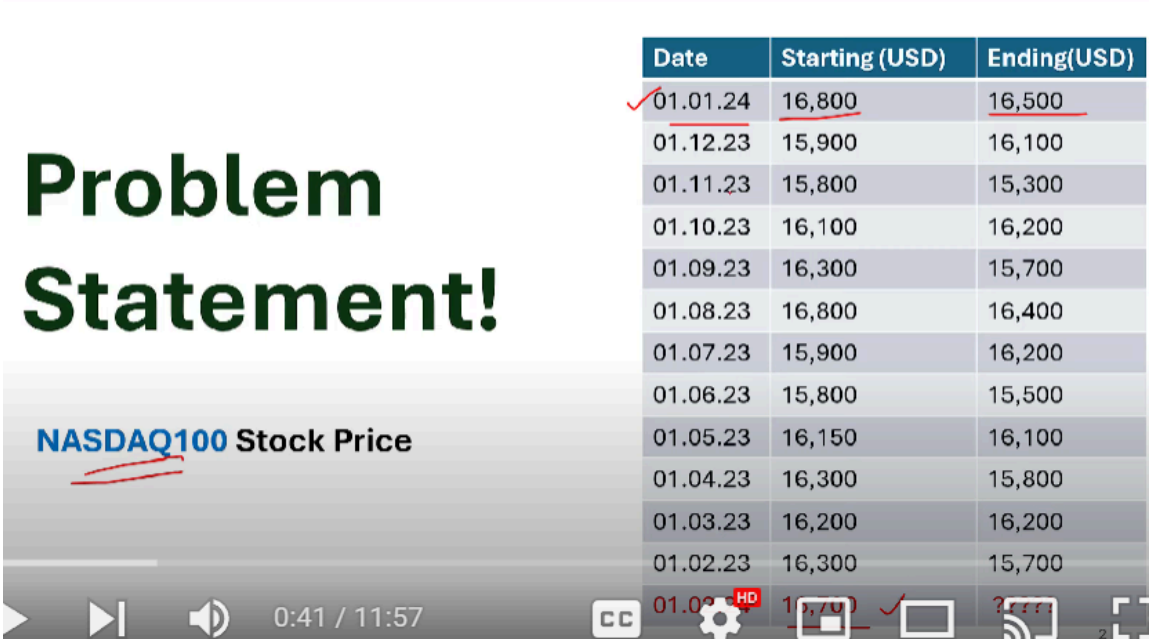
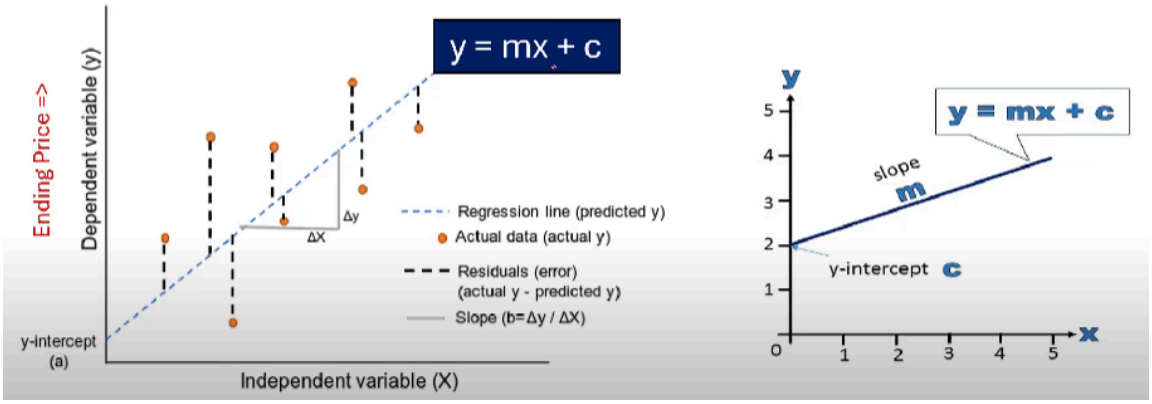


the last row: start price on 1/2/2024 is USD 16700. We have to find out the predicted end price

we will ignore the date column. X is the starting price and Y is the ending price

we will apply simple linear regreesion algorithm Y = mX + C



we will use supervised ML algorithm. Scikit Learn library has algorithms to solve supervised ML problems (classifications, linear regression etc)

if you don't have install the libraries: pandas, matplotlib, sklearn

In [3]:
```
import pandas as pd
from matplotlib import pyplot as plt
import sklearn
```

In [4]:
```
df = pd.read_csv('nasdaq100.csv')
df
```

Out[4]:

	date	starting_price	ending_price
0	01.01.24	16800	16500
1	01.12.23	15900	16100
2	01.11.23	15800	15300
3	01.10.23	16100	16200
4	01.09.23	16300	15700
5	01.08.23	16800	16400
6	01.07.23	15900	16200
7	01.06.23	15800	15500
8	01.05.23	16150	16100
9	01.04.23	16300	15800
10	01.03.23	16200	16200
11	01.02.23	16300	15700

In [5]:
```
# checking for null values in any of the entries
df.isnull().sum()
```

Out[5]:
```
date              0
starting_price    0
ending_price      0
dtype: int64
```

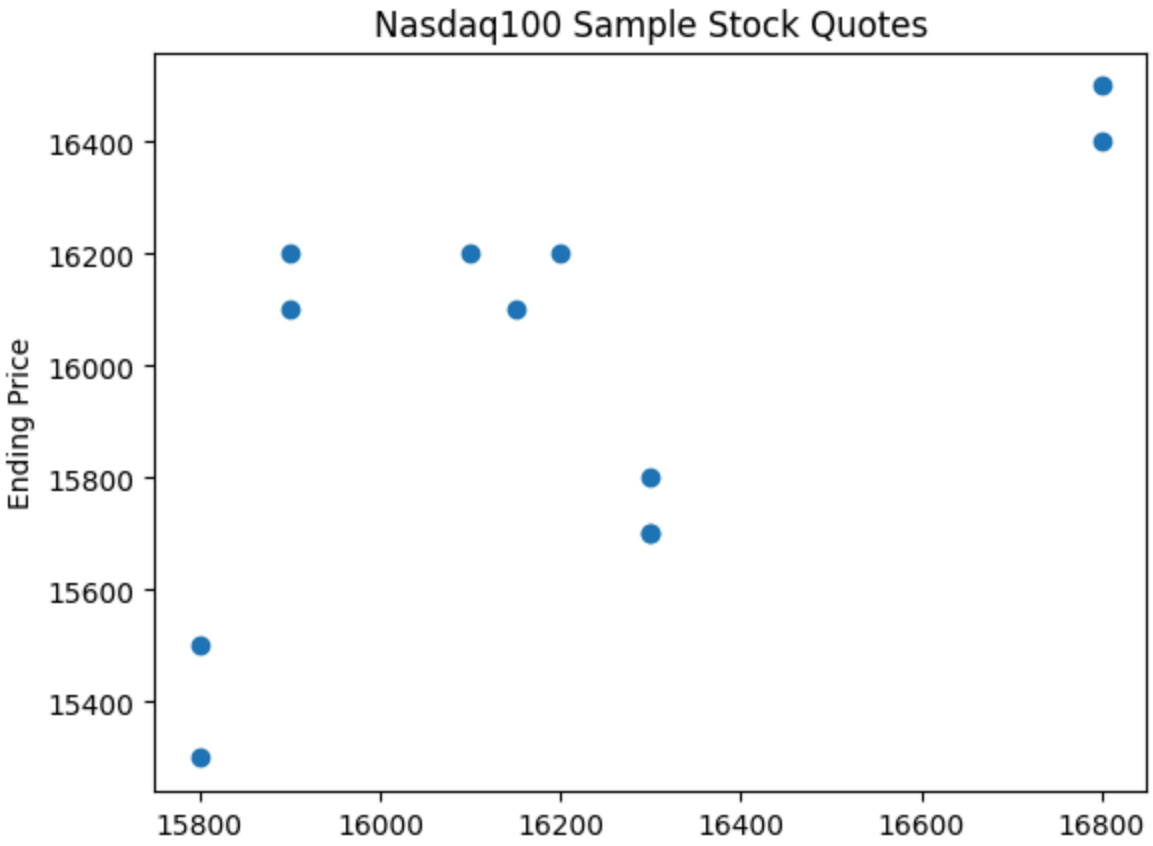
In [6]:
```
# drop the date column
df2 = df.drop(columns='date')
df2
```

Out[6]:

	starting_price	ending_price
0	16800	16500
1	15900	16100
2	15800	15300
3	16100	16200
4	16300	15700
5	16800	16400
6	15900	16200
7	15800	15500
8	16150	16100
9	16300	15800
10	16200	16200
11	16300	15700

In [11]:
```
# generate a scatter plot with X and Y data including X & Y labels and a title

plt.scatter(df['starting_price'], df['ending_price'])
plt.xlabel = ('Starting Price')
plt.ylabel('Ending Price')
plt.title('Nasdaq100 Sample Stock Quotes')
```

Out[11]:
```
Text(0.5, 1.0, 'Nasdaq100 Sample Stock Quotes')
```



In [47]:
```
# DO NOT DROP COLUMN 'ending_price'. this will not make the best fit line go throug

x = df2[['starting_price']]
x.head()
```

Out[47]:

	starting_price
0	16800
1	15900
2	15800
3	16100
4	16300

In [48]:
```
y = df2[['ending_price']]
y.head()
```

Out[48]:

	ending_price
0	16500
1	16100
2	15300
3	16200
4	15700

Linear Regression

search for linear regression sklearn

In [22]:
```
from sklearn.linear_model import LinearRegression # to seel other algorithms press

reg = LinearRegression()
```

In [23]:
```
x.mean()
```

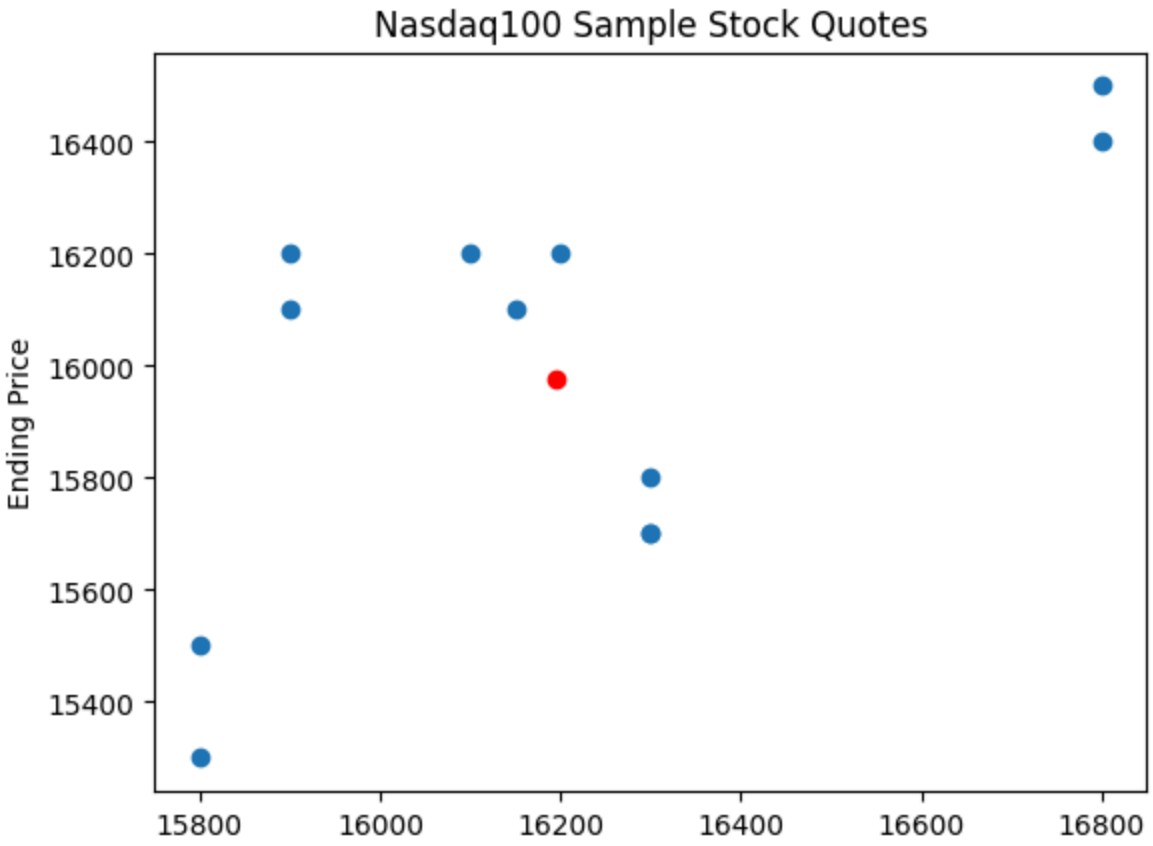
Out[23]:
```
starting_price    16195.833333
dtype: float64
```

In [24]:
```
y.mean()
```

Out[24]:
```
ending_price    15975.0
dtype: float64
```

In [28]:
```
plt.scatter(x.mean(), y.mean(), color='red')
plt.scatter(df['starting_price'], df['ending_price'])
plt.xlabel = ('Starting Price')
plt.ylabel('Ending Price')
plt.title('Nasdaq100 Sample Stock Quotes')
```

Out[28]:
```
Text(0.5, 1.0, 'Nasdaq100 Sample Stock Quotes')
```



In [29]:
```
# train the model

reg.fit(x, y)
```

Out[29]:
```
LinearRegression()
```

In [36]:
```
# value of m of the regression formula Y = mX + C
m = reg.coef_
```

In [37]:
```
# value of C
c = reg.intercept_
```

In [38]:
```
# now solve the equation Y =  mX + C for the value of X=16700. The predicted result

y = m * 16700 + c
y
```

Out[38]:
```
array([[16304.0105628]])
```

In [41]:
```
# don't need to find m or c or y. use the predict function
reg.predict([[16700]])
```

```
C:\Users\islam\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[41]:
```
array([[16304.0105628]])
```

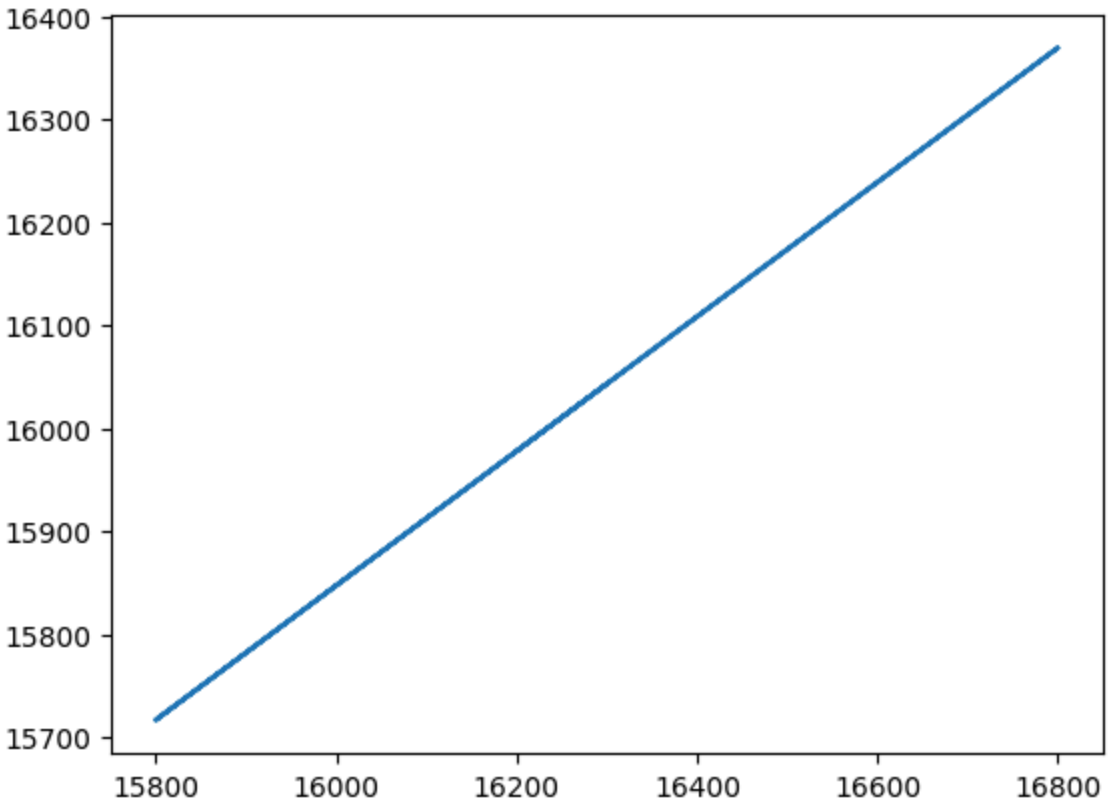
In [42]:
```
# check if the best fit line goes through the red dot

df['Predicted_y'] = reg.predict(x)
df
```

Out[42]:

	date	starting_price	ending_price	Predicted_y
0	01.01.24	16800	16500	16369.268856
1	01.12.23	15900	16100	15781.944215
2	01.11.23	15800	15300	15716.685922
3	01.10.23	16100	16200	15912.460802
4	01.09.23	16300	15700	16042.977389
5	01.08.23	16800	16400	16369.268856
6	01.07.23	15900	16200	15781.944215
7	01.06.23	15800	15500	15716.685922
8	01.05.23	16150	16100	15945.089949
9	01.04.23	16300	15800	16042.977389
10	01.03.23	16200	16200	15977.719096
11	01.02.23	16300	15700	16042.977389

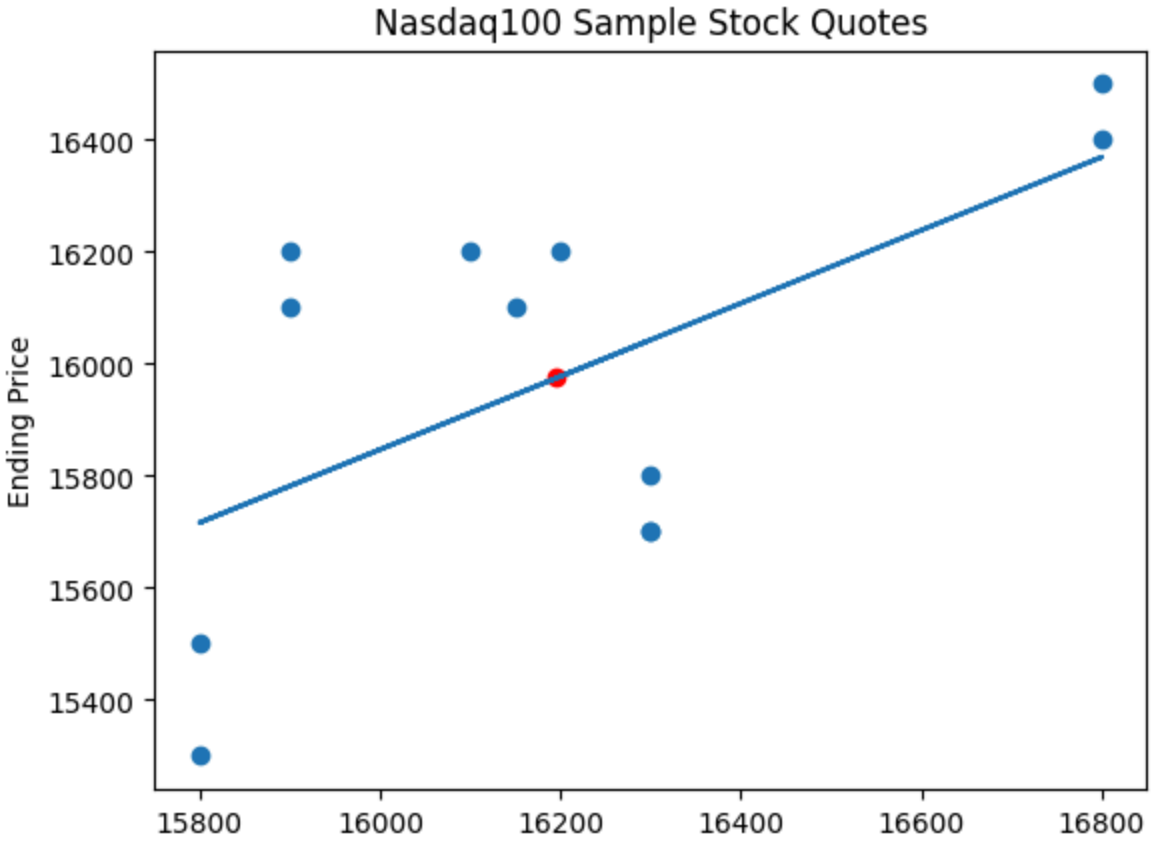
In [44]:
```
plt.plot(x, reg.predict(x))
```

Out[44]:
```
[<matplotlib.lines.Line2D at 0x188c837c920>]
```



In [49]:
```
plt.plot(x, reg.predict(x))
plt.scatter(x.mean(), y.mean(), color='red')
plt.scatter(df['starting_price'], df['ending_price'])
plt.xlabel = ('Starting Price')
plt.ylabel('Ending Price')
plt.title('Nasdaq100 Sample Stock Quotes')
```

Out[49]:
```
Text(0.5, 1.0, 'Nasdaq100 Sample Stock Quotes')
```

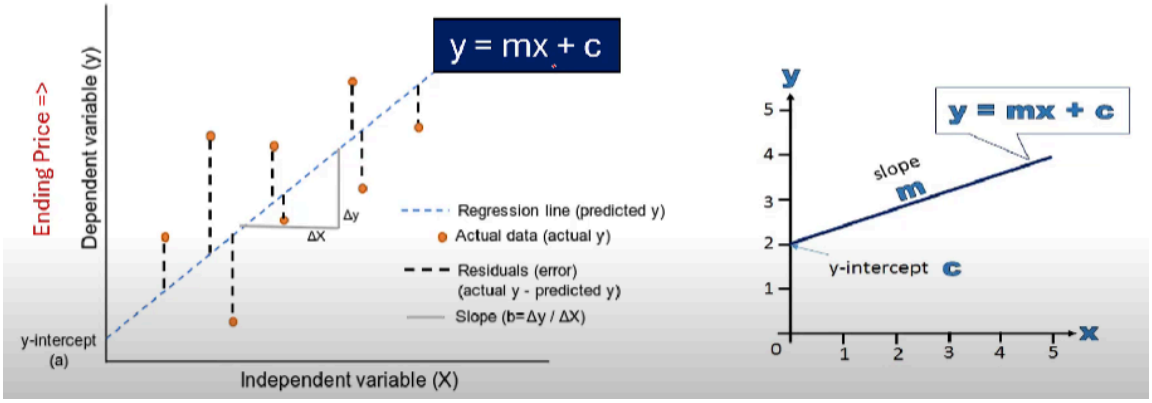


Note: the above best fitting line would not go through the red dot when I tried to set the x and y value by dropping columns

example: x = df.drop(columns='ending_price')

simply do: x = df[['ending_price']]

Loss & Cost



Loss: every delta changes of x & y is loss.

Cost: sum of all delta changes of x& y is cost

In [52]:
```
# check delta loss

df['loss'] = df['ending_price'] - df['Predicted_y']
df
```

Out[52]:

	date	starting_price	ending_price	Predicted_y	loss
0	01.01.24	16800	16500	16369.268856	130.731144
1	01.12.23	15900	16100	15781.944215	318.055785
2	01.11.23	15800	15300	15716.685922	-416.685922
3	01.10.23	16100	16200	15912.460802	287.539198
4	01.09.23	16300	15700	16042.977389	-342.977389
5	01.08.23	16800	16400	16369.268856	30.731144
6	01.07.23	15900	16200	15781.944215	418.055785
7	01.06.23	15800	15500	15716.685922	-216.685922
8	01.05.23	16150	16100	15945.089949	154.910051
9	01.04.23	16300	15800	16042.977389	-242.977389
10	01.03.23	16200	16200	15977.719096	222.280904
11	01.02.23	16300	15700	16042.977389	-342.977389

We may use MAE, MSE or RMSE to determine the Cost

In [57]:
```
# go to  Scikit learn and copy the libraries for the MAE, MSE and RMSE

from sklearn.metrics import mean_absolute_error, mean_squared_error, root_mean_squa
```

In [55]:
```
mae = mean_absolute_error(df['ending_price'], df['Predicted_y'])
mae
```

Out[55]: 260.3840017604666

In [56]:
```
mse = mean_squared_error(df['ending_price'], df['Predicted_y'])
mse
```

Out[56]: 80411.23397700385

In [58]:
```
rmse = root_mean_squared_error(df['ending_price'], df['Predicted_y'])
rmse
```

Out[58]:
```
283.5687464742965
```

In [59]:
```
# cross check the function valueof mae by calculating manually

df['loss']
```

Out[59]:
```
0     130.731144
1     318.055785
2    -416.685922
3     287.539198
4    -342.977389
5      30.731144
6     418.055785
7    -216.685922
8     154.910051
9    -242.977389
10    222.280904
11   -342.977389
Name: loss, dtype: float64
```

In [60]:
```
abs(df['loss'])
```

Out[60]:
```
0     130.731144
1     318.055785
2     416.685922
3     287.539198
4     342.977389
5      30.731144
6     418.055785
7     216.685922
8     154.910051
9     242.977389
10    222.280904
11    342.977389
Name: loss, dtype: float64
```

In [61]:
```
sum(abs(df['loss']))
```

Out[61]:
```
3124.6080211255994
```

In [63]:
```
avg_cost = sum(abs(df['loss']))/len(x)
avg_cost
```

Out[63]:
```
260.3840017604666
```

check performance of the model

In [64]:
```
performance = reg.score(x, y)
performance
```

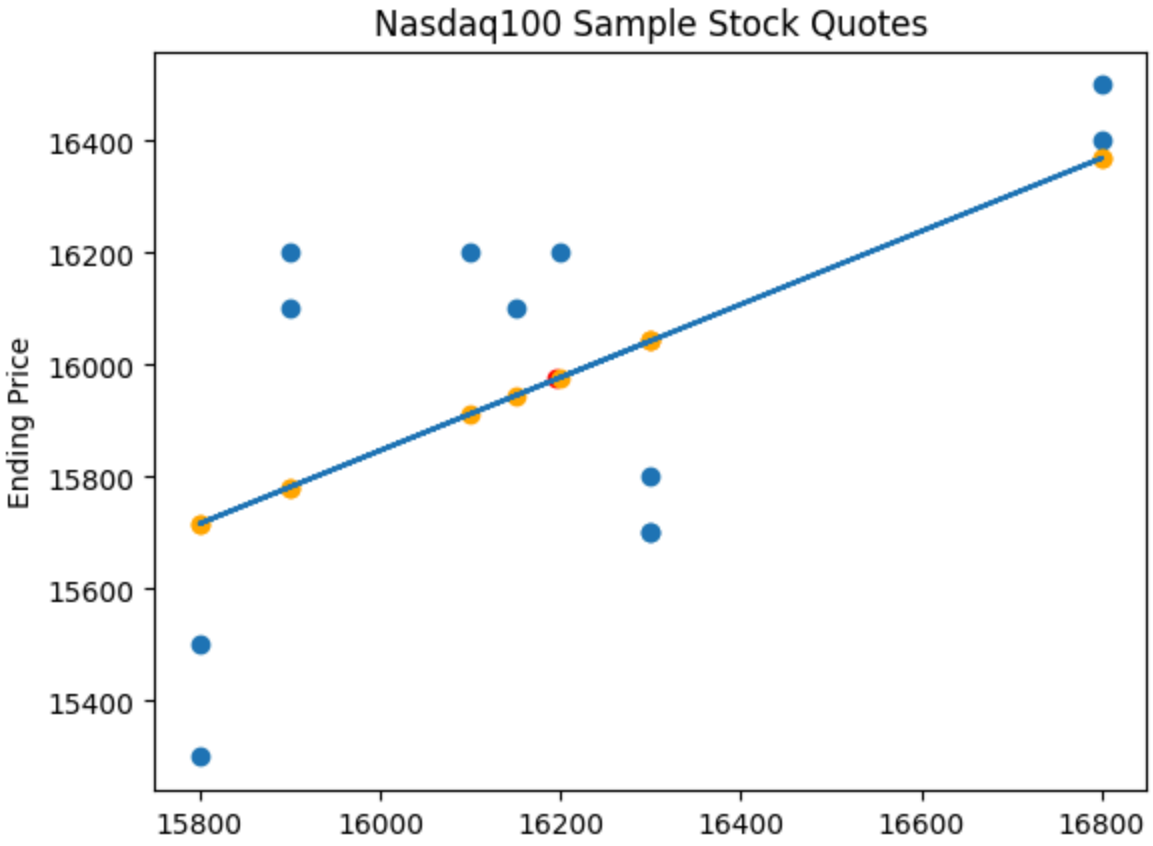
Out[64]: 0.3577804940272571

36% is a poor performance of the model. Look at the scatter plot. values are all over the places

if you use the reg.predict(x) instead of df['ending_price'] you will see the predicted_y would mostly fall on the best fiteed line

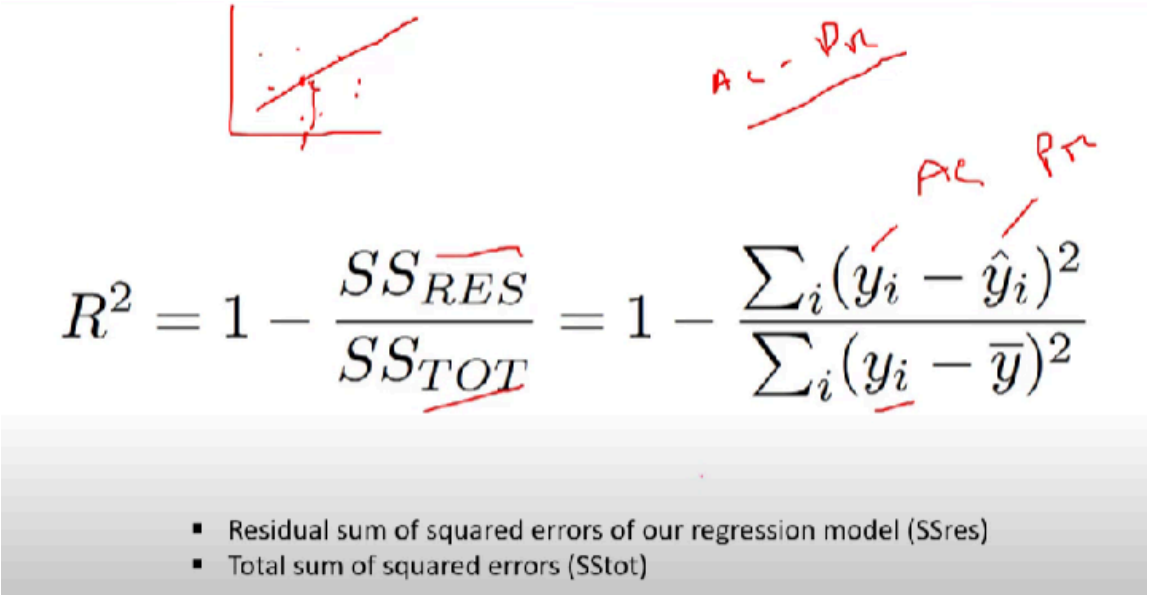
In [66]:
```
plt.plot(x, reg.predict(x))
plt.scatter(x.mean(), y.mean(), color='red')
plt.scatter(df['starting_price'], df['ending_price'])
plt.scatter(df['starting_price'], reg.predict(x), color='orange')
plt.xlabel = ('Starting Price')
plt.ylabel('Ending Price')
plt.title('Nasdaq100 Sample Stock Quotes')
```

Out[66]: Text(0.5, 1.0, 'Nasdaq100 Sample Stock Quotes')



You can check the performance of a model with other methods. One of them is R**2

actual - predicted = residual or loss



In [67]:
```
# will measure performance with r2_score. import r2_score syntax from sklearn
from sklearn.metrics import r2_score
```

In [68]:
```
# let's revisit the performance with score function. use strating_price or x value
performance = reg.score(x, y)
performance
```

Out[68]: 0.3577804940272571

In [73]:
```
# function using ending_price or y and predicted y
performance2 = r2_score(y, reg.predict(x))
performance2
```

Out[73]: 0.3577804940272571

In [ ]: