GEOVIS - Process Weather Data

Consolidate **Weather** data calculating monthly averages for each weather variable, for each county in TX and NY

0. Libraries and Variables

In [1]:
```
#  variables

import arcpy
import arcgis
import os
import numpy as np
import pandas as pd

from arcgis import GIS
from arcgis.features import GeoAccessor, GeoSeriesAccessor
```

In [2]:
```
_gis = GIS('Pro')

_inGdb = r'C:\Users\Ameri\OneDrive - The University of Texas at Dallas\UT Dalla

_outFolder = r'C:\Users\Ameri\OneDrive - The University of Texas at Dallas\UT D
_outGdb = 'Weather2.gdb'
_outGdb = os.path.join(_outFolder, _outGdb)

l_WeatherTypes = ['cdd', 'hdd', 'pcp', 'pdsi', 'phdi', 'pmdi', 'tavg', 'zndx']

print(_inGdb, os.path.exists(_inGdb))
print(_outFolder, os.path.exists(_outFolder))
```

```
C:\Users\Ameri\OneDrive - The University of Texas at Dallas\UT Dallas\2024 03
Fall\EPPS 6356 Data Visualization\GeoVis\RawData\Weather\monthly_1985to2023_c
ntyFIPS_detailed.gdb True
C:\Users\Ameri\OneDrive - The University of Texas at Dallas\UT Dallas\2024 03
Fall\EPPS 6356 Data Visualization\GeoVis\DataProcess True
```

In [3]:
```
#  Verify all weather feature classes exist

for tag in l_WeatherTypes:
    fc = f'{tag}_monthly_1985to2023_cntyFIPS'
    print(fc, arcpy.Exists(os.path.join(_inGdb, fc)))
```

```
cdd_monthly_1985to2023_cntyFIPS True
hdd_monthly_1985to2023_cntyFIPS True
pcp_monthly_1985to2023_cntyFIPS True
pdsi_monthly_1985to2023_cntyFIPS True
phdi_monthly_1985to2023_cntyFIPS True
pmdi_monthly_1985to2023_cntyFIPS True
tavg_monthly_1985to2023_cntyFIPS True
zndx_monthly_1985to2023_cntyFIPS True
```

1. Filter all weather feature classes in Texas & New York

In [4]:
```
#  Filter all weather feature classes

d_DFs_Unprocessed = {}  #  dictionary of weather types dataframes

#  loop all weather types,
#     filter NY & TX data,
#     save filtered dataframe in dictionary

for tag in l_WeatherTypes:
    #  get feature class name
    fc = f'{tag}_monthly_1985to2023_cntyFIPS'
    print(f"~   ... filtering feature class --> {fc}")
    fc = os.path.join(_inGdb, fc)

    #  Read feature class
    sdf = pd.DataFrame.spatial.from_featureclass(fc)

    #  filter the data  # NY & TX
    filtered_df = sdf[sdf['STATEFP'].isin(['36', '48'])].copy()

    #  save filtered data in dictionary d_DFs for later processing
    d_DFs_Unprocessed[tag] = filtered_df
```

```
~   ... filtering feature class --> cdd_monthly_1985to2023_cntyFIPS
~   ... filtering feature class --> hdd_monthly_1985to2023_cntyFIPS
~   ... filtering feature class --> pcp_monthly_1985to2023_cntyFIPS
~   ... filtering feature class --> pdsi_monthly_1985to2023_cntyFIPS
~   ... filtering feature class --> phdi_monthly_1985to2023_cntyFIPS
~   ... filtering feature class --> pmdi_monthly_1985to2023_cntyFIPS
~   ... filtering feature class --> tavg_monthly_1985to2023_cntyFIPS
~   ... filtering feature class --> zndx_monthly_1985to2023_cntyFIPS
```

2. Calculate monthly averages

In [5]:
```
#  create dictionary for months that will be used to process data

d_month = {}
d_month['01'] = 'Jan'
d_month['02'] = 'Feb'
d_month['03'] = 'Mar'
d_month['04'] = 'Apr'
d_month['05'] = 'May'
d_month['06'] = 'Jun'
d_month['07'] = 'Jul'
d_month['08'] = 'Aug'
d_month['09'] = 'Sep'
d_month['10'] = 'Oct'
d_month['11'] = 'Nov'
d_month['12'] = 'Dec'

l_months = sorted(list(d_month.keys()))

print(d_month)
print(l_months)
```

```
{'01': 'Jan', '02': 'Feb', '03': 'Mar', '04': 'Apr', '05': 'May', '06': 'Ju
n', '07': 'Jul', '08': 'Aug', '09': 'Sep', '10': 'Oct', '11': 'Nov', '12': 'D
ec'}
['01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12']
```

In [7]:
```
#  function to calculate monthly averages in dataframe
def ProcessWeatherDF(df, weatherType):

    sdf = df.copy()  # make a copy to preserve original df

    l_columns = list(sdf.columns)
    l_colsProcessed = []

    #  calculate monthly averages
    for m in l_months:

        #  get columns ending in m (month)
        l_cols2Process = [x for x in l_columns if x.endswith(f'_{m}')]

        #  add cols2process to list of processed cols
        l_colsProcessed.extend(l_cols2Process)

        #  create new monthly avg column
        #colName = f'{weatherType}_avg_{m}_{d_month[m]}'
        colName = f'w_{weatherType}_{d_month[m]}'

        #  calculate mean value in monthly column
        print(f"~      ... Processing column: {colName}")
        sdf[colName] = sdf[l_cols2Process].mean(axis=1)

    #  get updated columns list
    l_columns = list(sdf.columns)

    #  columns excluding processed ones
    l_newColumns = [x for x in l_columns if x not in l_colsProcessed]
    sdf = sdf[l_newColumns].copy()

    #  return updated sdf
    return sdf
```

In [8]:
```
#  process all weather dataframes

d_DFs_Processed = {}  #  dictionary of processed weather dataframes

for tag in l_WeatherTypes:

    #  get dataframe to process from dict
    df = d_DFs_Unprocessed[tag]

    print(f"~   ... calculating monthly averages for weather variable --> {tag}
    sdf = ProcessWeatherDF(df, tag)

    # add processed df to processed dict
    d_DFs_Processed[tag] = sdf
```

```
~   ... calculating monthly averages for weather variable --> cdd
~      ... Processing column: w_cdd_Jan
~      ... Processing column: w_cdd_Feb
~      ... Processing column: w_cdd_Mar
~      ... Processing column: w_cdd_Apr
~      ... Processing column: w_cdd_May
~      ... Processing column: w_cdd_Jun
~      ... Processing column: w_cdd_Jul
~      ... Processing column: w_cdd_Aug
~      ... Processing column: w_cdd_Sep
~      ... Processing column: w_cdd_Oct
~      ... Processing column: w_cdd_Nov
~      ... Processing column: w_cdd_Dec
~   ... calculating monthly averages for weather variable --> hdd
~      ... Processing column: w_hdd_Jan
~      ... Processing column: w_hdd_Feb
~      ... Processing column: w_hdd_Mar
~      ... Processing column: w_hdd_Apr
~      ... Processing column: w_hdd_May
~      ... Processing column: w_hdd_Jun
~      ... Processing column: w_hdd_Jul
~      ... Processing column: w_hdd_Aug
~      ... Processing column: w_hdd_Sep
~      ... Processing column: w_hdd_Oct
~      ... Processing column: w_hdd_Nov
~      ... Processing column: w_hdd_Dec
~   ... calculating monthly averages for weather variable --> pcp
~      ... Processing column: w_pcp_Jan
~      ... Processing column: w_pcp_Feb
~      ... Processing column: w_pcp_Mar
~      ... Processing column: w_pcp_Apr
~      ... Processing column: w_pcp_May
~      ... Processing column: w_pcp_Jun
~      ... Processing column: w_pcp_Jul
~      ... Processing column: w_pcp_Aug
~      ... Processing column: w_pcp_Sep
~      ... Processing column: w_pcp_Oct
~      ... Processing column: w_pcp_Nov
~      ... Processing column: w_pcp_Dec
~   ... calculating monthly averages for weather variable --> pdsi
~      ... Processing column: w_pdsi_Jan
~      ... Processing column: w_pdsi_Feb
~      ... Processing column: w_pdsi_Mar
~      ... Processing column: w_pdsi_Apr
~      ... Processing column: w_pdsi_May
~      ... Processing column: w_pdsi_Jun
~      ... Processing column: w_pdsi_Jul
~      ... Processing column: w_pdsi_Aug
~      ... Processing column: w_pdsi_Sep
~      ... Processing column: w_pdsi_Oct
~      ... Processing column: w_pdsi_Nov
~      ... Processing column: w_pdsi_Dec
~   ... calculating monthly averages for weather variable --> phdi
~      ... Processing column: w_phdi_Jan
~      ... Processing column: w_phdi_Feb
~      ... Processing column: w_phdi_Mar
~      ... Processing column: w_phdi_Apr
```

```
~      ... Processing column: w_phdi_May
~      ... Processing column: w_phdi_Jun
~      ... Processing column: w_phdi_Jul
~      ... Processing column: w_phdi_Aug
~      ... Processing column: w_phdi_Sep
~      ... Processing column: w_phdi_Oct
~      ... Processing column: w_phdi_Nov
~      ... Processing column: w_phdi_Dec
~   ... calculating monthly averages for weather variable --> pmdi
~      ... Processing column: w_pmdi_Jan
~      ... Processing column: w_pmdi_Feb
~      ... Processing column: w_pmdi_Mar
~      ... Processing column: w_pmdi_Apr
~      ... Processing column: w_pmdi_May
~      ... Processing column: w_pmdi_Jun
~      ... Processing column: w_pmdi_Jul
~      ... Processing column: w_pmdi_Aug
~      ... Processing column: w_pmdi_Sep
~      ... Processing column: w_pmdi_Oct
~      ... Processing column: w_pmdi_Nov
~      ... Processing column: w_pmdi_Dec
~   ... calculating monthly averages for weather variable --> tavg
~      ... Processing column: w_tavg_Jan
~      ... Processing column: w_tavg_Feb
~      ... Processing column: w_tavg_Mar
~      ... Processing column: w_tavg_Apr
~      ... Processing column: w_tavg_May
~      ... Processing column: w_tavg_Jun
~      ... Processing column: w_tavg_Jul
~      ... Processing column: w_tavg_Aug
~      ... Processing column: w_tavg_Sep
~      ... Processing column: w_tavg_Oct
~      ... Processing column: w_tavg_Nov
~      ... Processing column: w_tavg_Dec
~   ... calculating monthly averages for weather variable --> zndx
~      ... Processing column: w_zndx_Jan
~      ... Processing column: w_zndx_Feb
~      ... Processing column: w_zndx_Mar
~      ... Processing column: w_zndx_Apr
~      ... Processing column: w_zndx_May
~      ... Processing column: w_zndx_Jun
~      ... Processing column: w_zndx_Jul
~      ... Processing column: w_zndx_Aug
~      ... Processing column: w_zndx_Sep
~      ... Processing column: w_zndx_Oct
~      ... Processing column: w_zndx_Nov
~      ... Processing column: w_zndx_Dec
```

3. Consolidate Weather Variables Monthly Averages into a single Dataset

In [12]:
```
#  merge processed columns into a single dataframe

df_Final = None  # this will hold the final dataframe

for tag in l_WeatherTypes:

    #  get processed dataframe
    df = d_DFs_Processed[tag]

    if df_Final is None:
        df_Final = df.copy() # initialize with first dataframe
    else:
        # append dataframe df to df_Final

        #  get only processed columns
        l_columns = list(df.columns)
        #l_cols_desired = [x for x in l_columns if x.startswith(f'{tag}_avg_')]
        l_cols_desired = [x for x in l_columns if x.startswith(f'w_{tag}_')]
        l_cols_desired.insert(0, 'GEOID') # keep id for merging purposes

        #  merge to df final
        df = df[l_cols_desired].copy()
        df_Final = pd.merge(df_Final, df, on='GEOID')


df_Final.head()
```

Out[12]:

w_tavg_Mar	w_tavg_Apr	w_tavg_May	w_tavg_Jun	w_tavg_Jul	w_tavg_Aug	w_tavg_Sep	w_tavg_(
57.587179	64.979487	73.025641	79.200000	81.674359	81.489744	74.869231	65.623(
50.923077	58.671795	67.856410	76.461538	79.071795	77.671795	70.669231	60.061!
50.443590	58.456410	67.694872	76.494872	80.289744	78.887179	71.541026	60.192(
64.589744	70.615385	77.353846	82.305128	83.935897	84.633333	80.782051	73.384(
55.448718	63.192308	71.882051	79.879487	84.297436	84.107692	76.100000	64.917!

In [13]:
```
# Create filegdb
if (not os.path.exists(_outGdb)):
    _outGdb = 'Weather2.gdb'
    arcpy.management.CreateFileGDB(_outFolder, _outGdb)
    _outGdb = os.path.join(_outFolder, _outGdb)

print(_outGdb, os.path.exists(_outGdb))
```

```
C:\Users\Ameri\OneDrive - The University of Texas at Dallas\UT Dallas\2024 03 Fall\EPPS 6356 Data Visualization\GeoVis\DataProcess\Weather2.gdb True
```

In [14]:
```
#  convert to spatial dataframe

sdf_Final = GeoAccessor.from_df(df_Final, geometry_column='SHAPE')
sdf_Final.head()
```

Out[14]:

	OBJECTID	STATEFP	COUNTYFP	COUNTYNS	GEOID	NAME	NAMELSAD	LSAD	CLASSI
0	8	48	327	01383949	48327	Menard	Menard County	06	
1	12	48	189	01383880	48189	Hale	Hale County	06	
2	14	48	011	01383791	48011	Armstrong	Armstrong County	06	
3	36	48	057	01383814	48057	Calhoun	Calhoun County	06	
4	39	48	077	01383824	48077	Clay	Clay County	06	

5 rows × 115 columns

In [15]:
```
#  save consolidated data to feature class
outWeatherFC = 'Weather_NY_TX_v2'
outWeatherFC = os.path.join(_outGdb, outWeatherFC)

sdf_Final.spatial.to_featureclass(outWeatherFC)
```

Out[15]:
```
'C:\\Users\\Ameri\\OneDrive - The University of Texas at Dallas\\UT Dallas\\2024 03 Fall\\EPPS 6356 Data Visualization\\GeoVis\\DataProcess\\Weather2.gdb\\Weather_NY_TX_v2'
```

In [ ]: