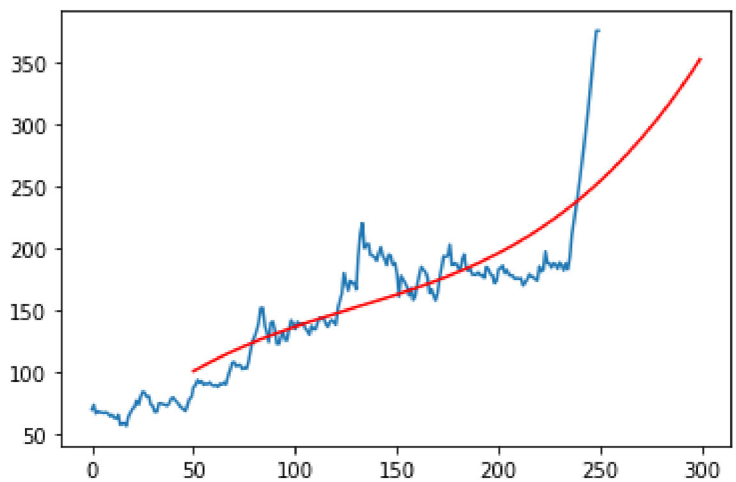
In [1]:
```
import yfinance as yf
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from __future__ import annotations
import datetime
```

In [3]:
```
stock_data = yf.Ticker("TEJASNET.NS")
stock_data = stock_data.history(period = '1y')[['Open']]
MINDT = stock_data.reset_index(drop = True)
```

In [4]:
```
x = MINDT.index
y= MINDT.Open
model = np.polyfit(x,y,3)
predict = np.poly1d(model)
x_pol_reg = range(50,300)
y_pol_reg = predict(x_pol_reg)
plt.plot(x,y)
plt.plot(x_pol_reg, y_pol_reg, c='r')
```

Out[4]:
```
[<matplotlib.lines.Line2D at 0x7ff2b91759a0>]
```



import yfinance as yf import pandas as pd tesla = yf.Ticker('TSLA') tesla = tesla.history(period="max") tesla = tesla[['Open']] nio = yf.Ticker('NIO') nio = nio.history(period="max") nio = nio[['Open']] stonks = tesla.merge(nio, how = 'outer', left_index = True, right_index = True) stonks.columns = ['TSLA', 'NIO'] stonks

tesla = yf.Ticker('TSLA')

MY COLLUMMNS FOR STUDY

tesla = yf.Ticker('TSLA') tesla=tesla.history(period="max") tesla = tesla[['Open','Close','Volume']] tesla

CHOOSING STOCK LIST

In [36]:
```
stock_list = ['TSLA','NIO','IQQH.F','BTC-USD','BTC-INR',
              'ETH-USD','LTC-USD','AMZN','TWTR','FB','SQ','PYPL','BRK-A','CSPX.AS
              'EURINR=X','^CNXIT','HAPPSTMNDS.NS','MPHASIS.NS','WIPRO.NS','MINDTR
              'TCS.NS','TECHM.NS','ASHOKLEY.NS','BOSCHLTD.NS','MARUTI.NS','TATAMO
              'EXIDEIND.NS','AMARAJABAT.NS','BALKRISIND.NS','MRF.NS','NATIONALUM.
              'VEDL.NS','TATASTEEL.NS','JINDALSTEL.NS','TATASTEEL.NS','JSWSTEEL.N
```

HALF DONE LIST

In [37]:
```
stock_names = ['TESLA','NIO','CLEAN_ENERGE_ETF','BIT_COIN','BIT_IND','ETH_USD','L
              'SQUARE','PAYPAL','BERKSHR','S&P500','GOLD','SILV','CRUDE','UARMOR'
               'NIFTY_IT','HAPPYMIND','MPHASIS','WIPRO','MINDTREE','INFY','COFORG
```

basic checks

In [38]:
```
len(stock_names), len(stock_list)
```

Out[38]:
```
(28, 50)
```

DEFINE A COLLECTION AND CLEANING

In [39]:
```
def collect_clean(lists: list[str]) :
    res=[]
    for stock in lists:
        idx=stock.find(".")
        if idx == -1:
            res.append(stock)
        else:
            res.append(stock[:idx])
    return res
```

In [40]:
```
temp_list = collect_clean(stock_list[len(stock_names):])
```

In [41]:
```
stock_names = stock_names+temp_list
```

In [42]:
```
len(stock_names), len(stock_list)
```

Out[42]:
```
(50, 50)
```

In [43]:
```
stock_dict = {i:j for i,j in zip(stock_names,stock_list)}
```

In [44]:
```
stock_dict
```

Out[44]:
```
{'TESLA': 'TSLA',
 'NIO': 'NIO',
 'CLEAN_ENERGE_ETF': 'IQQH.F',
 'BIT_COIN': 'BTC-USD',
 'BIT_IND': 'BTC-INR',
 'ETH_USD': 'ETH-USD',
 'LTC_USD': 'LTC-USD',
 'AMAZON': 'AMZN',
 'TWITTER': 'TWTR',
 'FACEBOOK': 'FB',
 'SQUARE': 'SQ',
 'PAYPAL': 'PYPL',
 'BERKSHR': 'BRK-A',
 'S&P500': 'CSPX.AS',
 'GOLD': 'GC=F',
 'SILV': 'SI=F',
 'CRUDE': 'CL=F',
 'UARMOR': 'UA',
 'GARTNER': 'IT',
 'USD_N_RS': 'INR=X',
 'EUROINR': 'EURINR=X',
 'NIFTY_IT': '^CNXIT',
 'HAPPYMIND': 'HAPPSTMNDS.NS',
 'MPHASIS': 'MPHASIS.NS',
 'WIPRO': 'WIPRO.NS',
 'MINDTREE': 'MINDTREE.NS',
 'INFY': 'INFY.NS',
 'COFORGE': 'COFORGE.NS',
 'HCLTECH': 'HCLTECH.NS',
 'TCS': 'TCS.NS',
 'TECHM': 'TECHM.NS',
 'ASHOKLEY': 'ASHOKLEY.NS',
 'BOSCHLTD': 'BOSCHLTD.NS',
 'MARUTI': 'MARUTI.NS',
 'TATAMOTORS': 'TATAMOTORS.NS',
 'ESCORTS': 'ESCORTS.NS',
 'BAJAJ-AUTO': 'BAJAJ-AUTO.NS',
 'EXIDEIND': 'EXIDEIND.NS',
 'AMARAJABAT': 'AMARAJABAT.NS',
 'BALKRISIND': 'BALKRISIND.NS',
 'MRF': 'MRF.NS',
 'NATIONALUM': 'NATIONALUM.NS',
 'NMDC': 'NMDC.NS',
 'COALINDIA': 'COALINDIA.NS',
 'VEDL': 'VEDL.NS',
 'TATASTEEL': 'TATASTEEL.NS',
 'JINDALSTEL': 'JINDALSTEL.NS',
 'JSWSTEEL': 'JSWSTEEL.NS',
 'SAIL': 'SAIL.NS'}
```

stock_dict.items()

append to a new DataFrame

In [ ]:
```
master_df = pd.DataFrame()
for key,val in stock_dict.items():
    df = yf.Ticker(val)
    df=df.history(period="max")
    df.dropna(inplace=True)
    #df = df[['Open','Close','Volume']]
    #master_df[key+'_Open'] = df['Open']
    master_df[key+'_Close'] = df['Close']
    #master_df[key+'_Volume'] = df['Volume']
    #print(f"{key} is done")
```

In [55]:
```
base = datetime.date.today()
date_list = [base - datetime.timedelta(days=x) for x in range(1)]
date_list.sort()
stocks_master_df = pd.DataFrame(index= date_list)

for i in stock_list:
    i = yf.Ticker(i)
    i = i.history(period='max')
    i= i[['Close']]
    stocks_master_df = stocks_master_df.merge(i, how = 'outer', left_index = True

stocks_master_df.columns = stock_list
```

quality check

In [57]:
```
stocks_master_df = stocks_master_df[~stocks_master_df.index.duplicated()]
```

In [79]:
```
stocks_master_df[stocks_master_df.index.duplicated()]
```

Out[79]:

TSLA	NIO	IQQH.F	BTC-USD	BTC-INR	ETH-USD	LTC-USD	AMZN	TWTR	FB	...	MRF.NS	NATIONALUM.NS

0 rows × 50 columns

Remove Nan values for generated during weekend

In [59]:
```
stocks_master_df = stocks_master_df[stocks_master_df['TSLA'].notna()]
```

In [80]:
```
#stocks_master_df.describe()
stocks_master_df.tail()
```

Out[80]:

TWTR	FB	...	MRF.NS	NATIONALUM.NS	NMDC.NS	COALINDIA.NS	VEDL.NS	TATASTEEL.N
).868752	0.971003	...	0.811423	1.000000	0.848574	0.243568	0.955280	0.9907
).838730	0.972521	...	0.805916	0.970371	0.837333	0.247101	0.963443	0.9771
).833071	0.967178	...	0.815658	0.829015	0.811544	0.231614	0.947117	0.9643
).812166	0.965828	...	0.802291	0.774077	0.771538	0.225908	0.930081	0.9346
).808708	0.962537	...	0.788898	0.824077	0.794021	0.235417	1.000000	0.9750

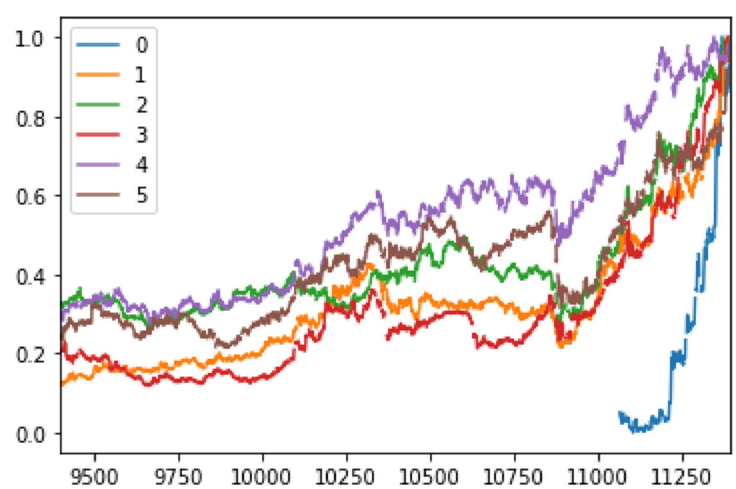
Normalization and standardization

In [50]:
```
from sklearn.preprocessing import MaxAbsScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
```

Method #1: Min max scaler

In [23]:
```
scaler = MinMaxScaler()
pd.DataFrame(scaler.fit_transform(stocks_master_df[['HAPPSTMNDS.NS','MPHASIS.NS',
'TCS.NS','TECHM.NS']])).tail(2000).plot()
```

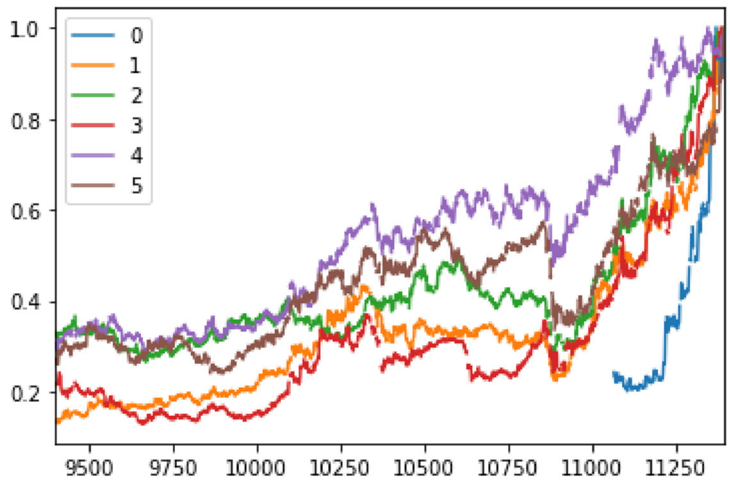
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff2a7246a90>



Method #2 : MaxAbs Scalar

In [24]:
```
abs_scaler = MaxAbsScaler()
pd.DataFrame(abs_scaler.fit_transform(stocks_master_df[['HAPPSTMNDS.NS','MPHASIS.
'TCS.NS','TECHM.NS']])).tail(2000).plot()
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff2a718fe80>

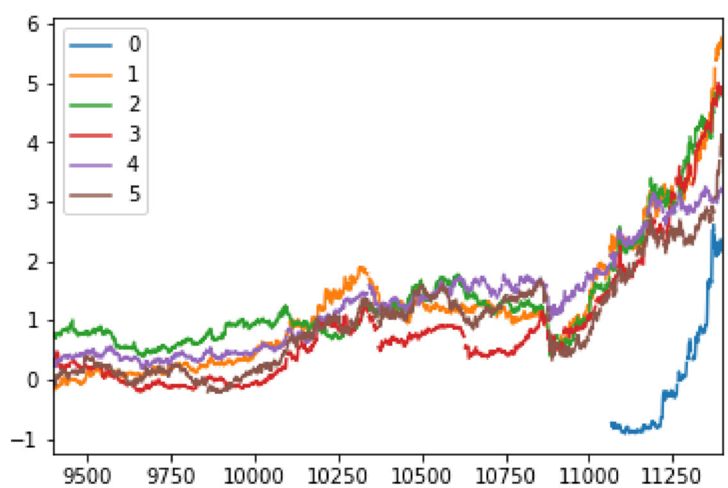


Method #3 : Z-score Method(standardization)

The number of Stddev that a given point is from the mean. It can be usefull but not for this data as it is not normally distributed

In [25]:
```
std_scaler =StandardScaler()
pd.DataFrame(std_scaler.fit_transform(stocks_master_df[['HAPPSTMNDS.NS','MPHASIS.
'TCS.NS','TECHM.NS']])).tail(2000).plot()
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff2a71594f0>

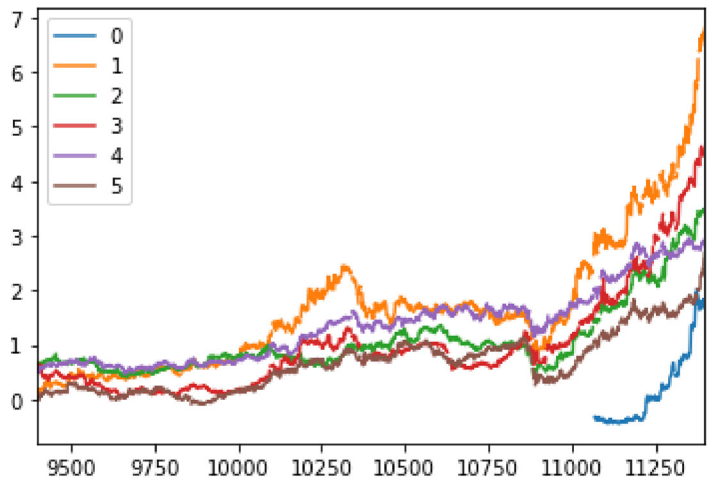


Method #4 Robust Scaler

Not many outliers so not that useful in this data set

In [28]:
```
rob_scaler = RobustScaler()
pd.DataFrame(rob_scaler.fit_transform(stocks_master_df[['HAPPSTMNDS.NS','MPHASIS.
'TCS.NS','TECHM.NS']])).tail(2000).plot()
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff2a7110550>



Going with min- max normalization

In [64]:
```
scaler = MinMaxScaler()
stocks_master_df = pd.DataFrame(scaler.fit_transform(stocks_master_df), columns=
```

In [65]:
```
stocks_master_df
```

Out[65]:

	TSLA	NIO	IQQH.F	BTC-USD	BTC-INR	ETH-USD	LTC-USD	AMZN	TWTR	
0	0.001839	NaN	0.314475	NaN	NaN	NaN	NaN	0.000000	NaN	
1	0.001825	NaN	0.308456	NaN	NaN	NaN	NaN	0.000179	NaN	
2	0.001400	NaN	0.290400	NaN	NaN	NaN	NaN	0.000649	NaN	
3	0.000773	NaN	0.306952	NaN	NaN	NaN	NaN	0.000146	NaN	
4	0.000070	NaN	0.325008	NaN	NaN	NaN	NaN	0.000400	NaN	
...	...	...	...	...	...	...	...	...	...	
2795	0.808553	0.719766	0.626693	0.642578	0.603347	0.678194	0.378168	0.901894	0.868752	0
2796	0.790904	0.691157	0.628047	0.673323	0.637736	0.693455	0.389770	0.893323	0.838730	0
2797	0.807564	0.713427	NaN	0.729365	0.699921	0.759889	0.439515	0.892475	0.833071	0
2798	0.803280	0.697334	NaN	0.717042	0.686409	0.753612	0.436708	0.886626	0.812166	0
2799	0.800814	0.693270	NaN	0.717178	0.683084	0.759023	0.451262	0.878740	0.808708	0

2800 rows × 50 columns

Correlation

There are 4 basic correlation methods

1. CoVariance (for normally distributed data)
2. Pearsons's -> Thats similar for normal distributions as the std correlation coefficient
3. Searman's
4. Kendall

In [66]:
```
stocks_master_df.tail(1000)
```

Out[66]:

	TSLA	NIO	IQQH.F	BTC-USD	BTC-INR	ETH-USD	LTC-USD	AMZN	TWTR	
1800	0.073995	NaN	0.123383	0.061941	NaN	0.075415	0.121122	0.236913	0.041182	0
1801	0.076590	NaN	0.122630	0.062746	NaN	0.076071	0.138455	0.234457	0.046369	0
1802	0.076627	NaN	0.123383	0.065638	NaN	0.078012	0.130315	0.232925	0.045269	0
1803	0.075517	NaN	0.121125	0.066221	NaN	0.079525	0.132949	0.230940	0.041496	0
1804	0.074974	NaN	0.118869	0.066396	NaN	0.083323	0.162662	0.231150	0.043383	0
...	...	...	...	...	...	...	...	...	...	
2795	0.808553	0.719766	0.626693	0.642578	0.603347	0.678194	0.378168	0.901894	0.868752	0
2796	0.790904	0.691157	0.628047	0.673323	0.637736	0.693455	0.389770	0.893323	0.838730	0
2797	0.807564	0.713427	NaN	0.729365	0.699921	0.759889	0.439515	0.892475	0.833071	0
2798	0.803280	0.697334	NaN	0.717042	0.686409	0.753612	0.436708	0.886626	0.812166	0
2799	0.800814	0.693270	NaN	0.717178	0.683084	0.759023	0.451262	0.878740	0.808708	0

1000 rows × 50 columns

In [67]:
```
stocks_master_df.tail(1000).corr(method = 'pearson')
```

Out[67]:

	TSLA	NIO	IQQH.F	BTC-USD	BTC-INR	ETH-USD	LTC-USD	A
TSLA	1.000000	0.960292	0.970995	0.866233	0.843085	0.753243	0.443870	0.88
NIO	0.960292	1.000000	0.950726	0.815069	0.780773	0.756872	0.654887	0.83
IQQH.F	0.970995	0.950726	1.000000	0.820607	0.777331	0.660425	0.381090	0.87
BTC-USD	0.866233	0.815069	0.820607	1.000000	0.999795	0.890069	0.711133	0.69
BTC-INR	0.843085	0.780773	0.777331	0.999795	1.000000	0.900090	0.947173	0.64
ETH-USD	0.753243	0.756872	0.660425	0.890069	0.900090	1.000000	0.776998	0.58
LTC-USD	0.443870	0.654887	0.381090	0.711133	0.947173	0.776998	1.000000	0.23
AMZN	0.884403	0.837973	0.871739	0.696727	0.644339	0.581590	0.230788	1.00
TWTR	0.821358	0.835368	0.812659	0.828291	0.884999	0.718923	0.448674	0.80
FB	0.886076	0.805714	0.857952	0.776130	0.713039	0.752516	0.406330	0.89
SQ	0.966986	0.954258	0.943407	0.853626	0.849859	0.741603	0.403357	0.92
PYPL	0.953743	0.904064	0.935237	0.836344	0.827146	0.734020	0.382639	0.94
BRK-A	0.742256	0.715434	0.712247	0.811265	0.796521	0.829611	0.535120	0.66
CSPX.AS	0.839677	0.782858	0.859275	0.774853	0.814482	0.666674	0.303783	0.89
GC=F	0.793064	0.642233	0.814633	0.571841	0.331839	0.449387	0.142716	0.88
SI=F	0.917486	0.847644	0.898029	0.772794	0.699176	0.699682	0.405119	0.85
CL=F	-0.003035	0.234395	-0.046494	0.224775	0.602948	0.370875	0.421871	-0.13
UA	-0.065860	0.033697	-0.041871	0.151925	0.588508	0.180404	0.201947	-0.15
IT	0.620519	0.598451	0.588760	0.697596	0.709942	0.765616	0.422578	0.56
INR=X	0.516447	0.344975	0.530354	0.332975	-0.009642	0.164785	-0.220845	0.75
EURINR=X	0.851571	0.819089	0.801222	0.657448	0.630670	0.569037	0.215460	0.90
^CNXIT	0.915964	0.922915	0.909821	0.842241	0.855457	0.764927	0.397070	0.87
HAPPSTMNDS.NS	0.194337	0.055464	-0.189313	0.343274	0.353749	0.696746	0.299718	0.68
MPHASIS.NS	0.855207	0.859777	0.813259	0.802642	0.801895	0.801642	0.408965	0.81
WIPRO.NS	0.864932	0.856516	0.837503	0.838399	0.830965	0.839403	0.478942	0.76
MINDTREE.NS	0.878835	0.857996	0.826407	0.832477	0.831843	0.835722	0.443749	0.84
INFY.NS	0.925018	0.918086	0.917882	0.837929	0.848256	0.749405	0.368745	0.90
COFORGE.NS	0.860341	0.821432	0.842394	0.771464	0.735137	0.748683	0.321144	0.86
HCLTECH.NS	0.959041	0.935953	0.953944	0.852907	0.845829	0.742558	0.396617	0.90
TCS.NS	0.878271	0.924185	0.899311	0.786111	0.867922	0.656071	0.285000	0.90
TECHM.NS	0.801405	0.841035	0.818927	0.744721	0.792064	0.670448	0.328032	0.77
ASHOKLEY.NS	0.275695	0.711069	0.186290	0.412764	0.835672	0.556979	0.574301	-0.02
BOSCHLTD.NS	-0.389227	-0.082333	-0.454968	-0.247765	0.448562	-0.070883	0.117974	-0.62
MARUTI.NS	-0.067982	0.473712	-0.122178	-0.011141	0.382121	0.141103	0.352135	-0.29

	TSLA	NIO	IQQH.F	BTC-USD	BTC-INR	ETH-USD	LTC-USD	A
TATAMOTORS.NS	0.092031	0.697420	-0.010804	0.281200	0.876129	0.455661	0.584371	-0.26
ESCORTS.NS	0.872791	0.871841	0.825773	0.682095	0.641266	0.611219	0.371200	0.82
BAJAJ-AUTO.NS	0.847726	0.836243	0.823947	0.838619	0.812959	0.827947	0.544007	0.67
EXIDEIND.NS	-0.309443	-0.038269	-0.358542	-0.187203	0.569907	-0.075353	0.059326	-0.42
AMARAJABAT.NS	0.550608	0.770453	0.536011	0.490708	0.585582	0.427731	0.445479	0.30
BALKRISIND.NS	0.866417	0.872368	0.789712	0.774939	0.742017	0.831217	0.488382	0.79
MRF.NS	0.655975	0.859689	0.594643	0.701292	0.847128	0.704135	0.572766	0.40
NATIONALUM.NS	0.173271	0.466016	0.038941	0.371997	0.735709	0.611478	0.516312	-0.08
NMDC.NS	0.584730	0.624614	0.502925	0.726652	0.735821	0.882740	0.648746	0.40
COALINDIA.NS	-0.653951	-0.462892	-0.687399	-0.431558	-0.000226	-0.239929	0.075555	-0.77
VEDL.NS	0.206779	0.572195	0.073903	0.424306	0.836592	0.646475	0.640952	-0.10
TATASTEEL.NS	0.593598	0.703338	0.479346	0.701631	0.787103	0.877882	0.624802	0.37
JINDALSTEL.NS	0.751237	0.831851	0.651409	0.817481	0.888002	0.915848	0.709131	0.58
TATASTEEL.NS	0.593598	0.703338	0.479346	0.701631	0.787103	0.877882	0.624802	0.37
JSWSTEEL.NS	0.709600	0.727826	0.616578	0.773142	0.806816	0.893917	0.551337	0.59
SAIL.NS	0.405932	0.636537	0.267678	0.574485	0.787788	0.809028	0.678931	0.18

50 rows × 50 columns

Mathematicians recomends spearmans for stock analysis

In [68]:
```
stocks_master_df.tail(1000).corr(method = 'spearman')
```

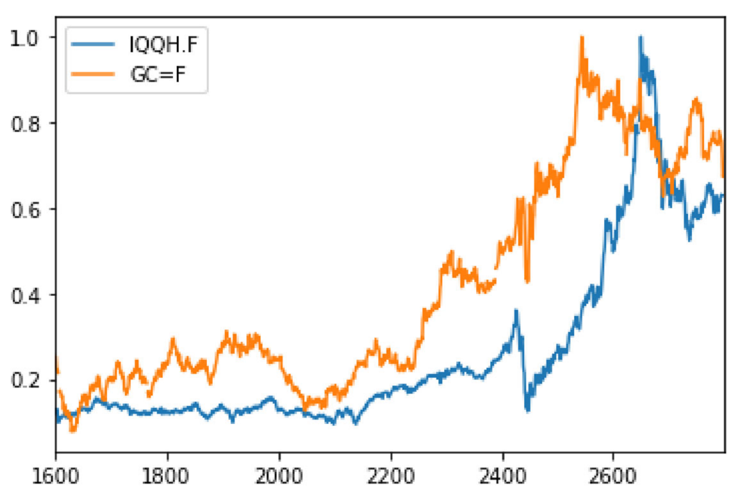
Out[68]:

TWTR	FB	...	MRF.NS	NATIONALUM.NS	NMDC.NS	COALINDIA.NS	VEDL.NS	TATASTEEL.N
28003	0.690620	...	0.521364	-0.053200	0.268836	-0.691254	0.019674	0.27303
40500	0.631973	...	0.655231	0.381977	0.410093	-0.401693	0.456400	0.66279
12451	0.849018	...	0.216246	-0.320727	0.196192	-0.790541	-0.256240	-0.00415
36606	0.810589	...	0.448274	0.015182	0.465022	-0.495672	0.103093	0.31615
95263	0.874221	...	0.723526	0.620849	0.633212	-0.161832	0.660090	0.77642
54350	0.580280	...	0.713127	0.494937	0.636185	-0.052979	0.591554	0.71508
11946	0.376512	...	0.554113	0.537936	0.720114	0.220668	0.624447	0.69553
41831	0.796328	...	0.200038	-0.297325	0.094467	-0.720070	-0.274755	-0.00205
00000	0.732767	...	0.361282	-0.050222	0.208877	-0.435339	-0.000586	0.19599
32767	1.000000	...	0.368230	-0.143640	0.333020	-0.643134	-0.066519	0.16479
57666	0.704427	...	0.280424	-0.152856	0.175011	-0.600882	-0.135055	0.14854
33580	0.834096	...	0.171040	-0.312970	0.139349	-0.757480	-0.271043	-0.00620
35548	0.567462	...	0.416226	0.155205	0.541232	-0.246729	0.188059	0.41146
58039	0.844301	...	0.237934	-0.276916	0.220472	-0.722500	-0.248813	0.01819
57884	0.791646	...	0.136190	-0.410995	0.093481	-0.846944	-0.321222	-0.11480
69300	0.815511	...	0.430010	-0.033908	0.377217	-0.635553	0.059779	0.22839
63554	-0.023747	...	0.458451	0.723539	0.576745	0.613667	0.698948	0.61737
24287	-0.087350	...	0.066592	0.354846	0.269979	0.417301	0.312972	0.27081
10405	0.468465	...	0.306539	0.231370	0.522561	-0.093405	0.226890	0.40304
29772	0.497553	...	-0.013603	-0.485651	-0.255699	-0.795699	-0.509842	-0.29965
02220	0.549293	...	0.413234	-0.116900	0.057767	-0.586756	-0.084793	0.14746
21387	0.710821	...	0.244240	-0.118371	0.258429	-0.537069	-0.097637	0.16548
34901	0.726972	...	0.388310	0.924357	0.870938	0.716891	0.918871	0.91014
27226	0.605357	...	0.421293	0.095090	0.233894	-0.343184	0.097124	0.31197
98580	0.579885	...	0.161487	0.036679	0.351044	-0.431462	0.077319	0.30488
55861	0.679515	...	0.449659	0.001543	0.222561	-0.456158	0.009248	0.26232
26740	0.773234	...	0.204317	-0.241406	0.174721	-0.691068	-0.208765	0.03817
66927	0.798899	...	0.255609	-0.255543	0.192200	-0.714176	-0.229798	0.02404
56301	0.784576	...	0.264300	-0.209130	0.247256	-0.645303	-0.168671	0.09504
06668	0.795817	...	0.215670	-0.234600	0.211941	-0.656362	-0.202294	0.05844
48606	0.583497	...	0.312167	-0.013389	0.326077	-0.410046	0.001033	0.24620
63415	0.039687	...	0.784085	0.845264	0.706608	0.488465	0.867659	0.87708
81121	-0.619800	...	0.146248	0.729186	0.302227	0.833080	0.690731	0.51611
57678	-0.153858	...	0.601675	0.630361	0.465838	0.573919	0.656536	0.62464

TWTR	FB	...	MRF.NS	NATIONALUM.NS	NMDC.NS	COALINDIA.NS	VEDL.NS	TATASTEEL.N
18118	-0.079453	...	0.598238	0.882376	0.726696	0.555908	0.921162	0.87870
48024	0.658321	...	0.668521	0.104151	0.297309	-0.424545	0.167485	0.39886
33591	0.744125	...	0.470054	0.163428	0.570911	-0.414862	0.237424	0.44835
90650	-0.536666	...	0.271739	0.615584	0.227565	0.807158	0.564569	0.46278
63950	0.276393	...	0.789471	0.441324	0.573650	0.068531	0.486991	0.66243
37684	0.669681	...	0.738745	0.225604	0.438834	-0.367349	0.258523	0.49530
61282	0.368230	...	1.000000	0.565253	0.635405	0.043310	0.600490	0.71962
50222	-0.143640	...	0.565253	1.000000	0.682407	0.587727	0.961360	0.85906
08877	0.333020	...	0.635405	0.682407	1.000000	0.215354	0.737200	0.82193
35339	-0.643134	...	0.043310	0.587727	0.215354	1.000000	0.544641	0.32791
00586	-0.066519	...	0.600490	0.961360	0.737200	0.544641	1.000000	0.90362
95996	0.164792	...	0.719624	0.859068	0.821937	0.327912	0.903620	1.00000
08209	0.487527	...	0.792460	0.539543	0.677973	0.039895	0.598806	0.77544
95996	0.164792	...	0.719624	0.859068	0.821937	0.327912	0.903620	1.00000
73793	0.349597	...	0.685997	0.519491	0.552986	0.075050	0.532234	0.71259
40106	0.071827	...	0.704106	0.907944	0.766153	0.492817	0.928555	0.93242

In [69]:
```
stocks_master_df[['IQQH.F','GC=F']].tail(1200).plot()
```

Out[69]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff2a6e8b040>



In [ ]:

In [ ]:

In [ ]:

```

```

In [ ]:

```

```

correlation plotting on seaborn

In [70]:

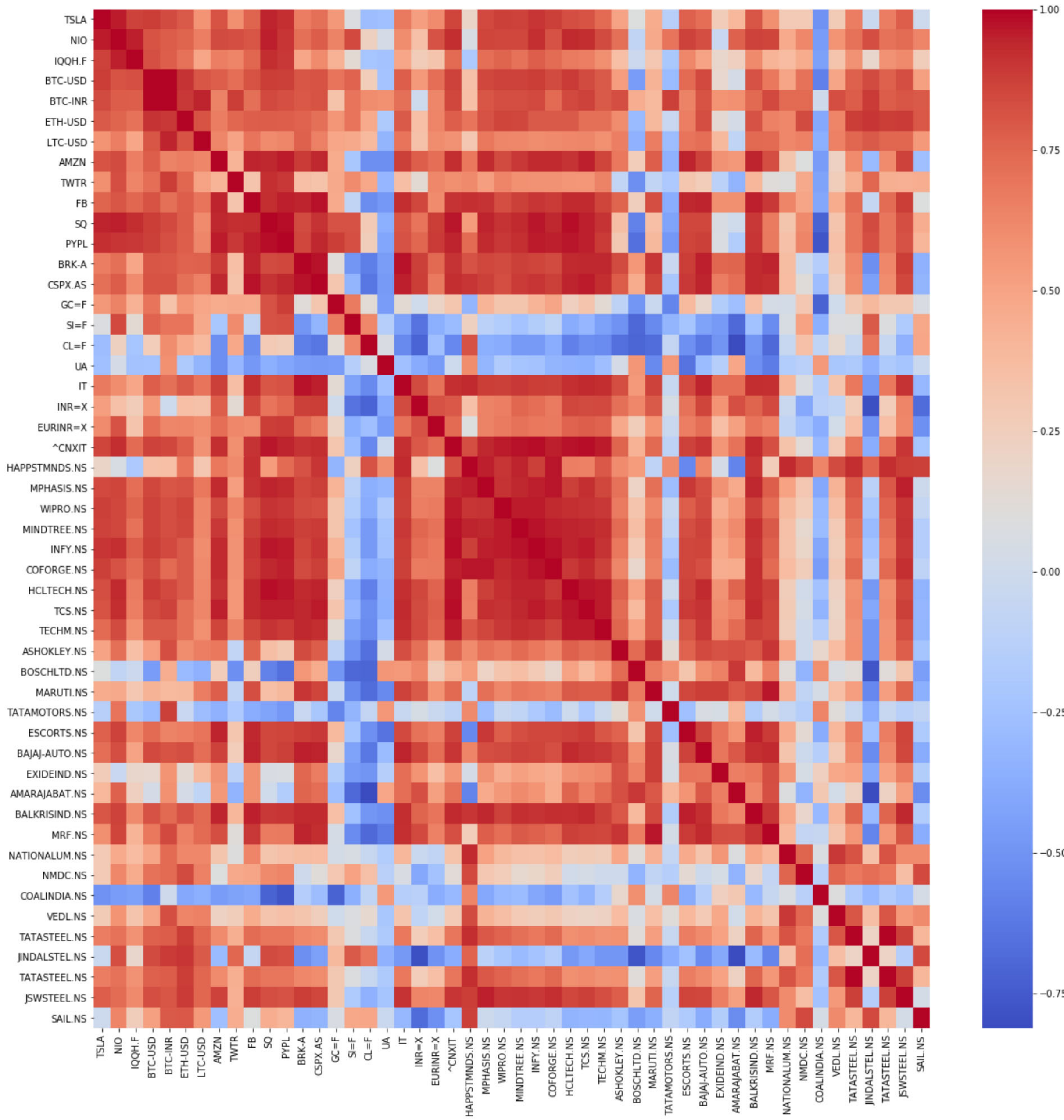
```
import seaborn as sns
```

In [81]:

```
plt.figure(figsize=(20,20))
sns.heatmap(stocks_master_df.corr(),cmap='coolwarm')
```

Out[81]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff2a60a2100>
```

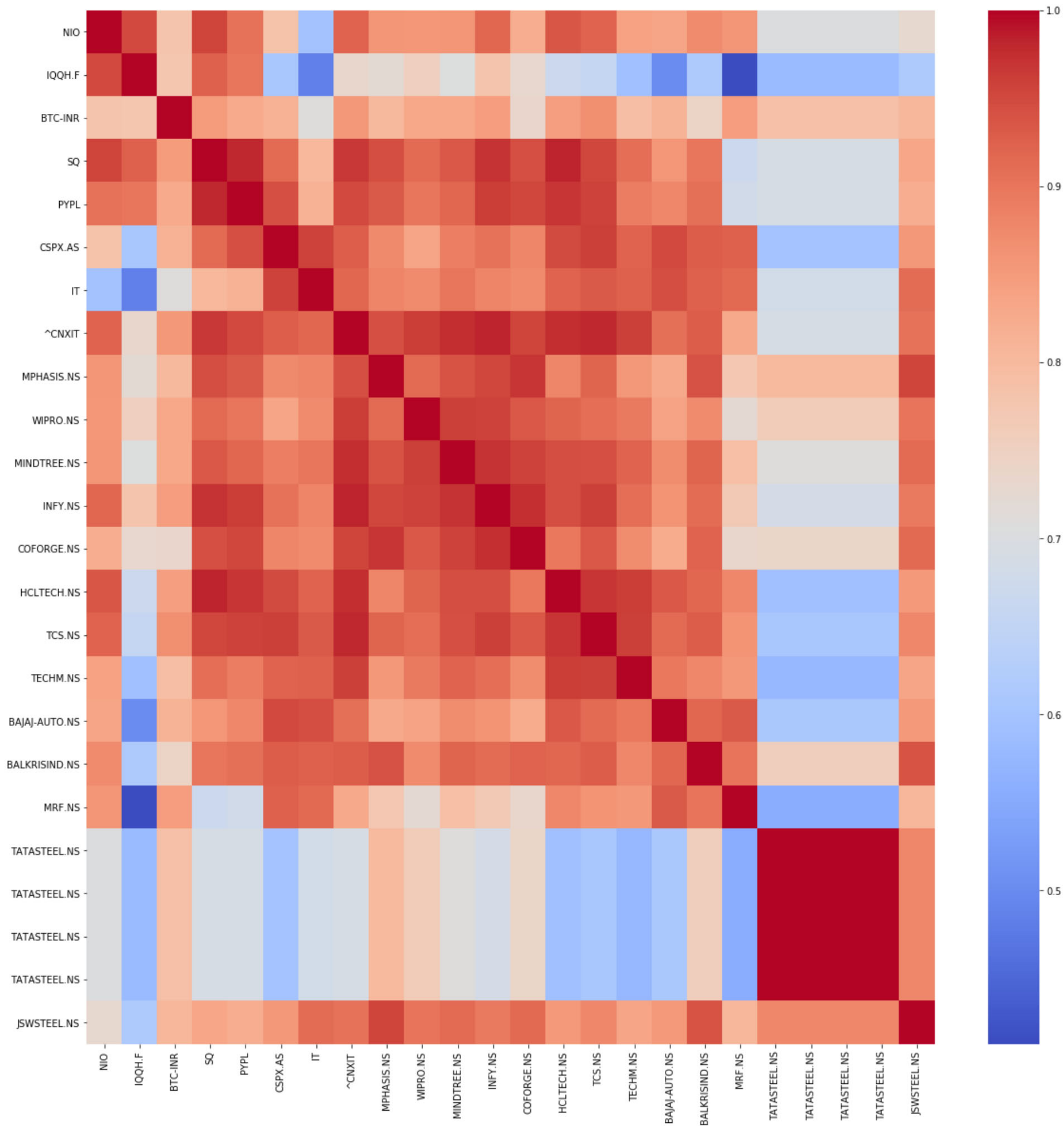


Plotting higger correlations

In [82]:
```
corr_matrix = stocks_master_df.corr(method='pearson').abs()
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool
corr_cols = [column for column in upper.columns if any(upper[column] > 0.95)]
```

In [83]:
```
plt.figure(figsize=(20,20))
sns.heatmap(stocks_master_df[corr_cols].corr(),cmap='coolwarm')
```

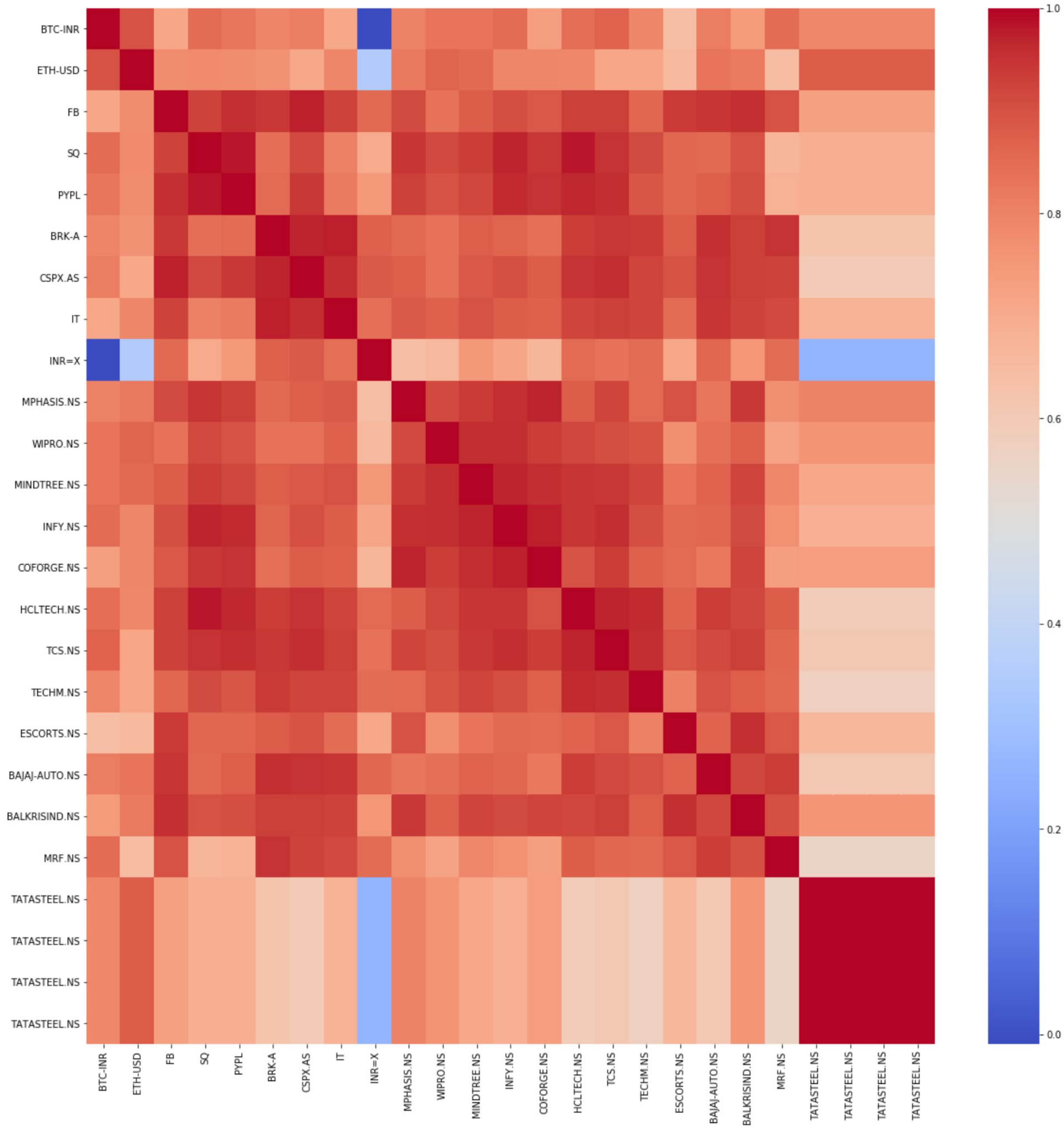
Out[83]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff2a70afa00>



In [77]:
```
corr_matrix = stocks_master_df.corr(method='spearman').abs()
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool
corr_cols = [column for column in upper.columns if any(upper[column] > 0.95)]
```

In [78]:
```
plt.figure(figsize=(20,20))
sns.heatmap(stocks_master_df[corr_cols].corr(),cmap='coolwarm')
```

Out[78]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff2a5cf6820>



In [84]:
```
stocks_master_df.tail()
```

Out[84]:

TWTR	FB	...	MRF.NS	NATIONALUM.NS	NMDC.NS	COALINDIA.NS	VEDL.NS	TATASTEEL.N
).868752	0.971003	...	0.811423	1.000000	0.848574	0.243568	0.955280	0.9907
).838730	0.972521	...	0.805916	0.970371	0.837333	0.247101	0.963443	0.9771
).833071	0.967178	...	0.815658	0.829015	0.811544	0.231614	0.947117	0.9643
).812166	0.965828	...	0.802291	0.774077	0.771538	0.225908	0.930081	0.9346
).808708	0.962537	...	0.788898	0.824077	0.794021	0.235417	1.000000	0.9750

In [ ]:

In [ ]:

In [ ]: