NORTHEASTERN UNIVERSIY
MIE Department
IE5390 Excel

FINAL PROJECT

NAME: BALAJI PAMIDI
NEU ID: 002644804

Data Analysis Report

Uncovering Trends in Weekly Provisional Counts of Deaths

Introduction :

As data analyst, my work included classification and causes of COVID-19 cases by U.S. states, specifically referred to with ICD-10 code U07.1. Our ultimate goal is to distil this complexity into actionable intelligence, empowering stakeholders to make informed decisions. We will examine key variables more closely including control, MMWR year, week, all-cause mortality, cardiovascular diseases, COVID-19-related deaths and other causes. Through this rigorous research, we aim to identify trends and policies that shed light on important public health issues. By identifying these trends, we can provide targeted interventions and strategies to better address emerging challenges. This report is an important resource, translating complex data into meaningful insights that help solve public health challenges. Ultimately, our goal is to help protect and prosper all Americans by providing actionable intelligence that guides strategic action and decision-making processes.

1.Data collected from the Data.Gov

link : https://catalog.data.gov/dataset/weekly-counts-of-deaths-by-state-and-select-causes-2019-2020

Here I am going to use the Python Jupyter notebook to this task

Data Analysis Report: Uncovering Trends in Weekly Provisional Counts of Deaths.

1. Data collected from the Data Gov.

link : https://catalog.data.gov/dataset/weekly-counts-of-deaths-by-state-and-select-causes-2019-2020

In [2928]:
```
# Calling the .CSV Dataset file using the pandas Library.

import pandas as pd

# Assuming the dataset is in CSV format and located in the same directory as your script or notebook
file_path = "Weekly_Provisional_Counts_of_Deaths_by_State_and_Select_Causes__2020-2023.csv"

# Load the dataset into a Pandas DataFrame
df = pd.read_csv(file_path)

# Display the first few rows of the DataFrame to verify that the data has been loaded correctly
df
```

we can see that there are 35 columns and 10476 rows in the dataset.

```
# Display the first few rows of the DataFrame to verify that the data has been loaded correctly
df
```

Out[2928]:

	Data As Of	Jurisdiction of Occurrence	MMWR Year	MMWR Week	Week Ending Date	All Cause	Natural Cause	Septicemia (A40-A41)	Malignant neoplasms (C00-C97)	Diabetes mellitus (E10-E14)	...	flag_alz	flag_inflpn	flag_clrd	flag_otherresp
0	09/27/2023	United States	2020	1	2020-01-04	60179	55010	843.0	11567.0	1829.0	...	NaN	NaN	NaN	NaN
1	09/27/2023	United States	2020	2	2020-01-11	60735	55755	861.0	11963.0	1942.0	...	NaN	NaN	NaN	NaN
2	09/27/2023	United States	2020	3	2020-01-18	59363	54516	829.0	11701.0	1819.0	...	NaN	NaN	NaN	NaN
3	09/27/2023	United States	2020	4	2020-01-25	59162	54401	828.0	11879.0	1864.0	...	NaN	NaN	NaN	NaN
4	09/27/2023	United States	2020	5	2020-02-01	58834	54001	811.0	11963.0	1828.0	...	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10471	09/27/2023	Puerto Rico	2023	33	2023-08-19	612	590	NaN	92.0	48.0	...	NaN	NaN	NaN	NaN
10472	09/27/2023	Puerto Rico	2023	34	2023-08-26	657	624	15.0	110.0	48.0	...	NaN	NaN	NaN	NaN
10473	09/27/2023	Puerto Rico	2023	35	2023-09-02	580	552	16.0	97.0	70.0	...	NaN	NaN	NaN	Suppressed (counts 1-9)
10474	09/27/2023	Puerto Rico	2023	36	2023-09-09	533	516	10.0	81.0	50.0	...	NaN	NaN	NaN	NaN
10475	09/27/2023	Puerto Rico	2023	37	2023-09-16	453	453	NaN	85.0	40.0	...	NaN	NaN	NaN	NaN

10476 rows × 35 columns

By using the df.info() we can see that each column data type and whether it contains null values or not.

```
In [2929]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10476 entries, 0 to 10475
Data columns (total 35 columns):
 #   Column                                                                                                  Non-Null Co
unt  Dtype
---  ------                                                                                                  ----------
---  -----
 0   Data As Of                                                                                              10476 non-n
ull  object
 1   Jurisdiction of Occurrence                                                                              10476 non-n
ull  object
 2   MMWR Year                                                                                               10476 non-n
ull  int64
 3   MMWR Week                                                                                               10476 non-n
ull  int64
 4   Week Ending Date                                                                                        10476 non-n
ull  object
 5   All Cause                                                                                               10476 non-n
ull  int64
 6   Natural Cause                                                                                           10476 non-n
ull  int64
 7   Septicemia (A40-A41)                                                                                    6083 non-nu
ll   float64
 8   Malignant neoplasms (C00-C97)                                                                           10466 non-n
ull  float64
 9   Diabetes mellitus (E10-E14)                                                                             8234 non-nu
ll   float64
 10  Alzheimer disease (G30)                                                                                 8732 non-nu
ll   float64
 11  Influenza and pneumonia (J09-J18)                                                                       6241 non-nu
ll   float64
 12  Chronic lower respiratory diseases (J40-J47)                                                            8965 non-nu
ll   float64
 13  Other diseases of respiratory system (J00-J06,J30-J39,J67,J70-J98)                                      6305 non-nu
```

2. Data Cleaning

Data cleaning is a critical process that precedes analysis, ensuring data accuracy and reliability.I recognize the importance of refining our dataset for optimal insights. This involves detecting and rectifying anomalies like missing values, duplicates, and outliers. Leveraging automated tools and manual techniques, we address discrepancies and format inconsistencies. By refining our data, we bolster the integrity of our analyses, fostering informed decision-making. Ultimately, effective data cleaning is foundational to deriving meaningful insights and driving actionable outcomes.

Here we are going to use the some Python libraires for cleaning.

First Method is Renaming the columns Headings

2. Data Cleaning

First Method is Renaming the columns Headings

```
In [2930]: #Columns Headings
           df.columns

Out[2930]: Index(['Data As Of', 'Jurisdiction of Occurrence', 'MMWR Year', 'MMWR Week',
                  'Week Ending Date', 'All Cause', 'Natural Cause',
                  'Septicemia (A40-A41)', 'Malignant neoplasms (C00-C97)',
                  'Diabetes mellitus (E10-E14)', 'Alzheimer disease (G30)',
                  'Influenza and pneumonia (J09-J18)',
                  'Chronic lower respiratory diseases (J40-J47)',
                  'Other diseases of respiratory system (J00-J06,J30-J39,J67,J70-J98)',
                  'Nephritis, nephrotic syndrome and nephrosis (N00-N07,N17-N19,N25-N27)',
                  'Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified (R00-R99)',
                  'Diseases of heart (I00-I09,I11,I13,I20-I51)',
                  'Cerebrovascular diseases (I60-I69)',
                  'COVID-19 (U071, Multiple Cause of Death)',
                  'COVID-19 (U071, Underlying Cause of Death)', 'flag_allcause',
                  'flag_natcause', 'flag_sept', 'flag_neopl', 'flag_diab', 'flag_alz',
                  'flag_inflpn', 'flag_clrd', 'flag_otherresp', 'flag_nephr',
                  'flag_otherunk', 'flag_hd', 'flag_stroke', 'flag_cov19mcod',
                  'flag_cov19ucod'],
                 dtype='object')
```

Here we are going to rename some the columns headings

```
In [2931]: df.rename(columns={"Septicemia (A40-A41)":"Septicemia",
                              'Malignant neoplasms (C00-C97)':'Malignant neoplasms',
                              'Diabetes mellitus (E10-E14)':'Diabetes mellitus',
                              'Alzheimer disease (G30)':'Alzheimer disease',
                              'Influenza and pneumonia (J09-J18)':'Influenza and pneumonia',
                              'Chronic lower respiratory diseases (J40-J47)':'Chronic lower respiratory diseases',
                              'Other diseases of respiratory system (J00-J06,J30-J39,J67,J70-J98)':'Other diseases of respirato
                              'Nephritis, nephrotic syndrome and nephrosis (N00-N07,N17-N19,N25-N27)':'Nephritis',
                              'Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified (R00-R99
                              'Diseases of heart (I00-I09,I11,I13,I20-I51)':'Diseases of heart',
                              'Cerebrovascular diseases (I60-I69)':'Cerebrovascular diseases',
                              'COVID-19 (U071, Multiple Cause of Death)':'COVID-19_Multiple Cause of Death',
                              'COVID-19 (U071, Underlying Cause of Death)':'COVID-19_Underlying Cause of Death'}, inplace=True)

           #New columns headings
           df.columns

Out[2931]: Index(['Data As Of', 'Jurisdiction of Occurrence', 'MMWR Year', 'MMWR Week',
                  'Week Ending Date', 'All Cause', 'Natural Cause', 'Septicemia',
                  'Malignant neoplasms', 'Diabetes mellitus', 'Alzheimer disease',
                  'Influenza and pneumonia', 'Chronic lower respiratory diseases',
                  'Other diseases of respiratory system', 'Nephritis',
                  'not elsewhere classified', 'Diseases of heart',
                  'Cerebrovascular diseases', 'COVID-19_Multiple Cause of Death',
                  'COVID-19_Underlying Cause of Death', 'flag_allcause', 'flag_natcause',
                  'flag_sept', 'flag_neopl', 'flag_diab', 'flag_alz', 'flag_inflpn',
                  'flag_clrd', 'flag_otherresp', 'flag_nephr', 'flag_otherunk', 'flag_hd',
                  'flag_stroke', 'flag_cov19mcod', 'flag_cov19ucod'],
                 dtype='object')
```

The column names are changed

```
In [2932]: # After Renaming the columns we see the new columns names with Datatype for each column.
           df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10476 entries, 0 to 10475
Data columns (total 35 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Data As Of                              10476 non-null  object
 1   Jurisdiction of Occurrence              10476 non-null  object
 2   MMWR Year                               10476 non-null  int64
 3   MMWR Week                               10476 non-null  int64
 4   Week Ending Date                        10476 non-null  object
 5   All Cause                               10476 non-null  int64
 6   Natural Cause                           10476 non-null  int64
 7   Septicemia                              6083 non-null   float64
 8   Malignant neoplasms                     10466 non-null  float64
 9   Diabetes mellitus                       8234 non-null   float64
 10  Alzheimer disease                       8732 non-null   float64
 11  Influenza and pneumonia                 6241 non-null   float64
```

Second Method: Finding the Missing values in the each column

Second Method is Finding the missing values.

```
In [2933]: # To check for Missing values
           df.isnull().sum()

Out[2933]: Data As Of                                    0
           Jurisdiction of Occurrence                    0
           MMWR Year                                     0
           MMWR Week                                     0
           Week Ending Date                              0
           All Cause                                     0
           Natural Cause                                 0
           Septicemia                                 4393
           Malignant neoplasms                          10
           Diabetes mellitus                          2242
           Alzheimer disease                          1744
           Influenza and pneumonia                    4235
           Chronic lower respiratory diseases         1511
           Other diseases of respiratory system       4171
           Nephritis                                  3760
           not elsewhere classified                   4663
           Diseases of heart                            12
           Cerebrovascular diseases                   1445
           COVID-19_Multiple Cause of Death           1755
           COVID-19_Underlying Cause of Death         2296
           flag_allcause                             10476
           flag_natcause                             10476
           flag_sept                                  6083
           flag_neopl                                10466
           flag_diab                                  8234
           flag_alz                                   8732
           flag_inflpn                                6241
           flag_clrd                                  8965
           flag_otherresp                             6305
           flag_nephr                                 6716
           flag_otherunk                              5813
```

Dropping those columns when the columns values are missing more than 20 percent of the dataset.

Dropping some columns due more missing values than threshold value is 20%.

In [2934]:
```
# Calculate the threshold number of missing values(Here i have consider that if a column have more than 20% of missi
threshold_columns = int(0.2 * df.shape[0])

# Drop columns with more than the threshold number of missing values
df.dropna(axis=1, thresh=df.shape[0] - threshold_columns, inplace=True)

# Now columns with more than 20% missing values have been dropped from the DataFrame
#df = df.drop(df.columns[df.apply(lambda col: col.isna().sum() > 2000)], axis=1)
df.shape
```

Out[2934]:
```
(10476, 13)
```

In [2935]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10476 entries, 0 to 10475
Data columns (total 13 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Data As Of                          10476 non-null  object
 1   Jurisdiction of Occurrence          10476 non-null  object
 2   MMWR Year                           10476 non-null  int64
 3   MMWR Week                           10476 non-null  int64
 4   Week Ending Date                    10476 non-null  object
 5   All Cause                           10476 non-null  int64
 6   Natural Cause                       10476 non-null  int64
 7   Malignant neoplasms                 10466 non-null  float64
 8   Alzheimer disease                   8732 non-null   float64
 9   Chronic lower respiratory diseases  8965 non-null   float64
 10  Diseases of heart                   10464 non-null  float64
 11  Cerebrovascular diseases            9031 non-null   float64
 12  COVID-19_Multiple Cause of Death    8721 non-null   float64
dtypes: float64(6), int64(4), object(3)
memory usage: 1.0+ MB
```

After dropping the columns, The dataset contains 13 columns with 10476 rows.
Third Method is Imputation using to fill the missing values.

For the Missing values by using the imputation method to fill those values. By using the function “df.isnull().sum()” we will call the data frame and we can see that there zero null values.

Third Method is Imputation using to fill the missing values.

In [2936]:
```
#Here we have used the Mean value of each column and filled the missing values of that paticular column.
# Calculate the mean for each specified column
mean_malignant_neoplasms = df['Malignant neoplasms'].mean()
mean_alzheimer_disease = df['Alzheimer disease'].mean()
mean_chronic_respiratory_diseases = df['Chronic lower respiratory diseases'].mean()
mean_diseases_of_heart = df['Diseases of heart'].mean()
mean_cerebrovascular_diseases = df['Cerebrovascular diseases'].mean()
mean_covid_multiple_cause_of_death = df['COVID-19_Multiple Cause of Death'].mean()


# Fill missing values in each specified column with the respective mean
df['Malignant neoplasms'].fillna(mean_malignant_neoplasms, inplace=True)
df['Alzheimer disease'].fillna(mean_alzheimer_disease, inplace=True)
df['Chronic lower respiratory diseases'].fillna(mean_chronic_respiratory_diseases, inplace=True)
df['Diseases of heart'].fillna(mean_diseases_of_heart, inplace=True)
df['Cerebrovascular diseases'].fillna(mean_cerebrovascular_diseases, inplace=True)
df['COVID-19_Multiple Cause of Death'].fillna(mean_covid_multiple_cause_of_death, inplace=True)


# Now the DataFrame 'df' will have missing values in these columns filled with their respective means

df.isnull().sum()
```

Out[2936]:
```
Data As Of                          0
Jurisdiction of Occurrence          0
MMWR Year                           0
MMWR Week                           0
Week Ending Date                    0
All Cause                           0
Natural Cause                       0
Malignant neoplasms                 0
Alzheimer disease                   0
Chronic lower respiratory diseases  0
Diseases of heart                   0
Cerebrovascular diseases            0
COVID-19_Multiple Cause of Death    0
dtype: int64
```

“df.isnull().any()” using this function also we cross check for the finding the missing values. In the all columns it is showing the false so we can conclude that no missing values in the all columns.

```
In [2937]: # Here we see the null or missing values in the dataset
           df.isnull().any()

Out[2937]: Data As Of                            False
           Jurisdiction of Occurrence            False
           MMWR Year                             False
           MMWR Week                             False
           Week Ending Date                      False
           All Cause                             False
           Natural Cause                         False
           Malignant neoplasms                   False
           Alzheimer disease                     False
           Chronic lower respiratory diseases    False
           Diseases of heart                     False
           Cerebrovascular diseases              False
           COVID-19_Multiple Cause of Death      False
           dtype: bool
```

Fourth Method : Finding and deleting the Duplicate values.

Next step we are going to check is any Duplicate values present or not in the data frame. by using the function “df.duplicated(keep=False)”. In the below screenshot of the jupyter notebook we clearly say that there are no duplicated values.

Fourth Method is Finding and deleting the Duplicate values.

```
In [2938]: # Find duplicate values in both rows and columns
           duplicate_values = df[df.duplicated(keep=False)]

           # Display duplicate values if any are found
           if not duplicate_values.empty:
               print("Duplicate Values in Both Rows and Columns:")
               print(duplicate_values)
           else:
               print("No duplicate values found in both rows and columns.")

           df

           No duplicate values found in both rows and columns.
```

Out[2938]:

	Data As Of	Jurisdiction of Occurrence	MMWR Year	MMWR Week	Week Ending Date	All Cause	Natural Cause	Malignant neoplasms	Alzheimer disease	Chronic lower respiratory diseases	Diseases of heart	Cerebrovascular diseases	COVID-19_Multiple Cause of Death
0	09/27/2023	United States	2020	1	2020-01-04	60179	55010	11567.0	2537.0	3501.0	14204.0	3110.0	0.0
1	09/27/2023	United States	2020	2	2020-01-11	60735	55755	11963.0	2566.0	3708.0	13911.0	3189.0	1.0
2	09/27/2023	United States	2020	3	2020-01-18	59363	54516	11701.0	2491.0	3526.0	13593.0	3256.0	2.0
3	09/27/2023	United States	2020	4	2020-01-25	59162	54401	11879.0	2517.0	3403.0	13612.0	3185.0	3.0
4	09/27/2023	United States	2020	5	2020-02-01	58834	54001	11963.0	2480.0	3313.0	13465.0	3084.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
10471	09/27/2023	Puerto Rico	2023	33	2023-08-19	612	590	92.0	48.0	15.0	112.0	27.0	15.0

Fifth Method : Finding the unique values

Fifth Method

Finding the unquie values

```
In [2939]: #Finding the unquie values in column the "Jurisdiction of Occurrence"

           unique_values = df['Jurisdiction of Occurrence'].unique()

           # Print the unique values
           print(unique_values)

           ['United States' 'Alabama' 'Alaska' 'Arizona' 'Arkansas' 'California'
            'Colorado' 'Connecticut' 'Delaware' 'District of Columbia' 'Florida'
            'Georgia' 'Hawaii' 'Idaho' 'Illinois' 'Indiana' 'Iowa' 'Kansas'
            'Kentucky' 'Louisiana' 'Maine' 'Maryland' 'Massachusetts' 'Michigan'
            'Minnesota' 'Mississippi' 'Missouri' 'Montana' 'Nebraska' 'Nevada'
            'New Hampshire' 'New Jersey' 'New Mexico' 'New York' 'New York City'
            'North Carolina' 'North Dakota' 'Ohio' 'Oklahoma' 'Oregon' 'Pennsylvania'
            'Rhode Island' 'South Carolina' 'South Dakota' 'Tennessee' 'Texas' 'Utah'
            'Vermont' 'Virginia' 'Washington' 'West Virginia' 'Wisconsin' 'Wyoming'
            'Puerto Rico']
```

We have observed that New York and New York city so I have replaced the New York with New York city values. Rest of the values are the unique.

We can see that the columns Contains all States of USA. Except ('New York' 'New York City'). We are going to replace the New York City values to New York.

In [2940]:
```
# Assuming df is your DataFrame with the 'Jurisdiction of Occurrence' column
# Replace df with your actual DataFrame name

# Replace 'New York City' with 'New York'
df['Jurisdiction of Occurrence'].replace('New York City', 'New York', inplace=True)
#new unique values
unique_values = df['Jurisdiction of Occurrence'].unique()
# Count the occurrences of each state
state_counts = df['Jurisdiction of Occurrence'].value_counts()

# Print the unique values and their counts
print("Counts of each state:")
# Print the unique values
print(unique_values)
print(state_counts)
```

```
Counts of each state:
['United States' 'Alabama' 'Alaska' 'Arizona' 'Arkansas' 'California'
 'Colorado' 'Connecticut' 'Delaware' 'District of Columbia' 'Florida'
 'Georgia' 'Hawaii' 'Idaho' 'Illinois' 'Indiana' 'Iowa' 'Kansas'
 'Kentucky' 'Louisiana' 'Maine' 'Maryland' 'Massachusetts' 'Michigan'
 'Minnesota' 'Mississippi' 'Missouri' 'Montana' 'Nebraska' 'Nevada'
 'New Hampshire' 'New Jersey' 'New Mexico' 'New York' 'North Carolina'
 'North Dakota' 'Ohio' 'Oklahoma' 'Oregon' 'Pennsylvania' 'Rhode Island'
 'South Carolina' 'South Dakota' 'Tennessee' 'Texas' 'Utah' 'Vermont'
 'Virginia' 'Washington' 'West Virginia' 'Wisconsin' 'Wyoming'
 'Puerto Rico']
```

Sixth Method is changing the Format of the Date standard format (dd/mm/yyyy).

Sixth Method is changing the Format of the Date column¶

Here we have changed the date format into standard format dd/mm/yyyy.

In [2941]:
```
# Assuming df is your DataFrame with columns 'Data As Of' and 'Week Ending Date'
# Replace df with your actual DataFrame name

# Convert 'Data As Of' column to desired format
df['Data As Of'] = pd.to_datetime(df['Data As Of'], errors='coerce').dt.strftime('%d/%m/%Y')

# Convert 'Week Ending Date' column to desired format
df['Week Ending Date'] = pd.to_datetime(df['Week Ending Date'], errors='coerce').dt.strftime('%d/%m/%Y')

# Now both columns are in the "day/month/year" format
df
```

Out[2941]:

	Data As Of	Jurisdiction of Occurrence	MMWR Year	MMWR Week	Week Ending Date	All Cause	Natural Cause	Malignant neoplasms	Alzheimer disease	Chronic lower respiratory diseases	Diseases of heart	Cerebrovascular diseases	COVID-19_Multiple Cause of Death
0	27/09/2023	United States	2020	1	04/01/2020	60179	55010	11567.0	2537.0	3501.0	14204.0	3110.0	0.0
1	27/09/2023	United States	2020	2	11/01/2020	60735	55755	11963.0	2566.0	3708.0	13911.0	3189.0	1.0
2	27/09/2023	United States	2020	3	18/01/2020	59363	54516	11701.0	2491.0	3526.0	13593.0	3256.0	2.0
3	27/09/2023	United States	2020	4	25/01/2020	59162	54401	11879.0	2517.0	3403.0	13612.0	3185.0	3.0
4	27/09/2023	United States	2020	5	01/02/2020	58834	54001	11963.0	2480.0	3313.0	13465.0	3084.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
10471	27/09/2023	Puerto Rico	2023	33	19/08/2023	612	590	92.0	48.0	15.0	112.0	27.0	15.0

Seventh Method is changing the datatype for numeric Columns

Converting the datatype of columns having float64 to int64.This will make us easy way while performing the any regression analysis.

Seventh Method is changing the datatype for numeric values

from float64 to int64

In [375]:
```
import pandas as pd

# Assuming df is your DataFrame containing the data
# Replace df with the name of your DataFrame

# Columns to convert from float64 to int64
columns_to_convert = [
    'Malignant neoplasms',
    'Alzheimer disease',
    'Chronic lower respiratory diseases',
    'Diseases of heart',
    'Cerebrovascular diseases',
    'COVID-19_Multiple Cause of Death'
]

# Convert each column to int64
df[columns_to_convert] = df[columns_to_convert].astype('int64')

# Verify the changes
print(df.dtypes)
```

output :

```
# Convert each column to int64
df[columns_to_convert] = df[columns_to_convert].astype('int64')

# Verify the changes
print(df.dtypes)
```

```
Data As Of                            object
Jurisdiction of Occurrence            object
MMWR Year                              int64
MMWR Week                              int64
Week Ending Date                      object
All Cause                              int64
Natural Cause                          int64
Malignant neoplasms                    int64
Alzheimer disease                      int64
Chronic lower respiratory diseases     int64
Diseases of heart                      int64
Cerebrovascular diseases               int64
COVID-19_Multiple Cause of Death       int64
dtype: object
```

Eight Method is to remove leading and trailing spaces from values in all columns and column names.

Eight Method is to remove leading and trailing spaces from values in all columns

removing at column names and columns values

In [376]:
```
import pandas as pd

# Example DataFrame
data = df

df = pd.DataFrame(data)

# Remove leading and trailing spaces from column names
df.columns = df.columns.str.strip()

# Remove leading and trailing spaces from values in all columns
df = df.apply(lambda x: x.str.strip() if x.dtype == "object" else x)

# Print the resulting DataFrame
print(df)
```

```
       Data As Of Jurisdiction of Occurrence  MMWR Year  MMWR Week  \
0      27/09/2023               United States       2020          1
1      27/09/2023               United States       2020          2
2      27/09/2023               United States       2020          3
3      27/09/2023               United States       2020          4
4      27/09/2023               United States       2020          5
...           ...                         ...        ...        ...
10471  27/09/2023                 Puerto Rico       2023         33
10472  27/09/2023                 Puerto Rico       2023         34
10473  27/09/2023                 Puerto Rico       2023         35
10474  27/09/2023                 Puerto Rico       2023         36
10475  27/09/2023                 Puerto Rico       2023         37

      Week Ending Date  All Cause  Natural Cause  Malignant neoplasms  \
0           04/01/2020      60179          55010                11567
1           11/01/2020      60735          55755                11963
2           18/01/2020      59363          54516                11701
3           25/01/2020      59162          54401                11879
4           01/02/2020      58834          54001                11963
...                ...        ...            ...                  ...
10471       19/08/2023        612            590                   92
10472       26/08/2023        657            624                  110
10473       02/09/2023        580            552                   97
10474       09/09/2023        533            516                   81
10475       16/09/2023        453            453                   85

       Alzheimer disease  Chronic lower respiratory diseases  \
0                   2537                                3501
1                   2566                                3708
2                   2491                                3526
3                   2517                                3403
4                   2480                                3313
...                  ...                                 ...
10471                 48                                  15
10472                 64                                  15
10473                 62                                  14
10474                 47                                  19
10475                 25                                  13

       Diseases of heart  Cerebrovascular diseases  \
0                  14204                      3110
1                  13911                      3189
2                  13593                      3256
3                  13612                      3185
4                  13465                      3084
...                  ...                       ...
10471                112                        27
10472                120                        30
10473                 98                        21
10474                 99                        18
10475                 79                        21

       COVID-19_Multiple Cause of Death
0                                     0
1                                     1
2                                     2
3                                     3
4                                     0
...                                 ...
10471                                15
10472                                21
10473                                16
10474                                16
10475                                11
```

In [377]:
```
df
```

Out[377]:

	Data As Of	Jurisdiction of Occurrence	MMWR Year	MMWR Week	Week Ending Date	All Cause	Natural Cause	Malignant neoplasms	Alzheimer disease	Chronic lower respiratory diseases	Diseases of heart	Cerebrovascular diseases	COVID-19_Multiple Cause of Death
0	27/09/2023	United States	2020	1	04/01/2020	60179	55010	11567	2537	3501	14204	3110	0
1	27/09/2023	United States	2020	2	11/01/2020	60735	55755	11963	2566	3708	13911	3189	1
2	27/09/2023	United States	2020	3	18/01/2020	59363	54516	11701	2491	3526	13593	3256	2
3	27/09/2023	United States	2020	4	25/01/2020	59162	54401	11879	2517	3403	13612	3185	3
4	27/09/2023	United States	2020	5	01/02/2020	58834	54001	11963	2480	3313	13465	3084	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
10471	27/09/2023	Puerto Rico	2023	33	19/08/2023	612	590	92	48	15	112	27	15
10472	27/09/2023	Puerto Rico	2023	34	26/08/2023	657	624	110	64	15	120	30	21
10473	27/09/2023	Puerto Rico	2023	35	02/09/2023	580	552	97	62	14	98	21	16
10474	27/09/2023	Puerto Rico	2023	36	09/09/2023	533	516	81	47	19	99	18	16
10475	27/09/2023	Puerto Rico	2023	37	16/09/2023	453	453	85	25	13	79	21	11

10476 rows × 13 columns

3. Data Manipulation :

Performs data manipulation and feature engineering tasks on a Data Frame:

Extracting Year and Week: Extracts the year and week information from the 'Week Ending Date' column.

Calculating Days Since Last Week: Computes the number of days since the last week based on the difference between consecutive dates in the 'Week Ending Date' column.

Computing Total Deaths: Calculates the total number of deaths by summing up columns related to causes of death.

Calculating Cause-specific Mortality Rates: Computes mortality rates for specific causes of death by dividing the count of each cause by the total number of deaths and multiplying by 100.

Calculating Ratios: Calculates ratios between specific causes of death and the total number of deaths.

Identifying and Encoding Seasonal Patterns: Maps the 'Season' column to human-readable seasonal patterns (Winter, Spring, Summer, Autumn).

These operations aim to pre-process the data, extract meaningful features, and derive additional insights for further analysis or modelling.

```
In [463]: import pandas as pd

# Assuming cleaned_df is your DataFrame with the provided columns
df = df

# Convert 'Week Ending Date' to datetime with the appropriate format
df['Week Ending Date'] = pd.to_datetime(df['Week Ending Date'], format='%d/%m/%Y')

# Feature Engineering
df['Year'] = df['Week Ending Date'].dt.year
df['Week'] = df['Week Ending Date'].dt.isocalendar().week

# Days since last week (assuming data is sorted by date)
df['Days_Since_Last_Week'] = df['Week Ending Date'].diff().dt.days.fillna(0)

# Total Deaths
# Sum up all the columns related to causes of death
df['Total_Deaths'] = df[['All Cause', 'Natural Cause', 'Malignant neoplasms', 'Alzheimer disease', 'Chronic lower re

# Cause-specific Mortality Rates (examples)
df['Malignancy_Mortality_Rate'] = df['Malignant neoplasms'] / df['Total_Deaths'] * 100
df['Heart_Disease_Mortality_Rate'] = df['Diseases of heart'] / df['Total_Deaths'] * 100

# Ratio (example)
df['Natural_Cause_Prop'] = cleaned_df['Natural Cause'] / df['Total_Deaths']

df['Month'] = df['Week Ending Date'].dt.month
# Assuming 'Season' is derived from 'Month' column
df['Season'] = df['Month'].apply(lambda x: (x%12 + 3)//3)
# Identifying and encoding seasonal patterns
season_map = {1: 'Winter', 2: 'Spring', 3: 'Summer', 4: 'Autumn'}
df['Seasonal Pattern'] = df['Season'].map(season_map)

print(df)
```

```
      Data As Of Jurisdiction of Occurrence  MMWR Year  MMWR Week  \
0      27/09/2023              United States       2020          1
1      27/09/2023              United States       2020          2
2      27/09/2023              United States       2020          3
3      27/09/2023              United States       2020          4
4      27/09/2023              United States       2020          5
...           ...                        ...        ...        ...
10471  27/09/2023                Puerto Rico       2023         33
10472  27/09/2023                Puerto Rico       2023         34
10473  27/09/2023                Puerto Rico       2023         35
10474  27/09/2023                Puerto Rico       2023         36
10475  27/09/2023                Puerto Rico       2023         37

      Week Ending Date  All Cause  Natural Cause  Malignant neoplasms  \
0           2020-01-04      60179          55010                11567
1           2020-01-11      60735          55755                11963
2           2020-01-18      59363          54516                11701
3           2020-01-25      59162          54401                11879
4           2020-02-01      58834          54001                11963
...                ...        ...            ...                  ...
10471       2023-08-19        612            590                   92
10472       2023-08-26        657            624                  110
10473       2023-09-02        580            552                   97
10474       2023-09-09        533            516                   81
10475       2023-09-16        453            453                   85

      Alzheimer disease  Chronic lower respiratory diseases  ...  Year  Week  \
0                  2537                                3501  ...  2020     1
1                  2566                                3708  ...  2020     2
2                  2491                                3526  ...  2020     3
3                  2517                                3403  ...  2020     4
4                  2480                                3313  ...  2020     5
...                 ...                                 ...  ...   ...   ...
10471                48                                  15  ...  2023    33
```

```
       Days_Since_Last_Week  Total_Deaths  Malignancy_Mortality_Rate  \
0                       0.0        150108                   7.705785
1                       7.0        151828                   7.879311
2                       7.0        148448                   7.882221
3                       7.0        148162                   8.017575
4                       7.0        147140                   8.130352
...                     ...           ...                        ...
10471                   7.0          1511                   6.088683
10472                   7.0          1641                   6.703230
10473                   7.0          1440                   6.736111
10474                   7.0          1329                   6.094808
10475                   7.0          1140                   7.456140

       Heart_Disease_Mortality_Rate  Natural_Cause_Prop  Month  Season  \
0                          9.462520            0.366469      1       1
1                          9.162342            0.367225      1       1
2                          9.156742            0.367240      1       1
3                          9.187241            0.367172      1       1
4                          9.151149            0.367004      2       1
...                             ...                 ...    ...     ...
10471                      7.412310            0.390470      8       3
10472                      7.312614            0.380256      8       3
10473                      6.805556            0.383333      9       4
10474                      7.449210            0.388262      9       4
10475                      6.929825            0.397368      9       4

       Seasonal Pattern
0                Winter
1                Winter
2                Winter
3                Winter
4                Winter
...                 ...
10471            Summer
10472            Summer
10473            Autumn
10474            Autumn
10475            Autumn
```

In [464]: `df`

Out[464]:

...	Year	Week	Days_Since_Last_Week	Total_Deaths	Malignancy_Mortality_Rate	Heart_Disease_Mortality_Rate	Natural_Cause_Prop	Month	Season	Seasonal Pattern
...	2020	1	0.0	150108	7.705785	9.462520	0.366469	1	1	Winter
...	2020	2	7.0	151828	7.879311	9.162342	0.367225	1	1	Winter
...	2020	3	7.0	148448	7.882221	9.156742	0.367240	1	1	Winter
...	2020	4	7.0	148162	8.017575	9.187241	0.367172	1	1	Winter
...	2020	5	7.0	147140	8.130352	9.151149	0.367004	2	1	Winter
...	...	...	...	...	...	...	...	...	...	...
...	2023	33	7.0	1511	6.088683	7.412310	0.390470	8	3	Summer
...	2023	34	7.0	1641	6.703230	7.312614	0.380256	8	3	Summer
...	2023	35	7.0	1440	6.736111	6.805556	0.383333	9	4	Autumn
...	2023	36	7.0	1329	6.094808	7.449210	0.388262	9	4	Autumn
...	2023	37	7.0	1140	7.456140	6.929825	0.397368	9	4	Autumn

In [465]: `df.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10476 entries, 0 to 10475
Data columns (total 23 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Data As Of                            10476 non-null  object
 1   Jurisdiction of Occurrence            10476 non-null  object
 2   MMWR Year                             10476 non-null  int64
 3   MMWR Week                             10476 non-null  int64
 4   Week Ending Date                      10476 non-null  datetime64[ns]
 5   All Cause                             10476 non-null  int64
 6   Natural Cause                         10476 non-null  int64
 7   Malignant neoplasms                   10476 non-null  int64
 8   Alzheimer disease                     10476 non-null  int64
 9   Chronic lower respiratory diseases    10476 non-null  int64
 10  Diseases of heart                     10476 non-null  int64
 11  Cerebrovascular diseases              10476 non-null  int64
 12  COVID-19_Multiple Cause of Death      10476 non-null  int64
 13  Year                                  10476 non-null  int32
 14  Week                                  10476 non-null  UInt32
 15  Days_Since_Last_Week                  10476 non-null  float64
 16  Total_Deaths                          10476 non-null  int64
 17  Malignancy_Mortality_Rate             10476 non-null  float64
 18  Heart_Disease_Mortality_Rate          10476 non-null  float64
 19  Natural_Cause_Prop                    10476 non-null  float64
 20  Month                                 10476 non-null  int32
 21  Season                                10476 non-null  int64
 22  Seasonal Pattern                      10476 non-null  object
dtypes: UInt32(1), datetime64[ns](1), float64(4), int32(2), int64(12), object(3)
memory usage: 1.7+ MB
```

we have used this Engineering feature method to create dependent column from using the original dataset columns.

4. Data Analysis:

The Uncovering Trends in Weekly Provisional Counts of Deaths has generated a wealth of data, but its true value lies in transforming it into actionable intelligence. As a data analyst, I'm tackling this challenge by analysing classifications and causes of U.S. COVID-19 cases (ICD-10 code U07.1). This analysis aims to distill this complex data into a clear picture that informs strategic public health actions.

Descriptive Statistics: Measures like mean, median, and standard deviation will reveal central tendencies and data spread, providing a foundational understanding.

Correlation Analysis: We'll explore relationships between key variables like control measures, mortality rates, and other causes of death. This will help identify potential connections and patterns.

Multiple Linear Regression: This technique will model the relationship between Total Deaths and other relevant factors, allowing us to understand their influence.By uncovering trends in control measures, mortality rates, and other factors, this analysis aims to inform targeted interventions and strategies to better address emerging challenges. Ultimately, this data-driven approach seeks to transform complex data into actionable insights that can help protect public health

Descriptive Statistics:

- **Overall Mortality:** The average (mean) number of deaths across all causes is quite high (around 6,060). There's significant variation in the data, as indicated by the large standard deviation (over 21,600).
- **Causes of Death:** Malignant neoplasms (cancers) have the highest average mortality (around 432), followed by heart diseases (around 493) and COVID-19 related deaths (around 263).
- **Temporal Trends:** The data likely spans multiple years (MMWR Year: 2020-2023). Days since the last week are typically low (mean: 0.13 weeks), suggesting the data might be collected frequently.
- **Seasonality:** There's a potential seasonal component (Season: 1-4) present in the data.

Code :

Data Analysis

Descriptive Statistics:

Calculate mean, median, mode, standard deviation and variance

In [487]:
```
import pandas as pd

# Assuming your DataFrame is named 'df'http://localhost:8888/notebooks/Balaji_Pamidi_Final_project_002644804.ipynb#C

# Select numeric columns
numeric_columns = df.select_dtypes(include=['int64', 'float64'])

# Calculate descriptive statistics
descriptive_stats = numeric_columns.describe()

# Calculate mode
#mode = numeric_columns.mode()

# Loop through each numeric column and calculate mode separately
for column in numeric_columns:
    mode = numeric_columns[column].mode()
    print(f"Mode for {column}: {mode.tolist()}")

# Calculate variance
variance = numeric_columns.var()


# Display the results
print("Descriptive Statistics:")
print(descriptive_stats)

#print("\nMode:")
#print(mode)

print("\nVariance:")
print(variance)
```

Output : Descriptive Statistics

```
Descriptive Statistics:
          MMWR Year     MMWR Week     All Cause  Natural Cause  \
count  10476.000000  10476.000000  10476.000000   10476.000000
mean    2021.376289     25.206186   2362.293910    2151.945208
std        1.078259     14.682156   8552.044924    7799.519725
min     2020.000000      1.000000     12.000000      12.000000
25%     2020.000000     13.000000    366.750000     331.000000
50%     2021.000000     25.000000    931.500000     837.000000
75%     2022.000000     37.000000   1548.000000    1405.000000
max     2023.000000     53.000000  87415.000000   81622.000000

       Malignant neoplasms  Alzheimer disease  \
count         10476.000000       10476.000000
mean            431.874380         103.788183
std            1550.675559         313.861633
min              10.000000           0.000000
25%              66.000000          25.000000
50%             171.000000          47.000000
75%             282.000000         103.000000
max           12284.000000        3075.000000

       Chronic lower respiratory diseases  Diseases of heart  \
count                        10476.000000       10476.000000
mean                           121.146334         493.532837
std                            375.488612        1777.161012
min                              0.000000          10.000000
25%                             31.000000          72.000000
50%                             56.000000         191.000000
75%                            114.000000         329.000000
max                           3708.000000       16538.000000

       Cerebrovascular diseases  COVID-19_Multiple Cause of Death  \
count              10476.000000                      10476.000000
mean                 133.053169                        262.532932
std                  415.103496                       1160.062676
min                    0.000000                          0.000000
25%                   29.000000                         27.000000
50%                   56.000000                         75.500000
75%                  126.000000                        262.000000

        Days_Since_Last_Week   Total_Deaths  Malignancy_Mortality_Rate  \
count          10476.000000   10476.000000                10476.000000
mean               0.128961    6060.166953                    6.634836
std               96.351661   21635.364667                    1.984468
min            -1351.000000     340.000000                    1.002506
25%                7.000000    1056.000000                    5.640041
50%                7.000000    2348.500000                    7.042898
75%                7.000000    3903.500000                    7.877299
max                7.000000  231114.000000                   46.746204

       Heart_Disease_Mortality_Rate  Natural_Cause_Prop        Season
count                  10476.000000        10476.000000  10476.000000
mean                       7.506880            0.324006      2.453608
std                        2.423096            0.072995      1.084341
min                        0.945830            0.011101      1.000000
25%                        6.518929            0.324171      2.000000
50%                        7.904952            0.357918      2.000000
75%                        8.808975            0.363222      3.000000
max                       68.854749            0.397368      4.000000
```

Mode :

This summary highlights the most frequent values (modes) for various variables in the dataset:

- **Time:** The data likely covers multiple years, with 2020 being the most frequent year (MMWR Year). Weeks are spread throughout the year, with no single week appearing most often (MMWR Week).
- **Mortality:** The most common number of deaths across all causes is relatively low (125), but there's significant variation (refer to descriptive statistics for details). Similar patterns hold for deaths from specific causes (Natural Cause, Malignant neoplasms, etc.).
- **Other Variables:** Days since the last week tend to be low (mode: 7.0), suggesting frequent data collection. Specific values appear most often for Malignancy_Mortality_Rate, Heart_Disease_Mortality_Rate, and Natural_Cause_Prop, but these might require further investigation in the context of your analysis. Seasonality seems to be present, with spring (Season: 2) and summer (Season: 3) being the most frequent seasons.

```
Mode for MMWR Year: [2020]
Mode for MMWR Week: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37]
Mode for All Cause: [125]
Mode for Natural Cause: [106]
Mode for Malignant neoplasms: [22]
Mode for Alzheimer disease: [103]
Mode for Chronic lower respiratory diseases: [121]
Mode for Diseases of heart: [30]
Mode for Cerebrovascular diseases: [133]
Mode for COVID-19_Multiple Cause of Death: [262]
Mode for Days_Since_Last_Week: [7.0]
Mode for Total_Deaths: [842]
Mode for Malignancy_Mortality_Rate: [6.25, 7.142857142857142, 7.6923076923076925]
Mode for Heart_Disease_Mortality_Rate: [7.142857142857142, 9.090909090909092]
Mode for Natural_Cause_Prop: [0.36363636363636365]
Mode for Season: [2, 3]
```

Variance:

- **High Variance:** Variables like "All Cause," "Total Deaths," and "Days Since Last Week" have very high variances. This implies a significant spread in the number of deaths and the timeliness of reporting across different data points.
- **Moderate Variance:** Variables related to specific causes of death (e.g., "Malignant neoplasms," "Diseases of heart") show moderate variance. This suggests some variation in these mortalities, but not as extreme as the overall death count.
- **Low Variance:** Variables like "MMWR Year," "Season," and "Natural Cause Prop" have relatively low variance. This indicates these values tend to be clustered around their respective means.

```
Variance:
MMWR Year                              1.162642e+00
MMWR Week                              2.155657e+02
All Cause                              7.313747e+07
Natural Cause                          6.083251e+07
Malignant neoplasms                    2.404595e+06
Alzheimer disease                      9.850912e+04
Chronic lower respiratory diseases     1.409917e+05
Diseases of heart                      3.158301e+06
Cerebrovascular diseases               1.723109e+05
COVID-19_Multiple Cause of Death       1.345745e+06
Days_Since_Last_Week                   9.283643e+03
Total_Deaths                           4.680890e+08
Malignancy_Mortality_Rate              3.938114e+00
Heart_Disease_Mortality_Rate           5.871395e+00
Natural_Cause_Prop                     5.328327e-03
Season                                 1.175795e+00
dtype: float64
```

Correlation :

Summary of Correlation Analysis :

Strong Correlations: High mortality in one category (total deaths, natural causes) is linked to high mortality in others (diseases, cancers). Deaths from specific causes (heart disease, cancers) are also strongly linked, suggesting common risk factors.

Moderate Correlations: COVID-19 deaths show some connection to deaths from other respiratory diseases. Mortality rates for heart disease and cancers are moderately linked to the proportion of natural cause deaths. Week and season have a moderate correlation, reflecting data collection patterns.

Weak Correlations: Year has little impact on data collection within a week or season. The mortality rate for COVID-19 shows weak or no relationship to the mortality rate for cancers.

code :

Correlation

In [466]:
```
# Select only numeric columns
numeric_columns = df.select_dtypes(include=['int64', 'float64','UInt32'])

# Calculate correlation
correlation_matrix = numeric_columns.corr()

# Display correlation matrix
print(correlation_matrix)


# Finding the strongest correlation
strongest_corr = correlation_matrix.unstack().sort_values(ascending=False).drop_duplicates()
print("\nStrongest Correlation:")
print(strongest_corr[:30])  # Displaying the top 5 strongest correlations

# Finding the moderate correlation (between 0.3 and 0.7)
moderate_corr = strongest_corr[(strongest_corr < 0.7) & (strongest_corr >= 0.3)]
print("\nModerate Correlation:")
print(moderate_corr)

# Finding the weakest correlation
weakest_corr = correlation_matrix.unstack().sort_values().drop_duplicates()
print("\nWeakest Correlation:")
print(weakest_corr[:5])  # Displaying the top 5 weakest correlations
```

output :

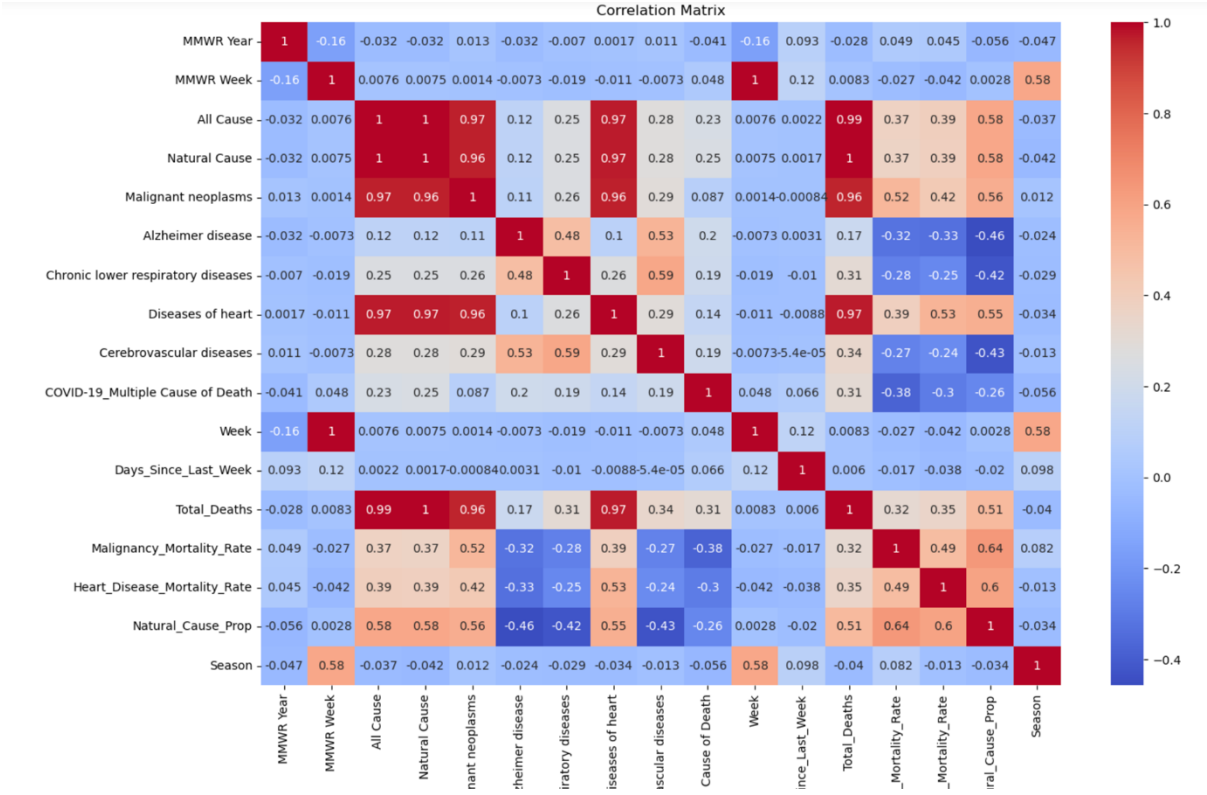
```
                                    MMWR Year  MMWR Week  All Cause  \
MMWR Year                            1.000000  -0.158600  -0.010860
MMWR Week                           -0.158600   1.000000   0.000712
All Cause                           -0.010860   0.000712   1.000000
Natural Cause                       -0.011232   0.000617   0.999878
Malignant neoplasms                  0.000507   0.000387   0.992298
Alzheimer disease                   -0.014961  -0.001048   0.988653
Chronic lower respiratory diseases  -0.004665  -0.006989   0.985634
Diseases of heart                   -0.003930  -0.003451   0.996925
Cerebrovascular diseases             0.000523  -0.002440   0.989995
COVID-19_Multiple Cause of Death    -0.056272   0.012200   0.756529
Week                                -0.158600   1.000000   0.000712
Days_Since_Last_Week                 0.091031   0.117582   0.010096
Total_Deaths                        -0.011934   0.000719   0.999752
Malignancy_Mortality_Rate            0.064687  -0.024684   0.076491
Heart_Disease_Mortality_Rate         0.051316  -0.044226   0.079460
Natural_Cause_Prop                  -0.053027   0.002755   0.127399
Season                              -0.044591   0.577298  -0.012889

                                    Natural Cause  Malignant neoplasms  \
```

```
Strongest Correlation:
MMWR Year                           MMWR Year                           1.000000
Total_Deaths                        Natural Cause                       0.999881
All Cause                           Natural Cause                       0.999878
Total_Deaths                        All Cause                           0.999752
Diseases of heart                   Malignant neoplasms                 0.997139
All Cause                           Diseases of heart                   0.996925
Natural Cause                       Diseases of heart                   0.996135
Total_Deaths                        Diseases of heart                   0.995289
Diseases of heart                   Cerebrovascular diseases            0.992986
Malignant neoplasms                 Cerebrovascular diseases            0.992359
All Cause                           Malignant neoplasms                 0.992298
Chronic lower respiratory diseases  Cerebrovascular diseases            0.991540
Diseases of heart                   Chronic lower respiratory diseases  0.991100
Malignant neoplasms                 Natural Cause                       0.991056
All Cause                           Cerebrovascular diseases            0.989995
Malignant neoplasms                 Total_Deaths                        0.989730
Cerebrovascular diseases            Alzheimer disease                   0.989455
                                    Natural Cause                       0.989117
Total_Deaths                        Cerebrovascular diseases            0.988786
                                    Alzheimer disease                   0.988711
Malignant neoplasms                 Chronic lower respiratory diseases  0.988655
Alzheimer disease                   All Cause                           0.988653
Natural Cause                       Alzheimer disease                   0.988531
Alzheimer disease                   Diseases of heart                   0.988065
Chronic lower respiratory diseases  Alzheimer disease                   0.987334
All Cause                           Chronic lower respiratory diseases  0.985634
Chronic lower respiratory diseases  Natural Cause                       0.984874
Alzheimer disease                   Malignant neoplasms                 0.984426
Chronic lower respiratory diseases  Total_Deaths                        0.984151
COVID-19_Multiple Cause of Death    Total_Deaths                        0.769791
dtype: float64


Moderate Correlation:
Cerebrovascular diseases          COVID-19_Multiple Cause of Death      0.698365
COVID-19_Multiple Cause of Death  Chronic lower respiratory diseases    0.681292
Malignant neoplasms               COVID-19_Multiple Cause of Death      0.674407
Malignancy_Mortality_Rate         Natural_Cause_Prop                    0.630406
Natural_Cause_Prop                Heart_Disease_Mortality_Rate          0.601936
MMWR Week                         Season                                0.577298
Malignancy_Mortality_Rate         Heart_Disease_Mortality_Rate          0.494980
dtype: float64

Weakest Correlation:
MMWR Year                         Week                        -0.158600
COVID-19_Multiple Cause of Death  Malignancy_Mortality_Rate   -0.066400
                                  MMWR Year                   -0.056272
Natural_Cause_Prop                MMWR Year                   -0.053027
Season                            MMWR Year                   -0.044591
dtype: float64
```

We can see this correlation through sns plot.

```
In [490]: # Heatmap for correlation matrix
          plt.figure(figsize=(15, 10))
          sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
          plt.title("Correlation Matrix")
          plt.show()
```

Heatmap of the Correlation between the columns.



Multiple Linear Regression Analysis :

Using sklearn and using statsmodels from the python libraries.
Here I have chosen this(X) predictor columns based correlation these columns are independent variables.(y) here is the Total_Deaths column which is dependent variable from the (Cerebrovascular diseases, Alzheimer disease , Chronic lower respiratory diseases).

Correlation between the (y) and (X) and there is no collinearity between the variables.

(y)	(X)	
Total_Deaths	Cerebrovascular diseases	0.988786
	Alzheimer disease	0.988711
	Chronic lower respiratory diseases	0.984151

```
In [467]: import pandas as pd
          import statsmodels.api as sm

          # Assuming cleaned_df is your DataFrame with the provided columns
          # Select predictor variables and outcome variable
          predictor_cols = ['Cerebrovascular diseases','Alzheimer disease','Chronic lower respiratory diseases']

          outcome_col = ['Total_Deaths']

          # Create a DataFrame with predictor variables
          X = df[predictor_cols]

          # Add a constant term to the predictor variables (for intercept)
          X = sm.add_constant(X)

          # Create a Series with the outcome variable
          y = df[outcome_col]

          # Concatenate X and y along the columns axis
          df_copy = pd.concat([X, y], axis=1)

          # Print the concatenated DataFrame
          print(df_copy.head(5))

          # Fit the multiple linear regression model
          model = sm.OLS(y, X).fit()

          # Print the model summary
          print(model.summary())
```

Output :
Data frame of the X and y Variables .

	const	Cerebrovascular diseases	Alzheimer disease	\
0	1.0	3110	2537	
1	1.0	3189	2566	
2	1.0	3256	2491	
3	1.0	3185	2517	
4	1.0	3084	2480	

	Chronic lower respiratory diseases	Total_Deaths
0	3501	150108
1	3708	151828
2	3526	148448
3	3403	148162
4	3313	147140

OLS regression results, we can draw several conclusions:

High explanatory power: R-squared of 0.983 implies 98.3% of Total_Deaths variability is explained by included variables (Cerebrovascular diseases, Alzheimer disease, and Chronic lower respiratory diseases).

Statistical significance: All coefficients are highly significant ($p < 0.0001$), providing strong evidence against the null hypothesis that coefficients are zero.

Positive relationships: Positive coefficients for Cerebrovascular diseases, Alzheimer disease, and Chronic lower respiratory diseases indicate higher levels of these diseases correlate with increased Total_Deaths.

Relative importance: Alzheimer disease's coefficient (33.1267) surpasses Cerebrovascular diseases' (24.5240), implying Alzheimer disease might have a stronger impact on Total_Deaths in this model.

Model fit: High F-statistic (1.999e+05) with p-value effectively zero demonstrates joint significance of independent variables in explaining Total_Deaths.

OLS Regression Results

Dep. Variable:	Total_Deaths	R-squared:	0.983
Model:	OLS	Adj. R-squared:	0.983
Method:	Least Squares	F-statistic:	1.999e+05
Date:	Wed, 10 Apr 2024	Prob (F-statistic):	0.00
Time:	14:06:02	Log-Likelihood:	-98145.
No. Observations:	10476	AIC:	1.963e+05
Df Residuals:	10472	BIC:	1.963e+05
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-942.0375	29.203	-32.259	0.000	-999.280	-884.795
Cerebrovascular diseases	24.5240	0.597	41.073	0.000	23.354	25.694
Alzheimer disease	33.1267	0.646	51.273	0.000	31.860	34.393
Chronic lower respiratory diseases	2.4850	0.603	4.124	0.000	1.304	3.666

Omnibus:	5801.986	Durbin-Watson:	0.224
Prob(Omnibus):	0.000	Jarque-Bera (JB):	477771.487
Skew:	1.795	Prob(JB):	0.00
Kurtosis:	35.889	Cond. No.	709.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Next we are going to build model by and splitting the dataset into train and test.

Model Building

Training Models

In [468]:
```
#split the dataset in training set and test set

X_train, X_test, y_train, y_test = train_test_split(df[predictor_cols], cleaned_df[outcome_col], test_size=0.3, rand

# Convert y_train and y_test to DataFrame with a single column
y_train = pd.DataFrame(y_train, columns=outcome_col)
y_test = pd.DataFrame(y_test, columns=outcome_col)

print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (7333, 3)
X_test shape: (3143, 3)
y_train shape: (7333, 1)
y_test shape: (3143, 1)
```

Fitting the Model and calling the coefficients and intercept value.

```
y_test shape: (3143, 1)
```

Using the sklearn we are going to implement the Multiple linear regression

In [469]:
```
from sklearn.linear_model import LinearRegression

# Create an instance of the LinearRegression class
lin_reg = LinearRegression()

# Fit the model using the training data
lin_reg.fit(X_train, y_train)

# After fitting, you can access the coefficients and intercept of the model
coefficients = lin_reg.coef_
intercept = lin_reg.intercept_

# Print the coefficients and intercept
print("Coefficients:", coefficients)
print("Intercept:", intercept)
```

```
Coefficients: [[ 2.74878287e+01  3.27094228e+01 -3.39925188e-03]]
Intercept: [-970.51379839]
```

Making the predictions for the X_test data.

Making Predictions

In [495]:
```
# Make predictions on the testing data
y_pred = lin_reg.predict(X_test)

# Print the first few predictions
print("y_pred:", y_pred[:5])
```

```
y_pred: [[ 2967.00376817]
 [11613.61714689]
 [  458.43647989]
 [ 2838.30761808]
 [ 2956.61836722]]
```

Evaluating the model predict by using the values from the data frame.

Evaluating Model Predict

In [496]:
```
# Use model to predict

lin_reg.predict([[3110,2537,3501]])
```

Out[496]:
```
array([[167488.53813431]])
```

Based on the r2_score we can predict the values up to 98% .

R2_Score

```
In [497]: from sklearn.metrics import r2_score
          r2_score(y_test,y_pred)

Out[497]: 0.9837079490658794
```

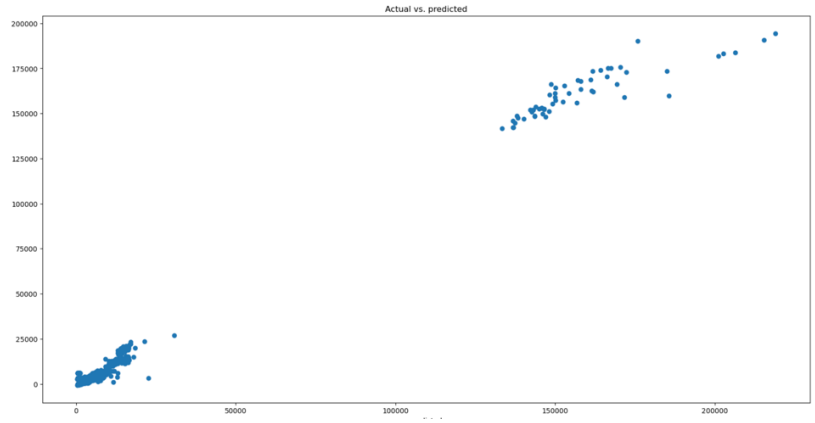
Plotting the Actual and predicated values using the matplotlib

Plot the results using matplotlib

```
In [498]: #plot the results
          import matplotlib.pyplot as plt
          plt.figure(figsize=(20,10))
          plt.scatter(y_test,y_pred)
          plt.xlabel('Actual')
          plt.xlabel('predicted')
          plt.title('Actual vs. predicted')

Out[498]: Text(0.5, 1.0, 'Actual vs. predicted')
```

By observing the plot there are so many outliers. we can improve the model by removing the outliers and using the better features to improve this model.

```
In [498]: #plot the results
          import matplotlib.pyplot as plt
          plt.figure(figsize=(20,10))
          plt.scatter(y_test,y_pred)
          plt.xlabel('Actual')
          plt.xlabel('predicted')
          plt.title('Actual vs. predicted')

Out[498]: Text(0.5, 1.0, 'Actual vs. predicted')
```



We can see the difference between the actual and predict values of the y_test. There are so many negative values and large values we can consider them as outliers.

Differnce between the test and predicted values

```
In [474]: # Convert y_test and y_pred to one-dimensional arrays
          y_test_1d = y_test.values.flatten()
          y_pred_1d = y_pred.flatten()

          # Create DataFrame with one-dimensional arrays
          pred_y_df = pd.DataFrame({'Actual Value': y_test_1d, 'Predicted Value': y_pred_1d, 'Difference': y_test_1d - y_pred_
          pred_y_df.head(20)
```

Out[474]:

	Actual Value	Predicted Value	Difference
0	5119	2967.003768	2151.996232
1	11407	11613.617147	-206.617147
2	1553	458.436480	1094.563520
3	937	2838.307618	-1901.307618
4	3822	2956.618367	865.381633
5	1480	360.315010	1119.684990
6	1387	-110.775623	1497.775623
7	2883	1711.046753	1171.953247
8	1872	824.878974	1047.121026
9	992	6054.373373	-5062.373373
10	1057	6054.026650	-4997.026650
11	2351	1290.734416	1060.265584
12	734	-67.623012	801.623012
13	916	168.980933	747.019067
14	18082	15014.939513	3067.060487

After removing the outliers we are going the predict the model is there any further removing.

Removing the Outliers

In [475]:
```
import pandas as pd
import numpy as np

# Load your dataset (replace 'your_dataset.csv' with the actual path to your dataset)
df_uncleaned = df

# Make a copy of the original DataFrame to work with
df_uncleaned = df.copy()

# Specify the numerical columns in your dataset
numeric_columns = [

   'Cerebrovascular diseases',
    'Alzheimer disease',
    'Chronic lower respiratory diseases',
    'Total_Deaths'
]

# Function to detect outliers using IQR method
def detect_outliers_iqr(data):
    Q1 = np.percentile(data, 25)
    Q3 = np.percentile(data, 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = (data < lower_bound) | (data > upper_bound)
    return outliers

# Detect outliers in each numerical column
outliers = {}
for col in numeric_columns:
    outliers[col] = detect_outliers_iqr(df_uncleaned[col])

# Print count of outliers for each column
for col, is_outlier in outliers.items():
# Detect outliers in each numerical column
outliers = {}
for col in numeric_columns:
    outliers[col] = detect_outliers_iqr(df_uncleaned[col])

# Print count of outliers for each column
for col, is_outlier in outliers.items():
    num_outliers = sum(is_outlier)
    print(f"Outliers detected in '{col}': {num_outliers}")

# Create a copy of the DataFrame for trimming outliers
df_trimmed = df_uncleaned.copy()

# Filter out rows containing outliers
for col, is_outlier in outliers.items():
    df_trimmed = df_trimmed.loc[~is_outlier]

# Reset the index of the trimmed DataFrame
df_trimmed.reset_index(drop=True, inplace=True)

# Print count of rows after removing outliers
print(f"Count of rows after removing outliers: {len(df_trimmed)}")

# Print count of values in each column after trimming outliers
for col in df_trimmed.columns:
    print(f"Count of values in '{col}': {df_trimmed[col].count()}")
```

```
Outliers detected in 'Cerebrovascular diseases': 581
Outliers detected in 'Alzheimer disease': 438
Outliers detected in 'Chronic lower respiratory diseases': 314
Outliers detected in 'Total_Deaths': 861
Count of rows after removing outliers: 9615
```

Multiple linear regression after removing the outliers

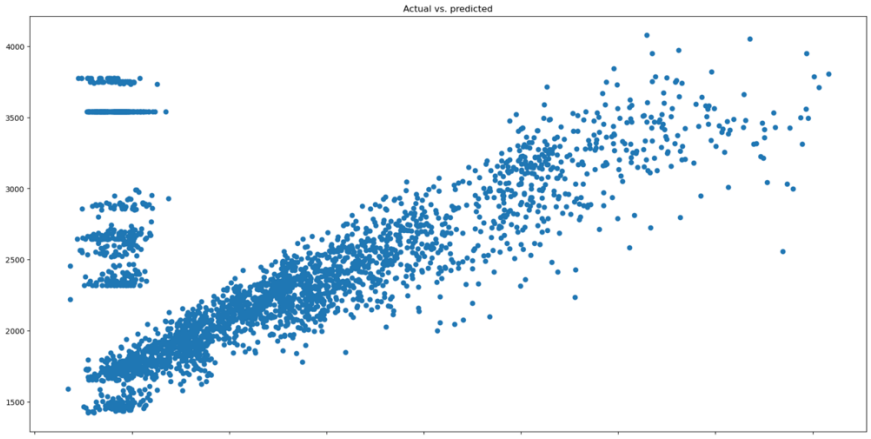
Multiple linear regression after removing the outliers

In [477]:
```
import pandas as pd
import statsmodels.api as sm

# Assuming cleaned_df is your DataFrame with the provided columns
# Select predictor variables and outcome variable
predictor_cols = ['Cerebrovascular diseases','Alzheimer disease','Chronic lower respiratory diseases']

outcome_col = ['Total_Deaths']

# Create a DataFrame with predictor variables
X = df_trimmed[predictor_cols]

# Add a constant term to the predictor variables (for intercept)
X = sm.add_constant(X)

# Create a Series with the outcome variable
y = df_trimmed[outcome_col]

# Concatenate X and y along the columns axis
df_copy = pd.concat([X, y], axis=1)

# Print the concatenated DataFrame
print(df_copy.head(5))

# Fit the multiple linear regression model
model = sm.OLS(y, X).fit()

# Print the model summary
print(model.summary())
```

```
   const  Cerebrovascular diseases  Alzheimer disease  \
0    1.0                        81                 54
1    1.0                        68                 41
2    1.0                        45                 53
3    1.0                        70                 54
4    1.0                        55                 58

   Chronic lower respiratory diseases  Total_Deaths
0                                  86          2776
1                                  72          2839
2                                  73          2626
3                                  61          2959
4                                  76          2586
                            OLS Regression Results
==============================================================================
Dep. Variable:           Total_Deaths   R-squared:                       0.134
Model:                          OLS   Adj. R-squared:                  0.134
Method:               Least Squares   F-statistic:                     495.6
Date:              Wed, 10 Apr 2024   Prob (F-statistic):          1.75e-299
Time:                      14:06:02   Log-Likelihood:                -84347.
No. Observations:              9615   AIC:                         1.687e+05
Df Residuals:                  9611   BIC:                         1.687e+05
Df Model:                         3
Covariance Type:          nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                         1488.6473     33.943     43.857      0.000    1422.112    1555.182
Cerebrovascular diseases         9.9878      0.491     20.345      0.000       9.025      10.950
Alzheimer disease               -2.7276      0.591     -4.616      0.000      -3.886      -1.569
Chronic lower respiratory diseases  8.1970      0.533     15.380      0.000       7.152       9.242
==============================================================================
Omnibus:                    21.256   Durbin-Watson:                   0.080
Prob(Omnibus):               0.000   Jarque-Bera (JB):               21.376
Skew:                        0.113   Prob(JB):                     2.28e-05
Kurtosis:                    2.952   Cond. No.                         247.
==============================================================================
```

R_2 score was very it is .134 or 13% to predicting the values .

```
In [483]: #plot the results

          import matplotlib.pyplot as plt
          plt.figure(figsize=(20,10))
          plt.scatter(y_test,y_pred)
          plt.xlabel('Actual')
          plt.xlabel('predicted')
          plt.title('Actual vs. predicted')

Out[483]: Text(0.5, 1.0, 'Actual vs. predicted')
```



I can conclude that even after improving the model by removing the outliers there very less in the pre dictions and we can see the plot there are still more outliers as compare with old model . The model w ith r_2 score (0.9837079490658794) is the best model.

Data Visualizations:

For the Visualization, I am going to use the R because it have every unique feature of shiny dashboard. where we see the all visualization in shiny dashboard in different various way.

Source Visual

```r
# Load necessary libraries
library(caret)
library(class)
# Load the dataset
df <- read.csv("/Users/balajipamidi/Desktop/Final Project /data.csv")
# Explore dataset
str(df)
```

```
'data.frame':   10476 obs. of  23 variables:
 $ Data.As.Of                        : chr  "27/09/2023" "27/09/2023" "27/09/2023" "27/09/2023" ...
 $ Jurisdiction.of.Occurrence        : chr  "United States" "United States" "United States" "United States" ...
 $ MMWR.Year                         : int  2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 ...
 $ MMWR.Week                         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Week.Ending.Date                  : chr  "2020-01-04" "2020-01-11" "2020-01-18" "2020-01-25" ...
 $ All.Cause                         : int  60179 60735 59363 59162 58834 59491 58815 58912 59334 59716 ...
 $ Natural.Cause                     : int  55010 55755 54516 54401 54001 54416 53969 53987 54316 54401 ...
 $ Malignant.neoplasms               : int  11567 11963 11701 11879 11963 11704 11807 11780 11790 11712 ...
 $ Alzheimer.disease                 : int  2537 2566 2491 2517 2480 2515 2537 2515 2519 2511 ...
 $ Chronic.lower.respiratory.diseases: int  3501 3708 3526 3403 3313 3413 3477 3454 3460 3471 ...
 $ Diseases.of.heart                 : int  14204 13911 13593 13612 13465 14005 13641 13632 13714 13693 ...
 $ Cerebrovascular.diseases          : int  3110 3189 3256 3185 3084 3056 3089 3083 3126 3097 ...
 $ COVID.19_Multiple.Cause.of.Death  : int  0 1 2 3 0 4 6 6 9 38 ...
 $ Year                              : int  2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 ...
 $ Week                              : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Days_Since_Last_Week              : num  0 7 7 7 7 7 7 7 7 7 ...
 $ Total_Deaths                      : int  150108 151828 148448 148162 147140 148604 147341 147369 148268 148639 ...
 $ Malignancy_Mortality_Rate         : num  7.71 7.88 7.88 8.02 8.13 ...
 $ Heart_Disease_Mortality_Rate      : num  9.46 9.16 9.16 9.19 9.15 ...
 $ Natural_Cause_Prop                : num  0.366 0.367 0.367 0.367 0.367 ...
 $ Month                             : int  1 1 1 1 2 2 2 2 2 3 ...
 $ Season                            : int  1 1 1 1 1 1 1 1 1 2 ...
 $ Seasonal.Pattern                  : chr  "Winter" "Winter" "Winter" "Winter" ...
```

```r
dim(df)
```

```
[1] 10476    23
```

```r
head(df)
```

Description: df [6 × 23]

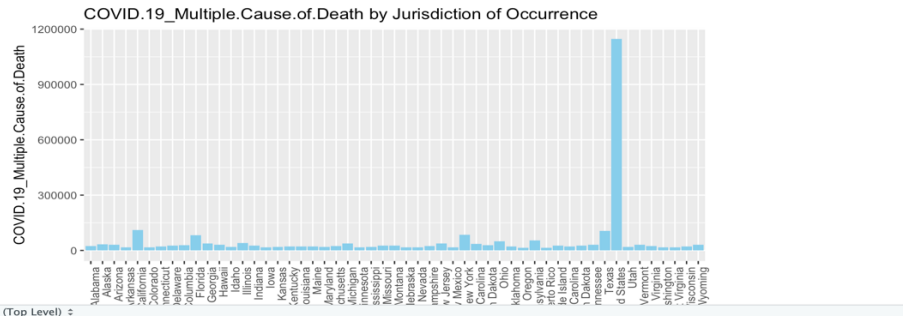
Total_Deaths <int>	Malignancy_Mortality_Rate <dbl>	Heart_Disease_Mortality_Rate <dbl>	Natural_Cause_Prop <dbl>	Month <int>	Season <int>	Seasonal.Pattern <chr>
150108	7.705785	9.462520	0.3664695	1	1	Winter
151828	7.879311	9.162342	0.3672248	1	1	Winter
148448	7.882221	9.156742	0.3672397	1	1	Winter
148162	8.017575	9.187241	0.3671724	1	1	Winter
147140	8.130352	9.151149	0.3670042	2	1	Winter
148604	7.875966	9.424376	0.3661813	2	1	Winter

6 rows | 18-24 of 23 columns

we have call the dataset and read the dataset .

Bar plot (COVID.19_Multiple.Cause.of.Death vs Jurisdiction.of.Occurrence)
Where United states of America has more cases followed by California and then Texas.

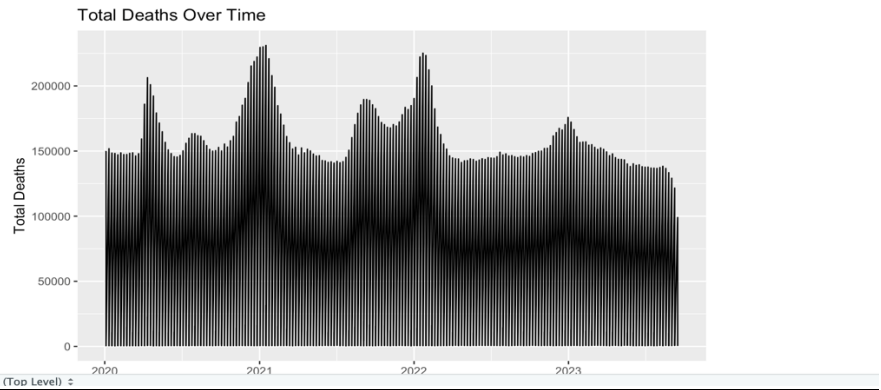
```r
library(ggplot2)

# Summarize total deaths by jurisdiction
total_deaths_by_jurisdiction <- df %>%
  group_by(Jurisdiction.of.Occurrence) %>%
  summarise(COVID.19_Multiple.Cause.of.Death = sum(COVID.19_Multiple.Cause.of.Death))

# Create bar chart
ggplot(total_deaths_by_jurisdiction, aes(x = Jurisdiction.of.Occurrence, y = COVID.19_Multiple.Cause.of.Death)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "COVID.19_Multiple.Cause.of.Death by Jurisdiction of Occurrence",
       x = "Jurisdiction of Occurrence",
       y = "COVID.19_Multiple.Cause.of.Death") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Time Series Plot of Total Deaths each year from 2020 to 2023.

In the Year 2021 to 2022 there was high rise of deaths as compared with other two years.

```r
## Time Series Plot of Total Deaths

library(ggplot2)

# Convert Date columns to Date format
df$Week.Ending.Date <- as.Date(df$Week.Ending.Date, format = "%Y-%m-%d")

# Time series plot of Total Deaths
ggplot(df, aes(x = Week.Ending.Date, y = Total_Deaths)) +
  geom_line() +
  labs(title = "Total Deaths Over Time",
       x = "Week Ending Date",
       y = "Total Deaths")
```

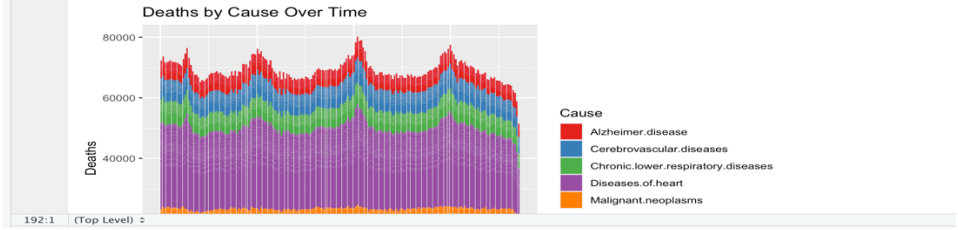


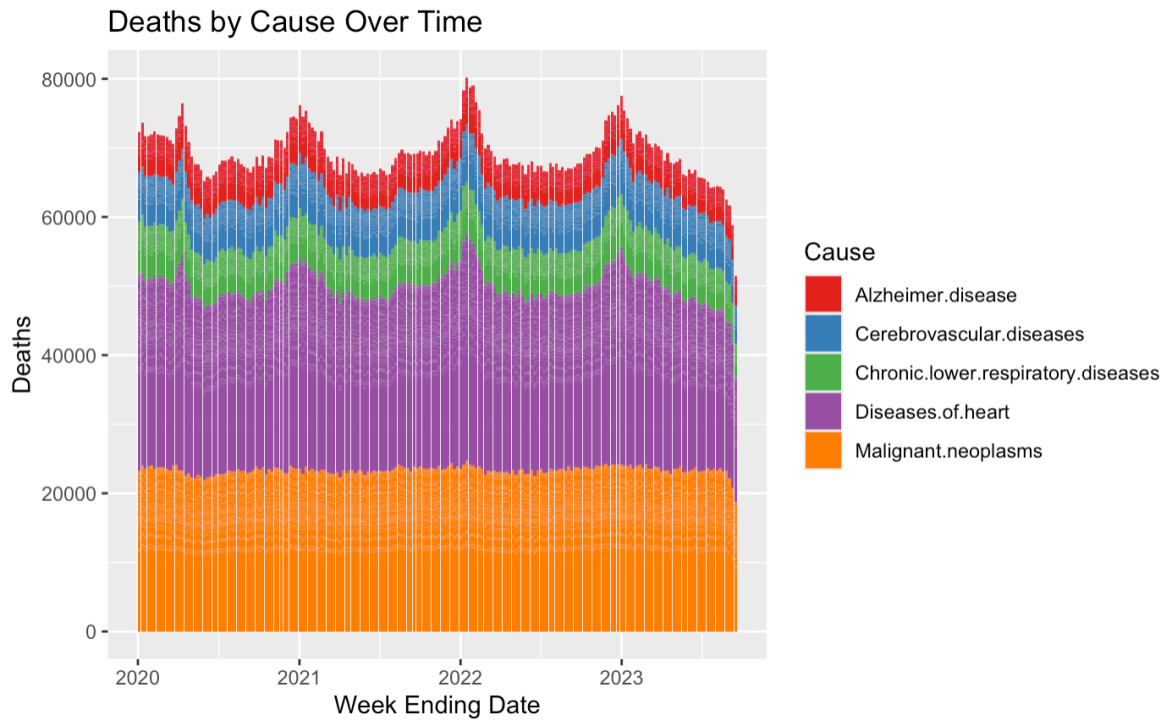
Bar Plot for Deaths by Cause over the time (each year)

- Alzheimer.disease(Highest)
- Cerebrovascular.diseases
- Chronic.lower.respiratory.diseases
- Diseases.of.heart
- Malignant.neoplasms(lowest)

from graph we can say that Alzheimer.disease related deaths count are high in the each year from 2020 to 2023 as compared with other four deiseases.

```r
## Bar Plot of Deaths by Cause

# Summarize deaths by cause
library(tidyr)
library(ggplot2)

df_cause <- df %>%
  select(Week.Ending.Date, Malignant.neoplasms, Alzheimer.disease, Chronic.lower.respiratory.diseases, Diseases.of.heart, Cerebrovascular.diseases) %>%
  pivot_longer(cols = -Week.Ending.Date, names_to = "Cause", values_to = "Deaths")

# Bar plot of deaths by cause
ggplot(df_cause, aes(x = Week.Ending.Date, y = Deaths, fill = Cause)) +
  geom_col(position = "stack") +
  labs(title = "Deaths by Cause Over Time",
       x = "Week Ending Date",
       y = "Deaths",
       fill = "Cause") +
  scale_fill_brewer(palette = "Set1")
```

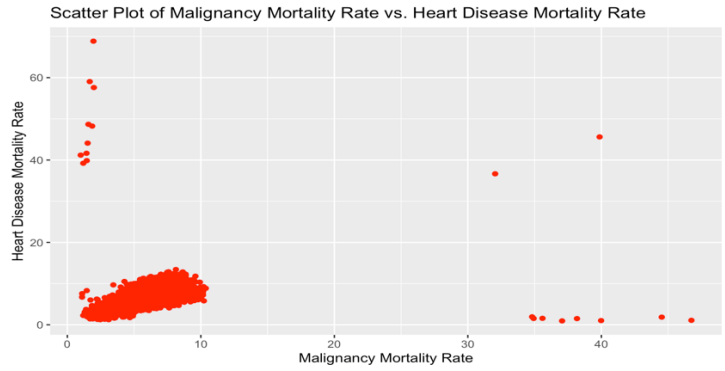




Scatter Plot of Malignancy Mortality Rate vs. Heart Disease Mortality Rate

This plot tells us the correlation between each other. we can see there linear between each other and having outliers.

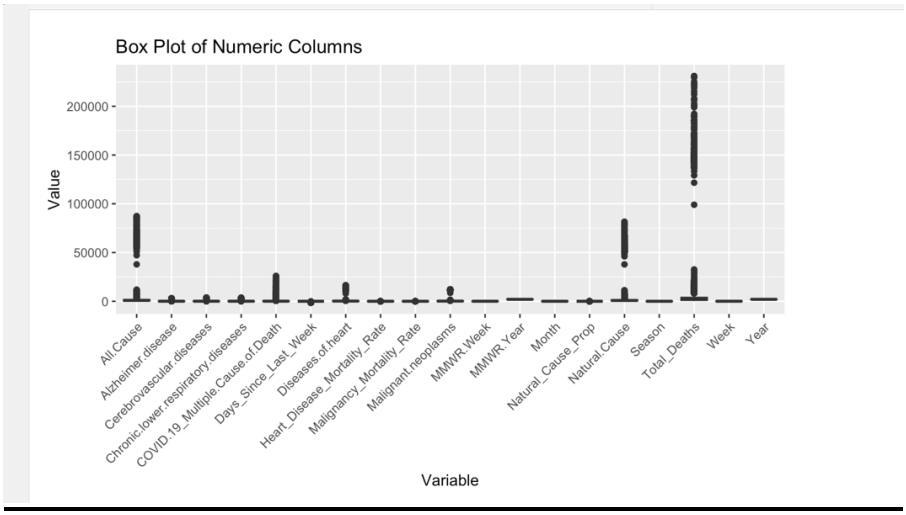
```r
library(ggplot2)

# Create scatter plot
ggplot(df, aes(x = Malignancy_Mortality_Rate, y = Heart_Disease_Mortality_Rate)) +
  geom_point(color = "red") +
  labs(title = "Scatter Plot of Malignancy Mortality Rate vs. Heart Disease Mortality Rate",
       x = "Malignancy Mortality Rate",
       y = "Heart Disease Mortality Rate")
```



Box plot for finding the outliers

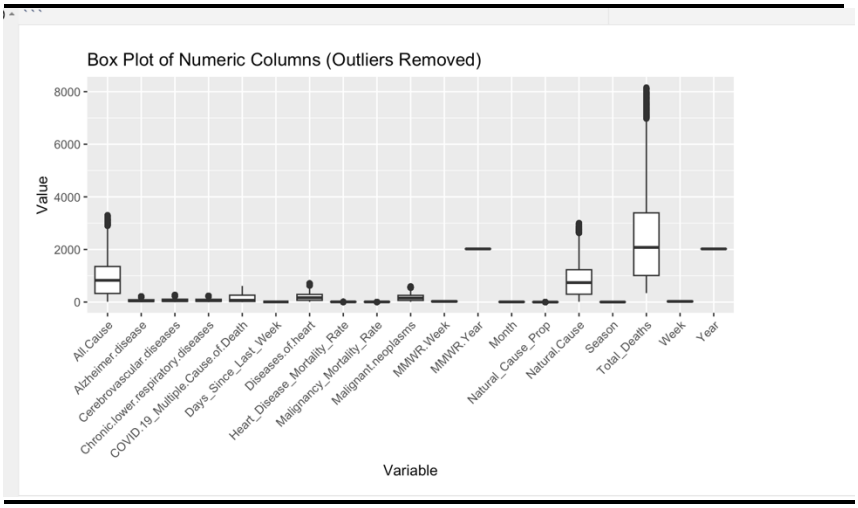
```r
library(ggplot2)
library(dplyr)
library(tidyr)

# Select only numeric columns
numeric_cols <- df %>%
  select_if(is.numeric)

# Reshape the data for box plot
df_long <- numeric_cols %>%
  pivot_longer(everything(), names_to = "Variable", values_to = "Value")

# Create box plot
ggplot(df_long, aes(x = Variable, y = Value)) +
  geom_boxplot() +
  labs(title = "Box Plot of Numeric Columns",
       x = "Variable",
       y = "Value") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

In this we Total_deaths top the having more number of outliers and also with natural and all cause in the next with having more outliers.



After removing the outliers

```r
library(ggplot2)
library(dplyr)

# Define a function to remove outliers based on IQR method
remove_outliers <- function(x) {
  qnt <- quantile(x, probs=c(.25, .75), na.rm = TRUE)
  H <- 1.5 * IQR(x, na.rm = TRUE)
  x[x < (qnt[1] - H)] <- NA
  x[x > (qnt[2] + H)] <- NA
  x
}

# Select only numeric columns
numeric_cols <- df %>%
  select_if(is.numeric)

# Remove outliers from numeric columns
df_clean <- numeric_cols %>%
  mutate(across(everything(), remove_outliers))

# Reshape the data for box plot
df_long <- pivot_longer(df_clean, everything(), names_to = "Variable", values_to = "Value", values_drop_na = TRUE)

# Create box plot without outliers
ggplot(df_long, aes(x = Variable, y = Value)) +
  geom_boxplot() +
  labs(title = "Box Plot of Numeric Columns (Outliers Removed)",
       x = "Variable",
       y = "Value") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Using the Shiny dashboard, I have merged the all graph plot into single page.

```r
library(shiny)
library(shinydashboard)
library(ggplot2)
library(dplyr)
library(tidyr)


# Define UI
ui <- dashboardPage(
  dashboardHeader(title = "Weekly Provisional Counts of Deaths"),
  dashboardSidebar(
    sidebarMenu(
      menuItem("Total Deaths", tabName = "total_deaths"),
      menuItem("Total Deaths Over Time", tabName = "total_deaths_over_time"),
      menuItem("Deaths by Cause Over Time", tabName = "deaths_by_cause"),
      menuItem("Scatter Plot", tabName = "scatter_plot"),
      menuItem("Box Plot", tabName = "box_plot")
    )
  ),
  dashboardBody(
    tabItems(
      tabItem(tabName = "total_deaths",
              fluidRow(
                box(title = "Total Deaths by Jurisdiction", status = "primary", solidHeader = TRUE,
                    plotOutput("total_deaths_plot"))
              )
      ),
      tabItem(tabName = "total_deaths_over_time",
              fluidRow(
                box(title = "Total Deaths Over Time", status = "primary", solidHeader = TRUE,
                    plotOutput("total_deaths_over_time_plot"))
              )
      ),
      tabItem(tabName = "deaths_by_cause",
              fluidRow(
                box(title = "Deaths by Cause Over Time", status = "primary", solidHeader = TRUE,
                    plotOutput("deaths_by_cause_plot"))
              )
```

Output : Shiny dashboard

Conclusion:

The comprehensive data analysis and visualization conducted revealed a significant impact of total deaths, encompassing all causes including diseases and other factors, on multiple linear regression models. This underscores the critical importance of addressing various causes of mortality to effectively control diseases and prevent adverse health outcomes. By utilizing this insight, targeted awareness campaigns can be implemented to educate the public about preventive measures and promote healthier lifestyles, thereby mitigating the risk factors associated with different causes of death and reducing overall mortality rates. Furthermore, the insights derived from this analysis serve as valuable predictive tools for future planning and intervention strategies, enabling proactive measures to anticipate and prevent potential health challenges. In summary, leveraging these findings empowers stakeholders to develop proactive interventions aimed at preventing causes of death and controlling diseases, ultimately leading to the creation of healthier communities and improved public health outcomes.