Import Libaries and Load Data

In [11]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
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()
```

Split Data into Train and Test Sets

In [12]:

```
# Normalize the data
scaler = MinMaxScaler()
normalized_data = scaler.fit_transform(mean_temp_data.values.reshape(-1, 1))

# Split data into training and testing sets
train_size = int(0.8 * len(normalized_data))
train_data, test_data = normalized_data[:train_size], normalized_data[train_size:]
```

Fit ETS Model

In [14]:

```
trend_list = ['add', None]
seasonal_list = ['add', None]

# Initialize variables to keep track of the best model
best_mse = np.inf
best_ets_fit = None

# Iterate through different trend and seasonal parameters
for trend in trend_list:
    for seasonal in seasonal_list:
        # Fit ETS model to training data
        ets_model = ExponentialSmoothing(train_data, trend=trend, seasonal=seasona
        ets_fit = ets_model.fit()

        # Make predictions
        test_predictions = ets_fit.forecast(len(test_data))

        # Inverse transform the predictions
        test_predictions = scaler.inverse_transform(test_predictions.reshape(-1, 1

        # Calculate Mean Squared Error
        mse = mean_squared_error(mean_temp_data.values[train_size:], test_predicti
        # Compare MSE and update best model if needed
        if mse < best_mse:
            best_mse = mse
            best_ets_fit = ets_fit
```

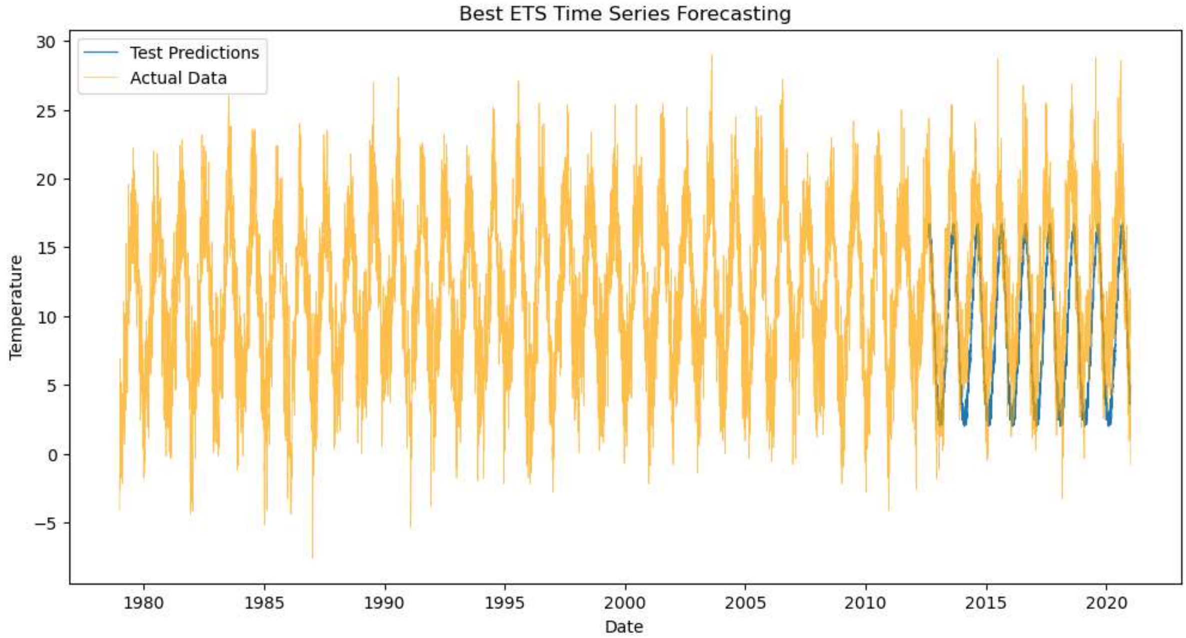
In [15]:

```
# Make final predictions with the best model
final_test_predictions = best_ets_fit.forecast(len(test_data))
final_test_predictions = scaler.inverse_transform(final_test_predictions.reshape(-
```

Plot Results

In [19]:

```
# Plot the results of the best model
plt.figure(figsize=(12, 6))
plt.plot(mean_temp_data.index[train_size:], final_test_predictions, label='Test Pr
plt.plot(mean_temp_data.index, mean_temp_data.values, label='Actual Data', color='
plt.legend()
plt.xlabel('Date')
plt.ylabel('Temperature')
plt.title('Best ETS Time Series Forecasting')
plt.show()
```


Best ETS Time Series Forecasting
Test Predictions
Actual Data
Temperature
Date