The World Bank Carbon Pricing Data

Here's a breakdown of the sheet names:

1. Compliance_Gen Info: General information about compliance schemes.
2. Compliance_Emissions: Data related to emissions under compliance schemes.
3. Compliance_Revenue: Revenue generated from carbon pricing mechanisms.
4. Compliance_Price: Pricing data under compliance schemes.
5. Crediting_Detail: Details about crediting mechanisms.
6. Crediting_Issuance: Issuance data for carbon credits.
7. Cooperative Approaches: Data related to cooperative approaches across different regions.

Given the scope of cost prediction and emissions reduction, the most relevant sheets likely include:

1. Compliance_Price: Contains pricing data, which is key for analyzing the cost-effectiveness of carbon pricing and CCUS projects.
2. Compliance_Emissions: Emissions data, crucial for understanding the impact of pricing on emissions reduction.
3. Compliance_Revenue: Revenue data, useful for financial analysis of carbon pricing mechanisms.

Data Dictionary for World Bank Carbon Pricing Data (Compliance_Price Sheet)

Column	Description	Relevance to Project
`Name of the initiative`	The name of the carbon pricing initiative (e.g., Alberta Carbon Tax).	Identifies the carbon pricing schemes in different regions.
`Instrument Type`	Type of instrument used (e.g., Carbon Tax, Emissions Trading Scheme).	Important for understanding the pricing mechanism used by the jurisdiction.
`Jurisdiction Covered`	The country or region where the pricing mechanism is implemented.	Links to CCUS project data for regional analysis.
`Region`	The geographic region of the initiative (e.g., North America, Europe).	Useful for regional comparisons of carbon pricing mechanisms.

Column	Description	Relevance to Project
`Income group`	The income classification of the jurisdiction (e.g., High income).	Provides insights into how income level may correlate with carbon pricing.
`Start date`	The year the pricing mechanism was introduced.	Helps track when a pricing initiative started, important for trend analysis.
`Price rate label`	Specifies if the price is a single rate or varies by sector.	Helps in understanding the structure of carbon pricing across sectors.
`Metric`	The unit of measure for the price (typically USD/tCO_2e).	Provides a common measure for comparing the cost of carbon across regions.
`1990, 1991, ... 2024`	Year-specific columns indicating the price per ton of CO_2 in USD/tCO_2e.	Main variable for tracking how carbon pricing has changed over time.

Data Dictionary for Compliance_Emissions Sheet

Column	Description	Relevance to Project
`Name of the initiative`	The name of the carbon pricing initiative (e.g., Finland Carbon Tax).	Identifies carbon pricing schemes for emissions analysis.
`1990, 1991, ... 2024`	Share of global emissions covered for the corresponding year.	Tracks how much of the global emissions are covered by the pricing scheme each year.

Data Dictionary for Compliance_Revenue Sheet

Column	Description	Relevance to Project
`Name of the initiative`	The name of the carbon pricing initiative (e.g., Finland Carbon Tax).	Identifies carbon pricing schemes in different regions.
`Instrument Type`	Type of instrument used (e.g., Carbon Tax, Emissions Trading Scheme).	Understands the financial structure of the initiative.
`Jurisdiction Covered`	The country or region where the pricing mechanism is implemented.	Links to CCUS project data for regional analysis.
`Metric`	The unit of measure for revenue (typically in US$ millions).	Key for tracking the revenue generated by the carbon pricing mechanisms.
`1990, 1991, ... 2024`	Revenue generated from the carbon pricing mechanism for each corresponding year.	Important for analyzing the financial impact of carbon pricing over time.

Analytical Approach

Our analysis will follow a structured approach to ensure comprehensive insights into the cost and effectiveness of carbon pricing mechanisms:

1. **Data Understanding**:
    - We will explore each dataset, identifying key features and understanding their significance for the project.
    - This will include summary statistics and an overview of the main variables that will be used for analysis.
2. **Data Cleaning**:
    - We will clean the data by handling missing values, standardizing data formats, and ensuring consistency across the datasets.
3. **Exploratory Data Analysis (EDA)**:
    - Through visualization and statistical analysis, we will explore the relationships between carbon pricing, revenue generation, and emissions coverage across different regions and years.
4. **Correlation Analysis**:
    - We will analyze the correlation between carbon pricing, the share of emissions covered, and revenue generated to understand how effective these initiatives are in achieving emissions reductions.
5. **Conclusion & Insights**:
    - Finally, we will summarize our findings and highlight any key patterns or trends in the data.

In [1]:
```python
#Importing libraries
import pandas as pd

# Loading the datasets from the Excel file
file_path = 'World Bank Carbon Pricing Data.xlsx'

# Load each relevant sheet
compliance_price = pd.read_excel(file_path, sheet_name='Compliance_Price', header=1)
compliance_emissions = pd.read_excel(file_path, sheet_name='Compliance_Emissions', header=1)
compliance_revenue = pd.read_excel(file_path, sheet_name='Compliance_Revenue', header=1)

# Display the first few rows of each dataset to confirm the load
print("Compliance_Price:")
display(compliance_price.head())

print("Compliance_Emissions:")
display(compliance_emissions.head())

print("Compliance_Revenue:")
display(compliance_revenue.head())
```

```
Compliance_Price:
```

	Name of the initiative	Instrument Type	Jurisdiction Covered	Region	Income group	Start date	Price rate label	Metric	1990	1991	...	2015	2016	2017	2018	2019	
0	Albania Carbon tax	Carbon tax	Albania	Europe & Central Asia	Upper middle income	2008	Single price	US$/tCO2e	-	-	...	-	-	NaN	NaN	NaN	
1	Alberta carbon tax	Carbon tax	Alberta	North America	High income	2017	Single price	US$/tCO2e	-	-	...	-	-	15.026357	23.252205	22.493795	
2	Alberta TIER	ETS	Alberta	North America	High income	2007	Single price	US$/tCO2e	-	-	...	11.89	15.37	22.539536	23.252205	22.493795	21.
3	Argentina carbon tax	Carbon tax	Argentina	Latin America & Caribbean	Upper middle income	2018	Gasoline (Nafta over and under 92 RON)	US$/tCO2e	-	-	...	-	-	NaN	8.914348	6.187569	6.
4	Argentina carbon tax	Carbon tax	Argentina	Latin America & Caribbean	Upper middle income	2018	Natural gasoline	US$/tCO2e	-	-	...	-	-	NaN	10.129136	7.030770	7.

5 rows × 43 columns

```
Compliance_Emissions:
```

	Share of global emissions covered (accounting for overlap of coverage between instruments)	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 26
0	Name of the initiative	1990.000000	1991.000000	1992.000000	1993.000000	1994.000000	1995.000000	1996.000000	1997.000000	1998.000000	...	2015.000000
1	Finland carbon tax	0.001157	0.001162	0.001111	0.001128	0.001213	0.001129	0.001174	0.001145	0.001090	...	0.000638
2	Poland carbon tax	0.003712	0.003622	0.003528	0.003488	0.003403	0.003324	0.003354	0.003252	0.002974	...	0.001867
3	Norway carbon tax	NaN	0.001173	0.001189	0.001247	0.001303	0.001277	0.001294	0.001334	0.001347	...	0.000963
4	Sweden carbon tax	NaN	0.000950	0.000986	0.000987	0.001008	0.000976	0.001019	0.000954	0.000953	...	0.000532

5 rows × 36 columns

```
Compliance_Revenue:
```

	Name of the initiative	Instrument Type	Jurisdiction Covered	Metric	1990	1991	1992	1993	1994	1995	...	2014	201
0	Finland carbon tax	Carbon tax	Finland	US$ millions	160.891089	144.124168	111.683849	179.487179	0.000000	0.000000	...	1137.056975	1456.48252
1	Poland carbon tax	Carbon tax	Poland	US$ millions	2.065790	1.427680	1.124912	0.856440	0.771435	0.717332	...	1.222900	1.21144
2	Norway carbon tax	Carbon tax	Norway	US$ millions	0.000000	124.805513	280.827556	311.907280	417.033268	398.163615	...	1247.205259	1500.43596
3	Sweden carbon tax	Carbon tax	Sweden	US$ millions	0.000000	1408.962186	1205.533338	1362.281768	1522.388060	1713.432119	...	2704.366068	3046.35281
4	Denmark carbon tax	Carbon tax	Denmark	US$ millions	0.000000	0.000000	227.110390	484.298781	611.049724	561.188811	...	531.642289	567.73718

5 rows × 38 columns

Data Understanding

In this section, we will explore the contents of each dataset in detail. This will help us understand the data's structure, identify potential issues, and assess the quality of the data. The key goals of this section include:

1. Getting an overview of each dataset, including the number of rows and columns.
2. Summarizing the key statistics, including mean, median, and range for numerical features.
3. Analyzing missing data and identifying any potential issues with data quality.
4. Understanding the distribution of key variables to better prepare for the analysis.

We will proceed by examining the following datasets:

- **Compliance_Price**: Carbon pricing data over time for various jurisdictions.
- **Compliance_Emissions**: Share of global emissions covered by carbon pricing mechanisms.
- **Compliance_Revenue**: Revenue generated by carbon pricing initiatives over time.

Compliance_Price

In [2]:
```
# Compliance_Price Overview
print("Compliance_Price Dataset Overview:")
print(f"Number of Rows: {compliance_price.shape[0]}")
print(f"Number of Columns: {compliance_price.shape[1]}")
```

```
Compliance_Price Dataset Overview:
Number of Rows: 142
Number of Columns: 43
```

In [3]:
```
# Displaying column names
print("\nColumn Names:")
print(compliance_price.columns)
```

```
Column Names:
Index(['Name of the initiative',        'Instrument Type',
         'Jurisdiction Covered',                 'Region',
                 'Income group',             'Start date',
             'Price rate label',                 'Metric',
                           1990,                     1991,
                           1992,                     1993,
                           1994,                     1995,
                           1996,                     1997,
                           1998,                     1999,
                           2000,                     2001,
                           2002,                     2003,
                           2004,                     2005,
                           2006,                     2007,
                           2008,                     2009,
                           2010,                     2011,
                           2012,                     2013,
                           2014,                     2015,
                           2016,                     2017,
                           2018,                     2019,
                           2020,                     2021,
                           2022,                     2023,
                           2024],
      dtype='object')
```

In [4]:
```
# Checking for missing values
print("\nMissing Values in Compliance_Price:")
print(compliance_price.isnull().sum())
```

```
Missing Values in Compliance_Price:
Name of the initiative     0
Instrument Type            0
Jurisdiction Covered       0
Region                     0
Income group               0
Start date                 0
Price rate label           0
Metric                     0
1990                       0
1991                       0
1992                       0
1993                       0
1994                       0
1995                       0
1996                       0
1997                       0
1998                       0
1999                       0
2000                       0
2001                       0
2002                       0
2003                       0
2004                       0
2005                       0
2006                       0
2007                       0
2008                       0
2009                       0
2010                       0
2011                       0
2012                       0
2013                       0
2014                       0
2015                       0
2016                       0
2017                      77
2018                      68
2019                      57
2020                      49
2021                      43
2022                      36
2023                      30
2024                      30
dtype: int64
```

In [5]:

```
# Re-check for missing values in Compliance_Price
print("Missing Values in Compliance_Price before re-cleaning:")
print(compliance_price.isnull().sum())
```

```python
# Handle missing values in yearly columns by forward filling, followed by backward filling
yearly_columns = [col for col in compliance_price.columns if isinstance(col, int)]

# Fill missing values in the yearly columns with forward fill, followed by backward fill
compliance_price[yearly_columns] = compliance_price[yearly_columns].fillna(method='ffill')
compliance_price[yearly_columns] = compliance_price[yearly_columns].fillna(method='bfill')

# Check again for missing values
print("\nMissing Values in Compliance_Price after filling:")
print(compliance_price.isnull().sum())
```

```
Missing Values in Compliance_Price before re-cleaning:
Name of the initiative     0
Instrument Type            0
Jurisdiction Covered       0
Region                     0
Income group               0
Start date                 0
Price rate label           0
Metric                     0
1990                       0
1991                       0
1992                       0
1993                       0
1994                       0
1995                       0
1996                       0
1997                       0
1998                       0
1999                       0
2000                       0
2001                       0
2002                       0
2003                       0
2004                       0
2005                       0
2006                       0
2007                       0
2008                       0
2009                       0
2010                       0
2011                       0
2012                       0
2013                       0
2014                       0
2015                       0
2016                       0
2017                      77
2018                      68
2019                      57
2020                      49
2021                      43
2022                      36
2023                      30
2024                      30
dtype: int64

Missing Values in Compliance_Price after filling:
Name of the initiative    0
Instrument Type           0
Jurisdiction Covered      0
```

```
Region               0
Income group         0
Start date           0
Price rate label     0
Metric               0
1990                 0
1991                 0
1992                 0
1993                 0
1994                 0
1995                 0
1996                 0
1997                 0
1998                 0
1999                 0
2000                 0
2001                 0
2002                 0
2003                 0
2004                 0
2005                 0
2006                 0
2007                 0
2008                 0
2009                 0
2010                 0
2011                 0
2012                 0
2013                 0
2014                 0
2015                 0
2016                 0
2017                 0
2018                 0
2019                 0
2020                 0
2021                 0
2022                 0
2023                 0
2024                 0
dtype: int64
```

```
C:\Users\jigar\AppData\Local\Temp\ipykernel_33380\3691048044.py:9: FutureWarning: DataFrame.fillna with 'method' is deprecated and will rai
se in a future version. Use obj.ffill() or obj.bfill() instead.
  compliance_price[yearly_columns] = compliance_price[yearly_columns].fillna(method='ffill')
C:\Users\jigar\AppData\Local\Temp\ipykernel_33380\3691048044.py:10: FutureWarning: DataFrame.fillna with 'method' is deprecated and will ra
ise in a future version. Use obj.ffill() or obj.bfill() instead.
  compliance_price[yearly_columns] = compliance_price[yearly_columns].fillna(method='bfill')
```

Compliance_Emissions: Data Overview

The **Compliance_Emissions** sheet tracks the share of global emissions covered by different carbon pricing mechanisms over time. This dataset will help us analyze how much of global emissions are subject to carbon pricing initiatives and how this share has changed across different regions and years.

In [6]:
```
# Compliance_Emissions Overview
print("Compliance_Emissions Dataset Overview:")
print(f"Number of Rows: {compliance_emissions.shape[0]}")
print(f"Number of Columns: {compliance_emissions.shape[1]}")
```

```
Compliance_Emissions Dataset Overview:
Number of Rows: 87
Number of Columns: 36
```

In [7]:
```
# Displaying column names
print("\nColumn Names:")
print(compliance_emissions.columns)
```

```
Column Names:
Index(['Share of global emissions covered (accounting for overlap of coverage between instruments)',
       'Unnamed: 1', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4', 'Unnamed: 5',
       'Unnamed: 6', 'Unnamed: 7', 'Unnamed: 8', 'Unnamed: 9', 'Unnamed: 10',
       'Unnamed: 11', 'Unnamed: 12', 'Unnamed: 13', 'Unnamed: 14',
       'Unnamed: 15', 'Unnamed: 16', 'Unnamed: 17', 'Unnamed: 18',
       'Unnamed: 19', 'Unnamed: 20', 'Unnamed: 21', 'Unnamed: 22',
       'Unnamed: 23', 'Unnamed: 24', 'Unnamed: 25', 'Unnamed: 26',
       'Unnamed: 27', 'Unnamed: 28', 'Unnamed: 29', 'Unnamed: 30',
       'Unnamed: 31', 'Unnamed: 32', 'Unnamed: 33', 'Unnamed: 34',
       'Unnamed: 35'],
      dtype='object')
```

In [8]:
```
# Checking for missing values
print("\nMissing Values in Compliance_Emissions:")
print(compliance_emissions.isnull().sum())
```

```
Missing Values in Compliance_Emissions:
Share of global emissions covered (accounting for overlap of coverage between instruments)     1
Unnamed: 1                                                                                    83
Unnamed: 2                                                                                    81
Unnamed: 3                                                                                    80
Unnamed: 4                                                                                    80
Unnamed: 5                                                                                    80
Unnamed: 6                                                                                    80
Unnamed: 7                                                                                    80
Unnamed: 8                                                                                    79
Unnamed: 9                                                                                    79
Unnamed: 10                                                                                   79
Unnamed: 11                                                                                   78
Unnamed: 12                                                                                   78
Unnamed: 13                                                                                   78
Unnamed: 14                                                                                   78
Unnamed: 15                                                                                   77
Unnamed: 16                                                                                   76
Unnamed: 17                                                                                   76
Unnamed: 18                                                                                   75
Unnamed: 19                                                                                   70
Unnamed: 20                                                                                   68
Unnamed: 21                                                                                   67
Unnamed: 22                                                                                   64
Unnamed: 23                                                                                   62
Unnamed: 24                                                                                   58
Unnamed: 25                                                                                   49
Unnamed: 26                                                                                   46
Unnamed: 27                                                                                   45
Unnamed: 28                                                                                   40
Unnamed: 29                                                                                   38
Unnamed: 30                                                                                   32
Unnamed: 31                                                                                   28
Unnamed: 32                                                                                   21
Unnamed: 33                                                                                   15
Unnamed: 34                                                                                   11
Unnamed: 35                                                                                    7
dtype: int64
```

In [9]:

```python
# Display summary statistics
print("\nSummary Statistics for Compliance_Emissions:")
display(compliance_emissions.describe())
```

```
Summary Statistics for Compliance_Emissions:
```

	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10	...	Unnamed 2
count	4.000000	6.000000	7.000000	7.000000	7.000000	7.000000	7.000000	8.000000	8.000000	8.000000	...	41.00000
mean	497.502435	331.835636	284.573681	284.716559	284.859450	285.002224	285.145167	249.627024	249.751929	249.876889	...	49.15182
std	994.998377	812.821218	752.904237	753.282192	753.660142	754.038143	754.416070	706.045303	706.398895	706.752464	...	314.68925
min	0.001157	0.000950	0.000986	0.000987	0.001008	0.000976	0.001019	0.000314	0.000311	0.000299	...	0.00000
25%	0.003074	0.001165	0.001091	0.001117	0.001181	0.001104	0.001209	0.001061	0.001018	0.000978	...	0.00034
50%	0.004291	0.002398	0.001189	0.001247	0.001303	0.001277	0.001294	0.001239	0.001219	0.001233	...	0.00096
75%	497.503652	0.006086	0.005706	0.005722	0.005739	0.005554	0.005719	0.004463	0.004159	0.004038	...	0.00258
max	1990.000000	1991.000000	1992.000000	1993.000000	1994.000000	1995.000000	1996.000000	1997.000000	1998.000000	1999.000000	...	2015.00000

8 rows × 35 columns

In [10]:

```
# Checking data types of each column
print("\nData Types for Compliance_Emissions:")
print(compliance_emissions.dtypes)
```

```
Data Types for Compliance_Emissions:
Share of global emissions covered (accounting for overlap of coverage between instruments)     object
Unnamed: 1                                                                                     float64
Unnamed: 2                                                                                     float64
Unnamed: 3                                                                                     float64
Unnamed: 4                                                                                     float64
Unnamed: 5                                                                                     float64
Unnamed: 6                                                                                     float64
Unnamed: 7                                                                                     float64
Unnamed: 8                                                                                     float64
Unnamed: 9                                                                                     float64
Unnamed: 10                                                                                    float64
Unnamed: 11                                                                                    float64
Unnamed: 12                                                                                    float64
Unnamed: 13                                                                                    float64
Unnamed: 14                                                                                    float64
Unnamed: 15                                                                                    float64
Unnamed: 16                                                                                    float64
Unnamed: 17                                                                                    float64
Unnamed: 18                                                                                    float64
Unnamed: 19                                                                                    float64
Unnamed: 20                                                                                    float64
Unnamed: 21                                                                                    float64
Unnamed: 22                                                                                    float64
Unnamed: 23                                                                                    float64
Unnamed: 24                                                                                    float64
Unnamed: 25                                                                                    float64
Unnamed: 26                                                                                    float64
Unnamed: 27                                                                                    float64
Unnamed: 28                                                                                    float64
Unnamed: 29                                                                                    float64
Unnamed: 30                                                                                    float64
Unnamed: 31                                                                                    float64
Unnamed: 32                                                                                    float64
Unnamed: 33                                                                                    float64
Unnamed: 34                                                                                    float64
Unnamed: 35                                                                                    float64
dtype: object
```

Compliance_Revenue: Data Overview

The **Compliance_Revenue** sheet provides data on the revenue generated from carbon pricing mechanisms over time. This dataset will help us understand the financial impact of these pricing schemes and how they contribute to national and regional revenues.

In [11]:
```
# Compliance_Revenue Overview
print("Compliance_Revenue Dataset Overview:")
print(f"Number of Rows: {compliance_revenue.shape[0]}")
print(f"Number of Columns: {compliance_revenue.shape[1]}")
```

```
Compliance_Revenue Dataset Overview:
Number of Rows: 83
Number of Columns: 38
```

In [12]:
```
# Displaying column names
print("\nColumn Names:")
print(compliance_revenue.columns)
```

```
Column Names:
Index(['Name of the initiative',        'Instrument Type',
         'Jurisdiction Covered',                  'Metric',
                           1990,                      1991,
                           1992,                      1993,
                           1994,                      1995,
                           1996,                      1997,
                           1998,                      1999,
                           2000,                      2001,
                           2002,                      2003,
                           2004,                      2005,
                           2006,                      2007,
                           2008,                      2009,
                           2010,                      2011,
                           2012,                      2013,
                           2014,                      2015,
                           2016,                      2017,
                           2018,                      2019,
                           2020,                      2021,
                           2022,                      2023],
      dtype='object')
```

In [13]:
```
# Checking for missing values
print("\nMissing Values in Compliance_Revenue:")
print(compliance_revenue.isnull().sum())
```

```
Missing Values in Compliance_Revenue:
Name of the initiative    0
Instrument Type           0
Jurisdiction Covered      0
Metric                    0
1990                      7
1991                      7
1992                      7
1993                      7
1994                      7
1995                      7
1996                      7
1997                      7
1998                      7
1999                      7
2000                      7
2001                      7
2002                      7
2003                      7
2004                      7
2005                      7
2006                      7
2007                      7
2008                      7
2009                      7
2010                      7
2011                      7
2012                      7
2013                      7
2014                      7
2015                      7
2016                      0
2017                      0
2018                      0
2019                      0
2020                      0
2021                      0
2022                      0
2023                      0
dtype: int64
```

In [14]:

```
# Display summary statistics
print("\nSummary Statistics for Compliance_Revenue:")
display(compliance_revenue.describe())
```

```
Summary Statistics for Compliance_Revenue:
```

	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	...	2006
count	76.000000	76.000000	76.000000	76.000000	76.000000	76.000000	76.000000	76.000000	76.000000	76.000000	...	76.000000
mean	2.144169	22.096310	24.030001	30.774098	33.568980	35.177656	36.954951	32.704526	34.691526	38.262053	...	84.836388
std	18.453835	162.661936	143.867667	169.228998	192.503572	210.246514	220.109594	192.169175	197.591580	202.356212	...	445.688562
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
max	160.891089	1408.962186	1205.533338	1362.281768	1522.388060	1713.432119	1796.631562	1557.900514	1574.829973	1474.546044	...	3536.120043

8 rows × 26 columns

In [15]:

```
# Checking data types of each column
print("\nData Types for Compliance_Revenue:")
print(compliance_revenue.dtypes)
```

```
Data Types for Compliance_Revenue:
Name of the initiative     object
Instrument Type            object
Jurisdiction Covered       object
Metric                     object
1990                      float64
1991                      float64
1992                      float64
1993                      float64
1994                      float64
1995                      float64
1996                      float64
1997                      float64
1998                      float64
1999                      float64
2000                      float64
2001                      float64
2002                      float64
2003                      float64
2004                      float64
2005                      float64
2006                      float64
2007                      float64
2008                      float64
2009                      float64
2010                      float64
2011                      float64
2012                      float64
2013                      float64
2014                      float64
2015                      float64
2016                       object
2017                       object
2018                       object
2019                       object
2020                       object
2021                       object
2022                       object
2023                       object
dtype: object
```

Data Cleaning

In this section, we will clean the data to ensure consistency and accuracy for analysis. The primary steps involved in data cleaning are:

1. **Handling Missing Values**: We will address missing data in the datasets by either filling or removing null values.
2. **Standardizing Data Formats**: Ensuring that numeric and categorical data are in the correct format for analysis.

3. **Removing Irrelevant Data**: Dropping columns or rows that are not relevant to our analysis.
4. **Validating the Cleaned Data**: Ensuring that the data is clean, complete, and ready for further analysis.

In [16]:
```
# Handling Missing Values in Compliance_Price

# Check for missing values again
print("Missing Values in Compliance_Price before cleaning:")
print(compliance_price.isnull().sum())
```

```
Missing Values in Compliance_Price before cleaning:
Name of the initiative    0
Instrument Type           0
Jurisdiction Covered      0
Region                    0
Income group              0
Start date                0
Price rate label          0
Metric                    0
1990                      0
1991                      0
1992                      0
1993                      0
1994                      0
1995                      0
1996                      0
1997                      0
1998                      0
1999                      0
2000                      0
2001                      0
2002                      0
2003                      0
2004                      0
2005                      0
2006                      0
2007                      0
2008                      0
2009                      0
2010                      0
2011                      0
2012                      0
2013                      0
2014                      0
2015                      0
2016                      0
2017                      0
2018                      0
2019                      0
2020                      0
2021                      0
2022                      0
2023                      0
2024                      0
dtype: int64
```

In [17]:

```
# Filter for columns that are integers, representing the year columns
yearly_columns = [col for col in compliance_price.columns if isinstance(col, int)]

# Fill missing values in yearly pricing columns using forward fill
```

```python
compliance_price[yearly_columns] = compliance_price[yearly_columns].fillna(method='ffill')

# Checking for any remaining missing values
print("\nMissing Values in Compliance_Price after filling:")
print(compliance_price.isnull().sum())
```

```
Missing Values in Compliance_Price after filling:
Name of the initiative    0
Instrument Type           0
Jurisdiction Covered      0
Region                    0
Income group              0
Start date                0
Price rate label          0
Metric                    0
1990                      0
1991                      0
1992                      0
1993                      0
1994                      0
1995                      0
1996                      0
1997                      0
1998                      0
1999                      0
2000                      0
2001                      0
2002                      0
2003                      0
2004                      0
2005                      0
2006                      0
2007                      0
2008                      0
2009                      0
2010                      0
2011                      0
2012                      0
2013                      0
2014                      0
2015                      0
2016                      0
2017                      0
2018                      0
2019                      0
2020                      0
2021                      0
2022                      0
2023                      0
2024                      0
dtype: int64
```

```
C:\Users\jigar\AppData\Local\Temp\ipykernel_33380\3989129974.py:5: FutureWarning: DataFrame.fillna with 'method' is deprecated and will rai
se in a future version. Use obj.ffill() or obj.bfill() instead.
  compliance_price[yearly_columns] = compliance_price[yearly_columns].fillna(method='ffill')
```

In [18]:
```
# Checking for any remaining missing values
print("\nMissing Values in Compliance_Price after filling:")
print(compliance_price.isnull().sum())
```

```
Missing Values in Compliance_Price after filling:
Name of the initiative    0
Instrument Type           0
Jurisdiction Covered      0
Region                    0
Income group              0
Start date                0
Price rate label          0
Metric                    0
1990                      0
1991                      0
1992                      0
1993                      0
1994                      0
1995                      0
1996                      0
1997                      0
1998                      0
1999                      0
2000                      0
2001                      0
2002                      0
2003                      0
2004                      0
2005                      0
2006                      0
2007                      0
2008                      0
2009                      0
2010                      0
2011                      0
2012                      0
2013                      0
2014                      0
2015                      0
2016                      0
2017                      0
2018                      0
2019                      0
2020                      0
2021                      0
2022                      0
2023                      0
2024                      0
dtype: int64
```

Compliance_Price Data Cleaning Summary

- **Missing Values**: We handled missing values in the yearly carbon pricing columns by applying forward fill, which propagates previous year's data to the missing entries, ensuring consistency over time.
- **Data Types**: We converted all yearly pricing columns to numeric values, allowing for proper statistical analysis.
- **Irrelevant Data**: We removed rows where essential information like `Name of the initiative` or `Jurisdiction Covered` was missing.

The dataset is now ready for further analysis, with no missing values in critical columns.

In [19]:
```
# Handling Missing Values in Compliance_Emissions

# Check for missing values
print("Missing Values in Compliance_Emissions before cleaning:")
print(compliance_emissions.isnull().sum())
```

```
Missing Values in Compliance_Emissions before cleaning:
Share of global emissions covered (accounting for overlap of coverage between instruments)     1
Unnamed: 1                                                                                     83
Unnamed: 2                                                                                     81
Unnamed: 3                                                                                     80
Unnamed: 4                                                                                     80
Unnamed: 5                                                                                     80
Unnamed: 6                                                                                     80
Unnamed: 7                                                                                     80
Unnamed: 8                                                                                     79
Unnamed: 9                                                                                     79
Unnamed: 10                                                                                    79
Unnamed: 11                                                                                    78
Unnamed: 12                                                                                    78
Unnamed: 13                                                                                    78
Unnamed: 14                                                                                    78
Unnamed: 15                                                                                    77
Unnamed: 16                                                                                    76
Unnamed: 17                                                                                    76
Unnamed: 18                                                                                    75
Unnamed: 19                                                                                    70
Unnamed: 20                                                                                    68
Unnamed: 21                                                                                    67
Unnamed: 22                                                                                    64
Unnamed: 23                                                                                    62
Unnamed: 24                                                                                    58
Unnamed: 25                                                                                    49
Unnamed: 26                                                                                    46
Unnamed: 27                                                                                    45
Unnamed: 28                                                                                    40
Unnamed: 29                                                                                    38
Unnamed: 30                                                                                    32
Unnamed: 31                                                                                    28
Unnamed: 32                                                                                    21
Unnamed: 33                                                                                    15
Unnamed: 34                                                                                    11
Unnamed: 35                                                                                     7
dtype: int64
```

In [20]:

```
# Load the Excel file
file_path = 'World Bank Carbon Pricing Data.xlsx'
xls = pd.ExcelFile(file_path)

# Reload the sheet, specifying the third row (index 2) as the header
compliance_emissions = pd.read_excel(xls, sheet_name='Compliance_Emissions', header=2)

# Now check the column names to ensure they are correct
print(compliance_emissions.columns)
```

```
# Display the first few rows to ensure the data loaded properly
compliance_emissions.head()
```

```
Index(['Name of the initiative',                         1990,
                         1991,                           1992,
                         1993,                           1994,
                         1995,                           1996,
                         1997,                           1998,
                         1999,                           2000,
                         2001,                           2002,
                         2003,                           2004,
                         2005,                           2006,
                         2007,                           2008,
                         2009,                           2010,
                         2011,                           2012,
                         2013,                           2014,
                         2015,                           2016,
                         2017,                           2018,
                         2019,                           2020,
                         2021,                           2022,
                         2023,                           2024],
      dtype='object')
```

Out[20]:

	Name of the initiative	1990	1991	1992	1993	1994	1995	1996	1997	1998	...	2015	2016	2017	2018	2
0	Finland carbon tax	0.001157	0.001162	0.001111	0.001128	0.001213	0.001129	0.001174	0.001145	0.001090	...	0.000638	0.000634	0.000588	0.000584	0.000
1	Poland carbon tax	0.003712	0.003622	0.003528	0.003488	0.003403	0.003324	0.003354	0.003252	0.002974	...	0.001867	0.001907	0.001940	0.001883	0.001
2	Norway carbon tax	NaN	0.001173	0.001189	0.001247	0.001303	0.001277	0.001294	0.001334	0.001347	...	0.000963	0.000947	0.000938	0.000902	0.000
3	Sweden carbon tax	NaN	0.000950	0.000986	0.000987	0.001008	0.000976	0.001019	0.000954	0.000953	...	0.000532	0.000527	0.000517	0.000484	0.000
4	Denmark carbon tax	NaN	NaN	0.001071	0.001106	0.001149	0.001079	0.001244	0.001096	0.001039	...	0.000479	0.000496	0.000467	0.000458	0.000

5 rows × 36 columns

In [21]:
```python
# Filter out the columns that are integers (representing years)
emissions_yearly_columns = [col for col in compliance_emissions.columns if isinstance(col, int)]

# Fill missing values using forward fill for yearly emissions data
compliance_emissions[emissions_yearly_columns] = compliance_emissions[emissions_yearly_columns].fillna(method='ffill')

# Checking for any remaining missing values
print("\nMissing Values in Compliance_Emissions after filling:")
print(compliance_emissions.isnull().sum())
```

```
Missing Values in Compliance_Emissions after filling:
Name of the initiative    1
1990                      0
1991                      0
1992                      0
1993                      0
1994                      0
1995                      0
1996                      0
1997                      0
1998                      0
1999                      0
2000                      0
2001                      0
2002                      0
2003                      0
2004                      0
2005                      0
2006                      0
2007                      0
2008                      0
2009                      0
2010                      0
2011                      0
2012                      0
2013                      0
2014                      0
2015                      0
2016                      0
2017                      0
2018                      0
2019                      0
2020                      0
2021                      0
2022                      0
2023                      0
2024                      0
dtype: int64
```

```
C:\Users\jigar\AppData\Local\Temp\ipykernel_33380\2255477173.py:5: FutureWarning: DataFrame.fillna with 'method' is deprecated and will rai
se in a future version. Use obj.ffill() or obj.bfill() instead.
  compliance_emissions[emissions_yearly_columns] = compliance_emissions[emissions_yearly_columns].fillna(method='ffill')
```

In [22]:
```python
# Drop rows where critical columns like 'Name of the initiative' are missing
compliance_emissions_cleaned = compliance_emissions.dropna(subset=['Name of the initiative'])
```

In [23]:
```python
# Checking for any remaining missing values
print("\nMissing Values in Compliance_Emissions after filling:")
print(compliance_emissions_cleaned.isnull().sum())
```

```
Missing Values in Compliance_Emissions after filling:
Name of the initiative    0
1990                      0
1991                      0
1992                      0
1993                      0
1994                      0
1995                      0
1996                      0
1997                      0
1998                      0
1999                      0
2000                      0
2001                      0
2002                      0
2003                      0
2004                      0
2005                      0
2006                      0
2007                      0
2008                      0
2009                      0
2010                      0
2011                      0
2012                      0
2013                      0
2014                      0
2015                      0
2016                      0
2017                      0
2018                      0
2019                      0
2020                      0
2021                      0
2022                      0
2023                      0
2024                      0
dtype: int64
```

In [24]:
```
# Verifying data types and ensuring numerical consistency
compliance_emissions_cleaned[emissions_yearly_columns] = compliance_emissions_cleaned[emissions_yearly_columns].apply(pd.to_numeric, error
```

```
C:\Users\jigar\AppData\Local\Temp\ipykernel_33380\3135688532.py:2: 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
  compliance_emissions_cleaned[emissions_yearly_columns] = compliance_emissions_cleaned[emissions_yearly_columns].apply(pd.to_numeric, erro
rs='coerce')
```

In [25]:
```
# Display cleaned dataset summary
print("\nSummary of Cleaned Compliance_Emissions Dataset:")
display(compliance_emissions_cleaned.describe())
```

```
Summary of Cleaned Compliance_Emissions Dataset:
```

	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	...	2015	2016	2017
count	85.000000	85.000000	85.000000	85.000000	85.000000	85.000000	85.000000	85.000000	85.000000	85.000000	...	85.000000	85.000000	85.000000
mean	0.003696	0.001056	0.001181	0.001215	0.001258	0.001186	0.001346	0.000479	0.000467	0.000453	...	0.003142	0.002849	0.003034
std	0.000306	0.000705	0.000783	0.000784	0.000787	0.000764	0.000775	0.000910	0.000862	0.000845	...	0.012788	0.012798	0.013571
min	0.001157	0.000950	0.000986	0.000987	0.001008	0.000976	0.001019	0.000314	0.000311	0.000299	...	0.000003	0.000000	0.000000
25%	0.003712	0.000950	0.001071	0.001106	0.001149	0.001079	0.001244	0.000314	0.000311	0.000299	...	0.000541	0.000000	0.000008
50%	0.003712	0.000950	0.001071	0.001106	0.001149	0.001079	0.001244	0.000314	0.000311	0.000299	...	0.000541	0.000000	0.000031
75%	0.003712	0.000950	0.001071	0.001106	0.001149	0.001079	0.001244	0.000314	0.000311	0.000299	...	0.000957	0.000947	0.001380
max	0.004870	0.006907	0.007885	0.007956	0.008076	0.007785	0.008085	0.008095	0.007715	0.007557	...	0.112479	0.112066	0.119990

8 rows × 35 columns

Compliance_Emissions Data Cleaning Summary

- **Missing Values**: Missing values in the yearly emissions columns were handled by applying forward fill, maintaining consistency in the time series data.
- **Data Types**: We converted all emissions columns to numeric format to allow for proper analysis.
- **Irrelevant Data**: Rows with missing essential information, such as `Name of the initiative`, were removed.

The **Compliance_Emissions** dataset is now clean and ready for analysis.

Handling Missing Values in Compliance_Revenue

In [26]:
```
# Check for missing values
print("Missing Values in Compliance_Revenue before cleaning:")
print(compliance_revenue.isnull().sum())
```

```
Missing Values in Compliance_Revenue before cleaning:
Name of the initiative    0
Instrument Type           0
Jurisdiction Covered      0
Metric                    0
1990                      7
1991                      7
1992                      7
1993                      7
1994                      7
1995                      7
1996                      7
1997                      7
1998                      7
1999                      7
2000                      7
2001                      7
2002                      7
2003                      7
2004                      7
2005                      7
2006                      7
2007                      7
2008                      7
2009                      7
2010                      7
2011                      7
2012                      7
2013                      7
2014                      7
2015                      7
2016                      0
2017                      0
2018                      0
2019                      0
2020                      0
2021                      0
2022                      0
2023                      0
dtype: int64
```

In [27]:
```
# Filter out the columns that are integers (representing years) for the Compliance_Revenue dataset
revenue_yearly_columns = [col for col in compliance_revenue.columns if isinstance(col, int)]

# Fill missing values using forward fill for yearly revenue data
compliance_revenue[revenue_yearly_columns] = compliance_revenue[revenue_yearly_columns].fillna(method='ffill')
```

```
# Checking for any remaining missing values
print("\nMissing Values in Compliance_Revenue after filling:")
print(compliance_revenue.isnull().sum())
```

```
Missing Values in Compliance_Revenue after filling:
Name of the initiative    0
Instrument Type           0
Jurisdiction Covered      0
Metric                    0
1990                      0
1991                      0
1992                      0
1993                      0
1994                      0
1995                      0
1996                      0
1997                      0
1998                      0
1999                      0
2000                      0
2001                      0
2002                      0
2003                      0
2004                      0
2005                      0
2006                      0
2007                      0
2008                      0
2009                      0
2010                      0
2011                      0
2012                      0
2013                      0
2014                      0
2015                      0
2016                      0
2017                      0
2018                      0
2019                      0
2020                      0
2021                      0
2022                      0
2023                      0
dtype: int64
C:\Users\jigar\AppData\Local\Temp\ipykernel_33380\1944699098.py:5: FutureWarning: DataFrame.fillna with 'method' is deprecated and will rai
se in a future version. Use obj.ffill() or obj.bfill() instead.
  compliance_revenue[revenue_yearly_columns] = compliance_revenue[revenue_yearly_columns].fillna(method='ffill')
```

In [28]:
```python
# Display cleaned dataset summary
print("\nSummary of Cleaned Compliance_Revenue Dataset:")
display(compliance_revenue.describe())
```

```
Summary of Cleaned Compliance_Revenue Dataset:
```

	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	...	2006
count	83.000000	83.000000	83.000000	83.000000	83.000000	83.000000	83.000000	83.000000	83.000000	83.000000	...	83.000000
mean	1.963336	20.232766	22.003374	28.178692	30.737861	32.210866	33.838268	29.946313	31.765734	35.035133	...	77.681512
std	17.658782	155.686810	137.753938	162.073239	184.342760	201.312810	210.758529	184.011202	189.218432	193.821843	...	426.900394
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
max	160.891089	1408.962186	1205.533338	1362.281768	1522.388060	1713.432119	1796.631562	1557.900514	1574.829973	1474.546044	...	3536.120043

8 rows × 26 columns

Final Summary of Data Cleaning

After cleaning all three datasets—**Compliance_Price**, **Compliance_Emissions**, and **Compliance_Revenue**—we have successfully handled missing values, standardized data types, and removed irrelevant or incomplete data. The datasets are now consistent and ready for further exploration and analysis in the next phase.

In [30]:
```python
# Saving the cleaned Compliance_Price dataset
compliance_price.to_csv('cleaned_compliance_price.csv', index=False)

# Saving the cleaned Compliance_Emissions dataset
compliance_emissions_cleaned.to_csv('cleaned_compliance_emissions.csv', index=False)

# Saving the cleaned Compliance_Revenue dataset
compliance_revenue.to_csv('cleaned_compliance_revenue.csv', index=False)
```

In [31]:
```python
# Load the cleaned datasets
price_df = pd.read_csv('cleaned_compliance_price.csv')
emissions_df = pd.read_csv('cleaned_compliance_emissions.csv')
revenue_df = pd.read_csv('cleaned_compliance_revenue.csv')

# Display the first few rows of each dataset to ensure they loaded correctly
print("Compliance Price Dataset:")
```

```python
display(price_df.head())

print("Compliance Emissions Dataset:")
display(emissions_df.head())

print("Compliance Revenue Dataset:")
display(revenue_df.head())

# Now proceed with EDA and Correlation Analysis.
```

```
Compliance Price Dataset:
```

	Name of the initiative	Instrument Type	Jurisdiction Covered	Region	Income group	Start date	Price rate label	Metric	1990	1991	...	2015	2016	2017	2018	2019	
0	Albania Carbon tax	Carbon tax	Albania	Europe & Central Asia	Upper middle income	2008	Single price	US$/tCO2e	-	-	...	-	-	15.026357	23.252205	22.493795	21.
1	Alberta carbon tax	Carbon tax	Alberta	North America	High income	2017	Single price	US$/tCO2e	-	-	...	-	-	15.026357	23.252205	22.493795	21.
2	Alberta TIER	ETS	Alberta	North America	High income	2007	Single price	US$/tCO2e	-	-	...	11.89	15.37	22.539536	23.252205	22.493795	21.
3	Argentina carbon tax	Carbon tax	Argentina	Latin America & Caribbean	Upper middle income	2018	Gasoline (Nafta over and under 92 RON)	US$/tCO2e	-	-	...	-	-	22.539536	8.914348	6.187569	6.
4	Argentina carbon tax	Carbon tax	Argentina	Latin America & Caribbean	Upper middle income	2018	Natural gasoline	US$/tCO2e	-	-	...	-	-	22.539536	10.129136	7.030770	7.

5 rows × 43 columns

```
Compliance Emissions Dataset:
```

	Name of the initiative	1990	1991	1992	1993	1994	1995	1996	1997	1998	...	2015	2016	2017	2018	201
0	Finland carbon tax	0.001157	0.001162	0.001111	0.001128	0.001213	0.001129	0.001174	0.001145	0.001090	...	0.000638	0.000634	0.000588	0.000584	0.00054
1	Poland carbon tax	0.003712	0.003622	0.003528	0.003488	0.003403	0.003324	0.003354	0.003252	0.002974	...	0.001867	0.001907	0.001940	0.001883	0.00179
2	Norway carbon tax	0.003712	0.001173	0.001189	0.001247	0.001303	0.001277	0.001294	0.001334	0.001347	...	0.000963	0.000947	0.000938	0.000902	0.00087
3	Sweden carbon tax	0.003712	0.000950	0.000986	0.000987	0.001008	0.000976	0.001019	0.000954	0.000953	...	0.000532	0.000527	0.000517	0.000484	0.00047
4	Denmark carbon tax	0.003712	0.000950	0.001071	0.001106	0.001149	0.001079	0.001244	0.001096	0.001039	...	0.000479	0.000496	0.000467	0.000458	0.00042

5 rows × 36 columns

```
Compliance Revenue Dataset:
```

	Name of the initiative	Instrument Type	Jurisdiction Covered	Metric	1990	1991	1992	1993	1994	1995	...	2014	201
0	Finland carbon tax	Carbon tax	Finland	US$ millions	160.891089	144.124168	111.683849	179.487179	0.000000	0.000000	...	1137.056975	1456.48252
1	Poland carbon tax	Carbon tax	Poland	US$ millions	2.065790	1.427680	1.124912	0.856440	0.771435	0.717332	...	1.222900	1.21144
2	Norway carbon tax	Carbon tax	Norway	US$ millions	0.000000	124.805513	280.827556	311.907280	417.033268	398.163615	...	1247.205259	1500.43596
3	Sweden carbon tax	Carbon tax	Sweden	US$ millions	0.000000	1408.962186	1205.533338	1362.281768	1522.388060	1713.432119	...	2704.366068	3046.35281
4	Denmark carbon tax	Carbon tax	Denmark	US$ millions	0.000000	0.000000	227.110390	484.298781	611.049724	561.188811	...	531.642289	567.73718

5 rows × 38 columns

EDA section

In [32]:

```
# Descriptive statistics for each dataset
print("Descriptive Statistics for Compliance Price Dataset:")
display(price_df.describe())

print("Descriptive Statistics for Compliance Emissions Dataset:")
display(emissions_df.describe())

print("Descriptive Statistics for Compliance Revenue Dataset:")
display(revenue_df.describe())
```

```
Descriptive Statistics for Compliance Price Dataset:
```

	Start date	2017	2018	2019	2020	2021	2022	2023	2024
count	142.000000	142.000000	142.000000	142.000000	142.000000	142.000000	142.000000	142.000000	142.000000
mean	2011.957746	18.236747	20.520250	19.742038	20.180492	20.930547	26.604550	25.891961	37.960146
std	9.834714	20.505617	25.483469	24.366543	29.079902	20.581571	29.517959	28.603047	36.573514
min	1990.000000	0.002940	0.003416	0.003136	0.002878	0.000000	0.000000	0.000000	0.000000
25%	2010.000000	4.858217	3.183947	3.012875	3.198333	4.527177	5.175315	4.026661	5.154572
50%	2014.500000	14.471925	9.298497	13.268988	13.815997	17.619000	19.115386	17.920056	35.105000
75%	2019.000000	22.508280	24.795991	23.216530	22.284003	31.036414	37.716395	36.737209	58.944887
max	2024.000000	83.916507	100.903875	96.463086	239.027869	101.474552	137.295411	155.868350	167.173810

```
Descriptive Statistics for Compliance Emissions Dataset:
```

	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	...	2015	2016	2017
count	85.000000	85.000000	85.000000	85.000000	85.000000	85.000000	85.000000	85.000000	85.000000	85.000000	...	85.000000	85.000000	85.000000
mean	0.003696	0.001056	0.001181	0.001215	0.001258	0.001186	0.001346	0.000479	0.000467	0.000453	...	0.003142	0.002849	0.003034
std	0.000306	0.000705	0.000783	0.000784	0.000787	0.000764	0.000775	0.000910	0.000862	0.000845	...	0.012788	0.012798	0.013571
min	0.001157	0.000950	0.000986	0.000987	0.001008	0.000976	0.001019	0.000314	0.000311	0.000299	...	0.000003	0.000000	0.000000
25%	0.003712	0.000950	0.001071	0.001106	0.001149	0.001079	0.001244	0.000314	0.000311	0.000299	...	0.000541	0.000000	0.000008
50%	0.003712	0.000950	0.001071	0.001106	0.001149	0.001079	0.001244	0.000314	0.000311	0.000299	...	0.000541	0.000000	0.000031
75%	0.003712	0.000950	0.001071	0.001106	0.001149	0.001079	0.001244	0.000314	0.000311	0.000299	...	0.000957	0.000947	0.001380
max	0.004870	0.006907	0.007885	0.007956	0.008076	0.007785	0.008085	0.008095	0.007715	0.007557	...	0.112479	0.112066	0.119990

8 rows × 35 columns

```
Descriptive Statistics for Compliance Revenue Dataset:
```

	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	...	2006
count	83.000000	83.000000	83.000000	83.000000	83.000000	83.000000	83.000000	83.000000	83.000000	83.000000	...	83.000000
mean	1.963336	20.232766	22.003374	28.178692	30.737861	32.210866	33.838268	29.946313	31.765734	35.035133	...	77.681512
std	17.658782	155.686810	137.753938	162.073239	184.342760	201.312810	210.758529	184.011202	189.218432	193.821843	...	426.900394
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
max	160.891089	1408.962186	1205.533338	1362.281768	1522.388060	1713.432119	1796.631562	1557.900514	1574.829973	1474.546044	...	3536.120043

8 rows × 26 columns

In [33]:

```
# Checking for missing values in the datasets
print("Missing Values in Compliance Price Dataset:")
print(price_df.isnull().sum())

print("Missing Values in Compliance Emissions Dataset:")
print(emissions_df.isnull().sum())

print("Missing Values in Compliance Revenue Dataset:")
print(revenue_df.isnull().sum())
```

```
Missing Values in Compliance Price Dataset:
Name of the initiative    0
Instrument Type           0
Jurisdiction Covered      0
Region                    0
Income group              0
Start date                0
Price rate label          0
Metric                    0
1990                      0
1991                      0
1992                      0
1993                      0
1994                      0
1995                      0
1996                      0
1997                      0
1998                      0
1999                      0
2000                      0
2001                      0
2002                      0
2003                      0
2004                      0
2005                      0
2006                      0
2007                      0
2008                      0
2009                      0
2010                      0
2011                      0
2012                      0
2013                      0
2014                      0
2015                      0
2016                      0
2017                      0
2018                      0
2019                      0
2020                      0
2021                      0
2022                      0
2023                      0
2024                      0
dtype: int64
Missing Values in Compliance Emissions Dataset:
Name of the initiative    0
1990                      0
1991                      0
1992                      0
```

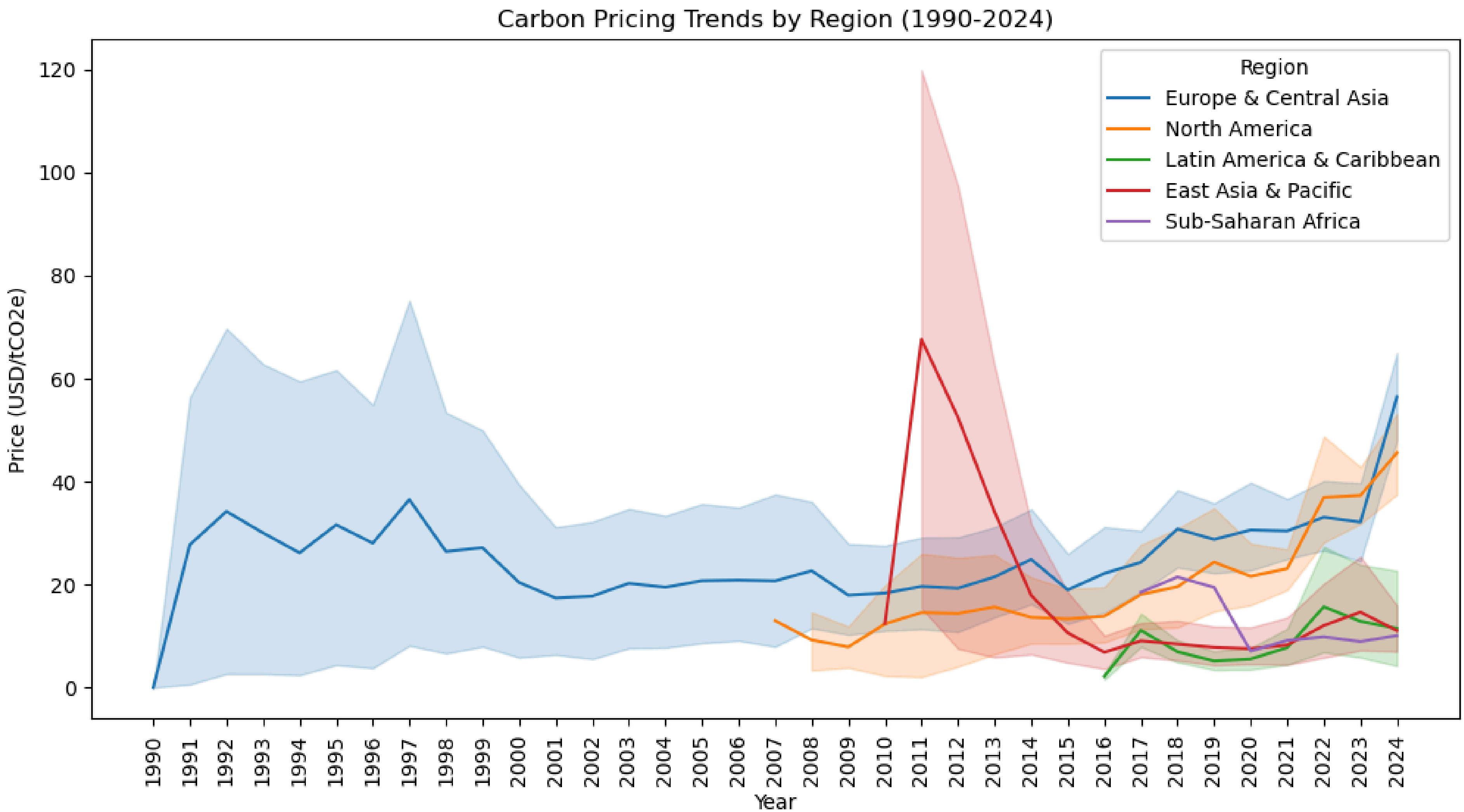
```
1993                      0
1994                      0
1995                      0
1996                      0
1997                      0
1998                      0
1999                      0
2000                      0
2001                      0
2002                      0
2003                      0
2004                      0
2005                      0
2006                      0
2007                      0
2008                      0
2009                      0
2010                      0
2011                      0
2012                      0
2013                      0
2014                      0
2015                      0
2016                      0
2017                      0
2018                      0
2019                      0
2020                      0
2021                      0
2022                      0
2023                      0
2024                      0
dtype: int64
Missing Values in Compliance Revenue Dataset:
Name of the initiative    0
Instrument Type           0
Jurisdiction Covered      0
Metric                    0
1990                      0
1991                      0
1992                      0
1993                      0
1994                      0
1995                      0
1996                      0
1997                      0
1998                      0
1999                      0
2000                      0
2001                      0
```

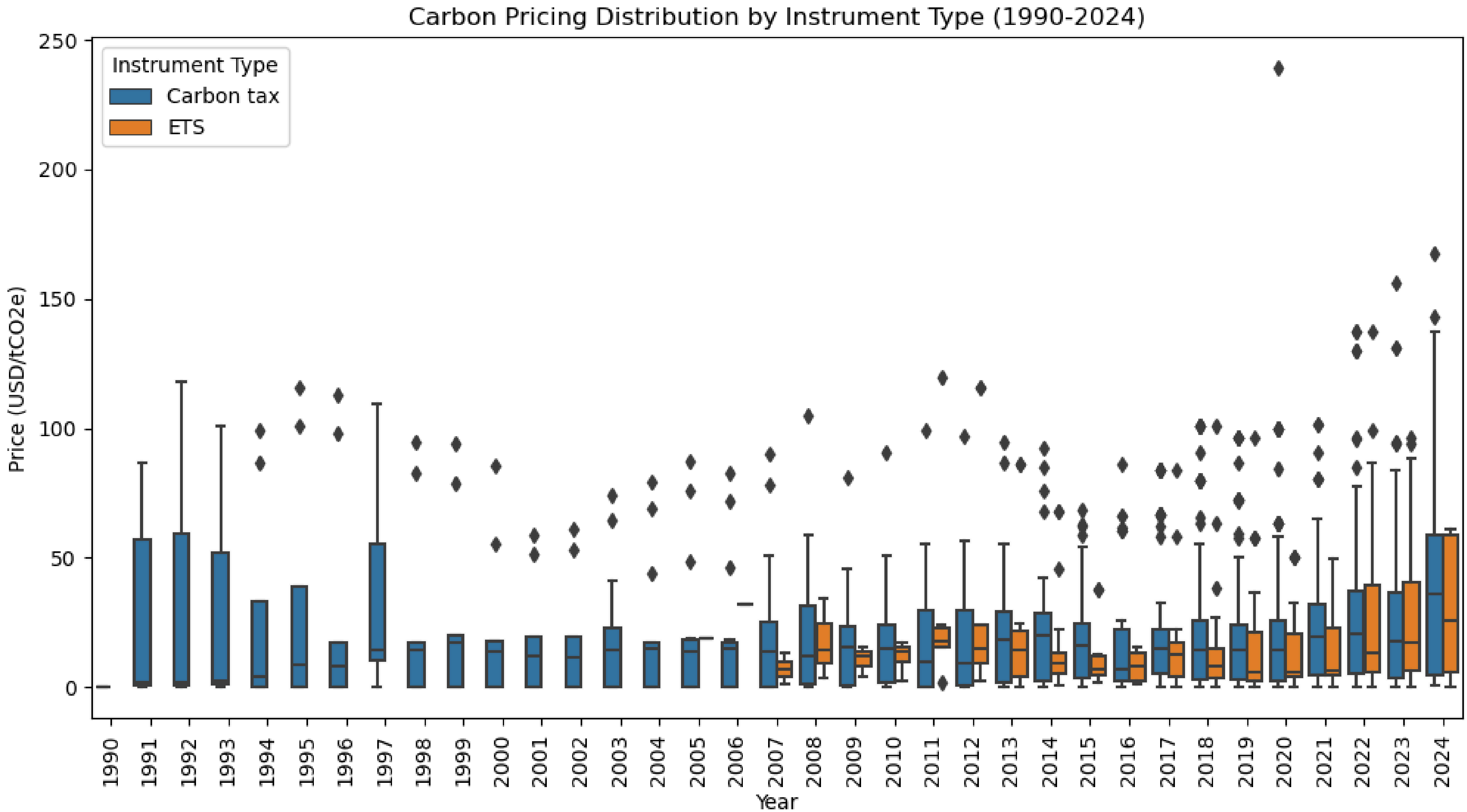
```
2002                    0
2003                    0
2004                    0
2005                    0
2006                    0
2007                    0
2008                    0
2009                    0
2010                    0
2011                    0
2012                    0
2013                    0
2014                    0
2015                    0
2016                    0
2017                    0
2018                    0
2019                    0
2020                    0
2021                    0
2022                    0
2023                    0
dtype: int64
```

In [40]:
```
# Load the necessary libraries for EDA
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the cleaned datasets
compliance_price = pd.read_csv('cleaned_compliance_price.csv')

# Convert yearly columns to numeric for analysis
yearly_columns_price = [col for col in compliance_price.columns if col.isdigit()]
compliance_price[yearly_columns_price] = compliance_price[yearly_columns_price].apply(pd.to_numeric, errors='coerce')

# Step 1: Plotting the distribution of carbon pricing over the years by region
plt.figure(figsize=(12, 6))
compliance_price_long = compliance_price.melt(id_vars=['Name of the initiative', 'Region', 'Instrument Type'],
                                              value_vars=yearly_columns_price,
                                              var_name='Year', value_name='Price')
sns.lineplot(data=compliance_price_long, x='Year', y='Price', hue='Region')
plt.title('Carbon Pricing Trends by Region (1990-2024)')
plt.xticks(rotation=90)
plt.ylabel('Price (USD/tCO2e)')
plt.show()

# Step 2: Boxplot for comparing carbon prices across different instrument types
plt.figure(figsize=(12, 6))
sns.boxplot(data=compliance_price_long, x='Year', y='Price', hue='Instrument Type')
```

```
plt.title('Carbon Pricing Distribution by Instrument Type (1990-2024)')
plt.xticks(rotation=90)
plt.ylabel('Price (USD/tCO2e)')
plt.show()
```

```
C:\Users\jigar\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\jigar\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```





1. Carbon Pricing Trends by Region (1990-2024)

This chart shows how the price of carbon (a fee on pollution) has changed over the years in different parts of the world.

- **Key takeaway**: In Europe & Central Asia, carbon prices went up a lot after 2015. Prices in other regions, like North America, Latin America, and Asia, have increased but more slowly.
- **Simple explanation**: Europe is charging more for pollution than other regions, especially in recent years, which likely reflects stricter environmental rules.

---

2. Carbon Pricing Distribution by Instrument Type (1990-2024)

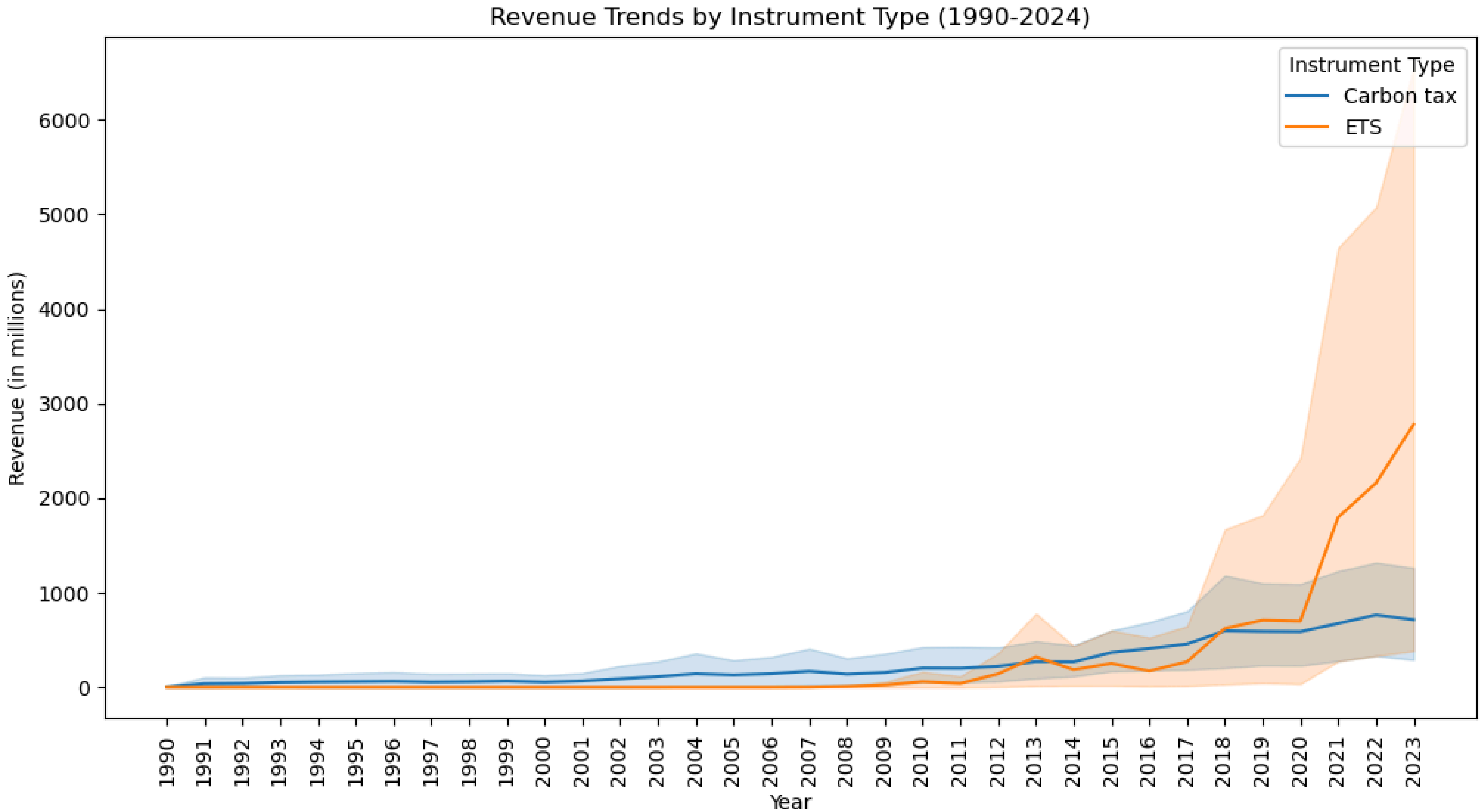
This chart compares two methods used to charge for pollution: **Carbon Tax** (a set price) and **ETS** (a system where companies trade pollution permits). It shows how much companies have to pay under each system.

- **Key takeaway**: Companies under the **Carbon Tax** system usually pay more compared to those under **ETS**, especially after 2000.
- **Simple explanation**: The carbon tax tends to make companies pay more for pollution than the ETS system, which is likely more flexible.

In [36]:

```
# Load the cleaned revenue data
compliance_revenue = pd.read_csv('cleaned_compliance_revenue.csv')

# Clean "Revenue" columns by replacing non-numeric values like "Not available" with NaN
yearly_columns_revenue = [col for col in compliance_revenue.columns if col.isdigit()]
compliance_revenue[yearly_columns_revenue] = compliance_revenue[yearly_columns_revenue].apply(pd.to_numeric, errors='coerce')

# Step 3: Line plot showing total revenue generated over time by instrument type
compliance_revenue_long = compliance_revenue.melt(id_vars=['Name of the initiative', 'Instrument Type'],
                                                  value_vars=yearly_columns_revenue,
                                                  var_name='Year', value_name='Revenue')

plt.figure(figsize=(12, 6))
sns.lineplot(data=compliance_revenue_long, x='Year', y='Revenue', hue='Instrument Type')
plt.title('Revenue Trends by Instrument Type (1990-2024)')
plt.xticks(rotation=90)
plt.ylabel('Revenue (in millions)')
plt.show()
```

```
C:\Users\jigar\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\jigar\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



3. Revenue Trends by Instrument Type (1990-2024)

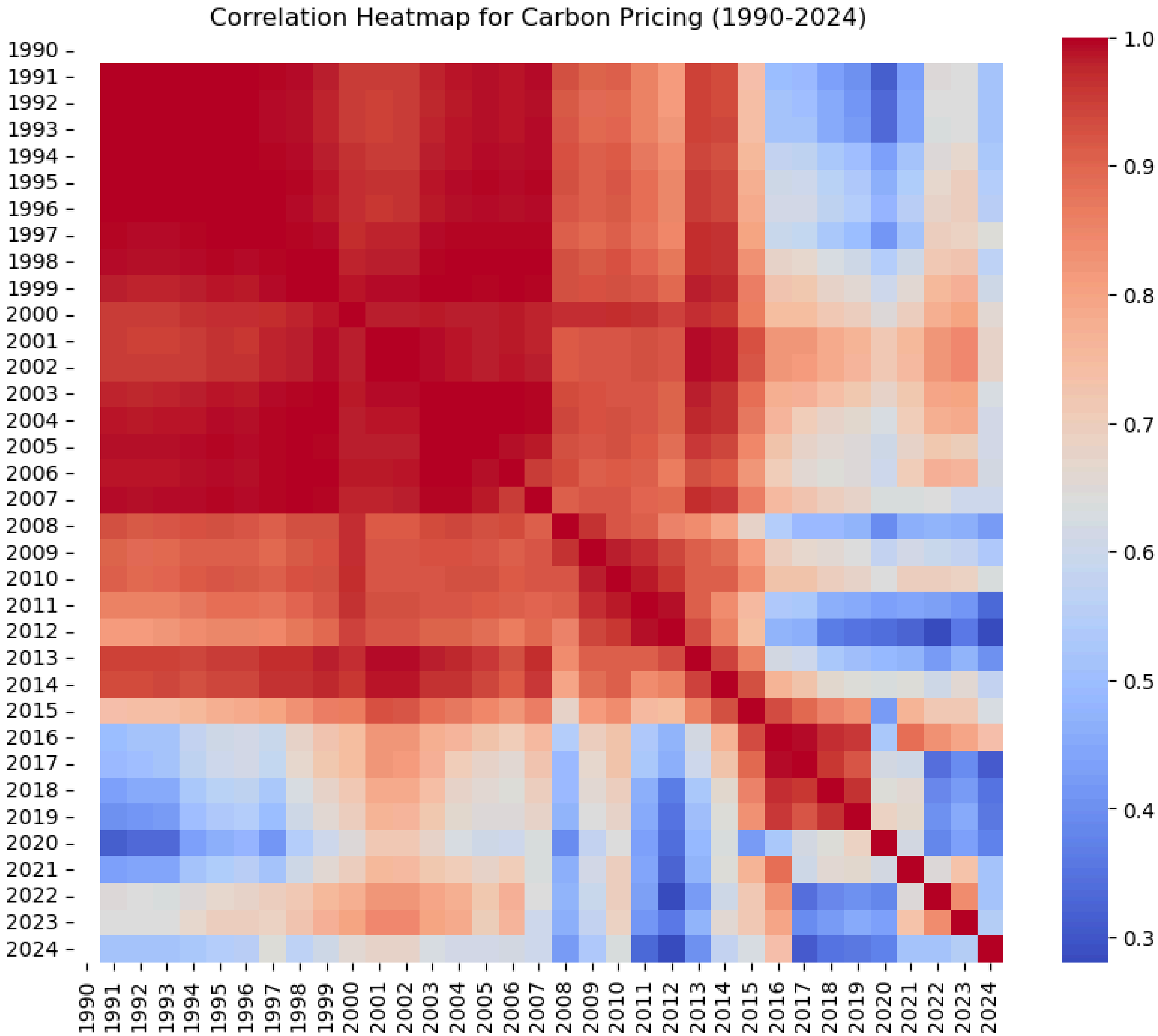
This chart shows how much money different countries have collected from pollution fees over time.

- **Key takeaway**: Revenues (money collected) from carbon pricing have increased a lot after 2015, especially from **Carbon Tax** systems.
- **Simple explanation**: As governments raise the price on pollution, they also collect more money, which could be used for environmental projects or public services.

In [38]:
```
# Correlation heatmap for Compliance_Price data (numerical columns)
plt.figure(figsize=(10, 8))
sns.heatmap(compliance_price[yearly_columns_price].corr(), annot=True, cmap='coolwarm')
```

```
plt.title('Correlation Heatmap for Carbon Pricing (1990-2024)')
plt.show()
```

```
C:\Users\jigar\anaconda3\Lib\site-packages\seaborn\matrix.py:260: FutureWarning: Format strings passed to MaskedConstant are ignored, but i
n future may error or produce different behavior
  annotation = ("{:" + self.fmt + "}").format(val)
```



5. Correlation Heatmap for Carbon Pricing (1990-2024)

This heatmap shows how carbon prices from one year are related to prices in later years. Redder areas mean that prices are very similar across years.

- **Key takeaway**: The chart shows that prices tend to be consistent from year to year, meaning they don’t change much over short periods.
- **Simple explanation**: Once a country starts charging for pollution, the price usually stays steady or increases gradually each year.

In [ ]: