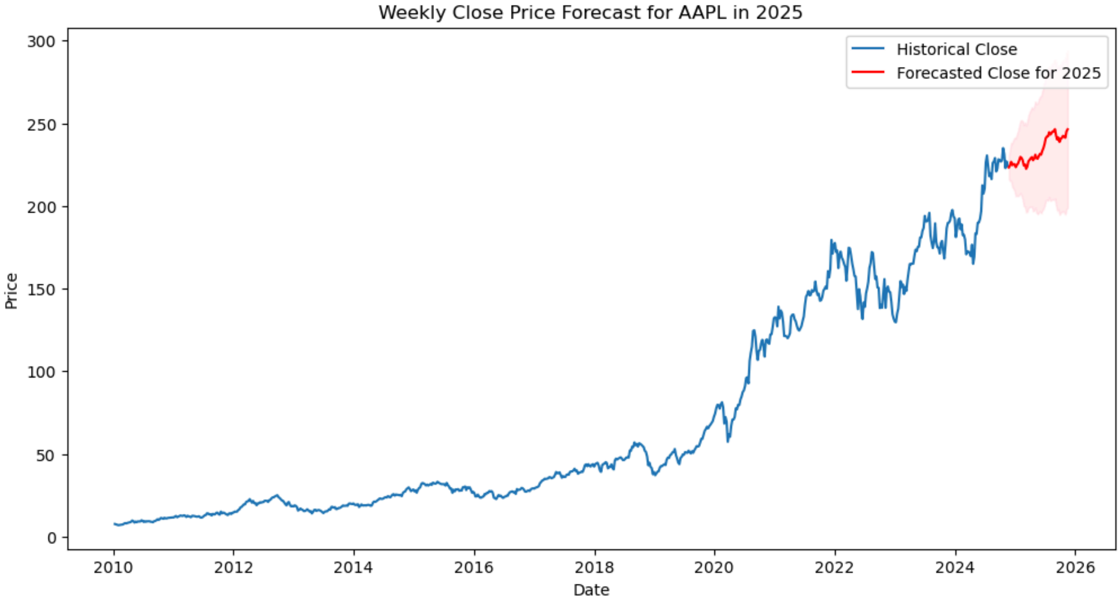
In [16]:

```
import yfinance as yf
import pandas as pd
from statsmodels.tsa.statespace.sarimax import SARIMAX
import matplotlib.pyplot as plt
import datetime

# Define the stock symbol and download historical data
symbol = "AAPL"
start_date = "2010-01-01"
end_date = datetime.datetime.now()

# Download data
stock_data = yf.download(symbol, start=start_date, end=end_date)

# Step 1: Resample to weekly frequency using MultiIndex column names
weekly_data = stock_data.resample('W').agg({
    ('Open', 'AAPL'): 'first',     # First observation of the week for Open
    ('High', 'AAPL'): 'max',       # Maximum of the week for High
    ('Low', 'AAPL'): 'min',        # Minimum of the week for Low
    ('Close', 'AAPL'): 'last'      # Last observation of the week for Close
})

# Drop any rows with NaN values to ensure continuity
weekly_data = weekly_data.dropna()

# Define a function to forecast using SARIMA for each column
def forecast_sarima(series, steps=52):
    # Fit SARIMA model with weekly seasonality
    model = SARIMAX(series, order=(1, 1, 1), seasonal_order=(1, 1, 1, 52))
    model_fit = model.fit(disp=False)

    # Forecast for the desired number of future steps
    forecast = model_fit.get_forecast(steps=steps)
    forecast_values = forecast.predicted_mean
    confidence_intervals = forecast.conf_int()

    # Adjust index for forecast to start after the last historical date
    forecast_index = pd.date_range(start=series.index[-1] + pd.Timedelta(we
    forecast_values.index = forecast_index
    confidence_intervals.index = forecast_index

    return forecast_values, confidence_intervals

# Forecast for 2025 (52 weeks) using MultiIndex column names
open_forecast, open_conf = forecast_sarima(weekly_data[('Open', 'AAPL')], s
high_forecast, high_conf = forecast_sarima(weekly_data[('High', 'AAPL')], s
low_forecast, low_conf = forecast_sarima(weekly_data[('Low', 'AAPL')], step
close_forecast, close_conf = forecast_sarima(weekly_data[('Close', 'AAPL')]

# Combine forecasts into a DataFrame for easy viewing
forecast_2025 = pd.DataFrame({
    'Open_AAPL': open_forecast,
    'High_AAPL': high_forecast,
    'Low_AAPL': low_forecast,
    'Close_AAPL': close_forecast
})
```

In [18]:

```
# Plot the forecast for Close price as an example
plt.figure(figsize=(12, 6))
plt.plot(weekly_data[('Close', 'AAPL')], label='Historical Close')
plt.plot(close_forecast.index, close_forecast, color='red', label='Forecas
plt.fill_between(close_forecast.index, close_conf.iloc[:, 0], close_conf.i
plt.legend()
plt.title(f"Weekly Close Price Forecast for {symbol} in 2025")
plt.xlabel("Date")
plt.ylabel("Price")
plt.show()

# Display forecasted values
print("Forecasted weekly stock prices for 2025:")
print(forecast_2025)
```

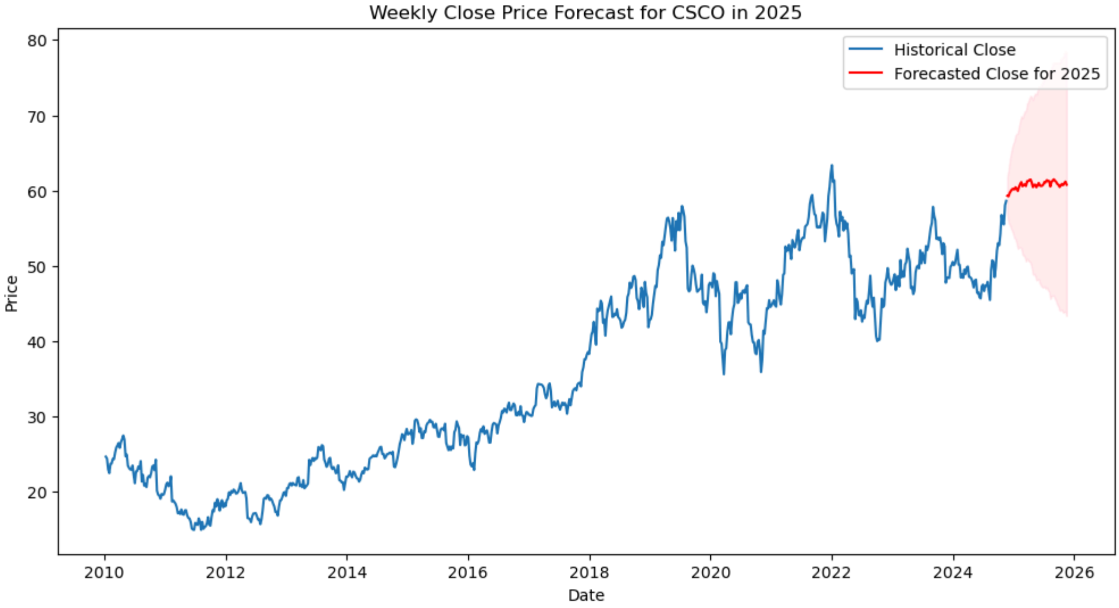


```
Forecasted weekly stock prices for 2025:
                            Open_AAPL   High_AAPL    Low_AAPL  Close_AAPL
2024-11-24 00:00:00+00:00  226.976881  225.314200  222.686236  223.306115
2024-12-01 00:00:00+00:00  225.758306  226.048951  221.862792  224.565380
2024-12-08 00:00:00+00:00  227.179032  228.126637  222.997682  226.705668
2024-12-15 00:00:00+00:00  229.406287  229.313351  223.630961  224.675753
2024-12-22 00:00:00+00:00  226.381739  226.844453  222.995321  225.342573
2024-12-29 00:00:00+00:00  227.981636  226.753910  224.198072  225.219239
2025-01-05 00:00:00+00:00  226.942726  225.638977  221.232544  223.448920
2025-01-12 00:00:00+00:00  224.954147  225.373362  222.163913  224.835321
2025-01-19 00:00:00+00:00  226.256619  227.046062  222.256448  225.739517
2025-01-26 00:00:00+00:00  227.953384  229.530886  223.735432  227.557695
2025-02-02 00:00:00+00:00  229.682772  230.596311  224.719707  229.655983
2025-02-09 00:00:00+00:00  231.713854  230.405589  227.604294  229.054809
2025-02-16 00:00:00+00:00  230.789496  229.649505  226.440606  227.655665
2025-02-23 00:00:00+00:00  228.290798  226.559699  221.466043  224.467992
2025-03-02 00:00:00+00:00  226.814662  226.607984  222.491427  225.087420
2025-03-09 00:00:00+00:00  226.530491  225.269470  220.272337  222.482090
2025-03-16 00:00:00+00:00  222.996567  224.903368  219.947192  223.830013
2025-03-23 00:00:00+00:00  226.003517  227.306805  222.245539  227.004296
2025-03-30 00:00:00+00:00  228.317873  227.779259  224.318189  227.878385
2025-04-06 00:00:00+00:00  229.974472  228.937471  225.583986  228.872662
2025-04-13 00:00:00+00:00  229.726175  229.470802  225.684230  229.548031
2025-04-20 00:00:00+00:00  230.926968  229.456667  225.491594  227.656085
2025-04-27 00:00:00+00:00  229.333376  229.097652  224.846172  228.673186
2025-05-04 00:00:00+00:00  230.680465  231.951040  225.331152  230.993087
2025-05-11 00:00:00+00:00  232.114167  229.957726  225.903792  228.830238
2025-05-18 00:00:00+00:00  231.058061  230.058992  225.947029  228.572603
2025-05-25 00:00:00+00:00  230.628476  230.158356  227.028617  230.213905
2025-06-01 00:00:00+00:00  232.922789  231.618212  229.203239  231.513201
2025-06-08 00:00:00+00:00  234.451376  233.061560  229.497755  231.018883
2025-06-15 00:00:00+00:00  232.768200  235.456978  229.262581  233.272076
2025-06-22 00:00:00+00:00  236.321417  236.558768  232.934869  234.979432
2025-06-29 00:00:00+00:00  237.978379  238.047682  233.955077  237.438783
2025-07-06 00:00:00+00:00  240.352438  241.020602  237.055930  240.912396
2025-07-13 00:00:00+00:00  243.964569  242.953027  239.625628  241.978597
2025-07-20 00:00:00+00:00  246.040244  245.181668  240.240277  242.154334
2025-07-27 00:00:00+00:00  246.021724  245.856421  240.276399  244.610532
2025-08-03 00:00:00+00:00  247.488766  246.217067  241.661956  243.210118
2025-08-10 00:00:00+00:00  243.836289  244.115027  238.748518  243.968717
2025-08-17 00:00:00+00:00  246.789236  246.512071  242.254797  245.065713
2025-08-24 00:00:00+00:00  248.355125  246.737253  243.890256  245.151012
2025-08-31 00:00:00+00:00  248.371249  248.988394  242.776929  246.465562
2025-09-07 00:00:00+00:00  248.937337  246.814472  240.808959  242.599675
2025-09-14 00:00:00+00:00  246.571787  244.494518  239.555521  240.158483
2025-09-21 00:00:00+00:00  241.811670  243.887036  238.401640  241.404198
2025-09-28 00:00:00+00:00  244.198739  242.929823  238.085429  238.714111
2025-10-05 00:00:00+00:00  241.835468  242.580457  238.438921  240.566623
2025-10-12 00:00:00+00:00  243.283332  243.272716  239.377111  241.132852
2025-10-19 00:00:00+00:00  244.144343  244.626902  240.924982  242.404054
2025-10-26 00:00:00+00:00  244.998512  245.274347  239.915017  242.051886
2025-11-02 00:00:00+00:00  244.978081  245.288463  238.218143  241.255344
2025-11-09 00:00:00+00:00  243.606422  245.160595  239.958566  245.009983
2025-11-16 00:00:00+00:00  247.527302  246.823001  243.534384  246.347833
```

In [19]:

```
import yfinance as yf
import pandas as pd
from statsmodels.tsa.statespace.sarimax import SARIMAX
import matplotlib.pyplot as plt
import datetime

# Define the stock symbol and download historical data
symbol = "CSCO"
start_date = "2010-01-01"
end_date = datetime.datetime.now()

# Download data
stock_data = yf.download(symbol, start=start_date, end=end_date)

# Step 1: Resample to weekly frequency using MultiIndex column names
weekly_data = stock_data.resample('W').agg({
    ('Open', 'CSCO'): 'first',     # First observation of the week for Oper
    ('High', 'CSCO'): 'max',       # Maximum of the week for High
    ('Low', 'CSCO'): 'min',        # Minimum of the week for Low
    ('Close', 'CSCO'): 'last'      # Last observation of the week for Close
})

# Drop any rows with NaN values to ensure continuity
weekly_data = weekly_data.dropna()

# Define a function to forecast using SARIMA for each column
def forecast_sarima(series, steps=52):
    # Fit SARIMA model with weekly seasonality
    model = SARIMAX(series, order=(1, 1, 1), seasonal_order=(1, 1, 1, 52))
    model_fit = model.fit(disp=False)

    # Forecast for the desired number of future steps
    forecast = model_fit.get_forecast(steps=steps)
    forecast_values = forecast.predicted_mean
    confidence_intervals = forecast.conf_int()

    # Adjust index for forecast to start after the last historical date
    forecast_index = pd.date_range(start=series.index[-1] + pd.Timedelta(we
    forecast_values.index = forecast_index
    confidence_intervals.index = forecast_index

    return forecast_values, confidence_intervals

# Forecast for 2025 (52 weeks) using MultiIndex column names
open_forecast, open_conf = forecast_sarima(weekly_data[('Open', 'CSCO')], s
high_forecast, high_conf = forecast_sarima(weekly_data[('High', 'CSCO')], s
low_forecast, low_conf = forecast_sarima(weekly_data[('Low', 'CSCO')], step
close_forecast, close_conf = forecast_sarima(weekly_data[('Close', 'CSCO')]

# Combine forecasts into a DataFrame for easy viewing
forecast_2025 = pd.DataFrame({
    'Open_CSCO': open_forecast,
    'High_CSCO': high_forecast,
    'Low_CSCO': low_forecast,
    'Close_CSCO': close_forecast
})
```

In [21]:
```
# Plot the forecast for Close price as an example
plt.figure(figsize=(12, 6))
plt.plot(weekly_data[('Close', 'CSCO')], label='Historical Close')
plt.plot(close_forecast.index, close_forecast, color='red', label='Forecas
plt.fill_between(close_forecast.index, close_conf.iloc[:, 0], close_conf.i
plt.legend()
plt.title(f"Weekly Close Price Forecast for {symbol} in 2025")
plt.xlabel("Date")
plt.ylabel("Price")
plt.show()

# Display forecasted values
print("Forecasted weekly stock prices for 2025:")
print(forecast_2025)
```



```
Forecasted weekly stock prices for 2025:
                           Open_CSCO  High_CSCO   Low_CSCO  Close_CSCO
2024-11-24 00:00:00+00:00  58.569331  58.700652  59.446253   59.302824
2024-12-01 00:00:00+00:00  59.168412  59.092060  59.606419   59.232228
2024-12-08 00:00:00+00:00  58.987305  59.240557  59.858578   59.692030
2024-12-15 00:00:00+00:00  59.504071  59.898320  60.212830   59.938911
2024-12-22 00:00:00+00:00  59.456423  59.879523  60.505938   60.155219
2024-12-29 00:00:00+00:00  59.739247  59.995353  60.762123   60.239008
2025-01-05 00:00:00+00:00  59.707944  60.161179  60.565207   60.135888
2025-01-12 00:00:00+00:00  59.533939  60.214233  60.781551   60.438307
2025-01-19 00:00:00+00:00  59.640440  60.323287  60.534526   60.237813
2025-01-26 00:00:00+00:00  59.404789  60.440765  60.405634   59.940788
2025-02-02 00:00:00+00:00  59.168621  60.539933  60.326174   60.472741
2025-02-09 00:00:00+00:00  59.699561  60.791565  60.858673   60.851292
2025-02-16 00:00:00+00:00  59.883190  60.767634  61.094470   61.122813
2025-02-23 00:00:00+00:00  59.979362  60.588787  60.744781   60.572359
2025-03-02 00:00:00+00:00  59.512456  60.227124  60.720925   60.744292
2025-03-09 00:00:00+00:00  59.459587  60.579989  60.806455   60.845357
2025-03-16 00:00:00+00:00  59.417802  60.724542  60.877170   60.597453
2025-03-23 00:00:00+00:00  59.479462  60.986231  61.000024   61.238425
2025-03-30 00:00:00+00:00  59.964657  61.013811  61.510085   61.264853
2025-04-06 00:00:00+00:00  60.034695  61.148997  61.520411   61.401181
2025-04-13 00:00:00+00:00  60.021678  61.176341  61.734442   61.468335
2025-04-20 00:00:00+00:00  60.085172  61.125542  61.183248   61.046400
2025-04-27 00:00:00+00:00  59.695128  60.437079  60.789717   60.454102
2025-05-04 00:00:00+00:00  58.962443  60.295402  60.479050   60.764520
2025-05-11 00:00:00+00:00  59.121209  60.316308  60.452992   60.776034
2025-05-18 00:00:00+00:00  59.280224  60.698067  60.579692   60.438714
2025-05-25 00:00:00+00:00  58.950222  60.061408  60.633249   60.718515
2025-06-01 00:00:00+00:00  59.006300  60.006275  60.867595   61.009009
2025-06-08 00:00:00+00:00  59.361223  60.381229  60.890981   60.658939
2025-06-15 00:00:00+00:00  58.811344  59.924751  60.517732   60.602147
2025-06-22 00:00:00+00:00  58.804909  59.984842  60.458892   60.641307
2025-06-29 00:00:00+00:00  58.766853  60.070130  60.834795   60.815762
2025-07-06 00:00:00+00:00  58.871873  60.239403  60.949797   61.149846
2025-07-13 00:00:00+00:00  59.293817  60.486633  61.078688   61.086342
2025-07-20 00:00:00+00:00  59.179899  60.813077  61.283072   61.384363
2025-07-27 00:00:00+00:00  59.465776  60.767117  61.071801   61.285498
2025-08-03 00:00:00+00:00  59.238606  60.753051  61.255309   61.253183
2025-08-10 00:00:00+00:00  59.339633  60.441329  60.436154   60.524967
2025-08-17 00:00:00+00:00  58.455066  60.738579  60.420911   61.136880
2025-08-24 00:00:00+00:00  59.086831  60.995979  61.605445   61.319519
2025-08-31 00:00:00+00:00  59.273031  61.179408  61.687019   61.498118
2025-09-07 00:00:00+00:00  59.346924  60.923706  61.562250   61.298348
2025-09-14 00:00:00+00:00  59.204437  60.978707  61.335004   61.149277
2025-09-21 00:00:00+00:00  58.834077  61.003371  61.147675   60.875730
2025-09-28 00:00:00+00:00  58.796906  61.126085  61.215474   60.753606
2025-10-05 00:00:00+00:00  58.595791  60.888102  60.956598   60.470725
2025-10-12 00:00:00+00:00  58.315769  60.955883  61.134025   60.688758
2025-10-19 00:00:00+00:00  58.641778  61.679482  61.552660   60.882372
2025-10-26 00:00:00+00:00  58.707109  61.815298  61.379292   60.717402
2025-11-02 00:00:00+00:00  58.560646  61.727316  61.347370   60.980325
2025-11-09 00:00:00+00:00  58.861826  62.261251  61.512313   61.186645
2025-11-16 00:00:00+00:00  59.065243  62.409618  61.707197   60.776436
```

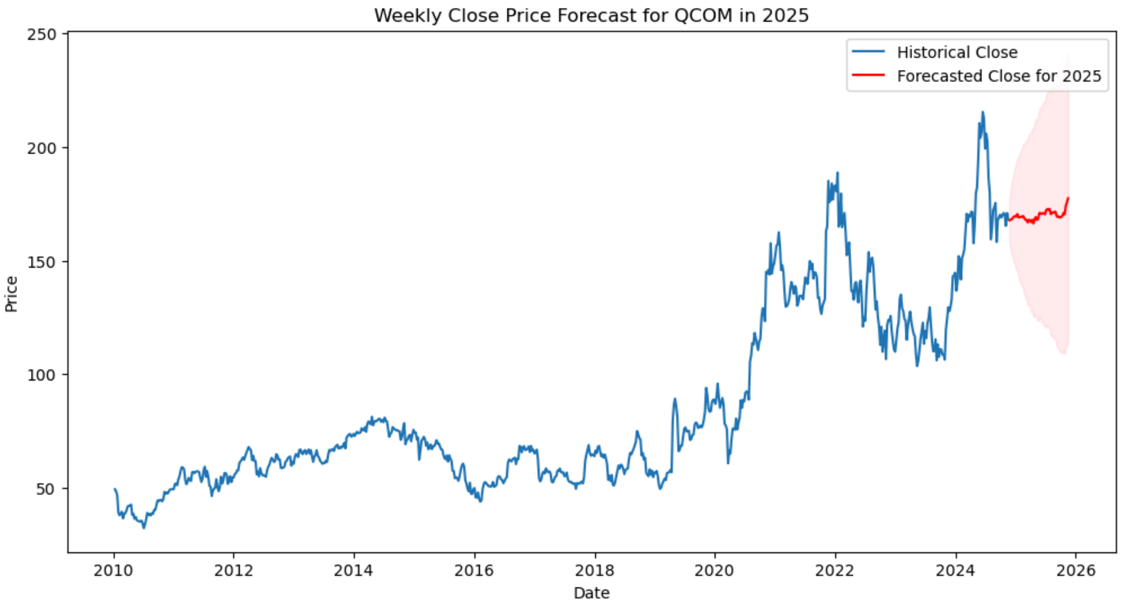
In [22]:

```
import yfinance as yf
import pandas as pd
from statsmodels.tsa.statespace.sarimax import SARIMAX
import matplotlib.pyplot as plt
import datetime

# Define the stock symbol and download historical data
symbol = "QCOM"
start_date = "2010-01-01"
end_date = datetime.datetime.now()

# Download data
stock_data = yf.download(symbol, start=start_date, end=end_date)

# Step 1: Resample to weekly frequency using MultiIndex column names
weekly_data = stock_data.resample('W').agg({
    ('Open', 'QCOM'): 'first',     # First observation of the week for Open
    ('High', 'QCOM'): 'max',       # Maximum of the week for High
    ('Low', 'QCOM'): 'min',        # Minimum of the week for Low
    ('Close', 'QCOM'): 'last'      # Last observation of the week for Close
})

# Drop any rows with NaN values to ensure continuity
weekly_data = weekly_data.dropna()

# Define a function to forecast using SARIMA for each column
def forecast_sarima(series, steps=52):
    # Fit SARIMA model with weekly seasonality
    model = SARIMAX(series, order=(1, 1, 1), seasonal_order=(1, 1, 1, 52))
    model_fit = model.fit(disp=False)

    # Forecast for the desired number of future steps
    forecast = model_fit.get_forecast(steps=steps)
    forecast_values = forecast.predicted_mean
    confidence_intervals = forecast.conf_int()

    # Adjust index for forecast to start after the last historical date
    forecast_index = pd.date_range(start=series.index[-1] + pd.Timedelta(we
    forecast_values.index = forecast_index
    confidence_intervals.index = forecast_index

    return forecast_values, confidence_intervals

# Forecast for 2025 (52 weeks) using MultiIndex column names
open_forecast, open_conf = forecast_sarima(weekly_data[('Open', 'QCOM')], s
high_forecast, high_conf = forecast_sarima(weekly_data[('High', 'QCOM')], s
low_forecast, low_conf = forecast_sarima(weekly_data[('Low', 'QCOM')], step
close_forecast, close_conf = forecast_sarima(weekly_data[('Close', 'QCOM')]

# Combine forecasts into a DataFrame for easy viewing
forecast_2025 = pd.DataFrame({
    'Open_QCOM': open_forecast,
    'High_QCOM': high_forecast,
    'Low_QCOM': low_forecast,
    'Close_QCOM': close_forecast
})

# Display the forecast
print(forecast_2025)
```

```
[*********************100%***********************]  1 of 1 completed

                           Open_QCOM   High_QCOM    Low_QCOM  Close_QCOM
2024-11-24 00:00:00+00:00  171.901544  168.782370  167.399697  167.698681
2024-12-01 00:00:00+00:00  172.045754  169.537793  166.645990  167.912981
2024-12-08 00:00:00+00:00  172.521278  170.575780  167.162287  168.123173
2024-12-15 00:00:00+00:00  172.695789  173.475285  168.760933  168.715892
2024-12-22 00:00:00+00:00  172.220276  173.001608  169.202342  169.305549
2024-12-29 00:00:00+00:00  173.615829  173.805270  170.711917  169.525038
2025-01-05 00:00:00+00:00  173.529402  173.289244  168.990173  169.302145
2025-01-12 00:00:00+00:00  173.689293  174.171968  169.458154  170.455842
2025-01-19 00:00:00+00:00  174.690480  176.502277  168.274981  168.949829
2025-01-26 00:00:00+00:00  173.513285  177.047010  169.991284  169.057993
2025-02-02 00:00:00+00:00  173.730898  176.404305  168.764560  169.126624
2025-02-09 00:00:00+00:00  173.744202  176.290346  169.436176  169.351900
2025-02-16 00:00:00+00:00  173.644491  176.959160  170.687764  169.564921
2025-02-23 00:00:00+00:00  172.786853  176.028662  168.435846  168.664821
2025-03-02 00:00:00+00:00  173.014664  177.951908  168.840148  167.909992
2025-03-09 00:00:00+00:00  171.958563  181.240682  167.535474  167.849815
2025-03-16 00:00:00+00:00  170.497083  179.349999  167.432467  166.747770
2025-03-23 00:00:00+00:00  170.986406  179.190668  168.602324  168.007613
2025-03-30 00:00:00+00:00  171.485118  178.391545  168.874392  167.556226
2025-04-06 00:00:00+00:00  171.740452  180.579368  168.901913  166.851855
2025-04-13 00:00:00+00:00  170.279164  180.528901  168.703722  167.941258
2025-04-20 00:00:00+00:00  171.711267  179.578575  167.246763  166.306889
2025-04-27 00:00:00+00:00  169.707472  177.927631  168.056301  168.257387
2025-05-04 00:00:00+00:00  171.786932  183.056484  168.490577  169.139032
2025-05-11 00:00:00+00:00  172.524998  182.342570  169.162606  167.878756
2025-05-18 00:00:00+00:00  171.176644  185.709306  169.953175  168.223543
2025-05-25 00:00:00+00:00  171.482794  189.906534  171.478385  170.880546
2025-06-01 00:00:00+00:00  176.082473  192.760128  174.223813  170.844283
2025-06-08 00:00:00+00:00  175.986844  191.223914  174.722504  170.491544
2025-06-15 00:00:00+00:00  174.497107  192.801435  174.357051  170.876850
2025-06-22 00:00:00+00:00  175.516076  196.090421  174.976954  170.716980
2025-06-29 00:00:00+00:00  174.749978  189.681182  173.211974  170.334035
2025-07-06 00:00:00+00:00  174.464326  189.959680  173.909475  172.179968
2025-07-13 00:00:00+00:00  176.501273  192.476180  175.646347  172.671788
2025-07-20 00:00:00+00:00  177.345522  192.987520  174.408064  172.283507
2025-07-27 00:00:00+00:00  177.278969  189.242844  172.819079  172.725446
2025-08-03 00:00:00+00:00  177.661905  185.172714  171.781406  170.378466
2025-08-10 00:00:00+00:00  174.251580  179.556400  170.221961  171.178353
2025-08-17 00:00:00+00:00  174.847775  182.128161  171.465703  170.963569
2025-08-24 00:00:00+00:00  174.915519  182.181429  173.039556  171.382184
2025-08-31 00:00:00+00:00  175.330741  182.994069  172.665198  171.481022
2025-09-07 00:00:00+00:00  174.763551  181.362573  170.459016  169.673182
2025-09-14 00:00:00+00:00  174.205949  180.109315  170.164369  169.100758
2025-09-21 00:00:00+00:00  171.840589  181.686206  169.886654  169.405104
2025-09-28 00:00:00+00:00  172.603079  181.369125  169.952460  168.920273
2025-10-05 00:00:00+00:00  172.420334  180.627748  169.509908  169.192488
2025-10-12 00:00:00+00:00  172.525782  179.885474  169.516821  169.577124
2025-10-19 00:00:00+00:00  173.386986  183.563072  171.635500  170.911016
2025-10-26 00:00:00+00:00  173.940254  182.000693  169.911479  170.278749
2025-11-02 00:00:00+00:00  173.678411  185.987914  170.018853  174.003173
2025-11-09 00:00:00+00:00  178.153938  187.652964  175.068116  175.582320
2025-11-16 00:00:00+00:00  179.898456  185.857581  177.749349  177.397611
```

In [24]:

```
# Plot the forecast for Close price as an example
plt.figure(figsize=(12, 6))
plt.plot(weekly_data[('Close', 'QCOM')], label='Historical Close')
plt.plot(close_forecast.index, close_forecast, color='red', label='Forecas
plt.fill_between(close_forecast.index, close_conf.iloc[:, 0], close_conf.il
plt.legend()
plt.title(f"Weekly Close Price Forecast for {symbol} in 2025")
plt.xlabel("Date")
plt.ylabel("Price")
plt.show()

# Display forecasted values
print("Forecasted weekly stock prices for 2025:")
print(forecast_2025)
```



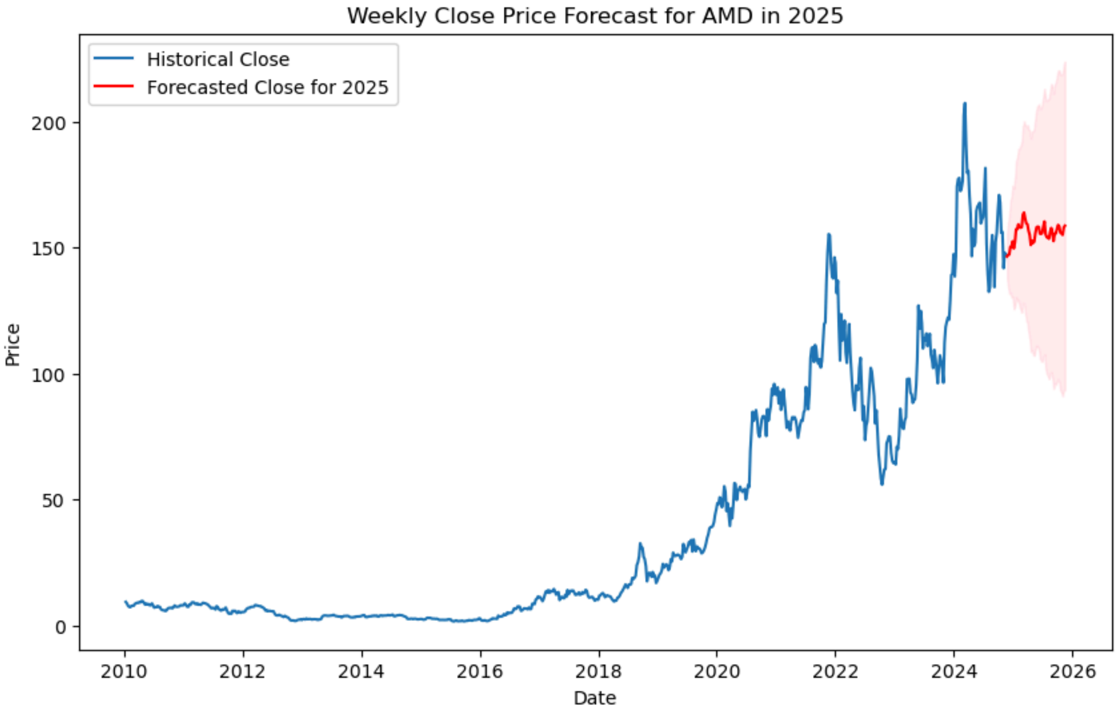
```
Forecasted weekly stock prices for 2025:
                            Open_QCOM   High_QCOM    Low_QCOM  Close_QCOM
2024-11-24 00:00:00+00:00  171.901544  168.782370  167.399697  167.698681
2024-12-01 00:00:00+00:00  172.045754  169.537793  166.645990  167.912981
2024-12-08 00:00:00+00:00  172.521278  170.575780  167.162287  168.123173
2024-12-15 00:00:00+00:00  172.695789  173.475285  168.760933  168.715892
2024-12-22 00:00:00+00:00  172.220276  173.001608  169.202342  169.305549
2024-12-29 00:00:00+00:00  173.615829  173.805270  170.711917  169.525038
2025-01-05 00:00:00+00:00  173.529402  173.289244  168.990173  169.302145
2025-01-12 00:00:00+00:00  173.689293  174.171968  169.458154  170.455842
2025-01-19 00:00:00+00:00  174.690480  176.502277  168.274981  168.949829
2025-01-26 00:00:00+00:00  173.513285  177.047010  169.991284  169.057993
2025-02-02 00:00:00+00:00  173.730898  176.404305  168.764560  169.126624
2025-02-09 00:00:00+00:00  173.744202  176.290346  169.436176  169.351900
2025-02-16 00:00:00+00:00  173.644491  176.959160  170.687764  169.564921
2025-02-23 00:00:00+00:00  172.786853  176.028662  168.435846  168.664821
2025-03-02 00:00:00+00:00  173.014664  177.951908  168.840148  167.909992
2025-03-09 00:00:00+00:00  171.958563  181.240682  167.535474  167.849815
2025-03-16 00:00:00+00:00  170.497083  179.349999  167.432467  166.747770
2025-03-23 00:00:00+00:00  170.986406  179.190668  168.602324  168.007613
2025-03-30 00:00:00+00:00  171.485118  178.391545  168.874392  167.556226
2025-04-06 00:00:00+00:00  171.740452  180.579368  168.901913  166.851855
2025-04-13 00:00:00+00:00  170.279164  180.528901  168.703722  167.941258
2025-04-20 00:00:00+00:00  171.711267  179.578575  167.246763  166.306889
2025-04-27 00:00:00+00:00  169.707472  177.927631  168.056301  168.257387
2025-05-04 00:00:00+00:00  171.786932  183.056484  168.490577  169.139032
2025-05-11 00:00:00+00:00  172.524998  182.342570  169.162606  167.878756
2025-05-18 00:00:00+00:00  171.176644  185.709306  169.953175  168.223543
2025-05-25 00:00:00+00:00  171.482794  189.906534  171.478385  170.880546
2025-06-01 00:00:00+00:00  176.082473  192.760128  174.223813  170.844283
2025-06-08 00:00:00+00:00  175.986844  191.223914  174.722504  170.491544
2025-06-15 00:00:00+00:00  174.497107  192.801435  174.357051  170.876850
2025-06-22 00:00:00+00:00  175.516076  196.090421  174.976954  170.716980
2025-06-29 00:00:00+00:00  174.749978  189.681182  173.211974  170.334035
2025-07-06 00:00:00+00:00  174.464326  189.959680  173.909475  172.179968
2025-07-13 00:00:00+00:00  176.501273  192.476180  175.646347  172.671788
2025-07-20 00:00:00+00:00  177.345522  192.987520  174.408064  172.283507
2025-07-27 00:00:00+00:00  177.278969  189.242844  172.819079  172.725446
2025-08-03 00:00:00+00:00  177.661905  185.172714  171.781406  170.378466
2025-08-10 00:00:00+00:00  174.251580  179.556400  170.221961  171.178353
2025-08-17 00:00:00+00:00  174.847775  182.128161  171.465703  170.963569
2025-08-24 00:00:00+00:00  174.915519  182.181429  173.039556  171.382184
2025-08-31 00:00:00+00:00  175.330741  182.994069  172.665198  171.481022
2025-09-07 00:00:00+00:00  174.763551  181.362573  170.459016  169.673182
2025-09-14 00:00:00+00:00  174.205949  180.109315  170.164369  169.100758
2025-09-21 00:00:00+00:00  171.840589  181.686206  169.886654  169.405104
2025-09-28 00:00:00+00:00  172.603079  181.369125  169.952460  168.920273
2025-10-05 00:00:00+00:00  172.420334  180.627748  169.509908  169.192488
2025-10-12 00:00:00+00:00  172.525782  179.885474  169.516821  169.577124
2025-10-19 00:00:00+00:00  173.386986  183.563072  171.635500  170.911016
2025-10-26 00:00:00+00:00  173.940254  182.000693  169.911479  170.278749
2025-11-02 00:00:00+00:00  173.678411  185.987914  170.018853  174.003173
2025-11-09 00:00:00+00:00  178.153938  187.652964  175.068116  175.582320
2025-11-16 00:00:00+00:00  179.898456  185.857581  177.749349  177.397611
```

In [27]:

```
import yfinance as yf
import pandas as pd
from statsmodels.tsa.statespace.sarimax import SARIMAX
import matplotlib.pyplot as plt
import datetime

# Define the stock symbol and download historical data
symbol = "AMD"
start_date = "2010-01-01"
end_date = datetime.datetime.now()

# Download data
stock_data = yf.download(symbol, start=start_date, end=end_date)

# Step 1: Resample to weekly frequency using MultiIndex column names
weekly_data = stock_data.resample('W').agg({
    ('Open', 'AMD'): 'first',     # First observation of the week for Open
    ('High', 'AMD'): 'max',       # Maximum of the week for High
    ('Low', 'AMD'): 'min',        # Minimum of the week for Low
    ('Close', 'AMD'): 'last'      # Last observation of the week for Close
})

# Drop any rows with NaN values to ensure continuity
weekly_data = weekly_data.dropna()

# Define a function to forecast using SARIMA for each column
def forecast_sarima(series, steps=52):
    # Fit SARIMA model with weekly seasonality
    model = SARIMAX(series, order=(1, 1, 1), seasonal_order=(1, 1, 1, 52))
    model_fit = model.fit(disp=False)

    # Forecast for the desired number of future steps
    forecast = model_fit.get_forecast(steps=steps)
    forecast_values = forecast.predicted_mean
    confidence_intervals = forecast.conf_int()

    # Adjust index for forecast to start after the last historical date
    forecast_index = pd.date_range(start=series.index[-1] + pd.Timedelta(we
    forecast_values.index = forecast_index
    confidence_intervals.index = forecast_index

    return forecast_values, confidence_intervals

# Forecast for 2025 (52 weeks) using MultiIndex column names
open_forecast, open_conf = forecast_sarima(weekly_data[('Open', 'AMD')], st
high_forecast, high_conf = forecast_sarima(weekly_data[('High', 'AMD')], st
low_forecast, low_conf = forecast_sarima(weekly_data[('Low', 'AMD')], steps
close_forecast, close_conf = forecast_sarima(weekly_data[('Close', 'AMD')]

# Combine forecasts into a DataFrame for easy viewing
forecast_2025 = pd.DataFrame({
    'Open_AMD': open_forecast,
    'High_AMD': high_forecast,
    'Low_AMD': low_forecast,
    'Close_AMD': close_forecast
})

# Display the forecast
print(forecast_2025)
```

```
[*********************100%***********************]  1 of 1 completed

                             Open_AMD    High_AMD     Low_AMD   Close_AMD
2024-11-24 00:00:00+00:00  148.618051  150.254286  146.719630  146.523049
2024-12-01 00:00:00+00:00  149.273350  151.507115  146.642048  147.153240
2024-12-08 00:00:00+00:00  148.222225  150.705194  145.228564  147.288336
2024-12-15 00:00:00+00:00  149.947645  153.719224  147.933649  150.329370
2024-12-22 00:00:00+00:00  151.509494  153.390875  149.937990  150.370289
2024-12-29 00:00:00+00:00  152.423295  155.686642  152.156643  152.528444
2025-01-05 00:00:00+00:00  153.379281  154.313957  149.204345  149.679078
2025-01-12 00:00:00+00:00  151.781064  155.513649  150.875195  152.110765
2025-01-19 00:00:00+00:00  154.176489  160.692076  153.178747  157.329275
2025-01-26 00:00:00+00:00  158.054179  162.112841  155.095429  157.282901
2025-02-02 00:00:00+00:00  158.394914  162.574765  155.748592  159.265905
2025-02-09 00:00:00+00:00  160.255272  163.638744  158.775358  158.019253
2025-02-16 00:00:00+00:00  159.089206  162.879808  158.169294  157.931249
2025-02-23 00:00:00+00:00  158.018114  162.351329  155.033994  158.123694
2025-03-02 00:00:00+00:00  159.595464  166.772678  157.987854  163.096874
2025-03-09 00:00:00+00:00  163.505032  171.393878  164.658539  163.941177
2025-03-16 00:00:00+00:00  162.805606  167.462064  160.608582  161.519183
2025-03-23 00:00:00+00:00  162.484083  165.928425  160.943653  159.653486
2025-03-30 00:00:00+00:00  159.123731  163.821304  159.948112  159.269667
2025-04-06 00:00:00+00:00  160.164867  163.378768  158.042952  156.657194
2025-04-13 00:00:00+00:00  157.128266  159.145422  156.302488  155.088542
2025-04-20 00:00:00+00:00  156.487343  157.080623  151.279086  151.046476
2025-04-27 00:00:00+00:00  153.375808  155.240094  150.396040  152.754702
2025-05-04 00:00:00+00:00  154.626966  155.962907  148.548376  151.904410
2025-05-11 00:00:00+00:00  153.881827  155.486525  151.800315  152.374940
2025-05-18 00:00:00+00:00  153.887898  159.841300  152.632448  155.748259
2025-05-25 00:00:00+00:00  157.251024  162.680396  156.735089  158.056806
2025-06-01 00:00:00+00:00  160.278921  164.394659  160.641669  158.353037
2025-06-08 00:00:00+00:00  160.508373  163.671301  159.891416  158.221987
2025-06-15 00:00:00+00:00  158.446716  161.315972  158.146373  155.585838
2025-06-22 00:00:00+00:00  156.896716  160.026186  155.678476  155.444139
2025-06-29 00:00:00+00:00  156.853020  159.816596  155.904733  155.864077
2025-07-06 00:00:00+00:00  157.263026  161.773764  156.001454  158.382867
2025-07-13 00:00:00+00:00  160.211992  165.150024  161.528571  160.376045
2025-07-20 00:00:00+00:00  162.007904  166.716973  156.159265  155.343901
2025-07-27 00:00:00+00:00  157.596907  161.270280  153.910877  154.150022
2025-08-03 00:00:00+00:00  156.658010  163.568755  154.892464  153.974414
2025-08-10 00:00:00+00:00  154.381766  158.952693  152.604806  153.557434
2025-08-17 00:00:00+00:00  155.957781  160.626175  155.210225  156.189650
2025-08-24 00:00:00+00:00  158.277658  163.515116  159.197028  157.812569
2025-08-31 00:00:00+00:00  159.627871  162.750063  157.145195  155.940696
2025-09-07 00:00:00+00:00  157.377782  158.670351  153.902734  152.631752
2025-09-14 00:00:00+00:00  155.410896  159.451701  153.691520  155.486124
2025-09-21 00:00:00+00:00  157.046893  159.537071  155.244230  155.645948
2025-09-28 00:00:00+00:00  157.632943  161.443432  156.514796  157.417667
2025-10-05 00:00:00+00:00  158.835432  162.475123  157.855625  159.014617
2025-10-12 00:00:00+00:00  160.434358  162.920401  159.693704  158.467212
2025-10-19 00:00:00+00:00  160.240470  161.894324  157.522170  155.928369
2025-10-26 00:00:00+00:00  157.832089  159.496726  155.679715  155.677256
2025-11-02 00:00:00+00:00  158.018826  164.081829  153.683078  155.070749
2025-11-09 00:00:00+00:00  157.353869  162.920534  157.627387  157.987425
2025-11-16 00:00:00+00:00  160.187365  163.495492  161.753596  158.697243
```

In [29]:
```
# Plot the forecast for Close price as an example
plt.figure(figsize=(10, 6))
plt.plot(weekly_data[('Close', 'AMD')], label='Historical Close')
plt.plot(close_forecast.index, close_forecast, color='red', label='Forecas
plt.fill_between(close_forecast.index, close_conf.iloc[:, 0], close_conf.i
plt.legend()
plt.title(f"Weekly Close Price Forecast for {symbol} in 2025")
plt.xlabel("Date")
plt.ylabel("Price")
plt.show()

# Display forecasted values
print("Forecasted weekly stock prices for 2025:")
print(forecast_2025)
```


Weekly Close Price Forecast for AMD in 2025
Historical Close
Forecasted Close for 2025
Price
Date