In [77]:

```python
import pandas as pd

# Read the CSV file
stock_df = pd.read_csv("Index closing price from 1994 to 2021.csv", parse_dates
weather_df = pd.read_csv("london_weather.csv", parse_dates = ['date'])

# View the first 5 rows
stock_df.head()
```

Out[77]:

	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 [78]:

```python
stock_df
```

Out[78]:

	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
...	...	...	...	...	...
7250	2021-10-22	4544.899902	15542.980469	7204.600098	28804.849609
7251	2021-10-25	4566.479980	15599.230469	7222.799805	28600.410156
7252	2021-10-26	4574.790039	15757.059570	7277.600098	29106.009766
7253	2021-10-27	4551.680176	15705.809570	7253.299805	29098.240234
7254	2021-10-28	4596.419922	15696.330078	7249.500000	28820.089844

7255 rows × 5 columns

In [79]:

```python
weather_df
```

Out[79]:

	date	cloud_cover	sunshine	global_radiation	max_temp	mean_temp	min_temp	pre
0	1979-01-01	2.0	7.0	52.0	2.3	-4.1	-7.5	
1	1979-01-02	6.0	1.7	27.0	1.6	-2.6	-7.5	
2	1979-01-03	5.0	0.0	13.0	1.3	-2.8	-7.2	
3	1979-01-04	8.0	0.0	13.0	-0.3	-2.6	-6.5	
4	1979-01-05	6.0	2.0	29.0	5.6	-0.8	-1.4	
...	...	...	...	...	...	...	...	
15336	2020-12-27	1.0	0.9	32.0	7.5	7.5	7.6	
15337	2020-12-28	7.0	3.7	38.0	3.6	1.1	-1.3	
15338	2020-12-29	7.0	0.0	21.0	4.1	2.6	1.1	
15339	2020-12-30	6.0	0.4	22.0	5.6	2.7	-0.1	
15340	2020-12-31	7.0	1.3	34.0	1.5	-0.8	-3.1	

15341 rows × 10 columns

In [80]:
```
weather_df.isna().sum()
```

Out[80]:
```
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 [67]:
```
stock_df.isna().sum()
```

Out[67]:
```
Date      0
spx       0
dax       0
ftse      0
nikkei    0
dtype: int64
```

In [68]:
```
weather_df.dtypes
```

Out[68]:
```
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 [69]:
```
stock_df.dtypes
```

Out[69]:
```
Date      datetime64[ns]
spx              float64
dax              float64
ftse             float64
nikkei           float64
dtype: object
```

In [70]:
```
weather_df.dropna(subset='cloud_cover',inplace=True)
```

In [71]:
```
weather_df[['cloud_cover', 'sunshine', 'global_radiation', 'max_temp', 'mean_te
```

Out[71]:
```
cloud_cover              5.268242
sunshine                 4.354418
global_radiation       118.861073
max_temp                15.397473
mean_temp               11.484515
min_temp                 7.568277
precipitation            1.669235
pressure            101535.718109
snow_depth               0.037747
dtype: float64
```

In [72]:
```
weather_df[['cloud_cover', 'sunshine', 'global_radiation', 'max_temp', 'mean_te
```

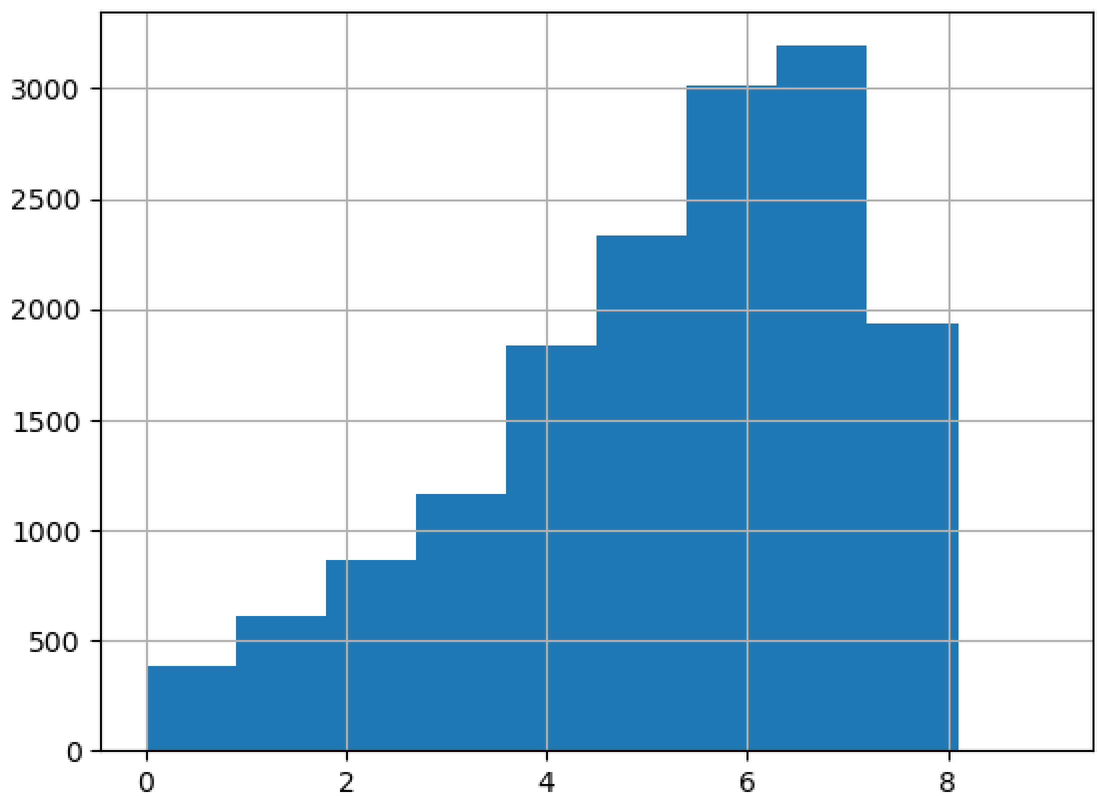
Out[72]:
```
cloud_cover            2.070072
sunshine               4.028619
global_radiation      88.895010
max_temp               6.551577
mean_temp              5.725319
min_temp               5.322984
precipitation          3.739198
pressure            1049.113961
snow_depth             0.544948
dtype: float64
```

In [73]:
```
weather_df[['cloud_cover', 'sunshine', 'global_radiation', 'max_temp', 'mean_te
```

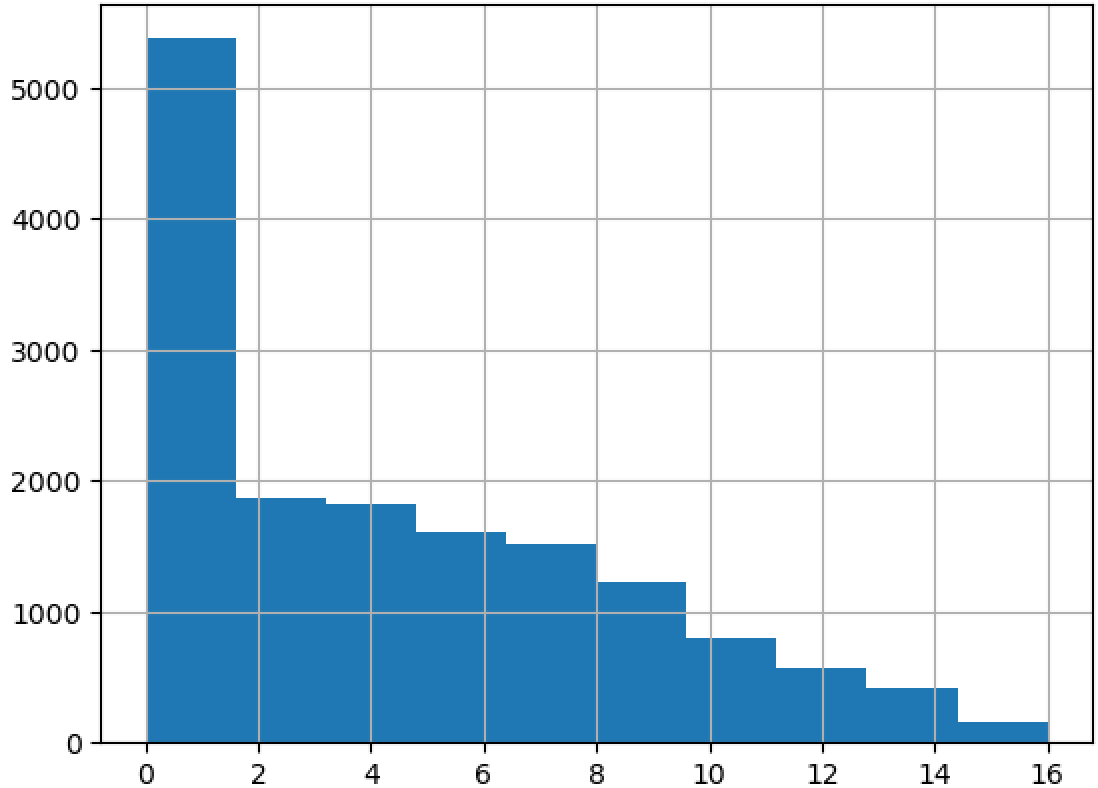
Out[73]:
```
cloud_cover              6.0
sunshine                 3.5
global_radiation        95.0
max_temp                15.0
mean_temp               11.4
min_temp                 7.8
precipitation            0.0
pressure            101620.0
snow_depth               0.0
dtype: float64
```

In [74]:
```
import matplotlib.pyplot as plt
cols = weather_df.columns
print(cols)
for col in cols:
    if col!='date':
        weather_df[col].hist()
        print(col)
        plt.show()
```

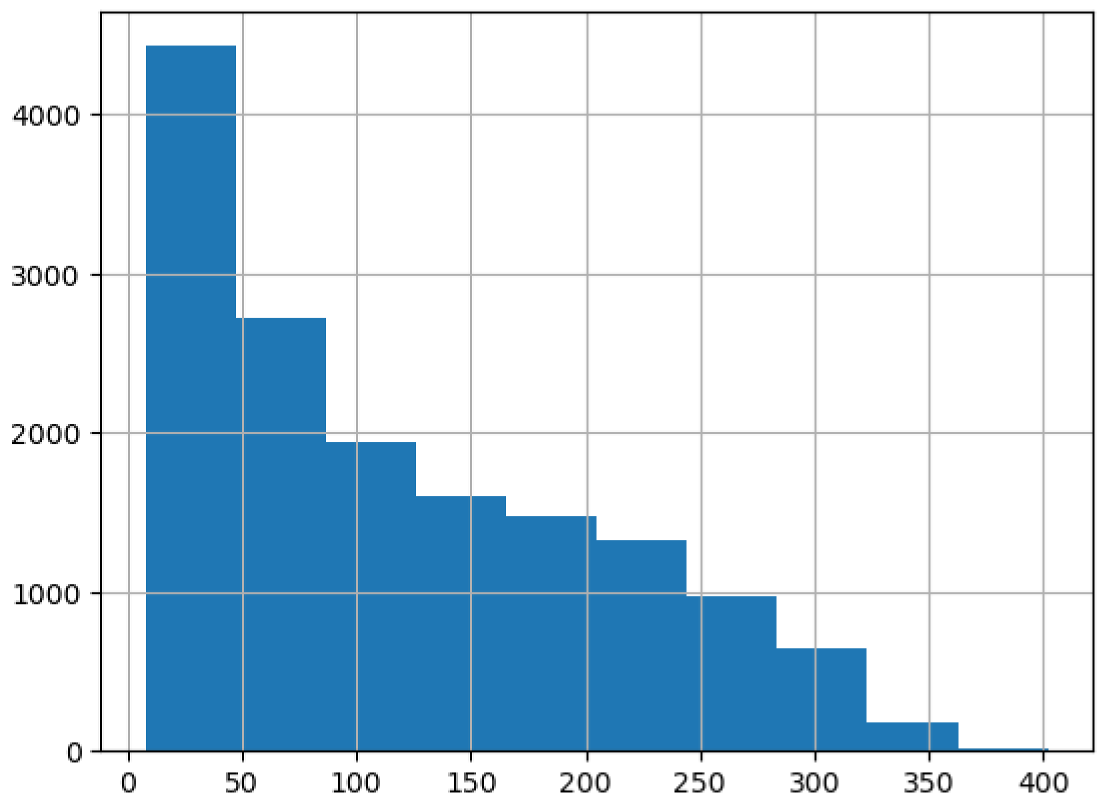
```
Index(['date', 'cloud_cover', 'sunshine', 'global_radiation', 'max_temp',
       'mean_temp', 'min_temp', 'precipitation', 'pressure', 'snow_depth'],
      dtype='object')
cloud_cover
```



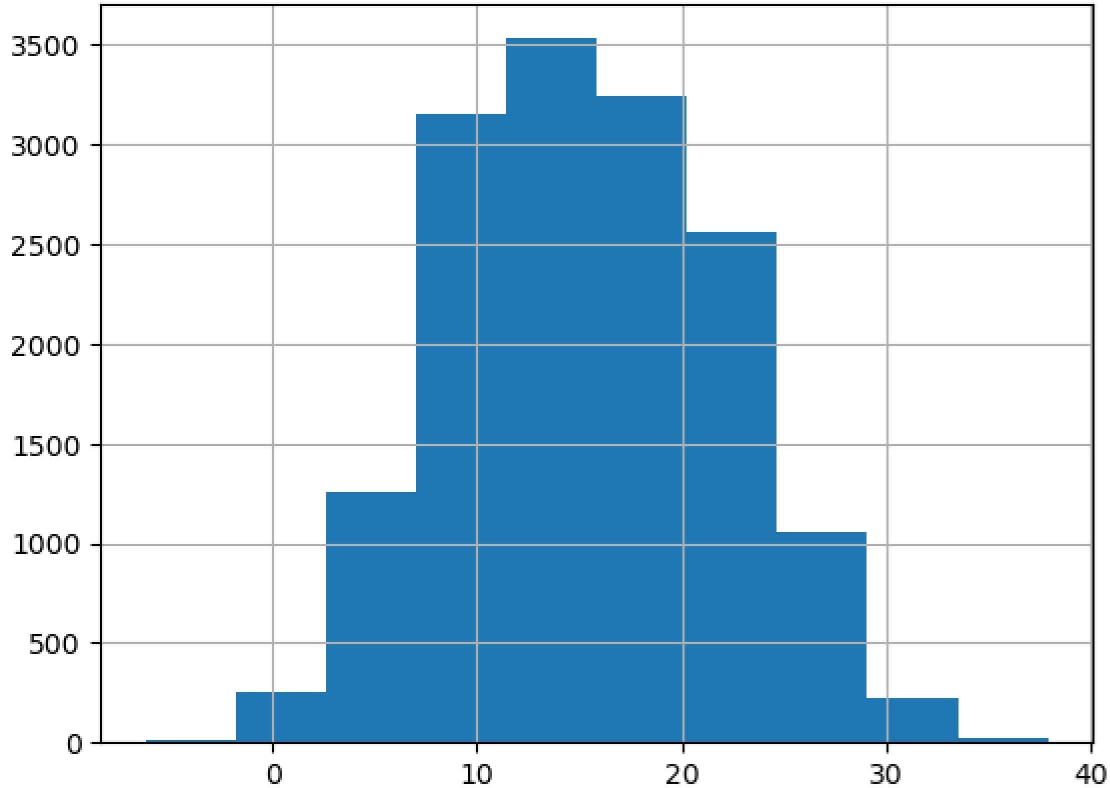

```
sunshine
```



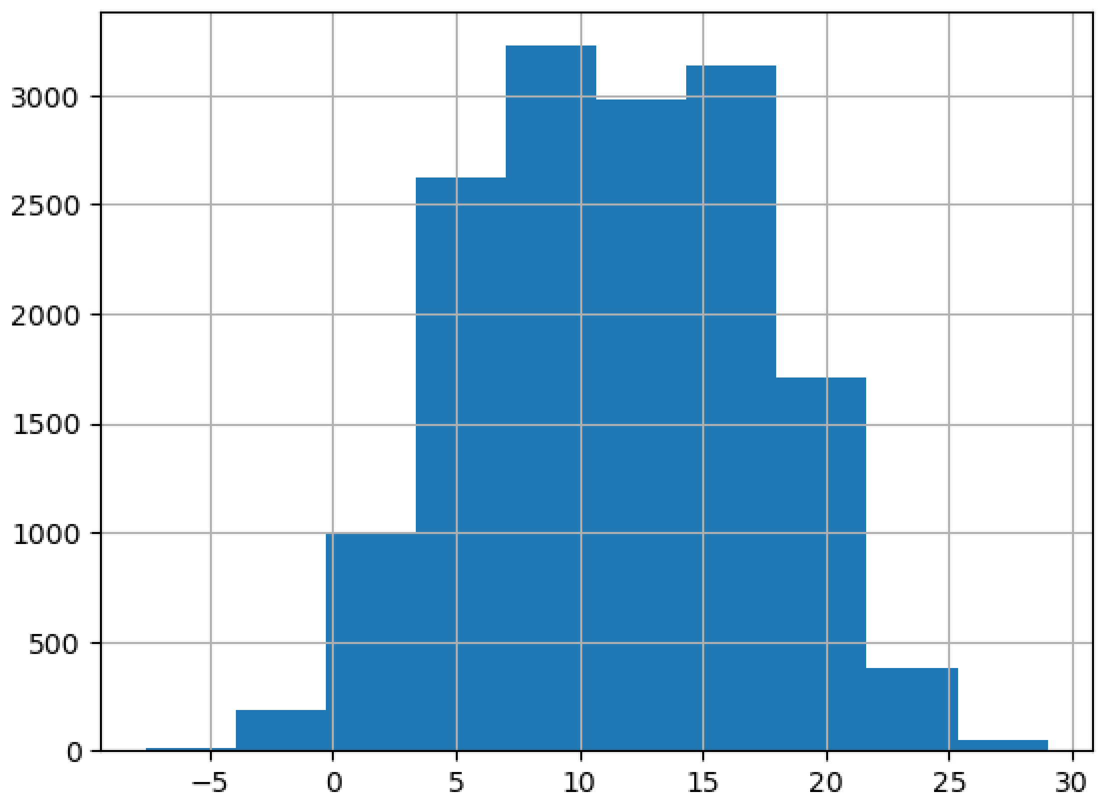
global_radiation



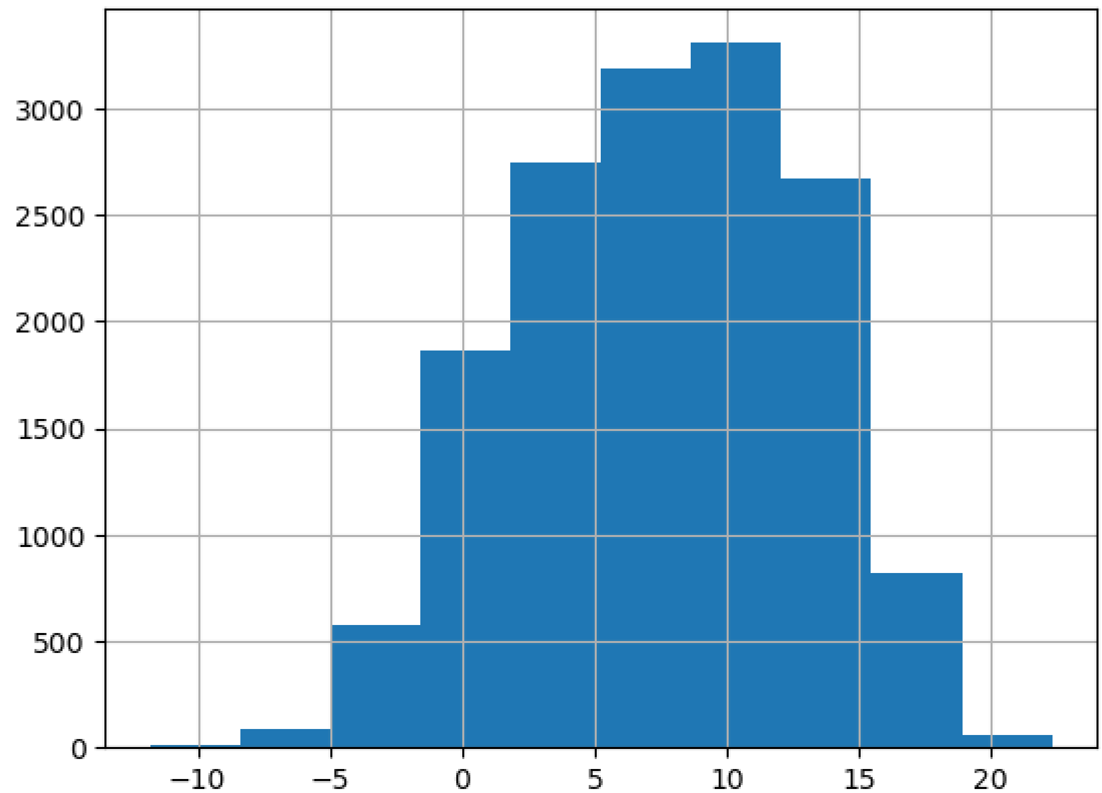
max_temp



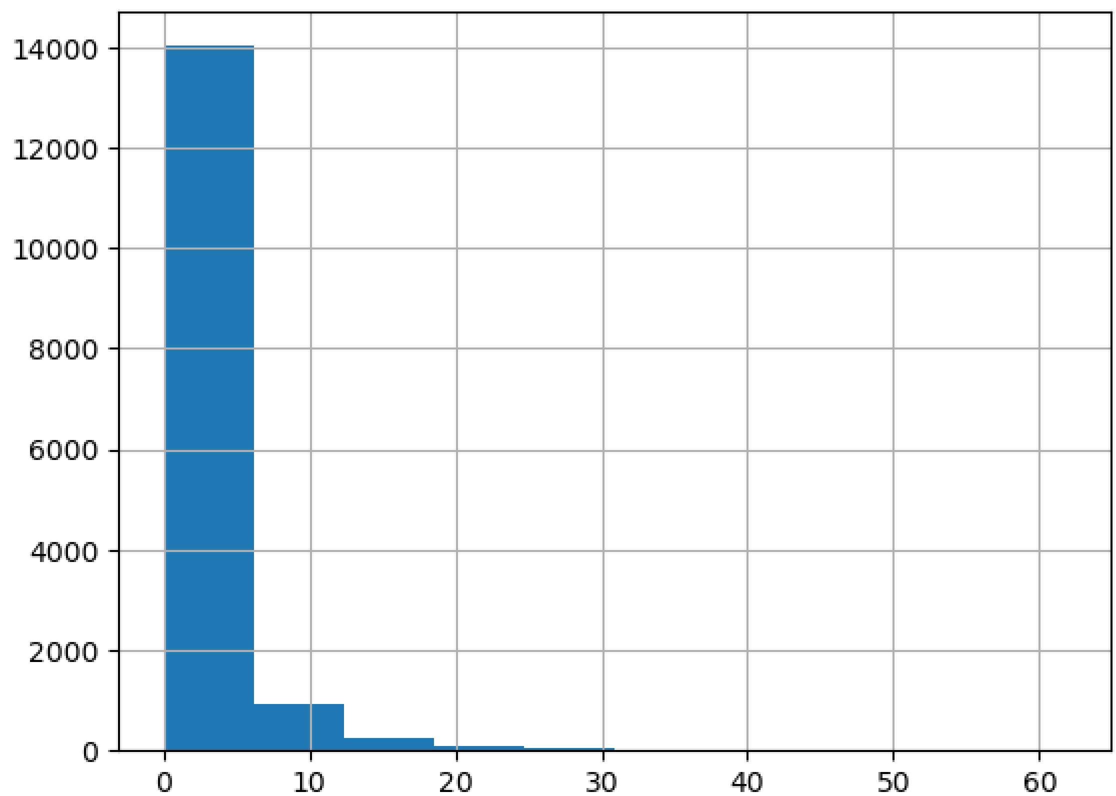
mean_temp



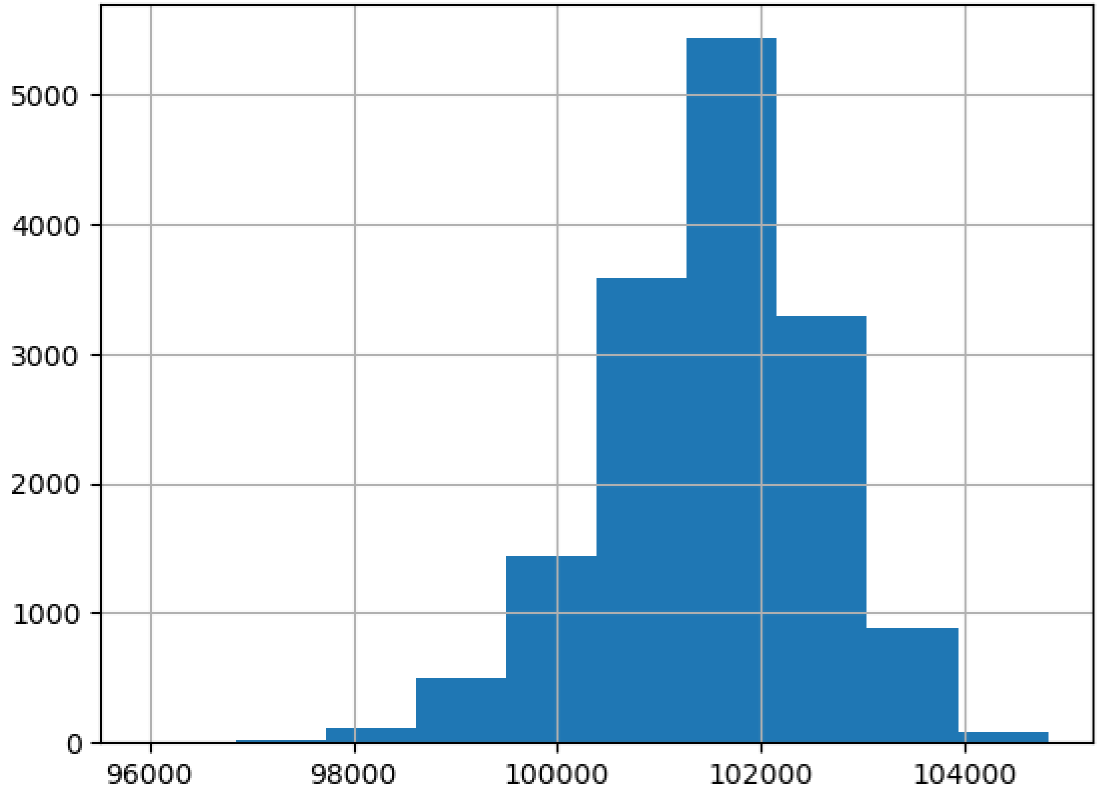
min_temp



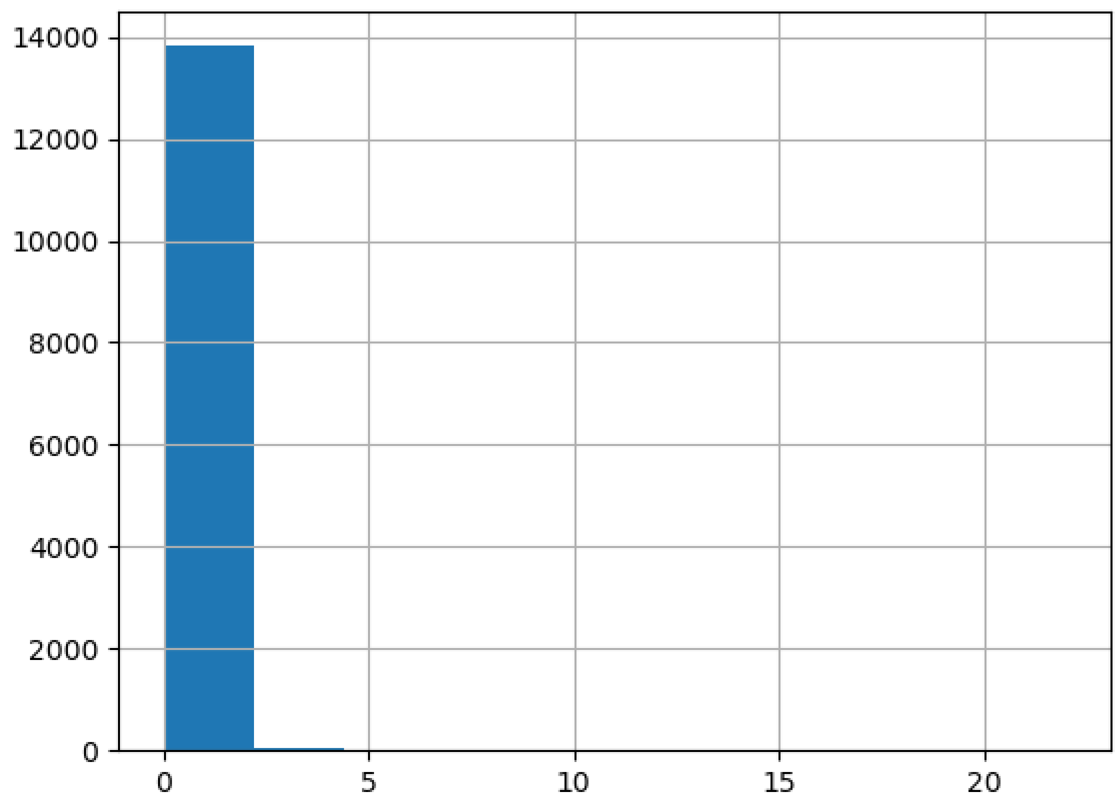
`precipitation`



`pressure`



snow_depth



In [75]:

```
weather_df.corr()
```

Out[75]:

	date	cloud_cover	sunshine	global_radiation	max_temp	mean_temp
date	1.000000	-0.107418	0.007392	0.005143	0.089504	0.097955
cloud_cover	-0.107418	1.000000	-0.738291	-0.485973	-0.212224	-0.110556
sunshine	0.007392	-0.738291	1.000000	0.852467	0.471538	0.395693
global_radiation	0.005143	-0.485973	0.852467	1.000000	0.690596	0.634935
max_temp	0.089504	-0.212224	0.471538	0.690596	1.000000	0.912065
mean_temp	0.097955	-0.110556	0.395693	0.634935	0.912065	1.000000
min_temp	0.099467	0.048838	0.217961	0.477336	0.810189	0.955504
precipitation	0.008279	0.235269	-0.232014	-0.162962	-0.071962	-0.010552
pressure	-0.013893	-0.241955	0.228081	0.151216	0.101722	0.006152
snow_depth	-0.044495	-0.001256	-0.033818	-0.061404	-0.129924	-0.154359

Question 1

(a) Plot a graph of the date (x-axis) versus ftse, the UK based stock index. Hover your cursor on the graph and guess the month and year when the highest value occurred.

In [6]:
```
import plotly.express as px
```

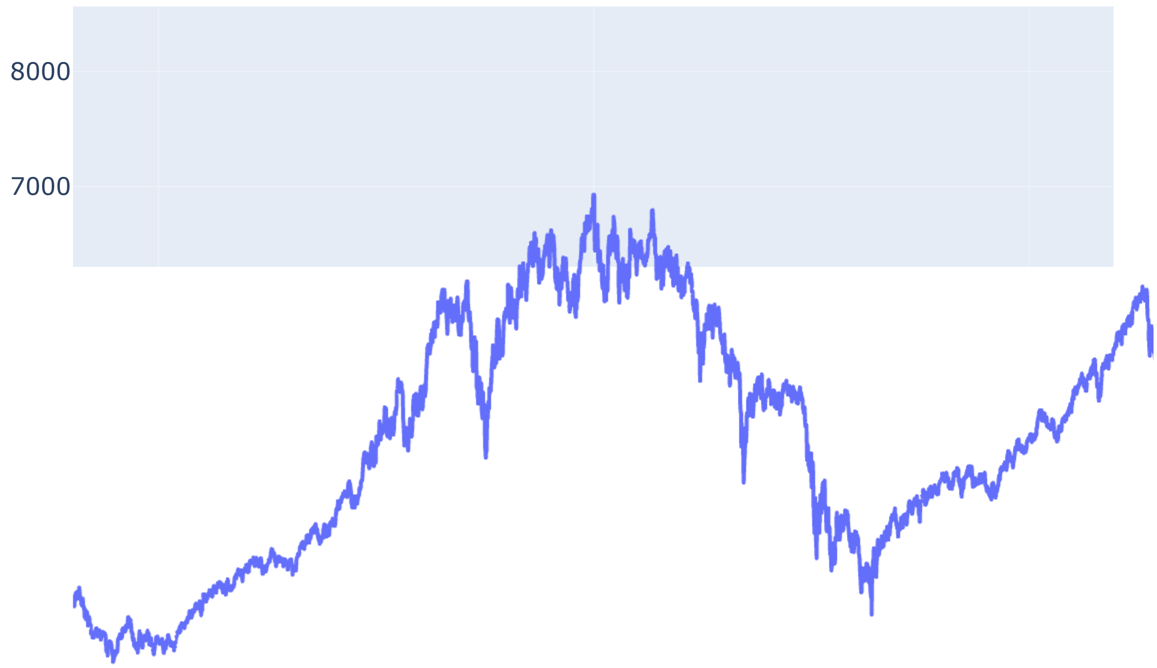
In [7]:
```
fig = px.line(stock_df, x='Date', y='ftse', title='Interactive Plot')
```

In [9]:
```
max_value = stock_df['ftse'].max()
max_value_row = stock_df[stock_df['ftse'] == max_value]
```

In [10]:
```
for index, row in max_value_row.iterrows():
    fig.add_annotation(x=row['Date'], y=row['ftse'],
                       text=f"Highest Value: {row['ftse']} on {row['Date'].str
                       showarrow=True, arrowhead=1)
```

In [11]:
```
fig.show()
```

Interactive Plot

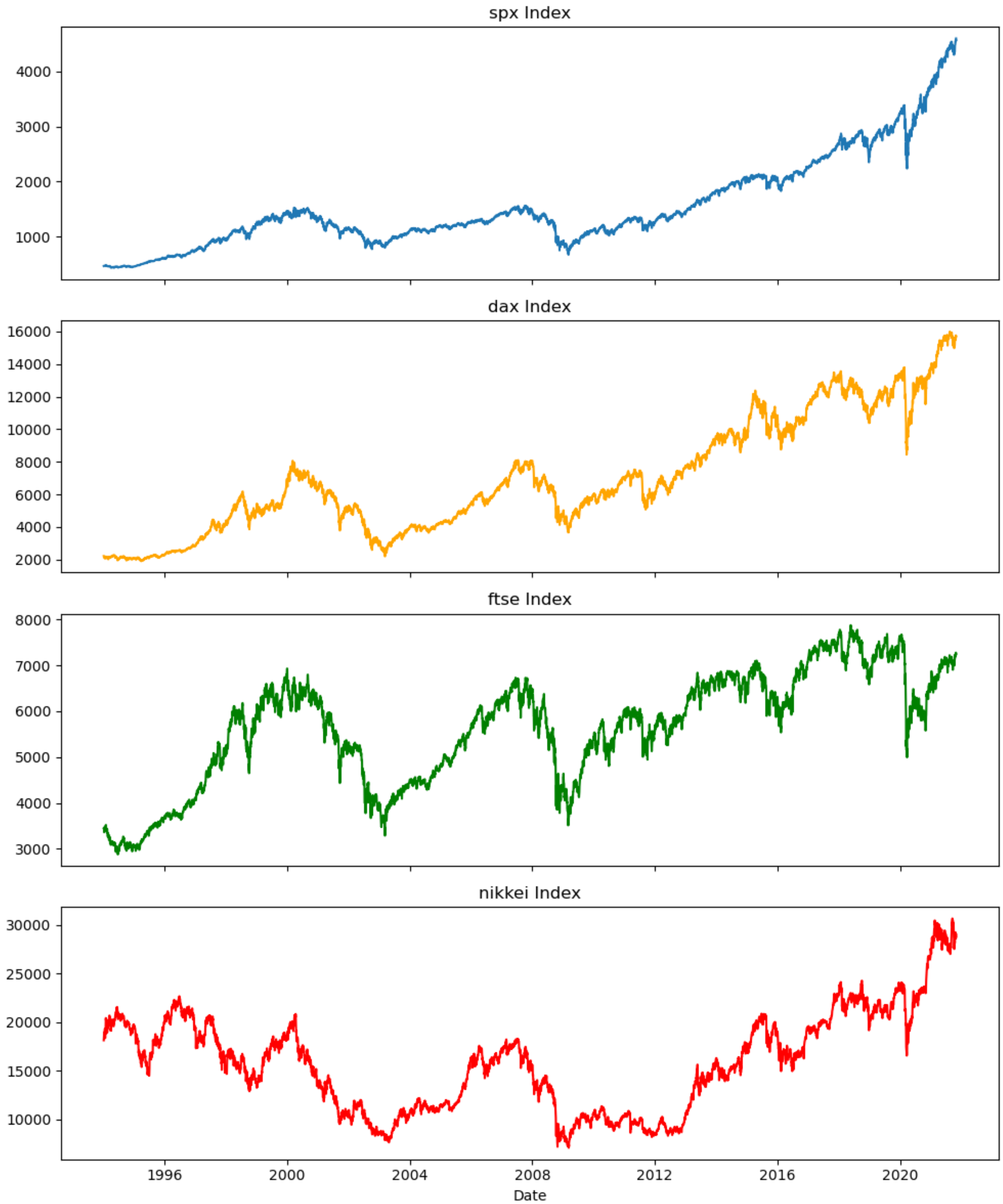


b) Create 4 sub-plots one on top of the other, one for each of the four stock indices (spx, dax, ftse, and nikkei) against dates in the x-axis. By just eyeballing which of the four had the greatest dip during COVID onset in 2020?

In [81]:

```
import matplotlib.pyplot as plt
fig1, axs = plt.subplots(4, 1, figsize=(10, 12), sharex=True)

axs[0].plot(stock_df['Date'], stock_df['spx'], label='spx')
axs[0].set_title('spx Index')

axs[1].plot(stock_df['Date'], stock_df['dax'], label='dax', color='orange')
axs[1].set_title('dax Index')

axs[2].plot(stock_df['Date'], stock_df['ftse'], label='ftse', color='green')
axs[2].set_title('ftse Index')

axs[3].plot(stock_df['Date'], stock_df['nikkei'], label='nikkei', color='red')
axs[3].set_title('nikkei Index')

# Set a common x-axis label
plt.xlabel('Date')

# Improve layout
plt.tight_layout()
```

```
# Show the plot
plt.show()
```

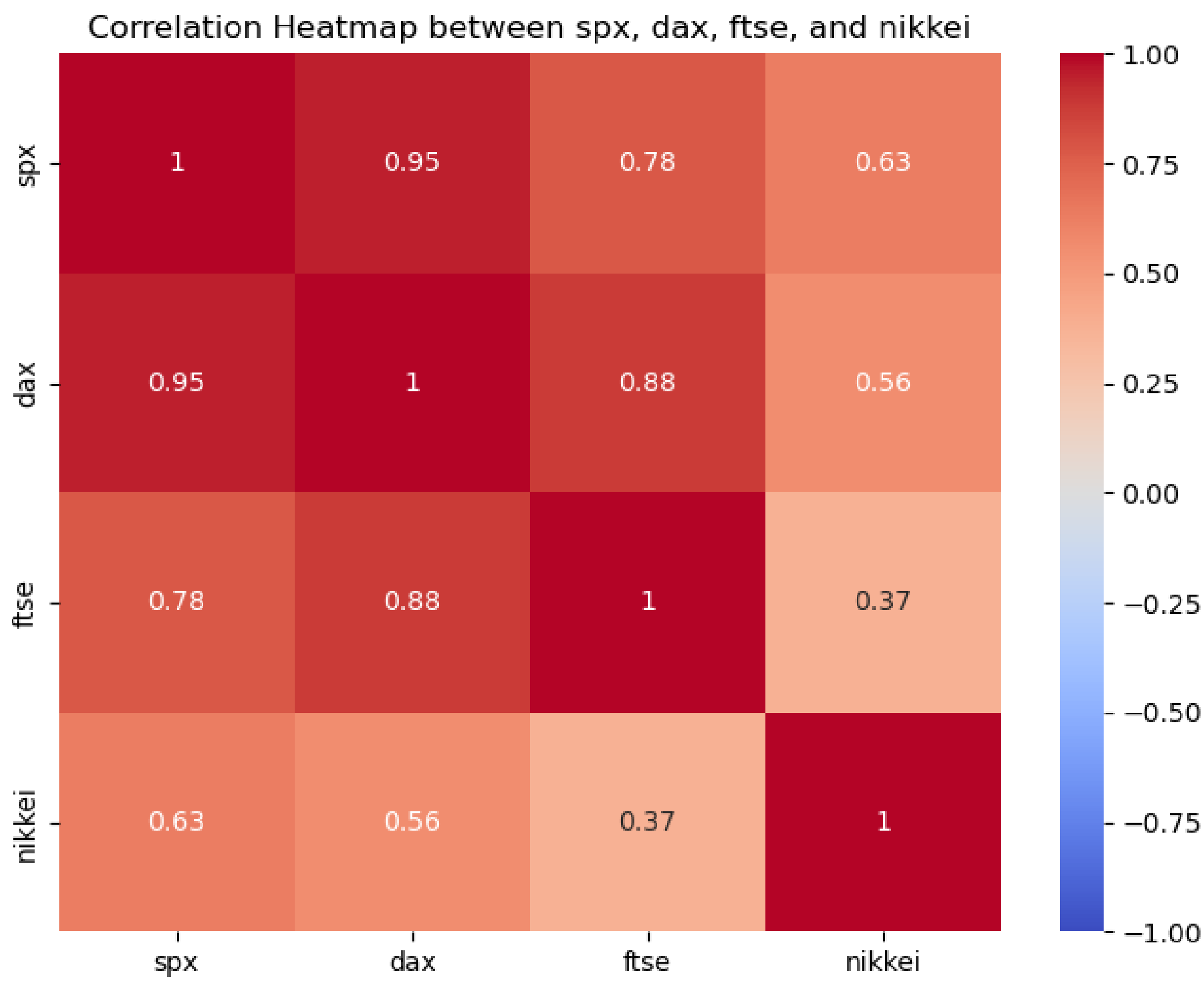


ftse had the greatest dip in 2020

(c) Using the above 4 subplots, what are the other times there is a global fall in stock markets? Can you state what events these corresponded to?

(d) Obtain a heat map of the correlations between all four indices (for the entire dura- tion). Comment on the correlations highlighting what you expected to be correlated or uncorrelated based on the graphs. Were there any surprises?

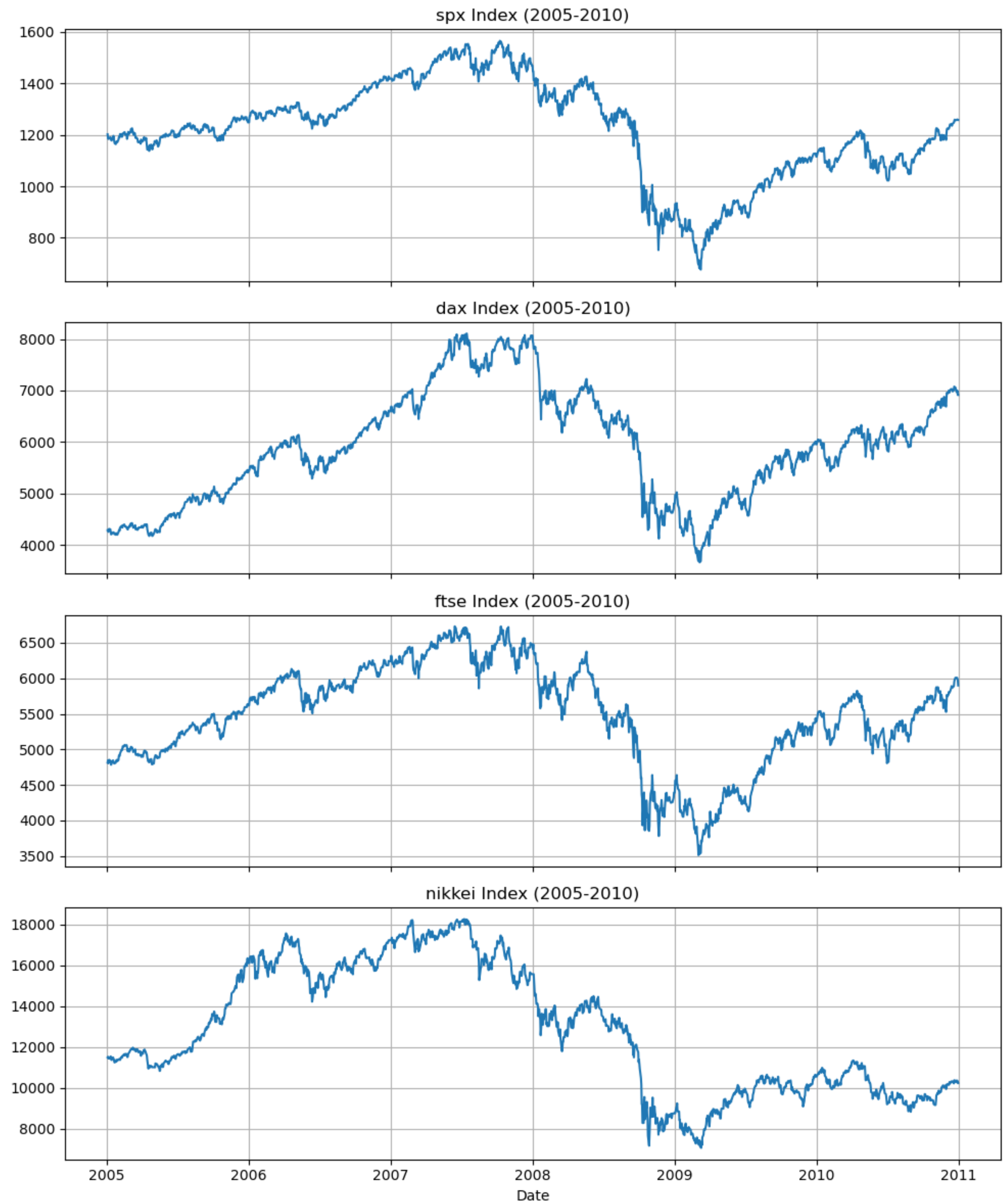
In [82]:
```
import seaborn as sns
df_without_date = stock_df.drop('Date', axis=1)
corr_matrix = df_without_date.corr()

# Create a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Heatmap between spx, dax, ftse, and nikkei')
plt.show()
```



(e) Create 4 more subplots, now just using years 2005, 2006, 2007, 2008, 2009, and 2010 data. Do the four indices behave similarly? Write your thoughts about the trends.

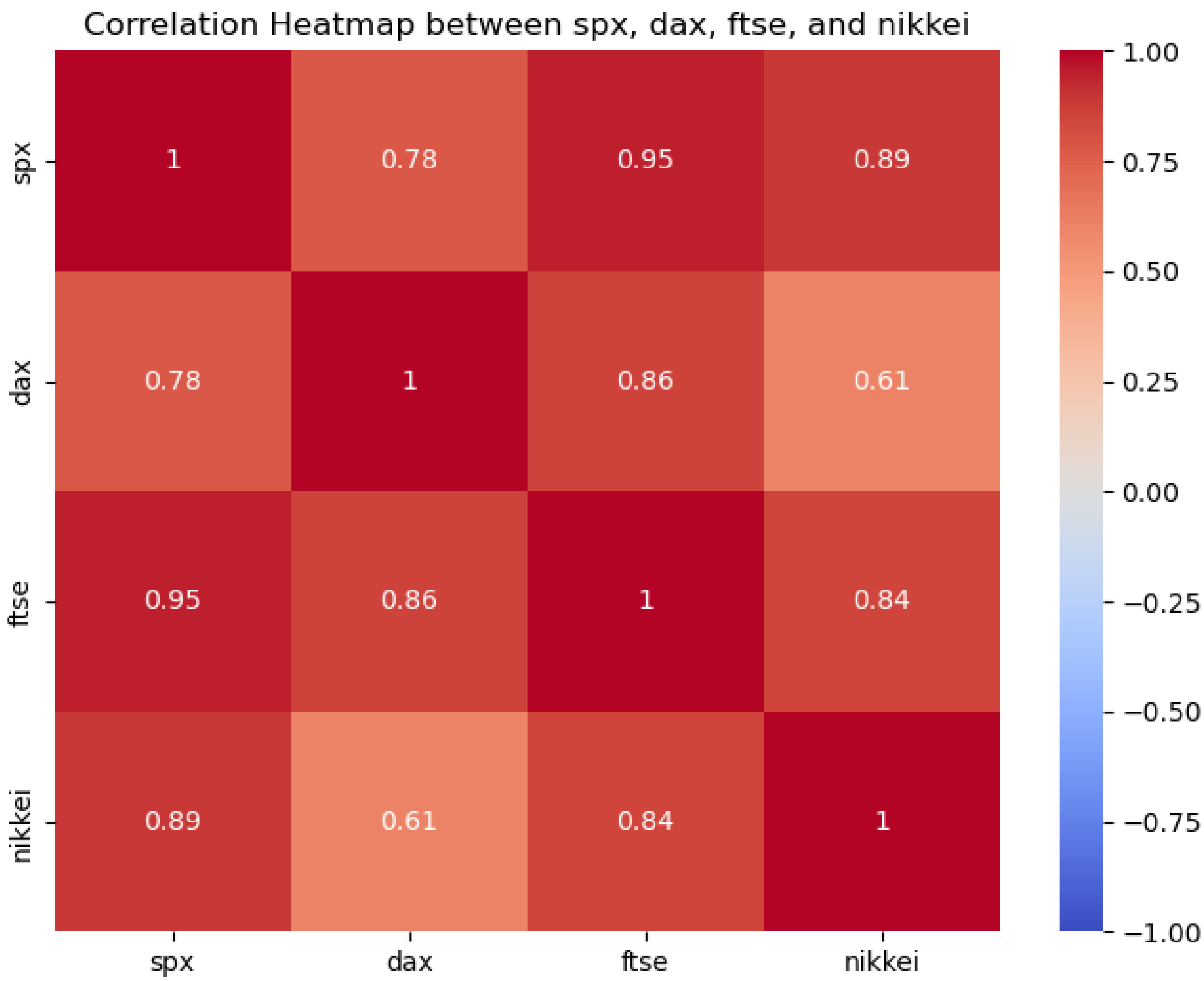
In [36]:
```
filtered_df = stock_df[stock_df['Date'] >= '2005-01-01']
filtered_df = filtered_df[filtered_df['Date'] <= '2010-12-31']

# Create subplots
fig, axs = plt.subplots(4, 1, figsize=(10, 12), sharex=True)

# Plot each stock index in a separate subplot
for i, index in enumerate(['spx', 'dax', 'ftse', 'nikkei']):
```

```
    axs[i].plot(filtered_df['Date'], filtered_df1[index], label=index)
    axs[i].set_title(f'{index} Index (2005-2010)')
    axs[i].grid(True)

# Set a common x-axis label
plt.xlabel('Date')

# Improve layout
plt.tight_layout()

# Show the plot
plt.show()
```



(f) Now obtain a heat map only for years 2005-2010 (both included). Which two indices were most correlated earlier for the full data and which two are most correlated now?

In [40]:
```
df_without_date = filtered_df.drop('Date', axis=1)

# Calculate the correlation matrix for the filtered data
corr_matrix1 = df_without_date.corr()

plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix1, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Heatmap between spx, dax, ftse, and nikkei')
plt.show()
```



Question 2

(a) Subset the data by only considering the years 2014, 2015, 2016, 2017, and 2018 for both the weather data as well as the stock index data. Which data set has NaN values? And in which columns are they?

In [44]:
```
filtered_weather_df = weather_df[(weather_df['date'].dt.year >= 2014) & (weathe
filtered_stock_df = stock_df[(stock_df['Date'].dt.year >= 2014) & (stock_df['Da

weather_na_columns = filtered_weather_df.columns[filtered_weather_df.isna().any
stock_na_columns = filtered_stock_df.columns[filtered_stock_df.isna().any()].t
```

```
print("Weather Data NaN columns:", weather_na_columns)
print("Stock Data NaN columns:", stock_na_columns)
```

```
Weather Data NaN columns: ['global_radiation', 'snow_depth']
Stock Data NaN columns: []
```

(b) Use df['column'].interpolate(inplace = True) to interpolate the values of NaN as the data is already sorted by dates. State the number of rows (n) at this stage for each of the data sets and also check there are no NaNs.

In [45]:
```
for column in weather_df.columns:
    if weather_df[column].isna().any():
        weather_df[column].interpolate(inplace=True)
```

In [46]:
```
for column in stock_df.columns:
    if stock_df[column].isna().any():
        stock_df[column].interpolate(inplace=True)
```

In [47]:
```
n_weather_df = len(weather_df)
n_stock_df = len(stock_df)
print("Number of rows in Weather DataFrame:", n_weather_df)
print("Number of rows in Stock DataFrame:", n_stock_df)
```

```
Number of rows in Weather DataFrame: 15341
Number of rows in Stock DataFrame: 7255
```

In [49]:
```
weather_df_na_columns = weather_df.columns[weather_df.isna().any()].tolist()
stock_df_na_columns = stock_df.columns[stock_df.isna().any()].tolist()

print("Weather DataFrame NaN columns after interpolation:", weather_df_na_colu
print("Stock DataFrame NaN columns after interpolation:", stock_df_na_columns)
```

```
Weather DataFrame NaN columns after interpolation: []
Stock DataFrame NaN columns after interpolation: []
```

(c) Use only the date and 'ftse' columns from the stock data, and merge those columns with the London weather data. Use the date field as the merge key. Use all the rows of the weather data. How many NaN rows are in the resulting set?

In [76]:
```
stock_df.rename(columns={'Date': 'date'}, inplace=True)
merged_df = pd.merge(weather_df, stock_df[['date', 'ftse']], on='date', how='le

# Count the number of NaN rows in the resulting dataset
nan_count = merged_df['ftse'].isna().sum()
print("Number of NaN rows in the resulting dataset:", nan_count)
```

```
Number of NaN rows in the resulting dataset: 8284
```

(d) The stock market does not have any data published on holidays. Fill those NaN using interpolate. Also drop the column 'Date' as it is the same as 'date'. How many rows of NaN are in the merged dataset now? Also, how many columns are in the merged set now?

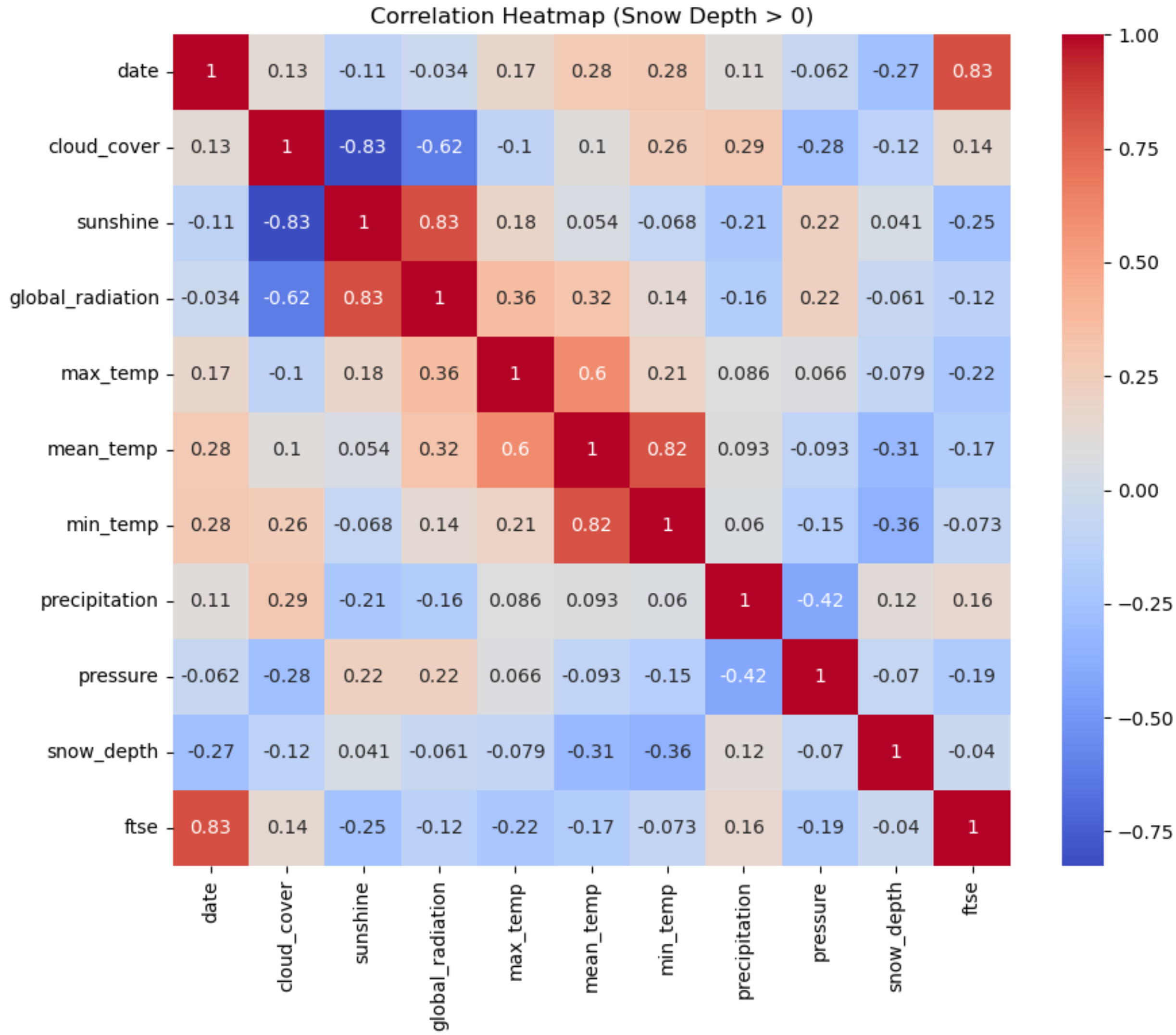
In [57]:
```
merged_df['ftse'].interpolate(inplace=True)

# Drop the duplicate 'date' column if exists
if 'Date' in merged_df.columns:
    merged_df.drop('Date', axis=1, inplace=True)
```

```
# Count the number of NaN rows after interpolation
nan_count_after = merged_df['ftse'].isna().sum()

# Count the number of columns in the merged dataset
num_columns = len(merged_df.columns)

print("Number of NaN rows in the merged dataset after interpolation:", nan_cou
print("Number of columns in the merged dataset:", num_columns)
```

```
Number of NaN rows in the merged dataset after interpolation: 5485
Number of columns in the merged dataset: 11
```

(e) Obtain a heat map of the correlations between all the numerical columns but only for a subset of merged data when snow depth is greater than zero. So looks like the closing index value is dependent on the weather that day provided there was some snow depth! Which variables is 'ftse' most and least (i.e. most negative) correlated?

In [59]:
```
snow_depth_df = merged_df[merged_df['snow_depth'] > 0]

# Calculate the correlation matrix for the filtered data
corr_matrix = snow_depth_df.corr()

# Create a heatmap using Seaborn
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap (Snow Depth > 0)')

plt.show()
```



In [60]:
```python
ftse_correlations = corr_matrix['ftse'].drop('ftse')  # Exclude self-correlati
most_correlated = ftse_correlations.idxmax()
least_correlated = ftse_correlations.idxmin()

print("Variable most correlated with 'ftse':", most_correlated)
print("Variable least correlated (most negatively) with 'ftse':", least_correla
```

```
Variable most correlated with 'ftse': date
Variable least correlated (most negatively) with 'ftse': sunshine
```

In [ ]: