```
! pip install missingno
```

```
# Filtering over Dataset

from google.colab import data_table
data_table.enable_dataframe_formatter()
```

Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import missingno as msno
import plotly.express as px
import gc
from sklearn import preprocessing as pr
from sklearn.model_selection import train_test_split, KFold , cross_val_score , GridSearchCV
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
import sklearn.metrics as mt
from sklearn.ensemble import BaggingRegressor, RandomForestRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor

sns.set_style("whitegrid")
plt.style.use("fivethirtyeight")

```

Reading XLSX

```
data= pd.read_excel('data.xlsx')
#data= pd.read_csv('data_Rest.csv')
df=data.copy()
df.head(5)
```

1 to 5 of 5 entries Filter

index	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude
0	7402935	Skye	94	Jakarta	Menara BCA, Lantai 56, Jl. MH. Thamrin, Thamrin, Jakarta	Grand Indonesia Mall, Thamrin	Grand Indonesia Mall, Thamrin, Jakarta	106.821999	-6.196778
1	7410290	Satoo - Hotel Shangri-La	94	Jakarta	Hotel Shangri-La, Jl. Jend. Sudirman	Hotel Shangri-La, Sudirman	Hotel Shangri-La, Sudirman, Jakarta	106.8189611	-6.203291667
2	7420899	Sushi Masa	94	Jakarta	Jl. Tuna Raya No. 5, Penjaringan	Penjaringan	Penjaringan, Jakarta	106.800144	-6.101298
3	7421967	3 Wise Monkeys	94	Jakarta	Jl. Suryo No. 26, Senopati, Jakarta	Senopati	Senopati, Jakarta	106.8134001	-6.235241091
4	7422489	Avec Moi Restaurant and Bar	94	Jakarta	Gedung PIC, Jl. Teluk Betung 43, Thamrin, Jakarta	Thamrin	Thamrin, Jakarta	106.821023	-6.19627

Exploring Dataset

```
df.shape
```

    (9551, 15)

```
df.columns
```

```
Index(['Restaurant', 'Country Code', 'City', 'Longitude', 'Latitude',
       'Cuisines', 'Cost', 'Currency', 'Booking', 'Delivery', 'Price range',
       'Aggregate rating', 'Rating color', 'Rating text', 'Votes'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9551 entries, 0 to 9550
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Restaurant        9550 non-null   object
 1   Country Code      9551 non-null   int64
 2   City              9551 non-null   object
 3   Longitude         9551 non-null   float64
 4   Latitude          9551 non-null   float64
 5   Cuisines          9542 non-null   object
 6   Cost              9551 non-null   int64
 7   Currency          9551 non-null   object
 8   Booking           9551 non-null   object
 9   Delivery          9551 non-null   object
 10  Price range       9551 non-null   int64
 11  Aggregate rating  9551 non-null   float64
 12  Rating color      9551 non-null   object
 13  Rating text       9551 non-null   object
 14  Votes             9551 non-null   int64
dtypes: float64(3), int64(4), object(8)
memory usage: 1.1+ MB
```

Renaming some Columns

```
df.rename(columns={'Restaurant Name':"Restaurant","Average Cost for two":"Cost","Has Table booking":"Booking","Has Online delivery":"Delivery"}, inplace
df.columns
```

```
Index(['Restaurant', 'Country Code', 'City', 'Longitude', 'Latitude',
       'Cuisines', 'Cost', 'Currency', 'Booking', 'Delivery', 'Price range',
       'Aggregate rating', 'Rating color', 'Rating text', 'Votes'],
      dtype='object')
```

---> Categorical variables

```
categorical_features = df.select_dtypes(include=["object"]).columns
categorical_features
```

```
Index(['Restaurant', 'City', 'Address', 'Locality', 'Locality Verbose',
       'Cuisines', 'Currency', 'Booking', 'Delivery', 'Rating color',
       'Rating text'],
      dtype='object')
```

Finding duplicate Row(s)

```
df.drop_duplicates(keep='first').shape
```

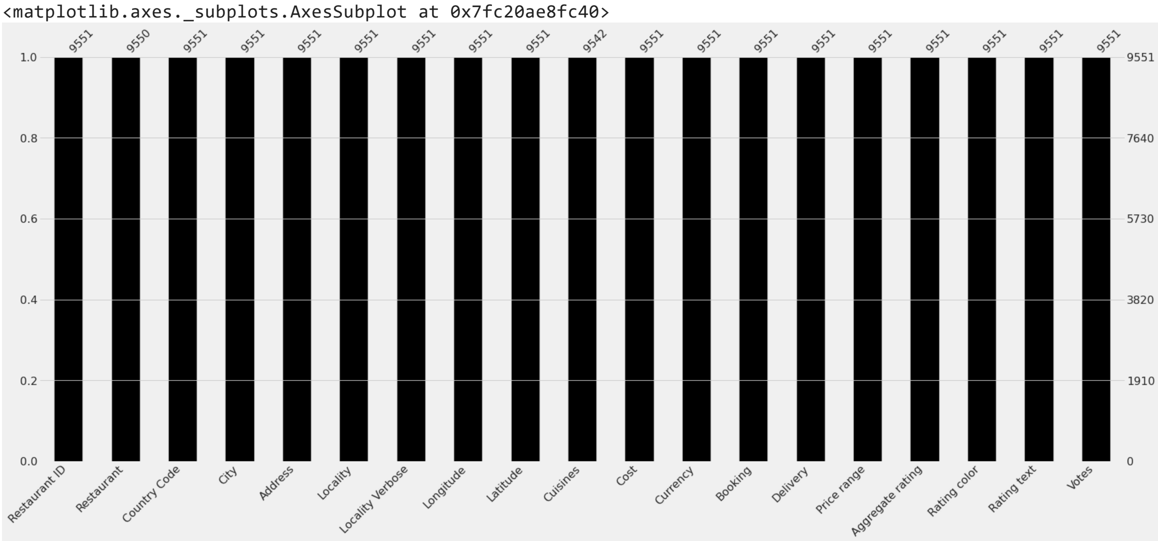
```
(9551, 19)
```

No duplicated row(s)

Missing Values

```
msno.bar(df, color='black')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc20ae8fc40>
```




```
# Percentage of missing values and the number of Missing Values
missingvalue=df.isnull().sum()
perc_missingvalue=df.isnull().sum()*100/len(df)
perc_mising_df = pd.DataFrame({'Variables': df.columns, 'MissingValues' :missingvalue ,'Percentage_Missing': perc_missingvalue })
perc_mising_df.sort_values('Percentage_Missing', ascending=False, inplace=True)
display(perc_mising_df)

print('the number of RESTAURANT missing Values   :', df['Restaurant'].isnull().sum(),'   and its missing value percentage :   %', df['Restaurant'].isnu
print('the number of CUISINES missing Values   :',df['Cuisines'].isnull().sum(), '   and its missing value percentage :   %', df['Cuisines'].isnull().s
```

1 to 19 of 19 entries Filter

index	Variables	MissingValues	Percentage_Missing
Cuisines	Cuisines	9	0.09423097057899696
Restaurant	Restaurant	1	0.010470107842110775
Cost	Cost	0	0.0
Rating text	Rating text	0	0.0
Rating color	Rating color	0	0.0
Aggregate rating	Aggregate rating	0	0.0
Price range	Price range	0	0.0
Delivery	Delivery	0	0.0
Booking	Booking	0	0.0
Currency	Currency	0	0.0
Restaurant ID	Restaurant ID	0	0.0
Latitude	Latitude	0	0.0
Longitude	Longitude	0	0.0
Locality Verbose	Locality Verbose	0	0.0
Locality	Locality	0	0.0
Address	Address	0	0.0
City	City	0	0.0
Country Code	Country Code	0	0.0
Votes	Votes	0	0.0

Show 25 per page

```
the number of RESTAURANT missing Values   : 1    and its missing value percentage :   % 0.010470107842110775
the number of CUISINES missing Values   : 9    and its missing value percentage :   % 0.09423097057899696
```

Drop the following columns in first step

```
df.drop(df.columns[[0,4,5,6]], axis = 1, inplace=True)
df.shape
```

```
(9551, 15)
```

Descibing The Dataset

```
df.describe().T
```

1 to 7 of 7 entries Filter

index	count	mean	std	min	25%	50%	75%	max
Country Code	9551.0	18.365616165846507	56.75054560094657	1.0	1.0	1.0	1.0	216.0
Longitude	9551.0	64.12657446168704	41.46705784761728	-157.948486	77.08134305	77.1919642	77.2820063	174.8320893
Latitude	9551.0	25.854380700074756	11.007935124784668	-41.330428	28.4787126	28.57046888	28.6427582	55.97698
Cost	9551.0	1199.2107632708617	16121.183073499647	0.0	250.0	400.0	700.0	800000.0
Price range	9551.0	1.804837189823055	0.9056088473976142	1.0	1.0	2.0	2.0	4.0
Aggregate rating	9551.0	2.6663700136111403	1.5163775396521326	0.0	2.5	3.2	3.7	4.9
Votes	9551.0	156.909747670401	430.1691453762912	0.0	5.0	31.0	131.0	10934.0

Checking Values in Columns

--- Country Code ---

```
display(df['Country Code'].unique())

```

```
array([ 94,   1, 162, 191, 189, 166, 184, 214,  30, 208, 215, 148,  14,
       216,  37])
```

```
data_country= pd.read_excel('Country-Code.xlsx')
df_country=data_country.copy()
display(df_country.head())
```

1 to 5 of 5 entries Filter

index	Country Code	Country
0	1	India
1	14	Australia
2	30	Brazil
3	37	Canada
4	94	Indonesia

Show 25 per page

```
df=pd.merge(df,df_country, on='Country Code', how='left')
df
```

1 to 25 of 9551 entries Filter

index	Restaurant	Country Code	City	Longitude	Latitude	Cuisines	Cost	Currency	Booking	Delivery	Price rang
0	Skye	94	Jakarta	106.821999	-6.196778	Italian, Continental	800000	Indonesian Rupiah(IDR)	No	No	
1	Satoo - Hotel Shangri-La	94	Jakarta	106.8189611	-6.203291667	Asian, Indonesian, Western	800000	Indonesian Rupiah(IDR)	No	No	
2	Sushi Masa	94	Jakarta	106.800144	-6.101298	Sushi, Japanese	500000	Indonesian Rupiah(IDR)	No	No	
3	3 Wise Monkeys	94	Jakarta	106.8134001	-6.235241091	Japanese	450000	Indonesian Rupiah(IDR)	No	No	
4	Avec Moi Restaurant and Bar	94	Jakarta	106.821023	-6.19627	French, Western	350000	Indonesian Rupiah(IDR)	No	No	
5	Lucky Cat Coffee & Kitchen	94	Jakarta	106.8317481	-6.218932479	Cafe, Western	300000	Indonesian Rupiah(IDR)	No	No	
6	Onokabe	94	Tangerang	106.652688	-6.241792	Indonesian	300000	Indonesian Rupiah(IDR)	No	No	
7	Lemongrass	94	Bogor	106.8078499	-6.576578026	Peranakan, Indonesian	250000	Indonesian Rupiah(IDR)	No	No	
8	MONKS	94	Jakarta	106.9113346	-6.163947933	Western, Asian, Cafe	250000	Indonesian Rupiah(IDR)	No	No	
9	Talaga Sampireun	94	Jakarta	106.7285083	-6.168466667	Sunda, Indonesian	200000	Indonesian Rupiah(IDR)	No	No	
10	OJJU	94	Jakarta	106.783162	-6.244221	Korean	200000	Indonesian Rupiah(IDR)	No	No	
11	Union Deli	94	Jakarta	106.8197488	-6.197150016	Desserts, Bakery, Western	200000	Indonesian Rupiah(IDR)	No	No	
12	Zenbu	94	Jakarta	106.8425	-6.224333333	Japanese, Sushi, Ramen	200000	Indonesian Rupiah(IDR)	No	No	
13	Talaga Sampireun	94	Jakarta	106.8335532	-6.12685982	Sunda, Indonesian	200000	Indonesian Rupiah(IDR)	No	No	
14	Talaga Sampireun	94	Tangerang	106.7261194	-6.269913889	Sunda, Indonesian	200000	Indonesian Rupiah(IDR)	No	No	
						Cafe,					

To check Data Types

Western,

```
1 df.dtypes
```

```
Restaurant           object
Country Code          int64
City                 object
Longitude           float64
Latitude            float64
Cuisines             object
Cost                  int64
Currency             object
Booking              object
Delivery             object
Price range           int64
Aggregate rating    float64
Rating color         object
Rating text          object
Votes                 int64
Country              object
dtype: object
```

-- Country --

```
1 df['Country'].value_counts()
```

```
India             8652
United States      434
United Kingdom      80
South Africa        60
UAE                 60
Brazil              60
New Zealand         40
Turkey              34
Australia           24
Phillipines         22
```

```
Indonesia           21
Sri Lanka           20
Qatar               20
Singapore           20
Canada               4
Name: Country, dtype: int64
```

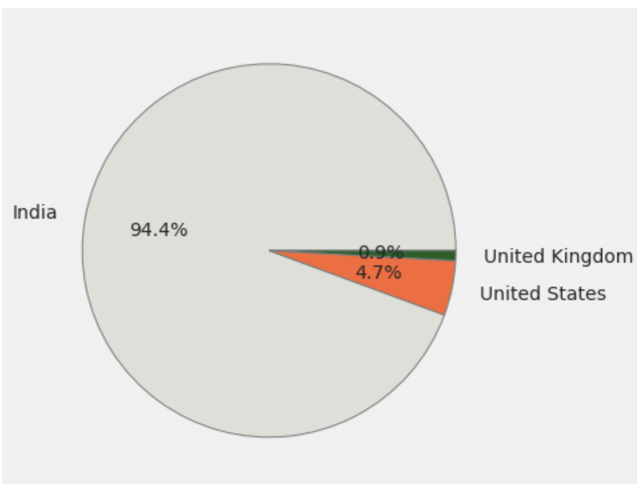
```
from locale import normalize
country_names=df['Country'].value_counts().index
country_names
```

```
Index(['India', 'United States', 'United Kingdom', 'South Africa', 'UAE',
       'Brazil', 'New Zealand', 'Turkey', 'Australia', 'Phillipines',
       'Indonesia', 'Sri Lanka', 'Qatar', 'Singapore', 'Canada'],
      dtype='object')
```

```
country_values=df['Country'].value_counts().values
country_values
```

```
array([8652,  434,   80,   60,   60,   60,   40,   34,   24,   22,   21,
         20,   20,   20,    4])
```

```
#Pie Chart (Top 3 countries  )
plt.figure(figsize=(10, 6))

colors = ['#DEDFD8', '#EB6F42',  '#2E5F2B']
plt.pie(country_values[:3], labels=country_names[:3], autopct="%1.1f%%",labeldistance=1.15, wedgeprops = { 'linewidth' : 1, 'edgecolor' : 'gray' }, colo

```




```
colors = sns.color_palette('pastel')[0:10]
plt.figure(figsize=(14, 8))
sns.barplot(x=country_names, y=country_values, data=df);
```

-- Rating --

Lets see the intervals of the Rating and the meaning of Rating Color and Rating Text

```
Rating1=df[['Aggregate rating','Rating color','Rating text','Votes']]
print(Rating1['Rating color'].value_counts())
display(Rating1['Rating color'].value_counts(normalize))  # Percentage
```

```
Orange        3737
White         2148
Yellow        2100
Green         1079
Dark Green     301
Red            186
Name: Rating color, dtype: int64
Orange        0.391268
White         0.224898
Yellow        0.219872
Green         0.112972
Dark Green    0.031515
Red           0.019474
Name: Rating color, dtype: float64
```

```
result_rating=Rating1.groupby('Rating color')['Aggregate rating', 'Rating text'].aggregate(['min','max'])
result_rating
```

```
<ipython-input-36-4c111074d48e>:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tupl
  result_rating=Rating1.groupby('Rating color')['Aggregate rating', 'Rating text'].aggregate(['min','max'])
```

	Aggregate rating		Rating text	
	min	max	min	max
Rating color				
Dark Green	4.5	4.9	Excellent	Excellent
Green	4.0	4.4	Very Good	Very Good
Orange	2.5	3.4	Average	Average
Red	1.8	2.4	Poor	Poor
White	0.0	0.0	Not rated	Not rated
Yellow	3.5	3.9	Good	Good

```
Rating2 =df.groupby(['Aggregate rating', 'Rating color', 'Rating text']).size().reset_index().rename(columns={0:"Rating Count"})
Rating2
```

1 to 25 of 33 entries Filter

index	Aggregate rating	Rating color	Rating text	Rating Count
0	0.0	White	Not rated	2148
1	1.8	Red	Poor	1
2	1.9	Red	Poor	2
3	2.0	Red	Poor	7

Observation :

- Rating Rates:
- 2.0 - 2.4 —> Poor (White)
- 2.5 - 2.9 —> Average (Red)
- 3.0 - 3.4 —> Average (Orange)
- 3.5 - 3.9 —> Good (Yellow)
- 4.0 - 4.4 —> Very Good (Green)
- 4.5 - 4.9 —> Excellent (Dark Green)

Zero rating has been given by many of the people has not rated

18	3.5	Yellow	Good	480

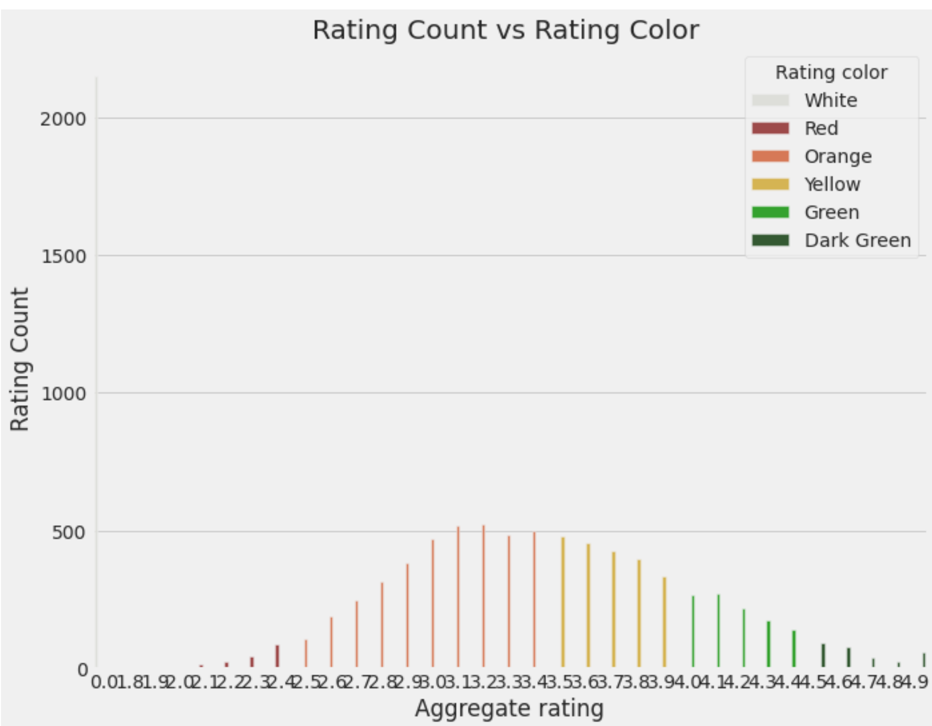
Double-click (or enter) to edit

21	3.8	Yellow	Good	400

Rating Count vs Rating Color

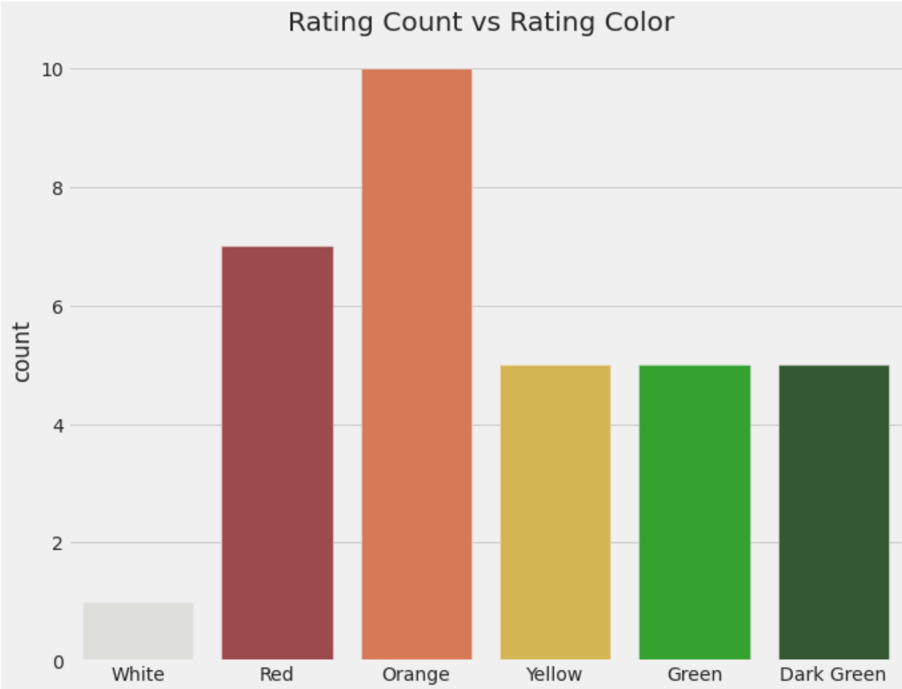
24	4.1	Green	Very Good	274

```
plt.figure(figsize=(10, 8))
plt.title("Rating Count vs Rating Color ")
sns.barplot(data=Rating2, x='Aggregate rating', y='Rating Count', hue='Rating color', palette=['#DEDFD8', '#AB3C3C', '#EB6F42', '#EBBF3D', '#23B51B', '#

```



1. NOT RATED (Gray Bar) count is very high
2. Maximum number of Rating is 3.2

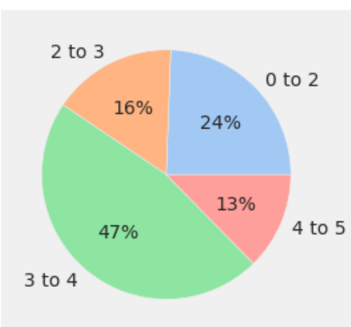
```
# Count Plot
plt.figure(figsize=(10, 8))
plt.title("Rating Count vs Rating Color ")
sns.countplot(data=Rating2, x='Rating color', palette=['#DEDFD8', '#AB3C3C', '#EB6F42', '#EBBF3D', '#23B51B', '#2E5F2B']);
```



Resturants with rating

```
Rest_rating = {}
Rest_rating['0 to 2'] = df[df['Aggregate rating'] < 2 ].shape[0]
Rest_rating['2 to 3'] = df[(df['Aggregate rating'] < 3) & (df['Aggregate rating'] > 2) ].shape[0]
Rest_rating['3 to 4'] = df[(df['Aggregate rating'] < 4) & (df['Aggregate rating'] > 3) ].shape[0]
Rest_rating['4 to 5'] = df[(df['Aggregate rating'] < 5) & (df['Aggregate rating'] > 4) ].shape[0]
```

```
colors = sns.color_palette('pastel')[0:4]
plt.pie(Rest_rating.values(),labels =Rest_rating.keys(),colors=colors,autopct = '%.f%%')
plt.show()
```



How could Delivery affect to Rating ?

[ ] ↳ *1 cell hidden*

Find the countries name that has given 0(ZERO) rating

```
Rate0_countries=df[df['Rating color']=='White'].groupby('Country').size().reset_index().rename(columns={0:"Not Rating Count"})
Rate0_countries


#df.groupby(['Aggregate rating', 'Country']).size().reset_index().rename(columns={0:"Zero Rating Count"})
```

1 to 4 of 4 entries Filter

index	Country	Not Rating Count
3	United States	3

- Observation: Maximum number of 0 zero ratings are from India.
- Plan : We have big quantity of ZERO outliers . It is not good idea to drop the. Better to fill with **the median of each 'Country** '

--- Table Booking ---

[ ] ↳ *2 cells hidden*

How does delivery affect the Rating?

```
df.groupby(["Delivery",'Rating text'])['Delivery'].count()
```

```
Delivery  Rating text
No        Average        2632
          Excellent       262
          Good           1282
          Not rated      2052
          Poor             70
          Very Good       802
Yes       Average        1105
          Excellent        39
          Good            818
          Not rated        96
          Poor            116
          Very Good       277
Name: Delivery, dtype: int64
```

The Restaurant that has NO delivery are rated with AVERAGE or NOT RATED. The Restaurant that has delivery are more than NO-Delivery Restaurant. But Rating is mostly 'AVERAGE'

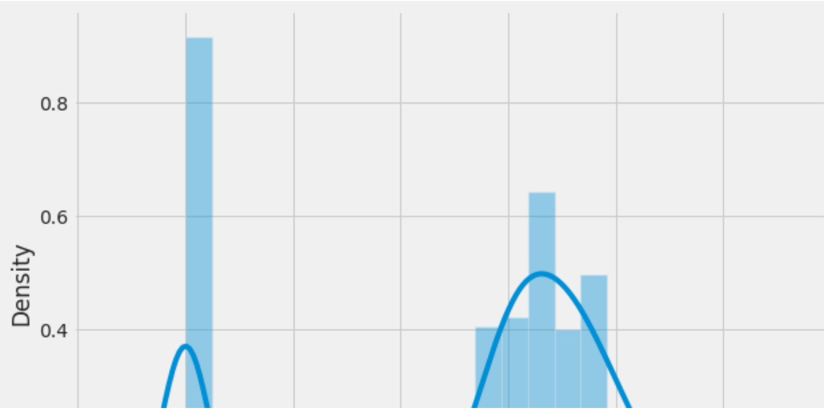
How does VOTED DELIVERY affect RATING?

```
colors = sns.color_palette('pastel')[0:4]
sns.barplot(x='Delivery', y='Votes', hue='Rating text',data=df,palette=['#2E5F2B', '#23B51B', '#EBBF3D' ,'#AB3C3C','#EB6F42','#DEDFD8'] );
```

Deliveries that are NOT VOTED has NOT RATED either.

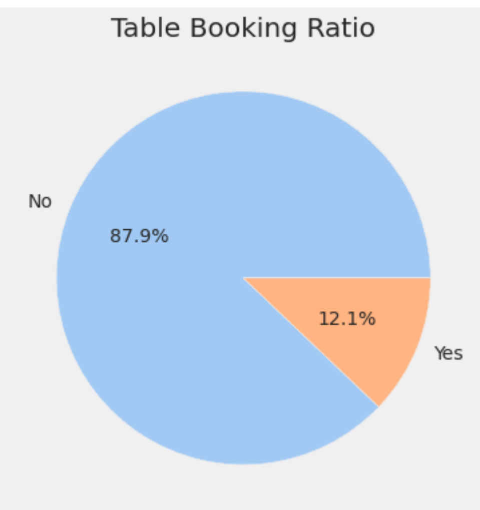
```
plt.figure(figsize=(9,7))

sns.distplot(df['Aggregate rating'],bins=20);
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecat
  warnings.warn(msg, FutureWarning)
```



What is the ratio between restaurants that allow table booking vs that do not allow table booking?

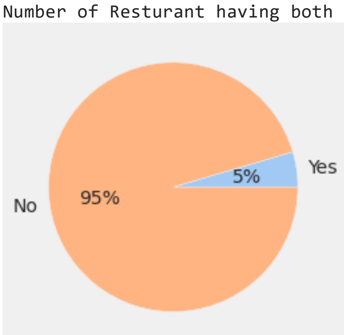
```
booking_values=df["Booking"].value_counts().values
booking_names=df["Booking"].value_counts().index

colors = sns.color_palette('pastel')[0:2]
plt.figure(figsize=(12, 6))
plt.title('Table Booking Ratio')
plt.pie(booking_values, labels = booking_names, colors = colors, autopct="%1.1f%%")
plt.show()
```



Resturants are having both Online Delivery and Table Booking

```
colors = sns.color_palette('pastel')[0:2]
delivery_booking = df.query('Delivery=="Yes" & Booking == "Yes"')
print('Number of Resturant having both online Delivery and Table Booking is',delivery_booking.shape[0])
plt.pie([delivery_booking.shape[0],df.shape[0]-delivery_booking.shape[0]],labels=['Yes','No'],autopct='%.0f%%', colors=colors)
plt.show()
```

```
Number of Resturant having both online Delivery and Table Booking is 435
```



-- Currency --

```
df['Currency']=df['Currency'].replace({'Dollar($)':'Dollar','Pounds(â£)':'Pounds','Brazilian Real(R$)':'Brazilian Real','NewZealand($)':'NewZealand Dol
```

Which country uses which currency???

```
CurrencyCountry = df.groupby(['Currency', 'Country']).size().reset_index().rename(columns={0:"Currency Count"})
CurrencyCountry

```

1 to 15 of 15 entries Filter

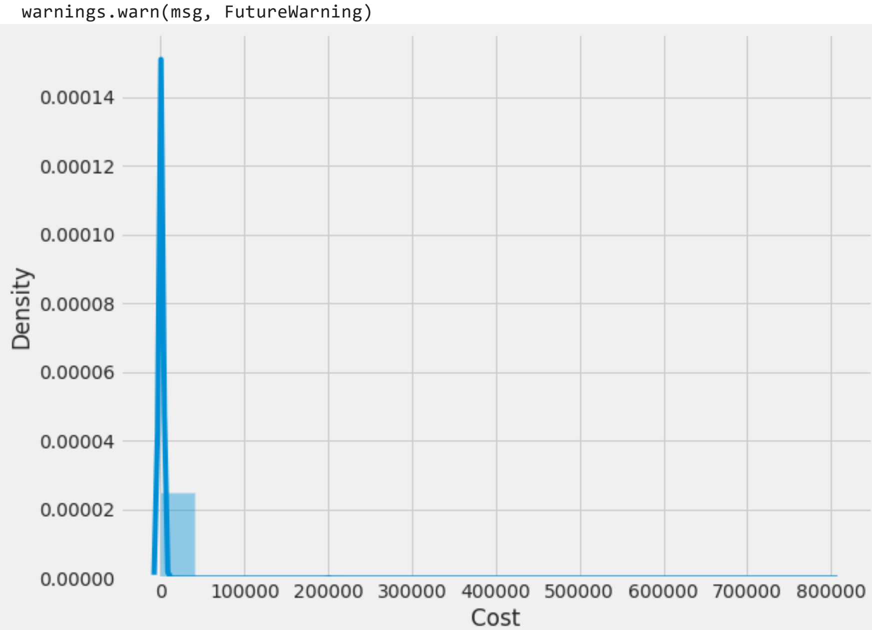
index	Currency	Country	Currency Count
0	Botswana Pula(P)	Phillipines	22
1	Brazilian Real	Brazil	60
2	Dollar	Australia	24
3	Dollar	Canada	4
4	Dollar	Singapore	20
5	Dollar	United States	434
6	Emirati Diram(AED)	UAE	60
7	Indian Rupees(Rs.)	India	8652
8	Indonesian Rupiah(IDR)	Indonesia	21
9	NewZealand Dollar	New Zealand	40
10	Pounds	United Kingdom	80
11	Qatari Rial(QR)	Qatar	20
12	Rand(R)	South Africa	60
13	Sri Lankan Rupee(LKR)	Sri Lanka	20
14	Turkish Lira(TL)	Turkey	34

Show 25 per page

Converting all Currencies to Indian Rupee and Creation New 'Cost' Column

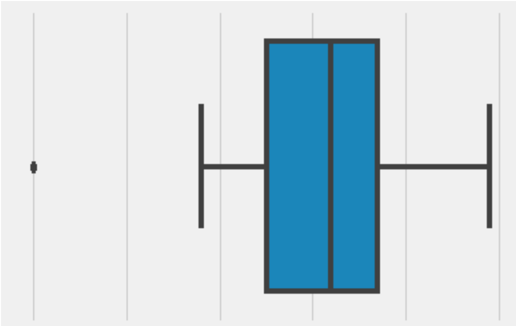
```
plt.figure(figsize=(9,7))

sns.distplot(df['Cost'],bins=20);
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecat
  warnings.warn(msg, FutureWarning)
```




```
sns.boxplot(data=df, x='Aggregate rating')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc2068ecc40>
```

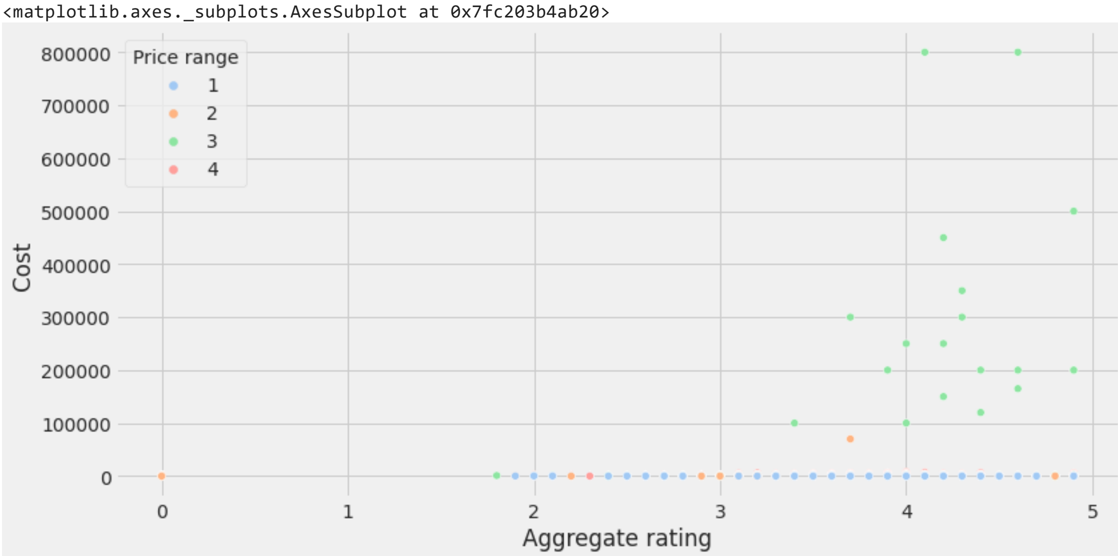


```
df.sort_values(by=["Cost"]);
```

Rating VS Cost of dinning

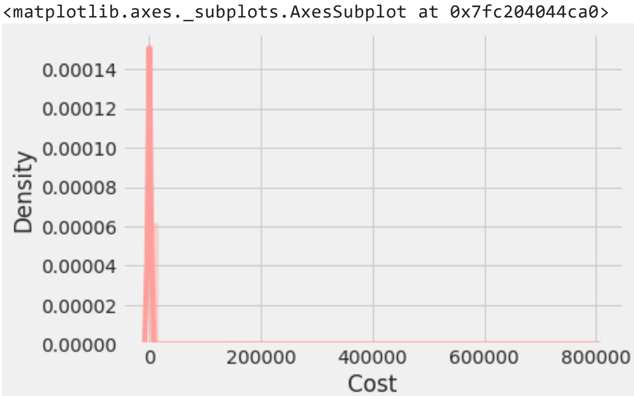
```
colors = sns.color_palette('pastel')[0:4]
plt.figure(figsize=(12,6))
#sns.scatterplot(x="Cost", y="Aggregate rating", hue='Price range', data=df, palette=colors)
sns.scatterplot(y="Cost", x="Aggregate rating", hue='Price range', data=df, palette=colors)

```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc203b4ab20>
```




```
colors = sns.color_palette('pastel')[3]
sns.distplot(df['Cost'], color=colors)
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use ei

<matplotlib.axes._subplots.AxesSubplot at 0x7fc204044ca0>
```



-- Deliveries --

Which country has online deliveries?

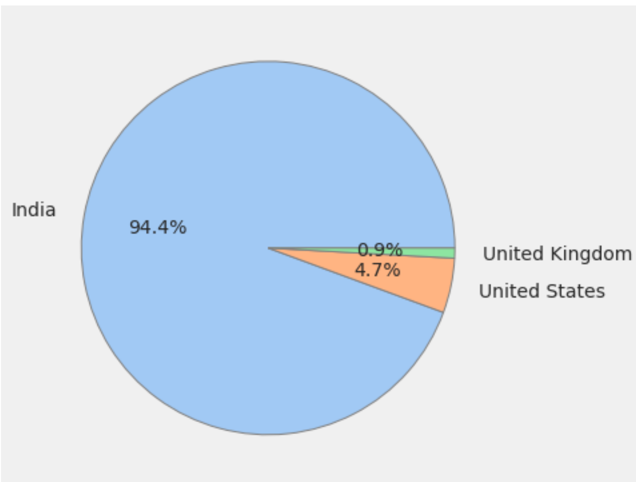
```
df.groupby(['Delivery']).size().reset_index().rename(columns={0:"Delivery Count"})
```

1 to 2 of 2 entries Filter

index	Delivery	Delivery Count
0	No	7100
1	Yes	2451

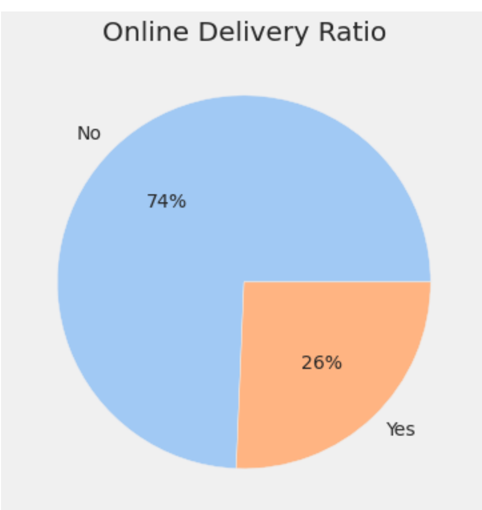
Show 25 per page

```
#Pie Chart (Top 3 countries )

plt.figure(figsize=(12, 6))
colors = sns.color_palette('pastel')[0:3]
plt.pie(country_values[:3], labels=country_names[:3], autopct="%1.1f%%",labeldistance=1.15, wedgeprops = { 'linewidth' : 1, 'edgecolor' : 'gray' }, col

```



What is the percentage of restaurants providing online delivery?

```
delivery_values=df['Delivery'].value_counts().values
delivery_names=df['Delivery'].value_counts().index

colors = sns.color_palette('pastel')[0:2]
plt.figure(figsize=(12, 6))
plt.title ('Online Delivery Ratio')
plt.pie(delivery_values, labels = delivery_names, colors = colors, autopct='%.0f%%')
plt.show()
```




```
df[df['Delivery']=="Yes"].Country.value_counts()
```

```
India    2423
UAE        28
Name: Country, dtype: int64
```

```
1 # All deliveries
2 df[['Delivery','Country']].groupby(['Delivery','Country']).size().reset_index().rename(columns={0:"Currency Count"})
```

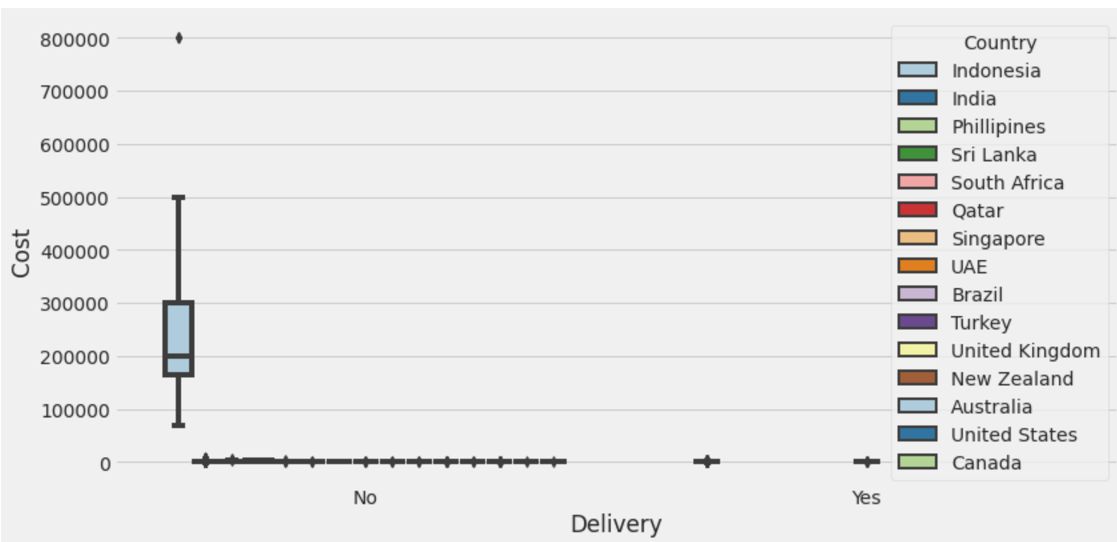
1 to 17 of 17 entries Filter

index	Delivery	Country	Currency Count
0	No	Australia	24
1	No	Brazil	60
2	No	Canada	4
3	No	India	6229
4	No	Indonesia	21
5	No	New Zealand	40
6	No	Phillipines	22
7	No	Qatar	20
8	No	Singapore	20
9	No	South Africa	60
10	No	Sri Lanka	20
11	No	Turkey	34
12	No	UAE	32
13	No	United Kingdom	80
14	No	United States	434
15	Yes	India	2423
16	Yes	UAE	28

Show 25 per page

- Observation : Online Deliveries are mostly available in UAE and India

```
1 plt.figure(figsize=(12, 6))
2 sns.boxplot(x='Delivery', y='Cost' ,hue='Country', data=df, palette="Paired");
```



-- Cities --

```
1 city_number=df[['Country']].groupby(['Country']).size().reset_index().rename(columns={0:"City Count"})
2 city_number
```

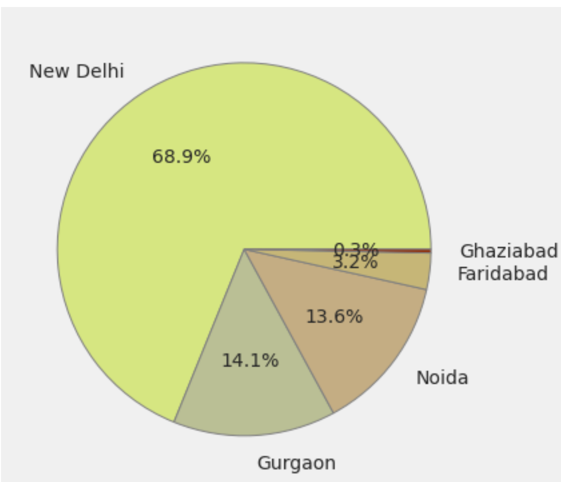
1 to 15 of 15 entries Filter

index	Country	City Count
0	Australia	24
1	Brazil	60
2	Canada	4
3	India	8652
4	Indonesia	21
5	New Zealand	40
6	Phillipines	22
7	Qatar	20

- Observation : 8652 of 9551 restaurants in cities are from Indian Cities

Top 5 cities in this dataset

```
City_values=df['City'].value_counts().values
City_names=df['City'].value_counts().index

# Create a Pie Chart for cities distribution
#Pie Chart (Top 5 cities that uses Zomato )
plt.figure(figsize=(8, 6))

# Create a set of colors
colors = ['#D6E681', '#BABF95', '#C4AD83', '#C6B677', '#912F0B']
plt.pie(City_values[:5], labels=City_names[:5], autopct="%1.1f%%",labeldistance=1.15, wedgeprops = { 'linewidth' : 1, 'edgecolor' : 'gray' }, colors=cc
```

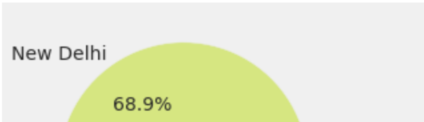


Top 5 indian Cities locations

```
Indian_Cities=df[df['Country']=='India']
Indian_Cities_Count= Indian_Cities['City'].value_counts().head(5)
Indian_Cities_Count
```

```
New Delhi     5473
Gurgaon       1118
Noida         1080
Faridabad      251
Ghaziabad       25
Name: City, dtype: int64
```

```
plt.pie (Indian_Cities_Count.values[:5],
    labels=Indian_Cities_Count.index[:5],
    autopct="%1.1f%%",
    colors=['#D6E681', '#BABF95', '#C4AD83', '#C6B677', '#E49D19', '#912F0B']);
```



- Observation : Indian cities are the top cities as like Top Cities in this Dataset too

--- Cuisines ---

```
#Helps in finding the popular cuisines
df.Cuisines.value_counts().head(10)
```

```
North Indian                      936
North Indian, Chinese             511
Fast Food                         354
Chinese                           354
North Indian, Mughlai             334
Cafe                              299
Bakery                            218
North Indian, Mughlai, Chinese    197
Bakery, Desserts                  170
Street Food                       149
Name: Cuisines, dtype: int64
```

```
# Find the index number of Null Values
print('Index Number of Null Value in Cuisines  :  ',df[df['Cuisines'].isnull()].index.tolist())

df.iloc[[9083, 9086, 9094, 9406, 9494, 9504, 9533, 9535, 9539]]
```

```
Index Number of Null Value in Cuisines  :   [9083, 9086, 9094, 9406, 9494, 9504, 9533, 9535, 9539]
```

1 to 9 of 9 entries Filter

index	Restaurant	Country Code	City	Longitude	Latitude	Cuisines	Cost	Currency	Booking	Delivery	Price range	Aggreg
9083	Corkscrew Cafe	216	Gainesville	-83.9858	34.5318	NaN	40	Dollar	No	No	3	
9086	Dovetail	216	Macon	-83.627979	32.83641	NaN	40	Dollar	No	No	3	
9094	Hillstone	216	Orlando	-81.36526	28.596682	NaN	40	Dollar	No	No	3	
9406	Jimmie's Hot Dogs	216	Albany	-84.1534	31.5751	NaN	10	Dollar	No	No	1	
9494	Leonard's Bakery	216	Rest of Hawaii	-157.813432	21.284586	NaN	10	Dollar	No	No	1	
9504	Tybee Island Social Club	216	Savannah	-80.848297	31.99581	NaN	10	Dollar	No	No	1	
9533	Cookie Shoppe	216	Albany	-84.154	31.5772	NaN	0	Dollar	No	No	1	
9535	Pearly's Famous Country Cookng	216	Albany	-84.1759	31.5882	NaN	0	Dollar	No	No	1	
9539	Hl Lite Bar	216	Miller	-98.9891	44.5158	NaN	0	Dollar	No	No	1	

All of Null Values located in USA And Filling the Nan Values with MODE

```
from typing import ValuesView
#Imputing the null values with the Mode in "no_of_cusisnes"

USA_Cities=df[df['Country']=='United States']
display(USA_Cities['Cuisines'].value_counts())
df['Cuisines'] = df['Cuisines'].fillna(USA_Cities['Cuisines'].mode()[0])
```

```
Mexican                         25
American                        16
BBQ                              9
Chinese                          9
Italian                          8
                                ..
Pizza, Bar Food, Sandwich        1
Chinese, Seafood, Vegetarian     1
American, Steak                  1
Asian, Japanese, Sushi           1
Desserts, Pizza, Ice Cream       1
Name: Cuisines, Length: 229, dtype: int64
```

```
df['Cuisines'].isnull().sum()
```

```
0
```

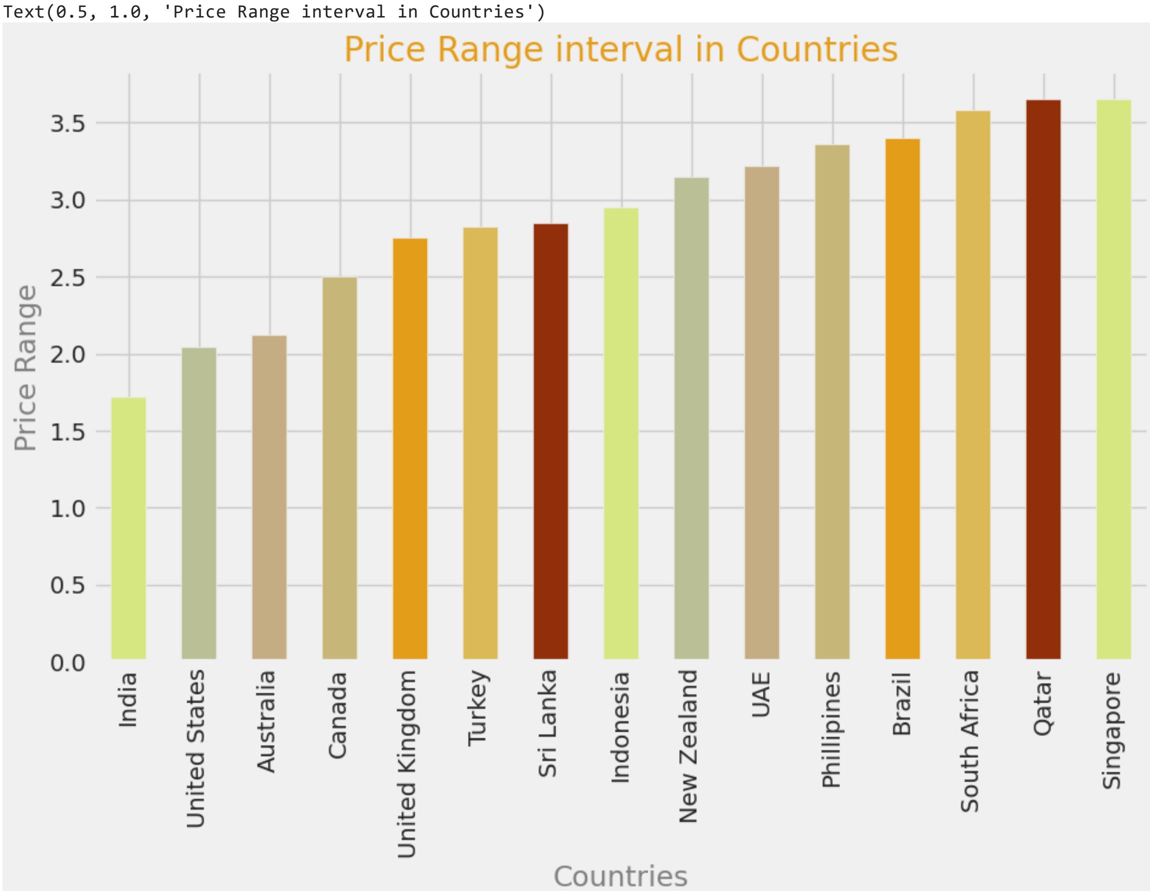
```
df["Total_Cuisines"]=df.apply((lambda x: len(x["Cuisines"].split(','))),axis=1)
df["Total_Cuisines"]
```

```
0       2
1       3
2       2
3       1
4       2
       ..
9546    3
9547    3
9548    6
9549    1
9550    2
Name: Total_Cuisines, Length: 9551, dtype: int64
```

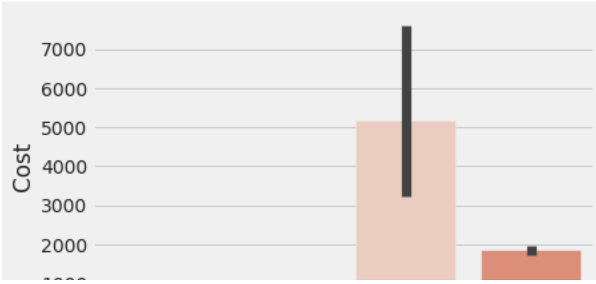
--- Price Range ---

```
plt.figure(figsize=(6,4),dpi=100)
df.groupby(['Country']).mean()['Price range'].sort_values().plot(kind='bar',figsize=(10,6), color=['#D6E681', '#BABF95', '#C4AD83', '#C6B677', '#E49D1
plt.xlabel('Countries', color='Gray')
plt.ylabel('Price Range', color='Gray')
plt.title('Price Range interval in Countries', color='#E49D19')
```

```
Text(0.5, 1.0, 'Price Range interval in Countries')
```

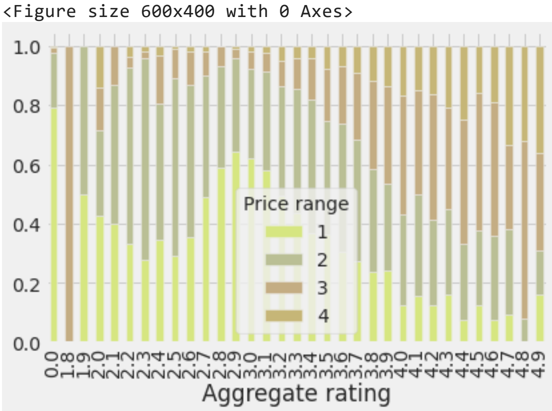



```
sns.barplot(x='Price range',y='Cost',palette="coolwarm",data=df)
plt.show()
```




```
plt.figure(figsize=(6,4),dpi=100)

table=pd.crosstab(df["Aggregate rating"],df["Price range"])
table.div(table.sum(1).astype(float),axis=0).plot(kind='bar',stacked=True,color=['#D6E681', '#BABF95', '#C4AD83', '#C6B677', '#E49D19', '#DBB957', '#9
```

```
<Figure size 600x400 with 0 Axes>
```



-- Votes --

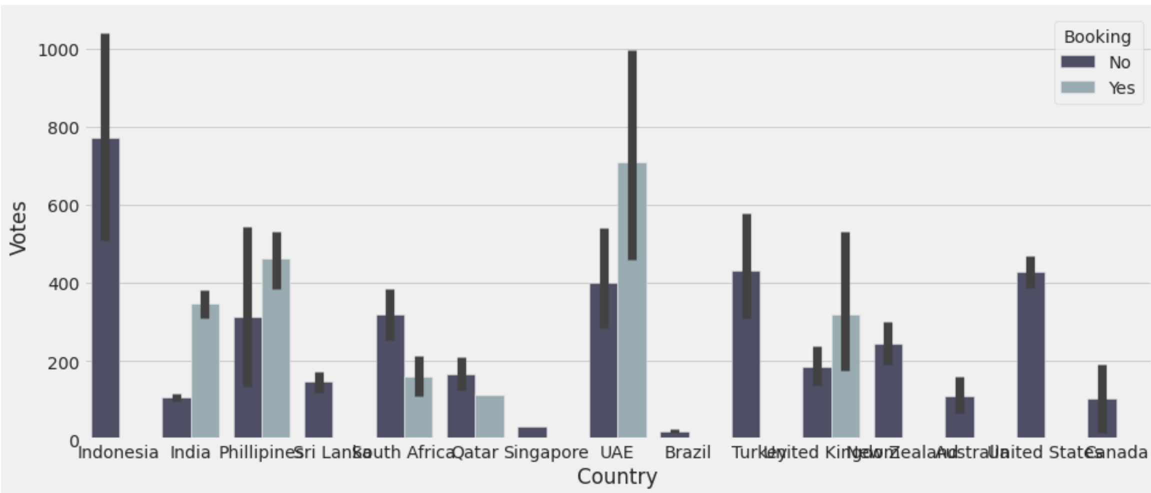
```
df['Votes'].unique()
```

```
array([1498,  873,  605, ...,  880,  680, 1868])
```

```
plt.figure(figsize=(6,4),dpi=100)
df.groupby(['Country']).mean()['Votes'].sort_values().plot(kind='bar',figsize=(10,6), color=['#D6E681', '#BABF95', '#C4AD83', '#C6B677', '#E49D19', '#
plt.xlabel('Votes', color='Gray')
plt.ylabel('Country', color='Gray')
plt.title('Votes by Countries', color='#AB3C3C')
```

```
Text(0.5, 1.0, 'Votes by Countries')
```


```
sns.barplot(x="Country",y="Votes",hue="Booking",palette="bone",data=df)
fig2 = plt.gcf()
fig2.set_size_inches(14,6)
plt.show()
```



Pivot Table Delivery vs Booking ???

```
pd.crosstab(df['Delivery'],df['Aggregate rating'])
#pd.crosstab(df['Rating text'], df['Delivery'])
```

Warning: Total number of columns (33) exceeds max_columns (20). Falling back to pandas display.

Aggregate rating	0.0	1.8	1.9	2.0	2.1	2.2	2.3	2.4	2.5	2.6	...	4.0	4.1	4.2	4.3	4.4	4.5	4.6	4.7	4.8	4
Delivery																					
No	2052	0	1	2	4	7	14	42	42	87	...	188	208	158	135	113	79	67	35	24	
Yes	96	1	1	5	11	20	33	45	68	104	...	78	66	63	39	31	16	11	7	1	

2 rows × 33 columns

How does Booking affect Votes upon Rating ?

```
sns.barplot(x='Rating text', y='Votes', hue='Booking',data=df );
```

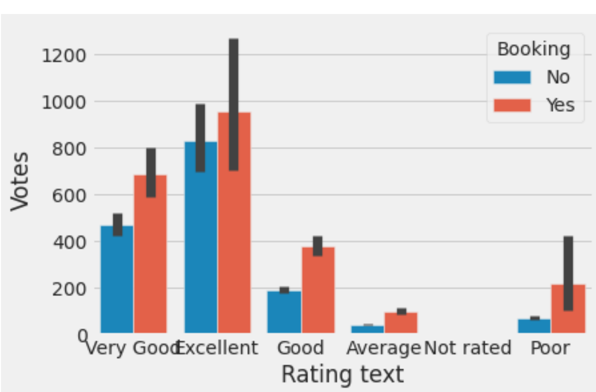


Table-Booking-Restaurants have rated and voted more than No-Booking-Restaurants.

-- Restaurants --

```
1 print('Index Number of Null Value in Restaurant  :  ',df[df['Restaurant'].isnull()].index.tolist())
```

```
Index Number of Null Value in Restaurant  :   [1646]
```

```
1 df['Restaurant'].fillna('Unknown Name')
```

```
0                                Skye
1            Satoo - Hotel Shangri-La
2                          Sushi Masa
3                      3 Wise Monkeys
4         Avec Moi Restaurant and Bar
                    ...
9546             BMG - All Day Dining
9547    Atmosphere Grill Cafe Sheesha
9548                       UrbanCrave
9549                Deena Chat Bhandar
9550                   VNS Live Studio
Name: Restaurant, Length: 9551, dtype: object
```

Find Most Expensive Restaurants

```
1 TopExpRest=df.sort_values(by="Cost", ascending=False)
2 TopExpRest.head()
```

1 to 5 of 5 entries Filter

index	Restaurant	Country Code	City	Longitude	Latitude	Cuisines	Cost	Currency	Booking	Delivery	Price range
0	Skye	94	Jakarta	106.821999	-6.196778	Italian, Continental	800000	Indonesian Rupiah(IDR)	No	No	3
1	Satoo - Hotel Shangri-La	94	Jakarta	106.8189611	-6.203291667	Asian, Indonesian, Western	800000	Indonesian Rupiah(IDR)	No	No	3
2	Sushi Masa	94	Jakarta	106.800144	-6.101298	Sushi, Japanese	500000	Indonesian Rupiah(IDR)	No	No	3
3	3 Wise Monkeys	94	Jakarta	106.8134001	-6.235241091	Japanese	450000	Indonesian Rupiah(IDR)	No	No	3
4	Avec Moi Restaurant and Bar	94	Jakarta	106.821023	-6.19627	French, Western	350000	Indonesian Rupiah(IDR)	No	No	3

Show 25 per page

Find Most Expensive Restaurants in **India**

```
1 TopExpRestIndia = df[(df.Country == 'India')].sort_values(by="Cost", ascending=False)
2 TopExpRestIndia.head(15)
```

1 to 15 of 15 entries Filter

index	Restaurant	Country Code	City	Longitude	Latitude	Cuisines	Cost	Currency	Booking	Delivery	Price range	A
21	Orient Express - Taj Palace Hotel	1	New Delhi	77.170087	28.5950077	European	8000	Indian Rupees(Rs.)	Yes	No	4	
22	Tian - Asian Cuisine Studio - ITC Maurya	1	New Delhi	77.1734547	28.5973505	Asian, Japanese, Korean, Thai, Chinese	7000	Indian Rupees(Rs.)	No	No	4	
23	Bukhara - ITC Maurya	1	New Delhi	77.1737243	28.5974659	North Indian	6500	Indian Rupees(Rs.)	No	No	4	
25	Nostalgia at 1911 Brasserie - The Imperial	1	New Delhi	77.218187	28.625445	European, Continental	6000	Indian Rupees(Rs.)	Yes	No	4	
26	1911 - The Imperial	1	New Delhi	77.218185	28.625443	North Indian, Chinese, South Indian, Italian	6000	Indian Rupees(Rs.)	Yes	No	4	
27	The Spice Route - The Imperial	1	New Delhi	77.218187	28.625445	Malaysian, Thai, Kerala, Vietnamese, Sri Lankan	6000	Indian Rupees(Rs.)	Yes	No	4	
28	Wasabi by Morimoto - The Taj Mahal Hotel	1	New Delhi	77.2243039	28.6052532	Japanese, Sushi	6000	Indian Rupees(Rs.)	Yes	No	4	
29	MEGU - The Leela Palace	1	New Delhi	77.1889651	28.5794009	Japanese, Sushi	5500	Indian Rupees(Rs.)	Yes	No	4	
30	House of Ming - The Taj Mahal Hotel	1	New Delhi	77.2246182	28.6051487	Chinese	5500	Indian Rupees(Rs.)	Yes	No	4	
31	24/7 Restaurant	1	New	77.22756944	28.63148611	Continental, North Indian,	5100	Indian	Yes	No	4	

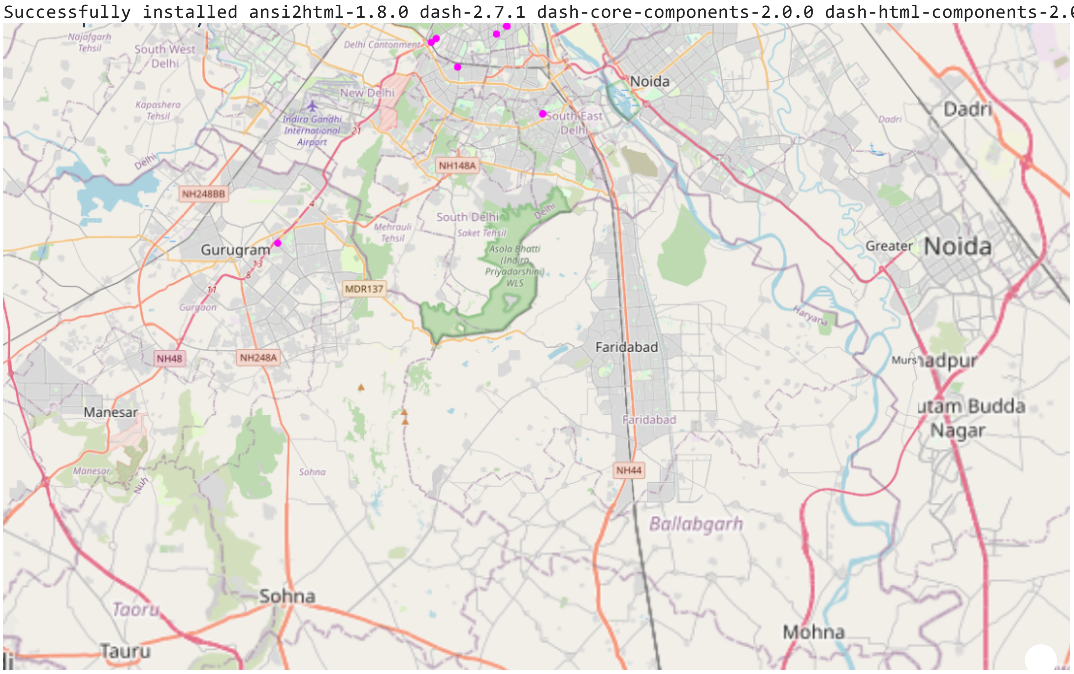
```
#Highest 25 costly restaurants  in India
TERI2=TopExpRestIndia.nlargest(25, "Cost")
TERI2
```

1 to 25 of 25 entries Filter

index	Restaurant	Country Code	City	Longitude	Latitude	Cuisines	Cost	Currency	Booking	Delivery	Price range
21	Orient Express - Taj Palace Hotel	1	New Delhi	77.170087	28.5950077	European	8000	Indian Rupees(Rs.)	Yes	No	4
22	Tian - Asian Cuisine Studio - ITC Maurya	1	New Delhi	77.1734547	28.5973505	Asian, Japanese, Korean, Thai, Chinese	7000	Indian Rupees(Rs.)	No	No	4
23	Bukhara - ITC Maurya	1	New Delhi	77.1737243	28.5974659	North Indian	6500	Indian Rupees(Rs.)	No	No	4
25	Nostalgia at 1911 Brasserie - The Imperial	1	New Delhi	77.218187	28.625445	European, Continental	6000	Indian Rupees(Rs.)	Yes	No	4
26	1911 - The Imperial	1	New Delhi	77.218185	28.625443	North Indian, Chinese, South Indian, Italian	6000	Indian Rupees(Rs.)	Yes	No	4
27	The Spice Route - The Imperial	1	New Delhi	77.218187	28.625445	Malaysian, Thai, Kerala, Vietnamese, Sri Lankan	6000	Indian Rupees(Rs.)	Yes	No	4
28	Wasabi by Morimoto - The Taj Mahal Hotel	1	New Delhi	77.2243039	28.6052532	Japanese, Sushi	6000	Indian Rupees(Rs.)	Yes	No	4
29	MEGU - The Leela Palace	1	New Delhi	77.1889651	28.5794009	Japanese, Sushi	5500	Indian Rupees(Rs.)	Yes	No	4
30	House of Ming - The Taj Mahal Hotel	1	New Delhi	77.2246182	28.6051487	Chinese	5500	Indian Rupees(Rs.)	Yes	No	4
31	24/7 Restaurant - The Lalit New Delhi	1	New Delhi	77.22756944	28.63148611	Continental, North Indian, Italian, Asian	5100	Indian Rupees(Rs.)	Yes	No	4
38	Jade - The Claridges	1	New Delhi	77.2168963	28.6001953	Chinese	5000	Indian Rupees(Rs.)	Yes	No	4
43	Machan - The Taj Mahal Hotel	1	New Delhi	77.2241369	28.6051648	North Indian, European, Continental	5000	Indian Rupees(Rs.)	No	No	4
42	The Grill Room - The Taj Mahal Hotel	1	New Delhi	77.2241228	28.6051535	Mediterranean, European	5000	Indian Rupees(Rs.)	No	No	4
41	Le Cirque - The Leela Palace	1	New Delhi	77.1889752	28.5793901	French, Italian	5000	Indian Rupees(Rs.)	Yes	No	4
39	San Gimignano - The	1	New Delhi	77.218187	28.625445	Italian	5000	Indian Rupees(Rs.)	Yes	No	4

```
# Top 25 costly Resturants Location in Kolkata
!pip install jupyter-dash
import plotly.express as px


fig = px.scatter_mapbox(TERI2, lat="Latitude", lon="Longitude", hover_name="City", hover_data=["Aggregate rating",
                                                                                                'Restaurant',"Cost",
                                                                                                "Booking"],
                        color_discrete_sequence=["fuchsia"], zoom=10, height=300)
fig.update_layout(mapbox_style="open-street-map")
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.update_layout(title='Top 25 costly Resturants Location',
                  autosize=True,
                  #hovermode='closest',
                  showlegend=False)
fig.update_layout(
    autosize=False,
    width=800,
    height=500,)

fig.show();
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting jupyter-dash
  Downloading jupyter_dash-0.4.2-py3-none-any.whl (23 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from jupyter-dash) (2.23.€
Collecting ansi2html
  Downloading ansi2html-1.8.0-py3-none-any.whl (16 kB)
Requirement already satisfied: ipykernel in /usr/local/lib/python3.8/dist-packages (from jupyter-dash) (5.3.4
Requirement already satisfied: flask in /usr/local/lib/python3.8/dist-packages (from jupyter-dash) (1.1.4)
Collecting nest-asyncio
  Downloading nest_asyncio-1.5.6-py3-none-any.whl (5.2 kB)
Collecting retrying
  Downloading retrying-1.3.4-py3-none-any.whl (11 kB)
Requirement already satisfied: ipython in /usr/local/lib/python3.8/dist-packages (from jupyter-dash) (7.9.0)
Collecting dash
  Downloading dash-2.7.1-py3-none-any.whl (9.9 MB)
     |████████████████████████████████| 9.9 MB 7.6 MB/s
Collecting dash-html-components==2.0.0
  Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Requirement already satisfied: plotly>=5.0.0 in /usr/local/lib/python3.8/dist-packages (from dash->jupyter-da
Collecting dash-table==5.0.0
  Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
Collecting dash-core-components==2.0.0
  Downloading dash_core_components-2.0.0-py3-none-any.whl (3.8 kB)
Requirement already satisfied: Werkzeug<2.0,>=0.15 in /usr/local/lib/python3.8/dist-packages (from flask->jup
Requirement already satisfied: Jinja2<3.0,>=2.10.1 in /usr/local/lib/python3.8/dist-packages (from flask->jup
Requirement already satisfied: itsdangerous<2.0,>=0.24 in /usr/local/lib/python3.8/dist-packages (from flask-
Requirement already satisfied: click<8.0,>=5.1 in /usr/local/lib/python3.8/dist-packages (from flask->jupyter
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.8/dist-packages (from Jinja2<3.0,>=
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.8/dist-packages (from plotly>=5.0.0-
Requirement already satisfied: six in /usr/local/lib/python3.8/dist-packages (from plotly>=5.0.0->dash->jupyt
Requirement already satisfied: traitlets>=4.1.0 in /usr/local/lib/python3.8/dist-packages (from ipykernel->ju
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.8/dist-packages (from ipykernel->jupy
Requirement already satisfied: tornado>=4.2 in /usr/local/lib/python3.8/dist-packages (from ipykernel->jupyte
Requirement already satisfied: pygments in /usr/local/lib/python3.8/dist-packages (from ipython->jupyter-dash
Requirement already satisfied: decorator in /usr/local/lib/python3.8/dist-packages (from ipython->jupyter-das
Requirement already satisfied: backcall in /usr/local/lib/python3.8/dist-packages (from ipython->jupyter-dash
Requirement already satisfied: pickleshare in /usr/local/lib/python3.8/dist-packages (from ipython->jupyter-d
Collecting jedi>=0.10
  Downloading jedi-0.18.2-py2.py3-none-any.whl (1.6 MB)
     |████████████████████████████████| 1.6 MB 47.4 MB/s
Requirement already satisfied: prompt-toolkit<2.1.0,>=2.0.0 in /usr/local/lib/python3.8/dist-packages (from i
Requirement already satisfied: pexpect in /usr/local/lib/python3.8/dist-packages (from ipython->jupyter-dash)
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.8/dist-packages (from ipython->jupy
Requirement already satisfied: parso<0.9.0,>=0.8.0 in /usr/local/lib/python3.8/dist-packages (from jedi>=0.10
Requirement already satisfied: wcwidth in /usr/local/lib/python3.8/dist-packages (from prompt-toolkit<2.1.0,>
Requirement already satisfied: jupyter-core>=4.6.0 in /usr/local/lib/python3.8/dist-packages (from jupyter-cl
Requirement already satisfied: pyzmq>=13 in /usr/local/lib/python3.8/dist-packages (from jupyter-client->ipyk
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.8/dist-packages (from jupyter-c
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.8/dist-packages (from jupyter-core
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.8/dist-packages (from pexpect->ipyth
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.8/dist-packa
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests->ju
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests->jupyter
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests->j
Installing collected packages: jedi, dash-table, dash-html-components, dash-core-components, retrying, nest-a
Successfully installed ansi2html-1.8.0 dash-2.7.1 dash-core-components-2.0.0 dash-html-components-2.0.0 dash-
```



FEATURE ENGINEERING

```
display(df.info())
display(df.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9551 entries, 0 to 9550
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Restaurant       9550 non-null   object
 1   Country Code     9551 non-null   int64
 2   City             9551 non-null   object
 3   Longitude        9551 non-null   float64
 4   Latitude         9551 non-null   float64
 5   Cuisines         9551 non-null   object
 6   Cost             9551 non-null   int64
 7   Currency         9551 non-null   object
 8   Booking          9551 non-null   object
 9   Delivery         9551 non-null   object
 10  Price range      9551 non-null   int64
 11  Aggregate rating 9551 non-null   float64
 12  Rating color     9551 non-null   object
 13  Rating text      9551 non-null   object
 14  Votes            9551 non-null   int64
 15  Country          9551 non-null   object
 16  Total_Cuisines   9551 non-null   int64
dtypes: float64(3), int64(5), object(9)
memory usage: 1.6+ MB
None
(9551, 17)
```

Dropping unnecessary columns mefore modelling

```
df.drop(columns=['Restaurant','Country Code','Longitude', 'Latitude','Cuisines','Rating color', 'Currency'], axis=1, inplace=True)
df.shape
```

```
(9551, 10)
```

Encoding

—> New Data

```
# Copying DF as df1 for Encoding
df1=df.copy()
```

—> One-Hot Encoding for Nominal Categories

```
# ENCODING
df1['Delivery1']=pr.LabelEncoder().fit_transform(df['Delivery'])
df1['Booking1']=pr.LabelEncoder().fit_transform(df['Booking'])
df1['Country1']=pr.LabelEncoder().fit_transform(df['Country'])
df1['City1']=pr.LabelEncoder().fit_transform(df['City'])


# GET DUMMY FOR preventing Logistic Regrression result
OneHot= pd.get_dummies(df1, columns=['Delivery', 'Booking', 'Country', 'City'], drop_first=True)
print(OneHot)

# dropping Columns that encoded
df1.drop(columns=['Delivery','Booking','Country', 'City'], axis=1, inplace=True)

print(df1)
```

```
4               0          0                    0                   0
...           ...        ...                  ...                 ...
9546            0          0                    0                   0
9547            0          0                    0                   0
9548            0          0                    0                   0
9549            0          0                    0                   0
9550            0          0                    0                   0

      City_Yorkton  City_€¡stanbul
0                0               0
1                0               0
2                0               0
3                0               0
4                0               0
...            ...             ...
9546             0               0
9547             0               0
9548             0               0
9549             0               0
9550             0               0

[9551 rows x 166 columns]
        Cost  Price range  Aggregate rating Rating text  Votes  \
0     800000            3               4.1   Very Good   1498
1     800000            3               4.6   Excellent    873
2     500000            3               4.9   Excellent    605
3     450000            3               4.2   Very Good    395
4     350000            3               4.3   Very Good    243
...      ...          ...               ...         ...    ...
9546       0            1               4.3   Very Good     63
9547       0            1               3.6        Good     34
9548       0            1               3.9        Good    127
9549       0            1               3.8        Good     78
9550       0            1               3.5        Good    109

      Total_Cuisines  Delivery1  Booking1  Country1  City1
0                  2          0         0         4     59
1                  3          0         0         4     59
2                  2          0         0         4     59
3                  1          0         0         4     59
4                  2          0         0         4     59
...              ...        ...       ...       ...    ...
9546               3          0         0         3     35
9547               3          0         0         3     61
9548               6          0         0         3     61
9549               1          0         0         3    130
9550               2          0         0         3    130

[9551 rows x 10 columns]
```

—> One-Hot Encoding for ORDINAL Categories

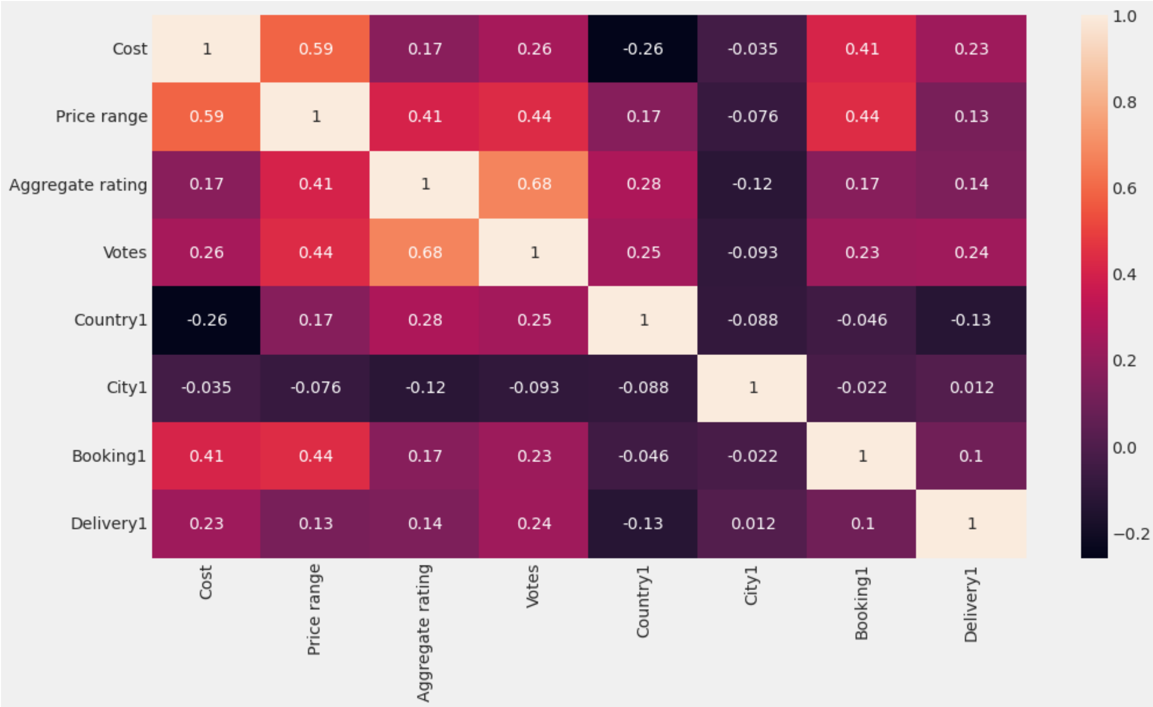
```
1 Rating_text1=pd.Categorical(df1['Rating text'], categories =['Excellent','Very Good','Good','Average','Not Rated'], ordered=True)
2 df['Rating_text'], Rate=pd.factorize(Rating_text1, sort=True)
3 df['Rating_text']
4
```

```
0       1
1       0
2       0
3       1
4       1
       ..
9546    1
9547    2
9548    2
9549    2
9550    2
Name: Rating_text, Length: 9551, dtype: int64
```

```
1 df1.drop(columns=['Rating text'], axis=1, inplace=True)
```

```
1 corr= df1[['Cost','Price range','Aggregate rating','Votes', 'Country1', 'City1','Booking1', 'Delivery1']].corr(method='kendall')
2 plt.figure(figsize=(15,8))
3 sns.heatmap(corr, annot=True)
4 df1.columns
```

```
Index(['Cost', 'Price range', 'Aggregate rating', 'Votes', 'Total_Cuisines',
       'Delivery1', 'Booking1', 'Country1', 'City1'],
      dtype='object')
```




```
# Discretizing the ratings into a categorical feature with 4 classes
#df["Aggregate rating"] = pd.cut(df["Aggregate rating"], bins = [0, 3.0, 3.5, 4.0, 5.0], labels = ["0", "1", "2", "3"])
#df["Aggregate rating"]
```

```
df1
```

1 to 25 of 9551 entries Filter

index	Cost	Price range	Aggregate rating	Votes	Total_Cuisines	Delivery1	Booking1	Country1	City1
0	800000	3	4.1	1498	2	0	0	4	59
1	800000	3	4.6	873	3	0	0	4	59
2	500000	3	4.9	605	2	0	0	4	59
3	450000	3	4.2	395	1	0	0	4	59
4	350000	3	4.3	243	2	0	0	4	59
5	300000	3	4.3	458	2	0	0	4	59
6	300000	3	3.7	155	1	0	0	4	125
7	250000	3	4.0	1159	2	0	0	4	19
8	250000	3	4.2	259	3	0	0	4	59
9	200000	3	4.9	1662	2	0	0	4	59
10	200000	3	3.9	137	1	0	0	4	59
11	200000	3	4.6	903	3	0	0	4	59
12	200000	3	4.4	841	3	0	0	4	59
13	200000	3	4.9	1640	2	0	0	4	59
14	200000	3	4.9	2212	2	0	0	4	125
15	165000	3	4.6	1476	5	0	0	4	59
16	150000	3	4.2	22	3	0	0	4	13
17	120000	3	4.4	410	1	0	0	4	59
18	100000	3	3.4	152	2	0	0	4	59
19	100000	3	4.0	331	2	0	0	4	59
20	70000	2	3.7	783	3	0	0	4	19
21	8000	4	4.0	145	1	0	1	3	88
22	7000	4	4.1	188	5	0	0	3	88
23	6500	4	4.4	2826	1	0	0	3	88
24	6000	4	4.9	621	3	0	1	6	94

Show 25 per page

1 2 10 100 300 380 383

```
df1.to_excel("data_output.xlsx", index=False)
```

MODEL BUILDING for REGRESSION

```
#lets create 2 dataframes in which one has target variable (i.e. Rating) and latter has predictor variables
# X datframe has predictor variables while y datframe has target variable

x=df1.drop(columns='Aggregate rating',axis=1)
y=df1['Aggregate rating']
```

Splitting the Model

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1 , random_state=42)
```

Standardization

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
```

Defining the Model Success

```
def model_predict(model):
  model.fit(x_train , y_train)
  predict=model.predict(x_test)
  r2=mt.r2_score(y_test, predict)
  rmse=mt.mean_squared_error(y_test, predict, squared=False)
  return[r2, rmse]

print(model_predict(LinearRegression()))
```

```
[0.3287777508810622, 1.2514204063230165]
```

Lets build and compare with other REGRESSION models

```
models=[LinearRegression(),Ridge(),Lasso(), ElasticNet(), SVR(), DecisionTreeRegressor(random_state=0),BaggingRegressor(random_state=0 , n_estimators=2

model_names = ['LinearRegression','Ridge','Lasso', 'ElasticNet', 'SVR', 'DecisionTreeRegressor','BaggingRegressor', 'RandomForestRegressor']
results=[]

for i in models:
  results.append(model_predict(i))
#print(results)


df1_models=pd.DataFrame(model_names, columns=['Model Name'])
print(df1_models)
```

```
             Model Name
0      LinearRegression
1                 Ridge
2                 Lasso
3            ElasticNet
4                   SVR
5 DecisionTreeRegressor
6      BaggingRegressor
7 RandomForestRegressor
```

```
df1_results=pd.DataFrame(results, columns=['R2', 'RMSE'])
print(df1_results)
```

```
         R2      RMSE
0  0.328778  1.251420
1  0.328776  1.251422
2 -0.000060  1.527506
3  0.059071  1.481659
4  0.440272  1.142769
5  0.922214  0.426011
6  0.956145  0.319876
7  0.958268  0.312036
```

Models' Results and Best Model

```
df1_comparison=df1_models.join(df1_results)
print(df1_comparison)
```

```
              Model Name        R2      RMSE
0       LinearRegression  0.328778  1.251420
1                  Ridge  0.328776  1.251422
2                  Lasso -0.000060  1.527506
3             ElasticNet  0.059071  1.481659
4                    SVR  0.440272  1.142769
5  DecisionTreeRegressor  0.922214  0.426011
6       BaggingRegressor  0.956145  0.319876
7  RandomForestRegressor  0.958268  0.312036
```

Best Model is RANDOM FOREST REGRESSOR

HYPERTUNING of Best Model

—> Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor

RF=RandomForestRegressor(random_state=0 )

RF.fit(x_train , y_train)
RF_predict = RF.predict(x_test)

print(RF_predict)
```

```
[3.635      4.018      3.759      2.90954603 4.183      3.3046
 0.         3.153      0.         3.74408333 3.4525     3.906
 3.54833333 4.444      0.         4.094      3.16175    3.4
 3.524      2.99591667 3.449      0.         3.33794286 2.901
 3.094      2.96353333 0.         3.439      3.1092119  2.97361667
 0.         3.56605    3.33916667 0.         2.92503095 3.923
 3.1335     4.217      2.98411084 4.523      3.694      0.
 3.521      3.858      3.3415     2.94483333 0.         3.45
 0.         0.         3.955      3.25066667 3.16723333 0.
 3.55       3.301      4.062      3.679      3.06228571 3.25972381
 3.071      3.6825     4.17       0.         2.714      3.0802
 2.99483023 2.618      0.         0.         0.         3.78525
 3.457      3.16650317 0.         0.         0.         3.27333333
 0.         3.758      4.035      3.5755     0.         4.432
 3.397      3.189      3.56       2.9033558  3.4675     4.125
 4.545      0.         3.085      3.628      4.369      0.
 0.         2.97519008 3.996      3.2098     3.456      0.
 3.243      3.0765     0.         3.29171429 3.00347619 3.688
 0.         2.847      3.42125    3.178      2.7494     2.94271905
 2.85402381 3.293      3.685      3.612      3.222      3.32133333
 0.         2.76156667 3.16913333 4.157      3.405      0.
 2.612      3.0166972  0.         0.         3.01962517 3.137
 3.14426667 0.         3.454      0.         3.653      0.
 2.71941667 3.376      3.065      4.274      3.06608333 2.98211848
 3.33975    3.905      4.821      3.329      3.759      3.512
 4.25       2.825      3.1165     3.499      0.         0.
 4.083      2.97402002 0.         3.36883333 3.4032     2.9301039
 3.983      3.42173333 3.993      3.307      0.         3.961
 3.66       0.         3.792      3.72       2.89248571 3.847
 3.00377543 2.659      3.196      0.         2.90276667 3.059
 0.         3.123      0.         3.08126667 2.99395714 3.608
 4.388      4.773      0.         3.157      3.26675    2.99281667
 3.798      0.         3.865      3.635      3.01633333 0.
 3.311      0.         0.         3.238      0.         3.662
 0.         3.306      3.10043561 0.         3.37933333 4.12
 3.37166667 2.99235971 3.543      3.06526667 4.143      2.90309048
 3.833      2.83992857 0.         3.803      0.         0.
 3.278      2.96292381 3.36706667 0.         3.946      3.629
 3.19001032 0.         3.677      4.485      0.         3.9
 4.178      3.069      3.349      3.41458333 3.133      3.968
 3.48385714 4.183      3.53271667 2.99684048 3.04205    2.903
 2.95345    3.354      3.565      3.4675     0.         3.593
 3.29       3.0275     0.         3.427      4.334      0.
 3.988      2.612      2.86141667 0.         3.07770476 3.639
 3.658      2.92461667 3.09868333 3.414      3.097      3.256
 3.24468333 3.3725     2.98863333 3.534      3.572      3.073
 3.487      2.904      3.572      4.114      2.8998     0.
 2.98059643 3.338      3.20766667 3.05506587 3.2401     3.41433333
 3.35536667 2.93028333 0.         0.         0.         3.893
 3.789      3.39866667 2.88047619 3.2162     4.289      3.344
 3.59848333 3.431      3.886      3.303      0.         0.
```

```
 4.27       3.807      3.22013333 3.658      2.92236667 3.561
 3.02063333 4.005      3.44941667 3.748      3.22       0.
 3.25532237 0.         0.         3.6259     3.16833333 3.407
 0.         3.576      3.17324524 3.79533333 4.082      3.486
 3.53       2.974      3.1505     4.12       0.         3.936
 2.862      3.18286667 3.944      0.         2.8416     3.83333333
 4.081      3.3395381  2.8846     4.579      3.0455     2.92024405
```

```
RF_R2= mt.r2_score(y_test, RF_predict)

RF_RMSE= mt.mean_squared_error(y_test, RF_predict , squared=False )

display ( 'RF R2 : ', RF_R2 , 'RF RMSE : ', RF_RMSE )
```

```
'RF R2 : '
0.958268036890316
'RF RMSE : '
0.31203583191641365
```

>>> I will use GridSearch to find the best Tuning - HYPERPARAMETER TUNING

```
from sklearn.model_selection import GridSearchCV
```

it will take long . You will see the better Hyperparameter to have better result the below

```
#RF_parameters={"max_depth": range(2,25), 'max_features': range(2,25),"n_estimators": range(2,25) }
#RF_Grid = GridSearchCV(estimator=RF, param_grid= RF_parameters , cv=10 )
#RF_Grid .fit(x_train , y_train)
#print( RF_Grid.best_params_)
```

**Random Forest Regressor Parameter metrics after Model Tuning* ——>*

'max_depth': 13, 'max_features': 2, 'n_estimators': 19

```
RF2=RandomForestRegressor(random_state=0 , max_depth= 13, max_features= 2, n_estimators= 19)
RF2.fit(x_train , y_train)
RF2_predict = RF2.predict(x_test)
#print(RF2_predict)

RF2_R2= mt.r2_score(y_test, RF2_predict)
RF2_RMSE= mt.mean_squared_error(y_test, RF2_predict , squared=False )
display ( 'RF2 R2 : ', RF2_R2 , 'RF2 RMSE : ', RF2_RMSE )
```

```
'RF2 R2 : '
0.9635364532333441
'RF2 RMSE : '
0.2916752075229647
```

+ Code + Text

==

FINAL RESULTS OF BEST TESTING OF THE MODEL

```
> **First Results were**

*   RandomForestRegressor     R2  : 0.958268       RMSE :  0.312036

> Second Results after HyperTuning

*   RandomForestRegressor     R2  :  0.963536     RMSE : 0.291675
```

> **First Results were**

- RandomForestRegressor R2 : 0.958268 RMSE : 0.312036

> Second Results after HyperTuning

- RandomForestRegressor R^2 : 0.963536 RMSE : 0.291675

✓ 1s completed at 5:47 PM

B