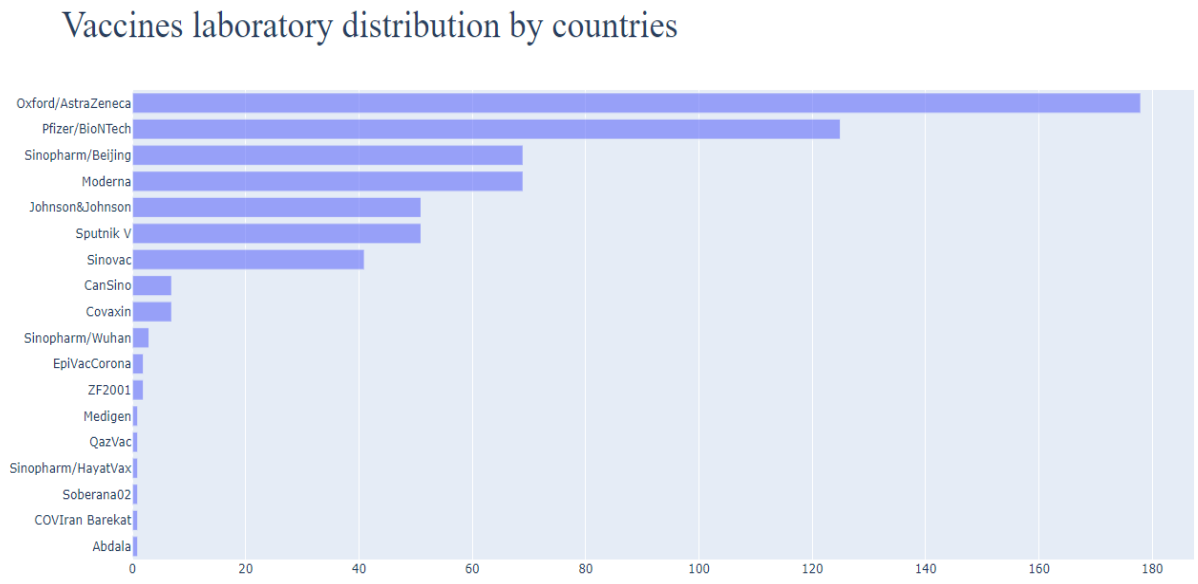
To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

This summary reports on the progress of COVID-19 vaccination, factors that influence vaccination & sentiments of public about vaccines.

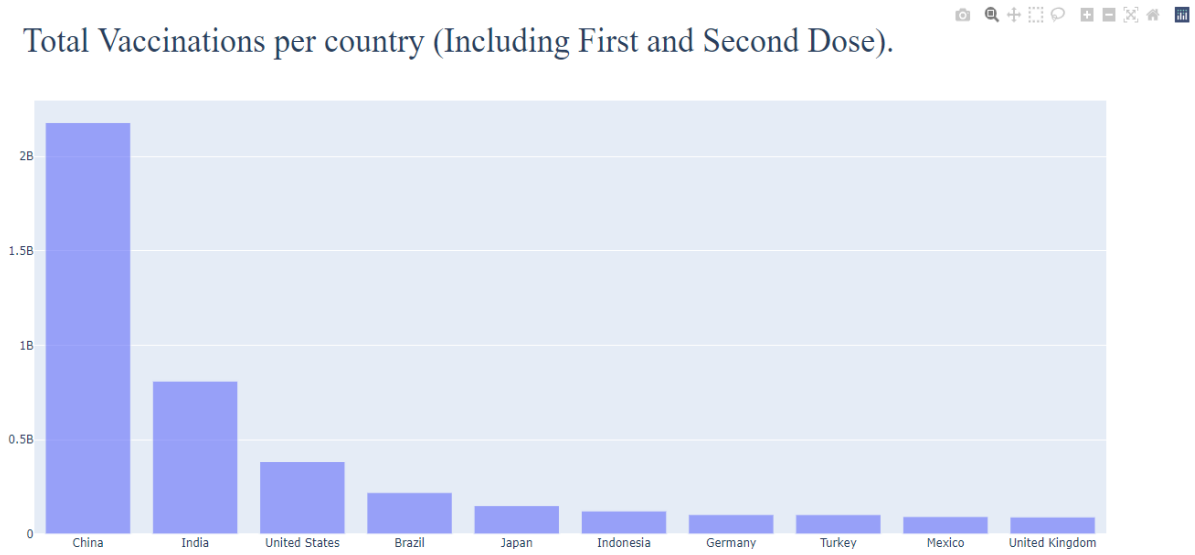
EXECUTIVE SUMMARY

Major Findings:

- According to the data and graphs, we can conclude that oxford/ AstraZeneca had manufactured and distributed the most number of vaccines.

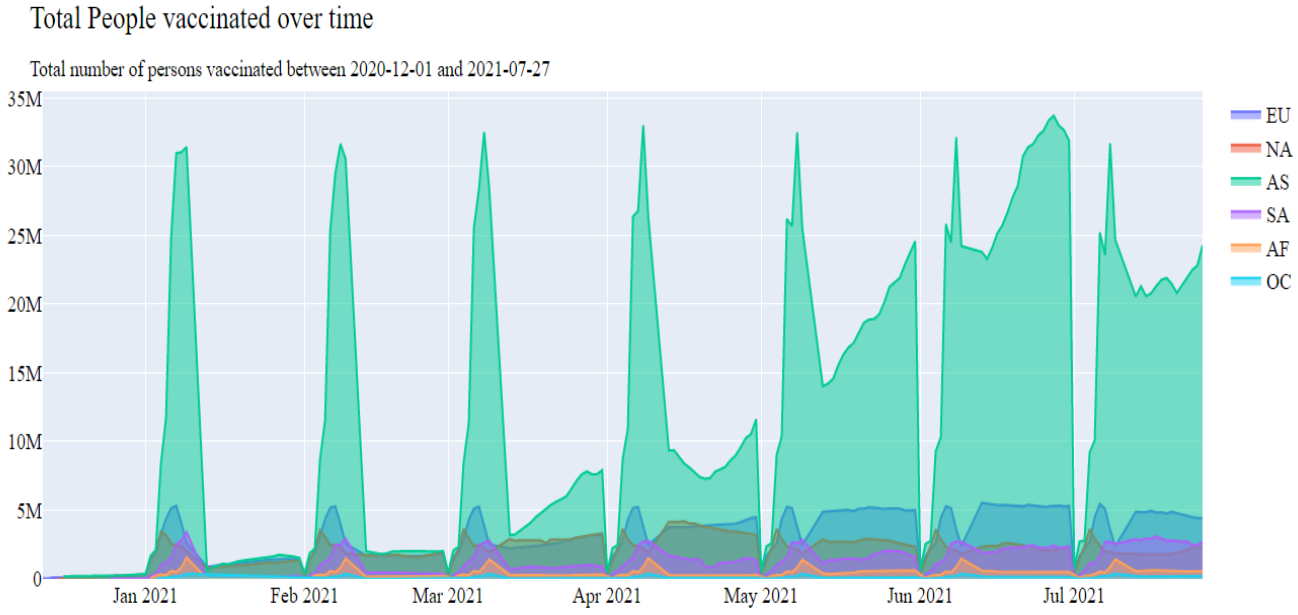


- China is the country to have highest number of populations vaccinated.

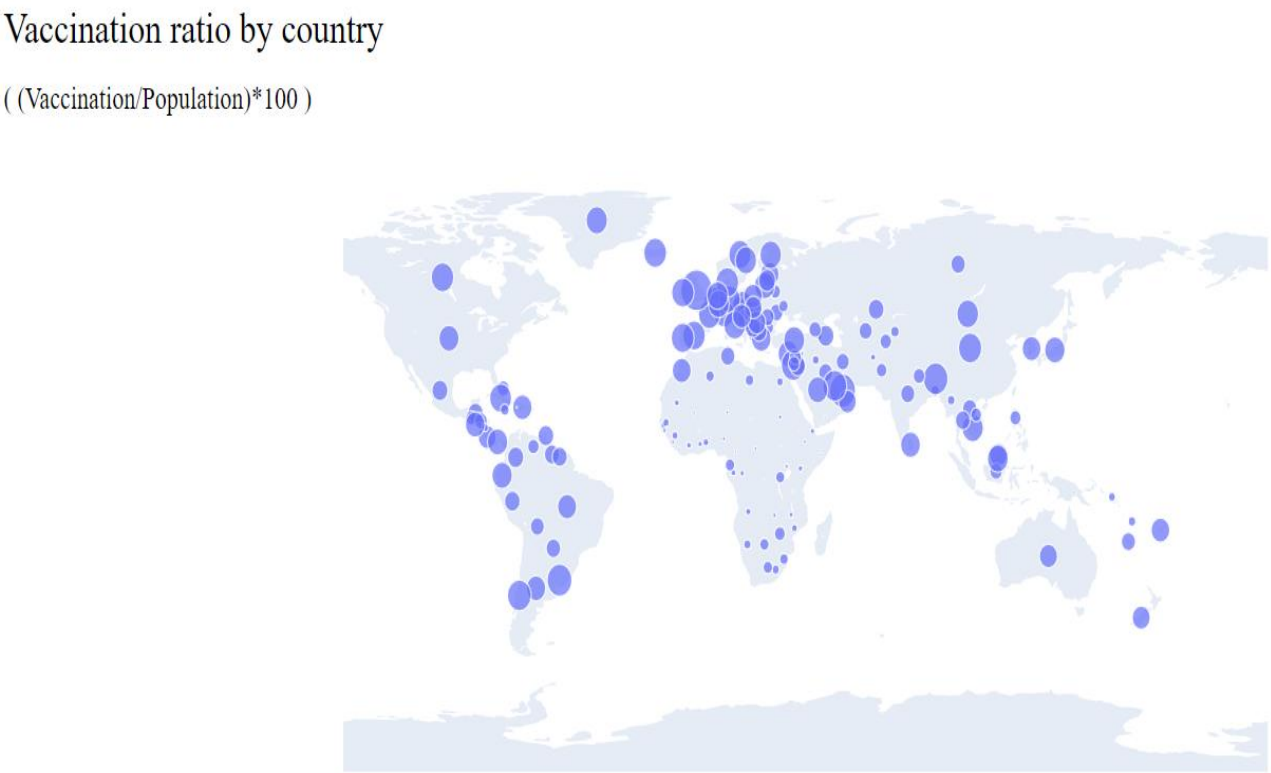


To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

- Asia had the best vaccination drive amongst all other continents.

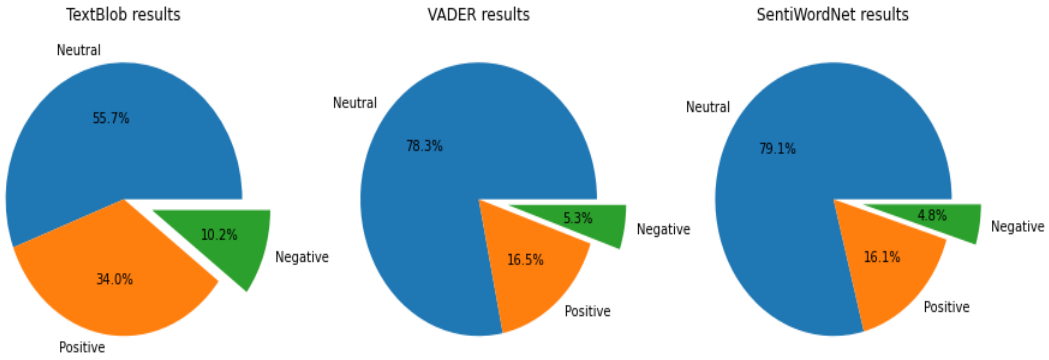


- Great Britain is the country that has vaccinated the largest percent of people from its population.



To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

- According to different sentiment analysis methods mostly the tweets from public were neutral. After that most tweets were positive about vaccination drives.



- We saw that population is inversely proportional to vaccine drive from the regression analysis as the r coefficient is negative between population and daily vaccinations.

Analytical Overview

- Excel data was cleaned before analysing such as duplicate rows and non-essential column was removed from the databases.
- For all graphs and data visualisation, we used different methods to showcase our knowledge of python.
- All major findings and recommendations are based on EDA which is explained in Documentation Page.

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

Documentation Page

Cleaning data

- First by using “describe” function we got to know there was no “NULL” data in excel.
- We also got a brief summary of all the columns.
- We generated reports for initial visualisation of data using ProfileReport.
- We also dropped data which were duplicate rows and a non-essential column for our first data set of country_vaccinations.csv.
  (This was done for all the datasets i.e. country_vaccinations_by_manufacturer, vaccination_all_tweets.csv & country_profile_variables)

```
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 18 08:30:31 2021

@author: patel
"""
import pandas as pd           #importing pandas
import pycountry_convert as pc #importing pycountry_convert, this will be useful when converting iso names of countries. like India to IND and vice versa.
import pypopulation as pop     #importing pypopulation for getting population from iso names of country.
from plotly.offline import plot # importing plotly for better UI of graphs.
import plotly.express as ex     # importing plotly.express for better UI of graphs.
import numpy as np              # importing numpy
import plotly.graph_objs as go       # importing plotly.graph_objs for better UI of graphs.
import matplotlib.pyplot as plt  # importing matplotlib for generating normal graphs.
from pandas_profiling import ProfileReport     # importing pandas_profiling for geenerating reports of initial data.
from sklearn.preprocessing import MultiLabelBinarizer  # importing sklearn.preprocessing for preproccing our data for sentiment analysis.
import re # importing regular expressions which is used while sentiment analysis.
import nltk  # importing nltk for sentiment analysis
nltk.download('punkt')
from nltk.tokenize import word_tokenize #importing word_tokenize for tokenizing the text while sentiment analysis.
from nltk import pos_tag   # importing pos_tagfor sentiment analysis
nltk.download('stopwords')
from nltk.corpus import stopwords # importing stopwords sentiment analysis
nltk.download('wordnet')
from nltk.corpus import wordnet # importing wordnet sentiment analysis
nltk.download('averaged_perceptron_tagger')
import seaborn as sns
from wordcloud import WordCloud # importing WordCloud sentiment analysis

data_CV = pd.read_csv("country_vaccinations.csv") #Data of vaccinations country wise.
report = ProfileReport(data_CV) # generating initial report for data_CV(vaccinations country wise).
report.to_file("Country_Vaccinations.html") # SAving our report as Country_Vaccinations.html
data_CV.describe(include='all')  # computes a summary of statistics pertaining to the DataFrame.
data_CV=data_CV.drop_duplicates() # drops all the duplicate rows from database.
data_CV = data_CV.drop('daily_vaccinations_raw', axis=1) #droping daily_vaccinations_raw as it is not useful column.
data_CV['date'] = pd.to_datetime(data_CV['date']) #converting date column to datetime type.
data_CV = data_CV.sort_values('date', ascending=True) #sorting our data date wise
data_CV['date'] = data_CV['date'].dt.strftime('%Y-%m-%d') # assigning a proper format of dates in database
data_CV['iso_code'] = data_CV['iso_code'].str.replace('OWID_ENG','GBR') # assigning the continent manually where the pycountry does not recognise a country name or the country is not recognised by the UN.
data_CV['iso_code'] = data_CV['iso_code'].str.replace('OWID_KOS','GBR')#KOSOVO does not have iso code so its continent is EUrope so we are assigning it as Great bretain
data_CV['iso_code'] = data_CV['iso_code'].str.replace('OWID_CYN','CYP')# assigning the continent manually where the pycountry does not recognise a country name or the country is not recognised by the UN.
data_CV['iso_code'] = data_CV['iso_code'].str.replace('OWID_NIR','GBR')# assigning the continent manually where the pycountry does not recognise a country name or the country is not recognised by the UN.
data_CV['iso_code'] = data_CV['iso_code'].str.replace('OWID_SCT','GBR')# assigning the continent manually where the pycountry does not recognise a country name or the country is not recognised by the UN.
data_CV['iso_code'] = data_CV['iso_code'].str.replace('OWID_WLS','GBR')# assigning the continent manually where the pycountry does not recognise a country name or the country is not recognised by the UN.
data_CV['iso_code'] = data_CV['iso_code'].str.replace('PCN','NZL')# assigning the continent manually where the pycountry does not recognise a country name or the country is not recognised by the UN.
data_CV['iso_code'] = data_CV['iso_code'].str.replace('SXM','USA')# assigning the continent manually where the pycountry does not recognise a country name or the country is not recognised by the UN.
data_CV['iso_code'] = data_CV['iso_code'].str.replace('TLS','IND')# assigning the continent manually where the pycountry does not recognise a country name or the country is not recognised by the UN.
Continent =[] #creating an empty list of continent.
for i in data_CV["iso_code"]:  # getting each country's continent
    x=pc.country_alpha3_to_country_alpha2(i) # first converting country name alpha 3 to alpha 2
    Continent.append(pc.country_alpha2_to_continent_code(x)) # then converting country's alpha 2 to continent code and appending that to continent list.
data_CV['Continent'] = Continent # creating a new column of Continent from continent List.
```

To: Analytic Manager, United Health Care.
From: Aditya Nagori
Subject: COVID- 19 Analysis

All graphs Description

GENERAL OVERVIEW

- **People Fully vaccinated in the world is : 2.554253e+09**

CODE:

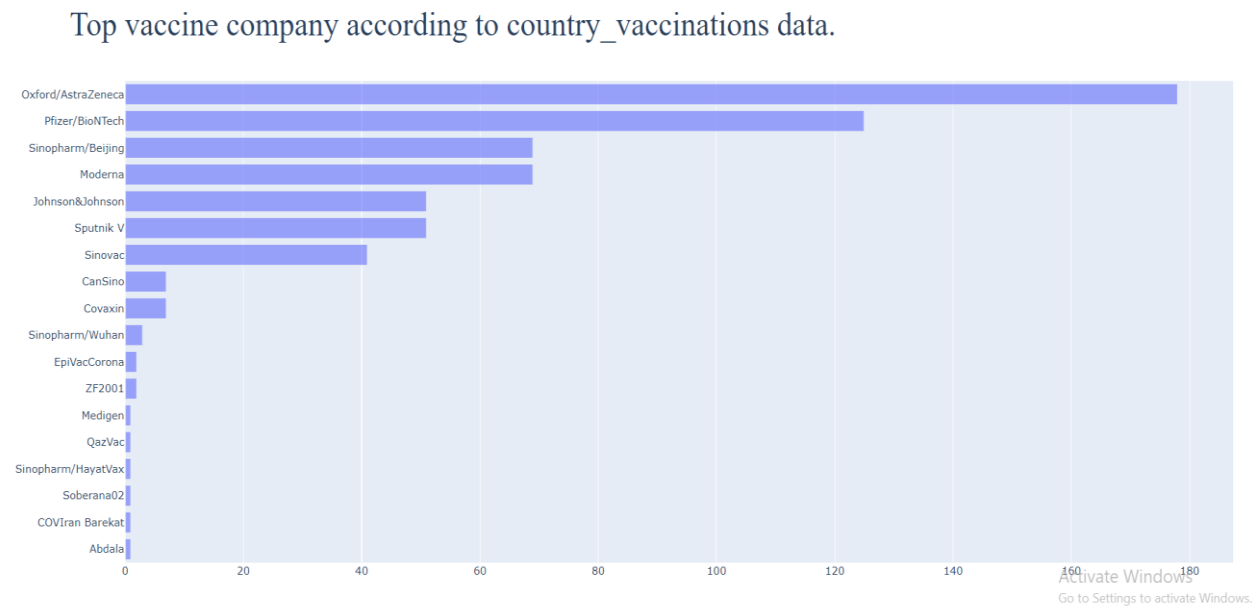
```
#General overview
data_CV['country'].nunique()  #Gets the count of unique country names
data_CV['vaccines'].nunique() #Gets the count of unique vaccine names
data_CV['daily_vaccinations'].sum() # gets the sum of total vaccinations done in world.

#fully vaccinated count country wise
data_CV_vacc = data_CV[["country", "daily_vaccinations"]]
p1=data_CV_vacc.groupby(['country']).sum()
print (p1)

#gets the count of people fully vaccinated country wise.
data_CV_fully_vacc = data_CV[["country", "people_fully_vaccinated"]]
p=data_CV_fully_vacc.groupby(['country']).max()
print(p)
print(p.sum()) #gets the total number of people fully vaccinated in world.
```

OUTPUT:

```
                daily_vaccinations
country
Afghanistan                2921017.0
Albania                    1631921.0
Algeria                    9248920.0
Andorra                      92667.0
Angola                     2691230.0
...                              ...
Wales                      4576771.0
Wallis and Futuna             9404.0
Yemen                       316493.0
Zambia                      634523.0
Zimbabwe                   4951761.0

[222 rows x 1 columns]
                people_fully_vaccinated
country
Afghanistan                   2149746.0
Albania                        731577.0
Algeria                       4174623.0
Andorra                         41831.0
Angola                         983587.0
...                                 ...
Wales                         2209568.0
Wallis and Futuna                4950.0
Yemen                           14909.0
Zambia                         291947.0
Zimbabwe                      2082964.0

[222 rows x 1 columns]
people_fully_vaccinated    2.554253e+09
dtype: float64
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

- Top Vaccine Company according to country vaccinations data is Oxford/AstraZeneca.



The above graph tells us about the total number of vaccines that are distributed by Vaccine Companies from all the Countries. Oxford/AstraZeneca is the Company that stands first across the list followed by Pfizer/BioNTech. All the Companies are arranged in the increasing order from Bottom to Top of the Graph.

Code:

```
#top vaccine company according to country_vaccinations data.
forsub_set_vacc = data_CV.copy() #copying data_CV to forsub_set_vacc
forsub_set_vacc = forsub_set_vacc.dropna(subset=['vaccines']) #droping all NAN in data set

df_vac = forsub_set_vacc.groupby(['iso_code','vaccines']).max().reset_index() # grouping data by iso_code and vaccines.
df_vac['vaccines_split'] = df_vac['vaccines'].apply(lambda x: [w.strip() for w in x.split(',')]) # splitting each vaccine name from eaach rows.
df_vac.head()

one_hot = MultiLabelBinarizer() # allows me to encode multiple labels per instance.

vac_data = one_hot.fit_transform(df_vac['vaccines_split'])
vac_names = one_hot.classes_
vac_countries = df_vac['country']

final_vac_df = pd.DataFrame(data=vac_data, columns=vac_names, index=vac_countries)
final_vac_df = final_vac_df.reset_index()
final_vac_df.head()
ncountrys_vac = final_vac_df[vac_names].sum(axis=0).sort_values()


#setting up the x axis and y axis.
fig = go.Figure(go.Bar(
    x = ncountrys_vac.values,
    y = ncountrys_vac.index,
    orientation = 'h',
))
#graph marker sze anad opacity.
fig.update_traces(
    marker_line_width=1.5,
    opacity=0.6,
)
fig.update_layout(
    title='<span style="font-size:36px; font-family:serif">Top vaccine company according to country_vaccinations data.</span>',# Graph title.
)
plot(fig)
```

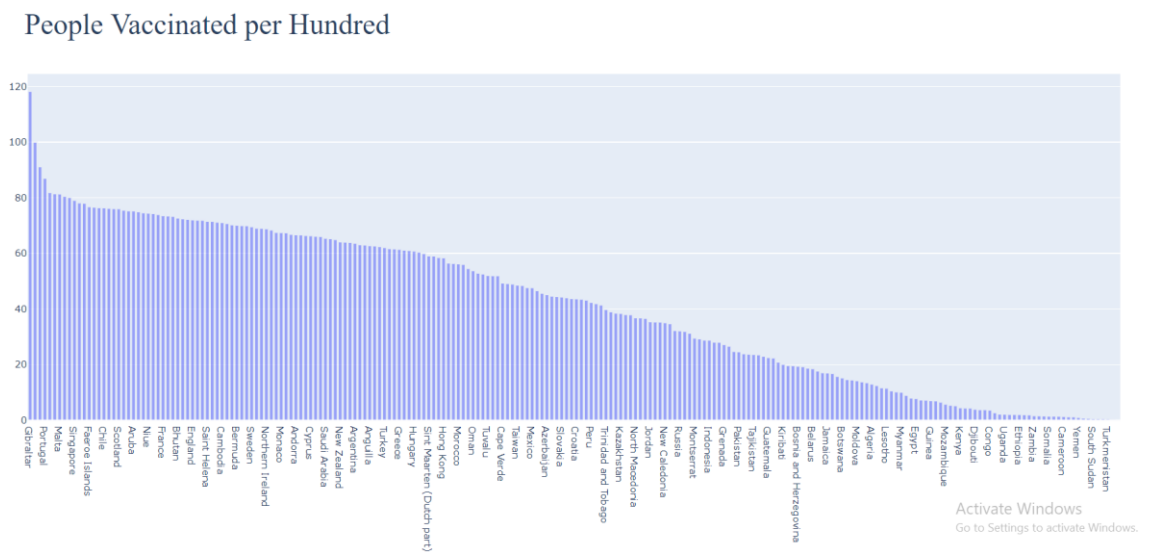
To: Analytic Manager, United Health Care.
From: Aditya Nagori
Subject: COVID- 19 Analysis

- **People Vaccinated per Hundred Country wise was most by Gibraltar.**

Code:

```
#People Vaccinated per Hundred Country wise
vacc_per_hundred = data_CV.copy()   #copying data_CV to vacc_per_hundred
vacc_per_hundred = vacc_per_hundred.sort_values('people_vaccinated_per_hundred', ascending=False).\
    drop_duplicates(subset=['country'], keep='first', ignore_index=True) #droping duplicate rows and sorting data .

fig_vacc_per_hundred = go.Figure(go.Bar(
    x = vacc_per_hundred['country'],
    y = vacc_per_hundred['people_vaccinated_per_hundred'],
))
fig_vacc_per_hundred.update_traces(
    marker_line_width=1.5,
    opacity=0.6,
)
fig_vacc_per_hundred.update_layout(
    title='<span style="font-size:36px; font-family:serif">People Vaccinated per Hundred</span>',
)
plot(fig_vacc_per_hundred)
```

Output:



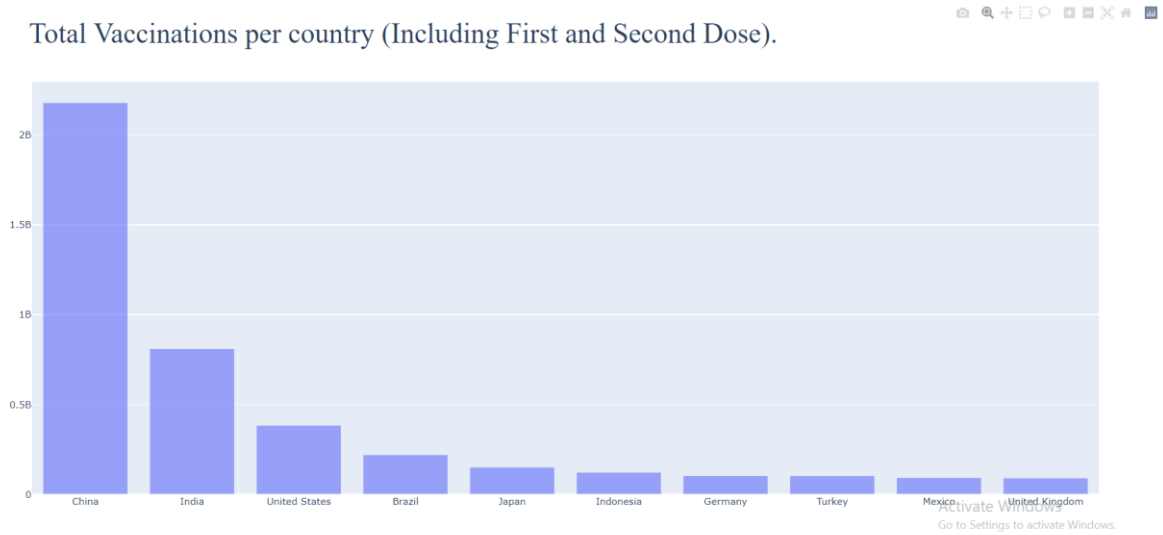
- China shows the highest rate of vaccinations of both the doses followed by India and United States. On the other hand Indonesia, Germany, Mexico and Turkey stand on almost same number of vaccinations.

CODE:

```
#Total Vaccinations per country (Including First and Second Dose).
vacc_per_country = data_CV.copy()
vacc_per_country = vacc_per_country.sort_values('total_vaccinations', ascending=False).\
    drop_duplicates(subset=['country'], keep='first', ignore_index=True)

vacc_per_country=vacc_per_country.head(10)

fig_vacc_per_country = go.Figure(go.Bar(
    x = vacc_per_country['country'],
    y = vacc_per_country['total_vaccinations'],
))
fig_vacc_per_country.update_traces(
    marker_line_width=1.5,
    opacity=0.6,
)
fig_vacc_per_country.update_layout(
    title='<span style="font-size:36px; font-family:serif">Total Vaccinations per country (Including First and Second Dose).</span>',
)
plot(fig_vacc_per_country)
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

Output:



From the above bar plot we can see that most of the vaccinations is done only in China from December of 2020 till October 2021. India stands in second place in that list. As the population of both the countries is very huge, the number of people vaccinated are more.

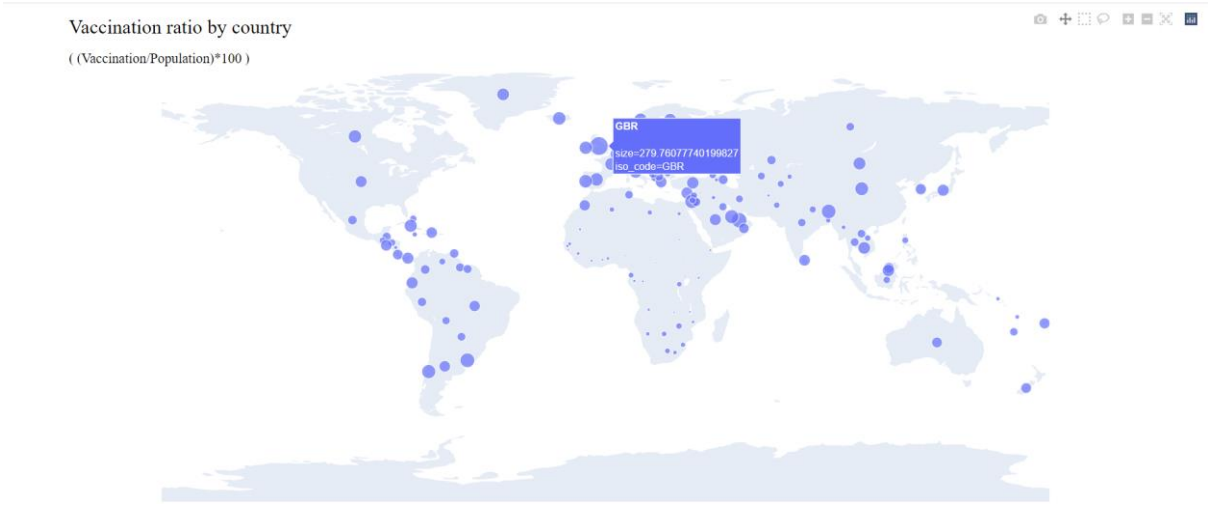
- The percentage of total number of people vaccinated by the total population of the country, GRB tops the list.

CODE:

```
#Vaccination ratio by country(Vaccination/Population )

ratio_cases_pop1 = data_CV.copy()
ratio_cases_pop0 = ratio_cases_pop1[["country","iso_code", "daily_vaccinations"]]
ratio_cases_pop = pd.DataFrame( (ratio_cases_pop0.groupby(['iso_code'],as_index=False).sum()))
ratio_cases_pop['Population'] = pop.get_population_a3(str(ratio_cases_pop["iso_code"]))


# getting population of each country and saving it to new column
Population = []
for i in ratio_cases_pop["iso_code"]:
    Population.append(str(pop.get_population_a3(i)))
ratio_cases_pop['Population']= Population
ratio_cases_pop['Population'] = pd.to_numeric(ratio_cases_pop['Population'], errors='coerce')
ratio_cases_pop = ratio_cases_pop.replace(np.nan, 0, regex=True)
ratio_cases_pop["Ratio"] = ratio_cases_pop['daily_vaccinations']/ratio_cases_pop['Population']

fig_ratio = ex.scatter_geo(
         ratio_cases_pop, # Passing the dataframe
         locations="iso_code", # Select the column with the name of the countries,
         locationmode='ISO-3', # We pass the parameter of determining the country on the map (by name)
         hover_name="iso_code",  # Passing values for the signature on hover
         size=ratio_cases_pop["Ratio"]*100 # Passing a column with values
)

fig_ratio.update_layout(
    # Set the name of the map
    title_text='Vaccination ratio by country<br><sub>( (Vaccination/Population)*100 ) </sub>',
    legend_orientation='h', # Place the legend caption under the chart
    legend_title_text='', # Remove the name of the legend group
    # Determine the map display settings (remove the frame, etc.)
    geo=dict(
       showframe=False,
       showcoastlines=False,
       projection_type='equirectangular'
    ),
     font=dict(
       family='TimesNewRoman',
       size=18,
       color='black'
    )
    )
plot(fig_ratio)
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

Output:



- Asia gets a count of 31.4452M of vaccinations within the first 10 days of January. Whereas North America and Oceania show the least number of vaccinations within the same period. Europe marks 5.31M vaccinations within the first week of January. Within the first four days coverage of January North America puts the pint to 3.45 M vaccinations. Africa gets 1.5M within the first nine-day period of January

  February sees same number of vaccinations with respect to Asia. Whereas Europe observes a slight increase i.e., 5.2M. Oceania sees a slight decrease on the contrast i.e., 371k. North America witnesses a rise in its vaccinations, which rises to 3. 54M.Africa gets a count of 1.8M vaccinations

  In the month of March Asia observes rise in its vaccination count which becomes 32.54M followed by Europe with 5.1148M, North America with 3.148M, South America WITH 2.63M, Africa 1.41M and Oceania with 371.021k.

  In similarity with March, April also sees increase in the vaccinations for Asia and Europe with 33.071M and 5.71M respectively. North America and South America maintain the constant level of vaccinations in the first week of April. Africa and Oceania also get the number of vaccinations which are similar to the month of March in the first 10-day period of April.

  In the month of May Asia sees a slight decrement in its vaccinations which is 32.50 M, Europe goes with 5.14 M vaccinations in the first week of May, North America sets to 3.61M vaccinations, South America gets 2.57 M vaccinations and Europe with 1.9M by 9th of May. Oceania makes 370.48k vaccinations in May.

  June witnessed internal rise in the number of vaccinations with respect to Asia from June 8 to June 27 i.e., 32.15M and 33.15M respectively. Europe marks a rise from June 7 with

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

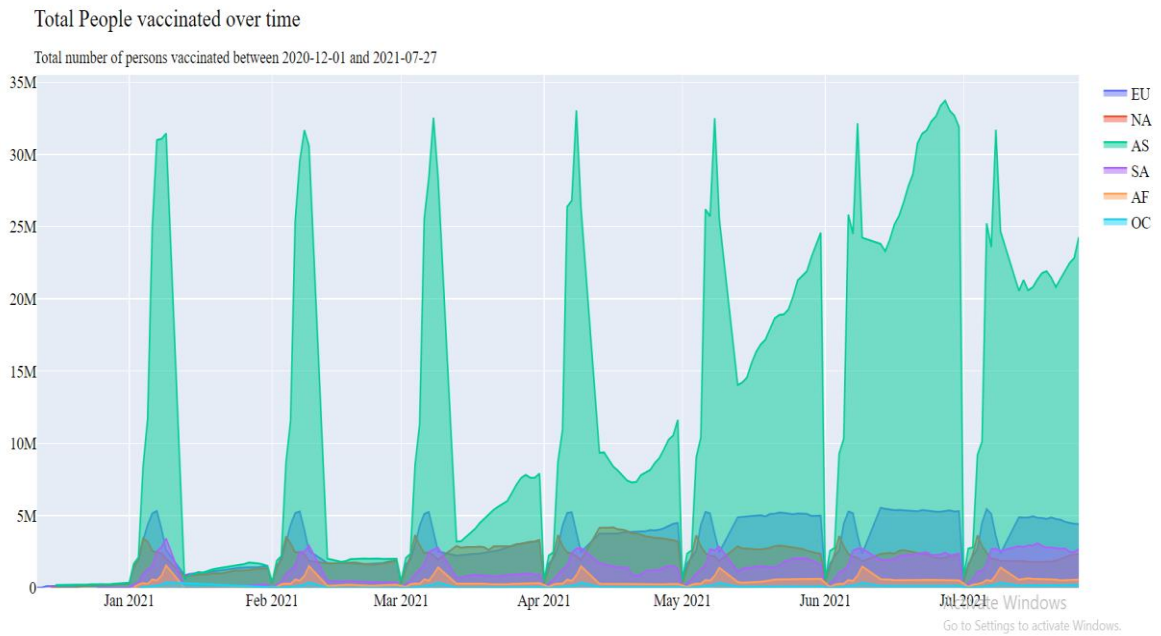
5.15M to June 13 with 5.54M North America sees 2.58M vaccinations within the first four days of June. South America gets a rise of number of vaccinations to 2.70M by June 9. Africa gets 1.47M vaccinations within the first 10 day period in June. Oceania observes decrease in its number of vaccinations from June 9th to June 13 i.e. 371k to 141k

Asia experienced 31.1M vaccinations on July 8 which went down to 21.77M by July 18. This figure has again seen a rise by July 26 resulting in 24.2806M vaccinations. Europe estimated 5.45M vaccinations by July 6 and by July 13 4.87 M vaccinations have taken place which marked a decrease. North America has increased number of vaccinations by the end of 4th of July marking 3.61M. Africa gets a count of 1.415M in the first 10 days of July. Oceania has risen of its number of vaccinations from the past month to 358.388k by 9th of July.

Code:

```
#vaccination_progress continent wise between 2 dates

people_vaccinated_overtime = data_CV.copy()
people_vaccinated_overtime=people_vaccinated_overtime.groupby(['Continent', 'date'],as_index=False).agg({'daily_vaccinations': 'sum',  'people_vaccinated_per_hundred': 'sum'})
people_vaccinated_overtime = people_vaccinated_overtime.reset_index().sort_values('date')
people_vaccinated_overtime = people_vaccinated_overtime.query('date > "2020-12-01" and date < "2021-07-27"')

people_vaccinated_overtime = people_vaccinated_overtime[people_vaccinated_overtime['daily_vaccinations'] != 0]
fig_people_vaccinated = go.Figure()
for region in people_vaccinated_overtime['Continent'].unique():
    fig_people_vaccinated.add_traces(go.Scatter(
        x = people_vaccinated_overtime.query(f'Continent == "{region}"')['date'],
        y = people_vaccinated_overtime.query(f'Continent == "{region}"')['daily_vaccinations'],
       fill = 'tozeroy',
        mode = 'lines',
        name = region,
    ))

fig_people_vaccinated.update_layout(
    # Set the name of the map
    title_text='Total People vaccinated over time <br><sub>Total number of persons vaccinated between 2020-12-01 and 2021-07-27</sub>',
    font=dict(
       family='Serif',
       size=18,
       color='black'
    )
)
plot(fig_people_vaccinated)
```

Output:



To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

- To begin with the month of the January , Europe's vaccination rate after the first ten days of January is 2678.49 whereas Asia had 1390.27. North America has a rate of 657.14 within the same duration. South America has 404.64. Africa has witnessed a rise in its vaccination rate from 81.21 on jan 4 to 232.73 by jan 9. Oceania had a rise from 96.38 to 200.12 in the first week of January .

  In the month of February Europe experienced a decrease in its vaccination rate from 2409.96 during 9th of February to 391.52 at the end of February. In the beginning of February Asia had a rate of 1391.41, at the end of first week of February, North America has 680.82 South America marked 311.27 whereas Oceania has secured a rate of 518.74. Africa had the least rate of vaccination reportedly 122.8.

  Europe had a vaccination rate of 2255.42 after the first week of march which has dropped to 773.23 by the end of the month. North America marked as 1143.83 and Asia has witnesses a rate of 1323.82. South America bearing a rate of 491.44, on the other hand Africa witnessed an increase from 94.94 to 218.96. Oceania had a rate of 100.32 after the first week of march.

  Europe marked a rate of 2203.11 at the end of first week of April by the end of April it has dropped to 796.18.Asia has seen a rate of 1309.8 whereas North America has witnessed a rate of 702.77 during the first week. South America has seen a rate of 459.67, Africa being the lowest with 137.96.

  2511.85 is the rate of vaccination of Europe during the beginning of May by the end of the month it has dropped down to 1601.38. Similarly, Asia had a rate of 1303.07 and it was changed to 596.35 by the end of May. North America had a vaccination rate of 493.55 in the mid od May and by the end of the month it has become 440.6. South America had a rate of 413.38 and Oceania had 103.2

  Europe marked a rate of 2285.28 during the beginning of June, at the end of the month it was 1483.1. Asia has witnessed a rate of 984.97 , North America with 785.29. South America had witnessed a rate of 352.69. Oceania vaccination rate was changed from 62.04 to 38.63.

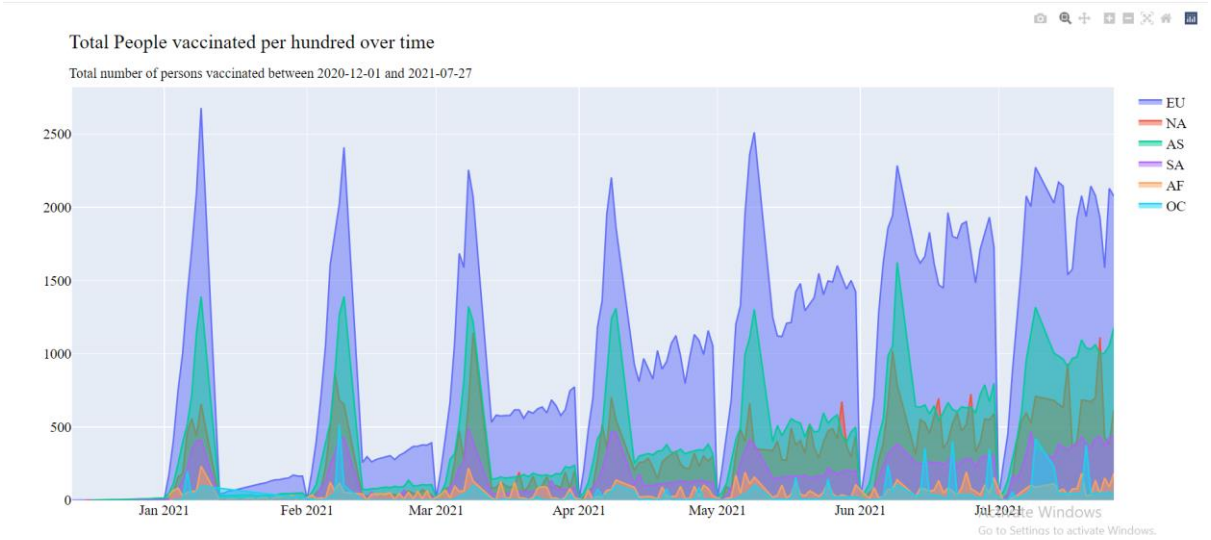
  Europe has marked 2273 at the beginning of July , in the mid July it was 2143.73 and at the end of July it was 1578.41. Asia's vaccination rate was 1317.42 on the other hand North America's vaccination rate was 672.08 South America had a rate of 290.24. Africa had a rate of 92.19 Oceania had a rate of 46.97

  CODE:

```
#vaccination_progress per hundred continent wise between 2 dates
fig_people_vaccinated_per_hundred = go.Figure()
for region in people_vaccinated_overtime['Continent'].unique():
    fig_people_vaccinated_per_hundred.add_traces(go.Scatter(
        x = people_vaccinated_overtime.query(f'Continent == "{region}"')['date'],
        y = people_vaccinated_overtime.query(f'Continent == "{region}"')['people_vaccinated_per_hundred'],
       fill = 'tozeroy',
        mode = 'lines',
        name = region,
    ))

fig_people_vaccinated_per_hundred.update_layout(
    # Set the name of the map
    title_text='Total People vaccinated per hundred over time <br><sub>Total number of persons vaccinated between 2020-12-01 and 2021-07-27</sub>',
    font=dict(
       family='Serif',
       size=18,
       color='black'
    )
)
plot(fig_people_vaccinated_per_hundred)
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

Output:

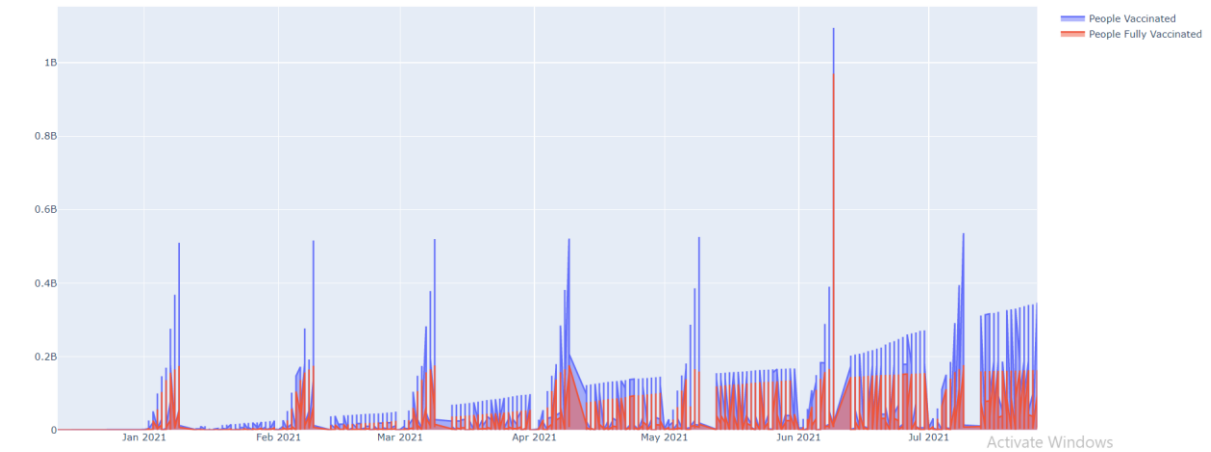


- People vaccinated vs people fully vaccinated in the world over time.

CODE:

```
#People vaccinated vs people fully vaccinated in the world

people_vaccinated_Vs_Full = data_CV.copy()
people_vaccinated_Vs_Full=people_vaccinated_Vs_Full.groupby(['Continent', 'date'],as_index=False).agg({'people_vaccinated': 'max',  'people_fully_vaccinated': 'max'})
people_vaccinated_Vs_Full=people_vaccinated_Vs_Full = people_vaccinated_Vs_Full.reset_index().sort_values('date')
people_vaccinated_Vs_Full = people_vaccinated_Vs_Full.query("date > "2020-12-01" and date < "2021-07-27"")
people_vaccinated_Vs_Full=people_vaccinated_Vs_Full.dropna()
df = ex.data.iris()

plot0 = go.Scatter(
    x = people_vaccinated_Vs_Full['date'],
    y = people_vaccinated_Vs_Full['people_vaccinated'],
    fill = 'tozeroy',
    mode = 'lines',
    name = 'People Vaccinated'
)

plot1 = go.Scatter(
    x = people_vaccinated_Vs_Full['date'],
    y = people_vaccinated_Vs_Full['people_fully_vaccinated'],
    fill = 'tozeroy',
    mode = 'lines',
    name = 'People Fully Vaccinated'
    )
output = [plot0, plot1]
plot(output)
```

Output:

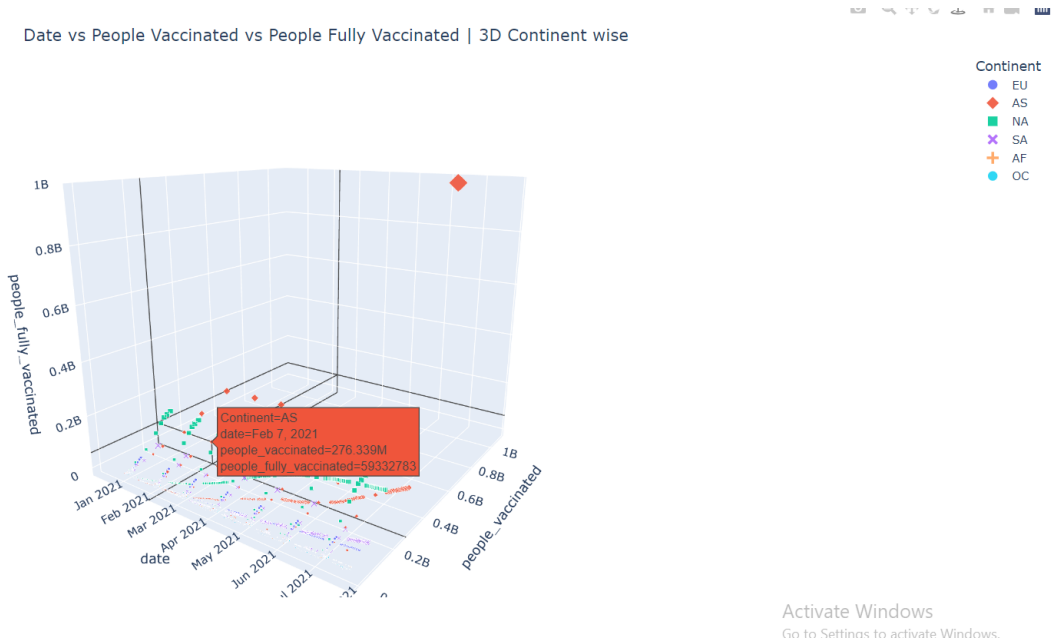


To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

- 3D figure of Date vs People Vaccinated vs People Fully Vaccinated.
  Code:

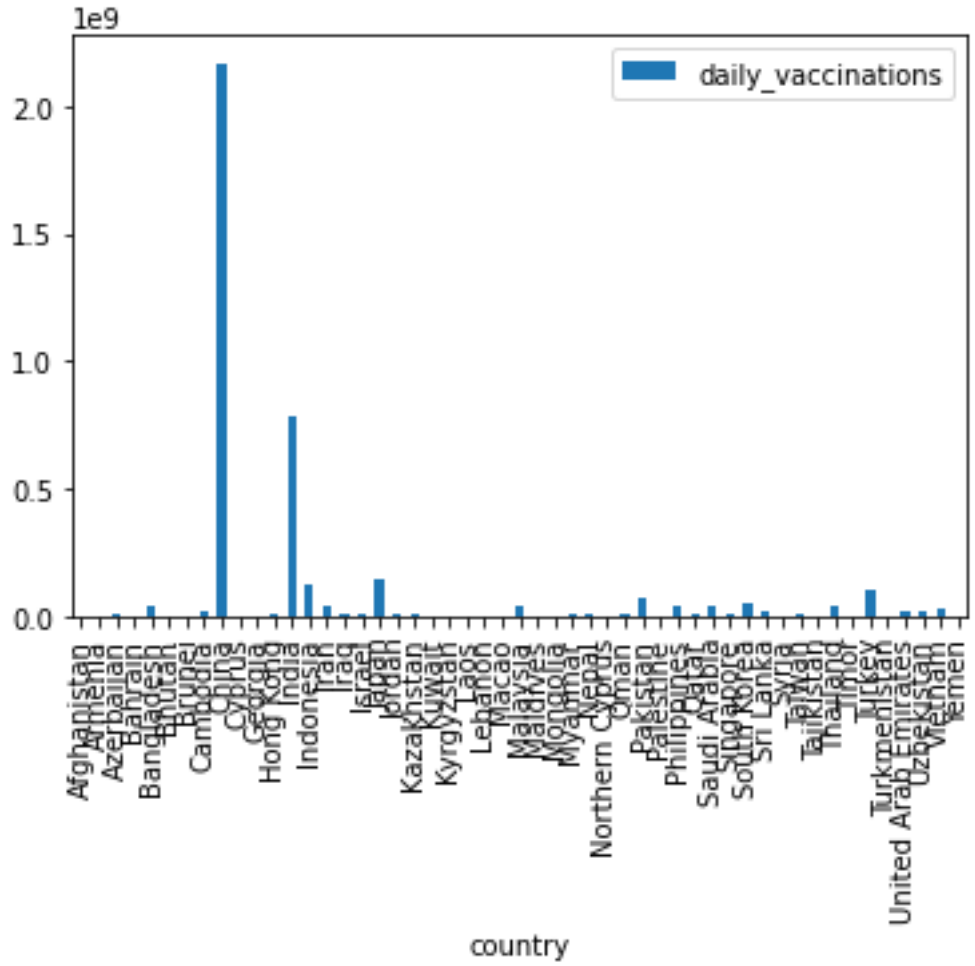
```
#3D figure of Date vs People Vaccinated vs People Fully Vaccinated

fig_3d = ex.scatter_3d(people_vaccinated_Vs_Full, x='date', y='people_vaccinated', z='people_fully_vaccinated',
                    color='Continent',
                    hover_data=['Continent'],
                    size = 'people_fully_vaccinated',
                    opacity=0.9,
                    symbol = 'Continent')

fig_3d.update_layout(title='Date vs People Vaccinated vs People Fully Vaccinated | 3D Continent wise')

fig_3d.update_layout(
        title={
            'y':0.95,
            'x':0.5
        }
    )
plot(fig_3d)
```

  Output:



To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

ASIA:

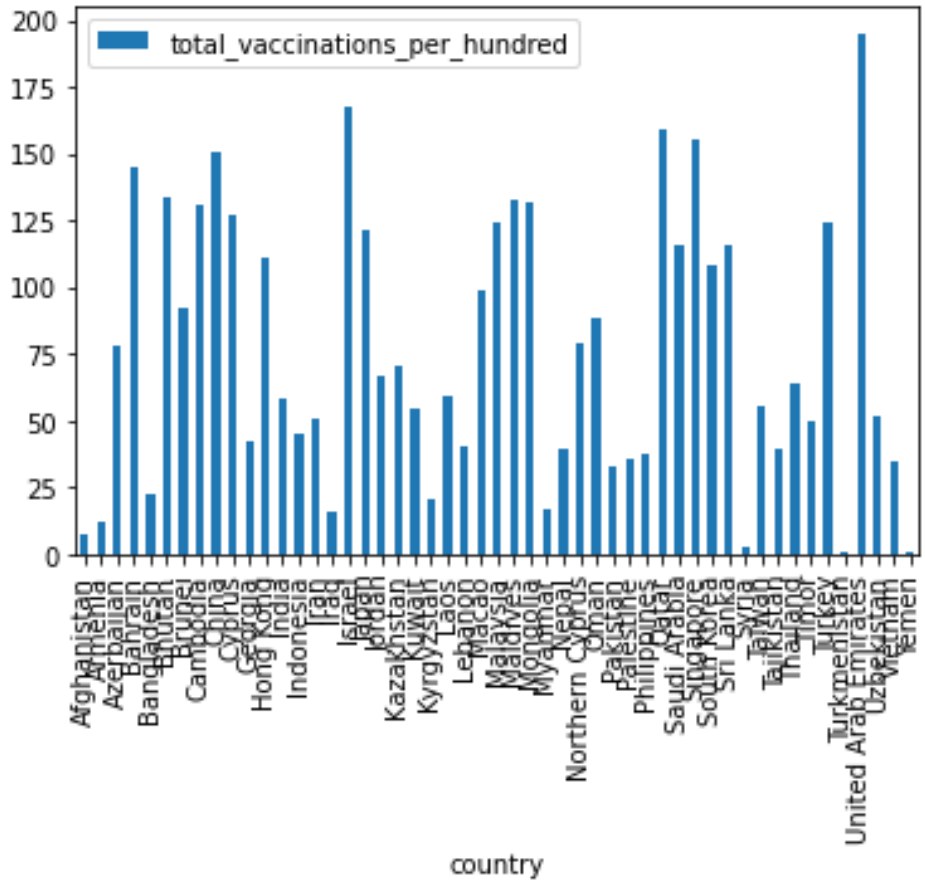


From the above bar plot we can see that most of the vaccinations is done only in China from December of 2020 till October 2021. India stands in second place in that list. As the population of both the countries is very huge, the number of people vaccinated are more.

CODE:

```
data_asia_for_grouping = data_asia[["country", "daily_vaccinations"]] #grouping asia data country wise and daily vaccinations i.e total vaccinations sum.
data_asia_grouped = (data_asia_for_grouping.groupby(['country']).sum().plot(kind='bar')) # plotting a bar graphs of the grouped data.
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

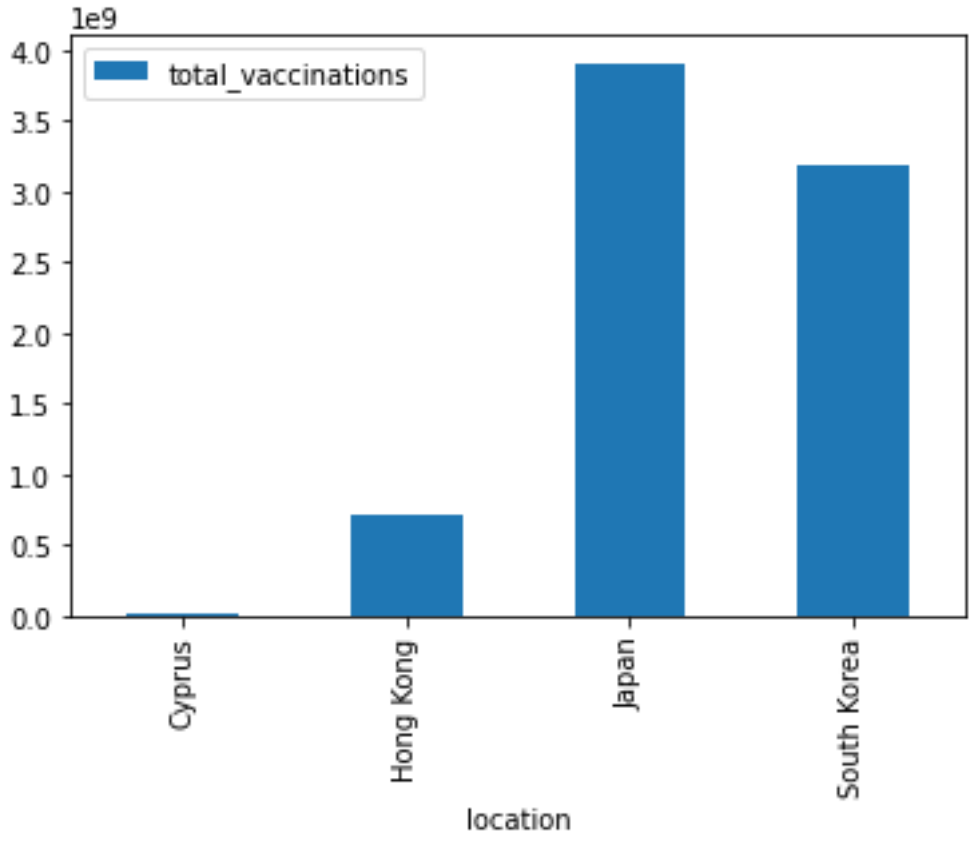


With a higher population Asia depicts mixed figures of number of vaccinations. Arab Emirates gets the greatest number of vaccinations. Next comes Israel, Azerbaijan, China, Arabia, Singapore, Bahrain, Bhutan, Cyprus and Japan. Syria, Turkmenistan and Yemen have marked the least count of vaccinations along with Afghanistan, Armenia, Myanmar, and Iraq. Average numbers are depicted by Malaysia, Maldives, Vietnam, Uzbekistan, Cambodia, Georgia, Lebanon, Kyrgyzstan, Jordan, Nepal, Northern Cyprus, Timor, Kazakhstan, and Tajikistan.

CODE:

```
data_asia_for_Vaccperhunderd = data_asia[["country", "total_vaccinations_per_hundred"]] #grouping asia data country wise
data_asia_for_Vaccperhunderd.groupby(['country']).max().plot(kind='bar')# plotting a bar graphs of the grouped data.
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

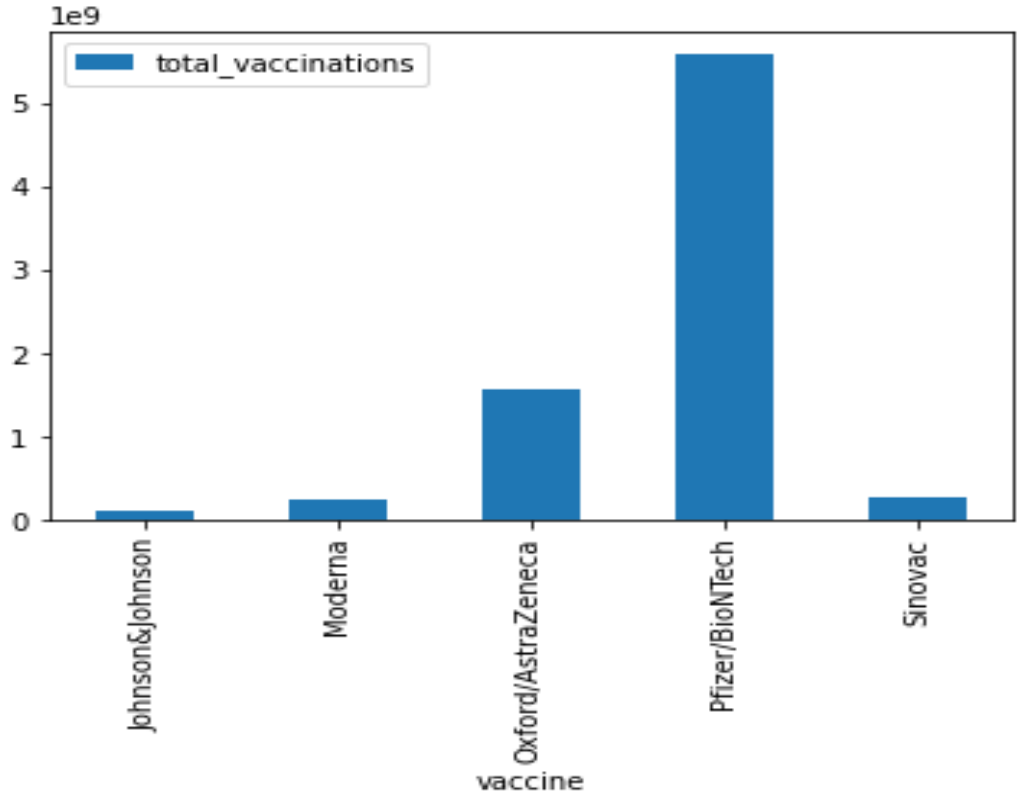


The above graph shows which Asian continent country received max doses from manufacture.

CODE:

```
data_CV_by_manufacturer = pd.read_csv("country_vaccinations_by_manufacturer.csv") #getting country_vaccinations_by_manufacturer
report_manufactur = ProfileReport(data_CV_by_manufacturer) # generating initial report of country_vaccinations_by_manufacturer
report_manufactur.to_file("Country_Vaccinations_by_Manufacturer.html") # saving rreport as Country_Vaccinations_by_Manufacturer html file.
data_CV_by_manufacturer.describe(include='all') # computes a summary of statistics pertaining to the DataFrame.
data_CV_by_manufacturer= data_CV_by_manufacturer.drop_duplicates() # removes duplicate rows from database
data_CV_by_manufacturer['date'] = pd.to_datetime(data_CV['date']) #converting date column to datetime type.
data_CV_by_manufacturer = data_CV_by_manufacturer.sort_values('date', ascending=True) #sorting our data date wise
data_CV_by_manufacturer['location'] = data_CV_by_manufacturer['location'].str.replace('European Union','France') # manually setting up the location where the Country name is not proper for diving out data
Continent_manufactur =[] # creating an empty list of Continent_manufactur
for i in data_CV_by_manufacturer["location"]: # getiing continent from country name
    x=pc.country_name_to_country_alpha3(i) # converting country name to alpha 3 name
    y=pc.country_alpha3_to_country_alpha2(x) # converting alpha 3 name to alpha 2.
    Continent_manufactur.append(pc.country_alpha2_to_continent_code(y)) # Converting alpha 2 name to continent and appending it to the list.
data_CV_by_manufacturer['Continent_manufactur'] = Continent_manufactur # Creating a continent column fron continent list.

#Creating a seperate dataframe  for each continent
data_asia_manufacturer = data_CV_by_manufacturer[data_CV_by_manufacturer["Continent_manufactur"] == "AS"]
data_africa_manufacturer = data_CV_by_manufacturer[data_CV_by_manufacturer["Continent_manufactur"] == "AF"]
data_europe_manufacturer = data_CV_by_manufacturer[data_CV_by_manufacturer["Continent_manufactur"] == "EU"]
data_antarctica_manufacturer = data_CV_by_manufacturer[data_CV_by_manufacturer["Continent_manufactur"] == "AN"]
data_north_america_manufacturer = data_CV_by_manufacturer[data_CV_by_manufacturer["Continent_manufactur"] == "NA"]
data_south_america_manufacturer = data_CV_by_manufacturer[data_CV_by_manufacturer["Continent_manufactur"] == "SA"]
data_oceania_manufacturer = data_CV_by_manufacturer[data_CV_by_manufacturer["Continent_manufactur"] == "OC"]

# which asian continent country recieved max doses from manufacture.
data_asia_manufacturer_for_grouping = data_asia_manufacturer[["location", "total_vaccinations"]] # grouping the manufacrurer asia data location wise for getting vaccination doses country wise.
data_asia_grouped_manufacturer = (data_asia_manufacturer_for_grouping.groupby(['location']).sum().plot(kind='bar')) # plotting the grouped data
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

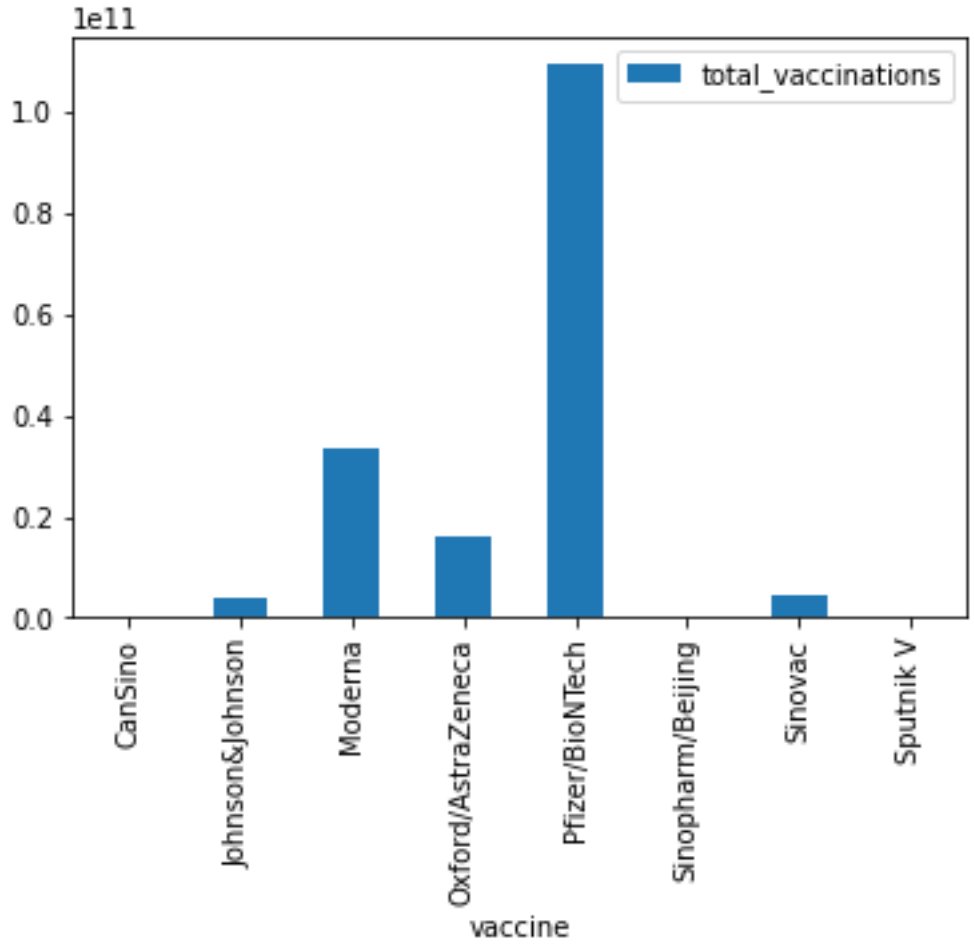


From the above bar plot we can see that almost 5.8 Billion Vaccines produced by the company Pfizer/BioNTech were manufactured by the Countries in Asia. A Decent amount of vaccines produced by the company Oxford/AstraZeneca were manufactured and a very few vaccines developed by the companies namely Johnson&Johnson, Moderna and Sinova were manufactured.

CODE:

```
# which vaccine companay sold max vaccines in asia
data_asia_manufacturer_companywise = data_asia_manufacturer[["vaccine", "total_vaccinations"]] # grouping data_asia_manufacturer by vaccine and total vaccination
data_asia_grouped_manufacturer_companywise = (data_asia_manufacturer_companywise.groupby(['vaccine']).sum().plot(kind='bar')) # plotting the grouped data.
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

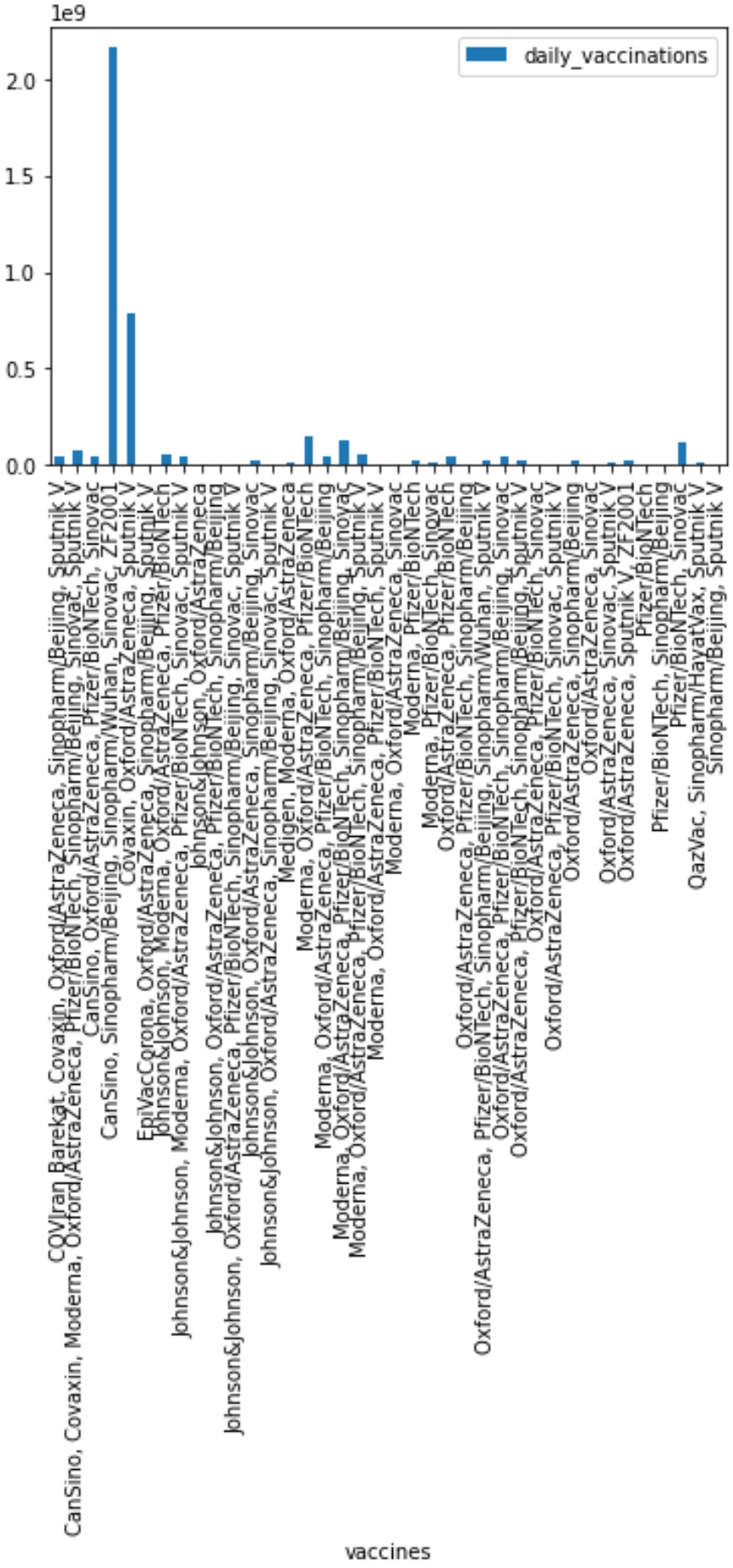


Most of the Vaccinations were manufactured by Pfizer/BioNTech. Companies like Sputnik V, Sinopharm/Beijing and CanSino has not manufactured any vaccines and few of the vaccines were supplied by the Companies namely Moderna, Oxford/AstraZeneca, Sinovac and Johnson&Johnson.

CODE:

```
# which vaccine companay sold max vaccines in world.
data_CV_by_manufacturer_all_countries = data_CV_by_manufacturer[["vaccine", "total_vaccinations"]]
data_asia_grouped_manufacturer_all_countries = (data_CV_by_manufacturer_all_countries.groupby(['vaccine']).sum().plot(kind='bar'))
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**



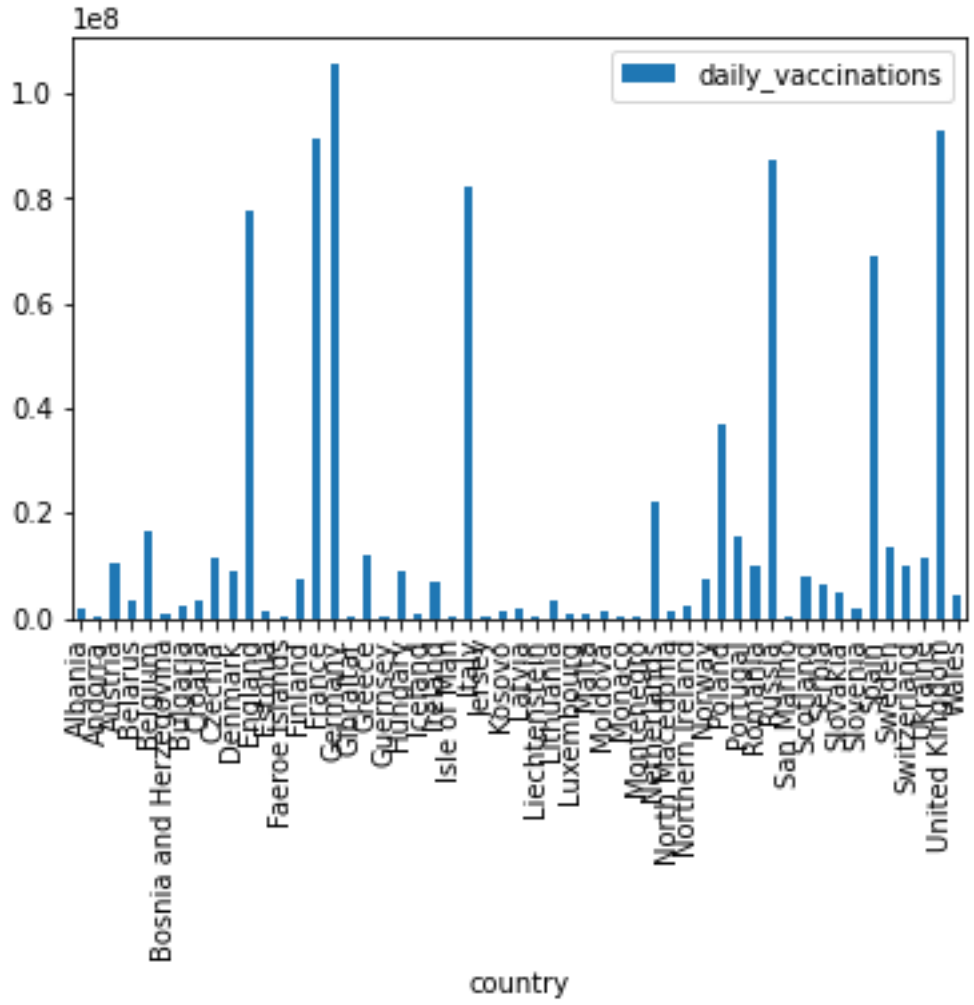
To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

The above graph shows which country uses which vaccine (Data vaccination country wise) according to the countries in Asia.

Code:

```
#Which country uses which vaccine (Data vaccination country wise) according to the countries in Asia.
data_asia_vaccination_company = data_asia[["country", "daily_vaccinations","vaccines"]]
data_asia_group_vacc_company = (data_asia_vaccination_company.groupby(['vaccines']).sum().plot(kind='bar'))
```

EUROPE ANALYSIS

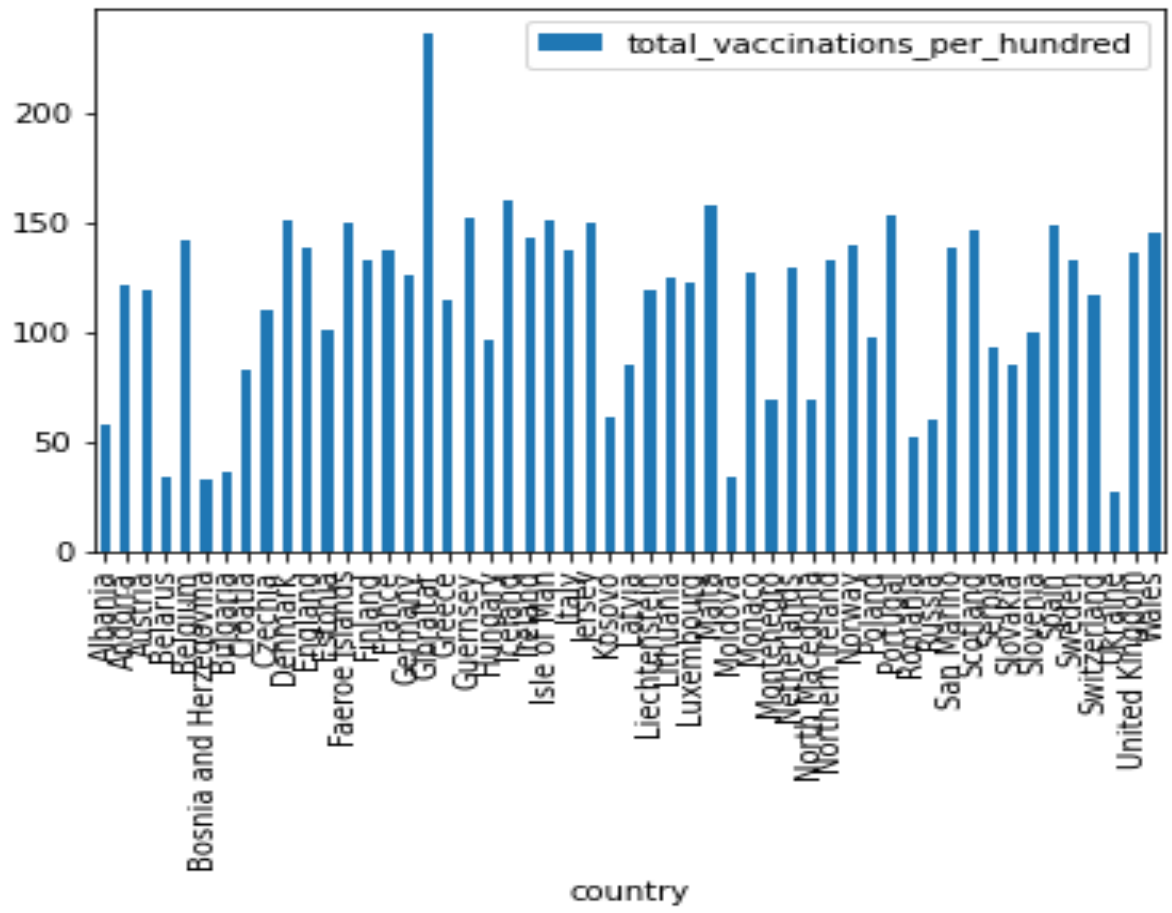


The above graph shows which European continent country received max doses from manufacture. In Europe, we can see that Germany stands first as more number of Daily Vaccinations vaccinations are done in Germany. Followed by France , United Kingdom, Russia , Italy, England and Spain. We can also notice that only few vaccinations were done in some countries like Albania, Andora, Bulgaria, Faeroe Islands, Guernsey, Iceland, Isle of Man, Kosovo, Malta , Moldova, Monaco and San Marino.

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

Code:

```
data_europe_for_grouping = data_europe[["country", "daily_vaccinations"]] #grouping Europe data country wise and daily vaccinations i.e total vaccinations sum.
data_europe_grouped = (data_europe_for_grouping.groupby(['country']).sum().plot(kind='bar')) # plotting a bar graphs of the grouped data.
```

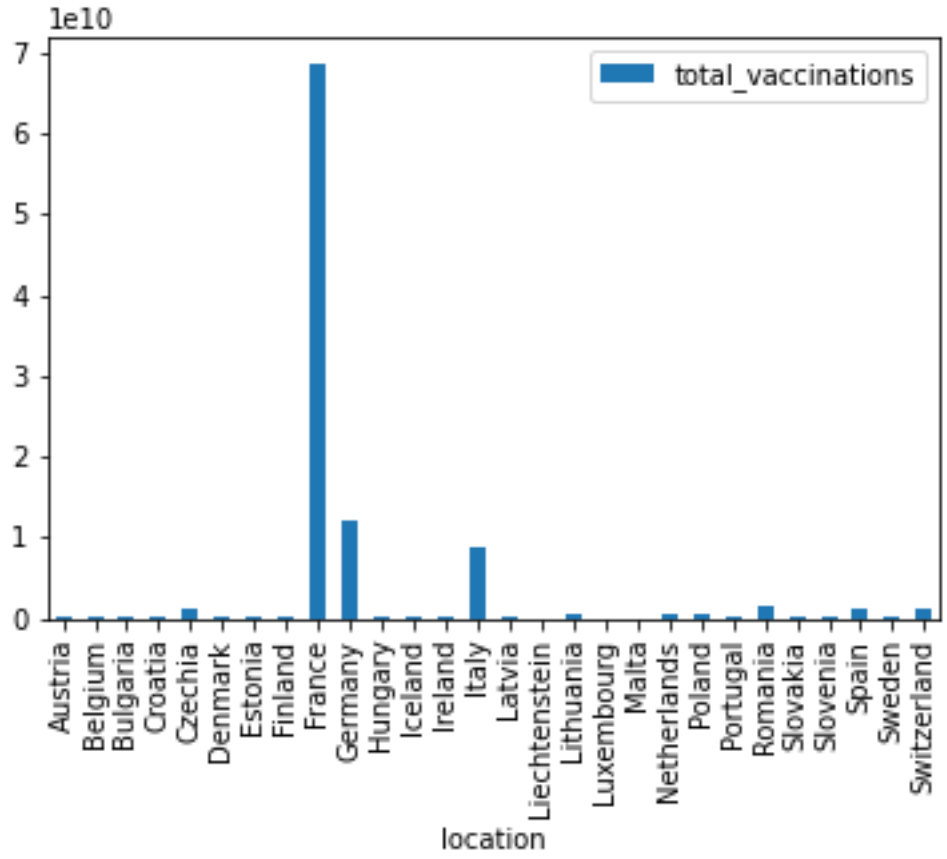


From the above statistics it is clear that Europe has a smaller number of countries with lower count on vaccinations. On that note Moldova, Belarus, Bulgaria, Croatia and Ukraine stand on a lower rate of vaccinations comparatively. Gibraltar has marked the highest count for vaccinations. Belgium, Sweden, Malta, Iceland, Faroe Islands, Portugal, Russia, Wales, Scotland and United Kingdom have had a fair number of vaccinations along with Andorra, Netherlands, Norway, Austria, Poland and Czechia.

Code:

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

```
data_europe_for_Vaccperhunderd = data_europe[["country", "total_vaccinations_per_hundred"]]
data_europe_for_Vaccperhunderd.groupby(['country']).max().plot(kind='bar') # plotting a bar
```

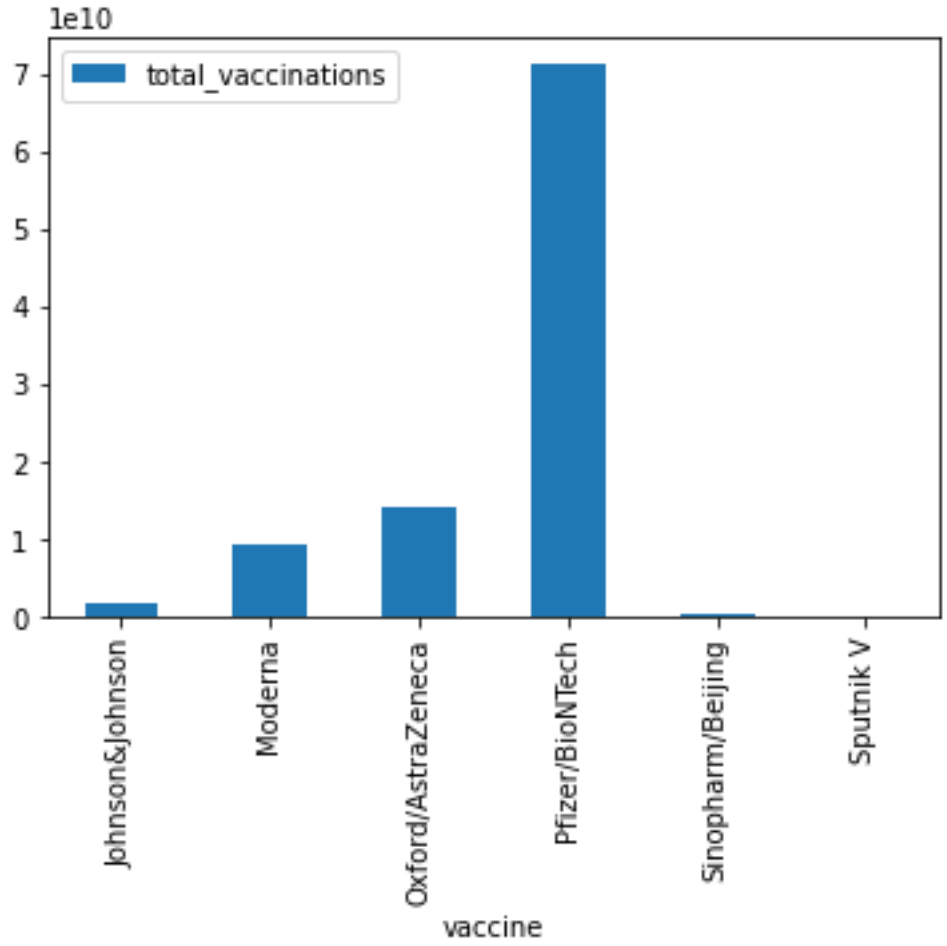


When coming to the total vaccinations, France stands first with more number of vaccinations done followed by Germany and then Italy. All the remaining countries namely Austria, Belgium, Bulgaria, Croatia, Czechia, Denmark, Estonia, Finland, Hungary, Iceland, Ireland, Latvia, Liechtenstein, Lithuania, Luxembourg, Malta, Netherlands, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden and Switzerland were almost approximately equal and the vaccinations done are also very less.

CODE:

```
data_europe_manufacturer_for_grouping = data_europe_manufacturer[["location", "total_vaccinations"]] # grouping the manufacrurer europe data location wise for getting vaccination doses country wise.
data_europe_grouped_manufacturer = (data_europe_manufacturer_for_grouping.groupby(['location']).sum().plot(kind='bar')) # plotting a bar graphs of the grouped data.
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

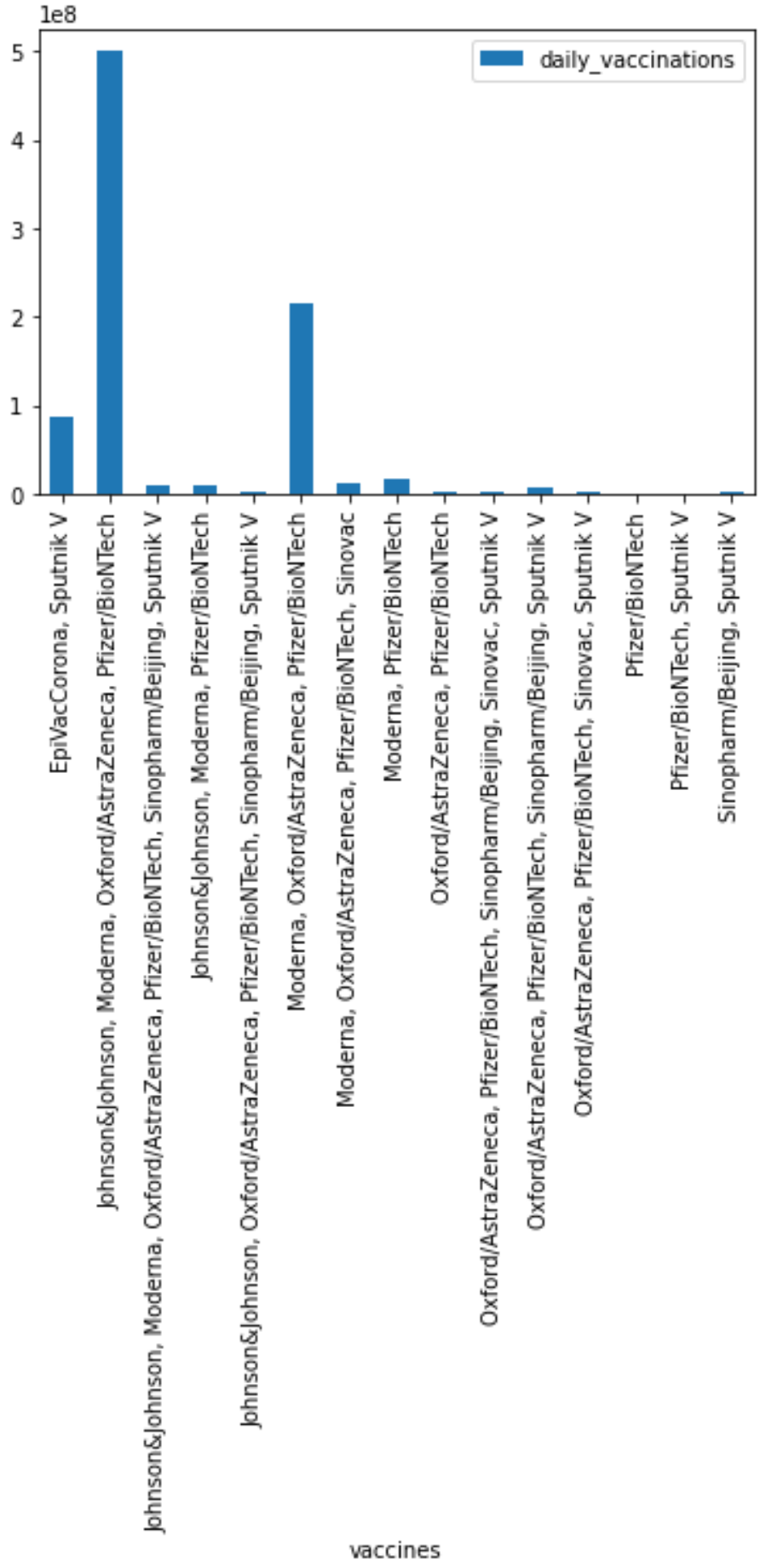


Most of the vaccines that were taken by the people living in the European Countries is supplied by the Company Pfizer/BioNTech. Almost 70% of the people has taken the vaccine produced by Pfizer/BioNTech. Very few of them has taken the shot of the vaccine developed by the companies Oxford/AstraZeneca, Moderna, Johnson&Johnson and almost 1% of the total vaccines were developed by Sinopharm/Beijing.

CODE:

```
# which vaccine companay sold max vaccines in Europe
data_europe_manufacturer_companywise = data_europe_manufacturer[["vaccine", "total_vaccinations"]]
data_europe_grouped_manufacturer_companywise = (data_europe_manufacturer_companywise.groupby(['vaccine']).sum().plot(kind='bar'))
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**



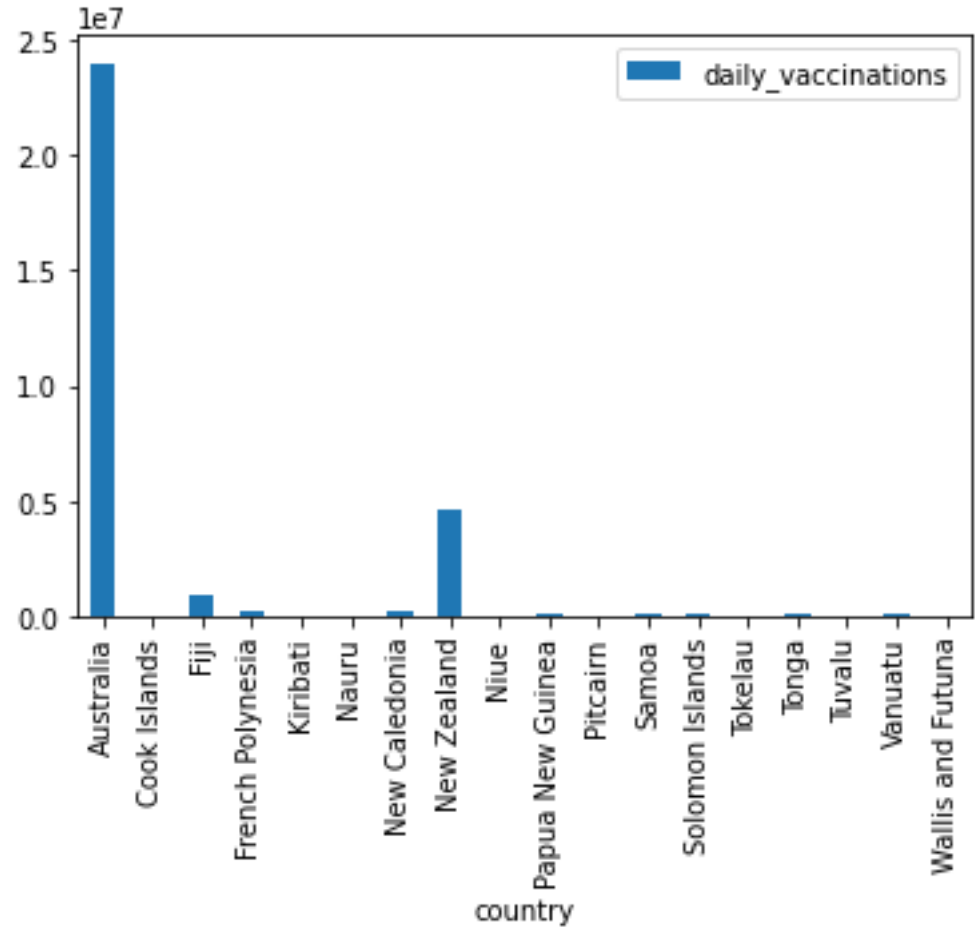
To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

The above graph shows which country uses which vaccine (Data vaccination country wise) according to the countries in Europe.

Code:

```
#Which country uses which vaccine (Data vaccination country wise) according to the countries in Europe.
data_europe_vaccination_company = data_europe[["country", "daily_vaccinations","vaccines"]]
data_europe_group_vacc_company = (data_europe_vaccination_company.groupby(['vaccines']).sum().plot(kind='bar'))
```

Oceania:

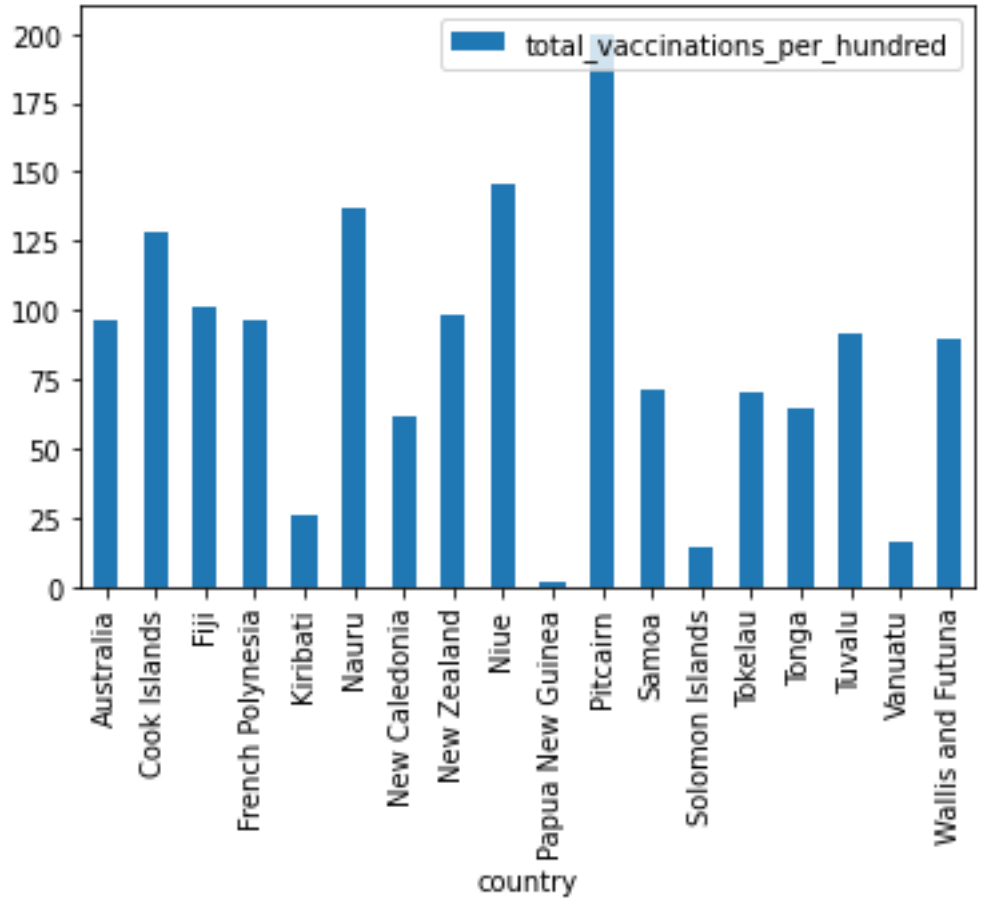


When comes to the continent Oceania, most of the daily vaccinations were done at Australia followed by New Zealand. Countries like French Polynesia, New Caledonia, Papua New Guinea, Samoa, Solomon Islands, Tonga and Vanuatu might have just started their daily vaccinations.

CODE:

```
#Oceania Analysis

data_oceania_for_grouping = data_oceania[["country", "daily_vaccinations"]] #grouping Oceania data country wise and daily vaccinations i.e total vaccinations sum.
data_oceania_grouped = (data_oceania_for_grouping.groupby(['country']).sum().plot(kind='bar'))  # plotting a bar graphs of the grouped data.
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

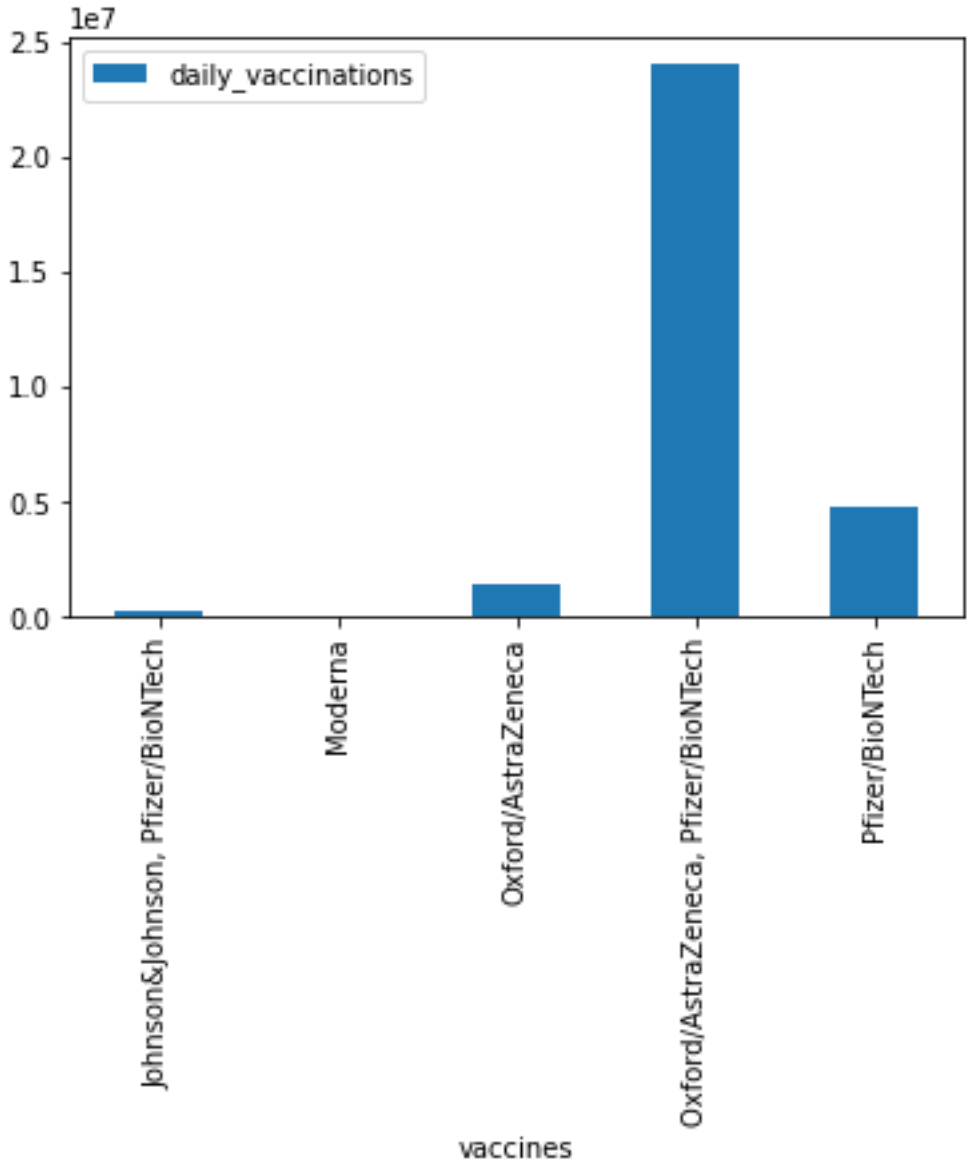


Pitcairn takes up the most number of total vaccinations as per the figures denoted above. After Pitcairn Cook islands, Nauru, Niue and Tuvalu have a greater number comparatively, followed by Australia, Fiji, French Polynesia and Wallis and Futuna.The lowest number of figures are depicted by Papa New Guinea followed by Solomon Islands, Vanuatu and Kiribati.

CODE:

```
data_oceania_for_Vaccperhunderd = data_oceania[["country", "total_vaccinations_per_hundred"]]
data_oceania_for_Vaccperhunderd.groupby(['country']).max().plot(kind='bar') # plotting a bar
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

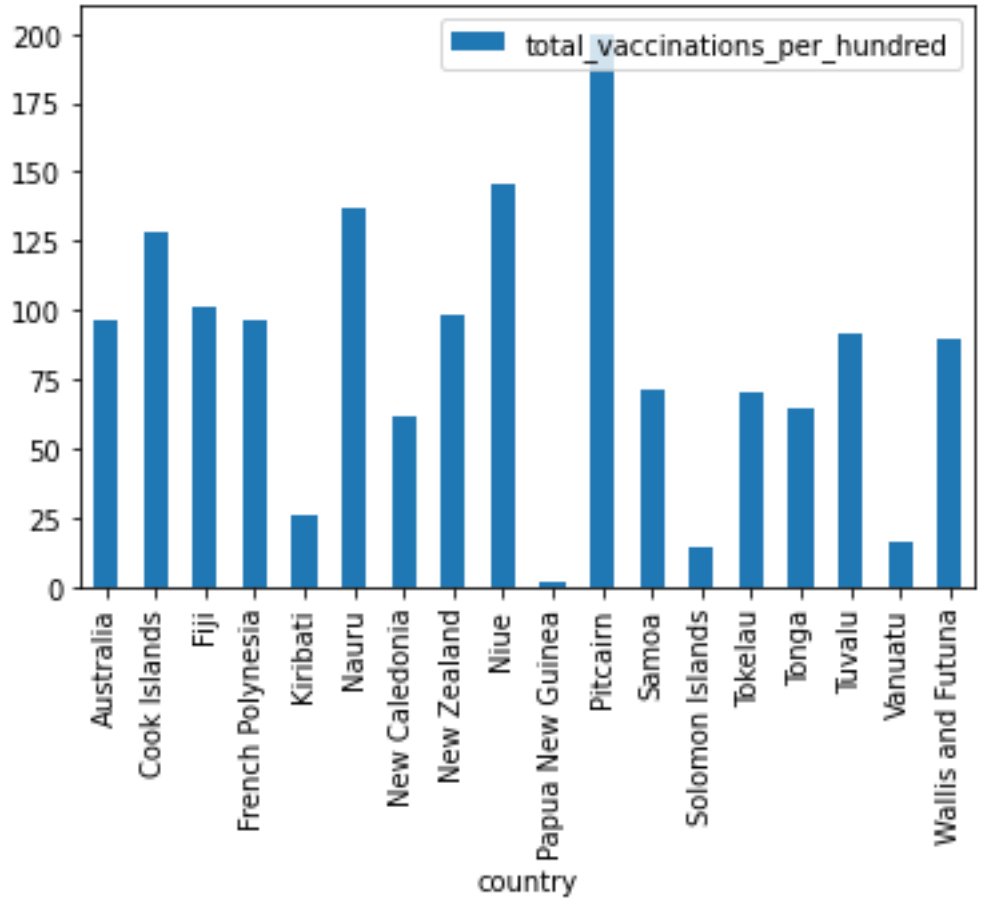


When coming to the daily Vaccinations, most of the Vaccinations were produced by the Companies Oxford/AstraZeneca, Pfizer/BioNTech. No vaccines were produced by the company Moderna.

CODE:

```
#Which country uses which vaccine (Data vaccination country wise) according to the countries in Ocenia.
data_oceania_vaccination_company = data_oceania[["country", "daily_vaccinations","vaccines"]]
data_oceania_group_vacc_company = (data_oceania_vaccination_company.groupby(['vaccines']).sum().plot(kind='bar'))
```

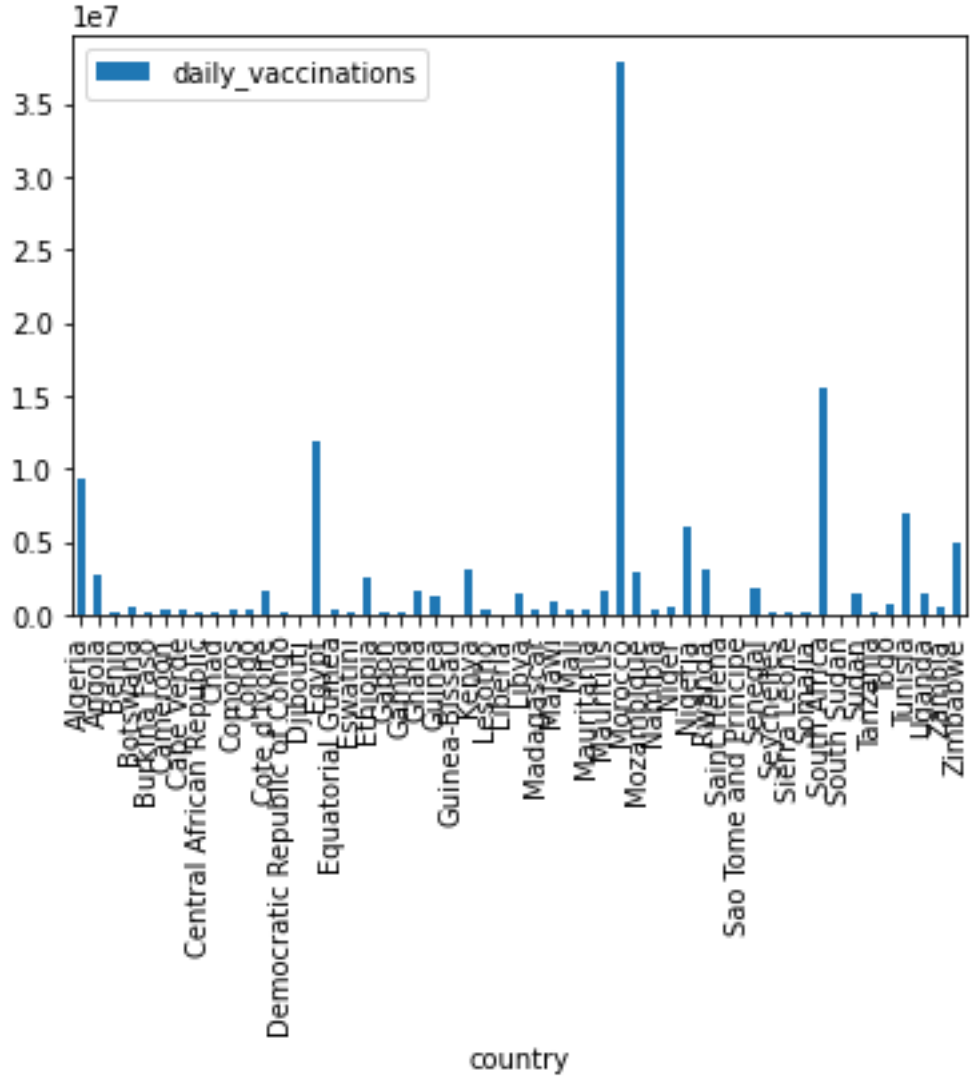
To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**



Of the total Vaccinations done from the continent Oceania, we can see that most vaccinations are done by the country Pitcairn followed by Niue, Nauru, Cook Islands, Fiji, Australia, French Polynesia, New Zealand and least number of vaccinations were taken by the people living in the country Papua New Guinea, Solomon Islands, Vanuatu and Kiribati.

To: Analytic Manager, United Health Care.
From: Aditya Nagori
Subject: COVID- 19 Analysis

AFRICA ANALYSIS

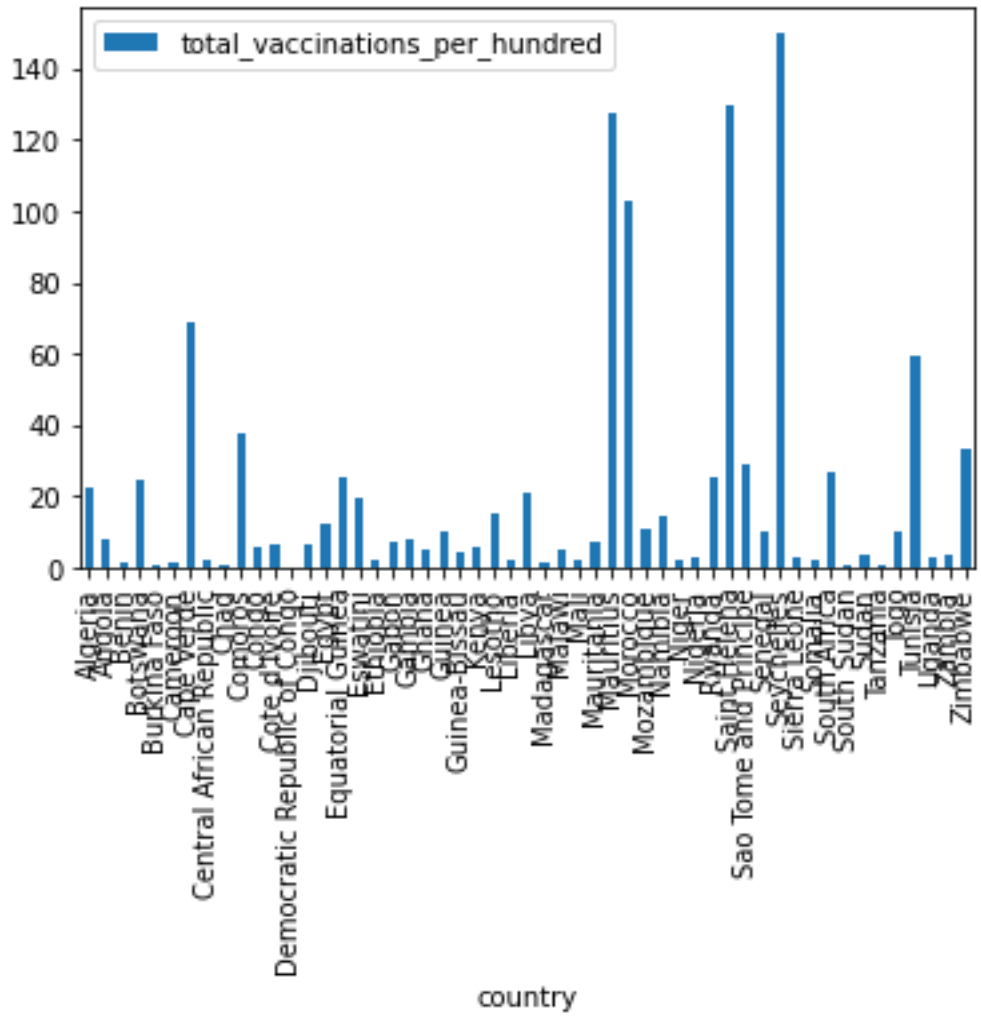


The above graph shows daily vaccinations in African countries.

CODE:

```
#AFRICA Analysis

data_africa_for_grouping = data_africa[["country", "daily_vaccinations"]]  #grouping Africa data country wise and daily vaccinations i.e total vaccinations sum.
data_africa_grouped = (data_africa_for_grouping.groupby(['country']).sum().plot(kind='bar'))  # plotting a bar graphs of the grouped data.
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

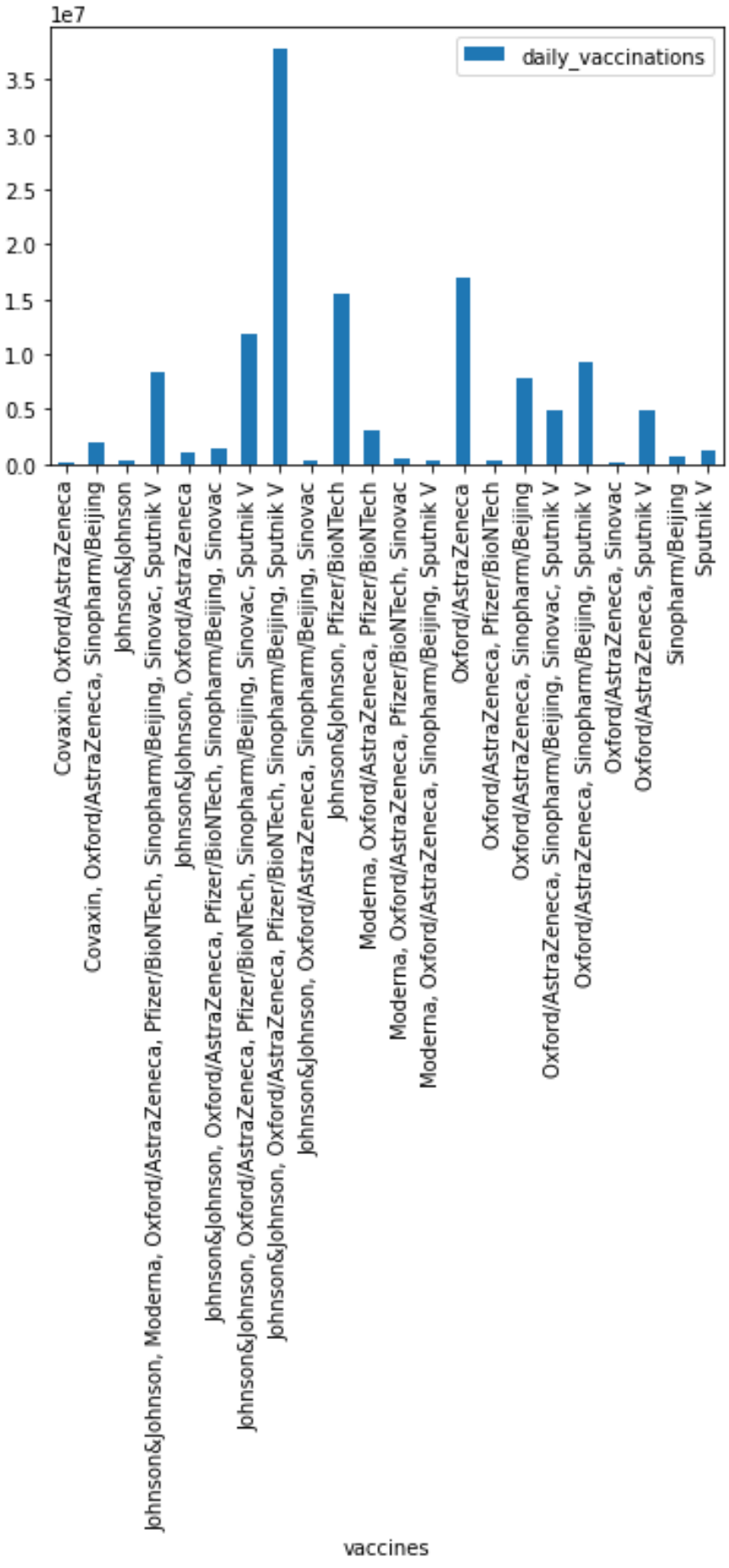


From the above picture of total number of vaccinations done, there are countries with more number of vaccinatons and countries with less number of vaccinations. The countries with lowest rate of vaccinations are Benin, Cameroon, Central African Republic, Comoros, Ethopia, Liberia, Madgascar,Mali, Niger and Nigeria as well. Seychelles takes up the maximum number of vaccinations whereas Saint Helena, Mauritus and Morocco have average rate of vaccinations.

CODE:

```
data_africa_for_Vaccperhunderd = data_africa[["country", "total_vaccinations_per_hundred"]]#
data_africa_for_Vaccperhunderd.groupby(['country']).max().plot(kind='bar') # plotting a bar
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**



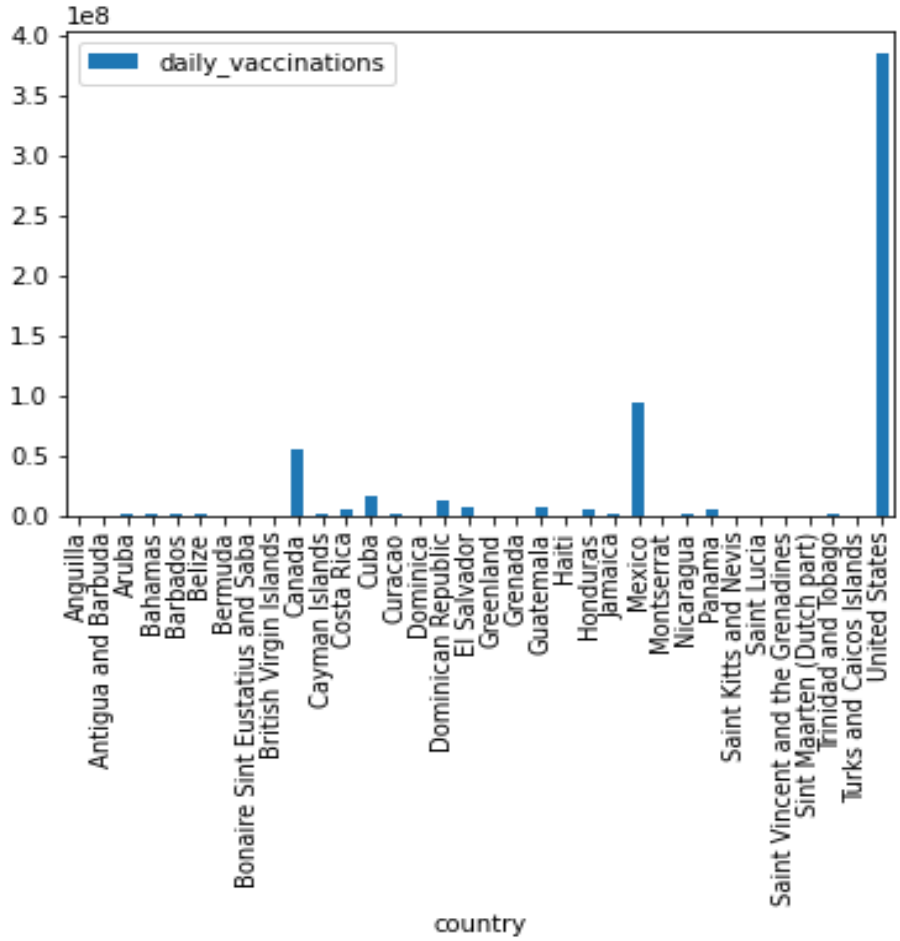
To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

The above graph shows which country uses which vaccine (Data vaccination country wise) according to the countries in Africa.

CODE:

```
#Which country uses which vaccine (Data vaccination country wise) according to the countries in Africa.
data_africa_vaccination_company = data_africa[["country", "daily_vaccinations","vaccines"]]
data_africa_group_vacc_company = (data_africa_vaccination_company.groupby(['vaccines']).sum().plot(kind='bar'))
```

South America

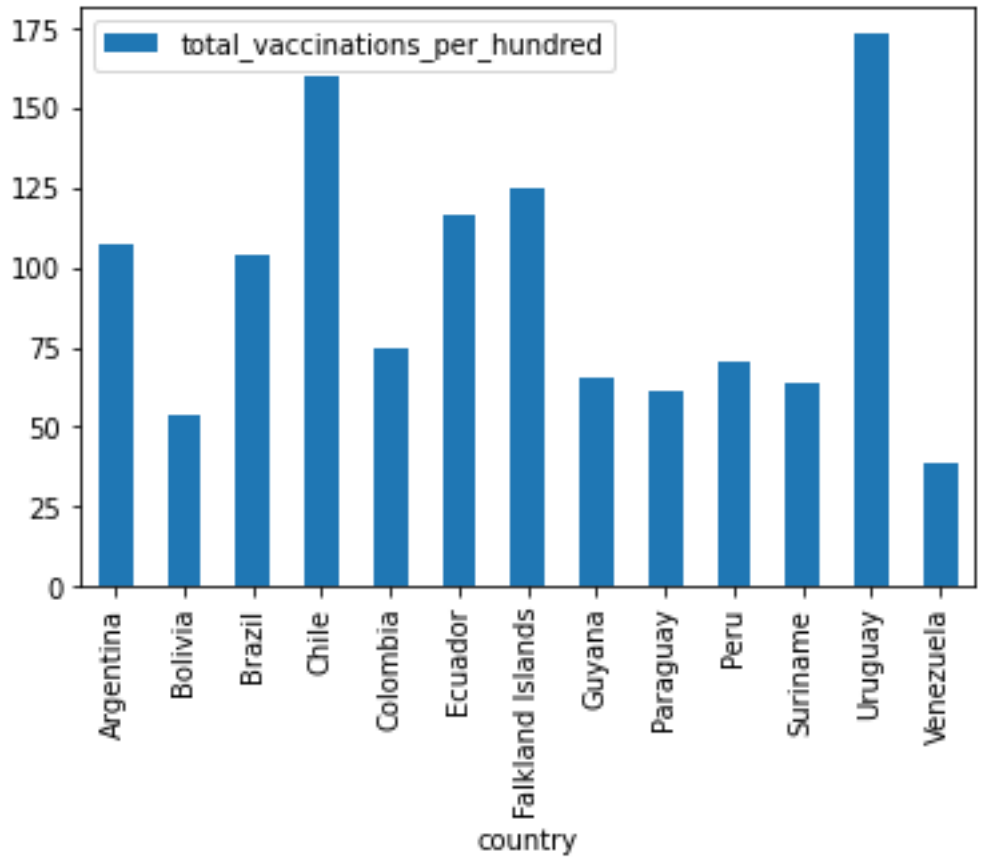


When you observe the daily vaccination status of the countries present in South America Continent, we can see most of the countries has very less progress in their Daily vaccines. Only few countries like United States, Mexico and Canada has some progress.

CODE:

```
#SOUTH AMERICA Analysis

data_south_america_for_grouping = data_south_america[["country", "daily_vaccinations"]]#grouping SOUTH AMERICA  data country wise and daily vaccinations i.e total vaccinations sum.
data_south_america_grouped = (data_south_america_for_grouping.groupby(['country']).sum().plot(kind='bar')) # plotting a bar graphs of the grouped data.
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

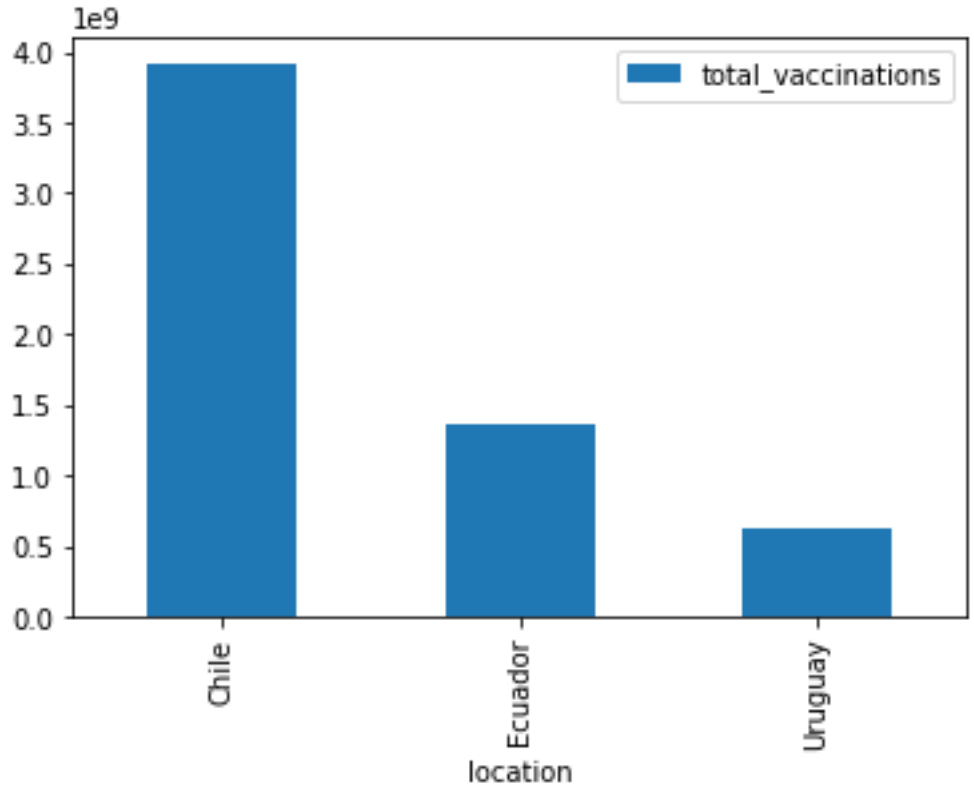


South America has a real fair set of vaccinations with almost no country bearing an extremely lower number of vaccinations. Among the above, Uruguay gets the highest number of vaccinations followed by Chile, Argentina, Ecuador, Falkland Islands. The countries with a mean value of number of vaccinations include Bolivia, Guyana, Columbia, Suriname and Venezuela.

CODE:

```
data_south_america_for_Vaccperhunderd = data_south_america[["country", "total_vaccinations_per_hundred"]]#g
data_south_america_for_Vaccperhunderd.groupby(['country']).max().plot(kind='bar')# plotting a bar graphs of
```

To: Analytic Manager, United Health Care.
From: Aditya Nagori
Subject: COVID- 19 Analysis

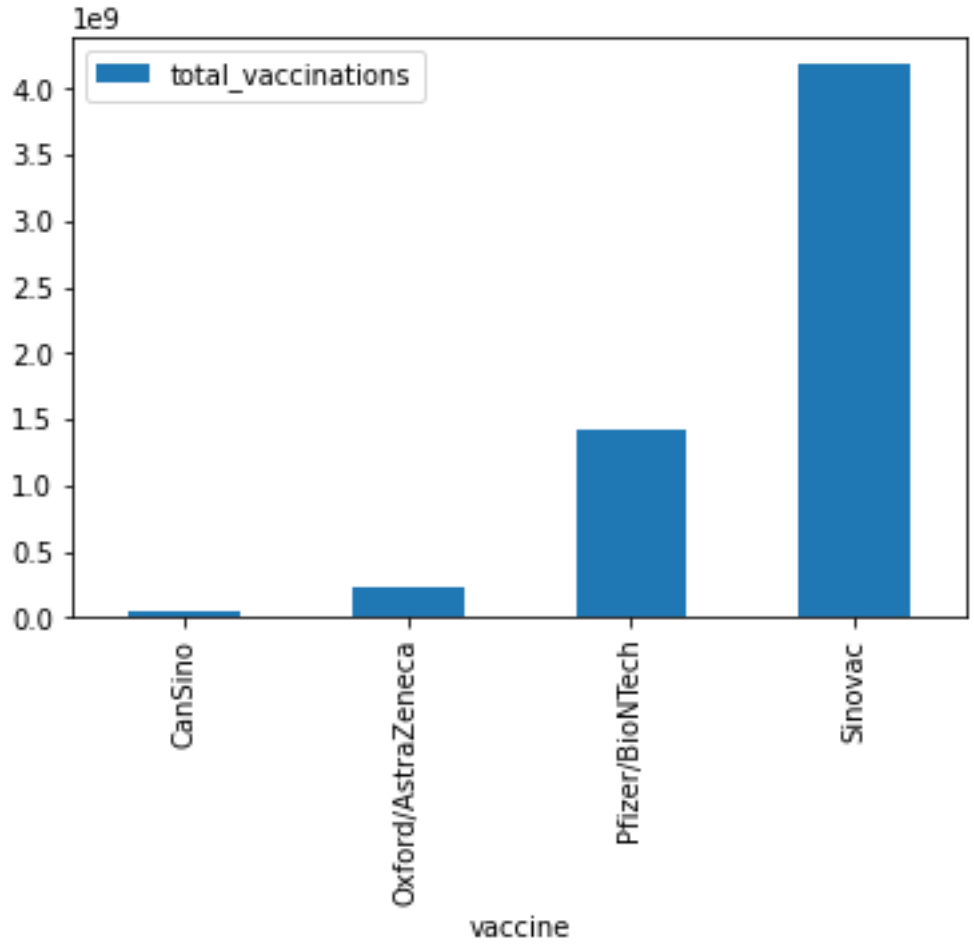


The above graph shows which SOUTH AMERICAN continent country received max doses from manufacture.

CODE:

```
# which SOUTH AMERICAN continent country recieved max doses from manufacture.
data_south_america_manufacturer_for_grouping = data_south_america_manufacturer[["location", "total_vaccinations"]]
data_south_america_grouped_manufacturer = (data_south_america_manufacturer_for_grouping.groupby(['location']).sum().plot(kind='bar'))
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

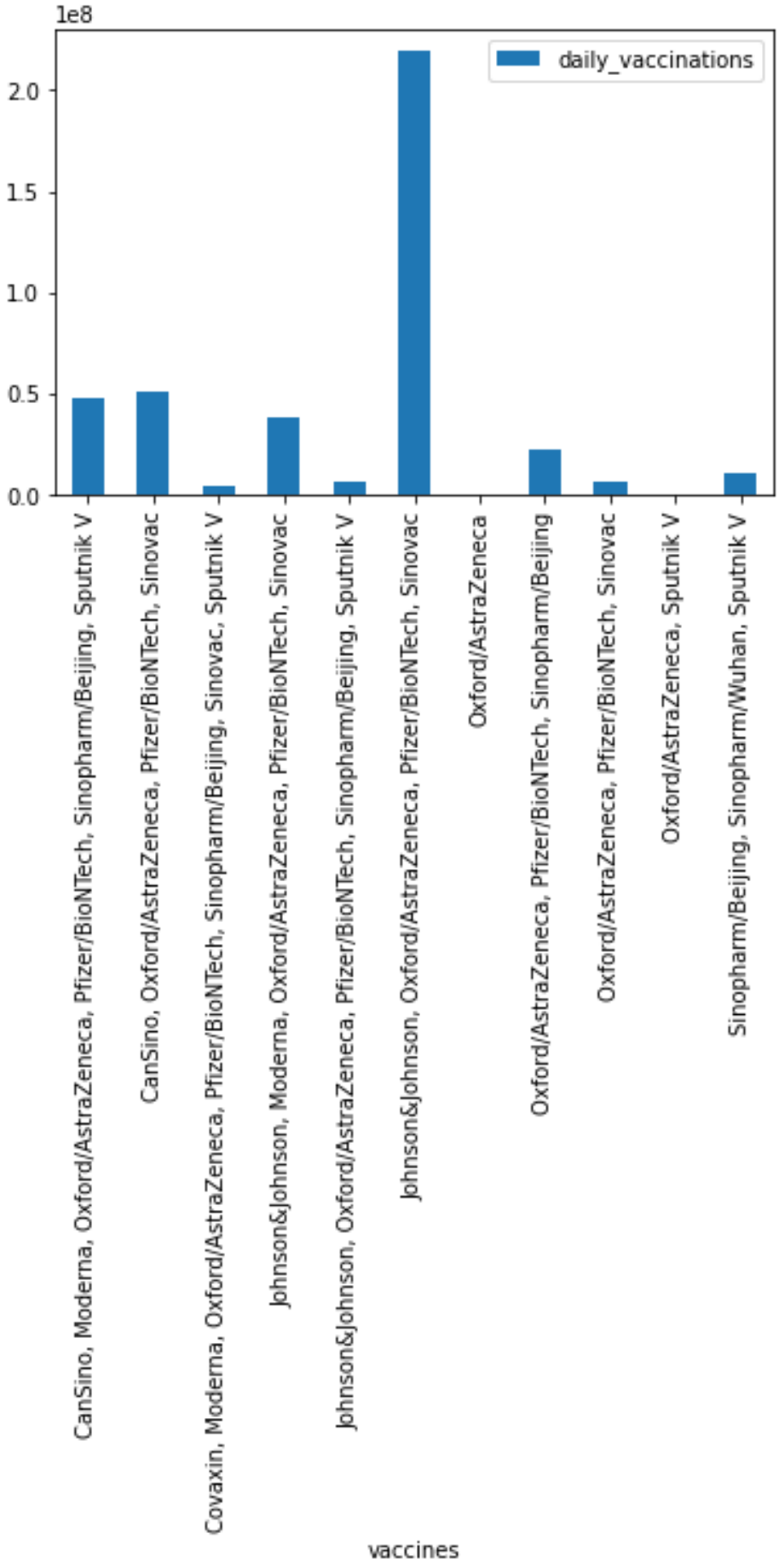


Sinovac has produce the most number of Vaccinations followed by Pfizer/BioNTech, Oxford/AstraZeneca and CanSino.

CODE:

```
# which vaccine companay sold max vaccines in South America
data_south_america_manufacturer_companywise = data_south_america_manufacturer[["vaccine", "total_vaccinations"]]
data_south_america_grouped_manufacturer_companywise = (data_south_america_manufacturer_companywise.groupby(['vaccine']).sum().plot(kind='bar'))
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**



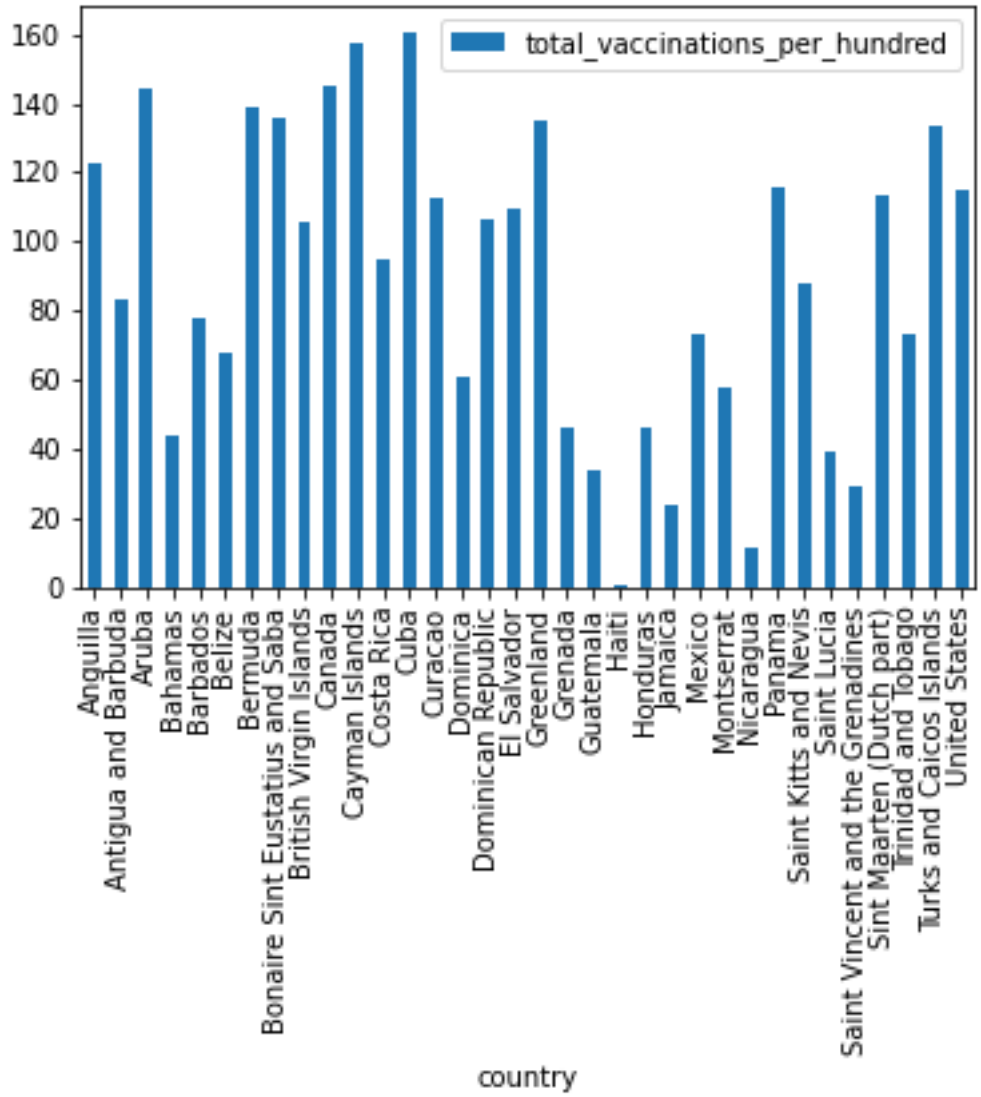
To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

The above graph shows which country uses which vaccine (Data vaccination country wise) according to the countries in South America.

Code:

```
#Which country uses which vaccine (Data vaccination country wise) according to the countries in South America.
data_south_america_vaccination_company = data_south_america[["country", "daily_vaccinations","vaccines"]]
data_south_america_group_vacc_company = (data_south_america_vaccination_company.groupby(['vaccines']).sum().plot(kind='bar'))
```

NORTH AMERICA

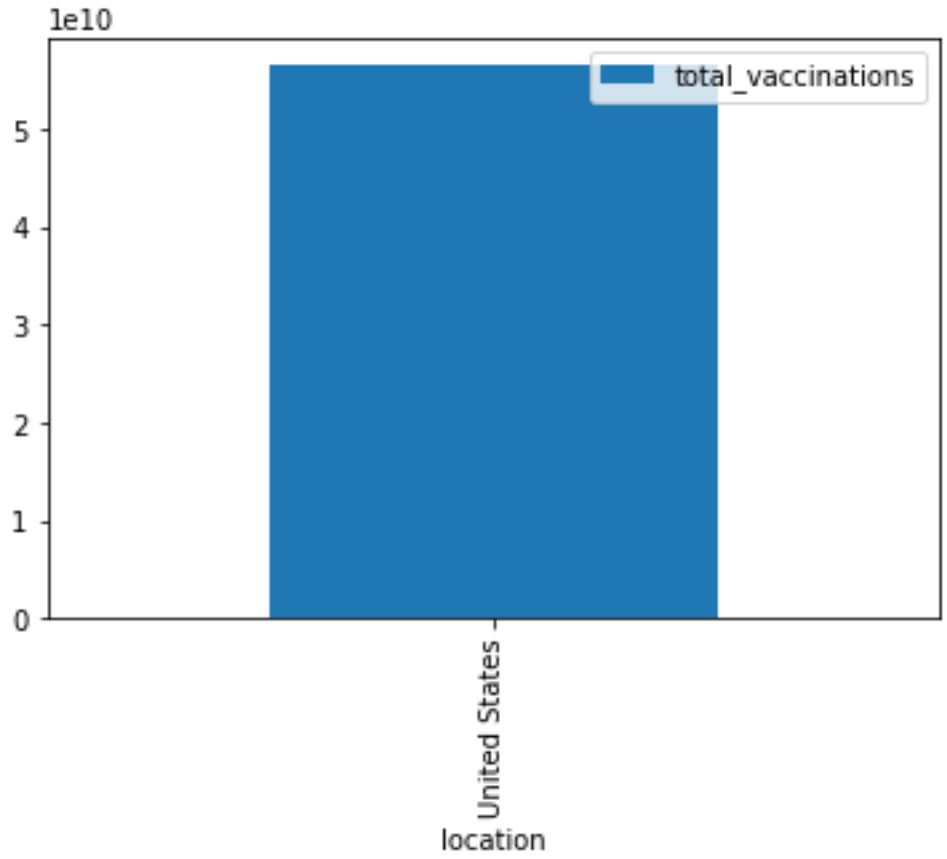


In the North American figures of vaccinations, Cuba stands the highest in the total number of vaccinations followed by Aruba, Cayman Islands, Panama, St. Maarten, Turks and Caico Islands, United States, Anguilla and Canada. Among the average count of vaccinations Bahamas, Dominica, Honduras, St. Lucia, Trinidad, and Tobago mark their number. Haiti followed by Nicaragua take up the least count of vaccinations

To: Analytic Manager, United Health Care.
From: Aditya Nagori
Subject: COVID- 19 Analysis

CODE:

```
data_north_america_for_Vaccperhunderd = data_north_america[["country", "total_vaccinations_per_hundred"]]
data_north_america_for_Vaccperhunderd.groupby(['country']).max().plot(kind='bar')# plotting a bar graphs o
```

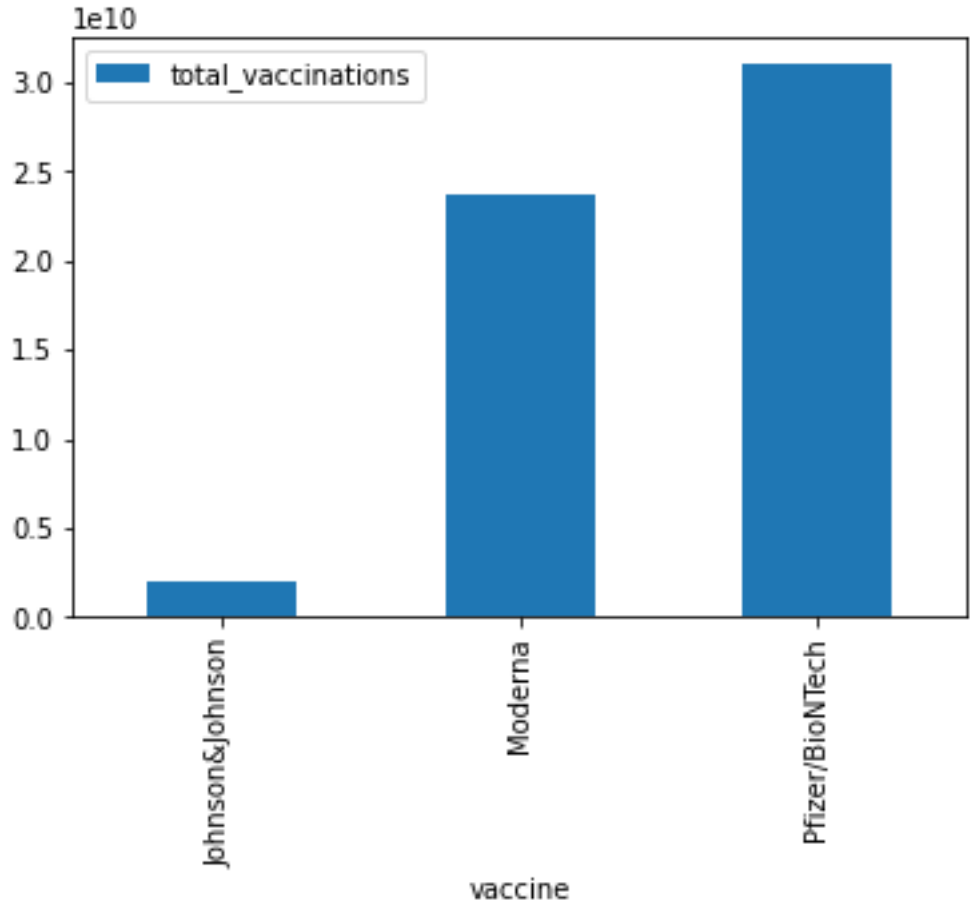


The above graph shows which North American continent country received max doses from manufacture.

CODE:

```
# which North AMERICAN continent country recieved max doses from manufacture.
data_north_america_manufacturer_for_grouping = data_north_america_manufacturer[["location", "total_vaccinations"]]
data_north_america_grouped_manufacturer = (data_north_america_manufacturer_for_grouping.groupby(['location']).sum().plot(kind='bar'))
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

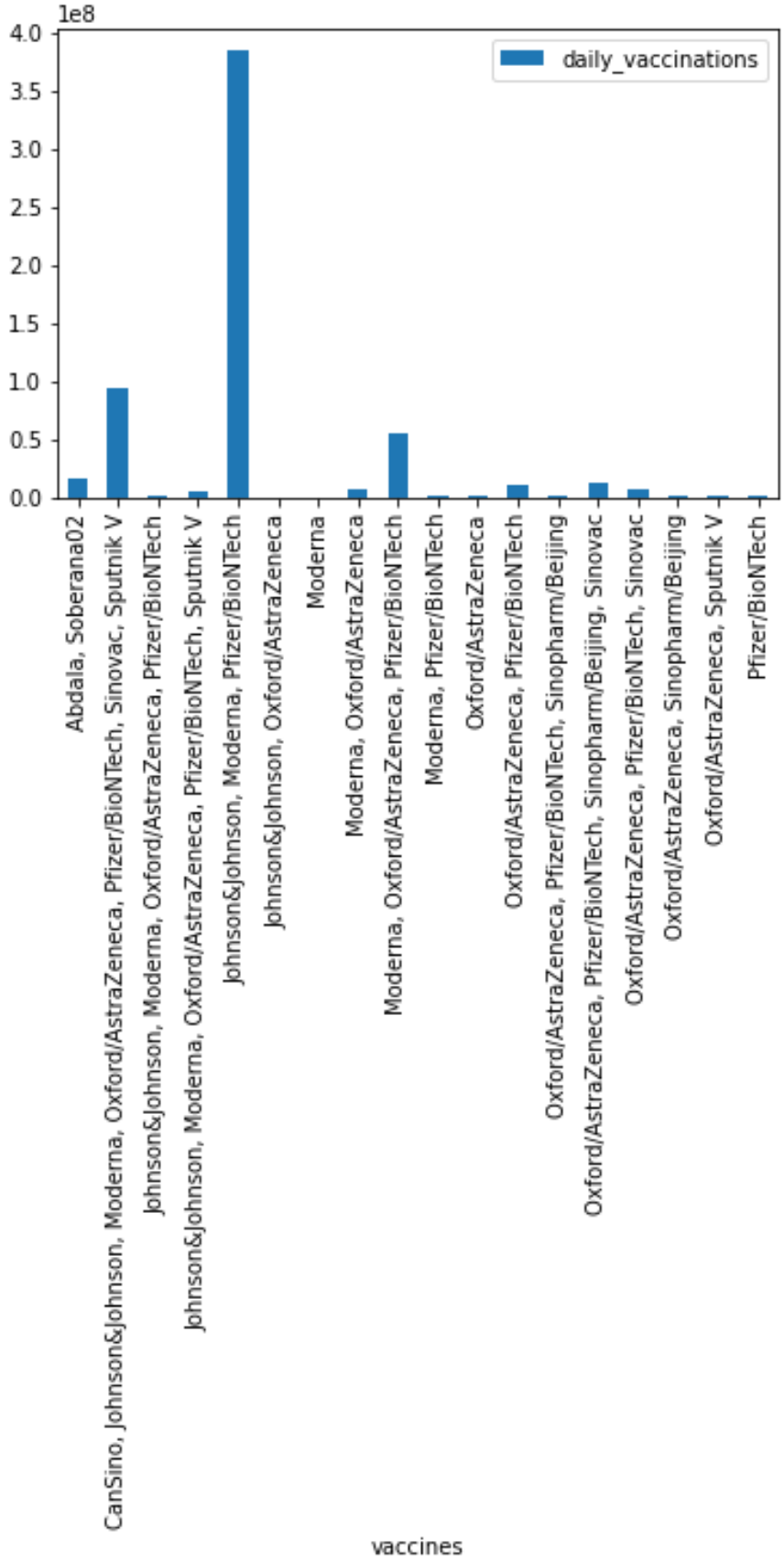


Of all the Vaccines produced, Pfizer/BioNTech company produced the most number of vaccines. Moderna also produced about 40% of the total vaccines. The least amount of vaccines were produced by Johnson&Johnson.

CODE:

```
# which vaccine companay sold max vaccines in South America
data_north_america_manufacturer_companywise = data_north_america_manufacturer[["vaccine", "total_vaccinations"]]
data_north_america_grouped_manufacturer_companywise = (data_north_america_manufacturer_companywise.groupby(['vaccine']).sum().plot(kind='bar'))
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**



To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

The above graph shows which country uses which vaccine (Data vaccination country wise) according to the countries in North America.

CODE:

```
#Which country uses which vaccine (Data vaccination country wise) according to the countries in North America.
data_north_america_vaccination_company = data_north_america[["country", "daily_vaccinations","vaccines"]]
data_north_america_group_vacc_company = (data_north_america_vaccination_company.groupby(['vaccines']).sum().plot(kind='bar'))
```

Sentiment Analysis

- Pre-processing Data

Code:

```
# Tweetsdata Sentiment analysis


#Preprossing of our data & Generating reports.
tweet_df=  pd.read_csv("vaccination_all_tweets.csv")
report_tweet = ProfileReport(tweet_df)
report_tweet.to_file("vaccination_all_tweets.html")
tweet_df.describe(include='all')
tweet_df.drop_duplicates()
tweet_df.info()

def clean(text):
# Removes all special characters and numericals leaving the alphabets
    text = re.sub('[^A-Za-z]+', ' ', text)
    return text

# Cleaning the text in the review column
tweet_df['Cleaned Reviews'] = tweet_df['text'].apply(clean)
tweet_df.head(15)

# POS tagger dictionary
pos_dict = {'J':wordnet.ADJ, 'V':wordnet.VERB, 'N':wordnet.NOUN, 'R':wordnet.ADV}
def token_stop_pos(text):
    tags = pos_tag(word_tokenize(text))
    newlist = []
    for word, tag in tags:
        if word.lower() not in set(stopwords.words('english')):
            newlist.append(tuple([word, pos_dict.get(tag[0])]))
    return newlist

tweet_df['POS tagged'] = tweet_df['Cleaned Reviews'].apply(token_stop_pos)
tweet_df.head()


from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
def lemmatize(pos_data):
    lemma_rew = " "
    for word, pos in pos_data:
        if not pos:
            lemma = word
            lemma_rew = lemma_rew + " " + lemma
        else:
            lemma = wordnet_lemmatizer.lemmatize(word, pos=pos)
            lemma_rew = lemma_rew + " " + lemma
    return lemma_rew

tweet_df['Lemma'] = tweet_df['POS tagged'].apply(lemmatize)
tweet_df.head()
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

- Sentiment Analysis using TextBlob Method.
  Code:

```
#Sentiment Analysis using TextBlob:

from textblob import TextBlob
# function to calculate subjectivity
def getSubjectivity(review):
    return TextBlob(review).sentiment.subjectivity
    # function to calculate polarity
def getPolarity(review):
        return TextBlob(review).sentiment.polarity

# function to analyze the reviews
def analysis(score):
    if score < 0:
        return 'Negative'
    elif score == 0:
        return 'Neutral'
    else:
        return 'Positive'
fin_data = pd.DataFrame(tweet_df[['text', 'Lemma']])
# fin_data['Subjectivity'] = fin_data['Lemma'].apply(getSubjectivity)
fin_data['Polarity'] = fin_data['Lemma'].apply(getPolarity)
fin_data['Analysis'] = fin_data['Polarity'].apply(analysis)
fin_data.head()
tb_counts = fin_data.Analysis.value_counts()
print(tb_counts)
```

- Sentiment Analysis using VADER method.
  Code:

```
#Sentiment Analysis using VADER

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyzer = SentimentIntensityAnalyzer()
# function to calculate vader sentiment
def vadersentimentanalysis(review):
    vs = analyzer.polarity_scores(review)
    return vs['compound']
fin_data['Vader Sentiment'] = fin_data['Lemma'].apply(vadersentimentanalysis)
# function to analyse
def vader_analysis(compound):
    if compound >= 0.5:
        return 'Positive'
    elif compound <= -0.5 :
        return 'Negative'
    else:
        return 'Neutral'
fin_data['Vader Analysis'] = fin_data['Vader Sentiment'].apply(vader_analysis)
fin_data.head()
vader_counts = fin_data['Vader Analysis'].value_counts()
print(vader_counts)
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

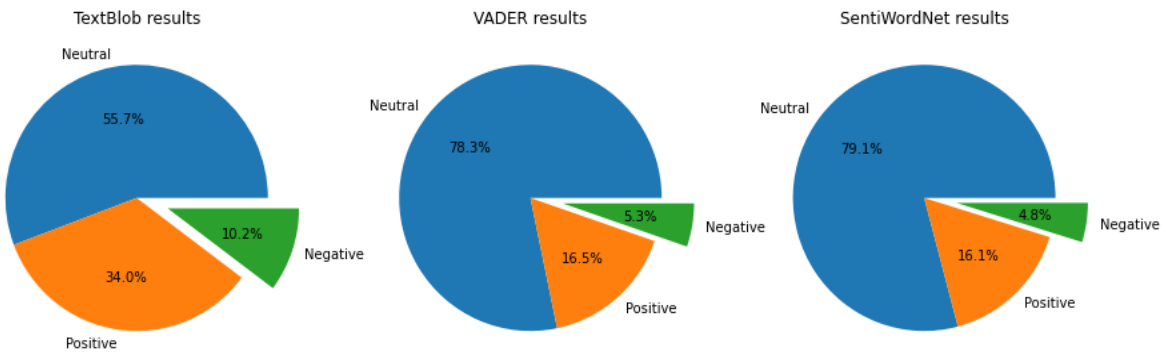
- Sentiment Analysis using SentiWordNet method.
  Code:

```
#Sentiment Analysis using SentiWordNet

nltk.download('sentiwordnet')
from nltk.corpus import sentiwordnet as swn
def sentiwordnetanalysis(pos_data):
    sentiment = 0
    tokens_count = 0
    for word, pos in pos_data:
        if not pos:
            continue
        lemma = wordnet_lemmatizer.lemmatize(word, pos=pos)
        if not lemma:
            continue
        synsets = wordnet.synsets(lemma, pos=pos)
        if not synsets:
            continue
        # Take the first sense, the most common
        synset = synsets[0]
        swn_synset = swn.senti_synset(synset.name())
        sentiment += swn_synset.pos_score() - swn_synset.neg_score()
        tokens_count += 1
        # print(swn_synset.pos_score(),swn_synset.neg_score(),swn_synset.obj_score())
    if not tokens_count:
        return 0
    if sentiment>0:
        return "Positive"
    if sentiment==0:
        return "Neutral"
    else:
        return "Negative"

fin_data['SWN analysis'] = tweet_df['POS tagged'].apply(sentiwordnetanalysis)
fin_data.head()
swn_counts= fin_data['SWN analysis'].value_counts()
print(swn_counts)
```

- Output of all 3 sentiment analysis using pie chart.
  Code:

```
plt.figure(figsize=(15,7))
plt.subplot(1,3,1)
plt.title("TextBlob results")
plt.pie(tb_counts.values, labels = tb_counts.index, explode = (0, 0, 0.25), autopct='%1.1f%%', shadow=False)
plt.subplot(1,3,2)
plt.title("VADER results")
plt.pie(vader_counts.values, labels = vader_counts.index, explode = (0, 0, 0.25), autopct='%1.1f%%', shadow=False)
plt.subplot(1,3,3)
plt.title("SentiWordNet results")
plt.pie(swn_counts.values, labels = swn_counts.index, explode = (0, 0, 0.25), autopct='%1.1f%%', shadow=False)
plt.savefig("Sentiment_Analysis.png")
```

  Output:



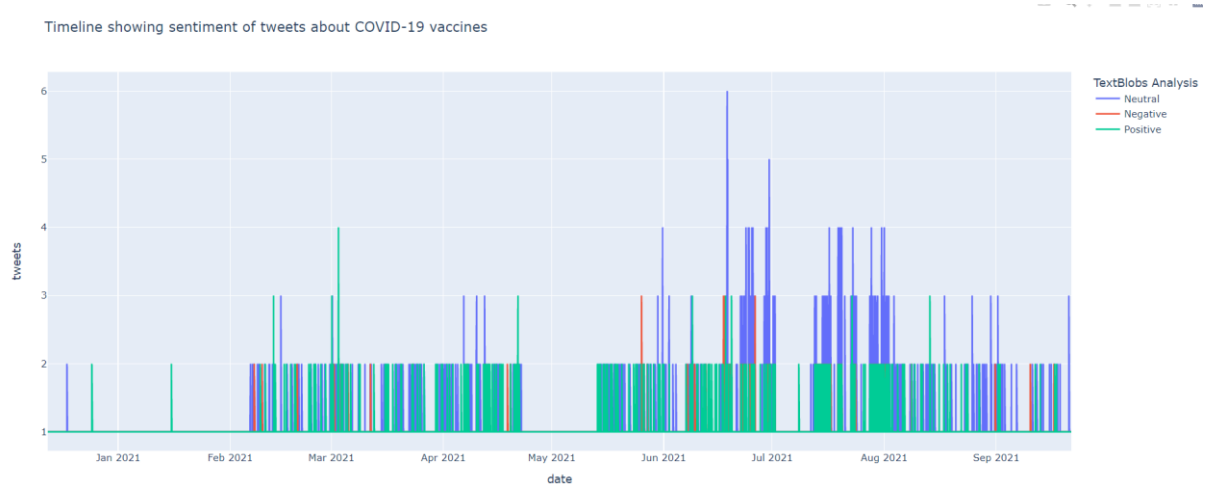
To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

- Timeline of sentiments of tweets about vaccines using TextBlob's analysis.

Code:

```
#Timeline of sentiments of tweets about vaccines using TextBlob's analysis
tweet_df['TextBlobs Analysis'] = fin_data['Analysis']
today = pd.Timestamp.today().date()
tweet_df = tweet_df[tweet_df['date']!=today]

# Get counts of number of tweets by sentiment for each date
timeline = tweet_df.groupby(['date', 'TextBlobs Analysis']).agg(**{'tweets': ('id', 'count')}).reset_index().dropna()

# Plot results
fig_timeline = ex.line(timeline, x='date', y='tweets', color='TextBlobs Analysis', category_orders={'TextBlobs Analysis': ['Neutral', 'Negative', 'Positive']},
             title='Timeline showing sentiment of tweets about COVID-19 vaccines', render_mode="SVG")
plot(fig_timeline)
```

Output:



The above graph shows you the TextBlob Analysis of the Timeline showing sentiment of tweets about COVID-19 vaccine from January of 2021 till September of 2021. The lines which are Green indicates the tweets as Positive, the lines which are Orange coloured shows the tweets are Negative and the lines which are coloured Blue are Neutral tweets.

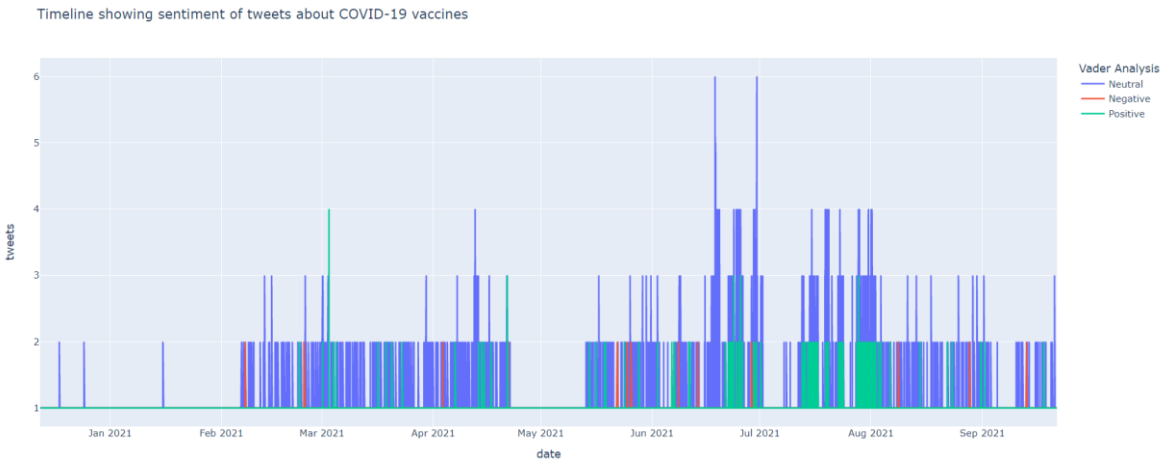
- Timeline of sentiments of tweets about vaccines using VADER analysis.

Code:

```
#Timeline of sentiments of tweets about vaccines using VADER analysis
tweet_df['Vader Analysis'] = fin_data['Vader Analysis']
today = pd.Timestamp.today().date()
tweet_df = tweet_df[tweet_df['date']!=today]

# Get counts of number of tweets by sentiment for each date
timeline = tweet_df.groupby(['date', 'Vader Analysis']).agg(**{'tweets': ('id', 'count')}).reset_index().dropna()

# Plot results
fig_timeline = ex.line(timeline, x='date', y='tweets', color='Vader Analysis', category_orders={'Vader Analysis': ['Neutral', 'Negative', 'Positive']},
             title='Timeline showing sentiment of tweets about COVID-19 vaccines', render_mode="SVG")
plot(fig_timeline)
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

Output:



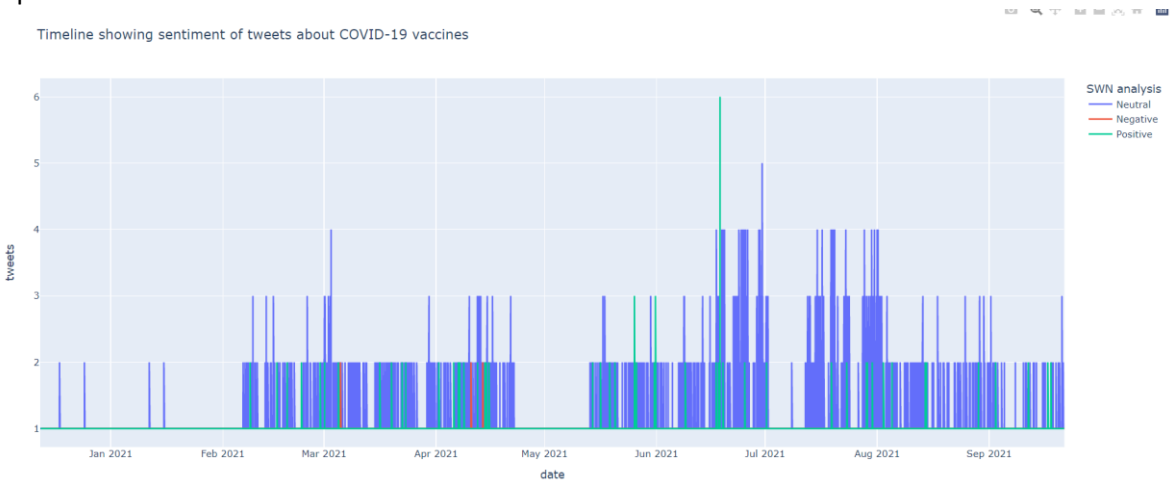
The above graph shows you the SentiWordNet Analysis of the Timeline showing sentiment of tweets about COVID-19 vaccine from January of 2021 till September of 2021. The lines which are Green indicates the tweets as Positive, the lines which are Orange coloured shows the tweets are Negative and the lines which are coloured Blue are Neutral tweets.

- Timeline of sentiments of tweets about vaccines using SentiWordNet analysis.

Code:

```
#Timeline of sentiments of tweets about vaccines using SentiWordNet analysis
tweet_df['SWN analysis'] = fin_data['SWN analysis']
today = pd.Timestamp.today().date()
tweet_df = tweet_df[tweet_df['date']!=today]

# Get counts of number of tweets by sentiment for each date
timeline = tweet_df.groupby(['date', 'SWN analysis']).agg(**{'tweets': ('id', 'count')}).reset_index().dropna()

# Plot results
fig_timeline = ex.line(timeline, x='date', y='tweets', color='SWN analysis', category_orders={'SWN analysis': ['Neutral', 'Negative', 'Positive']},
             title='Timeline showing sentiment of tweets about COVID-19 vaccines', render_mode="SVG")
plot(fig_timeline)
```

Output:



To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

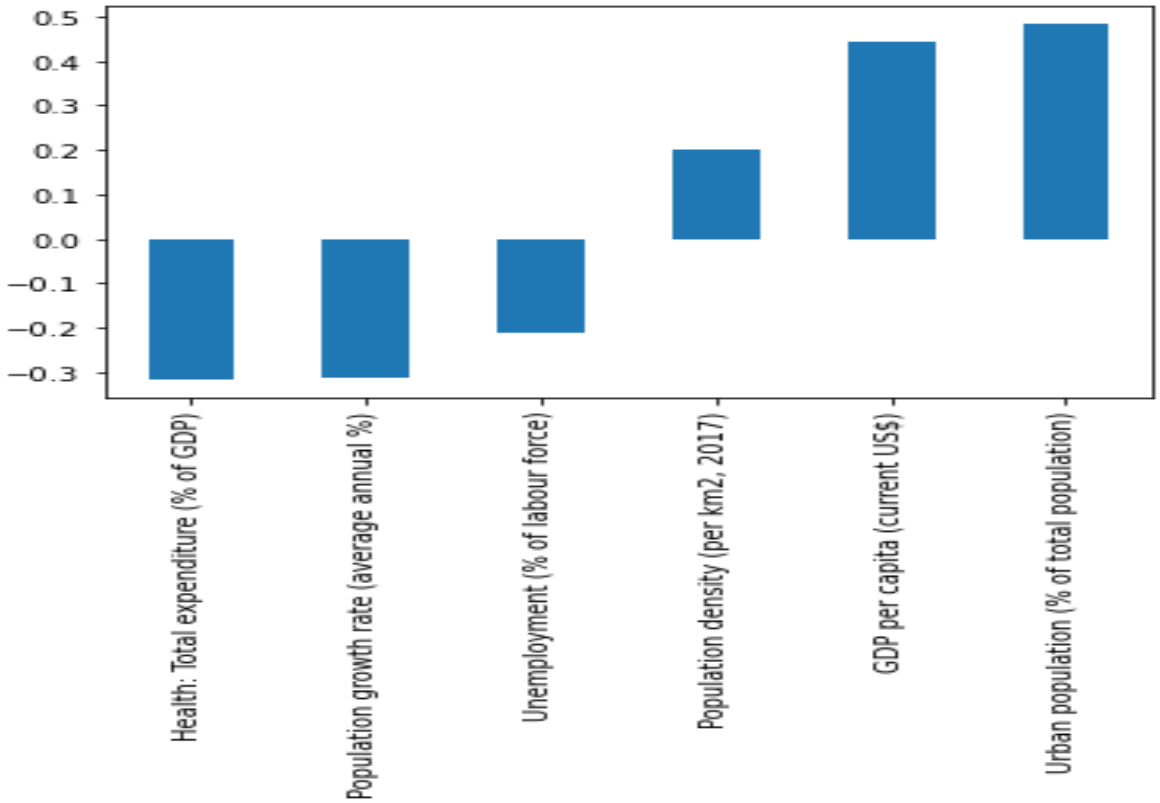
The above graph shows you the VADER Analysis of the Timeline showing sentiment of tweets about COVID-19 vaccine from January of 2021 till September of 2021. The lines which are Green indicates the tweets as Positive, the lines which are Orange coloured shows the tweets are Negative and the lines which are coloured Blue are Neutral tweets.

Demographics and vaccinations Correlations

Code:

```
#Demographies and vaccinations Correlations

df_CV_demography = data_CV.copy()
data_demograpy = pd.read_csv("country_profile_variables.csv")

vaccine=df_CV_demography.groupby('country')['people_vaccinated_per_hundred']
vaccine=pd.DataFrame(vaccine.mean('people_vaccinated_per_hundred'))
vaccine.fillna(value=0,inplace=True)

Related_data=data_demograpy.merge(vaccine,left_on='country',right_index=True).reset_index(drop=True)
Related_data.head()
Related_data.columns
Related_data.info()

Related_data=Related_data[['Population density (per km2, 2017)','GDP per capita (current US$)','Unemployment (% of labour force)',
                           'Population growth rate (average annual %)','Health: Total expenditure (% of GDP)',
                           'Urban population (% of total population)','people_vaccinated_per_hundred']]

Related_data['Unemployment (% of labour force)'] = Related_data['Unemployment (% of labour force)'].apply(str).str.replace('...','0')
Related_data['Unemployment (% of labour force)'] = Related_data['Unemployment (% of labour force)'].apply(str).str.replace('','0')
Related_data['Unemployment (% of labour force)'] = Related_data['Unemployment (% of labour force)'].astype(float)
Related_data['Population growth rate (average annual %)'] = Related_data['Population growth rate (average annual %)'].apply(str).str.replace('~0.0','0')
Related_data['Population growth rate (average annual %)'][43]=0.0
Related_data['Population growth rate (average annual %)']=Related_data['Population growth rate (average annual %)'].astype(float).astype(float)
Related_data.corr()[:-1]['people_vaccinated_per_hundred'].sort_values().plot(kind='bar')
sns.pairplot(Related_data,kind='reg')
```

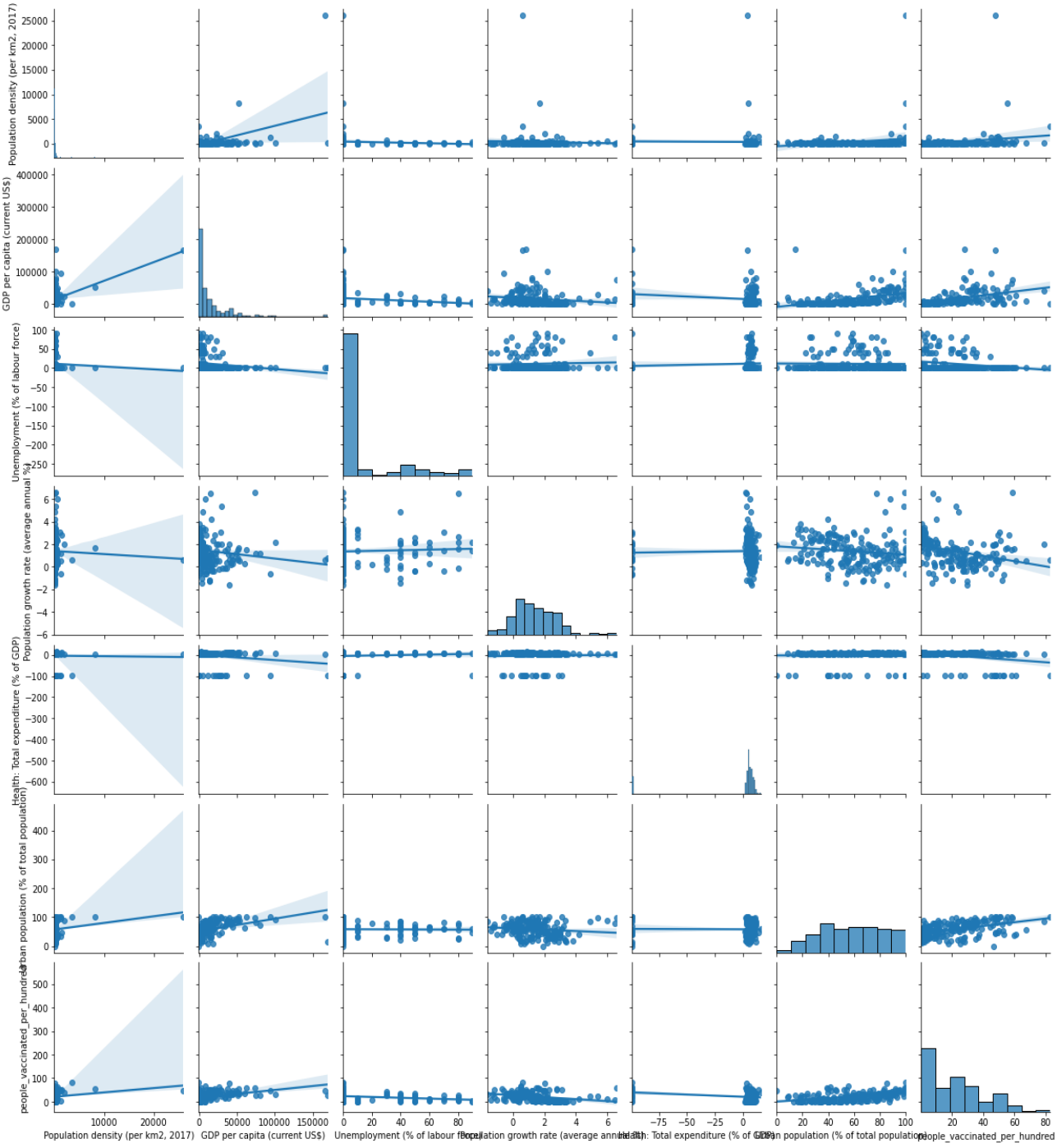
Output:



To: **Analytic Manager, United Health Care.**

From: **Aditya Nagori**

Subject: **COVID- 19 Analysis**



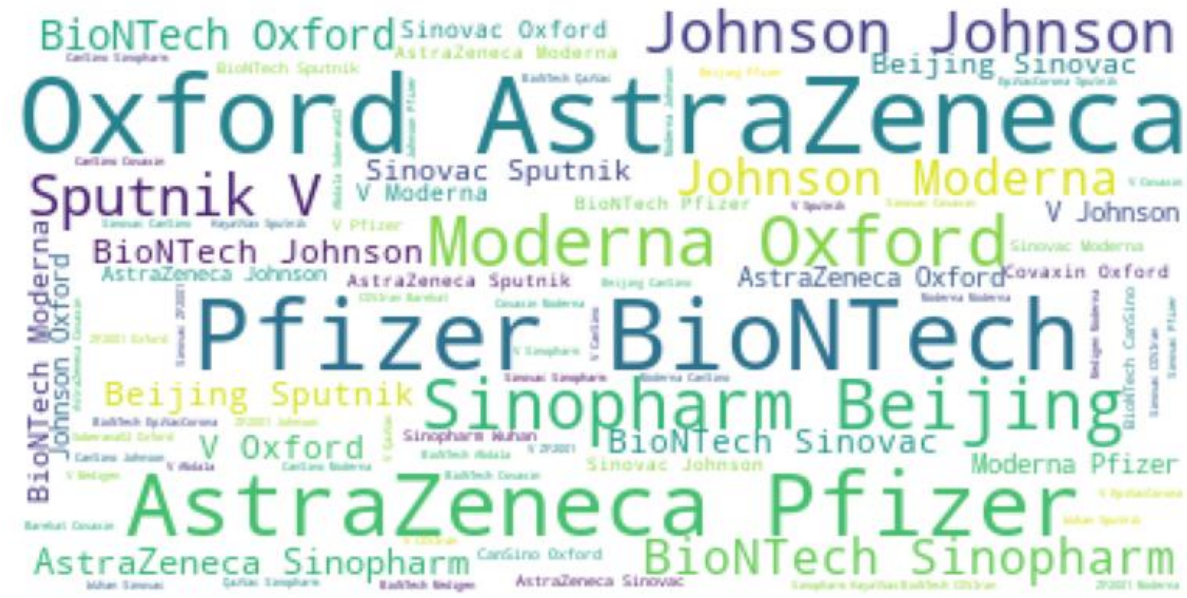
To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

Word Cloud for Vaccine names in dataCV.

Code:

```
#Word Cloud for Vaccine names in dataCV
wordCloud = WordCloud(
    background_color='white',
    max_font_size = 50).generate(' '.join(data_CV.vaccines))

plt.figure(figsize=(15,7))
plt.axis('off')
plt.imshow(wordCloud)
plt.show()


wordCloud_country = WordCloud(
    background_color='white',
    max_font_size = 50).generate(' '.join(data_CV.country))

plt.figure(figsize=(15,7))
plt.axis('off')
plt.imshow(wordCloud_country)
plt.show()
```

Output:



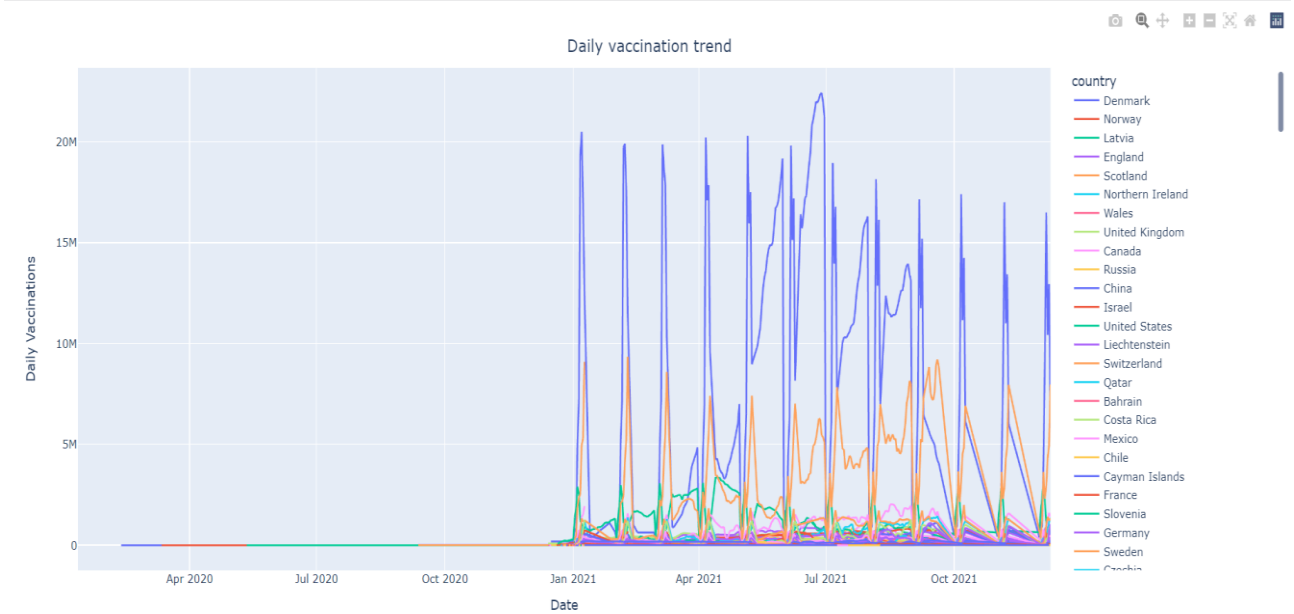
To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

Daily vaccination timeline

Code:

```
#Daily vaccination timeline
fig_vaccinationt_timeline = ex.line(data_CV, x = 'date', y ='daily_vaccinations', color = 'country')

fig_vaccinationt_timeline.update_layout(
    title={
            'text' : "Daily vaccination trend",
            'y':0.95,
            'x':0.5
        },
    xaxis_title="Date",
    yaxis_title="Daily Vaccinations"
)

plot(fig_vaccinationt_timeline)
```

Output:



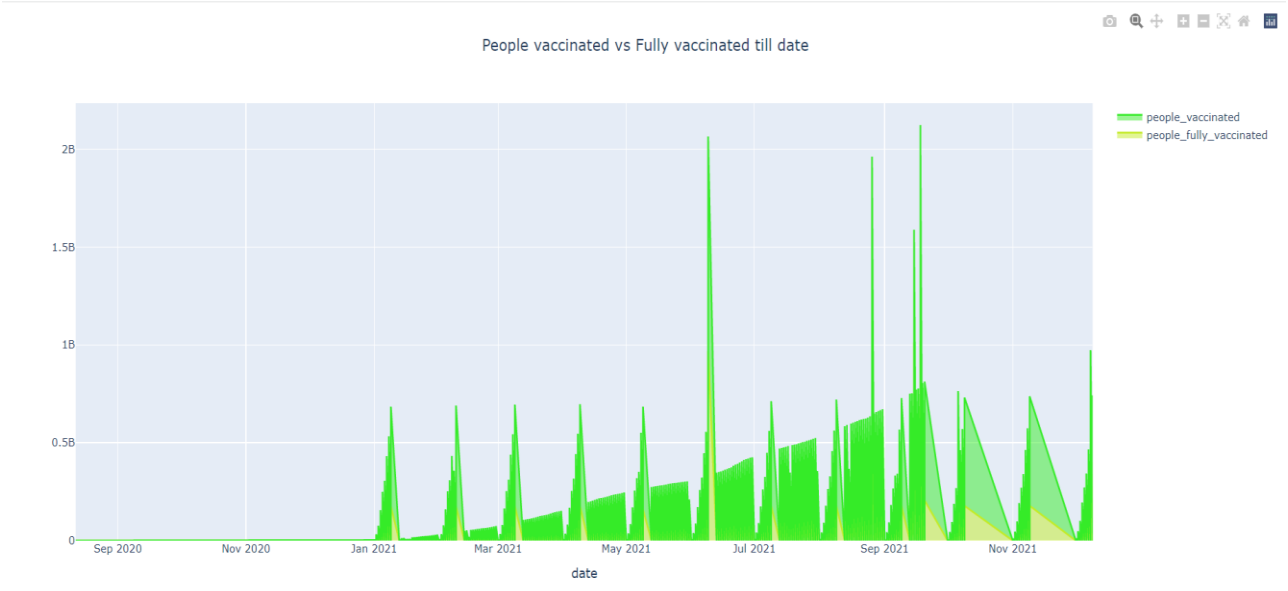
To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

People vaccinated vs fully vaccinated till date

Code:

```
#'People vaccinated vs Fully vaccinated till date
data_for_Comp = data_CV[["date", "people_fully_vaccinated"]]
data_for_Comp = (data_for_Comp.groupby(['people_fully_vaccinated'],as_index=False).sum())

data_for_Comp2 = data_CV[["date", "people_vaccinated"]]
data_for_Comp2 = (data_for_Comp2.groupby(['people_vaccinated'],as_index=False).sum())


figure_timeline = go.Figure(data=[go.Scatter(
                      x = data_for_Comp['date'],
                      y = data_for_Comp['people_fully_vaccinated'],
                      stackgroup='one',
                      name = 'people_fully_vaccinated',
                      marker_color= '#c4eb28'),

                                  go.Scatter(
                                     x = data_for_Comp2['date'],
                      y = data_for_Comp2['people_vaccinated'],
                      stackgroup='one',
                      name = 'people_vaccinated',
                      marker_color= '#35eb28'),
                                  ])

figure_timeline.update_layout(
    title={
            'text' : 'People vaccinated vs Fully vaccinated till date',
            'y':0.95,
            'x':0.5
        },
        xaxis_title="date"
    )

plot(figure_timeline)
```

Output:



To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

Regression Analysis between Population & people fully vaccinated.

Code:

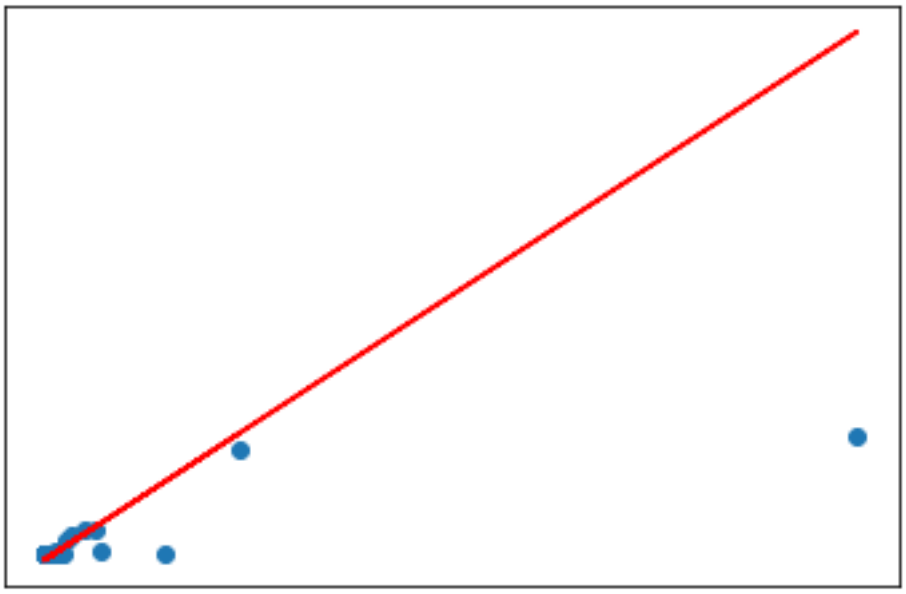
```
pop_regression = data_CV.copy()
pop_regression = pop_regression[["country","iso_code", "people_fully_vaccinated"]]
pop_regression = pd.DataFrame( (pop_regression.groupby(['iso_code'],as_index=False).max()))
pop_regression['Population'] = pop.get_population_a3(str(ratio_cases_pop["iso_code"]))

Population = []
for i in ratio_cases_pop["iso_code"]:
    Population.append(str(pop.get_population_a3(i)))
pop_regression['Population']= Population
pop_regression['Population'] = pd.to_numeric(ratio_cases_pop['Population'], errors='coerce')
pop_regression = pop_regression.replace(np.nan, 0, regex=True)


data_for_regression = pop_regression[["people_fully_vaccinated","Population"]]

from sklearn import linear_model
reg = linear_model.LinearRegression()

data_for_regression.shape
np.random.seed(0) #by setting a seed, if you re-run this code, you should get the same "randomly" generated numbers
numberRows = len(data_for_regression)
randomlyShuffledRows = np.random.permutation(numberRows)
trainingRows = randomlyShuffledRows[0:170]
testRows = randomlyShuffledRows[170:]
xTrainingData = data_for_regression.iloc[trainingRows,1]
yTrainingData = data_for_regression.iloc[trainingRows,0]
xTestData = data_for_regression.iloc[testRows,1]
yTestData = data_for_regression.iloc[testRows,0]
xTrainingData = xTrainingData.values.reshape(-1, 1)
xTestData = xTestData.values.reshape(-1, 1)
reg.fit(xTrainingData,yTrainingData)
print(reg.coef_) #pint value of beta1
print(reg.intercept_) #print value of beta0 (y-intercept)
yPredictions = reg.predict(xTestData)
errors = (yPredictions-yTestData)
sumsOfSquaredErrors = 0
for i in range(len(errors)): #for each row of test data
    squaredError = errors.iloc[i]**2 #compute squared error
    sumsOfSquaredErrors += squaredError #add that to the sum of squared errors

averageSquaredError = sumsOfSquaredErrors/len(errors)#
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(yTestData,yPredictions)
#Should be the same
print(averageSquaredError)
print(mse)
#R-squared value
rsquared = 1 - mse/yTestData.var() #.var() uses N-1=159 divisor
from sklearn.metrics import r2_score
r2 = r2_score(yTestData,yPredictions) #uses N=160 as divisor
rsquared = 1 - mse/yTestData.var(ddof=0)
print(rsquared)
print(r2)
```

```
plt.scatter(xTestData,yTestData)
plt.plot(xTestData, yPredictions,color="red")
plt.xticks(())
plt.yticks(())
plt.show()
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

Output:

```
[0.67129843]
-8960369.585611667
1.1970056026929102e+16
1.1970056026929102e+16
-6.014280832741169
-6.014280832741166
```



Here R square can have a negative value when the model selected does not follow the trend of the data, therefore leading to a worse fit than the horizontal line. It is usually the case when there are constraints on either the intercept or the slope of the linear regression line.

This is the worse model to predict something. As this is the worse fit.

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

Regression Analysis between Population & daily vaccinations.

Code:

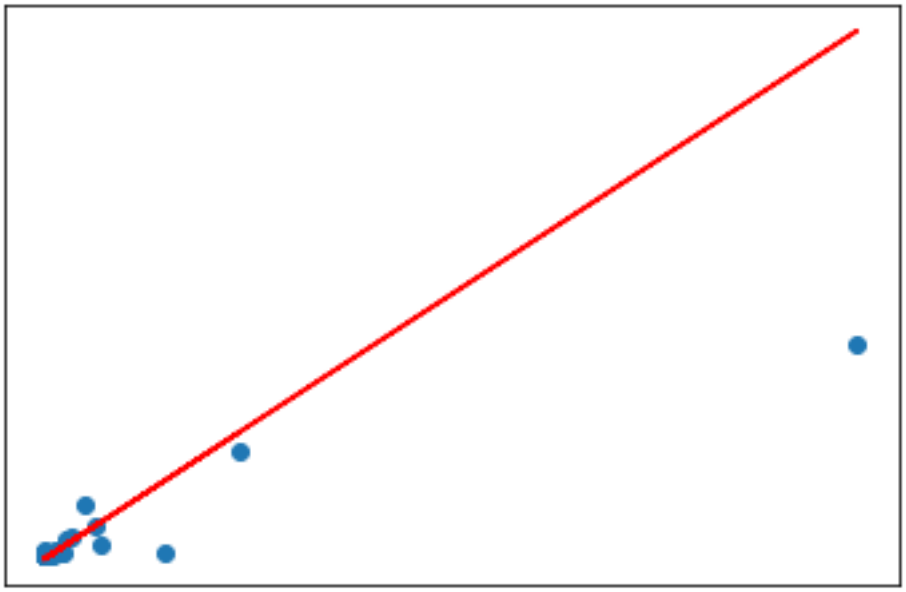
```
804     pop_regression1 = data_CV.copy()
805     pop_regression1 = pop_regression1[["country","iso_code", "daily_vaccinations"]]
806     pop_regression1 = pd.DataFrame( (pop_regression1.groupby(['iso_code'],as_index=False).sum()))
807     pop_regression1['Population'] = pop.get_population_a3(str(ratio_cases_pop["iso_code"]))
808
809     Population = []
810     for i in ratio_cases_pop["iso_code"]:
811         Population.append(str(pop.get_population_a3(i)))
812     pop_regression1['Population']= Population
813     pop_regression1['Population'] = pd.to_numeric(ratio_cases_pop['Population'], errors='coerce')
814     pop_regression1 = pop_regression1.replace(np.nan, 0, regex=True)
815
816
817     data_for_regression = pop_regression1[["daily_vaccinations","Population"]]
818
819     from sklearn import linear_model
820     reg = linear_model.LinearRegression()
821
822     data_for_regression.shape
823     np.random.seed(0) #by setting a seed, if you re-run this code, you should get the same "randomly" generated numbers
824     numberRows = len(data_for_regression)
825     randomlyShuffledRows = np.random.permutation(numberRows)
826     trainingRows = randomlyShuffledRows[0:170]
827     testRows = randomlyShuffledRows[170:]
828     xTrainingData = data_for_regression.iloc[trainingRows,1]
829     yTrainingData = data_for_regression.iloc[trainingRows,0]
830     xTestData = data_for_regression.iloc[testRows,1]
831     yTestData = data_for_regression.iloc[testRows,0]
832     xTrainingData = xTrainingData.values.reshape(-1, 1)
833     xTestData = xTestData.values.reshape(-1, 1)
834     reg.fit(xTrainingData,yTrainingData)
835     print(reg.coef_) #pint value of beta1
836     print(reg.intercept_) #print value of beta0 (y-intercept)
837     yPredictions = reg.predict(xTestData)
838     errors = (yPredictions-yTestData)
839     sumsOfSquaredErrors = 0
840     for i in range(len(errors)): #for each row of test data
841         squaredError = errors.iloc[i]**2 #compute squared error
842         sumsOfSquaredErrors += squaredError #add that to the sum of squared errors
843
844     averageSquaredError = sumsOfSquaredErrors/len(errors)#
845     from sklearn.metrics import mean_squared_error
846     mse = mean_squared_error(yTestData,yPredictions)
847     #Should be the same
848     print(averageSquaredError)
849     print(mse)
850     #R-squared value
851     rsquared = 1 - mse/yTestData.var() #.var() uses N-1=159 divisor
852     from sklearn.metrics import r2_score
853     r2 = r2_score(yTestData,yPredictions) #uses N=160 as divisor
854     rsquared = 1 - mse/yTestData.var(ddof=0)
855     print(rsquared)
856     print(r2)
```

```
858
859     plt.scatter(xTestData,yTestData)
860     plt.plot(xTestData, yPredictions,color="red")
861     plt.xticks(())
862     plt.yticks(())
863     plt.show()
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

Output:

```
[1.43395194]
-18289416.92476433
3.35727579895120 56e+16
3.357275798951205e+16
-0.9295479188239479
-0.9295479188239482
```



Here R square can have a negative value when the model selected does not follow the trend of the data, therefore leading to a worse fit than the horizontal line. It is usually the case when there are constraints on either the intercept or the slope of the linear regression line.

This is the worse model to predict something. As this is the worse fit.

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

Regression Analysis between Population & people vaccinated per hundred

Code:

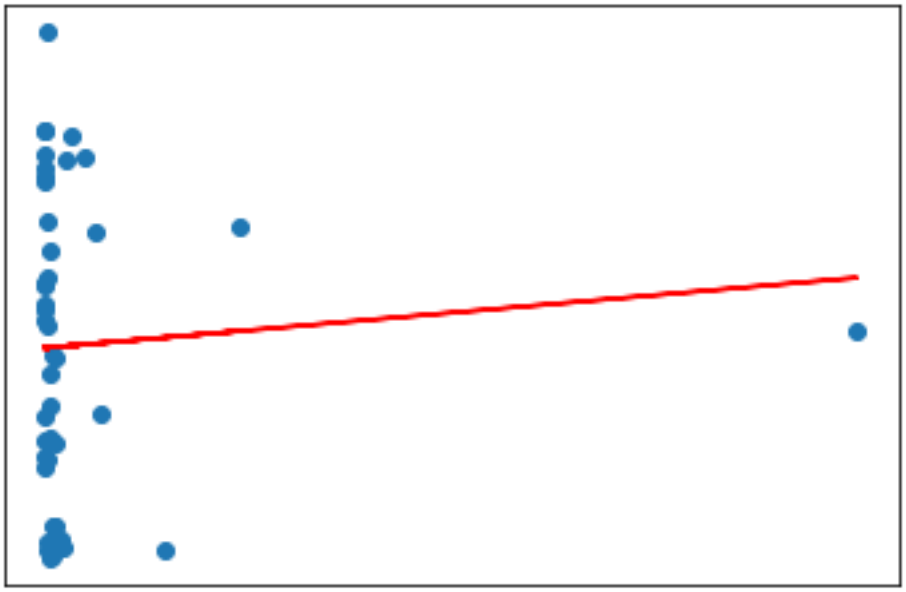
```
#Regression Analysis between Population & people_vaccinated_per_hundred

pop_regression1 = data_CV.copy()
pop_regression1 = pop_regression1[["country","iso_code", "people_vaccinated_per_hundred"]]
pop_regression1 = pd.DataFrame( (pop_regression1.groupby(['iso_code'],as_index=False).max()))
pop_regression1['Population'] = pop.get_population_a3(str(ratio_cases_pop["iso_code"]))

Population = []
for i in ratio_cases_pop["iso_code"]:
    Population.append(str(pop.get_population_a3(i)))
pop_regression1['Population']= Population
pop_regression1['Population'] = pd.to_numeric(ratio_cases_pop['Population'], errors='coerce')
pop_regression1 = pop_regression1.replace(np.nan, 0, regex=True)


data_for_regression = pop_regression1[["people_vaccinated_per_hundred","Population"]]

from sklearn import linear_model
reg = linear_model.LinearRegression()

data_for_regression.shape
np.random.seed(0) #by setting a seed, if you re-run this code, you should get the same "randomly" generated numbers
numberRows = len(data_for_regression)
randomlyShuffledRows = np.random.permutation(numberRows)
trainingRows = randomlyShuffledRows[0:170]
testRows = randomlyShuffledRows[170:]
xTrainingData = data_for_regression.iloc[trainingRows,1]
yTrainingData = data_for_regression.iloc[trainingRows,0]
xTestData = data_for_regression.iloc[testRows,1]
yTestData = data_for_regression.iloc[testRows,0]
xTrainingData = xTrainingData.values.reshape(-1, 1)
xTestData = xTestData.values.reshape(-1, 1)
reg.fit(xTrainingData,yTrainingData)
print(reg.coef_) #pint value of beta1
print(reg.intercept_) #print value of beta0 (y-intercept)
yPredictions = reg.predict(xTestData)
errors = (yPredictions-yTestData)
sumsOfSquaredErrors = 0
for i in range(len(errors)): #for each row of test data
    squaredError = errors.iloc[i]**2 #compute squared error
    sumsOfSquaredErrors += squaredError #add that to the sum of squared errors

averageSquaredError = sumsOfSquaredErrors/len(errors)#
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(yTestData,yPredictions)
#Should be the same
print(averageSquaredError)
print(mse)
#R-squared value
rsquared = 1 - mse/yTestData.var() #.var() uses N-1=159 divisor
from sklearn.metrics import r2_score
r2 = r2_score(yTestData,yPredictions) #uses N=160 as divisor
rsquared = 1 - mse/yTestData.var(ddof=0)
print(rsquared)
print(r2)
```

```
plt.scatter(xTestData,yTestData)
plt.plot(xTestData, yPredictions,color="red")
plt.xticks(())
plt.yticks(())
plt.show()
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

Output:

```
[9.67326689e-09]
40.31806409524198
770.8392578568878
770.8392578568878
-0.0039210077284743 68
-0.003921007728474368
```



Here R square can have a negative value when the model selected does not follow the trend of the data, therefore leading to a worse fit than the horizontal line. It is usually the case when there are constraints on either the intercept or the slope of the linear regression line.

This is the worse model to predict something. As this is the worse fit.

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

Regression Analysis between Population & daily vaccinations per million

Code:

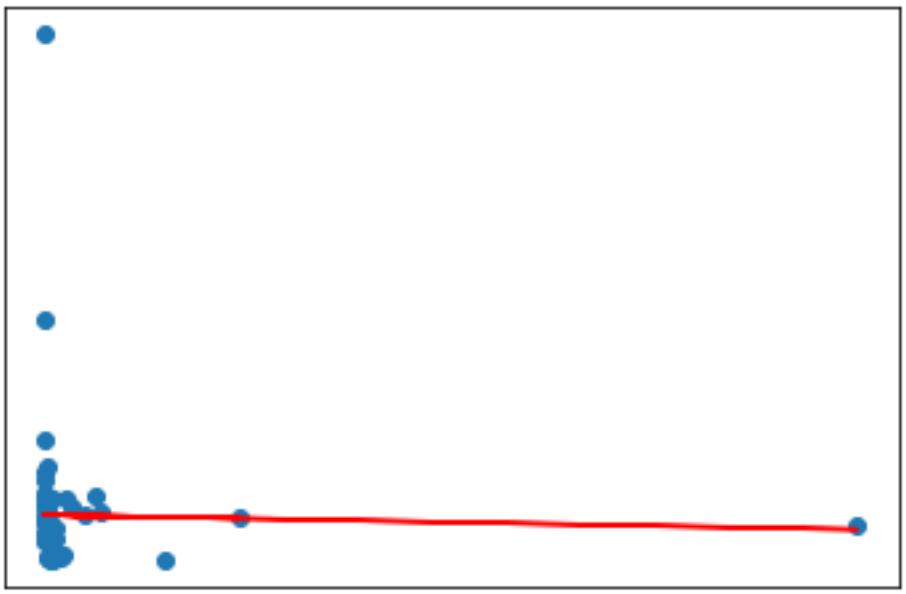
```
#Regression Analysis between Population & daily_vaccinations_per_million

pop_regression1 = data_CV.copy()
pop_regression1 = pop_regression1[["country","iso_code", "daily_vaccinations_per_million"]]
pop_regression1 = pd.DataFrame( (pop_regression1.groupby(['iso_code'],as_index=False).max()))
pop_regression1['Population'] = pop.get_population_a3(str(ratio_cases_pop["iso_code"]))

Population = []
for i in ratio_cases_pop["iso_code"]:
    Population.append(str(pop.get_population_a3(i)))
pop_regression1['Population']= Population
pop_regression1['Population'] = pd.to_numeric(ratio_cases_pop['Population'], errors='coerce')
pop_regression1 = pop_regression1.replace(np.nan, 0, regex=True)


data_for_regression = pop_regression1[["daily_vaccinations_per_million","Population"]]

from sklearn import linear_model
reg = linear_model.LinearRegression()

data_for_regression.shape
np.random.seed(0) #by setting a seed, if you re-run this code, you should get the same "randomly" generated numbers
numberRows = len(data_for_regression)
randomlyShuffledRows = np.random.permutation(numberRows)
trainingRows = randomlyShuffledRows[0:170]
testRows = randomlyShuffledRows[170:]
xTrainingData = data_for_regression.iloc[trainingRows,1]
yTrainingData = data_for_regression.iloc[trainingRows,0]
xTestData = data_for_regression.iloc[testRows,1]
yTestData = data_for_regression.iloc[testRows,0]
xTrainingData = xTrainingData.values.reshape(-1, 1)
xTestData = xTestData.values.reshape(-1, 1)
reg.fit(xTrainingData,yTrainingData)
print(reg.coef_) #pint value of beta1
print(reg.intercept_) #print value of beta0 (y-intercept)
yPredictions = reg.predict(xTestData)
errors = (yPredictions-yTestData)
sumsOfSquaredErrors = 0
for i in range(len(errors)): #for each row of test data
    squaredError = errors.iloc[i]**2 #compute squared error
    sumsOfSquaredErrors += squaredError #add that to the sum of squared errors

averageSquaredError = sumsOfSquaredErrors/len(errors)#
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(yTestData,yPredictions)
#Should be the same
print(averageSquaredError)
print(mse)
#R-squared value
rsquared = 1 - mse/yTestData.var() #.var() uses N-1=159 divisor
from sklearn.metrics import r2_score
r2 = r2_score(yTestData,yPredictions) #uses N=160 as divisor
rsquared = 1 - mse/yTestData.var(ddof=0)
print(rsquared)
print(r2)
```

```
plt.scatter(xTestData,yTestData)
plt.plot(xTestData, yPredictions,color="red")
plt.xticks(())
plt.yticks(())
plt.show()
```

To: **Analytic Manager, United Health Care.**
From: **Aditya Nagori**
Subject: **COVID- 19 Analysis**

Output:

```
[-2.37327889e-06]
10499.75142277471
347165654.1064997
347165654.1064998
-0.012980486264443915
-0.012980486264438706
```



Here the r2 is negative as well as regression coefficient which suggests there is an inverse relationship between the population and daily vaccinations per million, suggesting more the population less people are vaccinated per million.

Also here R square can have a negative value when the model selected does not follow the trend of the data, therefore leading to a worse fit than the horizontal line. It is usually the case when there are constraints on either the intercept or the slope of the linear regression line.

This is the worse model to predict something. As this is the worse fit.