Zomato Analysis: Crunching Food Insights □□

Introduction: Zomato, a leading online food delivery platform, has amassed a vast dataset that provides valuable insights into various aspects of the food industry, including restaurant locations, cuisines, pricing, and customer preferences. This report presents the findings from an exploratory data analysis (EDA) of Zomato's dataset, aiming to uncover trends and patterns that could inform strategic decision-making for the company.

KEY FINDNIGS

1.Top Countries and Cities:

- India emerges as the leading market for Zomato, with a significant portion of restaurants and customer transactions.
- Within India, cities like New Delhi, Mumbai, and Bangalore stand out for their high restaurant counts and customer engagement.

2.Customer Preferences:

- North Indian cuisine garners the most popularity among Zomato users, followed by other regional and international cuisines.
- Barbeque Nation and other renowned restaurants receive the highest number of votes, indicating their widespread popularity among customers.

3.Price Range Analysis:

- The majority of restaurants fall into the mid-price range categories, with fewer establishments offering premium or budget options.

- Pricey cuisines, such as those exceeding $200,000, cater to niche markets and are relatively scarce compared to more affordable options.

4.Online Delivery and Table Booking:

- A significant portion of restaurants on Zomato offer online delivery services, catering to the growing demand for convenience among customers.

- However, the provision of advance table booking remains less common among restaurants, indicating an area for potential growth and service enhancement.

5.Rating Distribution:

- Ratings on Zomato are diverse, ranging from low to high, with a substantial proportion falling within the average range.

- Ratings on Zomato are diverse, ranging from low to high, with a substantial proportion falling within the average range.

TOOLS & LIBRARIES :

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

- importing libraries refers to bringing in external modules or packages that provide pre-built tools and functions to help with tasks like data manipulation, visualization, statistical analysis, and machine learning.

- warnings.filterwarnings("ignore"): This tells Python to ignore all warnings. If there are any warnings during code execution, they will not be displayed.

Loading the Dataset

In [4]:
```
# Load the dataset
df = pd.read_csv('zomato.csv')
df.head()
```

Out[4]:

	Unnamed: 0.1	Unnamed: 0	restaurant name	restaurant type	rate (out of 5)	num of ratings	avg cost (two people)	on
0	0	0	#FeelTheROLL	Quick Bites	3.4	7	200.0	
1	1	1	#L-81 Cafe	Quick Bites	3.9	48	400.0	
2	2	2	#refuel	Cafe	3.7	37	400.0	
3	3	3	'@ Biryani Central	Casual Dining	2.7	135	550.0	
4	4	4	'@ The Bbq	Casual Dining	2.8	40	700.0	

- loading a dataset refers to the process of importing data into your working environment (e.g., Python, R, or Excel) so that you can manipulate and analyze it.

Getting some basic information about the dataset

1.head()

In [6]:
```
df.head(10)#show the 10 records of dataset
```

Out[6]:

	Unnamed: 0.1	Unnamed: 0	restaurant name	restaurant type
0	0	0	#FeelTheROLL	Quick Bites
1	1	1	#L-81 Cafe	Quick Bites
2	2	2	#refuel	Cafe
3	3	3	'@ Biryani Central	Casual Dining
4	4	4	'@ The Bbq	Casual Dining
5	5	5	'@99	Takeaway, Delivery
6	6	6	'@Italy	Casual Dining
7	7	7	'@North Parontha Hut	Takeaway, Delivery
8	8	8	1000 B.C	Quick Bites
9	9	9	100Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â°C	Casual Dining

- df: This is your pandas DataFrame, where you are storing the Zomato restaurant data.

- head(10): This method retrieves the first 10 rows of the DataFrame. If you don't specify a number, by default it returns the first 5 rows.

2.tail()

In [7]:
```
df.tail() # to show bottom 5 records of dataset
```

Out[7]:

	Unnamed: 0.1	Unnamed: 0	restaurant name	restaurant type	rate (out of 5)	num of ratings	avg cost (two people)	o
7100	7100	7100	Zoey's	Cafe	4.3	894	600.0	
7101	7101	7101	ZOROY Luxury Chocolate	Dessert Parlor	4.0	68	250.0	
7102	7102	7102	Zu's Doner Kebaps	Takeaway, Delivery	3.7	33	350.0	
7103	7103	7103	Zyara	Casual Dining	3.8	191	650.0	
7104	7104	7104	Zyksha	Food Truck	3.4	9	200.0	

- df: The pandas DataFrame containing your Zomato restaurant data.
- .tail(): This method retrieves the last 5 rows of the DataFrame by default. You can specify the number of rows by passing a value like .tail(10) to see the last 10 rows.

3.shape

In [18]:
```
df.shape #to show the No of Rows and Colums
```

Out[18]:
```
(7105, 12)
```

- The df.shape command is used to understand the structure of your dataset, specifically the number of rows and columns in the DataFrame. This is important because it gives you a sense of how large your dataset is and how many features (columns) you have to work with.

4.size

In [19]:
```
df.size # to show No. of total volums(elements) in the dataset
```

Out[19]:
```
85260
```

- The df.size command is used to determine the total number of elements (or data points) in the entire DataFrame. This is a simple way to get the total count of all cells in the DataFrame, which is calculated by multiplying the number of rows by the number of columns.

5.columns

In [21]:
```
df.columns # to show each column Name
```

Out[21]:
```
Index(['Unnamed: 0.1', 'Unnamed: 0', 'restaurant name', 'restaurant type',
       'rate (out of 5)', 'num of ratings', 'avg cost (two people)',
       'online_order', 'table booking', 'cuisines type', 'area',
       'local address'],
      dtype='object')
```

The size of a * dataset refers to the total number of elements (or data points) in the dataset.

6.dtype

In [22]:
```
df.dtypes # to show the data type of each column
```

Out[22]:
```
Unnamed: 0.1               int64
Unnamed: 0                 int64
restaurant name           object
restaurant type           object
rate (out of 5)          float64
num of ratings             int64
avg cost (two people)    float64
online_order              object
table booking             object
cuisines type             object
area                      object
local address             object
dtype: object
```

- The .dtypes attribute provides information about the data types of each column.

7.info()

In [24]:
```
df.info() # To  show indexes,columns,data-type of each columns, memory at or
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7105 entries, 0 to 7104
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Unnamed: 0.1           7105 non-null   int64  
 1   Unnamed: 0             7105 non-null   int64  
 2   restaurant name        7105 non-null   object 
 3   restaurant type        7105 non-null   object 
 4   rate (out of 5)        7037 non-null   float64
 5   num of ratings         7105 non-null   int64  
 6   avg cost (two people)  7048 non-null   float64
 7   online_order           7105 non-null   object 
 8   table booking          7105 non-null   object 
 9   cuisines type          7105 non-null   object 
 10  area                   7105 non-null   object 
 11  local address          7105 non-null   object 
dtypes: float64(2), int64(3), object(7)
memory usage: 666.2+ KB
```

- info(): This method displays:

In [27]:
```
df.describe()
```

Out[27]:

	Unnamed: 0.1	Unnamed: 0	rate (out of 5)	num of ratings	avg cost (two people)
count	7105.000000	7105.000000	7037.000000	7105.000000	7048.000000
mean	3552.000000	3552.000000	3.514253	188.921042	540.286464
std	2051.181164	2051.181164	0.463249	592.171049	462.902305
min	0.000000	0.000000	1.800000	1.000000	40.000000
25%	1776.000000	1776.000000	3.200000	16.000000	300.000000
50%	3552.000000	3552.000000	3.500000	40.000000	400.000000
75%	5328.000000	5328.000000	3.800000	128.000000	600.000000
max	7104.000000	7104.000000	4.900000	16345.000000	6000.000000

- describe(): This method generates descriptive statistics for the numerical columns by default, such as:
- count: The number of non-null entries in each column.
- mean: The average value.
- std: The standard deviation (a measure of the spread of data).
- min: The minimum value.
- 25%, 50% (median), and 75%: Percentile values (indicating the value below which a certain percentage of observations fall).
- max: The maximum value.

In [28]:
```
df.describe(include='object')
```

Out[28]:

	restaurant name	restaurant type	online_order	table booking	cuisines type	
count	7105	7105	7105	7105	7105	
unique	7105	81	2	2	2175	
top	Zyksha	Quick Bites	Yes	No	North Indian, Chinese	Byresandra,Tav
freq	1	2840	3727	6361	421	

- This method provides summary statistics for numerical columns (e.g., Rating, Votes, Average Cost for Two). The statistics include measures like count, mean, standard deviation, minimum, maximum, and percentiles.

In [29]:
```
df.duplicated().sum()
```

Out[29]:
```
np.int64(0)
```

- df.duplicated(): Returns a Series of True or False values, where True indicates that the row is a duplicate.
- sum(): Calculates the total number of True values, i.e., the total number of duplicated rows.

In [30]:
```
df.isnull().sum()
```

Out[30]:
```
Unnamed: 0.1              0
Unnamed: 0                0
restaurant name           0
restaurant type           0
rate (out of 5)          68
num of ratings            0
avg cost (two people)    57
online_order              0
table booking             0
cuisines type             0
area                      0
local address             0
dtype: int64
```

1. df.isnull():

- This method returns a DataFrame of the same shape as df, where each cell contains True if the value is NaN (missing), and False otherwise.

2. .sum():

- By chaining .sum() to df.isnull(), you get the total count of missing values for each column.

In [33]:
```
df['cuisines type'].dropna(axis = 0, inplace = True)
```

- To handle missing values specifically in the Cuisines column of your DataFrame

In [34]:
```
df.nunique()
```

Out[34]:
```
Unnamed: 0.1              7105
Unnamed: 0                7105
restaurant name           7105
restaurant type             81
rate (out of 5)             31
num of ratings             935
avg cost (two people)       65
online_order                 2
table booking                2
cuisines type             2175
area                        30
local address               90
dtype: int64
```

- df.nunique() is a method used to find the number of unique values in each column of the DataFrame

HISTOGRAM:

countplot:

- countplot is a visualization tool used to display the frequency distribution of categorical data
- This is a plotting function available in the Seaborn library (a statistical data visualization library built on top of Matplotlib) that creates a bar plot where each bar represents the count of occurrences for each category of a categorical variable.

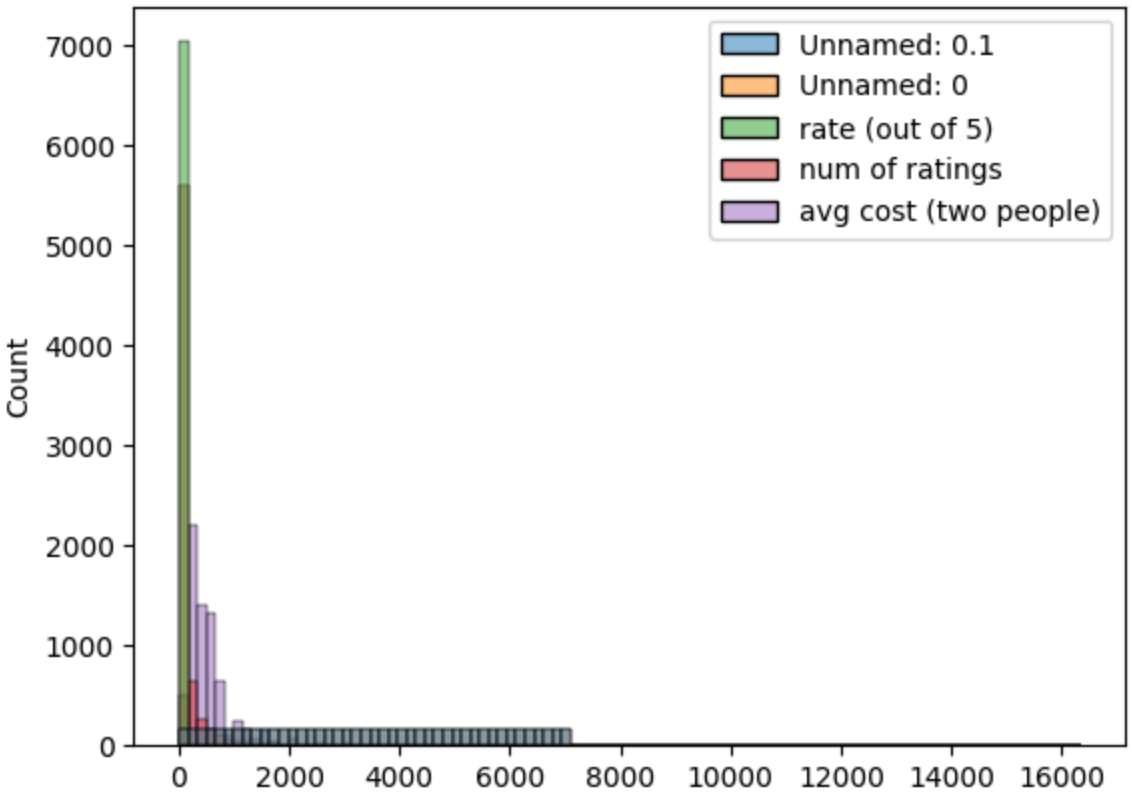
In [8]:
```
df.head()
```

Out[8]:

	Unnamed: 0.1	Unnamed: 0	restaurant name	restaurant type	rate (out of 5)	num of ratings	avg cost (two people)	on
0	0	0	#FeelTheROLL	Quick Bites	3.4	7	200.0	
1	1	1	#L-81 Cafe	Quick Bites	3.9	48	400.0	
2	2	2	#refuel	Cafe	3.7	37	400.0	
3	3	3	'@ Biryani Central	Casual Dining	2.7	135	550.0	
4	4	4	'@ The Bbq	Casual Dining	2.8	40	700.0	

- Overall, most customers think Zomato is a great food delivery application.

In [12]:
```
import numpy as np

data = np.random.randn(100)  # Generate 100 random numbers
sns.histplot(df)            # Plot the histogram
plt.show()
```



1.data = np.random.randn(100):

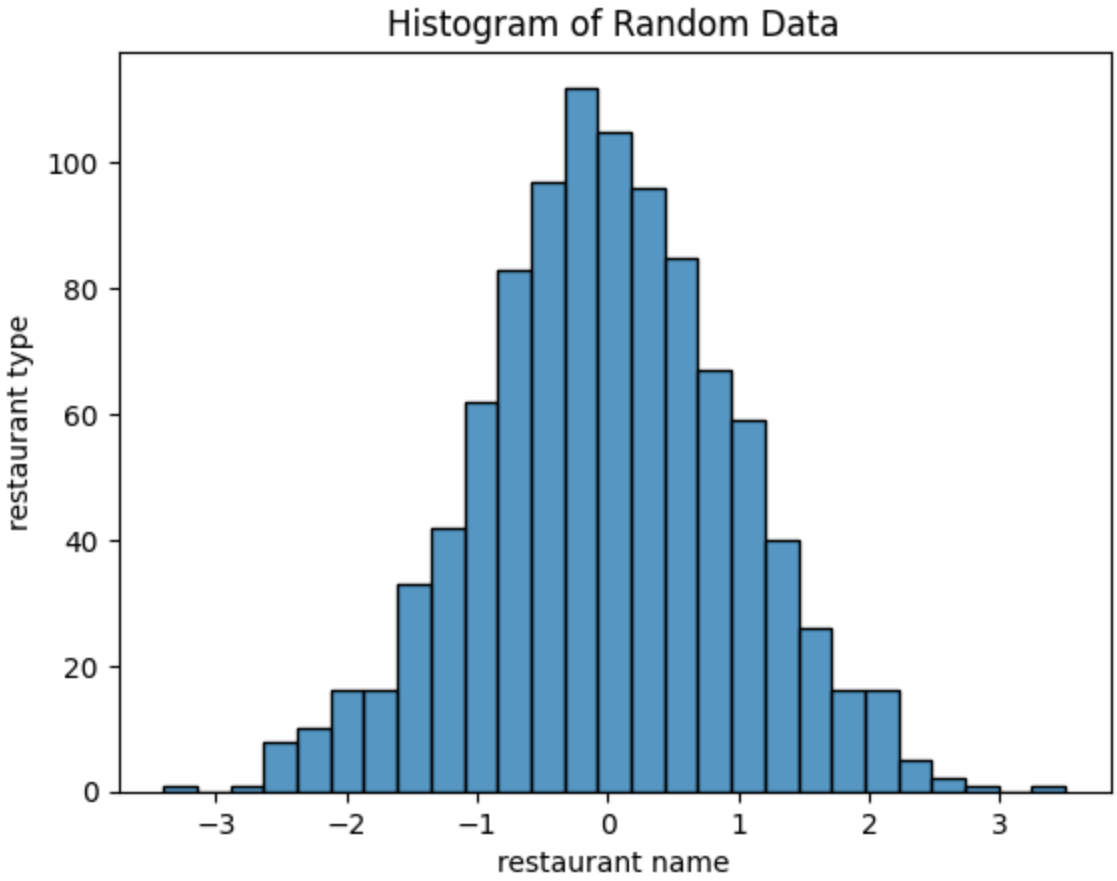
- This generates 100 random numbers from a normal distribution
- np.random.randn() is a function from the NumPy library for generating random numbers.

2.sns.histplot(data):

This plots a histogram using the Seaborn library (typically imported as sns). A histogram visualizes the distribution of the data by grouping it into "bins" and showing how frequently each value occurs. In this case, it visualizes the data generated earlier (100 random numbers).

In [ ]:

In [10]:
```
sns.histplot(data)
plt.xlabel('restaurant name')
plt.ylabel('restaurant type')
plt.title('Histogram of Random Data')
plt.show()
```



Histograms are designed to represent the distribution of numerical data.

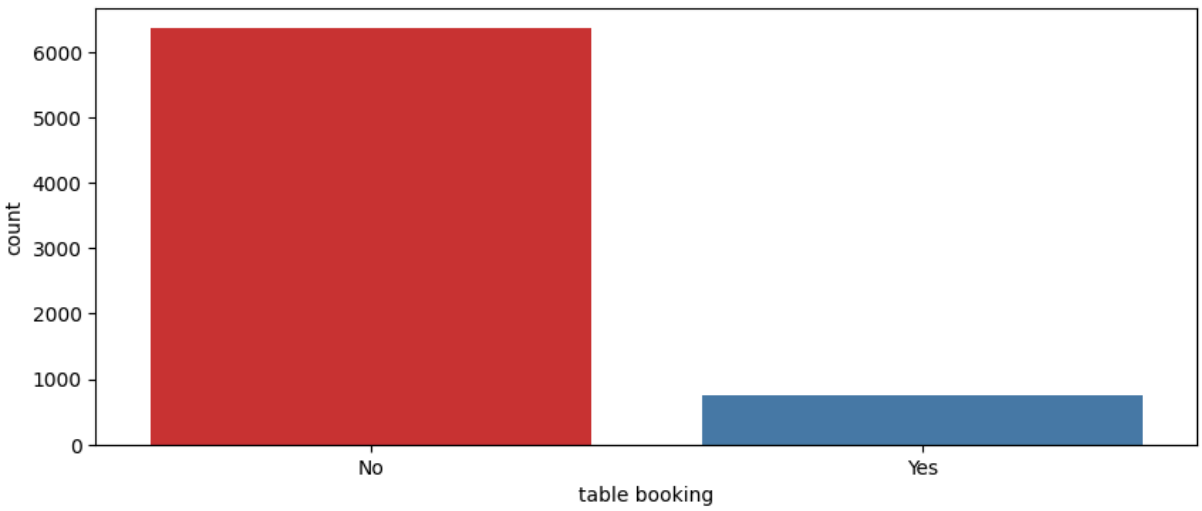
DATA VISUALIZATION

In [ ]:

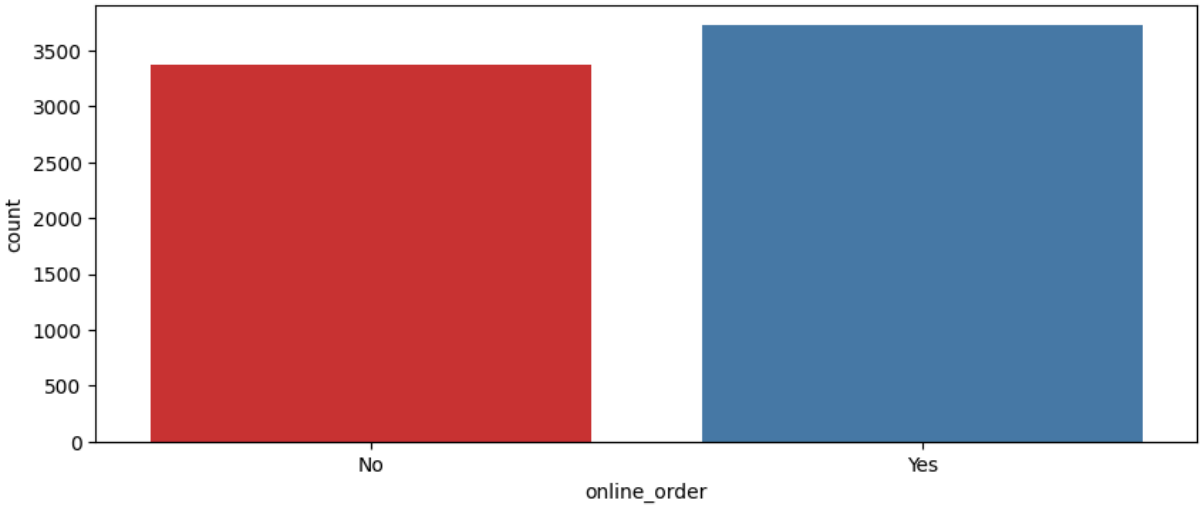
Data visualization involves representing data in a graphical or pictorial format, making it easier to understand patterns, trends, and insights. It transforms raw data into visuals like charts, graphs, and maps. Some popular types of visualizations include:

- Bar Charts: Compare different categories or groups.
- Line Charts: Display trends over time.
- Pie Charts: Show proportions or percentages.

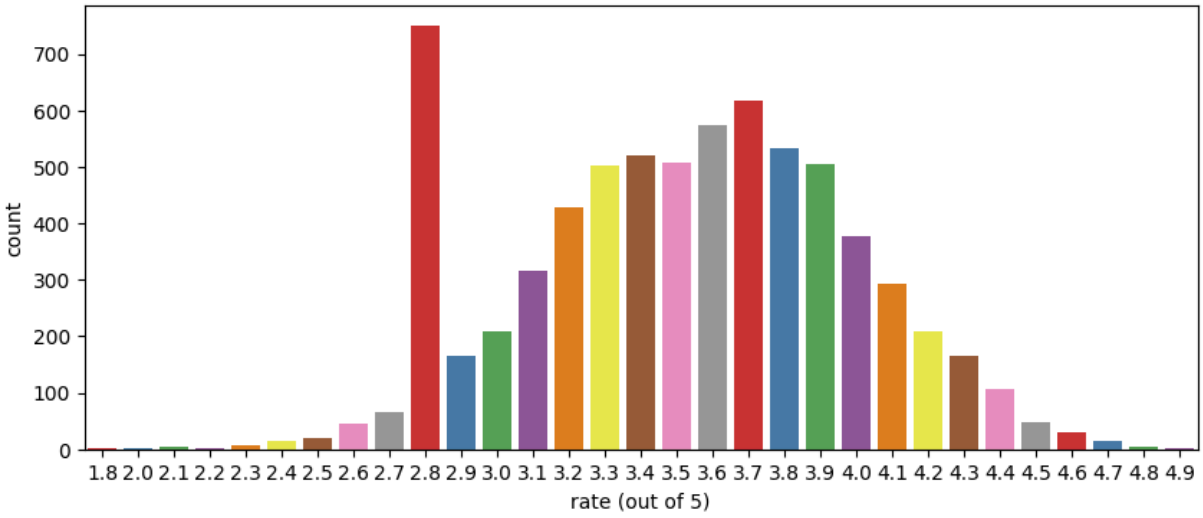
In [14]:
```
plt.figure(figsize = (10, 4))
sns.countplot(x = "table booking", data = df, palette="Set1")
plt.show()
```



In [17]:
```
plt.figure(figsize = (10, 4))
sns.countplot(x = "online_order", data = df, palette="Set1")
plt.show()
```

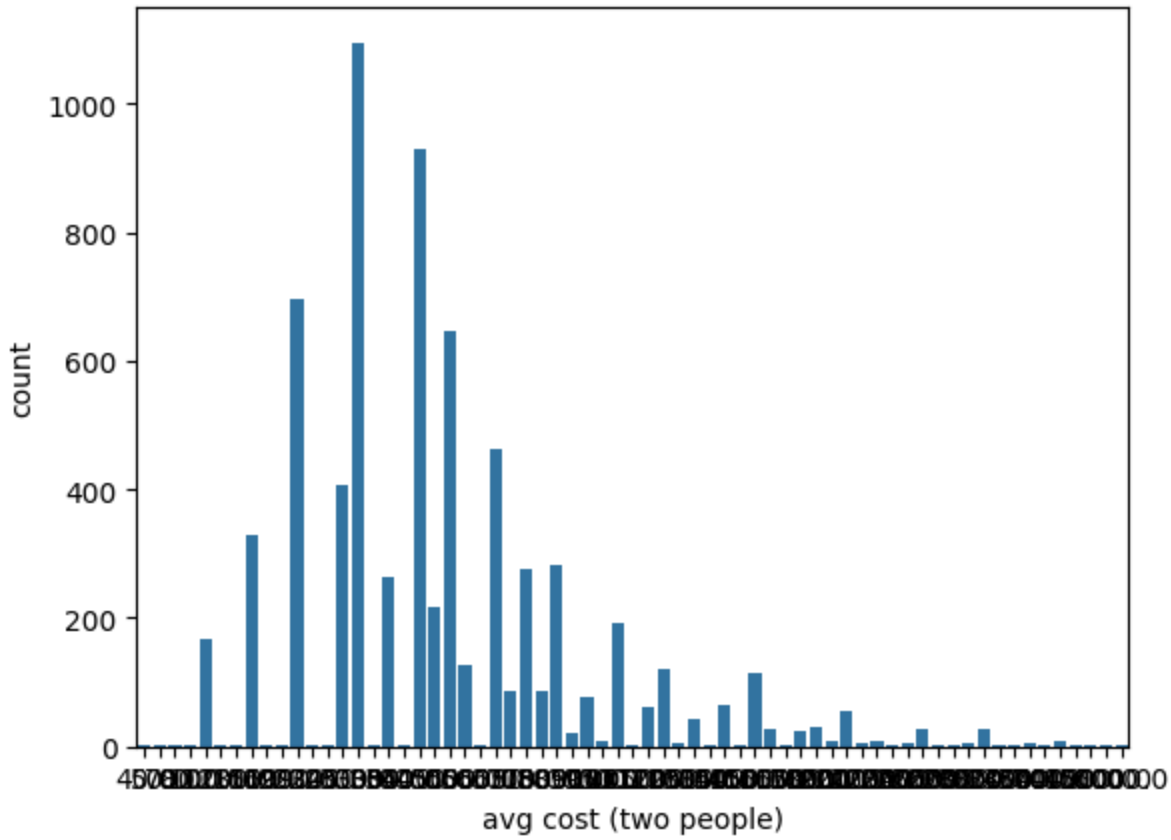


In [18]:
```
plt.figure(figsize = (10, 4))
sns.countplot(x = "rate (out of 5)", data = df, palette="Set1")
plt.show()
```



In [50]:
```
couple_data=df['avg cost (two people)']
sns.countplot(x=couple_data)
```

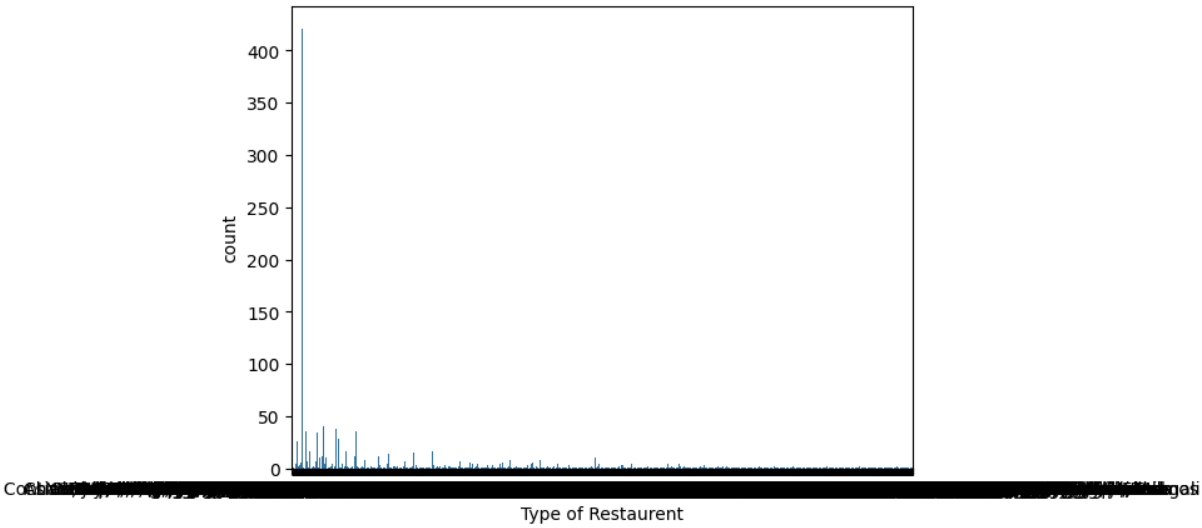
Out[50]:
```
<Axes: xlabel='avg cost (two people)', ylabel='count'>
```



Majority Prefer restaurents with appx cost of 1000 rs

In [58]:
```
sns.countplot(x=df['cuisines type'])
plt.xlabel("Type of Restaurent")
```

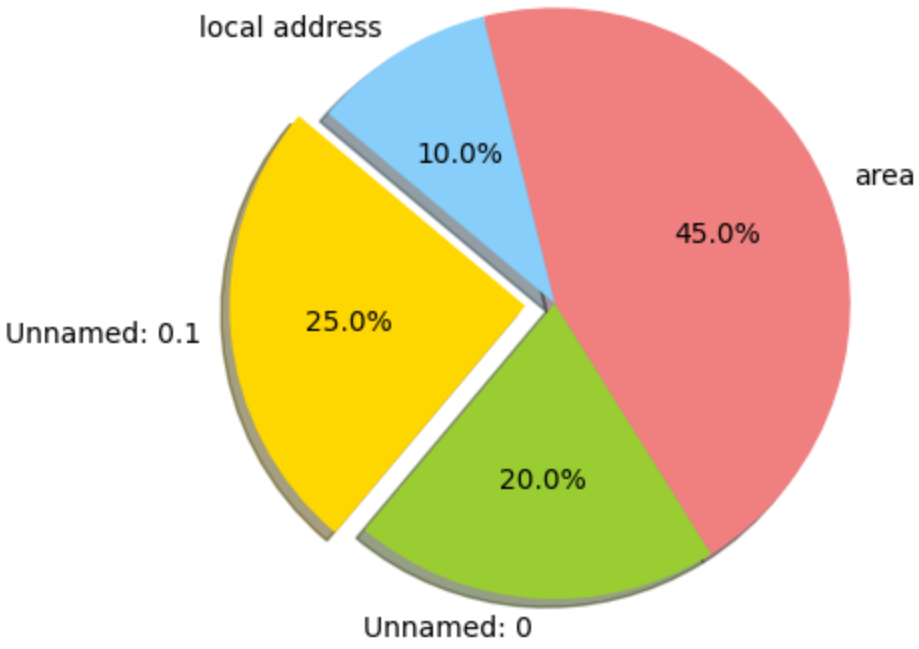
Out[58]:
```
Text(0.5, 0, 'Type of Restaurent')
```



Majority Restaurent >> Dining Category

In [29]:
```
# Data to plot
sizes = [25, 20, 45, 10]
                                    'Unnamed: 0', 'area', 'local address']
```

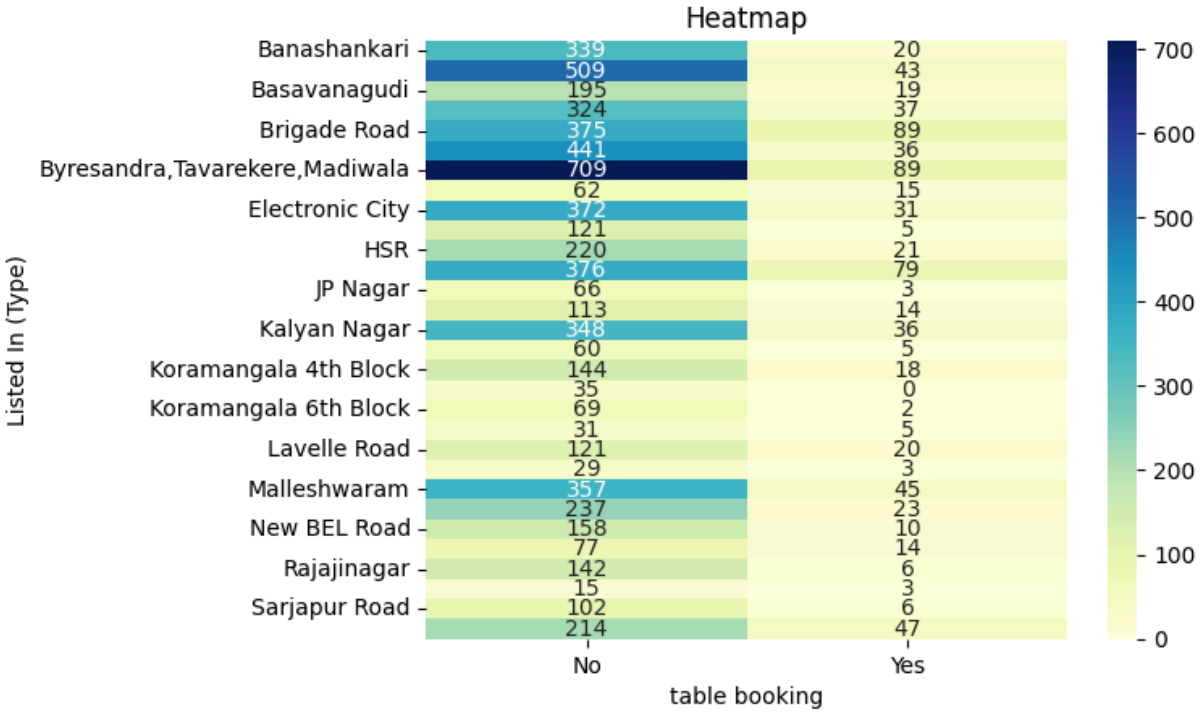
```
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue']
explode = (0.1, 0, 0, 0) # explode the first slice

# Create a pie chart with customizations
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f

# Display the chart
plt.show()
```



It helps illustrate how a whole dataset is divided into different parts by showing * the percentage or fraction of each category. The size of each "slice" of the pie corresponds to the size of the category in relation to the whole dataset.

HEAT MAP

In [62]:
```
pivot_table = df.pivot_table(index='area', columns='table booking', aggfunc=
sns.heatmap(pivot_table, annot=True, cmap="YlGnBu",fmt='d')
plt.title("Heatmap")
plt.xlabel("table booking")
plt.ylabel("Listed In (Type)")
plt.show()
```

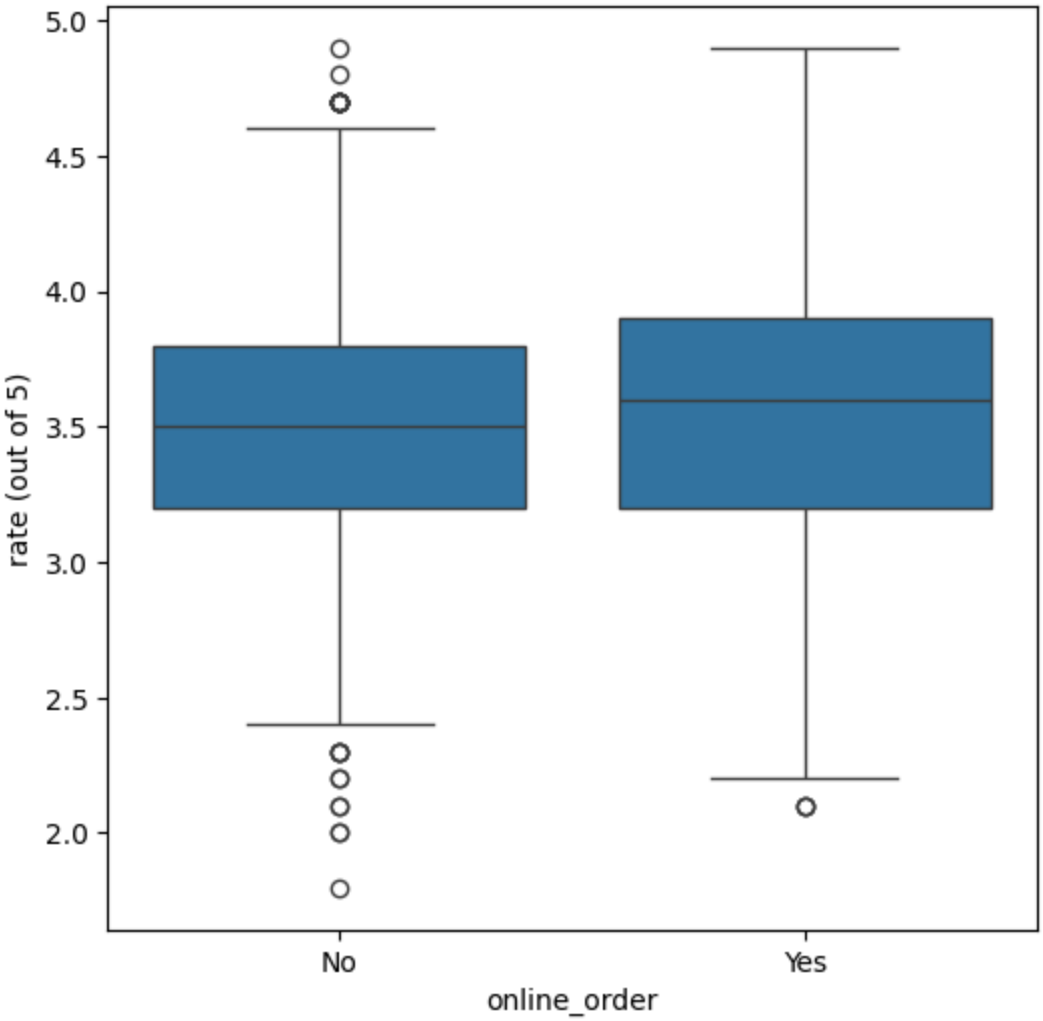


1. sns.histplot(data) This line uses the histplot function from the Seaborn library to create a histogram. data represents the dataset being visualized, which can be a Pandas DataFrame column, a list, or an array of values. A histogram shows the distribution of numeric data by dividing the range of data into bins and counting how many values fall into each bin.
2. plt.xlabel('restaurant name') This sets the x-axis label to "restaurant name" using Matplotlib's xlabel function. It suggests that the data likely contains information about restaurant names.
3. plt.ylabel('restaurant type') This sets the y-axis label to "restaurant type." However, this is unusual for a histogram, since histograms typically represent the frequency of data on the y-axis (like counts or probability density). It's possible that the dataset uses some type of encoding where the restaurant types have numeric representations.
4. plt.title('Histogram of Random Data') This adds a title, "Histogram of Random Data," to the plot.
5. plt.show() This command renders the plot for display in your output environment (like Jupyter Notebook, a Python script, or a terminal).

Dining restaurents Primarily accept offline orders unlike cafes primarily receive online orders. This suggest that clients prefer to place orders in person at restaurents, but prefer online ordering at cafes.

In [44]:
```
plt.figure(figsize =(6,6))
sns.boxplot(x = 'online_order', y= 'rate (out of 5)',data=df)
```

Out[44]: <Axes: xlabel='online_order', ylabel='rate (out of 5)'>



Conclusion:

- In this Zomato dataset analysis, we explored various aspects of the restaurant industry, uncovering key trends and insights:
- Cuisines and Preferences: Indian, North Indian, and Chinese cuisines dominate the restaurant scene, particularly in urban areas. However, niche cuisines are emerging in high-end restaurants, catering to diverse customer tastes.
- Ratings vs. Cost: There's a general trend that higher-priced restaurants tend to receive better ratings. However, certain budget-friendly restaurants also stand out with excellent customer reviews, indicating that good food and service can surpass pricing expectations.
- City-Level Insights: Major cities like Delhi, Bangalore, and Mumbai show the highest concentration of restaurants. While Delhi leads in quantity, Bangalore exhibits a more balanced distribution of price and ratings.

Offline order's rating is lower than the Online order rating

THANK YOU

END

In [ ]: