In [39]:

```python
# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly
plotly.offline.init_notebook_mode(connected=True)
from plotly.graph_objs import *
from plotly import tools
import plotly.express as px
import plotly.graph_objects as go
import seaborn as sns
```

In [40]:

```python
# Load datasets and parse date column
df_weather = pd.read_csv("./london_weather.csv", parse_dates = ['date'])
df_stock = pd.read_csv("./Index closing price from 1994 to 2021.csv", parse_dates = ['Date'])
```

In [41]:

```python
df_weather.head()
```

Out[41]:

	date	cloud_cover	sunshine	global_radiation	max_temp	mean_temp	min_temp	precipitation	pressure	snow_depth
0	1979-01-01	2.0	7.0	52.0	2.3	-4.1	-7.5	0.4	101900.0	9.0
1	1979-01-02	6.0	1.7	27.0	1.6	-2.6	-7.5	0.0	102530.0	8.0
2	1979-01-03	5.0	0.0	13.0	1.3	-2.8	-7.2	0.0	102050.0	4.0
3	1979-01-04	8.0	0.0	13.0	-0.3	-2.6	-6.5	0.0	100840.0	2.0
4	1979-01-05	6.0	2.0	29.0	5.6	-0.8	-1.4	0.0	102250.0	1.0

In [42]:

```python
df_stock.head()
```

Out[42]:

	Date	spx	dax	ftse	nikkei
0	1994-01-07	469.899994	2224.949951	3446.000000	18124.009766
1	1994-01-10	475.269989	2225.000000	3440.600098	18443.439453
2	1994-01-11	474.130005	2228.100098	3413.800049	18485.250000
3	1994-01-12	474.170013	2182.060059	3372.000000	18793.880859
4	1994-01-13	472.470001	2142.370117	3360.000000	18577.259766

In [43]:

```python
# Check number of NA values for all Columns
df_weather.isna().sum()
```

Out[43]:

```
date                   0
cloud_cover           19
sunshine               0
global_radiation      19
max_temp               6
mean_temp             36
min_temp               2
precipitation          6
pressure               4
snow_depth          1441
dtype: int64
```

In [44]:

```python
# Check number of NA values for all columns
df_stock.isna().sum()
```

Out[44]:

```
Date      0
spx       0
dax       0
ftse      0
nikkei    0
dtype: int64
```

In [45]:

```python
# Check datatypes of all the columns
df_weather.dtypes
```

Out[45]:

```
date                datetime64[ns]
cloud_cover                float64
sunshine                   float64
global_radiation           float64
max_temp                   float64
mean_temp                  float64
min_temp                   float64
precipitation              float64
pressure                   float64
snow_depth                 float64
dtype: object
```

In [46]:

```python
# Check datatypes of all the columns
df_stock.dtypes
```

Out[46]:

```
Date      datetime64[ns]
spx              float64
dax              float64
ftse             float64
nikkei           float64
dtype: object
```

In [47]:

```python
len(df_weather)
```

Out[47]:

```
15341
```

In [48]:

```python
len(df_stock)
```

Out[48]:

```
7255
```

In [49]:

```python
df_stock.shape
```

Out[49]:

```
(7255, 5)
```

In [50]:

```python
df_weather.shape
```

Out[50]:

```
(15341, 10)
```

In [51]:

```python
# Calculating mean
df_weather[['cloud_cover', 'sunshine', 'global_radiation', 'max_temp', 'mean_temp', 'min_temp', 'precipitation',
'pressure', 'snow_depth']].mean()
```

Out[51]:

```
cloud_cover              5.268242
sunshine                 4.350238
global_radiation       118.756951
max_temp                15.388777
mean_temp               11.475511
min_temp                 7.559867
precipitation            1.668634
pressure            101536.605594
snow_depth               0.037986
dtype: float64
```

In [52]:

```python
# Calculating standard deviation
df_weather[['cloud_cover', 'sunshine', 'global_radiation', 'max_temp', 'mean_temp', 'min_temp', 'precipitation',
'pressure', 'snow_depth']].std()
```

Out[52]:

```
cloud_cover            2.070072
sunshine               4.028339
global_radiation      88.898272
max_temp               6.554754
mean_temp              5.729709
min_temp               5.326756
precipitation          3.738540
pressure            1049.722604
snow_depth             0.545633
dtype: float64
```

In [53]:

```python
# Calculating median
df_weather[['cloud_cover', 'sunshine', 'global_radiation', 'max_temp', 'mean_temp', 'min_temp', 'precipitation',
'pressure', 'snow_depth']].median()
```

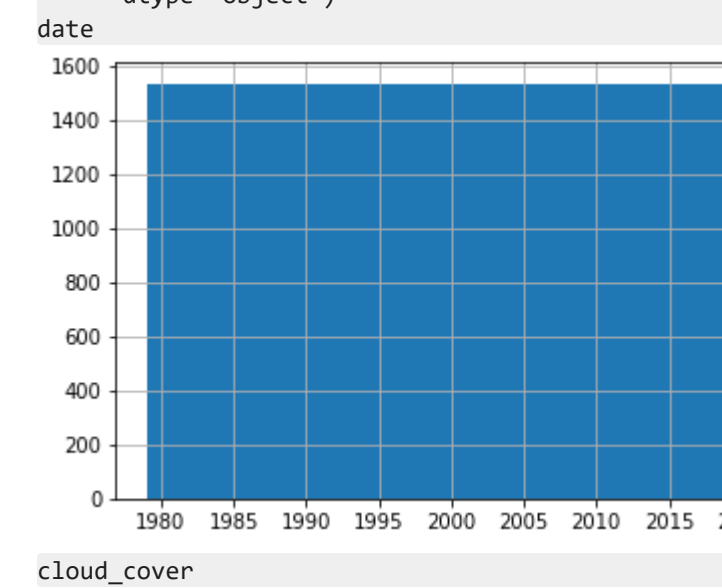
Out[53]:

```
cloud_cover              6.0
sunshine                 3.5
global_radiation        95.0
max_temp                15.0
mean_temp               11.4
min_temp                 7.8
precipitation            0.0
pressure            101620.0
snow_depth               0.0
dtype: float64
```

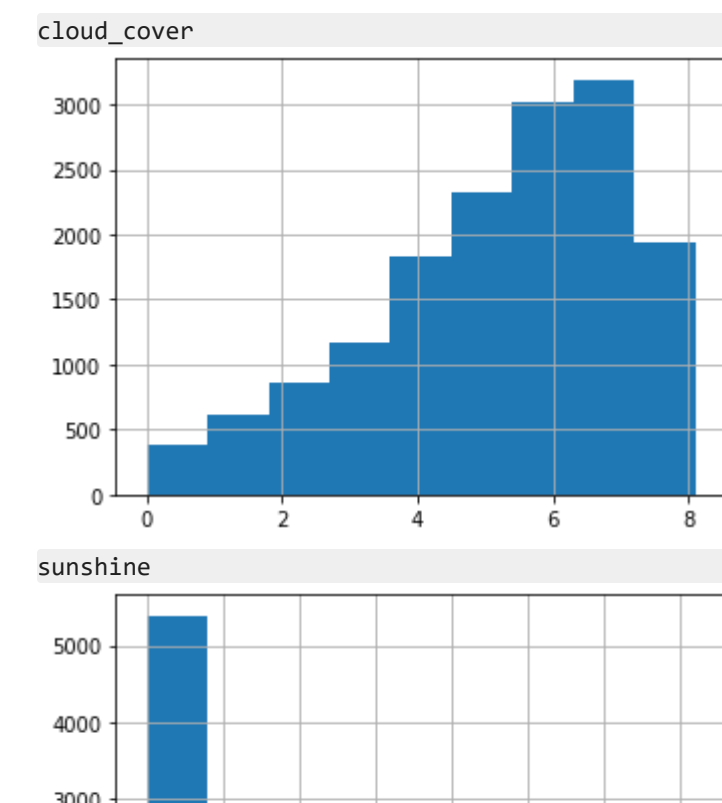
In [54]:

```python
# Load Histograms
cols = df_weather.columns
print(cols)
for col in cols:
    df_weather[col].hist()
    print(col)
    plt.show()
```

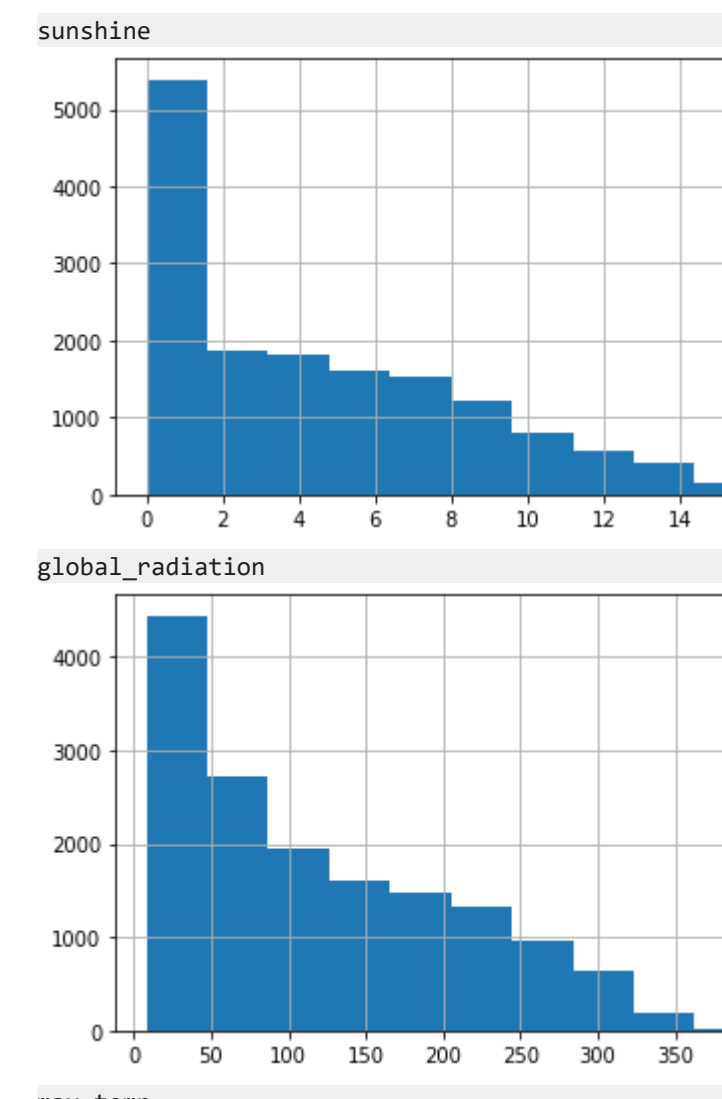
```
Index(['date', 'cloud_cover', 'sunshine', 'global_radiation', 'max_temp',
       'mean_temp', 'min_temp', 'precipitation', 'pressure', 'snow_depth'],
      dtype='object')
date
```



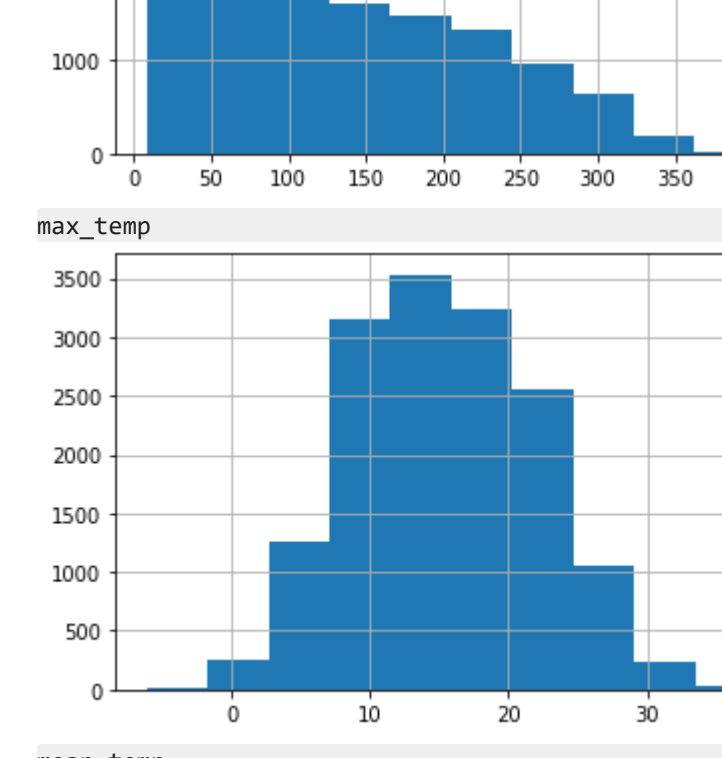
cloud_cover



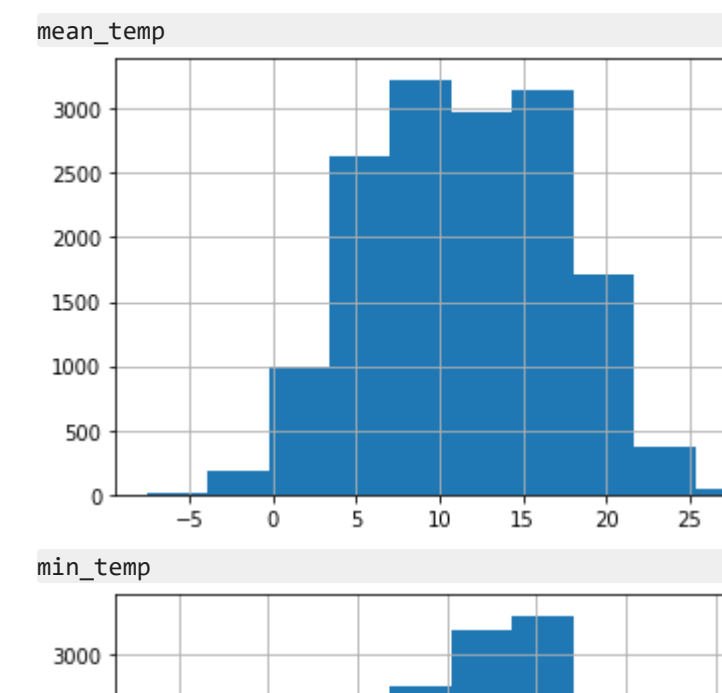
sunshine



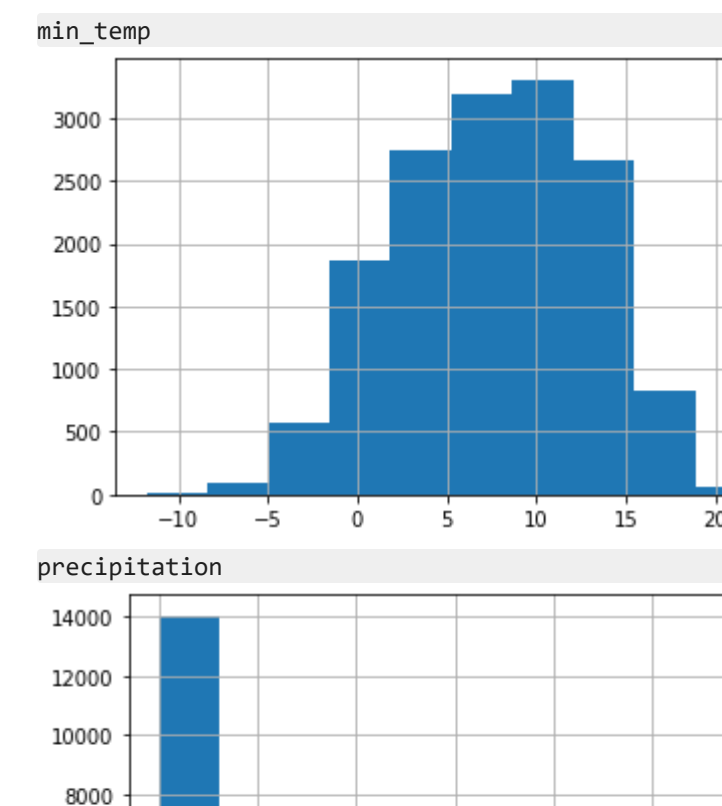
global_radiation



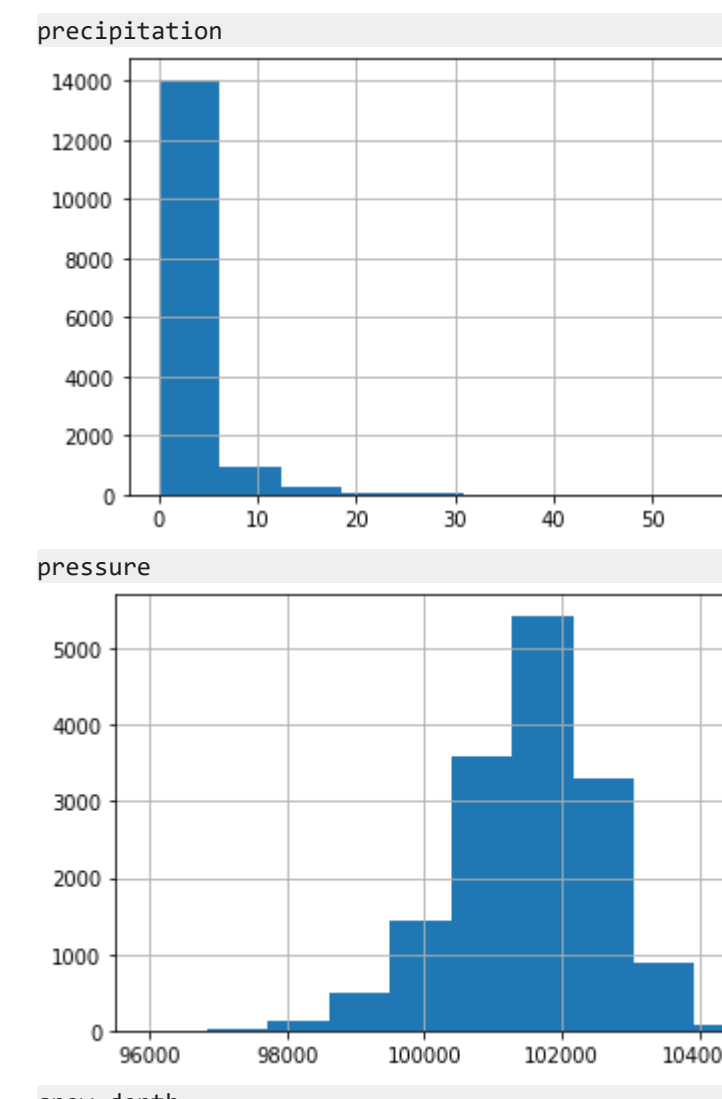
max_temp



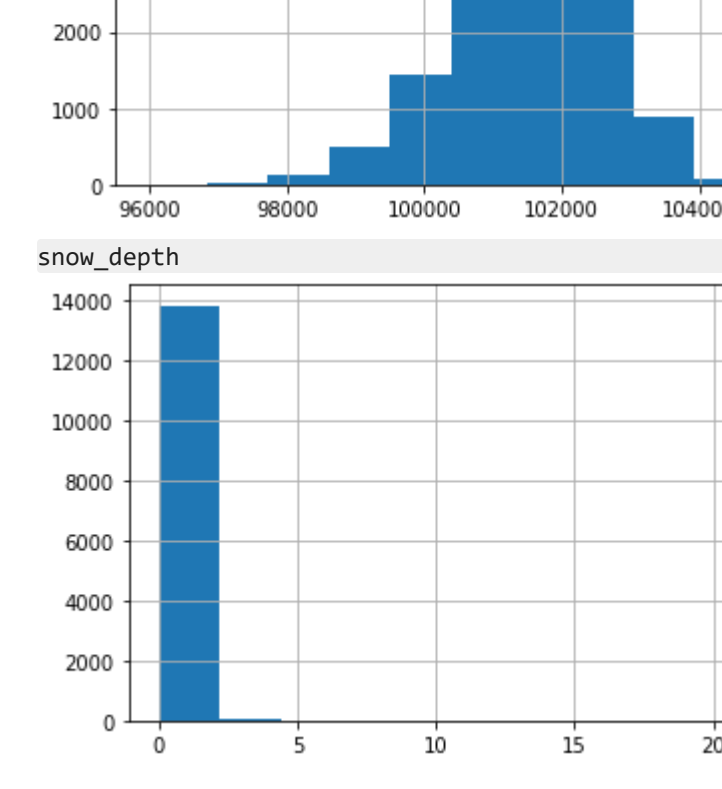
mean_temp



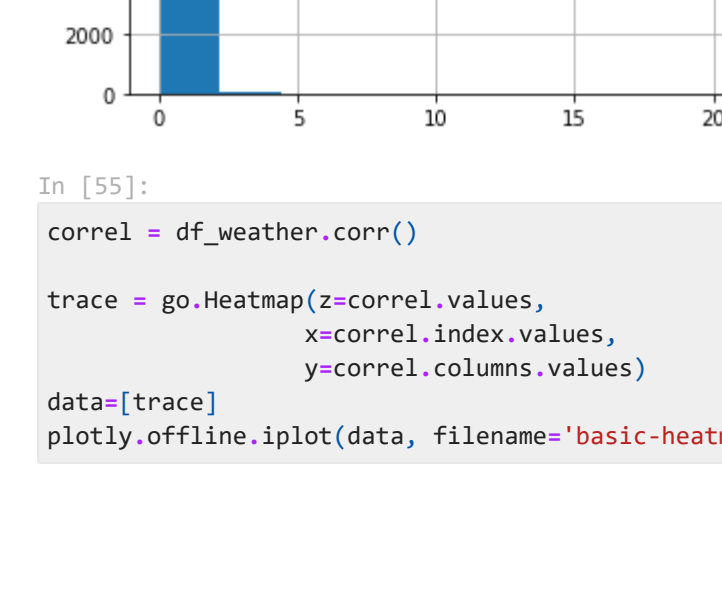
min_temp



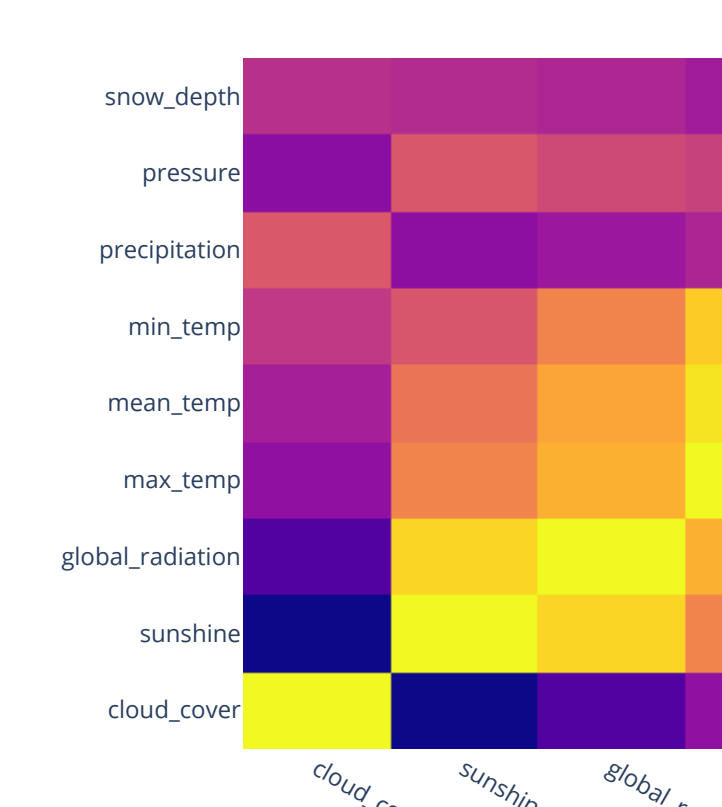
precipitation



pressure

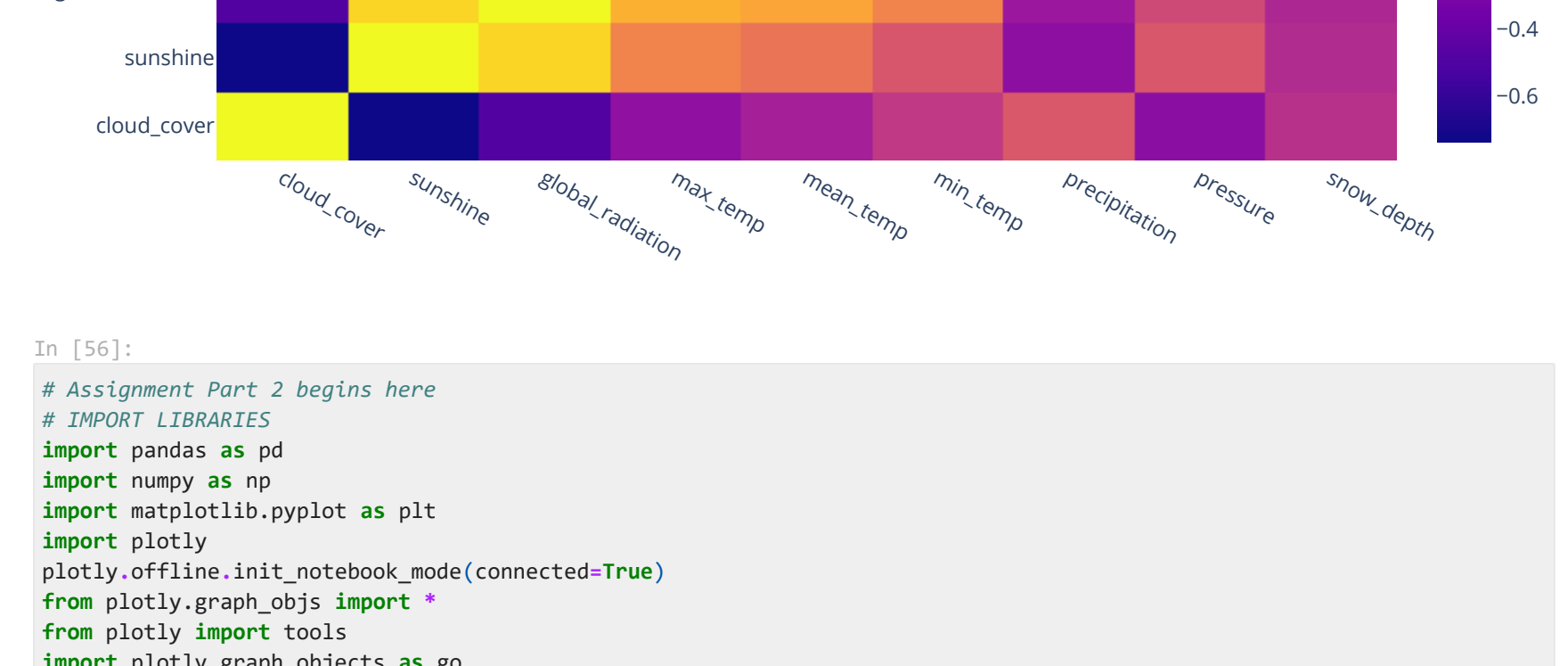


snow_depth



In [55]:

```python
correl = df_weather.corr()

trace = go.Heatmap(z=correl.values,
                   x=correl.index.values,
                   y=correl.columns.values)
data=[trace]
plotly.offline.iplot(data, filename='basic-heatmap')
```



In [56]:

```python
# Assignment Part 2 begins here
# IMPORT LIBRARIES
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly
plotly.offline.init_notebook_mode(connected=True)
from plotly.graph_objs import *
from plotly import tools
import plotly.graph_objects as go
import seaborn as sns
import plotly.express as px
import pandas as pd
```

In [57]:

```python
# QUESTION 1

# Load datasets and parse date column
df_weather = pd.read_csv("./london_weather.csv", parse_dates = ['date'])
df_stock = pd.read_csv("./Index closing price from 1994 to 2021.csv", parse_dates = ['Date'])
```

In [58]:

```python
df_weather.head()
```

Out[58]:

	date	cloud_cover	sunshine	global_radiation	max_temp	mean_temp	min_temp	precipitation	pressure	snow_depth
0	1979-01-01	2.0	7.0	52.0	2.3	-4.1	-7.5	0.4	101900.0	9.0
1	1979-01-02	6.0	1.7	27.0	1.6	-2.6	-7.5	0.0	102530.0	8.0
2	1979-01-03	5.0	0.0	13.0	1.3	-2.8	-7.2	0.0	102050.0	4.0
3	1979-01-04	8.0	0.0	13.0	-0.3	-2.6	-6.5	0.0	100840.0	2.0
4	1979-01-05	6.0	2.0	29.0	5.6	-0.8	-1.4	0.0	102250.0	1.0

In [59]:

```python
df_stock.head()
```

Out[59]:

	Date	spx	dax	ftse	nikkei
0	1994-01-07	469.899994	2224.949951	3446.000000	18124.009766
1	1994-01-10	475.269989	2225.000000	3440.600098	18443.439453
2	1994-01-11	474.130005	2228.100098	3413.800049	18485.250000
3	1994-01-12	474.170013	2182.060059	3372.000000	18793.880859
4	1994-01-13	472.470001	2142.370117	3360.000000	18577.259766

In [60]:

```python
# Used plotly express to create a hoverable line graph plotting the date vs ftse index value over time.
fig = px.line(df_stock, x='Date', y='ftse', labels={'ftse': 'FTSE Value'}, title='FTSE Index Over Time')

# Updating the properties to show hover content.
fig.update_layout(hovermode='x unified', hoverlabel=dict(bgcolor="white", font_size=12))


fig.show()
```

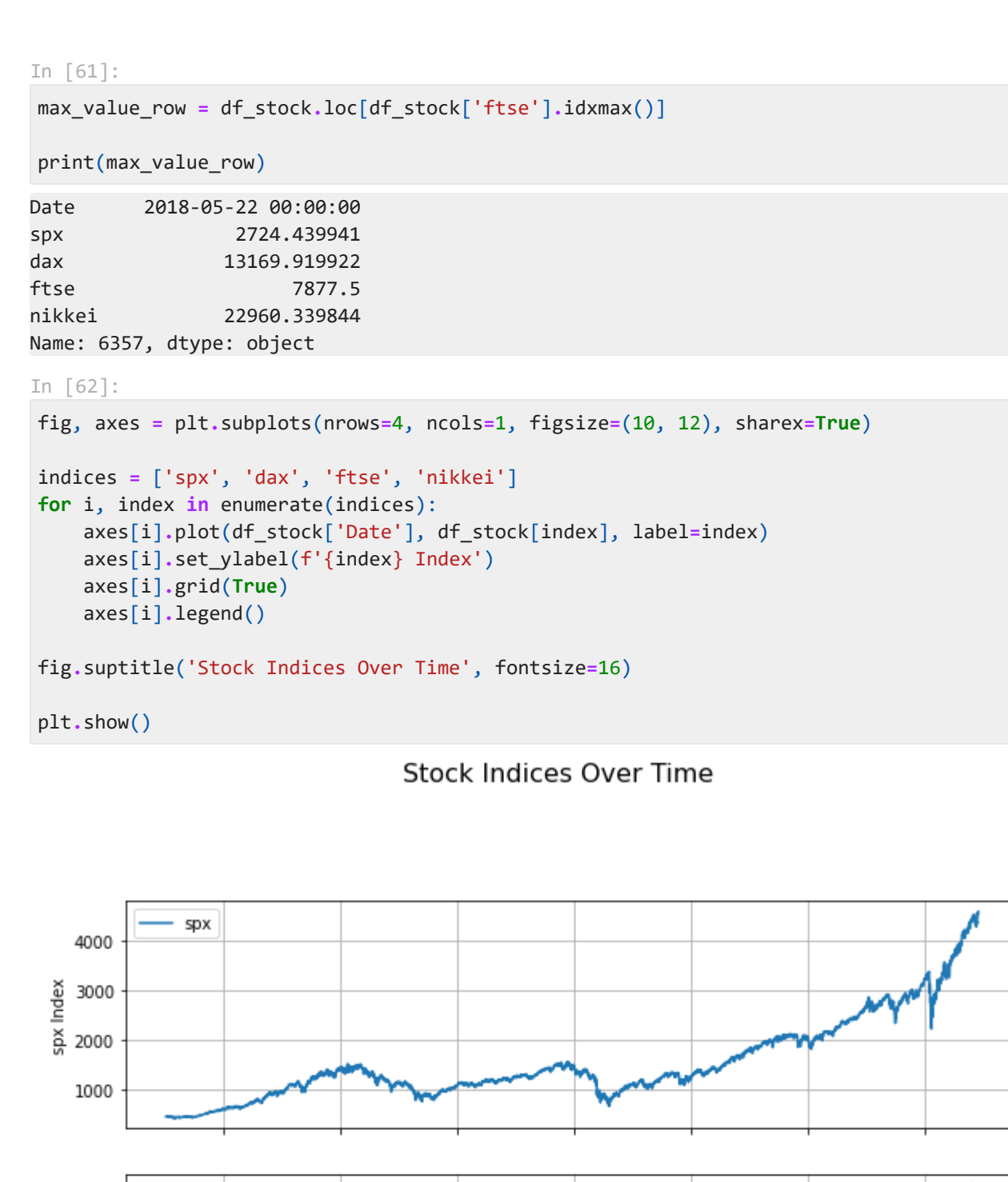
In [61]:

```python
max_value_row = df_stock.loc[df_stock['ftse'].idxmax()]

print(max_value_row)
```

```
Date      2018-05-22 00:00:00
spx               2724.439941
dax              13169.919922
ftse                   7877.5
nikkei           22960.339844
Name: 6357, dtype: object
```

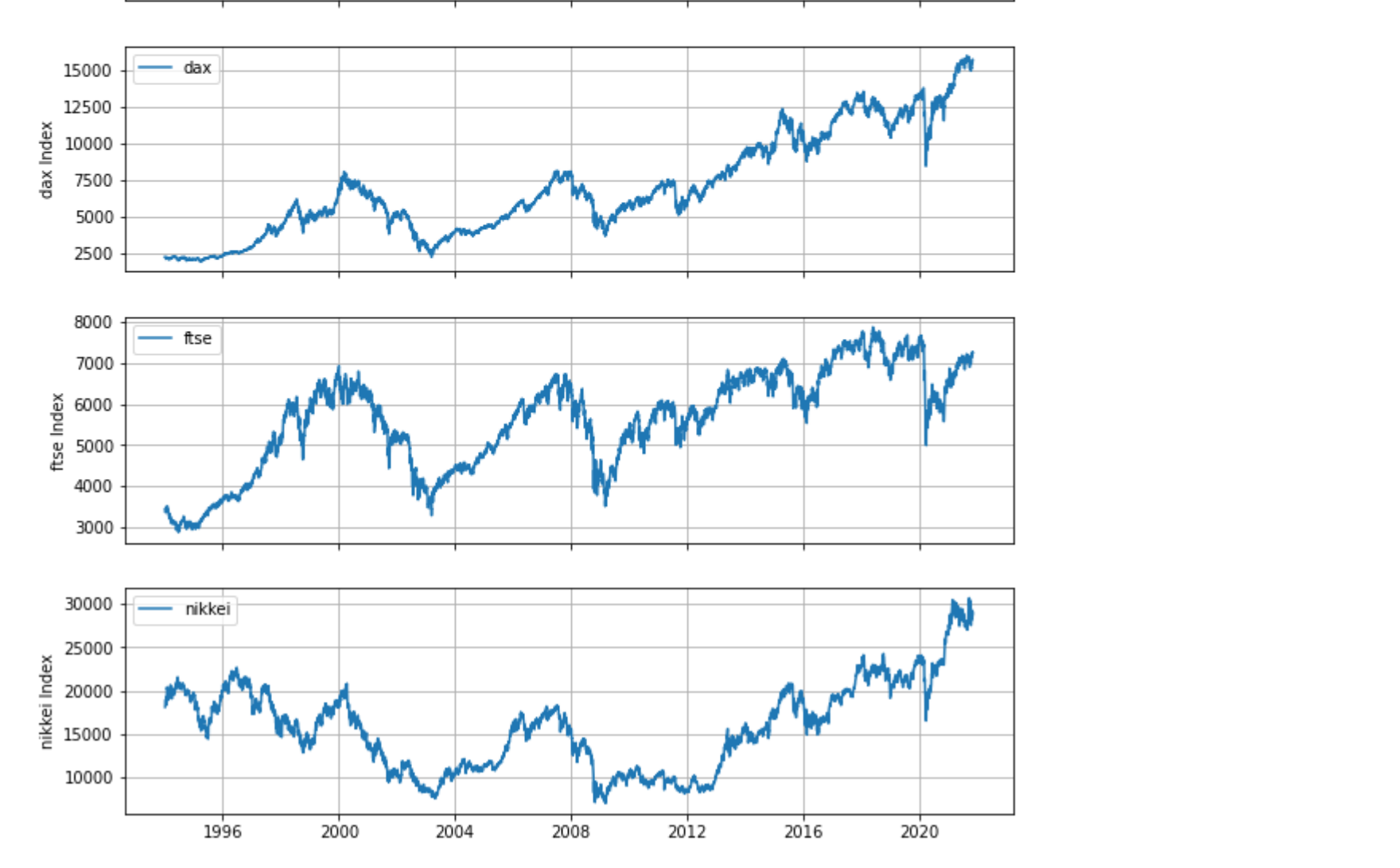
In [62]:

```python
fig, axes = plt.subplots(nrows=4, ncols=1, figsize=(10, 12), sharex=True)

indices = ['spx', 'dax', 'ftse', 'nikkei']
for i, index in enumerate(indices):
    axes[i].plot(df_stock['Date'], df_stock[index], label=index)
    axes[i].set_ylabel(f'{index} Index')
    axes[i].grid(True)
    axes[i].legend()

fig.suptitle('Stock Indices Over Time', fontsize=16)

plt.show()
```



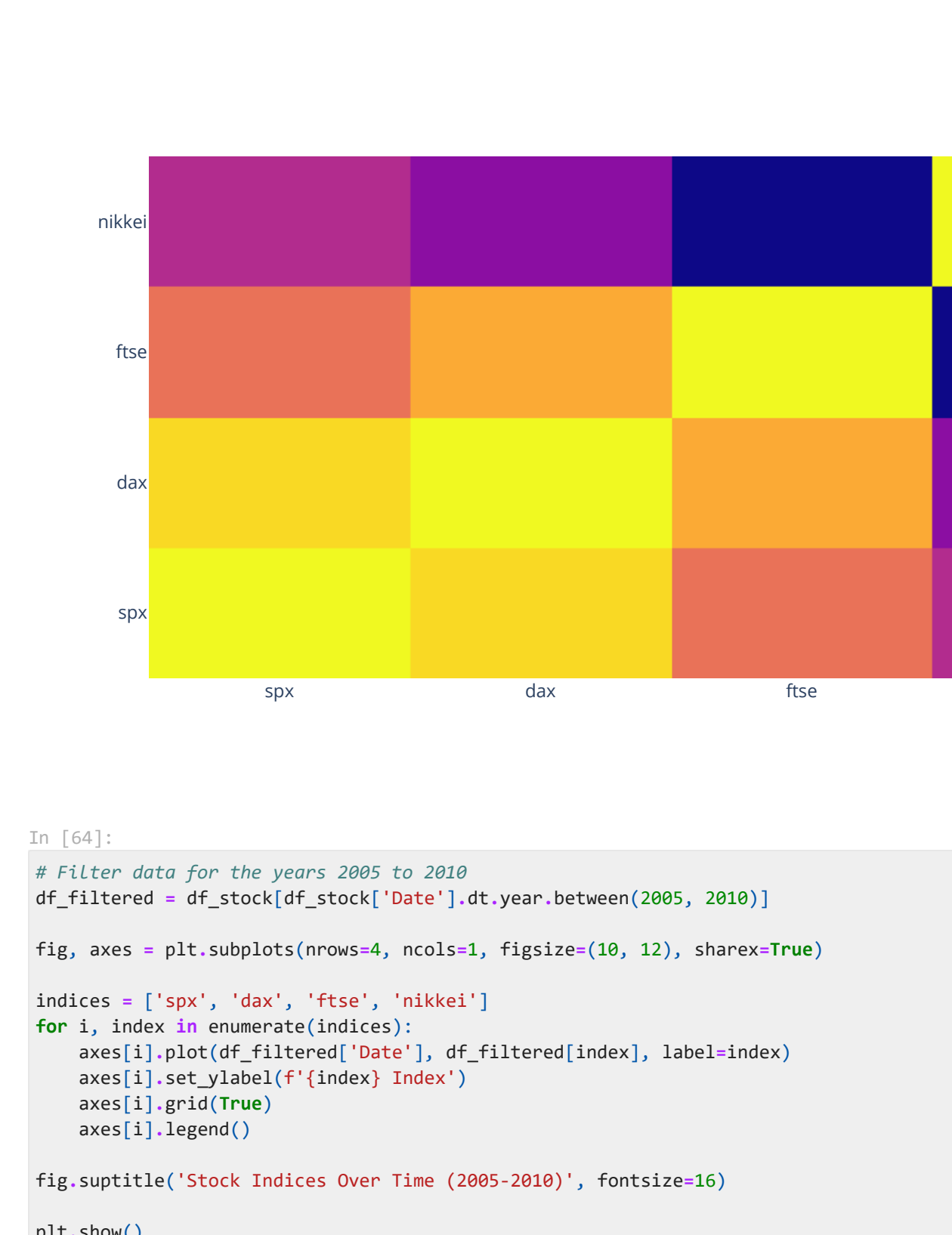
In [63]:

```python
# PLOT HEATMAP OF CORRELATION BETWEEN ALL PAIRS OF indices.
correl = df_stock.corr()

trace = go.Heatmap(z=correl.values,
                   x=correl.index.values,
                   y=correl.columns.values)
data=[trace]
plotly.offline.iplot(data, filename='basic-heatmap')
```



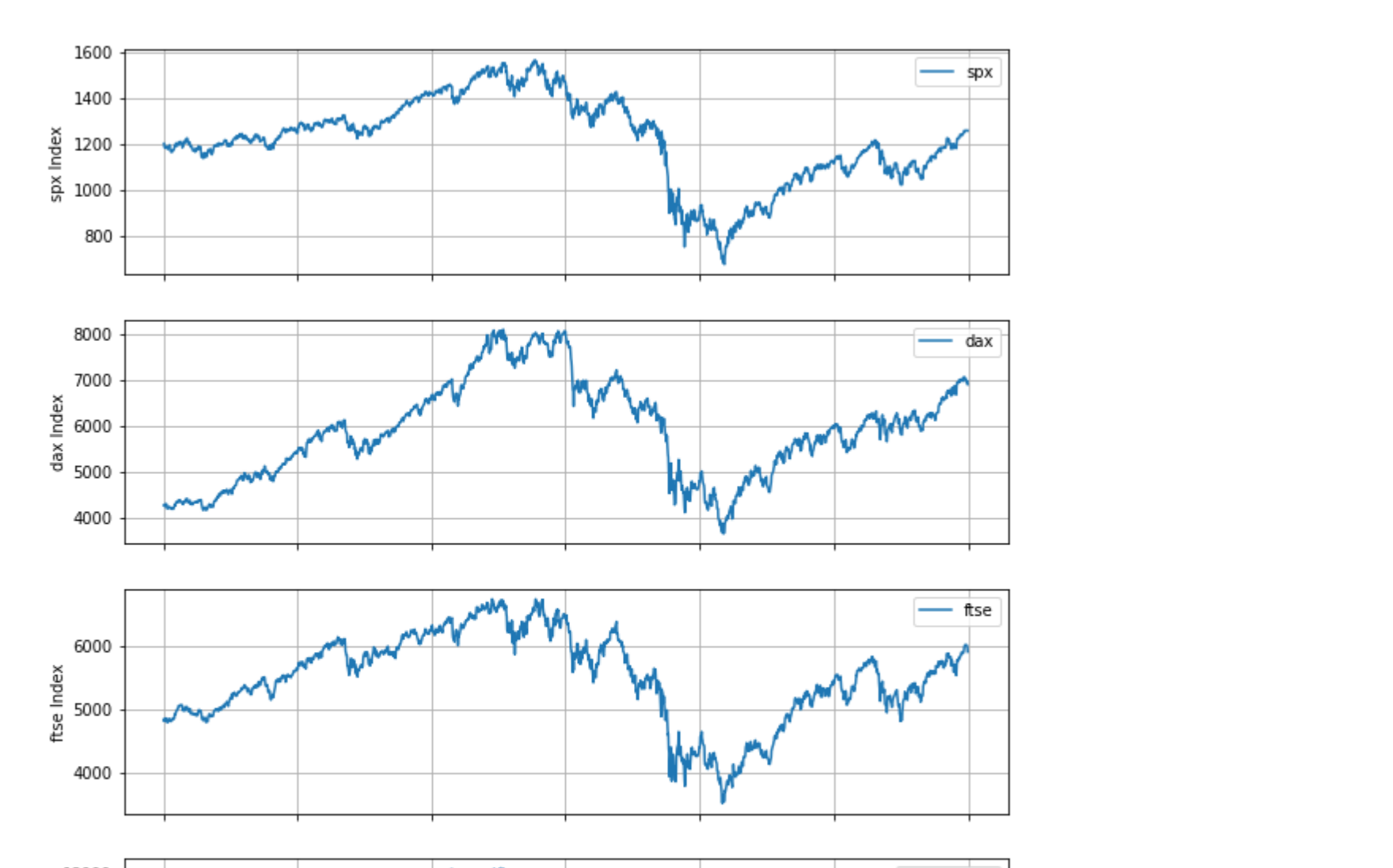
In [64]:

```python
# Filter data for the years 2005 to 2010
df_filtered = df_stock[df_stock['Date'].dt.year.between(2005, 2010)]

fig, axes = plt.subplots(nrows=4, ncols=1, figsize=(10, 12), sharex=True)

indices = ['spx', 'dax', 'ftse', 'nikkei']
for i, index in enumerate(indices):
    axes[i].plot(df_filtered['Date'], df_filtered[index], label=index)
    axes[i].set_ylabel(f'{index} Index')
    axes[i].grid(True)
    axes[i].legend()

fig.suptitle('Stock Indices Over Time (2005-2010)', fontsize=16)

plt.show()
```



In [65]:

```python
stock_dataset_filtered = df_stock[df_stock['Date'].dt.year.between(2005, 2010)]

correl_filtered = stock_dataset_filtered.corr()

trace = go.Heatmap(z=correl_filtered.values,
                   x=correl_filtered.index.values,
                   y=correl_filtered.columns.values)

layout = go.Layout(title='Correlation Matrix (2005-2010)',
                   xaxis=dict(title='Stock Indices'),
                   yaxis=dict(title='Stock Indices'))

fig = go.Figure(data=[trace], layout=layout)

fig.show()
```



In [66]:

```python
# QUESTION 2

weather_data_filtered = df_weather[df_weather['date'].dt.year.between(2014, 2018)]
weather_data_filtered.head()
```

Out[66]:

	date	cloud_cover	sunshine	global_radiation	max_temp	mean_temp	min_temp	precipitation	pressure	snow_depth
12784	2014-01-01	6.0	0.0	13.0	11.1	8.5	5.6	13.8	99300.0	0.0
12785	2014-01-02	4.0	5.2	45.0	10.7	8.6	6.1	4.8	99330.0	0.0
12786	2014-01-03	3.0	2.9	34.0	9.5	8.6	6.5	12.0	99440.0	0.0
12787	2014-01-04	6.0	0.0	13.0	12.5	7.4	5.3	3.4	99360.0	0.0
12788	2014-01-05	4.0	1.0	23.0	12.6	6.6	0.7	9.6	100010.0	0.0

In [67]:

```python
weather_data_filtered.isna().sum()
```

Out[67]:

```
date                0
cloud_cover         0
sunshine            0
global_radiation    6
max_temp            0
mean_temp           0
min_temp            0
precipitation       0
pressure            0
snow_depth          2
dtype: int64
```

In [68]:

```python
stock_data_filtered = df_stock[df_stock['Date'].dt.year.between(2014, 2018)]
stock_data_filtered.head()
```

Out[68]:

	Date	spx	dax	ftse	nikkei
5213	2014-01-01	1848.359985	9552.160156	6731.299805	16291.309570
5214	2014-01-02	1831.979980	9400.040039	6717.899902	16291.309570
5215	2014-01-03	1831.369995	9435.150391	6730.700195	16291.309570
5216	2014-01-06	1826.770020	9428.000000	6730.700195	15908.879883
5217	2014-01-07	1837.880005	9506.200195	6755.500000	15814.370117

In [69]:

```python
stock_data_filtered.isna().sum()
```

Out[69]:

```
Date      0
spx       0
dax       0
ftse      0
nikkei    0
dtype: int64
```

In [70]:

```python
#Interpolating the columns with NaN values
weather_data_filtered['global_radiation'].interpolate(inplace = True)
weather_data_filtered['snow_depth'].interpolate(inplace = True)
```

```
<ipython-input-70-3d97e0ce366b>:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy

<ipython-input-70-3d97e0ce366b>:3: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy
```

In [71]:

```python
weather_data_filtered.isna().sum()
```

Out[71]:

```
date                0
cloud_cover         0
sunshine            0
global_radiation    0
max_temp            0
mean_temp           0
min_temp            0
precipitation       0
pressure            0
snow_depth          0
dtype: int64
```

In [72]:

```python
stock_data_filtered.isna().sum()
```

Out[72]:

```
Date      0
spx       0
dax       0
ftse      0
nikkei    0
dtype: int64
```

In [73]:

```python
len(weather_data_filtered)
```

Out[73]:

```
1826
```

In [74]:

```python
len(stock_data_filtered)
```

Out[74]:

```
1304
```

In [75]:

```python
stock_subset = stock_data_filtered[['Date', 'ftse']]

# Merge the columns with the London weather data using the Date field, used left_on and right_on as
# the two date columns have different starting letters.
merged_data = pd.merge(weather_data_filtered,stock_subset, left_on='date', right_on='Date', how='left')
```

In [76]:

```python
merged_data.head()
```

Out[76]:

	date	cloud_cover	sunshine	global_radiation	max_temp	mean_temp	min_temp	precipitation	pressure	snow_depth	Date
0	2014-01-01	6.0	0.0	13.0	11.1	8.5	5.6	13.8	99300.0	0.0	2014-01-01
1	2014-01-02	4.0	5.2	45.0	10.7	8.6	6.1	4.8	99330.0	0.0	2014-01-02
2	2014-01-03	3.0	2.9	34.0	9.5	8.6	6.5	12.0	99440.0	0.0	2014-01-03
3	2014-01-04	6.0	0.0	13.0	12.5	7.4	5.3	3.4	99360.0	0.0	NaT
4	2014-01-05	4.0	1.0	23.0	12.6	6.6	0.7	9.6	100010.0	0.0	NaT

In [77]:

```python
len(merged_data)
```

Out[77]:

```
1826
```

In [78]:

```python
merged_data.isna().sum()
```

Out[78]:

```
date                  0
cloud_cover           0
sunshine              0
global_radiation      0
max_temp              0
mean_temp             0
min_temp              0
precipitation         0
pressure              0
snow_depth            0
Date                522
ftse                522
dtype: int64
```

In [79]:

```python
merged_data['ftse'].interpolate(inplace=True)
```

In [80]:

```python
merged_data.drop(columns=['Date'], inplace=True)
```

In [81]:

```python
merged_data.isna().sum()
```

Out[81]:

```
date                0
cloud_cover         0
sunshine            0
global_radiation    0
max_temp            0
mean_temp           0
min_temp            0
precipitation       0
pressure            0
snow_depth          0
ftse                0
dtype: int64
```

In [82]:

```python
# filtering data where snow depth is greater than 0
subset_data = merged_data[merged_data['snow_depth'] > 0]
subset_data.head()
```

Out[82]:

	date	cloud_cover	sunshine	global_radiation	max_temp	mean_temp	min_temp	precipitation	pressure	snow_depth	
1439	2017-12-10	8.0	0.0	12.0	3.5	1.3	0.0	11.2	98250.0	1.0	74
1519	2018-02-28	6.0	5.3	94.0	0.3	-3.3	-5.4	2.0	101630.0	1.0	72
1537	2018-03-18	8.0	0.0	45.0	5.4	-0.3	-1.4	1.0	100860.0	2.0	70

In [83]:

```python
correl = subset_data.corr()

trace = go.Heatmap(z=correl.values,
                   x=correl.index.values,
                   y=correl.columns.values)
data=[trace]
plotly.offline.iplot(data, filename='basic-heatmap')
```