Chennai Dining Trends - A Swiggy Data Analysis

An Exploratory Data Analysis Project by Ambika V

Objective:

Analyze Chennai restaurant data to uncover performance patterns and provide actionable recommendations for improving customer satisfaction and operational efficiency.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

Swiggy Chennai Dataset

In [2]:
```
df = pd.read_csv(r"C:\Ambika\Data Science\Datasets\Swiggy chennai dataset.csv")
```

In [3]:
```
df
```

Out[3]:

	city	subcity	restaurant	rating	rating count	cost	cuisine	menu	item	pr
0	Chennai	Poonamallee	LASSI AND SOUP CORNER	4.1	Too Few Ratings	200	Juices	Falooda	Regular Falooda	10
1	Chennai	Poonamallee	LASSI AND SOUP CORNER	4.1	Too Few Ratings	200	Juices	Falooda	Vanilla Falooda	11
2	Chennai	Poonamallee	LASSI AND SOUP CORNER	4.1	Too Few Ratings	200	Juices	Falooda	Strawberry Falooda	11
3	Chennai	Poonamallee	LASSI AND SOUP CORNER	4.1	Too Few Ratings	200	Juices	Falooda	Chocolate Falooda	13
4	Chennai	Poonamallee	LASSI AND SOUP CORNER	4.1	Too Few Ratings	200	Juices	Falooda	Butterscotch Falooda	13
...	...	...	...	...	...	...	...	...	...	
162515	Chennai	Purasawalkam	Jain Food Service Chennai	2.8	20+ ratings	400	Thalis	North Indian	Poha	13
162516	Chennai	Purasawalkam	Jain Food Service Chennai	2.8	20+ ratings	400	Thalis	North Indian	Upma	10
162517	Chennai	Purasawalkam	Jain Food Service Chennai	2.8	20+ ratings	400	Thalis	North Indian	North India Thali	17
162518	Chennai	Purasawalkam	Jain Food Service Chennai	2.8	20+ ratings	400	Thalis	North Indian	Aloo Parathas	13
162519	Chennai	Purasawalkam	Jain Food Service Chennai	2.8	20+ ratings	400	Thalis	North Indian	Tomato Rice	7

162520 rows × 11 columns

1. Data Inspection

In [4]:
```
df.shape
```

Out[4]:
```
(162520, 11)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 162520 entries, 0 to 162519
Data columns (total 11 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   city            162520 non-null  object 
 1   subcity         162520 non-null  object 
 2   restaurant      162520 non-null  object 
 3   rating          162520 non-null  float64
 4   rating count    162520 non-null  object 
 5   cost            162520 non-null  int64  
 6   cuisine         162520 non-null  object 
 7   menu            162520 non-null  object 
 8   item            162520 non-null  object 
 9   price           4949 non-null    float64
 10  veg_or_non_veg  162520 non-null  object 
dtypes: float64(2), int64(1), object(8)
memory usage: 13.6+ MB
```

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
city                  0
subcity               0
restaurant            0
rating                0
rating count          0
cost                  0
cuisine               0
menu                  0
item                  0
price            157571
veg_or_non_veg        0
dtype: int64
```

2. Data Cleaning

Handling Missing Values

In [7]:
```
df.drop(columns=['price'], inplace=True)
```

In [8]:
```
df
```

Out[8]:

	city	subcity	restaurant	rating	rating count	cost	cuisine	menu	item	ve
0	Chennai	Poonamallee	LASSI AND SOUP CORNER	4.1	Too Few Ratings	200	Juices	Falooda	Regular Falooda	
1	Chennai	Poonamallee	LASSI AND SOUP CORNER	4.1	Too Few Ratings	200	Juices	Falooda	Vanilla Falooda	
2	Chennai	Poonamallee	LASSI AND SOUP CORNER	4.1	Too Few Ratings	200	Juices	Falooda	Strawberry Falooda	
3	Chennai	Poonamallee	LASSI AND SOUP CORNER	4.1	Too Few Ratings	200	Juices	Falooda	Chocolate Falooda	
4	Chennai	Poonamallee	LASSI AND SOUP CORNER	4.1	Too Few Ratings	200	Juices	Falooda	Butterscotch Falooda	
...	...	...	...	...	...	...	...	...	...	
162515	Chennai	Purasawalkam	Jain Food Service Chennai	2.8	20+ ratings	400	Thalis	North Indian	Poha	
162516	Chennai	Purasawalkam	Jain Food Service Chennai	2.8	20+ ratings	400	Thalis	North Indian	Upma	
162517	Chennai	Purasawalkam	Jain Food Service Chennai	2.8	20+ ratings	400	Thalis	North Indian	North India Thali	
162518	Chennai	Purasawalkam	Jain Food Service Chennai	2.8	20+ ratings	400	Thalis	North Indian	Aloo Parathas	
162519	Chennai	Purasawalkam	Jain Food Service Chennai	2.8	20+ ratings	400	Thalis	North Indian	Tomato Rice	

162520 rows × 10 columns

Dropping unwanted columns

In [9]:
```
df.drop(columns=['menu', 'item'], inplace=True)
```

In [10]:
```
df
```

Out[10]:

	city	subcity	restaurant	rating	rating count	cost	cuisine	veg_or_non_veg
0	Chennai	Poonamallee	LASSI AND SOUP CORNER	4.1	Too Few Ratings	200	Juices	Veg
1	Chennai	Poonamallee	LASSI AND SOUP CORNER	4.1	Too Few Ratings	200	Juices	Veg
2	Chennai	Poonamallee	LASSI AND SOUP CORNER	4.1	Too Few Ratings	200	Juices	Veg
3	Chennai	Poonamallee	LASSI AND SOUP CORNER	4.1	Too Few Ratings	200	Juices	Veg
4	Chennai	Poonamallee	LASSI AND SOUP CORNER	4.1	Too Few Ratings	200	Juices	Veg
...	...	...	...	...	...	...	...	...
162515	Chennai	Purasawalkam	Jain Food Service Chennai	2.8	20+ ratings	400	Thalis	Veg
162516	Chennai	Purasawalkam	Jain Food Service Chennai	2.8	20+ ratings	400	Thalis	Veg
162517	Chennai	Purasawalkam	Jain Food Service Chennai	2.8	20+ ratings	400	Thalis	Veg
162518	Chennai	Purasawalkam	Jain Food Service Chennai	2.8	20+ ratings	400	Thalis	Veg
162519	Chennai	Purasawalkam	Jain Food Service Chennai	2.8	20+ ratings	400	Thalis	Veg

162520 rows × 8 columns

Standardising Text Data

In [11]:
```
df.columns = df.columns.str.lower().str.strip()
```

In [12]:
```
df = df.apply(lambda x: x.str.lower().str.strip() if x.dtype == "object" else x)
```

In [13]:
```
df
```

Out[13]:

	city	subcity	restaurant	rating	rating count	cost	cuisine	veg_or_non_veg
0	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
1	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
2	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
3	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
4	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
...	...	...	...	...	...	...	...	...
162515	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg
162516	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg
162517	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg
162518	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg
162519	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg

162520 rows × 8 columns

Handling Inconsistent Data

In [14]:
```
df['cost'] = df.groupby(['subcity','restaurant'])['cost'].transform('max')
```

In [15]:
```
df
```

Out[15]:

	city	subcity	restaurant	rating	rating count	cost	cuisine	veg_or_non_veg
0	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
1	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
2	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
3	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
4	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
...	...	...	...	...	...	...	...	...
162515	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg
162516	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg
162517	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg
162518	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg
162519	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg

162520 rows × 8 columns

In [16]:
```
df['food_category'] = df.groupby(['subcity','restaurant'])['veg_or_non_veg'].transf
```

In [17]:
```
df
```

Out[17]:

	city	subcity	restaurant	rating	rating count	cost	cuisine	veg_or_non_veg	food_ca
0	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg	
1	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg	
2	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg	
3	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg	
4	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg	
...	...	...	...	...	...	...	...	...	
162515	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg	
162516	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg	
162517	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg	
162518	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg	
162519	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg	

162520 rows × 9 columns

In [18]:
```
df.drop(columns=["veg_or_non_veg"],inplace = True)
```

In [19]:
```
df
```

Out[19]:

	city	subcity	restaurant	rating	rating count	cost	cuisine	food_category
0	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
1	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
2	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
3	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
4	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
...	...	...	...	...	...	...	...	...
162515	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg
162516	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg
162517	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg
162518	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg
162519	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg

162520 rows × 8 columns

Exploding columns with multiple values

In [20]:
```
df.cuisine.value_counts()
```

Out[20]:
```
south indian                 8051
north indian,chinese         6341
indian                       6210
south indian,north indian    5618
biryani,chinese              5412
                             ...
street food,burmese             1
grill,continental               1
barbecue,tandoor                1
tibetan,north eastern           1
healthy food                    1
Name: cuisine, Length: 643, dtype: int64
```

In [21]:
```
df['cuisine'] = df['cuisine'].str.split(',')
df = df.explode('cuisine')
df['cuisine'] = df['cuisine'].str.strip()
```

In [22]:
```
df
```

Out[22]:

	city	subcity	restaurant	rating	rating count	cost	cuisine	food_category
0	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
1	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
2	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
3	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
4	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
...	...	...	...	...	...	...	...	...
162515	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg
162516	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg
162517	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg
162518	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg
162519	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg

291298 rows × 8 columns

In [23]:
```
df.cuisine.value_counts()
```

Out[23]:
```
chinese           43020
south indian      36527
north indian      28752
biryani           28035
indian            23444
                  ...
middle eastern       12
north eastern        10
naga                  9
keto                  4
goan                  4
Name: cuisine, Length: 75, dtype: int64
```

Checking for Duplicate Records

In [24]:
```
df.duplicated().sum()
```

Out[24]:
```
279950
```

In [25]:
```
df.drop_duplicates(inplace = True)
```

In [26]:
```
df
```

Out[26]:

	city	subcity	restaurant	rating	rating count	cost	cuisine	food_category
0	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
23	chennai	poonamallee	biriyani chowk	4.1	100+ ratings	320	indian	non veg
23	chennai	poonamallee	biriyani chowk	4.1	100+ ratings	320	chinese	non veg
69	chennai	poonamallee	lakshmi restaurant	3.7	50+ ratings	350	indian	veg
72	chennai	poonamallee	1 point 2 devi mess	2.9	100+ ratings	250	south indian	non veg
...	...	...	...	...	...	...	...	...
162436	chennai	purasawalkam	tanu soup corner / a-1 chicken center	4.1	too few ratings	200	chinese	non veg
162436	chennai	purasawalkam	tanu soup corner / a-1 chicken center	4.1	too few ratings	200	south indian	non veg
162442	chennai	purasawalkam	hotel farook	4.3	1k+ ratings	450	south indian	non veg
162442	chennai	purasawalkam	hotel farook	4.3	1k+ ratings	450	arabian	non veg
162508	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg

11348 rows × 8 columns

Resetting Index

In [27]:
```
df.reset_index(drop = True,inplace = True)
```

In [28]:
```
df
```

Out[28]:

	city	subcity	restaurant	rating	rating count	cost	cuisine	food_category
0	chennai	poonamallee	lassi and soup corner	4.1	too few ratings	200	juices	veg
1	chennai	poonamallee	biriyani chowk	4.1	100+ ratings	320	indian	non veg
2	chennai	poonamallee	biriyani chowk	4.1	100+ ratings	320	chinese	non veg
3	chennai	poonamallee	lakshmi restaurant	3.7	50+ ratings	350	indian	veg
4	chennai	poonamallee	1 point 2 devi mess	2.9	100+ ratings	250	south indian	non veg
...	...	...	...	...	...	...	...	...
11343	chennai	purasawalkam	tanu soup corner / a-1 chicken center	4.1	too few ratings	200	chinese	non veg
11344	chennai	purasawalkam	tanu soup corner / a-1 chicken center	4.1	too few ratings	200	south indian	non veg
11345	chennai	purasawalkam	hotel farook	4.3	1k+ ratings	450	south indian	non veg
11346	chennai	purasawalkam	hotel farook	4.3	1k+ ratings	450	arabian	non veg
11347	chennai	purasawalkam	jain food service chennai	2.8	20+ ratings	400	thalis	veg

11348 rows × 8 columns

3. Data Transformation

Adding calculated columns and categorical columns

1. rating count

In [29]:
```
def convert_rating_count(value):
    if value == 'too few ratings':
        return 10
    elif value == '20+ ratings':
        return 20
    elif value == '50+ ratings':
        return 50
    elif value == '100+ ratings':
        return 100
    elif value == '500+ ratings':
        return 500
    elif value == '1k+ ratings':
        return 1000
    elif value == '5k+ ratings':
        return 5000
    else:
        return None
```

```
df['rating count'] = df['rating count'].apply(convert_rating_count)
```

In [30]:
```
df
```

Out[30]:

	city	subcity	restaurant	rating	rating count	cost	cuisine	food_category
0	chennai	poonamallee	lassi and soup corner	4.1	10	200	juices	veg
1	chennai	poonamallee	biriyani chowk	4.1	100	320	indian	non veg
2	chennai	poonamallee	biriyani chowk	4.1	100	320	chinese	non veg
3	chennai	poonamallee	lakshmi restaurant	3.7	50	350	indian	veg
4	chennai	poonamallee	1 point 2 devi mess	2.9	100	250	south indian	non veg
...	...	...	...	...	...	...	...	...
11343	chennai	purasawalkam	tanu soup corner / a-1 chicken center	4.1	10	200	chinese	non veg
11344	chennai	purasawalkam	tanu soup corner / a-1 chicken center	4.1	10	200	south indian	non veg
11345	chennai	purasawalkam	hotel farook	4.3	1000	450	south indian	non veg
11346	chennai	purasawalkam	hotel farook	4.3	1000	450	arabian	non veg
11347	chennai	purasawalkam	jain food service chennai	2.8	20	400	thalis	veg

11348 rows × 8 columns

2. rating_category

In [31]:
```
bins = [0, 1, 2, 3, 4, 5]
labels = ['very low', 'low', 'medium', 'high', 'very high']
df['rating_category'] = pd.cut(df['rating'], bins=bins, labels=labels, include_lowe
```

In [32]:
```
df
```

Out[32]:

	city	subcity	restaurant	rating	rating count	cost	cuisine	food_category	rating_cate
0	chennai	poonamallee	lassi and soup corner	4.1	10	200	juices	veg	very
1	chennai	poonamallee	biriyani chowk	4.1	100	320	indian	non veg	very
2	chennai	poonamallee	biriyani chowk	4.1	100	320	chinese	non veg	very
3	chennai	poonamallee	lakshmi restaurant	3.7	50	350	indian	veg	
4	chennai	poonamallee	1 point 2 devi mess	2.9	100	250	south indian	non veg	me
...	...	...	...	...	...	...	...	...	
11343	chennai	purasawalkam	tanu soup corner / a-1 chicken center	4.1	10	200	chinese	non veg	very
11344	chennai	purasawalkam	tanu soup corner / a-1 chicken center	4.1	10	200	south indian	non veg	very
11345	chennai	purasawalkam	hotel farook	4.3	1000	450	south indian	non veg	very
11346	chennai	purasawalkam	hotel farook	4.3	1000	450	arabian	non veg	very
11347	chennai	purasawalkam	jain food service chennai	2.8	20	400	thalis	veg	me

11348 rows × 9 columns

3. cost_category

In [33]:
```
bins = [0, 499, 999, 1499, df['cost'].max()]
labels = [
    'Budget-Friendly',
    'Standard Dining',
    'Premium Dining',
    'Gourmet']
df['cost_category'] = pd.cut(df['cost'], bins=bins, labels=labels, include_lowest=T
```

In [34]:
```
df
```

Out[34]:

	city	subcity	restaurant	rating	rating count	cost	cuisine	food_category	rating_cate
0	chennai	poonamallee	lassi and soup corner	4.1	10	200	juices	veg	very
1	chennai	poonamallee	biriyani chowk	4.1	100	320	indian	non veg	very
2	chennai	poonamallee	biriyani chowk	4.1	100	320	chinese	non veg	very
3	chennai	poonamallee	lakshmi restaurant	3.7	50	350	indian	veg	
4	chennai	poonamallee	1 point 2 devi mess	2.9	100	250	south indian	non veg	me
...	...	...	...	...	...	...	...	...	
11343	chennai	purasawalkam	tanu soup corner / a-1 chicken center	4.1	10	200	chinese	non veg	very
11344	chennai	purasawalkam	tanu soup corner / a-1 chicken center	4.1	10	200	south indian	non veg	very
11345	chennai	purasawalkam	hotel farook	4.3	1000	450	south indian	non veg	very
11346	chennai	purasawalkam	hotel farook	4.3	1000	450	arabian	non veg	very
11347	chennai	purasawalkam	jain food service chennai	2.8	20	400	thalis	veg	me

11348 rows × 10 columns

In [35]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11348 entries, 0 to 11347
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype   
---  ------           --------------  -----   
 0   city             11348 non-null  object  
 1   subcity          11348 non-null  object  
 2   restaurant       11348 non-null  object  
 3   rating           11348 non-null  float64 
 4   rating count     11348 non-null  int64   
 5   cost             11348 non-null  int64   
 6   cuisine          11348 non-null  object  
 7   food_category    11348 non-null  object  
 8   rating_category  11348 non-null  category
 9   cost_category    11348 non-null  category
dtypes: category(2), float64(1), int64(2), object(5)
memory usage: 731.9+ KB
```

In [36]:
```
df['cuisine_count'] = df.groupby(["subcity","restaurant"]).cuisine.transform('nunic
```

In [37]:
```
df
```

Out[37]:

	city	subcity	restaurant	rating	rating count	cost	cuisine	food_category	rating_cate
0	chennai	poonamallee	lassi and soup corner	4.1	10	200	juices	veg	very
1	chennai	poonamallee	biriyani chowk	4.1	100	320	indian	non veg	very
2	chennai	poonamallee	biriyani chowk	4.1	100	320	chinese	non veg	very
3	chennai	poonamallee	lakshmi restaurant	3.7	50	350	indian	veg	
4	chennai	poonamallee	1 point 2 devi mess	2.9	100	250	south indian	non veg	me
...	...	...	...	...	...	...	...	...	
11343	chennai	purasawalkam	tanu soup corner / a-1 chicken center	4.1	10	200	chinese	non veg	very
11344	chennai	purasawalkam	tanu soup corner / a-1 chicken center	4.1	10	200	south indian	non veg	very
11345	chennai	purasawalkam	hotel farook	4.3	1000	450	south indian	non veg	very
11346	chennai	purasawalkam	hotel farook	4.3	1000	450	arabian	non veg	very
11347	chennai	purasawalkam	jain food service chennai	2.8	20	400	thalis	veg	me

11348 rows × 11 columns

In [38]:
```
df["cuisine_count"].value_counts()
```

Out[38]:
```
2    9704
1    1638
3       6
Name: cuisine_count, dtype: int64
```

Exploratory Data Analysis

Cleaned and Transformed Dataset

- Dataset covers 11,348 rows and 12 columns
- 100% clean: no nulls
- Derived fields added: cost category, cuisine count, overall rating band

In [39]:
```
df
```

Out[39]:

	city	subcity	restaurant	rating	rating count	cost	cuisine	food_category	rating_cate
0	chennai	poonamallee	lassi and soup corner	4.1	10	200	juices	veg	very
1	chennai	poonamallee	biriyani chowk	4.1	100	320	indian	non veg	very
2	chennai	poonamallee	biriyani chowk	4.1	100	320	chinese	non veg	very
3	chennai	poonamallee	lakshmi restaurant	3.7	50	350	indian	veg	
4	chennai	poonamallee	1 point 2 devi mess	2.9	100	250	south indian	non veg	me
...	...	...	...	...	...	...	...	...	
11343	chennai	purasawalkam	tanu soup corner / a-1 chicken center	4.1	10	200	chinese	non veg	very
11344	chennai	purasawalkam	tanu soup corner / a-1 chicken center	4.1	10	200	south indian	non veg	very
11345	chennai	purasawalkam	hotel farook	4.3	1000	450	south indian	non veg	very
11346	chennai	purasawalkam	hotel farook	4.3	1000	450	arabian	non veg	very
11347	chennai	purasawalkam	jain food service chennai	2.8	20	400	thalis	veg	me

11348 rows × 11 columns

In [40]:
```
df.to_csv('swiggy_cleaned_df.csv', index=False)
```

In [41]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11348 entries, 0 to 11347
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   city             11348 non-null  object
 1   subcity          11348 non-null  object
 2   restaurant       11348 non-null  object
 3   rating           11348 non-null  float64
 4   rating count     11348 non-null  int64
 5   cost             11348 non-null  int64
 6   cuisine          11348 non-null  object
 7   food_category    11348 non-null  object
 8   rating_category  11348 non-null  category
 9   cost_category    11348 non-null  category
 10  cuisine_count    11348 non-null  int64
dtypes: category(2), float64(1), int64(3), object(5)
memory usage: 820.6+ KB
```

In [42]:
```
df.isnull().sum()
```

Out[42]:
```
city              0
subcity           0
restaurant        0
rating            0
rating count      0
cost              0
cuisine           0
food_category     0
rating_category   0
cost_category     0
cuisine_count     0
dtype: int64
```

In [43]:
```
df.duplicated().sum()
```

Out[43]:
```
0
```

Question 1: Which Chennai subcity has the highest and lowest average restaurant rating?

Approach:

To identify regional satisfaction trends, I calculated the average rating of restaurants in each subcity using groupby(), then visualized the result using a horizontal bar chart (Matplotlib). A vertical reference line at 4.0 rating helps highlight performance against a standard satisfaction threshold.

In [44]:
```
avg_rating = df.groupby('subcity')['rating'].mean().sort_values()
print(avg_rating)
```

```
subcity
omr perungudi          3.957426
urapakkam              3.964823
perumbakkam            3.966000
omr navalur            3.971685
mahindra world city    3.977027
karappakam             3.983838
poonamallee            3.991414
iyyapanthangal         3.998802
avadi                  4.007330
medavakkam             4.012780
ambattur               4.027763
porur                  4.030931
vadapalani             4.036887
velachery              4.037609
mogappair              4.049244
george town            4.050166
tambaram               4.050824
villivakkam            4.051471
madipakkam             4.054132
padur                  4.055030
perambur               4.057191
chrompet               4.061830
ramapuram              4.067606
guindy                 4.068246
purasawalkam           4.070776
ecr-kottivakkam        4.078241
arumbakkam             4.078539
egmore                 4.093529
alwarpet               4.100000
royapuram              4.100000
kilpauk                4.106122
annanagar              4.106579
kk nagar               4.116216
t nagar                4.126768
royapettah             4.153488
mylapore               4.159524
nungambakkam           4.161194
adyar                  4.162187
Name: rating, dtype: float64
```

In [45]:

```
plt.figure(figsize=(10,6))
avg_rating.plot(kind='barh', color='teal')
plt.title('Average Restaurant Rating by Subcity')
plt.xlabel('Average Rating')
plt.axvline(4.0, color='red', linestyle='--', label='4 Rating Threshold')
plt.legend()
plt.tight_layout()
```

Insight:

- Places like Adyar, Nungambakkam, and Mylapore have the highest average ratings over 4.3 rating.
- On the other hand, OMR Perungudi, Urapakkam, and Perumbakkam are on the lower end, just under 4.0 rating.
- This tells us that restaurants in central Chennai are generally doing better in customer satisfaction than some of the newer or outer subcities.

Recommendation:

- Focus on expanding partnerships or offers in the top-rated areas, since customers are already happy there it is easier to grow where the experience is strong.
- At the same time, Swiggy can dig deeper into the lower-rated zones by sending surveys, checking delivery wait times, or by analysing complaints.
- These lower-rated areas might benefit from a targeted quality push by onboarding new, better-performing restaurants.

Question 2: Does cost influence restaurant rating across subcities?

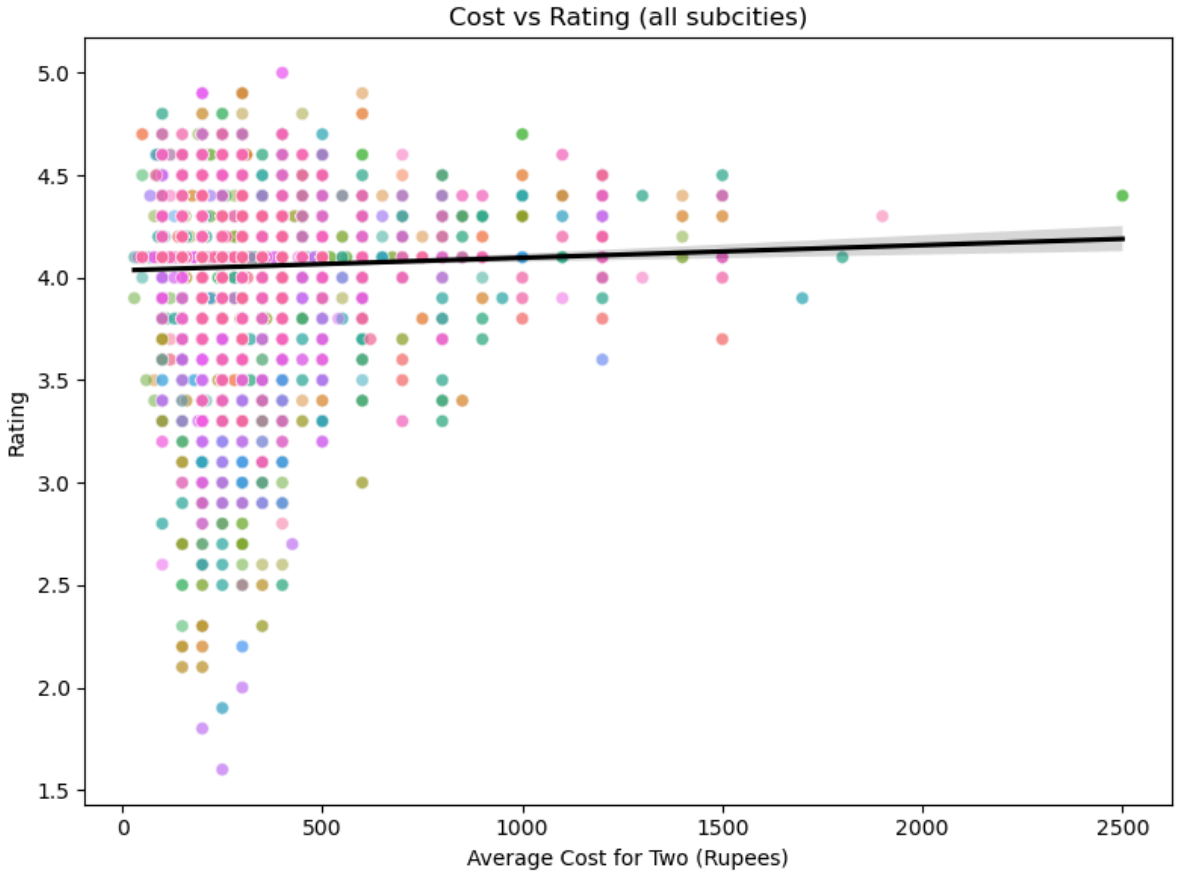
Approach:

I plotted a scatter chart with Seaborn to see if there's a relationship between a restaurant's cost and its average rating. I added a regression line for trend visibility and calculated Spearman's rank correlation.

In [46]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data=df, x='cost', y='rating', hue='subcity', alpha=0.5, legend=Fal
sns.regplot(data=df, x='cost', y='rating', scatter=False, color='black')
plt.title('Cost vs Rating (all subcities)')
```

```
plt.xlabel('Average Cost for Two (Rupees)')
plt.ylabel('Rating')
plt.tight_layout()

rho = df['rating'].corr(df['cost'], method='spearman')
print(f"Spearman correlation (cost & rating): {rho:.2f}")
```

```
Spearman correlation (cost & rating): -0.02
```



Insight:

- The correlation is nearly zero (–0.02), meaning there's no meaningful relationship between how much a restaurant charges and how well it's rated.
- The chart confirms that both low-cost and high-cost restaurants can be rated anywhere from 3 to 5.
- This suggests that price doesn't drive satisfaction, customers care about something else (taste, delivery time, packaging, etc.).

Recommendation:

- Swiggy should not assume that premium pricing leads to better customer experience.
- Instead, focus on actual rating metrics when promoting restaurants like highlighting "High Rated under ₹300" or "Best value in your area" to connect with cost-conscious users.
- For restaurant onboarding, don't filter by price band instead use quality indicators like rating count, cuisine consistency, or customer comments.

Question 3: Which cuisines dominate the 4.5+ rating segment?

Approach:

I filtered the dataset to include only restaurants with a rating of 4.5 or higher. Then, I counted how often each cuisine appeared in that top-rated segment and visualized the top 10 cuisines using a horizontal Plotly bar chart.

In [47]:
```
top_cuisine = (
    df[df['rating'] >= 4.5]['cuisine']
    .value_counts()
    .head(10)
    .sort_values()
)
print(top_cuisine)
```

```
juices          16
indian          21
sweets          25
chinese         27
snacks          42
north indian    43
bakery          44
desserts        52
beverages       58
south indian    68
Name: cuisine, dtype: int64
```

In [48]:
```
fig = px.bar(
    top_cuisine,
    orientation='h',
    title='Top Cuisines in 4.5+ Rating Restaurants',
    labels={'value':'Number of Restaurants', 'index':'Cuisine'}
)
fig.update_layout(yaxis_title='', xaxis_title='Count')
fig.show()
```

Insight:

- South Indian cuisine leads the pack among top-rated restaurants, followed by Beverages, Desserts, and Bakery.
- Classic categories like North Indian, Chinese, and Snacks are present, but slightly less dominant.
- The result shows that light, local, and simple options like South Indian meals and Beverages tend to receive stronger customer satisfaction.

Recommendation:

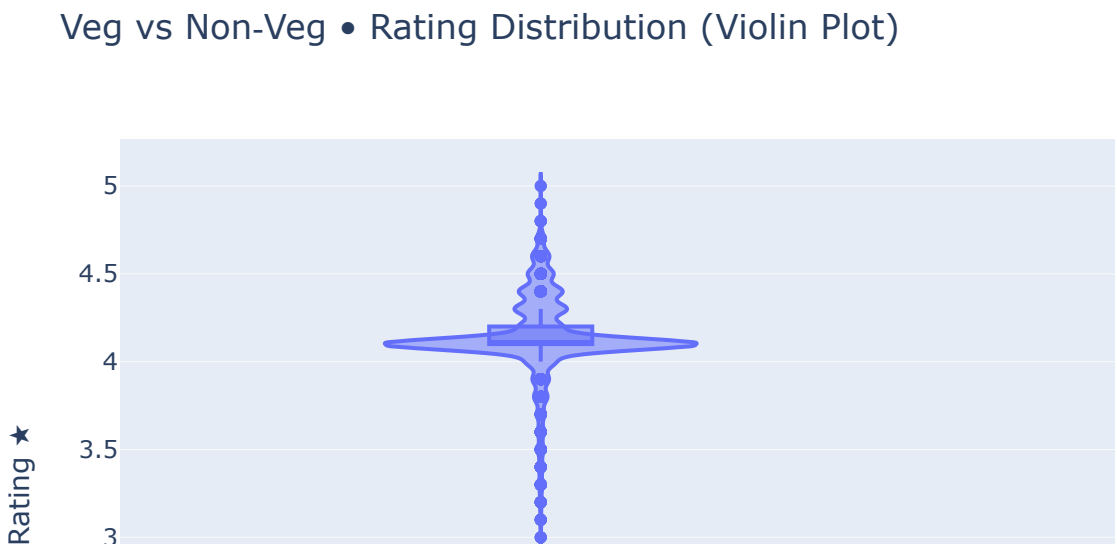
- Swiggy can highlight South Indian and light-snack categories more prominently in Chennai's "Top Picks" or "Most Loved" sections.
- These cuisines may offer a good entry point for onboarding in new subcities, as they seem to perform well with customers.
- Swiggy could also create a campaign like "Highly Rated under ₹200" that features these simple, high-rating cuisines.

Question 4: Are “veg” restaurants rated differently from “non veg”?

Approach:

To compare customer satisfaction between veg and non-veg restaurants, I used a Plotly violin plot. This helped visualize the full distribution of ratings, not just the average—showing density, median, spread, and outliers.To support this visually, I also conducted a two-sample t-test to check if the mean difference in ratings is statistically significant.

In [49]:
```
fig = px.violin(
    df,
    x="food_category",
    y="rating",
    color="food_category",
    box=True,
    title="Veg vs Non-Veg • Rating Distribution (Violin Plot)"
)

fig.update_layout(
    xaxis_title="Food Category",
    yaxis_title="Rating ★",
    showlegend=False
)

fig.show()
```



In [50]:
```
from scipy.stats import ttest_ind
veg = df[df['food_category']=='veg']['rating']
nonveg = df[df['food_category']=='non veg']['rating']
stat, p = ttest_ind(veg, nonveg, equal_var=False)
print(f"Veg mean: {veg.mean():.2f} | Non-veg mean: {nonveg.mean():.2f} | p-value: {
```

```
Veg mean: 4.13 | Non-veg mean: 4.02 | p-value: 0.0000
```

Insight:

- The median rating is nearly the same for both categories (4.1).
- However, veg restaurants show tighter consistency, with most ratings clustering between 4.0 and 4.3.
- Non-veg restaurants are more spread out, with several low-rated outliers down to 2.0 ratings.
- A t-test confirms that the mean rating for veg (4.13) is statistically higher than non-veg (4.02), with a p-value < 0.001.
- This means veg restaurants tend to perform more consistently, while non-veg includes both great and poor experiences.

Recommendation:

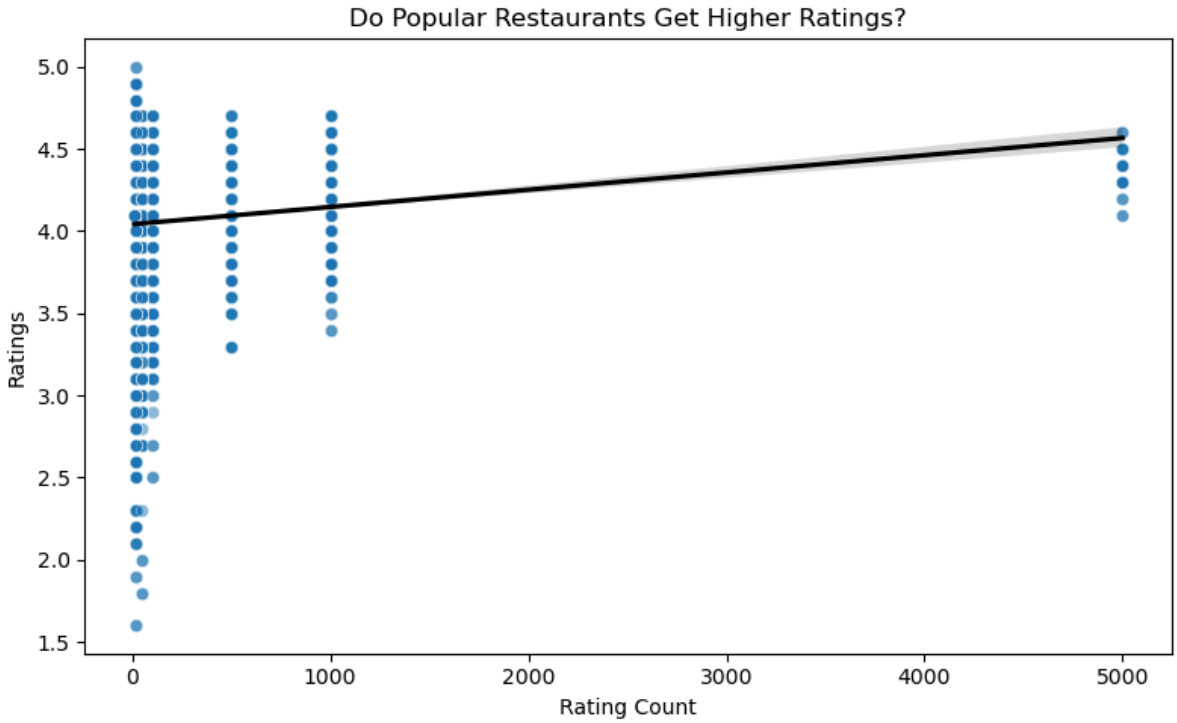
- Swiggy can highlight top-performing veg restaurants as “Consistently Rated” to attract users who value reliability.
- In the non-veg category, a “Top Rated Non-Veg” badge can help customers find quality options and avoid lower-rated ones.
- For non-veg partners with low ratings, offer targeted support or onboarding guidance to reduce rating variability.

Question 5: Is popularity (rating count) correlated with rating quality?

Approach:

To explore whether more popular restaurants (with more rating counts) tend to have better ratings, I plotted a scatter plot with rating count on the x-axis and average rating on the y-axis. A regression line was added to observe the overall trend visually.

In [51]:
```
plt.figure(figsize=(8, 5))
sns.scatterplot(x='rating count', y='rating', data=df, alpha=0.5)
sns.regplot(x='rating count', y='rating', data=df, scatter=False, color='black')
plt.title('Do Popular Restaurants Get Higher Ratings?')
plt.xlabel('Rating Count')
plt.ylabel('Ratings')
plt.tight_layout()
plt.show()
```



In [52]:
```
correlation = df["rating count"].corr(df["rating"])
correlation
```

Out[52]:
```
0.13042191428543964
```

Insight:

- The scatter plot shows a wide spread of ratings across all popularity levels.
- The regression line is mostly flat with a very slight upward trend, suggesting that popularity (number of ratings) has almost no relationship with the actual rating.
- The calculated Pearson correlation coefficient is 0.13, confirming that popularity (number of ratings) has very little influence on actual rating.
- This means highly-rated restaurants are not necessarily the most reviewed, and vice versa.

Recommendation:

- Swiggy should not rely solely on popularity (rating counts) when promoting restaurants. Quality (actual rating) should be considered separately.
- Consider highlighting restaurants with high ratings but fewer reviews to balance visibility and customer satisfaction.
- Add a tag like “Highly Rated but Less Known” to drive discovery and give newer or niche restaurants a chance to grow.

Question 6: Which cost category has the highest average rating?

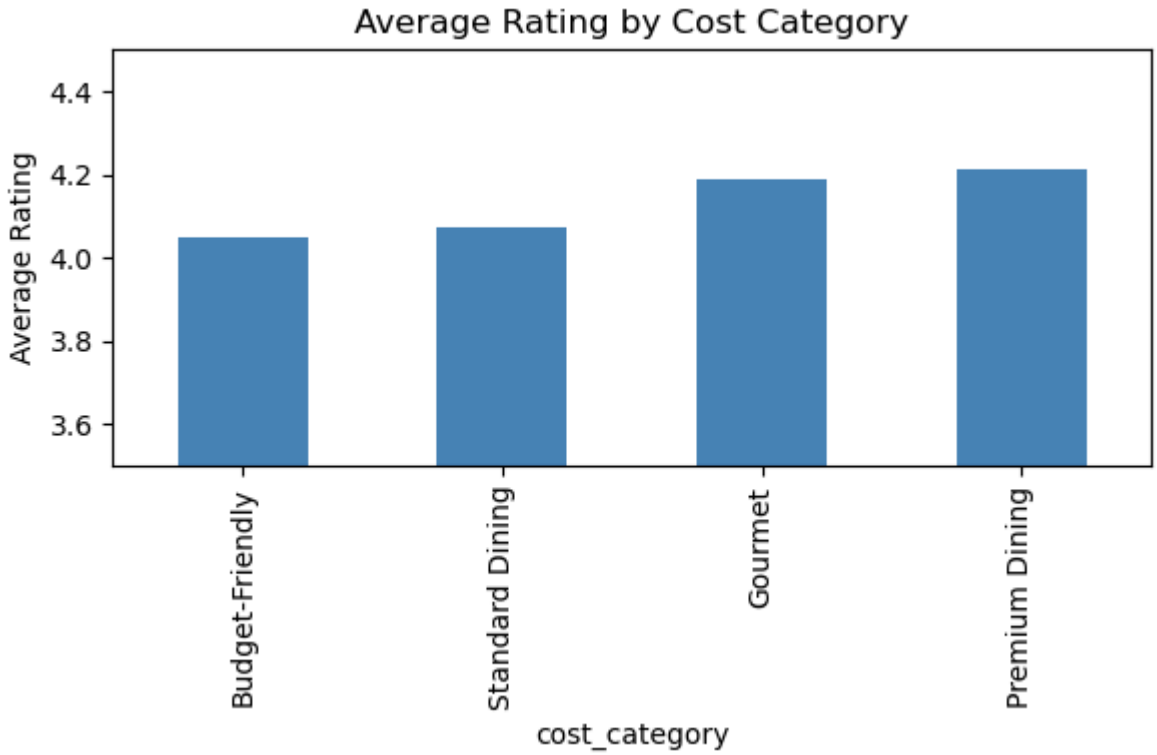
Approach:

To compare customer satisfaction across different pricing tiers, I grouped the data by cost_category and calculated the average restaurant rating in each group. I then used a bar chart (Matplotlib) to visualize which pricing segment consistently scores better in terms of customer reviews.

In [53]:
```
avg_costcat = df.groupby('cost_category')['rating'].mean().sort_values()
print(avg_costcat)
```

```
cost_category
Budget-Friendly    4.049777
Standard Dining    4.074801
Gourmet            4.188889
Premium Dining     4.211594
Name: rating, dtype: float64
```

In [54]:
```
plt.figure(figsize=(6,4))
avg_costcat.plot(kind='bar', color='steelblue')
plt.ylabel('Average Rating')
plt.title('Average Rating by Cost Category')
plt.ylim(3.5,4.5)
plt.tight_layout()
```



Insight:

- Premium Dining restaurants have the highest average rating at 4.21, followed closely by Gourmet at 4.19.
- Budget-Friendly and Standard Dining options have slightly lower averages (around 4.05), though still above 4 stars.
- This indicates that as cost increases, so does perceived quality, at least in customer ratings.

Recommendation:

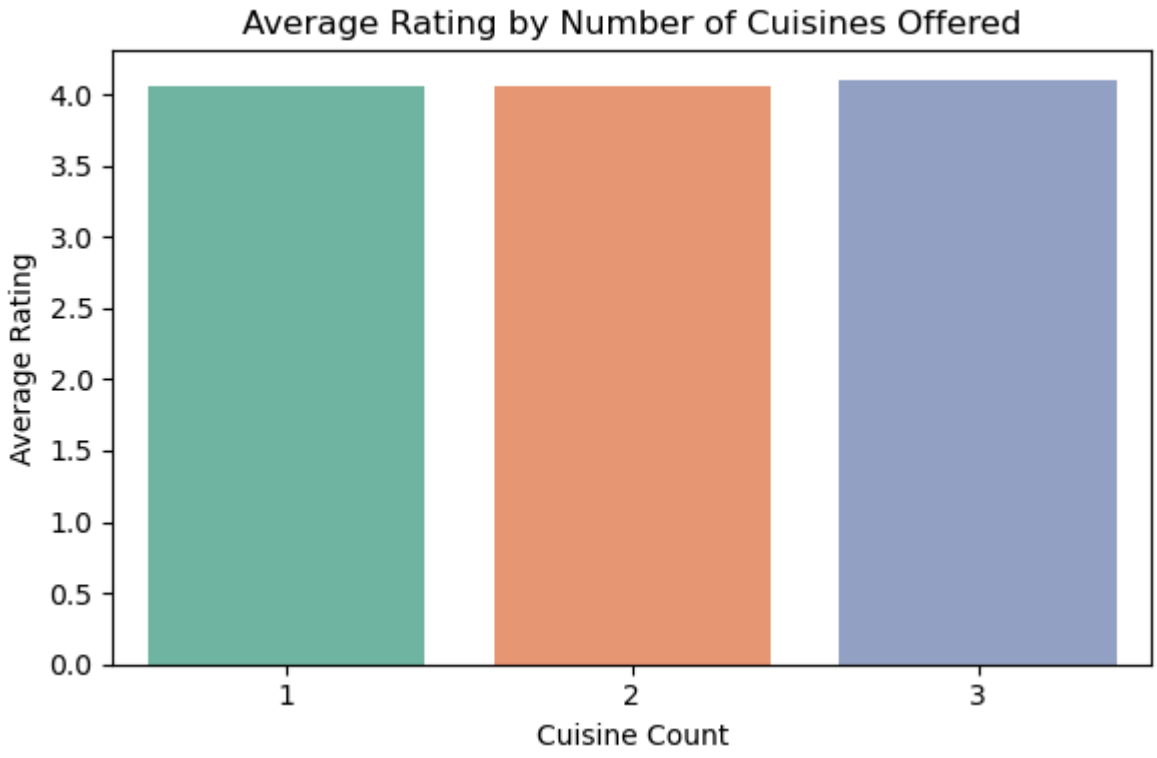
- Highlight Premium Dining and Gourmet restaurants more in the app to promote high-quality experiences.
- Swiggy could also use this insight to offer premium subscription bundles or exclusive deals featuring these top-rated, higher-cost outlets.
- On the other hand, monitoring Standard and Budget options for improvement could help close the experience gap.

Question 7: Does being multi-cuisine hurt or help ratings?

Approach:

I used a bar plot to compare the average ratings across restaurants offering 1, 2, and 3 different cuisines. The x-axis represents the number of cuisines (cuisine_count), while the y-axis shows the corresponding average customer rating.

In [55]:
```
plt.figure(figsize=(6,4))
sns.barplot(x='cuisine_count', y='rating', data=df, palette='Set2', errorbar=None)
plt.title('Average Rating by Number of Cuisines Offered')
plt.xlabel('Cuisine Count')
plt.ylabel('Average Rating')
plt.tight_layout()
```



Insight:

There is a very slight upward trend in average ratings with the number of cuisines offered. Restaurants offering 3 cuisines have a marginally higher average rating than those offering 1 or 2 cuisines. However, the difference is minimal, indicating that being multi-cuisine neither significantly helps nor hurts ratings.

Recommendation:

Swiggy should not prioritize multi-cuisine offerings as a key factor when onboarding or promoting restaurants. Instead, emphasis should be placed on curating high-quality restaurants, even if they specialize in just one cuisine. However, Swiggy can consider showcasing diverse-cuisine restaurants in specific campaigns (e.g., "One Stop for All Cravings") to improve convenience for users who prefer variety in a single order. Additionally, Swiggy can encourage quality control across all cuisines offered by multi-cuisine restaurants to ensure consistency in customer experience.

Question 8: Which sub-cities in Chennai have the highest number of low-rated high-cost and high-rated low-cost restaurants?

Approach:

To explore rating–cost performance extremes across Chennai, I created two filtered subsets of the data.The first subset includes restaurants in the 'Premium Dining' and 'Gourmet' categories with an average rating below 4.0, representing high-cost venues that underperform in customer satisfaction. The second subset captures restaurants in the 'Budget-Friendly' and 'Standard Dining' categories with a rating of 4.8 or higher, highlighting affordable outlets delivering strong customer experiences.For each group, I aggregated the number of restaurants by sub-city to identify where these contrasting patterns are most concentrated.Finally, I visualized both distributions using horizontal bar charts in Plotly to compare underperforming high-cost clusters against high-performing low-cost opportunities across the city.

In [56]:
```
low_rated_high_cost = df[
    (df['cost_category'].isin(['Premium Dining', 'Gourmet'])) &
    (df['rating'] < 4)]
low_rated_high_cost
```

Out[56]:

	city	subcity	restaurant	rating	rating count	cost	cuisine	food_category	rating
237	chennai	omr navalur	omr kitchen - fairfield by marriott	3.8	20	1200	indian	non veg	
238	chennai	omr navalur	omr kitchen - fairfield by marriott	3.8	20	1200	continental	non veg	
447	chennai	omr navalur	the eatery - four points by sheraton	3.8	50	1000	north indian	non veg	
448	chennai	omr navalur	the eatery - four points by sheraton	3.8	50	1000	asian	non veg	
456	chennai	omr navalur	kafe24 regenta central rs	3.7	20	1500	chinese	non veg	
457	chennai	omr navalur	kafe24 regenta central rs	3.7	20	1500	north indian	non veg	
5991	chennai	nungambakkam	impasta	3.9	20	1200	italian	veg	
5992	chennai	nungambakkam	impasta	3.9	20	1200	desserts	veg	
7180	chennai	ecr-kottivakkam	hub at ecr	3.9	50	1700	asian	non veg	
7181	chennai	ecr-kottivakkam	hub at ecr	3.9	50	1700	continental	non veg	
8562	chennai	mahindra world city	cafe 1st story - fairfield by marriott	3.6	20	1200	indian	non veg	
8563	chennai	mahindra world city	cafe 1st story - fairfield by marriott	3.6	20	1200	continental	non veg	
10228	chennai	karappakam	flower drum	3.9	100	1100	chinese	veg	
10871	chennai	t nagar	the right place - the residency	3.9	20	1000	indian	non veg	
10872	chennai	t nagar	the right place - the residency	3.9	20	1000	continental	non veg	

In [57]:

```
subcity_counts = low_rated_high_cost['subcity'].value_counts().reset_index()
subcity_counts.columns = ['subcity', 'count']
subcity_counts
```

Out[57]:

	subcity	count
0	omr navalur	6
1	nungambakkam	2
2	ecr-kottivakkam	2
3	mahindra world city	2
4	t nagar	2
5	karappakam	1

In [58]:
```
fig = px.bar(
    subcity_counts,
    x='count',
    y='subcity',
    orientation='h',
    color='count',
    title='Sub-cities with Low-rated High-cost Restaurants',
    labels={'subcity': 'Sub-city', 'count': 'Number of Restaurants'}
)
fig.update_layout(yaxis_title='Sub-city', xaxis_title='Number of Restaurants')
fig.show()
```

In [59]:
```
high_rated_low_cost = df[
    (df['cost_category'].isin(['Budget-Friendly', 'Standard Dining'])) &
    (df['rating'] >= 4.8)]
high_rated_low_cost
```

Out[59]:

	city	subcity	restaurant	rating	rating count	cost	cuisine	food_category	rating_c
259	chennai	omr navalur	ni foods and diet	4.9	20	300	healthy food	non veg	v
260	chennai	omr navalur	ni foods and diet	4.9	20	300	indian	non veg	v
854	chennai	guindy	fabelle chocolates - itc grand chola	4.8	20	600	bakery	veg	v
855	chennai	guindy	fabelle chocolates - itc grand chola	4.8	20	600	desserts	veg	v
980	chennai	annanagar	the lean bean chennai	4.9	20	600	healthy food	veg	v
1473	chennai	chrompet	cake waves	4.9	20	300	bakery	non veg	v
1474	chennai	chrompet	cake waves	4.9	20	300	desserts	non veg	v
1793	chennai	chrompet	hotel kumaran	4.8	20	200	south indian	veg	v
1794	chennai	chrompet	hotel kumaran	4.8	20	200	chinese	veg	v
2433	chennai	george town	mcrennett (parrys)	4.8	20	300	bakery	non veg	v
2894	chennai	omr perungudi	freshlings cafe	4.8	20	450	salads	non veg	v
4776	chennai	tambaram	rolls & bowls company	4.8	20	250	snacks	non veg	v
4777	chennai	tambaram	rolls & bowls company	4.8	20	250	chinese	non veg	v
5904	chennai	nungambakkam	madras coffee house	4.8	20	100	beverages	veg	v
5905	chennai	nungambakkam	madras coffee house	4.8	20	100	south indian	veg	v
9313	chennai	iyyapanthangal	spice nyce	4.9	20	200	indian	veg	v
9314	chennai	iyyapanthangal	spice nyce	4.9	20	200	chinese	veg	v
9924	chennai	villivakkam	not just a cake by jayaa kennethh	5.0	20	400	bakery	veg	v

	city	subcity	restaurant	rating	rating count	cost	cuisine	food_category	rating_c
9925	chennai	villivakkam	not just a cake by jayaa kennethh	5.0	20	400	desserts	veg	v
9980	chennai	villivakkam	amul ice cream parlour	4.9	20	200	ice cream	non veg	v
9981	chennai	villivakkam	amul ice cream	4.9	20	200	fast food	non veg	v

In [60]:
```
subcity_counts = (
    high_rated_low_cost['subcity']
    .value_counts()
    .reset_index()
    .rename(columns={'index': 'subcity', 'subcity': 'count'}))
subcity_counts
```

Out[60]:

	subcity	count
0	chrompet	4
1	villivakkam	4
2	omr navalur	2
3	guindy	2
4	tambaram	2
5	nungambakkam	2
6	iyyapanthangal	2
7	annanagar	1
8	george town	1
9	omr perungudi	1

In [61]:
```
fig = px.bar(
    subcity_counts,
    x='count',
    y='subcity',
    orientation='h',
    color='count',
    title='Sub-cities with High-rated Low-cost Restaurants',
    labels={'subcity': 'Sub-city', 'count': 'Number of Restaurants'}
)

fig.update_layout(
    yaxis_title='Sub-city',
    xaxis_title='Number of Restaurants'
)

fig.show()
```

Insight:

- Omr Navalur has the highest number of low-rated high-cost restaurants, indicating possible quality issues in expensive outlets in that sub-city.
- Chrompet and Villivakkam lead in high-rated low-cost restaurants, showing strong performance in budget-friendly segments.
- T Nagar, ECR-Kottivakkam, and Mahindra World City each have 2 low-rated high-cost restaurants, which may require service or quality improvements.
- Guindy, Tambaram, and Nungambakkam have multiple high-rated low-cost restaurants, signaling reliable value-for-money experiences in those areas.
- Overall, some sub-cities show a clear gap between pricing and quality, while others demonstrate consistent customer satisfaction even at lower price points.

Recommendation:

Swiggy should prioritize quality improvement efforts in sub-cities like Omr Navalur, T Nagar, and ECR-Kottivakkam, where several high-cost restaurants are underperforming in ratings. These areas represent a risk to customer satisfaction in the premium segment. At the same time, Swiggy can invest more in promoting and supporting high-rated low-cost restaurants in areas like Chrompet, Villivakkam, and Guindy, which consistently deliver great customer

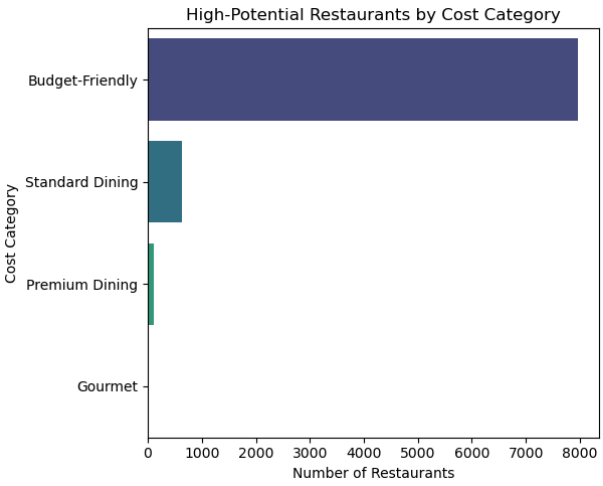
experiences at affordable prices. This dual approach will help reduce churn in premium zones while boosting profitability through high-performing budget-friendly outlets.

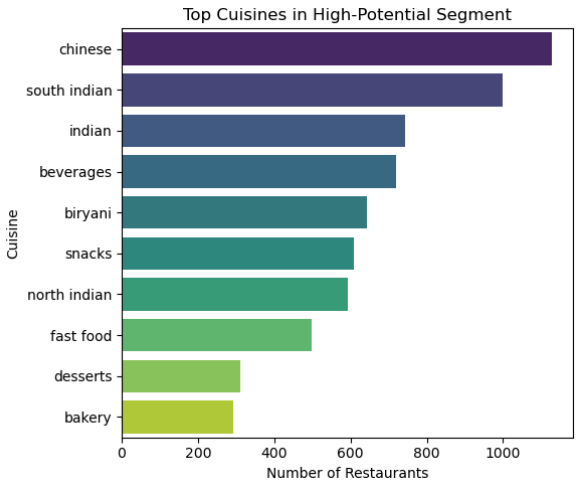
Question 9: What kind of restaurants should Swiggy invest in to increase profitability?

Approach:

To identify high-potential restaurant types, I filtered restaurants that have above-average ratings and at least median-level rating counts — implying strong customer satisfaction and sufficient customer base. I then analyzed which cost categories and cuisines are most common within this high-performing group to guide investment decisions.

In [62]:
```
rating_threshold       = df['rating'].mean()
rating_count_threshold = df['rating count'].median()

high_potential = df[
    (df['rating'] >= rating_threshold) &
    (df['rating count'] >= rating_count_threshold)]

cost_counts    = high_potential['cost_category'].value_counts()
cuisine_counts = high_potential['cuisine'].value_counts().head(10)


fig, axes = plt.subplots(1, 2, figsize=(12, 5))

sns.barplot(
    x=cost_counts.values, y=cost_counts.index,
    palette='viridis', ax=axes[0]
)
axes[0].set_title('High-Potential Restaurants by Cost Category')
axes[0].set_xlabel('Number of Restaurants')
axes[0].set_ylabel('Cost Category')

sns.barplot(
    x=cuisine_counts.values, y=cuisine_counts.index,
    palette='viridis', ax=axes[1]
)
axes[1].set_title('Top Cuisines in High-Potential Segment')
axes[1].set_xlabel('Number of Restaurants')
axes[1].set_ylabel('Cuisine')

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





Insight:

Most high-potential restaurants fall under the Budget-Friendly and Standard Dining categories, with Budget-Friendly being dominant. In terms of cuisine, Chinese, South Indian, Indian, and Beverages emerged as the most frequent among well-rated, widely reviewed outlets. These cuisines consistently perform well across cost categories and enjoy high customer engagement.

Recommendation:

Swiggy should prioritize partnerships and marketing support for Budget-Friendly and Standard Dining restaurants, especially those offering Chinese, South Indian, and Indian cuisines. These segments combine customer satisfaction with volume, offering strong potential for sustainable growth and increased order frequency. Investing in onboarding more such restaurants in underserved areas could drive profitability.

Conclusion

This project focused on uncovering actionable insights from Swiggy's restaurant data in Chennai. By analyzing patterns in ratings, cuisines and cost categories, I identified key factors that influence customer satisfaction and operational performance. The findings provide a data-driven foundation for strategic decisions related to partner onboarding, quality improvement, and targeted growth.