Introduction

Zomato is an Indian multinational restaurant aggregator and food delivery company. It was founded by Deepinder Goyal and Pankaj Chaddah in 2008. Zomato provides information, menus and user-reviews of restaurants as well as food delivery options from partner restaurants in more than 1,000 Indian cities and towns, as of 2022–23

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
!gdown "https://drive.google.com/uc?export=download&id=1lamgErENUmuzvENgl4nMtbaJohwEP8KP" -O Zomato.csv
```

```
Downloading...
From: https://drive.google.com/uc?export=download&id=1lamgErENUmuzvENgl4nMtbaJohwEP8KP
To: /content/Zomato.csv
100% 6.46k/6.46k [00:00<00:00, 16.1MB/s]
```

In [3]:
```
df = pd.read_csv("Zomato.csv")
df
```

Out[3]:

	name	online_order	book_table	rate	votes	approx_cost(for two people)	listed_in(type)
0	Jalsa	Yes	Yes	4.1/5	775	800	Buffet
1	Spice Elephant	Yes	No	4.1/5	787	800	Buffet
2	San Churro Cafe	Yes	No	3.8/5	918	800	Buffet
3	Addhuri Udupi Bhojana	No	No	3.7/5	88	300	Buffet
4	Grand Village	No	No	3.8/5	166	600	Buffet
...	...	...	...	...	...	...	...
143	Melting Melodies	No	No	3.3/5	0	100	Dining
144	New Indraprasta	No	No	3.3/5	0	150	Dining
145	Anna Kuteera	Yes	No	4.0/5	771	450	Dining
146	Darbar	No	No	3.0/5	98	800	Dining
147	Vijayalakshmi	Yes	No	3.9/5	47	200	Dining

148 rows × 7 columns

In [4]:
```
#getting the first and last few rows of the dataset
df.head()
```

Out[4]:

	name	online_order	book_table	rate	votes	approx_cost(for two people)	listed_in(type)
0	Jalsa	Yes	Yes	4.1/5	775	800	Buffet
1	Spice Elephant	Yes	No	4.1/5	787	800	Buffet
2	San Churro Cafe	Yes	No	3.8/5	918	800	Buffet
3	Addhuri Udupi Bhojana	No	No	3.7/5	88	300	Buffet
4	Grand Village	No	No	3.8/5	166	600	Buffet

In [5]:
```
df.tail()
```

Out[5]:

	name	online_order	book_table	rate	votes	approx_cost(for two people)	listed_in(type)
143	Melting Melodies	No	No	3.3/5	0	100	Dining
144	New Indraprasta	No	No	3.3/5	0	150	Dining
145	Anna Kuteera	Yes	No	4.0/5	771	450	Dining
146	Darbar	No	No	3.0/5	98	800	Dining
147	Vijayalakshmi	Yes	No	3.9/5	47	200	Dining

In [6]:
```
df.shape
```

Out[6]:
```
(148, 7)
```

There are 148 rows and 7 columns

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148 entries, 0 to 147
Data columns (total 7 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   name                         148 non-null    object
 1   online_order                 148 non-null    object
 2   book_table                   148 non-null    object
 3   rate                         148 non-null    object
 4   votes                        148 non-null    int64
 5   approx_cost(for two people)  148 non-null    int64
 6   listed_in(type)              148 non-null    object
dtypes: int64(2), object(5)
memory usage: 8.2+ KB
```

Here the rates are recognized as an object. We have to convert the "rate" column into a float and remove the denomenator

In [8]:
```
def rateHandle(data) :
  data = str(data).split("/")
  data = data[0]
  return float(data)

df['rate'] = df['rate'].apply(rateHandle)
df.head()
```

Out[8]:

	name	online_order	book_table	rate	votes	approx_cost(for two people)	listed_in(type)
0	Jalsa	Yes	Yes	4.1	775	800	Buffet
1	Spice Elephant	Yes	No	4.1	787	800	Buffet
2	San Churro Cafe	Yes	No	3.8	918	800	Buffet
3	Addhuri Udupi Bhojana	No	No	3.7	88	300	Buffet
4	Grand Village	No	No	3.8	166	600	Buffet

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148 entries, 0 to 147
Data columns (total 7 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   name                         148 non-null    object
 1   online_order                 148 non-null    object
 2   book_table                   148 non-null    object
 3   rate                         148 non-null    float64
 4   votes                        148 non-null    int64
 5   approx_cost(for two people)  148 non-null    int64
 6   listed_in(type)              148 non-null    object
dtypes: float64(1), int64(2), object(4)
memory usage: 8.2+ KB
```

In [10]:
```
#to check for null or missing values
df.isnull().sum()
```

Out[10]:
```
name                           0
online_order                   0
book_table                     0
rate                           0
votes                          0
approx_cost(for two people)    0
listed_in(type)                0
dtype: int64
```

There are no missing values in the dataset

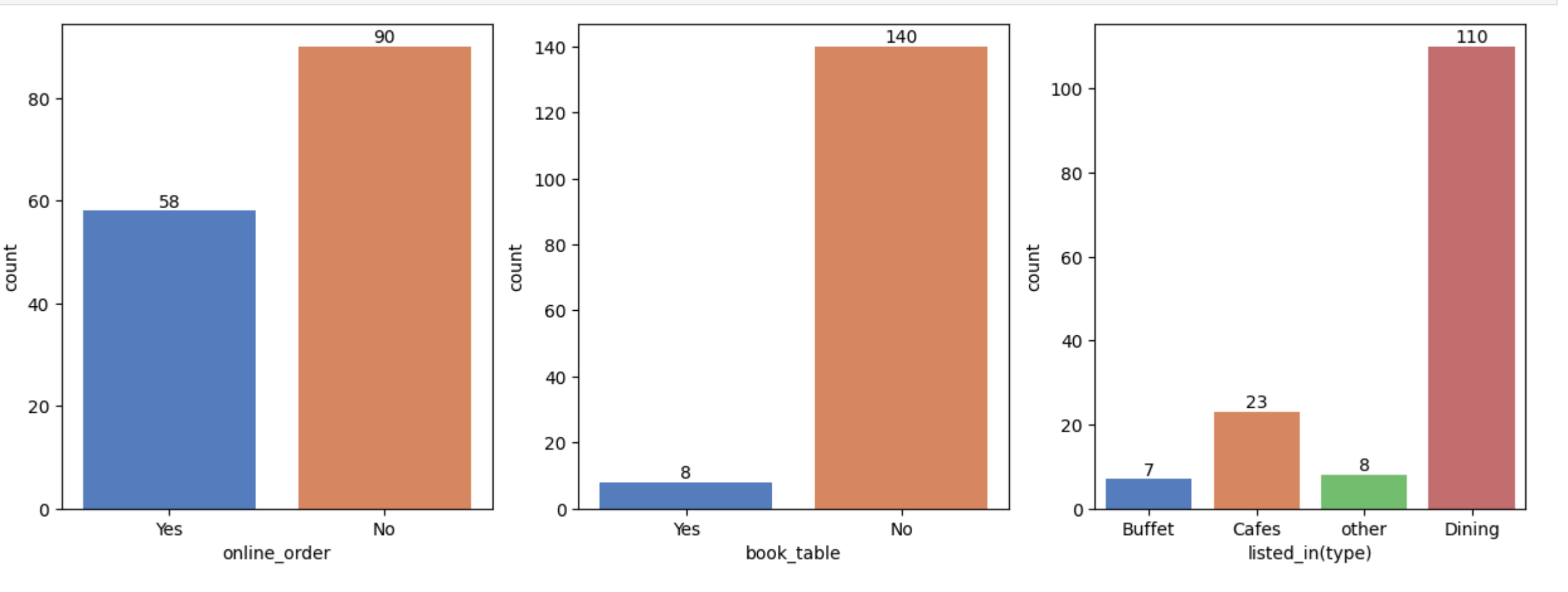
In [11]:
```
#to check if there are any duplicates in the dataset
df.duplicated().sum()
```

Out[11]:
```
0
```

In [12]:
```
cat_cols = ['online_order','book_table','listed_in(type)']
#checking the value count of each column
for i in cat_cols :
  print(df[i].value_counts())
  print("--"*20)
```

```
online_order
No     90
Yes    58
Name: count, dtype: int64
----------------------------------------
book_table
No     140
Yes      8
Name: count, dtype: int64
----------------------------------------
listed_in(type)
Dining    110
Cafes      23
other       8
Buffet      7
Name: count, dtype: int64
----------------------------------------
```

In [13]:
```
#visually we can see that
fig, axes = plt.subplots(nrows = 1, ncols = 3, figsize = (15,5))
index = 0
for i in range(3) :
  ax = sns.countplot(data=df, x = cat_cols[index], ax = axes[i],palette = 'muted')
  for j in ax.containers :
    ax.bar_label(j,)
  index += 1
```



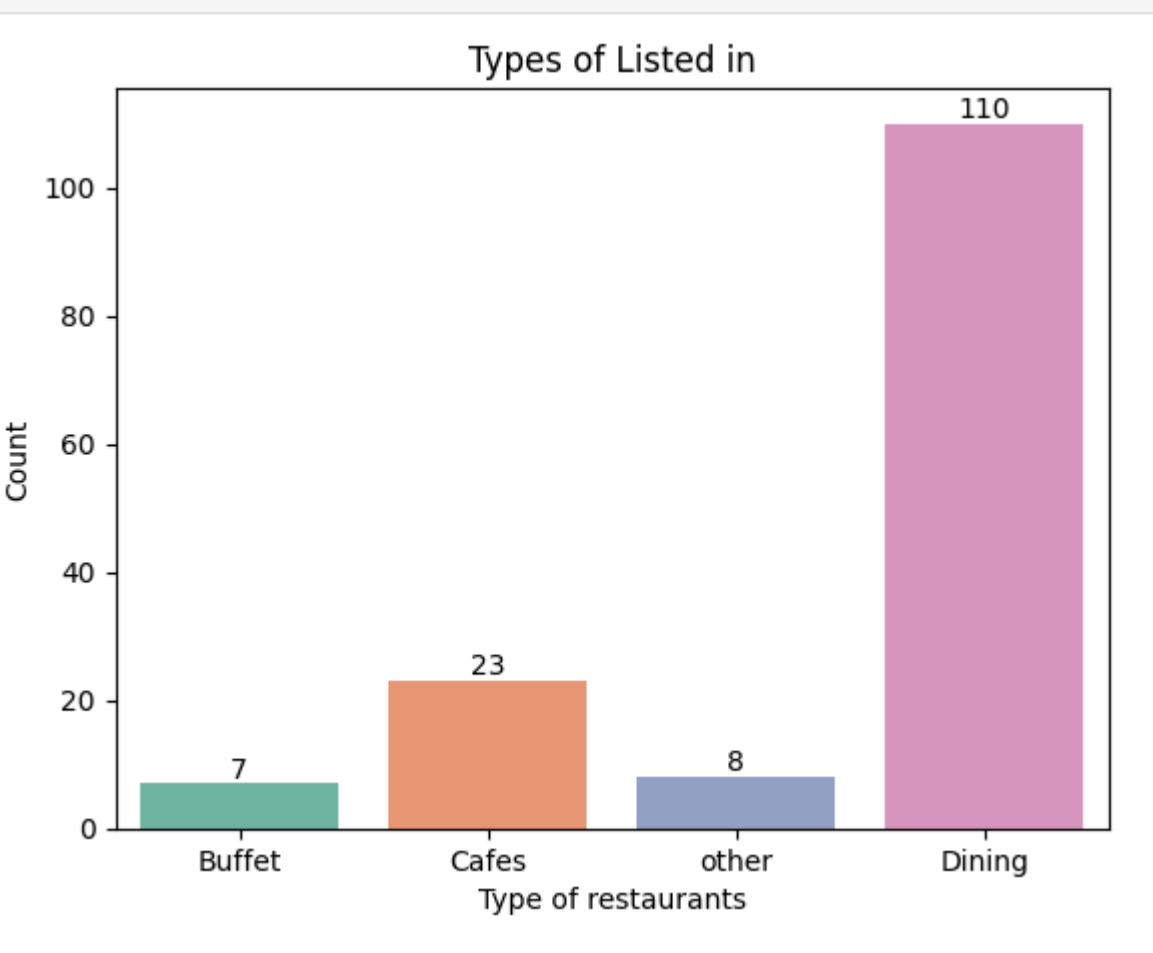
In [ ]:

Let us take a look at the listed_in(type) column

In [14]:
```
data = df['listed_in(type)'].value_counts()
print(data)
print()
print(data/df['listed_in(type)'].shape[0] * 100)
```

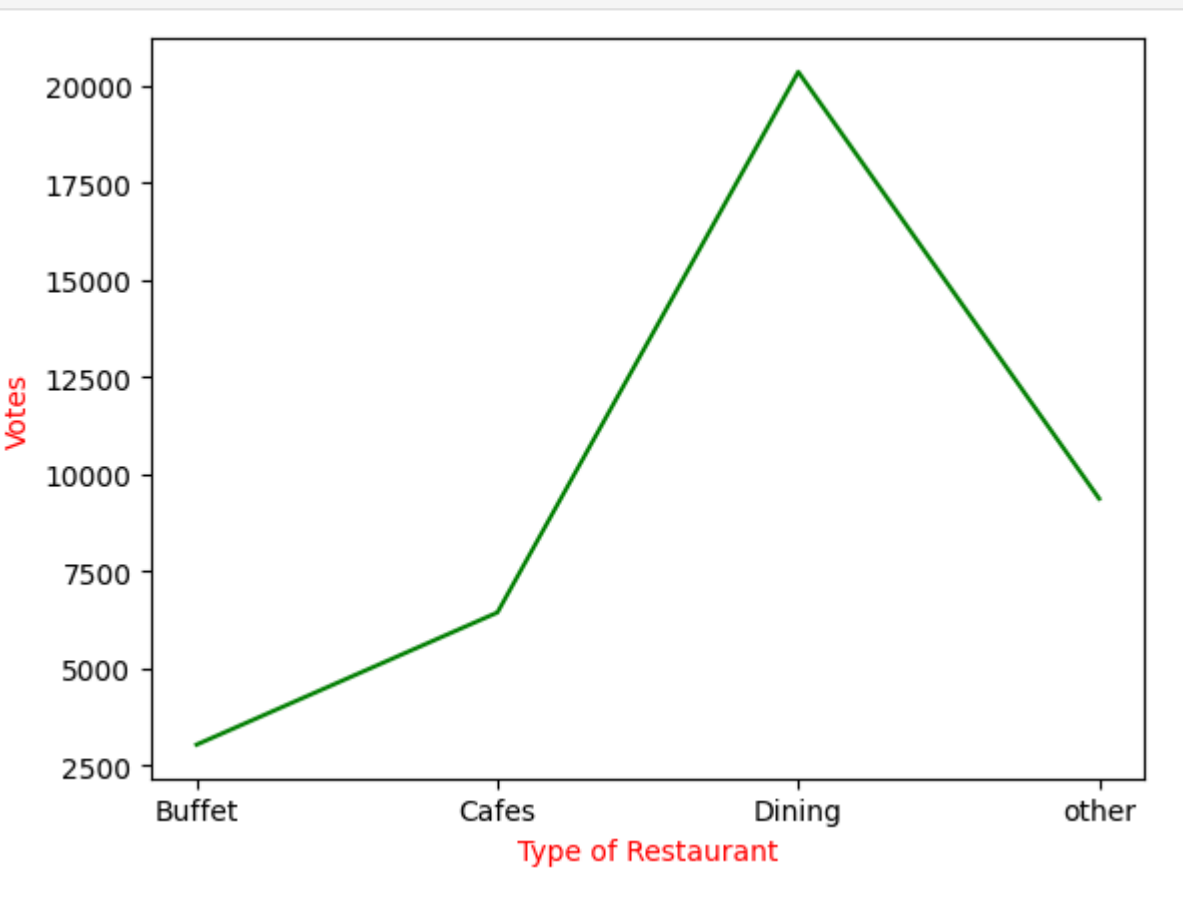
```
listed_in(type)
Dining    110
Cafes      23
other       8
Buffet      7
Name: count, dtype: int64

listed_in(type)
Dining    74.324324
Cafes     15.540541
other      5.405405
Buffet     4.729730
Name: count, dtype: float64
```

In [15]:
```
ax = sns.countplot(data=df,x=df['listed_in(type)'],palette = 'Set2')
plt.title("Types of Listed in")
plt.xlabel("Type of restaurants")
plt.ylabel("Count")
for i in ax.containers :
  ax.bar_label(i,)
plt.show()
```



The majority of Hotels fall into Dining Category. Around 74.32% opts Dining.

In [16]:
```
#number of votes
grouped_data = df.groupby(['listed_in(type)'])['votes'].sum()
#if we use a line plot to visualize it
sns.lineplot(data = grouped_data,c = 'green')
plt.xlabel("Type of Restaurant",color = 'red')
plt.ylabel("Votes",color = 'red')
plt.show()
```



A larger number of peaple prefers the Dining Restaurants

In [17]:
```
#to get the hotel with max number of votes
max_votes = df['votes'].max()
max_vote_restaurant = df['name'].loc[df['votes'] == max_votes]
print("The Restaurant with most number of votes is :")
print(max_vote_restaurant)
```

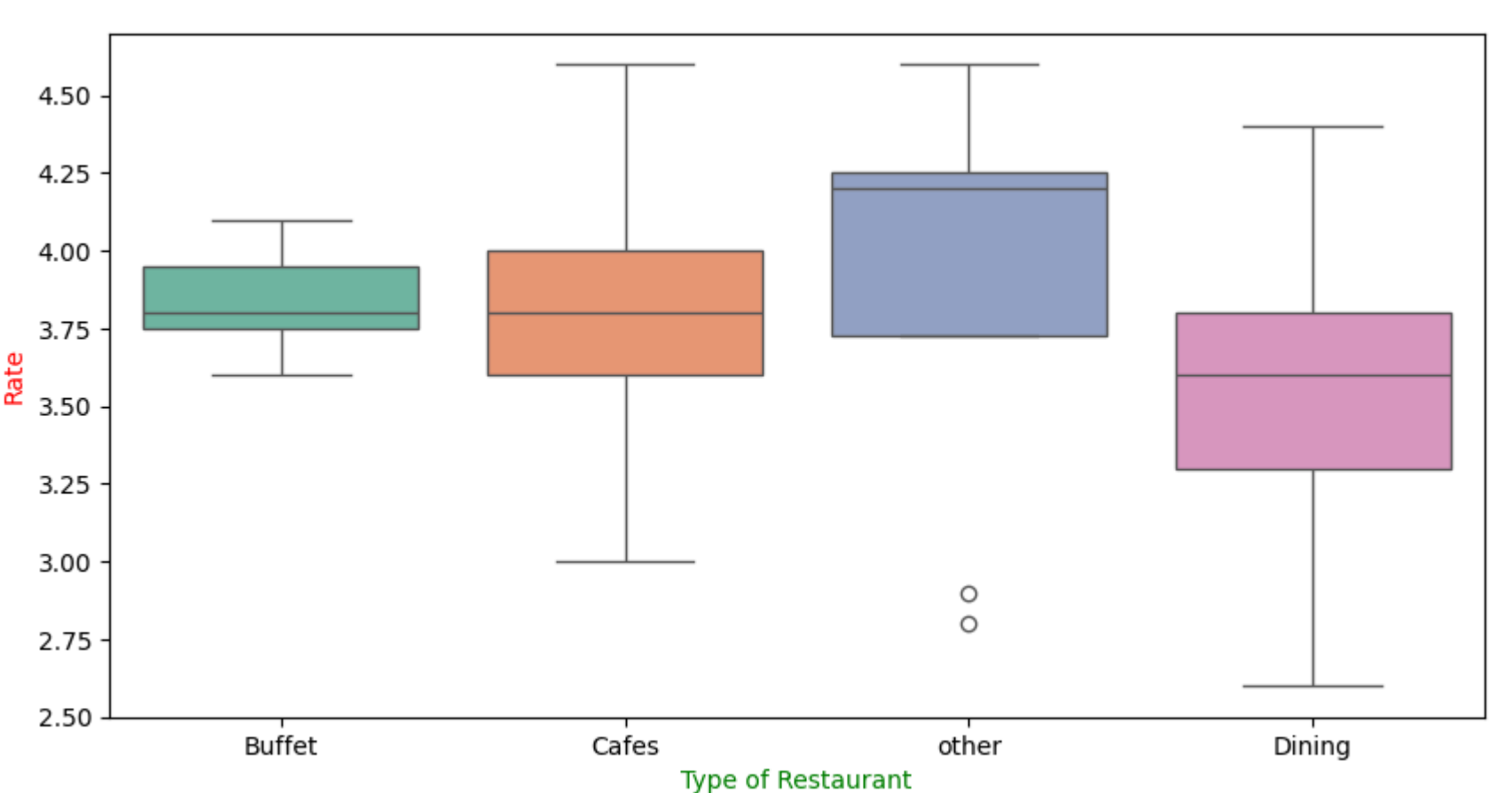
```
The Restaurant with most number of votes is :
38    Empire Restaurant
Name: name, dtype: object
```

In [ ]:

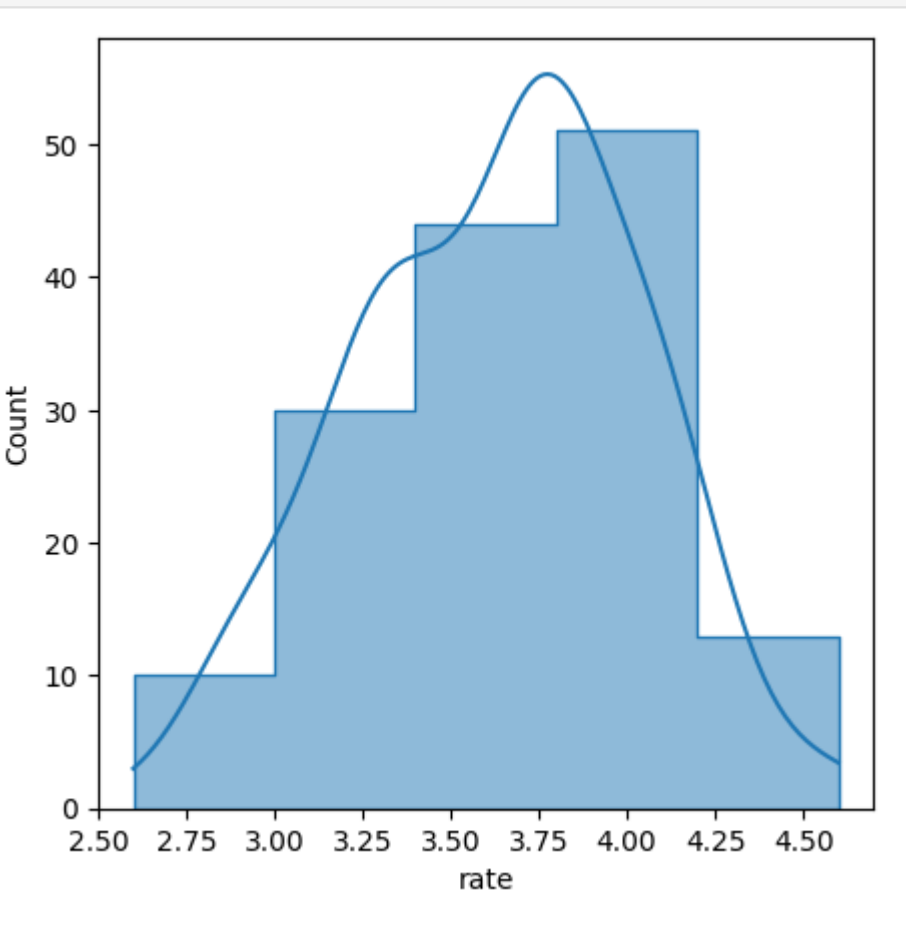
Looking at the ratings column

In [18]:
```
plt.figure(figsize = (10,5))
sns.boxplot(data=df,x='listed_in(type)',y='rate',palette='Set2')
plt.ylabel('Rate',color='red')
plt.xlabel('Type of Restaurant',color='green')
```

Out[18]:
```
Text(0.5, 0, 'Type of Restaurant')
```



In [19]:
```
#looking at the distribution of ratings
plt.figure(figsize=(5,5))
sns.histplot(data=df,x = 'rate',bins=5,element = 'step',kde=True)
plt.show()
```



The majority of ratings falls in the range of 3.50 and 4.00

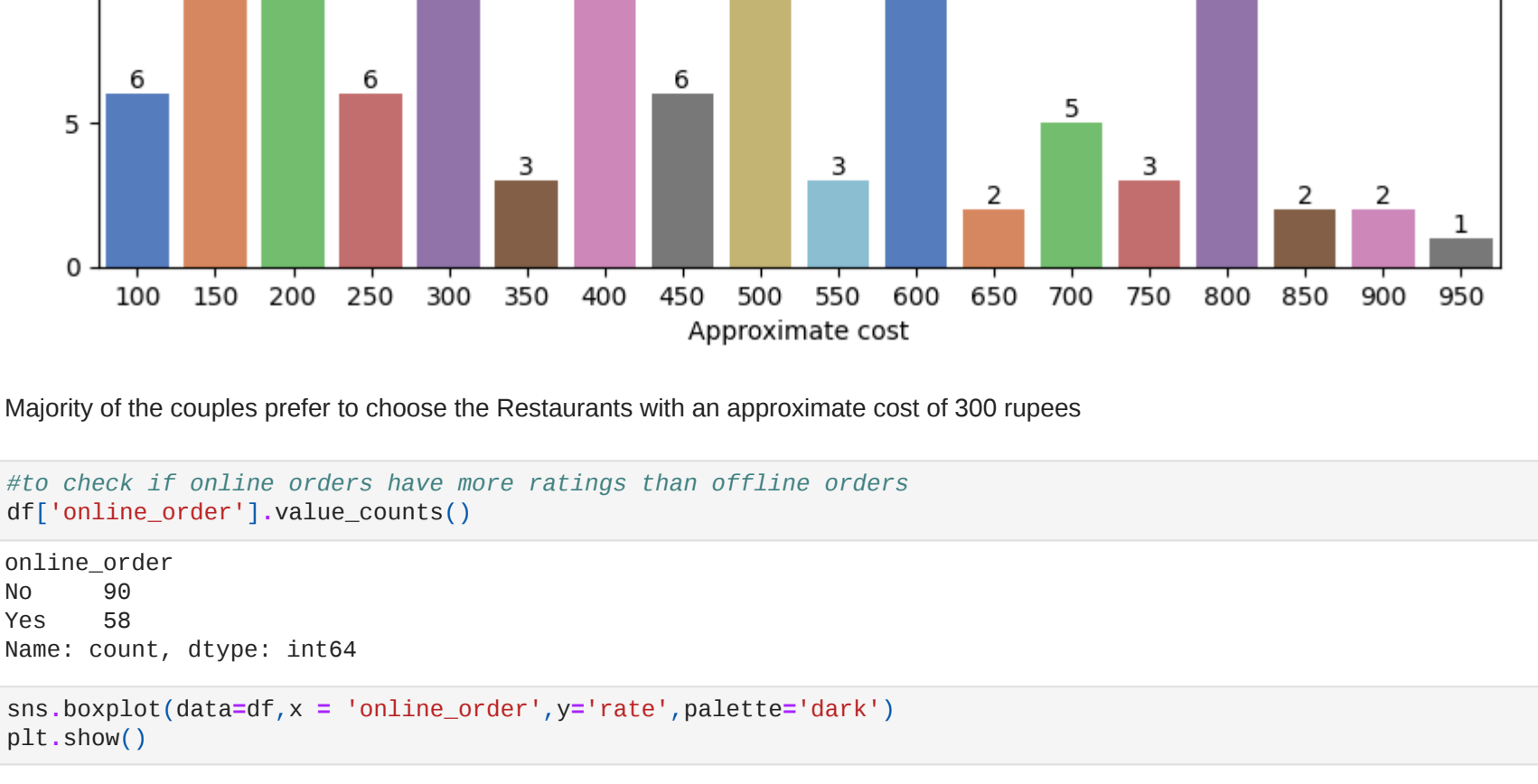
In [ ]:

Let us look at the 'approx_cost(for two people)' column which tells us the approximate cost spent for two people in a Restaurant

In [20]:
```
data = df['approx_cost(for two people)'].value_counts()
data
```

Out[20]:
```
approx_cost(for two people)
300    23
200    16
150    16
400    15
500    14
600    13
800    12
100     6
450     6
250     6
700     5
550     3
750     3
350     3
900     2
850     2
650     2
950     1
Name: count, dtype: int64
```

In [21]:
```
#plotting
plt.figure(figsize = (10,5))
ax = sns.barplot(data=data,palette='muted')
for i in ax.containers :
  ax.bar_label(i,)
plt.xlabel("Approximate cost")
plt.ylabel("Count")
plt.show()
```

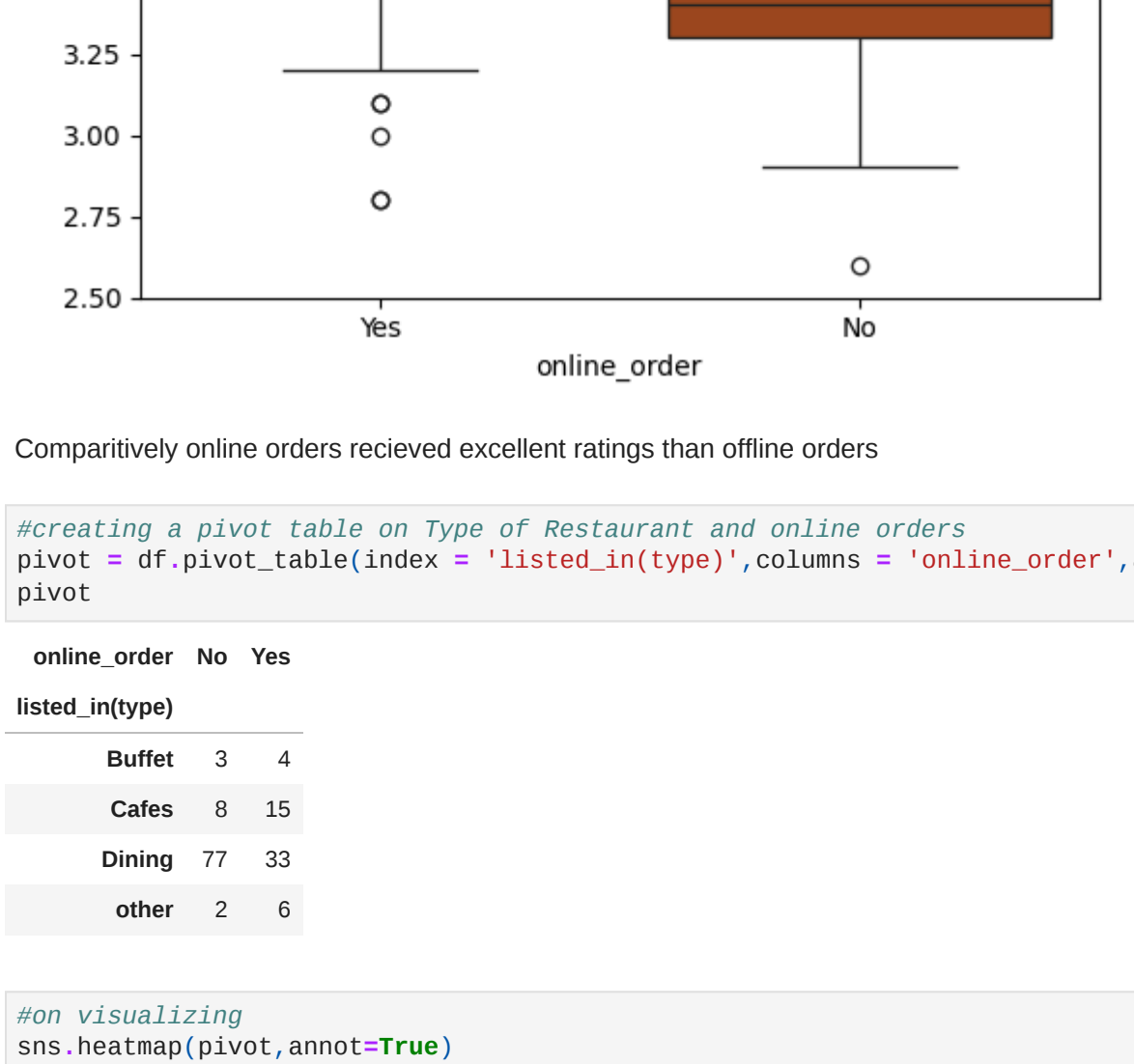


Majority of the couples prefer to choose the Restaurants with an approximate cost of 300 rupees

In [22]:
```
#to check if online orders have more ratings than offline orders
df['online_order'].value_counts()
```

Out[22]:
```
online_order
No     90
Yes    58
Name: count, dtype: int64
```

In [23]:
```
sns.boxplot(data=df,x = 'online_order',y='rate',palette='dark')
plt.show()
```



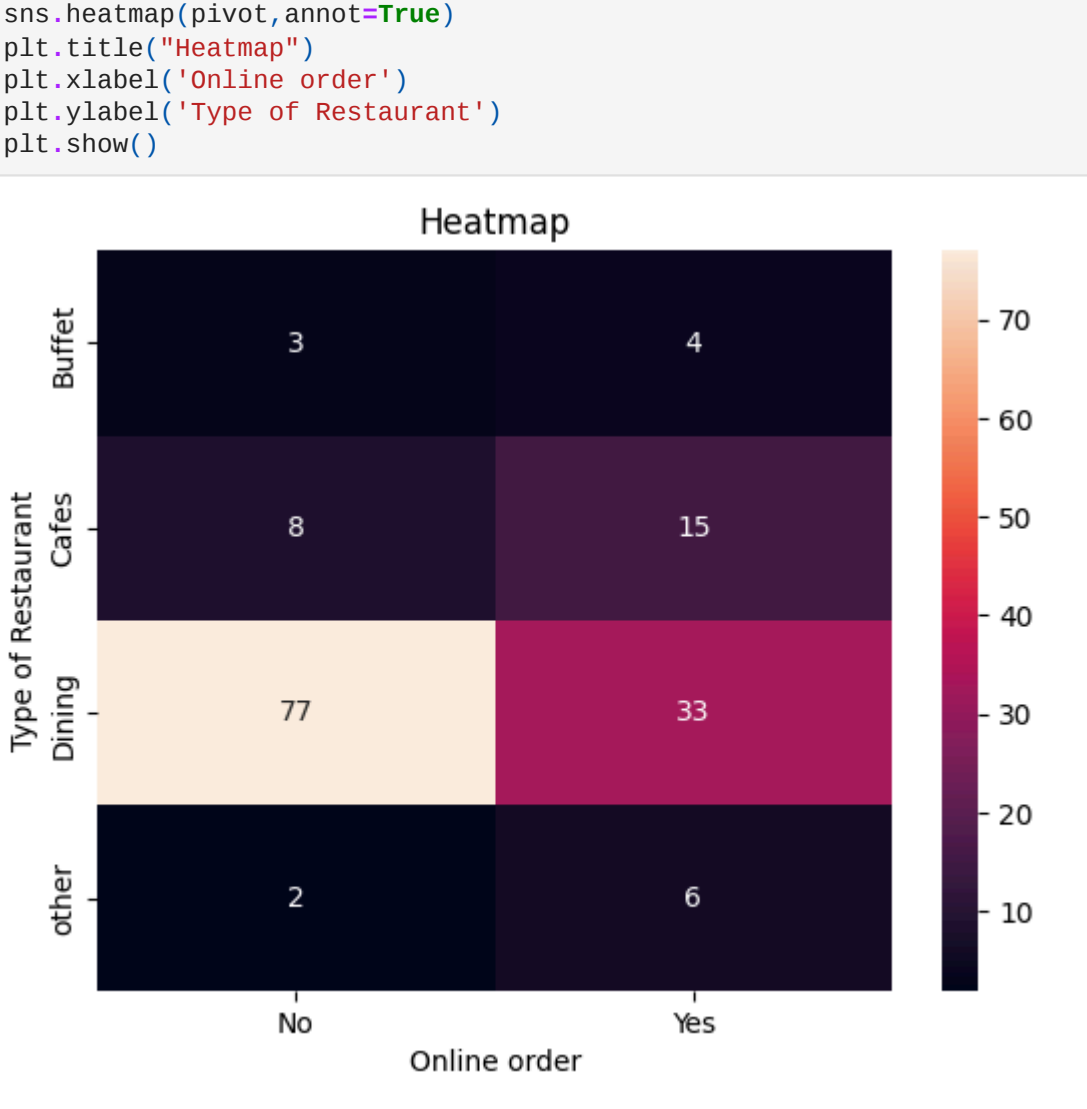
Comparitively online orders recieved excellent ratings than offline orders

In [24]:
```
#creating a pivot table on Type of Restaurant and online orders
pivot = df.pivot_table(index = 'listed_in(type)',columns = 'online_order',aggfunc = 'size')
pivot
```

Out[24]:

online_order	No	Yes
listed_in(type)		
Buffet	3	4
Cafes	8	15
Dining	77	33
other	2	6

In [25]:
```
#on visualizing
sns.heatmap(pivot,annot=True)
plt.title("Heatmap")
plt.xlabel('Online order')
plt.ylabel('Type of Restaurant')
plt.show()
```



Dining restaurants primarily accept offline orders, whereas cafes primarily receive online orders. This suggests that clients prefer to place orders in person at restaurants, but prefer online ordering at cafes.

In [ ]: