About Dataset

The dataset under scrutiny delves into the vibrant culinary landscape of Bengaluru, a city teeming with diverse restaurants offering an array of cuisines from across the globe. With an impressive count of approximately 12,000 restaurants, Bengaluru has established itself as a haven for food enthusiasts. Despite the burgeoning numbers, the industry remains dynamic, with new establishments sprouting daily. However, these newcomers face challenges such as high real estate costs, escalating food expenses, a scarcity of skilled personnel, a fragmented supply chain, and regulatory hurdles.

Context

Bengaluru, renowned for its gastronomic diversity, hosts restaurants catering to varied preferences, including delivery, dine-out, pubs, bars, drinks, buffets, and desserts. The dataset, sourced from Zomato, strives to unravel the demographic intricacies of different locations within Bengaluru. With the goal of assisting new restaurants in crafting their identity, menu, cuisine, and pricing strategies based on specific locales, the dataset also endeavors to identify similarities between neighborhoods concerning food preferences. In addition to restaurant details, the dataset incorporates customer reviews, a valuable resource for assessing overall restaurant ratings.

Content

The fundamental objective of scrutinizing the Zomato dataset is to comprehend the factors influencing the establishment of diverse restaurants across different locales in Bengaluru. The city boasts an extensive culinary landscape, featuring over 12,000 restaurants serving global cuisines. Despite the growing demand, emerging restaurants encounter challenges in competing with established counterparts, often offering similar menus. Given Bengaluru's status as the IT capital of India, where many residents rely on restaurant food due to time constraints, understanding location-based demography becomes crucial.

Analyzing factors such as the restaurant's location, approximate price range, theme, and the prevalent cuisine in a neighborhood aids in gauging local preferences. The dataset facilitates insights into whether a locality leans towards vegetarian or non-vegetarian fare, potentially influenced by the predominant demographic, like Jain, Marwari, or Gujarati communities. Key considerations include understanding the needs of residents seeking diverse cuisines, identifying neighborhoods renowned for specific foods, and assessing the popularity of theme-based restaurants.

The data, accurate up to March 15, 2019, was extracted from Zomato in two phases. In the initial phase, essential information such as restaurant names, addresses, and URLs were collected. This streamlined the subsequent data extraction process, minimizing computational load. The second phase involved scraping detailed data for each restaurant and category, encompassing variables like online ordering, table booking, ratings, votes, contact information, location, restaurant type, preferred dishes, cuisines, approximate costs, reviews, and menu items.

Phase I

During Phase I, URLs, names, and addresses of restaurants visible on the front page were extracted. These URLs were recorded in a CSV file, serving as a reference for subsequent individual data extraction. This two-step approach enhanced efficiency and reduced computational strain.

Phase II

In Phase II, comprehensive data for each restaurant and category were scraped. Fifteen variables, ranging from online order options to menu details, were collected for every neighborhood and category. Refer to Section 5 for an in-depth exploration of these variables. The dataset thus compiled provides a holistic view of Bengaluru's gastronomic landscape, offering insights beneficial for both new and established restaurants.

In essence, the dataset encapsulates Bengaluru's culinary narrative, making it a valuable resource for stakeholders seeking to navigate and thrive in this dynamic and diverse food ecosystem.

Acknowledgements

What is the purpose behind conducting this analysis?

1. Strategic Positioning: Gain insights for strategically positioning your restaurant in Bengaluru's diverse market.
2. Competitive Edge: Understand the competitive landscape, enabling you to differentiate and stand out.
3. Localized Decision-making:Tailor decisions based on specific neighborhood dynamics, preferences, and demands.
4. Operational Efficiency: Optimize operations by addressing challenges related to staffing, supply chain, and licensing.
5. Market Opportunities: Identify untapped markets, diversification opportunities, and emerging culinary trends.
6. Customer-Centric Approach: Enhance customer satisfaction by aligning menus and themes with local preferences.
7. Regulatory Compliance: Navigate regulatory hurdles effectively by understanding licensing requirements and challenges.
8. Continuous Improvement: Utilize customer reviews and ratings for continuous improvement and service enhancement.
9. Strategic Marketing: Develop targeted marketing strategies aligned with unique neighborhood characteristics.
10. Cost Optimization: Optimize pricing and costs based on local economic factors revealed through data analysis.
11. Cultural Sensitivity: Customize offerings to align with cultural influences prevalent in specific localities.
12. Effective Advertising: Maximize visibility through location-based advertising, reaching the target audience effectively.
13. Business Expansion: Identify opportunities for business expansion and diversification in the dynamic Bengaluru market.
14. Customer Retention: Craft menus and experiences that resonate with local tastes, enhancing customer retention.
15. Risk Mitigation: Mitigate risks associated with high real estate costs, rising food costs, and workforce shortages.
16. Theme Selection: Data-driven insights guide new restaurants in selecting themes aligned with local preferences.
17. Supply Chain Optimization: Streamline operations by identifying and addressing fragmented supply chain challenges.
18. Popular Dishes: Identify and offer the most liked dishes in a neighborhood.
19. Optimal Restaurant Type: Determine suitable restaurant types based on neighborhood demand and preferences.

Feature description

1. **url** contains the url of the restaurant in the zomato website
2. **address** contains the address of the restaurant in Bengaluru

3. **name** contains the name of the restaurant
4. **online_order** whether online ordering is available in the restaurant or not
5. **book_table** table book option available or not
6. **rate** contains the overall rating of the restaurant out of 5
7. **votes** contains total number of rating for the restaurant as of the above mentioned date
8. **phone** contains the phone number of the restaurant
9. **location** contains the neighborhood in which the restaurant is located
10. **rest_type** restaurant type
11. **dish_liked** dishes people liked in the restaurant
12. **cuisines** food styles, separated by comma
13. **approx_cost**(for two people) contains the approximate cost of meal for two people
14. **reviews_list** list of tuples containing reviews for the restaurant, each tuple
15. **menu_item** contains list of menus available in the restaurant
16. **listed_in(type)** type of meal
17. **listed_in(city)** contains the neighborhood in which the restaurant is listed

Objectives:

1. Conduct thorough Exploratory Data Analysis (EDA) on the Zomato Dataset.
2. Develop a suitable Machine Learning Model to assist Zomato Restaurants in predicting their Ratings using specific features.

Import Libraries

In [180…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import ast
import re
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score
from sklearn.preprocessing import StandardScaler, MultiLabelBinarizer
from sklearn.preprocessing import RobustScaler, LabelEncoder
from sklearn.model_selection import train_test_split,GridSearchCV, cross_validate
from sklearn.ensemble import ExtraTreesClassifier,GradientBoostingClassifier,RandomFores
from sklearn.ensemble import AdaBoostClassifier, VotingClassifier
from sklearn.svm import SVC
from lightgbm import LGBMClassifier
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from sklearn.tree import DecisionTreeClassifier
import warnings
warnings.filterwarnings("ignore")
```

Import Dataset

In [181…
```
df = pd.read_csv(r'C:\Users\user\Desktop\jupyter_practice\zomato.csv/zomato.csv')
```

Cleaning Dataset

In [182…
```
print('Shape of the dataset are: ',df.shape)
print(df.dtypes)
```

```
Shape of the dataset are:  (51717, 17)
url                            object
address                        object
name                           object
online_order                   object
book_table                     object
rate                           object
votes                           int64
phone                          object
location                       object
rest_type                      object
dish_liked                     object
cuisines                       object
approx_cost(for two people)    object
reviews_list                   object
menu_item                      object
listed_in(type)                object
listed_in(city)                object
dtype: object
```

Displaying the count of null values in each column of the DataFrame

In [183…
```
print(df.isnull().sum())
```

```
url                               0
address                           0
name                              0
online_order                      0
book_table                        0
rate                           7775
votes                             0
phone                          1208
location                         21
rest_type                       227
dish_liked                    28078
cuisines                         45
approx_cost(for two people)     346
reviews_list                      0
menu_item                         0
listed_in(type)                   0
listed_in(city)                   0
dtype: int64
```

Displaying the percentage of null values in each column of the DataFrame

In [184…
```
print(((df.isna().sum()/df.shape[0])*100).round(2))
```

```
url                             0.00
address                         0.00
name                            0.00
online_order                    0.00
book_table                      0.00
rate                           15.03
votes                           0.00
phone                           2.34
location                        0.04
rest_type                       0.44
dish_liked                     54.29
cuisines                        0.09
approx_cost(for two people)     0.67
reviews_list                    0.00
menu_item                       0.00
listed_in(type)                 0.00
listed_in(city)                 0.00
dtype: float64
```

Droping unnecessary columns and renames specific columns

In [185…
```
df = df.drop(columns= ['url', 'menu_item', 'phone','dish_liked', 'reviews_list'], axis=1

df.rename(columns={'approx_cost(for two people)':'cost',
                   'listed_in(type)':'type',
                   'listed_in(city)':'city',
                   'listed_in(type)':'meal_type'}, inplace=True)
```

In [186…
```
print(f'Number of Duplicate rows: {df.duplicated().sum()}')
df.drop_duplicates(inplace=True)
print(f'Number of Duplicate rows after dropping: {df.duplicated().sum()}')
```

```
Number of Duplicate rows: 85
Number of Duplicate rows after dropping: 0
```

Checking "Nan" location rows

In [187…
```
location_null = 'location'
null_rows = df[df[location_null].isnull()][['address',
                                            'location',
                                            'city',
                                            'rate',
                                            'cost',
                                            'rest_type']]
print("Rows where '{}' is null:".format(location_null))
print(null_rows.to_string(index=False))
```

```
Rows where 'location' is null:
                                                                                address locat
ion                  city rate cost rest_type
                      6, Abbiah Layout KC Halli Main Road, Bommanahalli, Bangalore
NaN     Bannerghatta Road  NaN  NaN       NaN
                      6, Abbiah Layout KC Halli Main Road, Bommanahalli, Bangalore
NaN                   BTM  NaN  NaN       NaN
Chandapura- Anakal Main Road, Oppsite Vemana Hosipatal, Electronic City, Bangalore
NaN       Electronic City  NaN  NaN       NaN
                                                    Kudlu Gate, Bommanahalli, Bangalore
NaN                   HSR  NaN  NaN       NaN
            IB Road, Lorry Stand, Kushaal Nagar, Ward 10, Kalyan Nagar, Bangalore
NaN          Kammanahalli  NaN  NaN       NaN
                                                      Koramangala 8th Block, Bangalore
NaN Koramangala 4th Block  NaN  NaN       NaN
              6, Kathalipalya Village, 6th Cross, Koramangala 6th Block, Bangalore
NaN Koramangala 4th Block  NaN  NaN       NaN
              6, Kathalipalya Village, 6th Cross, Koramangala 6th Block, Bangalore
NaN Koramangala 4th Block  NaN  NaN       NaN
                                                      Koramangala 8th Block, Bangalore
NaN Koramangala 5th Block  NaN  NaN       NaN
              6, Kathalipalya Village, 6th Cross, Koramangala 6th Block, Bangalore
NaN Koramangala 5th Block  NaN  NaN       NaN
              6, Kathalipalya Village, 6th Cross, Koramangala 6th Block, Bangalore
NaN Koramangala 5th Block  NaN  NaN       NaN
                                                      Koramangala 8th Block, Bangalore
NaN Koramangala 6th Block  NaN  NaN       NaN
                                   80 Feet Road, Koramangala 4th Block, Bangalore
NaN Koramangala 6th Block  NaN  NaN       NaN
                                                      Koramangala 8th Block, Bangalore
NaN Koramangala 7th Block  NaN  NaN       NaN
       399/34, 19th Main, Near Navarang Theatre, 2nd Block, Rajajinagar, Bangalore
NaN          Malleshwaram  NaN  NaN       NaN
                  1630, 6th Main Road, AECS Layout E Block, Marathahalli, Bangalore
NaN          Marathahalli  NaN  NaN       NaN
                  1630, 6th Main Road, AECS Layout E Block, Marathahalli, Bangalore
```

```
NaN          Marathahalli  NaN  NaN       NaN
                                    9, Magadi Main Road, Cholourpalya, Vijay Nagar
NaN           Rajajinagar  NaN  NaN       NaN
              West of Chord Road, Govindaraja Nagar, Below Hosahalli Metro Station
NaN           Rajajinagar  NaN  NaN       NaN
       399/34, 19th Main, Near Navarang Theatre, 2nd Block, Rajajinagar, Bangalore
NaN           Rajajinagar  NaN  NaN       NaN
                                    9, Magadi Main Road, Cholourpalya, Vijay Nagar
NaN           Rajajinagar  NaN  NaN       NaN
```

When faced with null values in the "location" column, I opted to replace them with the corresponding city names. However, I encountered instances where all columns for those rows were also filled with 'NaN'. Consequently, I chose to address this by eliminating those specific rows from the dataset.

Droping city and address columns

In [188…
```
df = df.drop(columns= ['city','address'], axis=1)
```

In [189…
```
print('unique rate:', df['rate'].unique())
print('-'*50)
print('unique cost:', df['cost'].unique())
print('-'*50)
```

```
unique rate: ['4.1/5' '3.8/5' '3.7/5' '3.6/5' '4.6/5' '4.0/5' '4.2/5' '3.9/5' '3.1/5'
 '3.0/5' '3.2/5' '3.3/5' '2.8/5' '4.4/5' '4.3/5' 'NEW' '2.9/5' '3.5/5' nan
 '2.6/5' '3.8 /5' '3.4/5' '4.5/5' '2.5/5' '2.7/5' '4.7/5' '2.4/5' '2.2/5'
 '2.3/5' '3.4 /5' '-' '3.6 /5' '4.8/5' '3.9 /5' '4.2 /5' '4.0 /5' '4.1 /5'
 '3.7 /5' '3.1 /5' '2.9 /5' '3.3 /5' '2.8 /5' '3.5 /5' '2.7 /5' '2.5 /5'
 '3.2 /5' '2.6 /5' '4.5 /5' '4.3 /5' '4.4 /5' '4.9/5' '2.1/5' '2.0/5'
 '1.8/5' '4.6 /5' '4.9 /5' '3.0 /5' '4.8 /5' '2.3 /5' '4.7 /5' '2.4 /5'
 '2.1 /5' '2.2 /5' '2.0 /5' '1.8 /5']
--------------------------------------------------
unique cost: ['800' '300' '600' '700' '550' '500' '450' '650' '400' '900' '200' '750'
 '150' '850' '100' '1,200' '350' '250' '950' '1,000' '1,500' '1,300' '199'
 '80' '1,100' '160' '1,600' '230' '130' '50' '190' '1,700' nan '1,400'
 '180' '1,350' '2,200' '2,000' '1,800' '1,900' '330' '2,500' '2,100'
 '3,000' '2,800' '3,400' '40' '1,250' '3,500' '4,000' '2,400' '2,600'
 '120' '1,450' '469' '70' '3,200' '60' '560' '240' '360' '6,000' '1,050'
 '2,300' '4,100' '5,000' '3,700' '1,650' '2,700' '4,500' '140']
--------------------------------------------------
```

The dataset reveals that certain data points are represented as strings, with some values containing commas (e.g., 5,000, 6,000). To facilitate numerical analysis, these commas need removal. Additionally, the 'Rating' column is in string format and must be converted to numeric format, necessitating the removal of '/5' from the values. Furthermore, the presence of 'NEW' in the 'Rating' column holds no meaningful information and should be excluded from the dataset.

Removeing 'NEW', 'nan', '-' and '/5' value from 'rate' column

In [190…
```
def filter_rate(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(filter_rate)
df['rate'].sample(5,random_state=1)
```

Out[190]:
```
4858     3.4
16948    4.1
```

```
21299    NaN
34300    2.8
722      NaN
Name: rate, dtype: float64
```

In [191…
```
df['rate'].mean()
```

Out[191]:
```
3.7000817268400525
```

Removing Null Rows of Rate Column

I attempted to address missing values in the 'rate' column by initially using the values from the 'review list' column. However, these values varried widely (0.2 to 1.9) to rate column, introducing inconsistency. Subsequently, I opted to fill the null values with the mean value of the 'rate' column. Unfortunately, this approach significantly skewed the comparisons involving the 'rate' column. Consequently, to maintain the integrity of independent variables and avoid distortions in comparisons, I chose to drop the rows with null values in the 'rate' column to ensure that the analysis and evaluations involving the 'rate' column remain unbiased and reflective of the available data.

In [192…
```
#df['rate'].fillna((round(df['rate'].mean(), 1)), inplace=True)
#df['rate'] = df['rate'].round(1)
#df['rate'].isnull().sum()
```

In [193…
```
print(df.isnull().sum())
df.dropna(inplace=True)
#print(df.isnull().sum())
```

```
name                0
online_order        0
book_table          0
rate            10030
votes               0
location           21
rest_type         227
cuisines           45
cost              344
meal_type           0
dtype: int64
```

Converting 'cost' column to integer after removing commas from the values.

In [194…
```
df['cost'] = df['cost'].apply(lambda x: x.replace(',',''))
df['cost']=df['cost'].astype(int)
```

Lowercase 'rest_type' and 'meal_type'; count subcategories for 'rest_type' with commas.

In [195…
```
df['rest_type'] = df['rest_type'].apply(lambda x: x.lower())
df['rest_type_count'] = df['rest_type'].apply(lambda x: len(x.split(', ')))
df['meal_type'] = df['meal_type'].apply(lambda x: x.lower())
```

Our data is ready for visualization.
There are 3 numerical features: rate, votes, cost
There are 2 binary features: online_order, book_table
The remaining columns are categorical

Exploratory Data Analysis

Overview of DataFrame Statistics

In [196…
```
df.describe(include='all').T
```

Out[196]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
name	41202	6602	Cafe Coffee Day	86	NaN	NaN	NaN	NaN	NaN	NaN	NaN
online_order	41202	2	Yes	27055	NaN	NaN	NaN	NaN	NaN	NaN	NaN
book_table	41202	2	No	34923	NaN	NaN	NaN	NaN	NaN	NaN	NaN
rate	41202.0	NaN	NaN	NaN	3.70167	0.439894	1.8	3.4	3.7	4.0	4.9
votes	41202.0	NaN	NaN	NaN	351.967599	883.351883	0.0	21.0	73.0	277.0	16832.0
location	41202	92	BTM	3873	NaN	NaN	NaN	NaN	NaN	NaN	NaN
rest_type	41202	87	quick bites	13866	NaN	NaN	NaN	NaN	NaN	NaN	NaN
cuisines	41202	2367	North Indian	2107	NaN	NaN	NaN	NaN	NaN	NaN	NaN
cost	41202.0	NaN	NaN	NaN	603.498762	464.602925	40.0	300.0	500.0	750.0	6000.0
meal_type	41202	7	delivery	20410	NaN	NaN	NaN	NaN	NaN	NaN	NaN
rest_type_count	41202.0	NaN	NaN	NaN	1.154823	0.36174	1.0	1.0	1.0	1.0	2.0

1. **Name:** The dataset includes information on 6,602 unique restaurant names. The most frequent name is "Cafe Coffee Day," appearing 86 times. This suggests a diverse culinary landscape with numerous establishments.
2. **Online Order:** Among the 41,202 records, 27,055 indicate that online orders are available, reflecting the growing trend of digital convenience in the food industry. The majority of restaurants offer this service.
3. **Book Table:** For the "Book Table" feature, 34,923 entries indicate that the service is not available. This is a common trend, as many casual dining places may not require table reservations.
4. **Rate:** The average rating across all restaurants is approximately 3.70, with a minimum of 1.8 and a maximum of 4.9. This indicates a generally favorable rating environment, with a tendency towards higher scores.
5. **Votes:** The average number of votes per restaurant is 352, ranging from 0 to 16,832. This wide distribution suggests varying levels of popularity and engagement among establishments.
6. **Location:** The dataset covers 92 unique locations, with "BTM" being the most prevalent, featuring 3,873 restaurants. Location diversity indicates a broad representation of neighborhoods in Bangalore.
7. **Rest Type:** There are 87 unique restaurant types, with "Quick Bites" being the most common (13,866 occurrences). This highlights the prevalence of casual, quick-service dining options.
8. **Cuisines:** The dataset encompasses 2,367 unique cuisines, with "North Indian" appearing 2,107 times. This diverse culinary offering showcases the richness of Bangalore's food scene.

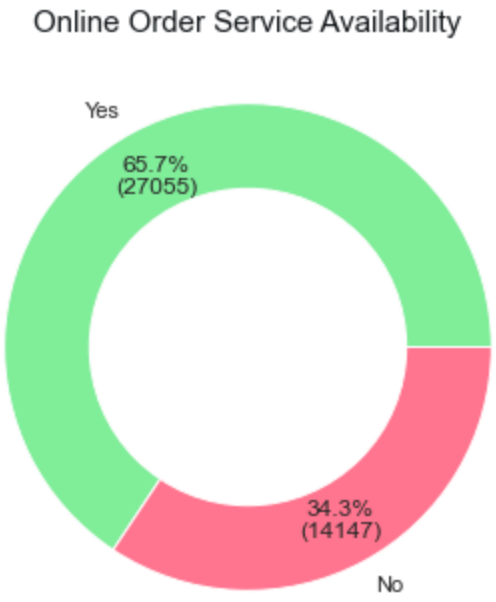
9. **Cost:** The average cost of dining is approximately 603 Rupees, with a minimum of 40 Rupees and a maximum of 6,000 Rupees. This wide cost range caters to various budget preferences.

10. **Meal Type:** The "Delivery" meal type is the most prevalent among the seven categories, with 20,410 instances. This aligns with the increasing popularity of food delivery services.

11. **Rest Type Count:** The average count of restaurant types associated with each record is 1.15, suggesting that most establishments primarily fall into one category. This indicates a clear classification of restaurant types in the dataset.

How many Restuarants have online order service?

In [197…
```
plt.rcParams['figure.figsize'] = 9, 6
plt.subplot(1, 2, 1)
online_order_counts = df.online_order.value_counts()
pie = df.online_order.value_counts().plot(kind='pie', colors=['#80ed99', '#ff758f'],
      autopct=lambda p: '{:.1f}%\n({:.0f})'.format(p, p * sum(online_order_counts) / 100
                                                   pctdistance=0.80)
w_circle = plt.Circle((0, 0), 0.65, color='white')
p = plt.gcf()
p.gca().add_artist(w_circle)
plt.title('Online Order Service Availability' ,fontsize=15, color='#161a1d')
plt.ylabel('')
plt.tight_layout()
plt.show()
```

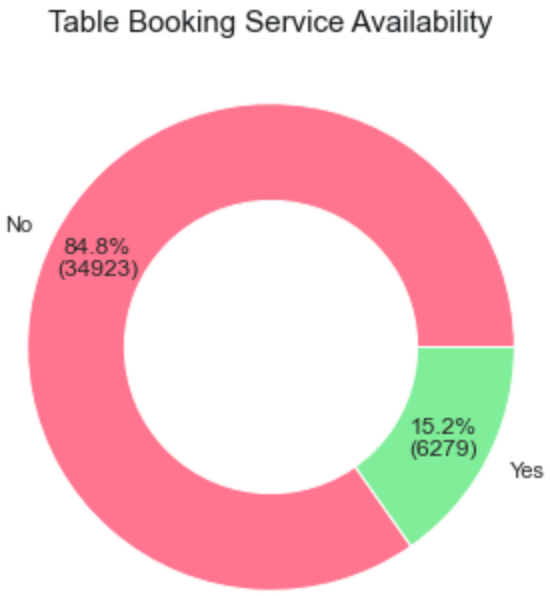


Most Restaurants (66.7%) offer online order service

How many Restuarants have book table service?

In [198…
```
plt.rcParams['figure.figsize'] = 9, 6
plt.subplot(1, 2, 1)
book_table_counts = df.book_table.value_counts()
pie = df.book_table.value_counts().plot(kind='pie', colors=['#ff758f', '#80ed99'],
      autopct=lambda p: '{:.1f}%\n({:.0f})'.format(p, p * sum(book_table_counts) / 100),
                                                   pctdistance=0.80)
w_circle = plt.Circle((0, 0), 0.6, color='white')
p = plt.gcf()
p.gca().add_artist(w_circle)
```

```
plt.title('Table Booking Service Availability' ,fontsize=15, color='#161a1d')
plt.ylabel('')
plt.tight_layout()
plt.show()
```



Most Restaurants (84.8%) don't offer table booking service

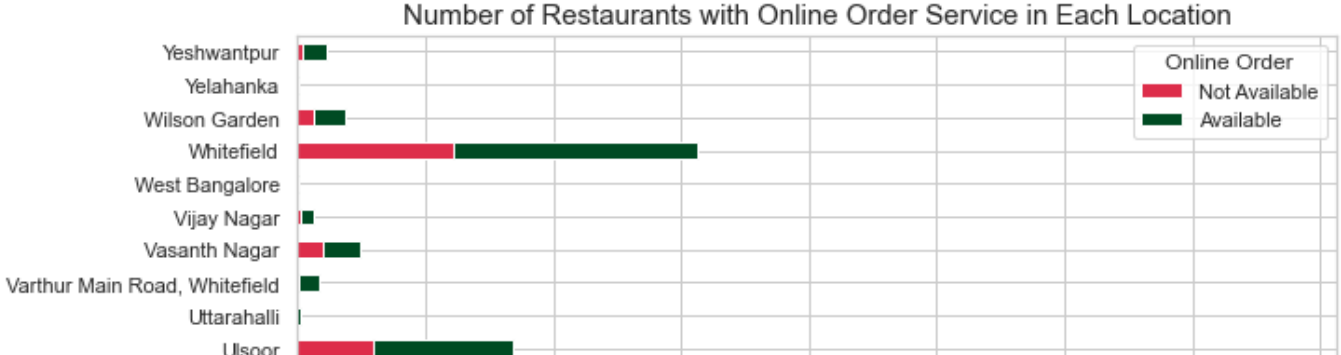
How many Restaurants have both online order and book table service?

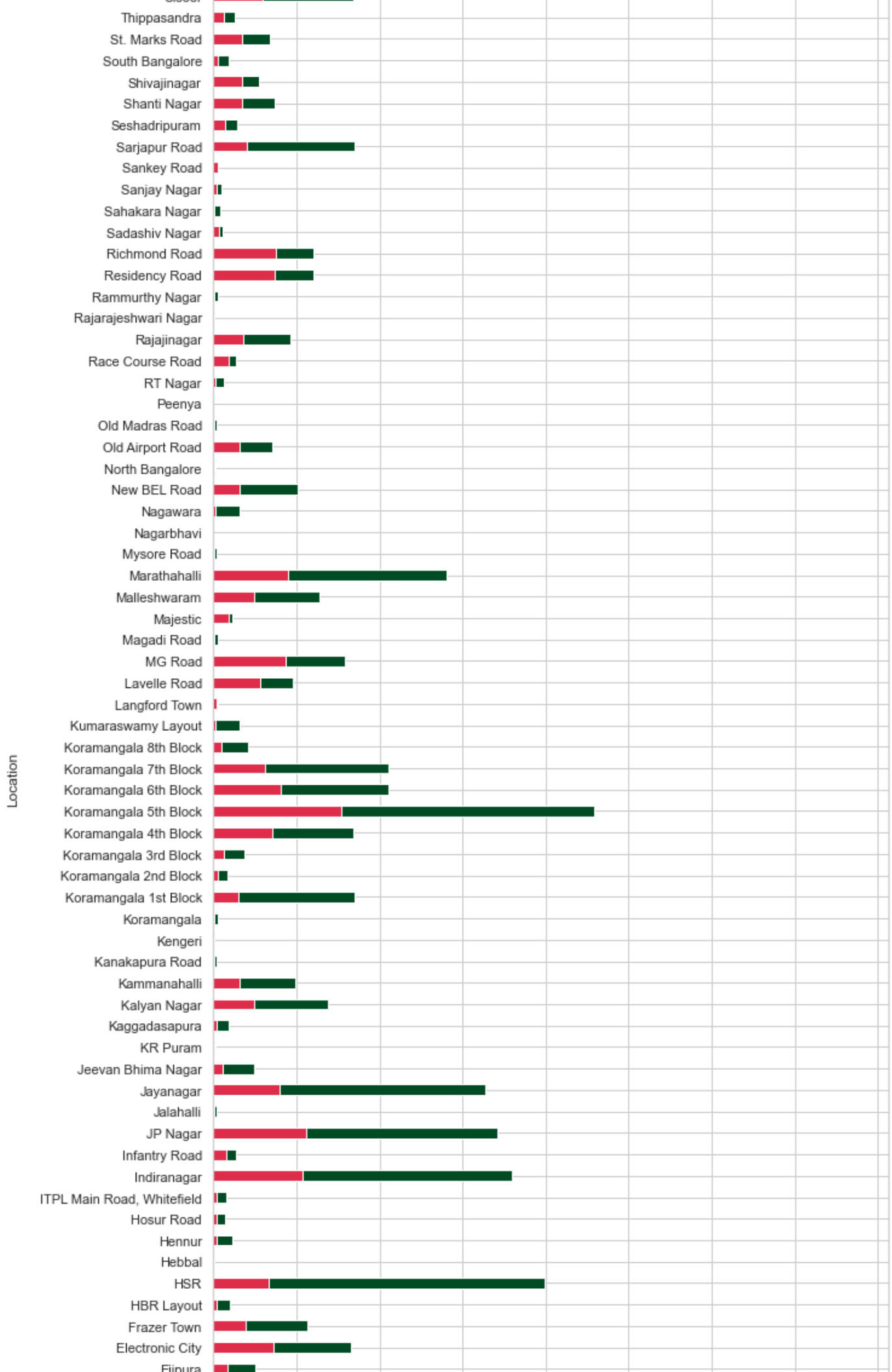
In [199…
```
filtered_restaurants = df[(df['online_order'] == 'Yes') & (df['book_table'] == 'Yes')]
unique_names = filtered_restaurants['name'].nunique()
total_names = df.name.nunique()
print(f'{unique_names} restaurants have both online order and book table services out of
```

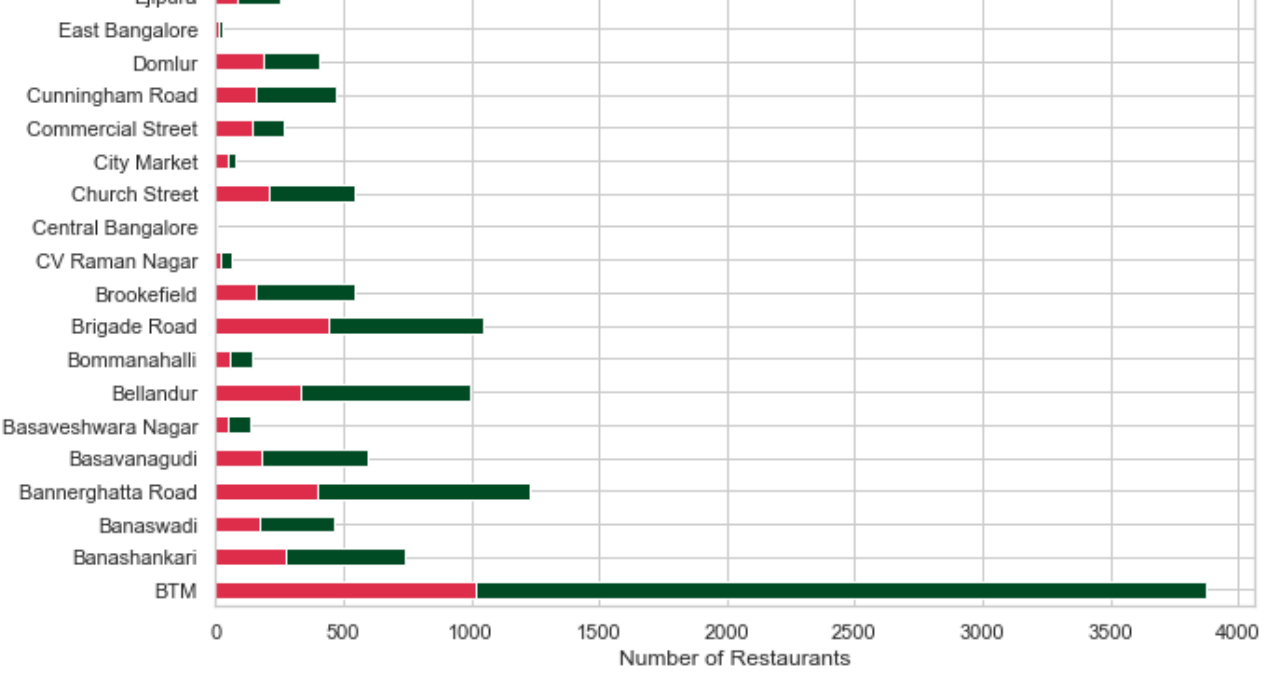
```
420 restaurants have both online order and book table services out of 6602 total restaur
ants.
```

Online order based on location

In [200…
```
order_loc = df.groupby(['location', 'online_order'])['name'].count().reset_index()
order_loc = pd.pivot_table(order_loc, values='name', index='location',
             columns='online_order', fill_value=0, aggfunc='sum')
colors = {'Yes': '#004b23', 'No': '#dd2d4a'}
order_loc.plot(kind='barh', stacked=True, figsize=(10, 30),
                color=order_loc.columns.map(colors))
plt.title('Number of Restaurants with Online Order Service in Each Location', fontsize=1
plt.xlabel('Number of Restaurants', fontsize=12)
plt.ylabel('Location', fontsize=12)
plt.legend(title='Online Order', loc='upper right',
 labels=['Not Available', 'Available'])
plt.show()
```




Location
Thippasandra
St. Marks Road
South Bangalore
Shivajinagar
Shanti Nagar
Seshadripuram
Sarjapur Road
Sankey Road
Sanjay Nagar
Sahakara Nagar
Sadashiv Nagar
Richmond Road
Residency Road
Rammurthy Nagar
Rajarajeshwari Nagar
Rajajinagar
Race Course Road
RT Nagar
Peenya
Old Madras Road
Old Airport Road
North Bangalore
New BEL Road
Nagawara
Nagarbhavi
Mysore Road
Marathahalli
Malleshwaram
Majestic
Magadi Road
MG Road
Lavelle Road
Langford Town
Kumaraswamy Layout
Koramangala 8th Block
Koramangala 7th Block
Koramangala 6th Block
Koramangala 5th Block
Koramangala 4th Block
Koramangala 3rd Block
Koramangala 2nd Block
Koramangala 1st Block
Koramangala
Kengeri
Kanakapura Road
Kammanahalli
Kalyan Nagar
Kaggadasapura
KR Puram
Jeevan Bhima Nagar
Jayanagar
Jalahalli
JP Nagar
Infantry Road
Indiranagar
ITPL Main Road, Whitefield
Hosur Road
Hennur
Hebbal
HSR
HBR Layout
Frazer Town
Electronic City



Areas like BTM and HSR exhibit a strong preference for online orders, with 2857 and 1658 restaurants respectively.

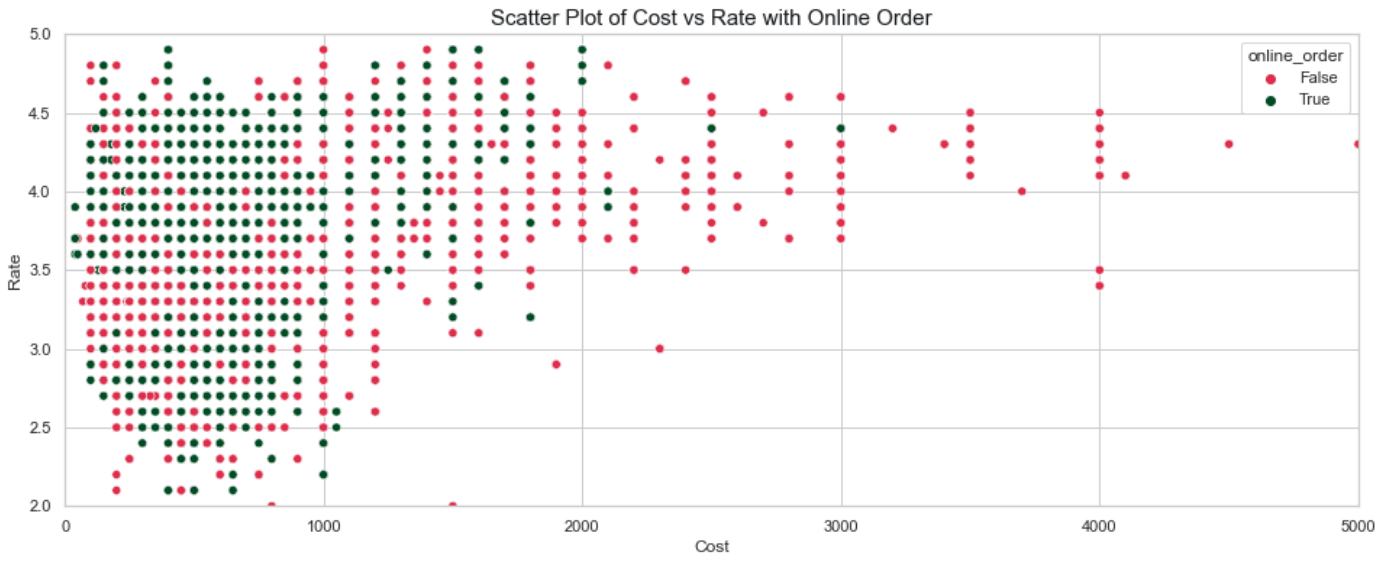
Relationship between cost and rating, considering the impact of online delivery?
Online order impact on restaurant ratings distribution?

In [201…

```
df.online_order.replace(('Yes','No'),(True,False),inplace =True)
df.book_table.replace(('Yes','No'),(True,False),inplace =True)
```

In [202…

```
sns.set(style='whitegrid')
plt.figure(figsize=(16, 6))
sns.scatterplot(x='cost', y='rate', hue='online_order', data=df,
                palette={True: '#004b23', False: '#dd2d4a'})
plt.xlabel('Cost')
plt.ylabel('Rate')
plt.xlim([0,5000])
plt.ylim([2.0,5.0])
plt.title('Scatter Plot of Cost vs Rate with Online Order',  fontsize=15, color='#161a1d
plt.show()
```



Many individuals prefer spending between 300-1000, with similar offline and online order counts. Those who spend 400-900 online influence both higher (3.7-4.7) and lower (2.5-3.5) ratings. Offline orders below

600 impact lower ratings (3.0-3.6). Spending above 1500 offline yields higher ratings than the average

In [203…
```
from scipy import stats
pear = df.dropna(subset=['cost', 'rate'])
pearson_coef, p_value = stats.pearsonr(pear['cost'], pear['rate'])
print("Pearson Correlation Coefficient: ", pearson_coef, "and a P-value of:", p_value) #
```

```
Pearson Correlation Coefficient:  0.38452931118762057 and a P-value of: 0.0
```

The statistically significant p-value (close to 0.0) indicates a relationship between dining cost and restaurant rating. Rejecting the null hypothesis suggests a moderate positive correlation. Thus, dining cost likely influences the restaurant rating, a finding supported by strong statistical evidence.

In [204…
```
colors = ['#004b23', '#dd2d4a']
sns.set_palette(sns.color_palette(colors))

plt.figure(figsize=(10, 6))
sns.boxplot(x='online_order', y='cost', data=df, hue='online_order')

plt.title('Box Plot of Cost by Online Order')
plt.xlabel('Online Order')
plt.ylabel('Cost')
plt.ylim([0,1100])
plt.show()
```

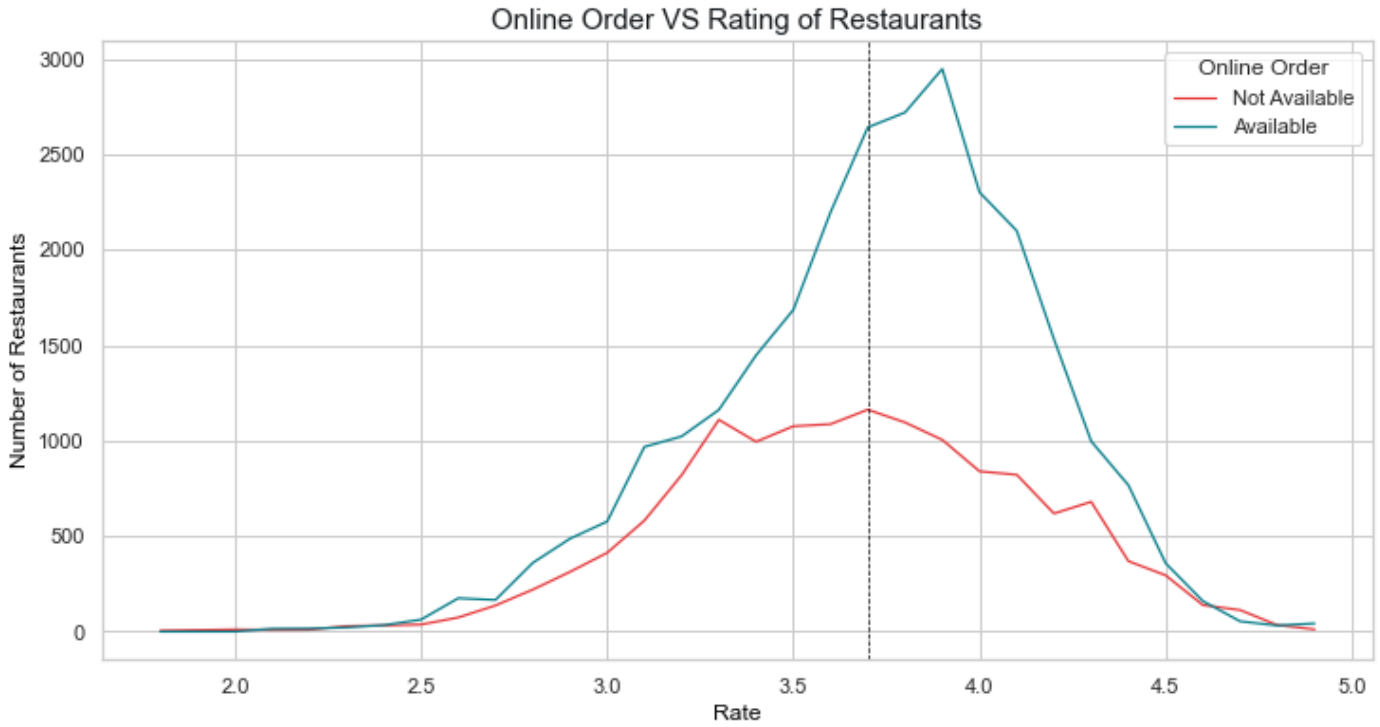
Many individuals prefer spending between 300-1000, with similar offline and online order counts. Those who spend 400-900 online influence both higher (3.7-4.7) and lower (2.5-3.5) ratings. Offline orders below 600 impact lower ratings (3.0-3.6). Spending above 1500 offline yields higher ratings than the average

In the context of online orders, the cost distribution includes a minimum of 40, a first quartile at 300, a median at 450, a third quartile at 700, an upper fence at 1300, and a maximum of 3000. Conversely, for offline orders, the cost distribution involves a minimum of 50, a first quartile at 300, a median at 500, a third quartile at 1000, an upper fence at 2000, and a maximum of 6000."

In [205…
```
plt.rcParams['figure.figsize'] = 12, 6
fig = plt.figure(facecolor='#fefae0')
```

```
cross_tab = pd.crosstab(df.rate, df.online_order)
cross_tab.plot(color=['#e5383b', '#087e8b'],linewidth=1.2)

plt.axvline(df.rate.mean(), color='black', linewidth=0.7, ls='--')

plt.title('Online Order VS Rating of Restaurants', fontsize=15, color='#161a1d')
plt.ylabel('Number of Restaurants', fontsize=12, color='black' )
plt.xlabel('Rate', fontsize=12, color='black')
plt.legend(title='Online Order', loc='upper right',
           labels=['Not Available', 'Available'])
plt.show()
```

```
<Figure size 864x432 with 0 Axes>
```



As we notice from the above chart that having the online order service has no effect on rating the restaurants , except that we can found at the same level of rate there is much number of restaurants that has online order serivce especially above the average rate of 3.7.

Is there any Correlation Between the Availability of Table Booking Services and Average Restaurant Rating?

In [206…

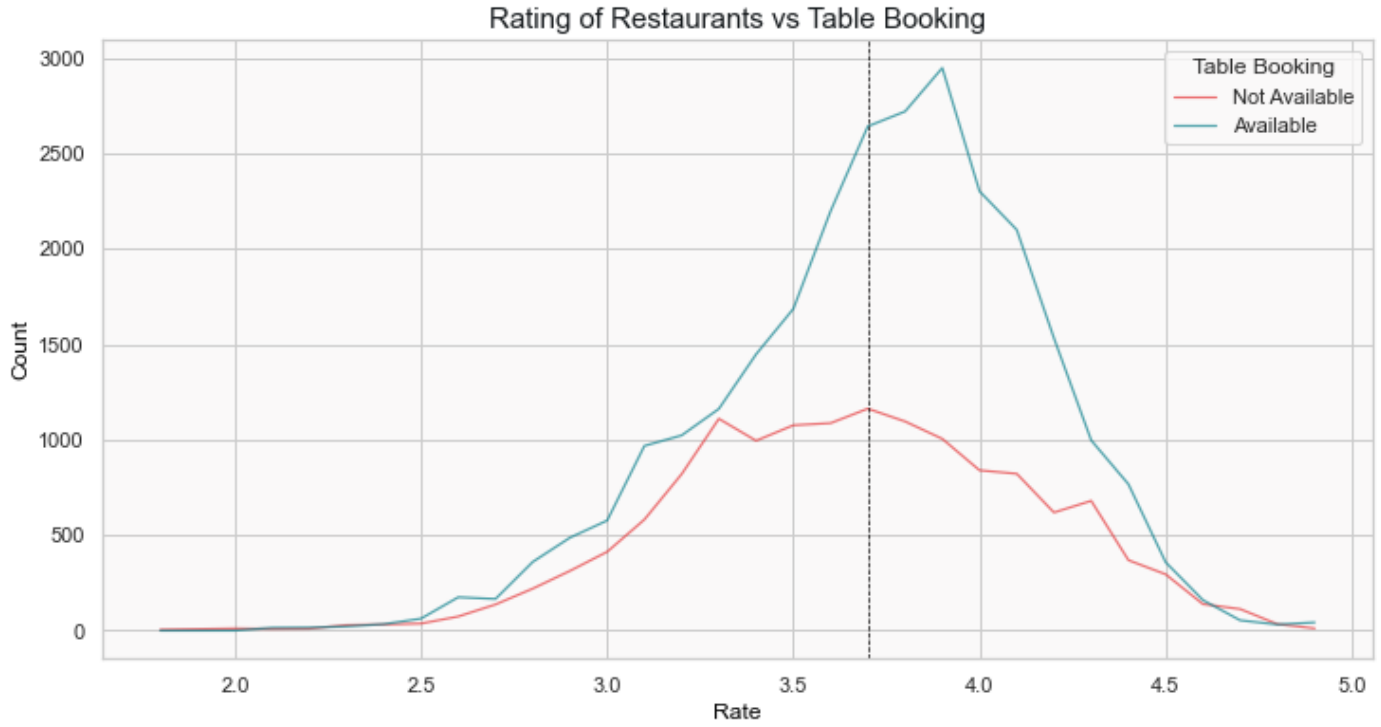
```
df['table_booking_binary'] = df['book_table'].apply(lambda x: 1 if x == True else 0)
average_rating_with_table = df[df['table_booking_binary'] == 1]['rate'].mean()
average_rating_without_table = df[df['table_booking_binary'] == 0]['rate'].mean()

print(f'Average Rating with Table Booking: {average_rating_with_table:.2f}')
print(f'Average Rating without Table Booking: {average_rating_without_table:.2f}')

correlation = df['rate'].corr(df['table_booking_binary'])

print(f'Correlation between Table Booking and Rating: {correlation:.2f}')
df.drop(columns=['table_booking_binary'], inplace=True)
```

```
Average Rating with Table Booking: 4.14
Average Rating without Table Booking: 3.62
Correlation between Table Booking and Rating: 0.43
```

In [207…

```
plt.rcParams['figure.figsize'] = 12, 6
plt.rcParams['axes.facecolor'] = '#faf9f9'

cross_tab = pd.crosstab(df.rate, df.online_order)
```

```
cross_tab.plot( color=['#e5383b', '#087e8b'], linewidth=1.2, alpha=0.7)

plt.axvline(df.rate.mean(), color='black', linewidth=0.7, ls='--')
plt.title('Rating of Restaurants vs Table Booking', fontsize=15, color='#161a1d')
plt.ylabel('Count', fontsize=12, color='black')
plt.xlabel('Rate', fontsize=12, color='black')
plt.legend(title='Table Booking', loc='upper right',
           labels=['Not Available', 'Available'])
plt.show()
```



The correlation coefficient of 0.42 indicates a moderate positive correlation between the availability of table booking services and restaurant ratings. This suggests that, on average, restaurants offering table booking services tend to have slightly higher ratings.

In [208…

```
import statsmodels.formula.api as smf
ols_model = smf.ols(formula='rate ~ C(book_table)', data=df).fit()
print(ols_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   rate   R-squared:                       0.181
Model:                            OLS   Adj. R-squared:                  0.181
Method:                 Least Squares   F-statistic:                     9102.
Date:                Wed, 07 Feb 2024   Prob (F-statistic):               0.00
Time:                        15:59:23   Log-Likelihood:                -20515.
No. Observations:               41202   AIC:                         4.103e+04
Df Residuals:                   41200   BIC:                         4.105e+04
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
                               coef    std err          t      P>|t|      [0.025      0.97
5]
--------------------------------------------------------------------------------
-
Intercept                    3.6223      0.002   1700.335      0.000       3.618       3.62
7
C(book_table)[T.True]        0.5206      0.005     95.405      0.000       0.510       0.53
1
==============================================================================
```

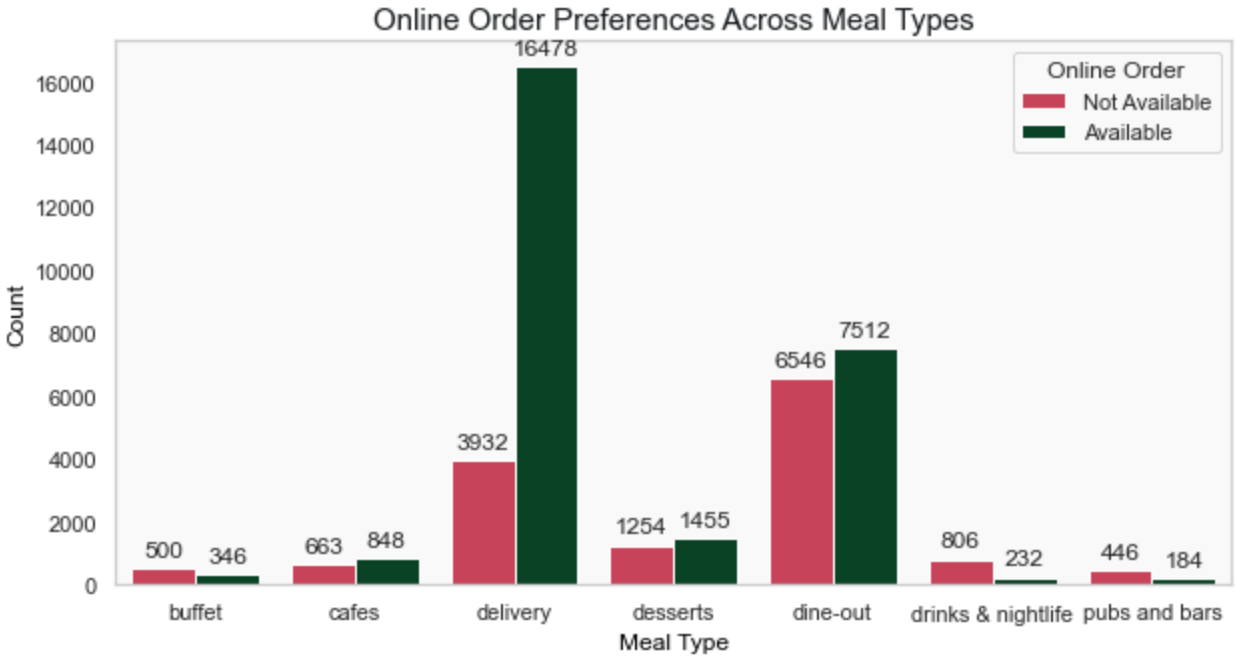
```
Omnibus:                       1219.041   Durbin-Watson:                   1.551
Prob(Omnibus):                    0.000   Jarque-Bera (JB):             1410.249
Skew:                            -0.390   Prob(JB):                    5.87e-307
Kurtosis:                         3.462   Cond. No.                         2.86
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

The Ordinary Least Squares (OLS) regression model indicates a significant relationship between the presence of booking tables and restaurant ratings ($p < 0.001$). With an R-squared value of 0.181, the model explains approximately 18.1% of the variability in ratings. Restaurants with booking tables tend to have higher ratings, as evidenced by the positive coefficient of 0.5206.

How Does the Preference for Online Orders differ across Different Meal Types?

In [209…

```
import seaborn as sns
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1, 1, figsize=(10, 5))
ax = sns.countplot(x='meal_type', hue='online_order', data=df,
                   ax=ax, palette={True: '#004b23', False: '#dd2d4a'})

for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', xytext=(0, 9), textcoords='offset points')

plt.title('Online Order Preferences Across Meal Types', fontsize=15, color='#161a1d')
plt.ylabel('Count', fontsize=12, color='black')
plt.xlabel('Meal Type', fontsize=12, color='black')
plt.legend(title='Online Order', loc='upper right',
           labels=['Not Available', 'Available'])
plt.grid(False)
plt.show()
```



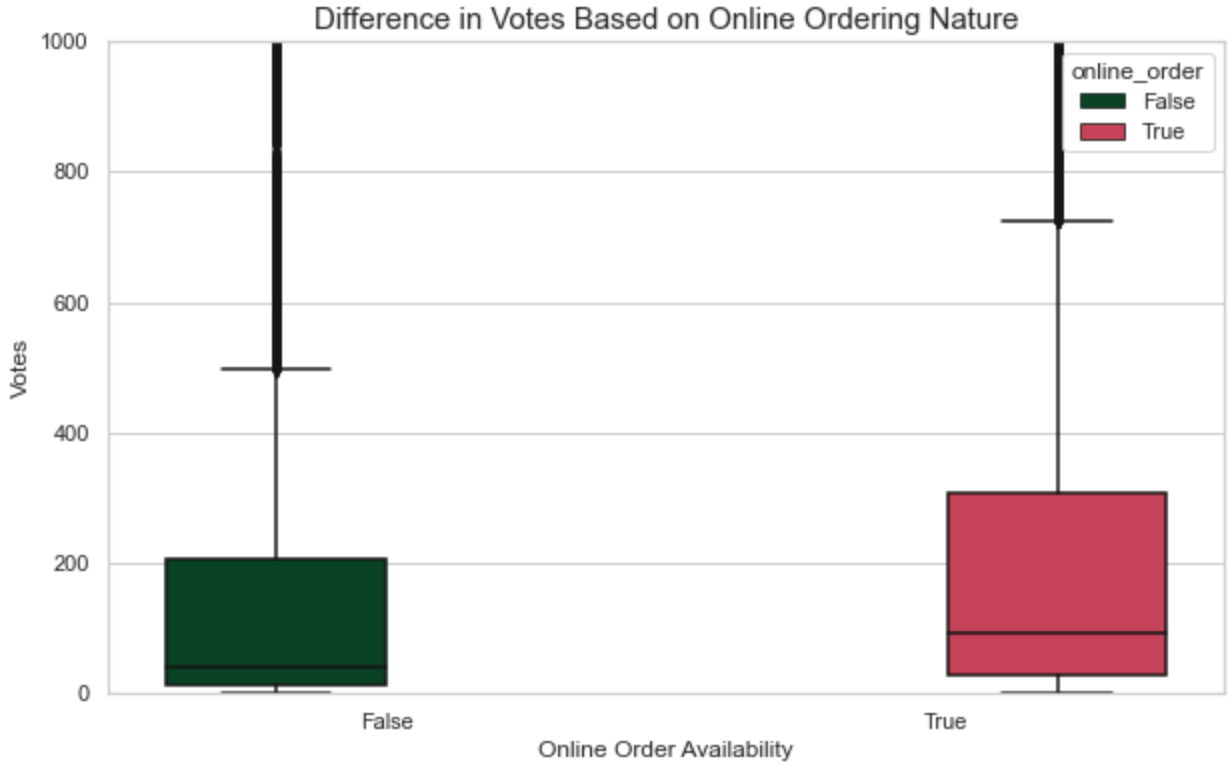
Online orders find favor across various meal types, including delivery, dine-out, desserts, and cafes.

Difference in Votes Based on Online Ordering Nature

In [210…

```
sns.set(style="whitegrid")
```

```
plt.figure(figsize=(10, 6))
sns.boxplot(x='online_order', y='votes', data=df, hue='online_order', palette=['#004b23'
plt.title('Difference in Votes Based on Online Ordering Nature', fontsize=15)
plt.xlabel('Online Order Availability', fontsize=12)
plt.ylabel('Votes', fontsize=12)
plt.ylim([0,1000])
plt.show()
```



The boxplot indicates that restaurants accepting online orders tend to receive higher vote counts on average compared to those without. This suggests a positive correlation between online order availability and customer engagement. Restaurants may benefit from emphasizing and promoting their online ordering services to boost customer interaction and votes.

Cost and Vote Vs Rate

In [211…

```
pd.set_option('display.max_rows', 100)
avg_data = df.groupby('location').agg({'cost': 'mean', 'rate': 'mean', 'votes': 'sum'}).

avg_data.rename(columns={'cost': 'avg_cost', 'rate': 'avg_rate', 'votes': 'total_votes'}

avg_data['avg_cost'] = avg_data['avg_cost'].round(2)
avg_data['avg_rate'] = avg_data['avg_rate'].round(2)

avg_data = avg_data.sort_values(by=['avg_rate', 'total_votes'], ascending=False)
print(avg_data)
```

	location	avg_cost	avg_rate	total_votes
50	Lavelle Road	1365.02	4.14	495777
80	St. Marks Road	883.67	4.02	266099
42	Koramangala 3rd Block	834.47	4.02	123644
44	Koramangala 5th Block	680.65	4.01	2214083
12	Church Street	839.96	3.99	590306
74	Sankey Road	2582.69	3.97	6411
43	Koramangala 4th Block	758.32	3.92	685156
15	Cunningham Road	867.41	3.90	287471
69	Residency Road	1027.87	3.86	290513
46	Koramangala 7th Block	602.56	3.85	488225

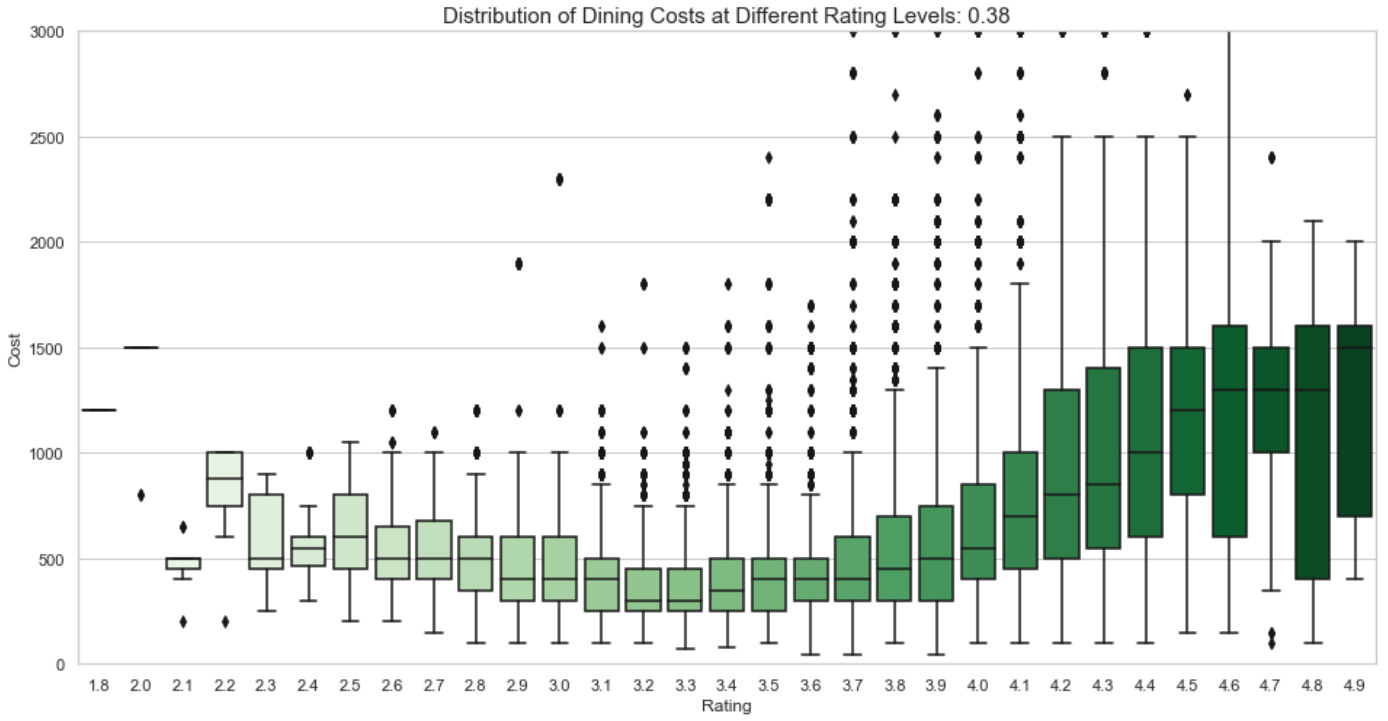
51	MG Road	1245.20	3.85	428266
67	Rajarajeshwari Nagar	725.00	3.85	732
71	Sadashiv Nagar	750.86	3.84	19727
27	Indiranagar	677.68	3.83	1165909
70	Richmond Road	903.90	3.82	118902
49	Langford Town	883.33	3.81	5281
28	Infantry Road	1071.22	3.80	51397
25	Hosur Road	661.81	3.79	8797
31	Jayanagar	500.40	3.78	480371
45	Koramangala 6th Block	637.60	3.78	463503
35	Kalyan Nagar	638.87	3.78	167992
65	Race Course Road	1321.48	3.78	27485
54	Malleshwaram	592.11	3.76	238967
82	Ulsoor	815.72	3.76	180232
41	Koramangala 2nd Block	685.06	3.75	7362
47	Koramangala 8th Block	396.23	3.74	23107
39	Koramangala	351.67	3.73	5764
38	Kengeri	500.00	3.71	382
8	Brigade Road	696.51	3.70	426682
40	Koramangala 1st Block	446.24	3.70	251424
76	Seshadripuram	737.32	3.70	17864
90	Yelahanka	325.00	3.70	176
75	Sarjapur Road	614.61	3.69	398599
59	New BEL Road	539.33	3.69	175687
29	JP Nagar	555.41	3.68	578010
22	HSR	501.47	3.68	498322
85	Vasanth Nagar	606.33	3.68	33955
21	HBR Layout	497.98	3.68	12371
36	Kammanahalli	503.25	3.67	105265
4	Basavanagudi	374.12	3.67	94914
56	Mysore Road	458.82	3.66	1142
1	Banashankari	451.95	3.65	161600
61	Old Airport Road	756.60	3.65	137832
5	Basaveshwara Nagar	467.29	3.65	17945
72	Sahakara Nagar	453.33	3.63	6817
88	Whitefield	679.53	3.62	465734
66	Rajajinagar	479.61	3.62	85274
11	Central Bangalore	387.50	3.62	1225
20	Frazer Town	451.69	3.61	97668
86	Vijay Nagar	395.65	3.61	7589
9	Brookefield	462.71	3.60	118962
23	Hebbal	425.00	3.60	450
32	Jeevan Bhima Nagar	406.88	3.59	46257
26	ITPL Main Road, Whitefield	589.10	3.58	7685
84	Varthur Main Road, Whitefield	347.62	3.58	5652
0	BTM	418.21	3.57	572188
14	Commercial Street	376.11	3.56	25563
55	Marathahalli	557.39	3.55	434235
16	Domlur	654.38	3.55	96721
81	Thippasandra	385.38	3.55	13023
89	Wilson Garden	428.07	3.54	7341
13	City Market	323.95	3.54	2778
33	KR Puram	385.00	3.54	277
6	Bellandur	571.18	3.53	205314
77	Shanti Nagar	513.80	3.52	55298
3	Bannerghatta Road	476.22	3.51	212496
34	Kaggadasapura	428.33	3.51	9231
83	Uttarahalli	416.67	3.51	702
2	Banaswadi	440.04	3.50	34861
58	Nagawara	461.78	3.50	19558
78	Shivajinagar	418.23	3.50	15668
24	Hennur	414.91	3.50	9605
91	Yeshwantpur	432.59	3.50	6980
53	Majestic	446.40	3.50	4034
79	South Bangalore	323.60	3.50	3646
73	Sanjay Nagar	348.96	3.50	3069

19	Electronic City	564.30	3.49	110774
30	Jalahalli	386.96	3.49	656
18	Ejipura	351.18	3.48	17015
10	CV Raman Nagar	345.83	3.48	3065
17	East Bangalore	412.50	3.48	2616
37	Kanakapura Road	523.68	3.48	747
48	Kumaraswamy Layout	373.10	3.47	16699
64	RT Nagar	443.33	3.46	6418
52	Magadi Road	362.50	3.44	847
57	Nagarbhavi	250.00	3.40	10
87	West Bangalore	466.67	3.37	1110
68	Rammurthy Nagar	426.92	3.35	1283
60	North Bangalore	325.00	3.34	1663
63	Peenya	300.00	3.20	5
7	Bommanahalli	477.78	3.19	7703
62	Old Madras Road	488.64	3.18	1608

Distribution of Dining Costs at Different Rating Levels

In [212…

```
correlation_coefficient = df['rate'].corr(df['cost'])

plt.figure(figsize=(16, 8))
sns.boxplot(x='rate', y='cost', data=df, palette='Greens')
plt.title(f'Distribution of Dining Costs at Different Rating Levels: {correlation_coeffi
plt.xlabel('Rating', fontsize=12)
plt.ylabel('Cost', fontsize=12)
plt.ylim([0,3000])
plt.show()
```



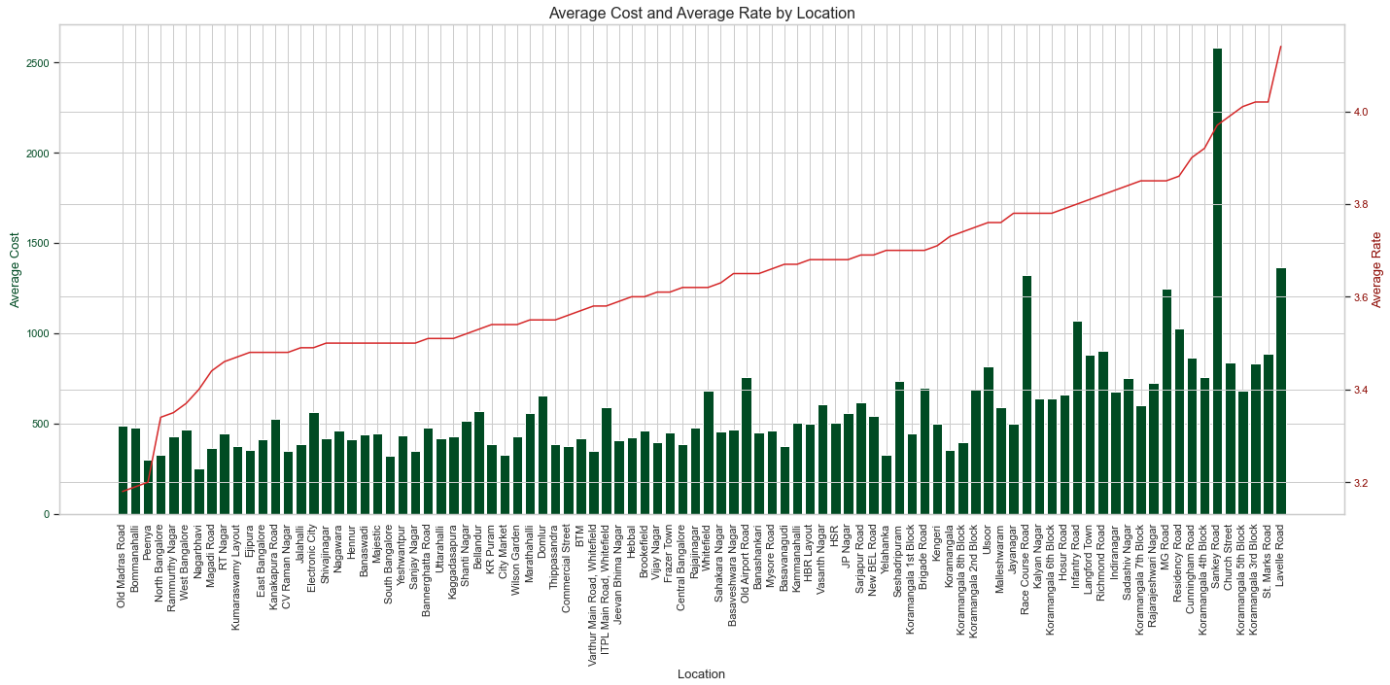
The correlation coefficient of 0.38 indicates a moderate positive correlation between restaurant ratings and dining costs. On average, higher-rated restaurants tend to have slightly higher dining costs

Average Cost and Average Rate by Location

In [213…

```
avg_data_sorted = avg_data.sort_values(by='avg_rate')

fig, ax1 = plt.subplots(figsize=(20, 10))

color = 'tab:blue'
ax1.set_xlabel('Location', fontsize=13)
```

```
ax1.set_ylabel('Average Cost', fontsize=13, color='#004b23')
ax1.bar(avg_data_sorted['location'], avg_data_sorted['avg_cost'], color='#004b23')
ax1.tick_params(axis='y', labelcolor='#004b23')

plt.xticks(rotation=90)

ax2 = ax1.twinx()
color = 'tab:red'
ax2.set_ylabel('Average Rate', fontsize=13, color='#8d0801')
ax2.plot(avg_data_sorted['location'], avg_data_sorted['avg_rate'], color=color)
ax2.tick_params(axis='y', labelcolor='#8d0801')

plt.title('Average Cost and Average Rate by Location', fontsize=16)
plt.tight_layout()
plt.show()
```

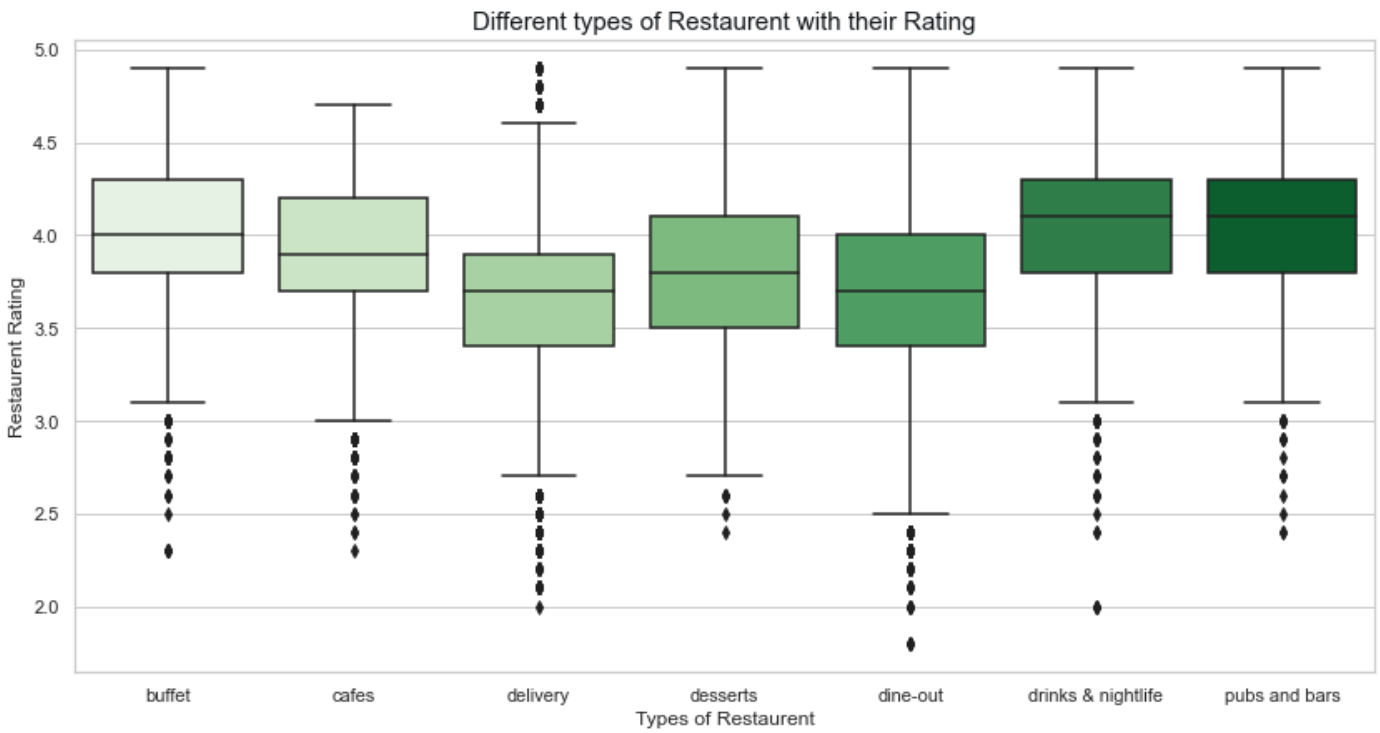


While Brigade Road exhibits the highest average cost at 696.51, St. Marks Road boasts the highest average rate of 4.02. This suggests a potential correlation between higher expenditure and superior dining experiences. Conversely, CV Raman Nagar represents the most budget-friendly option with an average cost of 345.83, albeit with a moderate average rate of 3.48. This suggests that cost doesn't always directly correlate with dining quality, as discerning diners may find hidden gems in more affordable locales.

Average Rating Based on Meal Types of Restaurent

In [214…

```
plt.figure(figsize = (14,7))
sns.boxplot(x= 'meal_type', y = 'rate', data = df, palette='Greens')
plt.title('Different types of Restaurent with their Rating',fontsize=15, color='#161a1d'
plt.xlabel('Types of Restaurent', fontsize = 12)
plt.ylabel('Restaurent Rating',  fontsize = 12)
plt.show()
```



The average ratings suggest that customers generally enjoy "Drinks & nightlife" and "Pubs and bars" more than other meal types. Consider focusing on enhancing the dining experience for "Desserts" to improve its rating.

Meal Types in terms of both High Average Ratings and Substantial Total Votes

In [215…
```
filtered_meal_type = df.groupby('meal_type').agg({'rate': 'mean', 'votes': 'sum'}).reset
filtered_meal_type = filtered_meal_type.sort_values(by='rate', ascending=False)
filtered_meal_type['rate'] = filtered_meal_type['rate'].round(2)
selected_columns = ['meal_type', 'rate', 'votes']

filtered_meal_type[selected_columns]
```

Out[215]:

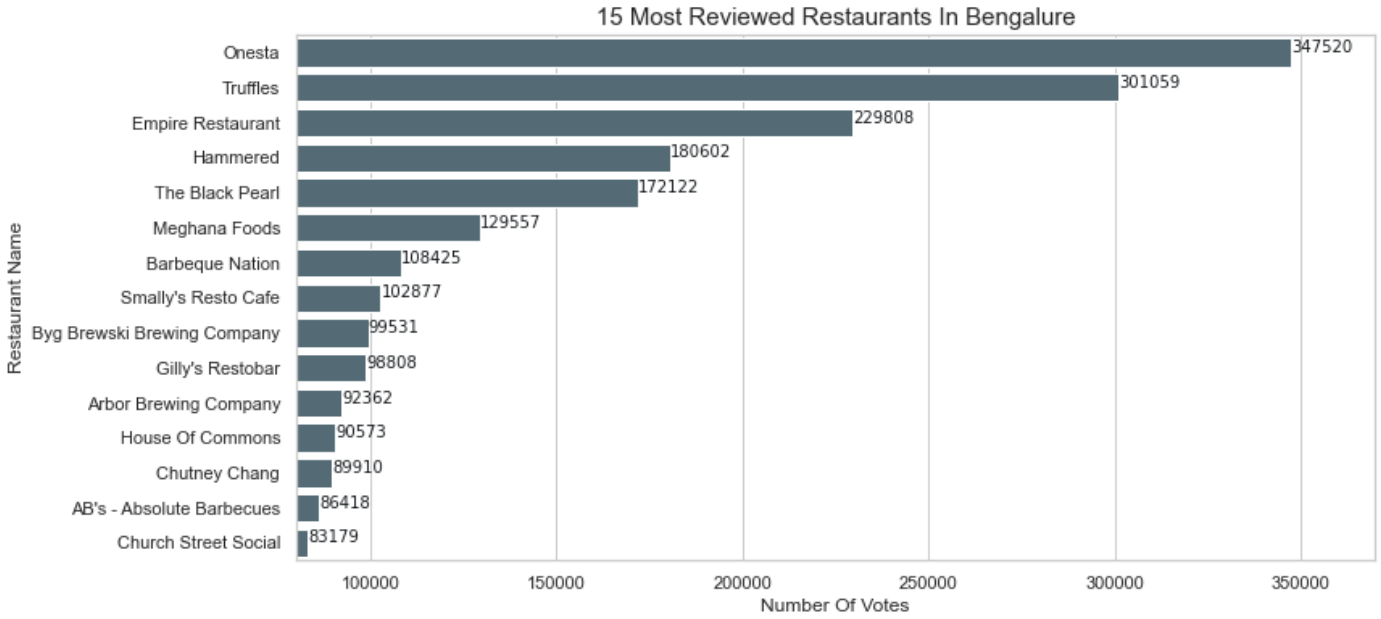
	meal_type	rate	votes
6	pubs and bars	4.02	692906
5	drinks & nightlife	4.01	1247364
0	buffet	3.98	871526
1	cafes	3.87	821558
3	desserts	3.78	536938
4	dine-out	3.68	5095623
2	delivery	3.65	5235854

"Drinks & nightlife" and "Pubs and bars" stand out with the highest average ratings of 4.00, signaling a positive customer experience. Interestingly, "Drinks & nightlife" boasts nearly double the votes compared to "Pubs and bars," highlighting its popularity. Notably, the "Delivery" category, while having a slightly lower average rating, garners the highest number of votes, underscoring its widespread appeal despite the rating difference.

Most Reviewed Restaurants In Bengalure

In [216…
```
df['votes'] = pd.to_numeric(df['votes'], errors='coerce')
```

```
top_voted = df.groupby(['name'])['votes'].agg({'sum'}).reset_index().rename(columns={'su
top_voted = top_voted.sort_values(by='total_votes', ascending=False).head(15)

fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(y='name', x='total_votes', data=top_voted,color ='#4f6d7a')

plt.ylabel('Restaurant Name', fontsize=12)
plt.xlabel('Number Of Votes', fontsize=12)
plt.title('15 Most Reviewed Restaurants In Bengalure', fontsize=15)

for v, count in enumerate(top_voted['total_votes']):
    ax.text(count, v, str(count), fontsize=10, family='DejaVu Sans', color='#161a1d')

plt.xlim([80000, 370000])
plt.show()
```



Bangalore's dining scene thrives with diverse choices, evident in the most-reviewed restaurants. "Onesta" and "Truffles" stand out, amassing impressive total votes of 347,520 and 301,059, respectively.
These establishments have cultivated a remarkable patronage, reflecting not only their culinary excellence but also their ability to captivate diners' hearts. As patrons lavish their votes, these restaurants stand as pillars of culinary distinction, enticing future restaurateurs to prioritize not just the cuisine but also the art of creating an unforgettable dining experience.

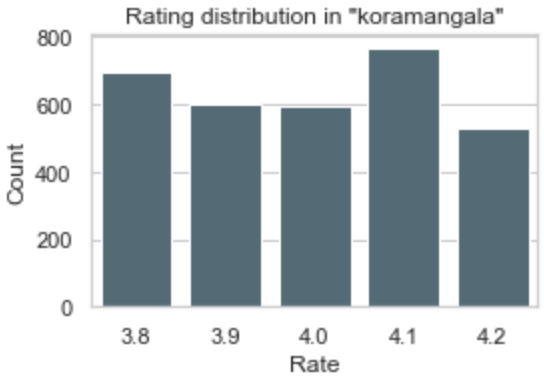
Highest Rated Restaurants Rating Distribution Based on Location

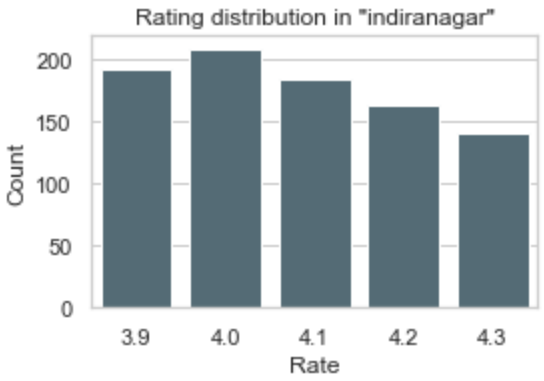
In [217…

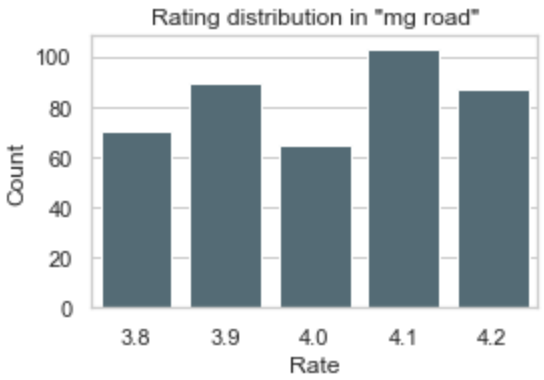
```
df['location_pr'] = df.location.apply(lambda x:re.sub('[0-9]+(st|th|nd|rd) block',
                                                      '', x.lower()).strip())
plt.figure(figsize = (14, 20))
loc_rate_grp = df.groupby(['location_pr', 'rate']).agg({'name':'count'}).reset_index()
loc_rate_grp.sort_values(by=['location_pr', 'name'], inplace=True, ascending=False)
loc_unq = df.location_pr.unique()
for i, loc in enumerate(loc_unq):
    data = loc_rate_grp[(loc_rate_grp['location_pr'] == loc)]
    data = data[:5]
    if data[data.rate > 4].shape[0] > 1:
        fig = plt.figure(figsize = (4, 2.5))
        sns.barplot(x='rate', y='name', data=data,color ='#4f6d7a')
        plt.title(f'Rating distribution in "{loc}"')
        plt.ylabel('Count')
        plt.xlabel('Rate')
```

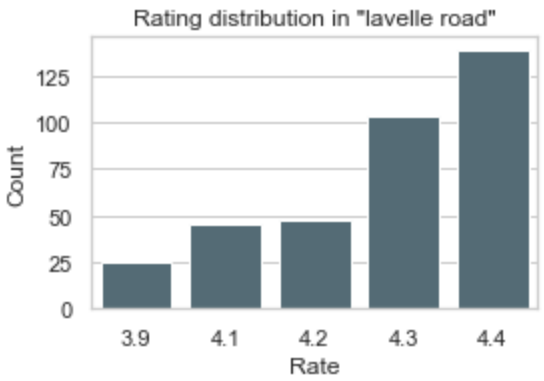
```
        plt.show()
df = df.drop('location_pr', axis=1)
```

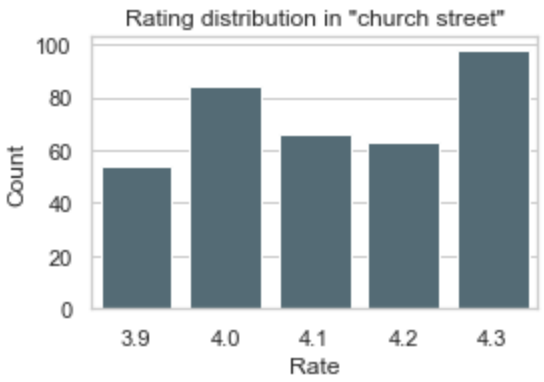
```
<Figure size 1008x1440 with 0 Axes>
```

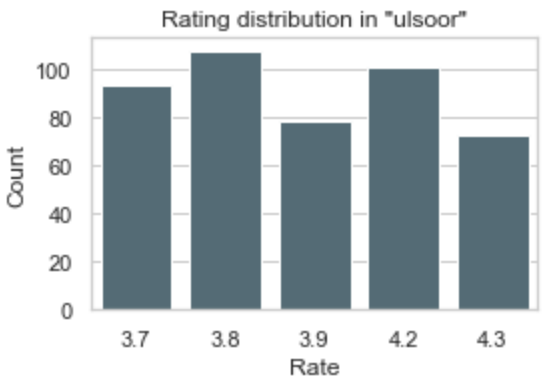


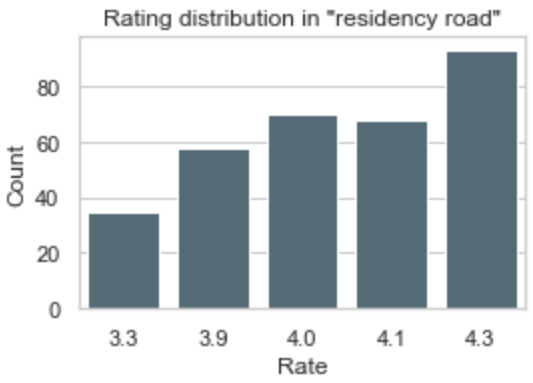


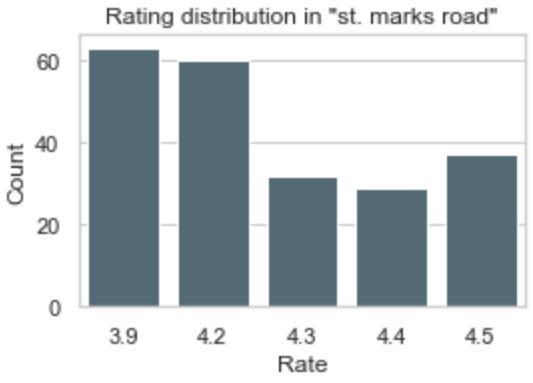


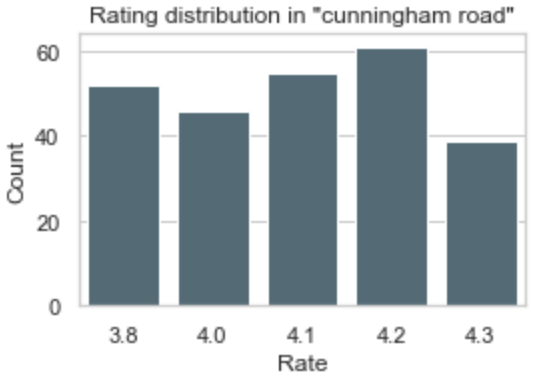


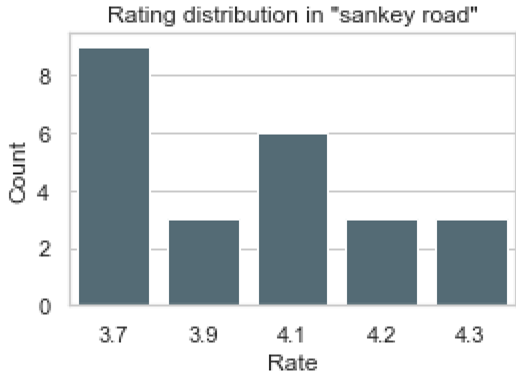

Rating distribution in "church street"
Count
Rate
0
20
40
60
80
100
3.9
4.0
4.1
4.2
4.3

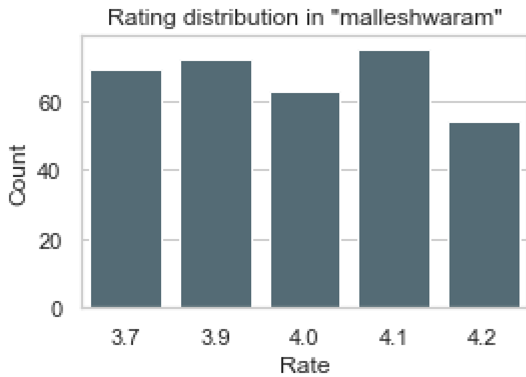

Rating distribution in "ulsoor"
Count
Rate
0
20
40
60
80
100
3.7
3.8
3.9
4.2
4.3

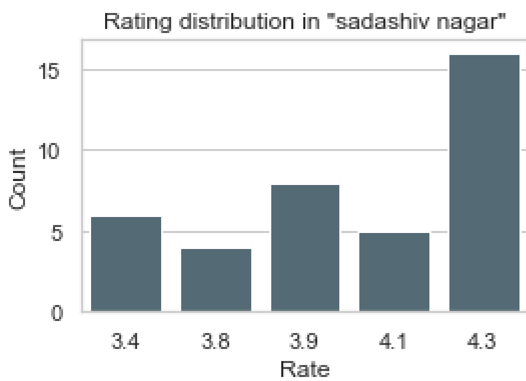

Rating distribution in "residency road"
Count
Rate
0
20
40
60
80
3.3
3.9
4.0
4.1
4.3


Rating distribution in "st. marks road"
Count
Rate
0
20
40
60
3.9
4.2
4.3
4.4
4.5


Rating distribution in "cunningham road"
Count
Rate
0
20
40
60
3.8
4.0
4.1
4.2
4.3







Lavelle Road, Church Street, and Residency Road emerge as culinary hotspots with restaurants boasting higher ratings compared to other locations. These prominent areas offer a gastronomic experience that has garnered favorable reviews and elevated ratings, making them stand out in the vibrant dining scene of the region

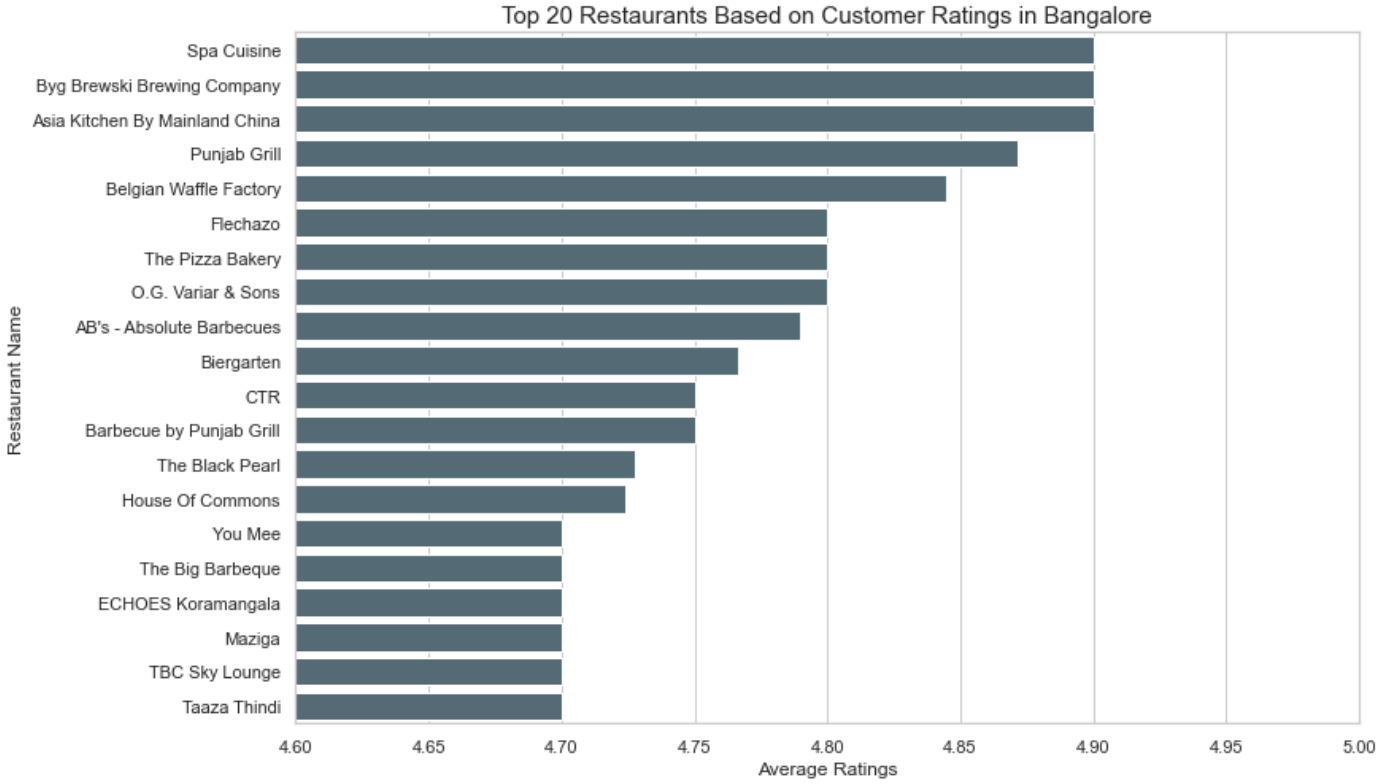
Top 20 Restaurants Based on Customer Ratings

In [218…

```
avg_rating = df.groupby(['name'])['rate'].mean().reset_index()
avg_rating= avg_rating.rename(columns={'rate':'avg_rating'}).sort_values(by='avg_rating'
avg_rating['name']= avg_rating.name.str.replace('SantÃ□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â
```

In [219…

```
fig, ax = plt.subplots(figsize=(12, 8))
sns.barplot(x='avg_rating',y='name',data=avg_rating.head(20),color='#4f6d7a')
plt.title('Top 20 Restaurants Based on Customer Ratings in Bangalore', fontsize=15)
plt.ylabel('Restaurant Name', fontsize=12)
plt.xlabel('Average Ratings', fontsize=12)
plt.xlim([4.6, 5.0])
plt.show()
```

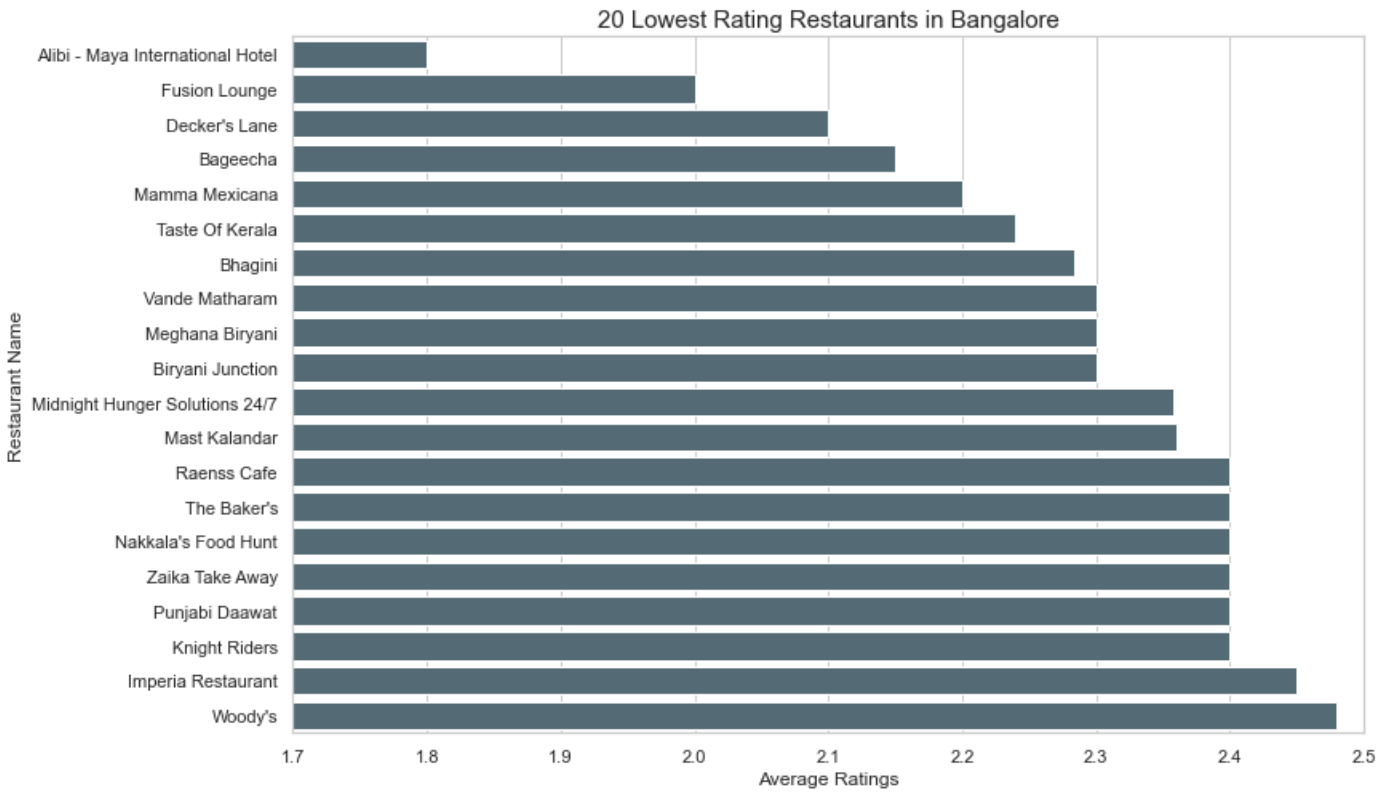


Restaurants like Punjab Grill, Spa Cuisine, and Byg Brewski Brewing Company have achieved exceptional average ratings, ranging up to 4.9. This highlights a trend where Bengaluru diners highly appreciate establishments that consistently deliver outstanding culinary experiences.
The top-rated restaurants offerings Belgian Waffle Factory's desserts to Flechazo's Asian-Mediterranean fusion. This diversity extends to the types of dining experiences, including casual dining, microbrewery, dessert parlors, and more. Entrepreneurs can leverage this insight to create unique dining destinations that cater to varied taste preferences and ambiance preferences.

20 Lowest Rating Restaurants in Bangalore

In [220…

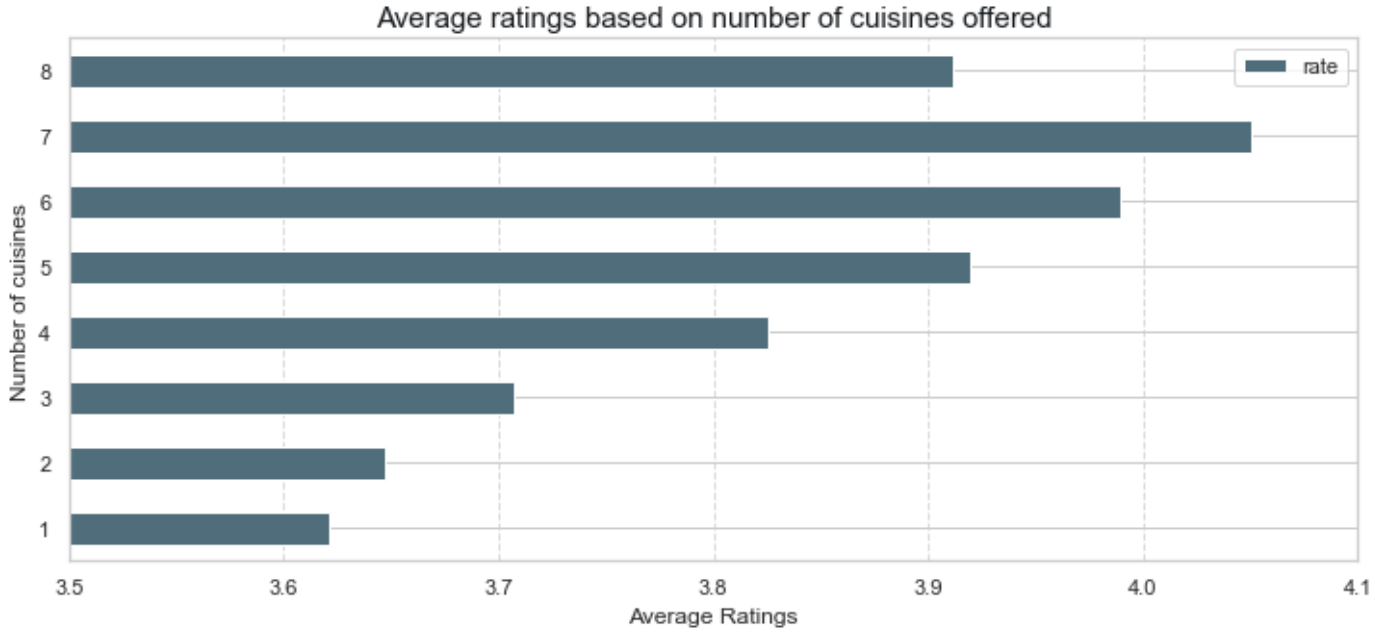
```
avg_rating = avg_rating.rename(columns={'rate': 'avg_rating'}).sort_values(by='avg_ratin
fig, ax = plt.subplots(figsize=(12, 8))
sns.barplot(x='avg_rating',y='name',data=avg_rating.head(20),color='#4f6d7a')
plt.title('20 Lowest Rating Restaurants in Bangalore', fontsize=15)
plt.ylabel('Restaurant Name', fontsize=12)
plt.xlabel('Average Ratings', fontsize=12)
plt.xlim([1.7, 2.5])
plt.show()
```



Average Ratings Based on Number of Cuisines Offered by a Restaurant

In [221…

```
df['Number_of_cuisines_offered'] = df['cuisines'].apply(lambda x : len(x.split(',')))

a = df.groupby('Number_of_cuisines_offered').agg({'rate':'mean'})

plt.rcParams['figure.figsize'] = (12,5)
ax = a.plot(kind='barh',  color='#4f6d7a')
ax.grid(axis='x', linestyle='--', alpha=0.7)

plt.title('Average ratings based on number of cuisines offered',
          fontsize=15, color='#161a1d')
plt.xlabel('Average Ratings', fontsize=12)
plt.ylabel('Number of cuisines', fontsize=12)
plt.xlim([3.5,4.1])
plt.show()
df = df.drop(columns= ['Number_of_cuisines_offered'], axis=1)
```



As the number of cuisines offered increases, there is a noticeable upward trend in average ratings. Restaurants presenting a diverse menu with 7 cuisines particularly shine, boasting a commendable mean rating of 4.05.
This suggests a strong positive correlation between menu variety and enhanced customer satisfaction, setting them apart in the competitive landscape.

Cheap Restaurants Whose Average Cost is Below 1000, Rating is Above 4.5 And Votes are Above 200 with Other Column

In [222…

```
restaurant_criteria = df.groupby('name').agg({'cost': 'mean',
                                              'rate': 'mean',
                                              'votes': 'sum',
                                               'location': 'first',
                                              'online_order': 'first',
                                               'meal_type': 'first',
                                              'book_table': 'first',
                                               'cuisines': 'first',
                                              'rest_type': 'first'}).reset_index()

cheap_restaurants = restaurant_criteria[(restaurant_criteria['cost'] < 1000) &
                                        (restaurant_criteria['rate'] > 4.5) &
                                        (restaurant_criteria['votes'] > 200)]

cheap_restaurants['cost'] = cheap_restaurants['cost'].round(2)
cheap_restaurants['rate'] = cheap_restaurants['rate'].round(2)

selected_columns = ['name', 'rate', 'votes', 'cost', 'location', 'online_order',
                    'meal_type', 'book_table', 'cuisines', 'rest_type']

cheap_restaurants[selected_columns].reset_index(drop=True)
```

Out[222]:

	name	rate	votes	cost	location	online_order	meal_type	book_table	cuisines	rest_type
0	Baar Union	4.60	3205	850.0	HSR	False	dine-out	True	Continental, Finger Food, Asian, Chinese	bar, pub
1	Belgian Waffle Factory	4.84	24882	400.0	Brigade Road	True	delivery	False	Desserts	dessert parlor
2	CTR	4.75	17658	150.0	Malleshwaram	True	delivery	False	South Indian	quick bites
3	Dock Frost'd	4.60	231	400.0	Marathahalli	True	delivery	False	Beverages, Desserts	beverage shop, dessert parlor
4	ECHOES Koramangala	4.70	16063	750.0	Koramangala 5th Block	False	dine-out	False	Chinese, American, Continental, Italian, North...	cafe, casual dining
5	Here & Now	4.60	6339	750.0	HSR	False	cafes	False	Cafe, American, Burger, Beverages	cafe
6	Kurtoskalacs	4.62	1421	390.0	Koramangala 5th Block	False	desserts	False	Desserts, Fast Food	dessert parlor,

										quick bites
7	Lot Like Crepes	4.62	20510	550.0	Koramangala 7th Block	True	cafes	False	Cafe, Desserts, Continental	cafe, dessert parlor
8	Milano Ice Cream	4.60	2703	400.0	Jayanagar	False	desserts	False	Ice Cream, Desserts	dessert parlor
9	Mugful Of Stories	4.60	548	300.0	Kalyan Nagar	False	desserts	False	Desserts, Bakery	dessert parlor
10	O.G. Variar & Sons	4.80	2317	200.0	Rajajinagar	False	desserts	False	Bakery, Desserts	bakery
11	Taaza Thindi	4.70	651	100.0	Banashankari	False	dine-out	False	South Indian	quick bites
12	The Blue Wagon - Kitchen	4.60	2334	400.0	Jayanagar	True	delivery	False	Cafe, Beverages	cafe
13	The Hole in the Wall Cafe	4.60	56999	600.0	Koramangala 4th Block	False	cafes	False	Cafe, American, Burger	cafe
14	The Pancake Story	4.60	717	300.0	Koramangala 1st Block	True	delivery	False	Desserts, Beverages	dessert parlor
15	Truffles	4.60	301059	900.0	St. Marks Road	False	cafes	False	Cafe, American, Burger, Steak	cafe

Unearth hidden culinary treasures in Bengaluru with Baar Union, CTR, Echoes Koramangala, and others. These budget-friendly havens offer exquisite dishes, from Continental and South Indian to desserts, at enticing prices below 1000 Rupees.
Explore a variety of dining experiences, from casual hangouts like Lot Like Crepes and Mugful of Stories to classic favorites like Truffles. These restaurants, scattered across locations like HSR, Koramangala, and St. Marks Road, not only exceed a 4.5 rating but also accumulate substantial votes, ensuring a delightful and popular dining escapade.

Expensive Restaurants Whose Average Cost Is Above 3000, Rating Is Above or Equal To 4.3 and Votes Are Above 200

In [223…

```
restaurant_criteria = df.groupby('name').agg({'cost': 'mean',
                                              'rate': 'mean',
                                              'votes': 'sum',
                                               'location': 'first',
                                              'online_order': 'first',
                                               'meal_type': 'first',
                                              'book_table': 'first',
                                               'cuisines': 'first',
                                              'rest_type': 'first'}).reset_index()

expensive_restaurants = restaurant_criteria[(restaurant_criteria['cost'] > 3000) &
                                            (restaurant_criteria['rate'] >= 4.3) &
                                            (restaurant_criteria['votes'] > 200)]

expensive_restaurants['cost'] = expensive_restaurants['cost'].round(2)
expensive_restaurants['rate'] = expensive_restaurants['rate'].round(2)
```

```
selected_columns = ['name', 'rate', 'votes', 'cost', 'location', 'online_order',
                    'meal_type', 'book_table', 'cuisines', 'rest_type']
expensive_restaurants[selected_columns].reset_index(drop=True)
```

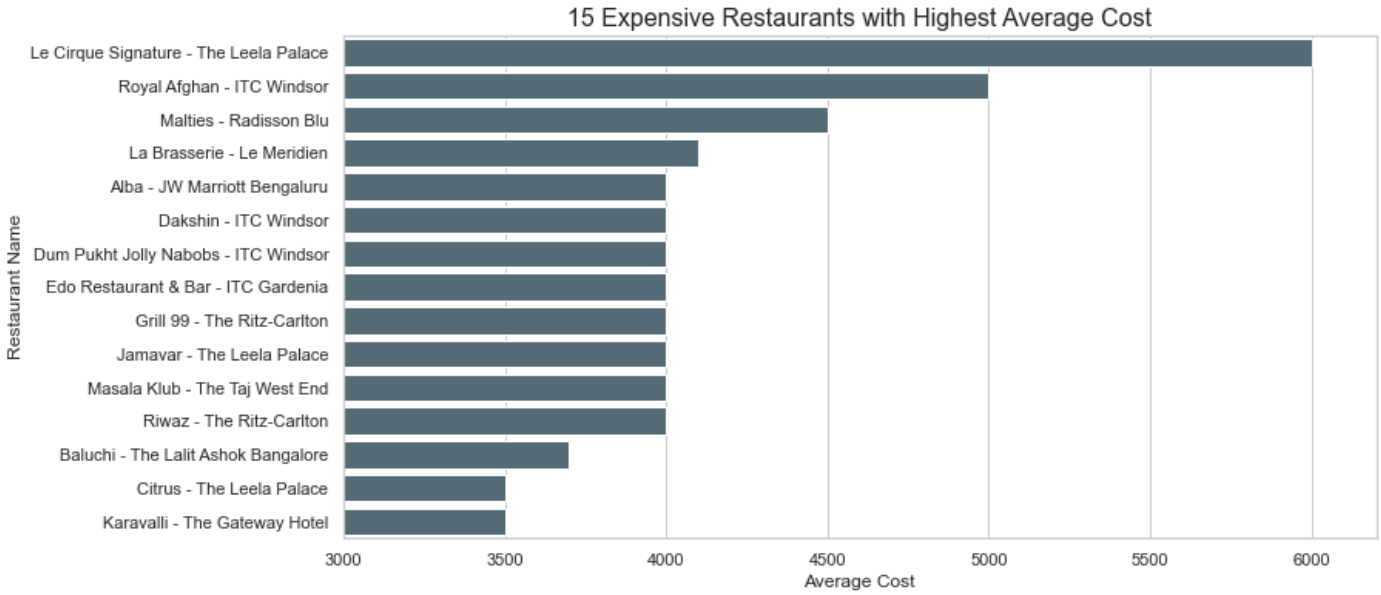
Out[223]:

	name	rate	votes	cost	location	online_order	n
0	Alba - JW Marriott Bengaluru	4.43	2293	4000.0	Lavelle Road	False	
1	Blue Ginger - The Taj West End	4.40	1030	3200.0	Race Course Road	False	
2	Dakshin - ITC Windsor	4.30	202	4000.0	Sankey Road	False	
3	Dum Pukht Jolly Nabobs - ITC Windsor	4.30	291	4000.0	Sankey Road	False	
4	Edo Restaurant & Bar - ITC Gardenia	4.30	1175	4000.0	Richmond Road	False	
5	Karavalli - The Gateway Hotel	4.50	3393	3500.0	Residency Road	False	
6	Kebabs & Kurries - ITC Gardenia	4.30	1800	3500.0	Richmond Road	False	
7	Le Cirque Signature - The Leela Palace	4.30	254	6000.0	Old Airport Road	False	
8	Masala Klub - The Taj West End	4.40	618	4000.0	Race Course Road	False	
9	Royal Afghan - ITC Windsor	4.30	413	5000.0	Sankey Road	False	
10	The Lantern Restaurant & Bar - The Ritz-Carlto...	4.34	3848	3500.0	Residency Road	False	
11	The Market - The Ritz-Carlton	4.30	6181	3400.0	Residency Road	False	
12	CafÃ□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã... b	4.30	1287	3400.0	Vasanth Nagar	False	

Bengaluru boasts elite fine dining establishments that offer an exquisite culinary experience. Renowned venues like "Alba - JW Marriott Bengaluru" and "Blue Ginger - The Taj West End" stand out, showcasing exceptional ratings, substantial votes, and a premium cost range.
The city's upscale dining scene encompasses a rich tapestry of cuisines. From the authentic South Indian flavors at "Dakshin - ITC Windsor" to the Japanese delights of "Edo Restaurant & Bar - ITC Gardenia," and the fusion brilliance of "The Lantern Restaurant & Bar - The Ritz-Carlton," each establishment promises a sophisticated gastronomic journey for discerning patrons.

15 Expensive Restaurants with Highest Average Cost

In [224…
```
expensive_avg_cost_rest = df.groupby('name')['cost'].mean().round()
expensive_avg_cost_rest = expensive_avg_cost_rest.astype(int).nlargest(15)
fig, ax = plt.subplots(figsize=(12, 6))
sns.barplot(x=expensive_avg_cost_rest,
            y=expensive_avg_cost_rest.index, color='#4f6d7a')

plt.title('15 Expensive Restaurants with Highest Average Cost',
          fontsize=16)
plt.xlabel('Average Cost', fontsize=12)
plt.ylabel('Restaurant Name', fontsize=12)
plt.xlim([3000,6200])
plt.show()
```



"Le Cirque Signature - The Leela Palace" leads with an opulent average cost of 6000, followed by renowned establishments like "Royal Afghan - ITC Windsor" and "Malties - Radisson Blu."
Exploring these premium venues promises an indulgent gastronomic journey for those seeking an extraordinary and lavish dining affair.

Most Famous Types of Restaurants in Bangalurue

Each types of restaurants frequency count

In [225…
```
Mlb = MultiLabelBinarizer()
df['rest_type_temp'] = df['rest_type'].apply(lambda r : r.replace(', ', ',').split(','))
df_rest_type=pd.DataFrame(Mlb.fit_transform(df['rest_type_temp']) ,
                          columns= Mlb.classes_)
rest_type_count = df_rest_type.sum().sort_values(ascending=False)
rest_type_count = pd.DataFrame({'rest_type': rest_type_count.index,
                                'frequency': rest_type_count.values})

rest_type_count
```

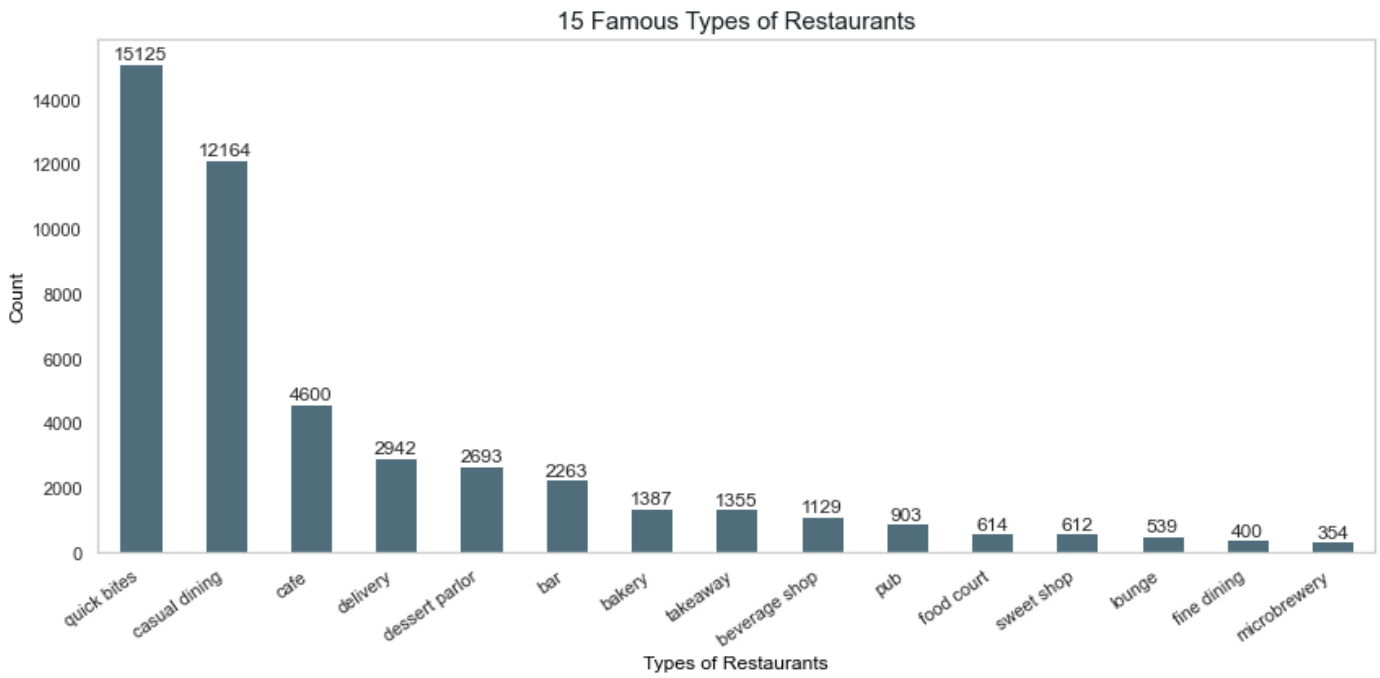
Out[225]:

	rest_type	frequency
0	quick bites	15125
1	casual dining	12164
2	cafe	4600
3	delivery	2942
4	dessert parlor	2693
5	bar	2263

6	bakery	1387
7	takeaway	1355
8	beverage shop	1129
9	pub	903
10	food court	614
11	sweet shop	612
12	lounge	539
13	fine dining	400
14	microbrewery	354
15	mess	184
16	kiosk	155
17	food truck	68
18	club	40
19	dhaba	17
20	confectionery	15
21	irani cafee	15
22	meat shop	4
23	bhojanalya	3

In [226…

```
fig, ax = plt.subplots(figsize=(12, 6))
top_rest_type = df_rest_type.sum().sort_values(ascending=False).head(15)
bars = top_rest_type.plot(kind='bar', color='#4f6d7a')

plt.title('15 Famous Types of Restaurants', fontsize=15, color='#161a1d')
plt.xlabel('Types of Restaurants', fontsize=12, color='black')
plt.ylabel('Count', fontsize=12, color='black')

for bar in bars.patches:
    plt.annotate(format(bar.get_height(), '.0f'),
                 (bar.get_x() + bar.get_width() / 2,
                  bar.get_height()), ha='center', va='center',
                 xytext=(0, 5), textcoords='offset points')

plt.xticks(rotation=35, ha='right')
plt.tight_layout()
plt.grid(False)
plt.show()
df.drop('rest_type_temp', axis=1, inplace=True)
```

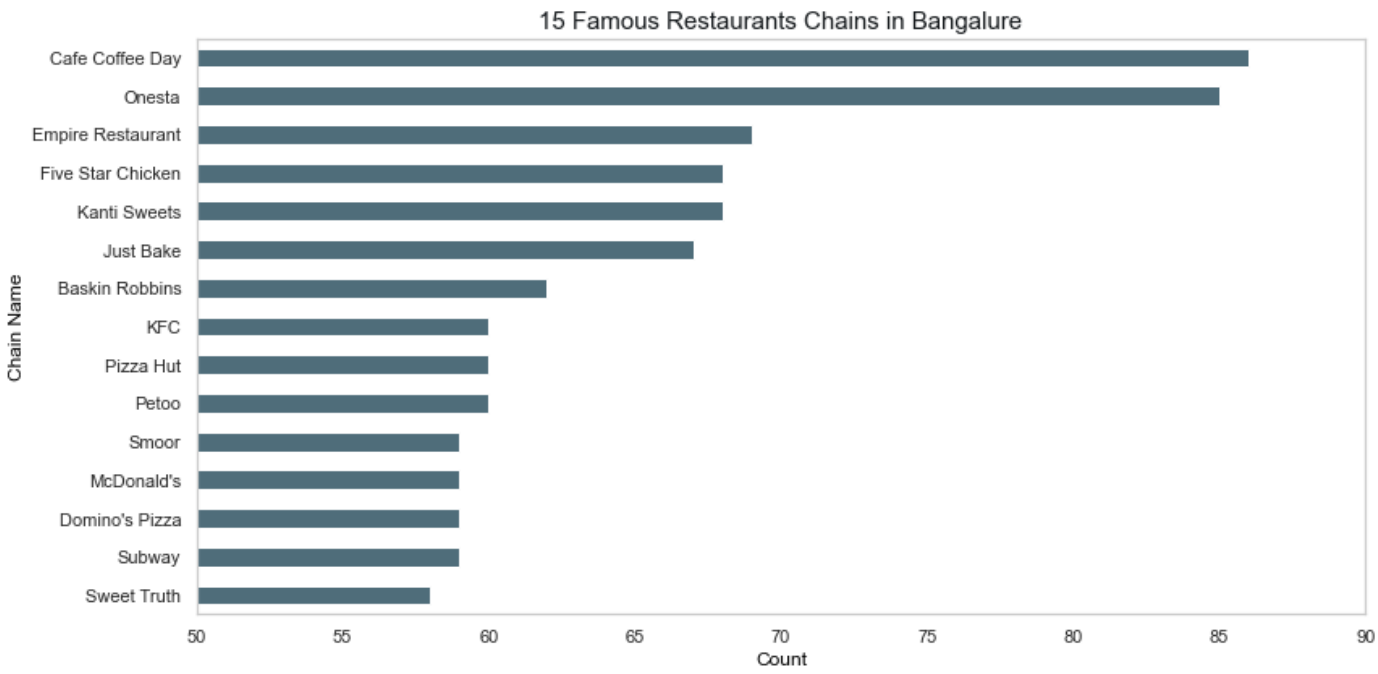


"Quick Bites" and "Casual Dining" dominate the restaurant landscape, with 15,125 and 12,164 establishments respectively. "Cafes" also play a significant role, totaling 4,600. Consider the vibrant quick bites and diverse casual dining options for a well-rounded dining experience in Bengaluru

Most Famous Restaurants Chains in Bangalure

In [227…

```
fig,ax = plt.subplots(figsize=(12,6))

df['name'].value_counts()[:15].sort_values(ascending=True).plot(kind='barh',
                                                                color='#4f6d7a')

plt.title('15 Famous Restaurants Chains in Bangalure',
          fontsize=15, color='#161a1d')
plt.ylabel('Chain Name', fontsize=12, color='black')
plt.xlabel('Count', fontsize=12, color='black')
plt.tight_layout()
plt.xlim([50,90])
ax.grid(axis='x', linestyle='--', alpha=0.4)
plt.grid(False)
plt.show()
```



Among the top restaurants in Bengaluru, "Onesta" and "Cafe Coffee Day" dominate with significantly higher occurrences, indicating their widespread popularity.
Cafe Coffee Day Dominance: Cafe Coffee Day stands out as Zomato Bengaluru's go-to spot, offering diverse dishes, from South Indian delights to Italian and Mexican favorites. Its inviting ambiance, multiple locations, and competitive prices contribute to its popularity and excellent value for customers.

Most Famous Cuisines in Bangalure

Top Cuisines Based on Location

In [228…

```
df_combined = pd.concat([df[['location']], df_cuisines], axis=1)

top_cuisines_by_location = df_combined.groupby('location').sum().idxmax(axis=1)

second_top_cuisines_by_location = df_combined.groupby('location').sum().apply(lambda row
                                                                               axis=1)

top_cuisines_df = pd.DataFrame({'location': top_cuisines_by_location.index,
                                'top_cuisine': top_cuisines_by_location.values,
                                'second_top_cuisine': second_top_cuisines_by_location.va

print(top_cuisines_df)
```

```
                      location   top_cuisine second_top_cuisine
0                          BTM  North Indian            Chinese
1                Banashankari  North Indian            Chinese
2                   Banaswadi  North Indian            Chinese
3           Bannerghatta Road  North Indian            Chinese
4                Basavanagudi  North Indian            Chinese
5           Basaveshwara Nagar       Biryani          Ice Cream
6                    Bellandur  North Indian            Chinese
7                Bommanahalli  North Indian            Chinese
8                Brigade Road  North Indian            Chinese
9                  Brookefield  North Indian            Chinese
10              CV Raman Nagar  North Indian            Chinese
11          Central Bangalore  North Indian             Bakery
12              Church Street  North Indian            Chinese
13                City Market  North Indian            Chinese
14          Commercial Street  North Indian            Chinese
15            Cunningham Road  North Indian            Chinese
16                      Domlur  North Indian            Chinese
```

East Bangalore	Chinese	North Indian	
Ejipura	North Indian	Chinese	
Electronic City	North Indian	Chinese	
Frazer Town	North Indian	Chinese	
HBR Layout	North Indian	Fast Food	
HSR	North Indian	Chinese	
Hebbal	Afghan	Afghani	
Hennur	North Indian	Chinese	
Hosur Road	North Indian	Chinese	
ITPL Main Road, Whitefield	Chinese	North Indian	
Indiranagar	North Indian	Chinese	
Infantry Road	North Indian	Chinese	
JP Nagar	North Indian	Chinese	
Jalahalli	Afghan	Afghani	
Jayanagar	North Indian	Chinese	
Jeevan Bhima Nagar	Chinese	North Indian	
KR Puram	Afghan	Afghani	
Kaggadasapura	Chinese	North Indian	
Kalyan Nagar	North Indian	Chinese	
Kammanahalli	North Indian	Chinese	
Kanakapura Road	North Indian	Desserts	
Kengeri	Chinese	Biryani	
Koramangala	North Indian	Biryani	
Koramangala 1st Block	North Indian	Chinese	
Koramangala 2nd Block	North Indian	Chinese	
Koramangala 3rd Block	North Indian	Chinese	
Koramangala 4th Block	North Indian	Chinese	
Koramangala 5th Block	North Indian	Chinese	
Koramangala 6th Block	North Indian	Chinese	
Koramangala 7th Block	North Indian	Chinese	
Koramangala 8th Block	North Indian	Chinese	
Kumaraswamy Layout	North Indian	Chinese	
Langford Town	Chinese	Desserts	
Lavelle Road	North Indian	Chinese	
MG Road	North Indian	Chinese	
Magadi Road	North Indian	Afghan	
Majestic	North Indian	Chinese	
Malleshwaram	North Indian	Chinese	
Marathahalli	North Indian	Chinese	
Mysore Road	North Indian	Chinese	
Nagarbhavi	Afghan	Afghani	
Nagawara	North Indian	Chinese	
New BEL Road	North Indian	Chinese	
North Bangalore	North Indian	Chinese	
Old Airport Road	North Indian	Chinese	
Old Madras Road	North Indian	Biryani	
Peenya	Afghan	Afghani	
RT Nagar	North Indian	Chinese	
Race Course Road	North Indian	Chinese	
Rajajinagar	North Indian	Chinese	
Rajarajeshwari Nagar	Beverages	Chinese	
Rammurthy Nagar	North Indian	Chinese	
Residency Road	North Indian	Chinese	
Richmond Road	North Indian	Chinese	
Sadashiv Nagar	Chinese	North Indian	
Sahakara Nagar	Afghan	Afghani	
Sanjay Nagar	North Indian	Desserts	
Sankey Road	North Indian	Chinese	
Sarjapur Road	North Indian	Chinese	
Seshadripuram	North Indian	Chinese	
Shanti Nagar	North Indian	Chinese	
Shivajinagar	North Indian	Chinese	
South Bangalore	North Indian	Chinese	
St. Marks Road	North Indian	Chinese	
Thippasandra	North Indian	Chinese	
Ulsoor	North Indian	Chinese	

```
83                    Uttarahalli       Chinese       North Indian
84  Varthur Main Road, Whitefield       Chinese       North Indian
85                  Vasanth Nagar  North Indian            Chinese
86                    Vijay Nagar   Street Food          Beverages
87                 West Bangalore       Biryani       North Indian
88                     Whitefield  North Indian            Chinese
89                   Wilson Garden  North Indian            Chinese
90                       Yelahanka       Chinese       North Indian
91                     Yeshwantpur  North Indian            Chinese
```

It highlights a consistent preference for North Indian cuisine across various locations, followed closely by Chinese cuisine. This indicates a widespread appeal for rich and diverse Indian flavors, with Chinese cuisine serving as a popular alternative. While North Indian dishes dominate as the top choice, the presence of other cuisines such as Afghan and Biryani in certain locations reflects diverse culinary influences.

Cuisines Frequency

In [229…

```
pd.set_option('display.max_rows', 105)
Mlb = MultiLabelBinarizer()

df['cuisines_temp'] = df['cuisines'].apply(lambda r : r.replace(', ', ',').split(','))
df_cuisines=pd.DataFrame(Mlb.fit_transform(df['cuisines_temp']) ,
                         columns= Mlb.classes_)
cuisines_count = df_cuisines.sum().sort_values(ascending=False)
cuisines_count = pd.DataFrame({'cuisines': cuisines_count.index,
                               'frequency': cuisines_count.values})
display(cuisines_count)
```

	cuisines	frequency
0	North Indian	17188
1	Chinese	12842
2	South Indian	6353
3	Fast Food	6312
4	Continental	5180
5	Biryani	5012
6	Cafe	4781
7	Desserts	4501
8	Beverages	3830
9	Italian	3167
10	Street Food	2207
11	Bakery	2023
12	Pizza	1866
13	Burger	1841
14	Seafood	1646
15	Andhra	1467
16	Ice Cream	1427
17	Mughlai	1384
18	American	1331

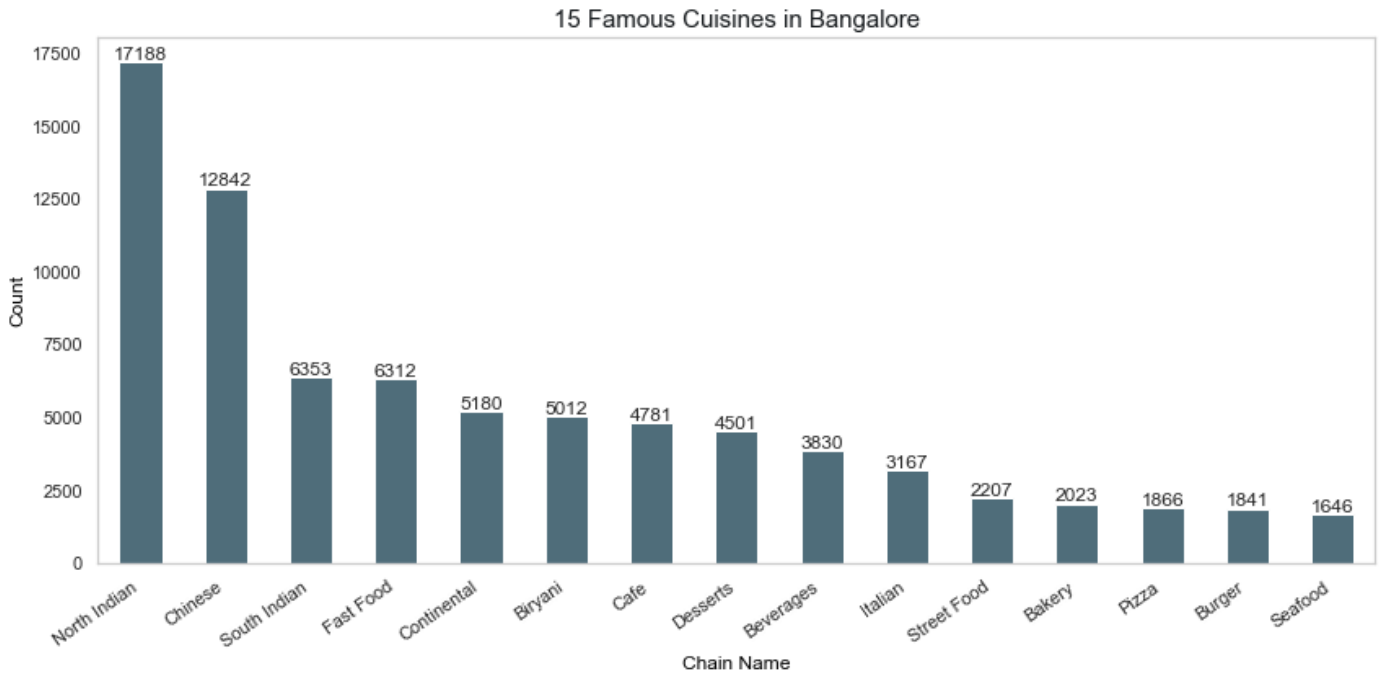
19	Asian	1196
20	Kerala	1188
21	Rolls	1166
22	Momos	1150
23	Finger Food	1091
24	Salad	1055
25	Thai	946
26	Arabian	907
27	Mithai	832
28	Juices	828
29	Healthy Food	772
30	Kebab	768
31	Sandwich	717
32	European	682
33	BBQ	674
34	Mangalorean	636
35	Steak	552
36	Mediterranean	543
37	Bengali	488
38	Mexican	483
39	Japanese	338
40	Tibetan	300
41	Tea	295
42	Hyderabadi	257
43	Chettinad	192
44	Lebanese	167
45	Vietnamese	166
46	Rajasthani	157
47	Modern Indian	143
48	Korean	140
49	Maharashtrian	122
50	Oriya	120
51	Goan	113
52	Malaysian	106
53	Gujarati	98
54	Coffee	97
55	Indonesian	96
56	Middle Eastern	85

57	Konkan	81
58	Turkish	80
59	French	74
60	Nepalese	70
61	Charcoal Chicken	62
62	Afghan	59
63	Bihari	58
64	Burmese	55
65	Tex-Mex	51
66	Wraps	50
67	Lucknowi	49
68	Singaporean	47
69	Spanish	46
70	Sushi	44
71	Greek	38
72	North Eastern	37
73	Iranian	29
74	Awadhi	27
75	Naga	25
76	Assamese	25
77	Grill	24
78	Parsi	23
79	Bar Food	23
80	Kashmiri	22
81	Roast Chicken	21
82	African	17
83	Sri Lankan	15
84	Mongolian	12
85	Portuguese	11
86	Bubble Tea	10
87	South American	10
88	Afghani	8
89	German	7
90	British	7
91	Cantonese	6
92	Jewish	6
93	Bohri	6

94	Tamil	6
95	Russian	6
96	Vegan	6
97	Australian	5
98	Raw Meats	4
99	Sindhi	4
100	Drinks Only	4
101	Belgian	3
102	Hot dogs	3
103	Paan	2
104	Pan Asian	1

In [230…

```
fig, ax = plt.subplots(figsize=(12, 6))
top_cuisines = df_cuisines.sum().sort_values(ascending=False).head(15)
bars = top_cuisines.plot(kind='bar', color='#4f6d7a')

plt.title('15 Famous Cuisines in Bangalore', fontsize=15, color='#161a1d')
plt.xlabel('Chain Name', fontsize=12, color='black')
plt.ylabel('Count', fontsize=12, color='black')

for bar in bars.patches:
    plt.annotate(format(bar.get_height(), '.0f'),
                 (bar.get_x() + bar.get_width() / 2,
                  bar.get_height()), ha='center', va='center',
                 xytext=(0, 5), textcoords='offset points')

plt.xticks(rotation=35, ha='right')
plt.tight_layout()
plt.grid(False)
plt.show()
df = df.drop(['cuisines_temp'], axis= 1)
```



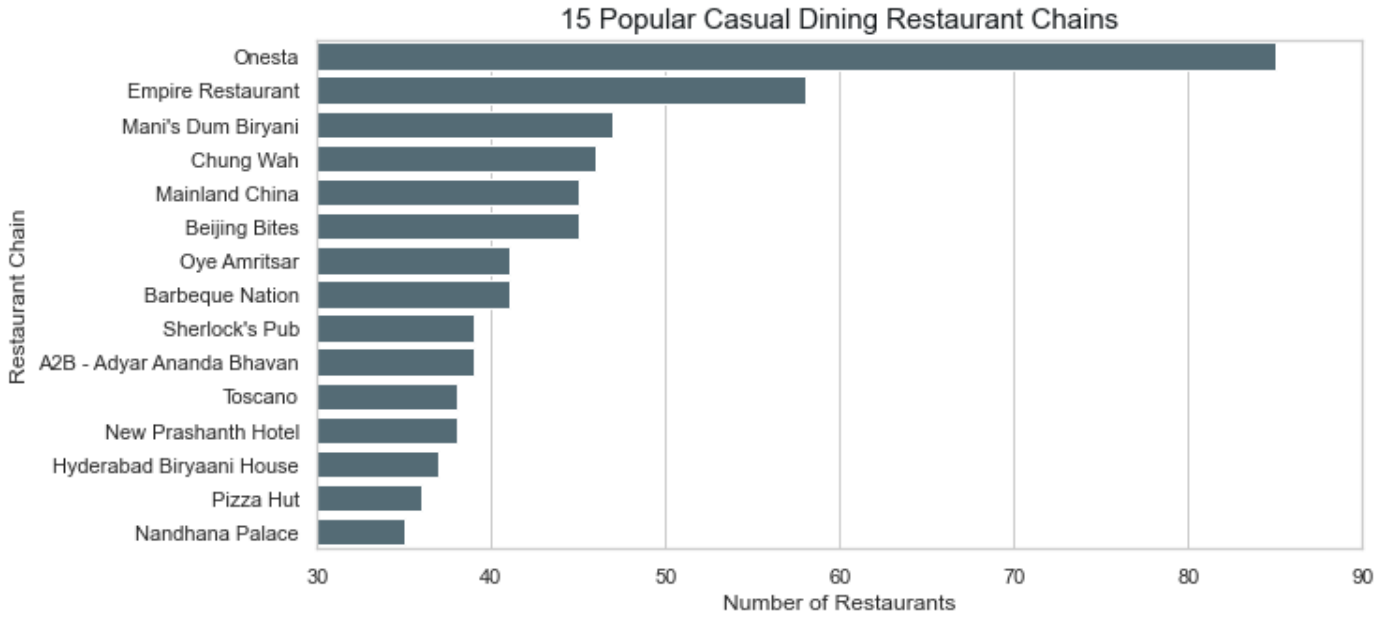
North Indian cuisine reigns supreme with 17,188 mentions, followed closely by Chinese delights at 12,842. South Indian and Fast Food also make a strong presence, reflecting the city's love for regional and quick-

bite options. The continental, biryani, and cafe offerings add to the flavorful mix. Desserts and beverages complete the top choices, showcasing the city's sweet tooth.
It is enchanted by North Indian cuisine, celebrated for its rich, aromatic flavors, diverse spices, and herb-infused dishes. The cuisine's vegetarian options further elevate its appeal, providing a delightful dining experience for all palates.

Popular Casual Dining Restaurant Chains

In [231…

```
casual_dining_chains = df[df['rest_type'].str.contains('Casual Dining',
                                                       case=False, na=False)]
popular_chains = casual_dining_chains['name'].value_counts().reset_index()

popular_chains.columns = ['Restaurant Chain', 'Count']
popular_chains = popular_chains.sort_values(by='Count', ascending=False)
top_15_chains = popular_chains.head(15)

plt.figure(figsize=(10, 5))
sns.barplot(x='Count', y='Restaurant Chain', data=top_15_chains,color='#4f6d7a')
plt.xlabel('Number of Restaurants', fontsize=12)
plt.title('15 Popular Casual Dining Restaurant Chains',
          fontsize=15, color='#161a1d')
plt.xlim([30,90])
plt.show()
```



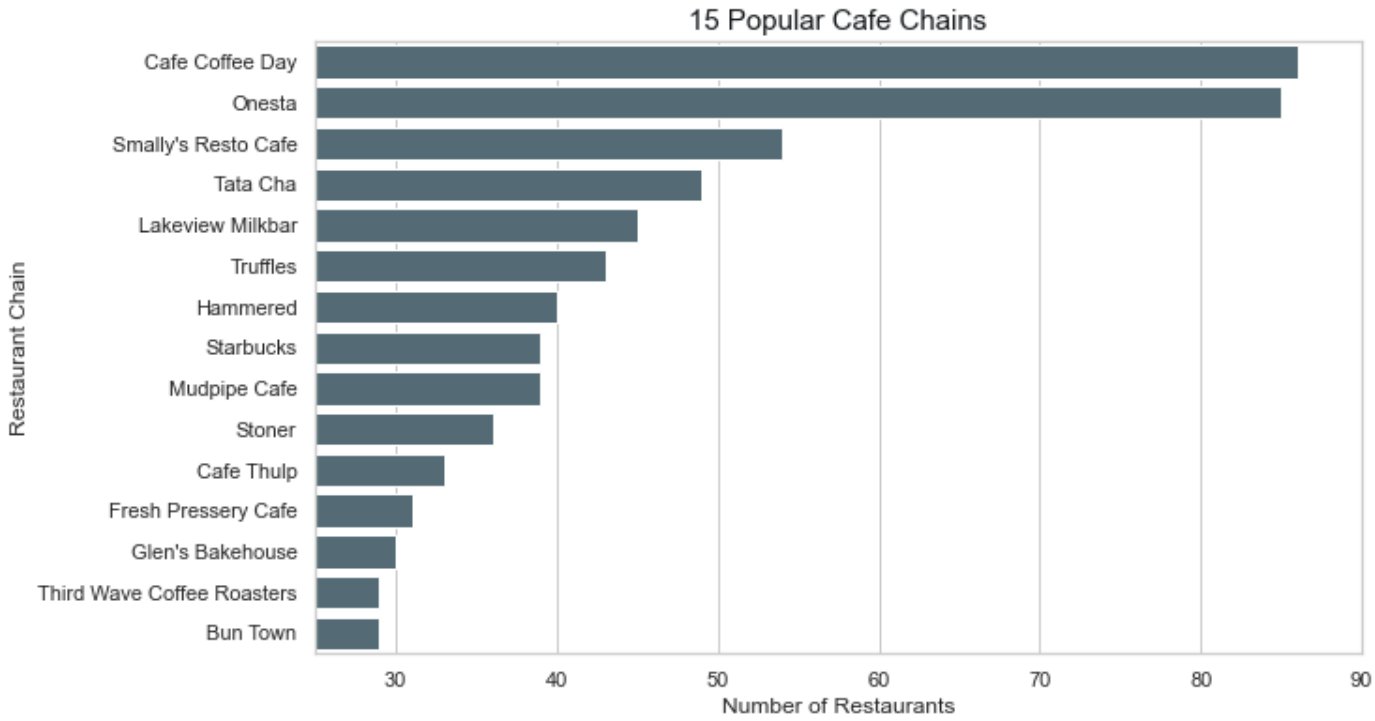
"Onesta" leading the pack with 85 locations. Other noteworthy chains include "Empire Restaurant," "Mani's Dum Biryani," and "Chung Wah."

Popular Cafe Chains

In [232…

```
Cafe_chains = df[df['rest_type'].str.contains('Cafe', case=False, na=False)]
popular_chains = Cafe_chains['name'].value_counts().reset_index()

popular_chains.columns = ['Restaurant Chain', 'Count']
popular_chains = popular_chains.sort_values(by='Count', ascending=False)
top_15_chains = popular_chains.head(15)

plt.figure(figsize=(10, 6))
sns.barplot(x='Count', y='Restaurant Chain', data=top_15_chains,color='#4f6d7a')
plt.xlabel('Number of Restaurants', fontsize=12)
plt.title('15 Popular Cafe Chains',  fontsize=15, color='#161a1d')
```

```
plt.xlim([25,90])
plt.show()
```

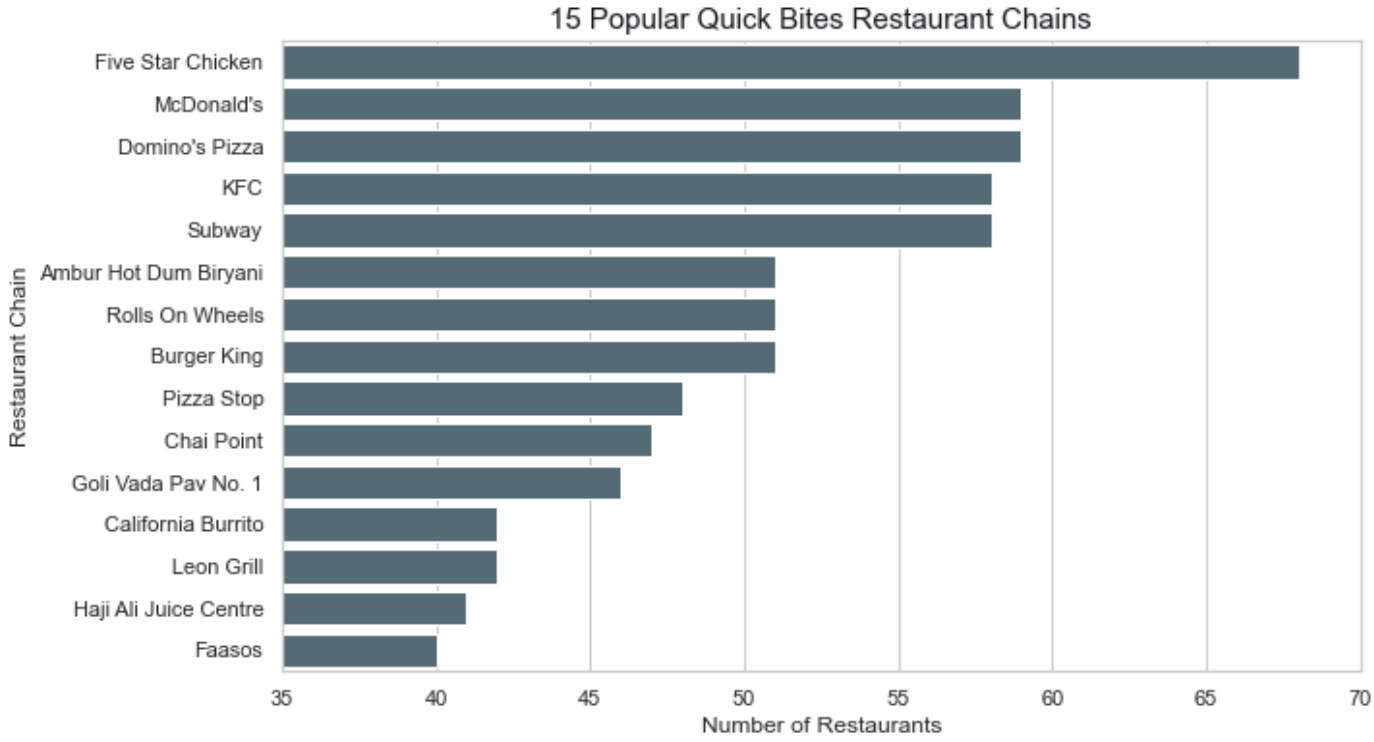


Cafe Coffee Day and Onesta lead the pack, boasting 86 and 85 outlets, respectively. Smally's Resto Cafe, Tata Cha, and Lakeview Milkbar also hold significant presence

15 Popular Quick Bites Restaurant Chains

In [233…

```
quick_bites_chains = df[df['rest_type'].str.contains('Quick Bites', case=False, na=False
popular_chains = quick_bites_chains['name'].value_counts().reset_index()

popular_chains.columns = ['Restaurant Chain', 'Count']
popular_chains = popular_chains.sort_values(by='Count', ascending=False)
top_15_chains = popular_chains.head(15)

plt.figure(figsize=(10, 6))
sns.barplot(x='Count', y='Restaurant Chain', data=top_15_chains,color='#4f6d7a')
plt.xlabel('Number of Restaurants', fontsize=12)
plt.title('15 Popular Quick Bites Restaurant Chains',  fontsize=15, color='#161a1d')
plt.xlim([35,70])
plt.show()
```

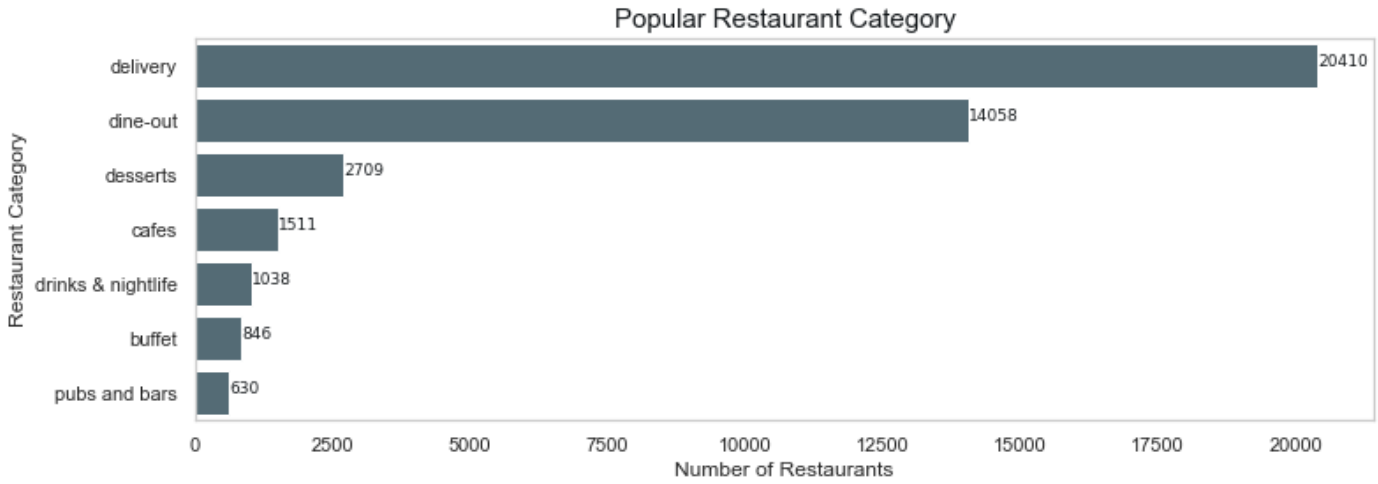


The dining landscape in Bengaluru is dominated by renowned casual dining chains, with "Five Star Chicken," "McDonald's," and "Domino's Pizza" leading the pack in popularity. These top 15 chains, including familiar names like "KFC," "Subway," and "Burger King," showcase the city's diverse culinary offerings.

Popular Restaurants Based on Meal Type

In [234…

```
meal_type_count = df['meal_type'].value_counts()
fig, ax = plt.subplots(figsize=(12, 4))

sns.barplot(x=meal_type_count, y=meal_type_count.index, color='#4f6d7a')

plt.title('Popular Restaurant Category', fontsize=15, color='#161a1d')
plt.xlabel('Number of Restaurants', fontsize=12)
plt.ylabel('Restaurant Category', fontsize=12)

for v, count in enumerate(meal_type_count):
    ax.text(count, v, str(count), fontsize=9, family='DejaVu Sans', color='#161a1d')
plt.grid(False)
plt.show()
```

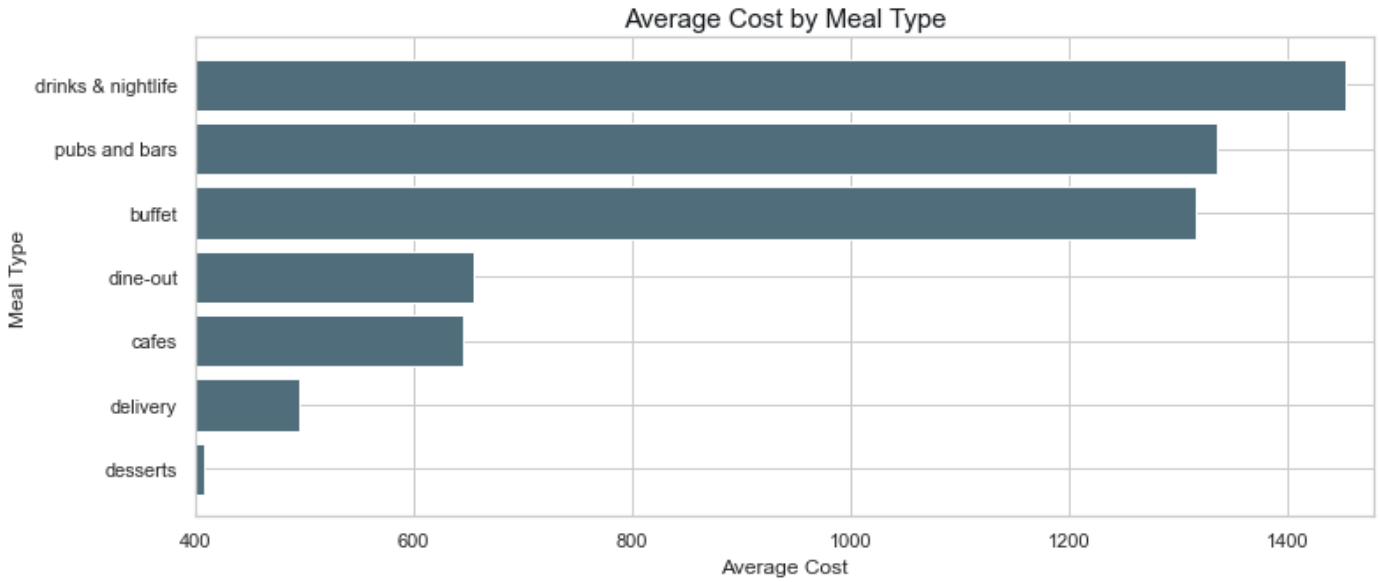


"Delivery" and "dine-out" lead the count, reflecting the city's diverse food culture. The dominance of "delivery" suggests a strong preference for convenient food services. Other categories like "desserts,"

"cafes," and "drinks & nightlife" contribute to the city's vibrant culinary scene.
Opting for delivery meals in Zomato Bengaluru restaurants is driven by the unparalleled convenience they offer. Beyond the time-saving advantage, delivery services often present cost-effective solutions with discounts. The extensive variety of cuisines available for delivery, coupled with the flexibility to order at any time, caters to diverse preferences and busy schedules.

Average Restaurant Costs for Different Meal Types

In [235…
```
avg_cost_by_meal = df.groupby('meal_type')['cost'].mean().sort_values(ascending=True)
avg_cost_by_meal = avg_cost_by_meal.reset_index(name='restaurant_avg_cost')

plt.barh(avg_cost_by_meal['meal_type'], avg_cost_by_meal['restaurant_avg_cost'],
         color='#4f6d7a')
plt.xlabel('Average Cost', fontsize=12)
plt.ylabel('Meal Type', fontsize=12)
plt.title('Average Cost by Meal Type', fontsize=15, color='#161a1d')
plt.xlim([400,1480])
plt.show()
```



Desserts and delivery options, such as cafes, tend to be more budget-friendly, with costs ranging from 409 to 646. Meanwhile, dine-out experiences, buffets, and pubs/bars exhibit higher average costs, peaking at 1454. "Drinks & Nightlife" category, with an average cost of 1454, which signals a trend towards higher spending for those seeking vibrant nocturnal experiences.

Top 10 Location Based on Each Meal Type

Meal type frequency in each location

In [236…
```
rest_type_count = df.groupby('location')['meal_type'].value_counts().unstack(fill_value=
rest_type_count
```

Out[236]:

meal_type location	buffet	cafes	delivery	desserts	dine-out	drinks & nightlife	pubs and bars
BTM	18	71	2432	125	1195	19	13
Banashankari	7	36	347	57	284	13	0
Banaswadi	0	20	227	25	188	6	1
Bannerghatta Road	9	37	639	83	447	9	2

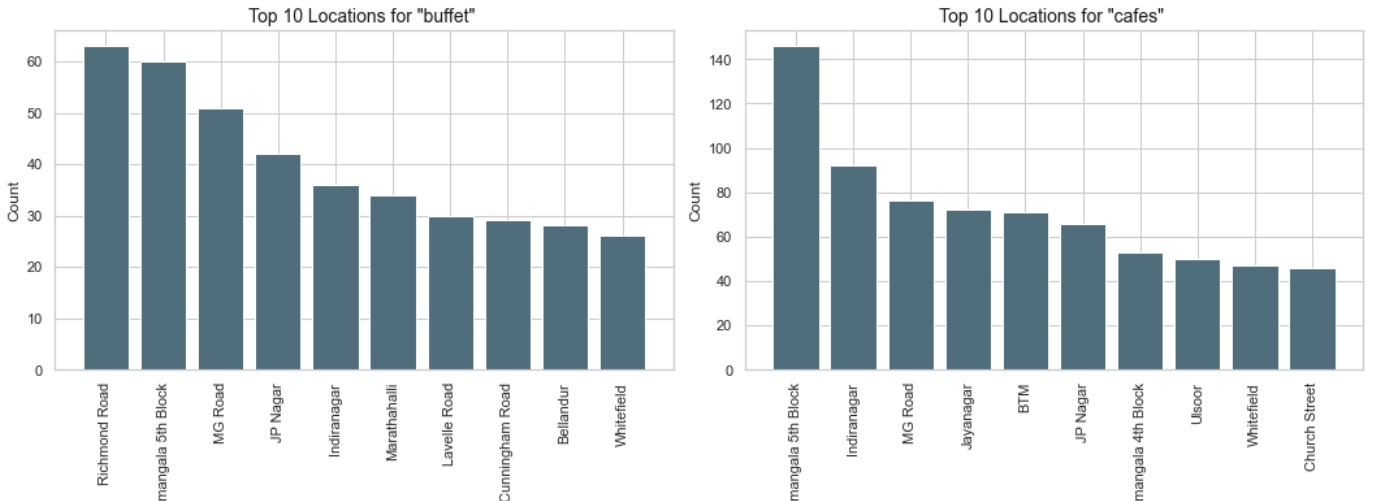
Basavanagudi	7	11	301	52	219	5	0
Basaveshwara Nagar	2	5	59	17	45	2	3
Bellandur	28	31	467	58	380	17	16
Bommanahalli	2	0	90	6	44	1	1
Brigade Road	25	41	427	83	393	57	22
Brookefield	6	17	277	37	202	4	0
CV Raman Nagar	0	1	36	4	19	0	0
Central Bangalore	0	0	8	0	0	0	0
Church Street	19	46	185	29	213	31	20
City Market	0	4	28	2	41	0	0
Commercial Street	0	7	80	60	123	0	0
Cunningham Road	29	28	190	26	178	16	7
Domlur	15	13	218	27	110	12	11
East Bangalore	0	0	24	1	3	0	0
Ejipura	0	0	151	8	96	0	0
Electronic City	21	18	355	46	353	20	19
Frazer Town	1	10	360	49	139	2	2
HBR Layout	0	3	47	8	39	2	0
HSR	16	45	1350	94	458	12	17
Hebbal	0	0	4	1	3	0	0
Hennur	0	3	76	7	28	0	0
Hosur Road	0	0	38	0	31	0	3
ITPL Main Road, Whitefield	0	2	45	7	21	1	2
Indiranagar	36	92	961	124	461	61	60
Infantry Road	22	12	23	6	66	6	4
JP Nagar	42	66	873	124	555	44	6
Jalahalli	0	0	11	4	8	0	0
Jayanagar	24	72	904	136	486	12	0
Jeevan Bhima Nagar	0	4	129	13	101	0	0
KR Puram	0	0	7	0	3	0	0
Kaggadasapura	0	0	74	2	14	0	0
Kalyan Nagar	9	37	289	64	275	18	0
Kammanahalli	2	21	261	21	182	6	0
Kanakapura Road	0	0	14	0	5	0	0
Kengeri	0	2	2	2	2	0	0
Koramangala	0	0	30	0	0	0	0
Koramangala 1st Block	3	21	509	41	262	7	8

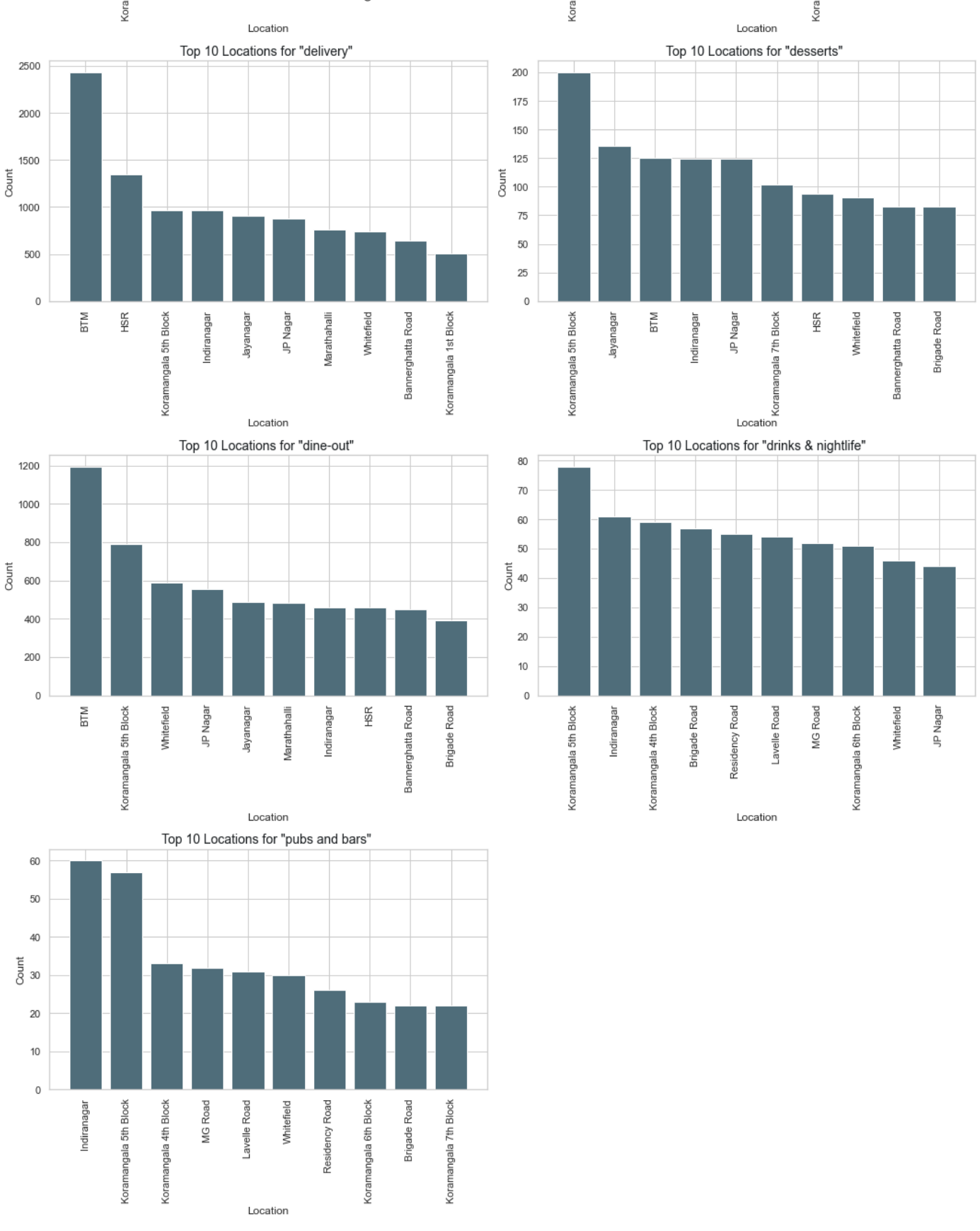
Koramangala 2nd Block	4	8	40	5	25	5	0
Koramangala 3rd Block	19	20	64	13	61	10	3
Koramangala 4th Block	21	53	359	56	260	59	33
Koramangala 5th Block	60	146	965	200	789	78	57
Koramangala 6th Block	18	40	467	70	383	51	23
Koramangala 7th Block	25	44	447	102	388	25	22
Koramangala 8th Block	0	10	140	16	44	0	2
Kumaraswamy Layout	0	1	132	2	23	0	0
Langford Town	0	0	6	0	11	6	4
Lavelle Road	30	22	115	43	181	54	31
MG Road	51	76	213	56	312	52	32
Magadi Road	0	0	10	0	14	0	0
Majestic	3	0	29	14	61	4	0
Malleshwaram	11	29	234	70	263	20	14
Marathahalli	34	26	763	75	485	22	2
Mysore Road	1	0	8	0	8	0	0
Nagarbhavi	0	0	1	0	0	0	0
Nagawara	0	0	143	2	12	0	0
New BEL Road	4	25	265	26	170	8	8
North Bangalore	0	0	5	0	5	0	0
Old Airport Road	12	4	159	29	128	12	9
Old Madras Road	0	0	17	0	5	0	0
Peenya	0	0	0	0	1	0	0
RT Nagar	0	0	54	0	6	0	0
Race Course Road	8	4	45	7	54	11	6
Rajajinagar	10	3	203	40	192	3	10
Rajarajeshwari Nagar	0	0	2	0	0	0	0
Rammurthy Nagar	0	0	26	0	0	0	0
Residency Road	20	26	154	53	267	55	26
Richmond Road	63	16	186	50	260	16	12
Sadashiv Nagar	4	5	20	7	21	1	0
Sahakara Nagar	0	0	39	0	6	0	0
Sanjay Nagar	0	0	31	1	16	0	0
Sankey Road	5	0	1	1	16	2	1
Sarjapur Road	25	14	447	70	256	19	21
Seshadripuram	5	3	54	7	53	11	9
Shanti Nagar	9	18	150	29	149	9	2
Shivajinagar	6	14	63	18	161	7	8

South Bangalore	0	3	72	3	11	0	0
St. Marks Road	5	10	111	10	145	40	22
Thippasandra	0	2	53	17	54	2	2
Ulsoor	16	50	375	54	309	18	21
Uttarahalli	0	0	8	0	1	0	0
Varthur Main Road, Whitefield	0	0	73	1	10	0	0
Vasanth Nagar	4	16	134	12	74	5	0
Vijay Nagar	0	0	42	1	17	1	1
West Bangalore	0	0	3	0	0	0	0
Whitefield	26	47	738	91	590	46	30
Wilson Garden	3	0	109	6	66	2	1
Yelahanka	0	0	2	0	2	0	0
Yeshwantpur	1	0	85	3	21	1	1

In [237…

```
meal_types = df['meal_type'].unique()

num_meal_types = len(meal_types)
num_cols = 2
num_rows = (num_meal_types + num_cols - 1) // num_cols

fig, axs = plt.subplots(num_rows, num_cols, figsize=(15, 6*num_rows))

for i, meal_type in enumerate(meal_types):
    row = i // num_cols
    col = i % num_cols
    rest_type_count = df[df['meal_type'] == meal_type].groupby('location').size().reset_
    top_locations = rest_type_count.nlargest(10, 'count')

    axs[row, col].bar(top_locations['location'], top_locations['count'],color ='#4f6d7a'
    axs[row, col].set_title(f'Top 10 Locations for "{meal_type}" ', fontsize=14, color='
    axs[row, col].set_xlabel('Location', fontsize=12)
    axs[row, col].set_ylabel('Count', fontsize=12)
    axs[row, col].tick_params(axis='x', rotation=90)

for i in range(num_meal_types, num_rows * num_cols):
    axs.flatten()[i].axis('off')

plt.tight_layout()
plt.show()
```



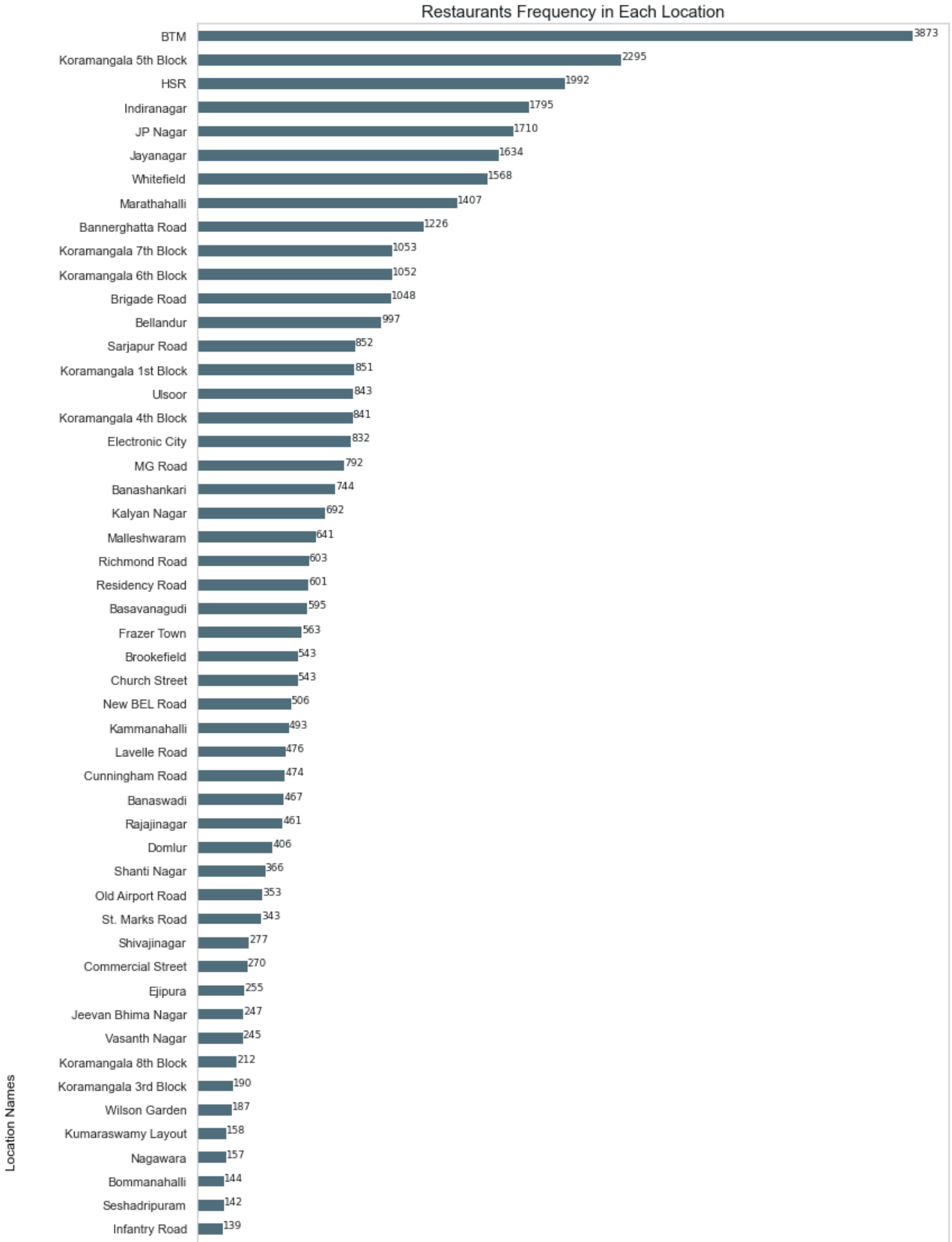


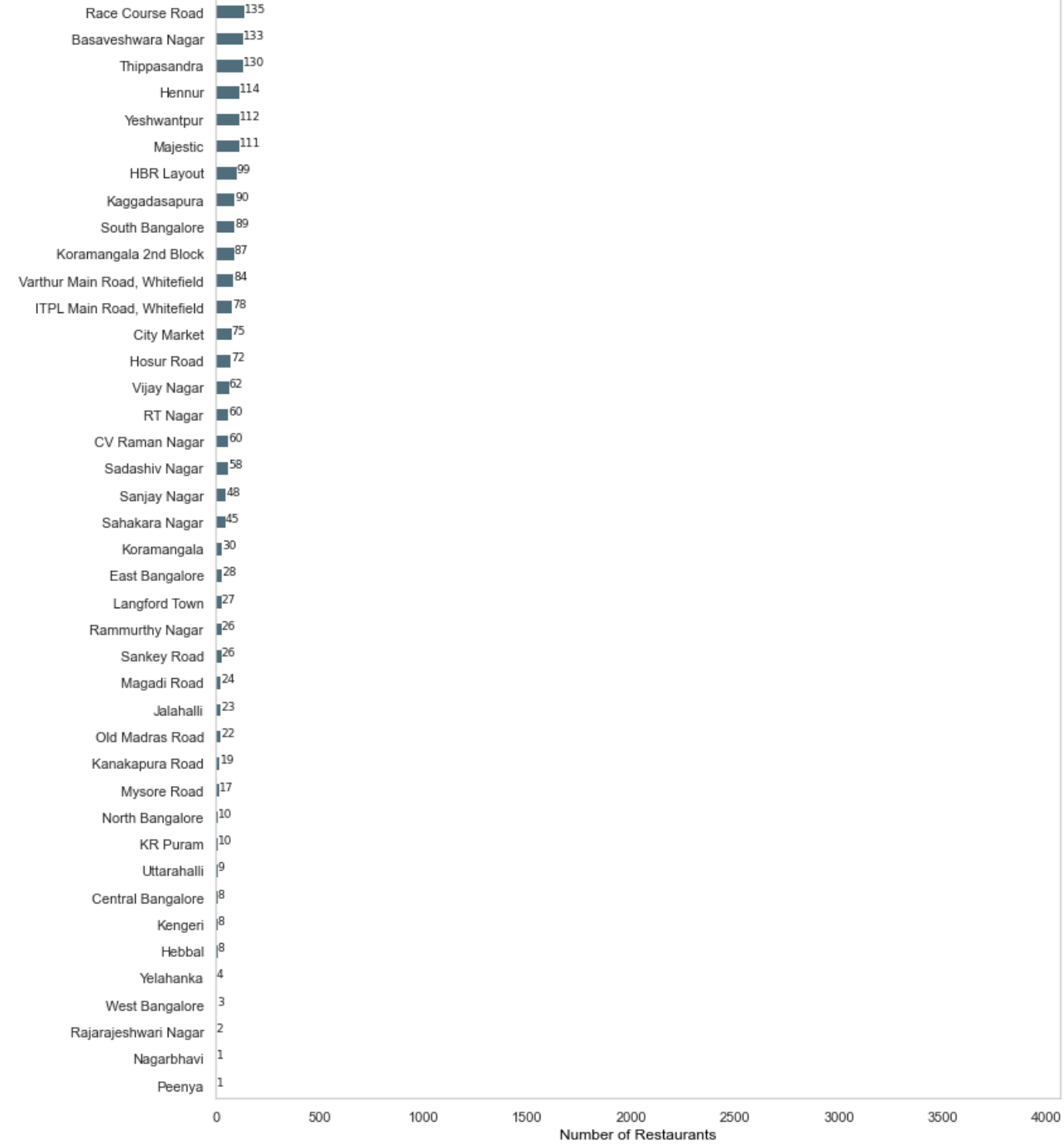
Restaurants Frequency in Each Location

In [257…

```
location_count = df['location'].value_counts().sort_values(ascending=True)

fig, ax = plt.subplots(figsize=(12, 36))
location_count.plot(kind='barh', color='#4f6d7a', ax=ax)

ax.set_ylabel('Location Names', fontsize=12, color='black')
ax.set_xlabel('Number of Restaurants', fontsize=12, color='black')
ax.set_title('Restaurants Frequency in Each Location',
```

```
            fontsize=15, color='#161a1d')

for v, count in enumerate(location_count):
    ax.text(count, v, str(count), fontsize=9,
            family= 'DejaVu Sans', color='#161a1d')

ax.grid(axis='x', linestyle='--', alpha=0.3)
plt.grid(False)
plt.show()
```





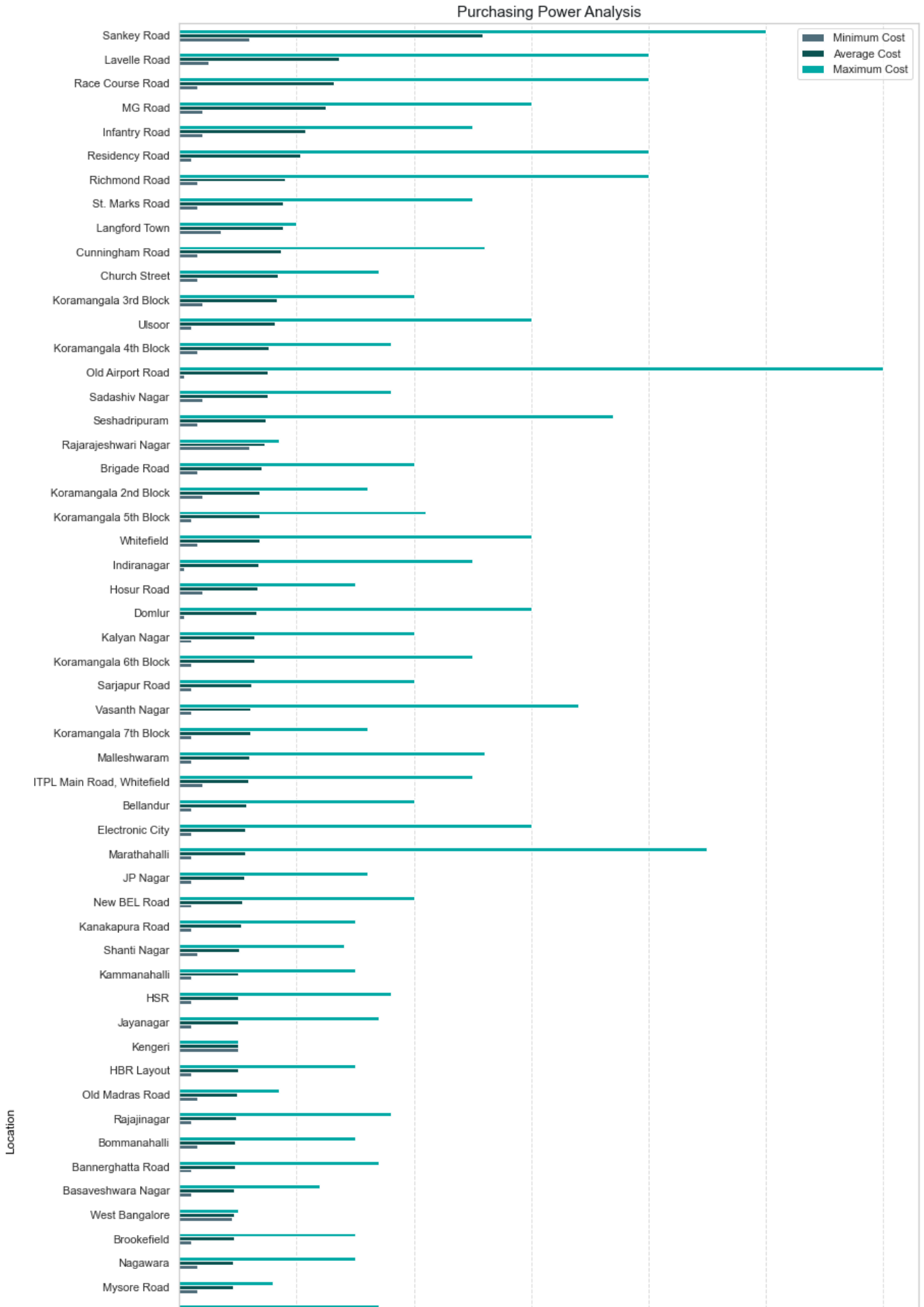
BTM emerges as the hotspot with a substantial count of 3873. Notably, popular areas like MG Road, Electronic City, and Koramangala boast a moderate presence, while Bannerghatta Road and Koramangala 5th Block stand out with higher counts.

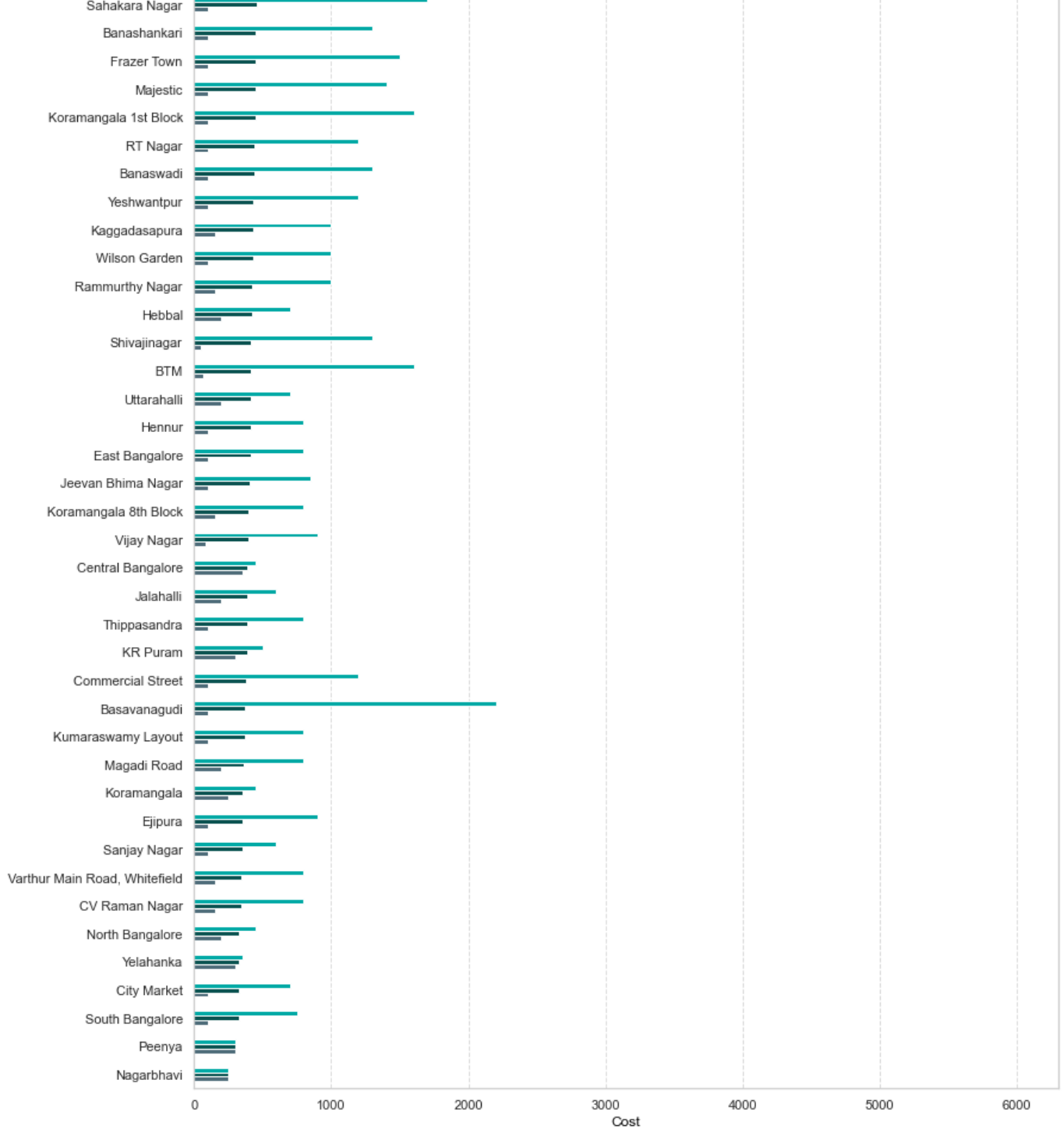
Purchasing Power Analysis

In [239…

```
purchasing_power_by_location = df.groupby('location').agg(minimum=('cost', 'min'),
                                                           avg=('cost', 'mean'),
                                                           maximum=('cost', 'max'))
purchasing_power_by_location = purchasing_power_by_location.sort_values(by='avg')
purchasing_power_by_location.plot(kind='barh', y=['minimum', 'avg', 'maximum'],
                                  figsize=(13, 40),
                                  color=['#4f6d7a', '#0b5351', '#00a9a5'])
plt.title('Purchasing Power Analysis',  fontsize=15, color='#161a1d')
plt.xlabel('Cost', fontsize=12, color='black')
plt.ylabel('Location', fontsize=12, color='black')
```

```
plt.legend(['Minimum Cost', 'Average Cost', 'Maximum Cost'])
plt.grid(False)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```





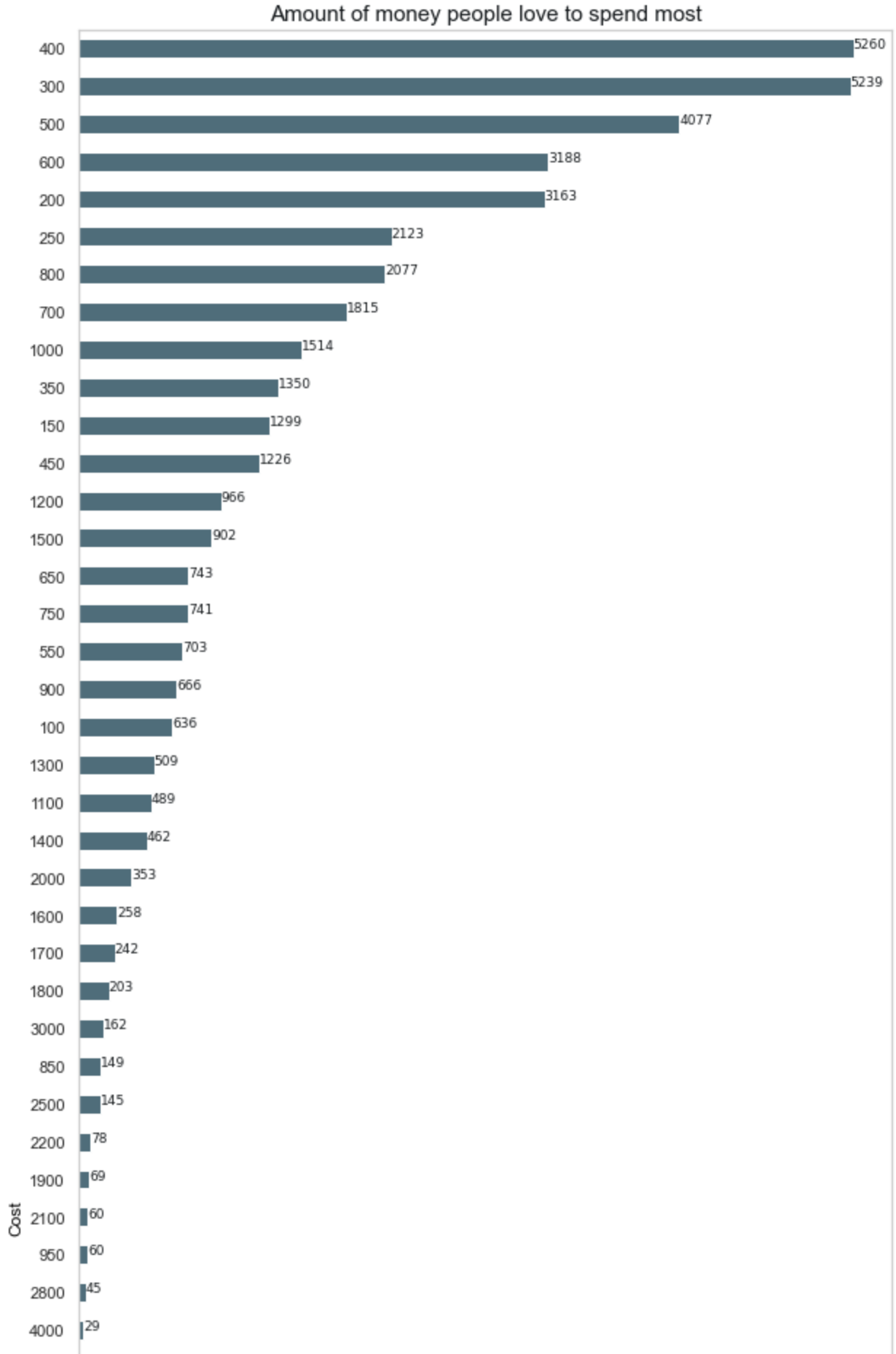
Embark on a journey through Bangalore's diverse landscapes of purchasing power. Koramangala 7th Block boasts a thriving balance with a range of 100 to 1600 Rupees, while Sankey Road stands as a pinnacle, featuring a robust spectrum from 600 to 5000 Rupees. Uncover the city's economic tapestry, from the vibrant buzz of MG Road to the serene pockets of Old Airport Road. Each location weaves a unique narrative in the intricate fabric of Bangalore's cost dynamics, creating a rich mosaic of lifestyle choices.

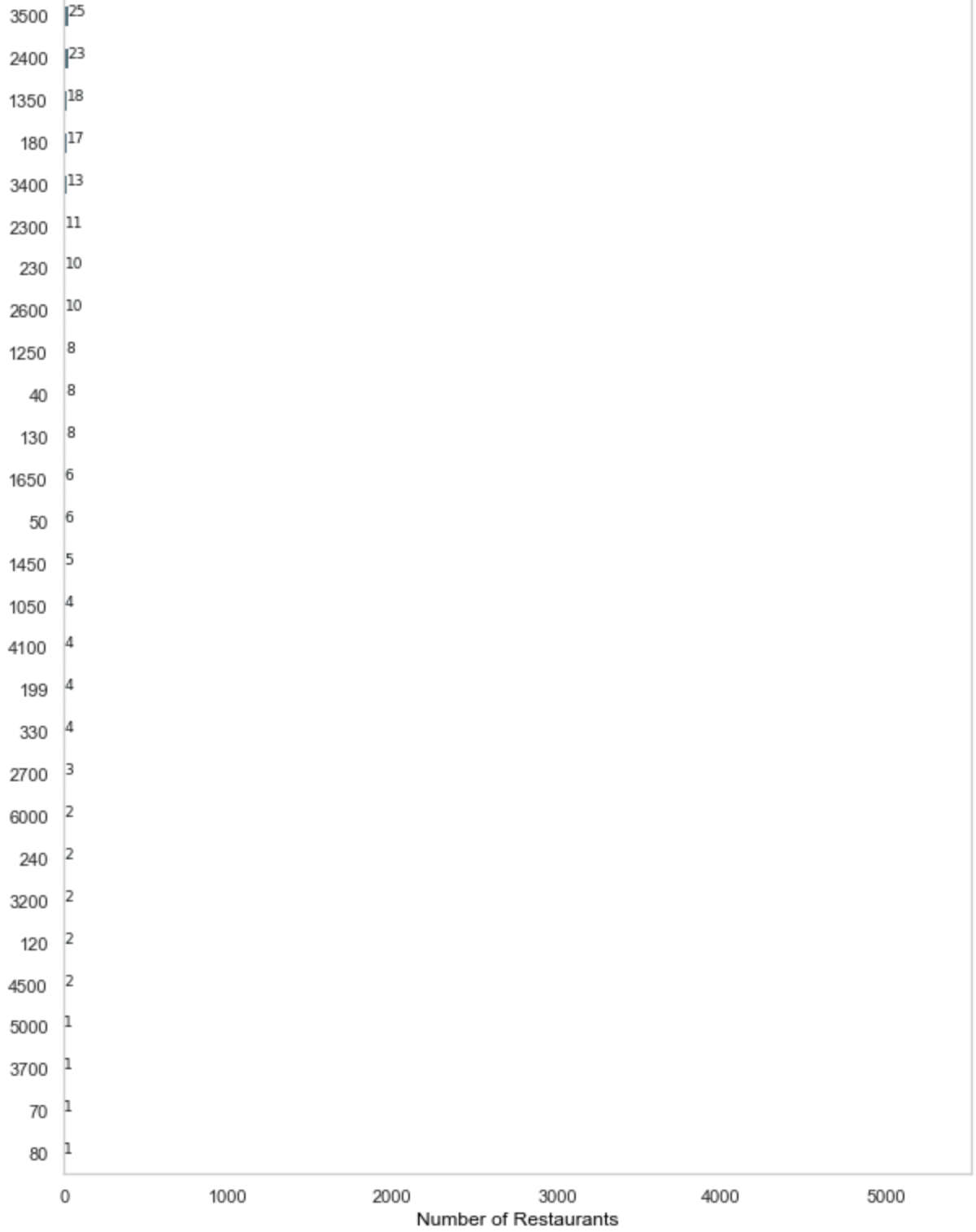
Amount of Money People Love to Spend Most

In [240…

```
cost_count = df['cost'].value_counts().sort_values(ascending=True)

fig, ax = plt.subplots(figsize=(10, 30))
cost_count.plot(kind='barh', color='#4f6d7a', ax=ax)

ax.set_ylabel('Cost', fontsize=12, color='black')
ax.set_xlabel('Number of Restaurants', fontsize=12, color='black')
ax.set_title('Amount of money people love to spend most', fontsize=15, color='#161a1d')
```

```
for v, count in enumerate(cost_count):
    ax.text(count, v, str(count), fontsize=9,family= 'DejaVu Sans', color='#161a1d')

ax.grid(axis='x', linestyle='--', alpha=0.3)
plt.grid(False)
plt.show()
```





This insightful visualization guides diners by showcasing the prevailing cost trends, aiding them in selecting restaurants that align with their budgetary preferences.
Notably, a substantial number of individuals prefer affordable options, with the majority spending between 200 and 600. Interestingly, the chart reveals a spike in choices around 500, suggesting a popular cost range.

In [241…

```
df_pred = df.copy()
df_pred.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 41202 entries, 0 to 51716
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   name             41202 non-null  object
 1   online_order     41202 non-null  bool
```

```
 2   book_table       41202 non-null  bool
 3   rate             41202 non-null  float64
 4   votes            41202 non-null  int64
 5   location         41202 non-null  object
 6   rest_type        41202 non-null  object
 7   cuisines         41202 non-null  object
 8   cost             41202 non-null  int32
 9   meal_type        41202 non-null  object
 10  rest_type_count  41202 non-null  int64
dtypes: bool(2), float64(1), int32(1), int64(2), object(5)
memory usage: 4.1+ MB
```

Data Preprocessing

In [242…
```
df_pred['cuisines_count'] = [len(cui) for cui in df['cuisines'].str.split(',')]
df_pred.drop(['cuisines','meal_type','name'], axis=1, inplace=True)
```

In [243…
```
df_pred['target'] = df_pred['rate'].apply(lambda x: 1 if x >= 3.70 else 0)
df_pred.drop('rate', axis=1, inplace=True)
```

In [244…
```
def grab_col_names(dataframe, cat_th=10, car_th=20):
    cat_cols = [col for col in dataframe.columns if dataframe[col].dtypes == 'O']

    num_but_cat = [col for col in dataframe.columns if dataframe[col].nunique() < cat_th
                   dataframe[col].dtypes != 'O']

    cat_but_car = [col for col in dataframe.columns if dataframe[col].nunique() > car_th
                   dataframe[col].dtypes == 'O']

    cat_cols = cat_cols + num_but_cat
    cat_cols = [col for col in cat_cols if col not in cat_but_car]
    num_cols = [col for col in dataframe.columns if dataframe[col].dtypes != 'O']
    num_cols = [col for col in num_cols if col not in num_but_cat]

    print(f'Observations: {dataframe.shape[0]}')
    print(f'Variables: {dataframe.shape[1]}')
    print(f'cat_cols: {len(cat_cols)}')
    print(f'num_cols: {len(num_cols)}')
    print(f'cat_but_car: {len(cat_but_car)}')
    print(f'num_but_cat: {len(num_but_cat)}')

    return cat_cols, num_cols, cat_but_car
cat_cols, num_cols, cat_but_car = grab_col_names(df_pred)
```

```
Observations: 41202
Variables: 9
cat_cols: 5
num_cols: 2
cat_but_car: 2
num_but_cat: 5
```

In [245…
```
def rare_encoder(dataframe, cat_cols, rare_perc=0.01):
    rare_columns = [col for col in cat_cols if
                    (dataframe[col].value_counts() / len(dataframe) < rare_perc).sum() >

    for col in rare_columns:
        tmp = dataframe[col].value_counts() / len(dataframe)
        rare_labels = tmp[tmp < rare_perc].index
        dataframe[col] = np.where(dataframe[col].isin(rare_labels), 'Rare', dataframe[co
    return dataframe

rare_encoder(df_pred, ['location'], rare_perc=0.03)
rare_encoder(df_pred, ['rest_type'], rare_perc=0.02)
```

Out[245]:

	online_order	book_table	votes	location	rest_type	cost	rest_type_count	cuisines_count	target
0	True	True	775	Rare	casual dining	800	1	3	1
1	True	False	787	Rare	casual dining	800	1	3	1
2	True	False	918	Rare	Rare	800	2	3	1
3	False	False	88	Rare	quick bites	300	1	2	1
4	False	False	166	Rare	casual dining	600	1	2	1
...	...	...	...	...	...	...	...	...	...
51709	False	False	34	Whitefield	casual dining, bar	800	2	2	1
51711	False	False	81	Whitefield	casual dining, bar	800	2	4	0
51712	False	False	27	Whitefield	Rare	1500	1	1	0
51715	False	True	236	Rare	Rare	2500	1	1	1
51716	False	False	13	Rare	Rare	1500	2	3	0

41202 rows × 9 columns

In [246…
```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()

binary_cols = [col for col in df_pred.columns if
               df_pred[col].dtype not in [int, float]
                   and df_pred[col].nunique() == 2 and col != 'target']

def label_encoder(dataframe, binary_col):
    labelencoder = LabelEncoder()
    dataframe[binary_col] = labelencoder.fit_transform(dataframe[binary_col])
    return dataframe

for col in binary_cols:
    df_pred = label_encoder(df_pred, col)
```

In [247…
```
ohe_cols = [col for col in df_pred.columns if 10 >= df_pred[col].nunique() > 2 and col !
            and col != 'rest_type_count']
def one_hot_encoder(dataframe, categorical_cols, drop_first=False):
    dataframe = pd.get_dummies(dataframe, columns=categorical_cols, drop_first=drop_firs
    return dataframe
df_pred = one_hot_encoder(df_pred, ohe_cols)
```

In [248…
```
scaler = RobustScaler()
df_pred[num_cols] = scaler.fit_transform(df_pred[num_cols])
df_pred = df_pred.rename(columns=lambda x: re.sub('[^A-Za-z0-9_]+', '', x))

df_pred.columns = [col.upper() for col in df_pred.columns]
```

Building Model

In [253…
```
X = df_pred.drop(['TARGET'], axis=1)
y = df_pred['TARGET']

metric = []
```

```
meth_auc = []
meth_f1 = []
meth_acc = []

models = [
    ('DecisionTree', DecisionTreeClassifier()),
    ('RandomForest', RandomForestClassifier()),
    ('Adaboost', AdaBoostClassifier()),
    ('GBM', GradientBoostingClassifier()),
    ('XGBoost', XGBClassifier(eval_metric='logloss', use_label_encoder=False)),
    ('LightGBM', LGBMClassifier(force_col_wise=True)),
    ('CatBoost', CatBoostClassifier(verbose=False))
]

for name, model in models:
    cv_results = cross_validate(model, X, y, cv=3, scoring=['roc_auc', 'f1', 'accuracy']

    meth_auc.append(round(np.mean(cv_results['test_roc_auc']), 4))
    meth_f1.append(round(np.mean(cv_results['test_f1']), 4))
    meth_acc.append(round(np.mean(cv_results['test_accuracy']), 4))
    metric.append(name)

result = {
    'Model Name': metric,
    'Accuracy Score': meth_acc,
    'F1 Score': meth_f1,
    'ROC-AUC': meth_auc
}

dataframe = pd.DataFrame(result).sort_values(by='ROC-AUC', ascending=False)
dataframe.reset_index(drop=True, inplace=True)

print('\nAggregate Scores:')
print(f'Mean ROC-AUC: {round(np.mean(meth_auc), 4)}')
print(f'Mean F1 Score: {round(np.mean(meth_f1), 4)}')
print(f'Mean Accuracy: {round(np.mean(meth_acc), 4)}')
print(dataframe)
```

```
[LightGBM] [Info] Number of positive: 15894, number of negative: 11574
[LightGBM] [Info] Total Bins 358
[LightGBM] [Info] Number of data points in the train set: 27468, number of used feature
s: 23
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.578637 -> initscore=0.317180
[LightGBM] [Info] Start training from score 0.317180
[LightGBM] [Info] Number of positive: 15895, number of negative: 11573
[LightGBM] [Info] Total Bins 359
[LightGBM] [Info] Number of data points in the train set: 27468, number of used feature
s: 23
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.578673 -> initscore=0.317330
[LightGBM] [Info] Start training from score 0.317330
[LightGBM] [Info] Number of positive: 15895, number of negative: 11573
[LightGBM] [Info] Total Bins 353
[LightGBM] [Info] Number of data points in the train set: 27468, number of used feature
s: 23
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.578673 -> initscore=0.317330
[LightGBM] [Info] Start training from score 0.317330

Aggregate Scores:
Mean ROC-AUC: 0.9222
Mean F1 Score: 0.8727
Mean Accuracy: 0.8489
     Model Name  Accuracy Score  F1 Score  ROC-AUC
0  RandomForest          0.9148    0.9259   0.9690
1       XGBoost          0.8566    0.8775   0.9398
2      CatBoost          0.8535    0.8759   0.9377
3      LightGBM          0.8238    0.8524   0.9221
```

```
4            GBM          0.8030    0.8360   0.8988
5   DecisionTree          0.8923    0.9059   0.8985
6       Adaboost          0.7985    0.8353   0.8898
```

Random Forest outshines other models with the highest mean ROC-AUC (0.9688), accuracy (0.9154), and F1 score (0.9264), showcasing robust overall performance. While XGBoost and CatBoost deliver competitive results, Random Forest consistently excels across all metrics. The balanced and superior performance of Random Forest makes it the recommended choice for predictive modeling in this scenario

Hyperparameter Optimization

In [254…

```
from sklearn.model_selection import GridSearchCV

param_grids = {
    'RandomForest': {'n_estimators': [50, 100, 200],
                     'max_depth': [None, 10, 20]},
    'XGBoost': {'n_estimators': [50, 100, 200],
                'max_depth': [3, 5, 7], 'learning_rate': [0.01, 0.1, 0.2]},
    'CatBoost': {'iterations': [50, 100, 200],
                 'depth': [4, 6, 8], 'learning_rate': [0.01, 0.1, 0.2]},
    'LightGBM':{'learning_rate': [0.01, 0.1, 0.2],
                'n_estimators': [300, 500, 1500], 'colsample_bytree': [0.5, 0.7, 1]}
}

metric = []
meth_auc = []
meth_f1 = []
meth_acc = []

for name, model in models:
    param_grid = param_grids.get(name)

    if param_grid is not None:
        # Use GridSearchCV for hyperparameter tuning
        grid_search = GridSearchCV(model, param_grid, cv=3,
                                   scoring=['roc_auc', 'f1', 'accuracy'], refit='roc_auc
        grid_search.fit(X, y)
        meth_auc.append(round(grid_search.best_score_, 4))
        meth_f1.append(round(grid_search.cv_results_['mean_test_f1'][grid_search.best_in
        meth_acc.append(round(grid_search.cv_results_['mean_test_accuracy'][grid_search.
        metric.append(name)

result = {
    'Model Name': metric,
    'Accuracy Score': meth_acc,
    'F1 Score': meth_f1,
    'ROC-AUC': meth_auc
}

dataframe = pd.DataFrame(result).sort_values(by='ROC-AUC', ascending=False)
dataframe.reset_index(drop=True, inplace=True)
print(dataframe)
```

```
[LightGBM] [Info] Number of positive: 15894, number of negative: 11574
[LightGBM] [Info] Total Bins 358
[LightGBM] [Info] Number of data points in the train set: 27468, number of used feature
s: 23
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.578637 -> initscore=0.317180
[LightGBM] [Info] Start training from score 0.317180
[LightGBM] [Info] Number of positive: 15895, number of negative: 11573
[LightGBM] [Info] Total Bins 359
[LightGBM] [Info] Number of data points in the train set: 27468, number of used feature
s: 23
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.578673 -> initscore=0.317330
```

```
[LightGBM] [Info] Total Bins 358
[LightGBM] [Info] Number of data points in the train set: 27468, number of used feature
s: 23
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.578637 -> initscore=0.317180
[LightGBM] [Info] Start training from score 0.317180
[LightGBM] [Info] Number of positive: 15895, number of negative: 11573
[LightGBM] [Info] Total Bins 359
[LightGBM] [Info] Number of data points in the train set: 27468, number of used feature
s: 23
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.578673 -> initscore=0.317330
[LightGBM] [Info] Start training from score 0.317330
[LightGBM] [Info] Number of positive: 15895, number of negative: 11573
[LightGBM] [Info] Total Bins 353
[LightGBM] [Info] Number of data points in the train set: 27468, number of used feature
s: 23
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.578673 -> initscore=0.317330
[LightGBM] [Info] Start training from score 0.317330
[LightGBM] [Info] Number of positive: 23842, number of negative: 17360
[LightGBM] [Info] Total Bins 360
[LightGBM] [Info] Number of data points in the train set: 41202, number of used feature
s: 23
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.578661 -> initscore=0.317280
[LightGBM] [Info] Start training from score 0.317280
     Model Name  Accuracy Score  F1 Score  ROC-AUC
0  RandomForest          0.9118    0.9242   0.9708
1      LightGBM          0.9031    0.9159   0.9645
2       XGBoost          0.8756    0.8927   0.9521
3      CatBoost          0.8772    0.8947   0.9509
```

In the initial model evaluations, RandomForest exhibited superior performance with the highest ROC-AUC (96.90%), F1 score (92.61%), and accuracy (91.51%) among the models, closely followed by XGBoost and CatBoost. Post-hyper-tuning, RandomForest maintained a strong ROC-AUC (97.08%), and while there was a marginal decrease in accuracy (91.04%) and F1 score (92.31%), it still outperformed other models.

Before hyperparameter tuning, LightGBM showed an accuracy of 82.38%, F1 score of 85.24%, and ROC-AUC of 92.21%. After tuning, significant improvement is observed with accuracy at 90.31%, F1 score 91.59%, and ROC-AUC 96.45%, demonstrating the effectiveness of optimization in enhancing model performance.

Voting Classifier

In [255…

```
catboost_model = CatBoostClassifier(iterations=100, depth=5,
                                    learning_rate=0.1, verbose=False)
random_forest_model = RandomForestClassifier(n_estimators=100)
lightgbm_model = GradientBoostingClassifier(n_estimators=100)

best_models = {'CatBoost': catboost_model,
               'RF': random_forest_model,
               'LightGBM': lightgbm_model}

voting_clf = VotingClassifier(
    estimators=[('CatBoost', best_models['CatBoost']),
                ('RF', best_models['RF']),
                ('LightGBM', best_models['LightGBM'])],voting='soft')

voting_clf.fit(X, y)

cv_results = cross_validate(voting_clf, X, y, cv=3,
                            scoring=['accuracy', 'f1', 'roc_auc'])

print('\nVotingClassifier:')
print('Mean Accuracy:', cv_results['test_accuracy'].mean())
print('Mean F1 Score:', cv_results['test_f1'].mean())
print('Mean ROC AUC:', cv_results['test_roc_auc'].mean())
```

```
VotingClassifier:
Mean Accuracy: 0.8886461822241639
Mean F1 Score: 0.9053457503915782
Mean ROC AUC: 0.9580924211237432
```

Summery and Recommendation:

1. **Prime Locations:** Choose bustling areas like Koramangala 7th Block for a thriving balance of affordability and diversity, attracting a range of customers.
2. **Diverse Cuisine:** Offer a diverse menu with 7 cuisines, as it correlates positively with higher customer satisfaction and ratings.
3. **Budget-Friendly Options:** Focus on quick bites, delivery, and cafes to cater to the prevalent demand for affordable and convenient food services.
4. **Online Ordering:** Prioritize online order services to boost customer engagement, votes, and overall restaurant visibility.
5. **Casual Dining Charm:** Embrace the popularity of casual dining chains like "Onesta" for widespread recognition and customer loyalty.
6. **Cafe Culture:** Capitalize on the café trend; "Cafe Coffee Day" and "Onesta" lead, indicating a thriving coffee and casual dining culture.
7. **Strategic Chain Presence:** Open multiple outlets to establish a strong presence; chains like "Onesta" and "Cafe Coffee Day" lead in popularity.
8. **Culinary Specialization:** Consider specializing in North Indian cuisine, which dominates the scene, followed closely by Chinese and South Indian options.
9. **Striking Ambiance:** Create an inviting ambiance for dine-out experiences, as restaurants with higher ratings tend to have slightly higher dining costs.
10. **Customer-Focused Pricing:** Set prices strategically between 300-1000 for a sweet spot; this range attracts a significant customer base without compromising on quality.
11. **Signature Dishes:** Introduce signature dishes to stand out; "Drinks & Nightlife" venues with unique offerings tend to command higher average costs and ratings.
12. **Innovative Desserts:** Enhance the dessert experience; while budget-friendly, desserts can elevate ratings and contribute to a positive overall dining experience.
13. **Table Booking Services:** If feasible, offer table booking services for an added advantage, as it shows a moderate positive correlation with restaurant ratings.
14. **Prime Real Estate:** Focus on culinary hotspots like Lavelle Road, Church Street, and Residency Road; these areas boast higher ratings, attracting discerning customers.
15. **Leverage Local Favorites:** Identify and incorporate popular local cuisines; understanding the preference for specific cuisines in each location can attract more customers.
16. **Interactive Online Presence:** Leverage a robust online presence; restaurants with online orders tend to receive higher votes, emphasizing the importance of visibility.

17. **Delivery Dynamics:** Acknowledge the popularity of delivery services; Quick Bites and Delivery options are prevalent, offering a significant market for convenient food solutions.

18. **Pricing Strategy:** Be mindful of pricing strategies; higher ratings are associated with spending above 1500 offline, signaling a potential market for premium dining experiences.

19. **Customer Engagement:** Prioritize customer engagement; restaurants with online orders tend to garner higher votes, indicating an engaged and satisfied customer base.

20. **Inclusive Meal Types:** Offer a well-rounded experience; combining delivery, dine-out, desserts, and cafes caters to diverse preferences, ensuring a broader customer base.

Opening a new restaurant with these considerations can significantly enhance chances of success, catering to the dynamic preferences and behaviors of the Bengaluru dining community.