Analyse of Swiggy sales and customer oriented data

Intoduction Our project aims to analyze and visualize the geographical distribution of restaurants listed in a dataset across various cities. By leveraging Python programming and data visualization techniques, we seek to gain insights into the spatial distribution of restaurants and their presence in different urban centers.44ur project aims to analyze and visualize the geographical distribution of restaurants listed in a dataset across various cities. By leveraging Python programming and data visualization techniques, we seek to gain insights into the spatial distribution of restaurants and their presence in different urban centers.

About the Dataset

link: https://drive.google.com/file/d/13M8upJ0gFkaJxQY0EpUAya5cx32Lb2gK/view?usp=drive_link

Dataset Analysis

Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import geopandas as gdp
```

Import dataset

In [2]:
```
df = pd.read_csv('swiggy.csv')
```

Display all column of dataset with 5 rows

In [3]:
```
df.head(5)
```

Out[3]:

	id	name	city	rating	rating_count	cost	lic_no	link	address	menu
0	567335	AB FOODS POINT	Abohar	--	Too Few Ratings	200.0	2.21E+13	https://www.swiggy.com/restaurants/ab-foods-po...	AB FOODS POINT, NEAR RISHI NARANG DENTAL CLINI...	Menu/567335.json
1	531342	Janta Sweet House	Abohar	4.4	50+ ratings	200.0	1.21E+13	https://www.swiggy.com/restaurants/janta-sweet...	Janta Sweet House, Bazar No.9, Circullar Road,...	Menu/531342.json
2	158203	theka coffee desi	Abohar	3.8	100+ ratings	100.0	2.21E+13	https://www.swiggy.com/restaurants/theka-coffe...	theka coffee desi, sahtiya sadan road city	Menu/158203.json
3	187912	Singh Hut	Abohar	3.7	20+ ratings	250.0	2.21E+13	https://www.swiggy.com/restaurants/singh-hut-n...	Singh Hut, CIRCULAR ROAD NEAR NEHRU PARK ABOHAR	Menu/187912.json
4	543530	GRILL MASTERS	Abohar	--	Too Few Ratings	250.0	1.21E+13	https://www.swiggy.com/restaurants/grill-maste...	GRILL MASTERS, ADA Heights, Abohar - Hanumanga...	Menu/543530.json

Display information of every column(non null values count , data type)

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148541 entries, 0 to 148540
Data columns (total 10 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   id            148541 non-null  int64
 1   name          148455 non-null  object
 2   city          148541 non-null  object
 3   rating        148455 non-null  object
 4   rating_count  148455 non-null  object
 5   cost          148410 non-null  float64
 6   lic_no        148312 non-null  object
 7   link          148541 non-null  object
 8   address       148455 non-null  object
 9   menu          148541 non-null  object
dtypes: float64(1), int64(1), object(8)
memory usage: 11.3+ MB
```

Display null values

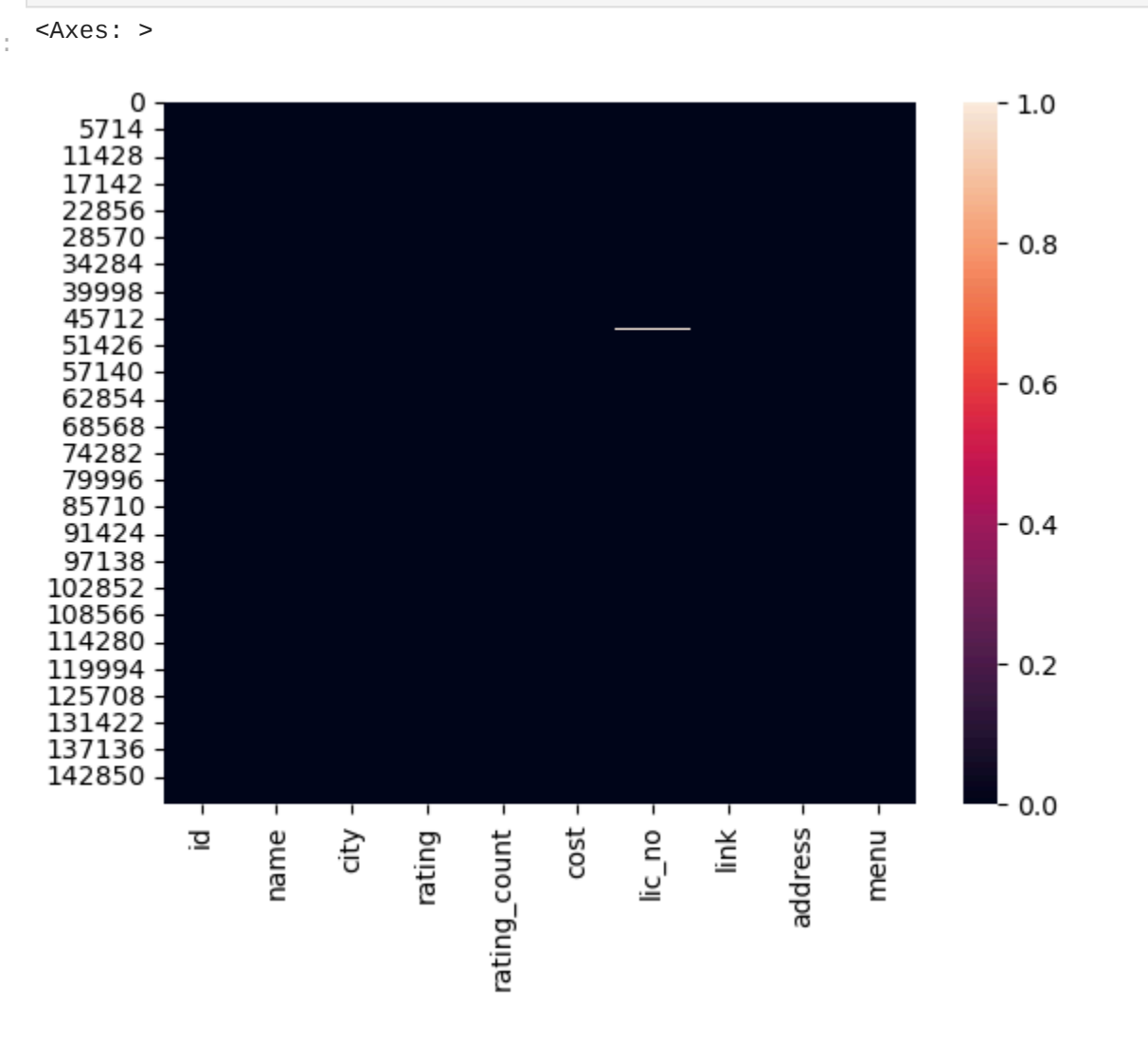
In [5]:
```
df.isnull().sum()
```

Out[5]:
```
id                0
name             86
city              0
rating           86
rating_count     86
cost            131
lic_no          229
link              0
address          86
menu              0
dtype: int64
```

Display null values in form of Heatmap

In [6]:
```
sns.heatmap(df.isnull())
```

Out[6]:
```
<Axes: >
```



In the dataset, lots of null values are present in every column. Need to handle all null values and clean the dataset.

Approach used to handlr null values-

1. Sample values from dataset.
2. Replacing null values with a new category (categorical).
3. Mean replacement.
4. Drop column.

Handle null values

In [7]:
```
df["rating"]=pd.to_numeric(df["rating"].replace("--",0), errors="coerce")
```

In [8]:
```
df["cost"]=pd.to_numeric(df["cost"].replace("nan",0.0), errors="coerce")
df["cost"] = df["cost"].fillna("0")
df['cost'] = df['cost'].astype(float)
```

In [9]:
```
df["rating"] = df["rating"].fillna(0)
df["rating"]=pd.to_numeric(df["rating"].replace("--",0), errors="coerce")
```

In [10]:
```
df["rating_count"] = df["rating_count"].fillna("No Rating")
```

In [11]:
```
df["address"] = df["address"].fillna("No Adress")
```

In [12]:
```
df["name"] = df["name"].fillna("No")
```

In [13]:
```
df["lic_no"] = df["lic_no"].str.replace("license","No License")
df["lic_no"] = df["lic_no"].str.replace("Pizzeria","No License")
df["lic_no"] = df["lic_no"].str.replace("NA","No License")
df['lic_no'] = df['lic_no'].replace(0, 'No License')
df["lic_no"] = df["lic_no"].fillna("No License")
```

In [14]:
```
df["lic_no"].unique()
```

Out[14]:
```
array(['2.21E+13', '1.21E+13', 'No License', '3.02E+13', '1.36E+13',
       '2.36E+13', '1.31E+13', '2.11E+13', '1.11E+13', '1.12E+13',
       '2.01E+13', '1.01E+13', '2.25E+13', '1.25E+13', '2.02E+13',
       '1.27E+13', '2.27E+13', '2.15E+13', '1.00E+13', '1.07E+13',
       '2.07E+13', '8.49E+13', '4.44E+13', '2.06E+13', '2.90E+13',
       '1.33E+13', '1.14E+13', '1.28E+13', '8.06E+13', '1.22E+13',
       '1.15E+13', '2.05E+13', '2.18E+13', '1.18E+13', '2.22E+13',
       '2.13E+13', '1.13E+13', '2.28E+13', '1.08E+13', '2.08E+13',
       '1.05E+13', '1.24E+13', '2.24E+13', 'ACA01510464281', '1.04E+13',
       '2.04E+13', '0', '1.09E+13', '2.09E+13', '2.12E+13', '2.14E+13',
       '2.20E+13', '1.20E+13', '2.33E+13', '9.03E+13', '1.30E+13',
       '9.07E+13', '1.72E+13', '1.19E+13', '9.11E+13', '6.46E+13',
       '1.44E+13', '5.15E+13', '1.26E+13', '1.80E+13', '1.52E+12',
       '2.03E+13', '1.03E+13', '3.87E+13', '1.23E+13', '1.06E+13',
       '2.32E+13', '2.30E+13', '1.42E+13', '2.26E+13', '2.24E+12',
       '8.54E+13', '4.20E+11', '1.35E+13', '2.32E+12', '13319010000579',
       '8.61E+13', '2.00E+13', '9.08E+13', '2.37E+13', '2.23E+13',
       '2.19E+13', '1.32E+13', '1.70E+13', '6.92E+13', '2.31E+13',
       '1.16E+13', '2.16E+13', '1.02E+13', '2.10E+13', '1.10E+13',
       '9.09E+13', '1.29E+13', '1.01E+12', '4.92E+13', '2.46E+13',
       '2.38E+13', '9.06E+13', '2.52E+13', '2.77E+13', '1.36E+12',
       '2.35E+13', '7.27E+13', 'BC100001219126', '2.29E+13', '1.34E+13',
       '3.22E+13', '4.11E+13', '2.94E+13', '1.17E+13', '2.17E+13',
       '4.21E+13', '4.99E+13', '9.12E+13', '3.83E+13', '3.01E+13'],
      dtype=object)
```

Check null values after hadle null values . All null values are repalced and dataset is clean.

In [15]:
```
df.isnull().sum()
```

Out[15]:
```
id              0
name            0
city            0
rating          0
rating_count    0
cost            0
lic_no          0
link            0
address         0
menu            0
dtype: int64
```

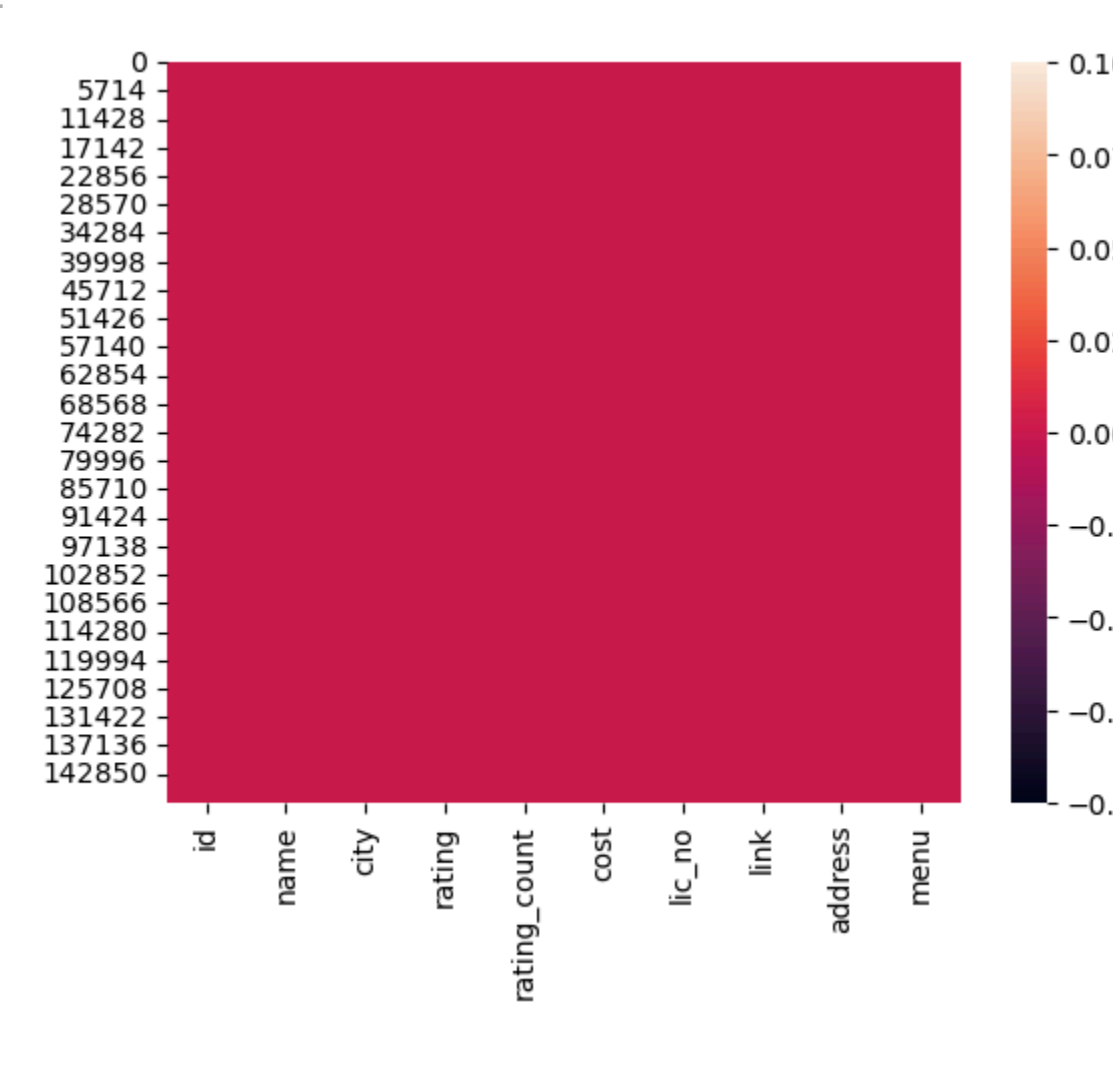
In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148541 entries, 0 to 148540
Data columns (total 10 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   id            148541 non-null  int64
 1   name          148541 non-null  object
 2   city          148541 non-null  object
 3   rating        148541 non-null  float64
 4   rating_count  148541 non-null  object
 5   cost          148541 non-null  float64
 6   lic_no        148541 non-null  object
 7   link          148541 non-null  object
 8   address       148541 non-null  object
 9   menu          148541 non-null  object
dtypes: float64(2), int64(1), object(7)
memory usage: 11.3+ MB
```

Display Heatmap

In [17]:
```
sns.heatmap(df.isnull())
```

Out[17]:
```
<Axes: >
```



Display descriptive statistics of dataset

In [18]:
```
df.head()
```

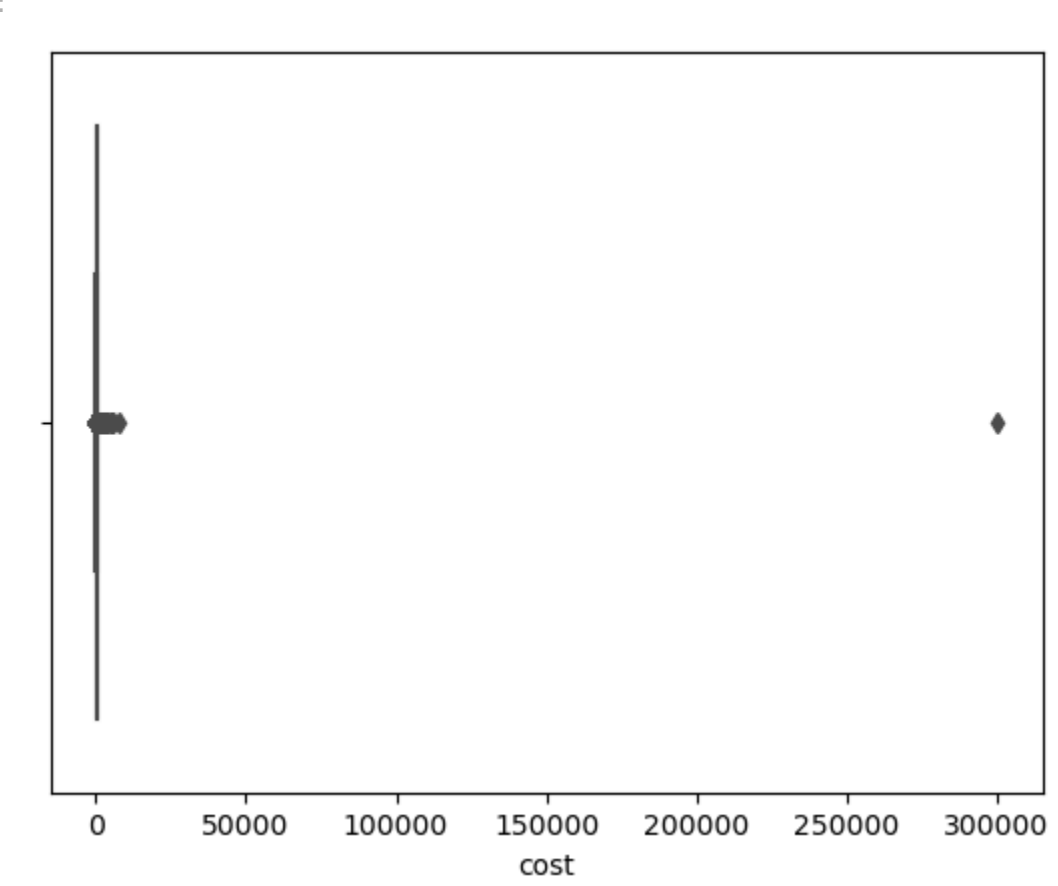
Out[18]:

	id	name	city	rating	rating_count	cost	lic_no	link	address	menu
0	567335	AB FOODS POINT	Abohar	0.0	Too Few Ratings	200.0	2.21E+13	https://www.swiggy.com/restaurants/ab-foods-po...	AB FOODS POINT, NEAR RISHI NARANG DENTAL CLINI...	Menu/567335.json
1	531342	Janta Sweet House	Abohar	4.4	50+ ratings	200.0	1.21E+13	https://www.swiggy.com/restaurants/janta-sweet...	Janta Sweet House, Bazar No.9, Circullar Road,...	Menu/531342.json
2	158203	theka coffee desi	Abohar	3.8	100+ ratings	100.0	2.21E+13	https://www.swiggy.com/restaurants/theka-coffe...	theka coffee desi, sahtiya sadan road city	Menu/158203.json
3	187912	Singh Hut	Abohar	3.7	20+ ratings	250.0	2.21E+13	https://www.swiggy.com/restaurants/singh-hut-n...	Singh Hut, CIRCULAR ROAD NEAR NEHRU PARK ABOHAR	Menu/187912.json
4	543530	GRILL MASTERS	Abohar	0.0	Too Few Ratings	250.0	1.21E+13	https://www.swiggy.com/restaurants/grill-maste...	GRILL MASTERS, ADA Heights, Abohar - Hanumanga...	Menu/543530.json

Outlier

In [19]:
```
sns.boxplot(data=df,x='cost',color='blue')
```

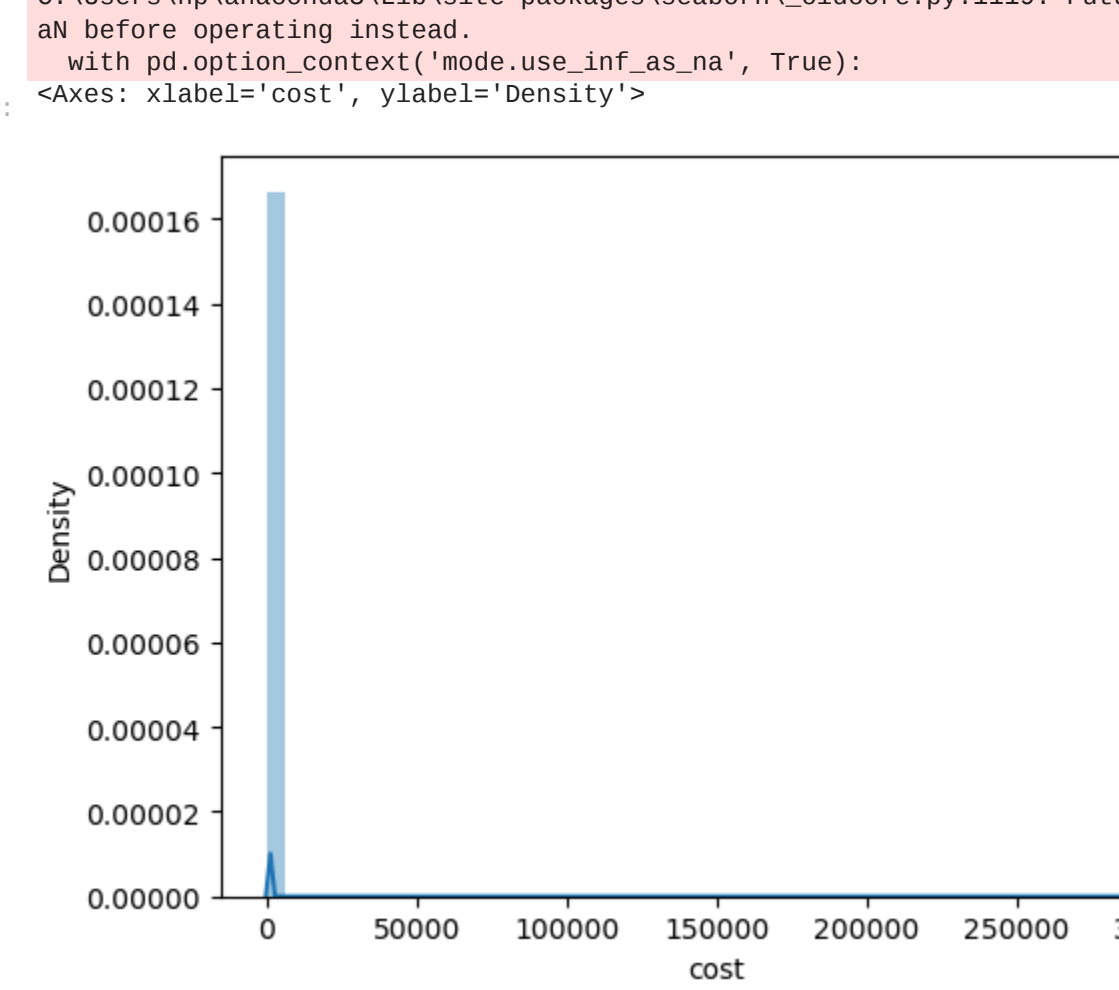
Out[19]:
```
<Axes: xlabel='cost'>
```



In [20]:
```
sns.distplot(df['cost'])
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_4388\1352492925.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['cost'])
C:\Users\hp\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to N
aN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[20]:
```
<Axes: xlabel='cost', ylabel='Density'>
```



Approach Use-

Remove outlier with IQR (Interquartile Range)

In [21]:
```
std=df["cost"].std()
std
```

Out[21]:
```
796.4250426177296
```

In [22]:
```
iqr=df["cost"].quantile(0.75)-df["cost"].quantile(0.25)
iqr
```

Out[22]:
```
100.0
```

In [23]:
```
upperlimit=iqr+1.5*std
lowerlimit=iqr-1.5*std
```

In [24]:
```
upperlimit
```

Out[24]:
```
1294.6375639265943
```

In [25]:
```
lowerlimit
```

Out[25]:
```
-1094.6375639265943
```

In [26]:
```
df["cost"] = df[(df["cost"] >= lowerlimit) & (df["cost"] <= upperlimit)]["cost"]
```

In [27]:
```
sd=df['cost'].std()
sd
```

Out[27]:
```
145.56961610948744
```

In [28]:
```
iqr=df["cost"].quantile(0.75)-df["cost"].quantile(0.25)
iqr
```

Out[28]:
```
100.0
```

In [29]:
```
upperlimit=iqr+1.5*sd
lowerlimit=iqr-1.5*sd
```

In [30]:
```
upperlimit
```

Out[30]:
```
318.35442416423115
```

In [31]:
```
lowerlimit
```

Out[31]:
```
-118.35442416423115
```

In [32]:
```
df.loc[df['cost']>318,"cost"]=318
```

In [33]:
```
lowerlimit = -118
df["cost"] = df[df["cost"] >= lowerlimit]["cost"]
```

In [34]:
```
sd=df['cost'].std()
sd
```

Out[34]:
```
67.98888890287151
```

In [35]:
```
iqr=df["cost"].quantile(0.75)-df["cost"].quantile(0.25)
iqr
```

Out[35]:
```
100.0
```

In [36]:
```
upperlimit=iqr+1.5*sd
lowerlimit=iqr-1.5*sd
```

In [37]:
```
lowerlimit
```

Out[37]:
```
-1.9833333543072627
```

In [38]:
```
df["cost"] = df[df["cost"] >= lowerlimit]["cost"]
```

In [39]:
```
std_d=df['cost'].std()
std_d
```

Out[39]:
```
67.98888890287151
```

In [40]:
```
df = df[df["cost"] >= max(0, lowerlimit)]
```

In [41]:
```
lowerlimit
```

Out[41]:
```
-1.9833333543072627
```

In [42]:
```
df["cost"] = df[df["cost"] >= lowerlimit]["cost"]
```

In [43]:
```
st_d=df['cost'].std()
st_d
```

Out[43]:
```
67.98888890287151
```

In [44]:
```
df = df[df["cost"] >= max(0, lowerlimit)]
```

In [45]:
```
lowerlimit
```

Out[45]:
```
-1.9833333543072627
```

In [46]:
```
lowerlimit = 0
df["cost"] = df[df["cost"] >= lowerlimit]["cost"]
std_dev = df['cost'].std()
print(std_dev)
df = df[df["cost"] >= max(0, lowerlimit)]
```

```
67.98888890287151
```

In [47]:
```
lowerlimit
```

Out[47]:
```
0
```

In [48]:
```
# lowerlimit = 0
df["cost"] = df[df["cost"] >= lowerlimit]["cost"]
std_devi = df['cost'].std()
print(std_devi)
df = df[df["cost"] >= max(0, lowerlimit)]
```

```
67.98888890287151
```

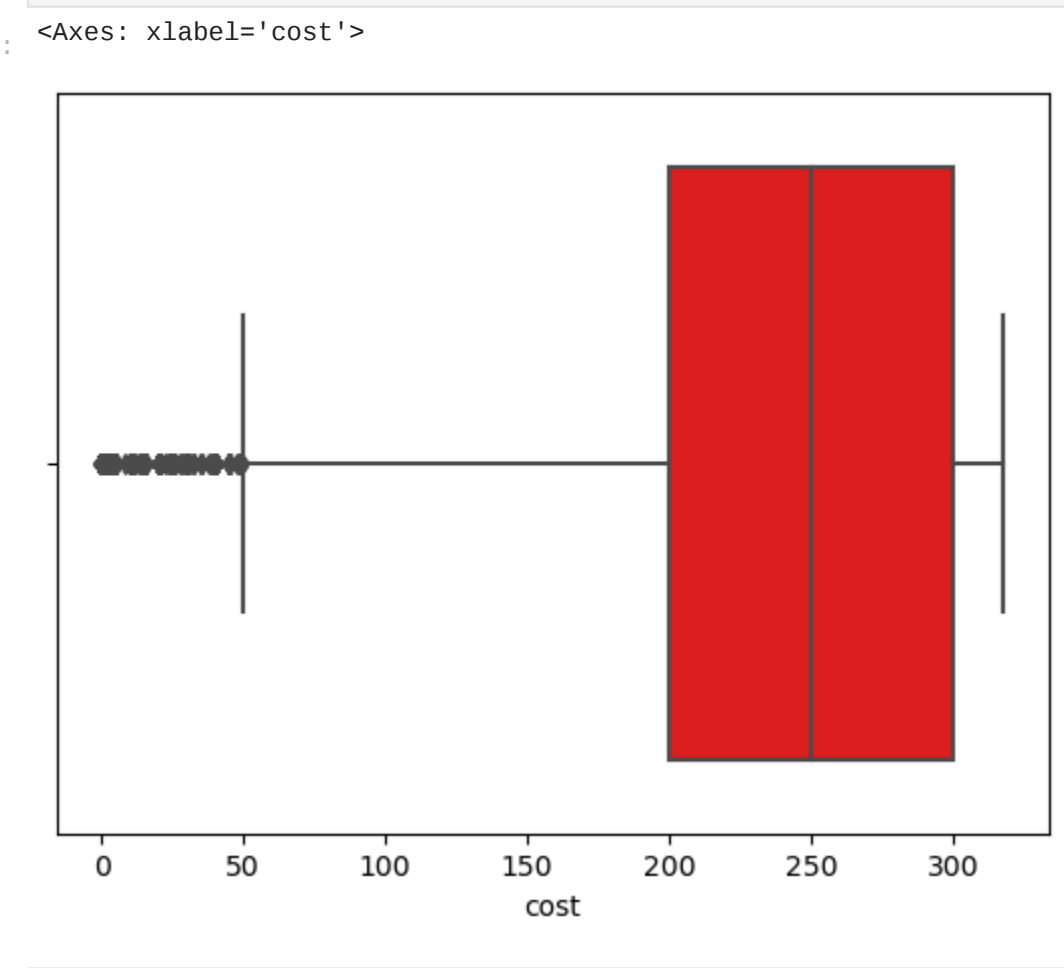
In [49]:
```
df = df[df["cost"] >= max(0, lowerlimit)]
```

In [50]:
```
lowerlimit
```

Out[50]:
```
0
```

In [51]:
```
sns.boxplot(data=df,x='cost',color='red')
```

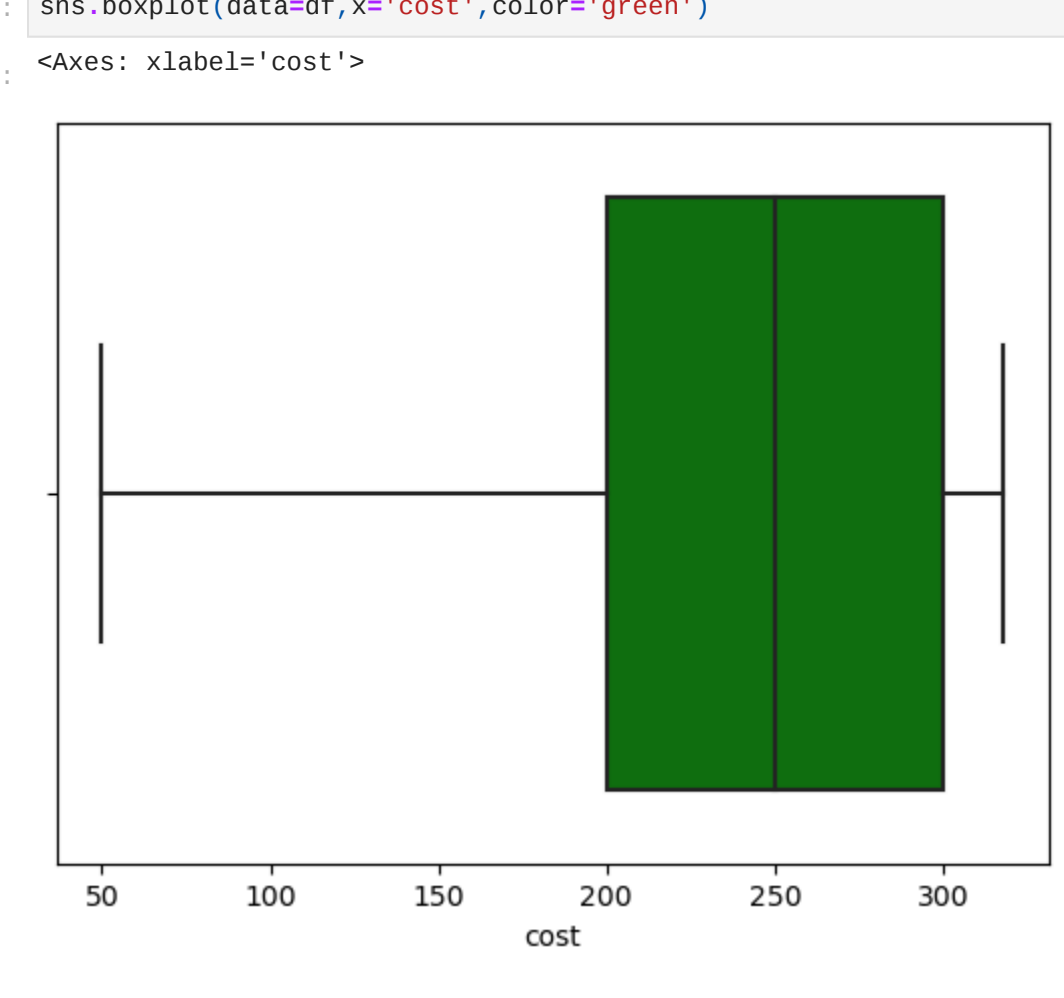
Out[51]:
```
<Axes: xlabel='cost'>
```



In [52]:
```
Q1 = df['cost'].quantile(0.25)
Q3 = df['cost'].quantile(0.75)
IQR = Q3 - Q1
filter = (df['cost'] >= Q1 - 1.5 * IQR) & (df['cost'] <= Q3 + 1.5 *IQR)
df = df.loc[filter]
```

In [53]:
```
sns.boxplot(data=df,x='cost',color='green')
```

Out[53]:
```
<Axes: xlabel='cost'>
```

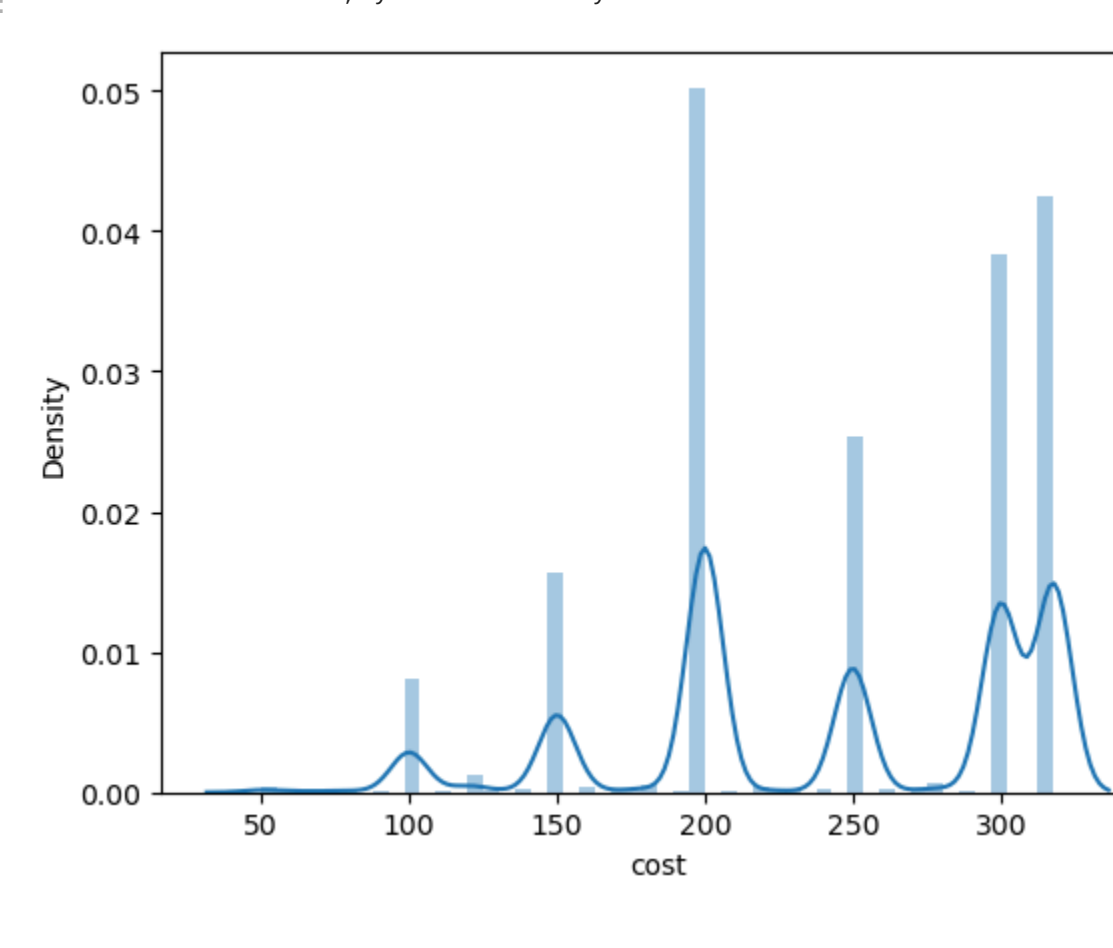


Distribution plot after remove outliers

In [54]:
```
sns.distplot(df['cost'])
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_4388\1352492925.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['cost'])
C:\Users\hp\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to N
aN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[54]:
```
<Axes: xlabel='cost', ylabel='Density'>
```



Q1: How many restaurants are listed on Swiggy?

In [55]:
```
unique_restaurants = df['name'].nunique()
print(f'Total number of restaurants listed on Swiggy: {unique_restaurants}')
```

```
Total number of restaurants listed on Swiggy: 112125
```

Q2: How many cities are having restaurants that are listed on Swiggy?

In [56]:
```
df_lic_no = df[df['lic_no'].notna()]
listed_restaurants = df_lic_no.groupby('lic_no')['city'].nunique().reset_index()
restaurants_lic_no = listed_restaurants['city'].sum()
print(f'Total restaurants with unique lic_no: {restaurants_lic_no}')
```

```
Total restaurants with unique lic_no: 3363
```

Q3: Which city is having the maximum number of restaurants?

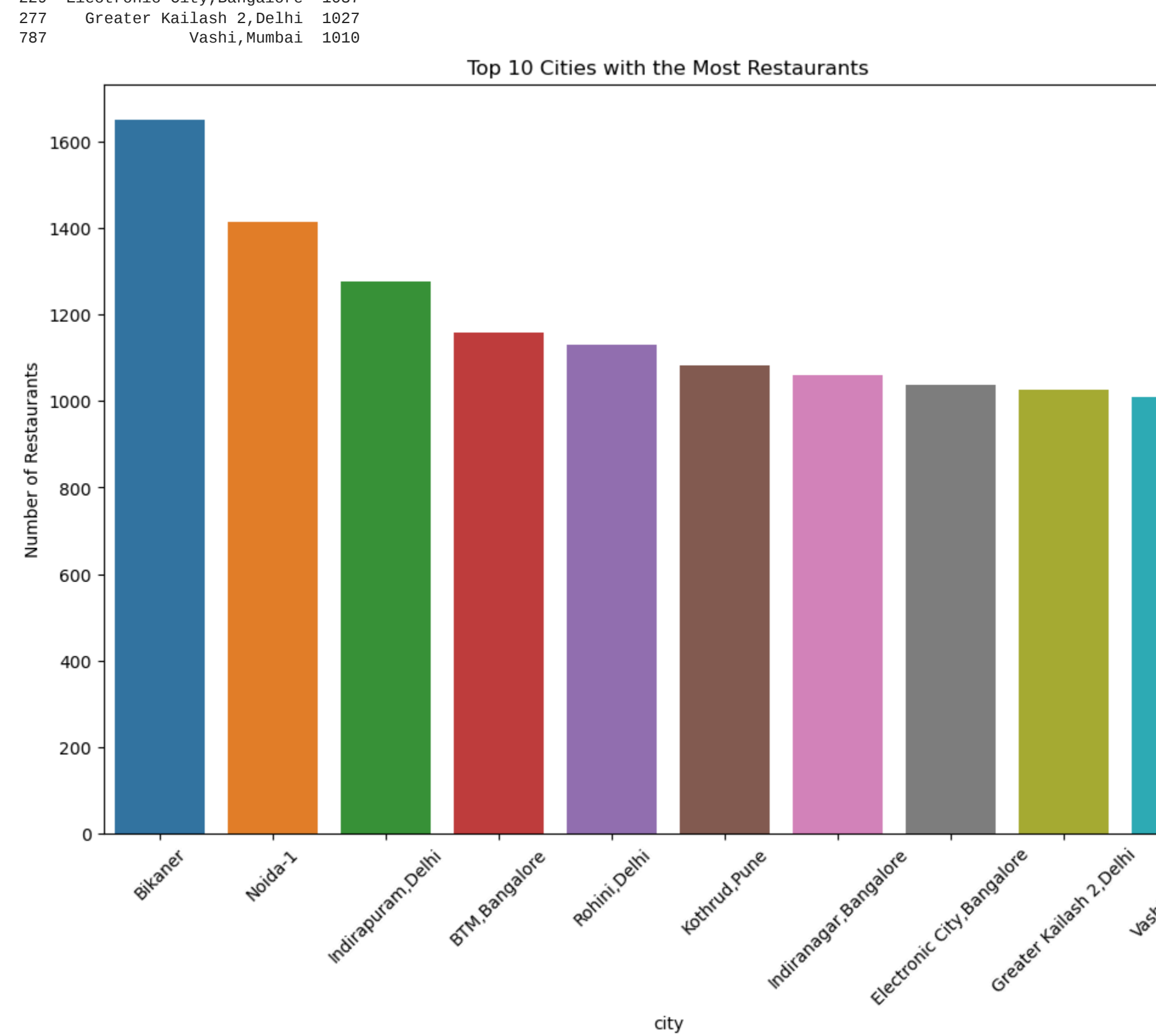
In [57]:
```
restaurants_in_city = df.groupby('city').size().reset_index(name='name')
max_restaurants = restaurants_in_city.loc[restaurants_in_city['name'].idxmax()]
print(f"City have the maximum number of restaurants '{max_restaurants['city']}'- {max_restaurants['name']} restaurants.")
```

```
City have the maximum number of restaurants 'Bikaner'- 1651 restaurants.
```

Q4: Top 10 cities as per the number of restaurants listed?

In [58]:
```
restaurant_within_city = df.groupby('city').size().reset_index(name='name')
top_10_cities = restaurant_within_city .sort_values(by='name', ascending= False).head(10)
print("Top 10 cities have the most no of restaurants:")
print(top_10_cities)
plt.figure(figsize=(12, 8))
sns.barplot(x='city', y='name', data=top_10_cities)
plt.xlabel('city')
plt.ylabel('Number of Restaurants')
plt.title('Top 10 Cities with the Most Restaurants')
plt.xticks(rotation=45)
plt.show()
```

```
Top 10 cities have the most no of restaurants:
                         city  name
138                   Bikaner  1651
567                   Noida-1  1416
322         Indirapuram,Delhi  1278
59              BTM,Bangalore  1160
657              Rohini,Delhi  1130
425              Kothrud,Pune  1084
321     Indiranagar,Bangalore  1061
229  Electronic City,Bangalore  1037
277     Greater Kailash 2,Delhi  1027
787               Vashi,Mumbai  1010
```



With the help of this bar graph, we can easily see the top 10 cities where Swiggy has the most restaurants present.

Q5: Most Popular hotel by rating throughout the dataset?

In [59]:
```
hotel_rating = df.groupby('name')['rating'].mean().reset_index()
hotel_5star_rating =hotel_rating.loc[hotel_rating['rating'].idxmax()]
print(f"Most popular hotel by rating throughout the dataset is '{hotel_5star_rating['name']}' with rating of- {hotel_5star_rating['rating']:.2f}.")
```

```
Most popular hotel by rating throughout the dataset is 'AP's Kitchen' with rating of- 5.00.
```

Q6: Which city is having the least expensive restaurant in terms of cost?

In [60]:
```
least_exp_restaurant = df.loc[df['cost'].idxmin()]
least_exp_res_in_city = least_exp_restaurant['city']
restaurant_name = least_exp_restaurant['name']
restaurant_cost = least_exp_restaurant['cost']
print(f"Least ecpensive res in city is: '{least_exp_res_in_city}', name of the resturant is '{restaurant_name}' cost of {restaurant_cost}.")
```

```
Least ecpensive res in city is: 'Adityapur', name of the resturant is 'Sehat World' cost of 50.0.
```

Q7 Top resturants those habe the 5 rating

In [61]:
```
restaurant_rating = df.groupby('name')['rating'].mean().reset_index()
restaurants =restaurant_rating.sort_values(by='rating', ascending=False).head(5)
print("Top 5 restaurants:")
print(restaurants)
```

```
Top 5 restaurants:
                        name  rating
109826   Zaika Darbaar Junction     5.0
33586        Fresh crunch pizza     5.0
70168              PEDRO DE GOA     5.0
82072   SHREE SAMARTHA CHAPATIS     5.0
24158          Creams and Bites     5.0
```