In [1]:
```
import numpy as np # linear algebra
import pandas as pd
```

In [2]:
```
zmt=pd.read_csv("zomato.csv (1).zip")
```

In [3]:
```
zmt.head()
```

Out[3]:

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

In [4]:
```
zmt.tail()
```

Out[4]:

	url	address	name	online_order	book_table
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
51713	https://www.zomato.com/bangalore/vinod-bar-and...	Number 10, Garudachar Palya, Mahadevapura, Whi...	Vinod Bar And Restaurant	No	No
51714	https://www.zomato.com/bangalore/plunge-sherat...	Sheraton Grand Bengaluru Whitefield Hotel & Co...	Plunge - Sheraton Grand Bengaluru Whitefield H...	No	No
51715	https://www.zomato.com/bangalore/chime-sherato...	Sheraton Grand Bengaluru Whitefield Hotel & Co...	Chime - Sheraton Grand Bengaluru Whitefield Ho...	No	Yes
51716	https://www.zomato.com/bangalore/the-nest-the-...	ITPL Main Road, KIADB Export Promotion Industr...	The Nest - The Den Bengaluru	No	No

In [5]:
```
#checking information of our dataset
```

In [6]:
```
zmt.info()
```

```
<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]:
```
#Here we can see we are having 17 columns and some columns including NaN values and inc
```

Droping Unnecessary Columns

In [8]:
```
#Unnecessary columns are those columns which are not that useful for analysis.
```

In [9]:
```
zmt.drop(['url','reviews_list','menu_item','address','phone','dish_liked'],axis=1,inpla
```

In [10]:
```
zmt.head(5)
```

Out[10]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	approx_cost(for two people)
0	Jalsa	Yes	Yes	4.1/5	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	80(
1	Spice Elephant	Yes	No	4.1/5	787	Banashankari	Casual Dining	Chinese, North Indian, Thai	80(
2	San Churro Cafe	Yes	No	3.8/5	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	80(
3	Addhuri Udupi Bhojana	No	No	3.7/5	88	Banashankari	Quick Bites	South Indian, North Indian	30(
4	Grand Village	No	No	3.8/5	166	Basavanagudi	Casual Dining	North Indian, Rajasthani	60(

In [11]:
```
#Here, we are geeting only those columns which are useful for analysis.
```

Renaming columns

In [12]:
```
zmt.rename(columns={'name':'restaurants','rate':'rating','cuisines':'food_type','listed_
                    'city','approx_cost(for two people)':'cost'},inplace=True)
```

In [13]:
```
zmt.head(5)
```

Out[13]:

	restaurants	online_order	book_table	rating	votes	location	rest_type	food_type	cost	typ
0	Jalsa	Yes	Yes	4.1/5	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800	Buff
1	Spice Elephant	Yes	No	4.1/5	787	Banashankari	Casual Dining	Chinese, North Indian, Thai	800	Buff
2	San Churro Cafe	Yes	No	3.8/5	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	800	Buff
3	Addhuri Udupi Bhojana	No	No	3.7/5	88	Banashankari	Quick Bites	South Indian, North Indian	300	Buff
4	Grand Village	No	No	3.8/5	166	Basavanagudi	Casual Dining	North Indian, Rajasthani	600	Buff

Droping NaN values

In [14]:
```
zmt.dropna(inplace=True) # Droping NaN values for better result using dropna()
```

Cleaning Individual Columns

In [15]:
```
zmt.groupby('restaurants').count().head() # we are grouping the restaurants to see the r
```

Out[15]:

restaurants	online_order	book_table	rating	votes	location	rest
#FeelTheROLL	2	2	2	2	2	
#L-81 Cafe	9	9	9	9	9	
#refuel	3	3	3	3	3	
1000 B.C	6	6	6	6	6	
100Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â°C	3	3	3	3	3	

In [16]:
```
#Here we can see we are got restaurant names having disturbed characters

#so we need to remove that characters from the name using pattern matcing to get orignal
```

In [17]:
```
# '[Ãx][^A-Za-z]+' pattern I got to find disturbed characters using regex.And Replacing
zmt['restaurants']=zmt['restaurants'].str.replace('[Ãx][^A-Za-z]+','',regex=True)
```

In [18]:
```
zmt.groupby('restaurants').count().head()
```

Out[18]:

restaurants	online_order	book_table	rating	votes	location	rest_type	food_type	cost	type	city
#FeelTheROLL	2	2	2	2	2	2	2	2	2	2
#L-81 Cafe	9	9	9	9	9	9	9	9	9	9
#refuel	3	3	3	3	3	3	3	3	3	3
1000 B.C	6	6	6	6	6	6	6	6	6	6
100C	3	3	3	3	3	3	3	3	3	3

In [19]:
```
#Here see we successfully removed the disturbed characters from restaurant name(refer 5t

#By reading the databse I found the names having incorrect spell. Let me correct them as
```

In [20]:
```
zmt['restaurants']=zmt['restaurants'].str.replace('Caf-|Caf ','Cafe ',regex=True)
# replacing the Caf- or Caf names with cafe using pattern matching
```

In [21]:
```
zmt['online_order'].unique()  # Using .unique() we can find the unique values from colum
```

Out[21]:
```
array(['Yes', 'No'], dtype=object)
```

In [22]:
```
zmt['book_table'].unique()
```

Out[22]:
```
array(['Yes', 'No'], dtype=object)
```

In [23]:
```
zmt['rating'].unique()
```

Out[23]:
```
array(['4.1/5', '3.8/5', '3.7/5', '3.6/5', '4.6/5', '4.0/5', '4.2/5',
       '3.9/5', '3.1/5', '3.0/5', '3.2/5', '3.3/5', '2.8/5', '4.4/5',
       '4.3/5', 'NEW', '2.9/5', '3.5/5', '2.6/5', '3.8 /5', '3.4/5',
       '4.5/5', '2.5/5', '2.7/5', '4.7/5', '2.4/5', '2.2/5', '2.3/5',
       '3.4 /5', '-', '3.6 /5', '4.8/5', '3.9 /5', '4.2 /5', '4.0 /5',
       '4.1 /5', '3.7 /5', '3.1 /5', '2.9 /5', '3.3 /5', '2.8 /5',
       '3.5 /5', '2.7 /5', '2.5 /5', '3.2 /5', '2.6 /5', '4.5 /5',
       '4.3 /5', '4.4 /5', '4.9/5', '2.1/5', '2.0/5', '1.8/5', '4.6 /5',
       '4.9 /5', '3.0 /5', '4.8 /5', '2.3 /5', '4.7 /5', '2.4 /5',
       '2.1 /5', '2.2 /5', '2.0 /5', '1.8 /5'], dtype=object)
```

Here, we can see rating column having string values and having object datatype including 'NEW' and '-' values.we know that rating is always out of 5 and in decimal format. So, we need to clean our 'rating' columns.

Need changes:

Remove 'NEW' and '-' values. Remove '/5' from the rating. Covert datatype object to float (float_datatype has decimal values). Try to do all changes in one block of code

In [24]:
```
replace=lambda x:x.replace('/5',"") # lambda function to replace /5 to empty string
l=[] # list to store cleaned values
```

```
for val in map(replace, zmt['rating']): # map function to read data from column and repl
    if val!='NEW' and val!='-':  # excluding 'NEW' and '-' values
        var=float(val) # converting the result in float datatype and storing into one va
    l.append(var) # appending cleaned values in created list
zmt['rating']=l  # updating rating column with new and cleaned values
```

In [25]:
```
zmt['rating'].unique(),zmt['rating'].dtype
```

Out[25]:
```
(array([4.1, 3.8, 3.7, 3.6, 4.6, 4. , 4.2, 3.9, 3.1, 3. , 3.2, 3.3, 2.8,
        4.4, 4.3, 2.9, 3.5, 2.6, 3.4, 4.5, 2.5, 2.7, 4.7, 2.4, 2.2, 2.3,
        4.8, 4.9, 2.1, 2. , 1.8]),
 dtype('float64'))
```

In [26]:
```
zmt['votes'].isnull().value_counts()# checking column having null values or not using .t
```

Out[26]:
```
False    43533
Name: votes, dtype: int64
```

In [27]:
```
zmt.location.unique()
```

Out[27]:
```
array(['Banashankari', 'Basavanagudi', 'Mysore Road', 'Jayanagar',
       'Kumaraswamy Layout', 'Rajarajeshwari Nagar', 'Vijay Nagar',
       'Uttarahalli', 'JP Nagar', 'South Bangalore', 'City Market',
       'Bannerghatta Road', 'BTM', 'Kanakapura Road', 'Bommanahalli',
       'CV Raman Nagar', 'Electronic City', 'Wilson Garden',
       'Shanti Nagar', 'Koramangala 5th Block', 'Richmond Road', 'HSR',
       'Marathahalli', 'Koramangala 7th Block', 'Bellandur',
       'Sarjapur Road', 'Whitefield', 'East Bangalore',
       'Old Airport Road', 'Indiranagar', 'Koramangala 1st Block',
       'Frazer Town', 'MG Road', 'Brigade Road', 'Lavelle Road',
       'Church Street', 'Ulsoor', 'Residency Road', 'Shivajinagar',
       'Infantry Road', 'St. Marks Road', 'Cunningham Road',
       'Race Course Road', 'Commercial Street', 'Vasanth Nagar', 'Domlur',
       'Koramangala 8th Block', 'Ejipura', 'Jeevan Bhima Nagar',
       'Old Madras Road', 'Seshadripuram', 'Kammanahalli',
       'Koramangala 6th Block', 'Majestic', 'Langford Town',
       'Central Bangalore', 'Sanjay Nagar', 'Brookefield',
       'ITPL Main Road, Whitefield', 'Varthur Main Road, Whitefield',
       'Koramangala 2nd Block', 'Koramangala 3rd Block',
       'Koramangala 4th Block', 'Koramangala', 'Hosur Road',
       'Rajajinagar', 'RT Nagar', 'Banaswadi', 'North Bangalore',
       'Nagawara', 'Hennur', 'Kalyan Nagar', 'HBR Layout',
       'Rammurthy Nagar', 'Thippasandra', 'Kaggadasapura', 'Hebbal',
       'Kengeri', 'New BEL Road', 'Sankey Road', 'Malleshwaram',
       'Sadashiv Nagar', 'Basaveshwara Nagar', 'Yeshwantpur',
       'West Bangalore', 'Magadi Road', 'Yelahanka', 'Sahakara Nagar',
       'Jalahalli', 'Nagarbhavi', 'Peenya', 'KR Puram'], dtype=object)
```

In [28]:
```
zmt.rest_type.unique(), zmt.rest_type.isnull().value_counts()
```

Out[28]:
```
(array(['Casual Dining', 'Cafe, Casual Dining', 'Quick Bites',
       'Casual Dining, Cafe', 'Cafe', 'Quick Bites, Cafe',
       'Cafe, Quick Bites', 'Delivery', 'Mess', 'Dessert Parlor',
       'Bakery, Dessert Parlor', 'Pub', 'Bakery', 'Takeaway, Delivery',
       'Fine Dining', 'Beverage Shop', 'Sweet Shop', 'Bar',
       'Dessert Parlor, Sweet Shop', 'Bakery, Quick Bites',
       'Sweet Shop, Quick Bites', 'Kiosk', 'Food Truck',
       'Quick Bites, Dessert Parlor', 'Beverage Shop, Quick Bites',
       'Beverage Shop, Dessert Parlor', 'Takeaway', 'Pub, Casual Dining',
       'Casual Dining, Bar', 'Dessert Parlor, Beverage Shop',
       'Quick Bites, Bakery', 'Microbrewery, Casual Dining', 'Lounge',
       'Bar, Casual Dining', 'Food Court', 'Cafe, Bakery', 'Dhaba',
       'Quick Bites, Sweet Shop', 'Microbrewery',
       'Food Court, Quick Bites', 'Quick Bites, Beverage Shop',
       'Pub, Bar', 'Casual Dining, Pub', 'Lounge, Bar',
       'Dessert Parlor, Quick Bites', 'Food Court, Dessert Parlor',
       'Casual Dining, Sweet Shop', 'Food Court, Casual Dining',
       'Casual Dining, Microbrewery', 'Lounge, Casual Dining',
       'Cafe, Food Court', 'Beverage Shop, Cafe', 'Cafe, Dessert Parlor',
       'Dessert Parlor, Cafe', 'Dessert Parlor, Bakery',
       'Microbrewery, Pub', 'Bakery, Food Court', 'Club',
       'Quick Bites, Food Court', 'Bakery, Cafe', 'Pub, Cafe',
       'Casual Dining, Irani Cafee', 'Fine Dining, Lounge',
       'Bar, Quick Bites', 'Confectionery', 'Pub, Microbrewery',
       'Microbrewery, Lounge', 'Fine Dining, Microbrewery',
       'Fine Dining, Bar', 'Dessert Parlor, Kiosk', 'Bhojanalya',
       'Casual Dining, Quick Bites', 'Cafe, Bar', 'Casual Dining, Lounge',
       'Bakery, Beverage Shop', 'Microbrewery, Bar', 'Cafe, Lounge',
       'Bar, Pub', 'Lounge, Cafe', 'Club, Casual Dining',
       'Quick Bites, Mess', 'Quick Bites, Meat Shop',
       'Quick Bites, Kiosk', 'Lounge, Microbrewery',
       'Food Court, Beverage Shop', 'Dessert Parlor, Food Court',
       'Bar, Lounge'], dtype=object),
 False    43533
 Name: rest_type, dtype: int64)
```

In [29]:
```
zmt.food_type.unique(), zmt.food_type.isnull().value_counts()
```

Out[29]:
```
(array(['North Indian, Mughlai, Chinese', 'Chinese, North Indian, Thai',
       'Cafe, Mexican, Italian', ..., 'Tibetan, Nepalese',
       'North Indian, Street Food, Biryani',
       'North Indian, Chinese, Arabian, Momos'], dtype=object),
 False    43533
 Name: food_type, dtype: int64)
```

In [30]:
```
zmt.cost.unique()
```

Out[30]:
```
array(['800', '300', '600', '700', '550', '500', '450', '650', '400',
       '900', '200', '750', '150', '850', '100', '1,200', '350', '250',
       '950', '1,000', '1,500', '1,300', '199', '80', '1,100', '160',
       '1,600', '230', '130', '1,700', '1,400', '1,350', '2,200', '2,000',
       '1,800', '1,900', '180', '330', '2,500', '2,100', '3,000', '2,800',
       '3,400', '50', '40', '1,250', '3,500', '4,000', '2,400', '2,600',
       '1,450', '70', '3,200', '560', '240', '360', '6,000', '1,050',
       '2,300', '4,100', '120', '5,000', '3,700', '1,650', '2,700',
       '4,500'], dtype=object)
```

Here see cost column string values with ',' values. And we know cost is always in int sp we need to make some transformations.

Changes need:

Remove ',' from values Change datatype object to integer(int)

In [31]:
```
zmt['cost']= zmt['cost'].apply(lambda x:x.replace(",","")).astype(int) # we can also app
```

In [32]:
```
zmt.cost.unique(),zmt.cost.dtype
```

Out[32]:
```
(array([ 800,  300,  600,  700,  550,  500,  450,  650,  400,  900,  200,
         750,  150,  850,  100, 1200,  350,  250,  950, 1000, 1500, 1300,
         199,   80, 1100,  160, 1600,  230,  130, 1700, 1400, 1350, 2200,
        2000, 1800, 1900,  180,  330, 2500, 2100, 3000, 2800, 3400,   50,
          40, 1250, 3500, 4000, 2400, 2600, 1450,   70, 3200,  560,  240,
         360, 6000, 1050, 2300, 4100,  120, 5000, 3700, 1650, 2700, 4500]),
 dtype('int32'))
```

Here, Using lambda function we replaced ',' to empty string and using astype(int) we converted datatype object to int and update in the cost column and we got cleaned data.

In [33]:
```
zmt.type.unique()
```

Out[33]:
```
array(['Buffet', 'Cafes', 'Delivery', 'Desserts', 'Dine-out',
       'Drinks & nightlife', 'Pubs and bars'], dtype=object)
```

In [34]:
```
zmt.city.unique()
```

Out[34]:
```
array(['Banashankari', 'Bannerghatta Road', 'Basavanagudi', 'Bellandur',
       'Brigade Road', 'Brookefield', 'BTM', 'Church Street',
       'Electronic City', 'Frazer Town', 'HSR', 'Indiranagar',
       'Jayanagar', 'JP Nagar', 'Kalyan Nagar', 'Kammanahalli',
       'Koramangala 4th Block', 'Koramangala 5th Block',
       'Koramangala 6th Block', 'Koramangala 7th Block', 'Lavelle Road',
       'Malleshwaram', 'Marathahalli', 'MG Road', 'New BEL Road',
       'Old Airport Road', 'Rajajinagar', 'Residency Road',
       'Sarjapur Road', 'Whitefield'], dtype=object)
```

Droping Duplicate

Dropping duplicates means we are removing repeated values or duplicate values from the dataset

In [35]:
```
zmt.duplicated().value_counts()
```

Out[35]:
```
False    43453
True        80
dtype: int64
```

We got 80 duplicate values in our dataset we need to remove those.

In [36]:
```
zmt.drop_duplicates(keep='last',inplace=True)
zmt.reset_index(drop=True,inplace=True) # reseting index
```

In [37]:
```
zmt.duplicated().value_counts()
```

Out[37]:
```
False    43453
dtype: int64
```

In [38]:
```
zmt
```

Out[38]:

	restaurants	online_order	book_table	rating	votes	location	rest_type	food_type	cost
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800
1	Spice Elephant	Yes	No	4.1	787	Banashankari	Casual Dining	Chinese, North Indian, Thai	800
2	San Churro Cafe	Yes	No	3.8	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	800
3	Addhuri Udupi Bhojana	No	No	3.7	88	Banashankari	Quick Bites	South Indian, North Indian	300
4	Grand Village	No	No	3.8	166	Basavanagudi	Casual Dining	North Indian, Rajasthani	600
...	...	...	...	...	...	...	...	...	...
43448	The Farm House Bar n Grill	No	No	3.7	34	Whitefield	Casual Dining, Bar	North Indian, Continental	800
43449	Bhagini	No	No	2.5	81	Whitefield	Casual Dining, Bar	Andhra, South Indian, Chinese, North Indian	800
43450	Best Brews - Four Points by Sheraton Bengaluru...	No	No	3.6	27	Whitefield	Bar	Continental	1500
43451	Chime - Sheraton Grand Bengaluru Whitefield Ho...	No	Yes	4.3	236	ITPL Main Road, Whitefield	Bar	Finger Food	2500
43452	The Nest - The Den Bengaluru	No	No	3.4	13	ITPL Main Road, Whitefield	Bar, Casual Dining	Finger Food, North Indian, Continental	1500

43453 rows × 11 columns

In [39]:
```
zmt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43453 entries, 0 to 43452
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   restaurants   43453 non-null  object 
 1   online_order  43453 non-null  object 
 2   book_table    43453 non-null  object 
 3   rating        43453 non-null  float64
 4   votes         43453 non-null  int64  
 5   location      43453 non-null  object 
 6   rest_type     43453 non-null  object 
 7   food_type     43453 non-null  object 
 8   cost          43453 non-null  int32  
 9   type          43453 non-null  object 
 10  city          43453 non-null  object 
dtypes: float64(1), int32(1), int64(1), object(8)
memory usage: 3.5+ MB
```

Data Visualization

In [40]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

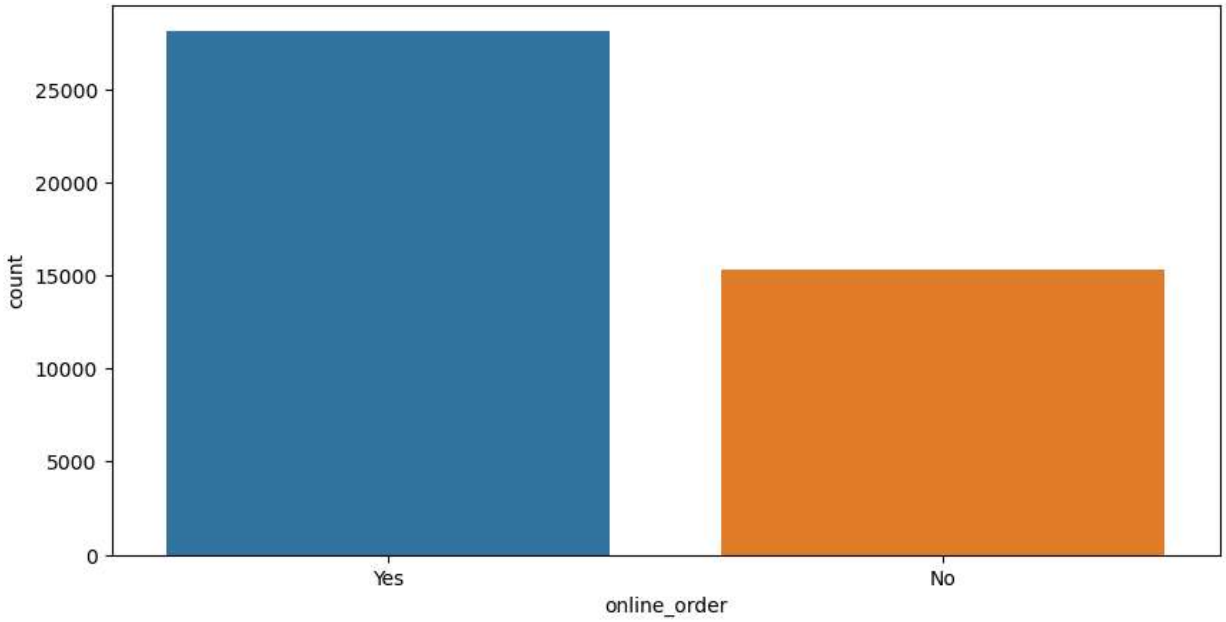
How many restaurants accpeting online order for zomato

In [41]:
```
zmt.head(1) # see the name of the cloumns
```

Out[41]:

	restaurants	online_order	book_table	rating	votes	location	rest_type	food_type	cost	typ
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800	Buffe

In [42]:
```
plt.figure(figsize=(10,5))
sns.countplot(x='online_order',data=zmt) # countplot gives us count of the values
plt.show()
```



By observing the graph, we can say that 25000+ restaurants are accepting online order for zomato and 15000 restaurants not accepting online orders.

Find best location by seeing dataset

We have two columns 'rating' and 'votes' so by geeting the average of those columns with repectto 'location' we can find the best location

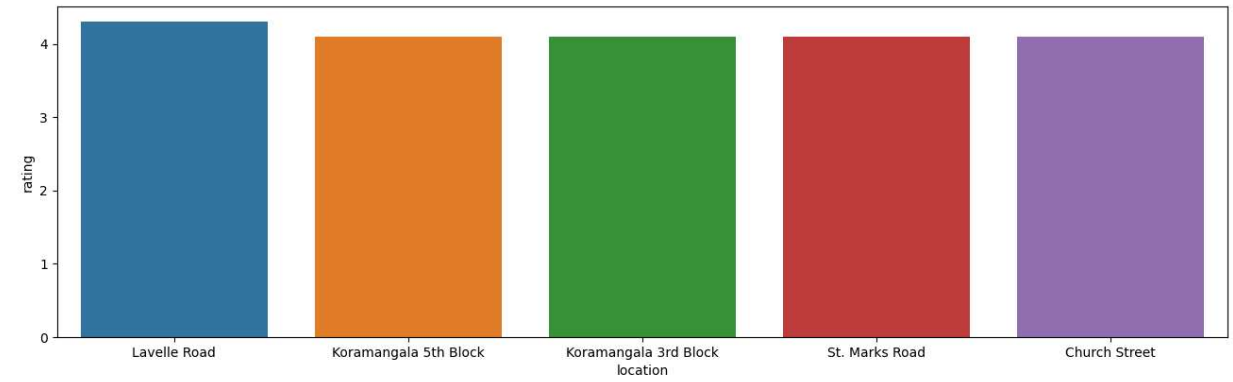
In [43]:
```
zmt.head(1)
```

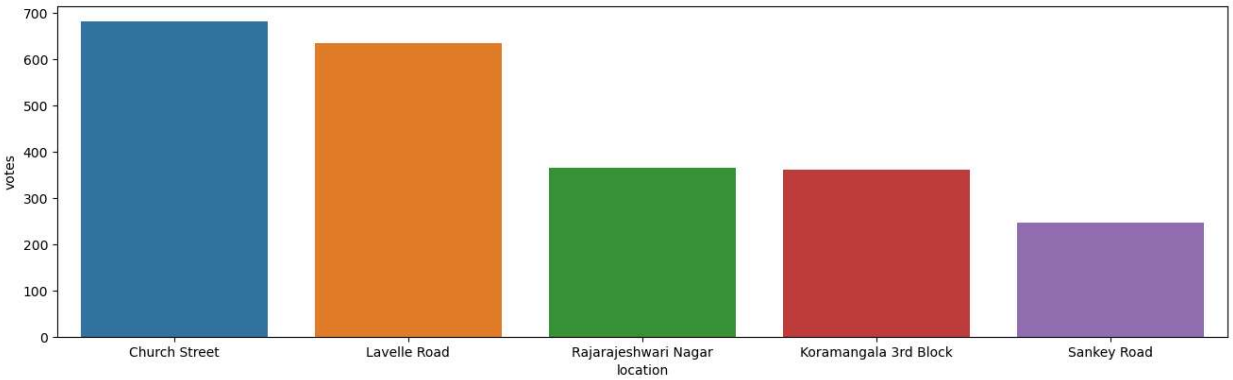
Out[43]:

	restaurants	online_order	book_table	rating	votes	location	rest_type	food_type	cost	typ
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800	Buffe

In [44]:
```
plt.figure(figsize=(16,10)) # seeting graph size using matplotlib
ax = plt.subplot(2,1,1)   # use of Subploting
loc_rating=zmt.groupby('location').agg({'rating':'median'})
rating_sorted_loc=loc_rating.sort_values('rating',ascending=False).head(5).reset_index()
sns.barplot(x='location',y='rating',data=rating_sorted_loc) # ploting graph using seabor

ax = plt.subplot(2,1,2)
loc_votes=zmt.groupby('location').agg({'votes':'median'})
votes_sorted_loc=loc_votes.sort_values('votes',ascending=False).head(5).reset_index() #
sns.barplot(x='location',y='votes',
            data=votes_sorted_loc)
plt.show()
```





By observing the above graphs,'levelle Road' has an high rating and votes as well as compare to other.

So, we can say that 'levelle Road' is a best location by comparing 'votes' and 'rating' of locations.

Find Types of restraunts and their count.

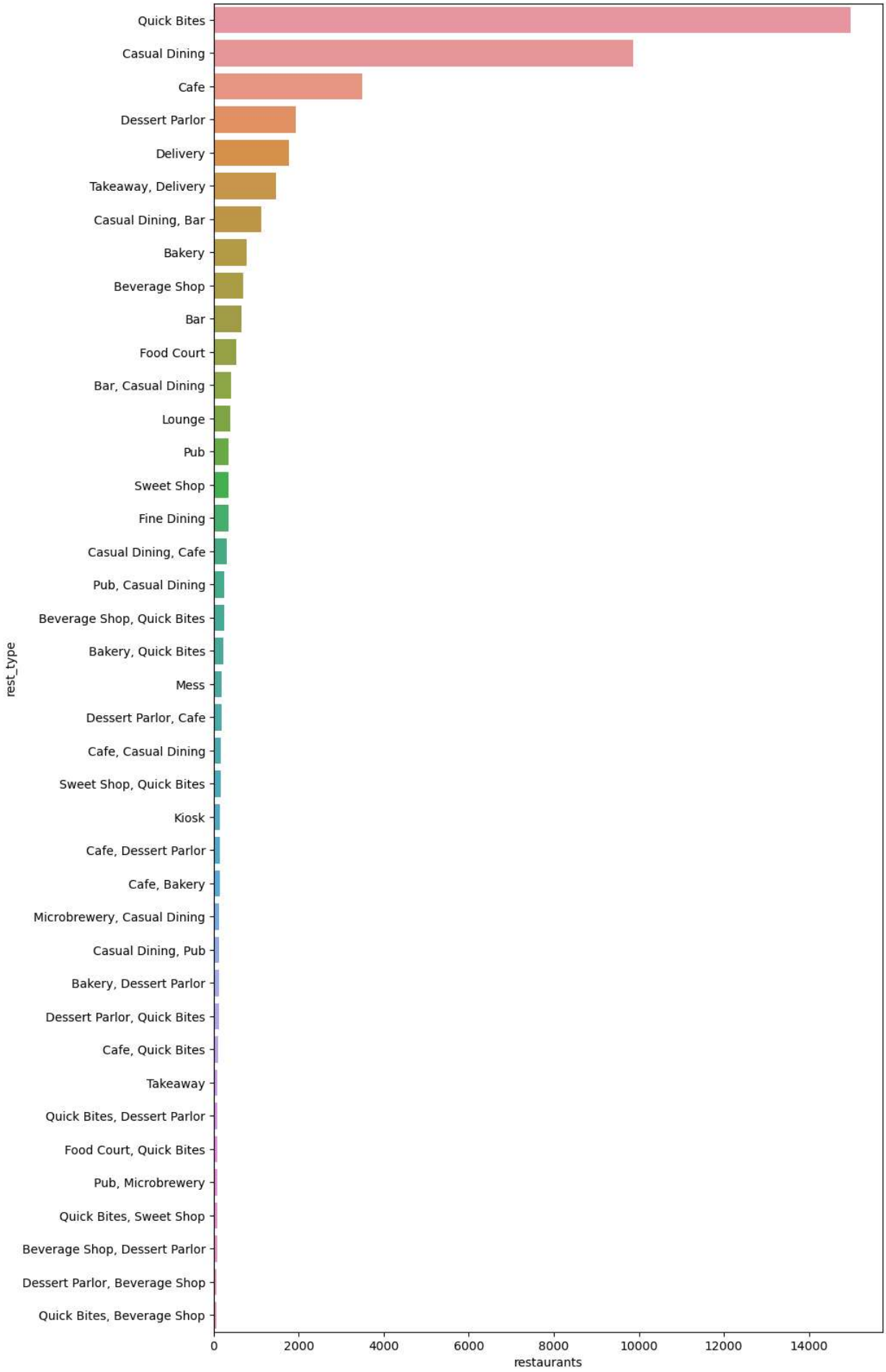
In [46]:
```
zmt.head(1)
```

Out[46]:

	restaurants	online_order	book_table	rating	votes	location	rest_type	food_type	cost	typ
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800	Buffe

In [47]:
```
plt.figure(figsize=(10,20))
rest_types=zmt.groupby('rest_type')['restaurants'].count().reset_index()
# grouping restaurants types and getting 'restaurant' count
sorted_rest_types=rest_types.sort_values('restaurants',ascending=False).head(40)
# sorting restaurant types on 'restaurants' counts in descending order and reading 40 v
rest_types.rest_type.count(),sns.barplot(x='restaurants',y='rest_type',data=sorted_rest_
# drawing horizontal bar plot to see results
```

Out[47]:
```
(87, <AxesSubplot:xlabel='restaurants', ylabel='rest_type'>)
```


Quick Bites
Casual Dining
Cafe
Dessert Parlor
Delivery
Takeaway, Delivery
Casual Dining, Bar
Bakery
Beverage Shop
Bar
Food Court
Bar, Casual Dining
Lounge
Pub
Sweet Shop
Fine Dining
Casual Dining, Cafe
Pub, Casual Dining
Beverage Shop, Quick Bites
Bakery, Quick Bites
Mess
Dessert Parlor, Cafe
Cafe, Casual Dining
Sweet Shop, Quick Bites
Kiosk
Cafe, Dessert Parlor
Cafe, Bakery
Microbrewery, Casual Dining
Casual Dining, Pub
Bakery, Dessert Parlor
Dessert Parlor, Quick Bites
Cafe, Quick Bites
Takeaway
Quick Bites, Dessert Parlor
Food Court, Quick Bites
Pub, Microbrewery
Quick Bites, Sweet Shop
Beverage Shop, Dessert Parlor
Dessert Parlor, Beverage Shop
Quick Bites, Beverage Shop
rest_type
0
2000
4000
6000
8000
10000
12000
14000
restaurants

by observing above graph, the 1st line of grapth shows the number of rest_type we have that is 87 and we plotted only top 40 types which are high count of restaurants types.

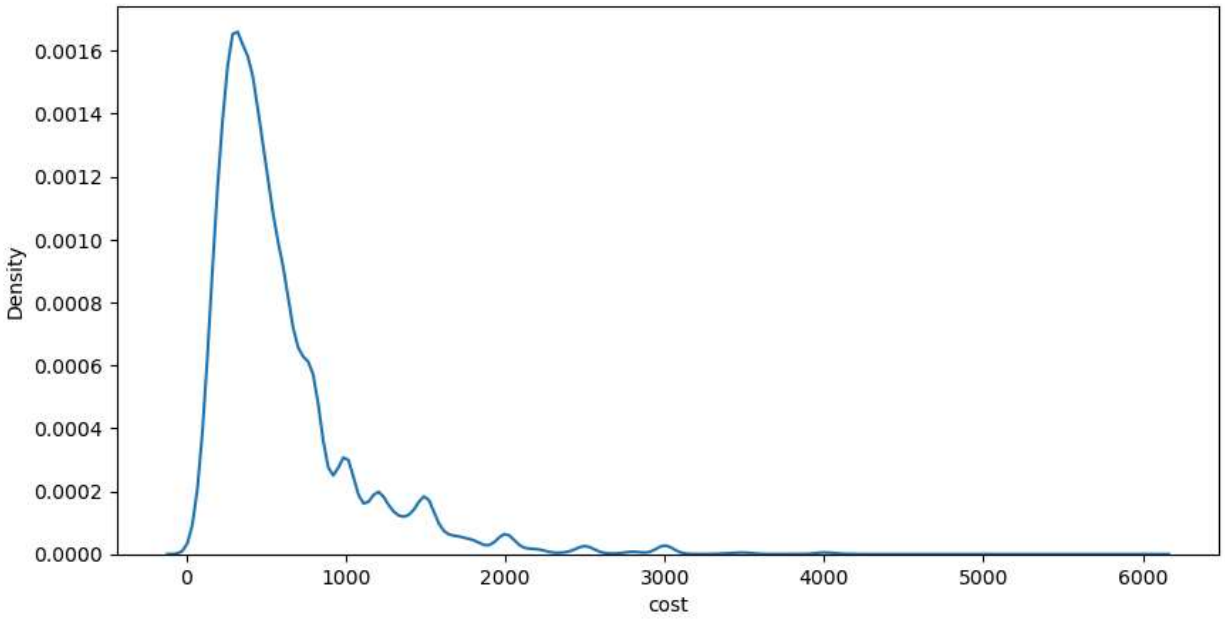
Find Cost's Of restaurants

In [51]:
```
zmt.head(1)
```

Out[51]:

	restaurants	online_order	book_table	rating	votes	location	rest_type	food_type	cost	typ
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800	Buffe

In [54]:
```
plt.figure(figsize=(10,5))
sns.kdeplot(x='cost',data=zmt)# KDE plot
plt.show()
```

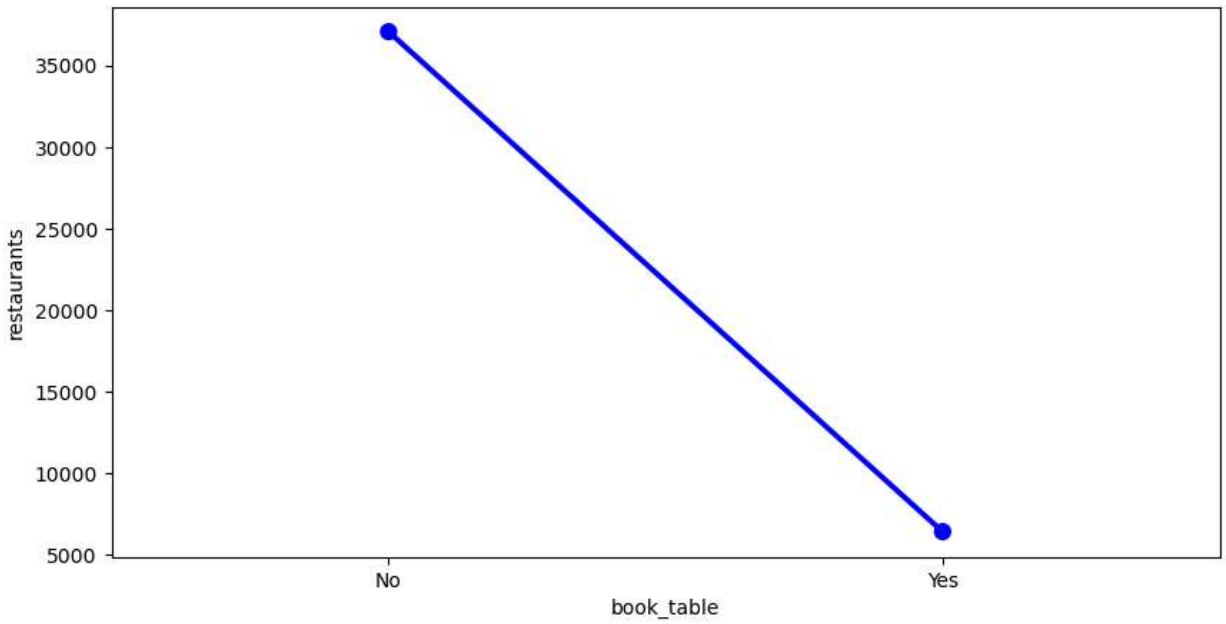


By observing above kde plot we can say that most of the restaurants have cost between 1 to 1000Rs. for food and remaining have above 1000Rs for their food.

Find count of restaurants have table booking facility.

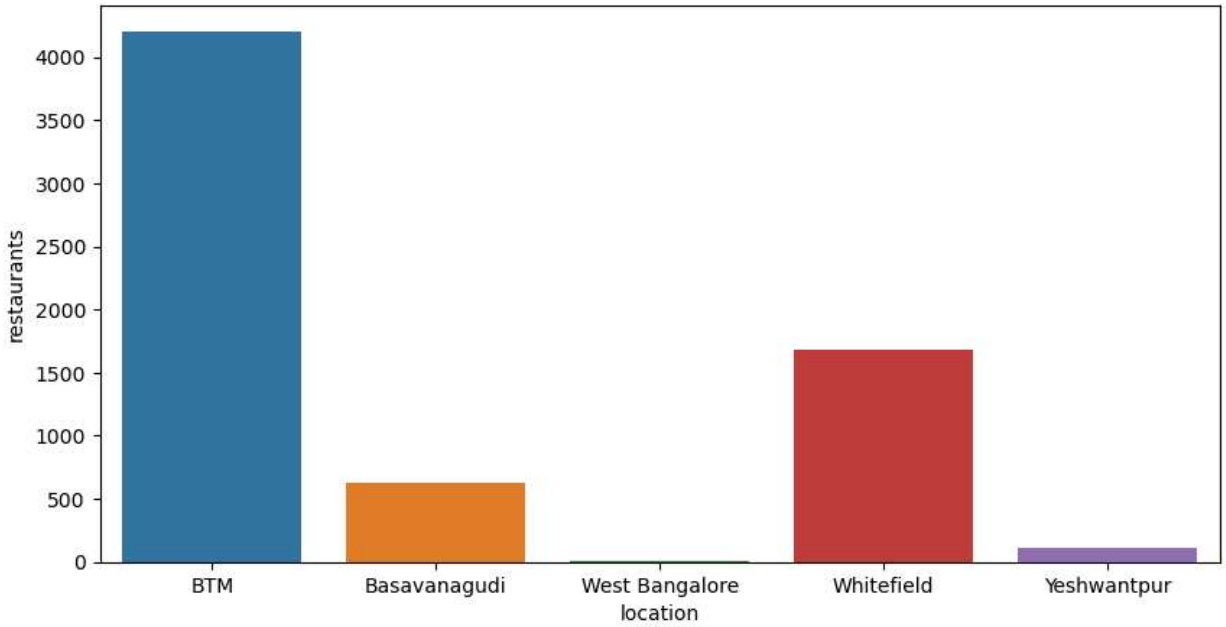
Here, we can use countplot but try to drow pointplot to see how it looks like.

In [55]:
```
plt.figure(figsize=(10,5))
table_booking= zmt.groupby('book_table')['restaurants'].count().reset_index()
sns.pointplot(x='book_table',y='restaurants',color='b',data=table_booking)
plt.show()
```



By seeing above pointplot we can say that only 7000 restaurants have table booking facility and 35000+ restaurants don't have facility of table booking.

In [56]:
```
df=pd.DataFrame(zmt.groupby('location')['restaurants'].count()).reset_index()
criteria=df['location'].isin(['BTM','Basavanagudi','West Bangalore','Whitefield','Yeshwa
plt.figure(figsize=(10,5))
sns.barplot(x='location',y='restaurants',data=df[criteria])
plt.show()
```



By seeing given plot, we can notice that how many restaurants present are at given locations.

Find most famous restaurants chains(like Franchise(Rastaurants having more than one

branch) in Bangalore

We working on a zomato dataset of bangalore location so we can asume that all locations are belongs to bangalore.

- In this problem we need to find restaurant at each location having more than 1 branch and high rating.

In [57]:
```
df1=pd.DataFrame(zmt.groupby(['location','restaurants','rating']).count()).reset_index()
# getting location,restraunts and rating and their count of restaurants
df1
```

Out[57]:

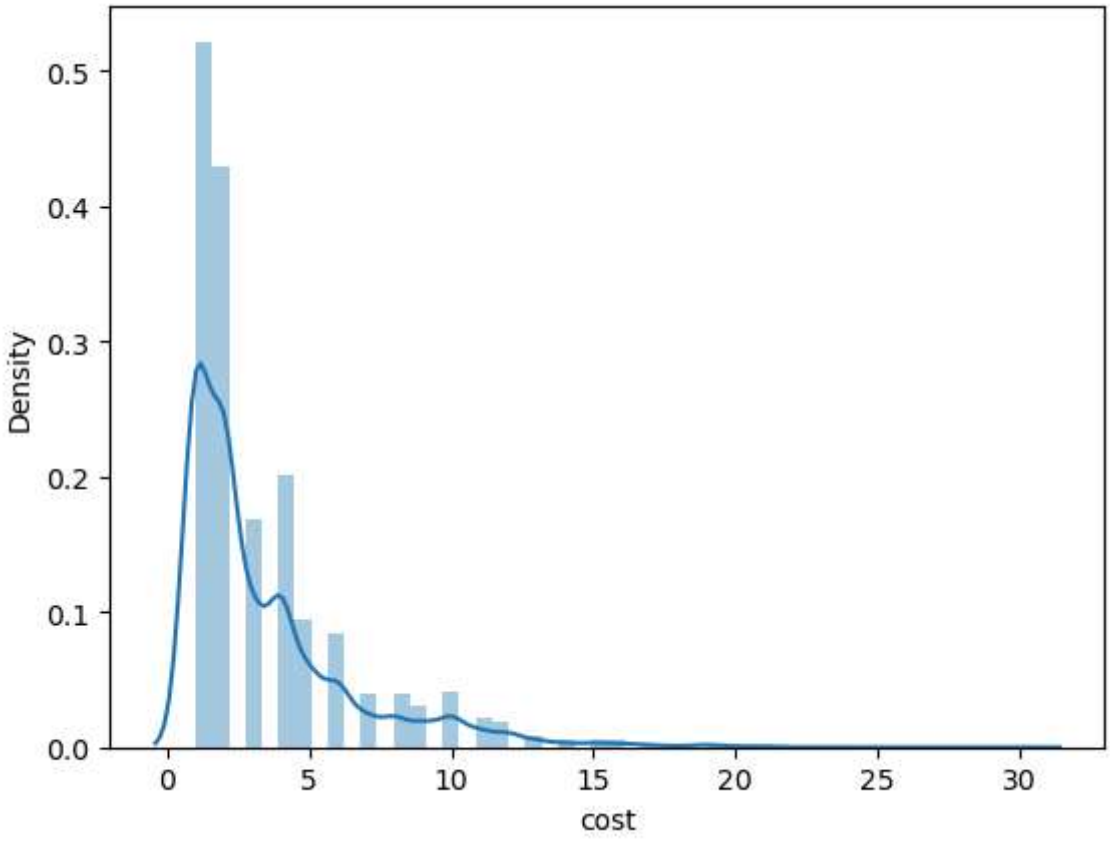
	location	restaurants	rating	online_order	book_table	votes	rest_type	food_type	cost
0	BTM	100C	3.7	3	3	3	3	3	3
1	BTM	2nd Home Restaurant	3.7	11	11	11	11	11	11
2	BTM	36th Cross Coffee Mane	3.7	1	1	1	1	1	1
3	BTM	3B's - Buddies, Bar & Barbecues	4.4	13	13	13	13	13	13
4	BTM	A2B - Adyar Ananda Bhavan	3.8	1	1	1	1	1	1
...	...	...	...	...	...	...	...	...	...
12577	Yeshwantpur	The Cupcake Story	3.5	1	1	1	1	1	1
12578	Yeshwantpur	The Duke Of Juices	3.3	1	1	1	1	1	1
12579	Yeshwantpur	Tuckinto	3.8	2	2	2	2	2	2
12580	Yeshwantpur	Udupi Garden	3.0	2	2	2	2	2	2
12581	Yeshwantpur	Zam Zam Kabab House	3.4	2	2	2	2	2	2

12582 rows × 11 columns

In [59]:
```
sns.distplot(df1['cost'])
```

```
C:\Users\DELL\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please ada
pt your code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[59]: <AxesSubplot:xlabel='cost', ylabel='Density'>



Note that all columns having some count values these are nothing but a count of that restaurant at given location.

We can see some restaurants having 1 count and some having more than 1. We want that restaurants which having more than 1 count and high rating

In [60]:
```
chains_restaurants=df1[(df1['book_table']>1)]
famous_restaurants=pd.DataFrame(chains_restaurants.groupby('location')[['restaurants','r
famous_restaurants
```

Out[60]:

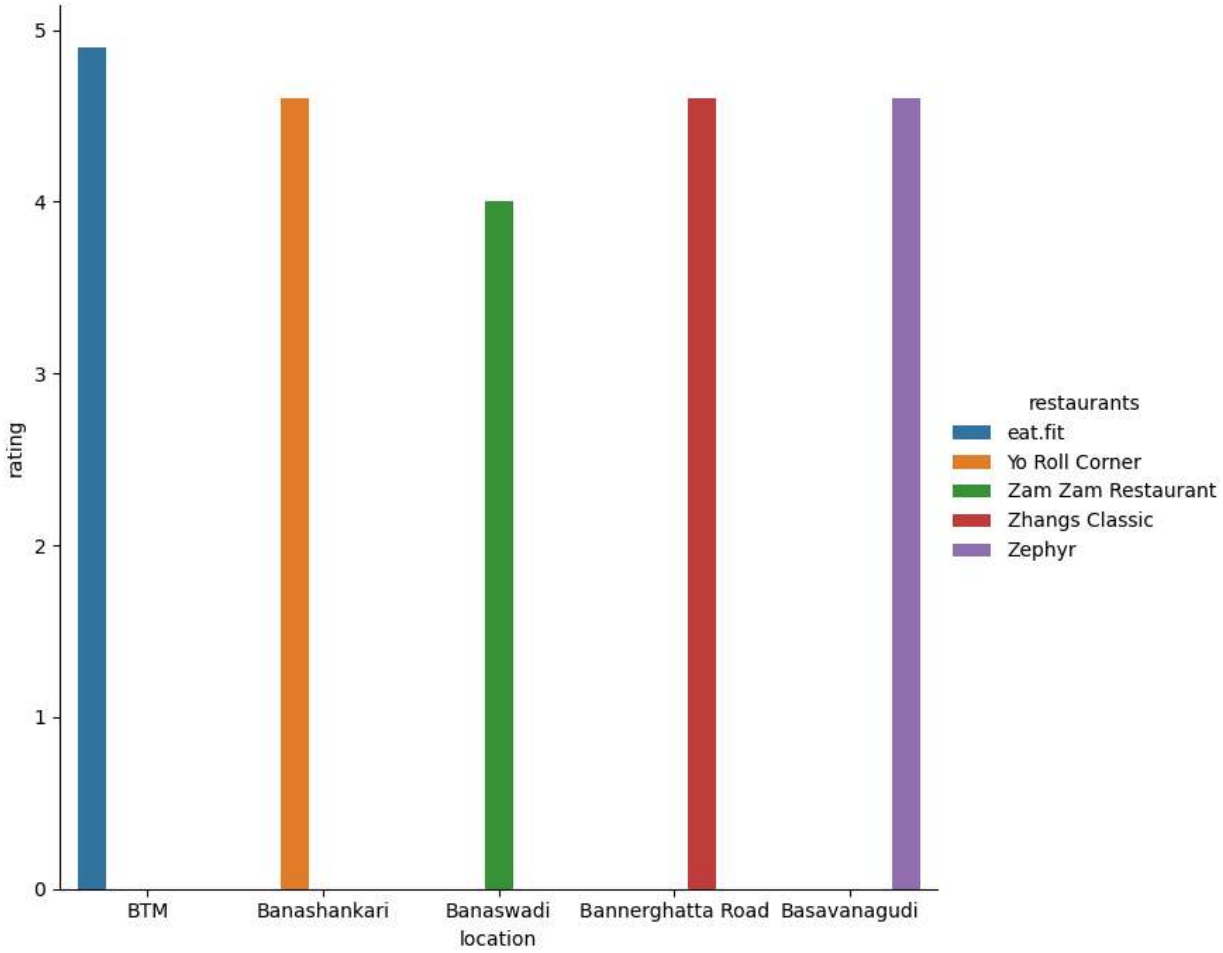
	location	restaurants	rating
0	BTM	eat.fit	4.9
1	Banashankari	Yo Roll Corner	4.6
2	Banaswadi	Zam Zam Restaurant	4.0
3	Bannerghatta Road	Zhangs Classic	4.6
4	Basavanagudi	Zephyr	4.6
...	...	...	...
84	West Bangalore	FreshMenu	3.3
85	Whitefield	nu.tree	4.9
86	Wilson Garden	Vijaya Sagar	4.0
87	Yelahanka	Prashanth Naati Corner	3.9
88	Yeshwantpur	Zam Zam Kabab House	4.2

89 rows × 3 columns

Finally, we got restaurant at each location has more that one branch and highest rating. Total 89 result we got. Let's try to plot only first 5 to get idea.

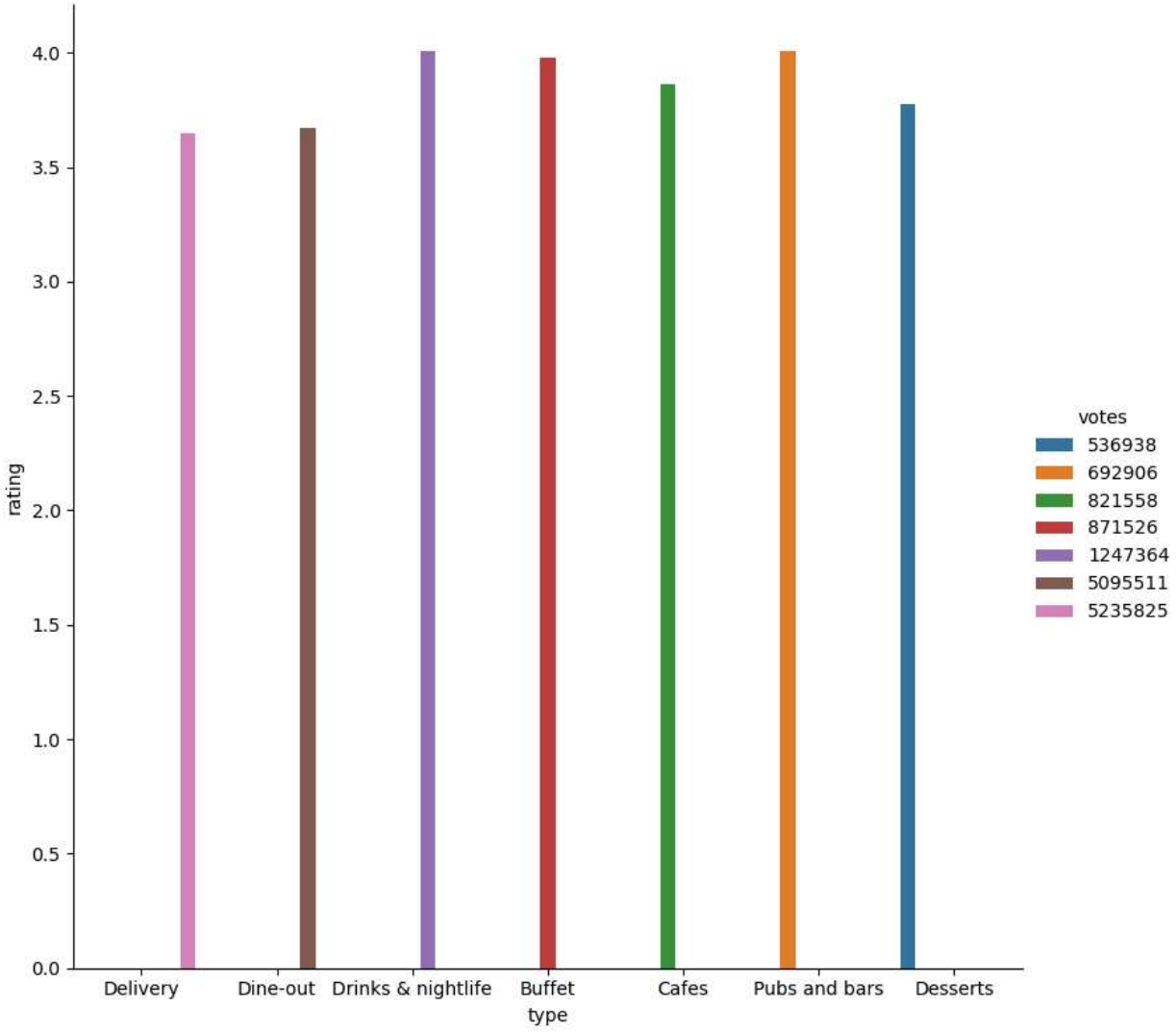
In [61]:
```
sns.catplot(x="location", y="rating", hue="restaurants",kind='bar',height=7,data=famous_
```

Out[61]:
```
<seaborn.axisgrid.FacetGrid at 0x26ab502ea00>
```



Find how many voters gives rating for each 'type' and aggregate rating of that 'type

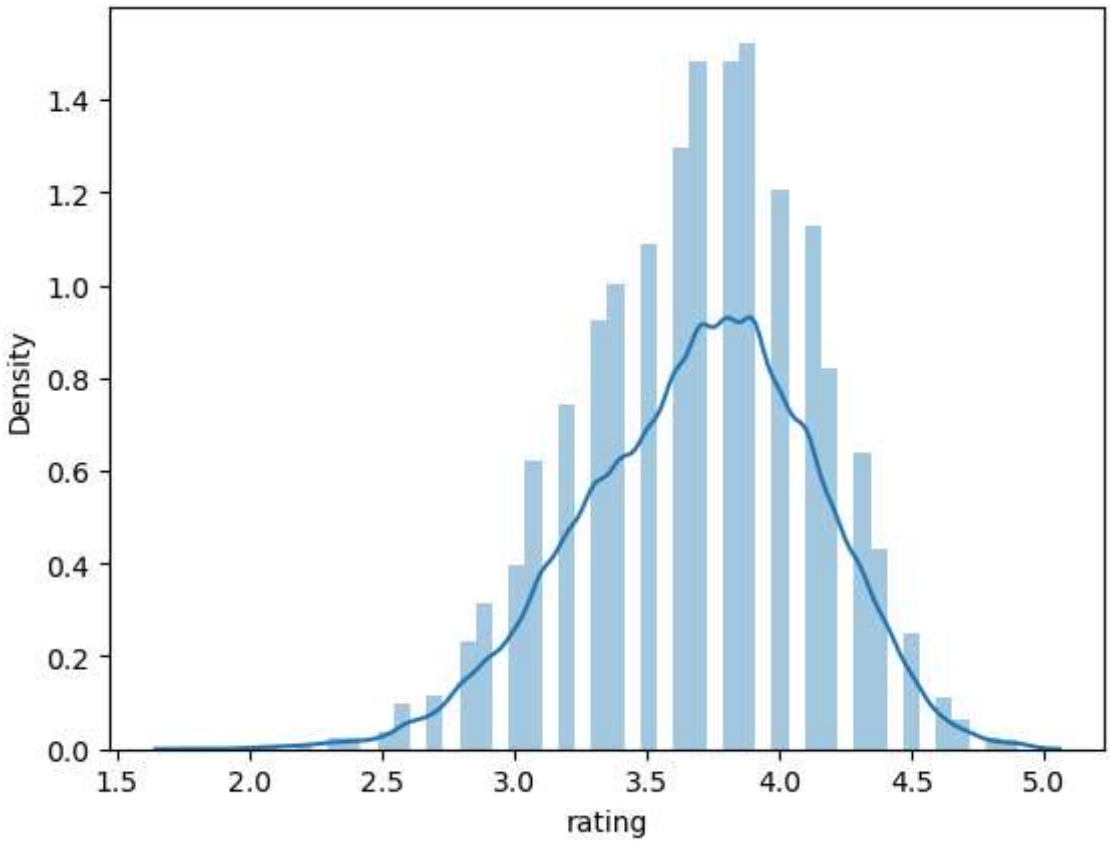
In [62]:
```
df2=zmt.groupby('type').agg({'votes':'sum','rating':'mean'}).nlargest(7,['votes']).reset
sns.catplot(x='type',y='rating',hue='votes',kind='bar',height=8,data=df2)
plt.show()
```



In [63]:
```
sns.distplot(zmt['rating'])
```

```
C:\Users\DELL\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please ada
pt your code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

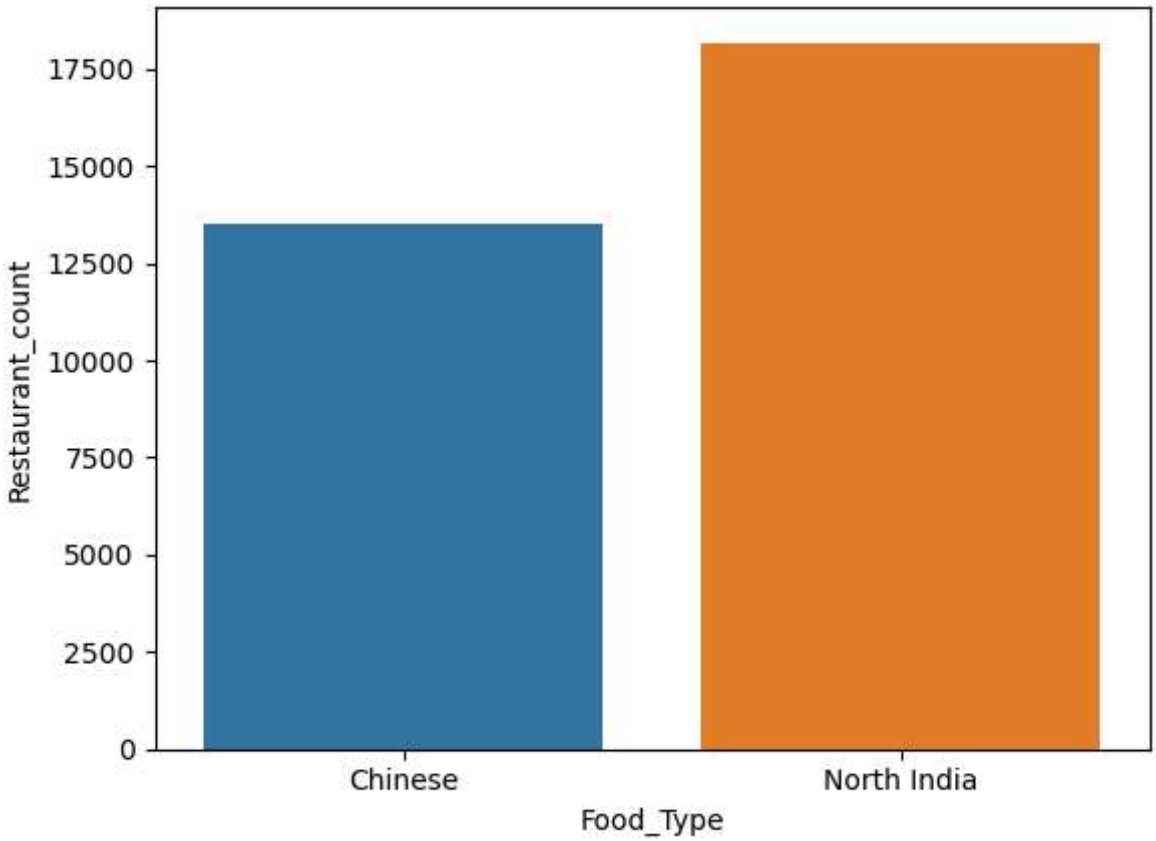
Out[63]:
```
<AxesSubplot:xlabel='rating', ylabel='Density'>
```



Find the how many Restaurants havign Chinese and North Indian food in their food type**

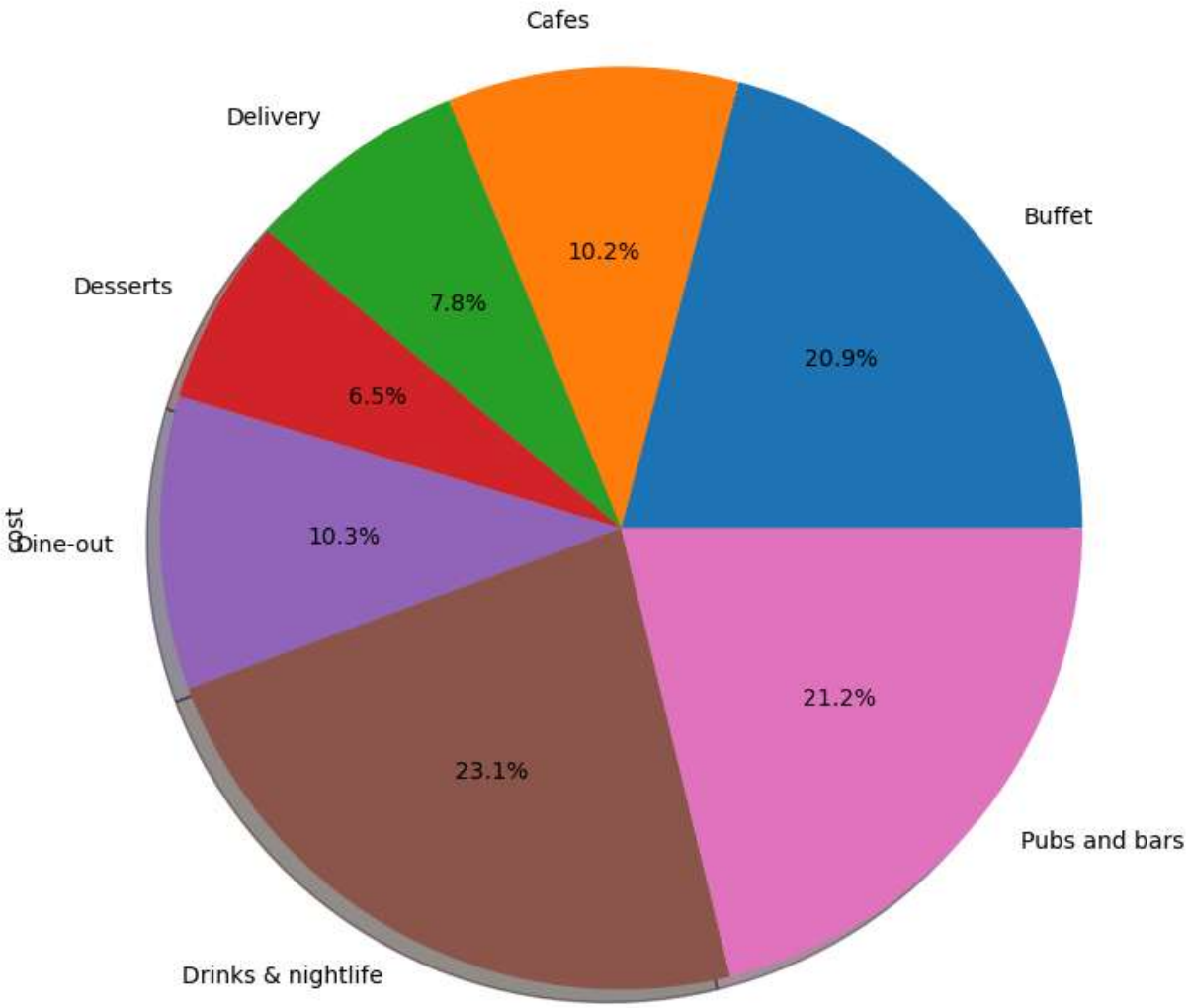
In [64]:
```
Chinese=len([i for i in zmt['food_type'] if 'Chinese' in i]) # count of restaurants who
North_India=len([i for i in zmt['food_type'] if 'North India' in i]) # count of restaurc
Restaurant_count=[Chinese,North_India] #creating list
Food_Type=['Chinese','North India']
df3 = pd.DataFrame({'Food_Type':Food_Type,'Restaurant_count':Restaurant_count})
#.set_index('Food_Type')# creating dataset of extracted data
```

In [65]:
```
sns.barplot(x='Food_Type',y='Restaurant_count',data=df3) # countplot gives us count of t
plt.show()
```



Find the most profitable type of restaurant.

In [69]:
```
df4=zmt.groupby('type').agg({'cost':'mean'})
df4.cost.plot(kind='pie',autopct='%1.1f%%',figsize=(9,9),shadow=True)
plt.show()
```



Here, by seeing the total percentage of average cost we can observe that. The restaurants having Drink and Nightlife facility are in high profit than other types.

In [ ]: