Hospital Beds Data India

Complete - Data Analysis

INDEX

Step 1:

Install Library

Hospital Beds Data India

```
[ ]  1 import pandas as pd
```

```
[ ]  1 import warnings
     2 warnings.filterwarnings('ignore')
```

Step 2: Read Dataset

1) Exploratory Data Analysis (EDA)

Reading Dataset

Analyzing the Data

```
[ ]  1 df=pd.read_csv('/content/hospital_beds_per_india_v1.csv')
     2 df.head()
```

	country	state	county	lat	lng	type	measure	beds	population	year	source	source_url
0	IN	AN	NaN	11.7401	92.6586	TOTAL	1000HAB	2.825081	380520	2016	nhp	http://www.cbhidghs.nic.in/showfile.php?lid=1147
1	IN	AP	NaN	15.9129	79.7400	TOTAL	1000HAB	0.436072	53060000	2017	nhp	http://www.cbhidghs.nic.in/showfile.php?lid=1147
2	IN	AR	NaN	28.2180	94.7278	TOTAL	1000HAB	1.427893	1683600	2018	nhp	http://www.cbhidghs.nic.in/showfile.php?lid=1147
3	IN	AS	NaN	26.2006	92.9376	TOTAL	1000HAB	0.497753	34438756	2017	nhp	http://www.cbhidghs.nic.in/showfile.php?lid=1147
4	IN	BR	NaN	25.0961	85.3131	TOTAL	1000HAB	0.094838	122988691	2018	nhp	http://www.cbhidghs.nic.in/showfile.php?lid=1147

Step 3:

Analyse rows and columns count - 37 rows and 12 columns

Visualize Column titles

```
[ ]  1 df.shape
```

```
(37, 12)
```

```
[ ]  1 df.columns.values
```

```
array(['country', 'state', 'county', 'lat', 'lng', 'type', 'measure',
       'beds', 'population', 'year', 'source', 'source_url'], dtype=object)
```

Step 4:

Defined columns as categorical and numerical

eg: object mean by string values , float or int

Total: float(4), int(2), object(6)

```
[ ]  1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37 entries, 0 to 36
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   country     37 non-null     object
 1   state       37 non-null     object
 2   county      0 non-null      float64
 3   lat         37 non-null     float64
 4   lng         37 non-null     float64
 5   type        37 non-null     object
 6   measure     37 non-null     object
 7   beds        37 non-null     float64
 8   population  37 non-null     int64
 9   year        37 non-null     int64
 10  source      37 non-null     object
 11  source_url  37 non-null     object
dtypes: float64(4), int64(2), object(6)
memory usage: 3.6+ KB
```

Step 7:

df.shape finds there are 37 rows and 12 columns -

find unique value df.nunique()

country 1 - > only 1 country for 37 rows..,

state 37 -> there are 37 different values for each rows..,

latitude 37 - there are 37 different values for each rows..,

longitude 33 - there are 33 different values and 4 values are repeated..,

types & measure 1 - only 1 value for 37 rows..,

beds and population 37 - there are unique values for each

year 5 - there are 5 values repeated for each rows

Check for Duplication

```
1 df.nunique()
```

	0
country	1
state	37
county	0
lat	37
lng	33
type	1
measure	1
beds	37
population	37
year	5
source	1
source_url	1

Step 8: **Missing Values Calculation**

```
1 df.isnull().sum()
```

	0
country	0
state	0
county	37
lat	0
lng	0
type	0
measure	0
beds	0
population	0
year	0
source	0
source_url	0

County Column have missing values

Data Reduction

source column has no use to predit so drop it

```
[ ]  1 df.drop('source',axis=1,inplace=True)
```

County has Dublit column mean by Country already that column have

***county have 100% column was empty - more than 70% column was empty means drop that column as per client advise ***

```
[ ]  1 df.drop('county',axis=1,inplace=True)
```

```
1 df.isnull().sum()
```

	0
country	0
state	0
lat	0
lng	0
type	0
measure	0
beds	0
population	0
year	0
source_url	0

dtype: int64

Missing value and duplicate columns was deleted

Step 9:

Install Library - reason for analysis Latitude and Longitude

Latitude and Longitude values already given find Area.

```
[ ]  1 pip install geopy
     2
```

```
Requirement already satisfied: geopy in /usr/local/lib/python3.10/dist-packages (2.4.1)
Requirement already satisfied: geographiclib<3,>=1.52 in /usr/local/lib/python3.10/dist-packages (from geopy) (2.0)
```

Step 10: Create Database and Table in SQL

```
[ ]
```

```python
import sqlite3
import pandas as pd

# File name of the uploaded CSV
csv_filename = "/content/hospital_beds_per_india_v1.csv"  # Replace this with the uploaded file name

# Load CSV into a pandas DataFrame
df = pd.read_csv(csv_filename)

# Connect to SQLite database (or create a new one)
conn = sqlite3.connect("example.db")
cursor = conn.cursor()

# Write DataFrame to SQLite table
table_name = "hospitaldata"  # Specify your table name
df.to_sql(table_name, conn, if_exists="replace", index=False)

print(f"Table '{table_name}' created in SQLite database.")

```

```
Table 'hospitaldata' created in SQLite database.
```

Step 11:

Import library geopy to find area

```python
import sqlite3
import pandas as pd
from geopy.geocoders import Nominatim

# Function to find area using latitude and longitude
def find_area(lat, lon):
    try:
        # Initialize the geolocator
        geolocator = Nominatim(user_agent="geo_locator")

        # Perform reverse geocoding
        location = geolocator.reverse((lat, lon), exactly_one=True)

        # Extract address
        if location:
            return location.address
        else:
            return "Area not found for the given coordinates."
    except Exception as e:
        return f"An error occurred: {e}"

# Connect to the SQLite database
conn = sqlite3.connect('example.db')  # Replace with your database path
cursor = conn.cursor()

```

```
# Table name
table_name = "hospitaldata"  # Replace with your table name

# Add a new column 'area' to store the location data if it doesn't already exist
try:
    cursor.execute(f"ALTER TABLE {table_name} ADD COLUMN area TEXT")
except sqlite3.OperationalError:
    print("Column 'area' already exists or cannot be added.")

# Fetch latitude and longitude from the table
cursor.execute(f"SELECT rowid, lat, lng FROM {table_name}")
rows = cursor.fetchall()

# List to store DataFrame rows
data = []

# Process each row and update the 'area' column
for row in rows:
    rowid, lat, lng = row
    area = find_area(lat, lng)
    print(f"Coordinates: ({lat}, {lng}) -> Location: {area}")

    # Update the table with the retrieved area
    cursor.execute(f"UPDATE {table_name} SET area = ? WHERE rowid = ?", (area, rowid))
    conn.commit()
```

Alter add new column name as area and update area of each rows

```
    # Append data to the list
    data.append({"Latitude": lat, "Longitude": lng, "Area": area})

# Close the database connection
conn.close()

# Create a DataFrame
df1 = pd.DataFrame(data)

# Print the DataFrame
print(df1)

# Optionally save the DataFrame to a CSV file
df1.to_csv("hospitaldata_with_areas.csv", index=False)
```

```
df1=pd.read_csv('/content/hospitaldata_with_areas.csv')
df1.head()
```

	Latitude	Longitude	Area
0	11.7401	92.6586	Andaman Trunk Road, Jirkatang Camp No. 7, Ferr...
1	15.9129	79.7400	Vinukonda - Mupparajuvaripalem Road, Lakshmipu...
2	28.2180	94.7278	Kamba, Kamba ADC, Siang, Arunachal Pradesh, India
3	26.2006	92.9376	Doboka, Hojai, Assam, India
4	25.0961	85.3131	Shobha infotech, Shivaganj, Islampur, Nalanda,...

Step 12:

```
[ ]  1 df1.to_csv('correctdata-area.csv')
```

```
[ ]  1 df1=pd.read_csv('/content/correctdata-area.csv')
```

```
     1 df1.head()
```

	Unnamed: 0	lat	lng	Area
0	0	11.7401	92.6586	Andaman Trunk Road, Jirkatang Camp No. 7, Ferr...
1	1	15.9129	79.7400	Vinukonda - Mupparajuvaripalem Road, Lakshmipu...
2	2	28.2180	94.7278	Kamba, Kamba ADC, Siang, Arunachal Pradesh, India
3	3	26.2006	92.9376	Doboka, Hojai, Assam, India
4	4	25.0961	85.3131	Shobha infotech, Shivaganj, Islampur, Nalanda,...

Latitude, Longitude and Area column

Upload and read separate data frame value as csv file

Step 13: Merge two data frame and join columns

```
[ ]  1 # merge df and df1 using lat and lng
```

```
[ ]  1 df2=pd.merge(df,df1,on=['lat','lng'])
```

```
     1 df2.head()
```

	country	state	county	lat	lng	type	measure	beds	population	year	source	source_url	Area
0	IN	AN	NaN	11.7401	92.6586	TOTAL	1000HAB	2.825081	380520	2016	nhp	http://www.cbhidghs.nic.in/showfile.php?lid=1147	Andaman Trunk Road, Jirkatang Camp No. 7, Ferr..
1	IN	AP	NaN	15.9129	79.7400	TOTAL	1000HAB	0.436072	53060000	2017	nhp	http://www.cbhidghs.nic.in/showfile.php?lid=1147	Vinukonda Mupparajuvaripalem Road, Lakshmipu..
2	IN	AR	NaN	28.2180	94.7278	TOTAL	1000HAB	1.427893	1683600	2018	nhp	http://www.cbhidghs.nic.in/showfile.php?lid=1147	Kamba, Kamba ADC, Siang Arunachal Pradesh India
3	IN	AS	NaN	26.2006	92.9376	TOTAL	1000HAB	0.497753	34438756	2017	nhp	http://www.cbhidghs.nic.in/showfile.php?lid=1147	Doboka, Hojai Assam, India
4	IN	BR	NaN	25.0961	85.3131	TOTAL	1000HAB	0.094838	122988691	2018	nhp	http://www.cbhidghs.nic.in/showfile.php?lid=1147	Shobha infotech Shivaganj, Islampur Nalanda,..

Step 14:

2018 have 13 rows

2017 have 11 rows

2015 have 7 rows

2016 have 5 rows

2014 have 1 row

```
1 df3['year'].value_counts()
```

year	count
2018	13
2017	11
2015	7
2016	5
2014	1

dtype: int64

```
[ ]  1 print(df3.year.unique())
     2 print(df3.year.nunique())
```

```
[2016 2017 2018 2015 2014]
5
```

Step 15: EDA

Our Data is ready to perform EDA

Categorical Columns and numerical columns

```
[ ]  1 import numpy as np # Import the numpy library with the alias 'np'
     2
     3 cat_cols=df3.select_dtypes(include=['object']).columns
     4 num_cols = df3.select_dtypes(include=np.number).columns.tolist() # Now np is defined and can be used
     5 print("Categorical Variables:")
     6 print(cat_cols)
     7 print("Numerical Variables:")
     8 print(num_cols)
```

```
Categorical Variables:
Index(['country', 'state', 'type', 'measure', 'source', 'source_url', 'Area'], dtype='object')
Numerical Variables:
['lat', 'lng', 'beds', 'population', 'year']
```

Step 16:

2)Statistics Summary

```
1 df3.describe()
```

	lat	lng	beds	population	year
count	37.000000	37.000000	37.000000	3.700000e+01	37.000000
mean	22.500439	81.595037	27.304251	7.063516e+07	2016.756757
std	6.490228	7.347561	159.577108	2.216202e+08	1.211184
min	8.295441	71.192400	0.094838	7.978800e+04	2014.000000
25%	19.751500	76.085600	0.436072	1.510000e+06	2016.000000
50%	23.164500	79.019300	0.815190	2.630000e+07	2017.000000
75%	27.023800	87.855000	1.427893	5.306000e+07	2018.000000
max	33.778200	94.727800	971.725071	1.353000e+09	2018.000000

mean value of 5 years is 2016

minimum value is 2014

25% quartile - 2016

50% quartitle - 2017

75% quartile - 2018

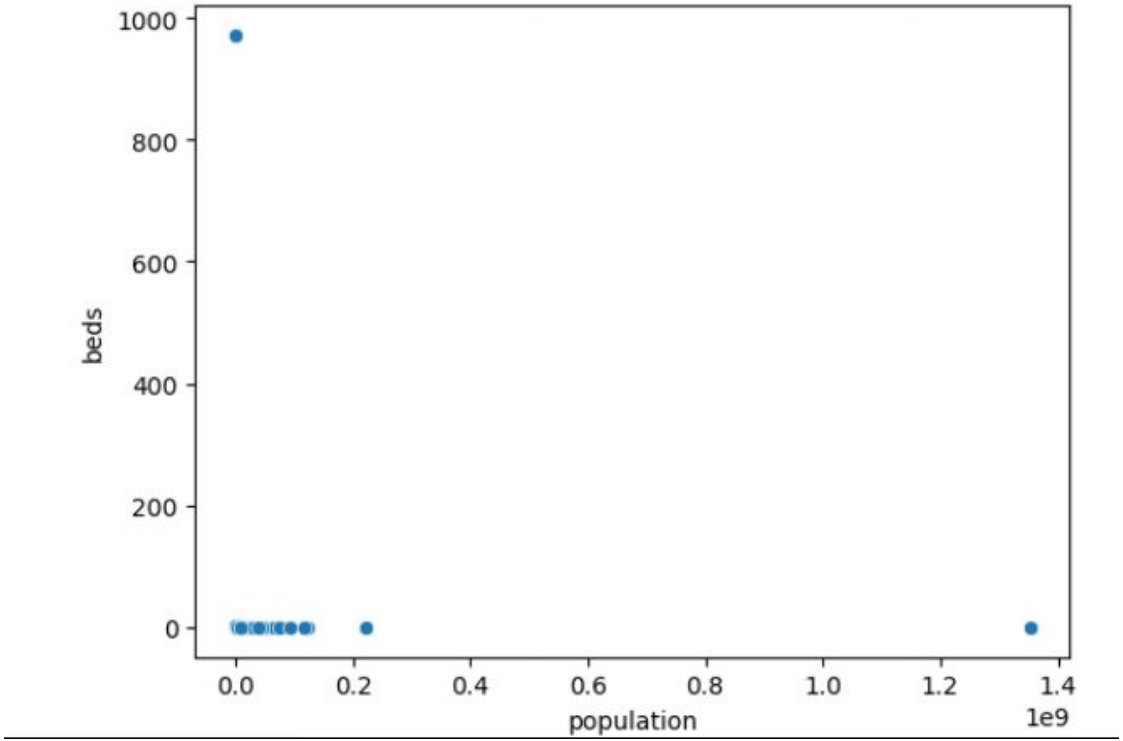
max value is 2018

Step 17: **check algorithm its Continuous or discrete**

Data taken as population and hospital beds

```
1 sns.scatterplot(data=df3,x='population',y='beds')
```

```
<Axes: xlabel='population', ylabel='beds'>
```



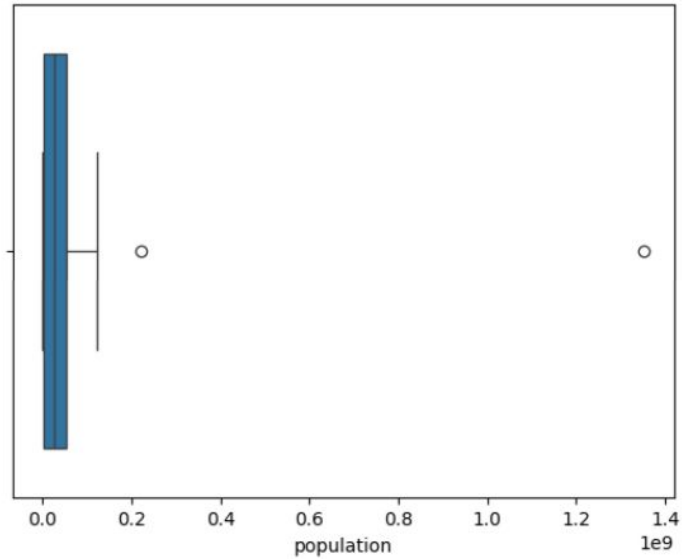
when machine learning topics using decision tree algorithm because of its disgret in nature

Step 18: **Data Preprocessing / data engineering**

Outlier check by Population and beds column

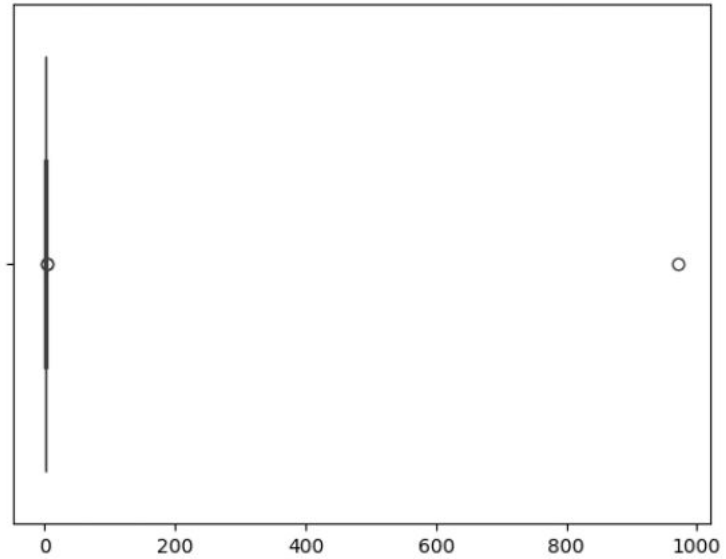
```
1 sns.boxplot(data=df3,x='population')
```

```
<Axes: xlabel='population'>
```




```
1 sns.boxplot(data=df3,x='beds')
```

```
<Axes: xlabel='beds'>
```



*in boxplot check outliers

*in population and beds columns have 2 outliers

*year column have no outliers

Step 19: Outlier Statistics Method - IQR Method

outlier value for 1353000000 (135 crore) is not for state population value its overall india population on 2017 so this row is no need, delete that particular row

```
[ ]
def detect_outlier_iqr(df3):
  outliers=[]
  data=sorted(df3)
  q1=np.percentile(data,25)
  q3=np.percentile(data,75)
  iqr=q3-q1
  lwr_bound=q1-(1.5*iqr)
  upr_bound=q3+(1.5*iqr)

  for i in data:
    if(i<lwr_bound or i>upr_bound):
      outliers.append(i)
  return outliers
sample_outliers=detect_outlier_iqr(df3['population'])
print("Outlier for Population : ",sample_outliers)
```

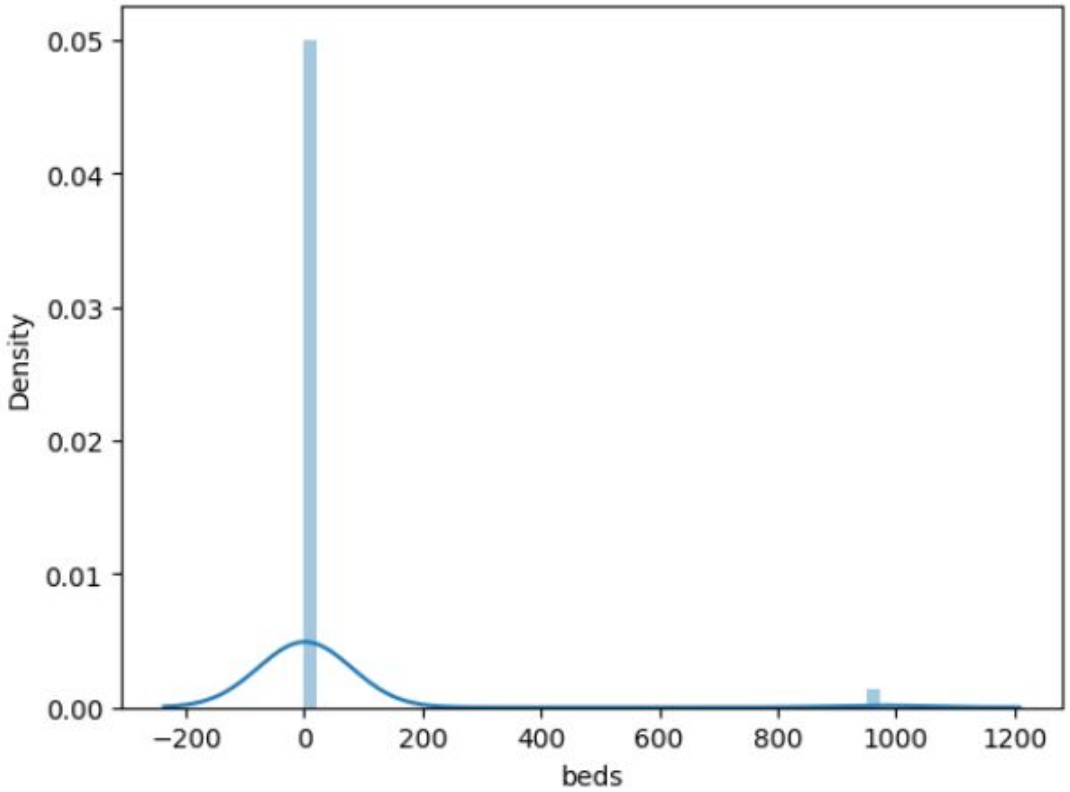
```
Outlier for Population :  [221073168, 1353000000]
```

```
[ ]
# Identify the rows to drop
rows_to_drop = df3[df3['state'] == 'IN'].index

# Drop those rows
df3.drop(rows_to_drop, axis=0, inplace=True)

```

Step 20:

```
1 sns.distplot(df3['beds'])
2 df3['beds'].skew()
```

```
5.999710160246187
```



Skew is between -0.5 to 0.5 is only normal distribution

skew 5.99 is positive distribution or right tail is highly skewed not a normal

Step 21:

```
[ ]  1 # Treatment of outliers
```

Treatment and Removal of outliers

using **capping** method as per client permission

1)forward fill - ffill

2)backward fill - bfill

***3)maximum or minimum ***

4) mode imputation

5) mean imputation

6) median imputation

7) standard deviation

8) moving average imputation

Step 22: Create Region for States (North,South,East,West)

```python
import pandas as pd
import sqlite3

# Reconnect to the database
conn = sqlite3.connect("example.db")  # Re-establish connection here

# Define state/UT code to region mapping
state_to_region = {
    "TN": "South",          # Tamil Nadu
    "KL": "South",          # Kerala
    "KA": "South",          # Karnataka
    "AP": "South",          # Andhra Pradesh
    "CT": "Central",         #Chhattisgarh
    "TS": "South",          # Telangana
    "MH": "West",           # Maharashtra
    "GJ": "West",           # Gujarat
    "GA": "West",           # Goa
    "RJ": "North",          # Rajasthan
    "UP": "North",          # Uttar Pradesh
    "PB": "North",          # Punjab
    "HR": "North",          # Haryana
    "JK": "North",          # Jammu & Kashmir
    "HP": "North",          # Himachal Pradesh
    "DL": "North",          # Delhi
    "UT": "North",          # Uttarakhand
    "WB": "East",           # West Bengal
    "OR": "East",           # Odisha
    "BR": "East",           # Bihar
    "JH": "East",           # Jharkhand
    "AS": "North-East",     # Assam
    "TG": "South",          #Telangana
    "MN": "North-East",     # Manipur
    "NL": "North-East",     # Nagaland
    "MZ": "North-East",     # Mizoram
    "TR": "North-East",     # Tripura
    "SK": "North-East",     # Sikkim
    "AR": "North-East",     # Arunachal Pradesh
    "ML": "North-East",     # Meghalaya
    "MP": "Central",        # Madhya Pradesh
    "CG": "Central",        # Chhattisgarh
    "PY": "South",          # Puducherry (Union Territory)
    "AN": "South",          # Andaman and Nicobar Islands (Union Territory)
    "LD": "South",          # Lakshadweep (Union Territory)
    "DD": "West",           # Daman and Diu (Union Territory)
    "DN": "West",           # Dadra and Nagar Haveli (Union Territory)
    "LA": "North",          # Ladakh (Union Territory)
}

# Step 1: Run your SQL query to fetch the data
query = f"SELECT state, year, population, beds FROM {table_name};"
result = pd.read_sql_query(query, conn)

# Step 2: Ensure no division by zero (replace 0 beds with None or handle it)
result["beds"] = result["beds"].replace(0, None)

# Step 3: Calculate the 'population per bed' for each state
result["population_per_bed"] = result["population"] / result["beds"]

# Step 4: Calculate the percentage of total beds
total_beds = result["beds"].sum()
result["bed_percentage"] = (result["beds"] / total_beds) * 100

# Step 5: Add the region column
result["region"] = result["state"].map(state_to_region)

# Step 6: Update the SQLite table with the new columns
result.to_sql(table_name, conn, if_exists="replace", index=False)  # Update the table

# Query the updated table
query = f"SELECT state, year, population, beds, bed_percentage, region FROM {table_name};"
updated_result = pd.read_sql_query(query, conn)

# Display the results
print(updated_result)

# Close the connection when done
conn.close()
```

```
  state  year  population        beds  bed_percentage      region
0    AN  2016      380520    2.825081        0.279640       South
1    AP  2017    53060000    0.436072        0.043164       South
2    AR  2018     1683600    1.427893        0.141340  North-East
3    AS  2017    34438756    0.497753        0.049270  North-East
4    BR  2018   122988691    0.094838        0.009388        East
```

Step 23: Row value Changed

Analysing the row one error value find

Tamilnadu first place for healthcare - but one mistake is tamil nadu population count is wrong because of 79788 instead of 7.9 crore was missing

```
[ ]  1 # Update the 'population' column where 'state' is 'TN'
     2 df4.loc[df4['state'] == 'TN', 'population'] = 79788000
     3
```

Step 24: Regions

```
1 df5['region'].value_counts()
```

region	count
North	9
South	8
North-East	8
West	5
East	4
Central	2

Regions wise analysing data

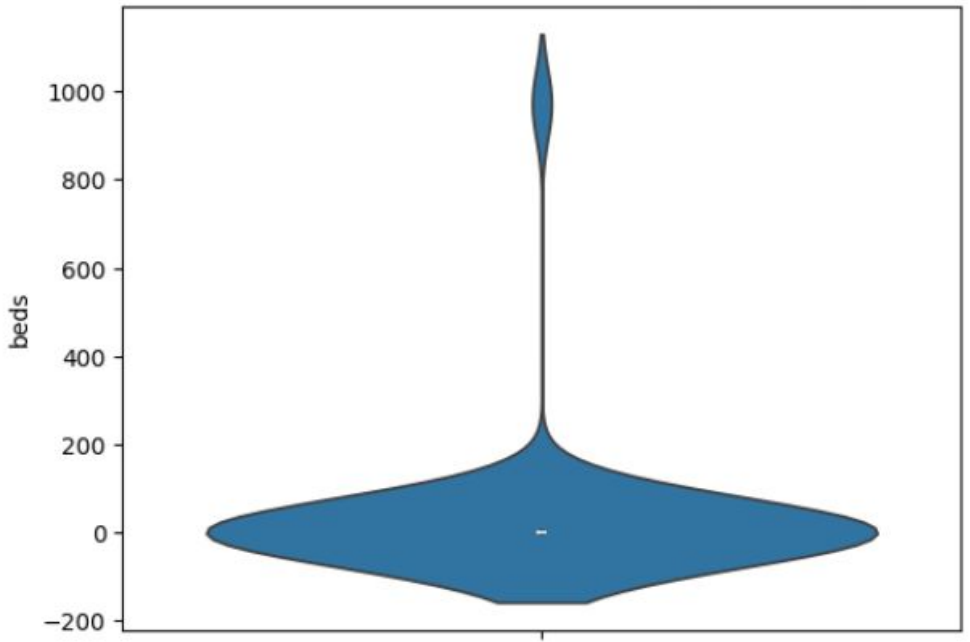
1) North Region have 9 States
2) South have 8 States
3) North-East have 8 States
4) West have 5 States
5) East have 4 States
6) Central have 2 States

Step 25: **Data Visualization**

Use Violin Plot
Analysing a beds most of the states provide near zero to 30 and most of the states have negative only very few state provide have highest number of beds

```
1 sns.violinplot(df5['beds'])
```

```
<Axes: ylabel='beds'>
```



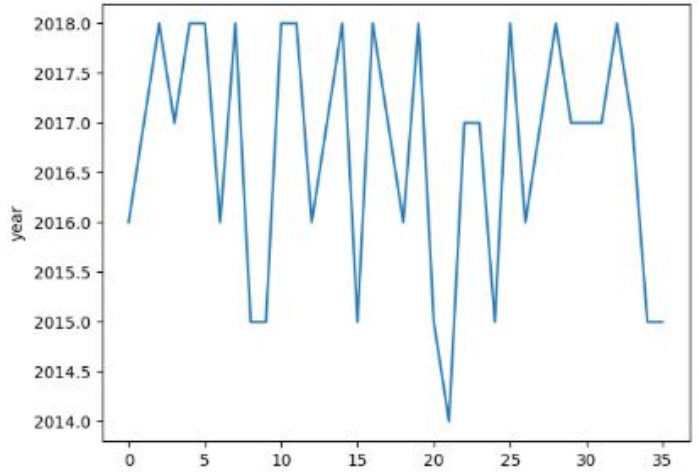
Step 26: **countplot of univarient**

Using Distplot analysing over year when every year population and beds are increase it was calculated by per 100 person for minimize the data

```
[ ]  1 import matplotlib.pyplot as plt # Import the pyplot module
```

```
1
2 print(df5['year'].value_counts()/len(df3)*100)
3 sns.lineplot(df5['year'])
4
```

```
year
2018    33.333333
2017    30.555556
2015    19.444444
2016    13.888889
2014     2.777778
Name: count, dtype: float64
<Axes: ylabel='year'>
```

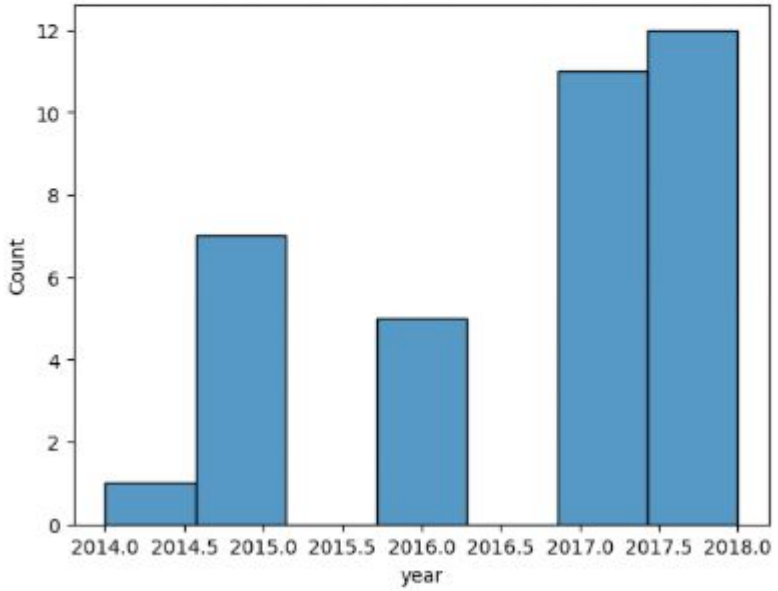


Step 27 : histplot

Same as distplot result

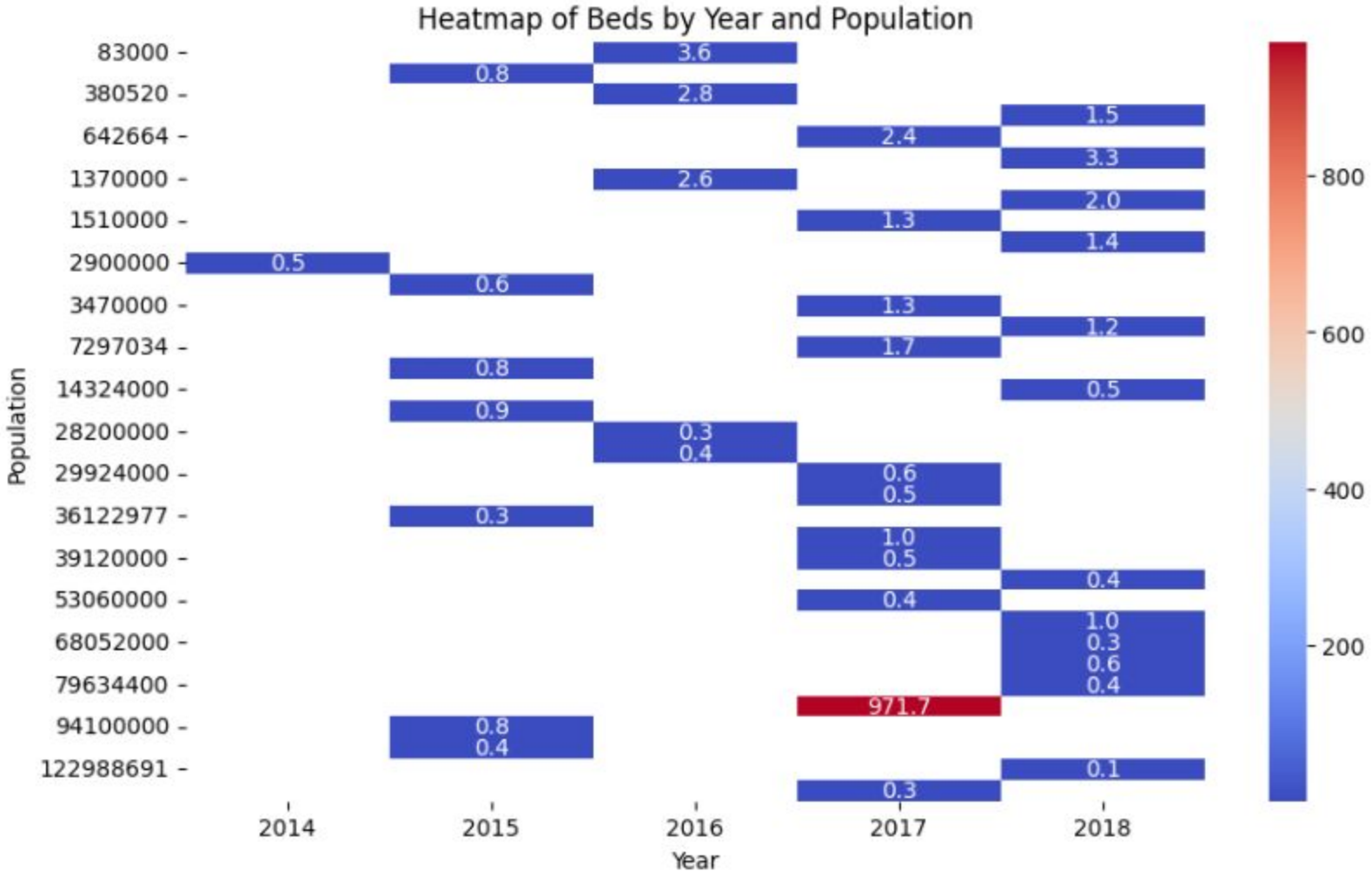
```
1
2 print(df5['year'].value_counts()/len(df3)*100)
3 sns.histplot(df5['year'])
4
```

```
year
2018    33.333333
2017    30.555556
2015    19.444444
2016    13.888889
2014     2.777778
Name: count, dtype: float64
<Axes: xlabel='year', ylabel='Count'>
```



Step 28: Multi Vaarient (Heat Map)

```
[ ]
heatmap_data = df5.pivot(index='population', columns='year', values='beds')

# Create the heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(heatmap_data, annot=True, fmt=".1f", cmap='coolwarm', cbar=True)

# Customize the plot
plt.title("Heatmap of Beds by Year and Population")
plt.xlabel("Year")
plt.ylabel("Population")
plt.show()

```


Heatmap of Beds by Year and Population
Population
Year
83000
380520
642664
1370000
1510000
2900000
3470000
7297034
14324000
28200000
29924000
36122977
39120000
53060000
68052000
79634400
94100000
122988691
2014
2015
2016
2017
2018
3.6
0.8
2.8
1.5
2.4
3.3
2.6
2.0
1.3
1.4
0.5
0.6
1.3
1.2
1.7
0.8
0.5
0.9
0.3
0.4
0.6
0.5
0.3
1.0
0.5
0.4
0.4
1.0
0.3
0.6
0.4
971.7
0.8
0.4
0.1
0.3
800
600
400
200

The heatmap shows the number of beds per year and population. The color intensity represents the number of beds, with darker colors indicating more beds

The number of beds generally increases over the years, with some fluctuations

The heatmap suggests that the number of beds is not directly proportional to the population.

Some areas with smaller populations have a higher number of beds per capita compared to larger populations

analysing this heatmap -: one of this population is 79788000 provide highest values of bed 971 highest value

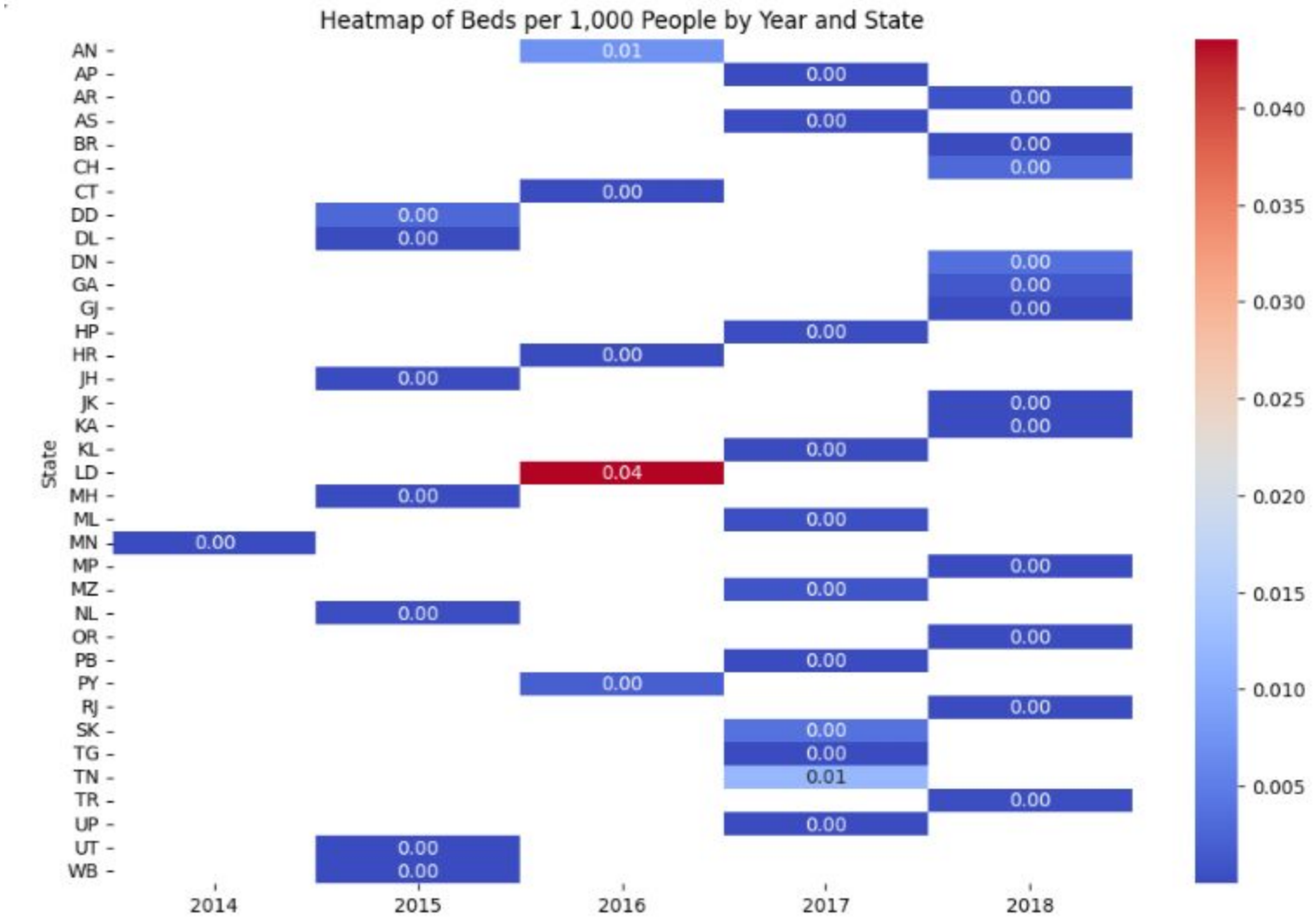
but highest population provide only 0.5 values of bed

least value is 122988691 is provide only 0.1 value of bed

Step: 29

heatmap now compare state wise instead of population to analyse deeply

```
# Calculate beds per 1,000 people
df5['beds_per_1000'] = (df5['beds'] / df5['population']) * 1000

# Pivot the data to create a heatmap-compatible format
heatmap_data = df5.pivot(index='state', columns='year', values='beds_per_1000')

# Create the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(heatmap_data, annot=True, fmt=".2f", cmap='coolwarm', cbar=True)

# Customize the plot
plt.title("Heatmap of Beds per 1,000 People by Year and State")
plt.xlabel("Year")
plt.ylabel("State")
plt.show()
```


Heatmap of Beds per 1,000 People by Year and State
State
AN
AP
AR
AS
BR
CH
CT
DD
DL
DN
GA
GJ
HP
HR
JH
JK
KA
KL
LD
MH
ML
MN
MP
MZ
NL
OR
PB
PY
RJ
SK
TG
TN
TR
UP
UT
WB
2014
2015
2016
2017
2018
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005

The heatmap shows the number of beds per 1,000 people by year and state. The color intensity represents the number of beds, with darker colors indicating more beds

1st place was LD - Ladakh Provide 0.04 value of beds per 1000 people

2nd place was TN and AN - Tamil Nadu and Andaman Nicobar has 0.01 value of beds per 1000 people

compare to 2 union territory 1st place taken state was Tamil Nadu

Conclusion:-

Overall Trend:

The number of beds generally increases over the years, with some fluctuations. The increase is more pronounced in the later years, especially from 2016 to 2018.

State-wise Variations:

There are significant variations in the number of beds across states. Some states consistently have a higher number of beds per 1,000 people compared to others.

Tamil Nadu (TN) stands out with a very high number of beds in 2018, indicating a significant improvement in healthcare infrastructure.

Year-to-Year Changes:

There are significant variations in the number of beds from year to year, even within the same state. This could be due to various factors such as economic conditions, healthcare policies, and natural disasters.

Specific Observations:

In 2014, most states had very few beds per 1,000 people.

In 2015, there was a slight increase in beds in some states.

In 2016, the number of beds increased significantly in several states.

In 2017, there was a slight decrease in beds in some states, while others continued to increase.

In 2018, there was a further increase in beds across most states, with Tamil Nadu (TN) showing a remarkable improvement.

Recommendations:

Continue to invest in healthcare infrastructure: The data suggests that increasing the number of beds can improve healthcare access and outcomes. Focus on states with low bed availability: Targeted interventions are needed to address the disparities in bed availability across states. Monitor bed occupancy rates: Tracking bed occupancy can help optimize resource allocation and identify areas with critical shortages. Conduct further research: Additional data, such as the type of healthcare facilities, bed occupancy rates, and the reasons for bed shortages or surpluses, would be valuable for a more comprehensive analysis.

Step 30:- ABC Inventory

All state, population, bed per 1000 person values all compare to rank wise

For better understanding and deep analysing

```
query = f"SELECT region,state,population,beds,beds_per_1000 FROM {table_name} order by beds_per_1000 desc;"
result = pd.read_sql_query(query, conn)

# Display the results
result
```

	region	state	population	beds	beds_per_1000
0	South	LD	83000	3.614458	4.354769e-02
1	South	TN	79788000	971.725071	1.217884e-02
2	South	AN	380520	2.825081	7.424264e-03
3	North-East	SK	642664	2.427396	3.777084e-03
4	West	DN	412174	1.501793	3.643590e-03
5	North	CH	1136382	3.305227	2.908553e-03
6	West	DD	294410	0.815190	2.768894e-03
7	South	PY	1370000	2.605109	1.901539e-03
8	West	GA	1508556	1.996611	1.323525e-03
9	North-East	MZ	1510000	1.322517	8.758391e-04
10	North-East	AR	1683600	1.427893	8.481189e-04
11	North-East	ML	3470000	1.284438	3.701550e-04

Analyse Max,Avg,Min (bed per 1000 persons)

```
]  1 query1 = f"SELECT state,max(beds_per_1000) FROM {table_name};"
   2 query2 = f"SELECT state,avg(beds_per_1000) FROM {table_name};"
   3 query3 = f"SELECT state,min(beds_per_1000) FROM {table_name};"
   4
   5 result1 = pd.read_sql_query(query1, conn)
   6 result2 = pd.read_sql_query(query2, conn)
   7 result3 = pd.read_sql_query(query3, conn)
   8
   9
  10 # Display the results
  11 print(result1)
  12 print(result2)
  13 print(result3)
```

```
  state  max(beds_per_1000)
0    LD            0.043548
  state  avg(beds_per_1000)
0    AN            0.002301
  state  min(beds_per_1000)
0    BR         7.711115e-07
```

Step: 31: Rank wise

```
# Rank A - Beds per 1000 persons - >0.001 is 1e-2
# Rank B - Beds per 1000 persons - >0.0001 to 0.00099 is 1e-3 to 9e-3
# Rank C - Beds per 1000 persons - >0.00001 to 0.000099 is 1e-4 to 9e-4
# Rank D - Beds per 1000 persons - >0.000001 to 0.0000099 is 1e-5 to 9e-5
# Rank E - Beds per 1000 persons - >0.0000001 to 0.00000099 is 1e-6 to
9e-6
# Rank F - Beds per 1000 persons - >0.00000001 to 0.000000099 is 1e-7 to
9e-7
```

Step 32: SQL Query for Rank

```
query = f"""
SELECT
   state,
   beds_per_1000,
   CASE
       WHEN beds_per_1000 > 0.01 THEN 'A'
       WHEN beds_per_1000 > 0.001 AND beds_per_1000 <= 0.01 THEN 'B'
       WHEN beds_per_1000 > 0.0001 AND beds_per_1000 <= 0.001 THEN 'C'
       WHEN beds_per_1000 > 0.00001 AND beds_per_1000 <= 0.0001 THEN 'D'
       WHEN beds_per_1000 > 0.000001 AND beds_per_1000 <= 0.00001 THEN 'E'
       WHEN beds_per_1000 > 0.0000001 AND beds_per_1000 <= 0.000001 THEN 'F'
       WHEN beds_per_1000 > 0.00000001 AND beds_per_1000 <= 0.0000001 THEN 'G'
       ELSE 'H'
   END AS rank
FROM
   {table_name};
"""

result = pd.read_sql_query(query, conn)

# Display the results
result
```

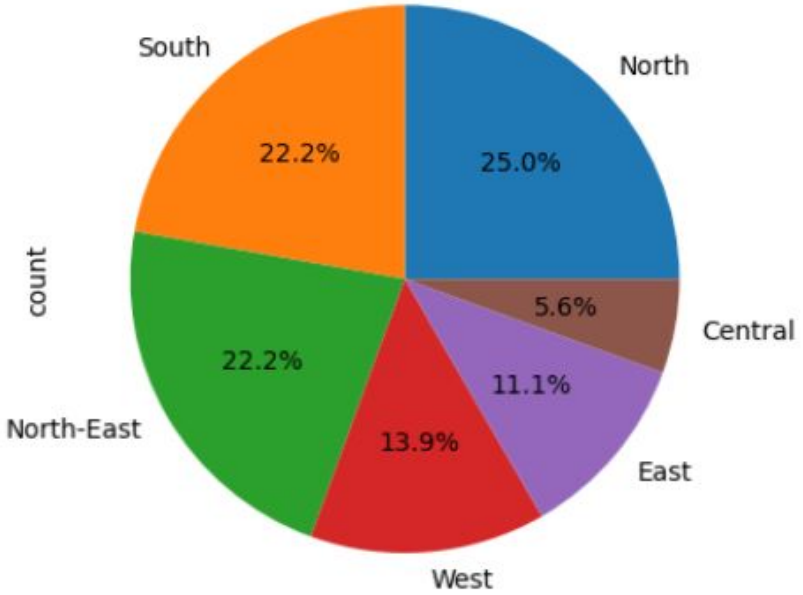
	state	beds_per_1000	rank
0	AN	7.424264e-03	B
1	AP	8.218470e-06	E
2	AR	8.481189e-04	C
3	AS	1.445328e-05	D
4	BR	7.711115e-07	F
5	CH	2.908553e-03	B
6	CT	1.183543e-05	D
7	DN	3.643590e-03	B
8	DD	2.768894e-03	B
9	DL	3.525133e-05	D
10	GA	1.323525e-03	B
11	GJ	4.355787e-06	E
12	HR	1.400272e-05	D
13	HP	2.328595e-04	C
14	JK	3.553519e-05	D
15	JH	8.264435e-06	E
16	KA	1.525704e-05	D

Step: 33

Univarient

```
1 df5['region'].value_counts().plot.pie(autopct='%1.1f%%')
```

```
<Axes: ylabel='count'>
```



North Region has maximum count of states - 1st

south and north-east region has equal number of states - 2nd

west have third largest count of states - 3rd

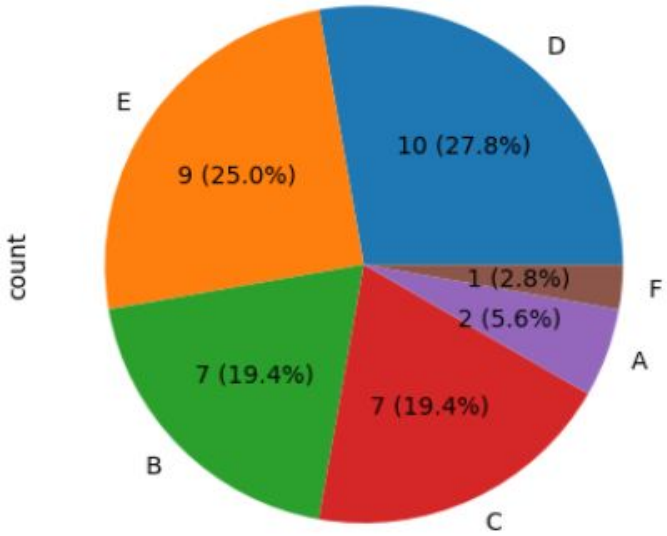
east have 10% count of staes in India - 4th

Central have minimal count of states - 5th

Step: 34 **Rank based on beds per 1000 peoples**

```
1 df5_rank['rank'].value_counts().plot.pie(autopct=lambda p: f'{int(round(p * df5_rank.shape[0] / 100))} ({p:.1f}%)')
2
```

```
<Axes: ylabel='count'>
```



Rank wise States health care focused

Rank A - only 2 states of 5.6% of total states

Rank B - Only 7 states of 19.4% of total states

Rank C - Only 7 states of 19.4% of total states

Rank D - only 9 states of 25%

Rank E - 10 states - 27.8%

Rank F - 1 state - 2.8%

Overall Class

Rank A is maximum number of hospitals - 1st class states

Rank B & C is Average number of hospitals - 2nd class states

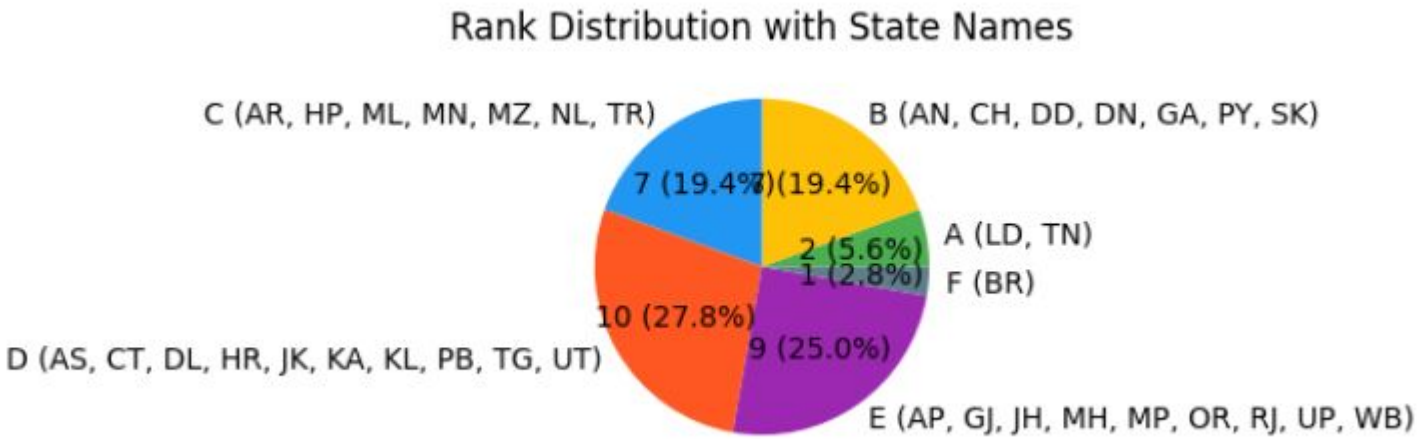
Rank D & E is minimum number of hospitals - 3rd class states

Rank F is Worst condition - worst class

Step: 35 **Rank Based States SQL Query**

```
import matplotlib.pyplot as plt

# Grouping by rank and state
rank_state_counts = df5_rank.groupby(['rank', 'state']).size().reset_index(name='counts')

# Aggregating total counts by rank
rank_counts = rank_state_counts.groupby('rank')['counts'].sum()

# Create a dictionary mapping ranks to state names (combined for multiple states)
rank_state_labels = rank_state_counts.groupby('rank')['state'].apply(lambda states: ', '.join(states)).to_dict()

# Plotting the pie chart
fig, ax = plt.subplots(figsize=(6, 6))
rank_counts.plot.pie(
    ax=ax,
    autopct=lambda p: f'{int(round(p * rank_counts.sum() / 100))} ({p:.1f}%)',
    labels=[f"{rank} ({rank_state_labels.get(rank, '')})" for rank in rank_counts.index],
    colors=['#4CAF50', '#FFC107', '#2196F3', '#FF5722', '#9C27B0', '#607D8B', '#E91E63'],  # Example colors
    title="Rank Distribution with State Names"
)

# Remove the default ylabel
plt.ylabel('')

# Show the plot
plt.tight_layout()
plt.show()
```

[2]


Rank Distribution with State Names
C (AR, HP, ML, MN, MZ, NL, TR)
B (AN, CH, DD, DN, GA, PY, SK)
7 (19.4%)
7 (19.4%)
A (LD, TN)
2 (5.6%)
1 (2.8%)
F (BR)
10 (27.8%)
9 (25.0%)
D (AS, CT, DL, HR, JK, KA, KL, PB, TG, UT)
E (AP, GJ, JH, MH, MP, OR, RJ, UP, WB)

Rank wise HealthCare Denoted States in India

Rank A - LD: Lakshadweep,TN: Tamil Nadu

Rank B -

AN: Andaman and Nicobar Islands CH: Chandigarh DD: Daman and Diu DN: Dadra and Nagar Haveli GA: Goa PY: Puducherry SK: Sikkim

Rank C -

AR: Arunachal Pradesh HP: Himachal Pradesh ML: Meghalaya MN: Manipur MZ: Mizoram NL: Nagaland TR: Tripura

Rank D -

AS: Assam CT: Chhattisgarh DL: Delhi HR: Haryana JK: Jammu and Kashmir KA: Karnataka KL: Kerala PB: Punjab TG: Telangana UT: Uttarakhand

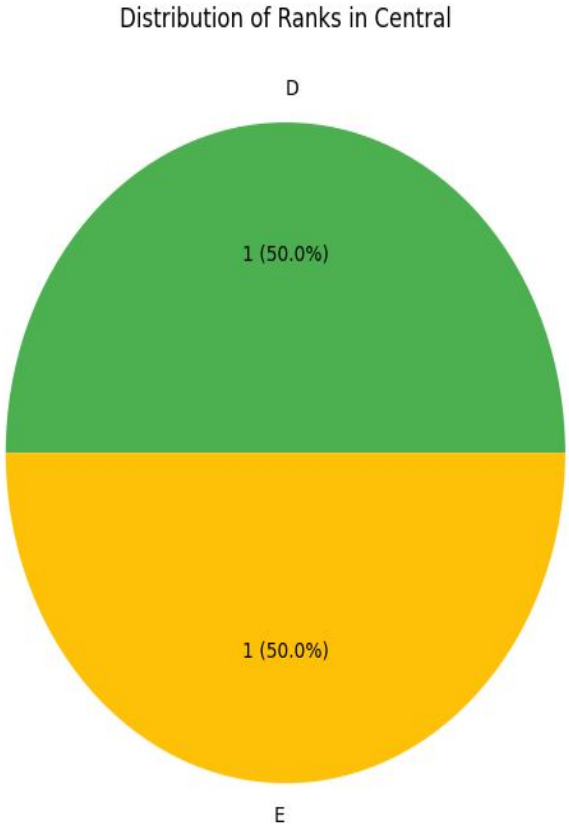
Rank E -

AP: Andhra Pradesh GJ: Gujarat JH: Jharkhand MH: Maharashtra MP: Madhya Pradesh OR: Odisha RJ: Rajasthan UP: Uttar Pradesh WB: West Bengal

Rank F -

BR: Bihar

Step: 36 **Region wise State Rank**

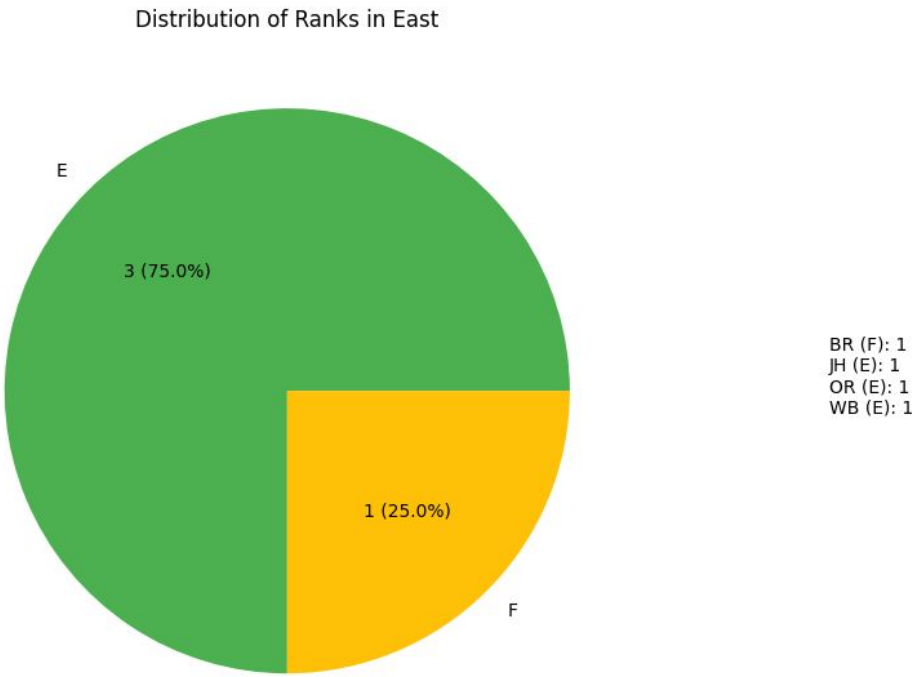


CT (D): 1
MP (E): 1

1)Central Regions

1st-place: CT(**Chhattisgarh**):- Rank D

2nd-place: MP(**Madhya Pradesh**):- Rank E



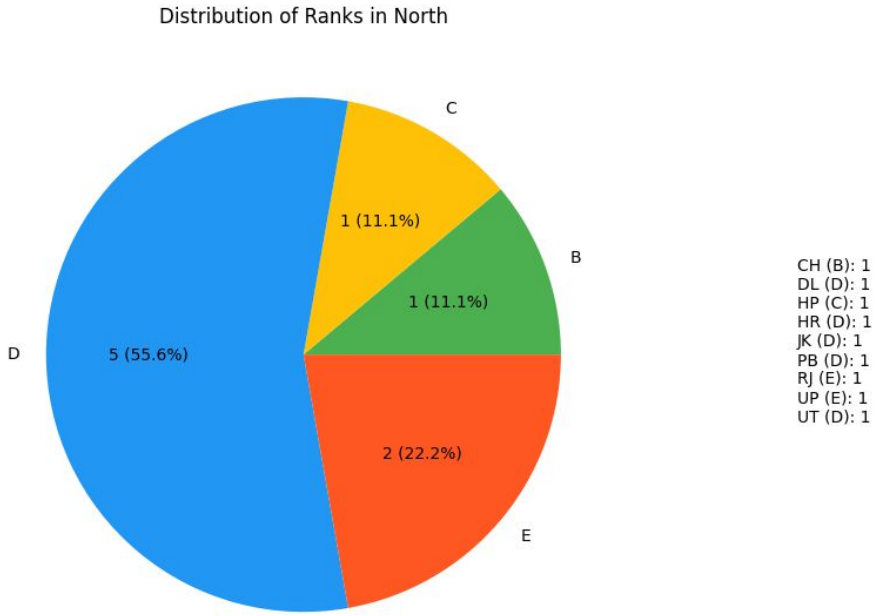
2)East Regions

1st-place:JH(**Jharkhand**):- Rank E

2nd-place:OR(**Orissa**):- Rank E

3rd-place:WB(**West Bengal**):- Rank E

4th-place:BR(**Bihar**):- Rank F



3)North Regions

1st-place:CH(**Chandigarh**):- Rank B

2nd-place:HP(**Himachal Pradesh**):- Rank C

3rd-place:DL(**Delhi**):- Rank D

4th-place: HR(**Haryana**):- Rank D

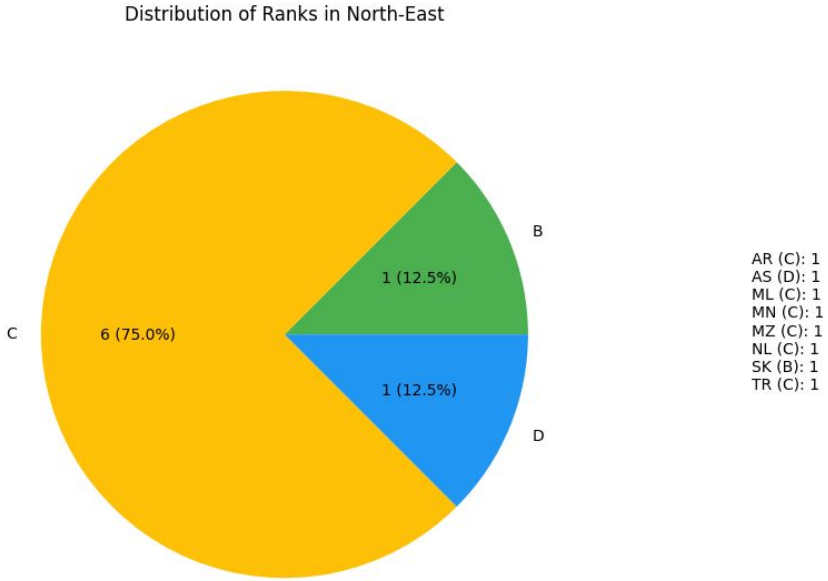
5th-place: JK(**Jammu and Kashmir**):- Rank D

6th-place: PB(**Punjab**):- Rank D

7th-place: UT(**Uttarakhand**):- Rank D

8th-place: UP(**Uttar Pradesh**):- Rank E

9th-place: RJ(**Rajastan**):- Rank E



4)North-East Regions

1st-place:SK(**Sikkim**):- Rank B

2nd-Place:AR(**Arunachal Pradesh**):- Rank C

3rd-Place:ML(**Meghalaya**):- Rank C

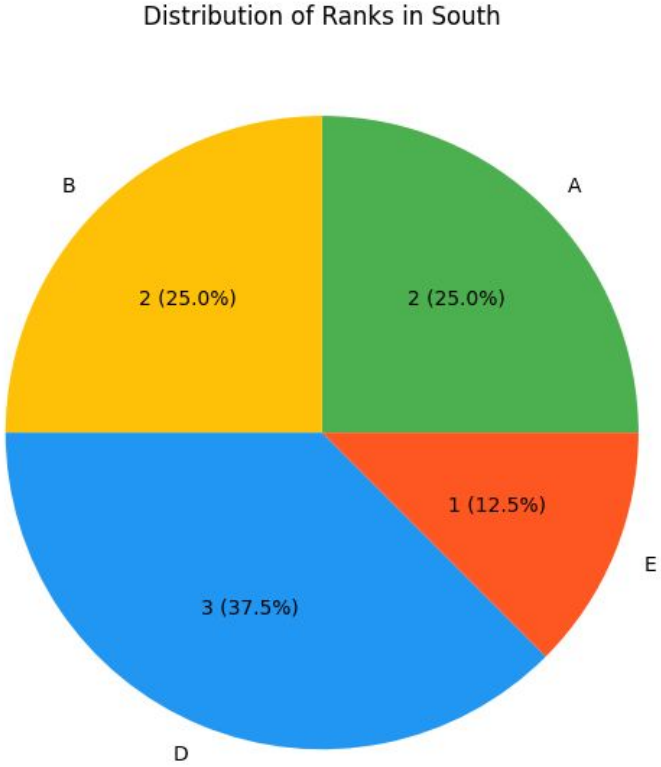
4th-Place:MN(**Manipur**):- Rank C

5th-place:MZ(**Mizoram**):- Rank C

6th-place:NL(**Nagaland**):- Rank C

7th-place:TR(**Tripura**):- Rank C

8th-place:AS(**Assam**):- Rank D



5)South Regions

1st-place:LD(**Lakshadweep**):- Rank A

2nd-place:TN(**Tamil Nadu**):- Rank A

3rd-place:AN(**Andaman & Nicobar Island**):- Rank B

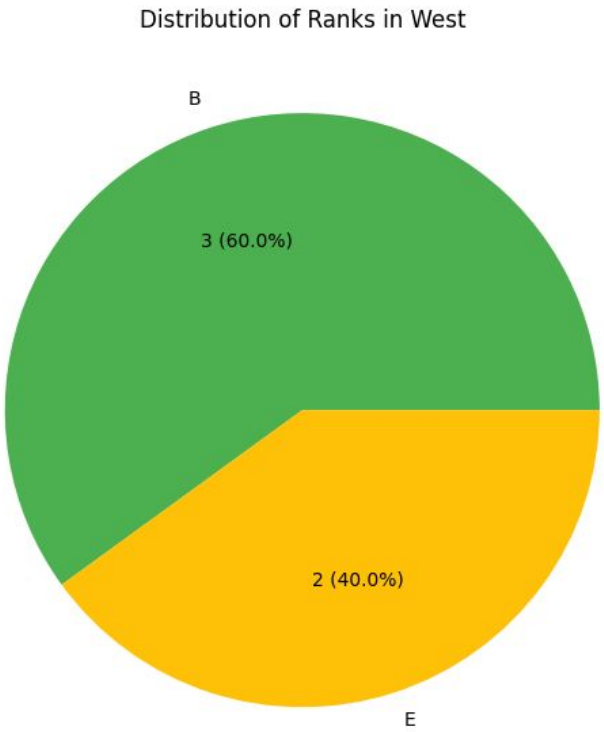
4th-place:PY(**Pondicherry**):- Rank B

5th-place:KA(**Karnataka**):- Rank D

6th-place:KL(**Kerala**):- Rank D

7th-place:TG(**Telangana**):- Rank D

8th-place:AP(**Andhra Pradesh**):- Rank E



6)West Regions

1st-place:DD(**Daman and Diu**):- Rank B

2nd-place:DN(**Dadra and Nagar Haveli**):- Rank B

3rd-place:-GA(**Goa**):- Rank B

4th-place:-GJ(**Gujarat**):- Rank E

5th-place:-MH(**Maharashtra**):- Rank E

Overall Conclusion:-

- Government should concentrate healthcare all over India
- 1st place of healthcare in india is Tamil Nadu -> bed per 1000 person is 0.0024 that means (1 bed for 416,666 people) it's not bad but any pandemic situation happen it's very touch
- But very saddest news is most of the north Indian states have bed per 1000 person is 0.000001 that means
- Bihar is very bad medical facilities Government should focus this Healthcare Department
- India is world largest democratic and high density population country (Education and healthcare must focus)