Importing required libraries

In [146…

```
# Libraries to help with reading and manipulating data
import pandas as pd
import numpy as np

# libaries to help with data visualization
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.patches import Rectangle

# Removes the limit for the number of displayed columns
pd.set_option("display.max_columns", None)

# Sets the limit for the number of displayed rows
pd.set_option("display.max_rows", 200)

# to scale the data using z-score
from sklearn.preprocessing import StandardScaler

# to perform statistical tests before PCA
from factor_analyzer import FactorAnalyzer

# to perform PCA
from sklearn.decomposition import PCA

# to suppress warnings
import warnings
warnings.filterwarnings("ignore")
```

Problem Statement:

PCA:

PCA FH (FT): Primary census abstract for female headed households excluding institutional households (India & States/UTs - District Level), Scheduled tribes - 2011 PCA for Female Headed Household Excluding Institutional Household. The Indian Census has the reputation of being one of the best in the world. The first Census in India was conducted in the year 1872. This was conducted at different points of time in different parts of the country. In 1881 a Census was taken for the entire country simultaneously. Since then, Census has been conducted every ten years, without a break. Thus, the Census of India 2011 was the fifteenth in this unbroken series since 1872, the seventh after independence and the second census of the third millennium and twenty first century. The census has been uninterruptedly continued despite of several adversities like wars, epidemics, natural calamities, political unrest, etc. The Census of India is conducted under the provisions of the Census Act 1948 and the Census Rules, 1990. The Primary Census Abstract which is important publication of 2011 Census gives basic information on Area, Total Number of Households, Total Population,

Scheduled Castes, Scheduled Tribes Population, Population in the age group 0-6, Literates, Main Workers and Marginal Workers classified by the four broad industrial categories, namely, (i) Cultivators, (ii) Agricultural Laborers, (iii) Household Industry Workers, and (iv) Other Workers and also Non-Workers. The characteristics of the Total Population include Scheduled Castes, Scheduled Tribes, Institutional and Houseless Population and are presented by sex and rural-urban residence. Census 2011 covered 35 States/Union Territories, 640 districts, 5,924 sub-districts, 7,935 Towns and 6,40,867 Villages.
The data collected has so many variables thus making it difficult to find useful details without using Data Science Techniques. You are tasked to perform detailed EDA and identify Optimum Principal Components that explains the most variance in data. Use Sklearn only.

Data Dictionary

State Code: State Code
Dist.Code: District Code
State: State Name
Area Name: Area Name
No_HH: No of Household
TOT_M: Total population Male
TOT_F: Total population Female
M_06: Population in the age group 0-6 Male
F_06: Population in the age group 0-6 Female
M_SC: Scheduled Castes population Male
F_SC: Scheduled Castes population Female
M_ST: Scheduled Tribes population Male
F_ST: Scheduled Tribes population Female
M_LIT: Literates population Male
F_LIT: Literates population Female
M_ILL: Illiterate Male
F_ILL: Illiterate Female
TOT_WORK_M: Total Worker Population Male
TOT_WORK_F: Total Worker Population Female
MAINWORK_M: Main Working Population Male
MAINWORK_F: Main Working Population Female
MAIN_CL_M: Main Cultivator Population Male
MAIN_CL_F: Main Cultivator Population Female
MAIN_AL_M: Main Agricultural Labourers Population Male
MAIN_AL_F:: Main Agricultural Labourers Population Female
MAIN_HH_M: Main Household Industries Population Male
MAIN_HH_F: Main Household Industries Population Female
MAIN_OT_M: Main Other Workers Population Male
MAIN_OT_F: Main Other Workers Population Female

MARGWORK_M: Marginal Worker Population Male
MARGWORK_F: Marginal Worker Population Female
MARG_CL_M: Marginal Cultivator Population Male
MARG_CL_F: Marginal Cultivator Population Female
MARG_AL_M: Marginal Agriculture Labourers Population Male
MARG_AL_F: Marginal Agriculture Labourers Population Female
MARG_HH_M: Marginal Household Industries Population Male
MARG_HH_F: Marginal Household Industries Population Female
MARG_OT_M: Marginal Other Workers Population Male
MARG_OT_F: Marginal Other Workers Population Female
MARGWORK_3_6_M: Marginal Worker Population 3-6 Male
MARGWORK_3_6_F: Marginal Worker Population 3-6 Female
MARG_CL_3_6_M: Marginal Cultivator Population 3-6 Male
MARG_CL_3_6_F: Marginal Cultivator Population 3-6 Female
MARG_AL_3_6_M: Marginal Agriculture Labourers Population 3-6 Male
MARG_AL_3_6_F: Marginal Agriculture Labourers Population 3-6 Female
MARG_HH_3_6_M: Marginal Household Industries Population 3-6 Male
MARG_HH_3_6_F: Marginal Household Industries Population 3-6 Female
MARG_OT_3_6_M: Marginal Other Workers Population Person 3-6 Male
MARG_OT_3_6_F: Marginal Other Workers Population Person 3-6 Female
MARGWORK_0_3_M: Marginal Worker Population 0-3 Male
MARGWORK_0_3_F: Marginal Worker Population 0-3 Female
MARG_CL_0_3_M: Marginal Cultivator Population 0-3 Male
MARG_CL_0_3_F: Marginal Cultivator Population 0-3 Female
MARG_AL_0_3_M: Marginal Agriculture Labourers Population 0-3 Male
MARG_AL_0_3_F: Marginal Agriculture Labourers Population 0-3 Female
MARG_HH_0_3_M: Marginal Household Industries Population 0-3 Male
MARG_HH_0_3_F: Marginal Household Industries Population 0-3 Female
MARG_OT_0_3_M: Marginal Other Workers Population 0-3 Male
MARG_OT_0_3_F: Marginal Other Workers Population 0-3 Female
NON_WORK_M: Non Working Population Male
NON_WORK_F: Non Working Population Female

Understanding the structure of data

In [147…
```
df_pca = pd.read_excel('PCA+India+Data_Census.xlsx')
```

In [148…
```
df_pca.head() # Returns first 5 rows
```

Out[148…

	State Code	Dist.Code	State	Area Name	No_HH	TOT_M	TOT_F	M_06	F_06	M_SC	F_S
0	1	1	Jammu & Kashmir	Kupwara	7707	23388	29796	5862	6196	3	
1	1	2	Jammu & Kashmir	Badgam	6218	19585	23102	4482	3733	7	
2	1	3	Jammu & Kashmir	Leh(Ladakh)	4452	6546	10964	1082	1018	3	
3	1	4	Jammu & Kashmir	Kargil	1320	2784	4206	563	677	0	
4	1	5	Jammu & Kashmir	Punch	11654	20591	29981	5157	4587	20	3

In [149…

```
df_pca.tail() # Returns last 5 rows
```

Out[149…

	State Code	Dist.Code	State	Area Name	No_HH	TOT_M	TOT_F	M_06	F_06	M_SC
635	34	636	Puducherry	Mahe	3333	8154	11781	1146	1203	21
636	34	637	Puducherry	Karaikal	10612	12346	21691	1544	1533	2234
637	35	638	Andaman & Nicobar Island	Nicobars	1275	1549	2630	227	225	0
638	35	639	Andaman & Nicobar Island	North & Middle Andaman	3762	5200	8012	723	664	0
639	35	640	Andaman & Nicobar Island	South Andaman	7975	11977	18049	1470	1358	0

Number of rows and columns in the dataset

In [150…

```
# checking shape of the data

rows = str(df_pca.shape[0])
columns = str(df_pca.shape[1])

print(f"There are \033[1m" + rows + "\033[0m rows and \033[1m" + columns + "\033[0m
```

There are **640** rows and **61** columns in the dataset.

Datatypes of the different columns in the dataset

In [151…

```
df_pca.info() # Concise summary of dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 640 entries, 0 to 639
Data columns (total 61 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   State Code      640 non-null    int64 
 1   Dist.Code       640 non-null    int64 
 2   State           640 non-null    object
 3   Area Name       640 non-null    object
 4   No_HH           640 non-null    int64 
 5   TOT_M           640 non-null    int64 
 6   TOT_F           640 non-null    int64 
 7   M_06            640 non-null    int64 
 8   F_06            640 non-null    int64 
 9   M_SC            640 non-null    int64 
 10  F_SC            640 non-null    int64 
 11  M_ST            640 non-null    int64 
 12  F_ST            640 non-null    int64 
 13  M_LIT           640 non-null    int64 
 14  F_LIT           640 non-null    int64 
 15  M_ILL           640 non-null    int64 
 16  F_ILL           640 non-null    int64 
 17  TOT_WORK_M      640 non-null    int64 
 18  TOT_WORK_F      640 non-null    int64 
 19  MAINWORK_M      640 non-null    int64 
 20  MAINWORK_F      640 non-null    int64 
 21  MAIN_CL_M       640 non-null    int64 
 22  MAIN_CL_F       640 non-null    int64 
 23  MAIN_AL_M       640 non-null    int64 
 24  MAIN_AL_F       640 non-null    int64 
 25  MAIN_HH_M       640 non-null    int64 
 26  MAIN_HH_F       640 non-null    int64 
 27  MAIN_OT_M       640 non-null    int64 
 28  MAIN_OT_F       640 non-null    int64 
 29  MARGWORK_M      640 non-null    int64 
 30  MARGWORK_F      640 non-null    int64 
 31  MARG_CL_M       640 non-null    int64 
 32  MARG_CL_F       640 non-null    int64 
 33  MARG_AL_M       640 non-null    int64 
 34  MARG_AL_F       640 non-null    int64 
 35  MARG_HH_M       640 non-null    int64 
 36  MARG_HH_F       640 non-null    int64 
 37  MARG_OT_M       640 non-null    int64 
 38  MARG_OT_F       640 non-null    int64 
 39  MARGWORK_3_6_M  640 non-null    int64 
 40  MARGWORK_3_6_F  640 non-null    int64 
 41  MARG_CL_3_6_M   640 non-null    int64 
 42  MARG_CL_3_6_F   640 non-null    int64 
 43  MARG_AL_3_6_M   640 non-null    int64 
 44  MARG_AL_3_6_F   640 non-null    int64 
 45  MARG_HH_3_6_M   640 non-null    int64 
 46  MARG_HH_3_6_F   640 non-null    int64 
 47  MARG_OT_3_6_M   640 non-null    int64 
 48  MARG_OT_3_6_F   640 non-null    int64 
 49  MARGWORK_0_3_M  640 non-null    int64 
 50  MARGWORK_0_3_F  640 non-null    int64 
```

```
 51  MARG_CL_0_3_M   640 non-null    int64
 52  MARG_CL_0_3_F   640 non-null    int64
 53  MARG_AL_0_3_M   640 non-null    int64
 54  MARG_AL_0_3_F   640 non-null    int64
 55  MARG_HH_0_3_M   640 non-null    int64
 56  MARG_HH_0_3_F   640 non-null    int64
 57  MARG_OT_0_3_M   640 non-null    int64
 58  MARG_OT_0_3_F   640 non-null    int64
 59  NON_WORK_M      640 non-null    int64
 60  NON_WORK_F      640 non-null    int64
dtypes: int64(59), object(2)
memory usage: 305.1+ KB
```

There are 61 columns in the dataset. Out of which 2 have object data type and 59 have integer data type.

Finding missing values in the dataset

In [152…

```
df_pca.isna().sum()  # Count NaN values in all columns of dataset
```

```
Out[152…  State Code        0
          Dist.Code         0
          State             0
          Area Name         0
          No_HH             0
          TOT_M             0
          TOT_F             0
          M_06              0
          F_06              0
          M_SC              0
          F_SC              0
          M_ST              0
          F_ST              0
          M_LIT             0
          F_LIT             0
          M_ILL             0
          F_ILL             0
          TOT_WORK_M        0
          TOT_WORK_F        0
          MAINWORK_M        0
          MAINWORK_F        0
          MAIN_CL_M         0
          MAIN_CL_F         0
          MAIN_AL_M         0
          MAIN_AL_F         0
          MAIN_HH_M         0
          MAIN_HH_F         0
          MAIN_OT_M         0
          MAIN_OT_F         0
          MARGWORK_M        0
          MARGWORK_F        0
          MARG_CL_M         0
          MARG_CL_F         0
          MARG_AL_M         0
          MARG_AL_F         0
          MARG_HH_M         0
          MARG_HH_F         0
          MARG_OT_M         0
          MARG_OT_F         0
          MARGWORK_3_6_M    0
          MARGWORK_3_6_F    0
          MARG_CL_3_6_M     0
          MARG_CL_3_6_F     0
          MARG_AL_3_6_M     0
          MARG_AL_3_6_F     0
          MARG_HH_3_6_M     0
          MARG_HH_3_6_F     0
          MARG_OT_3_6_M     0
          MARG_OT_3_6_F     0
          MARGWORK_0_3_M    0
          MARGWORK_0_3_F    0
          MARG_CL_0_3_M     0
          MARG_CL_0_3_F     0
          MARG_AL_0_3_M     0
          MARG_AL_0_3_F     0
          MARG_HH_0_3_M     0
```

```
MARG_HH_0_3_F      0
MARG_OT_0_3_M      0
MARG_OT_0_3_F      0
NON_WORK_M         0
NON_WORK_F         0
dtype: int64
```

There are no missing values in the dataset.

Checking for Duplicates

In [153…

```
df_pca.duplicated().sum()
```

Out[153…

```
0
```

There are no duplicate rows in the dataset.

Checking Summary Statistic

In [154…

```
df_pca.describe(include='all').T
```

Out[154…

	count	unique	top	freq	mean	std	min
State Code	640.0	NaN	NaN	NaN	17.114062	9.426486	1.0
Dist.Code	640.0	NaN	NaN	NaN	320.5	184.896367	1.0
State	640	35	Uttar Pradesh	71	NaN	NaN	NaN
Area Name	640	635	Raigarh	2	NaN	NaN	NaN
No_HH	640.0	NaN	NaN	NaN	51222.871875	48135.405475	350.0
TOT_M	640.0	NaN	NaN	NaN	79940.576563	73384.511114	391.0
TOT_F	640.0	NaN	NaN	NaN	122372.084375	113600.717282	698.0
M_06	640.0	NaN	NaN	NaN	12309.098438	11500.906881	56.0
F_06	640.0	NaN	NaN	NaN	11942.3	11326.294567	56.0
M_SC	640.0	NaN	NaN	NaN	13820.946875	14426.37313	0.0
F_SC	640.0	NaN	NaN	NaN	20778.392188	21727.887713	0.0
M_ST	640.0	NaN	NaN	NaN	6191.807813	9912.668948	0.0
F_ST	640.0	NaN	NaN	NaN	10155.640625	15875.701488	0.0
M_LIT	640.0	NaN	NaN	NaN	57967.979688	55910.282466	286.0
F_LIT	640.0	NaN	NaN	NaN	66359.565625	75037.860207	371.0
M_ILL	640.0	NaN	NaN	NaN	21972.596875	19825.605268	105.0
F_ILL	640.0	NaN	NaN	NaN	56012.51875	47116.693769	327.0
TOT_WORK_M	640.0	NaN	NaN	NaN	37992.407813	36419.537491	100.0
TOT_WORK_F	640.0	NaN	NaN	NaN	41295.760938	37192.360943	357.0
MAINWORK_M	640.0	NaN	NaN	NaN	30204.446875	31480.91568	65.0
MAINWORK_F	640.0	NaN	NaN	NaN	28198.846875	29998.262689	240.0
MAIN_CL_M	640.0	NaN	NaN	NaN	5424.342188	4739.161969	0.0
MAIN_CL_F	640.0	NaN	NaN	NaN	5486.042188	5326.362728	0.0
MAIN_AL_M	640.0	NaN	NaN	NaN	5849.109375	6399.507966	0.0
MAIN_AL_F	640.0	NaN	NaN	NaN	8925.995312	12864.287584	0.0
MAIN_HH_M	640.0	NaN	NaN	NaN	883.89375	1278.642345	0.0
MAIN_HH_F	640.0	NaN	NaN	NaN	1380.773438	3179.414449	0.0
MAIN_OT_M	640.0	NaN	NaN	NaN	18047.101562	26068.480886	36.0
MAIN_OT_F	640.0	NaN	NaN	NaN	12406.035938	18972.202369	153.0
MARGWORK_M	640.0	NaN	NaN	NaN	7787.960938	7410.791691	35.0

	count	unique	top	freq	mean	std	min	
MARGWORK_F	640.0	NaN	NaN	NaN	13096.914062	10996.474528	117.0	
MARG_CL_M	640.0	NaN	NaN	NaN	1040.7375	1311.546847	0.0	
MARG_CL_F	640.0	NaN	NaN	NaN	2307.682813	3564.626095	0.0	
MARG_AL_M	640.0	NaN	NaN	NaN	3304.326562	3781.555707	0.0	
MARG_AL_F	640.0	NaN	NaN	NaN	6463.28125	6773.876298	0.0	
MARG_HH_M	640.0	NaN	NaN	NaN	316.742188	462.661891	0.0	
MARG_HH_F	640.0	NaN	NaN	NaN	786.626562	1198.718213	0.0	
MARG_OT_M	640.0	NaN	NaN	NaN	3126.154687	3609.391821	7.0	
MARG_OT_F	640.0	NaN	NaN	NaN	3539.323438	4115.191314	19.0	1
MARGWORK_3_6_M	640.0	NaN	NaN	NaN	41948.16875	39045.316918	291.0	16
MARGWORK_3_6_F	640.0	NaN	NaN	NaN	81076.323438	82970.406216	341.0	2
MARG_CL_3_6_M	640.0	NaN	NaN	NaN	6394.9875	6019.806644	27.0	
MARG_CL_3_6_F	640.0	NaN	NaN	NaN	10339.864063	8467.473429	85.0	
MARG_AL_3_6_M	640.0	NaN	NaN	NaN	789.848438	905.639279	0.0	
MARG_AL_3_6_F	640.0	NaN	NaN	NaN	1749.584375	2496.541514	0.0	
MARG_HH_3_6_M	640.0	NaN	NaN	NaN	2743.635938	3059.586387	0.0	
MARG_HH_3_6_F	640.0	NaN	NaN	NaN	5169.85	5335.64096	0.0	1
MARG_OT_3_6_M	640.0	NaN	NaN	NaN	245.3625	358.728567	0.0	
MARG_OT_3_6_F	640.0	NaN	NaN	NaN	585.884375	900.025817	0.0	
MARGWORK_0_3_M	640.0	NaN	NaN	NaN	2616.140625	3036.964381	7.0	
MARGWORK_0_3_F	640.0	NaN	NaN	NaN	2834.545312	3327.836932	14.0	
MARG_CL_0_3_M	640.0	NaN	NaN	NaN	1392.973438	1489.707052	4.0	
MARG_CL_0_3_F	640.0	NaN	NaN	NaN	2757.05	2788.776676	30.0	
MARG_AL_0_3_M	640.0	NaN	NaN	NaN	250.889062	453.336594	0.0	
MARG_AL_0_3_F	640.0	NaN	NaN	NaN	558.098438	1117.642748	0.0	
MARG_HH_0_3_M	640.0	NaN	NaN	NaN	560.690625	762.578991	0.0	
MARG_HH_0_3_F	640.0	NaN	NaN	NaN	1293.43125	1585.377936	0.0	
MARG_OT_0_3_M	640.0	NaN	NaN	NaN	71.379688	107.897627	0.0	
MARG_OT_0_3_F	640.0	NaN	NaN	NaN	200.742188	309.740854	0.0	
NON_WORK_M	640.0	NaN	NaN	NaN	510.014063	610.603187	0.0	

	count	unique	top	freq	mean	std	min
NON_WORK_F	640.0	NaN	NaN	NaN	704.778125	910.209225	5.0

Observations and Insights:

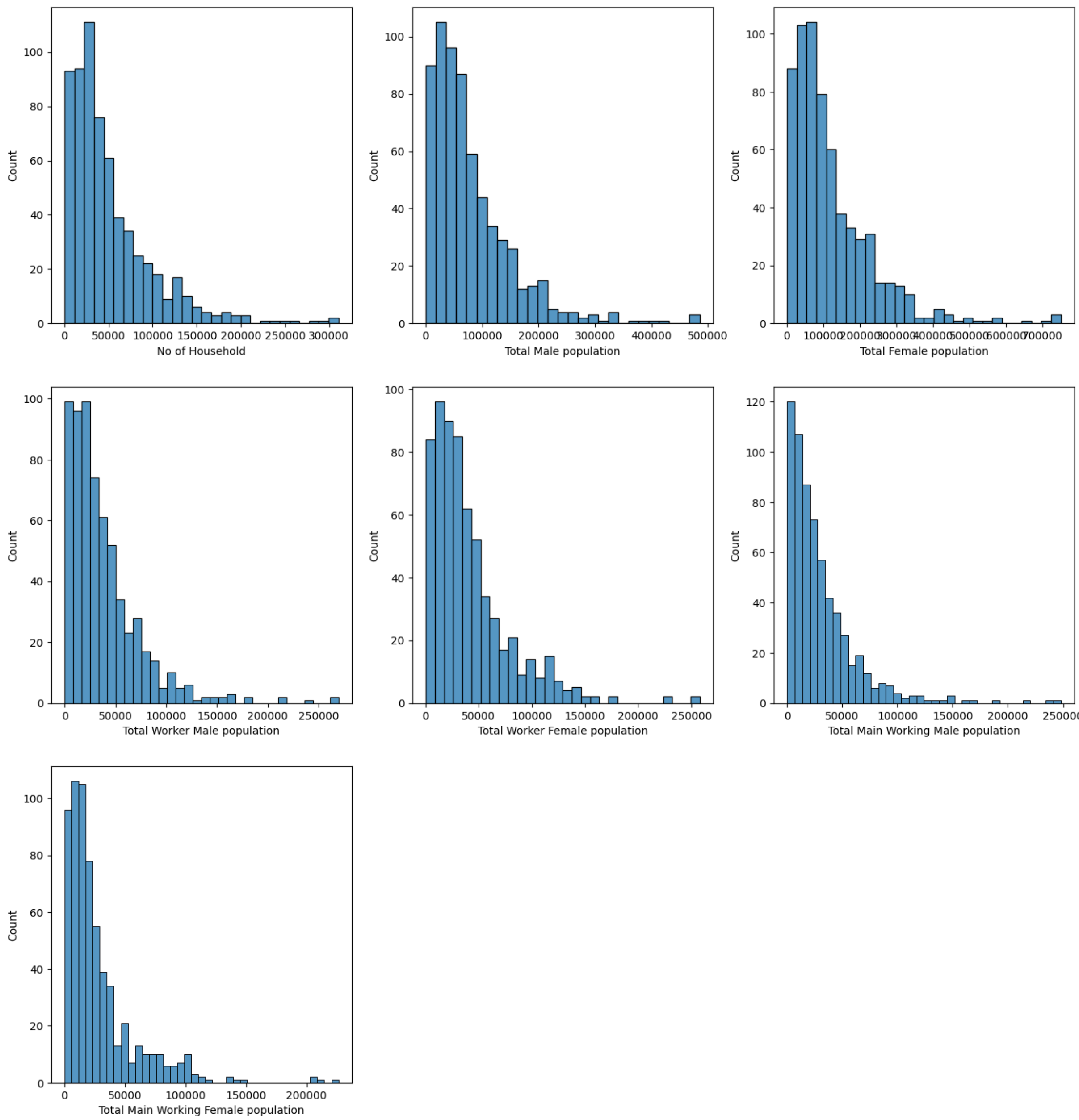
1. There are 35 unique States in the dataset.
2. There are 635 unique Area Names in the dataset.
3. Uttar Pradesh State has the greatest number of Area Names.
4. Raigarh Area Name is common in 2 States.
5. Female population is more than the Male population in the dataset.
6. There are more Female workers than Male workers in the dataset.
7. There are more Male main workers than Female main workers in the dataset.
8. There are more Female marginal workers than Male marginal workers in the dataset.
9. There are more Female non-workers than Males non-workers in the dataset.

Univariate analysis

In [155…

```
# Hist Plots for No_HH, TOT_M, TOT_F, TOT_WORK_M, TOT_WORK_F, MAINWORK_M, MAINWORK_

fig, axes = plt.subplots(3,3, figsize=(17, 18))

sns.histplot(ax=axes[0, 0], data=df_pca, x='No_HH')
sns.histplot(ax=axes[0, 1], data=df_pca, x='TOT_M')
sns.histplot(ax=axes[0, 2], data=df_pca, x='TOT_F')
sns.histplot(ax=axes[1, 0], data=df_pca, x='TOT_WORK_M')
sns.histplot(ax=axes[1, 1], data=df_pca, x='TOT_WORK_F')
sns.histplot(ax=axes[1, 2], data=df_pca, x='MAINWORK_M')
sns.histplot(ax=axes[2, 0], data=df_pca, x='MAINWORK_F')
axes[2,1].axis("off")
axes[2,2].axis("off")

axes[0,0].set(xlabel='No of Household')
axes[0,1].set(xlabel='Total Male population')
axes[0,2].set(xlabel='Total Female population')
axes[1,0].set(xlabel='Total Worker Male population')
axes[1,1].set(xlabel='Total Worker Female population')
axes[1,2].set(xlabel='Total Main Working Male population')
axes[2,0].set(xlabel='Total Main Working Female population')

plt.suptitle('Fig 1: Hist Plots: No_HH, TOT_M, TOT_F, TOT_WORK_M, TOT_WORK_F, MAINW

plt.show()
```

Fig 1: Hist Plots: No_HH, TOT_M, TOT_F, TOT_WORK_M, TOT_WORK_F, MAINWORK_M, MAINWORK_F



Observations and Insights:

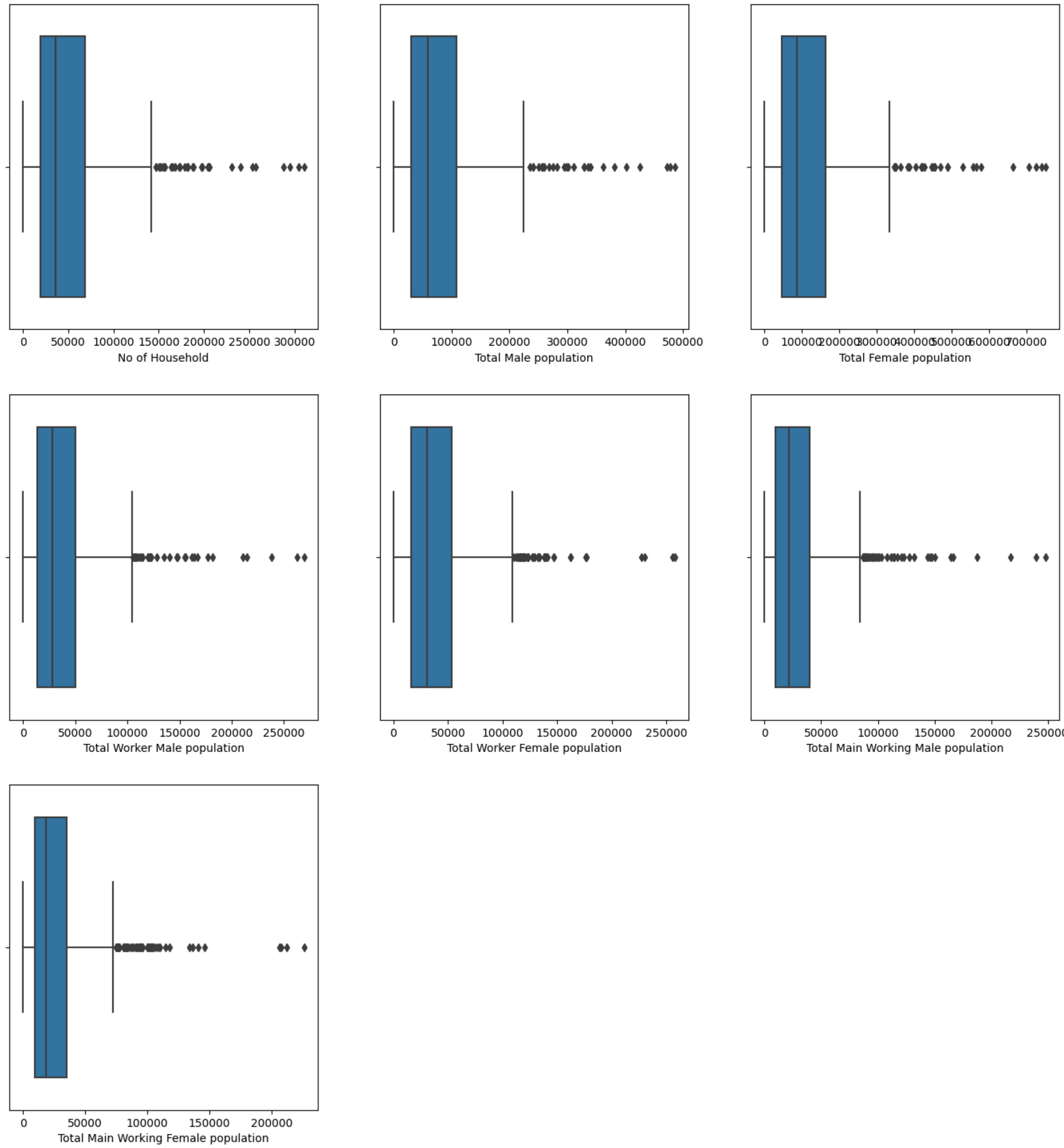
1. No distribution (No_HH, TOT_M, TOT_F, TOT_WORK_M, TOT_WORK_F, MAINWORK_M, MAINWORK_F) is evenly distributed (symmetric).
2. No_HH, TOT_M, TOT_F, TOT_WORK_M, TOT_WORK_F, MAINWORK_M, MAINWORK_F are Positively Skewed (mean is more than the mode).

In [156…

```
# Box Plots for No_HH, TOT_M, TOT_F, TOT_WORK_M, TOT_WORK_F, MAINWORK_M, MAINWORK_F

fig, axes = plt.subplots(3,3, figsize=(17, 18))

sns.boxplot(ax=axes[0, 0], data=df_pca, x='No_HH')
sns.boxplot(ax=axes[0, 1], data=df_pca, x='TOT_M')
sns.boxplot(ax=axes[0, 2], data=df_pca, x='TOT_F')
sns.boxplot(ax=axes[1, 0], data=df_pca, x='TOT_WORK_M')
```

```
sns.boxplot(ax=axes[1, 1], data=df_pca, x='TOT_WORK_F')
sns.boxplot(ax=axes[1, 2], data=df_pca, x='MAINWORK_M')
sns.boxplot(ax=axes[2, 0], data=df_pca, x='MAINWORK_F')
axes[2,1].axis("off")
axes[2,2].axis("off")

axes[0,0].set(xlabel='No of Household')
axes[0,1].set(xlabel='Total Male population')
axes[0,2].set(xlabel='Total Female population')
axes[1,0].set(xlabel='Total Worker Male population')
axes[1,1].set(xlabel='Total Worker Female population')
axes[1,2].set(xlabel='Total Main Working Male population')
axes[2,0].set(xlabel='Total Main Working Female population')

plt.suptitle('Fig 2: Box Plots: No_HH, TOT_M, TOT_F, TOT_WORK_M, TOT_WORK_F, MAINWO

plt.show()
```

Fig 2: Box Plots: No_HH, TOT_M, TOT_F, TOT_WORK_M, TOT_WORK_F, MAINWORK_M, MAINWORK_F



Observations and Insights:

No_HH, TOT_M, TOT_F, TOT_WORK_M, TOT_WORK_F, MAINWORK_M, MAINWORK_F columns are having outliers.

Bivariate Analysis

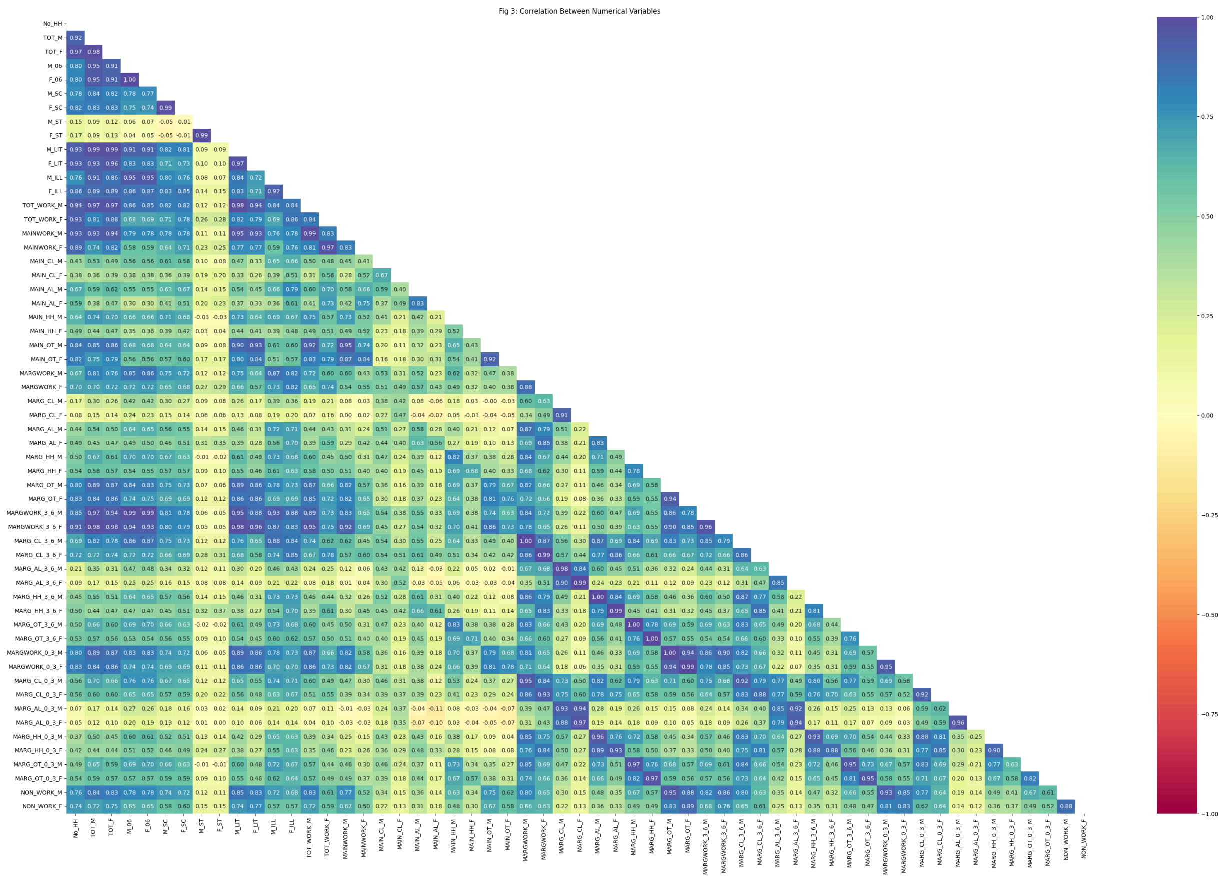
In [157…

```
df_pca_num_ext = df_pca.select_dtypes(include='number') # Selecting numerical colum
df_pca_num = df_pca_num_ext.drop(columns=['State Code', 'Dist.Code']) # Dropping nu
```

In [158…

```
# Heatmap to plot correlation between all numerical variables in the dataset

corr = df_pca_num.corr(method='pearson')
```

```python
mask = np.triu(np.ones_like(corr, dtype=bool))

plt.figure(figsize=(40, 25))
sns.heatmap(df_pca_num.corr(), annot=True, vmin=-1, vmax=1, fmt=".2f", cmap="Spectr
plt.title('Fig 3: Correlation Between Numerical Variables')
plt.show()
```



In [159…

```python
df_pca_ra = df_pca.copy()

# calculate ratios
sums = df_pca[['TOT_M', 'TOT_F']].sum(axis=1)
df_pca_ra['MaleRatio'] = df_pca['TOT_M'] / sums
df_pca_ra['FemaleRatio'] = df_pca['TOT_F'] / sums


#df_pca_ra
```

In [160…

```python
df_pca_ra_st_mr = df_pca_ra.groupby(['State']).agg(Total_Male_Ratio = ('MaleRatio',
df_pca_ra_st_fr = df_pca_ra.groupby(['State']).agg(Total_Female_Ratio = ('FemaleRat
```

In [161…

```python
df_pca_ra_st_mr.loc[df_pca_ra_st_mr['Total_Male_Ratio'].idxmax()]
```

Out[161…

```
State                 Uttar Pradesh
Total_Male_Ratio          30.691014
Name: 0, dtype: object
```

In [162…

```python
df_pca_ra_st_mr.loc[df_pca_ra_st_mr['Total_Male_Ratio'].idxmin()]
```

Out[162…
```
State                  Dadara & Nagar Havelli
Total_Male_Ratio                     0.391961
Name: 34, dtype: object
```

In [163…
```
df_pca_ra_st_fr.loc[df_pca_ra_st_fr['Total_Female_Ratio'].idxmax()]
```

Out[163…
```
State                    Uttar Pradesh
Total_Female_Ratio           40.308986
Name: 0, dtype: object
```

In [164…
```
df_pca_ra_st_fr.loc[df_pca_ra_st_fr['Total_Female_Ratio'].idxmin()]
```

Out[164…
```
State                    Lakshadweep
Total_Female_Ratio          0.535314
Name: 34, dtype: object
```

Observations and Insights:

State:

- Uttar Pradesh has the highest male ratio and Dadara & Nagar Havelli has the lowest male ratio.
- Uttar Pradesh has the highest female ratio and Lakshadweep has the lowest female ratio.

In [165…
```
df_pca_ra_an_mr = df_pca_ra.groupby(['Area Name']).agg(Total_Male_Ratio = ('MaleRat
df_pca_ra_an_fr = df_pca_ra.groupby(['Area Name']).agg(Total_Female_Ratio = ('Femal
```

In [166…
```
df_pca_ra_an_mr.loc[df_pca_ra_an_mr['Total_Male_Ratio'].idxmax()]
```

Out[166…
```
Area Name            Aurangabad
Total_Male_Ratio       0.814337
Name: 0, dtype: object
```

In [167…
```
df_pca_ra_an_mr.loc[df_pca_ra_an_mr['Total_Male_Ratio'].idxmin()]
```

Out[167…
```
Area Name             Krishna
Total_Male_Ratio     0.304576
Name: 634, dtype: object
```

In [168…
```
df_pca_ra_an_fr.loc[df_pca_ra_an_fr['Total_Female_Ratio'].idxmax()]
```

Out[168…
```
Area Name                Raigarh
Total_Female_Ratio      1.316205
Name: 0, dtype: object
```

In [169…
```
df_pca_ra_an_fr.loc[df_pca_ra_an_fr['Total_Female_Ratio'].idxmin()]
```

Out[169…
```
Area Name               Lakshadweep
Total_Female_Ratio         0.535314
Name: 634, dtype: object
```

Observations and Insights:

Area Name:

- Aurangabad has the highest male ratio and Krishna has the lowest male ratio.
- Raigarh has the highest female ratio and Lakshadweep has the lowest female ratio.

Outlier Treatment

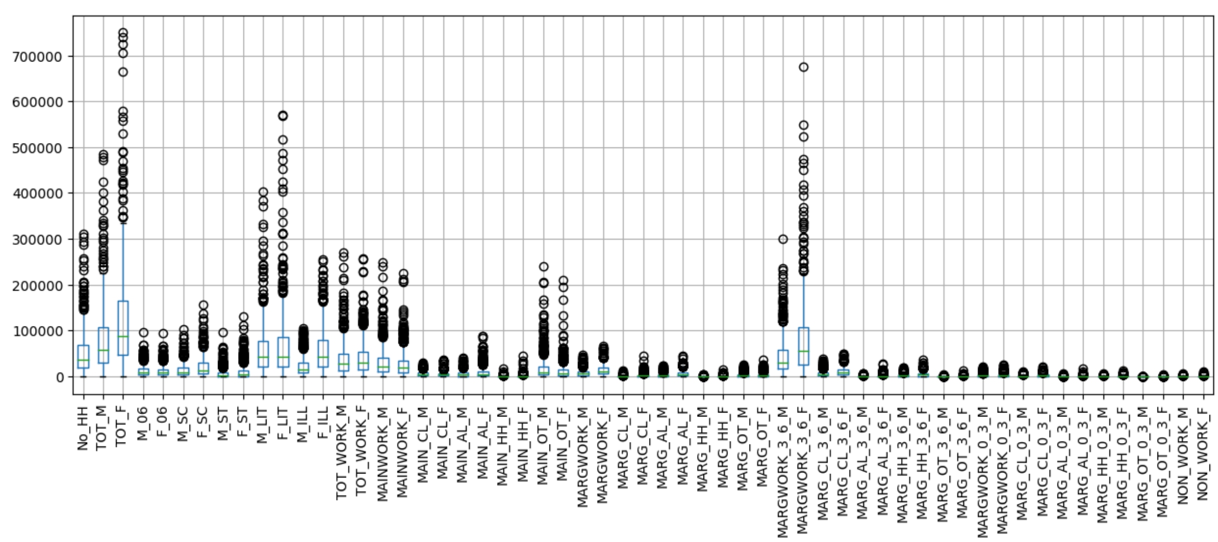
In [170…

```
# User Defined Function (UDF) to treat outliers
def treat_outlier(x):
    # taking 5,25,75 percentile of column
    q5=np.percentile(x,5)
    q25=np.percentile(x,25)
    q75=np.percentile(x,75)
    q95=np.percentile(x,95)
    #calculationg IQR range
    IQR=q75-q25
    #Calculating minimum threshold
    lower_bound=q25-(1.5*IQR)
    upper_bound=q75+(1.5*IQR)
    #Capping outliers
    return x.apply(lambda y: upper_bound if y > upper_bound else y).apply(lambda y:
```

In [171…

```
# Before outlier treatment

df_pca_num.boxplot(figsize=(15,5))
plt.suptitle('Fig 4: Box Plots: Before Outlier Treatment')
plt.xticks(rotation=90)
plt.show()
```

Fig 4: Box Plots: Before Outlier Treatment



In [172…

```
outlier_list = [x for x in df_pca_num.columns] # Numerical columns having outliers
```

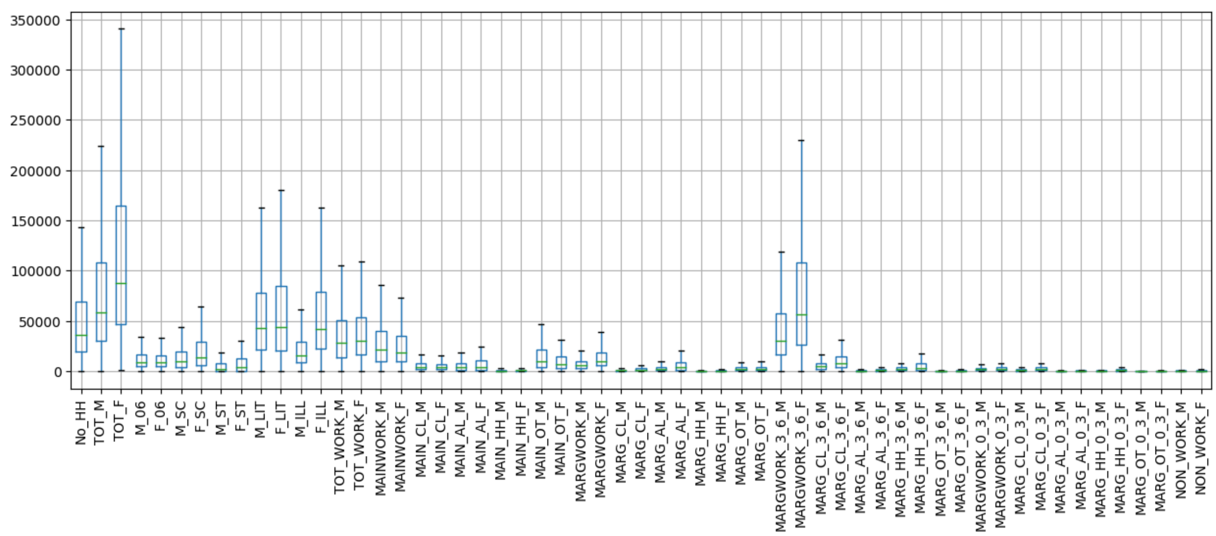
In [173…

```
# Using for loop to iterate over numerical columns and calling treat_outlier UDF to
for i in df_pca_num[outlier_list]:
    df_pca_num[i]=treat_outlier(df_pca_num[i])
```

In [174…

```
# After outlier treatment

df_pca_num.boxplot(figsize=(15,5))
plt.suptitle('Fig 5: Box Plots: After Outlier Treatment')
plt.xticks(rotation=90)
plt.show()
```

Fig 5: Box Plots: After Outlier Treatment



We can observe from above Box Plots that there are no outliers in the numerical columns (to be used for PCA) after the treatment.

Scaling

In [175…

```
# scaling the data before clustering
X = StandardScaler()
scaled_df = X.fit_transform(df_pca_num)
```

In [176…

```
# creating a dataframe of the scaled data
scaled_df_pca = pd.DataFrame(scaled_df, columns=df_pca_num.columns)
```

In [177…

```
scaled_df_pca.head() # Returns first 5 rows
```

Out[177…

	No_HH	TOT_M	TOT_F	M_06	F_06	M_SC	F_SC	M_ST
0	-1.038986	-0.874837	-0.937027	-0.624685	-0.561282	-1.080201	-1.079963	-0.510440
1	-1.076896	-0.938023	-1.009723	-0.773932	-0.835657	-1.079873	-1.079635	-0.771833
2	-1.121858	-1.154665	-1.141539	-1.141642	-1.138104	-1.080201	-1.079635	0.122588
3	-1.201599	-1.217171	-1.214930	-1.197772	-1.176091	-1.080447	-1.079963	-0.399531
4	-0.938495	-0.921309	-0.935018	-0.700931	-0.740523	-1.078807	-1.078160	0.432534

PCA

Statistical tests to be done before PCA

Bartletts Test of Sphericity

Bartlett's test of sphericity tests the hypothesis that the variables are uncorrelated in the population.

- H0: All variables in the data are uncorrelated
- Ha: At least one pair of variables in the data are correlated

If the null hypothesis cannot be rejected, then PCA is not advisable.

If the p-value is small, then we can reject the null hypothesis and agree that there is at least one pair of variables in the data which are correlated hence PCA is recommended.

In [178…

```
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity
chi_square_value,p_value=calculate_bartlett_sphericity(scaled_df_pca)
p_value
```

Out[178…

```
0.0
```

KMO Test

The Kaiser-Meyer-Olkin (KMO) - measure of sampling adequacy (MSA) is an index used to examine how appropriate PCA is.

Generally, if MSA is less than 0.5, PCA is not recommended, since no reduction is expected. On the other hand, MSA > 0.7 is expected to provide a considerable reduction is the dimension and extraction of meaningful components.

In [179…

```
from factor_analyzer.factor_analyzer import calculate_kmo
kmo_all,kmo_model=calculate_kmo(scaled_df_pca)
kmo_model
```

Out[179…

```
0.9361896166652609
```

In [180…

```
pca = PCA(n_components=57, random_state=123)
df_pca_ext = pca.fit_transform(scaled_df_pca)
df_pca_ext.transpose().round(2)
```

Out[180…

```
array([[-5.53, -5.49, -7.47, ..., -7.89, -7.86, -7.42],
       [ 0.43, -0.11, -0.22, ..., -1.  , -1.  , -1.41],
       [-1.47, -2.02, -0.25, ..., -0.91, -0.85, -0.87],
       ...,
       [ 0.01, -0.  , -0.  , ..., -0.  , -0.  , -0.  ],
       [ 0.  ,  0.01, -0.  , ...,  0.  , -0.  , -0.  ],
       [ 0.  , -0.01,  0.  , ..., -0.  ,  0.  ,  0.  ]])
```

In [181…

```
#Step 1: Obtaining the Eigen Vectors when the Principal Components are kept exactly
```

```
print('Eigen Vectors \n %s',pca.components_.round(2))
```

```
Eigen Vectors
 %s [[ 0.15  0.16  0.16 ...  0.14  0.15  0.14]
 [-0.12 -0.08 -0.09 ...  0.04 -0.05 -0.04]
 [ 0.1  -0.04  0.03 ... -0.1  -0.13 -0.03]
 ...
 [ 0.   -0.01  0.02 ... -0.01  0.06 -0.01]
 [ 0.    0.05  0.   ...  0.01 -0.08 -0.  ]
 [-0.   -0.    0.01 ...  0.    0.01  0.  ]]
```

In [182…

```
var_exp = pca.explained_variance_ratio_
print(var_exp.round(2))
```

```
[0.62 0.13 0.07 0.05 0.03 0.02 0.02 0.01 0.01 0.01 0.   0.   0.   0.
 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
 0.  ]
```

In [183…

```
# Step 2: Obtaining the Cumulative Sum of the Expalained Variance
cum_var_exp = np.cumsum(var_exp)
print('Cumulative Variance Explained in Percentage:',(cum_var_exp*100).round(2))
```

```
Cumulative Variance Explained in Percentage: [ 62.44  75.83  82.44  87.3   90.64  9
2.66  94.39  95.21  95.9   96.47
  96.95  97.36  97.68  97.97  98.22  98.45  98.63  98.79  98.95  99.09
  99.2   99.31  99.4   99.48  99.55  99.61  99.66  99.71  99.75  99.79
  99.82  99.85  99.87  99.89  99.91  99.93  99.94  99.95  99.96  99.97
  99.98  99.98  99.98  99.99  99.99  99.99  99.99 100.   100.   100.
 100.   100.   100.   100.   100.   100.   100.  ]
```

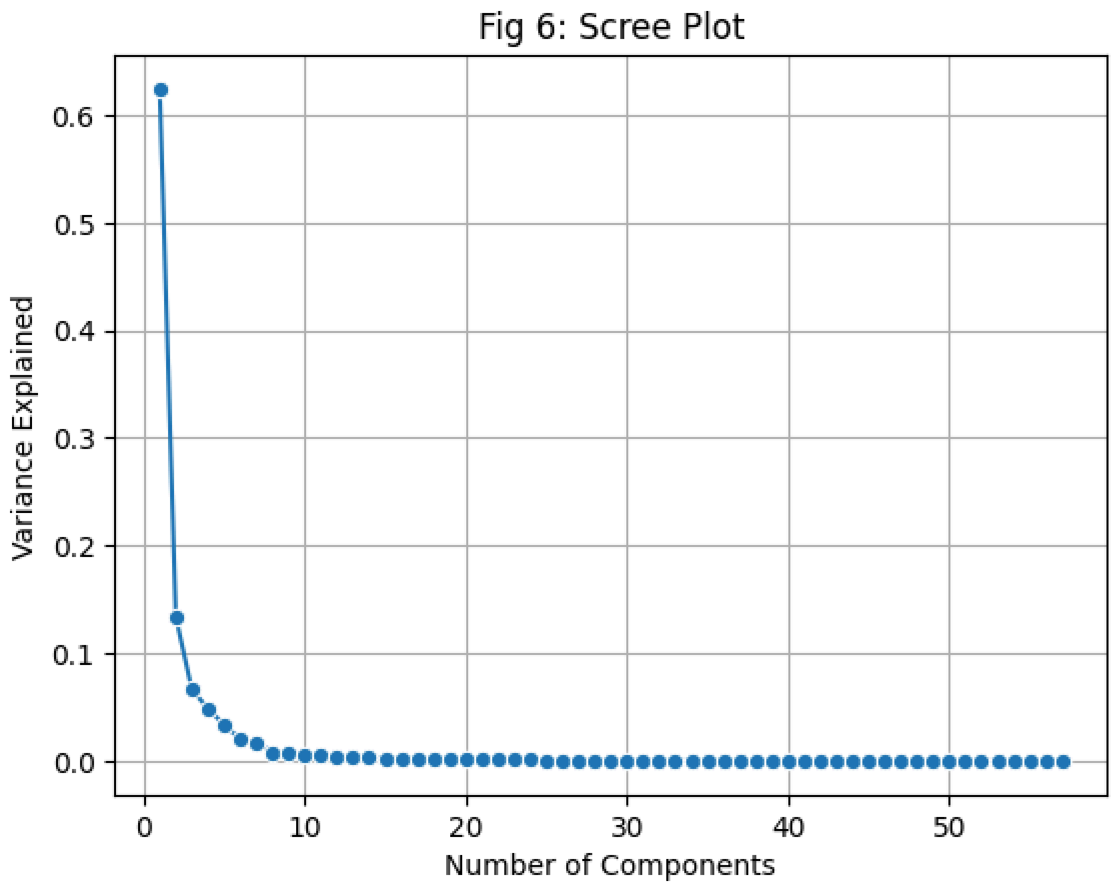
Observations and Insights:

- We can see above that more than 90% of the variance is explained by 5 Principal Components.
- Around 95% of the variance is explained by 8 Principal Components.
- Around 98% of the variance is explained by 15 Principal Components

Scree plot

In [184…

```
# Step 3 View Scree Plot to identify the number of components to be built

sns.lineplot(y=var_exp,x=range(1,len(var_exp)+1),marker='o')
plt.xlabel('Number of Components')
plt.ylabel('Variance Explained')
plt.grid()
plt.title('Fig 6: Scree Plot')
plt.show()
```



The number of components can be decided based upon the explained variance. Here, it is decided to keep the number of components as 8 (as the cumulative explained variance is around 95%).

In [185…

```
# Step 4 Apply PCA for the number of decided components to get the loadings and com

# NOTE - we are generating only 8 PCA dimensions (dimensionality reduction from 57
pca = PCA(n_components=8, random_state=123)
df_pca_final = pca.fit_transform(scaled_df_pca)
df_pca_final.transpose().round(2) # Component output
```

Out[185…

```
array([[-5.53, -5.49, -7.47, ..., -7.89, -7.86, -7.42],
       [ 0.43, -0.11, -0.22, ..., -1.  , -1.  , -1.41],
       [-1.47, -2.02, -0.25, ..., -0.91, -0.85, -0.87],
       ...,
       [ 0.54, -1.01, -0.15, ..., -0.05,  0.44,  0.55],
       [-0.38,  0.29, -0.19, ...,  0.36,  0.31,  0.28],
       [-0.26, -0.51, -0.12, ...,  0.04,  0.13,  0.07]])
```

In [186…

```
df_pca_final.shape
```

Out[186…

```
(640, 8)
```

In [187…

```
# Loading of each feature on the components
# Eigen Vectors when PC's are kept as 8
pca.components_.round(2)
```

```
Out[187…  array([[ 0.15,  0.16,  0.16,  0.16,  0.16,  0.14,  0.14,  0.02,  0.02,
                  0.16,  0.15,  0.15,  0.16,  0.15,  0.14,  0.14,  0.13,  0.11,
                  0.08,  0.12,  0.09,  0.14,  0.13,  0.12,  0.12,  0.16,  0.15,
                  0.09,  0.07,  0.13,  0.12,  0.15,  0.14,  0.15,  0.15,  0.16,
                  0.16,  0.16,  0.15,  0.09,  0.07,  0.13,  0.11,  0.15,  0.14,
                  0.15,  0.15,  0.14,  0.13,  0.06,  0.06,  0.12,  0.11,  0.14,
                  0.14,  0.15,  0.14],
                [-0.12, -0.08, -0.09, -0.02, -0.01, -0.08, -0.09,  0.07,  0.07,
                 -0.11, -0.13, -0.01, -0.02, -0.12, -0.08, -0.17, -0.14,  0.04,
                  0.1 , -0.05, -0.07, -0.1 , -0.11, -0.2 , -0.21,  0.08,  0.11,
                  0.27,  0.28,  0.16,  0.14,  0.04,  0.01, -0.07, -0.09, -0.04,
                 -0.09,  0.07,  0.09,  0.26,  0.27,  0.15,  0.12,  0.04, -0.  ,
                 -0.08, -0.1 ,  0.14,  0.17,  0.28,  0.29,  0.18,  0.18,  0.05,
                  0.04, -0.05, -0.04],
                [ 0.1 , -0.04,  0.03, -0.07, -0.07, -0.04,  0.02,  0.32,  0.34,
                 -0.03, -0.01, -0.05,  0.08, -0.  ,  0.19,  0.02,  0.21,  0.03,
                  0.19,  0.23,  0.36, -0.1 ,  0.02, -0.03,  0.07, -0.07,  0.1 ,
                 -0.1 , -0.04,  0.07,  0.26, -0.14, -0.09, -0.13, -0.05, -0.07,
                 -0.06, -0.06,  0.13, -0.1 , -0.02,  0.08,  0.28, -0.14, -0.09,
                 -0.13, -0.06, -0.1 ,  0.03, -0.12, -0.09,  0.03,  0.16, -0.14,
                 -0.1 , -0.13, -0.03],
                [ 0.08,  0.05,  0.07,  0.03,  0.02,  0.01,  0.02,  0.09,  0.08,
                  0.09,  0.13, -0.03, -0.01,  0.07,  0.11,  0.1 ,  0.13,  0.08,
                  0.27, -0.12, -0.02, -0.02, -0.05,  0.15,  0.16, -0.08,  0.02,
                  0.16,  0.29, -0.25, -0.15, -0.17, -0.15,  0.02,  0.06,  0.04,
                  0.05, -0.09,  0.02,  0.13,  0.29, -0.25, -0.14, -0.17, -0.14,
                  0.02,  0.06, -0.02,  0.01,  0.21,  0.24, -0.24, -0.19, -0.17,
                 -0.17,  0.02,  0.06],
                [-0.01, -0.04, -0.02, -0.08, -0.08, -0.17, -0.16,  0.42,  0.42,
                 -0.01,  0.03, -0.1 , -0.11, -0.02, -0.02, -0.04, -0.05, -0.3 ,
                 -0.26, -0.25, -0.2 , -0.06, -0.02,  0.07,  0.11,  0.07,  0.08,
                 -0.02, -0.06, -0.05, -0.01,  0.01,  0.04,  0.15,  0.19, -0.06,
                 -0.02,  0.06,  0.06, -0.01, -0.06, -0.06, -0.03,  0.  ,  0.04,
                  0.13,  0.17,  0.09,  0.11, -0.02, -0.04,  0.02,  0.05,  0.01,
                  0.05,  0.19,  0.25],
                [ 0.08,  0.07,  0.08,  0.09,  0.08,  0.05,  0.05, -0.23, -0.21,
                  0.08,  0.1 ,  0.04,  0.01,  0.04, -0.02,  0.02, -0.05, -0.29,
                 -0.27, -0.02, -0.06, -0.14, -0.32,  0.07,  0.03,  0.08,  0.1 ,
                 -0.03, -0.03,  0.08,  0.12, -0.17, -0.32,  0.02,  0.  ,  0.1 ,
                  0.12,  0.07,  0.07, -0.04, -0.05,  0.07,  0.09, -0.17, -0.34,
                  0.02, -0.  ,  0.11,  0.19, -0.  ,  0.02,  0.11,  0.19, -0.15,
                 -0.23,  0.02,  0.04],
                [ 0.11, -0.12, -0.01, -0.2 , -0.2 , -0.04,  0.05, -0.36, -0.33,
                 -0.07,  0.01, -0.24, -0.04, -0.09,  0.17, -0.09,  0.15, -0.29,
                  0.03, -0.11,  0.13, -0.06,  0.23, -0.01,  0.09, -0.06,  0.15,
                 -0.  ,  0.06, -0.09,  0.09, -0.06,  0.18, -0.02,  0.1 , -0.15,
                 -0.1 , -0.06,  0.14, -0.01,  0.06, -0.1 ,  0.09, -0.06,  0.18,
                 -0.02,  0.08, -0.03,  0.18,  0.01,  0.07, -0.08,  0.11, -0.05,
                  0.19, -0.02,  0.18],
                [-0.1 , -0.11, -0.12, -0.13, -0.14,  0.19,  0.18, -0.07, -0.08,
                 -0.1 , -0.13, -0.09, -0.05, -0.05, -0.07, -0.05, -0.08,  0.43,
                  0.2 ,  0.04,  0.05, -0.12, -0.25, -0.08, -0.08,  0.04,  0.06,
                 -0.06, -0.04,  0.02,  0.03,  0.03, -0.13,  0.18,  0.25, -0.15,
                 -0.12,  0.04,  0.07, -0.04, -0.01,  0.03,  0.06,  0.03, -0.12,
                  0.17,  0.22,  0.02, -0.  , -0.12, -0.1 , -0.05, -0.07,  0.04,
                 -0.15,  0.23,  0.33]])
```

In [188…

```python
# Explained variance for each PC (it gives the Eigen Values when PC's are kept at 8

pca.explained_variance_ratio_.round(2)
```

Out[188…

```
array([0.62, 0.13, 0.07, 0.05, 0.03, 0.02, 0.02, 0.01])
```

Creation of dataframe to include PC scores

In [189…

```python
df_pca_final = pd.DataFrame(df_pca_final.round(2),columns=['PC1','PC2','PC3','PC4',
                                         'PC5','PC6','PC7','PC8'])
df_pca_final.head()
```

Out[189…

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
0	-5.53	0.43	-1.47	-1.28	0.38	0.54	-0.38	-0.26
1	-5.49	-0.11	-2.02	-1.75	-0.01	-1.01	0.29	-0.51
2	-7.47	-0.22	-0.25	0.01	0.56	-0.15	-0.19	-0.12
3	-7.92	-0.65	-0.66	-0.74	0.27	0.21	0.11	-0.04
4	-5.18	2.30	-1.16	1.06	1.08	-0.05	0.07	-0.68

In [190…

```python
df_pca_final.shape
```

Out[190…

```
(640, 8)
```

Combining Categorical Fields & Principal Components

In [191…

```python
df_cat = df_pca.select_dtypes(include = ['object'])

df_new = pd.concat([df_cat, df_pca_final], axis=1)
```

In [192…

```python
df_new.shape
```

Out[192…

```
(640, 10)
```

In [193…

```python
df_new.head()
```

Out[193…

	State	Area Name	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
0	Jammu & Kashmir	Kupwara	-5.53	0.43	-1.47	-1.28	0.38	0.54	-0.38	-0.26
1	Jammu & Kashmir	Badgam	-5.49	-0.11	-2.02	-1.75	-0.01	-1.01	0.29	-0.51
2	Jammu & Kashmir	Leh(Ladakh)	-7.47	-0.22	-0.25	0.01	0.56	-0.15	-0.19	-0.12
3	Jammu & Kashmir	Kargil	-7.92	-0.65	-0.66	-0.74	0.27	0.21	0.11	-0.04
4	Jammu & Kashmir	Punch	-5.18	2.30	-1.16	1.06	1.08	-0.05	0.07	-0.68

```
In [194…
```

```python
df_new.describe(include='all').T
```

Out[194…

	count	unique	top	freq	mean	std	min	25%	50%	75%
State	640	35	Uttar Pradesh	71	NaN	NaN	NaN	NaN	NaN	NaN
Area Name	640	635	Raigarh	2	NaN	NaN	NaN	NaN	NaN	NaN
PC1	640.0	NaN	NaN	NaN	-0.000125	5.970708	-8.35	-4.475	-1.43	3.8025
PC2	640.0	NaN	NaN	NaN	-0.000109	2.76444	-7.66	-1.6	-0.295	1.695
PC3	640.0	NaN	NaN	NaN	-0.000141	1.94145	-4.32	-1.3725	-0.355	1.1825
PC4	640.0	NaN	NaN	NaN	0.000094	1.666404	-4.56	-0.96	-0.31	0.835
PC5	640.0	NaN	NaN	NaN	-0.000016	1.380962	-3.47	-0.87	-0.11	0.78
PC6	640.0	NaN	NaN	NaN	-0.000172	1.074742	-4.29	-0.5975	0.1	0.555
PC7	640.0	NaN	NaN	NaN	-0.000078	0.994013	-3.23	-0.5325	0.015	0.53
PC8	640.0	NaN	NaN	NaN	0.000094	0.681869	-2.84	-0.35	-0.01	0.3325

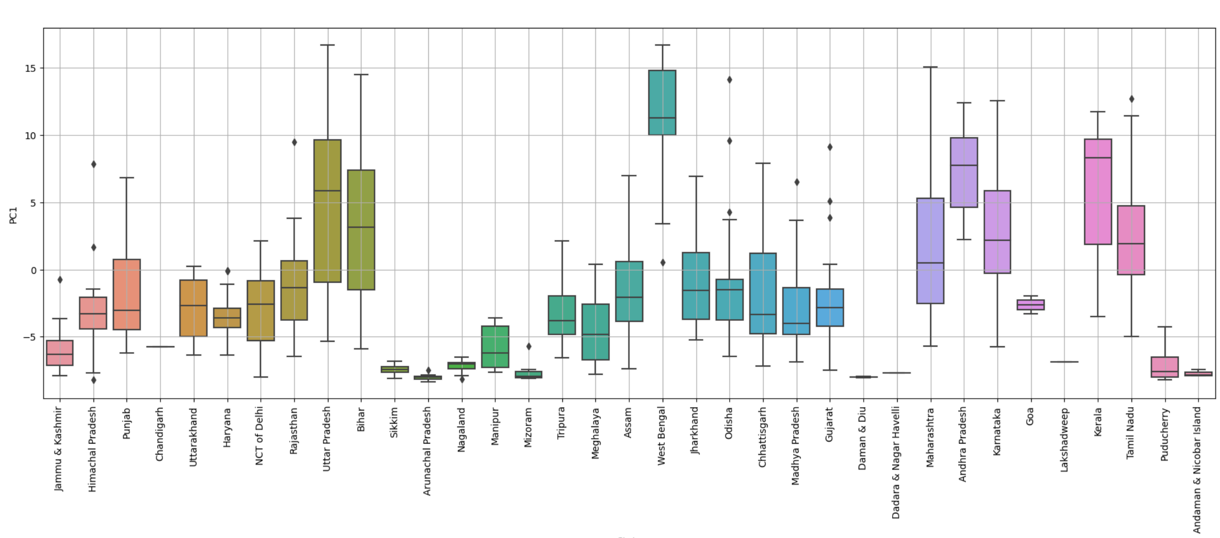
PC1 component is explaining the most variance when combined with State and Area Name.

EDA (Categorical Fields & Principal Components)

```
In [195…
```

```python
fig,ax = plt.subplots(figsize=(22,7))
sns.boxplot(x='State',y='PC1', data=df_new)
plt.suptitle('Fig 7: Box Plots: State vs PC1')
plt.grid()
ax.tick_params(axis='x', rotation=90)
plt.show()
```

Fig 7: Box Plots: State vs PC1

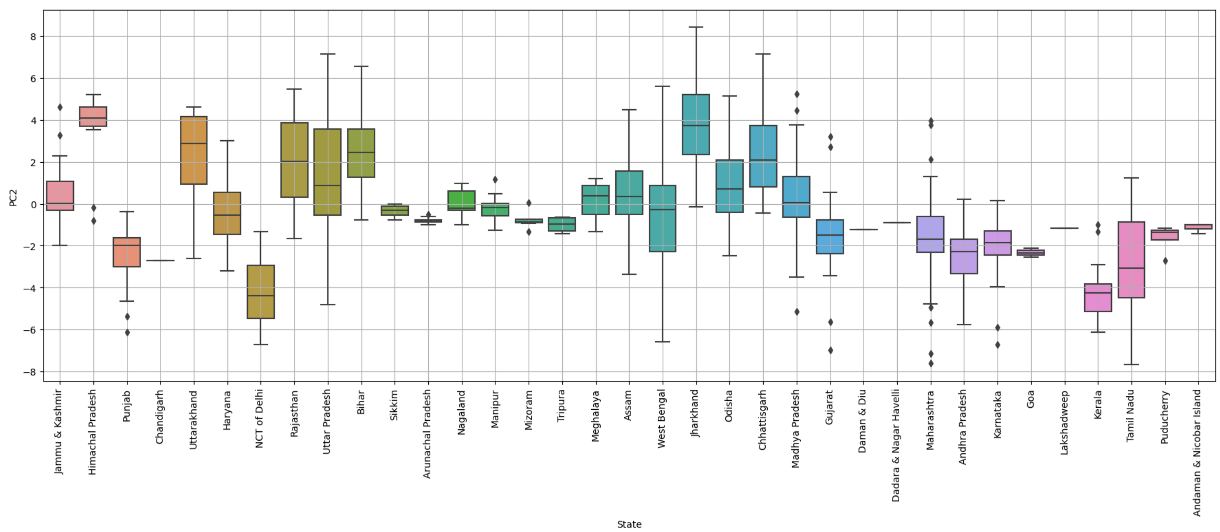


PC1 component is highest for West Bengal, Uttar Pradesh States & lowest for Daman and Diu State.

In [196...

```python
fig,ax = plt.subplots(figsize=(22,7))
sns.boxplot(x='State',y='PC2', data=df_new)
plt.suptitle('Fig 8: Box Plots: State vs PC2')
plt.grid()
ax.tick_params(axis='x', rotation=90)
plt.show()
```

Fig 8: Box Plots: State vs PC2

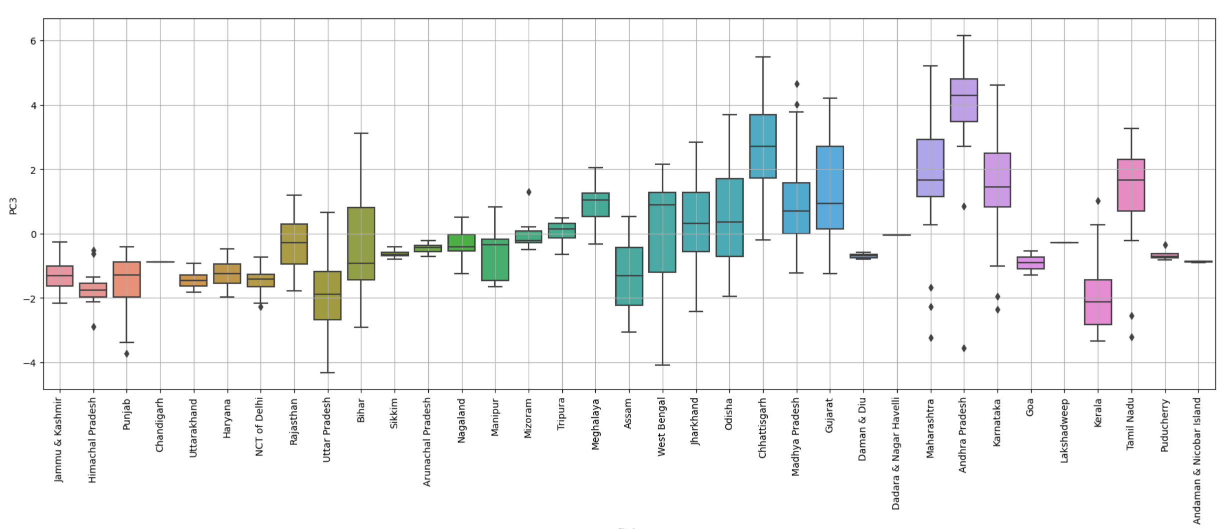


PC2 component is highest for Jharkhand State and lowest for Kerala State.

In [197...

```python
fig,ax = plt.subplots(figsize=(22,7))
sns.boxplot(x='State',y='PC3', data=df_new)
plt.suptitle('Fig 9: Box Plots: State vs PC3')
plt.grid()
ax.tick_params(axis='x', rotation=90)
plt.show()
```

Fig 9: Box Plots: State vs PC3

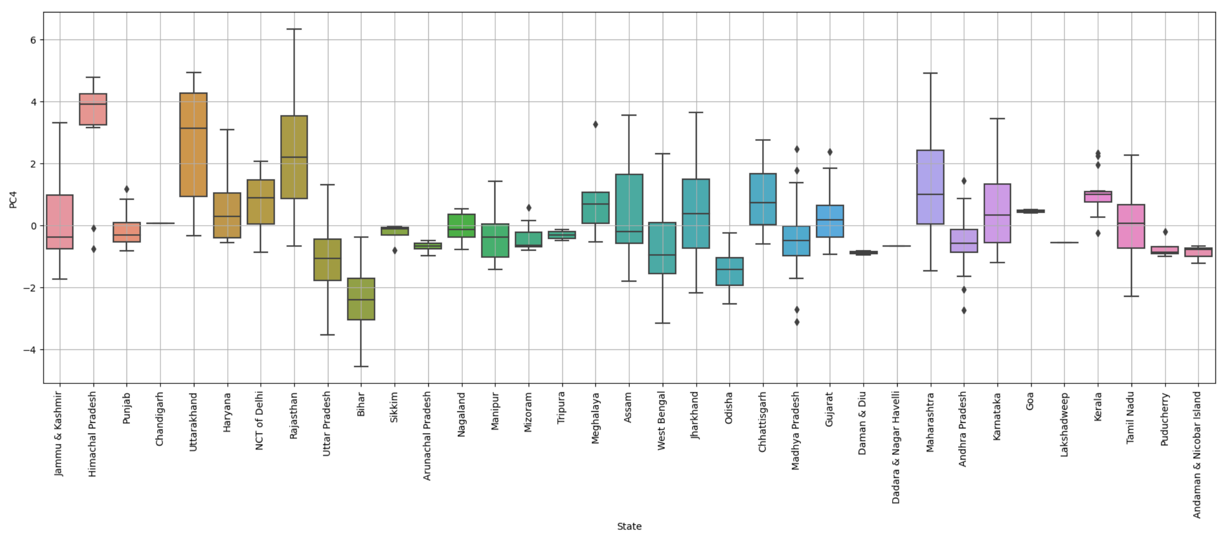


PC3 component is highest for Andhra Pradesh State and lowest for Kerala State.

In [198…

```
fig,ax = plt.subplots(figsize=(22,7))
sns.boxplot(x='State',y='PC4', data=df_new)
plt.suptitle('Fig 10: Box Plots: State vs PC4')
plt.grid()
ax.tick_params(axis='x', rotation=90)
plt.show()
```

Fig 10: Box Plots: State vs PC4

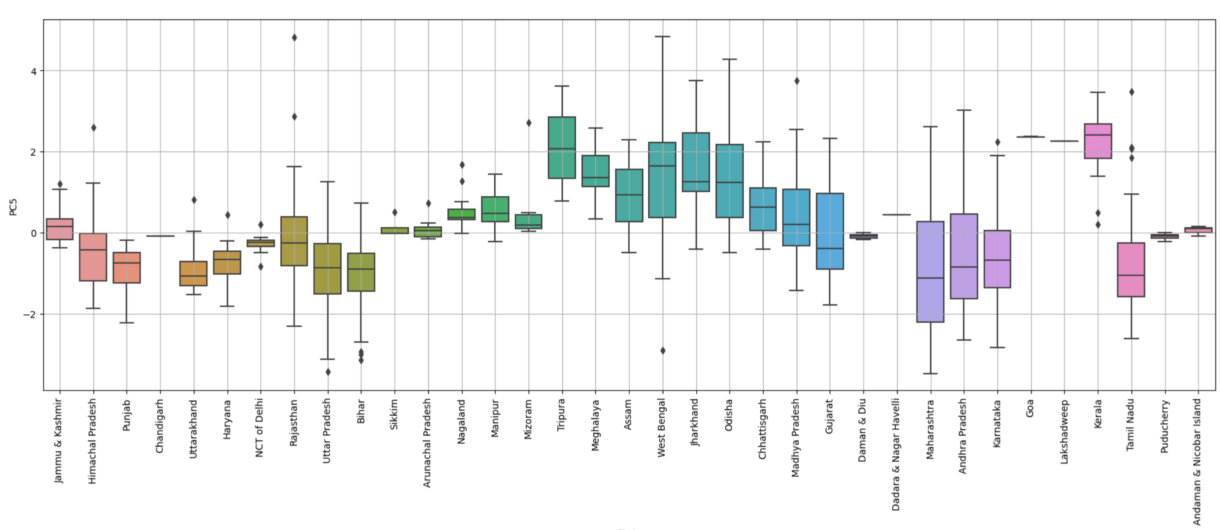


PC4 component is highest for Himachal Pradesh State and lowest for Bihar State.

In [199…

```
fig,ax = plt.subplots(figsize=(22,7))
sns.boxplot(x='State',y='PC5', data=df_new)
plt.suptitle('Fig 11: Box Plots: State vs PC5')
plt.grid()
ax.tick_params(axis='x', rotation=90)
plt.show()
```

Fig 11: Box Plots: State vs PC5

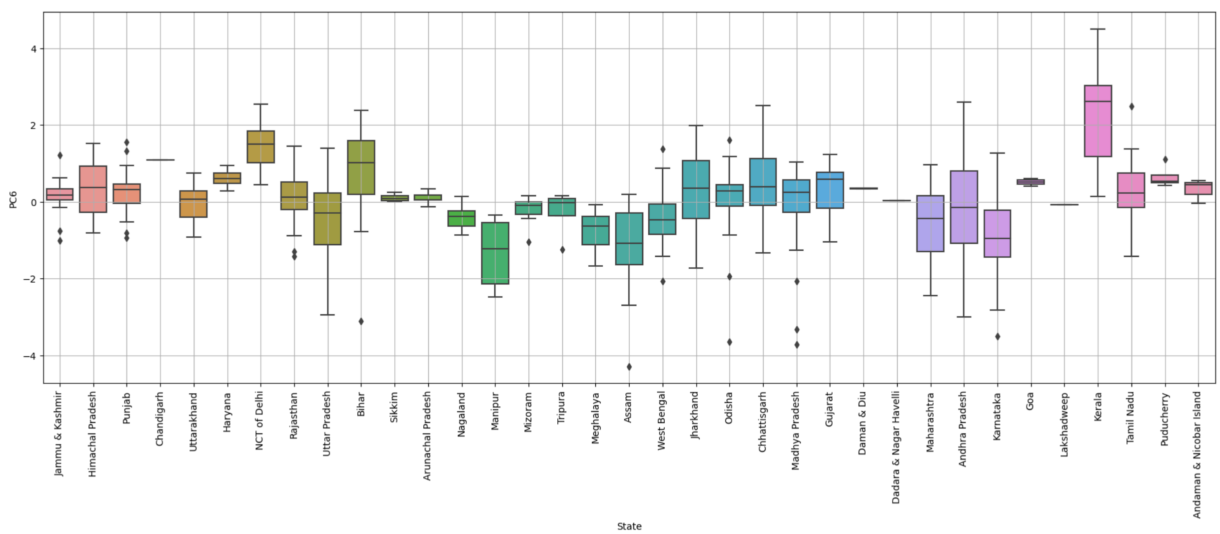


PC5 component is highest for Kerala State and lowest for Maharashtra State.

In [200…

```
fig,ax = plt.subplots(figsize=(22,7))
sns.boxplot(x='State',y='PC6', data=df_new)
plt.suptitle('Fig 12: Box Plots: State vs PC6Maharashtra ')
plt.grid()
ax.tick_params(axis='x', rotation=90)
plt.show()
```

Fig 12: Box Plots: State vs PC6Maharashtra

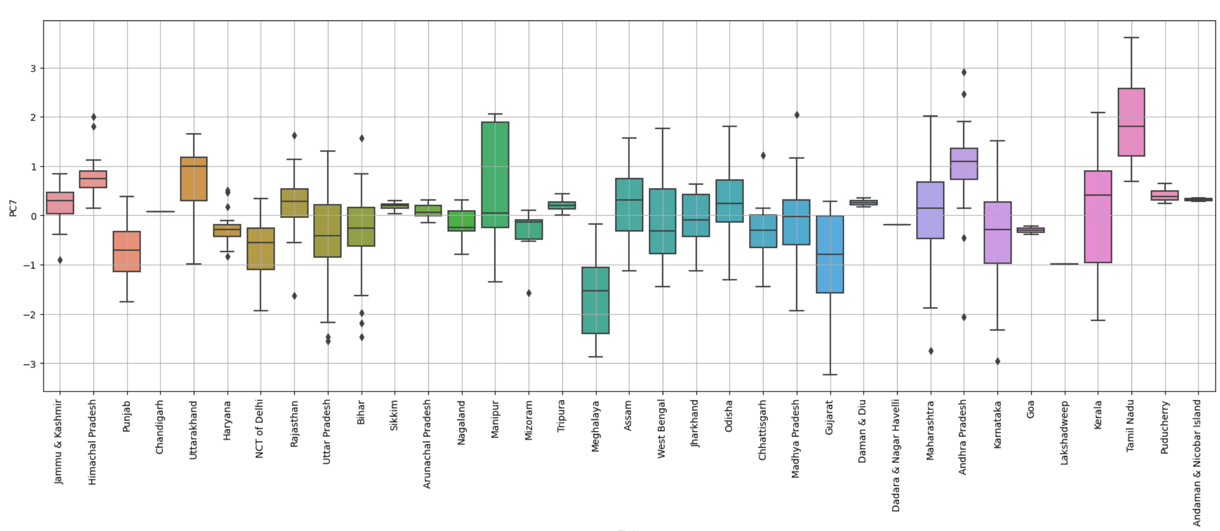


PC6 component is highest for Kerala State and lowest for Manipur State.

In [201…

```
fig,ax = plt.subplots(figsize=(22,7))
sns.boxplot(x='State',y='PC7', data=df_new)
plt.suptitle('Fig 13: Box Plots: State vs PC7')
plt.grid()
ax.tick_params(axis='x', rotation=90)
plt.show()
```

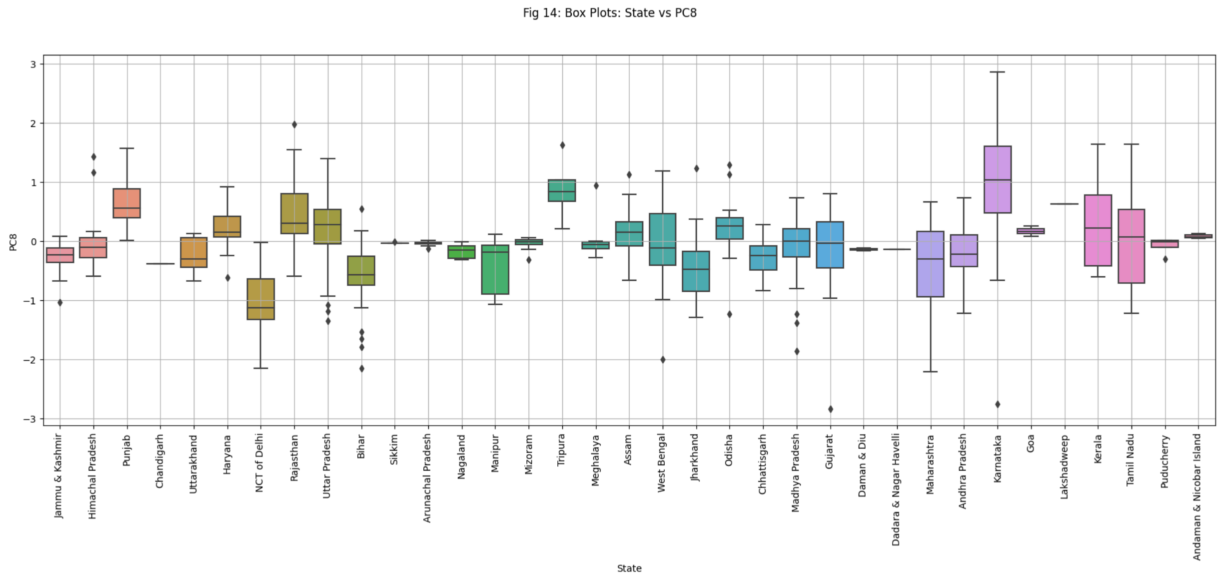
Fig 13: Box Plots: State vs PC7



PC7 component is highest for Tamil Nadu State and lowest for Meghalaya State.

In [202…

```
fig,ax = plt.subplots(figsize=(22,7))
sns.boxplot(x='State',y='PC8', data=df_new)
plt.suptitle('Fig 14: Box Plots: State vs PC8')
plt.grid()
ax.tick_params(axis='x', rotation=90)
plt.show()
```



PC8 component is highest for Karnataka State and lowest for NCT of Delhi State.

Observations and Insights:

From above plots we observe that PC components are higher for the States which have higher population in comparison to other States which are having lower population.

Linear equation for first PC

$Y1 = w11X1 + w12X2 + ... + w1pXp$

Here p (= 57) is the number of observations in the original dataset which was used for PCA.

$X1, X2, ..., Xp$ = Observed variables (original attributes) in the dataset.
$w11, w12, ..., wpp$ = Weights which were determined using PCA.

In [ ]: