Part-I: GDP Analysis of the Indian States

Part I-A

```
# Supress Warnings

import warnings

warnings.filterwarnings('ignore')


# Import the numpy and pandas packages

import numpy as np

import pandas as pd


# Read the csv file using 'read_csv'

DataA = pd.read_csv('Data 1a State-wise Gross Domestic Product (GDP) at current price on yearly basis.csv')

print(DataA)
```

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

```
DataA.columns = [c.replace(' ', '_') for c in DataA.columns] # replacing column spaces with '_' for easy access

indexNames = DataA.index[((DataA['Items__Description']=='(% Growth over previous year)') & (DataA['Duration']=='2016-17'))]

DataA = DataA.drop(indexNames)


indexNames = DataA.index[((DataA['Items__Description']=='GSDP - CURRENT PRICES (` in Crore)') & (DataA['Duration']=='2016-17'))]

DataA = DataA.drop(indexNames)

print(DataA)
```

Calculate the average growth of states for the duration 2013-14, 2014-15 and 2015-16 by taking the mean of the row '(% Growth over previous year)'.

```
Avg_growth_rate_states = DataA.loc[(DataA.Items__Description == "(% Growth over previous year)")&(DataA.Duration != '2011-12')&(DataA.Duration != '2012-13')].mean(axis=0)
```

Dropping All_India_GDP column since we are concentrating only on States and at same time removing null valued fields.

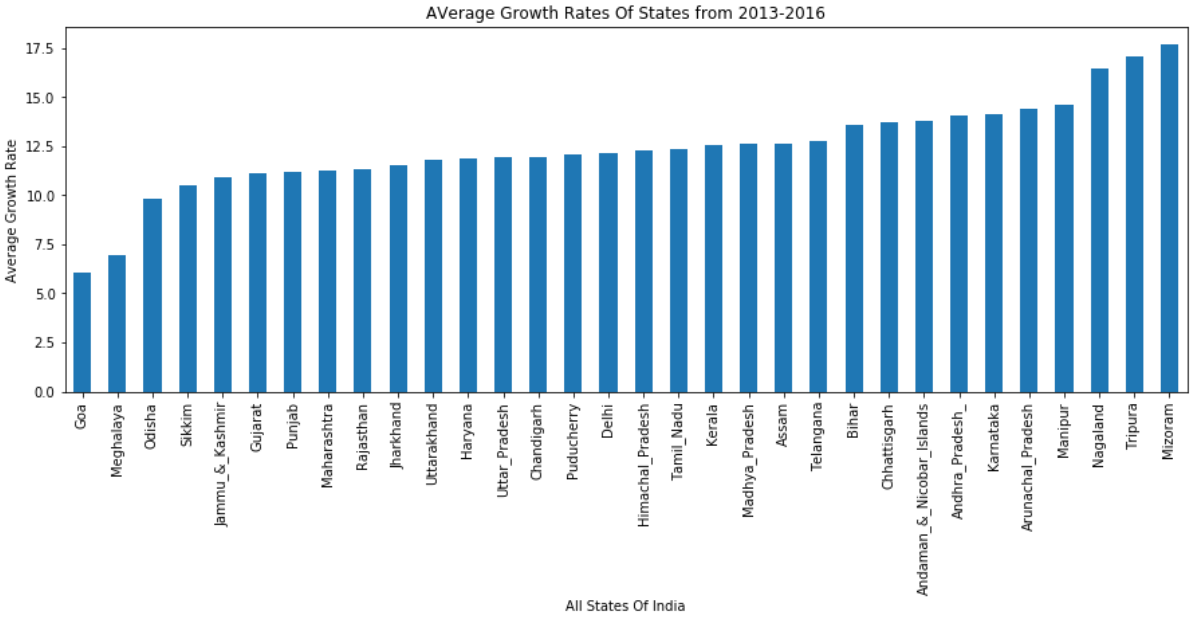
```
Avg_growth_rate_transformd = Avg_grw_rate_states.drop(["All_India_GDP"]).dropna()
```

Comparison chart for all states for average growth of states in percentage

Avg_growth_rate_transformd

Compare the calculated values and plot them for the states. Make appropriate transformations, if necessary, to plot the states in Increasing order.

```
import matplotlib.pyplot as plt

plt.figure(figsize=(15,5))

Avg_growth_rate_transformd.sort_values().plot(kind='bar')

plt.ylabel('Average Growth Rate')

plt.xlabel('All States Of India')

plt.title('AVerage Growth Rates Of States from 2013-2016')

plt.show()
```



Which states have been growing consistently fast, and which ones have been struggling?

#From the above bar graph-

#The states that are growing fast are: Arunachal pradesh, Manipur, Nagaland, Tripura and Mizoram

#The States that are struggling in their growth are: Goa, Meghalaya, Odisha, Sikkim and Jammu & Kashmir.

Curiosity exercise: What has been the average growth rate of your home state, and how does it compare to the national average over this duration?

#Average growth rate of my home state (Maharashtra) over the period 2013-2016 is as below:

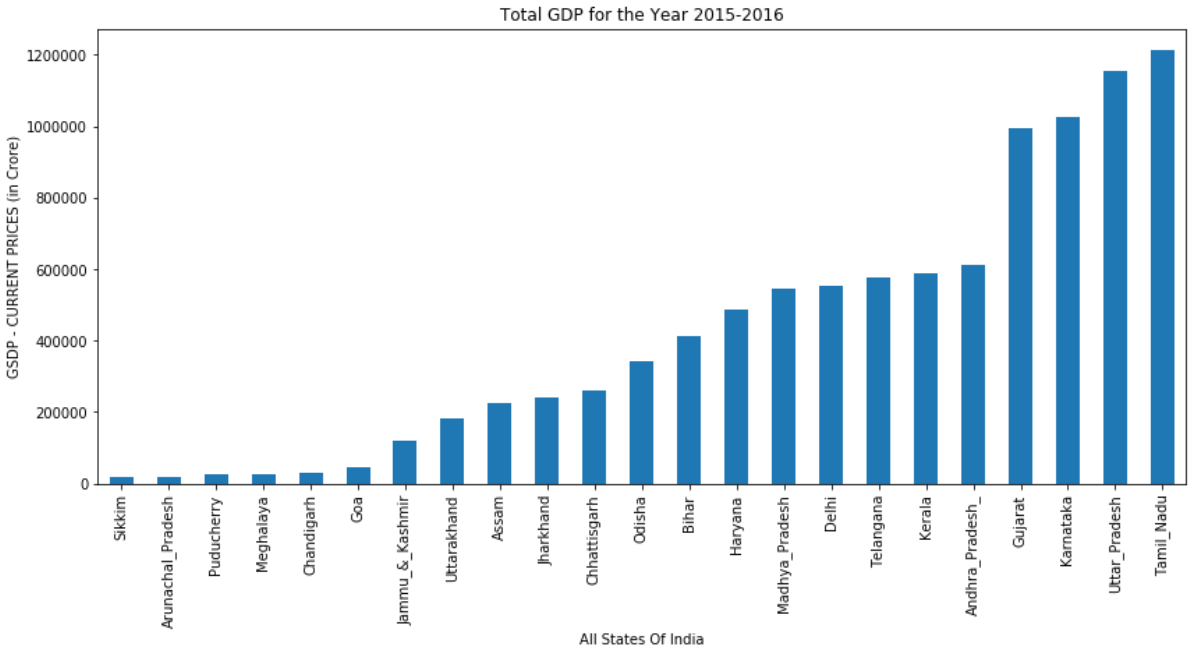
```
Avg_growth_rate_states['Maharashtra']

Avg_growth_rate_states['Maharashtra']/Avg_growth_rate_states['All_India_GDP']

#Average growth rate is 1.006% more than National Growth rate
```

Plot the total GDP of the states for the year 2015-16:

```
plt.figure(figsize=(14,6))


# In the below line of Code, First, we are selecting the required rows
using:[(DataA.Items__Description == "GSDP - CURRENT PRICES (` in Crore)")&(DataA.Duration ==
'2015-16')]

# Second, we are selecting the required columns (using: iloc[:,2:-1]), by removing the columns:
(Items__Description, Duration) and keeping only the states.

# Third, we are transposing the dataframe for the benefit of plotting, insted of keeping it as a single
row dataframe. Transposing will automatically convert columns into index.

# Fourth, converting the entire dataframe into a series by selecting the last column using: iloc[:,-1],
this will give clean plotting.

# Fifth, sorting the values (using: sort_values()) so that while plotting the graph it will be in order.

# Sixth, dropping all states that are havong null values (using: dropna())

# Seventh, plotting the bar graph using: plot(kind='bar'), this will take the give series into y-axis and
its index as x-axis.
```

Identify the top 5 and the bottom 5 states based on total GDP.

```
DataA.loc[(DataA.Items__Description == "GSDP - CURRENT PRICES (` in Crore)")&(DataA.Duration ==
'2015-16')].iloc[:,2:-1].T.iloc[:,-1].sort_values().dropna().plot(kind='bar')


plt.ylabel('GSDP - CURRENT PRICES (in Crore)')


plt.xlabel('All States Of India')


plt.title('Total GDP for the Year 2015-2016')


plt.show()
```



#Top 5 states based on total GSD as per the above plot are: Tamil Nadu, Uttar_Pradesh, Karnataka, Gujarat, Andhra_pradesh

#Bottom 5 stated based on the total GSD are as follows: Sikkim, Arunachal_pradesh, Puduchery, Meghalaya and Chandigarh

Part I-B:

Plot the GDP per capita for all the states.

```
import glob, os

import pandas.io.common

import pandas as pd


# Reading all the states csv files and creating an array with the file names.

# get data file names

file_paths = glob.glob("Data 1b" + "/*.csv")

# We are performing the analysis only for the duration : 2014-15 as requested in step 1

req_columns = ['S.No.','Item','2014-15']

# We are not taking the union territories while creating the data frame

# Creating a single dataframe (df_all_states) by merging all CSV files and creating a new column State, as per the order of below steps.

# First, read the csv files using: pd.read_csv(i, encoding = 'ISO8859', usecols=req_columns), ISO encoding is used since unicode encoding is not reading the white spaces properly, throwing errors.
```

Second, creating a new column 'State' using: assign(State = i.split('-')[1].replace('_',' '), we are splitting the file name and getting the column name from it.

Third, we are replacing the column names from Arunachal_Pradesh' to ‘Arunachal Pradesh' using: replace('_',' ')

Fourth, we are using the variables req_columns and union_terr respectively, to use on the required columns and remove union territories.

```
df_all_states = pd.concat([pd.read_csv(i, encoding = 'ISO8859', usecols=req_columns).assign(State = i.split('-')[1].replace('_',' '))
            for i in file_paths if i.split('-')[2].replace('_',' ')])
df_all_states
```

This dataframe has all datas merged from all CSV files plus a new column state to represent its respective states.

#Plot the GDP per capita for all the states

```
plt.figure(figsize=(14,6))
```

From the below code, first, selecting the necessary rows using: loc[(df_all_states.Item == "Per Capita GSDP (Rs.)")]

Second, selecting the necessary columns: State and 2014-15 using: iloc[:,2:]

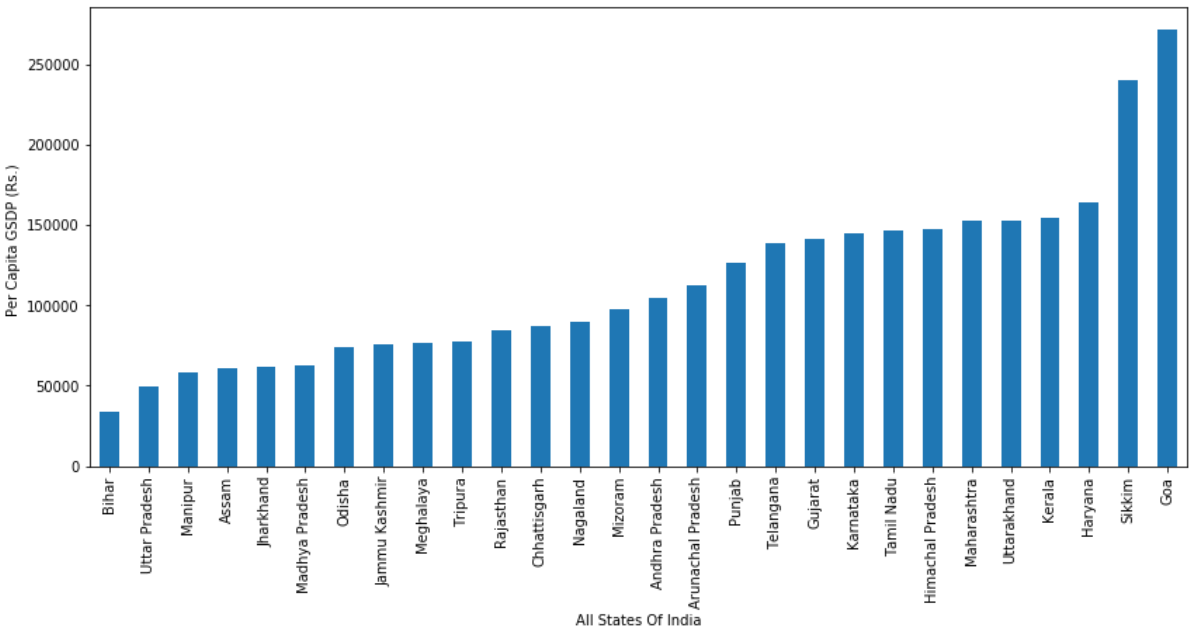
Third, we are sorting the values using: sort_values(by = '2014-15'), so that it will be in order while plotting.

Fouth, we are setting the index of this dataframe as 'State' column.

Fifth, converting the dataframe into series, which will be helpful for clean plotting.

Sixth, plotting the graph using: plot(kind='bar'), this will take y-axis as given series and its index (State) as x-axis.

```
df_all_states.loc[(df_all_states.Item == "Per Capita GSDP (Rs.)")].iloc[:,2:].sort_values(by = '2014-15').set_index('State').iloc[:,-1].plot(kind='bar')
plt.ylabel('Per Capita GSDP (Rs.) ')
plt.xlabel('All States Of India')
plt.show()
```



#Based on GDP per capita from above plot:

Identify the top 5 and the bottom 5 states based on the GDP per capita.

#Top-5 states are: Goa, Sikkim, Haryana, Kerala, Uttarakhand

#Bottom-5 states are: Bihar, Uttar pradesh, Manipur, Assam, Jharkhand

Find the ratio of the highest per capita GDP to the lowest per capita GDP.

#Based on above plot highest per capita state is Goa and the lowest is Bihar, their ratio is as follows:

```
df_all_states.loc[(df_all_states.State == "Goa")&(df_all_states.Item == "Per Capita GSDP (Rs.)")].iloc[:,-2].T[32] / df_all_states.loc[(df_all_states.State == "Bihar")&(df_all_states.Item == "Per Capita GSDP (Rs.)")].iloc[:,-2].T[32]
```

```
# Selecting required rows and columns using: loc[(df_all_states.Item == "Gross State Domestic Product")][['2014-15','State']]
```

```
# Renaming the column for convenience.
```

```
df_total_GDP = df_all_states.loc[(df_all_states.Item == "Gross State Domestic Product")][['2014-15','State']].rename(columns={'2014-15':'GSDP'})
```

```
df_total_GDP.head()
```

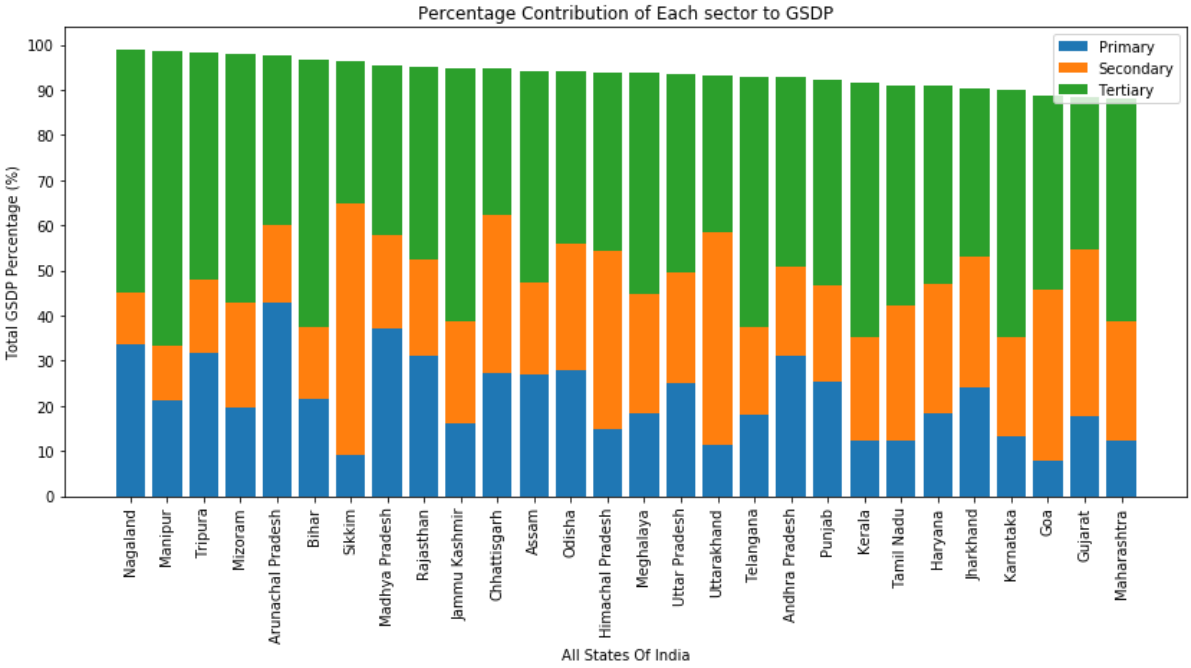
```
df_prim_sec_ter = df_all_states.loc[(df_all_states.Item == "Primary")][['2014-15','State']].rename(columns={'2014-15':'Primary_GSVA'})
```

```
# same as above


# Merging primary and secondary using state as common column


df_prim_sec_ter = pd.merge(df_prim_sec_ter, df_all_states.loc[(df_all_states.Item ==
"Secondary")][['2014-15','State']], how = 'inner', on = 'State').rename(columns={'2014-
15':'Secondary_GSVA'})


# Merging primary, secondary and tertiary using state as common column


df_prim_sec_ter = pd.merge(df_prim_sec_ter, df_all_states.loc[(df_all_states.Item ==
"Tertiary")][['2014-15','State']], how = 'inner', on = 'State').rename(columns={'2014-
15':'Tertiary_GSVA'})

df_prim_sec_ter.head()


# Merging the dataframes: df_prim_sec_ter, df_total_GDP to get the result as shown in below table.

df_total_GDP_pri_sec_ter = pd.merge(df_prim_sec_ter, df_total_GDP, how = 'inner', on = 'State')

df_total_GDP_pri_sec_ter.head()

# Creting a new column to calculate the percentage contribution of primary

df_total_GDP_pri_sec_ter['%_Primary_Contribution'] =
(df_total_GDP_pri_sec_ter['Primary_GSVA']/df_total_GDP_pri_sec_ter['GSDP'])*100


# Creting a new column to calculate the percentage contribution of Secondary

df_total_GDP_pri_sec_ter['%_Secondary_Contribution'] =
(df_total_GDP_pri_sec_ter['Secondary_GSVA']/df_total_GDP_pri_sec_ter['GSDP'])*100


# Creting a new column to calculate the percentage contribution of Tertiary

df_total_GDP_pri_sec_ter['%_Tertiary_Contribution'] =
(df_total_GDP_pri_sec_ter['Tertiary_GSVA']/df_total_GDP_pri_sec_ter['GSDP'])*100


# Creting a new column to calculate the percentage contribution of all sectors

df_total_GDP_pri_sec_ter['Total_pri_sec_tri_%'] =
df_total_GDP_pri_sec_ter['%_Primary_Contribution']+df_total_GDP_pri_sec_ter['%_Secondary_Con
tribution']+df_total_GDP_pri_sec_ter['%_Tertiary_Contribution']
```

```
# Sorting the dataframe to keep it in order.

df_total_GDP_pri_sec_ter =
df_total_GDP_pri_sec_ter.sort_values(by='Total_pri_sec_tri_%',ascending=False)

df_total_GDP_pri_sec_ter.head()
```

Plot the percentage contribution of the primary, secondary and tertiary sectors as a percentage of the total GDP for all the states.

```
# Plotting a Stack bar-chart to represent the percentage contribution of primary, secondary and
tertiary sectors as a percentage of total GDP for all the states.

import numpy as np

Primary = df_total_GDP_pri_sec_ter['%_Primary_Contribution']

Secondary = df_total_GDP_pri_sec_ter['%_Secondary_Contribution']

Tertiary = df_total_GDP_pri_sec_ter['%_Tertiary_Contribution']

States = df_total_GDP_pri_sec_ter['State']   # the x locations for the groups

plt.figure(figsize=(14,6))

p1 = plt.bar(States, Primary)

p2 = plt.bar(States, Secondary, bottom=Primary)

p3 = plt.bar(States, Tertiary, bottom=np.array(Primary)+np.array(Secondary))

plt.ylabel('Total GSDP Percentage (%)')

plt.title('Percentage Contribution of Each sector to GSDP')

plt.xticks(States,rotation=90)

plt.yticks(np.arange(0, 110, 10)); plt.xlabel('All States Of India')

plt.legend((p1[0], p2[0], p3[0]), ('Primary', 'Secondary', 'Tertiary'))

plt.show()
```



# Creating a dataframe by selecting necessay column from df_all_states using:
loc[df_all_states.Item=='Per Capita GSDP (Rs.)'] and renaming the columns for convenince using: rename(columns = {'2014-15':'per_capita_GSDP'}

```
states_per_capita_sorted = df_all_states.loc[df_all_states.Item=='Per Capita GSDP (Rs.)'].sort_values(by='2014-15')[['2014-15','State']].rename(columns = {'2014-15':'per_capita_GSDP'})
```

```
states_per_capita_sorted.head()
```

Creating the categories C1, C2, C3, C4

```
q1 = round(27*0.20) # total sttes count in the given dataset is 27.
```

```
q2 = round(27*0.5)
```

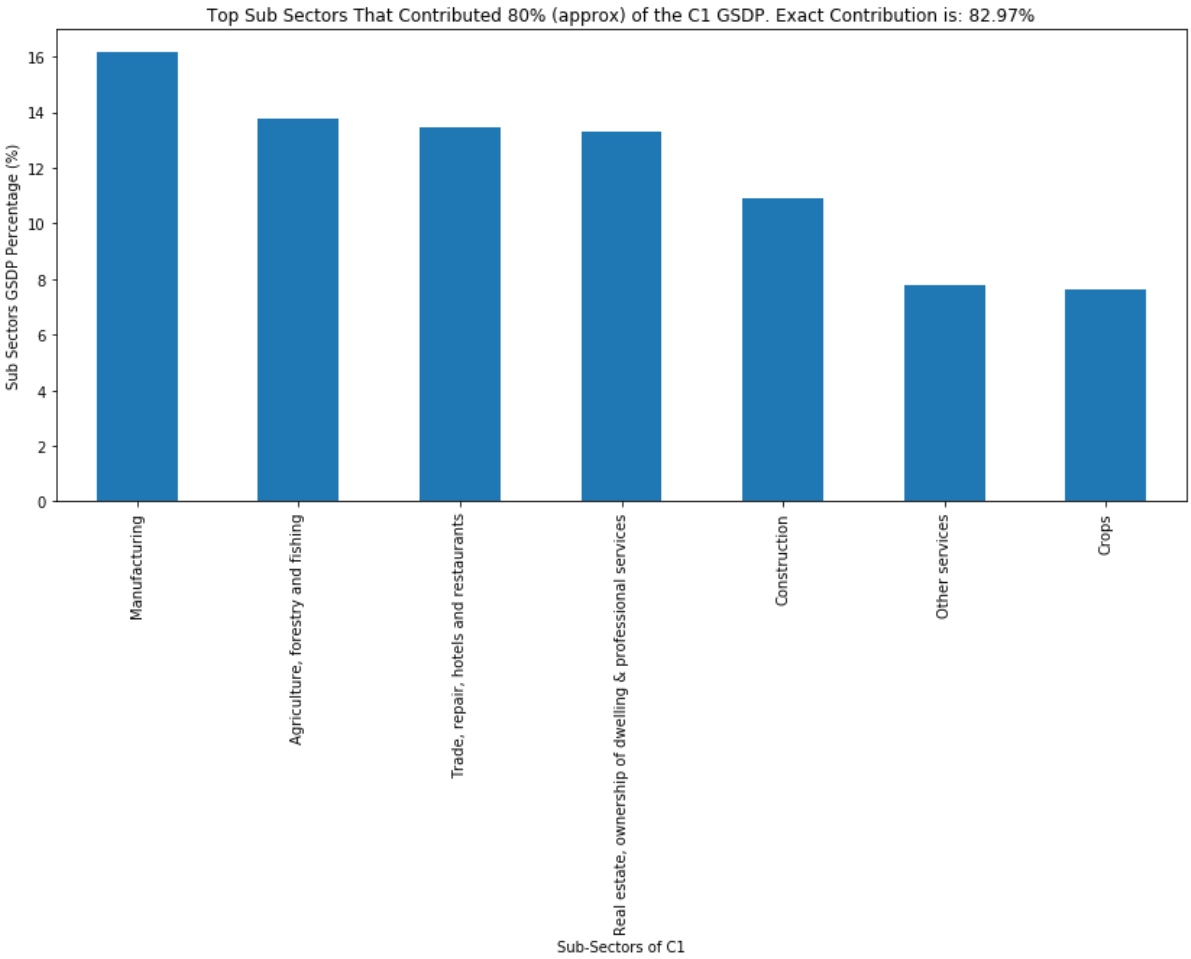
```
q3 = round(27*0.85)
```

```
q4 = round(27*1)
```

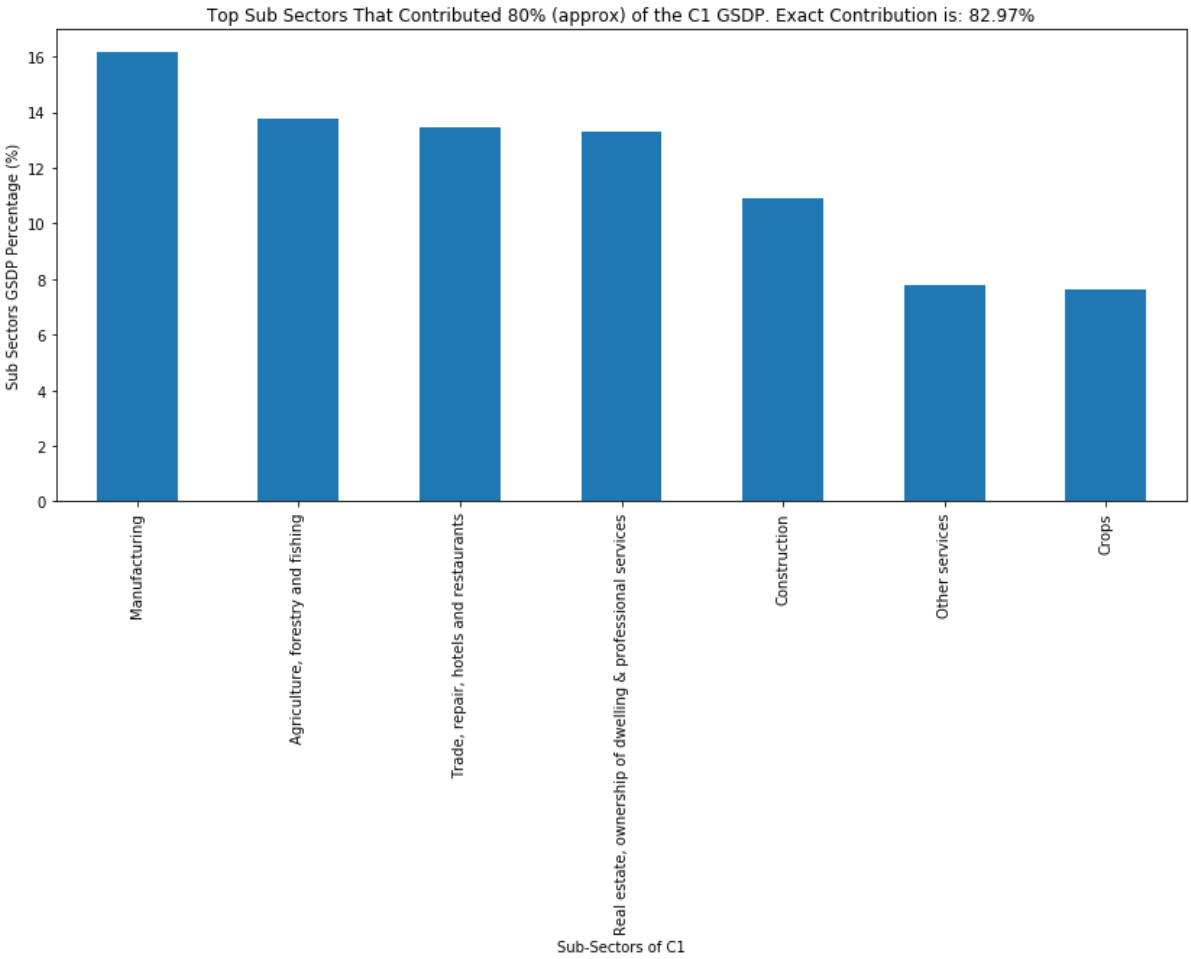
```
c4 = states_per_capita_sorted.iloc[:q1,:]
```

```
c3 = states_per_capita_sorted.iloc[q1:q2,:]

c2 = states_per_capita_sorted.iloc[q2:q3,:]

c1 = states_per_capita_sorted.iloc[q3:q4,:]
print(c4)
print(c3)
print(c2)
print(c1)

# Get all the fields that belong to the c1 states

df_C1 = df_all_states.loc[df_all_states.State.isin(c1.State)&(df_all_states['S.No.']!='Total')&
      (~df_all_states['Item'].isin(['TOTAL GSVA at basic prices','Taxes on Products','Subsidies on products',"Population ('00)",'Per Capita GSDP (Rs.)']))]

# Keeping only the necessary fields and grouping and sorting as required

df_C1 = df_C1[['Item','2014-15']].groupby(by='Item').sum().sort_values(by='2014-15',ascending=False).reset_index()

# Creating a new column Percentage_of_GSDP for easy analysis

df_C1['%_of_GSDP_Contribution'] = df_C1['2014-15']/(df_C1['2014-15'][0])*100
# here index index 0 has GSDP since we have sorted in descending order

# Finding which are the Sub-sectors that contribute approximately 80% to the GSDP (It should be 3 or more)

start =1; End = 4
# Takinfg first top 3 sectors initially to check whether it contributes approximately 80%. Starting with 1 to avoid first row which is GSDP
```

```
while df_C1.iloc[start:End ,-1].sum() <= 78:
#considering anything less than or equal to 78% does not contribute 80% approximately, only equal to greater than 79% does.
    End = End+1


# Contribution of subsectors approximately 80% For category C1 to the total GSDP are as follows


C1_Sub_Sectors_contributes_80_percent_apprx = df_C1[['Item','%_of_GSDP_Contribution']].iloc[start:End].append({'Item':'ABOVE C1 SUB-SECTORS EXACT CONTRIBUTION =','%_of_GSDP_Contribution':round(df_C1.iloc[start:End ,-1].sum(),2)},ignore_index=True).rename(columns={'Item':'C1_Sub_Sectors_that_contributes_80%_approximately_to_GSDP_in_Total'})


C1_Sub_Sectors_contributes_80_percent_apprx


Plot the contribution of the sub-sectors as a percentage of the GSDP of each category.
plt.figure(figsize=(14,6))


# Selecting the required rows and columns using: iloc[:-1,:]['%_of_GSDP_Contribution'] and plotting the graph using: plot(kind='bar')


C1_Sub_Sectors_contributes_80_percent_apprx.set_index("C1_Sub_Sectors_that_contributes_80%_approximately_to_GSDP_in_Total").iloc[:-1,:]['%_of_GSDP_Contribution'].plot(kind='bar')


plt.ylabel('Sub Sectors GSDP Percentage (%)')
plt.xlabel('Sub-Sectors of C1')


plt.title('Top Sub Sectors That Contributed 80% (approx) of the C1 GSDP. Exact Contribution is: {0}%'.format(C1_Sub_Sectors_contributes_80_percent_apprx.iloc[-1:,-1:].values[0][0]))


plt.show()
```




```
plt.figure(figsize=(14,6))

# Selecting the required rows and columns using: iloc[:-1,:]['%_of_GSDP_Contribution'] and plotting the graph using: plot(kind='bar')

C1_Sub_Sectors_contributes_80_percent_apprx.set_index("C1_Sub_Sectors_that_contributes_80%_approximately_to_GSDP_in_Total").iloc[:-1,:]['%_of_GSDP_Contribution'].plot(kind='bar')

plt.ylabel('Sub Sectors GSDP Percentage (%)')

plt.xlabel('Sub-Sectors of C1')

plt.title('Top Sub Sectors That Contributed 80% (approx) of the C1 GSDP. Exact Contribution is: {0}%'.format(C1_Sub_Sectors_contributes_80_percent_apprx.iloc[-1:,-1:].values[0][0]))

plt.show()
```



Get all the fields that belong to the c2 states

df_C2 = df_all_states.loc[df_all_states.State.isin(c2.State)&(df_all_states['S.No.']!='Total')&

(~df_all_states['Item'].isin(['TOTAL GSVA at basic prices','Taxes on Products','Subsidies on products',"Population ('00)",'Per Capita GSDP (Rs.)']))]

Keeping only the necessary fields and grouping and sorting as required

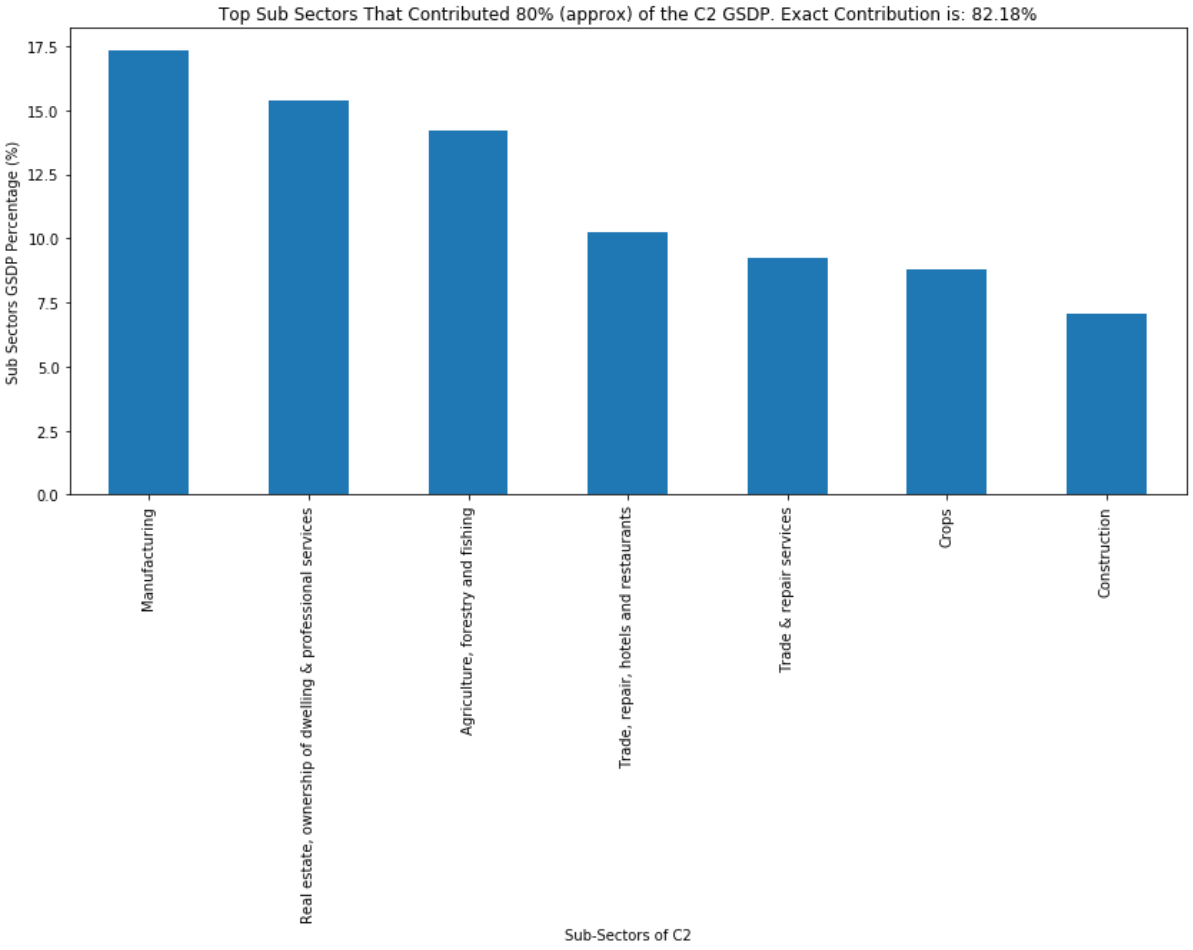
df_C2 = df_C2[['Item','2014-15']].groupby(by='Item').sum().sort_values(by='2014-15',ascending=False).reset_index()

Creating a new column Percentage_of_GSDP for easy analysis

df_C2['%_of_GSDP_Contribution'] = df_C2['2014-15']/(df_C2['2014-15'][0])*100 # here index index 0 has GSDP since we have sorted in descending order

```
# Finding which are the Sub-sectors that contribute approximately 80% to the GSDP (It should be 3 or more)


start =1; End = 4

# Takinfg first top 3 sectors initially to check whether it contributes approximately 80%. Starting with 1 to avoid first row which is GSDP


while df_C2.iloc[start:End ,-1].sum() <= 78:

#considering anything less than or equal to 78% does not contribute 80% approximately, only equal to greater than 79% does.

    End = End+1


# Contribution of subsectors approximately 80% For category C2 to the total GSDP are as follows


C2_Sub_Sectors_contributes_80_percent_apprx = df_C2[['Item','%_of_GSDP_Contribution']].iloc[start:End].append({'Item':'ABOVE C2 SUB-SECTORS EXACT CONTRIBUTION =','%_of_GSDP_Contribution':round(df_C2.iloc[start:End ,-1].sum(),2)},ignore_index=True).rename(columns={'Item':'C2_Sub_Sectors_that_contributes_80%_approximately_to_GSDP_in_Total'})


C2_Sub_Sectors_contributes_80_percent_apprx

plt.figure(figsize=(14,6))


# Selecting the required rows and columns using: iloc[:-1,:]['%_of_GSDP_Contribution'] and plotting the graph using: plot(kind='bar')


C2_Sub_Sectors_contributes_80_percent_apprx.set_index("C2_Sub_Sectors_that_contributes_80%_approximately_to_GSDP_in_Total").iloc[:-1,:]['%_of_GSDP_Contribution'].plot(kind='bar')


plt.ylabel('Sub Sectors GSDP Percentage (%)');plt.xlabel('Sub-Sectors of C2')


plt.title('Top Sub Sectors That Contributed 80% (approx) of the C2 GSDP. Exact Contribution is: {0}%'.format(C2_Sub_Sectors_contributes_80_percent_apprx.iloc[-1:,-1:].values[0][0]))
```

plt.show()



Get all the fields that belong to the c3 states

df_C3 = df_all_states.loc[df_all_states.State.isin(c3.State)&(df_all_states['S.No.']!='Total')&

(~df_all_states['Item'].isin(['TOTAL GSVA at basic prices','Taxes on Products','Subsidies on products',"Population ('00)",'Per Capita GSDP (Rs.)']))]

Keeping only the necessary fields and grouping and sorting as required

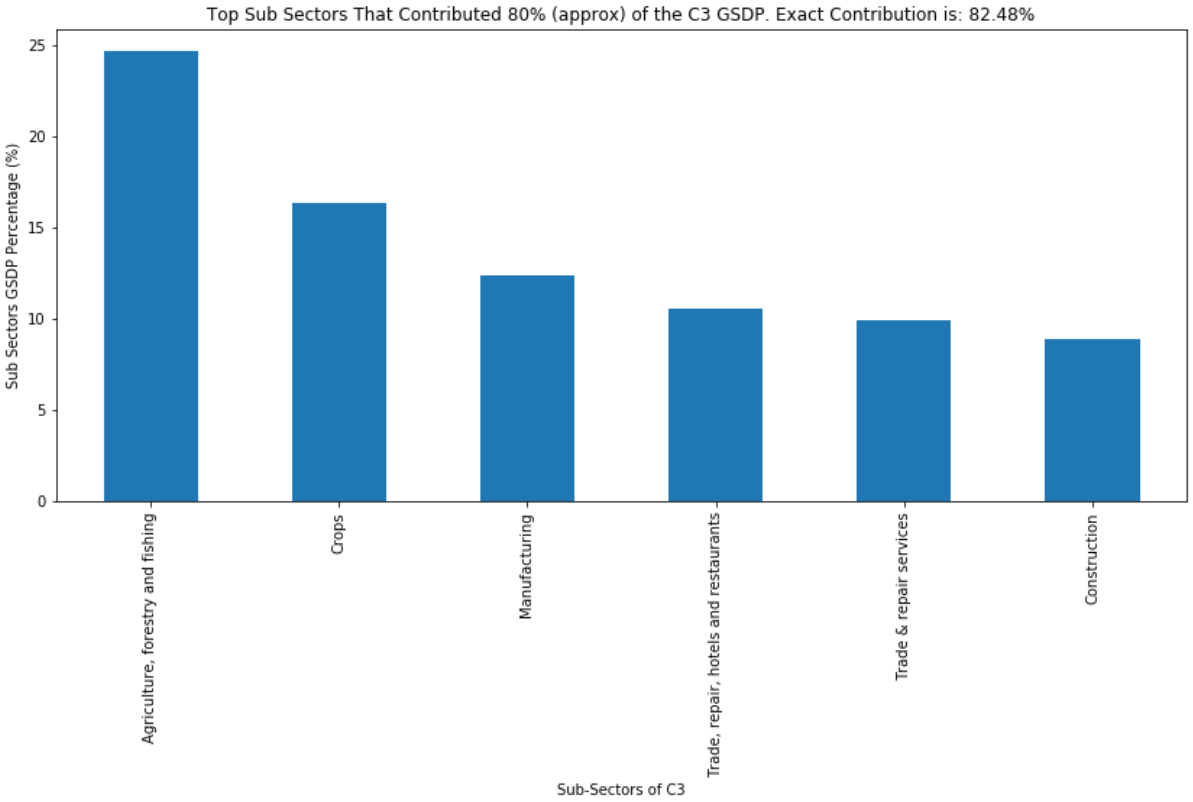
df_C3 = df_C3[['Item','2014-15']].groupby(by='Item').sum().sort_values(by='2014-15',ascending=False).reset_index()

Creating a new column Percentage_of_GSDP for easy analysis

df_C3['%_of_GSDP_Contribution'] = df_C3['2014-15']/(df_C3['2014-15'][0])*100 # here index index 0 has GSDP since we have sorted in descending order

```
# Finding which are the Sub-sectors that contribute approximately 80% to the GSDP (It should be 3 or more)


start =1; End = 4 # Takinfg first top 3 sectors initially to check whether it contributes approximately 80%. Starting with 1 to avoid first row which is GSDP


while df_C3.iloc[start:End ,-1].sum() <= 78: #considering anything less than or equal to 78% does not contribute 80% approximately, only equal to greater than 79% does.
    End = End+1


# Contribution of subsectors approximately 80% For category C3 to the total GSDP are as follows


C3_Sub_Sectors_contributes_80_percent_apprx = df_C3[['Item','%_of_GSDP_Contribution']].iloc[start:End].append({'Item':'ABOVE C3 SUB-SECTORS EXACT CONTRIBUTION =','%_of_GSDP_Contribution':round(df_C3.iloc[start:End ,-1].sum(),2)},ignore_index=True).rename(columns={'Item':'C3_Sub_Sectors_that_contributes_80%_approximately_to_GSDP_in_Total'})


C3_Sub_Sectors_contributes_80_percent_apprx


plt.figure(figsize=(14,6))


# Selecting the required rows and columns using: iloc[:-1,:]['%_of_GSDP_Contribution'] and plotting the graph using: plot(kind='bar')


C3_Sub_Sectors_contributes_80_percent_apprx.set_index("C3_Sub_Sectors_that_contributes_80%_approximately_to_GSDP_in_Total").iloc[:-1,:]['%_of_GSDP_Contribution'].plot(kind='bar')


plt.ylabel('Sub Sectors GSDP Percentage (%)'); plt.xlabel('Sub-Sectors of C3')


plt.title('Top Sub Sectors That Contributed 80% (approx) of the C3 GSDP. Exact Contribution is: {0}%'.format(C3_Sub_Sectors_contributes_80_percent_apprx.iloc[-1:,-1:].values[0][0]))
```

plt.show()



Get all the fields that belong to the c4 states

df_C4 = df_all_states.loc[df_all_states.State.isin(c4.State)&(df_all_states['S.No.']!='Total')&

(~df_all_states['Item'].isin(['TOTAL GSVA at basic prices','Taxes on Products','Subsidies on products',"Population ('00)",'Per Capita GSDP (Rs.)']))]

Keeping only the necessary fields and grouping and sorting as required

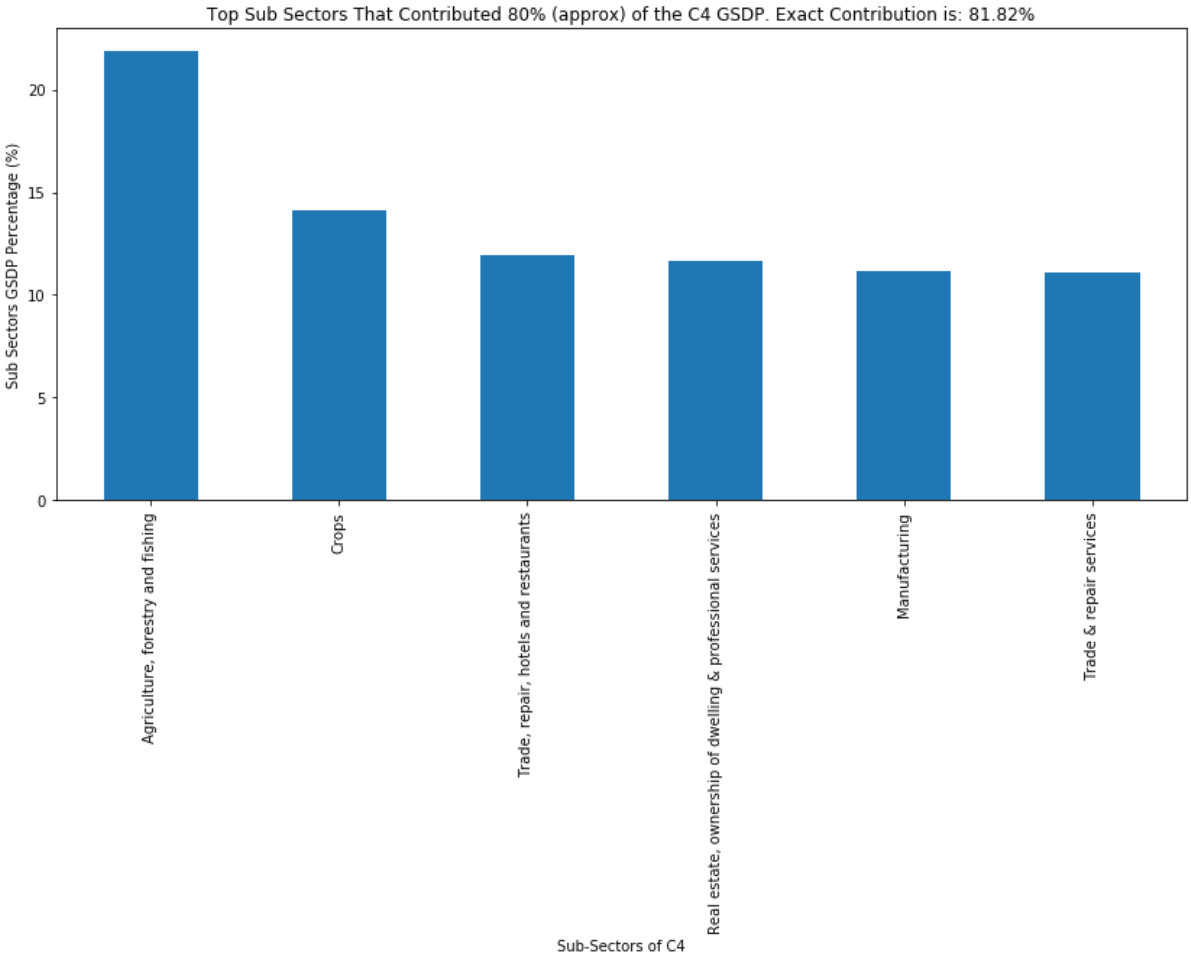
df_C4 = df_C4[['Item','2014-15']].groupby(by='Item').sum().sort_values(by='2014-15',ascending=False).reset_index()

Creating a new column Percentage_of_GSDP for easy analysis

df_C4['%_of_GSDP_Contribution'] = df_C4['2014-15']/(df_C4['2014-15'][0])*100 # here index index 0 has GSDP since we have sorted in descending order

```
# Finding which are the Sub-sectors that contribute approximately 80% to the GSDP (It should be 3 or more)


start =1; End = 4

 # Takinfg first top 3 sectors initially to check whether it contributes approximately 80%. Starting with 1 to avoid first row which is GSDP


while df_C4.iloc[start:End ,-1].sum() <= 78:

#considering anything less than or equal to 78% does not contribute 80% approximately, only equal to greater than 79% does.

    End = End+1


# Contribution of subsectors approximately 80% For category C4 to the total GSDP are as follows


C4_Sub_Sectors_contributes_80_percent_apprx = df_C4[['Item','%_of_GSDP_Contribution']].iloc[start:End].append({'Item':'ABOVE C4 SUB-SECTORS EXACT CONTRIBUTION =','%_of_GSDP_Contribution':round(df_C4.iloc[start:End ,-1].sum(),2)},ignore_index=True).rename(columns={'Item':'C4_Sub_Sectors_that_contributes_80%_approximately_to_GSDP_in_Total'})


C4_Sub_Sectors_contributes_80_percent_apprx

plt.figure(figsize=(14,6))


# Selecting the required rows and columns using: iloc[:-1,:]['%_of_GSDP_Contribution'] and plotting the graph using: plot(kind='bar')


C4_Sub_Sectors_contributes_80_percent_apprx.set_index("C4_Sub_Sectors_that_contributes_80%_approximately_to_GSDP_in_Total").iloc[:-1,:]['%_of_GSDP_Contribution'].plot(kind='bar')


plt.ylabel('Sub Sectors GSDP Percentage (%)'); plt.xlabel('Sub-Sectors of C4')


plt.title('Top Sub Sectors That Contributed 80% (approx) of the C4 GSDP. Exact Contribution is: {0}%'.format(C4_Sub_Sectors_contributes_80_percent_apprx.iloc[-1:,-1:].values[0][0]))
```

plt.show()



#Summarising and getting the insight:

#How does the GDP distribution of the top states (C1) differ from the others?

```
print('AVG of C1 GSDP: ',round(int(df_C1.iloc[0,1])/c1.shape[0]),', Per-capita AVG of C1 :',
round(c1['per_capita_GSDP'].mean()))
```

```
print('AVG of C2 GSDP: ',round(int(df_C2.iloc[0,1])/c2.shape[0]),', Per-capita AVG of C2 :',
round(c2['per_capita_GSDP'].mean()))
```

```
print('AVG of C3 GSDP: ',round(int(df_C3.iloc[0,1])/c3.shape[0]),', Per-capita AVG of C3 :',
round(c3['per_capita_GSDP'].mean()))
```

```
print('AVG of C4 GSDP: ',round(int(df_C4.iloc[0,1])/c4.shape[0]),', Per-capita AVG of C4 :',
round(c4['per_capita_GSDP'].mean()))
```

#From the above output we could clearly see that comparing to C1 average GSDP other categories average are Approximately 2 to 3 times (C4, C2) larger or equal (C3)

#We are taking the average GSDP here because the count of the states in each category is not the same.

#Which sub-sectors seem to be correlated with high GDP?

```
df_all_states[['Item','2014-15']].groupby('Item').sum().sort_values(by = '2014-15', ascending=False).head(11)
```

#From the above groupby chart (based on all state GSDP) we could see that the sub-sectors (max GDP contributors) that are correlated to high GDP are:

1. Agri,forestry, fishing

2. Manufacturing

3. real estate..

4. Trade, repair, hotels..

#Which sub-sectors do the various categories need to focus on?

Sub-sectors that needs to be concentrated are the one's that are contributing lowest to the GSDP.

```
print('Sub-sectors to be concentrated for C1: ',df_C1['Item'].tail().values,'\n')
```

```
print('Sub-sectors to be concentrated for C2: ',df_C2['Item'].tail().values, '\n')
```

```
print('Sub-sectors to be concentrated for C3: ',df_C3['Item'].tail().values, '\n')
```

```
print('Sub-sectors to be concentrated for C4: ',df_C4['Item'].tail().values, '\n')
```

Note: Sub-sectors in the dataframes df_C1,df_C2,df_C3,df_C4 is already aranged in descending order based on GSDP contribution

#As shown above, even the sub-sectors which are very huge is also contributiong less. example: transportation (railways, water-transport, road transport, Air transport). One main reason for the very high potential sub-sectors contributing less to GSDP is because of the heavy loss that is happening in these sub-sectors which needs to be facused and arrested immediately. (loss can be of any reasons which needs to be investigated).

#For each of the categories, as mentioned in the question 3 above, we need to concentrate on the sub-sectors which are having very high potential (example transportation) and are contributing very less to GSDP.

#Focus on Primary (having high potential) which are contributing less to GDP, in understanding the losses happening in these sectors and arresting them to increase its GDP contribution

#Focusing on Tertiary which are proven to have very high profits, to take it to the next level in boosting its profit.

Part-II: GDP and Education Drop-out Rates

#Analyse if there is any correlation of GDP per capita with dropout rates in education (primary, upper primary and secondary) for the year 2014-2015 for the states. Choose an appropriate plot to conduct this analysis.

```
df_drp_out = pd.read_csv('Data 2 dropout rates.csv')
```

```
df_drp_out.head()
```

As seen above there are two similar columns "Primary - 2014-2015","Primary - 2014-2015.1"

This is because both these columns have same name in the given data set.

So changing the names of columns accordingly as: 'Primary - 2013-2014' and 'Primary - 2014-2015'

We are alos changing the column name: "Level of Education - State" as "State" for convenience.

```
df_drp_out = df_drp_out.rename(columns = {'Primary - 2014-2015':'Primary - 2013-2014','Primary - 2014-2015.1':'Primary - 2014-2015','Level of Education - State':'State'})
```

```
df_drp_out
```

considering only for the year 2014-15 and for the class: primary, upper primary and secondary, as requested.

```
df_drp_out = df_drp_out[['State','Primary - 2014-2015','Upper Primary - 2014-2015','Secondary - 2014-2015']]
```

```
df_drp_out.head()
```

Dropping the states that are having null values. Here we are not using fillna command to fill the mean, median or mode because there is a hude difference between the dropout rates between each states.

```
df_drp_out = df_drp_out.dropna(how='any')
```

```
df_drp_out
```

In above States other than union territories and the deleted states that had Null Values we have two states with wrong name (Uttrakhand and Chhatisgarh)

We are going to correct the names to Uttarakhand and Chhattisgarh. If not while merging the columns with the part-1 df which had per-capita values, we will miss these 2 rows.

```
df_drp_out = df_drp_out.replace(['Chhatisgarh','Uttrakhand'],['Chhattisgarh','Uttarakhand'])

df_drp_out

# Merging two dataframes to get the per-capita-GSDP and the dropout rate in the same frame.

df_drpout_percap = pd.merge(df_all_states[df_all_states.Item=='Per Capita GSDP (Rs.)'],
df_drp_out, how = 'inner', on = 'State')

df_drpout_percap

# Adding a new column in above df: 'Total_dropout_in_2014-15'

df_drpout_percap['Total_dropout_in_2014-15'] = df_drpout_percap.iloc[:,-3:].sum(axis = 1)

df_drpout_percap


# Plotting


x = df_drpout_percap['2014-15'].values # per-capita GSDP


y1 = df_drpout_percap['Primary - 2014-2015'].values # primary dropout


y2 = df_drpout_percap['Upper Primary - 2014-2015'].values # upper primary dropout


y3 = df_drpout_percap['Secondary - 2014-2015'].values # Secondary dropout


y4 = df_drpout_percap['Total_dropout_in_2014-15'].values # Total_dropout_in_2014-15


plt.figure(figsize=(14,12))


plt.subplot(221)


plt.title('GSDP vs Dropout Rate in Primary During 2014-2015')


plt.xlabel('GSDP in Crores (Rs.)')


plt.ylabel('Dropout rate in Percentage')
```

```
plt.scatter(x,y2)

plt.subplot(222)

plt.title('GSDP vs Dropout Rate in Upper-Primary During 2014-2015')

plt.xlabel('GSDP in Crores (Rs.)')

plt.ylabel('Dropout rate in Percentage')

plt.scatter(x,y3)

plt.subplot(223)

plt.title('GSDP vs Dropout Rate in Secondary During 2014-2015')

plt.xlabel('GSDP in Crores (Rs.)')

plt.ylabel('Dropout rate in Percentage')

plt.scatter(x,y3)

plt.subplot(224)

plt.title('GSDP vs Total Dropout Rate During 2014-2015')

plt.xlabel('GSDP in Crores (Rs.)')

plt.ylabel('Dropout rate in Percentage')
```

plt.scatter(x,y4)

plt.show()