VIRGINIA COMMONWEALTH UNIVERSITY

STATISTICAL ANALYSIS & MODELLING

A5 – Perceptual Mapping using NSSO dataset

State: **RAJASTHAN**

Using R and Python

Submitted by

UJWAL P

V01107757

Date of Submission: 15/07/2024

Table of Contents

1. Introduction

1.1. About the Dataset

The NSSO-Consumption dataset is a comprehensive collection of data on consumption for all Indian states and union territories. It offers detailed insights into the consumption habits of various commodities such as grains, oils, fruits, vegetables, and more.

Moreover, the dataset includes basic demographic information for each sample, enabling a thorough analysis of consumption patterns across different regions of India.

All the data, including states and union territories, is provided in numerical format, making it easily accessible for statistical analysis.

1.2. Objective

To visualize and create a perceptual map that shows total consumption across various districts,

including district names

To illustrate consumption per district with the names of the districts.

To plot any chosen variable within Rajasthan.

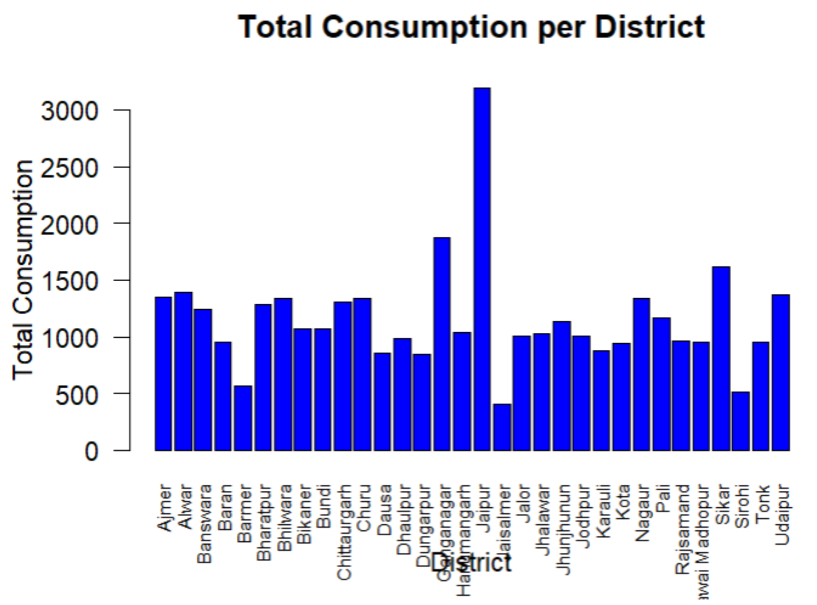
1.3. Business Significance

Creating a histogram of district-wise consumption data allows businesses to visualize and comprehend consumption patterns at a detailed level. This insight is vital for pinpointing high-demand areas and customizing products or services to suit local preferences. By mapping consumption across the state and highlighting each district's consumption levels, companies can easily spot regions with varying consumption rates. This approach facilitates strategic market penetration by targeting areas with untapped potential. Understanding district-wise consumption patterns enables businesses to allocate resources more effectively and efficiently.

2. Results in R and Python

2.1 Output and Interpretation

Bar Plot of Total Consumption per District



Interpretation:

X-axis (District): The x-axis lists the names of the districts in Rajasthan.
Y-axis (Total Consumption): The y-axis represents the total consumption values.

Bars: Each bar represents a district, and its height corresponds to the total consumption for that district.

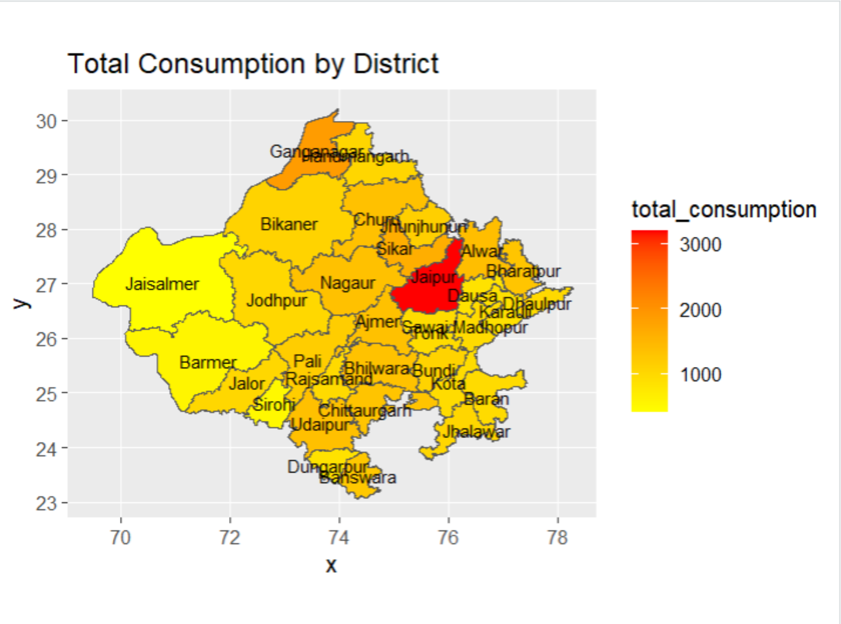
Districts like Ganganagar, Jaipur, and Jaisalmer have higher bars, indicating higher total consumption.
Other districts such as Dhaulpur, Sawai Madhopur, and Karauli have shorter bars, indicating lower total consumption.

Business Implications:
Consumption Disparity: There's a noticeable disparity in consumption across districts.

High Consumption Areas: Districts with high consumption can be targeted for more intensive market activities.

Low Consumption Areas: These districts may require further analysis to understand the reasons behind lower consumption.

Geographic Map of Total Consumption by District



Interpretation:

The map highlights the geographical distribution of consumption across Rajasthan.
Jaipur (in red) shows the highest consumption.

Districts like Barmer and Jaisalmer show lower consumption values (in yellow).

Consumption patterns appear to correlate with geographical locations.

The map allows businesses to identify regions with higher or lower consumption easily, aiding in strategic planning.

Companies can allocate resources more effectively by targeting high-consumption areas and investigating the needs of low-consumption areas.

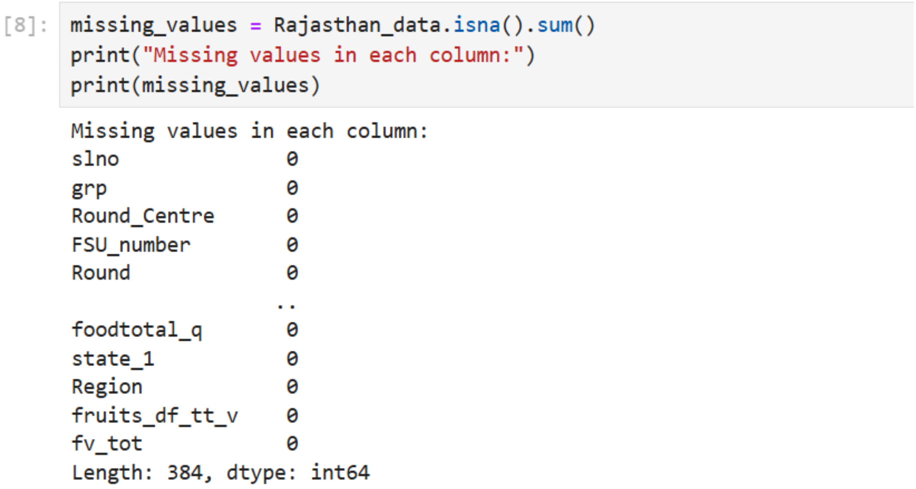
Focus on high-potential markets like Jaipur and Ganganagar, tailoring products and services to meet the specific demands.

Optimize supply chain management to reduce costs and improve efficiency, ensuring high-demand areas are well-stocked.

Collaborate with government and stakeholders to develop infrastructure in low-consumption areas to stimulate economic growth.

Adapt product offerings or marketing strategies to cater to different consumer preferences in various districts.
These visualizations provide valuable insights into consumption patterns across Rajasthan, enabling businesses to make informed decisions and strategic plans.

Missing Value:



```
missing_values = Rajasthan_data.isna().sum()
print("Missing values in each column:")
print(missing_values)
```

```
Missing values in each column:
slno             0
grp              0
Round_Centre     0
FSU_number       0
Round            0
                ..
foodtotal_q      0
state_1          0
Region           0
fruits_df_tt_v   0
fv_tot           0
Length: 384, dtype: int64
```

Interpretation:

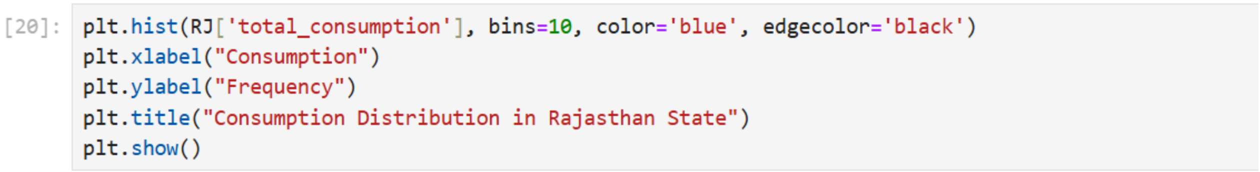
The output shows that for each column in the Rajasthan_data DataFrame, the count of missing values is 0.

This means that every column in the DataFrame is fully populated with data, and there are no missing entries.

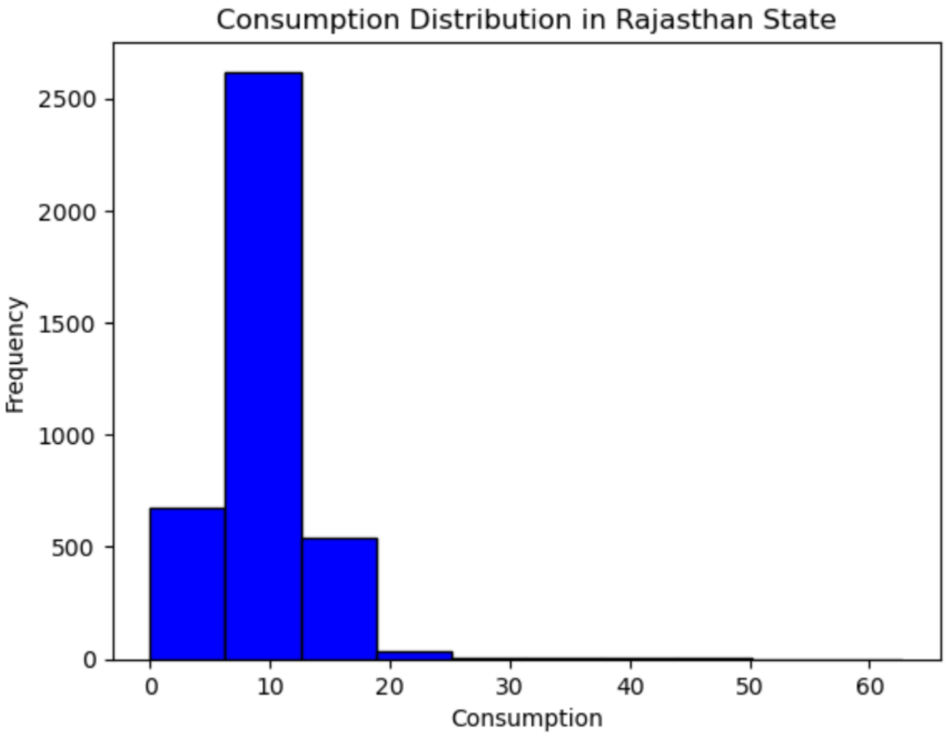
Each line shows a column name followed by 0, indicating no missing values in that column.

This confirms that the dataset is complete with no missing data, which is a good sign for further analysis and visualization.

2.2 Histogram indicating the consumption District wise



```
[20]: plt.hist(RJ['total_consumption'], bins=10, color='blue', edgecolor='black')
      plt.xlabel("Consumption")
      plt.ylabel("Frequency")
      plt.title("Consumption Distribution in Rajasthan State")
      plt.show()
```



Histogram Interpretation:

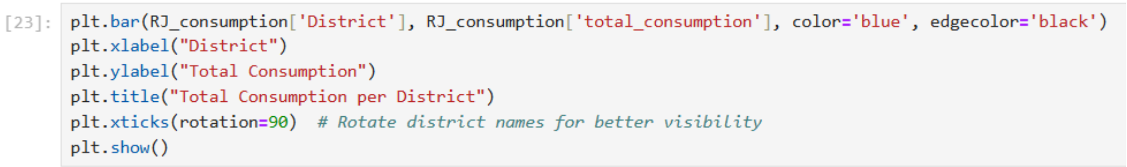
X-axis (Consumption): The x-axis represents the total consumption values. These are divided into intervals (bins).
Y-axis (Frequency): The y-axis shows the frequency (count) of districts falling into each consumption bin.

The height of each bar represents the number of districts that have a total consumption within the range of that bin.

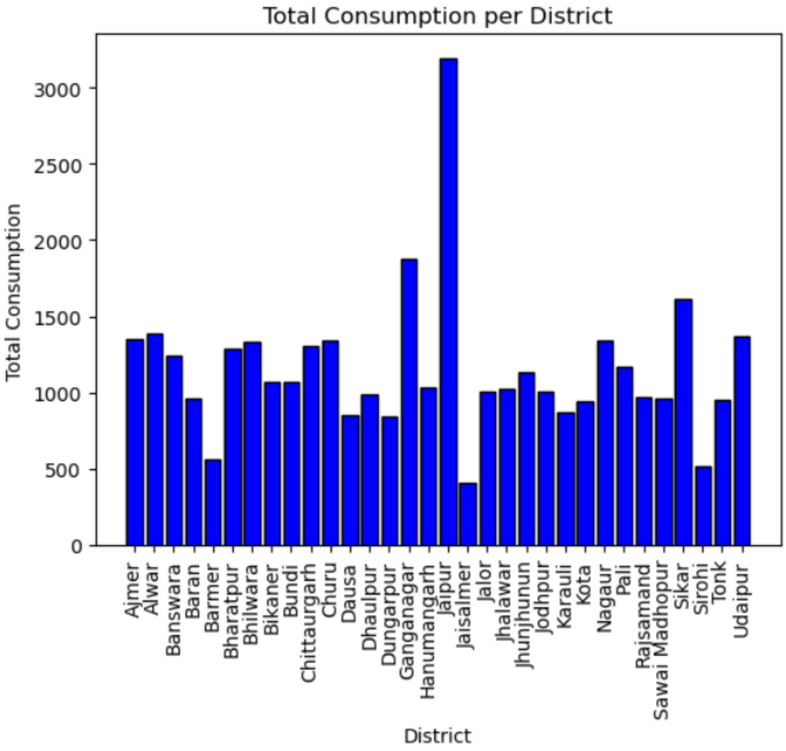
Most of the consumption values are clustered in the lower bins, indicating that the majority of districts have a lower total consumption.

There is a sharp decline in the number of districts as the consumption values increase, with very few districts having higher total consumption values.

This histogram helps visualize how total consumption is distributed across the districts of Rajasthan, highlighting that most districts fall into the lower consumption ranges.



```
[23]: plt.bar(RJ_consumption['District'], RJ_consumption['total_consumption'], color='blue', edgecolor='black')
      plt.xlabel("District")
      plt.ylabel("Total Consumption")
      plt.title("Total Consumption per District")
      plt.xticks(rotation=90)  # Rotate district names for better visibility
      plt.show()
```



Bar Plot Interpretation:

X-axis (District): The x-axis represents the names of the districts in Rajasthan.
Y-axis (Total Consumption): The y-axis shows the total consumption values for each district.

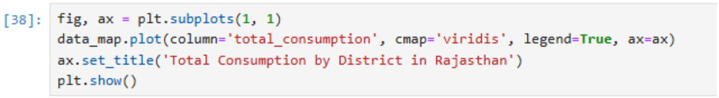
Each bar represents a district, with the height of the bar corresponding to the total consumption for that district.

The plot shows the distribution of total consumption across different districts.
There is significant variation in consumption across districts, with some districts having much higher consumption values than others.

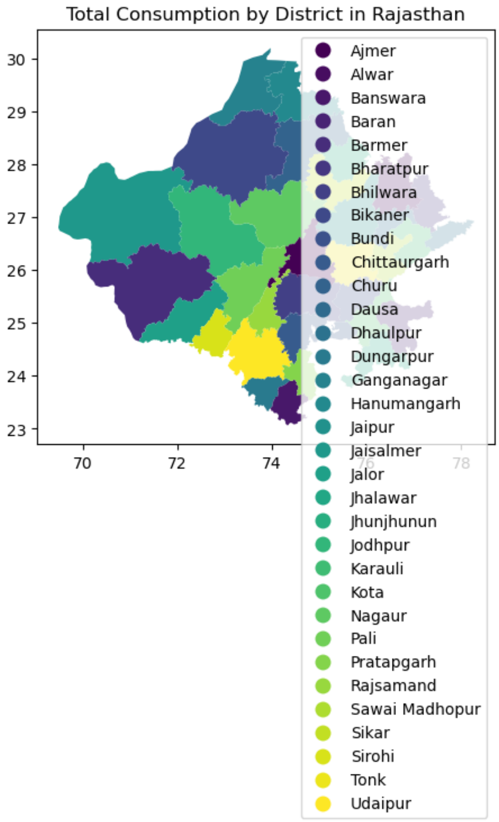
For example, districts like "Ganganagar" and "Jaipur" show higher total consumption compared to others like "Hanumangarh" and "Dhaulpur"

This bar plot helps visualize the total consumption per district in Rajasthan, allowing for easy comparison of consumption levels across different districts. It highlights which districts have higher or lower consumption, providing valuable insights for further analysis or strategic planning.

2.3. Rajasthan state map showing consumption in each district



```
[38]: fig, ax = plt.subplots(1, 1)
      data_map.plot(column='total_consumption', cmap='viridis', legend=True, ax=ax)
      ax.set_title('Total Consumption by District in Rajasthan')
      plt.show()
```



Interpretation:

It is a geographic visualization showing the total consumption by district in Rajasthan, India.

The map uses a colormap called 'viridis', where colors range from purple to yellow.
Dark purple represents lower consumption values, and bright yellow represents higher consumption values.

Intermediate colors (green, blue, etc.) represent varying levels of consumption between the lowest and highest values.

Each district in Rajasthan is color-coded based on its total consumption value. Districts like Jaipur, Ganganagar, and Barmer (shown in brighter colors) have higher total consumption compared to others.

Districts like Dhaulpur, Sirohi, and Banswara (shown in darker colors) have lower total consumption.

There is a noticeable variation in consumption across different districts.

The northern and central parts of Rajasthan seem to have higher consumption, whereas the southern and southeastern parts have relatively lower consumption.

The right side of the map provides a scale that translates colors into consumption values, allowing you to gauge the approximate consumption value for each district based on its color.

Districts with high consumption, such as Jaipur, might be targeted for more intensive market activities, resource allocation, and infrastructure development due to their higher demand.

Districts with lower consumption, like Dhaulpur and Sirohi, could be analyzed to understand the reasons behind the lower consumption and to explore potential opportunities for market expansion or resource support.

This map effectively visualizes the consumption patterns across Rajasthan, highlighting both high and low consumption districts. Such visualizations are invaluable for businesses, policymakers, and planners to make data-driven decisions regarding resource allocation, market strategies, and developmental planning.

3. Recommendations

3.1. Business Implications:

Consumption Disparity: There's a significant difference in consumption patterns across Rajasthan's districts, with districts like Jaipur and Ganganagar showing higher consumption, and districts like Dhaulpur and Sirohi showing lower consumption.

Geographical Influence: Consumption appears to correlate with geographical location, with higher consumption in the northern and central districts.

Potential Factors: Factors such as population density, economic development, and infrastructure might be influencing these consumption patterns.

3.2. Business Recommendations:

Market Segmentation: Businesses can identify high-potential markets (like Aizawl) and tailortheir products or services accordingly.

Market Segmentation: Businesses can identify high-potential markets, such as Jaipur and Ganganagar, and tailor their products or services to meet the specific demands of these areas.

Supply Chain Optimization: By understanding the consumption patterns, businesses can optimize their supply chain management to reduce costs and improve efficiency, ensuring that high-demand areas are well-stocked and serviced.

Infrastructure Development: Businesses can collaborate with government and other stakeholders to advocate for infrastructure development in low-consumption areas, such as Dhaulpur and Sirohi, to stimulate economic growth and increase consumption.

Product Adaptation: Businesses might need to adapt their product offerings or marketing strategies to cater to the different consumer preferences observed in various districts, ensuring that products are relevant and appealing to local markets.

These recommendations will help businesses to strategically target their efforts, optimize their operations, and potentially increase their market share in Rajasthan by leveraging the insights gained from the consumption data.

4. Codes

4.1. Python Jupyter Notebook codes

```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 1,
   "id": "f099039b-cae9-4194-8bea-fbbcb86b1efe",
   "metadata": {},
   "outputs": [
    {
   "source": [
    "import pandas as pd\n",
    "import numpy as np\n",
    "from scipy import stats\n",
    "import matplotlib.pyplot as plt\n",
    "import seaborn as sns\n",
    "import geopandas as gpd"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 2,
   "id": "743db998-ab57-4b3f-8207-5fbfee74337b",
   "metadata": {},
   "outputs": [
   ],
   "source": [
    "import pandas as pd\n",
    "import numpy as np\n",
    "from scipy import stats\n",
    "import matplotlib.pyplot as plt\n",
    "import seaborn as sns\n",
    "import geopandas as gpd"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 3,
   "id": "e4fedad6-50db-4443-80f1-2b13cc4cae72",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Installing collected packages: shapely, pyproj, pyogrio, geopandas\n",
      "Successfully installed geopandas-1.0.1 pyogrio-0.9.0 pyproj-3.6.1 shapely-2.0.5\n"
     ]
    }
   ],
   "source": [
    "# Install geopandas if not already installed\n",
    "!pip install geopandas"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 4,
   "id": "02f2d6b4-5203-48b1-a29b-87bd1bc5e4c1",
   "metadata": {},
   "outputs": [],
   "source": [
    "import pandas as pd\n",
    "import numpy as np\n",
    "from scipy import stats\n",
    "import matplotlib.pyplot as plt\n",
    "import seaborn as sns\n",
    "import geopandas as gpd"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 5,
   "id": "b775c0a1-3e6a-4665-8317-8f2e4894b1dd",
   "metadata": {},
   "outputs": [],
   "source": [
    "data = pd.read_csv(\"C:\\\\A5\\\\NSSO68.csv\", low_memory=False)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 6,
   "id": "b5a377ac-3032-48f5-aea8-de068c81ee62",
   "metadata": {},
   "outputs": [
{
 "data": {
  "text/html": [
   "<div>\n",
   "<style scoped>\n",
   "    .dataframe tbody tr th:only-of-type {\n",
   "        vertical-align: middle;\n",
   "    }\n",
   "\n",
   "    .dataframe tbody tr th {\n",
   "        vertical-align: top;\n",
   "    }\n",
   "\n",
   "    .dataframe thead th {\n",
   "        text-align: right;\n",
   "    }\n",
   "</style>\n",
   "<table border=\"1\" class=\"dataframe\">\n",
   "  <thead>\n",
   "    <tr style=\"text-align: right;\">\n",
   "      <th></th>\n",
   "      <th>slno</th>\n",
   "      <th>grp</th>\n",
   "      <th>Round_Centre</th>\n",
   "      <th>FSU_number</th>\n",
   "      <th>Round</th>\n",
   "      <th>Schedule_Number</th>\n",
   "      <th>Sample</th>\n",
   "      <th>Sector</th>\n",
   "      <th>state</th>\n",
   "      <th>State_Region</th>\n",
   "      <th>...</th>\n",
   "      <th>pickle_v</th>\n",
   "      <th>sauce_jam_v</th>\n",
   "      <th>Othrprocessed_v</th>\n",
   "      <th>Beveragestotal_v</th>\n",
   "      <th>foodtotal_v</th>\n",
   "      <th>foodtotal_q</th>\n",
   "      <th>state_1</th>\n",
   "      <th>Region</th>\n",
   "      <th>fruits_df_tt_v</th>\n",
   "      <th>fv_tot</th>\n",
   "    </tr>\n",
   "  </thead>\n",
   "  <tbody>\n",
   "    <tr>\n",
   "      <th>0</th>\n",
   "      <td>1</td>\n",
   "      <td>4.10E+31</td>\n",
   "      <td>1</td>\n",
   "      <td>41000</td>\n",
   "      <td>68</td>\n",
"      <td>10</td>\n",
"      <td>1</td>\n",
"      <td>2</td>\n",
"      <td>24</td>\n",
"      <td>242</td>\n",
"      <td>...</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.000000</td>\n",
"      <td>1141.492400</td>\n",
"      <td>30.942394</td>\n",
"      <td>GUJ</td>\n",
"      <td>2</td>\n",
"      <td>12.000000</td>\n",
"      <td>154.180000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>2</td>\n",
"      <td>4.10E+31</td>\n",
"      <td>1</td>\n",
"      <td>41000</td>\n",
"      <td>68</td>\n",
"      <td>10</td>\n",
"      <td>1</td>\n",
"      <td>2</td>\n",
"      <td>24</td>\n",
"      <td>242</td>\n",
"      <td>...</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>17.500000</td>\n",
"      <td>1244.553500</td>\n",
"      <td>29.286153</td>\n",
"      <td>GUJ</td>\n",
"      <td>2</td>\n",
"      <td>333.000000</td>\n",
"      <td>484.950000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>3</td>\n",
"      <td>4.10E+31</td>\n",
"      <td>1</td>\n",
"      <td>41000</td>\n",
"      <td>68</td>\n",
"      <td>10</td>\n",
"      <td>1</td>\n",
"      <td>2</td>\n",
"      <td>24</td>\n",
"      <td>242</td>\n",
"      <td>...</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.000000</td>\n",
"      <td>1050.315400</td>\n",
"      <td>31.527046</td>\n",
"      <td>GUJ</td>\n",
"      <td>2</td>\n",
"      <td>35.000000</td>\n",
"      <td>214.840000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>4</td>\n",
"      <td>4.10E+31</td>\n",
"      <td>1</td>\n",
"      <td>41000</td>\n",
"      <td>68</td>\n",
"      <td>10</td>\n",
"      <td>1</td>\n",
"      <td>2</td>\n",
"      <td>24</td>\n",
"      <td>242</td>\n",
"      <td>...</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>33.333333</td>\n",
"      <td>1142.591667</td>\n",
"      <td>27.834607</td>\n",
"      <td>GUJ</td>\n",
"      <td>2</td>\n",
"      <td>168.333333</td>\n",
"      <td>302.300000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>5</td>\n",
"      <td>4.10E+31</td>\n",
"      <td>1</td>\n",
"      <td>41000</td>\n",
"      <td>68</td>\n",
"      <td>10</td>\n",
"      <td>1</td>\n",
"      <td>2</td>\n",
"      <td>24</td>\n",
"      <td>242</td>\n",
"      <td>...</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>75.000000</td>\n",
"      <td>945.249500</td>\n",
"      <td>27.600713</td>\n",
"      <td>GUJ</td>\n",
"      <td>2</td>\n",
"      <td>15.000000</td>\n",
"      <td>148.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>...</th>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>101657</th>\n",
"      <td>101658</td>\n",
"      <td>8.00E+31</td>\n",
"      <td>1</td>\n",
"      <td>79998</td>\n",
"      <td>68</td>\n",
"      <td>10</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>12</td>\n",
"      <td>...</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.000000</td>\n",
"      <td>544.013667</td>\n",
"      <td>28.441750</td>\n",
"      <td>J$K</td>\n",
"      <td>2</td>\n",
"      <td>0.000000</td>\n",
"      <td>25.833333</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>101658</th>\n",
"      <td>101659</td>\n",
"      <td>8.00E+31</td>\n",
"      <td>1</td>\n",
"      <td>79998</td>\n",
"      <td>68</td>\n",
"      <td>10</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>12</td>\n",
"      <td>...</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>8.000000</td>\n",
"      <td>417.616600</td>\n",
"      <td>25.490282</td>\n",
"      <td>J$K</td>\n",
"      <td>2</td>\n",
"      <td>0.000000</td>\n",
"      <td>49.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>101659</th>\n",
"      <td>101660</td>\n",
"      <td>8.00E+31</td>\n",
"      <td>1</td>\n",
"      <td>79998</td>\n",
"      <td>68</td>\n",
"      <td>10</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>12</td>\n",
"      <td>...</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>7.142857</td>\n",
"      <td>378.300429</td>\n",
"      <td>25.800107</td>\n",
"      <td>J$K</td>\n",
"      <td>2</td>\n",
"      <td>0.000000</td>\n",
"      <td>32.285714</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>101660</th>\n",
"      <td>101661</td>\n",
"      <td>8.00E+31</td>\n",
"      <td>1</td>\n",
"      <td>79998</td>\n",
"      <td>68</td>\n",
"      <td>10</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>12</td>\n",
"      <td>...</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>14.000000</td>\n",
"      <td>510.023600</td>\n",
"      <td>30.220170</td>\n",
"      <td>J$K</td>\n",
"      <td>2</td>\n",
"      <td>0.000000</td>\n",
"      <td>39.200000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>101661</th>\n",
"      <td>101662</td>\n",
"      <td>8.00E+31</td>\n",
"      <td>1</td>\n",
"      <td>79998</td>\n",
"      <td>68</td>\n",
"      <td>10</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>12</td>\n",
"      <td>...</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>8.571429</td>\n",
"      <td>424.589714</td>\n",
"      <td>26.157279</td>\n",
"      <td>J$K</td>\n",
"      <td>2</td>\n",
"      <td>0.000000</td>\n",
"      <td>39.714286</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>101662 rows × 384 columns</p>\n",
      "</div>"
     ],
     "text/plain": [
      "        slno      grp  Round_Centre  FSU_number  Round
Schedule_Number  \\\n",
      "0          1  4.10E+31            1       41000     68            10   \n",
      "1          2  4.10E+31            1       41000     68            10   \n",
      "2          3  4.10E+31            1       41000     68            10   \n",
      "3          4  4.10E+31            1       41000     68            10   \n",
      "4          5  4.10E+31            1       41000     68            10   \n",
      "...      ...       ...          ...        ...    ...            ...   \n",
      "101657  101658  8.00E+31            1       79998     68            10   \n",
      "101658  101659  8.00E+31            1       79998     68            10   \n",
      "101659  101660  8.00E+31            1       79998     68            10   \n",
      "101660  101661  8.00E+31            1       79998     68            10   \n",
      "101661  101662  8.00E+31            1       79998     68            10   \n",
      "\n",
      "       Sample  Sector  state  State_Region  ...  pickle_v  sauce_jam_v  \\\n",
      "0           1       2     24           242  ...       0.0          0.0   \n",
      "1           1       2     24           242  ...       0.0          0.0   \n",
      "2           1       2     24           242  ...       0.0          0.0   \n",
      "3           1       2     24           242  ...       0.0          0.0   \n",
      "4           1       2     24           242  ...       0.0          0.0   \n",
      "...       ...     ...    ...           ...  ...       ...          ...   \n",
      "101657      1       1      1            12  ...       0.0          0.0   \n",
      "101658      1       1      1            12  ...       0.0          0.0   \n",
      "101659      1       1      1            12  ...       0.0          0.0   \n",
      "101660      1       1      1            12  ...       0.0          0.0   \n",
      "101661      1       1      1            12  ...       0.0          0.0   \n",
      "\n",
      "       Othrprocessed_v  Beveragestotal_v  foodtotal_v  foodtotal_q  state_1
\\\n",
      "0                  0.0          0.000000  1141.492400    30.942394      GUJ   \n",
      "1                  0.0         17.500000  1244.553500    29.286153      GUJ   \n",
      "2                  0.0          0.000000  1050.315400    31.527046      GUJ   \n",
      "3                  0.0         33.333333  1142.591667    27.834607      GUJ   \n",
      "4                  0.0         75.000000   945.249500    27.600713      GUJ   \n",
      "...                ...               ...          ...          ...      ...   \n",
      "101657             0.0          0.000000   544.013667    28.441750      J$K
\n",
      "101658             0.0          8.000000   417.616600    25.490282      J$K
\n",
      "101659             0.0          7.142857   378.300429    25.800107      J$K
\n",
      "101660             0.0         14.000000   510.023600    30.220170      J$K
\n",
      "101661             0.0          8.571429   424.589714    26.157279      J$K
\n",
      "\n",
      "       Region  fruits_df_tt_v      fv_tot  \n",
      "0           2       12.000000  154.180000  \n",
      "1           2      333.000000  484.950000  \n",
      "2           2       35.000000  214.840000  \n",
      "3           2      168.333333  302.300000  \n",
      "4           2       15.000000  148.000000  \n",
      "...       ...             ...         ...  \n",
      "101657      2        0.000000   25.833333  \n",
      "101658      2        0.000000   49.000000  \n",
      "101659      2        0.000000   32.285714  \n",
      "101660      2        0.000000   39.200000  \n",
      "101661      2        0.000000   39.714286  \n",
      "\n",
      "[101662 rows x 384 columns]"
     ]
    },
    "metadata": {},
    "output_type": "display_data"
   }
  ],
  "source": [
   "display(data)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 7,
  "id": "75367525-d02b-4a07-a5af-e8fdec46cec5",
  "metadata": {},
  "outputs": [],
  "source": [
   "Rajasthan_data = data[data['state_1'] == 'RJ']"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 8,
  "id": "feee6bfa-17d5-4353-9cde-6eb1240aade6",
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "Missing values in each column:\n",
     "slno              0\n",
     "grp               0\n",
     "Round_Centre      0\n",
     "FSU_number       0\n",
     "Round             0\n",
     "                 ..\n",
     "foodtotal_q       0\n",
     "state_1           0\n",
     "Region            0\n",
     "fruits_df_tt_v    0\n",
     "fv_tot            0\n",
     "Length: 384, dtype: int64\n"
    ]
   }
  ],
  "source": [
   "missing_values = Rajasthan_data.isna().sum()\n",
   "print(\"Missing values in each column:\")\n",
   "print(missing_values)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 9,
  "id": "a2ecb5c9-9b38-4b1b-9007-1c8de7ea0ae5",
  "metadata": {},
  "outputs": [],
  "source": [
   "RJ = Rajasthan_data[['state_1', 'District', 'Region', 'Sector', 'State_Region',
'Meals_At_Home', 'ricepds_v', 'Wheatpds_q', 'chicken_q', 'pulsep_q',
'wheatos_q', 'No_of_Meals_per_day']]"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 10,
  "id": "56660fad-496c-4f00-a6c6-cab405e8e82b",
  "metadata": {},
  "outputs": [],
  "source": [
   "def impute_with_mean(column):\n",
   "    if column.hasnans:\n",
   "        column.fillna(column.mean(), inplace=True)\n",
   "    return column"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 11,
  "id": "5f129423-2317-4e33-854e-1526dfe6db21",
  "metadata": {},
  "outputs": [
   {
    "name": "stderr",
    "output_type": "stream",
    "text": [
     "C:\\Users\\Prakash
R\\AppData\\Local\\Temp\\ipykernel_1608\\2826044316.py:3:
SettingWithCopyWarning: \n",
     "A value is trying to be set on a copy of a slice from a DataFrame\n",
     "\n",
     "See the caveats in the documentation: https://pandas.pydata.org/pandas-
```

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy\n",
      "  column.fillna(column.mean(), inplace=True)\n",
      "C:\\Users\\Prakash R\\AppData\\Local\\Temp\\ipykernel_1608\\3187162220.py:1: SettingWithCopyWarning: \n",
      "A value is trying to be set on a copy of a slice from a DataFrame.\n",
      "Try using .loc[row_indexer,col_indexer] = value instead\n",
      "\n",
      "See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy\n",
      "  RJ['Meals_At_Home'] = impute_with_mean(RJ['Meals_At_Home'])\n"
     ]
    }
   ],
   "source": [
    "RJ['Meals_At_Home'] = impute_with_mean(RJ['Meals_At_Home'])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 12,
   "id": "d0787cdc-a33f-4088-8f65-0ea96e15d59f",
   "metadata": {},
   "outputs": [],
   "source": [
    "def remove_outliers(df, column_name):\n",
    "    Q1 = df[column_name].quantile(0.25)\n",
    "    Q3 = df[column_name].quantile(0.75)\n",
    "    IQR = Q3 - Q1\n",
    "    lower_threshold = Q1 - (1.5 * IQR)\n",
    "    upper_threshold = Q3 + (1.5 * IQR)\n",
    "    df = df[(df[column_name] >= lower_threshold) & (df[column_name] <= upper_threshold)]\n",
    "    return df\n",
    "\n",
    "outlier_columns = ['ricepds_v', 'chicken_q']\n",
    "for col in outlier_columns:\n",
    "    RJ = remove_outliers(RJ, col)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 13,
   "id": "65017c07-8aed-46d0-b0bc-7a13e3355eff",
   "metadata": {},
   "outputs": [],
   "source": [
    "RJ['total_consumption'] = RJ[['ricepds_v', 'Wheatpds_q', 'chicken_q', 'pulsep_q', 'wheatos_q']].sum(axis=1)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 14,
   "id": "2d56a6cf-258c-4f54-a987-a4bab5d2e85f",
   "metadata": {},
   "outputs": [],
   "source": [
    "def summarize_consumption(group_col):\n",
    "    summary = RJ.groupby(group_col)['total_consumption'].sum().reset_index()\n",
    "    summary.sort_values(by='total_consumption', ascending=False, inplace=True)\n",
    "    return summary"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 15,
   "id": "7750242a-7535-43f8-9035-4cde55cb1111",
   "metadata": {},
   "outputs": [],
   "source": [
    "district_summary = summarize_consumption('District')\n",
    "region_summary = summarize_consumption('Region')"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 16,
   "id": "3c124ad1-e007-4780-927d-6934de9fe5e0",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Top Consuming Districts:\n",
      "    District  total_consumption\n",
      "11        12        3192.679460\n",
      "0          1        1875.184343\n",
      "12        13        1618.060832\n",
      "5          6        1387.333899\n",
      "Region Consumption Summary:\n",
      "   Region  total_consumption\n",
      "1       2       13170.497313\n",
      "4       5        8345.315881\n",
      "0       1        5741.479493\n",
      "3       4        5295.284359\n",
      "2       3        4419.917532\n"
     ]
    }
   ],
   "source": [
   "print(\"Top Consuming Districts:\")\n",
   "print(district_summary.head(4))\n",
   "print(\"Region Consumption Summary:\")\n",
   "print(region_summary)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 17,
  "id": "46f9413e-61dd-4da1-acca-f74e747406b8",
  "metadata": {},
  "outputs": [],
  "source": [
   "district = {'1': 'Ganganagar',\n",
   "    '2': 'Hanumangarh',\n",
   "    '3': 'Bikaner',\n",
   "    '4': 'Churu',\n",
   "    '5': 'Jhunjhunun',\n",
   "    '6': 'Alwar',\n",
   "    '7': 'Bharatpur',\n",
   "    '8': 'Dhaulpur',\n",
   "    '9': 'Karauli',\n",
   "    '10': 'Sawai Madhopur',\n",
   "    '11': 'Dausa',\n",
   "    '12': 'Jaipur',\n",
   "    '13': 'Sikar',\n",
   "    '14': 'Nagaur',\n",
   "    '15': 'Jodhpur',       \n",
   "    '16': 'Jaisalmer',\n",
   "    '17': 'Barmer',\n",
   "    '18': 'Jalor',\n",
   "    '19': 'Sirohi',\n",
   "    '20': 'Pali',\n",
   "    '21': 'Ajmer',\n",
   "    '22': 'Tonk',       \n",
   "    '23': 'Bundi',\n",
   "    '24': 'Bhilwara',\n",
   "    '25': 'Rajsamand',\n",
   "    '26': 'Udaipur',\n",
   "    '27': 'Dungarpur',\n",
   "    '28': 'Banswara',\n",
   "    '29': 'Chittaurgarh', \n",
   "    '30': 'Kota',\n",
   "    '31': 'Baran',\n",
   "    '32': 'Jhalawar',       \n",
   "}\n",
   "\n",
   "sector = {\n",
   "    '2': 'URBAN',\n",
   "    '1': 'RURAL'\n",
   "}"
  ]
  },
  {
   "cell_type": "code",
   "execution_count": 18,
   "id": "7161cbed-1c79-4c5e-95ed-e4baffcc4193",
   "metadata": {},
   "outputs": [],
   "source": [
    "RJ['District'] = RJ['District'].astype(str)\n",
    "RJ['Sector'] = RJ['Sector'].astype(str)\n",
    "\n",
    "RJ['District'] = RJ['District'].map(district).fillna(RJ['District'])\n",
    "RJ['Sector'] = RJ['Sector'].map(sector).fillna(RJ['Sector'])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 19,
   "id": "9ff590c5-ed39-4e22-ac96-6f0b1a76f54d",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "      state_1 District  Region Sector  State_Region  Meals_At_Home  
ricepds_v  \\\n",
      "32036      RJ   Jaipur      2  URBAN           82          54.0        0.0   \n",
      "32037      RJ   Jaipur      2  URBAN           82          59.0        0.0   \n",
      "32040      RJ   Jaipur      2  URBAN           82          60.0        0.0   \n",
      "32043      RJ   Jaipur      2  URBAN           82          60.0        0.0   \n",
      "32044      RJ   Jaipur      2  URBAN           82          52.0        0.0   \n",
      "\n",
      "       Wheatpds_q  chicken_q  pulsep_q  wheatos_q  No_of_Meals_per_day  
\\\n",
      "32036    0.000000        0.0  0.000000   6.666667                  2.0   \n",
      "32037    0.000000        0.0  0.285714   7.142857                  2.0   \n",
      "32040    0.000000        0.0  0.214286   5.000000                  2.0   \n",
      "32043    3.333333        0.0  0.000000   3.333333                  2.0   \n",
      "32044    0.000000        0.0  0.000000  10.000000                  2.0   \n",
      "\n",
      "       total_consumption  \n",
      "32036           6.666667  \n",
      "32037           7.428571  \n",
      "32040           5.214286  \n",
      "32043           6.666667  \n",
      "32044          10.000000  \n"
     ]
    }
   ],
   "source": [
    "print(RJ.head())"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 20,
   "id": "bbf20849-b68e-409d-b7a6-b821c00687e0",
   "metadata": {},
   "outputs": [
    {
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "plt.hist(RJ['total_consumption'], bins=10, color='blue',
edgecolor='black')\n",
    "plt.xlabel(\"Consumption\")\n",
    "plt.ylabel(\"Frequency\")\n",
    "plt.title(\"Consumption Distribution in Rajasthan State\")\n",
    "plt.show()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 21,
   "id": "bf4aef8b-c8e7-4633-8cb7-14470dab842f",
   "metadata": {},
   "outputs": [],
   "source": [
    "RJ_consumption =
RJ.groupby('District')['total_consumption'].sum().reset_index()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 22,
   "id": "be588230-fe8a-4aa5-9afb-aa2676075868",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "   District  total_consumption\n",
      "0     Ajmer        1352.323214\n",
      "1     Alwar        1387.333899\n",
      "2  Banswara        1244.138167\n",
      "3     Baran         956.822823\n",
      "4    Barmer         563.346825\n"
     ]
    }
   ],
   "source": [
    "print(RJ_consumption.head())"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 23,
   "id": "bbac8e17-5d63-485e-bc4f-832d7c2a9f47",
   "metadata": {},
   "outputs": [
    {
     "data": {
         "text/plain": [
      "<Figure size 640x480 with 1 Axes>"
      ]
     },
     "metadata": {},
     "output_type": "display_data"
    }
   ],
   "source": [
    "plt.bar(RJ_consumption['District'], RJ_consumption['total_consumption'],
color='blue', edgecolor='black')\n",
    "plt.xlabel(\"District\")\n",
    "plt.ylabel(\"Total Consumption\")\n",
    "plt.title(\"Total Consumption per District\")\n",
    "plt.xticks(rotation=90)  # Rotate district names for better visibility\n",
    "plt.show()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 24,
   "id": "adab44c9-b7cc-4713-be2e-d6690f8fbe52",
   "metadata": {},
   "outputs": [],
   "source": [
    "data_map =
gpd.read_file(\"C:\\\\A5\\\\RAJASTHAN_DISTRICTS.geojson\")"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 25,
   "id": "34213c91-87cd-443f-9c13-3fcad2a1cd30",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
      "text": [
       "Index(['dtname', 'stname', 'stcode11', 'dtcode11', 'year_stat',
'Shape_Length',\n",
       "       'Shape_Area', 'OBJECTID', 'test', 'Dist_LGD', 'State_LGD',
'geometry'],\n",
       "      dtype='object')\n",
       "Index(['District', 'total_consumption'], dtype='object')\n"
      ]
     }
    ],
    "source": [
     "print(data_map.columns)\n",
     "print(RJ_consumption.columns)"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 26,
    "id": "5e9c3aba-8aa6-4d34-b157-f00b009d0836",
    "metadata": {},
    "outputs": [],
    "source": [
     "data_map['District'] = RJ_consumption['District']"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 27,
    "id": "4924fd6c-1bc0-49ce-af37-c0bb5e9c074c",
    "metadata": {},
    "outputs": [],
    "source": [
     "data_map_data = data_map.merge(RJ_consumption, left_on='dtname',
right_on='District')"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 28,
    "id": "6302ac21-28cf-4ccc-a415-0f72f8a46ad2",
    "metadata": {},
    "outputs": [
     {
      "name": "stdout",
      "output_type": "stream",
      "text": [
       "Index(['dtname', 'stname', 'stcode11', 'dtcode11', 'year_stat',
'Shape_Length',\n",
       "       'Shape_Area', 'OBJECTID', 'test', 'Dist_LGD', 'State_LGD',
'geometry',\n",
       "       'District'],\n",
       "      dtype='object')\n"
     ]
    }
   ],
   "source": [
    "print(data_map.columns)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 29,
   "id": "622c97b8-0be9-4044-a0ba-1d18e11f5d03",
   "metadata": {},
   "outputs": [],
   "source": [
    "import geopandas as gpd\n",
    "import pandas as pd\n",
    "import matplotlib.pyplot as plt"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 30,
   "id": "f6d1de2c-f494-4fab-bb84-958780550ae8",
   "metadata": {},
   "outputs": [],
   "source": [
    "data_map =
gpd.read_file(\"C:\\\\A5\\\\RAJASTHAN_DISTRICTS.geojson\")"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 31,
   "id": "266c19fa-68ea-4026-9ac9-45db76afc7f6",
   "metadata": {},
   "outputs": [],
   "source": [
    "data_map = data_map.rename(columns={'dtname': 'District'})"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 32,
   "id": "77ce79ca-a900-4310-b283-0cdb2d788cf7",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "<bound method DataFrame.rename of          District     stname stcode11
dtcode11 year_stat  Shape_Length  \\\n",
       "0          Churu  RAJASTHAN       08      102    2011_c  1.186656e+06
```

```
\n",
    "1       Jhunjhunun  RAJASTHAN       08      103    2011_c  5.925478e+05
\n",
    "2        Jaisalmer  RAJASTHAN       08      114    2011_c  1.320650e+06
\n",
    "3            Sikar  RAJASTHAN       08      111    2011_c  8.314277e+05
\n",
    "4            Alwar  RAJASTHAN       08      104    2011_c  1.025893e+06
\n",
    "5           Jaipur  RAJASTHAN       08      110    2011_c  9.696342e+05
\n",
    "6          Jodhpur  RAJASTHAN       08      113    2011_c  1.185635e+06
\n",
    "7        Bharatpur  RAJASTHAN       08      105    2011_c  8.366005e+05
\n",
    "8           Nagaur  RAJASTHAN       08      112    2011_c  1.097482e+06
\n",
    "9            Dausa  RAJASTHAN       08      109    2011_c  5.884233e+05
\n",
    "10         Karauli  RAJASTHAN       08      107    2011_c  5.974678e+05
\n",
    "11        Dhaulpur  RAJASTHAN       08      106    2011_c  4.547446e+05
\n",
    "12          Barmer  RAJASTHAN       08      115    2011_c  1.191299e+06
\n",
    "13  Sawai Madhopur  RAJASTHAN       08      108    2011_c
6.534446e+05   \n",
    "14            Tonk  RAJASTHAN       08      120    2011_c  7.835835e+05
\n",
    "15            Pali  RAJASTHAN       08      118    2011_c  9.892842e+05
\n",
    "16        Bhilwara  RAJASTHAN       08      122    2011_c  8.258136e+05
\n",
    "17           Jalor  RAJASTHAN       08      116    2011_c  8.271411e+05
\n",
    "18           Bundi  RAJASTHAN       08      121    2011_c  5.495688e+05
\n",
    "19            Kota  RAJASTHAN       08      127    2011_c  7.201744e+05
\n",
    "20          Sirohi  RAJASTHAN       08      117    2011_c  4.995064e+05
\n",
    "21           Baran  RAJASTHAN       08      128    2011_c  7.104107e+05
\n",
    "22         Udaipur  RAJASTHAN       08      130    2011_c  9.779398e+05
\n",
    "23        Jhalawar  RAJASTHAN       08      129    2011_c  8.430501e+05
\n",
    "24       Dungarpur  RAJASTHAN       08      124    2011_c  4.265645e+05
\n",
    "25        Banswara  RAJASTHAN       08      125    2011_c  4.851035e+05
\n",
    "26           Ajmer  RAJASTHAN       08      119    2011_c  9.418691e+05
\n",
    "27      Rajsamand  RAJASTHAN      08     123    2011_c  7.904572e+05
\n",
    "28   Chittaurgarh  RAJASTHAN      08     126    2011_c  1.096767e+06
\n",
    "29     Ganganagar  RAJASTHAN      08     099    2011_c  9.753333e+05
\n",
    "30    Hanumangarh  RAJASTHAN      08     100    2011_c
1.091322e+06   \n",
    "31        Bikaner  RAJASTHAN      08     101    2011_c  1.388719e+06
\n",
    "32     Pratapgarh  RAJASTHAN      08     131    2011_c  5.130811e+05
\n",
    "\n",
    "     Shape_Area  OBJECTID  test  Dist_LGD  State_LGD  \\\n",
    "0   1.798435e+10       197     0        96          8  \n",
    "1   7.630806e+09       214     0       106          8  \n",
    "2   4.867759e+10       222     1       103          8  \n",
    "3   9.868631e+09       226     0       114          8  \n",
    "4   1.073342e+10       229     0        87          8  \n",
    "5   1.408238e+10       238     0       102          8  \n",
    "6   2.855696e+10       240     0       107          8  \n",
    "7   6.430485e+09       242     0        91          8  \n",
    "8   2.253599e+10       244     0       110          8  \n",
    "9   4.310939e+09       267     0        97          8  \n",
    "10  6.296818e+09       281     1       108          8  \n",
    "11  3.813713e+09       286     1        98          8  \n",
    "12  3.513685e+10       292     1        90          8  \n",
    "13  6.264683e+09       299     0       113          8  \n",
    "14  8.976419e+09       303     0       116          8  \n",
    "15  1.531742e+10       306     0       111          8  \n",
    "16  1.291780e+10       325     0        92          8  \n",
    "17  1.304395e+10       331     0       104          8  \n",
    "18  7.090133e+09       332     0        94          8  \n",
    "19  6.268233e+09       337     0       109          8  \n",
    "20  6.249981e+09       353     1       115          8  \n",
    "21  8.545512e+09       357     0        89          8  \n",
    "22  1.418545e+10       363     0       117          8  \n",
    "23  7.671897e+09       383     0       105          8  \n",
    "24  4.544680e+09       415     0        99          8  \n",
    "25  5.354947e+09       418     0        88          8  \n",
    "26  1.059786e+10       574     0        86          8  \n",
    "27  5.692261e+09       578     0       112          8  \n",
    "28  9.506564e+09       581     0        95          8  \n",
    "29  1.442324e+10       656     0       100          8  \n",
    "30  1.279695e+10       664     0       101          8  \n",
    "31  3.909304e+10       682     0        93          8  \n",
    "32  5.324407e+09       705     0       629          8  \n",
    "\n",
    "                                             geometry  \n",
    "0   POLYGON ((75.4274 28.99982, 75.42681 28.99986,...  \n",
    "1   POLYGON ((75.66998 28.51904, 75.66998 28.51904...  \n",
    "2   POLYGON ((70.50679 28.03657, 70.50542 28.03695...  \n",
     "3   POLYGON ((75.02761 28.20333, 75.02732 28.20413...  \n",
     "4   POLYGON ((76.85008 28.22136, 76.84847 28.22165...  \n",
     "5   POLYGON ((76.08882 27.86035, 76.08785 27.86065...  \n",
     "6   POLYGON ((72.04898 27.61588, 72.04566 27.61866...  \n",
     "7   POLYGON ((77.04525 27.8214, 77.04479 27.82183,...  \n",
     "8   POLYGON ((74.40347 27.69947, 74.40287 27.7006,...  \n",
     "9   POLYGON ((76.89436 27.23175, 76.89317 27.23225...  \n",
     "10  POLYGON ((76.8253 26.99983, 76.8244 26.99984, ...  \n",
     "11  POLYGON ((78.14448 26.94989, 78.14383 26.94992...  \n",
     "12  POLYGON ((71.45226 26.51415, 71.4489 26.51562,...  \n",
     "13  POLYGON ((76.47407 26.71738, 76.47267 26.71784...  \n",
     "14  POLYGON ((75.28566 26.56463, 75.28464 26.56469...  \n",
     "15  POLYGON ((74.12169 26.45687, 74.1214 26.45746,...  \n",
     "16  POLYGON ((74.38934 25.96141, 74.388 25.96294, ...  \n",
     "17  POLYGON ((72.758 25.80951, 72.75748 25.80993, ...  \n",
     "18  POLYGON ((75.86385 25.87563, 75.8621 25.87579,...  \n",
     "19  POLYGON ((76.56373 25.84869, 76.56331 25.84871...  \n",
     "20  POLYGON ((72.83078 25.28407, 72.82868 25.28511...  \n",
     "21  POLYGON ((76.49661 25.43027, 76.49551 25.43063...  \n",
     "22  POLYGON ((73.48739 25.10448, 73.48677 25.10475...  \n",
     "23  POLYGON ((76.35883 24.87065, 76.35883 24.87065...  \n",
     "24  POLYGON ((74.15153 24.01146, 74.15019 24.01143...  \n",
     "25  POLYGON ((74.48009 23.92511, 74.47965 23.92511...  \n",
     "26  MULTIPOLYGON (((74.08925 25.84816, 74.09164 25...  \n",
     "27  MULTIPOLYGON (((74.12093 25.86304, 74.11974 25...  \n",
     "28  MULTIPOLYGON (((75.40832 25.02031, 75.4082 25....  \n",
     "29  POLYGON ((73.97265 30.19795, 73.97259 30.19813...  \n",
     "30  POLYGON ((74.29826 29.95529, 74.29679 29.95534...  \n",
     "31  POLYGON ((73.77098 29.04956, 73.77087 29.04955...  \n",
     "32  POLYGON ((74.59663 24.51195, 74.59376 24.51258...  >"
    ]
   },
   "metadata": {},
   "output_type": "display_data"
  }
 ],
 "source": [
  "display(data_map.rename)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 33,
 "id": "99eb8604-7523-4c63-badf-fe55d389241b",
 "metadata": {},
 "outputs": [],
 "source": []
},
{
 "cell_type": "code",
 "execution_count": 34,
 "id": "3b42390c-15b0-4e72-a3fe-a36640a6849a",
   "metadata": {},
   "outputs": [],
   "source": [
    "RJ_consumption = pd.read_csv(\"C:\\\\A5\\\\NSSO68.csv\",
low_memory=False)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 35,
   "id": "45ebd945-cb3c-44ff-b885-de56fd9cc91e",
   "metadata": {},
   "outputs": [],
   "source": [
    "RJ_consumption =
RJ.groupby('District')['total_consumption'].sum().reset_index()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 36,
   "id": "da532a6a-99bd-4360-97e7-09a957b155ef",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "   District  total_consumption\n",
      "0     Ajmer        1352.323214\n",
      "1     Alwar        1387.333899\n",
      "2  Banswara        1244.138167\n",
      "3     Baran         956.822823\n",
      "4    Barmer         563.346825\n"
     ]
    }
   ],
   "source": [
    "print(RJ_consumption.head())"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 37,
   "id": "863a6410-a2e7-4d9b-aac6-d665b43b6e7a",
   "metadata": {},
   "outputs": [],
   "source": [
    "data_map =
gpd.read_file(\"C:\\\\A5\\\\RAJASTHAN_DISTRICTS.geojson\")\n",
    "data_map = data_map.rename(columns={'dtname': 'total_consumption'})"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "331b4fd3-b0d0-44fa-9771-d43445e68274",
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3 (ipykernel)",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.11.7"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```

4.2. R codes

```
# Set the working directory and verify it
setwd("C:\\A5")
getwd()
install.packages("sf")

#install.packages(dplyr)
# Function to install and load libraries
install_and_load <- function(package) {
  if (!require(package, character.only = TRUE)) {
    install.packages(package, dependencies = TRUE)
    library(package, character.only = TRUE)
  }
}

# Load required libraries
libraries <- c("dplyr", "readr", "readxl", "tidyr", "ggplot2", "BSDA")
lapply(libraries, install_and_load)
```

```
# Reading the file into R
data <- read.csv("C:\\A5\\NSSO68.csv")

# Filtering for RJ
df <- data %>%
  filter(state_1 == "RJ")

# Display dataset info
cat("Dataset Information:\n")
print(names(df))
print(head(df))
print(dim(df))

# Finding missing values
missing_info <- colSums(is.na(df))
cat("Missing Values Information:\n")
print(missing_info)

# Subsetting the data
RJnew <- df %>%
  select(state_1, District, Region, Sector, State_Region, Meals_At_Home, ricepds_v,
Wheatpds_q, chicken_q, pulsep_q, wheatos_q, No_of_Meals_per_day)

# Impute missing values with mean for specific columns
impute_with_mean <- function(column) {
  if (any(is.na(column))) {
    column[is.na(column)] <- mean(column, na.rm = TRUE)
  }
  return(column)
}
RJnew$Meals_At_Home <- impute_with_mean(RJnew$Meals_At_Home)

# Finding outliers and removing them
remove_outliers <- function(df, column_name) {
  Q1 <- quantile(df[[column_name]], 0.25)
  Q3 <- quantile(df[[column_name]], 0.75)
  IQR <- Q3 - Q1
  lower_threshold <- Q1 - (1.5 * IQR)
  upper_threshold <- Q3 + (1.5 * IQR)
  df <- subset(df, df[[column_name]] >= lower_threshold & df[[column_name]] <=
upper_threshold)
  return(df)
}
outlier_columns <- c("ricepds_v", "chicken_q")
for (col in outlier_columns) {
  RJnew <- remove_outliers(RJnew, col)
}

# Summarize consumption
RJnew$total_consumption <- rowSums(RJnew[, c("ricepds_v", "Wheatpds_q",
"chicken_q", "pulsep_q", "wheatos_q")], na.rm = TRUE)

# Summarize and display top consuming districts and regions
summarize_consumption <- function(group_col) {
  summary <- RJnew %>%
    group_by(across(all_of(group_col))) %>%
    summarise(total = sum(total_consumption)) %>%
    arrange(desc(total))
  return(summary)
}
district_summary <- summarize_consumption("District")
region_summary <- summarize_consumption("Region")

cat("Top Consuming Districts:\n")
print(head(district_summary, 4))
cat("Region Consumption Summary:\n")
print(region_summary)

# Rename districts and sectors
district_mapping <- c("1" = "Ganganagar","2" = "Hanumangarh","3" = "Bikaner","4" =
"Churu","5" = "Jhunjhunun","6" = "Alwar","7" = "Bharatpur","8" = "Dhaulpur","9" =
"Karauli","10" = "Sawai Madhopur","11" = "Dausa","12" = "Jaipur","13" = "Sikar","14" =
"Nagaur","15" = "Jodhpur","16" = "Jaisalmer","17" = "Barmer","18" = "Jalor","19" =
"Sirohi","20" = "Pali","21" = "Ajmer","22" = "Tonk","23" = "Bundi","24" =
"Bhilwara","25" = "Rajsamand","26" = "Udaipur","27" = "Dungarpur","28" =
"Banswara","29" = "Chittaurgarh","30" = "Kota","31" = "Baran","32" = "Jhalawar")
sector_mapping <- c("2" = "URBAN", "1" = "RURAL")

RJnew$District <- as.character(RJnew$District)
RJnew$Sector <- as.character(RJnew$Sector)
RJnew$District <- ifelse(RJnew$District %in% names(district_mapping),
district_mapping[RJnew$District], RJnew$District)
RJnew$Sector <- ifelse(RJnew$Sector %in% names(sector_mapping),
sector_mapping[RJnew$Sector], RJnew$Sector)

View(RJnew)

hist(RJnew$total_consumption, breaks = 10, col = 'blue', border = 'black',
     xlab = "Consumption", ylab = "Frequency", main = "Consumption Distribution in
Rajasthan State")

RJ_consumption <- aggregate(total_consumption ~ District, data = RJnew, sum)
View(RJ_consumption)
??barplot
barplot(RJ_consumption$total_consumption,
        names.arg = RJ_consumption$District,
        las = 2, # Makes the district names vertical
        col = 'blue',
        border = 'black',
        xlab = "District",
        ylab = "Total Consumption",
        main = "Total Consumption per District",
        cex.names = 0.7) # Adjust the size of district names if needed

# b) Plot {'any variable of your choice'} on the Rajasthan state map using NSSO68.csv data

library(ggplot2)
library(sf) # mapping
library(dplyr)
Sys.setenv("SHAPE_RESTORE_SHX" = "YES")

data_map <- st_read("C:\\A5\\RAJASTHAN_DISTRICTS.geojson")
```

```
View(data_map)

data_map <- data_map %>%
  rename(District = dtname)
colnames(data_map)
data_map_data <- merge(RJ_consumption,data_map,by = "District")
View(data_map_data)
ggplot(data_map_data) +
  geom_sf(aes(fill =total_consumption, geometry = geometry)) +
  scale_fill_gradient(low = "yellow", high = "red") +
  ggtitle("Total Consumption_by_District")

ggplot(data_map_data) +
  geom_sf(aes(fill = total_consumption, geometry = geometry)) +
  scale_fill_gradient(low = "yellow", high = "red") +
  ggtitle("Total Consumption by District") +
  geom_sf_text(aes(label = District, geometry = geometry), size = 3, color = "black")
```