Import Libraries and Load Data

In [54]:
```
import pandas as pd
import matplotlib.pyplot as plt
import itertools
import statsmodels.api as sm
import warnings

# Suppress the warning messages
warnings.filterwarnings("ignore")

# Read the CSV file into a pandas DataFrame
file_path = "london_weather[1].csv"
df = pd.read_csv(file_path)

# Prepare the data
df['date'] = pd.to_datetime(df['date'], format='%Y%m%d')
df.set_index('date', inplace=True)
mean_temp_data = df['mean_temp'].dropna()
```

ARIMA Model Building and Selection

In [58]:
```
import warnings
from pmdarima import auto_arima
import itertools
import statsmodels.api as sm

# Suppress the warning messages
warnings.filterwarnings("ignore")

# Set the typical ranges for p, d, q
p = d = q = range(0, 3)

# Take all possible combinations for p, d, and q
pdq = list(itertools.product(p, d, q))

# Using Grid Search, find the optimal ARIMA model that yields the best AIC
best_aic_grid = float("inf")
best_arima_model_grid = None

for param in pdq:
    try:
        model = sm.tsa.ARIMA(mean_temp_data.loc[:'2019'], order=param)
        results = model.fit()
        if results.aic < best_aic_grid:
            best_aic_grid = results.aic
            best_arima_model_grid = results
    except:
        continue

print("Best ARIMA Model from Grid Search (p, d, q):", best_arima_model_grid.params
print("AIC for Best ARIMA Model from Grid Search:", best_aic_grid)
```

```
Best ARIMA Model from Grid Search (p, d, q): const     11.199875
ar.L1      1.582304
ar.L2     -0.585711
ma.L1     -0.695793
ma.L2     -0.157379
sigma2     3.520631
dtype: float64
AIC for Best ARIMA Model from Grid Search: 61240.646935365294
```

In [59]:
```python
import warnings
from pmdarima import auto_arima

# Suppress the warning messages
warnings.filterwarnings("ignore")

# Using Auto-ARIMA, find the optimal ARIMA model
auto_arima_model = auto_arima(mean_temp_data.loc[:'2019'], seasonal=False, suppres
best_arima_order = auto_arima_model.order

print("Best ARIMA Model Order from Auto-ARIMA:", best_arima_order)
```

```
Best ARIMA Model Order from Auto-ARIMA: (1, 1, 2)
```

In [14]:
```python
# Choose the best model based on AIC values
if best_aic_grid < best_aic_auto:
    best_model = best_arima_model_grid
    best_source = "Grid Search"
else:
    best_model = best_arima_model_auto
    best_source = "Auto ARIMA"

print("Best ARIMA Model Source:", best_source)
print("Best ARIMA Model Summary:")
print(best_model.summary())
```

```
Best ARIMA Model Source: Grid Search
Best ARIMA Model Summary:
                               SARIMAX Results
==============================================================================
Dep. Variable:              mean_temp   No. Observations:                14946
Model:                 ARIMA(2, 0, 2)   Log Likelihood              -30614.323
Date:                Tue, 08 Aug 2023   AIC                          61240.647
Time:                        14:00:33   BIC                          61286.320
Sample:                             0   HQIC                         61255.803
                                - 14946
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         11.1999      0.646     17.343      0.000       9.934      12.466
ar.L1          1.5823      0.017     94.342      0.000       1.549       1.615
ar.L2         -0.5857      0.016    -35.537      0.000      -0.618      -0.553
ma.L1         -0.6958      0.018    -39.125      0.000      -0.731      -0.661
ma.L2         -0.1574      0.011    -14.868      0.000      -0.178      -0.137
sigma2         3.5206      0.039     89.524      0.000       3.444       3.598
===================================================================================
=
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):               127.8
9
Prob(Q):                              0.97   Prob(JB):                         0.0
0
Heteroskedasticity (H):               0.95   Skew:                            -0.2
0
Prob(H) (two-sided):                  0.09   Kurtosis:                         3.2
1
===================================================================================
=

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

Forecast Next Year

In [60]:

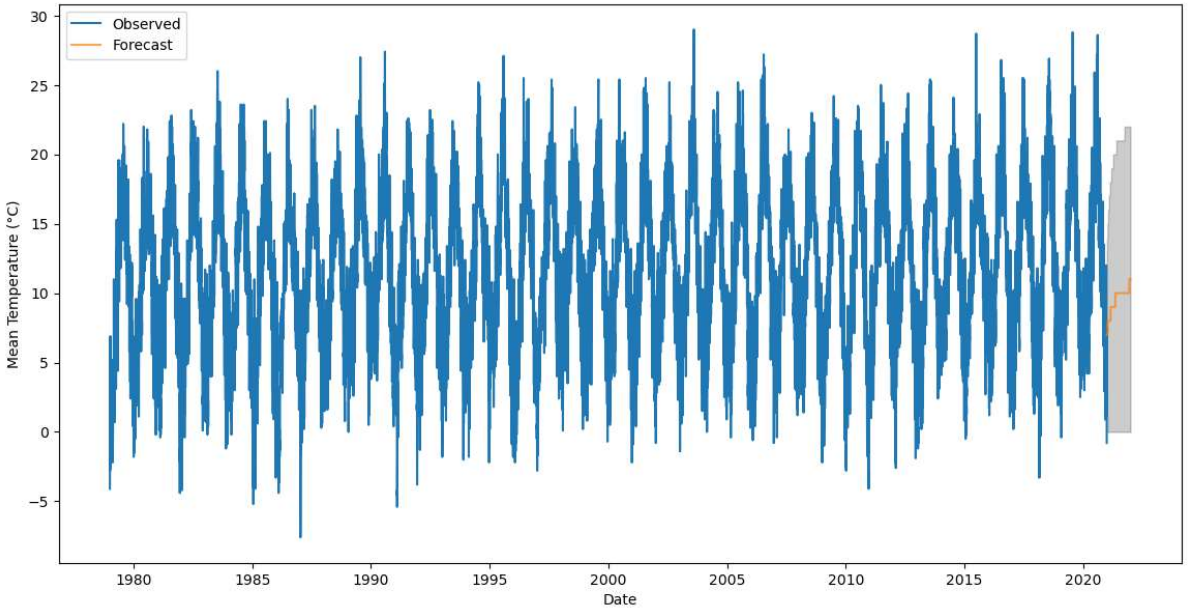
```
forecast_horizon = 365
forecast_dates = pd.date_range(start=mean_temp_data.index[-1], periods=forecast_ho

# Forecast using the best ARIMA model
forecast = best_model.get_forecast(steps=forecast_horizon)

# Extract forecasted mean and confidence intervals
forecast_mean = forecast.predicted_mean
forecast_ci = forecast.conf_int()

# Convert forecast values to integers
forecast_mean_int = forecast_mean.astype(int)
forecast_ci_int = forecast_ci.astype(int)

# Plot the forecast for the next five years
plt.figure(figsize=(14, 7))
plt.plot(mean_temp_data.index, mean_temp_data, label='Observed')
plt.plot(forecast_dates, forecast_mean_int, label='Forecast', alpha=0.7)
plt.fill_between(forecast_dates, forecast_ci_int.iloc[:, 0], forecast_ci_int.iloc[
plt.xlabel("Date")
plt.ylabel('Mean Temperature (°C)')
plt.legend()
plt.show()
```


Observed
Forecast
Mean Temperature (°C)
30
25
20
15
10
5
0
−5
1980
1985
1990
1995
2000
2005
2010
2015
2020
Date

In [ ]: