PART 2

This following part of this exercise can be done and delivered untill Sunday, 23/06/2024 up to 23:59 (11:59 PM).

Exercise: Analyzing Students.csv or whatever data base you may want to

(Data.gov, EU Open Data Portal, Kaggle Datasets) work through -

Performance Data

***You are provided with a dataset containing information about student performance in exams. Your task is to perform data analysis and visualization using Python libraries. Here are the steps to follow:

1) Load the Data:

1.1) Use pandas to read the dataset from a CSV file (students.csv). Data Exploration:

1.2) Display the first few rows of the dataset to understand its structure.

1.3) Check for missing values and handle them appropriately if necessary.

2) Data Analysis:

2.1) Calculate basic statistics of the dataset (mean, median, min, max, etc.).

Explore the distribution of scores using histograms and box plots.

3) Data Visualization:

3.1) Use matplotlib and seaborn to create visualizations such as:

a) Histograms of scores in different subjects.

b) Box plots to compare scores across different categories (e.g., gender, parental ##### level of education).

c) Scatter plots to explore relationships between variables (e.g., math vs. reading ##### scores).

4) Advanced Analysis:

4.1) Calculate correlations between different variables (e.g., scores in different subjects).

4.2) Create a heatmap using seaborn to visualize correlations.

Conclusion:

Summarize - State your findings from the analysis.

Provide insights or conclusions based on the visualizations and ### analyses performed.

Send these two exercises to

fischer.stefan@academico.domhelder.edu.br

Subeject: Project Capstone

Save versions in .py or ipynb and .pdf

Do not forget to write down your name!!

Enzo Rocha Leite Diniz Ribas - D24642

Data Base Selected: https://www.kaggle.com/datasets/crisparada/brazilian-cities?resource=download

In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import scipy
import matplotlib.pyplot as plt
%matplotlib inline
```

1) Load the Data:

1.1) Use pandas to read the dataset from a CSV file (students.csv). Data Exploration:

In [ ]:
```
BR_Cities_df = pd.read_csv('BRAZIL_CITIES_REV2022.CSV')
```

In [ ]:
```
BR_Cities_df
```

Out[ ]:

	CITY	STATE	CAPITAL	IBGE_RES_POP	IBGE_RES_POP_BRAS	IBGE_RES_POP_
0	Abadia De Goiás	GO	0	6876	6876	
1	Abadia Dos Dourados	MG	0	6704	6704	
2	Abadiânia	GO	0	15757	15609	
3	Abaetetuba	PA	0	141100	141040	
4	Abaeté	MG	0	22690	22690	
...	...	...	...	...	...	
5573	Áurea	RS	0	3665	3665	
5574	Ângulo	PR	0	2859	2844	
5575	Érico Cardoso	BA	0	10859	10859	
5576	Óbidos	PA	0	49333	49324	
5577	Óleo	SP	0	2673	2673	

5578 rows × 81 columns

1.2) Display the first few rows of the dataset to understand its structure.

In [ ]:
```
BR_Cities_df.head(10)
```

Out[ ]:

	CITY	STATE	CAPITAL	IBGE_RES_POP	IBGE_RES_POP_BRAS	IBGE_RES_POP_EST
0	Abadia De Goiás	GO	0	6876	6876	
1	Abadia Dos Dourados	MG	0	6704	6704	
2	Abadiânia	GO	0	15757	15609	14
3	Abaetetuba	PA	0	141100	141040	6
4	Abaeté	MG	0	22690	22690	
5	Abaiara	CE	0	10496	10496	
6	Abaré	BA	0	17064	17064	
7	Abatiá	PR	0	7764	7764	
8	Abaíra	BA	0	8316	8316	
9	Abdon Batista	SC	0	2653	2653	

10 rows × 81 columns

In [ ]:
```
BR_Cities_df.columns
```

Out[ ]:
```
Index(['CITY', 'STATE', 'CAPITAL', 'IBGE_RES_POP', 'IBGE_RES_POP_BRAS',
       'IBGE_RES_POP_ESTR', 'IBGE_DU', 'IBGE_DU_URBAN', 'IBGE_DU_RURAL',
       'IBGE_POP', 'IBGE_1', 'IBGE_1-4', 'IBGE_5-9', 'IBGE_10-14',
       'IBGE_15-59', 'IBGE_60+', 'IBGE_PLANTED_AREA', 'IBGE_CROP_PRODUCTION_$',
       'IDHM Ranking 2010', 'IDHM', 'IDHM_Renda', 'IDHM_Longevidade',
       'IDHM_Educacao', 'LONG', 'LAT', 'ALT', 'PAY_TV', 'FIXED_PHONES', 'AREA',
       'REGIAO_TUR', 'CATEGORIA_TUR', 'ESTIMATED_POP', 'RURAL_URBAN',
       'GVA_AGROPEC', 'GVA_INDUSTRY', 'GVA_SERVICES', 'GVA_PUBLIC',
       'GVA_TOTAL', 'TAXES', 'GDP', 'POP_GDP', 'GDP_CAPITA', 'GVA_MAIN',
       'MUN_EXPENDIT', 'COMP_TOT', 'COMP_A', 'COMP_B', 'COMP_C', 'COMP_D',
       'COMP_E', 'COMP_F', 'COMP_G', 'COMP_H', 'COMP_I', 'COMP_J', 'COMP_K',
       'COMP_L', 'COMP_M', 'COMP_N', 'COMP_O', 'COMP_P', 'COMP_Q', 'COMP_R',
       'COMP_S', 'COMP_T', 'COMP_U', 'HOTELS', 'BEDS', 'Pr_Agencies',
       'Pu_Agencies', 'Pr_Bank', 'Pu_Bank', 'Pr_Assets', 'Pu_Assets', 'Cars',
       'Motorcycles', 'Wheeled_tractor', 'UBER', 'MAC', 'WAL-MART',
       'POST_OFFICES'],
      dtype='object')
```

In [ ]:
```
BR_Cities_df.STATE
```

Out[ ]:
```
0       GO
1       MG
2       GO
3       PA
4       MG
        ..
5573    RS
5574    PR
5575    BA
5576    PA
5577    SP
Name: STATE, Length: 5578, dtype: object
```

In [ ]:
```
qntcidades = BR_Cities_df.STATE.value_counts()
qntcidades
```

Out[ ]:
```
STATE
MG    853
SP    646
RS    499
BA    418
PR    400
SC    295
GO    246
PI    224
PB    224
MA    217
PE    186
CE    184
RN    167
PA    144
MT    141
TO    139
AL    102
RJ     93
MS     79
ES     78
SE     75
AM     62
RO     52
AC     22
AP     16
RR     15
DF      1
Name: count, dtype: int64
```

In [ ]:
```
contestado = 0
for x in BR_Cities_df.STATE.unique():
    contestado+=1
print("Quantidade de estados: {}".format(contestado))
```

```
Quantidade de estados: 27
```

1.3) Check for missing values and handle them appropriately if necessary.

In [ ]:
```
missing_values = BR_Cities_df.isnull().sum()
print(missing_values)
```

```
CITY                0
STATE               0
CAPITAL             0
IBGE_RES_POP        0
IBGE_RES_POP_BRAS   0
                   ..
Wheeled_tractor     0
UBER                0
MAC                 0
WAL-MART            0
POST_OFFICES        0
Length: 81, dtype: int64
```

In [ ]:
```
missing_values = BR_Cities_df.isna().sum()
print(missing_values)
```

```
CITY                0
STATE               0
CAPITAL             0
IBGE_RES_POP        0
IBGE_RES_POP_BRAS   0
                   ..
Wheeled_tractor     0
UBER                0
MAC                 0
WAL-MART            0
POST_OFFICES        0
Length: 81, dtype: int64
```

2) Data Analysis:

2.1) Calculate basic statistics of the dataset (mean, median, min, max, etc.).

In [ ]:
```
BR_Cities_df.describe().transpose()
```

Out[ ]:

	count	mean	std	min	25%	50%	75
CAPITAL	5578.0	0.005916	0.076695	0.0	0.00	0.0	0
IBGE_RES_POP	5578.0	34223.130692	202882.884775	0.0	5217.00	10926.5	23409
IBGE_RES_POP_BRAS	5578.0	34145.726067	201262.674132	0.0	5214.00	10916.0	23380
IBGE_RES_POP_ESTR	5578.0	77.404625	1793.789719	0.0	0.00	0.0	10
IBGE_DU	5578.0	10283.126210	64691.991805	0.0	1565.25	3167.0	6722
...	...	...	...	...	...	...	
Wheeled_tractor	5578.0	5.739871	55.301718	0.0	0.00	0.0	1
UBER	5578.0	0.022409	0.148024	0.0	0.00	0.0	0
MAC	5578.0	0.127465	2.151446	0.0	0.00	0.0	0
WAL-MART	5578.0	0.037827	0.533446	0.0	0.00	0.0	0
POST_OFFICES	5578.0	2.035497	4.378558	0.0	1.00	1.0	2

75 rows × 8 columns

In [ ]:
```
BR_Cities_df.IBGE_RES_POP.describe() ## Statistics for the column IBGE_RES_POP
```

Out[ ]:
```
count    5.578000e+03
mean     3.422313e+04
std      2.028829e+05
min      0.000000e+00
25%      5.217000e+03
50%      1.092650e+04
75%      2.340900e+04
max      1.125350e+07
Name: IBGE_RES_POP, dtype: float64
```

In [ ]:
```
BR_Cities_df.Cars.describe() ## Statistics for the column cars
```

Out[ ]:
```
count    5.578000e+03
mean     9.839788e+03
std      9.175728e+04
min      0.000000e+00
25%      5.990000e+02
50%      1.431500e+03
75%      4.084000e+03
max      5.740995e+06
Name: Cars, dtype: float64
```

In [ ]:
```
BR_Cities_df.head(100).Cars.describe() ## Statistics for 100 head items from the
```

Out[ ]:
```
count      100.000000
mean      3477.960000
std       5904.379559
min          2.000000
25%        605.750000
50%       1301.000000
75%       3189.250000
max      36568.000000
Name: Cars, dtype: float64
```

In [ ]:
```
BR_Cities_df.IBGE_RES_POP.describe() ## Statistics for the column IBGE_RES_POP
```

Out[ ]:
```
count    5.578000e+03
mean     3.422313e+04
std      2.028829e+05
min      0.000000e+00
25%      5.217000e+03
50%      1.092650e+04
75%      2.340900e+04
max      1.125350e+07
Name: IBGE_RES_POP, dtype: float64
```

In [ ]:
```
BR_Cities_df.describe().transpose()
```

Out[ ]:

	count	mean	std	min	25%	50%	75
CAPITAL	5578.0	0.005916	0.076695	0.0	0.00	0.0	0
IBGE_RES_POP	5578.0	34223.130692	202882.884775	0.0	5217.00	10926.5	23409
IBGE_RES_POP_BRAS	5578.0	34145.726067	201262.674132	0.0	5214.00	10916.0	23380
IBGE_RES_POP_ESTR	5578.0	77.404625	1793.789719	0.0	0.00	0.0	10
IBGE_DU	5578.0	10283.126210	64691.991805	0.0	1565.25	3167.0	6722
...	...	...	...	...	...	...	
Wheeled_tractor	5578.0	5.739871	55.301718	0.0	0.00	0.0	1
UBER	5578.0	0.022409	0.148024	0.0	0.00	0.0	0
MAC	5578.0	0.127465	2.151446	0.0	0.00	0.0	0
WAL-MART	5578.0	0.037827	0.533446	0.0	0.00	0.0	0
POST_OFFICES	5578.0	2.035497	4.378558	0.0	1.00	1.0	2

75 rows × 8 columns

Explore the distribution of scores using histograms and box plots.

In [ ]:
```
BR_Cities_df
```

Out[ ]:

	CITY	STATE	CAPITAL	IBGE_RES_POP	IBGE_RES_POP_BRAS	IBGE_RES_POP_
0	Abadia De Goiás	GO	0	6876	6876	
1	Abadia Dos Dourados	MG	0	6704	6704	
2	Abadiânia	GO	0	15757	15609	
3	Abaetetuba	PA	0	141100	141040	
4	Abaeté	MG	0	22690	22690	
...	...	...	...	...	...	
5573	Áurea	RS	0	3665	3665	
5574	Ângulo	PR	0	2859	2844	
5575	Érico Cardoso	BA	0	10859	10859	
5576	Óbidos	PA	0	49333	49324	
5577	Óleo	SP	0	2673	2673	

5578 rows × 81 columns

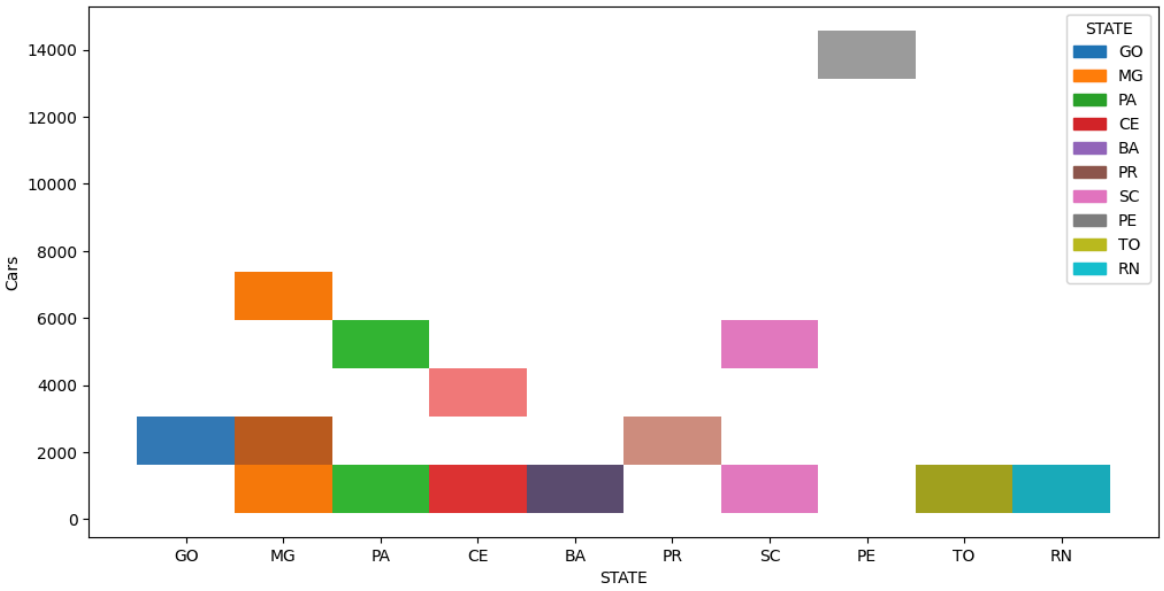
In [ ]:
```
BR_Cities_df.columns
```

Out[ ]:
```
Index(['CITY', 'STATE', 'CAPITAL', 'IBGE_RES_POP', 'IBGE_RES_POP_BRAS',
       'IBGE_RES_POP_ESTR', 'IBGE_DU', 'IBGE_DU_URBAN', 'IBGE_DU_RURAL',
       'IBGE_POP', 'IBGE_1', 'IBGE_1-4', 'IBGE_5-9', 'IBGE_10-14',
       'IBGE_15-59', 'IBGE_60+', 'IBGE_PLANTED_AREA', 'IBGE_CROP_PRODUCTION_$',
       'IDHM Ranking 2010', 'IDHM', 'IDHM_Renda', 'IDHM_Longevidade',
       'IDHM_Educacao', 'LONG', 'LAT', 'ALT', 'PAY_TV', 'FIXED_PHONES', 'AREA',
       'REGIAO_TUR', 'CATEGORIA_TUR', 'ESTIMATED_POP', 'RURAL_URBAN',
       'GVA_AGROPEC', 'GVA_INDUSTRY', 'GVA_SERVICES', 'GVA_PUBLIC',
       'GVA_TOTAL', 'TAXES', 'GDP', 'POP_GDP', 'GDP_CAPITA', 'GVA_MAIN',
       'MUN_EXPENDIT', 'COMP_TOT', 'COMP_A', 'COMP_B', 'COMP_C', 'COMP_D',
       'COMP_E', 'COMP_F', 'COMP_G', 'COMP_H', 'COMP_I', 'COMP_J', 'COMP_K',
       'COMP_L', 'COMP_M', 'COMP_N', 'COMP_O', 'COMP_P', 'COMP_Q', 'COMP_R',
       'COMP_S', 'COMP_T', 'COMP_U', 'HOTELS', 'BEDS', 'Pr_Agencies',
       'Pu_Agencies', 'Pr_Bank', 'Pu_Bank', 'Pr_Assets', 'Pu_Assets', 'Cars',
       'Motorcycles', 'Wheeled_tractor', 'UBER', 'MAC', 'WAL-MART',
       'POST_OFFICES'],
      dtype='object')
```

--- Analising Quantity of Cars by City in Each State

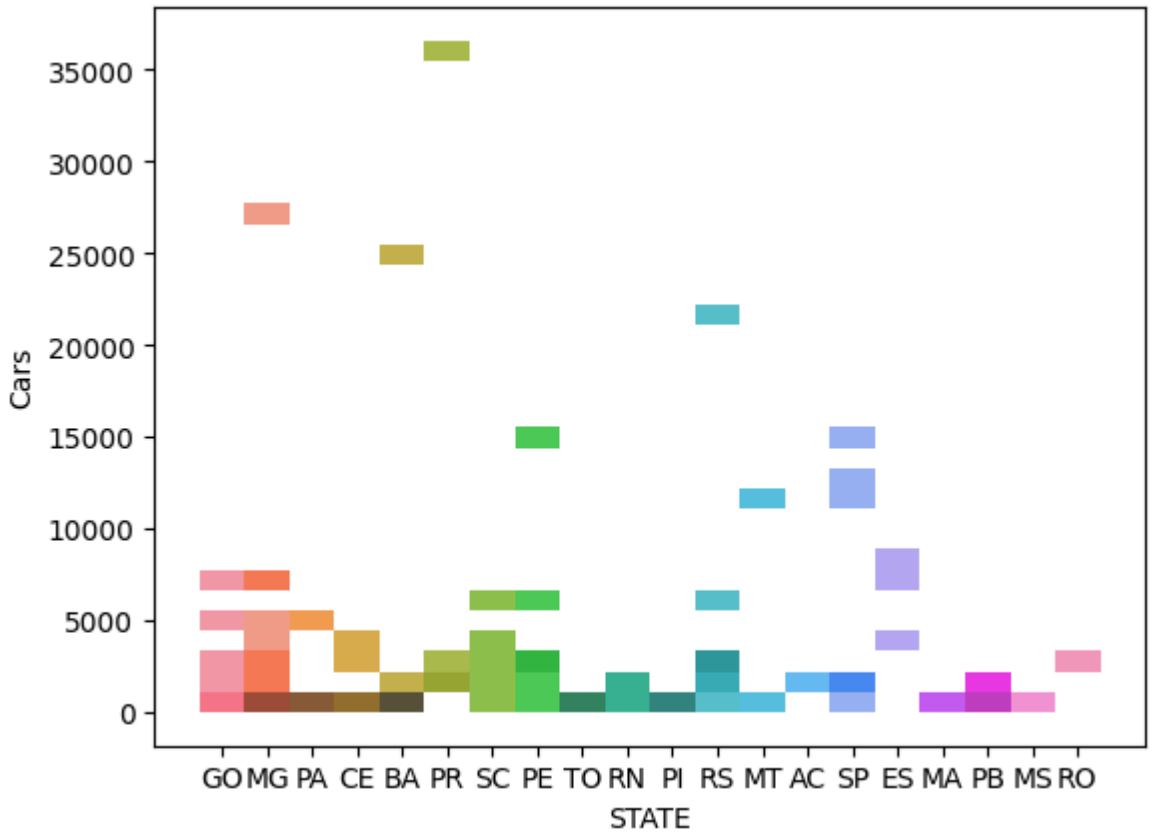
In [ ]:
```
plt.figure(figsize=(12, 6))
sns.histplot(data=BR_Cities_df.head(20), x='STATE', y='Cars', hue='STATE')
```

Out[ ]:
```
<Axes: xlabel='STATE', ylabel='Cars'>
```



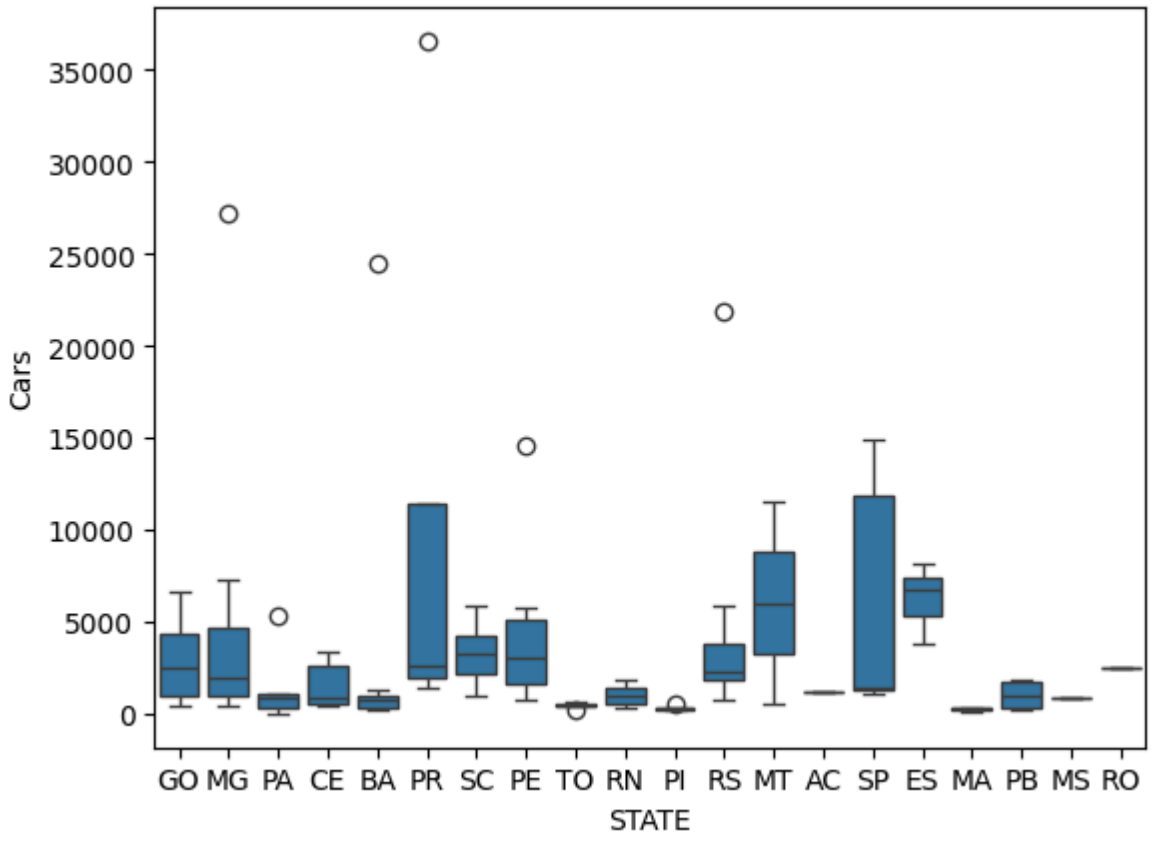
In [ ]:
```
sns.histplot(data=BR_Cities_df.head(100), x='STATE', y='Cars', hue='STATE',legen
```

Out[ ]:
```
<Axes: xlabel='STATE', ylabel='Cars'>
```



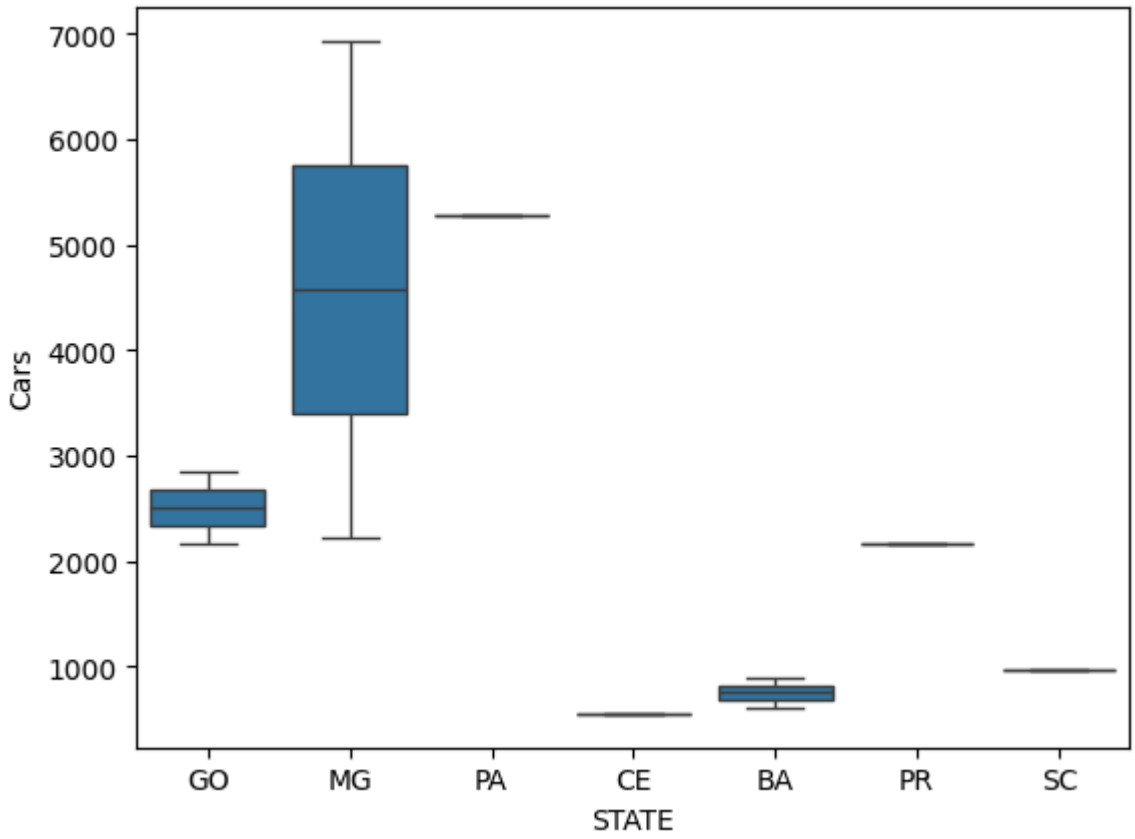
In [ ]:
```
sns.boxplot(x='STATE', y='Cars', data=BR_Cities_df.head(100))
```

Out[ ]: <Axes: xlabel='STATE', ylabel='Cars'>



In [ ]:
```
sns.boxplot(x='STATE', y='Cars', data=BR_Cities_df.head(10))
```

Out[ ]: <Axes: xlabel='STATE', ylabel='Cars'>



3) Data Visualization:

3.1) Use matplotlib and seaborn to create visualizations such as:

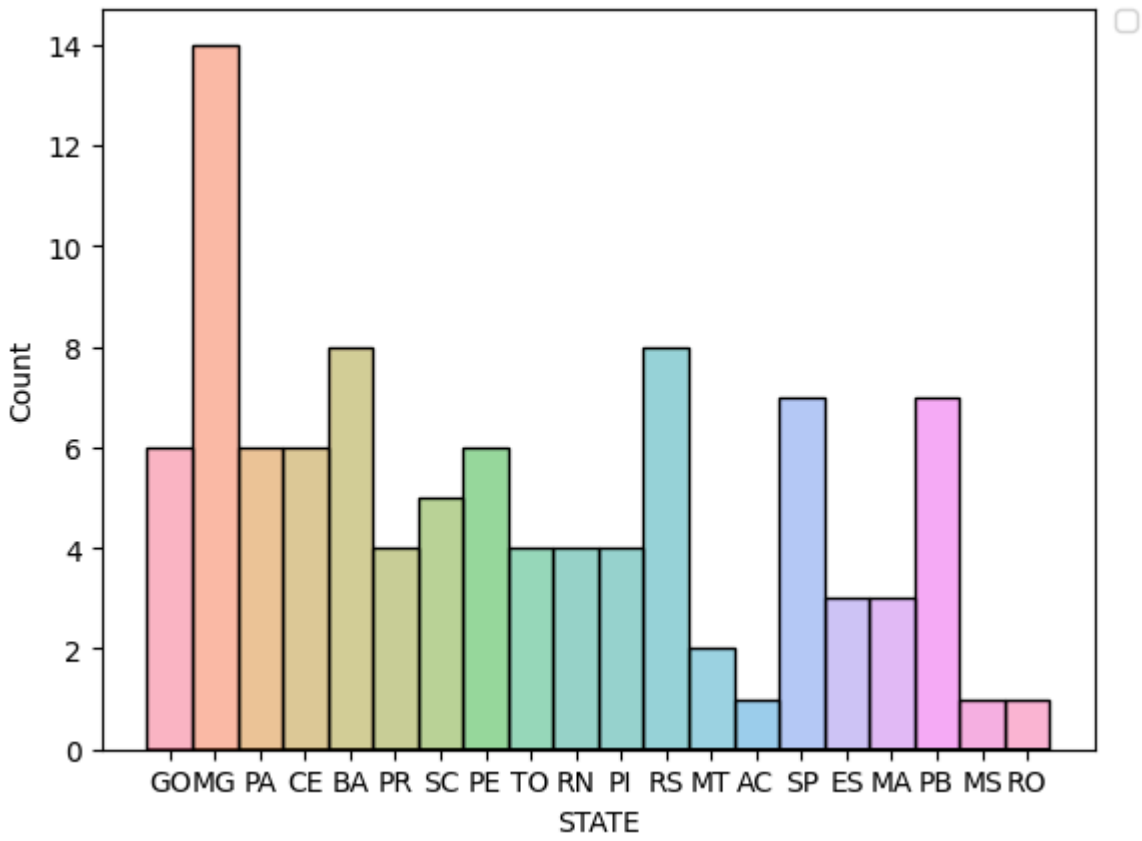
a) Histograms of scores in different subjects.

--- Analising Quantity of Cities in Each State

In [ ]:
```
sns.histplot(data=BR_Cities_df.head(100), x='STATE', hue='STATE')
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```

```
C:\Users\Enzo\AppData\Local\Temp\ipykernel_33976\4146132448.py:2: UserWarning: No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
  plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x2677625f620>
```

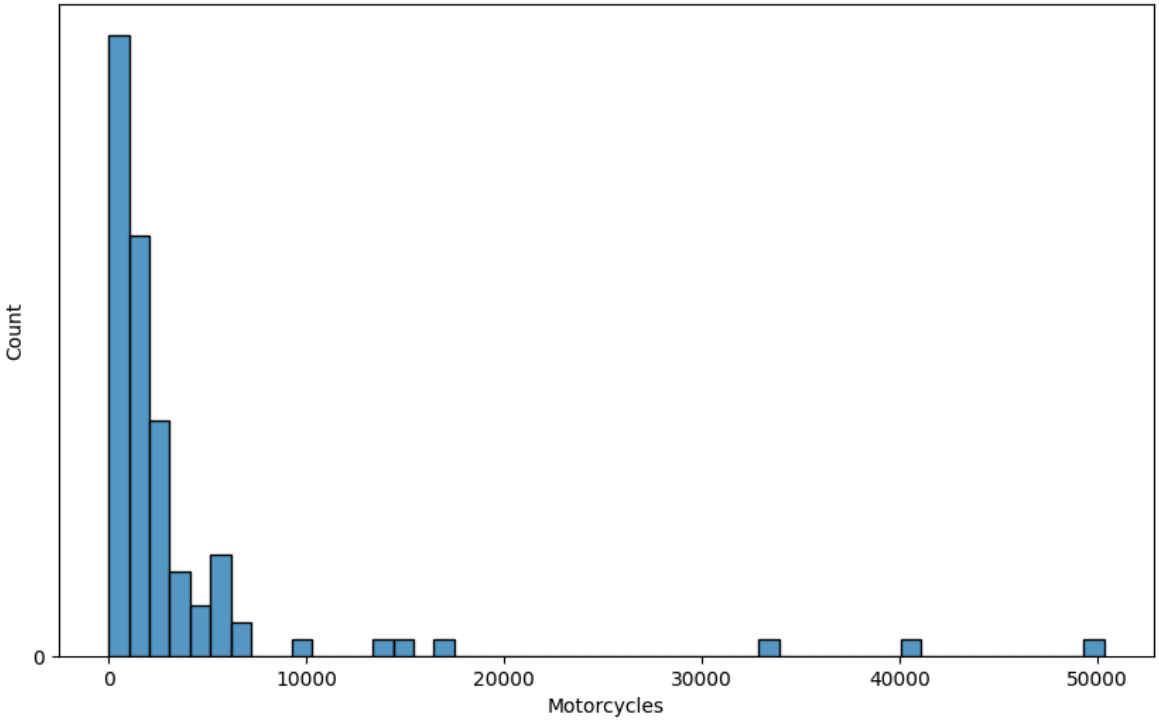


--- Analising Quantity of Motorcycles by City in Each State

Quantidade de Ocorrências de número de Motocicletas

In [ ]:
```
plt.figure(figsize=(10,6))
# set ticks of y axis to 1 by 1
plt.yticks(np.arange(0, 1000000, step=100000))
sns.histplot(x='Motorcycles', data=BR_Cities_df[100:199],)
```

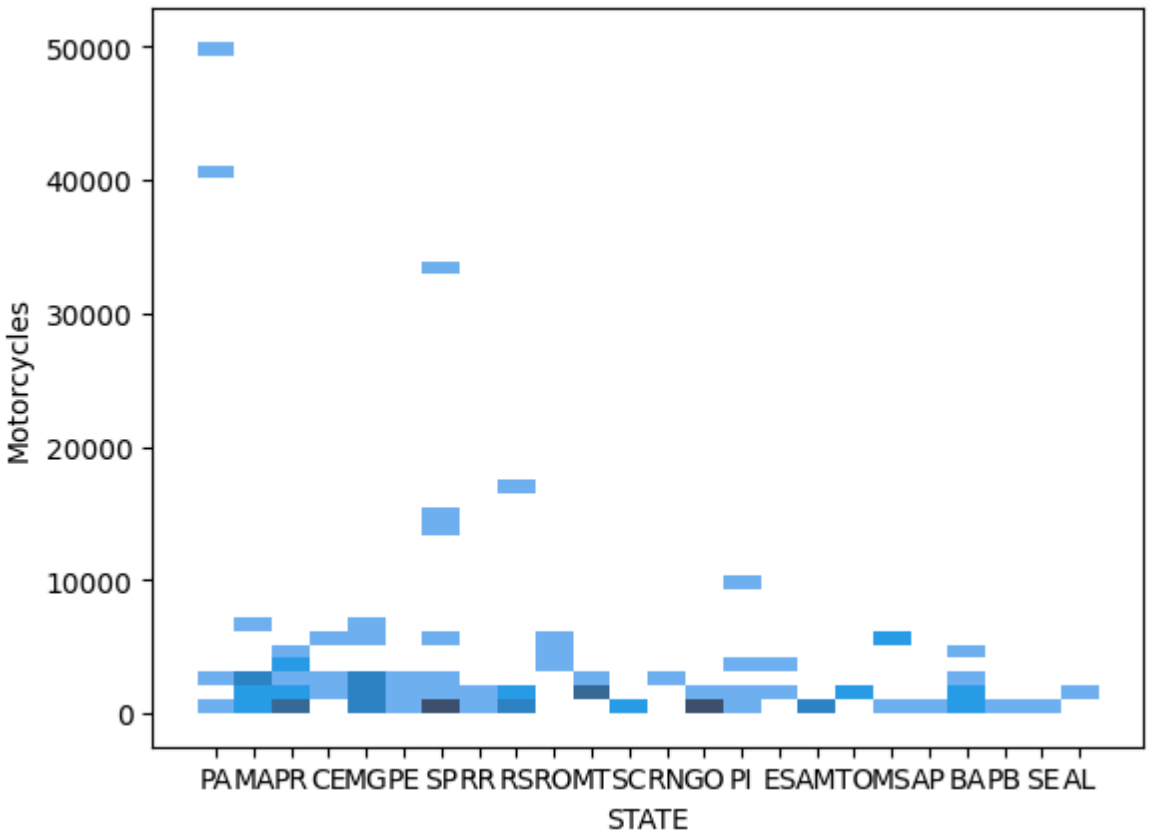
Out[ ]: <Axes: xlabel='Motorcycles', ylabel='Count'>



Quantity of Motorcycles by City in Each State

In [ ]:
```
sns.histplot(x='STATE', y='Motorcycles', data=BR_Cities_df[100:199])
```

Out[ ]:
```
<Axes: xlabel='STATE', ylabel='Motorcycles'>
```



--- Analising Quantity by City in Filtered States

In [ ]:
```
state = ['SP', 'RJ', 'MG', 'ES']
filtered_df = BR_Cities_df[BR_Cities_df['STATE'].isin(state)]
filtered_df
```

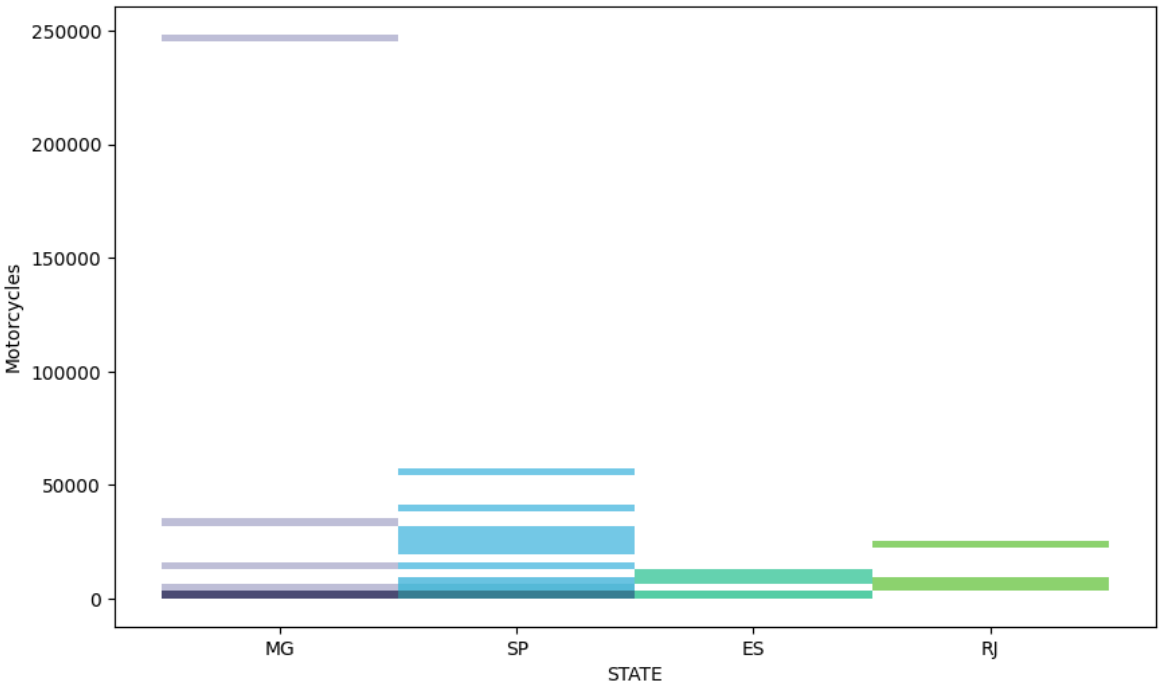
Out[ ]:

	CITY	STATE	CAPITAL	IBGE_RES_POP	IBGE_RES_POP_BRAS	IBGE_RES_POP
1	Abadia Dos Dourados	MG	0	6704	6704	
4	Abaeté	MG	0	22690	22690	
12	Abre Campo	MG	0	13311	13294	
15	Acaiaca	MG	0	3920	3920	
27	Adamantina	SP	0	33797	33769	
...	...	...	...	...	...	
5569	Águia Branca	ES	0	9519	9513	
5570	Álvares Florence	SP	0	3897	3894	
5571	Álvares Machado	SP	0	23513	23493	
5572	Álvaro De Carvalho	SP	0	4650	4645	
5577	Óleo	SP	0	2673	2673	

1670 rows × 81 columns

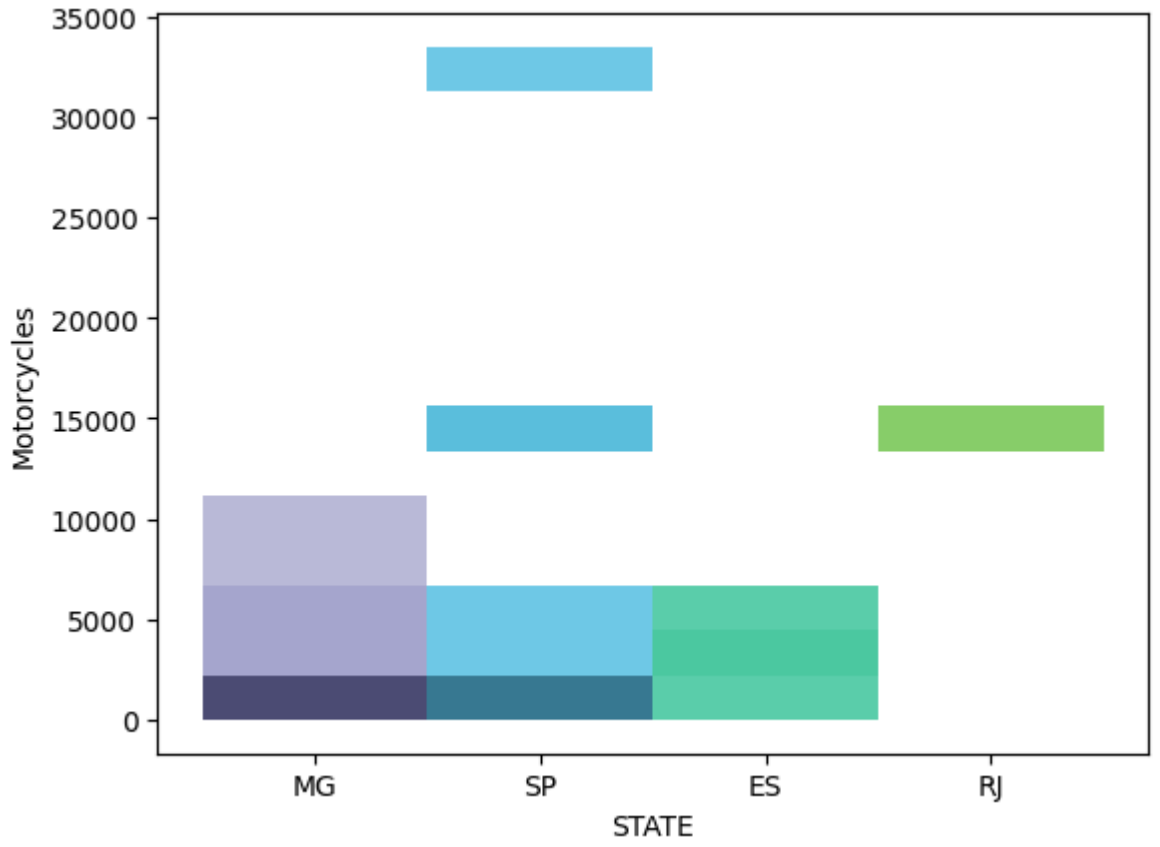
In [ ]:
```
state = ['SP', 'RJ', 'MG', 'ES']
filtered_df = BR_Cities_df[BR_Cities_df['STATE'].isin(state)]
plt.figure(figsize=(10,6))
sns.histplot(data=filtered_df[100:159], x='STATE', y='Motorcycles', hue='STATE',
```

Out[ ]:
```
<Axes: xlabel='STATE', ylabel='Motorcycles'>
```



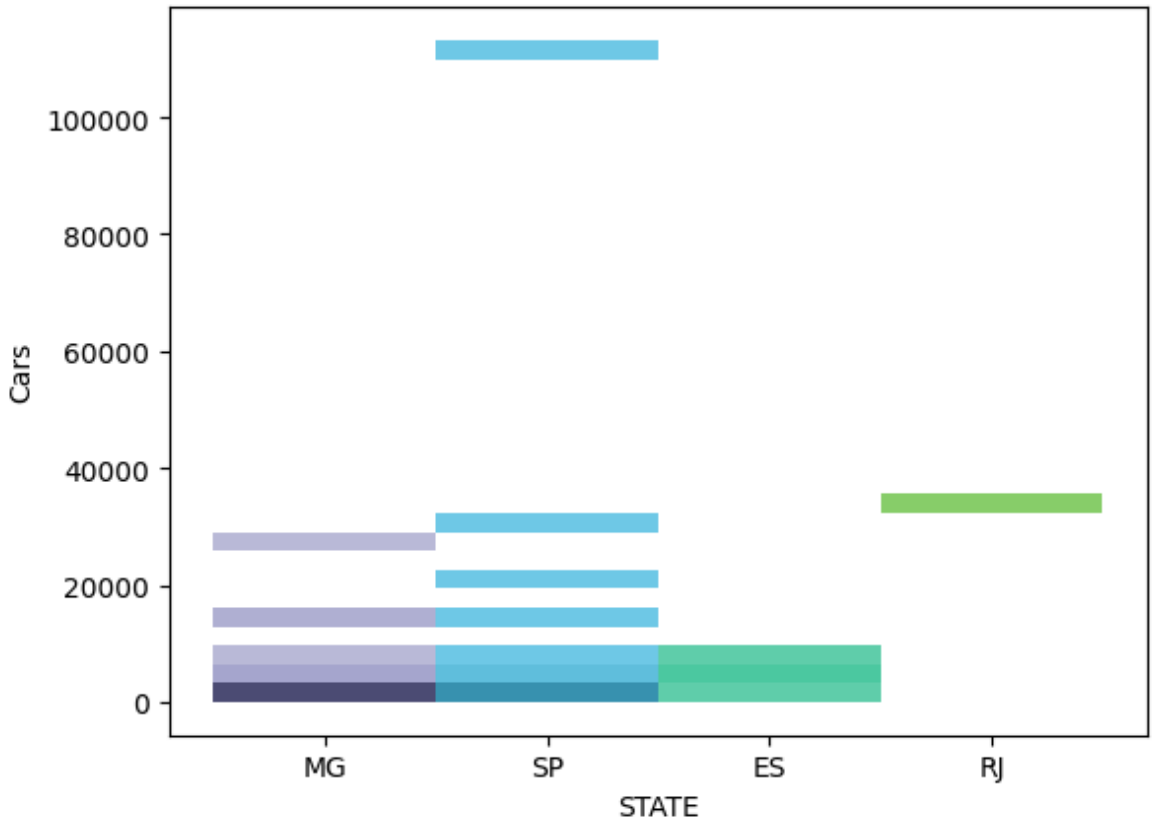
In [ ]:
```
state = ['SP', 'RJ', 'MG', 'ES']
filtered_df = BR_Cities_df[BR_Cities_df['STATE'].isin(state)]
sns.histplot(data=filtered_df[10:50], x='STATE', y='Motorcycles', hue='STATE',pa
```

Out[ ]:
```
<Axes: xlabel='STATE', ylabel='Motorcycles'>
```



In [ ]:
```
state = ['SP', 'RJ', 'MG', 'ES']
filtered_df = BR_Cities_df[BR_Cities_df['STATE'].isin(state)]
sns.histplot(data=filtered_df[10:50], x='STATE', y='Cars', hue='STATE',palette='
```

Out[ ]:
```
<Axes: xlabel='STATE', ylabel='Cars'>
```



--- Analising Quantity by City in Filtered Cities

--- Please notice that the database contains incorect data in the column City when you compare with the column Capital. But it won't affect our study

In [ ]:
```
capitais = True
filtered_df = BR_Cities_df[BR_Cities_df['CAPITAL'].isin([capitais])]
filtered_df.sort_values(by='STATE', ascending=True)
```

Out[ ]:

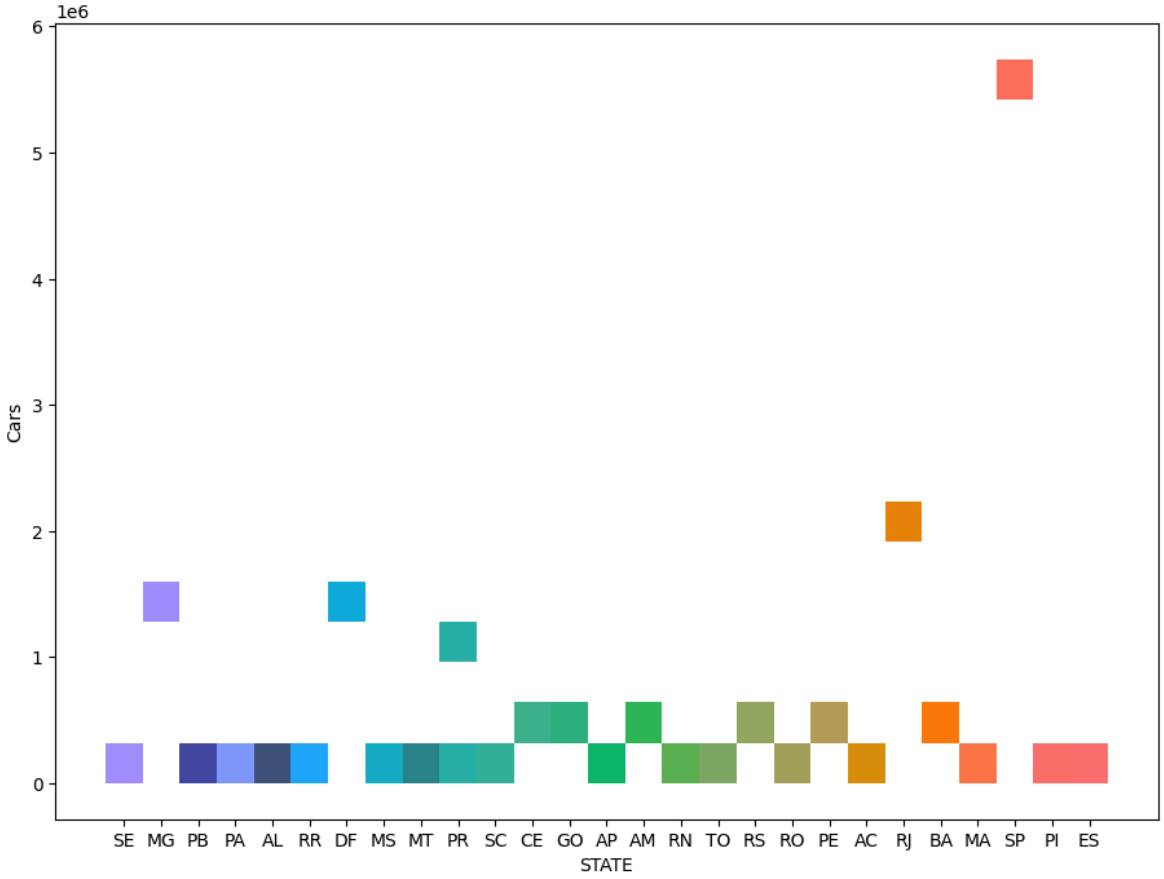
	CITY	STATE	CAPITAL	IBGE_RES_POP	IBGE_RES_POP_BRAS	IBGE_RES_POP
4161	Rio Branco	AC	1	336038	335634	
559	Belém	AL	1	4551	4551	
2806	Maceió	AL	1	932748	932134	
932	Campo Grande	AL	1	9032	9032	
2849	Manaus	AM	1	1802014	1798773	
2797	Macapá	AP	1	398204	397926	
4300	Salvador	BA	1	2675656	2671290	
1793	Fortaleza	CE	1	2452185	2449109	
719	Brasília	DF	1	2570160	2564370	
5491	Vitória	ES	1	327801	326735	
1900	Goiânia	GO	1	1302001	1299718	
4959	São Luís	MA	1	1014837	1014202	
550	Belo Horizonte	MG	1	2375151	2369063	
933	Campo Grande	MS	1	786797	785017	
1441	Cuiabá	MT	1	551098	550726	
4160	Rio Branco	MT	1	5070	5070	
558	Belém	PA	1	1393399	1391623	
609	Boa Vista	PB	1	6227	6227	
2580	João Pessoa	PB	1	723515	722363	
557	Belém	PB	1	17093	17093	
4078	Recife	PE	1	1537704	1535289	
5179	Teresina	PI	1	814230	814100	
3496	Palmas	PR	1	42888	42860	
1458	Curitiba	PR	1	1751907	1743036	
4175	Rio De Janeiro	RJ	1	6320446	6264915	
3206	Natal	RN	1	803739	802686	
3927	Porto Velho	RO	1	428527	427841	
608	Boa Vista	RR	1	284313	283523	
3897	Porto Alegre	RS	1	1409351	1403450	
1773	Florianópolis	SC	1	421240	417674	

	CITY	STATE	CAPITAL	IBGE_RES_POP	IBGE_RES_POP_BRAS	IBGE_RES_POP
256	Aracaju	SE	1	571149	570674	
4997	São Paulo	SP	1	11253503	11133776	1
3495	Palmas	TO	1	228332	228131	

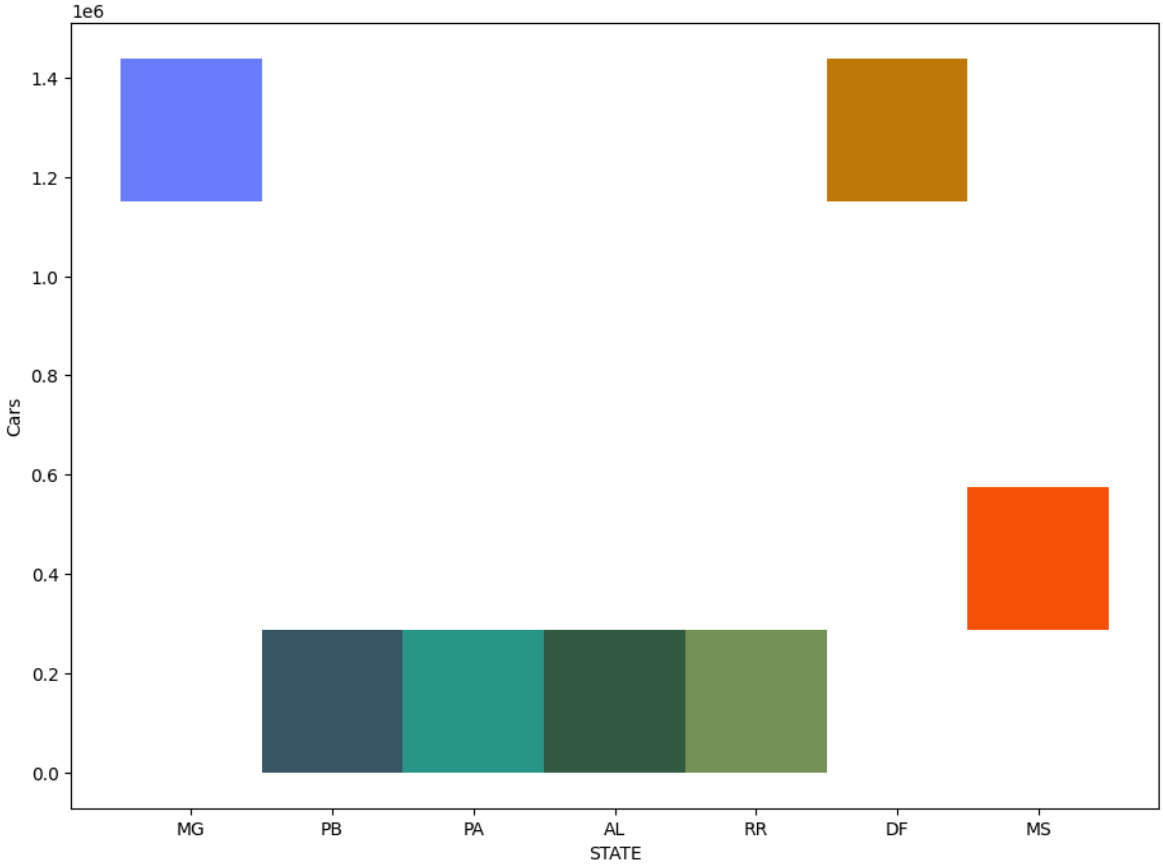
33 rows × 81 columns

In [ ]:
```
capitais = True
filtered_df = BR_Cities_df[BR_Cities_df['CAPITAL'].isin([capitais])]
plt.figure(figsize=(11, 8))
sns.histplot(data=filtered_df, x='STATE', y='Cars', hue='STATE',palette='rainbow
```

Out[ ]: <Axes: xlabel='STATE', ylabel='Cars'>

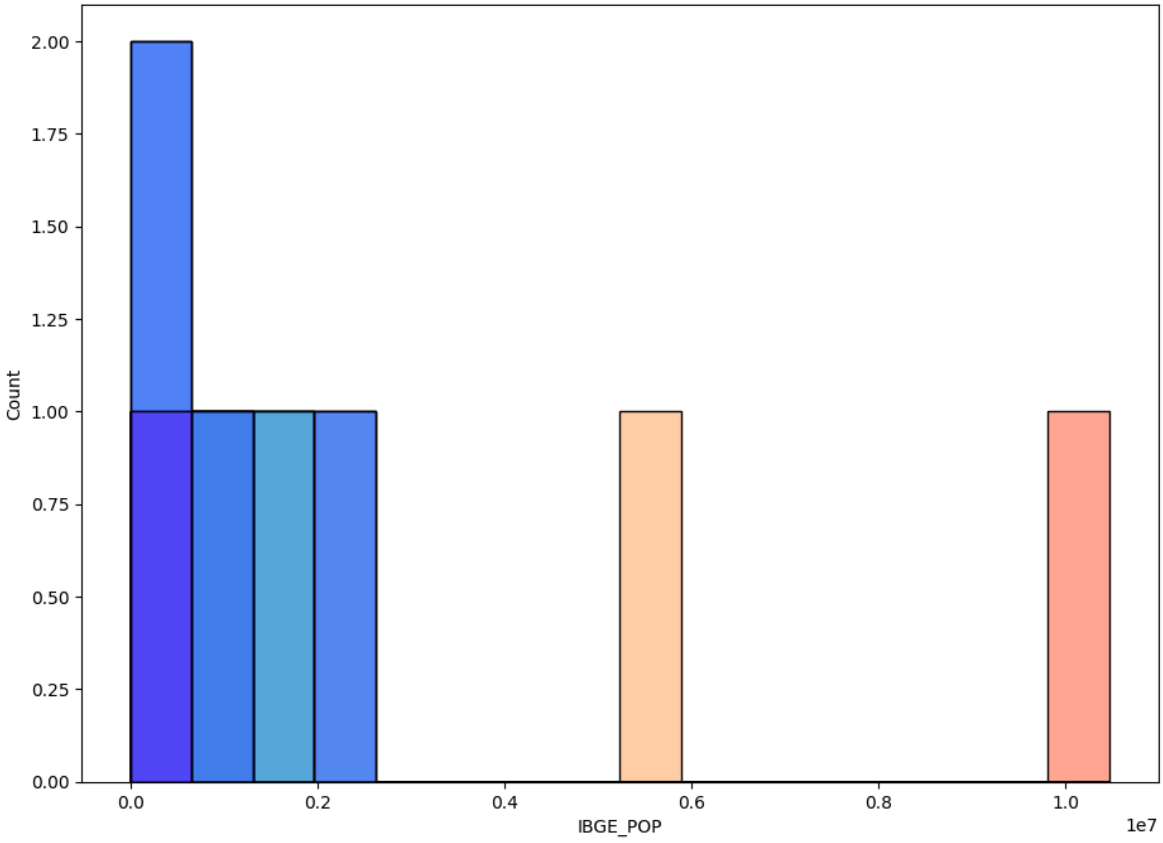


In [ ]:
```
capitais = True
filtered_df = BR_Cities_df[BR_Cities_df['CAPITAL'].isin([capitais])]
plt.figure(figsize=(11, 8))
plot = sns.histplot(data=filtered_df[1:10], x='STATE',y="Cars", hue='STATE',pale
```



In [ ]:
```
capitais = True
filtered_df = BR_Cities_df[BR_Cities_df['CAPITAL'].isin([capitais])]
plt.figure(figsize=(11, 8))
sns.histplot(data=filtered_df, x='IBGE_POP', hue='STATE',palette='rainbow', lege
```

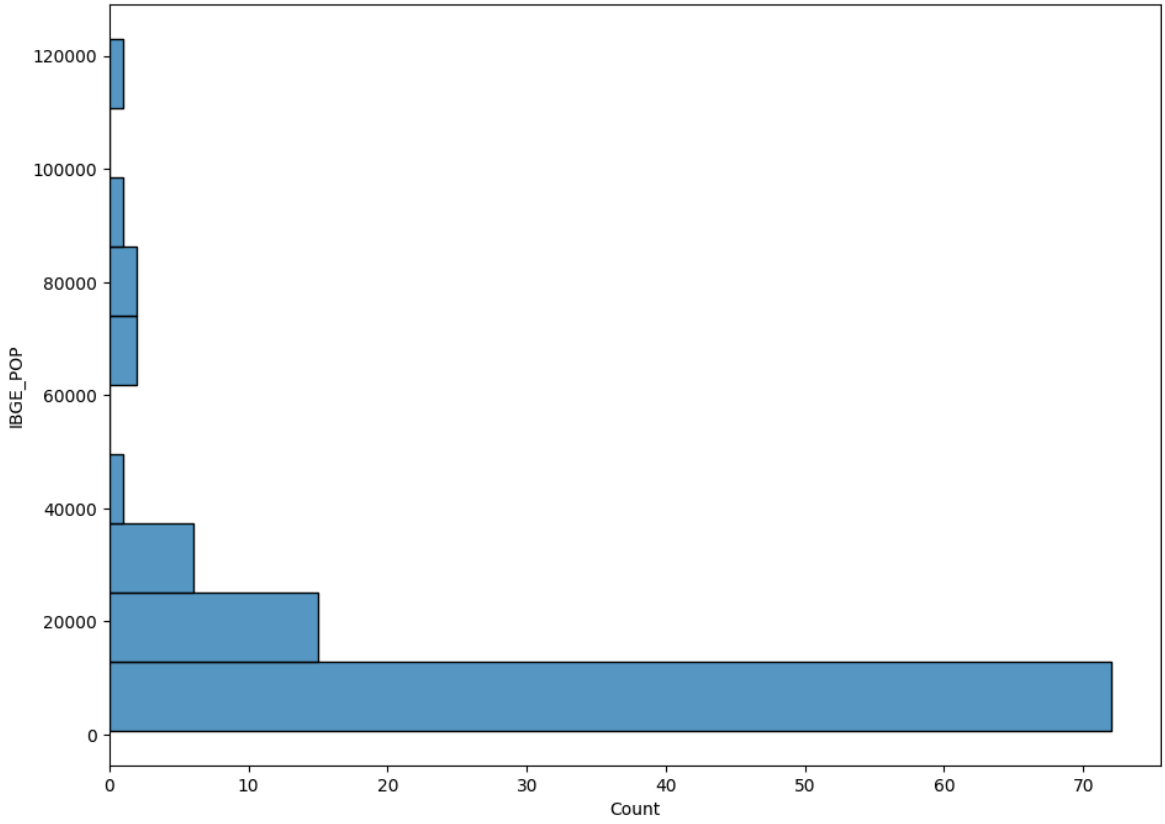
Out[ ]:
```
<Axes: xlabel='IBGE_POP', ylabel='Count'>
```



In [ ]:
```
plt.figure(figsize=(11, 8))
sns.histplot(data=BR_Cities_df.head(100), y='IBGE_POP',palette='rainbow',bins=10
```

```
C:\Users\Enzo\AppData\Local\Temp\ipykernel_33976\372057058.py:2: UserWarning: Ign
oring `palette` because no `hue` variable has been assigned.
  sns.histplot(data=BR_Cities_df.head(100), y='IBGE_POP',palette='rainbow',bins=1
0)
```

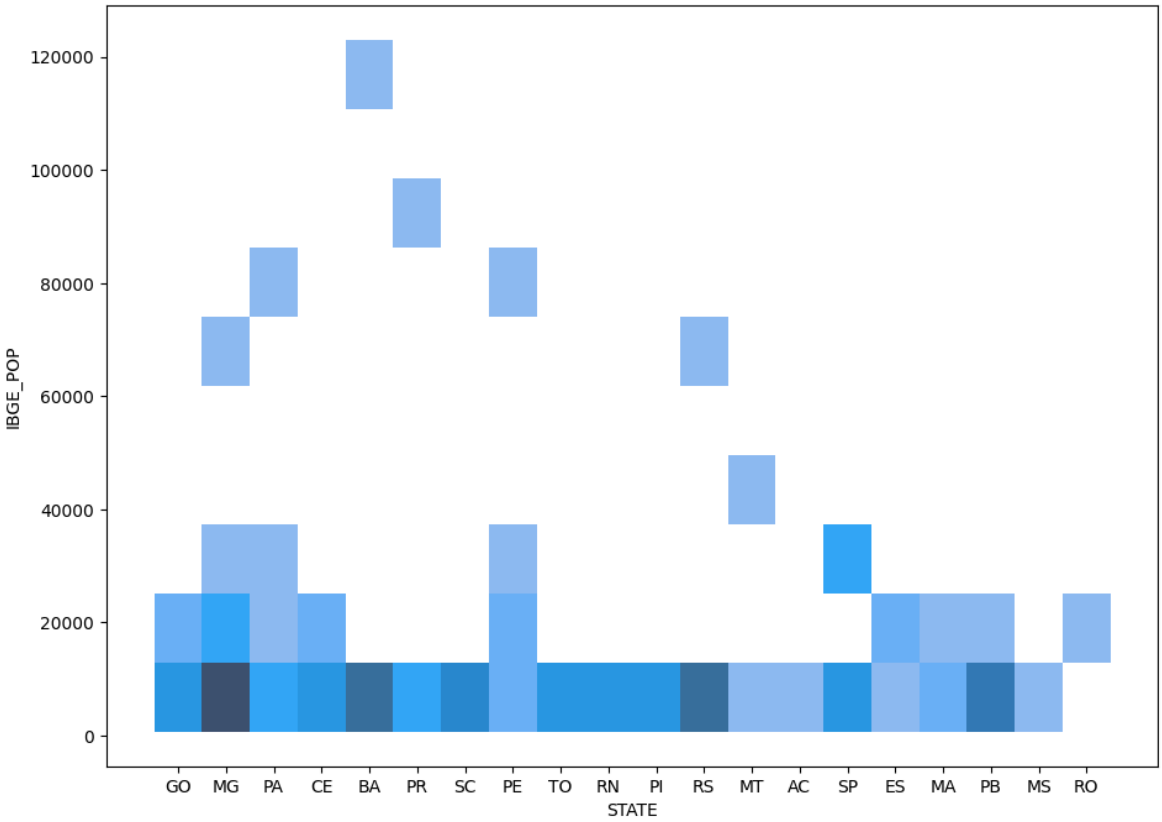
Out[ ]: <Axes: xlabel='Count', ylabel='IBGE_POP'>



In [ ]:
```
plt.figure(figsize=(11, 8))
sns.histplot(data=BR_Cities_df.head(100), y='IBGE_POP',x='STATE',palette='rainbo
```

```
C:\Users\Enzo\AppData\Local\Temp\ipykernel_33976\2562253573.py:2: UserWarning: Ig
noring `palette` because no `hue` variable has been assigned.
  sns.histplot(data=BR_Cities_df.head(100), y='IBGE_POP',x='STATE',palette='rainb
ow',bins=10)
```

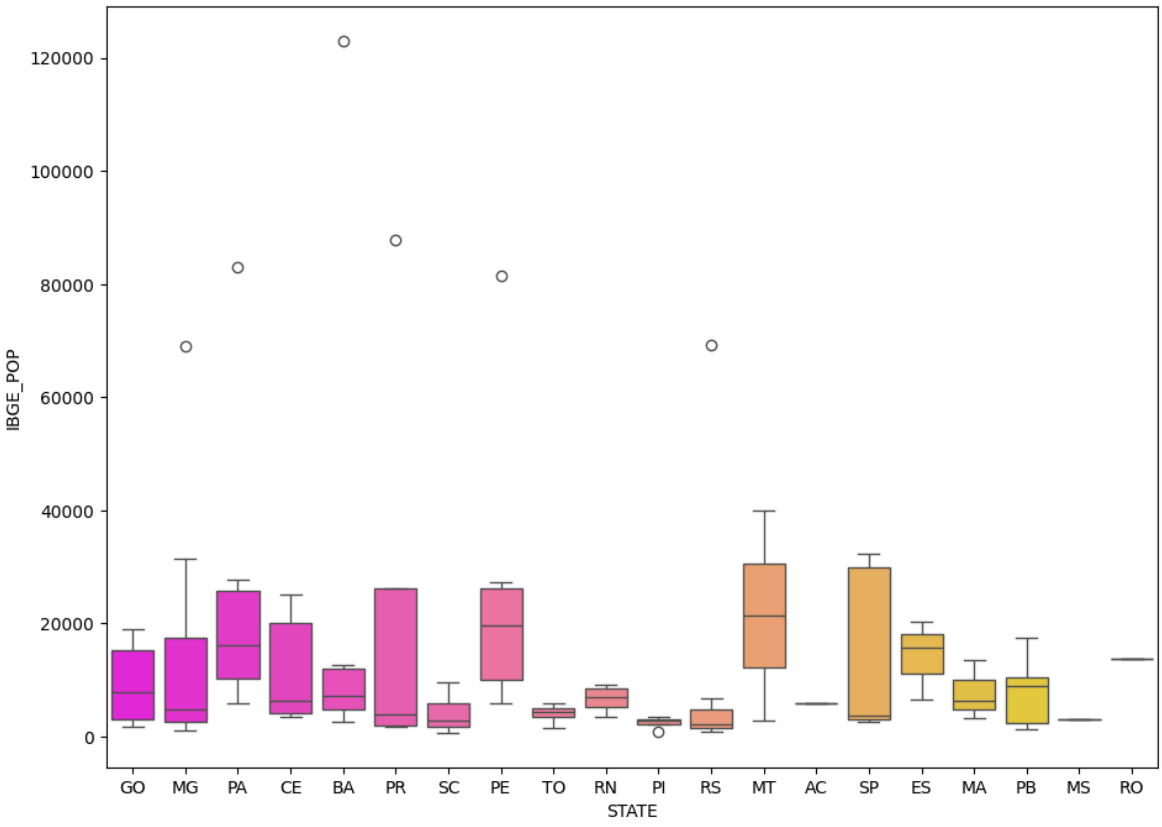
Out[ ]: <Axes: xlabel='STATE', ylabel='IBGE_POP'>



b) Box plots to compare scores across different categories (e.g., gender, parental ##### level of education).

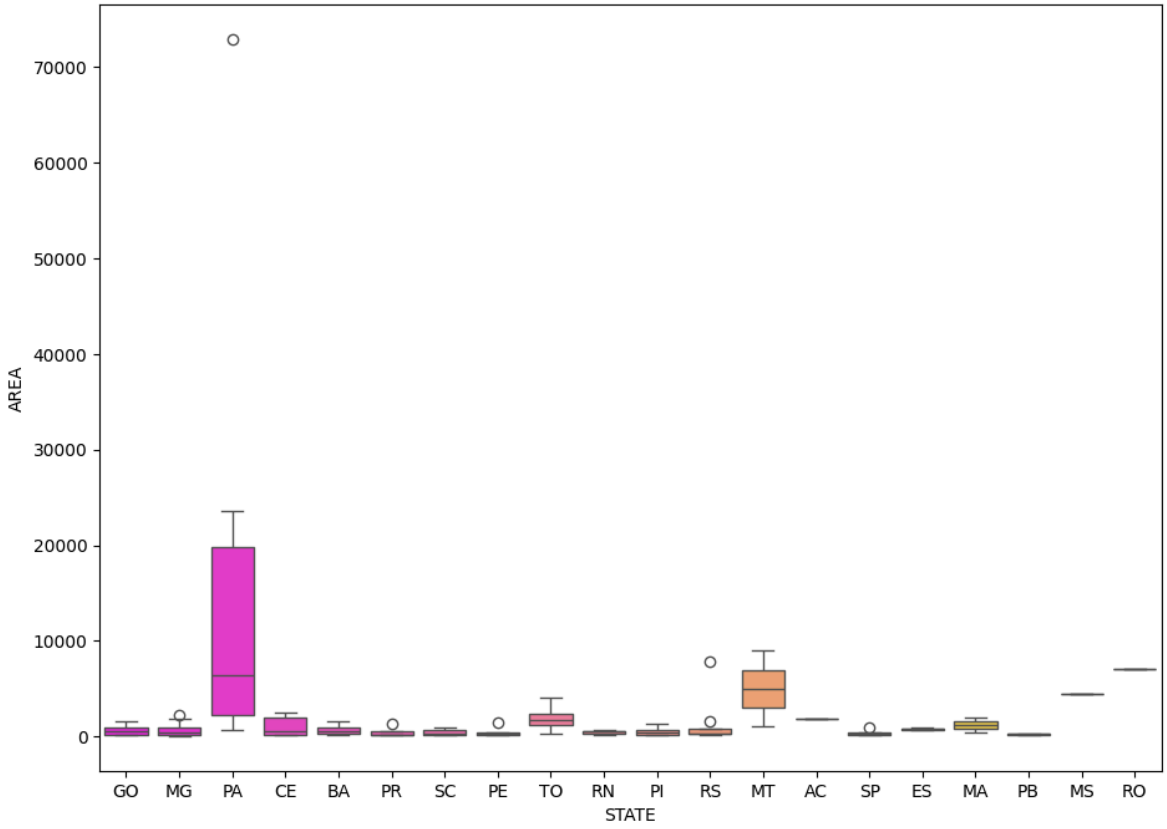
In [ ]:
```
plt.figure(figsize=(11, 8))
sns.boxplot(x='STATE', y='IBGE_POP', data=BR_Cities_df.head(100), hue = 'STATE',
```

Out[ ]:
```
<Axes: xlabel='STATE', ylabel='IBGE_POP'>
```



In [ ]:
```
plt.figure(figsize=(11, 8))
sns.boxplot(x='STATE', y='AREA', data=BR_Cities_df.head(100), hue = 'STATE',pale
```

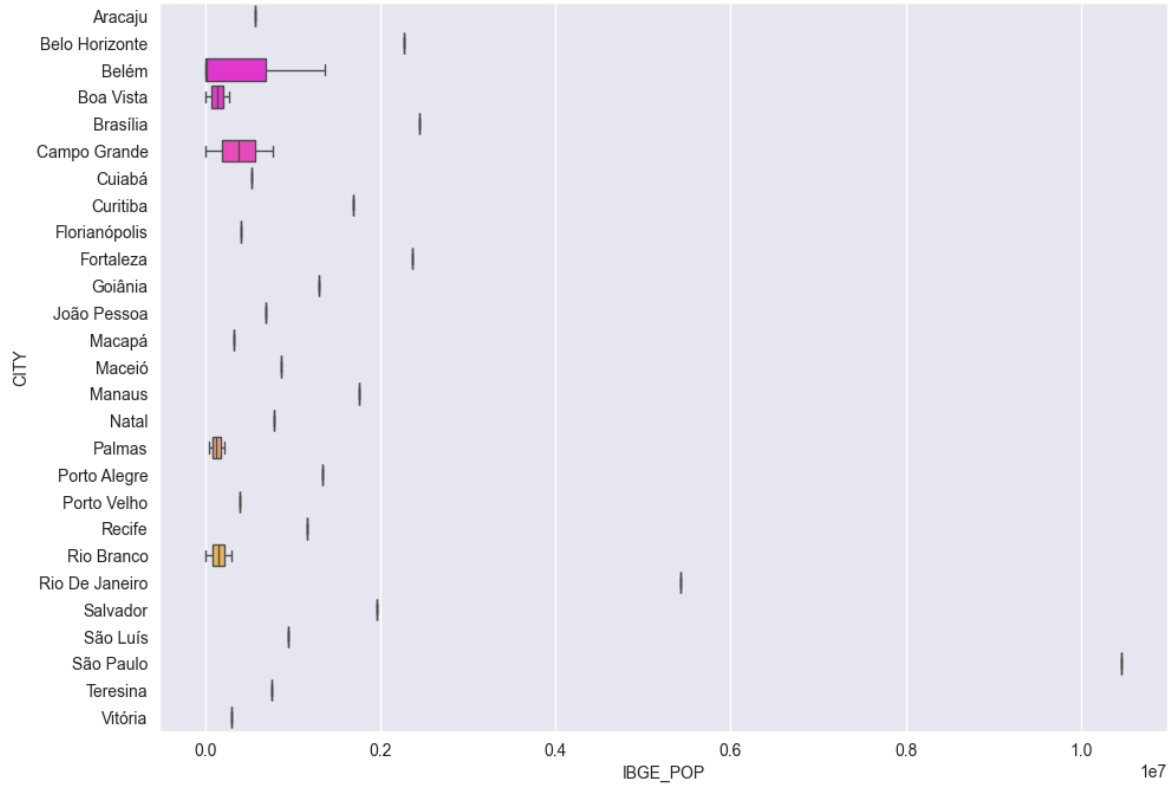
Out[ ]:
```
<Axes: xlabel='STATE', ylabel='AREA'>
```



In [ ]:
```
plt.figure(figsize=(11, 8))
sns.set_style('darkgrid')
sns.boxplot(y='CITY' ,x='IBGE_POP', data=filtered_df, palette= 'spring')
```

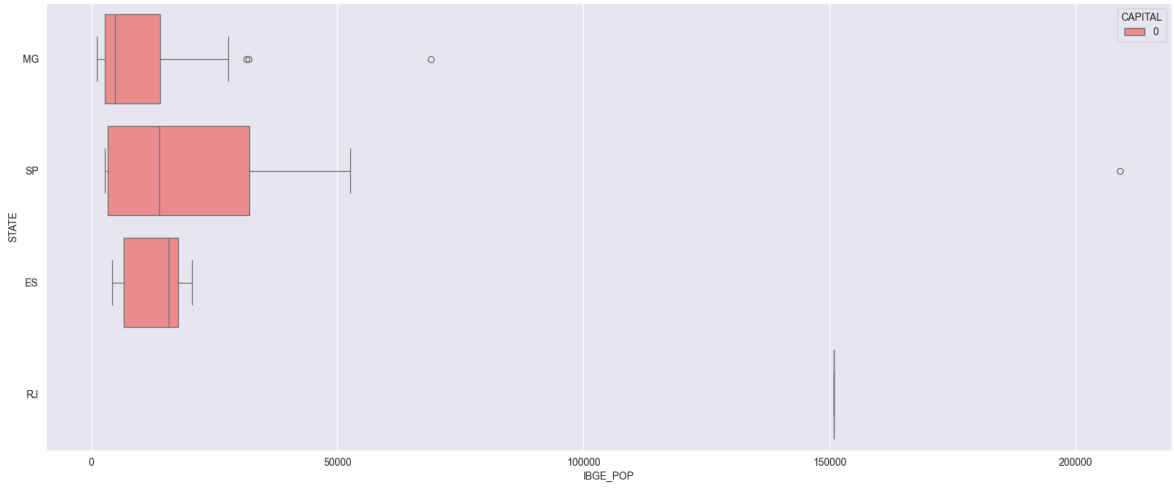
```
C:\Users\Enzo\AppData\Local\Temp\ipykernel_33976\1708772767.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.boxplot(y='CITY' ,x='IBGE_POP', data=filtered_df, palette= 'spring')
```

Out[ ]:
```
<Axes: xlabel='IBGE_POP', ylabel='CITY'>
```

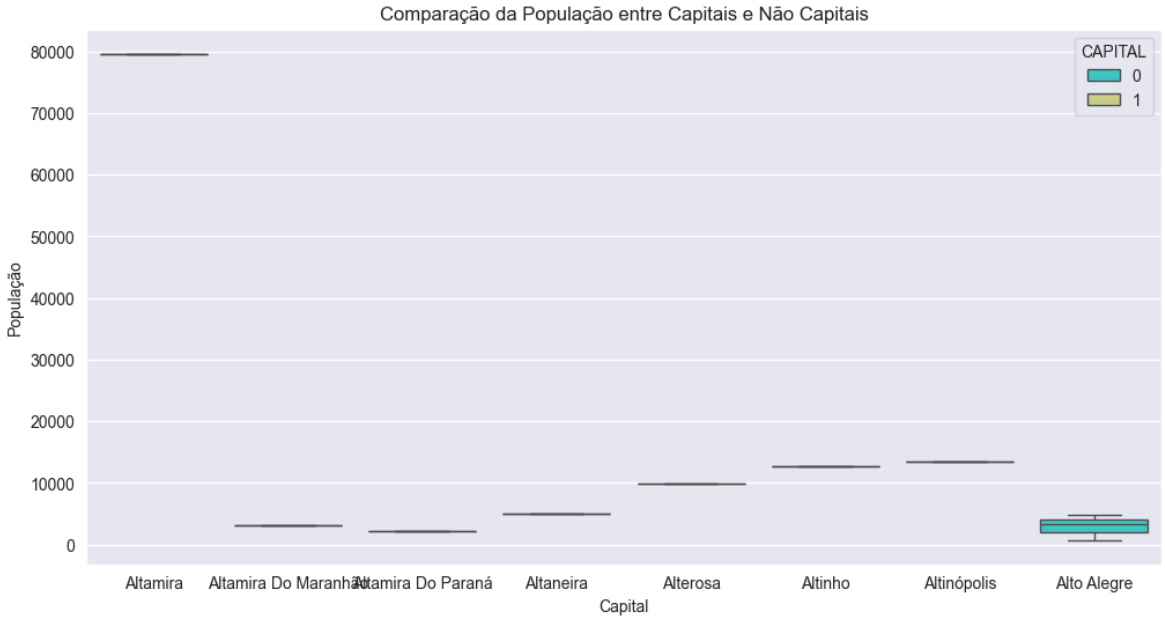


In [ ]:
```
southeastStates = ['SP', 'RJ', 'MG', 'ES']
southeastFilteredDF = BR_Cities_df[BR_Cities_df['STATE'].isin(southeastStates)]
plt.figure(figsize=(20, 8))
sns.set_style('darkgrid')
sns.boxplot(y='STATE' ,x='IBGE_POP', data=southeastFilteredDF.head(50), hue = 'C
```

Out[ ]:
```
<Axes: xlabel='IBGE_POP', ylabel='STATE'>
```

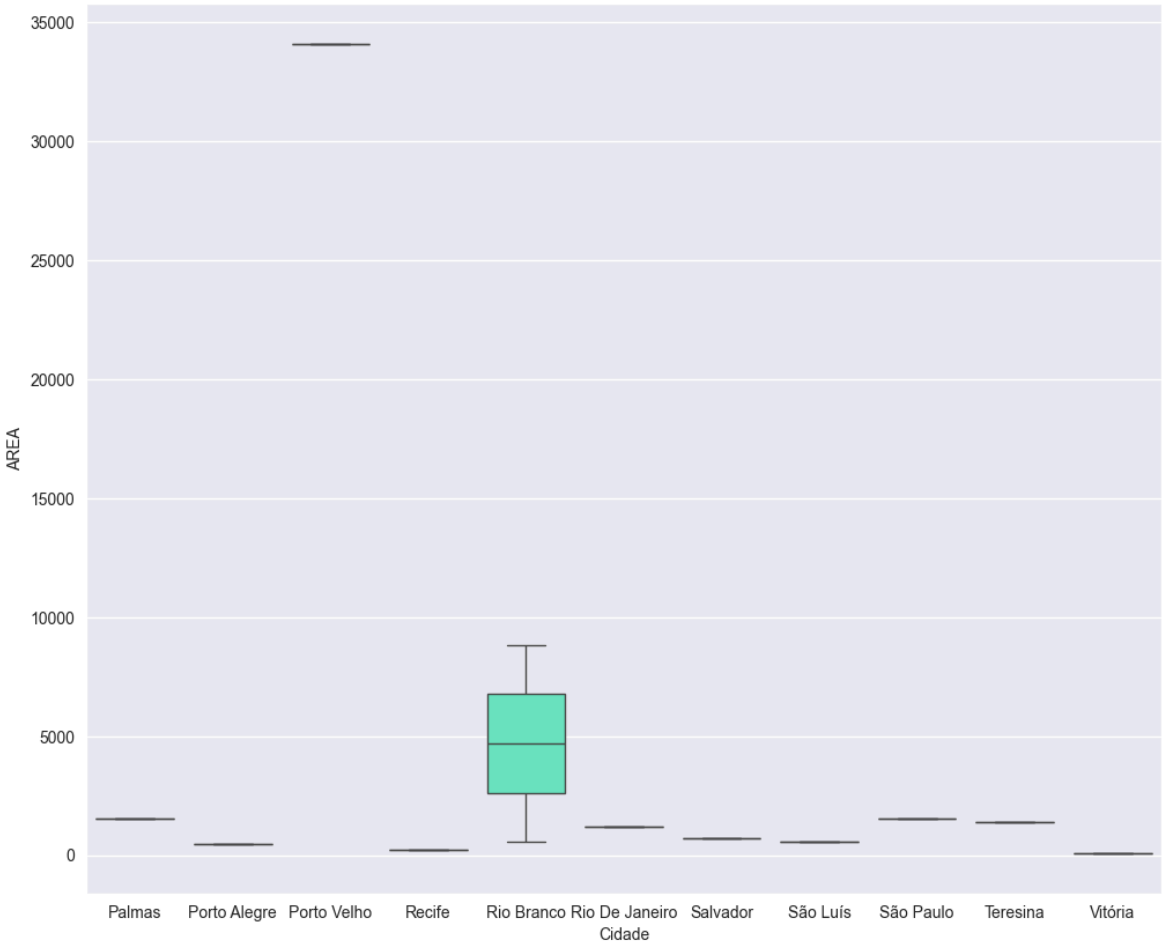


In [ ]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(data=BR_Cities_df, x=BR_Cities_df['CITY'][100:110], y='IBGE_POP', hu
plt.title('Comparação da População entre Capitais e Não Capitais')
plt.xlabel('Capital')
plt.ylabel('População')
plt.show()
```



In [ ]:
```
plt.figure(figsize=(12, 10))
sns.boxplot(data=filtered_df[21:33], x='CITY', y='AREA', hue='CITY', palette='ra
plt.xlabel('Cidade')
```

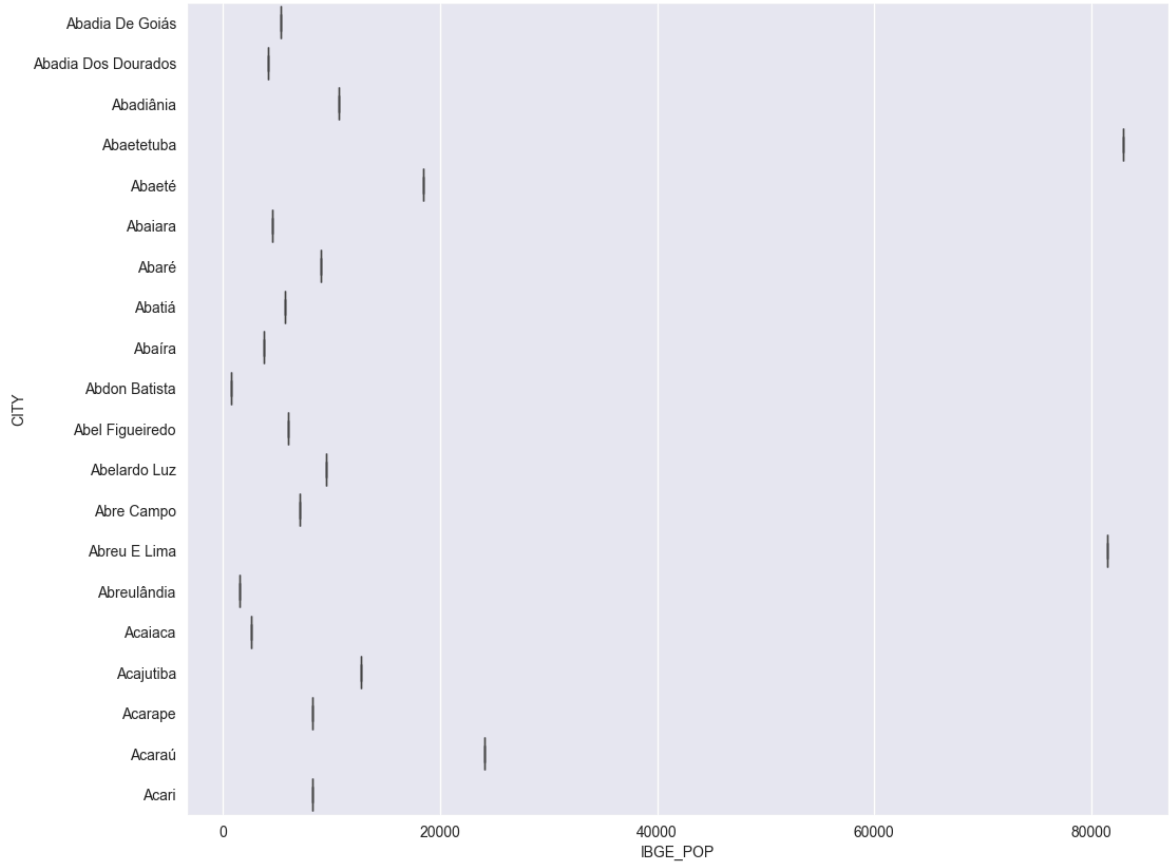
Out[ ]:
```
Text(0.5, 0, 'Cidade')
```



In [ ]:
```
spCities = BR_Cities_df[BR_Cities_df['STATE'] == 'SP']
plt.figure(figsize=(12, 10))
sns.boxplot( data = BR_Cities_df.head(20), y = 'CITY', x='IBGE_POP',palette='rai
```

```
C:\Users\Enzo\AppData\Local\Temp\ipykernel_33976\2511709882.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.boxplot( data = BR_Cities_df.head(20), y = 'CITY', x='IBGE_POP',palette='ra
inbow')
```

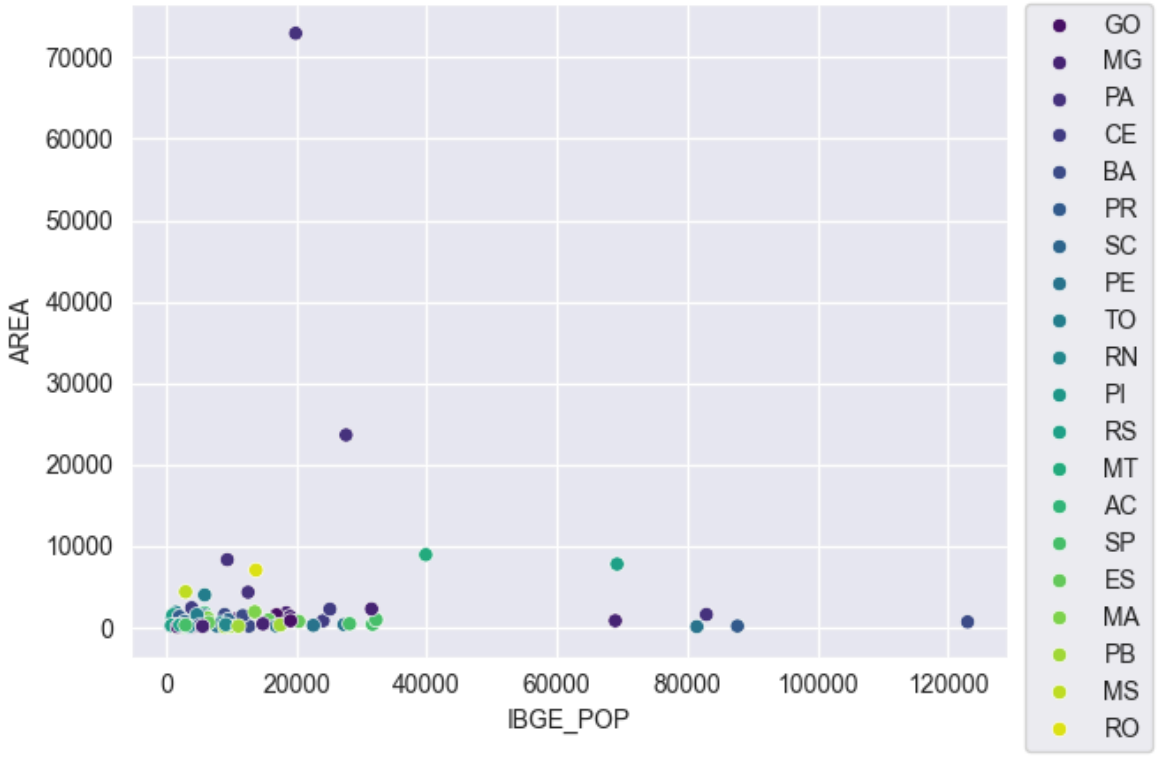
Out[ ]: <Axes: xlabel='IBGE_POP', ylabel='CITY'>



c) Scatter plots to explore relationships between variables (e.g., math vs. reading ##### scores).

In [ ]:
```
sns.scatterplot(data=BR_Cities_df.head(100), x='IBGE_POP', y='AREA', hue='STATE'
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```

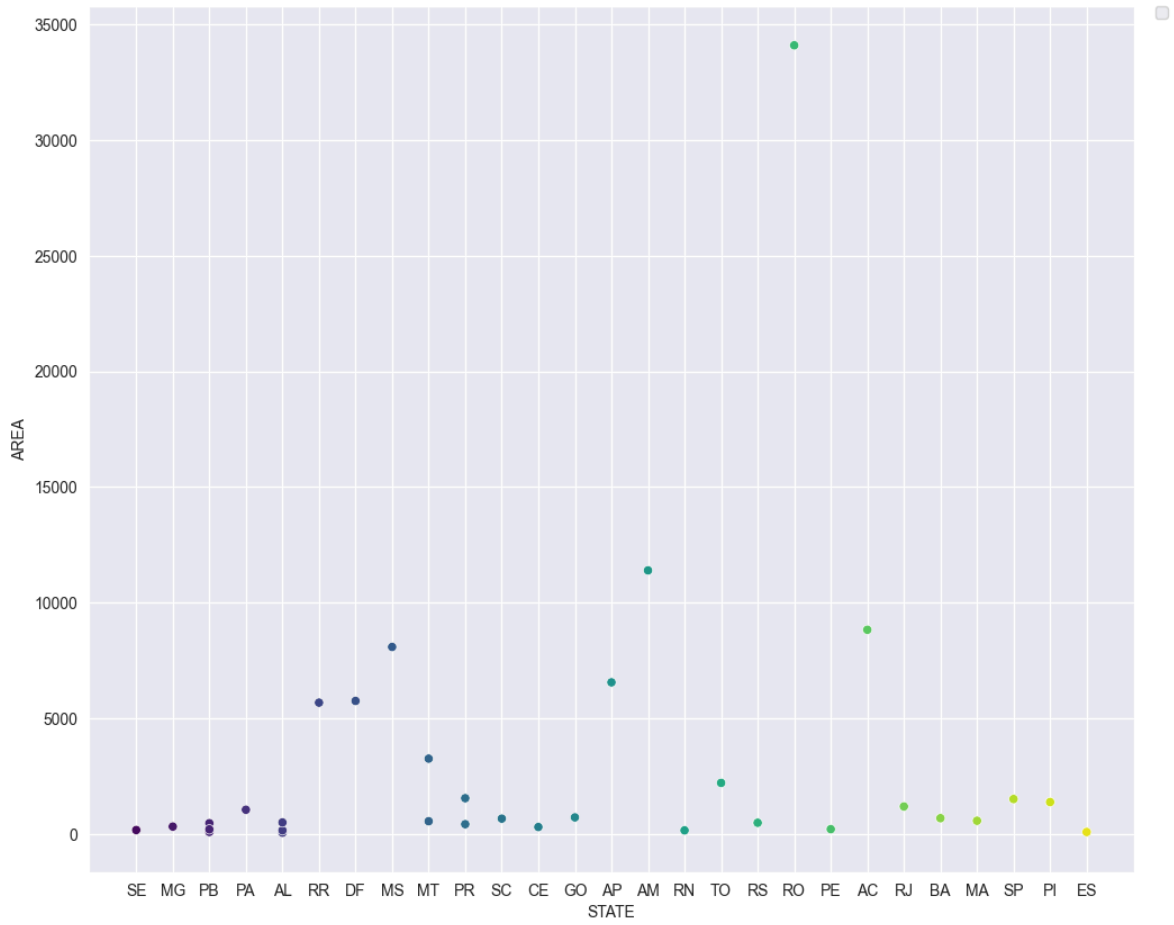
Out[ ]: <matplotlib.legend.Legend at 0x2677b3b8e30>



In [ ]:
```
plt.figure(figsize=(12, 10))
sns.scatterplot(data=filtered_df, x='STATE', y='AREA', hue='STATE', palette='vir
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```

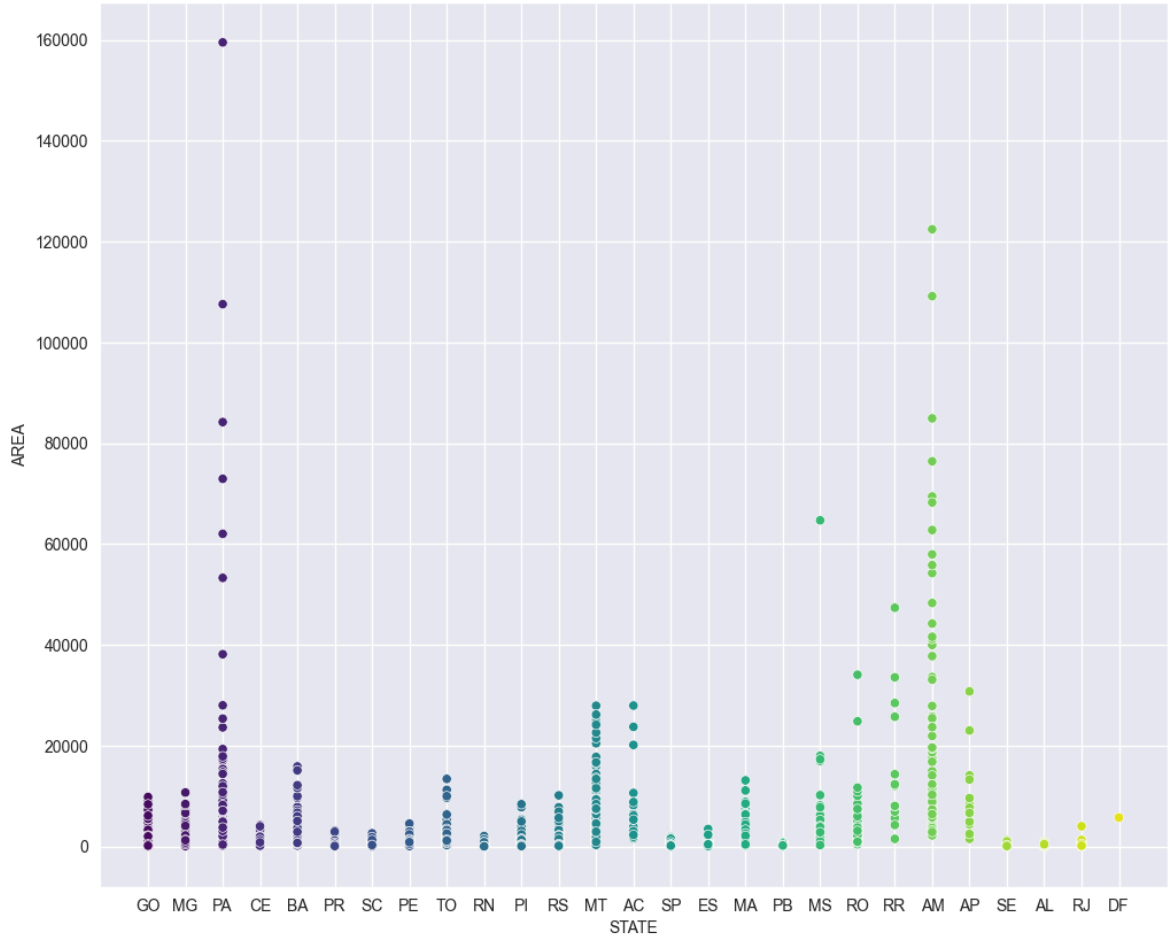
```
C:\Users\Enzo\AppData\Local\Temp\ipykernel_33976\1243795343.py:3: UserWarning: No
artists with labels found to put in legend.  Note that artists whose label start
with an underscore are ignored when legend() is called with no argument.
  plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x2677b591ee0>
```



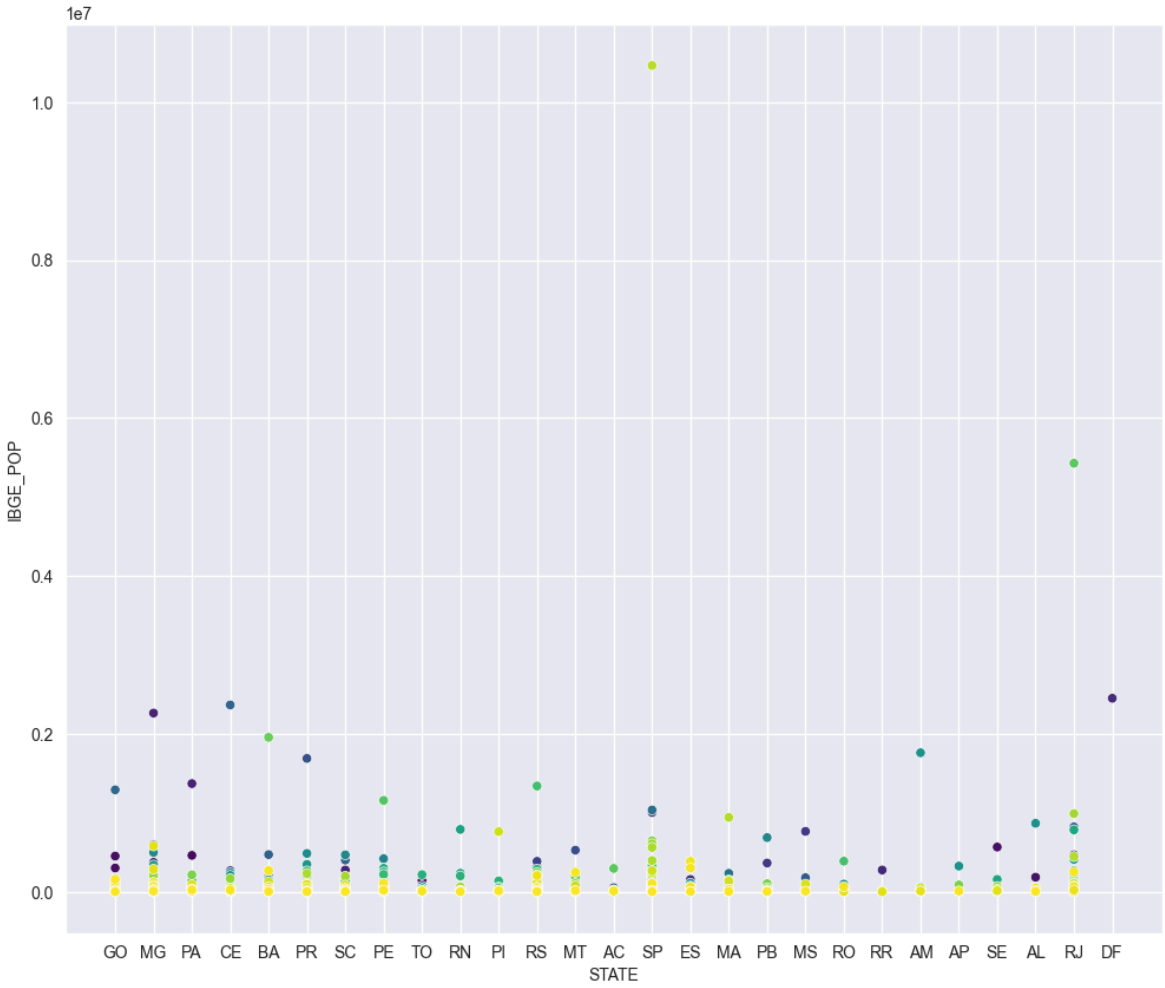
In [ ]:
```
plt.figure(figsize=(12, 10))
sns.scatterplot(data=BR_Cities_df, x='STATE', y='AREA', hue='STATE', palette='vi
```

Out[ ]:
```
<Axes: xlabel='STATE', ylabel='AREA'>
```



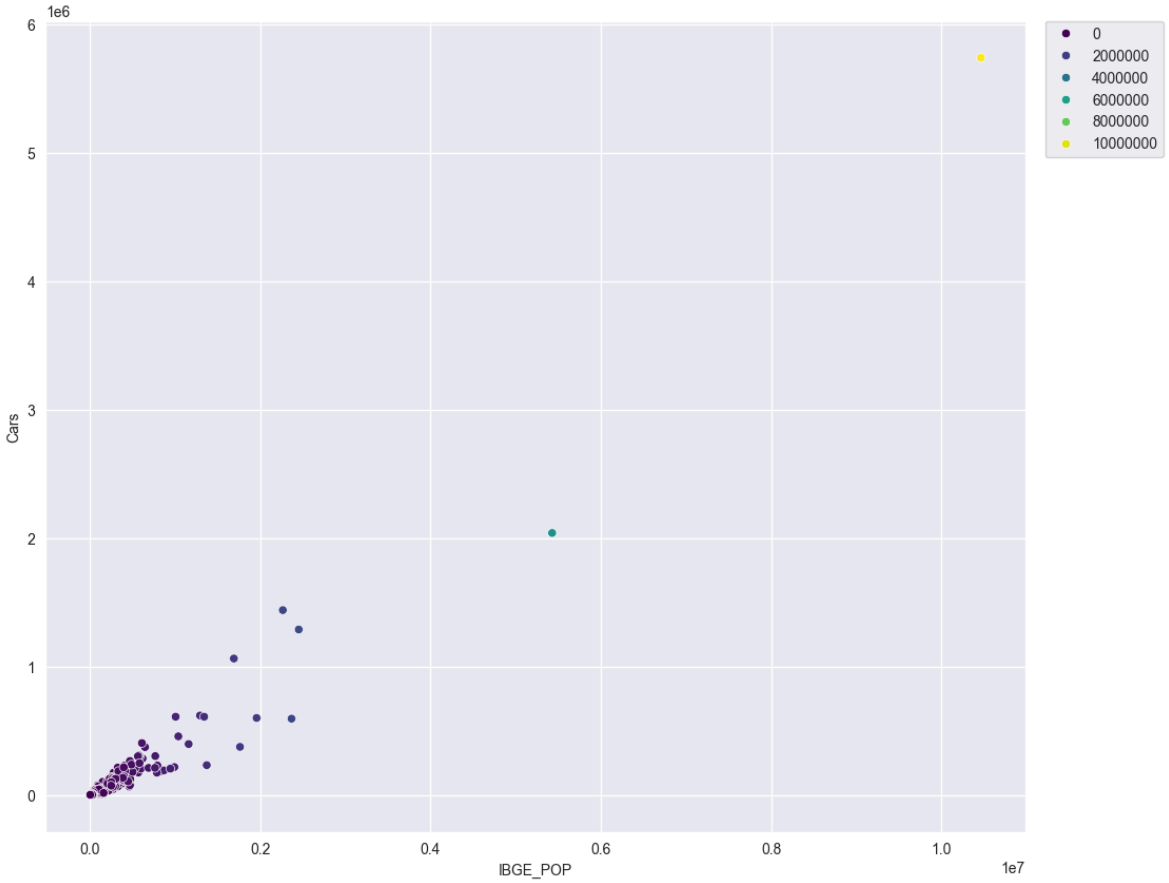
In [ ]:
```
plt.figure(figsize=(12, 10))
sns.scatterplot(data=BR_Cities_df, x='STATE', y='IBGE_POP', hue='CITY', palette=
```

Out[ ]:
```
<Axes: xlabel='STATE', ylabel='IBGE_POP'>
```



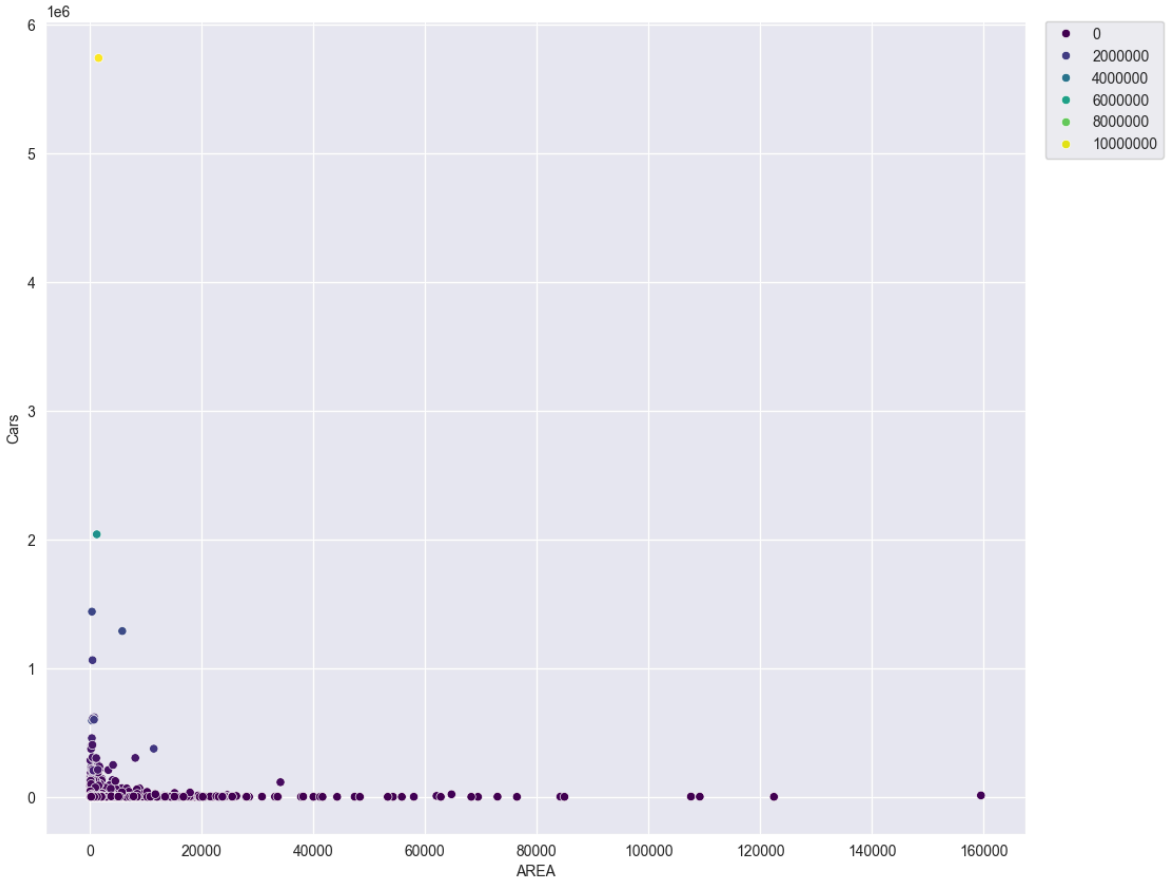
In [ ]:
```
plt.figure(figsize=(12, 10))
sns.scatterplot(data=BR_Cities_df, x='IBGE_POP', y='Cars', hue='IBGE_POP', palet
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x2677c8ddbe0>
```



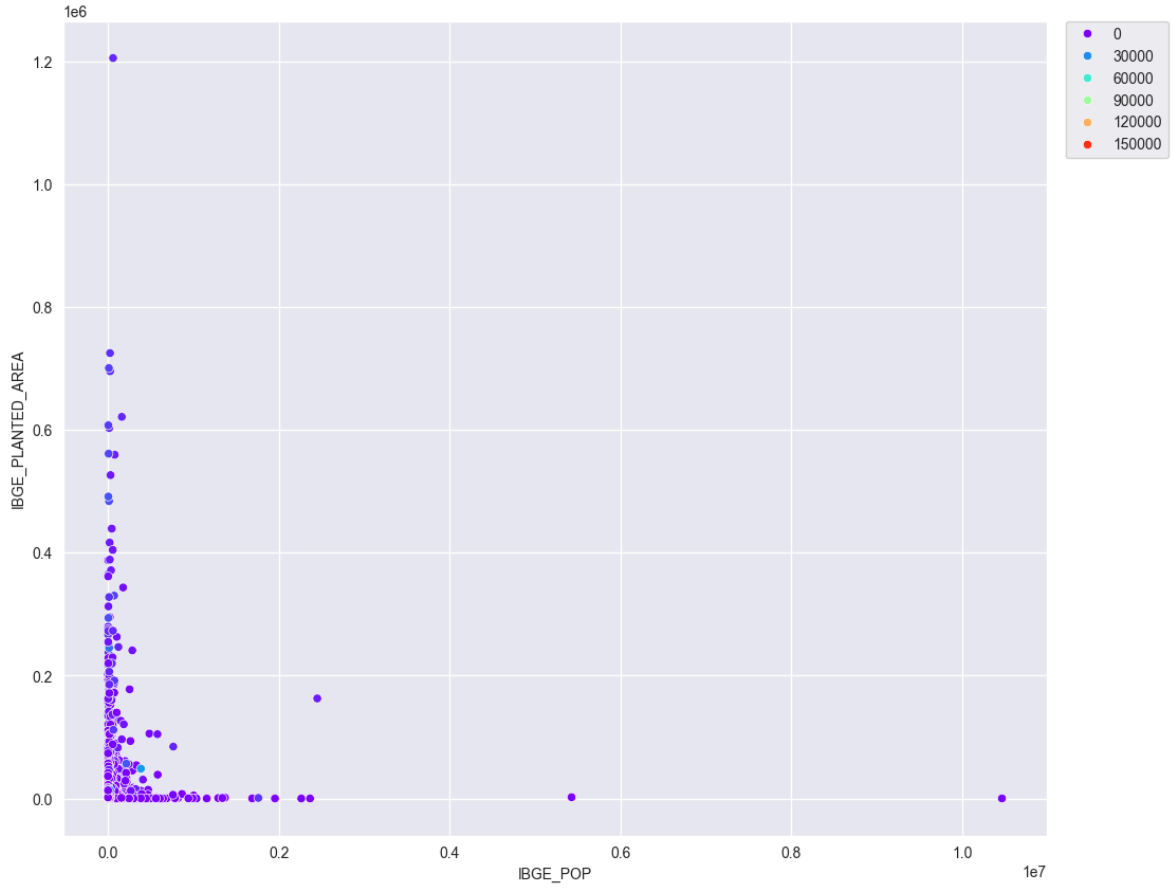
In [ ]:
```
plt.figure(figsize=(12, 10))
sns.scatterplot(data=BR_Cities_df, x='AREA', y='Cars', hue='IBGE_POP', palette='
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```

Out[ ]: <matplotlib.legend.Legend at 0x2677d786780>



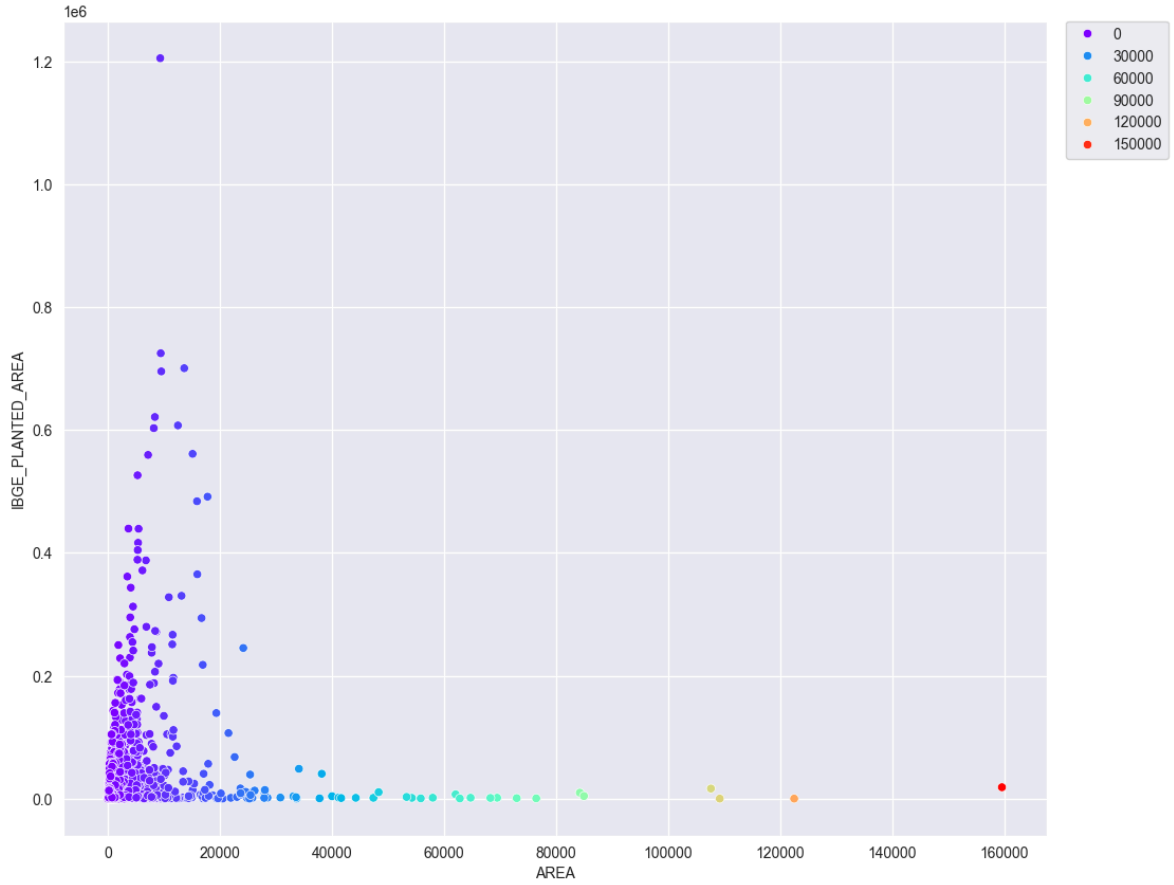
In [ ]:
```
plt.figure(figsize=(12, 10))
sns.scatterplot(data=BR_Cities_df, x='IBGE_POP', y='IBGE_PLANTED_AREA', hue='ARE
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x2677f5545c0>
```



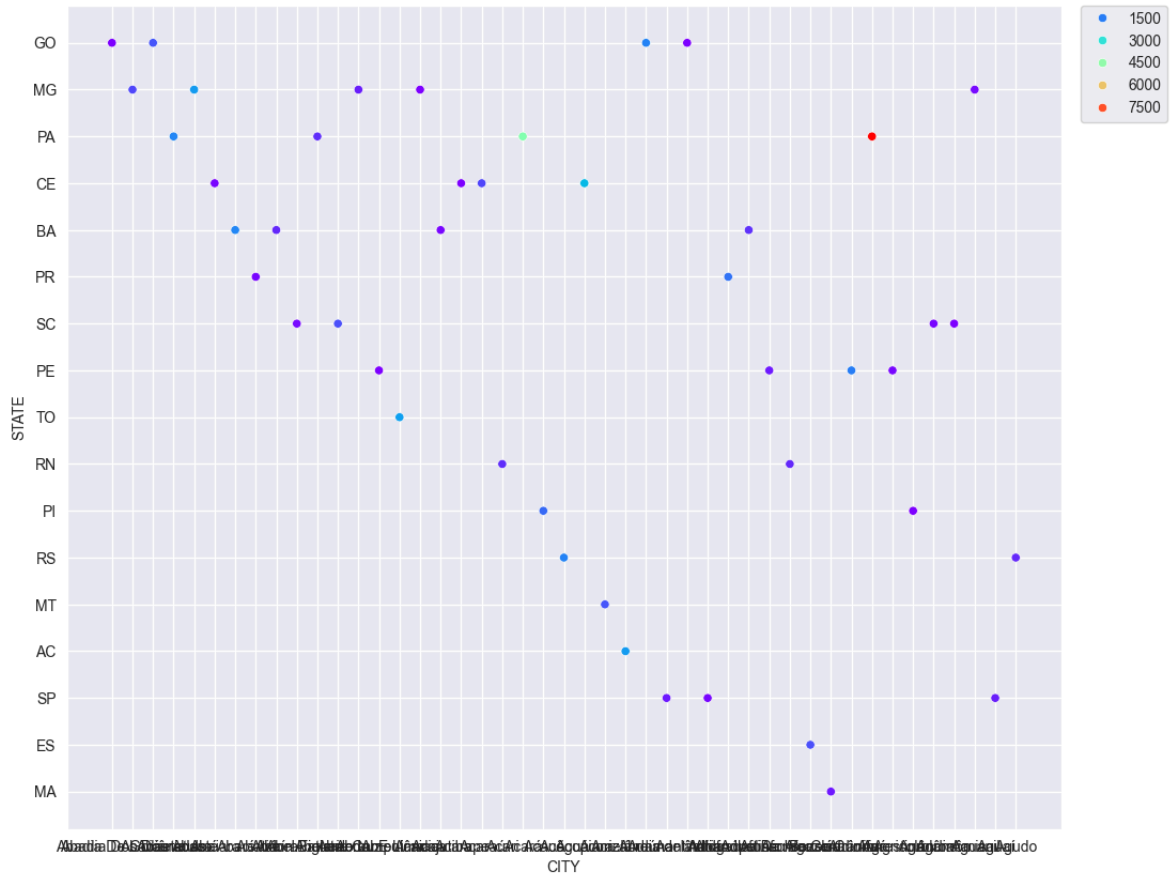
In [ ]:
```
plt.figure(figsize=(12, 10))
sns.scatterplot(data=BR_Cities_df, x='AREA', y='IBGE_PLANTED_AREA', hue='AREA',
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x2677d8c7a40>
```



In [ ]:
```
plt.figure(figsize=(12, 10))
sns.scatterplot(data=BR_Cities_df.head(45), x='CITY', y='STATE', hue='AREA', pal
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```

Out[ ]: <matplotlib.legend.Legend at 0x2677c96bc50>

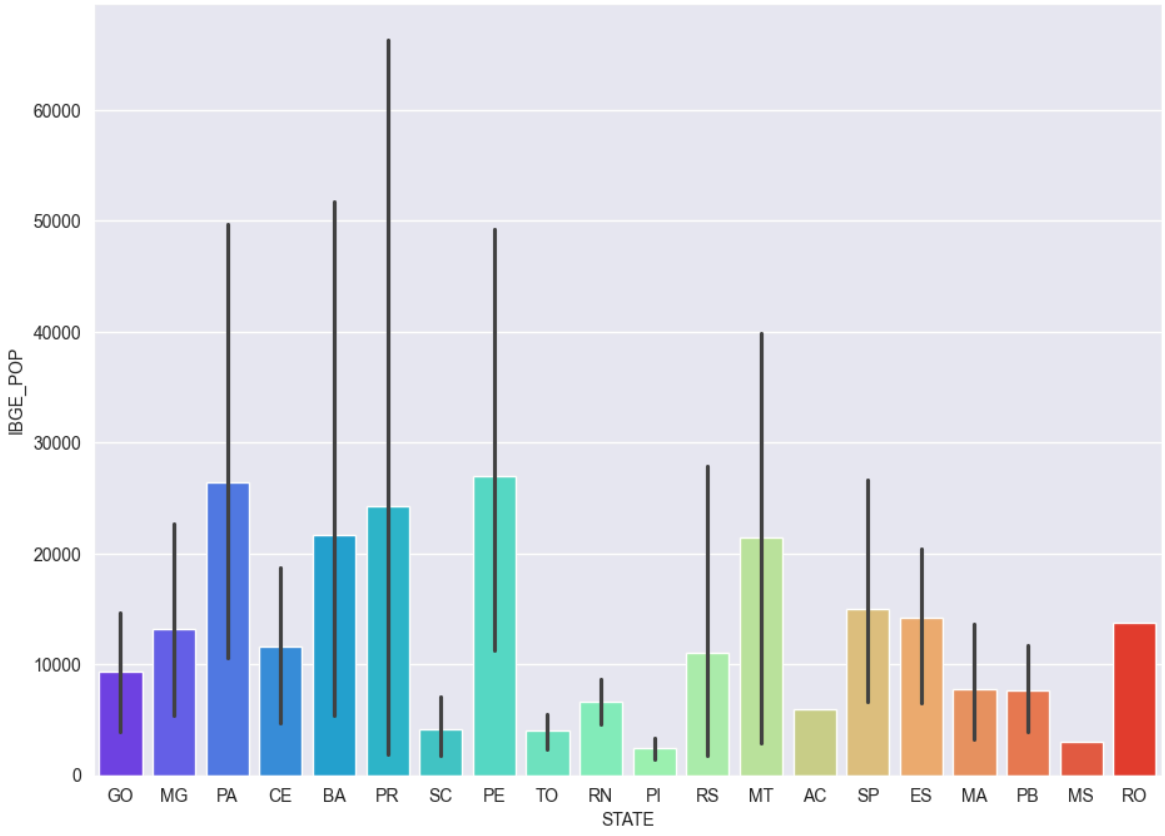


d) EXTRA: Some Bar Plots

In [ ]:
```
plt.figure(figsize=(11, 8))
sns.barplot(data=BR_Cities_df.head(100), y='IBGE_POP',x='STATE',palette='rainbow
```

```
C:\Users\Enzo\AppData\Local\Temp\ipykernel_33976\993632507.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(data=BR_Cities_df.head(100), y='IBGE_POP',x='STATE',palette='rainbo
w')
```

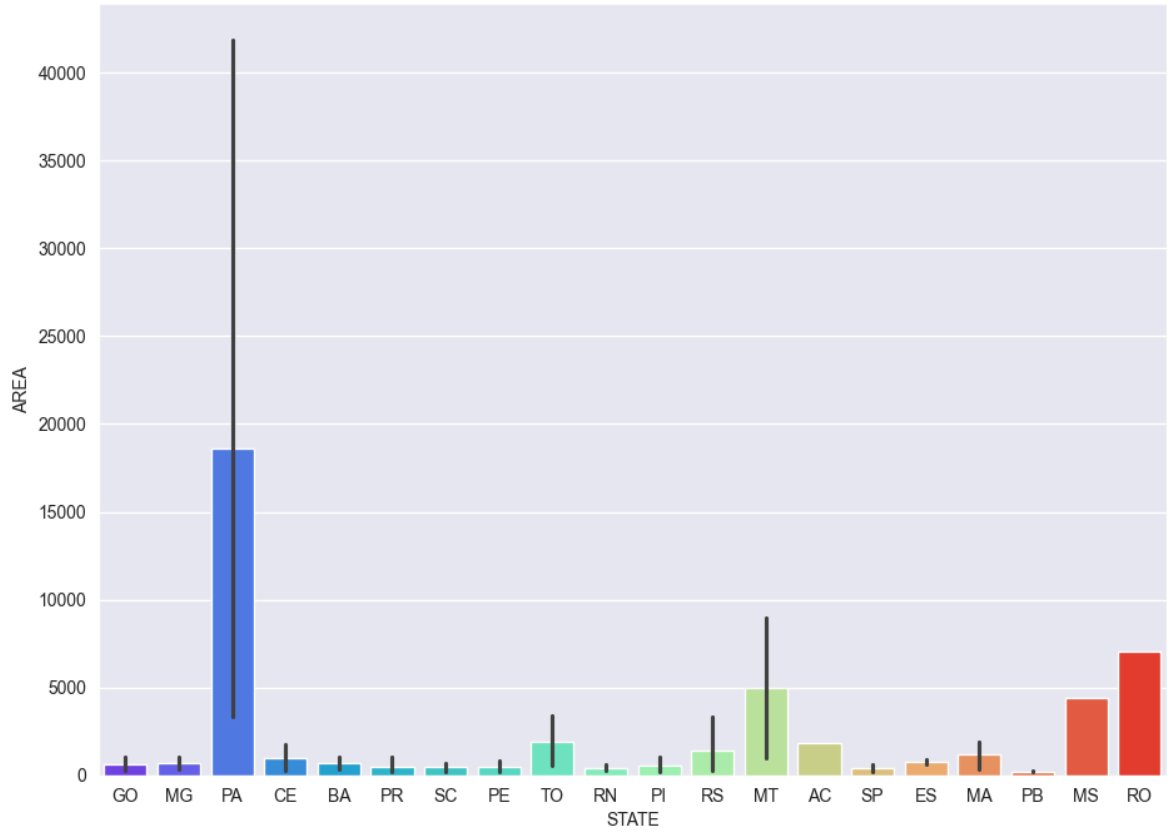
Out[ ]: <Axes: xlabel='STATE', ylabel='IBGE_POP'>



In [ ]:
```
plt.figure(figsize=(11, 8))
sns.barplot(data=BR_Cities_df.head(100), y='AREA',x='STATE',palette='rainbow')
```

```
C:\Users\Enzo\AppData\Local\Temp\ipykernel_33976\2573547415.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(data=BR_Cities_df.head(100), y='AREA',x='STATE',palette='rainbow')
```

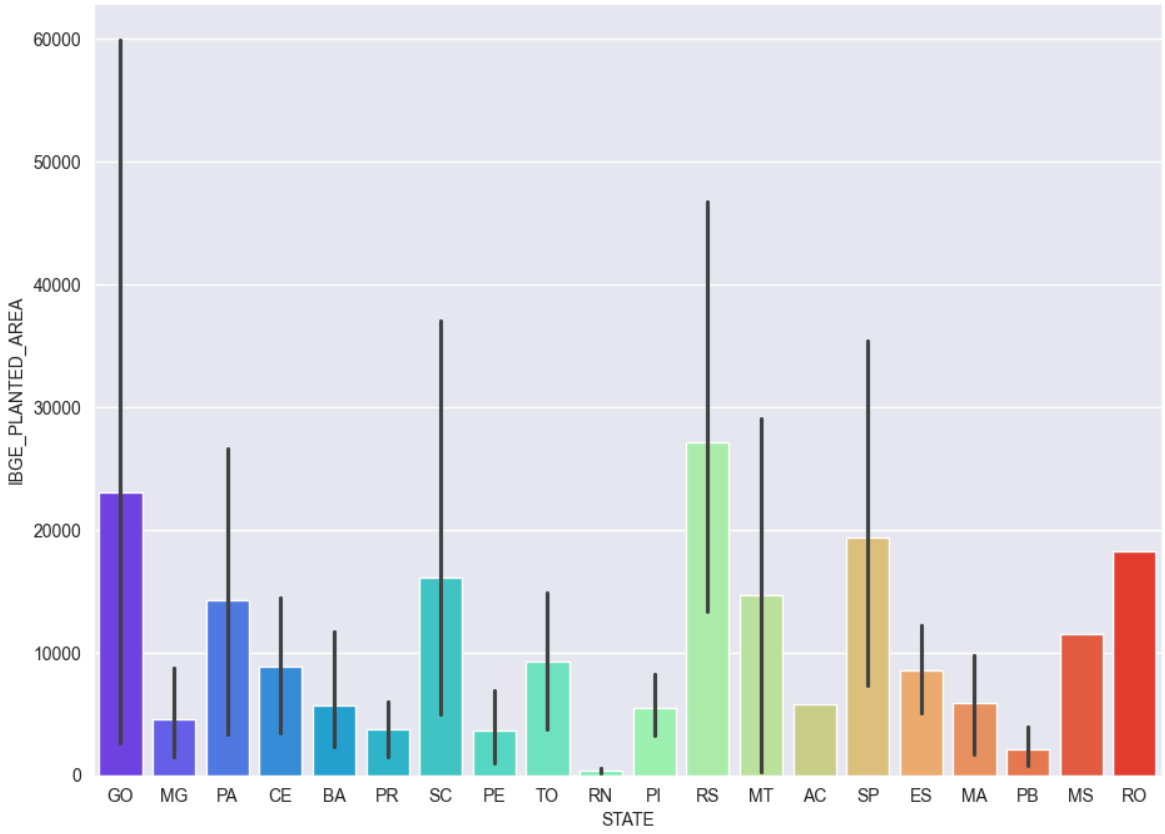
Out[ ]: <Axes: xlabel='STATE', ylabel='AREA'>



In [ ]:
```
plt.figure(figsize=(11, 8))
sns.barplot(data=BR_Cities_df.head(100), y='IBGE_PLANTED_AREA',x='STATE',palette
```

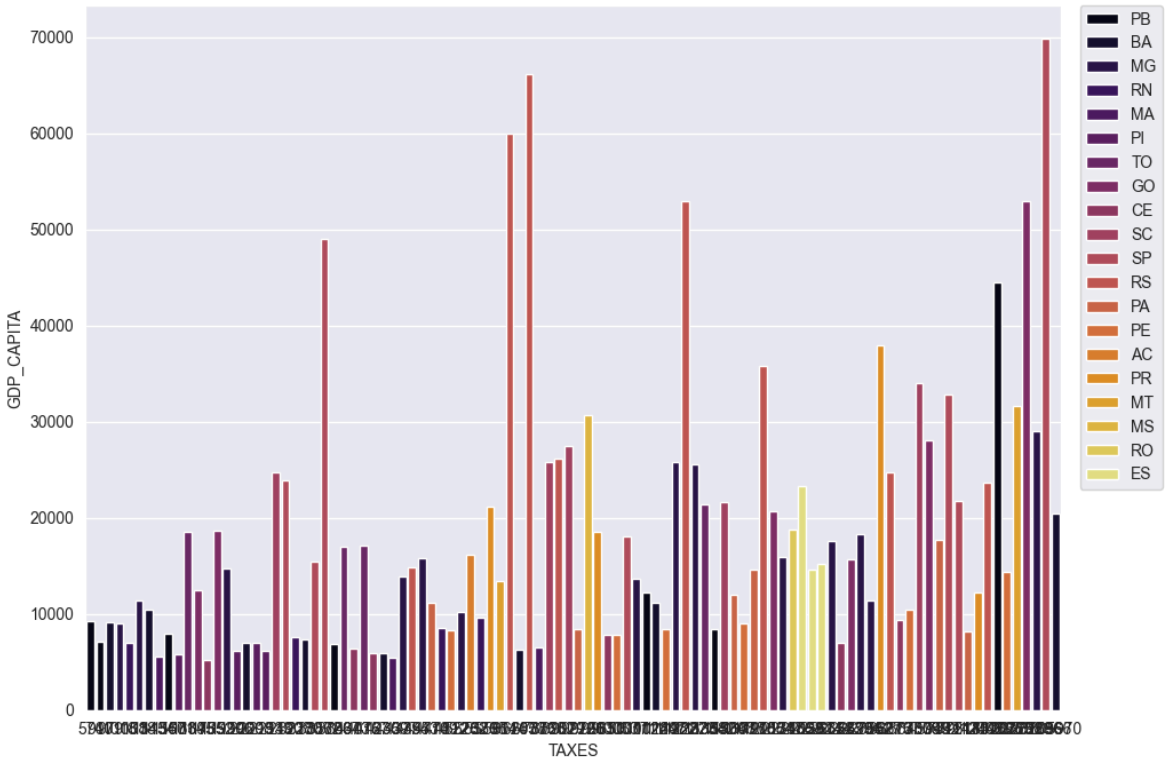
```
C:\Users\Enzo\AppData\Local\Temp\ipykernel_33976\2869131856.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(data=BR_Cities_df.head(100), y='IBGE_PLANTED_AREA',x='STATE',palett
e='rainbow')
```

Out[ ]:
```
<Axes: xlabel='STATE', ylabel='IBGE_PLANTED_AREA'>
```



In [ ]:
```
plt.figure(figsize=(11, 8))
sns.barplot(data=BR_Cities_df.head(100), y='GDP_CAPITA',x='TAXES', hue = 'STATE'
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```

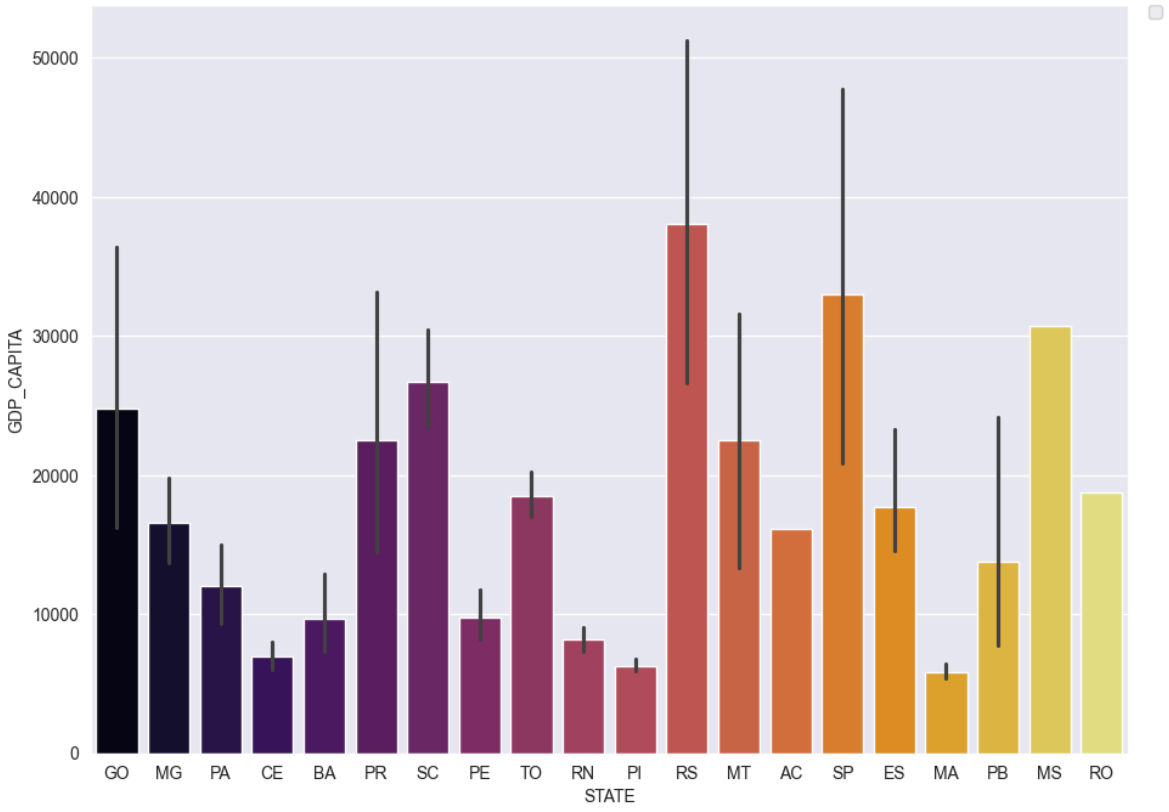
Out[ ]:
```
<matplotlib.legend.Legend at 0x2677f4437a0>
```



In [ ]:
```
plt.figure(figsize=(11, 8))
sns.barplot(data=BR_Cities_df.head(100), y='GDP_CAPITA',x='STATE', hue = 'STATE'
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```

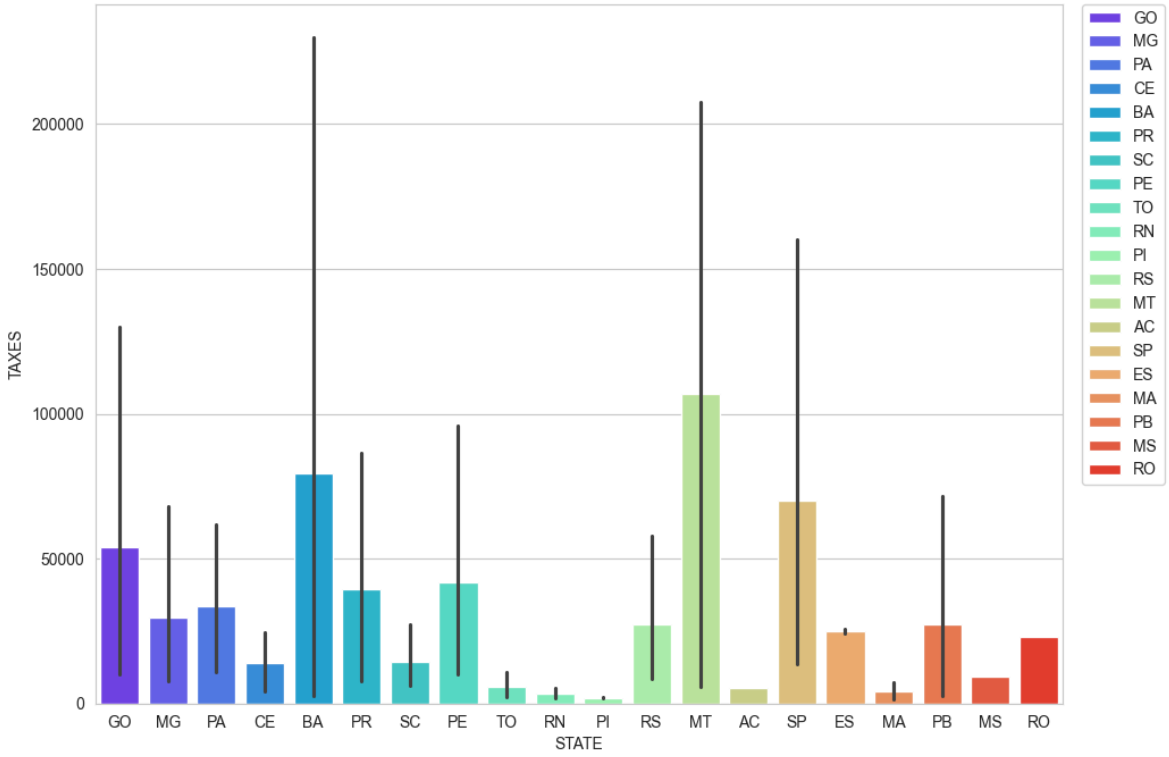
```
C:\Users\Enzo\AppData\Local\Temp\ipykernel_33976\3002157413.py:3: UserWarning: No
artists with labels found to put in legend.  Note that artists whose label start
with an underscore are ignored when legend() is called with no argument.
  plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```

Out[ ]: <matplotlib.legend.Legend at 0x2670257d580>



In [ ]:
```
plt.figure(figsize=(11, 8))
sns.set_style('whitegrid')
sns.barplot(data=BR_Cities_df.head(100), y='TAXES',x='STATE', hue = "STATE", pal
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```

Out[ ]: <matplotlib.legend.Legend at 0x26702695be0>



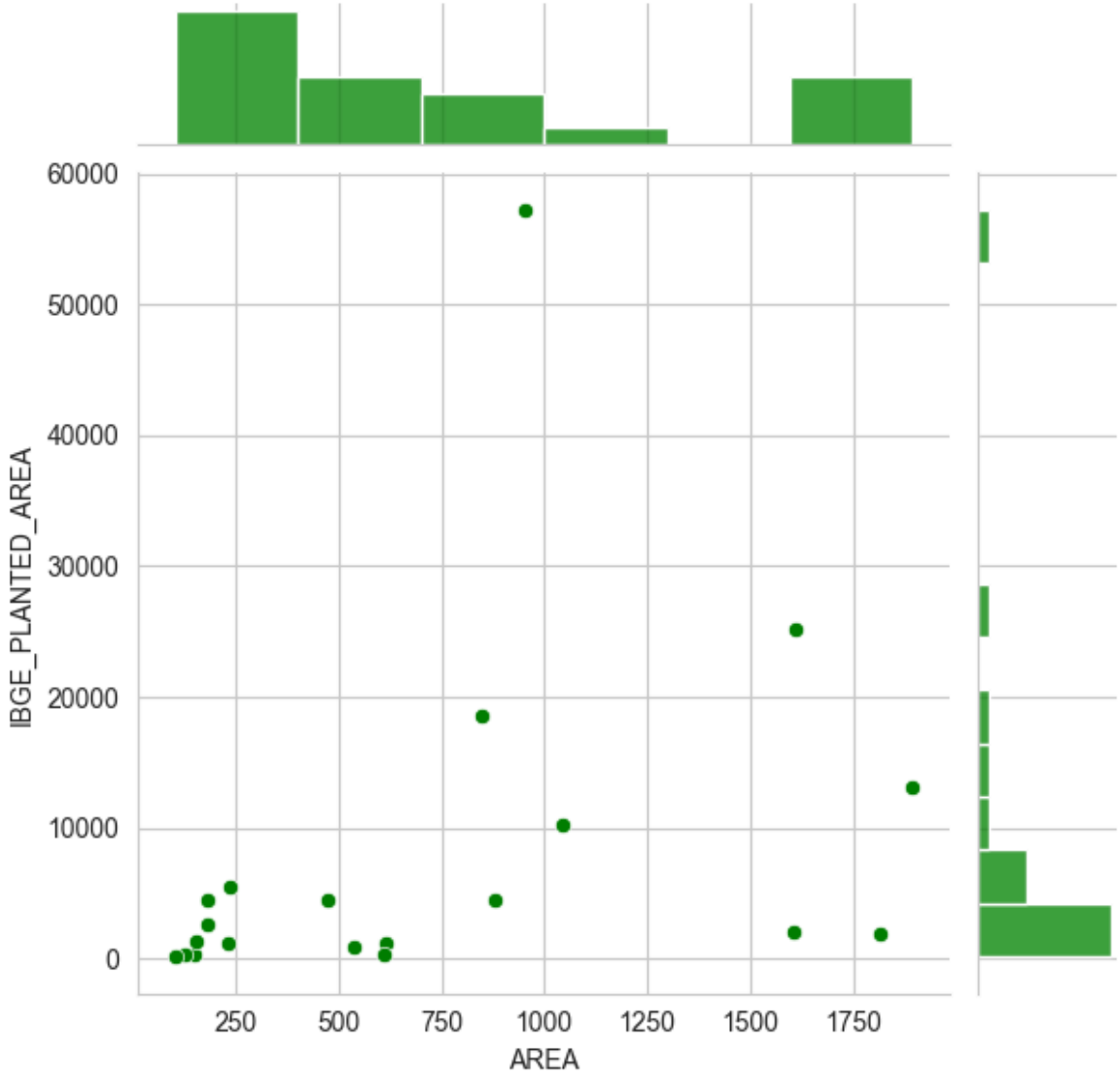
4) Advanced Analysis:

4.1) Calculate correlations between different variables (e.g., scores in different subjects).

In [ ]:
```
plt.figure(figsize=(12, 10))
sns.jointplot(data=BR_Cities_df.head(20), x='AREA', y='IBGE_PLANTED_AREA', kind=
```

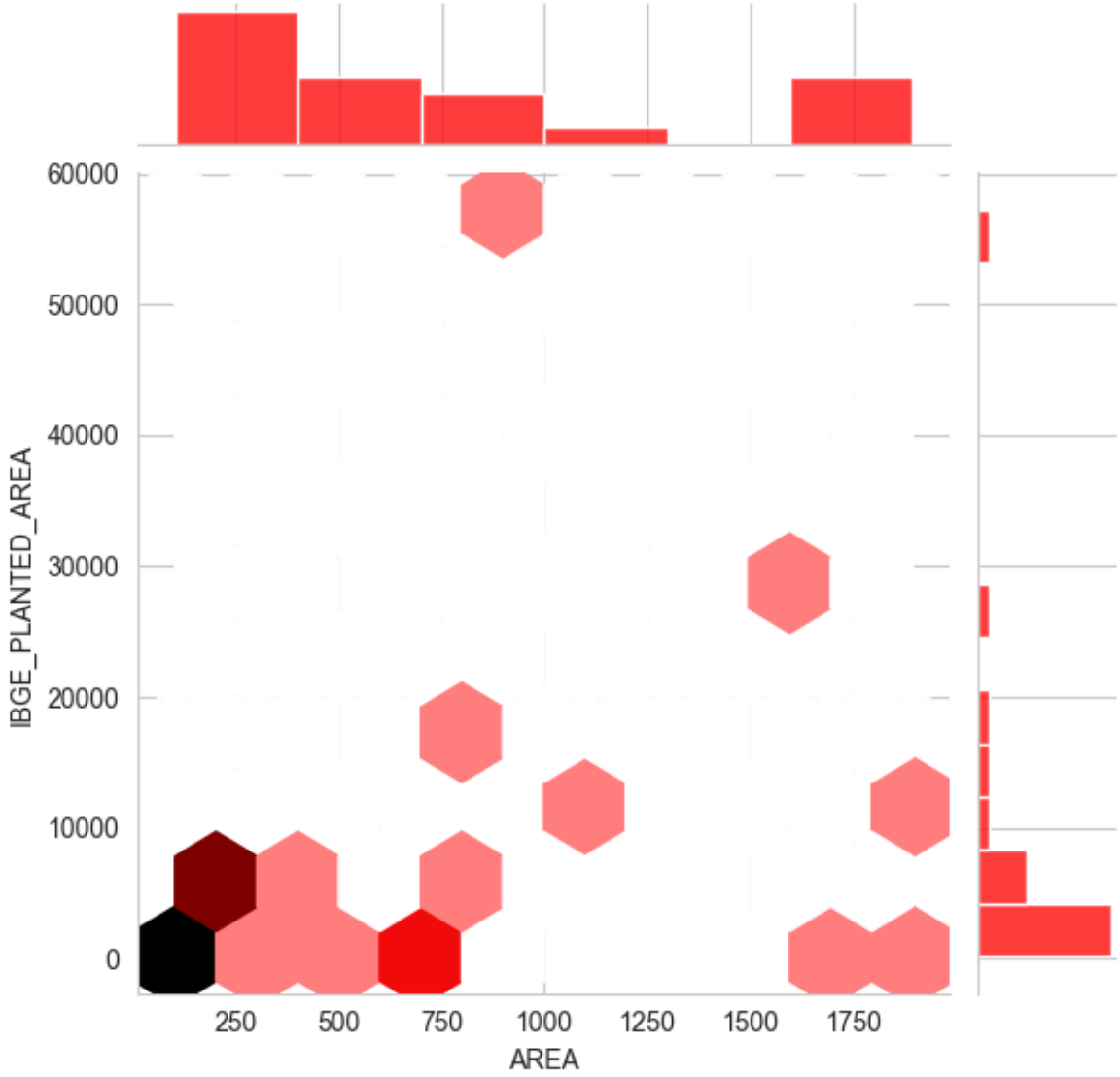
Out[ ]:
```
<seaborn.axisgrid.JointGrid at 0x267025ba750>
```

```
<Figure size 1200x1000 with 0 Axes>
```



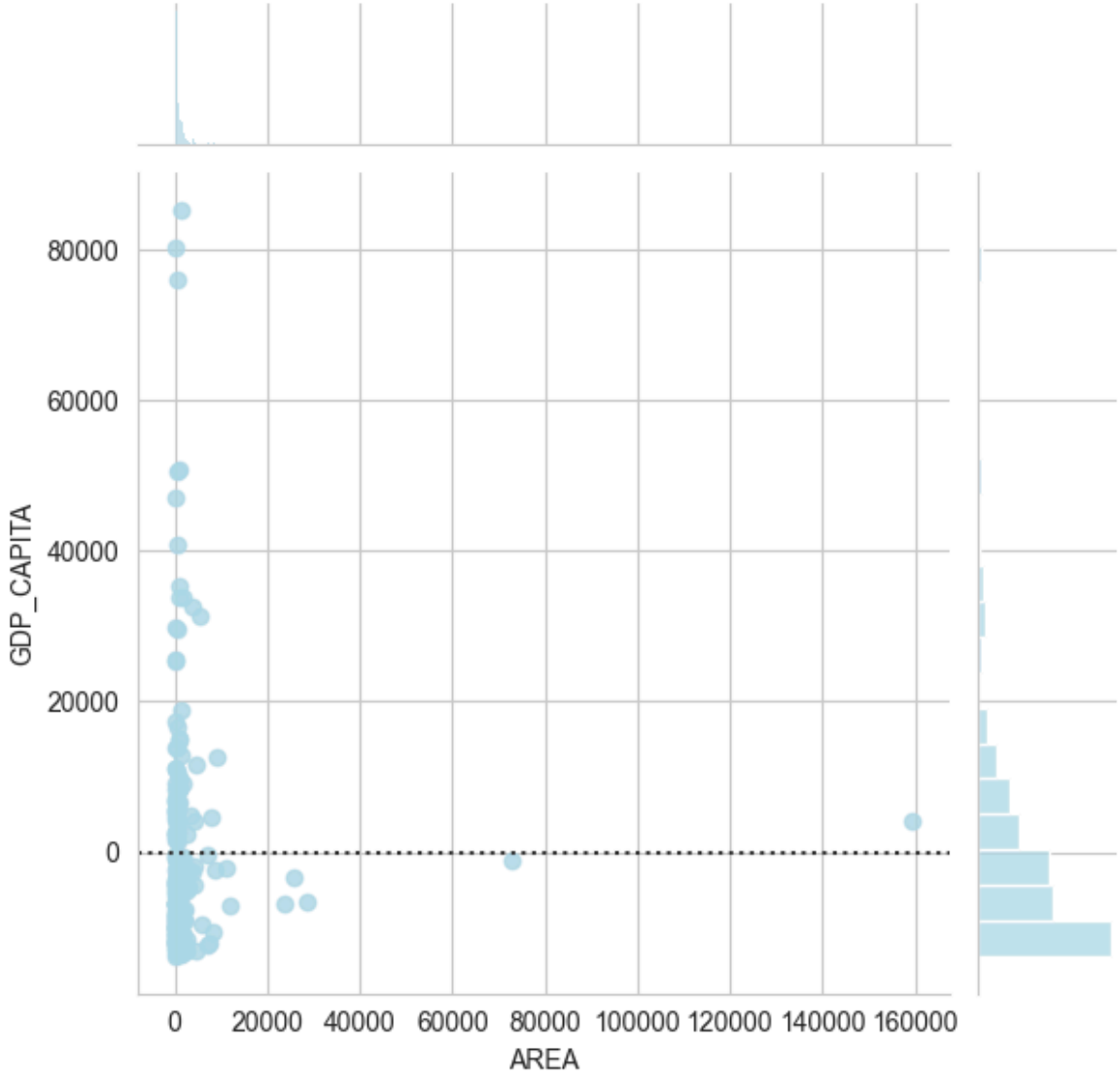
In [ ]:
```
plt.figure(figsize=(12, 10))
sns.jointplot(data=BR_Cities_df.head(20), x='AREA', y='IBGE_PLANTED_AREA', kind=
```

Out[ ]:
```
<seaborn.axisgrid.JointGrid at 0x26702bc3500>
<Figure size 1200x1000 with 0 Axes>
```



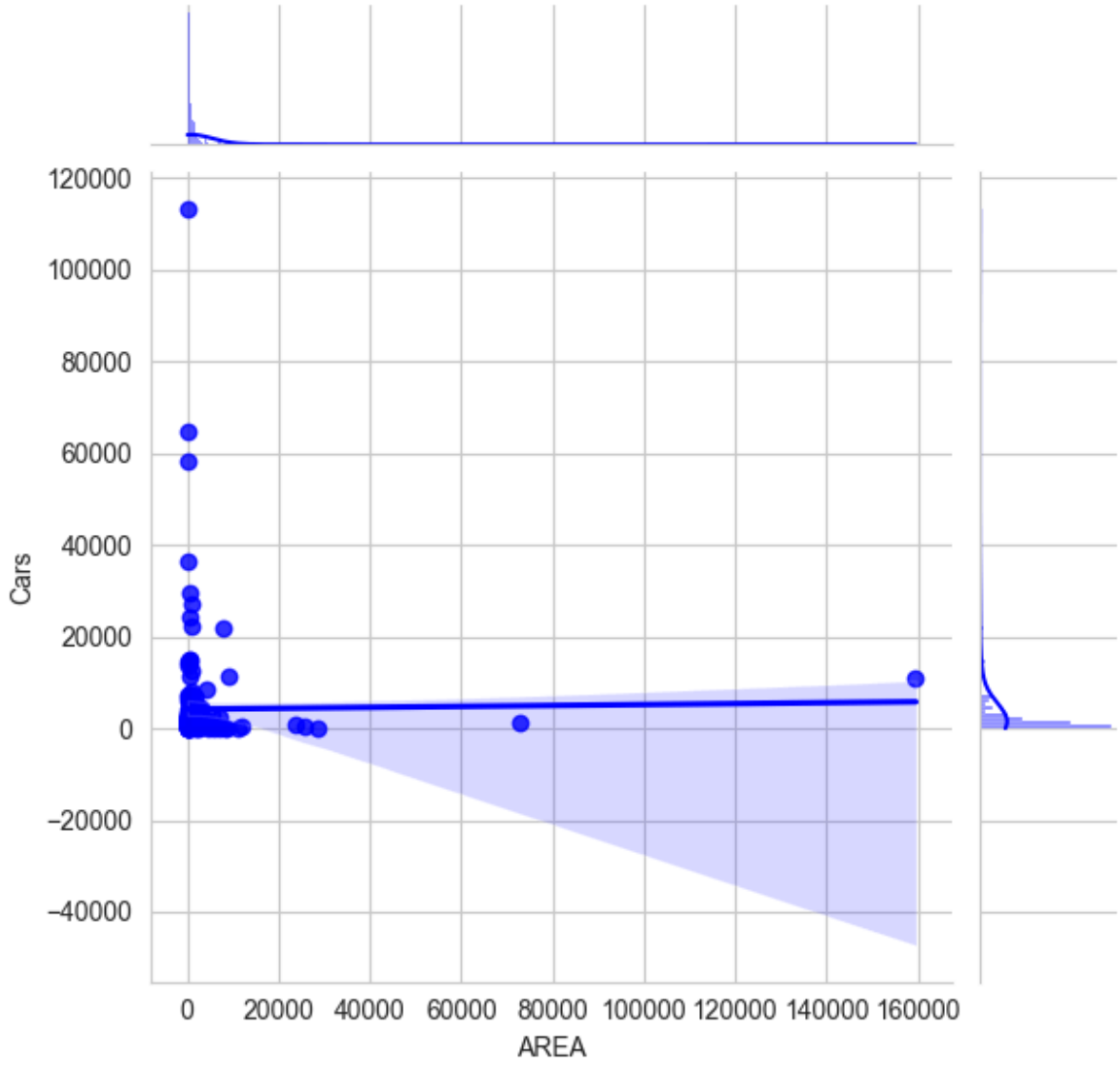
In [ ]:
```
plt.figure(figsize=(12, 10))
sns.jointplot(data=BR_Cities_df.head(203), x='AREA', y='GDP_CAPITA', kind='resid
```

Out[ ]:
```
<seaborn.axisgrid.JointGrid at 0x267044e2810>
<Figure size 1200x1000 with 0 Axes>
```



In [ ]:
```
plt.figure(figsize=(12, 10))
sns.jointplot(data=BR_Cities_df.head(203), x='AREA', y='Cars', kind='reg', color
```

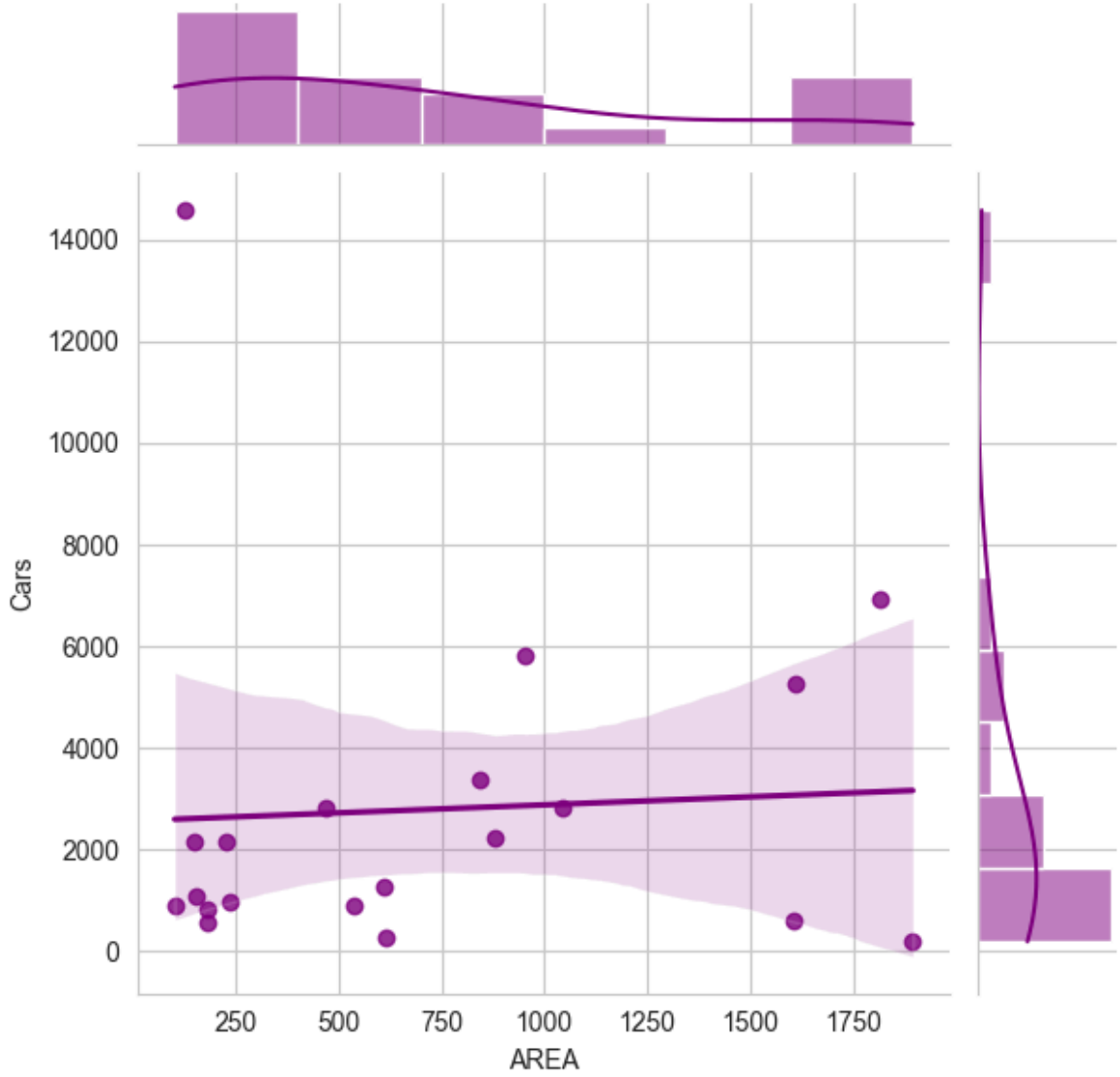
Out[ ]:
```
<seaborn.axisgrid.JointGrid at 0x267059bb2c0>
<Figure size 1200x1000 with 0 Axes>
```



In [ ]:
```
plt.figure(figsize=(12, 10))
sns.jointplot(data=BR_Cities_df.head(20), x='AREA', y='Cars', kind='reg', color=
```

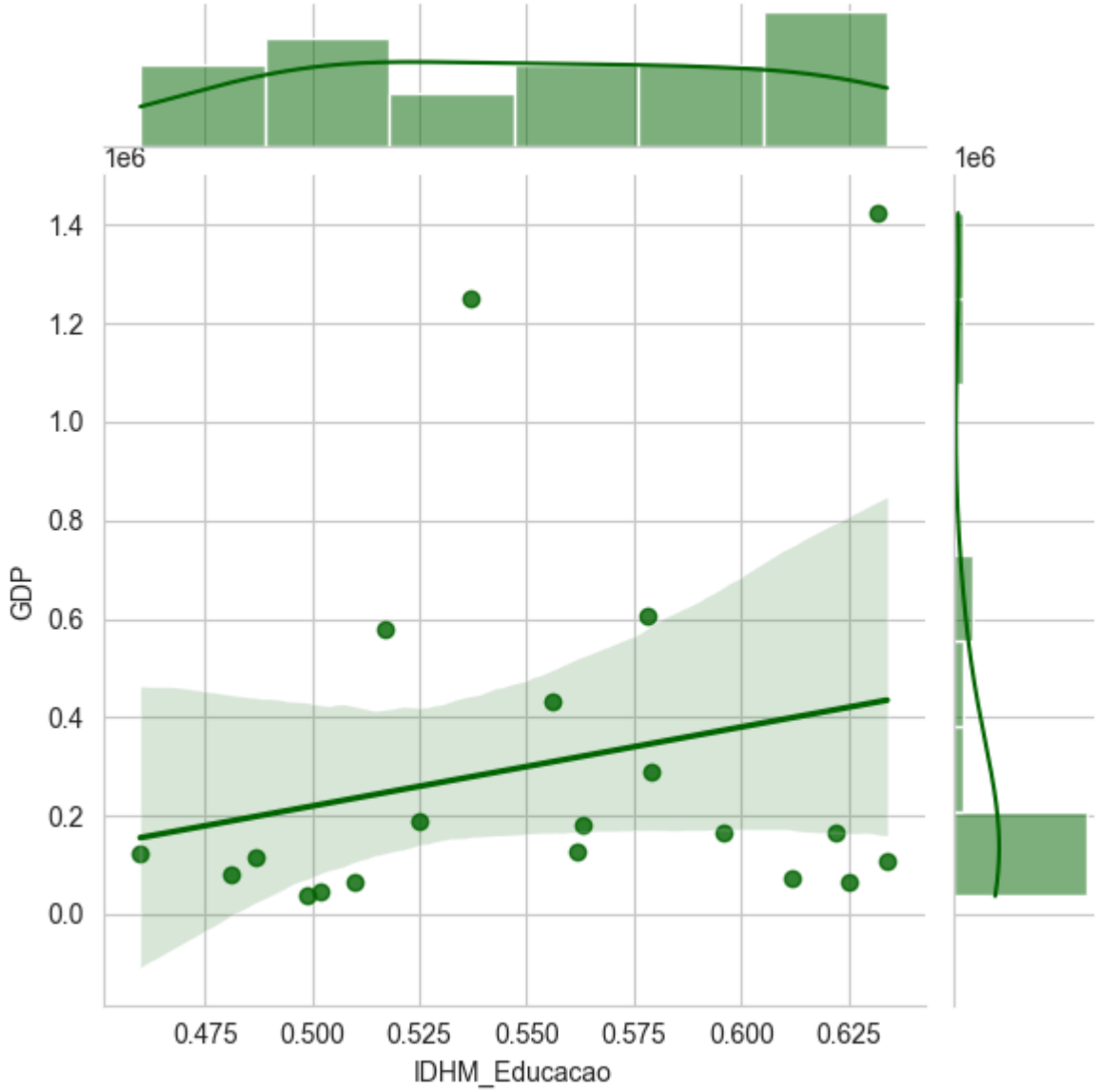
Out[ ]: <seaborn.axisgrid.JointGrid at 0x2677c29edb0>

<Figure size 1200x1000 with 0 Axes>



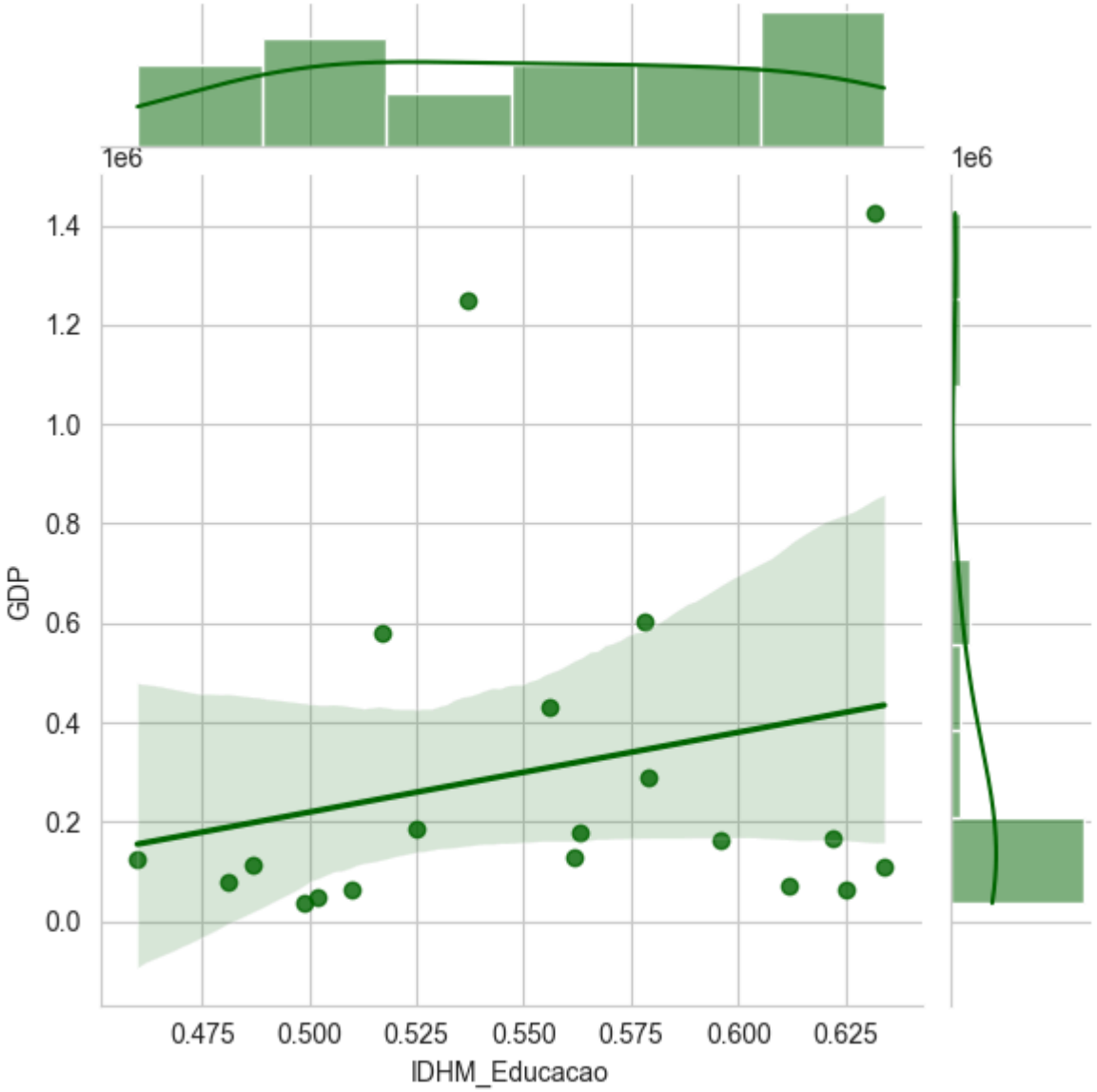
In [ ]:
```
plt.figure(figsize=(12, 10))
sns.jointplot(data=BR_Cities_df.head(20), x='IDHM_Educacao', y='GDP', kind='reg'
```

Out[ ]:
```
<seaborn.axisgrid.JointGrid at 0x26704be2930>
<Figure size 1200x1000 with 0 Axes>
```



In [ ]:
```
plt.figure(figsize=(12, 10))
sns.jointplot(data=BR_Cities_df.head(20), x='IDHM_Educacao', y='GDP', kind='reg'
```

Out[ ]:
```
<seaborn.axisgrid.JointGrid at 0x26704be3800>
<Figure size 1200x1000 with 0 Axes>
```



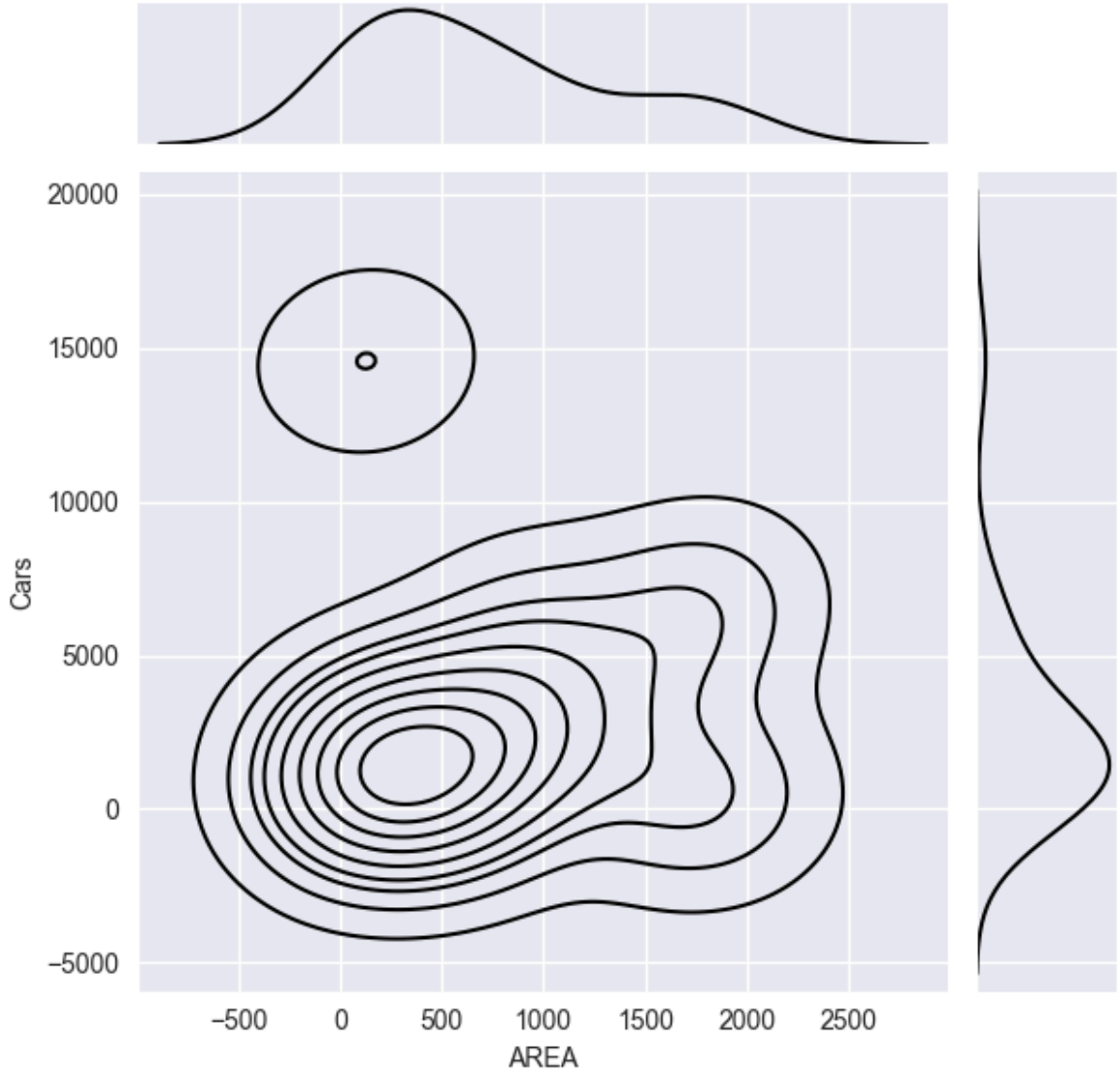
In [ ]:
```
BR_Cities_df.columns
```

Out[ ]:
```
Index(['CITY', 'STATE', 'CAPITAL', 'IBGE_RES_POP', 'IBGE_RES_POP_BRAS',
       'IBGE_RES_POP_ESTR', 'IBGE_DU', 'IBGE_DU_URBAN', 'IBGE_DU_RURAL',
       'IBGE_POP', 'IBGE_1', 'IBGE_1-4', 'IBGE_5-9', 'IBGE_10-14',
       'IBGE_15-59', 'IBGE_60+', 'IBGE_PLANTED_AREA', 'IBGE_CROP_PRODUCTION_$',
       'IDHM Ranking 2010', 'IDHM', 'IDHM_Renda', 'IDHM_Longevidade',
       'IDHM_Educacao', 'LONG', 'LAT', 'ALT', 'PAY_TV', 'FIXED_PHONES', 'AREA',
       'REGIAO_TUR', 'CATEGORIA_TUR', 'ESTIMATED_POP', 'RURAL_URBAN',
       'GVA_AGROPEC', 'GVA_INDUSTRY', 'GVA_SERVICES', 'GVA_PUBLIC',
       'GVA_TOTAL', 'TAXES', 'GDP', 'POP_GDP', 'GDP_CAPITA', 'GVA_MAIN',
       'MUN_EXPENDIT', 'COMP_TOT', 'COMP_A', 'COMP_B', 'COMP_C', 'COMP_D',
       'COMP_E', 'COMP_F', 'COMP_G', 'COMP_H', 'COMP_I', 'COMP_J', 'COMP_K',
       'COMP_L', 'COMP_M', 'COMP_N', 'COMP_O', 'COMP_P', 'COMP_Q', 'COMP_R',
       'COMP_S', 'COMP_T', 'COMP_U', 'HOTELS', 'BEDS', 'Pr_Agencies',
       'Pu_Agencies', 'Pr_Bank', 'Pu_Bank', 'Pr_Assets', 'Pu_Assets', 'Cars',
       'Motorcycles', 'Wheeled_tractor', 'UBER', 'MAC', 'WAL-MART',
       'POST_OFFICES'],
      dtype='object')
```

In [ ]:
```
plt.figure(figsize=(12, 10))
sns.set_style('darkgrid')
sns.jointplot(data=BR_Cities_df.head(20), x='AREA', y='Cars', kind='kde', color=
```

Out[ ]: <seaborn.axisgrid.JointGrid at 0x267045495e0>

<Figure size 1200x1000 with 0 Axes>



4.2) Create a heatmap using seaborn to visualize correlations.

In [ ]:
```
BR_Cities_df.columns
```

Out[ ]:
```
Index(['CITY', 'STATE', 'CAPITAL', 'IBGE_RES_POP', 'IBGE_RES_POP_BRAS',
       'IBGE_RES_POP_ESTR', 'IBGE_DU', 'IBGE_DU_URBAN', 'IBGE_DU_RURAL',
       'IBGE_POP', 'IBGE_1', 'IBGE_1-4', 'IBGE_5-9', 'IBGE_10-14',
       'IBGE_15-59', 'IBGE_60+', 'IBGE_PLANTED_AREA', 'IBGE_CROP_PRODUCTION_$',
       'IDHM Ranking 2010', 'IDHM', 'IDHM_Renda', 'IDHM_Longevidade',
       'IDHM_Educacao', 'LONG', 'LAT', 'ALT', 'PAY_TV', 'FIXED_PHONES', 'AREA',
       'REGIAO_TUR', 'CATEGORIA_TUR', 'ESTIMATED_POP', 'RURAL_URBAN',
       'GVA_AGROPEC', 'GVA_INDUSTRY', 'GVA_SERVICES', 'GVA_PUBLIC',
       'GVA_TOTAL', 'TAXES', 'GDP', 'POP_GDP', 'GDP_CAPITA', 'GVA_MAIN',
       'MUN_EXPENDIT', 'COMP_TOT', 'COMP_A', 'COMP_B', 'COMP_C', 'COMP_D',
       'COMP_E', 'COMP_F', 'COMP_G', 'COMP_H', 'COMP_I', 'COMP_J', 'COMP_K',
       'COMP_L', 'COMP_M', 'COMP_N', 'COMP_O', 'COMP_P', 'COMP_Q', 'COMP_R',
       'COMP_S', 'COMP_T', 'COMP_U', 'HOTELS', 'BEDS', 'Pr_Agencies',
       'Pu_Agencies', 'Pr_Bank', 'Pu_Bank', 'Pr_Assets', 'Pu_Assets', 'Cars',
       'Motorcycles', 'Wheeled_tractor', 'UBER', 'MAC', 'WAL-MART',
       'POST_OFFICES'],
      dtype='object')
```

In [ ]:
```
BR_Cities_df['STATE'].value_counts().head(5)
```

Out[ ]:
```
STATE
MG    853
SP    646
RS    499
BA    418
PR    400
Name: count, dtype: int64
```

In [ ]:
```
MgCitiesDF = BR_Cities_df[BR_Cities_df['STATE'] == 'MG']
MgCitiesDF
```

Out[ ]:

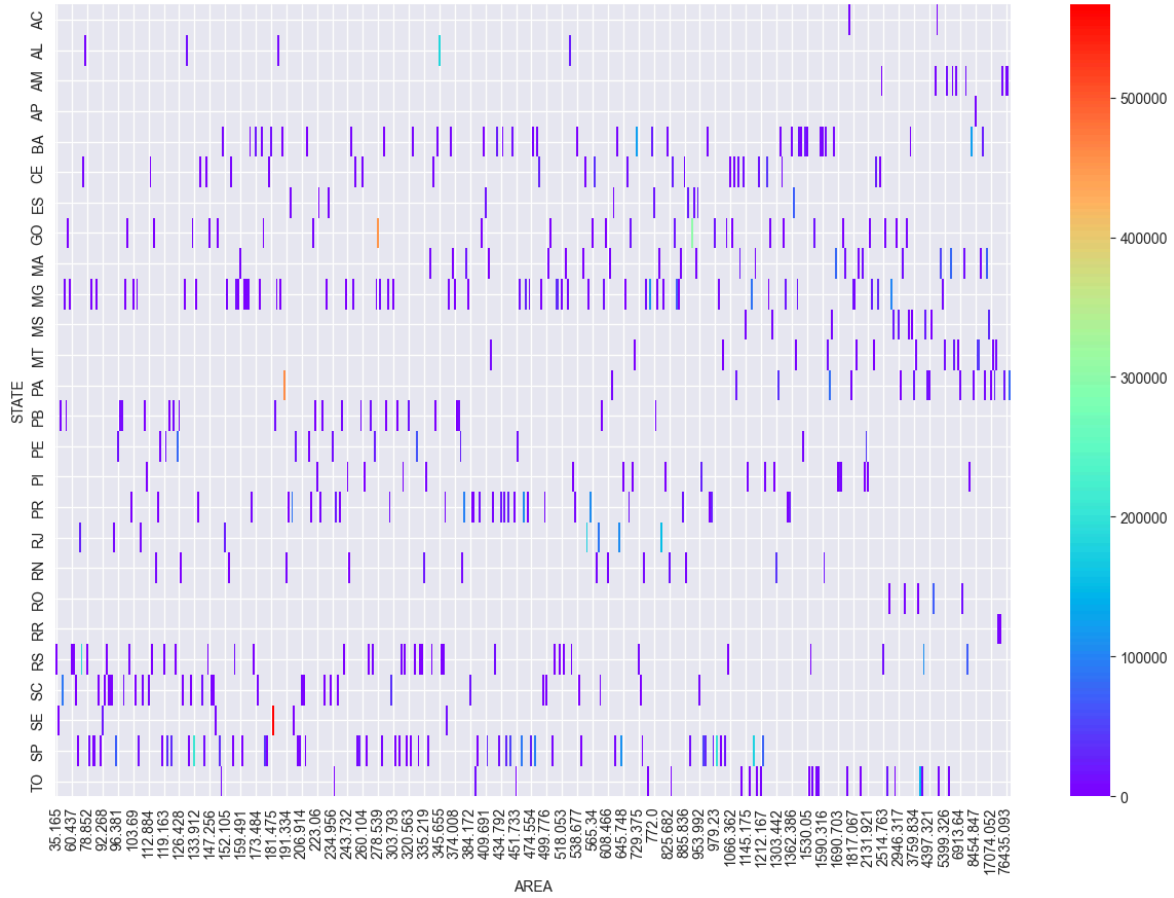
	CITY	STATE	CAPITAL	IBGE_RES_POP	IBGE_RES_POP_BRAS	IBGE_RES_POP_E
1	Abadia Dos Dourados	MG	0	6704	6704	
4	Abaeté	MG	0	22690	22690	
12	Abre Campo	MG	0	13311	13294	
15	Acaiaca	MG	0	3920	3920	
42	Aguanil	MG	0	4054	4054	
...	...	...	...	...	...	
5522	Wenceslau Braz	MG	0	2553	2553	
5543	Água Boa	MG	0	15195	15195	
5548	Água Comprida	MG	0	2025	2025	
5564	Águas Formosas	MG	0	18479	18474	
5568	Águas Vermelhas	MG	0	12722	12722	

853 rows × 81 columns

In [ ]:
```
matrixMgCities = MgCitiesDF[['Motorcycles', 'Cars','AREA','IBGE_PLANTED_AREA','U
```

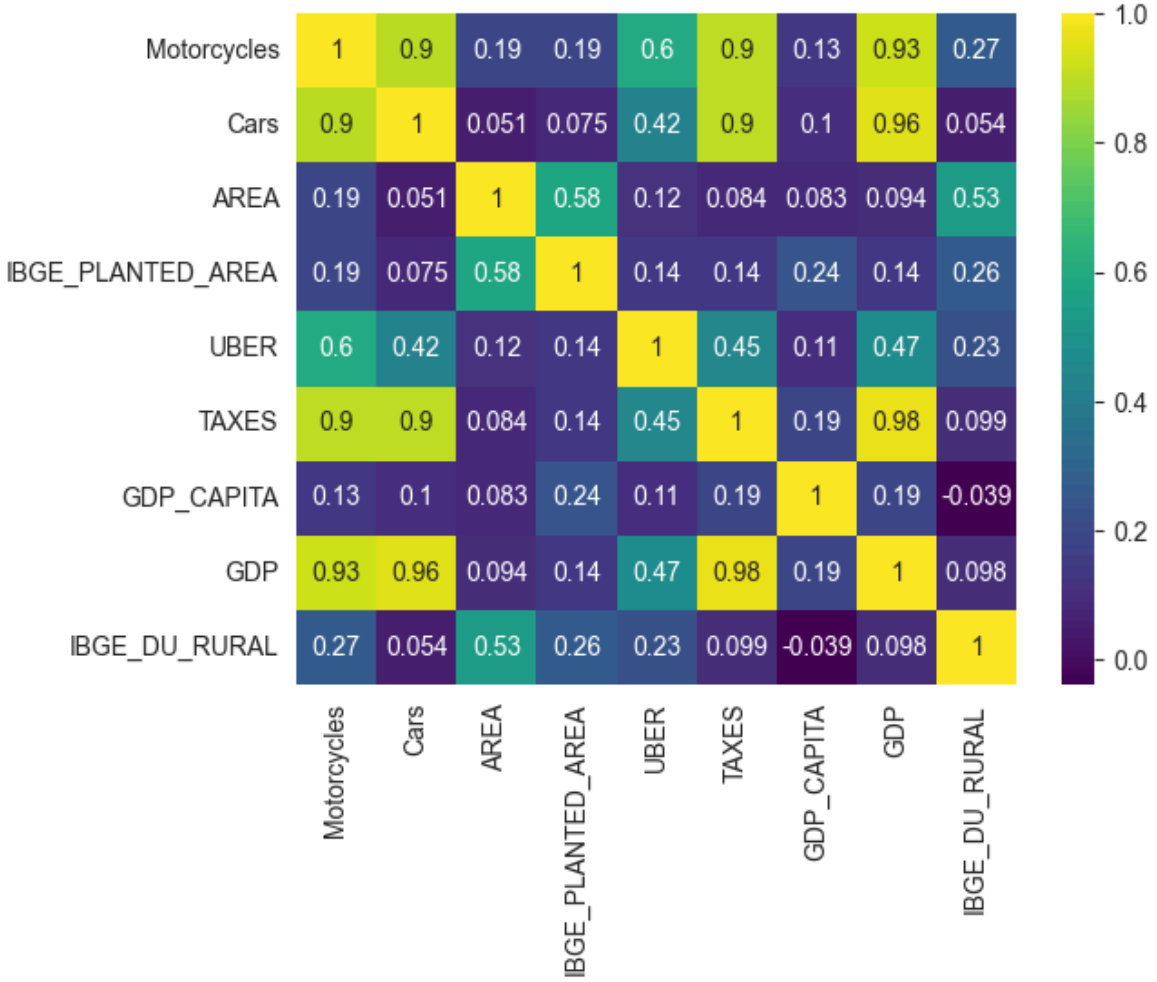
In [ ]:
```
plt.figure(figsize=(15, 10))
pvflights = BR_Cities_df.head(500).pivot_table(values='IBGE_POP',index='STATE',c
sns.heatmap(pvflights,cmap='rainbow')
```

Out[ ]:
```
<Axes: xlabel='AREA', ylabel='STATE'>
```



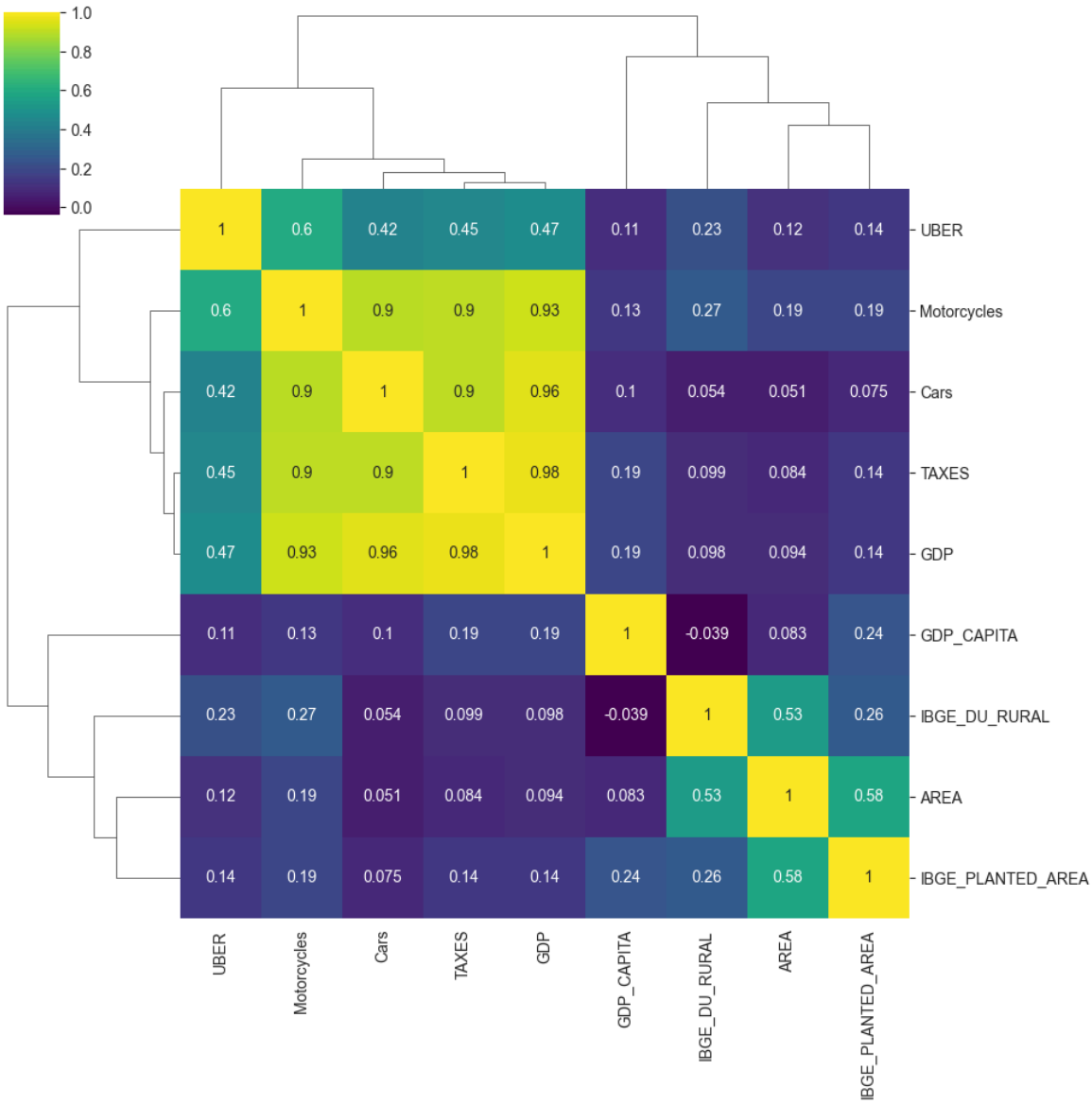
In [ ]:
```
sns.heatmap(matrixMgCities.corr(), annot=True, cmap='viridis')
```

Out[ ]: <Axes: >



In [ ]:
```
sns.clustermap(matrixMgCities.corr(), annot=True, cmap='viridis')
```

Out[ ]:
```
<seaborn.matrix.ClusterGrid at 0x2670aa06a20>
```



Interactive graphs

In [ ]:
```
import sys
import cufflinks as cf
cf.go_offline()
```

In [ ]:
```
BR_Cities_df.iplot(kind='scatter',mode='markers', x='STATE', y='IBGE_POP', xTitl
```

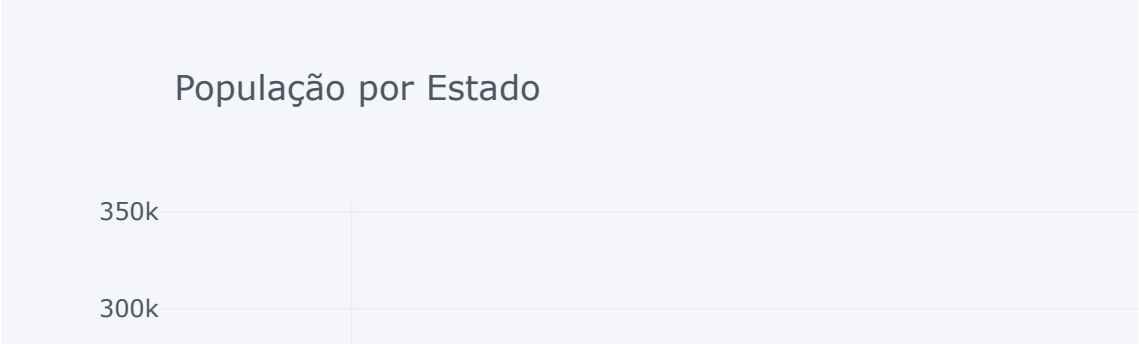
In [ ]:
```
BR_Cities_df.head(50).scatter_matrix()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[98], line 1
----> 1 BR_Cities_df.scatter_matrix()

File c:\Users\Enzo\AppData\Local\Programs\Python\Python312\Lib\site-packages\cuff
links\plotlytools.py:1280, in _scatter_matrix(self, theme, bins, color, size, asF
igure, **iplot_kwargs)
   1259 def _scatter_matrix(self,theme=None,bins=10,color='grey',size=2, asFigure
=False, **iplot_kwargs):
   1260         """
   1261         Displays a matrix with scatter plot for each pair of
   1262         Series in the DataFrame.
   (...)
   1278                         Keyword arguments to pass through to `iplot`
   1279         """
-> 1280         sm=tools.scatter_matrix(self,theme=theme,bins=bins,color=color,si
ze=size)
   1281         if asFigure:
   1282                 return sm

File c:\Users\Enzo\AppData\Local\Programs\Python\Python312\Lib\site-packages\cuff
links\tools.py:1024, in scatter_matrix(df, theme, bins, color, size)
   1022 layout['xaxis'].update(showgrid=False)
   1023 layout['yaxis'].update(showgrid=False)
-> 1024 sm=subplots(figs,shape=(len(df.columns),len(df.columns)),shared_xaxes=Fal
se,shared_yaxes=False,
   1025                                 horizontal_spacing=.05,vertical_spacing
=.07,base_layout=layout)
   1026 sm['layout'].update(bargap=.02,showlegend=False)
   1027 return sm

File c:\Users\Enzo\AppData\Local\Programs\Python\Python312\Lib\site-packages\cuff
links\tools.py:768, in subplots(figures, shape, shared_xaxes, shared_yaxes, start
_cell, theme, base_layout, **kwargs)
    766                 cols=2
    767                 rows=len(figures)//2+len(figures)%2
--> 768 sp,grid_ref=get_subplots(rows=rows,cols=cols,
    769                                         shared_xaxes=shared_xaxes, shared_ya
xes=shared_yaxes,
    770                                         start_cell=start_cell, theme=theme,b
ase_layout=base_layout,
    771                                         **kwargs)
    772 list_ref=(col for row in grid_ref for col in row)
    773 for i in range(len(figures)):

File c:\Users\Enzo\AppData\Local\Programs\Python\Python312\Lib\site-packages\cuff
links\tools.py:900, in get_subplots(rows, cols, shared_xaxes, shared_yaxes, start
_cell, theme, base_layout, **kwargs)
    897         theme = auth.get_config_file()['theme']
    899 layout= base_layout if base_layout else getLayout(theme,**check_kwargs(kw
args,__LAYOUT_AXIS))
--> 900 sp=make_subplots(rows=rows,cols=cols,shared_xaxes=shared_xaxes,
    901
shared_yaxes=shared_yaxes,print_grid=False,
    902
start_cell=start_cell,**kwargs)
    903 sp, grid_ref = sp.to_dict(), sp._grid_ref
    905 for k,v in list(layout.items()):
```

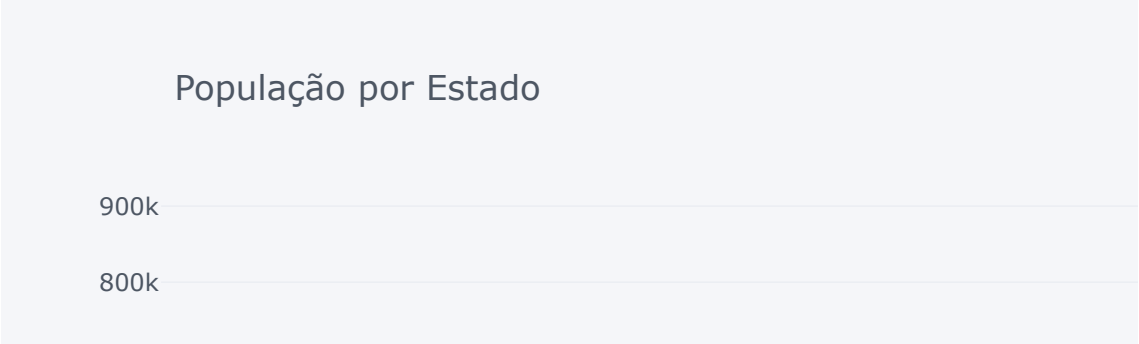
```
File c:\Users\Enzo\AppData\Local\Programs\Python\Python312\Lib\site-packages\plotly\subplots.py:304, in make_subplots(rows, cols, shared_xaxes, shared_yaxes, start_cell, print_grid, horizontal_spacing, vertical_spacing, subplot_titles, column_widths, row_heights, specs, insets, column_titles, row_titles, x_title, y_title, figure, **kwargs)
      6 def make_subplots(
      7     rows=1,
      8     cols=1,
   (...)
     25     **kwargs,
     26 ) -> go.Figure:
     27     """
     28     Return an instance of plotly.graph_objs.Figure with predefined subplots
     29     configured in 'layout'.
   (...)
    301     Figure(...)
    302     """
--> 304     return _sub.make_subplots(
    305         rows,
    306         cols,
    307         shared_xaxes,
    308         shared_yaxes,
    309         start_cell,
    310         print_grid,
    311         horizontal_spacing,
    312         vertical_spacing,
    313         subplot_titles,
    314         column_widths,
    315         row_heights,
    316         specs,
    317         insets,
    318         column_titles,
    319         row_titles,
    320         x_title,
    321         y_title,
    322         figure,
    323         **kwargs,
    324     )

File c:\Users\Enzo\AppData\Local\Programs\Python\Python312\Lib\site-packages\plotly\_subplots.py:555, in make_subplots(rows, cols, shared_xaxes, shared_yaxes, start_cell, print_grid, horizontal_spacing, vertical_spacing, subplot_titles, column_widths, row_heights, specs, insets, column_titles, row_titles, x_title, y_title, figure, **kwargs)
    553         horizontal_spacing = 0.2 / cols
    554 # check horizontal_spacing can be satisfied:
--> 555 _check_hv_spacing(cols, horizontal_spacing, "Horizontal", "cols", "columns")
    557 # ### vertical_spacing ###
    558 if vertical_spacing is None:

File c:\Users\Enzo\AppData\Local\Programs\Python\Python312\Lib\site-packages\plotly\_subplots.py:537, in make_subplots.<locals>._check_hv_spacing(dimsize, spacing, name, dimvarname, dimname)
    535         max_spacing = 1.0 / float(dimsize - 1)
    536         if spacing > max_spacing:
--> 537             raise ValueError(
    538                 """{name} spacing cannot be greater than (1 / ({dimvarname} - 1)) = {max_spacing:f}.
```

```
    539 The resulting plot would have {dimsize} {dimname} ({dimvarname}={dimsiz
e}).""".format(
    540                     dimvarname=dimvarname,
    541                     name=name,
    542                     dimname=dimname,
    543                     max_spacing=max_spacing,
    544                     dimsize=dimsize,
    545                 )
    546             )

ValueError: Horizontal spacing cannot be greater than (1 / (cols - 1)) = 0.01250
0.
The resulting plot would have 81 columns (cols=81).
```

In [ ]:
```
BR_Cities_df.iplot(kind='bar',mode='markers', x='CITY', y='IBGE_POP', xTitle='Ci
```

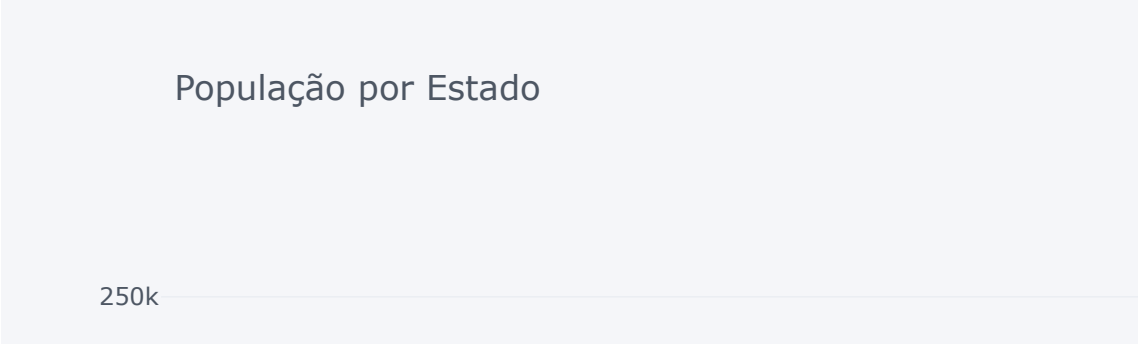


In [ ]:
```
BR_Cities_df.iplot(kind='bar',mode='markers', x='CITY', y='IBGE_POP', title='Pop
```



In [ ]:
```
BR_Cities_df.iplot(kind='box',mode='markers', x='CITY', y='IBGE_POP', title='Pop
```

In [ ]:
```
BR_Cities_df.iplot(kind='bubble', x='STATE', y='IBGE_POP', xTitle='Estado', yTit
```



In [ ]:
```
BR_Cities_df.iplot(kind='bubble', x='STATE', y='IBGE_POP', xTitle='Estado', yTit
```

In [ ]:
```
fig = px.scatter_3d(BR_Cities_df, x='STATE',
                    y='AREA',
                    z='IBGE_POP',
                    color='STATE',
                    title='População por Estado',
                    color_continuous_scale='rdylbu')
fig.update_layout(scene=dict(xaxis_title='Estado', yaxis_title='Área', zaxis_tit
fig.show()
```

População por Estado

Conclusion:

Summarize - State your findings from the analysis.

Provide insights or conclusions based on the visualizations and ### analyses performed.

Based on the DataFrame Select, we can conclude several things about the Brazilian Cities.

We can notice that:

The cities with the highest density on the data are the Capitals

The Biggest population are concetrated on the Southeast part of the Country

The Biggest Cities are localized in The State of PA

Although the Database contains some incorrect data, we didint have any problems with the study

I found some difficulties in the BoxPlots Studies because of my lack of Data Analisys Knowledge, so i couldn't select the better parameters to analise

This DATABASE also provides a vast perception of Economy in Brazil for each region.

You can conclude that by analising parameters like the number of veichles by cities and the IDHM Coluns

This Capstone Project Was made By

Enzo Rocha Leite Diniz Ribas - D24642 - 23/06/2024