FPI Report and Nifty Indices,Exploratory Data Analysis

- The goal of this notebook is to do data analysis on NSDL Sector-wise FPI Investment report, as well as the sectoral Indices.

Dataset

- Dataset consists of FPIs report and Nifty indices data from kaggle.
- Lets quicky understand what those are:

What is FPI?

- Foreign Investment inflow is an important reason for India's economic growth. So to simplify compliance requirements and have uniform guidelines for various categories of foreign investors like Foreign Institutional Investors (FIIs), Sub Accounts and Qualified Foreign Investors (QFIs) merged into a new investor class termed as Foreign Portfolio Investors (FPIs).
- SEBI has authorized NSDL to monitor of these Group investment and various data related to FPI activities to be displayed on NSDL web portal.
- NSDL provides this data at an interval of 15 days.

Source: *FPI NSDL*

Nifty sectorial Indices

- This includes NIFTY 50, NIFTY AUTO, NIFTY BANK, NIFTY FMCG, NIFTY IT, NIFTY METAL, NIFTY OILGAS, NIFTY PHARMA, NIFTY PRIVATE BANK.
- These indices are designed to reflect the behavior and performance of their respective sectors.

Tools used

- Pandas
- Datetime
- glob
- plotly

Downloading the Dataset

- Get nifty indices data from kaggle. *Source:* *INDICES DATA*

- Installing necessary packages

```
!pip install jovian openpyxl cufflinks plotly opendatasets --upgrade --quiet
```

Let's begin by downloading the data, and listing the files within the dataset.

```
# storing kaggle dataset url in a varaible
dataset_url = 'https://www.kaggle.com/atrisaxena/nifty-indices-data'
```

```
# download data from url
import opendatasets as od
od.download(dataset_url)
```

```
Skipping, found downloaded files in "./nifty-indices-data" (use force=True to force download)
```

```
# get FPI data from github
from urllib.request import urlopen
from io import BytesIO
from zipfile import ZipFile


# function to download and unzip files
def download_and_unzip(url, extract_to='./FPI_Data'):
    http_response = urlopen(url)
    zipfile = ZipFile(BytesIO(http_response.read()))
    zipfile.extractall(path=extract_to)
```

```
download_and_unzip('https://github.com/doke93/FPI_EDA/files/8159061/FPI_Data.zip')
```

- Store data path in variable

```
index_data_dir = './nifty-indices-data'
fpi_data_dir = './FPI_Data'
```

```
# check whether data is loaded in the notebook
import os
os.listdir(fpi_data_dir)[-1]
```

```
'FPI_28-Feb-2021.xlsx'
```

```
project_name = "fpi-indices-data-analysis"
```

```
jovian.commit(project=project_name)
```

```
[jovian] Updating notebook "dokeabhishek3/fpi-indices-data-analysis" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/dokeabhishek3/fpi-indices-data-analysis
```

```
'https://jovian.ai/dokeabhishek3/fpi-indices-data-analysis'
```

Importing libraries

```
import pandas as pd
import glob
import datetime
import warnings
from pandas.core.common import SettingWithCopyWarning

warnings.simplefilter(action="ignore", category=SettingWithCopyWarning)
```

Data Preparation and Cleaning

- Creating pandas dataframe by merging all the FPIs reports.

```
# function to read xlsx file, remove columns that are not relevant for our analysis.
def get_data(xlsx_file):
    temp = pd.read_excel(xlsx_file, sheet_name='Sheet1')
    df = temp[2:-1].copy()

    # Select subset of columns with the relevant data for our analysis
    df = df.drop(df.columns[createList(0,(len(df.columns)-1))], axis = 1)
    df.rename(columns={'Unnamed: 1': 'Sector'},inplace=True)
    return df

# function to create a list of numbers
def createList(r1, r2):
    return [item for item in range(r1, r2+1) if (item != 1)&(item != 2)& (item != 32)]

# fetch all the .xlsx files from the diretory
path = pd.DataFrame(glob.glob(fpi_data_dir + "/*.xlsx"),columns=['location'])

# Parse dates from the column name
path['data_date'] = path['location'].apply(lambda x: x.split('/')[2].split('_')[1].spli
path['data_date'] = path['data_date'].apply(lambda x: datetime.datetime.strptime(x,'%d-

# sort the data as per data_date in ascending order
path.sort_values(['data_date'], inplace=True)
path.reset_index(drop=True, inplace=True)

# for loop to store fpi data in a key and value pair and latter merging the data
my_dict = {}
i = 1
for index, row in path.iterrows():
    my_dict[f"df_{i}"] = get_data(row['location'])

    for c in my_dict[f"df_{i}"].columns[1:]:
        col_name = c.split(' ')[3:]
        col = ''.join(col_name)
        my_dict[f"df_{i}"].rename(columns={c:col},inplace=True)

    if len(my_dict)==1:
```

```
        merged_df = my_dict[f"df_{i}"]
    else:
        merged_df = pd.merge(merged_df, my_dict[f"df_{i}"], on="Sector")
    i += 1

fpi_df = merged_df.copy()
fpi_df['Sector'] = fpi_df['Sector'].str.replace(" ","_")
fpi_df.set_index('Sector', inplace=True)

#sort data as per index (in alphabetical order)
fpi_df.sort_index(axis = 0, inplace=True)
```

- Creating dictionary object containing five consecutive period data.

```
subset_df = {}
for i in range(0,len(fpi_df.columns)):
    start = i
    if i < len(fpi_df.columns)-5:
        end = i + 5
        subset_df[f"{fpi_df.columns[start]}:{fpi_df.columns[end]}"] = fpi_df[fpi_df.col
```

- Calculating consecutive column difference of FPIs to find out, how much the fund allocation for each sector has changed for every 15 days.
- Further calculate average for the past four period in each iteration and save it in a fpi_avg_df.

```
diff_df = {}
for key in subset_df.keys():
    diff_df[key] = subset_df[key].diff(periods=1,axis=1)
    diff_df[key][f"Average_for_{key.split(':')[1]}"]=diff_df[key].iloc[:,1:].mean(axis=
    diff_df[key].reset_index(inplace=True)
    if len(diff_df)==1:
        fpi_avg_df = diff_df[key].drop(diff_df[key].columns[[1,2,3,4,5]], axis=1)
    else:
        fpi_avg_df = pd.merge(fpi_avg_df, diff_df[key].drop(diff_df[key].columns[[1,2,3
```

- Creating a dictionary object for index data to make data handling easier in the later stage.
- Combining indices closing price in a single dataframe

```
index_dict = {}
for i in os.listdir(index_data_dir):
    key = i.split('.')[0].replace(" ","_")
    index_dict[key] = pd.read_csv(index_data_dir+f'/{i}', parse_dates=['Date'])
    try:
        index_dict[key].drop(['P/E','P/B', 'Div Yield','Turnover'], axis=1, inplace=Tru
    except:
        index_dict[key].drop(['Shares Traded','Turnover (Rs. Cr)'], axis=1, inplace=Tru

    if len(index_dict)==1:
        index_close_df = index_dict[key][['Date','Close']]
```

```
        index_close_df.rename(columns={'Close':key},inplace=True)
    elif key!='NIFTY_SMALLCAP_250':
        index_close_df = pd.merge(index_close_df, index_dict[key][['Date','Close']], or
        index_close_df.rename(columns={'Close':key},inplace=True)
```

```
index_close_df.keys()
```

```
Index(['Date', 'NIFTY_PHARMA', 'NIFTY_AUTO', 'Nifty_Private_Bank',
       'NIFTY_OILGAS', 'NIFTY_50', 'NIFTY_FMCG', 'NIFTY_MIDCAP_150',
       'NIFTY_BANK', 'NIFTY_IT', 'NIFTY_METAL', 'NIFTY_NEXT_50'],
      dtype='object')
```

```
index_dict['NIFTY_50'].head()
```

	Date	Open	High	Low	Close	Volume
0	2000-01-03	1482.15	1592.90	1482.15	1592.2	25358322
1	2000-01-04	1594.40	1641.95	1594.40	1638.7	38787872
2	2000-01-05	1634.55	1635.50	1555.05	1595.8	62153431
3	2000-01-06	1595.80	1639.00	1595.80	1617.6	51272875
4	2000-01-07	1616.60	1628.25	1597.20	1613.3	54315945

```
index_dict['NIFTY_50'].tail()
```

	Date	Open	High	Low	Close	Volume
5488	2022-01-21	17613.70	17707.60	17485.85	17617.15	277645373
5489	2022-01-24	17575.15	17599.40	16997.85	17149.10	323847388
5490	2022-01-25	17001.55	17309.15	16836.80	17277.95	326515896
5491	2022-01-27	17062.00	17182.50	16866.75	17110.15	395596577
5492	2022-01-28	17208.30	17373.50	17077.10	17101.95	355284285

```
fpi_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 45 entries, Airlines to Utilities3
Data columns (total 74 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   January152019     45 non-null     object
 1   January312019     45 non-null     object
 2   February152019    45 non-null     object
 3   February282019    45 non-null     object
 4   March152019       45 non-null     object
 5   March312019       45 non-null     object
 6   April152019       45 non-null     object
 7   April302019       45 non-null     object
```

```
8   May152019        45 non-null     object
9   May312019        45 non-null     object
10  June152019       45 non-null     object
11  June302019       45 non-null     object
12  July152019       45 non-null     object
13  July312019       45 non-null     object
14  August152019     45 non-null     object
15  August312019     45 non-null     object
16  September152019  45 non-null     object
17  September302019  45 non-null     object
18  October15,2019   45 non-null     object
19  October31,2019   45 non-null     object
20  November152019   45 non-null     object
21  November302019   45 non-null     object
22  December152019   45 non-null     object
23  December312019   45 non-null     object
24  January152020    45 non-null     object
25  January312020    45 non-null     object
26  February152020   45 non-null     object
27  February292020   45 non-null     object
28  March152020      45 non-null     object
29  March312020      45 non-null     object
30  April152020      45 non-null     object
31  April302020      45 non-null     object
32  May152020        45 non-null     object
33  May312020        45 non-null     object
34  June152020       45 non-null     object
35  June302020       45 non-null     object
36  July152020       45 non-null     object
37  July312020       45 non-null     object
38  August152020     45 non-null     object
39  August312020     45 non-null     object
40  September152020  45 non-null     object
41  September302020  45 non-null     object
42  October152020    45 non-null     object
43  October312020    45 non-null     object
44  November152020   45 non-null     object
45  November302020   45 non-null     object
46  December152020   45 non-null     object
47  December312020   45 non-null     object
48  January152021    45 non-null     object
49  January312021    45 non-null     object
50  February152021   45 non-null     object
```

```
 51  February282021    45 non-null    object
 52  March152021       45 non-null    object
 53  March312021       45 non-null    object
 54  April152021       45 non-null    object
 55  April302021       45 non-null    object
 56  May152021         45 non-null    object
 57  May312021         45 non-null    object
 58  June152021        45 non-null    object
 59  June302021        45 non-null    object
 60  July152021        45 non-null    object
 61  July312021        45 non-null    object
 62  August152021      45 non-null    object
 63  August312021      45 non-null    object
 64  September152021   45 non-null    object
 65  September302021   45 non-null    object
 66  October15,2021    45 non-null    object
 67  October31,2021    45 non-null    object
 68  November15,2021   45 non-null    object
 69  November30,2021   45 non-null    object
 70  December15,2021   45 non-null    object
 71  December31,2021   45 non-null    object
 72  January15,2022    45 non-null    object
 73  January31,2022    45 non-null    object
dtypes: object(74)
memory usage: 26.4+ KB
```

```
fpi_df.head()
```

	January152019	January312019	February152019	February282019	March152019
Sector					
Airlines	6500	7086	6870	6679	7615
Airport_Services	0	0	0	0	0
Automobiles_&_Auto_Components	169820	154934	156794	158610	167050
Banks	544394	546340	542563	547384	600544
Capital_Goods	98627	93736	88096	91977	99262

5 rows × 74 columns

- Now both the data has been converted into the desired format, lets look at the columns.
- In fpi_df,Sector column is set as index and rest of the column consist of Average investment values from 15th January 2019 to 31th January 2022.

```
index_dict['NIFTY_50'].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5493 entries, 0 to 5492
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    5493 non-null   datetime64[ns]
 1   Open    5493 non-null   float64
 2   High    5493 non-null   float64
 3   Low     5493 non-null   float64
 4   Close   5493 non-null   float64
 5   Volume  5493 non-null   int64
dtypes: datetime64[ns](1), float64(4), int64(1)
memory usage: 257.6 KB
```

```
index_dict.keys()
```

```
dict_keys(['NIFTY_PHARMA', 'NIFTY_AUTO', 'Nifty_Private_Bank', 'NIFTY_OILGAS',
'NIFTY_50', 'NIFTY_FMCG', 'NIFTY_MIDCAP_150', 'NIFTY_BANK', 'NIFTY_SMALLCAP_250',
'NIFTY_IT', 'NIFTY_METAL', 'NIFTY_NEXT_50'])
```

```
print(f"List of columns in FPI dataframe: {fpi_df.columns}")
```

```
List of columns in FPI dataframe: Index(['January152019', 'January312019',
'February152019', 'February282019',
       'March152019', 'March312019', 'April152019', 'April302019', 'May152019',
       'May312019', 'June152019', 'June302019', 'July152019', 'July312019',
       'August152019', 'August312019', 'September152019', 'September302019',
       'October15,2019', 'October31,2019', 'November152019', 'November302019',
       'December152019', 'December312019', 'January152020', 'January312020',
       'February152020', 'February292020', 'March152020', 'March312020',
       'April152020', 'April302020', 'May152020', 'May312020', 'June152020',
       'June302020', 'July152020', 'July312020', 'August152020',
       'August312020', 'September152020', 'September302020', 'October152020',
       'October312020', 'November152020', 'November302020', 'December152020',
       'December312020', 'January152021', 'January312021', 'February152021',
       'February282021', 'March152021', 'March312021', 'April152021',
       'April302021', 'May152021', 'May312021', 'June152021', 'June302021',
       'July152021', 'July312021', 'August152021', 'August312021',
       'September152021', 'September302021', 'October15,2021',
       'October31,2021', 'November15,2021', 'November30,2021',
       'December15,2021', 'December31,2021', 'January15,2022',
       'January31,2022'],
      dtype='object')
```

```
print(f"List of columns in Nifty_50 dataframe: {index_dict['NIFTY_50'].columns}")
```

```
List of columns in Nifty_50 dataframe: Index(['Date', 'Open', 'High', 'Low', 'Close',
'Volume'], dtype='object')
```

- Indices data ranges from **3rd Jan 2000 to 28th Jan 2022**.
- The dataset contains over 5493 rows and 6 columns.
- **Open and Close** indicate the opening and closing price of the index on a particular day.
- **High and Low** provide the highest and the lowest price for the index on a particular day, respectively.
- **Volume** indicate the total volume traded on a particular day.
- **Turnover** provide the total value of stocks traded during a specific period of time. The time period may be annually, quarterly, monthly or daily.

Let's now view some basic statistics about the index data frame.

```
index_dict['NIFTY_50'].describe()
```

	Open	High	Low	Close	Volume
count	5493.000000	5493.000000	5493.000000	5493.000000	5.493000e+03
mean	5935.464965	5973.681404	5888.592072	5931.981003	1.927721e+08
std	4060.406773	4073.488083	4036.579030	4055.471502	1.724929e+08
min	853.000000	877.000000	849.950000	854.200000	1.394931e+06
25%	2160.850000	2173.850000	2145.750000	2167.400000	8.232518e+07
50%	5286.600000	5331.800000	5245.500000	5285.000000	1.457869e+08
75%	8537.050000	8588.100000	8494.350000	8530.800000	2.256081e+08
max	18602.350000	18604.450000	18445.300000	18477.050000	1.811564e+09

Missing values

```
index_dict['NIFTY_50'].isnull().sum()
```

```
Date      0
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
```

```
fpi_df.isnull().sum()
```

```
January152019     0
January312019     0
February152019    0
```

```
February282019     0
March152019        0
                  ..
November30,2021    0
December15,2021    0
December31,2021    0
January15,2022     0
January31,2022     0
Length: 74, dtype: int64
```

There are no missing value.

```
jovian.commit()
```

Exploratory Analysis and Visualization

Let's begin by importing `plotly`

```
import plotly.express as px
from plotly.subplots import make_subplots
import cufflinks as cf
import plotly.graph_objects as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
cf.go_offline()
```

Visualising the NIFTY 50 close price from Aug 2019 to Jan 2022

```
nifty_50_2019 = index_dict['NIFTY_50'][index_dict['NIFTY_50']['Date'] >= '2019-08-01']


fig = go.Figure()
fig.add_trace(go.Scatter(x=nifty_50_2019.Date, y=nifty_50_2019.Close,
                        mode='lines', name='Nifty_50'))

fig.update_layout(title='Nifty_50 data 2019-2022',title_x=0.5,
                 xaxis_title='Date', yaxis_title='Price')
```

Monthly Total FPI investment

- Calculate sum for each period

```
total_investment = pd.DataFrame(fpi_df.sum())
```

```
total_investment.reset_index(inplace = True)
total_investment.rename(columns = {'index':'Period',0:'total_investment'},
            inplace = True)
total_investment['category'] = [str(i) for i in total_investment.index]
color_discrete_sequence = ['#609cd4']*len(total_investment)
# 1st Lockdown
color_discrete_sequence[29] = '#ec7c34'

# 2nd Lockdown
color_discrete_sequence[54] = '#ec7c34'

fig = px.bar(total_investment, y='total_investment', x='Period',
             color='category',
             color_discrete_sequence=color_discrete_sequence,
             title='Monthly Total FPI investment',
             labels={'total_investment':'Total_Investment', 'Period':'Period'})

fig.update_layout(uniformtext_minsize=8)
fig.update_layout(xaxis_tickangle=-45, showlegend=False)

fig
```

- The highlighted bars show the announcment of lockdown in India.
- if we comparing both lockdown, we can see that FPI investments decline dramatically in the first lockdown.

Comparing different sectoral indices

- Lets look at the close price of various sectoral indices.

```
d = {}
for key in index_dict.keys():
    if (key == 'NIFTY_50')|(key == 'NIFTY_MIDCAP_150')|\
    (key == 'NIFTY_SMALLCAP_250')|(key == 'NIFTY_NEXT_50')|(key == 'NIFTY_OILGAS'):
        pass
    else:
        temp = index_dict[key][index_dict[key]['Date'] >= '2019-08-01']
        d[f'{key} index'] = temp['Close'].values

indices_df = pd.DataFrame(data=d)
indices_df.index=nifty_50_2019['Date']
```

```
fig = indices_df.iplot(asFigure=True,hline=[0,4], vline=['2020-03-23','2021-04-15'], )

fig.update_layout(title='NIFTY INDICES',title_x=0.5,
                  yaxis_title='Close Price',
                  xaxis_title='Periods',
                  annotations=[dict(x='2020-03-23', y=0.95, xref='x', yref='paper',
                    showarrow=False, xanchor='right', text='First Lockdown'),
                              dict(x='2021-04-15', y=0.95, xref='x', yref='paper',
                    showarrow=False, xanchor='right', text='Second Lockdown')],
)
fig.show()
```

- Now lets from total investment to sector wise fund allocation
- Out of total 45 sectors we will look at those sectors where major changes took place.

Sectorwise Fund Allocation

- Out of total 45 sectors we will compare only those sectors which showed a significant change in fund allocation.

```
# Set sector column as index
# transpose the whole dataframe and then reset the index
fpi_avg_df_t = fpi_avg_df.set_index('Sector').T.reset_index()

# removing underscore from the string and fetch only the date part
fpi_avg_df_t['index'] = fpi_avg_df_t['index'].apply(lambda x: x.split('_')[-1])

# removing extra character from the string
try:
    fpi_avg_df_t['index']= fpi_avg_df_t['index'].apply(lambda x: x.replace(",",""))
except:
    pass

# Parsing strings into datetimes object
fpi_avg_df_t['index'] = fpi_avg_df_t['index'].apply(lambda x: datetime.datetime.strptim
fpi_avg_df_t.rename(columns={'index':'Date'}, inplace = True)
```

```
fig = px.bar(fpi_avg_df_t.iloc[:,:], x='Date', y=['Banks','Other_Financial_Services1',
                                          'Total_Financial_Services','Software_&_Servic
                                          'Telecom_Services','Utilities3'],
      barmode='group',title='Sector wise Fund Allocation')
fig.update_layout(showlegend=False)
fig
```

Overview of Fund allocation for the month of January 2022

```
fig=px.pie(fpi_df['January31,2022'],
           names=fpi_df['January31,2022'].index,
           values=fpi_df['January31,2022'],
      title='January Fund Allocation',
      color_discrete_sequence=px.colors.sequential.RdBu)

fig.update_traces(textposition='inside', textinfo='percent+label')
fig
```

HeatMap to check correlation between variables

```
common_df =index_close_df.copy()
common_df.set_index('Date', inplace=True)
common_df=common_df.rolling(window=30).mean()
common_df = common_df[common_df.index > '2018-02-12']
```

```
temp_fpi_df = fpi_avg_df_t.copy()

# handle saturdays and sunday and moving data ahead by 3 days
bd = pd.tseries.offsets.BusinessDay(n = 3)
temp_fpi_df['Date'] = temp_fpi_df['Date'] + bd
temp_fpi_df.at[temp_fpi_df[temp_fpi_df.Date=='2019-06-05'].index.values[0],'Date']='201
temp_fpi_df.set_index('Date', inplace=True)

#  adding columns from fpi data
common_df = common_df.reindex(columns = common_df.columns.tolist()+temp_fpi_df.columns.

# updating values from fpi to merged data
common_df.update(temp_fpi_df)
```

```
# subseting data from 2019-04-02 because we have limited fpi data
common_df = common_df[common_df.index>'2019-04-02']
```

```
common_df = common_df.ffill(axis=0)
```

```python
# Removing columns with zero values
common_df = common_df.loc[:,(common_df !=0).any(axis=0)]
```

```python
fig = px.imshow(common_df.corr(),color_continuous_scale='Viridis_r',
               )
fig.update_xaxes(side='top',
        tickangle = -50)

fig.layout.height=800
fig.layout.width=800
fig.show()
```

Heatmap on Indices data

```
fig = px.imshow(index_close_df[index_close_df.columns[1:]].corr(),color_continuous_scal
               )
# fig.update_traces(reversescale=False)
fig.update_xaxes(
        tickangle = -50)
fig.layout.height=800
fig.layout.width=800
fig.show()
```

Compare FPIs Oil & Gas sector data and Nifty_OILGAS

```
fig = go.Figure()
# Create figure with secondary y-axis
fig = make_subplots(specs=[[{"secondary_y": True}]])

fig.add_trace(go.Scatter(x=common_df.index, y=common_df['Oil_&_Gas'],
                        mode='lines', name='FPIs Oil_&_Gas'),secondary_y=True)

fig.add_trace(go.Scatter(x=common_df.index, y=common_df.NIFTY_OILGAS,
                        mode='lines', name='NIFTY OILGAS'))

fig.show()
```

OLS Slope

```
fig = px.scatter(common_df[['NIFTY_OILGAS','Oil_&_Gas']],x="Oil_&_Gas", y="NIFTY_OILGAS
                trendline="ols")
fig.show()
```

```
jovian.commit()
```

```
[jovian] Error: Failed to read the Jupyter notebook. Please re-run this cell to try again. If the issue persists, provide the "filename" argument to "jovian.commit" e.g. "jovian.commit(filename='my-notebook.ipynb')"
```

Asking and Answering Questions

Q1: Which sectors is displaying a positive shift as no 30th of September 2021, and do the sectorial indices reflect this?

```
fpi_avg_df[['Sector','Average_for_September302021']][fpi_avg_df['Average_for_September3
```

	Sector	Average_for_September302021
35	Software_&_Services	27016.50
41	Total_Financial_Services	20154.00
27	Other_Financial_Services1	16466.25
26	Oil_&_Gas	15607.50
44	Utilities3	7154.75

```
fig = px.bar(fpi_avg_df_t.iloc[-15:,:], x='Date', y=['Software_&_Services'],
      barmode='group',title='Fund Allocation in Software & Services')
fig.update_layout(showlegend=False)
fig
```

```python
nifty_IT = index_dict['NIFTY_IT'][index_dict['NIFTY_IT']['Date']>'2021-07-1']

fig = go.Figure()
fig.add_trace(go.Scatter(x=nifty_IT.Date, y=nifty_IT.Close,
                        mode='lines+markers', name='Nifty IT'))

fig.update_layout(title='Nifty_IT from September 2021 to January 2022',
                 xaxis_title='Date', yaxis_title='Price')

fig.show()
```

- Check correlation with the two

```
common_df[['NIFTY_IT','Software_&_Services']].corr()
```

	NIFTY_IT	Software_&_Services
NIFTY_IT	1.000000	0.366954
Software_&_Services	0.366954	1.000000

OLS Slope for entire data

```
fig = px.scatter(common_df[['NIFTY_IT','Software_&_Services']], x="Software_&_Services"
                trendline="ols")
fig
```

- The IT industry saw a net inflow of about 27,000 crore between July and September 2021 and the same can be observed in the nifty IT index.
- There is a positive correlation between the two variables and even the OLS plot shows a positive slope.

Q2: Which sectors showed a positive change during first phase of lockdown?

```
fpi_avg_df_t['Date'] = pd.to_datetime(fpi_avg_df_t['Date'], format='%Y-%m-%d')
first_phase =fpi_avg_df_t[(fpi_avg_df_t['Date']>'2020-03-31') & (fpi_avg_df_t['Date']<'
```

```
positive_sector =pd.DataFrame((first_phase[first_phase.columns[1:]]>0).any())
```

```
positive_sector[positive_sector[0]>0].index
```

```
Index(['Household_&_Personal_Products', 'Oil_&_Gas',
       'Pharmaceuticals_&_Biotechnology'],
      dtype='object', name='Sector')
```

```
first_phase[['Date','Household_&_Personal_Products','Oil_&_Gas','Pharmaceuticals_&_Biot
```

Sector	Date	Household_&_Personal_Products	Oil_&_Gas	Pharmaceuticals_&_Biotechnology
25	2020-04-15	-1073.75	-18313.75	-3785.00
26	2020-04-30	948.75	-19359.00	-263.50
27	2020-05-15	698.50	1198.75	3070.25

```
fig = px.bar(first_phase, x='Date', y=['Household_&_Personal_Products','Oil_&_Gas','Pha
       barmode='group',title='First Phase of Lockdown')
fig
```

- Only three sectors exhibited a positive shift at the end of the first phase of lock-down, as seen in the bar chart.

Q3: What was the scenario in Pharma sector before and after the announcement of lock down?

```
before_lockdown = fpi_avg_df_t[(fpi_avg_df_t['Date']>'2019-06-30') & (fpi_avg_df_t['Dat
nifty_Pharma = index_dict['NIFTY_PHARMA'][(index_dict['NIFTY_PHARMA']['Date']>'2019-07-
                                          (index_dict['NIFTY_PHARMA']['Date']<'2020-09-3

fig = px.bar(before_lockdown, x='Date', y='Pharmaceuticals_&_Biotechnology')

fig.add_trace(go.Scatter(x=nifty_Pharma.Date, y=nifty_Pharma.Close,
                        mode='lines', name='Nifty Pharma'))

fig.update_layout(title='Nifty_Pharma vs FPI Investment',
                 xaxis_title='Date', yaxis_title='Price')


fig.show()
```

- Till November 2019 Pharmaceuticals & Biotechnology FPIs have been net sellers for consecutive months.
- It is interesting to see, From December onwards FPIs started pumping in Pharma sector but nifty_pharma index was moving sideways.
- After the first phase of lockdown both the data is showing an up trend.

Q4: Considering the limited FPI data, which sector have shown significant changes ?

```
fig = px.bar(fpi_avg_df_t, x='Date', y=['Total_Financial_Services'],
      barmode='group')
fig
```

```
nifty_pvt_bank = index_dict['Nifty_Private_Bank'][index_dict['Nifty_Private_Bank']['Dat
nifty_pvt_bank.rename(columns={'Close':'Nifty_Private_Bank'},inplace=True)
nifty_bank = index_dict['NIFTY_BANK'][index_dict['NIFTY_BANK']['Date']>'2019-03-31'][['
nifty_bank.rename(columns={'Close':'NIFTY_BANK'},inplace=True)
finance = pd.merge(nifty_pvt_bank, nifty_bank, on="Date")
finance.set_index('Date', inplace=True)


fig = go.Figure()
fig.add_trace(go.Scatter(x=finance.index, y=finance.Nifty_Private_Bank,
```

```
                        mode='lines', name='Nifty Private Bank'))

fig.add_trace(go.Scatter(x=finance.index, y=finance.NIFTY_BANK,
                        mode='lines', name='Nifty Bank'))


fig.show()
```

```
common_df[['Nifty_Private_Bank','NIFTY_BANK','Total_Financial_Services']].corr()
```

	Nifty_Private_Bank	NIFTY_BANK	Total_Financial_Services
Nifty_Private_Bank	1.000000	0.986545	0.378344
NIFTY_BANK	0.986545	1.000000	0.310331
Total_Financial_Services	0.378344	0.310331	1.000000

```
fig = go.Figure()
# Create figure with secondary y-axis
fig = make_subplots(specs=[[{"secondary_y": True}]])

fig.add_trace(go.Scatter(x=common_df.index, y=common_df.Total_Financial_Services,
                        mode='lines', name='Financial Services'),secondary_y=True)

fig.add_trace(go.Scatter(x=common_df.index, y=common_df.Nifty_Private_Bank,
                        mode='lines', name='Nifty Private Bank'))
```

```
fig.add_trace(go.Scatter(x=common_df.index, y=common_df.NIFTY_BANK,
                         mode='lines', name='Nifty Bank'))

fig.show()
```

- Looking at sector wise fund allocation's bar plot we can conclude that financial_service sector has shown significant change in fund allocation by FPIs.
- When ever there is a pump in or pull back of funds by FPIs the same can be seen in the respective indices line chart.

Q5: Which sector has never benefited from foreign investment?

```
positive_sector =pd.DataFrame((fpi_avg_df_t[fpi_avg_df_t.columns[1:]]==0).any())
positive_sector[positive_sector[0]==True].index.tolist()
```

```
['Airport_Services',
 'Diversified_Consumer_Services',
 'Food_&_Drugs_Retailing',
 'Hardware_Technology_&_Equipment',
 'Real_Estate_Investment',
 'Sovereign',
 'Surface_Transportation',
 'Telecommunications_Equipment']
```

- FPIs have made no investment in the sectors listed above.

Q5:Calculate correlation between the indices and FPI report

```
common_df.corr().iloc[11:,:11]
```

	NIFTY_PHARMA	NIFTY_AUTO	Nifty_Private_Bank	NIFTY_OILGAS	NIFTY_50	NIFT
Airlines	0.323048	0.347811	0.258542	0.250706	0.246701	0.
Airport_Services	0.189091	0.380825	0.374654	0.393524	0.345190	0.
Automobiles_&_Auto_Components	0.243128	0.296235	0.137439	0.171586	0.156809	0.
Banks	0.158576	0.274818	0.302437	0.138278	0.161400	0.
Capital_Goods	0.139717	0.183506	0.064592	-0.006402	0.024713	-0.
Chemicals_&_Petrochemicals	0.428906	0.391654	0.302261	0.261495	0.267870	0.
Coal	0.169847	0.377479	0.429667	0.293449	0.250261	0.
Commercial_Services_&_Supplies	0.365553	0.362810	0.242721	0.200510	0.234407	0.
Construction_Materials	0.337975	0.339030	0.347337	0.191076	0.248076	0.
Consumer_Durables	0.526002	0.411662	0.358070	0.370666	0.415569	0.
Diversified2	0.145370	0.091195	0.111556	0.078188	0.117947	0.
Diversified_Consumer_Services	0.447138	0.570830	0.486737	0.601916	0.573070	0.
Food_&_Drugs_Retailing	0.137388	0.199142	0.125753	0.208314	0.203523	0.
Food_Beverages_&_Tobacco	0.321782	0.346516	0.334488	0.327193	0.305458	0.
Forest_Materials	0.504740	0.471497	0.276833	0.416907	0.405982	0.
General_Industrials	0.600376	0.546667	0.361926	0.422762	0.448381	0.
Hardware_Technology_&_Equipment	0.320381	0.477142	0.483949	0.546281	0.520831	0.
Healthcare_Equipment_&_Supplies	0.382344	0.413848	0.288527	0.346781	0.357144	0.
Healthcare_Services	0.580310	0.508631	0.421216	0.515192	0.567233	0.
Hotels_Restaurants_&_Tourism	0.528428	0.495626	0.439653	0.484451	0.500305	0.
Household_&_Personal_Products	0.064659	-0.124664	-0.082881	-0.077673	-0.094754	0.
Insurance	-0.074004	0.023042	0.169047	-0.034282	-0.042789	0.
Logistics	0.657302	0.700247	0.566042	0.616043	0.639281	0.
Marine_Port_&_Services	0.122322	0.211951	0.145108	0.047404	0.060908	0.
Media	0.280294	0.372273	0.238881	0.316300	0.290410	0.
Metals_&_Mining	0.387342	0.293595	0.242011	0.179220	0.207904	0.
Oil_&_Gas	0.199953	0.221720	0.087896	0.280022	0.172049	0.
Other_Financial_Services1	0.324374	0.443116	0.471438	0.315434	0.351209	0.
Others4	0.018848	0.010163	-0.056657	0.011377	-0.008823	0.
Pharmaceuticals_&_Biotechnology	0.108063	-0.188743	-0.388253	-0.209967	-0.250137	-0.
Realty	0.514426	0.684372	0.644254	0.614687	0.624676	0.
Retailing	0.579984	0.609705	0.546827	0.670429	0.693757	0.
Roads_&_Highways	0.352549	0.355913	0.244817	0.238711	0.239942	0.
Shipping	0.105861	0.247968	0.204137	0.131437	0.078940	0.

	NIFTY_PHARMA	NIFTY_AUTO	Nifty_Private_Bank	NIFTY_OILGAS	NIFTY_50	NIFT
Software_&_Services	0.527932	0.367985	0.219525	0.332938	0.362556	0.
Surface_Transportation	0.486321	0.556518	0.358811	0.560803	0.537859	0.
Telecom_Services	0.225339	0.371668	0.471563	0.414715	0.411611	0.
Telecommunications_Equipment	0.612642	0.695379	0.528193	0.748850	0.733199	0.
Textiles_Apparels_&_Accessories	0.619367	0.738427	0.604386	0.714252	0.703841	0.
Total_Financial_Services	0.228313	0.349506	0.378344	0.211726	0.240547	0.
Transport_Related_Services	0.199139	0.333660	0.370910	0.242204	0.218246	0.
Transportation	0.291358	0.412497	0.342129	0.261599	0.254624	0.
Utilities3	0.540661	0.690341	0.602021	0.669470	0.632653	0.

Conclusion

- In this project we have compared FPIs data and Nifty indices data.
- It's worth noting that there's a delay in receiving the FPI Report; it's the 2nd of March,2022 as I write this notebook, and the report for February 28th 2022 is still unavailable.
- It was interesting to examine the pre and post lockdown scenarios,since the pharma sector showed a fund inflow as per FPIs data ,while the nifty pharma index displayed consolidation and following the first phase of lock-down, the pharma index began to rise.
- We also analysed financial data,which demonstrated a high correlation between Nifty Bank and Financial sector.
- As a result, we can conclude that FPI data can assist us in detecting early indications and can be used to get a better understanding of the Indian market before investing.

Ideas for future works

- I have manually imported the FPI data from nsdl using google sheets and downloaded it as xlsx format, instead we can pharse the table using web scraping python library i.e Beautiful Soup or Selenium and maintain a SQL database.
- Further we can use FPIs data as one of the feature in machine learning model to backtest a trading strategy.
- We can also create an interactive dashboard using plotly and deploy it using streamlit or Django.

References

- [Data Analysis with Python: Zero to Pandas] (https://jovian.ml/learn/data-analysis-with-python-zero-to-pandas) -[Nifty Indices Data] (https://www.kaggle.com/atrisaxena/nifty-indices-data) -[Plotly] (https://plotly.com/) -[Pandas] (https://pandas.pydata.org/pandas-docs/stable/index.html)


```
jovian.commit()
```

```
[jovian] Updating notebook "dokeabhishek3/fpi-indices-data-analysis" on
https://jovian.ai
```

```
[jovian] Committed successfully! https://jovian.ai/dokeabhishek3/fpi-indices-data-analysis
```

```
'https://jovian.ai/dokeabhishek3/fpi-indices-data-analysis'
```