```
import pandas as pd
import plotly.express as px

# load the data set
df = pd.read_csv('zomato.csv')

# 1. Preview the first few rows:
df.head(1) #  # Displays the first 5 rows of the dataset
```

```
   Unnamed: 0.1  Unnamed: 0 restaurant name restaurant type  rate (out
of 5)  \
0             0           0    #FeelTheROLL     Quick Bites
3.4

   num of ratings  avg cost (two people) online_order table booking  \
0               7                  200.0           No             No

  cuisines type       area local address
0     Fast Food  Bellandur     Bellandur
```

Dropping Unnecessary Columns

```
df.drop(columns=['Unnamed: 0.1','Unnamed: 0'], inplace = True)
```

Introduction

The Zomato dataset contains information about various restaurants, including their cuisines, cost, ratings, and online ordering preferences. This analysis aims to extract meaningful insights to help businesses make informed decisions.

Project Objective

This project aims to analyze Zomato restaurant data to uncover trends related to cuisines, cost factors, and customer preferences, providing valuable insights for business strategies.

Data analysis

- `restaurant name`: Name of the restaurant
- `restaurant type`: Type of the restaurant (e.g., Casual Dining, Cafe)
- `rate (out of 5)`: Average rating given by customers
- `num of ratings`: Number of customer ratings
- `avg cost (two people)`: Average cost for two people
- `online_order`: Whether online ordering is available (Yes/No)
- `table booking`: Whether table booking is available (Yes/No)
- `cuisines type`: Types of cuisines offered
- `area`: Location of the restaurant
- `local address`: Specific address of the restaurant

3. General information about the Dataset

```
df.info() # Provides details like column names, non-null count, and datatype
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7105 entries, 0 to 7104
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   restaurant name        7105 non-null   object 
 1   restaurant type        7105 non-null   object 
 2   rate (out of 5)        7105 non-null   float64
 3   num of ratings         7105 non-null   int64  
 4   avg cost (two people)  7105 non-null   float64
 5   online_order           7105 non-null   object 
 6   table booking          7105 non-null   object 
 7   cuisines type          7105 non-null   object 
 8   area                   7105 non-null   object 
 9   local address          7105 non-null   object 
dtypes: float64(2), int64(1), object(7)
memory usage: 555.2+ KB
```

```
df .shape   # Provides details like column names, non-null count, and datatype
```

```
(7105, 10)
```

Missing Values in the Dataset

```
df.isnull().sum()
```

```
restaurant name           0
restaurant type           0
rate (out of 5)          68
num of ratings            0
avg cost (two people)    57
online_order              0
table booking             0
cuisines type             0
area                      0
local address             0
dtype: int64
```

Statistical Summary of the Dataset

```
df.describe()
```

```
       rate (out of 5)  num of ratings  avg cost (two people)
count      7105.000000     7105.000000            7105.000000
```

```
mean          3.514117      188.921042             539.161013
std           0.461028      592.171049             461.211329
min           1.800000        1.000000              40.000000
25%           3.200000       16.000000             300.000000
50%           3.500000       40.000000             400.000000
75%           3.800000      128.000000             600.000000
max           4.900000    16345.000000            6000.000000
```

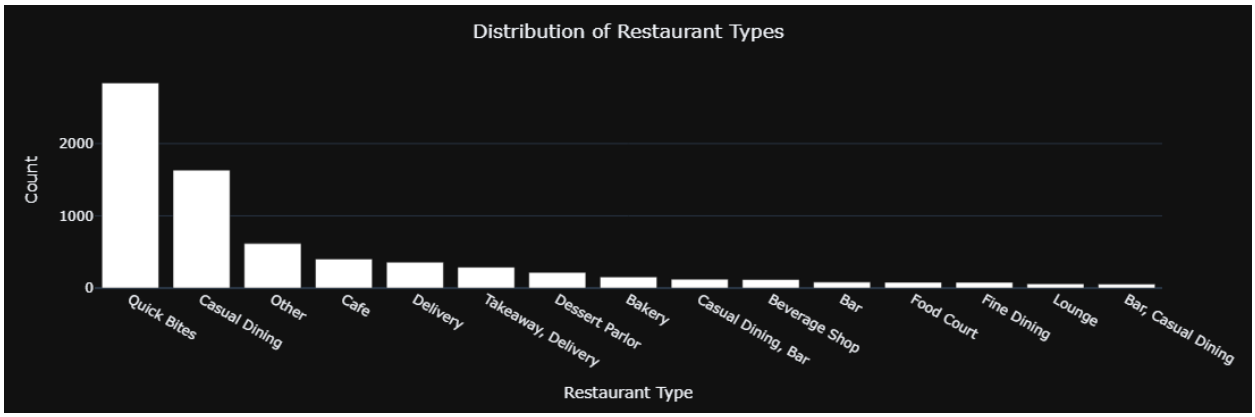
I have only 2 column contaion missing values named : rate (out of 5) and avg cost (two people)

The median was chosen to fill missing values because it represents the middle of the data and is not impacted by outliers.

```
# Fill missing values with median for all columns
df['rate (out of 5)'] = df['rate (out of 5)'].fillna(df['rate (out of 5)'].median())
df['avg cost (two people)'] = df['avg cost (two people)'].fillna(df['avg cost (two people)'].median())

EXPLORATORY DATA ANALYSIS 
```

Create a bar chart for 'restaurant type' column

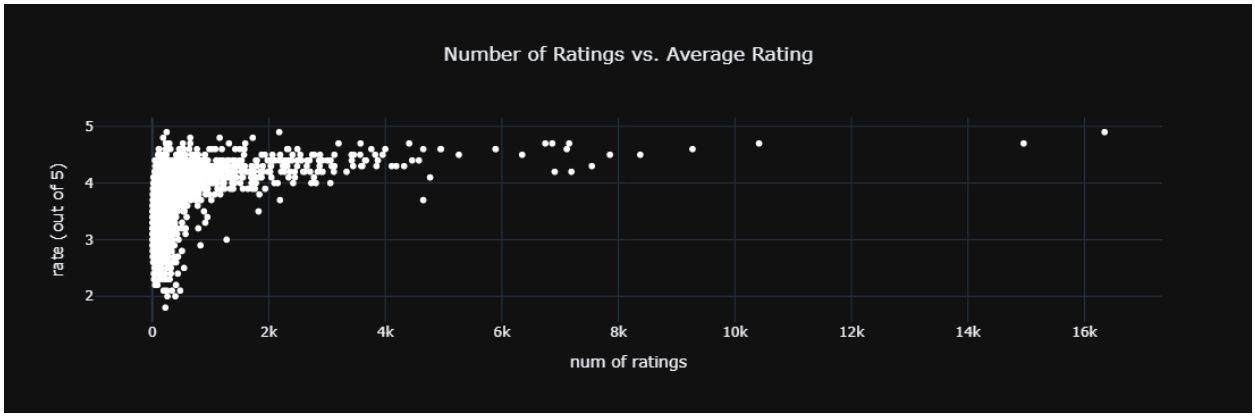
```
# Create a bar chart for 'restaurant type' column
restuarant_type_count = df['restaurant type'].value_counts() # calculates the count of each unique restaurant type
fig =px.bar(
    x = restuarant_type_count.index , # x-axis: Restaurant types (categories)
    y = restuarant_type_count.values, # y-axis: Count of each type (values)
    template='plotly_dark',  # Dark theme
    color_discrete_sequence=['white']  # Set bar color to white
)

# update the layout
fig.update_layout(
      title='Distribution of Restaurant Types',  # Title of the chart
      title_x = 0.5, # Align the title to the center
      xaxis_title='Restaurant Type',  # Label for x-axis
      yaxis_title='Count', # Label for y-axis
)
fig.show()
```



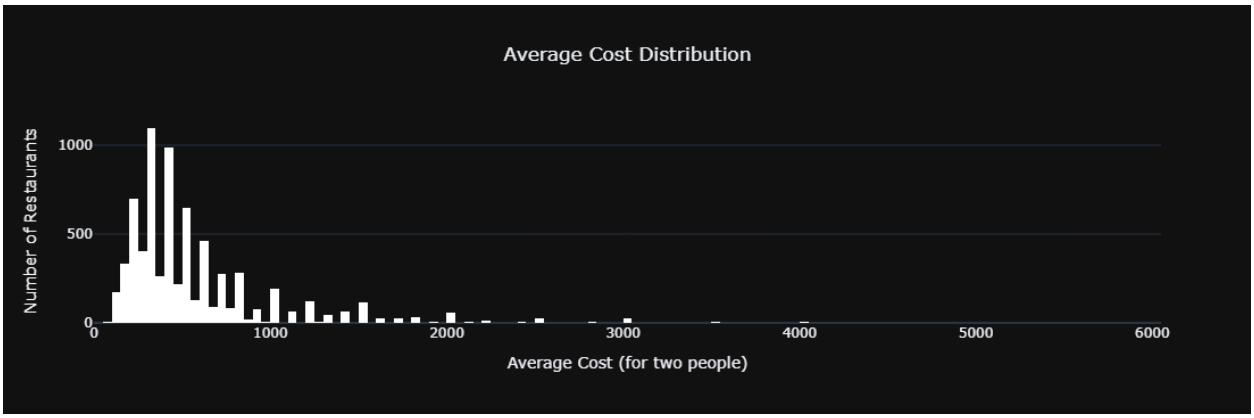

```
rating_count = df['rate (out of 5)'].value_counts().sort_index()
fig = px.bar(
    x = rating_count.index,
    y = rating_count.values,
    template= 'plotly_dark',
    color_discrete_sequence=['white']
)
fig.update_layout(
    title = 'Restaurant Ratings Distribution',
    title_x = 0.5,
    xaxis_title = 'Rating (out of 5)',
    yaxis_title = 'Number of Restaurants'

)
fig.show()
```


```
# Create a scatter plot to visualize the relationship between number
of ratings and average rating
fig = px.scatter(
    df,
    x = 'num of ratings', # Number of ratings received
    y = 'rate (out of 5)', # Average rating out of 5
    title ='Number of Ratings vs. Average Rating',
```

```
    template='plotly_dark',
    color_discrete_sequence=['white']
)
fig.update_layout(
    title_x = 0.5
)
fig.show()
```



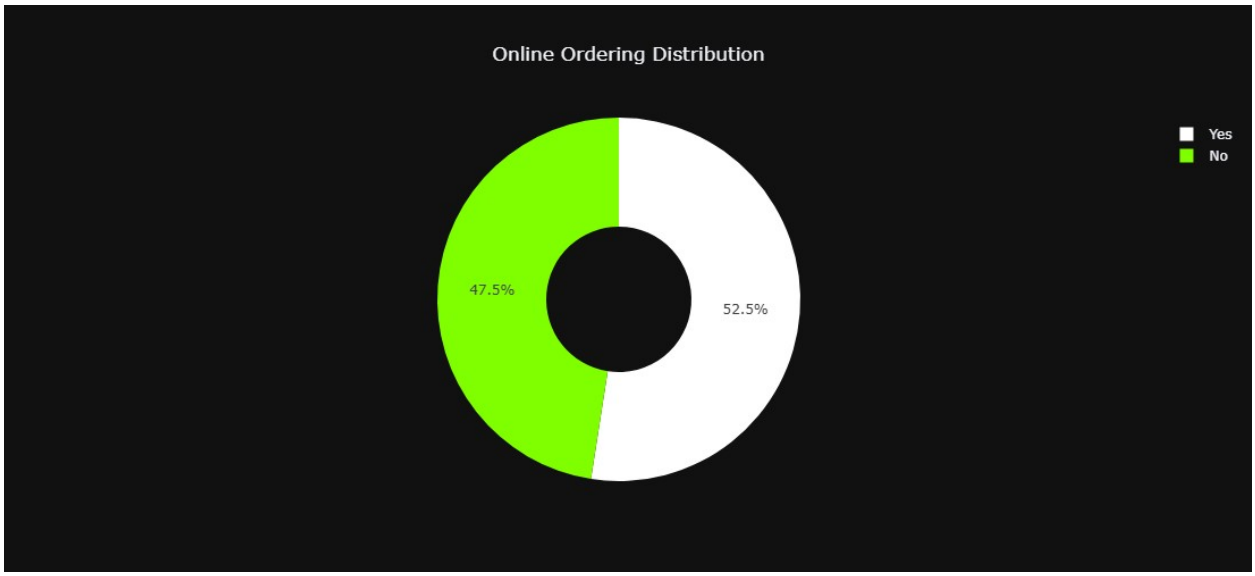

```
# Create a histogram to visualize the distribution of average cost for two people

fig = px.histogram(
    df,
    x = 'avg cost (two people)',
    title = 'Average Cost Distribution',
    template='plotly_dark',
    color_discrete_sequence=['white']
)
fig.update_layout(
    title_x = 0.5,
    xaxis_title='Average Cost (for two people)',
    yaxis_title='Number of Restaurants'
)
fig.show()
```




```
df.head(1)
```

```
  restaurant name restaurant type  rate (out of 5)  num of ratings  \
0    #FeelTheROLL     Quick Bites              3.4               7

   avg cost (two people) online_order table booking cuisines type
area  \
0                  200.0           No             No     Fast Food
Bellandur

  local address
0     Bellandur
```

```
online_order_counts = df['online_order'].value_counts()  # Count
online order

# Create a pie chart to show the distribution of online ordering
fig = px.pie(
    names=online_order_counts.index,  # Labels for pie chart (Yes/No)
    values=online_order_counts.values,  # Corresponding counts
    title='Online Ordering Distribution',  # Chart title
    template='plotly_dark',  #
    color_discrete_sequence=['white','#7FFF00'],
    hole=0.4, # make it a donut chart
    # pull=[0.1, 0]
)

fig.update_layout(
    title_x=0.5,
    width=500,  # Chart width
    height=500  # Chart height
)

fig.show()
```

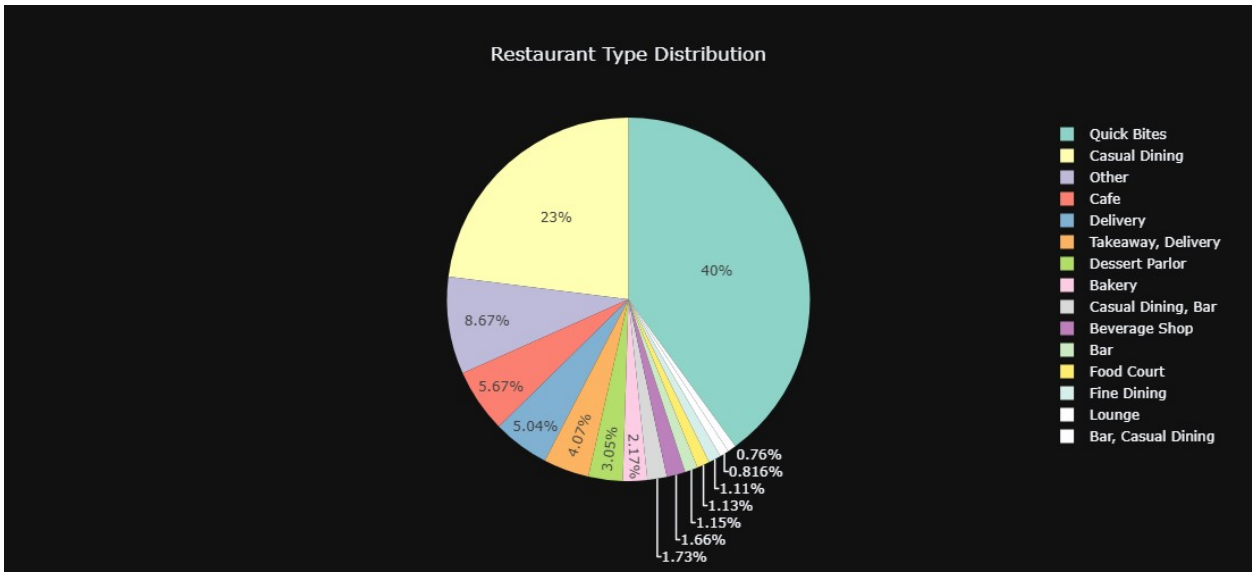



```
# Checking unique values in categorical columns
categorical_columns = ['restaurant type', 'cuisines type',
'online_order', 'table booking']
for col in categorical_columns:
    print(f"{col}: {df[col].nunique()} unique values")
```

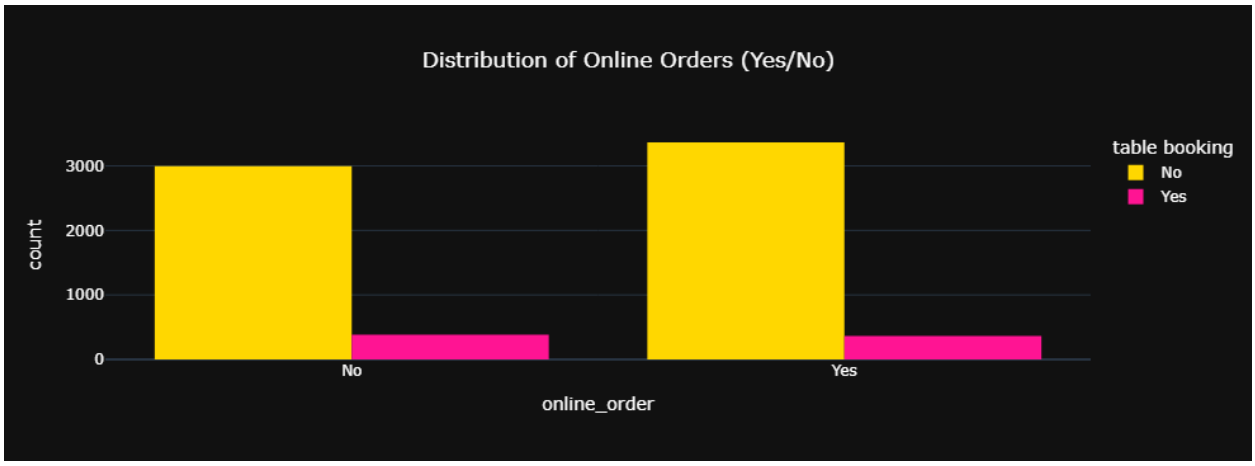
```
restaurant type: 15 unique values
cuisines type: 2175 unique values
online_order: 2 unique values
table booking: 2 unique values
```

```
restaurant_count = df['restaurant type'].value_counts() #  Count
restaurant types

# Replace types that appear 50 times or less with 'Other'
df.loc[df['restaurant type'].isin(restaurant_count[restaurant_count <=
50].index), 'restaurant type'] = 'Other'

#create a pie chart
fig = px.pie(
    df,
    names = 'restaurant type',
    title='Restaurant Type Distribution',
    template='plotly_dark',
    color_discrete_sequence=px.colors.qualitative.Set3  # Using a
better color palette

)
fig.update_layout(
    title_x=0.5,
    # font=dict(color='white'),
    width=1100,
    height=500
```

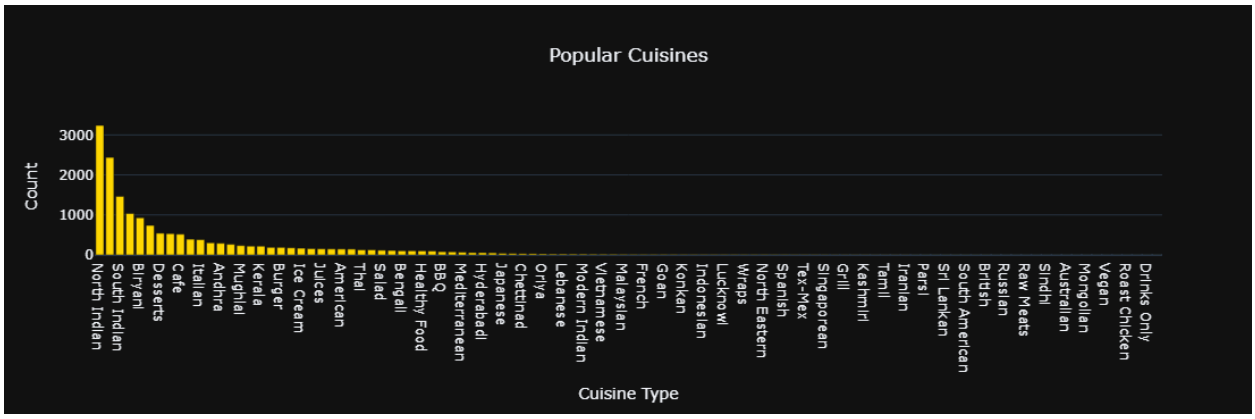
```
)
fig.show()
```




```
# Create histogram for online_order distribution
fig = px.histogram(
    df,
    x='online_order',
    color='table booking', # Color bars by table booking status
    barmode='group', # Group bars by table booking
    title='Distribution of Online Orders (Yes/No)',
    template='plotly_dark',
    color_discrete_sequence=[ '#FFD700', '#FF1493']
)

# Update layout with black background and white text
fig.update_layout(
    title_x = 0.5,
    font=dict(color='white')  # White text for visibility
)

# Show the plot
fig.show()
```

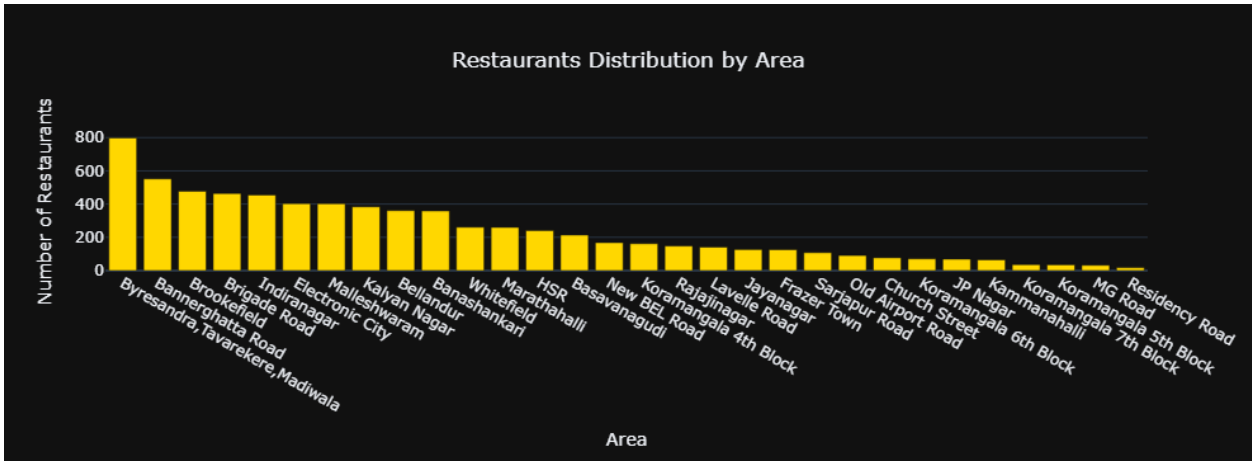



```
# Count the number of times each cuisine appears
cuisine_count = df['cuisines type'].str.split(',
').explode().value_counts()

#Create a bar chart to show the most popular cuisines
fig = px.bar(
    x=cuisine_count.index,
    y=cuisine_count.values,
    title = 'Popular Cuisines',
    template='plotly_dark',
    color_discrete_sequence=['#FFD700']  # Gold color bars

)
fig.update_layout(
    title_x = 0.5,
    xaxis_title = 'Cuisine Type',
    yaxis_title = 'Count'
)
fig.show()
```




```
area_count = df['area'].value_counts()
```

```
fig = px.bar(
    df,
    x=area_count.index,
    y=area_count.values,
    title='Restaurants Distribution by Area',
    template='plotly_dark',
    color_discrete_sequence=['#FFD700']
)
fig.update_layout(
    title_x = 0.5,
    yaxis_title='Number of Restaurants',  # Label for y-axis
    xaxis_title='Area',  # Label for x-axis
)
fig.show()
```

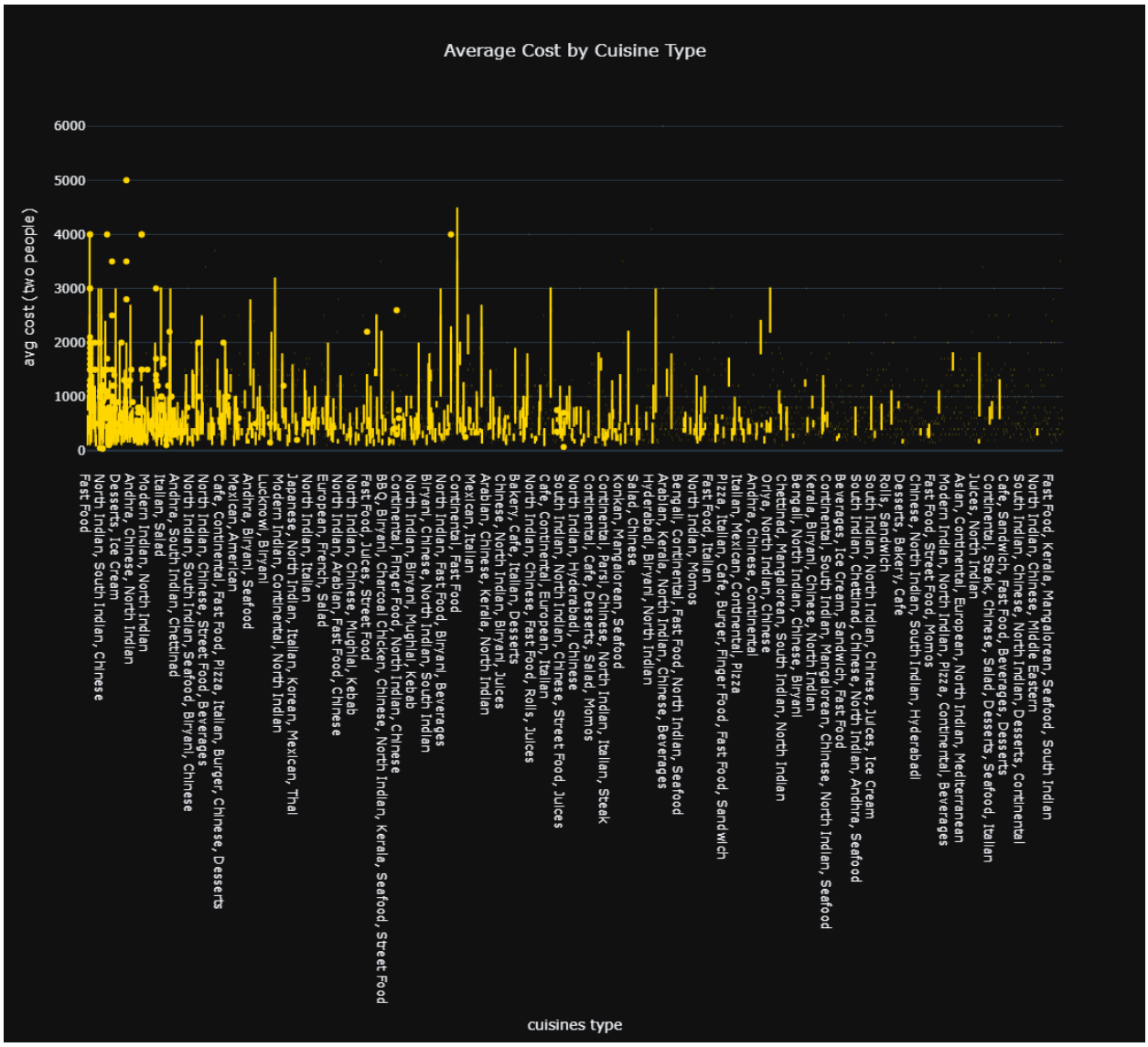



```
df.head(1)
```

```
  restaurant name restaurant type  rate (out of 5)  num of ratings  \
0    #FeelTheROLL     Quick Bites              3.4               7

   avg cost (two people) online_order table booking cuisines type
area  \
0                  200.0           No             No     Fast Food
Bellandur

  local address
0     Bellandur
```

```
fig = px.box(
    df,
    x='cuisines type',
    y='avg cost (two people)',
    title='Average Cost by Cuisine Type',
    template='plotly_dark',
    color_discrete_sequence=['#FFD700']
```

```
)

fig.update_layout(
    title_x=0.5,
    width=1150,
    height=1000
)

fig.show()
```




```
df_numeric = df.select_dtypes(include=['float64', 'int64']) # for choosig the numerical column from the data set

# compute the correlation
df_corr = df_numeric.corr()
```

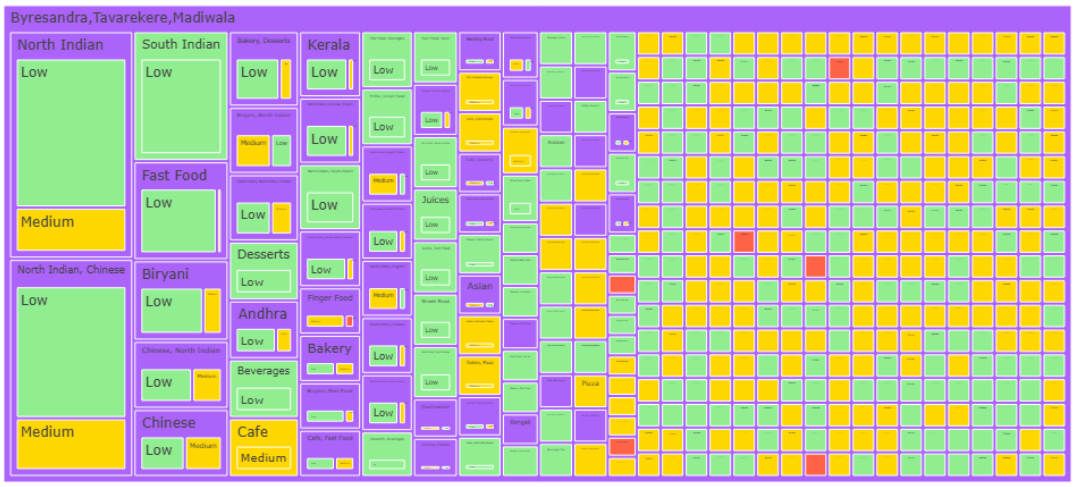
```
#create heatmap
fig = px.imshow(
    df_corr,   # Directly compute correlation matrix from the
DataFrame
    text_auto = True, # Show the correlation values in the heatmap
    color_continuous_scale='Inferno',
    template='plotly_dark',
    title="Correlation Heatmap",
    labels=dict(color="Correlation"),

)
# Update layout for better visualization
fig.update_layout(
    title_x=0.5,
    width=800,
    height=600
)

fig.show()
```

Categorizing Restaurants by Cost Levels

```
df['cost_category'] = df['avg cost (two people)'].apply(lambda x:
'Low' if x < 500 else 'Medium' if x <=1500 else 'High')

df.head(1)
```

```
  restaurant name restaurant type  rate (out of 5)  num of ratings  \
0    #FeelTheROLL     Quick Bites              3.4               7

   avg cost (two people) online_order table booking cuisines type
area  \
0                  200.0           No            No     Fast Food
Bellandur

  local address cost_category
0     Bellandur           Low
```

```python
# Assuming 'area', 'cuisine', and 'cost_category' columns exist in
your DataFrame

# Create the treemap
fig = px.treemap(
    df,
    path=['area', 'cuisines type', 'cost_category'],
    title='Treemap of Cost Categories',
    color='cost_category',  # Color by cost category
    color_discrete_map={
        'Low': 'lightgreen',
        'Medium': 'gold',
        'High': 'tomato'}
)

# Update layout for better aesthetics
fig.update_layout(
    title_x=0.5,  # Center the title
    width=1000,
    height=600,
    font=dict(color='white')  # Set font color to white for better
contrast
)
fig.show()
```



About this Treemap

-"Restaurant Cost Categories by Area and Cuisine Type"

-"Cost Category Breakdown Across Areas and Cuisines"

-"Hierarchical View of Restaurant Costs by Area and Cuisine"

Feature Engineering (New Columns)

Popularity Score Calculation

To better understand restaurant popularity, i created a new column named **'popularity_score'**, which combines customer ratings and the number of reviews.

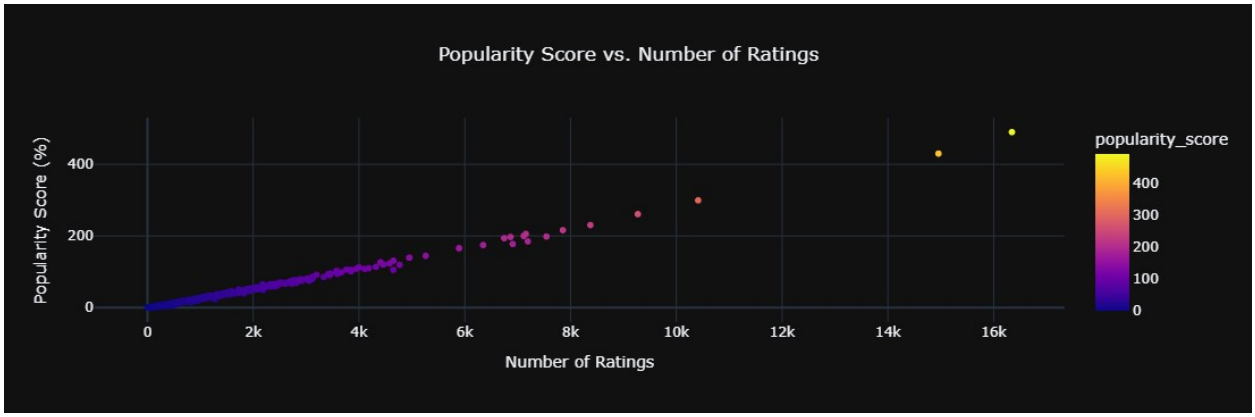
Purpose of Popularity Score

This feature helps quantify restaurant performance by considering both customer engagement (number of reviews) and satisfaction (ratings).

Benefits of Popularity Score

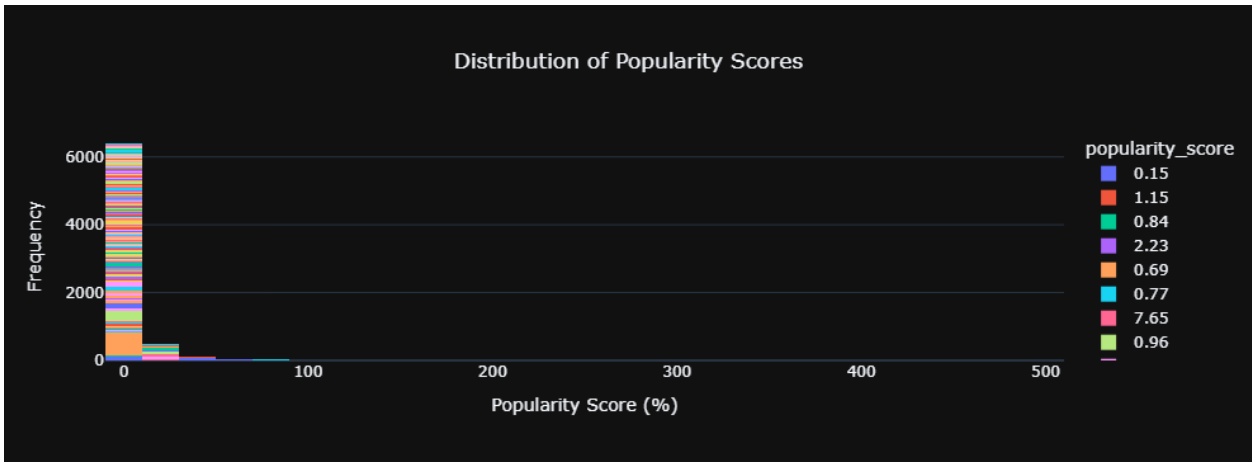
By normalizing the score, restaurants can be compared on a uniform scale, providing valuable insights for owners to improve services and for customers to identify popular establishments.

```
# Calculate the popularity score as a percentage and round it to 2
decimal places
df['popularity_score'] = round((df['rate (out of 5)']*df['num of
ratings'])/df['num of ratings'].max() *100, 2)

# Scatter plot for Popularity Score vs. Number of Ratings
fig = px.scatter(df,
                 x='num of ratings',
                 y='popularity_score',
                 title='Popularity Score vs. Number of Ratings',
                 color='popularity_score',  # Color by popularity
score
                 color_discrete_sequence=['#FFD700'],
                 template='plotly_dark'
)

fig.update_layout(
    xaxis_title='Number of Ratings',
    yaxis_title='Popularity Score (%)',
    title_x = 0.5


)

fig.show()
```




```
# Histogram for Popularity Score Distribution
fig = px.histogram(df,
                   x='popularity_score',
                   title='Distribution of Popularity Scores',
                   nbins=30,  # Adjust the number of bins
                   color='popularity_score',
                   template='plotly_dark'
                  )

fig.update_layout(
    title_x = 0.5,
```

```
    xaxis_title='Popularity Score (%)',
    yaxis_title='Frequency'
)

fig.show()
```



Conclusion and Recommendations

Popular Cuisines

- **Indian, Chinese, and Italian** are the most preferred cuisines.
- Restaurants should focus on these cuisines to attract more customers.

Cost Insights

- Most restaurants fall into the **medium price range**, indicating customer preference for affordable options.
- Offering **discounts** and **combo deals** can increase sales.

Online Ordering vs Table Booking

- A large number of customers prefer **online ordering**, making a strong digital presence essential.
- Improving delivery services can help retain more customers.

Area-Wise Demand

- Some areas have a **high restaurant density**, leading to competition, while others offer **growth opportunities**.

Customer Ratings & Service Improvements

- High ratings are linked to **service quality** and **affordable pricing**.
- Restaurants should focus on both to maintain customer satisfaction.

Feature Engineering Insights

- We created a **'popularity_score'** to measure restaurant performance based on ratings and reviews.
- This helps compare restaurants fairly and identify top-performing ones.