Part-I: GDP Analysis of the Indian States

PART 1A : ANALYSE THE INDIAN STATES BASED ON THEIR GSDP and % GROWTH OF GSDP OVER YEARS

1. Remove the rows: '(% Growth over the previous year)' and 'GSDP - CURRENT PRICES (` in Crore)' for the year 2016-17

In [16]:

```
## TASK - 1
## Removed the rows: '(% Growth over the previous year)' and 'GSDP - CURRENT PRICES (`
 in Crore)' for the year 2016-17.

import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

import statistics

pd.set_option('display.max_columns', 500)

df = pd.read_csv("State-wise Gross Domestic Product (GDP) at current price on yearly ba
sis.csv")

df_non_2016_2017 = df[df['Duration'] != "2016-17"]
df_non_2016_2017
```

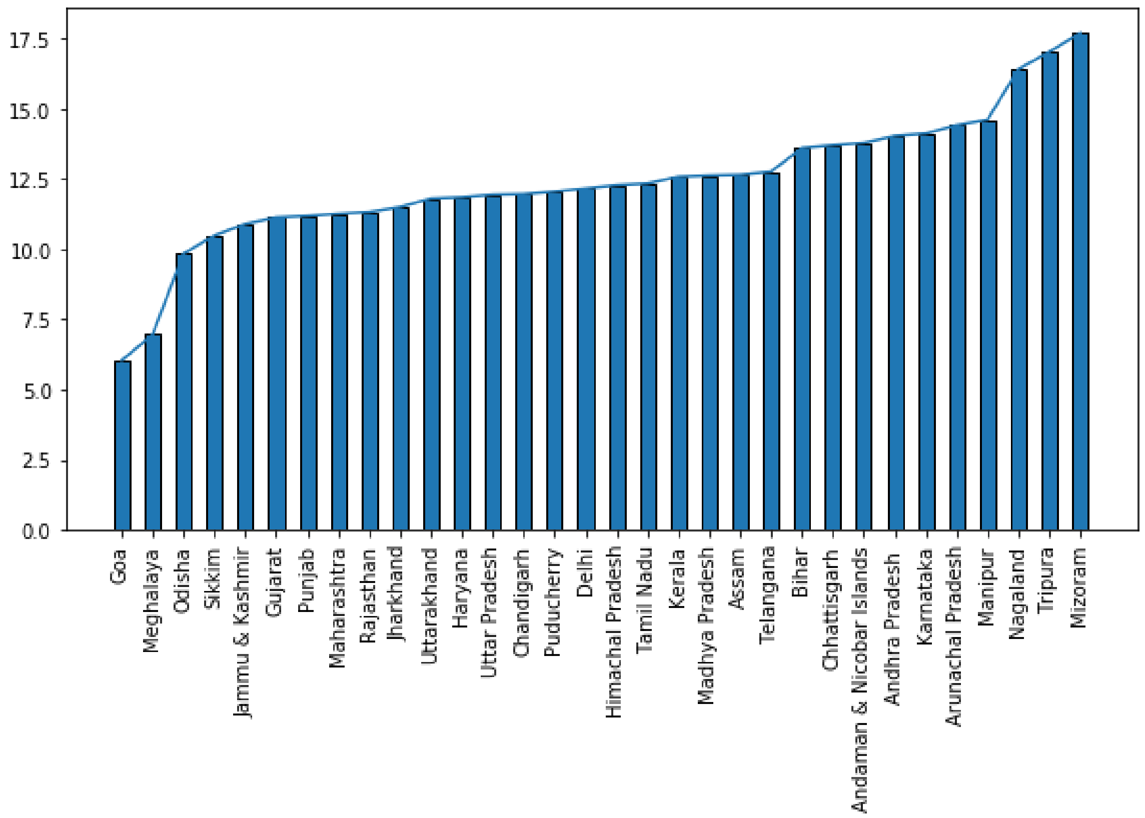
Out[16]:

	Items Description	Duration	Andhra Pradesh	Arunachal Pradesh	Assam	Bihar	Chhattisgarh	Goa
0	GSDP - CURRENT PRICES (` in Crore)	2011-12	379402.00	11063.00	143175.00	247144.00	158074.00	42367.00
1	GSDP - CURRENT PRICES (` in Crore)	2012-13	411404.00	12547.00	156864.00	282368.00	177511.00	38120.00
2	GSDP - CURRENT PRICES (` in Crore)	2013-14	464272.00	14602.00	177745.00	317101.00	206690.00	35921.00
3	GSDP - CURRENT PRICES (` in Crore)	2014-15	526468.00	16761.00	198098.00	373920.00	234982.00	40633.00
4	GSDP - CURRENT PRICES (` in Crore)	2015-16	609934.00	18784.00	224234.00	413503.00	260776.00	45002.00
6	(% Growth over previous year)	2012-13	8.43	13.41	9.56	14.25	12.30	-10.02
7	(% Growth over previous year)	2013-14	12.85	16.38	13.31	12.30	16.44	-5.77
8	(% Growth over previous year)	2014-15	13.40	14.79	11.45	17.92	13.69	13.12
9	(% Growth over previous year)	2015-16	15.85	12.07	13.19	10.59	10.98	10.75

2. CALCULATE THE AVERAGE GROWTH OF EACH STATE AND PLOT THE AVERAGE GROWTH OF THE STATES

In [17]:

```
## TASK - 2 :
## CALCULATE THE AVERAGE GROWTH OF EACH STATE AND PLOT THE AVERAGE GROWTH OF THE STATES

df2_temp = df_non_2016_2017.loc[df['Duration'] != "2012-13"].loc[df['Items  Description'] == "(% Growth over previous year)"]
## Get rid of the West Bengal as it doesnt have any details
df2_temp.dropna(axis=1,thresh=2, inplace = True)

## Creating a new dataframe for the purpose of calculating the average growth of the states

df_mean = df2_temp.drop('Items  Description', axis = 1)
df_mean.set_index('Duration', inplace = True)
df_mean.loc['mean'] = df_mean.mean()
# Rounding off the mean values to two digits
for column in list(df_mean.columns):
    df_mean[column] = list(map(lambda x: round(x,2),df_mean[column]))
# Removing the 'All India' column so the dataframe has only the states
# stroing All India mean first
df_states_mean = df_mean.drop('All_India GDP', axis = 1)

## Plot the mean values of the states
#numerics = ['float16', 'float32', 'float64']
state_list = list(df_states_mean.columns)
mean_list = list(df_states_mean[state_list].mean())

zippedList =  list(zip(state_list, mean_list))
df_mean_plot = pd.DataFrame(zippedList,columns = ['state' , 'mean_growth'])
df_mean_plot.sort_values('mean_growth', inplace = True)

plt.figure(figsize = (10,5))
plt.plot(list(df_mean_plot['state']), list(df_mean_plot['mean_growth']))
plt.bar(height = list(df_mean_plot['mean_growth']), x= list(df_mean_plot['state']), width = 0.5, edgecolor = "black")
plt.xticks(rotation='vertical')
plt.show()

df_states_mean
```



```
Out[17]:
```

	Andhra Pradesh	Arunachal Pradesh	Assam	Bihar	Chhattisgarh	Goa	Gujarat	Haryana	Himach Prades
Duration									
2013-14	12.85	16.38	13.31	12.30	16.44	-5.77	11.47	15.45	14.
2014-15	13.40	14.79	11.45	17.92	13.69	13.12	10.82	9.18	10.
2015-16	15.85	12.07	13.19	10.59	10.98	10.75	11.09	10.91	Na
mean	14.03	14.41	12.65	13.60	13.70	6.03	11.13	11.85	12.

3. FIND WHICH STATES HAVE BEEN CONSISTENTLY IMPROVING AND WHICH ONES ARE STRUGGLING

Method 1- Finding the top 5 and bottom 5 states as per the means of their average growth.

In [18]:

```
## This may not be the best way to determine the 'consistency', but let us see.

df_mean_ranking = df_states_mean.T
df_mean_ranking.sort_values('mean', inplace = True)


print("The top 5 states with the consistent growth rate are" + "  " + ', '.join(list(df
_mean_ranking.tail().index)))
print("The top 5 struggling states are " + "  " + ', '.join(list(df_mean_ranking.head()
.index)))
```

```
The top 5 states with the consistent growth rate are  Arunachal Pradesh, M
anipur, Nagaland, Tripura, Mizoram
The top 5 struggling states are   Goa, Meghalaya, Odisha, Sikkim, Jammu &
Kashmir
```

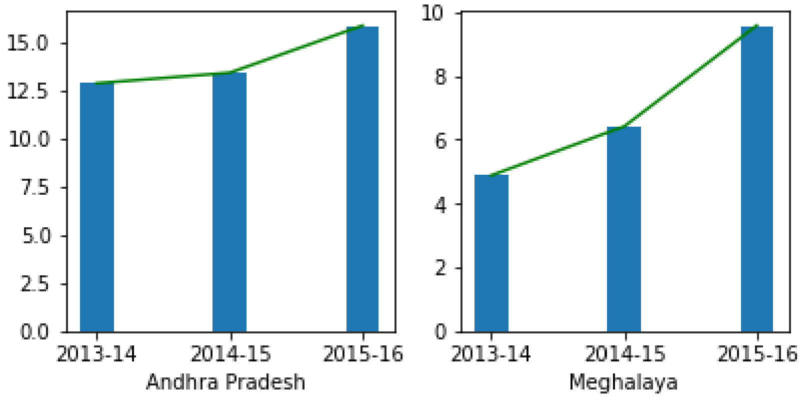
Method 2 (RECOMMENDED)- Measure the consistency based on the trend of average growth rates across three years

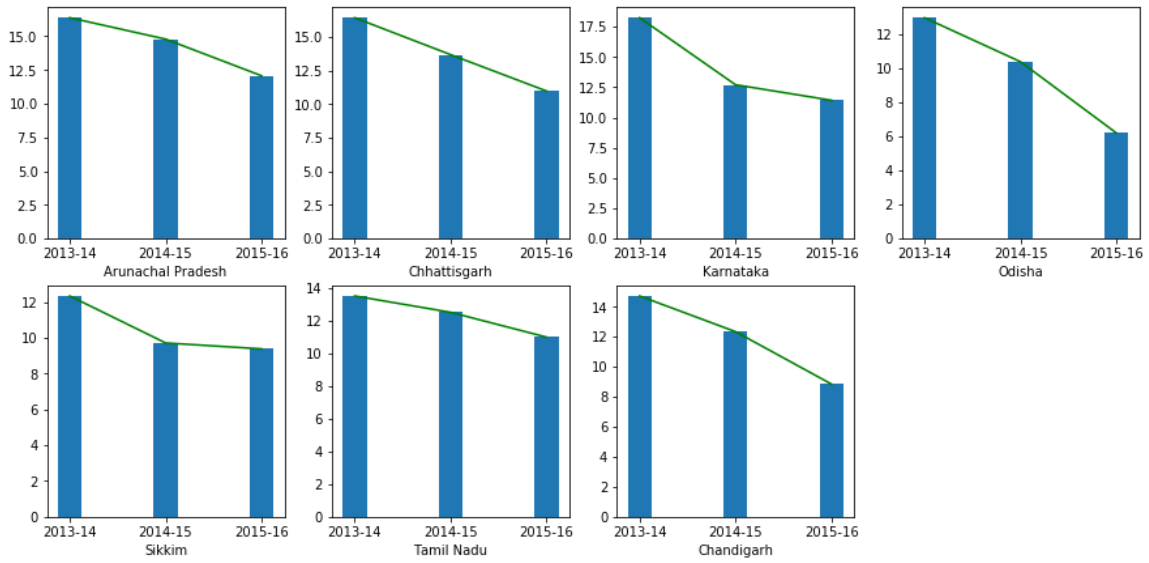
In [19]:

```
## TASK 3 - FIND WHICH STATES HAVE BEEN CONSISTENTLY IMPROVING AND WHICH ONES ARE STRUGGLING
## Compare the % growth of each state across three years. The states with continuous improvement from the previous FY
## the consistently performing high. And the ones consistently going down are the struggling states

## The same can be verified by visually plotting the % growth over the years


## Python code logic to figure out the performing and struggling states

df_states_perc_chg = df_states_mean.drop('mean', axis =0)

col_list = list(df_states_perc_chg.columns)
consistent_list = []
struggle_list = []

for column in col_list:

    val_list = list(df_states_perc_chg[column])
    if 'nan' not in str(val_list):
        if val_list == sorted(val_list):
            consistent_list.append(column)
        if val_list == sorted(val_list, reverse = True):
            struggle_list.append(column)

print("The list of states performing consistently are :" + "  " + ', '.join(consistent_list))
print("The list of states struggling consistently are :" + "  " + ', '.join(struggle_list))

## Visual representation of each state's data across three years

df_consistency = df_states_mean.T
df_consistency.drop('mean', axis =1, inplace = True)


## Plot the consistently growing states
subplot_cnt = 1

fig1 = plt.figure(figsize = (10,10))


for state in consistent_list:
    plt.subplot(3, 3, subplot_cnt, xlabel = state)
    #plt.title(state)
    plt.plot(['2013-14','2014-15','2015-16'],list(df_consistency.loc[state]), 'g')
    plt.bar(height=list(df_consistency.loc[state]), x =['2013-14','2014-15','2015-16'], width = 0.25)
    subplot_cnt = subplot_cnt+1


## Plot the consistently struggling states
subplot_cnt = 1

fig2 = plt.figure(figsize = (15,15))
```

```
for state in struggle_list:
    plt.subplot(4, 4, subplot_cnt, xlabel = state)
    #plt.title(state)
    plt.plot(['2013-14','2014-15','2015-16'],list(df_consistency.loc[state]), 'g')
    plt.bar(height=list(df_consistency.loc[state]), x =['2013-14','2014-15','2015-16'],
width = 0.25)
    subplot_cnt = subplot_cnt+1
```

```
The list of states performing consistently are :  Andhra Pradesh , Meghala
ya
The list of states struggling consistently are :  Arunachal Pradesh, Chhat
tisgarh, Karnataka, Odisha, Sikkim, Tamil Nadu, Chandigarh
```



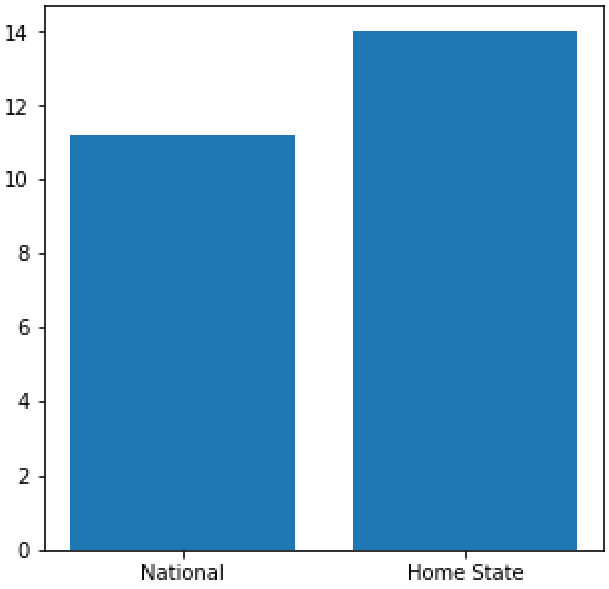


4. HOME STATE Vs ALL INDIA (NATIONAL) AVERAGE GROWTH

In [20]:

```
## TASK 4 - HOME STATE Vs ALL INDIA (NATIONAL) AVERAGE GROWTH

mean_national_GDP = df_mean.at['mean','All_India GDP']
mean_Andhra_Pradesh_GDP = df_mean.at['mean','Andhra Pradesh ']

home_vs_india = ((mean_Andhra_Pradesh_GDP-mean_national_GDP)/mean_national_GDP)*100

print("The average growth of my home state Andhra Pradesh is " + str(mean_Andhra_Pradesh_GDP) + "%")
print("The average growth of my home state Andhra Pradesh is " + str(mean_national_GDP) + "%")
print("Andhra Pradesh is performing " + str(int(home_vs_india)) + "% above the national average")

fig2 = plt.figure(figsize = (5,5))
plt.bar([0,1], height=[mean_national_GDP, mean_Andhra_Pradesh_GDP])
plt.xticks([0,1], ['National', 'Home State'])
#plt.bar(height=[mean_national_GDP, mean_Andhra_Pradesh_GDP] , x =['National', 'Home State'], width = 0.25, align = 'center')
plt.show()
```

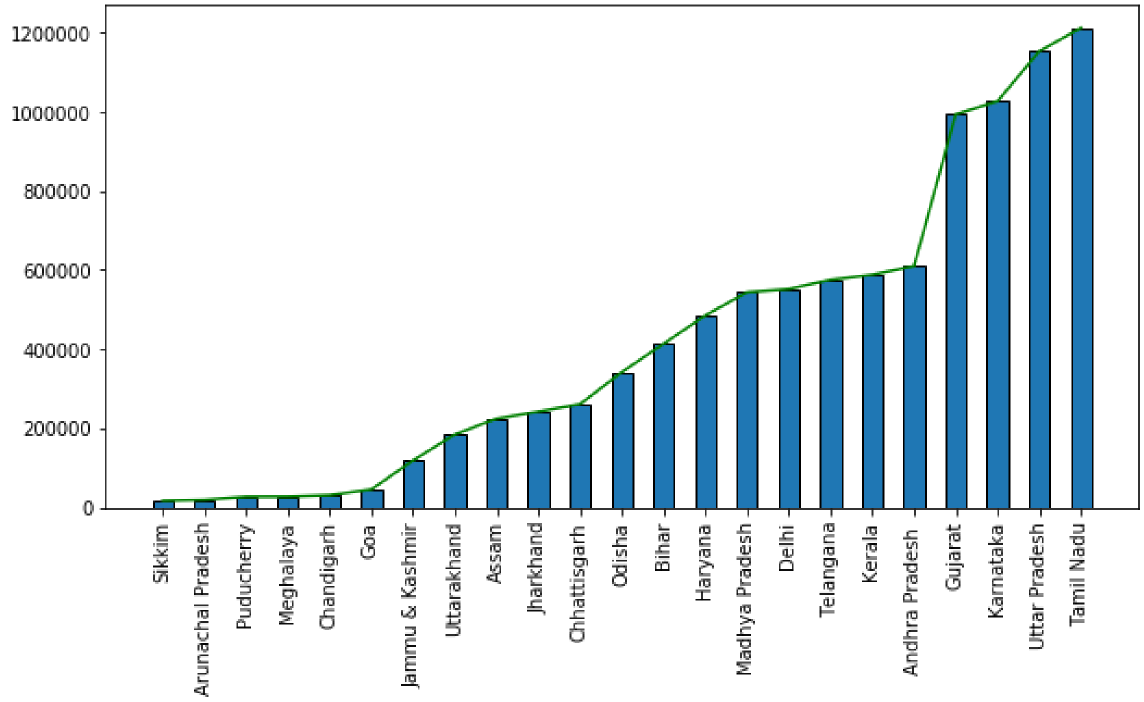
```
The average growth of my home state Andhra Pradesh is 14.03%
The average growth of my home state Andhra Pradesh is 11.2%
Andhra Pradesh is performing 25% above the national average
```



5. PLOT THE STATES AS PER THEIR GSDP VALUE. AND FIND THE TOP AND BOTTOM 5 STATES AS PER THEIR GSDP

In [21]:

```
## TASK 5 : PLOT THE STATES AS PER THEIR GSDP VALUE. AND FIND THE TOP AND BOTTOM 5 STAT
ES AS PER THEIR GSDP

df_GSDP = df_non_2016_2017.loc[df_non_2016_2017['Duration'] == "2015-16"].loc[df_non_20
16_2017['Items  Description'] == "GSDP - CURRENT PRICES (` in Crore)"]
df_GSDP_2015_16 = df_GSDP.drop('Items  Description', axis = 1)
df_GSDP_2015_16 = df_GSDP_2015_16.T
df_GSDP_2015_16.drop('Duration', inplace = True)

df_GSDP_2015_16.columns = ['GSDP']

df_GSDP_2015_16.sort_values('GSDP', inplace = True)
df_GSDP_2015_16.drop('All_India GDP', inplace = True)
df_GSDP_2015_16.dropna(inplace = True)

plt.figure(figsize = (10,5))
plt.plot(list(df_GSDP_2015_16.index), list(df_GSDP_2015_16['GSDP']),'g')
plt.bar(height=list(df_GSDP_2015_16['GSDP']), x =list(df_GSDP_2015_16.index), edgecolor
= "black", width = 0.5)

plt.xticks(rotation='vertical')
plt.show()

print("The top 5 states based on GSDP are:" + "  " + ', '.join(list(df_GSDP_2015_16.tai
l().index)))
print("The bottom 5 states based on GSDP are:" + "  " + ', '.join(list(df_GSDP_2015_16.
head().index)))
```



```
The top 5 states based on GSDP are:  Andhra Pradesh , Gujarat, Karnataka,
Uttar Pradesh, Tamil Nadu
The bottom 5 states based on GSDP are:  Sikkim, Arunachal Pradesh, Puduche
rry, Meghalaya, Chandigarh
```

Part 1-B : ANALYSIS OF INDIAN STATES BASED ON THEIR GDP PER CAPITA

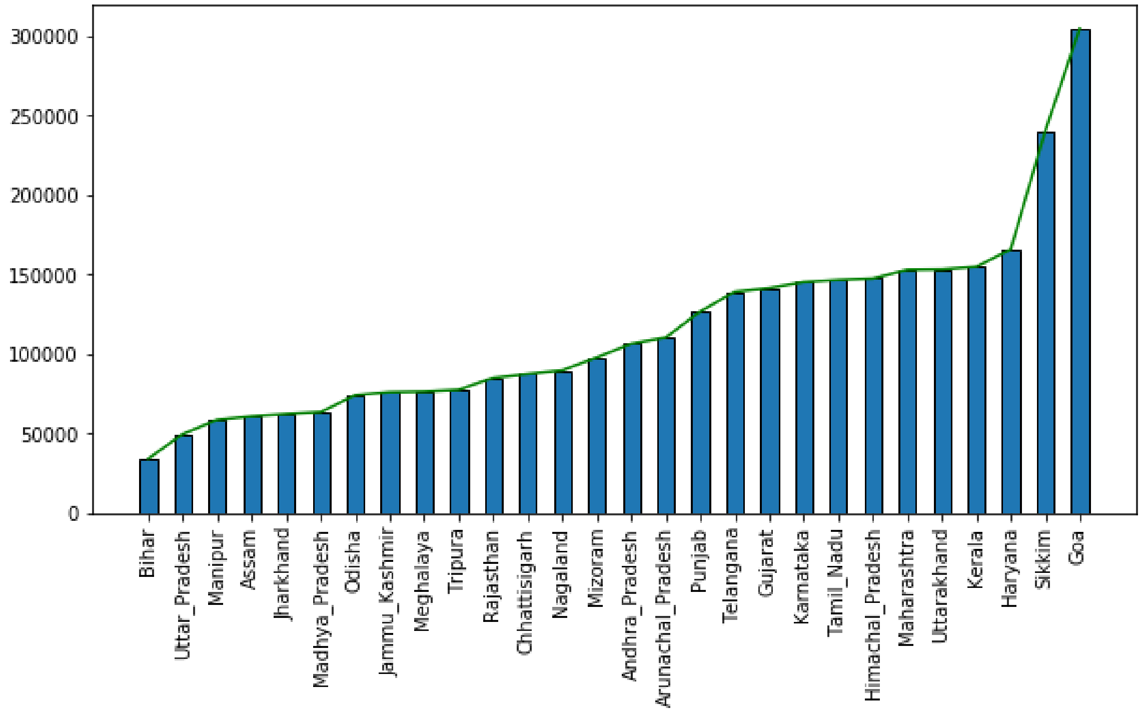
In [22]:

```
## Create the Master Data Frame combining the data from individual CSV files
import glob

pd.set_option('display.max_columns', 500)

all_files = glob.glob("GSVAfiles/*.csv")

li = []
state_list = ['Andhra_Pradesh', 'Arunachal_Pradesh', 'Assam', 'Bihar', 'Chhattisigarh',
              'Goa', 'Gujarat', 'Haryana', 'Himachal_Pradesh', 'Jammu_Kashmir', 'Jharkhand',
              'Karnataka', 'Kerala', 'Madhya_Pradesh', 'Maharashtra', 'Manipur', 'Meghalaya',
              'Mizoram', 'Nagaland', 'Odisha', 'Punjab', 'Rajasthan', 'Sikkim', 'Tamil_Nadu',
              'Telangana', 'Tripura', 'Uttar_Pradesh', 'Uttarakhand', 'West Bengal1']

for filename in all_files:
    for statename in state_list:
        if (filename.find(statename) != -1):
            df = pd.read_csv(filename, index_col = None, header = 0, encoding = "ISO-8859-1")
            df['state'] = statename
            li.append(df)

master_df = pd.concat(li, sort = False)

df = master_df.drop(['2011-12', '2012-13', '2013-14', '2015-16', 'S.No.', 'S. N.','2016-17'], axis = 1)
df.set_index('state', inplace = True)
```

1. Plot the GDP Per Capita of all the states

In [23]:

```
# TASK 1 :
## 1. PLOT THE GDP PER CAPITA OF ALL THE STATES
## 2. IDENTIFY THE TOP 5 AND BOTTOM 5 STATES BASED ON THE GDP PER CAPITA
## 3. FIND THE RATIO OF THE HIGHEST PER CAPITA GDP TO THE LOWEST PER CAPITA GDP

import matplotlib.pyplot as plt
import seaborn as sns

pd.options.mode.chained_assignment = None

df_per_cap_gdp = df[df['Item'].isin(['Per Capita GSDP (`)','Per Capita GSDP (Rs.)'])]
df_per_cap_gdp.columns = ['sector', 'Per Capita GDP']

df_per_cap_gdp['Per Capita GDP'] = list(map(lambda x: int(x),list(df_per_cap_gdp['Per C
apita GDP'])))

df_per_cap_gdp_sorted = df_per_cap_gdp.sort_values('Per Capita GDP')

# plot the states as per their per capita GDP
plt.figure(figsize = (10,5))
plt.plot(list(df_per_cap_gdp_sorted.index), list(df_per_cap_gdp_sorted['Per Capita GDP'
]),'g')
plt.bar(height = list(df_per_cap_gdp_sorted['Per Capita GDP']), x= list(df_per_cap_gdp_
sorted.index), width = 0.5, edgecolor = "black" )
plt.xticks(rotation='vertical')
plt.show()

## top and bottom 5 states per capita GDP
print("The top 5 performing states as per the GDP per capita are:" + "  " + ', '.join(l
ist(df_per_cap_gdp_sorted.tail().index)))
print("The bottom 5 performing states as per the GDP per capita are:"+ "  " + ', '.join
(list(df_per_cap_gdp_sorted.head().index)))

## Ratio of highest to lowest per capita GDP
gdp_list = (list(df_per_cap_gdp_sorted['Per Capita GDP']))

highest_GDP = gdp_list[-1]
lowest_GDP = gdp_list[0]

high_to_low_ratio = round((highest_GDP/lowest_GDP)*100,0)
print("The ratio of highest to lowest GDP is: " + str(int(high_to_low_ratio))+ "%")
```

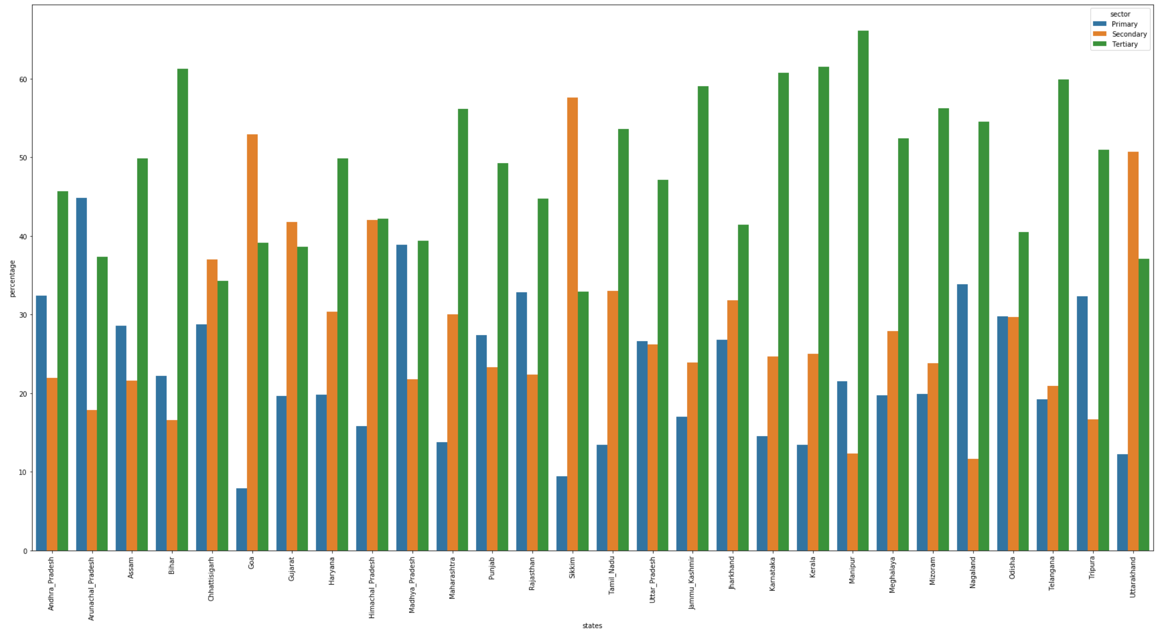


```
The top 5 performing states as per the GDP per capita are:  Uttarakhand, K
erala, Haryana, Sikkim, Goa
The bottom 5 performing states as per the GDP per capita are:  Bihar, Utta
r_Pradesh, Manipur, Assam, Jharkhand
The ratio of highest to lowest GDP is: 897%
```

2. Plot the percentage contribution of Primary, Secondary and Tertiary sectors as a percentage of total GDP for all states

In [24]:

```
## TASK 2 : PLOT THE PERCENTAGE CONTRIBUTION OF PRIMARY, SECONDARY AND TERTIARY SECTORS
AS A PERCENTAGE OF THE TOTAL GDP FOR ALL THE STATES

import matplotlib.pyplot as plt
import seaborn as sns

df_sectors = df[df['Item'].isin(['TOTAL GSVA at basic prices', 'Primary', 'Secondary',
'Tertiary', 'Tertiary  '])]
df_sectors.columns = ['sector', 'Per Capita GDP']
df_sectors.shape

def percentage(part, whole):
  return round((100 * (float(part)/float(whole))),2)



perc_list = []
gdp_sub_list = list(df_sectors['Per Capita GDP'])

for x in range(3,len(gdp_sub_list),4):
    perc_list.append(percentage(gdp_sub_list[x-3], gdp_sub_list[x]))
    perc_list.append(percentage(gdp_sub_list[x-2], gdp_sub_list[x]))
    perc_list.append(percentage(gdp_sub_list[x-1], gdp_sub_list[x]))
    perc_list.append('NA')

df_sectors['states'] = list(df_sectors.index)
df_sectors['percentage'] = perc_list
df_sectors_plot = df_sectors.loc[df_sectors['sector'] != 'TOTAL GSVA at basic prices']
df_sectors_plot.replace('Tertiary  ', 'Tertiary', inplace = True)

fig1 = plt.figure(figsize = (30,15))
sns.barplot(x="states", y="percentage", hue = "sector", data=df_sectors_plot)
plt.xticks(rotation='vertical')
plt.show()
```



3. Categorize the states into C1, C2, C3 and C4 as per the percentile of their GDP per cpita

In [25]:

```
## TASK 3 : CATEGORIZE THE STATES TO C1, C2, C3 and C4 BASED ON THEIR GDP PER CAPITA PE
RCENTILE

c1 = df_per_cap_gdp_sorted['Per Capita GDP'].quantile(0.85)
c2 = df_per_cap_gdp_sorted['Per Capita GDP'].quantile(0.5)
c3 = df_per_cap_gdp_sorted['Per Capita GDP'].quantile(0.2)

per_cap_gdp_list = list(df_per_cap_gdp_sorted['Per Capita GDP'])
category_list = []

for gdp_item in per_cap_gdp_list:
    if gdp_item >= c1:
        category_list.append('c1')
    elif gdp_item >= c2:
        category_list.append('c2')
    elif gdp_item >= c3:
        category_list.append('c3')
    else:
        category_list.append('c4')

df_per_cap_gdp_sorted['GDP Category'] = category_list
df_per_cap_gdp_sorted.drop('sector', axis = 1, inplace = True)
df_per_cap_gdp_sorted
```

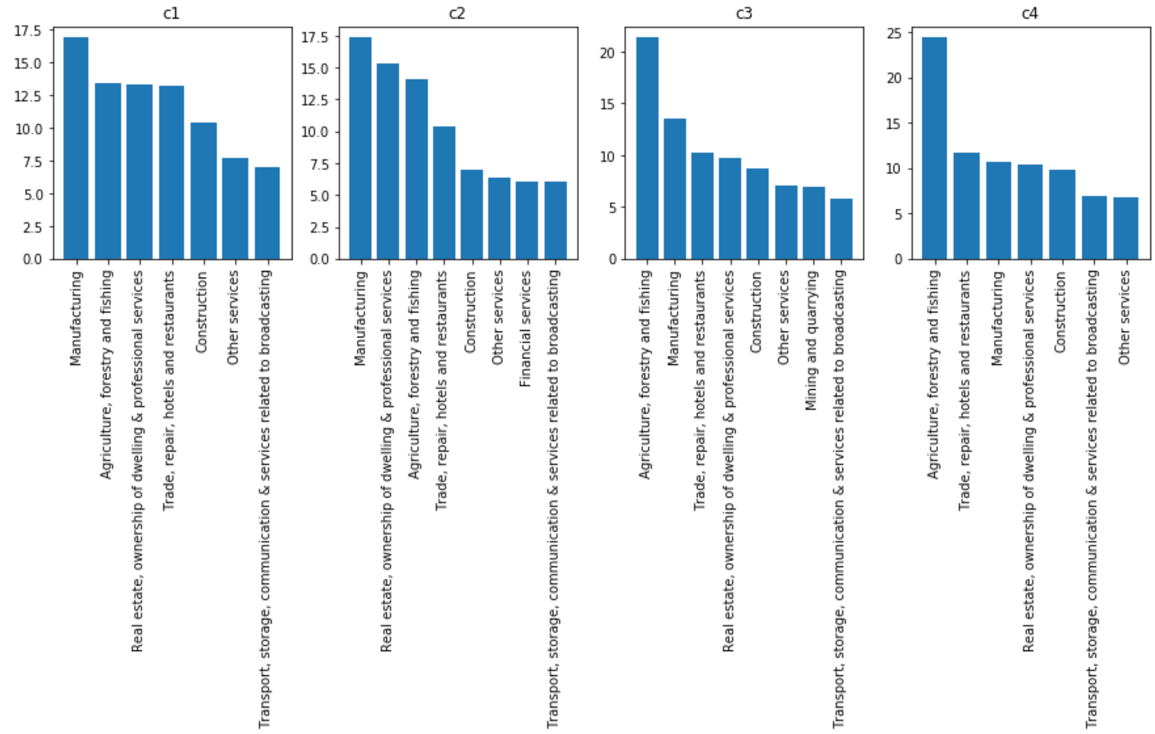
Out[25]:

	Per Capita GDP	GDP Category
state		
Bihar	33954	c4
Uttar_Pradesh	49450	c4
Manipur	58442	c4
Assam	60621	c4
Jharkhand	62091	c4
Madhya_Pradesh	63323	c4
Odisha	73979	c3
Jammu_Kashmir	75840	c3
Meghalaya	76228	c3
Tripura	77358	c3
Rajasthan	84837	c3
Chhattisigarh	87353	c3
Nagaland	89607	c3
Mizoram	97687	c3
Andhra_Pradesh	106263	c2
Arunachal_Pradesh	110216	c2
Punjab	126606	c2
Telangana	139035	c2
Gujarat	141404	c2
Karnataka	145141	c2
Tamil_Nadu	146503	c2
Himachal_Pradesh	147330	c2
Maharashtra	152852	c2
Uttarakhand	153076	c1
Kerala	154778	c1
Haryana	165728	c1
Sikkim	240273	c1
Goa	304665	c1

4. Find the subsectors contributing upto 80% of the GSDP for each category (C1-C4)

In [26]:

```
## TASK 4.1 : TOP 3/4/5 SECTORS CONTRIBUTING TO THE 80% of GSDP FOR C1 to C4 CATEGORY S
TATES

## Re-usable Functions for calculating the percentages
def perc_calc_function(list1):
    perc_list = list(map(lambda x: round((x/list1[-1])*100,2), list1))
    diff = len(list1) - len(perc_list)
    for i in range(0, diff):
        perc_list.append(0.01)
    return perc_list


def perc_80_contr_function(list1):
    benchmark = 80
    sum_80 = 0
    index_80 = 0

    for i in range(0,len(list1)):
        if sum_80 >= benchmark:
            break
        else:
            sum_80 = sum_80 + list1[i]
            index_80 = i

    contr_list = []
    for i in range(0, len(list1)):
        if i <= index_80:
            contr_list.append('Y')
        else:
            contr_list.append('N')
    return contr_list

subplot_category_count = 1
fig_c1 = plt.figure(figsize = (15,15))

category_list_for_plot = sorted(list(set(list(df_per_cap_gdp_sorted['GDP Category']))))

for category in category_list_for_plot:
    category_state_list = list(df_per_cap_gdp_sorted[df_per_cap_gdp_sorted['GDP Categor
y'] == category].index)
    df_category = df.loc[category_state_list]
    df_category_sliced = df_category[df_category['Item'].isin(['Agriculture, forestry a
nd fishing',
                                                           'Mining and quarrying', 'Manu
facturing',
                                                           'Electricity, gas, water supp
ly & other utility services',
                                                           'Construction', 'Trade, repai
r, hotels and restaurants',
                                                           'Transport, storage, communic
ation & services related to broadcasting',
                                                           'Financial services', 'Real e
state, ownership of dwelling & professional services',
                                                           'Public administration', 'Oth
er services', 'Gross State Domestic Product'])]

    df_category_sliced.set_index('Item', inplace = True)
    df_category_cleaned = df_category_sliced.rename(index = {"Trade & repair services*"
: "Trade & repair services",
                                                               "Road transport**":"Road t
ransport", "Population ('00)":"Population (In Million)",
                                                               "Road transport*":"Road tr
ansport",
                                                               "Services incidental to tr
ansport*" : "Services incidental to transport"})
    df_category_grouped = df_category_cleaned.groupby('Item', sort=False).sum()
    df_category_grouped['% Contribution to GSDP'] = perc_calc_function(list(df_category
_grouped['2014-15']))
    df_category_sorted = df_category_grouped.loc[df_category_grouped['% Contribution to
GSDP'] != 100.00].sort_values('% Contribution to GSDP', ascending = False)
    df_category_sorted['contr to 80%'] = perc_80_contr_function(list(df_category_sorted
['% Contribution to GSDP']))
    df_category_for_plot = df_category_sorted[df_category_sorted['contr to 80%'] == 'Y'
]

    plt.subplot(4, 4,subplot_category_count , title = category)
    plt.bar(height= list(df_category_for_plot['% Contribution to GSDP']), x = list(df_c
ategory_for_plot.index))
    plt.xticks(rotation='vertical')
    subplot_category_count = subplot_category_count+1

plt.show()
```



PART 2 : CORELATION BETWEEN GSDP and DROPOUT RATES

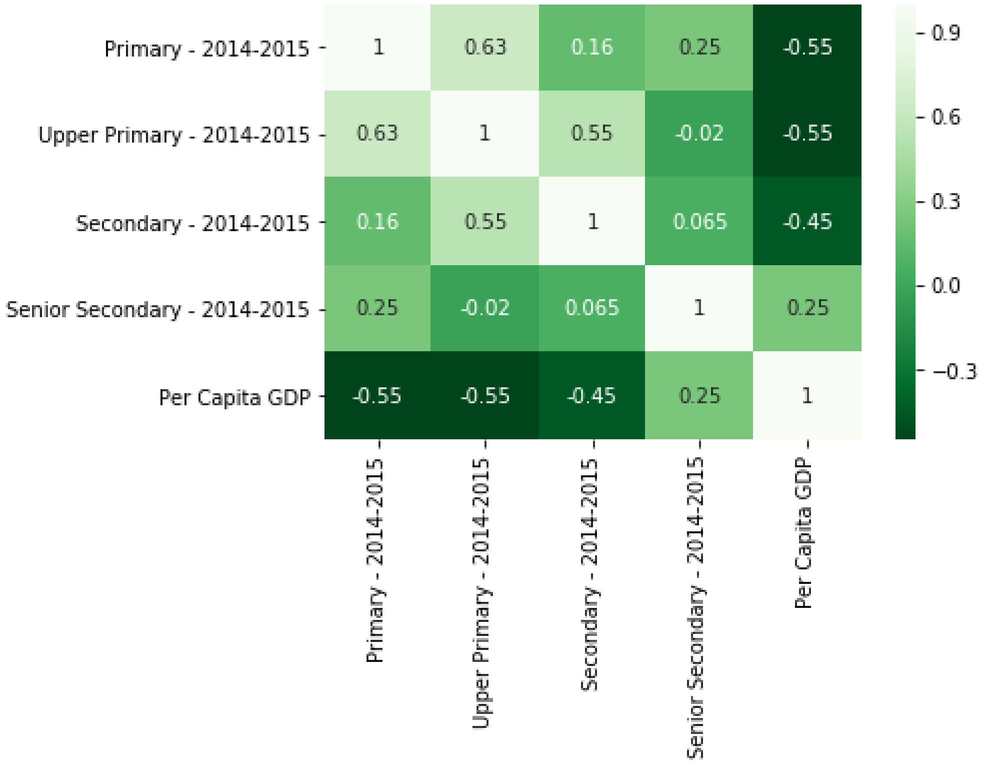
1. Find the corelation between GDP Per Capita and the dropout rates at different levels of education

In [27]:

```
## Dropout Rate
import seaborn as sns

df_dropout_raw = pd.read_csv(r'C:\Users\naren\Documents\Python Scripts\Exams\GDP Assign
ment\Part 2\rs_session243_au570_1.1.csv', index_col = None, header = 0, encoding = "ISO
-8859-1")
df_droupout_2014_15 = df_dropout_raw[['Primary - 2014-2015', 'Upper Primary - 2014-201
5',
                                                 'Secondary - 2014-2015', 'Senior Secon
dary - 2014-2015',
                                                'Level of Education - State']]

df_droupout_2014_15.set_index('Level of Education - State', inplace = True)
df_droupout_2014_15.dropna()

df_droupout_2014_15.drop(['A & N Islands', 'Dadra & Nagar Haveli', 'Daman & Diu', 'Delh
i',
                         'Lakshadweep', 'West Bengal', 'All India','Puducherry'], axis
= 0, inplace = True)


df_droupout_2014_15 = df_droupout_2014_15.rename(index = {"Andhra Pradesh": "Andhra_Pra
desh",
                                                          "Arunachal Pradesh" : "Arunac
hal_Pradesh",
                                                          "Himachal Pradesh" : "Himacha
l_Pradesh",
                                                          "Madhya Pradesh" : "Madhya_Pr
adesh",
                                                          "Uttar Pradesh" : "Uttar_Prad
esh",
                                                          "Jammu and Kashmir" : "Jammu_
Kashmir",
                                                          "Chhatisgarh":"Chhattisigarh"
,
                                                          "Tamil Nadu": "Tamil_Nadu",
                                                          "Uttrakhand":"Uttarakhand"})

df_gdp_dropout = df_droupout_2014_15.join(df_per_cap_gdp_sorted)

gdp_dropout_corr = df_gdp_dropout.corr()
sns.heatmap(gdp_dropout_corr, xticklabels = gdp_dropout_corr, yticklabels = gdp_dropout
_corr,
            cmap = "Greens_r", annot = True)
gdp_dropout_corr
```

Out[27]:

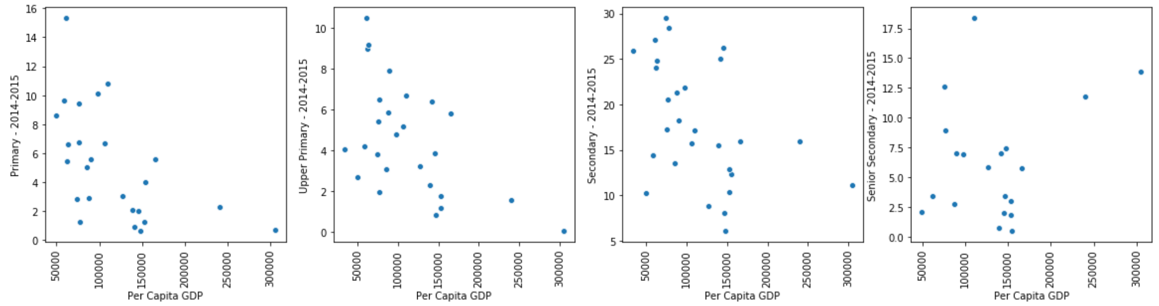
	Primary - 2014-2015	Upper Primary - 2014-2015	Secondary - 2014-2015	Senior Secondary - 2014-2015	Per Capita GDP
Primary - 2014-2015	1.000000	0.625241	0.161162	0.253925	-0.549948
Upper Primary - 2014-2015	0.625241	1.000000	0.545293	-0.019598	-0.545138
Secondary - 2014-2015	0.161162	0.545293	1.000000	0.065126	-0.450661
Senior Secondary - 2014-2015	0.253925	-0.019598	0.065126	1.000000	0.249290
Per Capita GDP	-0.549948	-0.545138	-0.450661	0.249290	1.000000



2. Check the distribution of the dropout rates using the scatter plots

In [28]:

```
fig_dropout = plt.figure(figsize = (20,20))
cmap = sns.cubehelix_palette(dark=.3, light=.8, as_cmap=True)

plt.subplot(4, 4,1)
snsplot1 = sns.scatterplot(x="Per Capita GDP", y="Primary - 2014-2015", data=df_gdp_dropout, palette = cmap)
plt.xticks(rotation='vertical')

plt.subplot(4, 4,2)
snsplot2 = sns.scatterplot(x="Per Capita GDP", y="Upper Primary - 2014-2015", data=df_gdp_dropout,palette = cmap)
plt.xticks(rotation='vertical')

plt.subplot(4, 4,3)
snsplot3 = sns.scatterplot(x="Per Capita GDP", y="Secondary - 2014-2015", data=df_gdp_dropout,palette = cmap)
plt.xticks(rotation='vertical')

plt.subplot(4, 4,4)
snsplot4 = sns.scatterplot(x="Per Capita GDP", y="Senior Secondary - 2014-2015", data=df_gdp_dropout,palette = cmap)
plt.xticks(rotation='vertical')


plt.show()
```


Primary - 2014-2015
Upper Primary - 2014-2015
Secondary - 2014-2015
Senior Secondary - 2014-2015
Per Capita GDP