```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn import metrics

import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('tatamotors.csv')
df.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2020-10-01	135.699997	136.500000	132.600006	133.500000	133.500000	1674311.0
1	2020-10-05	137.000000	137.500000	132.050003	133.899994	133.899994	2800303.0
2	2020-10-06	137.800003	145.699997	135.899994	144.850006	144.850006	10190922.0
3	2020-10-07	144.100006	144.500000	139.800003	141.000000	141.000000	4032654.0
4	2020-10-08	142.800003	143.350006	139.649994	140.899994	140.899994	2491175.0

Next steps: Generate code with df View recommended plots New interactive sheet

```
df.shape
```

```
(248, 7)
```

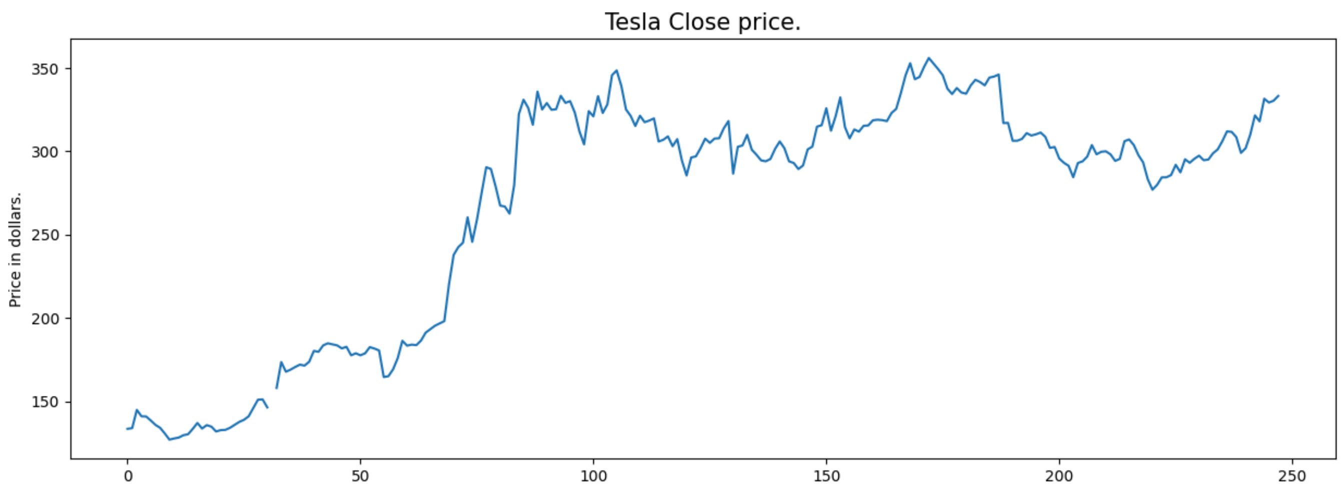
```
df.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	247.000000	247.000000	247.000000	247.000000	247.000000	2.470000e+02
mean	268.256882	272.963360	263.290485	267.972268	267.972268	3.918008e+06
std	70.885356	71.814761	69.668101	70.595849	70.595849	3.488840e+06
min	128.000000	130.000000	126.000000	127.000000	127.000000	4.781380e+05
25%	186.849998	188.675003	183.925003	186.375000	186.375000	1.715184e+06
50%	299.299988	303.500000	293.649994	298.799988	298.799988	2.891967e+06
75%	317.050003	324.125000	312.125000	318.025009	318.025009	4.656536e+06
max	356.500000	360.649994	351.200012	356.000000	356.000000	2.285476e+07

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 248 entries, 0 to 247
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       248 non-null    object 
 1   Open       247 non-null    float64
 2   High       247 non-null    float64
 3   Low        247 non-null    float64
 4   Close      247 non-null    float64
 5   Adj Close  247 non-null    float64
 6   Volume     247 non-null    float64
dtypes: float64(6), object(1)
memory usage: 13.7+ KB
```

```
plt.figure(figsize=(15,5))
plt.plot(df['Close'])
plt.title('Tesla Close price.', fontsize=15)
plt.ylabel('Price in dollars.')
plt.show()
```



```
df.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2020-10-01	135.699997	136.500000	132.600006	133.500000	133.500000	1674311.0
1	2020-10-05	137.000000	137.500000	132.050003	133.899994	133.899994	2800303.0
2	2020-10-06	137.800003	145.699997	135.899994	144.850006	144.850006	10190922.0
3	2020-10-07	144.100006	144.500000	139.800003	141.000000	141.000000	4032654.0
4	2020-10-08	142.800003	143.350006	139.649994	140.899994	140.899994	2491175.0

Next steps: Generate code with df View recommended plots New interactive sheet

```
df[df['Close'] == df['Adj Close']].shape
```

```
(247, 7)
```

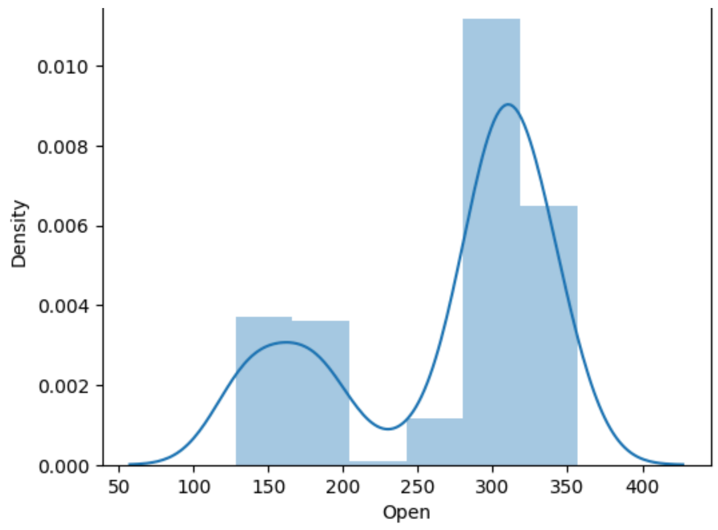
```
print(df.columns)
df = df.drop(['Adj Close'], axis=1)
```

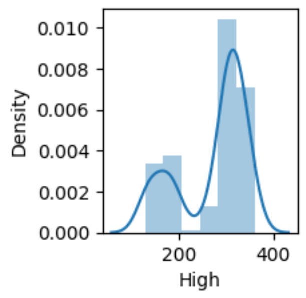
```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

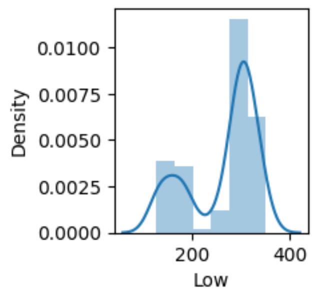
```
df.isnull().sum()
```

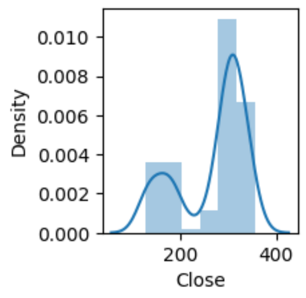
	0
Date	0
Open	1
High	1
Low	1
Close	1
Volume	1

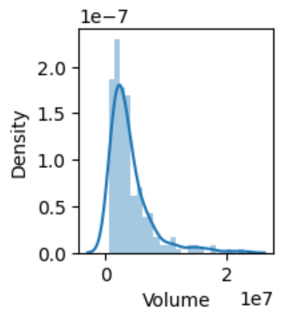
```
features = ['Open', 'High', 'Low', 'Close', 'Volume']

plt.subplots(figsize=(20,10))

for i, col in enumerate(features):
    plt.subplot(2,3,i+1) # Indent this line
    sb.distplot(df[col])
    plt.show()
```



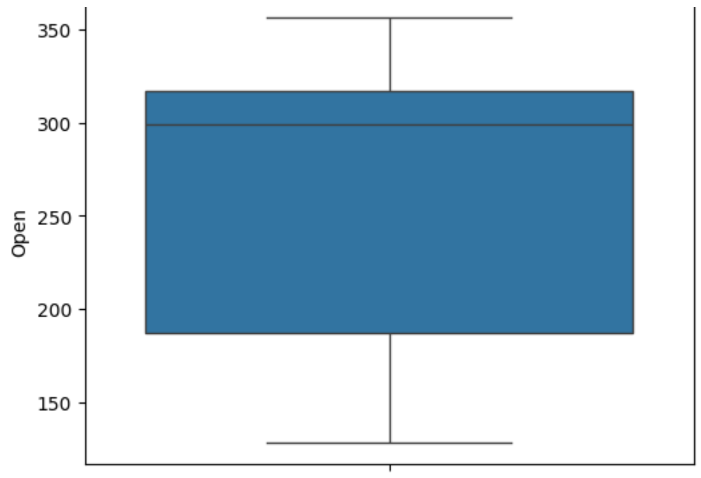


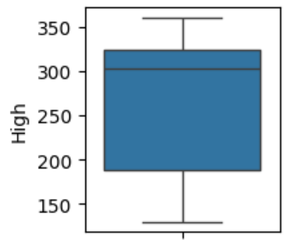


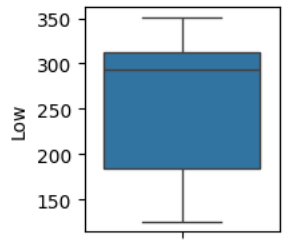


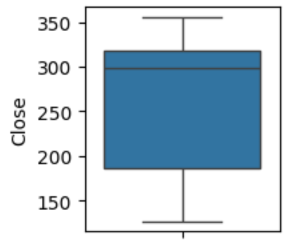


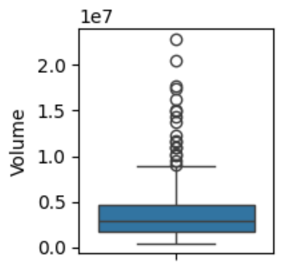
```
plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
    plt.subplot(2,3,i+1) # Indent this line by adding 4 spaces
    sb.boxplot(df[col])
    plt.show()
```







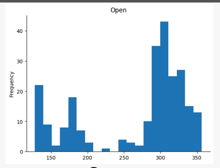


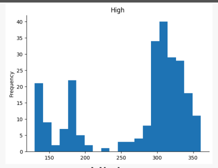


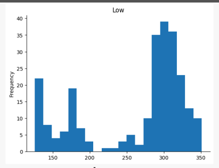
```
splitted = df['Date'].str.split('-', expand=True)

df['year'] = splitted[0].astype('int')
df['month'] = splitted[1].astype('int')
df['day'] = splitted[2].astype('int')

df.head()
```

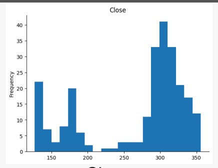
```
         Date        Open        High         Low       Close      Volume  year  month  day
0  2020-10-01  135.699997  136.500000  132.600006  133.500000   1674311.0  2020     10    1
1  2020-10-05  137.000000  137.500000  132.050003  133.899994   2800303.0  2020     10    5
2  2020-10-06  137.800003  145.699997  135.899994  144.850006  10190922.0  2020     10    6
3  2020-10-07  144.100006  144.500000  139.800003  141.000000   4032654.0  2020     10    7
4  2020-10-08  142.800003  143.350006  139.649994  140.899994   2491175.0  2020     10    8
```

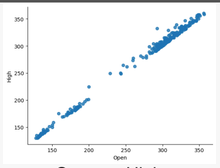
Next steps: Generate code with df | View recommended plots | New interactive sheet

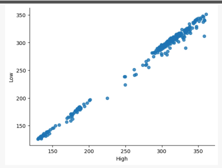

Open

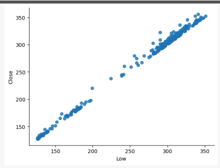

High

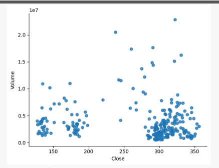

Low

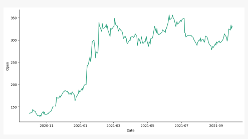

Close

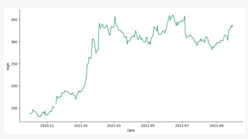

Open vs High

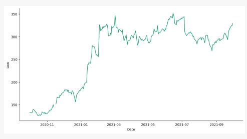

High vs Low

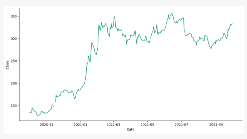

Low vs Close

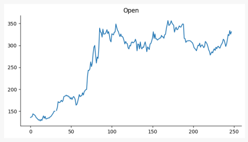

Close vs Volume

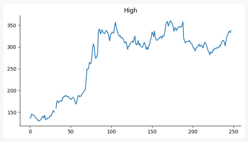

Date vs Open

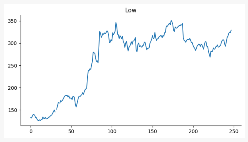

Date vs High

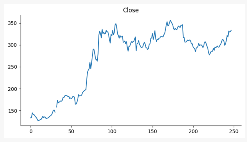

Date vs Low


Date vs Close


Open


High


Low

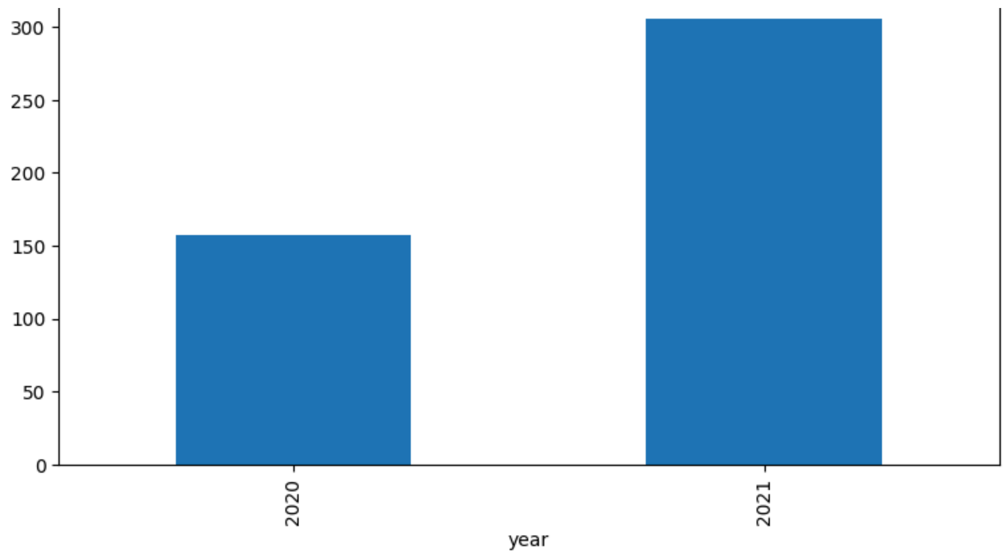

Close

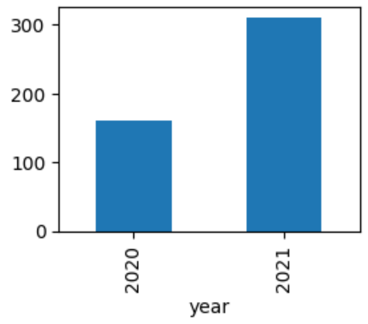
```
df['is_quarter_end'] = np.where(df['month']%3==0,1,0)
df.head()
```

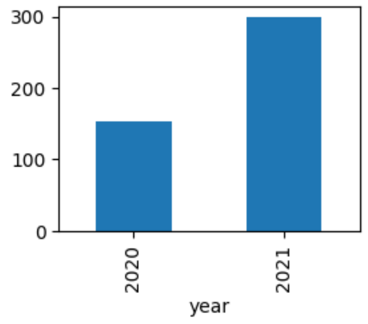
```
         Date        Open        High         Low       Close      Volume  year  month  day  is_quarter_end
0  2020-10-01  135.699997  136.500000  132.600006  133.500000   1674311.0  2020     10    1               0
1  2020-10-05  137.000000  137.500000  132.050003  133.899994   2800303.0  2020     10    5               0
2  2020-10-06  137.800003  145.699997  135.899994  144.850006  10190922.0  2020     10    6               0
3  2020-10-07  144.100006  144.500000  139.800003  141.000000   4032654.0  2020     10    7               0
4  2020-10-08  142.800003  143.350006  139.649994  140.899994   2491175.0  2020     10    8               0
```

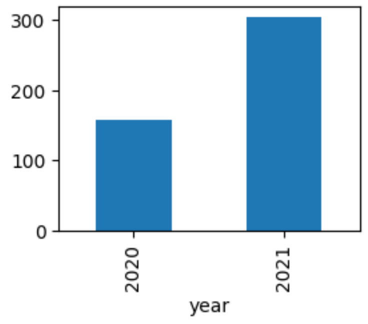
Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# Convert 'Date' to datetime object after splitting
splitted = df['Date'].str.split('-', expand=True)
df['year'] = splitted[0].astype('int')
df['month'] = splitted[1].astype('int')
df['day'] = splitted[2].astype('int')

# Create a datetime column
df['Date'] = pd.to_datetime(df[['year', 'month', 'day']])

# Now group by year and calculate the mean
data_grouped = df.groupby('year').mean()

# Continue with your plotting code
plt.subplots(figsize=(20,10))
for i, col in enumerate(['Open', 'High', 'Low', 'Close']):
    plt.subplot(2,2,i+1)
    data_grouped[col].plot.bar()
    plt.show()
```







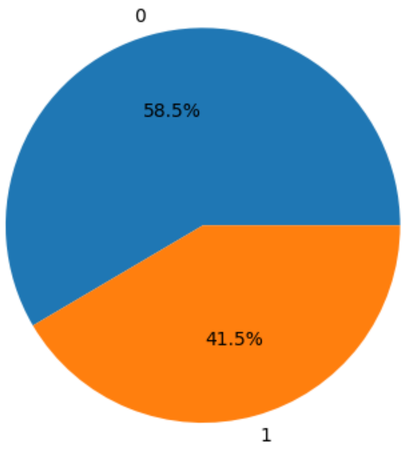


```
df.groupby('is_quarter_end').mean()
```

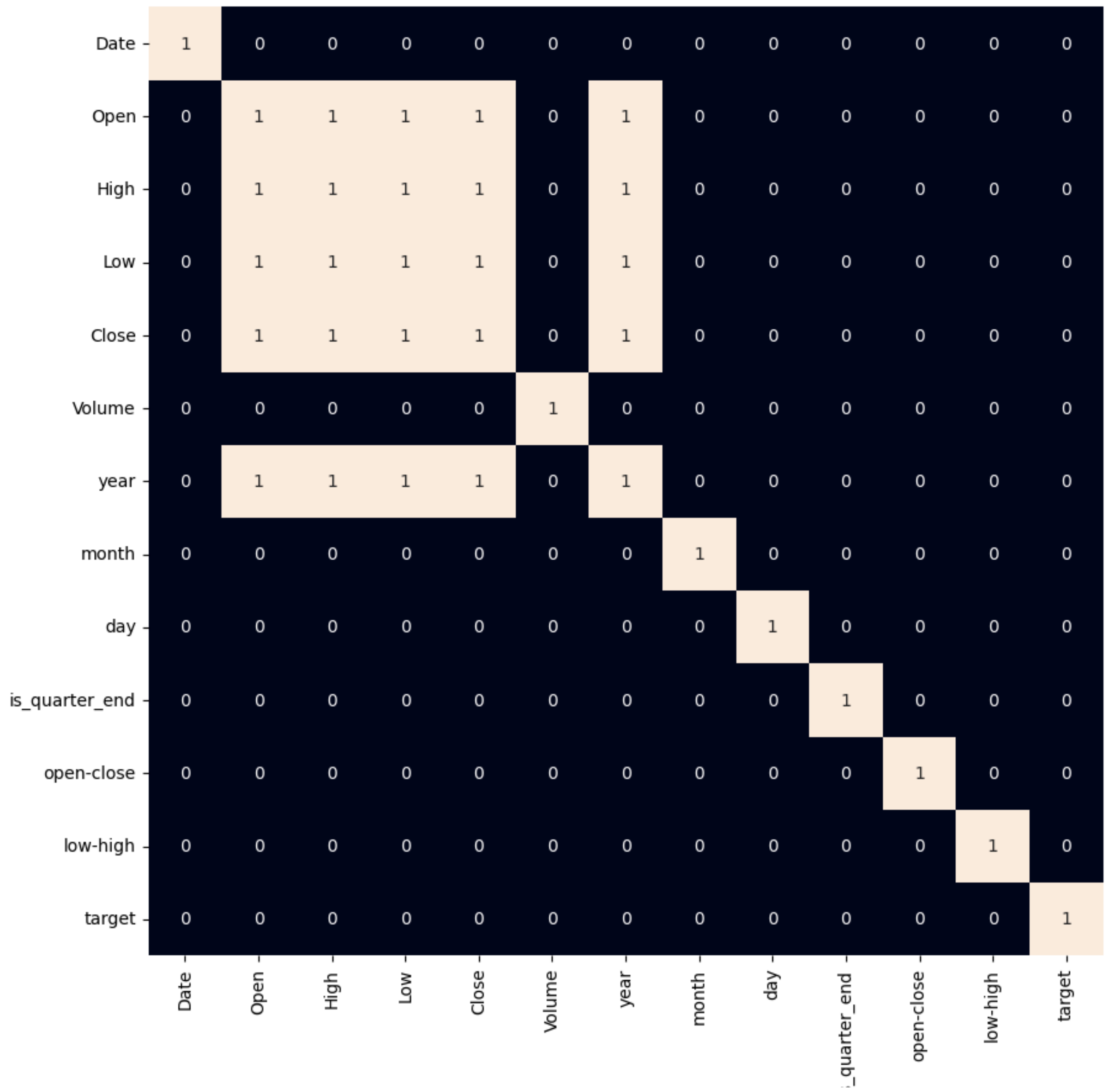
is_quarter_end	Date	Open	High	Low	Close	Volume	year	month	day
0	2021-03-17 06:04:26.666666752	258.995961	263.949068	253.988820	258.778261	4.499647e+06	2020.746914	6.055556	15.567901

```
df['open-close'] = df['Open'] - df['Close']
df['low-high'] = df['Low'] - df['High']
df['target'] = np.where(df['Close'].shift(-1) > df['Close'], 1, 0)
```

```
plt.pie(df['target'].value_counts().values,
        labels=[0, 1], autopct='%1.1f%%')
plt.show()
```



```
plt.figure(figsize=(10, 10))
sb.heatmap(df.corr() > 0.9, annot=True, cbar=False)
plt.show()
```


Date
Open
High
Low
Close
Volume
year
month
day
is_quarter_end
open-close
low-high
target