

## Introduction to Python Project : FoodHub Data Analysis

### Problem Statement and Objectives

FoodHub, a food aggregator company, wants to analyze its order data from New York City to improve its online food delivery service. The company needs insights into customer behavior and restaurant performance to enhance the customer experience and optimize its operations.

### Importing the Required Libraries

```
# Importing the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

### Understanding the structure of the Data

```
!pip install gdown
```

```
import pandas as pd
```

```
# Your Google Drive sharing link
file_url = 'https://drive.google.com/file/d/1DoDtX5t8r07GCNCKFMf6s_6jIviuWlB-/view?usp=shari
```

```
# Extract the file ID from the sharing link
file_id = file_url.split('/')[-2]
```

```
# Download the file to your Colab environment
!gdown --id $file_id
```

```
# Assuming the downloaded file is named 'your_file.csv'
file_path = 'foodhub_order.csv'
```

```
# Load the dataset
df = pd.read_csv(file_path)
```



```
Requirement already satisfied: gdown in /usr/local/lib/python3.10/dist-packages (5.1.0)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.10/dist-package
```

```
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from gdc
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.10/dist-packages (
/usr/local/lib/python3.10/dist-packages/gdown/__main__.py:132: FutureWarning: Option `--
warnings.warn(
Downloading...
From: https://drive.google.com/uc?id=1DoDtX5t8rO7GCNCKFMf6s\_6jIviuWLB-
To: /content/foodhub_order.csv
100% 124k/124k [00:00<00:00, 83.9MB/s]
```

```
# Display the first few rows of the dataframe
print(df.head())
```

```
➡ order_id  customer_id  restaurant_name  cuisine_type  \
0    1477147      337525      Hangawi      Korean
1    1477685      358141  Blue Ribbon Sushi Izakaya  Japanese
2    1477070      66393    Cafe Habana      Mexican
3    1477334      106968  Blue Ribbon Fried Chicken  American
4    1478249      76942    Dirty Bird to Go      American

cost_of_the_order  day_of_the_week  rating  food_preparation_time  \
0             30.75      Weekend  Not given             25
1             12.08      Weekend  Not given             25
2             12.23      Weekday         5             23
3             29.20      Weekend         3             25
4             11.59      Weekday         4             25

delivery_time
0             20
1             23
2             28
3             15
4             24
```

Question 1: How many rows and columns are present in the data?[link text](#)

```
# shape of the DataFrame
shape = df.shape
print(f"The dataset contains {shape[0]} rows and {shape[1]} columns.")
# Print the shape of the DataFrame directly
#print("The shape of the dataset is:", df.shape)
```

```
➡ The dataset contains 1898 rows and 9 columns.
```

Observations: Rows 1898 Columns 9

Question 2: What are the datatypes of the different columns in the dataset?

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1898 entries, 0 to 1897
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   order_id              1898 non-null   int64
1   customer_id           1898 non-null   int64
2   restaurant_name       1898 non-null   object
3   cuisine_type          1898 non-null   object
4   cost_of_the_order     1898 non-null   float64
5   day_of_the_week       1898 non-null   object
6   rating                1898 non-null   object
7   food_preparation_time 1898 non-null   int64
8   delivery_time         1898 non-null   int64
dtypes: float64(1), int64(4), object(4)
memory usage: 133.6+ KB
```

observations : Integer values(numerical without decimal values) : order\_id ,Customer\_id,Food Preparation Time,Delivery Time

Float(numerical with decimal values) : Cost of the order ,

Object (Text data) Restaurant Name , Cuisine Type Object Categorical data (days of the week , ratings)

Question:3 Are there any missing values in the data? If yes, treat them using an appropriate method.

```

# Check for missing values
missing_values = df.isnull().sum()

print("Missing values in each column:")
print(missing_values)

# Check if there are any missing values
if missing_values.sum() > 0:
    print("\nThere are missing values in the dataset. Let's handle them.")

    # For numeric columns, we can fill with the mean
    numeric_columns = df.select_dtypes(include=['int64', 'float64']).columns
    for col in numeric_columns:
        if df[col].isnull().sum() > 0:
            df[col].fillna(df[col].mean(), inplace=True)

    # For categorical columns, we can fill with the mode (most frequent value)
    categorical_columns = df.select_dtypes(include=['object']).columns
    for col in categorical_columns:
        if df[col].isnull().sum() > 0:
            df[col].fillna(df[col].mode()[0], inplace=True)

    # Check again to confirm all missing values are handled
    print("\nMissing values after treatment:")
    print(df.isnull().sum())
else:
    print("\nThere are no missing values in the dataset.")

```

```

➡ Missing values in each column:
order_id          0
customer_id       0
restaurant_name   0
cuisine_type      0
cost_of_the_order 0
day_of_the_week   0
rating            0
food_preparation_time 0
delivery_time     0
dtype: int64

```

There are no missing values in the dataset.

Observations: If there are missing values:

--For numeric columns, it fills them with the mean of that column. --For categorical columns, it fills them with the mode (most frequent value) of that column.

Question:4 Check the statistical summary of the data. What is the minimum, average, and maximum time it takes for food to be prepared once an order is placed?

```
# Get descriptive statistics of food_preparation_time
food_prep_time_stats = df['food_preparation_time'].describe()

# Minimum preparation time
min_prep_time = food_prep_time_stats['min']

# Average preparation time
avg_prep_time = food_prep_time_stats['mean']

# Maximum preparation time
max_prep_time = food_prep_time_stats['max']

print(f"Minimum time to prepare food: {min_prep_time:.2f} minutes")
print(f"Average time to prepare food: {avg_prep_time:.2f} minutes")
print(f"Maximum time to prepare food: {max_prep_time:.2f} minutes")
```

```
➞ Minimum time to prepare food: 20.00 minutes
Average time to prepare food: 27.37 minutes
Maximum time to prepare food: 35.00 minutes
```

Observations: The average preparation time of 27.37 minutes suggests there's room for optimization. While some orders might be prepared quickly, the gap between the minimum (20 minutes) and maximum (35 minutes) indicates potential bottlenecks in the food preparation process.

Question 5: How many orders are not rated?

```
# Check for missing values in the 'rating' column
if df['rating'].isnull().values.any():
    # Impute missing values with 'Not given'
    df['rating'].fillna('Not given', inplace=True)

# Count the number of orders with 'Not given' rating
number_of_not_rated = df[df['rating'] == 'Not given'].shape[0]

# Print result
print(f"Number of orders with 'Not Given' rating: {number_of_not_rated}")
```

```
➞ Number of orders with 'Not Given' rating: 736
```

Observations: Low Customer Engagement:

A high number of "Not Given" ratings could indicate low customer engagement with the rating system

Question 6: Explore all the variables and provide observations on their distributions. (Generally, histograms, boxplots, countplots, etc. are used for univariate exploration.)

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Numerical variables
numerical_vars = ['cost_of_the_order', 'rating', 'food_preparation_time', 'delivery_time']

# Categorical variables
categorical_vars = ['customer_id', 'restaurant_name', 'cuisine_type', 'day_of_the_week']

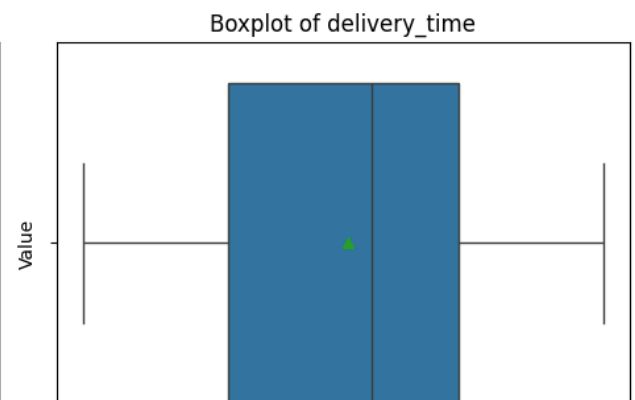
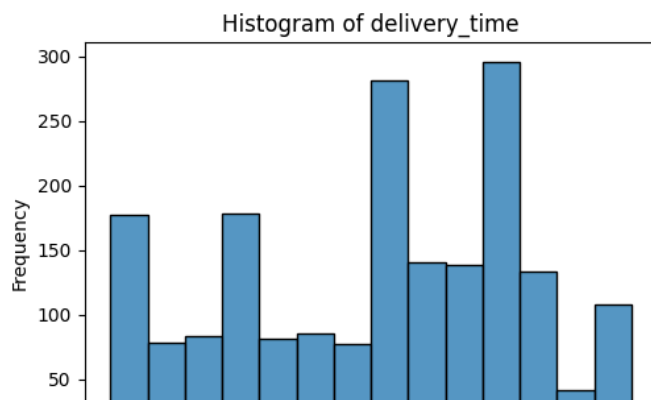
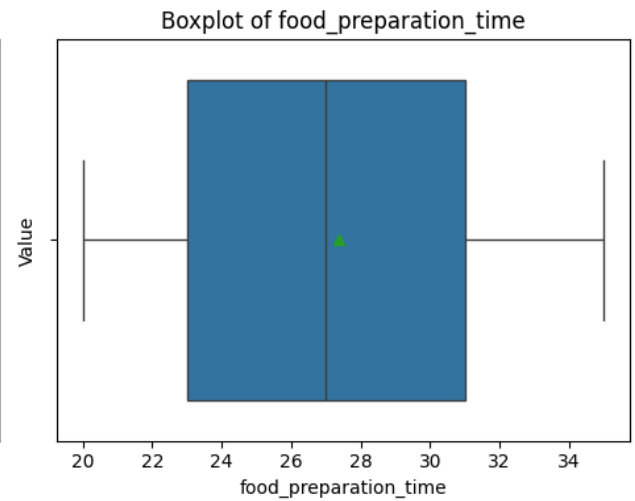
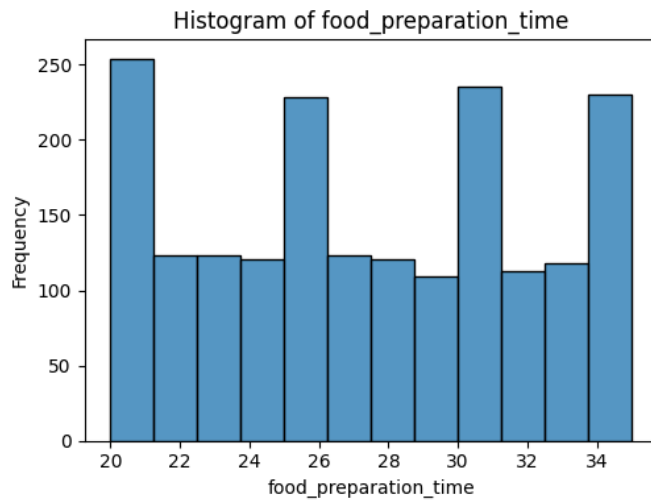
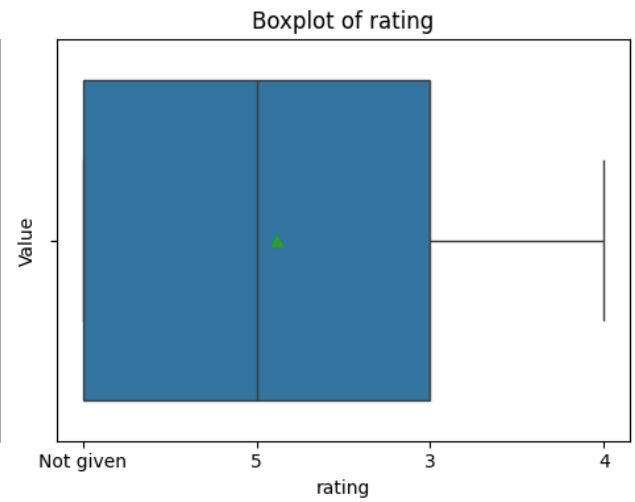
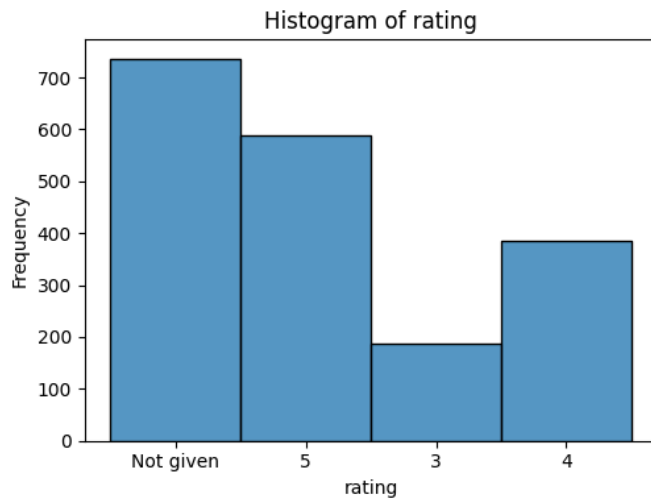
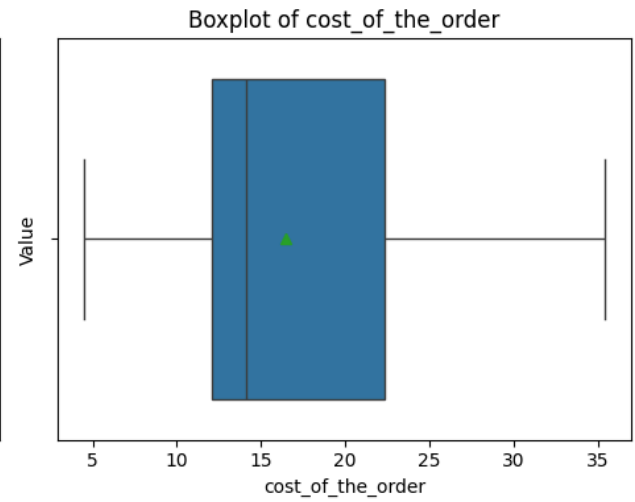
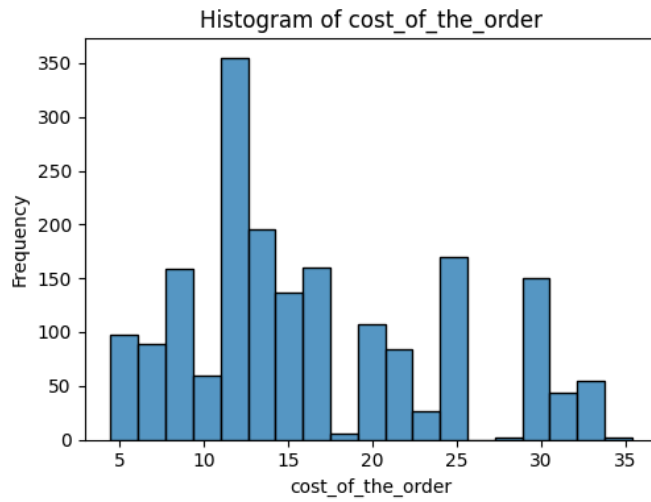
def plot_distribution(var):
    """Plots histogram and boxplot for a variable."""
    plt.figure(figsize=(10, 4))

    # Histogram
    plt.subplot(1, 2, 1)
    sns.histplot(data=df, x=var) # Pass df to sns.histplot
    plt.xlabel(var)
    plt.ylabel('Frequency')
    plt.title(f'Histogram of {var}')

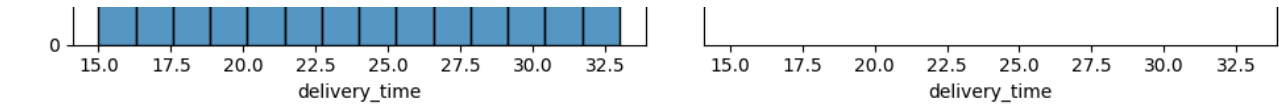
    # Boxplot
    plt.subplot(1, 2, 2)
    sns.boxplot(data=df, x=var, showmeans=True) # Pass df to sns.boxplot
    plt.xlabel(var)
    plt.ylabel('Value')
    plt.title(f'Boxplot of {var}')
    plt.tight_layout()
    plt.show()

# Create plots for numerical variables
for var in numerical_vars:
    plot_distribution(var)

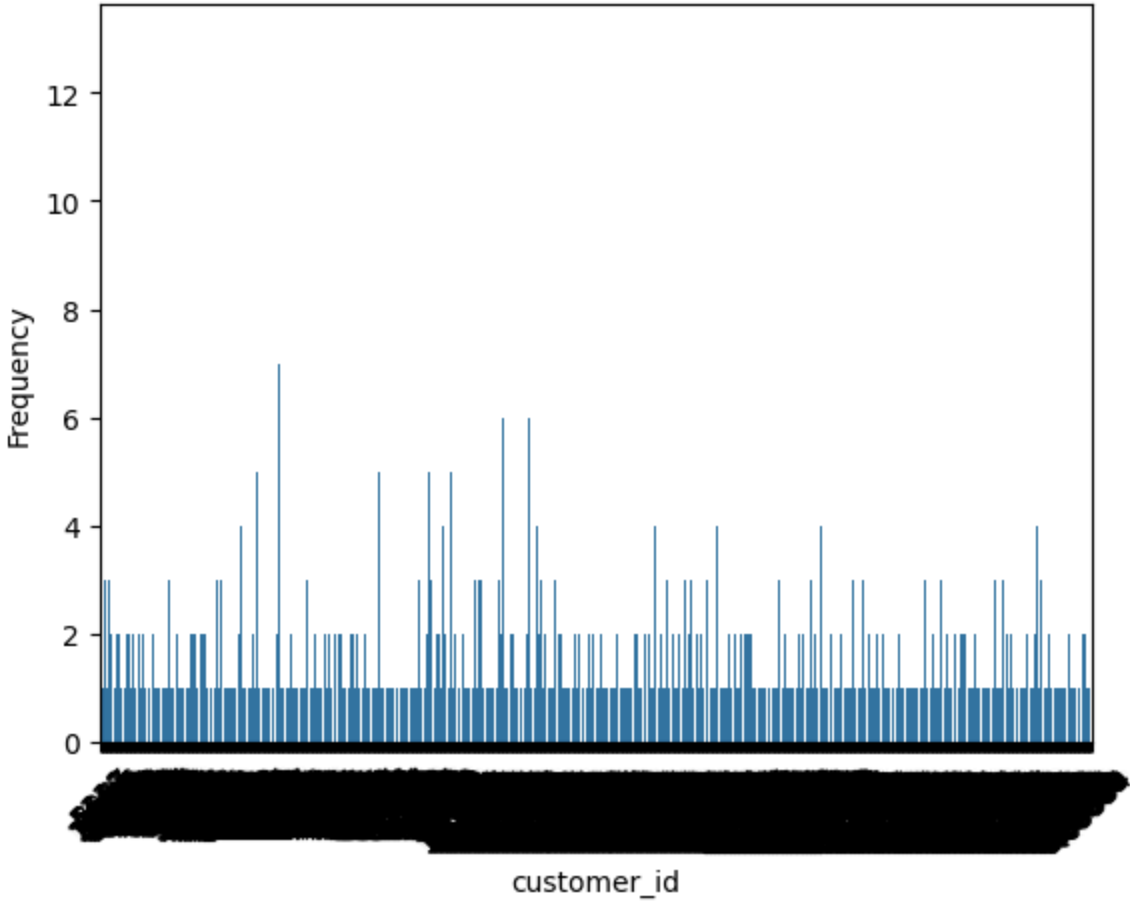
# Countplots for categorical variables
for var in categorical_vars:
    sns.countplot(data=df, x='customer_id') # Pass df to sns.countplot
    plt.xlabel(var)
    plt.ylabel('Frequency')
    plt.title(f'Distribution of {var}')
    plt.xticks(rotation=45) # Rotate x-axis labels for readability
    plt.show()
```



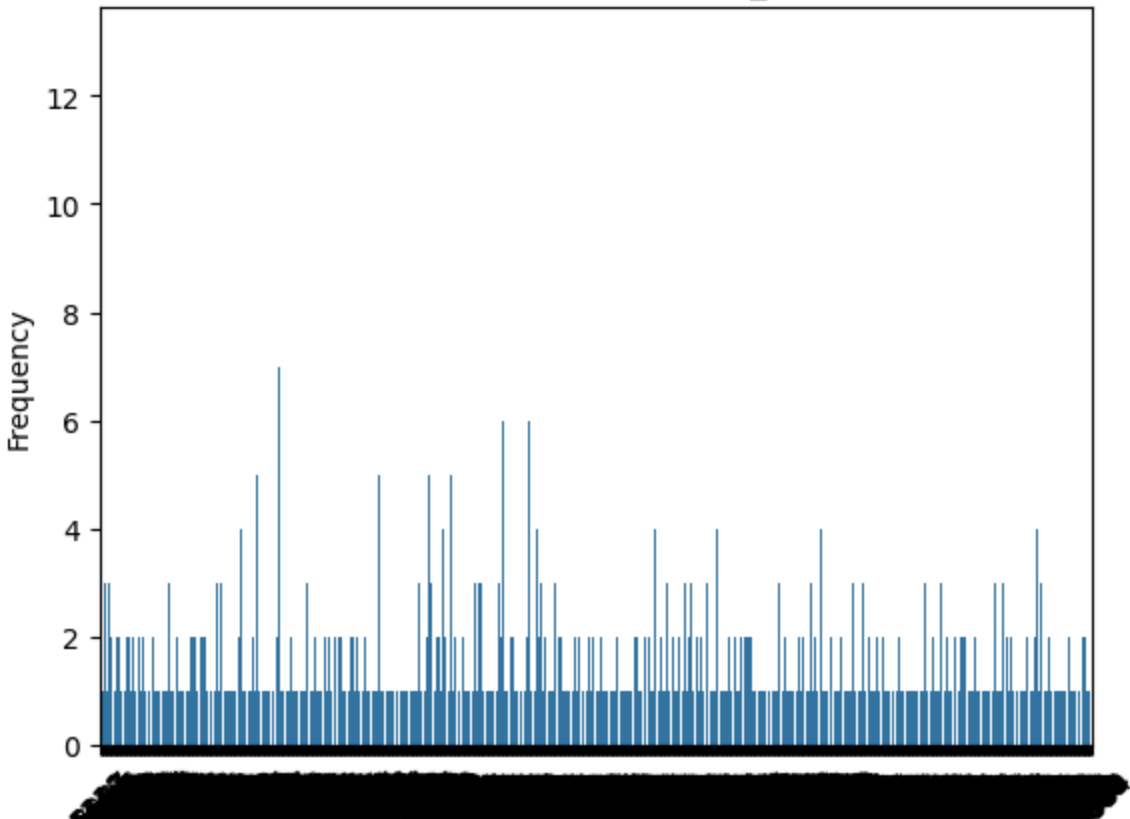




Distribution of customer\_id

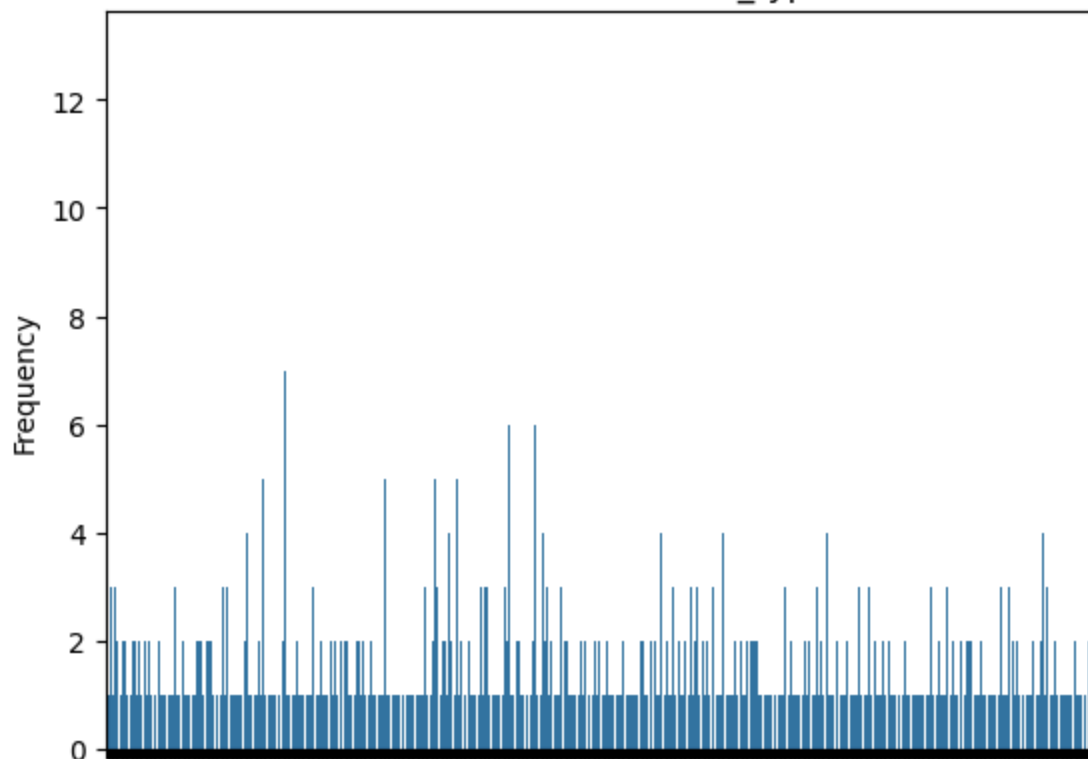


Distribution of restaurant\_name



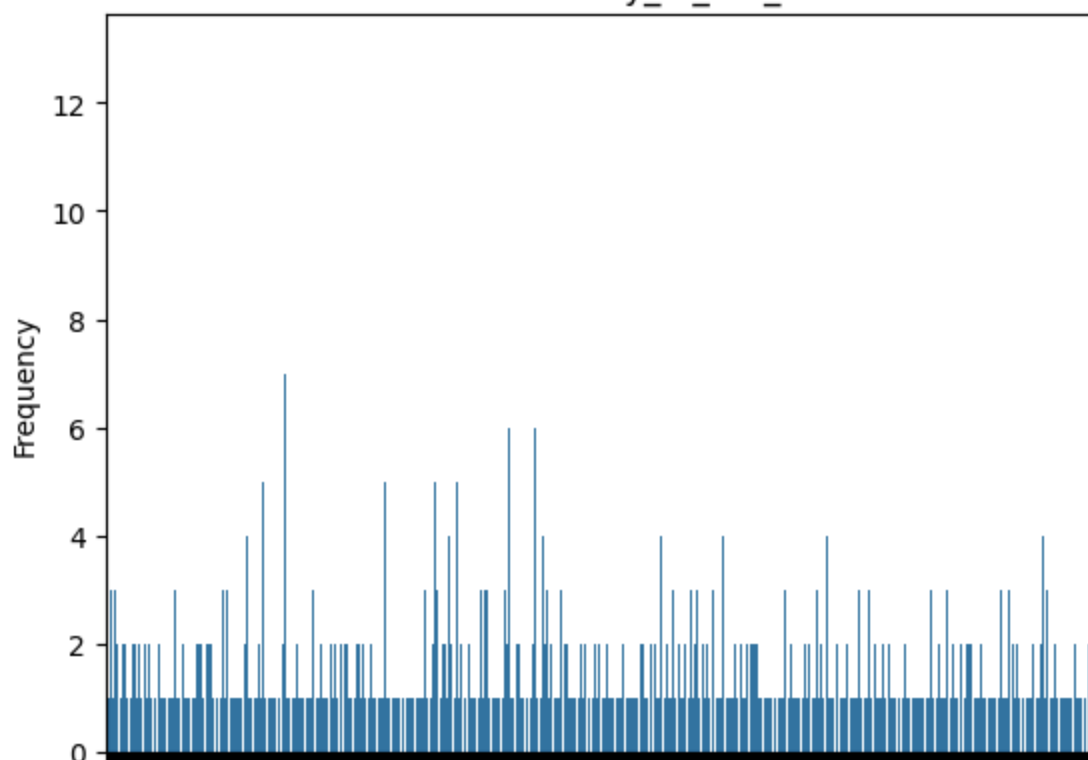
restaurant\_name


Distribution of cuisine\_type



cuisine\_type

Distribution of day\_of\_the\_week



  
`day_of_the_week`

#### Observations:

1. The **rating distribution** shows a positive skew with a peak at 4 stars, indicating mostly satisfied customers with a few exceptionally positive experiences.
2. The boxplot suggests potential outliers for long **food preparation times**, requiring further investigation to identify and support slow restaurants.
3. The boxplot of **delivery times** shows a possible right skew, suggesting there might be some deliveries experiencing delays compared to the typical delivery time
4. The counterplot of **cuisine type** and **days of the week** shows Weekends (Saturday and Sunday) see peak orders, with Italian food being the most popular cuisine overall.
5. The typical **cost of an order** appears to be between 10 and 20. There are a few outliers that appear to be more expensive than \$30.

Question 7: Which are the top 5 restaurants in terms of the number of orders received?

```

restaurant_order_counts = df['restaurant_name'].value_counts()
top_5_restaurants = restaurant_order_counts.head()
print("Top 5 restaurants by number of orders:")
print(top_5_restaurants)

```

```

→ Top 5 restaurants by number of orders:
restaurant_name
Shake Shack                219
The Meatball Shop          132
Blue Ribbon Sushi          119
Blue Ribbon Fried Chicken   96
Parm                       68
Name: count, dtype: int64

```

Observations : Top 5 restaurants by number of orders: Shake Shack, The Meatball Shop, Blue Ribbon Sushi, Blue Ribbon Fried Chicken, Parm

Question 8: Which is the most popular cuisine on weekends?

```

# Filter the data for weekends
weekend_orders = df[df['day_of_the_week'] == 'Weekend']

# Count the occurrences of each cuisine type on weekends
weekend_cuisine_counts = weekend_orders['cuisine_type'].value_counts()

# Get the most popular cuisine
most_popular_cuisine = weekend_cuisine_counts.index[0]
most_popular_cuisine_count = weekend_cuisine_counts.iloc[0]
print("observations")
print(f"The most popular cuisine on weekends is {most_popular_cuisine} with {most_popular_cu

```

```

→ observations
The most popular cuisine on weekends is American with 415 orders.

```

Question :9 What percentage of the orders cost more than 20 dollars?

```
# Calculate the number of orders costing more than $20
orders_over_20 = df[df['cost_of_the_order'] > 20]
num_orders_over_20 = len(orders_over_20)

# Calculate the total number of orders
total_orders = len(df)

# Calculate the percentage
percentage_over_20 = (num_orders_over_20 / total_orders) * 100

print(f"Number of orders costing more than $20: {num_orders_over_20}")
print(f"Total number of orders: {total_orders}")
print(f"Percentage of orders costing more than $20: {percentage_over_20:.2f}%")
```

```
➞ Number of orders costing more than $20: 555
   Total number of orders: 1898
   Percentage of orders costing more than $20: 29.24%
```

Observation: Orders costing more than \$20: 29.24% of total orders (555 out of 1898).

Question 10: What is the mean order delivery time?

```
delivery_time_stats = df['delivery_time'].describe()

print("\nDelivery Time Statistics (in minutes):")
print(delivery_time_stats)
```

```
➞
Delivery Time Statistics (in minutes):
count    1898.000000
mean      24.161749
std        4.972637
min       15.000000
25%       20.000000
50%       25.000000
75%       28.000000
max       33.000000
Name: delivery_time, dtype: float64
```

Observation Mean order delivery time is ~ 24 mins

Question 11 The company has decided to give 20% discount vouchers to the top 3 most frequent customers. Find the IDs of these customers and the number of orders they placed

```
# Count the number of orders for each customer
customer_order_counts = df['customer_id'].value_counts()

# Get the top 3 customers
top_3_customers = customer_order_counts.head(3)

print("Top 3 most frequent customers:")
for customer_id, order_count in top_3_customers.items():
    print(f"Customer ID: {customer_id}, Number of orders: {order_count}")
```

⇒ Top 3 most frequent customers:  
Customer ID: 52832, Number of orders: 13  
Customer ID: 47440, Number of orders: 10  
Customer ID: 83287, Number of orders: 9

Question 12: Perform a multivariate analysis to explore relationships between the important variables in the dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming df is your DataFrame

# 1. Correlation matrix for numerical variables
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
correlation_matrix = df[numerical_cols].corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1, center=0)
plt.title('Correlation Matrix of Numerical Variables')
plt.tight_layout()
plt.show()

# 2. Relationship between cost and rating
plt.figure(figsize=(10, 6))
sns.scatterplot(x='cost_of_the_order', y='rating', data=df)
plt.title('Relationship between Order Cost and Rating')
plt.xlabel('Cost of the Order')
plt.ylabel('Rating')
plt.show()

# 3. Relationship between delivery time and rating
plt.figure(figsize=(10, 6))
sns.scatterplot(x='delivery_time', y='rating', data=df)
plt.title('Relationship between Delivery Time and Rating')
plt.xlabel('Delivery Time (minutes)')
plt.ylabel('Rating')
plt.show()

# 4. Average cost by cuisine type
plt.figure(figsize=(12, 6))
sns.barplot(x='cuisine_type', y='cost_of_the_order', data=df)
plt.title('Average Cost by Cuisine Type')
plt.xlabel('Cuisine Type')
plt.ylabel('Average Cost')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# 5. Average rating by day of the week
plt.figure(figsize=(10, 6))
sns.barplot(x='day_of_the_week', y='rating', data=df)
plt.title('Average Rating by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Average Rating')
plt.show()

# 6. Distribution of delivery times by cuisine type
```

```
plt.figure(figsize=(12, 6))
sns.boxplot(x='cuisine_type', y='delivery_time', data=df)
plt.title('Distribution of Delivery Times by Cuisine Type')
plt.xlabel('Cuisine Type')
plt.ylabel('Delivery Time (minutes)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# 7. Pairplot for numerical variables
sns.pairplot(df[numerical_cols])
plt.tight_layout()
plt.show()

# 8. Average cost by day of the week
plt.figure(figsize=(10, 6))
sns.barplot(x='day_of_the_week', y='cost_of_the_order', data=df)
plt.title('Average Cost by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Average Cost')
plt.show()
```



