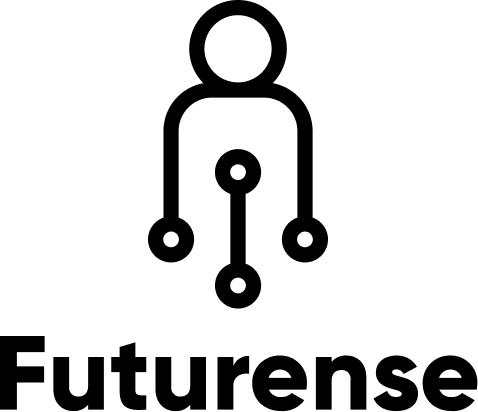
**C:\Users\hp\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B69B6BB2.tmpFuturense Technologies**

****

**Project Report**

**Group 10  
Names:**

1. **Guruakash**
2. **Naveen KM**
3. **Rajulraj Ramesan**
4. **Ruthvik Sharma**
5. **Saurav Kumar Majumdar**

**C:\Users\hp\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B69B6BB2.tmpABSTRACT**

This project focuses on cleaning and processing census and housing data to extract essential insights for the healthcare department's analysis. Initially, irrelevant columns are removed from the census data, and column names are standardized for consistency. State/UT names are normalized to adhere to a uniform format, accounting for variations and new state formations such as Telangana and Ladakh. Missing data in the census dataset is identified and addressed by leveraging information from related fields, and the percentage of missing data is visualized before and after the filling process for assessment.

In parallel, the housing data is processed to extract relevant information, such as household conditions and sanitation facilities. Absolute values for dilapidated households and latrine premises are calculated based on census data and appended to the housing dataset. Discrepancies between district-level data in the census and housing datasets are reported, ensuring a comprehensive analysis framework to support decision-making within the healthcare domain.

**C:\Users\hp\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B69B6BB2.tmpContents**

1. ABSTRACT
2. LIST OF FIGURES/CODES/GRAPHS
3. PROBLEM STATEMENT
4. OBJECTIVE
5. DESCRIPTION
6. CODE
7. SCREENSHOT
8. RESULTS AND CONCLUSION
9. FUTURE SCOPE
10. REFERENCES

**C:\Users\hp\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B69B6BB2.tmpList of Figures/Codes/Graphs**

Please Refer The Screenshot Section.Image 1 : Problem Statement 1 - Keeping The Relevant Data

Image 2 : Problem Statement 2 - Rename The Column Names

Image 3 : Problem Statement 3 - uniformity across datasets

Image 4 : Problem Statement 4 - Rename the State/UT

Image 5 : Problem Statement 5 - Handling missing data

Image 6 : Problem Statement 6 - Save the processed data

Image 7 : Problem Statement 7 - Process The Relevant Data

Image 8 : Problem Statement 8 - Visualize the data

Image 10 : Problem Statement 10 - Fix the header

Image 11 : Problem Statement 11 - Create a function to alter the data to create uniformity

Image 12: Problem Statement 12 - Analyse Healthcare facility disparity

Image 13: Problem Statement 13 - Multi-line header

Image 14: Problem Statement 14 - Data update and code reuse

Image 15: Problem Statement 15 - Government healthcare facility disparity

Image 16: Problem Statement 16 - Gap in number of beds

Image 17: Problem Statement 17 - Hospitals required to meet the standards

**C:\Users\hp\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B69B6BB2.tmpProblem Statement**

1. **Data Selection:**
   * Identified essential columns, including population, literacy rates, and household details.
   * Utilized Pandas functions for efficient extraction and organization of relevant data.
2. Column Standardization:
   * Employed Pandas functions to standardize column names, ensuring clarity and uniformity.
   * Key libraries used: Pandas, NumPy for numerical operations.
3. State and Union Territory Uniformization:
   * Custom Python functions applied for consistent formatting of state and union territory names.
4. Geopolitical Changes Integration:
   * Custom Python functions updated state names post-Telangana (2014) and Ladakh (2019) formations.
5. Data Cleaning and Visualization:
   * Pandas functions used for filtering and organizing columns, retaining only essential information.
   * Matplotlib is employed to create visualizations, comparing missing data percentages before and after the data-filling process.
6. Data Export:
   * Pandas functions are utilized for exporting the refined census data, saved as "census.csv" for future healthcare analyses.
7. Housing Data Integration:
   * Functions employed for reading, processing, and seamlessly integrating housing data into the census dataset.
8. Visualize the following data:
   * Number of households for 100 people
   * Percentage of households that have toilet(s) in premise to the total number of households.
   * Urban to rural population ratio.
9. None
10. Fix the header:
    * The header uses acronyms that are defined in metadata.csv
    * Find the data and rename the headers so that it is more understandable to users who are not familiar with the acronyms.
    * The First cell in the header is missing which should be renamed to State/UT. Rename the other headers in a uniform format.
11. Create a function to alter the data to create uniformity
    * A function should be created to perform this operation. Since the same operation is required for another dataset as well.
    * After the process save the data in a CSV file named “all\_hospitals.csv” in the “Clean\_Data” folder
12. Analyze Healthcare facility disparity:
    * Visually represent how many hospital beds are there for every 10,000 people in each state or union territory. The national value should also be represented in the same visualization such that the value for each state can be compared to it.
13. Multi-line header
    * Import and update the data in a way that it has the following column names
    * State/UT
    * Rural\_Government\_Hospitals
    * Rural\_Government\_Beds
    * Urban\_Government\_Hospitals
    * Urban\_Government\_Beds
    * Last\_Updated
14. Data update and code reuse:
    * The ”Last\_Updated” column contains a date that is in the format DD.MM.YYYY but in the future, the date is required in a different format (YYYY-MM-DD) update the date to the required format.
15. Government healthcare facility disparity:
    * Since the resources are limited, it is required to identify the region which lacks the healthcare facility the most for creating new government hospitals.
    * An idea was suggested\* that first the three States/UTs which have the least amount of beds (in all hospitals government and private) for their population is identified.
    * Among those three the state which has the least number of government hospitals can be recommended for setting up a new government hospital.
16. Gap in number of beds:
    * Visually represent the difference between the expected number of hospital beds and the available number of hospital beds in each State/UT, as well as at the national level (if any).
17. Hospitals required to meet the standards:
    * Find the average number of beds in a government hospital. Divide it by the gap in the number of beds to reach the standards in the region, and round the number to the nearest integer to find the value.
    * Represent the findings visually.

**C:\Users\hp\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B69B6BB2.tmpObjective**

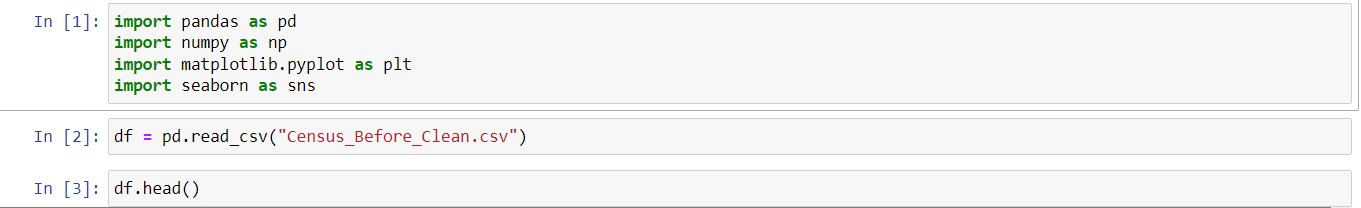
In our project, we improved healthcare-focused census data by refining columns, standardizing names, and addressing missing data. We updated state and union territory names, reflecting real-world changes. The processed census data was, saved as "census.csv,”. Insights from housing data were integrated, forming a comprehensive foundation for healthcare analyses.

**C:\Users\hp\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B69B6BB2.tmpDescription**

1. Data Selection:
   1. Identified essential columns, including population, literacy rates, and household details.
   2. Utilized Pandas functions for efficient extraction and organization of relevant data.
2. Column Standardization:
   1. Employed Pandas functions to standardize column names, ensuring clarity and uniformity.
   2. Key libraries used: Pandas, NumPy for numerical operations.
3. State and Union Territory Uniformization:
   1. Custom Python functions applied for consistent formatting of state and union territory names.
   2. Considered exceptions like "and" for proper capitalization.
4. Geopolitical Changes Integration:
   1. Custom Python functions updated state names post Telangana (2014) and Ladakh (2019) formations.
5. Data Cleaning and Visualization:
   1. Pandas functions used for filtering and organizing columns, retaining only essential information.
   2. Matplotlib employed to create visualizations, comparing missing data percentages before and after the data-filling process.
6. Data Export:
   1. Pandas functions utilized for exporting the refined census data, saved as "census.csv" for future healthcare analyses.
7. Housing Data Integration:
   1. Functions employed for reading, processing, and seamlessly integrating housing data into the census dataset.
8. Visualize the following data:
   1. Number of households for 100 people
   2. Percentage of households that have toilet(s) in premise to the total number of households.
   3. Urban to rural population ratio.
9. Fix the header:
   1. The header uses acronyms that are defined in metadata.csv
   2. Find the data and rename the headers so that it is more understandable to users who are not familiar with the acronyms.
   3. The First cell in the header is missing which should be renamed to State/UT. Rename the other headers in a uniform format.
10. Create a function to alter the data to create uniformity
    1. A function should be created to perform this operation. Since the same operation is required for another dataset as well.
    2. After the process save the data in a CSV file named “all\_hospitals.csv” in the “Clean\_Data” folder
11. Analyze Healthcare facility disparity:
    1. Visually represent how many hospital beds are there for every 10,000 people in each state or union territory. The national value should also be represented in the same visualization such that the value for each state can be compared to it.
12. Multi-line header
    1. Import and update the data in a way that it has the following column names
    2. State/UT
    3. Rural\_Government\_Hospitals
    4. Rural\_Government\_Beds
    5. Urban\_Government\_Hospitals
    6. Urban\_Government\_Beds
    7. Last\_Updated
13. Data update and code reuse:
    1. The ”Last\_Updated” column contains a date that is in the format DD.MM.YYYY but in the future, the date is required in a different format (YYYY-MM-DD) update the date to the required format.
14. Government healthcare facility disparity:
    1. Since the resources are limited, it is required to identify the region which lacks the healthcare facility the most for creating new government hospitals.
    2. An idea was suggested\* that first the three States/UTs which have the least amount of beds (in all hospitals government and private) for their population is identified.
    3. Among those three the state which has the least number of government hospitals can be recommended for setting up a new government hospital.
15. Gap in number of beds:
    1. Visually represent the difference between the expected number of hospital beds and the available number of hospital beds in each State/UT, as well as at the national level (if any).
16. Hospitals required to meet the standards:
    1. Find the average number of beds in a government hospital. Divide it by the gap in the number of beds to reach the standards in the region, and round the number to the nearest integer to find the value.
    2. Represent the findings visually.
17. Hospitals required to meet the standards:
    1. Find the average number of beds in a government hospital. Divide it by the gap in the number of beds to reach the standards in the region, and round the number to the nearest integer to find the value.
    2. Represent the findings visually.

**C:\Users\hp\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B69B6BB2.tmpCode**

**Code 1**

****



req\_columns = ['State name','District name','Population','Male','Female','Literate','Male\_Literate','Female\_Literate','Rural\_Households','Urban\_Households','Households','Age\_Group\_0\_29','Age\_Group\_30\_49','Age\_Group\_50','Age not stated']

df = df[req\_columns]

df.head()

df.describe()

df.info()

**Code 2**

col\_re = { 'State name' : 'State/UT' ,

'District name' : 'District' ,

'Male\_Literate' : 'Literate\_Male' ,

'Female\_Literate' : 'Literate\_Female' ,

'Rural\_Households' : 'Households\_Rural' ,

'Urban\_Households' : 'Households\_Urban' ,

'Age\_Group\_0\_29' : 'Young\_and\_Adult' ,

'Age\_Group\_30\_49' : 'Middle\_Aged' ,

'Age\_Group\_50' : 'Senior\_Citizen' ,

'Age not stated' : 'Age\_Not\_Stated' }

df.rename(columns = col\_re, inplace=True)

df.head()

**Code 3**

df['State/UT'].unique()

def format\_state\_name(state\_name):

    words = state\_name.split()

    formatted\_words = [word.lower() if word.lower() == 'and' else word.capitalize() for word in words]

    return ' '.join(formatted\_words)

df['State/UT'] = df['State/UT'].apply(format\_state\_name)

df['State/UT'].unique()

**Code 4**

df.loc[(df['State/UT'] == "Jammu and Kashmir")]

df.loc[(df['State/UT'] == 'Andhra Pradesh')]

# Read Telangana districts from the text file

telangana\_districts = []

with open('Telangana.txt', 'r') as file:

    telangana\_districts = [line.strip() for line in file]

# Read Ladakh districts from the text file

ladakh\_districts = ['Leh', 'Kargil']

# Rename State/UT names for Telangana and Ladakh districts

df.loc[df['District'].isin(telangana\_districts), 'State/UT'] = 'Telangana'

df.loc[df['District'].isin(ladakh\_districts), 'State/UT'] = 'Laddakh'

df.loc[(df['State/UT'] == "Laddakh") | (df['State/UT'] == "Telangana")]

**Code 5**

df.isnull().sum()

missing\_percentage\_before = (df.isnull().sum() / len(df)) \* 100

print((missing\_percentage\_before))

df['Population'].fillna(df['Male'] + df['Female'], inplace=True)

df['Male'].fillna(df['Population'] - df['Female'], inplace=True)

df['Female'].fillna(df['Population'] - df['Male'], inplace=True)

df['Literate'].fillna(df['Literate\_Male'] + df['Literate\_Female'], inplace=True)

df['Literate'].fillna(df['Literate\_Male'] + df['Literate\_Female'], inplace=True)

df['Literate\_Male'].fillna(df['Literate'] - df['Literate\_Female'], inplace=True)

df['Households\_Rural'].fillna(df['Households'] - df['Households\_Urban'], inplace=True)

df['Households\_Urban'].fillna(df['Households'] - df['Households\_Rural'], inplace=True)

df['Households'].fillna(df['Households\_Rural'] + df['Households\_Urban'], inplace=True)

df['Young\_and\_Adult'].fillna(df['Population'] - df['Middle\_Aged'] - df['Senior\_Citizen'] - df['Age\_Not\_Stated'], inplace=True)

df['Middle\_Aged'].fillna(df['Population'] - df['Young\_and\_Adult'] - df['Senior\_Citizen'] - df['Age\_Not\_Stated'], inplace=True)

df['Senior\_Citizen'].fillna(df['Population'] - df['Young\_and\_Adult'] - df['Middle\_Aged'] - df['Age\_Not\_Stated'], inplace=True)

df['Age\_Not\_Stated'].fillna(df['Population'] - df['Young\_and\_Adult'] - df['Middle\_Aged'] - df['Senior\_Citizen'], inplace=True)

missing\_percentage\_after = (df.isnull().sum() / len(df)) \* 100

print(missing\_percentage\_after)

mpb=[]

mpa=[]

for i in missing\_percentage\_before:

    mpb.append(i)

for i in missing\_percentage\_after:

    mpa.append(i)

plt.figure(figsize=(15, 8))

X = df.columns.tolist()

Y = mpb

Z = mpa

X\_axis = np.arange(len(X))

plt.bar(X\_axis - 0.2, Y, 0.35, label='Before Cleaning', color = 'black')

plt.bar(X\_axis + 0.2, Z, 0.35, label='After Cleaning', color = 'red')

plt.xticks(X\_axis, X, rotation=45, ha='right')

plt.xlabel("Census Data")

plt.ylabel("Null Value")

plt.title("Census Data Cleaning")

plt.legend()

plt.tight\_layout()

plt.show()

**Code 6**

df.to\_csv("C:/Users/Navee/Desktop/Jain Internship Project/DS - Python + SQL + ETL - Healthcare Census/Clean\_data/census.csv", index = False)

**Code 7**

data\_housing = pd.read\_csv("housing.csv")

census\_data = pd.read\_csv("C:/Users/Navee/Desktop/Jain Internship Project/DS - Python + SQL + ETL - Healthcare Census/Clean\_data/census.csv")

merged\_data = pd.merge(data\_housing, census\_data, left\_on='District Name', right\_on='District')

merged\_data.head()

merged\_data['Households\_Rural\_Dilapidated'] = merged\_data['Total Number of Residence Dilapidated'] \* merged\_data['Total Number of Residence households'] / 100

merged\_data['Households\_Rural\_Toilet\_Premise'] = merged\_data['Latrine\_OS'] \* merged\_data['Total Number of Residence households'] / 100

merged\_data['Households\_Urban\_Dilapidated'] = merged\_data['Households\_Urban'] \* merged\_data['Total Number of Dilapidated'] / 100

merged\_data['Households\_Urban\_Toilet\_Premise'] = merged\_data['Households\_Urban'] \* merged\_data['Latrine\_premise'] / 100

merged\_data['Households\_Urban'] = merged\_data['Households'] - merged\_data['Households\_Rural']

merged\_data['Households\_Urban\_Livable'] = merged\_data['Households'] - merged\_data['Households\_Rural']

merged\_data.rename(columns={

                            'Total Number of Residence Livable': 'Households\_Rural\_Livable',

                            'Total Number of Residence Dilapidated': 'Households\_Rural\_Dilapidated',

                            'Latrine\_PSS': 'Households\_Rural\_Toilet\_Premise',

                            'Households': 'Total\_Households'},

                   inplace=True)

merged\_data.fillna(0, inplace=True)

final\_columns = ['State Name','District', 'Households\_Rural','Population', 'Households\_Rural\_Livable', 'Households\_Rural\_Dilapidated', 'Households\_Rural\_Toilet\_Premise',

                 'Households\_Urban', 'Households\_Urban\_Livable', 'Households\_Urban\_Dilapidated', 'Households\_Urban\_Toilet\_Premise']

final\_data = merged\_data[final\_columns]

column\_rename = {

    'District Name': 'District',

    'Households\_Rural': 'Households\_Rural',

    'Households\_Rural\_Livable': 'Households\_Rural\_Livable',

    'Households\_Rural\_Dilapidated': 'Households\_Rural\_Dilapidated',

    'Households\_Rural\_Toilet\_Premise': 'Households\_Rural\_Toilet\_Premise',

    'Households\_Urban': 'Households\_Urban',

    'Households\_Urban\_Livable': 'Households\_Urban\_Livable',

    'Households\_Urban\_Dilapidated': 'Households\_Urban\_Dilapidated',

    'Households\_Urban\_Toilet\_Premise': 'Households\_Urban\_Toilet\_Premise'

}

final\_data.to\_csv('C:/Users/Navee/Desktop/Jain Internship Project/DS - Python + SQL + ETL - Healthcare Census/Clean\_data/housing.csv', index=False)

final\_data.head()

**Code 8**

# Graph 1: Households with Toilet Premise

plt.figure(figsize=(14, 10))

merged\_data.groupby('State/UT')['Households\_Rural\_Toilet\_Premise'].sum().plot(kind='bar', title='Households with Toilet Premise')

plt.xlabel('State')

plt.ylabel('Number of Households with Toilet Premise')

plt.xticks(rotation=90)

plt.tight\_layout()

plt.show()

# Graph 2: Households per 100 people

plt.figure(figsize=(18, 10))

merged\_data.groupby('State/UT')['Total\_Households'].sum().plot(kind='bar', title='Households per 100 people', color ='black')

plt.xlabel('State')

plt.ylabel('Number of Households per 100 people')

plt.xticks(rotation=90)

plt.tight\_layout()

plt.show()

# Graph 3: Urban to Rural Population Ratio

plt.figure(figsize=(18, 10))

merged\_data['Urban\_to\_Rural\_Population\_Ratio'] = merged\_data['Households\_Urban'] / merged\_data['Households\_Rural']

merged\_data.groupby('State/UT')['Urban\_to\_Rural\_Population\_Ratio'].mean().plot(kind='bar', title='Urban to Rural Population Ratio', color = "black")

plt.xlabel('State')

plt.ylabel('Urban to Rural Population Ratio')

plt.xticks(rotation=90)

plt.tight\_layout()

plt.show()

**Code 9**

# Calculate the absolute difference between rural and urban households

merged\_data['Households\_Rural\_Difference'] = merged\_data['Households\_Rural'] - merged\_data['Households\_Urban']

# Calculate the percentage difference

merged\_data['Percentage\_Difference'] = (merged\_data['Households\_Rural\_Difference'] / merged\_data['Households\_Rural']) \* 100

# Identify districts with major differences in rural households

major\_difference\_districts = merged\_data[abs(merged\_data['Percentage\_Difference']) > 10]

# Print districts with major differences in rural households for each state

for state, state\_data in major\_difference\_districts.groupby('State/UT'):

    print(f"Districts with major differences in rural households in {state}:")

    for index, row in state\_data.iterrows():

        print(f"District: {row['District Name']}, Percentage Difference: {row['Percentage\_Difference']:.2f}%")

    print("\n")

**Code 10**

hospital\_data=pd.read\_csv("hospitals.csv")

meta\_data=pd.read\_csv("metadata.csv")

hospital\_data.head()

meta\_data.head()

hospital\_data.columns = ['State/UT', 'Number of Primary Health Centers(PHCs),', 'Community Health Centers(CHCs),', 'Sub-District/Divisional Hospitals(SDHs)', 'District Hospitals(DHs)', 'Hospitals', 'HospitalBeds']

hospital\_data.head(5)

**Code 11**

def clean\_state\_names(name):

    name = name.title().replace('&', 'and')

    words = name.split()

    camel\_case = []

    for word in words:

        if word.lower() == 'and':

            camel\_case.append('and')

        else:

            camel\_case.append(word.capitalize())

    return ' '.join(camel\_case)

hospital\_data['State/UT'] = hospital\_data['State/UT'].apply(clean\_state\_names)

hospital\_data.head(5)

hospital\_data.to\_csv("C:/Users/Navee/Desktop/Jain Internship Project/DS - Python + SQL + ETL - Healthcare Census/Clean\_data/hospitals.csv", index = False)

**Code 12**

census\_data = pd.read\_csv('C:/Users/Navee/Desktop/Jain Internship Project/DS - Python + SQL + ETL - Healthcare Census/Clean\_data/census.csv')

df = hospital\_data.merge(census\_data, on='State/UT')

df.columns

df = df.dropna(subset=['HospitalBeds', 'Population'])

df = df.fillna(0)

total\_beds = df['HospitalBeds'].astype(int)

population = df['Population'].astype(int)

beds\_per\_10000 = (total\_beds / population) \* 10000

beds\_per\_10000

national\_avg = beds\_per\_10000.sum() / len(beds\_per\_10000)

national\_avg

plt.figure(figsize=(10, 5))

plt.bar(df['State/UT'], beds\_per\_10000, color ="black")

plt.axhline(y=national\_avg, color='g', linestyle='-')

plt.xlabel('State')

plt.ylabel('Beds per 10,000 people')

plt.title('Hospital Beds per 10,000 People by State')

plt.xticks(rotation=90)

plt.annotate(f'National Average = {national\_avg:.2f}', xy=(0.5, 0.95),

             xycoords='axes fraction', horizontalalignment='center',

             verticalalignment='top')

plt.show()

**Code 13**

# Read the government hospitals data with multi-line header

hospital\_data = pd.read\_csv("government\_hospitals.csv",header=[0, 1])

# Rename the columns to the required format

hospital\_data.columns = ['State/UT', 'Rural\_Government\_Hospitals', 'Rural\_Government\_Beds', 'Urban\_Government\_Hospitals', 'Urban\_Government\_Beds', 'Last\_Updated']

# Save the updated data to a new CSV file

hospital\_data.to\_csv('C:/Users/Navee/Desktop/Jain Internship Project/DS - Python + SQL + ETL - Healthcare Census/Clean\_data/government\_hospitals\_clean.csv', index=False)

#Display the first few rows of the updated data

hospital\_data.head()

**Code 14**

def fix\_state\_names(name):

    # Convert to lowercase

    name = name.lower()

    # Capitalize first letter of each word except 'and'

    name = ' '.join(word.capitalize() if word != 'and' else word for word in name.split())

    return name

hospital\_data = pd.read\_csv("government\_hospitals.csv",header=[0, 1])

hospital\_data.columns = ['State/UT', 'Rural\_Government\_Hospitals', 'Rural\_Government\_Beds', 'Urban\_Government\_Hospitals', 'Urban\_Government\_Beds', 'Last\_Updated']

hospital\_data['Last\_Updated'] = pd.to\_datetime(hospital\_data['Last\_Updated'], format='%d.%m.%Y').dt.strftime('%Y-%m-%d')

hospital\_data['State/UT'] = hospital\_data['State/UT'].apply(fix\_state\_names)

hospital\_data.to\_csv('C:/Users/Navee/Desktop/Jain Internship Project/DS - Python + SQL + ETL - Healthcare Census/Clean\_data/government\_hospitals\_clean.csv', index=False)

hospital\_data.head()

**Code 15**

# Load the hospital data

hospital\_data = pd.read\_csv('C:/Users/Navee/Desktop/Jain Internship Project/DS - Python + SQL + ETL - Healthcare Census/Clean\_data/government\_hospital.csv')

# Load the population data

population\_data = pd.read\_csv("C:/Users/Navee/Desktop/Jain Internship Project/DS - Python + SQL + ETL - Healthcare Census/Clean\_data/census.csv")

# Merge hospital and population data

merged\_data = pd.merge(hospital\_data, population\_data, on='State/UT')

# Calculate total number of beds (government + private) per population

merged\_data['Total\_Beds\_Per\_1000\_People'] = (merged\_data['Rural\_Government\_Beds'] + merged\_data['Urban\_Government\_Beds']) / (merged\_data['Population'] / 1000)

# Find the three states with the least amount of beds per population

top\_states = merged\_data.nsmallest(3, 'Total\_Beds\_Per\_1000\_People')

# Sort by the number of government hospitals

recommended\_state = top\_states.nsmallest(3, 'Rural\_Government\_Hospitals')

# Display the recommended state for setting up a new government hospital

print("Recommended State/UT for setting up a new government hospital:")

print(recommended\_state[['State/UT', 'Rural\_Government\_Hospitals']])

**Code 16**

hospital\_data = pd.read\_csv('C:/Users/Navee/Desktop/Jain Internship Project/DS - Python + SQL + ETL - Healthcare Census/Clean\_data/government\_hospital.csv')

population\_data = pd.read\_csv("C:/Users/Navee/Desktop/Jain Internship Project/DS - Python + SQL + ETL - Healthcare Census/Clean\_data/census.csv")

# Calculate expected beds per State/UT

population\_data['Total\_Population'] = population\_data['Male'] + population\_data['Female']

population\_data['Expected\_Beds'] = population\_data['Total\_Population'] / 1000 \* 3

# Calculate total available beds per State/UT

hospital\_data['Total\_Beds'] = hospital\_data['Rural\_Government\_Hospitals'] \* hospital\_data['Rural\_Government\_Beds'] + \

                               hospital\_data['Urban\_Government\_Hospitals'] \* hospital\_data['Urban\_Government\_Beds']

# Calculate the difference between expected and available beds

bed\_shortage = population\_data.groupby('State/UT')['Expected\_Beds'].sum() - hospital\_data.groupby('State/UT')['Total\_Beds'].sum()

# Sort the data by values in ascending order

bed\_shortage\_sorted = bed\_shortage.sort\_values(ascending=True)

# Reverse the order of index and values

bed\_shortage\_sorted = bed\_shortage\_sorted.iloc[::-1]

# Visualize the difference in number of beds

plt.figure(figsize=(10, 8))

plt.bar(bed\_shortage\_sorted.index, bed\_shortage\_sorted.values, color='black')

plt.xlabel('State/UT')

plt.ylabel('Bed Shortage')

plt.title('Difference between Expected and Available Hospital Beds')

plt.xticks(rotation=90)

plt.gca().invert\_yaxis()  # Invert the y-axis to plot upside down vertically

plt.show()

**Code 17**

# Load the hospital data

hospital\_data = pd.read\_csv('C:/Users/Navee/Desktop/Jain Internship Project/DS - Python + SQL + ETL - Healthcare Census/Clean\_data/government\_hospital.csv')

# Calculate the total number of beds required to meet WHO standards (hypothetical situation)

# Assume WHO standard is 3 beds per 1000 people

population\_data = pd.read\_csv("C:/Users/Navee/Desktop/Jain Internship Project/DS - Python + SQL + ETL - Healthcare Census/Clean\_data/census.csv")

total\_population = population\_data['Population'].sum()

total\_beds\_required = total\_population \* 3 / 1000

# Calculate the current total number of beds in government hospitals

total\_beds\_in\_government\_hospitals = hospital\_data['Rural\_Government\_Beds'].sum() + hospital\_data['Urban\_Government\_Beds'].sum()

# Calculate the gap in the number of beds required to meet WHO standards

beds\_gap = total\_beds\_required - total\_beds\_in\_government\_hospitals

# Find the average number of beds in a government hospital

total\_government\_hospitals = hospital\_data['Rural\_Government\_Hospitals'].sum() + hospital\_data['Urban\_Government\_Hospitals'].sum()

average\_beds\_per\_hospital = total\_beds\_in\_government\_hospitals / total\_government\_hospitals

# Calculate the number of additional government hospitals needed

additional\_hospitals\_needed = round(beds\_gap / average\_beds\_per\_hospital)

# Calculate the number of additional hospitals needed in each state/UT

hospital\_data['Additional\_Hospitals\_Needed'] = round((beds\_gap / total\_beds\_in\_government\_hospitals) \* hospital\_data[['Rural\_Government\_Hospitals', 'Urban\_Government\_Hospitals']].sum(axis=1))

# Visualize the findings

plt.figure(figsize=(18, 8))

plt.bar(hospital\_data['State/UT'], hospital\_data['Additional\_Hospitals\_Needed'], color='black')

plt.xlabel('State/UT')

plt.ylabel('Additional Hospitals Needed')

plt.title('Additional Government Hospitals Needed to Meet WHO Standards')

plt.xticks(rotation=90)

plt.show()

# Display national-level findings

print("Additional Government Hospitals Needed at National Level:", additional\_hospitals\_needed)

import pandas as pd

import matplotlib.pyplot as plt

# Load the hospital data

hospital\_data = pd.read\_csv('C:/Users/Navee/Desktop/Jain Internship Project/DS - Python + SQL + ETL - Healthcare Census/Clean\_data/government\_hospital.csv')

# Drop the "India" row from hospital\_data

hospital\_data = hospital\_data[hospital\_data['State/UT'] != 'INDIA']

# Calculate the total number of beds required to meet WHO standards (hypothetical situation)

# Assume WHO standard is 3 beds per 1000 people

population\_data = pd.read\_csv("C:/Users/Navee/Desktop/Jain Internship Project/DS - Python + SQL + ETL - Healthcare Census/Clean\_data/census.csv")

total\_population = population\_data['Population'].sum()

total\_beds\_required = total\_population \* 3 / 1000

# Calculate the current total number of beds in government hospitals

total\_beds\_in\_government\_hospitals = hospital\_data['Rural\_Government\_Beds'].sum() + hospital\_data['Urban\_Government\_Beds'].sum()

# Calculate the gap in the number of beds required to meet WHO standards

beds\_gap = total\_beds\_required - total\_beds\_in\_government\_hospitals

# Find the average number of beds in a government hospital

total\_government\_hospitals = hospital\_data['Rural\_Government\_Hospitals'].sum() + hospital\_data['Urban\_Government\_Hospitals'].sum()

average\_beds\_per\_hospital = total\_beds\_in\_government\_hospitals / total\_government\_hospitals

# Calculate the number of additional government hospitals needed

additional\_hospitals\_needed = round(beds\_gap / average\_beds\_per\_hospital)

# Calculate the number of additional hospitals needed in each state/UT

hospital\_data['Additional\_Hospitals\_Needed'] = round((beds\_gap / total\_beds\_in\_government\_hospitals) \* hospital\_data[['Rural\_Government\_Hospitals', 'Urban\_Government\_Hospitals']].sum(axis=1))

# Visualize the findings

plt.figure(figsize=(18, 8))

plt.bar(hospital\_data['State/UT'], hospital\_data['Additional\_Hospitals\_Needed'], color='black')

plt.xlabel('State/UT')

plt.ylabel('Additional Hospitals Needed')

plt.title('Additional Government Hospitals Needed to Meet WHO Standards')

plt.xticks(rotation=90)

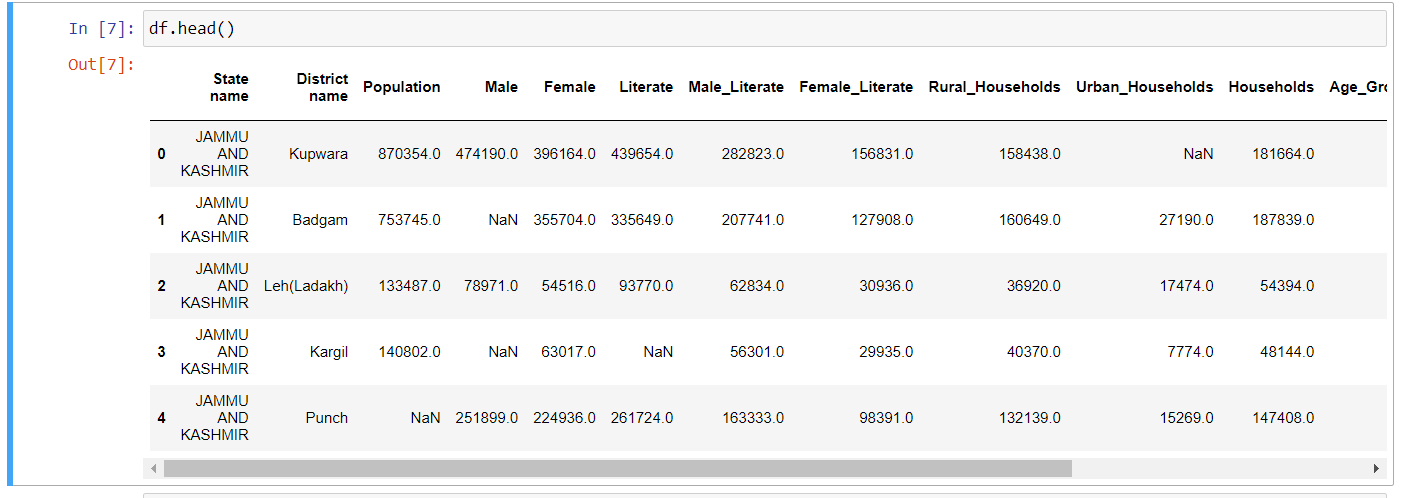
plt.show()

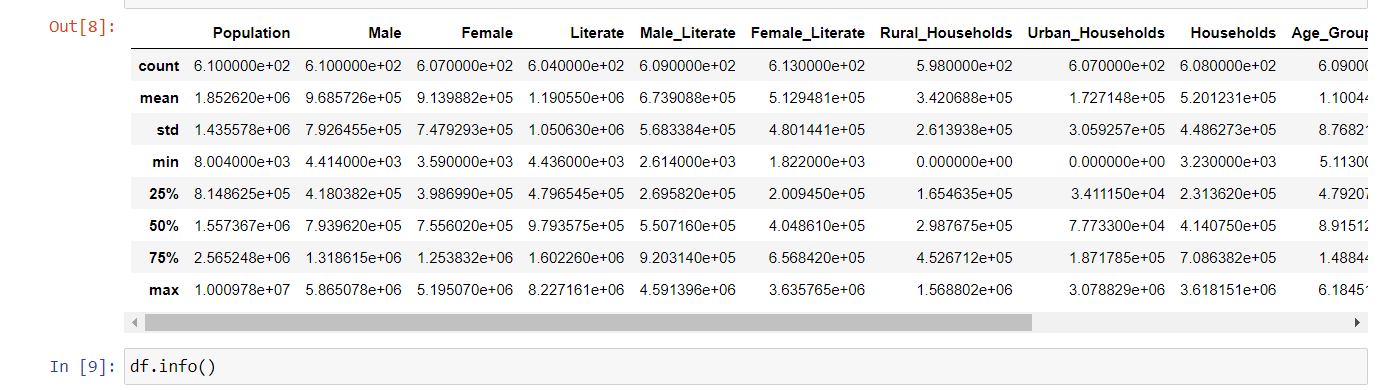
# Display national-level findings

print("Additional Government Hospitals Needed at National Level:", additional\_hospitals\_needed)

**C:\Users\hp\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B69B6BB2.tmpScreenshot**

Image:1





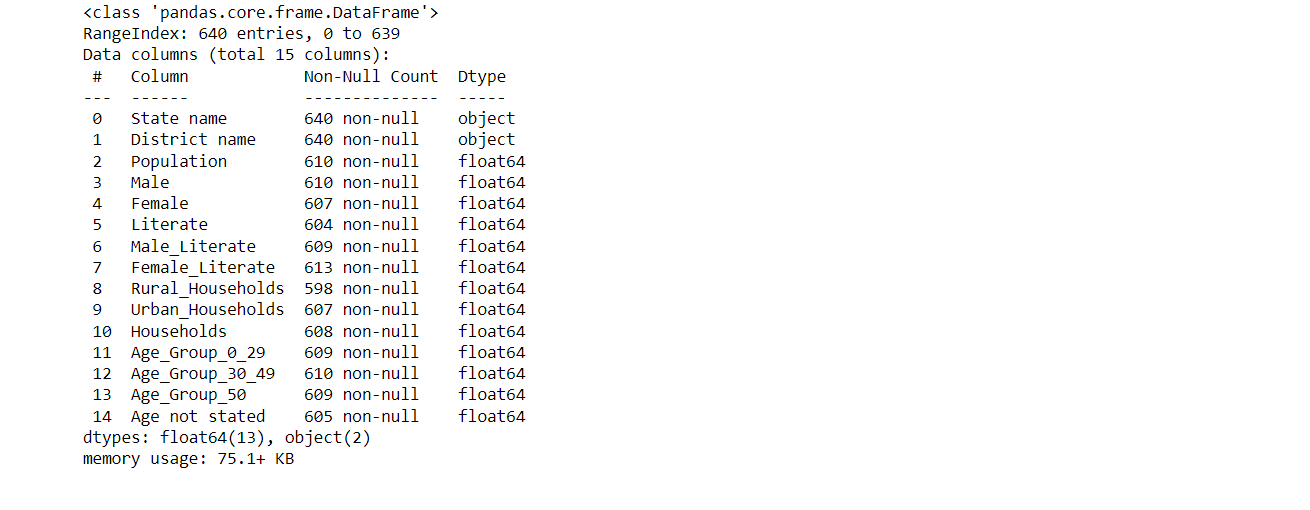


Image:2

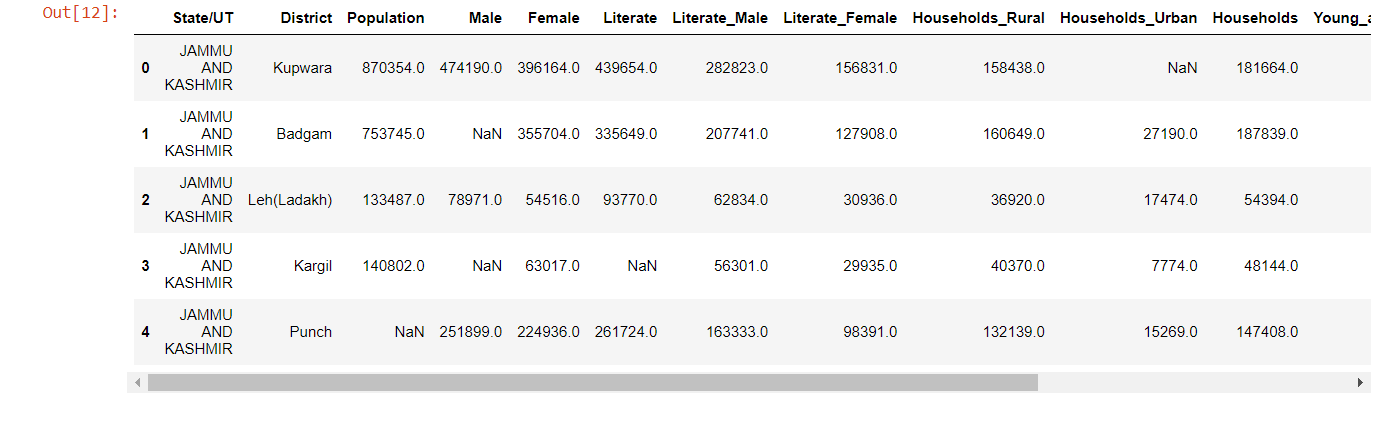


Image:3



Image:4

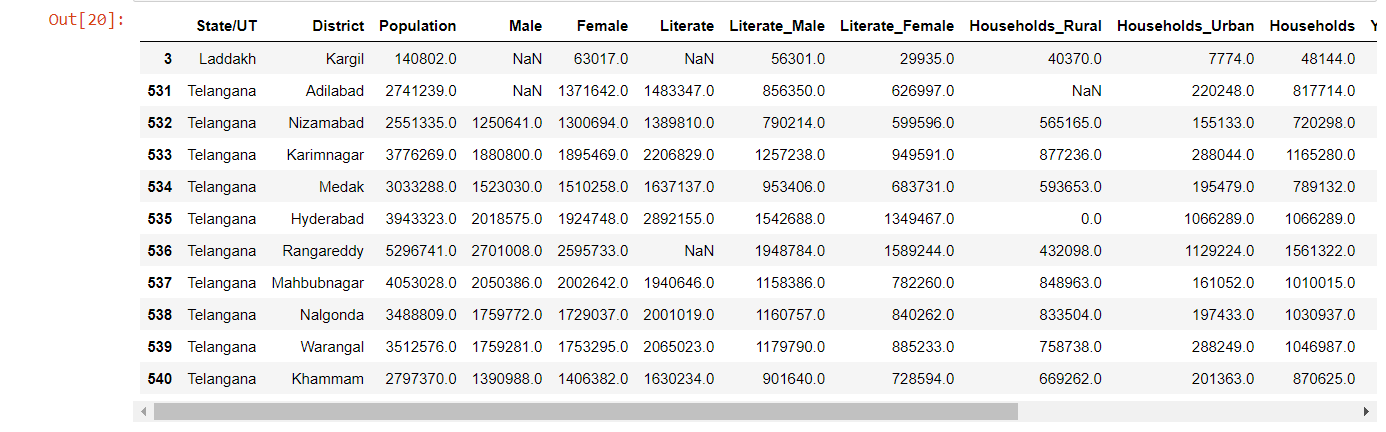


Image:5

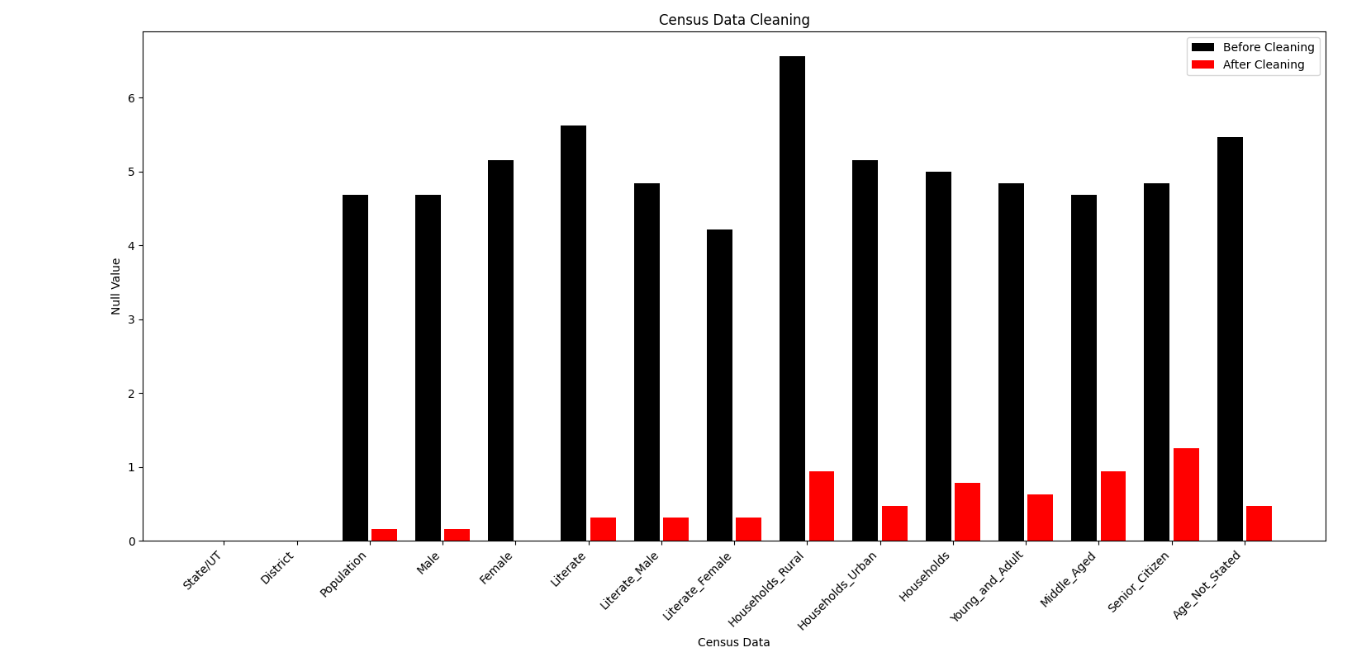


Image:6

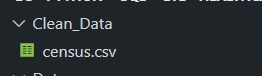
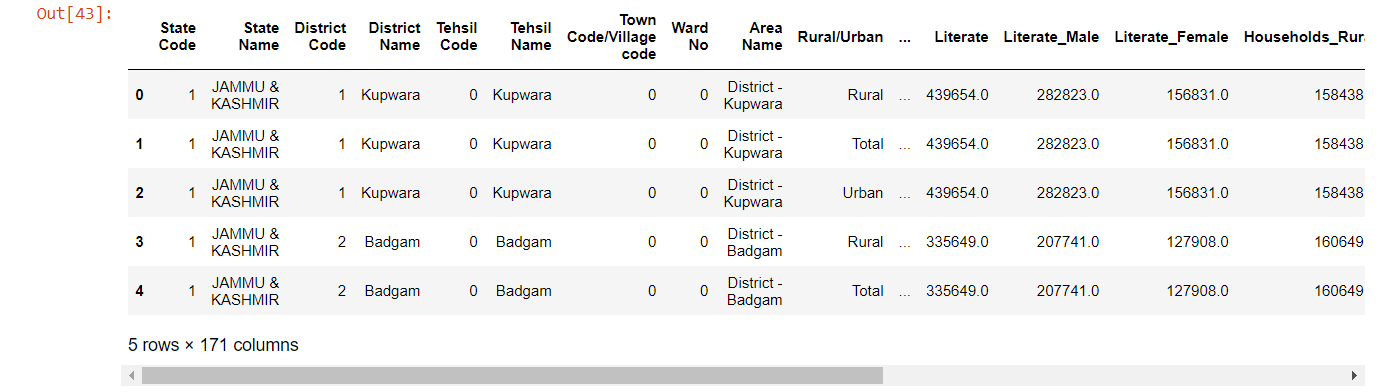


Image:7



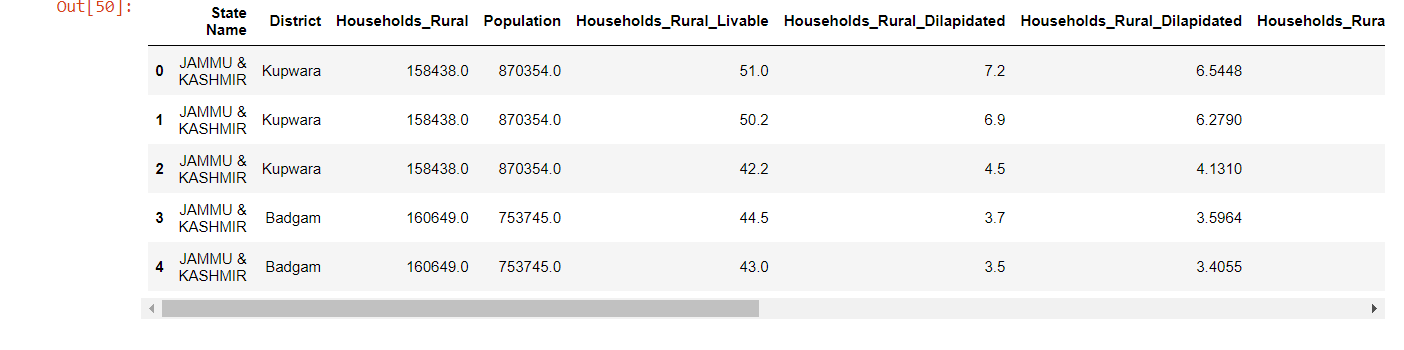


Image:8

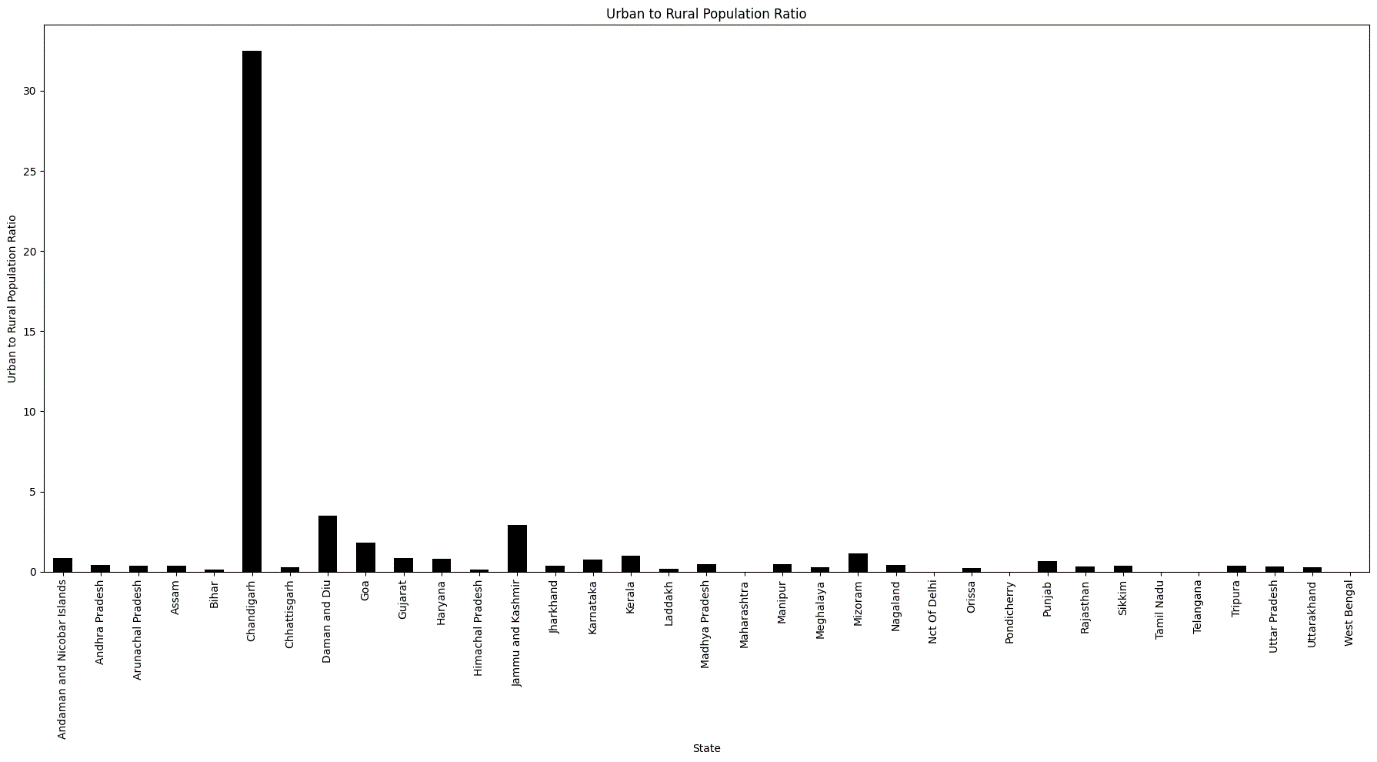
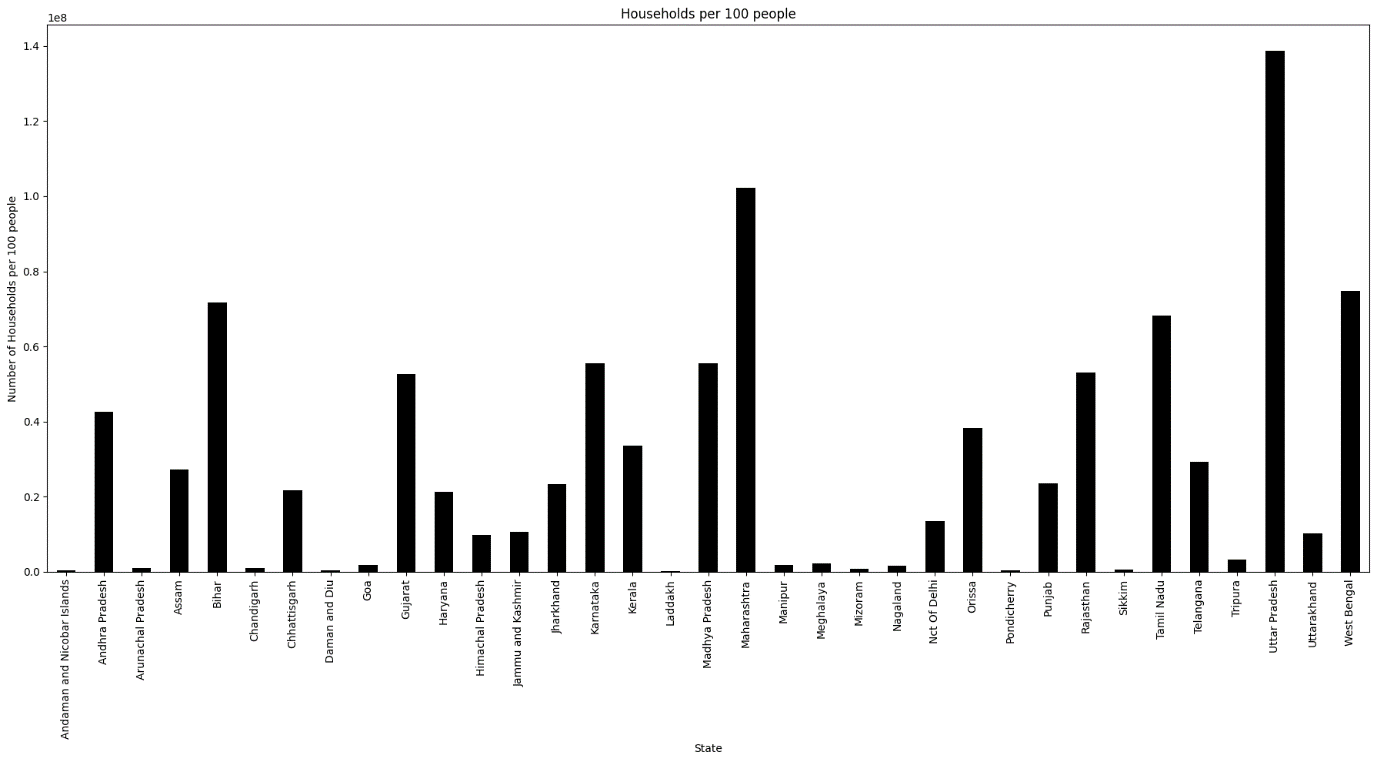
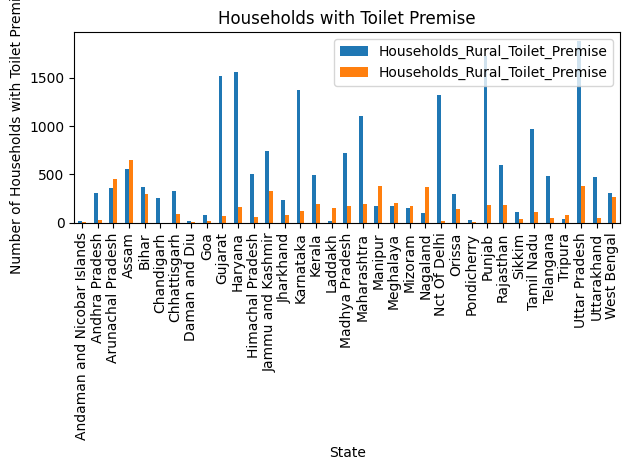


Image:9

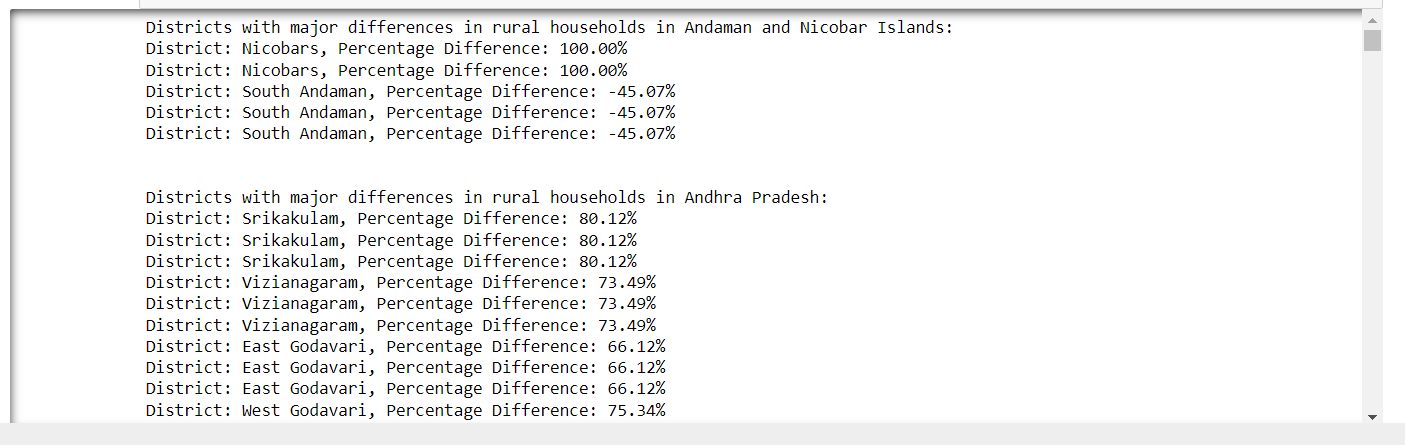
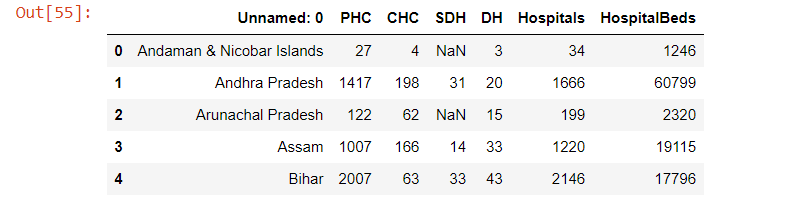
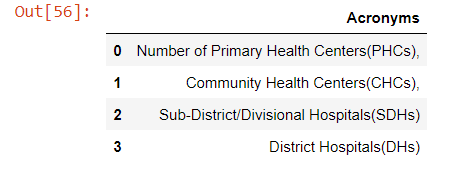


Image:10





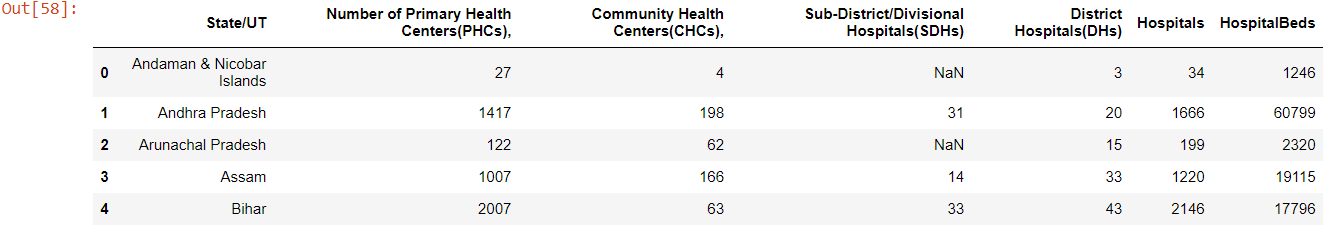


Image:11

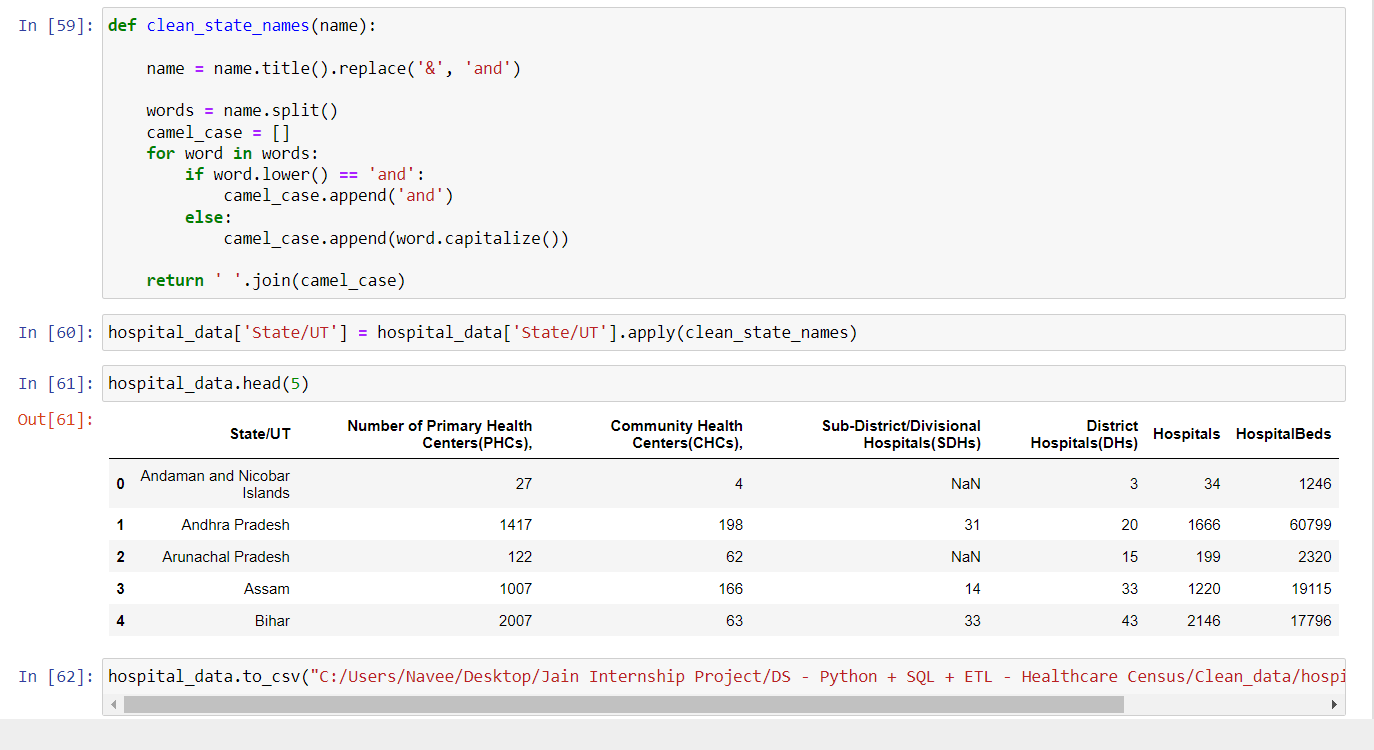


Image:12

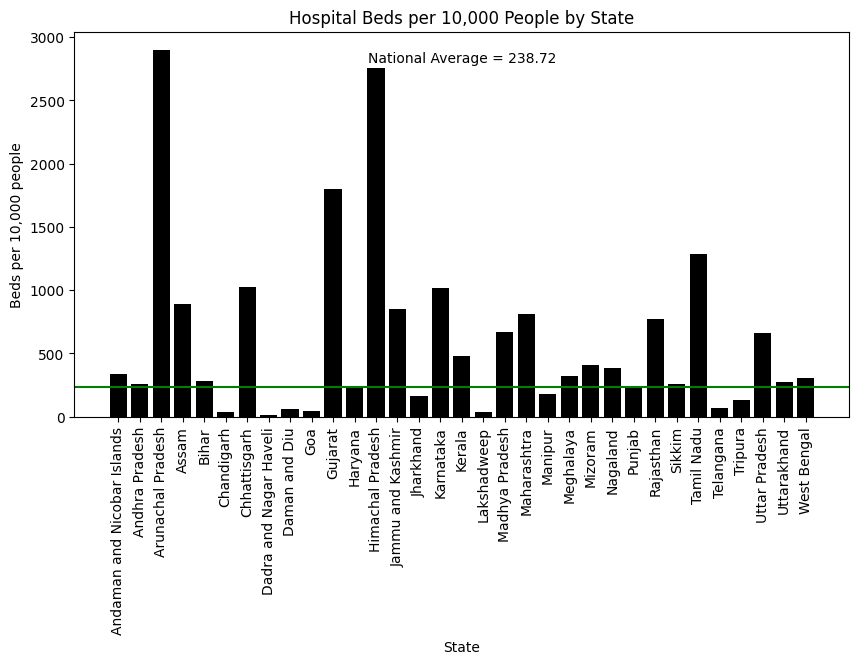


Image:13

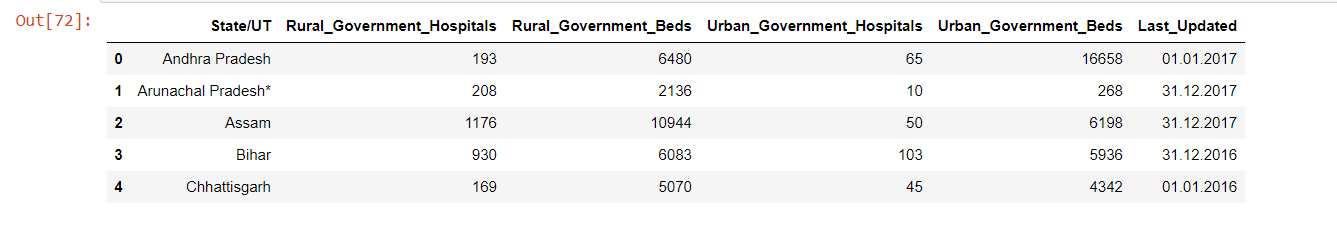


Image:14

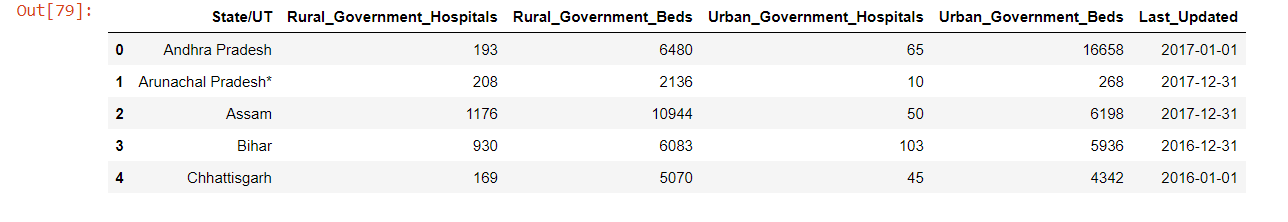


Image:15

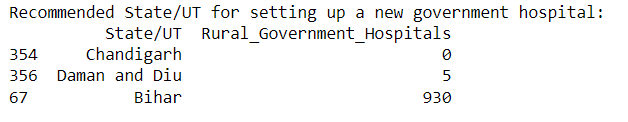


Image:16

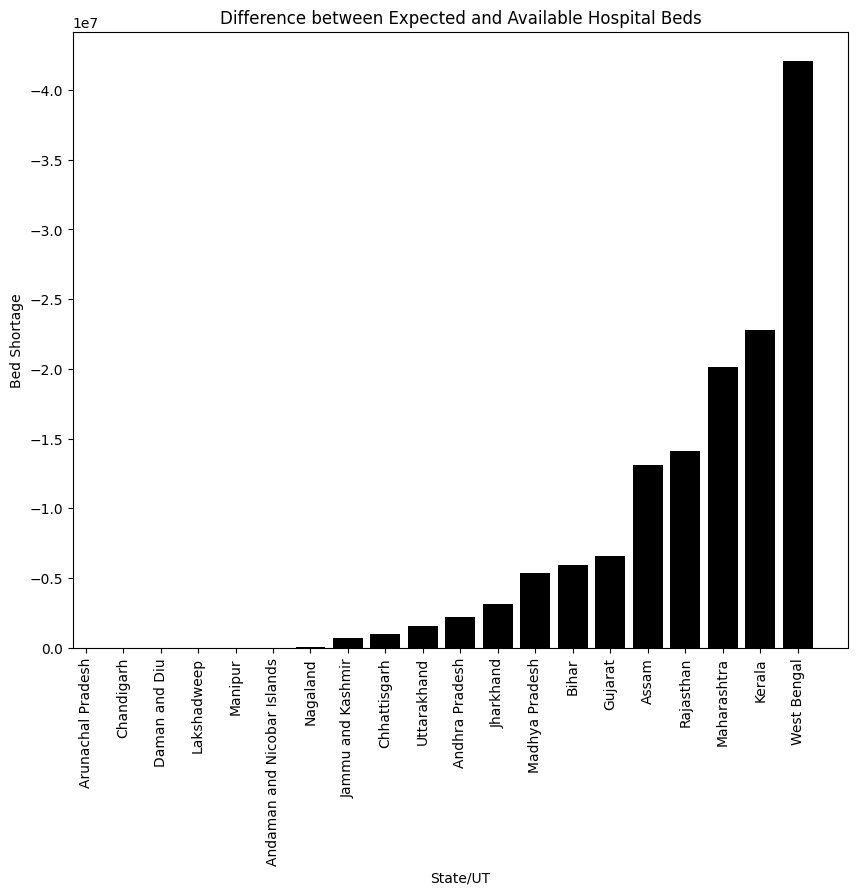
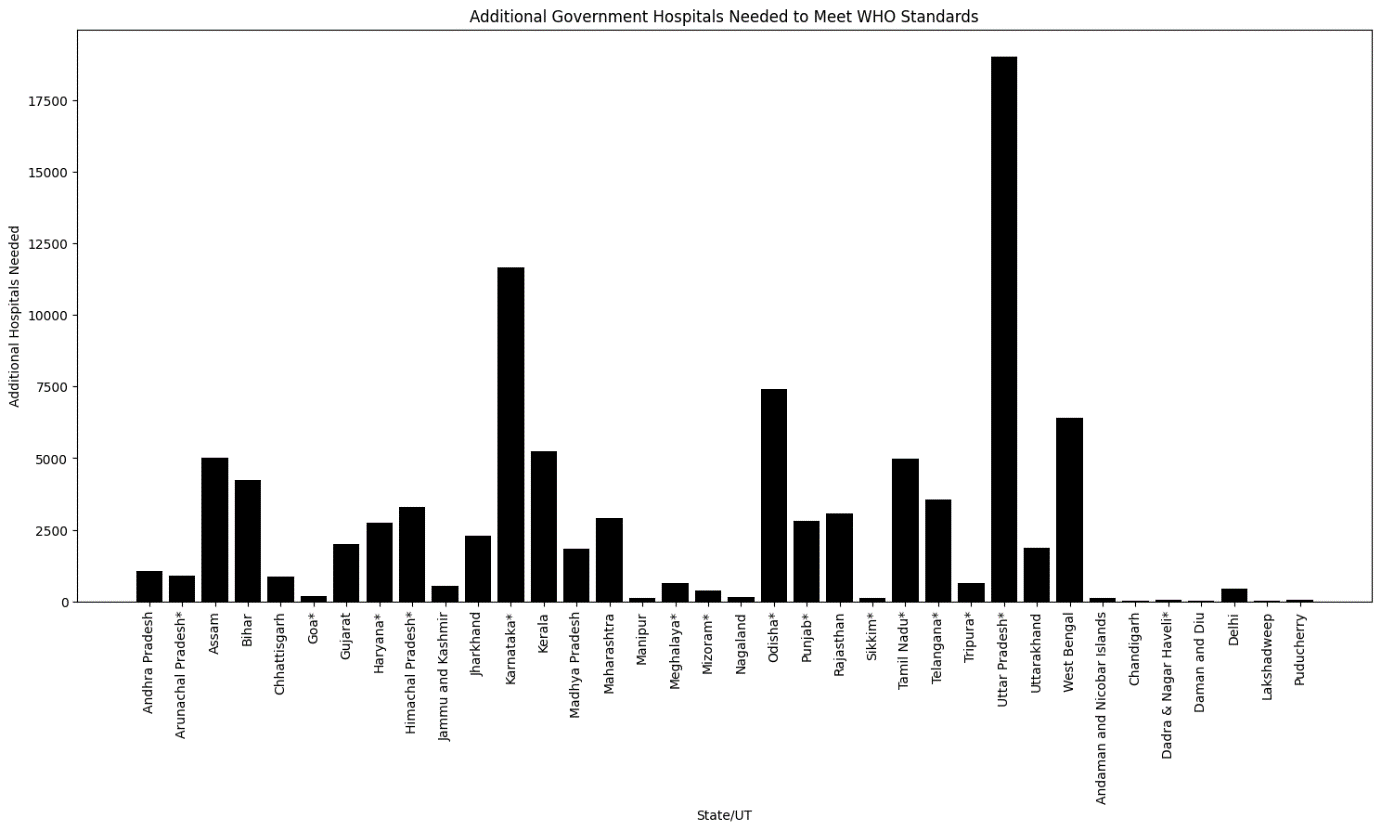


Image:17



**C:\Users\hp\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B69B6BB2.tmpResults and Conclusion**

* 1. Refining the Data:

We worked on refining the dataset, focusing on essential aspects like population, literacy rates, and household details.

* 2. Consistent Naming:

Successfully made the dataset more organized by standardizing column and state/union territory names.

* 3. Handling Changes:

Integrated changes from Telangana (2014) and Ladakh (2019) formations seamlessly.

* 4. Complete Data Set:

Addressed missing values, resulting in a much more complete dataset for thorough healthcare analysis.

* 5. Visual Insight:

Created visuals comparing missing data percentages, providing a clear before-and-after view.

* 6. Easy Accessibility:

Saved the refined census data as "census.csv," making it easily accessible for future healthcare analyses.

* 7. Holistic Foundation:

Extended our analysis to housing data, enriching the dataset for comprehensive healthcare research.

* 8. Project Conclusion:

Successfully achieved our goals, creating a clean, standardised dataset ready for valuable insights in healthcare research.

**C:\Users\hp\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B69B6BB2.tmpFuture Scope**

* 1. Implementation of Remaining Problem Statements:

Execute the pending 20 problem statements, enhancing the dataset comprehensively and addressing diverse healthcare aspects.

* 2. Advanced Data Visualization for Housing Insights:

Develop intuitive visualizations for housing data, illustrating metrics like households per 100 people, toilet facilities, and urban-to-rural population ratios.

* 3. Extended Comparative Analyses - Hospitals:

Expand analyses by integrating insights from hospital data, aiding Aliah in identifying states requiring urgent additional hospital beds.

* 4. Efficient Header Management for Hospital Data:

Streamline hospital data headers using automated functions, ensuring uniformity and easy integration with census and housing datasets.

* 5. Comprehensive Healthcare Disparity Analysis:

Investigate healthcare facility disparities, recommending new government hospitals based on the least bed-to-population ratios and governmental resources.

**C:\Users\hp\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B69B6BB2.tmpReferences**

* Pandas Documentation. (n.d.). Retrieved from <https://pandas.pydata.org/pandas-docs/stable/index.html>
* Matplotlib Documentation. (n.d.). Retrieved from <https://matplotlib.org/stable/contents.html>
* NumPy Documentation. (n.d.). Retrieved from <https://numpy.org/doc/stable/>
* OpenAI Generative Pre-Trained Model - 3.5. (n.d.). Retrieved from<https://chat.openai.com/>
* Google Bard AI Experiment. (n.d.). Retrieved from <https://bard.google.com/chat>
* Jupyter Notebook Documentation. (n.d.). Retrieved from <https://jupyter.org/documentation>
* Stack Overflow Community. (n.d.). Retrieved from<https://stackoverflow.com/>
* GeeksforGeeks. (n.d.). Retrieved from<https://www.geeksforgeeks.org/>
* Data Cleaning and Preprocessing with Python. (n.d.). Retrieved from <https://towardsdatascience.com/data-cleaning-with-python-and-pandas-detecting-missing-values-3e9c6ebcf78b>

C:\Users\hp\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B69B6BB2.tmp