Costco stock price analysis

The Costco stock price is interesting to look at. The stock price of Costco has exprienced ups and downs since Covid. A quantitative analysis will be conducted for Costco stock price.

Require information from Yahoo Finance, the examine the returned JSON object

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pandas_datareader import data as pdr

import yfinance as yf
```

In [2]:
```
costco = yf.Ticker("COST")

# The info property contains all the key financial data
costco.info
```

Out[2]:

```
{'zip': '98027',
 'sector': 'Consumer Defensive',
 'fullTimeEmployees': 304000,
 'longBusinessSummary': 'Costco Wholesale Corporation, together with its subsidiaries, e
ngages in the operation of membership warehouses in the United States, Puerto Rico, Cana
da, the United Kingdom, Mexico, Japan, Korea, Australia, Spain, France, Iceland, China,
and Taiwan. It offers branded and private-label products in a range of merchandise categ
ories. The company offers sundries, dry groceries, candies, coolers, freezers, liquor, a
nd tobacco and deli products; appliances, electronics, health and beauty aids, hardware,
garden and patio products, sporting goods, tires, toys and seasonal products, office sup
plies, automotive care products, postages, tickets, apparel, small appliances, furnitur
e, domestics, housewares, special order kiosks, and jewelry; and meat, produce, service
deli, and bakery products. It also operates pharmacies, optical, food courts, hearing ai
ds, and tire installation centers, as well as 668 gas stations; and offers online busine
ss delivery, travel, same-day grocery, and various other services. As of August 28, 202
2, the company operated 838 membership warehouses, including 578 in the United States an
d Puerto Rico, 107 in Canada, 40 in Mexico, 31 in Japan, 29 in the United Kingdom, 17 in
Korea, 14 in Taiwan, 13 in Australia, 4 in Spain, 1 in Iceland, 2 in France, and 2 in Ch
ina. It also operates e-commerce websites in the United States, Canada, the United Kingd
om, Mexico, Korea, Taiwan, Japan, and Australia. The company was formerly known as Costc
o Companies, Inc. and changed its name to Costco Wholesale Corporation in August 1999. C
ostco Wholesale Corporation was founded in 1976 and is based in Issaquah, Washington.',
 'city': 'Issaquah',
 'phone': '425 313 8100',
 'state': 'WA',
 'country': 'United States',
 'companyOfficers': [],
 'website': 'https://www.costco.com',
 'maxAge': 1,
 'address1': '999 Lake Drive',
 'industry': 'Discount Stores',
 'ebitdaMargins': 0.04271,
 'profitMargins': 0.02575,
 'grossMargins': 0.12149,
 'operatingCashflow': 7392000000,
 'revenueGrowth': 0.15,
 'operatingMargins': 0.03434,
 'ebitda': 9692999680,
 'targetLowPrice': 455,
 'recommendationKey': 'buy',
 'grossProfits': 27572000000,
 'freeCashflow': 2280875008,
```

```
'targetMedianPrice': 565,
'currentPrice': 523.67,
'earningsGrowth': 0.117,
'currentRatio': 1.022,
'returnOnAssets': 0.07892,
'numberOfAnalystOpinions': 29,
'targetMeanPrice': 563.03,
'debtToEquity': 52.821,
'returnOnEquity': 0.30549,
'targetHighPrice': 678,
'totalCash': 11048999936,
'totalDebt': 10906000384,
'totalRevenue': 226954002432,
'totalCashPerShare': 24.964,
'financialCurrency': 'USD',
'revenuePerShare': 511.56,
'quickRatio': 0.415,
'recommendationMean': 2.1,
'exchange': 'NMS',
'shortName': 'Costco Wholesale Corporation',
'longName': 'Costco Wholesale Corporation',
'exchangeTimezoneName': 'America/New_York',
'exchangeTimezoneShortName': 'EST',
'isEsgPopulated': False,
'gmtOffSetMilliseconds': '-18000000',
'quoteType': 'EQUITY',
'symbol': 'COST',
'messageBoardId': 'finmb_92817',
'market': 'us_market',
'annualHoldingsTurnover': None,
'enterpriseToRevenue': 1.021,
'beta3Year': None,
'enterpriseToEbitda': 23.898,
'52WeekChange': -0.029611766,
'morningStarRiskRating': None,
'forwardEps': 16.15,
'revenueQuarterlyGrowth': None,
'sharesOutstanding': 442604000,
'fundInceptionDate': None,
'annualReportExpenseRatio': None,
'totalAssets': None,
'bookValue': 46.631,
'sharesShort': 4558405,
'sharesPercentSharesOut': 0.010299999,
'fundFamily': None,
'lastFiscalYearEnd': 1661644800,
'heldPercentInstitutions': 0.69106,
'netIncomeToCommon': 5843999744,
'trailingEps': 13.14,
'lastDividendValue': 0.9,
'SandP52WeekChange': -0.15323704,
'priceToBook': 11.2300825,
'heldPercentInsiders': 0.00219,
'nextFiscalYearEnd': 1724803200,
'yield': None,
'mostRecentQuarter': 1661644800,
'shortRatio': 2.12,
'sharesShortPreviousMonthDate': 1664496000,
'floatShares': 441298463,
'beta': 0.717083,
'enterpriseValue': 231640514560,
'priceHint': 2,
'threeYearAverageReturn': None,
'lastSplitDate': 947808000,
'lastSplitFactor': '2:1',
'legalType': None,
```

```
'lastDividendDate': 1666828800,
'morningStarOverallRating': None,
'earningsQuarterlyGrowth': 0.119,
'priceToSalesTrailing12Months': 1.0212573,
'dateShortInterest': 1667174400,
'pegRatio': 3.17,
'ytdReturn': None,
'forwardPE': 32.42539,
'lastCapGain': None,
'shortPercentOfFloat': 0.010299999,
'sharesShortPriorMonth': 4175608,
'impliedSharesOutstanding': 0,
'category': None,
'fiveYearAverageReturn': None,
'previousClose': 521.32,
'regularMarketOpen': 526.87,
'twoHundredDayAverage': 513.1994,
'trailingAnnualDividendYield': 0.006483542,
'payoutRatio': 0.2572,
'volume24Hr': None,
'regularMarketDayHigh': 529.54,
'navPrice': None,
'averageDailyVolume10Day': 2052000,
'regularMarketPreviousClose': 521.32,
'fiftyDayAverage': 491.1172,
'trailingAnnualDividendRate': 3.38,
'open': 526.87,
'toCurrency': None,
'averageVolume10days': 2052000,
'expireDate': None,
'algorithm': None,
'dividendRate': 3.6,
'exDividendDate': 1666828800,
'circulatingSupply': None,
'startDate': None,
'regularMarketDayLow': 518.19,
'currency': 'USD',
'trailingPE': 39.85312,
'regularMarketVolume': 1584684,
'lastMarket': None,
'maxSupply': None,
'openInterest': None,
'marketCap': 231778435072,
'volumeAllCurrencies': None,
'strikePrice': None,
'averageVolume': 2134293,
'dayLow': 518.19,
'ask': 523.67,
'askSize': 900,
'volume': 1584684,
'fiftyTwoWeekHigh': 612.27,
'fromCurrency': None,
'fiveYearAvgDividendYield': 0.83,
'fiftyTwoWeekLow': 406.51,
'bid': 523.03,
'tradeable': False,
'dividendYield': 0.0069,
'bidSize': 1100,
'dayHigh': 529.54,
'coinMarketCapLink': None,
'regularMarketPrice': 523.67,
'preMarketPrice': None,
'logo_url': 'https://logo.clearbit.com/costco.com'}
```

1.Fundemental analysis

Grab the following key financial numbers, and store these numbers into a dataframe

- Market capitalization
- 52-week high
- 52-week low
- PE ratio (price to earning ratio)
- ROE Return on equity
- Dividend

In [5]:
```
df_fundementals = pd.DataFrame(
    {
        'Marketcap':[costco.info['marketCap']],
        '52-week high':[costco.info['fiftyTwoWeekHigh']],
        '52-week low':[costco.info['fiftyTwoWeekLow']],
        'PE ratio':[costco.info["trailingPE"]],
        'ROE':[costco.info["returnOnEquity"]]
    }
)

df_fundementals
```

Out[5]:

	Marketcap	52-week high	52-week low	PE ratio	ROE
0	231778435072	612.27	406.51	39.85312	0.30549

2.Technical analysis

perform a linear regression model

In [6]:
```
from sklearn.model_selection import train_test_split
```

Prepare the Costco data from 2020-01-01 to 2022-11-01

In [7]:
```
df_costco = pd.DataFrame()

startDate = "2020-01-01"
endDate = "2022-11-01"

df_costco = yf.download("COST", start=startDate, end=endDate)
```

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

Examine the first 5 and last 5 rows of data

In [8]:
```
df_costco
```

Out[8]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2020-01-02	294.059998	294.579987	291.000000	291.489990	277.666992	2103600
2020-01-03	290.049988	292.899994	289.329987	291.730011	277.895630	1926000
2020-01-06	290.549988	292.070007	288.619995	291.809998	277.971741	2655100
2020-01-07	291.320007	291.690002	289.279999	291.350006	277.533630	1963400
2020-01-08	290.989990	295.480011	290.500000	294.690002	280.715210	2492800
...	...	...	...	...	...	...

2022-10-25	493.320007	500.190002	490.500000	499.059998	498.160706	2130900
2022-10-26	498.000000	507.420013	495.690002	499.450012	498.550018	2088700
2022-10-27	499.809998	503.010010	494.140015	496.540009	496.540009	1576600
2022-10-28	497.299988	512.820007	495.970001	510.869995	510.869995	2369300
2022-10-31	509.640015	509.640015	500.500000	501.500000	501.500000	2192200

714 rows × 6 columns

Obtain some basic descriptive statistics

In [9]:
```
df_costco.describe()
```

Out[9]:

	Open	High	Low	Close	Adj Close	Volume
count	714.000000	714.000000	714.000000	714.000000	714.000000	7.140000e+02
mean	415.339944	419.694369	411.168502	415.586219	408.905486	2.507109e+06
std	88.367372	89.546005	87.095874	88.371574	92.404797	1.160210e+06
min	280.440002	287.329987	271.279999	279.850006	267.140991	9.725000e+05
25%	340.702507	342.565010	336.939987	339.962494	326.893501	1.770725e+06
50%	384.895004	387.349991	381.650009	384.370010	379.792557	2.170750e+06
75%	494.382492	500.112503	489.505005	495.182510	493.102180	2.839575e+06
max	607.280029	612.270020	597.729980	608.049988	604.924255	9.511600e+06

Perform regression analysis

Explanatory variables: Walmart rate of return, S&P500 return, US Dollar Index(ICE: DX)
Response variables: Costco rate of return

In [10]:
```
# Retrieve the adj-close price for these 4 indices
df_dataset = yf.download("COST WMT ^GSPC DX-Y.NYB", start=startDate, end=endDate)['Adj C
```

```
[*********************100%***********************]  4 of 4 completed
```

Examine the first and last 5 rows and see if it corresponds to actual data

In [11]:
```
df_dataset
```

Out[11]:

	COST	DX-Y.NYB	WMT	^GSPC
Date				
2020-01-02	277.666962	96.849998	113.801071	3257.850098
2020-01-03	277.895599	96.839996	112.796448	3234.850098
2020-01-06	277.971832	96.669998	112.566818	3246.280029
2020-01-07	277.533600	96.980003	111.523895	3237.179932
2020-01-08	280.715302	97.300003	111.141197	3253.050049
...	...	...	...	...
2022-10-25	498.160706	110.949997	140.070007	3859.110107

2022-10-26	498.550018	109.699997	141.139999	3830.600098
2022-10-27	496.540009	110.589996	140.729996	3807.300049
2022-10-28	510.869995	110.669998	142.509995	3901.060059
2022-10-31	501.500000	111.529999	142.330002	3871.979980

717 rows × 4 columns

Conduct a correlation analysis

In [12]:
```
df_dataset.corr()
```

Out[12]:

	COST	DX-Y.NYB	WMT	^GSPC
COST	1.000000	0.403460	0.610435	0.798184
DX-Y.NYB	0.403460	1.000000	-0.245682	-0.127848
WMT	0.610435	-0.245682	1.000000	0.726061
^GSPC	0.798184	-0.127848	0.726061	1.000000

Calculate the daily rate of return for each index, and store these data into a new dataframe

In [13]:
```
df_daily_returns = np.log(df_dataset/df_dataset.shift(1))
df_daily_returns = df_daily_returns.rename({'DX-Y.NYB': 'USDIndex', '^GSPC': 'SP500'}, a
df_daily_returns
```

Out[13]:

	COST	USDIndex	WMT	SP500
Date				
2020-01-02	NaN	NaN	NaN	NaN
2020-01-03	0.000823	-0.000103	-0.008867	-0.007085
2020-01-06	0.000274	-0.001757	-0.002038	0.003527
2020-01-07	-0.001578	0.003202	-0.009308	-0.002807
2020-01-08	0.011399	0.003294	-0.003437	0.004890
...	...	...	...	...
2022-10-25	0.004197	-0.009330	0.004723	0.016136
2022-10-26	0.000781	-0.011330	0.007610	-0.007415
2022-10-27	-0.004040	0.008080	-0.002909	-0.006101
2022-10-28	0.028451	0.000723	0.012569	0.024328
2022-10-31	-0.018512	0.007741	-0.001264	-0.007482

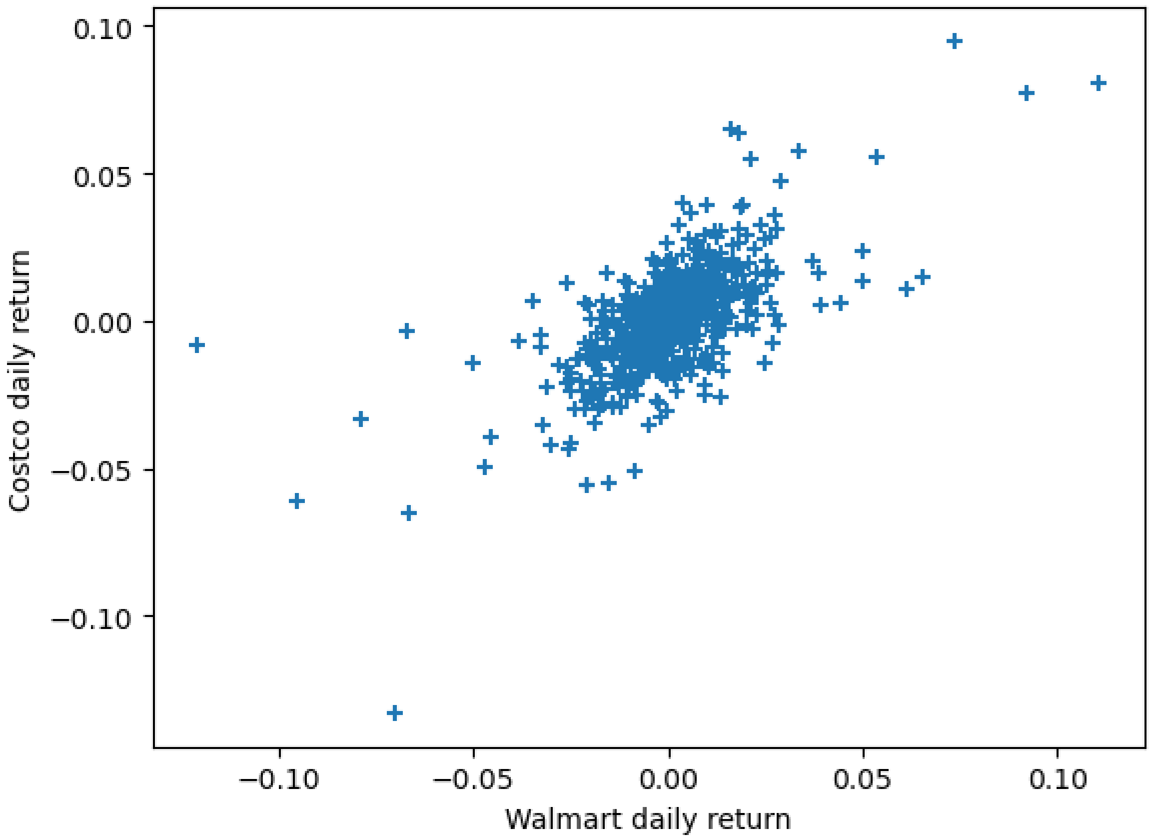
717 rows × 4 columns

Make a scatterplot with:

- explanatory variable: Walmart rate of return
- response variable: Costco rate of return

In [14]:
```
plt.scatter(df_daily_returns['WMT'],df_daily_returns['COST'],marker='+')
```

```
plt.xlabel("Walmart daily return")
plt.ylabel("Costco daily return")
plt.show()
```



There seems to exist a positive correlation between Walmart daily return and Costco daily return.

Perform a univariate regression

Perform a linear regression analysis and see if it gives any predictive power:

In [15]:
```
#import the statsmodel and sklearn library
import statsmodels.formula.api as smf
from sklearn.linear_model import LinearRegression

# perform a OLS regression
slr_sm_model = smf.ols('COST ~ WMT', data=df_daily_returns)

slr_sm_model_ko = slr_sm_model.fit()

print(slr_sm_model_ko.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   COST   R-squared:                       0.431
Model:                            OLS   Adj. R-squared:                  0.430
Method:                 Least Squares   F-statistic:                     536.2
Date:                Sun, 20 Nov 2022   Prob (F-statistic):           9.53e-89
Time:                        13:47:14   Log-Likelihood:                 2086.1
No. Observations:                 710   AIC:                            -4168.
Df Residuals:                     708   BIC:                            -4159.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.0006      0.000      1.282      0.200      -0.000       0.002
```

```
WMT            0.6851      0.030     23.156      0.000       0.627       0.743
==============================================================================
Omnibus:                       92.994   Durbin-Watson:                   2.231
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              847.875
Skew:                          -0.154   Prob(JB):                    7.70e-185
Kurtosis:                       8.345   Cond. No.                         61.4
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

The regression model is shown as follows:

$ \hat{Y_i} = 0.000617 + 0.685057 * X_i $

Intepretation:

- For a unit increase in Walmart ROI, the Costco ROI is expected to increase by 0.685057
- The R^2 coefficient of determination is 0.431, meaning that this model explains 43.1% of variation of the dependent variable..
- The p-value is 0 meaning that this variable is statistically significant

In [11]:
```
slr_sm_model_ko.params
```

Out[11]:
```
Intercept    0.000617
WMT          0.685057
dtype: float64
```

Perform a multivariate regression

The model is shown as follow:

$ \hat{Costco.Return} = \hat{\beta_0} + \hat{\beta_1}*Walmart.Return + \hat{\beta_2} *USD.Index + \hat{\beta_3}*SP500.Return $

In [12]:
```
# pass in dataframe columns as parameters:
mulvar_reg_model = smf.ols(formula='COST ~ WMT + USDIndex + SP500', data=df_daily_return

mulvar_reg_model_fit = mulvar_reg_model.fit()
print(mulvar_reg_model_fit.summary())
# extract the key coefficients
print(mulvar_reg_model_fit.params)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   COST   R-squared:                       0.591
Model:                            OLS   Adj. R-squared:                  0.589
Method:                 Least Squares   F-statistic:                     339.4
Date:                Sat, 19 Nov 2022   Prob (F-statistic):          2.12e-136
Time:                        11:05:56   Log-Likelihood:                 2203.0
No. Observations:                 710   AIC:                            -4398.
Df Residuals:                     706   BIC:                            -4380.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.0006      0.000      1.436      0.151      -0.000       0.001
WMT            0.4609      0.029     16.078      0.000       0.405       0.517
USDIndex      -0.0902      0.094     -0.959      0.338      -0.275       0.094
SP500          0.4701      0.029     16.019      0.000       0.413       0.528
```

```
==============================================================================
Omnibus:                      131.699   Durbin-Watson:                   2.130
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1442.429
Skew:                          -0.461   Prob(JB):                         0.00
Kurtosis:                       9.922   Cond. No.                         230.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
Intercept    0.000588
WMT          0.460883
USDIndex    -0.090166
SP500        0.470134
dtype: float64
```

Observations:

- The USDIndex is insignificant in this model
- The R^2 value goes up from 0.431 to 0.589, meaning that this multivariate model is a better fit compared to the previous univariate model.

Machine Learning forecasting

Use indicator: EMA10(Exponential Moving Average 10 days) for machine learning forecasting. The smoothing value by default is 2.

Prepare the data

In [16]:

```
# import the pandas_ta lib: a pandas library for technical analysis
import pandas_ta as ta

df_costco_ema = pd.DataFrame()

df_costco_ema['COST'] = df_dataset['COST']
df_costco_ema['Exponential MA'] = ta.ema(df_dataset['COST'], length=10)

#drop the first 9 rows where the EMA column shows NaN:
df_costco_ema = df_costco_ema.iloc[9:,:]
df_costco_ema.head(5)
```

Out[16]:

Date	COST	Exponential MA
2020-01-15	286.554565	281.788797
2020-01-16	288.659729	283.038057
2020-01-17	290.231445	284.345946
2020-01-21	298.404602	286.902065
2020-01-22	297.099609	288.756164

Clean the dataset, remove any rows that contains NaN

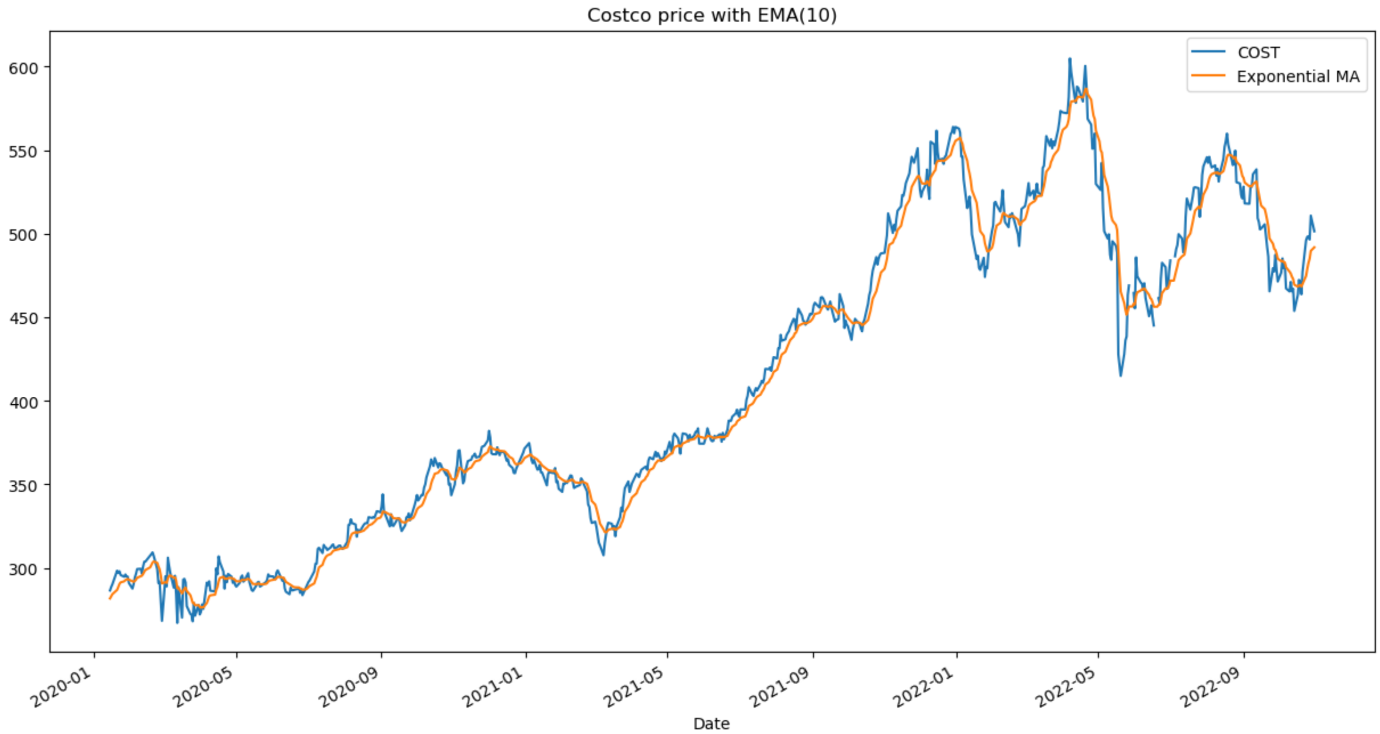
In [17]:

```
df_costco_ema[df_costco_ema.isna().any(axis=1)]
df_costco_ema = df_costco_ema[df_costco_ema['Exponential MA'].notna()]
```

In [18]:

```
df_costco_ema.plot(figsize=(15,8),title="Costco price with EMA(10)")
```

```
plt.show()
```



Split the data into training test & testing set

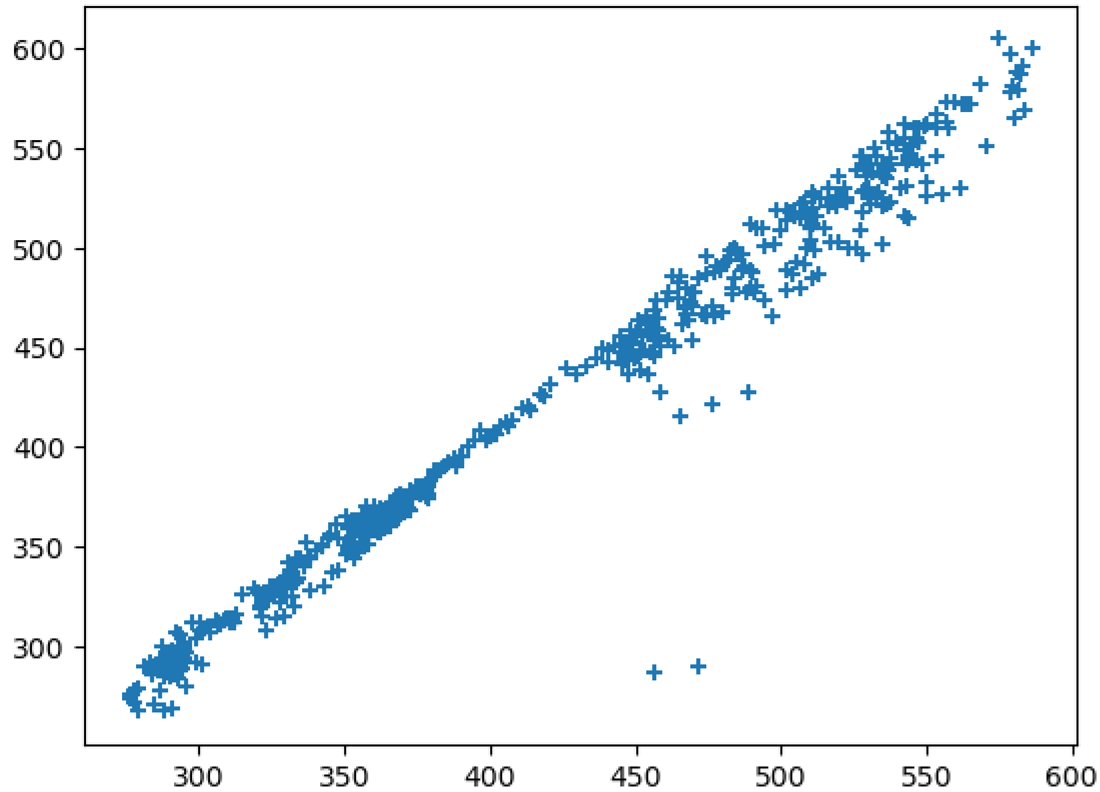
The ratio of training:testing would be 8:2, meaning we use 80% of the data to train the model, and the rest 20% to test the model

In [21]:
```
# prepare the training-testing data
train,test = train_test_split(df_costco_ema, test_size=0.2)

# data cleaning: there exists some NaN values, so fill the NaN with the previous non-NaN
train.isnull().sum()
test.isnull().sum()
train = train.fillna(method='ffill')
test = test.fillna(method='ffill')
```

Plot a scatterplot

In [22]:
```
plt.scatter(train["Exponential MA"],train["COST"], marker="+")
plt.show()
```



Try to fit a linear regression model on the testing set

In [23]:
```
model = LinearRegression()
model.fit(train[["Exponential MA"]], train["COST"])
print(model.intercept_, model.coef_, model.score(train[["Exponential MA"]], train["COST"
```

```
3.117770326699258 [0.99399718] 0.9746926183305339
```

The regression model is calculated as follow:

$ \hat{Costo.Price} = 3.11777 + 0.99399 * EMA(10) $

Test the model

In [113…
```
COST_pred = model.predict(test[["Exponential MA"]])
len(COST_pred)
```

Out[113]:
```
142
```

In [128…
```
plt.scatter(test["Exponential MA"], test["COST"], color = 'orange', s=5)
plt.plot(test["Exponential MA"], model.predict(test[["Exponential MA"]]), color = 'black
plt.title('Predicted price vs Real price')
plt.xlabel('EMA(10) value')
plt.ylabel('Costco price')
plt.legend(["Actual values","Predicted values"])
plt.grid(color = 'green', linestyle = '--', linewidth = 0.25)
plt.show()
```

Plot the residuals as well

In [136…

```
plt.scatter(test["Exponential MA"],test["COST"]-model.predict(test[["Exponential MA"]]),
plt.axhline(y=0, color='r', linestyle='--')
```

Out[136]:

```
<matplotlib.lines.Line2D at 0x14cc548df70>
```