Context:

The oil and gas industry plays a vital role in Texas's economy but is also a major source of air pollution, particularly hazardous air pollutants (HAPs) such as Benzene, Toluene, Xylenes, and 1,3-Butadiene. These volatile organic compounds (VOCs) are known to pose significant risks to human health, including respiratory diseases and cancer, especially in communities located near extraction and processing facilities. While regulatory monitoring stations collect air quality data across the state, the temporal and spatial patterns of these toxic pollutants are not always well understood.

In [46]:
```
# Library to suppress warnings or deprecation notes
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
import os
import requests
import zipfile
import subprocess
import pandas as pd
import folium
from IPython.display import display
import seaborn as sns
import geopandas as gpd
from shapely.geometry import Point
from scipy.stats import linregress
from matplotlib.colors import LinearSegmentedColormap
```

1. Accessing Texas HAPs Data:

In [13]:
```
# Step 1: Set up directory
output_dir = "/Users/nawarajadhikari/Capstone_Project"
os.makedirs(output_dir, exist_ok=True)

# Step 2: EPA file URLs
urls = [
    "https://www.epa.gov/system/files/other-files/2024-11/tx_ama_haps_pt1.zip",
    "https://www.epa.gov/system/files/other-files/2024-11/tx_ama_haps_pt2.zip"
]

# Step 3: Download and extract non-.accdb files
for url in urls:
```

```
    zip_filename = os.path.basename(url)
    zip_path = os.path.join(output_dir, zip_filename)
    print(f"Downloading {url} ...")

    try:
        response = requests.get(url, stream=True)
        response.raise_for_status()

        with open(zip_path, "wb") as f:
            for chunk in response.iter_content(chunk_size=8192):
                f.write(chunk)
        print(f"Downloaded {zip_filename}")

        # Extract files (skip .accdb)
        with zipfile.ZipFile(zip_path, 'r') as zip_ref:
            for member in zip_ref.namelist():
                if not member.lower().endswith(".accdb"):
                    print(f"  Extracting {member}")
                    zip_ref.extract(member, output_dir)

        print(f"✅ Done extracting from {zip_filename}")
        # ⚠️ DO NOT delete zip files this time

    except Exception as e:
        print(f"❌ Error: {e}")
```

```
Downloading https://www.epa.gov/system/files/other-files/2024-11/tx_ama_haps_pt1.zip ...
Downloaded tx_ama_haps_pt1.zip
✅ Done extracting from tx_ama_haps_pt1.zip
Downloading https://www.epa.gov/system/files/other-files/2024-11/tx_ama_haps_pt2.zip ...
Downloaded tx_ama_haps_pt2.zip
✅ Done extracting from tx_ama_haps_pt2.zip
```

In [36]:
```
# Downloaded the datatset of Texas from the given link.
# Dataset was too large, downloded with two separate files pt1 and pt2.
# Downloaded files are in .accdb format we need these to be converted into .csv file for the better readablil
```

2. Converting the downloaded dataset from .accdb to .csv format

In [14]:
```
# Folder containing .accdb files
source_dir = "/Users/nawarajadhikari/Capstone_Project"
output_dir = os.path.join(source_dir, "csv_exports")
os.makedirs(output_dir, exist_ok=True)
```

```
# Walk through all .accdb files
for root, dirs, files in os.walk(source_dir):
    for file in files:
        if file.endswith(".accdb"):
            accdb_path = os.path.join(root, file)
            print(f"\n🔍 Found {file}")
            try:
                # Get table names
                tables = subprocess.check_output(["mdb-tables", "-1", accdb_path]).decode().splitlines()
                for table in tables:
                    table = table.strip()
                    if not table:
                        continue
                    csv_name = f"{file.replace('.accdb','')}_{table}.csv"
                    csv_path = os.path.join(output_dir, csv_name)
                    print(f"➡️ Exporting table: {table}")
                    with open(csv_path, "w") as f:
                        subprocess.run(["mdb-export", accdb_path, table], stdout=f)
                    print(f"✅ Saved to {csv_path}")
            except Exception as e:
                print(f"❌ Error processing {file}: {e}")
```

```
🔍 Found TX_AMA_HAPS_2016.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2016_AMBIENT_MONITORING_ARCHIVE_O
UTPUT.csv

🔍 Found TX_AMA_HAPS_2014.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2014_AMBIENT_MONITORING_ARCHIVE_O
UTPUT.csv

🔍 Found TX_AMA_HAPS_2015.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2015_AMBIENT_MONITORING_ARCHIVE_O
UTPUT.csv

🔍 Found TX_AMA_HAPS_2017.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2017_AMBIENT_MONITORING_ARCHIVE_O
UTPUT.csv

🔍 Found TX_AMA_HAPS_2020.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2020_AMBIENT_MONITORING_ARCHIVE_O
UTPUT.csv

🔍 Found TX_AMA_HAPS_2022.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2022_AMBIENT_MONITORING_ARCHIVE_O
UTPUT.csv

🔍 Found TX_AMA_HAPS_2019.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2019_AMBIENT_MONITORING_ARCHIVE_O
UTPUT.csv

🔍 Found TX_AMA_HAPS_2021.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2021_AMBIENT_MONITORING_ARCHIVE_O
UTPUT.csv

🔍 Found TX_AMA_HAPS_2018.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2018_AMBIENT_MONITORING_ARCHIVE_O
UTPUT.csv
```

```
🔍 Found TX_AMA_HAPS_2009.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2009_AMBIENT_MONITORING_ARCHIVE_O
UTPUT.csv

🔍 Found TX_AMA_HAPS_2010.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2010_AMBIENT_MONITORING_ARCHIVE_O
UTPUT.csv

🔍 Found TX_AMA_HAPS_2012.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2012_AMBIENT_MONITORING_ARCHIVE_O
UTPUT.csv

🔍 Found TX_AMA_HAPS_2013.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2013_AMBIENT_MONITORING_ARCHIVE_O
UTPUT.csv

🔍 Found TX_AMA_HAPS_2008.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2008_AMBIENT_MONITORING_ARCHIVE_O
UTPUT.csv

🔍 Found TX_AMA_HAPS_2011.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2011_AMBIENT_MONITORING_ARCHIVE_O
UTPUT.csv

🔍 Found TX_AMA_HAPS_2006.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2006_AMBIENT_MONITORING_ARCHIVE_O
UTPUT.csv

🔍 Found TX_AMA_HAPS_1990_2000.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_1990_2000_AMBIENT_MONITORING_ARCH
IVE_OUTPUT.csv

🔍 Found TX_AMA_HAPS_2007.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2007_AMBIENT_MONITORING_ARCHIVE_O
UTPUT.csv
```

```
🔍 Found TX_AMA_HAPS_2001_2003.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2001_2003_AMBIENT_MONITORING_ARCHIVE_OUTPUT.csv

🔍 Found TX_AMA_HAPS_2004_2005.accdb
➡️ Exporting table: AMBIENT_MONITORING_ARCHIVE_OUTPUT
✅ Saved to /Users/nawarajadhikari/Capstone_Project/csv_exports/TX_AMA_HAPS_2004_2005_AMBIENT_MONITORING_ARCHIVE_OUTPUT.csv
```

In [18]:
```
# Converted all the files into a single .csv file and store it in a separate folder, csv_exports.
```

3. Extracting some sepecific columns from the dataset.

Columns extracted:

- AMA_SITE_CODE
- SAMPLE_DATE
- AQS_PARAMETER_NAME
- MONITOR_LATITUDE,
- MONITER_LONGITUDE

In [21]:
```
# Step 1: File path
folder = "/Users/nawarajadhikari/Capstone_Project/csv_exports"

# Step 2: Columns  want to extract
cols = ['AMA_SITE_CODE', 'SAMPLE_DATE', 'AQS_PARAMETER_NAME', 'MONITOR_LATITUDE', 'MONITOR_LONGITUDE']

# Step 3: Combine and filter
dfs = []

for file in os.listdir(folder):
    if file.endswith(".csv"):
        path = os.path.join(folder, file)
        try:
            df = pd.read_csv(path, usecols=cols, low_memory=False)
            df = df[df['AQS_PARAMETER_NAME'] == 'Benzene']  # Keep only Benzene
            dfs.append(df)
        except Exception as e:
            print(f"Skipping {file}: {e}")

# Step 4: Combine and keep only unique AMA_SITE_CODE rows
```

```
combined = pd.concat(dfs, ignore_index=True)
final_df = combined.drop_duplicates(subset='AMA_SITE_CODE')

# Step 5: Save the cleaned dataset
output_path = "/Users/nawarajadhikari/Capstone_Project/filtered_benzene_sites.csv"
final_df.to_csv(output_path, index=False)

print(f" Done! Saved to:\n{output_path}")
print(final_df.head())
```

```
 Done! Saved to:
/Users/nawarajadhikari/Capstone_Project/filtered_benzene_sites.csv
    AMA_SITE_CODE        SAMPLE_DATE AQS_PARAMETER_NAME  MONITOR_LATITUDE  \
0       480290051  01/01/01 00:00:00            Benzene         29.445045
183     480391003  01/01/01 00:00:00            Benzene         29.010841
377     480610006  01/01/01 00:00:00            Benzene         25.892517
561     481130057  01/01/01 00:00:00            Benzene         32.778889
745     481130069  01/01/01 00:00:00            Benzene         32.820061

     MONITOR_LONGITUDE
0           -98.406502
183         -95.397743
377         -97.493828
561         -96.873055
745         -96.860115
```

In [22]:
```
# Exported the data in a new file: filtered_benzene_sites.csv
# It is now much smaller and reday for modeling
# It contains only one row per AMA_SITE_CODE(Unique) where Benzene was measured.
```

4. Creating another dataset, File2

Contains:

- AMA_SITE_CODE
- SAMPLE_DATE
- BENZENE: Mesurement of benzene level of all years and each date and time
- MONITOR_LATITUDE
- MONITOR_LONGITUDE

In [24]:
```
# File path
csv_folder = "/Users/nawarajadhikari/Capstone_Project/csv_exports"
```

```
# Defining required columns and initializing list to hold filtered data
required_columns = ['AMA_SITE_CODE', 'SAMPLE_DATE', 'AQS_PARAMETER_NAME', 'SAMPLE_VALUE_REPORTED', 'MONITOR_L
filtered_dfs = []

# Loop through CSV files
for file in os.listdir(csv_folder):
    if file.endswith(".csv"):
        file_path = os.path.join(csv_folder, file)
        try:
            # Load and filter
            df = pd.read_csv(file_path, usecols=required_columns, low_memory=False)
            df = df[df['AQS_PARAMETER_NAME'] == 'Benzene']
            df = df[['AMA_SITE_CODE', 'SAMPLE_DATE', 'SAMPLE_VALUE_REPORTED','MONITOR_LATITUDE','MONITOR_LONG
            df = df.rename(columns={'SAMPLE_VALUE_REPORTED': 'Benzene'})
            filtered_dfs.append(df)
        except Exception as e:
            print(f"Skipping {file}: {e}")

# Combining all filtered data and save
benzene_full_df = pd.concat(filtered_dfs, ignore_index=True)

# Displaying the result
print(benzene_full_df.head())
```

```
  AMA_SITE_CODE        SAMPLE_DATE  Benzene  MONITOR_LATITUDE  \
0     480290051  01/01/01 00:00:00      NaN         29.445045
1     480290051  01/07/01 00:00:00      NaN         29.445045
2     480290051  01/13/01 00:00:00     0.85         29.445045
3     480290051  01/19/01 00:00:00     0.40         29.445045
4     480290051  01/25/01 00:00:00      NaN         29.445045

   MONITOR_LONGITUDE
0         -98.406502
1         -98.406502
2         -98.406502
3         -98.406502
4         -98.406502
```

In [ ]:
```
# Exporting
# Export path
output_path = "/Users/nawarajadhikari/Capstone_Project/File2.csv"

# Export to CSV
benzene_full_df.to_csv(output_path, index=False)
```

```
print(f" Benzene dataset saved to: {output_path}")
```

```
 Benzene dataset saved to: /Users/nawarajadhikari/Capstone_Project/File2.csv
```

5. Computing Average Benzene value, File3

Contains:

- AMA_SITE_CODE
- SAMPLE_DATE
- BENZENE: Mesurement of average value of benzene level of all years, same site, and each day.
- MONITOR_LATITUDE
- MONITOR_LONGITUDE

In [25]:
```
file_path = "/Users/nawarajadhikari/Capstone_Project/File2.csv"
benzene_full_df = pd.read_csv(file_path)

# Converting SAMPLE_DATE to datetime
benzene_full_df['SAMPLE_DATE'] = pd.to_datetime(benzene_full_df['SAMPLE_DATE'], errors='coerce')

# Dropping rows with missing Benzene values or invalid dates
benzene_full_df = benzene_full_df.dropna(subset=['Benzene', 'SAMPLE_DATE'])

# Filter data for the years 1990 to 2022
benzene_full_df = benzene_full_df[
    (benzene_full_df['SAMPLE_DATE'].dt.year >= 1990) &
    (benzene_full_df['SAMPLE_DATE'].dt.year <= 2022)
]

# Group by AMA_SITE_CODE and SAMPLE_DATE to get daily average Benzene
daily_benzene_df = benzene_full_df.groupby(
    ['AMA_SITE_CODE', 'SAMPLE_DATE']
).agg({
    'Benzene': 'mean',
    'MONITOR_LATITUDE': 'first',
    'MONITOR_LONGITUDE': 'first'
}).reset_index()

# Rename the Benzene column to BENZENE for columns name uniformaity in Capital letter
daily_benzene_df = daily_benzene_df.rename(columns={'Benzene': 'BENZENE'})
```

```
# Selecting and order the desired columns
final_daily_df = daily_benzene_df[[
    'AMA_SITE_CODE', 'SAMPLE_DATE', 'BENZENE', 'MONITOR_LATITUDE', 'MONITOR_LONGITUDE'
]]

# Sort by SAMPLE_DATE
final_daily_df = final_daily_df.sort_values(by='SAMPLE_DATE', ascending=True)

print(final_daily_df.head())
print(final_daily_df.tail())
```

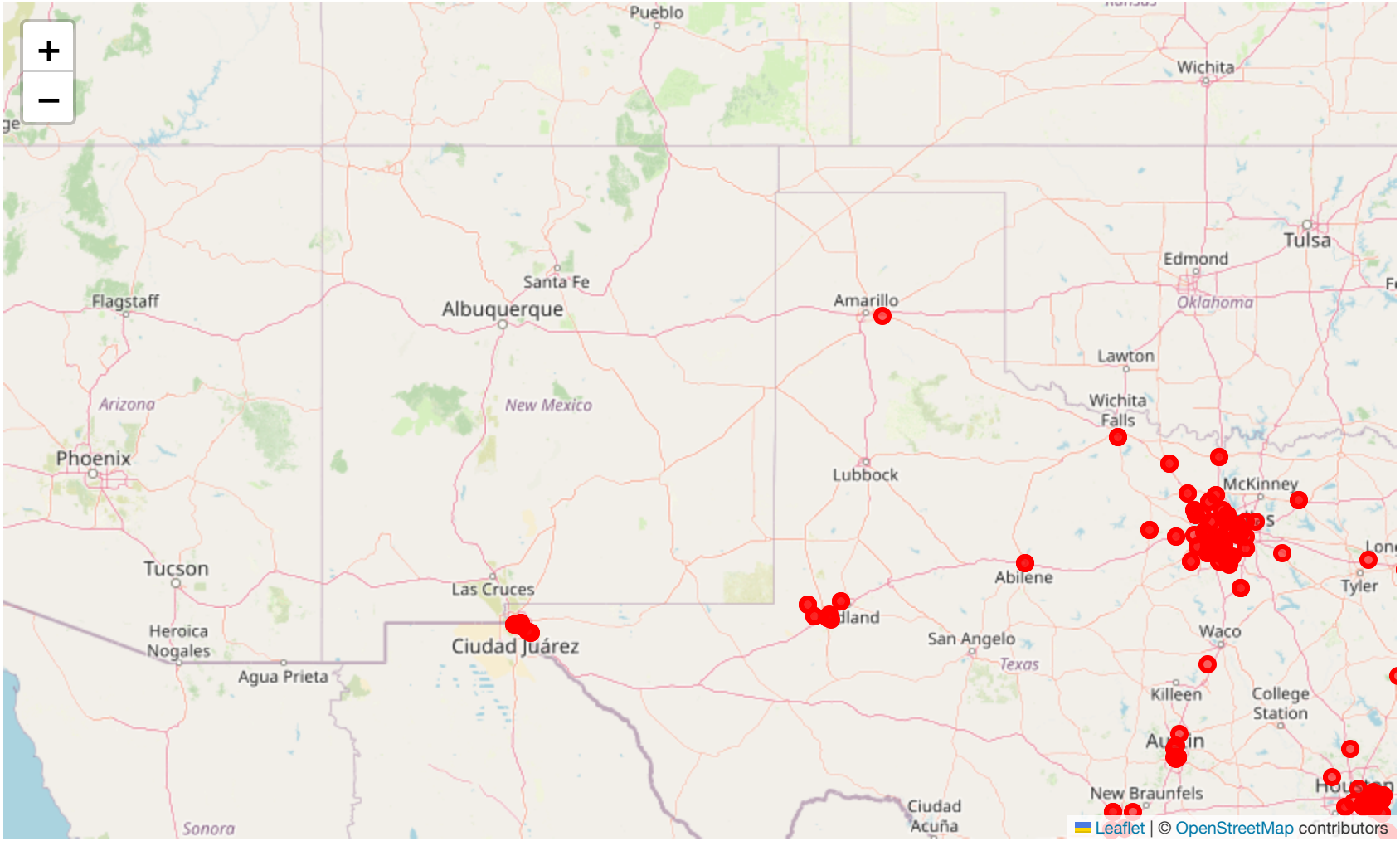
```
      AMA_SITE_CODE SAMPLE_DATE  BENZENE  MONITOR_LATITUDE  MONITOR_LONGITUDE
98991     482011034  1990-01-03     3.17         29.767998         -95.220581
98992     482011034  1990-01-11     2.59         29.767998         -95.220581
8105      481130069  1990-01-11     0.80         32.820061         -96.860115
98993     482011034  1990-01-17     1.58         29.767998         -95.220581
98994     482011034  1990-01-23     7.56         29.767998         -95.220581
       AMA_SITE_CODE SAMPLE_DATE   BENZENE  MONITOR_LATITUDE  \
69483      482010036  2022-12-31  1.221999         29.776100
113025     482011039  2022-12-31  0.026137         29.670025
158082     484390075  2022-12-31  0.155298         32.987892
71325      482010055  2022-12-31  0.470000         29.695728
176470     484392003  2022-12-31  0.300000         32.922474

        MONITOR_LONGITUDE
69483          -95.105103
113025         -95.128510
158082         -97.477173
71325          -95.499222
176470         -97.282089
```

In [18]:
```
# Exporting
# Defining export path
output_path = "/Users/nawarajadhikari/Capstone_Project/File3.csv"

# Export to CSV
final_daily_df.to_csv(output_path, index=False)

print(f"✅ Benzene dataset saved to: {output_path}")
```

```
✅ Benzene dataset saved to: /Users/nawarajadhikari/Capstone_Project/File3.csv
```

6. State boundary map with Benzene monitoring sites

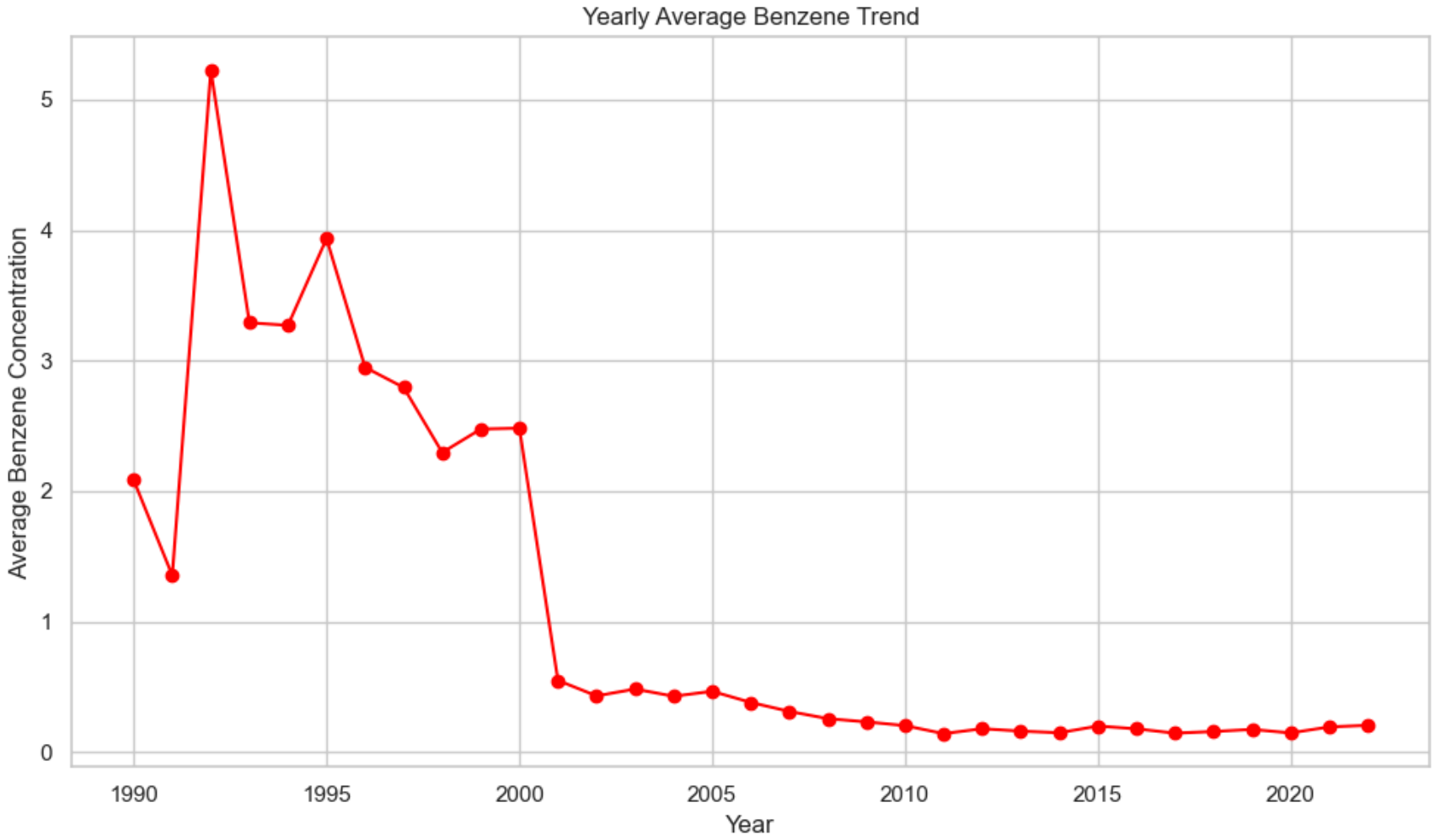
In [3]:
```
# Loading the CSV file
file_path = "/Users/nawarajadhikari/Capstone_Project/File1.csv"
df = pd.read_csv(file_path)

# Cleaning the column names and ensuring coordinates are numeric
df.columns = df.columns.str.strip()
df['MONITOR_LATITUDE'] = pd.to_numeric(df['MONITOR_LATITUDE'], errors='coerce')
df['MONITOR_LONGITUDE'] = pd.to_numeric(df['MONITOR_LONGITUDE'], errors='coerce')
df = df.dropna(subset=['MONITOR_LATITUDE', 'MONITOR_LONGITUDE'])

# Center the map around Texas
m = folium.Map(location=[31.5, -99.0], zoom_start=6)

# Monitoring sites
for _, row in df.iterrows():
    folium.CircleMarker(
        location=[row['MONITOR_LATITUDE'], row['MONITOR_LONGITUDE']],
        radius=4,
        color='Red',
        fill=True,
        fill_opacity=0.6,
        popup=f"Site: {row.get('AMA_SITE_CODE', 'Unknown')}"
    ).add_to(m)

# Displaying the map
display(m)
```



7. Year Over Year Trend of Benzene

In [24]:
```
# Loading the dataset
file_path = "/Users/nawarajadhikari/Capstone_Project/File3.csv"
df = pd.read_csv(file_path)

# Stripping column names of extra spaces
df.columns = df.columns.str.strip()

# Converting SAMPLE_DATE to datetime and Benzene to numeric
df['SAMPLE_DATE'] = pd.to_datetime(df['SAMPLE_DATE'], errors='coerce')
df['BENZENE'] = pd.to_numeric(df['BENZENE'], errors='coerce')
```

```
# Dropping rows with missing values in those two columns
df = df.dropna(subset=['SAMPLE_DATE', 'BENZENE'])

# Creating a 'Year' column
df['Year'] = df['SAMPLE_DATE'].dt.year

# Group by year and calculate average
yearly_avg = df.groupby('Year')['BENZENE'].mean().reset_index()

# Columns to clean numeric arrays
x = yearly_avg['Year'].astype(int).values
y = yearly_avg['BENZENE'].astype(float).values

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(x, y, marker='o', linestyle='-', color='red')
plt.title("Yearly Average Benzene Trend")
plt.xlabel("Year")
plt.ylabel("Average Benzene Concentration")
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [30]:
```
df.shape
```

Out[30]:
```
(240411, 6)
```

8. Long Term trend of Benzene by sites

In [44]:
```
# ----------------------------
# 1. City coordinates for labeling
# ----------------------------
cities = {
    "Houston": (29.7604, -95.3698),
```

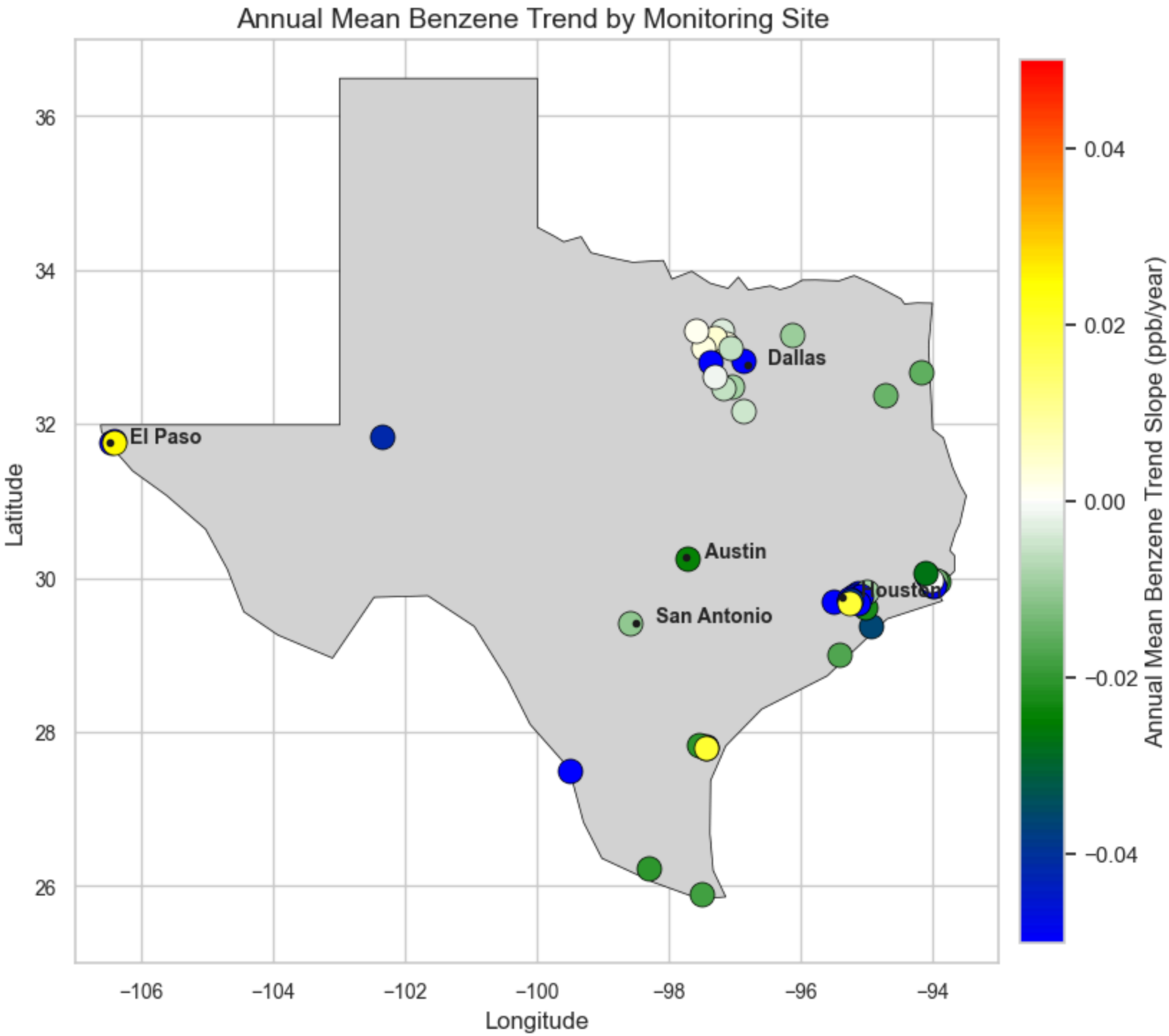
```
    "Dallas": (32.7767, -96.7970),
    "Austin": (30.2672, -97.7431),
    "San Antonio": (29.4241, -98.4936),
    "El Paso": (31.7619, -106.4850)
}

# ----------------------------
# 2. Colormap settings
# ----------------------------
varmin = -0.05
varmax = 0.05
colors = ["blue", "green", "white", "yellow", "red"]
cmap = LinearSegmentedColormap.from_list("bgyr", colors)
norm = plt.Normalize(vmin=varmin, vmax=varmax)
sm = plt.cm.ScalarMappable(cmap=cmap, norm=norm)
sm.set_array([])

# ----------------------------
# 3. Loading and preprocessing data
# ----------------------------
df = pd.read_csv('/Users/nawarajadhikari/Capstone_Project/File3.csv')

# Converting SAMPLE_DATE to datetime
df['SAMPLE_DATE'] = pd.to_datetime(df['SAMPLE_DATE'], errors='coerce')

# Dropping rows with missing benzene values
df.dropna(subset=['BENZENE'], inplace=True)

# Extract year from SAMPLE_DATE
df['Year'] = df['SAMPLE_DATE'].dt.year

# ----------------------------
# 4. Defining site selection criteria
# ----------------------------
def meets_criteria(site_df):
    years = site_df['Year'].unique()
    return len(years) >= 4 and years.min() <= 2011 and years.max() >= 2015

# ----------------------------
# 5. Calculation of  trend (slope) for annual mean Benzene per site
# ----------------------------
site_trends_mean = {}

for site_id, site_data in df.groupby('AMA_SITE_CODE'):
    if meets_criteria(site_data):
```

```
        annual_mean = site_data.groupby('Year')['BENZENE'].mean().reset_index()
        years = annual_mean['Year'].values
        benzene_mean = annual_mean['BENZENE'].values
        if len(years) >= 2:
            slope, _, _, _, _ = linregress(years, benzene_mean)
            lat = site_data['MONITOR_LATITUDE'].iloc[0]
            lon = site_data['MONITOR_LONGITUDE'].iloc[0]
            site_trends_mean[site_id] = {'slope': slope, 'lat': lat, 'lon': lon}
            print(f"Site ID: {site_id}, Annual Mean Trend Slope: {slope:.6f}")

# ----------------------------
# 6. Defining map boundaries
# ----------------------------
min_lon, max_lon = -107, -93
min_lat, max_lat = 25, 37

# ----------------------------
# 7. Loading shapefile
# ----------------------------
# Built-in naturalearth_lowres was not suportvie in this environmnet  and downloaded the natural earth file

# With this:
shapefile_path = '/Users/nawarajadhikari/Capstone_Project/ne_110m_admin_1_states_provinces'
states = gpd.read_file(shapefile_path)

# Now filtering for Texas (USA)
texas_shape = states[states['name'] == 'Texas']


# ----------------------------
# 8. Plotting setup
# ----------------------------
extent = max_lon - min_lon
aspect = np.cos(np.mean([min_lat, max_lat]) * np.pi / 180)
fig_height = 10
fig_width = fig_height * (extent / (max_lat - min_lat)) * aspect

fig, ax = plt.subplots(1, 1, figsize=(fig_width, fig_height))
ax.set_aspect('equal', adjustable='box')

# Plot base map (gray USA outline)
texas_shape.plot(ax=ax, color='lightgray', edgecolor='black', linewidth=0.5, zorder=1)

# Plot Benzene trend points
for site_id, info in site_trends_mean.items():
```

```
    slope = info['slope']
    lat = info['lat']
    lon = info['lon']
    color = cmap(norm(slope))
    ax.scatter(lon, lat, s=150, c=[color], edgecolor='black', linewidth=0.5, zorder=2)

# ----------------------------
# 9. Plot cities
# ----------------------------
for city, (lat, lon) in cities.items():
    ax.plot(lon, lat, 'ko', markersize=3)
    ax.text(lon + 0.3, lat, city, fontsize=10, weight='bold')

# ----------------------------
# 10. Final formatting
# ----------------------------
ax.set_xlim(min_lon, max_lon)
ax.set_ylim(min_lat, max_lat)
ax.set_xlabel('Longitude', fontsize=12)
ax.set_ylabel('Latitude', fontsize=12)
ax.tick_params(axis='both', labelsize=10)
plt.title('Annual Mean Benzene Trend by Monitoring Site ', fontsize=14)

# Adding colorbar
cbar = fig.colorbar(sm, ax=ax, orientation="vertical", pad=0.02, shrink=0.8)
cbar.set_label('Annual Mean Benzene Trend Slope (ppb/year)', fontsize=12)

# Save the figure
plt.savefig('annual_mean_benzene_trend_map.png', bbox_inches='tight')
print("✅ Annual mean benzene trend map saved as 'annual_mean_benzene_trend_map.png'")
```

```
Site ID: 480290677, Annual Mean Trend Slope: -0.010934
Site ID: 480391003, Annual Mean Trend Slope: -0.016884
Site ID: 480610006, Annual Mean Trend Slope: -0.018725
Site ID: 481130069, Annual Mean Trend Slope: -0.100819
Site ID: 481210034, Annual Mean Trend Slope: -0.003412
Site ID: 481211007, Annual Mean Trend Slope: 0.002717
Site ID: 481211013, Annual Mean Trend Slope: 0.004120
Site ID: 481350003, Annual Mean Trend Slope: -0.041755
Site ID: 481390016, Annual Mean Trend Slope: -0.008774
Site ID: 481391044, Annual Mean Trend Slope: -0.004579
Site ID: 481410044, Annual Mean Trend Slope: -0.333063
Site ID: 481410047, Annual Mean Trend Slope: -0.066359
Site ID: 481411011, Annual Mean Trend Slope: 0.025187
Site ID: 481670005, Annual Mean Trend Slope: -0.036306
Site ID: 481830001, Annual Mean Trend Slope: -0.014137
Site ID: 482010026, Annual Mean Trend Slope: -0.047887
Site ID: 482010036, Annual Mean Trend Slope: 0.007949
Site ID: 482010055, Annual Mean Trend Slope: -0.054246
Site ID: 482010057, Annual Mean Trend Slope: -0.036608
Site ID: 482010058, Annual Mean Trend Slope: -0.010763
Site ID: 482010061, Annual Mean Trend Slope: -0.022534
Site ID: 482010069, Annual Mean Trend Slope: -0.026541
Site ID: 482010307, Annual Mean Trend Slope: -0.029896
Site ID: 482010617, Annual Mean Trend Slope: -0.008614
Site ID: 482010803, Annual Mean Trend Slope: -0.045306
Site ID: 482011015, Annual Mean Trend Slope: -0.098565
Site ID: 482011035, Annual Mean Trend Slope: -0.171315
Site ID: 482011039, Annual Mean Trend Slope: -0.099585
Site ID: 482011049, Annual Mean Trend Slope: -0.039575
Site ID: 482016000, Annual Mean Trend Slope: 0.019218
Site ID: 482030002, Annual Mean Trend Slope: -0.015506
Site ID: 482150043, Annual Mean Trend Slope: -0.020394
Site ID: 482311006, Annual Mean Trend Slope: -0.009686
Site ID: 482450009, Annual Mean Trend Slope: -0.059366
Site ID: 482450011, Annual Mean Trend Slope: -0.027354
Site ID: 482450014, Annual Mean Trend Slope: -0.020211
Site ID: 482450017, Annual Mean Trend Slope: -0.012116
Site ID: 482450018, Annual Mean Trend Slope: -0.113287
Site ID: 482450019, Annual Mean Trend Slope: -0.067760
Site ID: 482451035, Annual Mean Trend Slope: -0.000024
Site ID: 482451050, Annual Mean Trend Slope: -0.026979
Site ID: 482511008, Annual Mean Trend Slope: -0.005496
Site ID: 483550029, Annual Mean Trend Slope: -0.042690
Site ID: 483550032, Annual Mean Trend Slope: -0.090755
Site ID: 483550034, Annual Mean Trend Slope: -0.011340
```

```
Site ID: 483550035, Annual Mean Trend Slope: -0.034722
Site ID: 483550041, Annual Mean Trend Slope: -0.020662
Site ID: 483550083, Annual Mean Trend Slope: 0.019799
Site ID: 484390075, Annual Mean Trend Slope: 0.002983
Site ID: 484391002, Annual Mean Trend Slope: -0.456255
Site ID: 484391009, Annual Mean Trend Slope: -0.001319
Site ID: 484393009, Annual Mean Trend Slope: -0.005647
Site ID: 484530021, Annual Mean Trend Slope: -0.024001
Site ID: 484790017, Annual Mean Trend Slope: -0.069676
Site ID: 484970088, Annual Mean Trend Slope: 0.001555
✅ Annual mean benzene trend map saved as 'annual_mean_benzene_trend_map.png'
```


Annual Mean Benzene Trend by Monitoring Site
Dallas
El Paso
Austin
San Antonio
Houston
Latitude
Longitude
36
34
32
30
28
26
−106
−104
−102
−100
−98
−96
−94
0.04
0.02
0.00
−0.02
−0.04
Annual Mean Benzene Trend Slope (ppb/year)

9. Extracting data for Formaldehyde

In [ ]:
```
# Step 1: Define folder path
folder = "/Users/nawarajadhikari/Capstone_Project/csv_exports"

# Step 2: Columns you want to extract
cols = ['AMA_SITE_CODE', 'SAMPLE_DATE', 'AQS_PARAMETER_NAME', 'MONITOR_LATITUDE', 'MONITOR_LONGITUDE']

# Step 3: Combine and filter
dfs = []

for file in os.listdir(folder):
    if file.endswith(".csv"):
        path = os.path.join(folder, file)
        try:
            df = pd.read_csv(path, usecols=cols, low_memory=False)
            df = df[df['AQS_PARAMETER_NAME'] == 'Formaldehyde']  # Keep only Formaldehyde
            dfs.append(df)
        except Exception as e:
            print(f"Skipping {file}: {e}")

# Step 4: Combine and keep only unique AMA_SITE_CODE rows
combined = pd.concat(dfs, ignore_index=True)
final_df = combined.drop_duplicates(subset='AMA_SITE_CODE')

# Step 5: Save the cleaned dataset
output_path = "/Users/nawarajadhikari/Capstone_Project/filtered_formaldehyde_sites.csv"
final_df.to_csv(output_path, index=False)

print(f"✅ Done! Saved to:\n{output_path}")
print(final_df.head())
```

```
✅ Done! Saved to:
/Users/nawarajadhikari/Capstone_Project/filtered_formaldehyde_sites.csv
    AMA_SITE_CODE        SAMPLE_DATE AQS_PARAMETER_NAME  MONITOR_LATITUDE  \
0       481130069  05/12/01 00:00:00       Formaldehyde         32.820061
298     481410044  01/01/01 00:00:00       Formaldehyde         31.765684
586     481410055  01/07/01 00:00:00       Formaldehyde         31.746775
616     482010024  01/01/01 00:00:00       Formaldehyde         29.901035
686     482010026  06/15/02 00:00:00       Formaldehyde         29.802708

     MONITOR_LONGITUDE
0           -96.860115
298        -106.455230
586        -106.402810
616         -95.326134
686         -95.125496
```

In [33]:

```
# File path
csv_folder = "/Users/nawarajadhikari/Capstone_Project/csv_exports"

# Defining required columns and initializing list to hold filtered data
required_columns = ['AMA_SITE_CODE', 'SAMPLE_DATE', 'AQS_PARAMETER_NAME', 'SAMPLE_VALUE_REPORTED', 'MONITOR_L
filtered_dfs = []

# Loop through CSV files
for file in os.listdir(csv_folder):
    if file.endswith(".csv"):
        file_path = os.path.join(csv_folder, file)
        try:
            # Load and filter
            df = pd.read_csv(file_path, usecols=required_columns, low_memory=False)
            df = df[df['AQS_PARAMETER_NAME'] == 'Formaldehyde']
            df = df[['AMA_SITE_CODE', 'SAMPLE_DATE', 'SAMPLE_VALUE_REPORTED','MONITOR_LATITUDE','MONITOR_LONG
            df = df.rename(columns={'SAMPLE_VALUE_REPORTED': 'Formaldehyde'})
            filtered_dfs.append(df)
        except Exception as e:
            print(f"Skipping {file}: {e}")

# Combining all filtered data and save
Formaldehyde_full_df = pd.concat(filtered_dfs, ignore_index=True)

# Displaying the result
print(Formaldehyde_full_df.head())
```

```
  AMA_SITE_CODE        SAMPLE_DATE  Formaldehyde  MONITOR_LATITUDE  \
0     481130069  05/12/01 00:00:00         3.968         32.820061
1     481130069  05/12/01 00:00:00         3.238         32.820061
2     481130069  05/12/01 00:00:00         5.214         32.820061
3     481130069  05/12/01 00:00:00         4.545         32.820061
4     481130069  06/02/01 00:00:00         4.718         32.820061

   MONITOR_LONGITUDE
0         -96.860115
1         -96.860115
2         -96.860115
3         -96.860115
4         -96.860115
```

10. Average Formaldehyde value of all years same site and each day.

Contains:

- AMA_SITE_CODE
- SAMPLE_DATE
- Formaldehyde: Mesurement of average value of Formaldehyde level of all years, same site, and each day.
- MONITOR_LATITUDE
- MONITOR_LONGITUDE

In [68]:
```
# Converting SAMPLE_DATE to datetime
Formaldehyde_full_df['SAMPLE_DATE'] = pd.to_datetime(Formaldehyde_full_df['SAMPLE_DATE'], errors='coerce')

# Dropping rows with missing Formaldehyde values or invalid dates
Formaldehyde_full_df = Formaldehyde_full_df.dropna(subset=['Formaldehyde', 'SAMPLE_DATE'])

# Filter data for the years 1990 to 2022
Formaldehyde_full_df = Formaldehyde_full_df[
    (Formaldehyde_full_df['SAMPLE_DATE'].dt.year >= 1990) &
    (Formaldehyde_full_df['SAMPLE_DATE'].dt.year <= 2022)
]

# Group by AMA_SITE_CODE and SAMPLE_DATE to get daily average Formaldehyde
daily_Formaldehyde_df = Formaldehyde_full_df.groupby(
    ['AMA_SITE_CODE', 'SAMPLE_DATE']
).agg({
    'Formaldehyde': 'mean',
    'MONITOR_LATITUDE': 'first',
```

```
    'MONITOR_LONGITUDE': 'first'
}).reset_index()

# Rename the Formaldehyde column to FORMALDEHYDE for columns name uniformaity in Capital letter
daily_Formaldehyde_df = daily_Formaldehyde_df.rename(columns={'Formaldehyde': 'FORMALDEHYDE'})

# Selecting and order the desired columns
final_daily_df = daily_Formaldehyde_df[[
    'AMA_SITE_CODE', 'SAMPLE_DATE', 'FORMALDEHYDE', 'MONITOR_LATITUDE', 'MONITOR_LONGITUDE'
]]

# Sort by SAMPLE_DATE
final_daily_df = final_daily_df.sort_values(by='SAMPLE_DATE', ascending=True)

print(final_daily_df.head())
print(final_daily_df.tail())
```

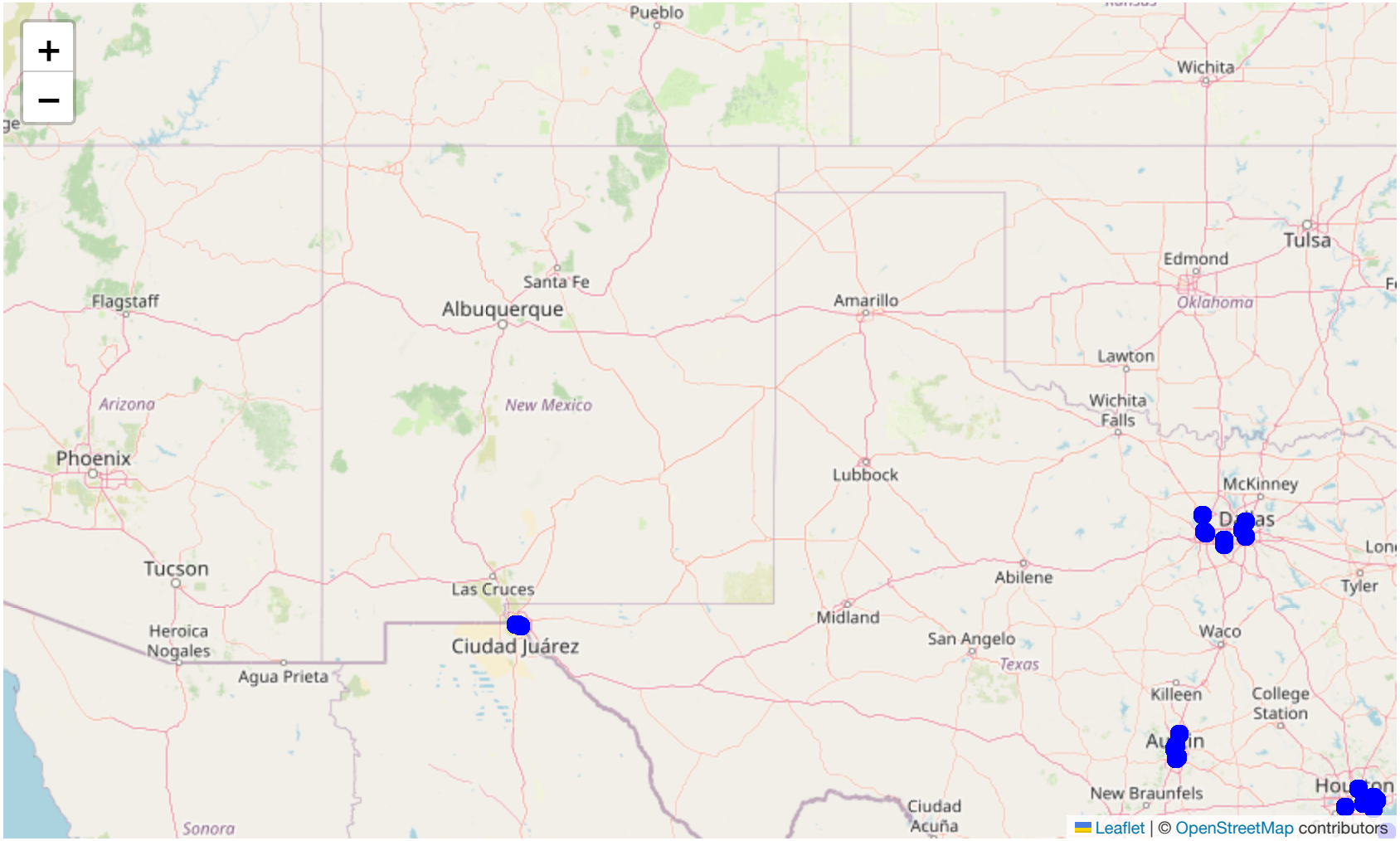
```
     AMA_SITE_CODE SAMPLE_DATE  FORMALDEHYDE  MONITOR_LATITUDE  \
4603     482011034  1992-07-23     12.000000         29.767998
4604     482011034  1992-07-26     14.670000         29.767998
4605     482011034  1992-07-29     17.490000         29.767998
4606     482011034  1992-08-01     29.219999         29.767998
4607     482011034  1992-08-04     12.890000         29.767998

      MONITOR_LONGITUDE
4603         -95.220581
4604         -95.220581
4605         -95.220581
4606         -95.220581
4607         -95.220581
    AMA_SITE_CODE SAMPLE_DATE  FORMALDEHYDE  MONITOR_LATITUDE  \
154     484391002  2022-10-20         3.970         32.805817
128     482011039  2022-10-20         0.200         29.670025
129     482011039  2022-10-26         0.372         29.670025
45      481130069  2022-10-26         4.191         32.820061
155     484391002  2022-10-26         2.996         32.805817

     MONITOR_LONGITUDE
154         -97.356567
128         -95.128510
129         -95.128510
45          -96.860115
155         -97.356567
```

In [69]:
```
# Exporting
# Defining export path
output_path = "/Users/nawarajadhikari/Capstone_Project/Formaldehyde_daily_avg.csv"

# Export to CSV
final_daily_df.to_csv(output_path, index=False)

print(f"Fomaldehyde dataset saved to: {output_path}")
```

```
Fomaldehyde dataset saved to: /Users/nawarajadhikari/Capstone_Project/Formaldehyde_daily_avg.csv
```

State Boundary Map

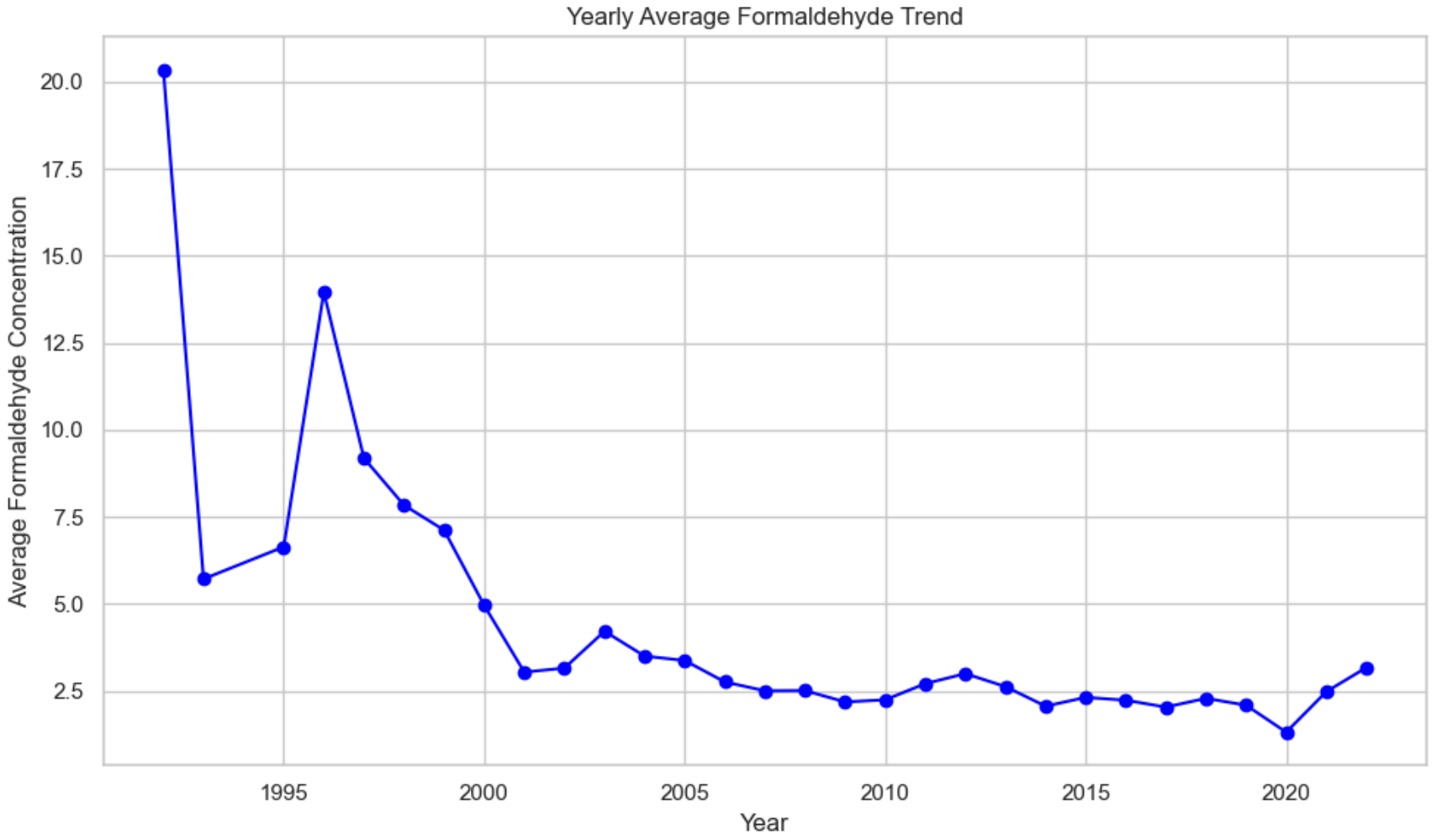
In [35]:
```
# Loading the CSV file
df = Formaldehyde_full_df

# Cleaning the column names and ensuring coordinates are numeric
df.columns = df.columns.str.strip()
df['MONITOR_LATITUDE'] = pd.to_numeric(df['MONITOR_LATITUDE'], errors='coerce')
df['MONITOR_LONGITUDE'] = pd.to_numeric(df['MONITOR_LONGITUDE'], errors='coerce')
df = df.dropna(subset=['MONITOR_LATITUDE', 'MONITOR_LONGITUDE'])

# Center the map around Texas
m = folium.Map(location=[31.5, -99.0], zoom_start=6)

# Monitoring sites
for _, row in df.iterrows():
    folium.CircleMarker(
        location=[row['MONITOR_LATITUDE'], row['MONITOR_LONGITUDE']],
        radius=4,
        color='Blue',
        fill=True,
        fill_opacity=0.6,
        popup=f"Site: {row.get('AMA_SITE_CODE', 'Unknown')}"
    ).add_to(m)

# Displaying the map
display(m)
```



11. Year Over Year Trend of Formaldehyde

In [31]:
```
# Loading the dataset
file_path = "/Users/nawarajadhikari/Capstone_Project/Formaldehyde_daily_avg.csv"
df = pd.read_csv(file_path)

# Stripping column names of extra spaces
df.columns = df.columns.str.strip()

# Converting SAMPLE_DATE to datetime and Formaldehyde to numeric
df['SAMPLE_DATE'] = pd.to_datetime(df['SAMPLE_DATE'], errors='coerce')
df['FORMALDEHYDE'] = pd.to_numeric(df['FORMALDEHYDE'], errors='coerce')
```

```
# Dropping rows with missing values in those two columns
df = df.dropna(subset=['SAMPLE_DATE', 'FORMALDEHYDE'])

# Creating a 'Year' column
df['Year'] = df['SAMPLE_DATE'].dt.year

# Group by year and calculate average
yearly_avg = df.groupby('Year')['FORMALDEHYDE'].mean().reset_index()

# Columns to clean numeric arrays
x = yearly_avg['Year'].astype(int).values
y = yearly_avg['FORMALDEHYDE'].astype(float).values

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(x, y, marker='o', linestyle='-', color='blue')
plt.title("Yearly Average Formaldehyde Trend")
plt.xlabel("Year")
plt.ylabel("Average Formaldehyde Concentration")
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [32]:
```
df.shape
```

Out[32]:
```
(10248, 6)
```

12. Long term trend of Formaldehyde

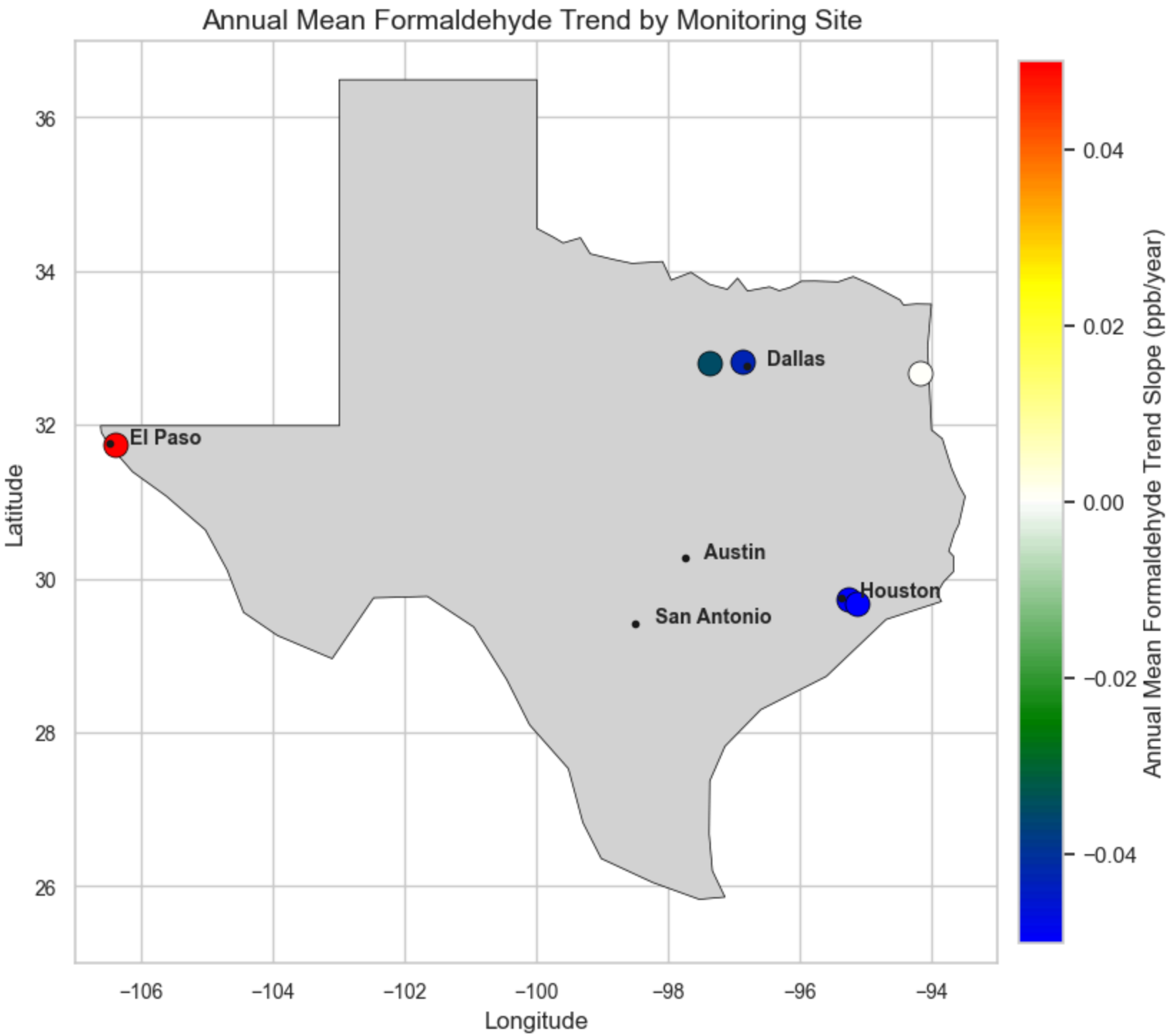
In [ ]:
```
# ----------------------------
# 1. City coordinates for labeling
# ----------------------------
cities = {
    "Houston": (29.7604, -95.3698),
```

```
    "Dallas": (32.7767, -96.7970),
    "Austin": (30.2672, -97.7431),
    "San Antonio": (29.4241, -98.4936),
    "El Paso": (31.7619, -106.4850)
}

# ----------------------------
# 2. Colormap settings
# ----------------------------
varmin = -0.05
varmax = 0.05
colors = ["blue", "green", "white", "yellow", "red"]
cmap = LinearSegmentedColormap.from_list("bgyr", colors)
norm = plt.Normalize(vmin=varmin, vmax=varmax)
sm = plt.cm.ScalarMappable(cmap=cmap, norm=norm)
sm.set_array([])

# ----------------------------
# 3. Loading and preprocessing data
# ----------------------------
df = pd.read_csv('/Users/nawarajadhikari/Capstone_Project/Formaldehyde_daily_avg.csv')

# Converting SAMPLE_DATE to datetime
df['SAMPLE_DATE'] = pd.to_datetime(df['SAMPLE_DATE'], errors='coerce')

# Dropping rows with missing Formaldehyde values
df.dropna(subset=['FORMALDEHYDE'], inplace=True)

# Extract year from SAMPLE_DATE
df['Year'] = df['SAMPLE_DATE'].dt.year

# ----------------------------
# 4. Defining site selection criteria
# ----------------------------
def meets_criteria(site_df):
    years = site_df['Year'].unique()
    return len(years) >= 4 and years.min() <= 2011 and years.max() >= 2015

# ----------------------------
# 5. Calculation of  trend (slope) for annual mean Formaldehyde per site
# ----------------------------
site_trends_mean = {}

for site_id, site_data in df.groupby('AMA_SITE_CODE'):
    if meets_criteria(site_data):
```

```
        annual_mean = site_data.groupby('Year')['FORMALDEHYDE'].mean().reset_index()
        years = annual_mean['Year'].values
        benzene_mean = annual_mean['FORMALDEHYDE'].values
        if len(years) >= 2:
            slope, _, _, _, _ = linregress(years, benzene_mean)
            lat = site_data['MONITOR_LATITUDE'].iloc[0]
            lon = site_data['MONITOR_LONGITUDE'].iloc[0]
            site_trends_mean[site_id] = {'slope': slope, 'lat': lat, 'lon': lon}
            print(f"Site ID: {site_id}, Annual Mean Trend Slope: {slope:.6f}")

# ----------------------------
# 6. Defining map boundaries
# ----------------------------
min_lon, max_lon = -107, -93
min_lat, max_lat = 25, 37

# ----------------------------
# 7. Loading shapefile
# ----------------------------
# Built-in naturalearth_lowres was not suportvie in this environmnet  and downloaded the natural earth file

# With this:
shapefile_path = '/Users/nawarajadhikari/Capstone_Project/ne_110m_admin_1_states_provinces'
states = gpd.read_file(shapefile_path)

# Now filtering for Texas (USA)
texas_shape = states[states['name'] == 'Texas']


# ----------------------------
# 8. Plotting setup
# ----------------------------
extent = max_lon - min_lon
aspect = np.cos(np.mean([min_lat, max_lat]) * np.pi / 180)
fig_height = 10
fig_width = fig_height * (extent / (max_lat - min_lat)) * aspect

fig, ax = plt.subplots(1, 1, figsize=(fig_width, fig_height))
ax.set_aspect('equal', adjustable='box')

# Plot base map (gray USA outline)
texas_shape.plot(ax=ax, color='lightgray', edgecolor='black', linewidth=0.5, zorder=1)

# Plot Formaldehyde trend points
for site_id, info in site_trends_mean.items():
```

```
    slope = info['slope']
    lat = info['lat']
    lon = info['lon']
    color = cmap(norm(slope))
    ax.scatter(lon, lat, s=150, c=[color], edgecolor='black', linewidth=0.5, zorder=2)

# ----------------------------
# 9. Plot cities
# ----------------------------
for city, (lat, lon) in cities.items():
    ax.plot(lon, lat, 'ko', markersize=3)
    ax.text(lon + 0.3, lat, city, fontsize=10, weight='bold')

# ----------------------------
# 10. Final formatting
# ----------------------------
ax.set_xlim(min_lon, max_lon)
ax.set_ylim(min_lat, max_lat)
ax.set_xlabel('Longitude', fontsize=12)
ax.set_ylabel('Latitude', fontsize=12)
ax.tick_params(axis='both', labelsize=10)
plt.title('Annual Mean Formaldehyde Trend by Monitoring Site ', fontsize=14)

# Adding colorbar
cbar = fig.colorbar(sm, ax=ax, orientation="vertical", pad=0.02, shrink=0.8)
cbar.set_label('Annual Mean Formaldehyde Trend Slope (ppb/year)', fontsize=12)

# Save the figure
plt.savefig('annual_mean_formaldehyde_trend_map.png', bbox_inches='tight')
print("✅ Annual mean formaldehyde trend map saved as 'annual_mean_formaldehyde_trend_map.png'")
```

```
Site ID: 481130069, Annual Mean Trend Slope: -0.042628
Site ID: 481410055, Annual Mean Trend Slope: 0.053055
Site ID: 482011035, Annual Mean Trend Slope: -0.201951
Site ID: 482011039, Annual Mean Trend Slope: -0.168722
Site ID: 482030002, Annual Mean Trend Slope: 0.000194
Site ID: 484391002, Annual Mean Trend Slope: -0.035107
✅ Annual mean formaldehyde trend map saved as 'annual_mean_formaldehyde_trend_map.png'
```


Annual Mean Formaldehyde Trend by Monitoring Site
Dallas
El Paso
Austin
Houston
San Antonio
Latitude
Longitude
36
34
32
30
28
26
−106
−104
−102
−100
−98
−96
−94
0.04
0.02
0.00
−0.02
−0.04
Annual Mean Formaldehyde Trend Slope (ppb/year)

13. Yearly Correlations Analysis of Benzene and Formaldehyde

In [5]:
```
# Datasets
benzene_df = pd.read_csv("/Users/nawarajadhikari/Capstone_Project/File3.csv")
formaldehyde_df = pd.read_csv("/Users/nawarajadhikari/Capstone_Project/Formaldehyde_daily_avg.csv")

# Converting SAMPLE_DATE columns to datetime
benzene_df['SAMPLE_DATE'] = pd.to_datetime(benzene_df['SAMPLE_DATE'], errors='coerce')
formaldehyde_df['SAMPLE_DATE'] = pd.to_datetime(formaldehyde_df['SAMPLE_DATE'], errors='coerce')

# Preview the structure of both datasets to understand how to merge them
benzene_df.head(), formaldehyde_df.head()
```

Out[5]:
```
(  AMA_SITE_CODE SAMPLE_DATE  BENZENE  MONITOR_LATITUDE  MONITOR_LONGITUDE
 0     482011034  1990-01-03     3.17         29.767998         -95.220581
 1     482011034  1990-01-11     2.59         29.767998         -95.220581
 2     481130069  1990-01-11     0.80         32.820061         -96.860115
 3     482011034  1990-01-17     1.58         29.767998         -95.220581
 4     482011034  1990-01-23     7.56         29.767998         -95.220581,
   AMA_SITE_CODE SAMPLE_DATE  FORMALDEHYDE  MONITOR_LATITUDE  MONITOR_LONGITUDE
 0     482011034  1992-07-23     12.000000         29.767998         -95.220581
 1     482011034  1992-07-26     14.670000         29.767998         -95.220581
 2     482011034  1992-07-29     17.490000         29.767998         -95.220581
 3     482011034  1992-08-01     29.219999         29.767998         -95.220581
 4     482011034  1992-08-04     12.890000         29.767998         -95.220581)
```

In [17]:
```
# Merging the datasets on AMA_SITE_CODE and SAMPLE_DATE
merged_df = pd.merge(
    benzene_df,
    formaldehyde_df,
    on=['AMA_SITE_CODE', 'SAMPLE_DATE'],
    suffixes=('_BENZENE', '_FORMALDEHYDE')
)

# Add YEAR column
merged_df['YEAR'] = merged_df['SAMPLE_DATE'].dt.year

# Initializing dictionary to store correlation results per site
site_correlation = {}

# Calculating yearly correlation per site
for site_id, site_data in merged_df.groupby('AMA_SITE_CODE'):
    yearly_corrs = []
```

```
    for year, year_data in site_data.groupby('YEAR'):
        if len(year_data) >= 2:
            corr = year_data['BENZENE'].corr(year_data['FORMALDEHYDE'])
            if pd.notna(corr):
                yearly_corrs.append(corr)
    # Storing mean correlation and lat/lon if sufficient data exists
    if yearly_corrs:
        mean_corr = sum(yearly_corrs) / len(yearly_corrs)
        lat = site_data['MONITOR_LATITUDE_BENZENE'].iloc[0]
        lon = site_data['MONITOR_LONGITUDE_BENZENE'].iloc[0]
        site_correlation[site_id] = {'mean_corr': mean_corr, 'lat': lat, 'lon': lon}

# Converting to DataFrame
correlation_df = pd.DataFrame([
    {'AMA_SITE_CODE': sid, 'CORRELATION': info['mean_corr'], 'LAT': info['lat'], 'LON': info['lon']}
    for sid, info in site_correlation.items()
])

correlation_df
```

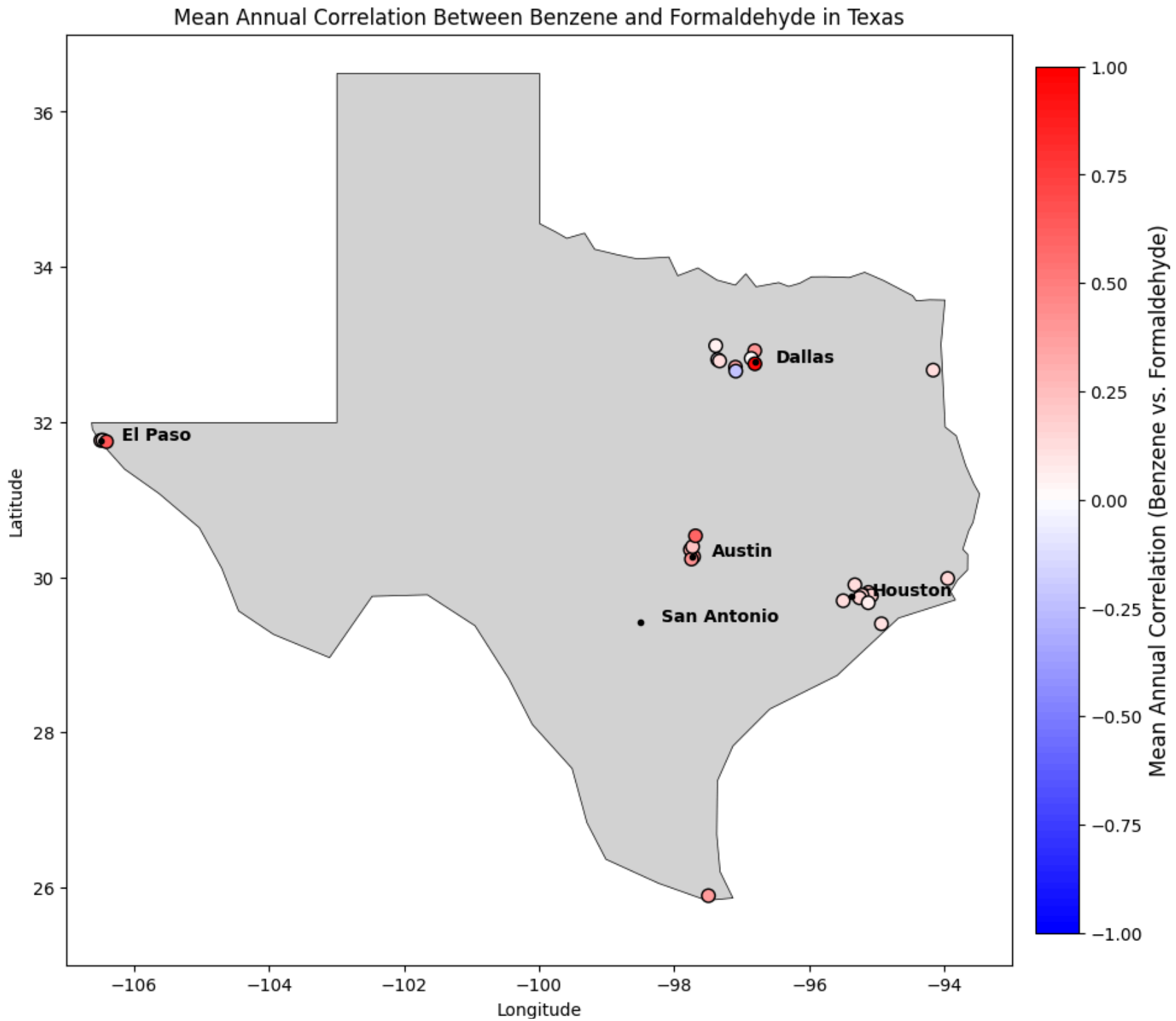
Out[17]:

	AMA_SITE_CODE	CORRELATION	LAT	LON
0	480610006	0.385342	25.892517	-97.493828
1	481130045	0.405996	32.919724	-96.808060
2	481130069	0.002740	32.820061	-96.860115
3	481135502	0.959519	32.750557	-96.805801
4	481410027	0.213804	31.763077	-106.486710
5	481410044	0.076832	31.765684	-106.455230
6	481410055	0.656403	31.746775	-106.402810
7	481671002	0.111751	29.398611	-94.933334
8	482010024	0.134042	29.901035	-95.326134
9	482010026	0.192123	29.802708	-95.125496
10	482010055	0.129061	29.695728	-95.499222
11	482010803	0.264001	29.764788	-95.178535
12	482011015	0.161376	29.761654	-95.081383
13	482011034	0.175177	29.767998	-95.220581
14	482011035	0.152121	29.733727	-95.257591
15	482011039	0.060262	29.670025	-95.128510
16	482030002	0.124683	32.668987	-94.167458
17	482450017	0.151024	29.982531	-93.952866
18	484390057	0.325704	32.706944	-97.093613
19	484391002	-0.047608	32.805817	-97.356567
20	484393011	-0.228642	32.656357	-97.088585
21	48439LS04	0.125088	32.787701	-97.328583
22	48439LS05	0.046489	32.984100	-97.385597
23	484537000	0.431324	30.263201	-97.713097
24	484537001	0.314272	30.354401	-97.760201
25	484537002	0.420717	30.232201	-97.744400

	AMA_SITE_CODE	CORRELATION	LAT	LON
26	484537003	0.226562	30.392599	-97.728500
27	484917004	0.597900	30.532600	-97.684898

In [21]:
```
# === 1. Creating GeoDataFrame from correlation_df ===
geometry = [Point(xy) for xy in zip(correlation_df['LON'], correlation_df['LAT'])]
correlation_gdf = gpd.GeoDataFrame(correlation_df, geometry=geometry, crs="EPSG:4326")

# === 3. Loading Texas shapefile from local path ===
shapefile_path = '/Users/nawarajadhikari/Capstone_Project/ne_110m_admin_1_states_provinces.shp'
states = gpd.read_file(shapefile_path)

# Filtering to Texas only
texas_shape = states[states['name'] == 'Texas']

# === 4. Plotting setup ===
varmin, varmax = -1, 1
colors = ["blue", "white", "red"]
cmap = LinearSegmentedColormap.from_list("bwr_custom", colors)
norm = plt.Normalize(vmin=varmin, vmax=varmax)
sm = plt.cm.ScalarMappable(cmap=cmap, norm=norm)
sm.set_array([])

# Plot setup
fig, ax = plt.subplots(1, 1, figsize=(10, 12))
ax.set_aspect('equal')

# Plotting Texas shape
texas_shape.plot(ax=ax, color='lightgray', edgecolor='black', linewidth=0.5, zorder=1)

# Plotting correlation points
correlation_gdf.plot(
    ax=ax,
    column='CORRELATION',
    cmap=cmap,
    markersize=60,
    vmin=varmin,
    vmax=varmax,
    edgecolor='black',
    legend=False,
    zorder=2
)
```

```
# Adding major cities
cities = {
    "Houston": (29.7604, -95.3698),
    "Dallas": (32.7767, -96.7970),
    "Austin": (30.2672, -97.7431),
    "San Antonio": (29.4241, -98.4936),
    "El Paso": (31.7619, -106.4850)
}

for city, (lat, lon) in cities.items():
    ax.plot(lon, lat, 'ko', markersize=3)
    ax.text(lon + 0.3, lat, city, fontsize=10, weight='bold')

# Final formatting
ax.set_xlim(-107, -93)
ax.set_ylim(25, 37)
ax.set_xlabel("Longitude")
ax.set_ylabel("Latitude")
ax.set_title("Mean Annual Correlation Between Benzene and Formaldehyde in Texas")

# Add colorbar
cbar = fig.colorbar(sm, ax=ax, orientation="vertical", pad=0.02, shrink=0.6)
cbar.set_label("Mean Annual Correlation (Benzene vs. Formaldehyde)", fontsize=12)

plt.grid(False)
plt.tight_layout()
plt.savefig("correlation_map_texas.png", bbox_inches='tight')
plt.show()

print("✅ Correlation map saved as 'correlation_map_texas.png'")
```


Mean Annual Correlation Between Benzene and Formaldehyde in Texas
Dallas
El Paso
Austin
Houston
San Antonio
Latitude
Longitude
36
34
32
30
28
26
−106
−104
−102
−100
−98
−96
−94
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Mean Annual Correlation (Benzene vs. Formaldehyde)

```
✅ Correlation map saved as 'correlation_map_texas.png'
```

14. Proximity Analysis

In [7]:
```
# Help see how pollutant levels vary with distance from potential sources like oil/gas wells or industrial si
```

In [8]:
```
# Inspecting the Columns
print("Benzene columns:", benzene_df.columns)
print("Formaldehyde columns:", formaldehyde_df.columns)

benzene_df.head()
formaldehyde_df.head()
```

```
Benzene columns: Index(['AMA_SITE_CODE', 'SAMPLE_DATE', 'BENZENE', 'MONITOR_LATITUDE',
       'MONITOR_LONGITUDE'],
      dtype='object')
Formaldehyde columns: Index(['AMA_SITE_CODE', 'SAMPLE_DATE', 'FORMALDEHYDE', 'MONITOR_LATITUDE',
       'MONITOR_LONGITUDE'],
      dtype='object')
```

Out[8]:

	AMA_SITE_CODE	SAMPLE_DATE	FORMALDEHYDE	MONITOR_LATITUDE	MONITOR_LONGITUDE
0	482011034	1992-07-23	12.000000	29.767998	-95.220581
1	482011034	1992-07-26	14.670000	29.767998	-95.220581
2	482011034	1992-07-29	17.490000	29.767998	-95.220581
3	482011034	1992-08-01	29.219999	29.767998	-95.220581
4	482011034	1992-08-04	12.890000	29.767998	-95.220581

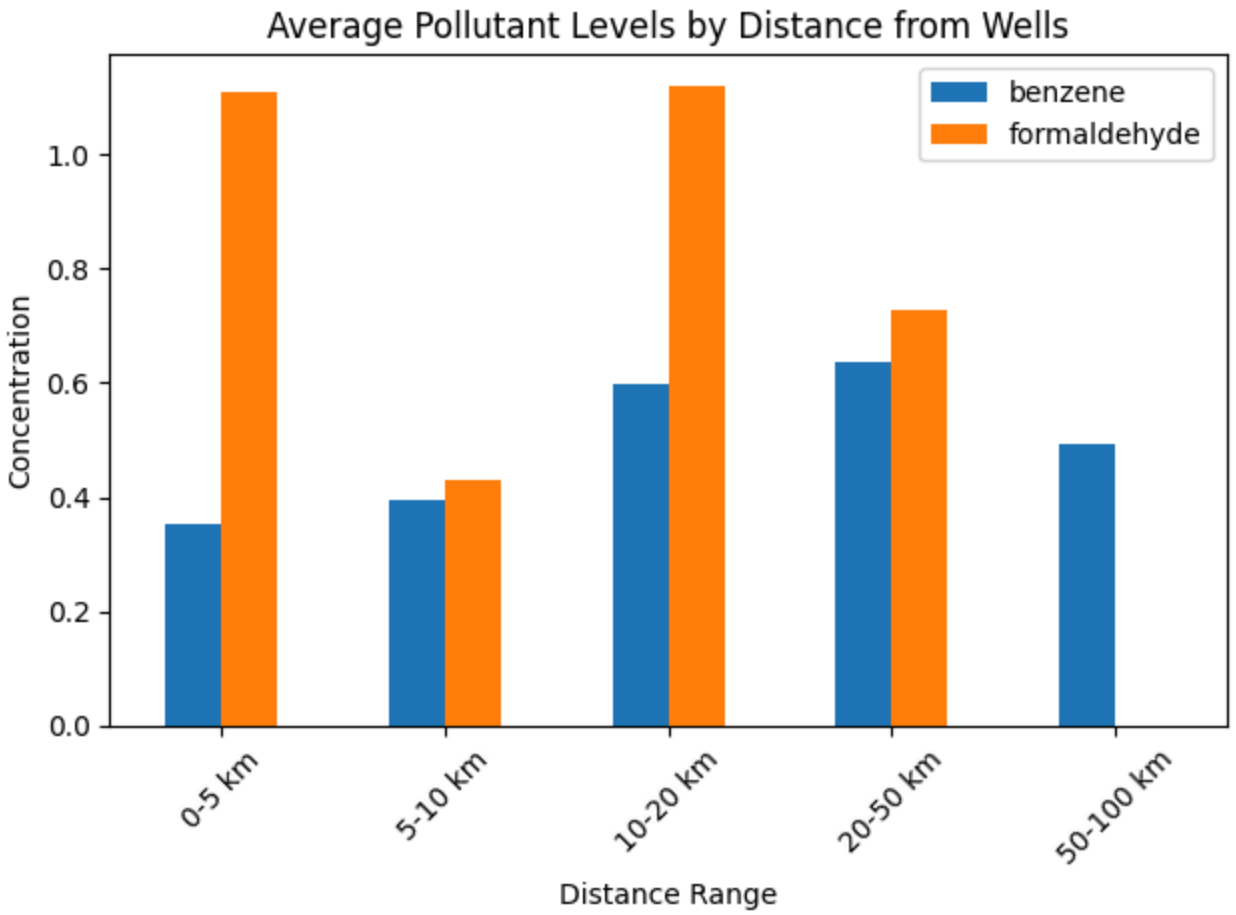
In [14]:
```
import pandas as pd
import numpy as np
from geopy.distance import geodesic
import matplotlib.pyplot as plt

# ---------------------------
# STEP 1: Aggregate Pollutants by Site
# ---------------------------

# Rename columns for consistency
benzene_df.rename(columns={
```

```
    'BENZENE': 'benzene',
    'MONITOR_LATITUDE': 'lat',
    'MONITOR_LONGITUDE': 'lon',
    'SAMPLE_DATE': 'date'
}, inplace=True)

formaldehyde_df.rename(columns={
    'FORMALDEHYDE': 'formaldehyde',
    'MONITOR_LATITUDE': 'lat',
    'MONITOR_LONGITUDE': 'lon',
    'SAMPLE_DATE': 'date'
}, inplace=True)

# Group by site and calculate average concentrations
benzene_avg = benzene_df.groupby('AMA_SITE_CODE').agg({
    'benzene': 'mean',
    'lat': 'first',
    'lon': 'first'
}).reset_index()

formaldehyde_avg = formaldehyde_df.groupby('AMA_SITE_CODE').agg({
    'formaldehyde': 'mean'
}).reset_index()

# Merge both pollutants
pollutants_df = pd.merge(
    benzene_avg,
    formaldehyde_avg,
    on='AMA_SITE_CODE',
    how='outer'
)

# Drop rows without coordinates
pollutants_df.dropna(subset=['lat', 'lon'], inplace=True)

# Fill missing pollutant values with 0
pollutants_df[['benzene', 'formaldehyde']] = pollutants_df[['benzene', 'formaldehyde']].fillna(0)

print("Pollutants dataset after aggregation:")
print(pollutants_df.head())

# ---------------------------
# STEP 2: Simulate Wells
# ---------------------------
```

```
np.random.seed(42)
num_wells = 500
wells_df = pd.DataFrame({
    'lat': np.random.uniform(25, 36, num_wells),
    'lon': np.random.uniform(-107, -93, num_wells),
    'well_type': np.random.choice(['oil', 'gas', 'oil & gas', 'dry'], num_wells)
})

print("\nSimulated wells dataset:")
print(wells_df.head())

# ---------------------------
# STEP 3: Compute Distance to Nearest Well
# ---------------------------

def nearest_well(lat, lon, wells):
    distances = wells.apply(lambda row: geodesic((lat, lon), (row['lat'], row['lon'])).km, axis=1)
    min_idx = distances.idxmin()
    return distances[min_idx], wells.loc[min_idx, 'well_type']

pollutants_df[['distance_km', 'nearest_well_type']] = pollutants_df.apply(
    lambda row: pd.Series(nearest_well(row['lat'], row['lon'], wells_df)), axis=1
)

print("\nPollutants dataset with distance and nearest well type:")
print(pollutants_df.head())

# ---------------------------
# STEP 4: Proximity Analysis (Distance Bins)
# ---------------------------

bins = [0, 5, 10, 20, 50, 100]
labels = ['0-5 km', '5-10 km', '10-20 km', '20-50 km', '50-100 km']
pollutants_df['distance_bin'] = pd.cut(pollutants_df['distance_km'], bins=bins, labels=labels)

proximity_analysis = pollutants_df.groupby('distance_bin')[['benzene', 'formaldehyde']].mean()
print("\nAverage pollutant levels by distance bins:")
print(proximity_analysis)

# ---------------------------
# STEP 5: Visualization
# ---------------------------

proximity_analysis.plot(kind='bar')
plt.title('Average Pollutant Levels by Distance from Wells')
```

```
plt.ylabel('Concentration')
plt.xlabel('Distance Range')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Pollutants dataset after aggregation:
  AMA_SITE_CODE   benzene        lat        lon  formaldehyde
0     480055501  0.214926  31.179911 -94.790489           0.0
1     480290051  0.413368  29.445045 -98.406502           0.0
2     480290052  0.057516  29.632057 -98.564934           0.0
3     480290677  0.259217  29.423944 -98.580498           0.0
4     480391003  0.300138  29.010841 -95.397743           0.0

Simulated wells dataset:
         lat         lon  well_type
0  29.119941  -97.225736  oil & gas
1  35.457857  -99.494651        dry
2  33.051933 -102.666613        gas
3  31.585243  -95.606870        dry
4  26.716205  -97.413764        dry

Pollutants dataset with distance and nearest well type:
  AMA_SITE_CODE   benzene        lat        lon  formaldehyde  distance_km  \
0     480055501  0.214926  31.179911 -94.790489           0.0    30.751770
1     480290051  0.413368  29.445045 -98.406502           0.0    31.182822
2     480290052  0.057516  29.632057 -98.564934           0.0    21.243903
3     480290677  0.259217  29.423944 -98.580498           0.0    16.032860
4     480391003  0.300138  29.010841 -95.397743           0.0     7.467758

  nearest_well_type
0               dry
1               gas
2               gas
3               gas
4               dry

Average pollutant levels by distance bins:
               benzene  formaldehyde
distance_bin
0-5 km        0.352071      1.106882
5-10 km       0.395472      0.430766
10-20 km      0.598607      1.117951
20-50 km      0.635015      0.728881
50-100 km     0.494101      0.000000
```



15. Kruskal-Wallis Test

- To check if pollutant concentrations significantly differ across distance bins, we'll use Kruskal-Wallis H-test (non-parametric, suitable for unequal sample sizes and non-normal data).

In [17]:
```
from scipy.stats import kruskal

# Group Benzene values by distance bins
benzene_groups = [group['benzene'].dropna().values for name, group in pollutants_df.groupby('distance_bin')]
formaldehyde_groups = [group['formaldehyde'].dropna().values for name, group in pollutants_df.groupby('distan

# Kruskal-Wallis test for Benzene
```

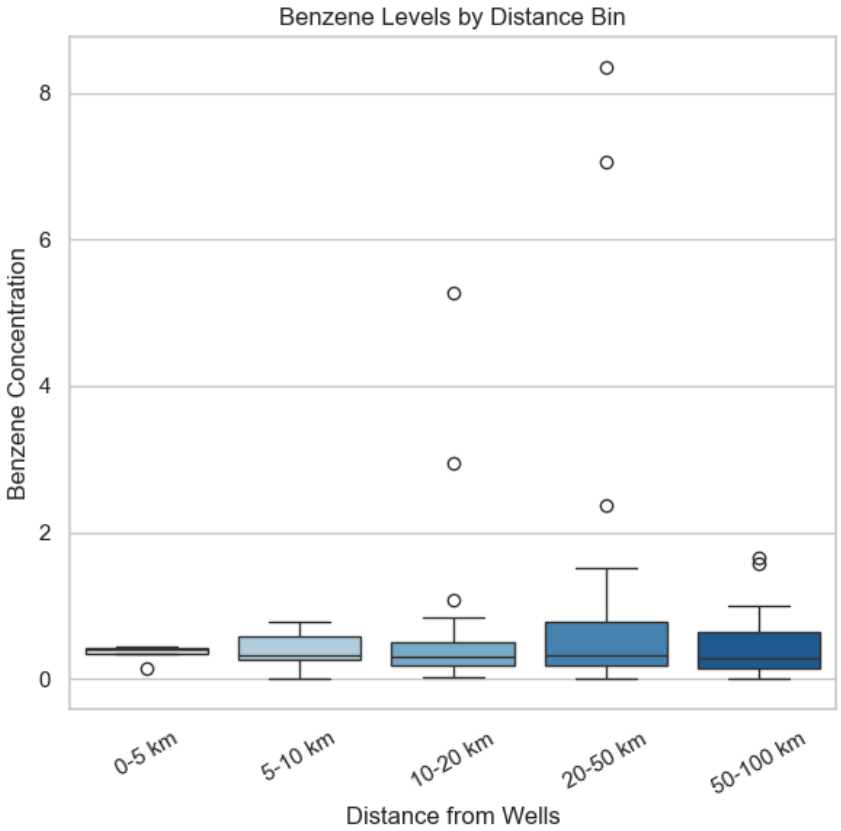
```
stat_benzene, p_benzene = kruskal(*benzene_groups)
print(f"Benzene: H-statistic={stat_benzene:.3f}, p-value={p_benzene:.5f}")

# Kruskal-Wallis test for Formaldehyde
stat_formal, p_formal = kruskal(*formaldehyde_groups)
print(f"Formaldehyde: H-statistic={stat_formal:.3f}, p-value={p_formal:.5f}")

# Interpretation
alpha = 0.05
if p_benzene < alpha:
    print("Benzene: Significant difference between at least two distance bins.")
else:
    print("Benzene: No significant difference between bins.")

if p_formal < alpha:
    print("Formaldehyde: Significant difference between at least two distance bins.")
else:
    print("Formaldehyde: No significant difference between bins.")
```

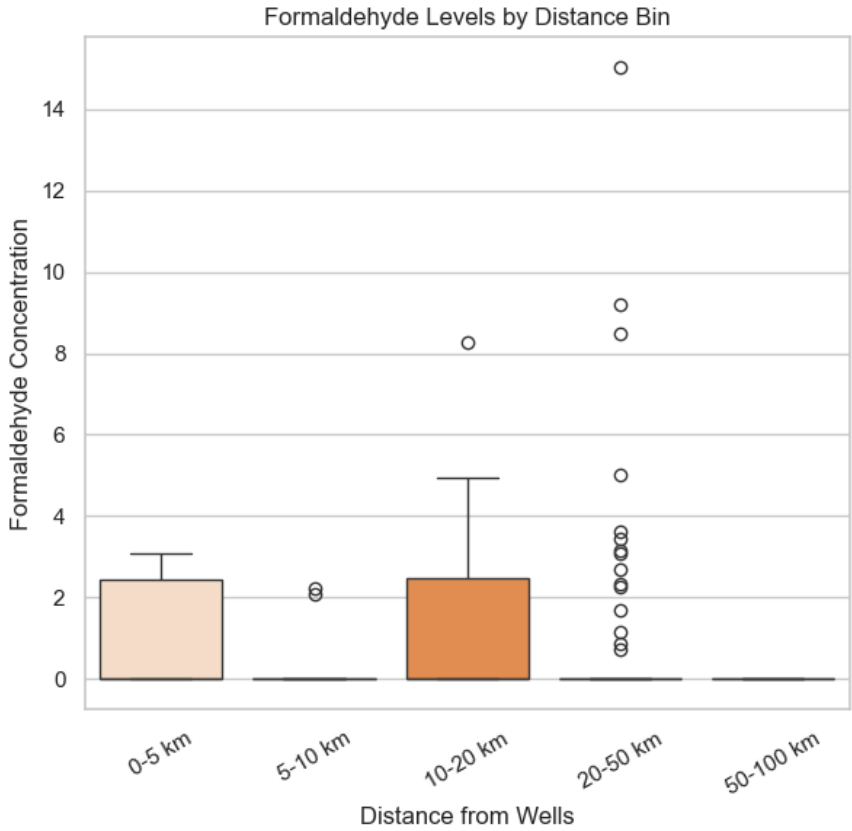
```
Benzene: H-statistic=0.209, p-value=0.99488
Formaldehyde: H-statistic=7.334, p-value=0.11926
Benzene: No significant difference between bins.
Formaldehyde: No significant difference between bins.
```

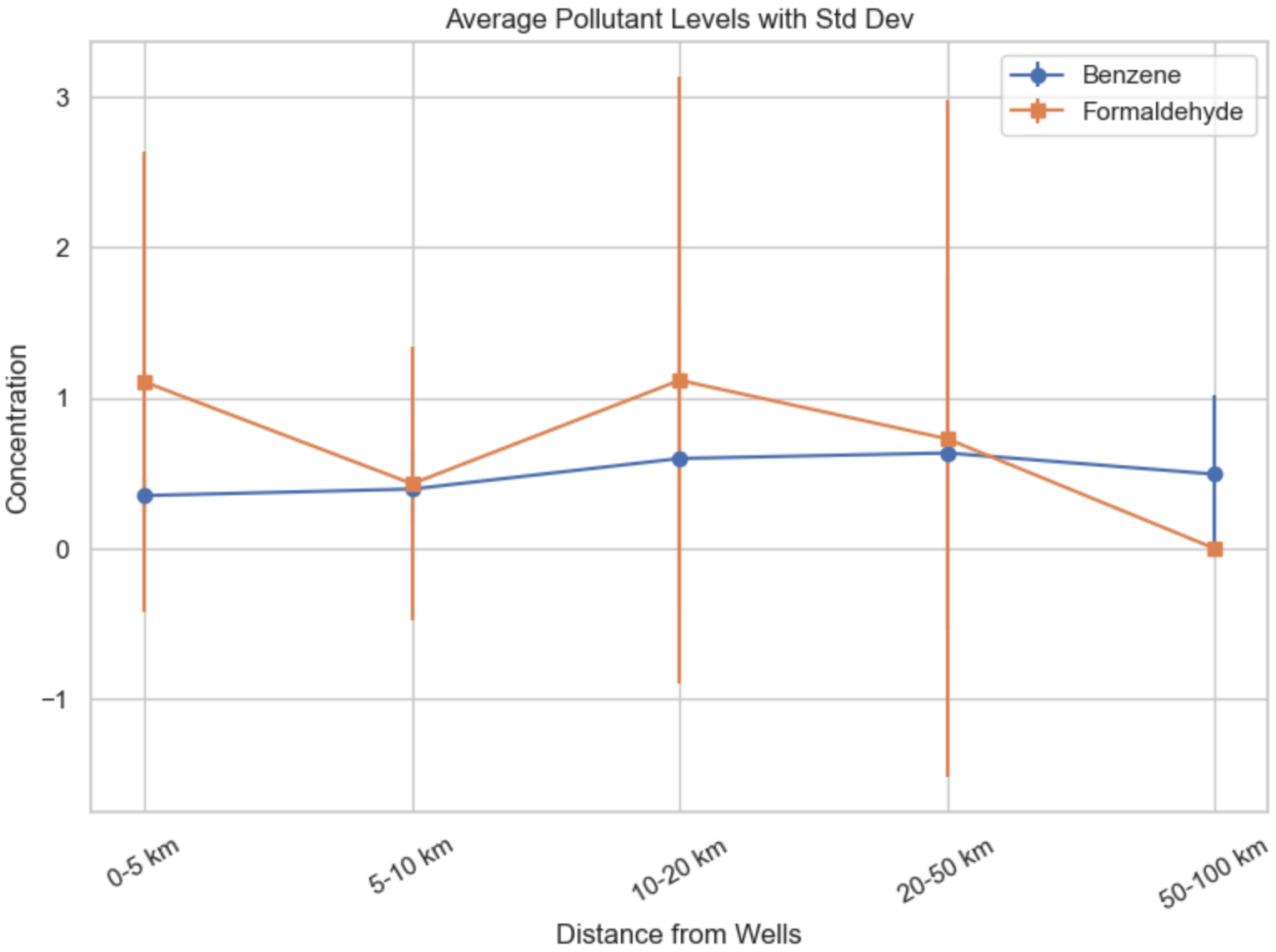
In [18]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Ensure seaborn uses a clean theme
sns.set(style="whitegrid")

# ------------------------------
# 1. Boxplots for Benzene & Formaldehyde
# ------------------------------
plt.figure(figsize=(12, 6))

# Benzene
plt.subplot(1, 2, 1)
sns.boxplot(x='distance_bin', y='benzene', data=pollutants_df, palette='Blues')
plt.title('Benzene Levels by Distance Bin')
plt.xlabel('Distance from Wells')
plt.ylabel('Benzene Concentration')
plt.xticks(rotation=30)

# Formaldehyde
plt.subplot(1, 2, 2)
```

```
sns.boxplot(x='distance_bin', y='formaldehyde', data=pollutants_df, palette='Oranges')
plt.title('Formaldehyde Levels by Distance Bin')
plt.xlabel('Distance from Wells')
plt.ylabel('Formaldehyde Concentration')
plt.xticks(rotation=30)

plt.tight_layout()
plt.show()

# ------------------------------
# 2. Mean Trend Plot with Error Bars
# ------------------------------
mean_df = pollutants_df.groupby('distance_bin')[['benzene', 'formaldehyde']].agg(['mean', 'std'])
mean_df.columns = ['benzene_mean', 'benzene_std', 'formaldehyde_mean', 'formaldehyde_std']
mean_df = mean_df.reset_index()

plt.figure(figsize=(8, 6))
plt.errorbar(mean_df['distance_bin'], mean_df['benzene_mean'], yerr=mean_df['benzene_std'], fmt='-o', label='
plt.errorbar(mean_df['distance_bin'], mean_df['formaldehyde_mean'], yerr=mean_df['formaldehyde_std'], fmt='-s
plt.title('Average Pollutant Levels with Std Dev')
plt.xlabel('Distance from Wells')
plt.ylabel('Concentration')
plt.xticks(rotation=30)
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```


Benzene Levels by Distance Bin
Benzene Concentration
0
2
4
6
8
0-5 km
5-10 km
10-20 km
20-50 km
50-100 km
Distance from Wells


Formaldehyde Levels by Distance Bin
Formaldehyde Concentration
0
2
4
6
8
10
12
14
0-5 km
5-10 km
10-20 km
20-50 km
50-100 km
Distance from Wells


Average Pollutant Levels with Std Dev
Benzene
Formaldehyde
Concentration
3
2
1
0
−1
0-5 km
5-10 km
10-20 km
20-50 km
50-100 km
Distance from Wells