Name : Rishabh Rajput

Roll No: 2301560003

Course: Mca

Topic : Zomato

Data Analysis Project

Data Cleaning

Import necessary libraries

In [1]:
```
import pandas as pd
```

Load the dataset

In [10]:
```
file_path = "D:\\Project\\dataset\\zomato.csv"
df = pd.read_csv(file_path)
```

Deleting redundant columns

In [11]:
```
redundant_columns = ['url', 'address', 'menu_item']
df = df.drop(redundant_columns, axis=1)
```

Renaming the columns

In [12]:
```
column_mapping = {
    'online_order': 'online_order_available',
    'book_table': 'table_booking_available',
    'approx_cost(for two people)': 'cost_for_two',
    'reviews_list': 'reviews',
    'listed_in(type)': 'listed_in_type',
    'listed_in(city)': 'listed_in_city'
}
df = df.rename(columns=column_mapping)
```

Dropping duplicates

In [13]:
```
df = df.drop_duplicates()
```

Cleaning individual columns

In [14]:
```
df['rate'] = df['rate'].str.extract('(\d+\.\d+)').astype(float)
df['cost_for_two'] = df['cost_for_two'].str.replace(',', '').astype(float)
```

```
<>:1: SyntaxWarning: invalid escape sequence '\d'
<>:1: SyntaxWarning: invalid escape sequence '\d'
C:\Users\kesha\AppData\Local\Temp\ipykernel_9224\3156990001.py:1: SyntaxWarni
ng: invalid escape sequence '\d'
  df['rate'] = df['rate'].str.extract('(\d+\.\d+)').astype(float)
```

Remove NaN values

In [15]:
```
df = df.dropna()
```

Data Visualization

Import necessary libraries

In [16]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

Restaurants delivering Online or not

In [17]:
```
sns.countplot(x='online_order_available', data=df)
plt.title('Restaurants Delivering Online or Not')
plt.show()
```

Restaurants allowing table booking or not

In [18]:
```
sns.countplot(x='table_booking_available', data=df)
plt.title('Restaurants Allowing Table Booking or Not')
plt.show()
```

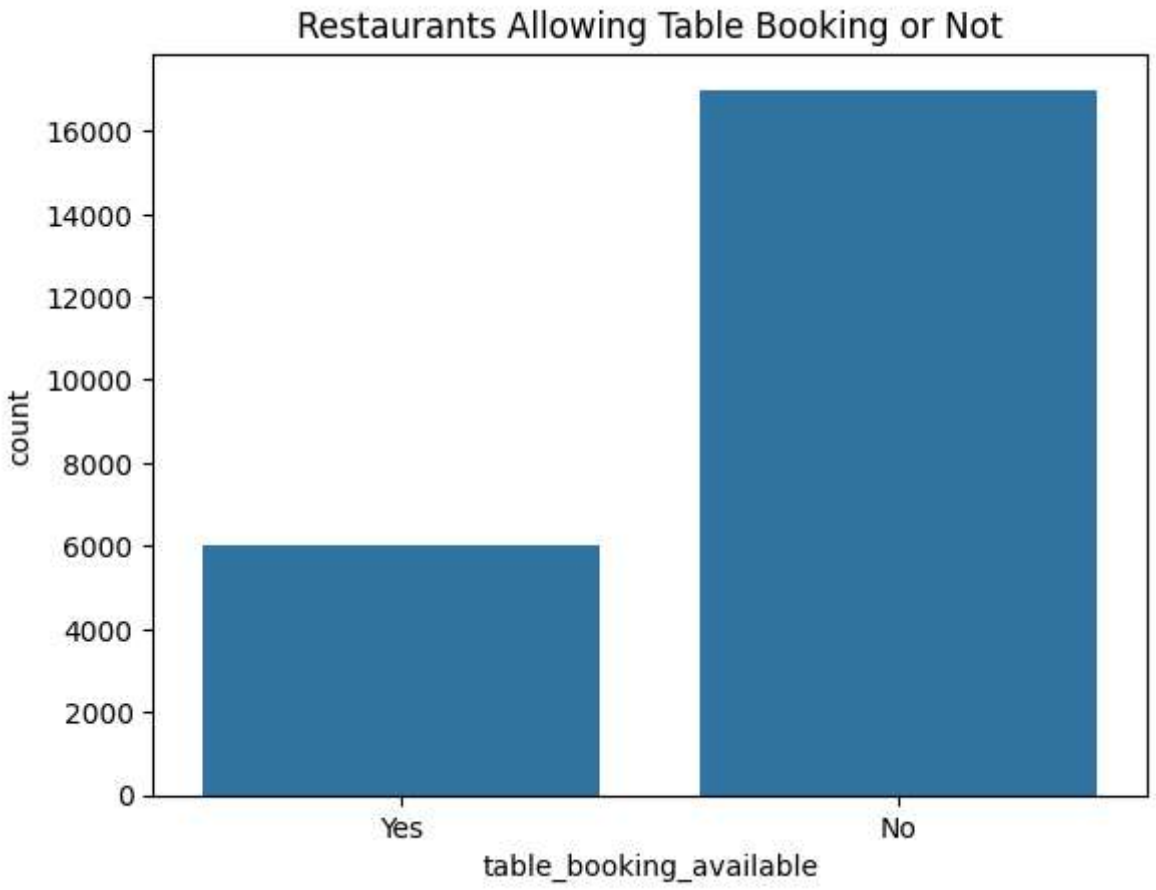
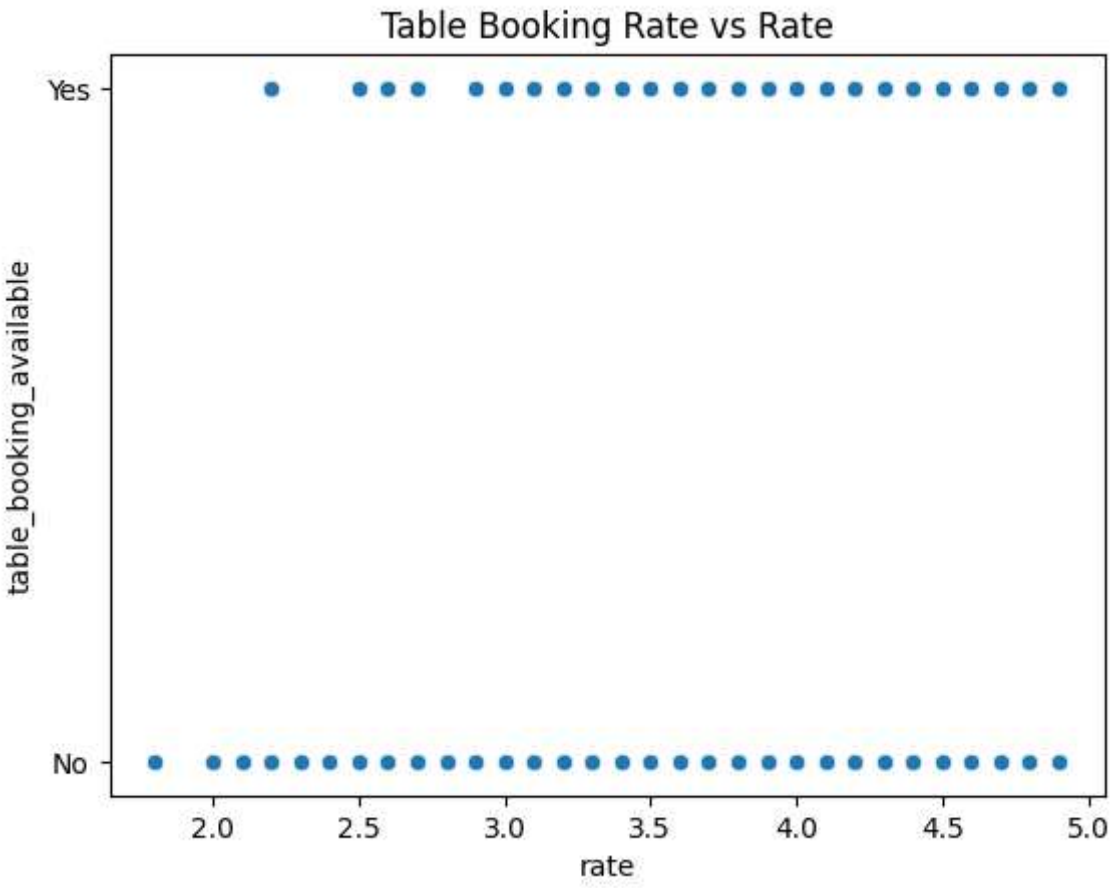


Table booking Rate vs Rate

In [19]:
```
sns.scatterplot(x='rate', y='table_booking_available', data=df)
plt.title('Table Booking Rate vs Rate')
plt.show()
```



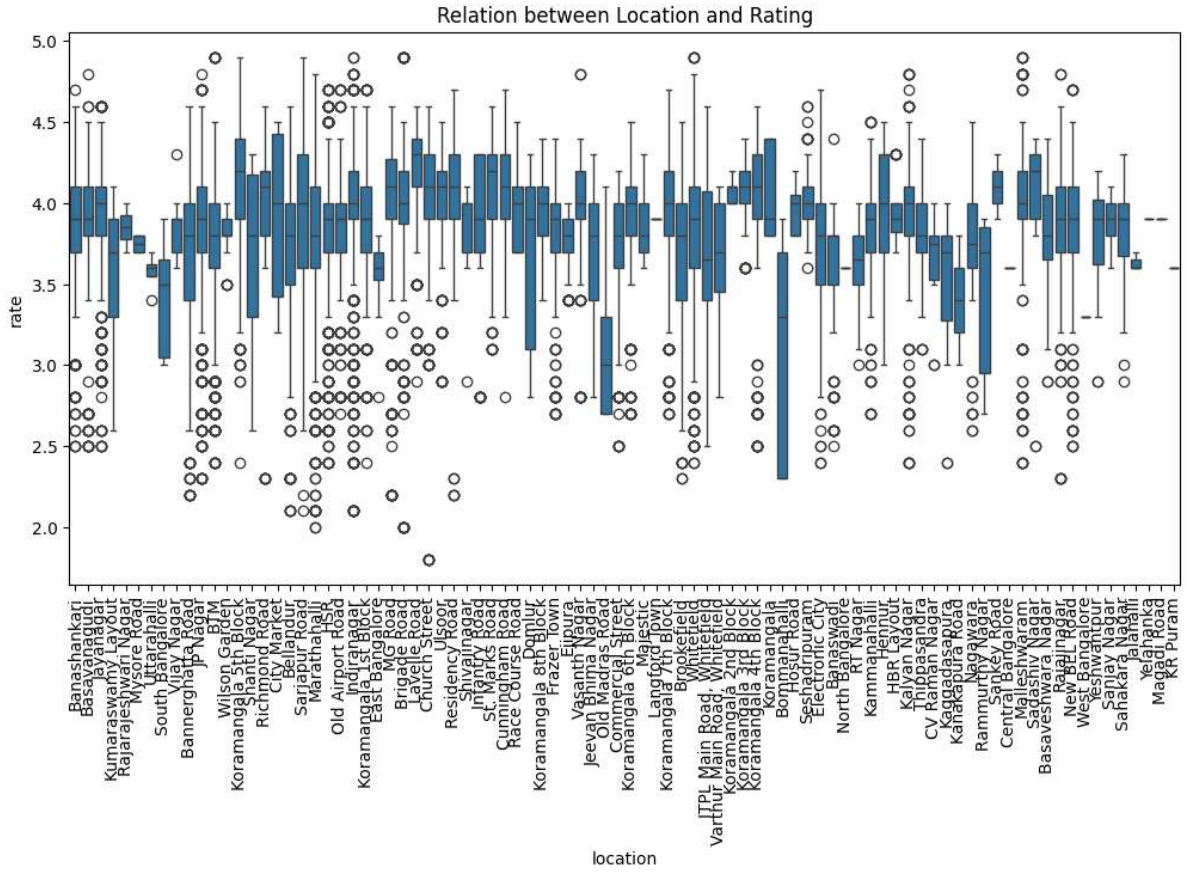
Best Location

In [20]:
```
best_location = df['location'].value_counts().idxmax()
print(f'Best Location: {best_location}')
```

```
Best Location: Koramangala 5th Block
```

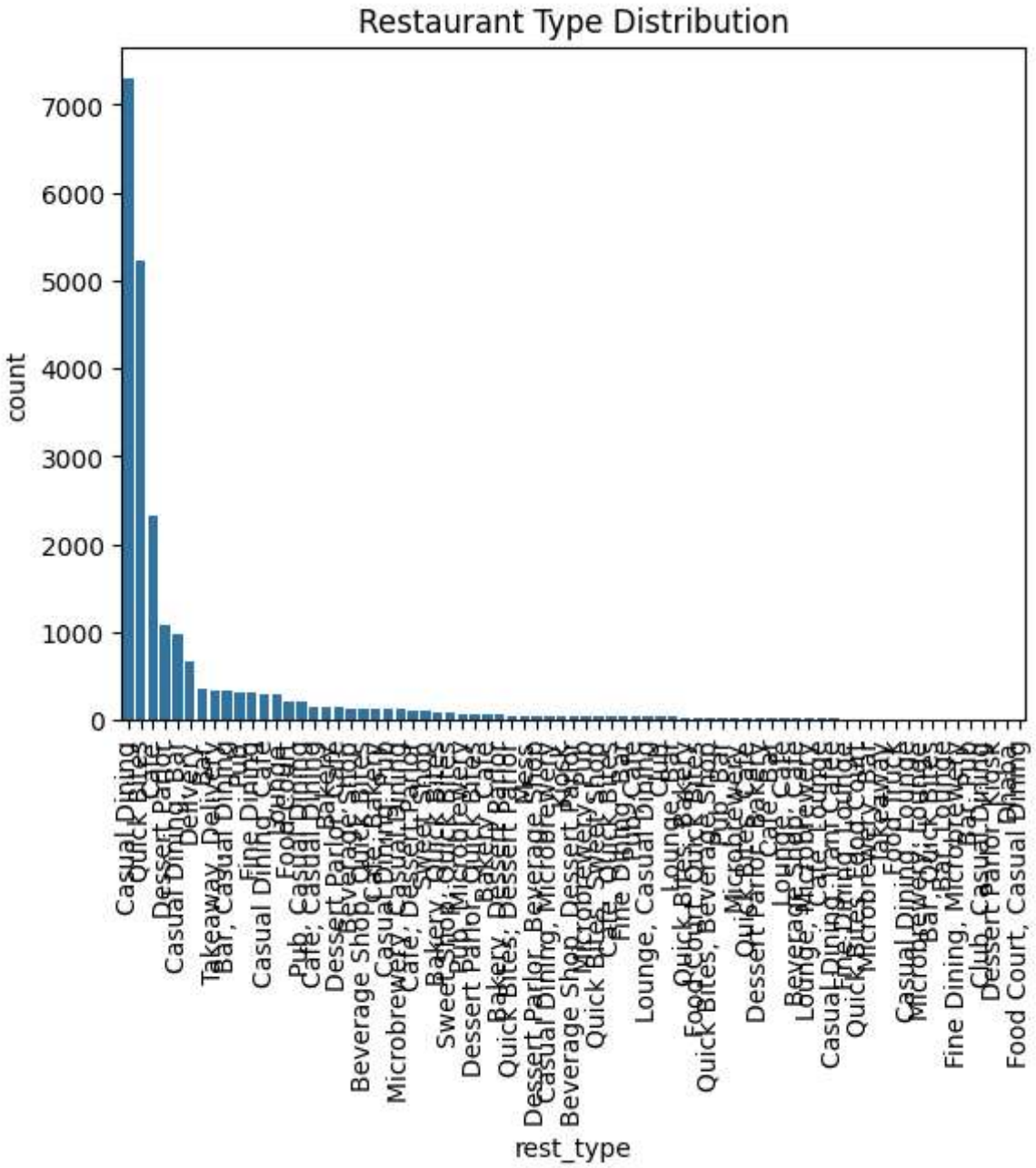
Relation between Location and Rating

In [21]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(x='location', y='rate', data=df)
plt.xticks(rotation=90)
plt.title('Relation between Location and Rating')
plt.show()
```



Restaurant Type

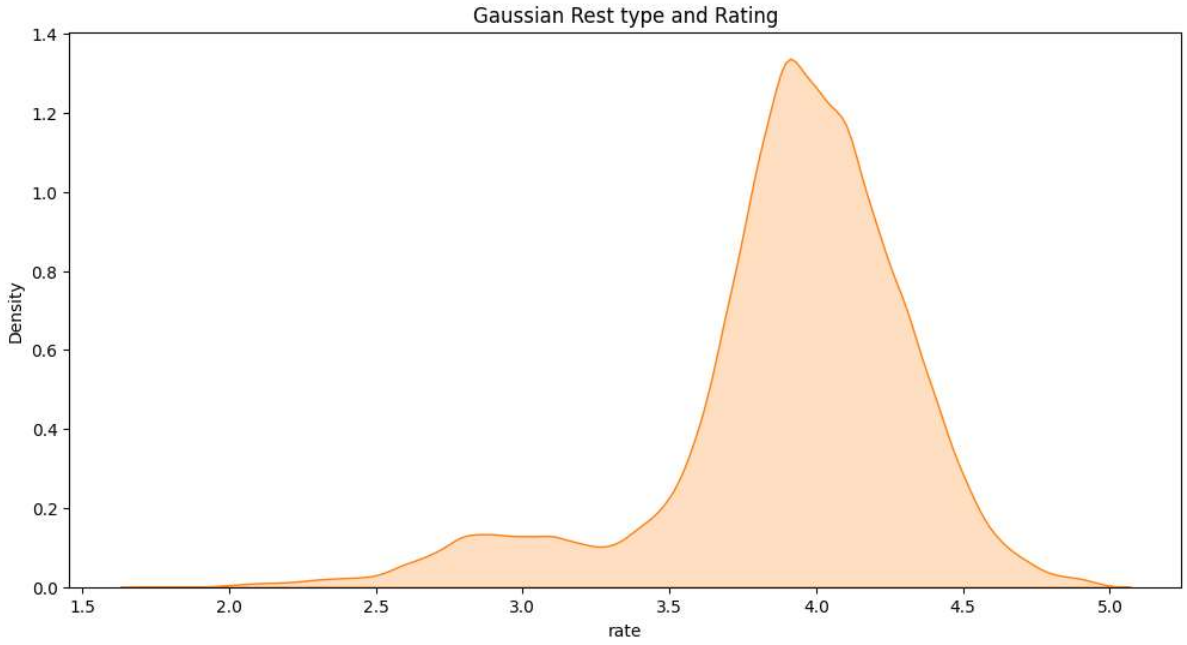
In [22]:
```
sns.countplot(x='rest_type', data=df, order=df['rest_type'].value_counts().inde
plt.xticks(rotation=90)
plt.title('Restaurant Type Distribution')
plt.show()
```



Gaussian Rest type and Rating

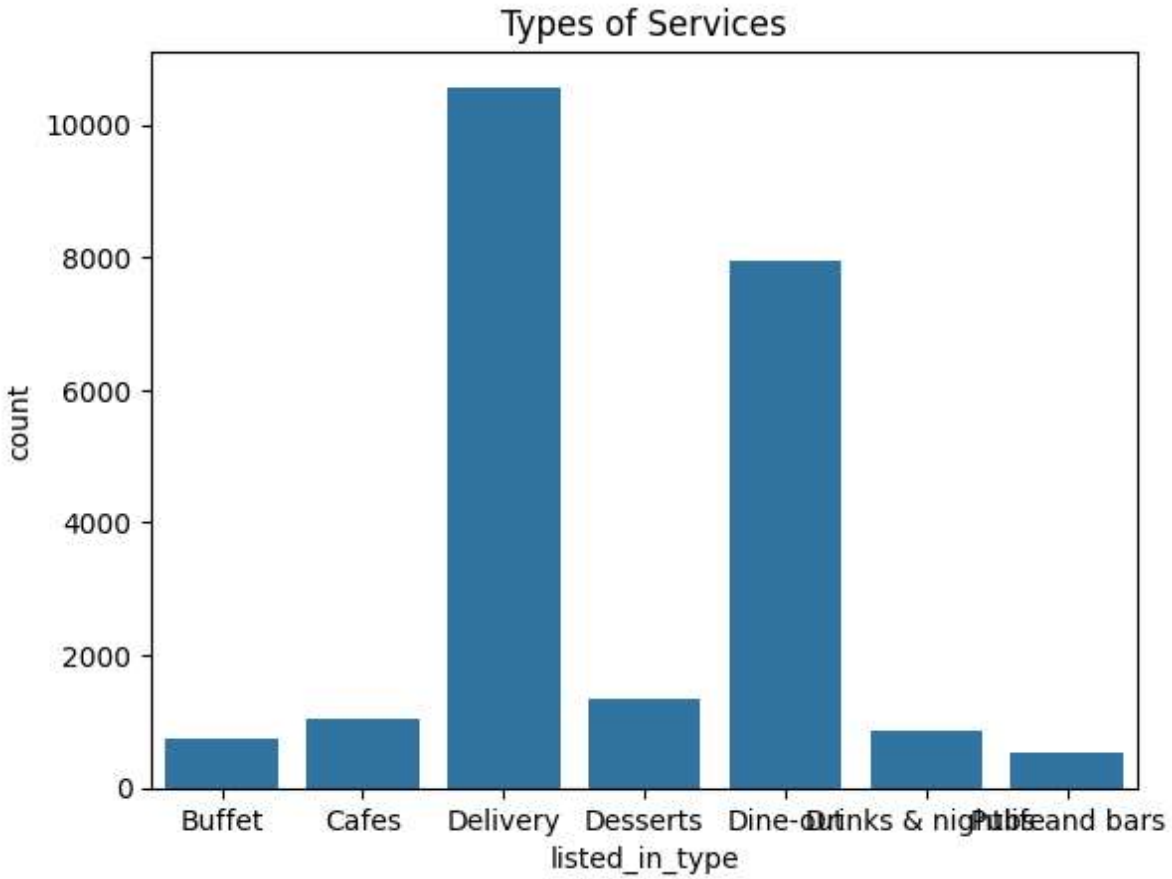
In [23]:
```
plt.figure(figsize=(12, 6))
sns.kdeplot(df['rate'][df['rest_type'] == 'Gaussian'], label='Gaussian', shade=
sns.kdeplot(df['rate'][df['rest_type'] != 'Gaussian'], label='Non-Gaussian', sh
plt.title('Gaussian Rest type and Rating')
plt.show()
```

```
C:\Users\kesha\AppData\Local\Temp\ipykernel_9224\2076677726.py:2: FutureWarni
ng:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df['rate'][df['rest_type'] == 'Gaussian'], label='Gaussian', sh
ade=True)
C:\Users\kesha\AppData\Local\Temp\ipykernel_9224\2076677726.py:3: FutureWarni
ng:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df['rate'][df['rest_type'] != 'Gaussian'], label='Non-Gaussia
n', shade=True)
```



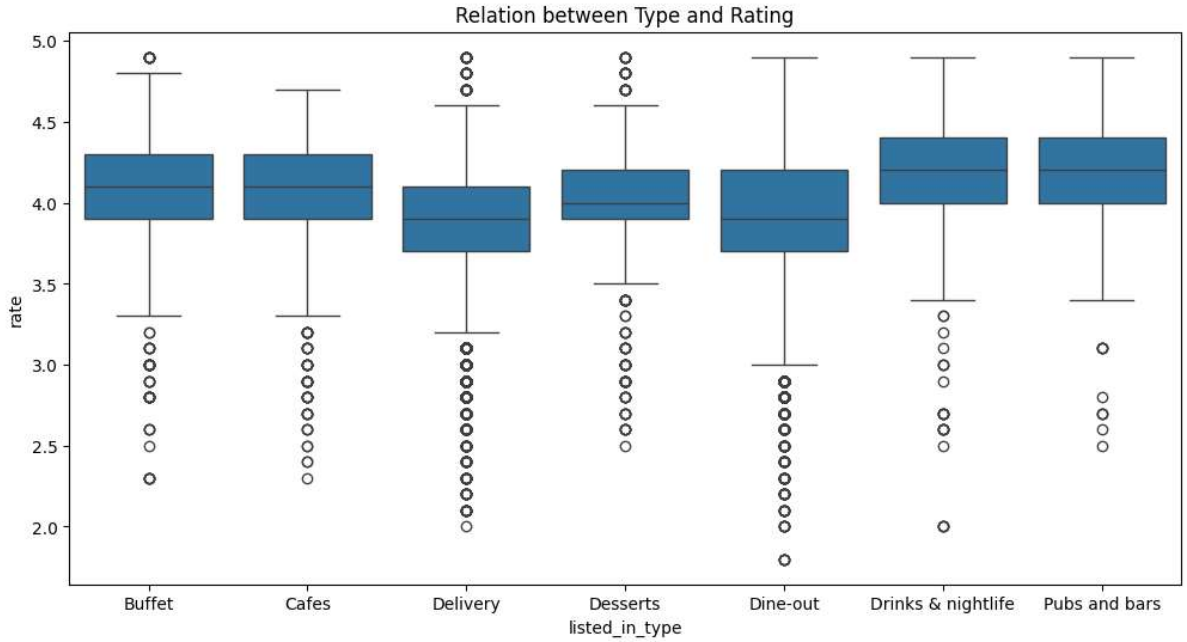
Types of Services

In [24]:
```
sns.countplot(x='listed_in_type', data=df)
plt.title('Types of Services')
plt.show()
```



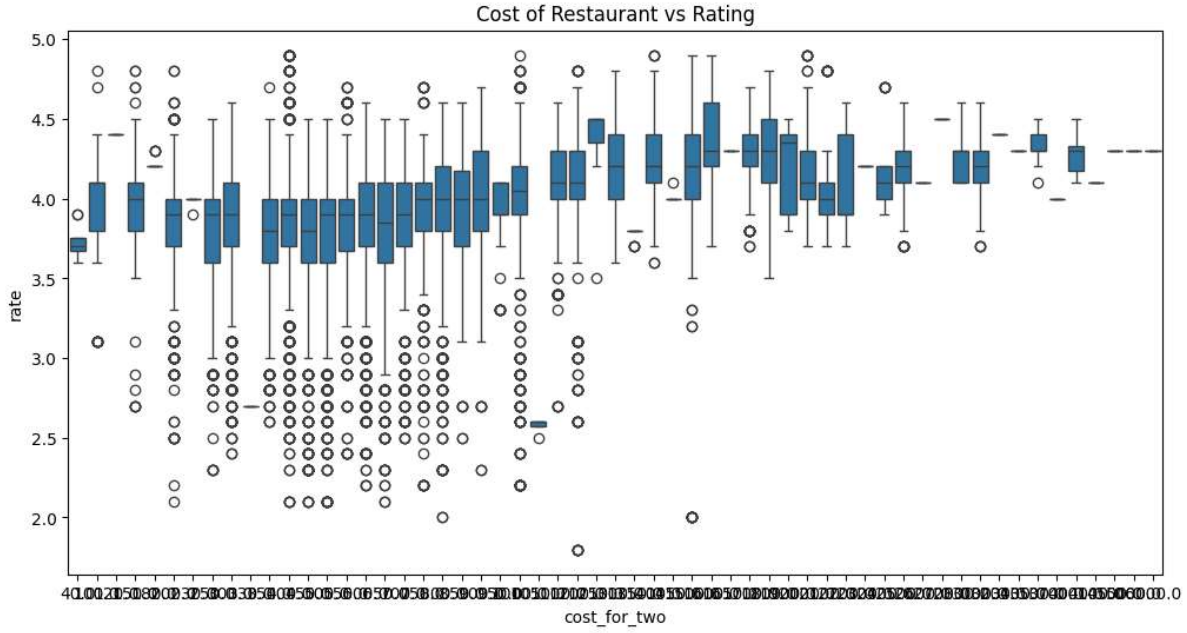
Relation between Type and Rating

In [25]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(x='listed_in_type', y='rate', data=df)
plt.title('Relation between Type and Rating')
plt.show()
```



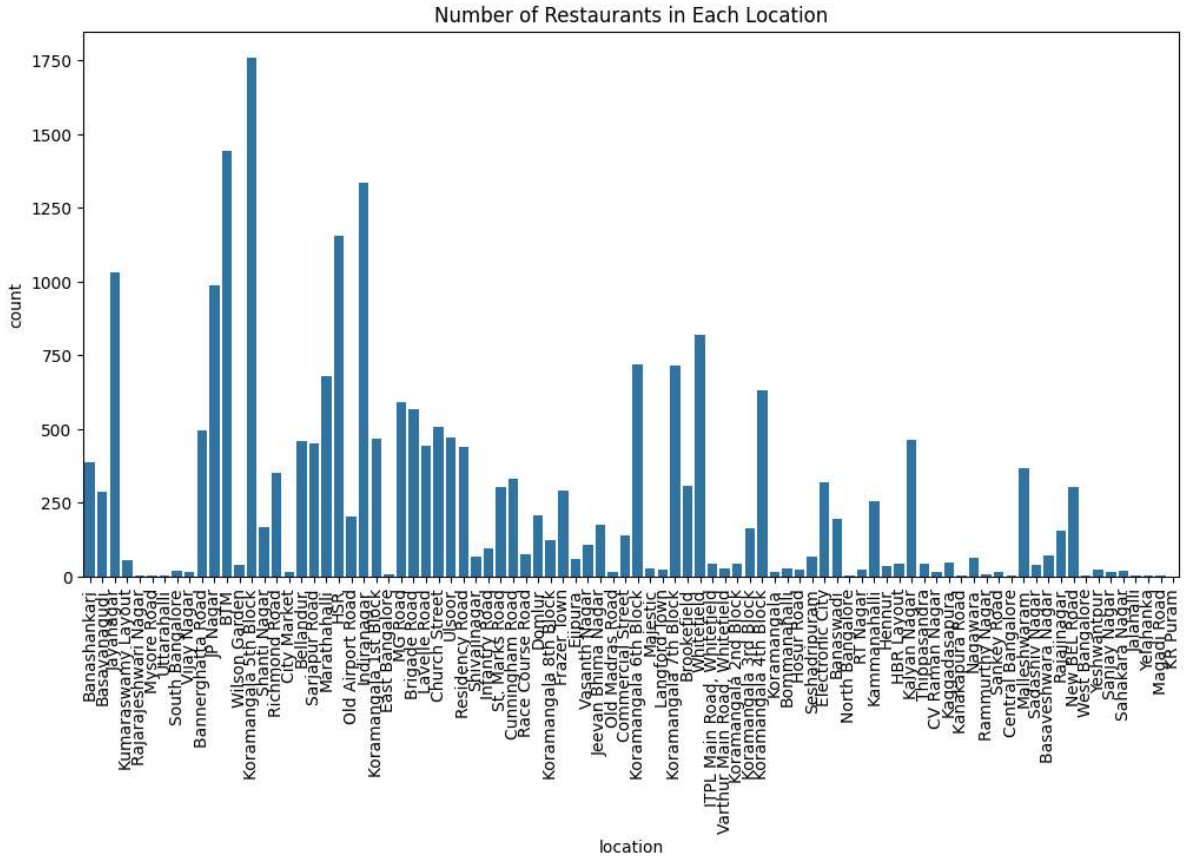
Cost of Restaurant

In [26]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(x='cost_for_two', y='rate', data=df)
plt.title('Cost of Restaurant vs Rating')
plt.show()
```



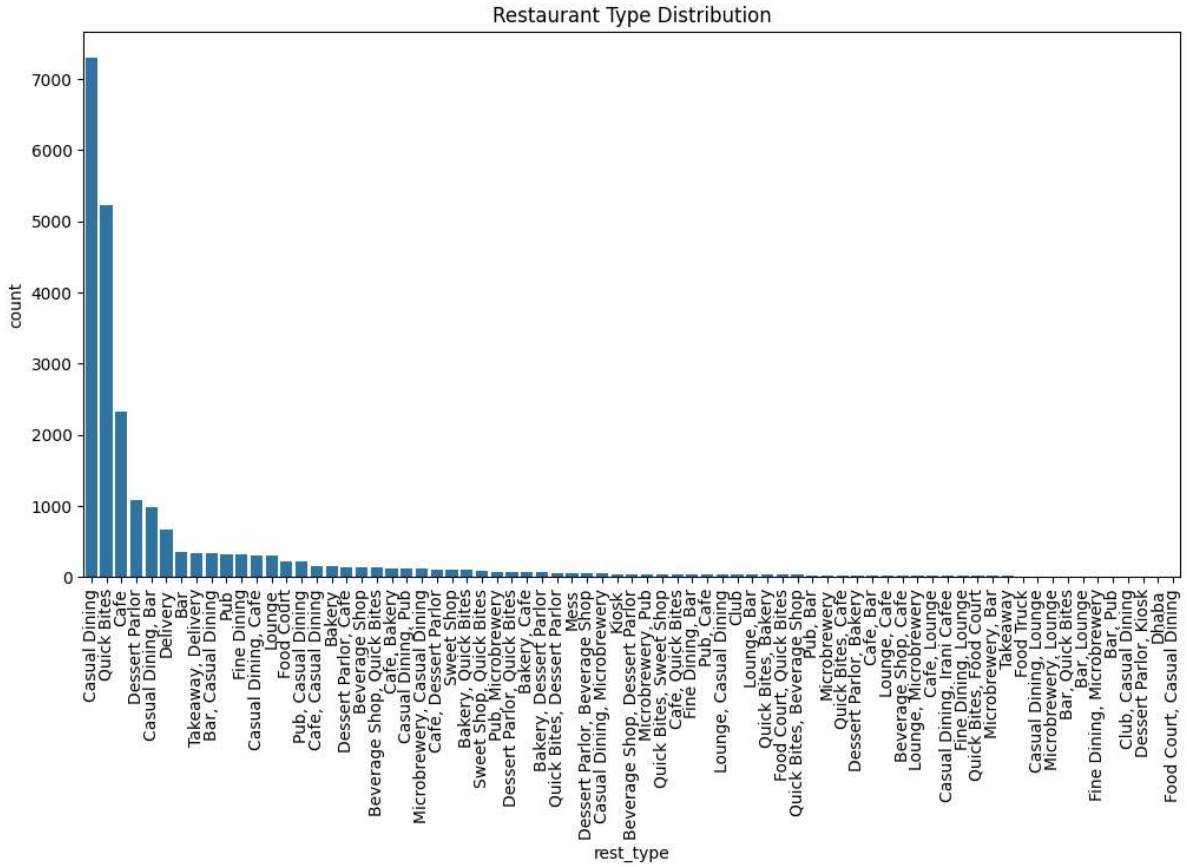
No. of restaurants in a Location

In [27]:
```
plt.figure(figsize=(12, 6))
sns.countplot(x='location', data=df)
plt.xticks(rotation=90)
plt.title('Number of Restaurants in Each Location')
plt.show()
```



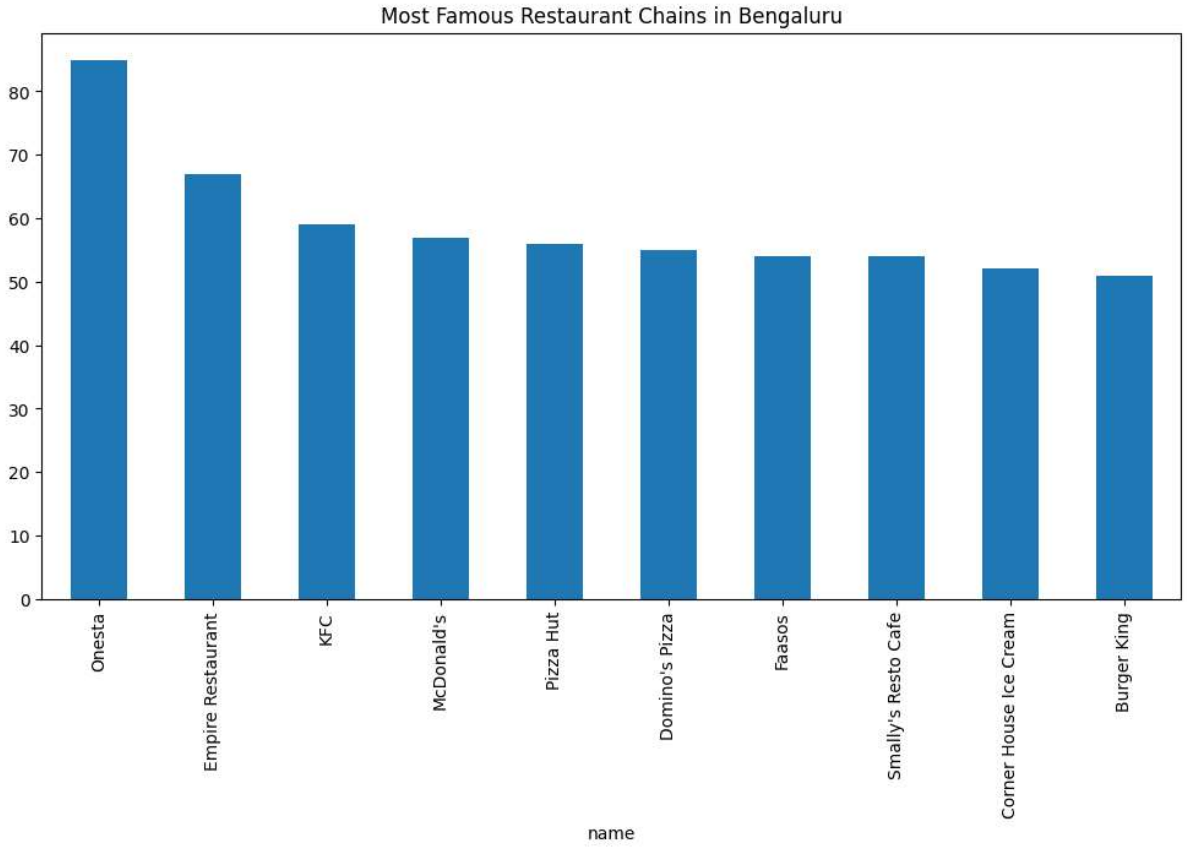
Restaurant type

In [28]:
```
plt.figure(figsize=(12, 6))
sns.countplot(x='rest_type', data=df, order=df['rest_type'].value_counts().inde
plt.xticks(rotation=90)
plt.title('Restaurant Type Distribution')
plt.show()
```



Most famous restaurant chains in Bengaluru

In [29]:
```
plt.figure(figsize=(12, 6))
df['name'].value_counts().head(10).plot(kind='bar')
plt.title('Most Famous Restaurant Chains in Bengaluru')
plt.show()
```



Regression Analysis

Import necessary libraries

In [32]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

Linear Regression

In [33]:
```
X = df[['cost_for_two']]
y = df['rate']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
```

In [34]:
```
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_predictions = lr_model.predict(X_test)
```

In [35]:
```
print(f'Linear Regression - Mean Squared Error: {mean_squared_error(y_test, lr_
print(f'Linear Regression - R2 Score: {r2_score(y_test, lr_predictions)}')
```

```
Linear Regression - Mean Squared Error: 0.15720575544683718
Linear Regression - R2 Score: 0.10350308182384993
```

Decision Tree Regression

In [36]:
```
dt_model = DecisionTreeRegressor()
dt_model.fit(X_train, y_train)
dt_predictions = dt_model.predict(X_test)
```

In [37]:
```
print(f'Decision Tree Regression - Mean Squared Error: {mean_squared_error(y_te
print(f'Decision Tree Regression - R2 Score: {r2_score(y_test, dt_predictions)}
```

```
Decision Tree Regression - Mean Squared Error: 0.1494087042613276
Decision Tree Regression - R2 Score: 0.1479673085870672
```

Random Forest Regression

In [38]:
```
rf_model = RandomForestRegressor()
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
```

In [39]:
```
print(f'Random Forest Regression - Mean Squared Error: {mean_squared_error(y_te
print(f'Random Forest Regression - R2 Score: {r2_score(y_test, rf_predictions)}
```

```
Random Forest Regression - Mean Squared Error: 0.14940083897642506
Random Forest Regression - R2 Score: 0.14801216193009936
```