Zomato Bengaluru Data Analysis

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('dark_background')
```

Imported pandas as pd, numpy as np, matplotlib.pyplt as plt, and seaborn as sns. Used dark background in the Kaggle kernel.

Reading CSV file

```
df =
pd.read_csv('/kaggle/input/zomato-bangalore-restaurants/zomato.
csv')
df.head()
```

Provided the path of the file where my CSV file is present. Using the head() function, I was able to see the first 5 rows of the dataset.

```
df.shape
```

It provides the number of columns and rows present in the dataset. In this dataset, there are 51,717 rows and 17 columns.

```
df.columns
```

It gives the list of columns present in the dataset. The list of columns are 'url', 'address', 'name', 'online_order', 'book_table', 'rate', 'votes','phone', 'location', 'rest_type', 'dish_li'cuisines','approx_cost(for two people)', 'reviews_list', 'menu_item', 'listed_in(type)', 'listed_in(city)'.

```
df =
df.drop(['url','address','phone','menu_item','dish_liked','revi
ews_list'],axis=1)
df.head()
```

In this ‘url’, ‘address’, ‘phone’, ‘menu_item’, ‘dish_liked’, and ‘reviews_list’ columns are not important, so I delete these columns.

```
df.info()
```

Provides information about the data like it will show how many non-null values are there in each column and the data type of the data in each column.

Dropping Duplicates

```
df.drop_duplicates(inplace=True)
df.shape
```

Dropped the duplicate values and we got the new shape of the data which is 51,609 rows and 11 columns.

Cleaning Rate Column

```
df['rate'].unique()
```

In the rate column, there is a '/5' and we don't need this. When I run the code above, I was able to see 'NEW', and '-'.

Removing "NEW", "-" and "/5" from Rate Column

```
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()
```

If the value is 'NEW' or '-' then it will return a null value because we have different methods to deal with the null values. Otherwise, split the string of value and '/' and return the float of the first part of the value so that we will get values like 3.1, 4.5….

Filling Null Values in Rate Column with Mean

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

There are 10019 null values in the rate column. So, I filled these null values with the mean of the rate column. inplace = True makes the applied changes in the original data frame.

```
df.info()
```

We found that there are very less null values in the remaining columns so, we will remove those null values.

Dropping Null Values

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

Now, we removed all the null values and applied our changes to the original data frame. Since the ‘approx_cost(for two people)’, and listed_in(type) columns are lengthy we will rename these columns.

```
df.rename(columns={'approx_cost(for two
people)':'Cost2plates','listed_in(type)':'Type'},inplace=True)
df.head()
```

We renamed ‘approx_cost(for two people)’ and ‘listed_in(type)’ columns with ‘Cost2plates’, and ‘listed_in(type)’ .

```
df['location'].unique()
```

We found that there are a lot of unique location values.

```
df[‘listed_in(city)’].unique()
```

Compared to the previous unique location values, the unique listed_in(city) values are less but both represent similar data like which city, and location. So, let's drop the listed_in(city) column.

Listed_in(city) and location, both are there, let's keep only one

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

We dropped the ‘listed_in(city)’ column from the data frame.

Let's see the unique values of the ‘Cost2plates’ column.

```
df['Cost2plates'].unique()
```

If we observe the unique values in the ‘Cost2plates’ columns, there is a ‘,’ in the numbers which restricts the values from converting into integers. Let's create a function to remove this comma.

Removing ‘,’ from Cost2plates Column

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

df['Cost2plates']=df['Cost2plates'].apply(handlecomma)
df['Cost2plates'].unique()
```

Now we got the float type of data in the ‘Cost2plates’ column. Values in the ‘Cost2plates’ column are clean now.

Cleaning Rest Type Column

```
rest_types=df['rest_type'].value_counts(ascending=False)
rest_types
```

Here also there are a lot of restaurants and restaurants with fewer numbers like 1,2 and below 1000, we will merge them as others. We assigned the ‘rest_types’ variable to the value counts of the restaurants.

```
rest_types_lessthan1000=rest_types[rest_types<1000]
rest_types_lessthan1000
```

rest_types less than 1000 will be stored in the ‘rest_types_lessthan1000’ variable.

Making Rest Types less than 1000 in frequency as others

```
def handle_rest_type(value):
    if value in rest_types_lessthan1000:
        return 'others'
    else:
        return value

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

After merging the restaurant's rest types with a frequency less than 1000 as ‘others’, we will have the other's frequency as 9003.

Cleaning Location Column

```
df['location'].value_counts()
```

Gives the value count of the unique location value.

```
location = df['location'].value_counts(ascending=False)
location_lessthan300 = location[location<300]
```

location variable stores the value counts of the ‘location’ value. location_lessthan300 variable stores the location less than 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()
```

If the value is in locatiion_lessthan300 then, it returns as ‘others’ otherwise returns the value. For that, we merged location value frequency less than 300 as ‘others’.

Cleaning Cuisines Column

```
cuisines = df['cuisines'].value_counts()
cuisines_lessthan100 = cuisines[cuisines<100]
```

Similarly, we stored value counts of the ‘cuisines’ column in the cuisines variable. We stored cuisines count less than 100 in the cuisines_lessthan100 variable.

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

If the value is in cuisines_lessthan100, it returns as ‘others’, otherwise it returns as value.

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

Like this, we clustered cuisines frequency less than 100 as ‘others’ using the handle_cuisines() function.

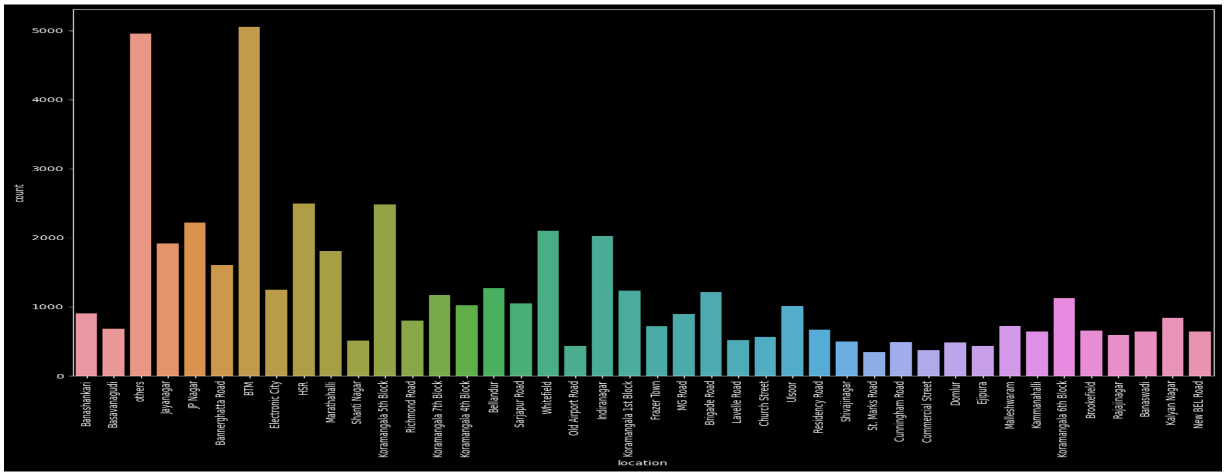
```
df['Type'].value_counts()
```

The ‘Type’ column is already clean and it has less unique so we cannot merge the less frequency value as others like how we did previously.

The data is Clean, Let’s jump to Visualization

Count Plot of Various Locations

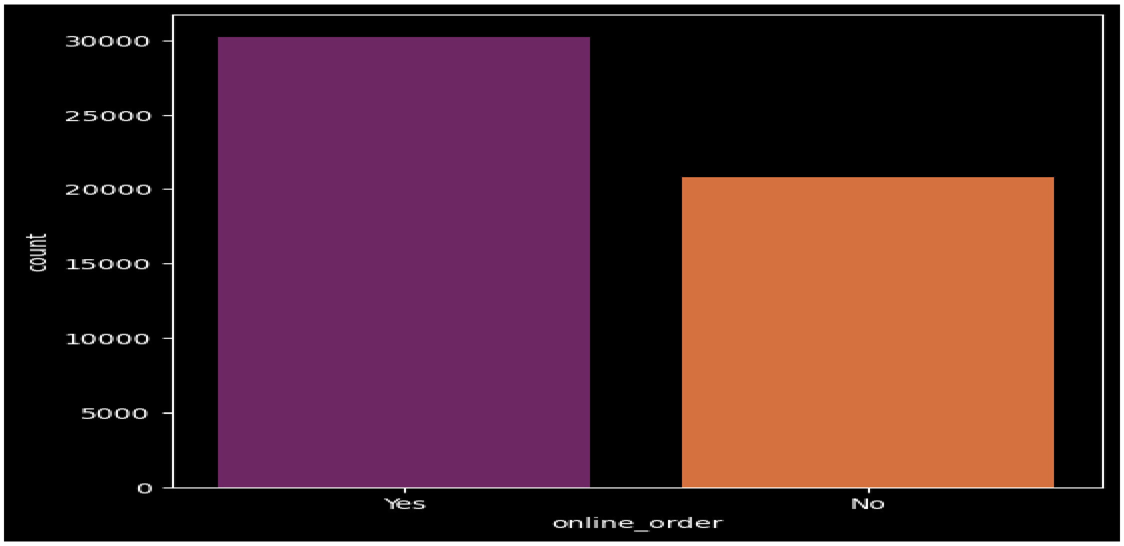
```
plt.figure(figsize = (16,10))
ax = sns.countplot(x=df['location'])
plt.xticks(rotation=90)
```



Created a count plot with ‘location’ as the x-axis. BTM got the highest count. So, we should not open a restaurant at BTM(Since it already has a lot of restaurants). We open a restaurant at locations like Old Airport Road, and St. Marks Road(since these have low restaurants).

Visualising Online Order

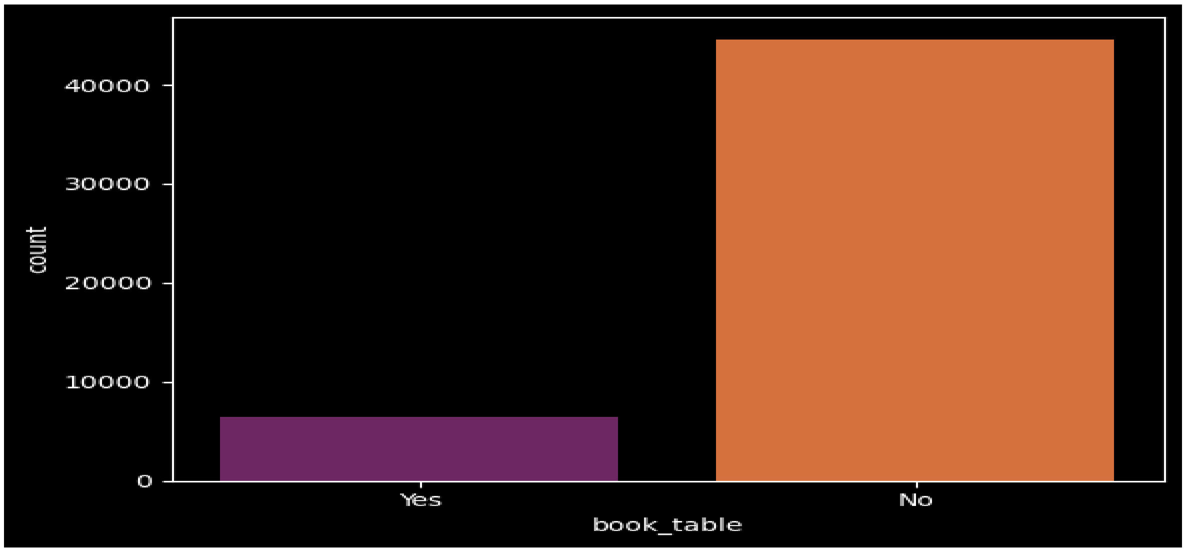
```
plt.figure(figsize= (6,6))
sns.countplot(x=df['online_order'],palette = 'inferno')
```



By this visualization, we can say that most of the restaurants have online order facilities.

Visualizing Book Table

```
sns.countplot(x=df['book_table'],palette='inferno')
```

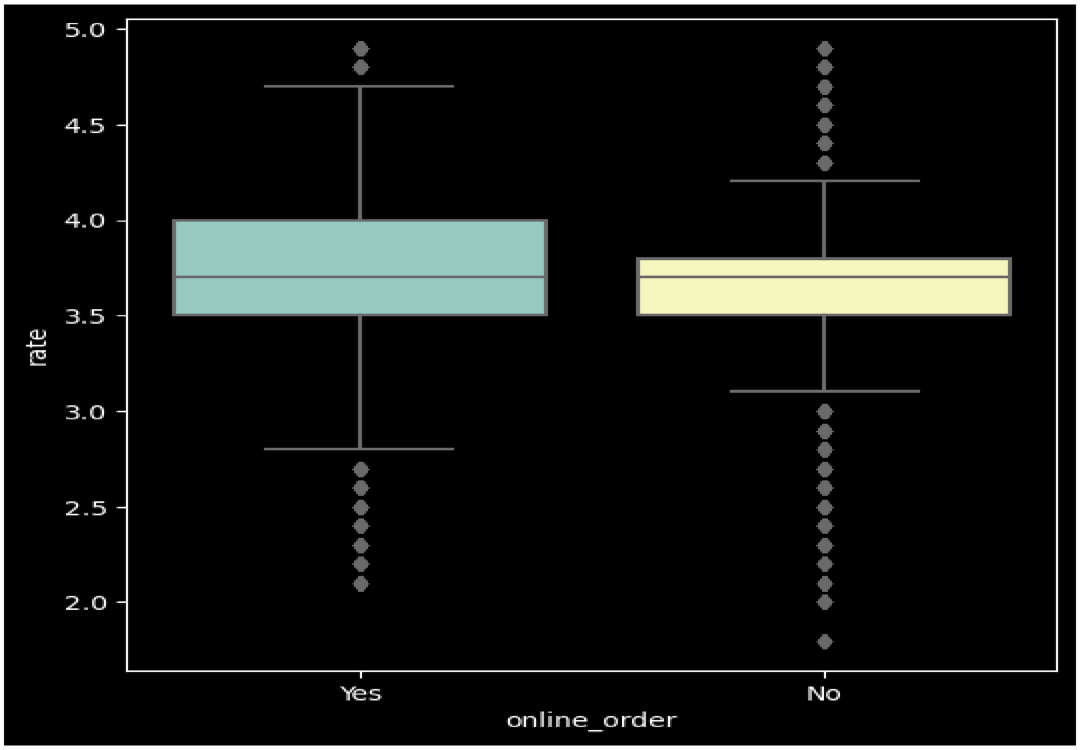


From this plot, we get to know that most of the restaurants are not having book table facilities.

Visualising Online Order vs Rate

```
plt.figure(figsize = (6,6))
```

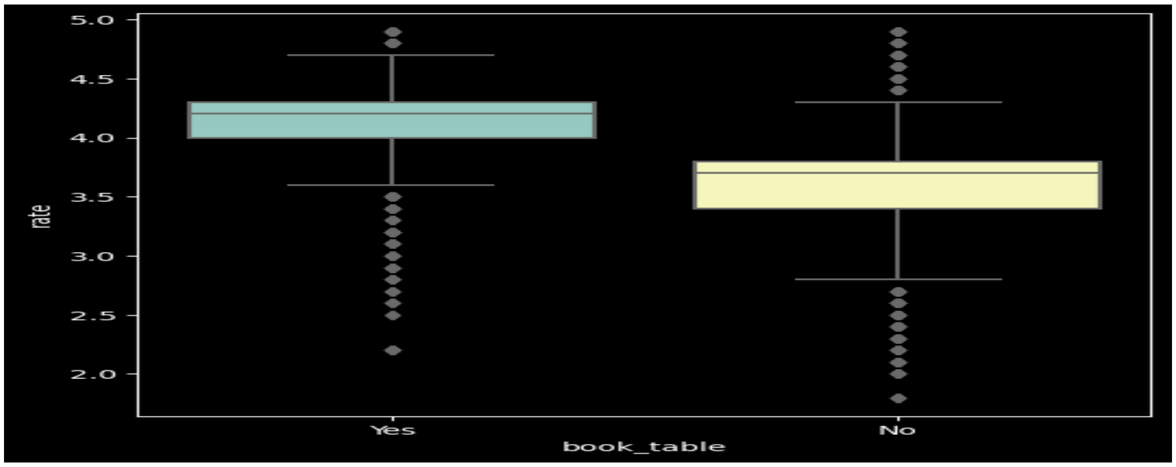
```
sns.boxplot(x='online_order',y='rate',data=df)
```



To check, if my restaurant has an online order facility the rate will be high or not? I created a boxplot with ‘online_order’ on the x-axis and ‘rate’ on the y-axis. I observed that the restaurants with online order facilities have a maximum rate(~4.7 for having online order facilities and ~ 3.7 for not having online order facilities) higher than the restaurants with no online order facilities.

Visualising Book Table vs Rate

```
plt.figure(figsize=(6,6))
sns.boxplot(x='book_table',y='rate',data=df)
```



Similarly, we created a boxplot with the rate on the y-axis and book_table on the x-axis. The restaurants which are having book-table facilities have average ratings greater than the restaurants with no book-table facilities. So, I should put a book table facility in my restaurant.

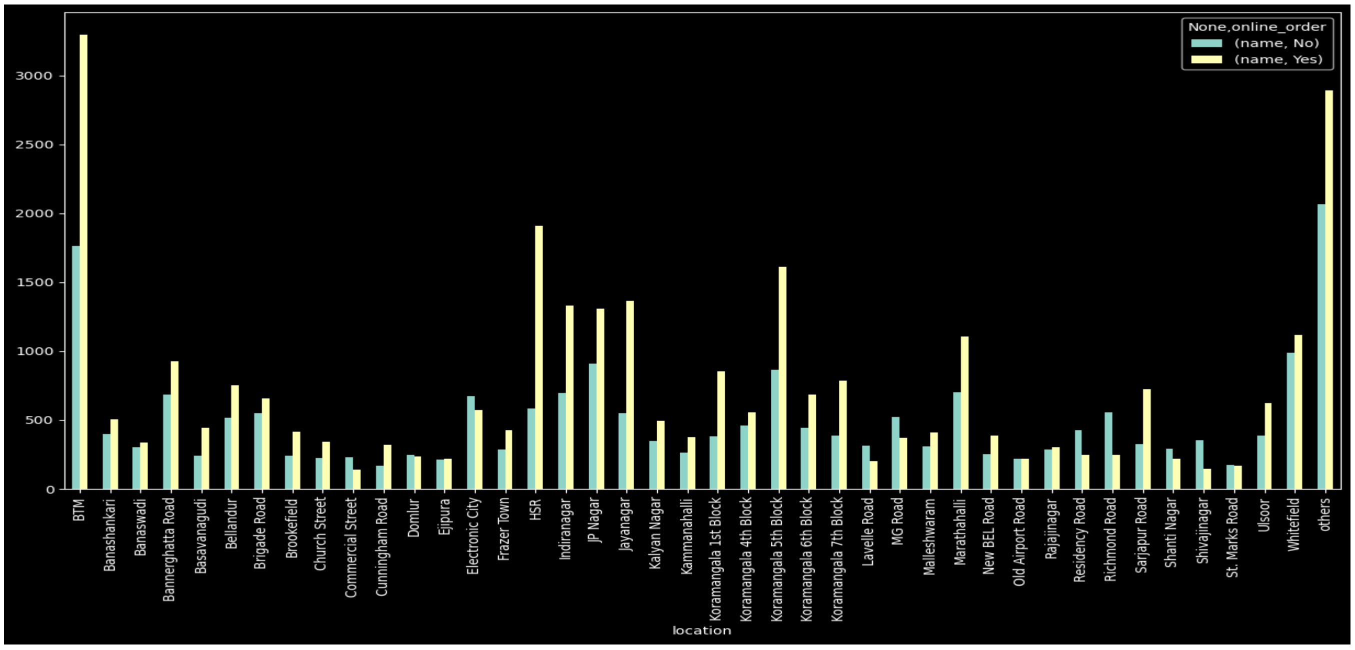
Visualising Online Order Facility, Location Wise

```
df1 = df.groupby(['location','online_order'])['name'].count()
df1.to_csv('location_online.csv')
df1 = pd.read_csv('location_online.csv')
df1 =
pd.pivot_table(df1,values=None,index=['location'],columns=['onli
ne_order'],fill_value=0,aggfunc = 'mean')

df1
```

Created a pivot table that gives the number of restaurants with online order facilities and the number of restaurants without online order facilities. Eg: In the BTM location there are 1763 restaurants without an online order facility and 3293 restaurants with an online order facility.

```
df1.plot(kind='bar',figsize=(15,8))
```



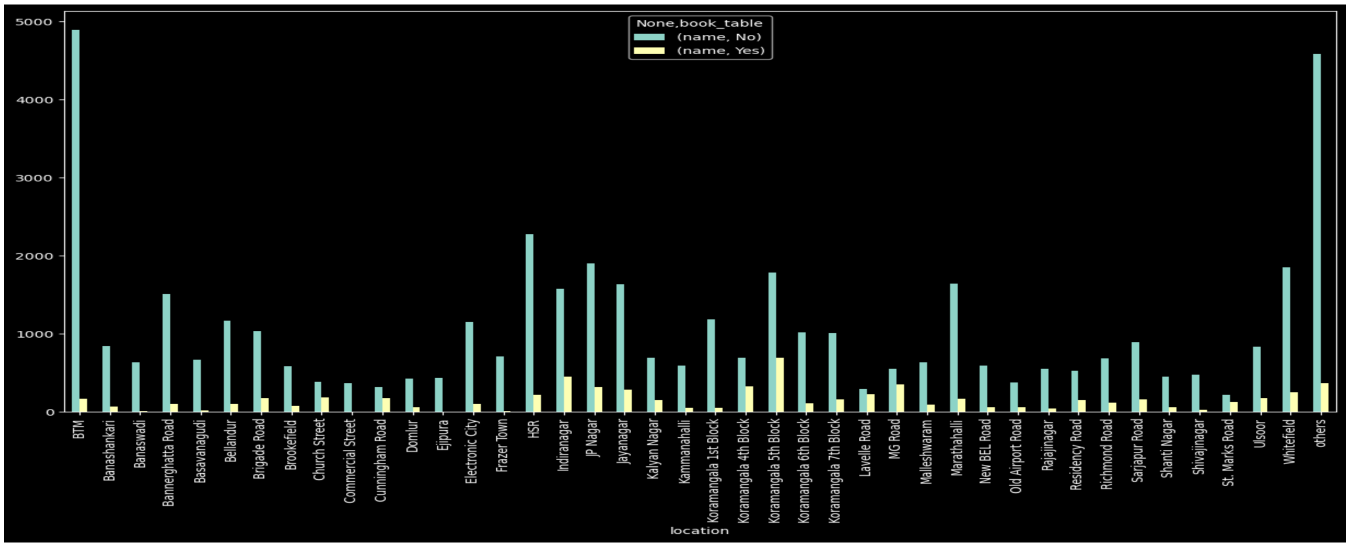
Visualising the above pivot table using the bar plot. From the plot, there are many restaurants with an online order facility in the BTM location. But in the Lavelle Road location, there are fewer restaurants having online order facilities compared to the restaurants with no online order facilities in that location.

Visualising Book Table Facility, Location Wise

```
df2 = df.groupby(['location','book_table'])['name'].count()
df2.to_csv('location_booktable.csv')
df2= pd.read_csv('location_booktable.csv')
df2 =
pd.pivot_table(df2,values=None,index='location',columns='book_ta
ble',aggfunc='sum',fill_value= 0)
df2
```

Similarly, created a pivot table that gives the number of restaurants with book table facility and the restaurants without a book table facility in a particular location.

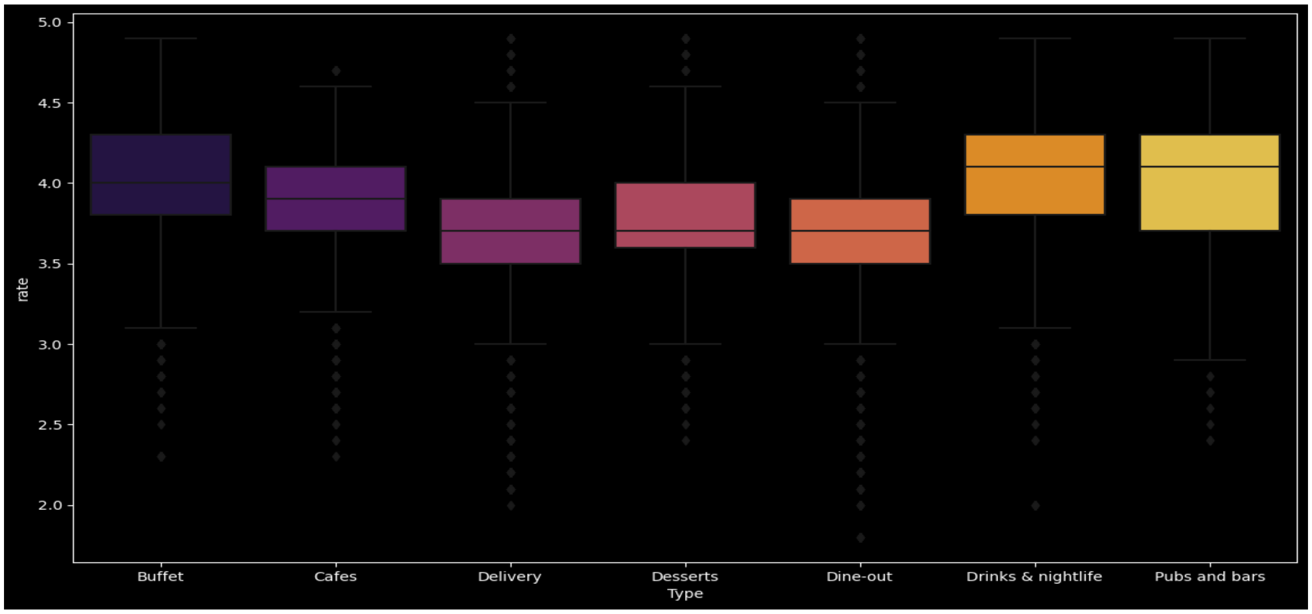
```
df2.plot(kind='bar',figsize=(15,8))
```



From the bar plot, the BTM location has more restaurants without a book table facility. Although it does not have restaurants with book table facilities we should not open a restaurant since it has a lot of restaurants. We can open a restaurant at the HSR location since there are also not many restaurants with book table facilities The chances of growth of a restaurant with book table facility is higher at the HSR location.

Visualising Types of Restaurants vs Rate

```
plt.figure(figsize=(14,8))
sns.boxplot(x='Type',y='rate',data=df,palette='inferno')
```



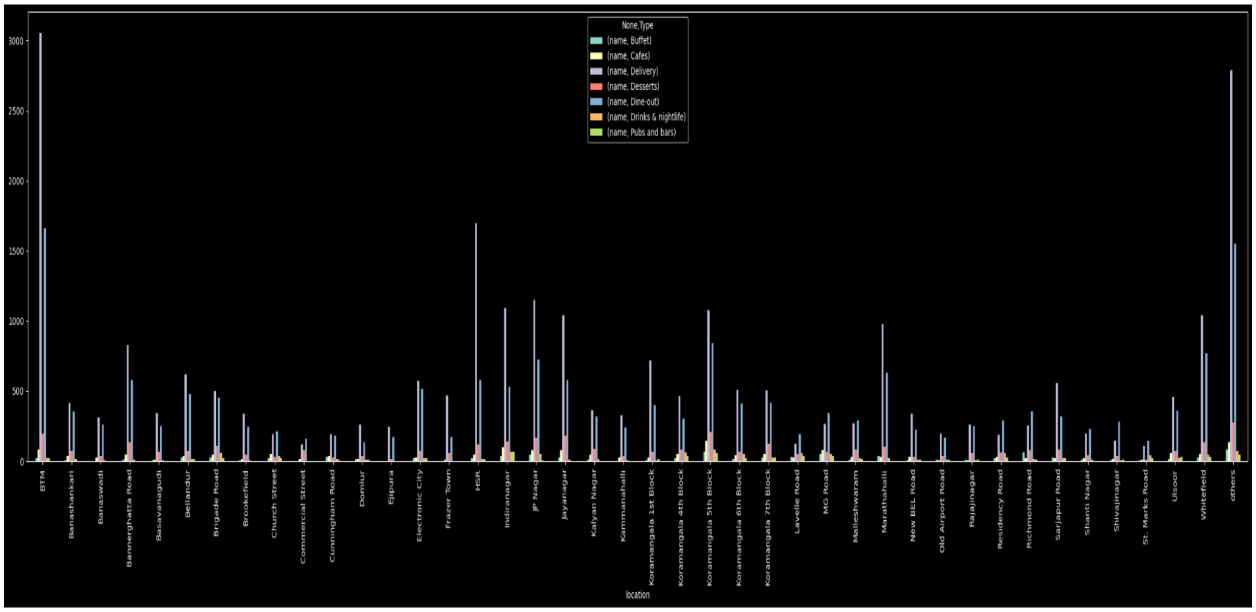
By this box plot, we came to know that Drinks & nightlife have a maximum average rating. ‘Delivery’ type of restaurant has a minimum average rating.

Grouping Types of Restaurants, location wise

```
df3 = df.groupby(['location','Type'])['name'].count()
df3.to_csv('location_Type.csv')
df3 = pd.read_csv('location_Type.csv')
df3 =
pd.pivot_table(df3,values=None,index=['location'],columns='Type'
,fill_value=0,aggfunc='sum')
df3
```

We will get the pivot table showing the number of different types of restaurants at a particular location.

```
df3.plot(kind='bar',figsize=(36,8))
```



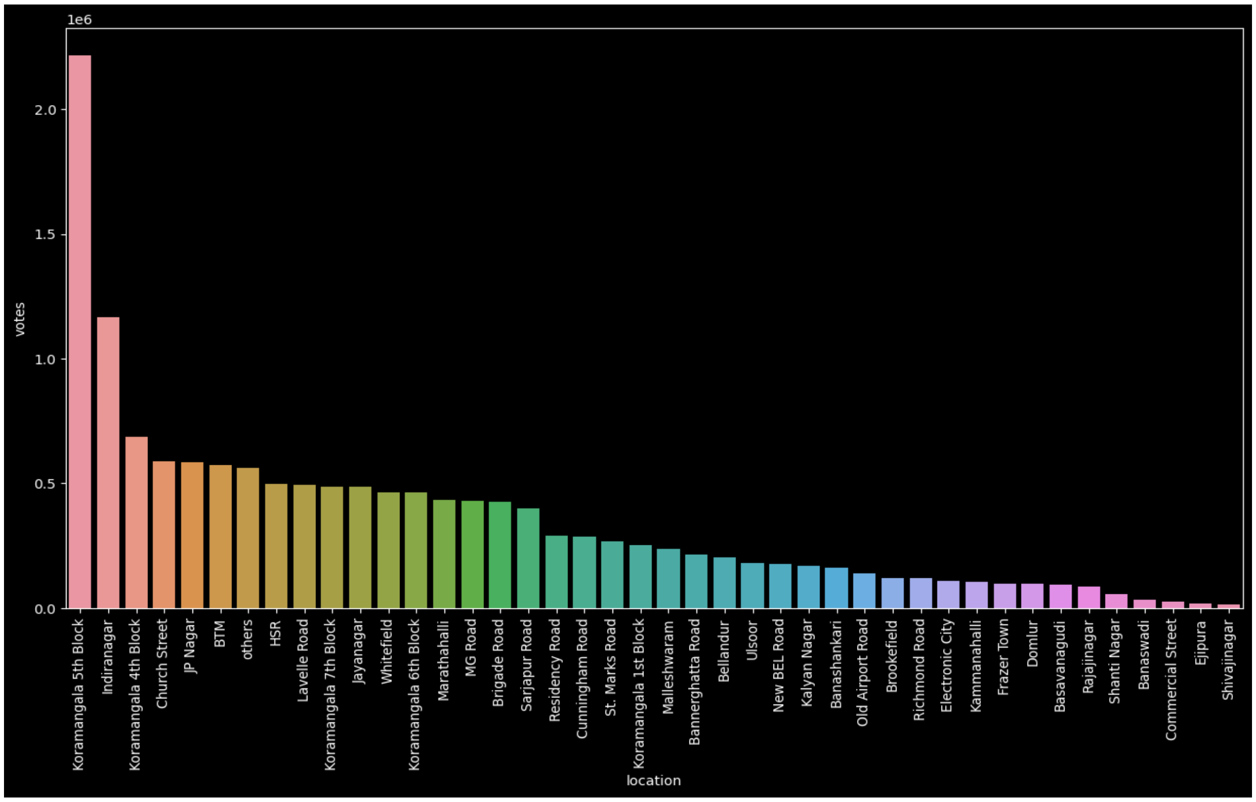
We created a bar plot with ‘Type’ on the x-axis and then we will get frequency on the y-axis. In Shivajinagar, pubs and bars are less. So, if we want to open a pub and bar we can open there.

No. of Votes, Location Wise

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

Since people's feedback is important, we will find which location of people are interested in voting. We will get a data frame with the location and the number of votes of that particular location.

```
plt.figure(figsize=(15,8))
sns.barplot(x=df5.index,y=df5['votes'])
plt.xticks(rotation=90)
```



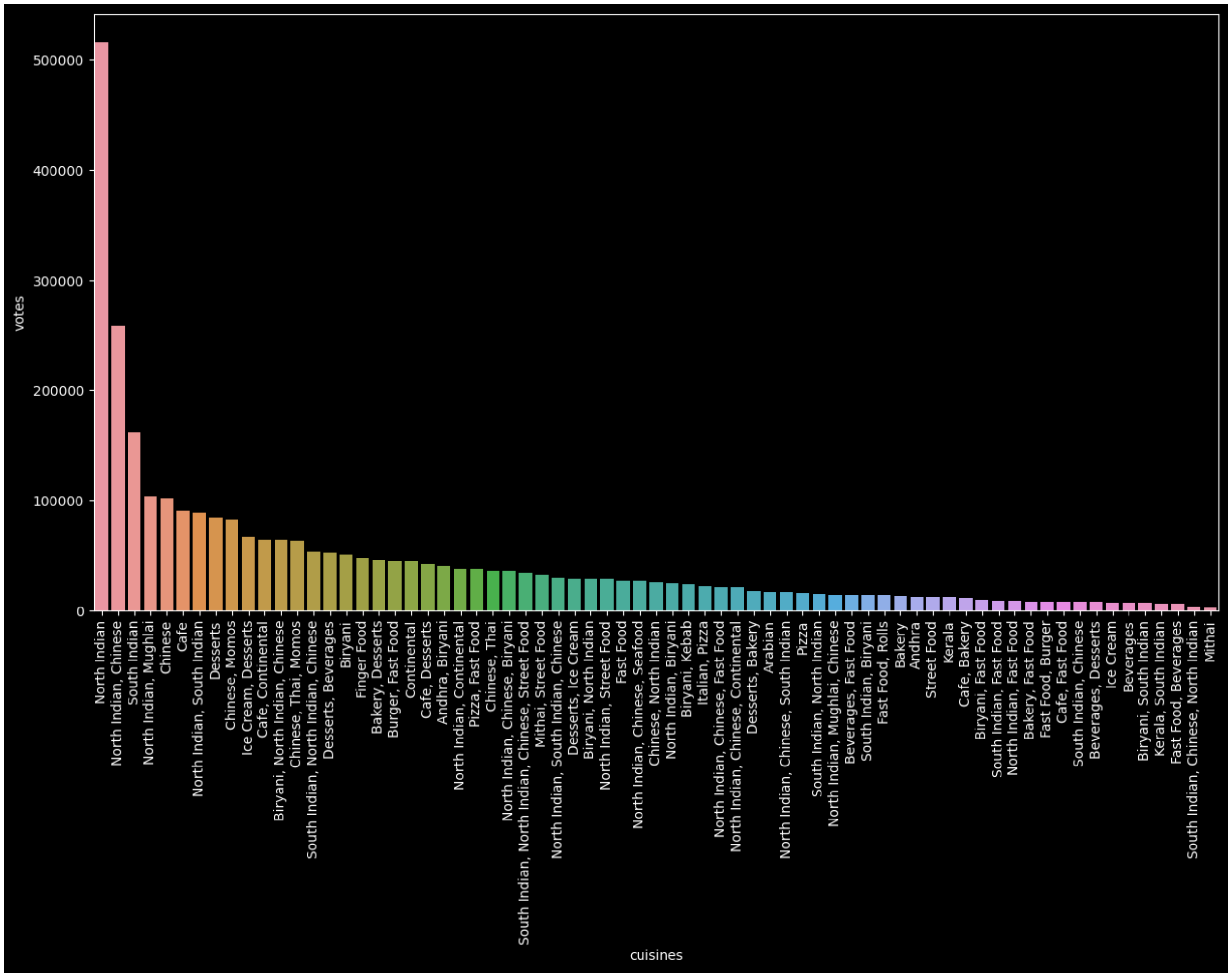
From the bar plot, people from the ‘Koramangala 5th Block’ voted highest compared to the people in the remaining locations. But Shivajinagar people are not interested in voting may be because of less number of restaurants. If we open a restaurant in ‘Koramangala 5th Block’ we will get very good customer feedback.

Visualising Top Cuisines

```
df6 = df[['cuisines','votes']]
df6.drop_duplicates()
df7 = df6.groupby(['cuisines'])['votes'].sum()
df7 = df7.to_frame()
df7 = df7.sort_values('votes',ascending=False)
df7.head()
```

If we want to open a particular type of cuisine restaurant, which type of cuisine restaurant should I open? By using the above code I created a data frame with the ‘cuisines’ column and the number of votes to the particular cuisine. Here North Indian has the highest number of votes with 516310 votes.

```
plt.figure(figsize=(15,8))
sns.barplot(x=df7.index,y=df7['votes'])
plt.xticks(rotation=90)
```



Created a bar plot with ‘cuisines’ on the x-axis and the ‘votes’ on the y-axis. ‘North Indian’, ‘North Indian, Chinese’, and ‘South Indian’ occupied the 1,2, and 3 positions in the votes frequency. If you want to open a cuisine, you can go for ‘North Indian’, ‘North Indian, Chinese’, and ‘South Indian’.