In [1]:
```
#importing libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
df=pd.read_csv('zomato.csv')
plt.rcParams["figure.figsize"] = (18,12)
```

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

Out[3]:

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

In [4]:
```
df.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 [5]:
```
#rate,phone,location,rest_type,dish_liked,cuisines,approx cost have null values
df.isna().sum()
```

Out[5]:
```
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 [6]:
```
#removing value which dont have any signficance
df = df.drop(['url', 'address', 'phone', 'menu_item', 'dish_liked', 'reviews_list'], axis = 1)
```

In [7]:
```
df.head()
```

Out[7]:

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

In [8]:
```
df.drop('location',axis=1)
```

Out[8]:

	name	online_order	book_table	rate	votes	rest_type	cuisines	approx_cost(for two people)
0	Jalsa	Yes	Yes	4.1/5	775	Casual Dining	North Indian, Mughlai, Chinese	800
1	Spice Elephant	Yes	No	4.1/5	787	Casual Dining	Chinese, North Indian, Thai	800
2	San Churro Cafe	Yes	No	3.8/5	918	Cafe, Casual Dining	Cafe, Mexican, Italian	800
3	Addhuri Udupi Bhojana	No	No	3.7/5	88	Quick Bites	South Indian, North Indian	300
4	Grand Village	No	No	3.8/5	166	Casual Dining	North Indian, Rajasthani	600
...	...	...	...	...	...	...	...	...
51712	Best Brews - Four Points by Sheraton Bengaluru...	No	No	3.6 /5	27	Bar	Continental	1,500
51713	Vinod Bar And Restaurant	No	No	NaN	0	Bar	Finger Food	600
51714	Plunge - Sheraton Grand Bengaluru Whitefield H...	No	No	NaN	0	Bar	Finger Food	2,000
51715	Chime - Sheraton Grand Bengaluru Whitefield Ho...	No	Yes	4.3 /5	236	Bar	Finger Food	2,500
51716	The Nest - The Den Bengaluru	No	No	3.4 /5	13	Bar, Casual Dining	Finger Food, North Indian, Continental	1,500

51717 rows × 10 columns

In [9]:
```
df.rename(columns={'listed_in(city)':'location_new','rest_type':'Category','listed_in(type)':'Types','approx_cost(for two people)':'Cost_for2'},inplace=True)
```

In [10]:
```
df.head()
```

Out[10]:

	name	online_order	book_table	rate	votes	location	Category	cuisines	Cost_for2
0	Jalsa	Yes	Yes	4.1/5	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	80(
1	Spice Elephant	Yes	No	4.1/5	787	Banashankari	Casual Dining	Chinese, North Indian, Thai	80(
2	San Churro Cafe	Yes	No	3.8/5	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	80(
3	Addhuri Udupi Bhojana	No	No	3.7/5	88	Banashankari	Quick Bites	South Indian, North Indian	30(
4	Grand Village	No	No	3.8/5	166	Basavanagudi	Casual Dining	North Indian, Rajasthani	60(

In [11]:
```
df.isna().sum()
```

Out[11]:
```
name              0
online_order      0
book_table        0
rate           7775
votes             0
location         21
Category        227
cuisines         45
Cost_for2       346
Types             0
location_new      0
dtype: int64
```

In [12]:
```
df.drop_duplicates(inplace = True)        #dropping the duplictate values
```

In [13]:
```
df.shape
```

Out[13]:
```
(51609, 11)
```

In [14]:
```
df.Cost_for2.info()
```

```
<class 'pandas.core.series.Series'>
Int64Index: 51609 entries, 0 to 51716
Series name: Cost_for2
Non-Null Count  Dtype
--------------  -----
51265 non-null  object
dtypes: object(1)
memory usage: 806.4+ KB
```

In [15]:
```
def handlecomma(value):             #removing comma from the values
    value = str(value)
    if ',' in value:
        value = value.replace(',', '')
        return float(value)
    else:
        return float(value)

df['Cost_for2'] = df['Cost_for2'].apply(handlecomma)
df['Cost_for2'].unique()
```

Out[15]:
```
array([ 800.,  300.,  600.,  700.,  550.,  500.,  450.,  650.,  400.,
        900.,  200.,  750.,  150.,  850.,  100., 1200.,  350.,  250.,
        950., 1000., 1500., 1300.,  199.,   80., 1100.,  160., 1600.,
        230.,  130.,   50.,  190., 1700.,   nan, 1400.,  180., 1350.,
       2200., 2000., 1800., 1900.,  330., 2500., 2100., 3000., 2800.,
       3400.,   40., 1250., 3500., 4000., 2400., 2600.,  120., 1450.,
        469.,   70., 3200.,   60.,  560.,  240.,  360., 6000., 1050.,
       2300., 4100., 5000., 3700., 1650., 2700., 4500.,  140.])
```

In [16]:
```
def handlerate(value):
    if(value=='nan'):
        return np.nan
```

In [17]:
```
df.Cost_for2.isna().sum()
```

Out[17]:
```
344
```

In [18]:
```
df['Cost_for2'].fillna(df['Cost_for2'].mode(), inplace = True)  #filling null
 values
```

In [19]:
```
df.Cost_for2.unique()
```

Out[19]:
```
array([ 800.,  300.,  600.,  700.,  550.,  500.,  450.,  650.,  400.,
        900.,  200.,  750.,  150.,  850.,  100., 1200.,  350.,  250.,
        950., 1000., 1500., 1300.,  199.,   80., 1100.,  160., 1600.,
        230.,  130.,   50.,  190., 1700.,   nan, 1400.,  180., 1350.,
       2200., 2000., 1800., 1900.,  330., 2500., 2100., 3000., 2800.,
       3400.,   40., 1250., 3500., 4000., 2400., 2600.,  120., 1450.,
        469.,   70., 3200.,   60.,  560.,  240.,  360., 6000., 1050.,
       2300., 4100., 5000., 3700., 1650., 2700., 4500.,  140.])
```

In [20]:
```
df.isna().sum()
```

Out[20]:
```
name              0
online_order      0
book_table        0
rate           7755
votes             0
location         21
Category        227
cuisines         45
Cost_for2       344
Types             0
location_new      0
dtype: int64
```

In [21]:
```
df.rate.unique()
```

Out[21]:
```
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)
```

In [22]:
```
def handlerate(value):            #removing "/"from ratings
    if(value=='NEW' or value=='-'):
        return np.nan
    else:
        value = str(value).split('/')
        value = value[0]
        return float(value)

df['rate'] = df['rate'].apply(handlerate)
df['rate'].head()
```

Out[22]:
```
0    4.1
1    4.1
2    3.8
3    3.7
4    3.8
Name: rate, dtype: float64
```

In [23]:
```
df=df.drop('location',axis=1)              #listed(in city) and location are same and hence removing the one with null values
```

In [24]:
```
df.isna().sum()
```

Out[24]:
```
name               0
online_order       0
book_table         0
rate           10019
votes              0
Category         227
cuisines          45
Cost_for2        344
Types              0
location_new       0
dtype: int64
```

In [25]:
```
df.shape
```

Out[25]:
```
(51609, 10)
```

In [26]:
```
df['rate'].fillna(df['rate'].mean(), inplace = True)      #filling null vlues
 of rate column
```

In [27]:
```
df.isna().sum()
```

Out[27]:
```
name             0
online_order     0
book_table       0
rate             0
votes            0
Category       227
cuisines        45
Cost_for2      344
Types            0
location_new     0
dtype: int64
```

In [28]:
```
df['cuisines'].fillna('Other', inplace = True)              #filling null vlu
es of cuisines column
```

In [29]:
```
df['Category'].fillna('Other', inplace = True               #filling null vlu
es of Category column
```

In [30]:
```
df.isna().sum()
```

Out[30]:
```
name             0
online_order     0
book_table       0
rate             0
votes            0
Category         0
cuisines         0
Cost_for2      344
Types            0
location_new     0
dtype: int64
```

In [31]:
```
df['Cost_for2'].fillna(df['Cost_for2'].mean(), inplace = True)          #fillin
g null vlues of Cost_for2 column
```

In [32]:
```
df['Types'].fillna('others', inplace = True)                            #filling
null vlues of Types column
```

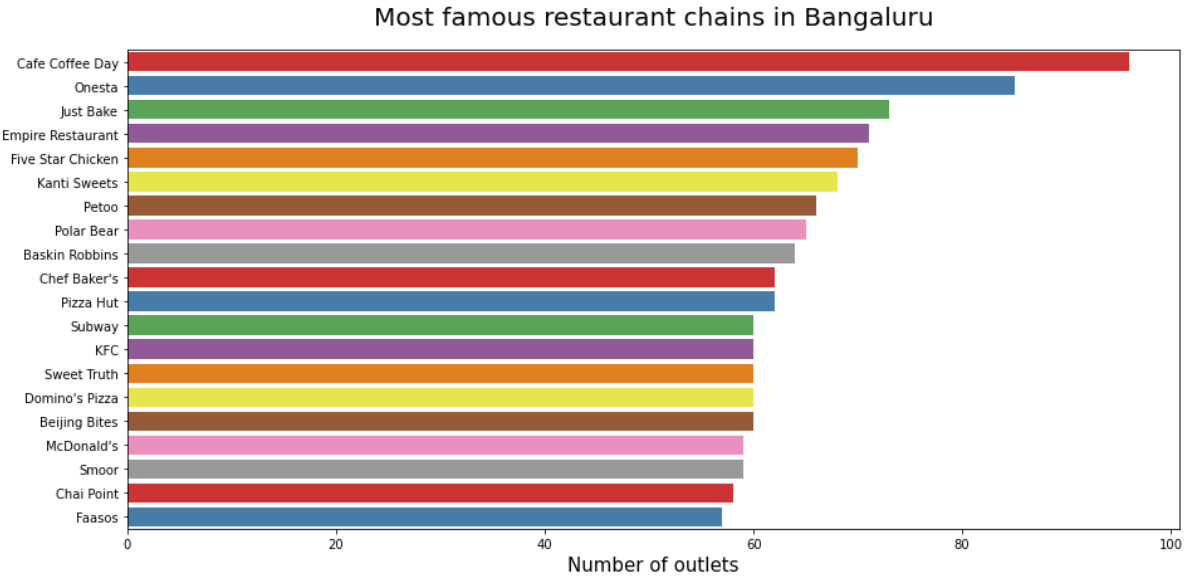
In [33]:
```
df.isna().sum()
```

Out[33]:
```
name            0
online_order    0
book_table      0
rate            0
votes           0
Category        0
cuisines        0
Cost_for2       0
Types           0
location_new    0
dtype: int64
```

In [34]:
```
#analysis
```

In [35]:
```
#famous restaurant
plt.figure(figsize=(15,7))
chains=df['name'].value_counts()[:20]
sns.barplot(x=chains,y=chains.index,palette='Set1')
plt.title("Most famous restaurant chains in Bangaluru",size=20,pad=20)
plt.xlabel("Number of outlets",size=15)
```

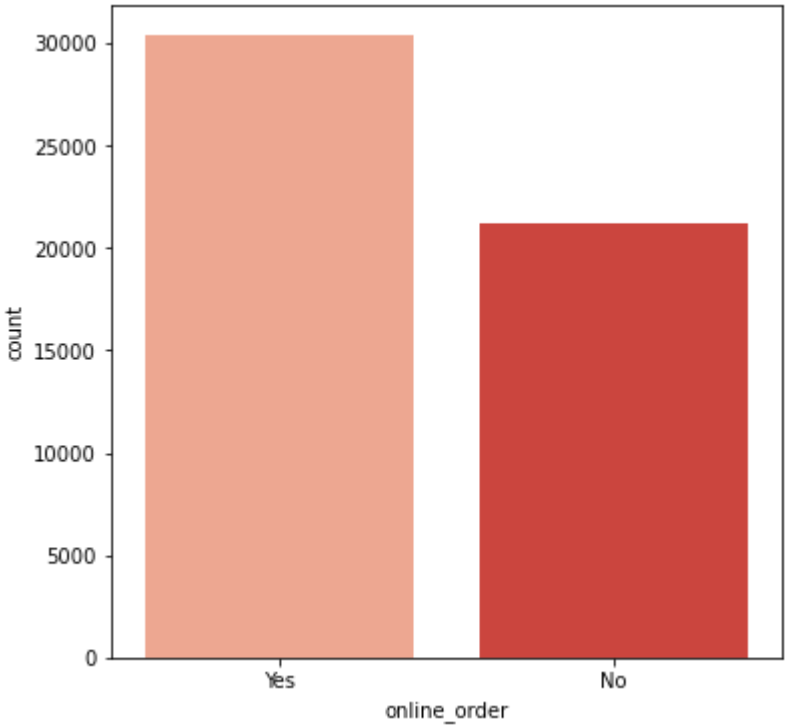
Out[35]:
```
Text(0.5, 0, 'Number of outlets')
```



In [36]:
```
#online order
plt.figure(figsize=(6,6))
sns.countplot(df['online_order'],palette='Reds')
```

```
C:\Users\my pc\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureW
arning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

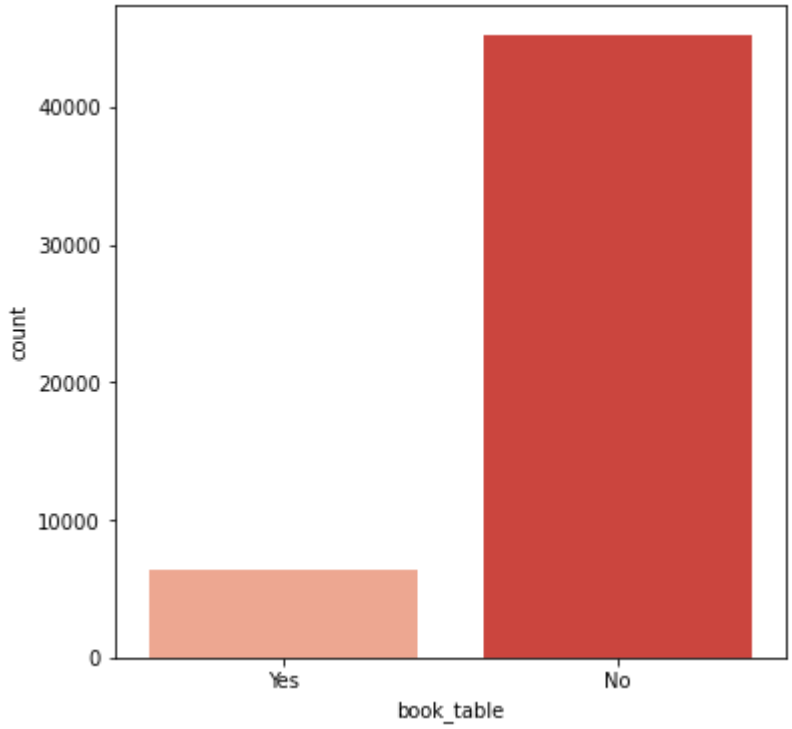
Out[36]: <AxesSubplot:xlabel='online_order', ylabel='count'>



In [37]:
```
#book table
plt.figure(figsize = (6,6))
sns.countplot(df['book_table'], palette = 'Reds')
```

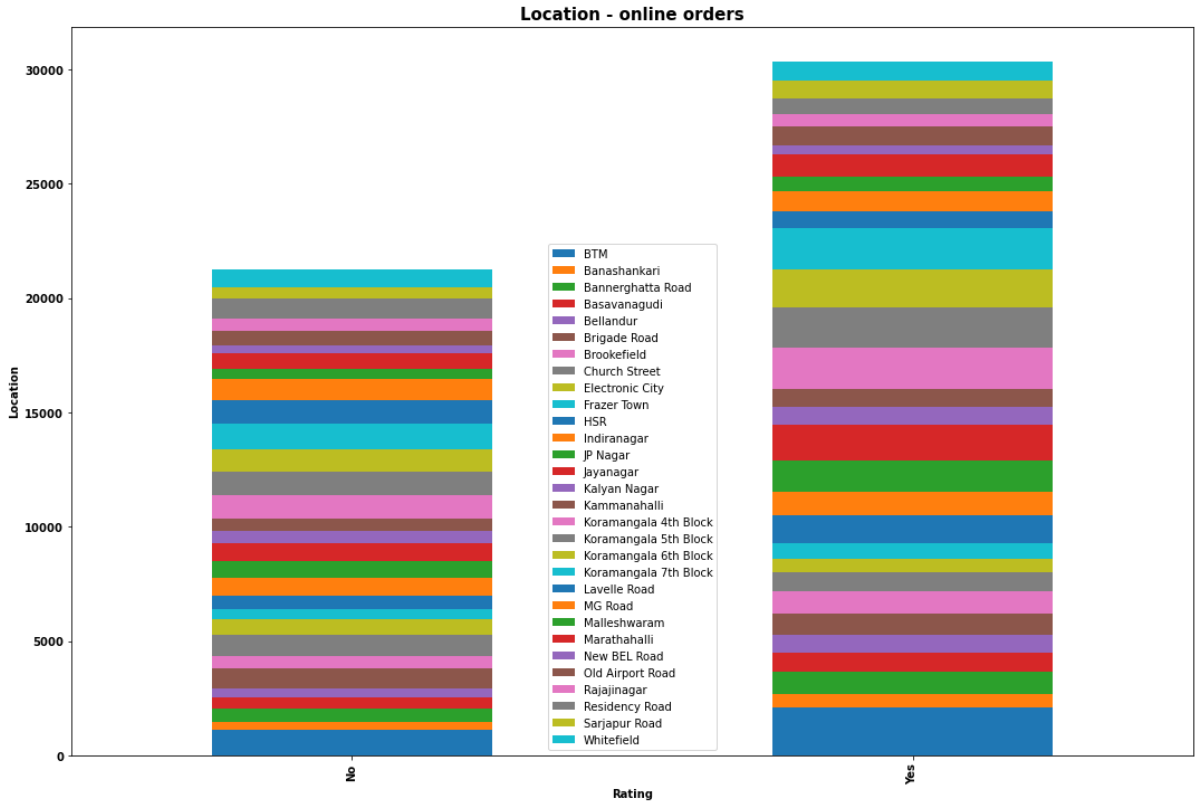
```
C:\Users\my pc\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureW
arning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[37]: <AxesSubplot:xlabel='book_table', ylabel='count'>



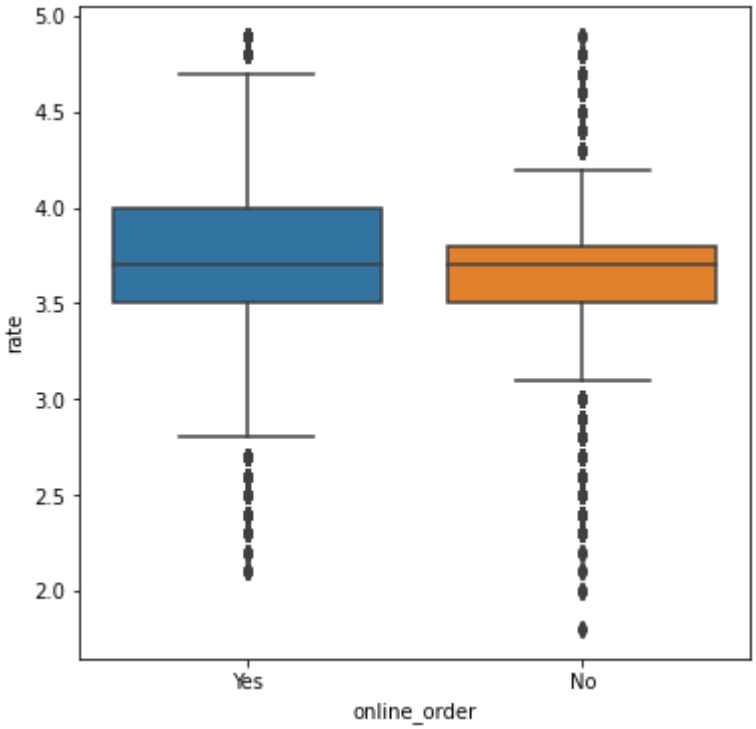
In [38]:
```
#location-wise online orders
loc_plt=pd.crosstab(df['online_order'],df['location_new'])
loc_plt.plot(kind='bar',stacked=True);
plt.title('Location - online orders',fontsize=15,fontweight='bold')
plt.ylabel('Location',fontsize=10,fontweight='bold')
plt.xlabel('Rating',fontsize=10,fontweight='bold')
plt.xticks(fontsize=10,fontweight='bold')
plt.yticks(fontsize=10,fontweight='bold');
plt.legend()
```

Out[38]: <matplotlib.legend.Legend at 0x21fc20a20d0>



In [39]:
```
#online_order vs Rate
plt.figure(figsize = (6,6))
sns.boxplot(x = 'online_order', y = 'rate', data = df)
```

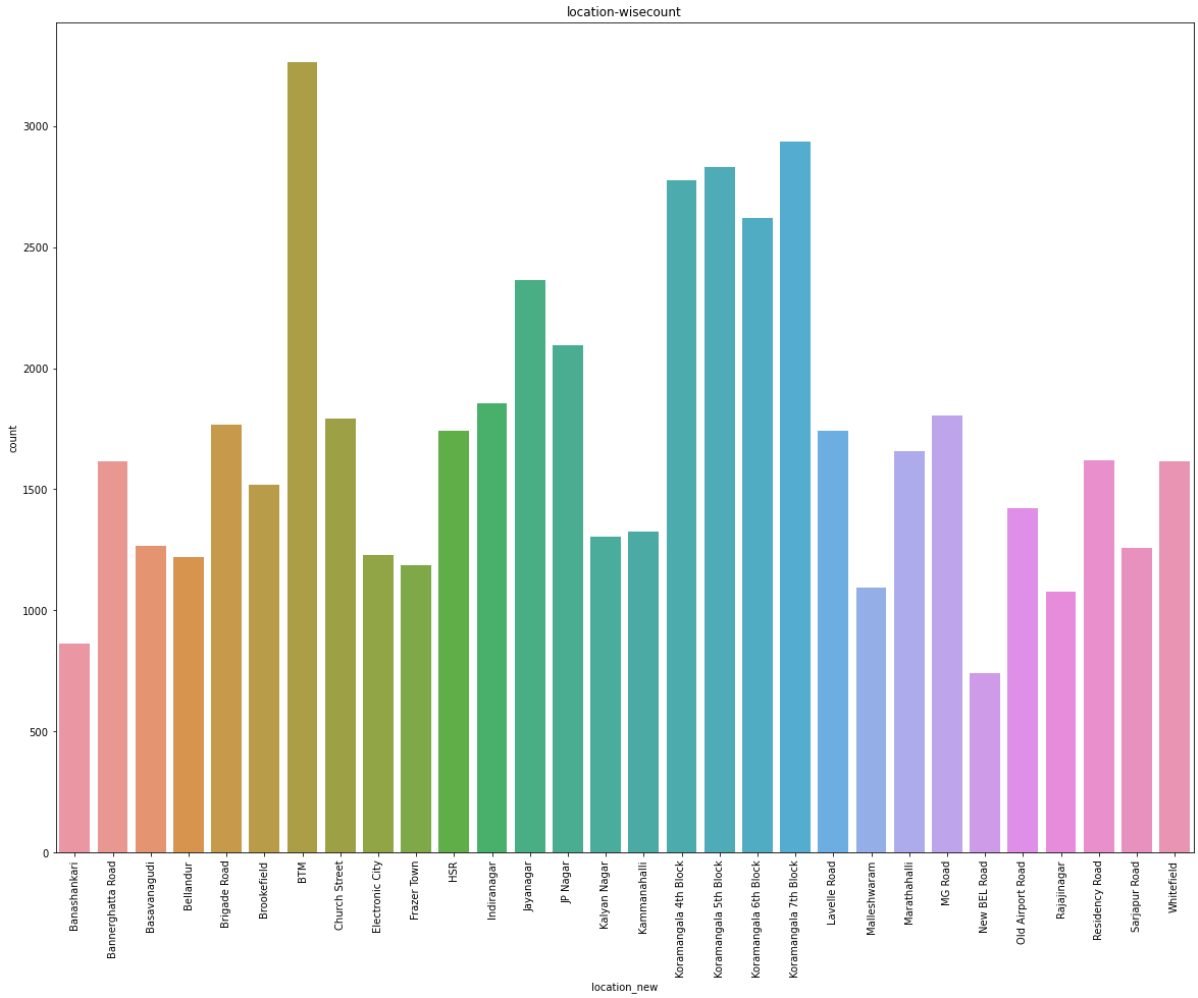
Out[39]: <AxesSubplot:xlabel='online_order', ylabel='rate'>



In [40]:
```
#location-count
plt.figure(figsize = (20,15))
ax = sns.countplot(df['location_new'])
plt.title('location-wisecount')
plt.xticks(rotation=90)
```

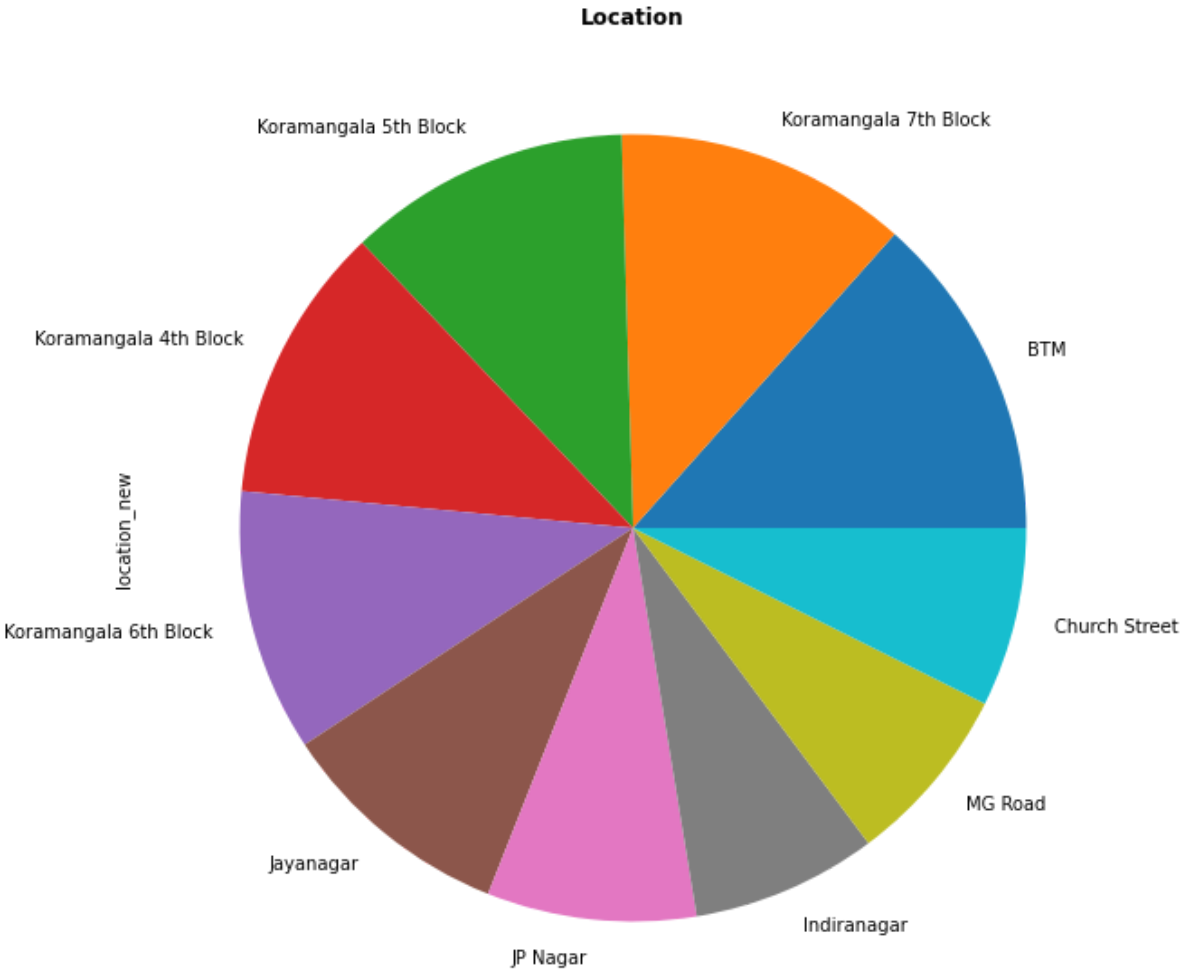
```
C:\Users\my pc\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureW
arning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[40]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]),
 [Text(0, 0, 'Banashankari'),
  Text(1, 0, 'Bannerghatta Road'),
  Text(2, 0, 'Basavanagudi'),
  Text(3, 0, 'Bellandur'),
  Text(4, 0, 'Brigade Road'),
  Text(5, 0, 'Brookefield'),
  Text(6, 0, 'BTM'),
  Text(7, 0, 'Church Street'),
  Text(8, 0, 'Electronic City'),
  Text(9, 0, 'Frazer Town'),
  Text(10, 0, 'HSR'),
  Text(11, 0, 'Indiranagar'),
  Text(12, 0, 'Jayanagar'),
  Text(13, 0, 'JP Nagar'),
  Text(14, 0, 'Kalyan Nagar'),
  Text(15, 0, 'Kammanahalli'),
  Text(16, 0, 'Koramangala 4th Block'),
  Text(17, 0, 'Koramangala 5th Block'),
  Text(18, 0, 'Koramangala 6th Block'),
  Text(19, 0, 'Koramangala 7th Block'),
  Text(20, 0, 'Lavelle Road'),
  Text(21, 0, 'Malleshwaram'),
  Text(22, 0, 'Marathahalli'),
  Text(23, 0, 'MG Road'),
  Text(24, 0, 'New BEL Road'),
  Text(25, 0, 'Old Airport Road'),
  Text(26, 0, 'Rajajinagar'),
  Text(27, 0, 'Residency Road'),
  Text(28, 0, 'Sarjapur Road'),
  Text(29, 0, 'Whitefield')])
```

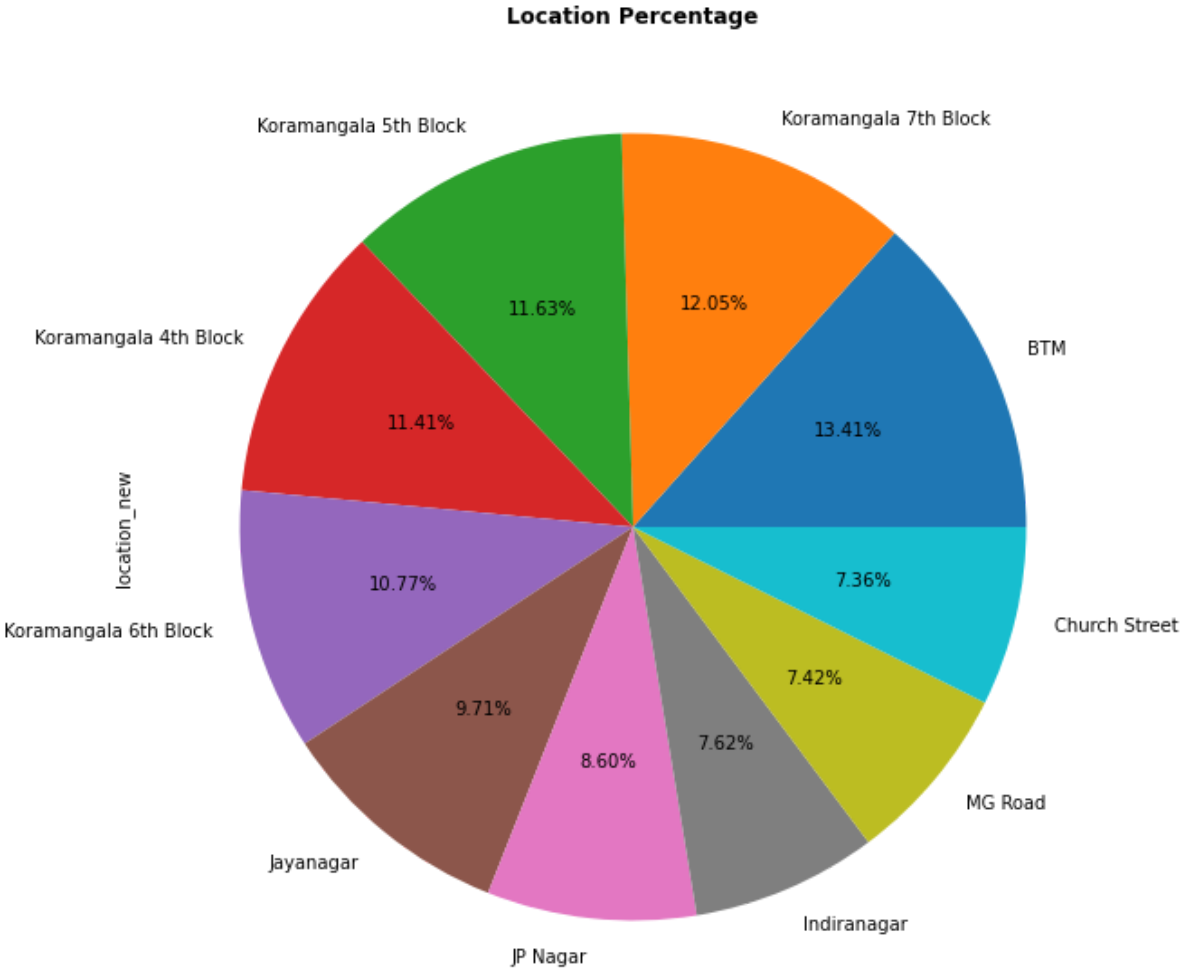

location-wisecount
count
location_new
0
500
1000
1500
2000
2500
3000
Banashankari
Bannerghatta Road
Basavanagudi
Bellandur
Brigade Road
Brookefield
BTM
Church Street
Electronic City
Frazer Town
HSR
Indiranagar
Jayanagar
JP Nagar
Kalyan Nagar
Kammanahalli
Koramangala 4th Block
Koramangala 5th Block
Koramangala 6th Block
Koramangala 7th Block
Lavelle Road
Malleshwaram
Marathahalli
MG Road
New BEL Road
Old Airport Road
Rajajinagar
Residency Road
Sarjapur Road
Whitefield

In [41]:
```
#loation distribution in pie chart
plt.figure(figsize=(15,10))
df['location_new'].value_counts()[:10].plot(kind = 'pie')
plt.title('Location', weight = 'bold')
```

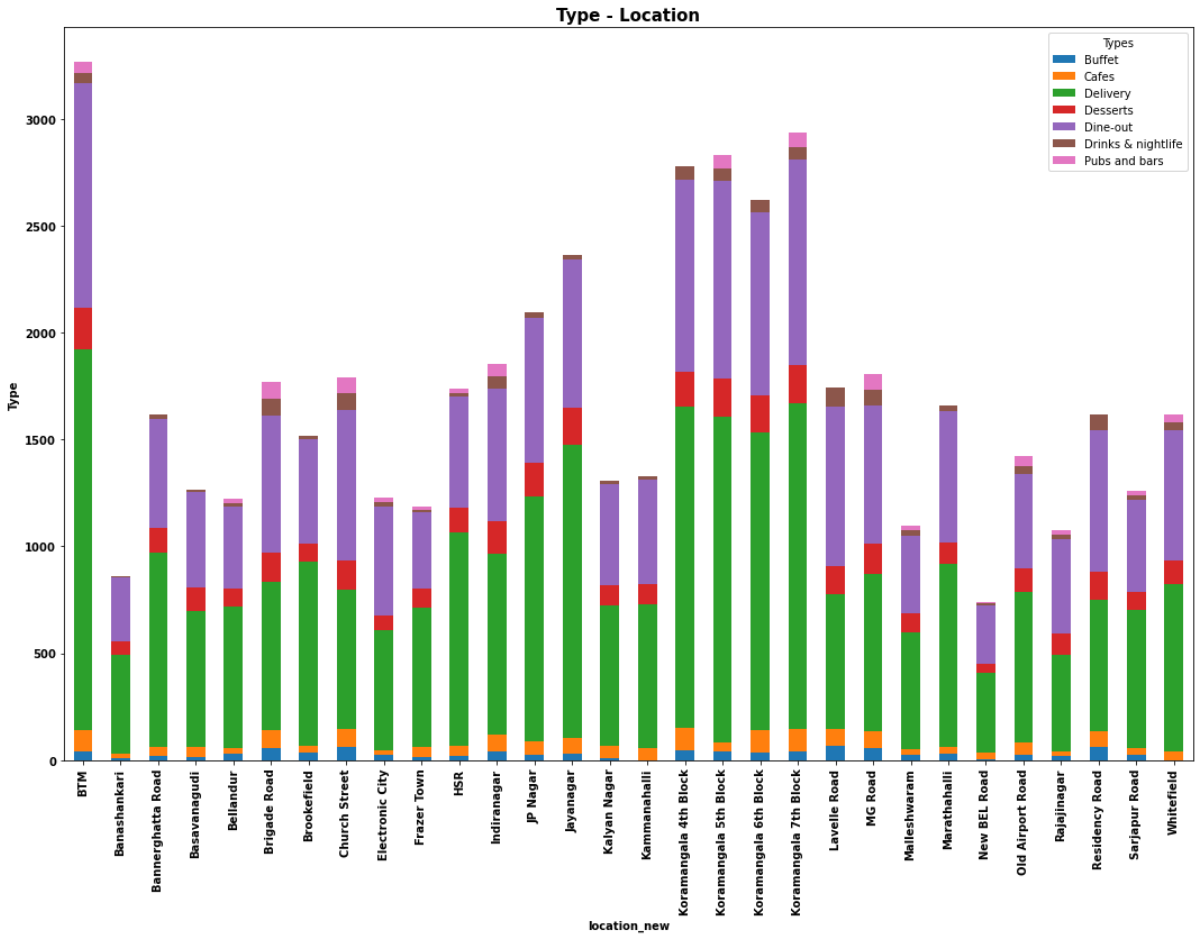
Out[41]: Text(0.5, 1.0, 'Location')



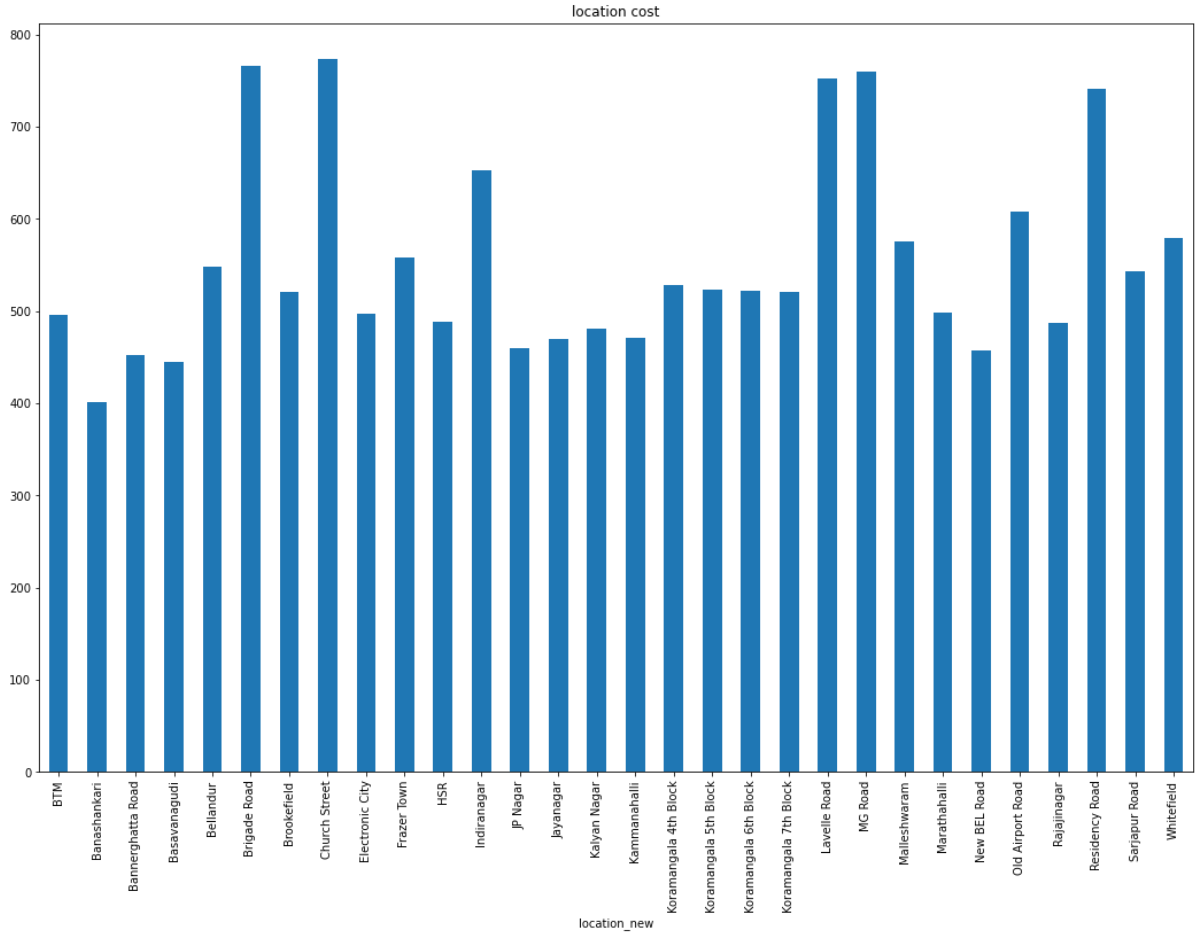
In [42]:
```
#location cout in pie chart
plt.figure(figsize=(15,10))
ax=df.location_new.value_counts()[:10].plot(kind='pie',autopct='%1.2f%%')
plt.title('Location Percentage', weight='bold')
plt.show()
```



In [43]:
```
#location-type
type_plt=pd.crosstab(df['location_new'],df['Types'])
type_plt.plot(kind='bar',stacked=True);
plt.title('Type - Location',fontsize=15,fontweight='bold')
plt.ylabel('Type',fontsize=10,fontweight='bold')
plt.xlabel('location_new',fontsize=10,fontweight='bold')
plt.xticks(fontsize=10,fontweight='bold')
plt.yticks(fontsize=10,fontweight='bold');
```



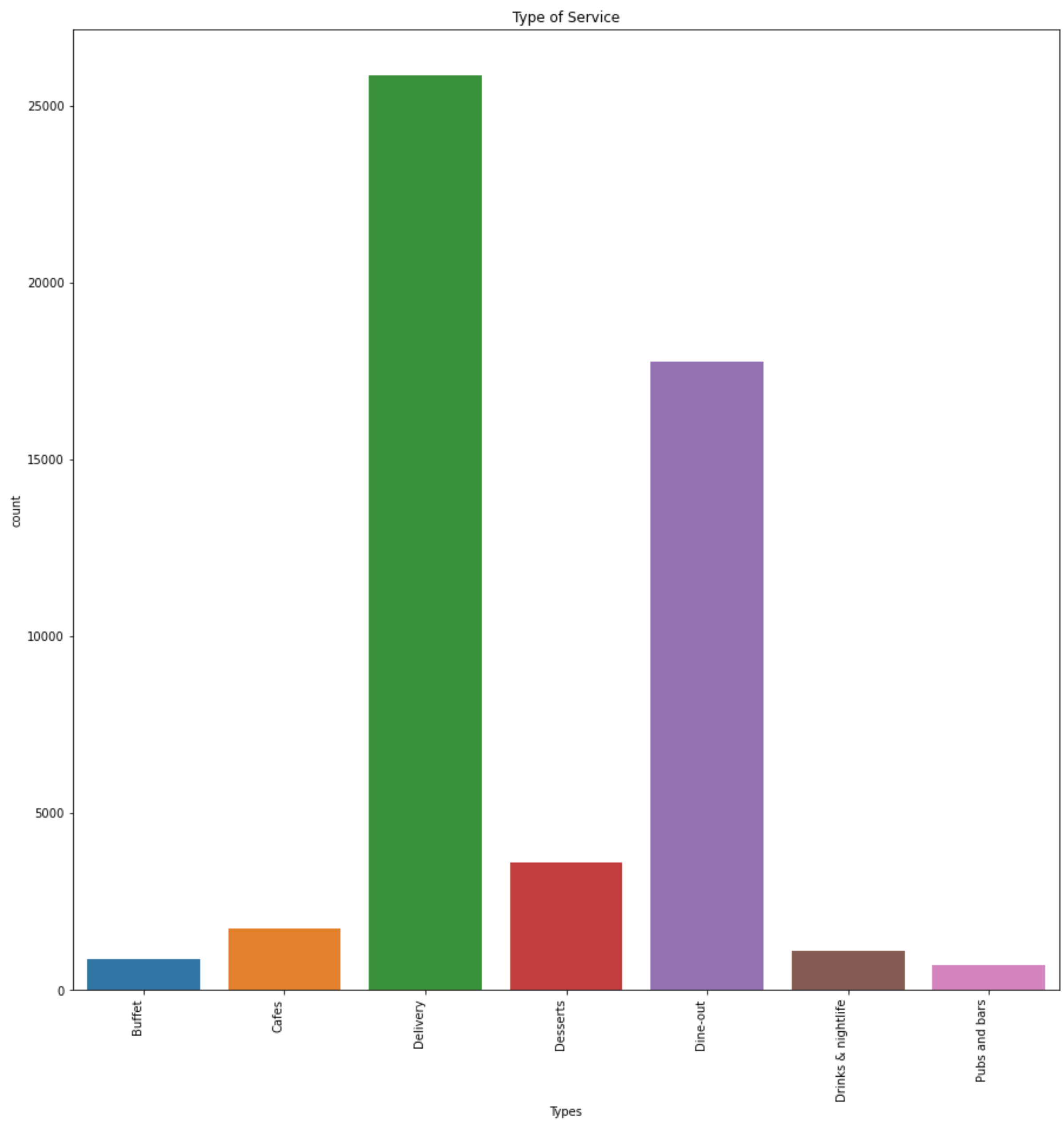
In [44]:
```
#plot the bar graph of loction and cost for two
df.groupby('location_new')['Cost_for2'].mean().plot.bar()
plt.title('location cost')
plt.show()
```



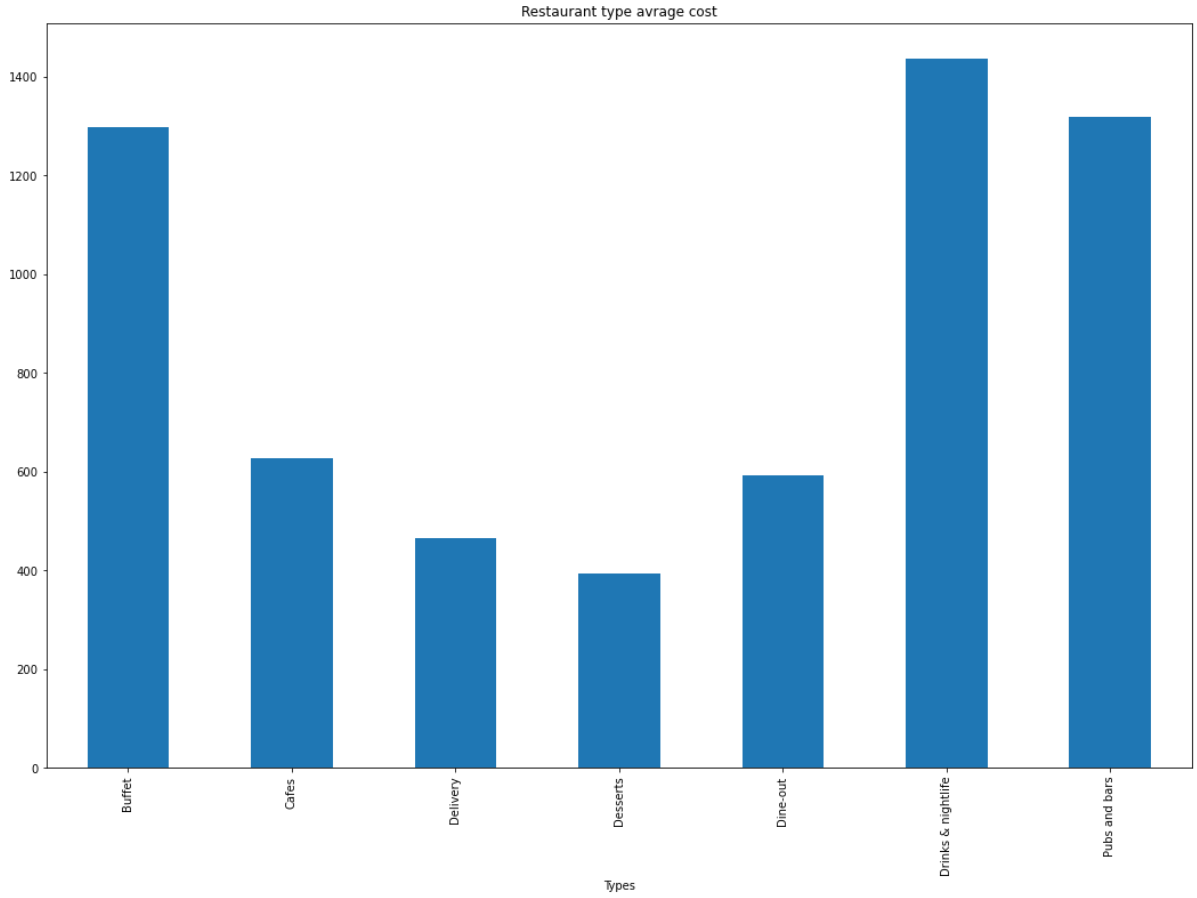
In [45]:
```
#type count
sns.countplot(df['Types'])
sns.countplot(df['Types']).set_xticklabels(sns.countplot(df['Types']).get_xtic
klabels(), rotation=90, ha="right")
fig = plt.gcf()
fig.set_size_inches(15,15)
plt.title('Type of Service')
```

```
C:\Users\my pc\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureW
arning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\my pc\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureW
arning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\my pc\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureW
arning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[45]: `Text(0.5, 1.0, 'Type of Service')`

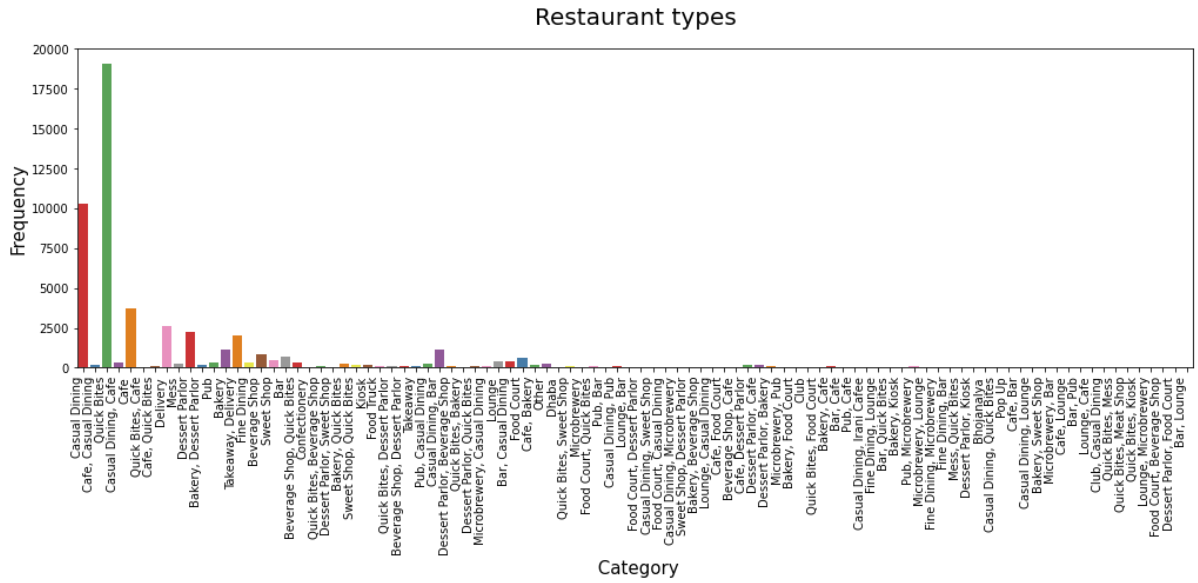


In [46]:
```
#graph between cost and type
df.groupby('Types')['Cost_for2'].mean().plot.bar()
plt.title('Restaurant type avrage cost')
plt.show()
```

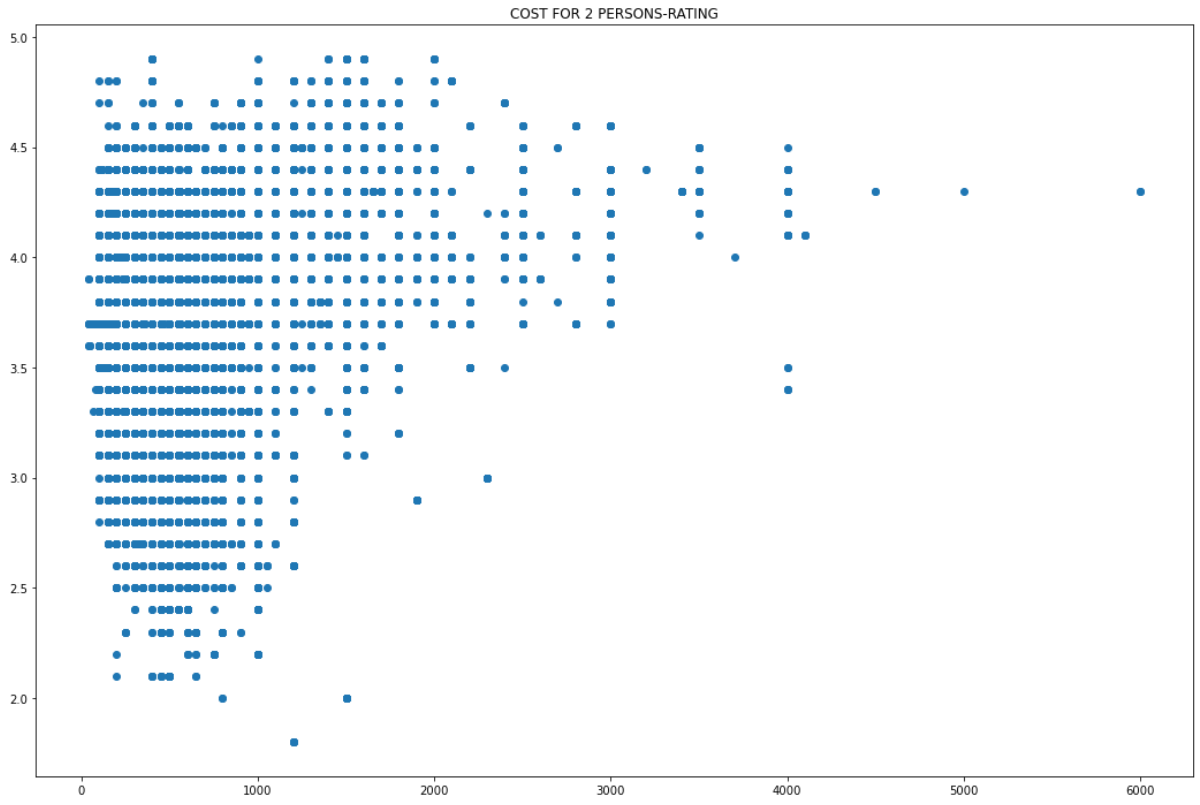


In [47]:
```
#Restaurant category
fig = plt.figure(figsize=(17,5))
rest = sns.countplot(x="Category",data=df, palette = "Set1")
rest.set_xticklabels(rest.get_xticklabels(), rotation=90, ha="right")
plt.ylabel("Frequency",size=15)
plt.xlabel(" Category",size=15)
rest
plt.title('Restaurant types',fontsize = 20 ,pad=20)
```

Out[47]: Text(0.5, 1.0, 'Restaurant types')



In [48]:
```
#avg_cost vs rate
plt.scatter(df.Cost_for2,df.rate)
plt.title('COST FOR 2 PERSONS-RATING')
plt.show()
```



In [49]:
```
df.describe()
```

Out[49]:

	rate	votes	Cost_for2
count	51609.000000	51609.000000	51609.000000
mean	3.700142	283.283361	555.170682
std	0.395393	803.282771	437.123484
min	1.800000	0.000000	40.000000
25%	3.500000	7.000000	300.000000
50%	3.700142	41.000000	400.000000
75%	3.900000	198.000000	650.000000
max	4.900000	16832.000000	6000.000000