Exploratory Data Analysis of Banglore-based Restaurants

In [1]:

```
# Import library:-

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

pd.pandas.set_option('display.max_columns',None)

```

Reading Data

The dataset can be downloaded from this link:- https://www.kaggle.com/datasets/himanshupoddar/zomato-bangalore-restaurants (https://www.kaggle.com/datasets/himanshupoddar/zomato-bangalore-restaurants)

In [3]:

```
dataset = pd.read_csv("E:\zomato\zomato.csv")
dataset.head()
```

Out[3]:

ook_table	rate	votes	phone	location	rest_type	dish_liked	cuisines	appr
Yes	4.1/5	775	080 42297555\r\n+91 9743772233	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...	North Indian, Mughlai, Chinese	
No	4.1/5	787	080 41714161	Banashankari	Casual Dining	Momos, Lunch Buffet, Chocolate Nirvana, Thai G...	Chinese, North Indian, Thai	
No	3.8/5	918	+91 9663487993	Banashankari	Cafe, Casual Dining	Churros, Cannelloni, Minestrone Soup, Hot Choc...	Cafe, Mexican, Italian	
No	3.7/5	88	+91 9620009302	Banashankari	Quick Bites	Masala Dosa	South Indian, North Indian	
No	3.8/5	166	+91 8026612447\r\n+91 9901210005	Basavanagudi	Casual Dining	Panipuri, Gol Gappe	North Indian, Rajasthani	

Dropping Unnecessary Columns

In [5]:

```
dataset = dataset.drop(['url','address','phone','dish_liked','reviews_list','menu_it
dataset.head()
```

Out[5]:

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

Renaming Columns

just making it simple to use

In [9]:

```
dataset = dataset.rename(columns={'approx_cost(for two people)':'cost_for_2','listed
dataset.head()
```

Out[9]:

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

Checking Null Values

can be done by .info() or .isnull()

In [10]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51717 entries, 0 to 51716
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   name          51717 non-null  object
 1   online_order  51717 non-null  object
 2   book_table    51717 non-null  object
 3   rate          43942 non-null  object
 4   votes         51717 non-null  int64 
 5   location      51696 non-null  object
 6   rest_type     51490 non-null  object
 7   cuisines      51672 non-null  object
 8   cost_for_2    51371 non-null  object
 9   type          51717 non-null  object
 10  city          51717 non-null  object
dtypes: int64(1), object(10)
memory usage: 4.3+ MB
```

Dropping Duplicates

In [11]:

```
dataset.drop_duplicates(inplace=True)
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51609 entries, 0 to 51716
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   name          51609 non-null  object
 1   online_order  51609 non-null  object
 2   book_table    51609 non-null  object
 3   rate          43854 non-null  object
 4   votes         51609 non-null  int64
 5   location      51588 non-null  object
 6   rest_type     51382 non-null  object
 7   cuisines      51564 non-null  object
 8   cost_for_2    51265 non-null  object
 9   type          51609 non-null  object
 10  city          51609 non-null  object
dtypes: int64(1), object(10)
memory usage: 4.7+ MB
```

Checking Numerical features

In [12]:

```
li=['rate','votes','cost_for_2']
for i in li:
    print(dataset[i].unique())
```

```
['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' nan
 '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']
[ 775  787  918 ... 4957 2382  843]
['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' '50' '190' '1,700' nan '1,400'
 '180' '1,350' '2,200' '2,000' '1,800' '1,900' '330' '2,500' '2,100'
 '3,000' '2,800' '3,400' '40' '1,250' '3,500' '4,000' '2,400' '2,600'
 '120' '1,450' '469' '70' '3,200' '60' '560' '240' '360' '6,000' '1,050'
 '2,300' '4,100' '5,000' '3,700' '1,650' '2,700' '4,500' '140']
```

Cleaning Rate Feature

It has /and new in the data so making it as numerical value.

In [25]:

```
def rate(value):
    if(value=='NEW' or value=='-'):
        return np.nan
    else:
        value=str(value).split('/')
        value=value[0]
        return float(value)
dataset['rate']=dataset['rate'].apply(rate)

dataset['rate'].unique()
```

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, nan, 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])
```

Cleaning Cost Feature

It has , in the data

In [28]:

```
def cost(value):
    value=str(value)
    if ',' in value:
        value1=value.replace(',','')
        return float(value1)
    else:
        return float(value)

dataset ['cost_for_2'] = dataset['cost_for_2'].apply(cost)


dataset ['cost_for_2'].unique()
```

Out[28]:

```
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.,   50.,  190., 1700.,   nan, 1400.,  180., 1350.,
       2200., 2000., 1800., 1900.,  330., 2500., 2100., 3000., 2800.,
       3400.,   40., 1250., 3500., 4000., 2400., 2600.,  120., 1450.,
        469.,   70., 3200.,   60.,  560.,  240.,  360., 6000., 1050.,
       2300., 4100., 5000., 3700., 1650., 2700., 4500.,  140.])
```

Perfect Numerical Features with nan

In [30]:

```
numerical_features=[feature for feature in dataset.columns if dataset[feature].dtype
numerical_features
```

Out[30]:

```
['rate', 'votes', 'cost_for_2']
```

Filling Null Values with Median.

In [32]:

```
for feature in numerical_features:
    median=dataset[feature].median()
    dataset[feature].fillna(median,inplace=True)

dataset[numerical_features]
```

Out[32]:

	rate	votes	cost_for_2
0	4.1	775	800.0
1	4.1	787	800.0
2	3.8	918	800.0
3	3.7	88	300.0
4	3.8	166	600.0
...	...	...	...
51712	3.6	27	1500.0
51713	3.7	0	600.0
51714	3.7	0	2000.0
51715	4.3	236	2500.0
51716	3.4	13	1500.0

51609 rows × 3 columns

Just confirming that it doesn't contain null values

In [33]:

```
dataset[numerical_features].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51609 entries, 0 to 51716
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   rate        51609 non-null  float64
 1   votes       51609 non-null  int64  
 2   cost_for_2  51609 non-null  float64
dtypes: float64(2), int64(1)
memory usage: 1.6 MB
```

Categorical Features

In [36]:

```
cat_features=[feature for feature in dataset.columns if dataset[feature].dtype=='O']
cat_features
```

Out[36]:

```
['name',
 'online_order',
 'book_table',
 'location',
 'rest_type',
 'cuisines',
 'type',
 'city']
```

Checking Categorical features Unique Values

In [40]:

```
for feature in cat_features:
    if feature !='name':
        print(dataset[feature].value_counts())
dataset[cat_features]
```

```
Name: location, Length: 93, dtype: int64
Quick Bites                      19096
Casual Dining                    10309
Cafe                              3727
Delivery                          2600
Dessert Parlor                    2260
                                 ...  
Dessert Parlor, Kiosk                2
Food Court, Beverage Shop            2
Dessert Parlor, Food Court           2
Sweet Shop, Dessert Parlor           1
Quick Bites, Kiosk                   1
Name: rest_type, Length: 93, dtype: int64
North Indian                                    2907
North Indian, Chinese                           2381
South Indian                                    1826
Biryani                                          915
Bakery, Desserts                                 910
                                                ...  
European, Asian, North Indian                      1
```

Cleaning Cat Features

Here online_order , book_table,city have less unique values and they are perfect to visualize whereas , other features need to cleaned. so, if any unique value less than 0.5% or 1% weigtage then i am considering it as rare_variable or simply as others.

In [46]:

```
cat_f=['cuisines','location','rest_type']

for feature in cat_f:
    temp=dataset[feature].value_counts()/len(dataset)
    index=temp[temp>0.005].index
    dataset[feature]=np.where(dataset[feature].isin(index),dataset[feature],'others'

    for feature in cat_f:
        print(dataset[feature].value_counts())
```

```
others                                 33860
North Indian                            2907
North Indian, Chinese                   2381
South Indian                            1826
Biryani                                  915
Bakery, Desserts                         910
Fast Food                                801
Desserts                                 760
Cafe                                     755
South Indian, North Indian, Chinese      726
Bakery                                   651
Chinese                                  554
Ice Cream, Desserts                      416
Chinese, North Indian                    415
Mithai, Street Food                      372
Desserts, Ice Cream                      353
North Indian, Chinese, Biryani           351
South Indian, North Indian               343
North Indian, South Indian               342
```

When it comes to a place category we have two features (location,city). Location is more specific than city feature so, i am droppinh city.

In [48]:

```
dataset.drop(['city'],axis=1,inplace=True)
dataset.head()
```

Out[48]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	cost_for
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	others	80
1	Spice Elephant	Yes	No	4.1	787	Banashankari	Casual Dining	others	80
2	San Churro Cafe	Yes	No	3.8	918	Banashankari	others	others	80
3	Addhuri Udupi Bhojana	No	No	3.7	88	Banashankari	Quick Bites	South Indian, North Indian	30
4	Grand Village	No	No	3.8	166	Basavanagudi	Casual Dining	others	60

Data is clean and perfect for visualizing

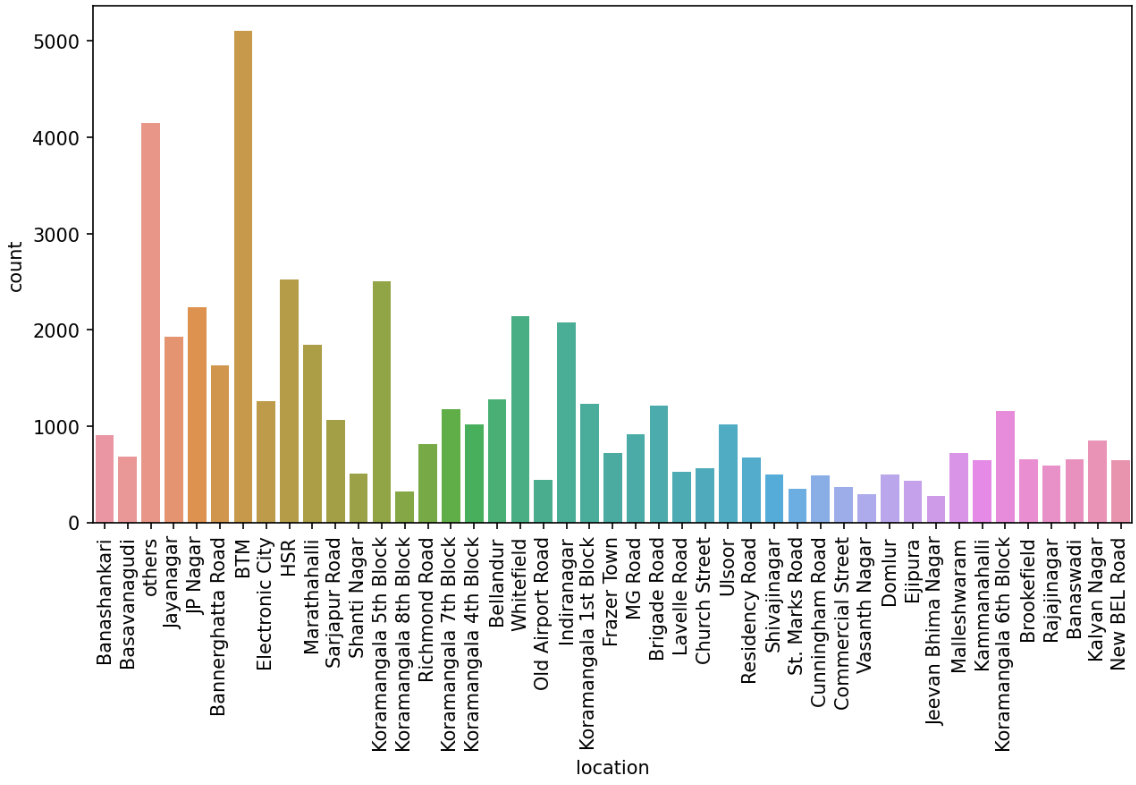
Visualizing Data

Restaurants according to location

From this graph , one can analyze which place is good for opening a restaurant.

In [49]:

```
plt.figure(figsize=(10,5),dpi=150)
sns.countplot(x='location', data = dataset)
plt.xticks(rotation=90)
plt.savefig('count(rest)-location.jpg',bbox_inches='tight')
```

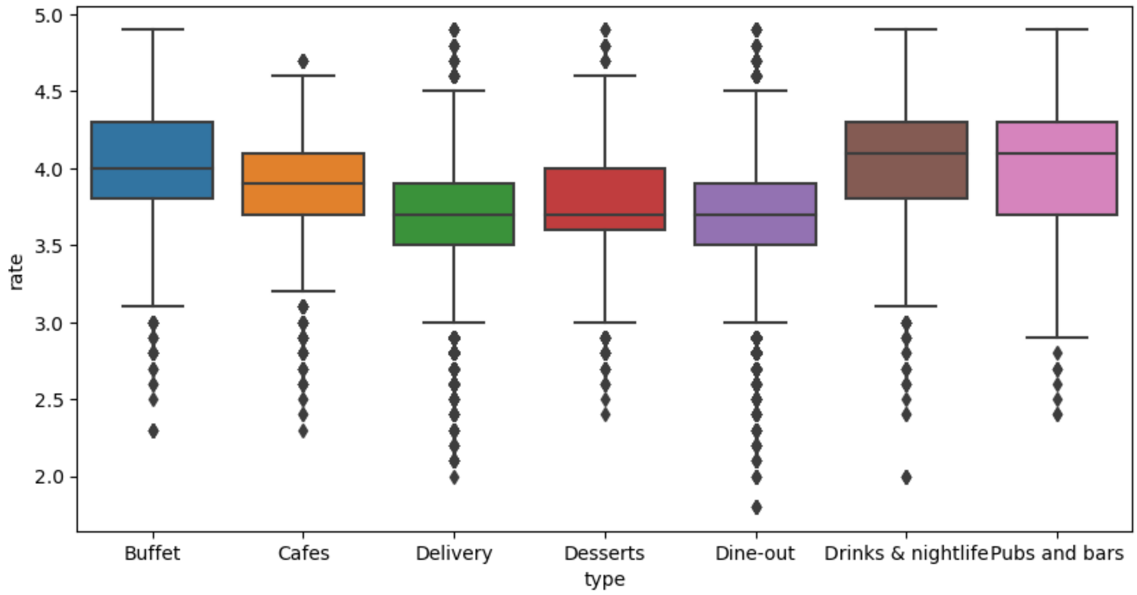


Rating according to type of restaurant

In [50]:

```
plt.figure(figsize=(10,5),dpi = 100)
sns.boxplot(x='type',y='rate',data=dataset)

plt.savefig('rate-type.jpg')
```

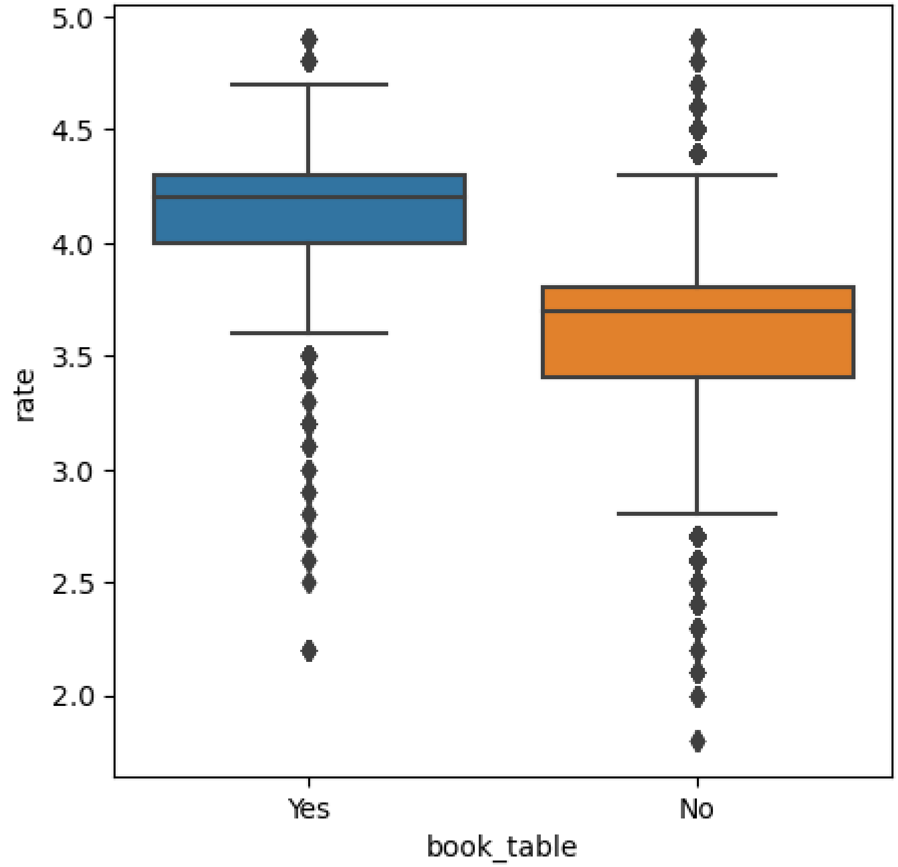


Booking table Option

From this graph one can know whether to give option for booking or not

In [51]:

```
plt.figure(figsize=(5,5),dpi=100)
sns.boxplot(x='book_table',y='rate',data=dataset)
plt.savefig('tablebooking.jpg')
```

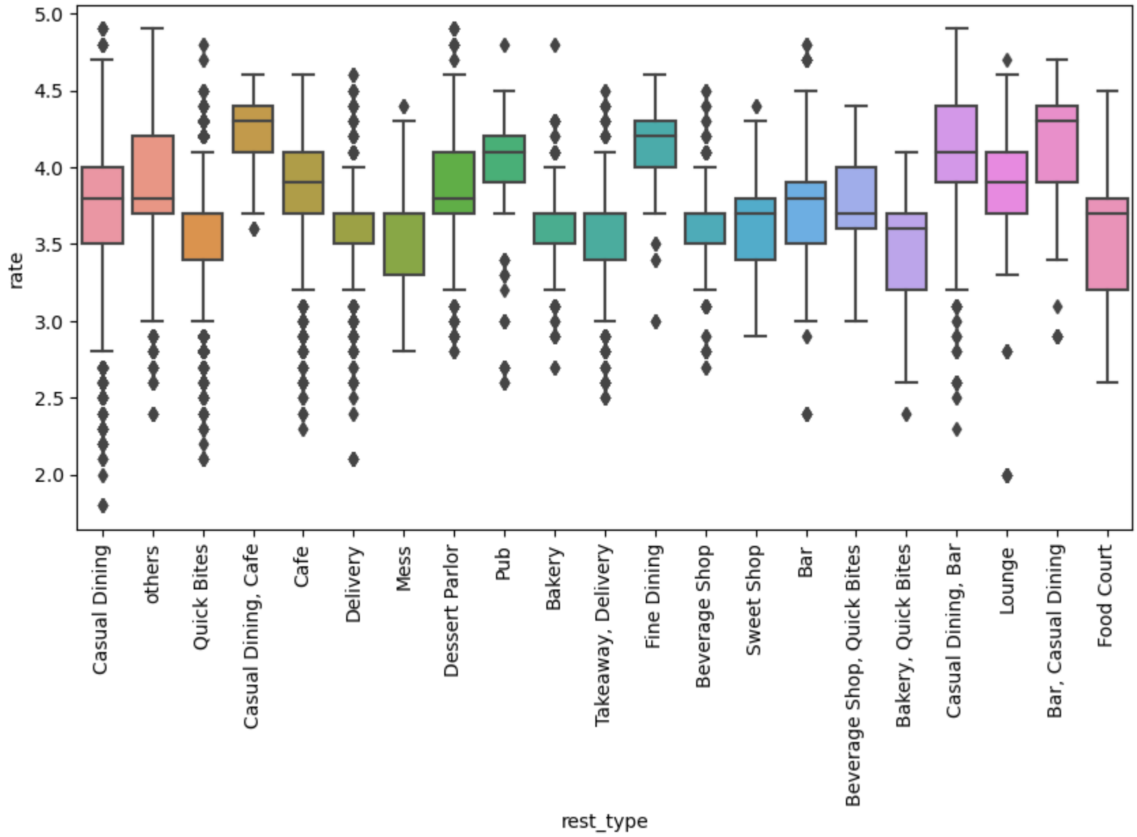


Rating according to a Restaurant type

From this graph, One can get to know in which type people are more satisfied and in which type people are less satisfied and interested.

In [53]:

```
plt.figure(figsize=(10,5),dpi=100)
sns.boxplot(x='rest_type',y ='rate',data=dataset)
plt.xticks(rotation=90)
plt.savefig('rate-rest_type.jpg',bbox_inches='tight')
```

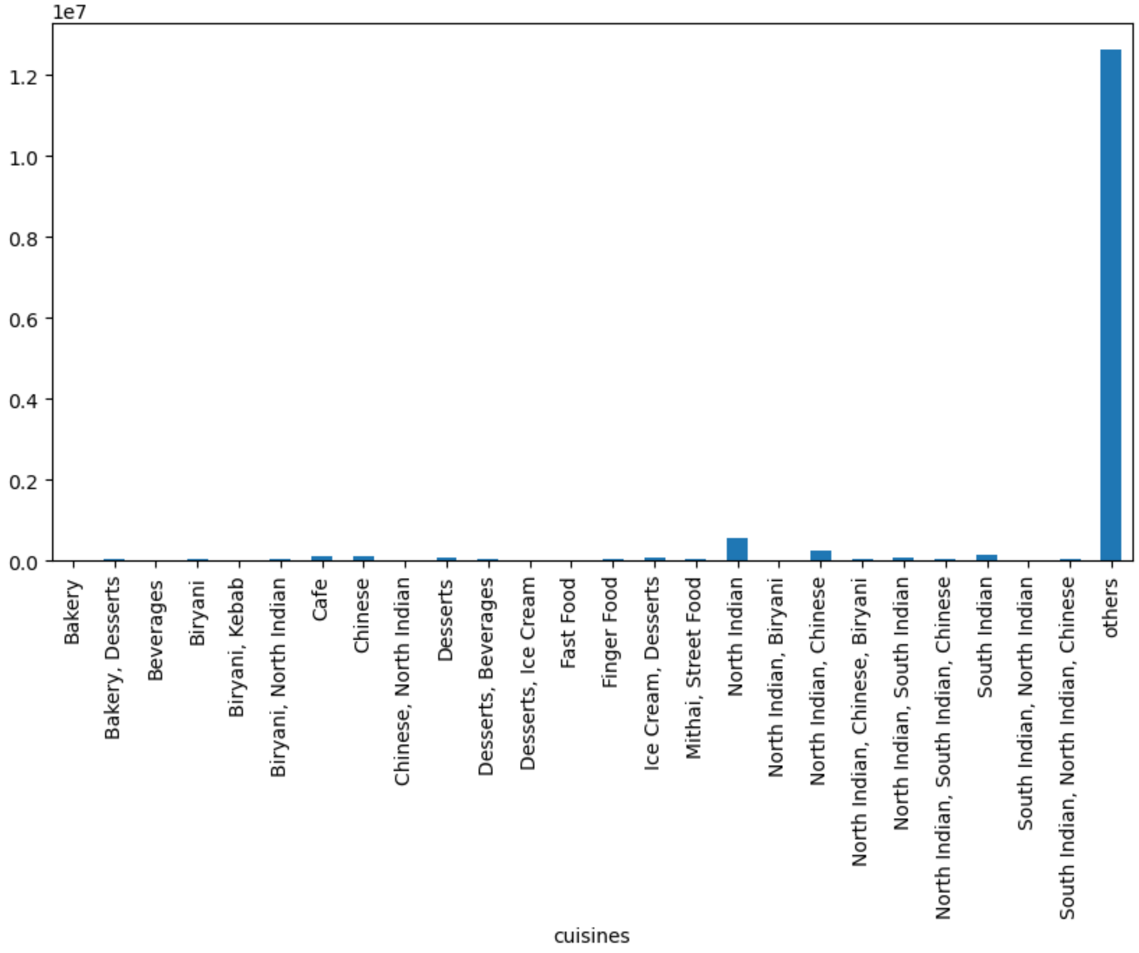


Votes according to thd cusines

here others option is dominating and not giving perfect answer. so without others need to plotted.

In [54]:

```
plt.figure(figsize=(10,5),dpi=100)
dataset.groupby(['cuisines'])['votes'].sum().plot.bar()
plt.show()
```



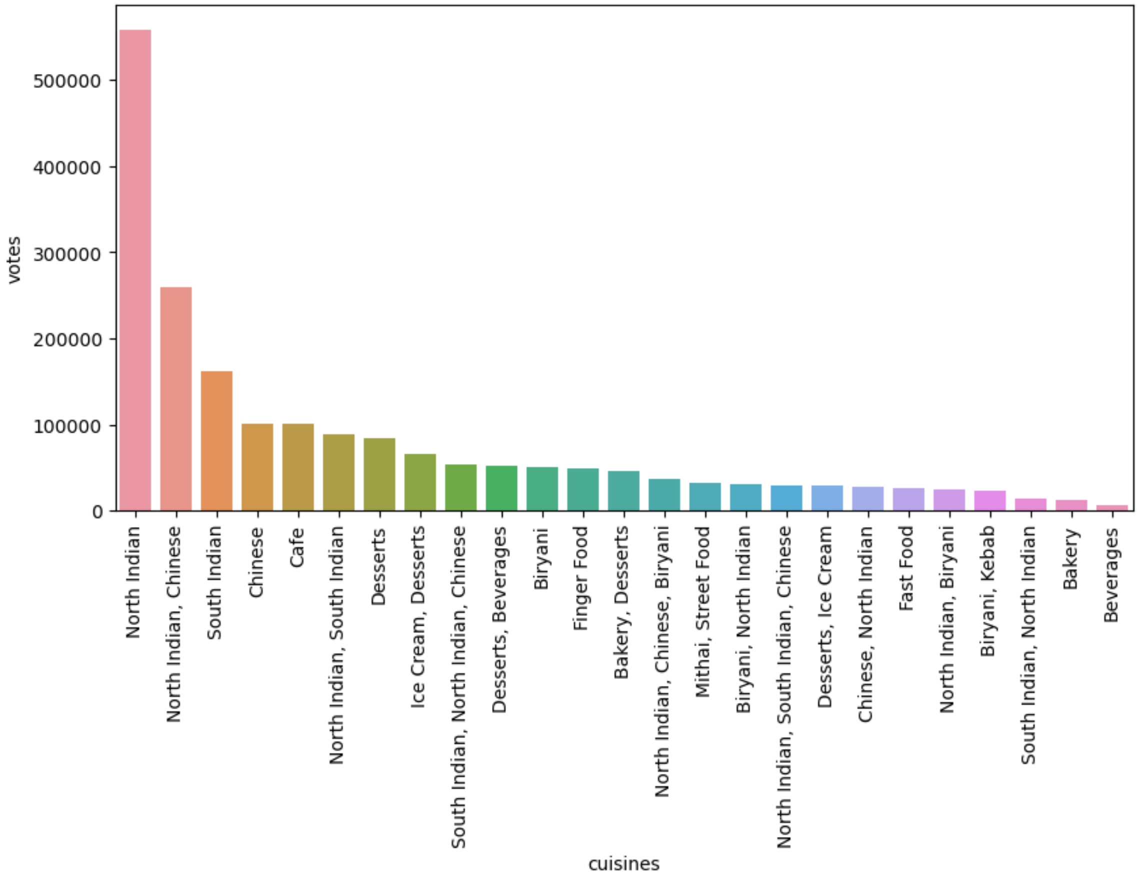
In [55]:

```
df=dataset.groupby(['cuisines'])['votes'].sum()
df=df.to_frame()
df=df.sort_values('votes',ascending=False)
df.drop('others',axis=0,inplace = True)
```

In [56]:

```
plt.figure(figsize=(10,5),dpi=100)
sns.barplot(df.index,df['votes'])
plt.xticks(rotation=90)
plt.savefig('votes-cuisines.jpg',bbox_inches='tight')
```

```
C:\Users\meanu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variables as keyword args: x, y. From versio
n 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misint
erpretation.
  warnings.warn(
```



Online order availabilty at diff Locations

From this graph , one can know at which locations there is ordering facility more and in which location it is less and can conclude to put the facility or not

In [57]:

```
df2=dataset.groupby(['location','online_order'])['name'].count()
df2=df2.to_frame()
df3=df2.pivot_table(index='location',columns='online_order')
df3
```

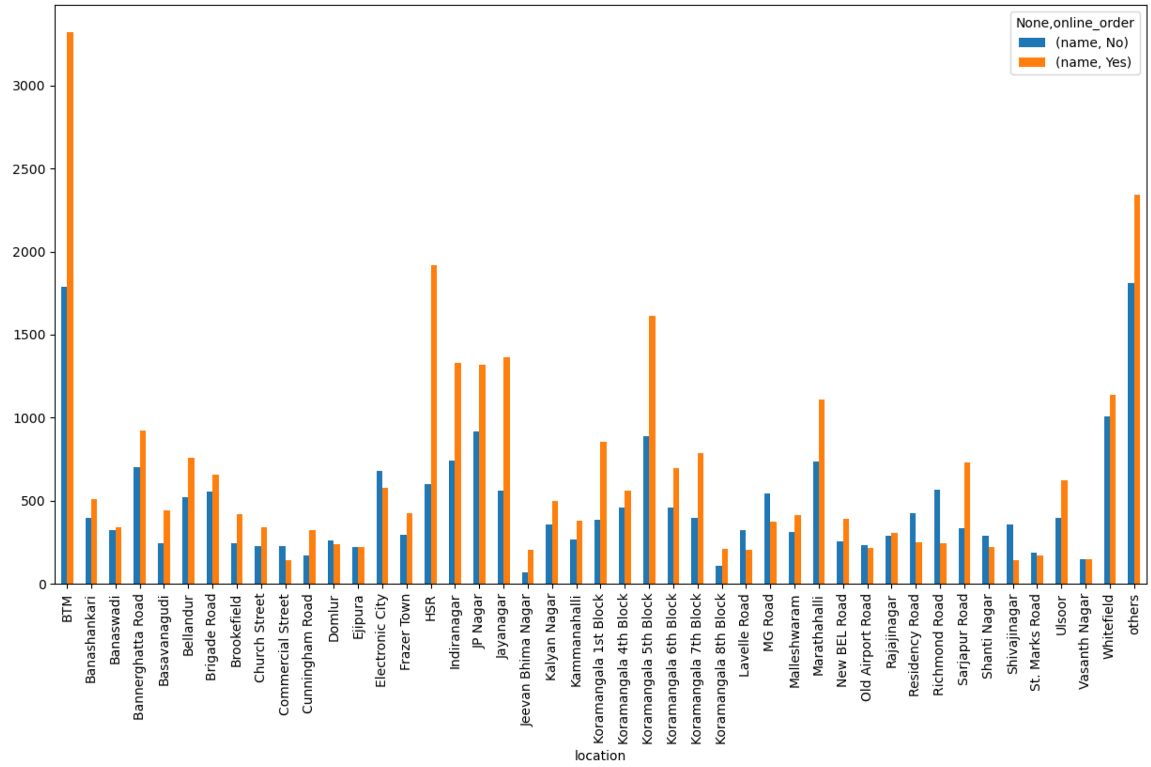
Out[57]:

	name	
online_order	No	Yes
location		
BTM	1789	3320
Banashankari	397	507
Banaswadi	321	338
Bannerghatta Road	704	924
Basavanagudi	243	441
Bellandur	523	760
Brigade Road	552	658
Brookefield	241	417
Church Street	226	340
Commercial Street	228	142
Cunningham Road	168	322
Domlur	261	235
Ejipura	219	219
Electronic City	681	575
Frazer Town	293	427
HSR	602	1919
Indiranagar	743	1332
JP Nagar	917	1317
Jayanagar	562	1364
Jeevan Bhima Nagar	68	204
Kalyan Nagar	355	498
Kammanahalli	267	380
Koramangala 1st Block	385	852
Koramangala 4th Block	459	558
Koramangala 5th Block	889	1613
Koramangala 6th Block	457	697
Koramangala 7th Block	394	785
Koramangala 8th Block	108	212
Lavelle Road	321	203
MG Road	544	373
Malleshwaram	310	412
Marathahalli	734	1109
New BEL Road	257	392
Old Airport Road	230	216
Rajajinagar	286	305

	name	
online_order	**No**	**Yes**
location		
Residency Road	425	247
Richmond Road	565	246
Sarjapur Road	335	728
Shanti Nagar	289	219
Shivajinagar	354	144
St. Marks Road	185	167
Ulsoor	395	622
Vasanth Nagar	148	147
Whitefield	1005	1135
others	1813	2340

In [58]:

```
df3.plot.bar(figsize=(15,8))
plt.savefig('Online_order.jpg',bbox_inches='tight')
```



No of types of restaurants according to location

In [59]:

```
df4=dataset.groupby(['location','type'])['name'].count()
df4=df4.to_frame()
df5=df4.pivot_table(index='location',columns='type',fill_value=0)
df5
```

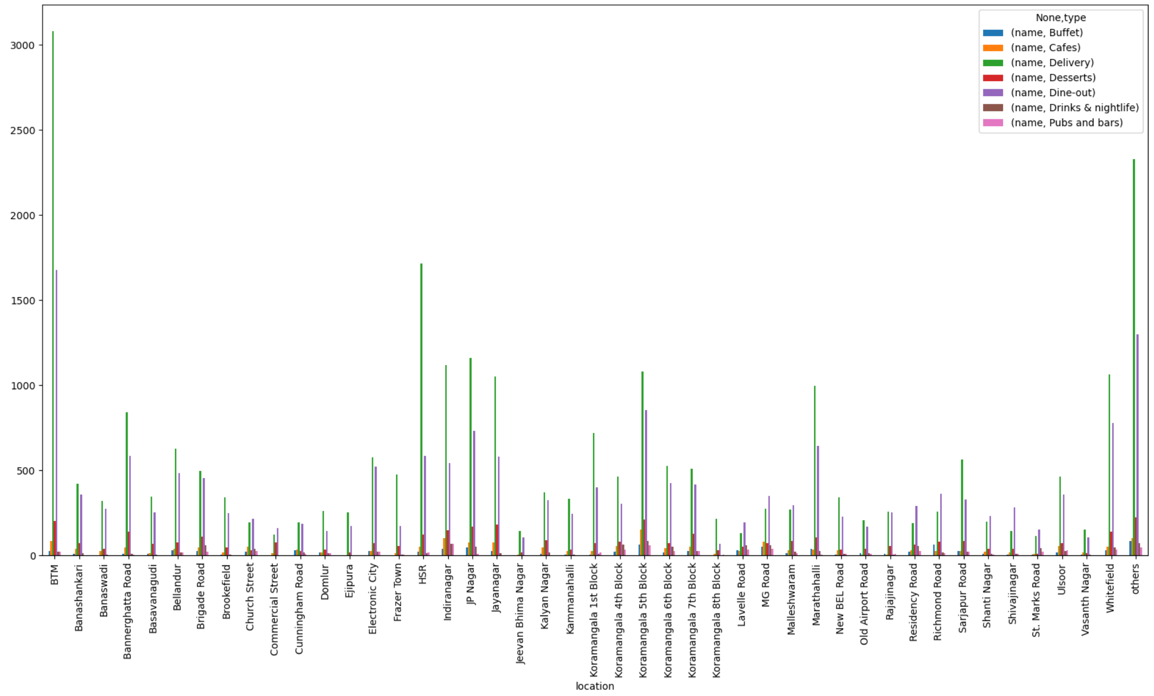
Out[59]:

	name						
type	Buffet	Cafes	Delivery	Desserts	Dine-out	Drinks & nightlife	Pubs and bars
location							
BTM	25	83	3082	202	1676	22	19
Banashankari	7	36	419	71	357	14	0
Banaswadi	0	24	320	37	271	6	1
Bannerghatta Road	9	46	842	137	583	9	2
Basavanagudi	7	11	344	66	251	5	0
Bellandur	28	36	624	77	485	17	16
Brigade Road	25	46	497	108	455	57	22
Brookefield	6	17	340	45	246	4	0
Church Street	19	51	193	29	215	36	23
Commercial Street	0	13	121	77	159	0	0
Cunningham Road	29	34	194	26	184	16	7
Domlur	15	17	261	35	144	12	12
Ejipura	0	0	250	16	172	0	0
Electronic City	23	24	575	71	521	21	21
Frazer Town	1	11	474	56	174	2	2
HSR	19	49	1714	123	584	14	18
Indiranagar	38	100	1116	146	540	67	68
JP Nagar	45	76	1159	166	730	51	7
Jayanagar	27	77	1049	182	579	12	0
Jeevan Bhima Nagar	0	6	141	18	107	0	0
Kalyan Nagar	9	45	370	88	323	18	0
Kammanahalli	2	27	332	35	245	6	0
Koramangala 1st Block	3	26	717	70	398	7	16
Koramangala 4th Block	21	53	464	81	302	62	34
Koramangala 5th Block	65	151	1082	210	852	84	58
Koramangala 6th Block	18	43	526	70	423	51	23
Koramangala 7th Block	25	52	508	127	417	25	25
Koramangala 8th Block	0	10	213	28	67	0	2
Lavelle Road	30	27	129	50	193	60	35
MG Road	51	78	271	73	349	57	38
Malleshwaram	11	31	269	85	292	20	14

	name						
type	**Buffet**	**Cafes**	**Delivery**	**Desserts**	**Dine-out**	**Drinks & nightlife**	**Pubs and bars**
location							
Marathahalli	37	32	995	107	643	25	4
New BEL Road	4	29	341	34	225	8	8
Old Airport Road	12	5	204	37	167	12	9
Rajajinagar	10	4	258	55	251	3	10
Residency Road	20	31	187	63	290	55	26
Richmond Road	63	25	257	78	360	16	12
Sarjapur Road	25	23	565	83	326	19	22
Shanti Nagar	9	22	198	39	229	9	2
Shivajinagar	6	17	143	37	280	7	8
St. Marks Road	5	10	115	10	150	40	22
Ulsoor	16	56	462	71	359	23	30
Vasanth Nagar	4	16	152	12	106	5	0
Whitefield	28	51	1064	139	778	47	33
others	83	102	2330	223	1300	70	45

In [60]:

```
df5.plot.bar(figsize=(20,10))
plt.savefig('type-location.jpg',bbox_inches='tight')
```



In [61]:

```
dataset.head()
```

Out[61]:

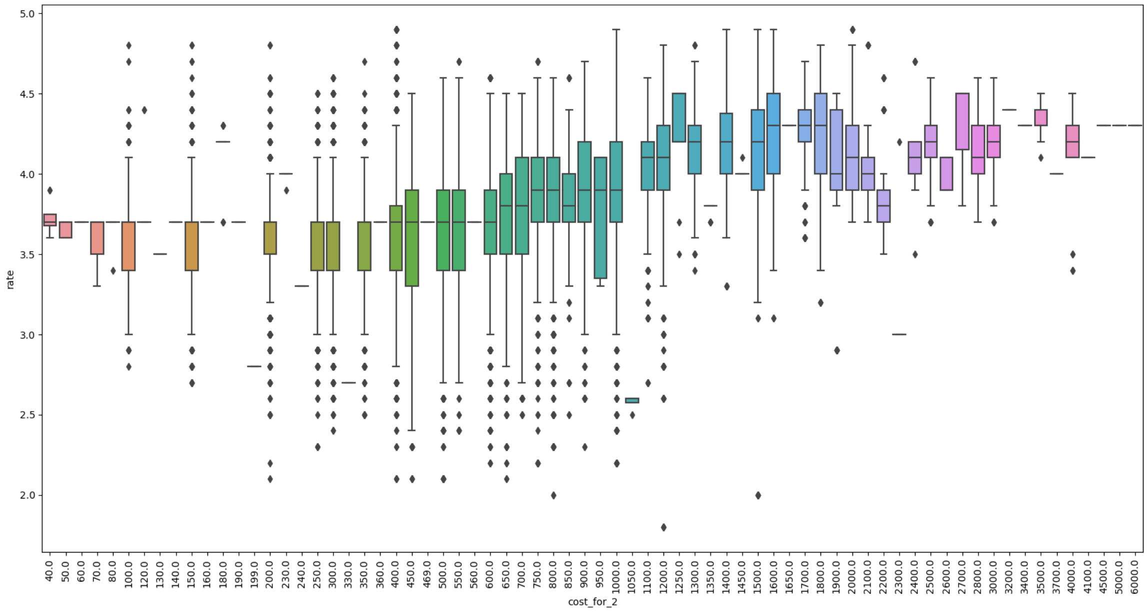
	name	online_order	book_table	rate	votes	location	rest_type	cuisines	cost_for
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	others	80
1	Spice Elephant	Yes	No	4.1	787	Banashankari	Casual Dining	others	80
2	San Churro Cafe	Yes	No	3.8	918	Banashankari	others	others	80
3	Addhuri Udupi Bhojana	No	No	3.7	88	Banashankari	Quick Bites	South Indian, North Indian	30
4	Grand Village	No	No	3.8	166	Basavanagudi	Casual Dining	others	60

Cost Vs Rating

From this graph , one can know at which price range people rated more and more satisfied

In [62]:

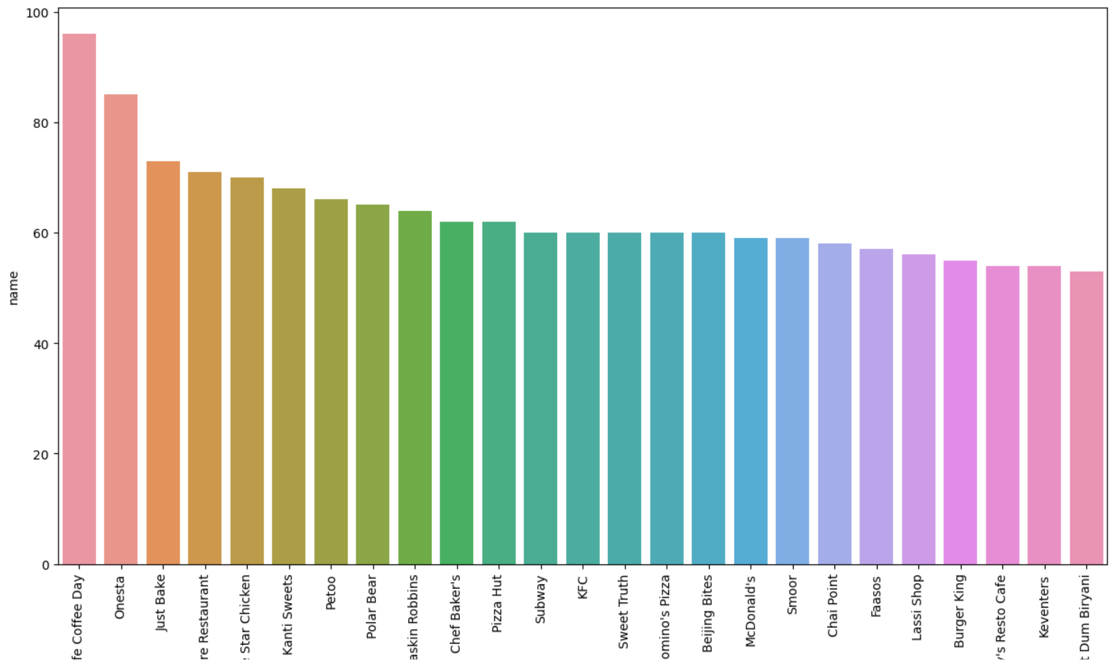
```
plt.figure(figsize=(20,10),dpi=100)
sns.boxplot(x='cost_for_2',y='rate',data=dataset)
plt.xticks(rotation=90)
plt.savefig('rate-cost.jpg')
```



Leading Franchises according to count

In [63]:

```
plt.figure(figsize=(15,8),dpi=100)
dff=dataset['name'].value_counts()[:25]
sns.barplot(x=dff.index,y=dff)
plt.xticks(rotation=90)
plt.savefig('Branches-count.jpg',bbox_inches='tight')
```



In [ ]:

```
 
```