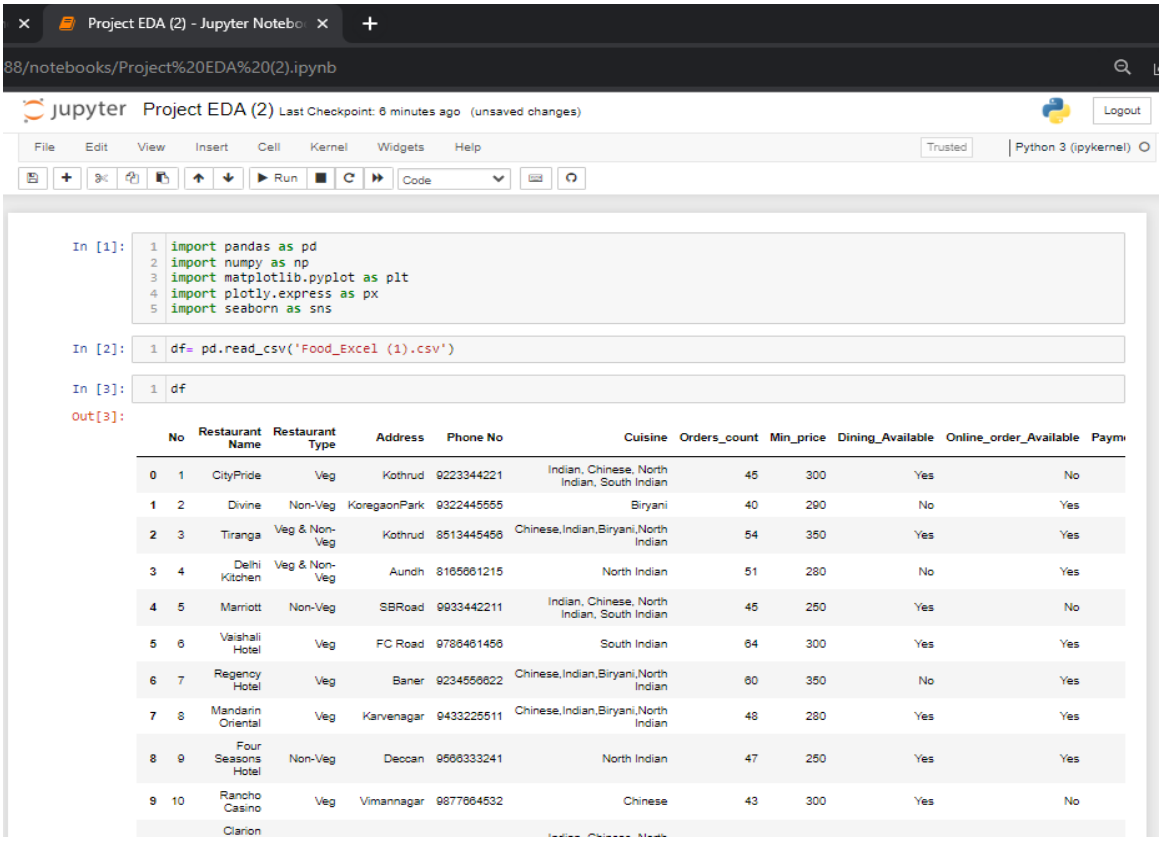
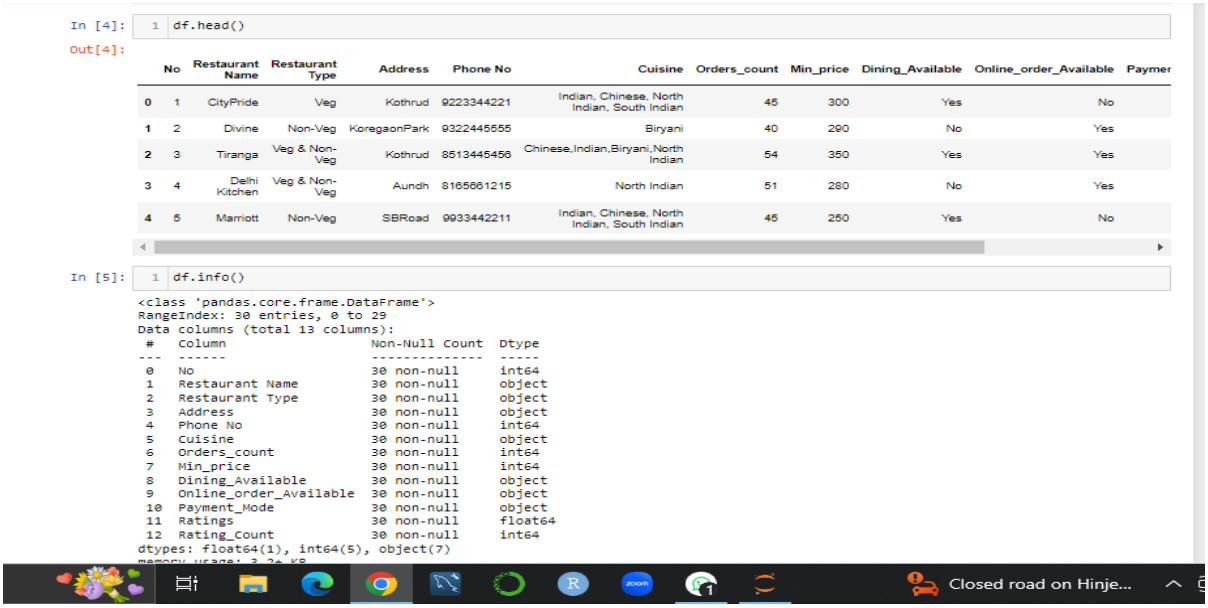
Project EDA



	No	Restaurant Name	Restaurant Type	Address	Phone No	Cuisine	Orders_count	Min_price	Dining_Available	Online_order_Available	Paym
0	1	CityPride	Veg	Kothrud	9223344221	Indian, Chinese, North Indian, South Indian	45	300	Yes	No	
1	2	Divine	Non-Veg	KoregaonPark	9322445555	Biryani	40	290	No	Yes	
2	3	Tiranga	Veg & Non-Veg	Kothrud	8513445456	Chinese,Indian,Biryani,North Indian	54	350	Yes	Yes	
3	4	Delhi Kitchen	Veg & Non-Veg	Aundh	8165661215	North Indian	51	280	No	Yes	
4	5	Marriott	Non-Veg	SBRoad	9933442211	Indian, Chinese, North Indian, South Indian	45	250	Yes	No	
5	6	Vaishali Hotel	Veg	FC Road	9786461456	South Indian	64	300	Yes	Yes	
6	7	Regency Hotel	Veg	Baner	9234556622	Chinese,Indian,Biryani,North Indian	60	350	No	Yes	
7	8	Mandarin Oriental	Veg	Karvenagar	9433225511	Chinese,Indian,Biryani,North Indian	48	280	Yes	Yes	
8	9	Four Seasons Hotel	Non-Veg	Deccan	9566333241	North Indian	47	250	Yes	Yes	
9	10	Rancho Casino	Veg	Vimannagar	9877664532	Chinese	43	300	Yes	No	
		Clarion									



	No	Restaurant Name	Restaurant Type	Address	Phone No	Cuisine	Orders_count	Min_price	Dining_Available	Online_order_Available	Paymer
0	1	CityPride	Veg	Kothrud	9223344221	Indian, Chinese, North Indian, South Indian	45	300	Yes	No	
1	2	Divine	Non-Veg	KoregaonPark	9322445555	Biryani	40	290	No	Yes	
2	3	Tiranga	Veg & Non-Veg	Kothrud	8513445456	Chinese,Indian,Biryani,North Indian	54	350	Yes	Yes	
3	4	Delhi Kitchen	Veg & Non-Veg	Aundh	8165661215	North Indian	51	280	No	Yes	
4	5	Marriott	Non-Veg	SBRoad	9933442211	Indian, Chinese, North Indian, South Indian	45	250	Yes	No	


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 13 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   No                      30 non-null     int64
 1   Restaurant Name         30 non-null     object
 2   Restaurant Type         30 non-null     object
 3   Address                 30 non-null     object
 4   Phone No                30 non-null     int64
 5   Cuisine                 30 non-null     object
 6   Orders_count            30 non-null     int64
 7   Min_price               30 non-null     int64
 8   Dining_Available        30 non-null     object
 9   Online_order_Available  30 non-null     object
 10  Payment_Mode            30 non-null     object
 11  Ratings                 30 non-null     float64
 12  Rating_Count            30 non-null     int64
dtypes: float64(1), int64(5), object(7)
memory usage: 3.2+ KB
```

```
dtypes: float64(1), int64(5), object(7)
memory usage: 3.2+ KB
```

In [6]:
```
null_values = df.isna().sum()
null_values
```

Out[6]:
```
No                          0
Restaurant Name             0
Restaurant Type             0
Address                     0
Phone No                    0
Cuisine                     0
Orders_count                0
Min_price                   0
Dining_Available            0
Online_order_Available      0
Payment_Mode                0
Ratings                     0
Rating_Count                0
dtype: int64
```

In [7]:
```
df.describe()
```

Out[7]:

	No	Phone No	Orders_count	Min_price	Ratings	Rating_Count
count	30.000000	3.000000e+01	30.000000	30.000000	30.000000	30.000000
mean	15.500000	8.296982e+09	48.833333	308.333333	3.720000	46.566667
std	8.803408	1.517605e+09	12.913728	49.902203	1.009404	13.174383
min	1.000000	4.484465e+09	23.000000	250.000000	1.000000	20.000000
25%	8.250000	7.470813e+09	40.750000	280.000000	3.000000	40.000000
50%	15.500000	8.463192e+09	47.500000	300.000000	4.000000	47.000000
75%	22.750000	9.516350e+09	58.750000	335.000000	4.375000	56.500000
max	30.000000	9.933442e+09	70.000000	450.000000	5.000000	70.000000

In [8]:
```
df[["Ratings","Rating_Count"]].describe().loc[["mean","min","max"]]
```


	No	Phone No	Orders_count	Min_price	Ratings	Rating_Count
min	1.000000	4.484465e+09	23.000000	250.000000	1.000000	20.000000
25%	8.250000	7.470813e+09	40.750000	280.000000	3.000000	40.000000
50%	15.500000	8.463192e+09	47.500000	300.000000	4.000000	47.000000
75%	22.750000	9.516350e+09	58.750000	335.000000	4.375000	56.500000
max	30.000000	9.933442e+09	70.000000	450.000000	5.000000	70.000000

In [8]:
```
df[["Ratings","Rating_Count"]].describe().loc[["mean","min","max"]]
```

Out[8]:

	Ratings	Rating_Count
mean	3.72	46.566667
min	1.00	20.000000
max	5.00	70.000000

In [9]:
```
df.shape
```

Out[9]:
```
(30, 13)
```

In [10]:
```
df.columns
```

Out[10]:
```
Index(['No', 'Restaurant Name', 'Restaurant Type', 'Address', 'Phone No',
       'Cuisine', 'Orders_count', 'Min_price', 'Dining_Available',
       'Online_order_Available', 'Payment_Mode', 'Ratings', 'Rating_Count'],
      dtype='object')
```

In [11]:
```
df['Dining_Available']
```

Out[11]:
```
0     Yes
1      No
2     Yes
3      No
4     Yes
5     Yes
6      No
7     Yes
```

```
In [12]:
df['Address'].unique()
```

```
Out[12]:
array(['Kothrud', 'KoregaonPark', 'Aundh', 'SBRoad', 'FC Road', 'Baner',
       'Karvenagar', 'Deccan', 'Vimannagar', 'Katraj', 'Swargate',
       'Hadapsar', 'Wakad', 'Erandwane', 'Kondhwa', 'Kalyani nagar',
       'Camp', 'Warje', 'PuneCity', 'ShivajiNagar', 'Kharadi',
       'Hinjawadi', 'Mahalunge', 'Sangamvadi'], dtype=object)
```

```
In [13]:
df['Address'].nunique()
```

```
Out[13]:
24
```

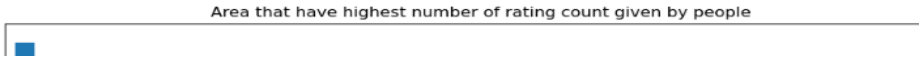
```
In [14]:
print(df["Cuisine"].nunique())
print(df["Cuisine"].unique())
```

```
7
['Indian, Chinese, North Indian, South Indian' 'Biryani'
 'Chinese,Indian,Biryani,North Indian' 'North Indian' 'South Indian'
 'Chinese' 'Fast Food, Chinese, Beverages']
```

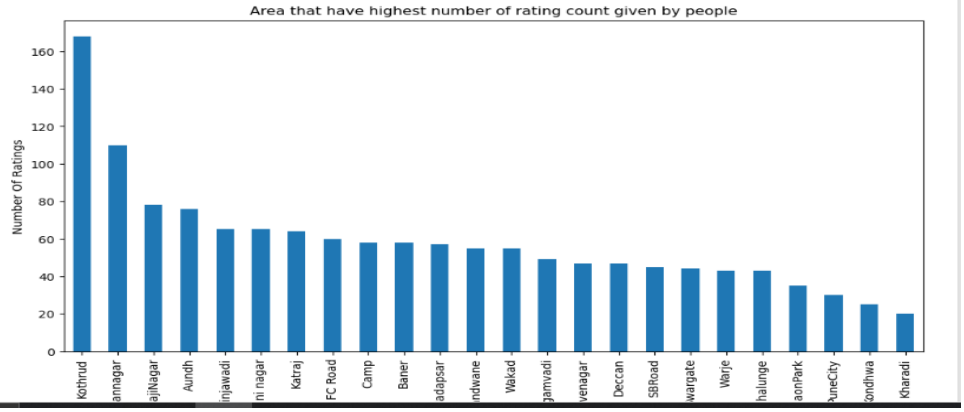
```
In [15]:
df["Min_price"].unique()
```

```
Out[15]:
array([300, 290, 350, 280, 250, 340, 380, 450, 400, 320, 260], dtype=int64)
```

```
In [16]:
# visualize Areas that have the highest number of rating count given by people

Highest_Area_Rating_Count = df.groupby('Address')['Rating_Count'].sum().sort_values(ascending=False)

plt.figure(figsize=(12,6))
Highest_Area_Rating_Count.plot.bar()
plt.xlabel('Area')
plt.ylabel('Number Of Ratings')
plt.title('Area that have highest number of rating count given by people')
plt.show();
```

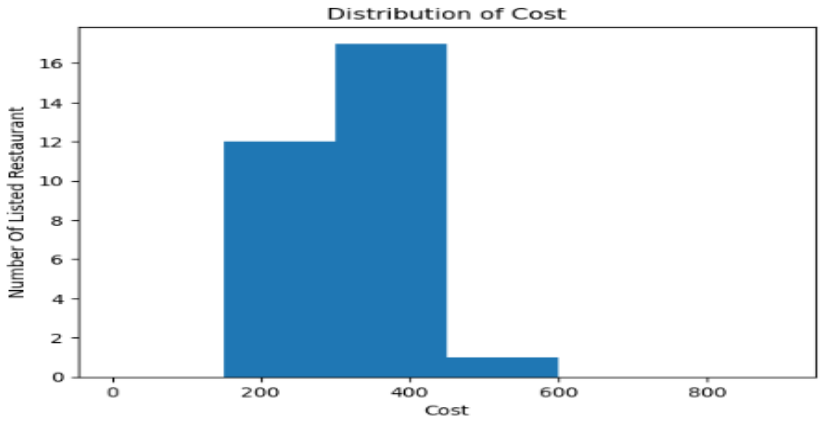


```
Highest_Area_Rating_Count = df.groupby('Address')['Rating_Count'].sum().sort_values(ascending=False)

plt.figure(figsize=(12,6))
Highest_Area_Rating_Count.plot.bar()
plt.xlabel('Area')
plt.ylabel('Number Of Ratings')
plt.title('Area that have highest number of rating count given by people')
plt.show();
```



In [17]:
```
#Plotting the distribution of cost

plt.hist(df['Min_price'], bins=np.arange(0, 1000, 150));
plt.title('Distribution of Cost')
plt.xlabel('Cost')
plt.ylabel('Number Of Listed Restaurant')
```

Out[17]: Text(0, 0.5, 'Number Of Listed Restaurant')

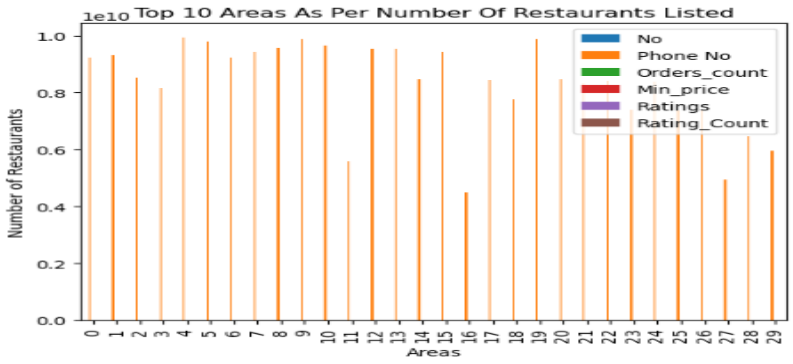


In [18]:
```
#What are the top 10 Areas as per the number of restaurants listed?

plt.figure(figsize=(12, 6))
top_rest=df.head(30).plot.bar()
```

Cost

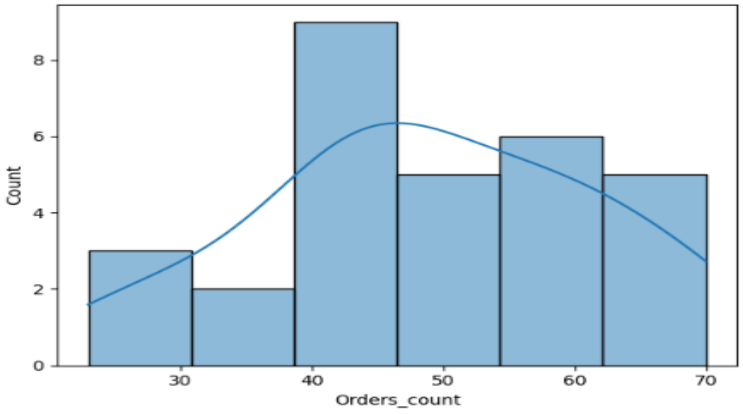
In [18]:
```
#What are the top 10 Areas as per the number of restaurants listed?

plt.figure(figsize=(12, 6))
top_rest=df.head(30).plot.bar()
plt.title('Top 10 Areas As Per Number Of Restaurants Listed')
plt.xlabel('Areas')
plt.ylabel('Number of Restaurants')
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```



In [19]:
```
sns.histplot(df.Orders_count,kde=True)
```

Out[19]: <Axes: xlabel='Orders_count', ylabel='Count'>



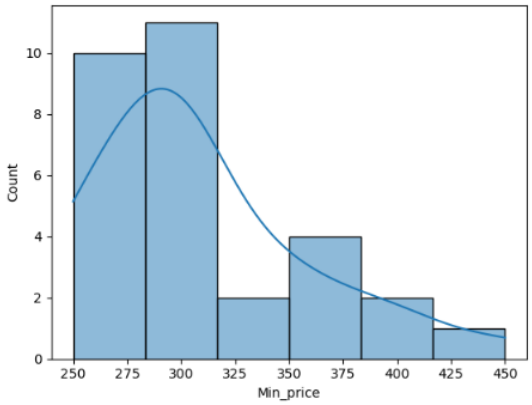
In [20]:
```
sns.histplot(df.Min_price,kde=True)
```

Out[20]: <Axes: xlabel='Min_price', ylabel='Count'>

Orders_count

In [20]:
```
1 sns.histplot(df.Min_price,kde=True)
```

Out[20]: <Axes: xlabel='Min_price', ylabel='Count'>

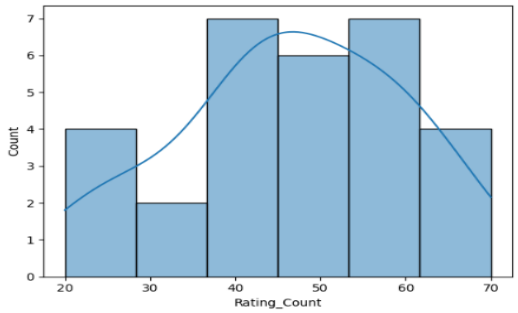


In [21]:
```
1 sns.histplot(df.Rating_Count,kde=True)
```

Out[21]: <Axes: xlabel='Rating_Count', ylabel='Count'>

In [21]:
```
1 sns.histplot(df.Rating_Count,kde=True)
```

Out[21]: <Axes: xlabel='Rating_Count', ylabel='Count'>



In [22]:
```
1 sns.countplot(df.Rating_Count)
```

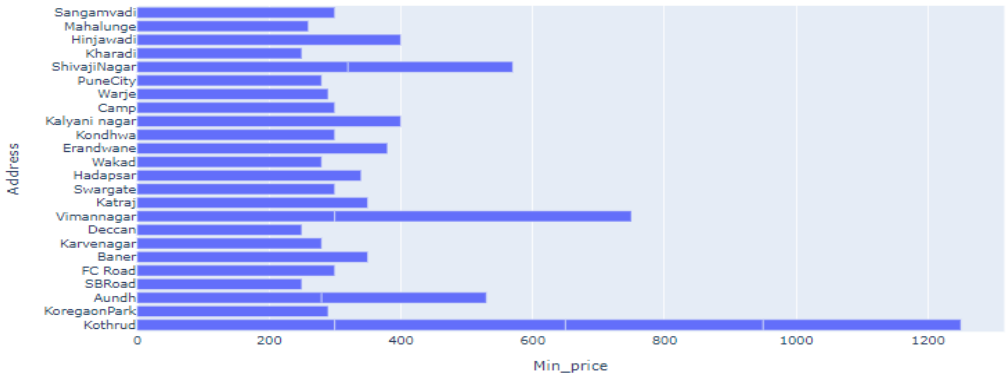
Out[22]: <Axes: ylabel='count'>

In [27]:

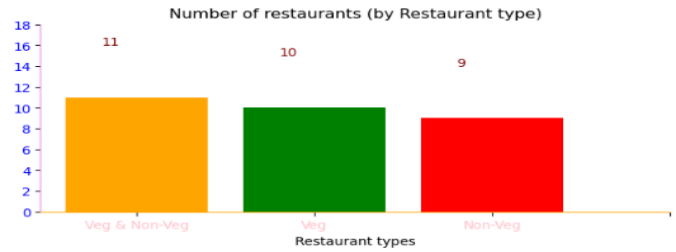
```
fig = px.scatter(df,
                 x = 'Min_price',
                 y = 'Rating_Count',
                 size = 'Min_price',
                 title = 'Areas That Has Most Expensive Food In Their Restaurants',
                 color = 'Ratings',
                 hover_data = {'Address': True})
fig.update_yaxes(title_text='Number Of Ratings Counts')
fig.update_xaxes(title_text='Cost')
fig.update_layout(width=800, height=600)
fig.show()
```

In [28]:

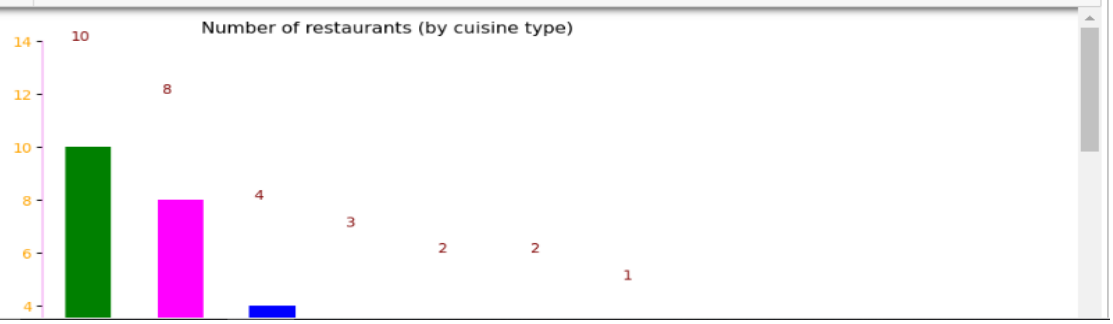
```
fig = px.bar(df.head(30),
             x = 'Min_price',
             y = 'Address',
             barmode = 'relative',
             title = 'Highest Cost Of Food In Every Address')
fig.show()
```



```
Rest_type_count = df.groupby("Restaurant Type").count()["No"].sort_values(ascending=False)[:3]

fig = plt.figure(figsize=[8,3], frameon=False)
ax = fig.gca()
ax.spines["top"].set_visible(False)
ax.spines["right"].set_visible(False)
ax.spines["left"].set_color("violet")
ax.spines["bottom"].set_color("orange")

colors = ["orange",'green',"red"]
plt.bar(Rest_type_count.index, Rest_type_count.values, color=colors)

plt.xticks(range(0, 4), color="pink")
plt.yticks(range(0, 20, 2), color="blue")
plt.xlabel("Restaurant types")

for i, v in enumerate(Rest_type_count):
    ax.text(i-0.2, v+5, str(v), color='maroon')
plt.title("Number of restaurants (by Restaurant type)")

plt.show()
```



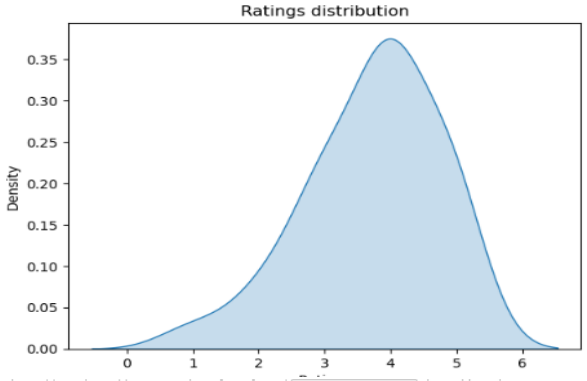

```
c_count = df.groupby('Cuisine').count()["No"].sort_values(ascending=False)[:7]

fig = plt.figure(figsize=[8,5], frameon=False)
ax = fig.gca()
ax.spines["top"].set_visible(False)
ax.spines["right"].set_visible(False)
ax.spines["left"].set_color("violet")
ax.spines["bottom"].set_color("orange")

colors = ['green', 'magenta', 'blue', 'cyan','gray','yellow','red']
c_count.plot.bar(color=colors)

plt.xticks(range(0, 8), color="violet")
plt.yticks(range(0, 16, 2), color="orange")
plt.xlabel("Top cuisines")

for i, v in enumerate(c_count):
    ax.text(i-0.2, v+4, str(v), color='maroon')
plt.title("Number of restaurants (by cuisine type)")

plt.show()
```



In [31]:

```
#Ratings distribution

sns.kdeplot(df['Ratings'], shade=True)
plt.title("Ratings distribution")
plt.show()
```

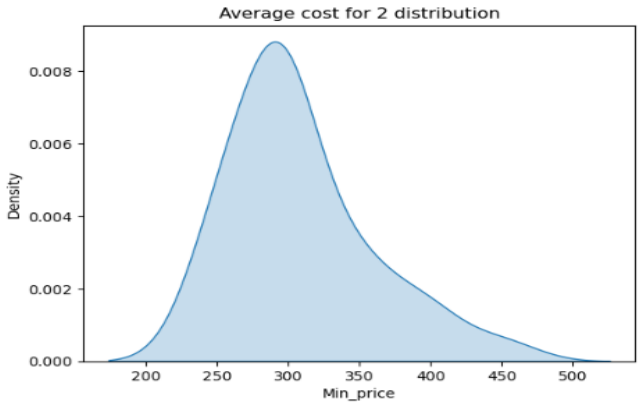
```
C:\Users\ADU7131\AppData\Local\Temp\ipykernel_6628\540300487.py:3: FutureWarning:


`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.
```



In [32]:

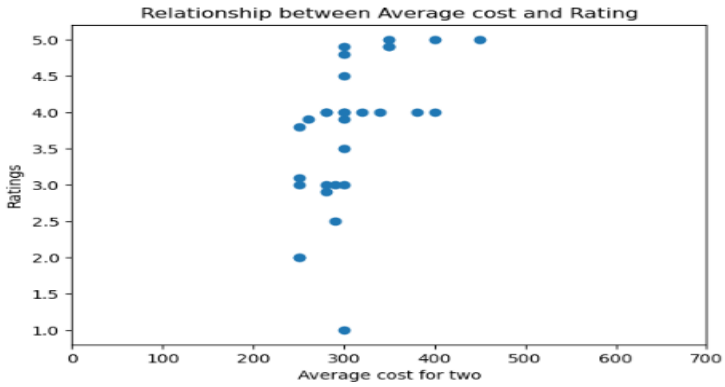
```
#Avergae cost for two distribution

sns.kdeplot(df['Min_price'], shade=True)
plt.title("Average cost for 2 distribution")
plt.show()
```

```
C:\Users\ADU7131\AppData\Local\Temp\ipykernel_6628\2014962385.py:3: FutureWarning:


`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.
```



Min_price

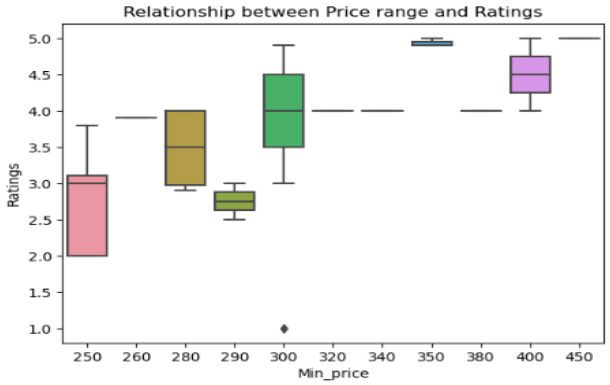
In [33]:
```
#Relation between Average price for two and Ratings

plt.plot("Min_price","Ratings", data=df, linestyle="none", marker="o")
plt.xlim([0,700])
plt.title("Relationship between Average cost and Rating")
plt.xlabel("Average cost for two")
plt.ylabel("Ratings")
plt.show()
```



In [34]:
```
sns.boxplot(x='Min_price', y='Ratings', data=df)
plt.ylim(0.8)
plt.title("Relationship between Price range and Ratings")
plt.show()
```

0 100 200 300 400 500 600 700
Average cost for two

In [34]:
```
sns.boxplot(x='Min_price', y='Ratings', data=df)
plt.ylim(0.8)
plt.title("Relationship between Price range and Ratings")
plt.show()
```



In [ ]:
```
1
```