

# Set Up

```
In [1]: # import modules and data
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

%matplotlib inline

countrycodes = pd.read_excel('Country-Code.xlsx')
restaurants = pd.read_excel('data.xlsx')

# merge datasets
df = pd.merge(countrycodes, restaurants, on='Country Code')
```

```
In [2]: df.head(2)
```

```
Out[2]:
```

	Country Code	Country	Restaurant ID	Restaurant Name	City	Address	Locality	Locality Verbose	Longitude	Latitude
0	1	India	2701	Orient Express - Taj Palace Hotel	New Delhi	Taj Palace Hotel, Diplomatic Enclave, Chanakya...	The Taj Palace Hotel, Chanakyapuri	The Taj Palace Hotel, Chanakyapuri, New Delhi	77.170087	28.5950
1	1	India	309548	Tian - Asian Cuisine Studio - ITC Maurya	New Delhi	ITC Maurya, Diplomatic Enclave, Chanakyapuri, ...	ITC Maurya, Chanakyapuri	ITC Maurya, Chanakyapuri, New Delhi	77.173455	28.5973

```
In [3]: # summary of dataframe
df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 9551 entries, 0 to 9550
Data columns (total 20 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Country Code                          9551 non-null   int64
1   Country                               9551 non-null   object
2   Restaurant ID                         9551 non-null   int64
3   Restaurant Name                       9550 non-null   object
4   City                                  9551 non-null   object
5   Address                               9551 non-null   object
6   Locality                             9551 non-null   object
7   Locality Verbose                     9551 non-null   object
8   Longitude                             9551 non-null   float64
9   Latitude                             9551 non-null   float64
10  Cuisines                              9542 non-null   object
11  Average Cost for two                  9551 non-null   int64
12  Currency                              9551 non-null   object
13  Has Table booking                     9551 non-null   object
14  Has Online delivery                   9551 non-null   object
15  Price range                           9551 non-null   int64
16  Aggregate rating                      9551 non-null   float64
```

```
17 Rating color          9551 non-null object
18 Rating text           9551 non-null object
19 Votes                 9551 non-null int64
dtypes: float64(3), int64(5), object(12)
memory usage: 1.5+ MB
```

- the data has 20 columns and 9551 rows including headers
- cuisines appear to have an extra row
- datatypes appear correct

## Data Wrangling

```
In [4]: # format columns to lower case
df.columns = df.columns.str.lower()

# remove spacing in columns
df.columns = df.columns.str.replace(' ', '_')
```

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

```
Out[5]: Index(['countrycode', 'country', 'restaurantid', 'restaurantname', 'city',
              'address', 'locality', 'localityverbose', 'longitude', 'latitude',
              'cuisines', 'averagecostfortwo', 'currency', 'hastablebooking',
              'hasonlinedelivery', 'pricerange', 'aggregaterating', 'ratingcolor',
              'ratingtext', 'votes'],
              dtype='object')
```

```
In [6]: # format to concise column names
df = df.rename(columns={'averagecostfortwo':'averagecost', 'hastablebooking':'tablebooki
```

```
In [7]: # convert datatype for feature engineering
df.countrycode = df.countrycode.apply(str)
```

```
In [8]: # check for duplicates
df.duplicated().any()
```

```
Out[8]: False
```

```
In [9]: # % of missing values to assess most suitable treatment
round(df.isnull().sum().sort_values(ascending=False)/len(df)*100,2)
```

```
Out[9]: cuisines          0.09
restaurantname  0.01
averagecost    0.00
ratingtext     0.00
ratingcolor    0.00
rating         0.00
pricerange     0.00
onlinedelivery 0.00
tablebooking   0.00
currency       0.00
countrycode    0.00
country        0.00
latitude       0.00
longitude      0.00
localityverbose 0.00
locality       0.00
address        0.00
city           0.00
restaurantid   0.00
```

```
votes          0.00
dtype: float64
```

```
In [10]: # Arbitrary Imputation of missing values
df.restaurantname.replace([np.nan], 'N/A - Missing Value', inplace=True)
```

```
In [11]: # fill cuisine missing values with mode
df.fillna(value={'cuisines':df['cuisines'].mode()[0]}, inplace=True)
```

```
In [12]: # final missing value check
df.isnull().sum()
```

```
Out[12]: countrycode      0
country      0
restaurantid  0
restaurantname  0
city         0
address      0
locality     0
localityverbose  0
longitude    0
latitude     0
cuisines     0
averagecost  0
currency     0
tablebooking 0
onlinedelivery 0
pricerange   0
rating       0
ratingcolor  0
ratingtext   0
votes        0
dtype: int64
```

```
In [13]: # correct country names
df.city.replace({'Brasî_lia':'Brasil Lia', 'Sîfo Paulo':'Sao Paulo', 'Eîstanbul':'Istanb
```

```
In [14]: # drop irrelevant columns
df.drop(['address', 'localityverbose'], axis=1, inplace=True)

# reset df
df.reset_index(drop=True, inplace=True)
```

## Exploratory Data Analysis

```
In [15]: # statistics of df
df.describe(include='all')
```

```
Out[15]:
```

	countrycode	country	restaurantid	restaurantname	city	locality	longitude	latitude	cuisine
count	9551	9551	9.551000e+03	9551	9551	9551	9551.000000	9551.000000	9551
unique	15	15	NaN	7446	141	1208	NaN	NaN	182
top	1	India	NaN	Cafe Coffee Day	New Delhi	Connaught Place	NaN	NaN	North India
freq	8652	8652	NaN	83	5473	122	NaN	NaN	94
mean	NaN	NaN	9.051128e+06	NaN	NaN	NaN	64.126574	25.854381	Na
std	NaN	NaN	8.791521e+06	NaN	NaN	NaN	41.467058	11.007935	Na
min	NaN	NaN	5.300000e+01	NaN	NaN	NaN	-157.948486	-41.330428	Na

<b>25%</b>	NaN	NaN	3.019625e+05	NaN	NaN	NaN	77.081343	28.478713	Na
<b>50%</b>	NaN	NaN	6.004089e+06	NaN	NaN	NaN	77.191964	28.570469	Na
<b>75%</b>	NaN	NaN	1.835229e+07	NaN	NaN	NaN	77.282006	28.642758	Na
<b>max</b>	NaN	NaN	1.850065e+07	NaN	NaN	NaN	174.832089	55.976980	Na

- There are 15 unique countries, with India being the most common country, appearing 8652 times
- There are 140 unique cities
- 'Cafe Coffee Day' is the most common 'restaurantname', implying there should be numerous branches
- New Delhi is the most common 'city' with 5473 restaurants
- North Indian is the most common 'cuisine'
- 'Orange' and 'Average' are the most common rating indicators
- The max 'average rating' is 4.9

```
In [16]: # no. of unique restaurantid
df.restaurantid.nunique()
```

```
Out[16]: 9551
```

```
In [17]: # no. of unique restaurantname
df.restaurantname.nunique()
```

```
Out[17]: 7446
```

## Geographical Distribution

### Total Resturants by Countries

```
In [18]: # restaurant count and % by country
geodf = df.groupby(['country'], as_index=False)['restaurantid'].count()
geodf.rename(columns={'restaurantid': 'totalrestaurants'}, inplace=True)
geodf['percent'] = (geodf.totalrestaurants/geodf.totalrestaurants.sum()*100).round(1)

geodf.sort_values(by='percent', ascending=False)
```

```
Out[18]:
```

	country	totalrestaurants	percent
<b>3</b>	India	8652	90.6
<b>14</b>	United States	434	4.5
<b>13</b>	United Kingdom	80	0.8
<b>1</b>	Brazil	60	0.6
<b>9</b>	South Africa	60	0.6
<b>12</b>	UAE	60	0.6
<b>5</b>	New Zealand	40	0.4
<b>11</b>	Turkey	34	0.4
<b>0</b>	Australia	24	0.3
<b>4</b>	Indonesia	21	0.2
<b>6</b>	Phillipines	22	0.2

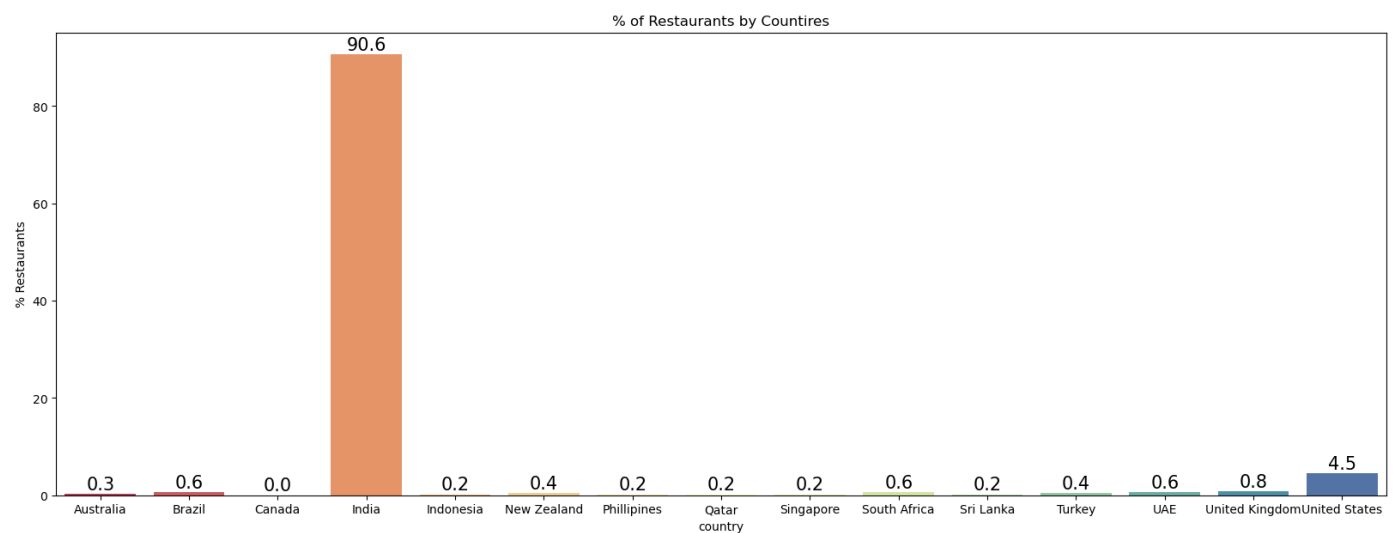
7	Qatar	20	0.2
8	Singapore	20	0.2
10	Sri Lanka	20	0.2
2	Canada	4	0.0

In [19]: *# Barplot of resturants vs countires*

```
plt.figure(figsize = (20,7))
plot_annotate = sns.barplot(geodf, x='country', y='percent', palette='Spectral')

# annotate %
for bar in plot_annotate.patches:
    plot_annotate.annotate(format(bar.get_height(), '.1f'),
                           (bar.get_x() + bar.get_width()/2,
                            bar.get_height()), ha='center', va='center', size=15,
                           xytext=(0,8), textcoords='offset points')

plt.ylabel('% Restaurants')
plt.title('% of Restaurants by Countires')
plt.show()
```



## Top 10 Cities with most Restaurants

In [20]: `city_df = df.groupby(['city'], as_index=False)['restaurantid'].count()  
city_df.rename(columns={'restaurantid': 'totalrestaurants'}, inplace=True)  
city_df['percent'] = (city_df.totalrestaurants/sum(city_df.totalrestaurants)*100).round(`

In [21]: `top_cities = city_df.nlargest(10, ['totalrestaurants'])  
top_cities`

Out[21]:

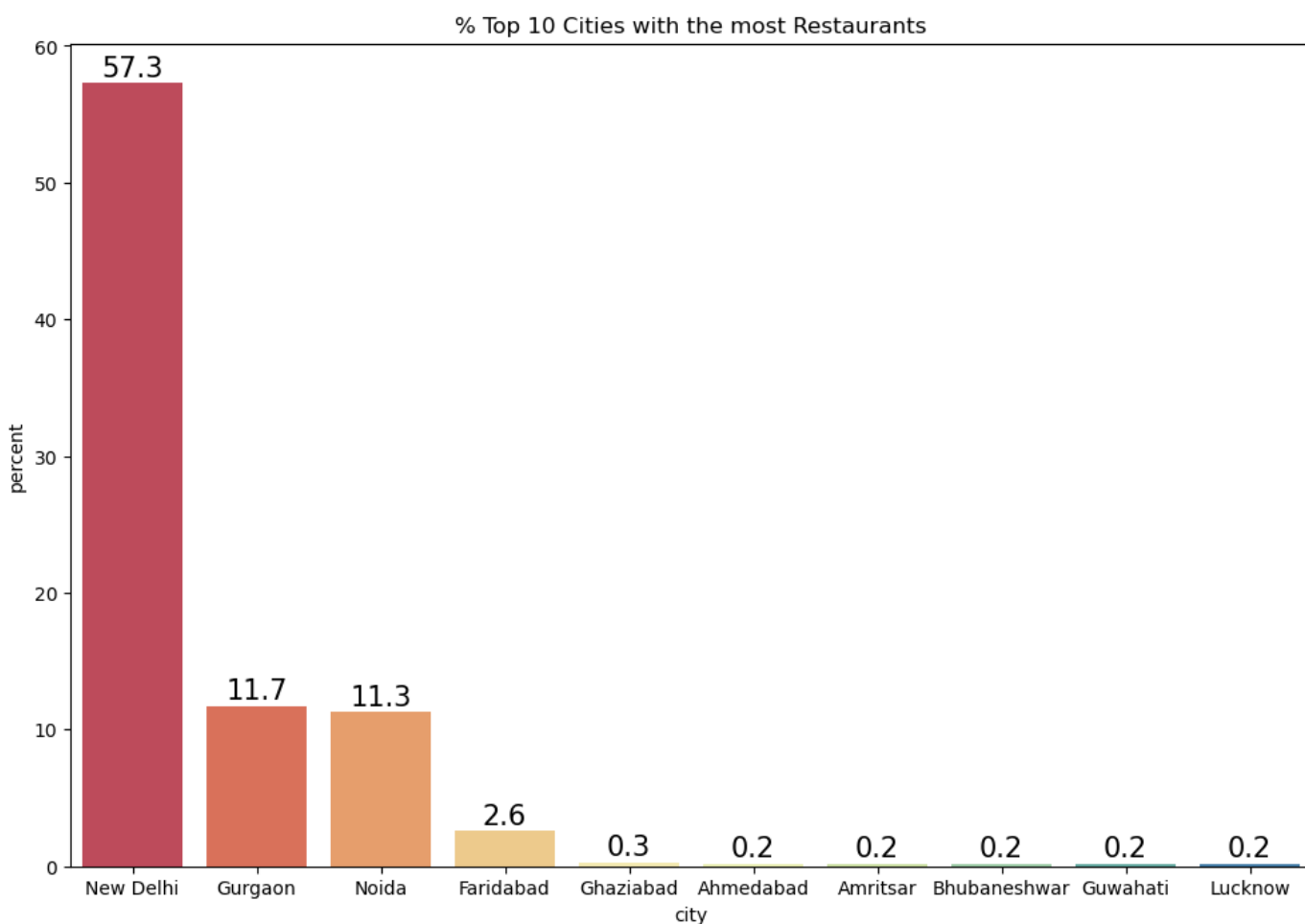
	city	totalrestaurants	percent
89	New Delhi	5473	57.3
50	Gurgaon	1118	11.7
90	Noida	1080	11.3
43	Faridabad	251	2.6
48	Ghaziabad	25	0.3
2	Ahmedabad	21	0.2

5	Amritsar	21	0.2
17	Bhubaneswar	21	0.2
51	Guwahati	21	0.2
70	Lucknow	21	0.2

```
In [22]: plt.figure(figsize=(12,8))
graph = sns.barplot(top_cities, x='city', y='percent', palette='Spectral')

# annotate graph
for bar in graph.patches:
    graph.annotate(format(bar.get_height(), '.1f'),
                    (bar.get_x() + bar.get_width()/2,
                     bar.get_height()), ha='center',
                    va='center', size=15, xytext=(0,8), textcoords='offset points'
                    )

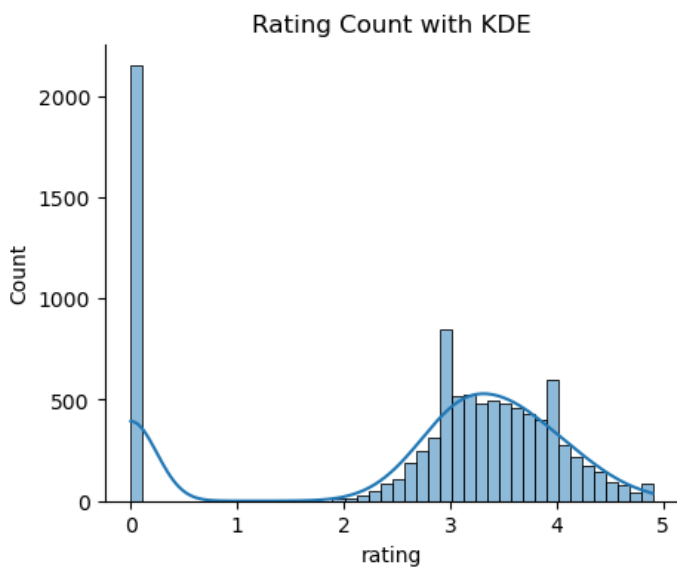
plt.title('% Top 10 Cities with the most Restaurants')
plt.show()
```



## Ratings Distribution

```
In [23]: fig, axes=plt.subplots(1,2,figsize=(11,4), sharey=False)
sns.histplot(df, x='rating', kde=True, palette='Spectral', ax=axes[0])
axes[0].set_title('Rating Count with KDE')
sns.kdeplot(df, x='rating', hue='ratingtext', ax=axes[1])
axes[1].set_title('Rating Text Plot')

sns.despine(right=True, top=True)
```



- Over 2000 ratings are 0, indicating no rating recorded
- Most valid ratings are between values 3 and 4
- 'Average' and 'Good' dominate 'textrating'
- 'Poor' and 'Excellent' appear almost equal in volume

## Largest Franchises

```
In [24]: franchise = df.groupby(['country', 'restaurantname'], as_index=False)['restaurantid'].count()
franchise.rename(columns={'restaurantid': 'restcount'}, inplace=True)
```

```
In [25]: # select top 10 largest franchises
top_franchise = franchise.nlargest(10, ['restcount'])
top_franchise
```

```
Out[25]:
```

	country	restaurantname	restcount
1061	India	Cafe Coffee Day	83
1975	India	Domino's Pizza	79
5523	India	Subway	63
2486	India	Green Chick Chop	51
3689	India	McDonald's	48
3169	India	Keventers	34
2408	India	Giani	29
4480	India	Pizza Hut	29
705	India	Baskin Robbins	28
690	India	Barbeque Nation	25

```
In [26]: # barplot
plt.figure(figsize=(12, 8))

plots = sns.barplot(top_franchise, x='restaurantname', y='restcount', palette='Spectral')

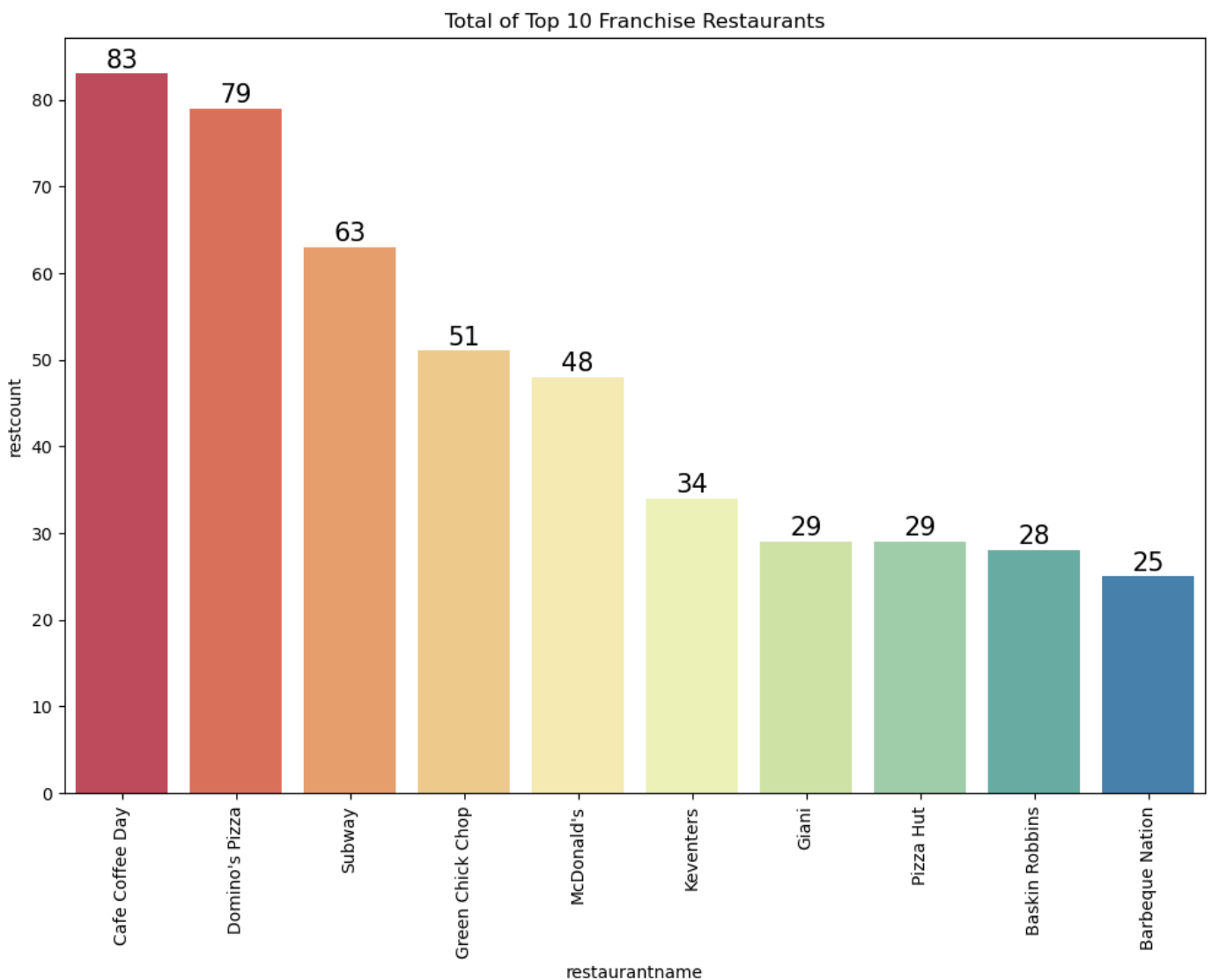
# annotate bars
for bar in plots.patches:
```

```

plots.annotate(format(bar.get_height(), '.0f'),
               (bar.get_x() + bar.get_width() / 2,
                bar.get_height()), ha='center', va='center',
               size=15, xytext=(0, 8),
               textcoords='offset points')

plt.xticks(rotation=90)
plt.title('Total of Top 10 Franchise Restaurants')
plt.show()

```



## Ratio of Table Booking Service

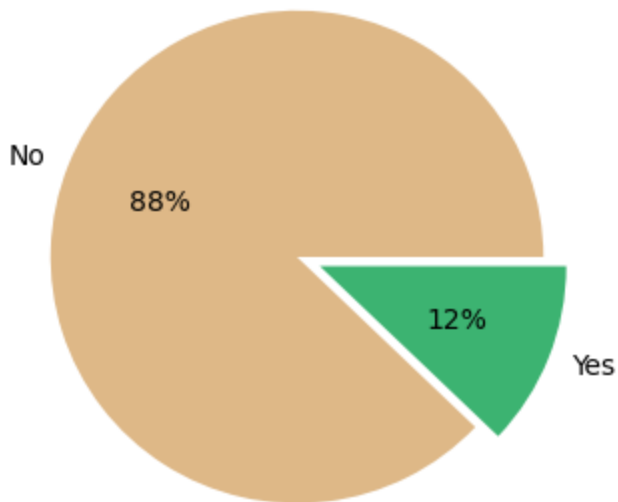
```

In [27]: plt.figure(figsize=(4,4))
plt.pie(x=df['tablebooking'].value_counts(), labels=df['tablebooking'].value_counts().in
        autopct='%0.0f%', explode=[0,0.1], colors=['burlywood', 'mediumseagreen'])
plt.title('Ratio of Table Booking Service')
plt.show()

```



## Ratio of Table Booking Service

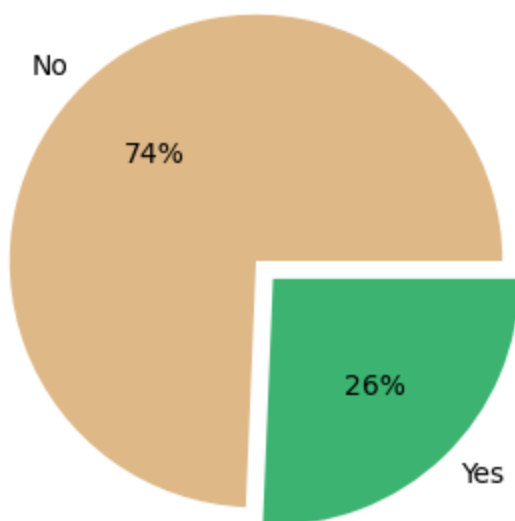


- The ratio of table booking services is approximately 9:1
- Majority of restaurants do not offer booking services

## % of Online Delivery Service

```
In [28]: plt.figure(figsize=(4,4))
plt.pie(x=df['onlinedelivery'].value_counts(), labels=df['onlinedelivery'].value_counts(
    autopct='%0f%%', explode=[0,0.1], colors=['burlywood', 'mediumseagreen'])
plt.title('% of Online Delivery Services')
plt.show()
```

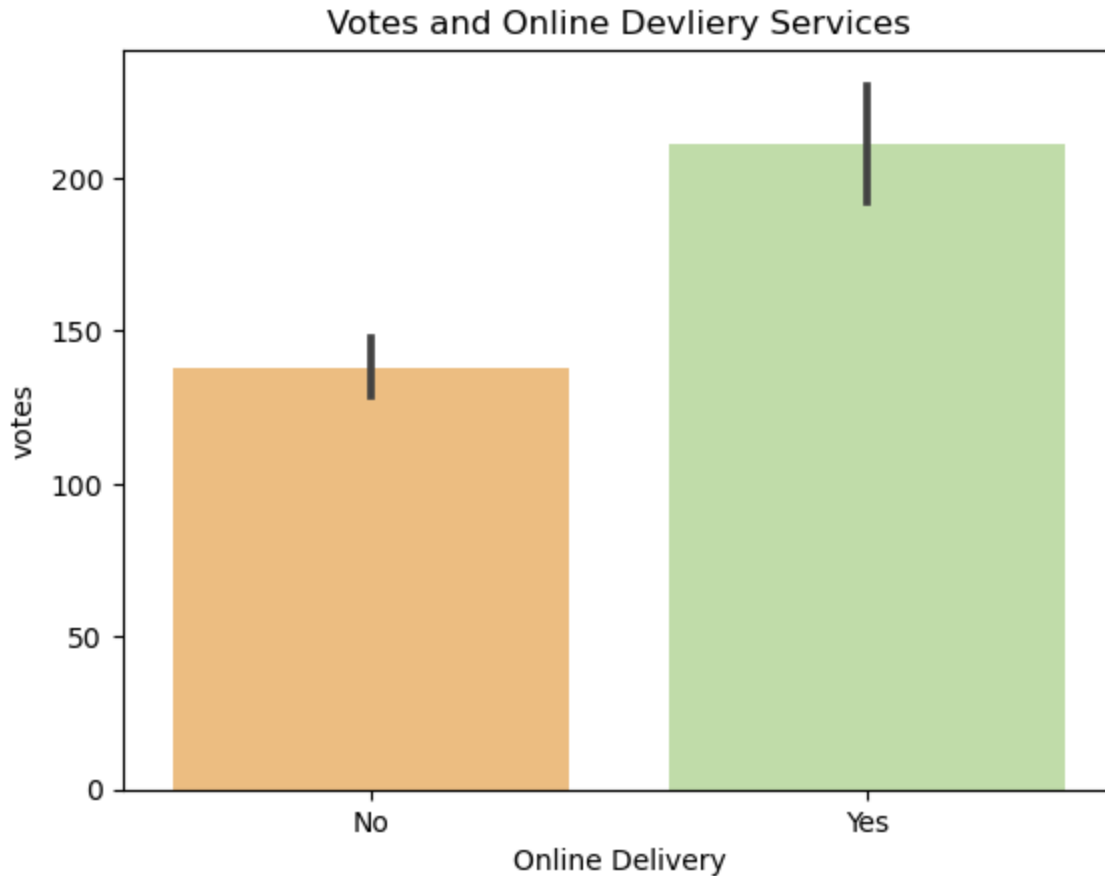
### % of Online Delivery Services



- Almost 75% of restaurants don't offer online delivery services
- Most restaurants do not provide delivery services

## Votes vs Online Delivery

```
In [29]: sns.barplot(df, x='onlinedelivery', y='votes', ci=95, palette='Spectral')
plt.title('Votes and Online Devliery Services')
plt.xlabel('Online Delivery')
plt.show()
```



- Higher volume of votes are seen for 'Yes' delivery services

## Top Cuisines

```
In [30]: # extract cuisines from string
l = []
for i in df.cuisines.str.split(', '):
    l.extend(i)
food = pd.Series([i.strip() for i in l])

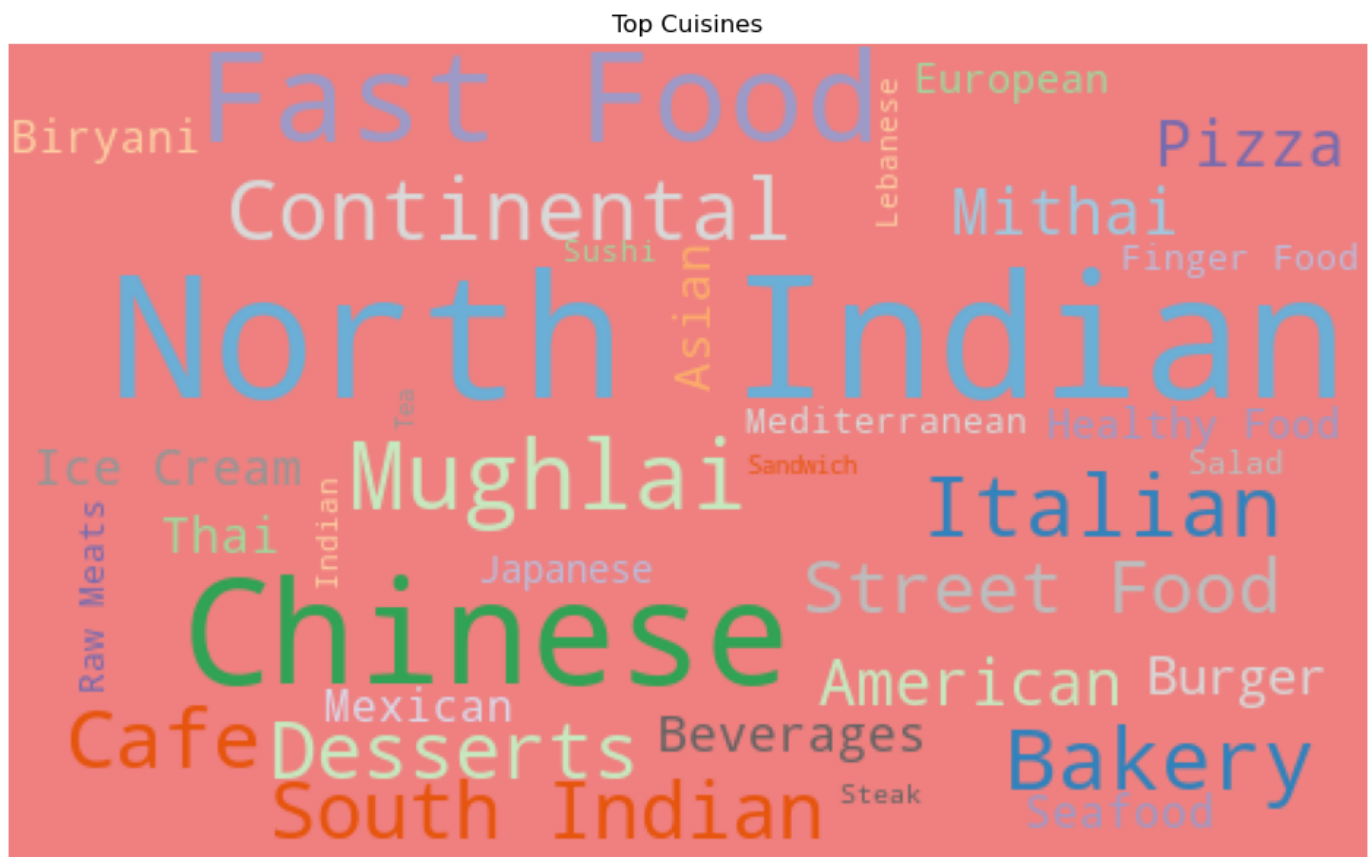
# list of food value counts
food.value_counts()
```

```
Out[30]: North Indian      3969
Chinese      2735
Fast Food    1986
Mughlai      995
Italian      764
...
Peranakan      1
Bì_rek         1
Dì_ner         1
Fish and Chips  1
Bubble Tea     1
Length: 145, dtype: int64
```

```
In [31]: from wordcloud import WordCloud, STOPWORDS
```

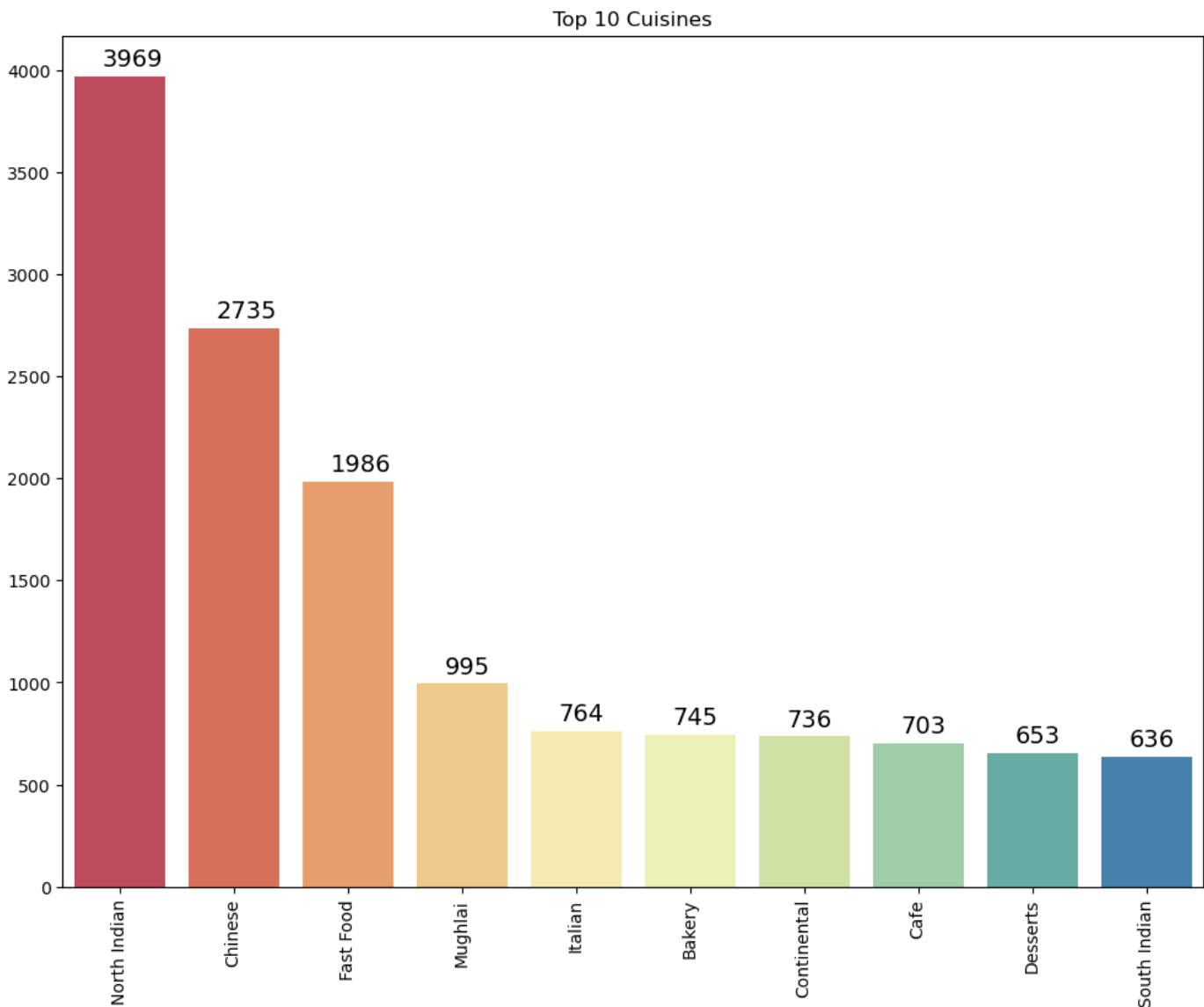
```
stopwords = set(STOPWORDS)
```

```
wordcloud = (WordCloud(width=500, height=300, random_state=1, background_color='lightcor',  
                        colormap='tab20c', stopwords=stopwords).generate_from_frequencies  
fig = plt.figure(1,figsize=(12, 10))  
plt.imshow(wordcloud)  
plt.title('Top Cuisines')  
plt.axis('off')  
plt.show()
```



## Top 10 Cuisines served in Restaurants

```
In [32]: plt.figure(figsize=(12,9))  
sns.barplot(x = food.value_counts()[:10].index, y = food.value_counts()[:10], palette='S  
  
for i in range(10):  
    plt.annotate(food.value_counts()[i], xy = (i-0.15,food.value_counts()[i]+50),  
                fontsize = 14)  
  
plt.xticks(rotation=90)  
plt.title('Top 10 Cuisines')  
plt.show()
```



## Total Cuisines served by Restaurant

```
In [33]: # create new column with cuisines count
df['cui_count'] = df.cuisines.apply(lambda x: len(x.split(', ')))
```

## Maximum Cuisines

```
In [34]: # select top 5 largest counts
maxfood = df.nlargest(5, ['cui_count'])
maxfood[['restaurantname', 'city', 'rating', 'averagecost', 'currency', 'cui_count']]
```

```
Out[34]:
```

	restaurantname	city	rating	averagecost	currency	cui_count
939	R' ADDA	Mumbai	4.0	1200	Indian Rupees(Rs.)	8
1200	Mumbai Vibe	Mumbai	3.8	1000	Indian Rupees(Rs.)	8
2716	Bikanervala	Gurgaon	3.4	600	Indian Rupees(Rs.)	8
3204	Indian Summer Cafe	Patna	3.4	600	Indian Rupees(Rs.)	8
3290	Bikanervala	New Delhi	3.5	550	Indian Rupees(Rs.)	8

- The maximum count of cuisines served in a resturant is 8

# Minimum Cuisines

```
In [35]: # select top smallest counts
minfood = df.nsmallest(5, ['cui_count'])
minfood[['restaurantname', 'city', 'rating', 'averagecost', 'currency', 'cui_count']]
```

```
Out[35]:
```

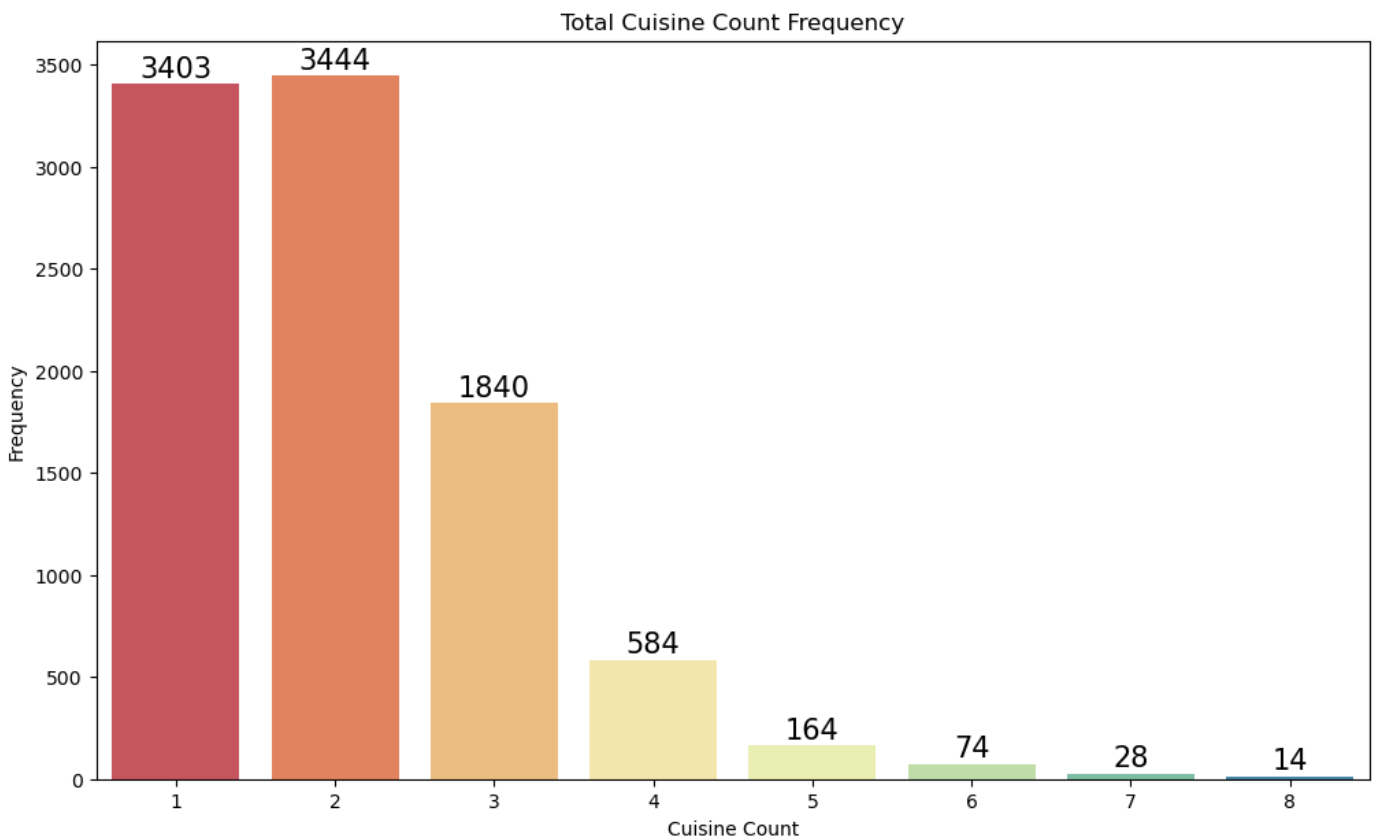
	restaurantname	city	rating	averagecost	currency	cui_count
0	Orient Express - Taj Palace Hotel	New Delhi	4.0	8000	Indian Rupees(Rs.)	1
2	Bukhara - ITC Maurya	New Delhi	4.4	6500	Indian Rupees(Rs.)	1
8	House of Ming - The Taj Mahal Hotel	New Delhi	4.0	5500	Indian Rupees(Rs.)	1
10	Wildfire - Crowne Plaza	Gurgaon	3.7	5000	Indian Rupees(Rs.)	1
12	Masala Library	New Delhi	4.9	5000	Indian Rupees(Rs.)	1

- The minimum amount of cuisines served is 1

## Frequency of Cuisine Counts

```
In [36]: plt.figure(figsize=(12,7))
plotting = sns.countplot(df, x='cui_count', palette='Spectral')
for bar in plotting.patches:
    plotting.annotate(format(bar.get_height(), '.0f'),
                      (bar.get_x() + bar.get_width() / 2,
                       bar.get_height()), ha='center', va='center',
                      size=15, xytext=(0, 8),
                      textcoords='offset points')

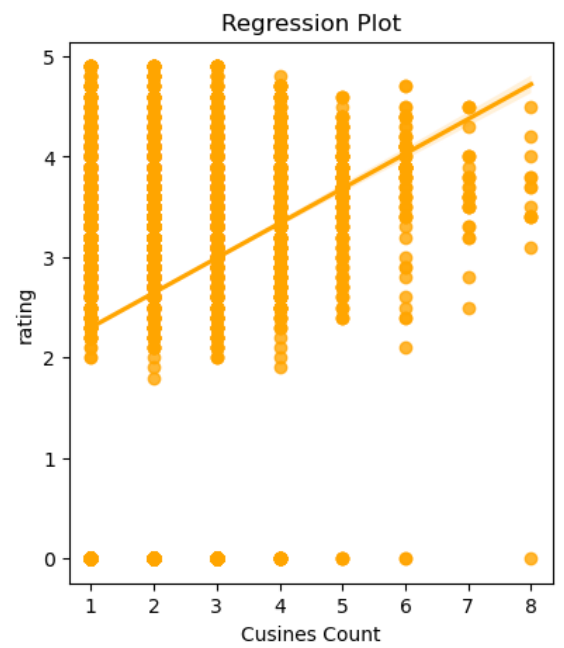
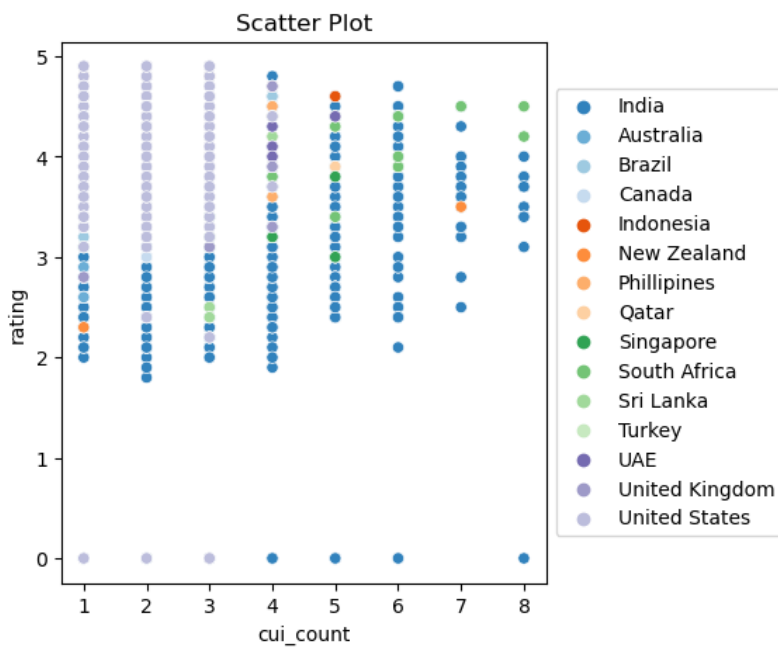
plt.title('Total Cuisine Count Frequency')
plt.xlabel('Cuisine Count')
plt.ylabel('Frequency')
plt.show()
```



## Relationship of Ratings vs No. of Cuisines

```
In [37]: fig, axes = plt.subplots(1,2,figsize=(12,5))
plt.subplots_adjust(hspace = 0.5 , wspace = 0.7)
sns.scatterplot(df, x= 'cui_count', y= 'rating', hue='country',
                ax = axes[0], palette = 'tab20c')
plt.xlabel('Cuisines Count')
axes[0].legend(loc='center left', bbox_to_anchor=(1, 0.5))
axes[0].set_title('Scatter Plot')

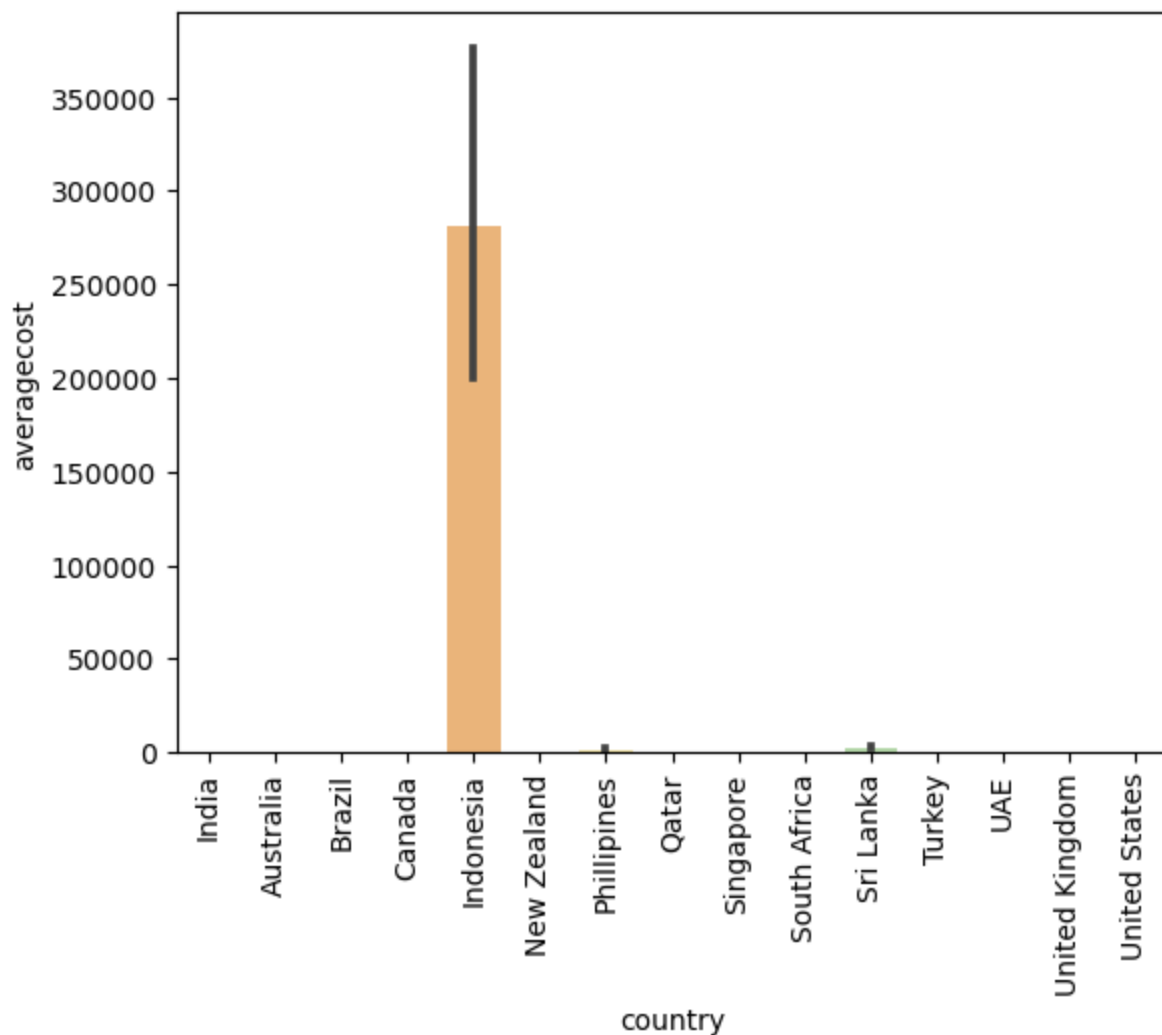
sns.regplot(df, x= 'cui_count', y= 'rating', color='orange', ax = axes[1], ci=80)
axes[1].set_title('Regression Plot')
plt.xlabel('Cusines Count')
plt.show()
```



- Restaurants serving less cuisines appear to receive higher average ratings than restaurants with more cuisines

## Average Cost by Country

```
In [38]: sns.barplot(x='country', y='averagecost', palette="Spectral", data=df)
plt.xticks(rotation='90')
plt.show()
```

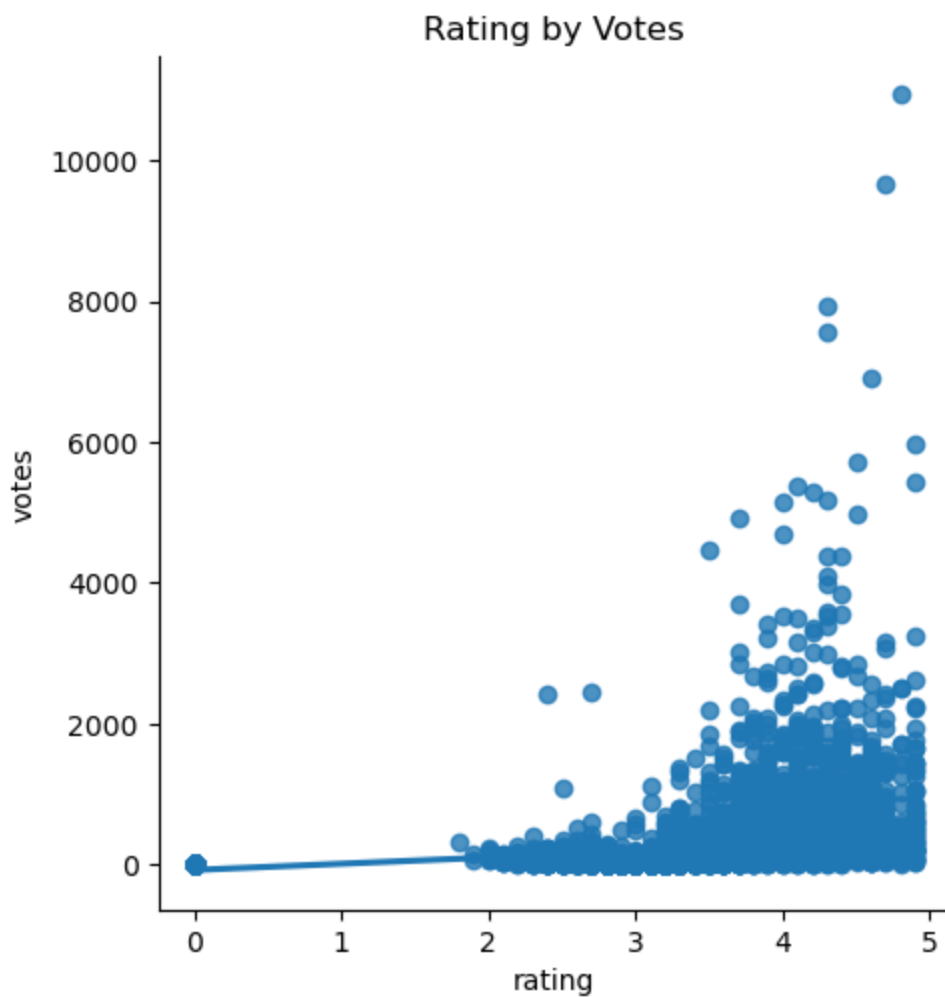


- it's difficult to compare the 'averagecost' amongst countries as each country's currency value and exchange rate is different
- Indonesia is appearing to have the highest 'averagecost' due to it's IDR currency which has many thousands but may not be equal in worth with other currencies

## Relationships of Ratings

```
In [39]: sns.lmplot(df, x='rating', y='votes')
plt.title('Rating by Votes')
plt.show()
```





- Higher ratings are received with higher volume of votes

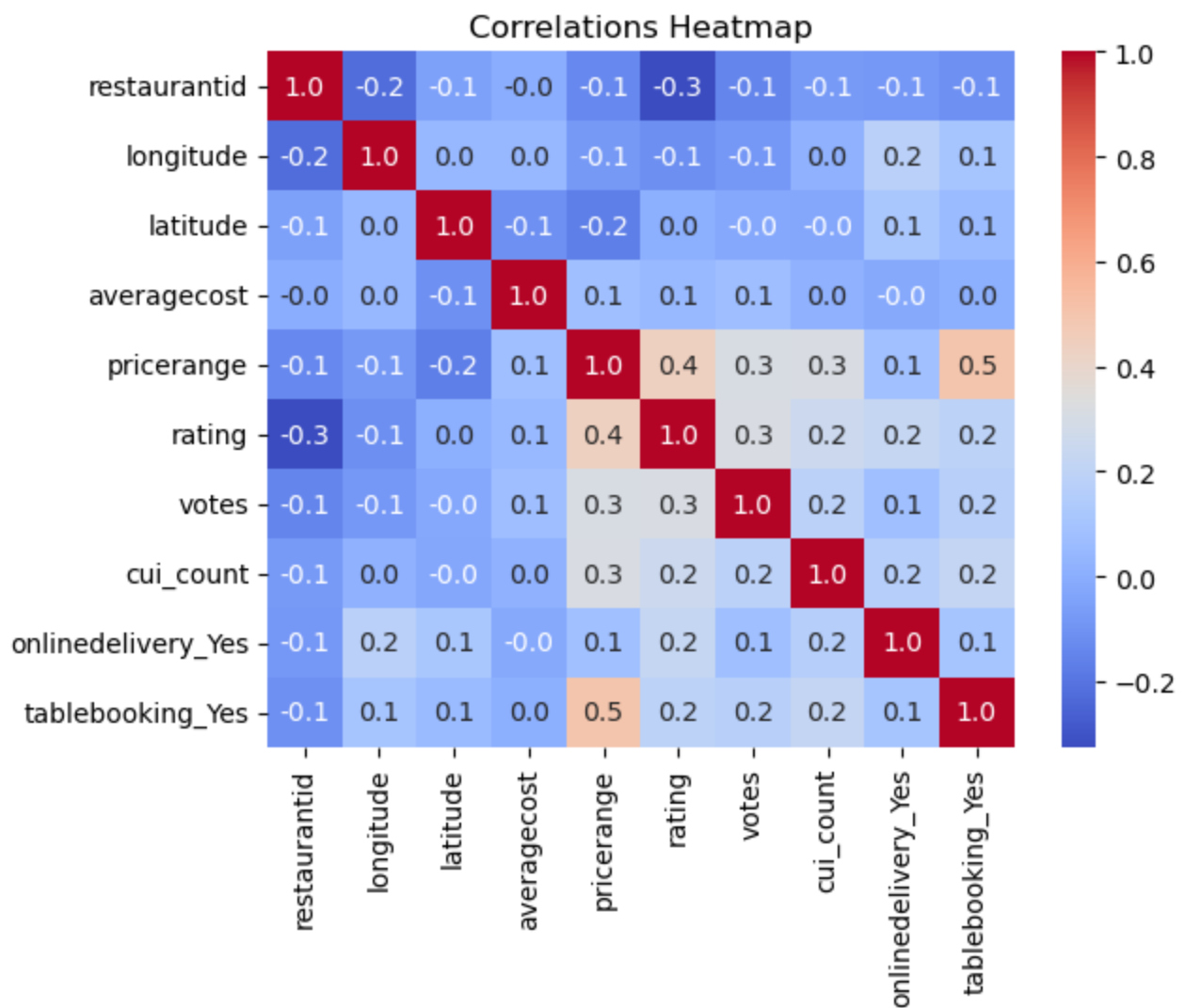
## Correlations of Numerical Attributes

```
In [40]: # enumerate 'onlineodelivery' and 'tablebooking'
dummy_df = df.copy()
dummy_df = pd.get_dummies(dummy_df, columns=['onlinedelivery', 'tablebooking'], drop_first=True)
dummy_df[['onlinedelivery_Yes', 'tablebooking_Yes']].head()
```

```
Out[40]:
```

	onlinedelivery_Yes	tablebooking_Yes
0	0	1
1	0	0
2	0	0
3	0	1
4	0	1

```
In [41]: sns.heatmap(dummy_df.corr(), annot=True, fmt='.1f', cmap='coolwarm')
plt.title('Correlations Heatmap')
plt.show()
```



- 'rating' and 'votes' have a correlation of 0.3, indicating the number of 'votes' affect the 'rating'
- 'pricerange' and 'rating' have a correlation of 0.4, suggesting a higher rating is received for more 'pricerange'
- 'pricerange' and 'tablebooking\_yes' have a correlation of 0.5, implying restaurants with more 'pricerange' has the 'tablebooking' service

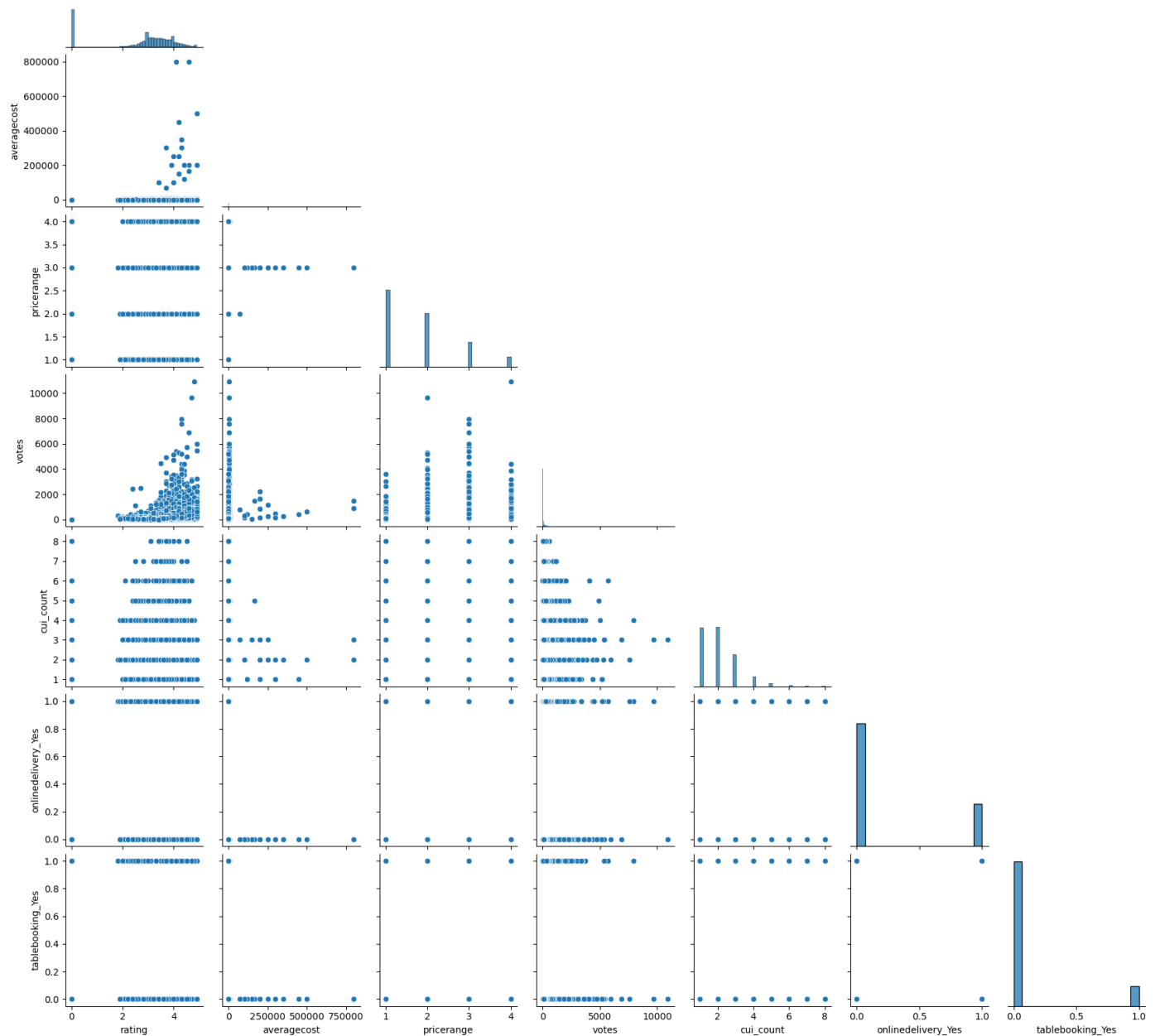
## Pairplot of Ratings vs other Attributes

```
In [42]: to_pairplot = dummy_df[['rating', 'averagecost', 'pricerange', 'votes', 'cui_count', 'co
```

```
In [43]: plt.style.use('fast')
plt.figure(figsize=(12,6))
sns.pairplot(to_pairplot, palette='Spectral', corner=True)
plt.suptitle('Ratings vs Numerical Attributes', fontsize='30', fontweight='heavy')
plt.show()
```

<Figure size 1200x600 with 0 Axes>

# Ratings vs Numerical Attributes



## Conclusion

- From our EDA, the data presents that 'rating' is mostly correlated to 'pricerange', at 0.4. The higher the 'pricerange', the higher the 'rating' score.
- 'rating' and 'votes' are also correlated by 0.3 which suggests more 'votes' contributes to higher 'rating' scores.
- There appears to be more 'rating' scores for restaurants that have between 1-4 'cuisines' than others
- The 'rating' is not clearly affected by restaurants providing 'onlinedelivery' or 'tablebooking' services.

```
In [44]: df.to_excel('checkedcapstone1restaurants.xlsx')
```

## Tableau Dashboard

Tableau Dashboard:

[https://public.tableau.com/views/RestaurantRatings\\_16749023337480/RestaurantDash?:language=en-US&publish=yes&:display\\_count=n&:origin=viz\\_share\\_link](https://public.tableau.com/views/RestaurantRatings_16749023337480/RestaurantDash?:language=en-US&publish=yes&:display_count=n&:origin=viz_share_link)