```
import pandas as pd
import numpy as np
import xlsxwriter
import matplotlib.pyplot as plt

df = pd.read_csv("USW00094014.csv")
df = pd.DataFrame(df)
df.columns = ['ID', 'Date', 'Element', 'Value',
              'MFlag', 'QFlag', 'SFlag',
              'Value2']
df1 = pd.DataFrame({'ID': ["USW00094014"],
                    'Date': ["18940101"],
                    'Element': ["TMAX"],
                    'Value': [0],
                    'MFlag': [np.NaN],
                    'QFlag': [np.NaN],
                    'SFlag': ["X"],
                    'Value2': [np.NaN]})
df = pd.concat([df1, df], ignore_index=True)
del df['ID']

means = []
median = []
mode = []
cardinality = []
n_median = []
n_mode = []
n_zero = []
max = []
min = []
stddev = []
n_missing = []
print(df)
for label in df.columns:
    card = 0
    sum = (df[label].isnull()).sum()
    if (sum != 0):
        card = 1
    count = 0
    if label not in ['Value', 'Value2']:
        # Write -999 if stat not
        # applicable
        means.append(-999)
        median.append(-999)
        # medVal = df[label].median()
        mode.append(-999)
        modeVal = df[label].mode()
        cardinality.append(
            str(len(df[label].unique())
                - card))
        n_median.append(-999)
        n_mode.append(-999)
        n_zero.append(
            str((df[label] == 0).sum()))
        max.append(-999)
        min.append(-999)
        stddev.append(-999)
        n_missing.append(
            str((df[label].isnull()).sum()
                +(df[label] == -9999).sum()))
        continue

    means.append(
        df[label].mean(skipna=True))
    median.append(
        df[label].median(skipna=True))
    medVal = df[label].median(skipna=True)
    mode.append(
        df[label].mode(dropna=True))
    modeVal = df[label].mode(dropna=True)
    cardinality.append(
        len(df[label].unique()) - card)
    n_median.append(
        (df[label] == medVal).sum())
    n_mode.append(
        (df[label] == modeVal[0]).sum())
    n_zero.append(
        (df[label] == 0).sum())
    max.append(
        df[label].max())
    min.append(df[label].min())
    stddev.append(
        df[label].std())
    n_missing.append(
        (df[label].isnull()).sum()
        + (df[label] == -9999).sum())

modes = []
for i in mode:
    modes.append(i)

df9 = pd.DataFrame({'Labels':
                        df.columns,
                    'Mean': means,
                    'Median':
                        median,
                    'Mode' :
                        modes,
                    'n_mode' :
                        n_mode,
                    'n_median':
                        n_median,
                    'cardinality':
                        cardinality,
                    'n_zero':
                        n_zero,
                    'max': max,
                    'min':
                        min,
                    'stddev':
                        stddev,
                    'nmissing':
                        n_missing})

# DQR of raw dataset stored as Excel file
df9.to_excel("DQR_RAW_DATASET.xlsx")
df2 = df.pivot(index="Date",
               columns="Element",
               values="Value")

eliminators = df2.columns[0:26]

for label in eliminators:
    mean = df2[label].mean()
    df2[label] = df2[label].fillna(
        value=0)
df2['WV03'] = df2['WV03'].fillna(
    value=0)
#print(df2)
wts = df2.columns[26:46]

# Set the NaN values in all the WT** columns
# to a random value other than 1 and 0.
# Then, replace this value with a 0 or 1
# based on the PRCP and SNOW columns.
# If PRCP column has a value that is not 0,
# then a flag like WT18 which indicates
# "Unknown Source of Precipitation" will
# be 1, as there was precipitation. Further
# , WT16 which indicates Rain will be set
# to 1 if SNOW value in the row is not 0.
# This is because we consider that 7 inches
# of snow is 1 inch of rain. So any non 0
# value of SNOW must be equivalent to some
# amount of rain, even if it is negligible.
for label in wts:
    df2[label] = df2[label].fillna(value
                                   =450)

colsToRetain = []
for i, row in df2.iterrows():
    if row['WT14'] == 450:
        if row['SNOW'] == 0.0:
            row['WT14'] = 0
        else:
            row['WT14'] = 1
        colsToRetain.append('WT14')
    if row['WT15'] == 450:
        if row['SNOW'] == 0.0:
            row['WT15'] = 0
        else:
            row['WT15'] = 1
        colsToRetain.append('WT15')
    if row['WT16'] == 450:
        colsToRetain.append('WT16')
        if row['SNOW'] != 0 \
                or row['PRCP'] != 0:
            row['WT16'] = 1
        else:
            row['WT16'] = 0
    if row['WT17'] == 450:
        if row['SNOW'] == 0.0:
            row['WT17'] = 0
        else:
            row['WT17'] = 1
        colsToRetain.append('WT17')
    if row['WT18'] == 450:
        if row['SNOW'] != 0 \
                or row['PRCP'] != 0:
            row['WT18'] = 1
        else:
            row['WT18'] = 0
        colsToRetain.append('WT18')
    if row['WT09'] == 450:
        if row['SNOW'] == 0:
            row['WT09'] = 0
        else:
            row['WT09'] = 1
        colsToRetain.append('WT09')


for label in wts:
    if label not in colsToRetain:
        del df2[label]

precipflag = []
precipamt = []

for i, row in df2.iterrows():
    if row['PRCP'] > 0:
        precipflag.append(1)
    else:
        precipflag.append(0)
    prcpamt = (row['PRCP']
               / (10 * 25.4)) + \
              (row['SNOW']
               / (25.4 * 8))
    precipamt.append(prcpamt)

df2['PRECIPFLAG'] = precipflag
df2['PRECIPAMT'] = precipamt

nextdayflag = []
nextdayamt = []
flag = 0
amt = 0
count = 0
for i, row in df2.iterrows():
    if count == 0:
        flag = row['PRECIPFLAG']
        amt = row['PRECIPAMT']
        count += 1
        continue
    nextdayflag.append(row['PRECIPFLAG'])
    nextdayamt.append(row['PRECIPAMT'])

nextdayflag.append(flag)
nextdayamt.append(amt)
df2['NEXTDAYPRECIPFLAG'] = nextdayflag
df2['NEXTDAYPRECIPAMT'] = nextdayamt

means = []
median = []
mode = []
cardinality = []
n_median = []
```

```
n_mode = []
n_zero = []
max = []
min = []
stddev = []
n_missing = []

for label in df2.columns:
    card = 0
    sum = (
        df2[label].isnull()).sum()
    if(sum != 0):
        card = 1
    count = 0
    means.append(
        df2[label].mean())
    median.append(
        df2[label].median())
    medVal = df2[label].median()
    mode.append(
        df2[label].mode())
    modeVal = df2[label].mode()
    cardinality.append(
        len(df2[label].unique()) - card)
    n_median.append(
        (df2[label] == medVal).sum())
    n_mode.append(
        (df2[label] == modeVal[0]).sum())
    n_zero.append(
        (df2[label] == 0).sum())
    max.append(
        df2[label].max())
    min.append(
        df2[label].min())
    stddev.append(
        df2[label].std())
    n_missing.append(
        (df2[label].isnull()).sum()
        +(df2[label] == -9999).sum())

modes = []
for i in mode:
    modes.append(i[0])

df4 = pd.DataFrame({'Labels':
                        df2.columns,
                    'Mean':
                        means,
                    'Median':
                        median,
                    'Mode':
                        modes,
                    'n_median':
                        n_median,
                    'n_mode':
                        n_mode,
                    'cardinality':
                        cardinality,
                    'n_zero':
                        n_zero,
                    'max': max,
                    'min': min,
                    'stddev':
                        stddev,
                    'nmissing':
                        n_missing})
# Final Dataset's DQR saved as Excel file.
df4.to_excel("DQR_CLEANED_DATASET.xlsx")

corr = df2.corr()
corr.to_excel("correlations.xlsx")

plt.hist(df2['NEXTDAYPRECIPAMT'])
# plt.show() # -> uncomment to show the
# histogram

freq = df2['NEXTDAYPRECIPFLAG'].value_counts()
print("NEXTDAYPRECIPFLAG Frequency counts",
      freq)

# Save the dataset by Pickling it

df2.to_hdf("TheDataset.h5", key='ML')

df2.to_pickle('Cleaned_Dataset.pkl')

df5 = pd.read_hdf('TheDataset.h5', key='ML')

df6 = pd.read_pickle('Cleaned_Dataset.pkl')
print(df5)
```