```
# Required Libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="whitegrid")

# Load dataset
df = pd.read_csv("/content/drive/MyDrive/Food data.csv")
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 1822,\n  \"fields\":
[\n    {\n      \"column\": \"ID\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 526,\n        \"min\": 0,\n
\"max\": 1821,\n        \"num_unique_values\": 1822,\n
\"samples\": [\n          555,\n          1741,\n          297\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"date\",\n      \"properties\":
{\n        \"dtype\": \"object\",\n        \"num_unique_values\":
867,\n        \"samples\": [\n          \"12/24/2023\",\n
\"10/13/2022\",\n          \"4/14/2024\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"meals_served\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
494.791972709125,\n        \"min\": 100.0,\n        \"max\": 4730.0,\n
\"num_unique_values\": 373,\n        \"samples\": [\n          242.0,\
n          470.0,\n          265.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"kitchen_staff\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 17,\n        \"samples\": [\n          \"13\",\
n          \"15\",\n          \"18\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"temperature_C\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
8.919939179273696,\n        \"min\": -10.37220651,\n        \"max\":
60.0,\n        \"num_unique_values\": 892,\n        \"samples\": [\n
34.70688178,\n          30.57935039,\n          20.09944691\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"humidity_percent\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 17.326232260928485,\n        \"min\":
30.12111106,\n        \"max\": 89.98282825,\n
\"num_unique_values\": 867,\n        \"samples\": [\n
31.96857745,\n          51.37113883,\n          43.60364294\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"day_of_week\",\n      \"properties\": {\n        \"dtype\":
```

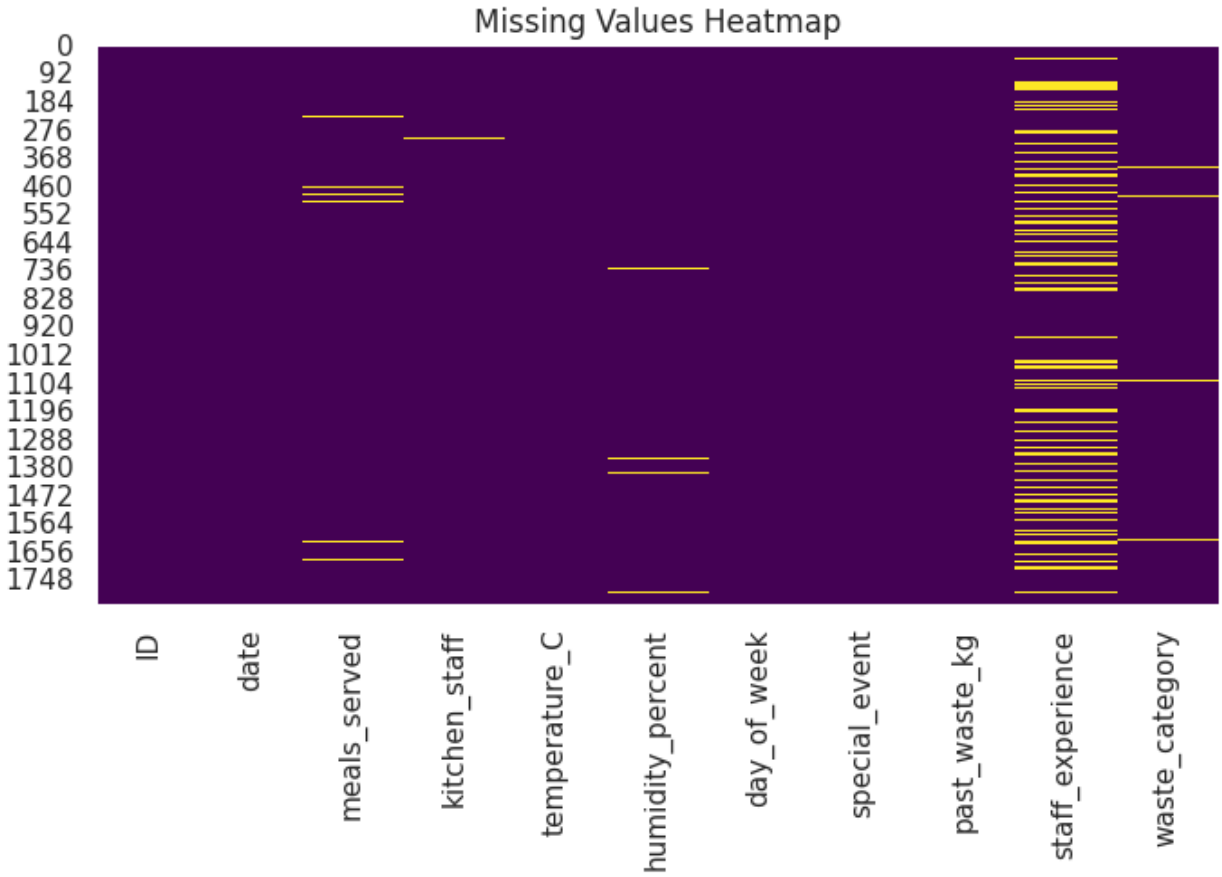
\"number\",\n   \"std\": 2,\n   \"min\": 0,\n
\"max\": 6,\n   \"num_unique_values\": 7,\n   \"samples\":
[\n   0,\n   1,\n   4\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"special_event\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n   \"samples\": [\n   \"0\",\n
\"1\",\n   \"One \"\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"past_waste_kg\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 12.791890523288723,\n
\"min\": 5.008393768,\n   \"max\": 49.80370251,\n
\"num_unique_values\": 867,\n   \"samples\": [\n
16.13298862,\n   42.96897698,\n   38.72205575\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"staff_experience\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 5,\n   \"samples\":
[\n   \"Beginner\",\n   \"Pro\",\n
\"Intermediate\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"waste_category\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 7,\n   \"samples\":
[\n   \"dairy\",\n   \"MeAt\",\n   \"Wheat\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"df"}

```
# Check missing values

missing = df.isnull().sum()
print("Missing values:\n", missing)
```

```
Missing values:
 ID                   0
date                  0
meals_served         32
kitchen_staff        18
temperature_C         0
humidity_percent     16
day_of_week           0
special_event         0
past_waste_kg        16
staff_experience    337
waste_category       21
dtype: int64
```

```
# Visualize missing values

plt.figure(figsize=(8, 4))
sns.heatmap(df.isnull(), cbar=False, cmap="viridis")
```

```
plt.title("Missing Values Heatmap")
plt.show()
```




```
# Fill numeric with median

df['meals_served'].fillna(df['meals_served'].median(), inplace=True)
df['past_waste_kg'].fillna(df['past_waste_kg'].median(), inplace=True)
```

```
<ipython-input-88-9fa918f3409f>:3: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df['meals_served'].fillna(df['meals_served'].median(), inplace=True)
```

```
<ipython-input-88-9fa918f3409f>:4: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df['past_waste_kg'].fillna(df['past_waste_kg'].median(),
inplace=True)

# Fill categorical with mode

df['staff_experience'].fillna(df['staff_experience'].mode()[0],
inplace=True)
df['waste_category'].fillna(df['waste_category'].mode()[0],
inplace=True)

<ipython-input-89-003864591a8d>:3: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df['staff_experience'].fillna(df['staff_experience'].mode()[0],
inplace=True)
<ipython-input-89-003864591a8d>:4: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.
```

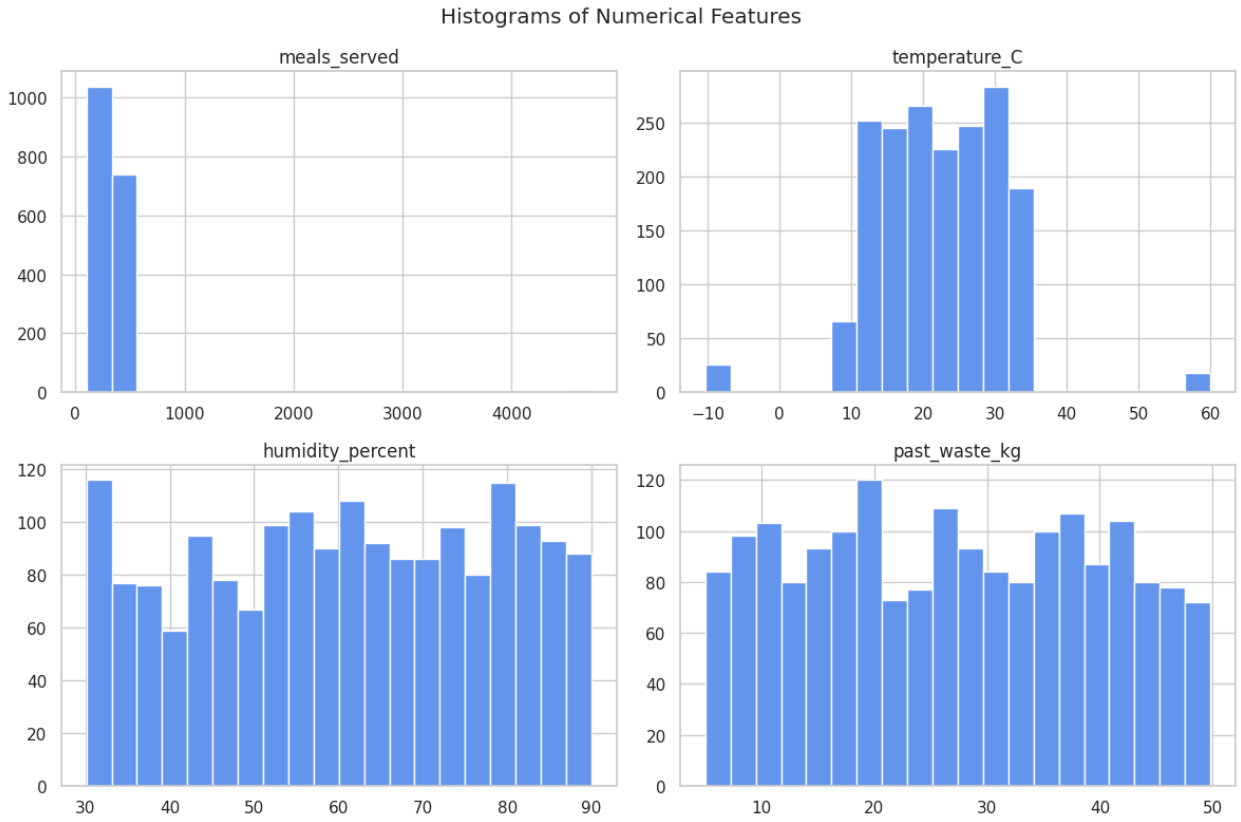
```
  df['waste_category'].fillna(df['waste_category'].mode()[0],
inplace=True)

# Check and remove duplicates

print(f"Duplicates: {df.duplicated().sum()}")
df.drop_duplicates(inplace=True)

Duplicates: 0

# Encoding 'staff_experience'

experience_map = {'Beginner': 0, 'Intermediate': 1, 'Expert': 2}
df['staff_experience_encoded'] =
df['staff_experience'].map(experience_map)

# Check unique categories

print(df['waste_category'].unique())

['dairy' 'MeAt' 'MEAT' 'Vegetables' 'GRAINS' 'Wheat' 'Barley']

# Convert 'date' to datetime

df['date'] = pd.to_datetime(df['date'])

# Verify data types

print(df.dtypes)

ID                                 int64
date                      datetime64[ns]
meals_served                      float64
kitchen_staff                      object
temperature_C                     float64
humidity_percent                  float64
day_of_week                         int64
special_event                      object
past_waste_kg                     float64
staff_experience                   object
waste_category                     object
staff_experience_encoded          float64
dtype: object
```

1. Exploratory Data Analysis (EDA)

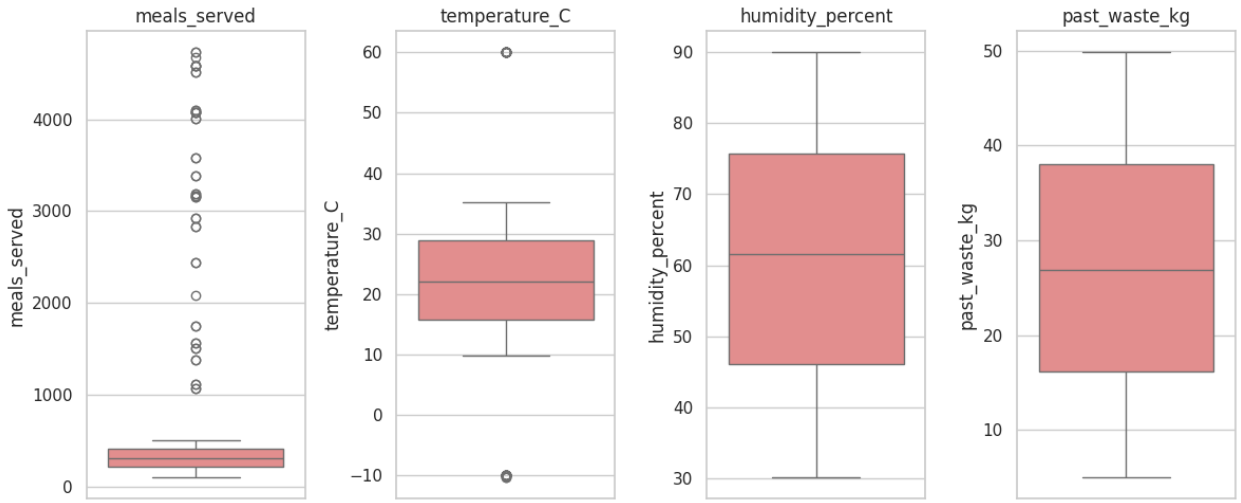
```
# Summary stats

print(df.describe())
print("\nMedian values:\n", df.median(numeric_only=True))
```

```
                ID                             date  meals_served  \
count  1822.000000                             1822   1822.000000
mean    910.500000  2023-04-22 09:30:37.541163520    372.327113
min       0.000000            2022-01-01 00:00:00    100.000000
25%     455.250000            2022-07-25 06:00:00    212.250000
50%     910.500000            2023-04-23 00:00:00    306.000000
75%    1365.750000            2024-01-07 18:00:00    405.750000
max    1821.000000            2024-09-26 00:00:00   4730.000000
std     526.110413                            NaN    490.505492

       temperature_C  humidity_percent  day_of_week  past_waste_kg  \
count    1822.000000       1806.000000   1822.00000    1822.000000
mean       22.189280         60.791257      3.01427      26.996085
min       -10.372207         30.121111      0.00000       5.008394
25%        15.684259         46.035158      1.00000      16.148956
50%        22.115040         61.634935      3.00000      26.832569
75%        28.807494         75.789317      5.00000      37.978663
max        60.000000         89.982828      6.00000      49.803703
std         8.919939         17.326232      2.00899      12.735579

       staff_experience_encoded
count               1086.000000
mean                   0.338858
min                    0.000000
25%                    0.000000
50%                    0.000000
75%                    1.000000
max                    1.000000
std                    0.473540

Median values:
 ID                          910.500000
meals_served                306.000000
temperature_C                22.115040
humidity_percent             61.634935
day_of_week                   3.000000
past_waste_kg                26.832569
staff_experience_encoded      0.000000
dtype: float64
```

```
# Histograms:

df[['meals_served', 'temperature_C', 'humidity_percent',
'past_waste_kg']].hist(
    bins=20, figsize=(12, 8), color='cornflowerblue'
)
plt.suptitle("Histograms of Numerical Features")
plt.tight_layout()
plt.show()
```

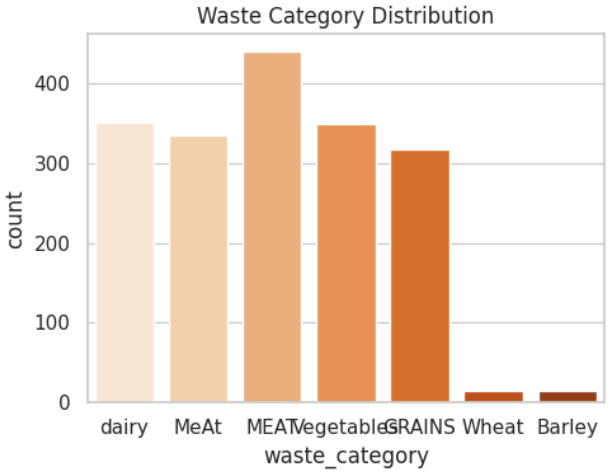



```
# Boxplots:

plt.figure(figsize=(12, 5))
cols = ['meals_served', 'temperature_C', 'humidity_percent',
'past_waste_kg']
for i, col in enumerate(cols):
    plt.subplot(1, 4, i+1)
    sns.boxplot(y=df[col], color='lightcoral')
    plt.title(col)
plt.tight_layout()
plt.show()
```

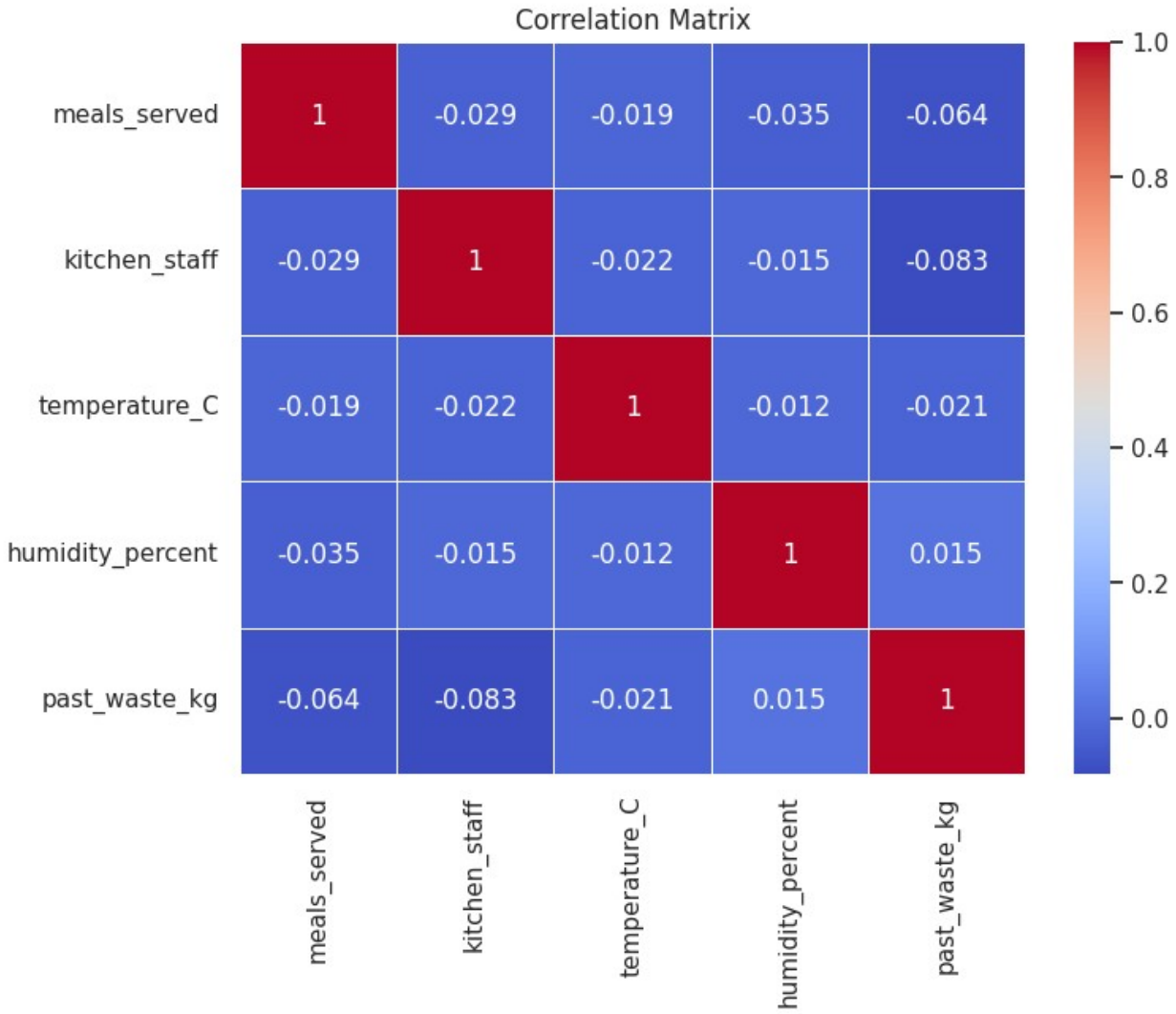



```
# Bar Plots:

plt.figure(figsize=(10, 4))

# Staff Experience
plt.subplot(1, 2, 1)
sns.countplot(x='staff_experience', data=df, palette='Blues')
plt.title("Staff Experience Distribution")

# Waste Category
plt.subplot(1, 2, 2)
sns.countplot(x='waste_category', data=df, palette='Oranges')
plt.title("Waste Category Distribution")

plt.tight_layout()
plt.show()
```

```
<ipython-input-96-cfe5ad575cc9>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(x='staff_experience', data=df, palette='Blues')
<ipython-input-96-cfe5ad575cc9>:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(x='waste_category', data=df, palette='Oranges')
```




```
# Correlation matrix:
cols = ['meals_served', 'kitchen_staff', 'temperature_C',
'humidity_percent', 'past_waste_kg']

# Convert all columns into numeric:
df[cols] = df[cols].apply(pd.to_numeric, errors='coerce')

corr_matrix = df[cols].corr()

plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title("Correlation Matrix")
plt.show()
```

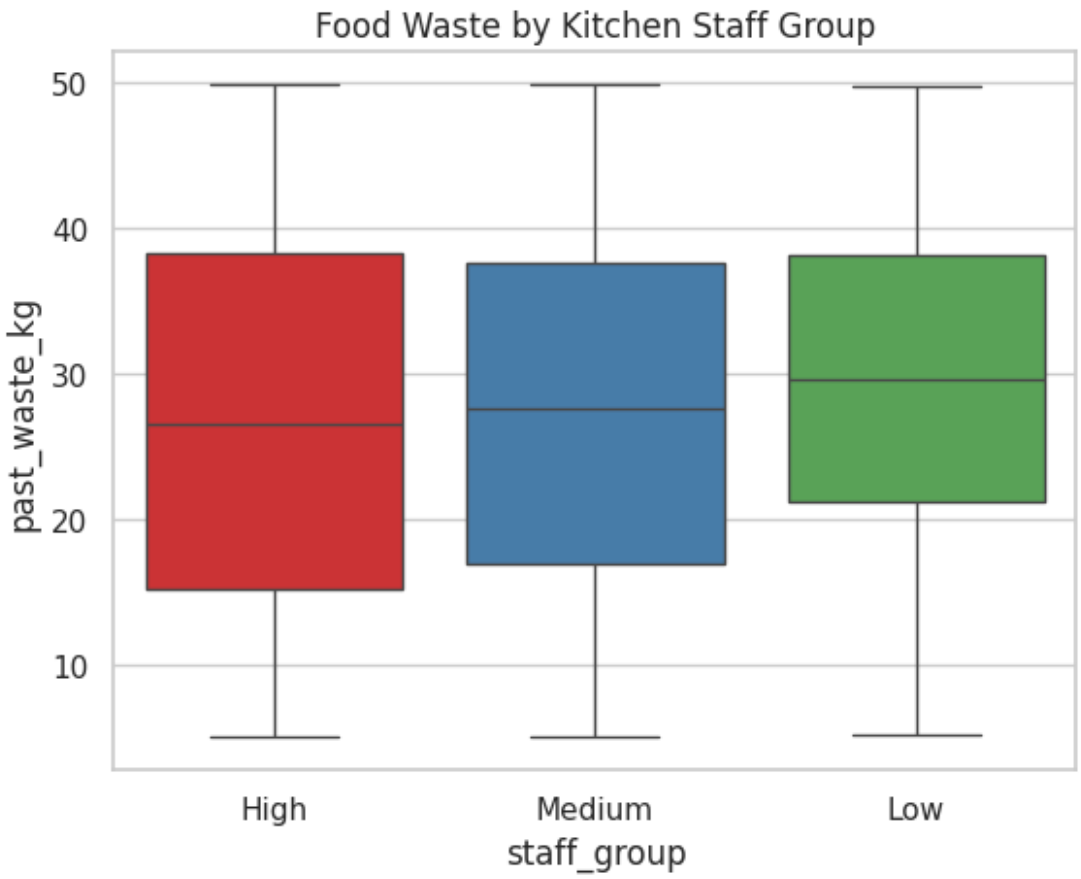


3.Hypothesis Testing

```
from scipy.stats import f_oneway

def staff_level(n):
    if n <= 6:
        return 'Low'
    elif n <= 10:
        return 'Medium'
    else:
        return 'High'

df['staff_group'] = df['kitchen_staff'].apply(staff_level)

# Boxplot
sns.boxplot(x='staff_group', y='past_waste_kg', data=df,
palette='Set1')
plt.title("Food Waste by Kitchen Staff Group")
```

```
plt.show()

# ANOVA
low = df[df['staff_group'] == 'Low']['past_waste_kg']
med = df[df['staff_group'] == 'Medium']['past_waste_kg']
high = df[df['staff_group'] == 'High']['past_waste_kg']

anova = f_oneway(low, med, high)
print("ANOVA p-value:", anova.pvalue)
```

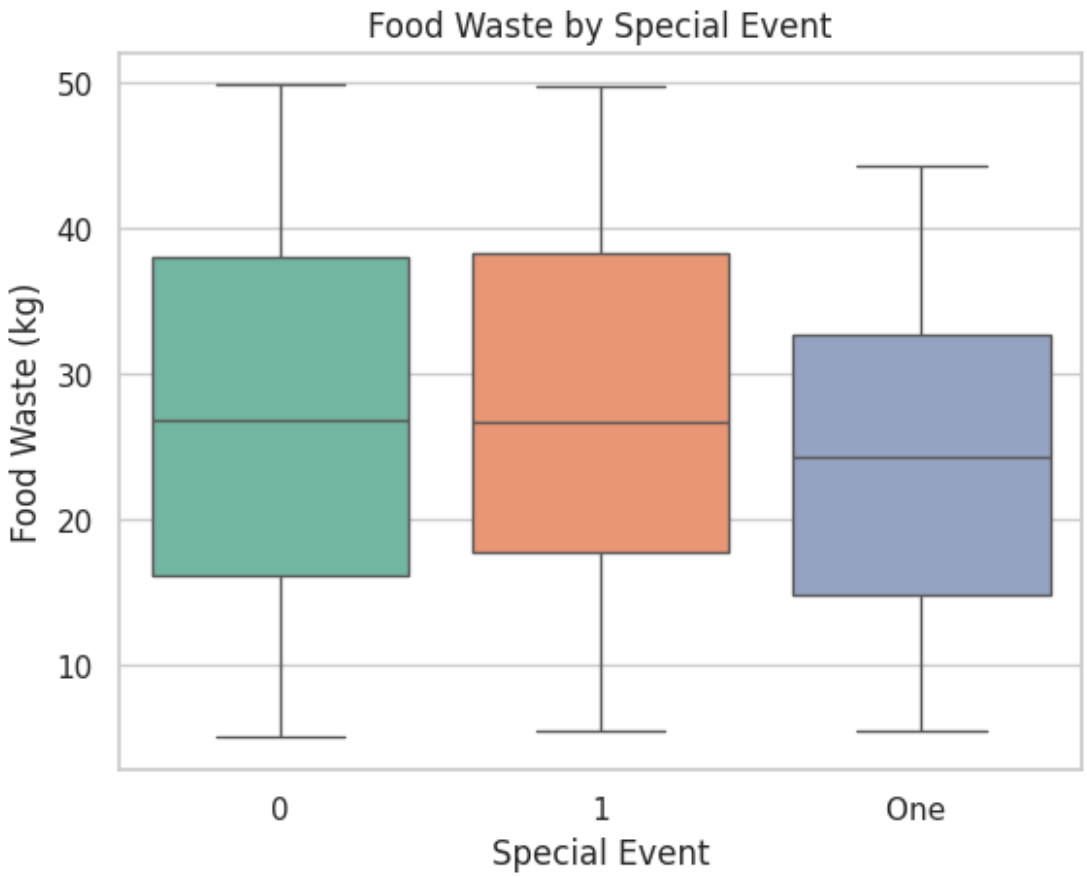
```
<ipython-input-99-ef0daa3fda1d>:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='staff_group', y='past_waste_kg', data=df,
palette='Set1')
```




```
ANOVA p-value: 0.005195810602112262
```

```
from scipy.stats import ttest_ind

# Boxplot
sns.boxplot(x='special_event', y='past_waste_kg', data=df,
palette='Set2')
plt.title("Food Waste by Special Event")
plt.xlabel("Special Event")
plt.ylabel("Food Waste (kg)")
plt.show()

# T-test
event = df[df['special_event'] == 1]['past_waste_kg']
no_event = df[df['special_event'] == 0]['past_waste_kg']
t_stat = ttest_ind(event, no_event, equal_var=False)
print("T-test p-value:", t_stat.pvalue)
```

```
<ipython-input-101-10efd923e92c>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='special_event', y='past_waste_kg', data=df,
palette='Set2')
```




```
T-test p-value: nan

/usr/local/lib/python3.11/dist-packages/scipy/_lib/deprecation.py:234:
SmallSampleWarning: One or more sample arguments is too small; all
returned values will be NaN. See documentation for sample size
requirements.
  return f(*args, **kwargs)
```

1. Key Insights and Recommendations

```
# Key Insights and Recommendations
print("Key Insights and Recommendations")

# Staffing Optimization
if anova.pvalue < 0.05:
    print("Staff level significantly affects food waste")
    print("Recommendation: Use 'Medium' staffing to balance workload
and minimize waste")
else:
    print("No significant relationship between staff level and food
waste")

# Environmental Factors
```

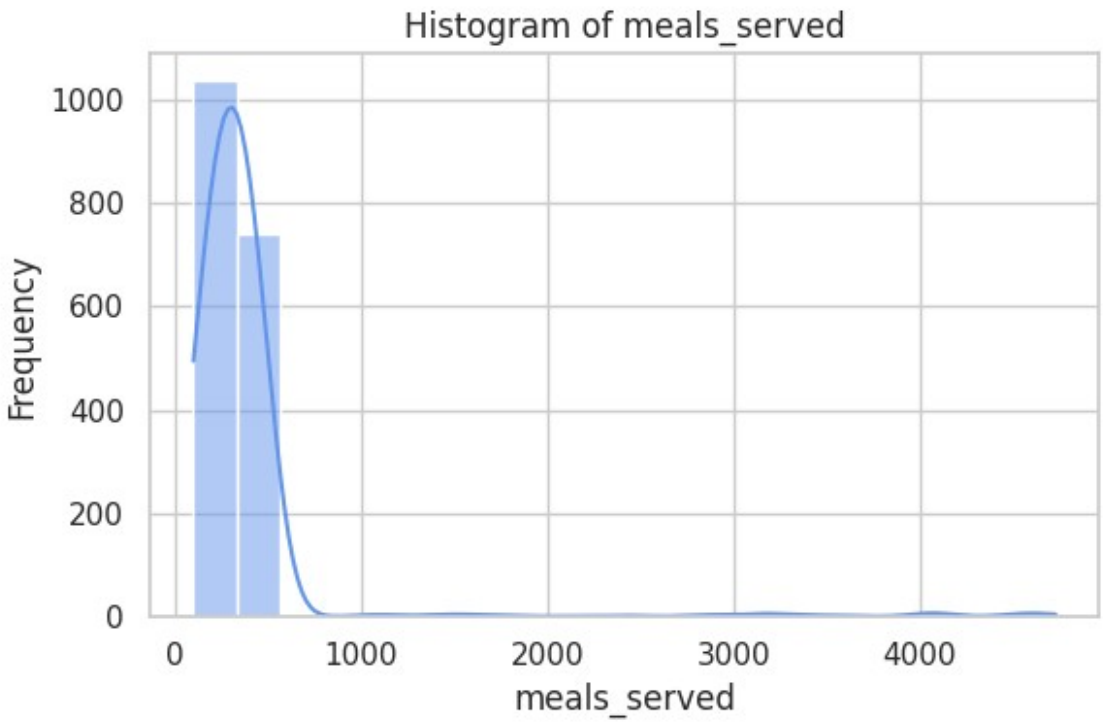
```
corr_temp = df['temperature_C'].corr(df['past_waste_kg'])
corr_humidity = df['humidity_percent'].corr(df['past_waste_kg'])

if abs(corr_temp) > 0.3:
    print(f"Temperature moderately correlates with food waste (r =
{corr_temp:.2f})")
    print("Recommendation: Adjust prep methods and storage on extreme
temperature days")
else:
    print("Weak or no correlation between temperature and food waste")

if abs(corr_humidity) > 0.3:
    print(f"Humidity moderately correlates with food waste (r =
{corr_humidity:.2f})")
    print("Recommendation: Improve storage and adjust prep on humid
days")
else:
    print("Weak or no correlation between humidity and food waste")

# Event Management
if t_stat.pvalue < 0.05:
    print("Food waste is significantly higher during special events")
    print("Recommendation: Improve planning (e.g., accurate guest
counts, donations)")
else:
    print("No significant difference in food waste on special event
days")
```

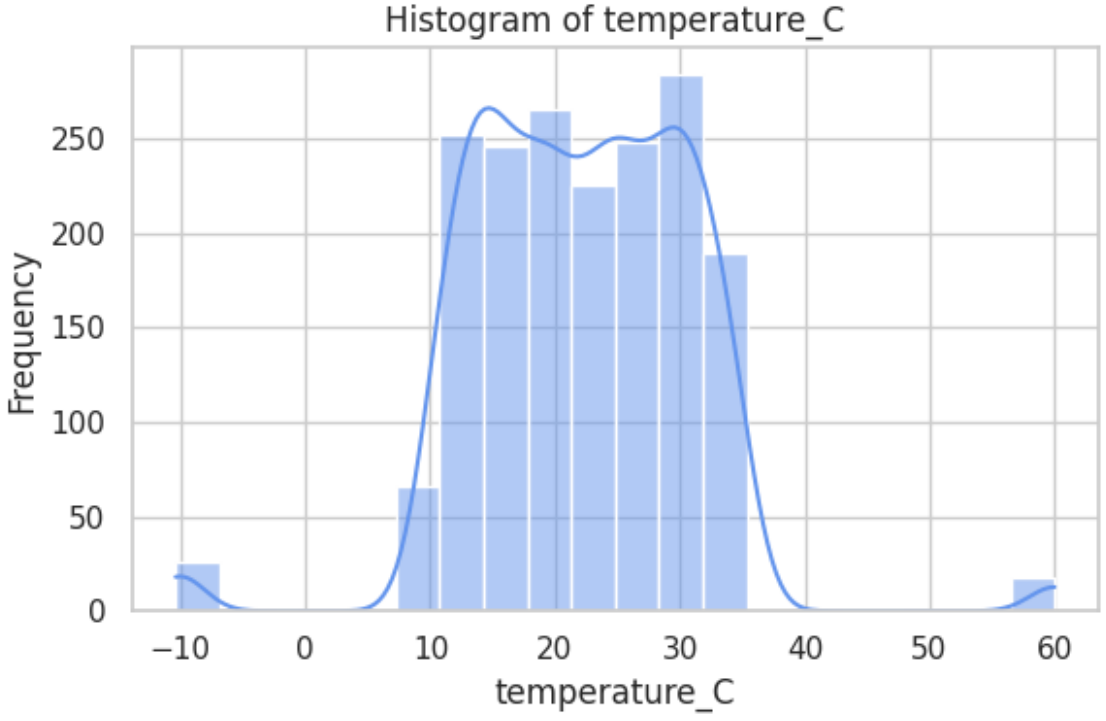
```
Key Insights and Recommendations
Staff level significantly affects food waste
Recommendation: Use 'Medium' staffing to balance workload and minimize
waste
Weak or no correlation between temperature and food waste
Weak or no correlation between humidity and food waste
No significant difference in food waste on special event days
```

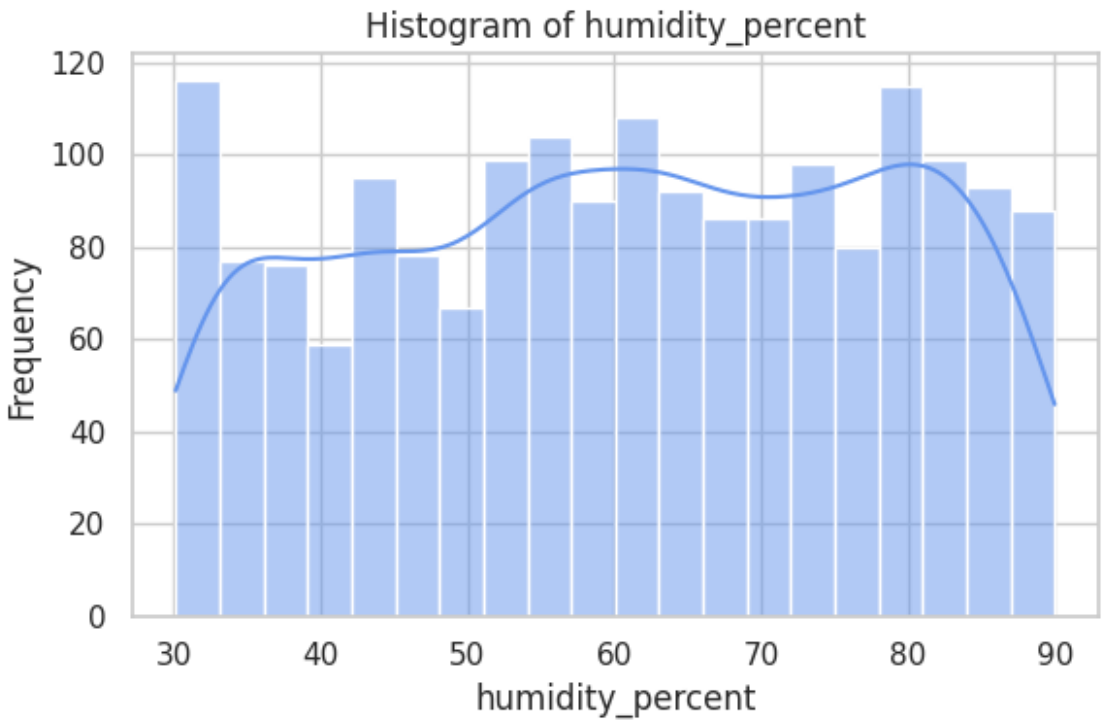
1. Data Visualization and Reporting

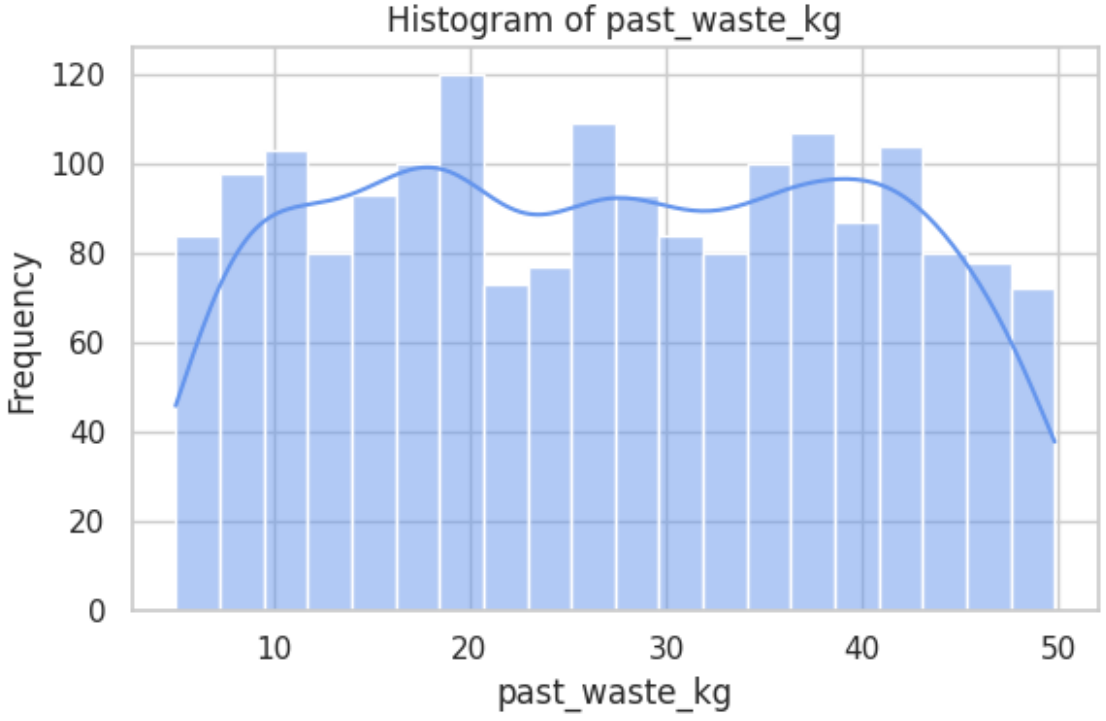
```
# Histograms
numeric_cols = ['meals_served', 'temperature_C', 'humidity_percent',
'past_waste_kg']
for col in numeric_cols:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[col], bins=20, kde=True, color='cornflowerblue')
    plt.title(f"Histogram of {col}")
    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.tight_layout()
    plt.show()
```

```
# Box plot
for col in numeric_cols:
    plt.figure(figsize=(6, 4))
    sns.boxplot(y=df[col], color='salmon')
    plt.title(f"Box Plot of {col}")
    plt.tight_layout()
    plt.show()

# Correlation Heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(df[numeric_cols].corr(), annot=True, cmap='coolwarm',
fmt=".2f")
plt.title("Correlation Heatmap")
plt.tight_layout()
plt.show()

#  Bar Plots: Food Waste by Waste Category
plt.figure(figsize=(8, 5))
sns.barplot(x='waste_category', y='past_waste_kg', data=df,
estimator='mean', ci=None, palette='Set2')
plt.title("Average Food Waste by Waste Category")
plt.xlabel("Waste Category")
plt.ylabel("Average Food Waste (kg)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

#  Bar Plot: Food Waste by Staff Experience
plt.figure(figsize=(8, 5))
sns.barplot(x='staff_experience', y='past_waste_kg', data=df,
estimator='mean', ci=None, palette='Set3')
plt.title("Average Food Waste by Staff Experience")
plt.xlabel("Staff Experience")
plt.ylabel("Average Food Waste (kg)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

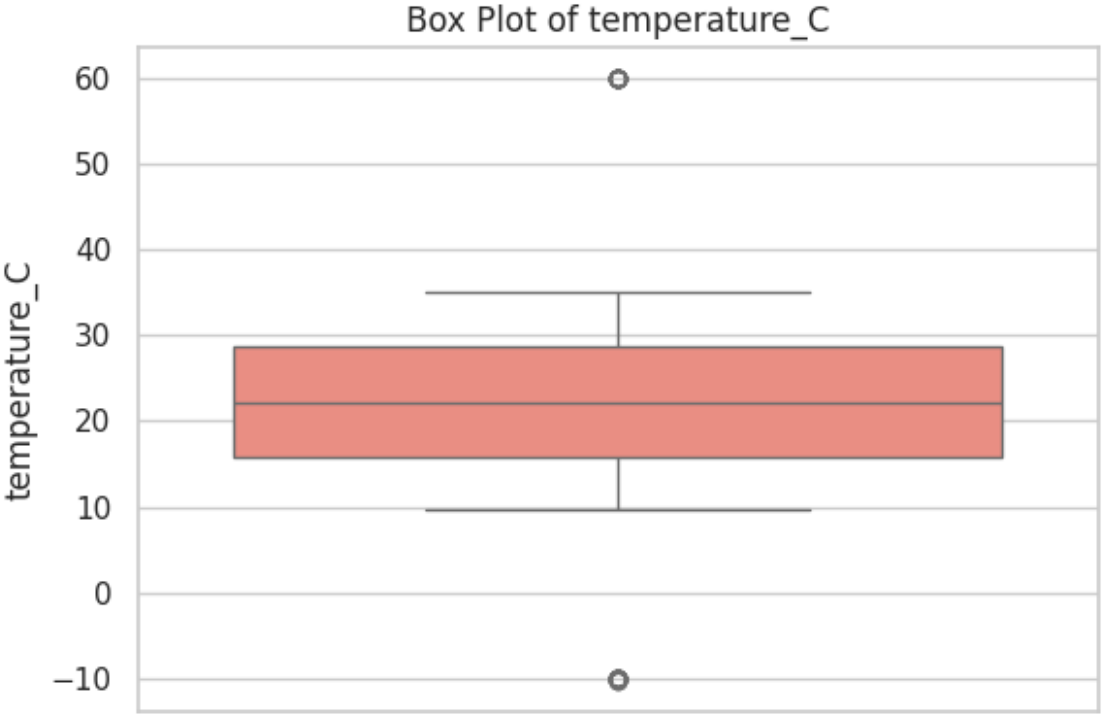

Histogram of meals_served
Frequency
meals_served
0
200
400
600
800
1000
0
1000
2000
3000
4000

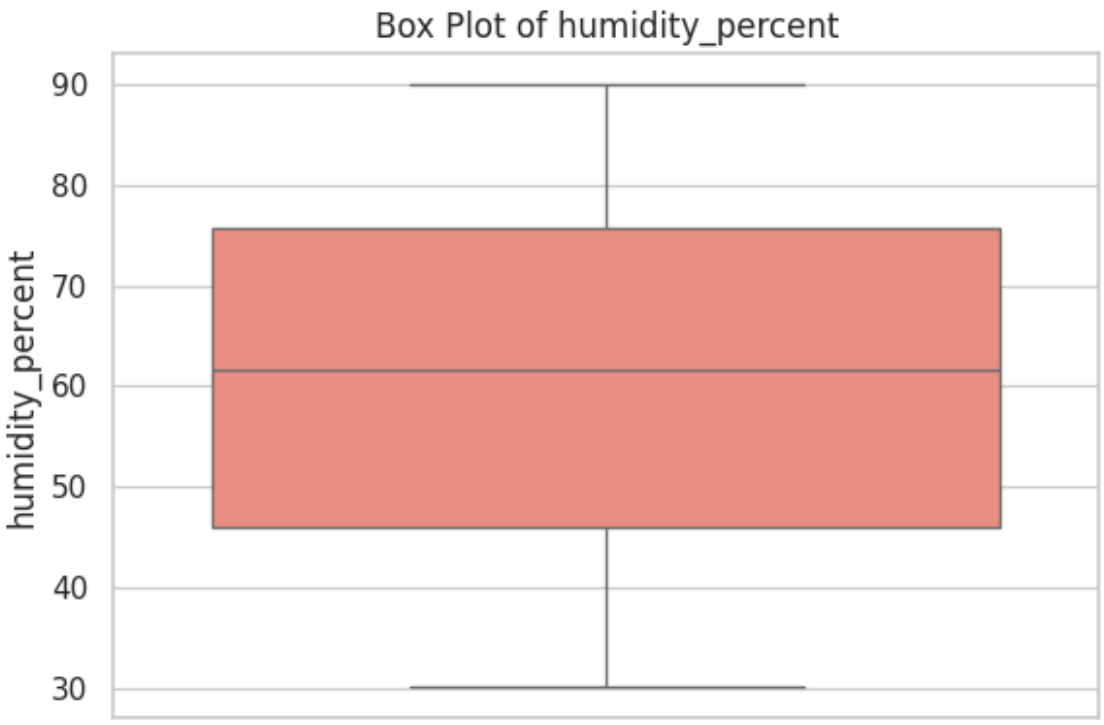

Histogram of temperature_C
Frequency
temperature_C
0
50
100
150
200
250
−10
0
10
20
30
40
50
60

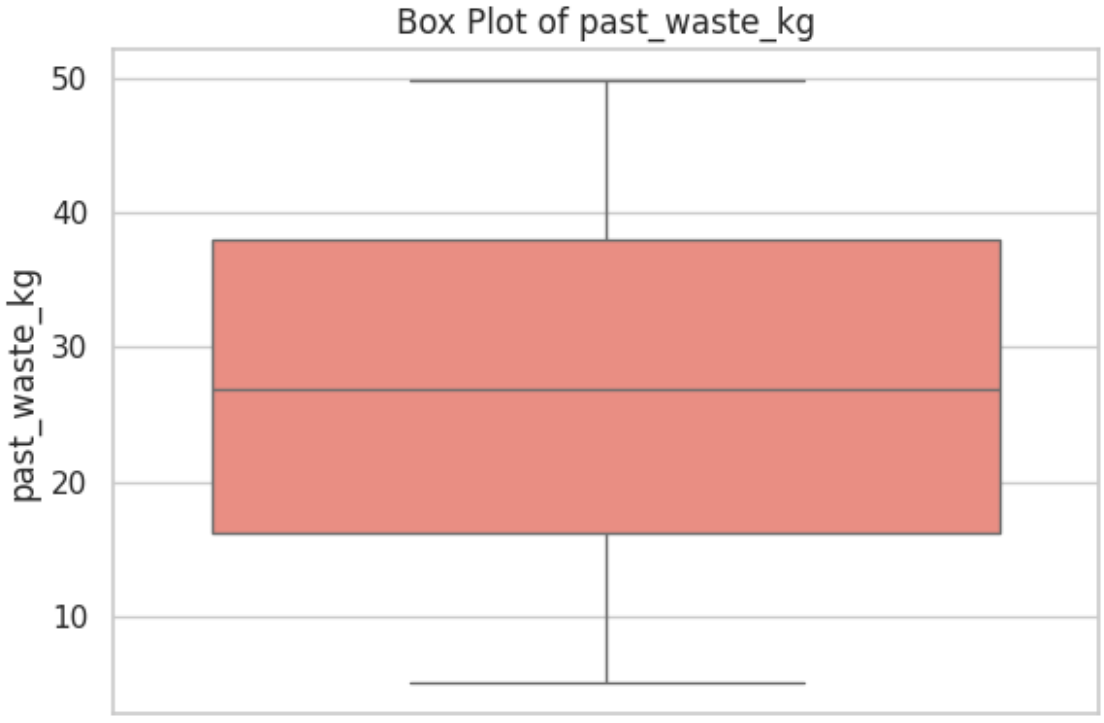

Histogram of humidity_percent
Frequency
0
20
40
60
80
100
120
30
40
50
60
70
80
90
humidity_percent

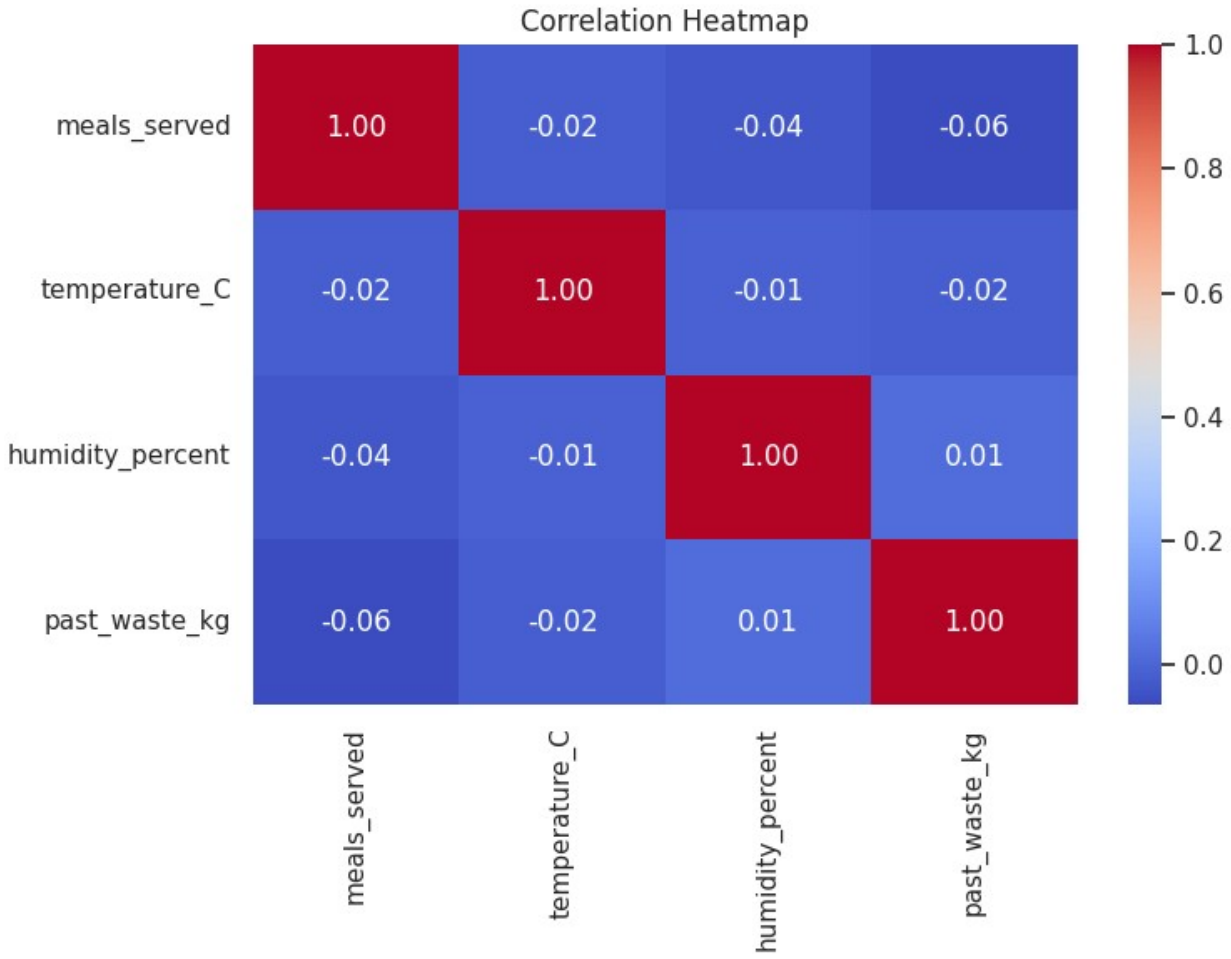

Histogram of past_waste_kg
Frequency
0
20
40
60
80
100
120
10
20
30
40
50
past_waste_kg


Box Plot of meals_served
meals_served
4000
3000
2000
1000
0


Box Plot of temperature_C
temperature_C
60
50
40
30
20
10
0
−10


Box Plot of humidity_percent
humidity_percent
90
80
70
60
50
40
30


Box Plot of past_waste_kg
past_waste_kg
50
40
30
20
10




```
<ipython-input-115-afbd3520bca4>:29: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same
effect.

  sns.barplot(x='waste_category', y='past_waste_kg', data=df,
estimator='mean', ci=None, palette='Set2')
<ipython-input-115-afbd3520bca4>:29: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='waste_category', y='past_waste_kg', data=df,
estimator='mean', ci=None, palette='Set2')
```


```
<ipython-input-115-afbd3520bca4>:39: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same
effect.

  sns.barplot(x='staff_experience', y='past_waste_kg', data=df,
estimator='mean', ci=None, palette='Set3')
<ipython-input-115-afbd3520bca4>:39: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='staff_experience', y='past_waste_kg', data=df,
estimator='mean', ci=None, palette='Set3')
```

What This Covers: Histograms & Box Plots: For distribution and outliers in meals served, temperature, humidity, and waste.

Heatmap: Shows correlations between numeric features (e.g., temperature vs. food waste).

Bar Plots: Compare average food waste across categories like waste_category and staff_experience.