Global Data Insights!

Introduction

Global Data Insights is a project that focuses on the analysis, and interpretation of data on a global scale to gain valuable insights into various aspects of our world. This project aims to harness the power of data to explore hidden knowledge about world and understand complex global issues.

Data Source and About Dataset

• The dataset used in this project was obtained from Kaggle. It is titled 'World Data' and can be found at https://www.kaggle.com/datasets/freeman007/world-data
• This dataset contains data about population, geographical, demographic indexes, etc and not include few countries.

Objectives

• Identify and list the top 10 highest populated countries in the world based on the most recent available data.
• Is there any relation ship between Population and CO2 Emission?
• Conduct a analysis to uncover and document key findings related to gross education.
• Conduct a visually appealing and insightful analysis of demographic indexes to provide a comprehensive understanding of a population's composition and dynamics.
• Perform an insightful and visually engaging analysis of GDP (Gross Domestic Product) to understand an economy's performance and trends.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import geopandas as gpd
import sqlite3
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import json
import plotly.offline as pyo
```

In [2]:
```
# Initialize Plotly for Jupyter Notebook
pyo.init_notebook_mode(connected=True)
```

In [3]:
```
#Connecting SQLite database
conn = sqlite3.connect(r'D:\Datasets\World Data\world.sqlite')
```

In [4]:
```
#SQL query to connect
query = '''SELECT
        Country,
        Population,
        Urban_population,
        Population_Density,
        LandArea,
        AgriculturalLand,
        ForestedArea,
        "Co2-Emissions",
        GrossPrimaryEducationEnrollment,
        GrossTertiaryEducationEnrollment,
        PhysiciansPerThousand,
        BirthRate,
        FertilityRate,
        InfantMortality,
        LifeExpectancy,
```

```
        Consumer_Price_Index,
        Consumer_price_index_change,
        GDP,
        MinimumWage,
        PopulationLaborForceParticipation,
        UnemploymentRate,
        TaxRevenue
FROM
        world
'''
```

In [5]:
```
#reading database with pandas library
world = pd.read_sql(query, conn)
```

In [6]:
```
pd.set_option('display.max_columns', None)
world.head()
```

Out[6]:

	Country	Population	Urban_population	Population_Density	LandArea	AgriculturalLand	ForestedArea	Co2-Emissions	GrossPrimaryEducatic
0	Afghanistan	38041754	9797273	60	652230	58.1	2.1	8672	
1	Albania	2854191	1747593	105	28748	43.1	28.1	4536	
2	Algeria	43053054	31510100	18	2381741	17.4	0.8	150006	
3	Angola	31825295	21061025	26	1246700	47.5	46.3	34693	
4	Argentina	44938712	41339571	17	2780400	54.3	9.8	201348	

In [7]:
```
#converting column names to lower case and removing extra spaces

world.rename(columns = lambda x : x.lower().strip(), inplace = True)
```

In [8]:
```
#renaming column names to make more readable

world = world.rename(columns ={'landarea':'land_area', 'agriculturalland':'agricultural_land','forestedarea':'f
                       'co2-emissions':'co2_emissions', 'grossprimaryeducationenrollment':'gross_primary_educati
                       'grosstertiaryeducationenrollment':'gross_tertiary_education_enrollment',
                       'physiciansperthousand':'physicians_per_thousand', 'birthrate':'birth_rate', 'fertilityra
                       'infantmortality':'infant_mortality', 'lifeexpectancy':'life_expectancy', 'minimumwage':'
                       'populationlaborforceparticipation':'population_labor_force_participation', 'unemployment
                       'taxrevenue':'tax_revenue'})
```

In [9]:
```
#Checking missing values and data types of columns
world.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 107 entries, 0 to 106
Data columns (total 22 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   country                               107 non-null    object
 1   population                            107 non-null    object
 2   urban_population                      107 non-null    object
 3   population_density                    107 non-null    object
 4   land_area                             107 non-null    object
 5   agricultural_land                     107 non-null    object
 6   forested_area                         107 non-null    object
 7   co2_emissions                         107 non-null    object
 8   gross_primary_education_enrollment    107 non-null    object
 9   gross_tertiary_education_enrollment   107 non-null    object
 10  physicians_per_thousand               107 non-null    object
 11  birth_rate                            107 non-null    object
 12  fertility_rate                        107 non-null    object
 13  infant_mortality                      107 non-null    object
 14  life_expectancy                       107 non-null    object
 15  consumer_price_index                  107 non-null    object
 16  consumer_price_index_change           107 non-null    object
 17  gdp                                   107 non-null    object
 18  minimum_wage                          107 non-null    object
 19  population_labor_force_participation  107 non-null    object
 20  unemployment_rate                     107 non-null    object
 21  tax_revenue                           107 non-null    object
dtypes: object(22)
memory usage: 18.5+ KB
```

In [10]:
```
#transforming object columns to relevent data types
exclude_columns = ['country']
world[world.columns.difference(exclude_columns)] = world[world.columns.difference(exclude_columns)].apply(pd.to
```

In [11]:
```
#loading world map loading data to graph world map charts
worldmap = json.load(open(r'D:\Datasets\World Data\New folder\countries.geojson'))
worldmap.keys()
```

Out[11]:
```
dict_keys(['type', 'features'])
```

In [12]:
```
worldmap['features'][0]
```

Out[12]:

```
{'type': 'Feature',
 'properties': {'ADMIN': 'Aruba', 'ISO_A3': 'ABW'},
 'geometry': {'type': 'Polygon',
  'coordinates': [[[-69.99693762899992, 12.577582098000036],
    [-69.93639075399994, 12.53172435100005],
    [-69.92467200399994, 12.519232489000046],
    [-69.91576087099992, 12.497015692000076],
    [-69.88019771999984, 12.453558661000045],
    [-69.87682044199994, 12.427394924000097],
    [-69.88809160099993, 12.417669989000046],
    [-69.90880286399994, 12.417792059000107],
    [-69.93053137899989, 12.425970770000035],
    [-69.94513912699992, 12.44037506700009],
    [-69.92467200399994, 12.44037506700009],
    [-69.92467200399994, 12.447211005000014],
    [-69.95856686099992, 12.463202216000099],
    [-70.02765865799992, 12.522935289000088],
    [-70.04808508999989, 12.53115469000008],
    [-70.05809485599988, 12.537176825000088],
    [-70.06240800699987, 12.546820380000057],
    [-70.06037350199995, 12.556952216000113],
    [-70.0510961579999, 12.574042059000064],
    [-70.04873613199993, 12.583726304000024],
    [-70.05264238199993, 12.600002346000053],
    [-70.05964107999992, 12.614243882000054],
    [-70.06110592399997, 12.625392971000068],
    [-70.04873613199993, 12.632147528000104],
    [-70.00715084499987, 12.5855166690001],
    [-69.99693762899992, 12.577582098000036]]]}}
```


In [13]:
```
#investigating json map data
worldmap['features'][3]['properties']
```

Out[13]:

```
{'ADMIN': 'Anguilla', 'ISO_A3': 'AIA'}
```


In [14]:
```
#making new dictionary using json data to connect with data frame
world_map_id = {}
for feature in worldmap['features']:
    feature['id'] = feature['properties']['ADMIN']
    world_map_id[feature['properties']['ADMIN']] = feature['id']
```

In [15]:
```
world_map_id
```

Out[15]:

```
{'Aruba': 'Aruba',
 'Afghanistan': 'Afghanistan',
 'Angola': 'Angola',
 'Anguilla': 'Anguilla',
 'Albania': 'Albania',
 'Aland': 'Aland',
 'Andorra': 'Andorra',
 'United Arab Emirates': 'United Arab Emirates',
 'Argentina': 'Argentina',
 'Armenia': 'Armenia',
 'American Samoa': 'American Samoa',
 'Antarctica': 'Antarctica',
 'Ashmore and Cartier Islands': 'Ashmore and Cartier Islands',
 'French Southern and Antarctic Lands': 'French Southern and Antarctic Lands',
 'Antigua and Barbuda': 'Antigua and Barbuda',
 'Australia': 'Australia',
 'Austria': 'Austria',
 'Azerbaijan': 'Azerbaijan',
 'Burundi': 'Burundi',
 'Belgium': 'Belgium',
 'Benin': 'Benin',
 'Burkina Faso': 'Burkina Faso',
 'Bangladesh': 'Bangladesh',
 'Bulgaria': 'Bulgaria',
 'Bahrain': 'Bahrain',
 'The Bahamas': 'The Bahamas',
 'Bosnia and Herzegovina': 'Bosnia and Herzegovina',
 'Bajo Nuevo Bank (Petrel Is.)': 'Bajo Nuevo Bank (Petrel Is.)',
 'Saint Barthelemy': 'Saint Barthelemy',
 'Belarus': 'Belarus',
 'Belize': 'Belize',
 'Bermuda': 'Bermuda',
 'Bolivia': 'Bolivia',
 'Brazil': 'Brazil',
 'Barbados': 'Barbados',
 'Brunei': 'Brunei',
 'Bhutan': 'Bhutan',
 'Botswana': 'Botswana',
 'Central African Republic': 'Central African Republic',
 'Canada': 'Canada',
 'Switzerland': 'Switzerland',
 'Chile': 'Chile',
 'China': 'China',
```

```
'Ivory Coast': 'Ivory Coast',
'Clipperton Island': 'Clipperton Island',
'Cameroon': 'Cameroon',
'Cyprus No Mans Area': 'Cyprus No Mans Area',
'Democratic Republic of the Congo': 'Democratic Republic of the Congo',
'Republic of Congo': 'Republic of Congo',
'Cook Islands': 'Cook Islands',
'Colombia': 'Colombia',
'Comoros': 'Comoros',
'Cape Verde': 'Cape Verde',
'Costa Rica': 'Costa Rica',
'Coral Sea Islands': 'Coral Sea Islands',
'Cuba': 'Cuba',
'CuraÃ§ao': 'CuraÃ§ao',
'Cayman Islands': 'Cayman Islands',
'Northern Cyprus': 'Northern Cyprus',
'Cyprus': 'Cyprus',
'Czech Republic': 'Czech Republic',
'Germany': 'Germany',
'Djibouti': 'Djibouti',
'Dominica': 'Dominica',
'Denmark': 'Denmark',
'Dominican Republic': 'Dominican Republic',
'Algeria': 'Algeria',
'Ecuador': 'Ecuador',
'Egypt': 'Egypt',
'Eritrea': 'Eritrea',
'Dhekelia Sovereign Base Area': 'Dhekelia Sovereign Base Area',
'Spain': 'Spain',
'Estonia': 'Estonia',
'Ethiopia': 'Ethiopia',
'Finland': 'Finland',
'Fiji': 'Fiji',
'Falkland Islands': 'Falkland Islands',
'France': 'France',
'Faroe Islands': 'Faroe Islands',
'Federated States of Micronesia': 'Federated States of Micronesia',
'Gabon': 'Gabon',
'United Kingdom': 'United Kingdom',
'Georgia': 'Georgia',
'Guernsey': 'Guernsey',
'Ghana': 'Ghana',
'Gibraltar': 'Gibraltar',
'Guinea': 'Guinea',
'Gambia': 'Gambia',
'Guinea Bissau': 'Guinea Bissau',
'Equatorial Guinea': 'Equatorial Guinea',
'Greece': 'Greece',
'Grenada': 'Grenada',
'Greenland': 'Greenland',
'Guatemala': 'Guatemala',
'Guam': 'Guam',
'Guyana': 'Guyana',
'Hong Kong S.A.R.': 'Hong Kong S.A.R.',
'Heard Island and McDonald Islands': 'Heard Island and McDonald Islands',
'Honduras': 'Honduras',
'Croatia': 'Croatia',
'Haiti': 'Haiti',
'Hungary': 'Hungary',
'Indonesia': 'Indonesia',
'Isle of Man': 'Isle of Man',
'India': 'India',
'Indian Ocean Territories': 'Indian Ocean Territories',
'British Indian Ocean Territory': 'British Indian Ocean Territory',
'Ireland': 'Ireland',
'Iran': 'Iran',
'Iraq': 'Iraq',
'Iceland': 'Iceland',
'Israel': 'Israel',
'Italy': 'Italy',
'Jamaica': 'Jamaica',
'Jersey': 'Jersey',
'Jordan': 'Jordan',
'Japan': 'Japan',
'Baykonur Cosmodrome': 'Baykonur Cosmodrome',
'Siachen Glacier': 'Siachen Glacier',
'Kazakhstan': 'Kazakhstan',
'Kenya': 'Kenya',
'Kyrgyzstan': 'Kyrgyzstan',
'Cambodia': 'Cambodia',
'Kiribati': 'Kiribati',
'Saint Kitts and Nevis': 'Saint Kitts and Nevis',
'South Korea': 'South Korea',
'Kosovo': 'Kosovo',
'Kuwait': 'Kuwait',
'Laos': 'Laos',
'Lebanon': 'Lebanon',
'Liberia': 'Liberia',
'Libya': 'Libya',
```

```
'Saint Lucia': 'Saint Lucia',
'Liechtenstein': 'Liechtenstein',
'Sri Lanka': 'Sri Lanka',
'Lesotho': 'Lesotho',
'Lithuania': 'Lithuania',
'Luxembourg': 'Luxembourg',
'Latvia': 'Latvia',
'Macao S.A.R': 'Macao S.A.R',
'Saint Martin': 'Saint Martin',
'Morocco': 'Morocco',
'Monaco': 'Monaco',
'Moldova': 'Moldova',
'Madagascar': 'Madagascar',
'Maldives': 'Maldives',
'Mexico': 'Mexico',
'Marshall Islands': 'Marshall Islands',
'Macedonia': 'Macedonia',
'Mali': 'Mali',
'Malta': 'Malta',
'Myanmar': 'Myanmar',
'Montenegro': 'Montenegro',
'Mongolia': 'Mongolia',
'Northern Mariana Islands': 'Northern Mariana Islands',
'Mozambique': 'Mozambique',
'Mauritania': 'Mauritania',
'Montserrat': 'Montserrat',
'Mauritius': 'Mauritius',
'Malawi': 'Malawi',
'Malaysia': 'Malaysia',
'Namibia': 'Namibia',
'New Caledonia': 'New Caledonia',
'Niger': 'Niger',
'Norfolk Island': 'Norfolk Island',
'Nigeria': 'Nigeria',
'Nicaragua': 'Nicaragua',
'Niue': 'Niue',
'Netherlands': 'Netherlands',
'Norway': 'Norway',
'Nepal': 'Nepal',
'Nauru': 'Nauru',
'New Zealand': 'New Zealand',
'Oman': 'Oman',
'Pakistan': 'Pakistan',
'Panama': 'Panama',
'Pitcairn Islands': 'Pitcairn Islands',
'Peru': 'Peru',
'Spratly Islands': 'Spratly Islands',
'Philippines': 'Philippines',
'Palau': 'Palau',
'Papua New Guinea': 'Papua New Guinea',
'Poland': 'Poland',
'Puerto Rico': 'Puerto Rico',
'North Korea': 'North Korea',
'Portugal': 'Portugal',
'Paraguay': 'Paraguay',
'Palestine': 'Palestine',
'French Polynesia': 'French Polynesia',
'Qatar': 'Qatar',
'Romania': 'Romania',
'Russia': 'Russia',
'Rwanda': 'Rwanda',
'Western Sahara': 'Western Sahara',
'Saudi Arabia': 'Saudi Arabia',
'Scarborough Reef': 'Scarborough Reef',
'Sudan': 'Sudan',
'South Sudan': 'South Sudan',
'Senegal': 'Senegal',
'Serranilla Bank': 'Serranilla Bank',
'Singapore': 'Singapore',
'South Georgia and South Sandwich Islands': 'South Georgia and South Sandwich Islands',
'Saint Helena': 'Saint Helena',
'Solomon Islands': 'Solomon Islands',
'Sierra Leone': 'Sierra Leone',
'El Salvador': 'El Salvador',
'San Marino': 'San Marino',
'Somaliland': 'Somaliland',
'Somalia': 'Somalia',
'Saint Pierre and Miquelon': 'Saint Pierre and Miquelon',
'Republic of Serbia': 'Republic of Serbia',
'Sao Tome and Principe': 'Sao Tome and Principe',
'Suriname': 'Suriname',
'Slovakia': 'Slovakia',
'Slovenia': 'Slovenia',
'Sweden': 'Sweden',
'Swaziland': 'Swaziland',
'Sint Maarten': 'Sint Maarten',
'Seychelles': 'Seychelles',
'Syria': 'Syria',
'Turks and Caicos Islands': 'Turks and Caicos Islands',
```

```
 'Chad': 'Chad',
 'Togo': 'Togo',
 'Thailand': 'Thailand',
 'Tajikistan': 'Tajikistan',
 'Turkmenistan': 'Turkmenistan',
 'East Timor': 'East Timor',
 'Tonga': 'Tonga',
 'Trinidad and Tobago': 'Trinidad and Tobago',
 'Tunisia': 'Tunisia',
 'Turkey': 'Turkey',
 'Tuvalu': 'Tuvalu',
 'Taiwan': 'Taiwan',
 'United Republic of Tanzania': 'United Republic of Tanzania',
 'Uganda': 'Uganda',
 'Ukraine': 'Ukraine',
 'United States Minor Outlying Islands': 'United States Minor Outlying Islands',
 'Uruguay': 'Uruguay',
 'United States of America': 'United States of America',
 'US Naval Base Guantanamo Bay': 'US Naval Base Guantanamo Bay',
 'Uzbekistan': 'Uzbekistan',
 'Vatican': 'Vatican',
 'Saint Vincent and the Grenadines': 'Saint Vincent and the Grenadines',
 'Venezuela': 'Venezuela',
 'British Virgin Islands': 'British Virgin Islands',
 'United States Virgin Islands': 'United States Virgin Islands',
 'Vietnam': 'Vietnam',
 'Vanuatu': 'Vanuatu',
 'Wallis and Futuna': 'Wallis and Futuna',
 'Akrotiri Sovereign Base Area': 'Akrotiri Sovereign Base Area',
 'Samoa': 'Samoa',
 'Yemen': 'Yemen',
 'South Africa': 'South Africa',
 'Zambia': 'Zambia',
 'Zimbabwe': 'Zimbabwe'}
```

In [16]:
```
#conneting json map data to data frame
world['id'] = world['country'].map(world_map_id, None)
```

In [17]:
```
world
```

Out[17]:

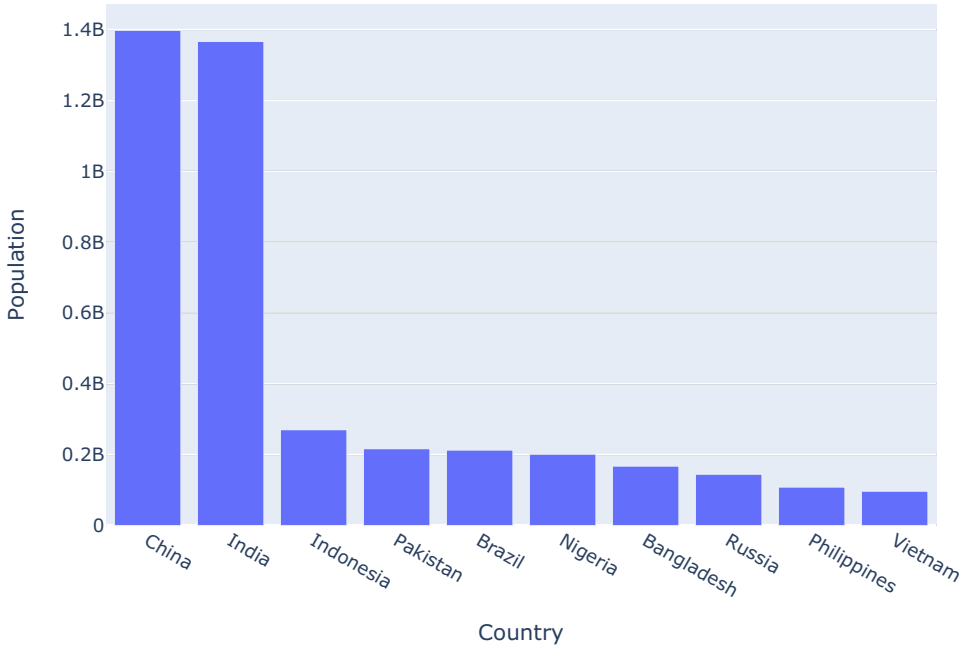
	country	population	urban_population	population_density	land_area	agricultural_land	forested_area	co2_emissions	gross_primary_
0	Afghanistan	38041754	9797273	60	652230	58.1	2.1	8672	
1	Albania	2854191	1747593	105	28748	43.1	28.1	4536	
2	Algeria	43053054	31510100	18	2381741	17.4	0.8	150006	
3	Angola	31825295	21061025	26	1246700	47.5	46.3	34693	
4	Argentina	44938712	41339571	17	2780400	54.3	9.8	201348	
...	...	...	...	...	...	...	...	...	
102	Ukraine	44385155	30835699	75	603550	71.7	16.7	202250	
103	United Kingdom	66834405	55908316	281	243610	71.7	13.1	379025	
104	Uruguay	3461734	3303394	20	176215	82.6	10.7	6766	
105	Vietnam	96462106	35332140	314	331210	39.3	48.1	192668	
106	Zambia	17861030	7871713	25	752618	32.1	65.2	5141	

107 rows × 23 columns

In [18]:
```
#world['population_scale'] = np.log10(world['population'])
#fig = px.choropleth_mapbox(world,
#                   locations= 'id',
#                   geojson= worldmap,
#                    color= 'population_scale',
#                    hover_name='country',
#                    hover_data=['population'],
#                    mapbox_style= 'carto-positron',
#                     center = {'lat':0, 'lon':0},
#                     zoom = 0.3,
#                     opacity = 0.5
#)
#fig.update_layout(
#    title = 'World Population'
#)
#
#fig.show()
```

In [19]:
```
#creating a bar chart using plotly.graph_objects to visualize top 10 highest populated countries

pop = world[['country', 'population']].sort_values(by='population', ascending= False)[:10] #creating data frame
fig = go.Figure(go.Bar(
                    y = pop['population'],
```

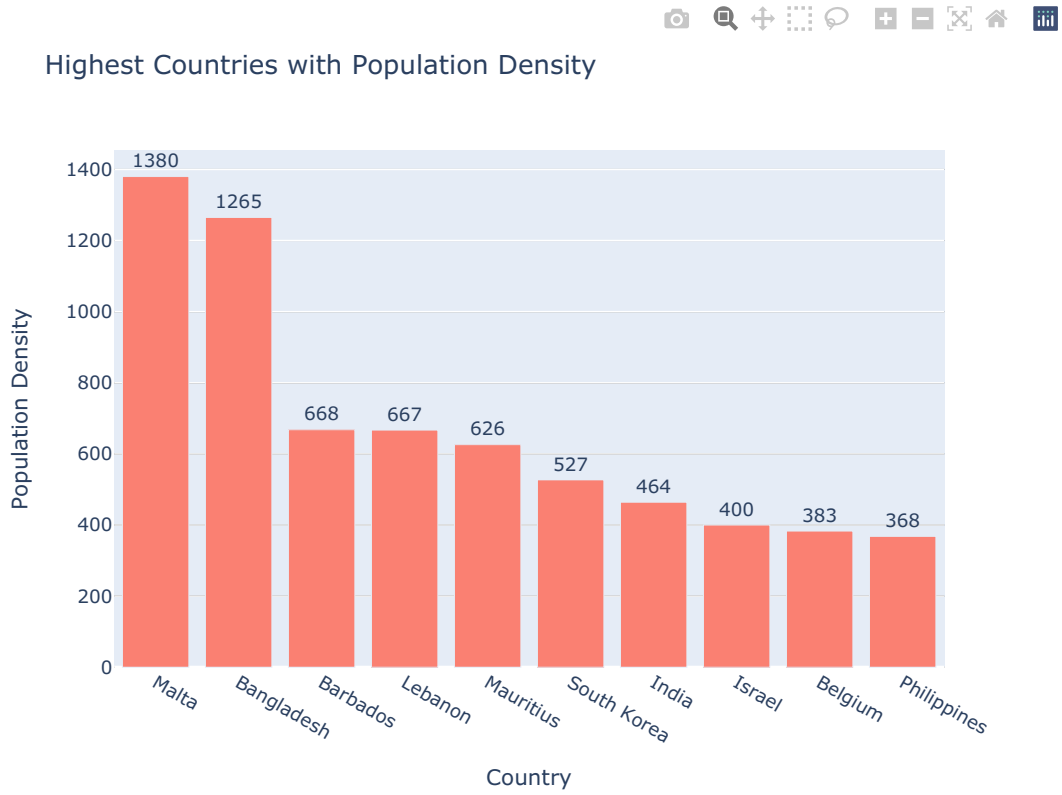
```
                    x = pop['country']
))
fig.update_layout(
    title = 'Top 10 Populated Countries',
    xaxis_title = 'Country',
    yaxis_title = 'Population')
fig.show()
```



• From above chart we can observe that China and India having significantly higher populations compared to other countries.

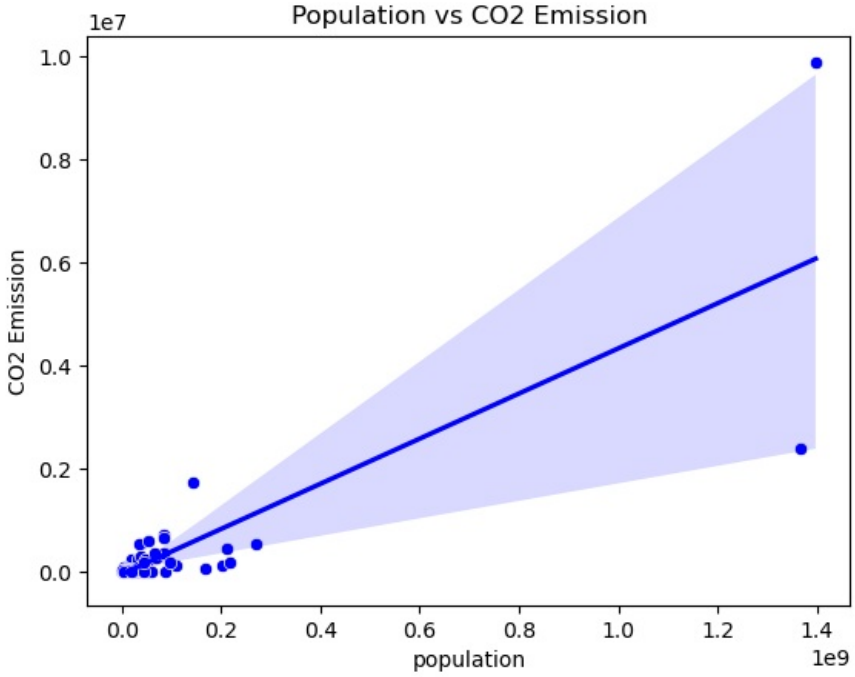
In [20]:

```
popd = world[['country', 'population_density']].sort_values(by='population_density', ascending = False)[:10]
fig = go.Figure(go.Bar(
                    x = popd['country'],
                    y = popd['population_density'],
                    marker=dict(color = 'salmon'),
                    text = popd['population_density'],
                    textposition = 'outside'
))
fig.update_layout(
    title = 'Highest Countries with Population Density',
    xaxis_title = 'Country',
    yaxis_title = 'Population Density'
)
fig.show()
```



• China and India had significantly higher populations but Malta and Bangladesh have higher population density,which means they have a larger number of people living in a relatively smaller land area.

In [21]:
```
sns.scatterplot(x =  world['population'],y =  world['co2_emissions'], color='blue')
sns.regplot(x =  world['population'],y =  world['co2_emissions'], scatter=False, color='blue', label='OLS trend
plt.xlabel('population')
plt.ylabel('CO2 Emission')
plt.title('Population vs CO2 Emission')
plt.show()
```



• From above scatterplot we can see that there is positive correlation between Population and CO2 Emission. Because a higher population typically leads to greater energy consumption, industrial activity, and transportation demands, which can result in higher CO2 emissions.

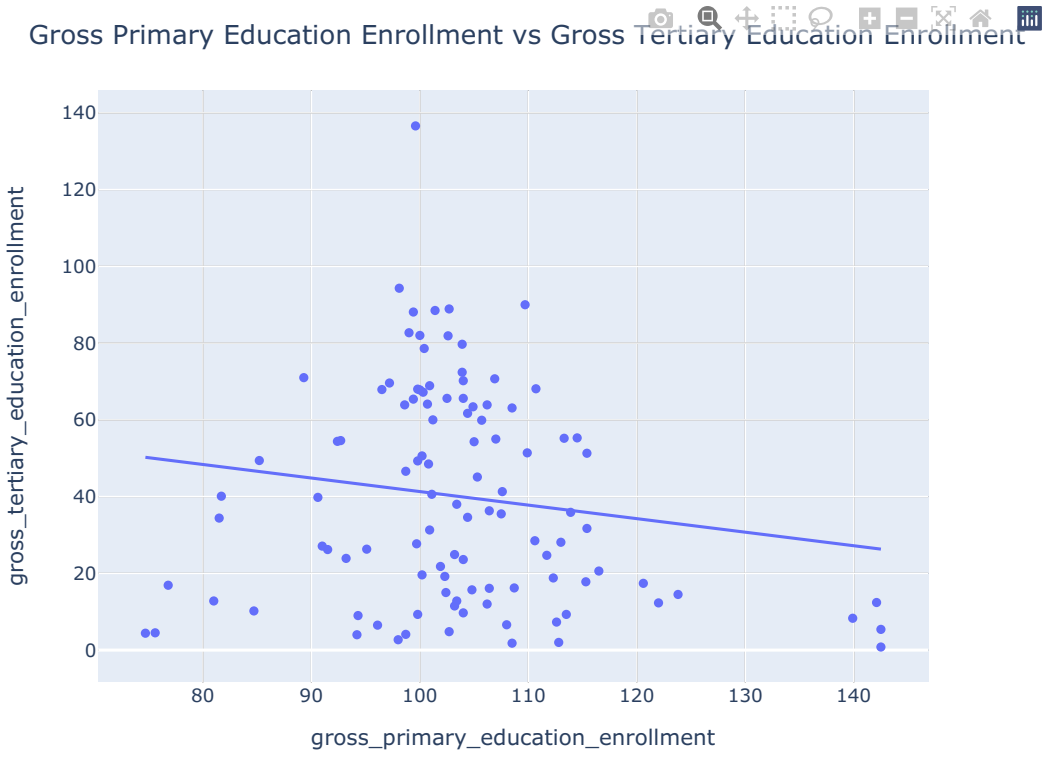
In [22]:
```
landarea_sorted = world.sort_values(by = 'land_area', ascending = False)[:10]
landarea_sorted[['country', 'land_area', 'agricultural_land', 'forested_area']]
```

Out[22]:

	country	land_area	agricultural_land	forested_area
81	Russia	17098240	13.3	49.8
19	Canada	9984670	6.9	38.2
21	China	9596960	56.2	22.4
13	Brazil	8515770	33.9	58.9
40	India	3287263	60.4	23.8
4	Argentina	2780400	54.3	9.8
47	Kazakhstan	2724900	80.4	1.2
2	Algeria	2381741	17.4	0.8
26	Democratic Republic of the Congo	2344858	11.6	67.2
82	Saudi Arabia	2149690	80.8	0.5

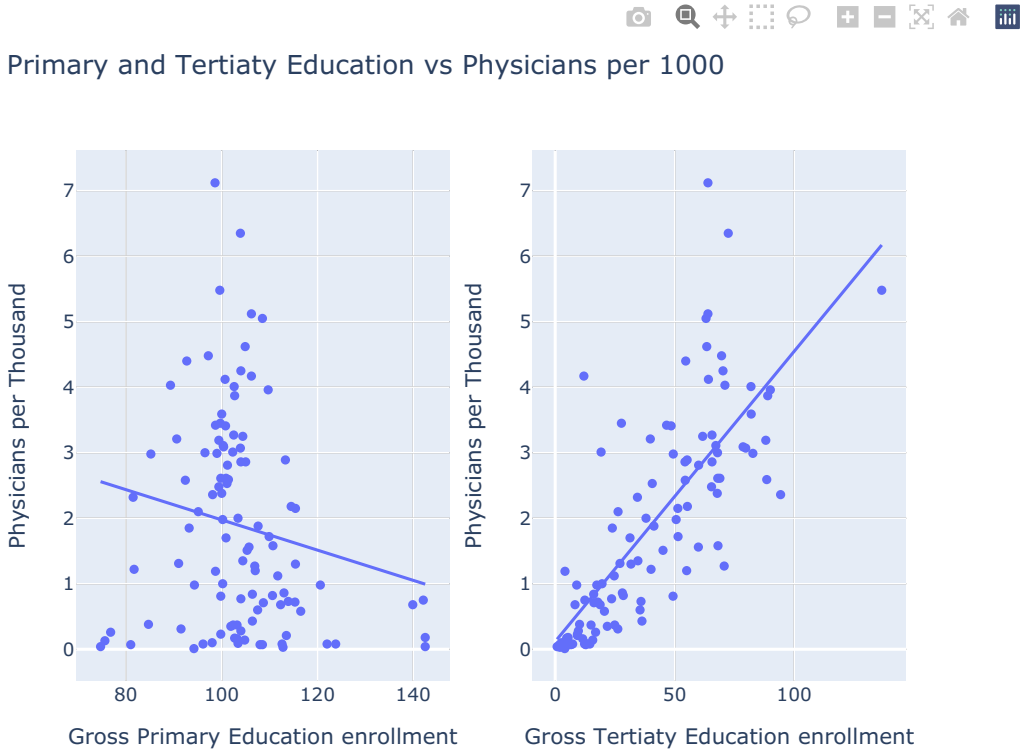
• Russia and Canada have highest land area but low agricultural land precentage and high forested area precentage reflecting their vast and often sparsely populated territories with significant natural landscape from all land while Kazakhstan and Saudi Arabia have more than 80% of land dedicated to agricultural and very low forested area precentage.

In [23]:
```
fig = px.scatter(world, x = 'gross_primary_education_enrollment', y = 'gross_tertiary_education_enrollment',
                 trendline='ols',  hover_data=world[['country']])
fig.update_layout(
    title = 'Gross Primary Education Enrollment vs Gross Tertiary Education Enrollment')
fig.show()
```



In [24]:
```
fig = make_subplots(rows= 1, cols=2)

scat1 = px.scatter(world, x = 'gross_primary_education_enrollment', y = 'physicians_per_thousand', trendline='o
                   hover_data= world[['country']])
scat1_t = scat1.data
for t in scat1_t:
    fig.add_trace(go.Scatter(t), row = 1, col = 1)

fig.update_xaxes(title_text = 'Gross Primary Education enrollment', row=1, col=1)
fig.update_yaxes(title_text = 'Physicians per Thousand', row=1, col=1)

scat2 = px.scatter(world, x = 'gross_tertiary_education_enrollment', y = 'physicians_per_thousand', trendline='
```

```
scat2_t = scat2.data
for t in scat2_t:
    fig.add_trace(go.Scatter(t), row = 1, col = 2)

fig.update_xaxes(title_text = 'Gross Tertiaty Education enrollment', row=1, col=2)
fig.update_yaxes(title_text = 'Physicians per Thousand', row=1, col=2)

fig.update_layout(title_text = 'Primary and Tertiaty Education vs Physicians per 1000')
fig.show()
```



•There are no any strong relationship between Gross Primary Education enrollment vs Gross Tertiaty Education enrollment and Physicians per 1000.
•Also from above second scatterplot we can identify that positive correlation between Tertiary education enrollment and Physicians per thousand.

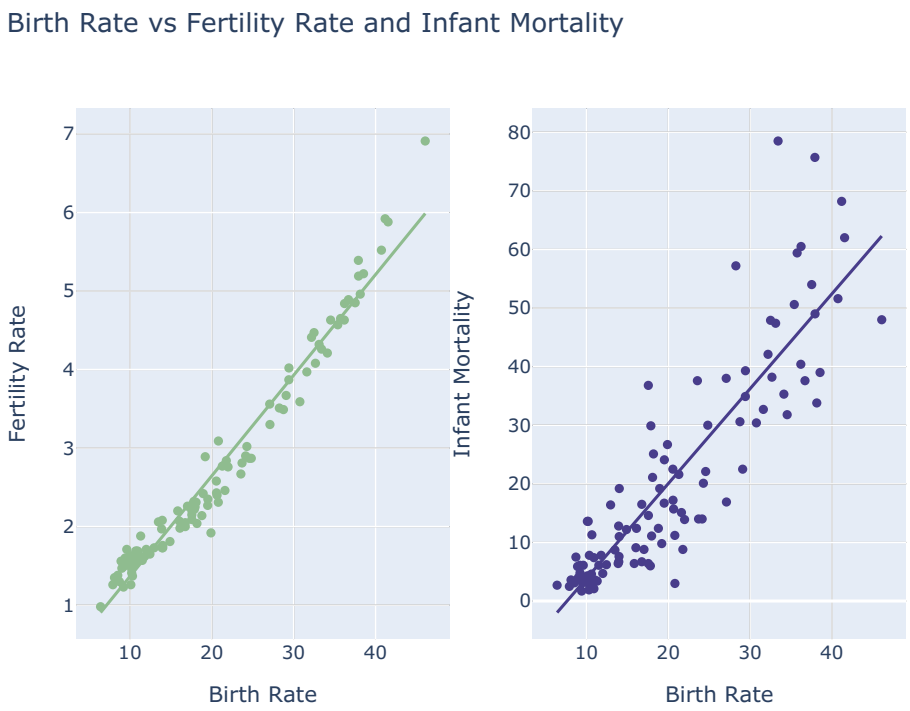
In [25]:
```
#retrieve the Country with the highest birth rate
max_birth_rate = world[world['birth_rate'] == world['birth_rate'].max()]
print(max_birth_rate[['country','birth_rate', 'life_expectancy']])

#retrieve the Country with the highest life expectancy
max_life_expectancy = world[world['life_expectancy'] == world['life_expectancy'].max()]
print(max_life_expectancy[['country','birth_rate', 'life_expectancy']])
```

```
   country  birth_rate  life_expectancy
70   Niger       46.08             62.0
   country  birth_rate  life_expectancy
89   Spain         7.9             83.3
```

In [26]:
```
fig = px.scatter(world, x = 'birth_rate', y = 'life_expectancy', trendline='ols', hover_data=world[['country']]

fig.update_traces(marker = dict(color = 'dimgrey'))

fig.update_layout(
    title = 'Birth Rate vs Life Expectancy',
    xaxis_title = 'Birth Rate',
    yaxis_title = 'Life Expectance'
)
fig.show()
```

• From available data Birth rate and Life expectancy are negatively related. Niger have highest Birth rate and low life expectancy when spain have highest life expectancy and low birth rate.

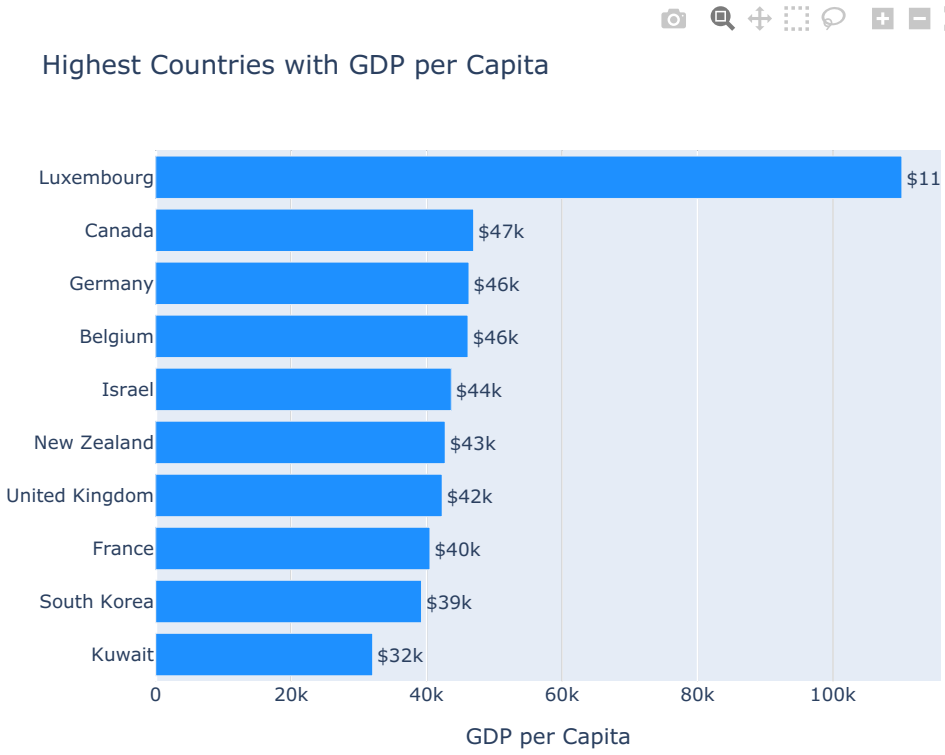
In [27]:
```
fig = make_subplots(rows = 1, cols = 2)

scatter1 = px.scatter(world, x = 'birth_rate', y = 'fertility_rate', trendline='ols', hover_data=world[['countr
scatter1_t = scatter1.data

for t in scatter1_t:
    fig.add_trace(go.Scatter(t, marker_color = 'darkseagreen'), row=1, col=1)

fig.update_xaxes(title_text = 'Birth Rate', row=1, col=1)
fig.update_yaxes(title_text = 'Fertility Rate', row=1, col=1)

scatter2 = px.scatter(world, x = 'birth_rate', y = 'infant_mortality',trendline='ols', hover_data=world[['count
scatter2_t = scatter2.data

for t in scatter2_t:
    fig.add_trace(go.Scatter(t, marker_color = 'darkslateblue'), row=1, col=2)

fig.update_xaxes(title_text='Birth Rate', row=1, col=2)
fig.update_yaxes(title_text='Infant Mortality', row=1, col=2)

fig.update_layout(title_text = 'Birth Rate vs Fertility Rate and Infant Mortality')

fig.show()
```



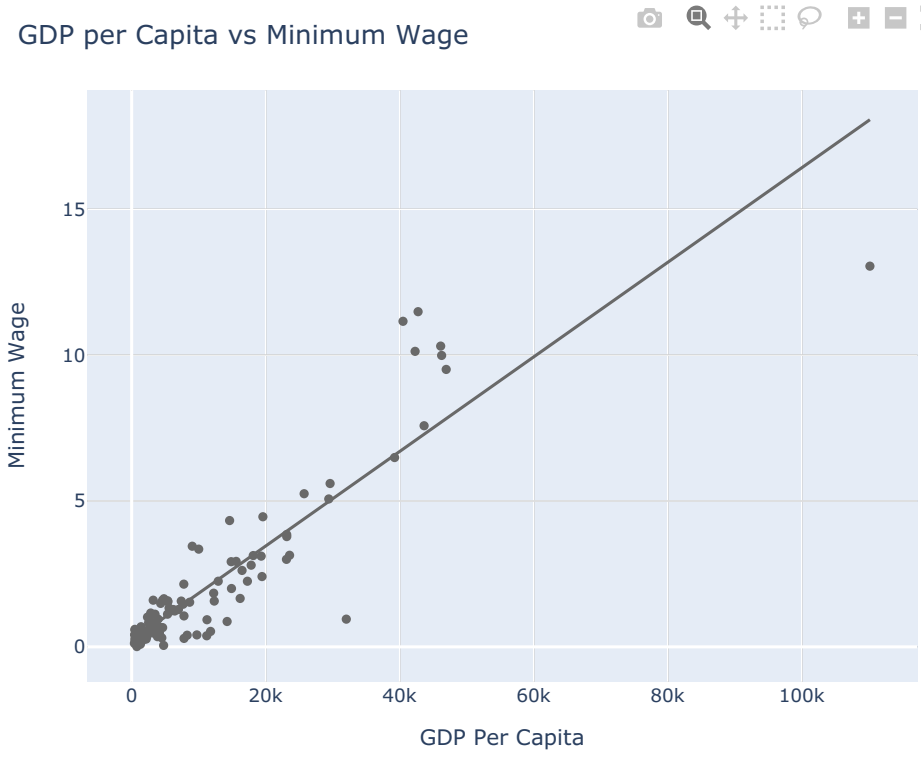
• It's evident that both Fertility Rate and Infant Mortality are positively correlated with the Birth Rate.

In [28]:
```
#creating GDP per capita column for each column

world['gdp_per_capita'] = world['gdp']/world['population']
```

In [29]:
```
gdp = world[['country', 'gdp_per_capita']].sort_values(by='gdp_per_capita', ascending = False)[:10]
fig = go.Figure(go.Bar(x = gdp['gdp_per_capita'][::-1],
                       y = gdp['country'][::-1],
                         orientation='h',
                        marker = dict(color = 'dodgerblue'),
                        customdata= world['gdp'],
                        text= gdp['gdp_per_capita'].apply(lambda x:'$'+str(round(x/1000))+'k')[::-1],
                        textposition='outside'
))

fig.update_traces(
    hovertemplate = '%{y}<br>GDP per capita:%{x}<br>GDP:%{customdata}$<br>')

fig.update_layout(
    title = "Highest Countries with GDP per Capita",
    xaxis_title = 'GDP per Capita',
    #yaxis_title = 'Country'
)
fig.show()
```



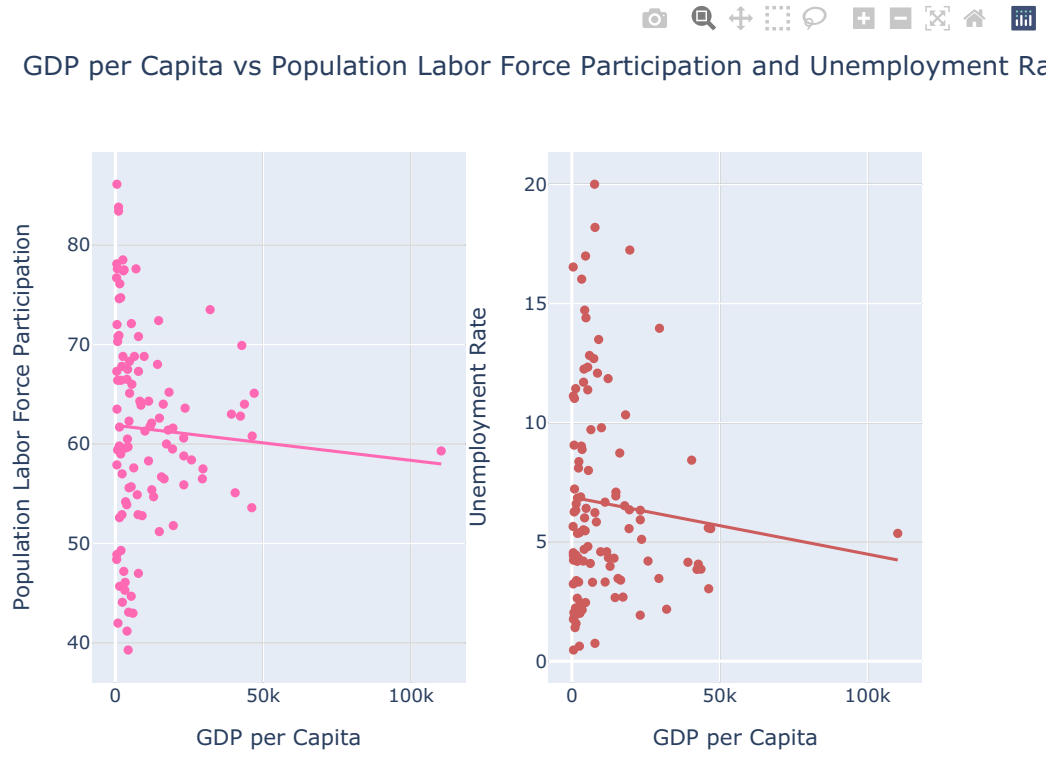
In [30]:
```
fig = px.scatter(world, x = 'gdp_per_capita', y = 'minimum_wage', trendline = 'ols', hover_data=world[['country

fig.update_traces(marker = dict(color = 'dimgray'),
                  hovertemplate='%{customdata[0]}<br>GDP per Capita: %{x}<br>Minimum Wage: %{y}')

fig.update_layout(
    title = 'GDP per Capita vs Minimum Wage',
    xaxis_title = 'GDP Per Capita',
    yaxis_title = 'Minimum Wage'
)
fig.show()
```



• Luxembourg has highest GDP per capita over $110k

• Based on the available data, it can be concluded that GDP per capita and minimum wage are positively correlated.

In [31]:
```
fig = make_subplots(rows = 1, cols = 2)

scatter1 = px.scatter(world, x = 'gdp_per_capita', y = 'population_labor_force_participation',trendline='ols',
                      hover_data=world[['country']])
scatter1_t = scatter1.data

for t in scatter1_t:
    fig.add_trace(go.Scatter(t,
                             marker_color = 'hotpink',
                            hovertemplate='%{customdata[0]}<br>GDP per Capita:$ %{x}<br>Population Labor Force
                              row=1, col=1)

fig.update_xaxes(title_text='GDP per Capita', row=1, col=1)
fig.update_yaxes(title_text='Population Labor Force Participation', row=1, col=1)

scatter2 = px.scatter(world, x = 'gdp_per_capita', y = 'unemployment_rate',trendline='ols',
                      hover_data=world[['country']])
scatter2_t = scatter2.data

for t in scatter2_t:
    fig.add_trace(go.Scatter(t,
                             marker_color = 'indianred',
                            hovertemplate='%{customdata[0]}<br>GDP per Capita:$%{x}<br>Unemployment Rate:%{y}')
                              row=1, col=2)

fig.update_xaxes(title_text='GDP per Capita', row=1, col=2)
fig.update_yaxes(title_text='Unemployment Rate', row=1, col=2)

fig.update_layout(title_text = 'GDP per Capita vs Population Labor Force Participation and Unemployment Rate')

fig.show()
```



•Based on the available data, there appears to be no significant relationship between population Labor Force Participation and unemployment rate with GDP per capita.