In [1]:

```
# TODO: Read data
import pandas as pd
TWITTER = pd.read_csv('/Users/khaladdin/Desktop/Twitter Project/TWITTERTASK4.csv')
```

In [2]:

```
# TODO: Change the format of datetime.
TWITTER['Date'] = pd.to_datetime(TWITTER['Date-Time'])
```

In [4]:

```
# TODO: Change format.
TWITTER['Date1'] = TWITTER['Date'].dt.date
```

In [6]:

```
# TODO: Create group columns.
TWITTER["maingroup"] = TWITTER["#RIC"] + TWITTER["Date1"].map(str)
TWITTER["maingroup1"] = TWITTER["#RIC"] + TWITTER["Date"].map(str)
```

In [8]:

```
# TODO: Compute return.
import numpy as np
TWITTER["logret"] = TWITTER.groupby("#RIC")['Last'].apply(lambda x: np.log(x) - np.log(x.shift()))
```

In [10]:

```
# TODO: Find absolute value.
TWITTER['abslogret'] = TWITTER['logret'].abs()
```

In [11]:

```
# TODO: Delete missing.
TWITTER = TWITTER[np.isfinite(TWITTER['abslogret'])]
```

In [13]:

```
# TODO: Ascengind order. For descengind write False
TWITTER = TWITTER.sort_values('Date', ascending=True)
```

In [14]:

```
# TODO: Compute cumulative sum for date.
TWITTER['cumsum'] = TWITTER.groupby(['Date1'])['abslogret'].apply(lambda x: x.cumsum())
```

In [16]:

```
# TODO: Keep only last. Only daily
cumsum = TWITTER.groupby('Date1', as_index=False).last()
```

In [18]:

```
# TODO: Keep some columns.
cumsum = cumsum[['Date1','cumsum']]
```

In [19]:

```
# TODO: Change the name of columns.
cumsum = cumsum.rename (columns ={'cumsum':'total'})
```

In [21]:

```
# TODO: Merge datasets.
TWITTER4 = pd.merge(cumsum, TWITTER, on='Date1', how='outer')
```

In [23]:

```
# TODO: Keep only last. Only daily
dailyreturn = TWITTER.groupby('maingroup', as_index=False).last()
```

In [25]:

```
# TODO: Ascengind order. For descengind write False
dailyreturn = dailyreturn.sort_values('maingroup', ascending=True)
```

In [26]:

```
# TODO: Look data
dailyreturn.head(4)
```

Out[26]:

	maingroup	#RIC	Alias Underlying RIC	Domain	Date-Time	GMT Offset	Ty
0	ADI.OQ2018-07-02	ADI.OQ	NaN	Market Price	2018-07-02T20:00:00.000000000Z	-4	Intrac 10Mi
1	ADI.OQ2018-07-03	ADI.OQ	NaN	Market Price	2018-07-03T17:00:00.000000000Z	-4	Intrac 10Mi
2	ADI.OQ2018-07-05	ADI.OQ	NaN	Market Price	2018-07-05T20:00:00.000000000Z	-4	Intrac 10Mi
3	ADI.OQ2018-07-06	ADI.OQ	NaN	Market Price	2018-07-06T20:00:00.000000000Z	-4	Intrac 10Mi

In [27]:

```
# TODO: Compute return.
import numpy as np
dailyreturn["dayret"] = dailyreturn.groupby("#RIC")['Last'].apply(lambda x: np
.log(x) - np.log(x.shift()))
```

In [29]:

```
# TODO: Delete missing.
dailyreturn = dailyreturn[np.isfinite(dailyreturn['dayret'])]
```

In [30]:

```
# TODO: Find absolute value.
dailyreturn['absdayret'] = dailyreturn['dayret'].abs()
```

In [32]:

```
# TODO: Keep some columns.
dailyreturn = dailyreturn[['maingroup','dayret','absdayret']]
```

In [34]:

```
# TODO: Merge datasets.
TWITTER5 = pd.merge(dailyreturn, TWITTER4, on='maingroup', how='outer')
```

In [35]:

```
# TODO: Ascendind order. For descendind write False
TWITTER5 = TWITTER5.sort_values('maingroup1', ascending=True)
```

In [37]:

```
# TODO: Delete missing.
TWITTER5 = TWITTER5[np.isfinite(TWITTER5['dayret'])]
```

In [39]:

```
# TODO: COmpute WPC.
TWITTER5['weight'] = TWITTER5['absdayret']/TWITTER5['total']
```

In [40]:

```
# TODO: COmpute WPC.
TWITTER5['WPC'] = (TWITTER5['weight']*TWITTER5['logret'])/TWITTER5['dayret']
```

In [42]:

```
# TODO: COmpute midquote and effective spread
TWITTER5['midprice'] = (TWITTER5['Close Bid']+TWITTER5['Close Ask'])/2
TWITTER5['diff'] = (TWITTER5['Last']-TWITTER5['midprice'])
TWITTER5['absdiff'] = TWITTER5['diff'].abs()
TWITTER5['effective'] = 2*TWITTER5['absdiff']
```

In [47]:

```
# TODO: COmpute logvolume
import numpy as np
TWITTER5['LOGVOLUME'] = np.log(TWITTER5['Volume'])
```

In [48]:

```
# TODO: COmpute averagesize
TWITTER5['averagesize'] = TWITTER5['Volume']/TWITTER5['No. Trades']
```

In [49]:

```
# TODO: COmpute logaveragesize
TWITTER5['LOGsize'] = np.log(TWITTER5['averagesize'])
```

In [50]:

```
# TODO: COmpute price changes. I test it with st. dev daily return too. Result
s are completely same. However, this method is simpler by computatinally
TWITTER5['prichan'] = TWITTER5['Last'] - TWITTER5['Last'].shift(1)
```

In [51]:

```
# TODO:Absolute price changes
TWITTER5['vol'] = TWITTER5['prichan'].abs()
```

In [52]:

```
# TODO:read twitter data
TWITTERADI = pd.read_csv('/Users/khaladdin/Desktop/Twitter Project/ADI.csv', p
arse_dates = [["Date", "Time"]], index_col=0)
TWITTERFID = pd.read_csv('/Users/khaladdin/Desktop/Twitter Project/Fidelity.cs
v', parse_dates = [["Date", "Time"]], index_col=0)
TWITTERFIS = pd.read_csv('/Users/khaladdin/Desktop/Twitter Project/Fiserv.csv'
, parse_dates = [["Date", "Time"]], index_col=0)
TWITTERGPN = pd.read_csv('/Users/khaladdin/Desktop/Twitter Project/Global.csv'
, parse_dates = [["Date", "Time"]], index_col=0)
TWITTERJUN = pd.read_csv('/Users/khaladdin/Desktop/Twitter Project/Juniper.csv
', parse_dates = [["Date", "Time"]], index_col=0)
```

In [53]:

```
# TODO:Create 10 minutes interval
TWITTERADIa = TWITTERADI.resample("10T").mean()
TWITTERFIDa = TWITTERFID.resample("10T").mean()
TWITTERFISa = TWITTERFIS.resample("10T").mean()
TWITTERGPNa = TWITTERGPN.resample("10T").mean()
TWITTERJUNa = TWITTERJUN.resample("10T").mean()
```

In [54]:

```
# TODO:convert index into colum
TWITTERADIa['Date'] = TWITTERADIa.index
TWITTERFIDa['Date'] = TWITTERFIDa.index
TWITTERFISa['Date'] = TWITTERFISa.index
TWITTERGPNa['Date'] = TWITTERGPNa.index
TWITTERJUNa['Date'] = TWITTERJUNa.index
```

In [55]:

```
# TODO: Delete missing.
TWITTERADIa = TWITTERADIa[np.isfinite(TWITTERADIa['Number of Tweets'])]
TWITTERFIDa = TWITTERFIDa[np.isfinite(TWITTERFIDa['Number of Tweets'])]
TWITTERFISa = TWITTERFISa[np.isfinite(TWITTERFISa['Number of Tweets'])]
TWITTERGPNa = TWITTERGPNa[np.isfinite(TWITTERGPNa['Number of Tweets'])]
TWITTERJUNa = TWITTERJUNa[np.isfinite(TWITTERJUNa['Number of Tweets'])]
```

In [56]:

```
# TODO: Change the format of datetime.
TWITTERADIa['Date'] = pd.to_datetime(TWITTERADIa['Date'])
TWITTERFIDa['Date'] = pd.to_datetime(TWITTERFIDa['Date'])
TWITTERFISa['Date'] = pd.to_datetime(TWITTERFISa['Date'])
TWITTERGPNa['Date'] = pd.to_datetime(TWITTERGPNa['Date'])
TWITTERJUNa['Date'] = pd.to_datetime(TWITTERJUNa['Date'])
```

In [57]:

```
# TODO: Create twit dummy
TWITTERADIa['Dummy'] = 1
TWITTERFIDa['Dummy'] = 1
TWITTERFISa['Dummy'] = 1
TWITTERGPNa['Dummy'] = 1
TWITTERJUNa['Dummy'] = 1
```

In [58]:

```
# TODO:We select a stock, since we will do further analysis stock by stock sep
arately.
ADI = TWITTER5.loc[TWITTER5['#RIC'] == 'ADI.OQ']
FID = TWITTER5.loc[TWITTER5['#RIC'] == 'FIS.N']
FIS = TWITTER5.loc[TWITTER5['#RIC'] == 'FISV.OQ']
GPN = TWITTER5.loc[TWITTER5['#RIC'] == 'GPN.N']
JUN = TWITTER5.loc[TWITTER5['#RIC'] == 'JNPR.N']
```

In [114]:

```
# TODO: Merge datasets.We merge two datasets by using daily group (interval).
SO, total sum of returns of all stocks is
# TODO: going to the front of each stock's daily return (this is the first par
t of WPC equation)
ADIMAIN = pd.merge(TWITTERADIa, ADI, on='Date', how='outer')
FIDMAIN = pd.merge(TWITTERFIDa, FID, on='Date', how='outer')
FISMAIN = pd.merge(TWITTERFISa, FIS, on='Date', how='outer')
GPNMAIN = pd.merge(TWITTERGPNa, GPN, on='Date', how='outer')
JUNMAIN = pd.merge(TWITTERJUNa, JUN, on='Date', how='outer')
```

In [115]:

```
# TODO: Delete missing.
ADIMAIN = ADIMAIN[np.isfinite(ADIMAIN['WPC'])]
FIDMAIN = FIDMAIN[np.isfinite(FIDMAIN['WPC'])]
FISMAIN = FISMAIN[np.isfinite(FISMAIN['WPC'])]
GPNMAIN = GPNMAIN[np.isfinite(GPNMAIN['WPC'])]
JUNMAIN = JUNMAIN[np.isfinite(JUNMAIN['WPC'])]
```

In [116]:

```
# TODO: 0s and 1s for dummy
ADIMAIN['Dummy'] = ADIMAIN['Dummy'].replace(np.nan, 0)
FIDMAIN['Dummy'] = FIDMAIN['Dummy'].replace(np.nan, 0)
FISMAIN['Dummy'] = FISMAIN['Dummy'].replace(np.nan, 0)
GPNMAIN['Dummy'] = GPNMAIN['Dummy'].replace(np.nan, 0)
JUNMAIN['Dummy'] = JUNMAIN['Dummy'].replace(np.nan, 0)
```

In [117]:

```
# TODO: Ascengind order. For descengind write False
ADIMAIN = ADIMAIN.sort_values('Date', ascending=True)
FIDMAIN = FIDMAIN.sort_values('Date', ascending=True)
FISMAIN = FISMAIN.sort_values('Date', ascending=True)
GPNMAIN = GPNMAIN.sort_values('Date', ascending=True)
JUNMAIN = JUNMAIN.sort_values('Date', ascending=True)
```

In [121]:

```
# TODO: I have already computed HFT proxy for task5. I just input and use it
HFTADI = pd.read_csv('/Users/khaladdin/Desktop/Twitter Project/HFTADI.csv')
HFTFID = pd.read_csv('/Users/khaladdin/Desktop/Twitter Project/HFTFID.csv')
HFTFIS = pd.read_csv('/Users/khaladdin/Desktop/Twitter Project/HFTFIS.csv')
HFTGPN = pd.read_csv('/Users/khaladdin/Desktop/Twitter Project/HFTGPN.csv')
HFTJUN = pd.read_csv('/Users/khaladdin/Desktop/Twitter Project/HFTJUN.csv')
```

In [122]:

```
# TODO: Change the format of datetime.
HFTADI['Date'] = pd.to_datetime(HFTADI['Datetime'])
HFTFID['Date'] = pd.to_datetime(HFTFID['Datetime'])
HFTFIS['Date'] = pd.to_datetime(HFTFIS['Datetime'])
HFTGPN['Date'] = pd.to_datetime(HFTGPN['Datetime'])
HFTJUN['Date'] = pd.to_datetime(HFTJUN['Datetime'])
```

In [123]:

```
# TODO: Keep some columns.
HFTADI = HFTADI[['Date','HFT']]
HFTFID = HFTFID[['Date','HFT']]
HFTFIS = HFTFIS[['Date','HFT']]
HFTGPN = HFTGPN[['Date','HFT']]
HFTJUN = HFTJUN[['Date','HFT']]
```

In [125]:

```
# TODO: Create index for 10 minute interval.
HFTADI = HFTADI.set_index(pd.DatetimeIndex(HFTADI['Date']))
HFTFID = HFTFID.set_index(pd.DatetimeIndex(HFTFID['Date']))
HFTFIS = HFTFIS.set_index(pd.DatetimeIndex(HFTFIS['Date']))
HFTGPN = HFTGPN.set_index(pd.DatetimeIndex(HFTGPN['Date']))
HFTJUN = HFTJUN.set_index(pd.DatetimeIndex(HFTJUN['Date']))
```

In [126]:

```
# TODO: Create 10 minute interval.
HFTADIa = HFTADI.resample("10T").mean()
HFTFIDa = HFTFID.resample("10T").mean()
HFTFISa = HFTFIS.resample("10T").mean()
HFTGPNa = HFTGPN.resample("10T").mean()
HFTJUNa = HFTJUN.resample("10T").mean()
```

In [128]:

```
# TODO: Create index into column.
HFTADIa['Date'] = HFTADIa.index
HFTFIDa['Date'] = HFTFIDa.index
HFTFISa['Date'] = HFTFISa.index
HFTGPNa['Date'] = HFTGPNa.index
HFTJUNa['Date'] = HFTJUNa.index
```

In [129]:

```
# TODO: Change the format of datetime.
HFTADIa['Date'] = pd.to_datetime(HFTADIa['Date'])
HFTFIDa['Date'] = pd.to_datetime(HFTFIDa['Date'])
HFTFISa['Date'] = pd.to_datetime(HFTFISa['Date'])
HFTGPNa['Date'] = pd.to_datetime(HFTGPNa['Date'])
HFTJUNa['Date'] = pd.to_datetime(HFTJUNa['Date'])
```

In [131]:

```
# TODO: Merge datasets.
ADIMAIN1 = pd.merge(ADIMAIN, HFTADIa, on='Date', how='outer')
FIDMAIN1 = pd.merge(FIDMAIN, HFTFIDa, on='Date', how='outer')
FISMAIN1 = pd.merge(FISMAIN, HFTFISa, on='Date', how='outer')
GPNMAIN1 = pd.merge(GPNMAIN, HFTGPNa, on='Date', how='outer')
JUNMAIN1 = pd.merge(JUNMAIN, HFTJUNa, on='Date', how='outer')
```

In [133]:

```
# TODO: Cleaning.
ADIMAIN1 = ADIMAIN1.replace([np.inf, -np.inf], np.nan)
FIDMAIN1 = FIDMAIN1.replace([np.inf, -np.inf], np.nan)
FISMAIN1 = FISMAIN1.replace([np.inf, -np.inf], np.nan)
GPNMAIN1 = GPNMAIN1.replace([np.inf, -np.inf], np.nan)
JUNMAIN1 = JUNMAIN1.replace([np.inf, -np.inf], np.nan)
```

In [134]:

```
# TODO: Cleaning.
ADIMAIN1 = ADIMAIN1[np.isfinite(ADIMAIN1['HFT'])]
FIDMAIN1 = FIDMAIN1[np.isfinite(FIDMAIN1['HFT'])]
FISMAIN1 = FISMAIN1[np.isfinite(FISMAIN1['HFT'])]
GPNMAIN1 = GPNMAIN1[np.isfinite(GPNMAIN1['HFT'])]
JUNMAIN1 = JUNMAIN1[np.isfinite(JUNMAIN1['HFT'])]
```

In [136]:

```
# TODO: Combine.
Totalmain = pd.concat([ADIMAIN1,FIDMAIN1,FISMAIN1,GPNMAIN1,JUNMAIN1])
```

In [137]:

```
# TODO: Create RIC and DATE idnex for FIXED Regression
Totalmain = Totalmain.set_index(['#RIC','Date1'])
```

In [138]:

```
# TODO: Cleaning.
TotalTask4 = Totalmain.replace([np.inf, -np.inf], np.nan)
```

In [139]:

```
# TODO: LOGHFT.
TotalTask4['logHFT'] = np.log(TotalTask4['HFT'])
```

In [140]:

```
# TODO: Cleaning.
TotalTask4['logHFT'] = TotalTask4['logHFT'].replace(np.nan, 0)
```

In [142]:

```
# TODO: ADD intercept.
TotalTask4['intercept'] = 1
```

In [ ]:

```
# TODO: FIXED EFFECT.
from linearmodels import PanelOLS
mod = PanelOLS(TotalTask4.WPC, TotalTask4[['intercept','Dummy','effective','LO
GVOLUME','LOGsize','vol','logHFT']], entity_effects=True)
res = mod.fit(cov_type='clustered', cluster_entity=True)
res
```

In [144]:

```
# TODO: Winsorised
from scipy.stats import mstats
def WinsorizeStats(TotalTask4):
    out = mstats.winsorize(TotalTask4, limits=[0.05, 0.05])
    return out
```

In [145]:

```
# TODO: Winsorised
TotalTask5 = TotalTask4[['WPC','effective','LOGVOLUME','LOGsize','vol','logHFT
']].apply(WinsorizeStats, axis=0)
```

In [146]:

```
# TODO: CLeaning
TotalTask5['vol'].fillna(0, inplace=True)
```

In [147]:

```
# TODO: Keep some columns.
dummy = TotalTask4[['intercept','Dummy']]
```

In [148]:

```
# TODO: Create new column. I need it since, I should merge dummy and TotalTask
5 datasets based on any column.
dummy['C']=dummy.reset_index().index
TotalTask5['C']=TotalTask5.reset_index().index
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:1: Se
ttingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
```

In [149]:

```
# TODO: Merge two datasets and set RIC (stock) and DATE as index for FIXED eff
ect regression
TotalTask6 = TotalTask5.reset_index().merge(dummy, on='C', how='outer').set_in
dex(['#RIC','Date1'])
```

In [153]:

```
# TODO: Delete missing.
TotalTask6 = TotalTask6[np.isfinite(TotalTask6['WPC'])]
```

In [ ]:

```
# TODO: FIXED effect regression
from linearmodels import PanelOLS
mod = PanelOLS(TotalTask6.WPC, TotalTask6[['intercept','Dummy','effective','LO
GVOLUME','LOGsize','vol','logHFT']], entity_effects=True)
res = mod.fit(cov_type='clustered', cluster_entity=True)
res
```