In [1]:
```
#Import Libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt

from sklearn.linear_model import LinearRegression
```

In [2]:
```
#Loading Dataset

gold_data = pd.read_excel('no_missing_values.xlsx')
gold_data['date'] = pd.to_datetime(gold_data['date'])
gold_data.set_index('date', inplace=True)
```

In [5]:
```
gold_data.shape
```

Out[5]:
```
(5258, 9)
```

In [27]:
```
gold_data.tail(10)
```

Out[27]:

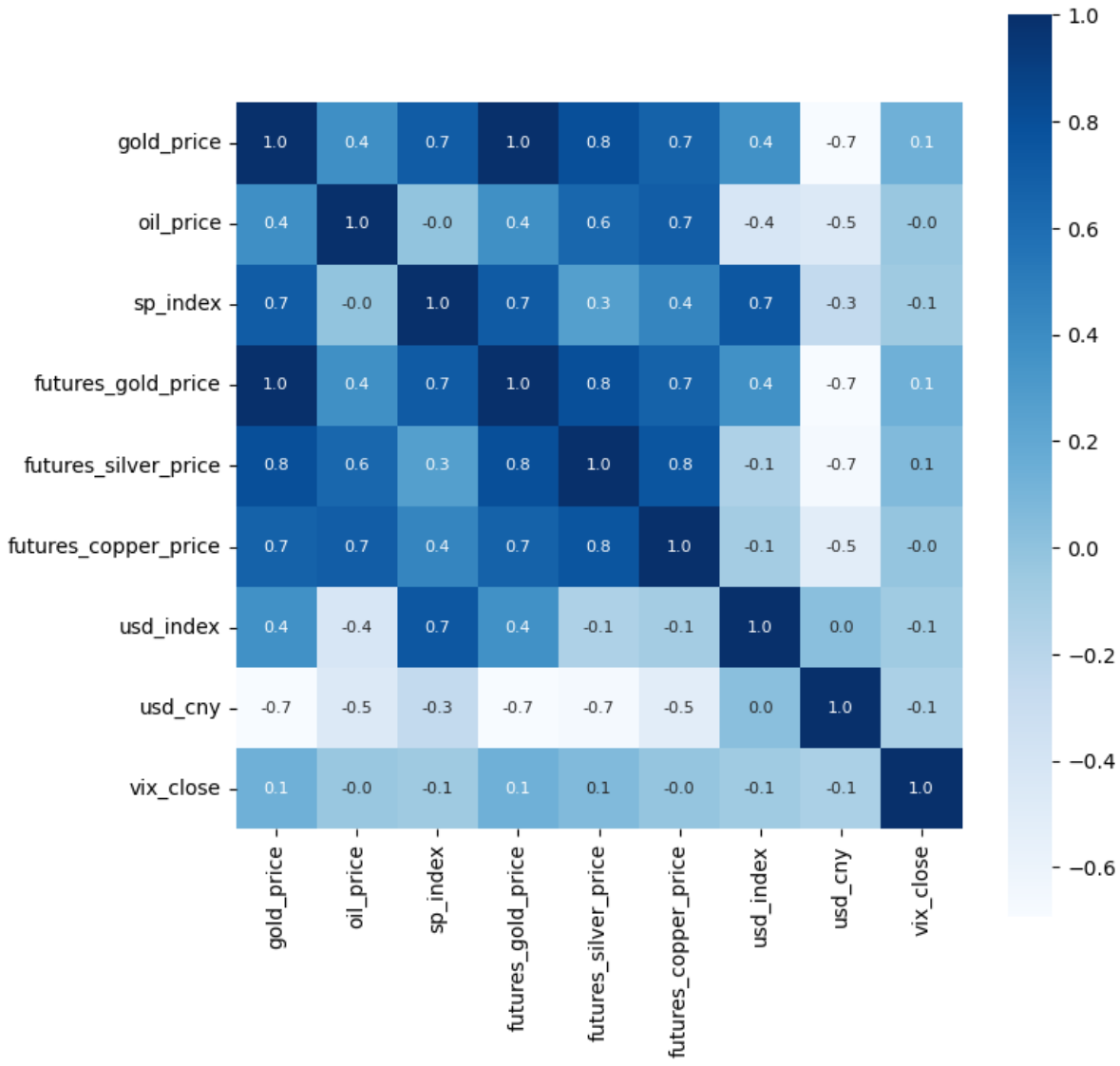
date	gold_price	oil_price	sp_index	futures_gold_price	futures_silver_price	futures_copper_price
2023-12-26	2069.40	80.97	4774.75	2060.40	24.3960	3.90200
2023-12-27	2069.40	80.97	4781.58	2083.40	24.5700	3.95600
2023-12-28	2078.40	79.04	4783.35	2083.50	24.3720	3.92450
2023-12-29	2078.40	77.69	4769.83	2071.80	24.0250	3.89150
2023-12-31	2078.40	77.69	4769.83	2072.35	24.0375	3.89515
2024-01-01	2078.40	76.24	4742.83	2072.90	24.0500	3.89880
2024-01-02	2067.55	76.24	4742.83	2073.40	23.9530	3.88050
2024-01-03	2042.10	77.18	4704.81	2042.80	23.1570	3.86150
2024-01-04	2039.55	75.79	4688.68	2050.00	23.1870	3.84400
2024-01-05	2056.35	78.31	4697.24	2049.80	23.3150	3.80600

In [7]:
```
gold_data.isnull().sum()
```

Out[7]:
```
gold_price              0
oil_price               0
sp_index                0
futures_gold_price      0
futures_silver_price    0
futures_copper_price    0
usd_index               0
usd_cny                 0
vix_close               0
dtype: int64
```

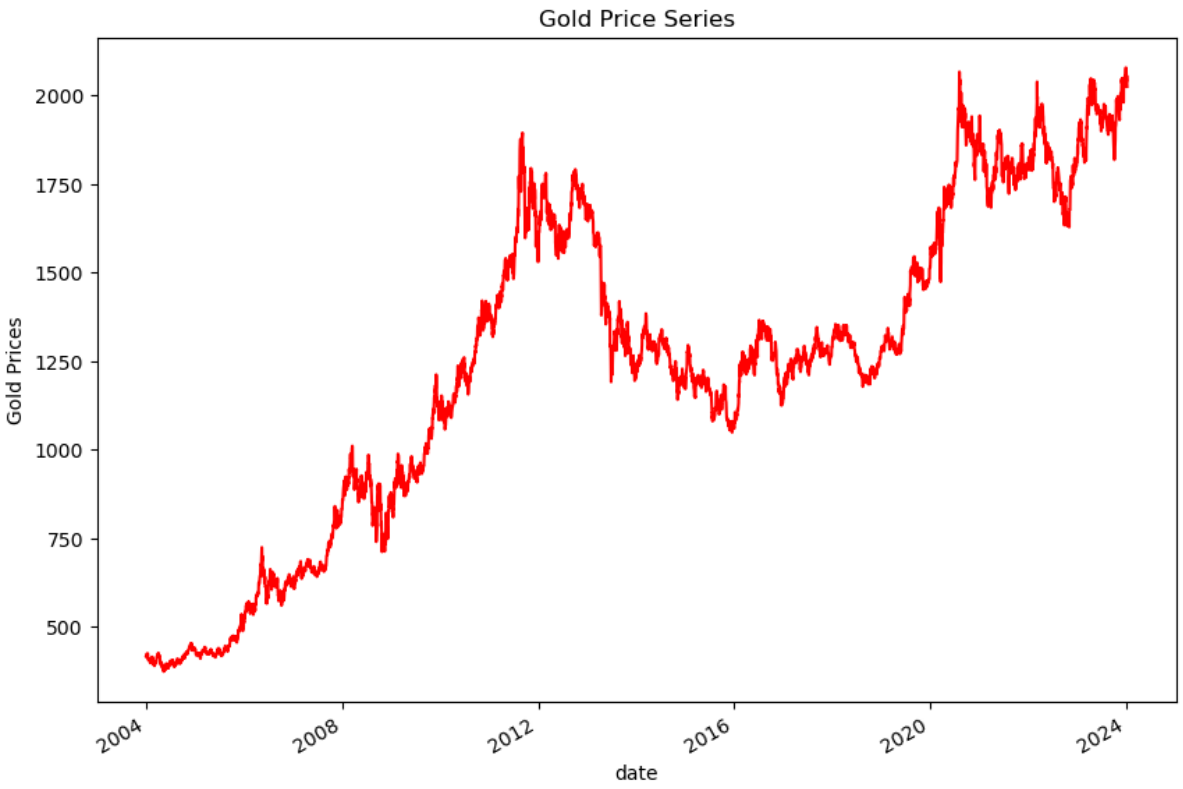
In [10]:
```
# Plot Correlation Heat Map

plt.figure(figsize = (8,8))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f',annot=True,
            annot_kws={'size':8}, cmap='Blues')
```

Out[10]:
```
<Axes: >
```



In [11]:
```
Df = gold_data
```

In [13]:
```
# Plot the historical price of gold
Df.gold_price.plot(figsize=(10, 7),color='r')
plt.ylabel("Gold Prices")
plt.title("Gold Price Series")
plt.show()
```



In [14]:
```
#Create rolling moving averages
Df['MA3'] = Df['gold_price'].rolling(window=3).mean()
Df['MA9'] = Df['gold_price'].rolling(window=9).mean()
Df['next_day_price'] = Df['gold_price'].shift(-1)

#Drop rows with empty values
Df = Df.dropna()

#Define explanatory and target variables
X = Df[['MA3', 'MA9','oil_price','sp_index','futures_silver_price',
       'futures_copper_price','usd_index','usd_cny','vix_close']]
Y = Df['next_day_price']
```

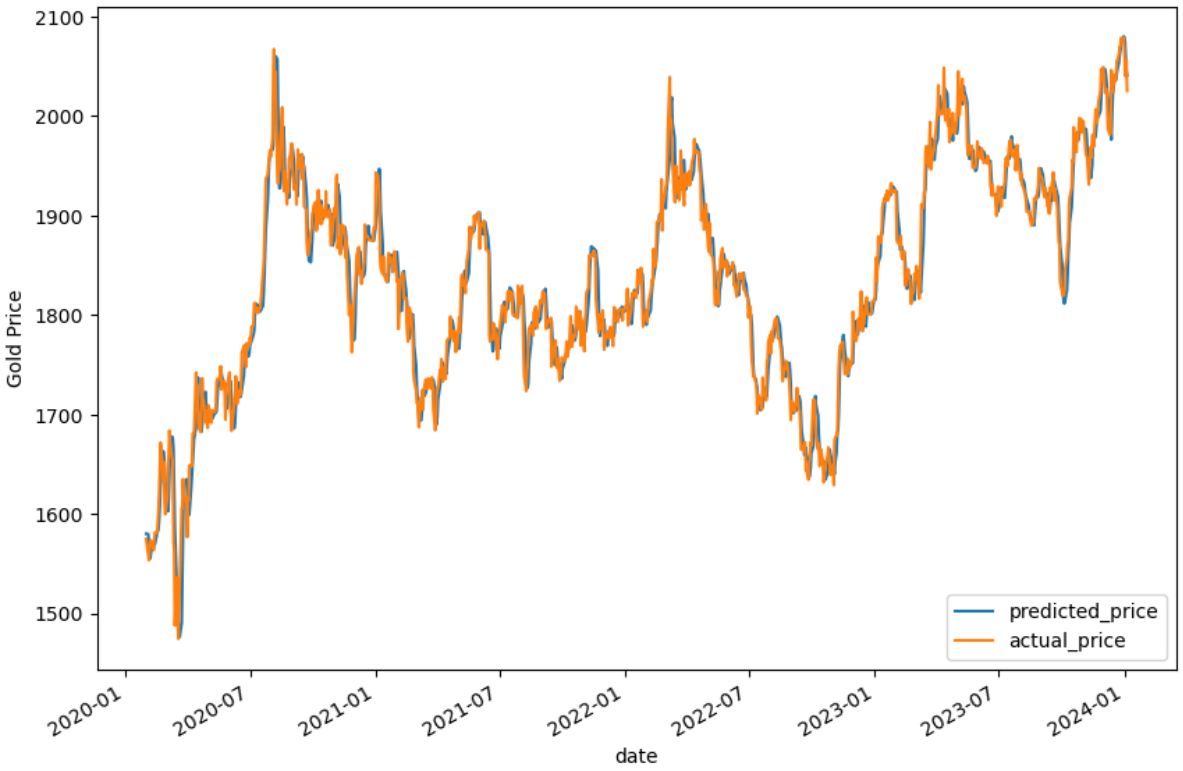
In [16]:
```
# Split the data into train and test dataset
t = .8
t = int(t*len(Df))

# Train dataset
X_train = X[:t]
Y_train = Y[:t]

# Test dataset
X_test = X[t:]
Y_test = Y[t:]
```

In [17]:
```
# Create a linear regression equation
linear = LinearRegression().fit(X_train, Y_train)
print("Linear Regression model")
print("Gold ETF Price (y) = %.2f * 3 Days Moving Average (x1) \
+ %.2f * 9 Days Moving Average (x2) \
+ %.2f * oil price \
+ %.2f * sp index \
+ %.2f * futures silver price \
+ %.2f * futures copper price \
+ %.2f * USD index \
+ %.2f * USD-CNY exchange rate \
+ %.2f * VIX close \
```

```
+ %.2f (constant)" % (linear.coef_[0], linear.coef_[1],linear.coef_[2],
                      linear.coef_[3],linear.coef_[4],linear.coef_[5],
                      linear.coef_[6],linear.coef_[7],linear.coef_[8],
                      linear.intercept_))
```

```
Linear Regression model
Gold ETF Price (y) = 1.18 * 3 Days Moving Average (x1) + -0.19 * 9 Days Moving Ave
rage (x2) + -0.05 * oil price + 0.00 * sp index + 0.69 * futures silver price + 0.
17 * futures copper price + -0.05 * USD index + -2.07 * USD-CNY exchange rate + 0.
10 * VIX close + 20.09 (constant)
```

In [26]:
```
# Predicting Gold prices
predicted_price = linear.predict(X_test)
predicted_price = pd.DataFrame(
    predicted_price, index=Y_test.index, columns=['price'])
predicted_price.plot(figsize=(10, 7))
Y_test.plot()
plt.legend(['predicted_price', 'actual_price'])
plt.ylabel("Gold Price")
plt.show()
```

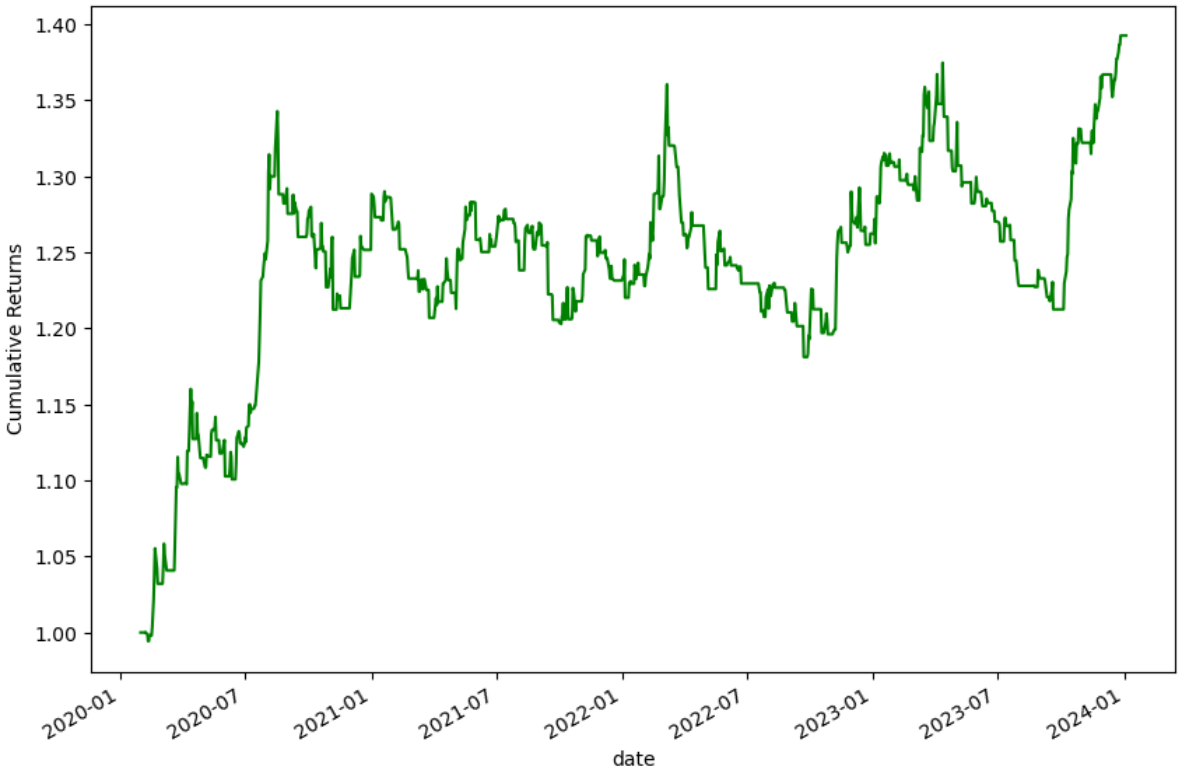


In [19]:
```
# R square error score
r2_score = linear.score(X[t:], Y[t:])*100
float("{0:.2f}".format(r2_score))
```

Out[19]:
```
96.07
```

In [20]:
```
#Plot cumulative returns
gold = pd.DataFrame()

gold['price'] = Df[t:]['gold_price']
gold['predicted_price_next_day'] = predicted_price
gold['actual_price_next_day'] = Y_test
gold['gold_returns'] = gold['price'].pct_change().shift(-1)

gold['signal'] = np.where(gold.predicted_price_next_day.shift(1) < gold.predicted_p

gold['strategy_returns'] = gold.signal * gold['gold_returns']
((gold['strategy_returns']+1).cumprod()).plot(figsize=(10,7),color='g')
```

```
plt.ylabel('Cumulative Returns')
plt.show()
```



In [21]:
```
# Calculate sharpe ratio
sharpe = gold['strategy_returns'].mean()/gold['strategy_returns'].std()*(252**0.5)
'Sharpe Ratio %.2f' % (sharpe)
```

Out[21]:
```
'Sharpe Ratio 0.79'
```

In [22]:
```
#Create a daily 'signal' to predict whether gold price will go up or down

# Get the data
data = gold_data
data.drop(data.tail(1).index,inplace=True)
data['S_3'] = data['gold_price'].rolling(window=3).mean()
data['S_9'] = data['gold_price'].rolling(window=9).mean()
data = data.dropna()

# Forecast the price
data['predicted_gold_price'] = linear.predict(data[['MA3', 'MA9', 'oil_price', 'sp_
       'futures_copper_price', 'usd_index', 'usd_cny', 'vix_close']])
data['signal'] = np.where(data.predicted_gold_price.shift(1) < data.predicted_gold_

# Print the signal
data.tail(1)[['signal','predicted_gold_price']].T
```

```
C:\Users\kesha\AppData\Local\Temp\ipykernel_10876\2408819926.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['predicted_gold_price'] = linear.predict(data[['MA3', 'MA9', 'oil_price', 'sp_index', 'futures_silver_price',
C:\Users\kesha\AppData\Local\Temp\ipykernel_10876\2408819926.py:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['signal'] = np.where(data.predicted_gold_price.shift(1) < data.predicted_gold_price,"Up","Down")
```

Out[22]:

date	**2024-01-05**
signal	Down
predicted_gold_price	2041.078942

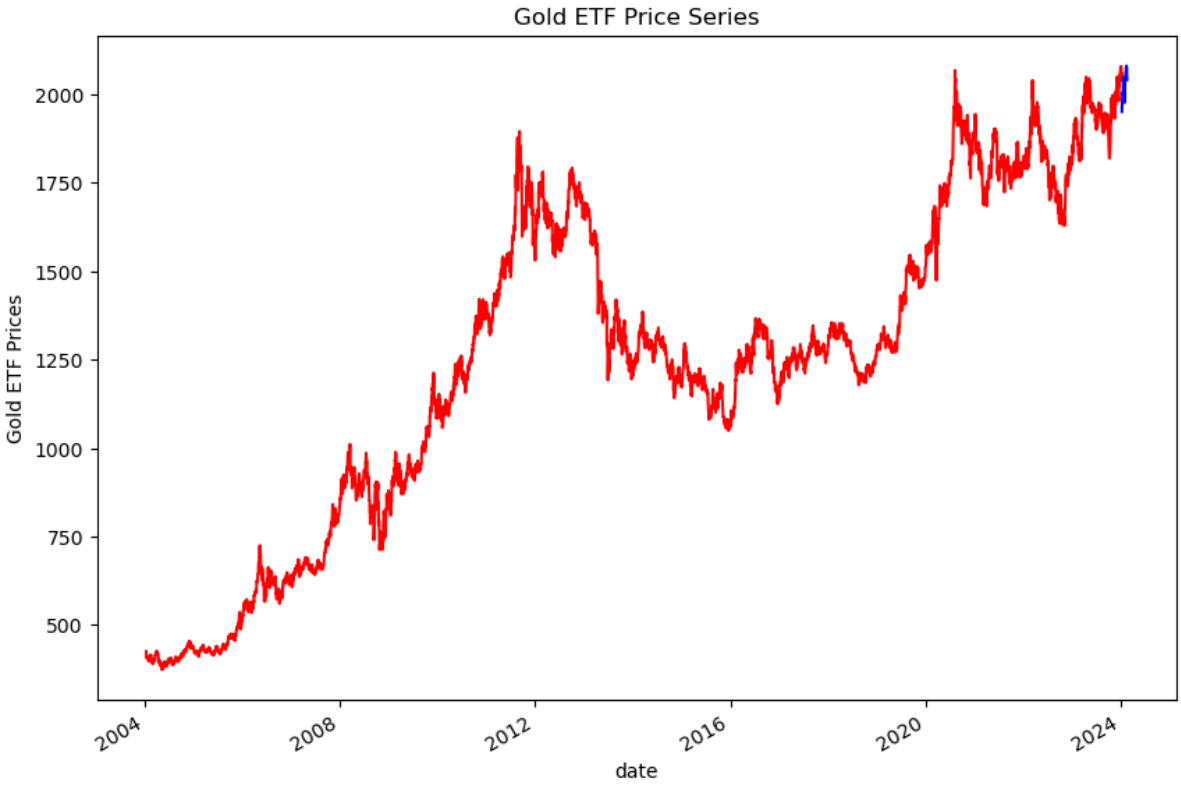
In [25]:

```python
# copy the gold dataframe
data = gold_data.copy()  # Make a copy to avoid modifying the original DataFrame
data.index = pd.to_datetime(data.index)

# Create a DataFrame with the next 30 days
future_dates = pd.date_range(start=data.index.max() + pd.DateOffset(days=1), perioc
future_data = pd.DataFrame(index=future_dates)

# Concatenate the original data with the future dates
extended_data = pd.concat([data, future_data])

# Calculate rolling means and drop NaN values
extended_data['S_3'] = extended_data['gold_price'].rolling(window=3).mean()
extended_data['S_9'] = extended_data['gold_price'].rolling(window=9).mean()
extended_data = extended_data.dropna()

# Extract features for prediction
features_for_prediction = extended_data[['MA3', 'MA9', 'oil_price', 'sp_index', 'fu
                                         'futures_copper_price', 'usd_index', 'usd_


linear = LinearRegression()
target = 'gold_price'

# Train the model on the historical data
linear.fit(extended_data[features_for_prediction.columns], extended_data[target])

# Predict the gold prices for the next 30 days
predicted_gold_prices = linear.predict(features_for_prediction.tail(30))

# Create a DataFrame for the predictions
predictions_df = pd.DataFrame(index=future_dates, columns=['predicted_gold_price'])
predictions_df['predicted_gold_price'] = predicted_gold_prices

# Print the predictions for the next 30 days
print("Predictions for the next 30 days:")
print(predictions_df)
```

```
Predictions for the next 30 days:
            predicted_gold_price
2024-01-06           2033.547921
2024-01-07           2041.130633
2024-01-08           2047.335914
2024-01-09           2046.878697
2024-01-10           2041.332348
2024-01-11           2032.851558
2024-01-12           2023.344696
2024-01-13           2017.072623
2024-01-14           2002.052559
2024-01-15           1985.151410
2024-01-16           1976.147023
2024-01-17           2000.428313
2024-01-18           2021.314279
2024-01-19           2037.883130
2024-01-20           2035.720970
2024-01-21           2036.893330
2024-01-22           2043.225614
2024-01-23           2047.307482
2024-01-24           2053.585298
2024-01-25           2060.059181
2024-01-26           2064.981784
2024-01-27           2069.648423
2024-01-28           2074.739991
2024-01-29           2077.363874
2024-01-30           2079.979307
2024-01-31           2079.131075
2024-02-01           2074.510749
2024-02-02           2060.009934
2024-02-03           2045.003371
2024-02-04           2040.857831
```

In [24]:

```python
# Plot the historic and predicted price of gold
Df.gold_price.plot(figsize=(10, 7),color='r')
predictions_df.predicted_gold_price.plot(color = 'b')
plt.ylabel("Gold ETF Prices")
plt.title("Gold ETF Price Series")
plt.show()
```


Gold ETF Price Series
Gold ETF Prices
2000
1750
1500
1250
1000
750
500
2004
2008
2012
2016
2020
2024
date

In [ ]: