Exploratory Data Analysis

In [1]:
```
# Importing Libraries:
import numpy as np  #NumPy is a general-purpose array-processing package.
import pandas as pd  #It contains high-level data structures and manipulation tools
import matplotlib.pyplot as plt  #It is a Plotting Library.
import re
import seaborn as sns  #Seaborn is a Python data visualization library based on matp
from sklearn.linear_model import LinearRegression  #Linear Regression is a regressio
from sklearn.ensemble import RandomForestRegressor, ExtraTreesRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split #Splitting of Dataset.
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import r2_score
from sklearn import neighbors
import warnings
warnings.filterwarnings("ignore")
```

Read data

In [2]:
```
#load the data by dropping Unnamed: 0, index columns
data=pd.read_csv("cleaned_data.csv").drop(["Unnamed: 0","index"],axis=1)
```

In [3]:
```
#filling null values in review list
data["reviews_list"].fillna("no reviews",inplace=True)
```

In [4]:
```
#removing null values of names
data.dropna(inplace=True,axis=0)
```

In [5]:
```
#let us check first 5 data instances
data.head(3)
```

Out[5]:

	name	online_order	book_table	rate	votes	location	rest_type	
0	jalsa	yes	yes	4.1	775	banashankari	casual_dining	pastalunch_buffetm
1	spice elephant	yes	no	4.1	787	banashankari	casual_dining	momoslunch_buffe
2	san churro cafe	yes	no	3.8	918	banashankari	cafecasual_dining	churros cannellon

In [40]:
```
import dataframe_image as dfi

dfi.export (data.head() ,'filename.png')
```

In [41]:
```
#shape of the data
print("Number of data instaces that this dataset contains : ",data.shape[0])
print("Number of columns(including target variable) that this dataset contains : ",d
```

```
Number of data instaces that this dataset contains :  51006
Number of columns(including target variable) that this dataset contains :  14
```

In [42]:
```
# columns of dataset:
data.columns
```

Out[42]:
```
Index(['name', 'online_order', 'book_table', 'rate', 'votes', 'location',
       'rest_type', 'dish_liked', 'cuisines', 'approx_cost(for two people)',
       'reviews_list', 'menu_item', 'listed_in(type)', 'listed_in(city)'],
      dtype='object')
```

In [43]:
```
# information about data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51006 entries, 0 to 51147
Data columns (total 14 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   name                         51006 non-null  object 
 1   online_order                 51006 non-null  object 
 2   book_table                   51006 non-null  object 
 3   rate                         51006 non-null  float64
 4   votes                        51006 non-null  int64  
 5   location                     51006 non-null  object 
 6   rest_type                    51006 non-null  object 
 7   dish_liked                   51006 non-null  object 
 8   cuisines                     51006 non-null  object 
 9   approx_cost(for two people)  51006 non-null  float64
 10  reviews_list                 51006 non-null  object 
 11  menu_item                    51006 non-null  object 
 12  listed_in(type)              51006 non-null  object 
 13  listed_in(city)              51006 non-null  object 
dtypes: float64(2), int64(1), object(11)
memory usage: 5.8+ MB
```

In [44]:
```
# data type:
data.dtypes
```

Out[44]:
```
name                            object
online_order                    object
book_table                      object
rate                           float64
votes                            int64
location                        object
rest_type                       object
dish_liked                      object
cuisines                        object
approx_cost(for two people)    float64
reviews_list                    object
menu_item                       object
listed_in(type)                 object
listed_in(city)                 object
dtype: object
```

Analysis of target variable (rating)

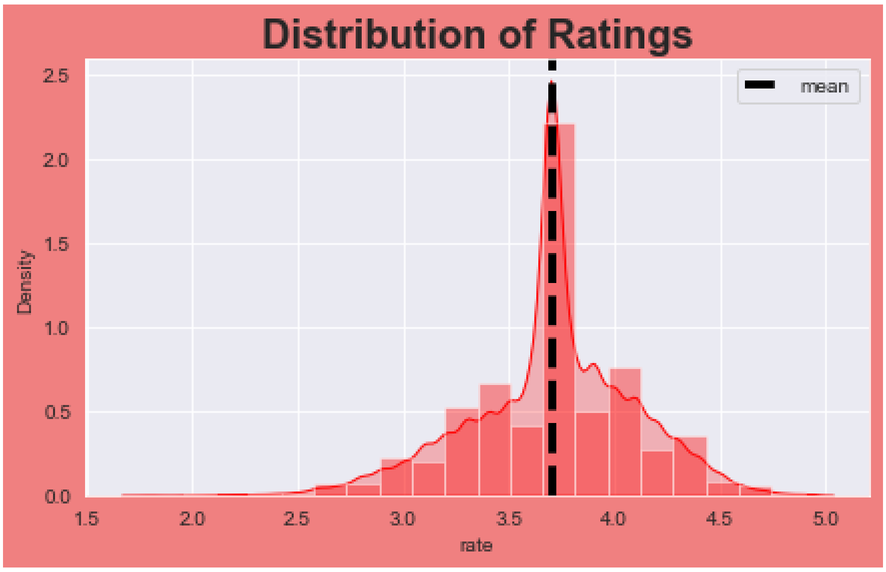
In [45]:
```
print('Restaurents on there unique ratings')
```

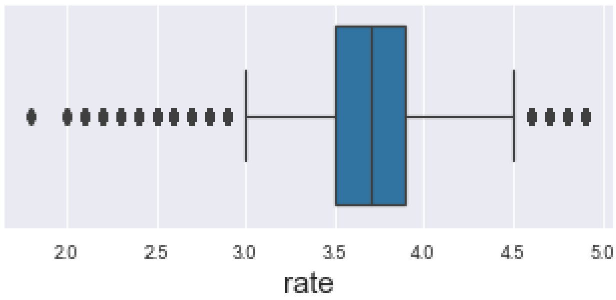
```
data.rate.unique()
```

Restaurents on there unique ratings

Out[45]:
```
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])
```

In [46]:
```
# Distribution of Ratings of restaurants in Bengalore.
fig = plt.figure(figsize=(7,4))
fig.patch.set_facecolor('lightcoral')

sns.set_style('darkgrid')
sns.distplot(data['rate'], bins = 20,  color= 'red',kde_kws={"shade": True});
plt.axvline(x= data.rate.mean(),ls='--',color='black',linewidth=4,label="mean")
plt.title("Distribution of Ratings",fontweight='bold',fontsize=20);
plt.legend(["mean"],prop={"size":10});

sns.set_context("paper",font_scale=1,rc={"font.size": 15,"axes.titlesize": 15,"axes.
b=sns.catplot(data=data,kind='count',x='rate',order=data['rate'].value_counts().inde
plt.title("Number of restuarants for each rating")
b.set_xticklabels(rotation=90)
plt.show()

fig = plt.figure(figsize=(12,7))
ax6 = fig.add_subplot(3,2,6)
sns.boxplot(data['rate'],ax=ax6)
plt.show()
```





In [47]:
```
print('First Quantile of rate distribution is {} '.format(np.quantile(data['rate'],
print('Second Quantile of rate distribution is {} '.format(np.quantile(data['rate'],
print('Third Quantile of rate distribution is {} '.format(np.quantile(data['rate'],
print('Forth Quantile of rate distribution is {} '.format(np.quantile(data['rate'],
print('Average Rating is {} '.format(np.round(data['rate'].mean(),1)))
```

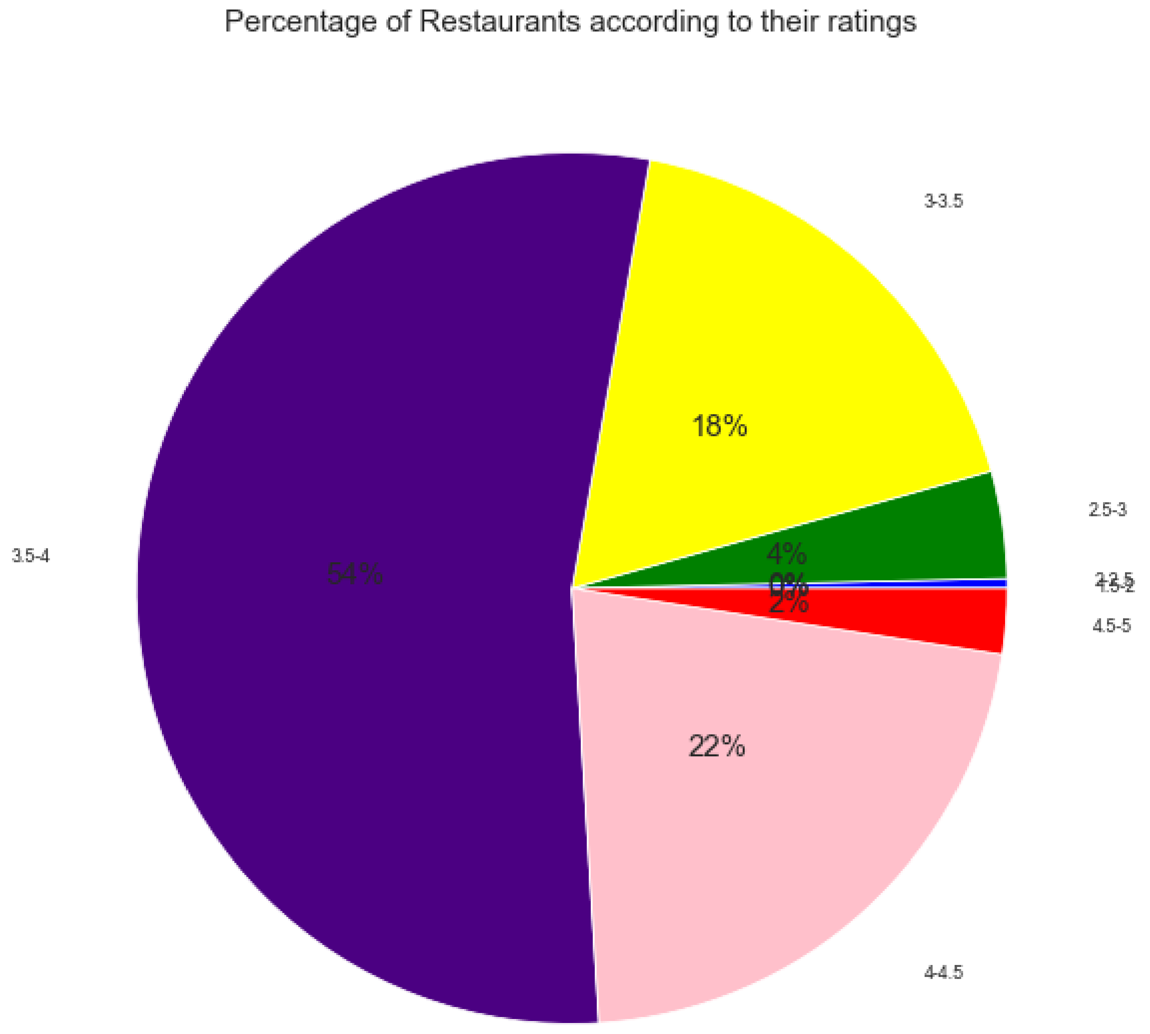
```
First Quantile of rate distribution is 3.5
Second Quantile of rate distribution is 3.7
Third Quantile of rate distribution is 3.9
Forth Quantile of rate distribution is 4.9
Average Rating is 3.7
```

Maximum restaurants have ratings between 3 and 4. Restaurants with rating higher than 4.5 are very rare. 3.7 is the most common rating, i.e. most Bangaloreans have above-average dining experiences when they go out. There are very few ratings between 2 to 2.5 and 4.5 to 5, and hardly any under 2. 50% of the rate distribution lies between 3.4 and 4.0 with an average rating of 3.7.

Rating of a restaurant play major role in success. Nearly everyone checks out the rating before even planing to go out.

To run a successful restaurant business above avaerage zomato rating is a must. Maximum of the restaurants are pretty NEW. Apart from the recently opened restaurants, most of the Restaurants received 3.9 rating, followed by 3.7 and 3.8. Only a few restaurants have 4.8 or 4.9 rating. Let's see which are these restaurants.

In [48]:
```
slices=[((data.rate>=1.5) & (data.rate<2)).sum(),
        ((data.rate>=2) & (data.rate<2.5)).sum(),
        ((data.rate>=2.5) & (data.rate<3)).sum(),
        ((data.rate>=3.0) & (data.rate<3.5)).sum(),
        ((data.rate>=3.5) & (data.rate<4)).sum(),
        ((data.rate>=4) & (data.rate<4.5)).sum(),
        ((data.rate>=4.5) & (data.rate<5)).sum()
       ]
labels=['1.5-2','2-2.5','2.5-3','3-3.5','3.5-4','4-4.5','4.5-5']
colors = ['Red','blue','Green','yellow','indigo','pink']
plt.pie(slices,colors=colors, labels=labels, autopct='%1.0f%%', pctdistance=.5, labe
fig = plt.gcf()
plt.title("Percentage of Restaurants according to their ratings", bbox={'facecolor':

fig.set_size_inches(10,10)
plt.show()
```

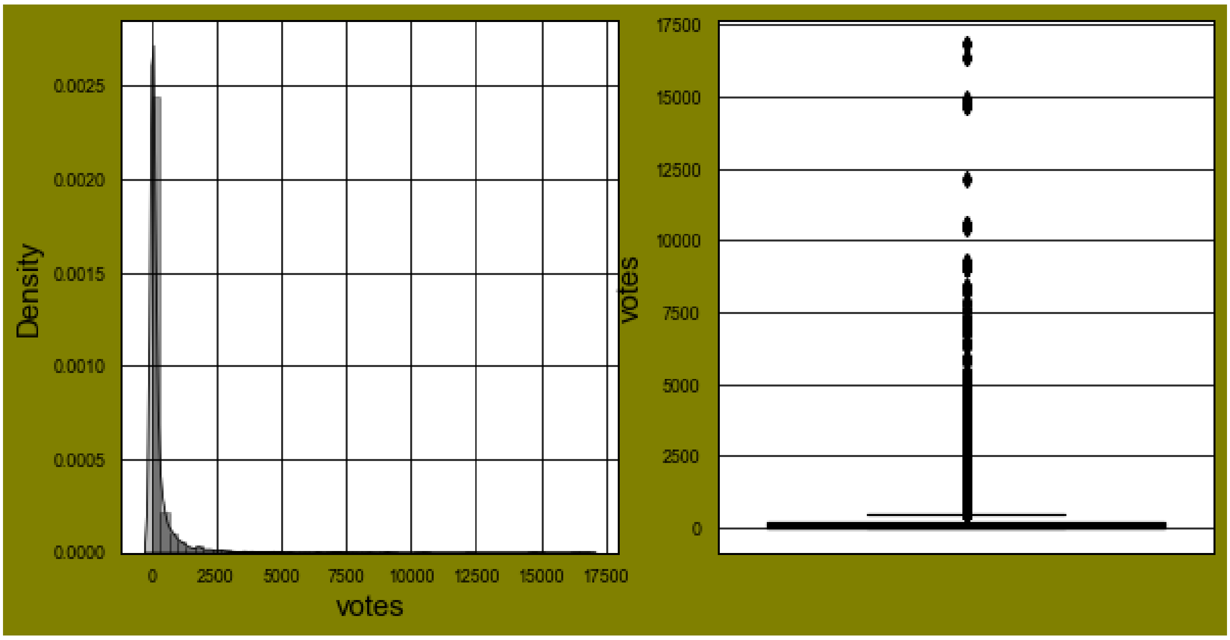


Analysis of each features

Votes

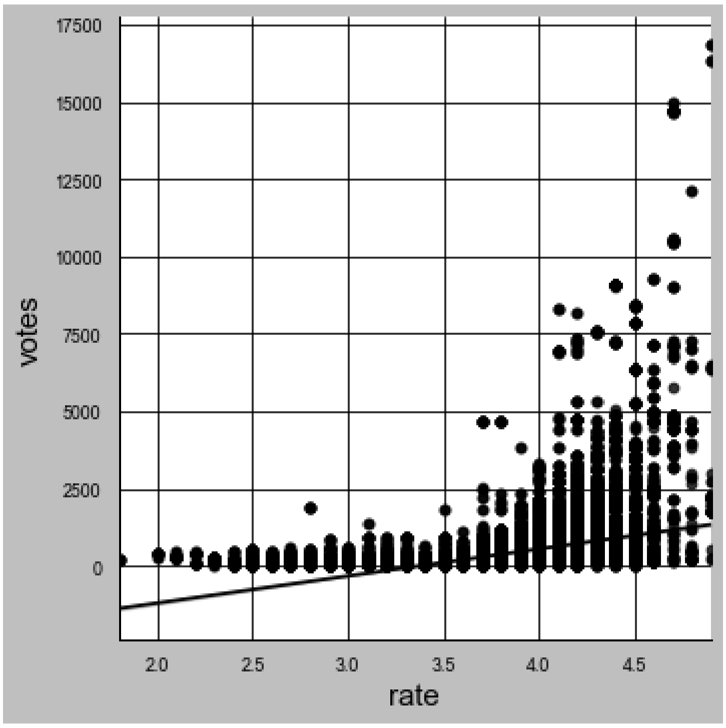
In [49]:
```
# Lets look at distribution of Continues variables
fig = plt.figure(figsize=(10,5))
fig.patch.set_facecolor('olive')
plt.style.use('grayscale')
```

```
plt.subplot(121)
sns.distplot(data['votes'],kde_kws={"shade": True})
plt.subplot(122)
sns.boxplot(y=data['votes']);
```



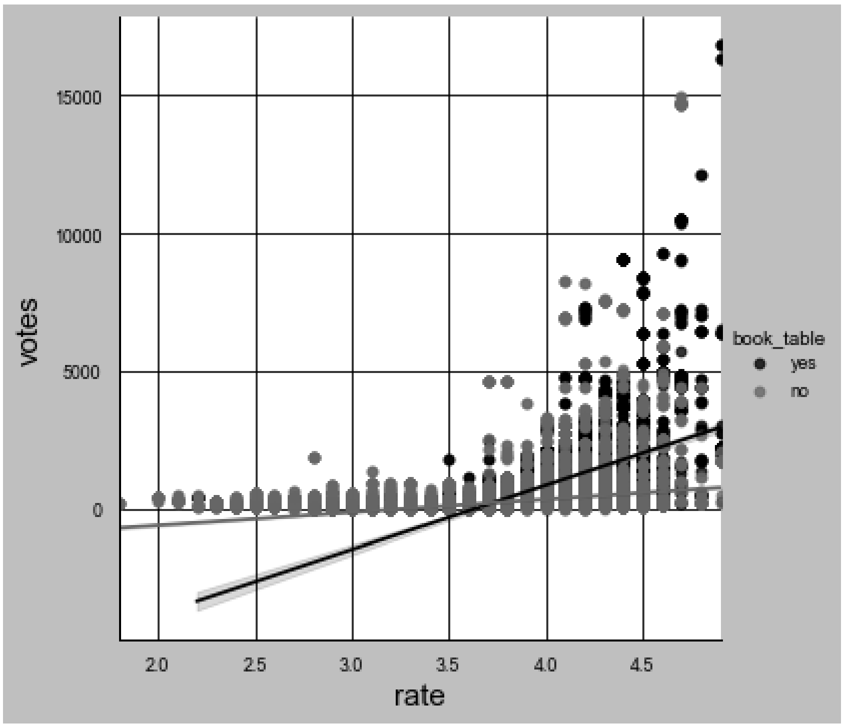
There are very less number of restaurens which has more number of votes and density of votes is very peak at lower values of votes

In [50]:
```
#Linear Relationship between rate and votes shown below:
sns.lmplot(x="rate",y="votes", data=data);
```



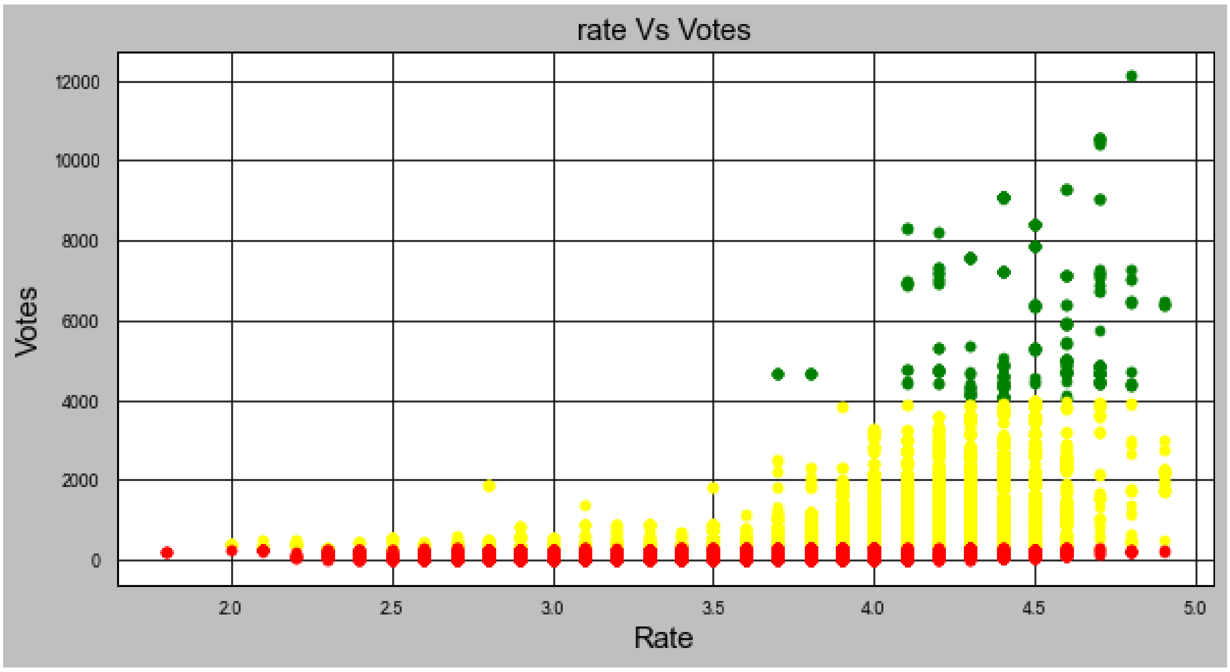
In [ ]:

In [51]:
```
sns.lmplot(x="rate",y="votes", hue="book_table",data=data);
```



In [52]:
```
plt.figure(figsize=(10,5))
df3=data[(data.votes>=4000)&(data.votes<12500)]
plt.scatter(df3.rate,df3.votes,color="green")
df2=data[(data.votes>=data.votes.mean())&(data.votes<4000)]
plt.scatter(df2.rate,df2.votes,color="yellow")
df1=data[data.votes<data.votes.mean()]
plt.scatter(df1.rate,df1.votes,color="red")
plt.xlabel('Rate')
plt.ylabel('Votes')
plt.title('rate Vs Votes')
```

Out[52]: Text(0.5, 1.0, 'rate Vs Votes')



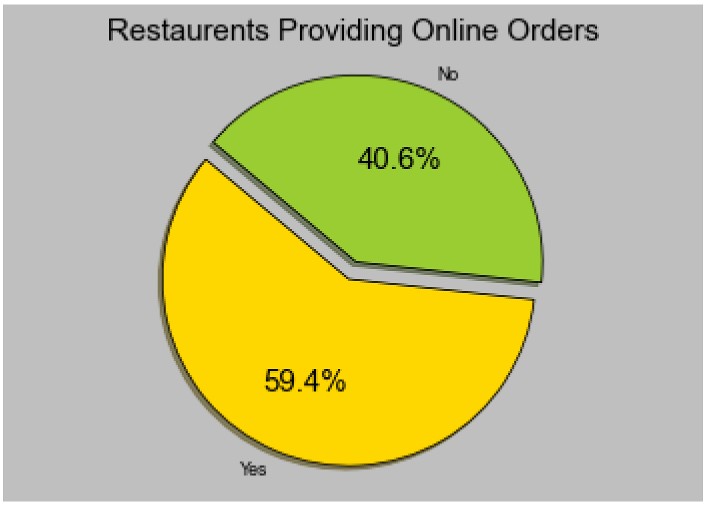
Concidering the No. of votes as popularity of the restaurent. Restaurennts with lesser votes are having lower Ratings. Higher the no. of votes the higher is the potential probability of a company to get higher ratings. Rating may depend on many other unexplored factors.

Analysing online_order facility

In [53]:
```
online_order = (data.online_order == 'yes').sum()
print('number of restaurents with online delivery:',online_order)
online_order = (data.online_order == 'no').sum()
print('number of restaurents without online delivery:',online_order)
```

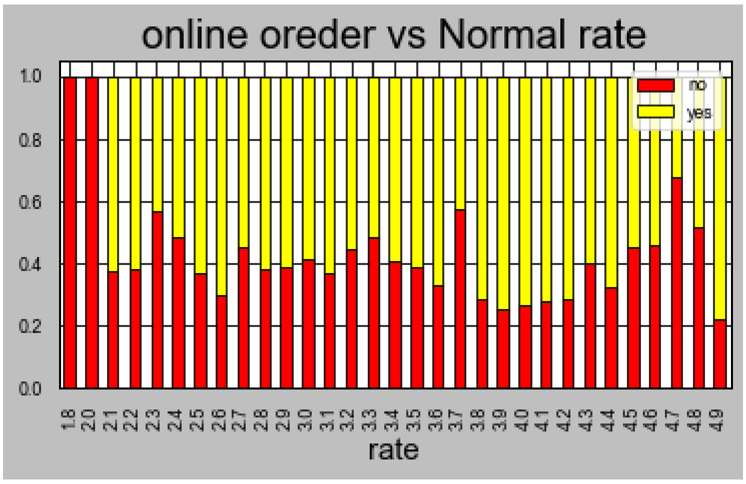
```
number of restaurents with online delivery: 30273
number of restaurents without online delivery: 20733
```

In [54]:
```
#Distribution of online_order
x=data.groupby("online_order")["votes"].count()
labels = 'Yes', 'No'
sizes = [x.yes, x.no]
colors = ['gold', 'yellowgreen']
explode = (0.1, 0,)
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)
plt.title("Restaurents Providing Online Orders")
plt.axis('equal')
plt.show()
```

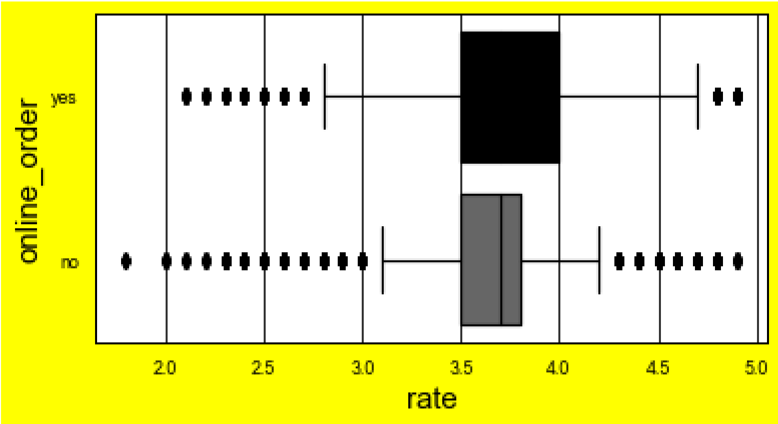


Here 59% restaurents accept online order and 41% restaurents not accept the online order.

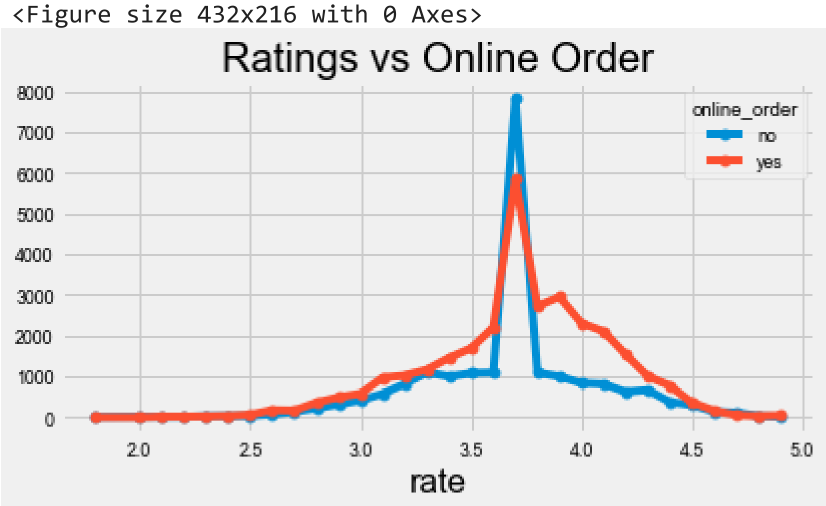
In [55]:
```
#relation between online order option and rating of the restaurant?
plt.rcParams['figure.figsize'] = (6,3)
Y = pd.crosstab(data['rate'], data['online_order'])
Y.div(Y.sum(1).astype(float), axis = 0).plot(kind = 'bar', stacked = True,color=['re
plt.title('online oreder vs Normal rate', fontweight = 30, fontsize = 20)
plt.legend(loc="upper right")
plt.show()

# Rating v/s Online Order #multivariate analysis
fig = plt.figure(figsize=(6,3))
ax1 = fig.add_subplot(1,1,1)
sns.boxplot(x=data['rate'],y=data['online_order'])

#Comparing Ratings Vs Online Orders
plt.figure(figsize=(6,3))
fig.patch.set_facecolor('yellow')
plt.style.use('fivethirtyeight')

pd.crosstab(data.rate,data.online_order).plot(kind='line',marker='o',figsize=(6,3))
plt.title("Ratings vs Online Order")
```



Out[55]: Text(0.5, 1.0, 'Ratings vs Online Order')



<Figure size 432x216 with 0 Axes>



We can observe from the above plot that those restaurants which offer online order has a higher median rating as compared to those restaurants that don't.

As IQR for restaurants offering online order is much less than that of restaurants not offering online order, we can say that restaurants offering online order has better ratings in general.

It makes sense also because Zomato offers home delivery for online orders also, so more people will give rating for online_order restaurants on their platform.

Restaurants are more likely to receive a higher rating if it offers online order option

Restaurants which provide online order facility seem to have better rating than the restaurants which don't

In [56]:
```
#groupby using rate and analyse online_order using this
```

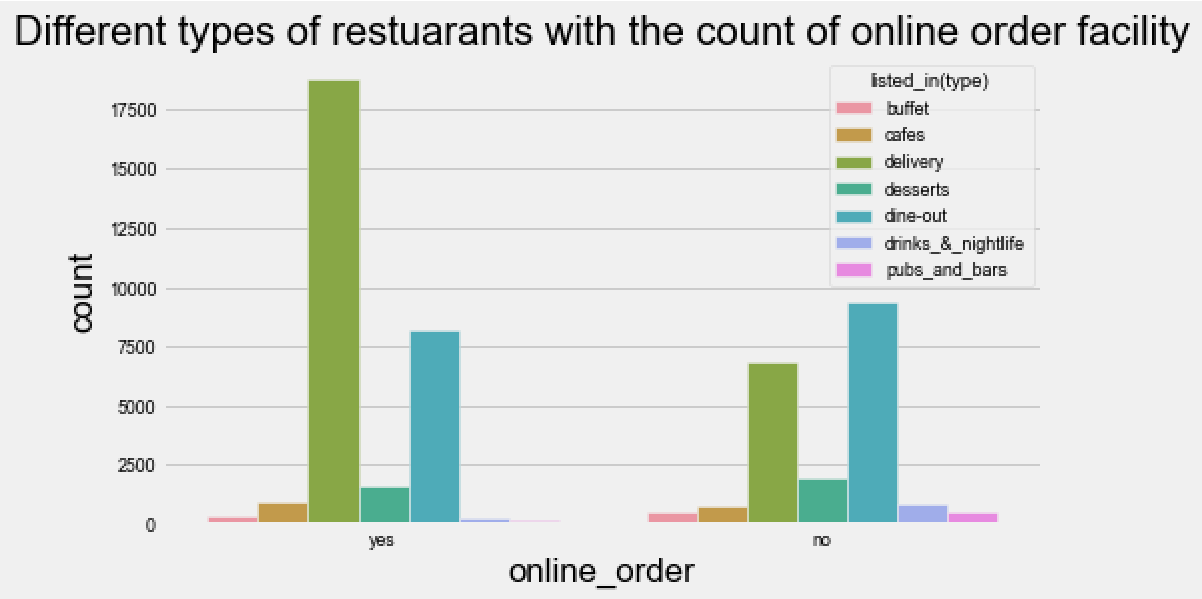
```
rate_online = data.groupby("rate")["online_order"].value_counts().unstack()
```

In [57]:
```
rate_online.head()
```

Out[57]:

online_order	no	yes
rate		
1.8	5.0	NaN
2.0	11.0	NaN
2.1	9.0	15.0
2.2	10.0	16.0
2.3	29.0	22.0

In [58]:
```
sns.countplot(x=data['online_order'],hue=data['listed_in(type)'],)
fig = plt.gcf()  # here gcf means 'GET THE CURRENT FIGURE'
fig.set_size_inches(7,4)
plt.title('Different types of restuarants with the count of online order facility')
plt.show()
```



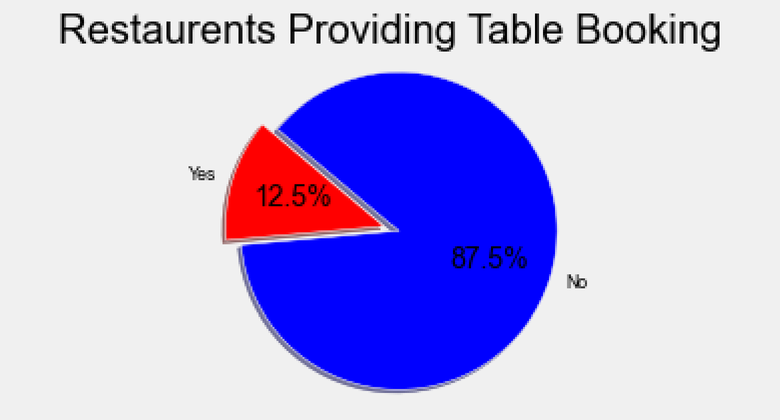
Analysing book_table facility

In [59]:
```
book_table=(data.book_table == 'yes').sum()
print('number of restaurents with table book facility:',book_table)
book_table=(data.book_table == 'no').sum()
print('number of restaurents without table book facility:',book_table)
```

```
number of restaurents with table book facility: 6391
number of restaurents without table book facility: 44615
```

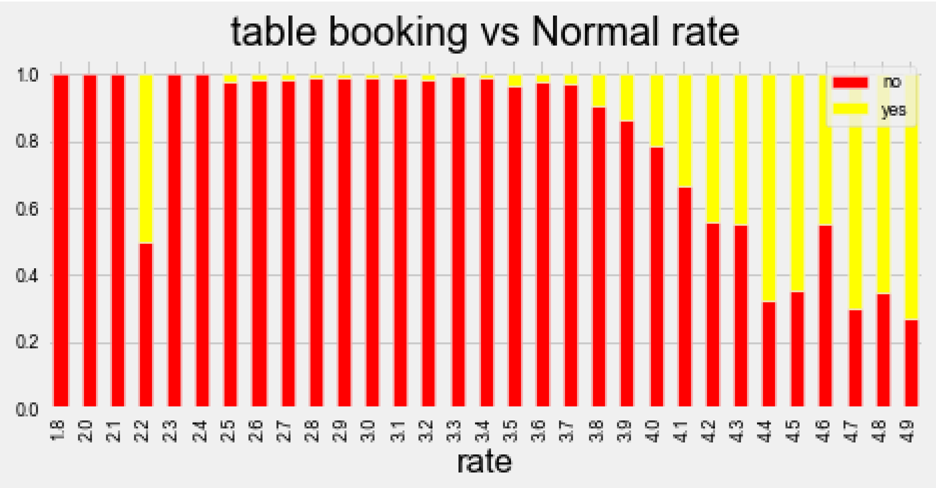
In [60]:
```
x=data.groupby("book_table")["votes"].count()
labels = 'Yes', 'No'
sizes = [x.yes, x.no]
colors = ['red', 'blue']
explode = (0.1, 0,)
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)
plt.title("Restaurents Providing Table Booking")
```

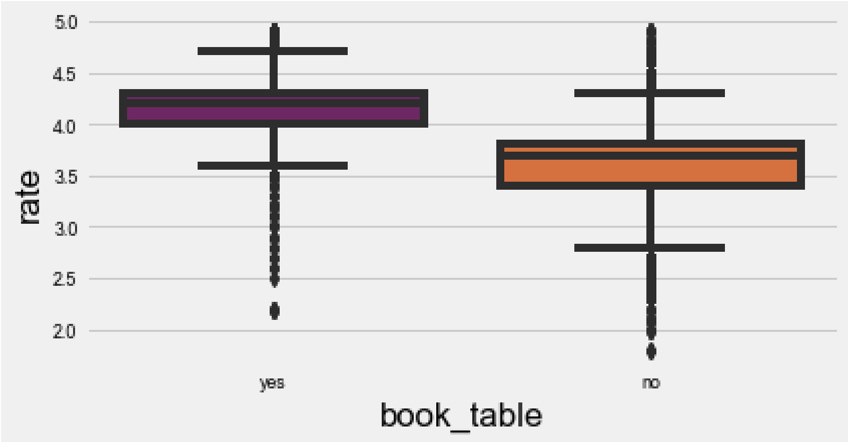
```
plt.axis('equal')
plt.show()
```



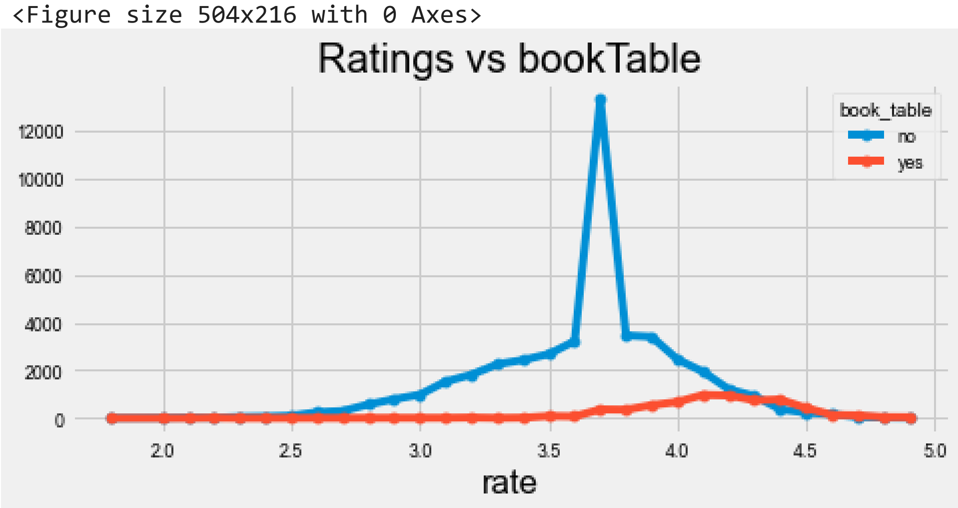
Here 13% restaurents provide table booking and 87% restaurents not provide table booking facility.

In [61]:
```
# relation between table booking option and rating of the restaurant
plt.rcParams['figure.figsize'] = (7,3)
Y = pd.crosstab(data['rate'], data['book_table'])
Y.div(Y.sum(1).astype(float), axis = 0).plot(kind = 'bar', stacked = True,color=['re
plt.title('table booking vs Normal rate', fontweight = 30, fontsize = 20)
plt.legend(loc="upper right")
plt.show()

# Rating v/s book table #multivariate analysis
plt.figure(figsize = (6,3))
sns.boxplot(x = 'book_table', y = 'rate', data = data, palette = 'inferno')

plt.figure(figsize=(7,3))
fig.patch.set_facecolor('yellow')
plt.style.use('fivethirtyeight')
pd.crosstab(data.rate,data.book_table).plot(kind='line',marker='o',figsize=(7,3));
plt.title("Ratings vs bookTable");
```





<Figure size 504x216 with 0 Axes>



Restaurants are more likely to receive a higher rating if it offers table book option Eventhough there are some outliers for the book_table class, we can see that the lower whisker of '1''s boxplots which represents the minimum rating of the restaurants that book table in advance, is greater than the 50th percentile value or the median of the ratings of the restaurants that don't book table in advance.

Some restaurants that don't book table in advance also have ratings close to 5. The IQR for '1' boxplot is quite small which represents small variation of the ratings around median. Therefore, if the restaurants offer to book table in advance, more ratings are given.

More clear here that if your restaturat has not the book table service you still have the opportinity to have a similar rate as other restaurant provide this service. Most of the restaurant has not this service

while at rate around 4.2 and above we notice higher number of restaurants at this rate and provide book_table service

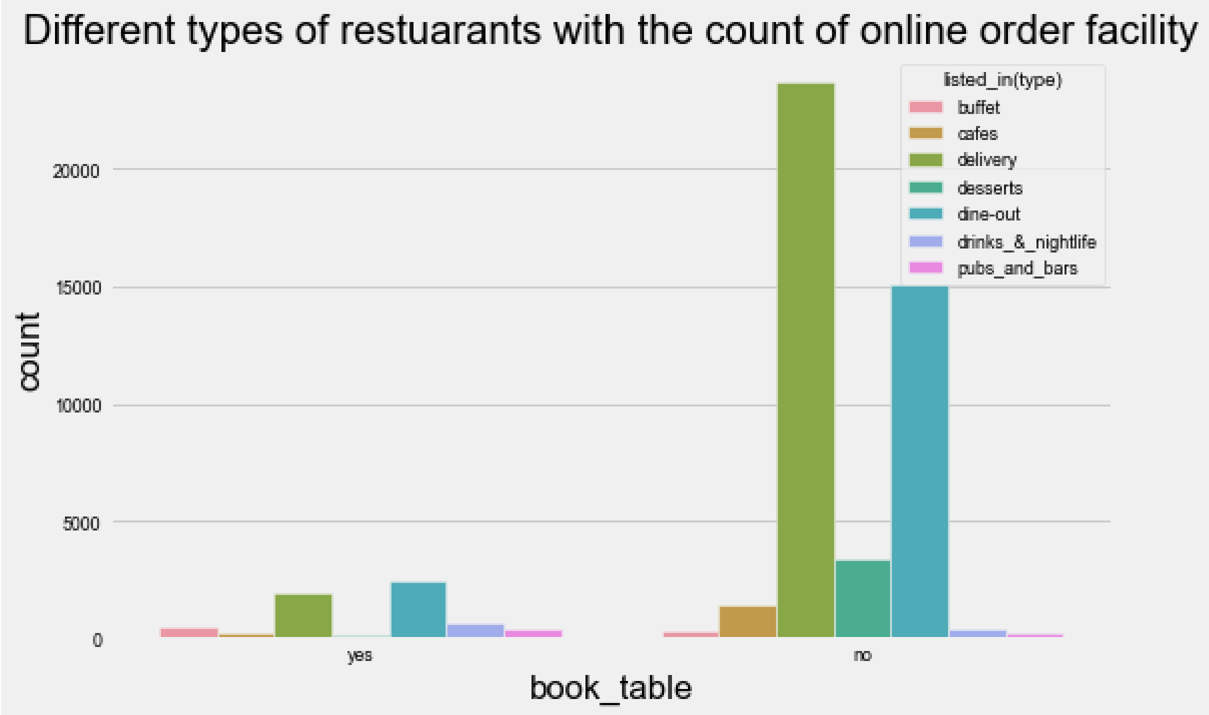
We can see if customer hasn't done online order and table booking then their ratings are highly distributed between ratings 3.2- 3.6 then decreases.

If customer has Online ordered and have't booked table then their ratings are better than above case and are highly distributed between ratings 3.2- 4.1 then decreases.

If customer has Online ordered and booked table enen though their percent is less still they have rated above average between 3.7 and 4.5

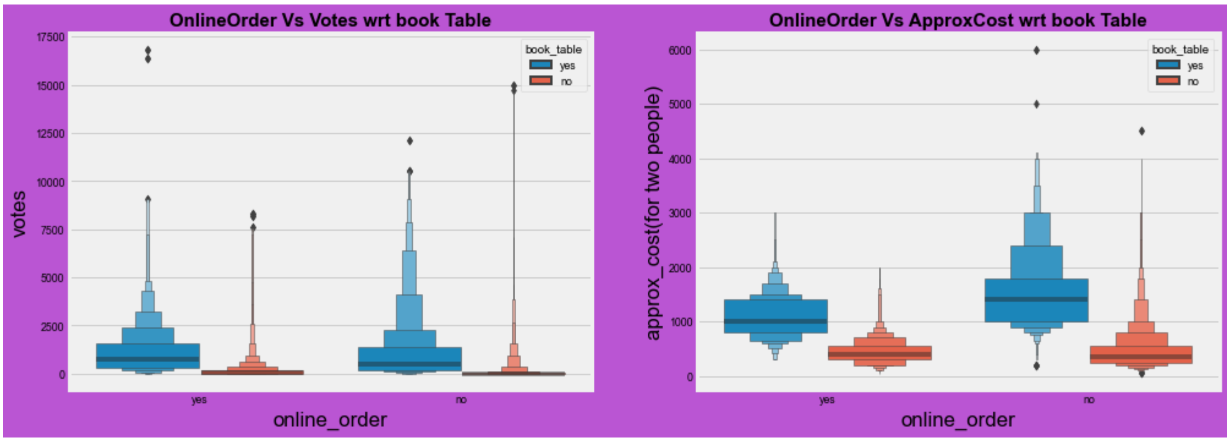
In [62]:
```
sns.countplot(x=data['book_table'],hue=data['listed_in(type)'],)
```

```
fig = plt.gcf()  # here gcf means 'GET THE CURRENT FIGURE'
fig.set_size_inches(8,5)
plt.title('Different types of restuarants with the count of online order facility')
plt.show()
```



OnlineOrder Vs Votes and OnlineOrder Vs ApproxCost wrt book Table

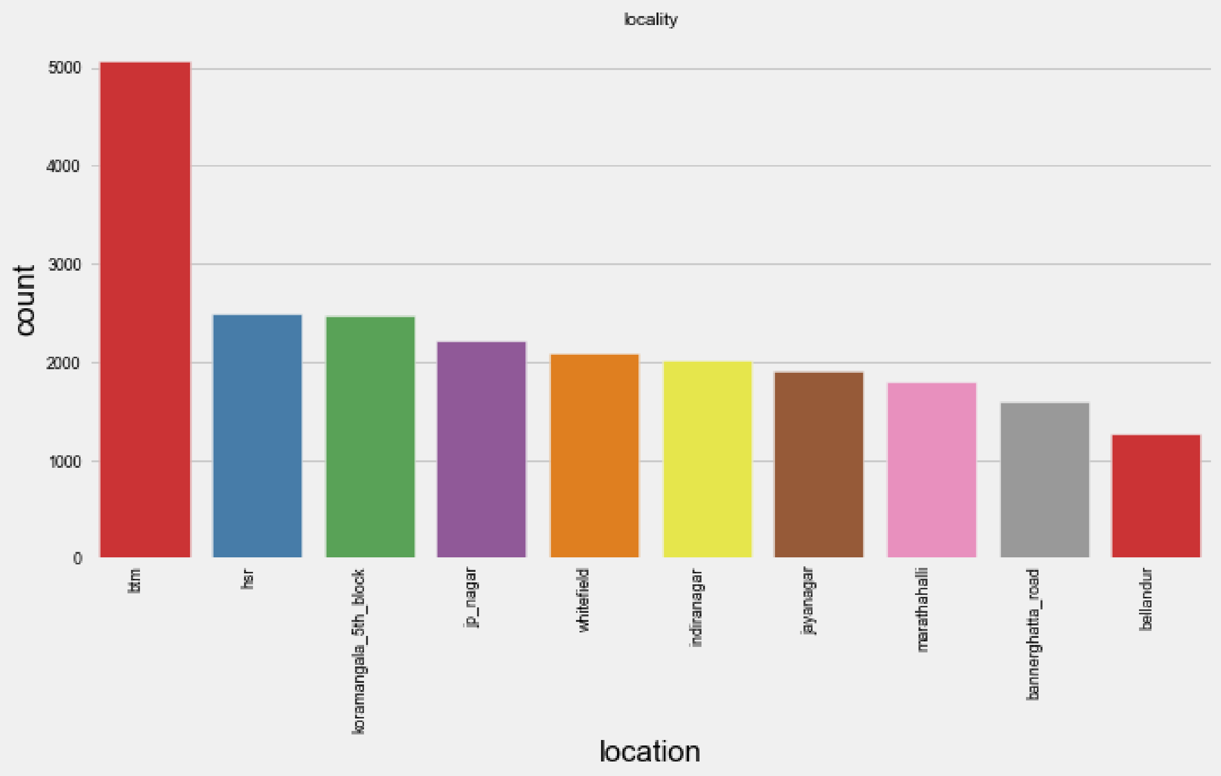
In [63]:
```
#OnlineOrder Vs Votes and OnlineOrder Vs ApproxCost wrt book Table
fig = plt.figure(figsize=(15,5))
fig.patch.set_facecolor('mediumorchid')
plt.style.use('fivethirtyeight')

plt.subplot(121)
sns.boxenplot(data=data,x='online_order',y='votes',hue='book_table');
plt.title("OnlineOrder Vs Votes wrt book Table",fontweight='bold',fontsize=15);

plt.subplot(122)
sns.boxenplot(data=data,x='online_order',y='approx_cost(for two people)',hue='book_t
plt.title("OnlineOrder Vs ApproxCost wrt book Table",fontweight='bold',fontsize=15);
```



Restaurants accepting online orders get more umber of votes. Median number of votes are different in both categoies. The cost is significantly less when restaurants accept orders online

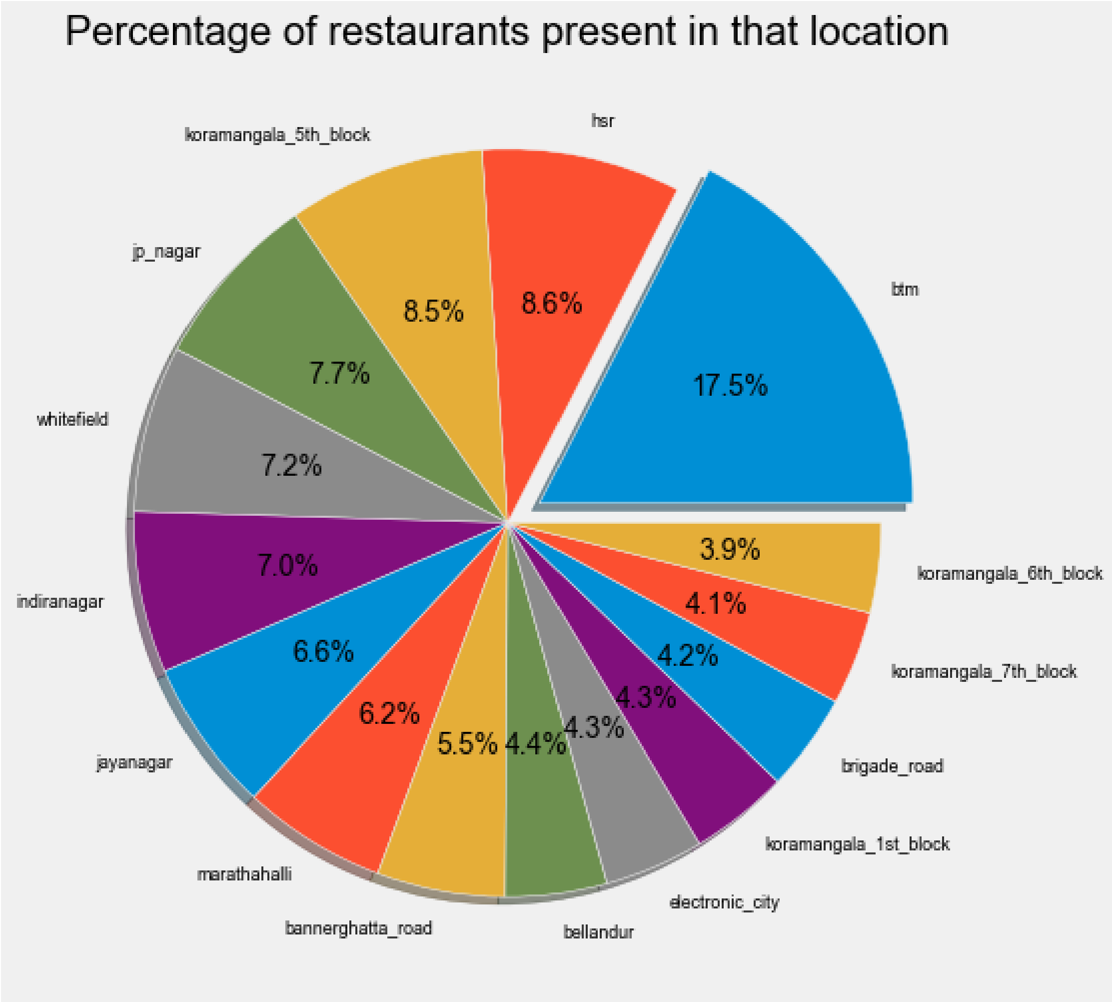
Analysing Location:

In [64]:
```
# Lets look at distribution of Location Variable
g = sns.countplot(x="location",data=data, palette = "Set1",order = data['location'].
g.set_xticklabels(g.get_xticklabels(), rotation=90, ha="right")
g
plt.title('locality',size = 10)
fig = plt.gcf()
fig.set_size_inches(10,5)
```



Koramangala has been split blockwise or it would be at the top with the others We can see that BTM,HSR and Koranmangala 5th block has the most number of restaurants. BTM dominates the section by having more than 5000 restaurants.

In [65]:
```
plt.figure(figsize=(8,8))
names = data.location.value_counts()[:15].index
values = data.location.value_counts()[:15].values
explode = [0.1,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

plt.pie(values, explode=explode, autopct='%0.1f%%', shadow=True, labels = names)
plt.title("Percentage of restaurants present in that location")
plt.show()
```



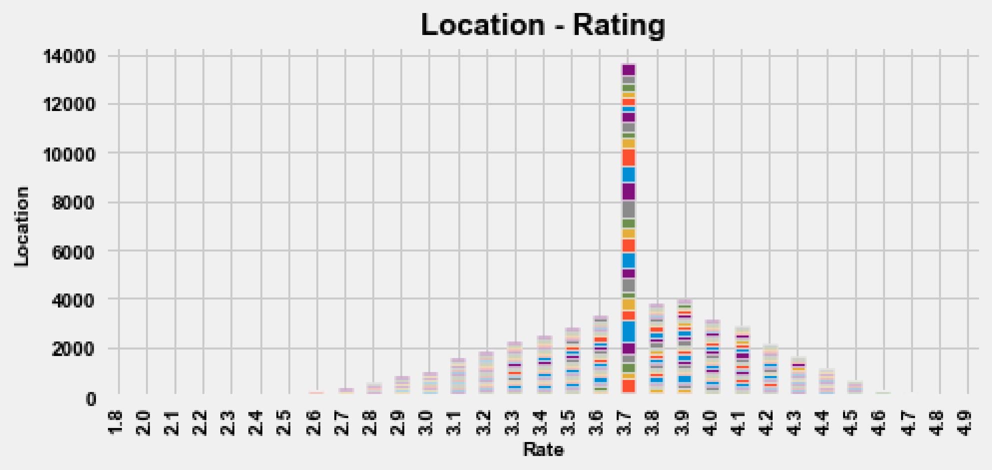
In [ ]:

In [66]:

```
#Location and rating
data.groupby('location')['rate'].mean().sort_values(ascending = False).head(10)
#Location and Rating
loc_plt=pd.crosstab(data['rate'],data['listed_in(city)'])
loc_plt.plot(kind='bar',stacked=True);
plt.title('Location - Rating',fontsize=15,fontweight='bold')
plt.ylabel('Location',fontsize=10,fontweight='bold')
plt.xlabel('Rate',fontsize=10,fontweight='bold')
plt.xticks(fontsize=10,fontweight='bold')
plt.yticks(fontsize=10,fontweight='bold');
plt.legend().remove();
# Top Location in town to get good food.
top_places = data.groupby('location')['rate'].median().sort_values(ascending=False)
pd.DataFrame(top_places)
# Top 5 Locations with the highest ratings
(pd.DataFrame(data.groupby("location")["rate"].mean())).sort_values("rate", ascendin
```

Out[66]:

	rate
location	
lavelle_road	4.106310
st._marks_road	4.017201
koramangala_5th_block	3.985818
koramangala_3rd_block	3.983333
church_street	3.980107



The top two locations with high ratings are also the two most expensive locations (Sankey Road and Lavelle Road) In general we can see that restaurants around the MG Road area are more expensive

In [67]:

```
df1 = data.groupby(['location','online_order'])['name'].count()

# converting df1 data to csv

df1.to_csv('location_online.csv')

# reading the csv file

df1 = pd.read_csv('location_online.csv')

# conversting that into pivot table

df1 = pd.pivot_table(df1, values=None, index=['location'], columns=['online_order'],

df1
```

Out[67]:

	name	
online_order	no	yes
location		
banashankari	395	507
banaswadi	302	343
bannerghatta_road	683	924
basavanagudi	243	441
basaveshwara_nagar	87	100
...	...	...
west_bangalore	4	2
whitefield	972	1115
wilson_garden	112	134
yelahanka	0	5
yeshwantpur	26	93

93 rows × 2 columns

In [68]:
```
df2 = data.groupby(['location','book_table'])['name'].count()

# converting df1 data to csv

df2.to_csv('location_book_table.csv')

# reading the csv file

df2 = pd.read_csv('location_book_table.csv')

# conversting that into pivot table

df2 = pd.pivot_table(df2, values=None, index=['location'], columns=['book_table'], f

df2
```

Out[68]:

	name	
book_table	**no**	**yes**
location		
banashankari	840	62
banaswadi	637	8
bannerghatta_road	1508	99
basavanagudi	668	16
basaveshwara_nagar	169	18
...	...	...
west_bangalore	6	0
whitefield	1844	243
wilson_garden	241	5
yelahanka	5	0
yeshwantpur	117	2

93 rows × 2 columns

In [69]:
```
df3 = data.groupby(['location','rest_type'])['name'].count()

# converting df1 data to csv

df3.to_csv('location_type.csv')

# reading the csv file

df3 = pd.read_csv('location_type.csv')

# conversting that into pivot table

df3 = pd.pivot_table(df3, values=None, index=['location'], columns=['rest_type'], fi

df3
```

Out[69]:

rest_type	bakery	bakery cafe	bakery kiosk	bakerybeverage_shop	bakerydessert_parlor	bakeryfoo

rest_type	bakery	bakery cafe	bakery kiosk	bakerybeverage_shop	bakerydessert_parlor	bakeryfoo
location						
banashankari	20	0	0	0	2	
banaswadi	27	0	0	0	0	
bannerghatta_road	53	0	0	0	0	
basavanagudi	35	0	0	0	0	
basaveshwara_nagar	2	0	0	0	1	
...	...	...	...	...	...	
west_bangalore	0	0	0	0	0	
whitefield	58	14	0	0	0	
wilson_garden	6	0	0	0	0	
yelahanka	0	0	0	0	0	
yeshwantpur	3	2	0	0	0	

93 rows × 92 columns

In [70]:
```
df4 = data[['votes','location']]
df4.drop_duplicates()
df5 = df4.groupby(['location'])['votes'].sum()
df5 = df5.to_frame()
df5 = df5.sort_values('votes', ascending = False)
df5.head()
```

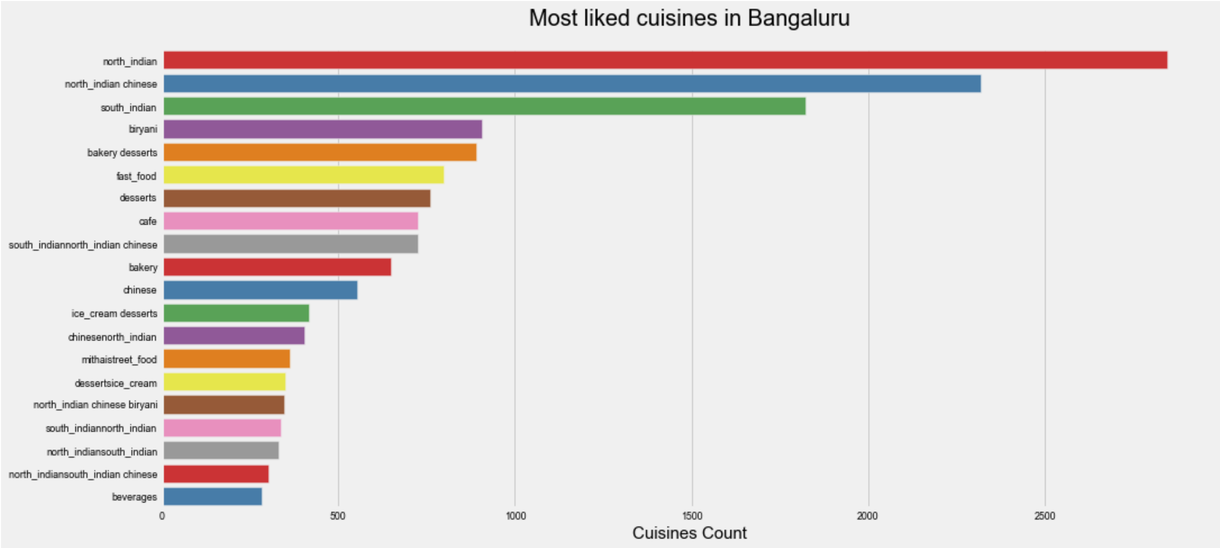
Out[70]:

	votes
location	
koramangala_5th_block	2214533
indiranagar	1164314
koramangala_4th_block	685104
church_street	594157
jp_nagar	586522

Analysing cuisines :

In [71]:
```
plt.figure(figsize=(15,7))
chains=data['cuisines'].value_counts()[:20]
sns.barplot(x=chains,y=chains.index,palette='Set1')
plt.title("Most liked cuisines in Bangaluru",size=20,pad=20)
plt.xlabel("Cuisines Count",size=15)
```

Out[71]:
```
Text(0.5, 0, 'Cuisines Count')
```



We have cuisines such as North Indian, Chinese, Continental, Caffe, Fast food and several others. After looking at the graph you can see that we have North Indian is the Most liked cuisine.

In [72]:
```
# Identifying the top 10 cuisines in Bangalore?
pd.DataFrame(data.groupby(["cuisines"])["cuisines"].agg(['count']).sort_values("coun
```

Out[72]:

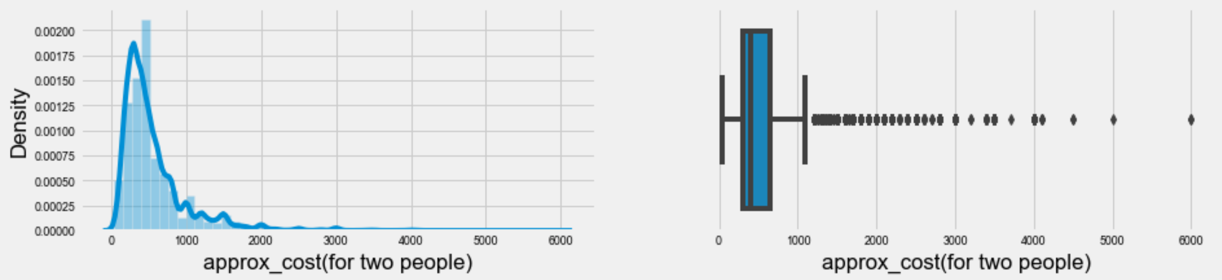
cuisines	count
north_indian	2846
north_indian chinese	2318
south_indian	1822
biryani	906
bakery desserts	891
fast_food	797
desserts	760
cafe	726
south_indiannorth_indian chinese	724
bakery	649

In [ ]:

Analysing approx_cost:

In [ ]:

In [73]:
```
#approximate cost distrubution:
fig = plt.figure(figsize=(14,10))
ax3 = fig.add_subplot(3,2,3)
ax4 = fig.add_subplot(3,2,4)
sns.distplot(data['approx_cost(for two people)'],ax=ax3)
sns.boxplot(data['approx_cost(for two people)'],ax=ax4)
plt.show()
```



This is a graph for the 'Approximate cost of 2 people' for dining in a restaurant. We can see that the distribution if left skewed. This means almost 90percent of restaurants serve food for budget less than 1000

In [ ]:

In [74]:

```
# Top 5 most expensive and cheap locations (cost = cost for two)
df=(pd.DataFrame(data.groupby("location")["approx_cost(for two people)"].mean())).so
df
```

Out[74]:

location	approx_cost(for two people)
sankey_road	2505.555556
race_course_road	1309.352518
lavelle_road	1307.934990
mg_road	1155.704698
residency_road	966.320475

In [75]:

```
import dataframe_image as dfi

dfi.export (df ,'df.png')
```

In [76]:

```
dff=(pd.DataFrame(data.groupby("location")["approx_cost(for two people)"].mean())).s
dff
```

Out[76]:

location	approx_cost(for two people)
peenya	300.000000
city_market	302.426230
yelahanka	310.000000
cv_raman_nagar	311.111111
ejipura	320.506912

In [77]:

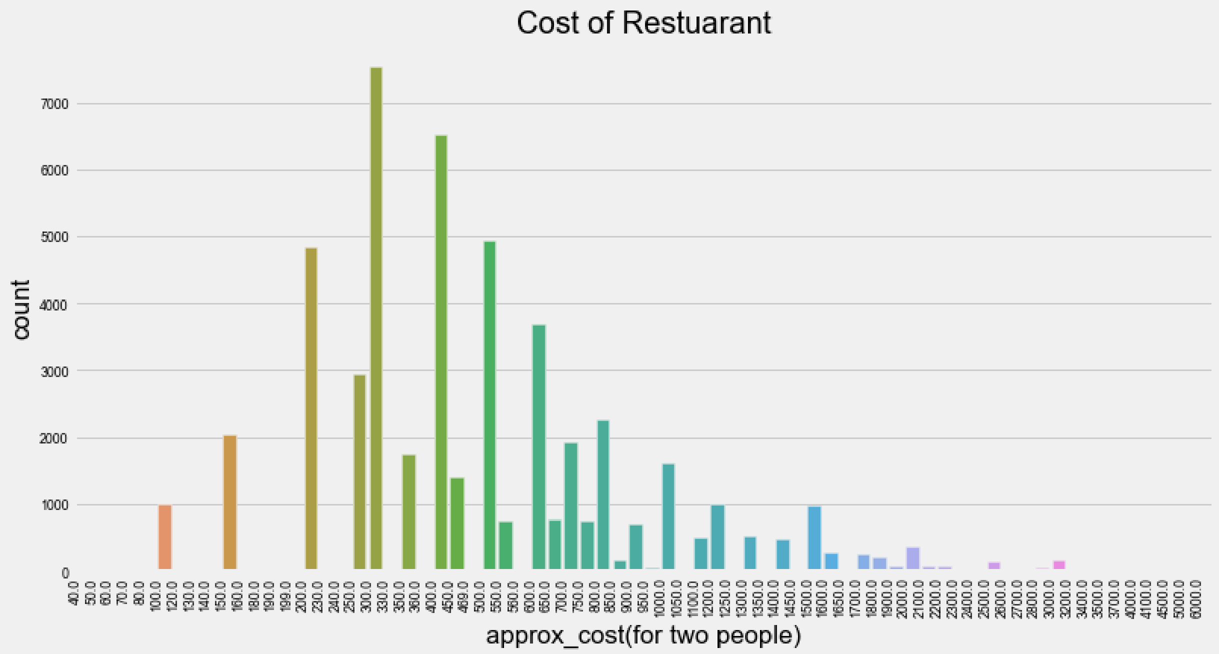
```
import dataframe_image as dfi

dfi.export (dff,'dff.png')
```

In [78]:

```
#Cost of Restuarant
sns.countplot(data['approx_cost(for two people)'])
sns.countplot(data['approx_cost(for two people)']).set_xticklabels(sns.countplot(dat
```

```
fig = plt.gcf()
fig.set_size_inches(12,6)
plt.title('Cost of Restuarant')
```

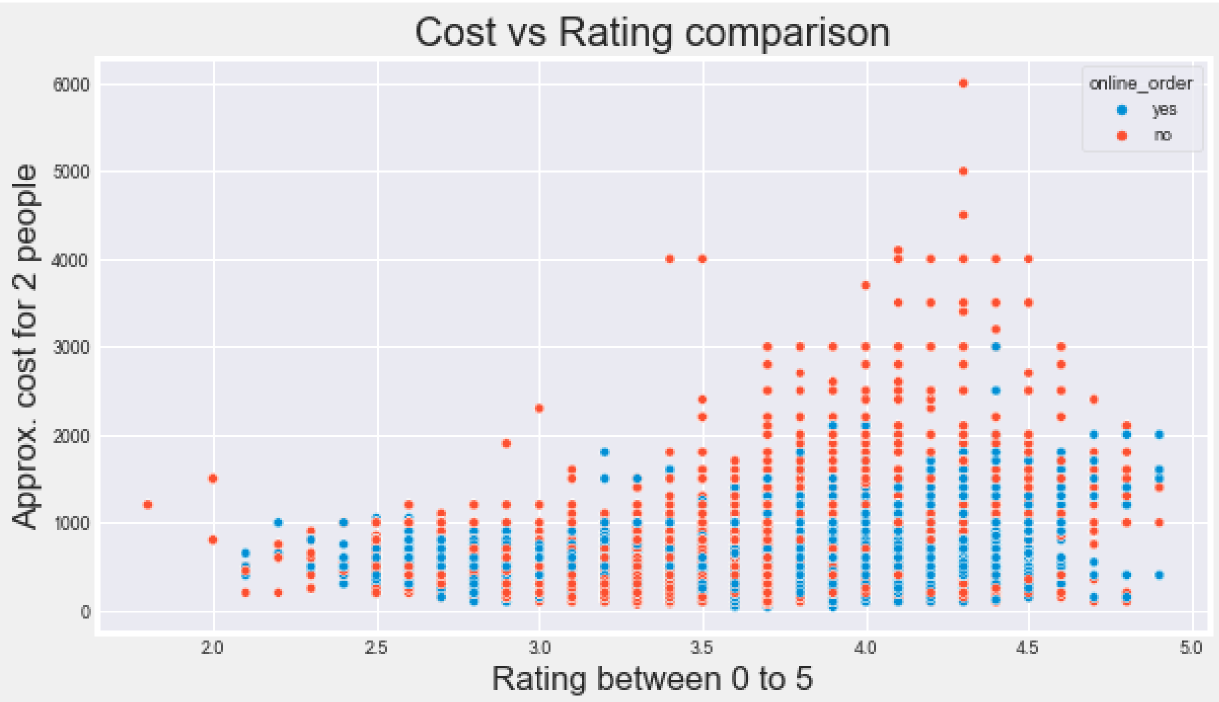
Out[78]: `Text(0.5, 1.0, 'Cost of Restuarant')`



In [ ]:

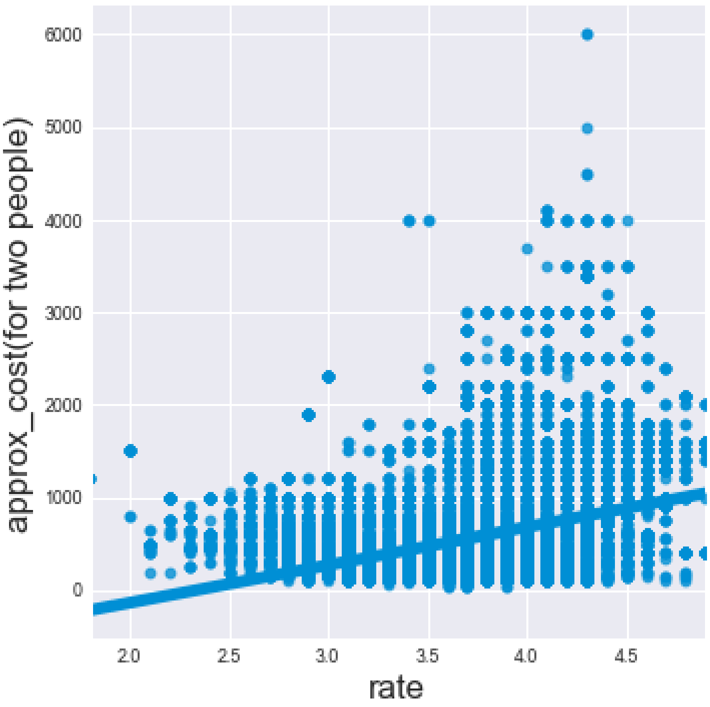
In [79]:
```
plt.figure(figsize=(9,5))
sns.set_style('darkgrid')
sns.scatterplot( x= 'rate', y = 'approx_cost(for two people)', hue= 'online_order',
plt.title('Cost vs Rating comparison')
plt.xlabel('Rating between 0 to 5')
plt.ylabel('Approx. cost for 2 people')
```

Out[79]: `Text(0, 0.5, 'Approx. cost for 2 people')`



In [80]:
```
#Linear Relationship between rates and approx_cost_for_2_people shown below
sns.lmplot(x="rate",y="approx_cost(for two people)", data=data)
```

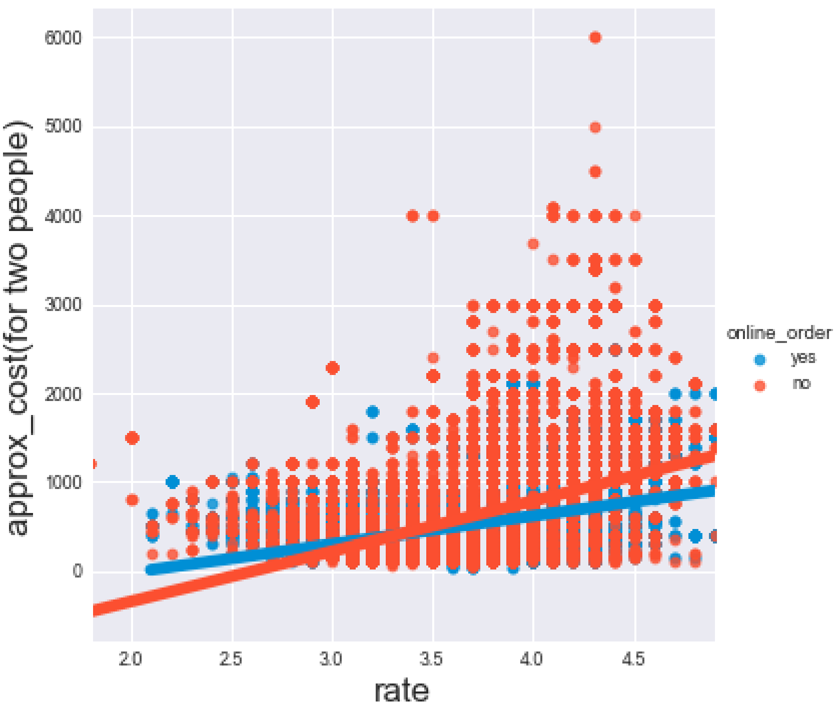
Out[80]: <seaborn.axisgrid.FacetGrid at 0x11b5f57f7f0>



In [ ]:

In [81]:
```
sns.lmplot(x="rate",y="approx_cost(for two people)",hue="online_order", data=data)
```

Out[81]: <seaborn.axisgrid.FacetGrid at 0x11b4e931190>



Analysing Restaurent type:

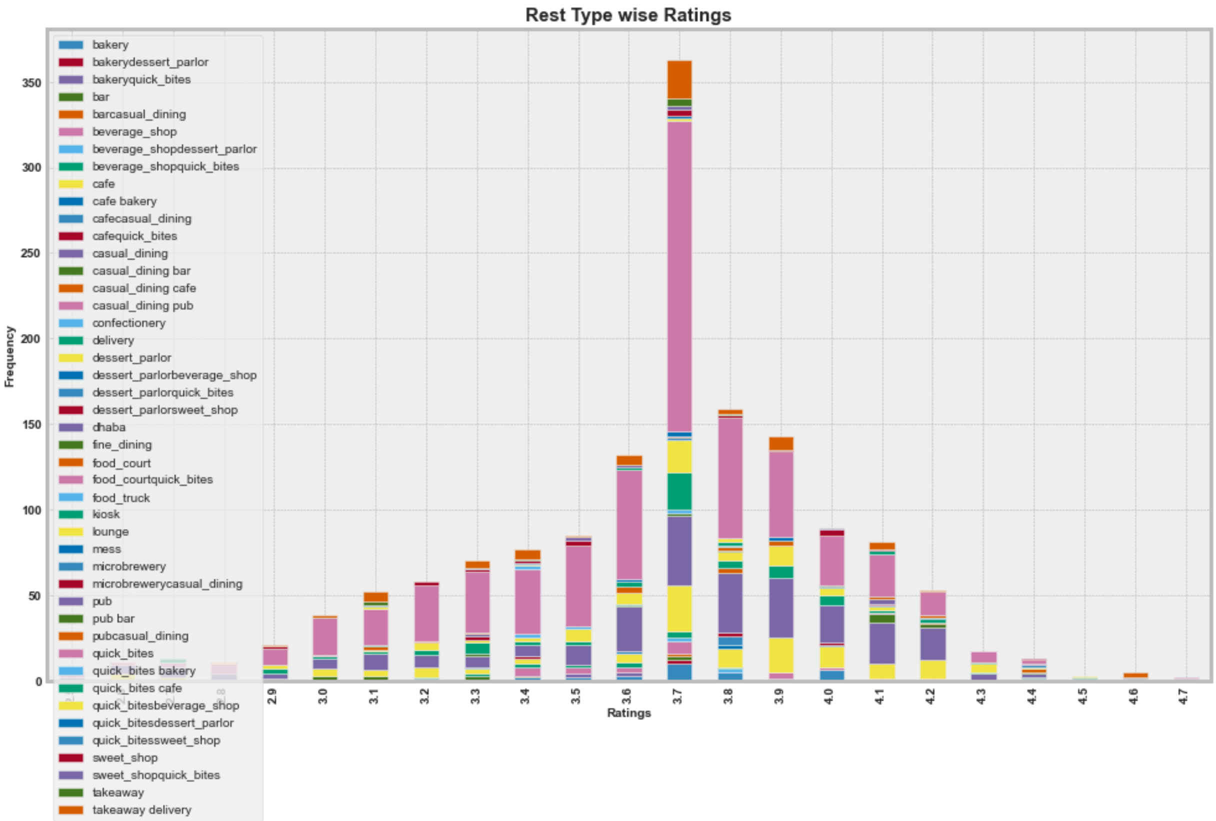
In [82]:
```
RestTypes =data.groupby('rest_type')['name'].count().sort_values(ascending= False).h
RestTypes
```

Out[82]:
```
rest_type
quick_bites        19019
casual_dining      10238
```

```
cafe                   3683
delivery               2564
dessert_parlor         2245
takeaway delivery      2011
Name: name, dtype: int64
```

In [83]:
```
# Rest type and Rating
fig = plt.figure(figsize=(10,5))
fig.patch.set_facecolor('forestgreen')
plt.style.use('bmh')

pd.crosstab(data.rate.head(1500),data.rest_type.head(1500)).plot(kind='bar',stacked=
plt.title('Rest Type wise Ratings',fontsize=15,fontweight='bold')
plt.ylabel('Frequency',fontsize=10,fontweight='bold')
plt.xlabel('Ratings',fontsize=10,fontweight='bold')
plt.xticks(fontsize=10,fontweight='bold')
plt.yticks(fontsize=10,fontweight='bold');
plt.legend(loc = 'upper left',prop={"size":10});

# Top rated restaurant types
top_types = data.groupby('rest_type')['rate'].median().sort_values(ascending=False)
top_types
```

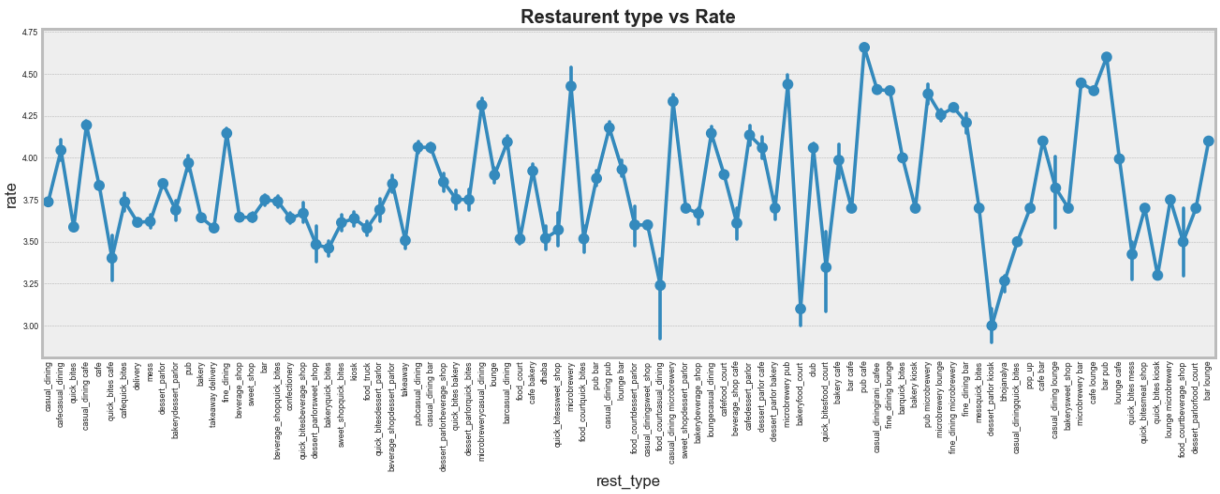
Out[83]:
```
rest_type
pub cafe                    4.7
bar pub                     4.6
microbrewery pub            4.5
casual_diningirani_cafee    4.4
cafe lounge                 4.4
                           ... 
dessert_parlorsweet_shop    3.4
quick_bites kiosk           3.3
bhojanalya                  3.3
bakeryfood_court            3.1
dessert_parlor kiosk        3.0
Name: rate, Length: 92, dtype: float64
<Figure size 720x360 with 0 Axes>
```



We can see Restaurant Type marked with Pink Color are highly distributed for both good and bad reviews.

Restaurants Marked with Purple Color are 2nd largest distibuted with ratings 3.6 - 4.2

In [84]:
```
f,ax=plt.subplots(figsize=(20,6))
g = sns.pointplot(x=data["rest_type"], y=data["rate"], data=data)
g.set_xticklabels(g.get_xticklabels(), rotation=90)
plt.title('Restaurent type vs Rate', weight = 'bold')
plt.show()

fig = plt.figure(figsize=(9,5))
fig.patch.set_facecolor('mediumpurple')
plt.style.use('fivethirtyeight')
rest_type = data.rest_type.value_counts().index[:12].tolist()

rest_typeData = data[data.rest_type.isin(top_12_rest_type)]

pd.crosstab(rest_typeData.rate,rest_typeData.rest_type).plot(kind='line',marker='o',
plt.title("Ratings Vs Top 12 RestType",fontsize='15',fontweight=15);
plt.xlabel("RatingBy 5",fontsize='10',fontweight='bold')
plt.ylabel("Frequency",fontsize='10',fontweight='bold');
```




```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-84-a46295ea1274> in <module>
     10 rest_type = data.rest_type.value_counts().index[:12].tolist()
     11 
---> 12 rest_typeData = data[data.rest_type.isin(top_12_rest_type)]
     13 
     14 pd.crosstab(rest_typeData.rate,rest_typeData.rest_type).plot(kind='line',mar
ker='o',figsize=(12,7));

NameError: name 'top_12_rest_type' is not defined
<Figure size 648x360 with 0 Axes>
```

We can notice Quick Bites got majority of ratings between 3.3 - 3.7 .

Casual Dinings got the 2nd hightest ratings but after 3.5 the ratings get better and gets better ratings than Quick Bites from 4.0 and above. Thus its average rating is more than Quick Bites.

The other Resttypes gets almost same ratinga but Cafe which gets 3rd highest ratings increase and gets better than Quick Bites after 4.2

In [ ]:

In [ ]:
```
all_ratings = []

for name,ratings in tqdm(zip(data['name'],data['reviews_list'])):
    ratings = eval(ratings)
```

```
    for score, doc in ratings:
        if score:
            score = score.strip("Rated").strip()
            doc = doc.strip('RATED')
            trip()
            score = float(score)
            all_ratings.append([name,score, doc])
```

In [ ]:
```
rating_data=data.DataFrame(all_ratings,columns=['name','rating','review'])
rating_data['review']=rating_data['review'].apply(lambda x : re.sub('[^a-zA-Z0-9\s]'
```

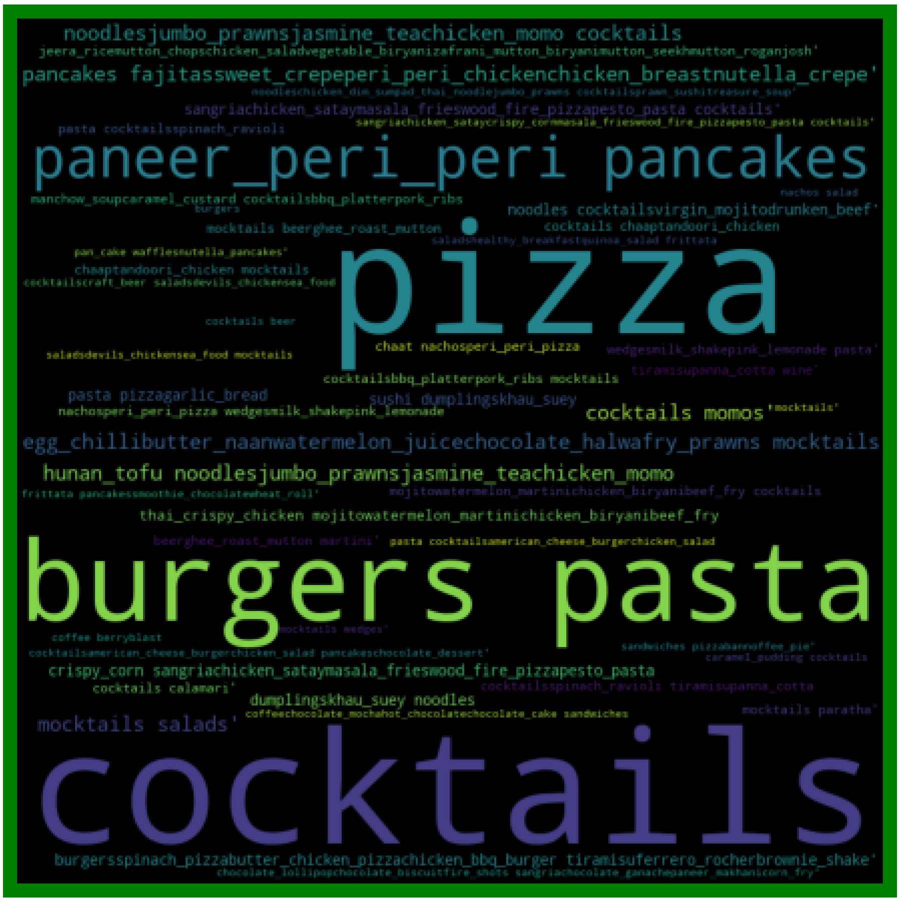
Analysing dish liked:

In [85]:
```
conda install -c conda-forge wordcloud
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

In [86]:
```
topRatedRest = data[data.rate >= 4.5]

topDishes = []
for i in topRatedRest[topRatedRest.dish_liked != 'Unknown']['dish_liked']:
    for j in i.split(', '):
        topDishes.append(j)
```

In [87]:
```
from wordcloud import WordCloud, STOPWORDS

comment_words = ' '
stopwords = set(STOPWORDS)

wordcloud = WordCloud(width = 800, height = 800,
                background_color ='black',
                stopwords = stopwords,
                min_font_size = 10).generate(str(topDishes))

# plot the WordCloud image
plt.figure(figsize = (6,6), facecolor = 'green')
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0) ;
```



In [88]:
```
low_budget = data.groupby(['dish_liked'])['approx_cost(for two people)'].sum().sort_
low_budget = low_budget[low_budget["approx_cost(for two people)"] <= 1500]
```

In [89]:
```
# High budget restaurent
high_budget = data.groupby(['dish_liked'])['approx_cost(for two people)'].sum().sort
high_budget = high_budget[(high_budget["approx_cost(for two people)"] > 3000) & (hig
```

In [90]:
```
print(high_budget["dish_liked"].value_counts()[:10])
```

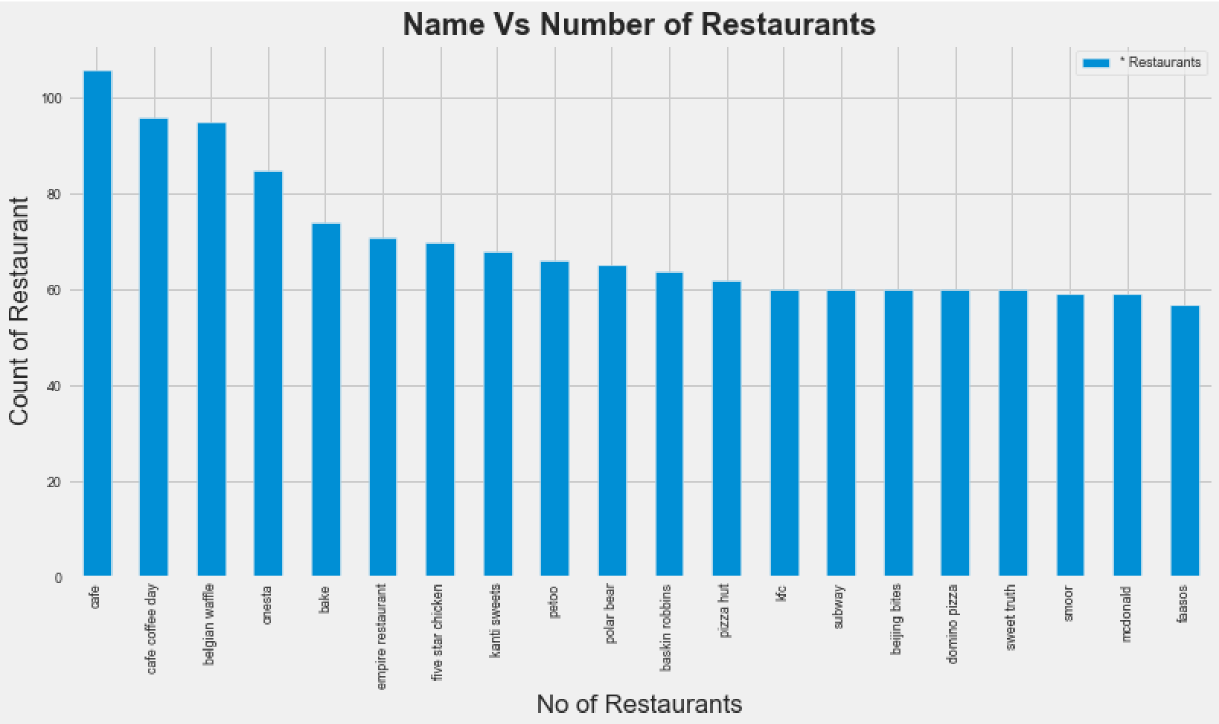
```
biryani haleemjumbo_shawarmakerala_parottabarbeque_chickenmutton_raan_biriyanidum_al
oo    1
cup_cakefruit_gateausponge_cakemango_cakeeggless_cakechocolate_truffle_cake cheeseca
ke    1
ulavacharu biryanibangla_kodirajugari_kodi_pulaogadwal_kodi_pulaocurd_rice fish
1
butter_naanfilter_coffeepav_bhaji vadababycorn_manchurian teapaneer_tikka_masala
1
beerbutter_chickenvegetable_biryani cocktails kulfikeema_pavchur_chur_paratha
1
hyderabadi_biryani
1
pizza nachos pastapotli_biryani mojitodraught_beerlong_island_iced_tea
1
pizza pastabubble_tea brownie pancakescheesy_garlic_breadchocolate_waffles
1
hot_chocolate_fudge
1
manchurian noodlesbasil_chickenchop_sueytriple_schezwandragon_chickenschezwan_rice
1
Name: dish_liked, dtype: int64
```

Analysing name:

In [91]:
```
plt.figure(figsize=(12,6))
ax =data.name.value_counts()[:20].plot(kind='bar')
```

```
ax.legend(['* Restaurants'])
plt.xlabel('No of Restaurants')
plt.ylabel('Count of Restaurant')
plt.title("Name Vs Number of Restaurants", fontsize=20, weight='bold')
```

Out[91]: Text(0.5, 1.0, 'Name Vs Number of Restaurants')



In [92]:
```
data.groupby('name')['votes', 'rate'].max().sort_values(ascending = False, by = 'vot
```

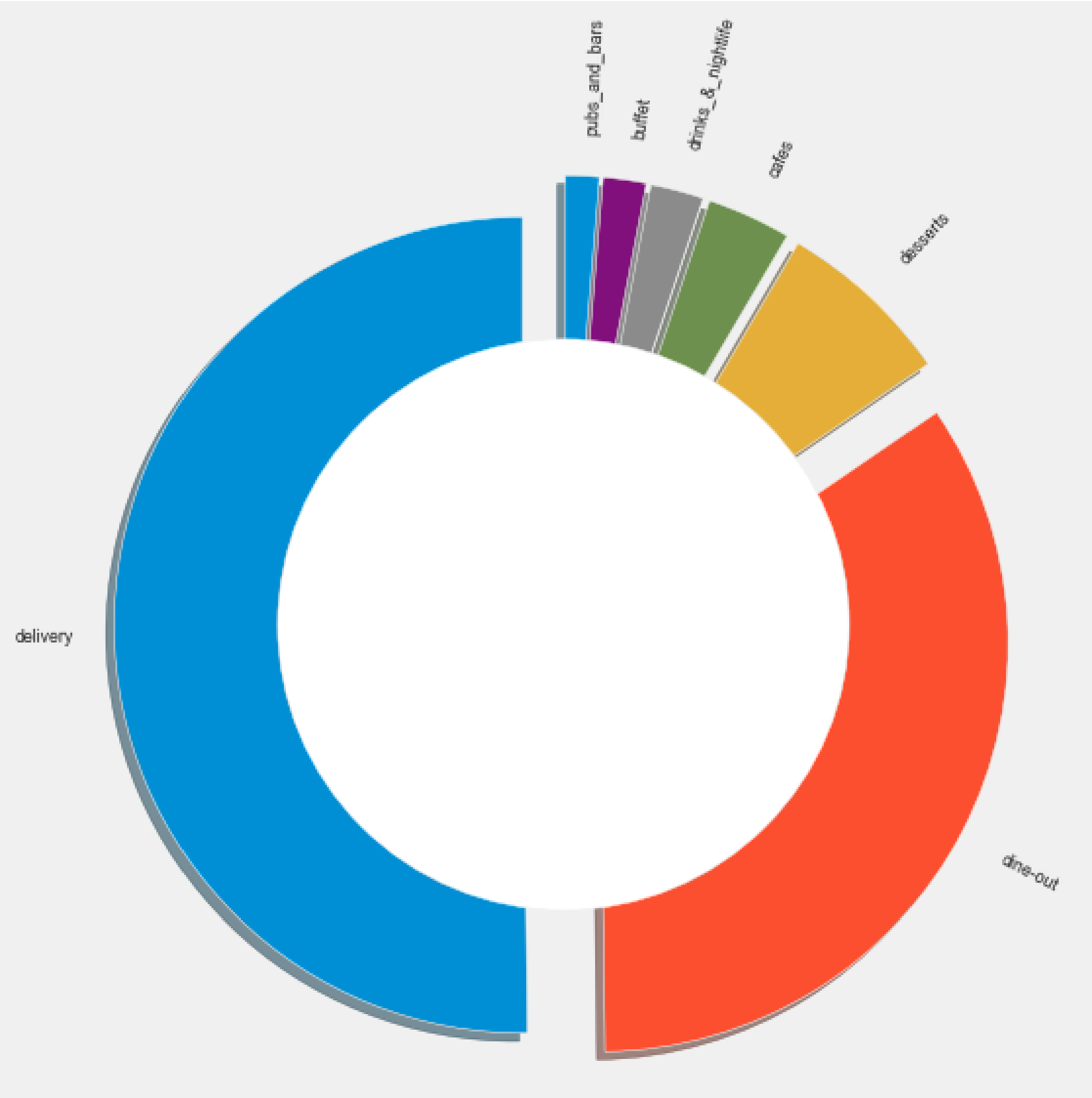
Out[92]:

name	votes	rate
byg brewski brewing company	16832	4.9
toit	14956	4.7
truffles	14726	4.7
absolute barbecues	12121	4.9
black pearl	10550	4.8
big pitcher	9300	4.7
onesta	9085	4.6
arbor brewing company	8419	4.5
empire restaurant	8304	4.4
prost brew pub	7871	4.5
church street social	7584	4.3
hoot	7330	4.2
barbeque nation	7270	4.8
meghana foods	7238	4.5
flechazo	7154	4.9

Above are the 15 restaurants that have got the highest number of user votes. The ratings for these restaurants are also very high as expected. More votes most probably leads to better rating

Analysing listed_in(type):

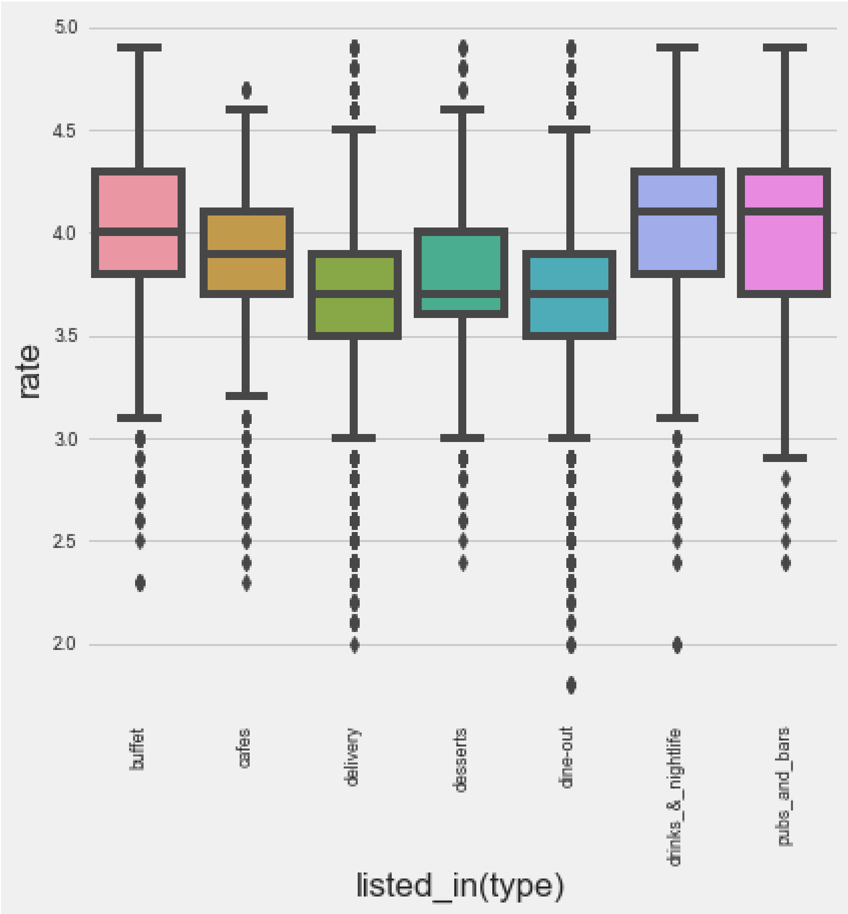
In [93]:
```
labels = data['listed_in(type)'].value_counts().index
sizes = data['listed_in(type)'].value_counts().values
# only "explode" the 2nd slice (i.e. 'Hogs')
explode = (0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)
fig1, ax1 = plt.subplots(figsize = (8, 8))

ax1.pie(sizes, labels = labels,
        shadow = True, startangle = 90, explode = explode, rotatelabels = True)
centre_circle = plt.Circle((0, 0), 0.70,fc = 'white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
# Equal aspect ratio ensures that pie is drawn as a circle
ax1.axis('equal')
plt.tight_layout()
plt.show()
```



Here the two main service types are Delivery and Dine-out

In [94]:
```
plt.figure(figsize = (6, 6))
g = sns.boxplot(x = 'listed_in(type)', y = 'rate', data = data)
```

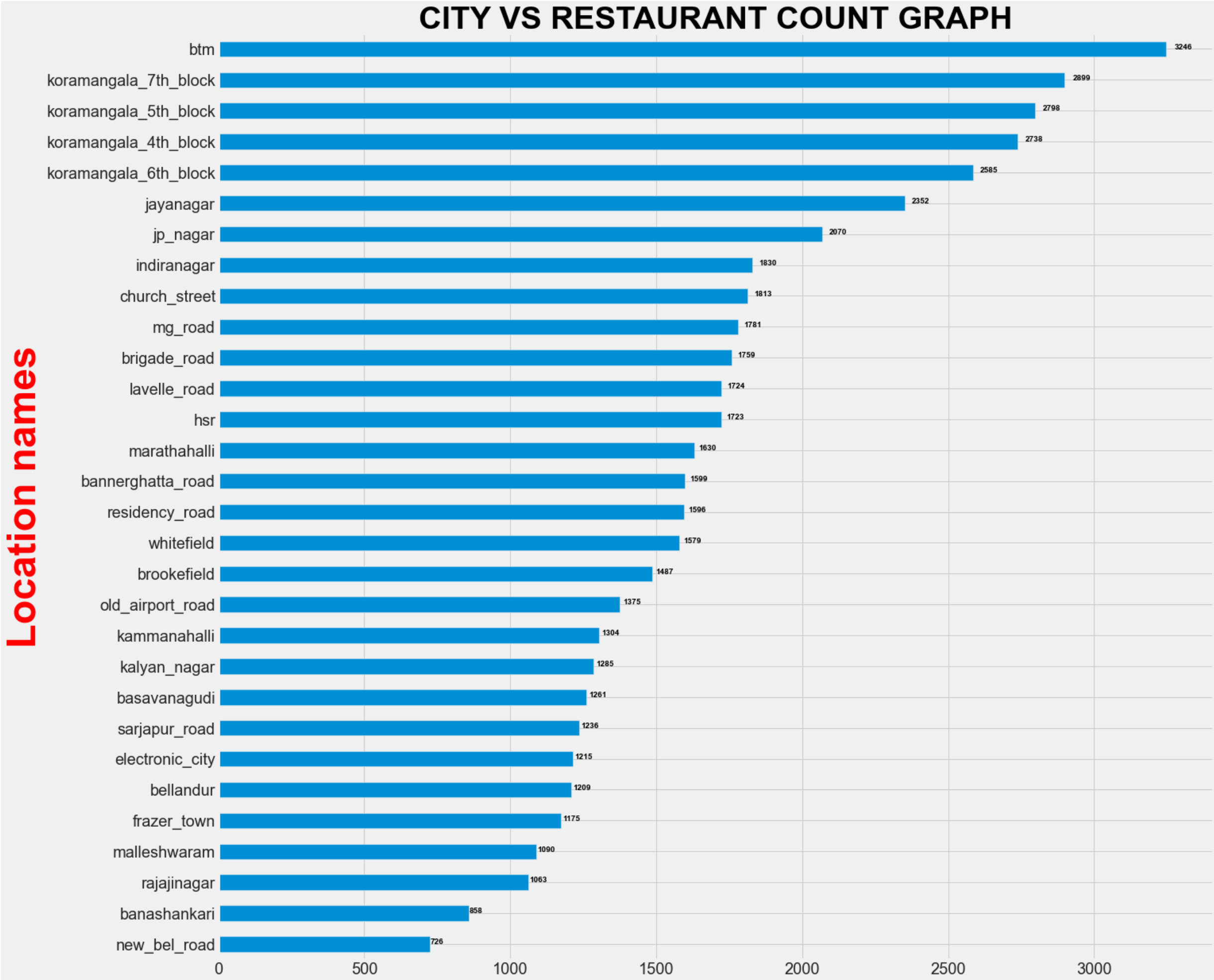
```
plt.xticks(rotation = 90)
plt.show()
```



Majority of the Restaurants of type 'Drinks & nightlife' and 'Pubs and bars' have a high median rating. The median value of these kind of restaurants is greater than the 75th Percentile value of rest of the restaurant types except that of 'Buffet' type. The IQR is highest for 'Desserts' category which indicates large amount of variation about median.

Analysing listed_in(city):

In [95]:
```
CityCount=data['listed_in(city)'].value_counts().sort_values(ascending=True)
fig=plt.figure(figsize=(20,20))
CityCount.plot(kind="barh",fontsize=20)
plt.ylabel("Location names",fontsize=50,color="red",fontweight='bold')
plt.title("CITY VS RESTAURANT COUNT GRAPH",fontsize=40,color="BLACK",fontweight='bol
for i in range(len(CityCount)):

    plt.text(i+CityCount[i],i,CityCount[i],fontsize=10,color="BLACK",fontweight='bol
```



In [ ]:

In [96]:
```
avgCityWiseRating = data.groupby('listed_in(city)').agg({'rate':['max','min']}).rese
avgCityWiseRating.columns =['listed_in(city)','MaxRatings','MinRatings']
avgCityWiseRating.head(15)
```

Out[96]:

	listed_in(city)	MaxRatings	MinRatings
0	banashankari	4.7	2.5
1	bannerghatta_road	4.7	2.2
2	basavanagudi	4.8	2.5
3	bellandur	4.9	2.1
4	brigade_road	4.9	1.8
5	brookefield	4.9	2.1
6	btm	4.9	2.2
7	church_street	4.9	1.8
8	electronic_city	4.7	2.4
9	frazer_town	4.9	2.1
10	hsr	4.7	2.3
11	indiranagar	4.9	2.1
12	jayanagar	4.9	2.3
13	jp_nagar	4.9	2.2

	listed_in(city)	MaxRatings	MinRatings
14	kalyan_nagar	4.8	2.3

In [100...

```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
stopwords = set(STOPWORDS)

wordcloud = WordCloud(width = 800, height = 800,
                background_color ='black',
                stopwords = stopwords,
                min_font_size = 10).generate(str(data['reviews_list']))

# plot the WordCloud image
plt.figure(figsize = (8,8), facecolor = 'blue')
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0) ;
```

We can observe that good,place,beautiful,friendly,nice,great,dine,service,food, ect so this shows majaority of people have given good reviews about that as size shows the count.

In [101...

```
#What are the best restaurants in Bangalore ?
#### has the highest possible rate , above average plus,
#### has the highest number of votes as it will more reliable plus
#### has the lowest possible cost
```

```
avg_Rating = data.rate.mean()
avg_Votes = data.votes.mean()

best_Rest_Banglore = data[(data.rate >=avg_Rating) & (data.votes >=avg_Votes)]
best_Rest_Banglore = best_Rest_Banglore.sort_values(['rate','votes','approx_cost(for

dfff=best_Rest_Banglore[['name','rate','votes','cuisines','approx_cost(for two peopl
dfff
```

Out[101…

	name	rate	votes	cuisines	approx_cost(for two people)	location	rest_type	
0	byg brewski brewing company	4.9	16832	continentalnorth_indian italiansouth_indianfin...	1600.0	sarjapur_road	microbrewery	cocktai
1	byg brewski brewing company	4.9	16832	continentalnorth_indian italiansouth_indianfin...	1600.0	sarjapur_road	microbrewery	cocktai
2	byg brewski brewing company	4.9	16832	continentalnorth_indian italiansouth_indianfin...	1600.0	sarjapur_road	microbrewery	cocktai
3	byg brewski brewing company	4.9	16345	continentalnorth_indian italiansouth_indianfin...	1600.0	sarjapur_road	microbrewery	cocktai
4	byg brewski brewing company	4.9	16345	continentalnorth_indian italiansouth_indianfin...	1600.0	sarjapur_road	microbrewery	cocktai

For Continental, North Indian, Chinese, European restaurants located in Koramangala 5th Block,Electronic City,Whitefield are the best like Biergarten,The Big Barbeque,You Mee restaurant.

For North Indian Food restaurants located in Whitefield are the best like Punjab Grill restaurant.

For South Indian Food restaurants located in Banashankari,Jayanagar are the best like Taaza Thindi,Puliyogare Point,Brahmin Tiffins & Coffee,Taaza Thindi,Sri Laxmi Venkateshwara Coffee Bar restaurant.

For Dessert restaurants located in Koramangala 5th Block,Vasanth Nagar,Kalyan Nagar are the best like Belgian Waffle Factory and Kurtoskalacs restaurant.

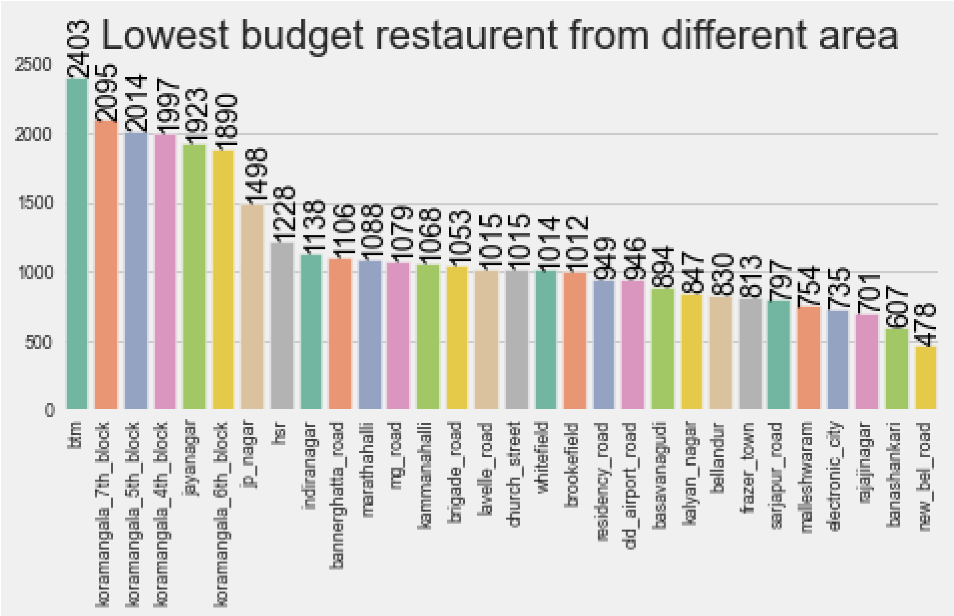
In [102…

```
# Low budget restaurent
low_budget = data.groupby(['name','rest_type','cuisines', 'listed_in(city)', 'rate',
low_budget = low_budget[low_budget["approx_cost(for two people)"] <= 1500]

# High budget restaurent
high_budget = data.groupby(['name','rest_type','cuisines', 'listed_in(city)', 'rate'
high_budget = high_budget[(high_budget["approx_cost(for two people)"] > 3000) & (hig
```

In [103…

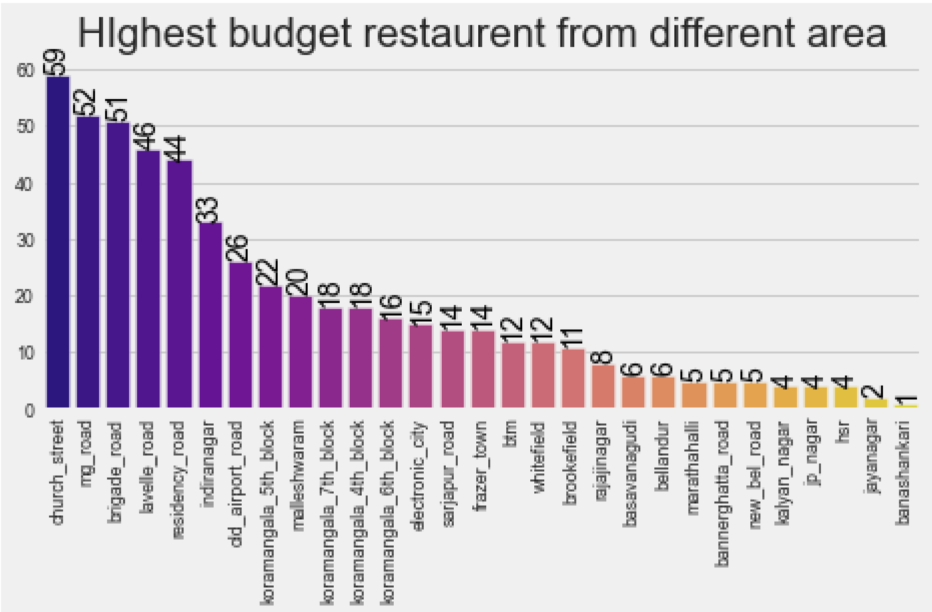
```
# Lowest Budget restaurent
```

```python
low = low_budget["listed_in(city)"].value_counts()
g = sns.barplot(y=low.values, x=low.index, palette="Set2")
plt.xticks(rotation=90)
plt.title("Lowest budget restaurent from different area")
for p in g.patches:
    g.annotate('{:.0f}'.format(p.get_height()), (p.get_x()+0.6, p.get_height()+1.3),
```



In [104…

```python
# High budget Restaurent

high = high_budget["listed_in(city)"].value_counts()
g = sns.barplot(x=high.index, y=high.values, palette="plasma")
plt.xticks(rotation=90)
plt.title("HIghest budget restaurent from different area")
for p in g.patches:
    g.annotate('{:.0f}'.format(p.get_height()), (p.get_x()+0.45, p.get_height()+0.1)
```



In [105…

```python
import dataframe_image as dfi

dfi.export (dfff ,'dfff.png')
```

Train test splitting

In [6]:

```python
#train test splitting:
```

```
x = data.iloc[:,[0,1,2,4,5,6,7,8,9,10,11,12,13]]
y = data['rate']
```

In [7]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.2,random_state=353)
```

In [8]:
```
x_train.shape
```

Out[8]: (40804, 13)

In [9]:
```
y_train.shape
```

Out[9]: (40804,)

In [10]:
```
x_test.shape
```

Out[10]: (10202, 13)

In [11]:
```
y_test.shape
```

Out[11]: (10202,)

To determine there is a relation between online order and ratings:

Here we have to conduct T test for this objective

Null hypothisis:

There is no difference in mean ratings between restaurants which are having online booking facility and not having online book facility

Alternative hypothisis:

There is a difference in mean ratings between restaurants which are having online booking facility and not having online book facility

Here we consider significance level of 5%

In [112...
```
from scipy import stats

yes = data[(data['online_order']=='yes')]
no = data[(data['online_order']=='no')]
```

In [113...
```
stats.ttest_ind(yes['rate'], no['rate'])
```

Out[113... Ttest_indResult(statistic=13.797616939342255, pvalue=3.1507216314588605e-43)

Here we got p value=1.5137681112879192e-42

p value is less than 0.05 hence we reject null value and accept alternative hypothisis

To determine there is a relation between book table and ratings:

In [114…
```
yes = data[(data['book_table']=='yes')]
no = data[(data['book_table']=='no')]
stats.ttest_ind(yes['rate'], no['rate'])
```

Out[114…
```
Ttest_indResult(statistic=102.77380626812825, pvalue=0.0)
```

Here we got p value=0.0

p value is less than 0.05 hence we reject null value and accept alternative hypothisis

To determine there is a relation between approx cost and ratings:

In [ ]:
```
import statsmodels.api as sm
Y = data['rate']
X = data['approx_cost(for two people)']
X = sm.add_constant(X)
model = sm.OLS(Y,X)
results = model.fit()
results.params
```

In [ ]:
```
results.tvalues
```

In [ ]:
```
print(results.summary())
```

BoW vectorizer

In [115…
```
data.head(2)
```

Out[115…

	name	online_order	book_table	rate	votes	location	rest_type	
0	jalsa	yes	yes	4.1	775	banashankari	casual_dining	pastalunch_buffetmasal
1	spice elephant	yes	no	4.1	787	banashankari	casual_dining	momoslunch_buffetch

In [13]:
```
# BoW vectorizer
from sklearn.feature_extraction.text import CountVectorizer
```

Review_list

In [14]:
```
vec = CountVectorizer()

#fitting countvectorizer using only train data
vec.fit(x_train["reviews_list"].values)
```

```
#transforming to vector representation for train,test data
x_train_reviews = vec.transform(x_train["reviews_list"].values)
x_test_reviews = vec.transform(x_test["reviews_list"].values)

print(x_train_reviews.shape)
print(x_test_reviews.shape)
```

```
(40804, 53997)
(10202, 53997)
```

online_order

In [15]:

```
#initializing the vectorizer
vec = CountVectorizer()

#fitting countvectorizer using only train data
vec.fit(x_train["online_order"].values)

#transforming to vector representation for train,test data
x_train_order = vec.transform(x_train["online_order"].values)
x_test_order = vec.transform(x_test["online_order"].values)

print(x_train_order.shape)
print(x_test_order.shape)
```

```
(40804, 2)
(10202, 2)
```

book_table

In [16]:

```
#initializing the vectorizer
vec = CountVectorizer()

#fitting countvectorizer using only train data
vec.fit(x_train["book_table"].values)

#transforming to vector representation for train,test data
x_train_table = vec.transform(x_train["book_table"].values)
x_test_table = vec.transform(x_test["book_table"].values)

print(x_train_table.shape)
print(x_test_table.shape)
```

```
(40804, 2)
(10202, 2)
```

location

In [17]:

```
#initializing the vectorizer
vec = CountVectorizer()

#fitting countvectorizer using only train data
vec.fit(x_train["location"].values)

#transforming to vector representation for train,test data
x_train_location = vec.transform(x_train["location"].values)
x_test_location = vec.transform(x_test["location"].values)

print(x_train_location.shape)
print(x_test_location.shape)
```

```
(40804, 93)
(10202, 93)
```

rest_type

In [18]:

```
#initializing the vectorizer
vec = CountVectorizer()

#fitting countvectorizer using only train data
vec.fit(x_train["rest_type"].values)

#transforming to vector representation for train,test data
x_train_rest = vec.transform(x_train["rest_type"].values)
x_test_rest = vec.transform(x_test["rest_type"].values)

print(x_train_rest.shape)
print(x_test_rest.shape)
```

```
(40804, 58)
(10202, 58)
```

dish_liked

In [19]:

```
#initializing the vectorizer
vec = CountVectorizer()

#fitting countvectorizer using only train data
vec.fit(x_train["dish_liked"].values)

#transforming to vector representation for train,test data
x_train_dish = vec.transform(x_train["dish_liked"].values)
x_test_dish = vec.transform(x_test["dish_liked"].values)

print(x_train_dish.shape)
print(x_test_dish.shape)
```

```
(40804, 6474)
(10202, 6474)
```

cuisines

In [20]:

```
#initializing the vectorizer
vec = CountVectorizer()

#fitting countvectorizer using only train data
vec.fit(x_train["cuisines"].values)

#transforming to vector representation for train,test data
x_train_cuisines = vec.transform(x_train["cuisines"].values)
x_test_cuisines = vec.transform(x_test["cuisines"].values)

print(x_train_cuisines.shape)
print(x_test_cuisines.shape)
```

```
(40804, 468)
(10202, 468)
```

menu_item

In [21]:

```
#initializing the vectorizer
vec = CountVectorizer()

#fitting countvectorizer using only train data
vec.fit(x_train["menu_item"].values)

#transforming to vector representation for train,test data
```

```
x_train_menu = vec.transform(x_train["menu_item"].values)
x_test_menu = vec.transform(x_test["menu_item"].values)

print(x_train_menu.shape)
print(x_test_menu.shape)
```

```
(40804, 120531)
(10202, 120531)
```

listed_in(type)

In [22]:
```
#initializing the vectorizer
vec = CountVectorizer()

#fitting countvectorizer using only train data
vec.fit(x_train["listed_in(type)"].values)

#transforming to vector representation for train,test data
x_train_type = vec.transform(x_train["listed_in(type)"].values)
x_test_type = vec.transform(x_test["listed_in(type)"].values)

print(x_train_type.shape)
print(x_test_type.shape)
```

```
(40804, 9)
(10202, 9)
```

listed_in(city)

In [23]:
```
#initializing the vectorizer
vec = CountVectorizer()

#fitting countvectorizer using only train data
vec.fit(x_train["listed_in(city)"].values)

#transforming to vector representation for train,test data
x_train_city = vec.transform(x_train["listed_in(city)"].values)
x_test_city = vec.transform(x_test["listed_in(city)"].values)

print(x_train_city.shape)
print(x_test_city.shape)
```

```
(40804, 30)
(10202, 30)
```

Standardization of numerical variables

In [24]:
```
#Standardization of numerical variables
from sklearn.preprocessing import StandardScaler
```

votes

In [25]:
```
std = StandardScaler()

#finding mean and standrd deviation using train data
std.fit(x_train["votes"].values.reshape(-1,1))

#standardizing train and test data using mean and std calculated using train data
x_train_votes = std.transform(x_train["votes"].values.reshape(-1,1))
x_test_votes = std.transform(x_test["votes"].values.reshape(-1,1))
```

```
print(x_train_votes.shape)
print(x_test_votes.shape)
```

```
(40804, 1)
(10202, 1)
```

approx_cost(for two people)

In [26]:
```
std = StandardScaler()

#finding mean and standrd deviation using train data
std.fit(x_train["approx_cost(for two people)"].values.reshape(-1,1))

#standardizing train and test data using mean and std calculated using train data
x_train_approx_cost = std.transform(x_train["approx_cost(for two people)"].values.re
x_test_approx_cost  = std.transform(x_test["approx_cost(for two people)"].values.res


print(x_train_approx_cost.shape)
print(x_test_approx_cost.shape)
```

```
(40804, 1)
(10202, 1)
```

Concatenating all features

In [27]:
```
from scipy.sparse import hstack
```

In [28]:
```
#Concatenating all features
x_tr=hstack((x_train_reviews,x_train_order,x_train_table,x_train_location,x_train_cu
x_te=hstack((x_test_reviews,x_test_order,x_test_table,x_test_location,x_test_cuisine

print("FINAL DATA MATRIX SHAPE IS ........")
print(x_tr.shape,y_train.shape)
print(x_te.shape,y_test.shape)
print("*"*100)
```

```
FINAL DATA MATRIX SHAPE IS ........
(40804, 181666) (40804,)
(10202, 181666) (10202,)
************************************************************************************
****************
```

The data is ready for modelling

ML Models

1. Linear regression

In [132…
```
#Linear regression
linear_regression = LinearRegression()

linear_regression.fit(x_tr, y_train)
```

Out[132…
```
LinearRegression()
```

In [133…
```
y_1_pred = linear_regression.predict(x_te)
print(r2_score(y_test, y_1_pred, multioutput='uniform_average'))
```

```
-306.3282845472227
```

3. Decision Tree regressor

In [32]:
```
#Decision Tree regressor
decision_tree = DecisionTreeRegressor()
decision_tree.fit(x_tr, y_train)
```

Out[32]:
```
DecisionTreeRegressor()
```

In [33]:
```
y_pred = decision_tree.predict(x_te)
print(r2_score(y_test, y_pred, multioutput='uniform_average'))
```

```
0.8596560244990527
```