Yahoo Finance Ticker Statistics Page Scraper

This script takes a list of tickers and scraps its info from the yahoo finance statistics page.

In [2]:
```
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.chrome.service import Service as ChromeService
import pandas as pd
from datetime import date
```

In [8]:
```
tickers = ['CRWD',
           'TEAM',
           'ZS',
           'NET',
           'SNOW',
           'HUBS',
           'U',
           'DDOG',
           'DOCN',
           'ZM',
           'SEMR',
           'MDB',
           'SHOP']

hashtable = {}
```

Retrieving the columns and renaming accordingly

1. Create a driver to access a statistics page for a ticker to extract the metric titles of all the tables.
2. Compile all the metric titles into a single dataframe through concatenation.

In [4]:
```
driver = webdriver.Chrome(service=ChromeService(ChromeDriverManager().install()))
driver.maximize_window()
driver.get('https://finance.yahoo.com/quote/TSLA/key-statistics?p=TSLA')

dfs = []

#find all tables in the div container using xpath
tables = driver.find_elements(by='xpath',
                              value='//*[@id="Col1-0-KeyStatistics-Proxy"]/section/div[2]//table

#iterate through each table and rows and get the 1st td (table data), save into a list
#save each table into a list for retrieval later
for table in tables:
    spec_name = []
    for row in table.find_elements(by='xpath', value=".//tr"):
        box = row.find_elements(by='xpath', value = './/td')
        spec_name.append(box[0].get_attribute('textContent'))
    df = pd.DataFrame({"Metrics":spec_name})
    dfs.append(df)
```

```
driver.quit()

#combine all the dataframes from the list of dataframes
all_stats_df = pd.DataFrame()
for n in range(len(dfs)):
    all_stats_df = pd.concat([all_stats_df, dfs[n]])

all_stats_df.head(10)
```

Out[4]:

	Metrics
0	Market Cap (intraday)
1	Enterprise Value
2	Trailing P/E
3	Forward P/E
4	PEG Ratio (5 yr expected)
5	Price/Sales (ttm)
6	Price/Book (mrq)
7	Enterprise Value/Revenue
8	Enterprise Value/EBITDA
0	Beta (5Y Monthly)
1	52-Week Change 3
2	S&P500 52-Week Change 3
3	52 Week High 3
4	52 Week Low 3
5	50-Day Moving Average 3

Tidying up the dataframe

Notice that there are annotation numbers at the back of certain rows, there are also rows with dates.

1. Reset the index.
2. Use Regex to remove annotation and dates.

In [6]:

```
all_stats_df.reset_index(inplace=True)
all_stats_df.drop(['index'],axis = 1, inplace=True)
all_stats_df['Metrics'].replace(regex={r'[0-9]$': ''}, inplace = True)
#removes the annotations appearing at the end of rows

display(all_stats_df.iloc[23:28])
all_stats_df.iloc[23:28,0].replace(regex={r'(\(.+\))': ''}, inplace = True)
#remove the dates under rows 23-27

display(all_stats_df.iloc[23:28])

all_stats_df.head(10)
```

	Metrics
23	Shares Short (Aug 14, 2022)
24	Short Ratio (Aug 14, 2022)
25	Short % of Float (Aug 14, 2022)
26	Short % of Shares Outstanding (Aug 14, 2022)
27	Shares Short (prior month Jul 14, 2022)

	Metrics
23	Shares Short
24	Short Ratio
25	Short % of Float
26	Short % of Shares Outstanding
27	Shares Short

Out[6]:

	Metrics
0	Market Cap (intraday)
1	Enterprise Value
2	Trailing P/E
3	Forward P/E
4	PEG Ratio (5 yr expected)
5	Price/Sales (ttm)
6	Price/Book (mrq)
7	Enterprise Value/Revenue
8	Enterprise Value/EBITDA
9	Beta (5Y Monthly)

Scraping all the ticker stats and storing the dataframes into a hashtable

1. Access each ticker based on list fo Tickers input at start.
2. Similar to previous run, save the table contents but place each ticker stats into a hashtable for retrieval later.

In [9]:
```
driver = webdriver.Chrome(service=ChromeService(ChromeDriverManager().install()))
driver.maximize_window()

for i in range(len(tickers)):
    driver.get(f'https://finance.yahoo.com/quote/{tickers[i]}/key-statistics?p={tickers[i]}')
    #print(f'https://finance.yahoo.com/quote/{tickers[i]}/key-statistics?p={tickers[i]}')

    dfs = []

    tables = driver.find_elements(by='xpath', value='//*[@id="Col1-0-KeyStatistics-Proxy"]/secti
```

```
    for table in tables:
        spec_item = []
        for row in table.find_elements(by='xpath', value=".//tr"):
            box = row.find_elements(by='xpath', value = './/td')
            spec_item.append(box[1].get_attribute('textContent'))
        df = pd.DataFrame({f"{tickers[i]}" :spec_item})
        dfs.append(df)

    dummy_df = pd.DataFrame()
    for n in range(len(dfs)):
        dummy_df = pd.concat([dummy_df, dfs[n]])

    hashtable[f'{tickers[i]}_stats'] = dummy_df
    hashtable[f'{tickers[i]}_stats'].reset_index(inplace=True)
    hashtable[f'{tickers[i]}_stats'].drop(['index'],axis = 1, inplace=True)

driver.quit()
hashtable[f'{tickers[i]}_stats'].head()
```

Out[9]:

	SHOP
0	38.19B
1	32.43B
2	N/A
3	1.43k
4	N/A

Combining all the ticker stats

In [10]:

```
for i in range(len(tickers)):
    all_stats_df = pd.concat([all_stats_df, hashtable[f'{tickers[i]}_stats']], axis = 1)

all_stats_df
```

Out[10]:

	Metrics	CRWD	TEAM	ZS	NET	SNOW	HUBS	U	DDOG	DOCN	
0	Market Cap (intraday)	39.94B	59.25B	20.59B	19.23B	54.88B	15.14B	12.17B	30.68B	3.72B	24
1	Enterprise Value	38.40B	59.11B	19.93B	19.15B	51.17B	14.64B	12.23B	29.78B	4.02B	18
2	Trailing P/E	N/A	N/A	N/A	N/A	N/A	N/A	N/A	9.69k	N/A	
3	Forward P/E	144.93	140.85	136.99	625.00	1.11k	98.04	N/A	92.59	34.72	
4	PEG Ratio (5 yr expected)	3.55	4.63	2.73	N/A	5.55	2.56	N/A	1.58	N/A	
5	Price/Sales (ttm)	21.60	21.01	20.91	23.19	32.58	9.80	9.73	23.30	8.30	
6	Price/Book (mrq)	32.75	176.56	38.73	33.12	10.11	17.38	5.33	25.25	77.13	
7	Enterprise Value/Revenue	20.94	21.09	20.55	23.57	31.24	9.60	10.03	21.81	8.17	
8	Enterprise Value/EBITDA	-988.16	-132.89	-74.06	-119.54	-76.57	-618.97	-23.71	469.06	62.70	
9	Beta (5Y Monthly)	1.28	1.01	1.03	1.22	N/A	1.60	N/A	1.15	N/A	
10	52-Week Change	-35.45%	-39.38%	-48.24%	-54.25%	-46.93%	-54.37%	-71.17%	-34.47%	-43.73%	-72
11	S&P500 52-Week Change	-13.02%	-13.02%	-13.02%	-13.02%	-13.02%	-13.02%	-13.02%	-13.02%	-13.02%	-13
12	52 Week High	298.48	483.13	376.11	221.64	405.00	866.00	210.00	199.68	133.40	3(
13	52 Week Low	130.00	159.54	125.12	38.96	110.26	257.21	29.09	81.12	30.05	
14	50-Day Moving Average	185.83	230.12	160.78	58.29	158.42	326.62	42.95	103.28	42.73	1(
15	200-Day Moving Average	188.87	269.36	216.30	91.92	218.81	448.43	82.77	129.62	55.11	1
16	Avg Vol (3 month)	3.56M	1.85M	2.01M	5.18M	7.28M	694.73k	9.71M	4.75M	1.71M	5
17	Avg Vol (10 day)	5.32M	1.31M	1.86M	3.31M	10.95M	530.73k	5.57M	3.67M	1.12M	8
18	Shares Outstanding	213.42M	144.82M	141.86M	283.02M	318.1M	48.02M	298.16M	290.82M	96.93M	250
19	Implied Shares Outstanding	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	
20	Float	210.51M	144.62M	84.22M	277.47M	265.64M	45.65M	189.42M	251.1M	44.22M	217
21	% Held by Insiders	1.37%	0.32%	40.69%	6.17%	9.51%	4.91%	5.83%	9.68%	28.79%	12
22	% Held by Institutions	77.51%	90.71%	47.66%	88.90%	67.70%	93.95%	77.84%	80.88%	53.76%	62
23	Shares Short	12.9M	3.66M	6.35M	13.14M	12.15M	2.51M	33.54M	10.55M	8.84M	11
24	Short Ratio	4.12	1.65	3.26	2.28	2.1	3.1	2.86	2.02	5.66	

	Metrics	CRWD	TEAM	ZS	NET	SNOW	HUBS	U	DDOG	DOCN	
25	Short % of Float	6.14%	2.90%	7.53%	5.88%	3.97%	6.16%	15.97%	3.73%	12.77%	4
26	Short % of Shares Outstanding	5.56%	1.44%	4.47%	4.01%	3.82%	5.23%	11.25%	3.33%	9.12%	3
27	Shares Short	11.77M	3.36M	5.72M	16.95M	12.94M	2.6M	34.18M	9.26M	8.68M	10
28	Forward Annual Dividend Rate	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	
29	Forward Annual Dividend Yield	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	
30	Trailing Annual Dividend Rate	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
31	Trailing Annual Dividend Yield	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
32	5 Year Average Dividend Yield	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	
33	Payout Ratio	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
34	Dividend Date	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	
35	Ex-Dividend Date	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	
36	Last Split Factor	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	
37	Last Split Date	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	
38	Fiscal Year Ends	Jan 30, 2022	Jun 29, 2022	Jul 30, 2021	Dec 30, 2021	Jan 30, 2022	Dec 30, 2021	Dec 30, 2021	Dec 30, 2021	Dec 30, 2021	Ja
39	Most Recent Quarter (mrq)	Apr 29, 2022	Jun 29, 2022	Apr 29, 2022	Jun 29, 2022	Jul 30, 2022	Jun 29, 2022	Jun 29, 2022	Jun 29, 2022	Jun 29, 2022	J
40	Profit Margin	-11.08%	-21.91%	-38.52%	-35.66%	-41.25%	-6.28%	-54.01%	0.48%	-7.78%	23
41	Operating Margin (ttm)	-7.87%	-3.80%	-32.10%	-21.13%	-42.96%	-5.67%	-51.39%	0.79%	-6.47%	19
42	Return on Assets (ttm)	-2.40%	-2.11%	-8.49%	-5.48%	-6.72%	-2.52%	-10.44%	0.29%	-1.59%	7
43	Return on Equity (ttm)	-18.26%	-194.80%	-71.88%	-41.94%	-13.00%	-11.43%	-30.98%	0.62%	-8.45%	18
44	Revenue (ttm)	1.64B	2.8B	969.96M	812.63M	1.64B	1.53B	1.22B	1.37B	492.3M	
45	Revenue Per Share (ttm)	7.15	11.06	6.94	2.53	5.26	32.17	4.19	4.37	4.62	
46	Quarterly Revenue Growth (yoy)	61.10%	35.80%	62.60%	53.90%	82.70%	35.70%	8.60%	73.90%	29.00%	7
47	Gross Profit (ttm)	1.07B	2.34B	524.02M	509.29M	760.89M	1.04B	862.33M	794.88M	257.97M	
48	EBITDA	-68.24M	-77.73M	-283.47M	-116.63M	-666.13M	-63.48M	-502.04M	35.63M	49.6M	908

	Metrics	CRWD	TEAM	ZS	NET	SNOW	HUBS	U	DDOG	DOCN	
49	Net Income Avi to Common (ttm)	-181.28M	-614.12M	-373.65M	-289.75M	-675.61M	-95.81M	-658.52M	6.54M	-38.29M	990
50	Diluted EPS (ttm)	-0.79	-2.86	-2.67	-0.83	-2.10	-1.36	-2.10	0.00	-0.32	
51	Quarterly Earnings Growth (yoy)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	-85
52	Total Cash (mrq)	2.15B	1.47B	1.66B	1.64B	3.95B	1.25B	1.75B	1.7B	1.17B	
53	Total Cash Per Share (mrq)	9.28	5.77	11.69	5.01	12.42	25.95	5.88	5.38	12.03	
54	Total Debt (mrq)	772.05M	1.31B	1.01B	1.57B	241.76M	746.21M	1.82B	811.29M	1.47B	97
55	Total Debt/Equity (mrq)	69.12	391.68	189.10	270.03	4.45	85.63	79.59	66.77	3,037.97	
56	Current Ratio (mrq)	1.83	1.19	2.24	5.45	3.21	2.31	3.44	3.31	16.59	
57	Book Value Per Share (mrq)	4.75	1.32	3.75	1.77	17.05	18.19	7.66	3.84	0.50	
58	Operating Cash Flow (ttm)	642.21M	883.5M	263.51M	36.48M	343.48M	260.98M	62.54M	403.52M	148.57M	
59	Levered Free Cash Flow (ttm)	588.47M	253.75M	425.37M	91.88M	608.1M	225.78M	168.05M	353.84M	56.85M	

Further improvements

To convert values in table into proper floats and datetime formats for possible ML applications.

Exporting to a dated csv file

In [11]:
```
current_datetime = date.today().isoformat()
all_stats_df.to_csv(f'saas_{current_datetime}.csv')
print(f'Saved to file: saas_{current_datetime}.csv')
```

```
Saved to file: saas_2022-09-08.csv
```