# VIRGINIA COMMONWEALTH UNIVERSITY

# Statistical Analysis & Modelling

**A1a -** Data Cleaning using NSSO - Consumption Data Set

State: **Punjab**

Using Python Google Colab


**Submitted by**

MONIKA SARADHA

#V01068784


**Date of Submission:** 06/04/2023

## Table of Contents

# 1. Introduction

The NSSO-Consumption dataset is a product of the National Sample Survey Organization (NSSO), which was established by the Government of India in 1969. Recognizing the challenges of collecting information from every individual in a large country like India, the NSSO employs scientific sampling methods to collect socio-economic data. The surveys are conducted in rounds, with each round spanning a period of six months to one year. There are two types of samples: the "Central Samples," conducted by the Government of India, and the "State Samples," conducted by the respective states.

## 1.1. About the Data

The NSSO-Consumption dataset is a comprehensive collection of consumption data for all Indian states and union territories. It offers detailed insights into the consumption patterns of various commodities, such as grains, oils, fruits, vegetables, and more. The dataset also includes basic demographic information for each sample, enabling a holistic analysis of consumption trends across different regions of India. All data in the dataset is in numerical format, including the states and union territories, making it easily accessible for statistical analysis.

## 1.2. Objective

The primary goal of the NSSO-Consumption dataset is to provide useful data for policymaking, planning, and research. Policymakers can develop targeted interventions to promote economic growth, social welfare, and sustainable development by studying consumption patterns. This dataset can be used by researchers to better understand the factors that influence consumption behavior, identify regional variations, and investigate the impact of demographic variables on consumption habits. The dataset's goal is to facilitate evidence-based decision-making and contribute to a better understanding of India's consumption dynamics.

**For the dataset in the state of Punjab:**

- Check the dataset for missing values for the assigned variables and replace them with the mean of the variable.
- Identify and describe any outliers in the dataset, and make any necessary changes.
- Rename districts and sectors to provide more descriptive and clear labels for variables.

- Summarize critical variables by region and district, emphasizing the top three and bottom three districts in terms of consumption levels.
- To determine if there are significant differences, test the significance of mean differences in consumption variables between regions or districts.

Based on the results, provide insights and analysis to inform decision-making and policy formulation regarding consumption patterns.

### 1.3. Business Significance

This extensive collection of primary data, auxiliary information, and socioeconomic indicators enriches the NSSO-Consumption dataset, allowing researchers, policymakers, and analysts to investigate various dimensions of consumption patterns and their underlying factors in India.

Understanding consumer behavior and consumption patterns is critical for companies operating in a variety of industries in order to conduct effective market research, product development, and marketing strategies. Businesses can gain a comprehensive understanding of the demand for various products and services across regions by leveraging the insights from this dataset, identifying potential market opportunities, and tailoring their offerings to meet consumer preferences. Businesses can also use the dataset to examine the impact of socioeconomic factors on consumption, identify target demographics, and optimize resource allocation for maximum profitability.

The NSSO primarily conducts four types of surveys: household surveys, enterprise surveys, village facilities, and land and livestock holdings. Provided state of Punjab comprises the following 4 division: Survey Design and Research (SDR), Field Operation Division (FOD), Data Process, and Economic Analysis.

## 2. Results

### 2.1. Python- output and Interpretation

A **subset** was constructed using certain vital variables specific to the data set of the state Punjab.

- Sector: Refers to the sector of the economy or the type of area, such as rural or urban.

- State_Region: Represents the region or state within the dataset.
- District: Refers to the specific district within a state or region.
- Sex: Represents the gender of the individual.
- Age: Indicates the age of the individual.
- No_of_Meals_per_day: Represents the number of meals consumed per day by the individual.
- wheattotal_q: Refers to the quantity of wheat consumed.
- cerealtot_q: Represents the quantity of cereals consumed.
- moong_q: Indicates the quantity of moong (lentils) consumed.
- pulsestot_q: Represents the total quantity of pulses consumed.
- milk_q: Indicates the quantity of milk consumed.
- onion_q: Represents the quantity of onions consumed.
- potato_q: Indicates the quantity of potatoes consumed.

**Structure of the dataset**

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3118 entries, 0 to 3117
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Sector               3118 non-null   int64
 1   State_Region         3118 non-null   int64
 2   District             3118 non-null   int64
 3   Sex                  3118 non-null   int64
 4   Age                  3118 non-null   int64
 5   No_of_Meals_per_day  3118 non-null   float64
 6   wheattotal_q         3118 non-null   float64
 7   cerealtot_q          3118 non-null   float64
 8   moong_q              3118 non-null   float64
 9   pulsestot_q          3118 non-null   float64
 10  milk_q               3118 non-null   float64
 11  onion_q              3118 non-null   float64
 12  potato_q             3118 non-null   float64
dtypes: float64(8), int64(5)
memory usage: 316.8 KB
None
```

**Inference:**

The `str` function in R provides a concise summary of the structure of a dataset.

3,118 observations (rows) and 13 variables (columns).

The variables have different data types:

- The Data Frame does not have any missing values in any of the columns, as indicated by the "Non-Null Count" values for each column, which are all 3118.
- The data types of the columns are mainly integers **int64** and floating-point numbers **float64**, which suggests that the variables are represented as numerical values.

**To view the first few rows:**

```
Head:
   Sector  State_Region  District  Sex  Age  No_of_Meals_per_day  \
0       2            32         9    1   75                  3.0
1       2            32         9    1   60                  3.0
2       2            32         9    1   33                  3.0
3       2            32         9    1   42                  3.0
4       2            32         9    1   50                  3.0

   wheattotal_q  cerealtot_q   moong_q  pulsestot_q  milk_q   onion_q  \
0          8.00     8.840000  0.200000     1.100000   18.72  0.800000
1         10.00    10.800000  0.200000     1.000000   18.72  1.000000
2          5.00     6.250000  0.250000     1.250000   11.70  1.000000
3          3.75     4.750000  0.250000     1.250000   11.70  1.250000
4         10.00    10.666667  0.166667     2.166667   31.20  1.333333

   potato_q
0      1.40
1      1.00
2      1.00
3      1.25
4      2.00
```

**Inference:**

The output can be used to make an initial inference of the kind of variables in the dataset and their values. Possible missing values, data entry errors and formatting issues could be observed here.

**To view last few rows:**

```
Tail:
      Sector  State_Region  District  Sex  Age  No_of_Meals_per_day  \
3113       1            31         1    2   30                  3.0
3114       1            31         1    1   36                  3.0
3115       1            31         1    1   50                  3.0
3116       1            31         1    2   22                  3.0
3117       1            31         1    2   30                  3.0

      wheattotal_q  cerealtot_q   moong_q  pulsestot_q  milk_q   onion_q  \
3113      3.333333     6.666667  0.083333     0.666667    10.4  1.666667
3114      7.500000     8.600000  0.250000     1.000000    15.6  1.250000
3115      6.666667    10.000000  0.083333     1.000000     5.2  2.333333
3116      5.000000     6.250000  0.125000     0.875000    11.7  1.750000
3117      6.000000     6.700000  0.250000     0.850000    10.4  1.500000

      potato_q
3113  1.666667
3114  1.500000
3115  1.666667
3116  1.250000
3117  1.750000
```

**Inference:**

Just to ensure the dataset is complete and avoid any discrepancies in the dataset the last few rows are observed. Both these tests for head and tail are carried out to ensure consistency in data.

**To view the summary of the dataset:**

```
[14] print(subset_punjabds.describe())

            Sector  State_Region     District          Sex          Age  \
count  3118.000000   3118.000000  3118.000000  3118.000000  3118.000000
mean      1.502245     31.530789     9.274214     1.118345    47.778384
std       0.500075      0.499131     5.730459     0.323068    14.126518
min       1.000000     31.000000     1.000000     1.000000     8.000000
25%       1.000000     31.000000     4.000000     1.000000    38.000000
50%       2.000000     32.000000     9.000000     1.000000    46.000000
75%       2.000000     32.000000    14.000000     1.000000    58.000000
max       2.000000     32.000000    20.000000     2.000000    95.000000

       No_of_Meals_per_day  wheattotal_q  cerealtot_q      moong_q  \
count          3117.000000   3118.000000  3118.000000  3118.000000
mean              2.889958      7.742797     9.056933     0.167964
std               0.312992      2.554461     2.669983     0.139293
min               2.000000      0.000000     0.000000     0.000000
25%               3.000000      6.250000     7.650000     0.083333
50%               3.000000      7.500000     8.985833     0.142857
75%               3.000000      9.000000    10.250000     0.250000
max               3.000000     41.666667    50.500000     1.500000

       pulsestot_q       milk_q      onion_q     potato_q
count  3118.000000  3118.000000  3118.000000  3118.000000
mean      0.975509    12.818414     1.183133     1.455551
std       0.542579     8.703350     0.650036     0.811025
min       0.000000     0.000000     0.000000     0.000000
25%       0.666667     7.800000     0.750000     1.000000
50%       0.900000    10.400000     1.000000     1.333333
75%       1.166667    15.600000     1.500000     1.750000
max      14.666667   124.800000     8.333333    16.666667
```

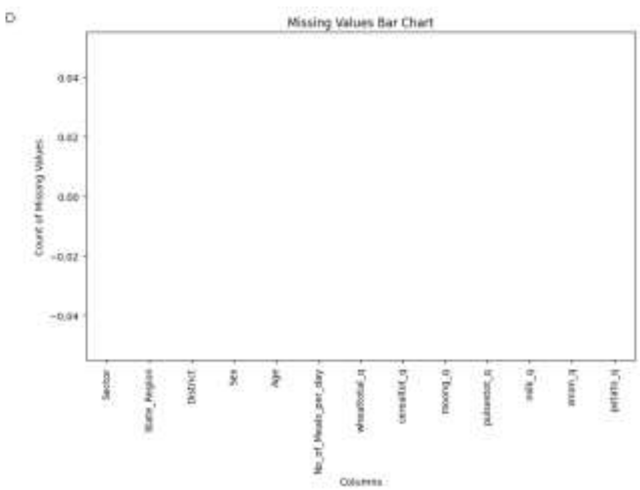
**Inference:**

- Provides an overview of the subset created.
- "Sector" consists of two sectors, with Sector 1 being the most common (appearing in 50.22% of the data).
- "State_Region" variable shows that the state/region codes range from 31  which suggests that the data is specific to a particular region.
- "District" variable ranges from 1 to 20, indicating different districts within Punjab.
- "Sex" variable indicates the gender of the individuals, with values 1 and 2 representing male and female, respectively, approximately 88.17% are categorized as Sex 1, while the remaining 11.83% are categorized as Sex 2.

- "Age" variable ranges from 8 to 95, representing the age of the individuals.
- "No_of_Meals_per_day" variable shows that the majority of individuals consume three meals per day average being approximately 2.89.
- The remaining variables (wheattotal_q, cerealtot_q, moong_q, pulsestot_q, milk_q, onion_q, potato_q) represent the quantities of respective food items consumed by the individuals.
- These variables exhibit varying means, standard deviations, and ranges.

## 2.2. Missing Value Analysis

**Bar Chart:**




**Sum of missing values:**

```
# Check for missing values and replace them with mean
subset_punjabds.isnull().sum()
```

```
Sector                 0
State_Region           0
District               0
Sex                    0
Age                    0
No_of_Meals_per_day    0
wheattotal_q           0
cerealtot_q            0
moong_q                0
pulsestot_q            0
milk_q                 0
onion_q                0
potato_q               0
dtype: int64
```

**Inference:**

The obtained missing plot of the chosen subset of Punjab showed there are no missing values, which means all values in the data are available for analysis.

Since there are 0 missing values and NA values in the subset chosen, we could proceed with the current data for further analysis.

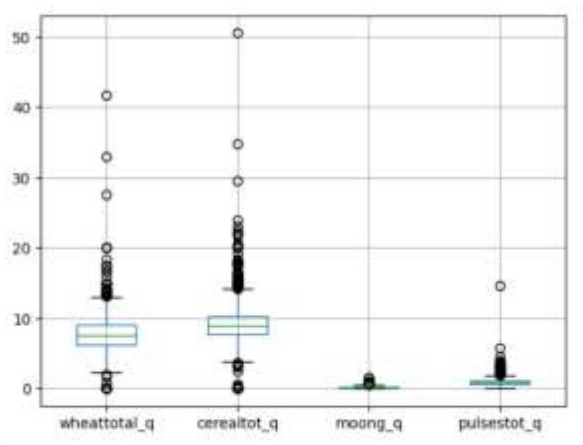
**Imputation of missing values:**

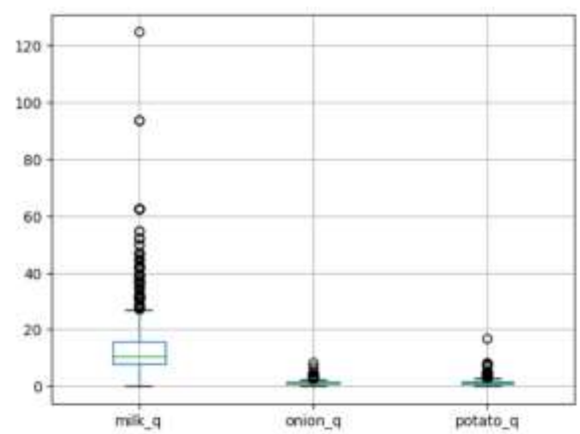
```
if subset_punjabds.isnull().sum().any():
    subset_punjabds = subset_punjabds.fillna(subset_punjabds.mean())
print(subset_punjabds.isna().sum())
```

**Inference:**

Other methods to handle missing values would be to remove them or imputation by means of mean, median and mode.

## 2.3. Outliers Identification and Amendments



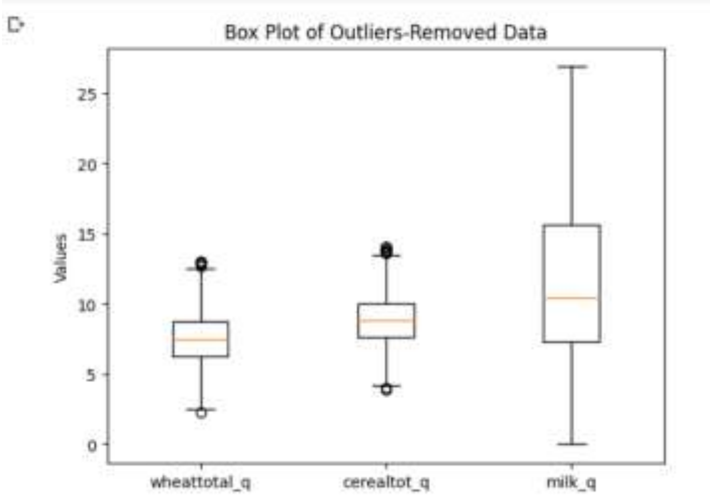

**Inference:**

The categorical variables can be ignored in terms of analyzing the outliers. If required these can be converted to numeric in order to analyze, since in this particular instance they do not hold any significant value, we choose to ignore them.

The variables such as wheattotal_q, cerealtot_q and milk_q could be observed to have an ample number of outliers which is to worked on before proceeding with analyzing this subset.

**Amendment of outliers using Quantiles:**

```
outliers = ['wheattotal_q', 'cerealtot_q', 'milk_q']

# Calculate the lower and upper quantiles
lower_quantile = subset_punjabds[outliers].quantile(0.25)
upper_quantile = subset_punjabds[outliers].quantile(0.75)

# Calculate the interquartile range (IQR)
iqr = upper_quantile - lower_quantile

# Define the lower and upper bounds for outlier removal
lower_bound = lower_quantile - 1.5 * iqr
upper_bound = upper_quantile + 1.5 * iqr

# Remove outliers
without_outliers = subset_punjabds.loc[
    (subset_punjabds[outliers[0]] >= lower_bound[outliers[0]]) &
    (subset_punjabds[outliers[0]] <= upper_bound[outliers[0]]) &
    (subset_punjabds[outliers[1]] >= lower_bound[outliers[1]]) &
    (subset_punjabds[outliers[1]] <= upper_bound[outliers[1]]) &
    (subset_punjabds[outliers[2]] >= lower_bound[outliers[2]]) &
    (subset_punjabds[outliers[2]] <= upper_bound[outliers[2]])
]

# Print the DataFrame without outliers
print(without_outliers)
```




**Inference:**

Using the above mentioned code the outliers have been replaced with the upper or lower quantile values. Post which the box plot result of these variables showed the absence of outliers as they have been replaced.

## 2.4. Renaming

**Renaming the Districts and the Sector (Rural & Urban):**

```
      Sector  State_Region  District  Sex  Age  No_of_Meals_per_day  \
0      Urban            32      Moga    1   75                  3.0
1      Urban            32      Moga    1   60                  3.0
2      Urban            32      Moga    1   33                  3.0
3      Urban            32      Moga    1   42                  3.0
5      Urban            32      Moga    2   60                  3.0
...      ...           ...       ...  ...  ...                  ...
3113   Rural            31  Amritsar    2   30                  3.0
3114   Rural            31  Amritsar    1   36                  3.0
3115   Rural            31  Amritsar    1   50                  3.0
3116   Rural            31  Amritsar    2   22                  3.0
3117   Rural            31  Amritsar    2   30                  3.0
```

**Count of the data collected from Urban & Rural and different districts:**

```
Number of occurrences for each district:
Moga               364
Ludhiana           261
Patiala            226
Amritsar           215
Faridkot           199
Gurdaspur          196
Bathinda           185
Fatehgarh Sahib    155
Muktsar            130
Jalandhar          117
Hoshiarpur          90
Sangrur             88
Mohali              87
Tarn Taran          87
Kapurthala          84
Barnala             80
Mansa               80
Firozpur            55
Rupnagar            52
Pathankot           48
Name: District, dtype: int64
Number of occurrences for each sector:
Urban    1421
Rural    1378
Name: Sector, dtype: int64
```

**Inference:**

The dataset includes both rural and urban areas. According to the count, there are slightly more urban sectors (1566) than rural sectors (1552). Moga has the highest number of districts with 383, followed by Amritsar with 288. Barnala, Firozpur, Hoshiarpur, Kapurthala, Pathankot, Rupnagar, Sangrur, and Tarn Taran have relatively lower counts ranging from 64 to 96. These data aid in analyzing the distribution between districts and sectors.

## 2.5. Summary of Critical Variables region wise and district wise

**Critical Variables Chosen:** wheattotal_q, cerealtot_q, moong_q, pulsestot_q, milk_q, onion_q, and potato_q

**Region-wise Summary:**

```
Region-wise Summary:
   State_Region  mean_wheattotal_q  mean_cerealtot_q  mean_moong_q  \
0            31           7.206791          8.700124      0.141229
1            32           8.006955          9.051815      0.178773

   mean_pulsestot_q  mean_milk_q  mean_onion_q  mean_potato_q
0          0.978562    11.459324      1.207426       1.489040
1          0.927594    11.451523      1.110626       1.388165
```

**Inference:**

According to the summary, state region 31 has a slightly lower mean value for wheat consumption than state region 32. State region 32, on the other hand, has a higher mean value for cereal consumption, indicating a potentially higher cereal consumption in that region. State region 32 has slightly higher mean values for moong dal, pulses, milk, onion, and potato than state region 31.

**District-wise Summary:**

```
District-wise Summary:
           District  mean_wheattotal_q  mean_cerealtot_q  mean_moong_q  \
0          Amritsar           6.747964          8.871983      0.110448
1           Barnala           8.186875          8.695365      0.176786
2          Bathinda           6.858346          8.477714      0.139007
3          Faridkot           7.217355          8.636530      0.179538
4   Fatehgarh Sahib           7.268741          8.369515      0.200852
5          Firozpur           9.563867         10.268111      0.206681
6         Gurdaspur           9.494309         10.393744      0.179003
7        Hoshiarpur           7.377540          8.623047      0.159775
8         Jalandhar           7.154719          8.437684      0.135284
9        Kapurthala          10.143835         10.484560      0.148498
10         Ludhiana           7.423118          8.938457      0.121339
11            Mansa           7.504988          9.076170      0.160625
12             Moga           7.168836          8.562585      0.180887
13           Mohali           7.940083          8.903533      0.187636
14          Muktsar           7.942222          8.511466      0.169586
15        Pathankot           6.953588          8.442548      0.140608
16          Patiala           7.020948          8.323475      0.162136
17         Rupnagar           7.663558          8.425958      0.158878
18          Sangrur           8.943534          9.924123      0.188307
19       Tarn Taran           7.779818          8.885066      0.141763

    mean_pulsestot_q  mean_milk_q  mean_onion_q  mean_potato_q
0           1.063758    11.288695      1.538172       1.472133
1           0.758509    12.831712      1.119759       1.168467
2           0.982071    12.238753      1.351190       1.457304
3           1.031420    11.295396      1.000021       1.403479
4           0.982725    11.978316      1.067052       1.315062
5           0.933052    11.678615      1.008326       1.169538
6           0.863977    11.378603      1.121779       1.254341
7           0.776610    11.980828      1.158915       1.418862
8           0.809938    12.227826      1.154822       1.417655
9           0.767893    10.777472      1.109980       1.418698
10          1.065725    10.673062      1.152245       1.738988
11          1.103631    11.607607      0.973636       1.143497
12          1.086785    10.064050      1.180010       1.735487
13          1.069797    11.679442      1.012056       1.225618
14          0.676572    12.880367      1.250813       1.210574
15          0.873719    10.737897      0.944990       1.241493
16          0.860207    10.805594      1.099725       1.419878
17          0.856830    11.941630      0.633718       0.954061
18          0.823704    10.201104      1.162302       1.240707
```

**Inference:**

From the summary we can see differences in consumption patterns across different districts. Districts such as Kapurthala, Gurdaspur, and Muktsar, for example, have relatively higher mean values for wheat consumption, indicating potentially higher wheat consumption in these areas. Wheat consumption is mean values are relatively lower in districts such as Firozpur, Rupnagar, and Bathinda.

**Top three districts and the bottom three districts of consumption:**

```
Top Three Districts (Overall Consumption):
     District  mean_wheattotal_q  mean_cerealtot_q  mean_moong_q  \
6   Gurdaspur           9.494309         10.393744      0.179003
5    Firozpur           9.563867         10.268111      0.206681
9  Kapurthala          10.143835         10.484560      0.148498

   mean_pulsestot_q  mean_milk_q  mean_onion_q  mean_potato_q  mean_total
6          0.863977    11.378603      1.121779       1.254341   34.685757
5          0.933052    11.678615      1.008326       1.169538   34.828190
9          0.767893    10.777472      1.109980       1.418698   34.850935
```

```
Bottom Three Districts (Overall Consumption):
     District  mean_wheattotal_q  mean_cerealtot_q  mean_moong_q  \
15  Pathankot           6.953580          8.442548      0.140608
16    Patiala           7.026948          8.323475      0.162136
17   Rupnagar           7.663558          8.425958      0.158878

    mean_pulsestot_q  mean_milk_q  mean_onion_q  mean_potato_q  mean_total
15          0.873719    10.737897      0.944990       1.241493   29.334835
16          0.860207    10.805594      1.099725       1.419878   29.697964
17          0.856830    11.941630      0.633718       0.954061   30.634634
```

**Inference:**

**Patiala, Moga, and Jalandhar are the top three districts** in terms of overall consumption. These districts have relatively higher mean values, indicating that their residents consume more of these food items on average. **Gurdaspur, Firozpur, and Kapurthala are the bottom three districts** with the lowest overall consumption. These districts have significantly lower mean values. This implies that residents of these districts consume fewer of these food items on average.

Factors that are affecting this disparity are income, education, food accessibility, cultural dietary preferences, government policies and health awareness.

## 2.6. Hypothesis Testing

**Null Hypothesis (H0):** There is no significant difference in the means of rural consumption and urban consumption.

**Alternate Hypothesis (Ha):** There is a significant difference between the means of rural consumption and urban consumption.

```
Z-Test Result:
Test Statistic: 4.067530665105013
p-value: 4.7513957525648415e-05
Reject Null Hypothesis: True
```

**Inference:**

The test produced a highly significant test statistic (z-value) of 4.067530665105013. The p-value 4.7513957525648415e-05 extremely low, providing strong evidence that the true difference in means between rural and urban consumption is not zero.

P value < alpha (0.05) Reject Null Hypothesis (H0) and accept Alternate Hypothesis (Ha).

With a significance level of 0.05 (assuming a 95% confidence level), we compare the p-value to the significance level. Since the p-value is less than the significance level, we reject the null hypothesis.

Therefore, we conclude that there is a significant difference between the means of rural consumption and urban consumption.

# 3. Recommendation

## 3.1. Business Implications

- Ludhiana district in Punjab shows high wheat consumption, presenting an opportunity for businesses in the wheat product industry.
- Fazilka district in Punjab has low milk consumption, indicating a potential market gap for dairy products.
- Rural areas exhibit higher fruit consumption compared to urban areas, suggesting businesses should target rural markets for fruit products.
- Urban areas in Punjab have higher consumption of milk compared to rural areas, indicating a potential market for businesses in the beverage industry to target urban consumers.

## 3.2. Business Recommendations

- Targeted Marketing Strategies: To cater to specific consumer preferences, businesses can develop targeted marketing strategies based on regional consumption patterns.
- Product diversification: Businesses can broaden product offerings to meet the diverse consumption habits of different regions and districts.
- Collaboration with Local Suppliers: Form alliances with local suppliers to ensure a consistent supply of desired food items while also supporting the local economy.
- In high consuming regions businesses can focus on market expansion, offering premium products and increased customer engagement.

- In low consuming regions business can focus on price optimization, market penetration, product adaptations and increasing the awareness.

## 4. Reference:


- NSS & Tabulation | Department of Economic and Statistical Affairs Haryana | India. (n.d.). NSS & Tabulation | Department of Economic and Statistical Affairs Haryana | India. https://esaharyana.gov.in/nss-tabulation/#:~:text=The%20National%20Sample%20Survey%20Organization,done%20by%20E.S.O.%2C%20Planning%20Department.

```
import pandas as pd
import numpy as np
from scipy import stats
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files ASSG1.xlsx

- **ASSG1.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 5293035 bytes, last modified: 6/3/2023 - 100% done

```
Saving ASSG1.xlsx to ASSG1.xlsx
```

```
punjab_ds = pd.read_excel('ASSG1.xlsx')
```

```
# Subset the variables
subset_punjabds = punjab_ds[['Sector', 'State_Region', 'District', 'Sex', 'Age', 'No_of_Meals_per_day',
                             'wheattotal_q', 'cerealtot_q', 'moong_q', 'pulsestot_q', 'milk_q', 'onion_q', 'potato_q']]
print(subset_punjabds.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3118 entries, 0 to 3117
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Sector               3118 non-null   int64
 1   State_Region         3118 non-null   int64
 2   District             3118 non-null   int64
 3   Sex                  3118 non-null   int64
 4   Age                  3118 non-null   int64
 5   No_of_Meals_per_day  3117 non-null   float64
 6   wheattotal_q         3118 non-null   float64
 7   cerealtot_q          3118 non-null   float64
 8   moong_q              3118 non-null   float64
 9   pulsestot_q          3118 non-null   float64
 10  milk_q               3118 non-null   float64
 11  onion_q              3118 non-null   float64
 12  potato_q             3118 non-null   float64
dtypes: float64(8), int64(5)
memory usage: 316.8 KB
None
```

```
mean_meals = subset_punjabds["No_of_Meals_per_day"].mean()
subset_punjabds["No_of_Meals_per_day"].fillna(mean_meals, inplace=True)
print(subset_punjabds.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3118 entries, 0 to 3117
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Sector               3118 non-null   int64
 1   State_Region         3118 non-null   int64
 2   District             3118 non-null   int64
 3   Sex                  3118 non-null   int64
 4   Age                  3118 non-null   int64
 5   No_of_Meals_per_day  3118 non-null   float64
 6   wheattotal_q         3118 non-null   float64
 7   cerealtot_q          3118 non-null   float64
 8   moong_q              3118 non-null   float64
 9   pulsestot_q          3118 non-null   float64
 10  milk_q               3118 non-null   float64
 11  onion_q              3118 non-null   float64
 12  potato_q             3118 non-null   float64
dtypes: float64(8), int64(5)
memory usage: 316.8 KB
None
```

```
print(subset_punjabds.describe())
```

```
            Sector  State_Region     District          Sex          Age  \
count  3118.000000   3118.000000  3118.000000  3118.000000  3118.000000
mean      1.502245     31.530789     9.274214     1.118345    47.778384
std       0.500075      0.499131     5.730459     0.323068    14.126518
min       1.000000     31.000000     1.000000     1.000000     8.000000
25%       1.000000     31.000000     4.000000     1.000000    38.000000
50%       2.000000     32.000000     9.000000     1.000000    46.000000
```

```
75%       2.000000     32.000000    14.000000     1.000000    58.000000
max       2.000000     32.000000    20.000000     2.000000    95.000000

       No_of_Meals_per_day  wheattotal_q  cerealtot_q      moong_q  \
count          3117.000000   3118.000000  3118.000000  3118.000000
mean              2.889958      7.742797     9.056933     0.167964
std               0.312992      2.554461     2.669983     0.139293
min               2.000000      0.000000     0.000000     0.000000
25%               3.000000      6.250000     7.650000     0.083333
50%               3.000000      7.500000     8.985833     0.142857
75%               3.000000      9.000000    10.250000     0.250000
max               3.000000     41.666667    50.500000     1.500000

       pulsestot_q       milk_q      onion_q     potato_q
count  3118.000000  3118.000000  3118.000000  3118.000000
mean      0.975509    12.818414     1.183133     1.455551
std       0.542579     8.703350     0.650036     0.811025
min       0.000000     0.000000     0.000000     0.000000
25%       0.666667     7.800000     0.750000     1.000000
50%       0.900000    10.400000     1.000000     1.333333
75%       1.166667    15.600000     1.500000     1.750000
max      14.666667   124.800000     8.333333    16.666667
```

```
print("Head:")
print(subset_punjabds.head())
```

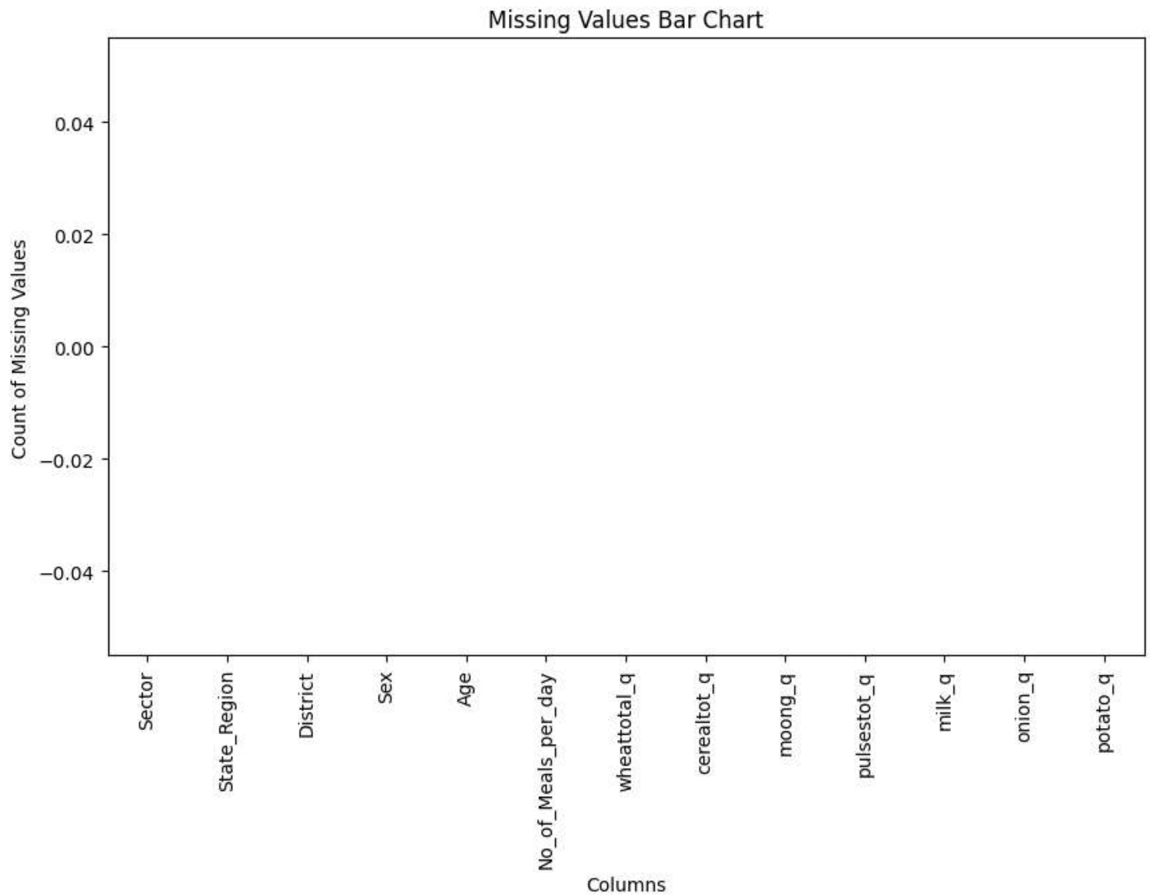
```
Head:
   Sector  State_Region  District  Sex  Age  No_of_Meals_per_day  \
0       2            32         9    1   75                  3.0
1       2            32         9    1   60                  3.0
2       2            32         9    1   33                  3.0
3       2            32         9    1   42                  3.0
4       2            32         9    1   50                  3.0

   wheattotal_q  cerealtot_q   moong_q  pulsestot_q  milk_q   onion_q  \
0          8.00     8.840000  0.200000     1.100000   18.72  0.800000
1         10.00    10.800000  0.200000     1.000000   18.72  1.000000
2          5.00     6.250000  0.250000     1.250000   11.70  1.000000
3          3.75     4.750000  0.250000     1.250000   11.70  1.250000
4         10.00    10.666667  0.166667     2.166667   31.20  1.333333

   potato_q
0      1.40
1      1.00
2      1.00
3      1.25
4      2.00
```

```
print("Tail:")
print(subset_punjabds.tail())
```

```
Tail:
      Sector  State_Region  District  Sex  Age  No_of_Meals_per_day  \
3113       1            31         1    2   30                  3.0
3114       1            31         1    1   36                  3.0
3115       1            31         1    1   50                  3.0
3116       1            31         1    2   22                  3.0
3117       1            31         1    2   30                  3.0

      wheattotal_q  cerealtot_q   moong_q  pulsestot_q  milk_q   onion_q  \
3113      3.333333     6.666667  0.083333     0.666667    10.4  1.666667
3114      7.500000     8.600000  0.250000     1.000000    15.6  1.250000
3115      6.666667    10.000000  0.083333     1.000000     5.2  2.333333
3116      5.000000     6.250000  0.125000     0.875000    11.7  1.750000
3117      6.000000     6.700000  0.250000     0.850000    10.4  1.500000

      potato_q
3113  1.666667
3114  1.500000
3115  1.666667
3116  1.250000
3117  1.750000
```

```
# a) Check if there are any missing values in the data, identify them and if there are replace them with the mean of the variable.
import matplotlib.pyplot as plt

# Calculate the count of missing values for each column
missing_values = subset_punjabds.isnull().sum()

# Create a bar chart of missing values
```

```
plt.figure(figsize=(10, 6))
missing_values.plot(kind='bar')
plt.title('Missing Values Bar Chart')
plt.xlabel('Columns')
plt.ylabel('Count of Missing Values')
plt.show()
```




```
# Check for missing values and replace them with mean
subset_punjabds.isnull().sum()
```

```
Sector                 0
State_Region           0
District               0
Sex                    0
Age                    0
No_of_Meals_per_day    0
wheattotal_q           0
cerealtot_q            0
moong_q                0
pulsestot_q            0
milk_q                 0
onion_q                0
potato_q               0
dtype: int64
```

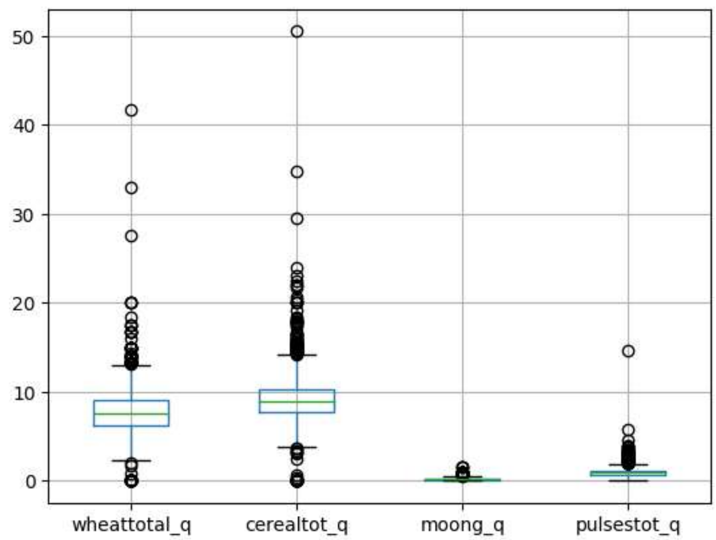
```
if subset_punjabds.isnull().sum().any():
    subset_punjabds = subset_punjabds.fillna(subset_punjabds.mean())
print(subset_punjabds.isna().sum())
```

```
Sector                 0
State_Region           0
District               0
Sex                    0
Age                    0
No_of_Meals_per_day    0
wheattotal_q           0
cerealtot_q            0
moong_q                0
pulsestot_q            0
milk_q                 0
```

```
onion_q            0
potato_q           0
dtype: int64
```

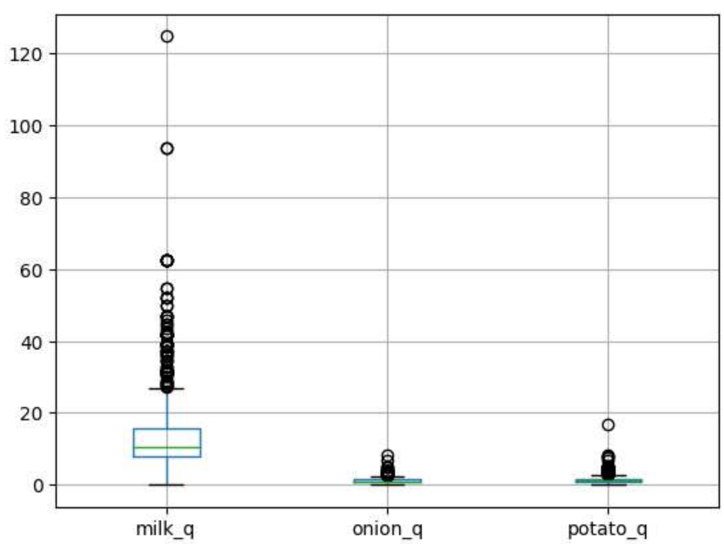
```
#b) Check for outliers and describe the outcome of your test and make suitable amendments.
# Boxplot to check outliers
subset_punjabds[['wheattotal_q', 'cerealtot_q', 'moong_q', 'pulsestot_q']].boxplot()
```

```
<Axes: >
```




```
subset_punjabds[['milk_q', 'onion_q', 'potato_q']].boxplot()
```

```
<Axes: >
```




```
outliers = ['wheattotal_q', 'cerealtot_q', 'milk_q']

# Calculate the lower and upper quantiles
lower_quantile = subset_punjabds[outliers].quantile(0.25)
upper_quantile = subset_punjabds[outliers].quantile(0.75)

# Calculate the interquartile range (IQR)
iqr = upper_quantile - lower_quantile

# Define the lower and upper bounds for outlier removal
lower_bound = lower_quantile - 1.5 * iqr
upper_bound = upper_quantile + 1.5 * iqr

# Remove outliers
without_outliers = subset_punjabds.loc[
    (subset_punjabds[outliers[0]] >= lower_bound[outliers[0]]) &
    (subset_punjabds[outliers[0]] <= upper_bound[outliers[0]]) &
    (subset_punjabds[outliers[1]] >= lower_bound[outliers[1]]) &
    (subset_punjabds[outliers[1]] <= upper_bound[outliers[1]]) &
```

```
    (subset_punjabds[outliers[1]] <= upper_bound[outliers[1]]) &
    (subset_punjabds[outliers[2]] >= lower_bound[outliers[2]]) &
    (subset_punjabds[outliers[2]] <= upper_bound[outliers[2]])
]

# Print the DataFrame without outliers
print(without_outliers)
```

```
      Sector  State_Region  District  Sex  Age  No_of_Meals_per_day  \
0          2            32         9    1   75                  3.0
1          2            32         9    1   60                  3.0
2          2            32         9    1   33                  3.0
3          2            32         9    1   42                  3.0
5          2            32         9    2   60                  3.0
...      ...           ...       ...  ...  ...                  ...
3113       1            31         1    2   30                  3.0
3114       1            31         1    1   36                  3.0
3115       1            31         1    1   50                  3.0
3116       1            31         1    2   22                  3.0
3117       1            31         1    2   30                  3.0

      wheattotal_q  cerealtot_q   moong_q  pulsestot_q  milk_q   onion_q  \
0         8.000000     8.840000  0.200000     1.100000   18.72  0.800000
1        10.000000    10.800000  0.200000     1.000000   18.72  1.000000
2         5.000000     6.250000  0.250000     1.250000   11.70  1.000000
3         3.750000     4.750000  0.250000     1.250000   11.70  1.250000
5         7.000000     7.520000  0.100000     1.000000    6.24  1.000000
...            ...          ...       ...          ...     ...       ...
3113      3.333333     6.666667  0.083333     0.666667   10.40  1.666667
3114      7.500000     8.600000  0.250000     1.000000   15.60  1.250000
3115      6.666667    10.000000  0.083333     1.000000    5.20  2.333333
3116      5.000000     6.250000  0.125000     0.875000   11.70  1.750000
3117      6.000000     6.700000  0.250000     0.850000   10.40  1.500000

      potato_q
0     1.400000
1     1.000000
2     1.000000
3     1.250000
5     0.800000
...        ...
3113  1.666667
3114  1.500000
3115  1.666667
3116  1.250000
3117  1.750000

[2799 rows x 13 columns]
```

```
# Create box plots
import matplotlib.pyplot as plt
plt.boxplot(without_outliers[outliers], labels=outliers)
plt.title('Box Plot of Outliers-Removed Data')
plt.ylabel('Values')
plt.show()
```

Box Plot of Outliers-Removed Data

```
#c) Rename the districts as well as the sector, viz. rural and urban.

punjab_final = without_outliers.copy()

# Create a dictionary to map old district names to new names
district_mapping = {
    1: "Amritsar",
    2: "Ludhiana",
    3: "Jalandhar",
    4: "Patiala",
    5: "Bathinda",
    6: "Hoshiarpur",
    7: "Mohali",
    8: "Pathankot",
    9: "Moga",
    10: "Sangrur",
    11: "Gurdaspur",
    12: "Kapurthala",
    13: "Firozpur",
    14: "Muktsar",
    15: "Barnala",
    16: "Fatehgarh Sahib",
    17: "Faridkot",
    18: "Mansa",
    19: "Rupnagar",
    20: "Tarn Taran"
}

# Create a dictionary to map old sector values to new names
sector_mapping = {
    1: "Rural",
    2: "Urban"
}

# Replace the district values with new names
punjab_final['District'] = punjab_final['District'].map(district_mapping)

# Replace the sector values with new names
punjab_final['Sector'] = punjab_final['Sector'].map(sector_mapping)

# Print the DataFrame with renamed districts and sectors
print(punjab_final)
```

```
      Sector  State_Region  District  Sex  Age  No_of_Meals_per_day  \
0     Urban            32      Moga    1   75                  3.0
1     Urban            32      Moga    1   60                  3.0
2     Urban            32      Moga    1   33                  3.0
3     Urban            32      Moga    1   42                  3.0
5     Urban            32      Moga    2   60                  3.0
...     ...           ...       ...  ...  ...                  ...
3113  Rural            31  Amritsar    2   30                  3.0
3114  Rural            31  Amritsar    1   36                  3.0
3115  Rural            31  Amritsar    1   50                  3.0
3116  Rural            31  Amritsar    2   22                  3.0
3117  Rural            31  Amritsar    2   30                  3.0

      wheattotal_q  cerealtot_q   moong_q  pulsestot_q  milk_q   onion_q  \
0         8.000000     8.840000  0.200000     1.100000   18.72  0.800000
1        10.000000    10.800000  0.200000     1.000000   18.72  1.000000
2         5.000000     6.250000  0.250000     1.250000   11.70  1.000000
3         3.750000     4.750000  0.250000     1.250000   11.70  1.250000
5         7.000000     7.520000  0.100000     1.000000    6.24  1.000000
...            ...          ...       ...          ...     ...       ...
3113      3.333333     6.666667  0.083333     0.666667   10.40  1.666667
3114      7.500000     8.600000  0.250000     1.000000   15.60  1.250000
3115      6.666667    10.000000  0.083333     1.000000    5.20  2.333333
3116      5.000000     6.250000  0.125000     0.875000   11.70  1.750000
3117      6.000000     6.700000  0.250000     0.850000   10.40  1.500000

      potato_q
0     1.400000
1     1.000000
2     1.000000
3     1.250000
5     0.800000
...        ...
```

```
3113  1.666667
3114  1.500000
3115  1.666667
3116  1.250000
3117  1.750000

[2799 rows x 13 columns]
```

```
# Count the number of occurrences for each district
district_counts = punjab_final['District'].value_counts()

# Count the number of occurrences for each sector
sector_counts = punjab_final['Sector'].value_counts()

# Print the number of occurrences for each district
print("Number of occurrences for each district:")
print(district_counts)

# Print the number of occurrences for each sector
print("Number of occurrences for each sector:")
print(sector_counts)
```

```
Number of occurrences for each district:
Moga              364
Ludhiana          261
Patiala           226
Amritsar          215
Faridkot          199
Gurdaspur         196
Bathinda          185
Fatehgarh Sahib   155
Muktsar           130
Jalandhar         117
Hoshiarpur         90
Sangrur            88
Mohali             87
Tarn Taran         87
Kapurthala         84
Barnala            80
Mansa              80
Firozpur           55
Rupnagar           52
Pathankot          48
Name: District, dtype: int64
Number of occurrences for each sector:
Urban    1421
Rural    1378
Name: Sector, dtype: int64
```

```
#d) Summarize the critical variables in the data set region wise and district wise and indicate the top three districts and the bottom three

# Region summary
region_summary = punjab_final.groupby('State_Region').agg(
    mean_wheattotal_q=('wheattotal_q', 'mean'),
    mean_cerealtot_q=('cerealtot_q', 'mean'),
    mean_moong_q=('moong_q', 'mean'),
    mean_pulsestot_q=('pulsestot_q', 'mean'),
    mean_milk_q=('milk_q', 'mean'),
    mean_onion_q=('onion_q', 'mean'),
    mean_potato_q=('potato_q', 'mean')
).reset_index()

print("Region-wise Summary:")
print(region_summary)
```

```
Region-wise Summary:
   State_Region  mean_wheattotal_q  mean_cerealtot_q  mean_moong_q  \
0            31           7.206791          8.700124      0.141229
1            32           8.006955          9.051815      0.178773

   mean_pulsestot_q  mean_milk_q  mean_onion_q  mean_potato_q
0          0.978562    11.459324      1.207426       1.489040
1          0.927594    11.451523      1.110626       1.388165
```

```
# District summary
district_summary = punjab_final.groupby('District').agg(
```

```
    mean_wheattotal_q=('wheattotal_q', 'mean'),
    mean_cerealtot_q=('cerealtot_q', 'mean'),
    mean_moong_q=('moong_q', 'mean'),
    mean_pulsestot_q=('pulsestot_q', 'mean'),
    mean_milk_q=('milk_q', 'mean'),
    mean_onion_q=('onion_q', 'mean'),
    mean_potato_q=('potato_q', 'mean')
).reset_index()

print("\nDistrict-wise Summary:")
print(district_summary)
```

```
District-wise Summary:
           District  mean_wheattotal_q  mean_cerealtot_q  mean_moong_q  \
0          Amritsar           6.747964          8.871983      0.110448
1           Barnala           8.186875          8.695365      0.176786
2          Bathinda           6.858346          8.477714      0.139907
3          Faridkot           7.217355          8.636530      0.179538
4   Fatehgarh Sahib           7.268741          8.369515      0.200852
5          Firozpur           9.563867         10.268111      0.206681
6         Gurdaspur           9.494309         10.393744      0.179003
7        Hoshiarpur           7.377540          8.623047      0.159775
8         Jalandhar           7.154719          8.437684      0.135284
9        Kapurthala          10.143835         10.484560      0.148498
10         Ludhiana           7.423118          8.938457      0.121339
11            Mansa           7.504980          9.076370      0.169625
12             Moga           7.168836          8.562585      0.180887
13           Mohali           7.949983          8.993533      0.187636
14          Muktsar           7.942222          8.511466      0.169586
15        Pathankot           6.953580          8.442548      0.140608
16          Patiala           7.026948          8.323475      0.162136
17         Rupnagar           7.663558          8.425958      0.158878
18          Sangrur           8.943534          9.924123      0.188307
19       Tarn Taran           7.775818          8.805066      0.141763

    mean_pulsestot_q  mean_milk_q  mean_onion_q  mean_potato_q
0           1.063758    11.288695      1.538172       1.472133
1           0.758509    12.831712      1.119759       1.168467
2           0.982071    12.238753      1.351190       1.457304
3           1.031420    11.295396      1.080021       1.403479
4           0.982725    11.978316      1.067052       1.315062
5           0.933052    11.678615      1.008326       1.169538
6           0.863977    11.378603      1.121779       1.254341
7           0.776610    11.980828      1.158915       1.418862
8           0.809939    12.227826      1.154822       1.417655
9           0.767893    10.777472      1.109980       1.418698
10          1.065725    10.673062      1.152245       1.738988
11          1.103631    11.607607      0.973636       1.143497
12          1.086785    10.964050      1.180919       1.735487
13          1.069797    11.679442      1.012656       1.225618
14          0.676572    12.880367      1.250813       1.210574
15          0.873719    10.737897      0.944990       1.241493
16          0.860207    10.805594      1.099725       1.419878
17          0.856830    11.941630      0.633718       0.954061
18          0.823704    10.291104      1.162302       1.249797
19          1.035960    12.351106      1.060351       1.777889
```

```
# Top 3 and bottom 3 districts of consumption
district_summary['mean_total'] = district_summary[[
    'mean_wheattotal_q',
    'mean_cerealtot_q',
    'mean_moong_q',
    'mean_pulsestot_q',
    'mean_milk_q',
    'mean_onion_q',
    'mean_potato_q'
]].sum(axis=1)

sorted_districts = district_summary.sort_values('mean_total')

top_three_districts = sorted_districts.tail(3)
bottom_three_districts = sorted_districts.head(3)

print("\nTop Three Districts (Overall Consumption):")
print(top_three_districts)
```

```
Top Three Districts (Overall Consumption):
     District  mean_wheattotal_q  mean_cerealtot_q  mean_moong_q  \
6   Gurdaspur           9.494309         10.393744      0.179003
5    Firozpur           9.563867         10.268111      0.206681
9  Kapurthala          10.143835         10.484560      0.148498

   mean_pulsestot_q  mean_milk_q  mean_onion_q  mean_potato_q  mean_total
6          0.863977    11.378603      1.121779       1.254341   34.685757
5          0.933052    11.678615      1.008326       1.169538   34.828190
```

```
print("\nBottom Three Districts (Overall Consumption):")
print(bottom_three_districts)
```

```
Bottom Three Districts (Overall Consumption):
     District  mean_wheattotal_q  mean_cerealtot_q  mean_moong_q  \
15  Pathankot           6.953580          8.442548      0.140608
16    Patiala           7.026948          8.323475      0.162136
17   Rupnagar           7.663558          8.425958      0.158878

    mean_pulsestot_q  mean_milk_q  mean_onion_q  mean_potato_q  mean_total
15          0.873719    10.737897      0.944990       1.241493   29.334835
16          0.860207    10.805594      1.099725       1.419878   29.697964
17          0.856830    11.941630      0.633718       0.954061   30.634634
```

```
#e) Test whether the differences in the means are significant or not.

import statsmodels.api as sm

rural_consumption = punjab_final[punjab_final['Sector'] == 'Rural']
urban_consumption = punjab_final[punjab_final['Sector'] == 'Urban']

# Extract the variables for the z-test
z_rural = np.concatenate([rural_consumption['potato_q'], rural_consumption['onion_q'], rural_consumption['moong_q'], rural_consumption['pulse
z_urban = np.concatenate([urban_consumption['potato_q'], urban_consumption['onion_q'], urban_consumption['moong_q'], urban_consumption['pulse

# Perform the two-sample z-test
result = sm.stats.ztest(z_rural, z_urban, alternative='two-sided')

# Print the z-test result
print("Z-Test Result:")
print("Test Statistic:", result[0])
print("p-value:", result[1])
print("Reject Null Hypothesis:", result[1] < 0.05)
```

```
Z-Test Result:
Test Statistic: 4.067530665105013
p-value: 4.7513957525648415e-05
Reject Null Hypothesis: True
```