In [1]:
```
# ZOMATO DATA SET ANALYSIS AND VISUALIZATION.
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('dark_background')
```

In [3]:
```
# reading csv
```

In [4]:
```
df=pd.read_csv('C:\\Users\\rajesh\\Downloads\\zomato.csv.zip')
df.head()
```

Out[4]:

	url	address	name	online_order	book_table	rate	votes	phone	location
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	080 42297555\r\n+91 9743772233	Banashankari
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	080 41714161	Banashankari
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5	918	+91 9663487993	Banashankari
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7/5	88	+91 9620009302	Banashankari
4	https://www.zomato.com/bangalore/grand-village...	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.8/5	166	+91 8026612447\r\n+91 9901210005	Basavanagudi

In [5]:
```
df.shape
```

Out[5]:
```
(51717, 17)
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['url', 'address', 'name', 'online_order', 'book_table', 'rate', 'votes',
       'phone', 'location', 'rest_type', 'dish_liked', 'cuisines',
       'approx_cost(for two people)', 'reviews_list', 'menu_item',
       'listed_in(type)', 'listed_in(city)'],
      dtype='object')
```

In [7]:
```
df=df.drop(['url','address','phone','menu_item','dish_liked','reviews_list'],axis=1)
df.head()
```

Out[7]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	approx_cost(for two people)	listed_in(type)	listed_in(city)
0	Jalsa	Yes	Yes	4.1/5	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800	Buffet	Banashankari
1	Spice Elephant	Yes	No	4.1/5	787	Banashankari	Casual Dining	Chinese, North Indian, Thai	800	Buffet	Banashankari
2	San Churro Cafe	Yes	No	3.8/5	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	800	Buffet	Banashankari
3	Addhuri Udupi Bhojana	No	No	3.7/5	88	Banashankari	Quick Bites	South Indian, North Indian	300	Buffet	Banashankari
4	Grand Village	No	No	3.8/5	166	Basavanagudi	Casual Dining	North Indian, Rajasthani	600	Buffet	Banashankari

In [8]:
```
df.info()
```

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

In [9]:
```
# DROPPING DUPLICATES
```

In [10]:
```
df.duplicated().sum()
```

Out[10]:
```
108
```

In [11]:
```
df.drop_duplicates(inplace=True)
df.shape
```

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

In [12]:
```
# CLEANING RATE COLUMN
```

In [13]:
```
df['rate'].unique()
```

Out[13]:
```
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 [14]:
```
value='2.0/5'
x=value.split("/")
x[0]
```

Out[14]:
```
'2.0'
```

In [15]:
```
# REMOVING "NEW","_" AND "/5" FROM RATE COLUMN
```

In [16]:
```
def handlerate(value):
    if (value=='NEW' or value=='-'):
        return np.nan
    else:
        value=str(value).split('/')
        value=value[0]
        return float(value)

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

Out[16]:
```
0    4.1
1    4.1
2    3.8
3    3.7
4    3.8
Name: rate, dtype: float64
```

In [17]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51609 entries, 0 to 51716
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   name                         51609 non-null  object
 1   online_order                 51609 non-null  object
 2   book_table                   51609 non-null  object
 3   rate                         41590 non-null  float64
 4   votes                        51609 non-null  int64
 5   location                     51588 non-null  object
 6   rest_type                    51382 non-null  object
 7   cuisines                     51564 non-null  object
 8   approx_cost(for two people)  51265 non-null  object
 9   listed_in(type)              51609 non-null  object
 10  listed_in(city)              51609 non-null  object
dtypes: float64(1), int64(1), object(9)
memory usage: 4.7+ MB
```

In [18]:
```
(df.isna().sum()/df.shape[0])*100
```

Out[18]:
```
name                           0.000000
online_order                   0.000000
book_table                     0.000000
rate                          19.413281
votes                          0.000000
location                       0.040691
rest_type                      0.439846
cuisines                       0.087194
approx_cost(for two people)    0.666550
listed_in(type)                0.000000
listed_in(city)                0.000000
dtype: float64
```

In [19]:
```
# FILLING NULL VALUES IN RATE COLUMN WITH MEAN
```

In [20]:
```
df['rate'].fillna(df['rate'].mean(),inplace = True)
df['rate'].isnull().sum()
```

Out[20]:
```
0
```

In [21]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51609 entries, 0 to 51716
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   name                         51609 non-null  object
 1   online_order                 51609 non-null  object
 2   book_table                   51609 non-null  object
 3   rate                         51609 non-null  float64
 4   votes                        51609 non-null  int64
 5   location                     51588 non-null  object
 6   rest_type                    51382 non-null  object
 7   cuisines                     51564 non-null  object
 8   approx_cost(for two people)  51265 non-null  object
 9   listed_in(type)              51609 non-null  object
 10  listed_in(city)              51609 non-null  object
dtypes: float64(1), int64(1), object(9)
memory usage: 4.7+ MB
```

In [22]:
```
# DROPPING NUL VALUES
```

In [23]:
```
df.dropna(inplace = True)
df.head()
```

Out[23]:

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

In [24]:
```
df['location'].unique()
```

Out[24]:
```
array(['Banashankari', 'Basavanagudi', 'Mysore Road', 'Jayanagar',
       'Kumaraswamy Layout', 'Rajarajeshwari Nagar', 'Vijay Nagar',
       'Uttarahalli', 'JP Nagar', 'South Bangalore', 'City Market',
       'Nagarbhavi', 'Bannerghatta Road', 'BTM', 'Kanakapura Road',
       'Bommanahalli', 'CV Raman Nagar', 'Electronic City', 'HSR',
       'Marathahalli', 'Wilson Garden', 'Shanti Nagar',
       'Koramangala 5th Block', 'Koramangala 8th Block', 'Richmond Road',
       'Koramangala 7th Block', 'Jalahalli', 'Koramangala 4th Block',
       'Bellandur', 'Sarjapur Road', 'Whitefield', 'East Bangalore',
       'Old Airport Road', 'Indiranagar', 'Koramangala 1st Block',
       'Frazer Town', 'RT Nagar', 'MG Road', 'Brigade Road',
       'Lavelle Road', 'Church Street', 'Ulsoor', 'Residency Road',
       'Shivajinagar', 'Infantry Road', 'St. Marks Road',
       'Cunningham Road', 'Race Course Road', 'Commercial Street',
       'Vasanth Nagar', 'HBR Layout', 'Domlur', 'Ejipura',
       'Jeevan Bhima Nagar', 'Old Madras Road', 'Malleshwaram',
       'Seshadripuram', 'Kammanahalli', 'Koramangala 6th Block',
       'Majestic', 'Langford Town', 'Central Bangalore', 'Sanjay Nagar',
       'Brookefield', 'ITPL Main Road, Whitefield',
       'Varthur Main Road, Whitefield', 'KR Puram',
       'Koramangala 2nd Block', 'Koramangala 3rd Block', 'Koramangala',
       'Hosur Road', 'Rajajinagar', 'Banaswadi', 'North Bangalore',
       'Nagawara', 'Hennur', 'Kalyan Nagar', 'New BEL Road', 'Jakkur',
       'Rammurthy Nagar', 'Thippasandra', 'Kaggadasapura', 'Hebbal',
       'Kengeri', 'Sankey Road', 'Sadashiv Nagar', 'Basaveshwara Nagar',
       'Yeshwantpur', 'West Bangalore', 'Magadi Road', 'Yelahanka',
       'Sahakara Nagar', 'Peenya'], dtype=object)
```

In [25]:
```
df['listed_in(city)'].unique()
```

Out[25]:
```
array(['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'], dtype=object)
```

In [26]:
```
df = df.drop(['listed_in(city)'], axis = 1)
```

In [27]:
```
df['approx_cost(for two people)'].unique()
```

Out[27]:
```
array(['800', '300', '600', '700', '550', '500', '450', '650', '400',
       '900', '200', '750', '150', '850', '100', '1,200', '350', '250',
       '950', '1,000', '1,500', '1,300', '199', '80', '1,100', '160',
       '1,600', '230', '130', '50', '190', '1,700', '1,400', '180',
       '1,350', '2,200', '2,000', '1,800', '1,900', '330', '2,500',
       '2,100', '3,000', '2,800', '3,400', '40', '1,250', '3,500',
       '4,000', '2,400', '2,600', '120', '1,450', '469', '70', '3,200',
       '60', '560', '240', '360', '6,000', '1,050', '2,300', '4,100',
       '5,000', '3,700', '1,650', '2,700', '4,500', '140'], dtype=object)
```

In [28]:
```
def handlecomma(value):
    value = str(value)
    if ',' in value:
        value = value.replace(',','')
        return float(value)
    else:
        return float(value)

df['approx_cost(for two people)'] = df['approx_cost(for two people)'].apply(handlecomma)
df['approx_cost(for two people)'].unique()
```

Out[28]:
```
array([ 800.,  300.,  600.,  700.,  550.,  500.,  450.,  650.,  400.,
        900.,  200.,  750.,  150.,  850.,  100., 1200.,  350.,  250.,
        950., 1000., 1500., 1300.,  199.,   80., 1100.,  160., 1600.,
        230.,  130.,   50.,  190., 1700., 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 [29]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51042 entries, 0 to 51716
Data columns (total 10 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   name                         51042 non-null  object
 1   online_order                 51042 non-null  object
 2   book_table                   51042 non-null  object
 3   rate                         51042 non-null  float64
 4   votes                        51042 non-null  int64
 5   location                     51042 non-null  object
 6   rest_type                    51042 non-null  object
 7   cuisines                     51042 non-null  object
 8   approx_cost(for two people)  51042 non-null  float64
 9   listed_in(type)              51042 non-null  object
dtypes: float64(2), int64(1), object(7)
memory usage: 4.3+ MB
```

In [30]:
```
df.head()
```

Out[30]:

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

In [31]:
```
# CLEANING REST TYPE COLUMN
```

In [32]:
```
rest_type = df['rest_type'].value_counts(ascending = False)
rest_type
```

Out[32]:
```
Quick Bites                   19010
Casual Dining                 10253
Cafe                           3682
Delivery                       2574
Dessert Parlor                 2242
                              ...
Dessert Parlor, Kiosk             2
Food Court, Beverage Shop         2
Dessert Parlor, Food Court        2
Quick Bites, Kiosk                1
Sweet Shop, Dessert Parlor        1
Name: rest_type, Length: 93, dtype: int64
```

In [33]:
```
rest_type_lessthan1000 = rest_type[rest_type<1000]
rest_type_lessthan1000
```

Out[33]:
```
Beverage Shop                 863
Bar                           686
Food Court                    616
Sweet Shop                    468
Bar, Casual Dining            411
                             ...
Dessert Parlor, Kiosk           2
Food Court, Beverage Shop       2
Dessert Parlor, Food Court      2
Quick Bites, Kiosk              1
Sweet Shop, Dessert Parlor      1
Name: rest_type, Length: 85, dtype: int64
```

In [34]:
```
def handle_rest_type(value):
    if(value in rest_type_lessthan1000):
        return 'others'
    else:
        return value

df['rest_type'] = df['rest_type'].apply(handle_rest_type)
df['rest_type'].value_counts()
```

Out[34]:
```
Quick Bites           19010
Casual Dining         10253
others                 9003
Cafe                   3682
Delivery               2574
Dessert Parlor         2242
Takeaway, Delivery     2008
Bakery                 1140
Casual Dining, Bar     1130
Name: rest_type, dtype: int64
```

In [35]:
```
# cleaning location column
```

In [36]:
```
df['location'].value_counts
```

Out[36]:
```
<bound method IndexOpsMixin.value_counts of 0                      Banashankari
1                      Banashankari
2                      Banashankari
3                      Banashankari
4                      Basavanagudi
                   ...
51712                   Whitefield
51713                   Whitefield
51714                   Whitefield
51715    ITPL Main Road, Whitefield
51716    ITPL Main Road, Whitefield
Name: location, Length: 51042, dtype: object>
```

In [38]:
```
location = df['location'].value_counts(ascending = False)

location_lessthan300 = location[location<300]
```

In [39]:
```
location = df['location'].value_counts(ascending = False)

location_lesssthan300 = location[location<300]

def handle_location(value):
    if(value in location_lessthan300):
        return 'others'
    else:
        return value

df['location'] = df['location'].apply(handle_location)
df['location'].value_counts()
```

Out[39]:
```
BTM                    5056
others                 4954
HSR                    2494
Koramangala 5th Block  2479
JP Nagar               2218
Whitefield             2105
Indiranagar            2026
Jayanagar              1916
Marathahalli           1805
Bannerghatta Road      1609
Bellandur              1268
Electronic City        1246
Koramangala 1st Block  1236
Brigade Road           1210
Koramangala 7th Block  1174
Koramangala 6th Block  1127
Sarjapur Road          1047
Koramangala 4th Block  1017
Ulsoor                 1011
Banashankari            902
MG Road                 893
Kalyan Nagar            841
Richmond Road           803
Malleshwaram            721
Frazer Town             714
Basavanagudi            684
Residency Road          671
Brookefield             656
New BEL Road            644
Banaswadi               640
Kammanahalli            639
Rajajinagar             591
Church Street           566
Lavelle Road            518
Shanti Nagar            508
Shivajinagar            498
Cunningham Road         490
Domlur                  482
Old Airport Road        437
Ejipura                 433
Commercial Street       370
St. Marks Road          343
Name: location, dtype: int64
```

In [ ]:
```
# CLEANING CUSINES COLUMN
```

In [40]:
```
cuisines = df['cuisines'].value_counts(ascending = False)
cuisines_lessthan100 = cuisines[cuisines<300]

def handle_cuisines(value):
    if(value in cuisines_lessthan100):
        return 'others'
    else:
        return value

df['cuisines'] = df['cuisines'].apply(handle_cuisines)
df['cuisines'].value_counts()
```

Out[40]:
```
others                                  35170
North Indian                             2852
North Indian, Chinese                    2351
South Indian                             1820
Biryani                                   903
Bakery, Desserts                          898
Fast Food                                 796
Desserts                                  754
Cafe                                      725
South Indian, North Indian, Chinese       724
Bakery                                    649
Chinese                                   552
Ice Cream, Desserts                       415
Chinese, North Indian                     405
Mithai, Street Food                       363
Desserts, Ice Cream                       349
North Indian, Chinese, Biryani            345
South Indian, North Indian                337
North Indian, South Indian                329
North Indian, South Indian, Chinese       305
Name: cuisines, dtype: int64
```

In [41]:
```
df.head()
```

Out[41]:

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

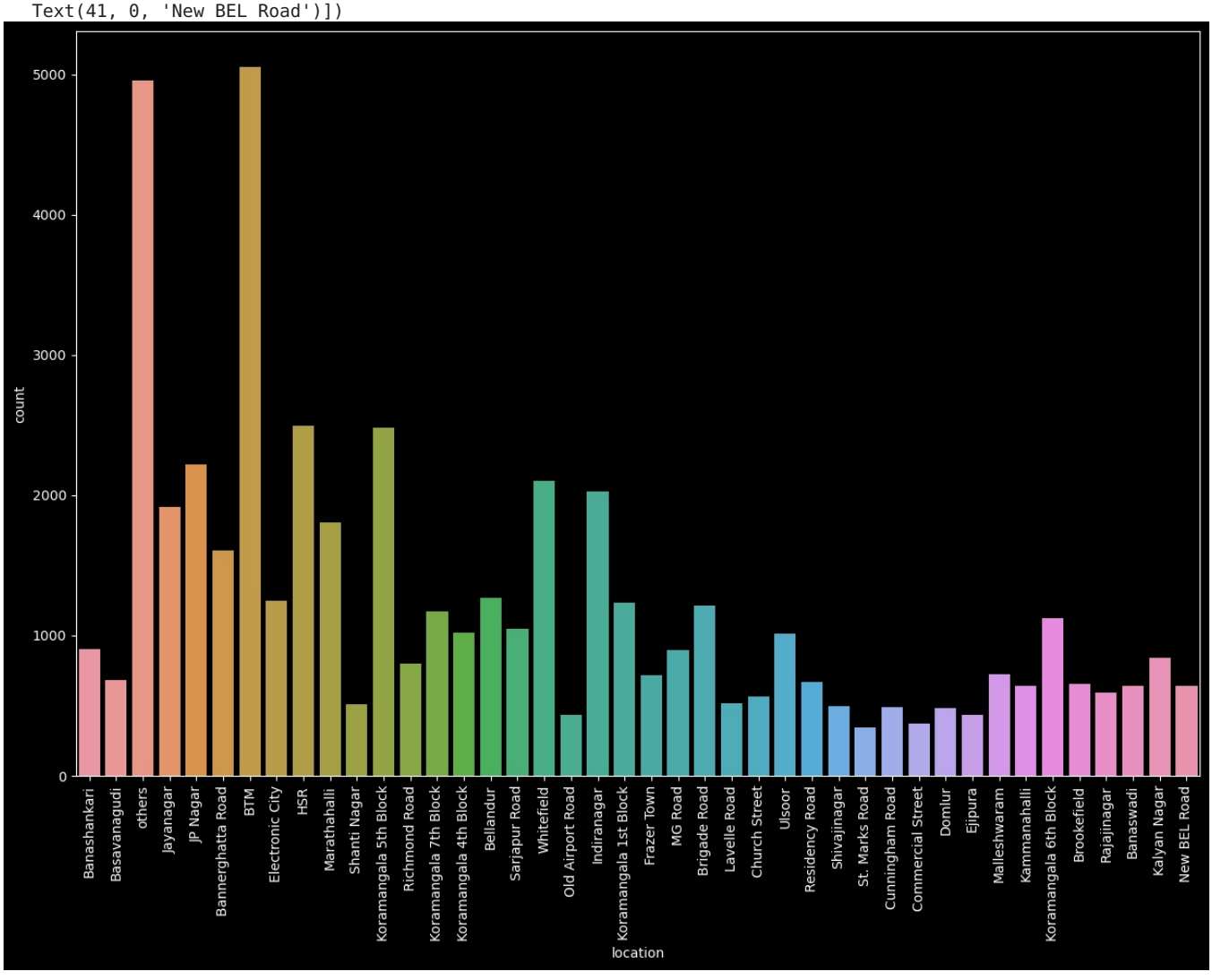
In [ ]:
```
# DATA IS CLEAN,LETS JUMP TO VISUALIZATION
```

In [ ]:
```
# COUNT PLOT OF VARIOOUS LOCATIONS
```

In [42]:
```
plt.figure(figsize = (15,10))
ax = sns.countplot(df['location'])
plt.xticks(rotation=90)
```

```
C:\Users\rajesh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the 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[42]:
```
(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, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41]),
 [Text(0, 0, 'Banashankari'),
  Text(1, 0, 'Basavanagudi'),
  Text(2, 0, 'others'),
  Text(3, 0, 'Jayanagar'),
  Text(4, 0, 'JP Nagar'),
  Text(5, 0, 'Bannerghatta Road'),
  Text(6, 0, 'BTM'),
  Text(7, 0, 'Electronic City'),
  Text(8, 0, 'HSR'),
  Text(9, 0, 'Marathahalli'),
  Text(10, 0, 'Shanti Nagar'),
  Text(11, 0, 'Koramangala 5th Block'),
  Text(12, 0, 'Richmond Road'),
  Text(13, 0, 'Koramangala 7th Block'),
  Text(14, 0, 'Koramangala 4th Block'),
  Text(15, 0, 'Bellandur'),
  Text(16, 0, 'Sarjapur Road'),
  Text(17, 0, 'Whitefield'),
  Text(18, 0, 'Old Airport Road'),
  Text(19, 0, 'Indiranagar'),
  Text(20, 0, 'Koramangala 1st Block'),
  Text(21, 0, 'Frazer Town'),
  Text(22, 0, 'MG Road'),
  Text(23, 0, 'Brigade Road'),
  Text(24, 0, 'Lavelle Road'),
  Text(25, 0, 'Church Street'),
  Text(26, 0, 'Ulsoor'),
  Text(27, 0, 'Residency Road'),
  Text(28, 0, 'Shivajinagar'),
  Text(29, 0, 'St. Marks Road'),
  Text(30, 0, 'Cunningham Road'),
  Text(31, 0, 'Commercial Street'),
  Text(32, 0, 'Domlur'),
  Text(33, 0, 'Ejipura'),
  Text(34, 0, 'Malleshwaram'),
  Text(35, 0, 'Kammanahalli'),
  Text(36, 0, 'Koramangala 6th Block'),
  Text(37, 0, 'Brookefield'),
  Text(38, 0, 'Rajajinagar'),
  Text(39, 0, 'Banaswadi'),
  Text(40, 0, 'Kalyan Nagar'),
  Text(41, 0, 'New BEL Road')])
```

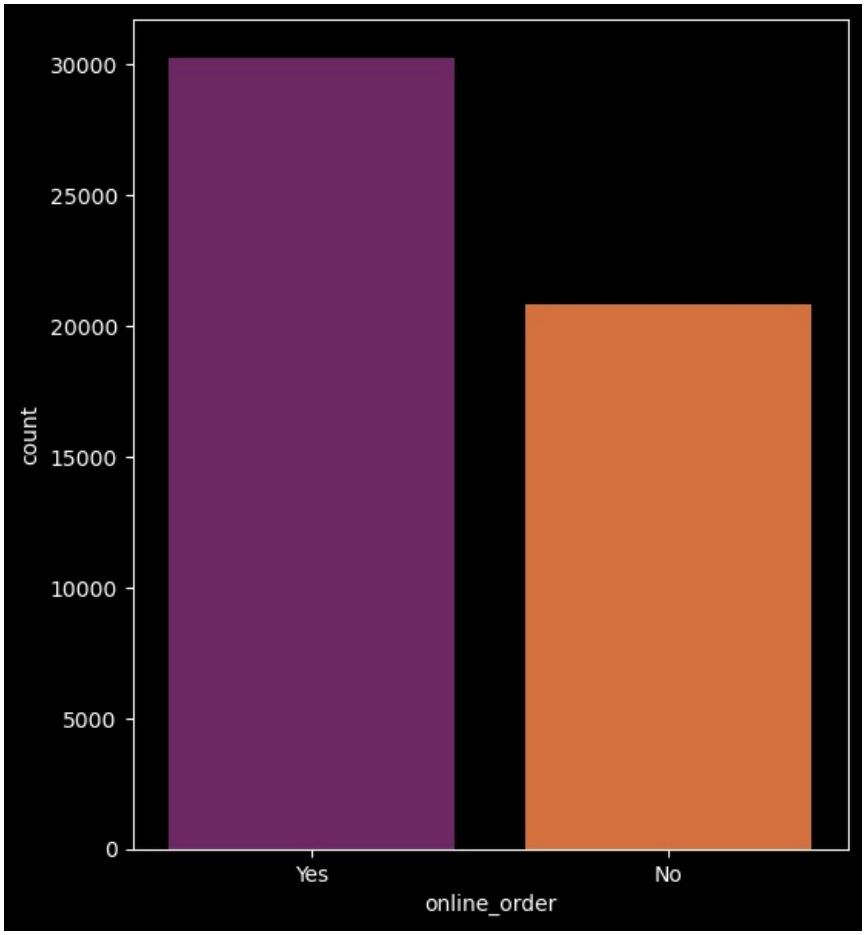


In [ ]:
```
# VISUALIZING ONLINE ORDER
```

In [43]:
```
plt.figure(figsize = (6,7))
sns.countplot(df['online_order'],palette = 'inferno')
```

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

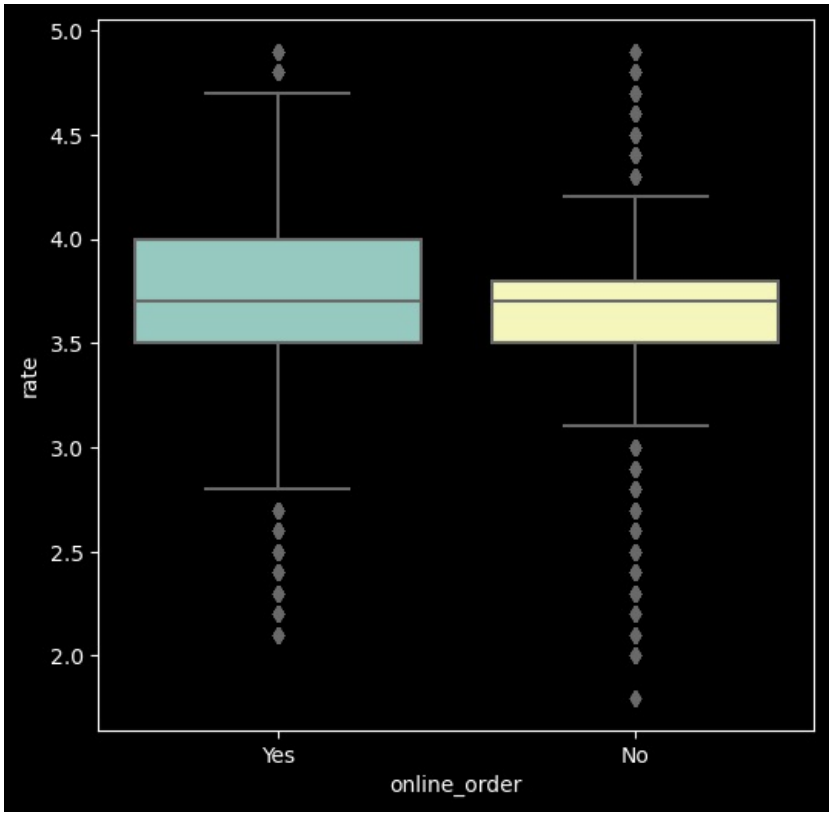
Out[43]:
```
<AxesSubplot:xlabel='online_order', ylabel='count'>
```



In [ ]:
```
# VISUALIZING ONLINE ORDER VS RATE
```

In [44]:
```
plt.figure(figsize = (6,6))
sns.boxplot(x = 'online_order', y = 'rate' , data = df)
```

Out[44]:
```
<AxesSubplot:xlabel='online_order', ylabel='rate'>
```



In [ ]: