Unsupervised Learning: KMeans Clustering with Country Dataset

DeAundrie Howard

The problem statement from Kaggle: "HELP International have been able to raise around $ 10 million. Now the CEO of the NGO needs to decide how to use this money strategically and effectively. So, CEO has to make decision to choose the countries that are in the direst need of aid. Hence, your Job as a Data scientist is to categorise the countries using some socio-economic and health factors that determine the overall development of the country. Then you need to suggest the countries which the CEO needs to focus on the most".

Load the Libraries

In [1]:
```
# General
import pandas as pd
pd.set_option('display.max_columns', None)
import numpy as np

# Visualization
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set(style="whitegrid")
import plotly.express as px
from plotly.subplots import make_subplots
colors = ['#DB1C18','#DBDB3B','#51A2DB']
sns.set(palette=colors, font='Serif', style='white', rc={'axes.facecolor':'whitesmoke'

# Models
from sklearn.cluster import KMeans
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

Load the Dataset

In [3]:
```
# Specify location of the dataset
filename = 'Country-data.csv'
# Load the data into a Pandas DataFrame
df = pd.read_csv(filename)
```

Loading the data dictionary

- It is always a good idea to insert the dictionary if you can find a data dictionary included with the dataset.

- If one is not included with the data set, it is always good practice to include some information about the variables.
- If a data dictionary is included, upload the file into the same folder that you keep your dataset so you can easily refer to the file.

In [4]:
```
# Specify location of the dataset
filename2 = 'data-dictionary.csv'
# Load the data into a Pandas DataFrame
data_dict = pd.read_csv(filename2)
```

In [5]:
```
data_dict
```

Out[5]:

	Column Name	Description
0	country	Name of the country
1	child_mort	Death of children under 5 years of age per 100...
2	exports	Exports of goods and services per capita. Give...
3	health	Total health spending per capita. Given as %ag...
4	imports	Imports of goods and services per capita. Give...
5	Income	Net income per person
6	Inflation	The measurement of the annual growth rate of t...
7	life_expec	The average number of years a new born child w...
8	total_fer	The number of children that would be born to e...
9	gdpp	The GDP per capita. Calculated as the Total GD...

Non-graphical EDA

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 167 entries, 0 to 166
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   country     167 non-null    object 
 1   child_mort  167 non-null    float64
 2   exports     167 non-null    float64
 3   health      167 non-null    float64
 4   imports     167 non-null    float64
 5   income      167 non-null    int64  
 6   inflation   167 non-null    float64
 7   life_expec  167 non-null    float64
 8   total_fer   167 non-null    float64
 9   gdpp        167 non-null    int64  
dtypes: float64(7), int64(2), object(1)
memory usage: 13.2+ KB
```

In [7]:
```
df.shape
```

Out[7]: (167, 10)

Question 1: Notice the .T after the describe () and then look back to your homework. What is the difference? Add a code block below to answer question. Be sure that the new code block is run as markdown and not code.

The .T after describe() mean to transpose. In other words, the dataframe is being transposed by reflecting the records and fields across the diagonal axis.

In [8]:
```
df.describe().T
```

Out[8]:

	count	mean	std	min	25%	50%	75%	max
child_mort	167.0	38.270060	40.328931	2.6000	8.250	19.30	62.10	208.00
exports	167.0	41.108976	27.412010	0.1090	23.800	35.00	51.35	200.00
health	167.0	6.815689	2.746837	1.8100	4.920	6.32	8.60	17.90
imports	167.0	46.890215	24.209589	0.0659	30.200	43.30	58.75	174.00
income	167.0	17144.688623	19278.067698	609.0000	3355.000	9960.00	22800.00	125000.00
inflation	167.0	7.781832	10.570704	-4.2100	1.810	5.39	10.75	104.00
life_expec	167.0	70.555689	8.893172	32.1000	65.300	73.10	76.80	82.80
total_fer	167.0	2.947964	1.513848	1.1500	1.795	2.41	3.88	7.49
gdpp	167.0	12964.155689	18328.704809	231.0000	1330.000	4660.00	14050.00	105000.00

Question 2: In the next two code blocks, you notice the commands df.isnull and df.isna. What is the difference between the two? Add a code block below to answer question. Be sure that the new code block is run as markdown and not code.

There is not a difference. They both detect missing values; however, isnull is generally preferred.

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
country       0
child_mort    0
exports       0
health        0
imports       0
income        0
inflation     0
life_expec    0
total_fer     0
gdpp          0
dtype: int64
```

In [10]:
```
df.isna().sum()
```

Out[10]:
```
country       0
child_mort    0
exports       0
health        0
imports       0
income        0
inflation     0
life_expec    0
total_fer     0
gdpp          0
dtype: int64
```

In [11]:
```
df.head()
```

Out[11]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4460
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	3530
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200

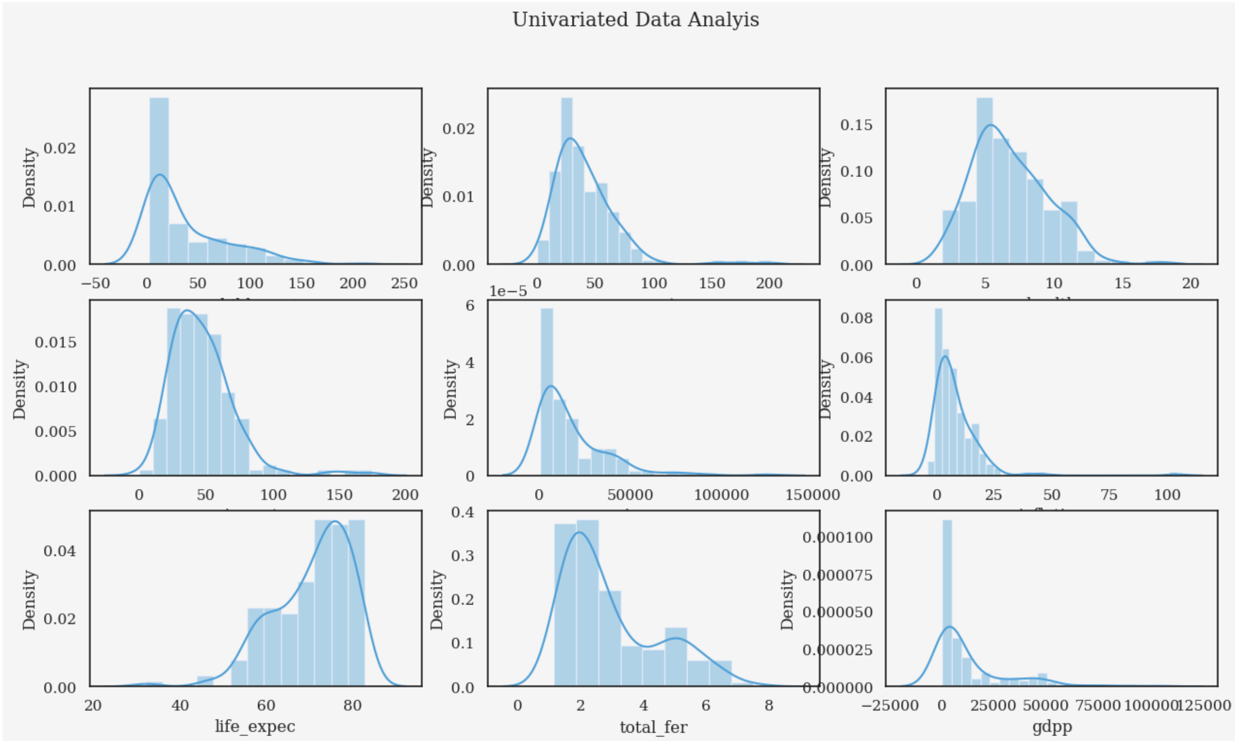
In [12]:
```
df['country'].count()
```

Out[12]:
```
167
```

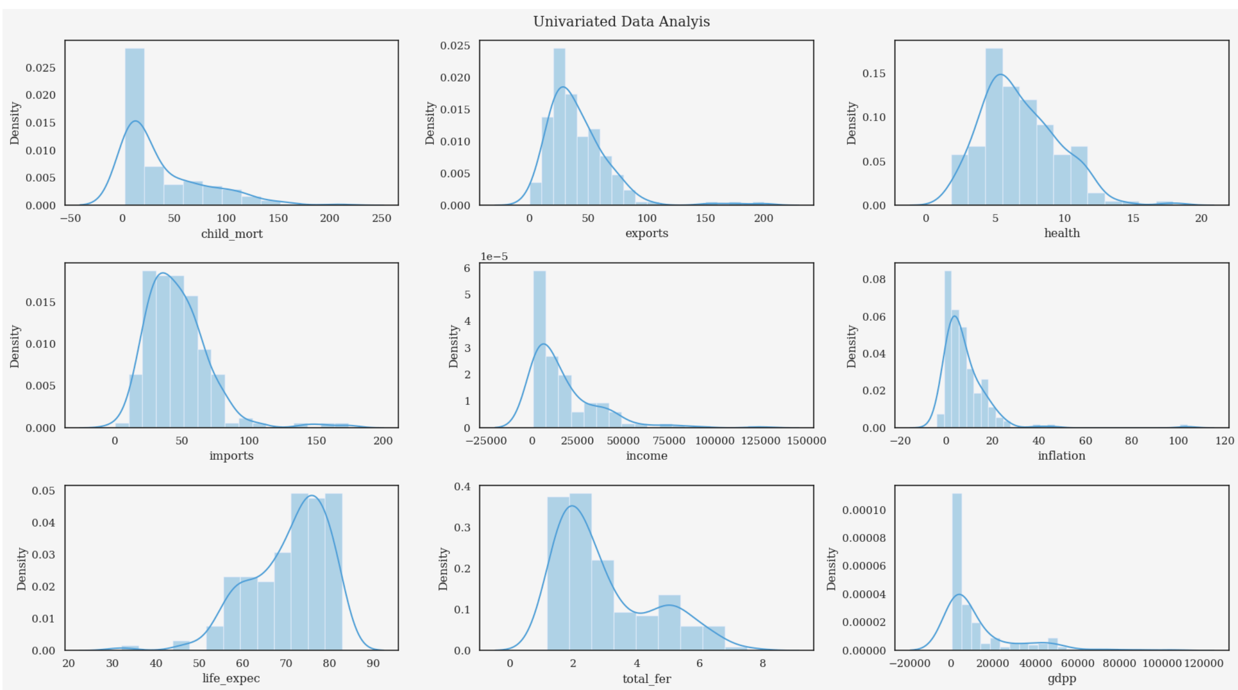
Graphical EDA

Question 3: In the code block below, how would you fix the size of the figure so you could see the labels for all variables? Add a code block below to answer question. Be sure that the new code block is run as code and not markdown.

In [13]:
```
fig, ax = plt.subplots(nrows=3,ncols=3, figsize=(15,8))
plt.suptitle("Univariated Data Analyis")
ax=ax.flatten()
int_cols= df.select_dtypes(exclude='object').columns
for x, i in enumerate(int_cols):
    sns.distplot(df[i], ax=ax[x], kde=True, color=colors[2])
```



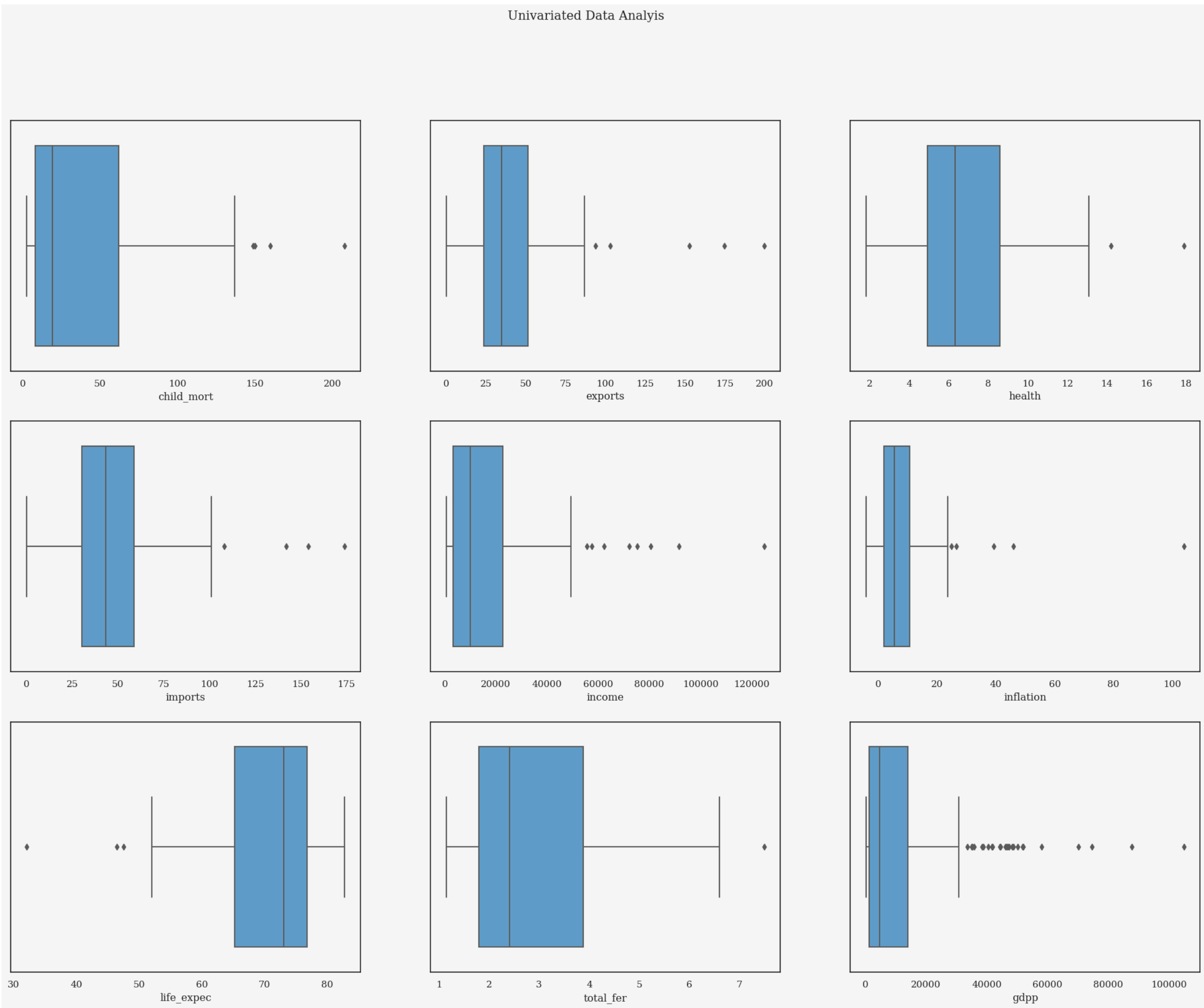
In [14]:
```
fig, ax = plt.subplots(nrows=3,ncols=3, figsize=(18,10))
plt.suptitle("Univariated Data Analyis")
ax=ax.flatten()
int_cols= df.select_dtypes(exclude='object').columns
for x, i in enumerate(int_cols):
    sns.distplot(df[i], ax=ax[x], kde=True, color=colors[2])
    fig.tight_layout()
```



Question 4: In the code block below, what do the marks above or below the wiskers of the boxplot represent? Add a code block below to answer question. Be sure that the new code block is run as markdown and not code.

The marks above or below the wiskers of the boxplot represent the outliers.

In [15]:
```
fig, ax = plt.subplots(nrows=3,ncols=3, figsize=(25,18))
plt.suptitle("Univariated Data Analyis")
ax=ax.flatten()
int_cols= df.select_dtypes(exclude='object').columns
for x, i in enumerate(int_cols):
    sns.boxplot(x=df[i], ax=ax[x], color=colors[2])
```

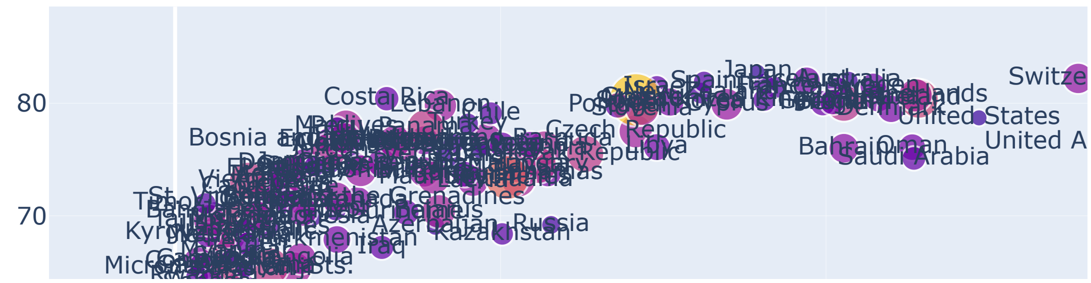


In [16]:
```
px.scatter(data_frame=df, x='exports', y='imports',size='gdpp', text='country', color=
```

Question 5: Using the code above, how would you change the x value to "income", the y value to "life_expec", and the size and color to "imports"? Add a code block below to answer the question by copying and pasting the code from the code block above and entering the given data. Ensure the new code block is run as **code and not markdown.** Don't forget to run the code.

In [17]:
```
px.scatter(data_frame=df, x='income', y='life_expec', size='imports', text='country',
```

Countries Income and Life Expectancy and corresponding Imp

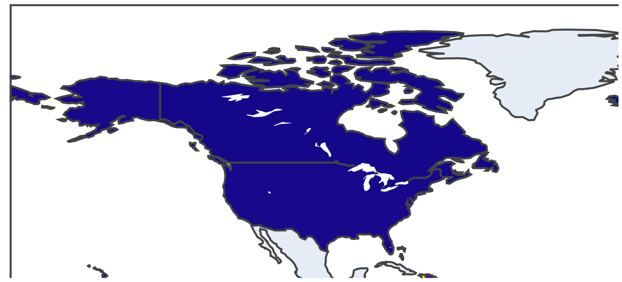


Question 6: Using the same graph that you used for question 5, what is the graph illustrating? Add a code block below to answer question. Be sure that the new code block is run as markdown and not code.

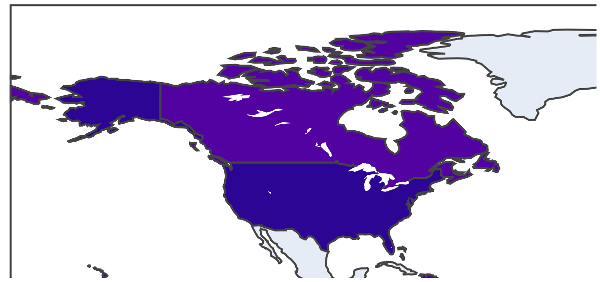
The graph directly above is illustrating that the greater the imports, the greater the income, the greater the income, the greater the life expectancy.

In [18]:
```
for i in int_cols:
    fig=px.choropleth(data_frame=df, locationmode='country names', locations='country'
    fig.show()
```

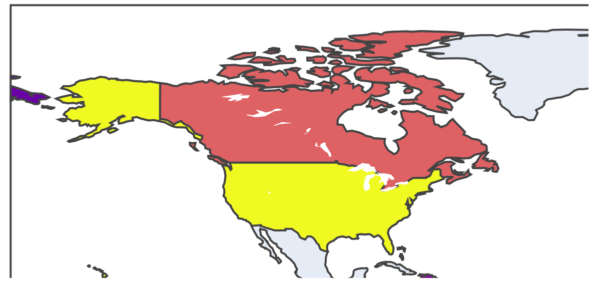
child_mort rate by countries



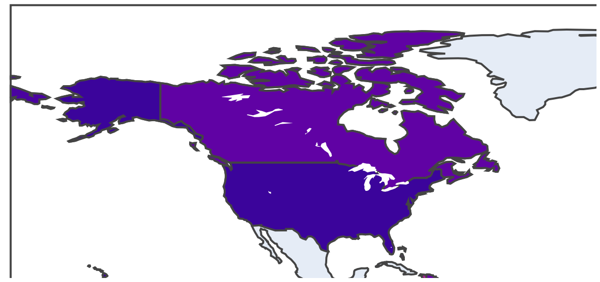
exports rate by countries



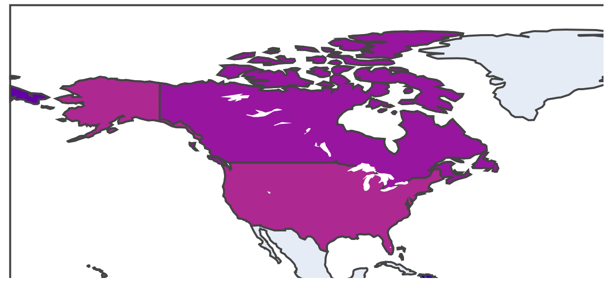
health rate by countries



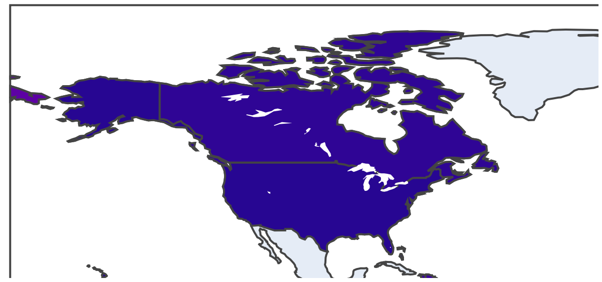
imports rate by countries



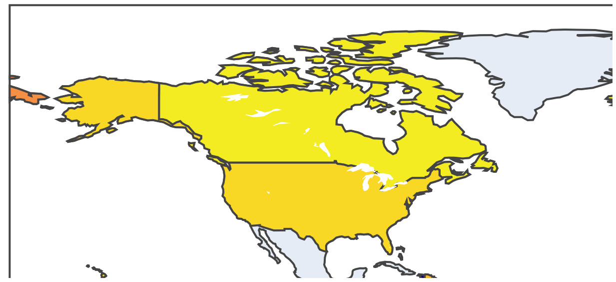
income rate by countries



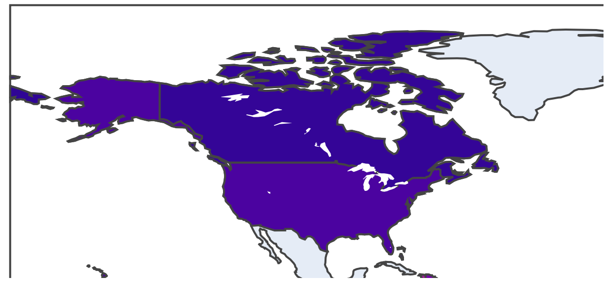
inflation rate by countries



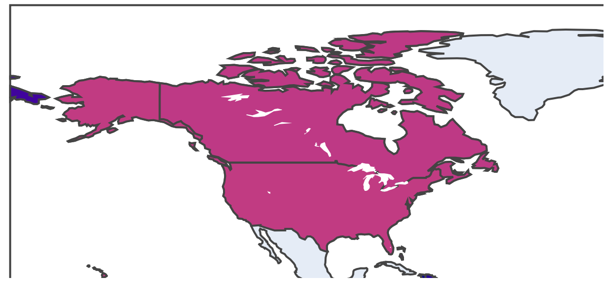
life_expec rate by countries



total_fer rate by countries

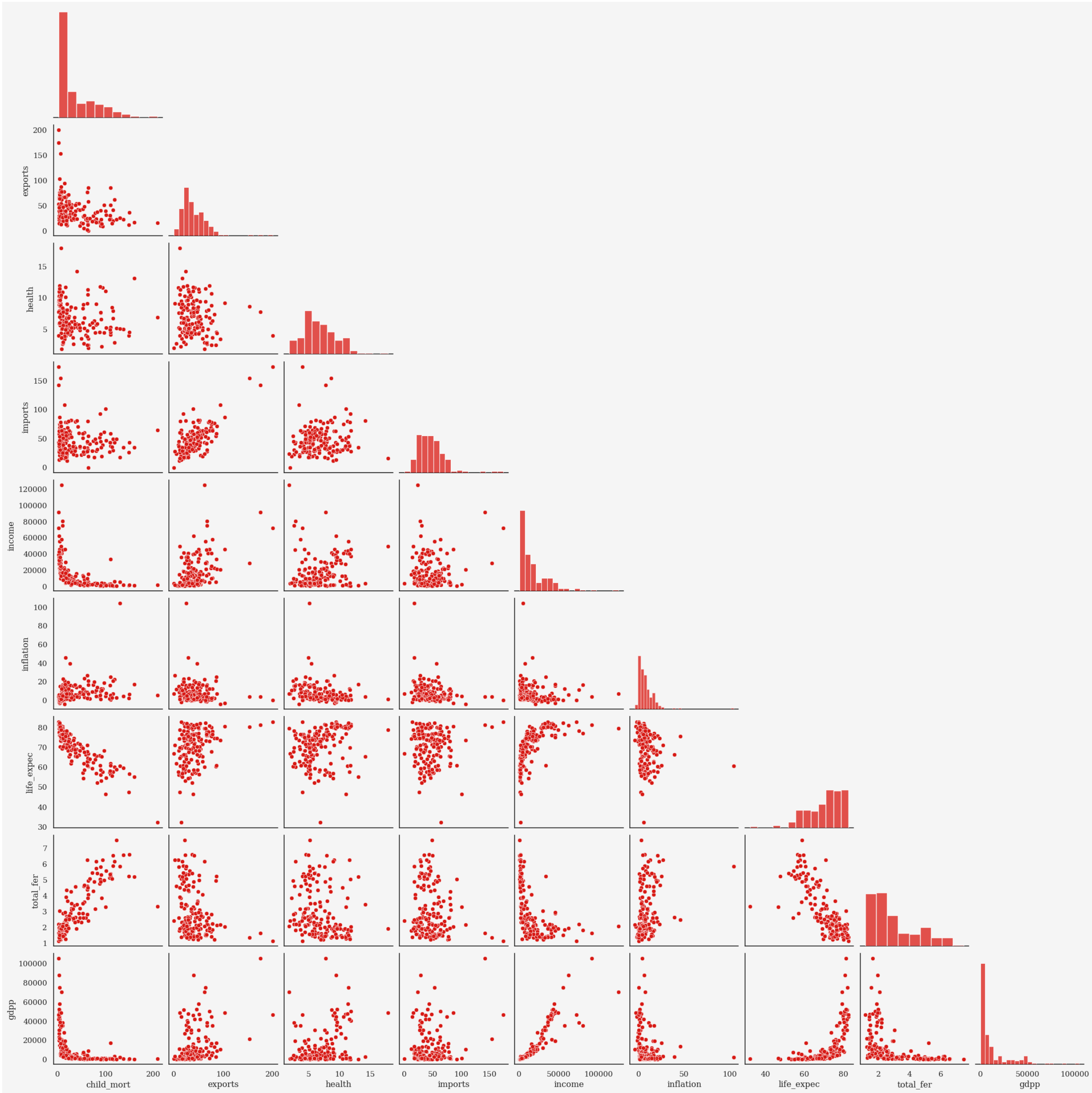


gdpp rate by countries



In [19]:
```
sns.pairplot(df, corner =True)
```

Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x198677d36d0>
```



Question 7: Looking at the next two code blocks, which graph do you prefer and why? There is no right or wrong answer. However, you will be graded on your ability to demonstrate that you understand what is going on in the graphs and are to explain them. Add a code block below to answer the question. Ensure the new code block is run as markdown and not code.

I prefer the second graph. Despite the first graph showing me the positive and negative correlations between every variable, which can be important, the second graph only shows variables with a positive correlation greater than 50% with another variable.

Question 8: Looking at the heatmap below, which variables have the highest correlation with each other? Name the top 5. An example would be child_mort, which has a high correlation with exports, with a value of -0.32 (this is not a correct answer).

However, it is important to understand what is going on in the graphs and be able to explain them. Add a code block below to answer the question. Be sure that the new code block is run as markdown and not code.

Top 5 Highest Correlations:

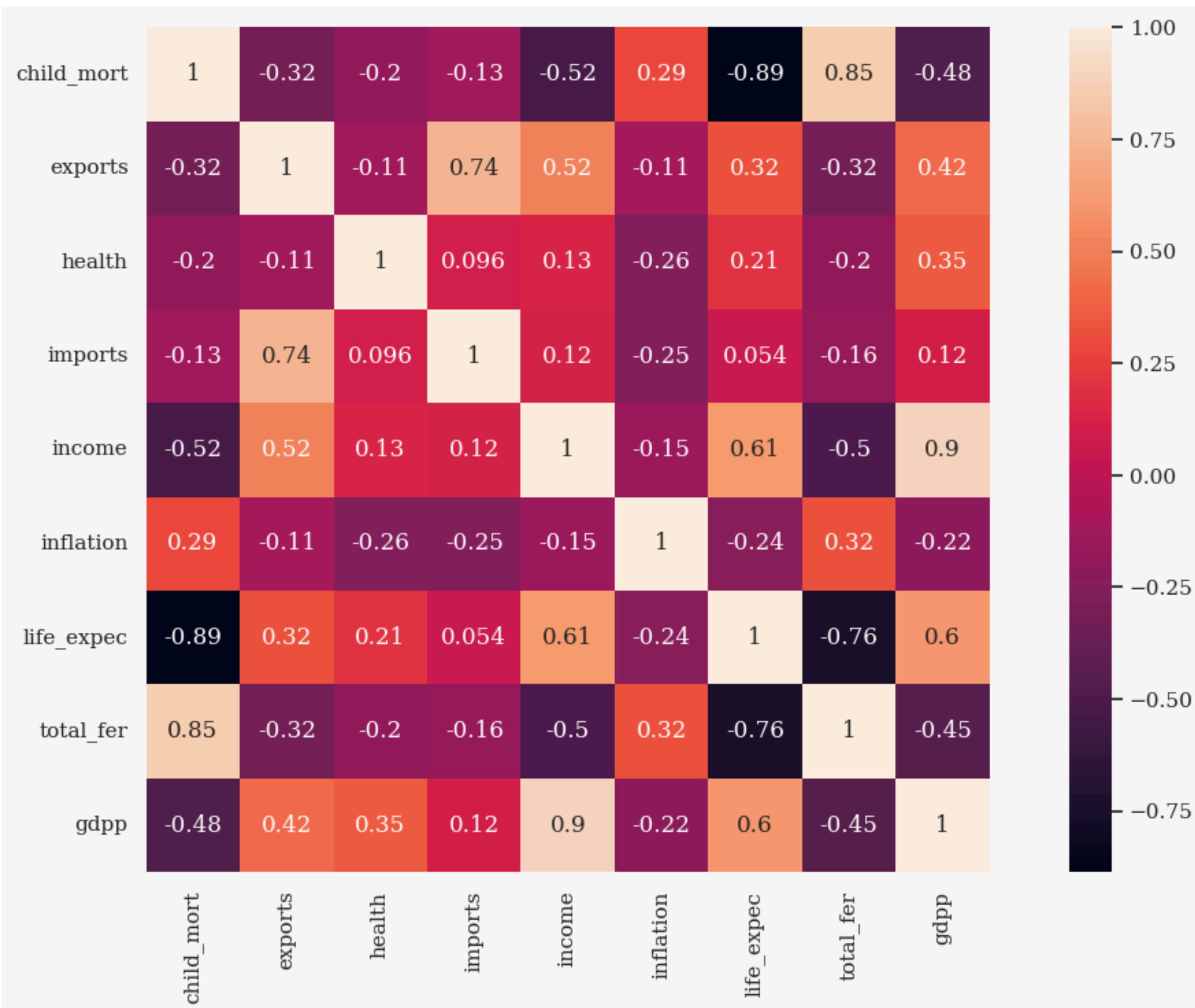
1. Income and Gdpp: 0.9; high positve correlation
2. child_mort and life_expec: -0.89; high negative correlation
3. child_mort and totla_fer: 0.85; high positive correlation
4. life_expec and total_fer: -0.76; high negative correlation
5. Imports and exports: 0.74; high positive correlation

In [20]:

```
df2 = df.iloc[: , 1:]
fig=plt.figure(figsize=(15,8))
sns.heatmap(df2.corr(), annot=True, square=True)
```

Out[20]:

```
<Axes: >
```



Question 9: Looking at these two correlation graphs, what variables would you drop from the dataset and why? Add a code

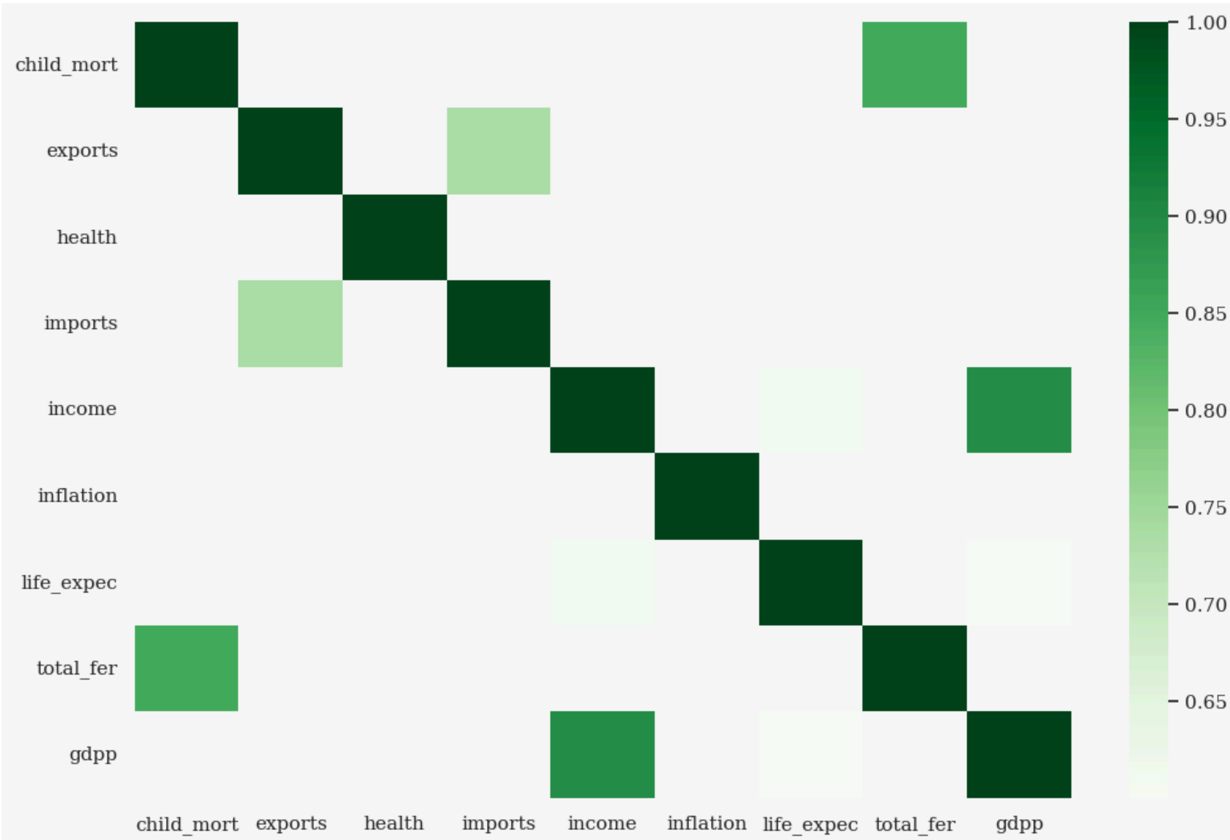
block below to answer question. Be sure that the new code block is run as markdown and not code.

Based on the two correlations graphs below, I would drop health and inflation as neither show a somewhat positive correlation with any other variable.

In [21]:
```
corr = df2.corr()

kot = corr[corr>=.6]
plt.figure(figsize=(12,8))
sns.heatmap(kot, cmap="Greens")
```

Out[21]: `<Axes: >`



We did not do scaling in the homework but I wanted to let you know this is how you you make variables that have different scales contribute equally with one another. This helps neutralize bias in the model.

In [22]:
```
from sklearn.preprocessing import StandardScaler
df_scaled = StandardScaler().fit_transform(df.drop(['country'], axis=1))
```

We also did not go into PCA or Pricipal Component Analysis in your homework but I want to let you know that it is a way that you can reduce dimensionality or variables in the dataset. This is a technique that transforms a large set of variables into a smaller set of variables. The important thing to remember is that you

don't want to reduce the size of the variables so much that you also decrease the accuracy.

In [23]:
```
from sklearn.decomposition import PCA
decom = PCA(svd_solver='auto')
decom.fit(df_scaled)
```

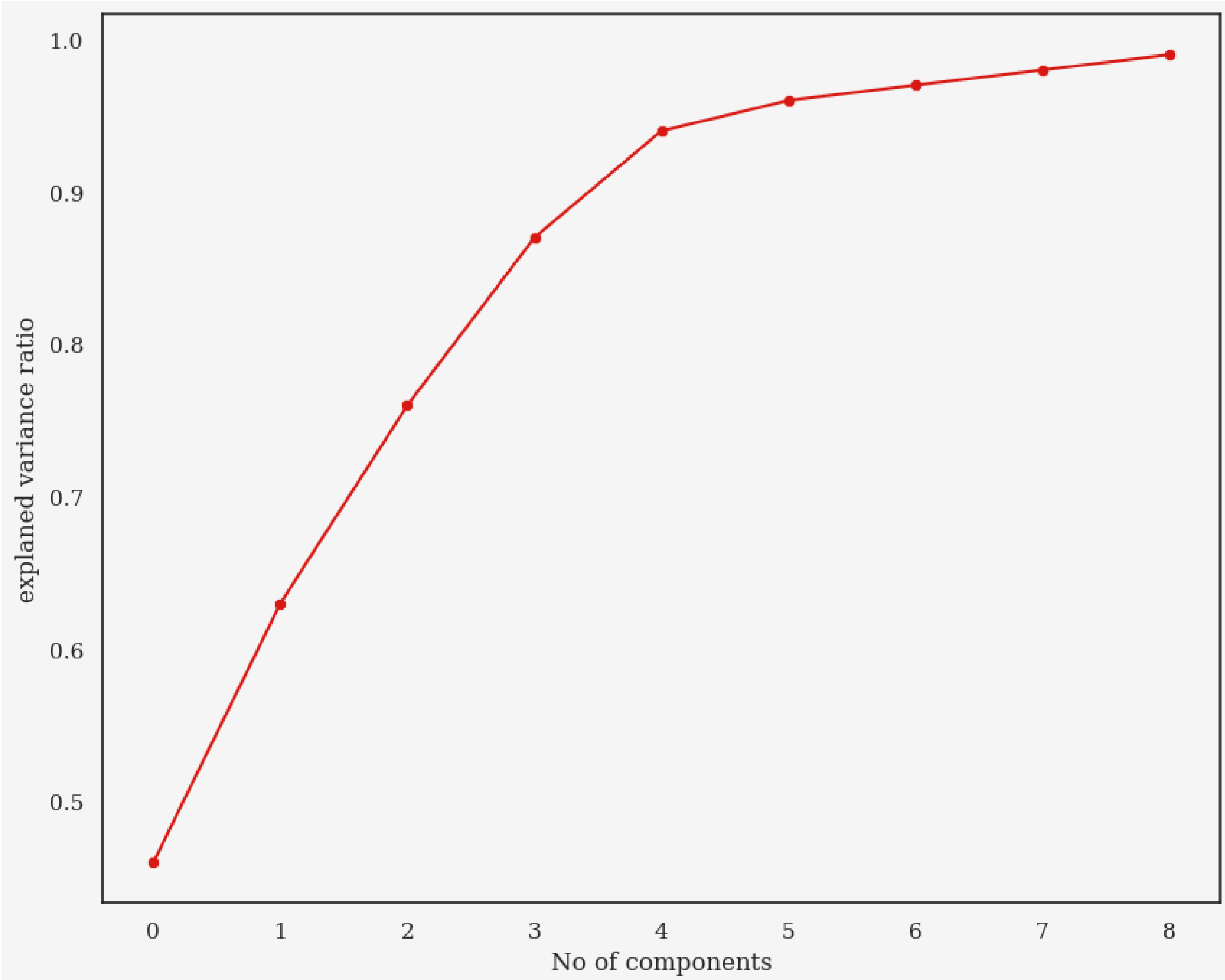
Out[23]:
```
▾ PCA
PCA()
```

This is one last concept I want you to be aware of without getting too detailed. The explained variance (see the numbers in parentheses below the code block: 0.46, 0.63, 0.76...), these numbers explain the variance that is attributed to the model by each component. It is a good rule of thumb to continue adding components until you reach around 0.8 or 80%. In this example, that means using 3 (0.76) or 4 (0.87) components.

In [24]:
```
cum_exp_ratio = np.cumsum(np.round(decom.explained_variance_ratio_,2))
print(cum_exp_ratio)
fig=plt.figure(figsize=(10,8))
ax=sns.lineplot(y=cum_exp_ratio, x=np.arange(0,len(cum_exp_ratio)))
ax=sns.scatterplot(y=cum_exp_ratio, x=np.arange(0,len(cum_exp_ratio)))
ax.set_xlabel('No of components')
ax.set_ylabel('explaned variance ratio')
```

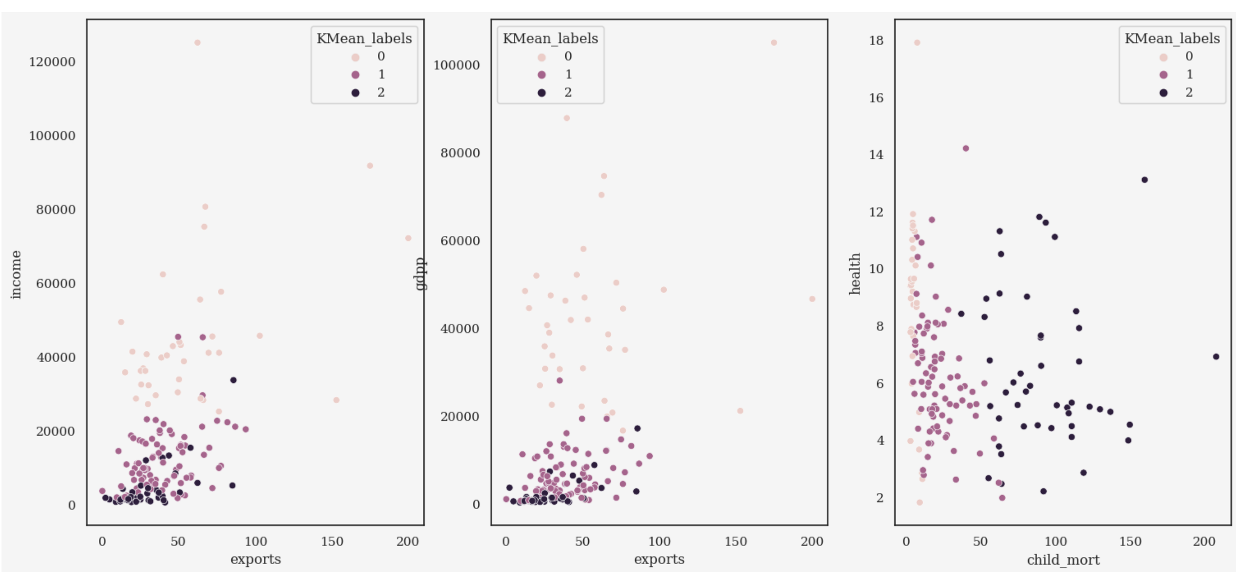
```
[0.46 0.63 0.76 0.87 0.94 0.96 0.97 0.98 0.99]
```

Out[24]:
```
Text(0, 0.5, 'explaned variance ratio')
```



In [25]:
```
model = KMeans(n_clusters=3, random_state=1)
model.fit(df_scaled)
df['KMean_labels']=model.labels_
fig,ax=plt.subplots(nrows=1,ncols=3,figsize=(18,8))
sns.scatterplot(data=df, x='exports', y='income', hue='KMean_labels', ax=ax[0])
sns.scatterplot(data=df, x='exports', y='gdpp', hue='KMean_labels', ax=ax[1])
sns.scatterplot(data=df, x='child_mort', y='health', hue='KMean_labels', ax=ax[2])
```

Out[25]:
```
<Axes: xlabel='child_mort', ylabel='health'>
```



In [26]:
```
df.groupby(['KMean_labels','country']).mean()
```

Out[26]:

KMean_labels	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer
0	**Australia**	4.8	19.8	8.73	20.9	41400.0	1.160	82.0	1.93
	Austria	4.3	51.3	11.00	47.8	43200.0	0.873	80.5	1.44
	Bahrain	8.6	69.5	4.97	50.9	41100.0	7.440	76.0	2.16
	Belgium	4.5	76.4	10.70	74.7	41100.0	1.880	80.0	1.86
	Brunei	10.5	67.4	2.84	28.0	80600.0	16.700	77.1	1.84
...	**...**	...	...	...	...	...	...	...	...
2	**Timor-Leste**	62.6	2.2	9.12	27.8	1850.0	26.500	71.1	6.23
	Togo	90.3	40.2	7.65	57.3	1210.0	1.180	58.7	4.87
	Uganda	81.0	17.1	9.01	28.6	1540.0	10.600	56.8	6.15
	Yemen	56.3	30.0	5.18	34.4	4480.0	23.600	67.5	4.67
	Zambia	83.1	37.0	5.89	30.9	3280.0	14.000	52.0	5.40

167 rows × 9 columns

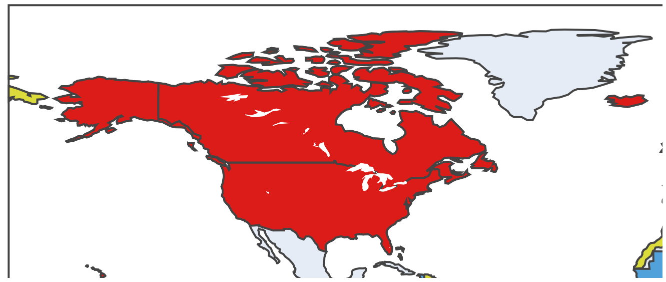
Question 10: Looking at the next code block, what are your thoughts on what the graph is illustrating? There is no right or wrong answer. However, you will be graded on your ability to demonstrate that you understand what is going on in the graph and to explain it. Add a code block below to answer the question. Ensure the new code block is run as markdown and not code.

Based on the graph below, is a graphical represenation of a k-means clustering algorithm, categorizing each country on their level of need based on their sum of data points withins each country (cluster).

In [27]:
```
#df['KMean_labels']=df['KMean_labels'].astype('category')
cat = {0:'Need Help',1:'Might need help',2:'No Help needed'}
df['KMean_labels']=df['KMean_labels'].map(cat)

px.choropleth(data_frame=df, locationmode='country names', locations='country', color=
              color_discrete_map={'Need Help':'#DB1C18','Might need help':'#DBDB3B','N
```

Countries by category that need help

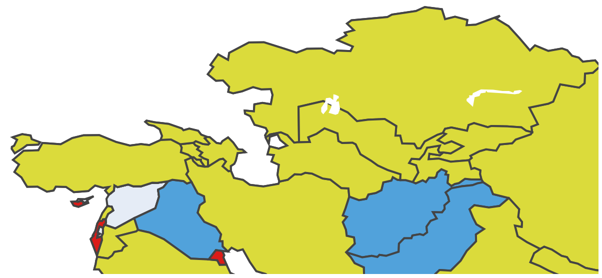


Question 11: Looking at the graph below, what are your thoughts on this type of geographical information? There is no right or wrong answer. However, you will be graded on your ability to demonstrate that you understand what is going on in the graph and to explain it. Add a code block below to answer the question. Ensure the new code block is run as markdown and not code.

The geographical information below shows us that out of the 42 Asian Countries displayed, 7 countries are in need of help, 27 countries might need help, 6 countries do not need help, and 2 countries have insufficient data to report.

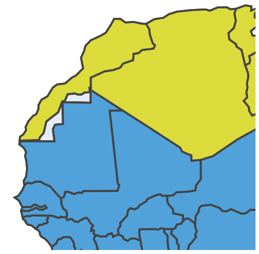
In [28]:
```
px.choropleth(data_frame=df, locationmode='country names', locations='country', color=
              color_discrete_map={'Need Help':'#DB1C18','Might need help':'#DBDB3B','N
```

Asian Countries by category that need help



In [29]:
```
px.choropleth(data_frame=df, locationmode='country names', locations='country', color=
              color_discrete_map={'Need Help':'#DB1C18','Might need help':'#DBDB3B','N
```

African Countries by category that need help



In [30]:
```
df[df['KMean_labels']=='Need Help']['country']
```

Out[30]:

```
7                    Australia
8                      Austria
11                     Bahrain
15                     Belgium
23                      Brunei
29                      Canada
42                      Cyprus
43              Czech Republic
44                     Denmark
53                     Finland
54                      France
58                     Germany
60                      Greece
68                     Iceland
73                     Ireland
74                      Israel
75                       Italy
77                       Japan
82                      Kuwait
91                  Luxembourg
98                       Malta
110                Netherlands
111                New Zealand
114                     Norway
122                   Portugal
123                      Qatar
133                  Singapore
134            Slovak Republic
135                   Slovenia
138                South Korea
139                      Spain
144                     Sweden
145                Switzerland
157       United Arab Emirates
158             United Kingdom
159              United States
Name: country, dtype: object
```

Dr. Randall and TAs, thank you for such a great semester! I learn more in the last 8 weeks about data analytic/science than I could have imagined! I wish you the best in all of your personal and professional endeavors.

-DeAundrie Howard.

In [ ]: