Zomato Bengaluru-

Summary-

In [ ]:

```
Summary of Zomato EDA using Python for Business Owners

Zomato, one of the leading online food delivery platforms, provides a vast amount of data that
can be analyzed to gain valuable insights for business owners. Exploratory Data Analysis (EDA)
using Python is an effective way to extract meaningful information from this data.

In this analysis, Python libraries such as Pandas, NumPy, and Matplotlib are utilized to explore
and visualize the Zomato dataset. The dataset contains information about restaurants, including
their ratings, cuisines, locations, and user reviews.

The first step in the EDA process is data cleaning and preprocessing. This involves handling
missing values, removing duplicates, and standardizing data formats. Once the data is cleaned,
statistical summaries and visualizations are generated to understand the characteristics of the
dataset.Business owners can benefit from this analysis by uncovering key insights that can drive
decision-making process. For instance, they can identify the most popular cuisines among custome
, allowing them to optimize their menu offerings. By analyzing the distribution of ratings, owne
can gain an understanding of customer satisfaction levels and make improvements accordingly.

Furthermore, EDA can provide geographical insights by plotting the locations of restaurants on
maps. This helps business owners identify areas with high restaurant density or areas where thei
business might be lacking representation. By analyzing user reviews, owners can also gain insigh
into customer preferences and sentiments towards different aspects of their dining experience.

```

In [ ]:

```
In addition to understanding customer preferences, EDA can help business owners optimize their
pricing strategies. Analyzing the relationship between ratings and average cost can provide
valuable insights into the pricing sweet spot that maximizes customer satisfaction while
maintaining profitability.

Overall, Zomato EDA using Python empowers business owners with actionable insights derived from
a rich dataset. By leveraging this analysis, owners can make informed decisions to enhance their
offerings, improve customer satisfaction, and optimize their business strategies.
```

EDA -

(Exploratory Data Analysis)

In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```python
data = pd.read_csv("zomato.csv")
```

Introduction :-

We will understand the dataset research the various aspects of it then will go to the EDA part.

In [3]:

```
data.head()
```

Out[3]:

	url	address	name	online_order	book_table	rate	votes
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775
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
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5	918
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7/5	88
4	https://www.zomato.com/bangalore/grand-village...	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.8/5	166

In [4]:

```
data.shape # we have 51717 rows and 17 columns in the dataset
```

Out[4]:

```
(51717, 17)
```

In [5]:

```
data.columns
```

Out[5]:

```
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')
```

In [6]:

```
data.isnull().sum() # number of null values inside each columns
```

Out[6]:

```
url                                0
address                            0
name                               0
online_order                       0
book_table                         0
rate                            7775
votes                              0
phone                           1208
location                          21
rest_type                        227
dish_liked                     28078
cuisines                          45
approx_cost(for two people)      346
reviews_list                       0
menu_item                          0
listed_in(type)                    0
listed_in(city)                    0
dtype: int64
```

In [7]:

```
data.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 [ ]:

```

```

Removing unnecessary columns -

In [8]:

```
df = data.drop(["url","address","phone","listed_in(city)","menu_item","dish_liked"],axis = 1)
```

In [9]:

```
df.head() #Now we will have the accurate data to perform visualization after cleaning it.
```

Out[9]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	approx_cost(for two people)	revi
0	Jalsa	Yes	Yes	4.1/5	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800	[('Ra 'RA pl
1	Spice Elephant	Yes	No	4.1/5	787	Banashankari	Casual Dining	Chinese, North Indian, Thai	800	[('Ra 'I H
2	San Churro Cafe	Yes	No	3.8/5	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	800	[('Ra "I Am n
3	Addhuri Udupi Bhojana	No	No	3.7/5	88	Banashankari	Quick Bites	South Indian, North Indian	300	[('Ra "I G and
4	Grand Village	No	No	3.8/5	166	Basavanagudi	Casual Dining	North Indian, Rajasthani	600	[('Ra 'I V rest

Removing Duplicates -

In [10]:

```
df.shape
```

Out[10]:

```
(51717, 11)
```

In [11]:

```
df.drop_duplicates(inplace =True)
```

In [12]:

```
df.shape
```

Out[12]:

```
(38930, 11)
```

In [13]:

```
print("Number of Rows droped- ",51717 - 38930)
```

```
Number of Rows droped-  12787
```

Analysing each and every columns -

Column - "rate"

In [14]:

```
df["rate"].head()
```

Out[14]:

```
0    4.1/5
1    4.1/5
2    3.8/5
3    3.7/5
4    3.8/5
Name: rate, dtype: object
```

In [15]:

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

Out[15]:

```
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', 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'], dtype=object)
```

Rating we have provided is outof 5 so we will convert it into a float removing "/5" part of it then we will deal with "NEW" , "-"

In [16]:

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

Out[16]:

```
0    4.1
1    4.1
2    3.8
3    3.7
4    3.8
Name: rate, dtype: float64
```

In [17]:

```
df.head()
```

Out[17]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	approx_cost(for two people)	revie
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800	[('Ra 'RA' b pla
1	Spice Elephant	Yes	No	4.1	787	Banashankari	Casual Dining	Chinese, North Indian, Thai	800	[('Ra 'R Ha
2	San Churro Cafe	Yes	No	3.8	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	800	[('Ra "R Amb nc
3	Addhuri Udupi Bhojana	No	No	3.7	88	Banashankari	Quick Bites	South Indian, North Indian	300	[('Ra "R Gr and
4	Grand Village	No	No	3.8	166	Basavanagudi	Casual Dining	North Indian, Rajasthani	600	[('Ra 'R Ve resta

In [18]:

```
df = df.drop(["reviews_list"] , axis =1) # removing the reviews_list column
```

In [19]:

```
df.head()
```

Out[19]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	approx_cost(for two people)	liste
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	

Fixing null values -

In [20]:

```
df.rate.isnull().sum()
```

Out[20]:

```
5344
```

In [21]:

```
df["rate"].fillna(df["rate"].mean(),inplace = True)
df["rate"].isnull().sum()
```

Out[21]:

```
0
```

In [22]:

```
df.isnull().sum()
```

Out[22]:

```
name                            0
online_order                    0
book_table                      0
rate                            0
votes                           0
location                       14
rest_type                     162
cuisines                       33
approx_cost(for two people)   253
listed_in(type)                 0
dtype: int64
```

In [23]:

```
'''
fixing other null values
1. location can't be fixed . Data insufiecient issue.
2. same goes with rest_type, cuisines, approx_cost(for two people)
therefore -
'''
```

Out[23]:

```
"\nfixing other null values  \n1. location can't be fixed . Data insufiecient issue.\n
2. same goes with rest_type, cuisines, approx_cost(for two people)\ntherefore - \n"
```

In [24]:

```
df.dropna(inplace = True)
```

In [25]:

```
df.isnull().sum()
```

Out[25]:

```
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
listed_in(type)                0
dtype: int64
```

Renaming columns -

In [26]:

```
df.head()
```

Out[26]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	approx_cost(for two people)	liste
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	

In [27]:

```
df.rename(columns = {"approx_cost(for two people)":"cost2","listed_in(type)":"types"},inplace = 
df.head()
```

Out[27]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	cost2	types
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800	Buffet
1	Spice Elephant	Yes	No	4.1	787	Banashankari	Casual Dining	Chinese, North Indian, Thai	800	Buffet
2	San Churro Cafe	Yes	No	3.8	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	800	Buffet
3	Addhuri Udupi Bhojana	No	No	3.7	88	Banashankari	Quick Bites	South Indian, North Indian	300	Buffet
4	Grand Village	No	No	3.8	166	Basavanagudi	Casual Dining	North Indian, Rajasthani	600	Buffet

Column - "location"

In [28]:

```
df["location"].isnull().sum()
```

Out[28]:

```
0
```

In [29]:

```
df["location"].unique()  # everythings is right with location so moving to next section
```

Out[29]:

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

Column - "cost2" -

In [30]:

```
df["cost2"].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', '50', '190', '1,700', '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'], dtype=object)
```

In [31]:

```
'''

Here data looks normal but when we closly observe values > 1000 has a comma in between therefore
converted into float type

'''
```

Out[31]:

```
'\n\nHere data looks normal but when we closly observe values > 1000 has a comma in be
tween therefore it wont be \nconverted into float type\n\n'
```

In [32]:

```
def handling_cost2(value):
    if "," in value:
        value = value.replace(",","")
        return float(value)
    else :
        return float(value)
df["cost2"] = df["cost2"].apply(handling_cost2)
df["cost2"].unique()
```

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.,   50.,  190., 1700., 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.])
```

In [33]:

```
df.head()
```

Out[33]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	cost2	types
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800.0	Buffet
1	Spice Elephant	Yes	No	4.1	787	Banashankari	Casual Dining	Chinese, North Indian, Thai	800.0	Buffet
2	San Churro Cafe	Yes	No	3.8	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	800.0	Buffet
3	Addhuri Udupi Bhojana	No	No	3.7	88	Banashankari	Quick Bites	South Indian, North Indian	300.0	Buffet
4	Grand Village	No	No	3.8	166	Basavanagudi	Casual Dining	North Indian, Rajasthani	600.0	Buffet

Column - "cuisines"

In [34]:

```
df["cuisines"].unique()
```

Out[34]:

```
array(['North Indian, Mughlai, Chinese', 'Chinese, North Indian, Thai',
       'Cafe, Mexican, Italian', ...,
       'North Indian, Street Food, Biryani', 'Chinese, Mughlai',
       'North Indian, Chinese, Arabian, Momos'], dtype=object)
```

In [35]:

```
df["cuisines"].value_counts()
```

Out[35]:

```
North Indian                              2121
North Indian, Chinese                     1749
South Indian                              1277
Bakery, Desserts                           640
Biryani                                    601
                                          ... 
Bengali, Chinese, Seafood                    1
Burger, Chinese, Fast Food, Pizza, Sandwich        1
Steak, Beverages                             1
Salad, Juices                                1
North Indian, Chinese, Arabian, Momos        1
Name: cuisines, Length: 2704, dtype: int64
```

In [36]:

```
"""
After carefull observation of the values we can observe that their are many cuisines which has no
value count which has now direct effect on the obseration so we have moved those value in the "o

"""
```

Out[36]:

```
'\nAfter carefull observation of the values we can observe that their are many cuisine
s which has no repetation and has low \nvalue count which has now direct effect on the
obseration so we have moved those value in the "other" section .\n\n'
```

In [37]:

```
df1 = df["cuisines"].value_counts(ascending=False)
```

In [38]:

```
cuisineslessthan100 = df1[df1<100]
```

In [39]:

```
def handling_cuisines(value):
    if value in cuisineslessthan100:
        return "other"
    else :
        return value
df["cuisines"]=df["cuisines"].apply(handling_cuisines)
df["cuisines"].value_counts()
```

Out[39]:

```
other                                              21546
North Indian                                        2121
North Indian, Chinese                               1749
South Indian                                        1277
Bakery, Desserts                                     640
Biryani                                              601
Fast Food                                            536
South Indian, North Indian, Chinese                  517
Desserts                                             506
Cafe                                                 498
Bakery                                               424
Chinese                                              369
Ice Cream, Desserts                                  330
Chinese, North Indian                                286
Mithai, Street Food                                  277
North Indian, Chinese, Biryani                       259
Desserts, Ice Cream                                  240
North Indian, South Indian                           233
North Indian, South Indian, Chinese                  227
South Indian, North Indian                           224
Finger Food                                          224
Desserts, Beverages                                  224
North Indian, Biryani                                206
Street Food                                          190
Biryani, North Indian                                188
Biryani, Kebab                                       187
Chinese, Momos                                       180
Beverages, Fast Food                                 177
Beverages                                            176
Cafe, Fast Food                                      171
South Indian, Biryani                                163
North Indian, Mughlai                                160
Desserts, Bakery                                     158
South Indian, North Indian, Chinese, Street Food     156
South Indian, Chinese                                154
Burger, Fast Food                                    154
Continental                                          151
Cafe, Continental                                    146
North Indian, Fast Food                              144
Ice Cream                                            144
Fast Food, Beverages                                 144
Kerala                                               143
Pizza, Fast Food                                     133
Chinese, Thai                                        132
North Indian, Chinese, Fast Food                     131
Bakery, Fast Food                                    131
Biryani, North Indian, Chinese                       129
North Indian, Chinese, South Indian                  128
Fast Food, Rolls                                     127
Pizza                                                122
Andhra                                               120
Cafe, Desserts                                       117
Biryani, Fast Food                                   116
Andhra, Biryani                                      112
Arabian                                              110
North Indian, Street Food                            106
Fast Food, Burger                                    103
North Indian, Chinese, Continental                   100
Name: cuisines, dtype: int64
```

Column - "location"

In [40]:

```
df.head()
```

Out[40]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	cost2	types
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	other	800.0	Buffet
1	Spice Elephant	Yes	No	4.1	787	Banashankari	Casual Dining	other	800.0	Buffet
2	San Churro Cafe	Yes	No	3.8	918	Banashankari	Cafe, Casual Dining	other	800.0	Buffet
3	Addhuri Udupi Bhojana	No	No	3.7	88	Banashankari	Quick Bites	South Indian, North Indian	300.0	Buffet
4	Grand Village	No	No	3.8	166	Basavanagudi	Casual Dining	other	600.0	Buffet

In [41]:

```
df["location"].unique()
```

Out[41]:

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

In [42]:

```
df["location"].value_counts()
```

Out[42]:

```
BTM                          3001
Koramangala 5th Block        1991
Whitefield                   1964
HSR                          1931
Indiranagar                  1830
                             ...
Yelahanka                       5
Nagarbhavi                      4
Rajarajeshwari Nagar            2
Jakkur                          1
Peenya                          1
Name: location, Length: 93, dtype: int64
```

We have the same observation as "cuisines" column so we will handle it accordingly.

In [43]:

```
location1 = df["location"].value_counts()
```

In [63]:

```
locationlessthan500 = location1[location1<300]
```

In [64]:

```
def handlinglocation(value):
    if value in locationlessthan500:
        return "others"
    else:
        return value
df["location"] = df["location"].apply(handlinglocation)
df["location"].value_counts()
```

Out[64]:

```
others                   9807
BTM                      3001
Koramangala 5th Block    1991
Whitefield               1964
HSR                      1931
Indiranagar              1830
Marathahalli             1664
JP Nagar                 1584
Jayanagar                1424
Electronic City          1214
Bannerghatta Road        1153
Bellandur                1138
Sarjapur Road             957
Koramangala 7th Block     867
Koramangala 6th Block     835
Brigade Road              781
Koramangala 4th Block     756
Koramangala 1st Block     716
Kalyan Nagar              663
MG Road                   643
Ulsoor                    643
Banashankari              635
Brookefield               607
Malleshwaram              596
New BEL Road              594
Frazer Town               523
Name: location, dtype: int64
```

Column - "rest_type"

In [65]:

```
df.head()
```

Out[65]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	cost2	types
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	other	800.0	Buffet
1	Spice Elephant	Yes	No	4.1	787	Banashankari	Casual Dining	other	800.0	Buffet
2	San Churro Cafe	Yes	No	3.8	918	Banashankari	others	other	800.0	Buffet
3	Addhuri Udupi Bhojana	No	No	3.7	88	Banashankari	Quick Bites	South Indian, North Indian	300.0	Buffet
4	Grand Village	No	No	3.8	166	others	Casual Dining	other	600.0	Buffet

In [66]:

```
df['rest_type'].value_counts()
```

Out[66]:

```
Quick Bites           13318
Casual Dining          8499
others                 7943
Cafe                   2938
Delivery               1731
Dessert Parlor         1672
Takeaway, Delivery     1383
Casual Dining, Bar     1033
Name: rest_type, dtype: int64
```

In [67]:

```
rest_type = df["rest_type"].value_counts(ascending = False)
```

In [68]:

```
resttypelessthan1000 = rest_type[rest_type<1000]
```

In [69]:

```
def handlingresttype(value):
    if value in resttypelessthan1000:
        return "others"
    else :
        return value
df["rest_type"] = df["rest_type"].apply(handlingresttype)
df["rest_type"].value_counts()
```

Out[69]:

```
Quick Bites           13318
Casual Dining          8499
others                 7943
Cafe                   2938
Delivery               1731
Dessert Parlor         1672
Takeaway, Delivery     1383
Casual Dining, Bar     1033
Name: rest_type, dtype: int64
```

Hence our data is clean now, we will move to visualization part and understand the business Problems.

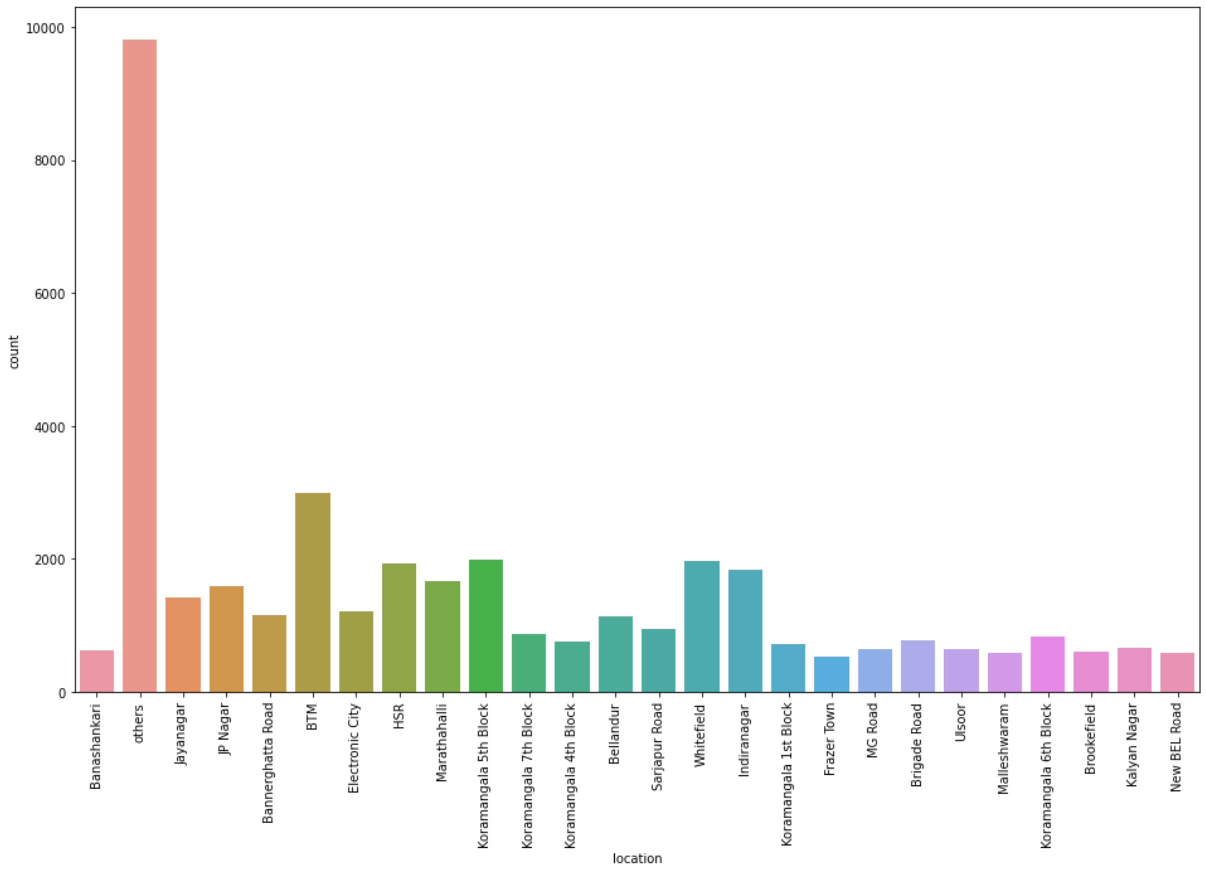
Visualization-

Count Plot for various locations -

In [70]:

```python
plt.figure(figsize= (16,10))
x =sns.countplot(df["location"])
plt.xticks(rotation = 90)
plt.show()
```

```
C:\Users\Ankit\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: P
ass the following variable as a keyword arg: x. From version 0.12, the only valid posi
tional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```



In [ ]:

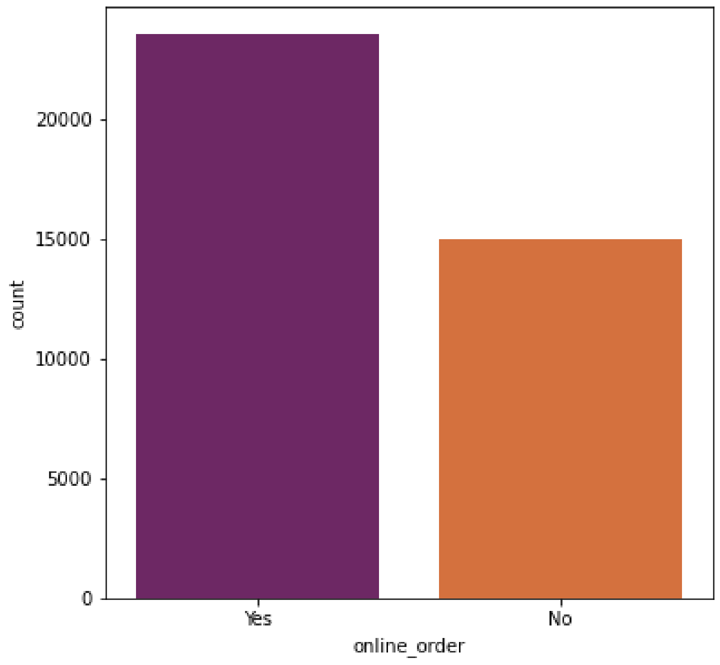
```
"""
Observation -

1. There many restaurants present in  **BTM** location .Therefore we need to find another locati
2. There are many location which have less number of restarurant.

"""
```

Visualising - "online_order"

In [73]:

```python
plt.figure(figsize= (6,6))
sns.countplot(df["online_order"] , palette = "inferno")
plt.show()
```

```
C:\Users\Ankit\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: P
ass the following variable as a keyword arg: x. From version 0.12, the only valid posi
tional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```



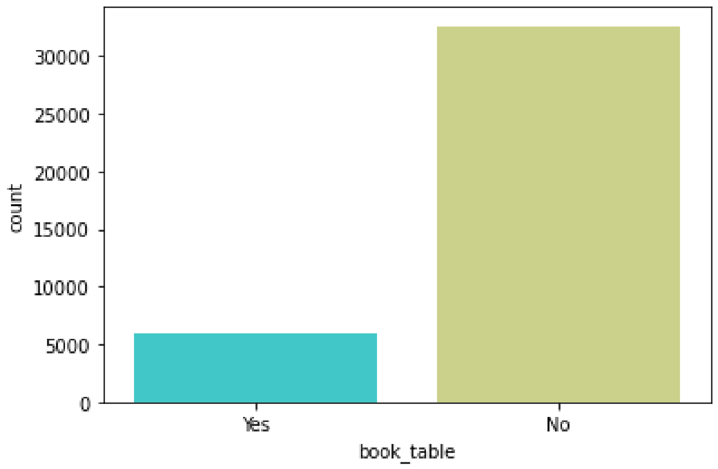
In [ ]:

```python
"""

Observation - Many restaurants have online facility .

"""
```

In [76]:

```
sns.countplot(df['book_table'] ,palette = "rainbow")
plt.show()
```

```
C:\Users\Ankit\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: P
ass the following variable as a keyword arg: x. From version 0.12, the only valid posi
tional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```



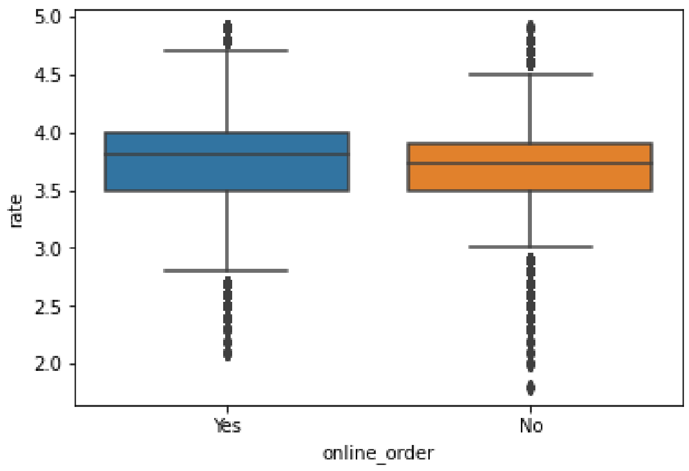
In [ ]:

```
"""
Observation - Maximum restaurants does not have Book table facility in advance.
"""
```

Visualising - "online_order" - "rate"

In [78]:

```
sns.boxplot(x= 'online_order', y = 'rate' , data = df)
plt.show()
```



In [ ]:

```
"""
Observation - Restaurants having online order facility has the maximum rating.

"""
```

In [79]:

```
df.head()
```

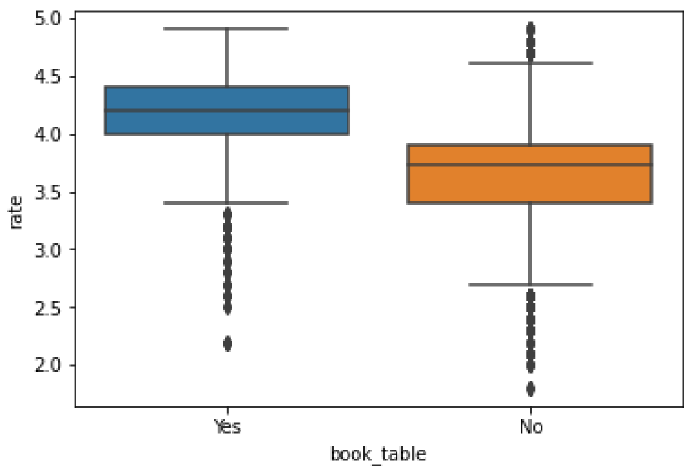
Out[79]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	cost2	types
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	other	800.0	Buffet
1	Spice Elephant	Yes	No	4.1	787	Banashankari	Casual Dining	other	800.0	Buffet
2	San Churro Cafe	Yes	No	3.8	918	Banashankari	others	other	800.0	Buffet
3	Addhuri Udupi Bhojana	No	No	3.7	88	Banashankari	Quick Bites	South Indian, North Indian	300.0	Buffet
4	Grand Village	No	No	3.8	166	others	Casual Dining	other	600.0	Buffet

Visualising - "book_table" - "rate"

In [80]:

```
sns.boxplot(x="book_table",y="rate",data =df)
plt.show()
```



In [ ]:

```
"""
Observation - Restaurants having online booking facility has the maximum rating.

"""
```

Visualising - "online_order" - "location"

In [81]:

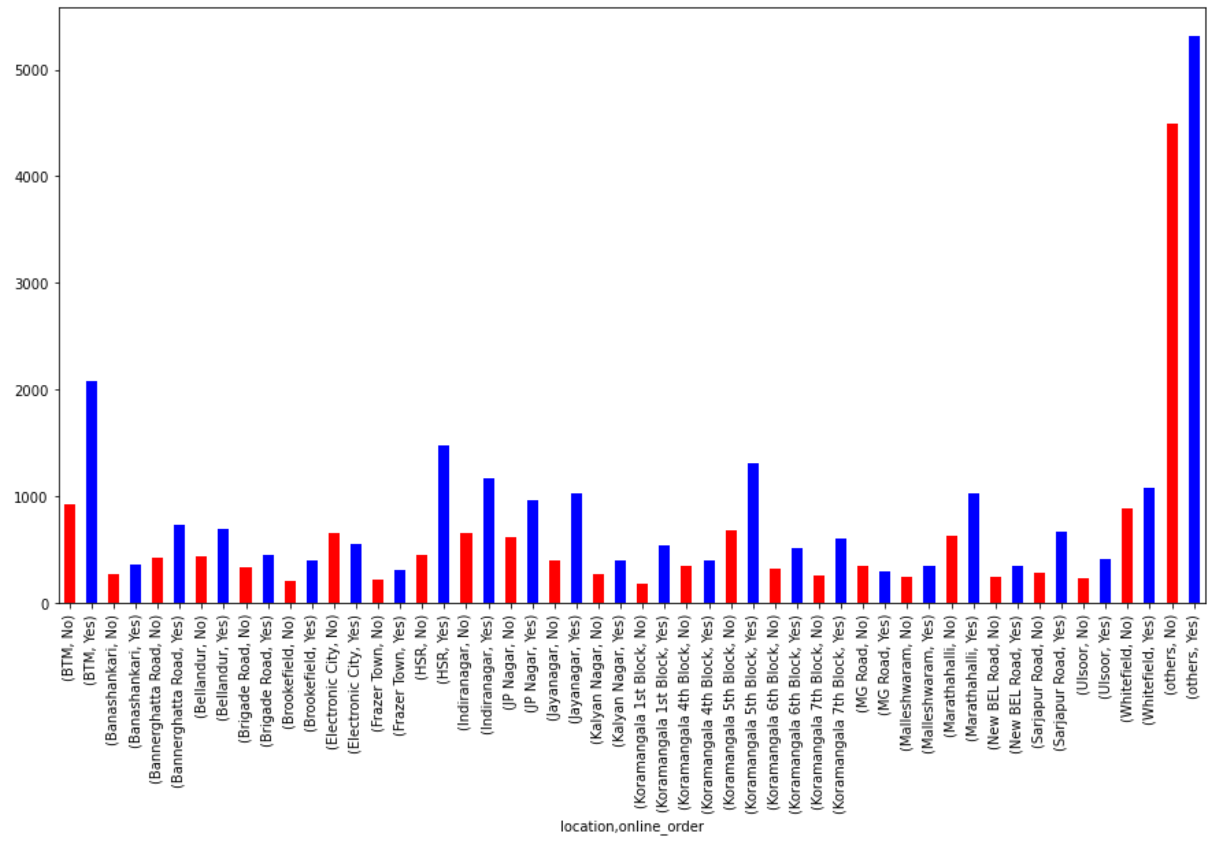
```
df1 = df.groupby(["location","online_order"])["name"].count()
```

In [84]:

```
df1.plot(kind = "bar",figsize = (15,8),color = ["r","b"])
```

Out[84]:

```
<AxesSubplot:xlabel='location,online_order'>
```



In [ ]:

```
"""
Observation - Blueline indicate restaurant having online facility and red line shows their are no
              online order facility.
"""
```

Visualising - "book_table" - "location"

In [88]:

```
oupby(["location","book_table"])["name"].count()
('location_booktable.csv')
ead_csv('location_booktable.csv')
ivot_table(df2, values=None,index=['location'], columns=['book_table'],fill_value =0 ,aggfunc = np.su

```

In [89]:

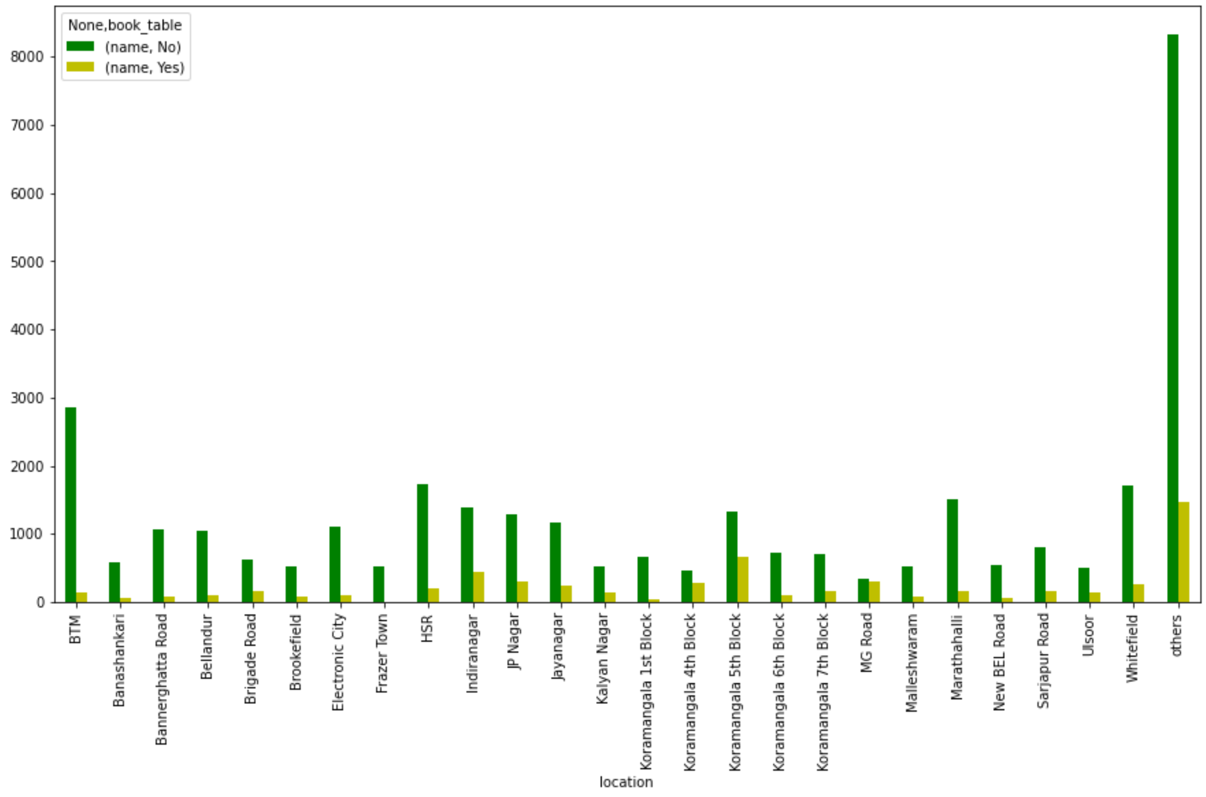
```
df2
```

Out[89]:

	name	
book_table	**No**	**Yes**
location		
BTM	2857	144
Banashankari	583	52
Bannerghatta Road	1062	91
Bellandur	1042	96
Brigade Road	624	157
Brookefield	534	73
Electronic City	1116	98
Frazer Town	515	8
HSR	1734	197
Indiranagar	1382	448
JP Nagar	1289	295
Jayanagar	1175	249
Kalyan Nagar	529	134
Koramangala 1st Block	666	50
Koramangala 4th Block	471	285
Koramangala 5th Block	1320	671
Koramangala 6th Block	731	104
Koramangala 7th Block	712	155
MG Road	338	305
Malleshwaram	514	82
Marathahalli	1502	162
New BEL Road	538	56
Sarjapur Road	801	156
Ulsoor	510	133
Whitefield	1711	253
others	8331	1476

In [90]:

```
a = ["g","y"]
df2.plot(kind="bar",figsize = (15,8),color = a)
plt.show()
```



In [ ]:

```
"""
Observation - Restuarants with Table booking facility has the highest rating.

"""
```

In [91]:

```
df.head()
```

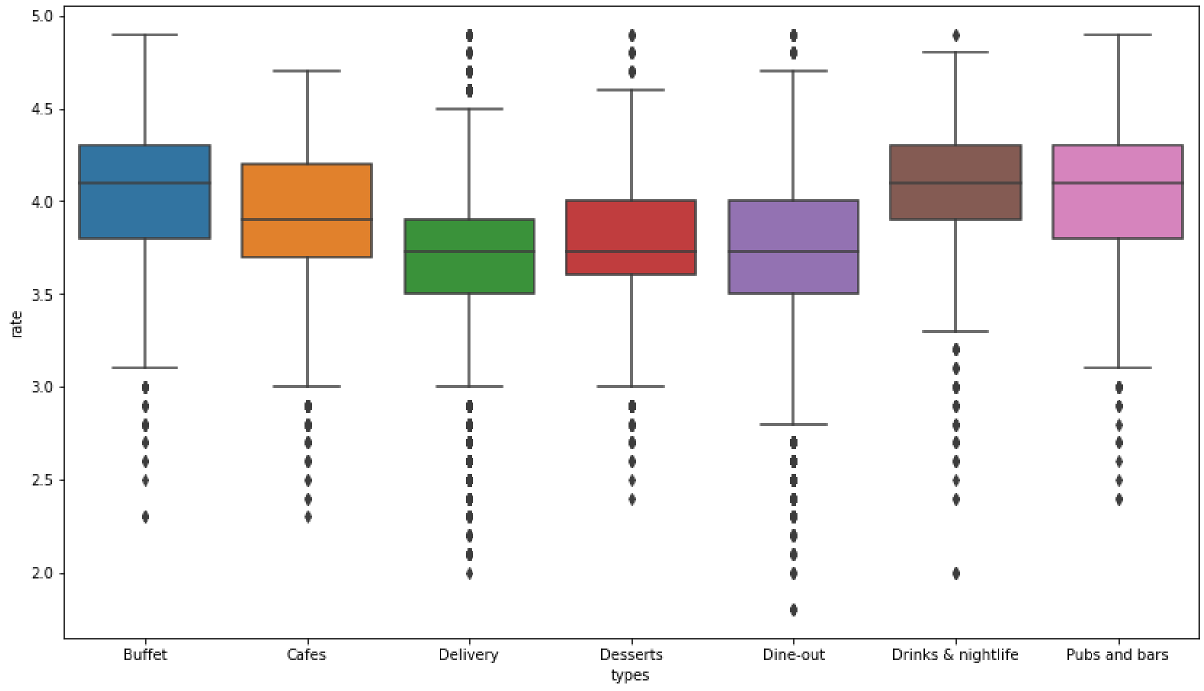
Out[91]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	cost2	types
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	other	800.0	Buffet
1	Spice Elephant	Yes	No	4.1	787	Banashankari	Casual Dining	other	800.0	Buffet
2	San Churro Cafe	Yes	No	3.8	918	Banashankari	others	other	800.0	Buffet
3	Addhuri Udupi Bhojana	No	No	3.7	88	Banashankari	Quick Bites	South Indian, North Indian	300.0	Buffet
4	Grand Village	No	No	3.8	166	others	Casual Dining	other	600.0	Buffet

Visualising - "type" - "rate"

In [93]:

```
plt.figure(figsize=(14,8))
sns.boxplot(x= 'types',y='rate',data = df)
plt.show()
```



In [ ]:

```
"""
Observation - Restarurants like -- Drinks & nightlife , pub and bars have the highest rating.

"""
```

Let's Group the types of restaurants location wise -

In [97]:

```
df3 = df.groupby(['location','types'])['name'].count()

df3.to_csv('location_type.csv')
df3 = pd.read_csv('location_type.csv')
df3 = pd.pivot_table(df3,values=None,index=['location'],columns=['types'],fill_value =0 ,aggfunc
df3
```

Out[97]:

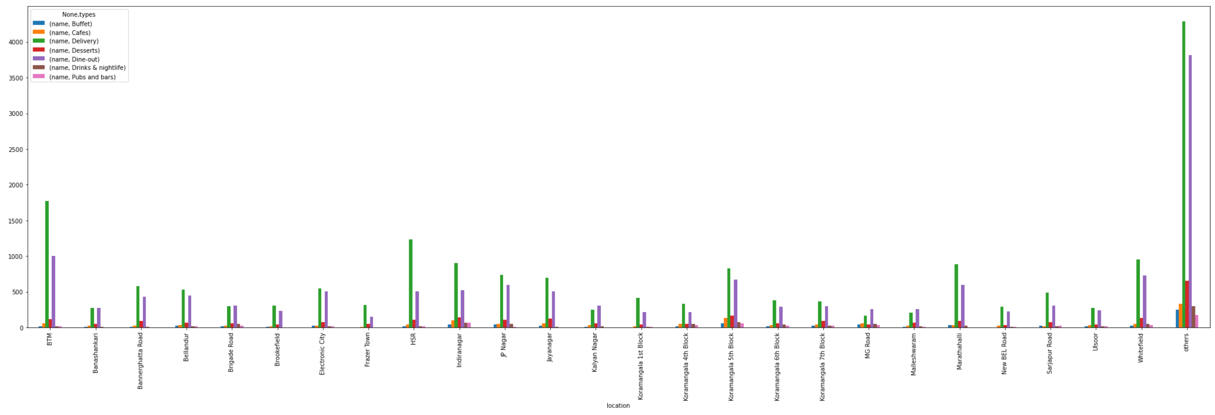
	name						
types	**Buffet**	**Cafes**	**Delivery**	**Desserts**	**Dine-out**	**Drinks & nightlife**	**Pubs and bars**
location							
BTM	19	55	1771	119	1004	18	15
Banashankari	6	29	274	47	272	7	0
Bannerghatta Road	9	29	576	94	434	9	2
Bellandur	28	34	528	70	445	17	16
Brigade Road	13	28	296	62	307	53	22
Brookefield	6	17	306	42	232	4	0
Electronic City	23	24	549	71	505	21	21
Frazer Town	1	10	311	48	149	2	2
HSR	15	43	1230	111	502	14	16
Indiranagar	38	97	906	137	519	67	66
JP Nagar	40	48	738	111	593	47	7
Jayanagar	22	59	697	125	509	12	0
Kalyan Nagar	9	31	249	55	303	16	0
Koramangala 1st Block	3	20	417	41	218	6	11
Koramangala 4th Block	15	50	333	54	219	53	32
Koramangala 5th Block	58	134	824	170	669	78	58
Koramangala 6th Block	14	36	381	56	286	39	23
Koramangala 7th Block	22	41	364	92	301	22	25
MG Road	39	58	168	40	257	49	32
Malleshwaram	11	24	209	67	255	18	12
Marathahalli	34	30	884	94	597	22	3
New BEL Road	4	29	289	33	223	8	8
Sarjapur Road	26	19	488	78	304	20	22
Ulsoor	13	36	274	42	242	18	18
Whitefield	28	50	950	129	728	46	33
others	252	330	4284	653	3816	295	177

In [101]:

```python
df3.plot(kind = "bar" , figsize = (36,10))
plt.show
```

Out[101]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [ ]:

```python
"""
Observation :Best opening place for pub and bar will be the location with very few of them.

"""
```

Number of votes Location Wise -

In [102]:

```python
df4 = df[['location','votes']]
df4.drop_duplicates()
df5 = df4.groupby(['location'])['votes'].sum()
df5 = df5.to_frame()
df5 = df5.sort_values('votes',ascending=False)
df5.head()
```

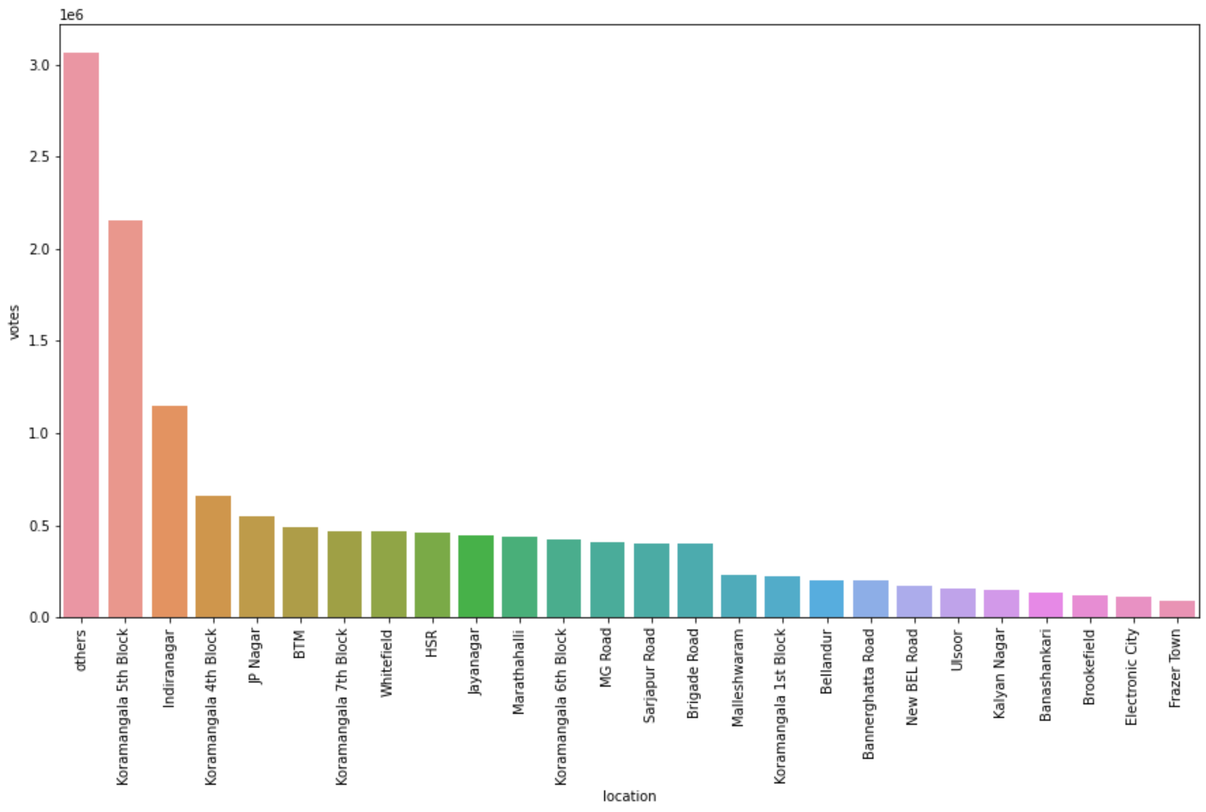
Out[102]:

	votes
location	
others	3066235
Koramangala 5th Block	2158161
Indiranagar	1150691
Koramangala 4th Block	657592
JP Nagar	548854

In [103]:

```
plt.figure(figsize=(15,8))
sns.barplot(df5.index, df5['votes'])
plt.xticks(rotation = 90)
plt.show()
```

```
C:\Users\Ankit\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: P
ass the following variables as keyword args: x, y. From version 0.12, the only valid p
ositional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```



In [ ]:

```
"""
Observation - People who are Koramangala 5th Block and Indiranagar they are only intrested in Vo

"""
```

Visualizing Top Cuisines

In [104]:

```
df6 = df[['cuisines','votes']]
df6.drop_duplicates()
df7 = df6.groupby(['cuisines'])['votes'].sum()
df7 = df7.to_frame()
df7 = df7.sort_values('votes',ascending=False)
df7.head()
```

Out[104]:

	votes
cuisines	
other	11269955
North Indian	486047
North Indian, Chinese	223104
South Indian	150280
North Indian, Mughlai	98129

In [105]:

```
# Drop "other" first-
df7 = df7.iloc[1:,:]
df7.head()
```

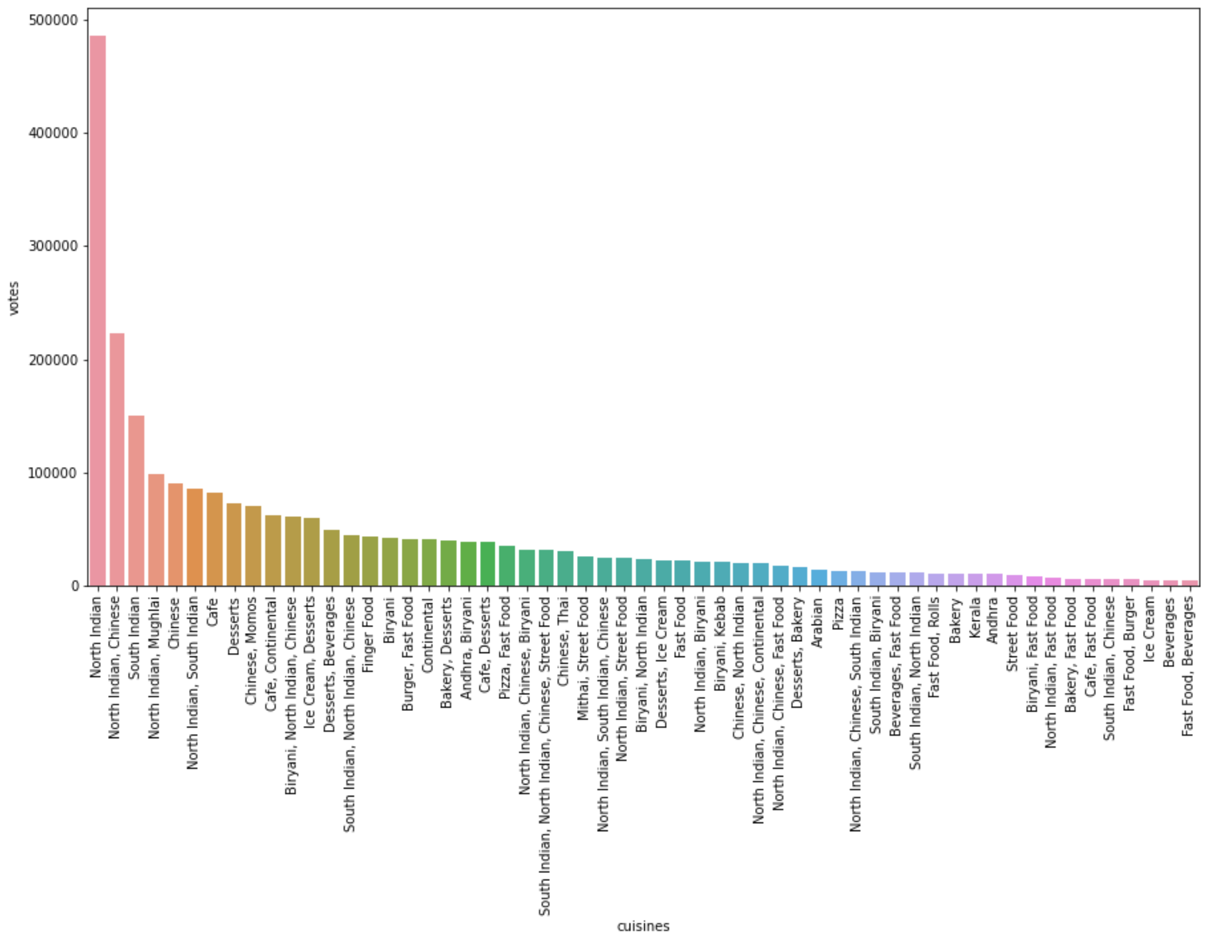
Out[105]:

	votes
cuisines	
North Indian	486047
North Indian, Chinese	223104
South Indian	150280
North Indian, Mughlai	98129
Chinese	90511

In [107]:

```
plt.figure(figsize=(15,8))
sns.barplot(df7.index,df7['votes'])
plt.xticks(rotation = 90)
plt.show()
```

```
C:\Users\Ankit\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: P
ass the following variables as keyword args: x, y. From version 0.12, the only valid p
ositional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```



In [ ]:

```
"""
Observation - Most demanded cuisine in Bengaluru is -"North Indian" , "Chinese" , "south Indian"
              "Mughlai". Therefore opening a restaurant in Bengaluru  with all these observation
              will be Profitable.

"""
```

thankyou.™