FE507 Term Project

1. Data Preperation

1.1 Getting Packages and Data

In [509…

```
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime,timedelta
from numpy import log as ln
import numpy as np
import math
import seaborn as sb
from scipy.stats import kstest, shapiro
import scipy.stats
import statistics
from statsmodels.graphics.tsaplots import plot_acf


# Set-up to have matplotlib use its support for notebook inline plots
%matplotlib inline
pd.options.mode.chained_assignment = None
```

In [375…

```
df_SP500=pd.read_excel('DATA FE507 2021.xlsx',sheet_name='SP500')
df_BIST100=pd.read_excel('DATA FE507 2021.xlsx',sheet_name='BIST100')
df_BISTAll=pd.read_excel('DATA FE507 2021.xlsx',sheet_name='BIST All')
df_BTC_ETH=pd.read_excel('DATA FE507 2021.xlsx', sheet_name='Bitcoin and Ethe
df_Gold=pd.read_excel('DATA FE507 2021.xlsx', sheet_name='Gold')
df_TLUSD=pd.read_excel('DATA FE507 2021.xlsx',sheet_name='TL USD')
df_TRInf=pd.read_excel('DATA FE507 2021.xlsx', sheet_name='TR Inflation')
df_TRRF=pd.read_excel('DATA FE507 2021.xlsx', sheet_name='TR Deposit Rate')
```

In [376…

```
df_BTC_ETH
```

Out[376…

	Date	Bitcoin	Ethereum
0	2014-11-04	324.467934	NaN
1	2014-11-05	328.644408	NaN
2	2014-11-06	337.921358	NaN
3	2014-11-07	348.992860	NaN
4	2014-11-08	341.459753	NaN
…	…	…	…
2591	2021-12-08	50638.162863	4310.180119
2592	2021-12-09	50512.038512	4440.333251
2593	2021-12-10	47594.381934	4106.330000
2594	2021-12-11	47162.324050	3901.851456
2595	2021-12-12	49353.419314	4082.900000

2596 rows × 3 columns

In [377…
```
df_BIST100.head()
```

Out[377…

	Date	Index_V	Market_cap (m TL)
0	1988-01-04	0.07	1
1	1988-01-05	0.07	1
2	1988-01-06	0.07	1
3	1988-01-07	0.07	1
4	1988-01-08	0.07	1

In [378…
```
df_BISTAll.head()
```

Out[378…

	Date	Index_V	Market_cap (m TL)
0	1997-01-02	9.94	2992
1	1997-01-03	10.30	3084
2	1997-01-06	10.50	3143
3	1997-01-07	10.87	3235
4	1997-01-08	11.32	3408

In [379…
```
df_SP500.head()
```

Out[379…

	Date	Index_V	Market_cap (m$)
0	1964-03-31	78.98	357907
1	1964-04-01	79.24	357907
2	1964-04-02	79.70	357907
3	1964-04-03	79.94	357907
4	1964-04-06	80.02	357907

1.2 Preparing Weekly Data

In [380…
```
#define a function finding and tagging the closing days of weeks for USD base
def week_aggregator_1(df):
    for i, row in df.iterrows():
        if i != df.shape[0]-1:
            lead=df['Date'].iloc[i+1]
            if lead!= row[0]+timedelta(days=1):
                df.at[i,'is close day'] = True
            else:
                df.at[i,'is close day'] = False
        else:
            df.at[i,'is close day'] = True

    df=df[(df['is close day'])]
    df['Year_ID']=df.loc[:, ('Date')].apply(lambda x: x.year)
    #I named the weeks by their Monday date
    df['Week_ID']=df.loc[:, ('Date')].apply(lambda x: x- timedelta(days=x.iso
```

```
    del df['Date']
    del df['is close day']
    df.reset_index(inplace=True,drop=True)

    for i, row in df.iterrows():
        if i!=0:
            df.at[i,'Weekly Return'] = ln(row[0]/lag)
        lag=row[0]
    return df
```

In [381…

```
df_Gold_w=df_Gold.pipe(week_aggregator_1)
df_SP500_w=df_SP500.pipe(week_aggregator_1)
```

In [382…

```
#define a function finding and tagging the closing days of weeks for FX based
def week_aggregator_2(df):
    for i, row in df.iterrows():
        if i != df.shape[0]-1:
            lead=df['Date'].iloc[i+1]
            if lead!= row[0]+timedelta(days=1):
                df.at[i,'is close day'] = True
            else:
                df.at[i,'is close day'] = False
        else:
            df.at[i,'is close day'] = True

    df=df[(df['is close day'])]
    df['Year_ID']=df.loc[:, ('Date')].apply(lambda x: x.year)

    #I named the weeks by their Monday date
    df['Week_ID']=df.loc[:, ('Date')].apply(lambda x: x- timedelta(days=x.iso

    del df['is close day']
    df.reset_index(inplace=True,drop=True)
    del df['Date']

    for i, row in df.iterrows():
        if i!=0:
            df.at[i,'Weekly Return in FX'] = ln(row[0]/lag_fx)
            df.at[i,'Weekly Return in USD'] =ln((row[0]/row[2])/lag_usd)
        lag_fx=row[0]
        lag_usd=row[0]/row[2]
    return df
```

In [383…

```
#Merge the Fx rates to the Turkish Currency DataFrames
df_BISTAll=df_BISTAll.merge(df_TLUSD, on='Date', how='left')
df_BIST100=df_BIST100.merge(df_TLUSD, on='Date', how='left')
```

In [384…

```
df_BISTAll_w=df_BISTAll.pipe(week_aggregator_2)
df_BIST100_w=df_BIST100.pipe(week_aggregator_2)
```

In [385…

```
df_BISTAll_w
```

Out[385…

	Index_V	Market_cap (m TL)	TL/USD	Year_ID	Week_ID	Weekly Return in FX	Weekly Return in USD
0	10.30	3084	0.10898	1997	1996-12-30	NaN	NaN

	Index_V	Market_cap (m TL)	TL/USD	Year_ID	Week_ID	Weekly Return in FX	Weekly Return in USD
1	11.42	3424	0.11134	1997	1997-01-06	0.103222	0.081798
2	11.87	3535	0.11355	1997	1997-01-13	0.038648	0.018993
3	14.66	4556	0.11530	1997	1997-01-20	0.211108	0.195814
4	15.63	4685	0.11615	1997	1997-01-27	0.064069	0.056724
...	...	...	...	...	...	...	...
1297	1843.32	1673481	9.98715	2021	2021-11-08	0.027449	-0.001492
1298	1943.97	1764971	11.23000	2021	2021-11-15	0.053164	-0.064126
1299	1997.75	1847589	12.38500	2021	2021-11-22	0.027289	-0.070608
1300	2140.52	1996929	13.73500	2021	2021-11-29	0.069027	-0.034434
1301	2267.86	2090624	13.86000	2021	2021-12-06	0.057788	0.048728

1302 rows × 7 columns

In [386…

```
#define a function finding and tagging the closing days of weeks for USD base
def week_aggregator_3(df):
    df['is close day'] = (df['Date'].apply(lambda x: x.isoweekday())==7)

    df=df[(df['is close day'])]
    df['Year_ID']=df.loc[:, ('Date')].apply(lambda x: x.year)

    #I named the weeks by their Monday date
    df['Week_ID']=df.loc[:, ('Date')].apply(lambda x: x- timedelta(days=x.iso

    del df['is close day']
    df.reset_index(inplace=True,drop=True)
    del df['Date']

    lag_2=''
    for i, row in df.iterrows():
        if i!=0:
            df.at[i,'BTC Weekly Return'] = ln(row[0]/lag_1)
            if not math.isnan(lag_2):
                df.at[i,'ETC Weekly Return'] = ln(row[1]/lag_2)
        lag_1=row[0]
        lag_2=row[1]

    return df
```

In [387…

```
df_BTC_ETH_w=df_BTC_ETH.pipe(week_aggregator_3)
```

In [388…

```
df_BTC_ETH_w
```

Out [388…

	Bitcoin	Ethereum	Year_ID	Week_ID	BTC Weekly Return	ETC Weekly Return
0	344.745289	NaN	2014	2014-11-03	NaN	NaN
1	374.983975	NaN	2014	2014-11-10	0.084077	NaN
2	352.080105	NaN	2014	2014-11-17	-0.063025	NaN
3	375.964613	NaN	2014	2014-11-24	0.065636	NaN
4	375.097528	NaN	2014	2014-12-01	-0.002309	NaN
...	...	...	...	...	...	...
366	64414.909083	4646.859294	2021	2021-11-08	0.045716	0.027196
367	59752.144235	4415.109310	2021	2021-11-15	-0.075140	-0.051159
368	54784.900000	4100.203900	2021	2021-11-22	-0.086790	-0.073996
369	49266.120000	4126.100000	2021	2021-11-29	-0.106178	0.006296
370	49353.419314	4082.900000	2021	2021-12-06	0.001770	-0.010525

371 rows × 6 columns

1.3 Prepare Monthly Data

In [389…

```
#define a function finding and tagging the cloosing days of manths for USD ba
def month_aggregator_1(df):
    for i, row in df.iterrows():
        if i != df.shape[0]-1:
            lead=df['Date'].iloc[i+1]
            if lead.month!= row[0].month:
                df.at[i,'is close day'] = True
            else:
                df.at[i,'is close day'] = False
        else:
                df.at[i,'is close day'] = False

    df=df[(df['is close day'])]
    df['Year_ID']=df.loc[:, ('Date')].apply(lambda x: x.year)

    #I named the weeks by yyyymm
    df['Month_ID']=df.loc[:, ('Date')].apply(lambda x: str(x.year)+str(x.montl

    del df['is close day']
    df.reset_index(inplace=True,drop=True)
    del df['Date']

    for i, row in df.iterrows():
        if i!=0:
            df.at[i,'Monthly Return'] = ln(row[0]/lag)
        lag=row[0]
```

```
    return df
```

In [390…

```
df_Gold_m=df_Gold.pipe(month_aggregator_1)
df_SP500_m=df_SP500.pipe(month_aggregator_1)
```

In [391…

```
#define a function finding and tagging the cloosing days of months for FX bas
def month_aggregator_2(df):
    for i, row in df.iterrows():
        if i != df.shape[0]-1:
            lead=df['Date'].iloc[i+1]
            if lead.month!= row[0].month:
                df.at[i,'is close day'] = True
            else:
                df.at[i,'is close day'] = False
        else:
                df.at[i,'is close day'] = False

    df=df[(df['is close day'])]
    df['Year_ID']=df.loc[:, ('Date')].apply(lambda x: x.year)

    #I named the weeks by yyyymm
    df['Month_ID']=df.loc[:, ('Date')].apply(lambda x: str(x.year)+str(x.month

    del df['is close day']
    df.reset_index(inplace=True,drop=True)
    del df['Date']

    for i, row in df.iterrows():
        if i!=0:
            df.at[i,'Monthly Return in FX'] = ln(row[0]/lag_fx)
            df.at[i,'Monthly Return in USD'] =ln((row[0]/row[2])/lag_usd)
        lag_fx=row[0]
        lag_usd=row[0]/row[2]
    return df
```

In [392…

```
df_BIST100_m=df_BIST100.pipe(month_aggregator_2)
df_BISTAll_m=df_BISTAll.pipe(month_aggregator_2)
```

In [393…

```
df_BIST100_m.head(100)
```

Out[393…

	Index_V	Market_cap (m TL)	TL/USD	Year_ID	Month_ID	Monthly Return in FX	Monthly Return in USD
0	0.09	2	0.00111	1988	19881	NaN	NaN
1	0.07	2	0.00118	1988	19882	-0.251314	-0.312469
2	0.06	2	0.00122	1988	19883	-0.154151	-0.187487
3	0.06	2	0.00126	1988	19884	0.000000	-0.032261
4	0.06	2	0.00132	1988	19885	0.000000	-0.046520
…	…	…	…	…	…	…	…
95	4.00	882	0.06090	1995	199512	0.022757	-0.080235
96	4.95	1222	0.06288	1996	19961	0.213093	0.181098
97	6.05	1445	0.06609	1996	19962	0.200671	0.150881

	Index_V	Market_cap (m TL)	TL/USD	Year_ID	Month_ID	Monthly Return in FX	Monthly Return in USD
98	6.70	1576	0.07088	1996	19963	0.102049	0.032078
99	6.47	1553	0.07522	1996	19964	-0.034931	-0.094360

100 rows × 7 columns

In [394…

```
#define a function finding and tagging the cloosing days of months for USD ba
def month_aggregator_3(df):
    for i, row in df.iterrows():
        if i != df.shape[0]-1:
            lead=df['Date'].iloc[i+1]
            if lead.month!= row[0].month:
                df.at[i,'is close day'] = True
            else:
                df.at[i,'is close day'] = False
        else:
                df.at[i,'is close day'] = False

    df=df[(df['is close day'])]
    df['Year_ID']=df.loc[:, ('Date')].apply(lambda x: x.year)

    #I named the weeks by yyyymm
    df['Month_ID']=df.loc[:, ('Date')].apply(lambda x: str(x.year)+str(x.montl

    del df['is close day']
    df.reset_index(inplace=True,drop=True)
    del df['Date']

    lag_2=''
    for i, row in df.iterrows():
        if i!=0:
            df.at[i,'BTC Monthly Return'] = ln(row[0]/lag_1)
            if not math.isnan(lag_2):
                df.at[i,'ETC Monthly Return'] = ln(row[1]/lag_2)
        lag_1=row[0]
        lag_2=row[1]

    return df
```

In [395…

```
df_BTC_ETH_m=df_BTC_ETH.pipe(month_aggregator_3)
```

In [396…

```
df_BTC_ETH_m.head(10)
```

Out[396…

	Bitcoin	Ethereum	Year_ID	Month_ID	BTC Monthly Return	ETC Monthly Return
0	375.964613	NaN	2014	201411	NaN	NaN
1	310.527122	NaN	2014	201412	-0.191224	NaN
2	226.717513	NaN	2015	20151	-0.314566	NaN
3	254.240418	NaN	2015	20152	0.114576	NaN
4	247.403936	NaN	2015	20153	-0.027258	NaN
5	225.062381	NaN	2015	20154	-0.094645	NaN
6	232.938981	NaN	2015	20155	0.034399	NaN

	Bitcoin	Ethereum	Year_ID	Month_ID	BTC Monthly Return	ETC Monthly Return
7	256.736691	NaN	2015	20156	0.097274	NaN
8	288.278261	NaN	2015	20157	0.115875	NaN
9	228.973938	NaN	2015	20158	-0.230318	NaN

1.4 Merge to Build Master DataFrames and Clear Intermediate Dataframes

In [397…

```
print('number of rows in df_SP500: ' ,df_SP500.shape[0])
print('number of rows in df_BIST100: ' ,df_BIST100.shape[0])
print('number of rows in df_BISTAll: ' ,df_BISTAll.shape[0])
print('number of rows in df_BTC_ETH: ' ,df_BTC_ETH.shape[0])
print('number of rows in df_Gold: ' ,df_Gold.shape[0])
```

```
number of rows in df_SP500:  15054
number of rows in df_BIST100:  8855
number of rows in df_BISTAll:  6507
number of rows in df_BTC_ETH:  2596
number of rows in df_Gold:  14073
```

In [398…

```
print('Max date in df_SP500: ' ,df_SP500['Date'].max(), 'Min date in df_SP500
print('Max date in df_BIST100: ' ,df_BIST100['Date'].max(), 'Min date in df_B
print('Max date in df_BISTAll: ' ,df_BISTAll['Date'].max(), 'Min date in df_B
print('Max date in df_BTC_ETH: ' ,df_BTC_ETH['Date'].max(), 'Min date in df_B
print('Max date in df_Gold: ' ,df_Gold['Date'].max(), 'Min date in df_Gold: '
```

```
Max date in df_SP500:  2021-12-10 00:00:00 Min date in df_SP500:  1964-03-31 0
0:00:00
Max date in df_BIST100:  2021-12-10 00:00:00 Min date in df_BIST100:  1988-01-
04 00:00:00
Max date in df_BISTAll:  2021-12-10 00:00:00 Min date in df_BISTAll:  1997-01-
02 00:00:00
Max date in df_BTC_ETH:  2021-12-12 00:00:00 Min date in df_BTC_ETH:  2014-11-
04 00:00:00
Max date in df_Gold:  2021-12-10 00:00:00 Min date in df_Gold:  1968-01-03 00:
00:00
```

SP500 is the largest data set so we will merge by the referance of SP500 dataset.

1.4.1 Weekly Data

In [399…

```
#start by renaming and dropping colunms
df_SP500_w=df_SP500_w.rename(columns={"Index_V": "SP500 Index", "Market_cap (
                           "Weekly Return":"SP500 Weekly Return"})

df_Gold_w=df_Gold_w.rename(columns={"Price ($/t oz)": "Gold Price ($/t oz)","
del df_Gold_w['Year_ID']
```

In [400…

```
df_BTC_ETH_w=df_BTC_ETH_w.rename(columns={"Bitcoin": "Bitcoin Price","Ethereu
del df_BTC_ETH_w['Year_ID']
```

In [401…

```
df_BIST100_w=df_BIST100_w.rename(columns={"Index_V": "BIST100 Index", "Market_
                           "Weekly Return in FX":"BIST100 Weekly Return in FX"
                              "Weekly Return in USD":"BIST100 Weekly Return in
del df_BIST100_w['Year_ID']
```

```
df_BISTAll_w=df_BISTAll_w.rename(columns={"Index_V": "BISTALL Index", "Market_
                                 "Weekly Return in FX":"BISTALL Weekly Return in FX"
                                    "Weekly Return in USD":"BISTALL Weekly Return in
del df_BISTAll_w['Year_ID']
del df_BISTAll_w['TL/USD']
```

In [402…
```
df_master_w=df_SP500_w.merge(df_Gold_w,on='Week_ID', how='left')
```

In [403…
```
df_master_w=df_master_w.merge(df_BTC_ETH_w,on='Week_ID', how='left')
```

In [404…
```
df_master_w=df_master_w.merge(df_BIST100_w,on='Week_ID', how='left')
```

In [405…
```
df_master_w=df_master_w.merge(df_BISTAll_w,on='Week_ID', how='left')
```

In [406…
```
df_master_w
```

Out[406…

	SP500 Index	SP500 Market_cap (m$)	Year_ID	Week_ID	SP500 Weekly Return	Gold Price ($/t oz)	Gold Weekly Return	Bitcoin Price
0	79.94	357907	1964	1964-03-30	NaN	NaN	NaN	NaN
1	79.85	357907	1964	1964-04-06	-0.001126	NaN	NaN	NaN
2	80.55	357907	1964	1964-04-13	0.008728	NaN	NaN	NaN
3	79.75	357907	1964	1964-04-20	-0.009981	NaN	NaN	NaN
4	80.17	363019	1964	1964-04-27	0.005253	NaN	NaN	NaN
…	…	…	…	…	…	…	…	…
3006	4682.85	39739140	2021	2021-11-08	-0.003130	1862.36	0.028258	64414.909083
3007	4697.96	39866160	2021	2021-11-15	0.003221	1859.66	-0.001451	59752.144235
3008	4594.62	38990540	2021	2021-11-22	-0.022242	1799.35	-0.032968	54784.900000
3009	4538.43	38520320	2021	2021-11-29	-0.012305	1774.90	-0.013681	49266.120000
3010	4712.02	39993410	2021	2021-12-06	0.037536	1784.90	0.005618	49353.419314

3011 rows × 20 columns

1.4.2 Monthly Data

In [407…
```
#start by renaming and dropping colunms
df_SP500_m=df_SP500_m.rename(columns={"Index_V": "SP500 Index", "Market_cap (
```

```
                          "Monthly Return":"SP500 Monthly Return"})

df_Gold_m=df_Gold_m.rename(columns={"Price ($/t oz)": "Gold Price ($/t oz)","
del df_Gold_m['Year_ID']
```

In [408…

```
df_BTC_ETH_m=df_BTC_ETH_m.rename(columns={"Bitcoin": "Bitcoin Price","Ethereu
del df_BTC_ETH_m['Year_ID']
```

In [409…

```
df_BIST100_m=df_BIST100_m.rename(columns={"Index_V": "BIST100 Index", "Market_
                          "Monthly Return in FX":"BIST100 Monthly Return in F
                             "Monthly Return in USD":"BIST100 Monthly Return
del df_BIST100_m['Year_ID']
df_BISTAll_m=df_BISTAll_m.rename(columns={"Index_V": "BISTALL Index", "Market_
                          "Monthly Return in FX":"BISTALL Monthly Return in F
                             "Monthly Return in USD":"BISTALL Monthly Return
del df_BISTAll_m['Year_ID']
del df_BISTAll_m['TL/USD']
```

In [410…

```
df_master_m=df_SP500_m.merge(df_Gold_m,on='Month_ID', how='left')
```

In [411…

```
df_master_m=df_master_m.merge(df_BTC_ETH_m,on='Month_ID', how='left')
```

In [412…

```
df_master_m=df_master_m.merge(df_BIST100_m,on='Month_ID', how='left')
```

In [413…

```
df_master_m=df_master_m.merge(df_BISTAll_m,on='Month_ID', how='left')
```

In [414…

```
df_master_m
```

Out[414…

	SP500 Index	SP500 Market_cap (m$)	Year_ID	Month_ID	SP500 Monthly Return	Gold Price ($/t oz)	Gold Monthly Return	Bitcoin Price
0	78.98	357907	1964	19643	NaN	NaN	NaN	NaN
1	79.46	363019	1964	19644	0.006059	NaN	NaN	NaN
2	80.37	365840	1964	19645	0.011387	NaN	NaN	NaN
3	81.69	372598	1964	19646	0.016291	NaN	NaN	NaN
4	83.18	378276	1964	19647	0.018075	NaN	NaN	NaN
…	…	…	…	…	…	…	…	…
688	4395.26	37213730	2021	20217	0.022493	1823.24	0.032221	42209.900000
689	4522.68	38312130	2021	20218	0.028578	1806.30	-0.009335	46992.592288
690	4307.54	36538380	2021	20219	-0.048738	1760.95	-0.025427	41538.222404
691	4605.38	39073900	2021	202110	0.066858	1776.81	0.008966	61864.893408
692	4567.00	38756190	2021	202111	-0.008369	1780.05	0.001822	57834.357549

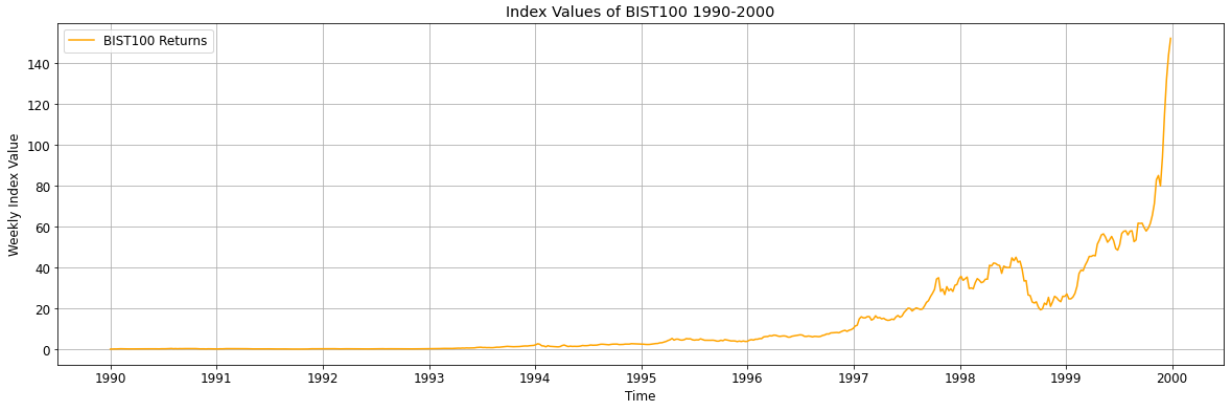
693 rows × 20 columns

2. Analysis

2.1 Graphical Analysis of BIST100 and SP500

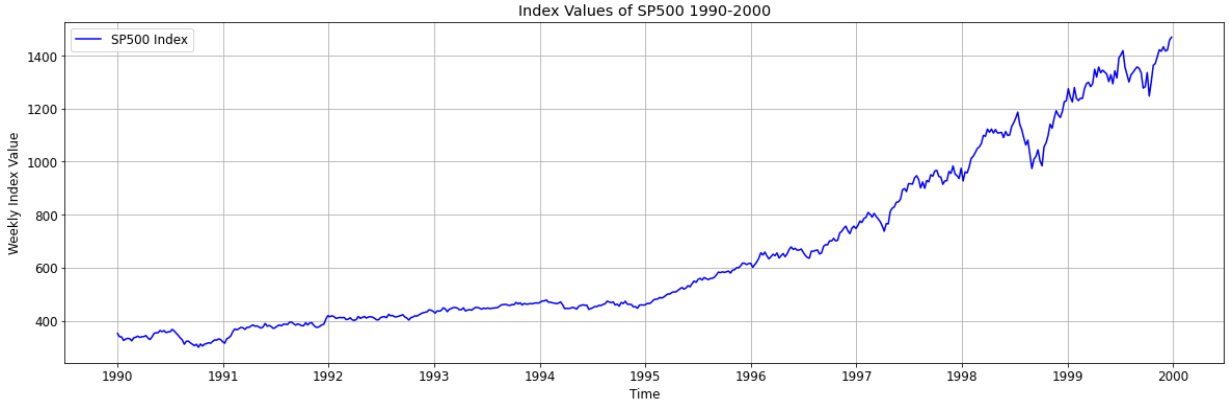
In [417…

```
plt.rc('font', size=12)
fig, ax = plt.subplots(figsize=(20, 6))

# Specify how our lines should look
ax.plot(df_master_w[(df_master_w["Year_ID"] >=1990) & (df_master_w["Year_ID"]
        df_master_w[(df_master_w["Year_ID"] >=1990) & (df_master_w["Year_ID"]
        color='orange', label='BIST100 Returns')

ax.set_xlabel('Time')
ax.set_ylabel('Weekly Index Value')
ax.set_title('Index Values of BIST100 1990-2000')
ax.grid(True)
ax.legend(loc='upper left');
```



In [424…

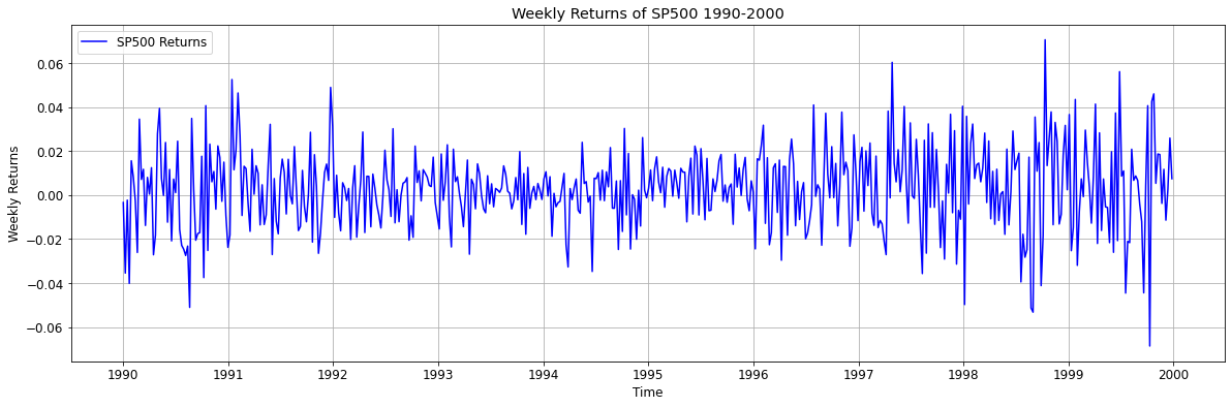
```
plt.rc('font', size=12)
fig, ax = plt.subplots(figsize=(20, 6))

# Specify how our lines should look
ax.plot(df_master_w[(df_master_w["Year_ID"] >=1990) & (df_master_w["Year_ID"]
        df_master_w[(df_master_w["Year_ID"] >=1990) & (df_master_w["Year_ID"]
        color='blue', label='SP500 Index')

ax.set_xlabel('Time')
ax.set_ylabel('Weekly Index Value')
ax.set_title('Index Values of SP500 1990-2000')
ax.grid(True)
ax.legend(loc='upper left');
```



In [419…

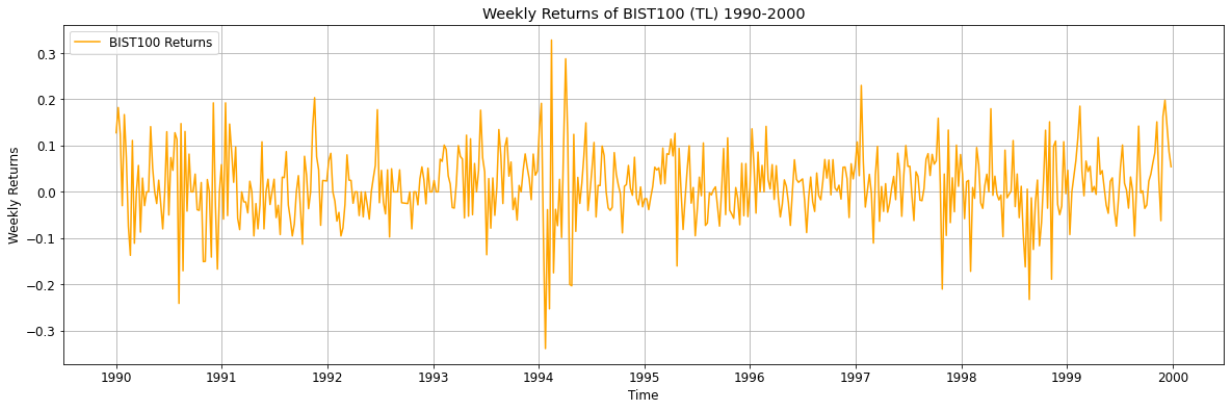
```
plt.rc('font', size=12)
```

```
fig, ax = plt.subplots(figsize=(20, 6))

# Specify how our lines should look
ax.plot(df_master_w[(df_master_w["Year_ID"] >=1990) & (df_master_w["Year_ID"]
        df_master_w[(df_master_w["Year_ID"] >=1990) & (df_master_w["Year_ID"]
        color='blue', label='SP500 Returns')

ax.set_xlabel('Time')
ax.set_ylabel('Weekly Returns')
ax.set_title('Weekly Returns of SP500 1990-2000')
ax.grid(True)
ax.legend(loc='upper left');
```



In [434…

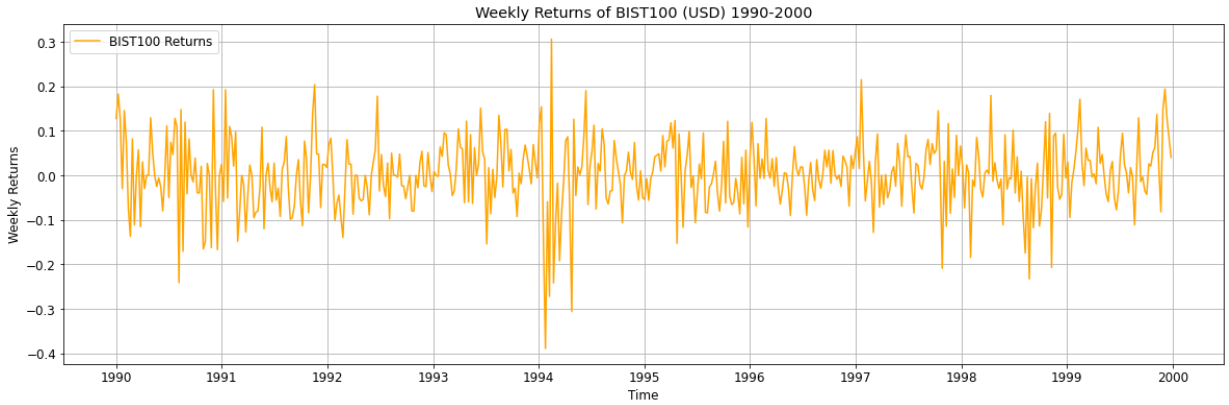
```
plt.rc('font', size=12)
fig, ax = plt.subplots(figsize=(20, 6))

# Specify how our lines should look
ax.plot(df_master_w[(df_master_w["Year_ID"] >=1990) & (df_master_w["Year_ID"]
        df_master_w[(df_master_w["Year_ID"] >=1990) & (df_master_w["Year_ID"]
        color='orange', label='BIST100 Returns')

ax.set_xlabel('Time')
ax.set_ylabel('Weekly Returns')
ax.set_title('Weekly Returns of BIST100 (TL) 1990-2000')
ax.grid(True)
ax.legend(loc='upper left');
```



In [437…

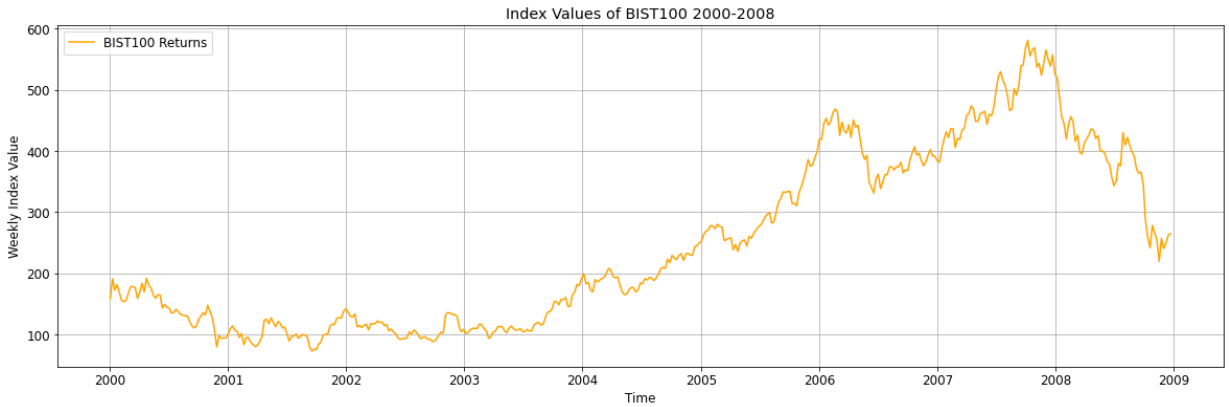
```
plt.rc('font', size=12)
fig, ax = plt.subplots(figsize=(20, 6))

# Specify how our lines should look
ax.plot(df_master_w[(df_master_w["Year_ID"] >=1990) & (df_master_w["Year_ID"]
        df_master_w[(df_master_w["Year_ID"] >=1990) & (df_master_w["Year_ID"]
        color='orange', label='BIST100 Returns')
```

```
ax.set_xlabel('Time')
ax.set_ylabel('Weekly Returns')
ax.set_title('Weekly Returns of BIST100 (USD) 1990-2000')
ax.grid(True)
ax.legend(loc='upper left');
```



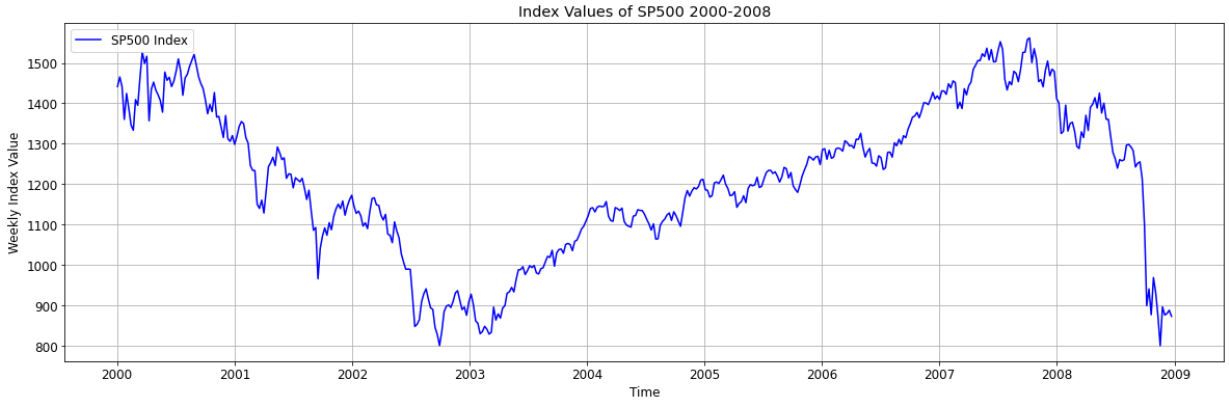
In [421…

```
plt.rc('font', size=12)
fig, ax = plt.subplots(figsize=(20, 6))

# Specify how our lines should look
ax.plot(df_master_w[(df_master_w["Year_ID"] >=2000) & (df_master_w["Year_ID"]
        df_master_w[(df_master_w["Year_ID"] >=2000) & (df_master_w["Year_ID"]
        color='orange', label='BIST100 Returns')

ax.set_xlabel('Time')
ax.set_ylabel('Weekly Index Value')
ax.set_title('Index Values of BIST100 2000-2008')
ax.grid(True)
ax.legend(loc='upper left');
```



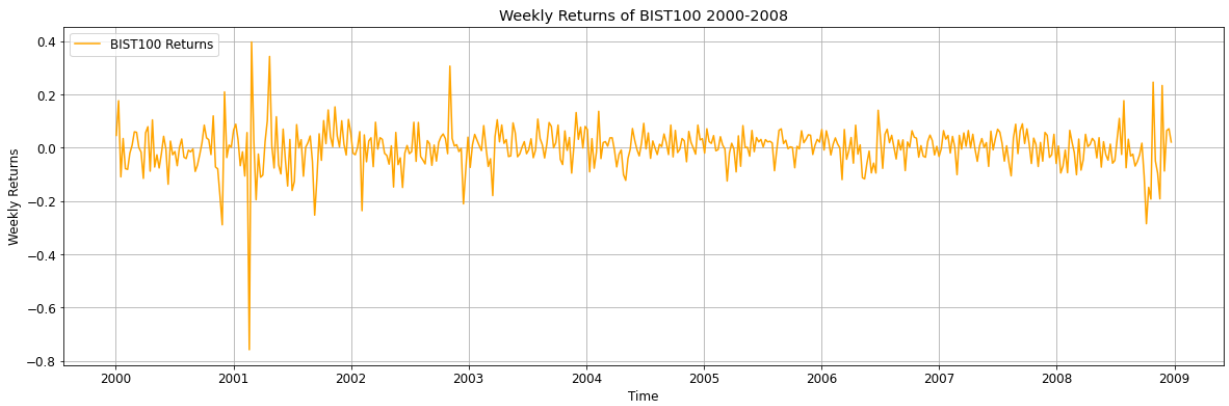
In [432…

```
plt.rc('font', size=12)
fig, ax = plt.subplots(figsize=(20, 6))

# Specify how our lines should look
ax.plot(df_master_w[(df_master_w["Year_ID"] >=2000) & (df_master_w["Year_ID"]
        df_master_w[(df_master_w["Year_ID"] >=2000) & (df_master_w["Year_ID"]
        color='blue', label='SP500 Index')

ax.set_xlabel('Time')
ax.set_ylabel('Weekly Index Value')
ax.set_title('Index Values of SP500 2000-2008')
ax.grid(True)
ax.legend(loc='upper left');
```



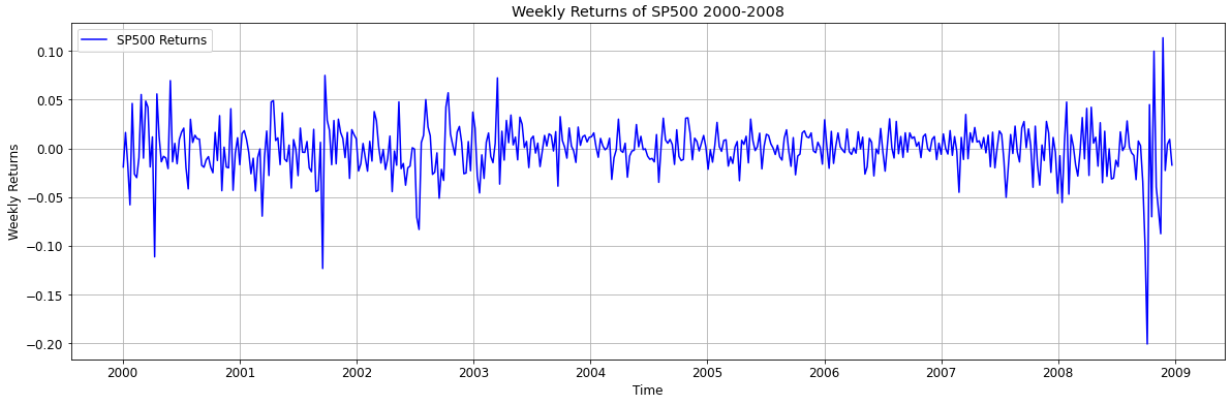
In [438…

```
plt.rc('font', size=12)
fig, ax = plt.subplots(figsize=(20, 6))

# Specify how our lines should look
ax.plot(df_master_w[(df_master_w["Year_ID"] >=2000) & (df_master_w["Year_ID"]
        df_master_w[(df_master_w["Year_ID"] >=2000) & (df_master_w["Year_ID"]
        color='orange', label='BIST100 Returns')

ax.set_xlabel('Time')
ax.set_ylabel('Weekly Returns')
ax.set_title('Weekly Returns of BIST100 2000-2008')
ax.grid(True)
ax.legend(loc='upper left');
```



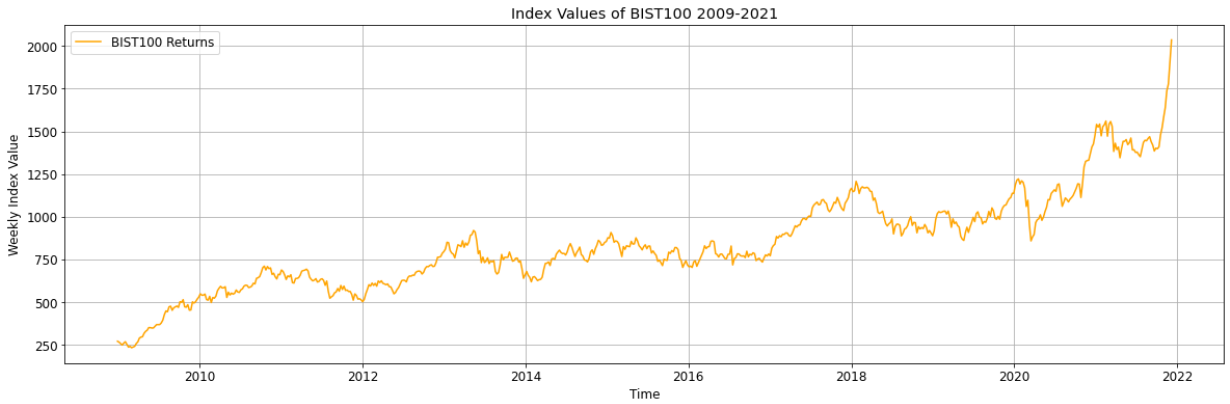
In [426…

```
plt.rc('font', size=12)
fig, ax = plt.subplots(figsize=(20, 6))

# Specify how our lines should look
ax.plot(df_master_w[(df_master_w["Year_ID"] >=2000) & (df_master_w["Year_ID"]
        df_master_w[(df_master_w["Year_ID"] >=2000) & (df_master_w["Year_ID"]
        color='blue', label='SP500 Returns')

ax.set_xlabel('Time')
ax.set_ylabel('Weekly Returns')
ax.set_title('Weekly Returns of SP500 2000-2008')
ax.grid(True)
ax.legend(loc='upper left');
```



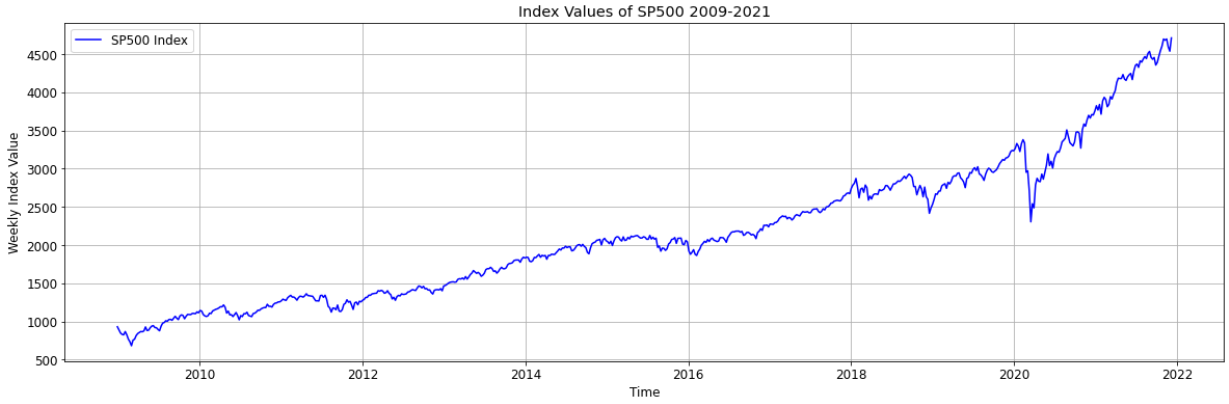
In [427…

```
plt.rc('font', size=12)
fig, ax = plt.subplots(figsize=(20, 6))

# Specify how our lines should look
ax.plot(df_master_w[(df_master_w["Year_ID"] >=2009) & (df_master_w["Year_ID"]
        df_master_w[(df_master_w["Year_ID"] >=2009) & (df_master_w["Year_ID"]
        color='orange', label='BIST100 Returns')

ax.set_xlabel('Time')
ax.set_ylabel('Weekly Index Value')
ax.set_title('Index Values of BIST100 2009-2021')
ax.grid(True)
ax.legend(loc='upper left');
```



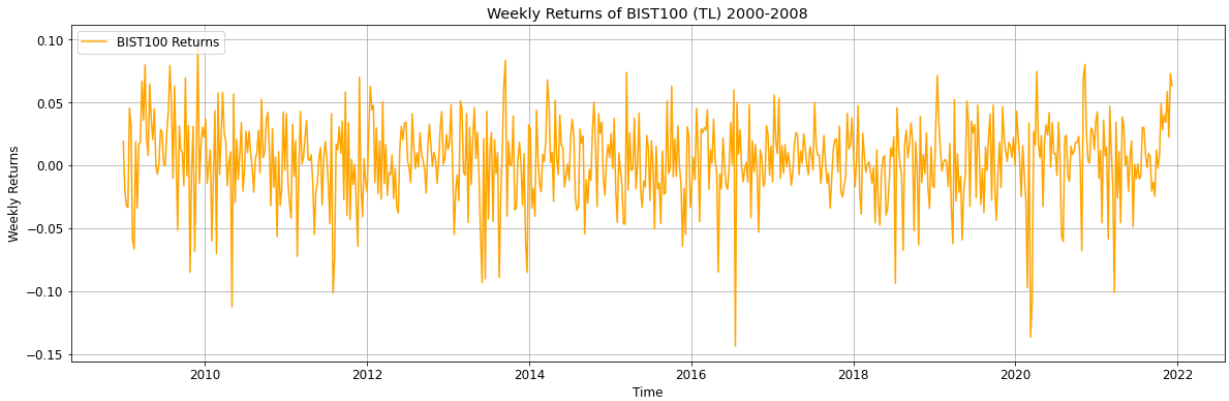
In [428…

```
plt.rc('font', size=12)
fig, ax = plt.subplots(figsize=(20, 6))

# Specify how our lines should look
ax.plot(df_master_w[(df_master_w["Year_ID"] >=2009) & (df_master_w["Year_ID"]
        df_master_w[(df_master_w["Year_ID"] >=2009) & (df_master_w["Year_ID"]
        color='blue', label='SP500 Index')

ax.set_xlabel('Time')
ax.set_ylabel('Weekly Index Value')
ax.set_title('Index Values of SP500 2009-2021')
ax.grid(True)
ax.legend(loc='upper left');
```



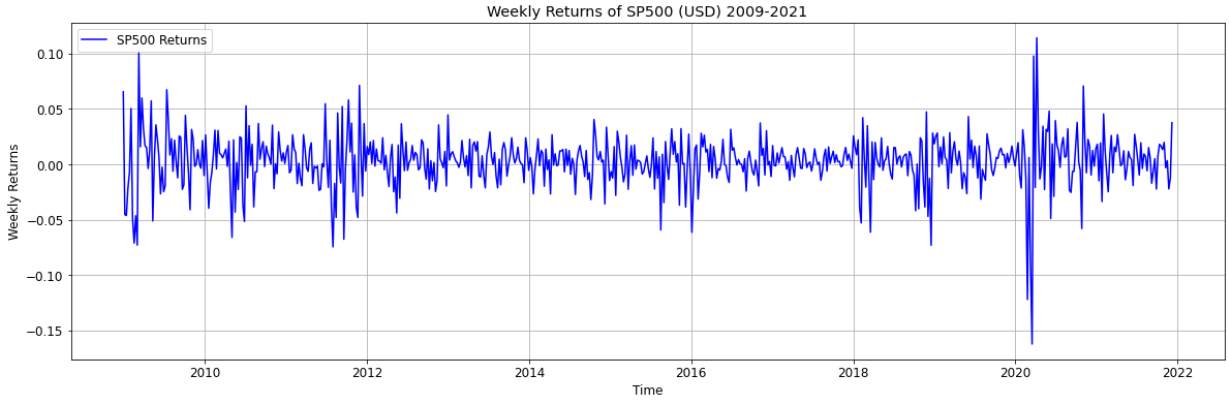
In [440…

```
plt.rc('font', size=12)
fig, ax = plt.subplots(figsize=(20, 6))

# Specify how our lines should look
ax.plot(df_master_w[(df_master_w["Year_ID"] >=2009) & (df_master_w["Year_ID"]
        df_master_w[(df_master_w["Year_ID"] >=2009) & (df_master_w["Year_ID"]
        color='orange', label='BIST100 Returns')

ax.set_xlabel('Time')
ax.set_ylabel('Weekly Returns')
ax.set_title('Weekly Returns of BIST100 (TL) 2000-2008')
ax.grid(True)
ax.legend(loc='upper left');
```



In [433…

```
plt.rc('font', size=12)
fig, ax = plt.subplots(figsize=(20, 6))

# Specify how our lines should look
ax.plot(df_master_w[(df_master_w["Year_ID"] >=2009) & (df_master_w["Year_ID"]
        df_master_w[(df_master_w["Year_ID"] >=2009) & (df_master_w["Year_ID"]
        color='blue', label='SP500 Returns')

ax.set_xlabel('Time')
ax.set_ylabel('Weekly Returns')
ax.set_title('Weekly Returns of SP500 (USD) 2009-2021')
ax.grid(True)
ax.legend(loc='upper left');
```



2.2 Statistical Analysis of BIST100, BISTALL, Gold, BTC and SP500

In [53]:
```
#Means
df_master_w[(df_master_w["Year_ID"] >=2015)][['SP500 Weekly Return','BIST100 
            'BTC Weekly Return']].mean(axis = 0, skipna = False)
```

Out[53]:
```
SP500 Weekly Return            0.002241
BIST100 Weekly Return in FX    0.002407
BISTALL Weekly Return in FX    0.002712
Gold Weekly Return             0.001109
BTC Weekly Return              0.013919
dtype: float64
```

In [54]:
```
#Variance
df_master_w[(df_master_w["Year_ID"] >=2015)][['SP500 Weekly Return','BIST100 
            'BTC Weekly Return']].var(axis = 0, skipna = False)
```

Out[54]:
```
SP500 Weekly Return            0.000550
BIST100 Weekly Return in FX    0.001005
BISTALL Weekly Return in FX    0.000970
Gold Weekly Return             0.000388
BTC Weekly Return              0.011842
dtype: float64
```

In [153…
```
#Standart Deviation
df_master_w[(df_master_w["Year_ID"] >=2015)][['SP500 Weekly Return','BIST100 
            'BTC Weekly Return']].std(axis = 0, skipna = False)
```

Out[153…
```
SP500 Weekly Return            0.023454
BIST100 Weekly Return in FX    0.031704
BISTALL Weekly Return in FX    0.031142
Gold Weekly Return             0.019687
BTC Weekly Return              0.108821
dtype: float64
```

In [56]:
```
#Minimum
df_master_w[(df_master_w["Year_ID"] >=2015)][['SP500 Weekly Return','BIST100 
            'BTC Weekly Return']].min(axis = 0, skipna = False)
```

Out[56]:
```
SP500 Weekly Return           -0.162279
BIST100 Weekly Return in FX   -0.143709
BISTALL Weekly Return in FX   -0.153627
Gold Weekly Return            -0.085737
BTC Weekly Return             -0.543942
dtype: float64
```

In [57]:
```
#Maximum
df_master_w[(df_master_w["Year_ID"] >=2015)][['SP500 Weekly Return','BIST100 
            'BTC Weekly Return']].max(axis = 0, skipna = False)
```

Out[57]:
```
SP500 Weekly Return            0.114237
BIST100 Weekly Return in FX    0.080043
BISTALL Weekly Return in FX    0.078798
Gold Weekly Return             0.090004
BTC Weekly Return              0.361871
dtype: float64
```

In [58]:
```
#define function to calculate interquartile range
def find_iqr(x):
  return np.subtract(*np.percentile(x, [75, 25]))
```

In [59]:
```
#Interquartile Range
df_master_w[(df_master_w["Year_ID"] >=2015)][['SP500 Weekly Return','BIST100 
            'BTC Weekly Return']].apply(find_iqr)
```

Out[59]:
```
SP500 Weekly Return            0.021827
BIST100 Weekly Return in FX    0.036944
BISTALL Weekly Return in FX    0.036087
Gold Weekly Return             0.023653
BTC Weekly Return              0.104727
dtype: float64
```

In [60]:
```
#Skewness
df_master_w[(df_master_w["Year_ID"] >=2015)][['SP500 Weekly Return','BIST100 
            'BTC Weekly Return']].skew()
```

Out[60]:
```
SP500 Weekly Return           -1.324548
BIST100 Weekly Return in FX   -0.874928
BISTALL Weekly Return in FX   -1.033347
Gold Weekly Return            -0.097260
BTC Weekly Return             -0.391917
dtype: float64
```

In [61]:
```
#Kurtosis
df_master_w[(df_master_w["Year_ID"] >=2015)][['SP500 Weekly Return','BIST100 
            'BTC Weekly Return']].kurtosis()
```

Out[61]:
```
SP500 Weekly Return            10.947682
BIST100 Weekly Return in FX     2.480672
BISTALL Weekly Return in FX     3.089286
Gold Weekly Return              2.183469
BTC Weekly Return               2.441050
dtype: float64
```

In [62]:
```
def df_autocorr(df, lag=1, axis=0):
    """Compute full-sample column-wise autocorrelation for a DataFrame."""
    return df.apply(lambda col: col.autocorr(lag), axis=axis)
```

In [63]:
```
#First-Order Autocorrelation
df_autocorr(df_master_w[(df_master_w["Year_ID"] >=2015)][['SP500 Weekly Retur
            'BTC Weekly Return']])
```

Out[63]:
```
SP500 Weekly Return           -0.095930
BIST100 Weekly Return in FX   -0.008243
BISTALL Weekly Return in FX   -0.003232
```

```
Gold Weekly Return            -0.003761
BTC Weekly Return             -0.013333
dtype: float64
```

In [151…

```
#Covariance
df_master_w[(df_master_w["Year_ID"] >=2015)][['SP500 Weekly Return','BIST100 
            'BTC Weekly Return']].cov()
```

Out[151…

	SP500 Weekly Return	BIST100 Weekly Return in FX	BISTALL Weekly Return in FX	Gold Weekly Return	BTC Weekly Return
SP500 Weekly Return	0.000550	0.000325	0.000328	0.000066	0.000359
BIST100 Weekly Return in FX	0.000325	0.001005	0.000983	0.000110	0.000467
BISTALL Weekly Return in FX	0.000328	0.000983	0.000970	0.000114	0.000502
Gold Weekly Return	0.000066	0.000110	0.000114	0.000388	0.000144
BTC Weekly Return	0.000359	0.000467	0.000502	0.000144	0.011842

In [149…

```
#Correlation
df_master_w[(df_master_w["Year_ID"] >=2015)][['SP500 Weekly Return','BIST100 
            'BTC Weekly Return']].corr()
```

Out[149…

	SP500 Weekly Return	BIST100 Weekly Return in FX	BISTALL Weekly Return in FX	Gold Weekly Return	BTC Weekly Return
SP500 Weekly Return	1.000000	0.436467	0.449521	0.142453	0.140780
BIST100 Weekly Return in FX	0.436467	1.000000	0.995505	0.176752	0.135227
BISTALL Weekly Return in FX	0.449521	0.995505	1.000000	0.185621	0.147998
Gold Weekly Return	0.142453	0.176752	0.185621	1.000000	0.067437
BTC Weekly Return	0.140780	0.135227	0.147998	0.067437	1.000000

In [464…

```
#Geometric Mean
g1,g2,g3,g4,g5=1,1,1,1,1
for i, row in df_master_w[(df_master_w["Year_ID"] >=2015)][['SP500 Weekly Ret
            'BTC Weekly Return']].iterrows():
    g1*=row[0]
    g2*=row[1]
    g3*=row[2]
    g4*=row[3]
    g5*=row[4]
t=df_master_w[(df_master_w["Year_ID"] >=2015)][['SP500 Weekly Return','BIST10
            'BTC Weekly Return']].shape[0]
g1=g1**(1/t)
g2=g2**(1/t)
g3=g3**(1/t)
```

```
g4=g4**(1/t)
g5=g5**(1/t)
print('Geometric Mean of SP500 Weekly Return = ' ,g1)
print('Geometric Mean of BIST100 Weekly Return  in TL= ' ,g2)
print('Geometric Mean of BISTALL Weekly Return  in TL= ' ,g3)
print('Geometric Mean of Gold Weekly Return = ' ,g4)
print('Geometric Mean of BTC Weekly Return       =' ,g5)
```

```
Geometric Mean of SP500 Weekly Return =  0.0
Geometric Mean of BIST100 Weekly Return  in TL=  0.0
Geometric Mean of BISTALL Weekly Return  in TL=  0.0
Geometric Mean of Gold Weekly Return =  0.0
Geometric Mean of BTC Weekly Return      = 0.0
```

In [452…

```
t=df_master_w[(df_master_w["Year_ID"] >=2015)][['SP500 Weekly Return','BIST10
           'BTC Weekly Return']].shape[0]
g1=g1**(1/t)
g2=g2**(1/t)
g3=g3**(1/t)
g4=g4**(1/t)
g5=g5**(1/t)
print('Geometric Mean of SP500 Weekly Return = %.3f' ,g1)
print('Geometric Mean of BIST100 Weekly Return  in TL= %.3f' ,g2)
print('Geometric Mean of BISTALL Weekly Return  in TL= %.3f' ,g3)
print('Geometric Mean of Gold Weekly Return = %.3f' ,g4)
print('Geometric Mean of BTC Weekly Return       %.3f=' ,g5)
```

Out[452…

```
363
```

2.3.a Moving Sample Analysis of BIST100

In [470…

```
#define a function finding Moving Avarages for 52 rolling Period
def MA52_finder_w(df):
    for i, row in df.iterrows():
        if i >52:
            total=0
            total_dif=0
            for j in range(1,53):
                total+=df['BIST100 Weekly Return in FX'].iloc[i-j]
            df.at[i,'MA52'] = total/52

            for j in range(1,53):
                total_dif+=(total/52-df['BIST100 Weekly Return in FX'].iloc[i
            df.at[i,'Sigma_MA52']=(total_dif/52)**(1/2)

    return df
```

In [472…

```
df_BIST100_w_MA52=df_master_w[(df_master_w["Year_ID"] >=1990)][['Year_ID','We
df_BIST100_w_MA52.reset_index(inplace=True,drop=True)
```

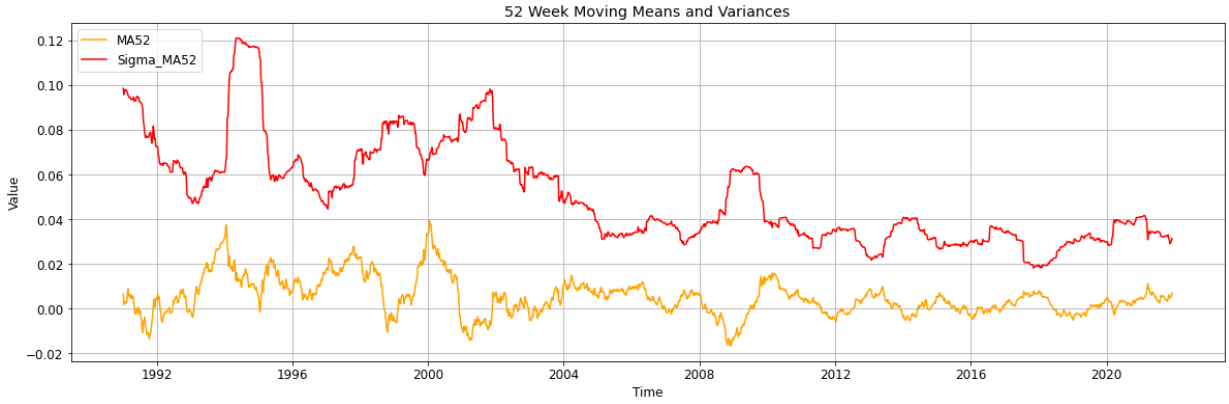
In [473…

```
df_BIST100_w_MA52=df_BIST100_w_MA52.pipe(MA52_finder_w)
```

In [477…

```
plt.rc('font', size=12)
fig, ax = plt.subplots(figsize=(20, 6))

# Specify how our lines should look
ax.plot(df_BIST100_w_MA52[(df_BIST100_w_MA52["Year_ID"] >=1990)]['Week_ID'],
```

```
        df_BIST100_w_MA52[(df_BIST100_w_MA52["Year_ID"] >=1990)]['MA52'],
        color='orange', label='MA52')
ax.plot(df_BIST100_w_MA52[(df_BIST100_w_MA52["Year_ID"] >=1990)]['Week_ID'],
        df_BIST100_w_MA52[(df_BIST100_w_MA52["Year_ID"] >=1990)]['Sigma_MA52'
        color='red', label='Sigma_MA52')

ax.set_xlabel('Time')
ax.set_ylabel('Value')
ax.set_title('52 Week Moving Means and Variances of BIST100 for 1990-2021')
ax.grid(True)
ax.legend(loc='upper left');
plt.show()
plt.show()
```



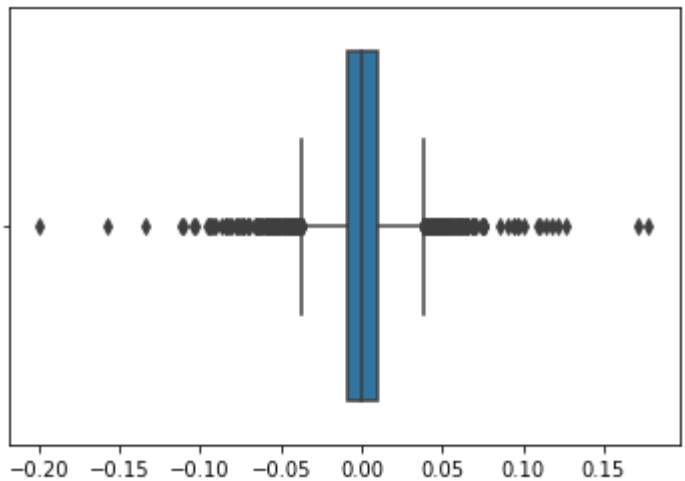
2.3.b Empirical Distribution of Daily and Monthly Returns for 2001-2021

In [242…

```
np_BIST100=np.array([])
df_BIST100_q3=df_BIST100[(df_BIST100["Date"].apply(lambda x: x.year) >=2000)]
df_BIST100_q3.reset_index(inplace=True,drop=True)
for i, row in df_BIST100_q3.iterrows():
    if i ==0:
        lag=row[1]
    else:
        np_BIST100=np.append(np_BIST100, [ln(row[1]/lag)])
        lag=row[1]
```

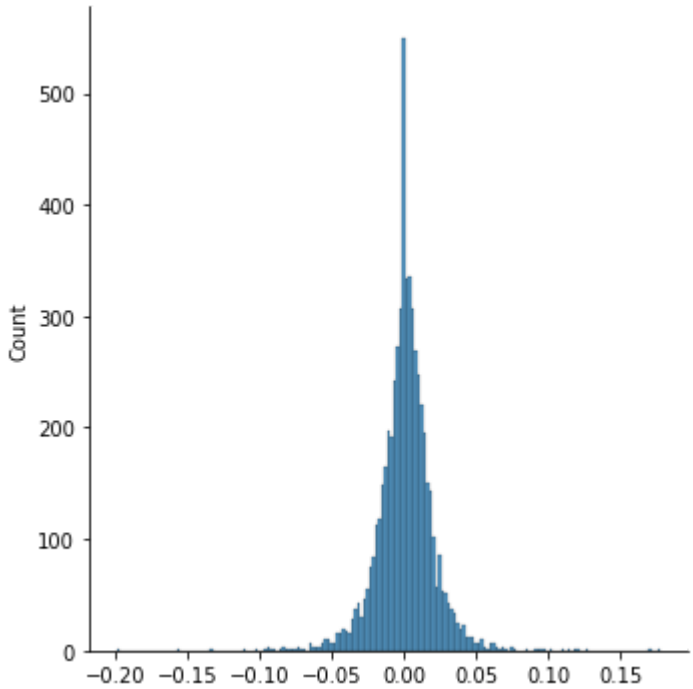
In [244…

```
ax=sb.boxplot(x=np_BIST100)
```



In [245…

```
bx=sb.displot(data=np_BIST100)
```



In [248…

```
statistic,pvalue=kstest(np_BIST100,'norm')
print('Staistic=%.3f, Pvalue= %.3f \n' %(statistic,pvalue))
if pvalue>0.05:
    print('Gausian with confidance interval 95%')
else:
    print('Not Gausian with confidance interval 95%')
```

```
Staistic=0.469, Pvalue= 0.000

Not Gausian with confidance interval 95%
```

In [478…

```
df_BIST100_m
```

Out[478…

	BIST100 Index	BIST100 Market_cap (m TL)	TL/USD	Month_ID	BIST100 Monthly Return in FX	BIST100 Monthly Return in USD
0	0.09	2	0.00111	19881	NaN	NaN
1	0.07	2	0.00118	19882	-0.251314	-0.312469
2	0.06	2	0.00122	19883	-0.154151	-0.187487
3	0.06	2	0.00126	19884	0.000000	-0.032261
4	0.06	2	0.00132	19885	0.000000	-0.046520
…	…	…	…	…	…	…
402	1392.91	1046037	8.42200	20217	0.026605	0.057643
403	1472.07	1110546	8.31450	20218	0.055274	0.068121
404	1406.39	1065637	8.88425	20219	-0.045643	-0.111923
405	1522.04	1200585	9.61515	202110	0.079025	-0.000035
406	1809.65	1463126	13.27775	202111	0.173082	-0.149668

407 rows × 6 columns

In [488…

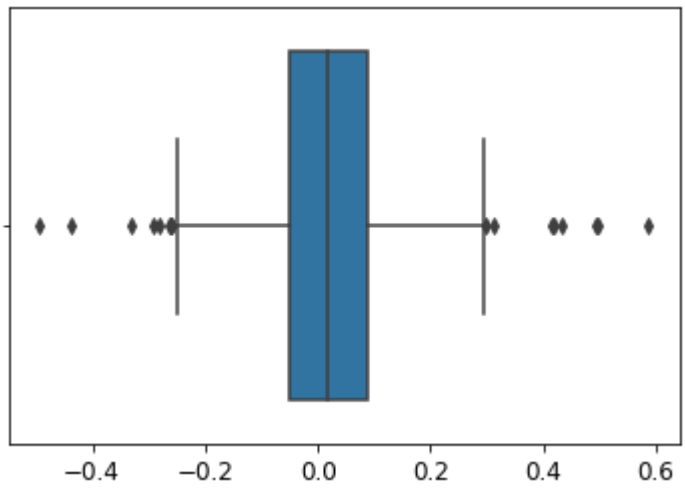
```
df_ara=df_BIST100_m[(df_BIST100_m["BIST100 Monthly Return in FX"].apply(lambd
```

In [496…

```
np_BIST100_m=df_ara[(df_ara["Month_ID"].apply(lambda x: int(x[:4])) >= 1990)]
```

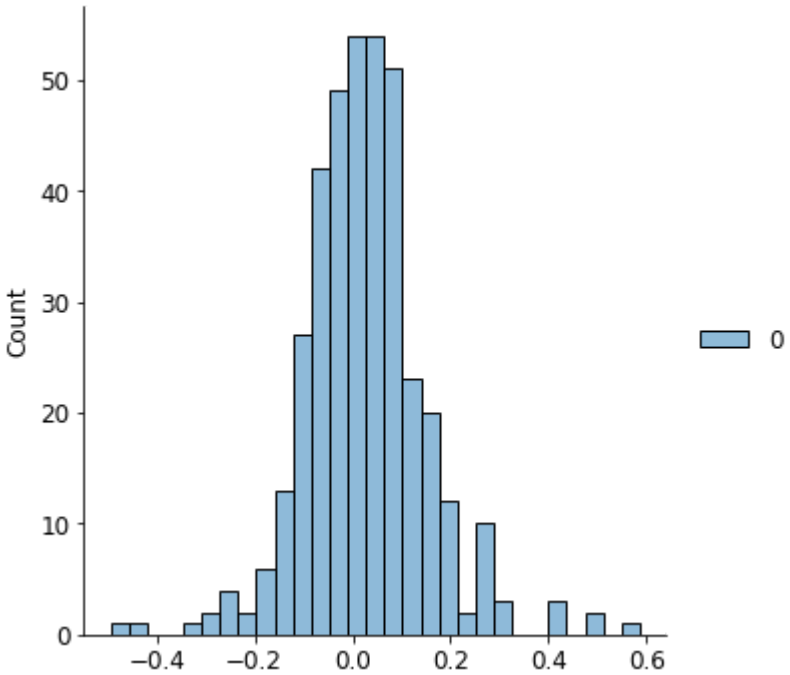
In [503…

```
ax=sb.boxplot(x=np_BIST100_m)
```



In [499…

```
bx=sb.displot(data=np_BIST100_m)
```



In [502…

```
statistic_m,pvalue_m=kstest(np_BIST100_m,'norm')
print('Staistic=%.3f, Pvalue= %.3f \n' %(statistic_m,pvalue_m))
if pvalue_m>0.05:
    print('Gausian with confidance interval 95%')
else:
    print('Not Gausian with confidance interval 95%')
```

```
Staistic=0.721, Pvalue= 0.000

Not Gausian with confidance interval 95%
```

In [501…

```
#Skewness
```

```
df_ara['BIST100 Monthly Return in FX'].skew()
```

Out[501… 0.39116095303093157

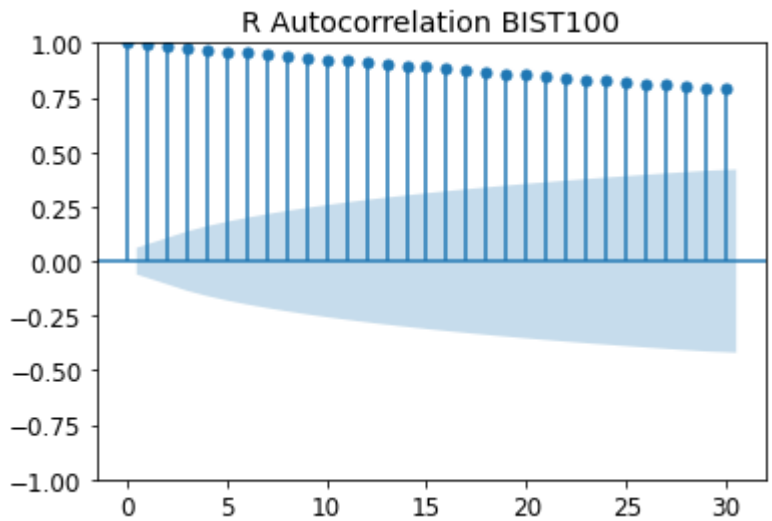
Autocorrelation Tests

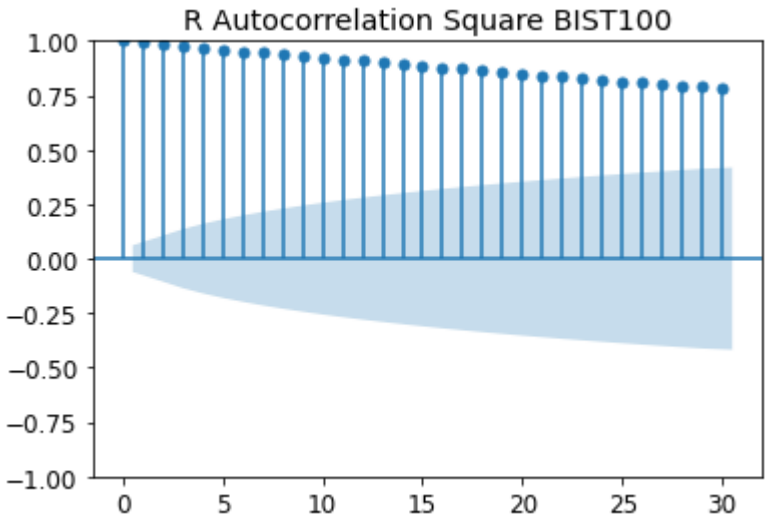
In [508…

```
def daily_ror(df):

    df_name=[x for x in globals() if globals()[x] is df][0]
    df_name.split("_",1)[1]

    daily_ror = ["None"]
    daily_ror_square = ["None"]

    for i in range(len(df) - 1):
        daily_ror.append(ln(df.iloc[i + 1]["Index_V"])/ df.iloc[i]["Index_V"]
        daily_ror_square.append((ln(df.iloc[i + 1]["Index_V"])/ df.iloc[i]["I

    df['Daily_ROR'] = daily_ror
    df['Daily_ROR_square'] = daily_ror_square

    # get the year

    year = []
    for i in range(len(df)):
        year.append(df["Date"][i].year)
    df["Year"]=year

    #get the subset of dataset with respect to year

    df_from=df[df["Year"]>=2018]

    plot_acf_ror = plot_acf(x=df_from["Daily_ROR"].values, lags=30,title="R A
    plot_acf_ror_square = plot_acf(x=df_from["Daily_ROR_square"].values, lags
```

In [506…

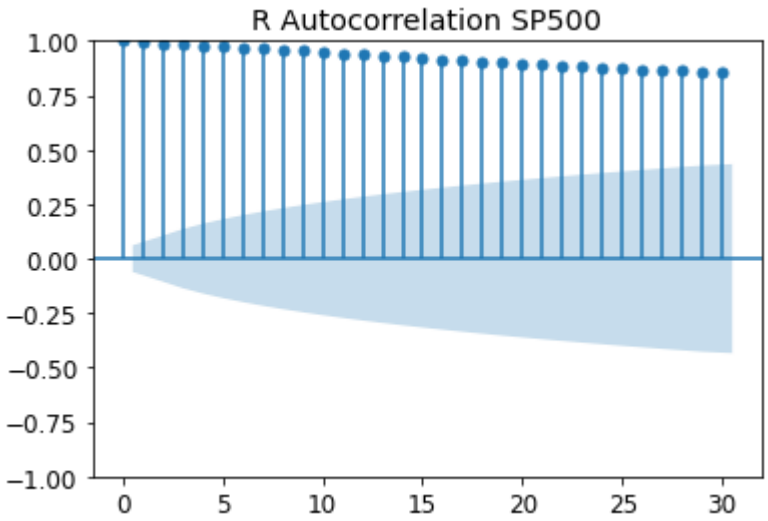
```
daily_ror(df_BIST100)
```

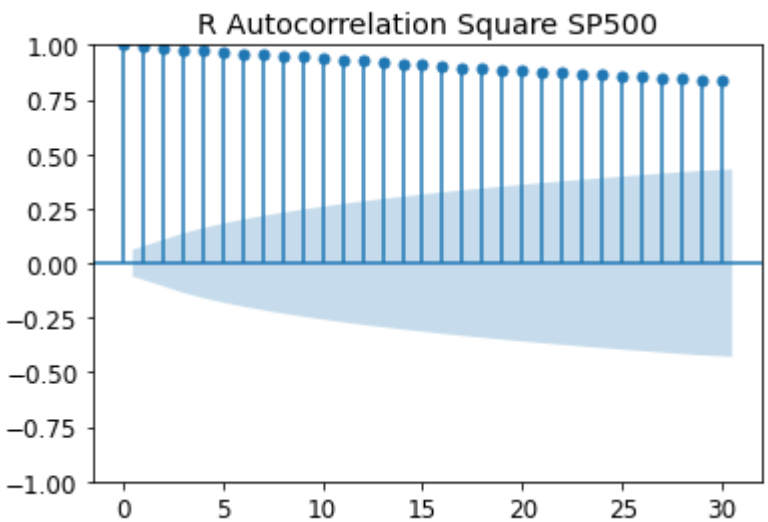




In [507…

```
daily_ror(df_SP500)
```





Normality Test

In [510…

```
def probabilities(df):

    daily_return = ["None"]

    for i in range(len(df) - 1):
        daily_return.append(((df.iloc[i + 1]["Index_V"])-df.iloc[i]["Index_V"
```

```
    df['Daily_Return'] = daily_return

    year = []
    for i in range(len(df)):
        year.append(df["Date"][i].year)
    df["Year"]=year

    df_from = df[df["Year"]>=2001]
    number_of_times_event_occurs = len(df_from[df_from["Daily_Return"]<=-0.08
    total_number_of_trials = len(df_from)

    print("Empirical probability is: " + str(number_of_times_event_occurs/tot

    # Normal probability

    print("Normal probability is: " + str(scipy.stats.norm(sum(df_from["Daily
```

In [511…
```
probabilities(df_BIST100)
```

```
Empirical probability is: 0.0018298261665141812
Normal probability is: 0.0005850425588465742
```

Value Calculations

Risk Premiums

In [290…
```
df_TRRF.head()
```

Out[290…

	Quarter	Deposit rate	Year_ID
0	Q4 1978	6.00	1978
1	Q1 1979	6.00	1979
2	Q2 1979	7.33	1979
3	Q3 1979	8.00	1979
4	Q4 1979	8.00	1979

In [288…
```
df_TRRF['Year_ID']=df_TRRF.loc[:, ('Quarter')].apply(lambda x: int(x[-4:]))
```

In [293…
```
df_TRRF_yearly=df_TRRF[(df_TRRF["Year_ID"] >=2001)][['Year_ID','Deposit rate'
```

In [294…
```
df_TRRF_yearly
```

Out[294…

	Deposit rate
Year_ID	
2001	74.7000
2002	50.4925
2003	37.6775
2004	24.2600

Year_ID	Deposit rate
2005	20.3975
2006	21.6500
2007	22.5600
2008	22.9125
2009	17.6475
2010	15.2700
2011	14.1050
2012	17.1925
2013	15.2950
2014	16.9325
2015	14.9175
2016	14.6100
2017	15.2875
2018	23.2850
2019	25.4075
2020	13.3575
2021	19.9900

In [310…

```
df_BIST100
```

Out[310…

	Date	Index_V	Market_cap (m TL)	TL/USD	is close day
0	1988-01-04	0.07	1	0.00102	False
1	1988-01-05	0.07	1	0.00102	False
2	1988-01-06	0.07	1	0.00102	False
3	1988-01-07	0.07	1	0.00102	False
4	1988-01-08	0.07	1	0.00102	False
…	…	…	…	…	…
8850	2021-12-06	1927.39	1561890	13.77700	False
8851	2021-12-07	1981.04	1601373	13.56825	False
8852	2021-12-08	2004.55	1616154	13.71250	False
8853	2021-12-09	2031.44	1635462	13.77625	False
8854	2021-12-10	2035.48	1632029	13.86000	False

8855 rows × 5 columns

In [333…

```
df_BIST100_y = pd.DataFrame(data= {'Year_ID': [], 'return': []})
for i, row in df_BIST100.iterrows():
    if i ==0 :
        lag=row[1]
```

```
    elif i != df_BIST100.shape[0]-1:
        lead=df_BIST100['Date'].iloc[i+1]
        if lead.year==row[0].year:
            0==0
        elif lead.year==row[0].year+1:
            df_BIST100_y.at[i,'Year_ID']= row[0].year
            df_BIST100_y.at[i,'return']= ln(row[1]/lag)
            lag= row[1]
df_BIST100_y.reset_index(inplace=True,drop=True)
```

In [329…

```
df_BIST100_y.head(5)
```

Out[329…

	year_ID	return
0	1988.0	-0.559616
1	1989.0	1.704748
2	1990.0	0.405465
3	1991.0	0.287682
4	1992.0	-0.095310

In [335…

```
df_BIST100_y=df_BIST100_y[(df_BIST100_y['Year_ID']>=2001)].merge(df_TRRF_year
```

In [336…

```
df_BIST100_y.head(5)
```

Out[336…

	Year_ID	return	Deposit rate
0	2001.0	0.378798	74.7000
1	2002.0	-0.284519	50.4925
2	2003.0	0.585588	37.6775
3	2004.0	0.293250	24.2600
4	2005.0	0.465559	20.3975

In [337…

```
df_BIST100_rp = pd.DataFrame(data= {'Year_ID': [], 'Risk_Premium': []})
for i, row in df_BIST100_y.iterrows():
            df_BIST100_rp.at[i,'Year_ID']= row[0]
            df_BIST100_rp.at[i,'Risk_Premium']= row[1]-(row[2]/100)
```

In [340…

```
df_BIST100_rp.head(10)
```

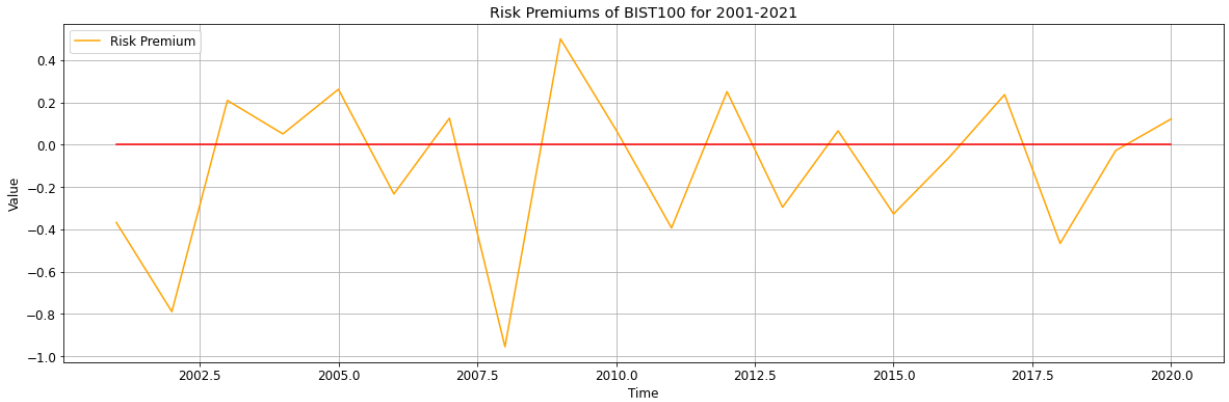
Out[340…

	Year_ID	Risk_Premium
0	2001.0	-0.368202
1	2002.0	-0.789444
2	2003.0	0.208813
3	2004.0	0.050650
4	2005.0	0.261584
5	2006.0	-0.233257

	Year_ID	Risk_Premium
6	2007.0	0.124910
7	2008.0	-0.955405
8	2009.0	0.499722
9	2010.0	0.070031

In [521…

```
plt.rc('font', size=12)
fig, ax = plt.subplots(figsize=(20, 6))

# Specify how our lines should look
ax.plot(df_BIST100_rp['Year_ID'],
        df_BIST100_rp['Risk_Premium'],
        color='orange', label='Risk Premium')
ax.plot(df_BIST100_rp['Year_ID'],
        np.zeros(df_BIST100_rp.shape[0]),
        color='red')


ax.set_xlabel('Time')
ax.set_ylabel('Value')
ax.set_title('Risk Premiums of BIST100 for 2001-2021')
ax.grid(True)
ax.legend(loc='upper left');
```



In [524…

```
print('Avarage Risk Premium of BIST100 for years 2001-2021 is: ',df_BIST100_r
```

```
Avarage Risk Premium of BIST100 for years 2001-2021 is:  -0.10146097674664487
```

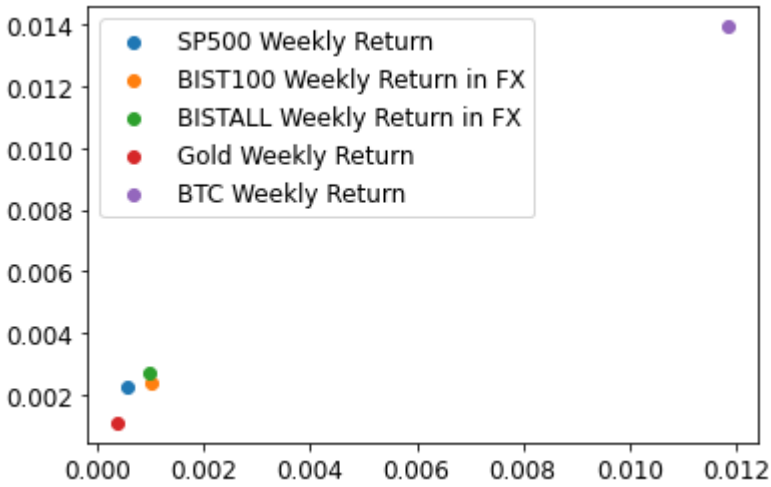
Portfolio Scenario

In [525…

```
y = df_master_w[(df_master_w["Year_ID"] >=2015)][['SP500 Weekly Return','BIST
           'BTC Weekly Return']].mean(axis = 0, skipna = False)
x = df_master_w[(df_master_w["Year_ID"] >=2015)][['SP500 Weekly Return','BIST
           'BTC Weekly Return']].var(axis = 0, skipna = False)

colors = ["red", "yellow", "blue","orange","black"]
c = ['SP500 Weekly Return','BIST100 Weekly Return in FX','BISTALL Weekly Retu

for i in range(0,5):
    plt.scatter(x[i], y[i], label=c[i])
plt.legend()
```

Out[525…

```
<matplotlib.legend.Legend at 0x11a1864d0>
```



In [529…
```
df_master_w[(df_master_w["Year_ID"] >=2016) & (df_master_w["Year_ID"] <=2020)
            'BTC Weekly Return']].mean(axis = 0, skipna = False)
```

Out[529…
```
SP500 Weekly Return             0.002245
BISTALL Weekly Return in FX     0.002925
Gold Weekly Return              0.002137
BTC Weekly Return               0.015902
dtype: float64
```

In [579…
```
t0=df_master_w[(df_master_w["Year_ID"] >=2021)][['SP500 Weekly Return','BISTA
            'BTC Weekly Return']].shape[0]
#t0=number of weeks in 2021
t1=df_master_w[(df_master_w["Year_ID"] >=2016) & (df_master_w["Year_ID"] <=20
            'BTC Weekly Return']].mean(axis = 0, skipna = False).mean()
#t2total mean return
print('By investing 1000$ equally to these 4 assets I expect to have ',(t0*t1
```

```
By investing 1000$ equally to these 4 assets I expect to have  1290.1052318286
754 $
```

In [570…
```
t2=250*int(df_master_w[(df_master_w["Year_ID"] >=2021)][['SP500 Index']].tail
```

In [571…
```
print('By investing 1000$ equally to these 4 assets I would  have ',t2, '$')
```

```
By investing 1000$ equally to these 4 assets I would  have  1271.706754514445
$
```