Sugarcane Production Analysis....

Importing libraries and reading the data

In [1]:
```
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
```

In [2]:
```
df=pd.read_csv("List of Countries by Sugarcane Production.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	Unnamed: 0	Country	Continent	Production (Tons)	Production per Person (Kg)	Acreage (Hectare)	Yield (Kg / Hectare)
0	0	Brazil	South America	768.678.382	3.668,531	10.226.205	75.167,5
1	1	India	Asia	348.448.000	260721	4.950.000	70.393,5
2	2	China	Asia	123.059.739	88287	1.675.215	73.459,1
3	3	Thailand	Asia	87.468.496	1.264,303	1.336.575	65.442,2
4	4	Pakistan	Asia	65.450.704	324219	1.130.820	57.879

Data preprocessing

In [4]:
```
df.shape
```

Out[4]:
```
(103, 7)
```

In [5]:
```
df["Production (Tons)"] = df["Production (Tons)"].str.replace(".","")
df["Production per Person (Kg)"] = df["Production per Person (Kg)"].str.replace(".","").
df["Acreage (Hectare)"] = df["Acreage (Hectare)"].str.replace(".","")
df["Yield (Kg / Hectare)"]= df["Yield (Kg / Hectare)"].str.replace(".","").str.replace("
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_6232\1067177181.py:1: FutureWarning: The defaul
t value of regex will change from True to False in a future version. In addition, single
character regular expressions will *not* be treated as literal strings when regex=True.
  df["Production (Tons)"] = df["Production (Tons)"].str.replace(".","")
C:\Users\HP\AppData\Local\Temp\ipykernel_6232\1067177181.py:2: FutureWarning: The defaul
t value of regex will change from True to False in a future version. In addition, single
character regular expressions will *not* be treated as literal strings when regex=True.
  df["Production per Person (Kg)"] = df["Production per Person (Kg)"].str.replace
(".","").str.replace(",",".")
C:\Users\HP\AppData\Local\Temp\ipykernel_6232\1067177181.py:3: FutureWarning: The defaul
t value of regex will change from True to False in a future version. In addition, single
character regular expressions will *not* be treated as literal strings when regex=True.
  df["Acreage (Hectare)"] = df["Acreage (Hectare)"].str.replace(".","")
C:\Users\HP\AppData\Local\Temp\ipykernel_6232\1067177181.py:4: FutureWarning: The defaul
t value of regex will change from True to False in a future version. In addition, single
character regular expressions will *not* be treated as literal strings when regex=True.
  df["Yield (Kg / Hectare)"]= df["Yield (Kg / Hectare)"].str.replace(".","").str.replace
(",",".")
```

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

Out[6]:

	Unnamed: 0	Country	Continent	Production (Tons)	Production per Person (Kg)	Acreage (Hectare)	Yield (Kg / Hectare)
0	0	Brazil	South America	768678382	3668.531	10226205	75167.5
1	1	India	Asia	348448000	260721	4950000	70393.5
2	2	China	Asia	123059739	88287	1675215	73459.1
3	3	Thailand	Asia	87468496	1264.303	1336575	65442.2
4	4	Pakistan	Asia	65450704	324219	1130820	57879

In [7]:
```
df.tail()
```

Out[7]:

	Unnamed: 0	Country	Continent	Production (Tons)	Production per Person (Kg)	Acreage (Hectare)	Yield (Kg / Hectare)
98	98	Lebanon	Asia	97	16	3	28386.4
99	99	Djibouti	Africa	53	51	NaN	NaN
100	100	Singapore	Asia	50	9	2	25
101	101	Samoa	Oceania	12	6	1	11949.8
102	102	Syria	Asia	1	0	0	83034.2

Checking the Null value and removing it

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
Unnamed: 0                      0
Country                         0
Continent                       0
Production (Tons)               0
Production per Person (Kg)      0
Acreage (Hectare)               1
Yield (Kg / Hectare)            1
dtype: int64
```

In [9]:
```
df[df["Acreage (Hectare)"].isnull()]
```

Out[9]:

	Unnamed: 0	Country	Continent	Production (Tons)	Production per Person (Kg)	Acreage (Hectare)	Yield (Kg / Hectare)
99	99	Djibouti	Africa	53	51	NaN	NaN

In [10]:
```
df=df.dropna().reset_index().drop("index",axis=1)
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
Unnamed: 0                      0
Country                         0
Continent                       0
Production (Tons)               0
Production per Person (Kg)      0
Acreage (Hectare)               0
Yield (Kg / Hectare)            0
dtype: int64
```

In [12]:
```
df=df.drop("Unnamed: 0",axis=1)
```

In [13]:
```
df.head()
```

Out[13]:

	Country	Continent	Production (Tons)	Production per Person (Kg)	Acreage (Hectare)	Yield (Kg / Hectare)
0	Brazil	South America	768678382	3668.531	10226205	75167.5
1	India	Asia	348448000	260721	4950000	70393.5
2	China	Asia	123059739	88287	1675215	73459.1
3	Thailand	Asia	87468496	1264.303	1336575	65442.2
4	Pakistan	Asia	65450704	324219	1130820	57879

In [14]:
```
df.rename(columns= {"Production (Tons)": "Production(Tons)"}, inplace = True)
df.rename(columns= {"Production per Person (Kg)": "Production_per_person(Kg)"}, inplace
df.rename(columns= {"Acreage (Hectare)": "Acreage(Hectare)"}, inplace = True)
df.rename(columns= {"Yield (Kg / Hectare)": "Yield(Kg/Hectare)"}, inplace = True)
```

In [ ]:
```
```

In [15]:
```
df.nunique()
```

Out[15]:
```
Country                        102
Continent                        6
Production(Tons)               102
Production_per_person(Kg)      101
Acreage(Hectare)               101
Yield(Kg/Hectare)              102
dtype: int64
```

In [16]:
```
df.dtypes
```

Out[16]:
```
Country                        object
Continent                      object
Production(Tons)               object
Production_per_person(Kg)      object
Acreage(Hectare)               object
Yield(Kg/Hectare)              object
dtype: object
```

In [17]:
```
df["Production(Tons)"]=df["Production(Tons)"].astype(float)
df["Production_per_person(Kg)"]=df["Production_per_person(Kg)"].astype(float)
df["Acreage(Hectare)"]=df["Acreage(Hectare)"].astype(float)
df["Yield(Kg/Hectare)"]=df["Yield(Kg/Hectare)"].astype(float)
```

In [18]:
```
df.dtypes
```

Out[18]:
```
Country                         object
Continent                       object
Production(Tons)               float64
Production_per_person(Kg)      float64
Acreage(Hectare)               float64
Yield(Kg/Hectare)              float64
dtype: object
```

In [19]:
```
df.duplicated().sum()
```

Out[19]:
```
0
```

Univariate Analysis

In [20]:
```
df.head()
```

Out[20]:

	Country	Continent	Production(Tons)	Production_per_person(Kg)	Acreage(Hectare)	Yield(Kg/Hectare)
0	Brazil	South America	768678382.0	3668.531	10226205.0	75167.5
1	India	Asia	348448000.0	260721.000	4950000.0	70393.5
2	China	Asia	123059739.0	88287.000	1675215.0	73459.1
3	Thailand	Asia	87468496.0	1264.303	1336575.0	65442.2
4	Pakistan	Asia	65450704.0	324219.000	1130820.0	57879.0

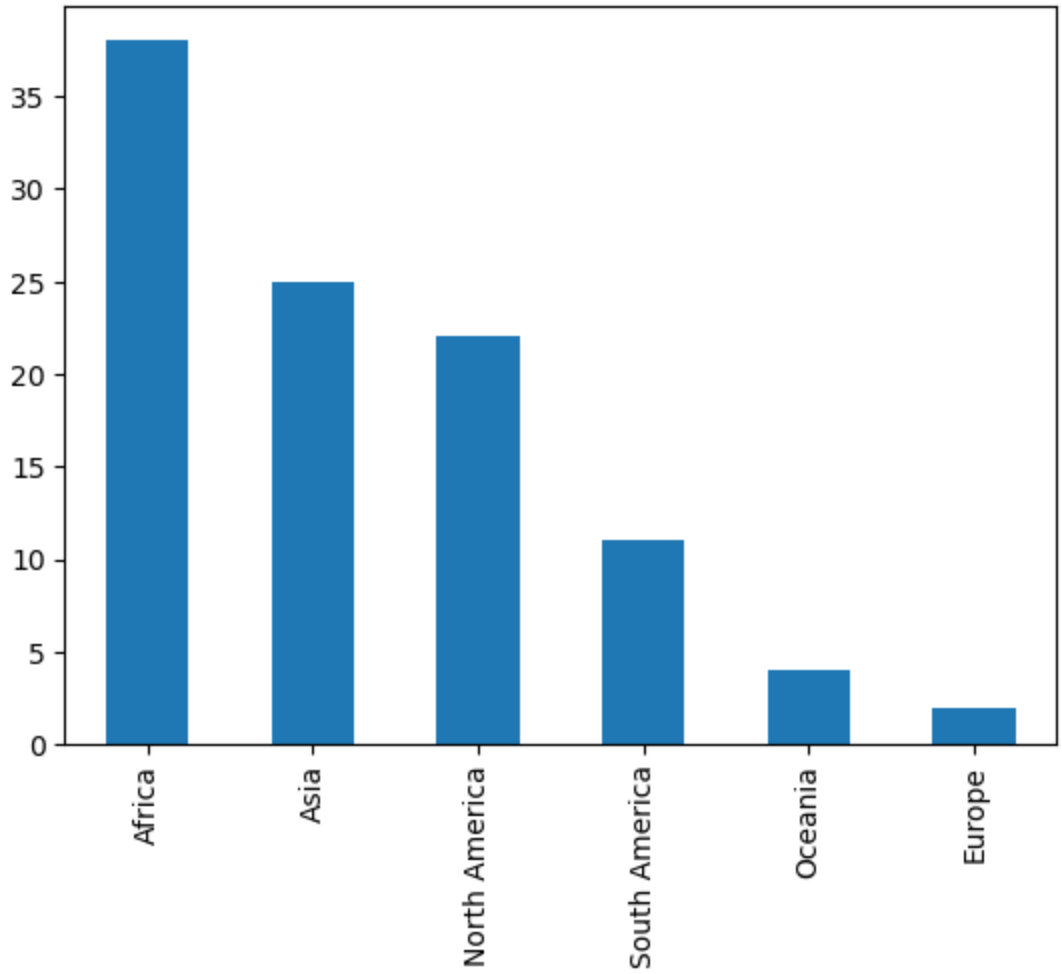
How many countries from each continent produces the sugarcane?

In [21]:
```
df["Continent"].value_counts()
```

Out[21]:
```
Africa           38
Asia             25
North America    22
South America    11
Oceania           4
Europe            2
Name: Continent, dtype: int64
```

In [22]:
```
df["Continent"].value_counts().plot(kind="bar")
```

Out[22]:
```
<Axes: >
```



In [23]:
```
df.describe()
```

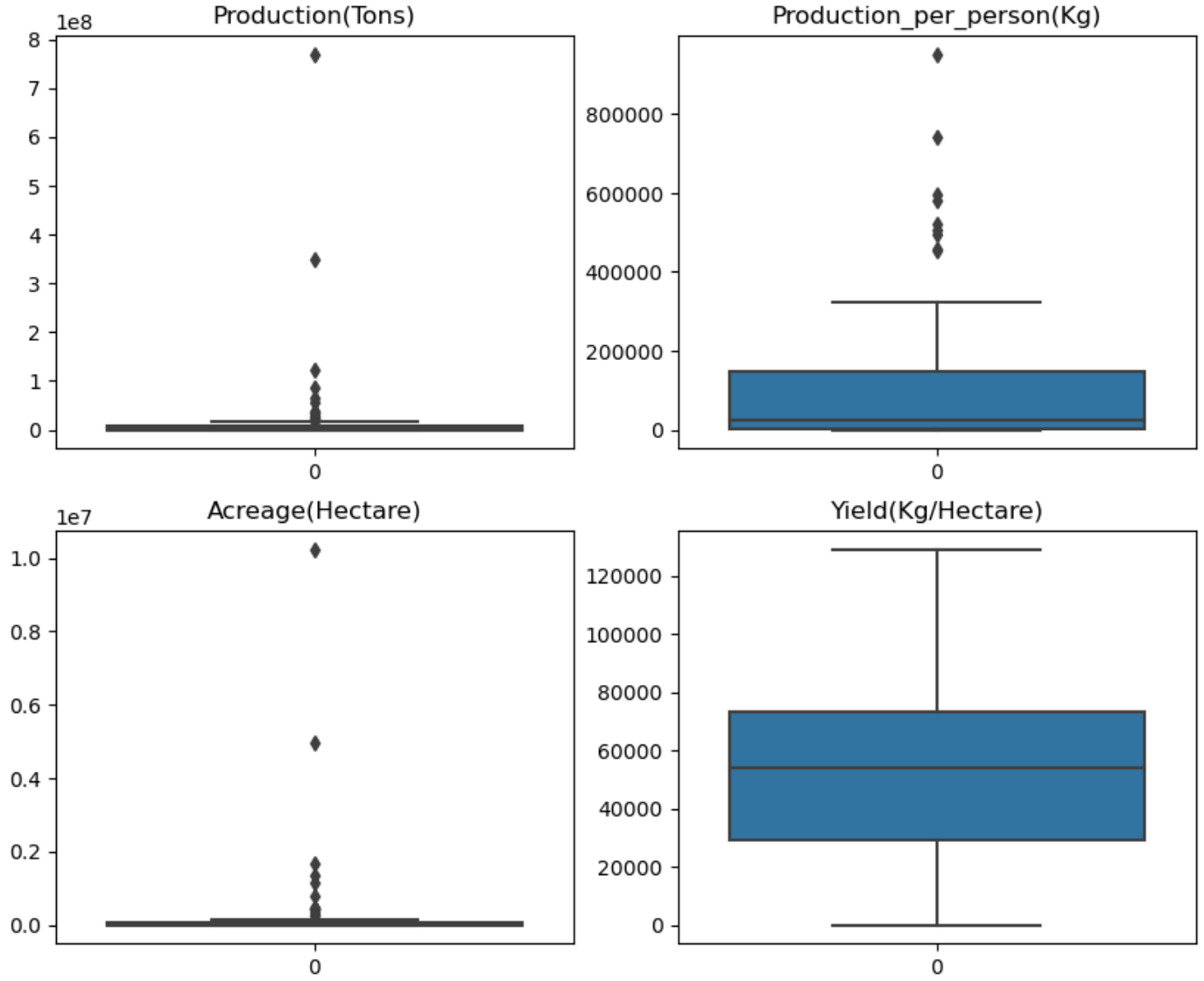
Out[23]:

	Production(Tons)	Production_per_person(Kg)	Acreage(Hectare)	Yield(Kg/Hectare)
count	1.020000e+02	102.000000	1.020000e+02	102.000000
mean	1.850372e+07	112952.435755	2.498981e+05	52628.078431
std	8.419149e+07	176651.341929	1.137003e+06	30504.676683
min	1.000000e+00	0.000000	0.000000e+00	10.000000
25%	6.251875e+04	3671.910000	1.104000e+03	29072.025000
50%	1.440044e+06	25572.500000	1.655800e+04	54108.950000
75%	6.426824e+06	146384.750000	8.047400e+04	73282.700000
max	7.686784e+08	951087.000000	1.022620e+07	129049.300000

Checking for Outliers

In [24]:
```
plt.figure(figsize= (10,8))
plt.subplot(2,2,1)
sns.boxplot(df["Production(Tons)"])
plt.title("Production(Tons)")
plt.subplot(2,2,2)
sns.boxplot(df["Production_per_person(Kg)"])
plt.title("Production_per_person(Kg)")
plt.subplot(2,2,3)
sns.boxplot(df["Acreage(Hectare)"])
plt.title("Acreage(Hectare)")
```

```
plt.subplot(2,2,4)
sns.boxplot(df["Yield(Kg/Hectare)"])
plt.title("Yield(Kg/Hectare)")
plt.show()
```



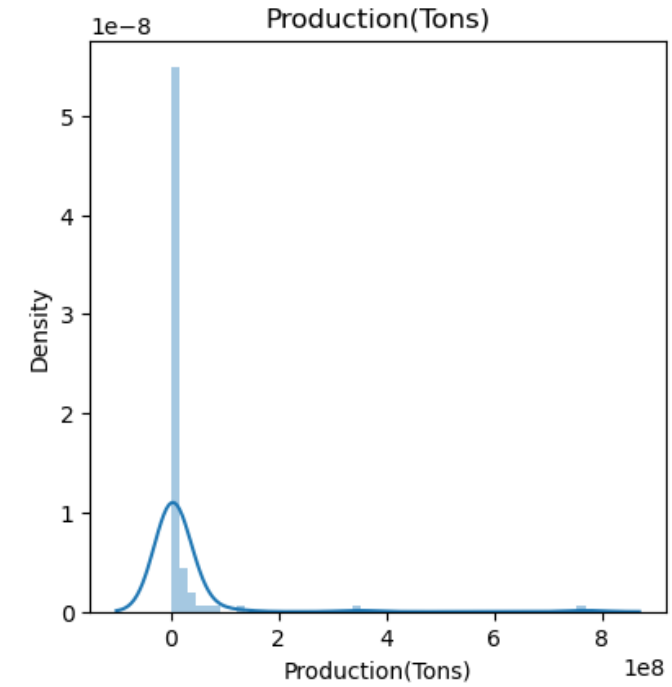
we have outliers in the data but outliers are required here as it shows the countries which has maximum production.

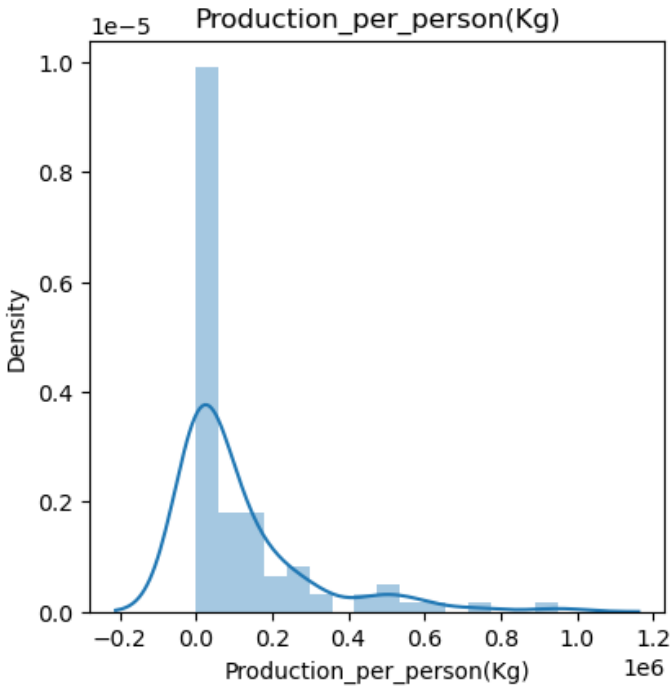
Distribution of Data in the Columns

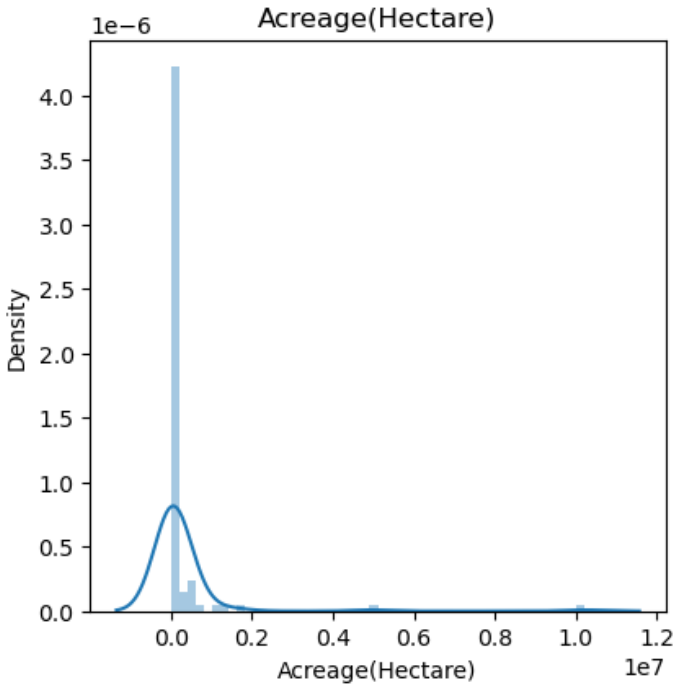
In [25]:
```
plt.figure(figsize=(10,10))
plt.subplot(2,2,1)
sns.distplot(df["Production(Tons)"])
plt.title("Production(Tons)")
plt.subplot(2,2,2)
sns.distplot(df["Production_per_person(Kg)"])
plt.title("Production_per_person(Kg)")
plt.subplot(2,2,3)
sns.distplot(df["Acreage(Hectare)"])
plt.title("Acreage(Hectare)")
plt.subplot(2,2,4)
sns.distplot(df["Yield(Kg/Hectare)"])
plt.title("Yield(Kg/Hectare)")
plt.show()
```

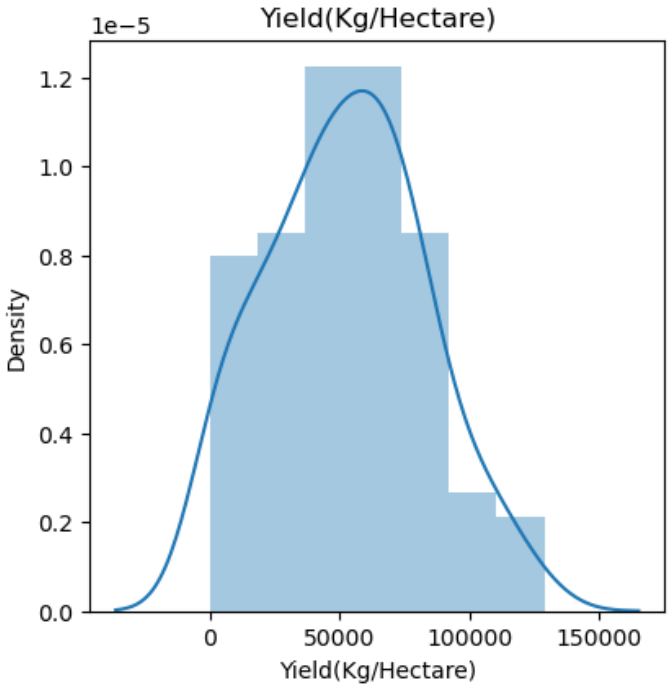
```
C:\Users\HP\AppData\Local\Temp\ipykernel_6232\2750422772.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
```

```
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Production(Tons)"])
C:\Users\HP\AppData\Local\Temp\ipykernel_6232\2750422772.py:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Production_per_person(Kg)"])
C:\Users\HP\AppData\Local\Temp\ipykernel_6232\2750422772.py:9: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Acreage(Hectare)"])
C:\Users\HP\AppData\Local\Temp\ipykernel_6232\2750422772.py:12: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Yield(Kg/Hectare)"])
```









Bivariate Analysis

Which country produces most amount of sugarcane in tons

In [26]:
```
df_new=df[ ["Country", "Production(Tons)"] ]
print(df_new)
```

```
       Country  Production(Tons)
0       Brazil       768678382.0
1        India       348448000.0
2        China       123059739.0
3     Thailand        87468496.0
4     Pakistan        65450704.0
..         ...               ...
97       Spain             394.0
98     Lebanon              97.0
99   Singapore              50.0
```

```
100       Samoa                12.0
101       Syria                 1.0

[102 rows x 2 columns]
```

In [27]:
```
df_new["Production(Tons)_percent"] = df_new["Production(Tons)"]*100/df_new["Production(T
df_new
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_6232\114540126.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  df_new["Production(Tons)_percent"] = df_new["Production(Tons)"]*100/df_new["Production
(Tons)"].sum()
```

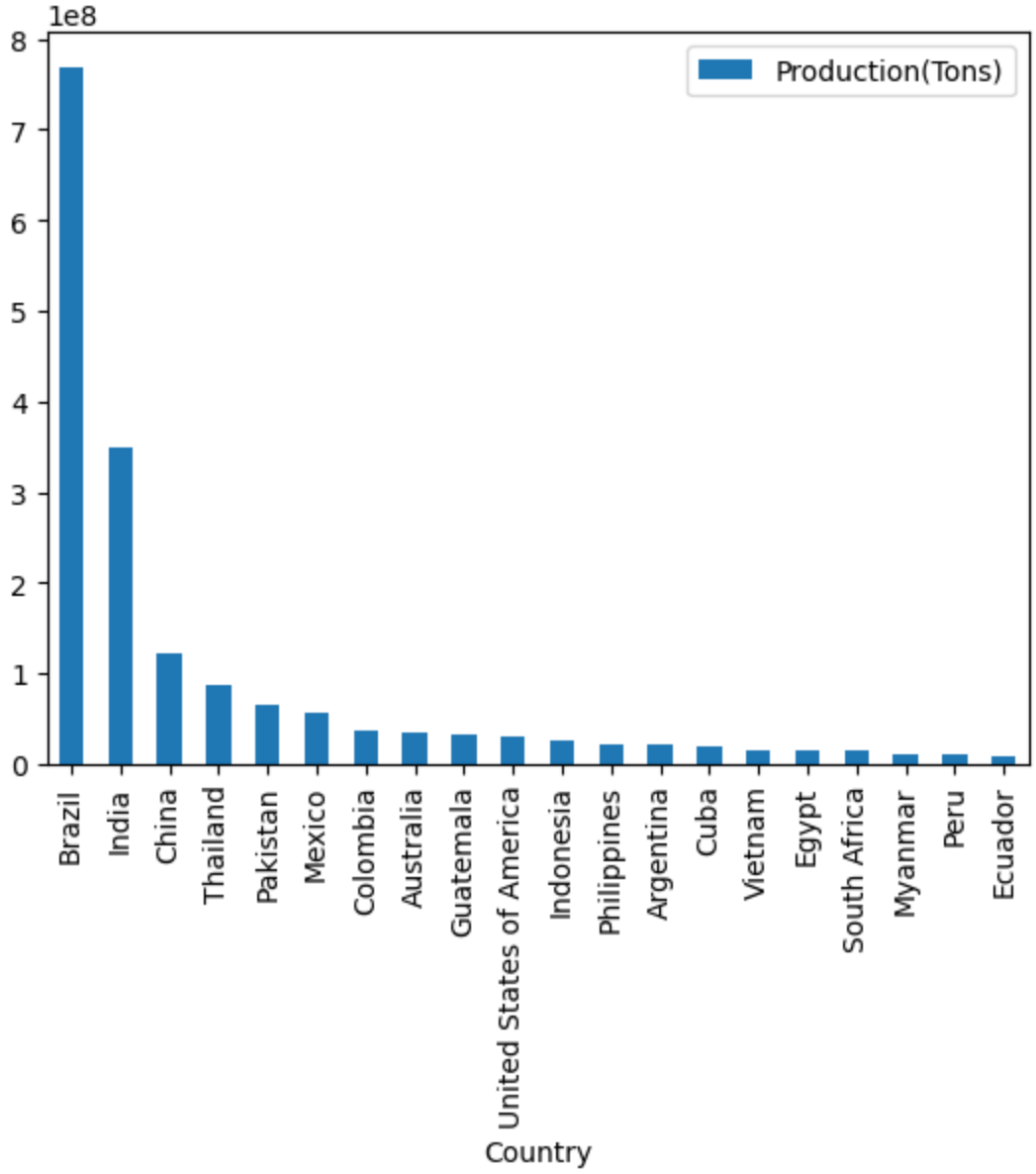
Out[27]:

	Country	Production(Tons)	Production(Tons)_percent
0	Brazil	768678382.0	4.072729e+01
1	India	348448000.0	1.846200e+01
2	China	123059739.0	6.520138e+00
3	Thailand	87468496.0	4.634389e+00
4	Pakistan	65450704.0	3.467809e+00
...	...	...	...
97	Spain	394.0	2.087551e-05
98	Lebanon	97.0	5.139401e-06
99	Singapore	50.0	2.649176e-06
100	Samoa	12.0	6.358022e-07
101	Syria	1.0	5.298352e-08

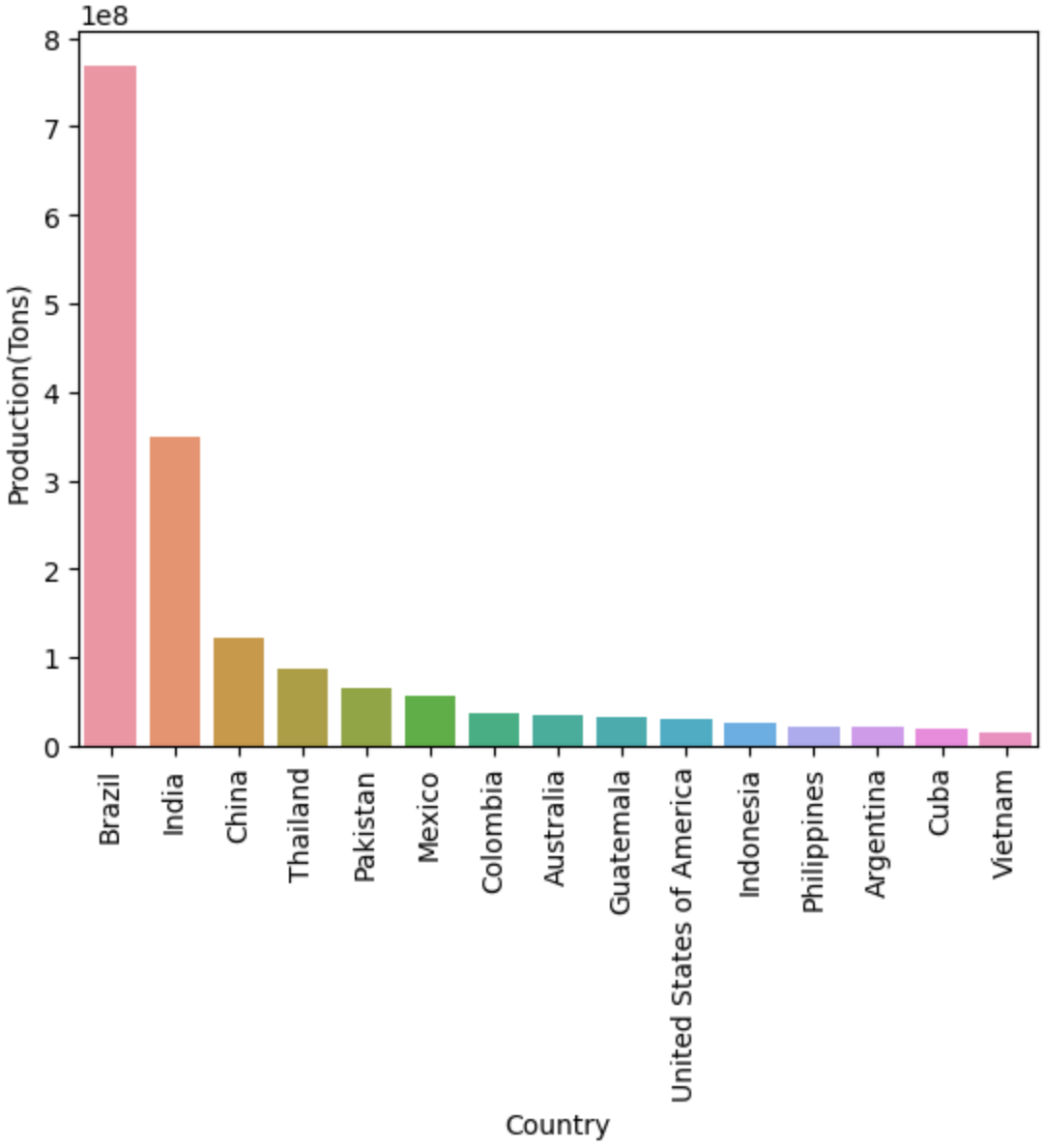
102 rows × 3 columns

In [28]:
```
df[["Country", "Production(Tons)"]].set_index("Country").sort_values("Production(Tons)",
```

Out[28]:
```
<Axes: xlabel='Country'>
```



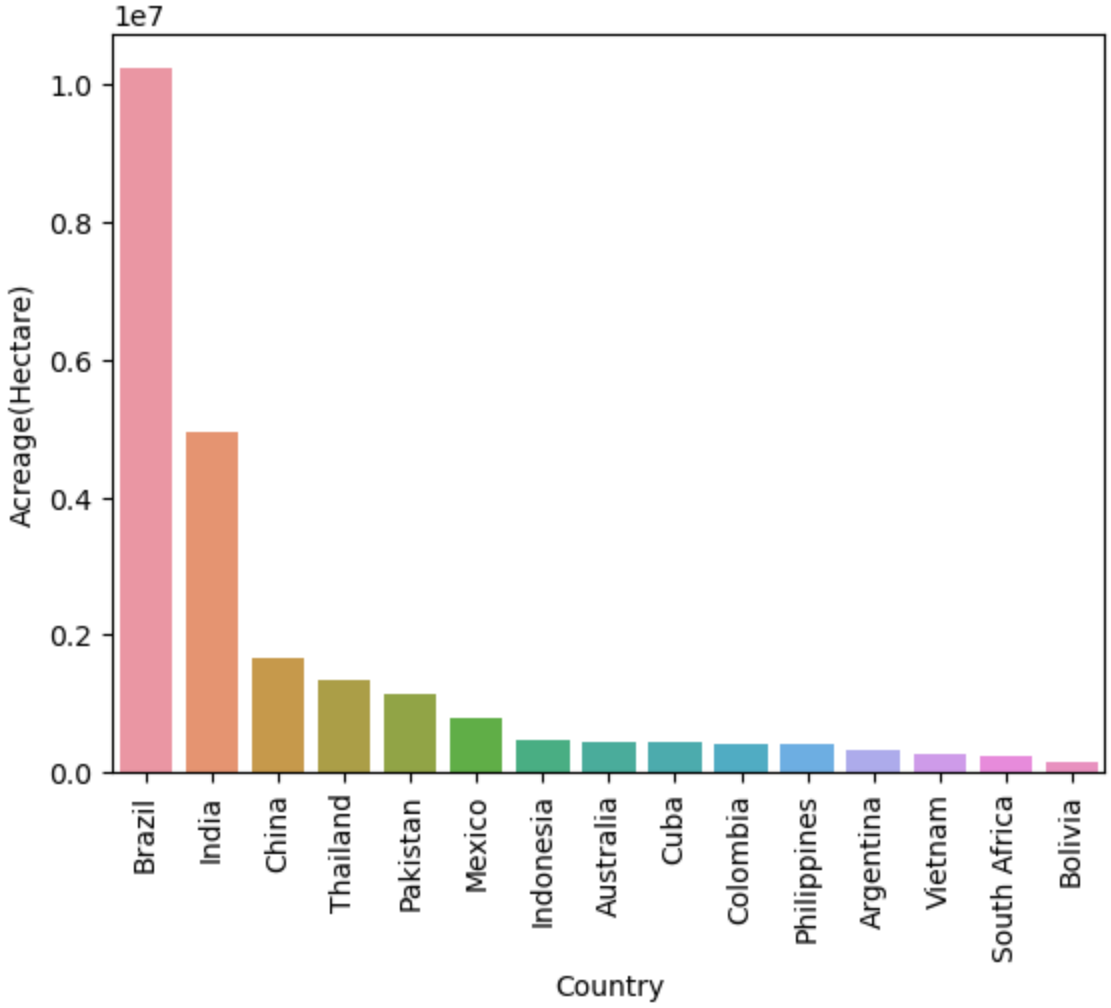
In [29]:
```
ax = sns.barplot(data = df.head(15),  x= "Country", y = "Production(Tons)")
ax.set_xticklabels(ax.get_xticklabels(),rotation =90)
plt.show()
```



Among all the countries, Brazil,India and China are the leading producers of sugarcane.

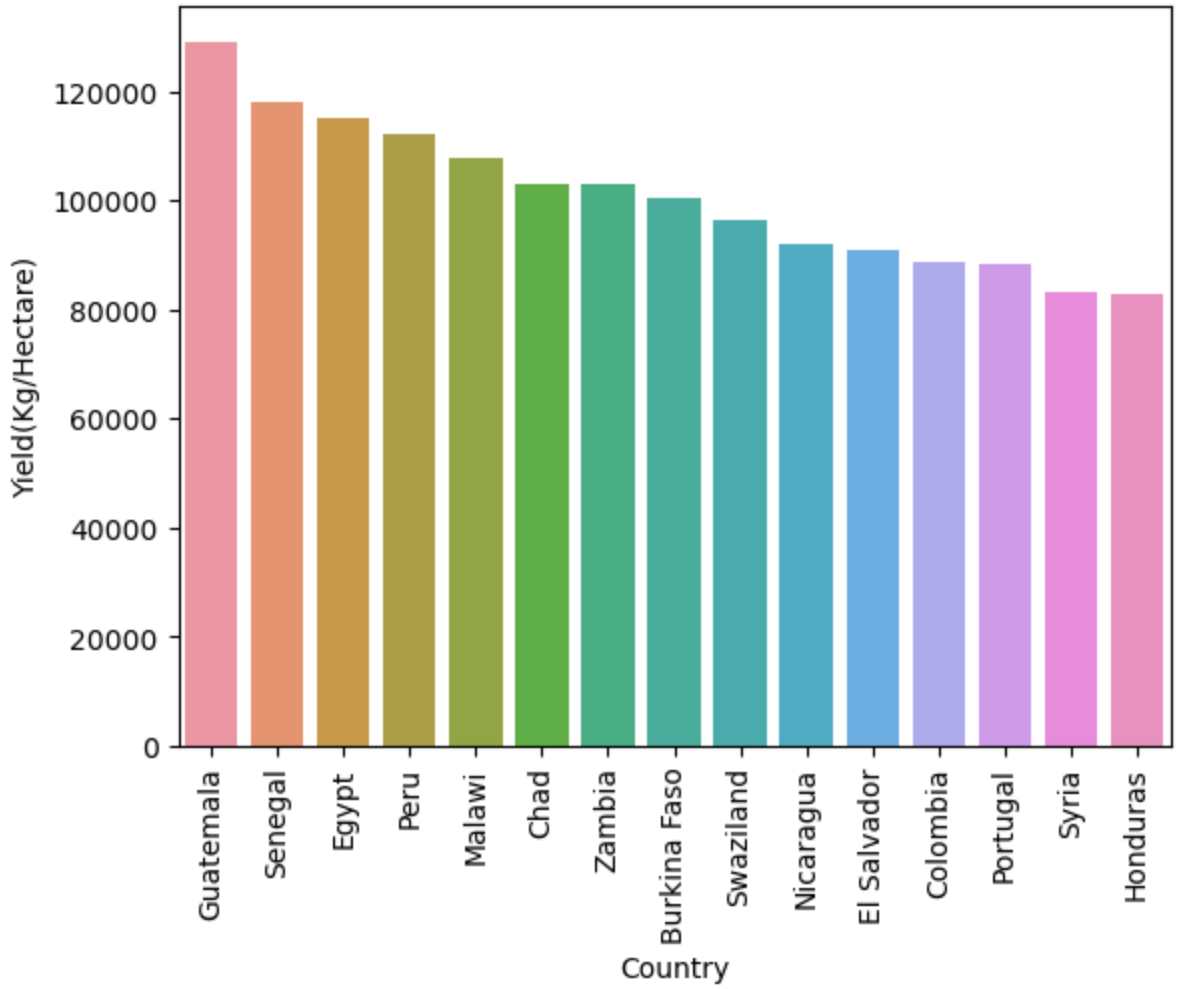
Which country has the most Area(Hectares)?

In [30]:
```
df_hectare=df.sort_values("Acreage(Hectare)",ascending=False).head(15)
ax = sns.barplot(data = df_hectare, x= "Country", y ="Acreage(Hectare)" )
ax.set_xticklabels(ax.get_xticklabels(),rotation =90)
plt.show()
```



Which country has the highest yield(Kg/Hectare)

In [31]:
```
df_yield=df.sort_values("Yield(Kg/Hectare)",ascending=False).head(15)
ax = sns.barplot(data = df_yield,  x= "Country", y ="Yield(Kg/Hectare)" )
ax.set_xticklabels(ax.get_xticklabels(),rotation =90)
plt.show()
```



Guatamala has the highest yield(kg/hectare

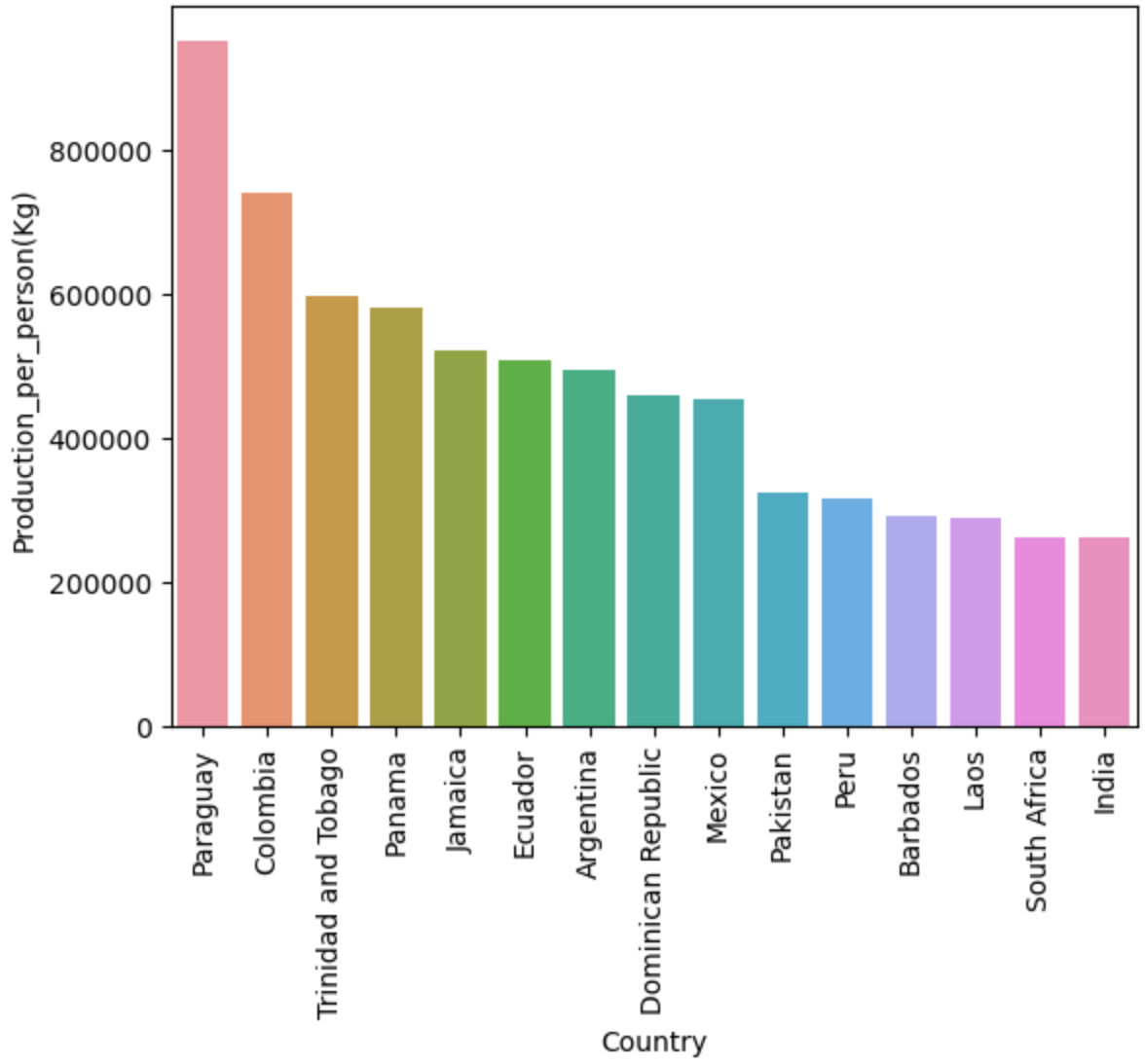
Which country has highest per person production?

In [32]:
```
df.head()
```

Out[32]:

	Country	Continent	Production(Tons)	Production_per_person(Kg)	Acreage(Hectare)	Yield(Kg/Hectare)
0	Brazil	South America	768678382.0	3668.531	10226205.0	75167.5
1	India	Asia	348448000.0	260721.000	4950000.0	70393.5
2	China	Asia	123059739.0	88287.000	1675215.0	73459.1
3	Thailand	Asia	87468496.0	1264.303	1336575.0	65442.2
4	Pakistan	Asia	65450704.0	324219.000	1130820.0	57879.0

In [33]:
```
df_person=df.sort_values("Production_per_person(Kg)", ascending=False).head(15)
ax=sns.barplot(data = df_person, x ="Country", y= "Production_per_person(Kg)")
ax.set_xticklabels(ax.get_xticklabels(),rotation=90)
plt.show()
```



Paraguay has the highest per person production

In [34]:
```
df.corr()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_6232\1134722465.py:1: FutureWarning: The defaul
t value of numeric_only in DataFrame.corr is deprecated. In a future version, it will de
fault to False. Select only valid columns or specify the value of numeric_only to silenc
e this warning.
  df.corr()
```

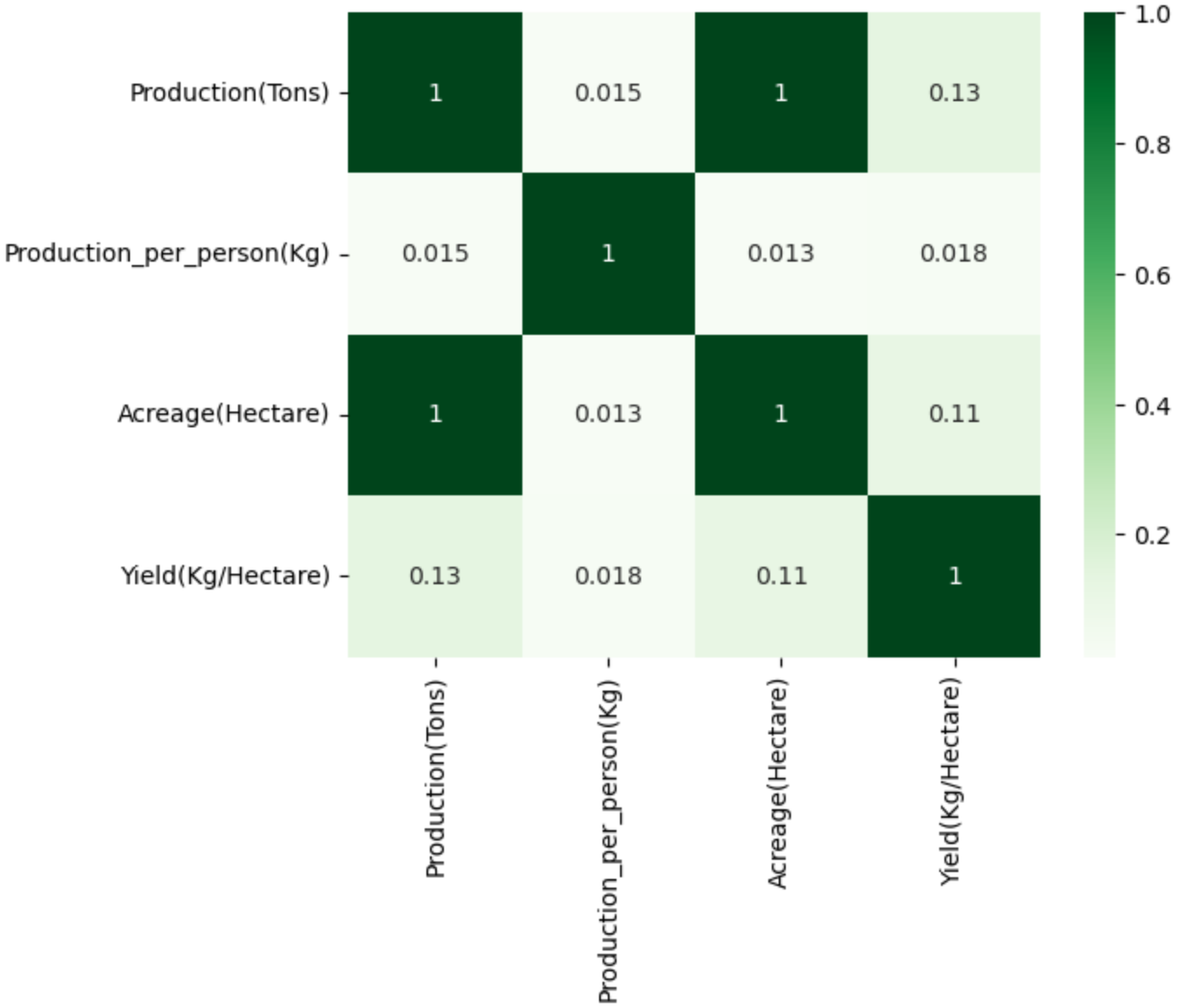
Out[34]:

	Production(Tons)	Production_per_person(Kg)	Acreage(Hectare)	Yield(Kg/Hectare)
Production(Tons)	1.000000	0.015000	0.997550	0.132812
Production_per_person(Kg)	0.015000	1.000000	0.012557	0.017999
Acreage(Hectare)	0.997550	0.012557	1.000000	0.113433
Yield(Kg/Hectare)	0.132812	0.017999	0.113433	1.000000

In [35]:
```
sns.heatmap(df.corr(), annot = True, cmap="Greens")
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_6232\642961471.py:1: FutureWarning: The default
value of numeric_only in DataFrame.corr is deprecated. In a future version, it will defa
ult to False. Select only valid columns or specify the value of numeric_only to silence
this warning.
  sns.heatmap(df.corr(), annot = True, cmap="Greens")
```

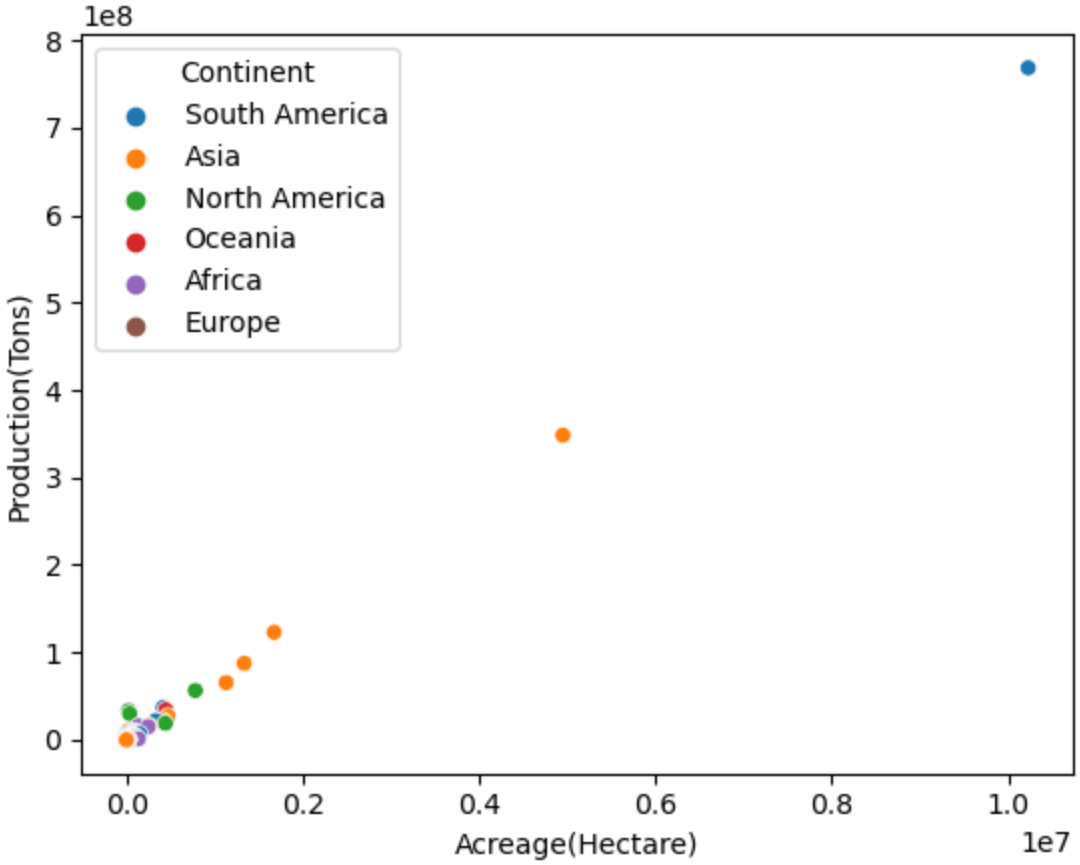
Out[35]:
```
<Axes: >
```



Do countries with highest land produce more sugarcane?

In [36]:
```
sns.scatterplot(data = df, x = "Acreage(Hectare)", y = "Production(Tons)", hue = "Contin
```

Out[36]:
```
<Axes: xlabel='Acreage(Hectare)', ylabel='Production(Tons)'>
```



Analysis for Continent

In [37]:
```
df.head()
```

Out[37]:

	Country	Continent	Production(Tons)	Production_per_person(Kg)	Acreage(Hectare)	Yield(Kg/Hectare)
0	Brazil	South America	768678382.0	3668.531	10226205.0	75167.5
1	India	Asia	348448000.0	260721.000	4950000.0	70393.5
2	China	Asia	123059739.0	88287.000	1675215.0	73459.1
3	Thailand	Asia	87468496.0	1264.303	1336575.0	65442.2
4	Pakistan	Asia	65450704.0	324219.000	1130820.0	57879.0

In [38]:
```
df_continent=df.groupby("Continent").sum()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_6232\1663322524.py:1: FutureWarning: The defaul
t value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, nume
ric_only will default to False. Either specify numeric_only or select only columns which
should be valid for the function.
  df_continent=df.groupby("Continent").sum()
```

In [39]:
```
df_continent["number_of_countries"] = df.groupby("Continent").count()["Country"]
df_continent
```

Out[39]:

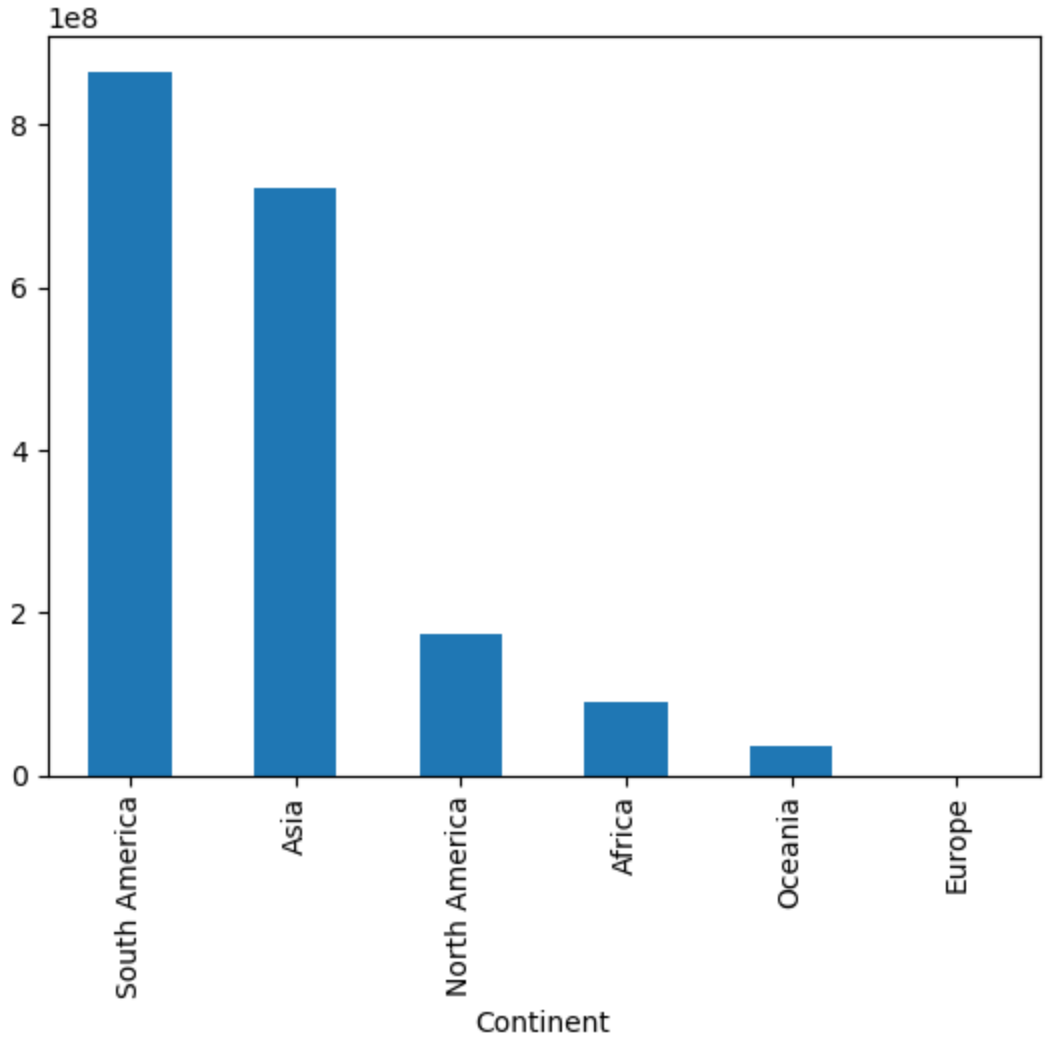
Continent	Production(Tons)	Production_per_person(Kg)	Acreage(Hectare)	Yield(Kg/Hectare)	number_of_countries
Africa	89681472.0	2332636.293	1439089.0	2142107.5	38
Asia	721930425.0	1857769.303	10608319.0	1171871.4	25

Europe	5823.0	536.000	71.0	131870.9	2
North America	173995947.0	3796081.508	1581983.0	1082602.4	22
Oceania	36177574.0	28593.605	490909.0	162419.1	4
South America	865588126.0	3505531.738	11369236.0	677192.7	11

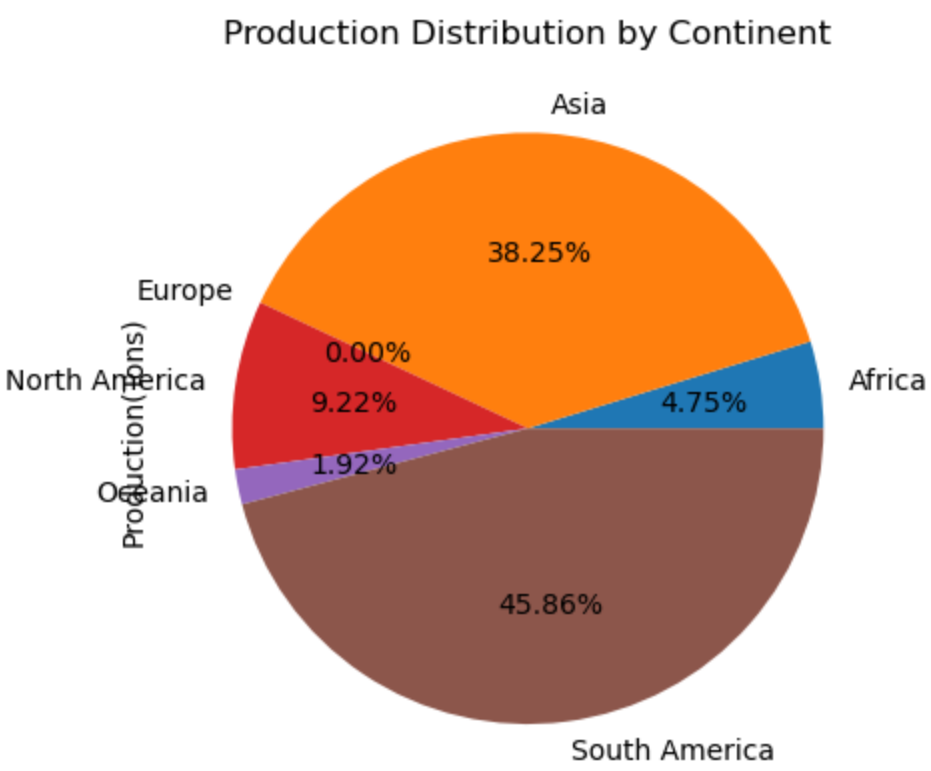
Which continent produce the maximum sugarcane?

In [40]:
```
df_continent["Production(Tons)"].sort_values(ascending = False).plot(kind = "bar")
```

Out[40]:
```
<Axes: xlabel='Continent'>
```



In [41]:
```
df_continent["Production(Tons)"].plot(kind = "pie", autopct = "%.2f%%")
plt.title('Production Distribution by Continent')
plt.show()
```



In [42]:
```
df_continent.corr()
```

Out[42]:

	Production(Tons)	Production_per_person(Kg)	Acreage(Hectare)	Yield(Kg/Hectare)	num
Production(Tons)	1.000000	0.522211	0.994897	0.091201	
Production_per_person(Kg)	0.522211	1.000000	0.463215	0.542961	
Acreage(Hectare)	0.994897	0.463215	1.000000	0.111166	
Yield(Kg/Hectare)	0.091201	0.542961	0.111166	1.000000	
number_of_countries	0.109244	0.540086	0.132817	0.989712	

In [ ]:
```
```