sea

October 29, 2024

1 1. Importing/Cleaning Data

[ ]:
```
import pandas as pd
slrdata = pd.read_csv('/content/SLR_TF U.S. Sea Level Projections.csv')
slrdata.columns = slrdata.columns.str.lower().str.replace(' ', '').str.
 ↪replace(r'[^\w]', '', regex=True)
slrdata = slrdata[slrdata['regionalclassification'] == 'Northeast']
#only selecting the 50th percentile results per slr scenario
slrdata = slrdata[slrdata['scenario'].str.contains('MED')]
#adding the offset to each projection
rsl_columns = [
    'rsl2005cm', 'rsl2020cm', 'rsl2030cm',
       'rsl2040cm', 'rsl2050cm', 'rsl2060cm', 'rsl2070cm', 'rsl2080cm',
       'rsl2090cm', 'rsl2100cm', 'rsl2110cm', 'rsl2120cm', 'rsl2130cm',
       'rsl2140cm', 'rsl2150cm'
]

# Add the offset value to each RSL column
for col in rsl_columns:
    slrdata[col] += slrdata['offset2000to2005cm']

# Remove columns
columns_to_remove = ['rsl2005cm', 'rsl2020cm']
slrdata.drop(columns=columns_to_remove, inplace=True)

# Function to convert to meters
def convert_rsl_to_meters(dataframe):
    rsl_columns = [col for col in dataframe.columns if 'rsl' in col]
    for col in rsl_columns:
        dataframe[col] = dataframe[col] / 100  # Convert cm to meters
    return dataframe

# Convert the RSL columns
slrdata = convert_rsl_to_meters(slrdata)
```

```
<ipython-input-35-01aece46fd61>:2: DtypeWarning: Columns (3) have mixed types.
Specify dtype option on import or set low_memory=False.
```

```
  slrdata = pd.read_csv('/content/SLR_TF U.S. Sea Level Projections.csv')
<ipython-input-35-01aece46fd61>:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  slrdata[col] += slrdata['offset2000to2005cm']
<ipython-input-35-01aece46fd61>:21: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  slrdata.drop(columns=columns_to_remove, inplace=True)
<ipython-input-35-01aece46fd61>:27: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  dataframe[col] = dataframe[col] / 100  # Convert cm to meters
```

[ ]:
```
slrdata
```

[ ]:

	psmslsite	psmslid	noaaid	noaaname	rslgridnum	\
391	EASTPORT	3.320000e+02	8410140.0	Eastport	483.53	
397	EASTPORT	3.320000e+02	8410140.0	Eastport	483.53	
403	EASTPORT	3.320000e+02	8410140.0	Eastport	483.53	
409	EASTPORT	3.320000e+02	8410140.0	Eastport	483.53	
415	EASTPORT	3.320000e+02	8410140.0	Eastport	483.53	
…	…	…	…	…	…	
19201	grid_37.0_286.0	1.005303e+09	NaN	NaN	454.66	
19207	grid_37.0_286.0	1.005303e+09	NaN	NaN	454.66	
19213	grid_37.0_286.0	1.005303e+09	NaN	NaN	454.66	
19219	grid_37.0_286.0	1.005303e+09	NaN	NaN	454.66	
19225	grid_37.0_286.0	1.005303e+09	NaN	NaN	454.66	

	lat	long	regionalclassification	uscoastlineintersect1yes	\
391	44.904598	-66.982903	Northeast	1.0	
397	44.904598	-66.982903	Northeast	1.0	
403	44.904598	-66.982903	Northeast	1.0	
409	44.904598	-66.982903	Northeast	1.0	
415	44.904598	-66.982903	Northeast	1.0	
…	…	…	…	…	
19201	37.000000	-74.000000	Northeast	1.0	
19207	37.000000	-74.000000	Northeast	1.0	
19213	37.000000	-74.000000	Northeast	1.0	

```
19219  37.000000 -74.000000              Northeast                      1.0
19225  37.000000 -74.000000              Northeast                      1.0

        scenario  …  rsl2060cm  rsl2070cm  rsl2080cm  rsl2090cm  rsl2100cm  \
391    0.3 - MED  …       0.33       0.36       0.39       0.42       0.45
397    0.5 - MED  …       0.40       0.46       0.52       0.58       0.63
403    1.0 - MED  …       0.46       0.58       0.72       0.89       1.08
409    1.5 - MED  …       0.55       0.73       0.93       1.15       1.39
415    2.0 - MED  …       0.64       0.88       1.15       1.46       1.79
…            …  …          …          …          …          …          …
19201  0.3 - MED  …       0.47       0.53       0.57       0.62       0.68
19207  0.5 - MED  …       0.52       0.61       0.70       0.78       0.87
19213  1.0 - MED  …       0.59       0.72       0.89       1.08       1.31
19219  1.5 - MED  …       0.70       0.91       1.15       1.42       1.70
19225  2.0 - MED  …       0.81       1.10       1.44       1.81       2.19

       rsl2110cm  rsl2120cm  rsl2130cm  rsl2140cm  rsl2150cm
391         0.48       0.52       0.55       0.58       0.61
397         0.70       0.78       0.85       0.92       0.99
403         1.30       1.50       1.68       1.84       2.00
409         1.62       1.81       1.98       2.14       2.31
415         2.14       2.47       2.78       3.03       3.24
…              …          …          …          …          …
19201       0.75       0.80       0.86       0.92       0.97
19207       0.97       1.07       1.17       1.27       1.37
19213       1.55       1.78       2.00       2.18       2.37
19219       1.99       2.24       2.44       2.64       2.84
19225       2.61       3.00       3.33       3.61       3.88

[635 rows x 26 columns]
```

2 2. Functions for Elevation and Local Sea Level Rise

[ ]:
```python
import requests
from geopy.distance import geodesic
import pandas as pd

# Your saved API key
api_key = 'AIzaSyDkKUEHOa5nPXq6xEqphlorpIZYSRAUCBE'

# Function to get elevation
def get_elevation(lat, lon, api_key):
    # Construct the request URL
    url = f"https://maps.googleapis.com/maps/api/elevation/json?
 ↪locations={lat},{lon}&key={api_key}"
```

```
    # Make the request
    response = requests.get(url)

    # Check if the request was successful
    if response.status_code == 200:
        data = response.json()
        # Extract the elevation from the response
        if 'results' in data and len(data['results']) > 0:
            elevation = data['results'][0]['elevation']
            return elevation
        else:
            print("No results found.")
            return None
    else:
        print(f"Error: {response.status_code}")
        return None

# Function to find the closest locations
def find_closest_scenario(lat, long, dataframe):
    # Calculate distance for each row
    dataframe['distance'] = dataframe.apply(
        lambda row: geodesic((lat, long), (row['lat'], row['long'])).meters,
        axis=1
    )

    # Find the minimum distance
    min_distance = dataframe['distance'].min()

    # Get all rows with the minimum distance
    closest_rows = dataframe[dataframe['distance'] == min_distance]

    # Extract the scenarios and sea level rise data
    scenarios = []
    for _, closest_row in closest_rows.iterrows():
        scenario = closest_row['scenario']
        sea_level_rise = {col: closest_row[col] for col in dataframe.columns if␣
 ↪'rsl2' in col}
        scenarios.append((scenario, sea_level_rise))

    # Return all scenarios and their sea level rise data
    return scenarios

# Function of print results
def print_sea_level_comparison(latitude, longitude, elevation, scenarios):
    # Format the elevation to two decimal points
    formatted_elevation = round(elevation, 2)
```

```
    print(f"Elevation at location ({latitude}, {longitude}):␣
↪{formatted_elevation} meters\n")

    # Iterate through each scenario and sea level rise data
    for scenario, sea_level_rise in scenarios:
        print(f"Scenario: {scenario}")
        for year, rise in sea_level_rise.items():
            # Extract only the year from the key (e.g., 'rsl2150cm' -> '2150')
            year_only = ''.join(filter(str.isdigit, year))
            rise_formatted = round(rise, 2)
            if formatted_elevation < rise_formatted:
                print(f"At year {year_only}, sea level rise ({rise_formatted}␣
↪cm) exceeds elevation ({formatted_elevation} meters).")
            else:
                print(f"At year {year_only}, elevation ({formatted_elevation}␣
↪meters) exceeds sea level rise ({rise_formatted} cm).")
        print()


# Call the function with example data
# print_sea_level_comparison(latitude, longitude, elevation, closest_scenarios)
```

```
Elevation at location (43.670618, -70.241456): 1.62 meters

Scenario: 0.3 - MED
At year 2030, elevation (1.62 meters) exceeds sea level rise (0.16 cm).
At year 2040, elevation (1.62 meters) exceeds sea level rise (0.21 cm).
At year 2050, elevation (1.62 meters) exceeds sea level rise (0.27 cm).
At year 2060, elevation (1.62 meters) exceeds sea level rise (0.31 cm).
At year 2070, elevation (1.62 meters) exceeds sea level rise (0.35 cm).
At year 2080, elevation (1.62 meters) exceeds sea level rise (0.37 cm).
At year 2090, elevation (1.62 meters) exceeds sea level rise (0.4 cm).
At year 2100, elevation (1.62 meters) exceeds sea level rise (0.42 cm).
At year 2110, elevation (1.62 meters) exceeds sea level rise (0.46 cm).
At year 2120, elevation (1.62 meters) exceeds sea level rise (0.49 cm).
At year 2130, elevation (1.62 meters) exceeds sea level rise (0.52 cm).
At year 2140, elevation (1.62 meters) exceeds sea level rise (0.54 cm).
At year 2150, elevation (1.62 meters) exceeds sea level rise (0.57 cm).

Scenario: 0.5 - MED
At year 2030, elevation (1.62 meters) exceeds sea level rise (0.17 cm).
At year 2040, elevation (1.62 meters) exceeds sea level rise (0.24 cm).
At year 2050, elevation (1.62 meters) exceeds sea level rise (0.31 cm).
At year 2060, elevation (1.62 meters) exceeds sea level rise (0.38 cm).
At year 2070, elevation (1.62 meters) exceeds sea level rise (0.44 cm).
At year 2080, elevation (1.62 meters) exceeds sea level rise (0.5 cm).
At year 2090, elevation (1.62 meters) exceeds sea level rise (0.56 cm).
```

```
At year 2100, elevation (1.62 meters) exceeds sea level rise (0.61 cm).
At year 2110, elevation (1.62 meters) exceeds sea level rise (0.68 cm).
At year 2120, elevation (1.62 meters) exceeds sea level rise (0.75 cm).
At year 2130, elevation (1.62 meters) exceeds sea level rise (0.82 cm).
At year 2140, elevation (1.62 meters) exceeds sea level rise (0.89 cm).
At year 2150, elevation (1.62 meters) exceeds sea level rise (0.96 cm).

Scenario: 1.0 - MED
At year 2030, elevation (1.62 meters) exceeds sea level rise (0.18 cm).
At year 2040, elevation (1.62 meters) exceeds sea level rise (0.26 cm).
At year 2050, elevation (1.62 meters) exceeds sea level rise (0.35 cm).
At year 2060, elevation (1.62 meters) exceeds sea level rise (0.45 cm).
At year 2070, elevation (1.62 meters) exceeds sea level rise (0.57 cm).
At year 2080, elevation (1.62 meters) exceeds sea level rise (0.7 cm).
At year 2090, elevation (1.62 meters) exceeds sea level rise (0.88 cm).
At year 2100, elevation (1.62 meters) exceeds sea level rise (1.07 cm).
At year 2110, elevation (1.62 meters) exceeds sea level rise (1.28 cm).
At year 2120, elevation (1.62 meters) exceeds sea level rise (1.48 cm).
At year 2130, sea level rise (1.66 cm) exceeds elevation (1.62 meters).
At year 2140, sea level rise (1.83 cm) exceeds elevation (1.62 meters).
At year 2150, sea level rise (1.99 cm) exceeds elevation (1.62 meters).

Scenario: 1.5 - MED
At year 2030, elevation (1.62 meters) exceeds sea level rise (0.19 cm).
At year 2040, elevation (1.62 meters) exceeds sea level rise (0.28 cm).
At year 2050, elevation (1.62 meters) exceeds sea level rise (0.4 cm).
At year 2060, elevation (1.62 meters) exceeds sea level rise (0.55 cm).
At year 2070, elevation (1.62 meters) exceeds sea level rise (0.72 cm).
At year 2080, elevation (1.62 meters) exceeds sea level rise (0.92 cm).
At year 2090, elevation (1.62 meters) exceeds sea level rise (1.16 cm).
At year 2100, elevation (1.62 meters) exceeds sea level rise (1.4 cm).
At year 2110, sea level rise (1.63 cm) exceeds elevation (1.62 meters).
At year 2120, sea level rise (1.83 cm) exceeds elevation (1.62 meters).
At year 2130, sea level rise (1.99 cm) exceeds elevation (1.62 meters).
At year 2140, sea level rise (2.16 cm) exceeds elevation (1.62 meters).
At year 2150, sea level rise (2.33 cm) exceeds elevation (1.62 meters).

Scenario: 2.0 - MED
At year 2030, elevation (1.62 meters) exceeds sea level rise (0.19 cm).
At year 2040, elevation (1.62 meters) exceeds sea level rise (0.29 cm).
At year 2050, elevation (1.62 meters) exceeds sea level rise (0.43 cm).
At year 2060, elevation (1.62 meters) exceeds sea level rise (0.63 cm).
At year 2070, elevation (1.62 meters) exceeds sea level rise (0.88 cm).
At year 2080, elevation (1.62 meters) exceeds sea level rise (1.17 cm).
At year 2090, elevation (1.62 meters) exceeds sea level rise (1.48 cm).
At year 2100, sea level rise (1.81 cm) exceeds elevation (1.62 meters).
At year 2110, sea level rise (2.15 cm) exceeds elevation (1.62 meters).
At year 2120, sea level rise (2.51 cm) exceeds elevation (1.62 meters).
```

```
At year 2130, sea level rise (2.81 cm) exceeds elevation (1.62 meters).
At year 2140, sea level rise (3.05 cm) exceeds elevation (1.62 meters).
At year 2150, sea level rise (3.26 cm) exceeds elevation (1.62 meters).
```

3 3. Maine Stations

[ ]:
```
#creating station information so we can pull the MHHW mark from NOAA via their␣
 ↪API
stations = pd.DataFrame({
    'city': ['Portland', 'Bar Harbor', 'Cutler Farris Wharf', 'Eastport'],
    'state': ['ME', 'ME', 'ME', 'ME'],
    'stationid': ['8418150', '8413320', '8411060', '8410140'],
    'latitude': ['43.657951','44.393106','44.656544','44.904939' ],
    'longitude': ['-70.244189','-68.203704','-67.210228','-66.981884']
})
```

4 4. Function to Find Closest Maine Station

[ ]:
```
from math import radians, sin, cos, sqrt, atan2

def find_closest_station(lat, lon, stations):
    R = 6371  # Earth's radius in kilometers

    def haversine(lat1, lon1, lat2, lon2):
        lat1, lon1, lat2, lon2 = map(radians, [lat1, lon1, lat2, lon2])
        dlat = lat2 - lat1
        dlon = lon2 - lon1
        a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
        c = 2 * atan2(sqrt(a), sqrt(1-a))
        return R * c

    stations['distance'] = stations.apply(
        lambda row: haversine(lat, lon, float(row['latitude']),␣
 ↪float(row['longitude'])),
        axis=1
    )

    closest_station = stations.loc[stations['distance'].idxmin()]
    return closest_station
```

5 5. Functions to Find MHHW for a Given Station

[ ]:
```
import requests
from math import radians, sin, cos, sqrt, atan2

# Function to retrieve MHHW and MSL values for a specific station
def get_mhhw_and_msl(station_id):
    base_url = "https://api.tidesandcurrents.noaa.gov/mdapi/prod/webapi/
 ↪stations/{}/datums.json?units=metric"

    # Make the request to the API
    response = requests.get(base_url.format(station_id))

    if response.status_code == 200:
        data = response.json()

        # Extract MHHW and MSL values
        mhhw = next((float(datum['value']) for datum in data['datums'] if␣
 ↪datum['name'] == 'MHHW'), None)
        msl = next((float(datum['value']) for datum in data['datums'] if␣
 ↪datum['name'] == 'MSL'), None)

        if mhhw is not None and msl is not None:
            return mhhw, msl
        else:
            if mhhw is None:
                print(f"MHHW not found for station {station_id}")
            if msl is None:
                print(f"MSL not found for station {station_id}")
            return None, None
    else:
        print(f"Error fetching data for station {station_id}: {response.
 ↪status_code}")
        return None, None

# Function to find the closest station, retrieve MHHW, MSL, and calculate␣
 ↪highest tide
def get_mhhw_for_location(lat, lon, stations):
    closest_station = find_closest_station(lat, lon, stations)
    station_id = closest_station['stationid']

    # Get MHHW and MSL
    mhhw, msl = get_mhhw_and_msl(station_id)

    if mhhw is not None and msl is not None:
        # Calculate the highest tide (MHHW - MSL)
        highest_tide = mhhw - msl
```

```
        return {
            'input_lat': lat,
            'input_lon': lon,
            'closest_station': closest_station['city'],
            'station_id': station_id,
            'station_lat': closest_station['latitude'],
            'station_lon': closest_station['longitude'],
            'mhhw_meters': mhhw,
            'msl_meters': msl,
            'highest_tide': highest_tide
        }
    else:
        return None

# Example usage
lat = 43.65  # Example latitude
lon = -70.25  # Example longitude

result = get_mhhw_for_location(lat, lon, stations)

if result:
    print(f"\nFor location ({lat}, {lon}):")
    print(f"Closest station: {result['closest_station']} (ID:␣
 ↪{result['station_id']})")
    print(f"Station location: ({result['station_lat']},␣
 ↪{result['station_lon']})")
    print(f"MHHW: {result['mhhw_meters']} meters")
    print(f"MSL: {result['msl_meters']} meters")
    print(f"Highest Tide (MHHW - MSL): {result['highest_tide']} meters")
else:
    print("Unable to fetch MHHW or MSL data.")
```

```
For location (43.65, -70.25):
Closest station: Portland (ID: 8418150)
Station location: (43.657951, -70.244189)
MHHW: 5.626 meters
MSL: 4.113 meters
Highest Tide (MHHW - MSL): 1.513 meters
```

6 Main Function

[ ]:
```
import requests
from geopy.distance import geodesic
import pandas as pd
```

```python
def main(lat, lon, api_key, stations, slrdata):
    # Step 1: Get Elevation
    elevation = get_elevation(lat, lon, api_key)

    # Step 2: Find Closest Scenario Station and Get Scenario Data
    scenarios = find_closest_scenario(lat, lon, slrdata)

    # Step 3: Find Closest Tide Station and Get MHHW & MSL
    closest_station = find_closest_station(lat, lon, stations)
    mhhw, msl = get_mhhw_and_msl(closest_station['stationid'])
    highest_tide = mhhw - msl if mhhw is not None and msl is not None else None

    # Step 4: Compile Results
    return {
        'latitude': lat,
        'longitude': lon,
        'elevation': elevation,
        'scenarios': scenarios,
        'highest_tide': highest_tide,
        'closest_station': closest_station['city'],
        'station_id': closest_station['stationid'],
        'station_lat': closest_station['latitude'],
        'station_lon': closest_station['longitude']
    }

result = main(43.66843, -70.24020, api_key, stations, slrdata)

# Function to print results comparing elevation to sea level rise + highest tide
def print_sea_level_and_tide_comparison(latitude, longitude, elevation,␣
 ↪highest_tide, scenarios):
    # Format the elevation and highest tide to two decimal points
    formatted_elevation = f"{elevation:.2f}"
    formatted_highest_tide = f"{highest_tide:.2f}"

    print(f"Elevation at location ({latitude}, {longitude}):␣
 ↪{formatted_elevation} meters")
    print(f"Highest Tide (MHHW - MSL): {formatted_highest_tide} meters\n")

    # Iterate through each scenario and sea level rise data
    for scenario, sea_level_rise in scenarios:
        print(f"Scenario: {scenario}")
        for year, rise in sea_level_rise.items():
            # Extract only the year from the key (e.g., 'rsl2150cm' -> '2150')
            year_only = ''.join(filter(str.isdigit, year))
            rise_formatted = f"{rise:.2f}"
```

```
            # Calculate total sea level rise including highest tide, formatted␣
 ↪to two decimals
            total_rise_with_tide = f"{(float(rise_formatted) +␣
 ↪float(formatted_highest_tide)):.2f}"

            # Print detailed information
            print(f"Year {year_only}:")
            print(f" - Sea Level Rise: {rise_formatted} meters")
            print(f" - Highest Tide: {formatted_highest_tide} meters")
            print(f" - Sea Level Rise + Highest Tide: {total_rise_with_tide}␣
 ↪meters")

            # Compare the elevation to the total rise
            if float(formatted_elevation) < float(total_rise_with_tide):
                print(f"   -> Sea level rise + highest tide␣
 ↪({total_rise_with_tide} meters) exceeds elevation ({formatted_elevation}␣
 ↪meters).")
            else:
                print(f"   -> Elevation ({formatted_elevation} meters) exceeds␣
 ↪sea level rise + highest tide ({total_rise_with_tide} meters).")

        print()  # Blank line between scenarios


# Example usage
print_sea_level_and_tide_comparison(result['latitude'], result['longitude'],␣
 ↪result['elevation'], result['highest_tide'], result['scenarios'])
```

```
Elevation at location (43.66843, -70.2402): 0.27 meters
Highest Tide (MHHW - MSL): 1.51 meters

Scenario: 0.3 - MED
Year 2030:
 - Sea Level Rise: 0.16 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.67 meters
   -> Sea level rise + highest tide (1.67 meters) exceeds elevation (0.27
meters).
Year 2040:
 - Sea Level Rise: 0.21 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.72 meters
   -> Sea level rise + highest tide (1.72 meters) exceeds elevation (0.27
meters).
Year 2050:
 - Sea Level Rise: 0.27 meters
 - Highest Tide: 1.51 meters
```

```
 - Sea Level Rise + Highest Tide: 1.78 meters
   -> Sea level rise + highest tide (1.78 meters) exceeds elevation (0.27
meters).
Year 2060:
 - Sea Level Rise: 0.31 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.82 meters
   -> Sea level rise + highest tide (1.82 meters) exceeds elevation (0.27
meters).
Year 2070:
 - Sea Level Rise: 0.35 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.86 meters
   -> Sea level rise + highest tide (1.86 meters) exceeds elevation (0.27
meters).
Year 2080:
 - Sea Level Rise: 0.37 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.88 meters
   -> Sea level rise + highest tide (1.88 meters) exceeds elevation (0.27
meters).
Year 2090:
 - Sea Level Rise: 0.40 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.91 meters
   -> Sea level rise + highest tide (1.91 meters) exceeds elevation (0.27
meters).
Year 2100:
 - Sea Level Rise: 0.42 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.93 meters
   -> Sea level rise + highest tide (1.93 meters) exceeds elevation (0.27
meters).
Year 2110:
 - Sea Level Rise: 0.46 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.97 meters
   -> Sea level rise + highest tide (1.97 meters) exceeds elevation (0.27
meters).
Year 2120:
 - Sea Level Rise: 0.49 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.00 meters
   -> Sea level rise + highest tide (2.00 meters) exceeds elevation (0.27
meters).
Year 2130:
 - Sea Level Rise: 0.52 meters
 - Highest Tide: 1.51 meters
```

```
 - Sea Level Rise + Highest Tide: 2.03 meters
   -> Sea level rise + highest tide (2.03 meters) exceeds elevation (0.27
meters).
Year 2140:
 - Sea Level Rise: 0.54 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.05 meters
   -> Sea level rise + highest tide (2.05 meters) exceeds elevation (0.27
meters).
Year 2150:
 - Sea Level Rise: 0.57 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.08 meters
   -> Sea level rise + highest tide (2.08 meters) exceeds elevation (0.27
meters).

Scenario: 0.5 - MED
Year 2030:
 - Sea Level Rise: 0.17 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.68 meters
   -> Sea level rise + highest tide (1.68 meters) exceeds elevation (0.27
meters).
Year 2040:
 - Sea Level Rise: 0.24 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.75 meters
   -> Sea level rise + highest tide (1.75 meters) exceeds elevation (0.27
meters).
Year 2050:
 - Sea Level Rise: 0.31 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.82 meters
   -> Sea level rise + highest tide (1.82 meters) exceeds elevation (0.27
meters).
Year 2060:
 - Sea Level Rise: 0.38 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.89 meters
   -> Sea level rise + highest tide (1.89 meters) exceeds elevation (0.27
meters).
Year 2070:
 - Sea Level Rise: 0.44 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.95 meters
   -> Sea level rise + highest tide (1.95 meters) exceeds elevation (0.27
meters).
Year 2080:
```

```
 - Sea Level Rise: 0.50 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.01 meters
   -> Sea level rise + highest tide (2.01 meters) exceeds elevation (0.27
meters).
Year 2090:
 - Sea Level Rise: 0.56 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.07 meters
   -> Sea level rise + highest tide (2.07 meters) exceeds elevation (0.27
meters).
Year 2100:
 - Sea Level Rise: 0.61 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.12 meters
   -> Sea level rise + highest tide (2.12 meters) exceeds elevation (0.27
meters).
Year 2110:
 - Sea Level Rise: 0.68 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.19 meters
   -> Sea level rise + highest tide (2.19 meters) exceeds elevation (0.27
meters).
Year 2120:
 - Sea Level Rise: 0.75 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.26 meters
   -> Sea level rise + highest tide (2.26 meters) exceeds elevation (0.27
meters).
Year 2130:
 - Sea Level Rise: 0.82 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.33 meters
   -> Sea level rise + highest tide (2.33 meters) exceeds elevation (0.27
meters).
Year 2140:
 - Sea Level Rise: 0.89 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.40 meters
   -> Sea level rise + highest tide (2.40 meters) exceeds elevation (0.27
meters).
Year 2150:
 - Sea Level Rise: 0.96 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.47 meters
   -> Sea level rise + highest tide (2.47 meters) exceeds elevation (0.27
meters).
```

```
Scenario: 1.0 - MED
Year 2030:
 - Sea Level Rise: 0.18 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.69 meters
   -> Sea level rise + highest tide (1.69 meters) exceeds elevation (0.27
meters).
Year 2040:
 - Sea Level Rise: 0.26 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.77 meters
   -> Sea level rise + highest tide (1.77 meters) exceeds elevation (0.27
meters).
Year 2050:
 - Sea Level Rise: 0.35 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.86 meters
   -> Sea level rise + highest tide (1.86 meters) exceeds elevation (0.27
meters).
Year 2060:
 - Sea Level Rise: 0.45 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.96 meters
   -> Sea level rise + highest tide (1.96 meters) exceeds elevation (0.27
meters).
Year 2070:
 - Sea Level Rise: 0.57 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.08 meters
   -> Sea level rise + highest tide (2.08 meters) exceeds elevation (0.27
meters).
Year 2080:
 - Sea Level Rise: 0.70 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.21 meters
   -> Sea level rise + highest tide (2.21 meters) exceeds elevation (0.27
meters).
Year 2090:
 - Sea Level Rise: 0.88 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.39 meters
   -> Sea level rise + highest tide (2.39 meters) exceeds elevation (0.27
meters).
Year 2100:
 - Sea Level Rise: 1.07 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.58 meters
   -> Sea level rise + highest tide (2.58 meters) exceeds elevation (0.27
```

```
meters).
Year 2110:
 - Sea Level Rise: 1.28 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.79 meters
   -> Sea level rise + highest tide (2.79 meters) exceeds elevation (0.27
meters).
Year 2120:
 - Sea Level Rise: 1.48 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.99 meters
   -> Sea level rise + highest tide (2.99 meters) exceeds elevation (0.27
meters).
Year 2130:
 - Sea Level Rise: 1.66 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 3.17 meters
   -> Sea level rise + highest tide (3.17 meters) exceeds elevation (0.27
meters).
Year 2140:
 - Sea Level Rise: 1.83 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 3.34 meters
   -> Sea level rise + highest tide (3.34 meters) exceeds elevation (0.27
meters).
Year 2150:
 - Sea Level Rise: 1.99 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 3.50 meters
   -> Sea level rise + highest tide (3.50 meters) exceeds elevation (0.27
meters).

Scenario: 1.5 - MED
Year 2030:
 - Sea Level Rise: 0.19 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.70 meters
   -> Sea level rise + highest tide (1.70 meters) exceeds elevation (0.27
meters).
Year 2040:
 - Sea Level Rise: 0.28 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.79 meters
   -> Sea level rise + highest tide (1.79 meters) exceeds elevation (0.27
meters).
Year 2050:
 - Sea Level Rise: 0.40 meters
 - Highest Tide: 1.51 meters
```

```
 - Sea Level Rise + Highest Tide: 1.91 meters
   -> Sea level rise + highest tide (1.91 meters) exceeds elevation (0.27
meters).
Year 2060:
 - Sea Level Rise: 0.55 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.06 meters
   -> Sea level rise + highest tide (2.06 meters) exceeds elevation (0.27
meters).
Year 2070:
 - Sea Level Rise: 0.72 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.23 meters
   -> Sea level rise + highest tide (2.23 meters) exceeds elevation (0.27
meters).
Year 2080:
 - Sea Level Rise: 0.92 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.43 meters
   -> Sea level rise + highest tide (2.43 meters) exceeds elevation (0.27
meters).
Year 2090:
 - Sea Level Rise: 1.16 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.67 meters
   -> Sea level rise + highest tide (2.67 meters) exceeds elevation (0.27
meters).
Year 2100:
 - Sea Level Rise: 1.40 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.91 meters
   -> Sea level rise + highest tide (2.91 meters) exceeds elevation (0.27
meters).
Year 2110:
 - Sea Level Rise: 1.63 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 3.14 meters
   -> Sea level rise + highest tide (3.14 meters) exceeds elevation (0.27
meters).
Year 2120:
 - Sea Level Rise: 1.83 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 3.34 meters
   -> Sea level rise + highest tide (3.34 meters) exceeds elevation (0.27
meters).
Year 2130:
 - Sea Level Rise: 1.99 meters
 - Highest Tide: 1.51 meters
```

```
 - Sea Level Rise + Highest Tide: 3.50 meters
   -> Sea level rise + highest tide (3.50 meters) exceeds elevation (0.27
meters).
Year 2140:
 - Sea Level Rise: 2.16 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 3.67 meters
   -> Sea level rise + highest tide (3.67 meters) exceeds elevation (0.27
meters).
Year 2150:
 - Sea Level Rise: 2.33 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 3.84 meters
   -> Sea level rise + highest tide (3.84 meters) exceeds elevation (0.27
meters).

Scenario: 2.0 - MED
Year 2030:
 - Sea Level Rise: 0.19 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.70 meters
   -> Sea level rise + highest tide (1.70 meters) exceeds elevation (0.27
meters).
Year 2040:
 - Sea Level Rise: 0.29 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.80 meters
   -> Sea level rise + highest tide (1.80 meters) exceeds elevation (0.27
meters).
Year 2050:
 - Sea Level Rise: 0.43 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.94 meters
   -> Sea level rise + highest tide (1.94 meters) exceeds elevation (0.27
meters).
Year 2060:
 - Sea Level Rise: 0.63 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.14 meters
   -> Sea level rise + highest tide (2.14 meters) exceeds elevation (0.27
meters).
Year 2070:
 - Sea Level Rise: 0.88 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.39 meters
   -> Sea level rise + highest tide (2.39 meters) exceeds elevation (0.27
meters).
Year 2080:
```

```
 - Sea Level Rise: 1.17 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.68 meters
   -> Sea level rise + highest tide (2.68 meters) exceeds elevation (0.27
meters).
Year 2090:
 - Sea Level Rise: 1.48 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 2.99 meters
   -> Sea level rise + highest tide (2.99 meters) exceeds elevation (0.27
meters).
Year 2100:
 - Sea Level Rise: 1.81 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 3.32 meters
   -> Sea level rise + highest tide (3.32 meters) exceeds elevation (0.27
meters).
Year 2110:
 - Sea Level Rise: 2.15 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 3.66 meters
   -> Sea level rise + highest tide (3.66 meters) exceeds elevation (0.27
meters).
Year 2120:
 - Sea Level Rise: 2.51 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 4.02 meters
   -> Sea level rise + highest tide (4.02 meters) exceeds elevation (0.27
meters).
Year 2130:
 - Sea Level Rise: 2.81 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 4.32 meters
   -> Sea level rise + highest tide (4.32 meters) exceeds elevation (0.27
meters).
Year 2140:
 - Sea Level Rise: 3.05 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 4.56 meters
   -> Sea level rise + highest tide (4.56 meters) exceeds elevation (0.27
meters).
Year 2150:
 - Sea Level Rise: 3.26 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 4.77 meters
   -> Sea level rise + highest tide (4.77 meters) exceeds elevation (0.27
meters).
```

7 Find Most Mild Solution in which you are underwater by 2050

[ ]:
```
def find_lowest_scenario_underwater(scenarios, highest_tide, elevation):
    # Sort scenarios numerically based on the scenario title (before ' - ')
    sorted_scenarios = sorted(scenarios, key=lambda x: float(x[0].split(' -␣
 ↪')[0]))

    for scenario, sea_level_data in sorted_scenarios:
        # Retrieve the sea level rise value for 2050
        sea_level_2050 = sea_level_data.get('rsl2050cm', 0)

        # Calculate the sum of highest_tide and sea level rise for 2050
        combined_rise = highest_tide + sea_level_2050

        # Check if the combined rise exceeds the elevation
        if combined_rise > elevation:
            return scenario, combined_rise, sea_level_2050

    # If no scenario is found where location is underwater in 2050
    return None

# Call main to get the result data
result = main(43.66843, -70.24020, api_key, stations, slrdata)

# Extract parameters from the result to use with find_lowest_scenario_underwater
scenarios = result['scenarios']
highest_tide = result['highest_tide']
elevation = result['elevation']

# Call the function to find the lowest scenario where location is underwater in␣
 ↪2050
lowest_scenario_result = find_lowest_scenario_underwater(scenarios,␣
 ↪highest_tide, elevation)

# Print the elevation of the coordinates
print(f"Elevation at location ({result['latitude']}, {result['longitude']}):␣
 ↪{result['elevation']:.2f} meters")

# Display the result of the lowest scenario check
if lowest_scenario_result:
    scenario, combined_rise, sea_level_2050 = lowest_scenario_result
    print(f"Lowest scenario where location is underwater in 2050: {scenario}")
    print(f" - Sea Level Rise in 2050: {sea_level_2050:.2f} meters")
    print(f" - Highest Tide: {result['highest_tide']:.2f} meters")
    print(f" - Sea Level Rise + Highest Tide: {combined_rise:.2f} meters")
    print(f"   -> Sea level rise + highest tide ({combined_rise:.2f} meters)␣
 ↪exceeds elevation ({result['elevation']:.2f} meters).")
```

```
else:
    print("Location is not underwater in 2050 for any scenario.")
```

```
Elevation at location (43.66843, -70.2402): 0.27 meters
Lowest scenario where location is underwater in 2050: 0.3 - MED
 - Sea Level Rise in 2050: 0.27 meters
 - Highest Tide: 1.51 meters
 - Sea Level Rise + Highest Tide: 1.78 meters
   -> Sea level rise + highest tide (1.78 meters) exceeds elevation (0.27
meters).
```

[ ]:
```
import matplotlib.pyplot as plt
import pandas as pd

def plot_scenario_projections(result):
    # Extract elevation and highest tide from the result
    elevation = result['elevation']
    highest_tide = result['highest_tide']

    # Years and baseline for elevation
    years = [int(year[3:7]) for year in result['scenarios'][0][1].keys()]  #␣
 ↪Extract year numbers
    elevation_line = [elevation] * len(years)

    # Prepare data for each scenario
    scenario_lines = {}
    for scenario, data in result['scenarios']:
        # Start cumulative rise at the highest tide
        cumulative_rise = [highest_tide]
        for year in years[1:]:  # Start from the second year since the first is␣
 ↪just the highest tide
            year_key = f"rsl{year}cm"
            sea_level_rise = data.get(year_key, 0)  # Get sea level rise for␣
 ↪the year or 0 if missing
            cumulative_rise.append(cumulative_rise[-1] + sea_level_rise)
        scenario_lines[scenario] = cumulative_rise

    # Convert years to DataFrame for easy plotting
    df = pd.DataFrame(scenario_lines, index=years)
    df['Elevation'] = elevation_line

    # Plot
    plt.figure(figsize=(10, 6))
    for scenario in df.columns[:-1]:  # Plot each scenario line except␣
 ↪'Elevation'
        plt.plot(df.index, df[scenario], label=f'Scenario {scenario}')
```

```
    plt.plot(df.index, df['Elevation'], label='Elevation', linestyle='--',␣
 ↪color='black')

    # Labels and title
    plt.xlabel('Year')
    plt.ylabel('Meters')
    plt.title('Sea Level Rise Projections with Highest Tide')
    plt.legend()
    plt.grid()
    plt.show()

# Call the function to plot with result data
plot_scenario_projections(result)
```