In [1]:

```python
import pandas as pd
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller
from pmdarima import auto_arima
import warnings
from sklearn.metrics import mean_squared_error
from math import sqrt

warnings.filterwarnings("ignore")
def ad_test(data):
    dftest = adfuller(data,autolag = 'AIC')
    print("1. ADf:",dftest[0])
    print("2. p-value:",dftest[1])
    print("3. Num of Lags:",dftest[2])
    print("4. Num of observations used:",dftest[3])
    print("5. Critical Values:",dftest[3])
    for key,val in dftest[4].items():
        print(key,":",val)


# Load the data
data = pd.read_csv("ForecastData.csv", index_col="Month", parse_dates=True)
print(data)
ad_test(data)
# Fit ARIMA model
#model = ARIMA(data, order=(p, d, q))
#model_fit = model.fit()

# Forecast temperatures for Jan-24 to Dec-24
#forecast = model_fit.forecast(steps=12)
#print(forecast)
stepwise_fit  = auto_arima(data['Temperature'],trace = True,supress_warnings
stepwise_fit.summary()

train = data.iloc[:-30]
test = data.iloc[-30:]
print(train.shape,test.shape)
```

```
              Temperature
Month
Jan-00              82
Feb-00              82
Mar-00              85
Apr-00              90
May-00              97
...                ...
Aug-23             110
Sep-23             110
Oct-23              96
Nov-23              86
Dec-23              80

[288 rows x 1 columns]
1. ADf: -4.167090021487841
2. p-value: 0.0007487107862045819
3. Num of Lags: 16
4. Num of observations used: 271
5. Critical Values: 271
1% : -3.4547128138328875
5% : -2.8722649771800155
10% : -2.5724850011573914
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=1703.141, Time=0.87 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=2140.786, Time=0.02 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=1868.930, Time=0.08 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=1950.257, Time=0.11 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=3421.092, Time=0.04 sec
 ARIMA(1,0,2)(0,0,0)[0] intercept   : AIC=1818.365, Time=0.44 sec
 ARIMA(2,0,1)(0,0,0)[0] intercept   : AIC=1757.996, Time=0.59 sec
 ARIMA(3,0,2)(0,0,0)[0] intercept   : AIC=1745.490, Time=0.80 sec
 ARIMA(2,0,3)(0,0,0)[0] intercept   : AIC=1641.765, Time=0.80 sec
 ARIMA(1,0,3)(0,0,0)[0] intercept   : AIC=1791.416, Time=0.67 sec
 ARIMA(3,0,3)(0,0,0)[0] intercept   : AIC=1673.786, Time=0.87 sec
 ARIMA(2,0,4)(0,0,0)[0] intercept   : AIC=1694.819, Time=1.00 sec
 ARIMA(1,0,4)(0,0,0)[0] intercept   : AIC=inf, Time=0.91 sec
 ARIMA(3,0,4)(0,0,0)[0] intercept   : AIC=1650.033, Time=1.09 sec
 ARIMA(2,0,3)(0,0,0)[0]             : AIC=1861.991, Time=0.40 sec

Best model:  ARIMA(2,0,3)(0,0,0)[0] intercept
Total fit time: 8.689 seconds
(258, 1) (30, 1)
```

In [2]: ▶|

```python
train = data.iloc[:-30]
test = data.iloc[-30:]
print(train.shape,test.shape)

model = ARIMA(train['Temperature'],order = (2,0,3))
model = model.fit()
model.summary()
```

(258, 1) (30, 1)

Out[2]:

SARIMAX Results

Dep. Variable:	Temperature	No. Observations:	258
Model:	ARIMA(2, 0, 3)	Log Likelihood	-722.196
Date:	Wed, 01 May 2024	AIC	1458.392
Time:	15:19:53	BIC	1483.263
Sample:	0	HQIC	1468.393
	- 258		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	90.6981	0.342	265.081	0.000	90.028	91.369
ar.L1	1.7321	0.001	2079.580	0.000	1.730	1.734
ar.L2	-0.9998	0.000	-2740.549	0.000	-1.001	-0.999
ma.L1	-1.4233	0.256	-5.571	0.000	-1.924	-0.923
ma.L2	0.5008	0.233	2.151	0.031	0.044	0.957
ma.L3	0.2896	0.115	2.520	0.012	0.064	0.515
sigma2	15.0644	4.572	3.295	0.001	6.104	24.025

Ljung-Box (L1) (Q):	0.73	Jarque-Bera (JB):	2.43
Prob(Q):	0.39	Prob(JB):	0.30
Heteroskedasticity (H):	0.67	Skew:	-0.23
Prob(H) (two-sided):	0.07	Kurtosis:	3.15

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

In [7]: ▶

```python
start = len(train)
end = len(train) +len(test) -1
pred = model.predict(start = start,end = end,type = 'levels')
pred = pd.DataFrame(pred,columns = ['predicted_mean'])
predicted_index = pd.date_range(start='2021-07-01', periods=len(pred), freq=
predicted_index_str = predicted_index.strftime('%b-%y')

pred['predicted_index'] = predicted_index_str
pred.set_index('predicted_index', inplace=True)

print(pred)

print(test['Temperature'])
pred.plot(legend=True,label='Temperature predicted by model')
test['Temperature'].plot(legend = True)
```

```
                         predicted_mean
predicted_index
Jul-21                      102.910477
Aug-21                      102.158062
Sep-21                       98.068701
Oct-21                       92.006616
Nov-21                       85.595212
Dec-21                       80.551146
Jan-22                       78.224694
Feb-22                       79.238285
Mar-22                       83.319976
Apr-22                       89.376412
May-22                       95.785700
Jun-22                      100.831748
Jul-22                      103.163751
Aug-22                      102.157791
Sep-22                       98.083769
Oct-22                       92.032986
Nov-22                       85.625821
Dec-22                       80.577799
Jan-23                       78.240255
Feb-23                       79.238590
Mar-23                       83.304945
Apr-23                       89.350072
May-23                       95.755106
Jun-23                      100.805092
Jul-23                      103.148169
Aug-23                      102.157453
Sep-23                       98.098762
Oct-23                       92.059294
Nov-23                       85.656399
Dec-23                       80.604458
Month
Jul-21     102
Aug-21     100
Sep-21     100
Oct-21      93
Nov-21      83
Dec-21      83
Jan-22      79
Feb-22      85
Mar-22      94
Apr-22      95
May-22      97
Jun-22     103
Jul-22     109
Aug-22     104
Sep-22      98
Oct-22      95
Nov-22      83
Dec-22      81
Jan-23      85
Feb-23      90
Mar-23      88
Apr-23      94
May-23      96
Jun-23     102
Jul-23     108
```

```
Aug-23     110
Sep-23     110
Oct-23      96
Nov-23      86
Dec-23      80
Name: Temperature, dtype: int64
```

Out[7]:   <Axes: xlabel='Month'>
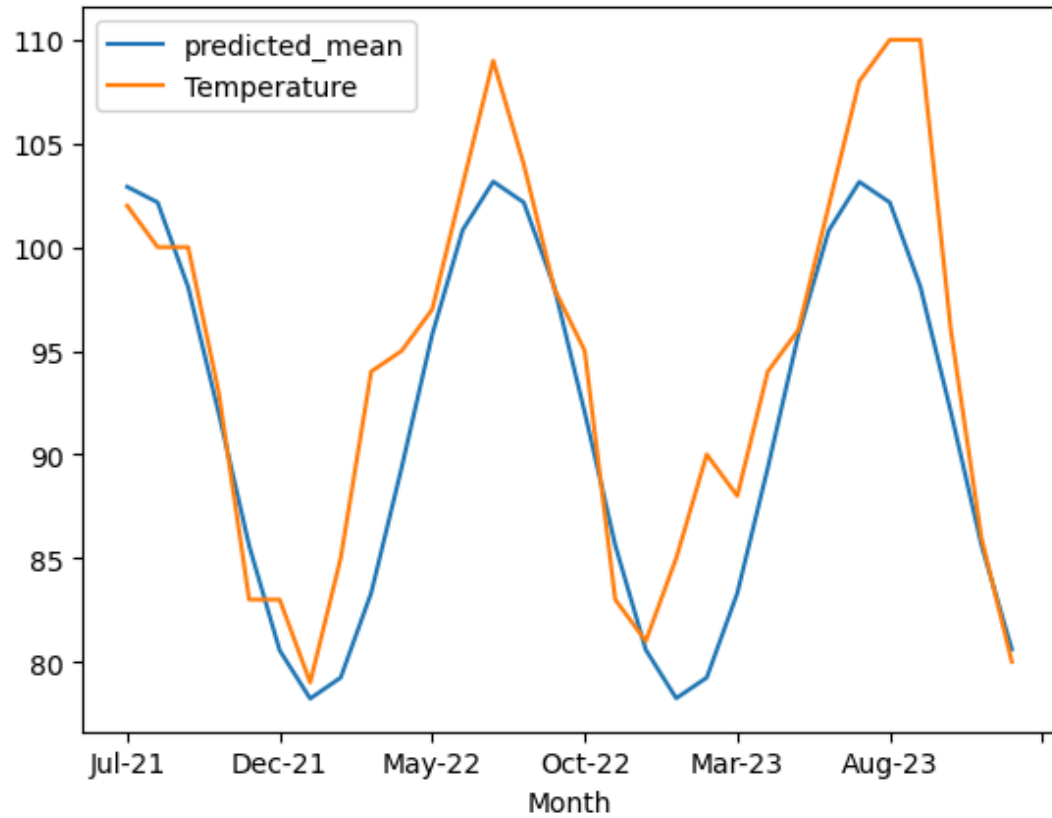
In [4]: ▶|
```python
print(test['Temperature'].mean())
rmse = sqrt(mean_squared_error(pred,test['Temperature']))
print(rmse)
```

```
94.3
4.869253682983691
```

In [5]: ▶|
```python
index_future_months = pd.date_range(start='2024-01-01',end='2025-01-01', fre
index_future_months = index_future_months.strftime('%b-%y')
print(index_future_months)
pred2 = model.predict(start = len(data),end = len(data)+11,type = 'levels').
pred2.index = index_future_months
print(pred2)
```

```
Index(['Jan-24', 'Feb-24', 'Mar-24', 'Apr-24', 'May-24', 'Jun-24', 'Jul-2
4',
        'Aug-24', 'Sep-24', 'Oct-24', 'Nov-24', 'Dec-24'],
      dtype='object')
Jan-24      78.255859
Feb-24      79.238962
Mar-24      83.289989
Apr-24      89.323795
May-24      95.724544
Jun-24     100.778430
Jul-24     103.132543
Aug-24     102.157047
Sep-24      98.113681
Oct-24      92.085541
Nov-24      85.686945
Dec-24      80.631124
Name: ARIMA PRedictions, dtype: float64
```