Global Vaccine Covid-19 Analysis

We'll be using pandas for data manipulation and analysis, matplotlib and seaborn for data visualization, and plotly for interactive visualizations.

In [1]:
```
#Import the Required Libraries:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
import plotly.express as px
import plotly.io as pio
import numpy as np
pio.templates.default = "plotly_white"
```

In [2]:
```
#The reliable source for global COVID-19 vaccination data, such as Our World in Data;
data = pd.read_csv("owid-covid-data.csv")
```

Explore the Data

In [3]:
```
#Examining the structure and content of the dataset:
# View the first few rows
data.head ()
```

Out[3]:

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	...
0	AFG	Asia	Afghanistan	2020-01-03	NaN	0.0	NaN	NaN	0.0	NaN	...
1	AFG	Asia	Afghanistan	2020-01-04	NaN	0.0	NaN	NaN	0.0	NaN	...
2	AFG	Asia	Afghanistan	2020-01-05	NaN	0.0	NaN	NaN	0.0	NaN	...
3	AFG	Asia	Afghanistan	2020-01-06	NaN	0.0	NaN	NaN	0.0	NaN	...
4	AFG	Asia	Afghanistan	2020-01-07	NaN	0.0	NaN	NaN	0.0	NaN	...

5 rows × 67 columns

In [4]:
```
#Examining the structure and content of your dataset:
data.describe()
```

Out[4]:

	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	total_cases_per_million	new_ca
count	2.848410e+05	3.130240e+05	3.117650e+05	2.638450e+05	313099.000000	311869.000000	284841.000000	
mean	6.039621e+06	1.039078e+04	1.043132e+04	8.236171e+04	92.787703	93.143020	92086.296625	
std	3.743113e+07	1.149393e+05	9.850403e+04	4.239938e+05	639.411651	582.578465	142411.486279	
min	1.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00	0.000000	0.000000	0.000000	
25%	7.144000e+03	0.000000e+00	7.140000e-01	1.230000e+02	0.000000	0.000000	2165.393000	
50%	6.343800e+04	5.000000e+00	3.328600e+01	1.210000e+03	0.000000	0.286000	22767.189000	
75%	6.651880e+05	3.440000e+02	5.962860e+02	1.096600e+04	4.000000	6.286000	116671.211000	
max	7.675180e+08	8.401710e+06	6.402720e+06	6.947179e+06	27940.000000	14824.000000	737554.506000	

8 rows × 62 columns

In [5]:
```
#Identifying the columns and their extent of missing data
data.isnull().sum()
```

Out[5]:
```
iso_code                                  0
continent                             15276
location                                  0
date                                      0
total_cases                           37063
                                      ...
population                                0
excess_mortality_cumulative_absolute  310758
excess_mortality_cumulative           310758
excess_mortality                      310758
excess_mortality_cumulative_per_million    310758
Length: 67, dtype: int64
```

Analyze the Data

In [6]:
```
#total number of cases per continent
cases_per_continent = data.groupby('continent').sum()
#total number of cases per country
cases_per_country = data.groupby('location').sum()
```

In [7]:
```
#Extract the population data for each unique location in the data
population = data.groupby('location').tail(1)
population = population[['location','population']]
```

In [8]:
```
#The filtered population data, excluding the rows with matching 'location' values.
filter_condition = population['location'].str.contains('World|High income|Asia|Upper middle income|South America|Europe|European
population = population[filter_condition]
```

In [9]:
```
#The population data, with the largest populations appearing at the top.
population.sort_values(by='population',ascending=False)
```

Out[9]:

	location	population
57286	China	1.425887e+09
129723	India	1.417173e+09
304176	United States	3.382899e+08
130996	Indonesia	2.755013e+08
217701	Pakistan	2.358249e+08
...	...	...
94117	Falkland Islands	3.801000e+03
205627	Niue	1.952000e+03
288900	Tokelau	1.893000e+03
311814	Vatican	8.080000e+02
227885	Pitcairn	4.700000e+01

241 rows × 2 columns

In [10]:
```
#The relative magnitude of the population values, and the maximum value in the 'population' column will be highlighted.
population = population.sort_values(by='population',ascending=False).reset_index(drop=True) #sort the data first
country_population = population.style.background_gradient(cmap='gist_ncar').highlight_max('population').set_caption('Population')
```

In [11]:
```
#The gradient background and highlighted maximum value in the 'population' column.
display(country_population)
```

Population

	location	population
0	China	1425887360.000000
1	India	1417173120.000000
2	United States	338289856.000000
3	Indonesia	275501344.000000
4	Pakistan	235824864.000000
5	Nigeria	218541216.000000
6	Brazil	215313504.000000
7	Bangladesh	171186368.000000
8	Russia	144713312.000000
9	Mexico	127504120.000000
10	Japan	123951696.000000
11	Ethiopia	123379928.000000
12	Philippines	115559008.000000
13	Egypt	110990096.000000
14	Democratic Republic of Congo	99010216.000000
15	Vietnam	98186856.000000
16	Iran	88550568.000000
17	Turkey	85341248.000000
18	Germany	83369840.000000
19	Thailand	71697024.000000
20	France	67813000.000000
21	United Kingdom	67508936.000000
22	Tanzania	65497752.000000
23	Italy	59037472.000000
24	England	56550000.000000
25	Myanmar	54179312.000000
26	Kenya	54027484.000000
27	Colombia	51874028.000000
28	South Korea	51815808.000000
29	Spain	47558632.000000
30	Uganda	47249588.000000
31	Sudan	46874200.000000
32	Argentina	45510324.000000
33	Algeria	44903228.000000
34	Iraq	44496124.000000
35	Afghanistan	41128772.000000
36	Poland	39857144.000000
37	Ukraine	39701744.000000
38	Canada	38454328.000000
39	Morocco	37457976.000000
40	Saudi Arabia	36408824.000000
41	Angola	35588996.000000
42	Uzbekistan	34627648.000000
43	Peru	34049588.000000
44	Malaysia	33938216.000000
45	Yemen	33696612.000000
46	Ghana	33475870.000000
47	Mozambique	32969520.000000
48	Nepal	30547586.000000

	location	population
49	Madagascar	29611718.000000
50	Venezuela	28301700.000000
51	Cote d'Ivoire	28160548.000000
52	Cameroon	27914542.000000
53	Niger	26207982.000000
54	Australia	26177410.000000
55	North Korea	26069416.000000
56	Taiwan	23893396.000000
57	Burkina Faso	22673764.000000
58	Mali	22593598.000000
59	Syria	22125242.000000
60	Sri Lanka	21832150.000000
61	Malawi	20405318.000000
62	Zambia	20017670.000000
63	Romania	19659270.000000
64	Chile	19603736.000000
65	Kazakhstan	19397998.000000
66	Ecuador	18001002.000000
67	Guatemala	17843914.000000
68	Chad	17723312.000000
69	Somalia	17597508.000000
70	Netherlands	17564020.000000
71	Senegal	17316452.000000
72	Cambodia	16767851.000000
73	Zimbabwe	16320539.000000
74	Guinea	13859349.000000
75	Rwanda	13776702.000000
76	Benin	13352864.000000
77	Burundi	12889583.000000
78	Tunisia	12356116.000000
79	Bolivia	12224114.000000
80	Belgium	11655923.000000
81	Haiti	11585003.000000
82	Jordan	11285875.000000
83	Dominican Republic	11228821.000000
84	Cuba	11212198.000000
85	South Sudan	10913172.000000
86	Sweden	10549349.000000
87	Czechia	10493990.000000
88	Honduras	10432858.000000
89	Greece	10384972.000000
90	Azerbaijan	10358078.000000
91	Portugal	10270857.000000
92	Papua New Guinea	10142625.000000
93	Hungary	9967304.000000
94	Tajikistan	9952789.000000
95	Belarus	9534956.000000
96	Israel	9449000.000000
97	United Arab Emirates	9441138.000000
98	Austria	8939617.000000

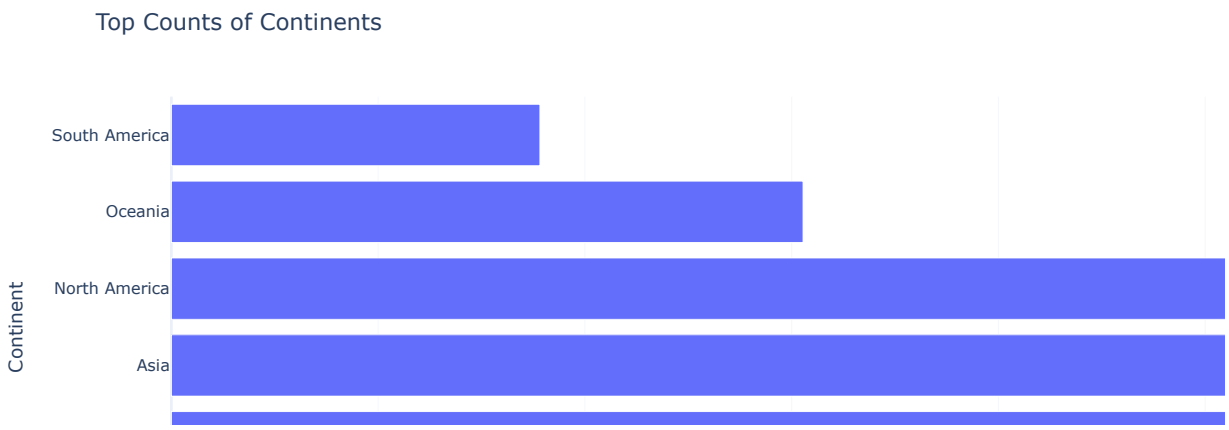
	location	population
99	Togo	8848700.000000
100	Switzerland	8740471.000000
101	Sierra Leone	8605723.000000
102	Laos	7529477.000000
103	Hong Kong	7488863.000000
104	Nicaragua	6948395.000000
105	Serbia	6871547.000000
106	Libya	6812344.000000
107	Bulgaria	6781955.000000
108	Paraguay	6780745.000000
109	Kyrgyzstan	6630621.000000
110	Turkmenistan	6430777.000000
111	El Salvador	6336393.000000
112	Congo	5970430.000000
113	Denmark	5882259.000000
114	Slovakia	5643455.000000
115	Singapore	5637022.000000
116	Finland	5540745.000000
117	Lebanon	5489744.000000
118	Scotland	5466000.000000
119	Norway	5434324.000000
120	Liberia	5302690.000000
121	Palestine	5250076.000000
122	New Zealand	5185289.000000
123	Costa Rica	5180836.000000
124	Ireland	5023108.000000
125	Mauritania	4736146.000000
126	Oman	4576300.000000
127	Panama	4408582.000000
128	Kuwait	4268886.000000
129	Croatia	4030361.000000
130	Georgia	3744385.000000
131	Eritrea	3684041.000000
132	Uruguay	3422796.000000
133	Mongolia	3398373.000000
134	Moldova	3272993.000000
135	Puerto Rico	3252412.000000
136	Bosnia and Herzegovina	3233530.000000
137	Wales	3170000.000000
138	Albania	2842318.000000
139	Jamaica	2827382.000000
140	Armenia	2780472.000000
141	Lithuania	2750058.000000
142	Gambia	2705995.000000
143	Qatar	2695131.000000
144	Botswana	2630300.000000
145	Namibia	2567024.000000
146	Gabon	2388997.000000
147	Lesotho	2305826.000000
148	Slovenia	2119843.000000

	location	population
149	Guinea-Bissau	2105580.000000
150	North Macedonia	2093606.000000
151	Northern Ireland	1896000.000000
152	Latvia	1850654.000000
153	Kosovo	1782115.000000
154	Equatorial Guinea	1674916.000000
155	Trinidad and Tobago	1531043.000000
156	Bahrain	1472237.000000
157	Timor	1341298.000000
158	Estonia	1326064.000000
159	Mauritius	1299478.000000
160	Eswatini	1201680.000000
161	Djibouti	1120851.000000
162	Reunion	974062.000000
163	Fiji	929769.000000
164	Cyprus	896007.000000
165	Comoros	836783.000000
166	Guyana	808727.000000
167	Bhutan	782457.000000
168	Solomon Islands	724272.000000
169	Macao	695180.000000
170	Luxembourg	647601.000000
171	Montenegro	627082.000000
172	Suriname	618046.000000
173	Cape Verde	593162.000000
174	Western Sahara	576005.000000
175	Malta	533293.000000
176	Maldives	523798.000000
177	Brunei	449002.000000
178	Bahamas	409989.000000
179	Belize	405285.000000
180	Guadeloupe	395762.000000
181	Northern Cyprus	382836.000000
182	Iceland	372903.000000
183	Martinique	367512.000000
184	Vanuatu	326744.000000
185	Mayotte	326113.000000
186	French Polynesia	306292.000000
187	French Guiana	304568.000000
188	New Caledonia	289959.000000
189	Barbados	281646.000000
190	Sao Tome and Principe	227393.000000
191	Samoa	222390.000000
192	Curacao	191173.000000
193	Saint Lucia	179872.000000
194	Guam	171783.000000
195	Kiribati	131237.000000
196	Grenada	125459.000000
197	Micronesia (country)	114178.000000
198	Jersey	110796.000000

	location	population
199	Seychelles	107135.000000
200	Tonga	106867.000000
201	Aruba	106459.000000
202	Saint Vincent and the Grenadines	103959.000000
203	United States Virgin Islands	99479.000000
204	Antigua and Barbuda	93772.000000
205	Isle of Man	84534.000000
206	Andorra	79843.000000
207	Dominica	72758.000000
208	Cayman Islands	68722.000000
209	Bermuda	64207.000000
210	Guernsey	63329.000000
211	Greenland	56494.000000
212	Faeroe Islands	53117.000000
213	Northern Mariana Islands	49574.000000
214	Saint Kitts and Nevis	47681.000000
215	Turks and Caicos Islands	45726.000000
216	American Samoa	44295.000000
217	Sint Maarten (Dutch part)	44192.000000
218	Marshall Islands	41593.000000
219	Liechtenstein	39355.000000
220	Monaco	36491.000000
221	San Marino	33690.000000
222	Gibraltar	32677.000000
223	Saint Martin (French part)	31816.000000
224	British Virgin Islands	31332.000000
225	Bonaire Sint Eustatius and Saba	27052.000000
226	Palau	18084.000000
227	Cook Islands	17032.000000
228	Anguilla	15877.000000
229	Nauru	12691.000000
230	Wallis and Futuna	11596.000000
231	Tuvalu	11335.000000
232	Saint Barthelemy	10994.000000
233	Saint Pierre and Miquelon	5885.000000
234	Saint Helena	5401.000000
235	Montserrat	4413.000000
236	Falkland Islands	3801.000000
237	Niue	1952.000000
238	Tokelau	1893.000000
239	Vatican	808.000000
240	Pitcairn	47.000000

In [12]:
```
# Calculate the value counts
continent_counts = data['continent'].value_counts()[:20]

# Create the bar chart trace
bar_chart = go.Bar(
    x=continent_counts.values,
    y=continent_counts.index,
    orientation='h'
)

# Create the figure
fig = go.Figure(data=[bar_chart])
```

```
# Set layout options
fig.update_layout(
    title='Top Counts of Continents',
    xaxis=dict(title='Count'),
    yaxis=dict(title='Continent'),
)

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



In [13]:
```
#The top unique values in the 'continent' column of the data
data['continent'].value_counts()
```

Out[13]:
```
continent
Africa           72562
Europe           69686
Asia             63809
North America    52195
Oceania          30552
South America    17824
Name: count, dtype: int64
```

In [14]:
```
#The data specific to India within dataset:
data[data['location']=='India']
```

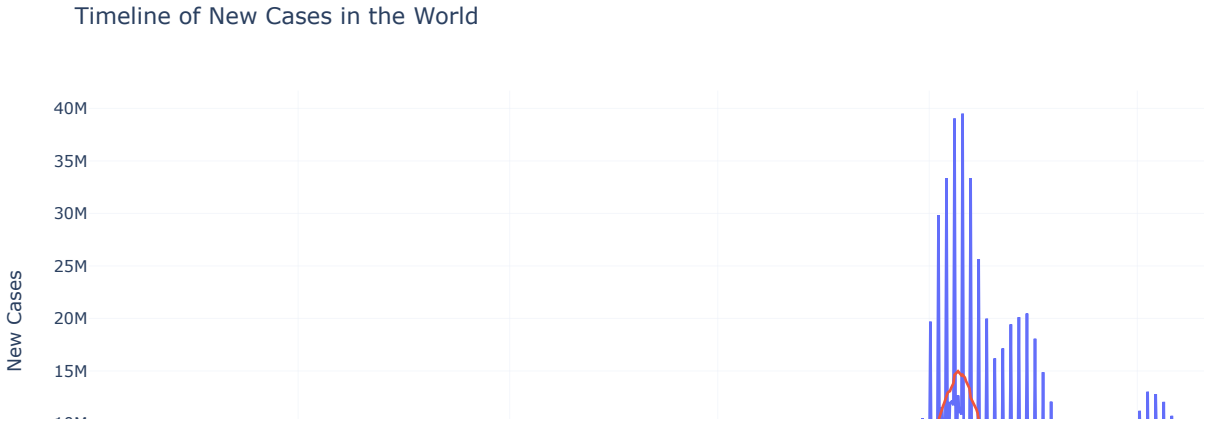
Out[14]:

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed
128451	IND	Asia	India	2020-01-03	NaN	0.0	NaN	NaN	0.0	NaN
128452	IND	Asia	India	2020-01-04	NaN	0.0	NaN	NaN	0.0	NaN
128453	IND	Asia	India	2020-01-05	NaN	0.0	NaN	NaN	0.0	NaN
128454	IND	Asia	India	2020-01-06	NaN	0.0	NaN	NaN	0.0	NaN
128455	IND	Asia	India	2020-01-07	NaN	0.0	NaN	NaN	0.0	NaN
...	...	...	...	...	...	...	...	...	...	...
129719	IND	Asia	India	2023-06-24	44993872.0	55.0	68.857	531903.0	1.0	1.429
129720	IND	Asia	India	2023-06-25	44993952.0	80.0	67.429	531903.0	0.0	1.143
129721	IND	Asia	India	2023-06-26	44993999.0	47.0	65.143	531903.0	0.0	1.000
129722	IND	Asia	India	2023-06-27	44993999.0	0.0	60.000	531903.0	0.0	0.857
129723	IND	Asia	India	2023-06-28	44994097.0	98.0	60.857	531905.0	2.0	1.000

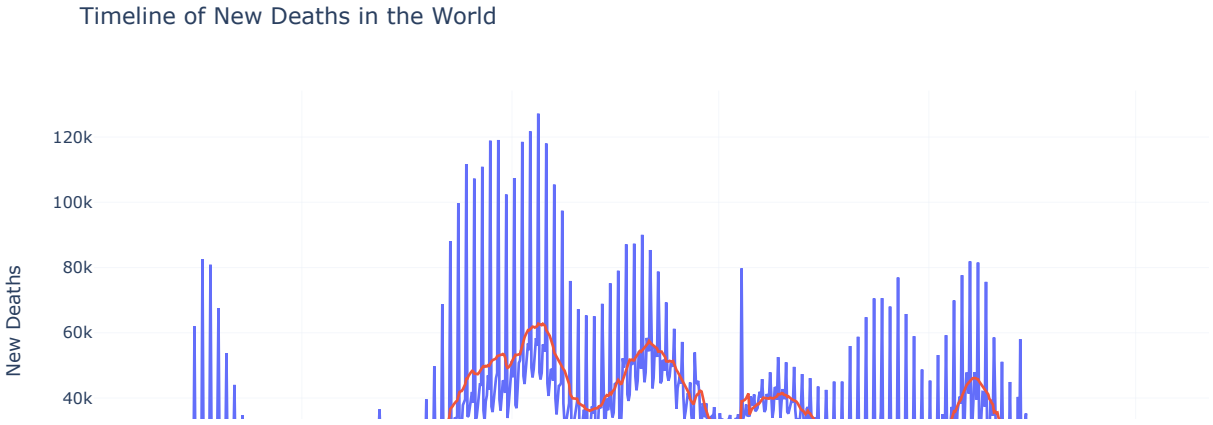
1273 rows × 67 columns

In [15]:
```
#Moving Average
data2=data.copy()
data2 = data.copy()
data2.date = pd.to_datetime(data2['date'])
data2 = data2.groupby('date').sum()
data2['7 days MA new cases'] = 0
data2['7 days MA new cases'] = data2['new_cases'].rolling(7).mean() #Moving average of new cases
data2['7 days MA new deaths'] = 0
data2['7 days MA new deaths'] = data2['new_deaths'].rolling(7).mean()  #Moving average of new deaths
```

In [16]:
```
# Create traces for 'new_cases' and '7 days MA new cases' timeline:
trace_new_cases = go.Scatter(
    x=data2.index,
    y=data2['new_cases'],
    mode='lines',
    name='New Cases'
)

trace_ma_new_cases = go.Scatter(
    x=data2.index,
    y=data2['7 days MA new cases'],
    mode='lines',
    name='7 days MA New Cases'
)

# Create the figure
fig = go.Figure(data=[trace_new_cases, trace_ma_new_cases])

# Set layout options
fig.update_layout(
    title='Timeline of New Cases in the World',
    xaxis=dict(title='Date'),
    yaxis=dict(title='New Cases'),
)

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



In [17]:
```
# Calculate the moving average for new deaths
data2['7 days MA new deaths'] = data2['new_deaths'].rolling(7).mean()

# Create traces for 'new_deaths' and '7 days MA new deaths'
trace_new_deaths = go.Scatter(
    x=data2.index,
    y=data2['new_deaths'],
    mode='lines',
    name='New Deaths'
)

trace_ma_new_deaths = go.Scatter(
    x=data2.index,
    y=data2['7 days MA new deaths'],
    mode='lines',
    name='7 days MA New Deaths'
)

# Create the figure
fig = go.Figure(data=[trace_new_deaths, trace_ma_new_deaths])

# Set layout options
fig.update_layout(
    title='Timeline of New Deaths in the World',
    xaxis=dict(title='Date'),
    yaxis=dict(title='New Deaths'),
)

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

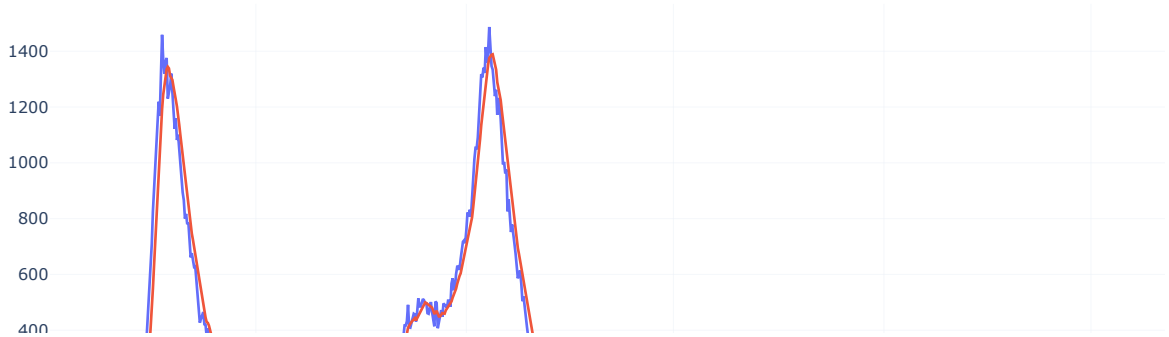


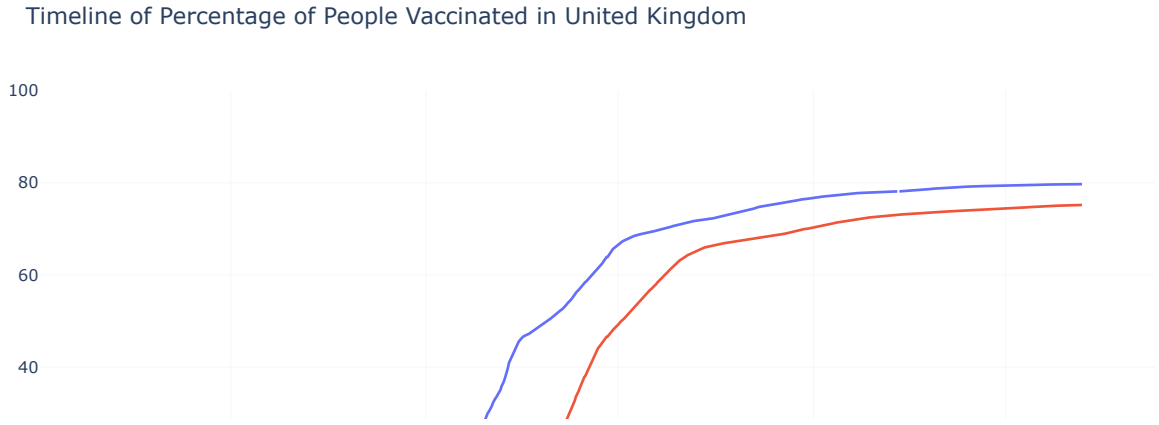
In [18]:
```
import plotly.graph_objects as go

def create_and_plot_df(df, country):
    # Selecting the 7 key columns for the country in the dataset
    df = df[df['location'] == country].copy()
    df = df[['date', 'total_cases', 'new_cases', 'total_deaths', 'new_deaths',
             'people_vaccinated_per_hundred', 'people_fully_vaccinated_per_hundred',
             'new_deaths_per_million', 'new_cases_per_million']].copy()

    # Convert to datetime
    df.date = pd.to_datetime(df['date'])

    # Set the date as the index and compute the moving average with window=7 for new_cases and new_deaths
    df.set_index('date', inplace=True)
    df['7 days MA new cases'] = df['new_cases'].rolling(7).mean()
    df['7 days MA new deaths'] = df['new_deaths'].rolling(7).mean()
    df['7 days MA new cases per million'] = df['new_cases_per_million'].rolling(7).mean()
    df['7 days MA new deaths per million'] = df['new_deaths_per_million'].rolling(7).mean()

    # Plot new cases, new deaths, and people vaccinated
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=df.index, y=df['new_cases'], mode='lines', name='New Cases'))
    fig.add_trace(go.Scatter(x=df.index, y=df['7 days MA new cases'], mode='lines', name='7 days MA New Cases'))
    fig.update_layout(title=f'Timeline of New Cases in {country}')
    fig.show()

    fig = go.Figure()
    fig.add_trace(go.Scatter(x=df.index, y=df['new_deaths'], mode='lines', name='New Deaths'))
    fig.add_trace(go.Scatter(x=df.index, y=df['7 days MA new deaths'], mode='lines', name='7 days MA New Deaths'))
    if country == 'Chile':
        fig.update_layout(title=f'Timeline of New Deaths in {country}', yaxis_range=[0, 400])
    else:
        fig.update_layout(title=f'Timeline of New Deaths in {country}')
    fig.show()

    fig = go.Figure()
    fig.add_trace(go.Scatter(x=df.index, y=df['people_vaccinated_per_hundred'], mode='lines', name='People Vaccinated per Hundred
    fig.add_trace(go.Scatter(x=df.index, y=df['people_fully_vaccinated_per_hundred'], mode='lines', name='People Fully Vaccinate
    fig.update_layout(title=f'Timeline of Percentage of People Vaccinated in {country}', yaxis_range=[0, 100])
    fig.show()

    return df
```

In [19]:
```
df_uk = create_and_plot_df(data, 'United Kingdom')
```

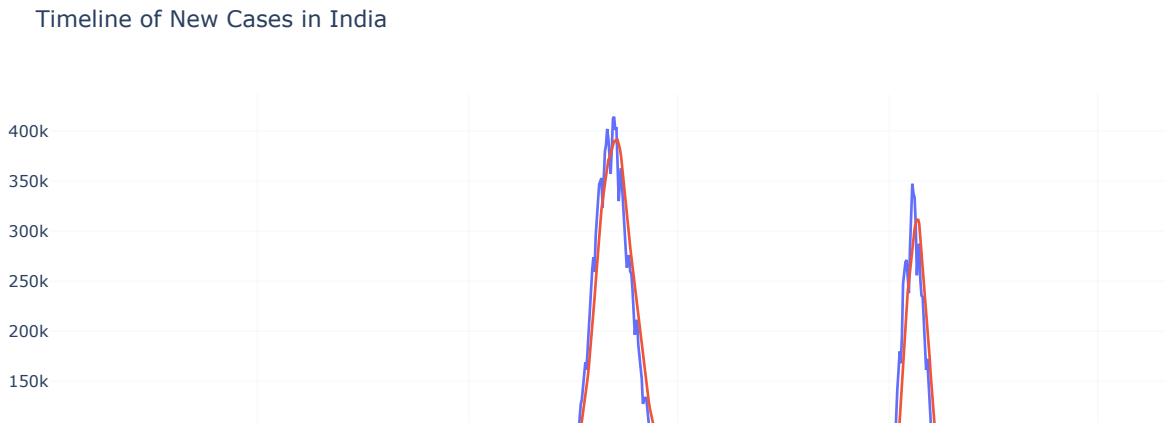
Timeline of New Cases in United Kingdom

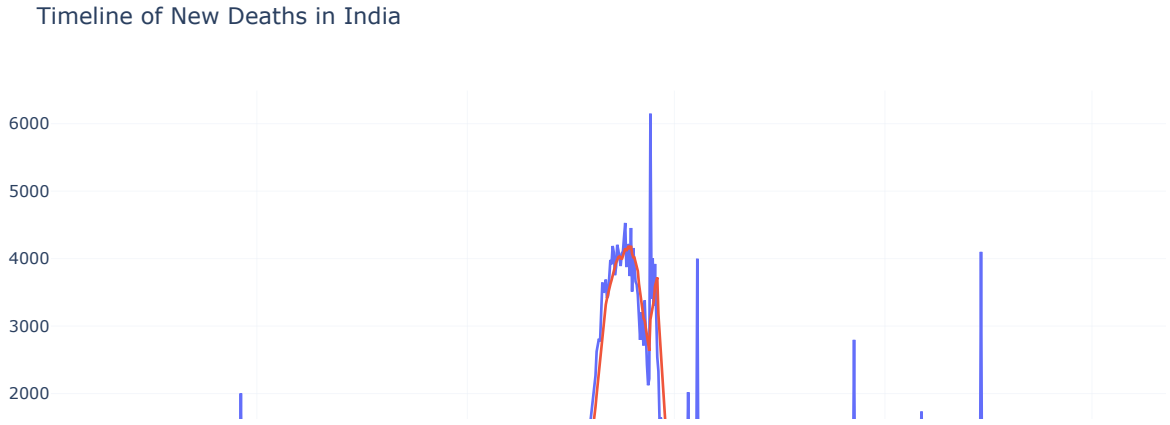
Timeline of New Deaths in United Kingdom

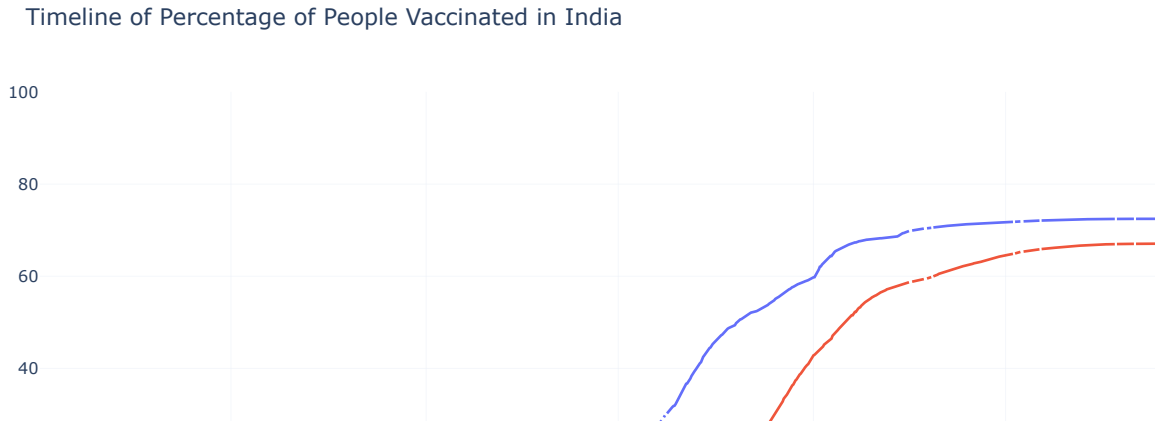




In [20]:
```
df_india=create_and_plot_df(data,'India')
```



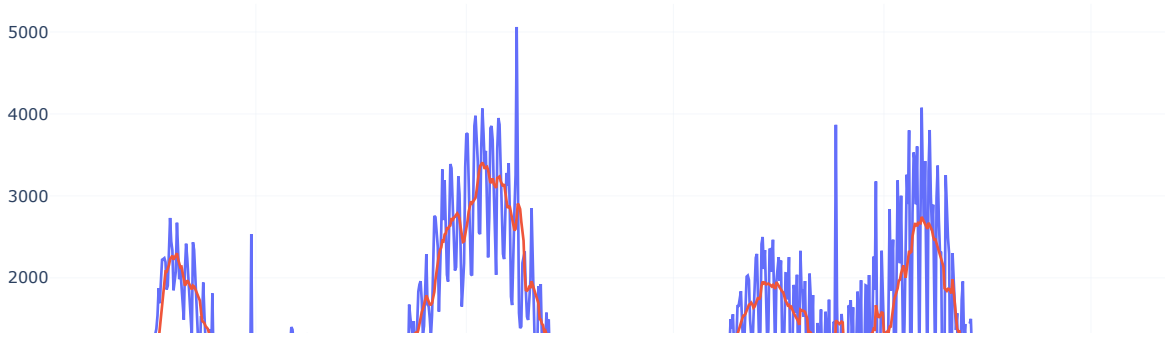


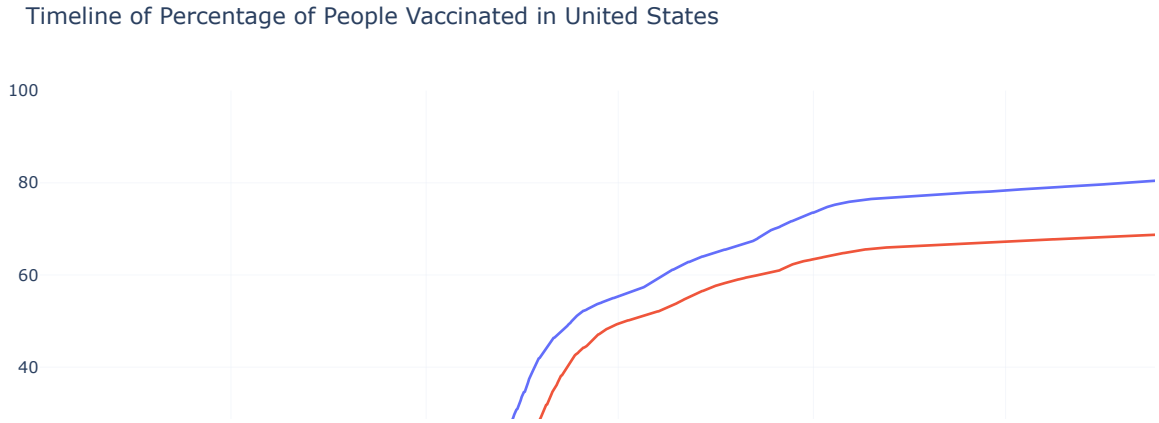


In [21]:
```
df_us=create_and_plot_df(data,'United States')
```

Timeline of New Cases in United States

Timeline of New Deaths in United States





In [22]:
```
df_india
```

Out[22]:

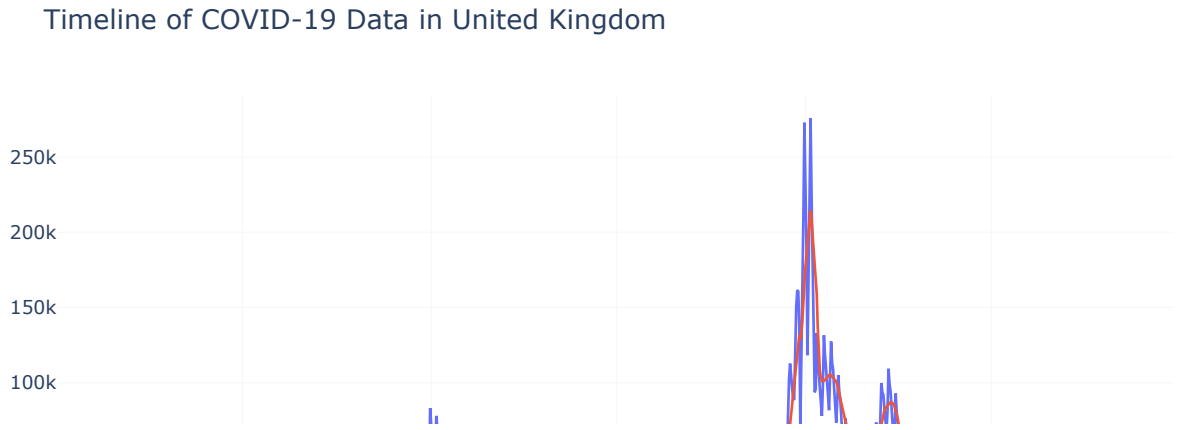
date	total_cases	new_cases	total_deaths	new_deaths	people_vaccinated_per_hundred	people_fully_vaccinated_per_hundred	new_deaths_pe
2020-01-03	NaN	0.0	NaN	0.0	NaN	NaN	
2020-01-04	NaN	0.0	NaN	0.0	NaN	NaN	
2020-01-05	NaN	0.0	NaN	0.0	NaN	NaN	
2020-01-06	NaN	0.0	NaN	0.0	NaN	NaN	
2020-01-07	NaN	0.0	NaN	0.0	NaN	NaN	
...	...	...	...	...	...	...	
2023-06-24	44993872.0	55.0	531903.0	1.0	72.5	67.17	
2023-06-25	44993952.0	80.0	531903.0	0.0	72.5	67.17	
2023-06-26	44993999.0	47.0	531903.0	0.0	72.5	67.17	
2023-06-27	44993999.0	0.0	531903.0	0.0	72.5	67.17	
2023-06-28	44994097.0	98.0	531905.0	2.0	72.5	67.17	

1273 rows × 12 columns

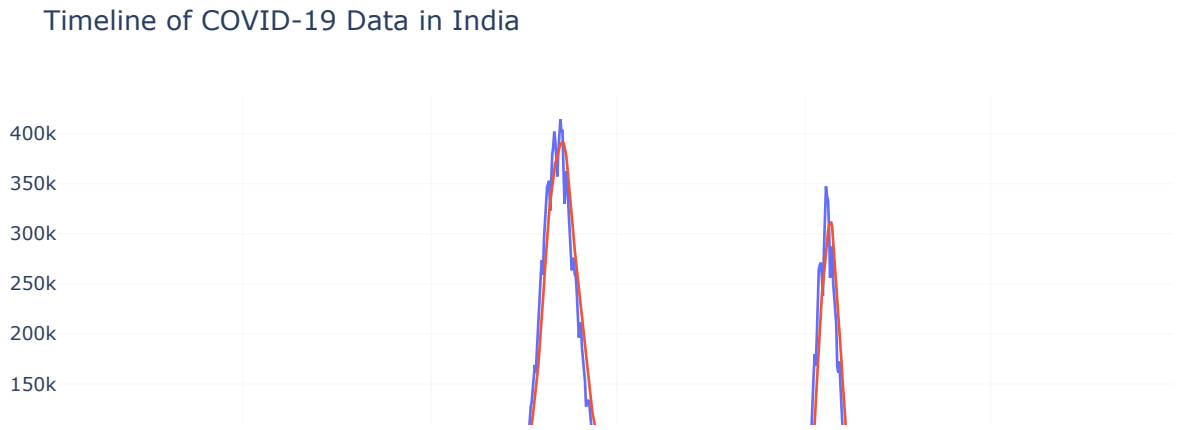
In [23]:
```
import plotly.graph_objects as go

def create_and_plot_df(df, country):
    # Selecting the 7 key columns for country in dataset
    df = df[df['location'] == country].copy()
    df = df[['date', 'total_cases', 'new_cases', 'total_deaths', 'new_deaths', 'people_vaccinated_per_hundred',
             'people_fully_vaccinated_per_hundred', 'new_deaths_per_million', 'new_cases_per_million']].copy()

    # Convert to datetime
    df.date = pd.to_datetime(df['date'])

    # Set the date as index and compute moving average with window=7 for new_cases and new_deaths
    df.set_index('date', inplace=True)
    df['7 days MA new cases'] = df['new_cases'].rolling(7).mean()
```

```
    df['7 days MA new deaths'] = df['new_deaths'].rolling(7).mean()
    df['7 days MA new cases per million'] = df['new_cases_per_million'].rolling(7).mean()
    df['7 days MA new deaths per million'] = df['new_deaths_per_million'].rolling(7).mean()

    # Create Plotly figure
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=df.index, y=df['new_cases'], name='New Cases'))
    fig.add_trace(go.Scatter(x=df.index, y=df['7 days MA new cases'], name='7 Days Moving Average New Cases'))
    fig.add_trace(go.Scatter(x=df.index, y=df['new_deaths'], name='New Deaths'))
    fig.add_trace(go.Scatter(x=df.index, y=df['7 days MA new deaths'], name='7 Days Moving Average New Deaths'))

    fig.update_layout(
        title=f"Timeline of COVID-19 Data in {country}",
        xaxis_title="Date",
        font=dict(size=14),
    )

    fig.show()

    return df
```

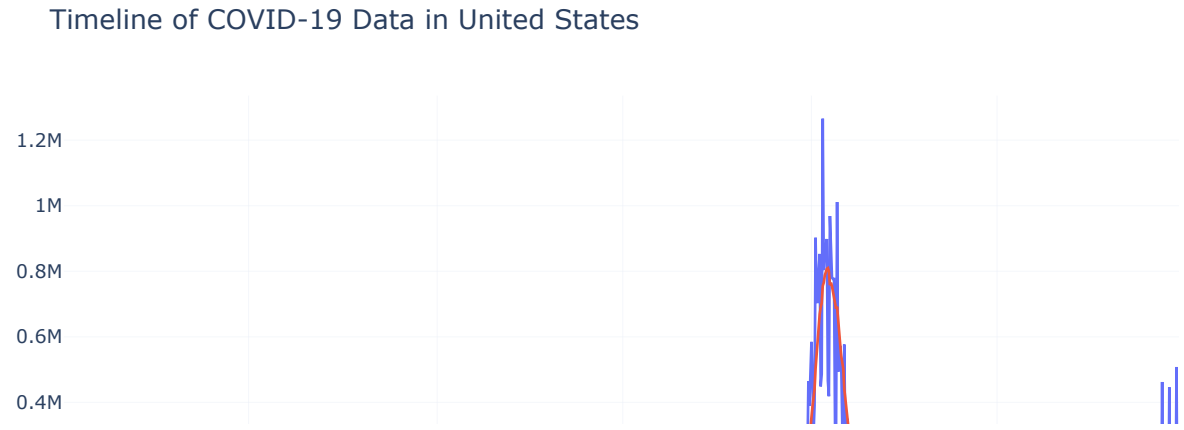
In [24]:
```
df_uk=create_and_plot_df(data,'United Kingdom')
```



In [25]:
```
df_india=create_and_plot_df(data,'India')
```



In [26]:
```
df_us=create_and_plot_df(data,'United States')
```



In [27]:
```
df_india
```

Out[27]:

date	total_cases	new_cases	total_deaths	new_deaths	people_vaccinated_per_hundred	people_fully_vaccinated_per_hundred	new_deaths_per
2020-01-03	NaN	0.0	NaN	0.0	NaN	NaN	
2020-01-04	NaN	0.0	NaN	0.0	NaN	NaN	
2020-01-05	NaN	0.0	NaN	0.0	NaN	NaN	
2020-01-06	NaN	0.0	NaN	0.0	NaN	NaN	
2020-01-07	NaN	0.0	NaN	0.0	NaN	NaN	
...	...	...	...	...	...	...	
2023-06-24	44993872.0	55.0	531903.0	1.0	72.5	67.17	
2023-06-25	44993952.0	80.0	531903.0	0.0	72.5	67.17	
2023-06-26	44993999.0	47.0	531903.0	0.0	72.5	67.17	
2023-06-27	44993999.0	0.0	531903.0	0.0	72.5	67.17	
2023-06-28	44994097.0	98.0	531905.0	2.0	72.5	67.17	

1273 rows × 12 columns

COMPARISON OF THREE COUNTRIES

In [28]:
```
#The data for the specified countries and metrics.
df3_countries=pd.DataFrame()
df3_countries['New cases per million uk'] = df_uk['7 days MA new cases per million']
df3_countries['New cases per million us'] = df_us['7 days MA new cases per million']
df3_countries['New cases per million india'] = df_india['7 days MA new cases per million']
df3_countries['New deaths per million uk'] = df_uk['7 days MA new deaths per million']
df3_countries['New deaths per million us'] = df_us['7 days MA new deaths per million']
df3_countries['New deaths per million india'] = df_india['7 days MA new deaths per million']
df3_countries['date'] = df_uk.index
df3_countries.set_index('date', inplace=True)
```

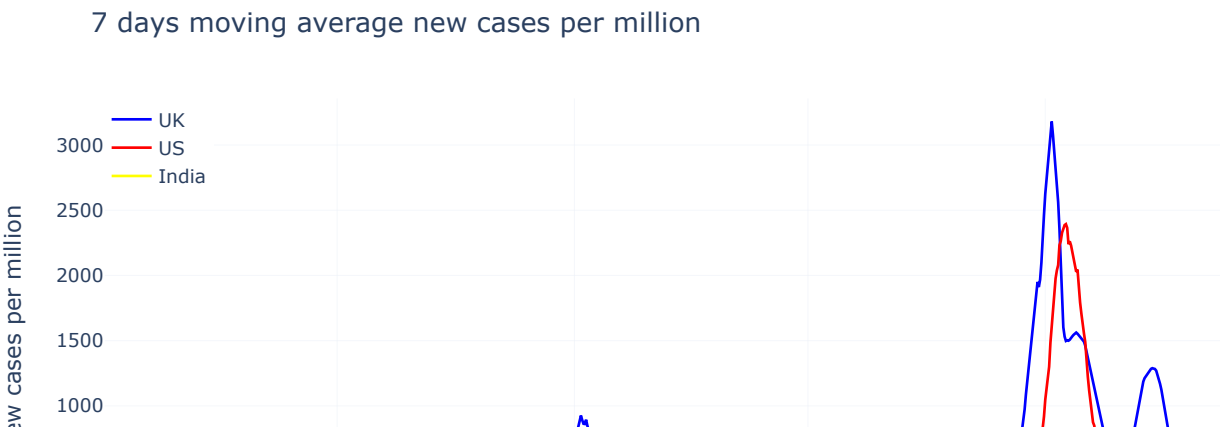
```
df3_countries.head()
```

Out[28]:

date	New cases per million uk	New cases per million us	New cases per million india	New deaths per million uk	New deaths per million us	New deaths per million india
2020-01-03	NaN	NaN	NaN	NaN	NaN	NaN
2020-01-04	NaN	NaN	NaN	NaN	NaN	NaN
2020-01-05	NaN	NaN	NaN	NaN	NaN	NaN
2020-01-06	NaN	NaN	NaN	NaN	NaN	NaN
2020-01-07	NaN	NaN	NaN	NaN	NaN	NaN

In [29]:

```
fig = go.Figure()

fig.add_trace(go.Scatter(x=df3_countries.index, y=df3_countries['New cases per million uk'], name='UK', line=dict(color='blue')))
fig.add_trace(go.Scatter(x=df3_countries.index, y=df3_countries['New cases per million us'], name='US', line=dict(color='red')))
fig.add_trace(go.Scatter(x=df3_countries.index, y=df3_countries['New cases per million india'], name='India', line=dict(color='ye

fig.update_layout(
    title="7 days moving average new cases per million",
    xaxis_title="Date",
    yaxis_title="New cases per million",
    legend=dict(x=0, y=1),
    font=dict(size=14),
)

fig.show()
```



In [30]:

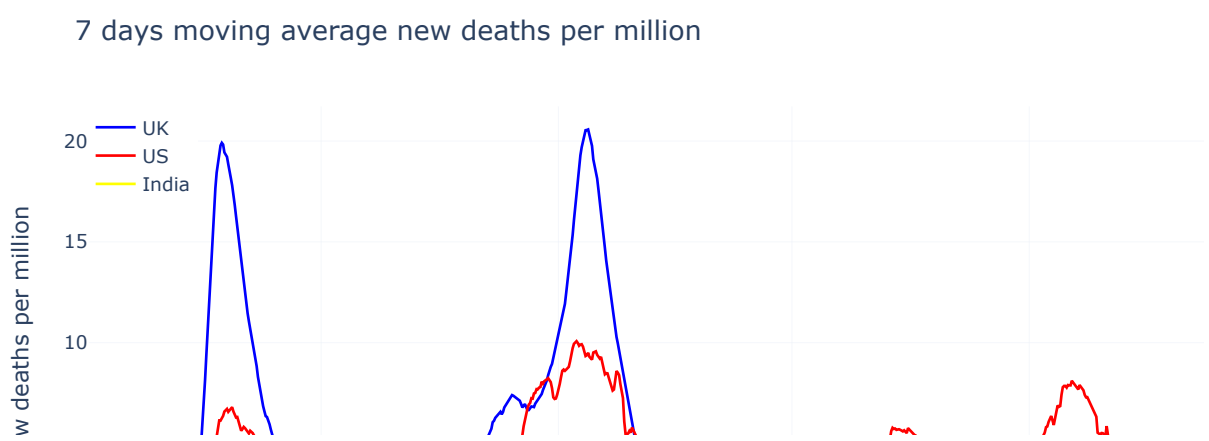
```
print('AUC New cases per million in the UK: ', df3_countries['New cases per million uk'].sum())
print('AUC New cases per million in the US: ', df3_countries['New cases per million us'].sum())
print('AUC New cases per million in India: ', df3_countries['New cases per million india'].sum())
```

```
AUC New cases per million in the UK:  364923.728
AUC New cases per million in the US:  305763.90599999996
AUC New cases per million in India:  31746.789000000004
```

In [31]:

```
fig = go.Figure()

fig.add_trace(go.Scatter(x=df3_countries.index, y=df3_countries['New deaths per million uk'], name='UK', line=dict(color='blue')))
fig.add_trace(go.Scatter(x=df3_countries.index, y=df3_countries['New deaths per million us'], name='US', line=dict(color='red')))
fig.add_trace(go.Scatter(x=df3_countries.index, y=df3_countries['New deaths per million india'], name='India', line=dict(color='y

fig.update_layout(
    title="7 days moving average new deaths per million",
    xaxis_title="Date",
    yaxis_title="New deaths per million",
    legend=dict(x=0, y=1),
    font=dict(size=14),
```

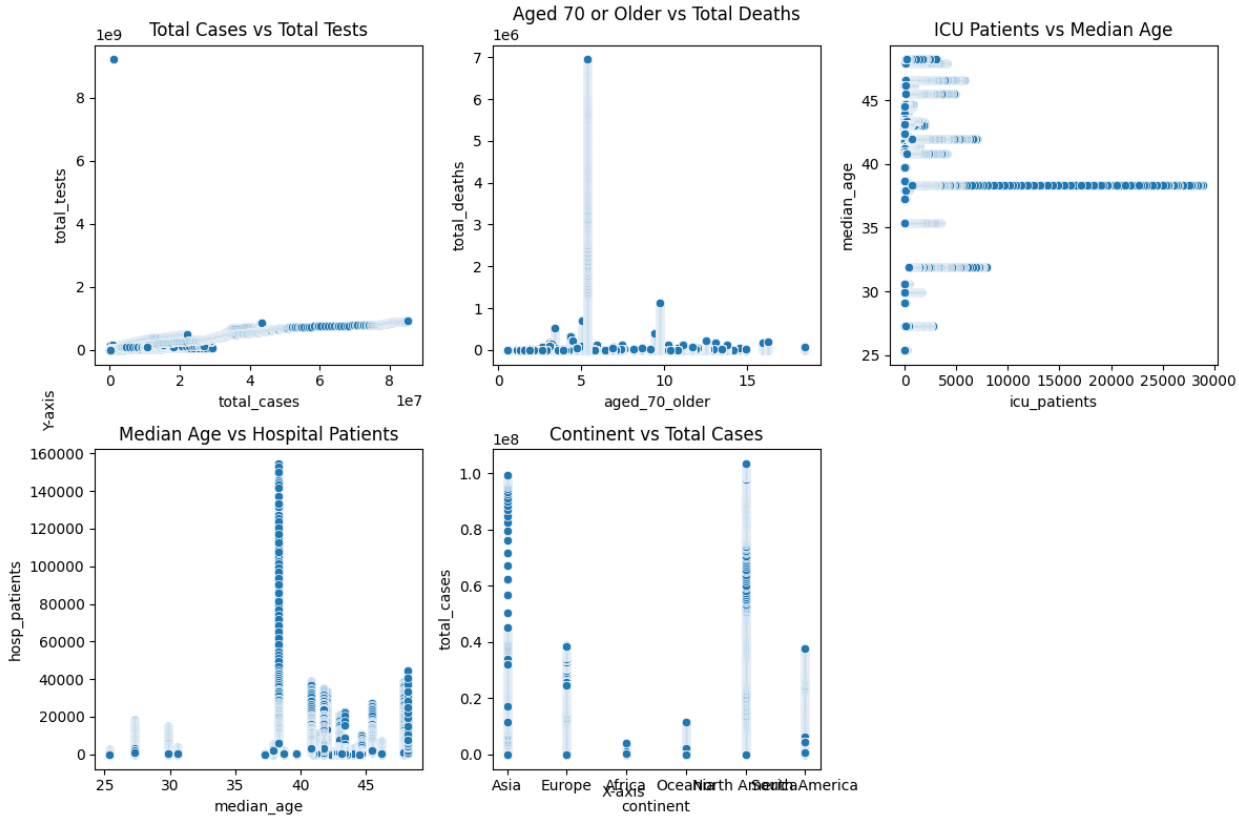
```
)

fig.show()
```



In [32]:
```
print('AUC New deaths per million in the UK: ', df3_countries['New deaths per million uk'].sum())
print('AUC New deaths per million in the US: ', df3_countries['New deaths per million us'].sum())
print('AUC New deaths per million in India: ', df3_countries['New deaths per million india'].sum())
```

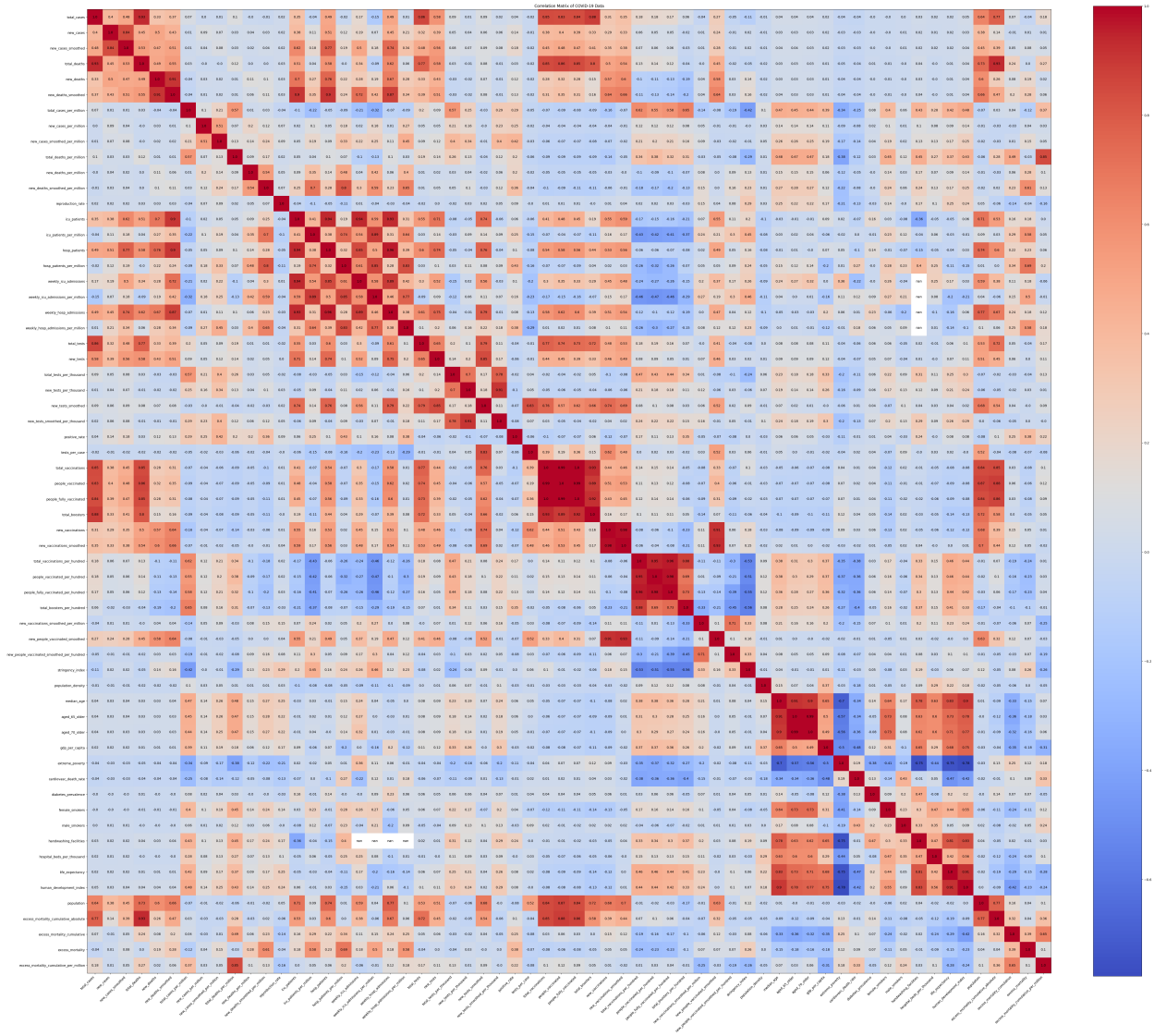
```
AUC New deaths per million in the UK:  3370.279
AUC New deaths per million in the US:  3321.068142857143
AUC New deaths per million in India:  375.3207142857143
```

In [37]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Create the figure and subplots
fig, axs = plt.subplots(2, 3, figsize=(12, 8))

# Scatter plot for 'total_cases' vs 'total_tests'
sns.scatterplot(x='total_cases', y='total_tests', data=data, ax=axs[0, 0])

# Scatter plot for 'aged_70_older' vs 'total_deaths'
sns.scatterplot(x='aged_70_older', y='total_deaths', data=data, ax=axs[0, 1])

# Scatter plot for 'icu_patients' vs 'median_age'
sns.scatterplot(x='icu_patients', y='median_age', data=data, ax=axs[0, 2])

# Scatter plot for 'median_age' vs 'hosp_patients'
sns.scatterplot(x='median_age', y='hosp_patients', data=data, ax=axs[1, 0])

# Scatter plot for 'continent' vs 'total_cases'
sns.scatterplot(x='continent', y='total_cases', data=data, ax=axs[1, 1])

# Remove the empty subplot
fig.delaxes(axs[1, 2])

# Set titles for each subplot
axs[0, 0].set_title('Total Cases vs Total Tests')
axs[0, 1].set_title('Aged 70 or Older vs Total Deaths')
axs[0, 2].set_title('ICU Patients vs Median Age')
axs[1, 0].set_title('Median Age vs Hospital Patients')
axs[1, 1].set_title('Continent vs Total Cases')

# Set common xlabel and ylabel
fig.text(0.5, 0.04, 'X-axis', ha='center')
fig.text(0.04, 0.5, 'Y-axis', va='center', rotation='vertical')

# Adjust the spacing between subplots
plt.tight_layout()

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



Correleration Matrix

In [34]:
```
# Select only numeric columns
numeric_columns = data.select_dtypes(include=np.number)

# Create a correlation matrix
corr = numeric_columns.corr()

# Initialise figure
fig, ax = plt.subplots(figsize=(50, 50))

# Create heatmap
im = ax.imshow(corr.values, cmap='coolwarm')

# Set labels
ax.set_xticks(np.arange(len(corr.columns)))
ax.set_yticks(np.arange(len(corr.columns)))
ax.set_xticklabels(corr.columns, rotation=45, ha="right")
ax.set_yticklabels(corr.columns)

# Loop over data dimensions and create text annotations
for i in range(len(corr.columns)):
    for j in range(len(corr.columns)):
        text = ax.text(j, i, np.around(corr.iloc[i, j], decimals=2),
                       ha="center", va="center", color="black")

# Set colorbar
cbar = ax.figure.colorbar(im, ax=ax, fraction=0.046, pad=0.04)
cbar.ax.set_ylabel("Correlation", rotation=-90, va="bottom")

# Set title
ax.set_title("Correlation Matrix of COVID-19 Data")

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



In [35]:
```
# Select only numeric columns
numeric_columns = data.select_dtypes(include=np.number)

# Create a correlation matrix
corr = numeric_columns.corr()

# Define the heatmap trace
heatmap = go.Heatmap(
    z=corr.values,
    x=corr.columns,
    y=corr.columns,
    colorscale='RdBu',
    zmin=-1,
    zmax=1,
    colorbar=dict(title='Correlation')
)

# Create the figure
fig = go.Figure(data=heatmap)

# Set layout options
fig.update_layout(
    title='Correlation Matrix of COVID-19 Data',
    width=1200,
    height=1500,
    xaxis=dict(title='Columns'),
    yaxis=dict(title='Columns')
)

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


Correlation Matrix of COVID-19 Data
excess_mortality_cumulative_per_million
excess_mortality
excess_mortality_cumulative
excess_mortality_cumulative_absolute
population
human_development_index
life_expectancy
hospital_beds_per_thousand
handwashing_facilities
male_smokers
female_smokers
diabetes_prevalence
cardiovasc_death_rate
extreme_poverty
gdp_per_capita
aged_70_older
aged_65_older
median_age
population_density
stringency_index
new_people_vaccinated_smoothed_per_hundred
new_people_vaccinated_smoothed
new_vaccinations_smoothed_per_million
total_boosters_per_hundred
people_fully_vaccinated_per_hundred
people_vaccinated_per_hundred
total_vaccinations_per_hundred
new_vaccinations_smoothed
new_vaccinations
total_boosters
people_fully_vaccinated
people_vaccinated
total_vaccinations
tests_per_case
positive_rate
new_tests_smoothed_per_thousand
new_tests_smoothed
new_tests_per_thousand
total_tests_per_thousand
new_tests
total_tests
weekly_hosp_admissions_per_million
weekly_hosp_admissions
weekly_icu_admissions_per_million
weekly_icu_admissions
hosp_patients_per_million
hosp_patients
icu_patients_per_million
icu_patients
reproduction_rate
new_deaths_smoothed_per_million
new_deaths_per_million
total_deaths_per_million
new_cases_smoothed_per_million
new_cases_per_million
total_cases_per_million
new_deaths_smoothed
new_deaths
total_deaths
new_cases_smoothed
new_cases
total_cases
Columns
total_cases
new_cases_smoothed
new_deaths
total_cases_per_million
new_cases_smoothed_per_million
new_deaths_per_million
reproduction_rate
icu_patients_per_million
hosp_patients_per_million
weekly_icu_admissions_per_million
weekly_hosp_admissions_per_million
new_tests
new_tests_per_thousand
new_tests_smoothed_per_thousand
tests_per_case
people_vaccinated
total_boosters
new_vaccinations_smoothed
people_vaccinated_per_hundred
total_boosters_per_hundred
new_people_vaccinated_smoothed
stringency_index
median_age
aged_70_older
extreme_poverty
Columns