Reto-> Entrega 1 (16/11) -> Limpieza, análisis, visualización y kmeans

Sábado, Nov 12 2022

Objetivo: Implementar conocimientos adquiridos a lo largo de curso en el desarrollo de un proyecto con datos reales.

- **Alumno: César Iván Pedrero Martínez**
- **Matrícula: A01366501**
- **Profesor: Julio César Galindo López**

Parte 1: Limpieza de la base de datos

In [759]:

```
import numpy as np
import os.path
import pandas as pd
import matplotlib.pyplot as plt
```

In [760]:

```
DATASET_PATH = "../data/"
DATASET_NAME = "Datos_de_calidad_del_agua_de_sitios_de_monitoreo_de_aguas_subterraneas_2020.csv"
```

In [761]:

```
def warn(msg):
    """
    Simple replacement to the warning method
    from the common library. Prints a given message
    with a warning prefix.
    """
    warnPrefix = "[WARNING]"
    print(warnPrefix, msg)

def log(msg, start=" ", end="\n"):
    """
    Simple replacement to the print method
    from the common library. Prints a given message
    with a log prefix.
    """
    logPrefix = "[INFO]" + start
    print(logPrefix + str(msg), end=end)
```

In [762]:

```
def showRowsWithNan(df: pd.DataFrame, col: str) -> pd.DataFrame:
    """
    Gets the rows from a column with NaN values.
    """
    log("Showing current rows containing NaN for the {} column:".format(col))
    return df[df[col].isna()]

def showRowsWithNanMultiColumn(df: pd.DataFrame, colArray: list) -> pd.DataFrame:
    """
    Gets the rows from a column with NaN values.
    """
    log("Showing current rows containing NaN for the {} columns:".format(colArray))
    nanDF = df[df[colArray].isna().any(axis=1)]
```

```
    return nanDF[colArray]

def showColumnswithNaN(df: pd.DataFrame) -> pd.DataFrame:
    """
    Shows all the columns in the data frame along with a
    boolean value that indicates if the column has NaN values
    or not.
    """
    log("Showing current column NaN state:")
    return df.isna().any()

def showUniques(df: pd.DataFrame, col: str) -> list:
    """
    Returns the unique values of a given column
    in the data frame.
    """
    log("Showing unique values for the {} column:".format(col))
    return df[col].unique()

def showAllDataframeTypes(df) -> pd.DataFrame:
    """
    Shows each column of the data frame
    along with its current data type.
    """
    return df.dtypes
```

In [763]:

```
def replaceNanWithMode(df: pd.DataFrame, col: str) -> pd.DataFrame:
    """
    Replaces the NaN values of a data frame column
    with the mode. Be careful when using this method
    as the mode could be NaN. If this is the case,
    the method will fallback to using the median instead.

    Returns the modified dataframe column.
    """
    log("Cleaning using mode. Number of NaN in the column {}: {}".format(col, df[col].is
na().sum()))
    if np.isnan(df[col].mode().any()):
        warn("The mode of the column {} is NaN. Using the median instead.".format(col))
        return replaceNanWithMedian(df, col)
    return df[col].fillna(df[col].mode()[0])

def replaceNanWithMedian(df: pd.DataFrame, col: str) -> pd.DataFrame:
    """
    Replaces the NaN values of a data frame column
    with the median.

    Returns the modified dataframe column.
    """
    log("Cleaning using median. Number of NaN in the column {}: {}".format(col, df[col].
isna().sum()))
    return df[col].fillna(df[col].median())

def replaceNanWithMean(df: pd.DataFrame, col: str) -> pd.DataFrame:
    """
    Replaces the NaN values of a data frame column
    with the algebraic mean.

    Returns the modified dataframe column.
    """
    log("Cleaning using mean. Number of NaN in the column {}: {}".format(col, df[col].is
na().sum()))
    return df[col].fillna(df[col].mean())

def replaceNanWithCustomValue(df: pd.DataFrame, col: str, customValue: any) -> pd.DataFr
ame:
    """
    Replaces the NaN values of a data frame column
    with the algebraic mean.
```

```
    Returns the modified dataframe column.
    """
    log("Cleaning using a custom value. Number of NaN in the column {}: {}".format(col,
df[col].isna().sum()))
    return df[col].fillna(customValue)

def replaceSpecificValue(df: pd.DataFrame, col: str, oldValue: str, newValue: any) -> pd
.DataFrame:
    """
    Replaces the NaN values of a data frame column
    with the algebraic mean.

    Returns the modified dataframe column.
    """
    log("Number of '{}' in {} being replaced for '{}': {}".format(oldValue, col, newValu
e, len(df[df[col] == oldValue])))
    return df[col].replace(oldValue, newValue)

def changeColumnType(df: pd.DataFrame, col: str, newType: str) -> pd.DataFrame:
    """
    Changes the type of a given column
    to a given type. Be careful with the type
    parameter, as this is a string representing
    the new type.
    """
    log("Updating column {} to {}".format(col, newType))
    return df[col].astype(newType)
```

In [764]:

```
df = pd.read_csv(os.path.join(DATASET_PATH, DATASET_NAME), encoding='latin1')
df
```

Out[764]:

	CLAVE	SITIO	ORGANISMO_DE_CUENCA	ESTADO	MUNICIPIO	ACUIFERO	S
0	DLAGU6	POZO SAN GIL	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	ASIENTOS	VALLE DE CHICALOTE	
1	DLAGU6516	POZO R013 CAÑADA HONDA	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	AGUASCALIENTES	VALLE DE CHICALOTE	
2	DLAGU7	POZO COSIO	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	COSIO	VALLE DE AGUASCALIENTES	
3	DLAGU9	POZO EL SALITRILLO	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	RINCON DE ROMOS	VALLE DE AGUASCALIENTES	
4	DLBAJ107	RANCHO EL TECOLOTE	PENINSULA DE BAJA CALIFORNIA	BAJA CALIFORNIA SUR	LA PAZ	TODOS SANTOS	
...	...	...	...	...	...	...	
1063	OCRBR5101M1	L-310 (COMUNIDAD SAN MANUEL)	RIO BRAVO	NUEVO LEON	LINARES	CITRICOLA SUR	
1064	OCRBR5102M1	L-305 (EJIDO OJO DE AGUA LAS CRUCESITAS)	RIO BRAVO	NUEVO LEON	LINARES	CITRICOLA SUR	
1065	OCRBR5105M2	HACIENDA MEXIQUITO POZO 01	RIO BRAVO	NUEVO LEON	CADEREYTA JIMENEZ	CITRICOLA NORTE	
1066	OCRBR5106M1	COMUNIDAD LOS POCITOS	RIO BRAVO	NUEVO LEON	GALEANA	NAVIDAD-POTOSI-RAICES	
1067	OCRBR5109M1	COMUNIDAD LA REFORMA	RIO BRAVO	NUEVO LEON	GALEANA	NAVIDAD-POTOSI-RAICES	

1068 rows × 57 columns

In [765]:

```
df.dtypes
```

Out[765]:

```
CLAVE                    object
SITIO                    object
ORGANISMO_DE_CUENCA      object
ESTADO                   object
MUNICIPIO                object
ACUIFERO                 object
SUBTIPO                  object
LONGITUD                float64
LATITUD                 float64
PERIODO                   int64
ALC_mg/L                float64
CALIDAD_ALC              object
CONDUCT_mS/cm           float64
CALIDAD_CONDUC           object
SDT_mg/L                float64
SDT_M_mg/L               object
CALIDAD_SDT_ra           object
CALIDAD_SDT_salin        object
FLUORUROS_mg/L           object
CALIDAD_FLUO             object
DUR_mg/L                 object
CALIDAD_DUR              object
COLI_FEC_NMP/100_mL      object
CALIDAD_COLI_FEC         object
N_NO3_mg/L               object
CALIDAD_N_NO3            object
AS_TOT_mg/L              object
CALIDAD_AS               object
CD_TOT_mg/L              object
CALIDAD_CD               object
CR_TOT_mg/L              object
CALIDAD_CR               object
HG_TOT_mg/L              object
CALIDAD_HG               object
PB_TOT_mg/L              object
CALIDAD_PB               object
MN_TOT_mg/L              object
CALIDAD_MN               object
FE_TOT_mg/L              object
CALIDAD_FE               object
SEMAFORO                 object
CONTAMINANTES            object
CUMPLE_CON_ALC           object
CUMPLE_CON_COND          object
CUMPLE_CON_SDT_ra        object
CUMPLE_CON_SDT_salin     object
CUMPLE_CON_FLUO          object
CUMPLE_CON_DUR           object
CUMPLE_CON_CF            object
CUMPLE_CON_NO3           object
CUMPLE_CON_AS            object
CUMPLE_CON_CD            object
CUMPLE_CON_CR            object
CUMPLE_CON_HG            object
CUMPLE_CON_PB            object
CUMPLE_CON_MN            object
CUMPLE_CON_FE            object
dtype: object
```

In [766]:

```
# Replace the < values for a very small value not in the column. The idea here is to repl
```

```
ace the < with the same smallest.
# This to prevent the replacement of a number that may actually be the smallest in the da
taframe. Some big int values
# are going to be left like that because that's the smallest value in the data (1.1, 20).
df["COLI_FEC_NMP/100_mL"] = replaceSpecificValue(df, "COLI_FEC_NMP/100_mL", "<1.1", "1.1
")
df["AS_TOT_mg/L"] = replaceSpecificValue(df, "AS_TOT_mg/L", "<0.01", "0.001")
df["CD_TOT_mg/L"] = replaceSpecificValue(df, "CD_TOT_mg/L", "<0.003", "0.0003")
df["CR_TOT_mg/L"] = replaceSpecificValue(df, "CR_TOT_mg/L", "<0.005", "0.0005")
df["HG_TOT_mg/L"] = replaceSpecificValue(df, "HG_TOT_mg/L", "<0.0005", "0.00005")
df["PB_TOT_mg/L"] = replaceSpecificValue(df, "PB_TOT_mg/L", "<0.005", "0.0005")
df["MN_TOT_mg/L"] = replaceSpecificValue(df, "MN_TOT_mg/L", "<0.0015", "0.00015")
df["FE_TOT_mg/L"] = replaceSpecificValue(df, "FE_TOT_mg/L", "<0.025", "0.0025")
df["FLUORUROS_mg/L"] = replaceSpecificValue(df, "FLUORUROS_mg/L", "<0.2", "0.002")
df["DUR_mg/L"] = replaceSpecificValue(df, "DUR_mg/L", "<20", "20")
df["N_NO3_mg/L"] = replaceSpecificValue(df, "N_NO3_mg/L", "<0.02", "0.002")
df["SDT_M_mg/L"] = replaceSpecificValue(df, "SDT_M_mg/L", "<25", "25")
```

```
[INFO] Number of '<1.1' in COLI_FEC_NMP/100_mL being replaced for '1.1': 737
[INFO] Number of '<0.01' in AS_TOT_mg/L being replaced for '0.001': 815
[INFO] Number of '<0.003' in CD_TOT_mg/L being replaced for '0.0003': 1066
[INFO] Number of '<0.005' in CR_TOT_mg/L being replaced for '0.0005': 854
[INFO] Number of '<0.0005' in HG_TOT_mg/L being replaced for '0.00005': 968
[INFO] Number of '<0.005' in PB_TOT_mg/L being replaced for '0.0005': 1038
[INFO] Number of '<0.0015' in MN_TOT_mg/L being replaced for '0.00015': 545
[INFO] Number of '<0.025' in FE_TOT_mg/L being replaced for '0.0025': 401
[INFO] Number of '<0.2' in FLUORUROS_mg/L being replaced for '0.002': 162
[INFO] Number of '<20' in DUR_mg/L being replaced for '20': 26
[INFO] Number of '<0.02' in N_NO3_mg/L being replaced for '0.002': 65
[INFO] Number of '<25' in SDT_M_mg/L being replaced for '25': 1
```

In [767]:

```
# Now lets change the type of all the columns meant to be floats.

def changeValuesToFloat(cols):
    for col in cols:
        df[col] = changeColumnType(df, col, "float64")

cols = ["COLI_FEC_NMP/100_mL", "AS_TOT_mg/L", "CD_TOT_mg/L", "CR_TOT_mg/L",
        "HG_TOT_mg/L", "PB_TOT_mg/L", "MN_TOT_mg/L", "FE_TOT_mg/L",
        "FLUORUROS_mg/L", "DUR_mg/L", "N_NO3_mg/L", "SDT_M_mg/L"]

changeValuesToFloat(cols)
```

```
[INFO] Updating column COLI_FEC_NMP/100_mL to float64
[INFO] Updating column AS_TOT_mg/L to float64
[INFO] Updating column CD_TOT_mg/L to float64
[INFO] Updating column CR_TOT_mg/L to float64
[INFO] Updating column HG_TOT_mg/L to float64
[INFO] Updating column PB_TOT_mg/L to float64
[INFO] Updating column MN_TOT_mg/L to float64
[INFO] Updating column FE_TOT_mg/L to float64
[INFO] Updating column FLUORUROS_mg/L to float64
[INFO] Updating column DUR_mg/L to float64
[INFO] Updating column N_NO3_mg/L to float64
[INFO] Updating column SDT_M_mg/L to float64
```

In [768]:

```
# Now lets change the boolean values. We will replace the 'ND' (NO DATA) with the mode of
the column.

cols = ["CUMPLE_CON_ALC", "CUMPLE_CON_COND", "CUMPLE_CON_SDT_ra",
        "CUMPLE_CON_SDT_salin", "CUMPLE_CON_FLUO", "CUMPLE_CON_DUR",
        "CUMPLE_CON_CF", "CUMPLE_CON_NO3", "CUMPLE_CON_AS", "CUMPLE_CON_CD",
        "CUMPLE_CON_CR", "CUMPLE_CON_HG", "CUMPLE_CON_PB", "CUMPLE_CON_MN",
        "CUMPLE_CON_FE"]

for col in cols:
    df[col] = replaceSpecificValue(df, col, "ND", df[col].mode()[0])
    df[col] = replaceSpecificValue(df, col, "SI", 1)
```

```
    df[col] = replaceSpecificValue(df, col, "NO", 0)
    df[col] = changeColumnType(df, col, "int64")
```

```
[INFO] Number of 'ND' in CUMPLE_CON_ALC being replaced for 'SI': 4
[INFO] Number of 'SI' in CUMPLE_CON_ALC being replaced for '1': 1009
[INFO] Number of 'NO' in CUMPLE_CON_ALC being replaced for '0': 59
[INFO] Updating column CUMPLE_CON_ALC to int64
[INFO] Number of 'ND' in CUMPLE_CON_COND being replaced for 'SI': 6
[INFO] Number of 'SI' in CUMPLE_CON_COND being replaced for '1': 945
[INFO] Number of 'NO' in CUMPLE_CON_COND being replaced for '0': 123
[INFO] Updating column CUMPLE_CON_COND to int64
[INFO] Number of 'ND' in CUMPLE_CON_SDT_ra being replaced for 'SI': 2
[INFO] Number of 'SI' in CUMPLE_CON_SDT_ra being replaced for '1': 997
[INFO] Number of 'NO' in CUMPLE_CON_SDT_ra being replaced for '0': 71
[INFO] Updating column CUMPLE_CON_SDT_ra to int64
[INFO] Number of 'ND' in CUMPLE_CON_SDT_salin being replaced for 'SI': 2
[INFO] Number of 'SI' in CUMPLE_CON_SDT_salin being replaced for '1': 997
[INFO] Number of 'NO' in CUMPLE_CON_SDT_salin being replaced for '0': 71
[INFO] Updating column CUMPLE_CON_SDT_salin to int64
[INFO] Number of 'ND' in CUMPLE_CON_FLUO being replaced for 'SI': 0
[INFO] Number of 'SI' in CUMPLE_CON_FLUO being replaced for '1': 876
[INFO] Number of 'NO' in CUMPLE_CON_FLUO being replaced for '0': 192
[INFO] Updating column CUMPLE_CON_FLUO to int64
[INFO] Number of 'ND' in CUMPLE_CON_DUR being replaced for 'SI': 1
[INFO] Number of 'SI' in CUMPLE_CON_DUR being replaced for '1': 842
[INFO] Number of 'NO' in CUMPLE_CON_DUR being replaced for '0': 226
[INFO] Updating column CUMPLE_CON_DUR to int64
[INFO] Number of 'ND' in CUMPLE_CON_CF being replaced for 'SI': 0
[INFO] Number of 'SI' in CUMPLE_CON_CF being replaced for '1': 1007
[INFO] Number of 'NO' in CUMPLE_CON_CF being replaced for '0': 61
[INFO] Updating column CUMPLE_CON_CF to int64
[INFO] Number of 'ND' in CUMPLE_CON_NO3 being replaced for 'SI': 1
[INFO] Number of 'SI' in CUMPLE_CON_NO3 being replaced for '1': 986
[INFO] Number of 'NO' in CUMPLE_CON_NO3 being replaced for '0': 82
[INFO] Updating column CUMPLE_CON_NO3 to int64
[INFO] Number of 'ND' in CUMPLE_CON_AS being replaced for 'SI': 0
[INFO] Number of 'SI' in CUMPLE_CON_AS being replaced for '1': 941
[INFO] Number of 'NO' in CUMPLE_CON_AS being replaced for '0': 127
[INFO] Updating column CUMPLE_CON_AS to int64
[INFO] Number of 'ND' in CUMPLE_CON_CD being replaced for 'SI': 0
[INFO] Number of 'SI' in CUMPLE_CON_CD being replaced for '1': 1066
[INFO] Number of 'NO' in CUMPLE_CON_CD being replaced for '0': 2
[INFO] Updating column CUMPLE_CON_CD to int64
[INFO] Number of 'ND' in CUMPLE_CON_CR being replaced for 'SI': 0
[INFO] Number of 'SI' in CUMPLE_CON_CR being replaced for '1': 1053
[INFO] Number of 'NO' in CUMPLE_CON_CR being replaced for '0': 15
[INFO] Updating column CUMPLE_CON_CR to int64
[INFO] Number of 'ND' in CUMPLE_CON_HG being replaced for 'SI': 0
[INFO] Number of 'SI' in CUMPLE_CON_HG being replaced for '1': 1067
[INFO] Number of 'NO' in CUMPLE_CON_HG being replaced for '0': 1
[INFO] Updating column CUMPLE_CON_HG to int64
[INFO] Number of 'ND' in CUMPLE_CON_PB being replaced for 'SI': 0
[INFO] Number of 'SI' in CUMPLE_CON_PB being replaced for '1': 1056
[INFO] Number of 'NO' in CUMPLE_CON_PB being replaced for '0': 12
[INFO] Updating column CUMPLE_CON_PB to int64
[INFO] Number of 'ND' in CUMPLE_CON_MN being replaced for 'SI': 0
[INFO] Number of 'SI' in CUMPLE_CON_MN being replaced for '1': 982
[INFO] Number of 'NO' in CUMPLE_CON_MN being replaced for '0': 86
[INFO] Updating column CUMPLE_CON_MN to int64
[INFO] Number of 'ND' in CUMPLE_CON_FE being replaced for 'SI': 0
[INFO] Number of 'SI' in CUMPLE_CON_FE being replaced for '1': 932
[INFO] Number of 'NO' in CUMPLE_CON_FE being replaced for '0': 136
[INFO] Updating column CUMPLE_CON_FE to int64
```

In [769]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1068 entries, 0 to 1067
Data columns (total 57 columns):
 #   Column                Non-Null Count  Dtype
```

```
---  ------                 --------------  -----
 0   CLAVE                  1068 non-null   object
 1   SITIO                  1068 non-null   object
 2   ORGANISMO_DE_CUENCA    1068 non-null   object
 3   ESTADO                 1068 non-null   object
 4   MUNICIPIO              1068 non-null   object
 5   ACUIFERO               1068 non-null   object
 6   SUBTIPO                1068 non-null   object
 7   LONGITUD               1068 non-null   float64
 8   LATITUD                1068 non-null   float64
 9   PERIODO                1068 non-null   int64
 10  ALC_mg/L               1064 non-null   float64
 11  CALIDAD_ALC            1064 non-null   object
 12  CONDUCT_mS/cm          1062 non-null   float64
 13  CALIDAD_CONDUC         1062 non-null   object
 14  SDT_mg/L               0 non-null      float64
 15  SDT_M_mg/L             1066 non-null   float64
 16  CALIDAD_SDT_ra         1066 non-null   object
 17  CALIDAD_SDT_salin      1066 non-null   object
 18  FLUORUROS_mg/L         1068 non-null   float64
 19  CALIDAD_FLUO           1068 non-null   object
 20  DUR_mg/L               1067 non-null   float64
 21  CALIDAD_DUR            1067 non-null   object
 22  COLI_FEC_NMP/100_mL    1068 non-null   float64
 23  CALIDAD_COLI_FEC       1068 non-null   object
 24  N_NO3_mg/L             1067 non-null   float64
 25  CALIDAD_N_NO3          1067 non-null   object
 26  AS_TOT_mg/L            1068 non-null   float64
 27  CALIDAD_AS             1068 non-null   object
 28  CD_TOT_mg/L            1068 non-null   float64
 29  CALIDAD_CD             1068 non-null   object
 30  CR_TOT_mg/L            1068 non-null   float64
 31  CALIDAD_CR             1068 non-null   object
 32  HG_TOT_mg/L            1068 non-null   float64
 33  CALIDAD_HG             1068 non-null   object
 34  PB_TOT_mg/L            1068 non-null   float64
 35  CALIDAD_PB             1068 non-null   object
 36  MN_TOT_mg/L            1068 non-null   float64
 37  CALIDAD_MN             1068 non-null   object
 38  FE_TOT_mg/L            1068 non-null   float64
 39  CALIDAD_FE             1068 non-null   object
 40  SEMAFORO               1068 non-null   object
 41  CONTAMINANTES          634 non-null    object
 42  CUMPLE_CON_ALC         1068 non-null   int64
 43  CUMPLE_CON_COND        1068 non-null   int64
 44  CUMPLE_CON_SDT_ra      1068 non-null   int64
 45  CUMPLE_CON_SDT_salin   1068 non-null   int64
 46  CUMPLE_CON_FLUO        1068 non-null   int64
 47  CUMPLE_CON_DUR         1068 non-null   int64
 48  CUMPLE_CON_CF          1068 non-null   int64
 49  CUMPLE_CON_NO3         1068 non-null   int64
 50  CUMPLE_CON_AS          1068 non-null   int64
 51  CUMPLE_CON_CD          1068 non-null   int64
 52  CUMPLE_CON_CR          1068 non-null   int64
 53  CUMPLE_CON_HG          1068 non-null   int64
 54  CUMPLE_CON_PB          1068 non-null   int64
 55  CUMPLE_CON_MN          1068 non-null   int64
 56  CUMPLE_CON_FE          1068 non-null   int64
dtypes: float64(17), int64(16), object(24)
memory usage: 475.7+ KB
```

In [770]:

```
df.isna()
```

Out[770]:

	CLAVE	SITIO	ORGANISMO_DE_CUENCA	ESTADO	MUNICIPIO	ACUIFERO	SUBTIPO	LONGITUD	LATITUD	PERIODO
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False

	CLAVE	SITIO	ORGANISMO_DE_CUENCA	ESTADO	MUNICIPIO	ACUIFERO	SUBTIPO	LONGITUD	LATITUD	PERIODO
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
1063	False	False	False	False	False	False	False	False	False	False
1064	False	False	False	False	False	False	False	False	False	False
1065	False	False	False	False	False	False	False	False	False	False
1066	False	False	False	False	False	False	False	False	False	False
1067	False	False	False	False	False	False	False	False	False	False

1068 rows × 57 columns

In [771]:

```
# Drop empty columns.
df.dropna(how='all', axis=1, inplace=True)
```

In [772]:

```
df.columns
```

Out[772]:

```
Index(['CLAVE', 'SITIO', 'ORGANISMO_DE_CUENCA', 'ESTADO', 'MUNICIPIO',
       'ACUIFERO', 'SUBTIPO', 'LONGITUD', 'LATITUD', 'PERIODO', 'ALC_mg/L',
       'CALIDAD_ALC', 'CONDUCT_mS/cm', 'CALIDAD_CONDUC', 'SDT_M_mg/L',
       'CALIDAD_SDT_ra', 'CALIDAD_SDT_salin', 'FLUORUROS_mg/L', 'CALIDAD_FLUO',
       'DUR_mg/L', 'CALIDAD_DUR', 'COLI_FEC_NMP/100_mL', 'CALIDAD_COLI_FEC',
       'N_NO3_mg/L', 'CALIDAD_N_NO3', 'AS_TOT_mg/L', 'CALIDAD_AS',
       'CD_TOT_mg/L', 'CALIDAD_CD', 'CR_TOT_mg/L', 'CALIDAD_CR', 'HG_TOT_mg/L',
       'CALIDAD_HG', 'PB_TOT_mg/L', 'CALIDAD_PB', 'MN_TOT_mg/L', 'CALIDAD_MN',
       'FE_TOT_mg/L', 'CALIDAD_FE', 'SEMAFORO', 'CONTAMINANTES',
       'CUMPLE_CON_ALC', 'CUMPLE_CON_COND', 'CUMPLE_CON_SDT_ra',
       'CUMPLE_CON_SDT_salin', 'CUMPLE_CON_FLUO', 'CUMPLE_CON_DUR',
       'CUMPLE_CON_CF', 'CUMPLE_CON_NO3', 'CUMPLE_CON_AS', 'CUMPLE_CON_CD',
       'CUMPLE_CON_CR', 'CUMPLE_CON_HG', 'CUMPLE_CON_PB', 'CUMPLE_CON_MN',
       'CUMPLE_CON_FE'],
      dtype='object')
```

In [773]:

```
showColumnswithNaN(df)
```

```
[INFO] Showing current column NaN state:
```

Out[773]:

```
CLAVE                    False
SITIO                    False
ORGANISMO_DE_CUENCA      False
ESTADO                   False
MUNICIPIO                False
ACUIFERO                 False
SUBTIPO                  False
LONGITUD                 False
LATITUD                  False
PERIODO                  False
ALC_mg/L                  True
CALIDAD_ALC               True
CONDUCT_mS/cm             True
CALIDAD_CONDUC            True
SDT_M_mg/L                True
CALIDAD_SDT_ra            True
CALIDAD_SDT_salin         True
FLUORUROS_mg/L           False
CALIDAD_FLUO             False
```

```
DUR_mg/L                 True
CALIDAD_DUR              True
COLI_FEC_NMP/100_mL     False
CALIDAD_COLI_FEC        False
N_NO3_mg/L               True
CALIDAD_N_NO3            True
AS_TOT_mg/L             False
CALIDAD_AS              False
CD_TOT_mg/L             False
CALIDAD_CD              False
CR_TOT_mg/L             False
CALIDAD_CR              False
HG_TOT_mg/L             False
CALIDAD_HG              False
PB_TOT_mg/L             False
CALIDAD_PB              False
MN_TOT_mg/L             False
CALIDAD_MN              False
FE_TOT_mg/L             False
CALIDAD_FE              False
SEMAFORO                False
CONTAMINANTES            True
CUMPLE_CON_ALC          False
CUMPLE_CON_COND         False
CUMPLE_CON_SDT_ra       False
CUMPLE_CON_SDT_salin    False
CUMPLE_CON_FLUO         False
CUMPLE_CON_DUR          False
CUMPLE_CON_CF           False
CUMPLE_CON_NO3          False
CUMPLE_CON_AS           False
CUMPLE_CON_CD           False
CUMPLE_CON_CR           False
CUMPLE_CON_HG           False
CUMPLE_CON_PB           False
CUMPLE_CON_MN           False
CUMPLE_CON_FE           False
dtype: bool
```

In [774]:

```
# Cleanse the columns that have null values.
df["ALC_mg/L"] = replaceNanWithMean(df, "ALC_mg/L")
df["CONDUCT_mS/cm"] = replaceNanWithMean(df, "CONDUCT_mS/cm")
df["SDT_M_mg/L"] = replaceNanWithMean(df, "SDT_M_mg/L")
df["DUR_mg/L"] = replaceNanWithMean(df, "DUR_mg/L")
df["N_NO3_mg/L"] = replaceNanWithMean(df, "N_NO3_mg/L")

df["CALIDAD_ALC"] = replaceNanWithMode(df, "CALIDAD_ALC")
df["CALIDAD_CONDUC"] = replaceNanWithMode(df, "CALIDAD_CONDUC")
df["CALIDAD_SDT_ra"] = replaceNanWithMode(df, "CALIDAD_SDT_ra")
df["CALIDAD_SDT_salin"] = replaceNanWithMode(df, "CALIDAD_SDT_salin")
df["CALIDAD_DUR"] = replaceNanWithMode(df, "CALIDAD_DUR")
df["CALIDAD_N_NO3"] = replaceNanWithMode(df, "CALIDAD_N_NO3")

df["CONTAMINANTES"] = replaceNanWithCustomValue(df, "CONTAMINANTES", "N/A")
```

```
[INFO] Cleaning using mean. Number of NaN in the column ALC_mg/L: 4
[INFO] Cleaning using mean. Number of NaN in the column CONDUCT_mS/cm: 6
[INFO] Cleaning using mean. Number of NaN in the column SDT_M_mg/L: 2
[INFO] Cleaning using mean. Number of NaN in the column DUR_mg/L: 1
[INFO] Cleaning using mean. Number of NaN in the column N_NO3_mg/L: 1
[INFO] Cleaning using mode. Number of NaN in the column CALIDAD_ALC: 4
[INFO] Cleaning using mode. Number of NaN in the column CALIDAD_CONDUC: 6
[INFO] Cleaning using mode. Number of NaN in the column CALIDAD_SDT_ra: 2
[INFO] Cleaning using mode. Number of NaN in the column CALIDAD_SDT_salin: 2
[INFO] Cleaning using mode. Number of NaN in the column CALIDAD_DUR: 1
[INFO] Cleaning using mode. Number of NaN in the column CALIDAD_N_NO3: 1
[INFO] Cleaning using a custom value. Number of NaN in the column CONTAMINANTES: 434
```

In [775]:

```
# Re-check empty values.
showColumnswithNaN(df)
```

```
[INFO] Showing current column NaN state:
```

Out[775]:

```
CLAVE                   False
SITIO                   False
ORGANISMO_DE_CUENCA     False
ESTADO                  False
MUNICIPIO               False
ACUIFERO                False
SUBTIPO                 False
LONGITUD                False
LATITUD                 False
PERIODO                 False
ALC_mg/L                False
CALIDAD_ALC             False
CONDUCT_mS/cm           False
CALIDAD_CONDUC          False
SDT_M_mg/L              False
CALIDAD_SDT_ra          False
CALIDAD_SDT_salin       False
FLUORUROS_mg/L          False
CALIDAD_FLUO            False
DUR_mg/L                False
CALIDAD_DUR             False
COLI_FEC_NMP/100_mL     False
CALIDAD_COLI_FEC        False
N_NO3_mg/L              False
CALIDAD_N_NO3           False
AS_TOT_mg/L             False
CALIDAD_AS              False
CD_TOT_mg/L             False
CALIDAD_CD              False
CR_TOT_mg/L             False
CALIDAD_CR              False
HG_TOT_mg/L             False
CALIDAD_HG              False
PB_TOT_mg/L             False
CALIDAD_PB              False
MN_TOT_mg/L             False
CALIDAD_MN              False
FE_TOT_mg/L             False
CALIDAD_FE              False
SEMAFORO                False
CONTAMINANTES           False
CUMPLE_CON_ALC          False
CUMPLE_CON_COND         False
CUMPLE_CON_SDT_ra       False
CUMPLE_CON_SDT_salin    False
CUMPLE_CON_FLUO         False
CUMPLE_CON_DUR          False
CUMPLE_CON_CF           False
CUMPLE_CON_NO3          False
CUMPLE_CON_AS           False
CUMPLE_CON_CD           False
CUMPLE_CON_CR           False
CUMPLE_CON_HG           False
CUMPLE_CON_PB           False
CUMPLE_CON_MN           False
CUMPLE_CON_FE           False
dtype: bool
```

In [776]:

```
# Save the clean dataframe
df.to_csv("../cleansed/Datos_de_calidad_del_agua_de_sitios_de_monitoreo_de_aguas_subterraneas_2020.csv")
```

Parte 2: Exploración de los datos

In [777]:

```
df.describe()
```

Out[777]:

	LONGITUD	LATITUD	PERIODO	ALC_mg/L	CONDUCT_mS/cm	SDT_M_mg/L	FLUORUROS_mg/L	DUR_mg/L
count	1068.000000	1068.000000	1068.0	1068.000000	1068.000000	1068.000000	1068.000000	1068.000000
mean	-101.891007	23.163618	2020.0	235.633759	1138.953013	896.101567	1.045566	347.938073
std	6.703263	3.887670	0.0	116.655014	1242.056683	2748.950627	1.939212	359.500870
min	-116.664250	14.561150	2020.0	26.640000	50.400000	25.000000	0.002000	20.000000
25%	-105.388865	20.212055	2020.0	164.048750	505.500000	337.700000	0.267175	121.274100
50%	-102.174180	22.617190	2020.0	215.825000	822.500000	551.400000	0.503500	245.662350
75%	-98.974716	25.510285	2020.0	292.423750	1321.250000	915.900000	1.139850	453.930000
max	-86.864120	32.677713	2020.0	1650.000000	18577.000000	82170.000000	34.803300	3810.692200

8 rows × 32 columns

In [778]:

```
df.var()
```

```
/var/folders/3z/3340wyws0q3fr4z6_yx4yc7r0000gn/T/ipykernel_37589/1568254755.py:1: FutureW
arning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') i
s deprecated; in a future version this will raise TypeError.  Select only valid columns b
efore calling the reduction.
  df.var()
```

Out[778]:

```
LONGITUD                4.493373e+01
LATITUD                 1.511398e+01
PERIODO                 0.000000e+00
ALC_mg/L                1.360839e+04
CONDUCT_mS/cm           1.542705e+06
SDT_M_mg/L              7.556730e+06
FLUORUROS_mg/L          3.760541e+00
DUR_mg/L                1.292409e+05
COLI_FEC_NMP/100_mL     4.212580e+06
N_NO3_mg/L              6.958544e+01
AS_TOT_mg/L             1.386591e-03
CD_TOT_mg/L             9.734551e-07
CR_TOT_mg/L             2.389952e-02
HG_TOT_mg/L             2.822587e-07
PB_TOT_mg/L             1.360958e-05
MN_TOT_mg/L             1.418600e-01
FE_TOT_mg/L             3.067579e+01
CUMPLE_CON_ALC          5.224052e-02
CUMPLE_CON_COND         1.020003e-01
CUMPLE_CON_SDT_ra       6.211805e-02
CUMPLE_CON_SDT_salin    6.211805e-02
CUMPLE_CON_FLUO         1.475943e-01
CUMPLE_CON_DUR          1.669878e-01
CUMPLE_CON_CF           5.390433e-02
CUMPLE_CON_NO3          7.095044e-02
CUMPLE_CON_AS           1.048715e-01
CUMPLE_CON_CD           1.870904e-03
CUMPLE_CON_CR           1.386066e-02
CUMPLE_CON_HG           9.363296e-04
CUMPLE_CON_PB           1.112012e-02
CUMPLE_CON_MN           7.410957e-02
CUMPLE_CON_FE           1.112293e-01
dtype: float64
```

In [779]:

```
modeDf = df.mode().loc[:0].drop(["CLAVE", "LONGITUD", "LATITUD"], axis=1)

log("Printing mode of all columns:", end="\n"*2)
for col in modeDf.columns:
    log("{}: {}".format(modeDf[col].name, modeDf[col].values))
```

```
[INFO] Printing mode of all columns:

[INFO] SITIO: ['EL FUERTE']
[INFO] ORGANISMO_DE_CUENCA: ['CUENCAS CENTRALES DEL NORTE']
[INFO] ESTADO: ['DURANGO']
[INFO] MUNICIPIO: ['LA PAZ']
[INFO] ACUIFERO: ['PENINSULA DE YUCATAN']
[INFO] SUBTIPO: ['POZO']
[INFO] PERIODO: [2020.]
[INFO] ALC_mg/L: [157.62]
[INFO] CALIDAD_ALC: ['Alta']
[INFO] CONDUCT_mS/cm: [777.]
[INFO] CALIDAD_CONDUC: ['Permisible para riego']
[INFO] SDT_M_mg/L: [292.]
[INFO] CALIDAD_SDT_ra: ['Excelente para riego']
[INFO] CALIDAD_SDT_salin: ['Potable - Dulce']
[INFO] FLUORUROS_mg/L: [0.002]
[INFO] CALIDAD_FLUO: ['Baja']
[INFO] DUR_mg/L: [20.]
[INFO] CALIDAD_DUR: ['Potable - Dura']
[INFO] COLI_FEC_NMP/100_mL: [1.1]
[INFO] CALIDAD_COLI_FEC: ['Potable - Excelente']
[INFO] N_NO3_mg/L: [0.002]
[INFO] CALIDAD_N_NO3: ['Potable - Excelente']
[INFO] AS_TOT_mg/L: [0.001]
[INFO] CALIDAD_AS: ['Potable - Excelente']
[INFO] CD_TOT_mg/L: [0.0003]
[INFO] CALIDAD_CD: ['Potable - Excelente']
[INFO] CR_TOT_mg/L: [0.0005]
[INFO] CALIDAD_CR: ['Potable - Excelente']
[INFO] HG_TOT_mg/L: [5.e-05]
[INFO] CALIDAD_HG: ['Potable - Excelente']
[INFO] PB_TOT_mg/L: [0.0005]
[INFO] CALIDAD_PB: ['Potable - Excelente']
[INFO] MN_TOT_mg/L: [0.00015]
[INFO] CALIDAD_MN: ['Potable - Excelente']
[INFO] FE_TOT_mg/L: [0.0025]
[INFO] CALIDAD_FE: ['Potable - Excelente']
[INFO] SEMAFORO: ['Verde']
[INFO] CONTAMINANTES: ['N/A']
[INFO] CUMPLE_CON_ALC: [1.]
[INFO] CUMPLE_CON_COND: [1.]
[INFO] CUMPLE_CON_SDT_ra: [1.]
[INFO] CUMPLE_CON_SDT_salin: [1.]
[INFO] CUMPLE_CON_FLUO: [1.]
[INFO] CUMPLE_CON_DUR: [1.]
[INFO] CUMPLE_CON_CF: [1.]
[INFO] CUMPLE_CON_NO3: [1.]
[INFO] CUMPLE_CON_AS: [1.]
[INFO] CUMPLE_CON_CD: [1.]
[INFO] CUMPLE_CON_CR: [1.]
[INFO] CUMPLE_CON_HG: [1.]
[INFO] CUMPLE_CON_PB: [1.]
[INFO] CUMPLE_CON_MN: [1.]
[INFO] CUMPLE_CON_FE: [1.]
```

In [780]:

```
# Lets identify correlations.
df.corr()
```

Out[780]:

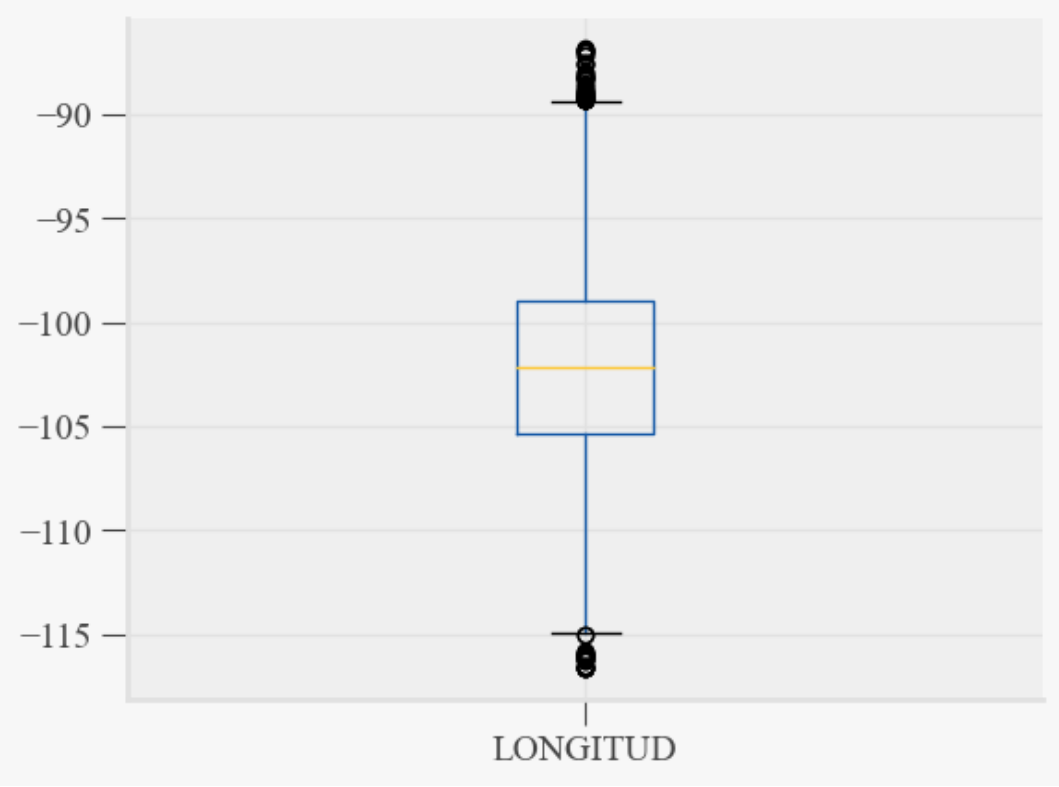
	LONGITUD	LATITUD	PERIODO	ALC_mg/L	CONDUCT_mS/cm	SDT_M_mg/L	FLUORUROS_mg/L
	LONGITUD	LATITUD	PERIODO	ALC_mg/L	CONDUCT_mS/cm	SDT_M_mg/L	FLUORUROS_mg/L
LONGITUD	1.000000	-0.760204	NaN	0.166915	0.060803	-0.013926	-0.133217
LATITUD	-0.760204	1.000000	NaN	-0.079797	0.053506	0.059834	0.140488
PERIODO	NaN	NaN	NaN	NaN	NaN	NaN	NaN
ALC_mg/L	0.166915	-0.079797	NaN	1.000000	0.219654	0.079285	0.068852
CONDUCT_mS/cm	0.060803	0.053506	NaN	0.219654	1.000000	0.286406	-0.023766
SDT_M_mg/L	-0.013926	0.059834	NaN	0.079285	0.286406	1.000000	-0.012572
FLUORUROS_mg/L	-0.133217	0.140488	NaN	0.068852	-0.023766	-0.012572	1.000000
DUR_mg/L	0.088164	0.085163	NaN	0.242957	0.692270	0.347115	-0.149211
COLI_FEC_NMP/100_mL	-0.084443	0.063130	NaN	-0.016448	0.017807	-0.001133	0.002740
N_NO3_mg/L	-0.107550	0.148267	NaN	-0.000437	0.219304	0.101594	-0.018409
AS_TOT_mg/L	-0.095641	0.109621	NaN	0.072679	-0.008135	-0.012825	0.456390
CD_TOT_mg/L	0.043540	-0.052377	NaN	0.030788	0.027355	0.010175	-0.018970
CR_TOT_mg/L	0.058785	-0.059949	NaN	-0.014151	0.004184	0.000106	-0.004528
HG_TOT_mg/L	0.048932	-0.138484	NaN	0.076239	0.049301	0.035854	-0.035151
PB_TOT_mg/L	0.012071	-0.071142	NaN	0.022499	0.026263	0.027313	-0.042707
MN_TOT_mg/L	-0.036184	-0.036322	NaN	0.130048	0.096167	0.019068	-0.049021
FE_TOT_mg/L	0.022348	-0.041960	NaN	0.043616	0.083500	0.020254	-0.009852
CUMPLE_CON_ALC	-0.030184	0.067683	NaN	-0.613898	-0.180084	-0.038048	-0.107561
CUMPLE_CON_COND	-0.059786	-0.030695	NaN	-0.102957	-0.696160	-0.304561	0.028173
CUMPLE_CON_SDT_ra	0.107056	-0.135951	NaN	-0.068904	-0.583125	-0.360502	-0.008076
CUMPLE_CON_SDT_salin	0.107056	-0.135951	NaN	-0.068904	-0.583125	-0.360502	-0.008076
CUMPLE_CON_FLUO	0.096616	-0.105501	NaN	-0.015334	0.009874	0.018691	-0.614892
CUMPLE_CON_DUR	-0.159677	-0.048965	NaN	-0.260131	-0.514644	-0.254651	0.103126
CUMPLE_CON_CF	0.107847	-0.021252	NaN	-0.009116	-0.011593	0.004391	0.038585
CUMPLE_CON_NO3	0.113675	-0.120808	NaN	-0.028578	-0.187521	-0.071404	-0.022988
CUMPLE_CON_AS	0.135760	-0.153838	NaN	-0.036902	0.008659	0.011832	-0.362150
CUMPLE_CON_CD	-0.034033	0.035924	NaN	-0.009830	-0.008742	-0.003265	0.023320
CUMPLE_CON_CR	-0.079644	0.121189	NaN	0.012871	-0.004401	0.001929	0.016322
CUMPLE_CON_HG	-0.021223	0.037977	NaN	-0.031603	-0.067863	-0.015686	0.006216
CUMPLE_CON_PB	-0.012521	0.069544	NaN	-0.039506	-0.057906	-0.005510	0.033417
CUMPLE_CON_MN	0.084977	0.032386	NaN	-0.170677	-0.069763	-0.007210	0.088241
CUMPLE_CON_FE	-0.010544	0.046176	NaN	-0.062126	-0.116753	-0.040282	-0.008678

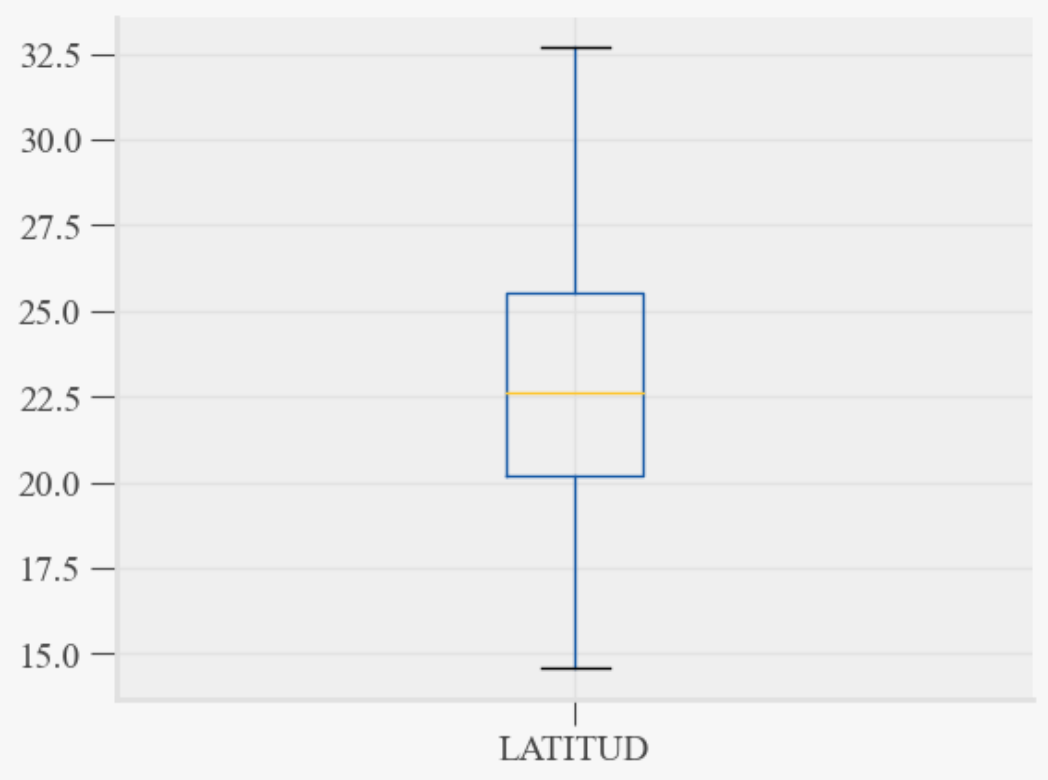
32 rows × 32 columns

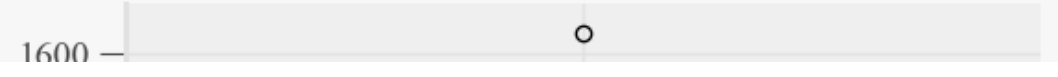
In [70]:

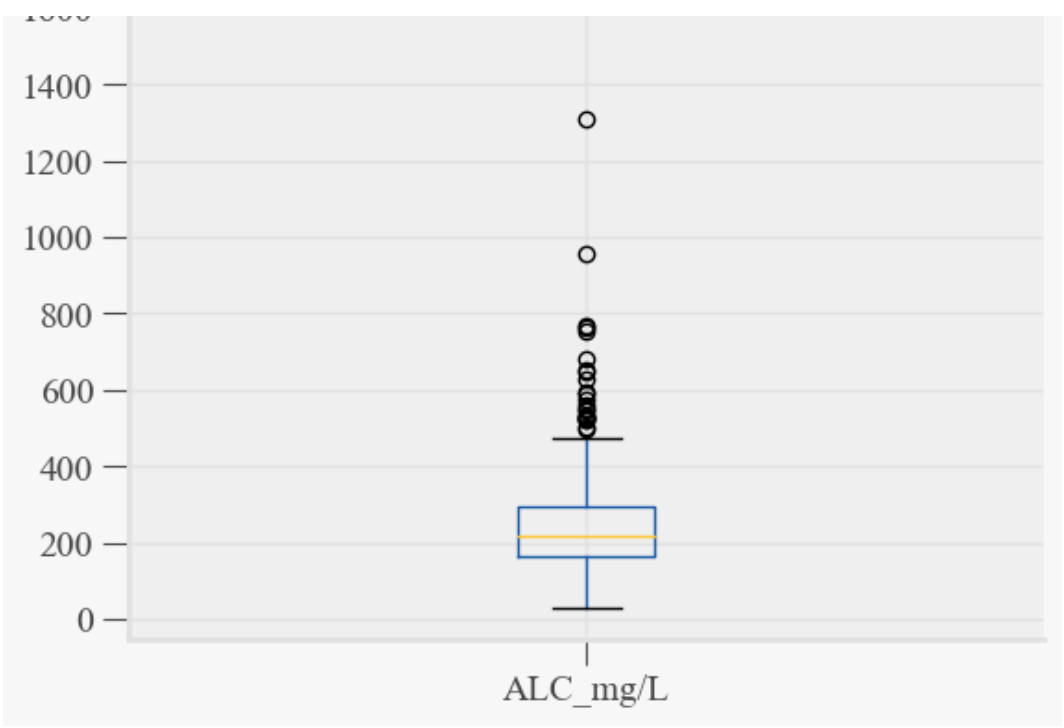
In [781]:

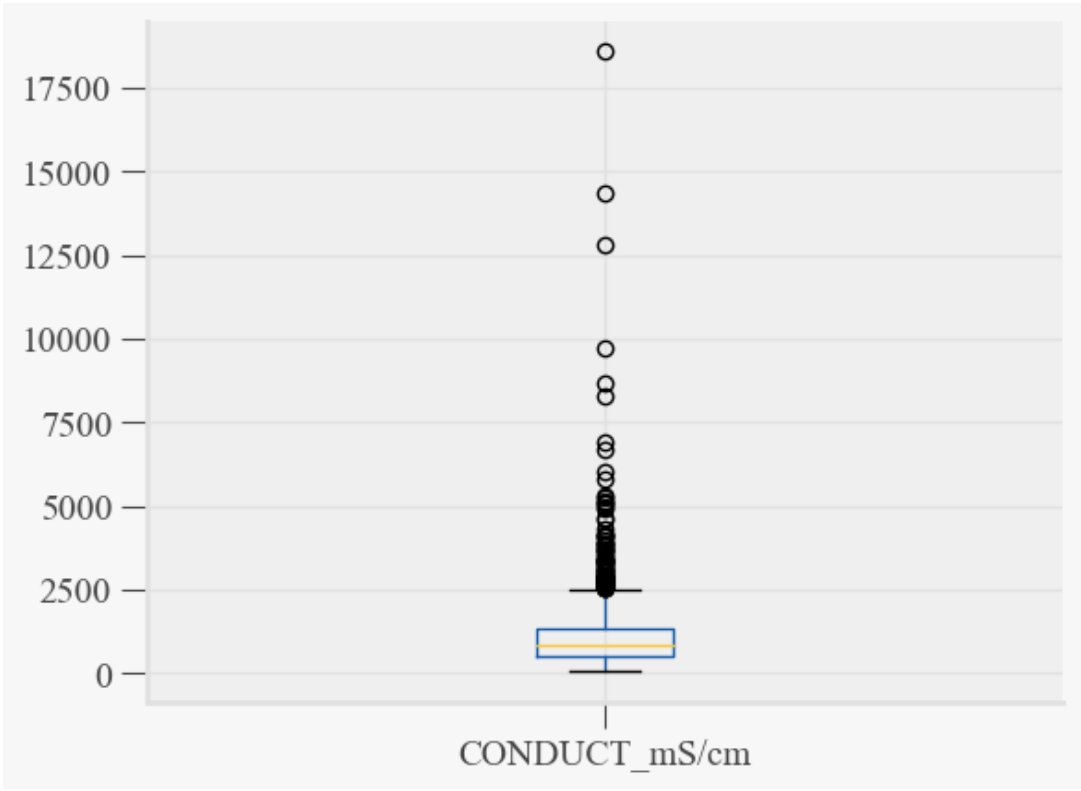
```
# Let's create a box plot for each of the float columns.

def plotBoxForFloatColumns():
    for col in df.columns:
        if df[col].dtype != "float64":
            continue

        name = list(col.lower())
        name[0] = name[0].upper()
        name = "".join(name)

        df.boxplot(column = col).plot()
        plt.show()

plotBoxForFloatColumns()
```

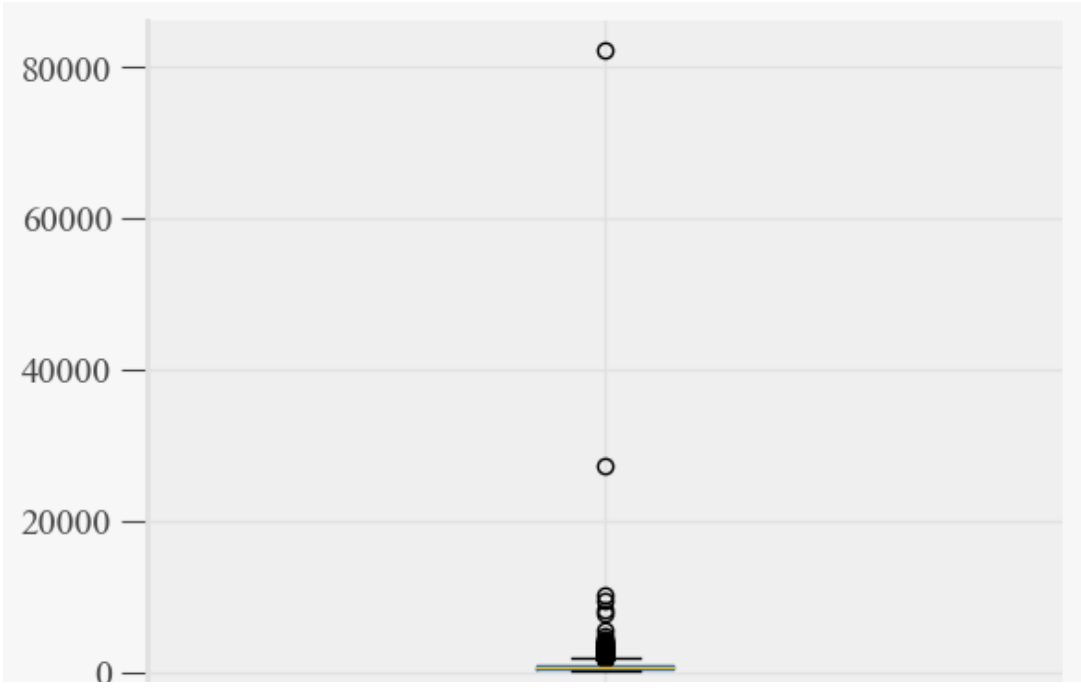




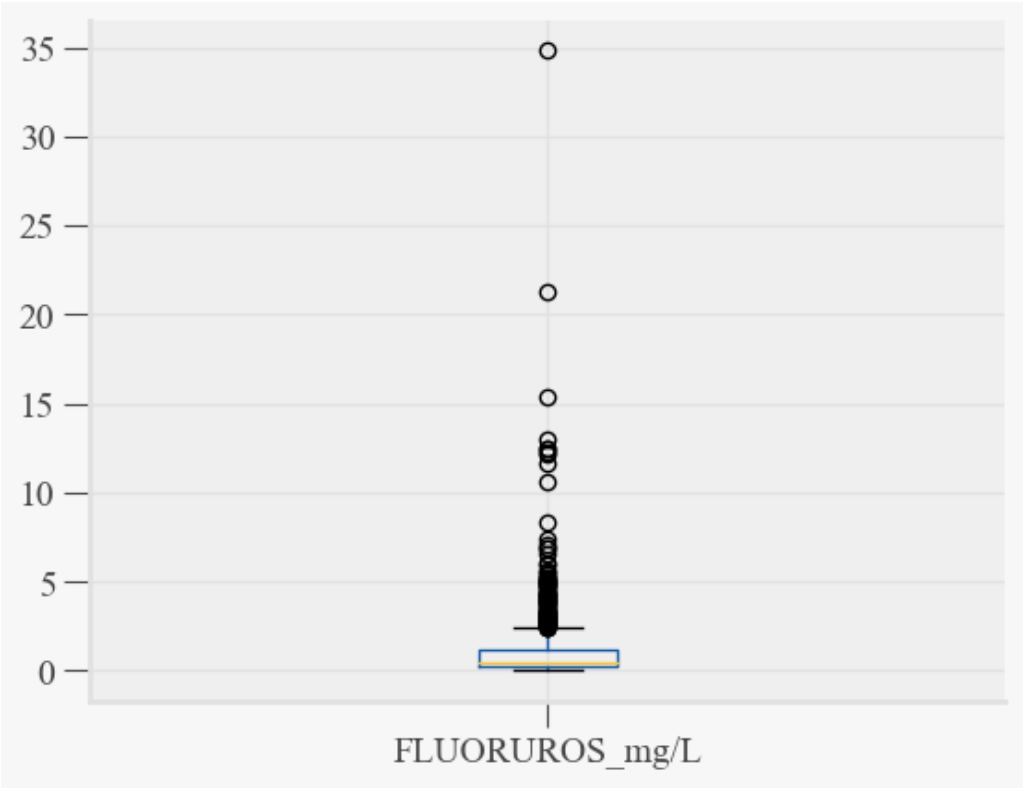


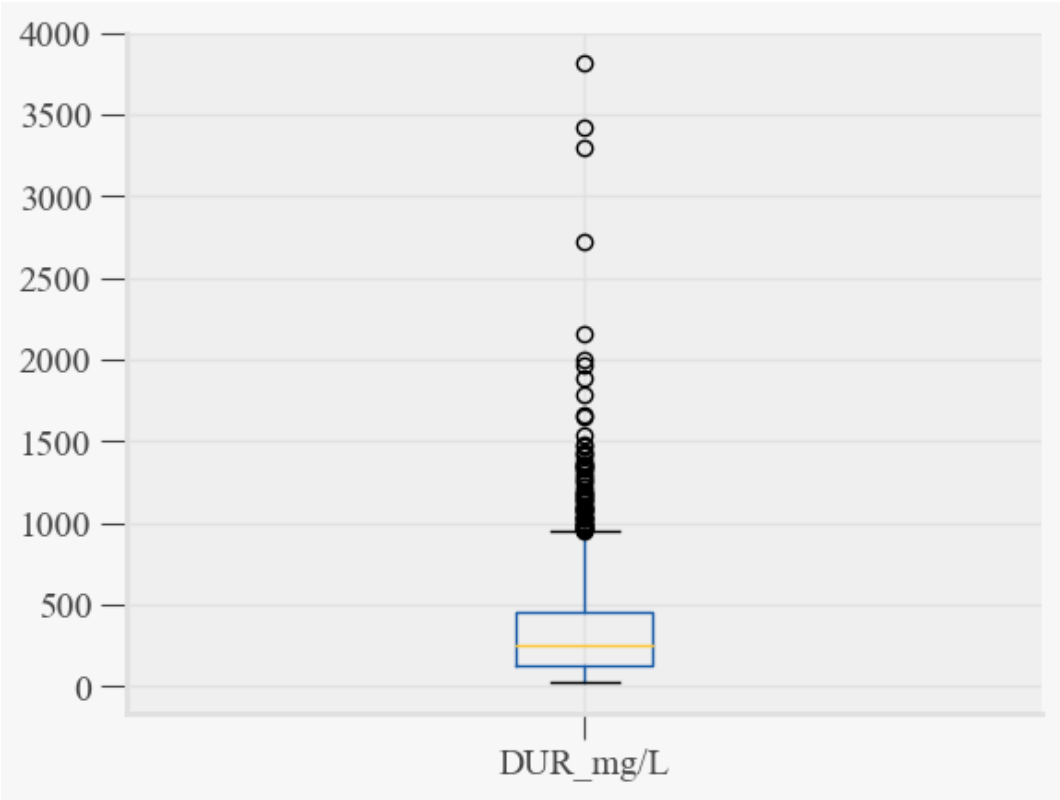

1400
1200
1000
800
600
400
200
0
ALC_mg/L


17500
15000
12500
10000
7500
5000
2500
0
CONDUCT_mS/cm

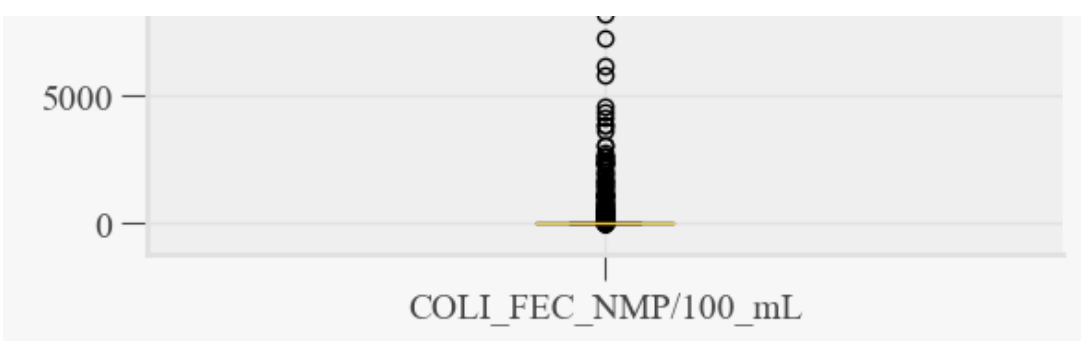

80000
60000
40000
20000
0

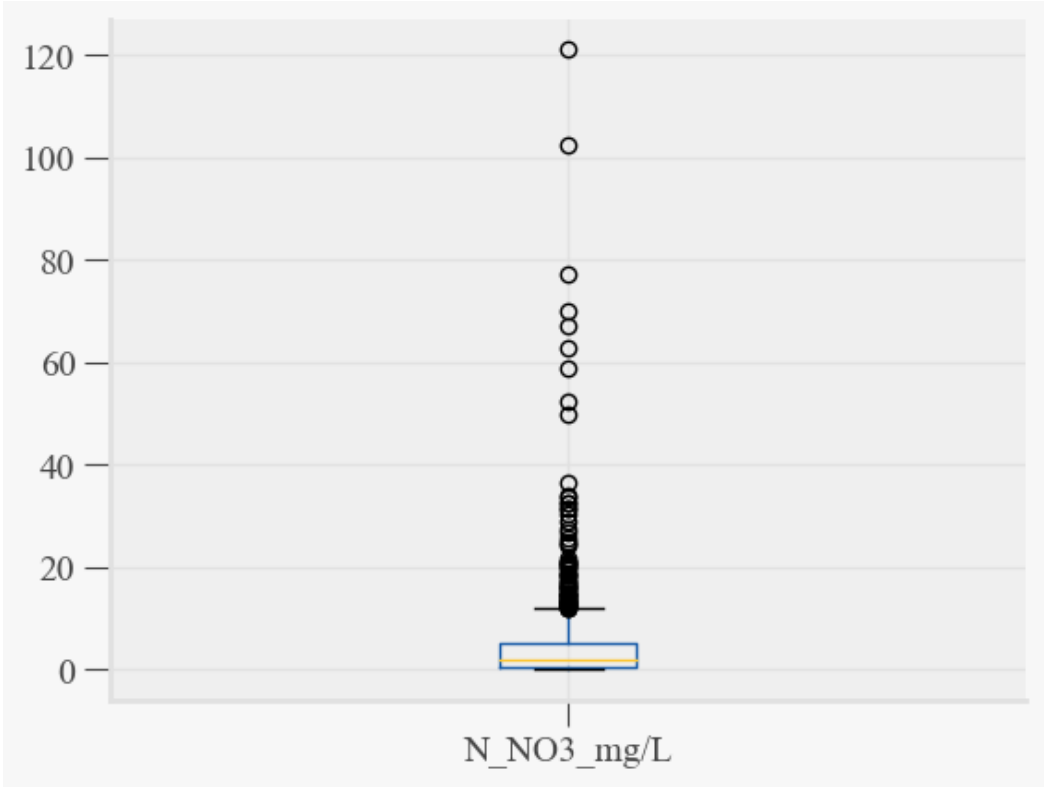

SDT_M_mg/L

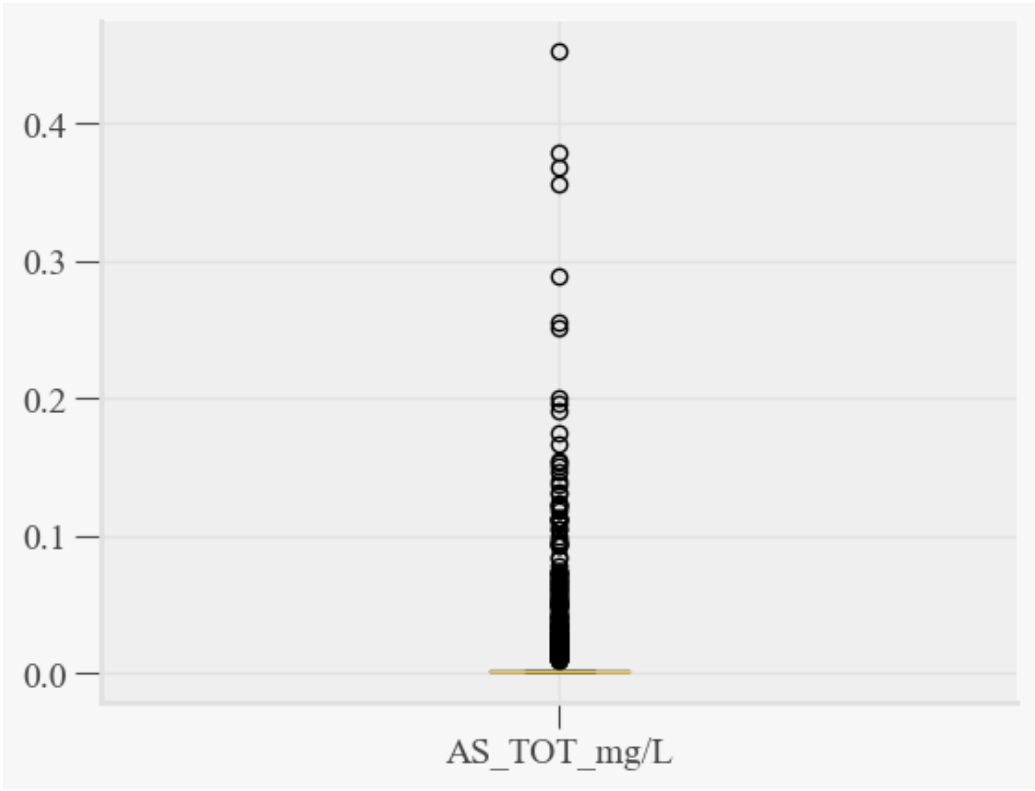

35
30
25
20
15
10
5
0
FLUORUROS_mg/L


4000
3500
3000
2500
2000
1500
1000
500
0
DUR_mg/L

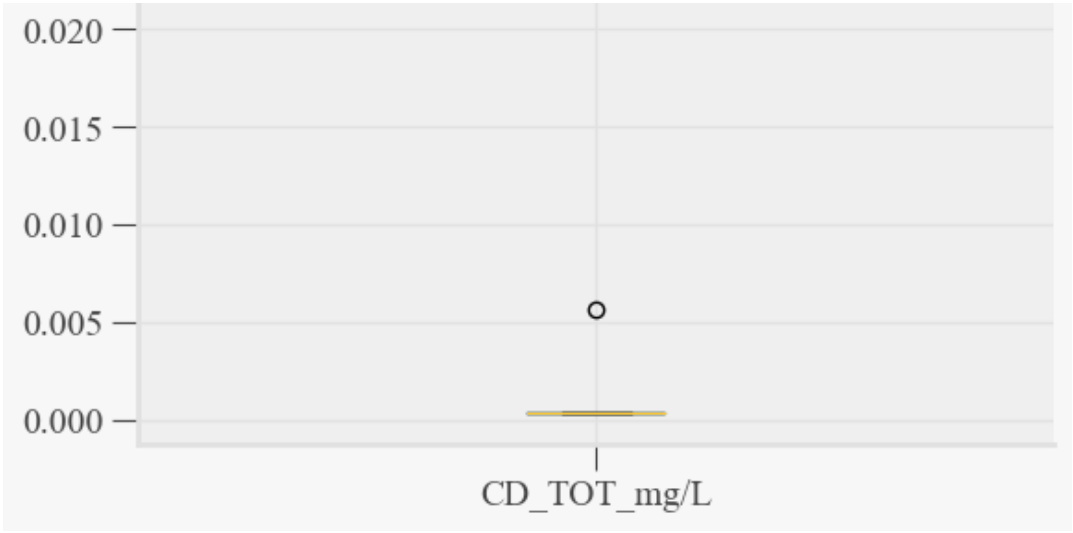

25000
20000
15000
10000

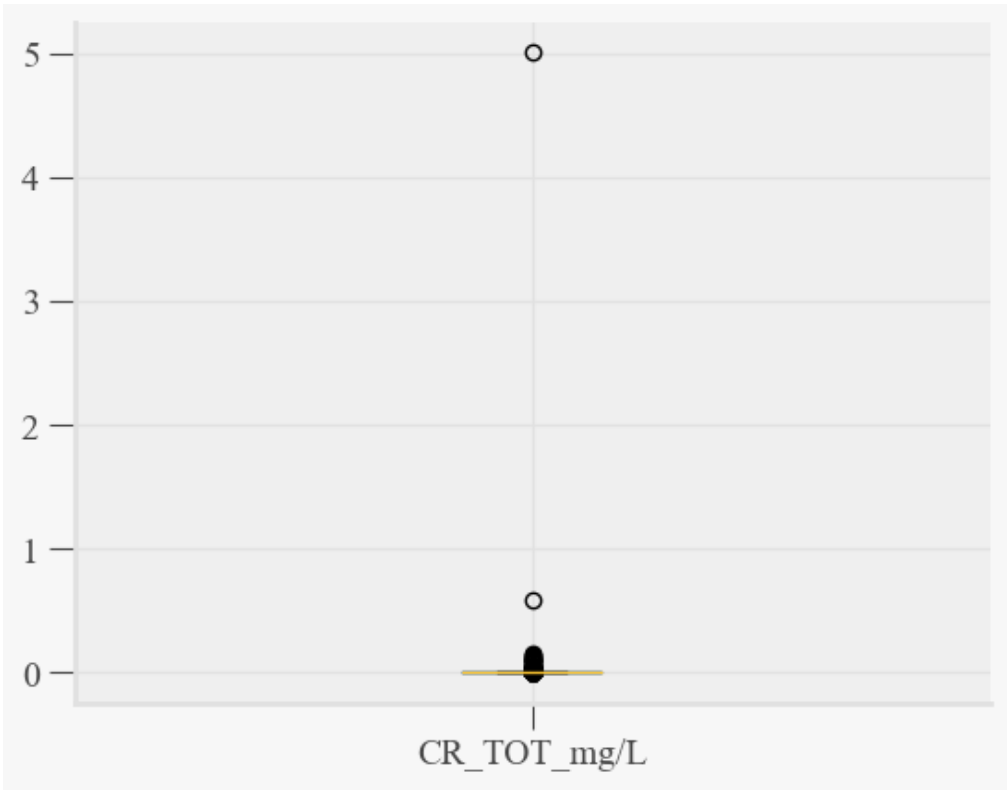

5000
0
COLI_FEC_NMP/100_mL

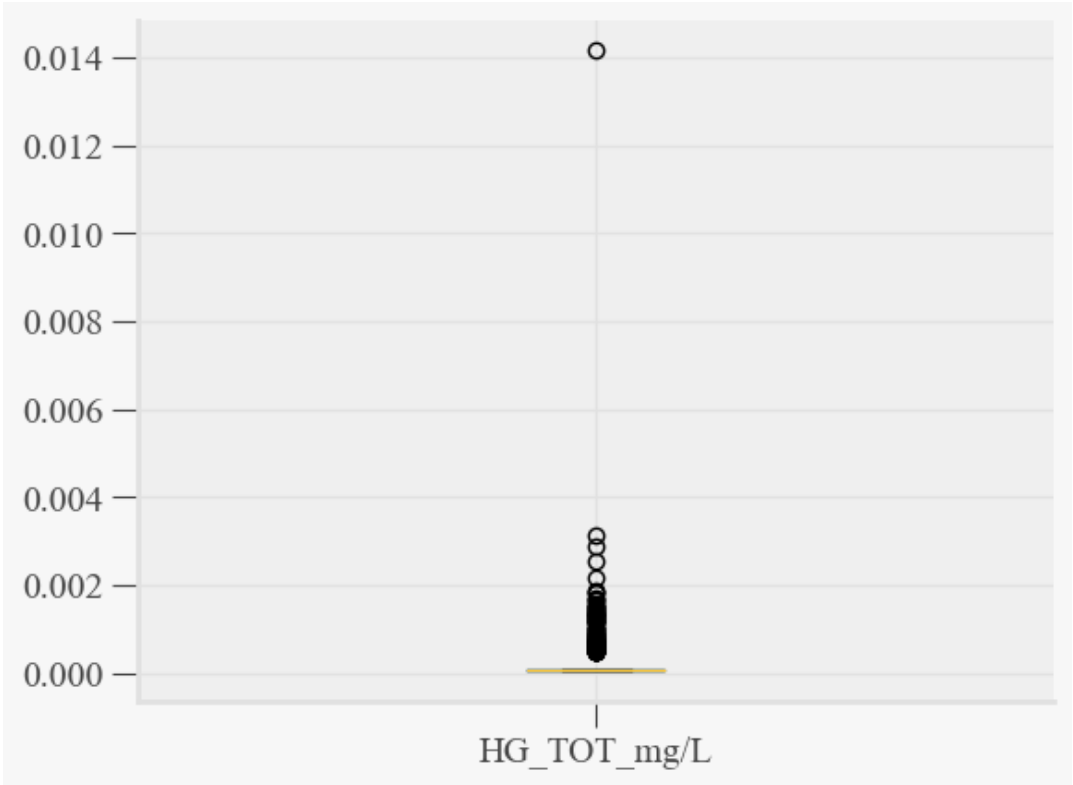

120
100
80
60
40
20
0
N_NO3_mg/L

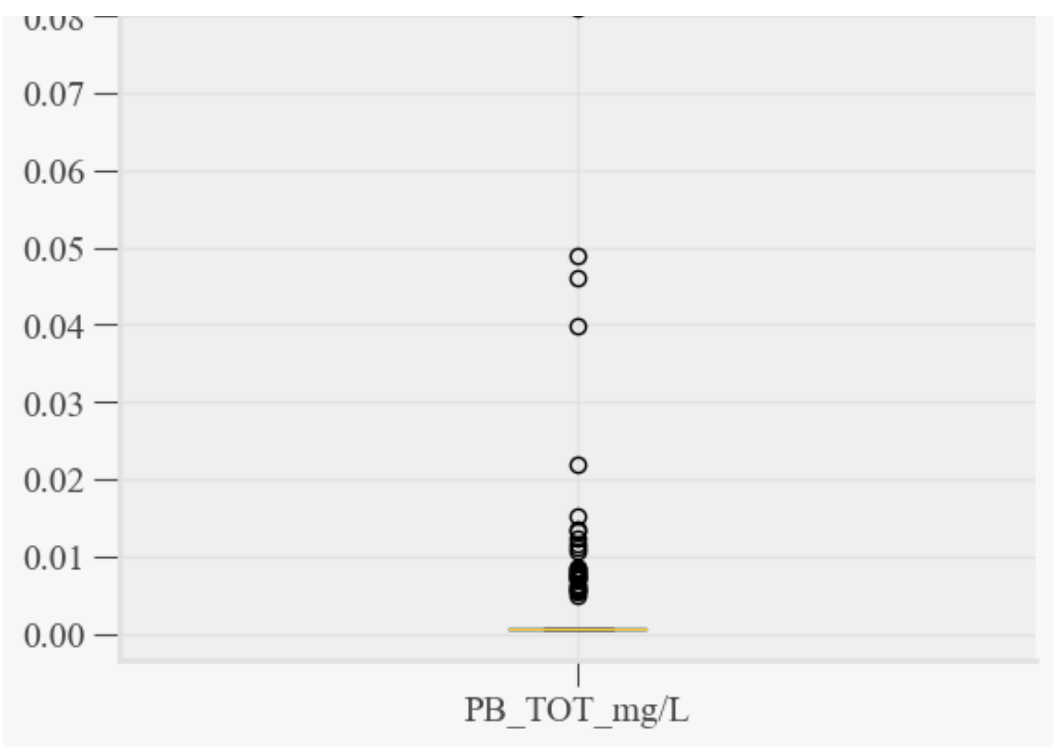

0.4
0.3
0.2
0.1
0.0
AS_TOT_mg/L

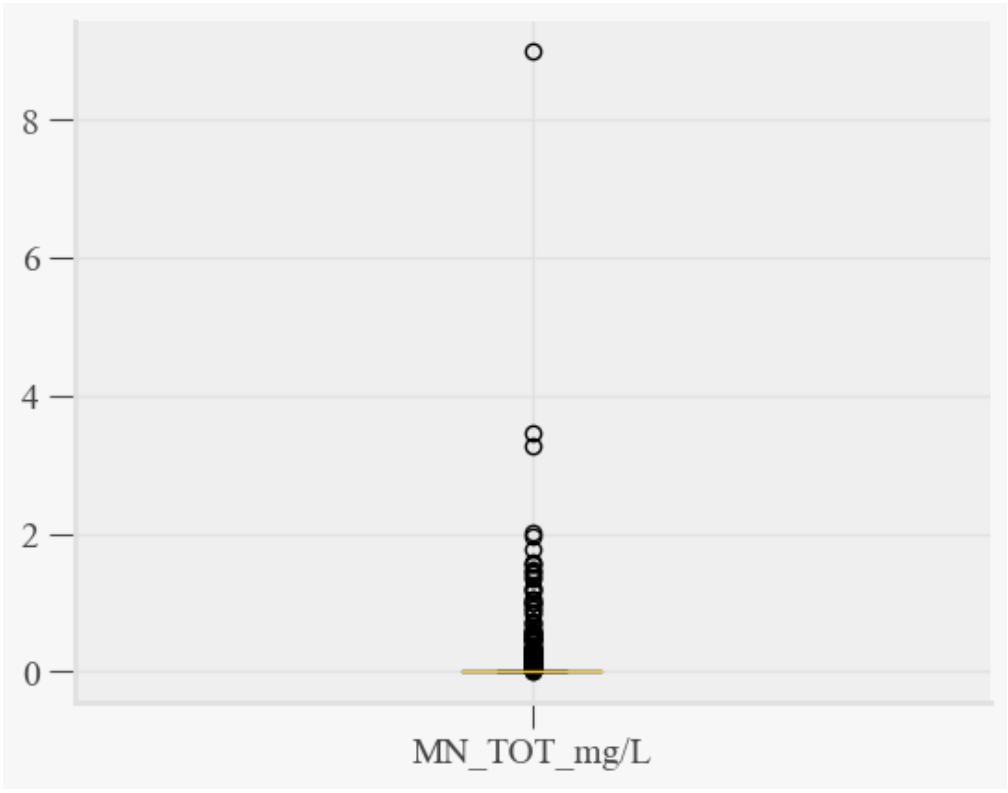

0.030
0.025

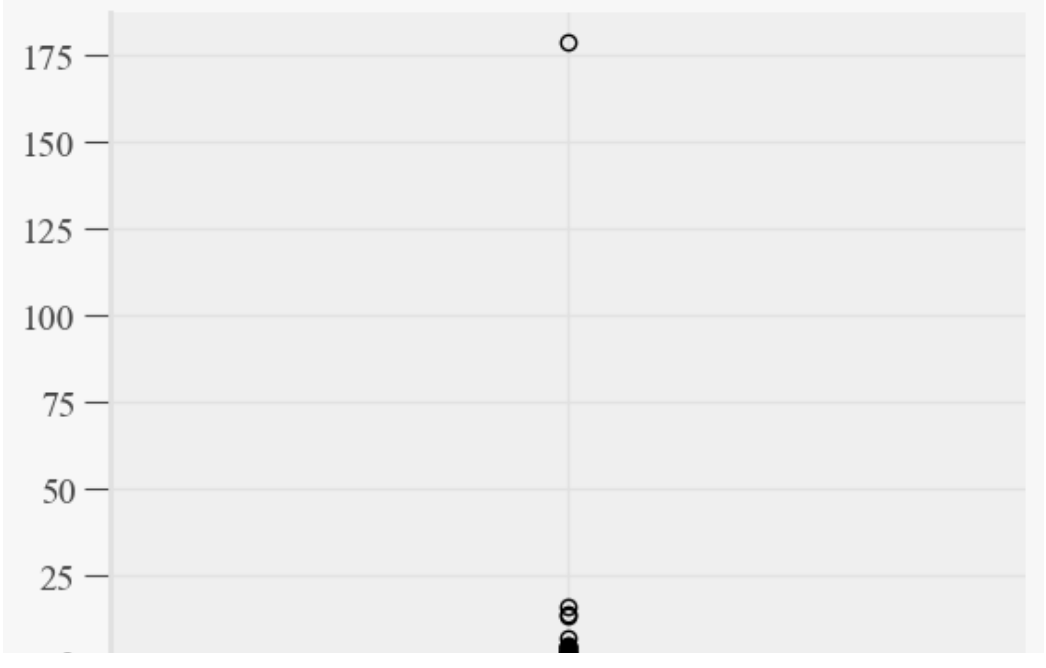

0.020
0.015
0.010
0.005
0.000
CD_TOT_mg/L


5
4
3
2
1
0
CR_TOT_mg/L


0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
HG_TOT_mg/L


0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
PB_TOT_mg/L


8
6
4
2
0
MN_TOT_mg/L


175
150
125
100
75
50
25

Parte 3: Análisis K-Means

In [782]:

```
import geopandas as gpd
import matplotlib.pyplot as plt
import pandas as pd

from shapely.geometry import Point

%matplotlib inline
# activate plot theme
import qeds
qeds.themes.mpl_style();

plt.rcParams['font.family'] = ['serif']
plt.rcParams['font.serif'] = ['Times New Roman']
```

In [783]:

```
mapDf = df[["SITIO", "LATITUD", "LONGITUD", "SEMAFORO"]]

mapDf = mapDf.rename({"SITIO":"name", "LATITUD": "latitude", "LONGITUD": "longitude", "SEMAFORO": "semaphore"}, axis=1)
mapDf
```

Out[783]:

	name	latitude	longitude	semaphore
0	POZO SAN GIL	22.20887	-102.02210	Verde
1	POZO R013 CAÑADA HONDA	21.99958	-102.20075	Verde
2	POZO COSIO	22.36685	-102.28801	Rojo
3	POZO EL SALITRILLO	22.18435	-102.29449	Verde
4	RANCHO EL TECOLOTE	23.45138	-110.24480	Rojo
...	...	...	...	...
1063	L-310 (COMUNIDAD SAN MANUEL)	24.76036	-99.54191	Rojo
1064	L-305 (EJIDO OJO DE AGUA LAS CRUCESITAS)	24.78280	-99.70099	Rojo
1065	HACIENDA MEXIQUITO POZO 01	25.55197	-99.82249	Rojo
1066	COMUNIDAD LOS POCITOS	24.80118	-100.32683	Verde
1067	COMUNIDAD LA REFORMA	25.09380	-100.73302	Verde

1068 rows × 4 columns

In [784]:

```
mapDf["Coordinates"] = list(zip(mapDf.longitude, mapDf.latitude))
mapDf["Coordinates"] = mapDf["Coordinates"].apply(Point)
mapDf.head()
```

Out[784]:

	name	latitude	longitude	semaphore	Coordinates
0	POZO SAN GIL	22.20887	-102.02210	Verde	POINT (-102.0221 22.20887)
1	POZO R013 CAÑADA HONDA	21.99958	-102.20075	Verde	POINT (-102.20075 21.99958)

	name	latitude	longitude	semaphore	Coordinates
2	POZO COSIO	22.36685	-102.28801	Rojo	POINT (-102.28801 22.36685)
3	POZO EL SALITRILLO	22.18435	-102.29449	Verde	POINT (-102.29449 22.18435)
4	RANCHO EL TECOLOTE	23.45138	-110.24480	Rojo	POINT (-110.2448 23.45138)

In [785]:

```
gdf = gpd.GeoDataFrame(mapDf, geometry="Coordinates")
gdf.head()
```

Out[785]:

	name	latitude	longitude	semaphore	Coordinates
0	POZO SAN GIL	22.20887	-102.02210	Verde	POINT (-102.02210 22.20887)
1	POZO R013 CAÑADA HONDA	21.99958	-102.20075	Verde	POINT (-102.20075 21.99958)
2	POZO COSIO	22.36685	-102.28801	Rojo	POINT (-102.28801 22.36685)
3	POZO EL SALITRILLO	22.18435	-102.29449	Verde	POINT (-102.29449 22.18435)
4	RANCHO EL TECOLOTE	23.45138	-110.24480	Rojo	POINT (-110.24480 23.45138)

In [786]:

```
mex = gpd.read_file(gpd.datasets.get_path("naturalearth_lowres"))
mex = mex.set_index("iso_a3")
```

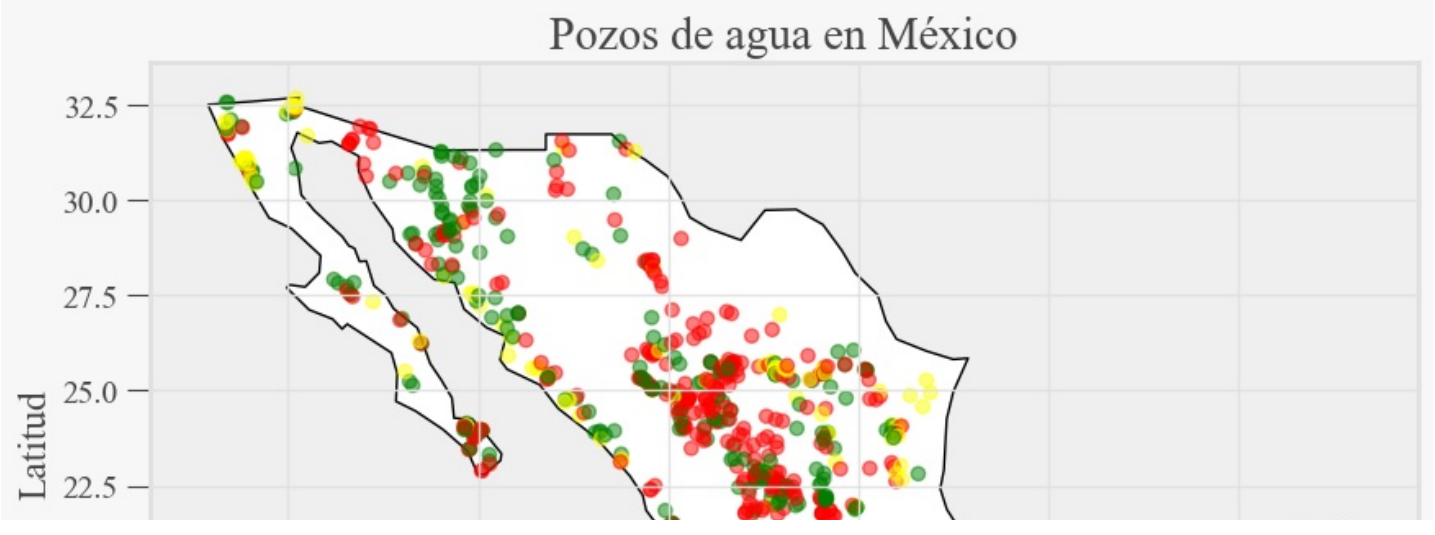
In [787]:

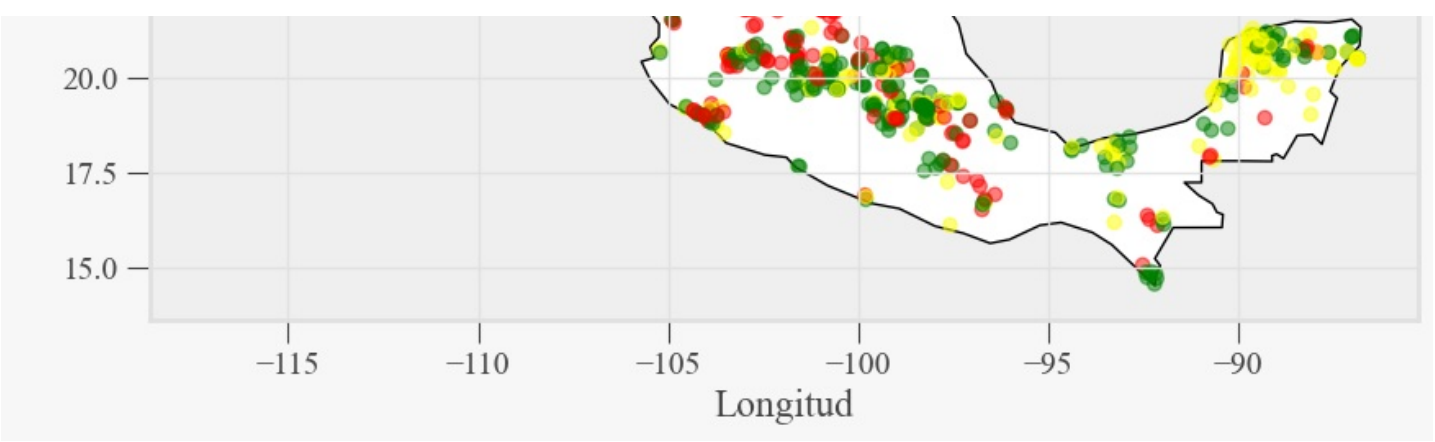
```
gdf["semaphore"] = replaceSpecificValue(gdf, "semaphore", "Verde", "green")
gdf["semaphore"] = replaceSpecificValue(gdf, "semaphore", "Rojo", "red")
gdf["semaphore"] = replaceSpecificValue(gdf, "semaphore", "Amarillo", "yellow")
```

```
[INFO] Number of 'Verde' in semaphore being replaced for 'green': 434
[INFO] Number of 'Rojo' in semaphore being replaced for 'red': 387
[INFO] Number of 'Amarillo' in semaphore being replaced for 'yellow': 247
```

In [788]:

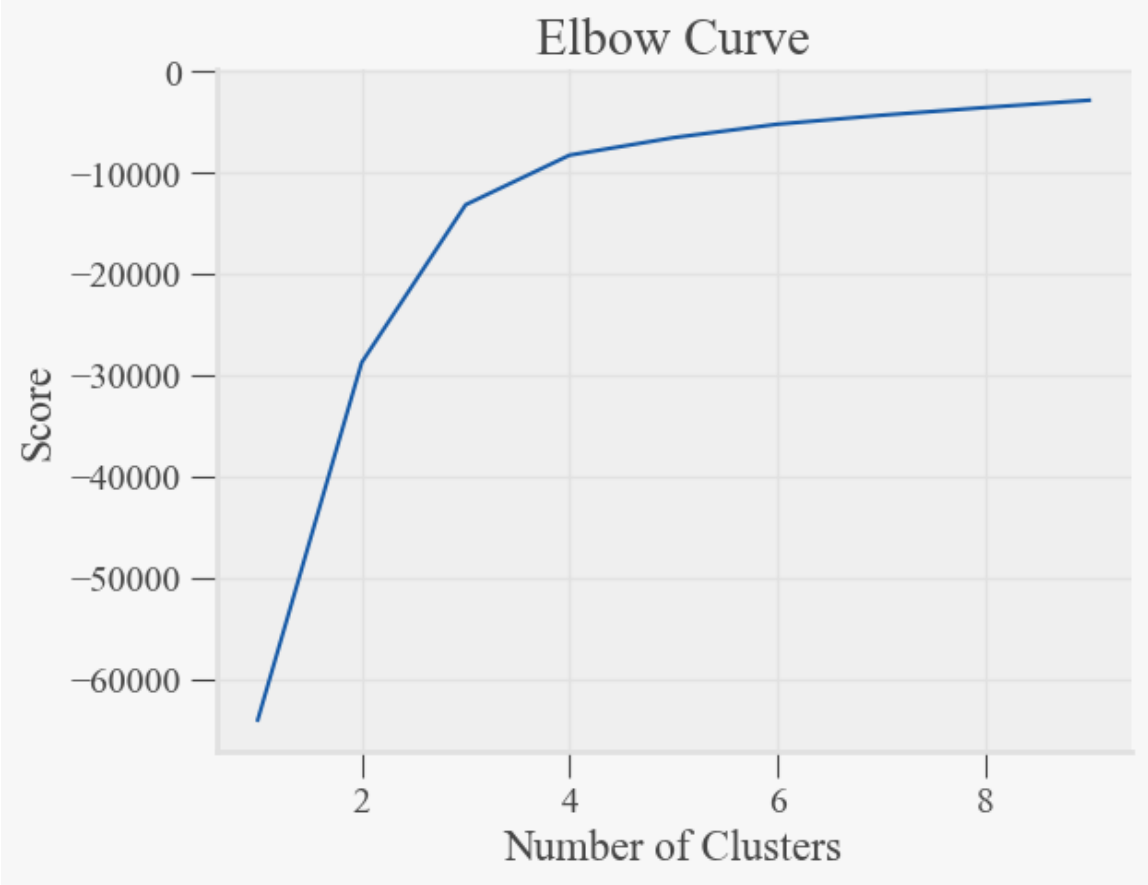
```
fig, gax = plt.subplots(figsize=(10,10))

mex.query("name == 'Mexico'").plot(ax = gax, edgecolor='black', color='white')

gdf.plot(ax=gax, color=gdf["semaphore"], alpha = 0.5)

gax.set_xlabel('Longitud')
gax.set_ylabel('Latitud')
gax.set_title('Pozos de agua en México')

gax.spines['top'].set_visible(True)
gax.spines['right'].set_visible(True)

plt.show()
```





In [789]:

```
# Lets start the kmeans.

from sklearn.cluster import KMeans

# First lets find the number of warehouses using the elbow method:

latlong = pd.DataFrame(mapDf[["latitude", "longitude"]])

Nc = range(1, 10)
kmeans = [KMeans(n_clusters=i) for i in Nc]
kmeans
score = [kmeans[i].fit(latlong).score(latlong) for i in range(len(kmeans))]
score
plt.plot(Nc,score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```



In [790]:

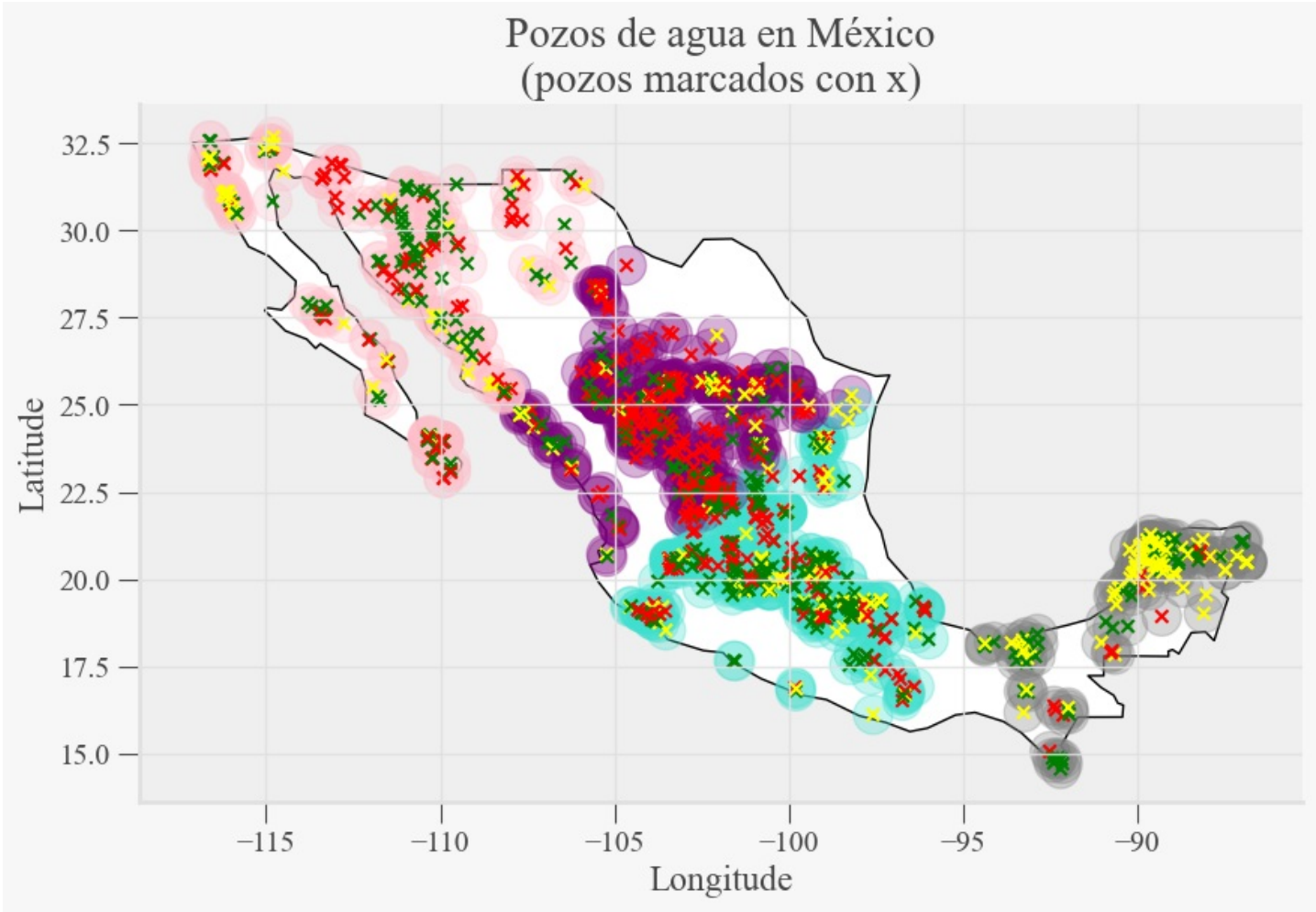
```
# Let's go with 4.

k = 4
```

```
kmeans = KMeans(n_clusters=k).fit(latlong)
centroids = kmeans.cluster_centers_

print("Centroids:")
print(centroids)
```

```
Centroids:
[[  20.14821379 -100.16403027]
 [  28.62571787 -111.31264726]
 [  19.6502625   -90.09271578]
 [  24.64046098 -103.45756561]]
```

In [791]:

```
# Plot again the same map but using the kmeans clusters (indicated by color).
fig, gax = plt.subplots(figsize=(10,10))
mex.query("name == 'Mexico'").plot(ax = gax, edgecolor='black', color='white')

gax.set_xlabel('Longitude')
gax.set_ylabel('Latitude')
gax.set_title('Pozos de agua en México\n(pozos marcados con x)')


gax.spines['top'].set_visible(False)
gax.spines['right'].set_visible(False)

colorMap = {0: "turquoise", 1: "pink", 2: "grey", 3: "purple"}
colors = [colorMap[i] for i in kmeans.labels_.astype(float)]
plt.scatter(latlong['longitude'], latlong['latitude'], c=colors, alpha=0.3, s=350)

gdf.plot(ax=gax, color=gdf["semaphore"], alpha = 1, marker="x")

plt.show()
```



Parece ser que existe una correlación entre la calidad del agua y su ubicación geográfica.

- **Se puede apreciar que en el sureste de México prevalecen más las aguas con semáforo amarillo .**
- **El centro superior de la república la gran mayoría de los pozos tienen un semáforo rojo .**

- **El centro inferior de la república y la península de Baja California (y una parte del norte) tienen una combinación de semáforos verdes con amarillo (con un procentaje más bajo de semáforos rojos).**