```
import plotly
import plotly.express as px
import pandas as pd
import geopandas as gpd
import shapely as shp
from shapely.geometry import Polygon, LineString
import folium
from folium.plugins import MarkerCluster
from datetime import datetime
```

```
country_data= px.data.gapminder()
country_data.head()
```

	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	AFG	4
1	Afghanistan	Asia	1957	30.332	9240934	820.853030	AFG	4
2	Afghanistan	Asia	1962	31.997	10267083	853.100710	AFG	4
3	Afghanistan	Asia	1967	34.020	11537966	836.197138	AFG	4
4	Afghanistan	Asia	1972	36.088	13079460	739.981106	AFG	4

```
covid_data=pd.read_csv('/content/WHO-COVID-19-global-data-2.csv')
covid_data.head()
```

	Date_reported	Country_code	Country	WHO_region	New_cases	Cumulative_cases	I
0	1/3/2020	AF	Afghanistan	EMRO	0	0	
1	1/4/2020	AF	Afghanistan	EMRO	0	0	
2	1/5/2020	AF	Afghanistan	EMRO	0	0	
3	1/6/2020	AF	Afghanistan	EMRO	0	0	
4	1/7/2020	AF	Afghanistan	EMRO	0	0	

```
country_data.rename(columns={'country':'Country'}, inplace=True)
country_data.head()
```

	Country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	AFG	4
1	Afghanistan	Asia	1957	30.332	9240934	820.853030	AFG	4
2	Afghanistan	Asia	1962	31.997	10267083	853.100710	AFG	4
3	Afghanistan	Asia	1967	34.020	11537966	836.197138	AFG	4
4	Afghanistan	Asia	1972	36.088	13079460	739.981106	AFG	4

```
data =covid_data.merge(country_data[['Country']], on=['Country'], how='left')
```

```
data.head()
```

	Date_reported	Country_code	Country	WHO_region	New_cases	Cumulative_cases	I
0	1/3/2020	AF	Afghanistan	EMRO	0	0	
1	1/3/2020	AF	Afghanistan	EMRO	0	0	
2	1/3/2020	AF	Afghanistan	EMRO	0	0	
3	1/3/2020	AF	Afghanistan	EMRO	0	0	
4	1/3/2020	AF	Afghanistan	EMRO	0	0	

```
data.head()
```

	Date_reported	Country_code	Country	WHO_region	New_cases	Cumulative_cases	I
0	1/3/2020	AF	Afghanistan	EMRO	0	0	
1	1/3/2020	AF	Afghanistan	EMRO	0	0	
2	1/3/2020	AF	Afghanistan	EMRO	0	0	
3	1/3/2020	AF	Afghanistan	EMRO	0	0	
4	1/3/2020	AF	Afghanistan	EMRO	0	0	

```
data =covid_data.merge(country_data[['Country','iso_alpha']], on=['Country'], how='left')
data.head()
```

	Date_reported	Country_code	Country	WHO_region	New_cases	Cumulative_cases	I
0	1/3/2020	AF	Afghanistan	EMRO	0	0	
1	1/3/2020	AF	Afghanistan	EMRO	0	0	
2	1/3/2020	AF	Afghanistan	EMRO	0	0	
3	1/3/2020	AF	Afghanistan	EMRO	0	0	
4	1/3/2020	AF	Afghanistan	EMRO	0	0	

```
latlong_data=pd.read_csv('/content/world_country_and_usa_states_latitude_and_longitude_values.csv')
latlong_data.head()
```

	country_code	latitude	longitude	country	usa_state_code	usa_state_latitude
0	AD	42.546245	1.601554	Andorra	AK	63.588753
1	AE	23.424076	53.847818	United Arab Emirates	AL	32.318231
2	AF	33.939110	67.709953	Afghanistan	AR	35.201050
3	AG	17.060816	-61.796428	Antigua and	AZ	34.048928

```
all_data =data.merge(latlong_data[['country','latitude', 'longitude']], left_on=['Country'], right_on=['country'], how='left')
all_data.head()
```

	Date_reported	Country_code	Country	WHO_region	New_cases	Cumulative_cases	I
0	1/3/2020	AF	Afghanistan	EMRO	0	0	
1	1/3/2020	AF	Afghanistan	EMRO	0	0	
2	1/3/2020	AF	Afghanistan	EMRO	0	0	
3	1/3/2020	AF	Afghanistan	EMRO	0	0	
4	1/3/2020	AF	Afghanistan	EMRO	0	0	

```
all_data.describe()
```

	New_cases	Cumulative_cases	New_deaths	Cumulative_deaths	latitude
count	1.202656e+06	1.202656e+06	1.202656e+06	1.202656e+06	1.155869e+06
mean	2.667411e+03	5.940905e+05	4.000597e+01	1.328813e+04	1.729593e+01
std	1.407419e+04	2.468180e+06	1.734525e+02	4.933097e+04	2.449681e+01
min	-3.295200e+04	0.000000e+00	-9.200000e+01	0.000000e+00	-4.090056e+01
25%	1.000000e+00	2.285000e+03	0.000000e+00	3.400000e+01	4.210484e+00
50%	1.040000e+02	3.267100e+04	1.000000e+00	6.080000e+02	1.578347e+01
75%	1.001000e+03	2.857400e+05	1.500000e+01	5.818000e+03	3.620482e+01
max	1.327469e+06	7.572524e+07	8.786000e+03	8.938700e+05	7.170694e+01

```
df = all_data.groupby(['Country_code','Country','WHO_region','iso_alpha', 'latitude', 'longitude']).sum().reset_index()
df.head()
```

```
<ipython-input-14-07320e5a65d1>:1: FutureWarning: The default value of numeric_only i
  df = all_data.groupby(['Country_code','Country','WHO_region','iso_alpha', 'latitude
```

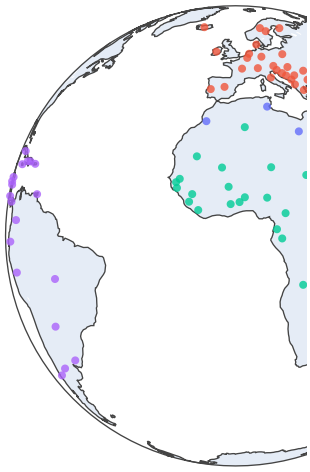
	Country_code	Country	WHO_region	iso_alpha	latitude	longitude	New_cases
0	AF	Afghanistan	EMRO	AFG	33.939110	67.709953	1994292
1	AL	Albania	EURO	ALB	41.153332	20.168331	3175488
2	AO	Angola	AFRO	AGO	-11.202692	17.873887	1180368
3	AR	Argentina	AMRO	ARG	-38.416097	-63.616672	103078548
4	AT	Austria	EURO	AUT	47.516231	14.550072	24833952

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import mean_squared_error
columns_to_analyze = ['Country_code', 'Country', 'WHO_region', 'New_cases', 'Cumulative_cases', 'New_deaths', 'Cumulative_deaths', 'lat:
df_selected = df[columns_to_analyze]
numeric_cols = df_selected.select_dtypes(include=['number']).columns
categorical_cols = df_selected.select_dtypes(include=['object']).columns
results_dict = {}
target_variables = ['New_cases', 'Cumulative_cases', 'New_deaths', 'Cumulative_deaths']
for target in target_variables:
    y_target = df_selected[target]
    encoder = OneHotEncoder(drop='first', sparse=False)
    X_encoded = encoder.fit_transform(df_selected[categorical_cols])
    X_encoded = pd.DataFrame(X_encoded, columns=encoder.get_feature_names_out(categorical_cols))
    X = pd.concat([df_selected[numeric_cols], X_encoded], axis=1)
    X_train, X_test, y_train, y_test = train_test_split(X, y_target, test_size=0.2, random_state=42)
    model = LinearRegression()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    predictions_df = pd.DataFrame({'Country': df_selected.loc[X_test.index, 'Country'], f'{target}_pred': y_pred})
    results_dict[target] = {'predictions': predictions_df.groupby('Country')[f'{target}_pred'].mean(), 'mse': mse}
for target, results in results_dict.items():
    print(f"MSE for {target}: {results['mse']:.2f}\n")
```

```
MSE for New_cases: 0.00

MSE for Cumulative_cases: 0.00

MSE for New_deaths: 0.00

MSE for Cumulative_deaths: 0.00

/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_outpu
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_outpu
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_outpu
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_outpu
  warnings.warn(
```

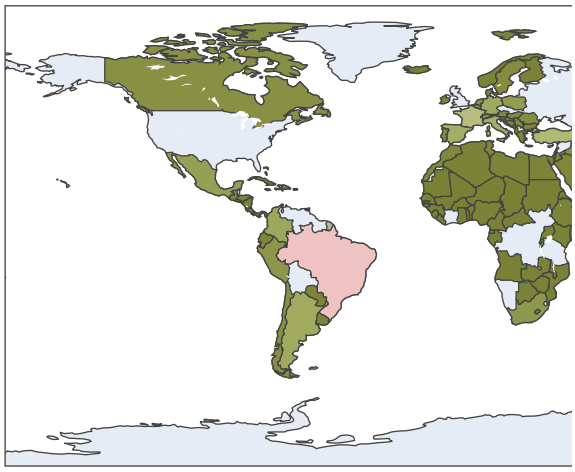
```
map_fig = px.scatter_geo(df,
                         locations='iso_alpha',
                         projection = 'orthographic',
                         color = 'WHO_region',
                         opacity = .8,
                         hover_name = 'Country',
                         hover_data = ['New_cases', 'Cumulative_cases', 'New_deaths',   'Cumulative_deaths'],
                         title = "Visualization of the WHO regions around the world"
                         )
map_fig.show()
```

Visualization of the WHO regions around the world



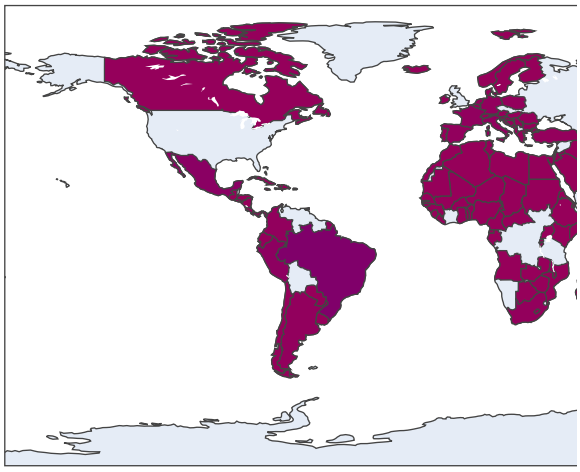
```
fig = px.choropleth(df,
                    locations = 'iso_alpha',
                    locationmode = 'ISO-3',
                    scope = 'world',
                    color = 'Cumulative_cases',
                    hover_name = 'Country',
                    hover_data = ['Country','WHO_region',   'New_cases',    'Cumulative_cases', 'New_deaths',   'Cumulative_deaths'],
                    range_color = [20000000,141000000000],
                    color_continuous_scale='armyrose',
                    title = 'World Covid19 Cumulative Cases')

fig.show()
```

World Covid19 Cumulative Cases



```
fig = px.choropleth(df,
                    locations = 'iso_alpha',
                    locationmode = 'ISO-3',
                    scope = 'world',
                    color = 'Cumulative_deaths',
                    hover_name = 'Country',
                    hover_data = ['Country','WHO_region',   'New_cases',    'Cumulative_cases', 'New_deaths',   'Cumulative_deaths'],
                    range_color = [20000000,141000000000],
                    color_continuous_scale='rainbow',
                    title = 'World Covid19 Cumulative deaths')

fig.show()
```

World Covid19 Cumulative deaths



```
import plotly.express as px
print("Country Names:")
for i in range(0, len(df['Country']), 5):
    for j in range(5):
        if i + j < len(df['Country']):
            print(df['Country'][i + j], end="\t")
    print("\n")
print("Column Names")
print("New_cases\nNew_deaths\nCumulative_cases\nCumulative_deaths")

user_input_country = input("Enter the country (e.g., India): ")
user_input_column = input("Enter the column (e.g., New_cases): ")

# Filter DataFrame based on user input country
filtered_df = df[df['Country'].str.lower() == user_input_country.lower()]
fig = px.choropleth(filtered_df,
                    locations='iso_alpha',
                    locationmode='ISO-3',
                    scope='world',
                    color='WHO_region',
                    hover_name=user_input_column,
                    hover_data=['Country', user_input_column],
                    title=f'Covid19 {user_input_column} for {user_input_country}')

fig.show()
```

```
Country Names:
Afghanistan     Albania Angola  Argentina       Austria

Australia       Bosnia and Herzegovina  Bangladesh      Belgium Burkina Faso

Bulgaria        Bahrain Burundi Benin   Brazil

Botswana        Canada  Central African Republic        Switzerland     Chile

Cameroon        China   Colombia        Costa Rica      Cuba

Germany Djibouti        Denmark Dominican Republic      Algeria

Ecuador Egypt   Eritrea Spain   Ethiopia

Finland France  Gabon   Ghana   Gambia

Guinea  Equatorial Guinea       Greece  Guatemala       Guinea-Bissau

Honduras        Croatia Haiti   Hungary Indonesia

Ireland Israel  India   Iraq    Iceland

Italy   Jamaica Jordan  Japan   Kenya

Cambodia        Comoros Kuwait  Lebanon Sri Lanka

Liberia Lesotho Libya   Morocco Montenegro

Madagascar      Mali    Mongolia        Mauritania      Mauritius

Malawi  Mexico  Malaysia        Mozambique      Niger
```