In [1]:
```
import pandas as pd
import panel as pn
import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output
import plotly.graph_objects as go
import hvplot.pandas
import matplotlib.pyplot as plt
import numpy as np
import os
import sys
from pathlib import Path
import seaborn as sns
%matplotlib inline
import alpaca_trade_api as tradeapi
from dotenv import load_dotenv
import requests

import warnings
warnings.filterwarnings('ignore')
```

```
WARNING:param.main: pandas could not register all extension types imports failed with the following error: cannot import name 'ABCIndexClass' from 'pandas.core.dtypes.generic' (C:\Users\srava\anaconda3\envs\pyvizenv\lib\site-packages\pandas\core\dtypes\generic.py)
```

```
Bad key "text.kerning_factor" on line 4 in
C:\Users\srava\anaconda3\envs\pyvizenv\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_test_patch.mplstyle.
You probably need to get an updated matplotlibrc file from
http://github.com/matplotlib/matplotlib/blob/master/matplotlibrc.template
or from the matplotlib source distribution
```

In [2]:
```
airpassengers_Path = Path("C:/Users/srava/OneDrive/Documents/Rutgers FinTech Bootcamp/Project 1/US Monthly Air Passengers.csv")
airpass_df = pd.read_csv(airpassengers_Path, index_col='DEST_COUNTRY_NAME')
airpass_df.head()
```

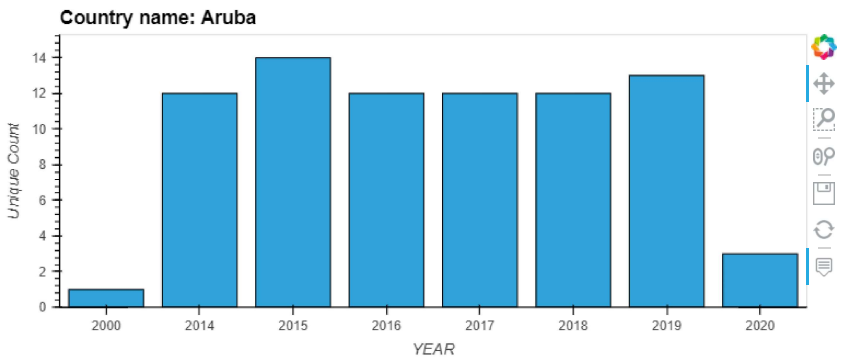
Out[2]:

DEST_COUNTRY_NAME	Sum_PASSENGERS	AIRLINE_ID	CARRIER_NAME	ORIGIN	ORIGIN_CITY_NAME	ORIGIN_STATE_ABR	ORIGIN_STATE_NM	ORIGIN_COUNTRY	ORIGIN_COUNTRY_NAME	DEST	DEST_CITY_N
United States	0	NaN	NaN	AEX	Alexandria, LA	LA	Louisiana	US	United States	AEX	Alexand
United States	0	NaN	NaN	AEX	Alexandria, LA	LA	Louisiana	US	United States	AFW	Dallas/Fort V
United States	0	NaN	NaN	AEX	Alexandria, LA	LA	Louisiana	US	United States	ATL	Atlan
Colombia	89	NaN	NaN	AEX	Alexandria, LA	LA	Louisiana	US	United States	BOG	Bogota, Col
Colombia	108	NaN	NaN	AEX	Alexandria, LA	LA	Louisiana	US	United States	BOG	Bogota, Col

In [3]:
```
airpass_df = airpass_df.rename_axis('Country name')

airpass_df_dropped = airpass_df.drop(columns =['Sum_PASSENGERS', 'AIRLINE_ID','CARRIER_NAME', 'ORIGIN','ORIGIN_CITY_NAME', 'ORIGIN_STATE_ABR',
                                'ORIGIN_STATE_NM', 'ORIGIN_COUNTRY', 'ORIGIN_COUNTRY_NAME', 'DEST', 'DEST_CITY_NAME', 'DEST_STATE_ABR',
                                'DEST_STATE_NM','DEST_COUNTRY','MONTH'])

airpass_df_dropped['Unique Count'] = 0
airpass_df_grp = airpass_df_dropped.groupby(['Country name','YEAR'])['Unique Count'].count()


airpass_updated = pd.DataFrame(airpass_df_grp).reset_index()
```

In [4]:
```
airpass_updated.hvplot.bar(x='YEAR',
                           y='Unique Count',
                           groupby='Country name'
                           )
```

Out[4]:



In [5]:
```
hist_world_happiness_Path = Path("C:/Users/srava/OneDrive/Documents/Rutgers FinTech Bootcamp/Project 1/Historical World Happiness Data.xls")
hist_happiness_df = pd.read_excel(hist_world_happiness_Path, index_col='Country name')
hist_happiness_df.head()
```

Out[5]:

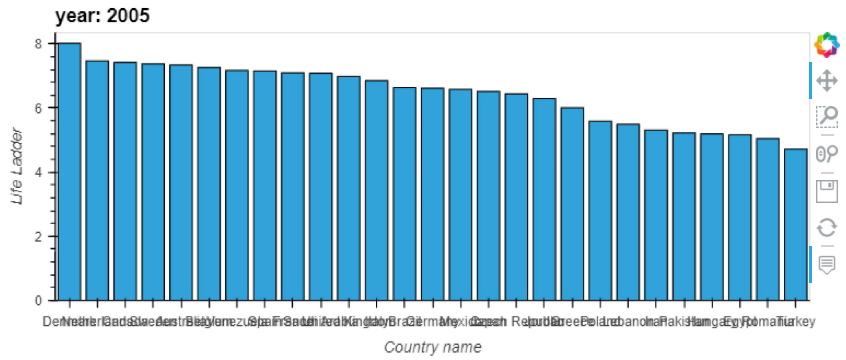
Country name	year	Life Ladder	Log GDP per capita	Social support	Healthy life expectancy at birth	Freedom to make life choices	Generosity	Perceptions of corruption	Positive affect	Negative affect
Denmark	2005	8.018934	10.851397	0.972372	69.599998	0.971135	NaN	0.236522	0.859549	0.153672
Denmark	2008	7.970892	10.880102	0.953912	70.080002	0.969788	0.272087	0.247505	0.756866	0.163091

Country name	year	Life Ladder	Log GDP per capita	Social support	Healthy life expectancy at birth	Freedom to make life choices	Generosity	Perceptions of corruption	Positive affect	Negative affect
Finland	2020	7.889350	10.750446	0.961621	72.099998	0.962424	-0.115532	0.163636	0.744292	0.192898
Finland	2018	7.858107	10.782932	0.962155	71.900002	0.937807	-0.127380	0.198605	0.781546	0.181781
Denmark	2007	7.834233	10.891111	0.954201	69.919998	0.932086	0.240024	0.206006	0.827860	0.194324

In [6]:
```
hist_happiness_df_drp = hist_happiness_df.drop(columns =['Log GDP per capita', 'Social support','Healthy life expectancy at birth',
                                                         'Freedom to make life choices', 'Generosity','Perceptions of corruption',
                                                         'Positive affect', 'Negative affect' ])
cons_hist_happiness=hist_happiness_df_drp.reset_index()
```

In [7]:
```
cons_hist_happiness.hvplot.bar(x="Country name",
                               y="Life Ladder",
                               groupby="year")
```

Out[7]:



In [8]:
```
curr_world_happiness_path = Path("C:/Users/srava/OneDrive/Documents/Rutgers FinTech Bootcamp/Project 1/world-happiness-report-2021.csv")
curr_happiness_df = pd.read_csv(curr_world_happiness_path, index_col = 'Country name')
curr_happiness_df.head()
```

Out[8]:

Country name	Regional indicator	Ladder score	Standard error of ladder score	upperwhisker	lowerwhisker	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption	Ladder score in Dystopia	Explained by: Log GDP per capita	Explained by: Social support	Explained by: Healthy life expectancy	Explained by: Freedom to make life choices	Explained by: Generosity
Finland	Western Europe	7.842	0.032	7.904	7.780	10.775	0.954	72.0	0.949	-0.098	0.186	2.43	1.446	1.106	0.741	0.691	0.124
Denmark	Western Europe	7.620	0.035	7.687	7.552	10.933	0.954	72.7	0.946	0.030	0.179	2.43	1.502	1.108	0.763	0.686	0.208
Switzerland	Western Europe	7.571	0.036	7.643	7.500	11.117	0.942	74.4	0.919	0.025	0.292	2.43	1.566	1.079	0.816	0.653	0.204
Iceland	Western Europe	7.554	0.059	7.670	7.438	10.878	0.983	73.0	0.955	0.160	0.673	2.43	1.482	1.172	0.772	0.698	0.293
Netherlands	Western Europe	7.464	0.027	7.518	7.410	10.932	0.942	72.4	0.913	0.175	0.338	2.43	1.501	1.079	0.753	0.647	0.302

In [9]:
```
curr_happiness_df['year'] = '2021'
curr_happiness_df_drp = curr_happiness_df.drop(columns =['Regional indicator','Standard error of ladder score','upperwhisker',
                                                         'lowerwhisker','Logged GDP per capita', 'Social support', 'Freedom to make life choices',
                                                         'Generosity','Perceptions of corruption', 'Ladder score in Dystopia','Explained by: Log GDP per capita',
                                                         'Explained by: Social support', 'Explained by: Healthy life expectancy', 'Explained by: Freedom to make life choices',
                                                         'Explained by: Generosity', 'Explained by: Perceptions of corruption', 'Dystopia + residual','Healthy life expectancy'
                                                          ])

curr_happiness_df_drp = curr_happiness_df.rename(columns = {"Ladder score":"Life Ladder"})
```

In [10]:
```
cons_happiness_df = pd.concat([hist_happiness_df_drp,curr_happiness_df_drp], axis = 'rows', join = 'inner')

cons_happiness_df_chart = cons_happiness_df.reset_index()
```

In [11]:
```
cons_happiness_df_chart.hvplot.table(columns=['Country name', 'year', 'Life Ladder'], width=400, groupby='year')
```

Out[11]:

year: 2005

#	Country name	year	Life Ladder
0	Denmark	2005	8.018934
1	Netherlands	2005	7.463979
2	Canada	2005	7.418048
3	Sweden	2005	7.376316
4	Australia	2005	7.340688
5	Belgium	2005	7.26229
6	Venezuela	2005	7.169621

7	Spain	2005	7.152786
8	France	2005	7.093393
9	Saudi Arabia	2005	7.079644
10	United Kingdom	2005	6.983557

In [12]:
```
combined_df = pd.merge(cons_happiness_df,
                       airpass_df_grp,
                       how='left',
                        left_on=[cons_happiness_df.index,'year'],
                       right_on=['Country name', 'YEAR']
                      )

combined_df
```

Out[12]:

	Country name	year	Life Ladder	Unique Count
0	Denmark	2005	8.018934	1.0
1	Denmark	2008	7.970892	NaN
2	Finland	2020	7.889350	NaN
3	Finland	2018	7.858107	NaN
4	Denmark	2007	7.834233	NaN
...	...	...	...	...
2093	Lesotho	2021	3.512000	NaN
2094	Botswana	2021	3.467000	NaN
2095	Rwanda	2021	3.415000	NaN
2096	Zimbabwe	2021	3.145000	NaN
2097	Afghanistan	2021	2.523000	NaN

2098 rows × 4 columns

In [13]:
```
combined_df.hvplot.table(columns=['Country name', 'Life Ladder', 'Unique Count'], width=400, groupby='year')
```

Out[13]:

year: 2005

#	Country name	Life Ladder	Unique Count
0	Denmark	8.018934	1.0
1	Netherlands	7.463979	63.0
2	Canada	7.418048	453.0
3	Sweden	7.376316	NaN
4	Australia	7.340688	NaN
5	Belgium	7.26229	NaN
6	Venezuela	7.169621	NaN
7	Spain	7.152786	NaN
8	France	7.093393	12.0
9	Saudi Arabia	7.079644	NaN
10	United Kingdom	6.983557	23.0

In [14]:
```
#Load dot_enc
load_dotenv()
#Loading alpaca api key/secret key
alpaca_api_key = os.getenv("ALPACA_API_KEY")
alpaca_secret_key = os.getenv("ALPACA_SECRET_KEY")
```

In [16]:
```
alpaca = tradeapi.REST(
    alpaca_api_key,
    alpaca_secret_key,
    api_version="v2")

today = pd.Timestamp("2021-07-15", tz="America/New_York").isoformat()

tickers = ["AAL", "UAL", "DAL", "RCL", "CCL", "IHG", "MAR", "WH", "WYNN",
           "YUM", "MCD", "DRI","BKNG", "EXPE"]

timeframe = "1D"

start = pd.Timestamp("2019-07-15", tz="America/New_York").isoformat()
end = pd.Timestamp("2021-07-15", tz="America/New_York").isoformat()
```

In [17]:
```
# Get current prices for Tickers
df_portfolio = alpaca.get_barset(
    tickers,
    timeframe,
    start = start,
    end = end,
    limit = 1000
).df

# Display sample data
df_portfolio
```

Out[17]:

	AAL					BKNG					...	WYNN					YUM				
	open	high	low	close	volume	open	high	low	close	volume	...	open	high	low	close	volume	open	high	low	close	volume

time	AAL					BKNG					...	WYNN					YUM				
	open	high	low	close	volume	open	high	low	close	volume	...	open	high	low	close	volume	open	high	low	close	volume
time																					
2019-07-15 00:00:00-04:00	33.67	33.9500	33.42	33.61	3024568	1881.89	1885.24	1872.290	1881.91	250052	...	133.9300	137.99	133.420	137.79	2388872	111.29	112.13	111.21	111.92	774893
2019-07-16 00:00:00-04:00	33.70	34.6700	33.70	34.21	5870319	1879.89	1897.22	1875.570	1888.89	216107	...	138.0000	141.26	136.850	139.34	2123795	111.81	112.15	111.18	111.52	758445
2019-07-17 00:00:00-04:00	34.34	34.4199	33.11	33.25	5452541	1887.75	1897.09	1866.530	1866.69	214952	...	139.2500	139.33	135.450	135.90	1167572	111.53	112.10	111.37	111.77	724956
2019-07-18 00:00:00-04:00	33.41	33.8200	33.11	33.75	2594330	1861.83	1895.68	1861.830	1885.91	196706	...	135.4400	135.73	134.370	135.09	748169	112.23	113.37	112.00	113.07	962982
2019-07-19 00:00:00-04:00	33.82	34.0200	33.06	33.07	3913009	1896.03	1901.59	1882.840	1883.00	164141	...	135.8900	137.01	134.565	134.66	798910	113.46	113.59	112.48	112.54	966126
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2021-07-09 00:00:00-04:00	20.79	21.0000	20.50	20.88	20301340	2182.08	2205.34	2160.445	2194.44	256517	...	113.1989	114.94	112.340	113.86	2967611	118.36	118.85	117.88	118.56	828657
2021-07-12 00:00:00-04:00	20.73	20.9800	20.35	20.85	19443662	2182.06	2206.18	2166.500	2204.08	207439	...	112.8700	114.10	111.710	114.00	1603254	118.03	118.26	117.36	117.86	872340
2021-07-13 00:00:00-04:00	20.74	20.7000	19.99	20.01	27563886	2204.40	2228.74	2185.810	2191.39	291733	...	112.9300	113.30	111.380	111.53	1748400	117.68	117.72	116.89	117.09	997915
2021-07-14 00:00:00-04:00	20.88	21.5700	20.43	20.62	51025349	2206.02	2218.10	2175.890	2179.82	160705	...	112.2000	113.32	109.800	110.70	1826333	117.21	118.04	116.92	117.13	799400
2021-07-15 00:00:00-04:00	20.56	21.0590	20.15	20.46	32684463	2173.00	2183.31	2145.500	2170.41	222206	...	109.9000	110.66	106.750	108.20	2646857	116.81	117.21	116.09	117.00	860891

506 rows × 70 columns

In [18]:
```
df_closing_prices = pd.DataFrame()
df_closing_prices["AAL"] = df_portfolio["AAL"]["close"]
df_closing_prices["UAL"] = df_portfolio["UAL"]["close"]
df_closing_prices["DAL"] = df_portfolio["DAL"]["close"]
df_closing_prices["RCL"] = df_portfolio["RCL"]["close"]
df_closing_prices["CCL"] = df_portfolio["CCL"]["close"]
df_closing_prices["IHG"] = df_portfolio["IHG"]["close"]
df_closing_prices["MAR"] = df_portfolio["MAR"]["close"]
df_closing_prices["WH"] = df_portfolio["WH"]["close"]
df_closing_prices["WYNN"] = df_portfolio["WYNN"]["close"]
df_closing_prices["YUM"] = df_portfolio["YUM"]["close"]
df_closing_prices["MCD"] = df_portfolio["MCD"]["close"]
df_closing_prices["DRI"] = df_portfolio["DRI"]["close"]
df_closing_prices["BKNG"] = df_portfolio["BKNG"]["close"]
df_closing_prices["EXPE"] = df_portfolio["EXPE"]["close"]

# Drop the time component of the date
df_closing_prices.index = df_closing_prices.index.date

df_closing_prices
```

Out[18]:

	AAL	UAL	DAL	RCL	CCL	IHG	MAR	WH	WYNN	YUM	MCD	DRI	BKNG	EXPE
2019-07-15	33.61	91.31	61.68	112.840	46.95	69.61	142.750	60.73	137.79	111.92	214.18	125.43	1881.91	134.71
2019-07-16	34.21	94.01	63.17	114.560	46.45	69.91	143.960	61.19	139.34	111.52	213.70	123.21	1888.89	136.37
2019-07-17	33.25	94.75	62.23	111.700	45.64	69.70	141.240	59.60	135.90	111.77	213.73	122.21	1866.69	135.12
2019-07-18	33.75	95.25	62.66	111.000	45.49	69.78	139.800	59.97	135.09	113.07	215.89	124.06	1885.91	136.45
2019-07-19	33.07	93.82	60.91	110.170	45.09	69.56	137.700	58.79	134.66	112.54	213.91	124.39	1883.00	135.10
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2021-07-09	20.88	51.10	42.92	82.330	24.26	67.00	142.480	72.20	113.86	118.56	235.70	148.64	2194.44	167.75
2021-07-12	20.85	50.62	42.85	81.555	23.86	65.76	141.770	71.93	114.00	117.86	235.69	149.95	2204.08	167.37
2021-07-13	20.01	48.51	41.34	78.520	22.85	64.61	138.350	70.91	111.53	117.09	236.23	147.40	2191.39	162.03
2021-07-14	20.62	48.16	40.69	76.210	22.57	64.35	139.679	70.44	110.70	117.13	237.14	147.37	2179.82	161.05
2021-07-15	20.46	47.72	41.35	74.210	21.95	64.18	139.540	70.22	108.20	117.00	236.92	144.82	2170.41	159.28

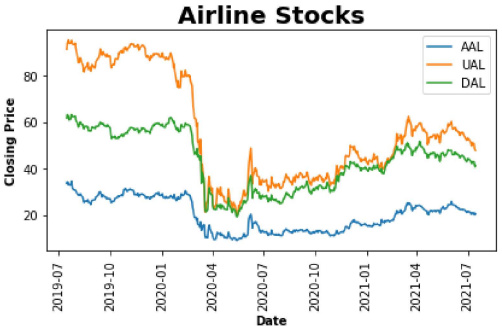
506 rows × 14 columns

In [19]:
```
#creating data frame for subsector airlines
df_airlines = pd.DataFrame()
df_airlines["AAL"] = df_closing_prices["AAL"]
df_airlines["UAL"] = df_closing_prices["UAL"]
df_airlines["DAL"] = df_closing_prices["DAL"]

#creating data frame for subsector cruiselines
df_cruiselines = pd.DataFrame()
df_cruiselines["RCL"] = df_closing_prices["RCL"]
df_cruiselines["CCL"] = df_closing_prices["CCL"]

#creating data frame for subsector hotels
df_hotels = pd.DataFrame()
df_hotels["IHG"] = df_closing_prices["IHG"]
df_hotels["MAR"] = df_closing_prices["MAR"]
df_hotels["WH"] = df_closing_prices["WH"]
df_hotels["WYNN"] = df_closing_prices["WYNN"]

#creating data frame for subsector restaurants
df_restaurants = pd.DataFrame()
df_restaurants["YUM"] = df_closing_prices["YUM"]
df_restaurants["MCD"] = df_closing_prices["MCD"]
df_restaurants["DRI"] = df_closing_prices["DRI"]


#creating data frame for subsector online travel services
df_online_travel_services = pd.DataFrame()
df_online_travel_services["BKNG"] = df_closing_prices["BKNG"]
df_online_travel_services["EXPE"] = df_closing_prices["EXPE"]
```

In [20]:
```
plt.figure(figsize=(30,20))
df_airlines.plot()
```

```
plt.xticks(fontsize=10, fontweight='normal',rotation=90)
plt.yticks(fontsize=10, fontweight='normal')
plt.xlabel('Date',fontsize=10, fontweight='bold')
plt.ylabel('Closing Price',fontsize=10, fontweight='bold')
plt.title('Airline Stocks',fontsize=20, fontweight='bold')
plt.tight_layout()
```

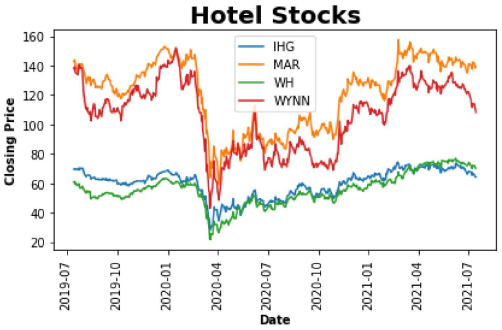
```
<Figure size 2160x1440 with 0 Axes>
```



In [21]:

```
plt.figure(figsize=(30,20))
df_hotels.plot()
plt.xticks(fontsize=10, fontweight='normal',rotation=90)
plt.yticks(fontsize=10, fontweight='normal')
plt.xlabel('Date',fontsize=10, fontweight='bold')
plt.ylabel('Closing Price',fontsize=10, fontweight='bold')
plt.title('Hotel Stocks',fontsize=20, fontweight='bold')
plt.tight_layout()
```

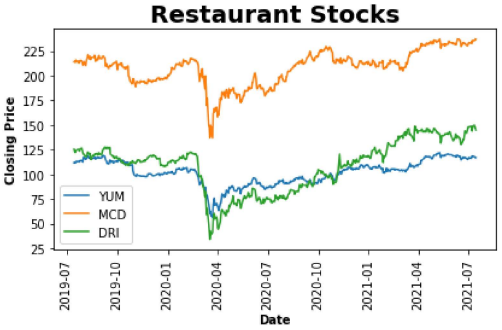
```
<Figure size 2160x1440 with 0 Axes>
```



In [22]:

```
plt.figure(figsize=(30,20))
df_restaurants.plot()
plt.xticks(fontsize=10, fontweight='normal',rotation=90)
plt.yticks(fontsize=10, fontweight='normal')
plt.xlabel('Date',fontsize=10, fontweight='bold')
plt.ylabel('Closing Price',fontsize=10, fontweight='bold')
plt.title('Restaurant Stocks',fontsize=20, fontweight='bold')
plt.tight_layout()
```

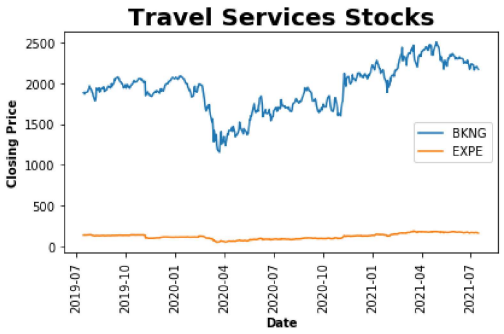
```
<Figure size 2160x1440 with 0 Axes>
```



In [23]:

```
plt.figure(figsize=(30,20))
df_online_travel_services.plot()
plt.xticks(fontsize=10, fontweight='normal',rotation=90)
plt.yticks(fontsize=10, fontweight='normal')
plt.xlabel('Date',fontsize=10, fontweight='bold')
plt.ylabel('Closing Price',fontsize=10, fontweight='bold')
plt.title('Travel Services Stocks',fontsize=20, fontweight='bold')
plt.tight_layout()
```

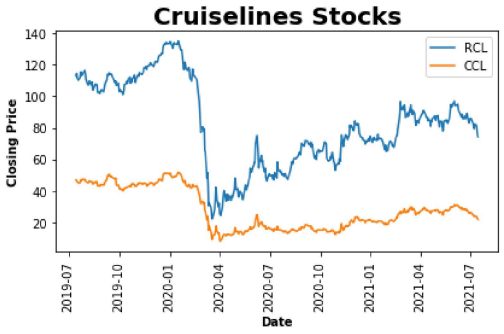
```
<Figure size 2160x1440 with 0 Axes>
```



In [24]:

```
plt.figure(figsize=(30,20))
df_cruiselines.plot()
plt.xticks(fontsize=10, fontweight='normal',rotation=90)
plt.yticks(fontsize=10, fontweight='normal')
plt.xlabel('Date',fontsize=10, fontweight='bold')
plt.ylabel('Closing Price',fontsize=10, fontweight='bold')
plt.title('Cruiselines Stocks',fontsize=20, fontweight='bold')
plt.tight_layout()
```

```
<Figure size 2160x1440 with 0 Axes>
```

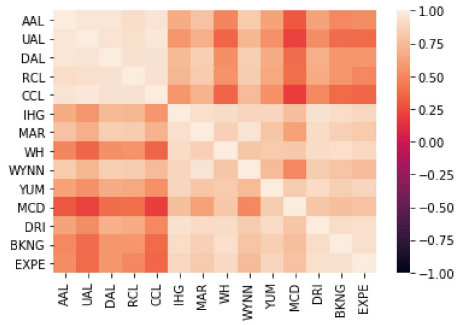


In [25]:

```
corr= df_closing_prices.corr()
sns.heatmap(corr, vmin=-1, vmax=1)
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2288e7eed48>
```



In [ ]: