In [19]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
data = pd.read_csv('Monthly_Counts_of_Deaths_by_Select_Causes__2014-2019.csv')

# Display the first few rows of the dataset
print(data.head())
```

```
  Jurisdiction of Occurrence  Year  Month  All Cause  Natural Cause  \
0              United States  2014      1     243298         226621
1              United States  2015      1     265355         247269
2              United States  2016      1     245823         227341
3              United States  2017      1     262832         241918
4              United States  2018      1     286744         265418

   Septicemia  Malignant Neoplasms  Diabetes Mellitus  Alzheimer Disease  \
0        3944                51101               7344               8305
1        4194                52346               8053              11638
2        3846                51863               7392              10612
3        4089                52120               7907              12018
4        4502                52876               8674              13410

   Influenza and Pneumonia  ...  Other Diseases of Respiratory System  \
0                     7929  ...                                  3466
1                    10005  ...                                  3797
2                     5295  ...                                  3705
3                     6925  ...                                  4083
4                    12164  ...                                  4603

   Nephritis, Nephrotic Syndrome, and Nephrosis  \
0                                          4600
1                                          4979
2                                          4645
3                                          4818
4                                          5346

   Symptoms, Signs, and Abnormal Clinical and Laboratory Findings, Not Elsewhere Classified  \
0                                                2815
1                                                3005
2                                                2755
3                                                2769
4                                                3138

   Diseases of Heart  Cerebrovascular Diseases  \
0              58229                     12074
1              63190                     13576
2              58049                     12968
3              61650                     13595
4              67024                     14653

   Accidents (Unintentional Injuries)  Motor Vehicle Accidents  \
0                               11461                     2572
1                               12311                     2754
2                               12559                     2734
3                               14520                     3034
4                               14748                     3010

   Intentional Self-Harm (Suicide)  Assault (Homicide)  Drug Overdose
0                             3320                1213           4026
1                             3618                1437           4354
2                             3720                1499           4631
3                             3709                1726           6233
4                             3966                1674           5659

[5 rows x 21 columns]
```

In [20]:
```
# Display the data types and non-null counts to check for any immediate cleaning needs
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72 entries, 0 to 71
Data columns (total 21 columns):
 #   Column
Non-Null Count  Dtype
---  ------
--------------  -----
 0   Jurisdiction of Occurrence
72 non-null     object
 1   Year
72 non-null     int64
 2   Month
72 non-null     int64
 3   All Cause
72 non-null     int64
 4   Natural Cause
72 non-null     int64
 5   Septicemia
72 non-null     int64
 6   Malignant Neoplasms
72 non-null     int64
 7   Diabetes Mellitus
72 non-null     int64
 8   Alzheimer Disease
72 non-null     int64
 9   Influenza and Pneumonia
72 non-null     int64
 10  Chronic Lower Respiratory Diseases
72 non-null     int64
 11  Other Diseases of Respiratory System
72 non-null     int64
 12  Nephritis, Nephrotic Syndrome, and Nephrosis
72 non-null     int64
 13  Symptoms, Signs, and Abnormal Clinical and Laboratory Findings, Not Elsewhere Classified
72 non-null     int64
 14  Diseases of Heart
72 non-null     int64
 15  Cerebrovascular Diseases
72 non-null     int64
 16  Accidents (Unintentional Injuries)
72 non-null     int64
 17  Motor Vehicle Accidents
72 non-null     int64
 18  Intentional Self-Harm (Suicide)
72 non-null     int64
 19  Assault (Homicide)
72 non-null     int64
 20  Drug Overdose
72 non-null     int64
dtypes: int64(20), object(1)
memory usage: 11.9+ KB
None
```

In [21]:
```
# Check for missing values in the dataset
print(data.isnull().sum())
```

```
Jurisdiction of Occurrence                                                    0
Year                                                                          0
Month                                                                         0
All Cause                                                                     0
Natural Cause                                                                 0
Septicemia                                                                    0
Malignant Neoplasms                                                           0
Diabetes Mellitus                                                             0
Alzheimer Disease                                                             0
Influenza and Pneumonia                                                       0
Chronic Lower Respiratory Diseases                                            0
Other Diseases of Respiratory System                                          0
Nephritis, Nephrotic Syndrome, and Nephrosis                                  0
Symptoms, Signs, and Abnormal Clinical and Laboratory Findings, Not Elsewhere Classified    0
Diseases of Heart                                                             0
Cerebrovascular Diseases                                                      0
Accidents (Unintentional Injuries)                                            0
Motor Vehicle Accidents                                                       0
Intentional Self-Harm (Suicide)                                               0
Assault (Homicide)                                                            0
Drug Overdose                                                                 0
dtype: int64
```

In [22]:
```
#summary of the numerical columns
print(data.describe())
```

	Year	Month	All Cause	Natural Cause	Septicemia
count	72.00000	72.000000	72.000000	72.000000	72.000000
mean	2016.50000	6.500000	230428.361111	211008.333333	3338.847222
std	1.71981	3.476278	15602.341422	15118.931548	328.925466
min	2014.00000	1.000000	204687.000000	187644.000000	2886.000000
25%	2015.00000	3.750000	219880.250000	199788.500000	3120.250000
50%	2016.50000	6.500000	227895.000000	208178.500000	3248.000000
75%	2018.00000	9.250000	237163.750000	216745.250000	3527.500000
max	2019.00000	12.000000	286744.000000	265418.000000	4502.000000

	Malignant Neoplasms	Diabetes Mellitus	Alzheimer Disease
count	72.000000	72.000000	72.000000
mean	49772.930556	6836.638889	9515.652778
std	1543.032804	611.213600	1267.264717
min	45558.000000	5802.000000	6755.000000
25%	48775.750000	6454.250000	8933.750000
50%	49820.500000	6714.000000	9419.000000
75%	50880.750000	7154.750000	10129.500000
max	52876.000000	8674.000000	13410.000000

	Influenza and Pneumonia	Chronic Lower Respiratory Diseases
count	72.00000	72.000000
mean	4561.12500	12963.944444
std	1692.39442	1666.018201
min	2882.00000	10426.000000
25%	3406.75000	11679.500000
50%	3876.00000	12308.000000
75%	5195.25000	14033.500000
max	12164.00000	18271.000000

	Other Diseases of Respiratory System
count	72.000000
mean	3379.388889
std	388.342181
min	2690.000000
25%	3129.250000
50%	3330.500000
75%	3647.750000
max	4603.000000

	Nephritis, Nephrotic Syndrome, and Nephrosis
count	72.000000
mean	4190.763889
std	344.254103
min	3631.000000
25%	3925.750000
50%	4110.500000
75%	4373.250000
max	5346.000000

	Symptoms, Signs, and Abnormal Clinical and Laboratory Findings, Not Elsewhere Classified
count	72.000000
mean	2713.222222
std	233.464068
min	2449.000000
25%	2551.000000
50%	2633.500000
75%	2812.000000
max	3465.000000

	Diseases of Heart	Cerebrovascular Diseases
count	72.000000	72.000000
mean	53407.347222	11941.194444
std	4143.565866	911.097413
min	46909.000000	9973.000000
25%	50367.000000	11337.500000
50%	52765.000000	11814.000000
75%	54980.500000	12428.000000
max	67024.000000	14653.000000

```
       Accidents (Unintentional Injuries)  Motor Vehicle Accidents  \
count                           72.000000               72.000000
mean                         13249.666667             3225.333333
std                           1255.711414              348.237089
min                          10286.000000             2248.000000
25%                          12306.500000             3029.250000
50%                          13488.000000             3262.000000
75%                          14225.500000             3517.250000
max                          15292.000000             3834.000000

       Intentional Self-Harm (Suicide)  Assault (Homicide)  Drug Overdose
count                        72.000000           72.000000      72.000000
mean                       3819.611111         1534.833333    5157.291667
std                         294.549145          162.753075     793.379408
min                        3091.000000         1050.000000    3733.000000
25%                        3596.750000         1434.750000    4435.000000
50%                        3835.500000         1563.500000    5477.500000
75%                        4042.250000         1658.250000    5820.000000
max                        4378.000000         1804.000000    6299.000000
```

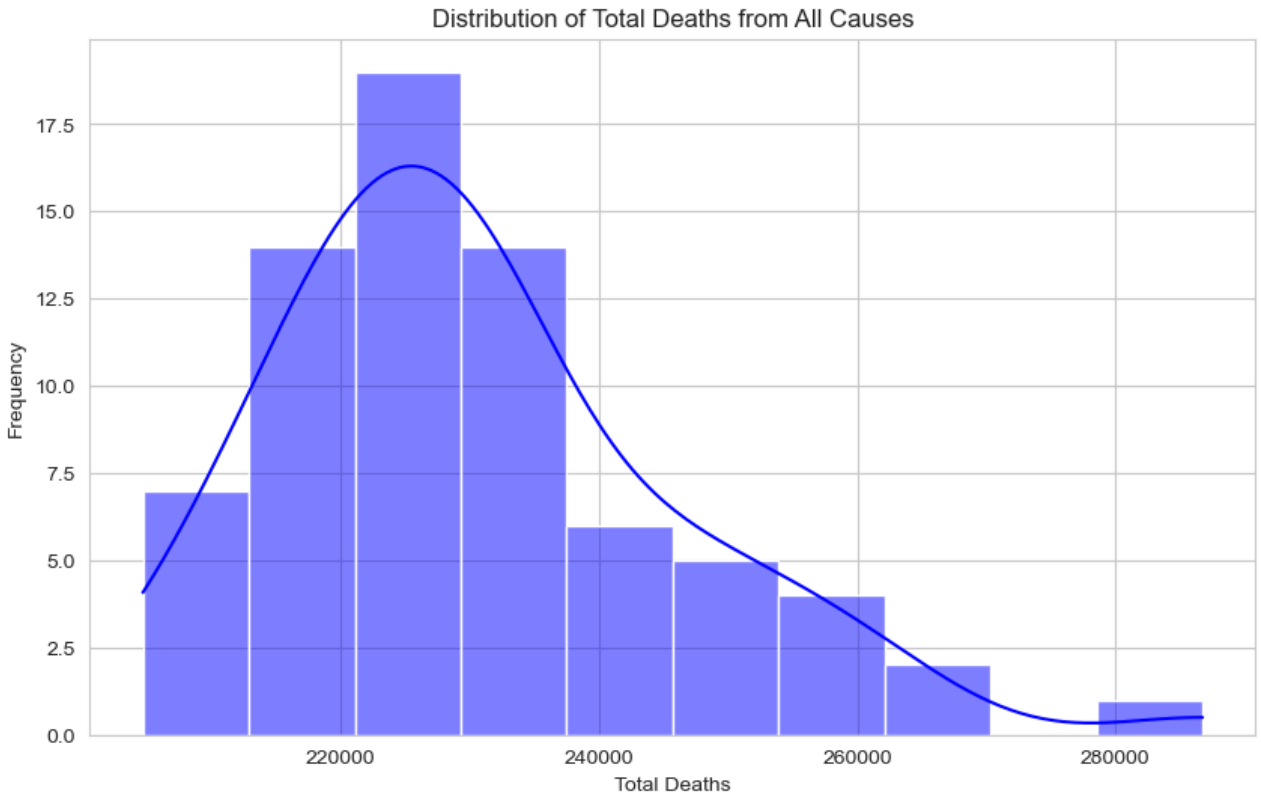
In [23]:
```
#column names to check the actual names in the dataset
print(data.columns)
```

```
Index(['Jurisdiction of Occurrence', 'Year', 'Month', 'All Cause',
       'Natural Cause', 'Septicemia', 'Malignant Neoplasms',
       'Diabetes Mellitus', 'Alzheimer Disease', 'Influenza and Pneumonia',
       'Chronic Lower Respiratory Diseases',
       'Other Diseases of Respiratory System',
       'Nephritis, Nephrotic Syndrome, and Nephrosis',
       'Symptoms, Signs, and Abnormal Clinical and Laboratory Findings, Not Elsewhere Classifie
d',
       'Diseases of Heart', 'Cerebrovascular Diseases',
       'Accidents (Unintentional Injuries)', 'Motor Vehicle Accidents',
       'Intentional Self-Harm (Suicide)', 'Assault (Homicide)',
       'Drug Overdose'],
      dtype='object')
```

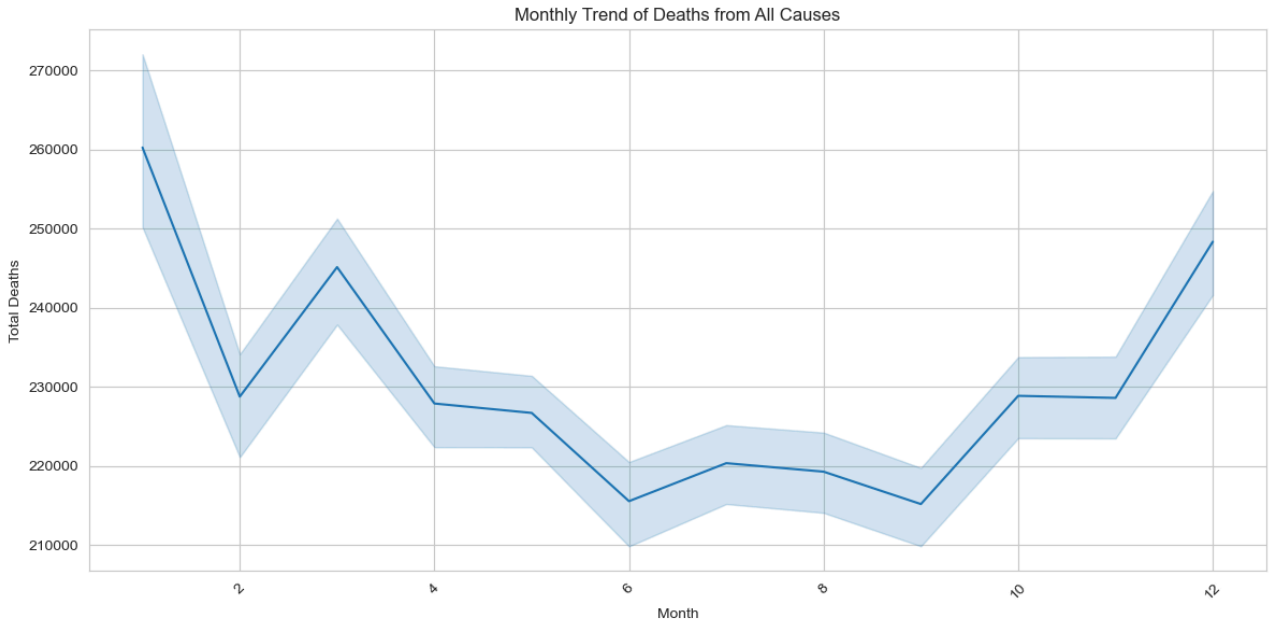
In [24]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [25]:
```
sns.set_style("whitegrid")
```

In [26]:
```
# Plotting the distribution of total deaths for 'All Cause' to see the general trend
plt.figure(figsize=(10, 6))
sns.histplot(data['All Cause'], kde=True, color='blue')
plt.title('Distribution of Total Deaths from All Causes')
plt.xlabel('Total Deaths')
plt.ylabel('Frequency')
plt.show()
```



In [27]:
```
plt.figure(figsize=(12, 6))
sns.lineplot(data=data, x='Month', y='All Cause')
plt.title('Monthly Trend of Deaths from All Causes')
plt.xlabel('Month')
plt.ylabel('Total Deaths')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



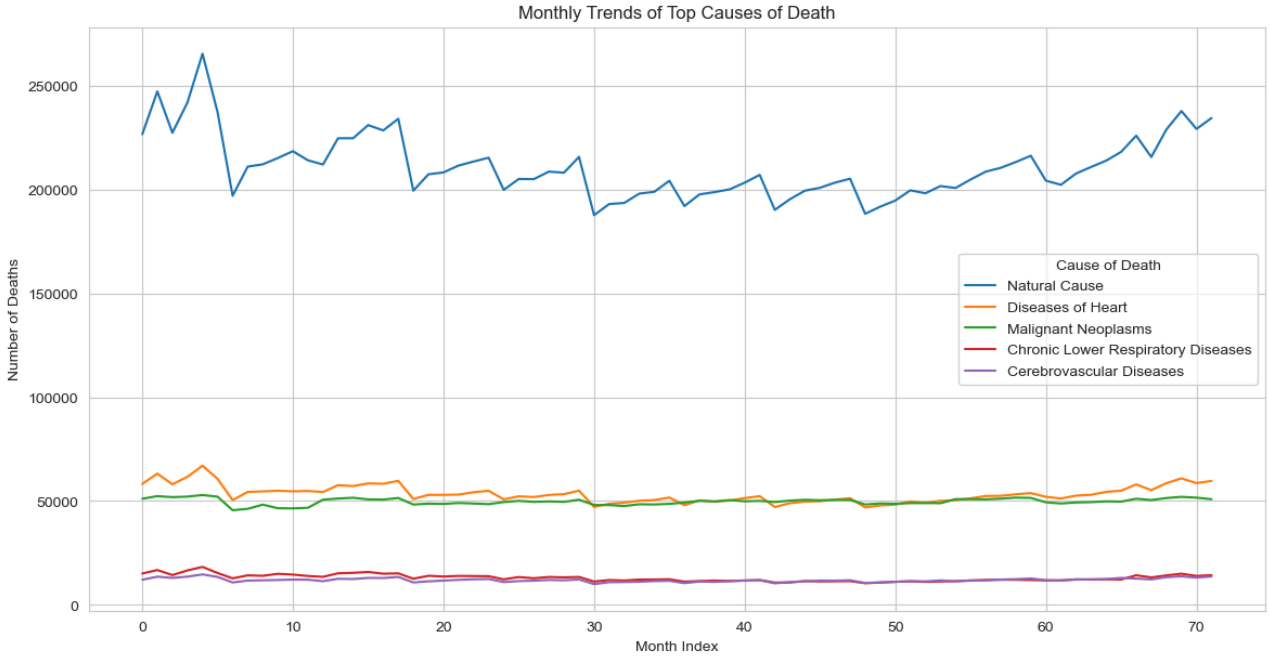
In [29]:
```
#sum of each cause of death column
total_deaths_by_cause = data[[col for col in data.columns if 'Cause' in col]].sum()

#sums to see the total deaths for each cause
print(total_deaths_by_cause)
```

```
All Cause        16590842
Natural Cause    15192600
dtype: int64
```

In [30]:
```
# Sort the total deaths to find the top causes
top_causes = total_deaths_by_cause.sort_values(ascending=False)
print("Top Causes of Death:")
print(top_causes.head(5))
```

```
Top Causes of Death:
All Cause        16590842
Natural Cause    15192600
dtype: int64
```

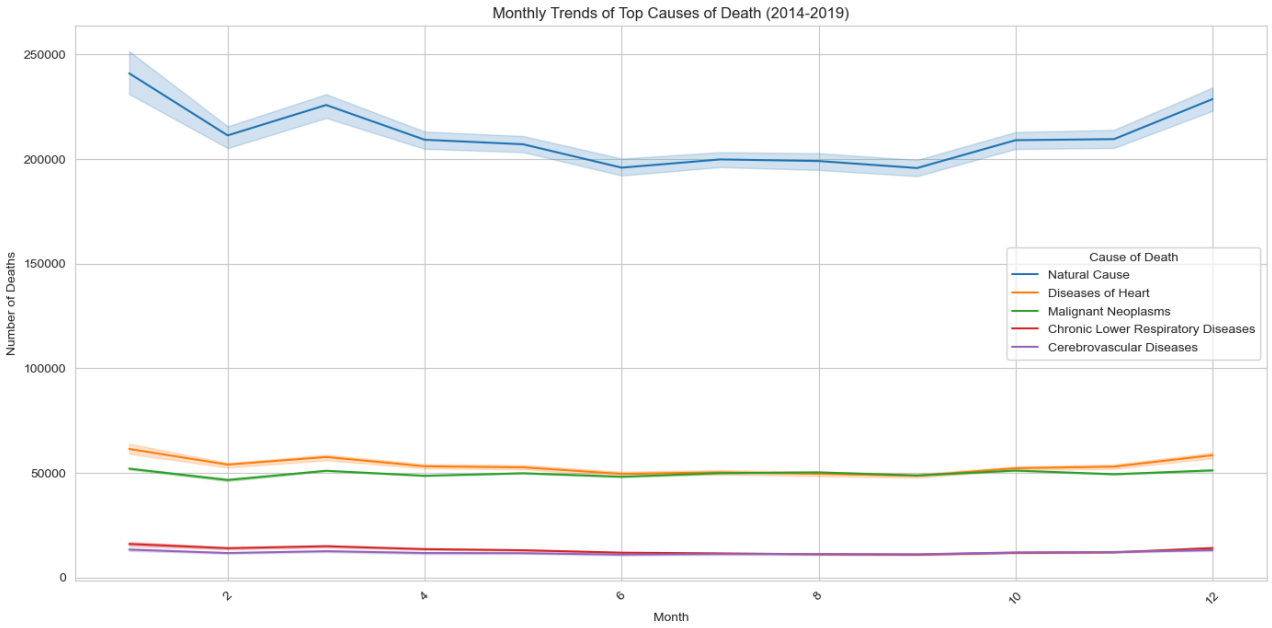
In [32]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Set plot style
sns.set_style("whitegrid")

# Plot trends for the top 5 causes of death
plt.figure(figsize=(14, 7))
for cause in top_causes.head(5).index:
    sns.lineplot(x=data.index, y=data[cause], label=cause)

plt.title('Monthly Trends of Top Causes of Death')
plt.xlabel('Month Index')
plt.ylabel('Number of Deaths')
plt.legend(title='Cause of Death')
plt.show()
```



In [33]:
```
# Sum the death counts for each cause over the entire period to identify the top causes
top_causes = data[['Natural Cause', 'Malignant Neoplasms', 'Diseases of Heart', 'Chronic Lower

print("Top Causes of Death (2014-2019):")
print(top_causes)

# Plotting trends of the top causes of death
plt.figure(figsize=(14, 7))
for cause in top_causes.index:
    sns.lineplot(data=data, x='Month', y=cause, label=cause)

plt.title('Monthly Trends of Top Causes of Death (2014-2019)')
plt.xlabel('Month')
plt.ylabel('Number of Deaths')
plt.legend(title='Cause of Death')
```

```
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

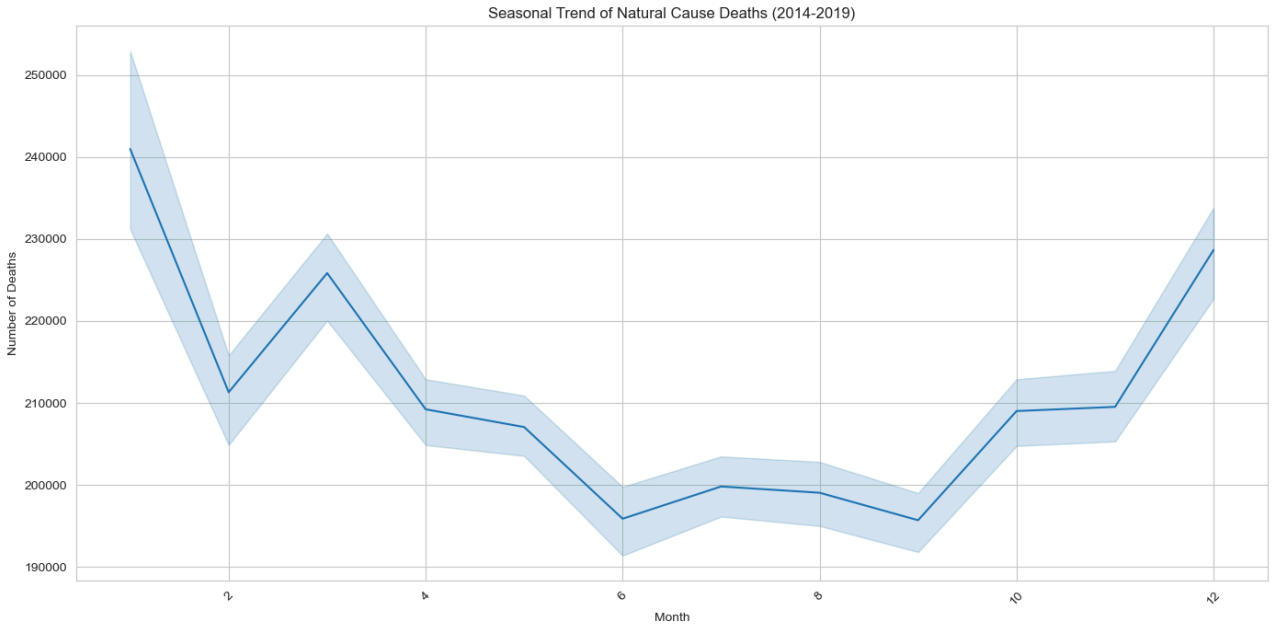
```
Top Causes of Death (2014-2019):
Natural Cause                          15192600
Diseases of Heart                       3845329
Malignant Neoplasms                     3583651
Chronic Lower Respiratory Diseases       933404
Cerebrovascular Diseases                 859766
dtype: int64
```



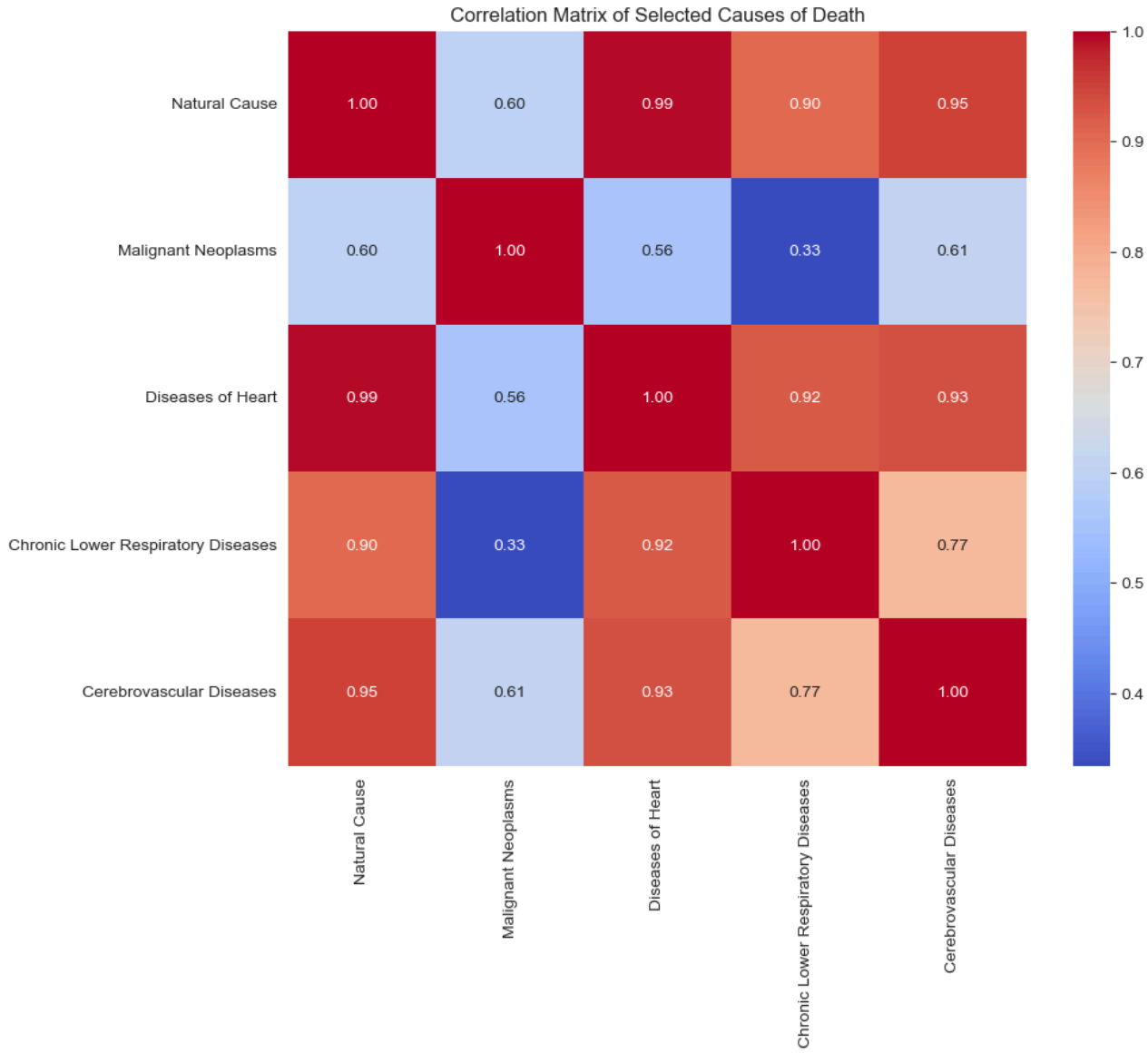
In [34]:
```
# Using 'Diseases of Heart' as an example for seasonal analysis
plt.figure(figsize=(14, 7))
sns.lineplot(data=data, x='Month', y='Diseases of Heart')
plt.title('Seasonal Trend of Heart Disease Deaths (2014-2019)')
plt.xlabel('Month')
plt.ylabel('Number of Deaths')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

In [35]:
```
plt.figure(figsize=(14, 7))
sns.lineplot(data=data, x='Month', y='Natural Cause')
```

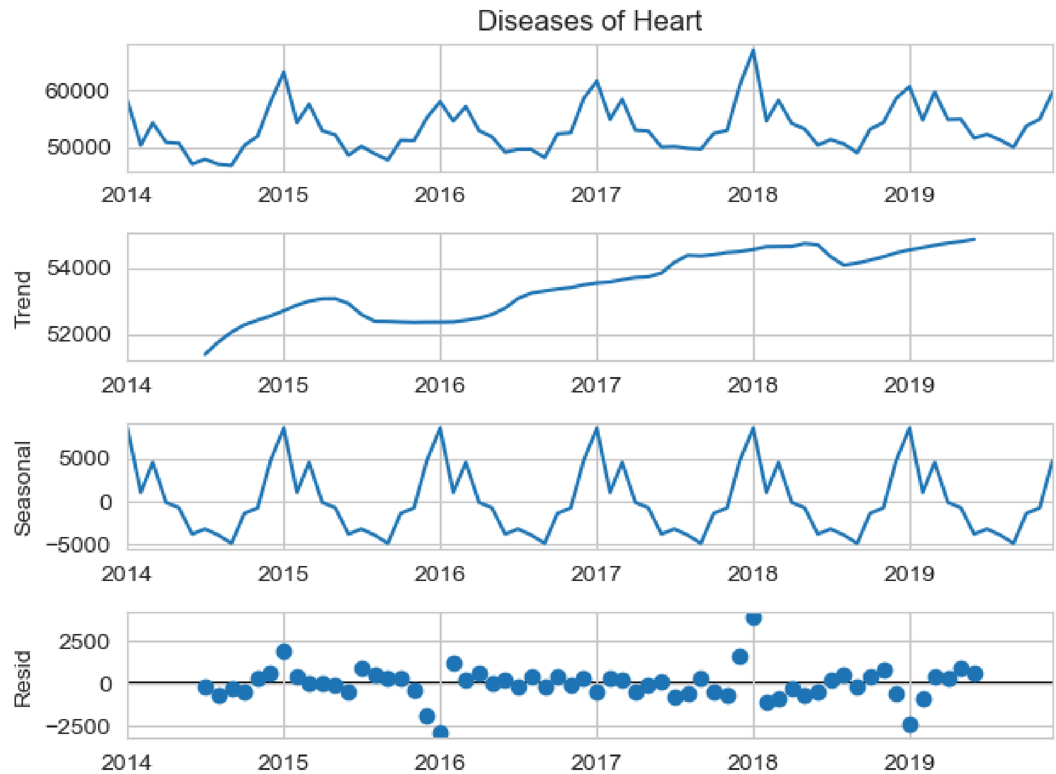
```
plt.title('Seasonal Trend of Natural Cause Deaths (2014-2019)')
plt.xlabel('Month')
plt.ylabel('Number of Deaths')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [36]:
```
# Selecting a subset of causes for correlation analysis
causes_subset = data[['Natural Cause', 'Malignant Neoplasms', 'Diseases of Heart', 'Chronic Low
correlation_matrix = causes_subset.corr()

# Plotting the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of Selected Causes of Death')
plt.show()
```



In [38]:
```
import statsmodels.api as sm

# focusing on a single cause, e.g., 'Diseases of Heart'
data['Date'] = pd.to_datetime(data['Year'].astype(str) + '-' + data['Month'].astype(str))
heart_disease_data = data.sort_values('Date').set_index('Date')['Diseases of Heart']

# Decompose the time series
decomposition = sm.tsa.seasonal_decompose(heart_disease_data, model='additive', period=12)
fig = decomposition.plot()
plt.show()
```



In [39]:
```
from statsmodels.tsa.arima.model import ARIMA

# ARIMA Model for 'Diseases of Heart'
model = ARIMA(heart_disease_data, order=(1,1,1)) # The order (p,d,q) needs to be adjusted based
model_fit = model.fit()

# Forecast the next 12 months
forecast = model_fit.forecast(steps=12)
print(forecast)
```

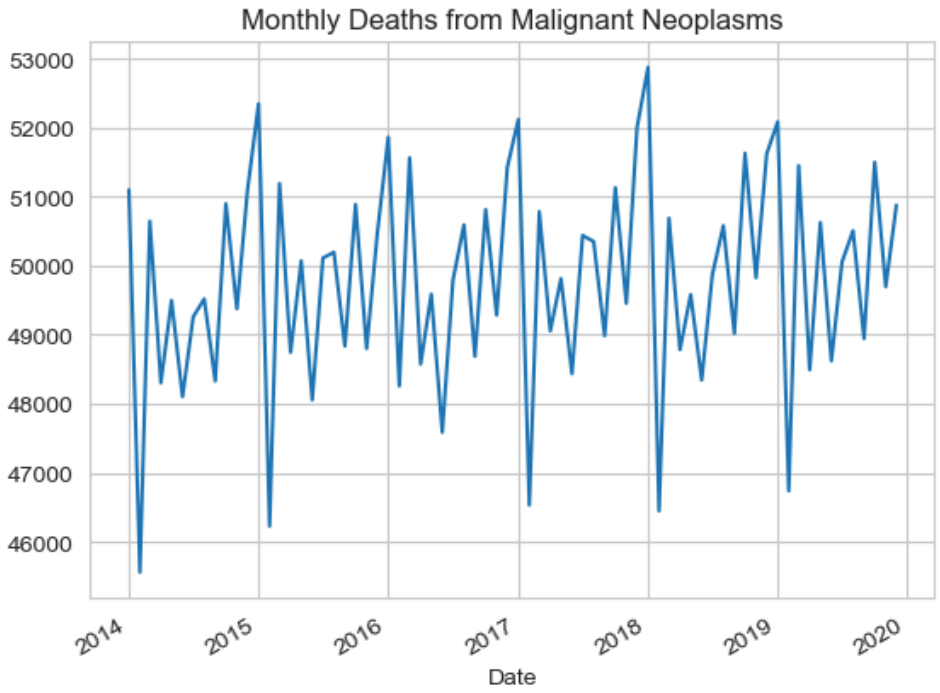
```
2020-01-01    58902.730231
2020-02-01    59276.070064
2020-03-01    59092.736941
2020-04-01    59182.764929
2020-05-01    59138.555577
2020-06-01    59160.265124
2020-07-01    59149.604384
2020-08-01    59154.839471
2020-09-01    59152.268717
2020-10-01    59153.531118
2020-11-01    59152.911200
2020-12-01    59153.215618
Freq: MS, Name: predicted_mean, dtype: float64
```

```
/Users/sheyambitar/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:
471: ValueWarning: No frequency information was provided, so inferred frequency MS will be use
d.
  self._init_dates(dates, freq)
/Users/sheyambitar/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:
471: ValueWarning: No frequency information was provided, so inferred frequency MS will be use
d.
  self._init_dates(dates, freq)
/Users/sheyambitar/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:
471: ValueWarning: No frequency information was provided, so inferred frequency MS will be use
d.
  self._init_dates(dates, freq)
```

In [43]:
```
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima.model import ARIMA

data['Date'] = pd.to_datetime(data['Year'].astype(str) + '-' + data['Month'].astype(str))
```

```
data.set_index('Date', inplace=True)

# Select the time series for "Malignant Neoplasms"
cancer_deaths = data['Malignant Neoplasms']

# Check for stationarity
result = adfuller(cancer_deaths.dropna())  # dropna() is used to remove any missing values
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])

# Plot the data
cancer_deaths.plot(title='Monthly Deaths from Malignant Neoplasms')
plt.show()
```

```
ADF Statistic: -1.178399
p-value: 0.682818
```

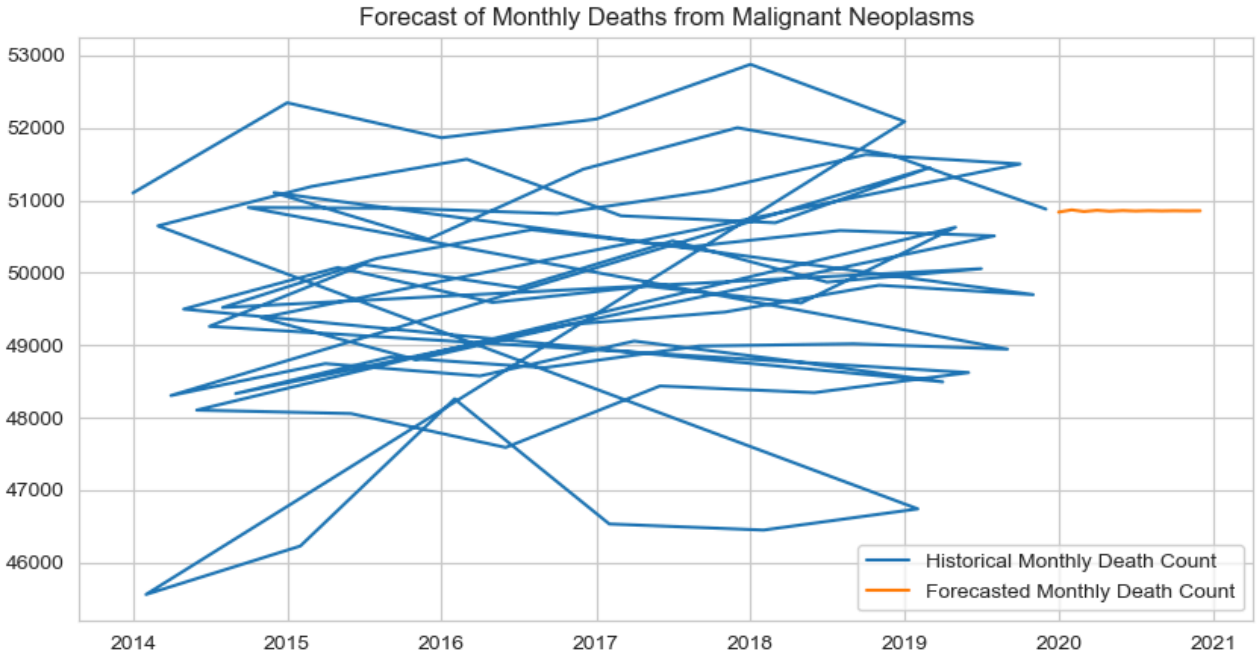


In [44]:

```
# Assuming the data is stationary or after you've differenced it if needed
# Fit an ARIMA model
model = ARIMA(cancer_deaths, order=(1,1,1))  # You may need to adjust the order based on ACF an
model_fit = model.fit()

# Print out the summary of the model's performance
print(model_fit.summary())

# Forecast the next 12 months
forecast = model_fit.forecast(steps=12)
plt.figure(figsize=(10,5))
plt.plot(cancer_deaths.index, cancer_deaths, label='Historical Monthly Death Count')
plt.plot(pd.date_range(cancer_deaths.index[-1], periods=12, freq='M'), forecast, label='Forecas
plt.title('Forecast of Monthly Deaths from Malignant Neoplasms')
plt.legend()
plt.show()
```

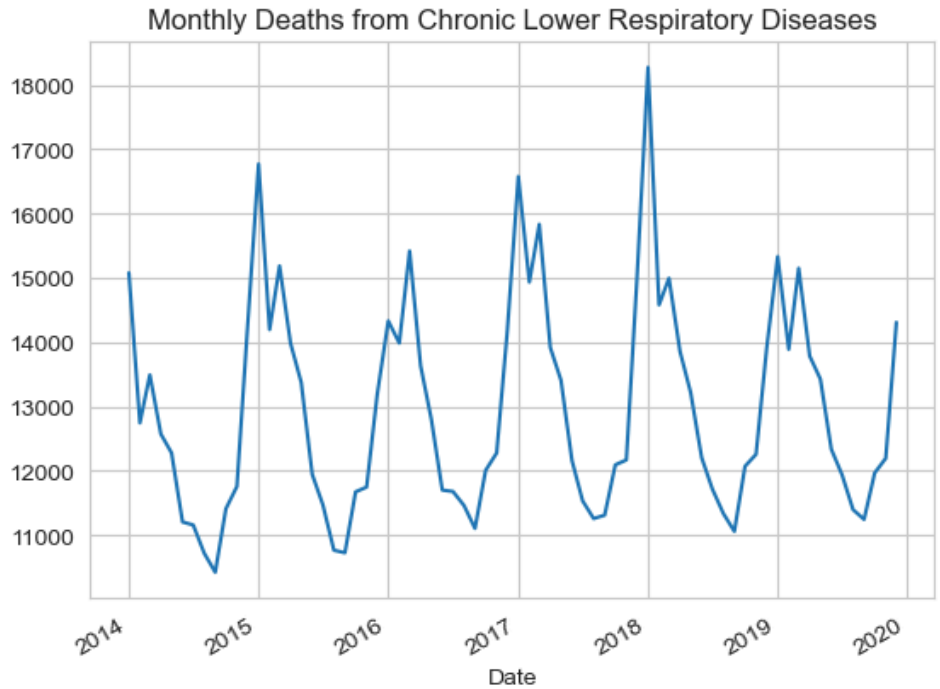
```
/Users/sheyambitar/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:
471: ValueWarning: A date index has been provided, but it has no associated frequency informati
on and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
/Users/sheyambitar/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:
471: ValueWarning: A date index has been provided, but it is not monotonic and so will be ignor
ed when e.g. forecasting.
  self._init_dates(dates, freq)
/Users/sheyambitar/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:
471: ValueWarning: A date index has been provided, but it has no associated frequency informati
on and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
/Users/sheyambitar/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:
471: ValueWarning: A date index has been provided, but it is not monotonic and so will be ignor
ed when e.g. forecasting.
  self._init_dates(dates, freq)
/Users/sheyambitar/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:
471: ValueWarning: A date index has been provided, but it has no associated frequency informati
on and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
/Users/sheyambitar/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:
471: ValueWarning: A date index has been provided, but it is not monotonic and so will be ignor
ed when e.g. forecasting.
  self._init_dates(dates, freq)
/Users/sheyambitar/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:
834: ValueWarning: No supported index is available. Prediction results will be given with an in
teger index beginning at `start`.
  return get_prediction_index(
```

```
                               SARIMAX Results                                
==============================================================================
Dep. Variable:     Malignant Neoplasms   No. Observations:                   72
Model:                  ARIMA(1, 1, 1)   Log Likelihood                -606.866
Date:                 Sun, 12 May 2024   AIC                           1219.732
Time:                         01:10:15   BIC                           1226.520
Sample:                              0   HQIC                          1222.431
                                  - 72                                         
Covariance Type:                   opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.7349      0.474     -1.550      0.121      -1.664       0.194
ma.L1          0.7692      0.396      1.944      0.052      -0.006       1.545
sigma2       1.62e+06   1.99e+05      8.155      0.000    1.23e+06    2.01e+06
===================================================================================
Ljung-Box (L1) (Q):                   0.58   Jarque-Bera (JB):               249.22
Prob(Q):                              0.44   Prob(JB):                         0.00
Heteroskedasticity (H):               0.24   Skew:                            -1.64
Prob(H) (two-sided):                  0.00   Kurtosis:                        11.57
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```



In [45]:
```
respiratory_deaths = data['Chronic Lower Respiratory Diseases']

# Check for stationarity
result = adfuller(respiratory_deaths.dropna())  # Ensure there are no NaN values
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])

# Plot the data
respiratory_deaths.plot(title='Monthly Deaths from Chronic Lower Respiratory Diseases')
plt.show()
```

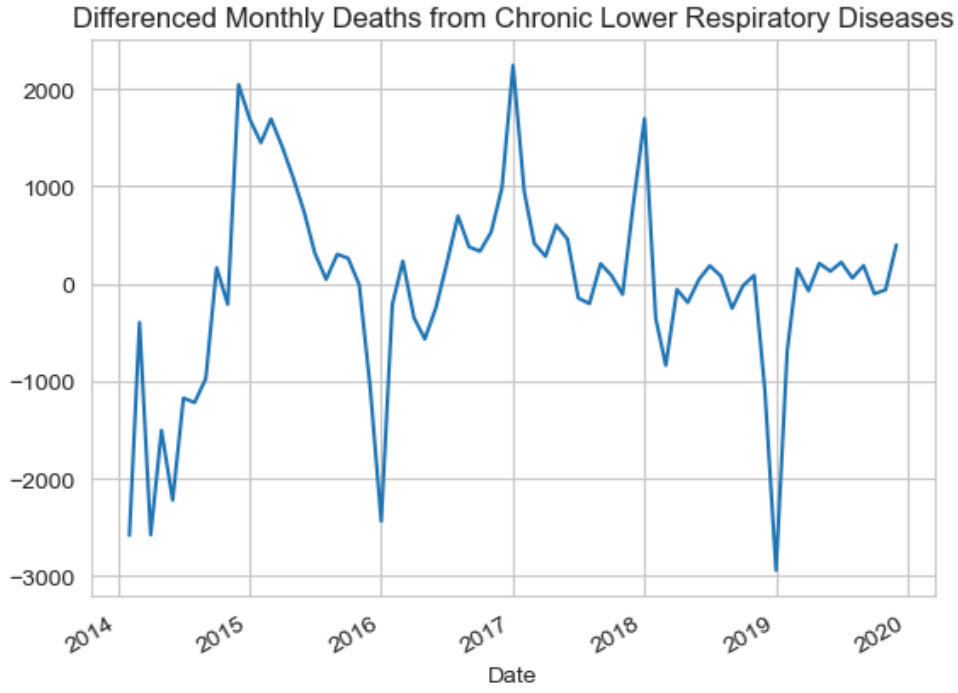
```
ADF Statistic: -1.412512
p-value: 0.576263
```



In [46]:
```
# Difference the data if not stationary
respiratory_diff = respiratory_deaths.diff().dropna()

# Re-check for stationarity
result_diff = adfuller(respiratory_diff)
```

```
print('Differenced data ADF Statistic: %f' % result_diff[0])
print('Differenced data p-value: %f' % result_diff[1])

# Plot differenced data
respiratory_diff.plot(title='Differenced Monthly Deaths from Chronic Lower Respiratory Diseases
plt.show()
```

```
Differenced data ADF Statistic: 0.170373
Differenced data p-value: 0.970550
```

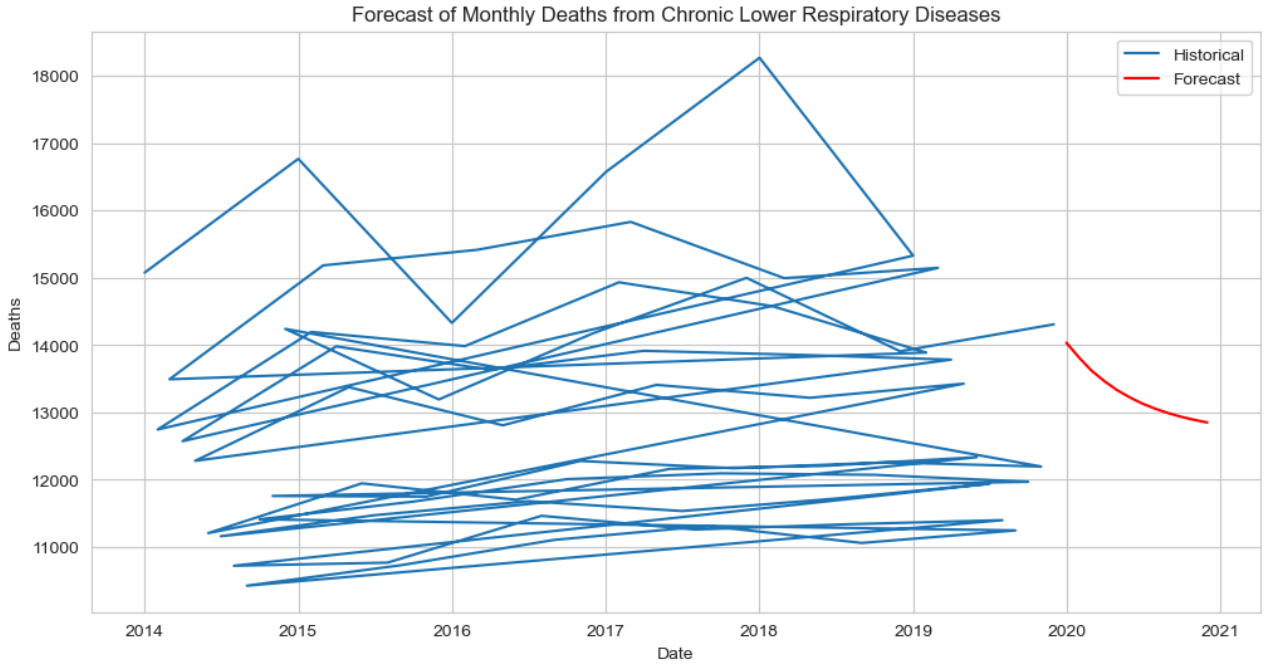


In [47]:

```
# Fit an ARIMA model to the (differenced) stationary data
model = ARIMA(respiratory_deaths, order=(1,1,1))  # Adjust the order based on ACF and PACF plot
model_fit = model.fit()

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

# Forecast the next 12 months
forecast = model_fit.forecast(steps=12)

# Plot the historical data and the forecast
plt.figure(figsize=(12, 6))
plt.plot(respiratory_deaths.index, respiratory_deaths, label='Historical')
plt.plot(pd.date_range(respiratory_deaths.index[-1], periods=12, freq='M'), forecast, label='Fo
plt.title('Forecast of Monthly Deaths from Chronic Lower Respiratory Diseases')
plt.xlabel('Date')
plt.ylabel('Deaths')
plt.legend()
plt.show()
```

```
/Users/sheyambitar/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:
471: ValueWarning: A date index has been provided, but it has no associated frequency informati
on and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
/Users/sheyambitar/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:
471: ValueWarning: A date index has been provided, but it is not monotonic and so will be ignor
ed when e.g. forecasting.
  self._init_dates(dates, freq)
/Users/sheyambitar/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:
471: ValueWarning: A date index has been provided, but it has no associated frequency informati
on and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
/Users/sheyambitar/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:
471: ValueWarning: A date index has been provided, but it is not monotonic and so will be ignor
ed when e.g. forecasting.
  self._init_dates(dates, freq)
/Users/sheyambitar/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:
471: ValueWarning: A date index has been provided, but it has no associated frequency informati
on and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
/Users/sheyambitar/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:
471: ValueWarning: A date index has been provided, but it is not monotonic and so will be ignor
ed when e.g. forecasting.
  self._init_dates(dates, freq)
/Users/sheyambitar/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:
834: ValueWarning: No supported index is available. Prediction results will be given with an in
teger index beginning at `start`.
  return get_prediction_index(
```

```
                                        SARIMAX Results                                        
===============================================================================================
Dep. Variable:     Chronic Lower Respiratory Diseases   No. Observations:                   72
Model:                                 ARIMA(1, 1, 1)   Log Likelihood                -589.531
Date:                                Sun, 12 May 2024   AIC                           1185.062
Time:                                        01:12:39   BIC                           1191.850
Sample:                                             0   HQIC                          1187.762
                                                 - 72                                         
Covariance Type:                                  opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8347      0.053     15.799      0.000       0.731       0.938
ma.L1         -0.9986      0.137     -7.313      0.000      -1.266      -0.731
sigma2      9.069e+05    1.5e-07   6.06e+12      0.000    9.07e+05    9.07e+05
===================================================================================
Ljung-Box (L1) (Q):                   4.07   Jarque-Bera (JB):                 2.92
Prob(Q):                              0.04   Prob(JB):                         0.23
Heteroskedasticity (H):               0.21   Skew:                             0.10
Prob(H) (two-sided):                  0.00   Kurtosis:                         3.97
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```


Forecast of Monthly Deaths from Chronic Lower Respiratory Diseases
Historical
Forecast
Deaths
Date
18000
17000
16000
15000
14000
13000
12000
11000
2014
2015
2016
2017
2018
2019
2020
2021

In [ ]: