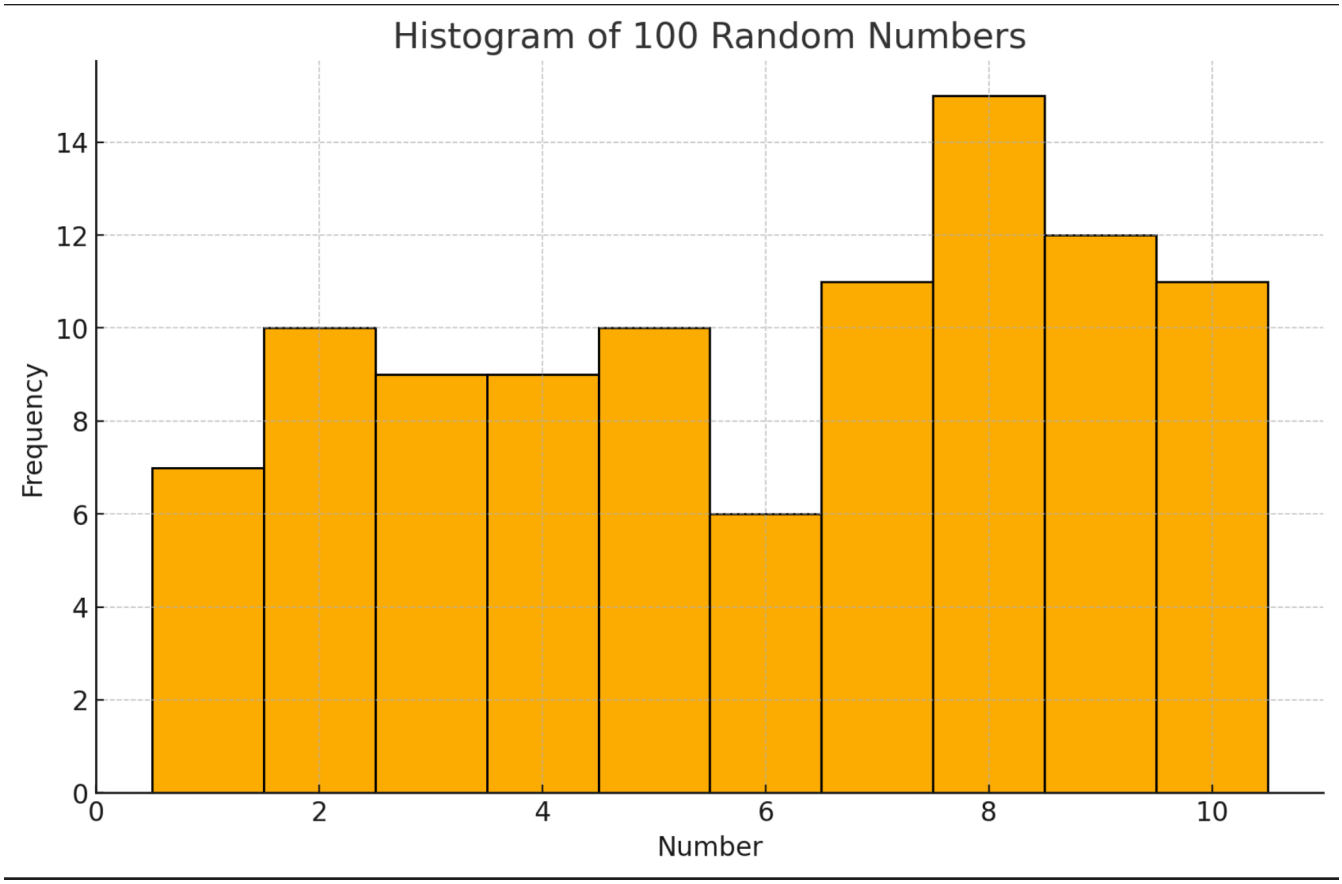


Start coding or generate with AI.

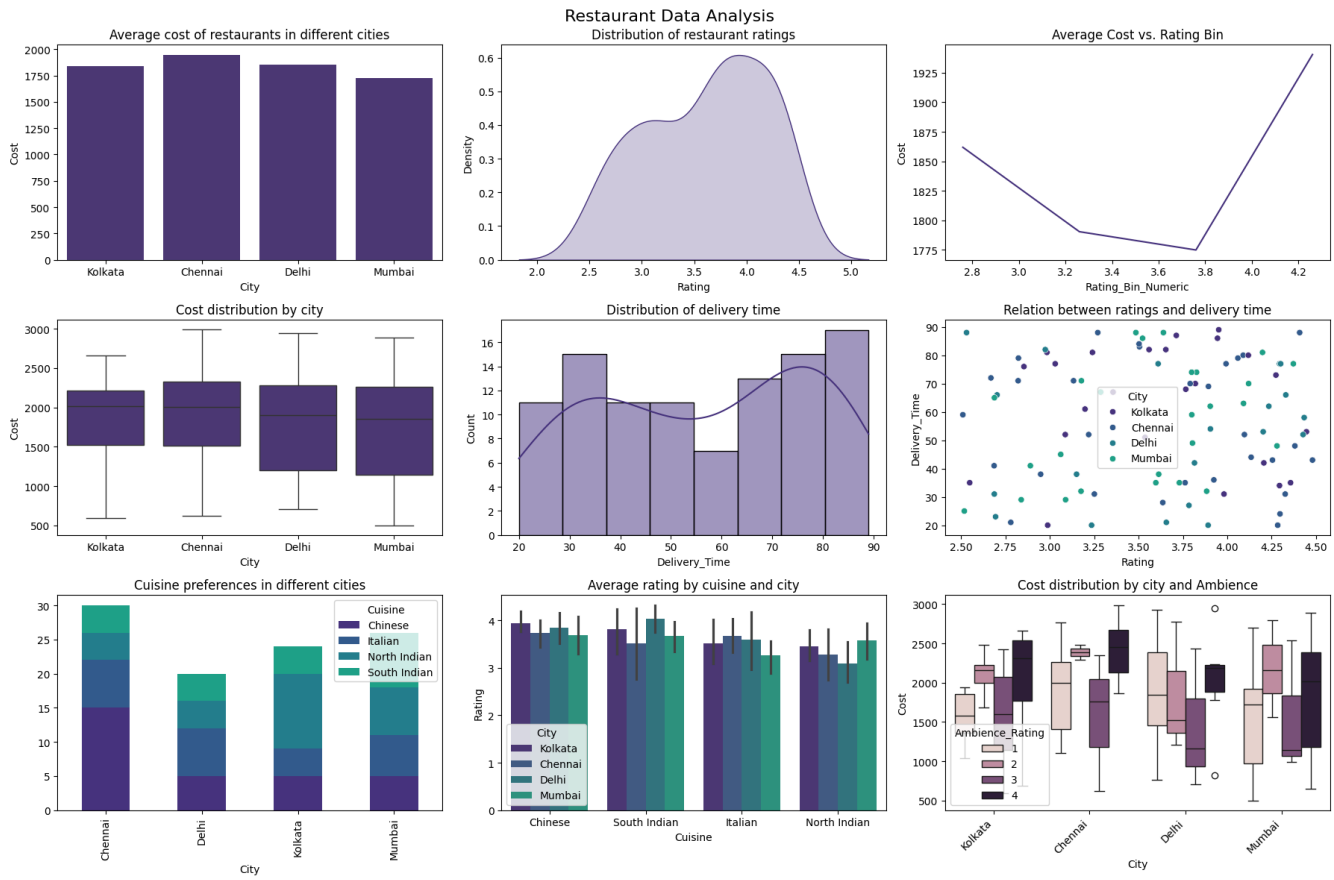
```
# Boilerplate Code - NO NEED TO UNDERSTAND (FOR NOW)
import os
import gdown

file_name = 'restaurants_all.csv'
file_path = os.path.join(os.getcwd(), file_name)

if not os.path.exists(file_path):
    url = "https://drive.google.com/uc?id=1qgMZdOpZ_KJAgqc46WW5pvG7xgkJ2jQ6"
    gdown.download(url, file_path, quiet=False)
    print(f"The file '{file_path}' has been downloaded.")
else:
    print(f"The file '{file_path}' already exists.")

import pandas as pd
df = pd.read_csv('restaurants_all.csv')
top_cities = df['city'].value_counts().index[:5]  # Top 5 cities
top_cuisines = df['cusine'].value_counts().index[:5]  # Top 5 cuisines
df['cusine_list'] = df['cusine'].apply(lambda x: x.split(',') if isinstance(x, str) else []) # str to list
cusine_exploded = df.explode('cusine_list') # explode the list (save each value as different row)
top_cusines = cusine_exploded["cusine_list"].value_counts().index[:10] # Top 10 cusines
sample_df = cusine_exploded[cusine_exploded['city'].isin(top_cities)] # only limit dataset for top-5 cities to make clean pl
sample_df = sample_df[sample_df['cusine_list'].isin(top_cuisines)] # limit dataset to top-10 cuisines
sample_df = sample_df.sample(n=1000, random_state=42)
df2 = sample_df
```

```
Downloading...
From (original): https://drive.google.com/uc?id=1qgMZdOpZ_KJAgqc46WW5pvG7xgkJ2jQ6
From (redirected): https://drive.google.com/uc?id=1qgMZdOpZ_KJAgqc46WW5pvG7xgkJ2jQ6&confirm=t&uuid=1d0e149c-2d0d-4633-b5
To: /content/restaurants_all.csv
100%|██████████| 190M/190M [00:03<00:00, 58.0MB/s]
The file '/content/restaurants_all.csv' has been downloaded.
```

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_palette(sns.color_palette("tab20"))
```

Why Data Visualisation?

Lets see what all you know - Which all plots have you seen before or studied/used in school?



Double-click (or enter) to edit

Can't We Just Look at the Numbers?

Before we start plotting, let's ask:

- If you have a dataset with thousands of values, can you quickly see patterns just by reading the table?

  "Would you prefer looking at a table with 1000 rows or a simple bar chart to compare restaurants?"

💡 Point to make: ➡ Numbers alone are hard to interpret. Instead of scrolling through thousands of records, we use graphs to instantly identify trends & insights.

What Does Visualization Help With?

✅ **Quickly spot trends & patterns** – Instead of reading numbers, we can *see* the distribution.
✅ **Identify outliers & anomalies** – A boxplot immediately highlights extreme values.
✅ **Compare categories effectively** – A bar chart is easier to interpret than a list of counts.
✅ **Communicate insights clearly** – A good plot makes data understandable for everyone.

Example: Restaurant Counts in Cities

Imagine you get a table like this:

City	Number of Restaurants
Delhi NCR	363
Mumbai	228
Bengaluru	160
Pune	154
Hyderabad	95

💡 Can you immediately tell which city has the most restaurants?
📊 **A bar chart makes it obvious in seconds!**

Univariate Data Visualisation

Univariate Analysis: Exploring One Variable at a Time

Step 1: Ask the Right Questions

Before plotting, always check:

✅ What question am I trying to answer?

✅ How many variables are involved?

✅ What is the data type (categorical or numerical)?

Univariate - Categorical Data

Double-click (or enter) to edit

📊 **Categories = Groups or labels (e.g., city, cuisine).**

✅ **Best Plots:**

- **Bar Plot** -- Best for showing category frequencies.
- **Pie Chart** -- Good for proportions, but harder to compare categories.

❌ **Avoid:**

- **Histograms & Line Plots** -- These are for numerical data!

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1000 entries, 65939 to 38136
Data columns (total 18 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   zomato_url         1000 non-null   object
 1   name               1000 non-null   object
 2   city               1000 non-null   object
 3   area               1000 non-null   object
 4   rating             647 non-null    float64
 5   rating_count       634 non-null    float64
 6   telephone          988 non-null    object
 7   cusine             1000 non-null   object
 8   cost_for_two       996 non-null    float64
 9   address            982 non-null    object
 10  timings            993 non-null    object
 11  online_order       1000 non-null   bool
 12  table_reservation  1000 non-null   bool
 13  delivery_only      1000 non-null   bool
 14  famous_food        260 non-null    object
 15  longitude          1000 non-null   float64
 16  latitude           1000 non-null   float64
 17  cusine_list        1000 non-null   object
dtypes: bool(3), float64(5), object(10)
memory usage: 127.9+ KB
```

```
#“How many restaurants are there in each city in our dataset?”
city_counts = df2['city'].value_counts()
print(city_counts)
```

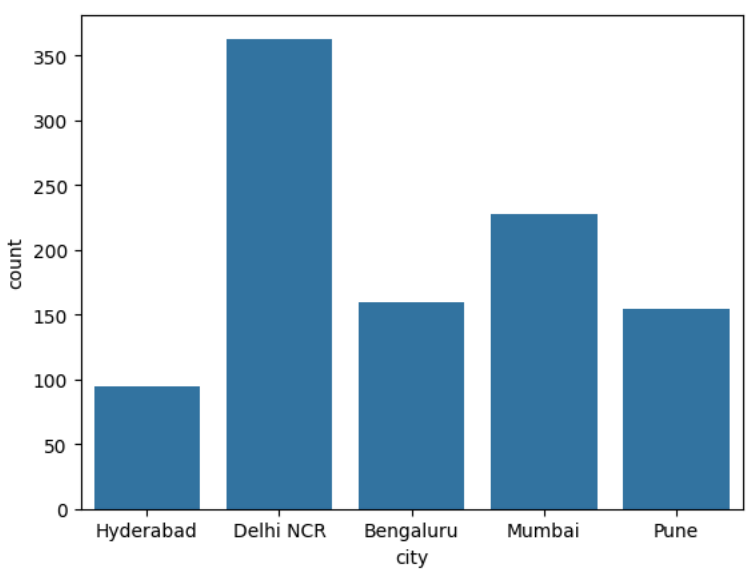
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-f7e08ded72da> in <cell line: 0>()
      1 #“How many restaurants are there in each city in our dataset?”
----> 2 city_counts = df2['city'].value_counts()
      3 print(city_counts)

NameError: name 'df2' is not defined
```

Next steps: Explain error

What kind of plot can we use to visualize this information?

```
Way-1: Barplot/Count plot in seaborn (categories on x-axis, their counts on Y-axis)
Way-2  Piechart (proporation of categories)
```

```
sns.countplot(x='city',data=df2) # calculates counts automatically
plt.show()
```



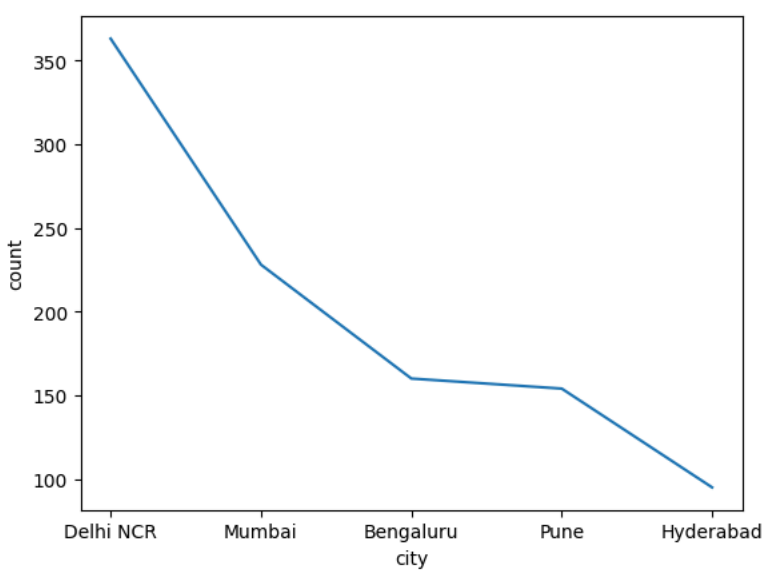
What if instead of actual frequencues, we want to see the proportion of the categories?

```
city_counts.plot.pie(autopct='%1.1f%%') # seaborn doesn't suppport creating a pie-chart, not used in scientific community
plt.title('Proportion of Restaurants by City') # student learns other properties like title
plt.ylabel('') # Remove the ylabel - first show the plot without using this
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-b521befe25be> in <cell line: 0>()
----> 1 city_counts.plot.pie(autopct='%1.1f%%') # seaborn doesn't suppport creating a pie-chart, not used in scientific
community
      2 plt.title('Proportion of Restaurants by City') # student learns other properties like title
      3 plt.ylabel('') # Remove the ylabel - first show the plot without using this
      4 plt.show()

NameError: name 'city_counts' is not defined
```

Next steps: Explain error

```
sns.lineplot(data=city_counts) # bad example of plot - no relation between cities
plt.show()
```



Univariate Numerical Data

📈 **Numbers = Measurable values (e.g., price, ratings).**

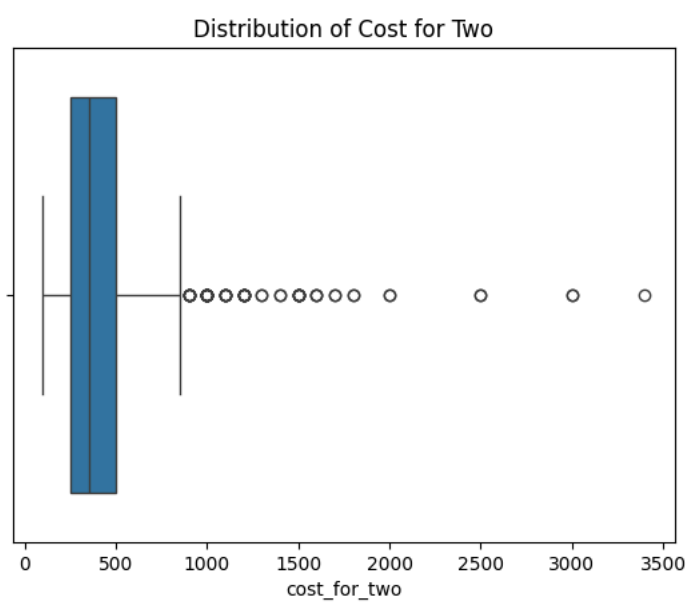
✅ **Best Plots:**

- **Boxplot** -- Highlights median, quartiles, and outliers.
- **Histogram** -- Shows how values are distributed.
- **KDE Plot** -- A smooth version of a histogram.

❌ **Avoid:**

- **Pie Chart & Bar Plot** -- These are not meant for numerical distributions.

```
# Numerical Data - Plotting the boxplot for distribution of `cost_for_two`
sns.boxplot(x='cost_for_two', data=df2)
plt.title('Distribution of Cost for Two')
plt.show()
```

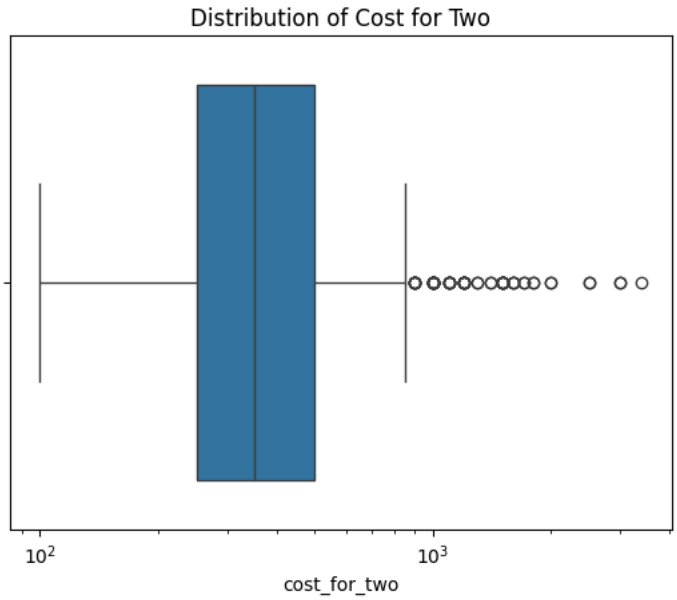



```
# Numerical Data - Plotting the boxplot for distribution of `cost_for_two`  after removing super extreme values
sns.boxplot(x='cost_for_two', data=df2[df2["cost_for_two"] <= 1500])
plt.title('Distribution of Cost for Two')
plt.show()
```

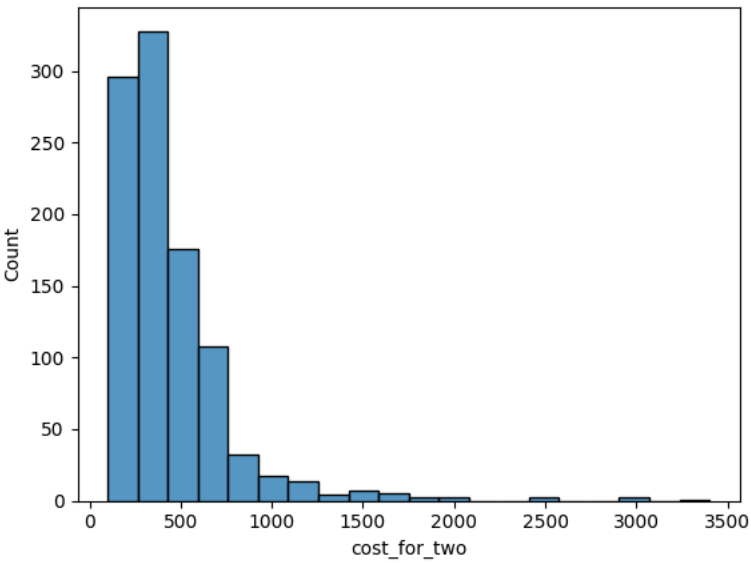
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-060a204d576e> in <cell line: 0>()
      1 # Numerical Data - Plotting the boxplot for distribution of `cost_for_two`  after removing super extreme values
----> 2 sns.boxplot(x='cost_for_two', data=df2[df2["cost_for_two"] <= 1500])
      3 plt.title('Distribution of Cost for Two')
      4 plt.show()

NameError: name 'sns' is not defined
```

Next steps: Explain error

```
# Supplementary - Feature Transformation (convert the data to log scale)
sns.boxplot(x='cost_for_two', data=df2)
plt.title('Distribution of Cost for Two')
plt.xscale('log')
plt.show()
```



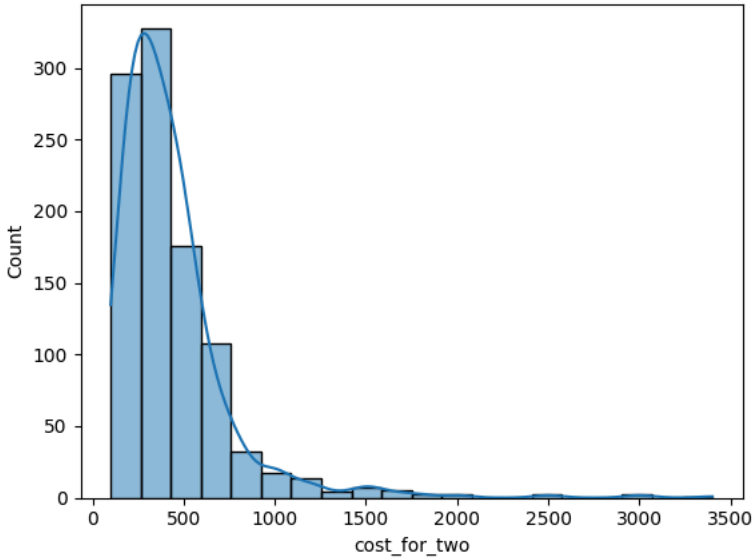

```
# Plotting the histogram for `cost_for_two`
sns.histplot(df2['cost_for_two'], bins = 20)
plt.show()
```



"Are most restaurants affordable, or do high-end places dominate?"

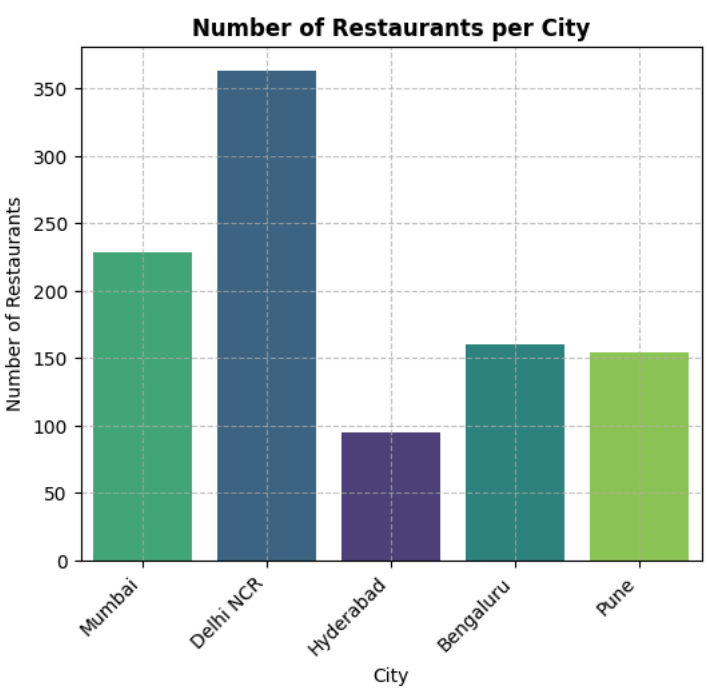
📝 Hint: Instead of looking at averages, let's see how restaurant costs are spread.

```
# KDE - curve estimation of histogram, will discuss in later module
sns.histplot(df2['cost_for_two'], kde = True, bins = 20)
plt.show()
```



Try it yourself - Make your plots more readiable and pleasing :)

```
# Create a barplot of the number of restaurants per city
plt.figure(figsize=(6, 5))  # Set the figure size (width=12, height=6) in inches

# Plot the barplot
sns.countplot(x='city', data=sample_df,
              order= ['Mumbai', 'Delhi NCR',  'Hyderabad', 'Bengaluru', 'Pune',],# can display bars in a specific order
              palette="viridis", hue="city") # use a softer color palatte

# # Beautification - Alternate way of selecting palette, applies to all bars
# sns.set_palette("Set3")  # Use a predefined Seaborn color palette for softer colors

# Beautification - adding a title and axis labels
plt.title('Number of Restaurants per City', fontsize=12, weight='bold')  # Title with font size and bold weight
plt.xlabel('City', fontsize=10)  # X-axis label with font size
plt.ylabel('Number of Restaurants', fontsize=10)  # Y-axis label with font size

# Beautification - rotating x-axis labels for better readability
plt.xticks(rotation=45, ha='right')  # Rotate labels by 45 degrees and align them to the right


# Beautification - adding gridlines for better readability
plt.grid(True, linestyle='--', alpha=0.7)  # Add dashed gridlines with 70% opacity

# Show the plot
plt.show()
```

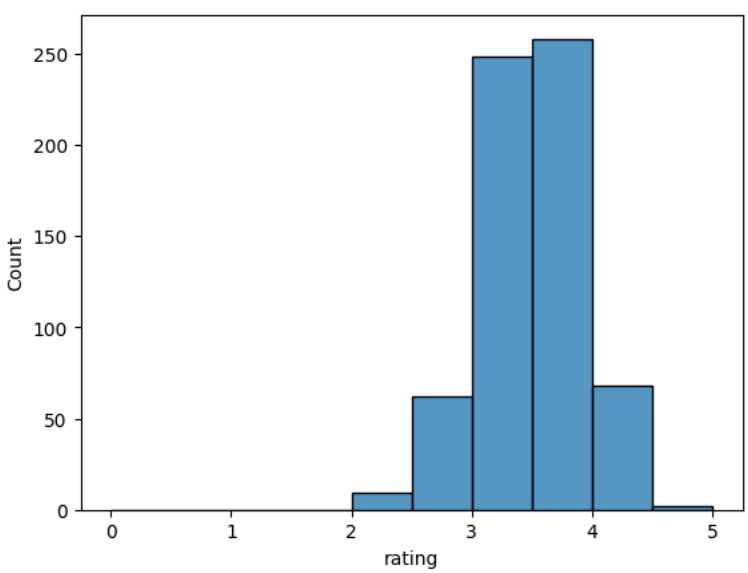


Is there a case when I can use Histogram for categorical data?

"Are most restaurants rated highly, or do we see a normal spread of ratings?"

Create use a histogram for ordinal data (rating)

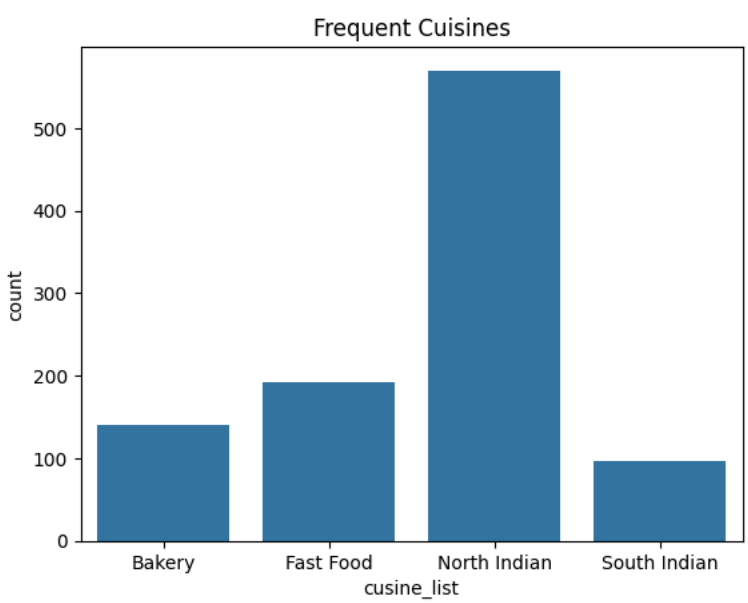
```
sns.histplot(df2['rating'], bins = [0, 1, 2, 2.5, 3, 3.5, 4, 4.5, 5]) # pre-defined bins
# plt.yscale("log") # try without using it first
plt.show()
```



Some questions for students to practice

Which type of cuisine is served by the most restaurants, and how do different cuisines compare?

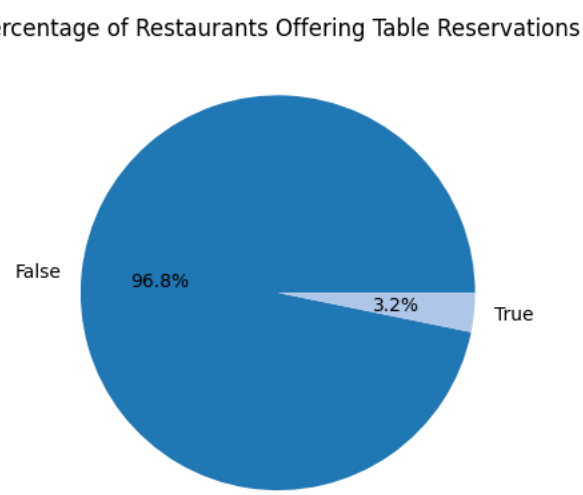
```
sns.countplot(x=df2["cusine_list"])
plt.title("Frequent Cuisines")

plt.show()
```



#"Do most restaurants allow table reservations?"

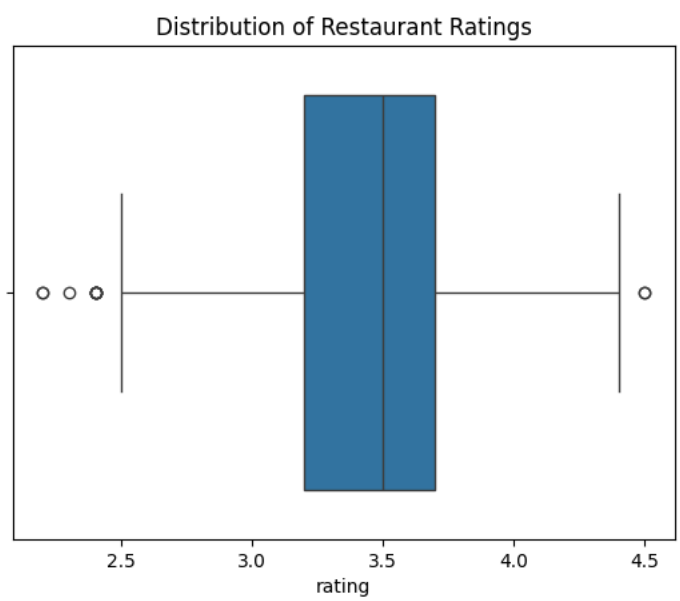
```
df2["table_reservation"].value_counts().plot.pie(autopct='%1.1f%%')
```

```
plt.title("Percentage of Restaurants Offering Table Reservations")
plt.ylabel("")  # Remove y-label for cleaner visualization
plt.show()
```



💡 **How spread out are restaurant ratings? Are there many poorly rated places? **

```
sns.boxplot(x=df2["rating"],data=df2)

plt.title("Distribution of Restaurant Ratings")

plt.show()
```



Bivariate Data Visualisation

Step 1: Why Do We Need Bivariate Analysis?
Before plotting, always ask:

- **What question am I trying to answer?**
- **How many variables are involved?**
- **What are their types? (Categorical or Numerical?)**

Example Questions:

1. Does the cost for two affect restaurant ratings?
2. Is there a relationship between the city and the cuisine type?
3. Do expensive restaurants get higher ratings?

Step 2: Identifying Variable Types

Before choosing a plot, check:

✅ **Are both variables categorical?**

✅ **Are both numerical?**

✅ **Is one categorical and the other numerical?**

Variable 1	Variable 2	Suggested Plot
Categorical	Categorical	Stacked Bar Plot, Grouped Bar Chart
Categorical	Numerical	Boxplot, Violin Plot
Numerical	Numerical	Scatter Plot, Correlation Heatmap

```
sns.set_palette(sns.color_palette("viridis"))
sample_df_wo_outlier = sample_df[sample_df["cost_for_two"] <= 1500]
```

Case 1: Categorical vs. Categorical (City vs. Cuisine)

🔷 **Question:** Does the distribution of cuisines vary by city?

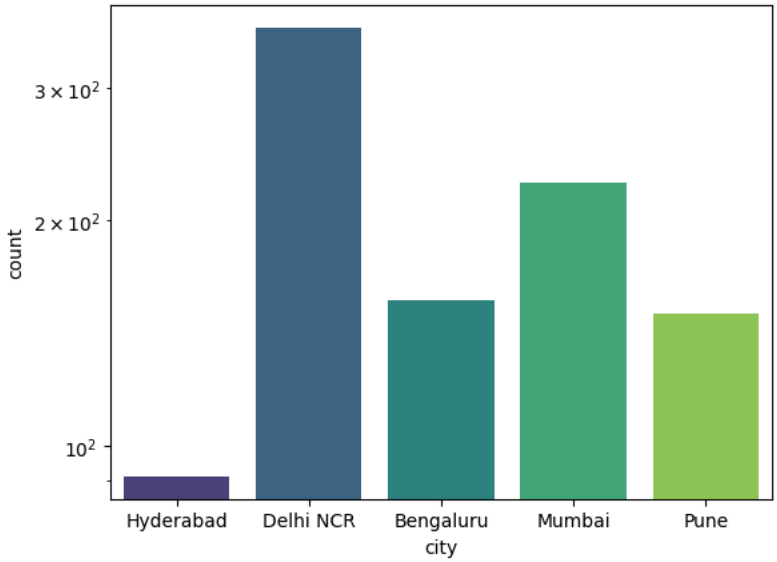
✅ **Best Plot:** Stacked or Grouped Bar Chart

```
# Cross-tabulation of city and cuisine (optional)
city_cuisine_table = pd.crosstab(sample_df['city'], sample_df_wo_outlier['cusine_list'])
print(city_cuisine_table)
```

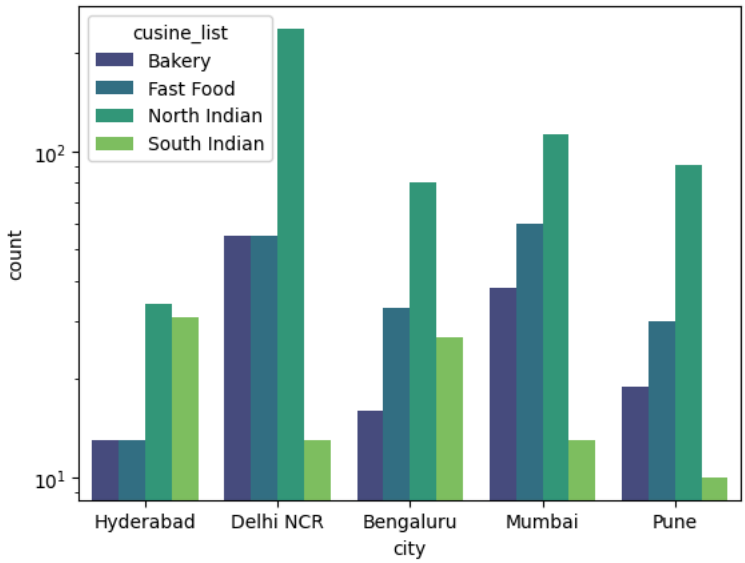
```
cusine_list  Bakery  Fast Food  North Indian  South Indian
city
Bengaluru        16         33            80            27
Delhi NCR        55         55           238            13
Hyderabad        13         13            34            31
Mumbai           38         60           113            13
Pune             19         30            91            10
```

```
sns.countplot(x='city', data=sample_df_wo_outlier, palette="viridis")
plt.yscale('log') # scaling to show less variatipn
plt.show()
```

```
<ipython-input-6-6c40043723d2>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

  sns.countplot(x='city', data=sample_df_wo_outlier, palette="viridis")
```




```
sns.countplot(x='city', hue='cusine_list', data=sample_df_wo_outlier, palette="viridis")
plt.yscale('log') # scaling to show less variatipn
plt.show()
```

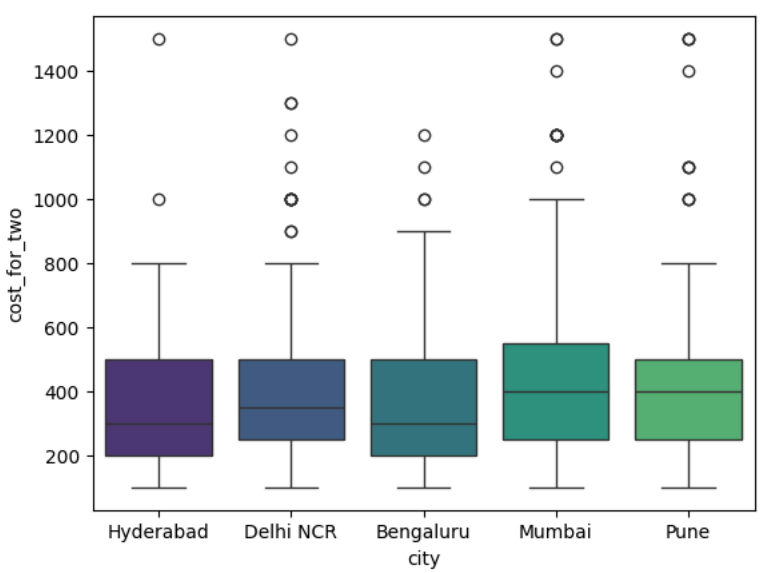


Case-2 Categorical vs. Numerical (City vs. Cost for Two)

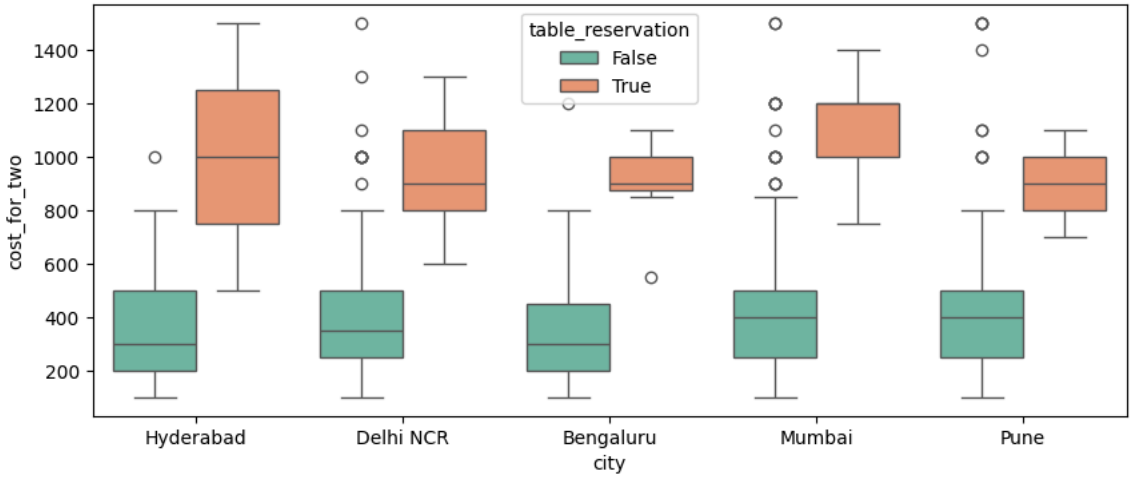
🔷 Question: Does the average cost for two vary by city?

✅ Best Plot: Boxplot

```
sns.boxplot(x='city', y='cost_for_two',
            data = sample_df_wo_outlier, hue="city")
plt.show()
```




```
# Lets see if table reservation makes a difference (optional)
plt.figure(figsize=(10, 4))
sns.boxplot(x='city', y='cost_for_two', hue = "table_reservation",
            data = sample_df_wo_outlier, palette="Set2")
plt.show() # should this even be considered a bivariate analysis?
```

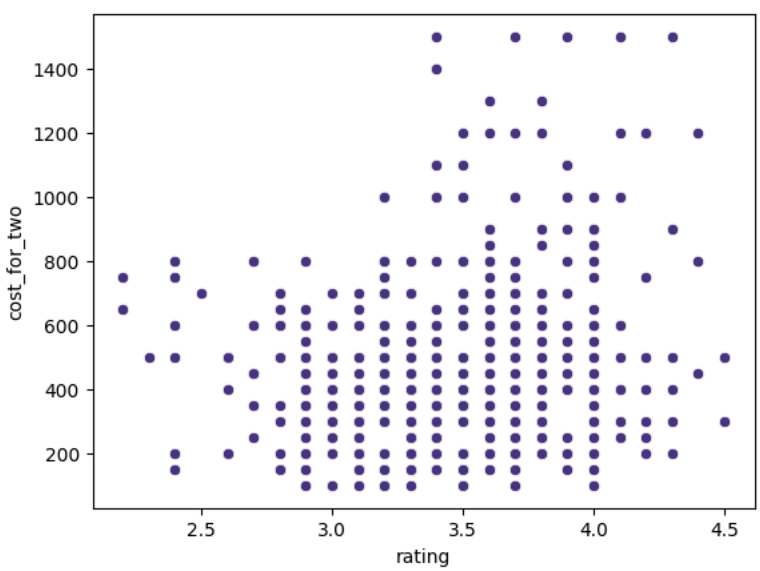


Case 3: Numerical vs. Numerical (Cost for Two vs. Ratings)

🔷 **Question:** Do expensive restaurants get higher ratings?

✅ **Best Plot:** Scatter Plot or Line Plot (if 1:1 between x and y like y = x^2)

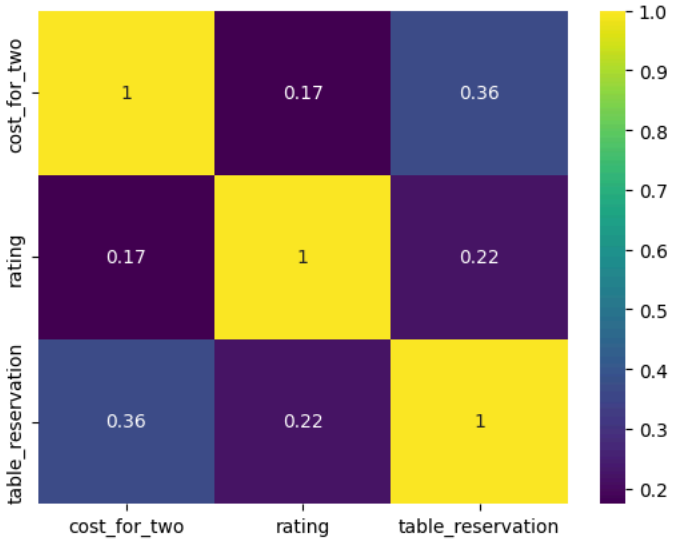
```
sns.scatterplot(x='rating', y='cost_for_two', data = sample_df_wo_outlier)
plt.show()
```



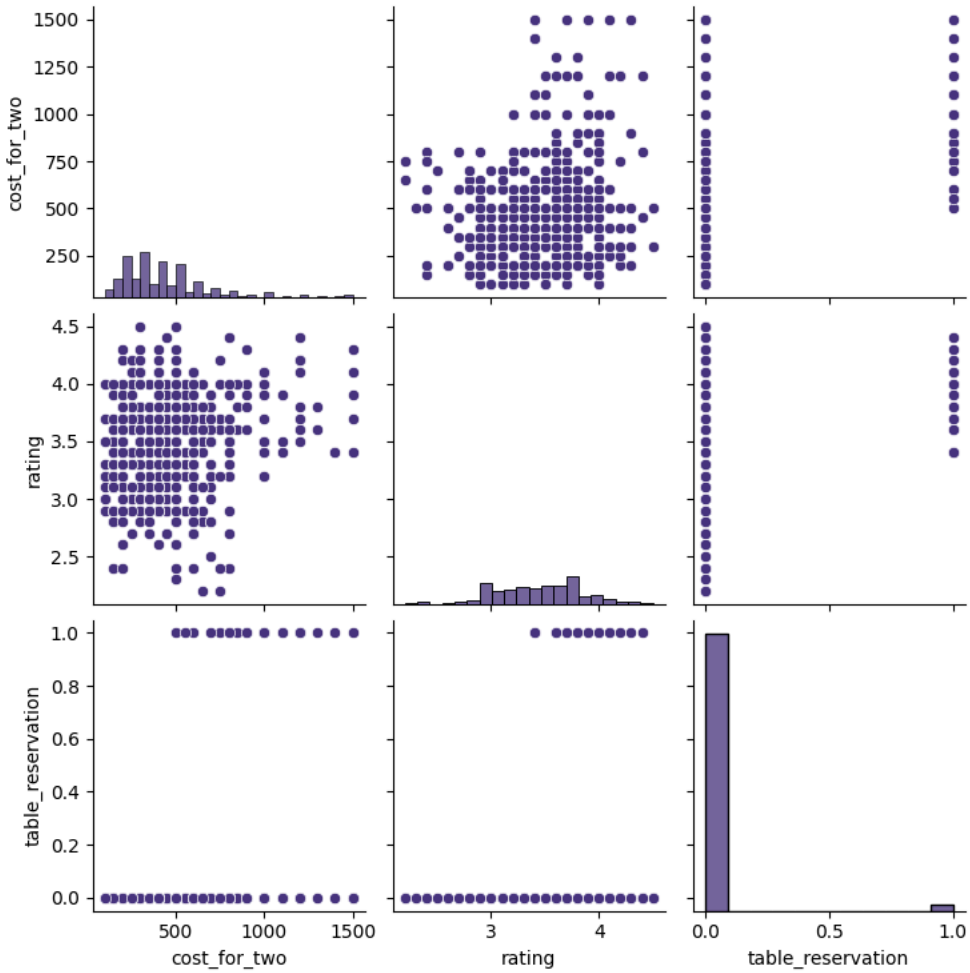

```
correlation = sample_df_wo_outlier[['cost_for_two', 'rating', "table_reservation"]].corr()
print(correlation)
```

```
                    cost_for_two    rating  table_reservation
cost_for_two            1.000000  0.174339           0.362819
rating                  0.174339  1.000000           0.218634
table_reservation       0.362819  0.218634           1.000000
```

```
sns.heatmap(correlation,annot=True,cmap="viridis")
plt.show()
```




```
sns.pairplot(sample_df_wo_outlier, vars = ['cost_for_two', 'rating', "table_reservation"])
plt.show()
```



Extending the idea to Multivariate - go and try yourself :)

Start coding or generate with AI.

Boilterplate code (to retrieve a subset from entire dataset)

Optional - Students may try understanding the code by themselves

```
import os
import gdown
```

```
file_name = 'restaurants_all.csv'
file_path = os.path.join(os.getcwd(), file_name)

if not os.path.exists(file_path):
    url = "https://drive.google.com/uc?id=1qgMZdOpZ_KJAgqc46WW5pvG7xgkJ2jQ6
    gdown.download(url, file_path, quiet=False)
    print(f"The file '{file_path}' has been downloaded.")
else:
```

```
---------------------------------------------------------------------------
```