In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_csv("G:\Sagar\College\Machine Learning 21-22\Lab\Lab 6 K Means\Country_data.cs
```

In [3]:

```
df.head()
```

Out[3]:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gc
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	ε
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4(
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4⁄
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	3ε
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12ź

In [4]:

```
df.describe()
```

Out[4]:

	child_mort	exports	health	imports	income	inflation	life_expec
count	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000	167.000000
mean	38.270060	41.108976	6.815689	46.890215	17144.688623	7.781832	70.555689
std	40.328931	27.412010	2.746837	24.209589	19278.067698	10.570704	8.893172
min	2.600000	0.109000	1.810000	0.065900	609.000000	-4.210000	32.100000
25%	8.250000	23.800000	4.920000	30.200000	3355.000000	1.810000	65.300000
50%	19.300000	35.000000	6.320000	43.300000	9960.000000	5.390000	73.100000
75%	62.100000	51.350000	8.600000	58.750000	22800.000000	10.750000	76.800000
max	208.000000	200.000000	17.900000	174.000000	125000.000000	104.000000	82.800000

In [5]:

```
df['child_mort'].isnull().sum()
```

Out[5]:

0

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.corr()
```

Out[7]:

	child_mort	exports	health	imports	income	inflation	life_expec	total_
child_mort	1.000000	-0.318093	-0.200402	-0.127211	-0.524315	0.288276	-0.886676	0.8484
exports	-0.318093	1.000000	-0.114408	0.737381	0.516784	-0.107294	0.316313	-0.3200
health	-0.200402	-0.114408	1.000000	0.095717	0.129579	-0.255376	0.210692	-0.1966
imports	-0.127211	0.737381	0.095717	1.000000	0.122406	-0.246994	0.054391	-0.1590
income	-0.524315	0.516784	0.129579	0.122406	1.000000	-0.147756	0.611962	-0.5018
inflation	0.288276	-0.107294	-0.255376	-0.246994	-0.147756	1.000000	-0.239705	0.3169
life_expec	-0.886676	0.316313	0.210692	0.054391	0.611962	-0.239705	1.000000	-0.7608
total_fer	0.848478	-0.320011	-0.196674	-0.159048	-0.501840	0.316921	-0.760875	1.0000
gdpp	-0.483032	0.418725	0.345966	0.115498	0.895571	-0.221631	0.600089	-0.4549

In [8]:

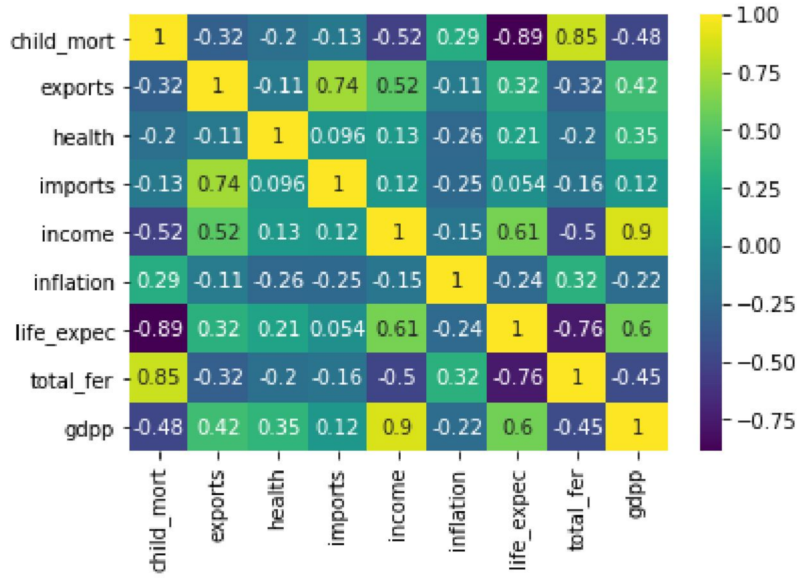
```
import seaborn as sns
```

In [9]:

```
sns.heatmap(df.corr(),annot=True,cmap='viridis')
```

Out[9]:

```
<AxesSubplot:>
```

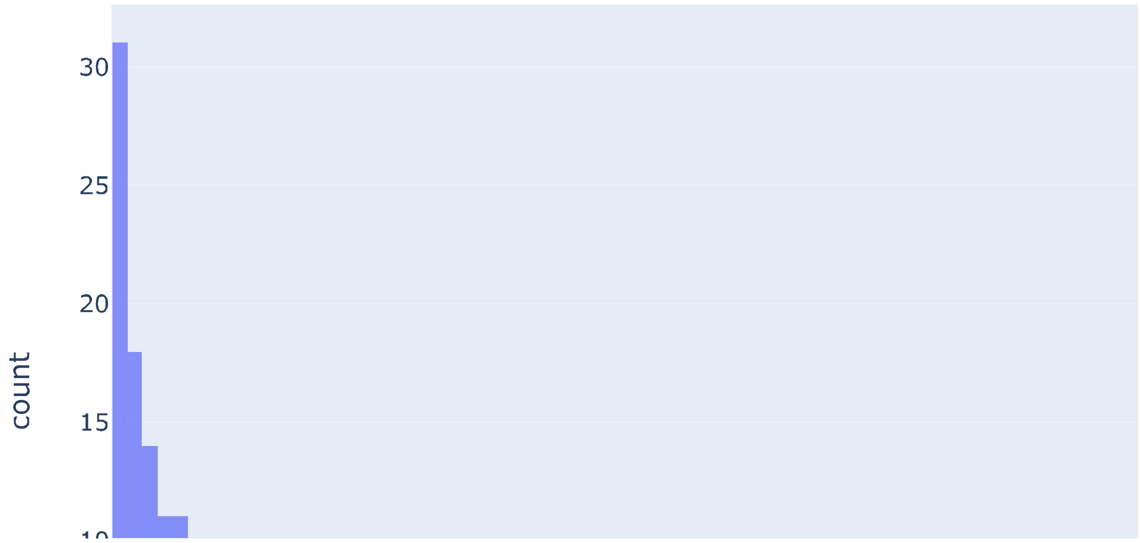


In [10]:

```
import plotly.express as exp
```

In [11]:

```
exp.histogram(data_frame=df,x = 'gdpp',nbins=167,opacity=0.75,barmode='overlay')
```



In [12]:

```
df['child_mort'].mean()
```

Out[12]:

```
38.270059880239515
```

In [13]:

```
df['child_mort'].max()
```

Out[13]:

```
208.0
```

In [14]:

```
df.drop('country',axis=1)
```

Out[14]:

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553
1	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090
2	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4460
3	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	3530
4	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200
...	...	...	...	...	...	...	...	...	...
162	29.2	46.6	5.25	52.7	2950	2.62	63.0	3.50	2970
163	17.1	28.5	4.91	17.6	16500	45.90	75.4	2.47	13500
164	23.3	72.0	6.84	80.2	4490	12.10	73.1	1.95	1310
165	56.3	30.0	5.18	34.4	4480	23.60	67.5	4.67	1310
166	83.1	37.0	5.89	30.9	3280	14.00	52.0	5.40	1460

167 rows × 9 columns

In [15]:

```
from sklearn.preprocessing import MinMaxScaler
```

In [16]:

```
scaler = MinMaxScaler()
```

In [17]:

```
scaled_data = scaler.fit_transform(df.drop('country',axis=1))
```

In [18]:

```
scaled_data
```

Out[18]:

```
array([[0.42648491, 0.04948197, 0.35860783, ..., 0.47534517, 0.73659306,
        0.00307343],
       [0.06815969, 0.13953104, 0.29459291, ..., 0.87179487, 0.07886435,
        0.03683341],
       [0.12025316, 0.1915594 , 0.14667495, ..., 0.87573964, 0.27444795,
        0.04036499],
       ...,
       [0.10077897, 0.35965101, 0.31261653, ..., 0.8086785 , 0.12618297,
        0.01029885],
       [0.26144109, 0.1495365 , 0.20944686, ..., 0.69822485, 0.55520505,
        0.01029885],
       [0.39191821, 0.18455558, 0.25357365, ..., 0.39250493, 0.670347  ,
        0.01173057]])
```

In [19]:

```
df.drop('country',axis=1).columns
```

Out[19]:

```
Index(['child_mort', 'exports', 'health', 'imports', 'income', 'inflation',
       'life_expec', 'total_fer', 'gdpp'],
      dtype='object')
```

In [20]:

```
data=pd.DataFrame(scaled_data,columns=df.drop('country',axis=1).columns)
```

In [21]:

```
data.head()
```

Out[21]:

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	0.426485	0.049482	0.358608	0.257765	0.008047	0.126144	0.475345	0.736593	0.003073
1	0.068160	0.139531	0.294593	0.279037	0.074933	0.080399	0.871795	0.078864	0.036833
2	0.120253	0.191559	0.146675	0.180149	0.098809	0.187691	0.875740	0.274448	0.040365
3	0.566699	0.311125	0.064636	0.246266	0.042535	0.245911	0.552268	0.790221	0.031488
4	0.037488	0.227079	0.262275	0.338255	0.148652	0.052213	0.881657	0.154574	0.114242

In [22]:

```
data['country'] = df['country']
```

In [23]:

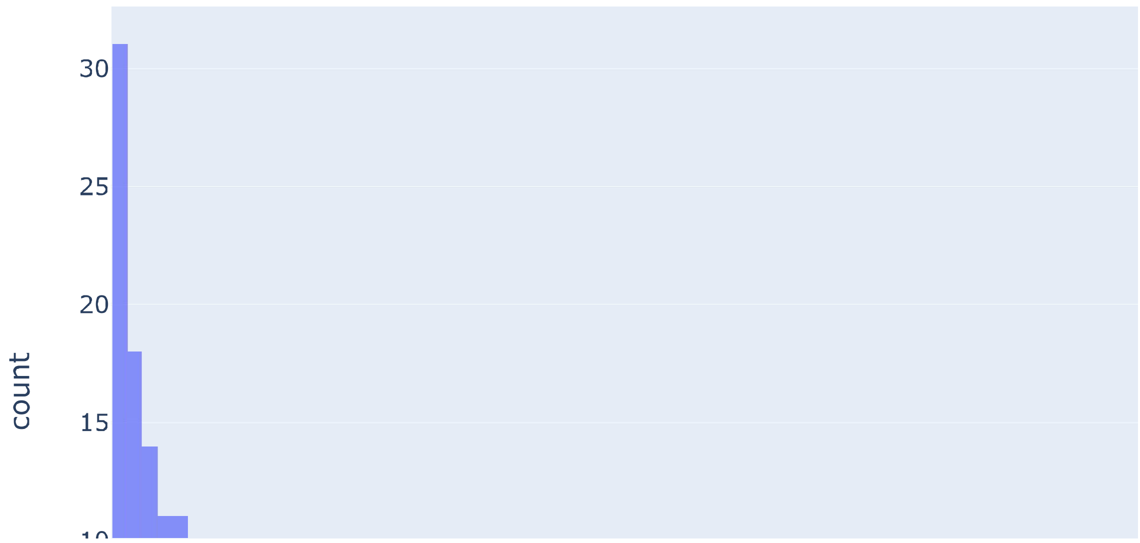
```
data.head()
```

Out[23]:

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	0.426485	0.049482	0.358608	0.257765	0.008047	0.126144	0.475345	0.736593	0.003073
1	0.068160	0.139531	0.294593	0.279037	0.074933	0.080399	0.871795	0.078864	0.036833
2	0.120253	0.191559	0.146675	0.180149	0.098809	0.187691	0.875740	0.274448	0.040365
3	0.566699	0.311125	0.064636	0.246266	0.042535	0.245911	0.552268	0.790221	0.031488
4	0.037488	0.227079	0.262275	0.338255	0.148652	0.052213	0.881657	0.154574	0.114242

In [24]:

```
exp.histogram(data_frame=df,x = 'gdpp',nbins=167,opacity=0.75,barmode='overlay')
```

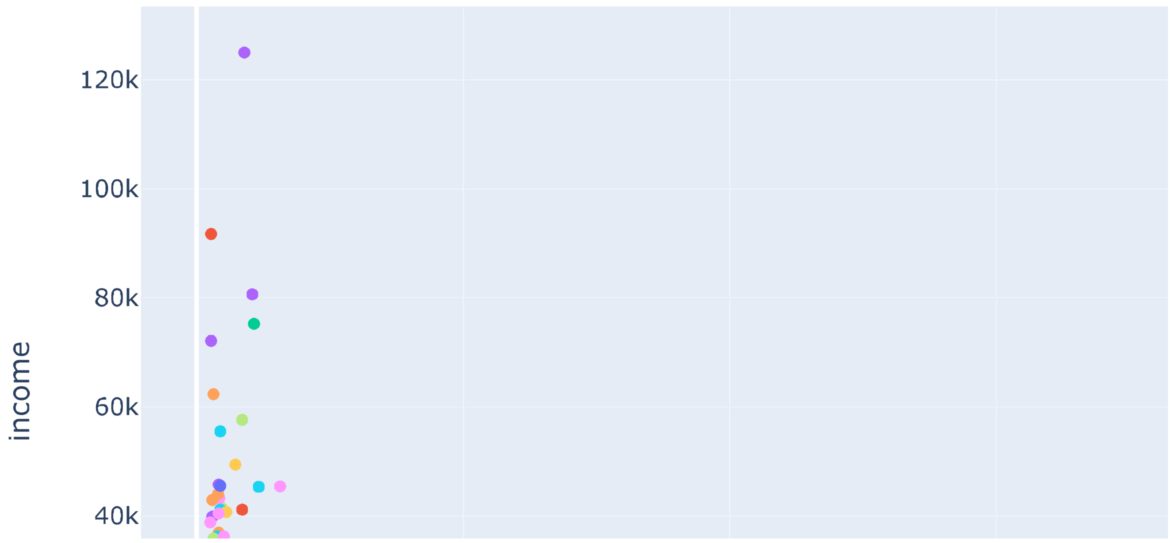


In [25]:

```
import matplotlib.pyplot as plt
```

In [26]:

```
exp.scatter(data_frame = df,x='child_mort',y='income',color='country')
```



In [27]:

```
from sklearn.cluster import KMeans
```

In [28]:

```
k_means =  KMeans(n_clusters=5)
```

In [29]:

```
k_means.fit(data.drop('country',axis=1))
```

Out[29]:

```
KMeans(n_clusters=5)
```

In [30]:

```
k_means.labels_
```

Out[30]:

```
array([3, 1, 1, 3, 1, 1, 1, 2, 2, 1, 1, 1, 0, 1, 1, 2, 1, 3, 1, 0, 1, 0,
       1, 2, 1, 3, 3, 0, 3, 2, 1, 3, 3, 1, 1, 1, 0, 3, 0, 1, 3, 1, 2, 1,
       2, 1, 1, 0, 1, 3, 0, 1, 0, 2, 2, 0, 3, 1, 2, 0, 2, 1, 0, 3, 3, 0,
       3, 1, 2, 0, 0, 1, 0, 2, 2, 2, 1, 2, 1, 1, 0, 0, 2, 0, 0, 1, 1, 3,
       3, 1, 1, 4, 1, 0, 3, 1, 1, 3, 4, 3, 1, 0, 1, 0, 1, 1, 3, 0, 0, 0,
       2, 2, 3, 3, 2, 1, 0, 1, 1, 1, 0, 1, 2, 2, 1, 1, 0, 0, 1, 0, 1, 1,
       3, 4, 1, 2, 0, 0, 1, 2, 1, 1, 0, 1, 2, 2, 0, 3, 1, 3, 3, 0, 1, 1,
       0, 3, 1, 2, 2, 2, 1, 0, 0, 1, 1, 0, 3])
```

In [31]:

```
k_means.inertia_
```

Out[31]:

```
14.984580851917167
```

In [32]:

```
k_means =  KMeans(n_clusters=4)
k_means.fit(data.drop('country',axis=1))
k_means.labels_
```

Out[32]:

```
array([0, 1, 1, 0, 1, 1, 1, 3, 3, 1, 1, 1, 1, 1, 1, 3, 1, 0, 1, 1, 1, 1,
       1, 3, 1, 0, 0, 1, 0, 3, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 3, 1,
       3, 1, 1, 1, 1, 0, 0, 1, 1, 3, 3, 0, 0, 1, 3, 0, 3, 1, 1, 0, 0, 1,
       0, 1, 3, 1, 1, 1, 0, 3, 3, 3, 1, 3, 1, 1, 0, 0, 3, 1, 0, 1, 1, 0,
       0, 1, 1, 2, 1, 0, 0, 1, 1, 0, 2, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
       3, 3, 0, 0, 3, 1, 0, 1, 1, 1, 1, 1, 3, 3, 1, 1, 0, 1, 1, 0, 1, 1,
       0, 2, 1, 3, 0, 1, 3, 3, 1, 1, 0, 1, 3, 3, 1, 0, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 3, 3, 3, 1, 1, 1, 1, 1, 0, 0])
```

In [33]:

```
k_means.inertia_
```

Out[33]:

```
16.781002591696133
```

In [35]:

```
K = range(1,10)
ssd = []
for k in K:
    k_means =  KMeans(n_clusters=k)
    k_means.fit(data.drop('country',axis=1))
    ssd.append(k_means.inertia_)
```

In [36]:

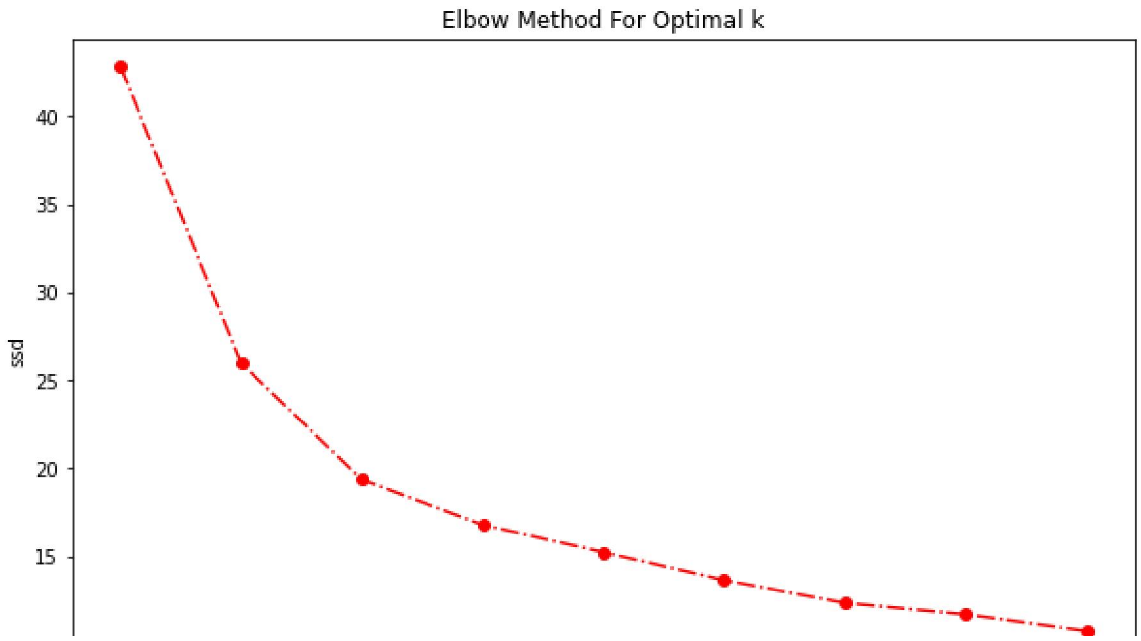
```
ssd
```

Out[36]:

```
[42.79871877568751,
 25.94736093352987,
 19.345118591450642,
 16.781002591696133,
 15.220965868061919,
 13.642962629065122,
 12.33833644974413,
 11.695149667266776,
 10.754062835758939]
```

In [37]:

```
plt.figure(figsize=(10,6))
plt.plot(K, ssd, 'ro-.')
plt.xlabel('k')
plt.ylabel('ssd')
plt.title('Elbow Method For Optimal k')
plt.show()
```



In [38]:

```
k_means =  KMeans(n_clusters=3)
k_means.fit(data.drop('country',axis=1))
pred = k_means.labels_
```

In [39]:

```
exp.scatter(data_frame=data,x='child_mort',y='income',color=pred)
```