>importing necessary python libraries.

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

> Creating the data frame.

In [5]:
```
dataframe = pd.read_csv("zomato_data.csv")
print(dataframe.head())
```

```
                  name online_order book_table   rate  votes  \
0                Jalsa          Yes        Yes  4.1/5    775
1        Spice Elephant          Yes         No  4.1/5    787
2       San Churro Cafe          Yes         No  3.8/5    918
3  Addhuri Udupi Bhojana           No         No  3.7/5     88
4          Grand Village           No         No  3.8/5    166

   approx_cost(for two people) listed_in(type)
0                          800          Buffet
1                          800          Buffet
2                          800          Buffet
3                          300          Buffet
4                          600          Buffet
```

>Data Cleaning and Preparation

In [18]:
```
def handleRate(value):
    value=str(value).split('/')
    value=value[0];
    return float(value)

dataframe['rate']=dataframe['rate'].apply(handleRate)
print(dataframe.head())
```

```
                  name online_order book_table  rate  votes  \
0                Jalsa          Yes        Yes   4.1    775
1        Spice Elephant          Yes         No   4.1    787
2       San Churro Cafe          Yes         No   3.8    918
3  Addhuri Udupi Bhojana           No         No   3.7     88
4          Grand Village           No         No   3.8    166

   approx_cost(for two people) listed_in(type)
0                          800          Buffet
1                          800          Buffet
2                          800          Buffet
3                          300          Buffet
4                          600          Buffet
```

>Getting summary of the dataframe use df.info().

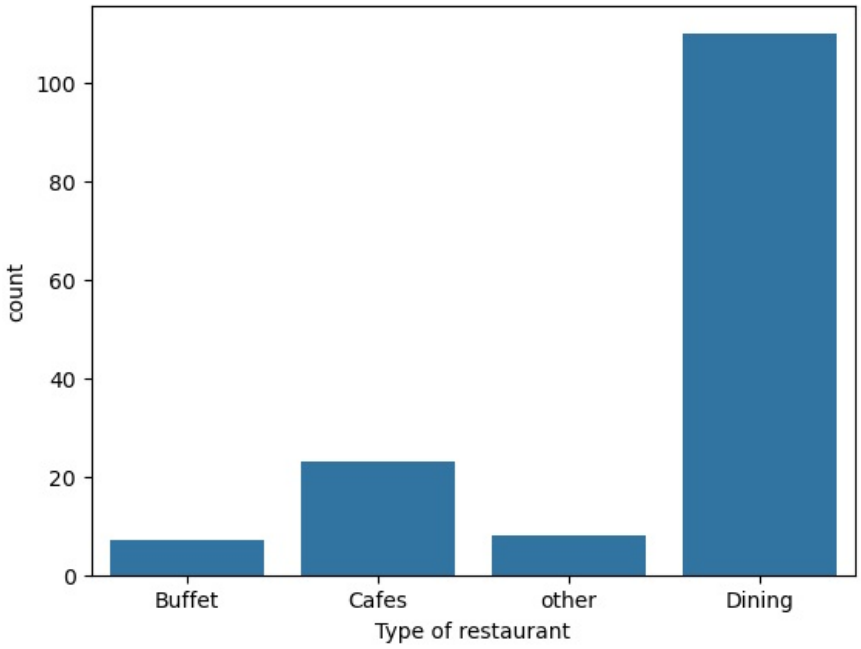
In [20]:
```
dataframe.info()
```

```
<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
```

>Exploring Restaurant Types

In [27]:
```
sns.countplot(x=dataframe['listed_in(type)'])
plt.xlabel("Type of restaurant")
```

Out[27]: Text(0.5, 0, 'Type of restaurant')

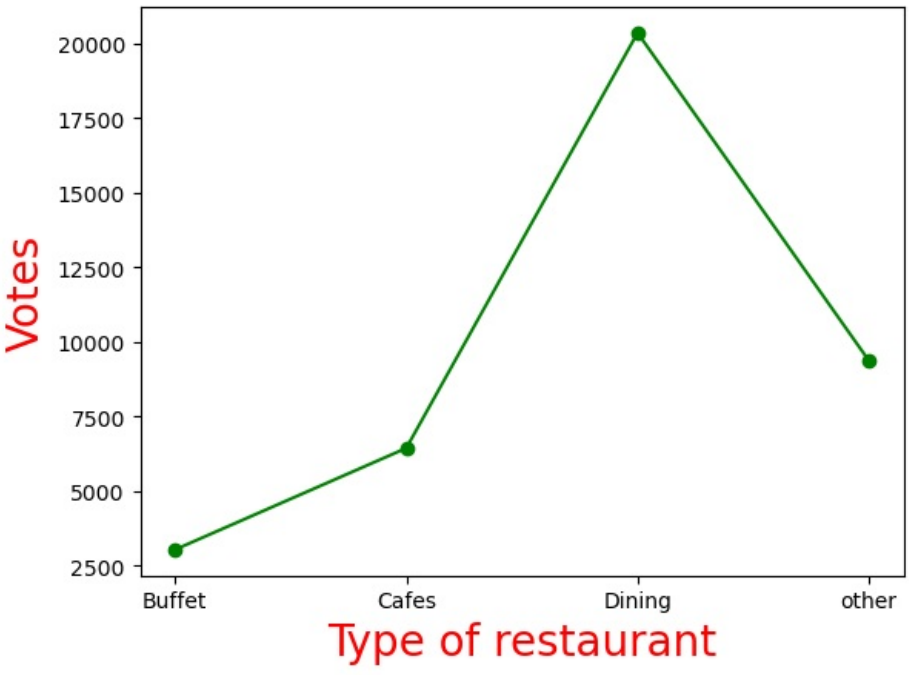


>Votes by Restaurant Type

Here we get the count of votes for each category.

In [30]:
```
grouped_data = dataframe.groupby('listed_in(type)')['votes'].sum()
result = pd.DataFrame({'votes': grouped_data})
plt.plot(result, c='green', marker='o')
plt.xlabel('Type of restaurant', c='red', size=20)
plt.ylabel('Votes', c='red', size=20)
```

Out[30]: Text(0, 0.5, 'Votes')



>Identifing the Most Voted Restaurant

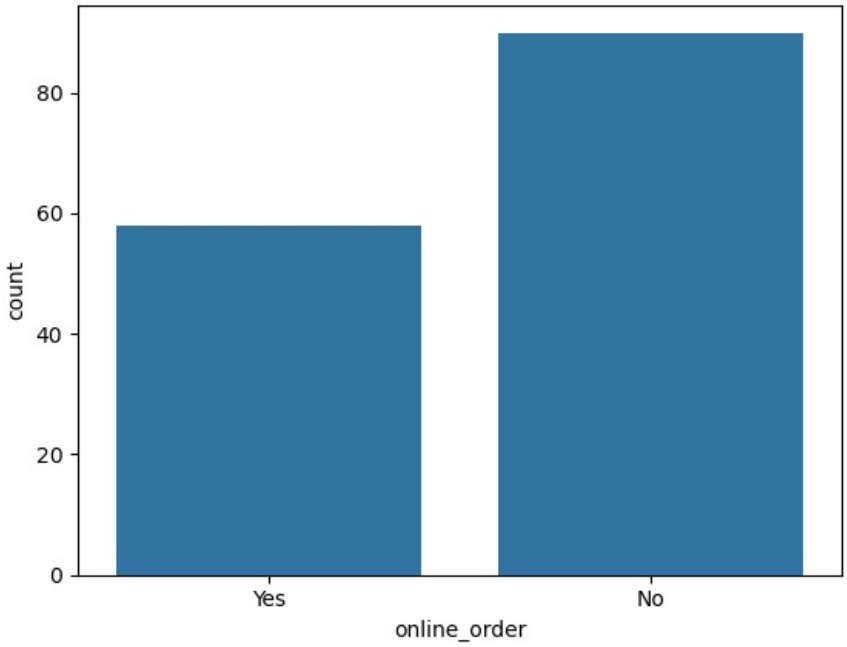
In [34]:
```
max_votes = dataframe['votes'].max()
restaurant_with_max_votes = dataframe.loc[dataframe['votes'] == max_votes, 'name']

print('Restaurant(s) with the maximum votes:')
print(restaurant_with_max_votes)
```

```
Restaurant(s) with the maximum votes:
38    Empire Restaurant
Name: name, dtype: object
```

>Online Order Availability

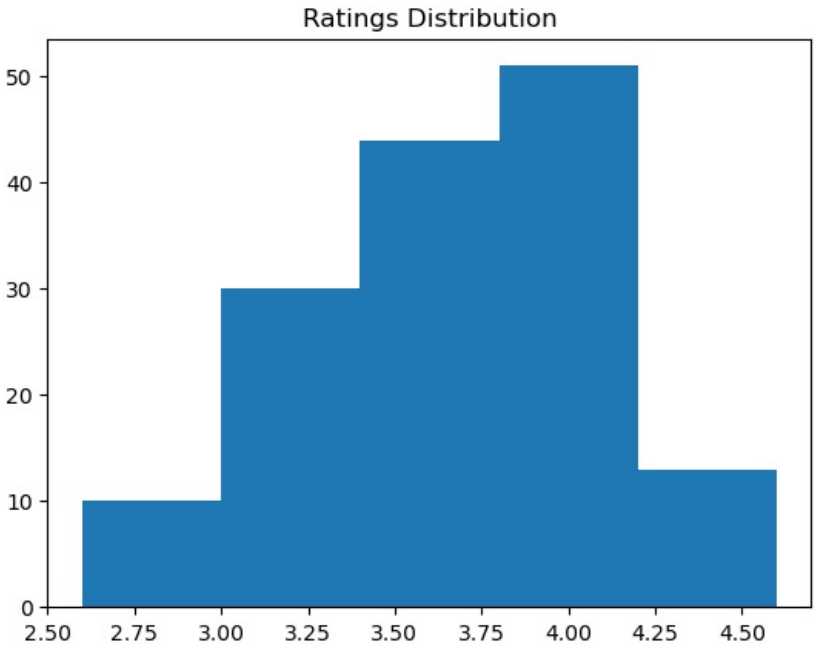
In [37]:
```
sns.countplot(x=dataframe['online_order'])
```

Out[37]:
```
<Axes: xlabel='online_order', ylabel='count'>
```



>Analyzing the Ratings

In [40]:
```
plt.hist(dataframe['rate'],bins=5)
plt.title('Ratings Distribution')
plt.show()
```

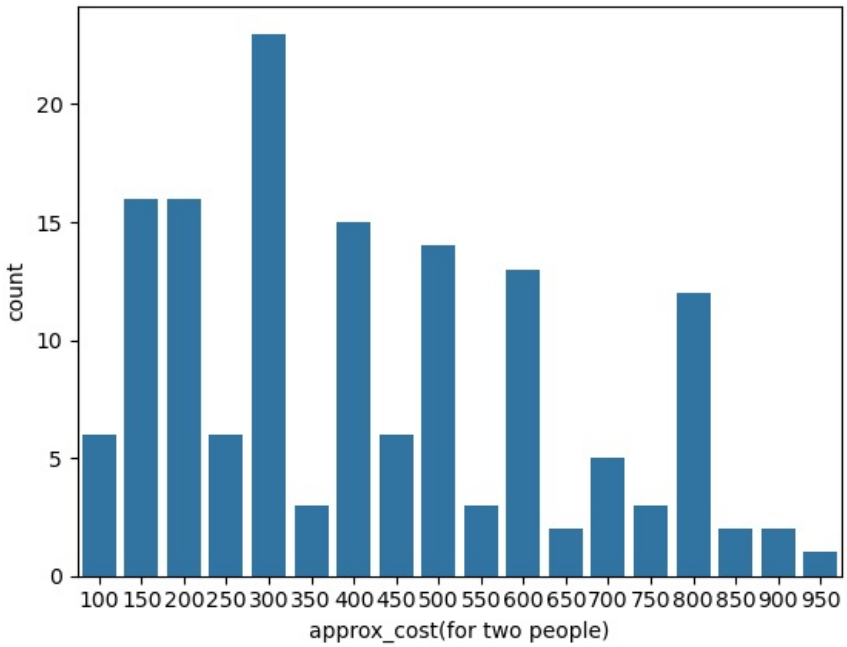


here we can observe that the majority of restaurants received ratings ranging from 3.5 to 4.

>Approximate Cost for Couples

In [44]:
```
couple_data=dataframe['approx_cost(for two people)']
sns.countplot(x=couple_data)
```

Out[44]:
```
<Axes: xlabel='approx_cost(for two people)', ylabel='count'>
```

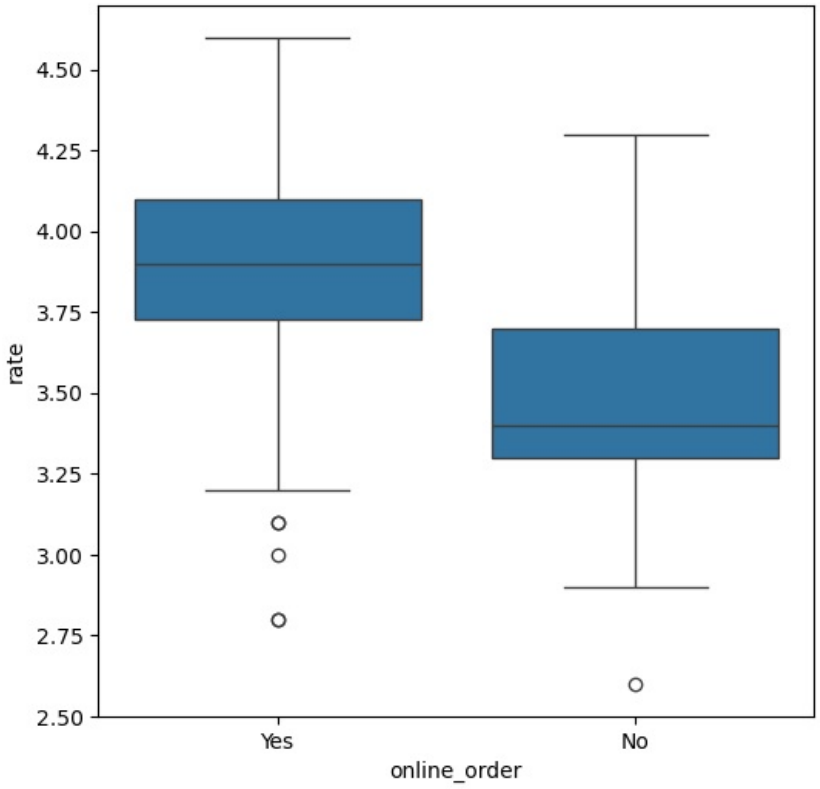


As per calculations Average cost for couple (two people) is Rs.300

>Ratings Comparison - Online vs Offline Orders

In [48]:
```
plt.figure(figsize = (6,6))
sns.boxplot(x = 'online_order', y = 'rate', data = dataframe)
```

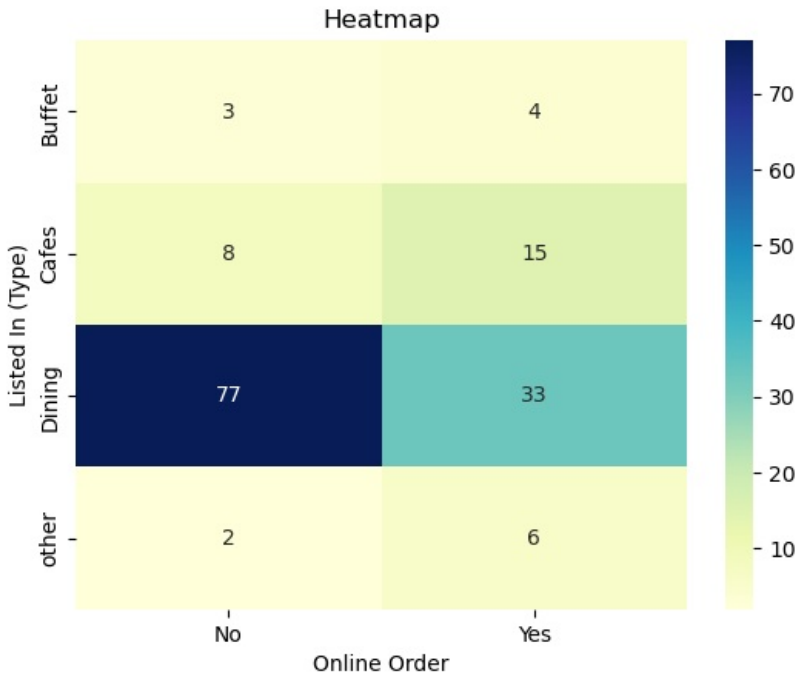
Out[48]:
```
<Axes: xlabel='online_order', ylabel='rate'>
```



from the above figure we conclude that Offline orders received lower ratings in comparison to online orders which obtained excellent ratings.

>Order Mode Preferences by Restaurant Type

In [52]:
```
pivot_table = dataframe.pivot_table(index='listed_in(type)', columns='online_order', aggfunc='size', fill_value=
sns.heatmap(pivot_table, annot=True, cmap='YlGnBu', fmt='d')
plt.title('Heatmap')
plt.xlabel('Online Order')
plt.ylabel('Listed In (Type)')
plt.show()
```



the final conclusion is 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.

Top 5 Most Voted Restaurants

In [9]:
```
top_voted = dataframe[['name', 'votes']].sort_values(by='votes', ascending=False).head(5)
print(top_voted)
```

```
                 name  votes
38  Empire Restaurant   4884
86      Meghana Foods   4401
7             Onesta   2556
44            Onesta   2556
65       Kabab Magic   1720
```

Restaurants Offering Both Online Order and Table Booking

In [13]:
```
combo = dataframe[(dataframe['online_order'] == 'Yes') & (dataframe['book_table'] == 'Yes')]
print(combo[['name', 'online_order', 'book_table']].head())
```

```
                name online_order book_table
0              Jalsa          Yes        Yes
7             Onesta          Yes        Yes
11      Cafe Shuffle          Yes        Yes
12  The Coffee Shack          Yes        Yes
44            Onesta          Yes        Yes
```

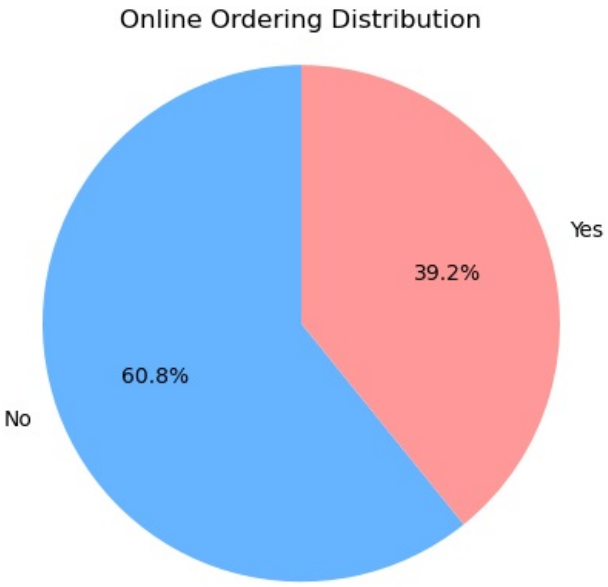
Unique Restaurant Types and Their Count

In [20]:
```
unique_types = dataframe['listed_in(type)'].value_counts()
print(unique_types)
```

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

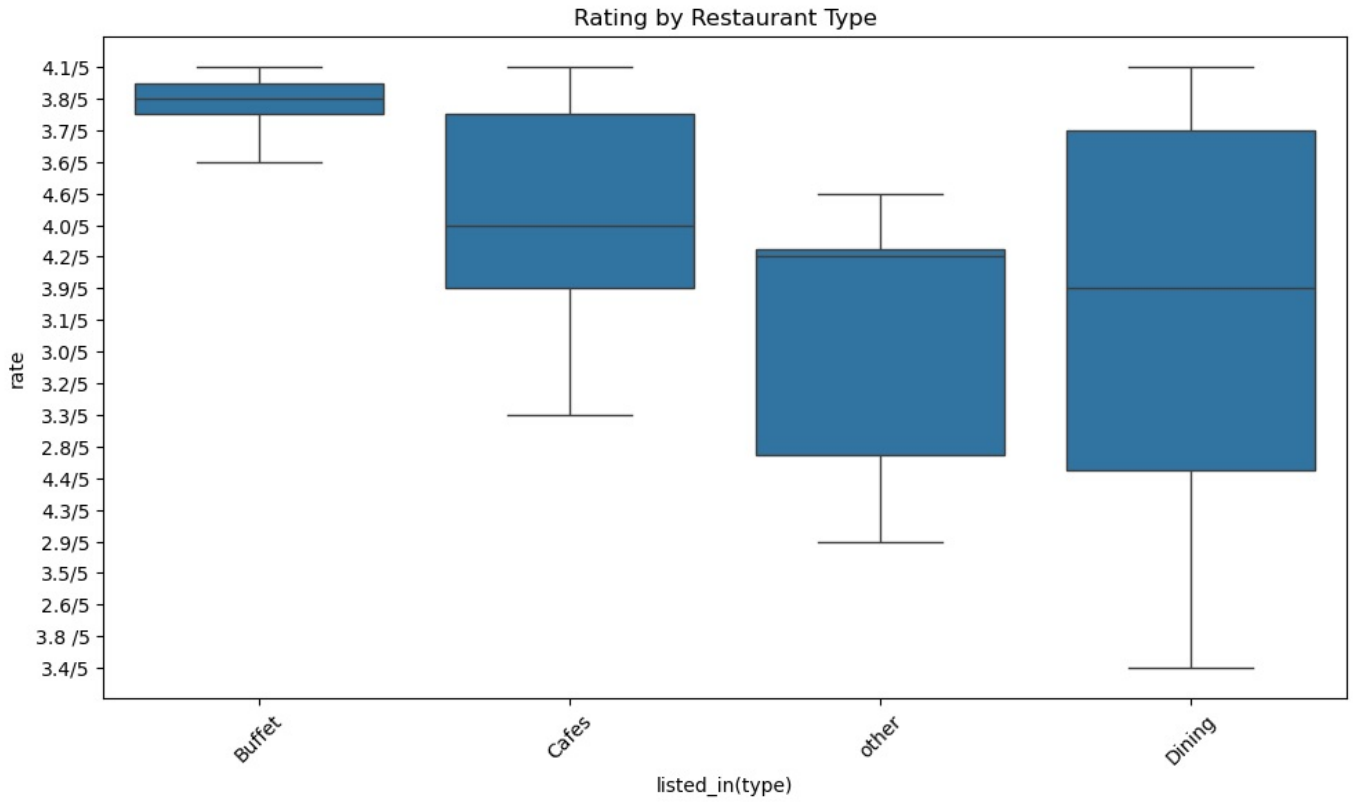
Pie Chart: Distribution of Online Ordering

In [29]:
```
online_counts = dataframe['online_order'].value_counts()
plt.pie(online_counts, labels=online_counts.index, autopct='%1.1f%%', startangle=90, colors=['#66b3ff','#ff9999
plt.title("Online Ordering Distribution")
plt.axis('equal')
plt.show()
```



Boxplot: Rating Distribution by Restaurant Type

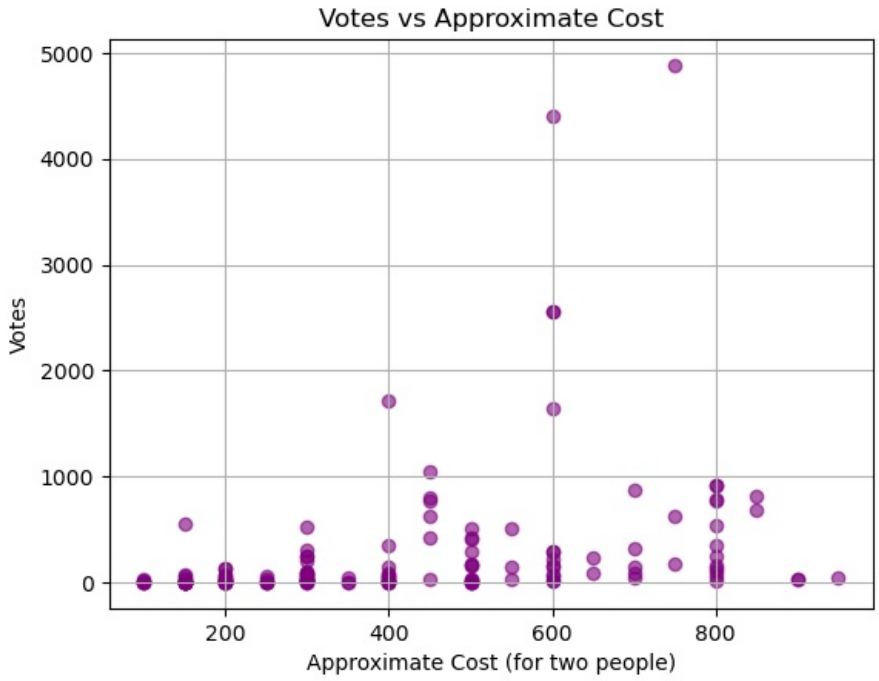
In [36]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='listed_in(type)', y='rate', data=dataframe)
plt.xticks(rotation=45)
plt.title("Rating by Restaurant Type")
plt.tight_layout()
plt.show()
```



Scatter Plot: Votes vs Cost

In [39]:
```
plt.scatter(dataframe['approx_cost(for two people)'], dataframe['votes'], alpha=0.6, color='purple')
plt.title("Votes vs Approximate Cost")
plt.xlabel("Approximate Cost (for two people)")
plt.ylabel("Votes")
plt.grid(True)
```

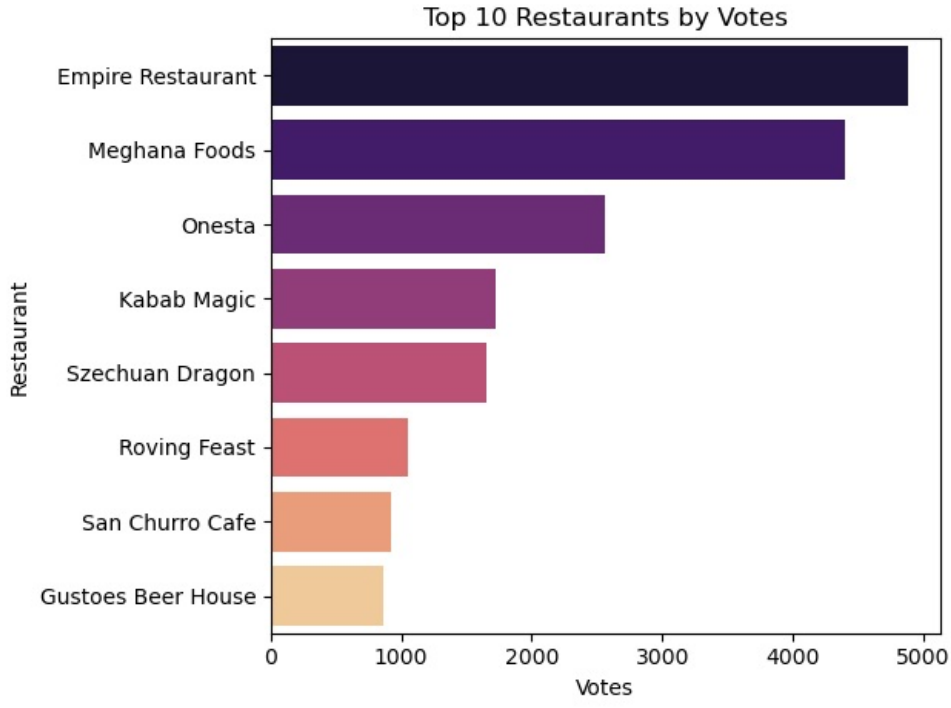
```
plt.show()
```



Top 10 Restaurants by Votes (Bar Plot)

In [58]:
```
top_voted = dataframe.sort_values(by='votes', ascending=False).head(10)
sns.barplot(x='votes', y='name', data=top_voted, palette='magma')
plt.title("Top 10 Restaurants by Votes")
plt.xlabel("Votes")
plt.ylabel("Restaurant")
plt.tight_layout()
plt.show()
```

```
C:\Users\rohit\AppData\Local\Temp\ipykernel_28284\3907550076.py:2: FutureWarning:

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

  sns.barplot(x='votes', y='name', data=top_voted, palette='magma')
```



Violin Plot: Rating Distribution per Restaurant Type

In [61]:
```
plt.figure(figsize=(10, 6))
sns.violinplot(x='listed_in(type)', y='rate', data=dataframe)
plt.title("Rating Distribution by Restaurant Type")
```

```
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

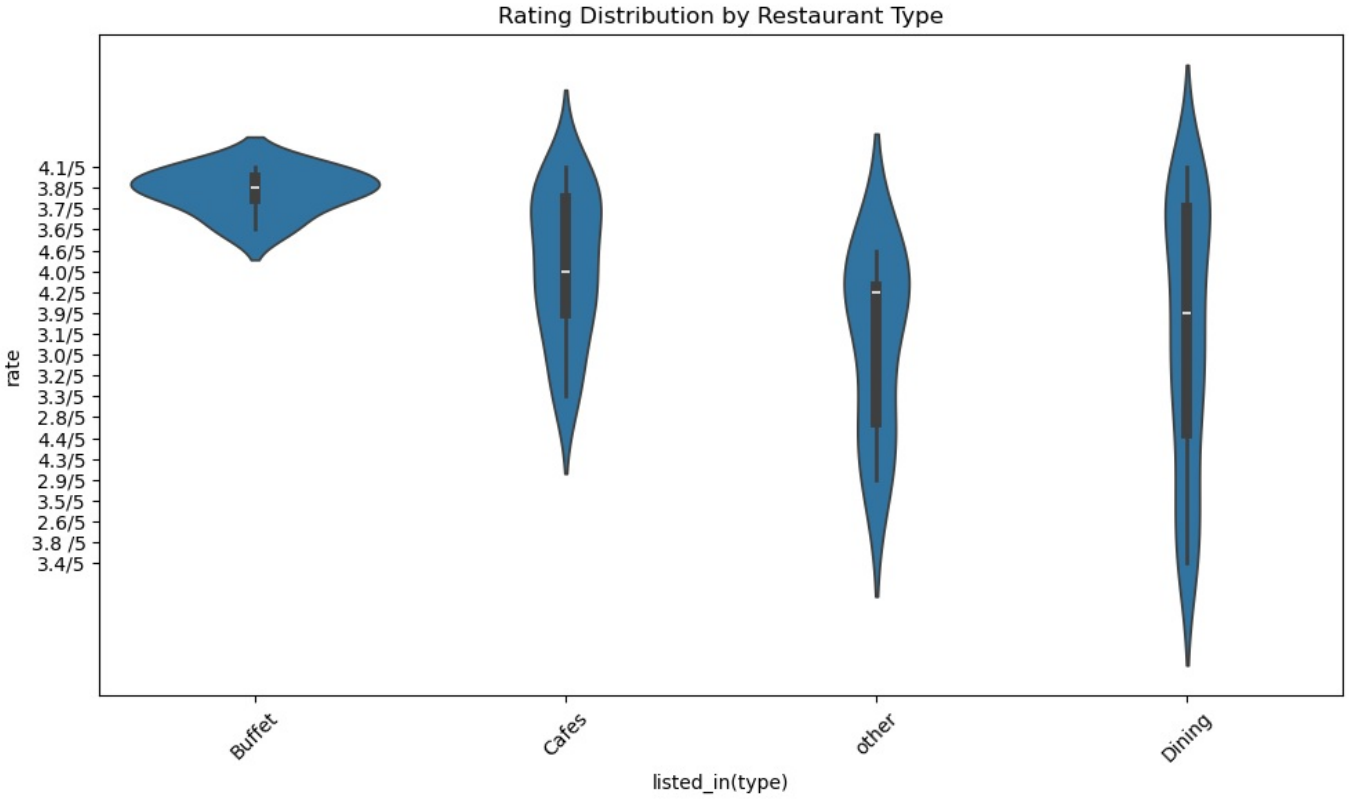
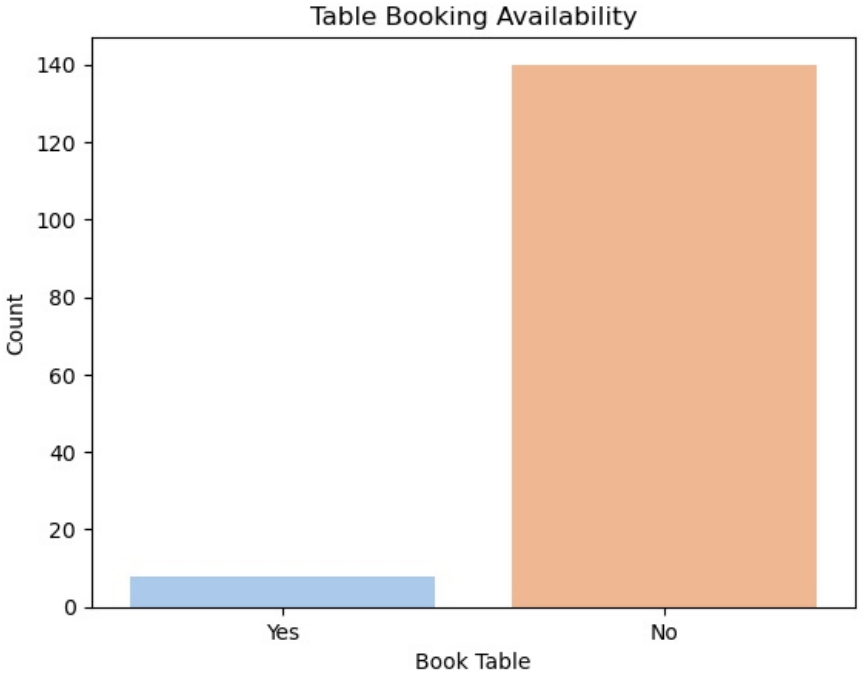


Table Booking Availability Countplot

In [64]:
```
sns.countplot(x='book_table', data=dataframe, palette='pastel')
plt.title("Table Booking Availability")
plt.xlabel("Book Table")
plt.ylabel("Count")
plt.show()
```

```
C:\Users\rohit\AppData\Local\Temp\ipykernel_28284\3465451720.py:1: 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='book_table', data=dataframe, palette='pastel')
```



Percentage of Online Ordering Restaurants

In [67]:
```
online_counts = dataframe['online_order'].value_counts(normalize=True) * 100
print("\nPercentage of Online vs Offline Order Availability:")
print(online_counts.round(2))
```

```
Percentage of Online vs Offline Order Availability:
online_order
No     60.81
Yes    39.19
Name: proportion, dtype: float64
```

Statistical Summary of Cost Column

In [69]:
```
print("\nCost Statistics:")
print(dataframe['approx_cost(for two people)'].describe())
```

```
Cost Statistics:
count    148.000000
mean     418.243243
std      223.085098
min      100.000000
25%      200.000000
50%      400.000000
75%      600.000000
max      950.000000
Name: approx_cost(for two people), dtype: float64
```

Restaurants with Maximum and Minimum Cost

In [79]:
```
max_cost = dataframe['approx_cost(for two people)'].max()
min_cost = dataframe['approx_cost(for two people)'].min()

max_cost_restaurants = dataframe[dataframe['approx_cost(for two people)'] == max_cost]
min_cost_restaurants = dataframe[dataframe['approx_cost(for two people)'] == min_cost]

print("\nMost Expensive Restaurants:")
print(max_cost_restaurants[['name', 'approx_cost(for two people)', 'rate']])

print("\nLeast Expensive Restaurants:")
print(min_cost_restaurants[['name', 'approx_cost(for two people)', 'rate']])
```

```
Most Expensive Restaurants:
                    name  approx_cost(for two people)   rate
97  Ayda Persian Kitchen                          950  3.7/5

Least Expensive Restaurants:
                          name  approx_cost(for two people)   rate
66        Namma Brahmin's Idli                          100  3.6/5
84                  Chill Out                          100  3.8/5
122               Coffee Bytes                          100  3.1/5
127              Ruchi Maayaka                          100  3.3/5
131  Foodlieious Multi Cuisine                          100  3.4/5
143           Melting Melodies                          100  3.3/5
```

Restaurants Having Same Cost but Different Ratings

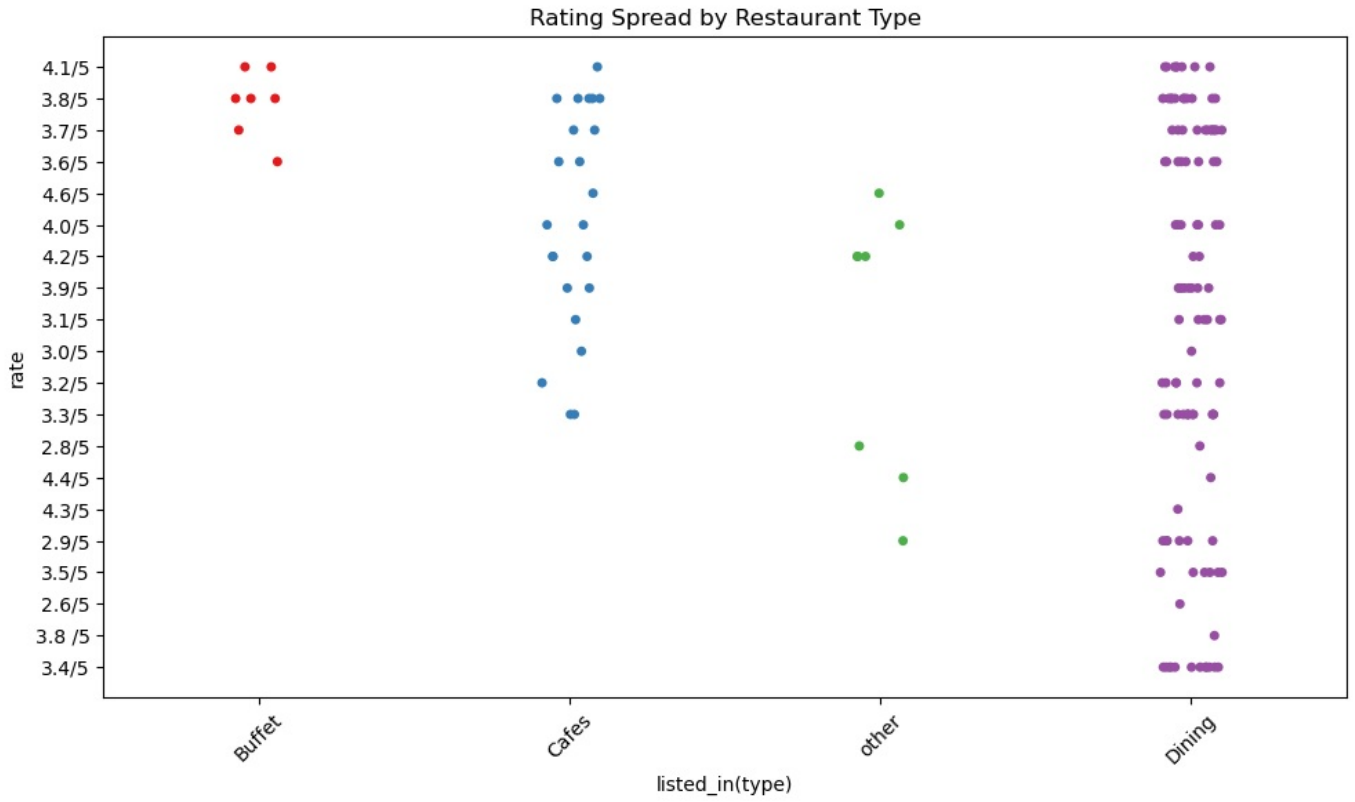
In [84]:
```
duplicate_cost = dataframe.groupby('approx_cost(for two people)').filter(lambda x: x['rate'].nunique() > 1)
print("\nRestaurants with Same Cost but Different Ratings:")
print(duplicate_cost[['name', 'approx_cost(for two people)', 'rate']].head(10))
```

```
Restaurants with Same Cost but Different Ratings:
                                          name  \
0                                        Jalsa
1                                Spice Elephant
2                               San Churro Cafe
3                          Addhuri Udupi Bhojana
4                                  Grand Village
5                                Timepass Dinner
6  Rosewood International Hotel - Bar & Restaurant
7                                       Onesta
8                                Penthouse Cafe
9                                     Smacznego

   approx_cost(for two people)   rate
0                          800  4.1/5
1                          800  4.1/5
2                          800  3.8/5
3                          300  3.7/5
4                          600  3.8/5
5                          600  3.8/5
6                          800  3.6/5
7                          600  4.6/5
8                          700  4.0/5
9                          550  4.2/5
```

Strip Plot: Rating vs Restaurant Type

In [92]:
```
plt.figure(figsize=(10,6))
sns.stripplot(x='listed_in(type)', y='rate', data=dataframe, jitter=True, palette='Set1')
plt.title("Rating Spread by Restaurant Type")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

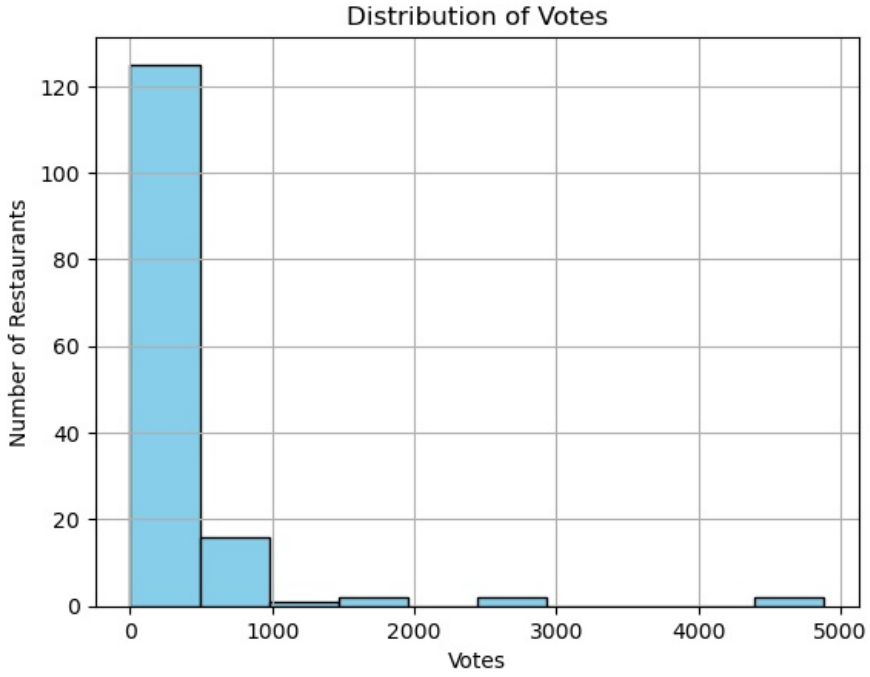
```
C:\Users\rohit\AppData\Local\Temp\ipykernel_28284\1191919607.py:2: FutureWarning:

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.stripplot(x='listed_in(type)', y='rate', data=dataframe, jitter=True, palette='Set1')
```



Histogram of Votes

In [95]:
```
plt.hist(dataframe['votes'], bins=10, color='skyblue', edgecolor='black')
plt.title("Distribution of Votes")
plt.xlabel("Votes")
plt.ylabel("Number of Restaurants")
plt.grid(True)
```

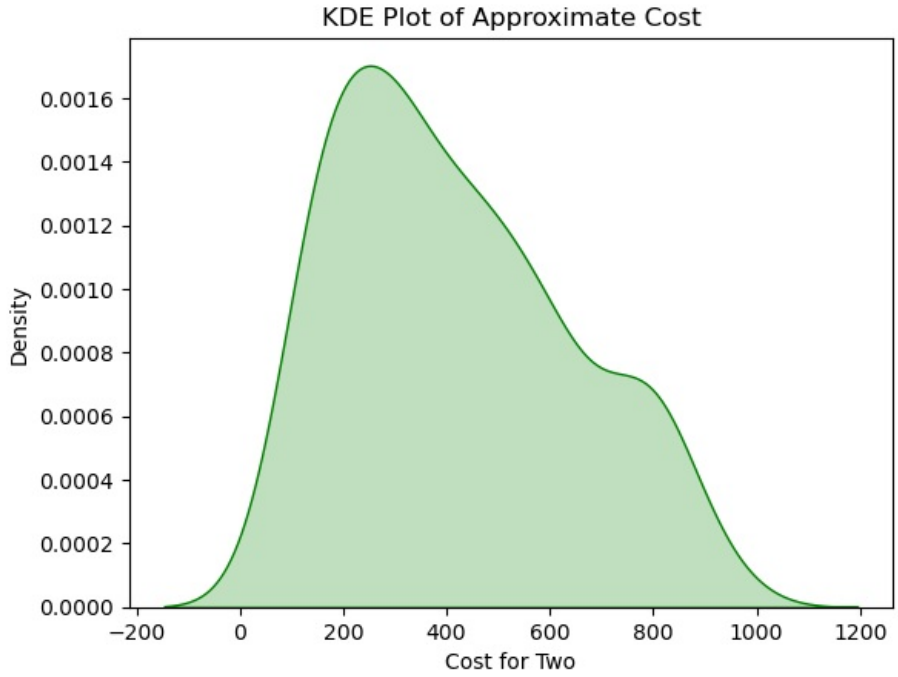
```
plt.show()
```



KDE Plot for Cost

In [98]:
```
sns.kdeplot(data=dataframe, x='approx_cost(for two people)', shade=True, color='green')
plt.title("KDE Plot of Approximate Cost")
plt.xlabel("Cost for Two")
plt.show()
```

```
C:\Users\rohit\AppData\Local\Temp\ipykernel_28284\1751678316.py:1: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data=dataframe, x='approx_cost(for two people)', shade=True, color='green')
```



In [ ]: