[Analysis of Restaurant Registered with Zomato[Bengaluru]

“

Project Name:

Analysis of Restaurant Registered with Zomato[Bengaluru]

Dataset link:

https://www.kaggle.com/datasets/rajeshrampure/zomato-dataset

Project Aim:

1) Displaying online and offline orders.
2) Tables booked.
3) Most Preferred Locations.
4) Most Votes Restaurant.
5) Most Rating Restaurant.
6) Most Preferred Restaurant Types.
7) Most Preferred Dishes.
8) Cuisines
9) Approximate cost of two people.
10) Service Types.
11) Listed Cities.
12) Locations.
13) Data Preprocessing.

<Code>:

```
# Importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# Importing dataset
zomato_df = pd.read_csv('zomato.csv')

# Looking first 5 rows of data
print(zomato_df.head())

'''
                                                 url
address  ... listed_in(type) listed_in(city)
0  https://www.zomato.com/bangalore/jalsa-banasha...  942, 21st Main Road,
2nd Stage, Banashankari, ...  ...          Buffet    Banashankari
1  https://www.zomato.com/bangalore/spice-elephan...  2nd Floor, 80 Feet
Road, Near Big Bazaar, 6th ...  ...          Buffet    Banashankari
2  https://www.zomato.com/SanchurroBangalore?cont...  1112, Next to KIMS
Medical College, 17th Cross...  ...          Buffet    Banashankari
3  https://www.zomato.com/bangalore/addhuri-udupi...  1st Floor,
Annakuteera, 3rd Stage, Banashankar...  ...          Buffet
 Banashankari
4  https://www.zomato.com/bangalore/grand-village...  10, 3rd Floor,
Lakshmi Associates, Gandhi Baza...  ...          Buffet    Banashankari

'''
# Looking last 5 rows of the dataset
print(zomato_df.tail())

'''
                                                   url
address  ... listed_in(type) listed_in(city)
51712  https://www.zomato.com/bangalore/best-brews-fo...  Four Points by
Sheraton Bengaluru, 43/3, White...  ...   Pubs and bars      Whitefield
51713  https://www.zomato.com/bangalore/vinod-bar-and...  Number 10,
Garudachar Palya, Mahadevapura, Whi...  ...   Pubs and bars
 Whitefield
51714  https://www.zomato.com/bangalore/plunge-sherat...  Sheraton Grand
Bengaluru Whitefield Hotel & Co...  ...   Pubs and bars      Whitefield
51715  https://www.zomato.com/bangalore/chime-sherato...  Sheraton Grand
Bengaluru Whitefield Hotel & Co...  ...   Pubs and bars      Whitefield
```

```
51716  https://www.zomato.com/bangalore/the-nest-the-...  ITPL Main Road,
KIADB Export Promotion Industr...  ...   Pubs and bars      Whitefield

'''

# Checking the Shape of the data frame
print(zomato_df.shape)

'''
(51717, 17)

'''
# Checking features
print(zomato_df.columns)

'''

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')

'''

# Checking info
print(zomato_df.info())

## Checking statistical summary of all features
print(zomato_df.describe(include= 'all'))

'''
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)

'''
# Checking null values
print(zomato_df.isnull().sum())

# Dropping all unnecessary columns
zomato_df = zomato_df.drop(['url', 'address',
'phone','dish_liked','reviews_list', 'menu_item'], axis=1)
print(zomato_df)

# Checking for duplicates values
print(zomato_df[zomato_df.duplicated()].columns)

# Dropping duplicates values
zomato_df = zomato_df.drop_duplicates()
print(zomato_df)

#Analysing "rate" columns since there are a lot of null values present
print(zomato_df['rate'].unique())
print(zomato_df['rate'])


# Defining a function to handle the rating column and also covert them
from text to integer form
def treat_ratings(values):
    if (values == 'NEW' or values == '-'):
        return np.nan                                   # NumPy NAN stands
for not a number and
                                                        #np.nan is defined
as a substitute for declaring values which are numerical values that are
missing values in an array as NumPy is used to deal with arrays in Python
    else :
        values = str(values).split('/')                 # since the split
method splits the string into a list of int
                                                        # i.e."4.1/5" will
split it into 4.1 and /5 ---> list is =   values=[4.1][/5]
```

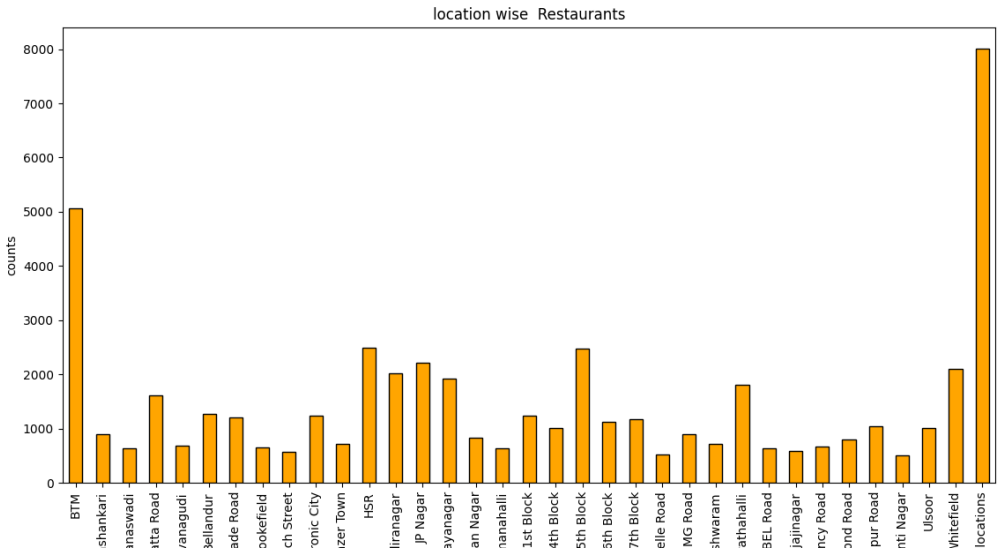
```
        values = values[0]                                  # But here we only
need the numerator of the rating because all the ratings are out of 5
                                                            # So we have stated
its index value.
        return float(values)                                # Since we want
these values in floating form.


# Now applying the function on the "rate" column
zomato_df.rate = zomato_df['rate'].apply(treat_ratings)
print(zomato_df['rate'].unique())
print(zomato_df.info())

# Checking null Values
print(zomato_df.isnull().sum())

# Using fill na method
print(zomato_df.rate.fillna(zomato_df.rate.mean(), inplace= True))
print(zomato_df.isnull().sum())

# So dropping other features with null values
print(zomato_df.dropna(inplace = True))
print(zomato_df.isnull().sum())

#Now analysing Restaurant type feature
print(zomato_df['rest_type'].value_counts())
rest_type = zomato_df['rest_type'].value_counts()
print(rest_type)

#for a better understanding of the feature we will group rest_type which
has less than 1000 counts
other_resto_types = rest_type[rest_type <1000]
print(other_resto_types)

# Checking types of data
print(type(other_resto_types))

# Defining Function
def handle_rest_type(type):
    if (type in other_resto_types):
        return 'other_resto_types'

    else:
        return type
```

```
zomato_df['rest_type'] = zomato_df['rest_type'].apply(handle_rest_type)
print(zomato_df.head())

print(zomato_df['rest_type'].value_counts())

# Now Analysing where these restaurants are located i.e. "location" column
print(zomato_df['location'])
print(zomato_df['location'].value_counts())


'''
Observation:

1) Most of the restaurants are in the BTM area i.e. 5056 restaurants and
from this, we can infer that this area is highly populated and has a
strong strong consumer base.
2) Since there are many locations present in the dataset which have less
number of restaurants it would be better for our analysis and
visualization to group these locations as we did for the rest_type
feature.
4) We will group the locations which have less than 500 restaurants and
store them in the "other location" variable.

'''

locations = zomato_df['location'].value_counts()
other_locations = locations[locations < 500]
print(other_locations)
print(other_locations.value_counts().sum())


# defining function
def handle_location (location):
    if location in other_locations :
        return 'other locations'
    else:
        return location

# applying function
zomato_df['location'] = zomato_df['location'].apply(handle_location)
print(zomato_df['location'].value_counts())

# Now Analysing "Cuisines" feature i.e. Food styles
print(zomato_df['cuisines'])
print(zomato_df['cuisines'].value_counts())
```

```
'''
Observation

1) From the cuisines column we can say that most of the food items in restaurants are from North Indian cuisine.
2) Followed by North Indian Chinese cuisine having 2351 items.
3) Also there are a lot of cuisines that are present in the data set which have fewer count of items.
4) So for good analysis, we will group these cuisines which have less than 100 counts of items in variable "less_num_cuisines"


'''

cuisines = zomato_df['cuisines'].value_counts()
less_num_cuisines = cuisines[cuisines <100]

# defining function
def handling_cusines(cuisines):
    if cuisines in less_num_cuisines:
        return 'less_num_cuisines'
    else:
        return cuisines

# Applying fun on the data frame
zomato_df['cuisines'] = zomato_df['cuisines'].apply(handling_cusines)
zomato_df['cuisines'].value_counts()
print(zomato_df.head())
print(zomato_df.info())

# Now analysing cost of plate per cuisines in restaurant i.e. ¶
"approx_cost(for two people) feature
print(zomato_df['approx_cost(for two people)'].value_counts())
print(zomato_df['approx_cost(for two people)'].unique())


'''
Observation :

1) as we see that data type of cost feature is an object because of comas in between numbers when the approx cost is above 999. i.e. 1,900 and so we have to remove these comas i.e. ","

'''
```

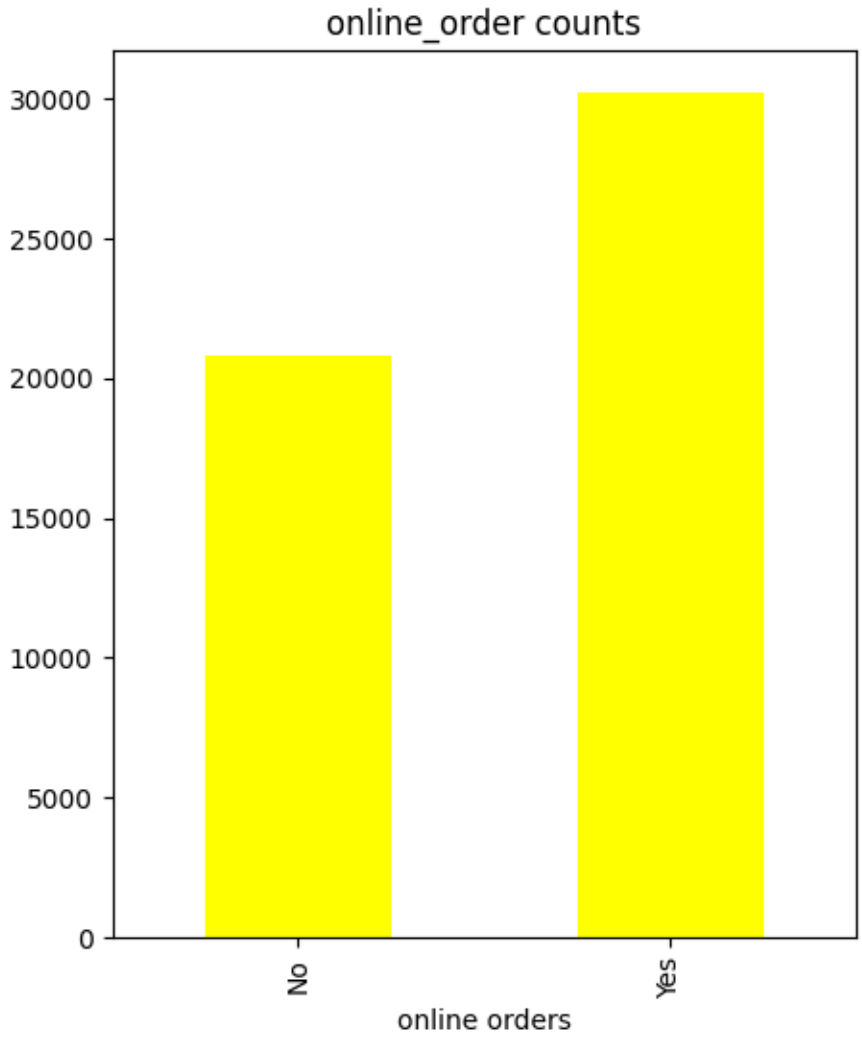
```
# Writing function to remove "," and convert this column into a float
def handling_approx_cost(cost):
    cost = str(cost)
    if "," in cost:
        cost = cost.replace("," , "")           # we simply replaced coma
with empty value
        return float(cost)                      # converting data type
into float
    else:
        return float(cost)

# applying fun on feature
zomato_df['approx_cost(for two people)'] = zomato_df['approx_cost(for two
people)'].apply(handling_approx_cost)
print(zomato_df['approx_cost(for two people)'].unique())
print(zomato_df['approx_cost(for two people)'].describe())


'''
Observation :

1) Now we have successfully performed operation on approx_cost feature
2) lowest approximate cost for two people is  Rs. 40.
3) highest approximate cost for two people is Rs. 6000.
'''

# Now Analysing listed_in(type) i.e. types of meals.
print(zomato_df['listed_in(type)'])
print(zomato_df['listed_in(type)'].value_counts())


'''
Observations :

1) In restaurants most types of meals are delivery meals i.e. 25579 which
infers that most people try to order food online from restaurants
2) Then 17562 meal types are of Dine-out type which infers that most of
the people love to eat at restaurants
3) 3559 meal types are Desserts or sweets.
4) 1703 meal types are of cafes.
5) 1084 types of meals are Drinks & nightlife type
6) 869  types of meals are buffet type.
7) 689 pubs and bars are present in a dataset of Bangalore

'''
```

```
# Since "location" and "listed_in(city)" both gives same meaning. i.e.
both columns show the area or location of restaurants in the dataset.
zomato_df = zomato_df.drop(columns= 'listed_in(city)')
print(zomato_df.head())
print(zomato_df.columns)

# Number of restaurants in different locations
print(zomato_df['location'].value_counts())


# Plotting Locations
counts = zomato_df['location'].value_counts().sort_index()
fig = plt.figure(figsize=(20, 7))
ax = fig.gca()

counts.plot.bar(ax = ax, color='Orange', edgecolor= 'black' )
ax.set_title( 'location'+ ' wise '+' Restaurants')
ax.set_xlabel('location')
ax.set_ylabel("counts")
plt.show()



'''
```




```
'''
```

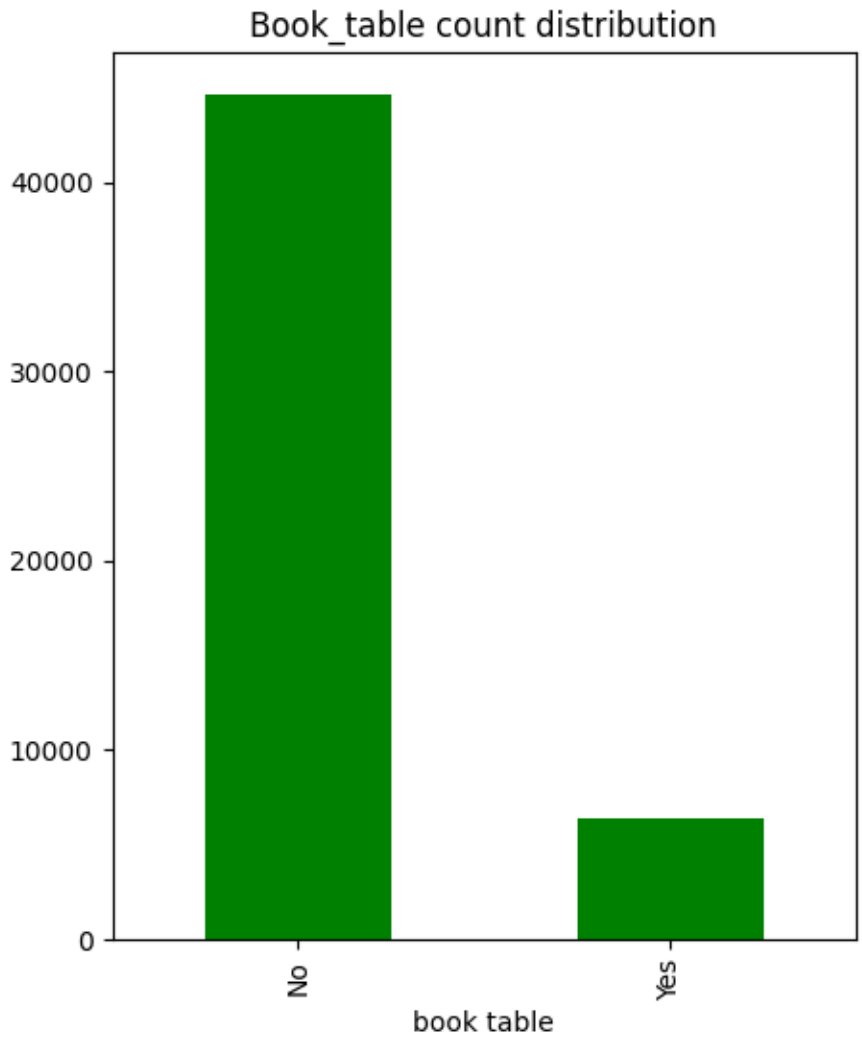
```
'''
Observation :

1) BTM place has 5000 plus restaurants
2) That means restaurants in BTM are where most people prefer to go or
order their food.
3) In the dataset there is a quite high count of locations which have less
than 500 restaurants and such locations are grouped into other locations
feature.
'''

# Checking the Restaurant's way of delivery.
print(zomato_df['online_order'].value_counts())
counts = zomato_df['online_order'].value_counts().sort_index()
fig = plt.figure(figsize=(5,6))
ax = fig.gca()

counts.plot.bar(ax = ax, color='yellow')
ax.set_title( 'online_order'+' counts')
ax.set_xlabel('online orders')
ax.set_ylabel("counts")
plt.show()



'''
```




```
'''
```

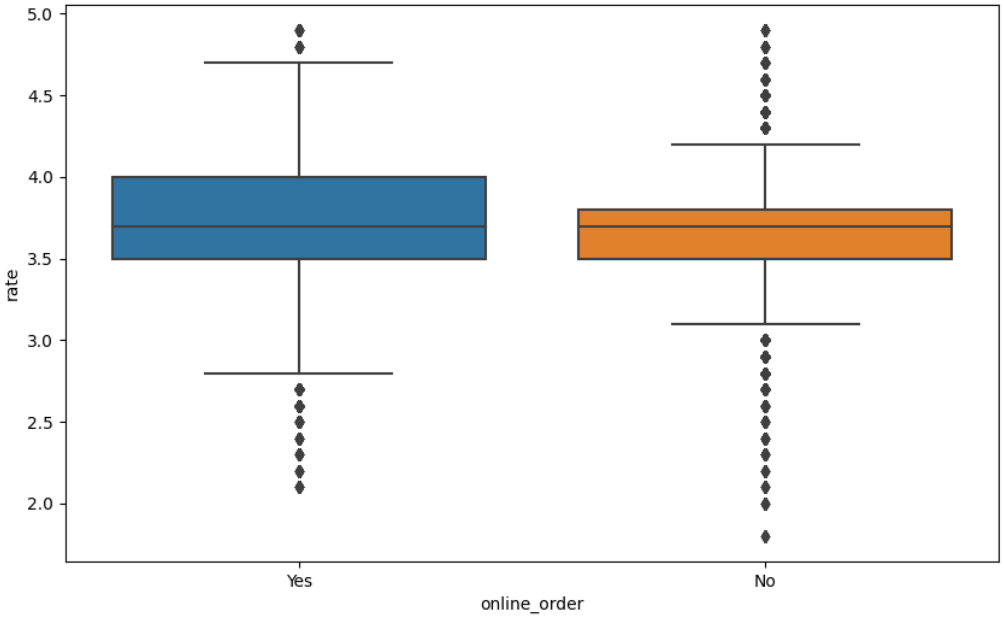
```
'''
Observations :

1) In Dataset 30228 restaurants have an online delivery facility
2) Whereas 20814 restaurants do not provide online delivery service.

'''

# Checking whether restaurants have a Table Booking facility or not!
print(zomato_df['book_table'].value_counts())
```

```
fig = plt.figure(figsize= (5,6))
counts = zomato_df['book_table'].value_counts()
ax= fig.gca()
counts.plot.bar(ax=ax, color='green')
plt.title('Book_table'+ ' count distribution')
plt.xlabel('book table')
plt.ylabel('counts')
plt.show()

'''
```



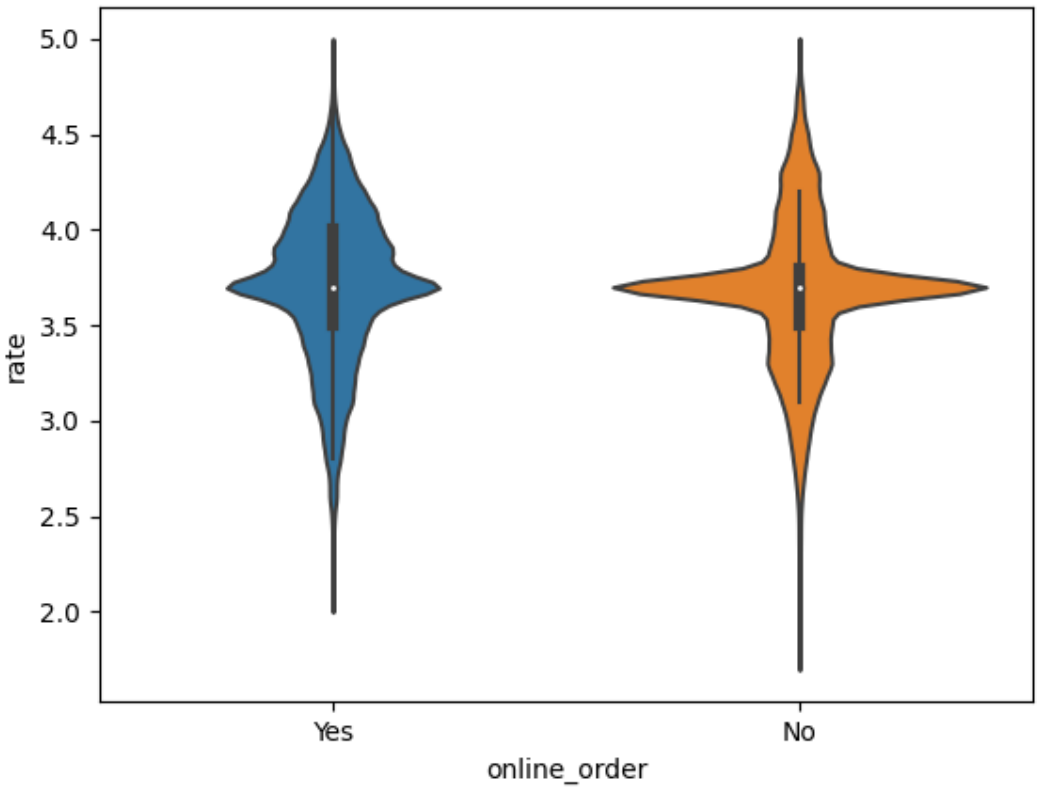

```
'''
'''
Observation :
```

```
1)6416 restaurants have a table booking facility.
2) Whereas 44626 restaurants don't provide this kind of service.

'''
# Checking how many Ratings are given by people to restaurants' online
delivery
print(zomato_df.head())

fig = plt.figure(figsize=(10,6))
ax= fig.gca()

sns.boxplot(x= zomato_df['online_order'], y= zomato_df['rate'])
plt.show()

'''
```




```
'''

'''
Observations = These Observations are due to Outliers present in features

1) Whether a restaurant is having online delivery service or not the
average rating of restaurants given by the customers is 3.70142
2) maximum rating given for restaurants providing online delivery is
around 4.7
```

```
3) minimum rating given for restaurants providing online delivery is
around 2.7
4) And when people go directly to restaurants they give a maximum rating
of 4.4
5) And when people go directly to restaurants they give a minimum rating
of 3.3

'''


# Ploting Violin plot to overcome outliers
sns.violinplot(x= zomato_df['online_order'], y= zomato_df['rate'])
plt.show()

'''
```



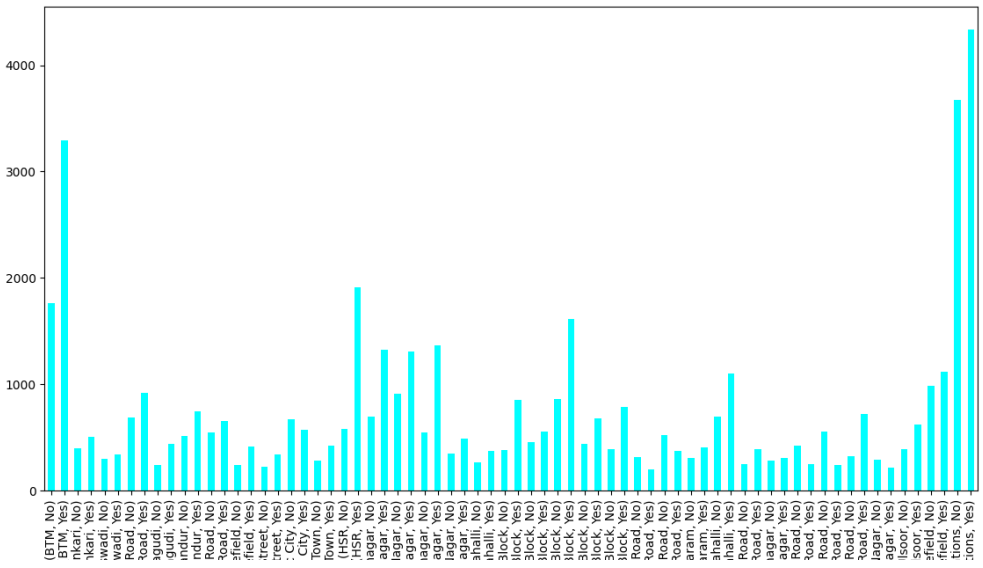

```
'''

'''
Actual Observations from the rating for restaurant's online delivery :
```

1) Whether a restaurant is having online delivery service or not the average rating of restaurants given by the customers is 3.70142
2) maximum rating given for restaurants providing online delivery is around 4.9
3) minimum rating given for restaurants providing online delivery is around 2.1
4) And when people go directly to restaurants they give a maximum rating of 4.9
5) And when people go directly to restaurants they give a minimum rating of 1.8

'''

```
# Checking From which Location of restaurants receiving most online orders
locationwise_online_order= zomato_df.groupby(['location','online_order'])
['name'].count()
print(locationwise_online_order)

locationwise_online_order.plot(kind= 'bar',color='cyan', figsize= (16,8))
plt.show()
```

'''



'''

'''

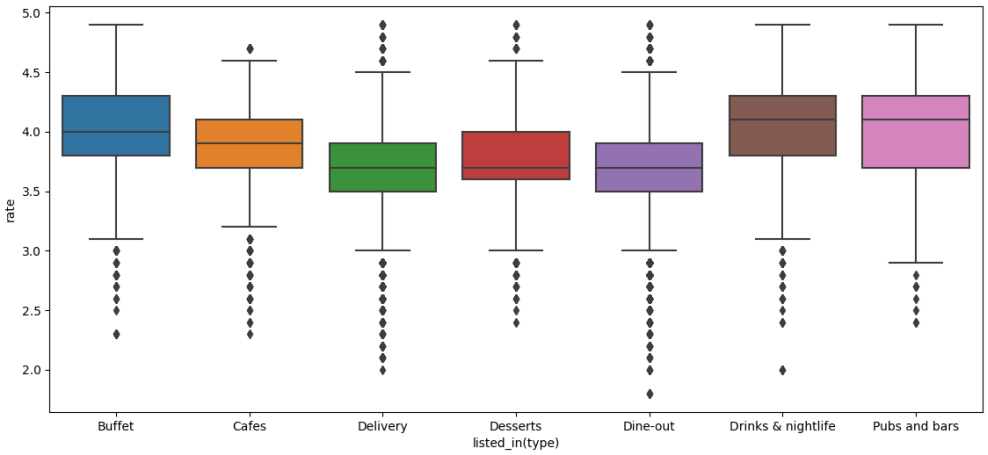
Observations :

1) As we have already seen that most of the restaurants are present in the BTM area and it is obvious that most of the restaurants present there will have online delivery facilities.
2) In the HSR location around 2000 restaurants have online delivery facilities.

'''

```
# Visualizing restaurants types and their ratings
plt.figure(figsize=(16,6))
sns.boxplot(x= zomato_df['listed_in(type)'], y= zomato_df['rate'])
plt.show()
```

'''



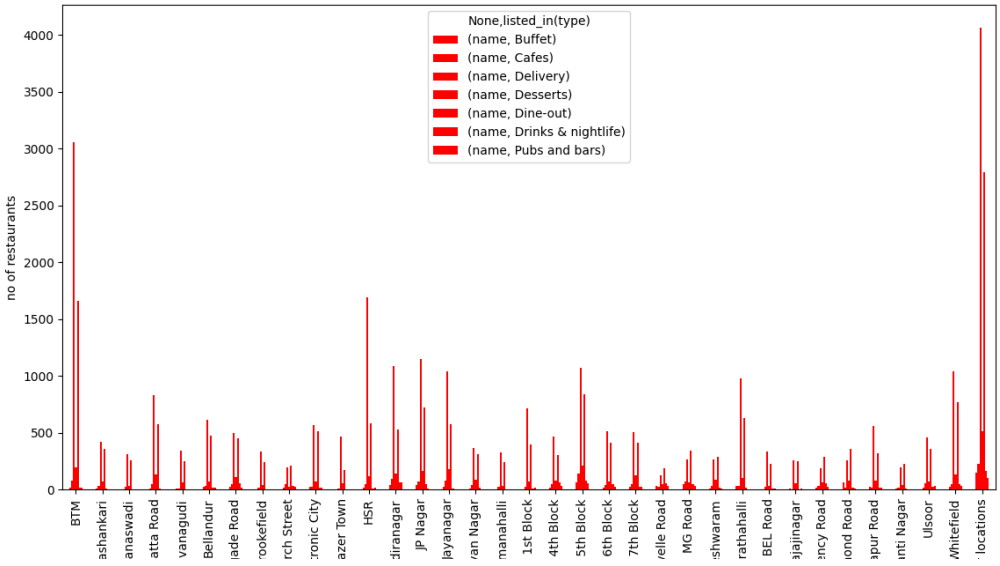
'''

'''
Observations :

1)Drinks and nightlife Restaurants types has a maximum average rating of 4.2 among the other 6 types of restaurants
2) Then Buffet restaurants have an average rating of 4.0
3) Restaurant types such as Delivery, Desserts, and Dine-out have some of the worst ratings given by the customer.

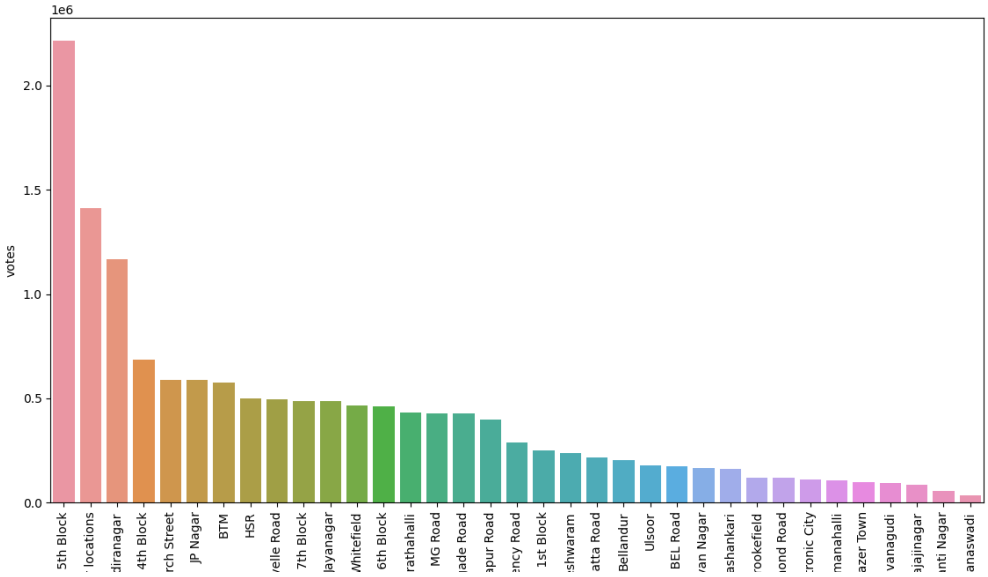
'''

```
# visualizing location-wise Restaurant types!
# creating a separate Dataframe of "location" as a Row index, and Types of
restaurants as (listed _in(type)) with their names as columns

loc_type_df = zomato_df.groupby(['listed_in(type)','location'])
['name'].count()

# Creating CSV File
loc_type_df.to_csv('loc_type.csv')
loc_type_df = pd.read_csv('loc_type.csv')
loc_type_df= pd.pivot_table(loc_type_df, values=None, index='location',
columns='listed_in(type)',fill_value=0, aggfunc= np.sum)
print(loc_type_df)


# plotting bar plot to get better visualization
loc_type_df.plot(kind = 'bar', color='red', figsize=(30,10))
plt.ylabel('no of restaurants')
plt.show()


'''
```




```
'''
```

```
# Checking which location got the most votes
# putting the "location" column and "votes" column in one data frame
loc_votes_df = zomato_df[['votes','location']]

# there are some duplicates values in the number of restaurants in the
location
# So Dropping duplicate values
print(loc_votes_df.drop_duplicates())

# summing all the votes for that particular location
loc_votes_df2=loc_votes_df.groupby(['location'])['votes'].sum()
loc_votes_df2=loc_votes_df2.to_frame()

#  now sorting the  Data in descending order of the number of votes
loc_votes_df2=loc_votes_df2.sort_values('votes',ascending=False)
print(loc_votes_df2.head())

# Now plotting Bar plot for better visualization
plt.figure(figsize= (27,8))
sns.barplot(x= loc_votes_df2.index,y= loc_votes_df2['votes'] )

# Rotation used to not overlap values
plt.xticks(rotation=90)
plt.show()



'''
```



'''

'''

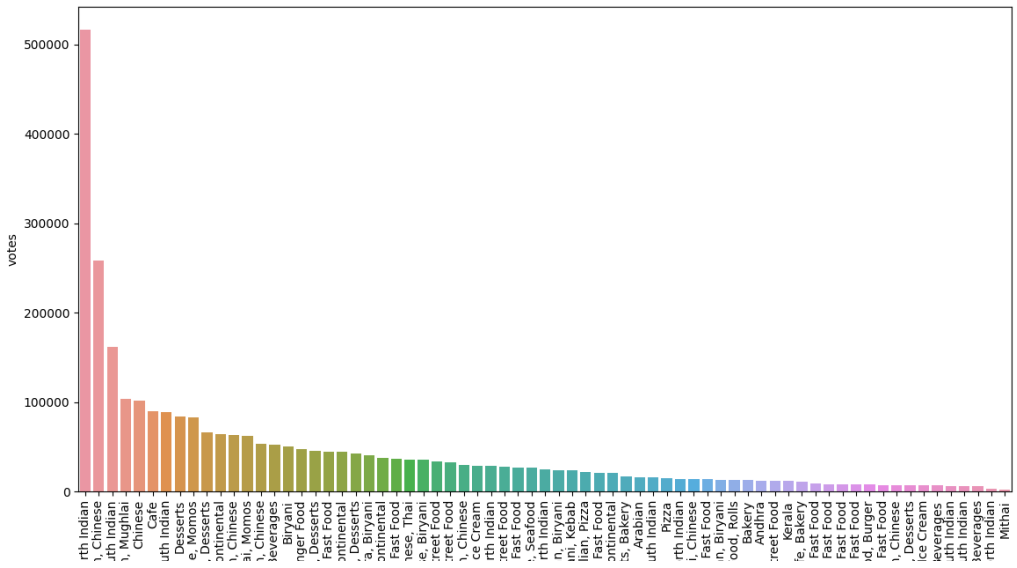
Observations :

1) Here in the graph Votes are in 10^6 format.
2) 2214083 is the highest number of the vote given and is from the "Koramangala 5th Block" location.
3) And least number of votes are from the "Banaswadi" location

'''

```
print(zomato_df.head())

# Checking which cuisines got how many votes
# creating a data frame
cuisines_votes_df = zomato_df[['votes','cuisines']]
print(cuisines_votes_df)

# Droping Duplicates
cuisines_votes_df.drop_duplicates()

# Combining cuisines and votes
```

```
cuisines_votes_df2 = cuisines_votes_df.groupby(['cuisines'])
['votes'].sum()
print(cuisines_votes_df2)


# Converting it into data frame
cuisines_votes_df2 = cuisines_votes_df2.to_frame()
# Converting it into data frame
print(cuisines_votes_df2.head())

# Sorting the data
cuisines_votes_df2=
cuisines_votes_df2.sort_values('votes',ascending=False)
print(cuisines_votes_df2.head())

# Removing first row i.e. "less_num_cuisines" because it will get biased
visualization
cuisines_votes_df2= cuisines_votes_df2.iloc[1: , :]
print(cuisines_votes_df2.head())

# Plortting Data
plt.figure(figsize=(30,8))
sns.barplot(x=cuisines_votes_df2.index,y=cuisines_votes_df2['votes'])
plt.xticks(rotation=90)
plt.show()

'''
```



'''
'''
Observations :

1) From the above bar plot we see that the restaurants severing "North Indian” cuisine have the highest vote i.e. 516310
2) Followed by "North Indian, Chinese" restaurants gets 258225 votes
3) Restaurants serving "mithai" have the lowest voting count.

'''