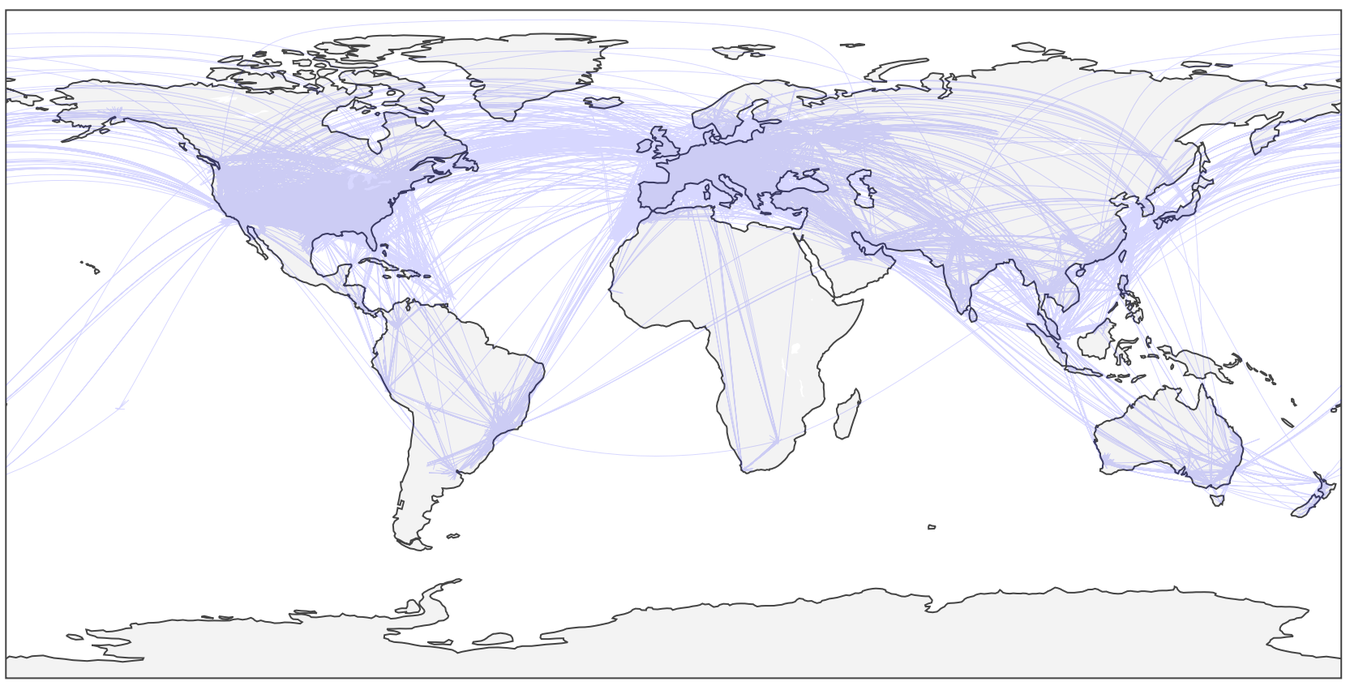
In [1]:
```
import plotly.graph_objects as go
import plotly.express as px
import pandas as pd
import datetime
import numpy as np
from scipy import signal


def geographical_density_plot(input_file,date,title_text,color_plot):

    '''
    This function takes the path to flight dataset for a specific month and
    given the date in input, it generates a geographical density plot for the
    flights being run on that specific date

    input_file   : string : file path to the flight data set for a specific month
    date         : string : date on which we want to generate the density plot
    title_text   : string : string input for the title you want to display
    color_plot   : string : color used for the density plot -> red/blue

    '''
    assert isinstance(input_file,str)
    assert isinstance(date,str)
    assert isinstance(title_text,str)
    assert isinstance(color_plot,str)

    downsampled_routes = pd.read_csv(input_file)

    fig = go.Figure()

    lons = []
    lats = []
    lons = np.empty(3 * len(downsampled_routes))
    lons[::3] = downsampled_routes['longitude_1']
    lons[1::3] = downsampled_routes['longitude_2']
    lons[2::3] = None
    lats = np.empty(3 * len(downsampled_routes))
    lats[::3] = downsampled_routes['latitude_1']
    lats[1::3] = downsampled_routes['latitude_2']
    lats[2::3] = None

    fig.add_trace(
        go.Scattergeo(
            locationmode = 'ISO-3',
            lon = lons,
            lat = lats,
            mode = 'lines',
            line = dict(width = 0.5,color = color_plot),
            opacity = 0.16

        )
    )

    fig.update_layout(
        title_text = title_text,
        showlegend = False,
        geo = go.layout.Geo(
            scope = 'world',
            showland = True,
            landcolor = 'rgb(243, 243, 243)',
            countrycolor = 'rgb(204, 204, 204)',
        ),
        height=700,
        )

    fig.show()


def country_wise_line_plot(input_file, airport_list, country_name, date1_strt, date1_end, date2_strt, date2_end):

    '''
    This function takes the path to cleaned dataset having country wise data set for 2020
    makes the line chart for the entire year depending on the major airports located in a
    particular country/continent

    input_file   : string : file path to the cleaned data set
    airport_list : list   : list of major airport from each country
    country_name : string : name of the country
    date1_strt   : string : Strict Lockdown Start Date
    date1_end    : string : Strict Lockdown End Date
    date2_strt   : string : Relaxed Lockdown Start Date
    date2_end    : string : Relaxed Lockdown End Date

    '''

    assert isinstance(input_file,str)
    assert isinstance(airport_list,list)
    assert isinstance(country_name,str)
    assert isinstance(date1_strt,str)
    assert isinstance(date1_end,str)
    assert isinstance(date2_strt,str)
    assert isinstance(date2_end,str)

    data = pd.read_csv(input_file)
    data['day'] = data['day'].apply(lambda x: datetime.datetime.strptime(x.split()[0], '%Y-%m-%d'))
    data['day'] = data['day'].apply(lambda x: x.date()).apply(str)
    data.fillna(0, inplace=True)

    fig = go.Figure()

    for airport in airport_list:
        fig.add_trace(go.Scatter(x=data['day'], y=signal.savgol_filter(data[airport],15,2),mode='lines+markers',name=airport))

    fig.update_xaxes(title_text="Day")
    fig.update_yaxes(title_text="Number of departing flights")

    date_start_str = date1_strt
    date_end_str   = date1_end
    date_start = datetime.datetime.strptime(date_start_str, "%Y-%m-%d")
    date_end = datetime.datetime.strptime(date_end_str, "%Y-%m-%d")
    fig.add_vrect(x0=date_start, x1=date_end, fillcolor="green", opacity=0.1, annotation_text="Strict Lockdown", annotation_position="top left")
    date_start_str = date2_strt
    date_end_str   = date2_end
    date_start = datetime.datetime.strptime(date_start_str, "%Y-%m-%d")
    date_end = datetime.datetime.strptime(date_end_str, "%Y-%m-%d")
    fig.add_vrect(x0=date_start, x1=date_end, fillcolor="red", opacity=0.1, annotation_text="Relaxed Lockdown", annotation_position="top left")
    title_string = "Major " + country_name + " Airports"
    fig.update_layout(title_text=title_string)
    fig.update_layout(legend_title="Major Ariports")

    fig.show()



def flight_covid_correlation_plot(flight_file, covid_file):
```

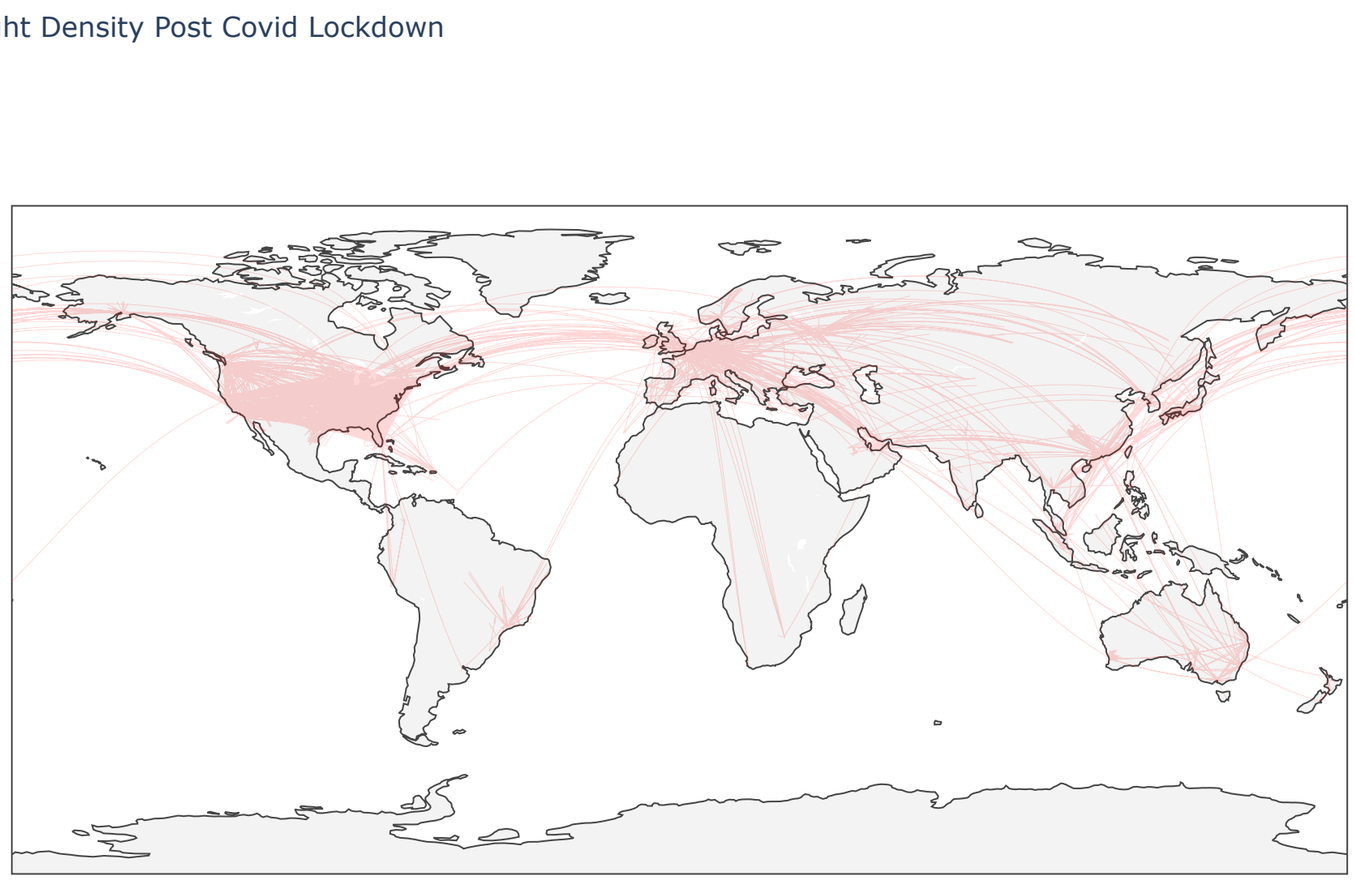
```
'''

This function takes the path to cleaned dataset having country wise data set for 2020
and path to covid data set for 2020 and generates a scatter plot correlating the impact
of flight travel with the number of covid cases per month. It concludes which country handled
covid cases better

flight_file : string : file path to the flight data set
covid_file  : string : file path to covid dataset

'''

assert isinstance(flight_file,str)
assert isinstance(covid_file,str)

data = pd.read_csv(flight_file)
data['day'] = data['day'].apply(lambda x: datetime.datetime.strptime(x.split()[0], '%Y-%m-%d'))
data['day'] = data['day'].apply(lambda x: x.date()).apply(str)
data.fillna(0, inplace=True)

covid_continent = pd.read_csv(covid_file)

australian_airports = ['YSSY','YMML']
asian_airports = ['OMDB','VABB','VIDP','WSSS','VHHH','RJBB','RJTT','RKSI','RCTP','RPLL']
europe_airports = ['LFPG','EGLL','EHAM','EDDF','LEMD','LIRF','LSZH','UUEE']
american_airpots = ["CYYZ", "KSFO", "KLAX", "KORD", "KJFK", "SBGR"]

airline_continent = pd.DataFrame(columns=['Europe','Asia','America','Australia'])

airline_continent['Europe'] = data[europe_airports].T.sum().T
airline_continent['America'] = data[american_airpots].T.sum().T
airline_continent['Asia'] = data[asian_airports].T.sum().T
airline_continent['Australia'] = data[australian_airports].T.sum().T

data.day = data.day.apply(lambda x: datetime.datetime.strptime(x.split()[0], '%Y-%m-%d'))

temp_data = data[data.day>=datetime.datetime(2020,3,1,0,0)]
temp_data.set_index('day',inplace=True)

result_df = pd.DataFrame()

result_df['date'] = temp_data.index
result_df.set_index('date',inplace=True)
result_df['Europe_flights'] = temp_data[europe_airports].T.sum().T.to_list()
result_df['American_flights'] = temp_data[american_airpots].T.sum().T.to_list()
result_df['Asian_flights'] = temp_data[asian_airports].T.sum().T.to_list()
result_df['Australia_flights'] = temp_data[australian_airports].T.sum().T.to_list()
result_df['date'] = result_df.index.astype('str')

covid_continent.date = covid_continent.date.apply(lambda x: datetime.datetime.strptime(x.split()[0], '%Y-%m-%d'))
covid_continent[covid_continent['date']<datetime.datetime(2021, 1, 1, 0, 0)]
covid_continent['month'] = covid_continent.date.dt.month

continents = ['Asia','North America','Europe','Oceania']

for continent in continents:
    cont_data = covid_continent[covid_continent.continent==continent][covid_continent.date>=datetime.datetime(2020, 3, 1, 0, 0)][covid_continent.date<datetime.datetime(2021, 1,
    result_df[continent] = cont_data.sum()['new_cases']

result_df['Asia'] = result_df['Asia'] / 45870000
result_df['North America'] = result_df['North America'] / 3670000
result_df['Europe'] = result_df['Europe'] / 7460000
result_df['Oceania'] = result_df['Oceania'] / 410000

    result_df['day'] = result_df.date.apply(lambda x: datetime.datetime.strptime(x.split()[0], '%Y-%m-%d'))
    result_df['month'] = result_df['day'].dt.month
    by_month = result_df.groupby('month').sum()
    by_month['Month'] = ['Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']

    europe = by_month[['Europe','Europe_flights','Month']]
    europe['Continent'] = ['Europe'] * len(europe)
    europe.rename(columns={'Europe': 'Total Cases', 'Europe_flights': 'Total Flights'},inplace=True)
    america = by_month[['North America','American_flights','Month']]
    america['Continent'] = ['North America'] * len(america)
    america.rename(columns={'North America': 'Total Cases', 'American_flights': 'Total Flights'},inplace=True)
    asia = by_month[['Asia','Asian_flights','Month']]
    asia['Continent'] = ['Asia'] * len(asia)
    asia.rename(columns={'Asia': 'Total Cases', 'Asian_flights': 'Total Flights'},inplace=True)
    australia = by_month[['Oceania','Australia_flights','Month']]
    australia['Continent'] = ['Oceania'] * len(australia)
    australia.rename(columns={'Oceania': 'Total Cases', 'Australia_flights': 'Total Flights'},inplace=True)

    cont = [europe,america,asia,australia]
    merged_df = pd.concat(cont)
    merged_df['Total Cases'] = merged_df['Total Cases']

    fig = px.scatter(merged_df,x='Month',y='Total Flights',size='Total Cases',color='Continent',size_max = 50)
    fig.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\__init__.py:137: UserWarning: NumPy 1.16.5 or above is required for this version of SciPy (detected version 1.16.4)
  UserWarning)
```

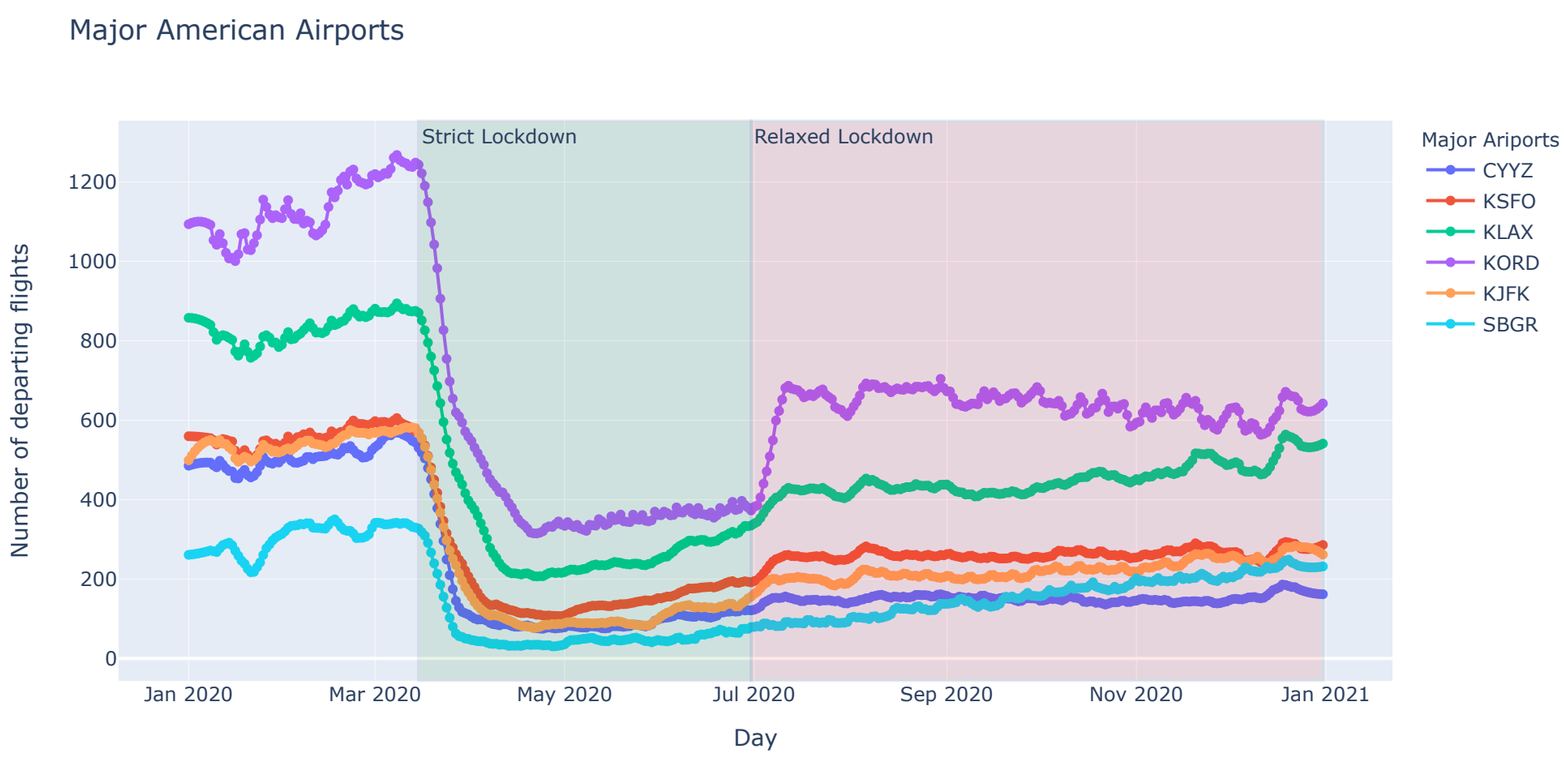
In [2]:
```
path_to_data = 'Data/flightlist_feb_2020.csv'
analysis_date = '2020-02-25'
text_title = 'Fight Density Pre Covid Lockdown'
clr = 'blue'
geographical_density_plot(path_to_data,analysis_date,text_title,clr)
```



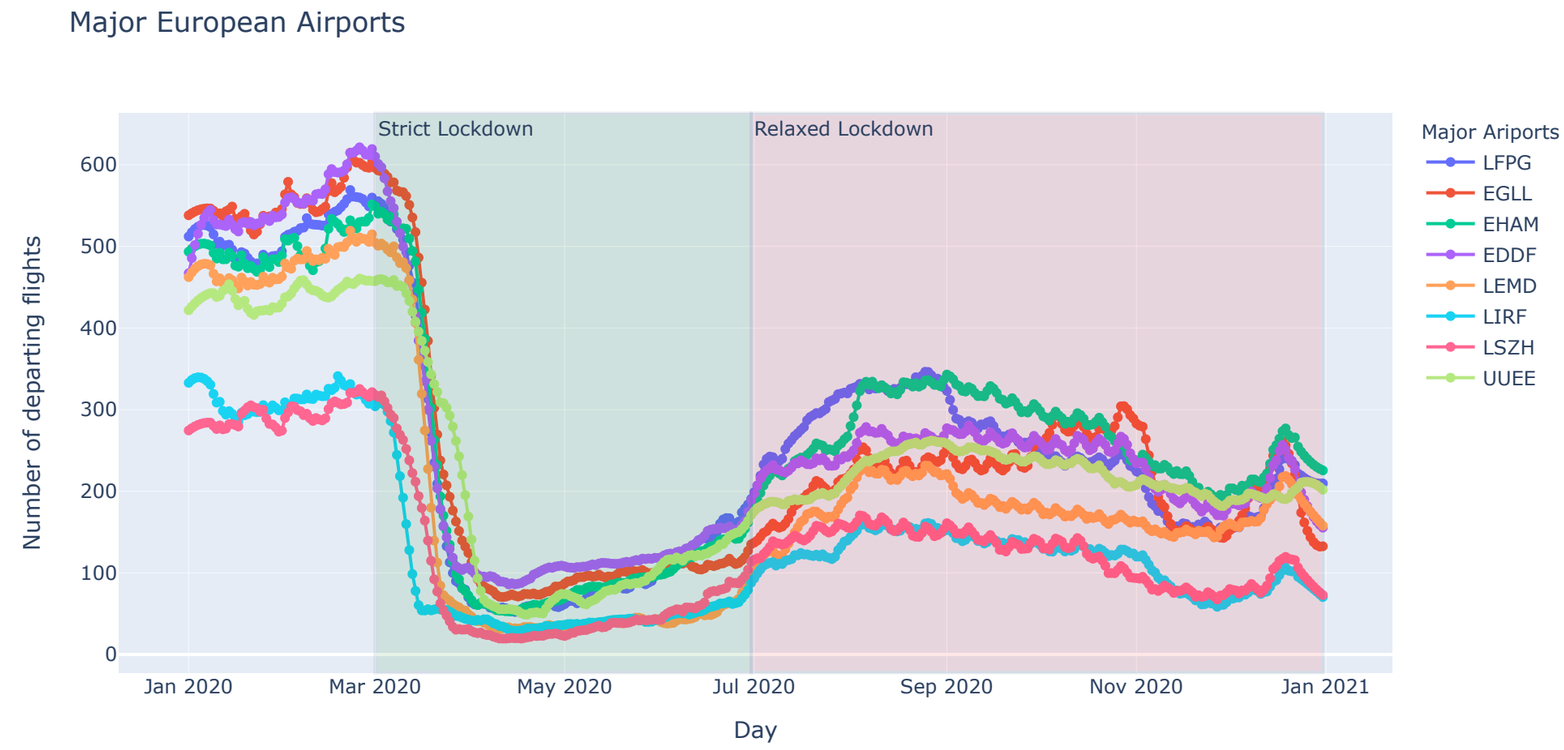
In [3]:
```
path_to_data = 'Data/flightlist_april_2020.csv'
analysis_date = '2020-04-07'
text_title = 'Fight Density Post Covid Lockdown'
clr = 'red'
geographical_density_plot(path_to_data,analysis_date,text_title,clr)
```



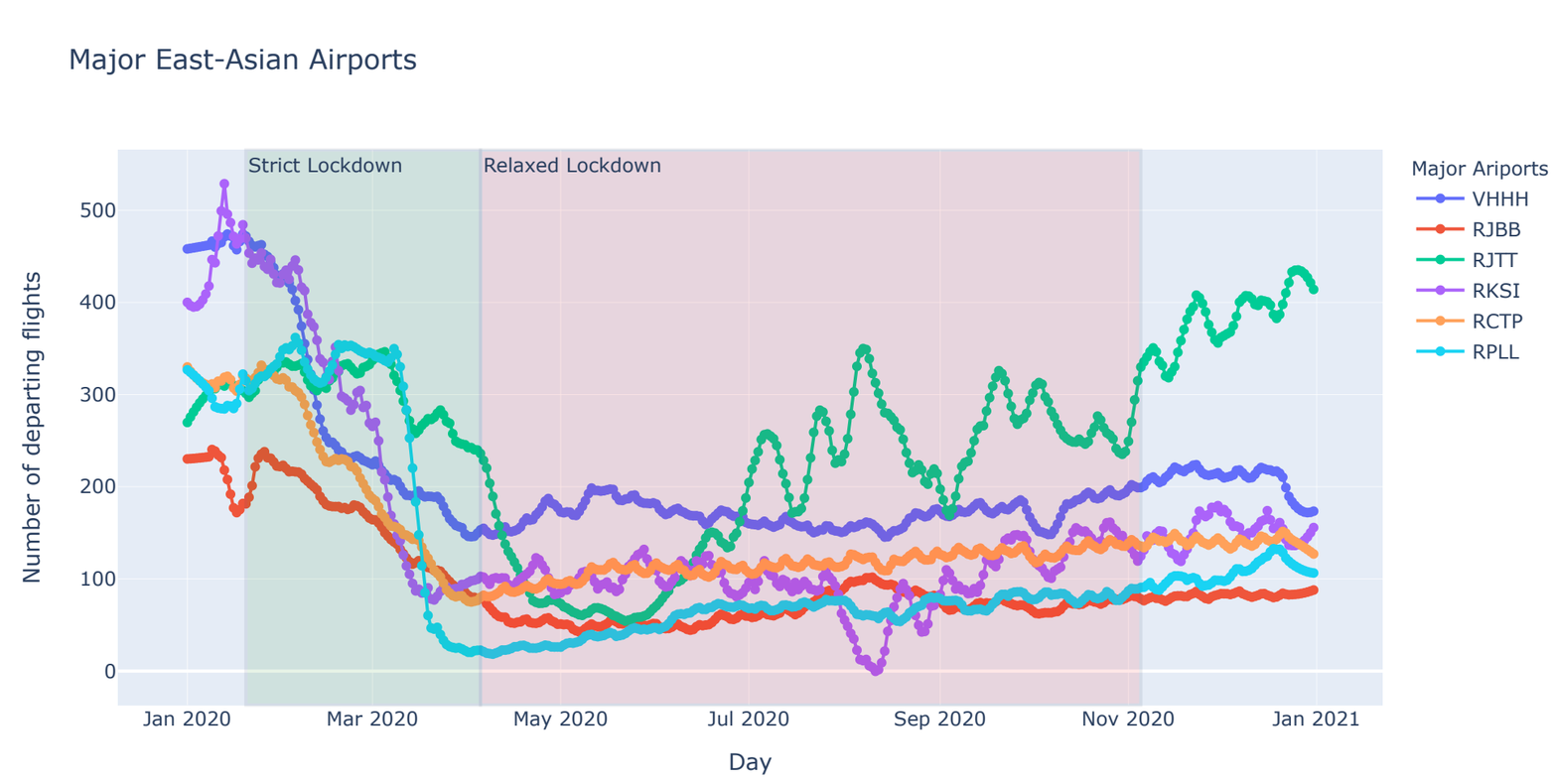
In [4]:
```
#American Airport Plots
path_to_data = 'Data/total_data_country_2020.csv'
american_airpots = ["CYYZ", "KSFO", "KLAX", "KORD", "KJFK", "SBGR"]
country = "American"
date1_strt = "2020-03-15"
date1_end = "2020-06-30"
date2_strt = "2020-06-30"
date2_end = "2020-12-31"
country_wise_line_plot(path_to_data,american_airpots,country,date1_strt,date1_end,date2_strt,date2_end)
```



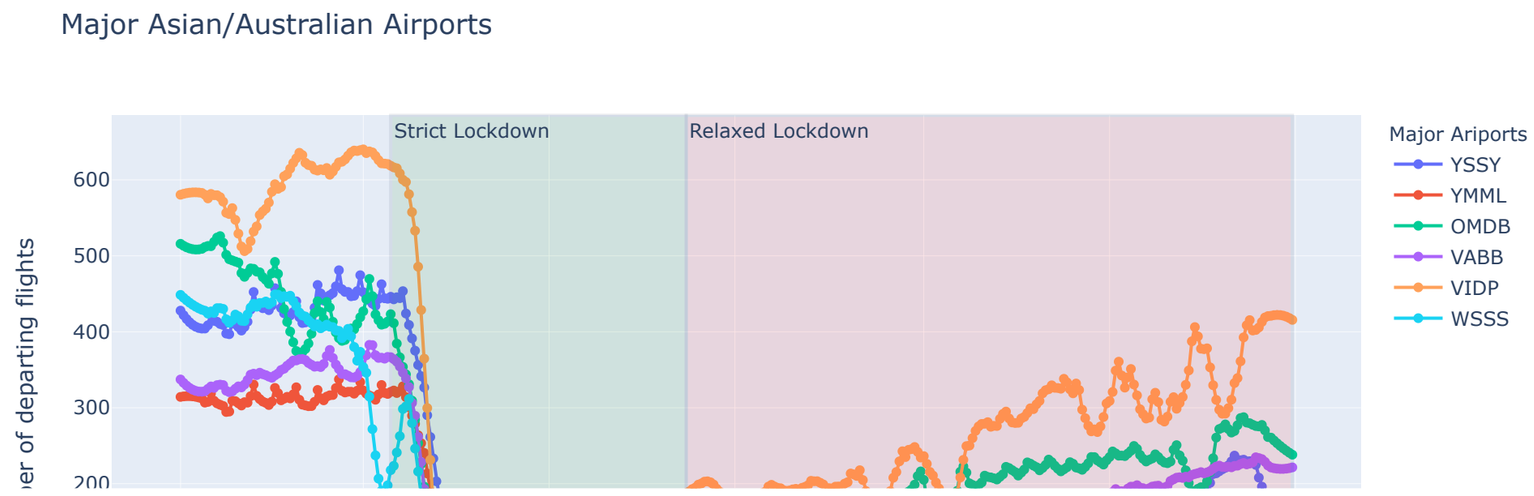
In [5]:
```
#European Airport Plots
path_to_data = 'Data/total_data_country_2020.csv'
european_airports = ["LFPG", "EGLL", "EHAM", "EDDF", "LEMD", "LIRF", "LSZH", "UUEE"]
country = "European"
date1_strt = '2020-03-01'
date1_end = '2020-06-30'
date2_strt = '2020-06-30'
date2_end = '2020-12-31'
country_wise_line_plot(path_to_data,european_airports,country,date1_strt,date1_end,date2_strt,date2_end)
```



In [6]:
```
#East-Asian Airport Plots
path_to_data = 'Data/total_data_country_2020.csv'
east_asian_airports = ["VHHH", "RJBB", "RJTT", "RKSI", "RCTP", "RPLL"]
country = "East-Asian"
date1_strt = '2020-01-20'
date1_end = '2020-04-05'
date2_strt = '2020-04-05'
date2_end = '2020-11-05'
country_wise_line_plot(path_to_data,east_asian_airports,country,date1_strt,date1_end,date2_strt,date2_end)
```



In [7]:
```
#Other-Asian Airport Plots
path_to_data = 'Data/total_data_country_2020.csv'
other_asian_airports = ["YSSY", "YMML", "OMDB", "VABB", "VIDP", "WSSS"]
country = "Asian/Australian"
date1_strt = '2020-03-10'
date1_end = '2020-06-15'
date2_strt = '2020-06-15'
date2_end = '2020-12-31'
country_wise_line_plot(path_to_data,other_asian_airports,country,date1_strt,date1_end,date2_strt,date2_end)
```



In [8]:
```
#coorelating scatter plot between covid cases and flight data
path_to_flight_data = 'Data/total_data_country_2020.csv'
path_to_covid_data = 'Data/owid-covid-data.csv'
flight_covid_correlation_plot(path_to_flight_data,path_to_covid_data)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:183: UserWarning:

Boolean Series key will be reindexed to match DataFrame index.

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:197: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py:4025: SettingWithCopyWarning:


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

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:200: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:203: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:206: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
```