Name: Navneet Yadav

Roll no. 2301560044

Program: MCA-I

Subject: AI-ML

Question 2

Data Aalysis and Cleaning

- GitHub: https://github.com/navYadav20/AI-ML (https://github.com/navYadav20/AI-ML)

In [2]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use("dark_background")
```

In [4]:
```
df = pd.read_csv("D:/ai-ml assignment/assignment 3/zomato.csv")
```

In [4]:
```
df.head()
```

Out[4]:

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

In [5]:
```
print(f" Rows in dataset : {df.shape[0]}\n Columns in dataset : {df.shape[1
```

```
 Rows in dataset : 51717
 Columns in dataset : 17
```

In [6]:
```
print("Columns in dataset :\n",df.columns)
```

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

Make a copy of file

In [7]:
```
df1 = df.copy()
```

Drop not important Columns like URL, address, phone numbers, menu_items, dish_liked and review list

In [8]:
```
df1.drop(['url', 'address',"phone",'dish_liked','menu_item','reviews_list']

df1.head()
```

Out[8]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	appro t
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 [9]:
```
df1.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
```

Check null values

In [10]:
```
null_values = {i:df1[i].isnull().sum() for i in df1.columns}

null_values
```

Out[10]:
```
{'name': 0,
 'online_order': 0,
 'book_table': 0,
 'rate': 7775,
 'votes': 0,
 'location': 21,
 'rest_type': 227,
 'cuisines': 45,
 'approx_cost(for two people)': 346,
 'listed_in(type)': 0,
 'listed_in(city)': 0}
```

In [11]:
```
# cleaning rate columns
print("There are {} values are unique in Rate Columns".format(len(df1["rate

df1["rate"].unique()
```

```
There are 65 values are unique in Rate Columns
```

Out[11]:
```
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 [12]:
```
def rate(data):
    if data == "NEW" or data == "-":
        return np.nan
    else:
        data = str(data).split("/")
        data = data[0]
        return float(data)
```

In [13]:
```
df1["rate"] = df1["rate"].apply(rate)

# Now replace null value with mode

df1["rate"] = df1["rate"].fillna(df1["rate"].mode()[0])
df1.head()
```

Out[13]:

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

In [14]:
```
# There are less null values in other columns so we drop null values

df1.dropna(inplace = True)
df1.head()
```

Out[14]:

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

In [15]:
```
# Rename columns

df1.rename(columns = {"approx_cost(for two people)":"Cost2plate","listed_in

df1.head()
```

Out[15]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	Cost2
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	
1	Spice Elephant	Yes	No	4.1	787	Banashankari	Casual Dining	Chinese, North Indian, Thai	
2	San Churro Cafe	Yes	No	3.8	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	
3	Addhuri Udupi Bhojana	No	No	3.7	88	Banashankari	Quick Bites	South Indian, North Indian	
4	Grand Village	No	No	3.8	166	Basavanagudi	Casual Dining	North Indian, Rajasthani	

In [ ]:
```
# location and listed_in(city) containg same values so we drop one

df1.drop("listed_in(city)", axis = 1, inplace = True)
```

In [18]:
```
df1.head()
```

Out[18]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	Cost2
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	
1	Spice Elephant	Yes	No	4.1	787	Banashankari	Casual Dining	Chinese, North Indian, Thai	
2	San Churro Cafe	Yes	No	3.8	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	
3	Addhuri Udupi Bhojana	No	No	3.7	88	Banashankari	Quick Bites	South Indian, North Indian	
4	Grand Village	No	No	3.8	166	Basavanagudi	Casual Dining	North Indian, Rajasthani	

In [19]:
```
df1["Cost2plate"].unique()
```

Out[19]:
```
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 [20]:
```
# in Cost2plate columns most of value containg comma

def remove_comma(data):
    data = data.replace(",","")
    return float(data)
```

In [21]:
```
df1["Cost2plate"] = df1["Cost2plate"].apply(remove_comma)
```

In [22]:
```
# Lets see Rest Type columns

df1["rest_type"].value_counts()
```

Out[22]:
```
rest_type
Quick Bites                   19046
Casual Dining                 10273
Cafe                           3687
Delivery                       2578
Dessert Parlor                 2245
                              ...
Dessert Parlor, Kiosk             2
Food Court, Beverage Shop         2
Dessert Parlor, Food Court        2
Quick Bites, Kiosk                1
Sweet Shop, Dessert Parlor        1
Name: count, Length: 93, dtype: int64
```

In [23]:
```
rest_type = df1["rest_type"].value_counts(ascending = False)
rest_less_than_1000 = rest_type[rest_type<1000]
rest_less_than_1000
```

Out[23]:
```
rest_type
Beverage Shop                 865
Bar                           686
Food Court                    619
Sweet Shop                    468
Bar, Casual Dining            415
                             ...
Dessert Parlor, Kiosk           2
Food Court, Beverage Shop       2
Dessert Parlor, Food Court      2
Quick Bites, Kiosk              1
Sweet Shop, Dessert Parlor      1
Name: count, Length: 85, dtype: int64
```

In [28]:
```
def rest_type(data):
    if data in rest_less_than_1000:
        return "other"
    else:
        return data
```

In [29]:
```
df1["rest_type"] = df1["rest_type"].apply(rest_type)
```

In [30]:
```
df1["rest_type"].value_counts()
```

Out[30]:
```
rest_type
Quick Bites           19046
Casual Dining         10273
other                  9028
Cafe                   3687
Delivery               2578
Dessert Parlor         2245
Takeaway, Delivery     2014
Bakery                 1141
Casual Dining, Bar     1136
Name: count, dtype: int64
```

In [31]:
```
df1["location"].value_counts()
```

Out[31]:
```
location
BTM                    5071
HSR                    2496
Koramangala 5th Block  2481
JP Nagar               2219
Whitefield             2109
                       ...
West Bangalore            6
Yelahanka                 5
Jakkur                    3
Rajarajeshwari Nagar      2
Peenya                    1
Name: count, Length: 93, dtype: int64
```

In [32]:
```
location = df1["location"].value_counts(ascending = True)
location
```

Out[32]:
```
location
Peenya                    1
Rajarajeshwari Nagar      2
Jakkur                    3
Yelahanka                 5
West Bangalore            6
                       ...
Whitefield             2109
JP Nagar               2219
Koramangala 5th Block  2481
HSR                    2496
BTM                    5071
Name: count, Length: 93, dtype: int64
```

In [33]:
```
location_lessthan_300 = location[location<300]
```

In [34]:
```
def location(data):
    if data in location_lessthan_300:
        return "Other"
    else:
        return data
```

In [35]:
```
df1["location"] = df1["location"].apply(location)
```

In [36]:
```
df1["location"].value_counts()
```

Out[36]:
```
location
BTM                    5071
Other                  4962
HSR                    2496
Koramangala 5th Block  2481
JP Nagar               2219
Whitefield             2109
Indiranagar            2033
Jayanagar              1916
Marathahalli           1808
Bannerghatta Road      1611
Bellandur              1271
Electronic City        1248
Koramangala 1st Block  1237
Brigade Road           1218
Koramangala 7th Block  1176
Koramangala 6th Block  1129
Sarjapur Road          1049
Koramangala 4th Block  1017
Ulsoor                 1017
Banashankari            904
MG Road                 894
Kalyan Nagar            841
Richmond Road           804
Malleshwaram            724
Frazer Town             720
Basavanagudi            684
Residency Road          674
Brookefield             656
Banaswadi               645
New BEL Road            644
Kammanahalli            640
Rajajinagar             591
Church Street           569
Lavelle Road            523
Shanti Nagar            511
Shivajinagar            499
Cunningham Road         491
Domlur                  482
Old Airport Road        437
Ejipura                 434
Commercial Street       370
St. Marks Road          343
Name: count, dtype: int64
```

Cleaning Cuisines Columns

In [46]:
```
cuisines = df1["cuisines"].value_counts(ascending = False)
cuisines
```

Out[46]:
```
cuisines
Other                                    26220
North Indian                              2858
North Indian, Chinese                     2355
South Indian                              1822
Biryani                                    906
                                         ...
South Indian, Chinese, North Indian        105
North Indian, Mughlai, Chinese             104
South Indian, Fast Food                    104
Italian, Pizza                             102
North Indian, Chinese, Seafood             102
Name: count, Length: 70, dtype: int64
```

In [39]:
```
cuisines_lessthan_100 = cuisines[cuisines<100]
```

In [40]:
```
def cuisines(data):
    if data in cuisines_lessthan_100:
        return "Other"
    else:
        return data
```

In [41]:
```
df1["cuisines"] = df1["cuisines"].apply(cuisines)
```

In [42]:
```
len(df1["cuisines"].value_counts())
```

Out[42]:
```
70
```

In [43]:
```
df1.head()
```

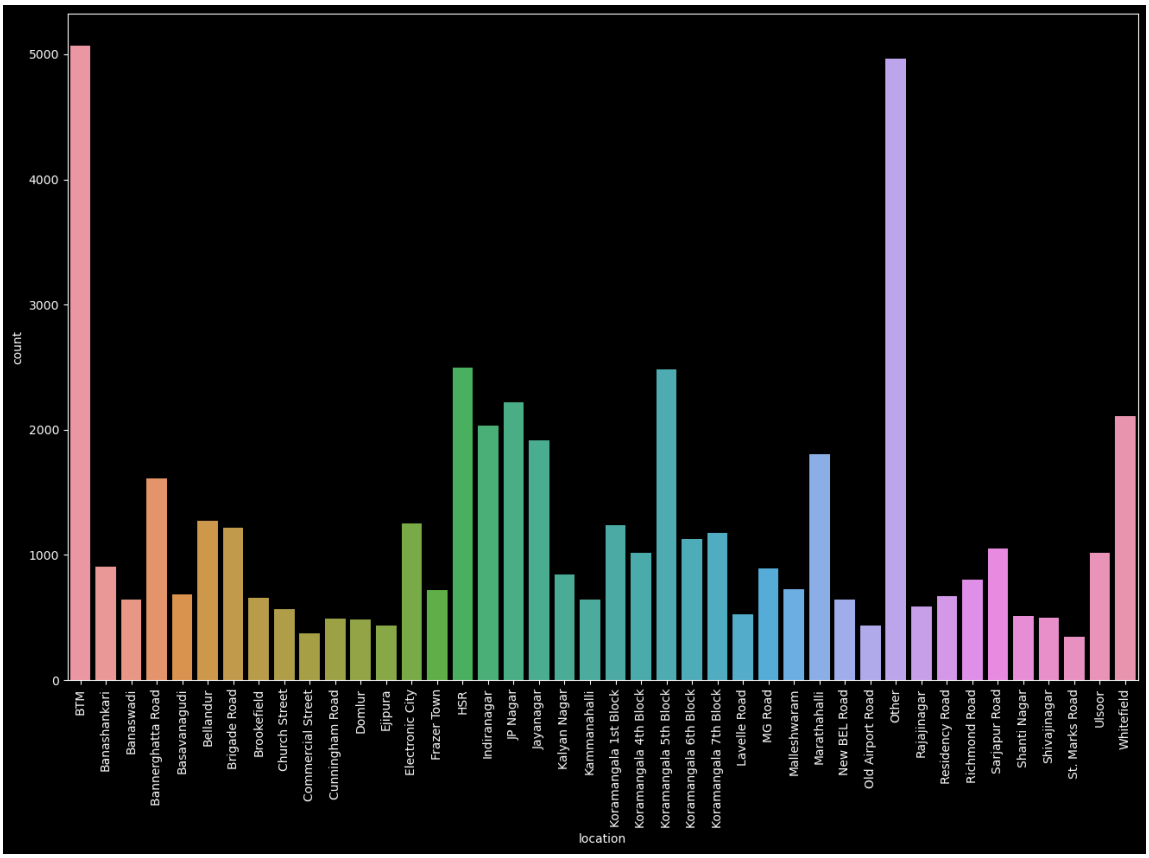
Out[43]:

	name	online_order	book_table	rate	votes	location	rest_type	cuisines	Cost2pl
0	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	80
1	Spice Elephant	Yes	No	4.1	787	Banashankari	Casual Dining	Other	80
2	San Churro Cafe	Yes	No	3.8	918	Banashankari	other	Other	80
3	Addhuri Udupi Bhojana	No	No	3.7	88	Banashankari	Quick Bites	South Indian, North Indian	30
4	Grand Village	No	No	3.8	166	Basavanagudi	Casual Dining	Other	60

Visualization

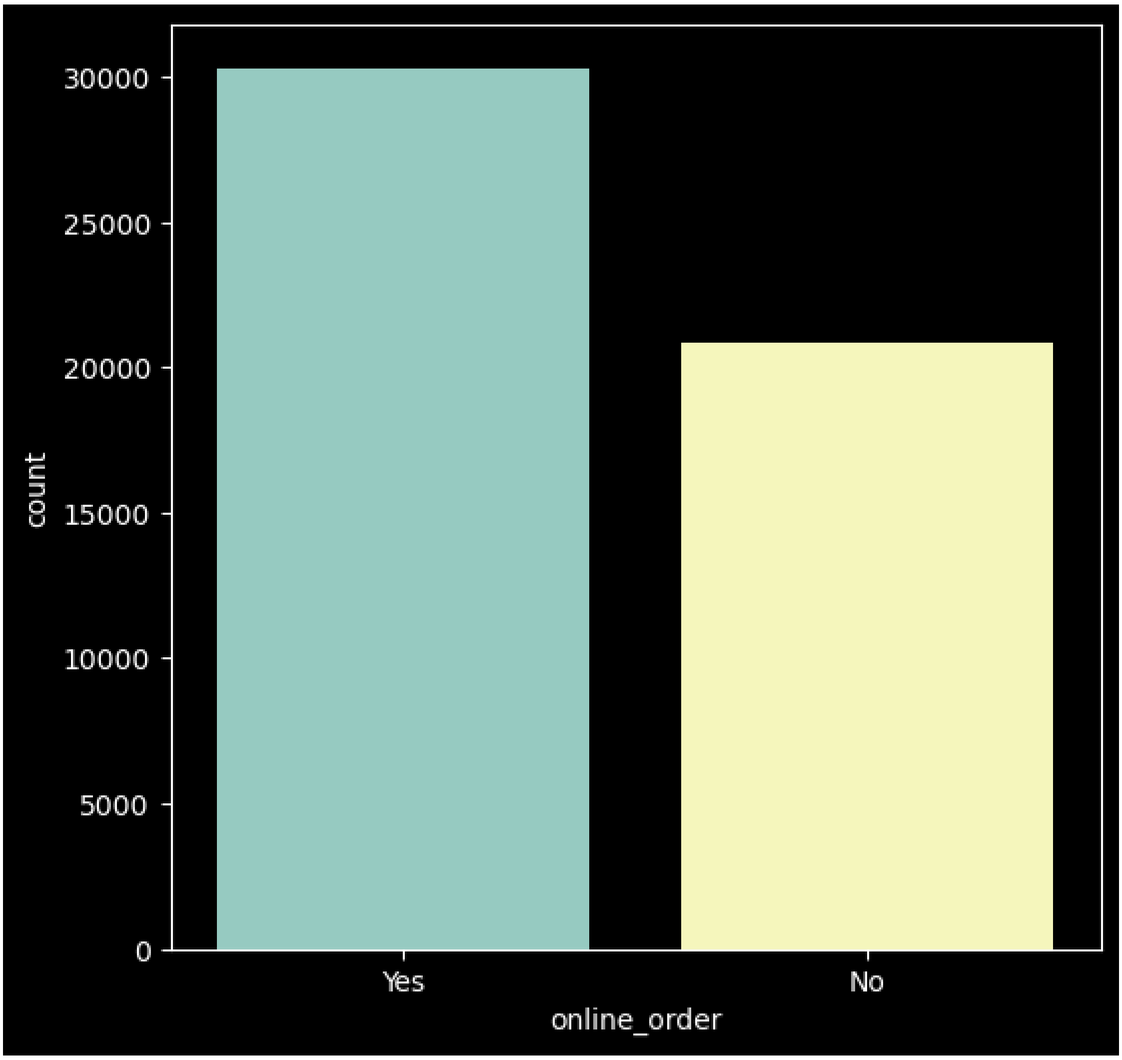
In [60]:
```
# Let's Analyse location vs number of restarant

plt.figure(figsize = (16,10))
ax = sns.countplot(data=df1,x="location")
plt.xticks(rotation = 90)
plt.show()
```

```
C:\Users\msuse\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\msuse\anaconda3\Lib\site-packages\seaborn\categorical.py:641: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
```



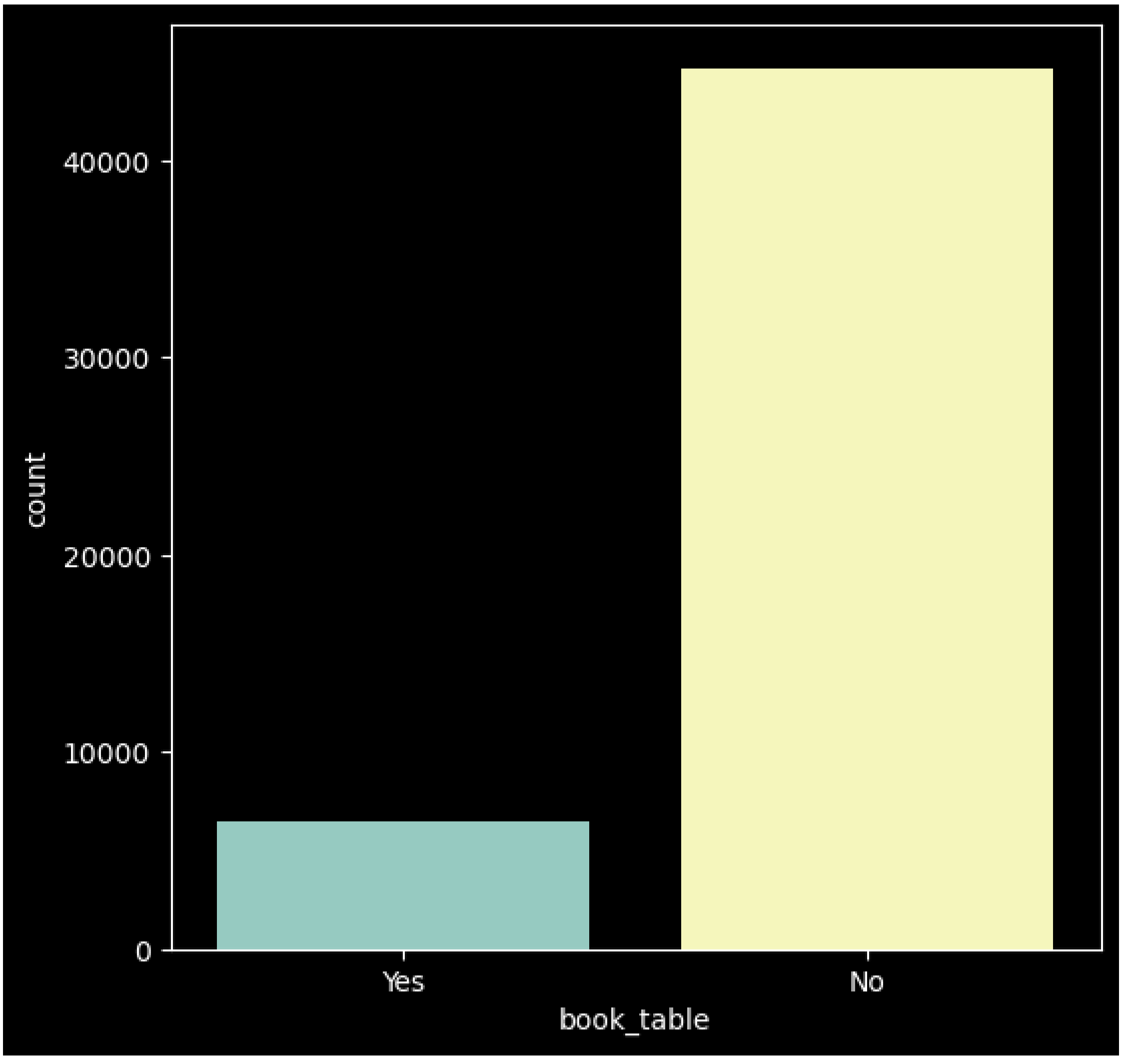
In [59]:
```
# How many have online facility or how many have not

plt.figure(figsize = (6,6))
ax = sns.countplot(data=df1,x="online_order")
plt.xticks(rotation = 0)
plt.show()
```

```
C:\Users\msuse\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Futur
eWarning: is_categorical_dtype is deprecated and will be removed in a futu
re version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\msuse\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Futur
eWarning: is_categorical_dtype is deprecated and will be removed in a futu
re version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\msuse\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Futur
eWarning: is_categorical_dtype is deprecated and will be removed in a futu
re version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



In [57]:
```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize = (6,6))
sns.countplot(data=df1,x="book_table")
plt.show()
```

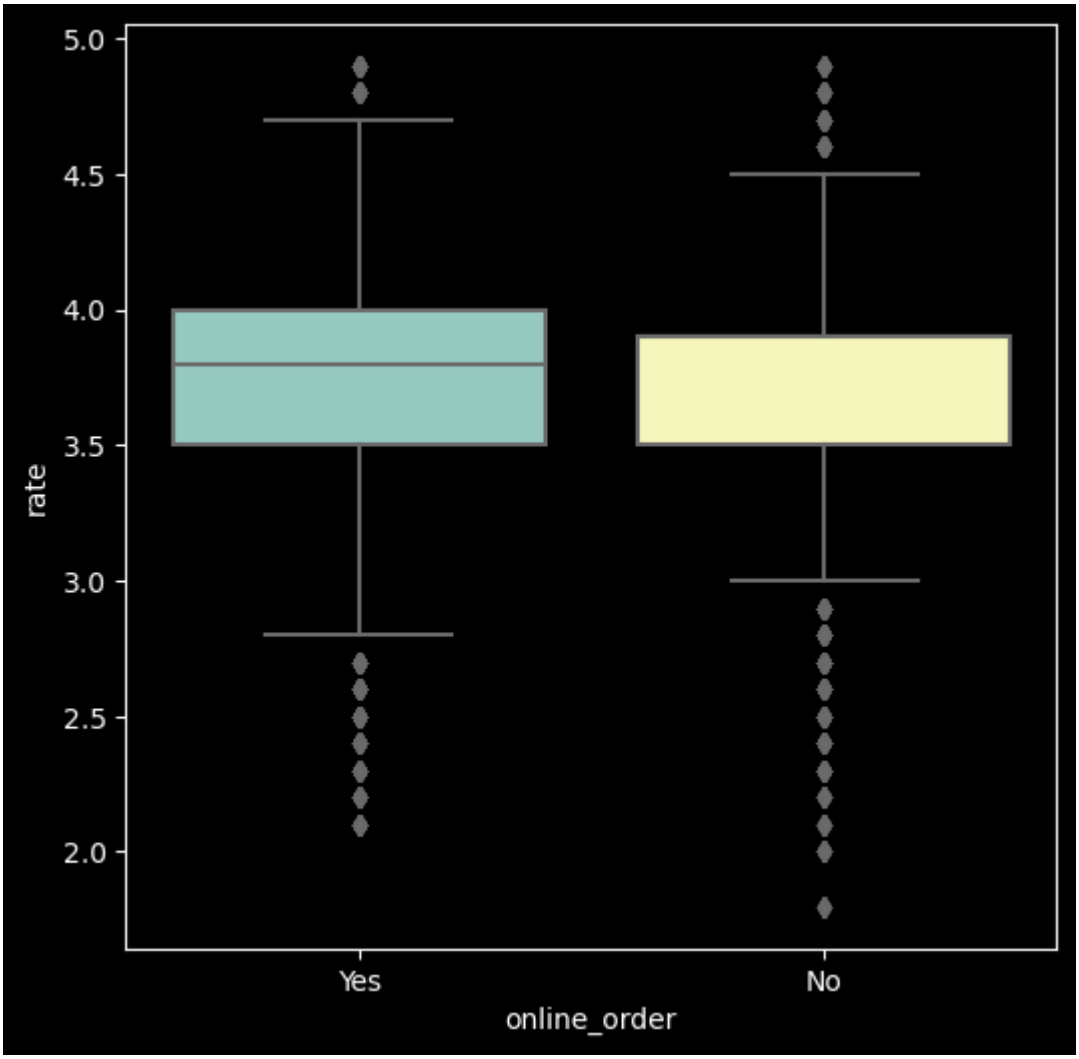
```
C:\Users\msuse\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Futur
eWarning: is_categorical_dtype is deprecated and will be removed in a futu
re version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\msuse\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Futur
eWarning: is_categorical_dtype is deprecated and will be removed in a futu
re version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\msuse\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Futur
eWarning: is_categorical_dtype is deprecated and will be removed in a futu
re version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



Online order VS Rate (Rating given by customer)

In [61]:
```
plt.figure(figsize = (6,6))
sns.boxplot(data = df1, x = "online_order",y = "rate")
plt.show()
```

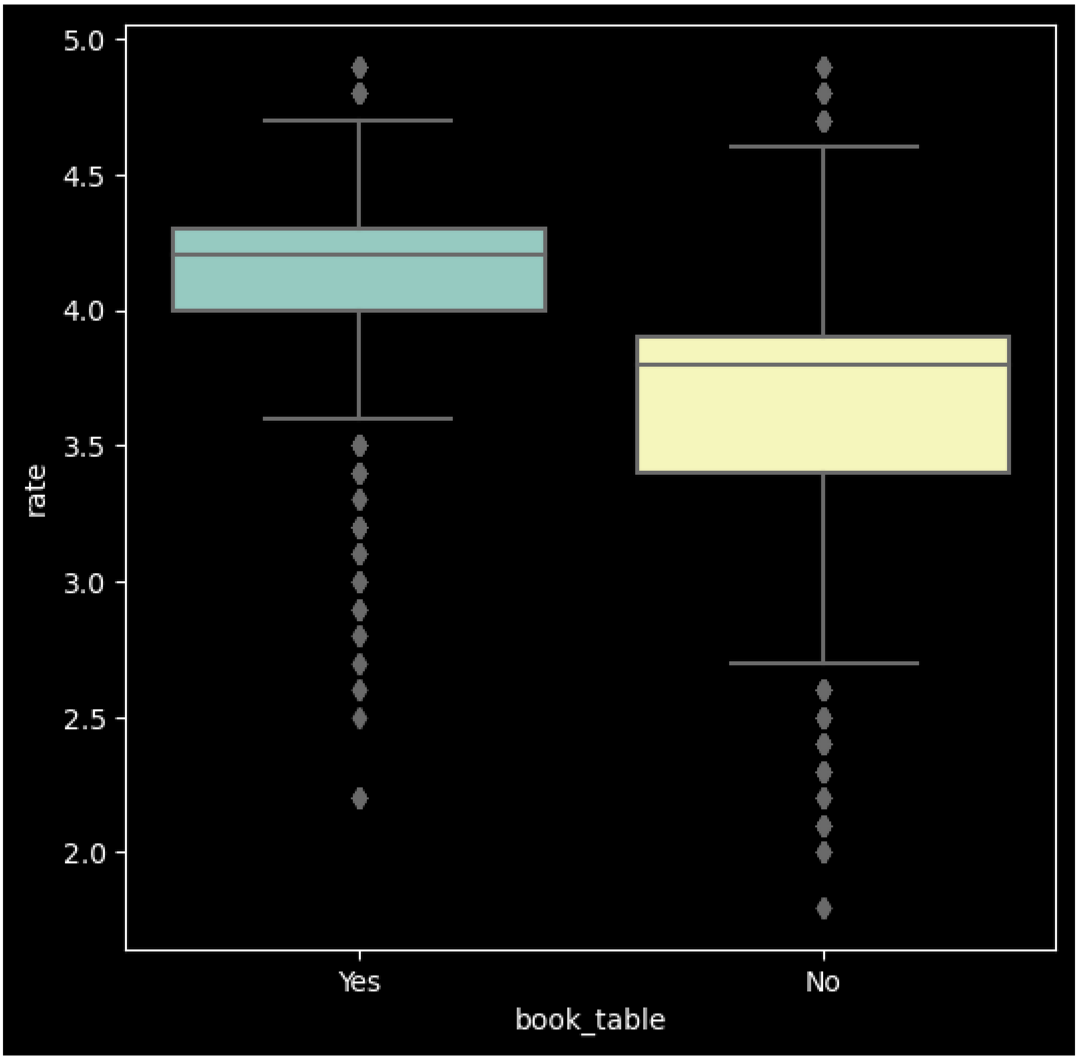
```
C:\Users\msuse\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Futur
eWarning: is_categorical_dtype is deprecated and will be removed in a futu
re version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\msuse\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Futur
eWarning: is_categorical_dtype is deprecated and will be removed in a futu
re version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



Book Table vs Rate (Rating)

In [62]:
```
plt.figure(figsize = (6,6))
sns.boxplot(data = df1, x = "book_table", y= "rate")
plt.show()

print("Here we can see who is proving book table facility they have high ra
```

```
C:\Users\msuse\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Futur
eWarning: is_categorical_dtype is deprecated and will be removed in a futu
re version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\msuse\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Futur
eWarning: is_categorical_dtype is deprecated and will be removed in a futu
re version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\msuse\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Futur
eWarning: is_categorical_dtype is deprecated and will be removed in a futu
re version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```




```
Here we can see who is proving book table facility they have high rating
```

Location vs Online order facility

In [63]:
```
df2 = df1.groupby(["location","online_order"])["name"].count()
```

```
C:\Users\msuse\AppData\Local\Temp\ipykernel_9040\1340783485.py:1: FutureWa
rning: The default of observed=False is deprecated and will be changed to
True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this war
ning.
  df2 = df1.groupby(["location","online_order"])["name"].count()
```

In [64]:
```
df1.groupby(["location","online_order"])
```

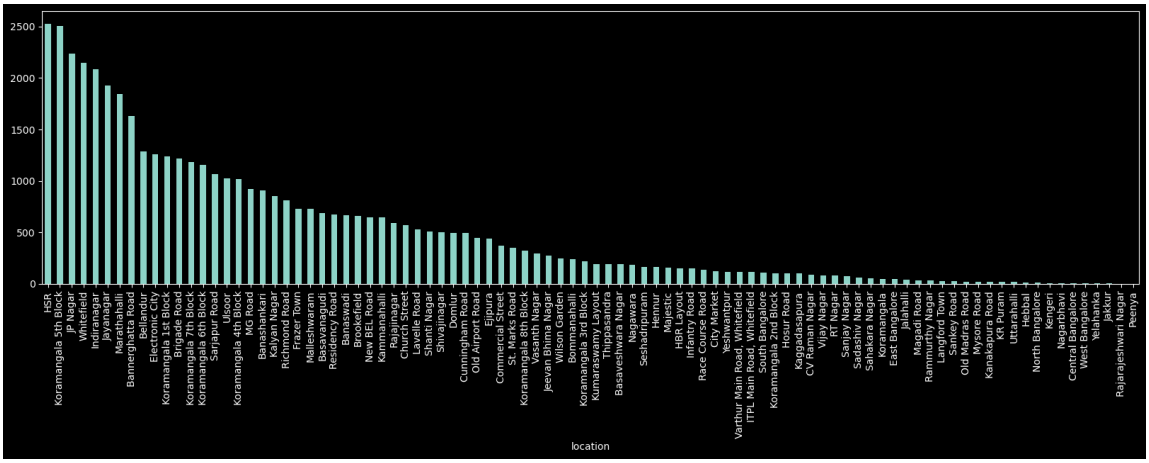
```
C:\Users\msuse\AppData\Local\Temp\ipykernel_9040\3813153573.py:1: FutureWa
rning: The default of observed=False is deprecated and will be changed to
True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this war
ning.
  df1.groupby(["location","online_order"])
```

Out[64]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001881ACF1750
>
```

Best Location

In [6]:
```
#lcoations without "other" category
plt.figure(figsize=(20,5))
df['location'].value_counts().tail(-1).plot(kind='bar')
print(f"Inference - BTM  has highest no of restaurant and also all koramang
```

```
Inference - BTM  has highest no of restaurant and also all koramangala blo
cks as combine have highest no of restaurants and can be consider as best
location
```



In [ ]:
```

```