# 1. Abrindo Bibliotecas

### Bibliotecas genérias

In [18]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

### As seguintes bibliotecas serão alvo do nosso estudo:

In [19]:
```
from pandas_datareader import data
import quandl
import cufflinks as cf
import yahoofinancials
```

A intenção desse estudo é trazer dados úteis que podem esclarecer um pouco mais o que aconteceu com GME na situação do short squeeze

Conceito de Data Hedging https://www.investopedia.com/terms/d/deltahedging.asp (https://www.investopedia.com/terms/d/deltahedging.asp)

O que é o Short Interest?

https://www.investopedia.com/articles/01/082201.asp (https://www.investopedia.com/articles/01/082201.asp)

# 2. Cufflinks

In [2]:
```
cf.set_config_file(theme='pearl',sharing='public',offline=True)
```

In [5]:
```
gme = data.DataReader('GME', data_source='yahoo', start='2020-12-01')
```

In [6]:
```
gme.iplot(kind="candle",
                          keys=["Open", "High", "Low", "Close"],
                          rangeslider=True
                          )
```

In [7]:

```
qf=cf.QuantFig(gme,title='First Quant Figure',legend='top',name='GS')
qf.add_bollinger_bands()
qf.iplot()
```

In [8]:

```
qf=cf.QuantFig(gme,title='First Quant Figure',legend='top',name='GS')
qf.add_bollinger_bands()
qf.add_volume()
qf.iplot()
```

In [158]:
```
qf=cf.QuantFig(gme,title='GME Quant Figure',legend='top',name='GS')

qf.add_adx()
qf.add_cci()
qf.add_dmi()
qf.add_ema()

qf.iplot()
```

# 3. Quandl

A Quandl será usada aqui para capturar dados do Volume de Ações Vendidas (Short Sale Volume) obtidos da FINRA.

O que é isso?

Seguindo uma norma da SEC, a FINRA disponibiliza publicamente dados diários do volume de papéis vendidtos (short) de negociações de ações não listadas em exchanges (over the counter securities), reportadas ao OTC Reporting Facility.

Esses dados fornecem o volume agregado por ação de todas as negociações vendidas executadas e reportadas à OFR durante o horário de pregão.

Para saber mais sobre os dados: https://www.finra.org/filing-reporting/orf/orf-regulation-sho (https://www.finra.org/filing-reporting/orf/orf-regulation-sho)

Para saber mais sobre a OFR: https://www.finra.org/filing-reporting/over-the-counter-reporting-facility-orf#:~:text=The%20OTC%20Reporting%20Facility%20 (https://www.finra.org/filing-reporting/over-the-counter-reporting-facility-orf#:~:text=The%20OTC%20Reporting%20Facility%20)(ORF,dissemination%20of%20last%20sale%20reports.

Documentação da FINRA na Quandl

https://www.quandl.com/data/FINRA/FNYX_GME-FINRA-NYSE-TRF-Short-Interest-GME (https://www.quandl.com/data/FINRA/FNYX_GME-FINRA-NYSE-TRF-Short-Interest-GME)

Agora que você já obteve seu token de acesso a Quandl, vamos salvá-lo num arquivo .txt de nome "senha"

A seguir, vamos abrir esse arquivo, e chamar de "token" o seu conteúdo

Chave da Quandl vem num formato parecido com esse:

In [5]:
```
f = open("senha.txt", "r")

token = f.read()
```

In [6]:
```
token
```

Out[6]: '77TfLxTY9XzxycV_uzSe'

In [8]:
```
quandl.ApiConfig.api_key = token
```

In [5]:
```
quandl.ApiConfig.api_key = "77TfLxTY9XzxycV_uzSe"
```

Note que tanto esse resultado quanto o do S&P 500 são consolidados por mês

In [21]:
```
finra = quandl.get("FINRA/FNYX_GME", trim_start = "2020-08-01")
```

In [22]:
```
finra.head()
```

Out[22]:

| Date | ShortVolume | ShortExemptVolume | TotalVolume |
|---|---|---|---|
| **2020-08-03** | 28678.0 | 0.0 | 167741.0 |
| **2020-08-04** | 434953.0 | 0.0 | 1411726.0 |
| **2020-08-05** | 205375.0 | 0.0 | 459608.0 |
| **2020-08-06** | 69919.0 | 0.0 | 199033.0 |
| **2020-08-07** | 143366.0 | 0.0 | 339322.0 |

# Explicar

- O que é a FINRA
- O que são Market Makers
- O que é hedge
- O que são calls - opções de compra
- Falar que nos EUA a margem precisa ser respeitada, se não o vendido (short seller) precisa comprar o papel
- O que é o short interest?

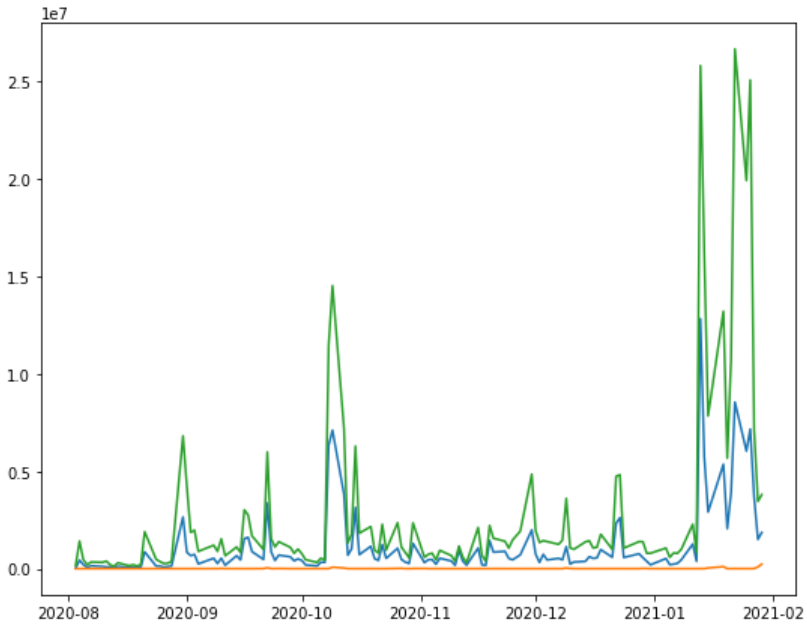
In [ ]:
```
# Procurar dados de compra do papel e hold por parte dos acionistas

# Uma opção é mostrar isso através da diminuição do float geral do papel

# Quanto do free float passou a ser 'segurado' pelos acionistas?
```

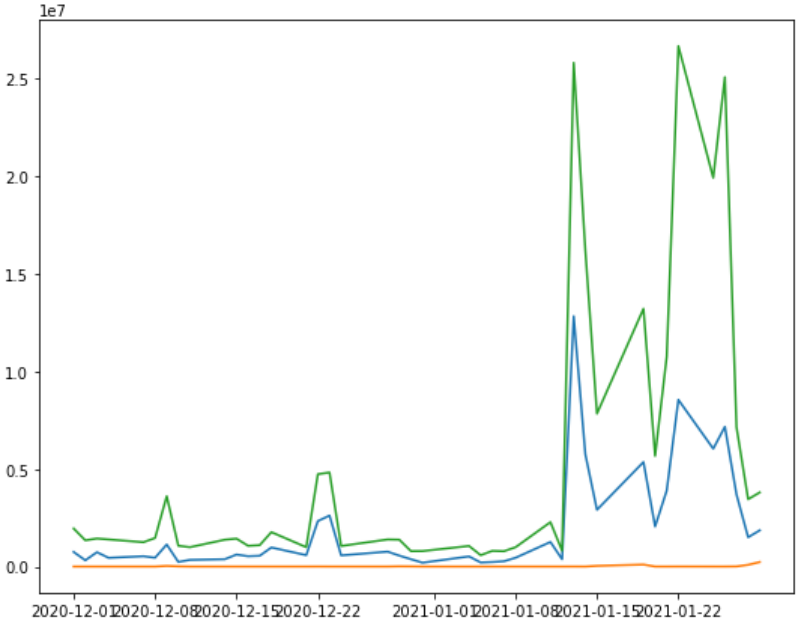
In [20]:
```
finra = quandl.get("FINRA/FNYX_GME", trim_start = "2020-08-01")
plt.figure(figsize=(9,7))
plt.plot(finra)
```

Out[20]:
```
[<matplotlib.lines.Line2D at 0x191eb0d3d00>,
 <matplotlib.lines.Line2D at 0x191eb369550>,
 <matplotlib.lines.Line2D at 0x191eb369490>]
```



In [21]:
```
finra = quandl.get("FINRA/FNYX_GME", trim_start = "2020-12-01")
plt.figure(figsize=(9,7))
plt.plot(finra)
```

Out[21]:
```
[<matplotlib.lines.Line2D at 0x191eb58b490>,
 <matplotlib.lines.Line2D at 0x191eb3e1c70>,
 <matplotlib.lines.Line2D at 0x191eb3e1bb0>]
```




Exemplo de como verificar cotação vs. volume

https://www.reddit.com/r/wallstreetbets/comments/l81luf/hey_sec_heres_your_market_manipulation_rh_working/ (https://www.reddit.com/r/wallstreetbets/comments/l81luf/hey_sec_heres_your_market_manipulation_rh_working/)

In [ ]:
```
```

OBS.: Nesse ponto tentar pegar os dados de short interest de vários outros papéis e compará-los no mesmo gráfico

# 4. yahoofinancials

A biblioteca *yahoofinancials* é uma das melhores formas de se obter dados fundamentalistas direto do Yahoo Finance, em poucas linhas de código

A grande desvantagem é que os dados são retornados em elementos JSON, que por natureza são difcíceis de se realizar o parse, e consequentemente torna o processamento e a análise posterior um pouco mais difícil

Referências para trabalhar com JSONs

https://towardsdatascience.com/how-to-parse-json-data-with-python-pandas-f84fbd0b1025 (https://towardsdatascience.com/how-to-parse-json-data-with-python-pandas-f84fbd0b1025)

https://stackoverflow.com/questions/16729574/how-to-get-a-value-from-a-cell-of-a-dataframe (https://stackoverflow.com/questions/16729574/how-to-get-a-value-from-a-cell-of-a-dataframe)

https://stackoverflow.com/questions/42354001/python-json-object-must-be-str-bytes-or-bytearray-not-dict (https://stackoverflow.com/questions/42354001/python-json-object-must-be-str-bytes-or-bytearray-not-dict)

https://stackoverflow.com/questions/21104592/json-to-pandas-dataframe (https://stackoverflow.com/questions/21104592/json-to-pandas-dataframe)

In [22]:
```
! pip install yahoofinancials
```

```
Collecting yahoofinancials
  Downloading yahoofinancials-1.6.tar.gz (27 kB)
Requirement already satisfied: beautifulsoup4 in c:\programdata\anaconda3\lib\site-packages (from yahoofinancials)
(4.9.3)
Requirement already satisfied: pytz in c:\programdata\anaconda3\lib\site-packages (from yahoofinancials) (2020.1)
Requirement already satisfied: soupsieve>1.2; python_version >= "3.0" in c:\programdata\anaconda3\lib\site-packages
(from beautifulsoup4->yahoofinancials) (2.0.1)
Building wheels for collected packages: yahoofinancials
  Building wheel for yahoofinancials (setup.py): started
  Building wheel for yahoofinancials (setup.py): finished with status 'done'
  Created wheel for yahoofinancials: filename=yahoofinancials-1.6-py3-none-any.whl size=15195 sha256=e0c2ac737f8dd5c3
36dc101036356d2cc2b3e1a1ab832d81369b044eec0154c4
  Stored in directory: c:\users\victo\appdata\local\pip\cache\wheels\6a\90\0c\08c7ac2ce60b9ac91529417d471e59244f9f968
48c86f14809
Successfully built yahoofinancials
Installing collected packages: yahoofinancials
Successfully installed yahoofinancials-1.6
```

In [23]:
```
from yahoofinancials import YahooFinancials

ticker = 'GME'
yahoo_financials = YahooFinancials(ticker)
```

### Vamos verificar os dados anuais pra ver se bate com o Yahoo Finance

In [ ]:
```
balance_sheet_data_qt = yahoo_financials.get_financial_stmts('quarterly', 'balance')
income_statement_data_qt = yahoo_financials.get_financial_stmts('quarterly', 'income')
all_statement_data_qt =  yahoo_financials.get_financial_stmts('quarterly', ['income', 'cash', 'balance'])
GME_earnings_data = yahoo_financials.get_stock_earnings_data()
GME_net_income = yahoo_financials.get_net_income()
```

In [132]:
```
balance_sheet_data_qt = yahoo_financials.get_financial_stmts('annual', 'balance')
income_statement_data_qt = yahoo_financials.get_financial_stmts('annual', 'income')

# Outra opção de obter o dado de statment é usando todos os statements: income, cash e balance
all_statement_data_qt =  yahoo_financials.get_financial_stmts('annual', ['income', 'cash', 'balance'])

GME_earnings_data = yahoo_financials.get_stock_earnings_data()
GME_net_income = yahoo_financials.get_net_income()
```

Vamos fazer um teste: que tipo são esses dados nos outputs?

In [70]:
```
type(all_statement_data_qt)
```

Out[70]:
```
dict
```

In [133]:
```
all_statement_data_qt
```

Out[133]:
```
{'incomeStatementHistory': {'GME': [{'2020-02-01': {'researchDevelopment': None,
     'effectOfAccountingCharges': None,
     'incomeBeforeTax': -426800000,
     'minorityInterest': None,
     'netIncome': -470900000,
     'sellingGeneralAdministrative': 1893600000,
     'grossProfit': 1908700000,
     'ebit': 15100000,
     'operatingIncome': 15100000,
     'otherOperatingExpenses': None,
     'interestExpense': -38500000,
     'extraordinaryItems': None,
     'nonRecurring': None,
     'otherItems': None,
     'incomeTaxExpense': 37600000,
     'totalRevenue': 6466000000,
     'totalOperatingExpenses': 6450900000,
     'costOfRevenue': 4557300000,
     'totalOtherIncomeExpenseNet': -441900000,
```

Verifique que esses dados são do tipo "JSON", por isso vem no formato de dicionário. Por causa disso, teremos que trabalhar com métodos específicos para tratar JSON

As principais funções para se trabalhar com JSON são:

json.loads take a string as input and returns a dictionary as output.

json.dumps take a dictionary as input and returns a string as output.

Além disso, na package pandas, a função pd.json_normalize pega um JSON de input e transforma num pandas Data Frame de output

In [139]:
```
import json
#data = json.dumps(data)
#data = json.loads(data)

incomeStatementHistory = pd.json_normalize(all_statement_data_qt['incomeStatementHistory'])
```

In [140]:
```
incomeStatementHistory
```

Out[140]:

| | GME |
|---|---|
| **0** | [{'2020-02-01': {'researchDevelopment': None, ... |

In [143]:
```
dictionary = incomeStatementHistory.iloc[0]

string = list(dictionary[[0]])

stringa = string[0]
```

In [144]:
```
stringa
```

Out[144]:

```
[{'2020-02-01': {'researchDevelopment': None,
   'effectOfAccountingCharges': None,
   'incomeBeforeTax': -426800000,
   'minorityInterest': None,
   'netIncome': -470900000,
   'sellingGeneralAdministrative': 1893600000,
   'grossProfit': 1908700000,
   'ebit': 15100000,
   'operatingIncome': 15100000,
   'otherOperatingExpenses': None,
   'interestExpense': -38500000,
   'extraordinaryItems': None,
   'nonRecurring': None,
   'otherItems': None,
   'incomeTaxExpense': 37600000,
   'totalRevenue': 6466000000,
   'totalOperatingExpenses': 6450900000,
   'costOfRevenue': 4557300000,
   'totalOtherIncomeExpenseNet': -441900000,
   'discontinuedOperations': -6500000,
   'netIncomeFromContinuingOps': -464400000,
   'netIncomeApplicableToCommonShares': -470900000}},
 {'2019-02-02': {'researchDevelopment': None,
   'effectOfAccountingCharges': None,
   'incomeBeforeTax': -753100000,
   'minorityInterest': None,
   'netIncome': -673000000,
   'sellingGeneralAdministrative': 1997200000,
   'grossProfit': 2308100000,
   'ebit': 310900000,
   'operatingIncome': 310900000,
   'otherOperatingExpenses': None,
   'interestExpense': -56800000,
   'extraordinaryItems': None,
   'nonRecurring': None,
   'otherItems': None,
   'incomeTaxExpense': 41700000,
   'totalRevenue': 8285300000,
   'totalOperatingExpenses': 7974400000,
   'costOfRevenue': 5977200000,
   'totalOtherIncomeExpenseNet': -1064000000,
   'discontinuedOperations': 121800000,
   'netIncomeFromContinuingOps': -794800000,
   'netIncomeApplicableToCommonShares': -673000000}},
 {'2018-02-03': {'researchDevelopment': None,
   'effectOfAccountingCharges': None,
   'incomeBeforeTax': 383900000,
   'minorityInterest': None,
   'netIncome': 34700000,
   'sellingGeneralAdministrative': 2040700000,
   'grossProfit': 2484900000,
   'ebit': 444200000,
   'operatingIncome': 444200000,
   'otherOperatingExpenses': None,
   'interestExpense': -56800000,
   'extraordinaryItems': None,
   'nonRecurring': None,
   'otherItems': None,
   'incomeTaxExpense': 153500000,
   'totalRevenue': 8547100000,
   'totalOperatingExpenses': 8102900000,
   'costOfRevenue': 6062200000,
   'totalOtherIncomeExpenseNet': -60300000,
   'discontinuedOperations': -195700000,
   'netIncomeFromContinuingOps': 230400000,
   'netIncomeApplicableToCommonShares': 34700000}},
 {'2017-01-28': {'researchDevelopment': None,
   'effectOfAccountingCharges': None,
   'incomeBeforeTax': 428700000,
   'minorityInterest': None,
   'netIncome': 353200000,
   'sellingGeneralAdministrative': 1861900000,
   'grossProfit': 2499900000,
   'ebit': 501300000,
   'operatingIncome': 501300000,
   'otherOperatingExpenses': None,
   'interestExpense': -53800000,
   'extraordinaryItems': None,
   'nonRecurring': None,
   'otherItems': None,
   'incomeTaxExpense': 124200000,
   'totalRevenue': 7965000000,
   'totalOperatingExpenses': 7463700000,
   'costOfRevenue': 5465100000,
   'totalOtherIncomeExpenseNet': -72600000,
```

```
 'discontinuedOperations': 48700000,
 'netIncomeFromContinuingOps': 304500000,
 'netIncomeApplicableToCommonShares': 353200000}}]
```

Observe que aqui obtivemos uma lista contendo vários dicionários, onde cada trimestre possui os resultados representados em um dicionário

Se quisermos trabalhar com um trimestre ou ano específico, precisaremos filtrar, e só em seguida criar um data frame

In [145]:
```
string_b = stringa[0]
```

In [146]:
```
string_b
```

Out[146]:
```
{'2020-02-01': {'researchDevelopment': None,
  'effectOfAccountingCharges': None,
  'incomeBeforeTax': -426800000,
  'minorityInterest': None,
  'netIncome': -470900000,
  'sellingGeneralAdministrative': 1893600000,
  'grossProfit': 1908700000,
  'ebit': 15100000,
  'operatingIncome': 15100000,
  'otherOperatingExpenses': None,
  'interestExpense': -38500000,
  'extraordinaryItems': None,
  'nonRecurring': None,
  'otherItems': None,
  'incomeTaxExpense': 37600000,
  'totalRevenue': 6466000000,
  'totalOperatingExpenses': 6450900000,
  'costOfRevenue': 4557300000,
  'totalOtherIncomeExpenseNet': -441900000,
  'discontinuedOperations': -6500000,
  'netIncomeFromContinuingOps': -464400000,
  'netIncomeApplicableToCommonShares': -470900000}}
```

Agora enfim podemos transformar isso num data frame e em seguida seguir com a análise

In [147]:
```
string_df = pd.json_normalize(string_b['2020-02-01'])
```

In [148]:
```
string_df
```

Out[148]:

| | researchDevelopment | effectOfAccountingCharges | incomeBeforeTax | minorityInterest | netIncome | sellingGeneralAdministrative | grossProfit | |
|---|---|---|---|---|---|---|---|---|
| **0** | None | None | -426800000 | None | -470900000 | 1893600000 | 1908700000 | 151 |

1 rows × 22 columns

In [ ]:
```

```

In [ ]:
```

```

Uma outra forma mais fácil de obter todas as estatísticas principais de uma só vez:

In [41]:
```
yahoo_financials = YahooFinancials('GME')
print(yahoo_financials.get_key_statistics_data())
```

```
{'GME': {'annualHoldingsTurnover': None, 'enterpriseToRevenue': 4.529, 'beta3Year': None, 'profitMargins': -0.0532399
97, 'enterpriseToEbitda': -180.657, '52WeekChange': 81.27848, 'morningStarRiskRating': None, 'forwardEps': -0.17, 're
venueQuarterlyGrowth': None, 'sharesOutstanding': 69747000, 'fundInceptionDate': '-', 'annualReportExpenseRatio': Non
e, 'totalAssets': None, 'bookValue': 5.095, 'sharesShort': 61782730, 'sharesPercentSharesOut': 0.8858, 'fundFamily':
None, 'lastFiscalYearEnd': 1580515200, 'heldPercentInstitutions': 1.2204499, 'netIncomeToCommon': -270000000, 'traili
ngEps': -4.224, 'lastDividendValue': 0.38, 'SandP52WeekChange': 0.14322305, 'priceToBook': 48.086384, 'heldPercentIns
iders': 0.27334, 'nextFiscalYearEnd': 1643673600, 'yield': None, 'mostRecentQuarter': 1604102400, 'shortRatio': 2.81,
'sharesShortPreviousMonthDate': '2020-12-15', 'floatShares': 46888789, 'beta': 1.433298, 'enterpriseValue': 233769615
36, 'priceHint': 2, 'threeYearAverageReturn': None, 'lastSplitDate': '2007-03-19', 'lastSplitFactor': '2:1', 'legalTy
pe': None, 'lastDividendDate': '2019-03-14', 'morningStarOverallRating': None, 'earningsQuarterlyGrowth': None, 'pric
eToSalesTrailing12Months': None, 'dateShortInterest': 1610668800, 'pegRatio': 0.18, 'ytdReturn': None, 'forwardPE': -
1441.1771, 'maxAge': 1, 'lastCapGain': None, 'shortPercentOfFloat': 2.2642, 'sharesShortPriorMonth': 68127116, 'impli
edSharesOutstanding': None, 'category': None, 'fiveYearAverageReturn': None}}
```

In [150]:
```
key_statistics = yahoo_financials.get_key_statistics_data()
```

Vamos transformar esse JSON em um data frame?

In [151]:
```
key_statistics_df = pd.json_normalize(key_statistics['GME'])
```

In [157]:
```
key_statistics_df.transpose()
```

Out[157]:

| | 0 |
|---|---|
| **annualHoldingsTurnover** | None |
| **enterpriseToRevenue** | 4.529 |
| **beta3Year** | None |
| **profitMargins** | -0.05324 |
| **enterpriseToEbitda** | -180.657 |
| **52WeekChange** | 81.2785 |
| **morningStarRiskRating** | None |
| **forwardEps** | -0.17 |
| **revenueQuarterlyGrowth** | None |
| **sharesOutstanding** | 69747000 |
| **fundInceptionDate** | - |
| **annualReportExpenseRatio** | None |
| **totalAssets** | None |
| **bookValue** | 5.095 |
| **sharesShort** | 61782730 |
| **sharesPercentSharesOut** | 0.8858 |
| **fundFamily** | None |
| **lastFiscalYearEnd** | 1580515200 |
| **heldPercentInstitutions** | 1.22045 |
| **netIncomeToCommon** | -270000000 |
| **trailingEps** | -4.224 |
| **lastDividendValue** | 0.38 |
| **SandP52WeekChange** | 0.143223 |
| **priceToBook** | 48.0864 |
| **heldPercentInsiders** | 0.27334 |
| **nextFiscalYearEnd** | 1643673600 |
| **yield** | None |
| **mostRecentQuarter** | 1604102400 |
| **shortRatio** | 2.81 |
| **sharesShortPreviousMonthDate** | 2020-12-15 |
| **floatShares** | 46888789 |
| **beta** | 1.4333 |
| **enterpriseValue** | 23376961536 |
| **priceHint** | 2 |
| **threeYearAverageReturn** | None |
| **lastSplitDate** | 2007-03-19 |
| **lastSplitFactor** | 2:1 |
| **legalType** | None |
| **lastDividendDate** | 2019-03-14 |
| **morningStarOverallRating** | None |
| **earningsQuarterlyGrowth** | None |
| **priceToSalesTrailing12Months** | None |
| **dateShortInterest** | 1610668800 |
| **pegRatio** | 0.18 |
| **ytdReturn** | None |
| **forwardPE** | -1441.18 |
| **maxAge** | 1 |
| **lastCapGain** | None |
| **shortPercentOfFloat** | 2.2642 |
| **sharesShortPriorMonth** | 68127116 |
| **impliedSharesOutstanding** | None |
| **category** | None |
| **fiveYearAverageReturn** | None |

# 5. Requests

Vamos usar agora um pouco de webscraping para obter as informações de short interest direto da internet?

In [10]:
```
import requests

import pandas as pd
```

In [13]:
```
url = "https://www.marketbeat.com/short-interest/"
```

In [14]:
```
response = requests.request("POST", url)

tables = pd.read_html(response.text)
```

In [15]:
```
tables
```

Out[15]:
```
[                                          Company  \
 0                          CHTRCharter Communications
 1                                      VIACViacomCBS
 2                                            AONAON
 3                                  ADIAnalog Devices
 4                                        MRNAModerna
 5                                DDDuPont de Nemours
 6                             PLTRPalantir Technologies
 7                                          SNAPSnap
 8              IFFInternational Flavors & Fragrances
 9                                       BYNDBeyond Meat
 10  (adsbygoogle = window.adsbygoogle || []).push(...
 11                                        KRThe Kroger
 12                                       CLXThe Clorox
 13                                       SNOWSnowflake
 14                               BBBYBed Bath & Beyond
 15                                        SPWRSunPower
 16                                     DISCADiscovery
 17                               PTONPeloton Interactive
```

In [16]:
```
pld_table = tables[0]
```

In [17]:
```
pld_table
```

Out[17]:

| | Company | Shares Sold Short (1/15/2021) | Dollar VolumeSold Short | Shares Sold Short (12/31/2020) | Change | % Change | % Float |
|---|---|---|---|---|---|---|---|
| 0 | CHTRCharter Communications | 10620000 | $6.95 billion | 10630000 | -10000 | -0.1% | 7.9% |
| 1 | VIACViacomCBS | 121250000 | $6.14 billion | 117920000 | 3330000 | 2.8% | 22.4% |
| 2 | AONAON | 22570000 | $4.66 billion | 21470000 | 1100000 | 5.1% | 10.0% |
| 3 | ADIAnalog Devices | 30400000 | $4.54 billion | 28260000 | 2140000 | 7.6% | 8.3% |
| 4 | MRNAModerna | 24800000 | $3.96 billion | 23390000 | 1410000 | 6.0% | 7.6% |
| 5 | DDDuPont de Nemours | 46490000 | $3.77 billion | 23670000 | 22820000 | 96.4% | 6.4% |
| 6 | PLTRPalantir Technologies | 98950000 | $3.53 billion | 90740000 | 8210000 | 0.0% | 9.7% |
| 7 | SNAPSnap | 65420000 | $3.48 billion | 63160000 | 2260000 | 3.6% | 7.0% |
| 8 | IFFInternational Flavors & Fragrances | 29980000 | $3.45 billion | 19880000 | 10100000 | 50.8% | 28.1% |
| 9 | BYNDBeyond Meat | 15440000 | $2.78 billion | 13690000 | 1750000 | 12.8% | 42.9% |
| 10 | (adsbygoogle = window.adsbygoogle \|\| []).push(... | (adsbygoogle = window.adsbygoogle \|\| []).push(... | (adsbygoogle = window.adsbygoogle \|\| []).push(... | (adsbygoogle = window.adsbygoogle \|\| []).push(... | (adsbygoogle = window.adsbygoogle \|\| []).push(... | (adsbygoogle = window.adsbygoogle \|\| []).push(... | (adsbygoogle = window.adsbygoogle \|\| []).push(... |
| 11 | KRThe Kroger | 76140000 | $2.69 billion | 73390000 | 2750000 | 3.8% | 10.1% |
| 12 | CLXThe Clorox | 12730000 | $2.67 billion | 12280000 | 450000 | 3.7% | 10.1% |
| 13 | SNOWSnowflake | 9270000 | $2.53 billion | 10650000 | -1380000 | 0.0% | 19.9% |
| 14 | BBBYBed Bath & Beyond | 74890000 | $2.52 billion | 76180000 | -1290000 | -1.7% | 65.5% |
| 15 | SPWRSunPower | 46520000 | $2.46 billion | 43350000 | 3170000 | 7.3% | 57.5% |
| 16 | DISCADiscovery | 58070000 | $2.38 billion | 54610000 | 3460000 | 6.3% | 37.6% |
| 17 | PTONPeloton Interactive | 15170000 | $2.21 billion | 17290000 | -2120000 | -12.3% | 6.2% |
| 18 | ATUSAltice USA | 57460000 | $2.10 billion | 51610000 | 5850000 | 11.3% | 23.8% |
| 19 | LGNDLigand Pharmaceuticals | 10010000 | $1.92 billion | 9910000 | 100000 | 1.0% | 64.9% |
| 20 | DASHDoorDash | 9810000 | $1.80 billion | 8820000 | 990000 | 0.0% | 7.9% |
| 21 | ABNBAirbnb | 9180000 | $1.72 billion | 7960000 | 1220000 | 0.0% | 11.3% |
| 22 | PANWPalo Alto Networks | 4810000 | $1.70 billion | 5360000 | -550000 | -10.3% | 5.1% |
| 23 | SPCEVirgin Galactic | 38600000 | $1.66 billion | 43740000 | -5140000 | -11.8% | 72.0% |
| 24 | IRMIron Mountain | 49800000 | $1.66 billion | 49690000 | 110000 | 0.2% | 17.5% |
| 25 | DDOGDatadog | 16110000 | $1.61 billion | 15350000 | 760000 | 5.0% | 8.8% |
| 26 | PENPenumbra | 6060000 | $1.60 billion | 5930000 | 130000 | 2.2% | 17.8% |
| 27 | FUTUFutu | 16060000 | $1.59 billion | 13600000 | 2460000 | 18.1% | 25.7% |
| 28 | MPCMarathon Petroleum | 35060000 | $1.57 billion | 31890000 | 3170000 | 9.9% | 5.4% |
| 29 | WLTWWillis Towers Watson Public | 7570000 | $1.56 billion | 6620000 | 950000 | 14.4% | 5.9% |
| 30 | FUBOfuboTV | 39170000 | $1.56 billion | 33860000 | 5310000 | 0.0% | 71.9% |
| 31 | MACThe Macerich | 78540000 | $1.49 billion | 80640000 | -2100000 | -2.6% | 56.9% |
| 32 | CHKPCheck Point Software Technologies | 10870000 | $1.44 billion | 11430000 | -560000 | -4.9% | 9.9% |
| 33 | RHRH | 2920000 | $1.39 billion | 2780000 | 140000 | 5.0% | 16.0% |
| 34 | MMacy's | 88790000 | $1.38 billion | 108650000 | -19860000 | -18.3% | 28.8% |
| 35 | EXPEExpedia Group | 10860000 | $1.38 billion | 13050000 | -2190000 | -16.8% | 9.2% |
| 36 | FTCHFarfetch | 21740000 | $1.32 billion | 21020000 | 720000 | 3.4% | 10.5% |
| 37 | STXSeagate Technology | 19700000 | $1.31 billion | 16980000 | 2720000 | 16.0% | 7.7% |
| 38 | VMWVMware | 9210000 | $1.29 billion | 9280000 | -70000 | -0.8% | 11.7% |
| 39 | IRBTiRobot | 10550000 | $1.29 billion | 10260000 | 290000 | 2.8% | 38.5% |
| 40 | CGCCanopy Growth | 31910000 | $1.29 billion | 32380000 | -470000 | -1.5% | 14.0% |
| 41 | SIRISirius XM | 198670000 | $1.27 billion | 191080000 | 7590000 | 4.0% | 18.4% |
| 42 | LMNDLemonade | 8490000 | $1.27 billion | 8280000 | 210000 | 0.0% | 30.9% |
| 43 | NUANNuance Communications | 27360000 | $1.26 billion | 24710000 | 2650000 | 10.7% | 9.7% |

| | Company | Shares Sold Short (1/15/2021) | Dollar VolumeSold Short | Shares Sold Short (12/31/2020) | Change | % Change | % Float |
|---|---|---|---|---|---|---|---|
| **44** | KEYSKeysight Technologies | 8590000 | $1.25 billion | 9410000 | -820000 | -8.7% | 5.3% |
| **45** | ENPHEnphase Energy | 6430000 | $1.25 billion | 8300000 | -1870000 | -22.5% | 5.5% |
| **46** | ESTCElastic | 8050000 | $1.23 billion | 7520000 | 530000 | 7.1% | 12.7% |
| **47** | ONON Semiconductor | 34890000 | $1.23 billion | 31090000 | 3800000 | 12.2% | 8.6% |
| **48** | NKLANikola | 52610000 | $1.22 billion | 54440000 | -1830000 | 0.0% | 35.7% |
| **49** | OLLIOllie's Bargain Outlet | 12190000 | $1.19 billion | 12730000 | -540000 | -4.2% | 21.5% |
| **50** | LYFTLyft | 25510000 | $1.18 billion | 24520000 | 990000 | 4.0% | 10.4% |
| **51** | (adsbygoogle = window.adsbygoogle \|\| []).push(... | (adsbygoogle = window.adsbygoogle \|\| []).push(... | (adsbygoogle = window.adsbygoogle \|\| []).push(... | (adsbygoogle = window.adsbygoogle \|\| []).push(... | (adsbygoogle = window.adsbygoogle \|\| []).push(... | (adsbygoogle = window.adsbygoogle \|\| []).push(... | (adsbygoogle = window.adsbygoogle \|\| []).push(... |

In [ ]: