
          10  Raw_Conc       26665 non-null  int64
          11  Conc_Unit      26665 non-null  object
          12  Duration       26665 non-null  object
          13  QC_Name        26665 non-null  object
         dtypes: float64(1), int64(6), object(7)
         memory usage: 4.1+ MB

Out[91]: Site               0
         Parameter          0
         Date_LT            0
         Year               0
         Month              0
         Day                0
         Hour               0
         NowCast_Conc       0
         AQI                0
         AQI_Category    1101
         Raw_Conc           0
         Conc_Unit          0
         Duration           0
         QC_Name            0
         dtype: int64
```

b. What type of missing data do you appear to have?

I found 3 types of missing data: blank, -999 values and data in an arrangement of date.

List all the results "-999" as missing data.



```
In [98]: # a=dataframe_master.AQI_Category.values[5]
         # y=dataframe_master.AQI.values[5]

         # print(y)

         count_nowcast=0
         count_aqi=0
         count_rawconc=0
         for i in range (1, 26665):
             a=dataframe_master.NowCast_Conc.values[i]
             b=dataframe_master.AQI.values[i]
             c=dataframe_master.Raw_Conc.values[i]

             if a == -999:
                 #print("NowCast_Conc.index: " + str(i), str(a) )
                 count_nowcast=count_nowcast+1
             if b == -999:
                 #print("AQI.index: " + str(i), str(b) )
                 count_aqi=count_aqi+1
             if c == -999:
                # print("Raw_Conc.index: " + str(i), str(c) )
                 count_rawconc=count_rawconc+1
             i +=1
         print("")
         print("Nowcast_Conc has: " + str(count_nowcast), "results of missing value")
         print("AQI has: " + str(count_aqi), "results of missing value")
         print("Raw_Conc has: " + str(count_rawconc), "results of missing value")
```

```
Nowcast_Conc has: 1097 results of missing value
AQI has: 1101 results of missing value
Raw_Conc has: 217 results of missing value
```

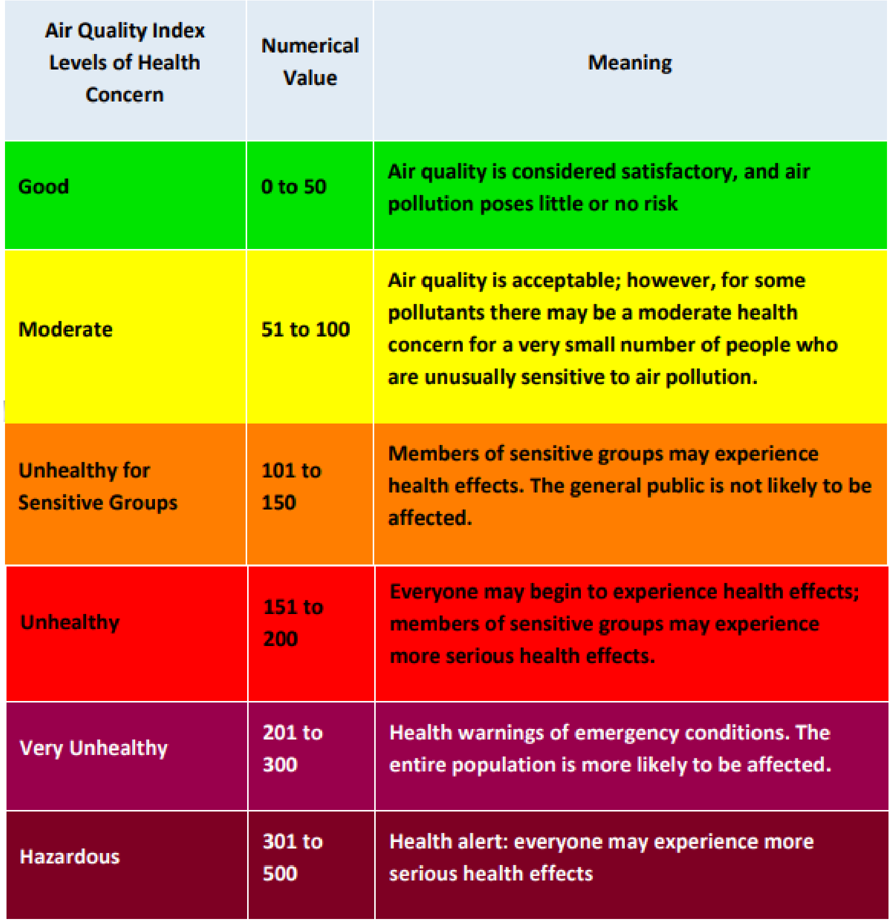
In the dataset 2021, it does not have enough data for 24h on some datetime. For example, on 01-01-2021, it is missing data from 1 am to 5 am. However, when I checked archive data on https://gispub.epa.gov/ about Ulaanbaatar, the data was unavailable between 01.01.2021 and 06.01.2021. Therefore, I ignored the missing fill-up range.

c. If necessary, “fix” the missing data (remember there are several options).

i. Describe what you did to the missing data.

Nowcast_Conc and AQI set missing data by “-999” while AQI_Catergory set by blanks.

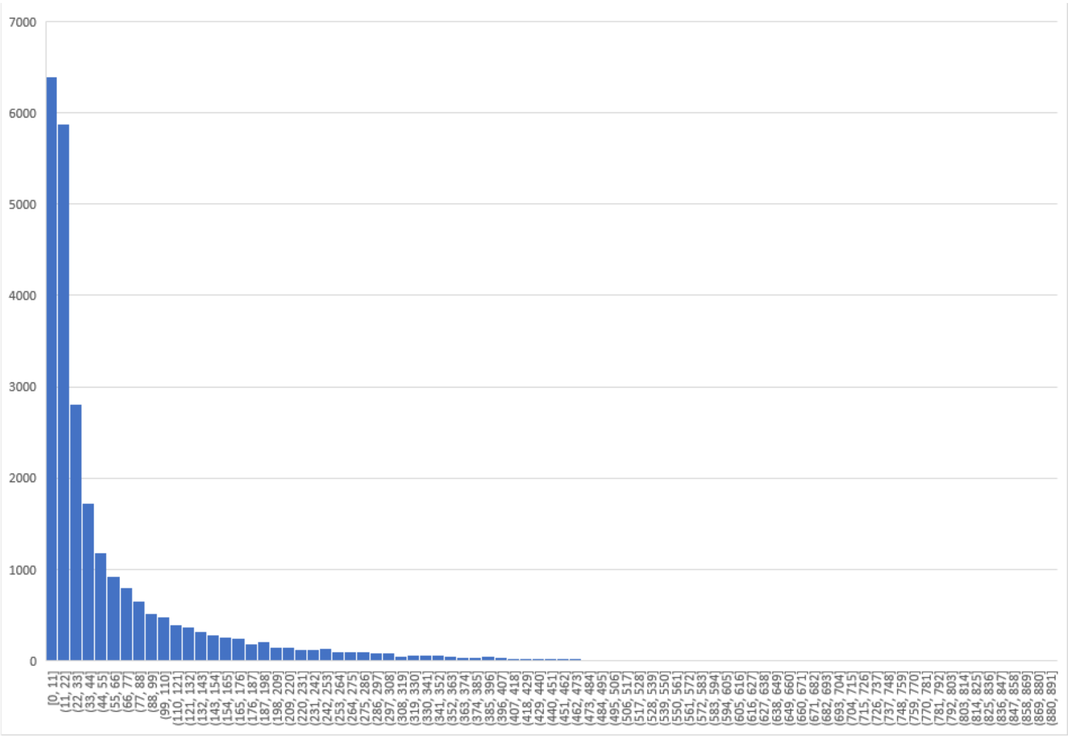
I will use mean or median to handle missing value and then I will categorize AQI_Catergory by air quality index level below:



Air Quality Index Levels of Health Concern	Numerical Value	Meaning
Good	0 to 50	Air quality is considered satisfactory, and air pollution poses little or no risk
Moderate	51 to 100	Air quality is acceptable; however, for some pollutants there may be a moderate health concern for a very small number of people who are unusually sensitive to air pollution.
Unhealthy for Sensitive Groups	101 to 150	Members of sensitive groups may experience health effects. The general public is not likely to be affected.
Unhealthy	151 to 200	Everyone may begin to experience health effects; members of sensitive groups may experience more serious health effects.
Very Unhealthy	201 to 300	Health warnings of emergency conditions. The entire population is more likely to be affected.
Hazardous	301 to 500	Health alert: everyone may experience more serious health effects

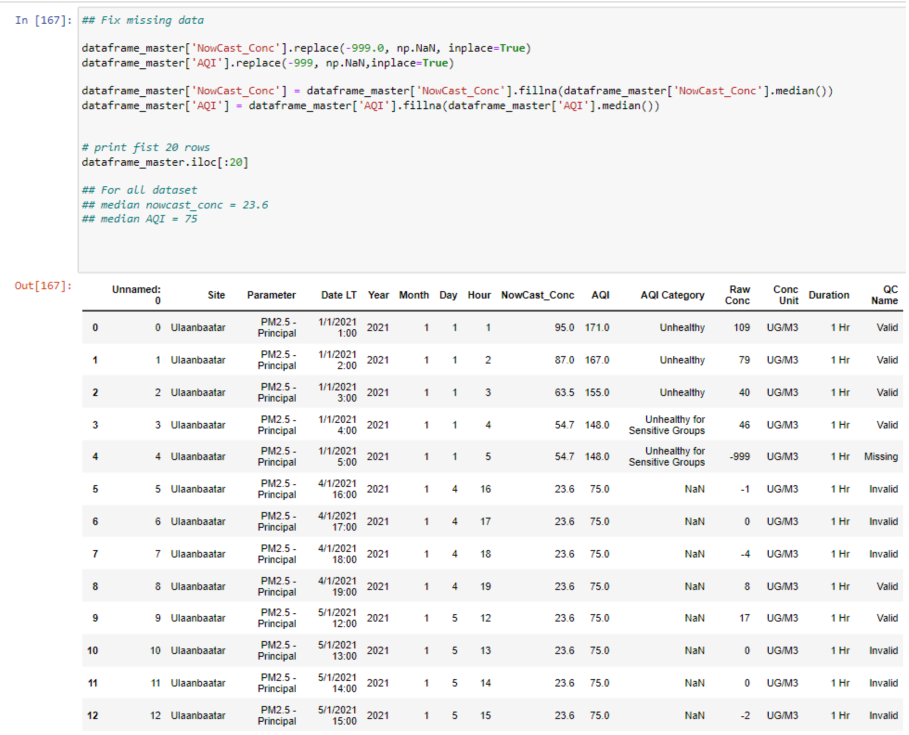
Drawing a histogram and calculating the frequency of Nowcast_Conc, we can see that the value range 0-20 appears the most. When I fix the missing value by mean, the result is 60, while using the median, the result is 23. Therefore, I decided to use the median value to fix all missing data. Nowcast_Conc value will be 23.6, and AQI value will be 75.

As the missing data is set to -999, I have to replace this value with NumPy NaN and then use the median function to calculate it. I tried to convert AQI and Nowcast concentration using Airnow’s testing tool. (ref: www.airnow.gov/aqi/aqi-calculator/)



Frequency of Nowcast_Conc values

ii. Include a R/Python output showing that you fixed the missing data (you may repeat a box you had earlier)



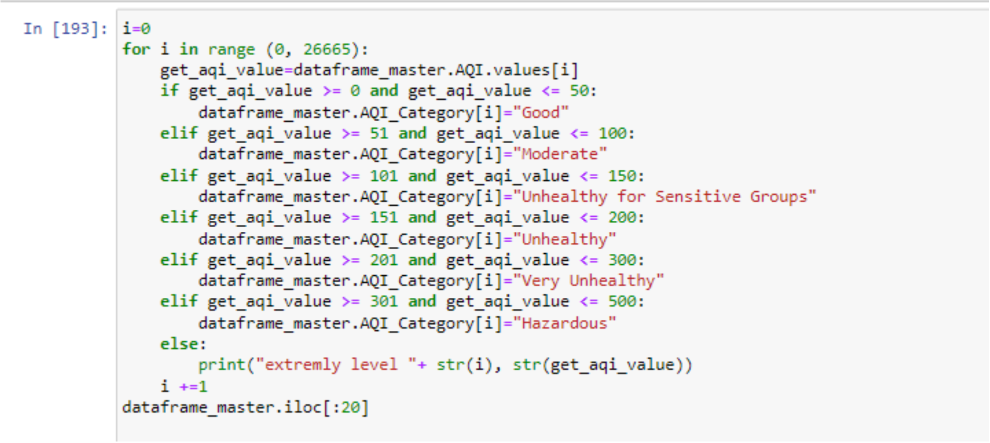
```
In [167]: ## Fix missing data

          dataframe_master['NowCast_Conc'].replace(-999.0, np.NaN, inplace=True)
          dataframe_master['AQI'].replace(-999, np.NaN,inplace=True)

          dataframe_master['NowCast_Conc'] = dataframe_master['NowCast_Conc'].fillna(dataframe_master['NowCast_Conc'].median())
          dataframe_master['AQI'] = dataframe_master['AQI'].fillna(dataframe_master['AQI'].median())


          # print fist 20 rows
          dataframe_master.iloc[:20]

          ## For all dataset
          ## median nowcast_conc = 23.6
          ## median AQI = 75
```

Out[167]:

	Unnamed: 0	Site	Parameter	Date LT	Year	Month	Day	Hour	NowCast_Conc	AQI	AQI Category	Raw Conc	Conc Unit	Duration	QC Name
0	0	Ulaanbaatar	PM2.5 - Principal	1/1/2021 1:00	2021	1	1	1	95.0	171.0	Unhealthy	109	UG/M3	1 Hr	Valid
1	1	Ulaanbaatar	PM2.5 - Principal	1/1/2021 2:00	2021	1	1	2	87.0	167.0	Unhealthy	79	UG/M3	1 Hr	Valid
2	2	Ulaanbaatar	PM2.5 - Principal	1/1/2021 3:00	2021	1	1	3	63.5	155.0	Unhealthy	40	UG/M3	1 Hr	Valid
3	3	Ulaanbaatar	PM2.5 - Principal	1/1/2021 4:00	2021	1	1	4	54.7	148.0	Unhealthy for Sensitive Groups	46	UG/M3	1 Hr	Valid
4	4	Ulaanbaatar	PM2.5 - Principal	1/1/2021 5:00	2021	1	1	5	54.7	148.0	Unhealthy for Sensitive Groups	-999	UG/M3	1 Hr	Missing
5	5	Ulaanbaatar	PM2.5 - Principal	4/1/2021 16:00	2021	1	4	16	23.6	75.0	NaN	-1	UG/M3	1 Hr	Invalid
6	6	Ulaanbaatar	PM2.5 - Principal	4/1/2021 17:00	2021	1	4	17	23.6	75.0	NaN	0	UG/M3	1 Hr	Invalid
7	7	Ulaanbaatar	PM2.5 - Principal	4/1/2021 18:00	2021	1	4	18	23.6	75.0	NaN	-4	UG/M3	1 Hr	Invalid
8	8	Ulaanbaatar	PM2.5 - Principal	4/1/2021 19:00	2021	1	4	19	23.6	75.0	NaN	8	UG/M3	1 Hr	Valid
9	9	Ulaanbaatar	PM2.5 - Principal	5/1/2021 12:00	2021	1	5	12	23.6	75.0	NaN	17	UG/M3	1 Hr	Valid
10	10	Ulaanbaatar	PM2.5 - Principal	5/1/2021 13:00	2021	1	5	13	23.6	75.0	NaN	0	UG/M3	1 Hr	Invalid
11	11	Ulaanbaatar	PM2.5 - Principal	5/1/2021 14:00	2021	1	5	14	23.6	75.0	NaN	0	UG/M3	1 Hr	Invalid
12	12	Ulaanbaatar	PM2.5 - Principal	5/1/2021 15:00	2021	1	5	15	23.6	75.0	NaN	-2	UG/M3	1 Hr	Invalid

.

Using the if statement to categorize AQI_Category value



```
In [193]: i=0
          for i in range (0, 26665):
              get_aqi_value=dataframe_master.AQI.values[i]
              if get_aqi_value >= 0 and get_aqi_value <= 50:
                  dataframe_master.AQI_Category[i]="Good"
              elif get_aqi_value >= 51 and get_aqi_value <= 100:
                  dataframe_master.AQI_Category[i]="Moderate"
              elif get_aqi_value >= 101 and get_aqi_value <= 150:
                  dataframe_master.AQI_Category[i]="Unhealthy for Sensitive Groups"
              elif get_aqi_value >= 151 and get_aqi_value <= 200:
                  dataframe_master.AQI_Category[i]="Unhealthy"
              elif get_aqi_value >= 201 and get_aqi_value <= 300:
                  dataframe_master.AQI_Category[i]="Very Unhealthy"
              elif get_aqi_value >= 301 and get_aqi_value <= 500:
                  dataframe_master.AQI_Category[i]="Hazardous"
              else:
                  print("extremly level "+ str(i), str(get_aqi_value))
              i +=1
          dataframe_master.iloc[:20]
```

4. Outliers:

i. Use scatterplots to detect outliers for each continuous variable.

ii. How many outliers did you have for each continuous variable?

iii. Explain the rationale of the outliers that you identified in (ii)?

5. Univariate Normality

a. Include histograms of the continuous variables.

b. Identify the shape of histograms?