In [34]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [35]:
```
import pandas as pd
import zipfile

# Specify the path to your ZIP file
zip_file_path = r"C:\Users\warul\Downloads\archive (1).zip"

# Specify the name of the CSV file inside the ZIP file
Unemployment_Rate = 'Unemployment_Rate_upto_11_2020.csv'

# Extract the CSV file from the ZIP archive
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extract(csv_file_name, 'temp_folder')

# Read the extracted CSV file into a pandas DataFrame
df = pd.read_csv('temp_folder/' + Unemployment_Rate)

# Display the first few rows of the dataset to understand its structure
print(df.head())
```

```
           Region         Date  Frequency   Estimated Unemployment Rate (%)  \
0  Andhra Pradesh   31-01-2020          M                              5.48
1  Andhra Pradesh   29-02-2020          M                              5.83
2  Andhra Pradesh   31-03-2020          M                              5.79
3  Andhra Pradesh   30-04-2020          M                             20.51
4  Andhra Pradesh   31-05-2020          M                             17.43

    Estimated Employed   Estimated Labour Participation Rate (%) Region.1  \
0             16635535                                    41.02    South
1             16545652                                    40.90    South
2             15881197                                    39.18    South
3             11336911                                    33.10    South
4             12988845                                    36.46    South

   longitude  latitude
0    15.9129     79.74
1    15.9129     79.74
2    15.9129     79.74
3    15.9129     79.74
4    15.9129     79.74
```

In [37]:
```
# Remove leading spaces from column names
df.columns = df.columns.str.strip()

# Print all column names
print(df.columns)

# Check if ' Date' is in the column names
if ' Date' in df.columns:
    # Convert the correct date column to datetime format
    df[' Date'] = pd.to_datetime(df[' Date'])

    # Set the correct date column as the index
    df.set_index(' Date', inplace=True)
```

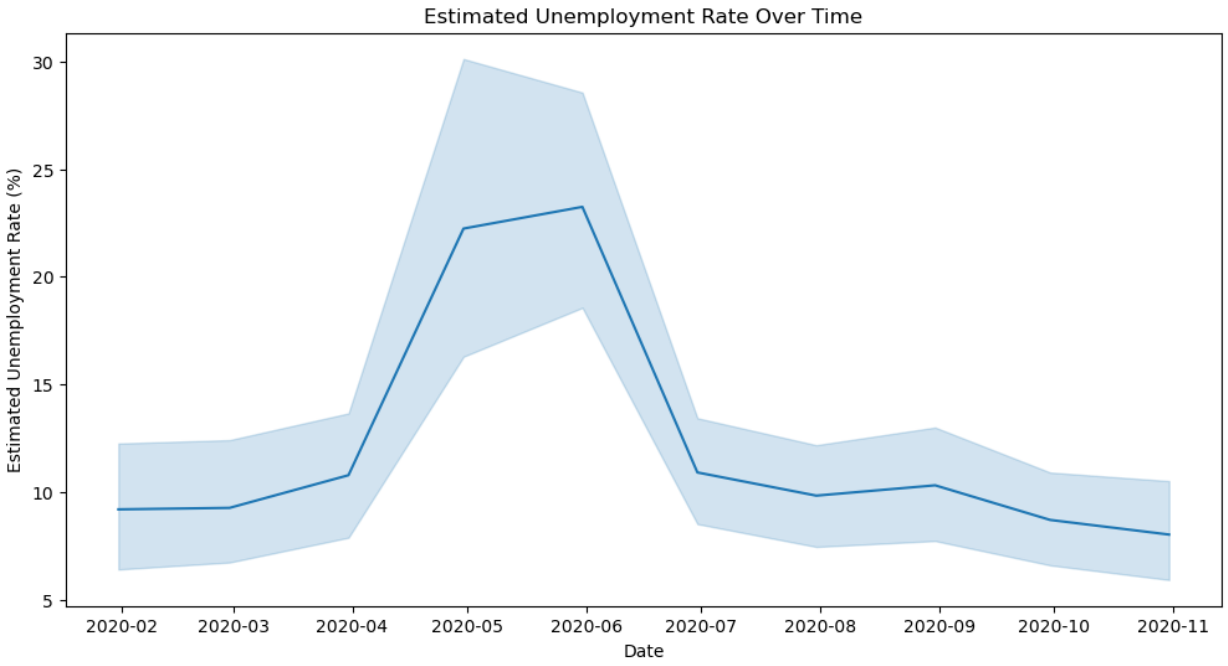
```
else:
    print("Column ' Date' not found.")
```

```
Index(['Region', 'Date', 'Frequency', 'Estimated Unemployment Rate (%)',
       'Estimated Employed', 'Estimated Labour Participation Rate (%)',
       'Region.1', 'longitude', 'latitude'],
      dtype='object')
Column ' Date' not found.
```

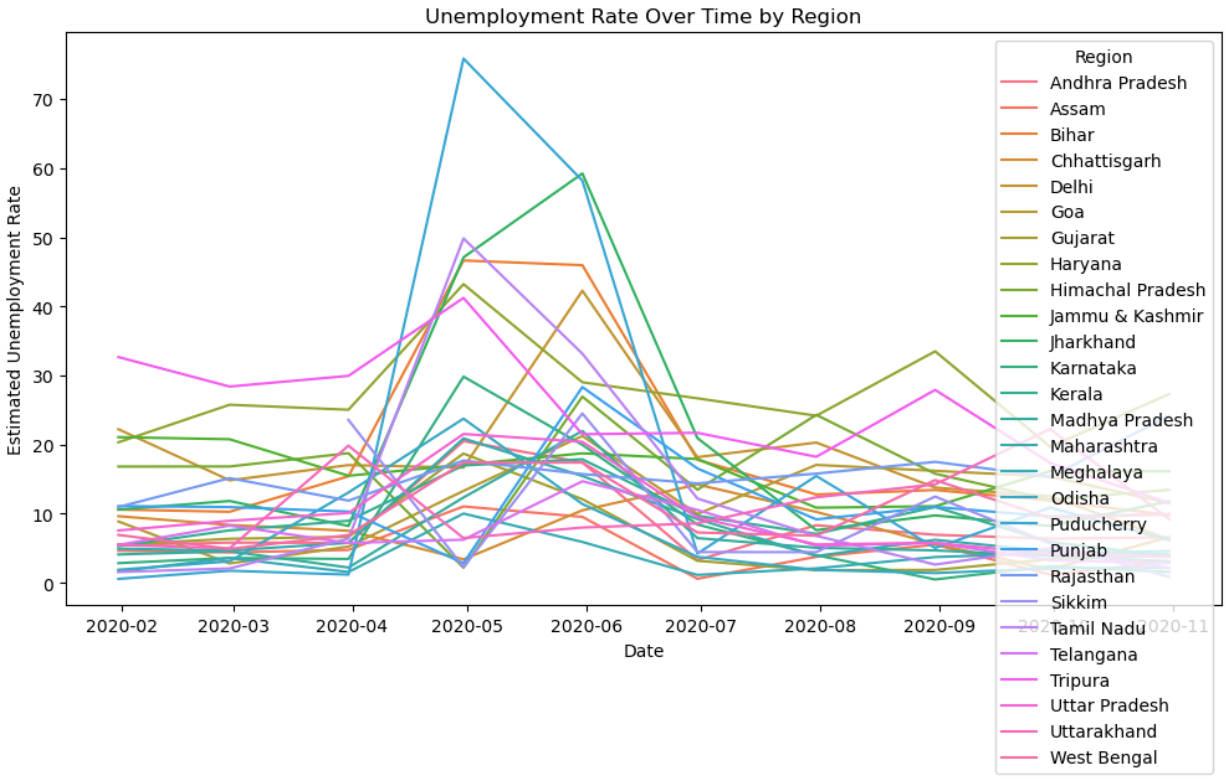
In [38]:
```
print(df.columns)
```

```
Index(['Region', 'Date', 'Frequency', 'Estimated Unemployment Rate (%)',
       'Estimated Employed', 'Estimated Labour Participation Rate (%)',
       'Region.1', 'longitude', 'latitude'],
      dtype='object')
```

In [39]:
```
# Convert the 'Date' column to datetime format
df['Date'] = pd.to_datetime(df['Date'])

# Set the 'Date' column as the index
df.set_index('Date', inplace=True)

# Plot the estimated unemployment rate over time
plt.figure(figsize=(12, 6))
sns.lineplot(data=df, x=df.index, y='Estimated Unemployment Rate (%)')
plt.title('Estimated Unemployment Rate Over Time')
plt.xlabel('Date')
plt.ylabel('Estimated Unemployment Rate (%)')
plt.show()
```

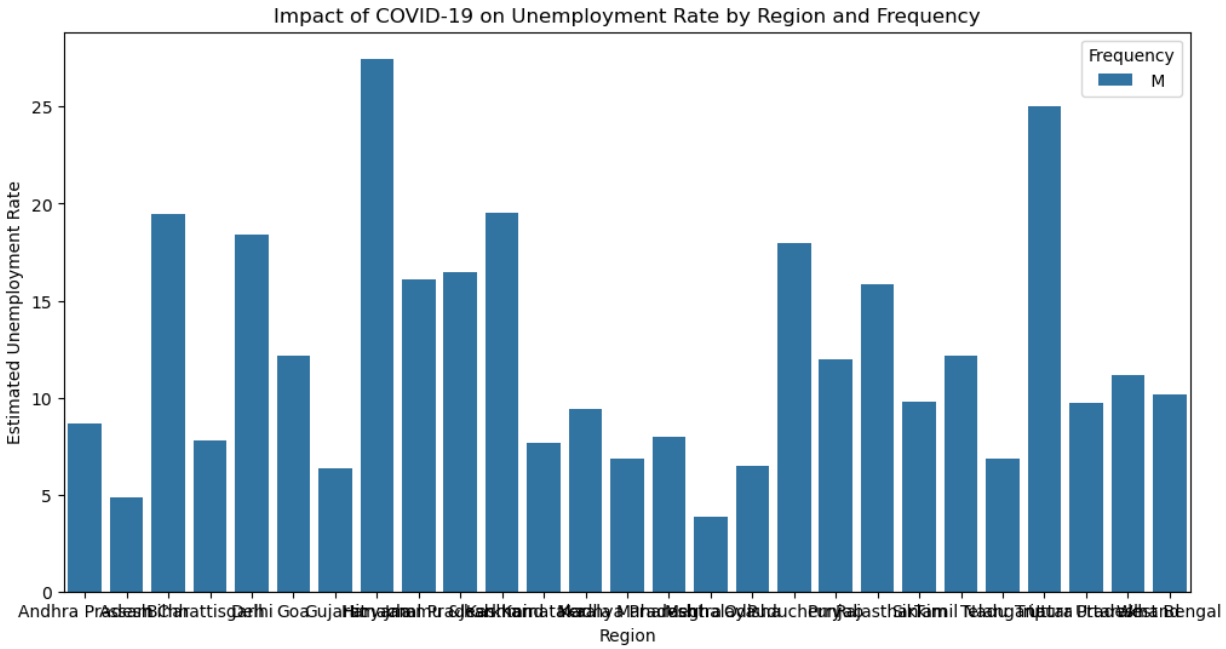


In [42]:
```
# Plot the unemployment rate over time
plt.figure(figsize=(12, 6))
sns.lineplot(data=df, x=df.index, y='Estimated Unemployment Rate (%)', hue='Region')
plt.title('Unemployment Rate Over Time by Region')
plt.xlabel('Date')
plt.ylabel('Estimated Unemployment Rate')
plt.legend(title='Region', loc='upper right')
plt.show()
```

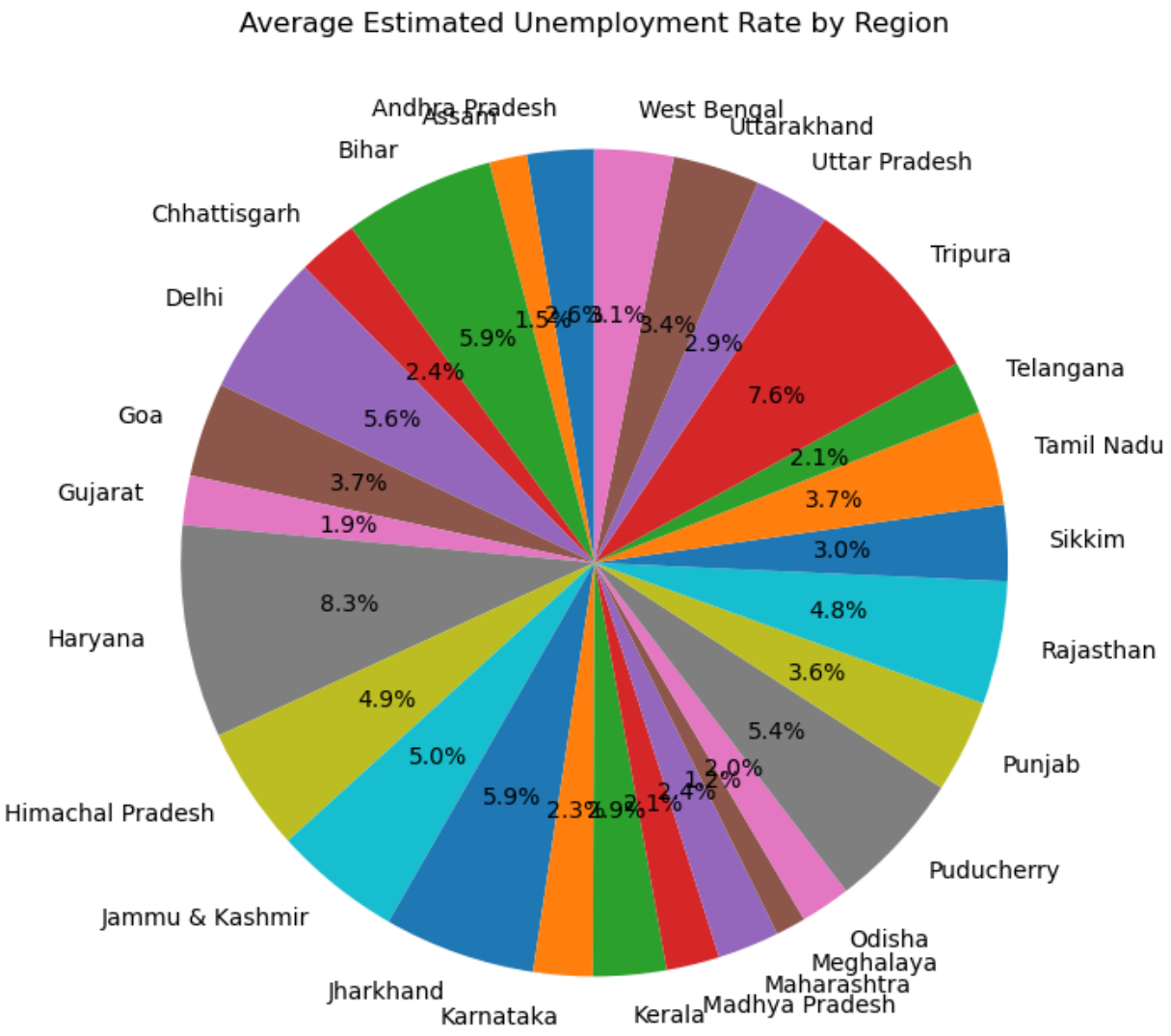


In [43]:
```
# Analyze impact on different demographic groups
plt.figure(figsize=(12, 6))
sns.barplot(data=df, x='Region', y='Estimated Unemployment Rate (%)', hue='Frequency',
plt.title('Impact of COVID-19 on Unemployment Rate by Region and Frequency')
plt.xlabel('Region')
plt.ylabel('Estimated Unemployment Rate')
plt.legend(title='Frequency', loc='upper right')
plt.show()
```

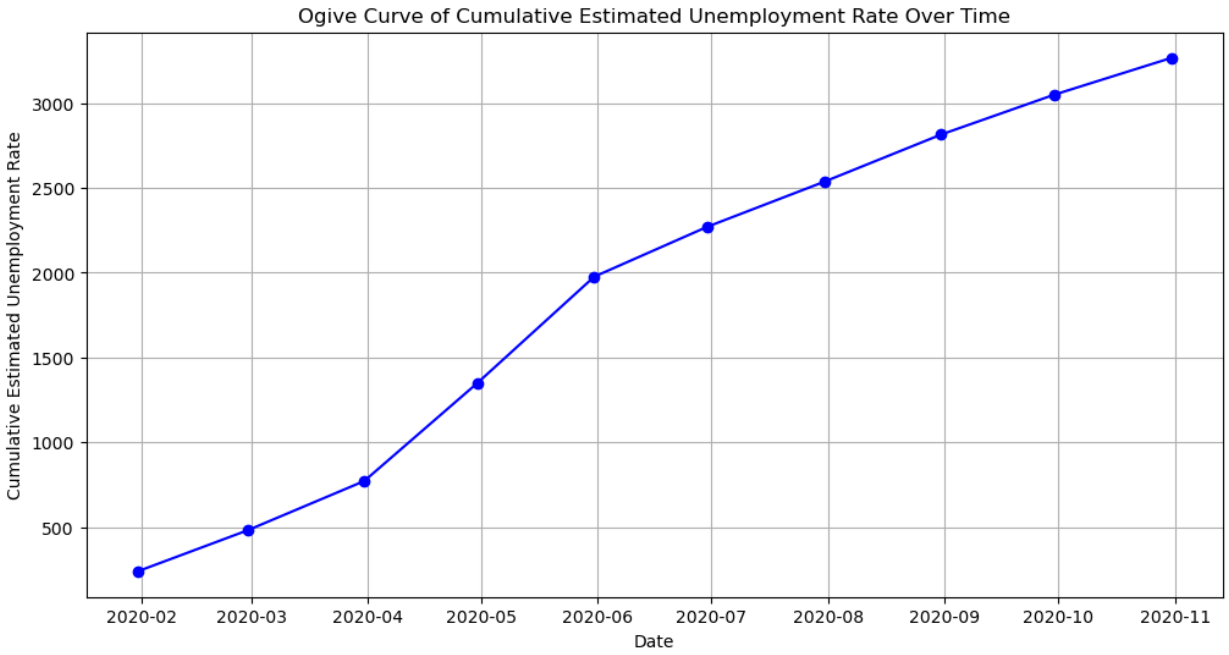
```
C:\Users\warul\AppData\Local\Temp\ipykernel_8336\3272219223.py:3: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(data=df, x='Region', y='Estimated Unemployment Rate (%)', hue='Frequenc
y', ci=None)
```



In [44]:
```
# Create a pie chart for the average estimated unemployment rate by region
average_unemployment_by_region = df.groupby('Region')['Estimated Unemployment Rate (%)

plt.figure(figsize=(8, 8))
plt.pie(average_unemployment_by_region, labels=average_unemployment_by_region.index, a
plt.title('Average Estimated Unemployment Rate by Region')
plt.show()
```



In [45]:
```
# Create an ogive curve for the cumulative sum of the estimated unemployment rate
ogive_data = df.groupby('Date')['Estimated Unemployment Rate (%)'].sum().cumsum()

plt.figure(figsize=(12, 6))
plt.plot(ogive_data.index, ogive_data.values, marker='o', linestyle='-', color='b')
plt.title('Ogive Curve of Cumulative Estimated Unemployment Rate Over Time')
plt.xlabel('Date')
plt.ylabel('Cumulative Estimated Unemployment Rate')
plt.grid(True)
plt.show()
```



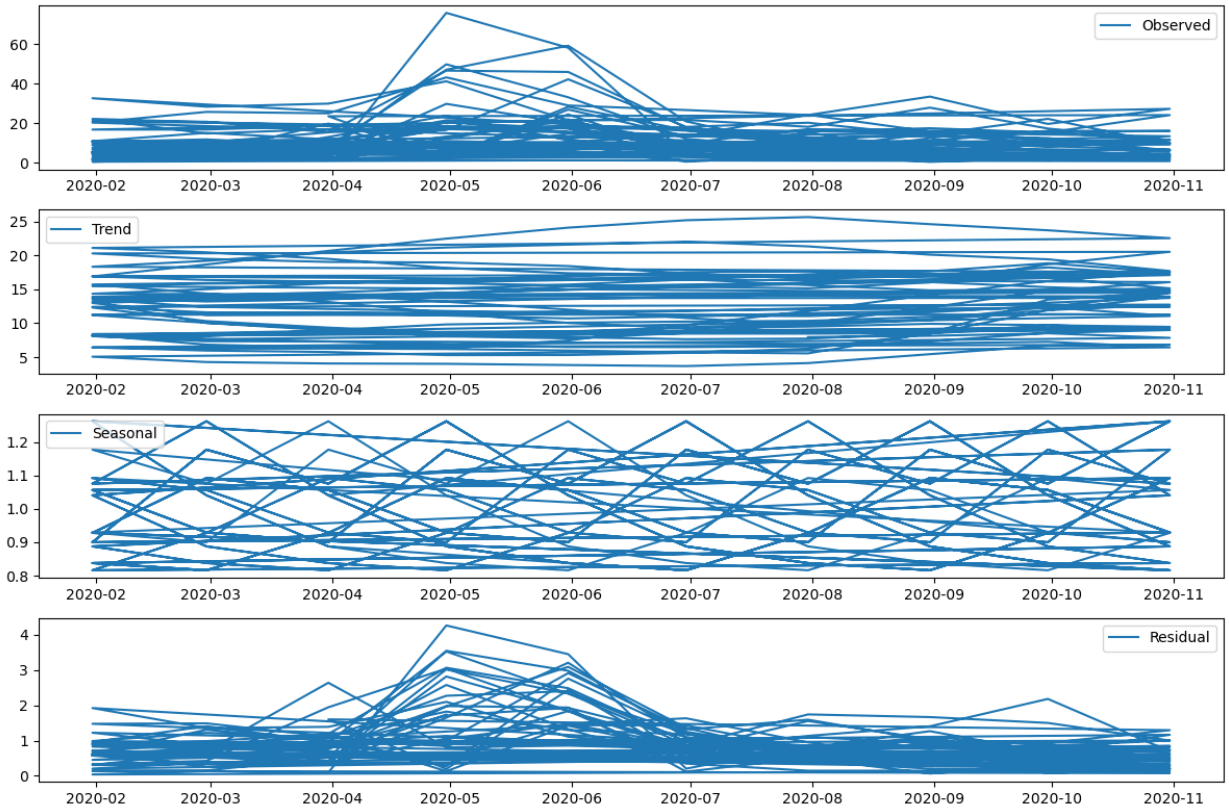
In [41]:
```
# Display basic statistics of the estimated unemployment rate
print(df['Estimated Unemployment Rate (%)'].describe())

# Calculate the average estimated unemployment rate
average_unemployment_rate = df['Estimated Unemployment Rate (%)'].mean()
print(f'Average Estimated Unemployment Rate: {average_unemployment_rate:.2f}%')
```

```
count    267.000000
mean      12.236929
std       10.803283
min        0.500000
25%        4.845000
50%        9.650000
75%       16.755000
max       75.850000
Name: Estimated Unemployment Rate (%), dtype: float64
Average Estimated Unemployment Rate: 12.24%
```

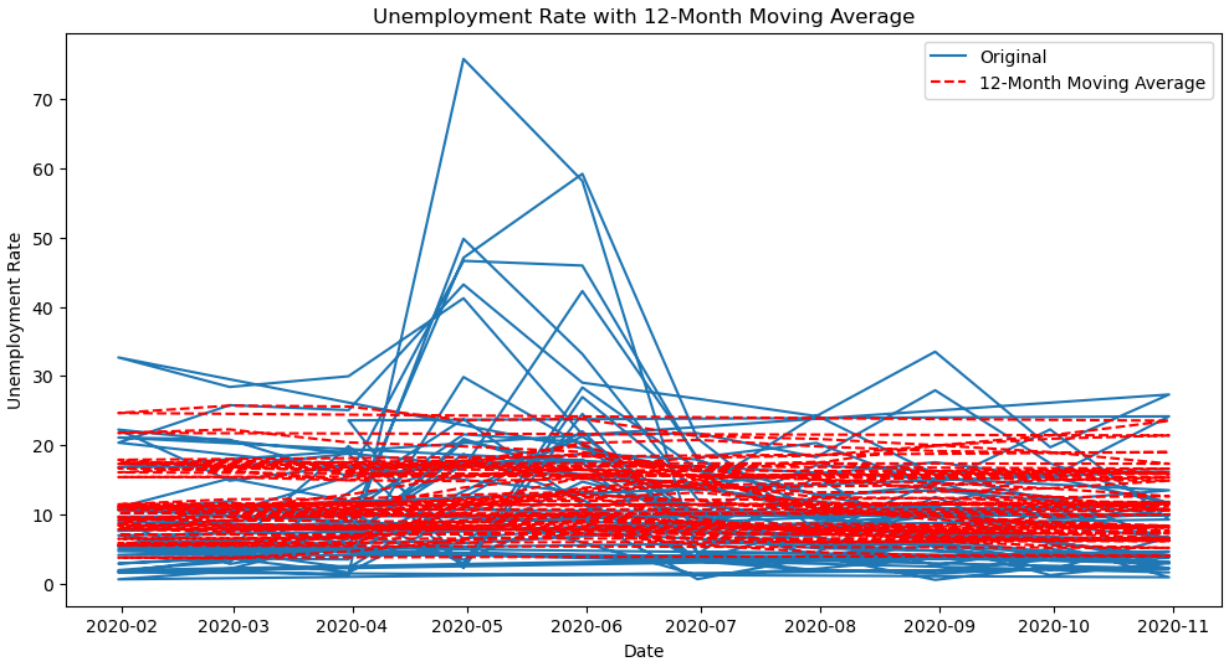
In [49]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA
```

In [50]:
```
# Time Series Decomposition
result = seasonal_decompose(df['Estimated Unemployment Rate (%)'], model='multiplicati

# Plot the decomposition components
plt.figure(figsize=(12, 8))

plt.subplot(4, 1, 1)
plt.plot(result.observed, label='Observed')
plt.legend()

plt.subplot(4, 1, 2)
plt.plot(result.trend, label='Trend')
plt.legend()

plt.subplot(4, 1, 3)
plt.plot(result.seasonal, label='Seasonal')
plt.legend()
```

```
plt.subplot(4, 1, 4)
plt.plot(result.resid, label='Residual')
plt.legend()

plt.tight_layout()
plt.show()
```



In [48]:
```
# Calculate 12-month moving average
df['MA_12'] = df['Estimated Unemployment Rate (%)'].rolling(window=12).mean()

# Plot original and moving average time series
plt.figure(figsize=(12, 6))
plt.plot(df['Estimated Unemployment Rate (%)'], label='Original')
plt.plot(df['MA_12'], label='12-Month Moving Average', linestyle='--', color='red')
plt.legend()
plt.title('Unemployment Rate with 12-Month Moving Average')
plt.xlabel('Date')
plt.ylabel('Unemployment Rate')
plt.show()
```



In [55]:
```
# Example: Replace these values with the ones you identified from the ACF and PACF plo
p = 1  # Autoregressive order
d = 1  # Differencing order
q = 1  # Moving average order

# Fit ARIMA model
model = ARIMA(df['Estimated Unemployment Rate (%)'], order=(p, d, q))
results = model.fit()



# Fit ARIMA model
model = ARIMA(df['Estimated Unemployment Rate (%)'], order=(p, d, q))  # Replace p, d,
results = model.fit()

# Forecast future values
forecast_steps = 12  # Adjust as needed
forecast = results.get_forecast(steps=forecast_steps)

# Plot original and forecasted time series
plt.figure(figsize=(12, 6))
plt.plot(df['Estimated Unemployment Rate (%)'], label='Observed')
plt.plot(forecast.predicted_mean, label='Forecasted', linestyle='--', color='red')
plt.fill_between(forecast.index, forecast.conf_int().iloc[:, 0], forecast.conf_int().i
plt.legend()
plt.title('ARIMA Forecasting')
plt.xlabel('Date')
plt.ylabel('Unemployment Rate')
```

```
C:\Users\warul\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\warul\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: Val
ueWarning: A date index has been provided, but it is not monotonic and so will be ign
ored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\warul\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\warul\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: Val
ueWarning: A date index has been provided, but it is not monotonic and so will be ign
ored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\warul\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\warul\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: Val
ueWarning: A date index has been provided, but it is not monotonic and so will be ign
ored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\warul\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\warul\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: Val
ueWarning: A date index has been provided, but it is not monotonic and so will be ign
ored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\warul\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\warul\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: Val
ueWarning: A date index has been provided, but it is not monotonic and so will be ign
ored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\warul\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\warul\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: Val
ueWarning: A date index has been provided, but it is not monotonic and so will be ign
ored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\warul\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:834: Val
ueWarning: No supported index is available. Prediction results will be given with an
integer index beginning at `start`.
  return get_prediction_index(
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[55], line 24
     22 plt.plot(df['Estimated Unemployment Rate (%)'], label='Observed')
     23 plt.plot(forecast.predicted_mean, label='Forecasted', linestyle='--', color
='red')
---> 24 plt.fill_between(forecast.index, forecast.conf_int().iloc[:, 0], forecast.con
f_int().iloc[:, 1], color='red', alpha=0.2)
     25 plt.legend()
     26 plt.title('ARIMA Forecasting')

File ~\anaconda3\lib\site-packages\statsmodels\base\wrapper.py:34, in ResultsWrapper.
__getattribute__(self, attr)
     31 except AttributeError:
     32     pass
---> 34 obj = getattr(results, attr)
     35 data = results.model.data
     36 how = self._wrap_attrs.get(attr)

AttributeError: 'PredictionResults' object has no attribute 'index'
```

In [56]:

In [ ]:

In [ ]: