ZOMATO_DATA_ANALYSIS_EDA_PROJECT_VIVEK_CHAUHAN

The analysis is divided into four main parts:

1. Data understanding

2. Data cleaning (cleaning missing values, removing redundant columns etc.)

3. Data Analysis

4. Recommendations

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]:
```
# show all the details which is present in the dataset

pd.options.display.max_rows = 100
pd.options.display.max_columns = 111
```

In [3]:
```
# load and print the dataset

data = pd.read_csv("C:/Users/VIVEK CHAUHAN/Desktop/ZOMATO-EDA-PROJECT/zomato.csv")
data
```

Out[3]:

	url	address	name	online_order	book_table	rate	votes	phone
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	080 42297555\r\n+91 9743772233
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	080 41714161
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5	918	+91 9663487993
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7/5	88	+91 9620009302
4	https://www.zomato.com/bangalore/grand-village...	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.8/5	166	+91 8026612447\r\n+91 9901210005
...	...	...	...	...	...	...	...	...

	url	address	name	online_order	book_table	rate	votes	phone
51712	https://www.zomato.com/bangalore/best-brews-fo...	Four Points by Sheraton Bengaluru, 43/3, White...	Best Brews - Four Points by Sheraton Bengaluru...	No	No	3.6 /5	27	080 40301477
51713	https://www.zomato.com/bangalore/vinod-bar-and...	Number 10, Garudachar Palya, Mahadevapura, Whi...	Vinod Bar And Restaurant	No	No	NaN	0	+91 8197675843
51714	https://www.zomato.com/bangalore/plunge-sherat...	Sheraton Grand Bengaluru Whitefield Hotel & Co...	Plunge - Sheraton Grand Bengaluru Whitefield H...	No	No	NaN	0	NaN
51715	https://www.zomato.com/bangalore/chime-sherato...	Sheraton Grand Bengaluru Whitefield Hotel & Co...	Chime - Sheraton Grand Bengaluru Whitefield Ho...	No	Yes	4.3 /5	236	080 49652769
51716	https://www.zomato.com/bangalore/the-nest-the-...	ITPL Main Road, KIADB Export Promotion Industr...	The Nest - The Den Bengaluru	No	No	3.4 /5	13	+91 8071117272

51717 rows × 17 columns

Data Understanding

In [4]:
```
# print the top 5 rows from the dataset

data.head()
```

Out[4]:

	url	address	name	online_order	book_table	rate	votes	phone	loca
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	080 42297555\r\n+91 9743772233	Banasha
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	080 41714161	Banasha
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5	918	+91 9663487993	Banasha
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7/5	88	+91 9620009302	Banasha
4	https://www.zomato.com/bangalore/grand-village...	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.8/5	166	+91 8026612447\r\n+91 9901210005	Basavana

In [5]:
```
# print last 10 rows from the dataset

data.tail()
```

Out[5]:

	url	address	name	online_order	book_table	rate	votes	phone	location
51712	https://www.zomato.com/bangalore/best-brews-fo...	Four Points by Sheraton Bengaluru, 43/3, White...	Best Brews - Four Points by Sheraton Bengaluru...	No	No	3.6 /5	27	080 40301477	Whitefield
51713	https://www.zomato.com/bangalore/vinod-bar-and...	Number 10, Garudachar Palya, Mahadevapura, Whi...	Vinod Bar And Restaurant	No	No	NaN	0	+91 8197675843	Whitefield
51714	https://www.zomato.com/bangalore/plunge-sherat...	Sheraton Grand Bengaluru Whitefield Hotel & Co...	Plunge - Sheraton Grand Bengaluru Whitefield H...	No	No	NaN	0	NaN	Whitefield
51715	https://www.zomato.com/bangalore/chime-sherato...	Sheraton Grand Bengaluru Whitefield Hotel & Co...	Chime - Sheraton Grand Bengaluru Whitefield Ho...	No	Yes	4.3 /5	236	080 49652769	ITPL Main Road, Whitefield
51716	https://www.zomato.com/bangalore/the-nest-the-...	ITPL Main Road, KIADB Export Promotion Industr...	The Nest - The Den Bengaluru	No	No	3.4 /5	13	+91 8071117272	ITPL Main Road, Whitefield

In [6]:
```
# print the information about the dataset
```

```
data.info(verbose=True, show_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51717 entries, 0 to 51716
Data columns (total 17 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   url                          51717 non-null  object
 1   address                      51717 non-null  object
 2   name                         51717 non-null  object
 3   online_order                 51717 non-null  object
 4   book_table                   51717 non-null  object
 5   rate                         43942 non-null  object
 6   votes                        51717 non-null  int64 
 7   phone                        50509 non-null  object
 8   location                     51696 non-null  object
 9   rest_type                    51490 non-null  object
 10  dish_liked                   23639 non-null  object
 11  cuisines                     51672 non-null  object
 12  approx_cost(for two people)  51371 non-null  object
 13  reviews_list                 51717 non-null  object
 14  menu_item                    51717 non-null  object
 15  listed_in(type)              51717 non-null  object
 16  listed_in(city)              51717 non-null  object
dtypes: int64(1), object(16)
memory usage: 6.7+ MB
```

In [7]:
```
# print overall statistics about the dataset

data.describe(include="all").T
```

Out[7]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
url	51717	51717	https://www.zomato.com/bangalore/jalsa-banasha...	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN
address	51717	11495	Delivery Only	128	NaN	NaN	NaN	NaN	NaN	NaN	NaN
name	51717	8792	Cafe Coffee Day	96	NaN	NaN	NaN	NaN	NaN	NaN	NaN
online_order	51717	2	Yes	30444	NaN	NaN	NaN	NaN	NaN	NaN	NaN
book_table	51717	2	No	45268	NaN	NaN	NaN	NaN	NaN	NaN	NaN
rate	43942	64	NEW	2208	NaN	NaN	NaN	NaN	NaN	NaN	NaN
votes	51717.0	NaN	NaN	NaN	283.697527	803.838853	0.0	7.0	41.0	198.0	16832.0
phone	50509	14926	080 43334321	216	NaN	NaN	NaN	NaN	NaN	NaN	NaN
location	51696	93	BTM	5124	NaN	NaN	NaN	NaN	NaN	NaN	NaN
rest_type	51490	93	Quick Bites	19132	NaN	NaN	NaN	NaN	NaN	NaN	NaN
dish_liked	23639	5271	Biryani	182	NaN	NaN	NaN	NaN	NaN	NaN	NaN
cuisines	51672	2723	North Indian	2913	NaN	NaN	NaN	NaN	NaN	NaN	NaN
approx_cost(for two people)	51371	70	300	7576	NaN	NaN	NaN	NaN	NaN	NaN	NaN
reviews_list	51717	22513	[]	7595	NaN	NaN	NaN	NaN	NaN	NaN	NaN
menu_item	51717	9098	[]	39617	NaN	NaN	NaN	NaN	NaN	NaN	NaN
listed_in(type)	51717	7	Delivery	25942	NaN	NaN	NaN	NaN	NaN	NaN	NaN
listed_in(city)	51717	30	BTM	3279	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [8]:
```
# print the all the column names which is present in the data dataset

data.columns
```

Out[8]:
```
Index(['url', 'address', 'name', 'online_order', 'book_table', 'rate', 'votes',
       'phone', 'location', 'rest_type', 'dish_liked', 'cuisines',
       'approx_cost(for two people)', 'reviews_list', 'menu_item',
       'listed_in(type)', 'listed_in(city)'],
      dtype='object')
```

Data Cleaning

In [9]:
```
# before data cleaning remove unwanted column so data-analysis is easy

drop_col = ["url","phone"]

new_data = data.drop(columns = drop_col,axis=1)
new_data
```

Out[9]:

	address	name	online_order	book_table	rate	votes	location	rest_type	dish_liked	cuisines	approx_cost(fo two people
0	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...	North Indian, Mughlai, Chinese	80
1	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	Banashankari	Casual Dining	Momos, Lunch Buffet, Chocolate Nirvana, Thai G...	Chinese, North Indian, Thai	80
2	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5	918	Banashankari	Cafe, Casual Dining	Churros, Cannelloni, Minestrone Soup, Hot Choc...	Cafe, Mexican, Italian	80
3	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7/5	88	Banashankari	Quick Bites	Masala Dosa	South Indian, North Indian	30
4	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.8/5	166	Basavanagudi	Casual Dining	Panipuri, Gol Gappe	North Indian, Rajasthani	60
...	...	...	...	...	...	...	...	...	...	...	.

	address	name	online_order	book_table	rate	votes	location	rest_type	dish_liked	cuisines	approx_cost(fo two people
51712	Four Points by Sheraton Bengaluru, 43/3, White...	Best Brews - Four Points by Sheraton Bengaluru...	No	No	3.6 /5	27	Whitefield	Bar	NaN	Continental	1,50(
51713	Number 10, Garudachar Palya, Mahadevapura, Whi...	Vinod Bar And Restaurant	No	No	NaN	0	Whitefield	Bar	NaN	Finger Food	60(
51714	Sheraton Grand Bengaluru Whitefield Hotel & Co...	Plunge - Sheraton Grand Bengaluru Whitefield H...	No	No	NaN	0	Whitefield	Bar	NaN	Finger Food	2,00(
51715	Sheraton Grand Bengaluru Whitefield Hotel & Co...	Chime - Sheraton Grand Bengaluru Whitefield Ho...	No	Yes	4.3 /5	236	ITPL Main Road, Whitefield	Bar	Cocktails, Pizza, Buttermilk	Finger Food	2,50(
51716	ITPL Main Road, KIADB Export Promotion Industr...	The Nest - The Den Bengaluru	No	No	3.4 /5	13	ITPL Main Road, Whitefield	Bar, Casual Dining	NaN	Finger Food, North Indian, Continental	1,50(

51717 rows × 15 columns

In [10]:
```
# finding null values of each column

missing_info = missing_info = pd.DataFrame(new_data.isnull().sum().sort_values(ascending = False)).reset_index()
missing_info
```

Out[10]:

	index	0
0	dish_liked	28078
1	rate	7775
2	approx_cost(for two people)	346
3	rest_type	227
4	cuisines	45
5	location	21
6	address	0
7	name	0
8	online_order	0
9	book_table	0
10	votes	0
11	reviews_list	0
12	menu_item	0
13	listed_in(type)	0
14	listed_in(city)	0

In [11]:
```
# check which dish is more liked_so we replace it in null values

new_data["dish_liked"].value_counts()
```

Out[11]:
```
dish_liked
Biryani                                                                    182
Chicken Biryani                                                             73
Friendly Staff                                                              69
Waffles                                                                     68
Paratha                                                                     57
                                                                          ...
Butter Chicken, Shawarma Roll, Chicken Shawarama, Chicken Grill, Rolls, Al Faham Chicken, Biryani     1
Filter Coffee, Sandwich, Bonda, Vada, Masala Dosa, Salad, Aloo Curry                                  1
Burgers, Fries, Jumbo Royale Burger, Salads, Peri Peri Chicken Salad, Potato Wedges, Rolls            1
Chaat, Pav Bhaji, Raj Kachori, Buttermilk, Ajwaini Paratha, Tawa Pulav, Sev Puri                      1
Paratha, Dal Makhani, Lassi, Naan, Veg Thali, Chole, Kulcha                                           1
Name: count, Length: 5271, dtype: int64
```

In [12]:
```
# print the rate column

new_data["rate"]
```

Out[12]:
```
0         4.1/5
1         4.1/5
2         3.8/5
3         3.7/5
4         3.8/5
          ...
51712    3.6 /5
51713       NaN
51714       NaN
51715    4.3 /5
51716    3.4 /5
Name: rate, Length: 51717, dtype: object
```

In [13]:
```
# split the /5 column so we get just only rating

new_data["rate"] = new_data["rate"].str.split("/").str[0]
new_data["rate"]
```

Out[13]:
```
0        4.1
1        4.1
2        3.8
3        3.7
4        3.8
        ...
51712    3.6
51713     NaN
51714     NaN
51715    4.3
51716    3.4
Name: rate, Length: 51717, dtype: object
```

In [14]:
```
# now value checks

new_data["rate"].unique()
```

Out[14]:
```
array(['4.1', '3.8', '3.7', '3.6', '4.6', '4.0', '4.2', '3.9', '3.1',
       '3.0', '3.2', '3.3', '2.8', '4.4', '4.3', 'NEW', '2.9', '3.5', nan,
       '2.6', '3.8 ', '3.4', '4.5', '2.5', '2.7', '4.7', '2.4', '2.2',
       '2.3', '3.4 ', '-', '3.6 ', '4.8', '3.9 ', '4.2 ', '4.0 ', '4.1 ',
       '3.7 ', '3.1 ', '2.9 ', '3.3 ', '2.8 ', '3.5 ', '2.7 ', '2.5 ',
       '3.2 ', '2.6 ', '4.5 ', '4.3 ', '4.4 ', '4.9', '2.1', '2.0', '1.8',
       '4.6 ', '4.9 ', '3.0 ', '4.8 ', '2.3 ', '4.7 ', '2.4 ', '2.1 ',
       '2.2 ', '2.0 ', '1.8 '], dtype=object)
```

In [15]:
```
# check the missing percentage of the rate column

missing_percent = new_data["rate"].isnull().mean() * 100
missing_percent
```

Out[15]:
```
15.033741322969234
```

In [16]:
```
# check the null column missing percentage

new_data["rate"].mode()
```

Out[16]:
```
0    NEW
Name: rate, dtype: object
```

In [17]:
```
# first of all relace NEW word to nan values so we get better idea about that column

new_data["rate"] = new_data["rate"].replace(["NEW","-"],np.nan)
new_data["rate"]
```

Out[17]:
```
0         4.1
1         4.1
2         3.8
3         3.7
4         3.8
         ...
51712    3.6
51713     NaN
51714     NaN
51715    4.3
51716    3.4
Name: rate, Length: 51717, dtype: object
```

In [18]:
```
# new_data["rate"] check the column data type so we get better understand

new_data["rate"].dtypes
```

Out[18]:
```
dtype('O')
```

In [19]:
```
# change the new_data["rate"] object datatype to float datatype

new_data["rate"] = new_data["rate"].astype(float)
```

In [20]:
```
# check the average rates so we get better idea to fill nan values
new_data["rate"].mean()
```

Out[20]:
```
3.700448817952718
```

In [21]:
```
# check the average rates so we get better idea to fill nan values

new_data["rate"] = new_data["rate"].fillna(new_data["rate"].mean())
```

In [22]:
```
# print the columns
```

```
new_data["rate"]
```

Out[22]:
```
0        4.100000
1        4.100000
2        3.800000
3        3.700000
4        3.800000
           ...
51712    3.600000
51713    3.700449
51714    3.700449
51715    4.300000
51716    3.400000
Name: rate, Length: 51717, dtype: float64
```

In [23]:
```
# cross check so we aware about still null values is present in the column or not
new_data["rate"].isnull().sum()
```

Out[23]:
```
0
```

In [24]:
```
new_data.isnull().sum().sort_values(ascending = False)
```

Out[24]:
```
dish_liked                     28078
approx_cost(for two people)      346
rest_type                        227
cuisines                          45
location                          21
address                            0
name                               0
online_order                       0
book_table                         0
rate                               0
votes                              0
reviews_list                       0
menu_item                          0
listed_in(type)                    0
listed_in(city)                    0
dtype: int64
```

In [25]:
```
# let's check it one by one and fill the null values in the location column

location_mode = new_data["location"].mode()[0]
location_mode
```

Out[25]:
```
'BTM'
```

In [26]:
```
# fill the null values by mode value

new_data["location"] = new_data["location"].fillna(location_mode)
```

In [27]:
```
# print the column

new_data["location"]
```

Out[27]:
```
0                    Banashankari
1                    Banashankari
2                    Banashankari
3                    Banashankari
4                    Basavanagudi
                       ...
51712                   Whitefield
51713                   Whitefield
51714                   Whitefield
51715    ITPL Main Road, Whitefield
51716    ITPL Main Road, Whitefield
Name: location, Length: 51717, dtype: object
```

In [28]:
```
# cross check is there any null values is present or not
new_data["location"].isnull().sum()
```

Out[28]:
```
0
```

In [29]:
```
# so fill null values with most frequent values

cuisines_mode = new_data["cuisines"].mode()[0]
cuisines_mode
```

Out[29]: 'North Indian'

In [30]:
```
# so fill null values with most frequent values

new_data["cuisines"] = new_data["cuisines"].fillna(cuisines_mode)
new_data["cuisines"]
```

Out[30]:
```
0               North Indian, Mughlai, Chinese
1                  Chinese, North Indian, Thai
2                       Cafe, Mexican, Italian
3                    South Indian, North Indian
4                      North Indian, Rajasthani
                          ...
51712                              Continental
51713                              Finger Food
51714                              Finger Food
51715                              Finger Food
51716    Finger Food, North Indian, Continental
Name: cuisines, Length: 51717, dtype: object
```

In [31]:
```
# now cross chech still null values is present or not

new_data["cuisines"].isnull().sum()
```

Out[31]: 0

In [32]:
```
# let's fill the rest type with most frequent values
resttype_mode = new_data["rest_type"].mode()[0]
resttype_mode
```

Out[32]: 'Quick Bites'

In [33]:
```
# let's fill the rest type with most frequent values

new_data["rest_type"] = new_data["rest_type"].fillna(resttype_mode)
```

In [34]:
```
# check still nulll values is present or not

new_data["rest_type"].isnull().sum()
```

Out[34]:
```
0
```

In [35]:
```
# print the column
new_data["rest_type"]
```

Out[35]:
```
0              Casual Dining
1              Casual Dining
2        Cafe, Casual Dining
3                Quick Bites
4              Casual Dining
                 ...
51712                    Bar
51713                    Bar
51714                    Bar
51715                    Bar
51716     Bar, Casual Dining
Name: rest_type, Length: 51717, dtype: object
```

In [36]:
```
# let's count the null values in approx_cost(for two people) column

new_data["approx_cost(for two people)"].isnull().sum()
```

Out[36]:
```
346
```

In [37]:
```
# in this column contains , that's why object datatype and we can't convert it in float or int datatype

new_data["approx_cost(for two people)"] = new_data["approx_cost(for two people)"].str.replace(",", "", regex=True)
new_data["approx_cost(for two people)"]
```

Out[37]:
```
0         800
1         800
2         800
3         300
4         600
         ...
51712    1500
51713     600
51714    2000
51715    2500
51716    1500
Name: approx_cost(for two people), Length: 51717, dtype: object
```

In [38]:
```
# now convert the new_data["approx_cost(for two people)"] datatype to int

new_data["approx_cost(for two people)"] = new_data["approx_cost(for two people)"].astype(float)
```

In [39]:
```
# print the datatype of new_data["approx_cost(for two people)"] column

new_data["approx_cost(for two people)"].dtypes
```

Out[39]:
```
dtype('float64')
```

In [40]:
```
# check the mean values and
mode_cost = new_data["approx_cost(for two people)"].mode()[0]
mode_cost
```

Out[40]:
```
300.0
```

In [41]:
```
# check most of the customers amount

new_data["approx_cost(for two people)"].value_counts()
```

Out[41]:

approx_cost(for two people)	
300.0	7576
400.0	6562
500.0	4980
200.0	4857
600.0	3714
250.0	2959
800.0	2285
150.0	2066
700.0	1948
350.0	1763
1000.0	1637
450.0	1417
1200.0	993
100.0	993
1500.0	971
650.0	776
550.0	761
750.0	758
900.0	700
1300.0	516
1100.0	512
1400.0	473
2000.0	363
1600.0	266
1700.0	247
1800.0	203
850.0	166
3000.0	162
2500.0	146
2200.0	78
1900.0	70
2100.0	67
950.0	62
2800.0	45
4000.0	29
3500.0	25
120.0	24
2400.0	23
180.0	20

```
1350.0      18
3400.0      13
2300.0      11
2600.0      10
80.0        10
230.0       10
1250.0       9
40.0         8
50.0         8
130.0        8
1650.0       6
1450.0       5
199.0        4
330.0        4
4100.0       4
1050.0       4
2700.0       3
70.0         3
6000.0       2
4500.0       2
190.0        2
360.0        2
240.0        2
3200.0       2
140.0        2
560.0        1
60.0         1
5000.0       1
3700.0       1
469.0        1
160.0        1
Name: count, dtype: int64
```

In [42]:
```
# fill the null values by most frequent values
new_data["approx_cost(for two people)"] = new_data["approx_cost(for two people)"].fillna(mode_cost)
```

In [43]:
```
new_data["approx_cost(for two people)"]
```

Out[43]:
```
0         800.0
1         800.0
2         800.0
3         300.0
4         600.0
          ...
51712    1500.0
51713     600.0
51714    2000.0
51715    2500.0
51716    1500.0
Name: approx_cost(for two people), Length: 51717, dtype: float64
```

In [44]:
```
# let's check again is there any column that contains null vaules

new_data.isnull().sum().sort_values(ascending = False)
```

Out[44]:
```
dish_liked                     28078
address                            0
name                               0
online_order                       0
book_table                         0
rate                               0
votes                              0
location                           0
rest_type                          0
cuisines                           0
approx_cost(for two people)        0
reviews_list                       0
menu_item                          0
listed_in(type)                    0
listed_in(city)                    0
dtype: int64
```

Word Frequency Count

In [45]:
```
# fill the dish_liked column wih not mentined status
```

```
new_data["dish_liked"] = new_data["dish_liked"].fillna("Not mentioned")
new_data["dish_liked"]
```

Out[45]:
```
0        Pasta, Lunch Buffet, Masala Papad, Paneer Laja...
1        Momos, Lunch Buffet, Chocolate Nirvana, Thai G...
2        Churros, Cannelloni, Minestrone Soup, Hot Choc...
3                                             Masala Dosa
4                                      Panipuri, Gol Gappe
                              ...
51712                                      Not mentioned
51713                                      Not mentioned
51714                                      Not mentioned
51715                      Cocktails, Pizza, Buttermilk
51716                                      Not mentioned
Name: dish_liked, Length: 51717, dtype: object
```

In [46]:
```
# let's analyse the each and every words in the dish_liked column and then we are going to count it.
# import the counter library first

from collections import Counter
import nltk
from nltk.corpus import stopwords

# Download stopwords (only first time)
nltk.download('stopwords')

# Get English stopwords
stop_words = set(stopwords.words('english'))

all_dish_words = []

for dish in new_data['dish_liked']:
    all_dish_words.extend(dish.split())

# Filter out stopwords
filtered_words = [word for word in all_dish_words if word.lower() not in stop_words]

# Count the words
dish_word_freq = Counter(filtered_words)
```

```
# Print top 5 most common words
print(dish_word_freq.most_common(5))
```

```
[nltk_data] Downloading package stopwords to C:\Users\VIVEK
[nltk_data]     CHAUHAN\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
[('mentioned', 28078), ('Chicken', 10845), ('Biryani,', 6058), ('Pizza,', 3945), ('Chicken,', 3927)]
```

Here,you can see the 'Chicken','Biryani',"Pizza" is a most frequent words while placed order or you can say Most Favourite dishes of the Customers.

In [47]:
```
# let's analyse the each and every words in the cuisines column and then we are going to count it.
# import the counter library first

from collections import Counter
import nltk
from nltk.corpus import stopwords

# Download stopwords (only needed once)
nltk.download('stopwords')

# Get English stopwords
stop_words = set(stopwords.words('english'))

all_cuisines_words = []  # create an empty list

for dish in new_data['cuisines']:  # loop through cuisines column
    all_cuisines_words.extend(dish.split())

# Filter out stopwords
filtered_cuisines_words = [word for word in all_cuisines_words if word.lower() not in stop_words]

# Count the words
cuisines_word_freq = Counter(filtered_cuisines_words)

# Print top 5 most common words
print(cuisines_word_freq.most_common(5))
```

```
[('North', 21187), ('Indian,', 20157), ('Indian', 9774), ('South', 8664), ('Chinese,', 8469)]
```

```
[nltk_data] Downloading package stopwords to C:\Users\VIVEK
[nltk_data]     CHAUHAN\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

You can see clearly the winner is in the category of cuisines is 'North'.

In [48]:
```python
# let's analyse the each and every words in the menu_item column and then we are going to count it.
# import the counter library first
from collections import Counter
import nltk
from nltk.corpus import stopwords

# Download stopwords (only needed once)
nltk.download('stopwords')

# Get English stopwords
stop_words = set(stopwords.words('english'))

all_menu_item_words = []  # create an empty list

for dish in new_data['menu_item']:  # loop through menu_item column
    all_menu_item_words.extend(dish.split())

# Filter out stopwords
filtered_menu_item_words = [word for word in all_menu_item_words if word.lower() not in stop_words]

# Count the words
menu_item_word_freq = Counter(filtered_menu_item_words)

# Print top 5 most common words
print(menu_item_word_freq.most_common(5))
```

```
[nltk_data] Downloading package stopwords to C:\Users\VIVEK
[nltk_data]     CHAUHAN\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[("'Chicken", 132408), ("'Veg", 77055), ("Rice',", 66644), ('Chicken', 65715), ('Fried', 50917)]
```

You can see clearly the most favourite menu item is 'Chicken'.

In [49]:
```
# let's analyse the each and every words in the location column and then we are going to count it so we get in which place mos
# import the counter library first

from collections import Counter
import nltk
from nltk.corpus import stopwords

# Download stopwords (only needed once)
nltk.download('stopwords')

# Get English stopwords
stop_words = set(stopwords.words('english'))

all_location_words = []  # create an empty list

for dish in new_data['location']:  # loop through location column
    all_location_words.extend(dish.split())

# Filter out stopwords
filtered_location_words = [word for word in all_location_words if word.lower() not in stop_words]

# Count the words
location_word_freq = Counter(filtered_location_words)

# Print top 5 most common words
print(location_word_freq.most_common(5))
```

```
[('Road', 9301), ('Koramangala', 7782), ('Block', 7734), ('BTM', 5145), ('Nagar', 4833)]
```

```
[nltk_data] Downloading package stopwords to C:\Users\VIVEK
[nltk_data]     CHAUHAN\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

You can see the most of the placed order location is 'Koramangala',& second position is 'BTM'.

In [50]:
```
# let's analyse the each and every words in the address column and then we are going to count it so we get in which place most

# import the counter library first

from collections import Counter
import nltk
from nltk.corpus import stopwords

# Download stopwords (only needed once)
nltk.download('stopwords')

# Get English stopwords
stop_words = set(stopwords.words('english'))

all_address_words = []  # create an empty list

for dish in new_data['address']:  # loop through address column
    all_address_words.extend(dish.split())

# Filter out stopwords
filtered_address_words = [word for word in all_address_words if word.lower() not in stop_words]

# Count the words
address_word_freq = Counter(filtered_address_words)

# Print top 5 most common words
print(address_word_freq.most_common(5))
```

```
[('Bangalore', 48442), ('Road,', 35506), ('Block,', 11901), ('Main', 10447), ('1st', 8996)]
```

```
[nltk_data] Downloading package stopwords to C:\Users\VIVEK
[nltk_data]     CHAUHAN\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Most of the order placed location is 'Bangalore' cause whole the dataset belongs to Bangalore.

In [51]:
```
# now again cross check is there any null values is present in any columns
```

```
new_data.isnull().sum()
```

Out[51]:
```
address                      0
name                         0
online_order                 0
book_table                   0
rate                         0
votes                        0
location                     0
rest_type                    0
dish_liked                   0
cuisines                     0
approx_cost(for two people)  0
reviews_list                 0
menu_item                    0
listed_in(type)              0
listed_in(city)              0
dtype: int64
```

In [52]:
```
# print all the column names of our new_data dataset

new_data.columns
```

Out[52]:
```
Index(['address', 'name', 'online_order', 'book_table', 'rate', 'votes',
       'location', 'rest_type', 'dish_liked', 'cuisines',
       'approx_cost(for two people)', 'reviews_list', 'menu_item',
       'listed_in(type)', 'listed_in(city)'],
      dtype='object')
```

In [53]:
```
# now again check the datatypes so we get easy to understand and smooth analysis

new_data.dtypes
```

Out[53]:
```
address                       object
name                          object
online_order                  object
book_table                    object
rate                         float64
votes                          int64
location                      object
rest_type                     object
dish_liked                    object
cuisines                      object
approx_cost(for two people)  float64
reviews_list                  object
menu_item                     object
listed_in(type)               object
listed_in(city)               object
dtype: object
```

In [54]:
```
# print the top 5 rows of the dataset

new_data.head()
```

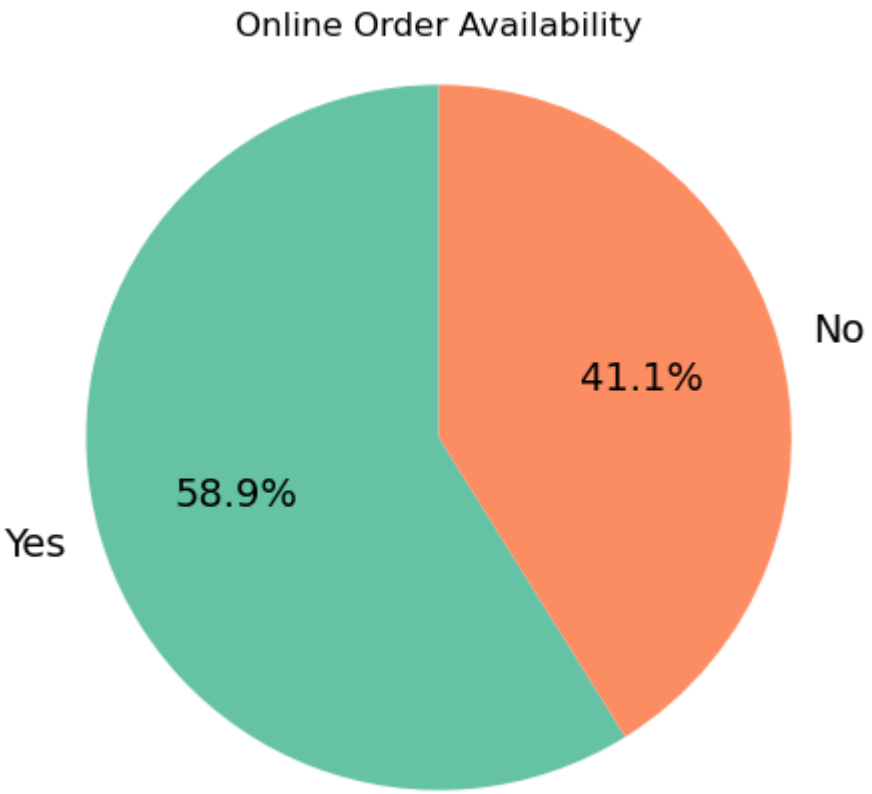
Out[54]:

	address	name	online_order	book_table	rate	votes	location	rest_type	dish_liked	cuisines	approx_cost(for two people)	reviews_
0	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...	North Indian, Mughlai, Chinese	800.0	[('Ra 'RATED\ beau place t
1	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1	787	Banashankari	Casual Dining	Momos, Lunch Buffet, Chocolate Nirvana, Thai G...	Chinese, North Indian, Thai	800.0	[('Ra 'RATE Had b here d
2	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8	918	Banashankari	Cafe, Casual Dining	Churros, Cannelloni, Minestrone Soup, Hot Choc...	Cafe, Mexican, Italian	800.0	[('Ra "RATE Ambienc not tha
3	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7	88	Banashankari	Quick Bites	Masala Dosa	South Indian, North Indian	300.0	[('Ra "RATE Great f prop
4	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.8	166	Basavanagudi	Casual Dining	Panipuri, Gol Gappe	North Indian, Rajasthani	600.0	[('Ra 'RATE Very g restau

Data Analysis

Uni-Variate-Analysis

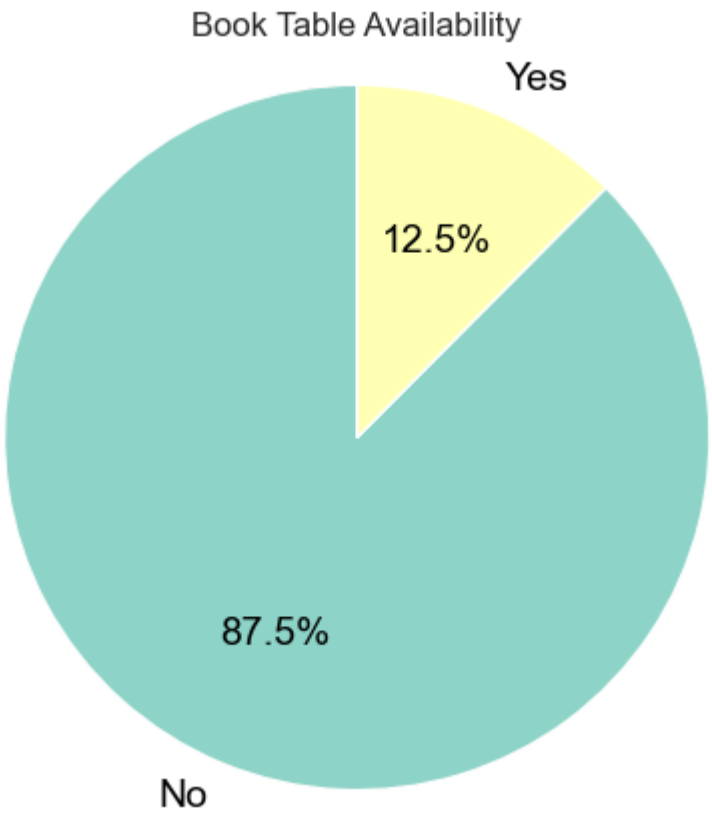
In [55]:
```
online_order_counts = new_data['online_order'].value_counts()

plt.figure(figsize=(5,5))
plt.pie(online_order_counts, labels=online_order_counts.index, autopct='%1.1f%%', startangle=90, colors=plt.cm.Set2.colors,tex
plt.title("Online Order Availability")
plt.axis('equal')  # Ensure the pie chart is circular
plt.show()
```



Most of the orders are in online mode.

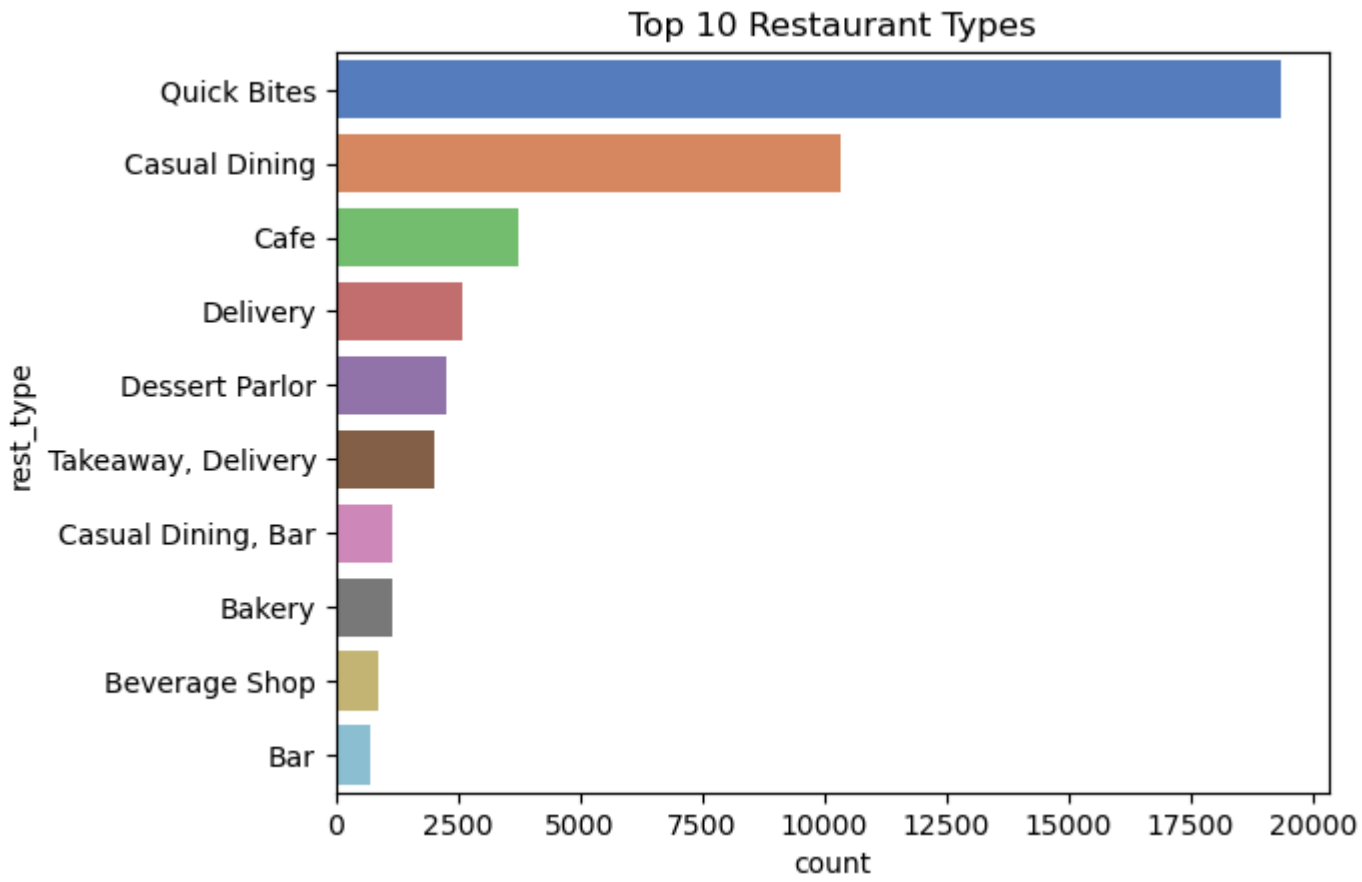
In [101…

```
book_table_counts = new_data['book_table'].value_counts()

plt.figure(figsize=(5,5))
plt.pie(book_table_counts, labels=book_table_counts.index, autopct='%1.1f%%', startangle=90, colors=plt.cm.Set3.colors,textpro
plt.title("Book Table Availability")
plt.axis('equal')  # Ensure the pie chart is circular
plt.show()
```



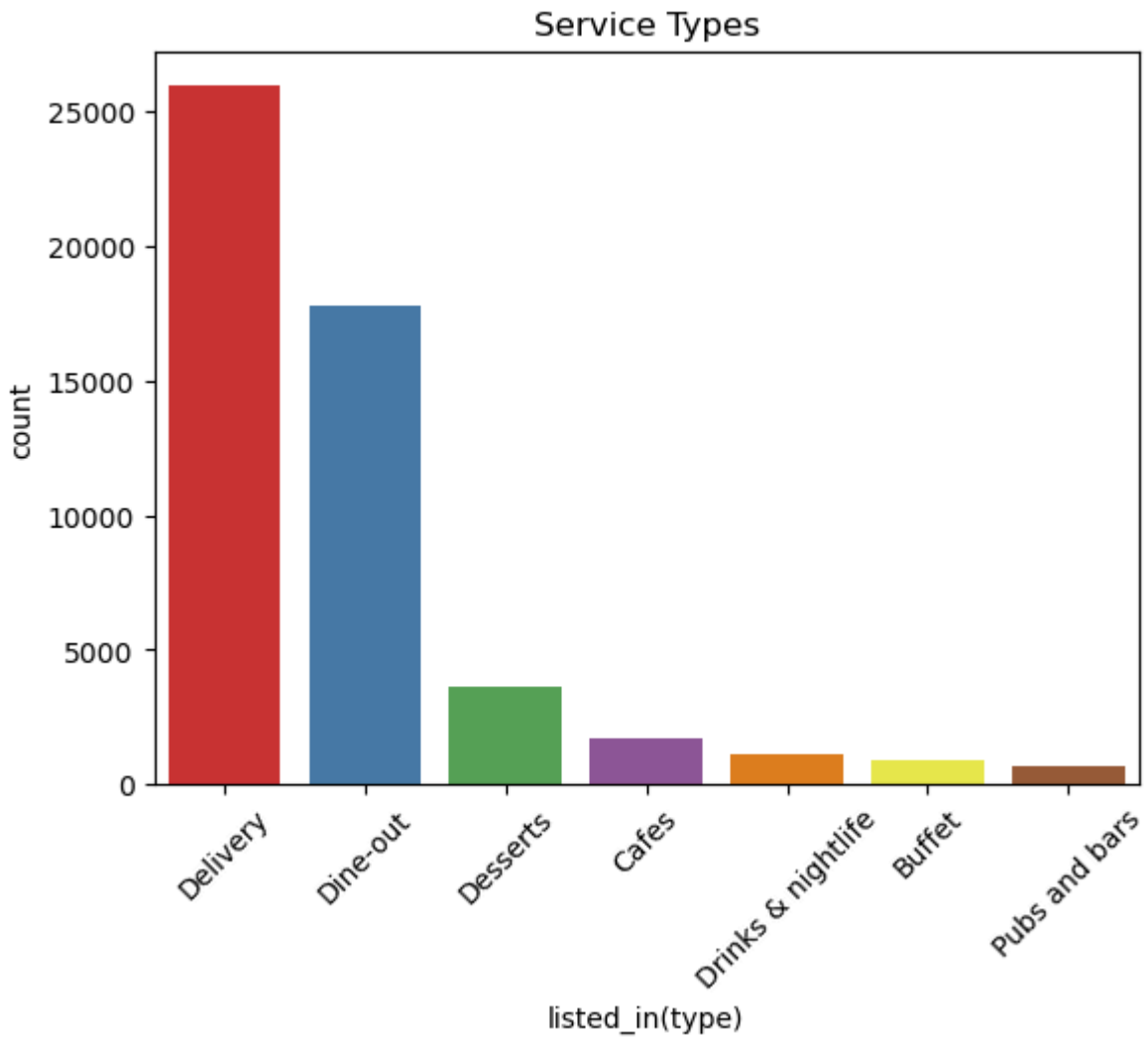
In most of the restaurants there is no table booking avalilability by zomato platform.

In [57]:
```
sns.countplot(y='rest_type', data=new_data, order=new_data['rest_type'].value_counts().iloc[:10].index, palette='muted')
plt.title("Top 10 Restaurant Types")
plt.show()
```



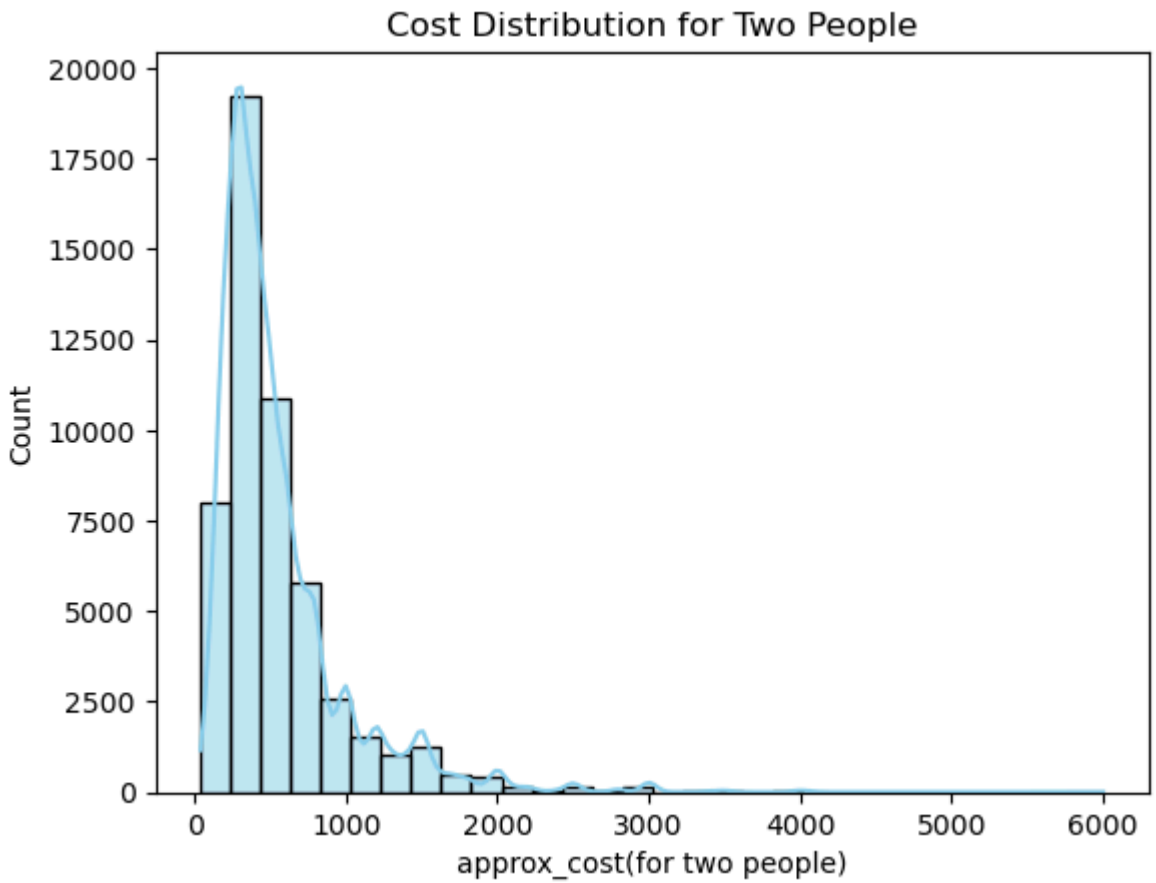
Quick Bites is the most common restaurant types

In [58]:
```
sns.countplot(data=new_data, x='listed_in(type)', order=new_data['listed_in(type)'].value_counts().index, palette='Set1')
plt.title("Service Types")
plt.xticks(rotation=45)
plt.show()
```



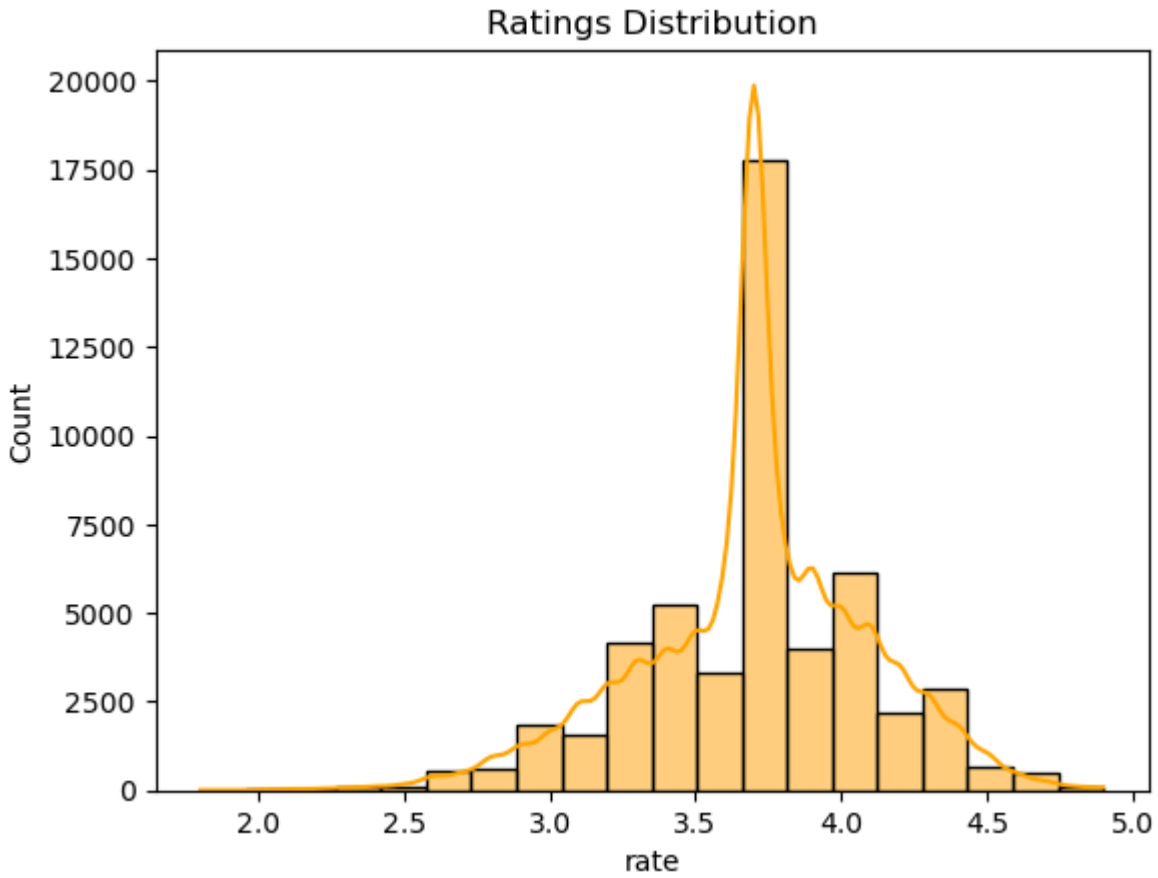
Delivery & Dine-Out is the most common service type on the zomato platform.

In [59]:
```
sns.histplot(data=new_data, x='approx_cost(for two people)', kde=True, bins=30, color='skyblue')
plt.title("Cost Distribution for Two People")
plt.show()
```



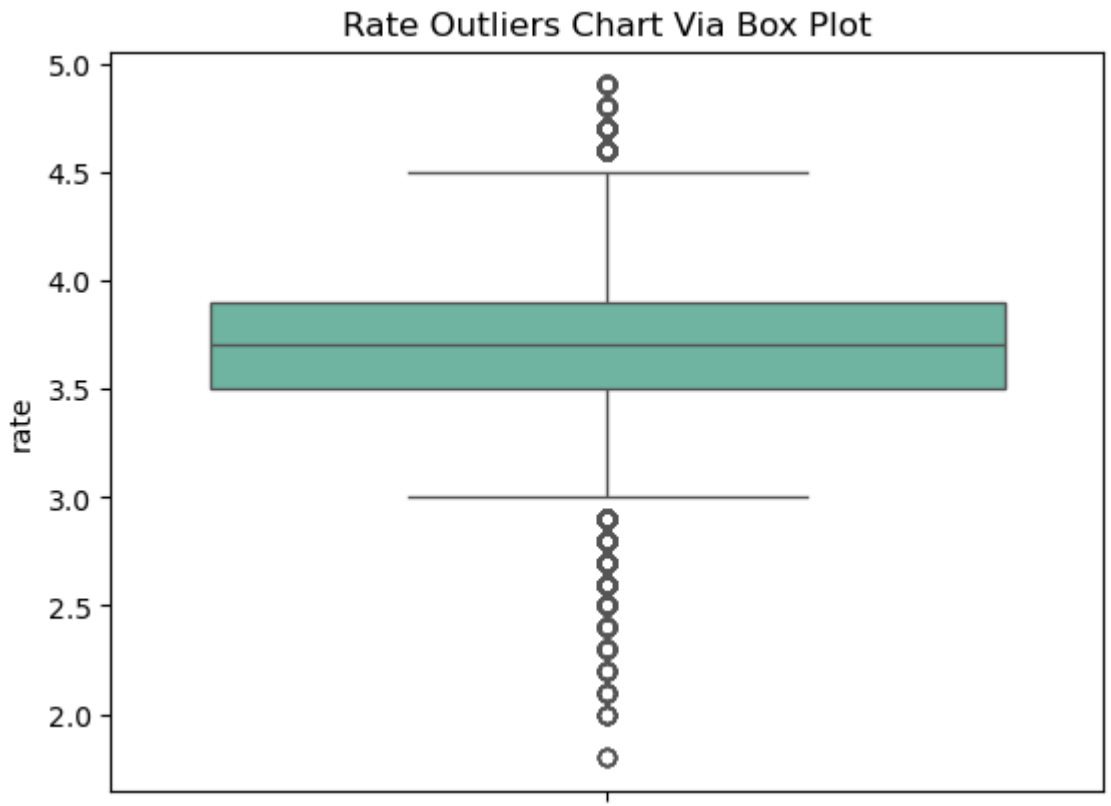
Our majority of the cost is between 0 to 1000rs caulse chart is left schewed.

In [60]:
```
sns.histplot(new_data["rate"], kde=True, bins=20, color="orange")
plt.title("Ratings Distribution")
plt.show()
```



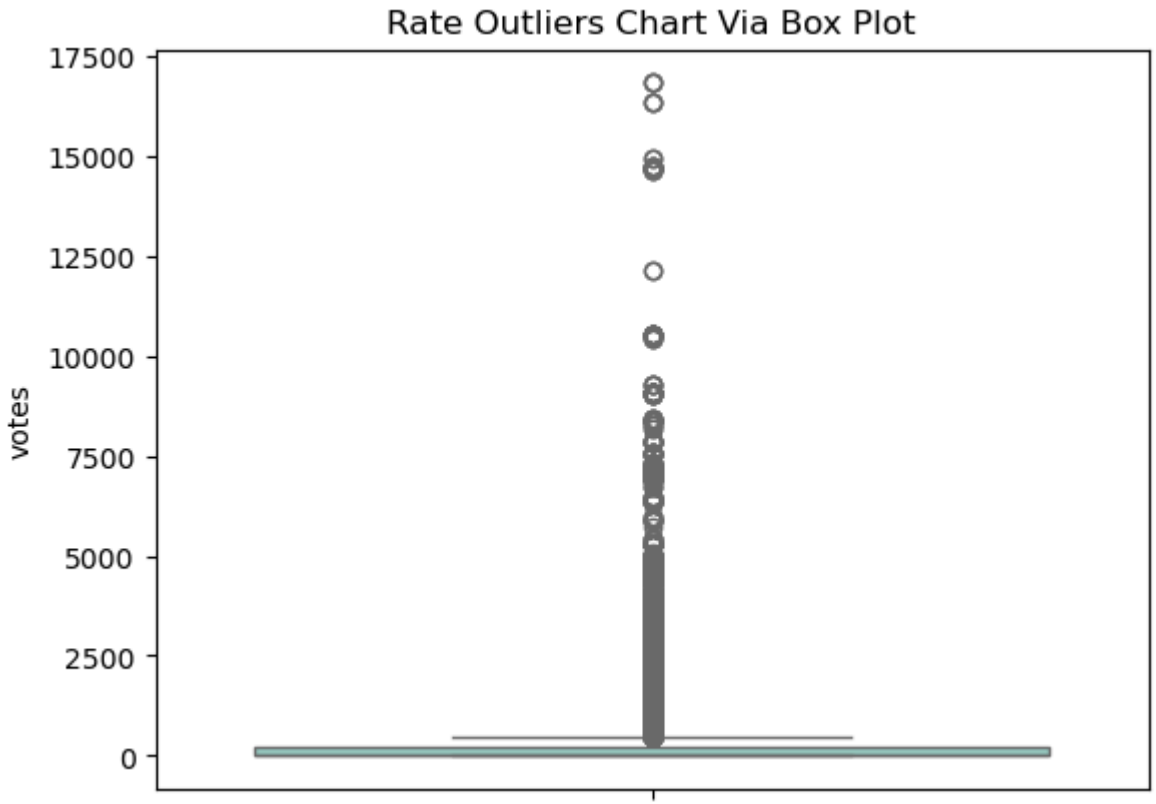
Our majority rate is 3.5 to 4.5 and graph is slightly right schewed.

In [61]:
```
# check is there any outliers present in the dataset or not?

sns.boxplot(y = "rate",data = new_data,palette = "Set2")
plt.title("Rate Outliers Chart Via Box Plot")
plt.show()
```



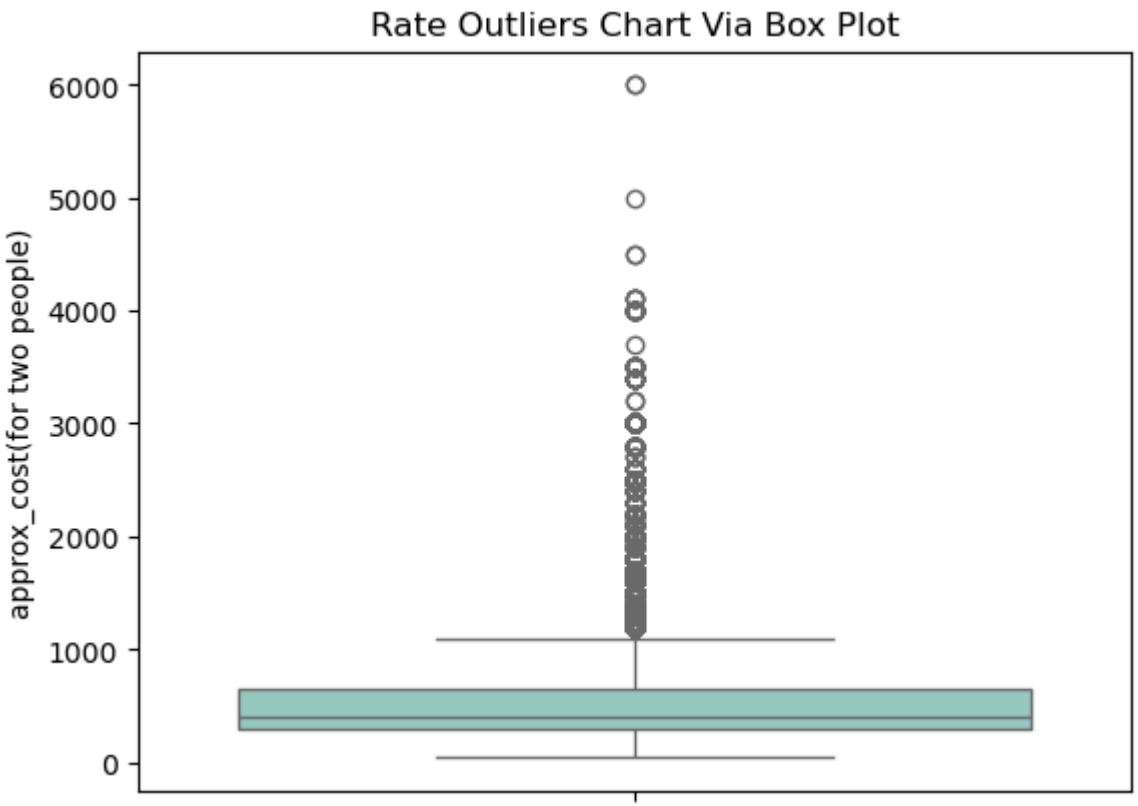
In above Chart you can say just one rating is below 2.0 So We Not consider As outlier.

In [62]:
```
# check is there any outliers present in the dataset or not?

sns.boxplot(y = "votes",data = new_data,palette = "Set3")
plt.title("Rate Outliers Chart Via Box Plot")
plt.show()
```



In the abive charts we can see the Some of rates is extreamely high above to the 15000.

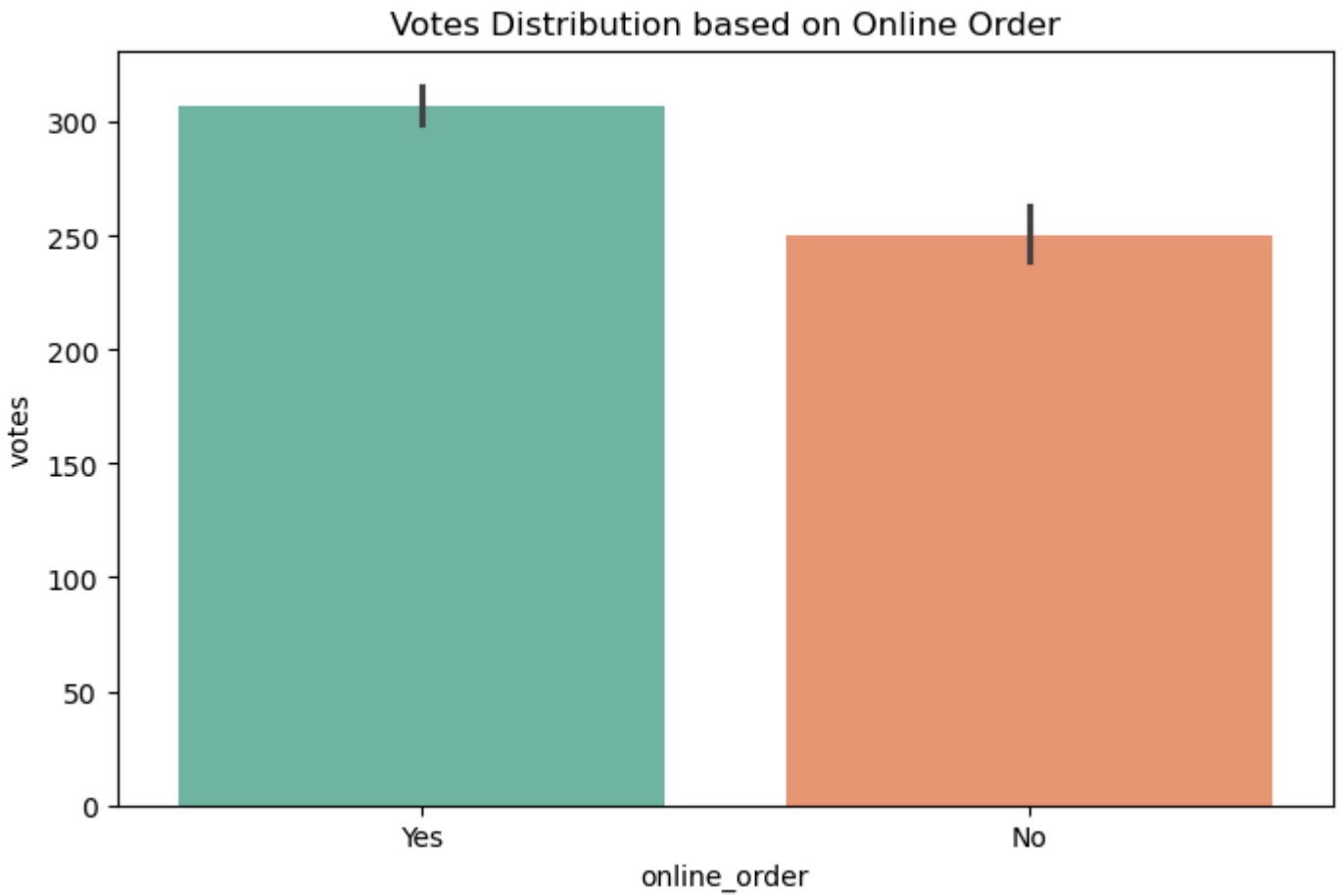
In [63]:
```
# check is there any outliers present in the dataset or not?

sns.boxplot(y = "approx_cost(for two people)",data = new_data,palette = "Set3")
plt.title("Rate Outliers Chart Via Box Plot")
plt.show()
```



In the above chart we can see one outlier 6000 approx rates for two people may it's genuine or may be it's wrongly entered data so we keep it as it is.

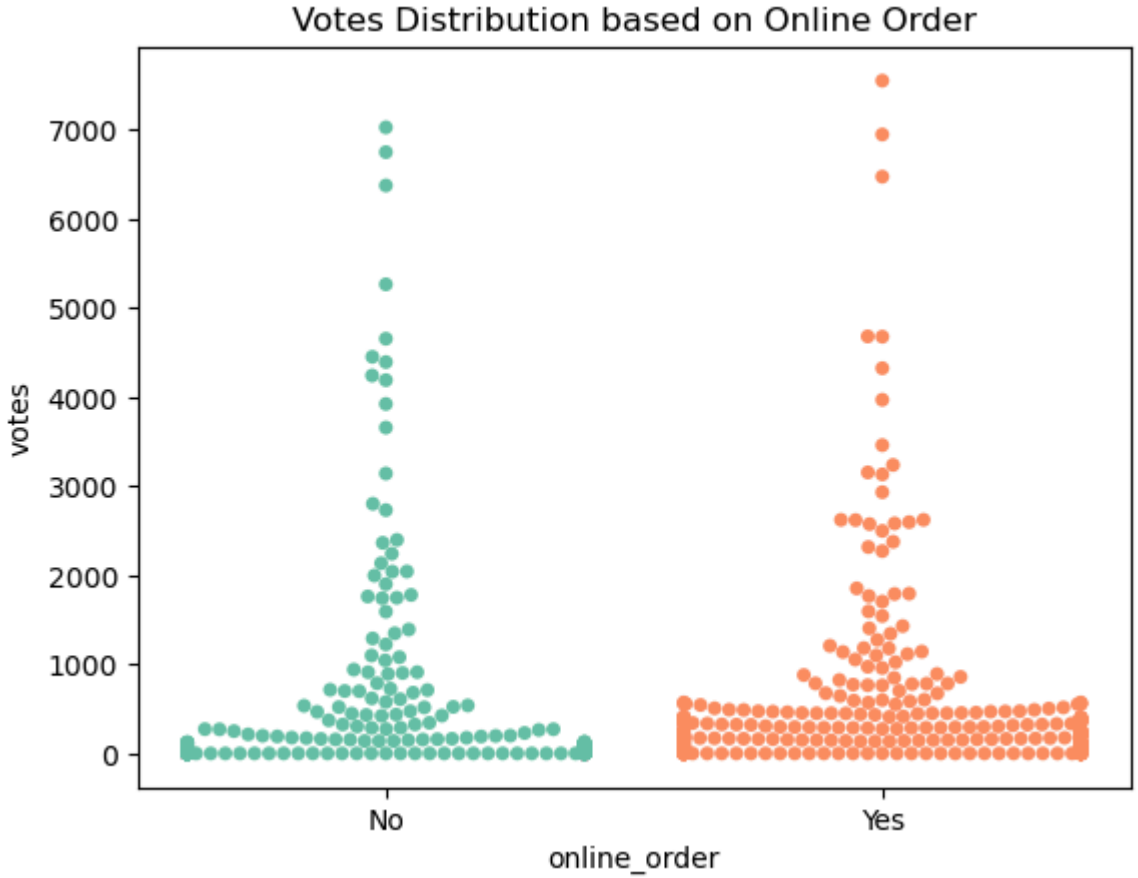
Bi-Variate-Analysis

In [64]:
```
plt.figure(figsize=(8, 5))
sns.barplot(x="online_order", y="votes", data=new_data, palette="Set2")
plt.title("Votes Distribution based on Online Order")
plt.show()
```



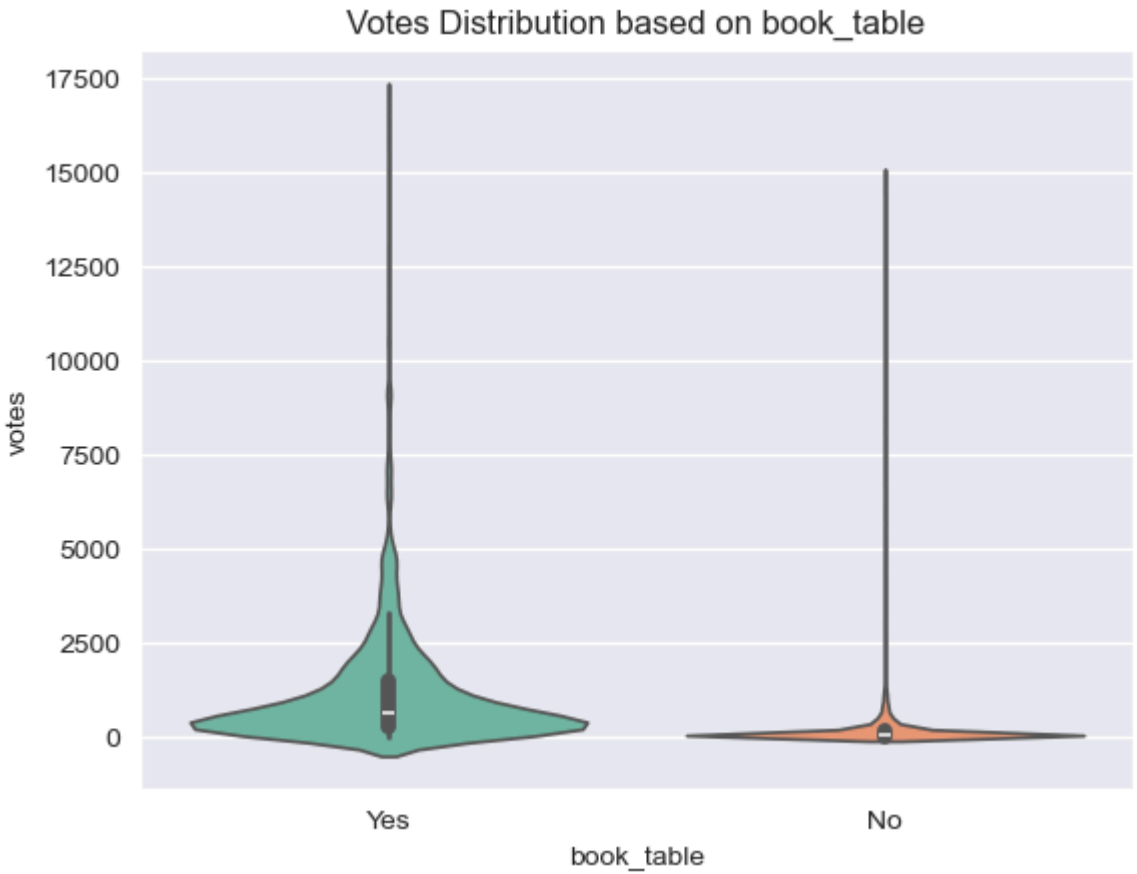
In above Chart You Can see the online order catch the higher votes as compare to the none Online order Mode on Zomato platform.

In [65]:
```
sample_data = new_data.sample(1000, random_state=1)
sns.swarmplot(x="online_order", y="votes", data=sample_data, palette="Set2")
plt.title("Votes Distribution based on Online Order")
plt.show()
```



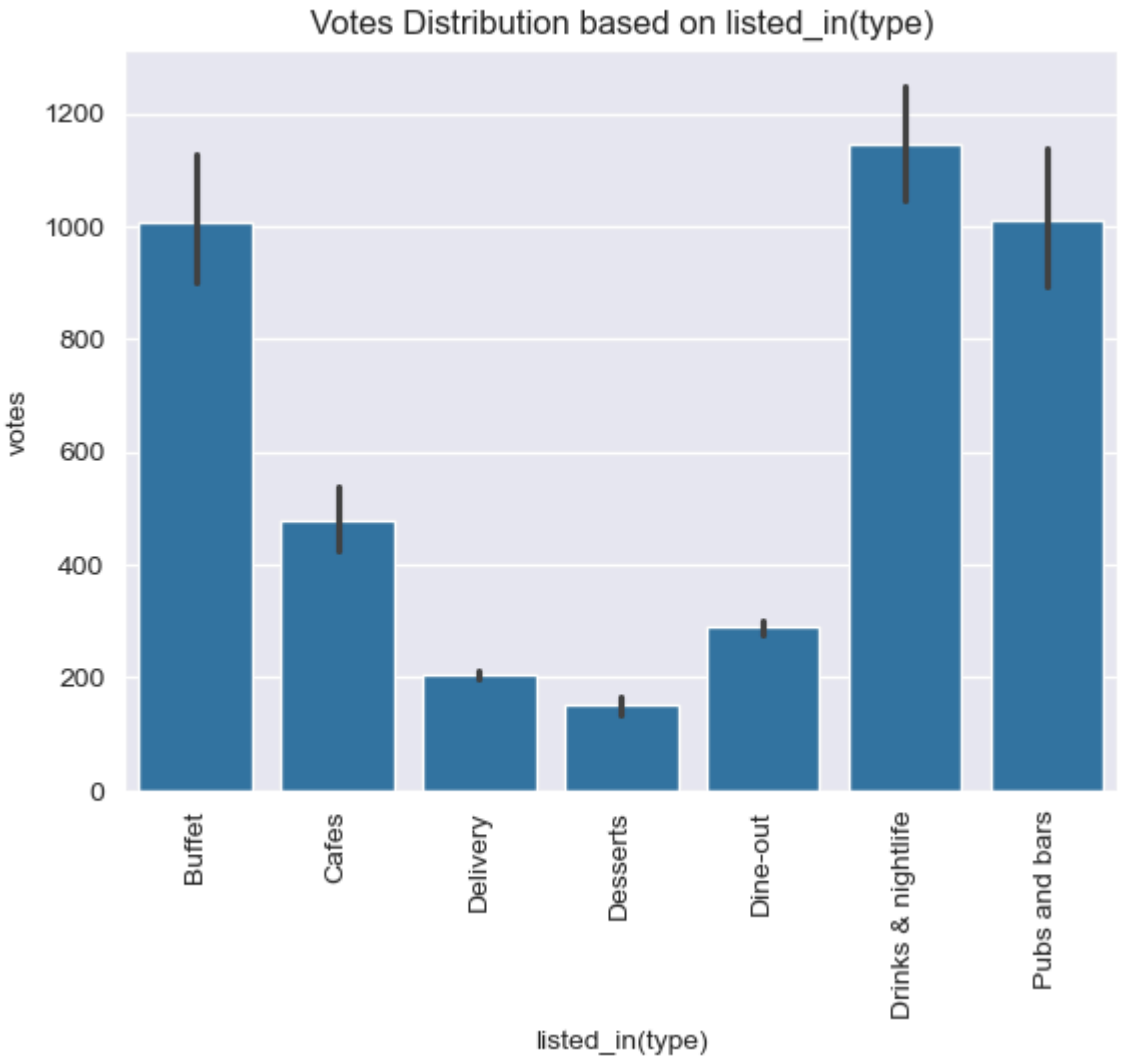
In above chart we can see and try to extract the information in the sample 1000 people but the majority is online order is yes get the higher votes as comparare to the none higher order on the zomato platform.

In [66]:
```
sns.set_style("darkgrid")
sns.violinplot(x = "book_table",y = "votes",data = new_data,palette = "Set2")
plt.title("Votes Distribution based on book_table")
plt.show()
```



In the above violineplot as you can see the those who booked the table get the high votes on zomato platform.

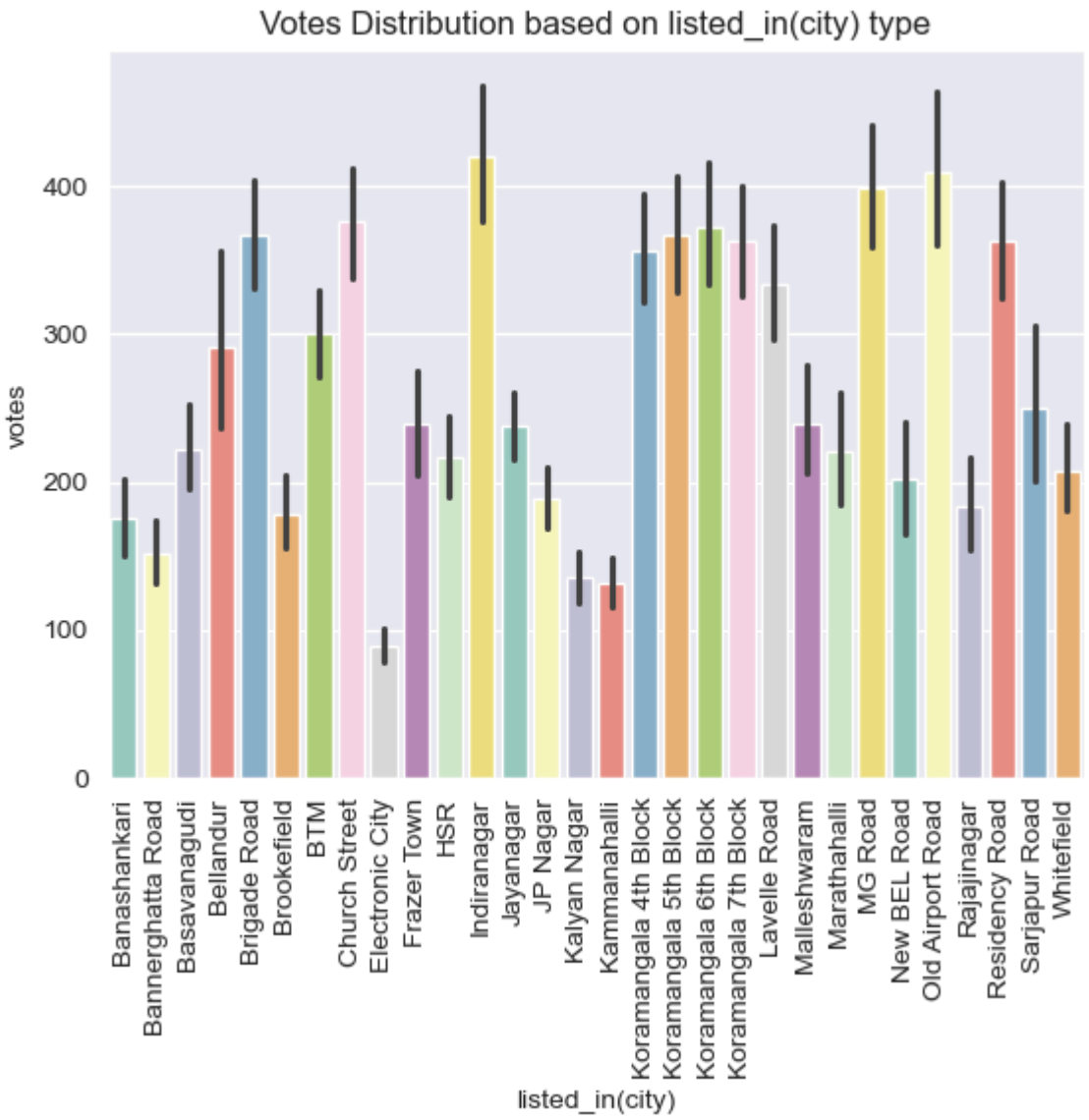
In [67]:
```
sns.barplot(x="listed_in(type)",y="votes",data = new_data)
plt.title("Votes Distribution based on listed_in(type)")
plt.xticks(rotation = "vertical")
plt.show()
```



As you can see the Drinks & Nightlife type restaurants get the higher votes on the zomato platform.

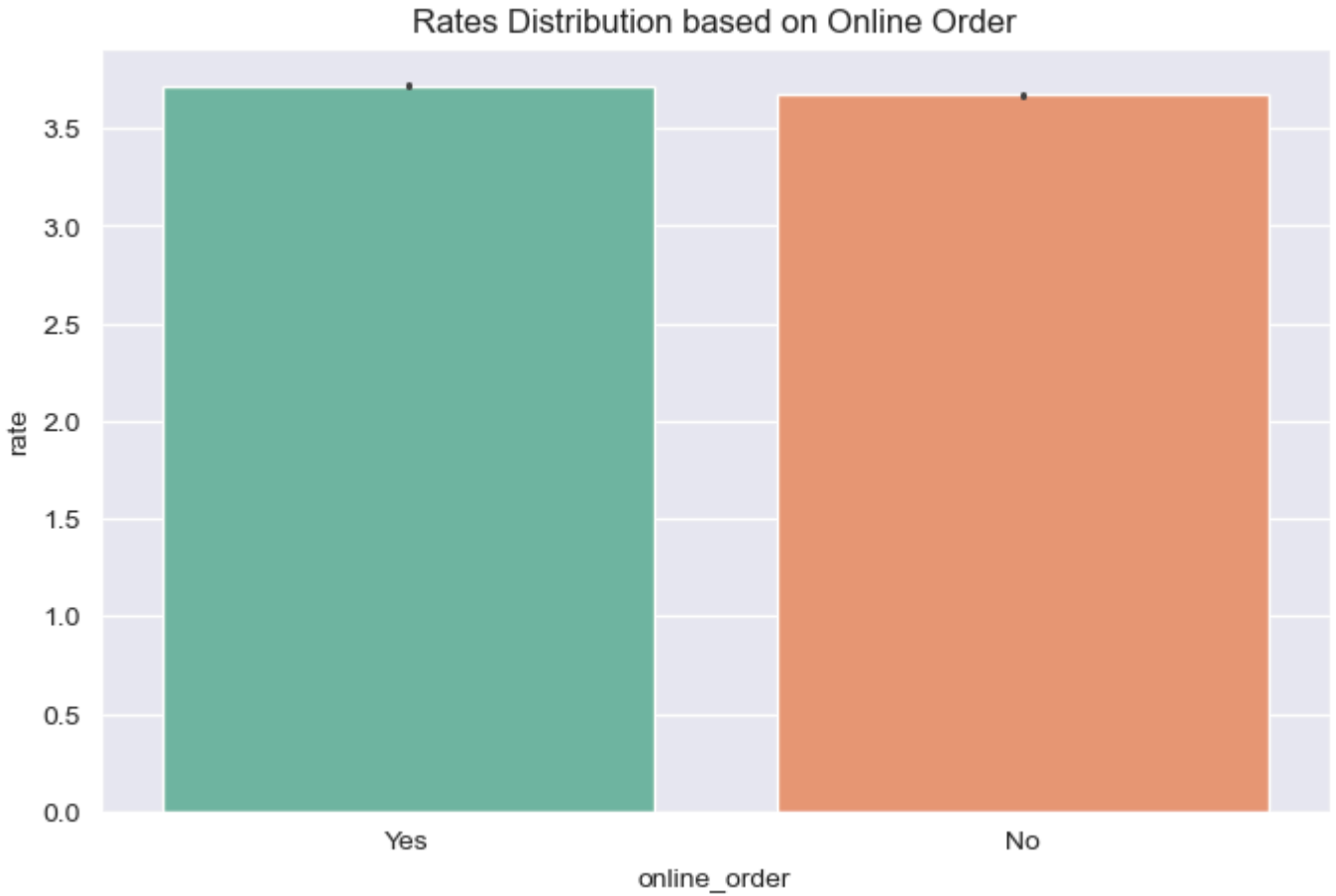
In [68]:
```
sns.barplot(x="listed_in(city)",y="votes",data = new_data,palette = "Set3")
plt.title("Votes Distribution based on listed_in(city) type")
```

```
plt.xticks(rotation = "vertical")
plt.show()
```



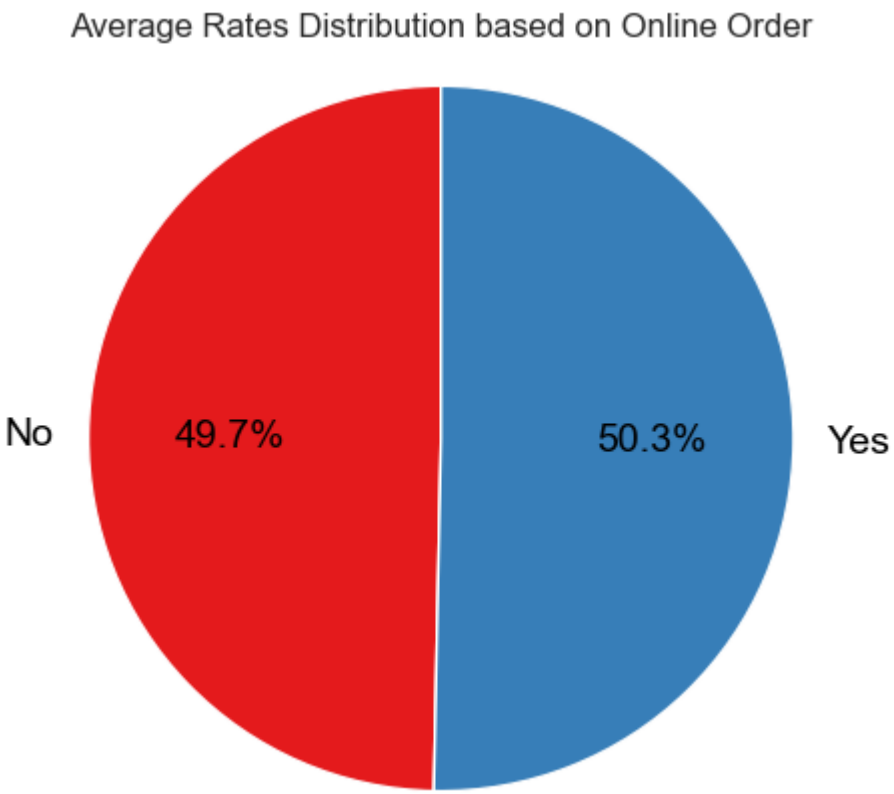
In the above chart as you can see the Indiranagar & Old Airport Road side restaurants get the higher votes in zomato platform.

In [69]:
```
plt.figure(figsize=(8, 5))
sns.barplot(x="online_order", y="rate", data=new_data, palette="Set2")
plt.title("Rates Distribution based on Online Order")
plt.show()
```

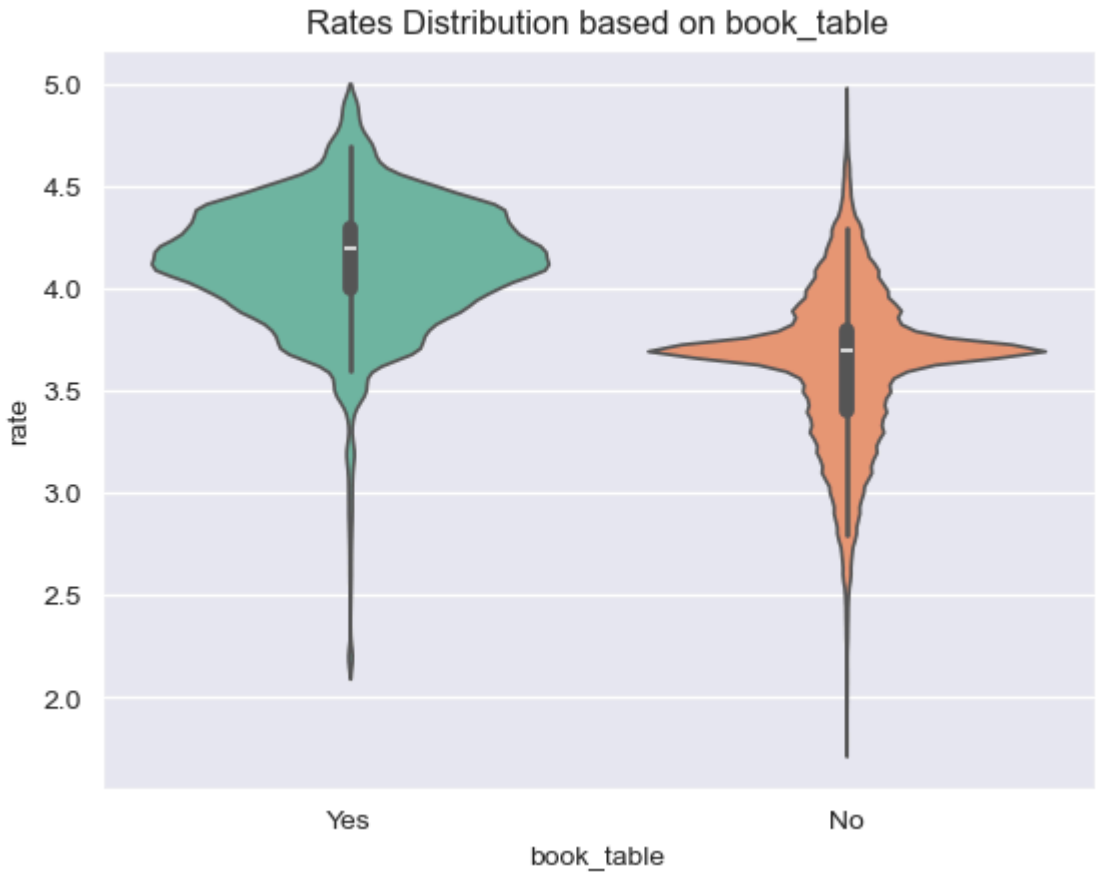


Those who placed online order get the higher ratings on zomato platform.

In [70]:
```
average_rate = new_data.groupby('online_order')['rate'].mean()

plt.figure(figsize=(5,5))
plt.pie(average_rate, labels=average_rate.index, autopct='%1.1f%%', startangle=90, colors=plt.cm.Set1.colors,textprops={'color
plt.title("Average Rates Distribution based on Online Order")
plt.axis('equal')  # Ensure the pie chart is circular
plt.show()
```

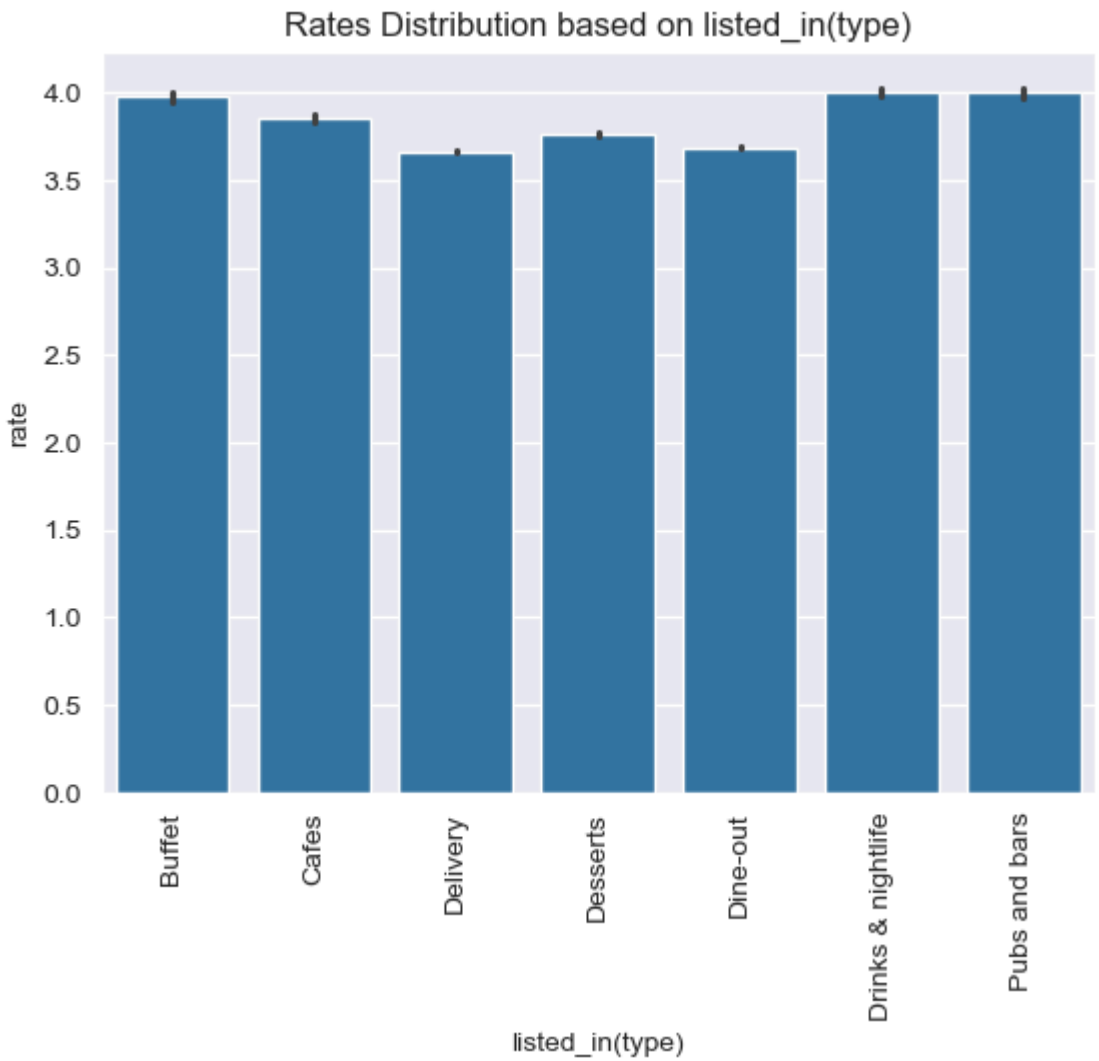


In [71]:
```
sns.set_style("darkgrid")
sns.violinplot(x = "book_table",y = "rate",data = new_data,palette = "Set2")
plt.title("Rates Distribution based on book_table")
plt.show()
```



Those who Booked the online table get the highest rates on the zomato platform.

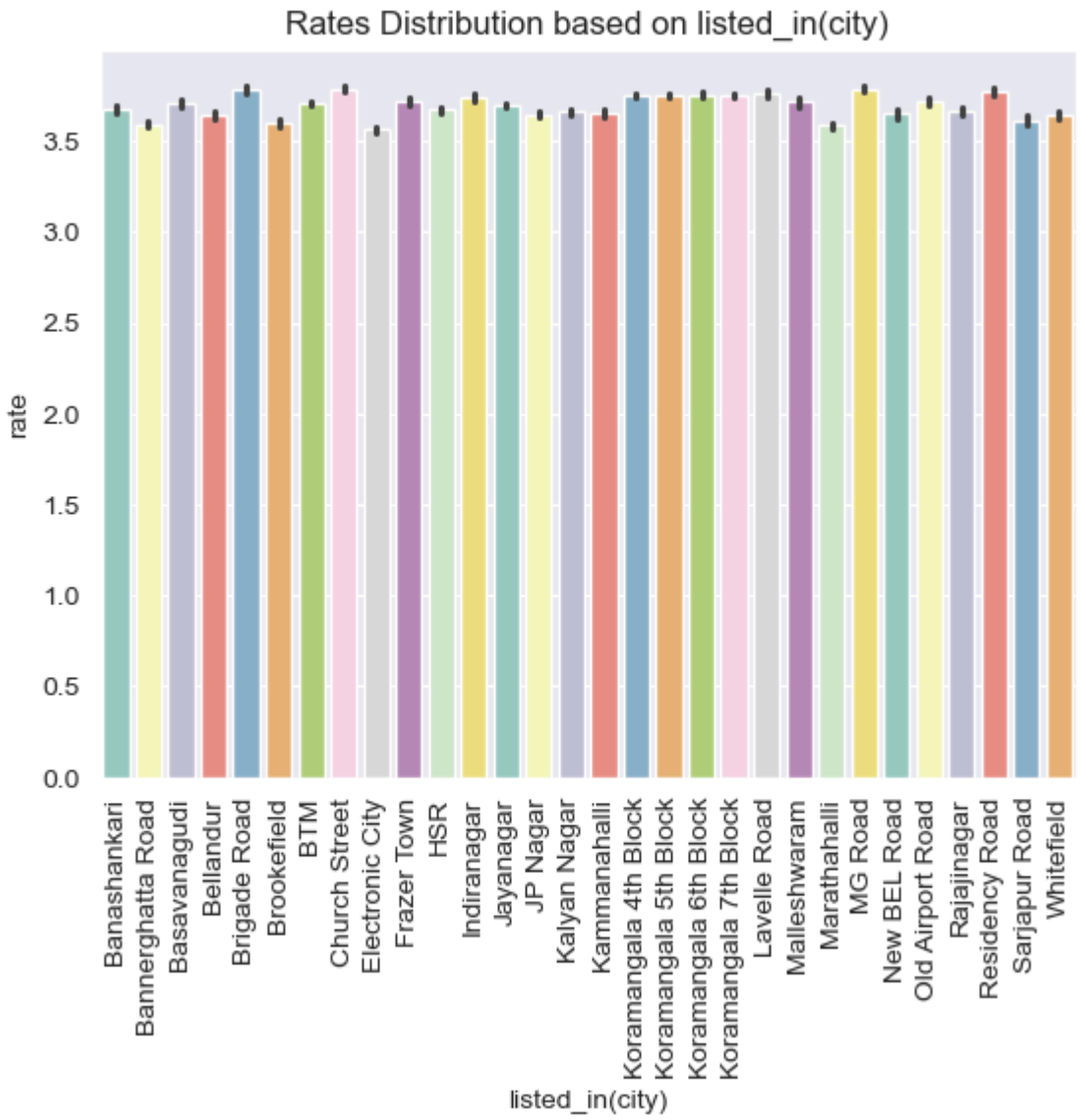
In [72]:
```
sns.barplot(x="listed_in(type)",y="rate",data = new_data)
plt.title("Rates Distribution based on listed_in(type)")
plt.xticks(rotation = "vertical")
plt.show()
```



In Above chart as you can see the Drinks & Nightlife & Pubs and bars types restaurants get the highest ratings on zomato platform.

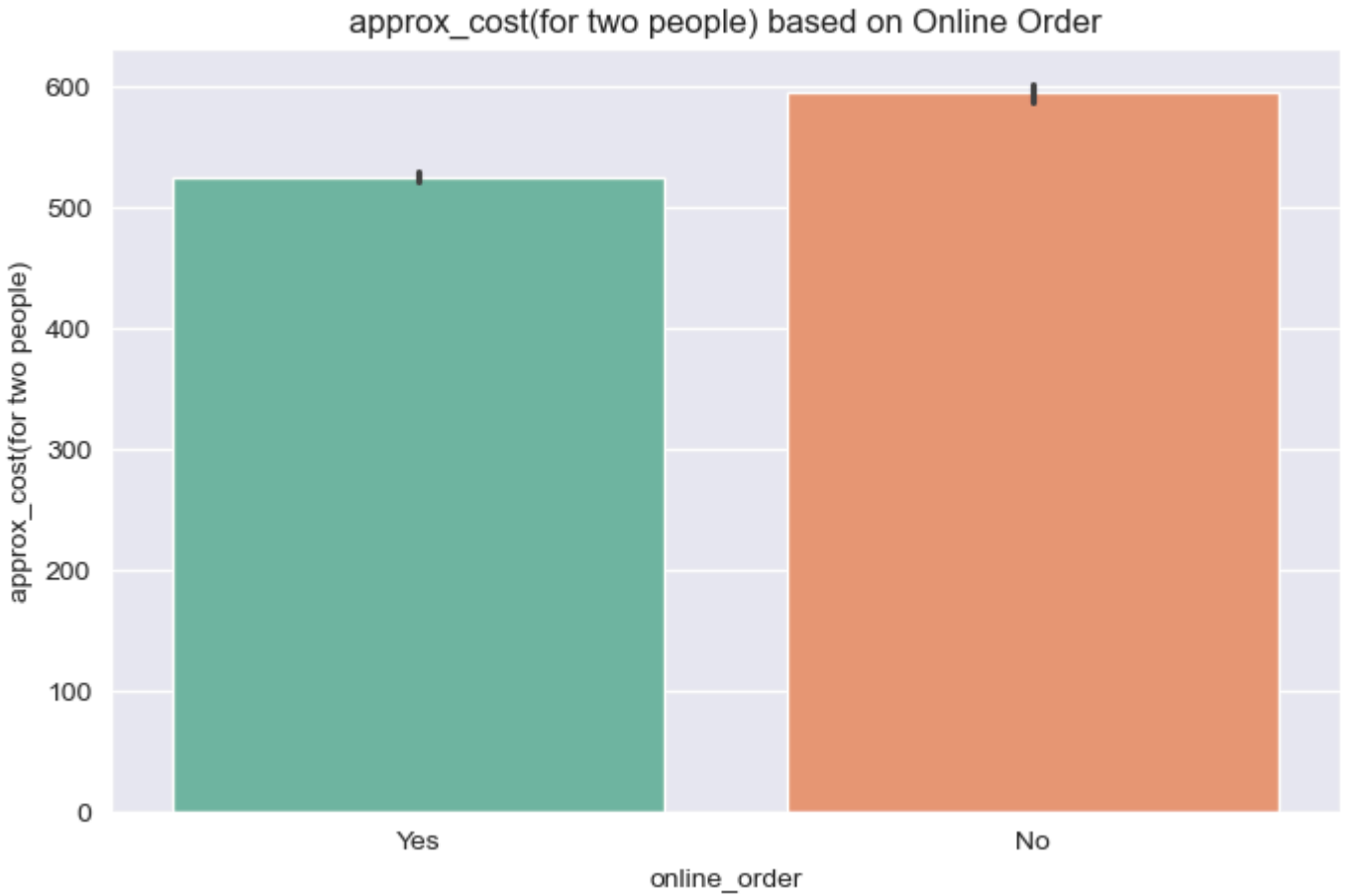
In [73]:
```
sns.barplot(x="listed_in(city)",y="rate",data = new_data,palette = "Set3")
plt.title("Rates Distribution based on listed_in(city)")
```

```
plt.xticks(rotation = "vertical")
plt.show()
```



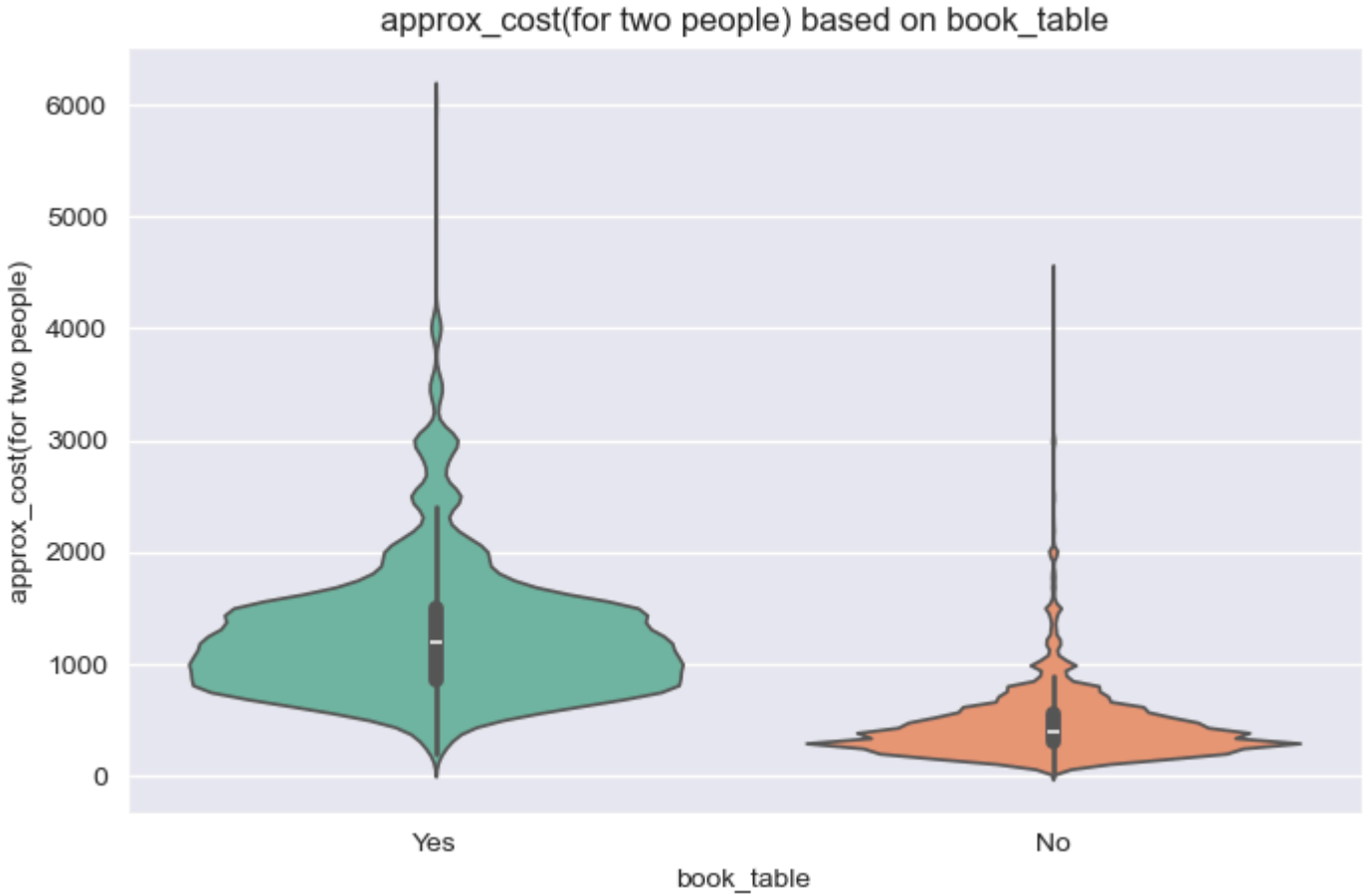
In above chart as you can see Most of the city restaurants is the top pick for highes ratings on zomato platform.

In [74]:
```
plt.figure(figsize=(8, 5))
sns.barplot(x="online_order", y="approx_cost(for two people)", data=new_data, palette="Set2")
plt.title("approx_cost(for two people) based on Online Order")
plt.show()
```



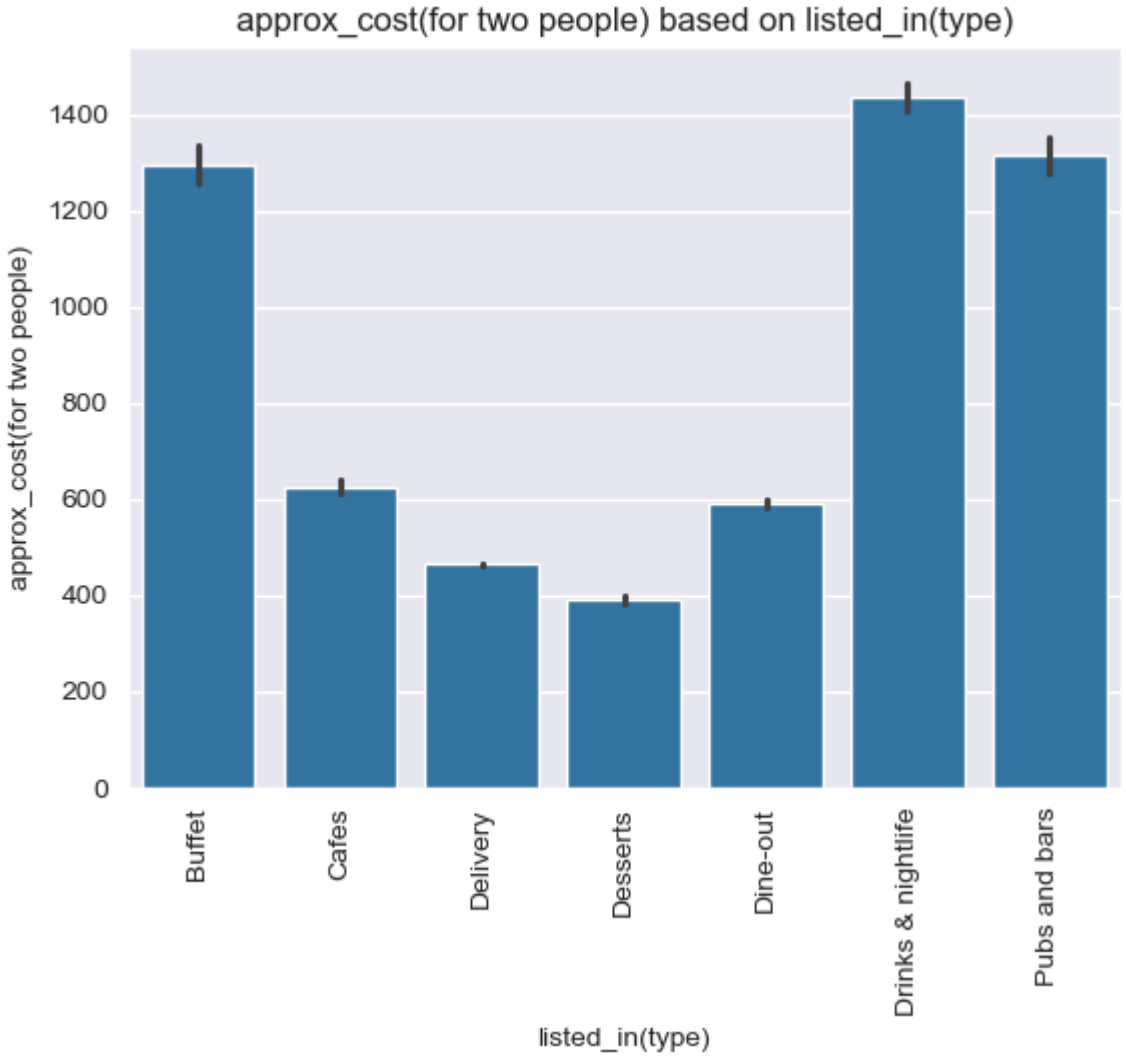
In the above chart you can see the average cost for two people is very high for those who are not placed the order online and that's why they did not get any offeres.

In [75]:
```
plt.figure(figsize=(8, 5))
sns.violinplot(x="book_table", y="approx_cost(for two people)", data=new_data, palette="Set2")
plt.title("approx_cost(for two people) based on book_table")
plt.show()
```



In above chart as you can see those who booked the table and pay the higher approx cost for two peoples.

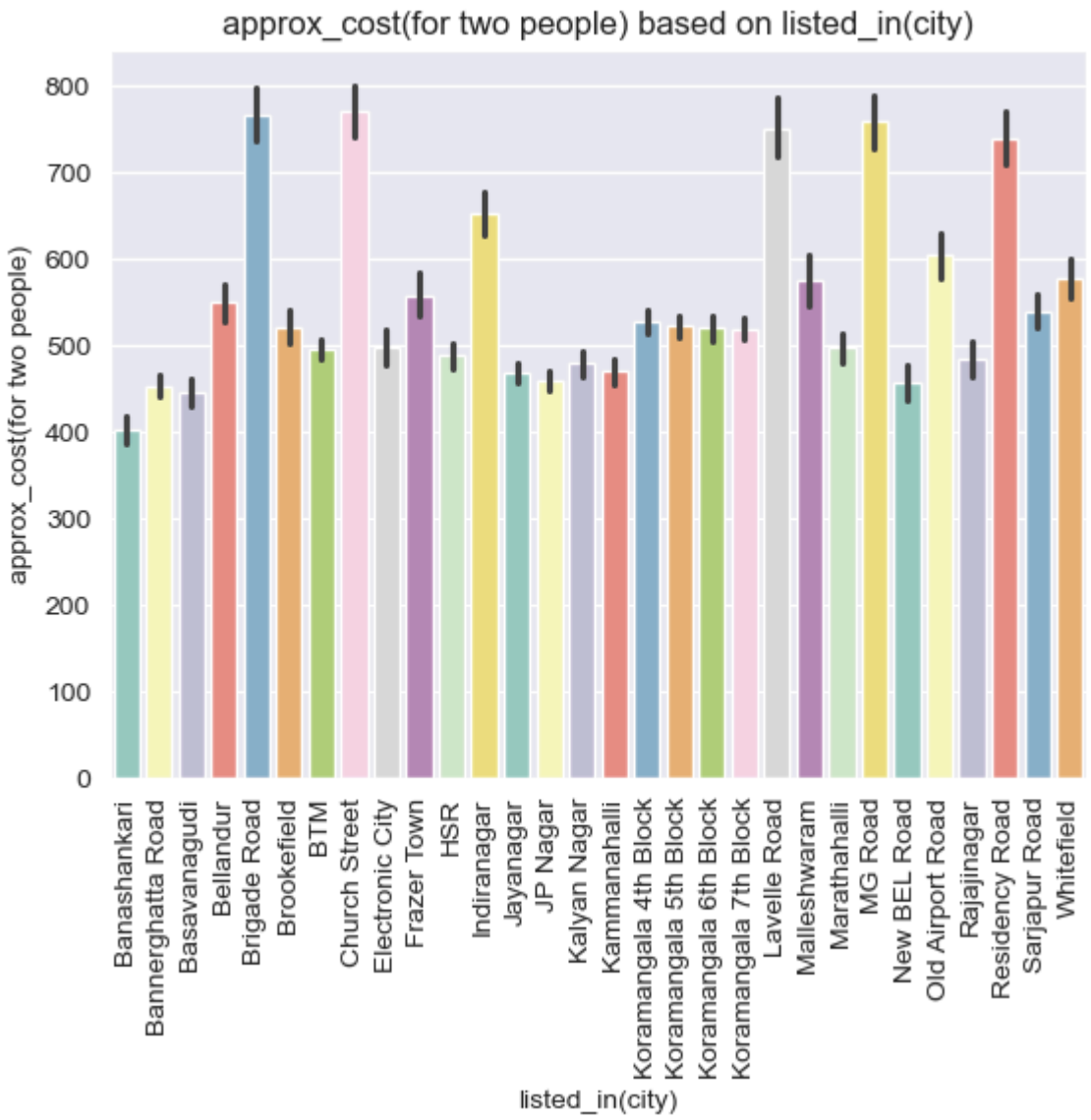
In [76]:
```
sns.barplot(x="listed_in(type)",y="approx_cost(for two people)",data = new_data)
plt.title("approx_cost(for two people) based on listed_in(type)")
plt.xticks(rotation = "vertical")
plt.show()
```



Obeviosely In the Drinks & Nighlife type restaurants Customers payed the approx cost for two people is very high.

In [77]:
```
sns.barplot(x="listed_in(city)",y="approx_cost(for two people)",data = new_data,palette = "Set3")
plt.title("approx_cost(for two people) based on listed_in(city)")
```

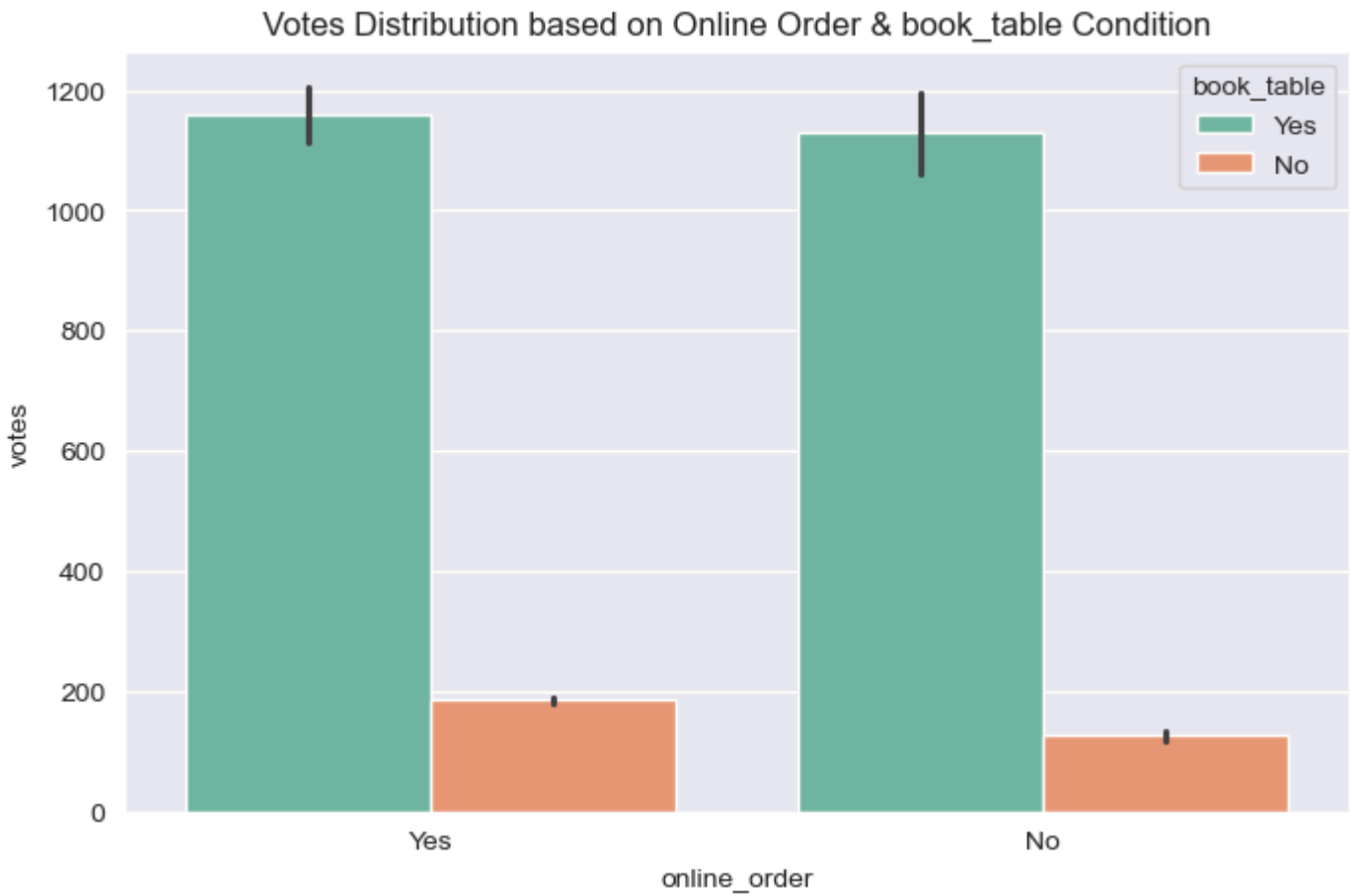
```
plt.xticks(rotation = "vertical")
plt.show()
```



In above chart you can see the Brigade Road,Church Street,Lavelle Road,MG Road,Residency Road side Restaurants Regular/Visited Customer paid a approx cost for two people above 700.

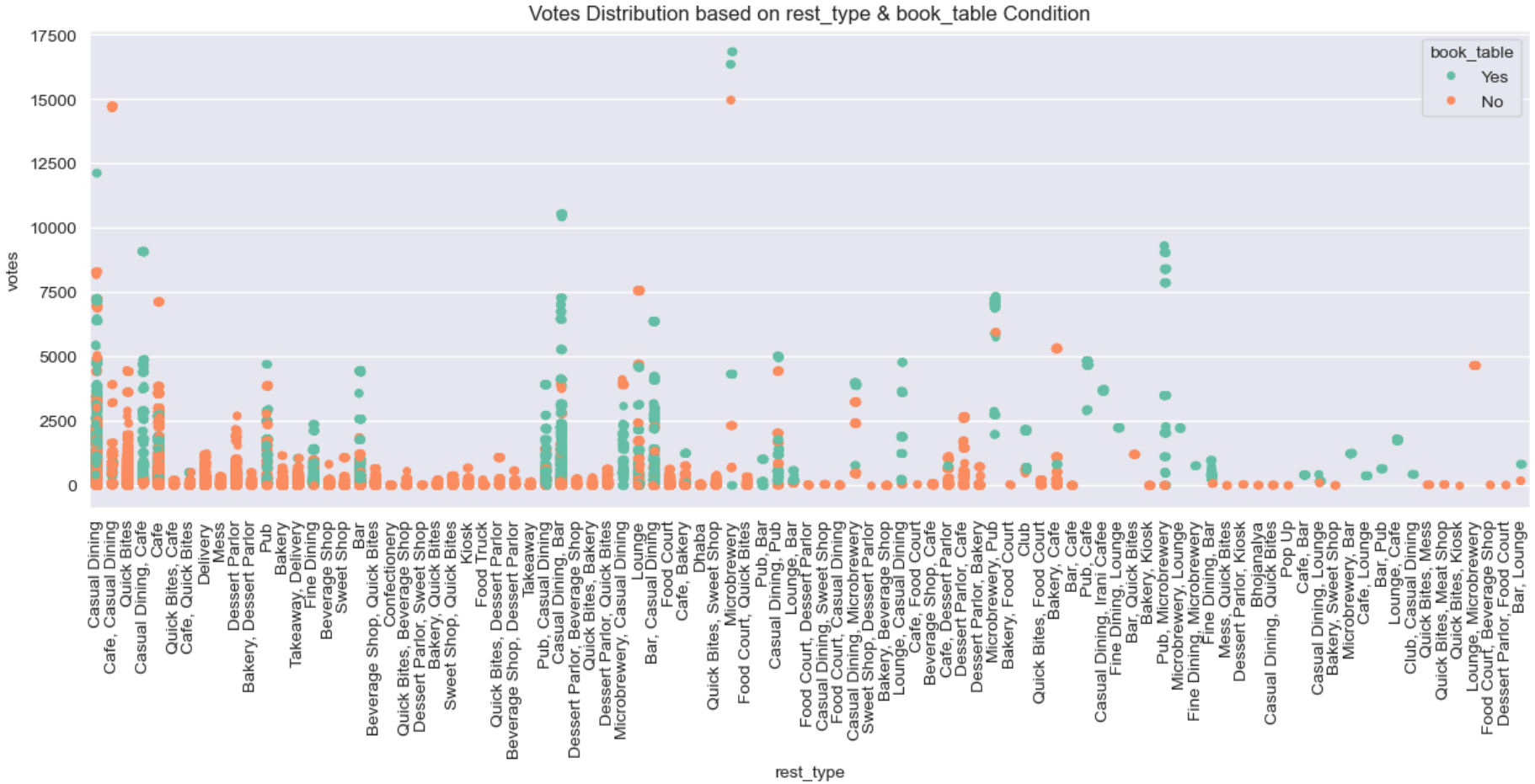
Multi-Variate-Analysis

In [78]:
```
plt.figure(figsize=(8, 5))
sns.barplot(x="online_order", y="votes",hue = "book_table" , data=new_data, palette="Set2")
plt.title("Votes Distribution based on Online Order & book_table Condition")
plt.show()
```



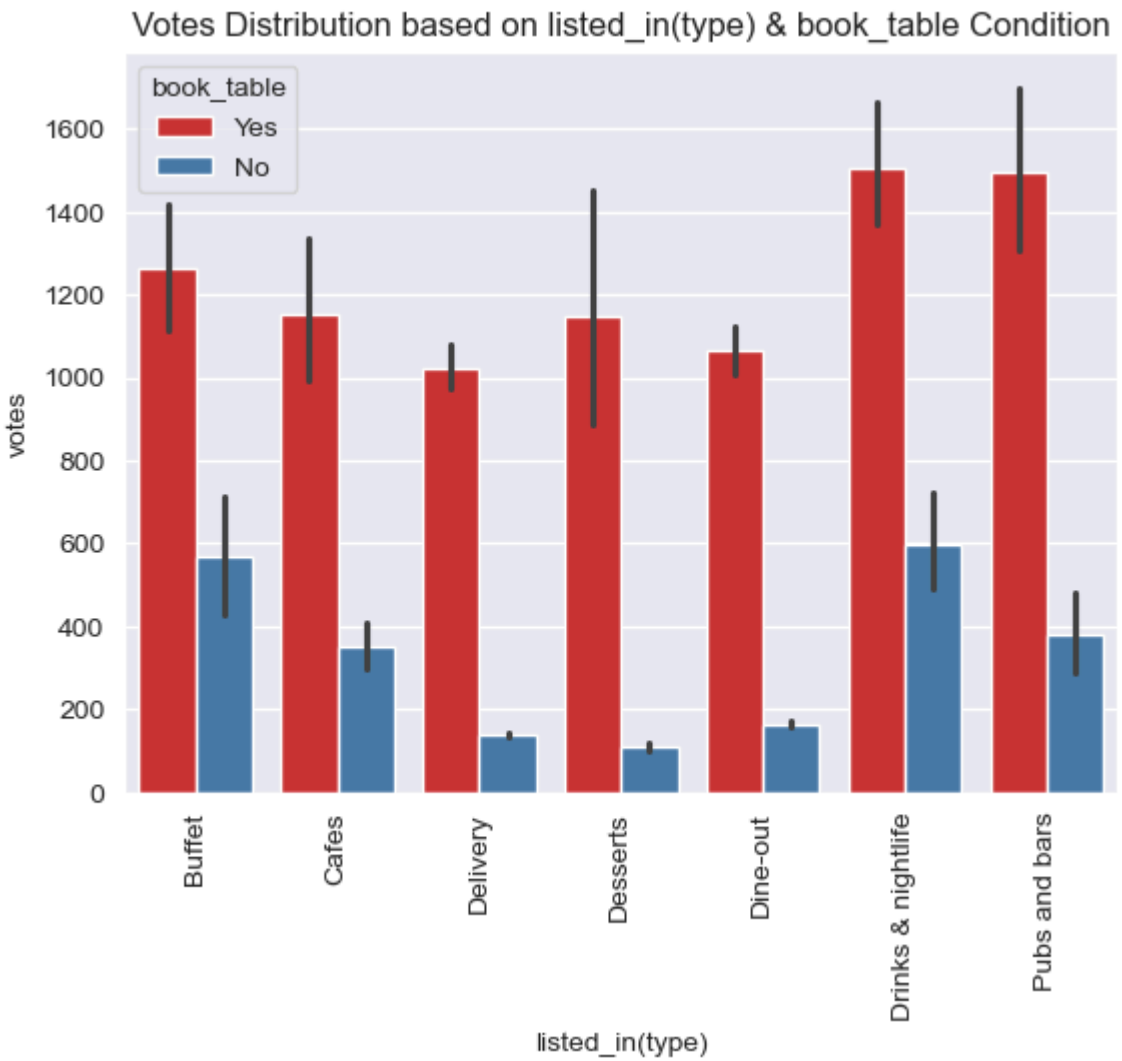
Those who placed the online order & even book the table online get the higher votes on zomato platform.

In [79]:
```
plt.figure(figsize=(15, 5))
sns.set_style("darkgrid")
sns.stripplot(x = "rest_type",y = "votes",hue = "book_table",data = new_data,palette = "Set2")
plt.title("Votes Distribution based on rest_type & book_table Condition")
plt.xticks(rotation = "vertical")
plt.show()
```



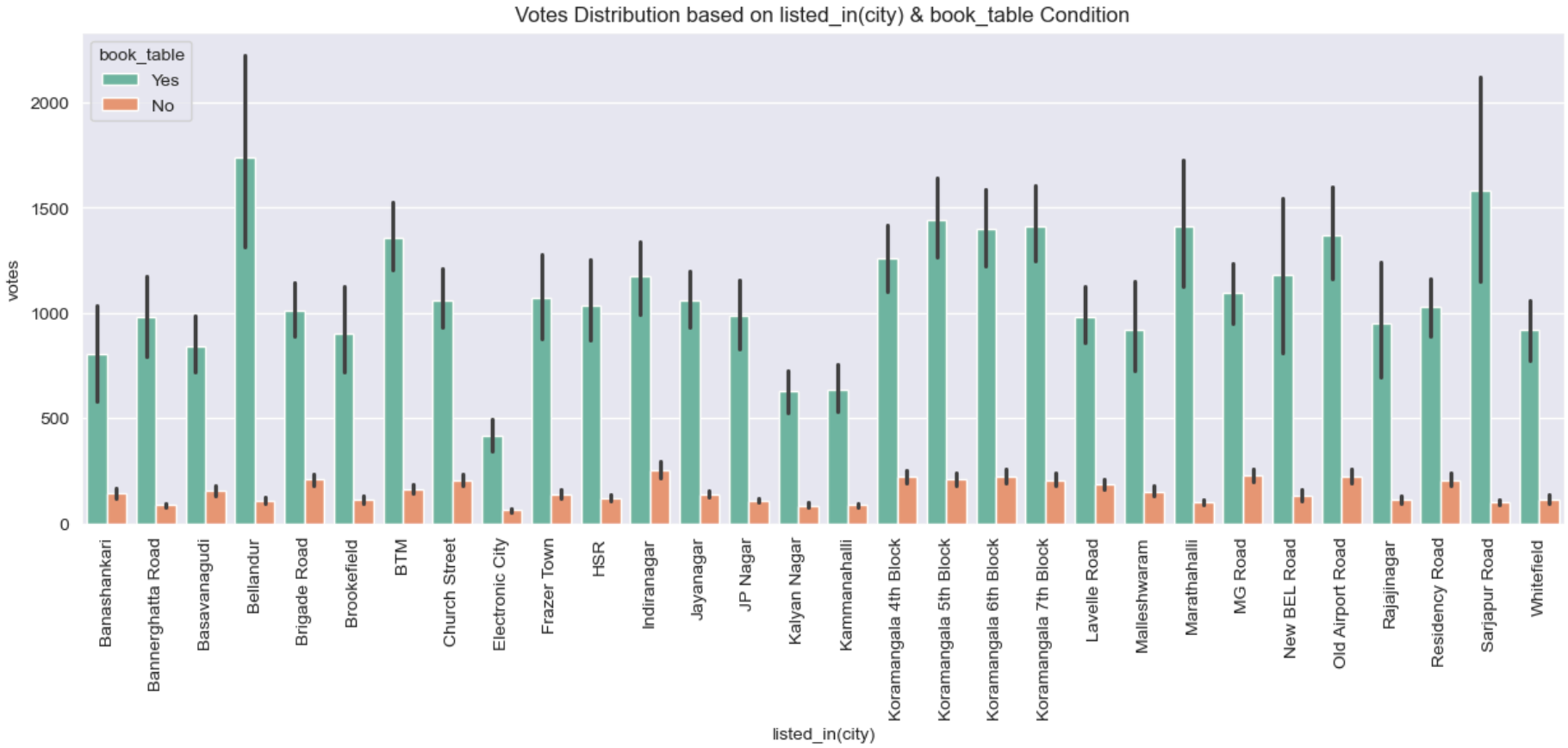
Most of the type of restaurant booked the table and get the below 7500 votes on zomato platform.

In [80]:
```
sns.barplot(x="listed_in(type)",y="votes",hue = "book_table",data = new_data,palette = "Set1")
plt.title("Votes Distribution based on listed_in(type) & book_table Condition")
plt.xticks(rotation = "vertical")
plt.show()
```



In the above chart as you can see the Drinks & Nightlife and the Pubs and Bars service type of the Restaurants and booked the table yes get the highest votes above 1400 and majority of the other service type of the Restaurants ge the Below 1200 votes.

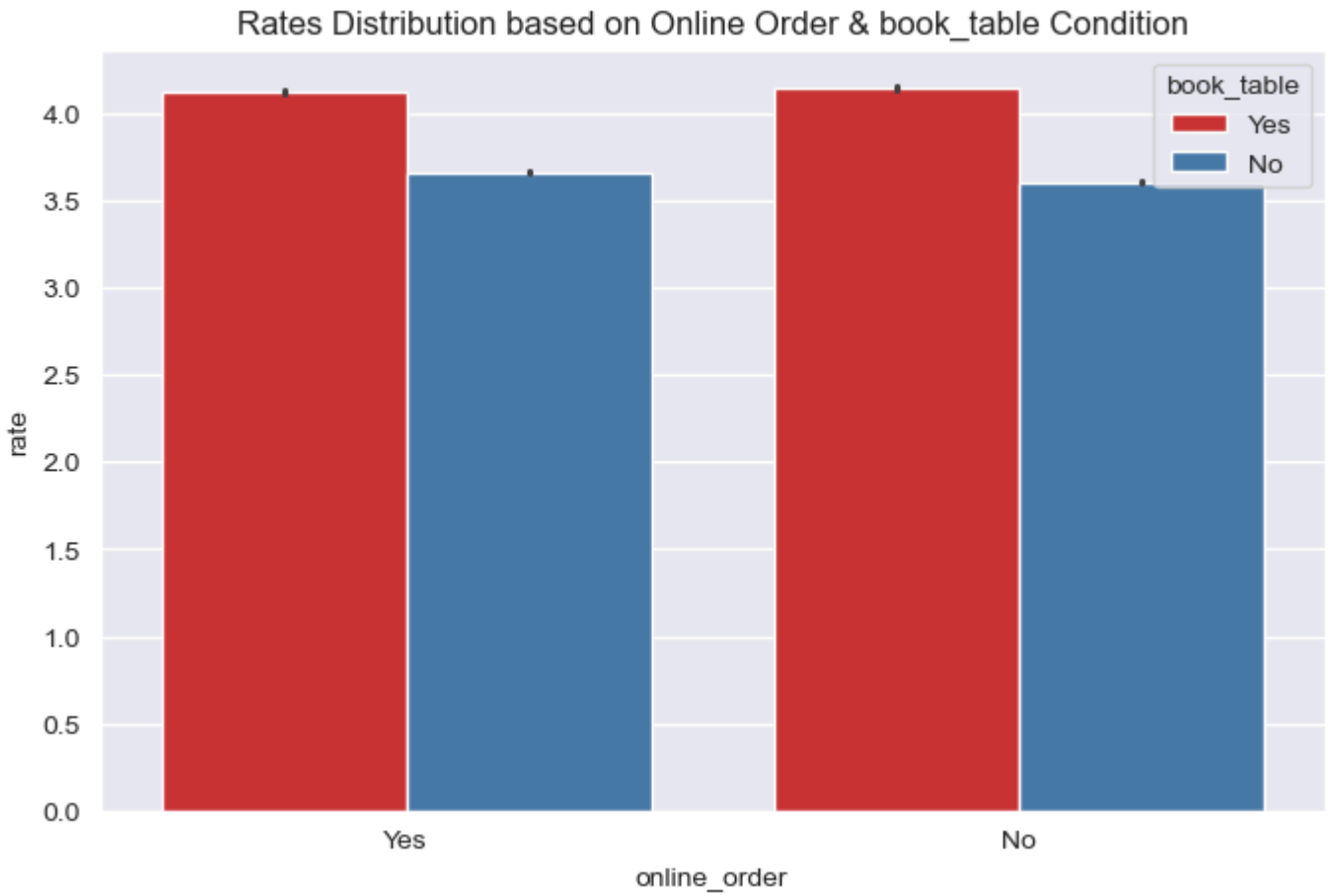
In [81]:
```
plt.figure(figsize=(15, 5))
sns.barplot(x="listed_in(city)",y="votes",hue = "book_table",data = new_data,palette = "Set2")
plt.title("Votes Distribution based on listed_in(city) & book_table Condition")
plt.xticks(rotation = "vertical")
plt.show()
```



In the Bellandur & Sarjapur Road side Restaurants Booked the table get the higher Public votes on zomato platform.

In [82]:
```
plt.figure(figsize=(8, 5))
sns.barplot(x="online_order", y="rate",hue = "book_table" , data=new_data, palette="Set1")
```

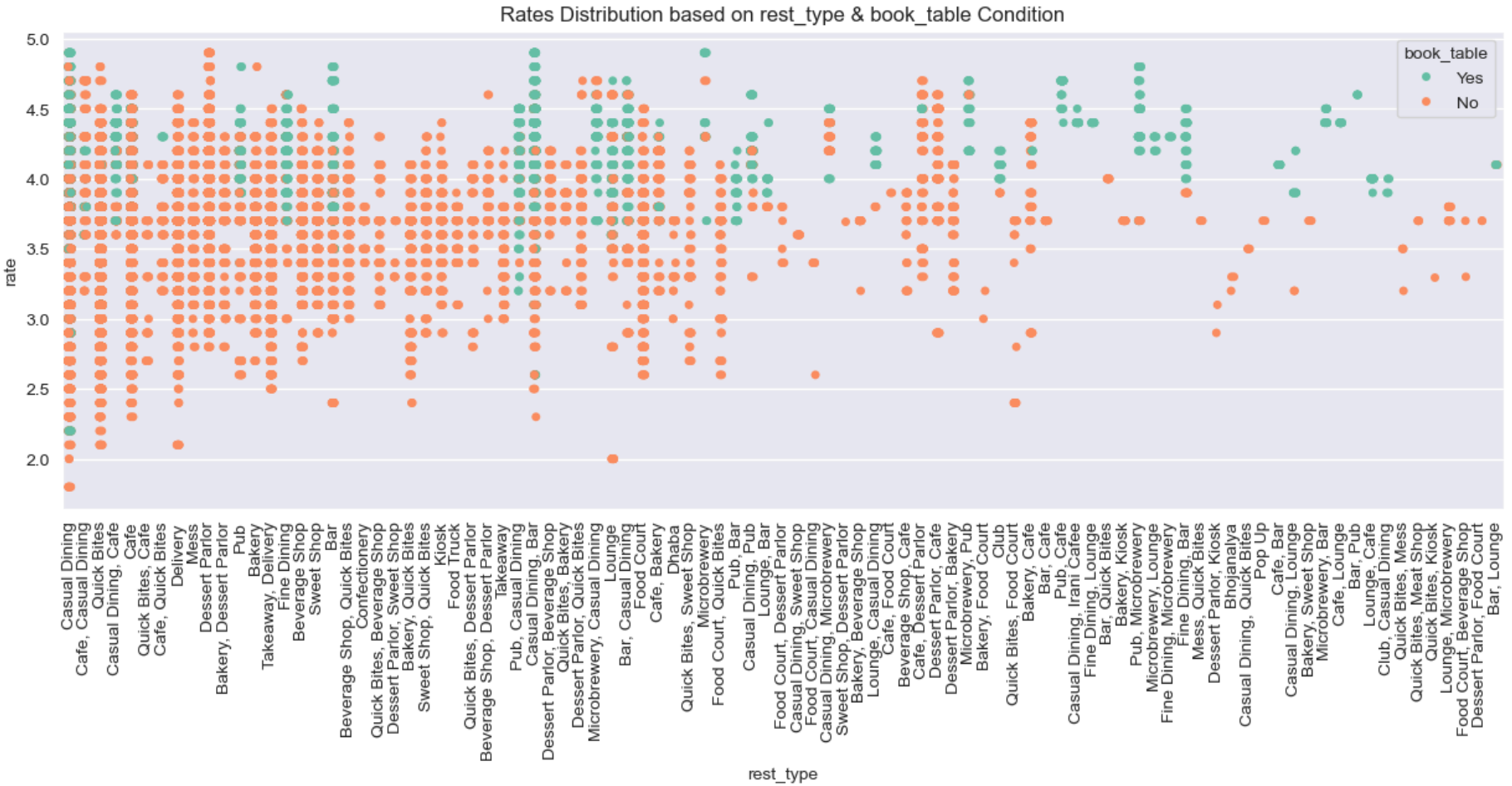
```
plt.title("Rates Distribution based on Online Order & book_table Condition")
plt.show()
```



Those who booked the table & Online order get the higher ratings on the zomato plateform.

In [83]:
```
plt.figure(figsize=(15, 5))
sns.set_style("darkgrid")
sns.stripplot(x = "rest_type",y = "rate",hue = "book_table",data = new_data,palette = "Set2")
plt.title("Rates Distribution based on rest_type & book_table Condition")
```

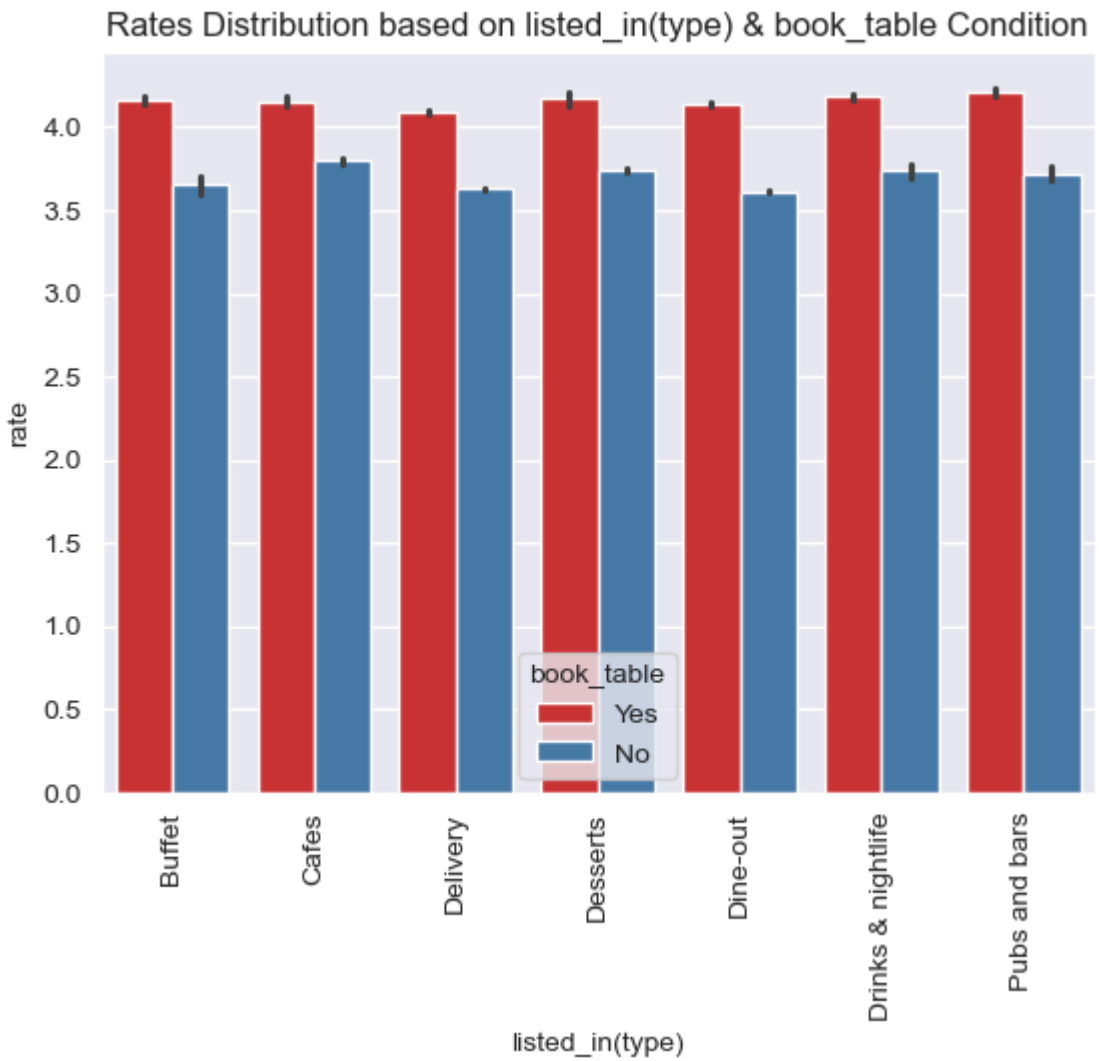
```
plt.xticks(rotation = "vertical")
plt.show()
```



Here you can see the most of the restaurants types those who not the booked the table get the more than 2.0 ratings.

Interesting thing is that type of restaurants booked the table get the highest ratings more than 3.0 so keep it up for table booking.

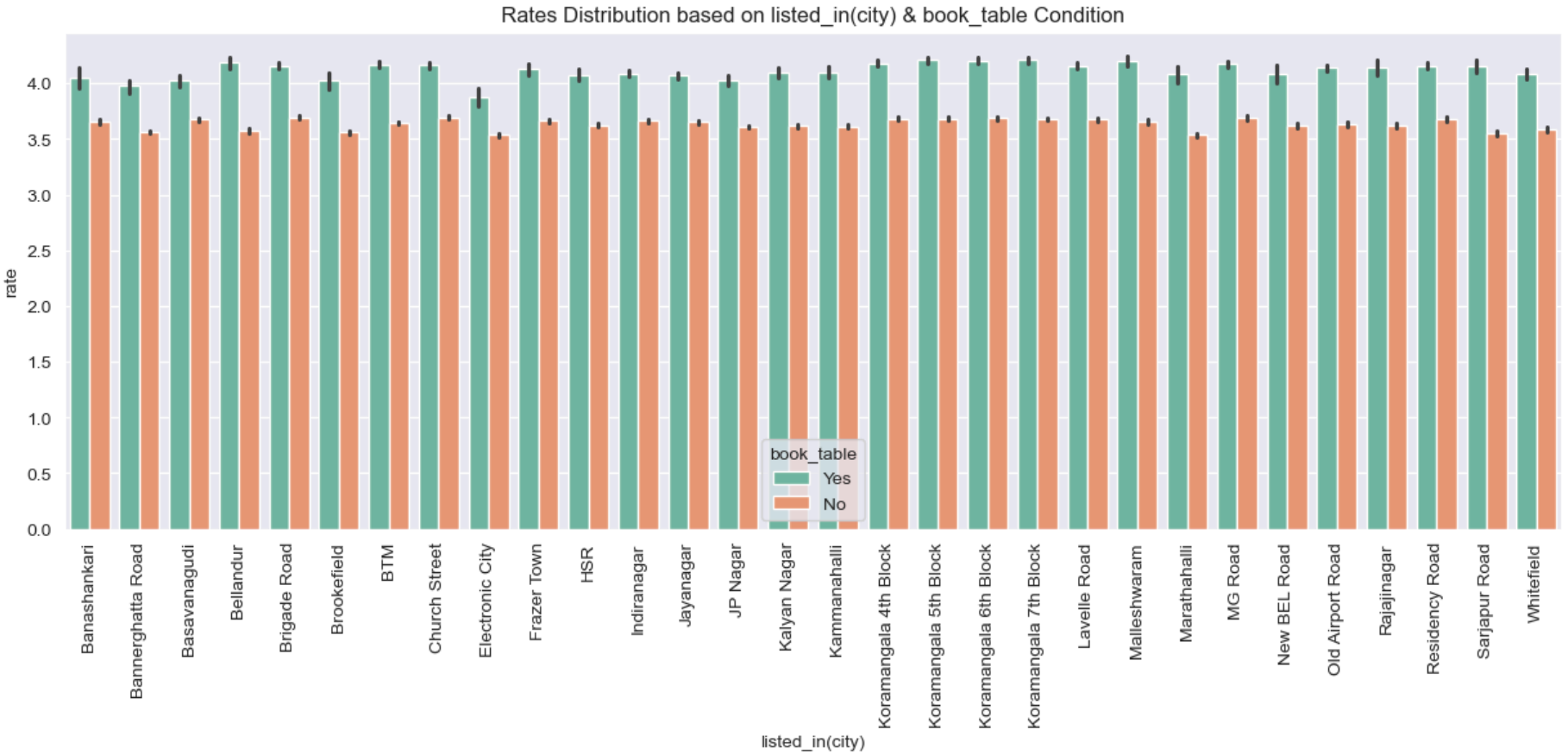
In [84]:
```
sns.barplot(x="listed_in(type)",y="rate",hue = "book_table",data = new_data,palette = "Set1")
plt.title("Rates Distribution based on listed_in(type) & book_table Condition")
plt.xticks(rotation = "vertical")
plt.show()
```



Most of the Restaurant & their service type get more than 4.0 ratings and those who not booked table get more than 3.5 ratings on zomato platform.

In [85]:
```
plt.figure(figsize=(15, 5))
sns.barplot(x="listed_in(city)",y="rate",hue = "book_table",data = new_data,palette = "Set2")
plt.title("Rates Distribution based on listed_in(city) & book_table Condition")
```

```
plt.xticks(rotation = "vertical")
plt.show()
```

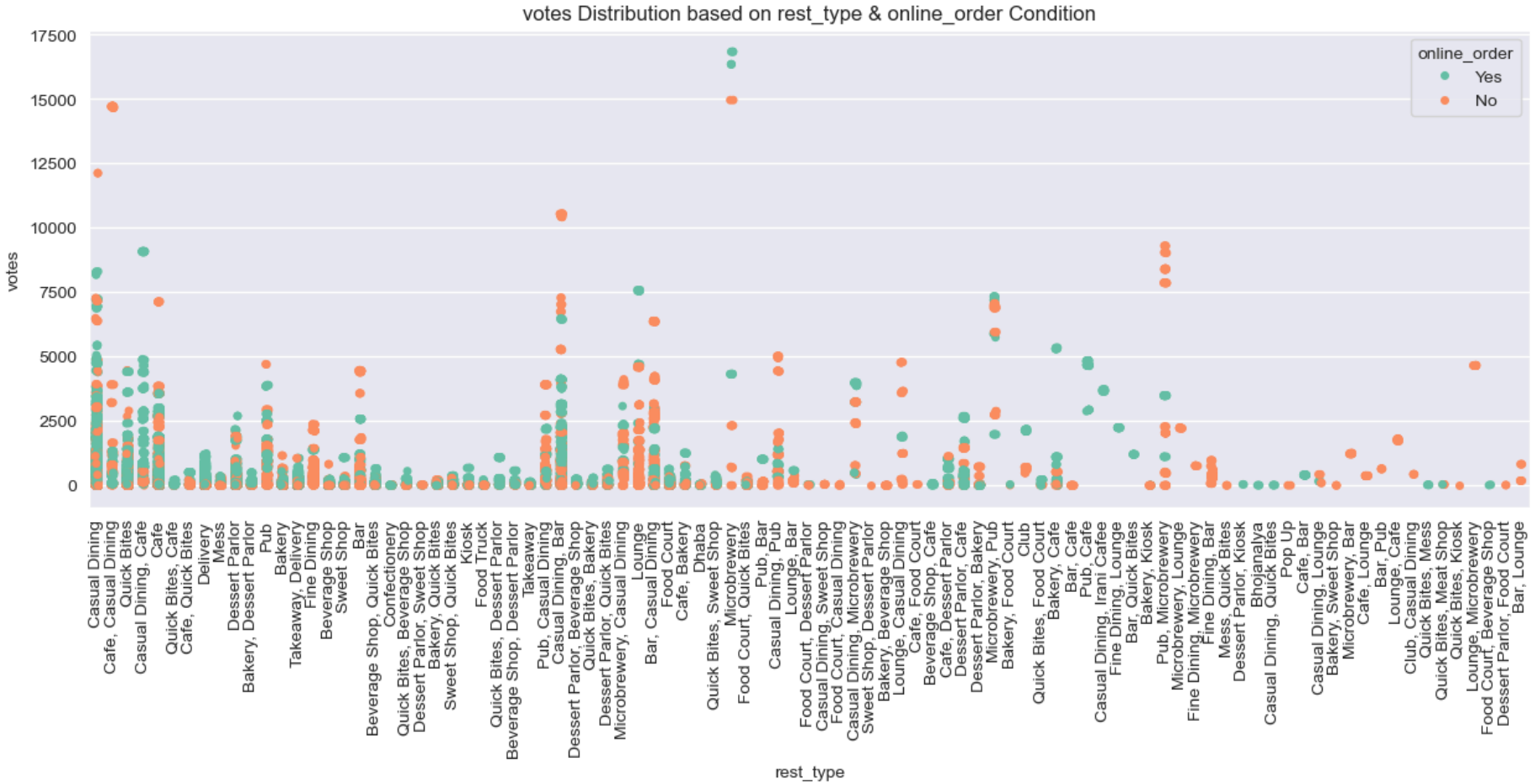


In the above chart as you can see the most of the city Restaurants and booked the table online and get above 4.0 ratings and those who not booked the table online get more than or nearest 3.5 ratings on zomato platform.

In [86]:
```
new_data.columns
```

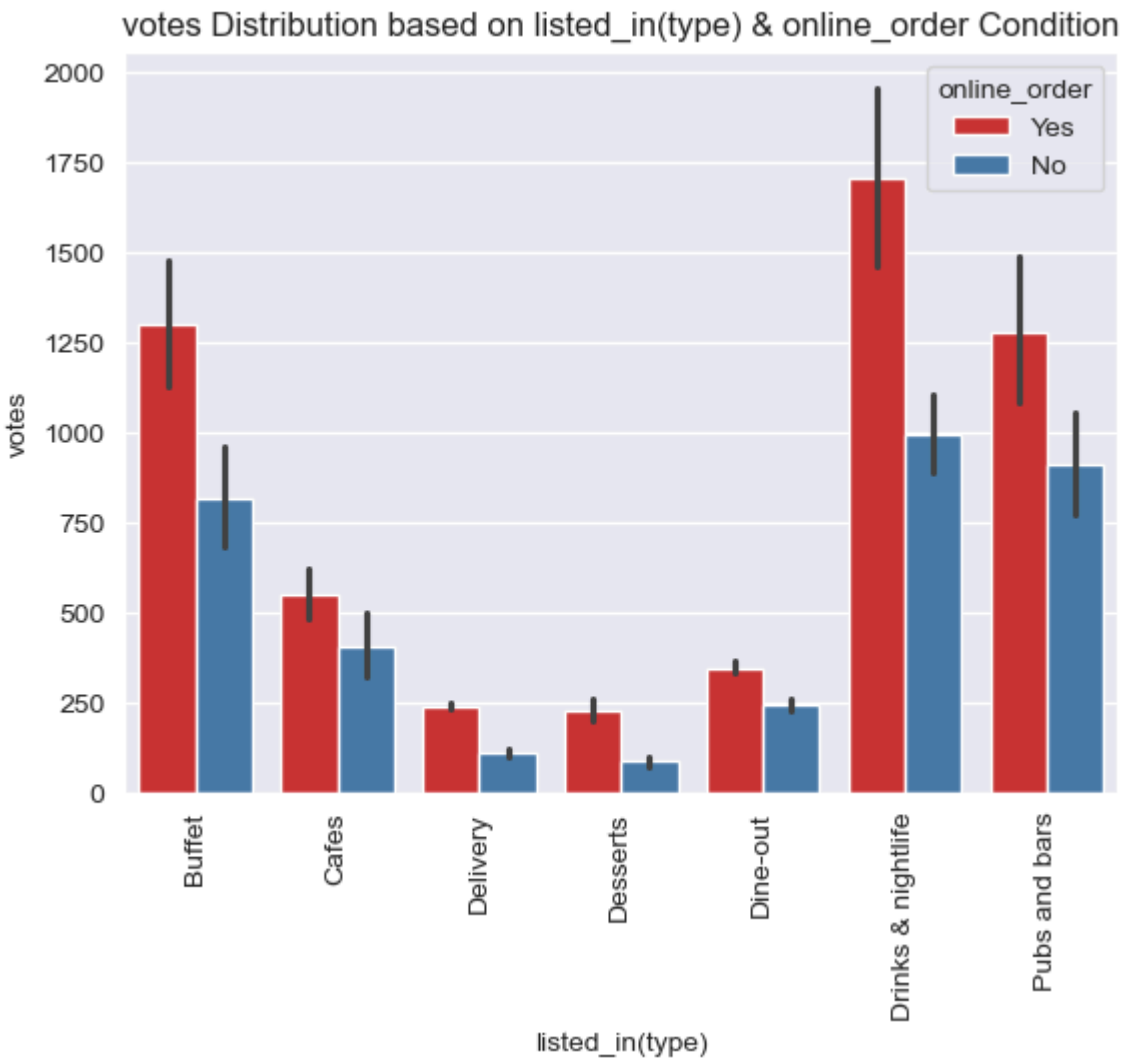
Out[86]:
```
Index(['address', 'name', 'online_order', 'book_table', 'rate', 'votes',
       'location', 'rest_type', 'dish_liked', 'cuisines',
       'approx_cost(for two people)', 'reviews_list', 'menu_item',
       'listed_in(type)', 'listed_in(city)'],
      dtype='object')
```

In [87]:
```
plt.figure(figsize=(15, 5))
sns.set_style("darkgrid")
sns.stripplot(x = "rest_type",y = "votes",hue = "online_order",data = new_data,palette = "Set2")
plt.title("votes Distribution based on rest_type & online_order Condition")
plt.xticks(rotation = "vertical")
plt.show()
```



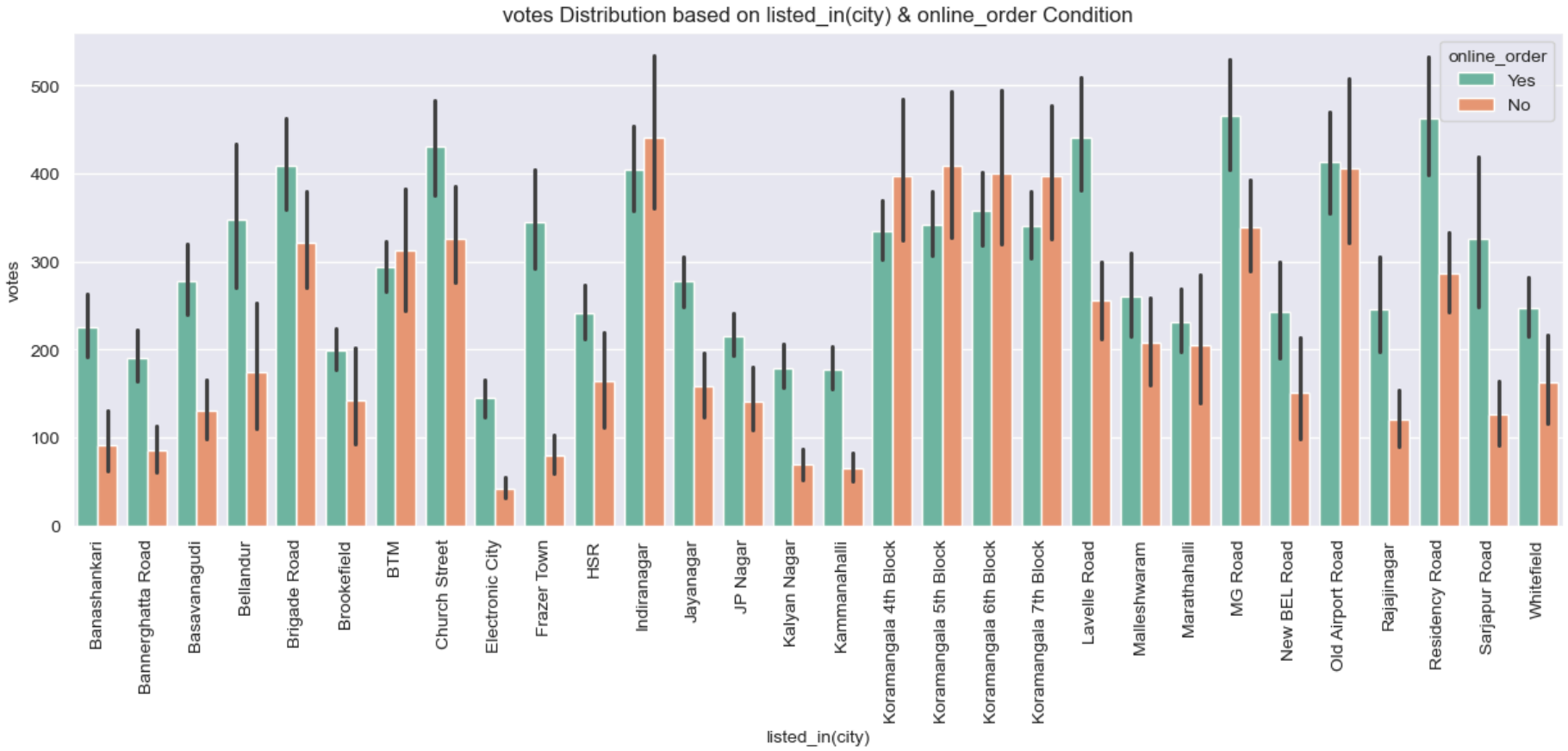
Most of the restaurant type take the online order or not get below 7500 votes on zomato platform.

In [88]:
```
sns.barplot(x="listed_in(type)",y="votes",hue = "online_order",data = new_data,palette = "Set1")
plt.title("votes Distribution based on listed_in(type) & online_order Condition")
plt.xticks(rotation = "vertical")
plt.show()
```



Drinks & Nightlife type Restaurant get the highest Votes cause they taked the online order & second position winner is Pubs and Bars type restaurants on zomato platform.

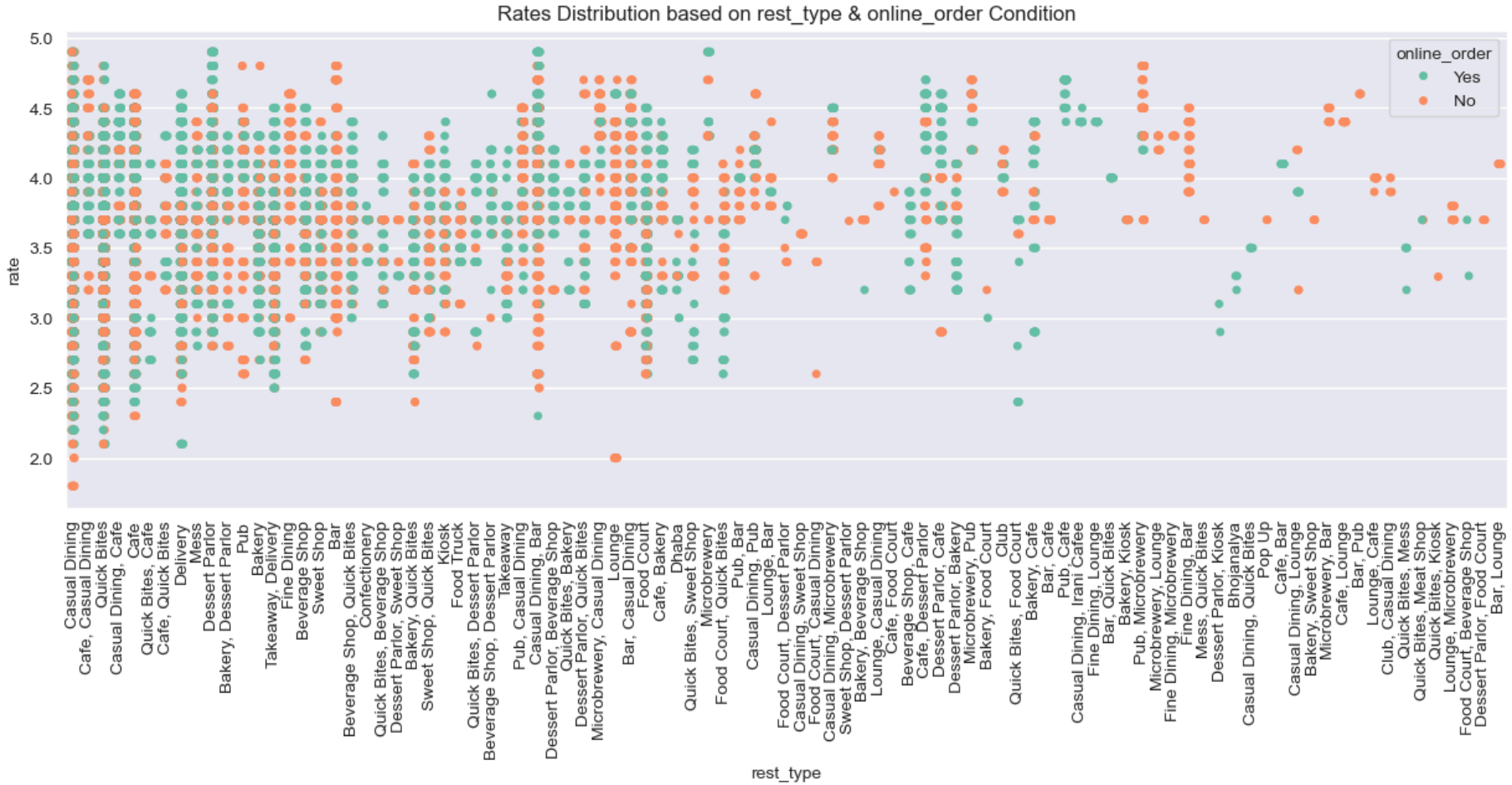
In [89]:
```
plt.figure(figsize=(15, 5))
sns.barplot(x="listed_in(city)",y="votes",hue = "online_order",data = new_data,palette = "Set2")
plt.title("votes Distribution based on listed_in(city) & online_order Condition")
plt.xticks(rotation = "vertical")
plt.show()
```



In the above chart you can see Lavelle Road,MG Road & Residency Road side restaurants get the high votes cause they take the online order but the opposite side interesting thing is Indiranagar side restaurants get the higher votes even not the online order.

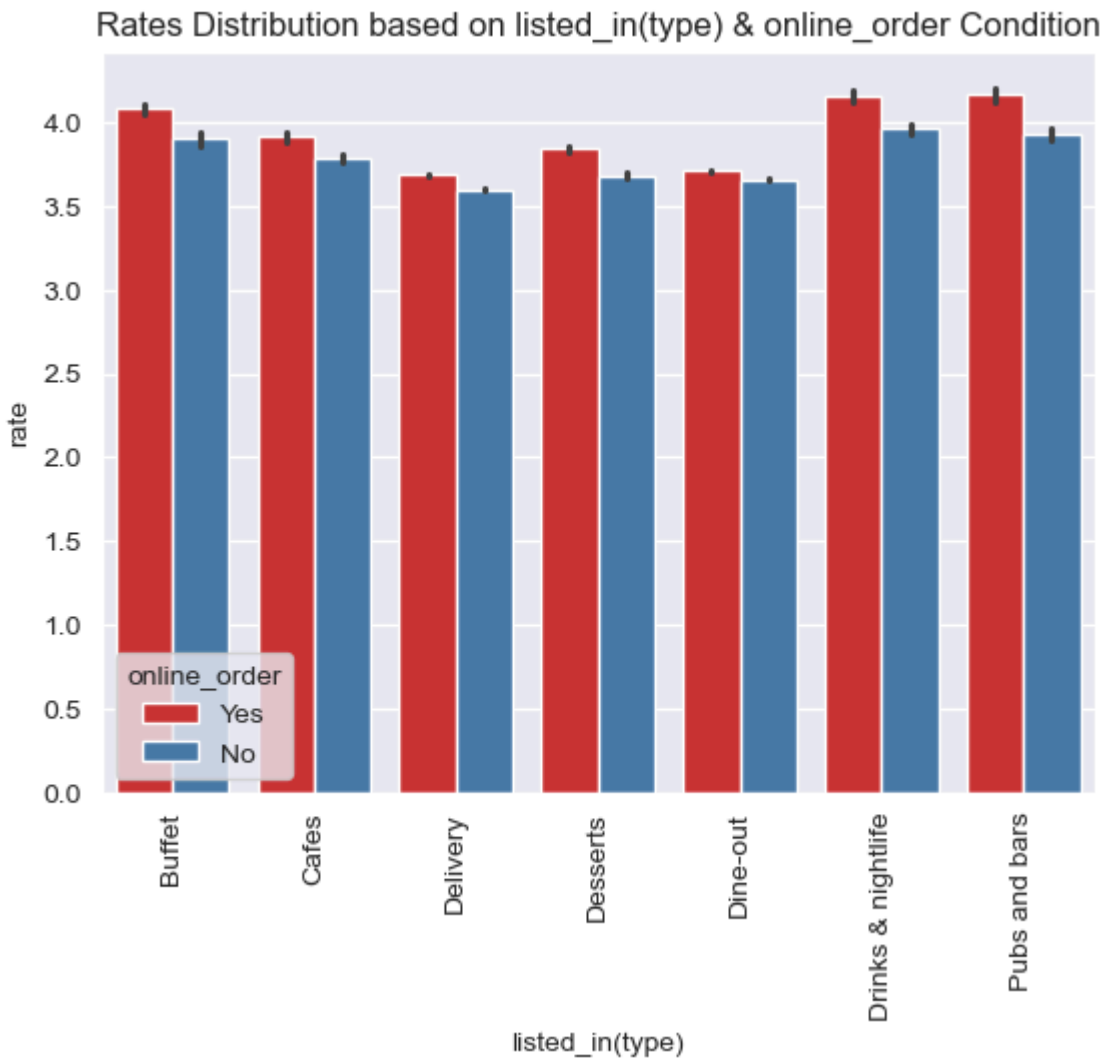
MG-ROAD & RESIDENCY ROAD IS HIGHEST FOR THE ONLINE ORDERS.

In [90]:
```
plt.figure(figsize=(15, 5))
sns.set_style("darkgrid")
sns.stripplot(x = "rest_type",y = "rate",hue = "online_order",data = new_data,palette = "Set2")
plt.title("Rates Distribution based on rest_type & online_order Condition")
plt.xticks(rotation = "vertical")
plt.show()
```



As you can see the most of the restaurants getting the highest ratings cause they take the online orders on zomato platform.

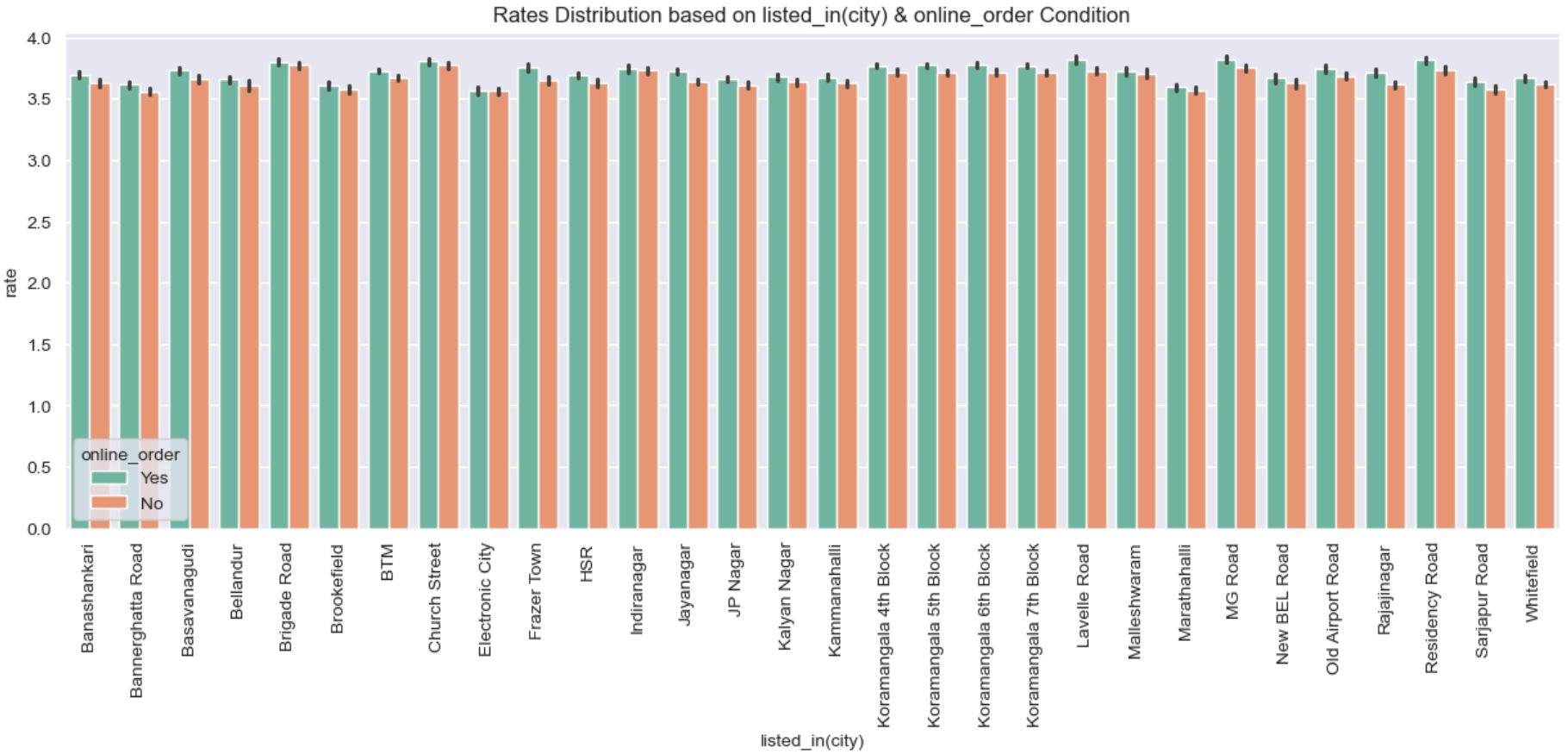
In [91]:
```
sns.barplot(x="listed_in(type)",y="rate",hue = "online_order",data = new_data,palette = "Set1")
plt.title("Rates Distribution based on listed_in(type) & online_order Condition")
plt.xticks(rotation = "vertical")
plt.show()
```



In the most of the restaurant type get Buffet,Drinks & Nightlife,Pubs and Bars get above 4.0 Ratings on the zomato platform.

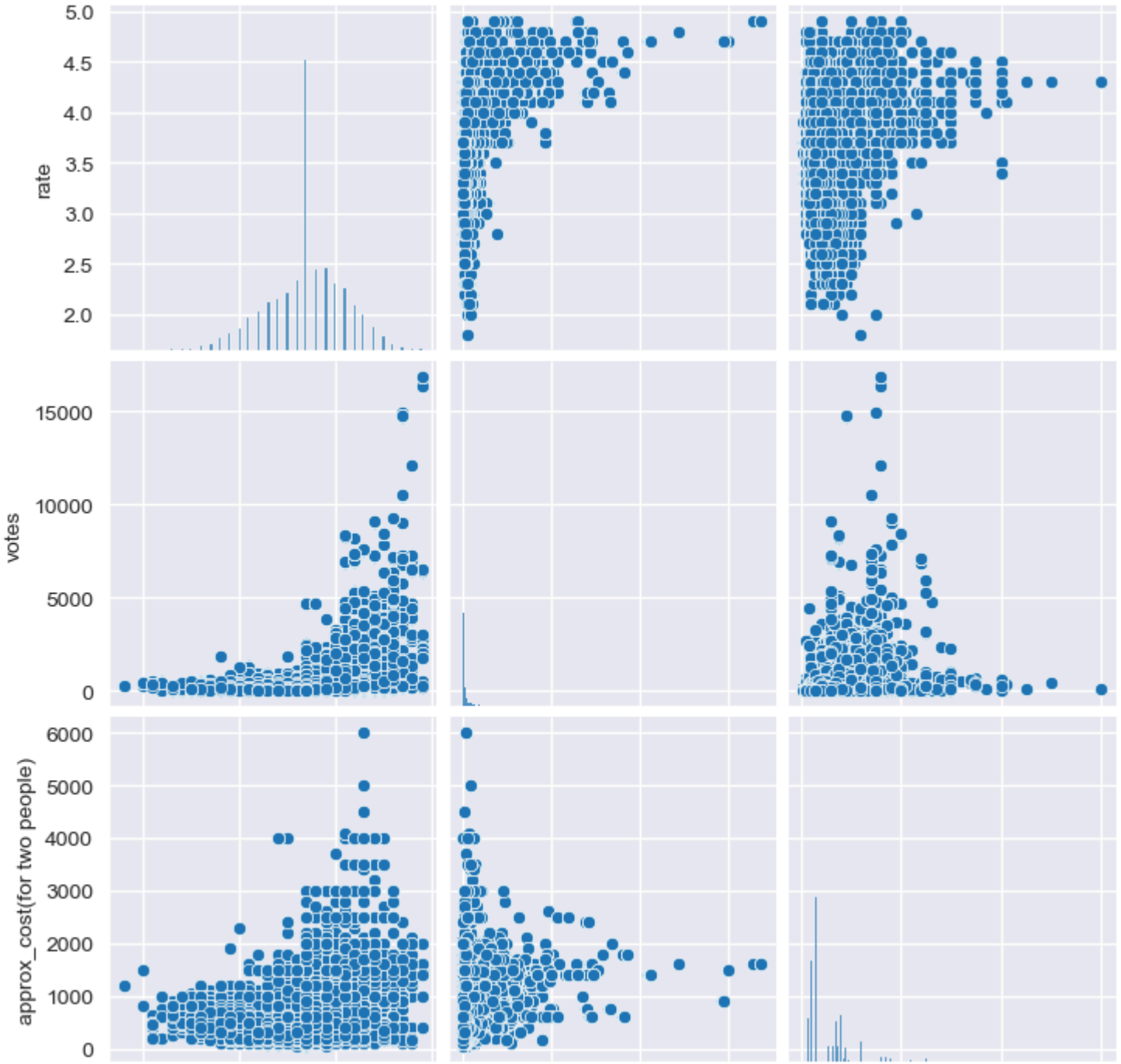
In [92]:
```
plt.figure(figsize=(15, 5))
sns.barplot(x="listed_in(city)",y="rate",hue = "online_order",data = new_data,palette = "Set2")
plt.title("Rates Distribution based on listed_in(city) & online_order Condition")
```

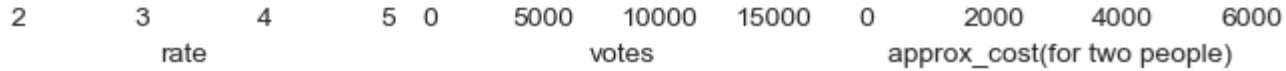
```
plt.xticks(rotation = "vertical")
plt.show()
```



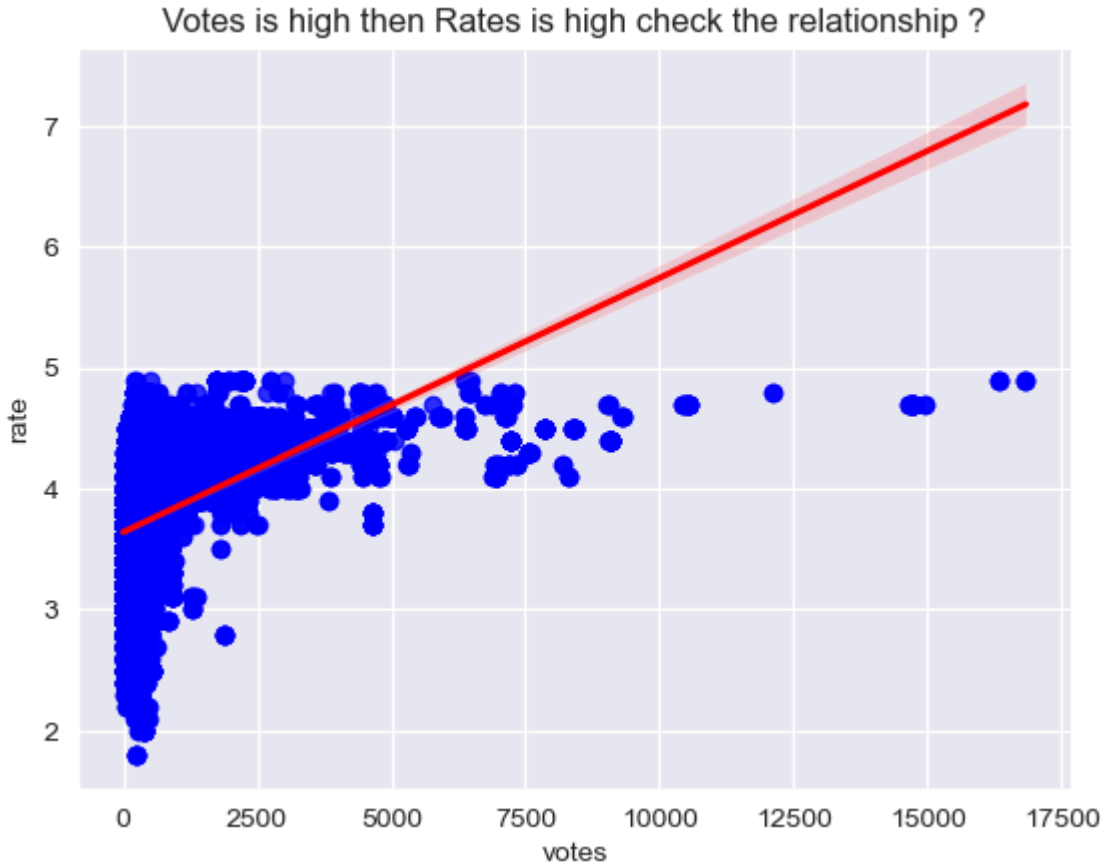
In the most of the city Restaurants below 4.0 rating whatever online order or not.

In [93]:
```
sns.pairplot(new_data)
plt.show()
```


rate
votes
approx_cost(for two people)
5.0
4.5
4.0
3.5
3.0
2.5
2.0
15000
10000
5000
0
6000
5000
4000
3000
2000
1000
0

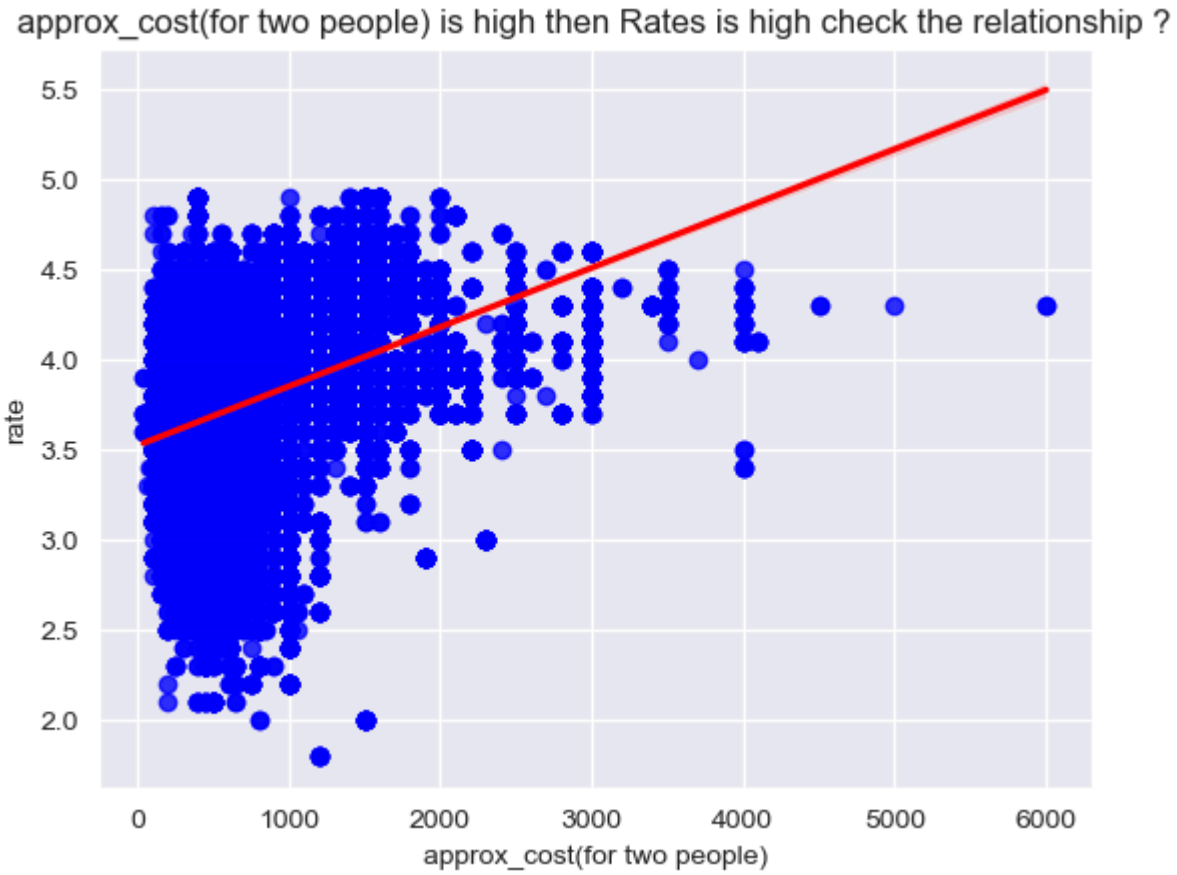


In [94]:
```
# let's check the relationship between two variables
sns.regplot(x = "votes",y="rate",data = new_data,scatter_kws={'color': 'blue'}, line_kws={'color': 'red'})
plt.title("Votes is high then Rates is high check the relationship ?")
plt.show()
```



Votes is high then ratings is low Not strong relationship.

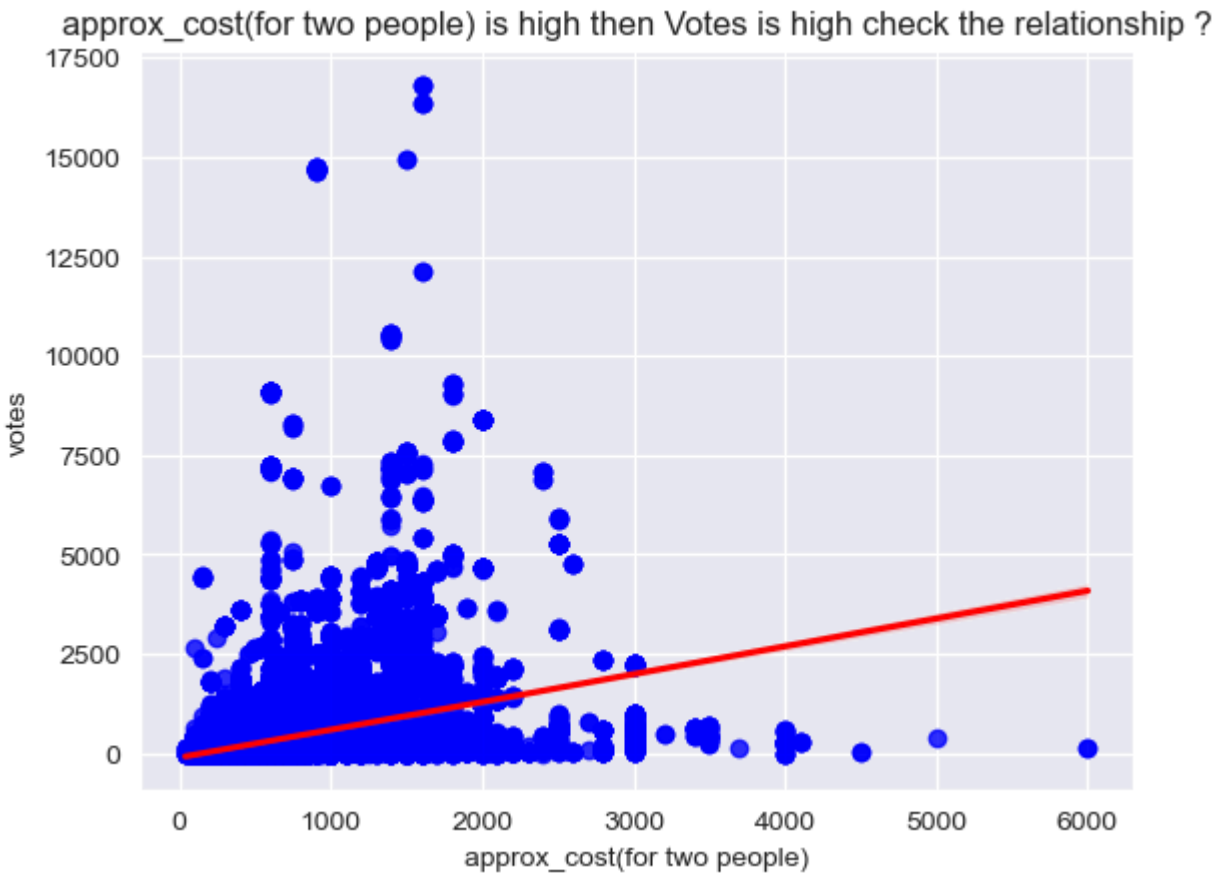
In [95]:
```
# let's check the relationship between two variables
sns.regplot(x = "approx_cost(for two people)",y="rate",data = new_data,scatter_kws={'color': 'blue'}, line_kws={'color': 'red'
plt.title("approx_cost(for two people) is high then Rates is high check the relationship ?")
plt.show()
```



In above scatterplot we can see the approx cost for two people is high then ratings is not much high not strong relationship.

In [96]:
```
# let's check the relationship between two variables
sns.regplot(x = "approx_cost(for two people)",y="votes",data = new_data,scatter_kws={'color': 'blue'}, line_kws={'color': 'red
```

```
plt.title("approx_cost(for two people) is high then Votes is high check the relationship ?")
plt.show()
```



In above scatterplot we can see the approx cost for two people is high then votings is okay-okay medium/average relationship.

In [97]:
```
# let's check the correlation between two variable -1,0,1.one indicates the strong relationship.
relationship = new_data.corr(numeric_only = True)
relationship
```

Out[97]:

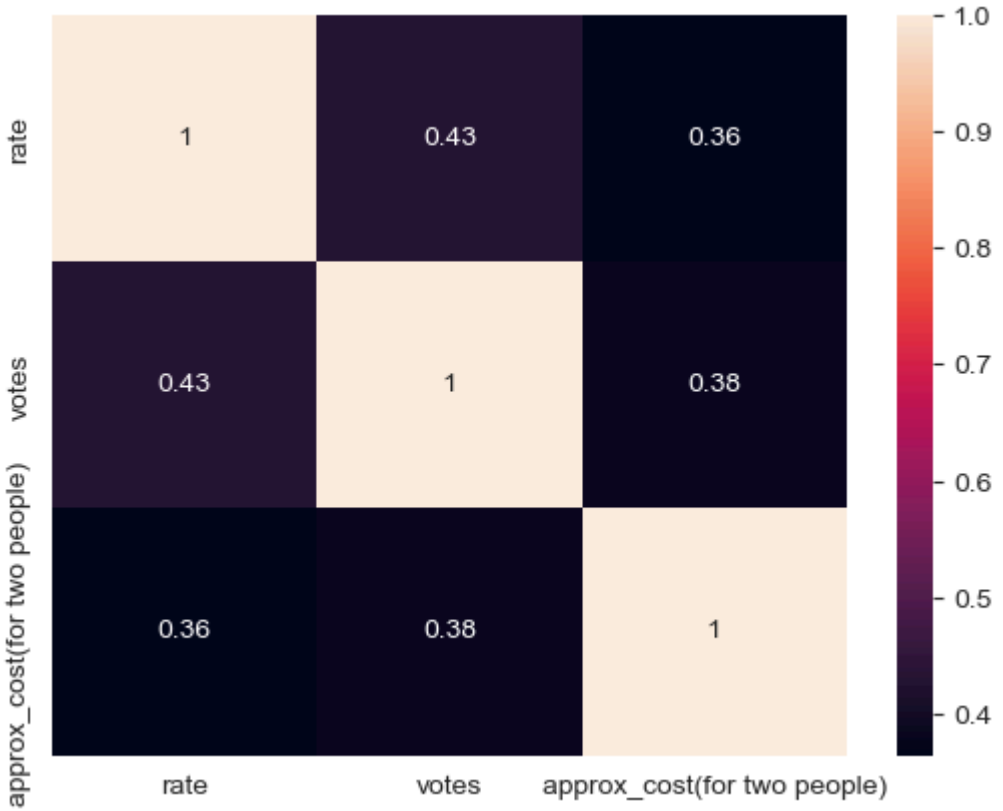
	rate	votes	approx_cost(for two people)
rate	1.000000	0.427366	0.364408
votes	0.427366	1.000000	0.380799
approx_cost(for two people)	0.364408	0.380799	1.000000

In above statistics table you can see the not much strong relationships between the two numerical variables.

In [102…

```
# see the heatmap for visualize the relationship

sns.heatmap(relationship,annot = True )
```

Out[102… <Axes: >



In above Heatmap you can see the not much strong relationships between the two numerical variables.

In [99]:
```
# print all the column names again
new_data.columns
```

Out[99]:
```
Index(['address', 'name', 'online_order', 'book_table', 'rate', 'votes',
       'location', 'rest_type', 'dish_liked', 'cuisines',
       'approx_cost(for two people)', 'reviews_list', 'menu_item',
       'listed_in(type)', 'listed_in(city)'],
      dtype='object')
```

In [100…

```
# now extract the information like name of restaurant get ratings and votes high etc.

max_votes = new_data["votes"].max()
important = new_data[new_data["votes"] == max_votes]
important
```

Out[100…

	address	name	online_order	book_table	rate	votes	location	rest_type	dish_liked	cuisines	approx_cost(for two people)	revie
49170	Behind MK Retail, Sarjapur Road, Bangalore	Byg Brewski Brewing Company	Yes	Yes	4.9	16832	Sarjapur Road	Microbrewery	Cocktails, Butter Chicken, Dahi Kebab, Rajma C...	Continental, North Indian, Italian, South Indi...	1600.0	[('Rat "R Th abso
49627	Behind MK Retail, Sarjapur Road, Bangalore	Byg Brewski Brewing Company	Yes	Yes	4.9	16832	Sarjapur Road	Microbrewery	Cocktails, Butter Chicken, Dahi Kebab, Rajma C...	Continental, North Indian, Italian, South Indi...	1600.0	[('Rat 'R Vi microb
50059	Behind MK Retail, Sarjapur Road, Bangalore	Byg Brewski Brewing Company	Yes	Yes	4.9	16832	Sarjapur Road	Microbrewery	Cocktails, Butter Chicken, Dahi Kebab, Rajma C...	Continental, North Indian, Italian, South Indi...	1600.0	[('Rat 'R Vi microb

Above all are highest voters and rating for this types of the restaurants.

"Zomato Restaurant Dataset – Key Insights & Recommendations":-

1) "Online order is 'Yes'" → Gets "higher votes" and "higher ratings".

2) "Book table is 'Yes'" → Gets "higher votes", "higher ratings", and "higher average cost" for two people.

3) "listed_in(type) = 'Drinks & Nightlife'" → Gets "higher votes", "higher ratings", and "higher average cost".

4) "Listed cities like 'Indiranagar' and 'Old Airport Road'" → Have "higher votes", "ratings", and "cost for two people".

5) If "online order = No", then "approx cost for two people is high" (indicating fine-dine or premium restaurants).

6) "MG Road" and "Residency Road" → Show "higher online orders" and also get "higher votes".

7) Locations like "Brigade Road", "Church Street", "Lavella Road", "MG Road", and "Residency Road" → Show "higher cost for two people" (Premium dining).

8) "Most restaurants have below 7500 votes" — only few exceed this range, indicating top performers.

9) For "restaurant types like 'Drinks & Nightlife', 'Pubs & Bars'", if "book_table = Yes", then "votes are high".

10) In "Bellandur" and "Sarjapur" areas → Votes are "higher" because "book_table = Yes" is more common.

11) Restaurants with "online order enabled" → Tend to get "better ratings".

12) If "table booking = Yes", then "ratings > 3.0" are common;If "table booking = No", most restaurants have ratings "above 2.0", but a few fall "below 2.0".

13) Here,you can see the 'Chicken','Biryani',"Pizza" is a most frequent words while placed order or you can say Most Favourite dishes of the Customers.

14) You can see clearly the winner is in the category of cuisines is 'North'.

15) You can see clearly the most favourite menu item is 'Chicken'.

16) You can see the most of the placed order location is 'Koramangala',& second position is 'BTM'.

17) Most of the order placed location is 'Bangalore' cause whole the dataset belongs to Bangalore.