1. Initial Data Loading and Overview

Load File: Start by loading your dataset into a DataFrame.

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

def load_json(file):
    with open(file) as f:
        data = json.load(f)
    return data

data = load_json('data.json')
df = pd.DataFrame(data)

copy = df.copy()
```

In [ ]:
```
copy.columns
```

Out[ ]:
```
Index(['venue_id', 'latitude', 'longitude', 'venue_name',
       'platform_listing_url', 'region', 'neighborhood', 'address', 'pho
ne',
       'website', 'price_point_bucket', 'review_count', 'average_rating'
,
       'review_sample', 'opening_hours', 'busy_times', 'venue_amenities'
,
       'platform_category', 'venue_main_image', 'venue_segment',
       'venue_subsegment'],
      dtype='object')
```

Inspect Data Structure: Determine the shape of the DataFrame and list all column names to understand what data you're working with.

In [ ]:
```
data_set = df.copy()
print(f"The data frame has {data_set.shape[0]} rows and {data_set.shape[1

data_set
```

```
The data frame has 11865 rows and 21 columns
```

Out[ ]:

	venue_id	latitude	longitude	venue_name	
0	ChIJxQ08Bag92jERe0_5mhoBqxs	1.375928	103.955003	HaveFun Karaoke & Live Music Bar (Downtown East)	https:
				The Oak	https:

1	ChIJ47eMgaYQ2jERFX21pRHSOY8	1.331406	103.806817	Room	
2	ChIJh_VG84QX2jER5bM8A0V6ma4	1.333001	103.895961	UB3 Bistro	https:
3	ChIJC-992qkZ2jERzMnlGfVSJow	1.303198	103.832124	Nam Sing Hokkien Mee	https:
4	ChIJG0xEFbIZ2jER95y7R-Z8JJs	1.293449	103.851829	Hopscotch (Capitol)	https:
...	...	...	...	...	
11860	ChIJr20MyOMb2jERtARSHD0tl_Q	1.264767	103.818013	Dimbulah Coffee	https:
11861	ChIJQyPbI7EZ2jERHde-6qsZTwY	1.302063	103.859749	Celebrate-t Cafe	https:
11862	ChIJIWYmQgIb2jEROqFfZu4rlow	1.305566	103.791569	Jaguarita's Craft Beer Bar	https:
11863	ChIJYb8mL94T2jER9YydJ0CYbsk	1.455534	103.798947	Hakka Fat Moi Lei Cha Rice	https:
11864	ChIJwSEc38oX2jERi8JrxEosqL0	1.362641	103.894010	Goldhill Family Restaurant	https:

11865 rows × 21 columns

In [ ]:
```
# name of columns

data_set.columns
```

Out[ ]:
```
Index(['venue_id', 'latitude', 'longitude', 'venue_name',
       'platform_listing_url', 'region', 'neighborhood', 'address', 'pho
ne',
       'website', 'price_point_bucket', 'review_count', 'average_rating'
,
       'review_sample', 'opening_hours', 'busy_times', 'venue_amenities'
,
       'platform_category', 'venue_main_image', 'venue_segment',
       'venue_subsegment'],
      dtype='object')
```

Initial Data Summary: Use methods like .describe() to get an overview of numerical columns, specifically focusing on average_rating and other key metrics.

In [ ]:
```
#describe columns

data_set.describe()
```

Out[ ]:

	latitude	longitude	review_count	average_rating
count	11865.000000	11865.000000	11227.00000	11227.000000
mean	1.331153	103.842041	341.78525	4.036181
std	0.044242	0.063083	1209.38386	0.667167
min	1.243299	103.623720	1.00000	1.000000
25%	1.300966	103.810052	16.00000	3.800000
50%	1.318974	103.846217	87.00000	4.100000
75%	1.354348	103.876209	331.50000	4.500000
max	1.469738	104.049163	83336.00000	5.000000

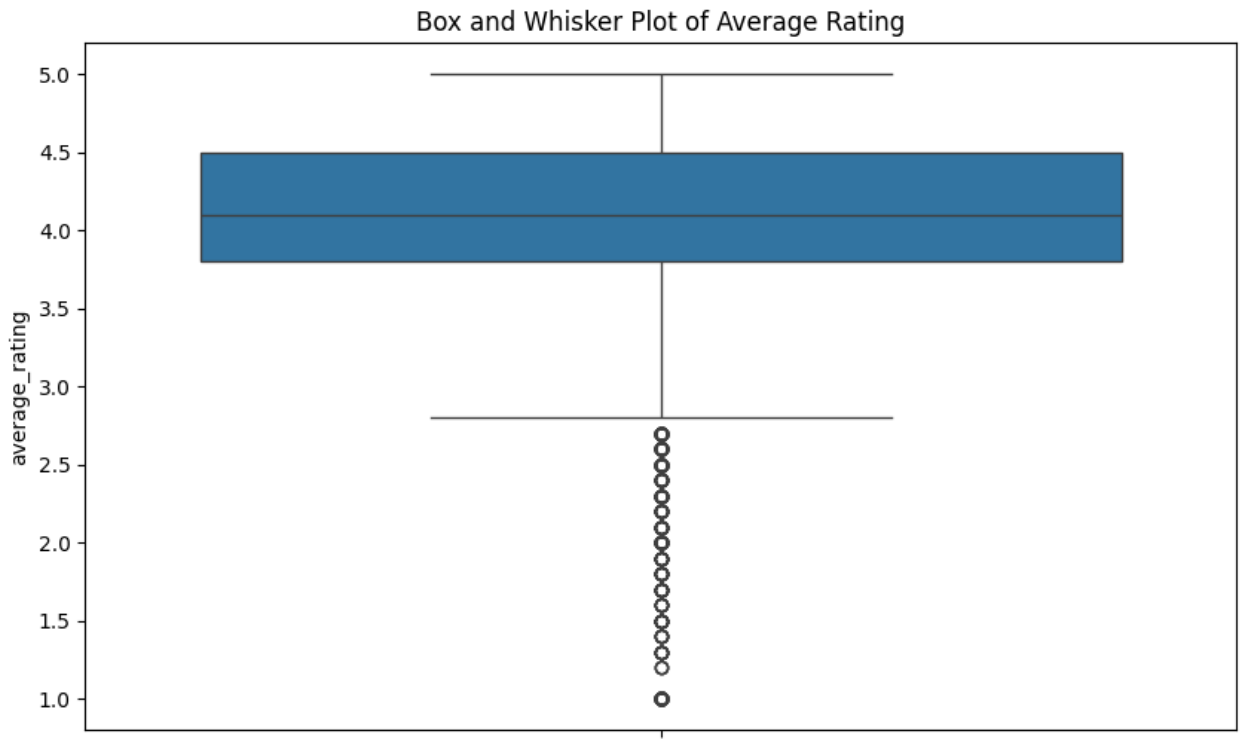
In [ ]:
```
#HEATMAP OPF
```

2. Data Exploration

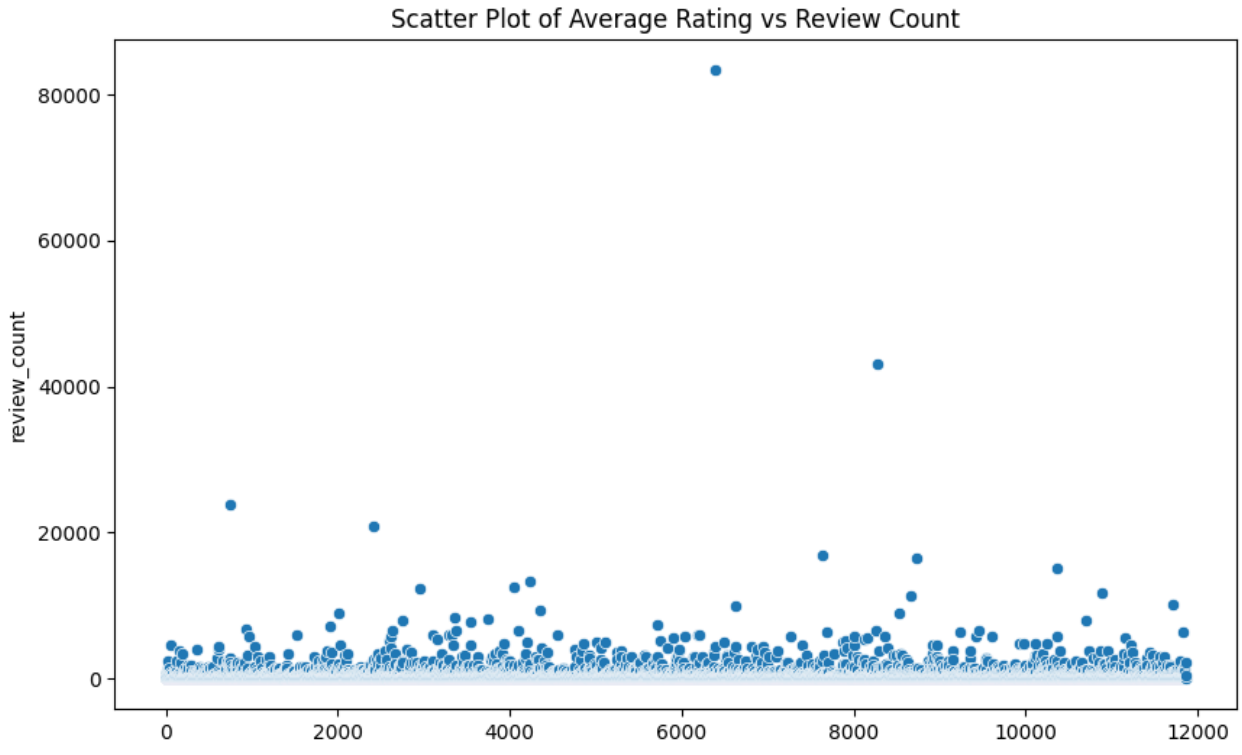
Box and Whisker Plot for Average Rating: Visualize the distribution of average_rating to identify outliers and understand its spread.

In [ ]:
```
# box and whisker plot of average_rating

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.boxplot(data_set['average_rating'])
plt.title('Box and Whisker Plot of Average Rating')
```

```
plt.show()
```



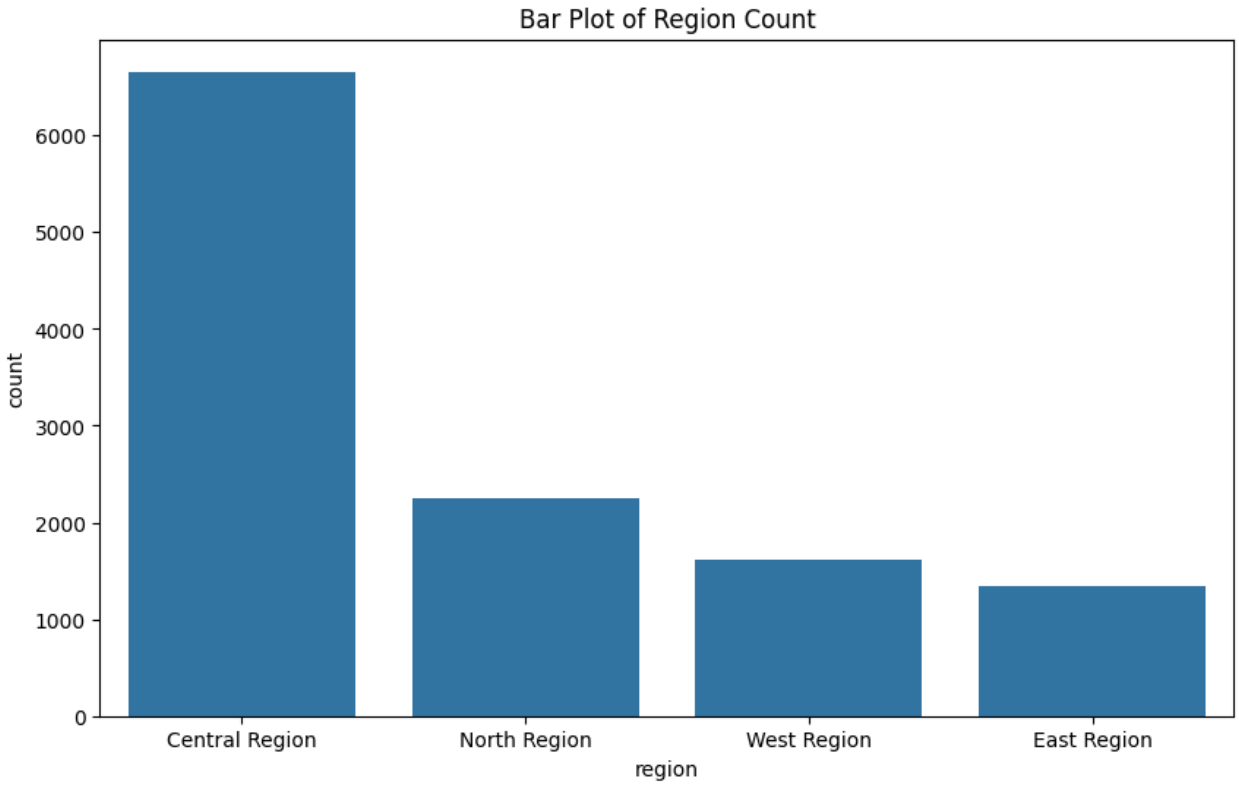
In [ ]:
```
# scatter plot of review count

plt.figure(figsize=(10, 6))
sns.scatterplot(data_set['review_count'])
plt.title('Scatter Plot of Average Rating vs Review Count')
plt.show()
```



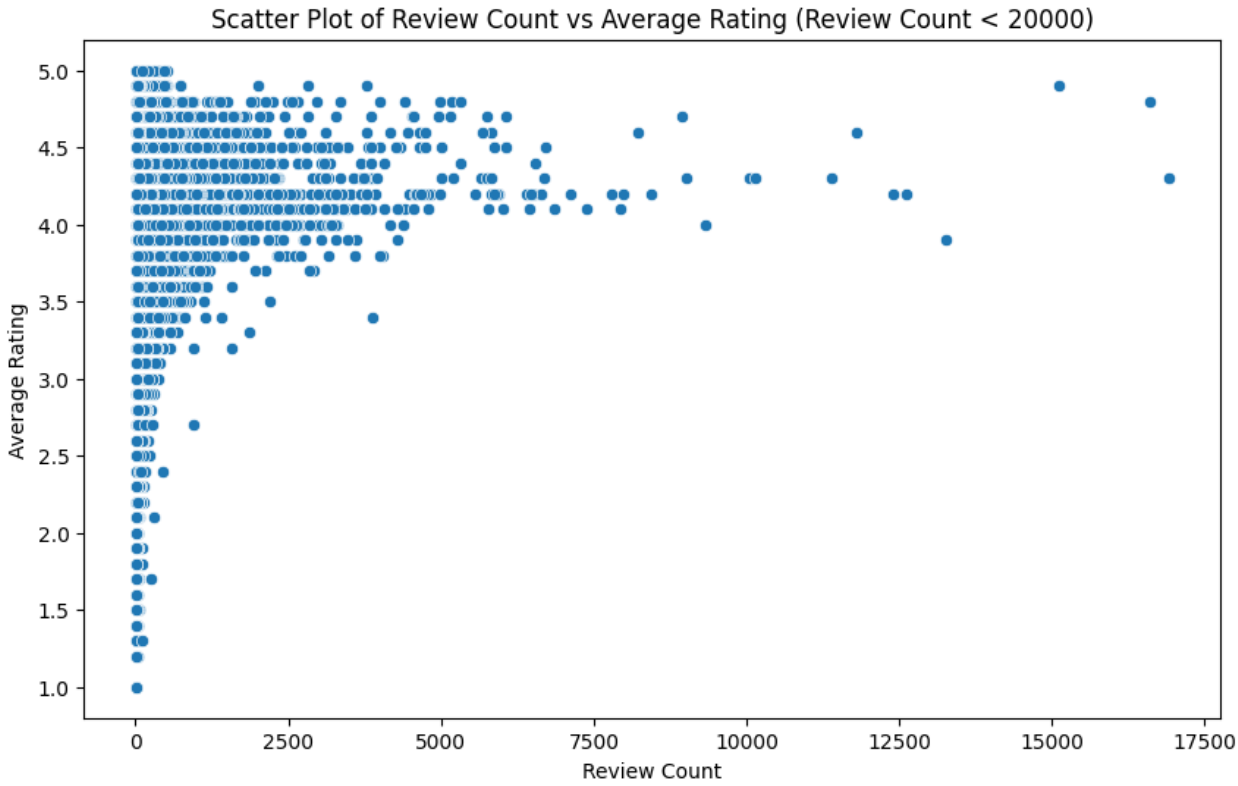
In [ ]:

Region Count Bar Plot: Visualize the count of venues per region. This is crucial for understanding geographical distribution.

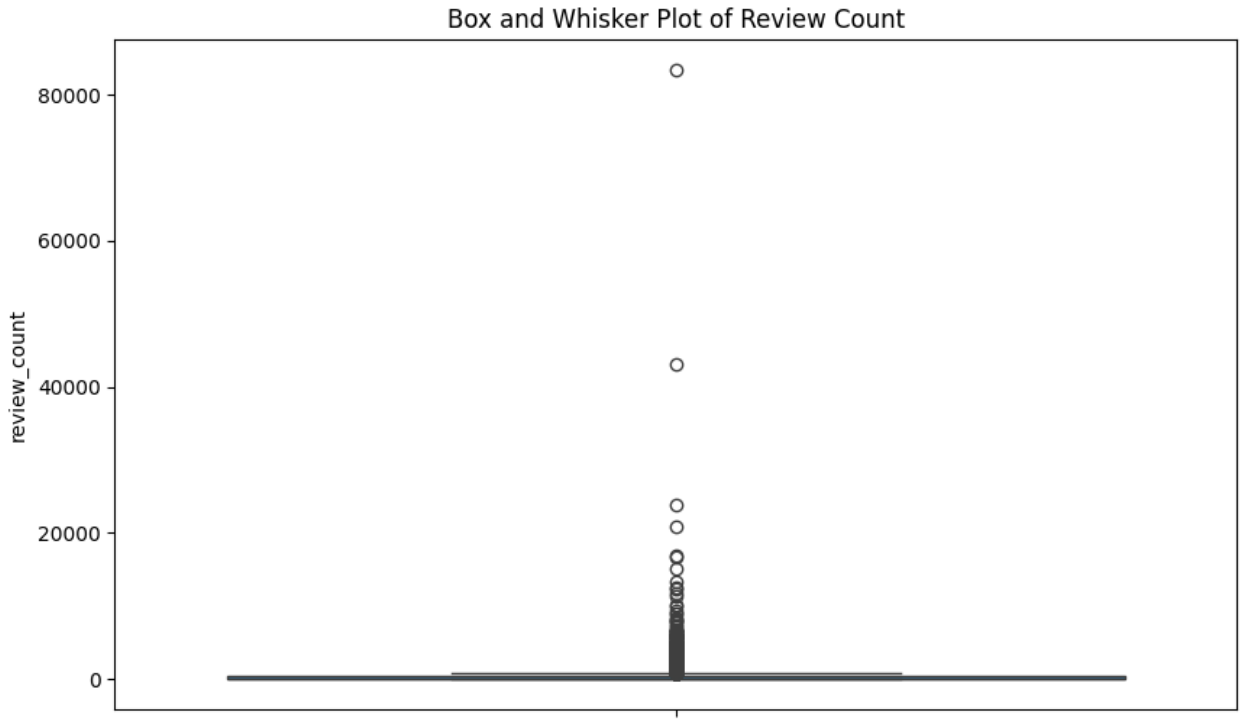
In [ ]:
```
#bar plot of region count

#combine north east and north

data_set['region'] = data_set['region'].replace('North-East Region', 'Nor


plt.figure(figsize=(10, 6))
sns.barplot( data_set['region'].value_counts())
plt.title('Bar Plot of Region Count')
plt.show()
```



In [ ]:
```
# Scatter Plots: like review count and average rating.

import seaborn as sns
import matplotlib.pyplot as plt

# Filter the dataset for review counts less than 20000
filtered_data = data_set[data_set['review_count'] < 20000]

plt.figure(figsize=(10, 6))
sns.scatterplot(x='review_count', y='average_rating', data=filtered_data)
plt.title('Scatter Plot of Review Count vs Average Rating (Review Count <
plt.xlabel('Review Count')
plt.ylabel('Average Rating')
plt.show()
```



In [ ]:
```
#make box and whisker of review_count

plt.figure(figsize=(10, 6))

sns.boxplot(data_set['review_count'])

plt.title('Box and Whisker Plot of Review Count')

plt.show()
```



In [ ]:
```
#COUNT OF PLACES IN EACH REGION

data_set['region'].value_counts()
```

Out[ ]:
```
region
Central Region    6659
North Region      2249
West Region       1620
East Region       1337
Name: count, dtype: int64
```

In [ ]:
```
#find the proportion of places in each region

data_set['region'].value_counts(normalize=True)
```

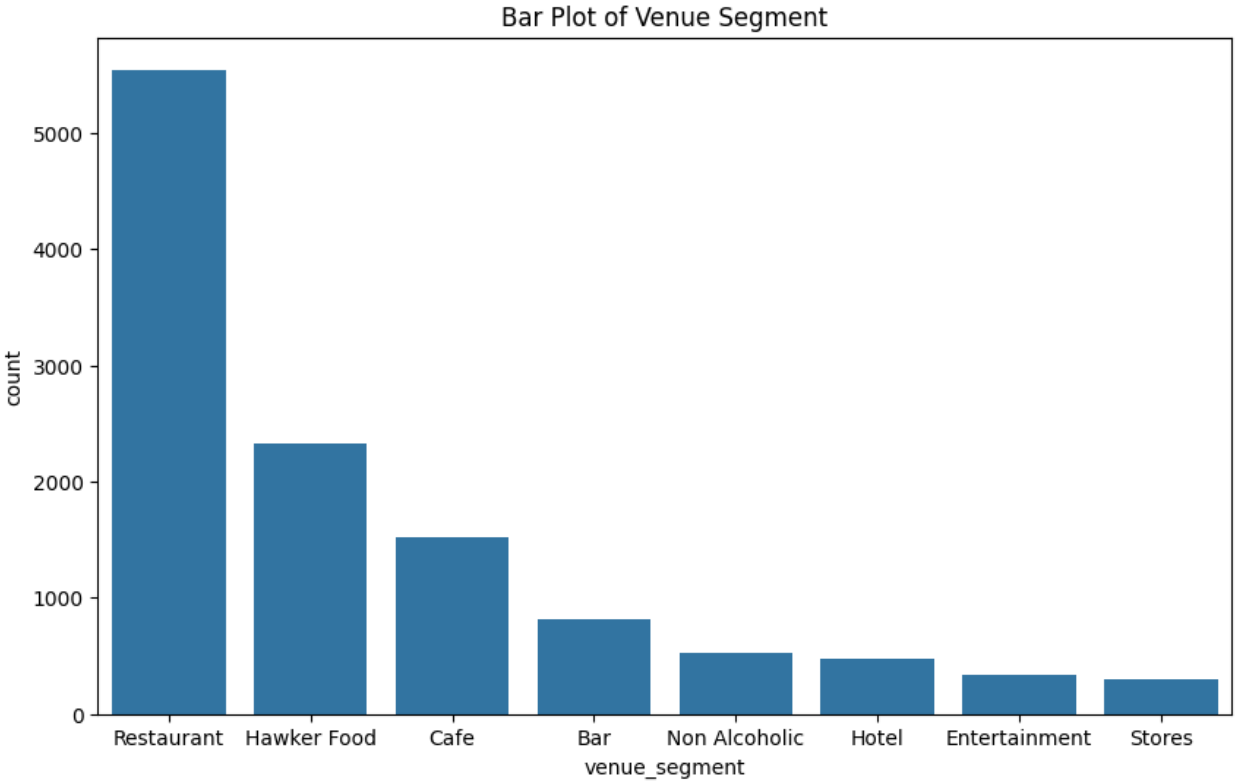
Out[ ]:
```
region
Central Region    0.561231
North Region      0.189549
West Region       0.136536
East Region       0.112684
Name: proportion, dtype: float64
```

Exploring platform_category: Investigate the diversity in platform_category to understand the variety of venues.

In [ ]:
```
#unique values in platform category

platforms = data_set['platform_category'].unique()
platforms
```

Out[ ]:
```
array(['Karaoke bar', 'Pub', 'Bistro', 'Hawker stall', 'Restaurant',
       'Bakery', 'Western restaurant', 'Cafe', 'Seafood restaurant',
       'Izakaya restaurant', 'Bar', 'North Indian restaurant',
       'Thai restaurant', 'Japanese restaurant', 'Sichuan restaurant',
       'Chinese restaurant', 'Hamburger restaurant', 'Hawker centre',
       'Cocktail bar', 'Family restaurant', 'Yakitori restaurant',
       'Beer garden', 'Italian restaurant', 'Food court', 'Hotel',
       'Bar & grill', 'Live music venue', 'Sports bar',
       'Pizza restaurant', 'Wine store', 'Ramen restaurant',
       'Noodle shop', 'Vietnamese restaurant', 'Liquor store',
       'Indian restaurant', 'Night club', 'Indian Muslim restaurant',
       'Disco club', 'American restaurant', 'Singaporean restaurant',
       'Sushi restaurant', 'Buffet restaurant', 'Halal restaurant',
       'Hot pot restaurant', 'Korean barbecue restaurant',
       'Indonesian restaurant', 'Spanish restaurant', 'Resort hotel',
       'Brewery', 'Japanese curry restaurant', 'Barbecue restaurant',
       'Fusion restaurant', 'Yakiniku restaurant', 'Takeout Restaurant',
       'Dim sum restaurant', 'Chinese noodle restaurant', 'Tapas bar',
       'Modern izakaya restaurant', 'Turkish restaurant',
       'Vegetarian restaurant', 'Chicken wings restaurant',
       'Asian restaurant', 'Malaysian restaurant', 'Steak house',
       'Fast food restaurant', 'Fine dining restaurant',
       'Korean restaurant', 'Chicken restaurant',
       'Fish and seafood restaurant', 'Nepalese restaurant',
       'Taiwanese restaurant', 'Authentic Japanese restaurant',
       'Modern European restaurant', 'Vegan restaurant', 'Live music bar
',
       'Mediterranean restaurant', 'French restaurant', 'Irish pub',
       'European restaurant', 'Health food restaurant',
       'Asian fusion restaurant', 'Beer store', 'Cantonese restaurant',
       'Concert hall', 'Nyonya restaurant', 'Mexican restaurant',
       'Breakfast restaurant', 'German restaurant', 'Event venue',
       'Art cafe', 'Wine bar', 'Bangladeshi restaurant', 'Brewpub',
       'Gay bar', 'Japanese steakhouse', 'Steamboat restaurant',
       'Modern French restaurant', 'Filipino restaurant',
       'Porridge restaurant', 'Middle Eastern restaurant',
       'Hong Kong style fast food restaurant',
       'Conveyor belt sushi restaurant', 'Diner', 'Gastropub', None,
       'Taco restaurant', 'Coffee roasters', 'Shanghainese restaurant',
       'Brunch restaurant', 'Dart bar', 'Cafeteria', 'Tex-Mex restaurant
',
       'Latin American restaurant', 'Greek restaurant', 'Dance club',
       'Winery', 'Social club', 'Argentinian restaurant',
       'Soul food restaurant', 'Kosher restaurant', 'Video karaoke',
       'Wine cellar'], dtype=object)
```

In [ ]:
```
#plot bar chart for venue_segment

plt.figure(figsize=(10, 6))
sns.barplot(data_set['venue_segment'].value_counts())
plt.title('Bar Plot of Venue Segment')
plt.show()
```



3. Data Cleaning

Column Reduction: Drop columns not relevant to the analysis to simplify the dataset.

Index(['venue_id', 'latitude', 'longitude', 'venue_name',

'platform_listing_url', 'region', 'neighborhood', 'address', 'phone',

'website', 'price_point_bucket', 'review_count', 'average_rating',

'review_sample', 'opening_hours', 'busy_times', 'venue_amenities',

'platform_category', 'venue_main_image',

'venue_segment',

'venue_subsegment', 'price_pb'],

dtype='object')

Review Sample to filter down last 25 venues?

In [ ]:
```
#drop column venue id, platform_listing_url, phone, website, review_sampl

data_set = data_set.drop(['venue_id', 'platform_listing_url', 'phone', 'w
```

Handling Null Values: Determine how many NA values are present in each column. Decide on a strategy for missing values, like filling null price_point with the mode.

In [ ]:
```
#find number of na values in each column

print(data_set.isna().sum())
```

```
latitude                0
longitude               0
venue_name              0
region                  0
price_point_bucket    7493
review_count           638
average_rating         638
review_sample            0
busy_times               0
venue_amenities          0
platform_category        8
venue_segment            9
venue_subsegment         9
dtype: int64
```

In [ ]:
```
#introduce a column in the data where price point bucket is replaced 1 if

data_set['price_point_bucket'] = data_set['price_point_bucket'].replace({

data_set
```

```
/var/folders/td/bwzm0ch95gv3n67hkws8_vwm0000gn/T/ipykernel_2702/3855637495
.py:3: FutureWarning: Downcasting behavior in `replace` is deprecated and
will be removed in a future version. To retain the old behavior, explicitl
y call `result.infer_objects(copy=False)`. To opt-in to the future behavio
r, set `pd.set_option('future.no_silent_downcasting', True)`
  data_set['price_point_bucket'] = data_set['price_point_bucket'].replace(
{'$': 1, '$$': 2, '$$$': 3, '$$$$': 4})
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cou
0	1.375928	103.955003	HaveFun Karaoke & Live Music Bar (Downtown East)	East Region	NaN	10
1	1.331406	103.806817	The Oak Room	Central Region	NaN	N
2	1.333001	103.895961	UB3 Bistro	Central Region	NaN	12
3	1.303198	103.832124	Nam Sing Hokkien Mee	Central Region	NaN	1
4	1.293449	103.851829	Hopscotch (Capitol)	Central Region	4.0	87
...	...	...	...	...	...	
11860	1.264767	103.818013	Dimbulah Coffee	Central Region	2.0	4
11861	1.302063	103.859749	Celebrate-t Cafe	Central Region	NaN	N
11862	1.305566	103.791569	Jaguarita's Craft Beer Bar	Central Region	2.0	6
11863	1.455534	103.798947	Hakka Fat Moi Lei Cha Rice	North Region	NaN	
11864	1.362641	103.894010	Goldhill Family Restaurant	North Region	1.0	46

11865 rows × 13 columns

In [ ]:
```
import numpy as np

high_end_categories = [
    "Fine dining restaurant", "Cocktail bar", "Wine bar", "Tapas bar",
    "Modern French restaurant", "Steak house", "Authentic Japanese restau
    "Seafood restaurant", "French restaurant", "European restaurant",
    "Mediterranean restaurant", "Korean barbecue restaurant",
    "Gastropub", "Concert hall", "Live music bar"
]
```

```
# Function to assign price point bucket considering platform_category, re
def assign_price_point(row):
    """Assigns price point bucket based on venue type, review count, and

    # If price point already exists, keep it
    if pd.notna(row["price_point_bucket"]):
        return row["price_point_bucket"]

    # High-end venues should have $$$ or $$$$ pricing
    if row["platform_category"] in high_end_categories:
        if row["average_rating"] >= 4.5:
            return 4
        elif 4.3 <= row["average_rating"] < 4.5:
            return 3
        else:
            return 2  # Luxury but not highly rated

    # Non-luxury venues assigned based on reviews & rating
    if row["average_rating"] >= 4.5 and row["review_count"] >= 450:
        return 4
    elif row["average_rating"] >= 4.0 and row["review_count"] >= 200:
        return 3
    else:
        return 2  # Default to $$ for mid-tier venues

# Apply function to dataset
data_set["price_point_bucket"] = data_set.apply(assign_price_point, axis=

# Display updated dataset
data_set.head()
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_count
0	1.375928	103.955003	HaveFun Karaoke & Live Music Bar (Downtown East)	East Region	2.0	101.0
1	1.331406	103.806817	The Oak Room	Central Region	2.0	NaN
2	1.333001	103.895961	UB3 Bistro	Central Region	2.0	128.0
3	1.303198	103.832124	Nam Sing Hokkien Mee	Central Region	2.0	13.0
4	1.293449	103.851829	Hopscotch (Capitol)	Central Region	4.0	874.0

In [ ]:

In [ ]:
```
#find number of na values in each column

print(data_set.isna().sum())
```

```
latitude                0
longitude               0
venue_name              0
region                  0
price_point_bucket      0
review_count          638
average_rating        638
review_sample           0
busy_times              0
venue_amenities         0
platform_category       8
venue_segment           9
venue_subsegment        9
dtype: int64
```

Refining venue_segment and busy_times: Remove rows with NA in venue_segment and drop the busy_times column if it's not usable.

In [ ]:
```
#show me rows that have na values in the column 'venue_segment'

na_venue = data_set[data_set['venue_segment'].isna()]

na_venue
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cou
2474	1.362391	103.951721	Platinum Wines & Spirits Pte. Ltd.	East Region	2.0	N
4148	1.455926	103.787727	8 Eight Ice Cold Beer	North Region	2.0	
5747	1.254018	103.814562	Rainbow Cocktail	Central Region	2.0	
6518	1.300650	103.858754	Bakery Bar	Central Region	2.0	N
6993	1.298777	103.856097	MK live music , family ktv , karaoke Disco pub...	Central Region	2.0	2
7287	1.442175	103.835121	Ohmine	North Region	2.0	N
7536	1.308103	103.731271	Taiwan Beer	West Region	2.0	N
7799	1.315400	103.866947	For Goodness Sake Pte. Ltd.	Central Region	2.0	N
11310	1.282750	103.843600	Capsule	Central Region	2.0	N

In [ ]:
```
#DROP values that are null in venue_subsegment, and platform_category

#since these venues dont seem too popular or rated or reviewed, we can dr

data_set = data_set.dropna(subset=['venue_subsegment', 'platform_category

#find number of na values in each column

print(data_set.isna().sum())
```

```
latitude                0
longitude               0
venue_name              0
region                  0
price_point_bucket      0
review_count          632
average_rating        632
review_sample           0
busy_times              0
venue_amenities         0
platform_category       0
venue_segment           0
venue_subsegment        0
dtype: int64
```

In [ ]:
```
#value_counts of busy times

data_set['busy_times'].value_counts()
```

Out[ ]:
```
busy_times
{}
11199
{'friday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0,
'10:00:00': 0, '11:00:00': 30, '12:00:00': 34, '13:00:00': 28, '14:00:00
': 19, '15:00:00': 0, '16:00:00': 0, '17:00:00': 0, '18:00:00': 58, '19:
00:00': 87, '20:00:00': 93, '21:00:00': 90, '22:00:00': 63, '23:00:00':
0}, 'saturday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00'
: 0, '10:00:00': 0, '11:00:00': 33, '12:00:00': 39, '13:00:00': 42, '14:
00:00': 39, '15:00:00': 0, '16:00:00': 0, '17:00:00': 0, '18:00:00': 63,
'19:00:00': 87, '20:00:00': 100, '21:00:00': 77, '22:00:00': 55, '23:00:
00': 0}, 'thursday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:0
0:00': 0, '10:00:00': 0, '11:00:00': 4, '12:00:00': 6, '13:00:00': 3, '1
4:00:00': 3, '15:00:00': 0, '16:00:00': 0, '17:00:00': 0, '18:00:00': 26
, '19:00:00': 41, '20:00:00': 42, '21:00:00': 30, '22:00:00': 14, '23:00
:00': 0}, 'tuesday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:0
0:00': 0, '10:00:00': 0, '11:00:00': 4, '12:00:00': 12, '13:00:00': 14,
'14:00:00': 19, '15:00:00': 0, '16:00:00': 0, '17:00:00': 0, '18:00:00':
33, '19:00:00': 38, '20:00:00': 36, '21:00:00': 20, '22:00:00': 7, '23:0
0:00': 0}, 'wednesday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '0
9:00:00': 0, '10:00:00': 0, '11:00:00': 1, '12:00:00': 6, '13:00:00': 12
, '14:00:00': 17, '15:00:00': 0, '16:00:00': 0, '17:00:00': 0, '18:00:00
': 36, '19:00:00': 41, '20:00:00': 47, '21:00:00': 33, '22:00:00': 26, '
23:00:00': 0}}
1
{'friday': {'00:00:00': 44, '01:00:00': 33, '02:00:00': 25, '03:00:00':
19, '06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:0
0': 0, '11:00:00': 0, '12:00:00': 0, '13:00:00': 0, '14:00:00': 0, '15:0
0:00': 0, '16:00:00': 74, '17:00:00': 88, '18:00:00': 94, '19:00:00': 92
, '20:00:00': 84, '21:00:00': 80, '22:00:00': 72, '23:00:00': 60}, 'mond
ay': {'00:00:00': 29, '06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:0
0:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 0, '13:00:00': 0, '1
4:00:00': 0, '15:00:00': 0, '16:00:00': 65, '17:00:00': 75, '18:00:00':
79, '19:00:00': 75, '20:00:00': 67, '21:00:00': 61, '22:00:00': 55, '23:
00:00': 44}, 'saturday': {'00:00:00': 45, '01:00:00': 33, '02:00:00': 26
```

, '03:00:00': 20, '06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 64, '13:00:00': 84, '14:00:00': 95, '15:00:00': 94, '16:00:00': 92, '17:00:00': 95, '18:00:00': 98, '19:00:00': 100, '20:00:00': 91, '21:00:00': 85, '22:00:00': 75, '23:00:00': 62}, 'sunday': {'00:00:00': 36, '01:00:00': 26, '02:00:00': 18, '06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 63, '13:00:00': 83, '14:00:00': 96, '15:00:00': 96, '16:00:00': 92, '17:00:00': 92, '18:00:00': 90, '19:00:00': 87, '20:00:00': 77, '21:00:00': 68, '22:00:00': 59, '23:00:00': 49}, 'thursday': {'00:00:00': 37, '01:00:00': 26, '02:00:00': 19, '06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 0, '13:00:00': 0, '14:00:00': 0, '15:00:00': 0, '16:00:00': 71, '17:00:00': 83, '18:00:00': 88, '19:00:00': 84, '20:00:00': 77, '21:00:00': 69, '22:00:00': 61, '23:00:00': 51}, 'tuesday': {'00:00:00': 33, '06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 0, '13:00:00': 0, '14:00:00': 0, '15:00:00': 0, '16:00:00': 63, '17:00:00': 74, '18:00:00': 79, '19:00:00': 77, '20:00:00': 69, '21:00:00': 65, '22:00:00': 60, '23:00:00': 49}, 'wednesday': {'00:00:00': 38, '01:00:00': 28, '02:00:00': 20, '03:00:00': 14, '06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 0, '13:00:00': 0, '14:00:00': 0, '15:00:00': 0, '16:00:00': 65, '17:00:00': 77, '18:00:00': 82, '19:00:00': 80, '20:00:00': 73, '21:00:00': 68, '22:00:00': 62, '23:00:00': 54}}
1
{'friday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 34, '12:00:00': 53, '13:00:00': 62, '14:00:00': 46, '15:00:00': 0, '16:00:00': 0, '17:00:00': 43, '18:00:00': 66, '19:00:00': 83, '20:00:00': 78, '21:00:00': 57, '22:00:00': 0, '23:00:00': 0}, 'monday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 40, '12:00:00': 75, '13:00:00': 77, '14:00:00': 57, '15:00:00': 0, '16:00:00': 0, '17:00:00': 33, '18:00:00': 46, '19:00:00': 45, '20:00:00': 34, '21:00:00': 15, '22:00:00': 0, '23:00:00': 0}, 'saturday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 72, '12:00:00': 81, '13:00:00': 62, '14:00:00': 40, '15:00:00': 0, '16:00:00': 0, '17:00:00': 43, '18:00:00': 69, '19:00:00': 89, '20:00:00': 87, '21:00:00': 57, '22:00:00': 0, '23:00:00': 0}, 'sunday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 43, '12:00:00': 57, '13:00:00': 50, '14:00:00': 66, '15:00:00': 0, '16:00:00': 0, '17:00:00': 65, '18:00:00': 87, '19:00:00': 84, '20:00:00': 60, '21:00:00': 37, '22:00:00': 0, '23:00:00': 0}, 'thursday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 56, '12:00:00': 86, '13:00:00': 98, '14:00:00': 78, '15:00:00': 0, '16:00:00': 0, '17:00:00': 46, '18:00:00': 50, '19:00:00': 62, '20:00:00': 59, '21:00:00': 43, '22:00:00': 0, '23:00:00': 0}, 'tuesday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 65, '12:00:00': 100, '13:00:00': 86, '14:00:00': 72, '15:00:00': 0, '16:00:00': 0, '17:00:00': 42, '18:00:00': 56, '19:00:00': 72, '20:00:00': 68, '21:00:00': 68, '22:00:00': 0, '23:00:00': 0}, 'wednesday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 43, '12:00:00': 63, '13:00:00': 62, '14:00:00': 43, '15:00:00': 0, '16:00:00': 0, '17:00:00': 39, '18:00:00': 34, '19:00:00': 37, '20:00:00': 43, '21:00:00': 43, '22:00:00': 0, '23:00:00': 0}}
1

{'friday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 0, '13:00:00': 0, '14:00:00': 15, '15:00:00': 23, '16:00:00': 34, '17:00:00': 15, '18:00:00': 7, '19:00:00': 3, '20:00:00': 7, '21:00:00': 7, '22:00:00': 7, '23:00:00': 19}, 'monday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 50, '13:00:00': 34, '14:00:00': 11, '15:00:00': 11, '16:00:00': 3, '17:00:00': 11, '18:00:00': 23, '19:00:00': 46, '20:00:00': 23, '21:00:00': 19, '22:00:00': 19, '23:00:00': 38}, 'saturday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 3, '13:00:00': 7, '14:00:00': 23, '15:00:00': 46, '16:00:00': 50, '17:00:00': 26, '18:00:00': 15, '19:00:00': 7, '20:00:00': 11, '21:00:00': 11, '22:00:00': 3, '23:00:00': 0}, 'sunday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 46, '13:00:00': 26, '14:00:00': 34, '15:00:00': 42, '16:00:00': 53, '17:00:00': 26, '18:00:00': 11, '19:00:00': 3, '20:00:00': 7, '21:00:00': 3, '22:00:00': 0, '23:00:00': 0}, 'thursday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 100, '13:00:00': 88, '14:00:00': 57, '15:00:00': 26, '16:00:00': 23, '17:00:00': 38, '18:00:00': 19, '19:00:00': 7, '20:00:00': 3, '21:00:00': 19, '22:00:00': 38, '23:00:00': 57}, 'tuesday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 0, '13:00:00': 3, '14:00:00': 7, '15:00:00': 7, '16:00:00': 7, '17:00:00': 3, '18:00:00': 3, '19:00:00': 7, '20:00:00': 7, '21:00:00': 0, '22:00:00': 0, '23:00:00': 3}, 'wednesday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 23, '13:00:00': 38, '14:00:00': 53, '15:00:00': 92, '16:00:00': 96, '17:00:00': 61, '18:00:00': 38, '19:00:00': 38, '20:00:00': 57, '21:00:00': 53, '22:00:00': 34, '23:00:00': 34}}
1

...
{'friday': {'00:00:00': 88, '01:00:00': 74, '02:00:00': 57, '06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 0, '13:00:00': 0, '14:00:00': 0, '15:00:00': 0, '16:00:00': 0, '17:00:00': 0, '18:00:00': 9, '19:00:00': 15, '20:00:00': 27, '21:00:00': 53, '22:00:00': 84, '23:00:00': 100}, 'monday': {'00:00:00': 16, '01:00:00': 12, '02:00:00': 11, '06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 0, '13:00:00': 0, '14:00:00': 0, '15:00:00': 0, '16:00:00': 0, '17:00:00': 0, '18:00:00': 7, '19:00:00': 14, '20:00:00': 20, '21:00:00': 23, '22:00:00': 30, '23:00:00': 27}, 'saturday': {'00:00:00': 52, '01:00:00': 44, '02:00:00': 32, '06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 0, '13:00:00': 0, '14:00:00': 0, '15:00:00': 0, '16:00:00': 0, '17:00:00': 0, '18:00:00': 7, '19:00:00': 10, '20:00:00': 20, '21:00:00': 38, '22:00:00': 57, '23:00:00': 64}, 'sunday': {'00:00:00': 13, '01:00:00': 5, '02:00:00': 4, '06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 0, '13:00:00': 0, '14:00:00': 0, '15:00:00': 0, '16:00:00': 0, '17:00:00': 0, '18:00:00': 4, '19:00:00': 9, '20:00:00': 23, '21:00:00': 28, '22:00:00': 34, '23:00:00': 24}, 'thursday': {'00:00:00': 49, '01:00:00': 32, '02:00:00': 14, '06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 0, '13:00:

00': 0, '14:00:00': 0, '15:00:00': 0, '16:00:00': 0, '17:00:00': 0, '18:
00:00': 0, '19:00:00': 8, '20:00:00': 27, '21:00:00': 55, '22:00:00': 75
, '23:00:00': 74}, 'tuesday': {'00:00:00': 32, '01:00:00': 19, '02:00:00
': 15, '06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:0
0:00': 0, '11:00:00': 0, '12:00:00': 0, '13:00:00': 0, '14:00:00': 0, '1
5:00:00': 0, '16:00:00': 0, '17:00:00': 0, '18:00:00': 5, '19:00:00': 12
, '20:00:00': 22, '21:00:00': 34, '22:00:00': 45, '23:00:00': 47}, 'wedn
esday': {'00:00:00': 29, '01:00:00': 18, '02:00:00': 14, '06:00:00': 0,
'07:00:00': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00':
0, '12:00:00': 0, '13:00:00': 0, '14:00:00': 0, '15:00:00': 0, '16:00:00
': 0, '17:00:00': 0, '18:00:00': 1, '19:00:00': 5, '20:00:00': 15, '21:0
0:00': 28, '22:00:00': 40, '23:00:00': 41}}
1
{'friday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0,
'10:00:00': 0, '11:00:00': 0, '12:00:00': 19, '13:00:00': 24, '14:00:00'
: 21, '15:00:00': 0, '16:00:00': 0, '17:00:00': 0, '18:00:00': 48, '19:0
0:00': 68, '20:00:00': 77, '21:00:00': 62, '22:00:00': 40, '23:00:00': 0
}, 'monday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0
, '10:00:00': 0, '11:00:00': 0, '12:00:00': 43, '13:00:00': 45, '14:00:0
0': 36, '15:00:00': 0, '16:00:00': 0, '17:00:00': 0, '18:00:00': 45, '19
:00:00': 54, '20:00:00': 49, '21:00:00': 34, '22:00:00': 16, '23:00:00':
0}, 'saturday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00'
: 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 20, '13:00:00': 19, '14:0
0:00': 20, '15:00:00': 0, '16:00:00': 0, '17:00:00': 0, '18:00:00': 61,
'19:00:00': 84, '20:00:00': 100, '21:00:00': 86, '22:00:00': 54, '23:00:
00': 0}, 'thursday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:0
0:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 46, '13:00:00': 43,
'14:00:00': 34, '15:00:00': 0, '16:00:00': 0, '17:00:00': 0, '18:00:00':
66, '19:00:00': 89, '20:00:00': 95, '21:00:00': 75, '22:00:00': 49, '23:
00:00': 0}, 'tuesday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09
:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 36, '13:00:00': 36
, '14:00:00': 30, '15:00:00': 0, '16:00:00': 0, '17:00:00': 0, '18:00:00
': 61, '19:00:00': 80, '20:00:00': 87, '21:00:00': 69, '22:00:00': 46, '
23:00:00': 0}, 'wednesday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0
, '09:00:00': 0, '10:00:00': 0, '11:00:00': 0, '12:00:00': 39, '13:00:00
': 42, '14:00:00': 30, '15:00:00': 0, '16:00:00': 0, '17:00:00': 0, '18:
00:00': 66, '19:00:00': 81, '20:00:00': 79, '21:00:00': 63, '22:00:00':
40, '23:00:00': 0}}
1
{'friday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0,
'10:00:00': 0, '11:00:00': 31, '12:00:00': 46, '13:00:00': 45, '14:00:00
': 37, '15:00:00': 28, '16:00:00': 31, '17:00:00': 47, '18:00:00': 68, '
19:00:00': 81, '20:00:00': 69, '21:00:00': 40, '22:00:00': 0, '23:00:00'
: 0}, 'monday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00'
: 0, '10:00:00': 0, '11:00:00': 21, '12:00:00': 30, '13:00:00': 31, '14:
00:00': 22, '15:00:00': 17, '16:00:00': 16, '17:00:00': 22, '18:00:00':
29, '19:00:00': 34, '20:00:00': 29, '21:00:00': 17, '22:00:00': 0, '23:0
0:00': 0}, 'saturday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09
:00:00': 0, '10:00:00': 0, '11:00:00': 46, '12:00:00': 70, '13:00:00': 8
0, '14:00:00': 73, '15:00:00': 56, '16:00:00': 52, '17:00:00': 68, '18:0
0:00': 92, '19:00:00': 100, '20:00:00': 80, '21:00:00': 48, '22:00:00':
0, '23:00:00': 0}, 'sunday': {'06:00:00': 0, '07:00:00': 0, '08:00:00':
0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 54, '12:00:00': 77, '13:00:
00': 81, '14:00:00': 66, '15:00:00': 53, '16:00:00': 52, '17:00:00': 68,

'18:00:00': 90, '19:00:00': 86, '20:00:00': 63, '21:00:00': 32, '22:00:0
0': 0, '23:00:00': 0}, 'thursday': {'06:00:00': 0, '07:00:00': 0, '08:00
:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 28, '12:00:00': 40, '
13:00:00': 39, '14:00:00': 32, '15:00:00': 23, '16:00:00': 23, '17:00:00
': 32, '18:00:00': 45, '19:00:00': 52, '20:00:00': 43, '21:00:00': 21, '
22:00:00': 0, '23:00:00': 0}, 'tuesday': {'06:00:00': 0, '07:00:00': 0,
'08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 23, '12:00:00':
38, '13:00:00': 41, '14:00:00': 32, '15:00:00': 24, '16:00:00': 19, '17:
00:00': 26, '18:00:00': 38, '19:00:00': 46, '20:00:00': 39, '21:00:00':
21, '22:00:00': 0, '23:00:00': 0}, 'wednesday': {'06:00:00': 0, '07:00:0
0': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 31, '12:
00:00': 49, '13:00:00': 49, '14:00:00': 36, '15:00:00': 22, '16:00:00':
20, '17:00:00': 32, '18:00:00': 48, '19:00:00': 55, '20:00:00': 44, '21:
00:00': 25, '22:00:00': 0, '23:00:00': 0}}
1
{'friday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0,
'10:00:00': 0, '11:00:00': 48, '12:00:00': 68, '13:00:00': 70, '14:00:00
': 52, '15:00:00': 41, '16:00:00': 43, '17:00:00': 59, '18:00:00': 75, '
19:00:00': 77, '20:00:00': 63, '21:00:00': 39, '22:00:00': 0, '23:00:00'
: 0}, 'monday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00'
: 0, '10:00:00': 0, '11:00:00': 26, '12:00:00': 54, '13:00:00': 49, '14:
00:00': 34, '15:00:00': 19, '16:00:00': 28, '17:00:00': 36, '18:00:00':
53, '19:00:00': 54, '20:00:00': 53, '21:00:00': 35, '22:00:00': 0, '23:0
0:00': 0}, 'saturday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09
:00:00': 0, '10:00:00': 0, '11:00:00': 21, '12:00:00': 50, '13:00:00': 6
7, '14:00:00': 71, '15:00:00': 68, '16:00:00': 73, '17:00:00': 87, '18:0
0:00': 99, '19:00:00': 100, '20:00:00': 75, '21:00:00': 50, '22:00:00':
0, '23:00:00': 0}, 'sunday': {'06:00:00': 0, '07:00:00': 0, '08:00:00':
0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 32, '12:00:00': 52, '13:00:
00': 57, '14:00:00': 65, '15:00:00': 62, '16:00:00': 62, '17:00:00': 65,
'18:00:00': 75, '19:00:00': 76, '20:00:00': 59, '21:00:00': 33, '22:00:0
0': 0, '23:00:00': 0}, 'thursday': {'06:00:00': 0, '07:00:00': 0, '08:00
:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 49, '12:00:00': 81, '
13:00:00': 84, '14:00:00': 64, '15:00:00': 40, '16:00:00': 35, '17:00:00
': 37, '18:00:00': 48, '19:00:00': 53, '20:00:00': 46, '21:00:00': 33, '
22:00:00': 0, '23:00:00': 0}, 'tuesday': {'06:00:00': 0, '07:00:00': 0,
'08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 36, '12:00:00':
59, '13:00:00': 56, '14:00:00': 49, '15:00:00': 28, '16:00:00': 25, '17:
00:00': 37, '18:00:00': 59, '19:00:00': 56, '20:00:00': 56, '21:00:00':
43, '22:00:00': 0, '23:00:00': 0}, 'wednesday': {'06:00:00': 0, '07:00:0
0': 0, '08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 45, '12:
00:00': 67, '13:00:00': 65, '14:00:00': 51, '15:00:00': 34, '16:00:00':
37, '17:00:00': 35, '18:00:00': 45, '19:00:00': 45, '20:00:00': 53, '21:
00:00': 37, '22:00:00': 0, '23:00:00': 0}}
1
{'friday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00': 0,
'10:00:00': 0, '11:00:00': 39, '12:00:00': 47, '13:00:00': 49, '14:00:00
': 46, '15:00:00': 50, '16:00:00': 52, '17:00:00': 57, '18:00:00': 57, '
19:00:00': 63, '20:00:00': 0, '21:00:00': 0, '22:00:00': 0, '23:00:00':
0}, 'monday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:00:00':
0, '10:00:00': 0, '11:00:00': 27, '12:00:00': 34, '13:00:00': 30, '14:00
:00': 24, '15:00:00': 23, '16:00:00': 23, '17:00:00': 34, '18:00:00': 38
, '19:00:00': 35, '20:00:00': 23, '21:00:00': 17, '22:00:00': 0, '23:00:

```
00': 0}, 'saturday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:0
0:00': 0, '10:00:00': 0, '11:00:00': 24, '12:00:00': 47, '13:00:00': 57,
'14:00:00': 62, '15:00:00': 47, '16:00:00': 41, '17:00:00': 42, '18:00:0
0': 0, '19:00:00': 0, '20:00:00': 0, '21:00:00': 0, '22:00:00': 0, '23:0
0:00': 0}, 'sunday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0, '09:0
0:00': 0, '10:00:00': 0, '11:00:00': 26, '12:00:00': 24, '13:00:00': 29,
'14:00:00': 32, '15:00:00': 38, '16:00:00': 34, '17:00:00': 43, '18:00:0
0': 51, '19:00:00': 68, '20:00:00': 69, '21:00:00': 52, '22:00:00': 0, '
23:00:00': 0}, 'thursday': {'06:00:00': 0, '07:00:00': 0, '08:00:00': 0,
'09:00:00': 0, '10:00:00': 0, '11:00:00': 15, '12:00:00': 19, '13:00:00'
: 26, '14:00:00': 26, '15:00:00': 29, '16:00:00': 30, '17:00:00': 38, '1
8:00:00': 41, '19:00:00': 43, '20:00:00': 44, '21:00:00': 40, '22:00:00'
: 0, '23:00:00': 0}, 'tuesday': {'06:00:00': 0, '07:00:00': 0, '08:00:00
': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 13, '12:00:00': 32, '13:
00:00': 46, '14:00:00': 49, '15:00:00': 38, '16:00:00': 36, '17:00:00':
39, '18:00:00': 44, '19:00:00': 40, '20:00:00': 32, '21:00:00': 26, '22:
00:00': 0, '23:00:00': 0}, 'wednesday': {'06:00:00': 0, '07:00:00': 0, '
08:00:00': 0, '09:00:00': 0, '10:00:00': 0, '11:00:00': 30, '12:00:00':
34, '13:00:00': 41, '14:00:00': 43, '15:00:00': 46, '16:00:00': 45, '17:
00:00': 40, '18:00:00': 46, '19:00:00': 42, '20:00:00': 50, '21:00:00':
42, '22:00:00': 0, '23:00:00': 0}}
1
Name: count, Length: 658, dtype: int64
```

In [ ]:
```
#drop busy times, because most of them are not filled

data_set = data_set.drop(['busy_times'], axis=1)
```

In [ ]:
```
# See rows with review_count = nULL

data_set[data_set['review_count'].isna()]
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cou
1	1.331406	103.806817	The Oak Room	Central Region	2.0	N
20	1.279834	103.825962	#02-18 Hawker Stall	Central Region	2.0	N
92	1.443216	103.785246	Lal and Sons	North Region	2.0	N
133	1.339292	103.705874	Kuriya Japanese Market	West Region	2.0	N
145	1.326906	103.846393	W XYZ Bar	Central Region	2.0	N
...	...	...	...	...	...	
11788	1.336178	103.886425	Retro Bar	Central Region	2.0	N
11807	1.363946	103.848001	Yap Heng	North Region	2.0	N
11847	1.418803	103.835619	D'Reedaque - Muslim Food	North Region	2.0	N
11858	1.305693	103.731557	Firestone Tavern	West Region	2.0	N
11861	1.302063	103.859749	Celebrate-t Cafe	Central Region	2.0	N

632 rows × 12 columns

In [ ]:
```
# remove rows with review_count = NULL

data_set = data_set.dropna(subset=['review_count'])

#find number of na values in each column

print(data_set.isna().sum())
```

```
latitude              0
longitude             0
venue_name            0
region                0
price_point_bucket    0
review_count          0
average_rating        0
review_sample         0
venue_amenities       0
platform_category     0
venue_segment         0
venue_subsegment      0
dtype: int64
```

In [ ]:
```
#FIND AVERAGE RATING OF PLACES IN EACH REGION

data_set.groupby('region')['average_rating'].mean()
```

Out[ ]:
```
region
Central Region    4.142374
East Region       3.938462
North Region      3.877505
West Region       3.893829
Name: average_rating, dtype: float64
```

In [ ]:
```
#print column names

print(data_set.columns)
```

```
Index(['latitude', 'longitude', 'venue_name', 'region', 'price_point_bucke
t',
       'review_count', 'average_rating', 'review_sample', 'venue_amenities
',
       'platform_category', 'venue_segment', 'venue_subsegment'],
      dtype='object')
```

In [ ]:
```
#show me rows that have venue_segment as 'Stores'

stores = data_set[data_set['venue_segment'] == 'Stores']

stores
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cou
77	1.314000	103.844250	Affordable Wines	Central Region	2.0	1
94	1.310328	103.864446	Rogue Merchants Pte Ltd	Central Region	2.0	
100	1.343465	103.870320	Khian Aik Provision Store	North Region	2.0	
171	1.324826	103.894681	Vin Republic	Central Region	2.0	
209	1.343746	103.824045	Wine-family.com	North Region	2.0	
...	...	...	...	...	...	
11632	1.403115	103.913221	Estate Wines Cellar	North Region	2.0	
11676	1.284040	103.850973	Bottles & Bottles	Central Region	2.0	1
11715	1.334480	103.897081	Cellarbration	Central Region	2.0	17
11732	1.283243	103.850836	Booze Wine Shop	Central Region	2.0	
11763	1.310588	103.795558	The Standish	Central Region	2.0	3

227 rows × 12 columns

In [ ]:
```
data_set
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cou
0	1.375928	103.955003	HaveFun Karaoke & Live Music Bar (Downtown East)	East Region	2.0	10
2	1.333001	103.895961	UB3 Bistro	Central Region	2.0	12
3	1.303198	103.832124	Nam Sing Hokkien Mee	Central Region	2.0	1
4	1.293449	103.851829	Hopscotch (Capitol)	Central Region	4.0	87
5	1.308232	103.885850	Bakes N Bites	Central Region	2.0	2
...	...	...	...	...	...	
11859	1.431854	103.828063	Ho Heng Kway Chap	North Region	2.0	1
11860	1.264767	103.818013	Dimbulah Coffee	Central Region	2.0	4
11862	1.305566	103.791569	Jaguarita's Craft Beer Bar	Central Region	2.0	6
11863	1.455534	103.798947	Hakka Fat Moi Lei Cha Rice	North Region	2.0	
11864	1.362641	103.894010	Goldhill Family Restaurant	North Region	1.0	46

11224 rows × 12 columns

Amenities Analysis: Explore the amenities column, extract unique words, and put them in a single column

In [ ]:
```
# value_counts of amenities

data_set['venue_amenities'].value_counts()
```

Out[ ]:
```
venue_amenities
{}
5673
{'ordering_options': ['delivery']}
4271
{'ordering_options': ['delivery'], 'payments': ['card']}
59
{'offerings': ['alcohol']}
42
{'payments': ['card']}
41


...
{'accessibility': ['wheelchairaccessible'], 'amenities': ['wifi'], 'atmo
sphere': ['cosy', 'casual'], 'crowd': ['groups', 'familyfriendly'], 'din
ing_options': ['seating'], 'food_options': ['vegetarian_options'], 'offe
rings': ['halal_food', 'coffee', 'alcohol'], 'ordering_options': ['deliv
ery'], 'payments': ['card'], 'planning': ['accepts_reservations'], 'type
s_of_alcohol': ['beer', 'wine', 'cocktails', 'hard_liquor']}
1
{'atmosphere': ['cosy', 'casual'], 'crowd': ['groups', 'familyfriendly',
'lgbtq_friendly'], 'offerings': ['coffee', 'alcohol'], 'payments': ['car
d'], 'planning': ['accepts_reservations', 'reservations_required'], 'typ
es_of_alcohol': ['beer', 'wine', 'cocktails', 'hard_liquor']}
1
{'atmosphere': ['casual'], 'crowd': ['groups'], 'dining_options': ['seat
ing'], 'highlights': ['bar_games', 'karaoke', 'live_performances', 'grea
t_cocktails'], 'offerings': ['food', 'alcohol'], 'planning': ['accepts_r
eservations'], 'types_of_alcohol': ['beer', 'wine', 'cocktails', 'hard_l
iquor']}
1
{'amenities': ['wifi'], 'atmosphere': ['cosy', 'casual'], 'crowd': ['gro
ups', 'familyfriendly', 'lgbtq_friendly'], 'dining_options': ['seating']
, 'happy_hour': ['happyhour_drinks'], 'highlights': ['sports', 'great_co
cktails'], 'offerings': ['coffee', 'alcohol'], 'ordering_options': ['del
ivery'], 'payments': ['card'], 'planning': ['accepts_reservations'], 'ty
pes_of_alcohol': ['beer', 'wine', 'cocktails', 'hard_liquor']}
1
{'amenities': ['wifi'], 'atmosphere': ['cosy', 'casual'], 'crowd': ['fam
ilyfriendly', 'groups', 'lgbtq_friendly'], 'dining_options': ['seating']
, 'food_options': ['vegetarian_options'], 'happy_hour': ['happyhour_drin
ks'], 'highlights': ['live_performances'], 'offerings': ['coffee', 'alco
hol'], 'ordering_options': ['delivery'], 'payments': ['card'], 'planning
': ['accepts_reservations', 'reservations_required'], 'types_of_alcohol'
: ['beer', 'wine', 'cocktails', 'hard_liquor']}       1
Name: count, Length: 970, dtype: int64
```

In [ ]:
```
# Convert the 'venue_amenities' dictionaries into separate columns
amenities_expanded = data_set['venue_amenities'].apply(pd.Series)

# Merge the expanded amenities back into the original DataFrame
df_final = pd.concat([data_set.drop(columns=['venue_amenities']), ameniti
```

In [ ]:
```
# Convert lists in the DataFrame to string (or handle them as needed)

for column in amenities_expanded.columns:
    df_final[column] = df_final[column].apply(lambda x: ', '.join(x) if i

df_final


print(amenities_expanded.columns)
```

```
Index(['accessibility', 'atmosphere', 'crowd', 'dining_options',
       'food_options', 'happy_hour', 'highlights', 'offerings', 'payments'
,
       'planning', 'types_of_alcohol', 'ordering_options', 'amenities',
       'popular_for'],
      dtype='object')
```

In [ ]:
```
#fill in NA values with 'Unknown'

df = df_final.fillna('Unknown')
df
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cou
0	1.375928	103.955003	HaveFun Karaoke & Live Music Bar (Downtown East)	East Region	2.0	10
2	1.333001	103.895961	UB3 Bistro	Central Region	2.0	12
3	1.303198	103.832124	Nam Sing Hokkien Mee	Central Region	2.0	1
4	1.293449	103.851829	Hopscotch (Capitol)	Central Region	4.0	87
5	1.308232	103.885850	Bakes N Bites	Central Region	2.0	2
...	...	...	...	...	...	
11859	1.431854	103.828063	Ho Heng Kway Chap	North Region	2.0	1
11860	1.264767	103.818013	Dimbulah Coffee	Central Region	2.0	4
11862	1.305566	103.791569	Jaguarita's Craft Beer Bar	Central Region	2.0	6
11863	1.455534	103.798947	Hakka Fat Moi Lei Cha Rice	North Region	2.0	
11864	1.362641	103.894010	Goldhill Family Restaurant	North Region	1.0	46

11224 rows × 25 columns

In [ ]:
```
import pandas as pd

# Assuming 'df' is your DataFrame and 'payments' is the column of interes
# Now, create a set of all unique words in the 'payments' column
unique_words = set()
df['payments'].str.lower().str.split().apply(unique_words.update)

# The 'unique_words' set now contains all unique words from the 'payments
print(unique_words)
```

```
#do the same for all columns in amenities.columns, and put them in sepera

unique_words = set()

for column in amenities_expanded.columns:
    unique_words = set()
    df[column] = df[column].astype(str)
    df[column].str.lower().str.split().apply(unique_words.update)
    print(unique_words)
```

```
{'checks', 'checks,', 'unknown', 'card'}
{'unknown', 'wheelchairaccessible'}
{'upscale,', 'casual', 'cosy', 'cosy,', 'unknown'}
{'groups', 'familyfriendly', 'groups,', 'lgbtq_friendly,', 'lgbtq_friendly
', 'familyfriendly,', 'unknown'}
{'seating', 'private_dining_room,', 'private_dining_room', 'unknown'}
{'vegan_options,', 'vegan_options', 'quick_bite,', 'organic_dishes,', 'veg
etarian_options', 'healthy_options', 'quick_bite', 'organic_dishes', 'unkn
own'}
{'happyhour_drinks', 'happyhour_food,', 'happyhour_drinks,', 'happyhour_fo
od', 'unknown'}
{'great_cocktails', 'unknown', 'rooftop_seating,', 'dancing', 'trivia_nigh
t', 'trivia_night,', 'rooftop_seating', 'live_performances,', 'bar_games',
'bar_games,', 'sports,', 'live_performances', 'sports', 'karaoke,', 'karao
ke', 'dancing,'}
{'halal_food,', 'food', 'alcohol', 'coffee,', 'food,', 'coffee', 'alcohol,
', 'halal_food', 'all_you_can_eat,', 'unknown'}
{'checks', 'checks,', 'unknown', 'card'}
{'accepts_reservations', 'accepts_reservations,', 'unknown', 'reservations
_required'}
{'beer', 'cocktails', 'wine,', 'hard_liquor,', 'hard_liquor', 'cocktails,'
, 'beer,', 'wine', 'unknown'}
{'delivery', 'unknown'}
{'wifi', 'unknown'}
{'dinner', 'lunch,', 'unknown'}
```

make a new column named amenities, that lists all the amenties together from the columns iaccessibility', 'atmosphere', 'crowd', 'dining_options', 'food_options', 'happy_hour', 'highlights', 'offerings', 'payments', 'planning', 'types_of_alcohol', 'ordering_options'

In [ ]:
```
# add a + ', ' + to each column

df['amenities'] = df['accessibility'] + ', ' + df['atmosphere'] + ', ' +
df
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cou
0	1.375928	103.955003	HaveFun Karaoke & Live Music Bar (Downtown East)	East Region	2.0	10
2	1.333001	103.895961	UB3 Bistro	Central Region	2.0	12
3	1.303198	103.832124	Nam Sing Hokkien Mee	Central Region	2.0	1
4	1.293449	103.851829	Hopscotch (Capitol)	Central Region	4.0	87
5	1.308232	103.885850	Bakes N Bites	Central Region	2.0	2
...	...	...	...	...	...	
11859	1.431854	103.828063	Ho Heng Kway Chap	North Region	2.0	1
11860	1.264767	103.818013	Dimbulah Coffee	Central Region	2.0	4
11862	1.305566	103.791569	Jaguarita's Craft Beer Bar	Central Region	2.0	6
11863	1.455534	103.798947	Hakka Fat Moi Lei Cha Rice	North Region	2.0	
11864	1.362641	103.894010	Goldhill Family Restaurant	North Region	1.0	46

11224 rows × 25 columns

In [ ]:
```
# in the amenities column, remove the word unknown completely

df['amenities'] = df['amenities'].str.replace('Unknown, ', '')

data_set = df
```

Remove Hawker Centres and Fast Food

In [ ]:
```
#can use some textual analysis on venue_amenities, and make a criteria fo

stores = data_set[data_set['venue_subsegment'] == 'Fast Food']

#dropping fast food anyway because none of them have high price point buc

data_set = data_set.drop(data_set[data_set['venue_subsegment'] == 'Fast F

data_set
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cou
0	1.375928	103.955003	HaveFun Karaoke & Live Music Bar (Downtown East)	East Region	2.0	10
2	1.333001	103.895961	UB3 Bistro	Central Region	2.0	12
3	1.303198	103.832124	Nam Sing Hokkien Mee	Central Region	2.0	1
4	1.293449	103.851829	Hopscotch (Capitol)	Central Region	4.0	87
5	1.308232	103.885850	Bakes N Bites	Central Region	2.0	2
...	...	...	...	...	...	
11859	1.431854	103.828063	Ho Heng Kway Chap	North Region	2.0	1
11860	1.264767	103.818013	Dimbulah Coffee	Central Region	2.0	4
11862	1.305566	103.791569	Jaguarita's Craft Beer Bar	Central Region	2.0	6
11863	1.455534	103.798947	Hakka Fat Moi Lei Cha Rice	North Region	2.0	
11864	1.362641	103.894010	Goldhill Family Restaurant	North Region	1.0	46

10977 rows × 25 columns

In [ ]:
```
#drop hawker centers from venue_segment

data_set = data_set.drop(data_set[data_set['venue_segment'] == 'Hawker Fo

data_set
```

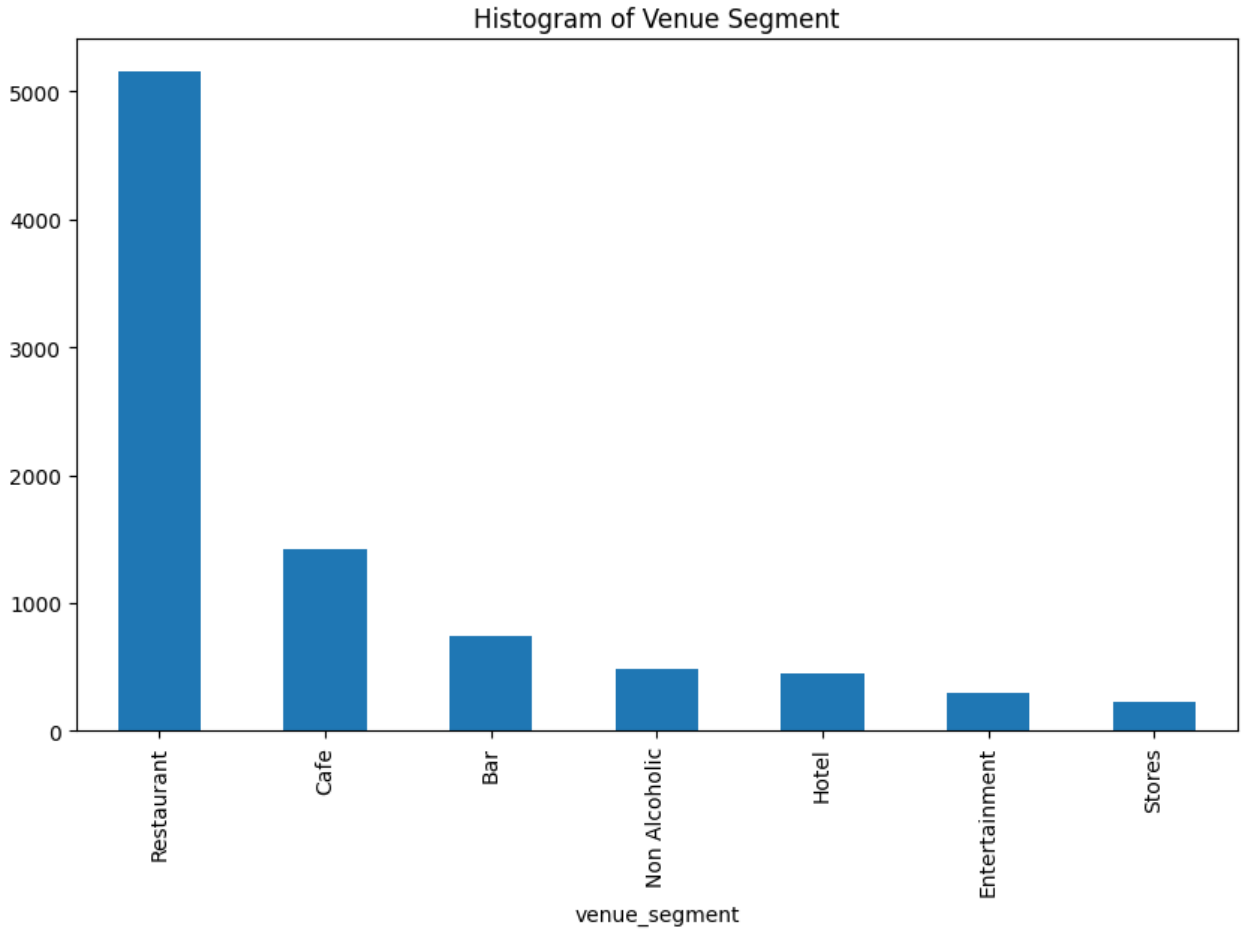
Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cou
0	1.375928	103.955003	HaveFun Karaoke & Live Music Bar (Downtown East)	East Region	2.0	10
2	1.333001	103.895961	UB3 Bistro	Central Region	2.0	12
4	1.293449	103.851829	Hopscotch (Capitol)	Central Region	4.0	87
5	1.308232	103.885850	Bakes N Bites	Central Region	2.0	2
6	1.289597	103.856171	Symphony Cafe Lounge	Central Region	2.0	2
...	...	...	...	...	...	
11856	1.333950	103.962808	ManNa Korean Restaurant	East Region	2.0	22
11857	1.305777	103.828119	Hard Rock Cafe	Central Region	3.0	232
11860	1.264767	103.818013	Dimbulah Coffee	Central Region	2.0	4
11862	1.305566	103.791569	Jaguarita's Craft Beer Bar	Central Region	2.0	6
11864	1.362641	103.894010	Goldhill Family Restaurant	North Region	1.0	46

8776 rows × 25 columns

In [ ]:
```
# give me unique values of venue_segment

data_set['venue_segment'].value_counts()

#make a lhistogram of venue_segment value counts, order by value counts

plt.figure(figsize=(10, 6))
data_set['venue_segment'].value_counts().plot(kind='bar')
```

```
plt.title('Histogram of Venue Segment')
plt.show()

#analyze why stores launch werent great (look later)
```



In [ ]:
```
# give me unique values of venue_subsegment

data_set['venue_subsegment'].value_counts()

#drop venue_subsegment where it is 'Fast FOOD'

data_set = data_set.drop(data_set[data_set['venue_subsegment'] == 'Fast F

data_set['venue_subsegment'].value_counts()
```

Out[ ]:
```
venue_subsegment
Asian            2321
Restaurant       2015
Cafe             1420
Bar               747
Western           495
Non Alcoholic     484
Hotel             452
Entertainment     295
Alcohol           216
Halal             169
Vegetarian        128
Fine Dining        23
Stores             11
Name: count, dtype: int64
```

In [ ]:
```
data_set
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cou
0	1.375928	103.955003	HaveFun Karaoke & Live Music Bar (Downtown East)	East Region	2.0	10
2	1.333001	103.895961	UB3 Bistro	Central Region	2.0	12
4	1.293449	103.851829	Hopscotch (Capitol)	Central Region	4.0	87
5	1.308232	103.885850	Bakes N Bites	Central Region	2.0	2
6	1.289597	103.856171	Symphony Cafe Lounge	Central Region	2.0	2
...	...	...	...	...	...	
11856	1.333950	103.962808	ManNa Korean Restaurant	East Region	2.0	22
11857	1.305777	103.828119	Hard Rock Cafe	Central Region	3.0	232
11860	1.264767	103.818013	Dimbulah Coffee	Central Region	2.0	4
11862	1.305566	103.791569	Jaguarita's Craft Beer Bar	Central Region	2.0	6
11864	1.362641	103.894010	Goldhill Family Restaurant	North Region	1.0	46

8776 rows × 25 columns

4. Feature Engineering

Categorizing platform_category: Simplify platform_category into broader categories for easier analysis.

In [ ]:
```
#unique values in platform_category
```

```
data_set['platform_category'].value_counts()

# how many of these have the word restaurant in them

#make a new column called is_restaurant, where it is 'restaurant; if plat

data_set['is_restaurant'] = data_set['platform_category'].apply(lambda x:

data_set
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cou
0	1.375928	103.955003	HaveFun Karaoke & Live Music Bar (Downtown East)	East Region	2.0	10
2	1.333001	103.895961	UB3 Bistro	Central Region	2.0	12
4	1.293449	103.851829	Hopscotch (Capitol)	Central Region	4.0	87
5	1.308232	103.885850	Bakes N Bites	Central Region	2.0	2
6	1.289597	103.856171	Symphony Cafe Lounge	Central Region	2.0	2
...	...	...	...	...	...	
11856	1.333950	103.962808	ManNa Korean Restaurant	East Region	2.0	22
11857	1.305777	103.828119	Hard Rock Cafe	Central Region	3.0	232
11860	1.264767	103.818013	Dimbulah Coffee	Central Region	2.0	4
11862	1.305566	103.791569	Jaguarita's Craft Beer Bar	Central Region	2.0	6
11864	1.362641	103.894010	Goldhill Family Restaurant	North Region	1.0	46

8776 rows × 26 columns

In [ ]:
```
# find me all unique values that have the word 'restaurant' in them in pl

data_set['platform_category'].value_counts()
```

Out[ ]:
```
platform_category
Cafe                            1414
Restaurant                      1359
Bar                              488
Bakery                           484
Hotel                            433
                                ...
Dance club                         1
Modern European restaurant         1
Soul food restaurant               1
Kosher restaurant                  1
Video karaoke                      1
Name: count, Length: 116, dtype: int64
```

In [ ]:
```
# find me all unique values that have the word 'restaurant' in them in pl

data_set['is_restaurant'].value_counts()

#filter out rows where is_restaurant is restaurant or Restaurant

df= data_set[(data_set['is_restaurant'] == 'restaurant') | (data_set['is_

# 'categorized all restaurants into one category, since most of them were
```

In [ ]:
```
data_set

#now within the is_restaurant column, rreplace non restaurant values with

data_set['is_restaurant'] = data_set['is_restaurant'].apply(lambda x: 'pu

data_set
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cou
0	1.375928	103.955003	HaveFun Karaoke & Live Music Bar (Downtown East)	East Region	2.0	10
2	1.333001	103.895961	UB3 Bistro	Central Region	2.0	12
4	1.293449	103.851829	Hopscotch (Capitol)	Central Region	4.0	87
5	1.308232	103.885850	Bakes N Bites	Central Region	2.0	2
6	1.289597	103.856171	Symphony Cafe Lounge	Central Region	2.0	2
...	...	...	...	...	...	
11856	1.333950	103.962808	ManNa Korean Restaurant	East Region	2.0	22
11857	1.305777	103.828119	Hard Rock Cafe	Central Region	3.0	232
11860	1.264767	103.818013	Dimbulah Coffee	Central Region	2.0	4
11862	1.305566	103.791569	Jaguarita's Craft Beer Bar	Central Region	2.0	6
11864	1.362641	103.894010	Goldhill Family Restaurant	North Region	1.0	46

8776 rows × 26 columns

In [ ]:
```
data_set['is_restaurant'].unique()

#replace Restaurant, japanese steakhouse, steak house, bistro, takeout re

data_set['is_restaurant'] = data_set['is_restaurant'].replace({'Restauran

#replace resort hotel, hotel, hotel bar with hotel
```

```
data_set['is_restaurant'] = data_set['is_restaurant'].replace({'Resort ho

data_set['is_restaurant'].unique()
```

Out[ ]:
```
array(['pub/bar', 'restaurant', 'Bakery', 'Cafe', 'Beer garden', 'hotel'
,
       'Live music venue', 'Wine store', 'Noodle shop', 'Liquor store',
       'Night club', 'Disco club', 'Beer store', 'Concert hall',
       'Brewery', 'Event venue', 'Coffee roasters', 'Cafeteria',
       'Dance club', 'Winery', 'Art cafe', 'Video karaoke', 'Wine cellar
'],
      dtype=object)
```

In [ ]:
```
data_set['is_restaurant'].value_counts()

#put these Liquor store, Wine store, Beer store, Coffee roasters, Wine ce

data_set['is_restaurant'] = data_set['is_restaurant'].replace({'Liquor st

data_set['is_restaurant'].value_counts()
```

Out[ ]:
```
is_restaurant
restaurant          5065
Cafe                1414
pub/bar              939
Bakery               484
hotel                452
retail               230
Night club            68
Disco club            31
Noodle shop           30
Live music venue      22
Beer garden           13
Brewery               10
Concert hall           8
Event venue            3
Winery                 2
Art cafe               2
Cafeteria              1
Dance club             1
Video karaoke          1
Name: count, dtype: int64
```

In [ ]:
```
#put these into live entertainment Live music venue, Concert hall, Event

data_set['is_restaurant'] = data_set['is_restaurant'].replace({'Live musi

data_set['is_restaurant'].value_counts()

#classify beer garden, night club, disco club, dance club in pubs/bars

data_set['is_restaurant'] = data_set['is_restaurant'].replace({'Beer gard

data_set['is_restaurant'].value_counts()
```

```
#drop rows with is_restaurant as Noodle shop

data_set = data_set.drop(data_set[data_set['is_restaurant'] == 'Noodle sh

#classify Cafeteria and Art Cafe as Cafe

data_set['is_restaurant'] = data_set['is_restaurant'].replace({'Cafe': 'c

data_set['is_restaurant'].value_counts()
```

Out[ ]:
```
is_restaurant
restaurant            5065
cafe                  1417
pub/bar               1054
Bakery                 484
hotel                  452
retail                 230
live entertainment      44
Name: count, dtype: int64
```

In [ ]:
```
data_set['venue_segment'].value_counts()
```

Out[ ]:
```
venue_segment
Restaurant       5121
Cafe             1420
Bar               747
Non Alcoholic     484
Hotel             452
Entertainment     295
Stores            227
Name: count, dtype: int64
```

In [ ]:
```
#change is_restaurant to venue_type

data_set = data_set.rename(columns={'is_restaurant': 'venue_type'})
data_set
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cou
0	1.375928	103.955003	HaveFun Karaoke & Live Music Bar (Downtown East)	East Region	2.0	10
2	1.333001	103.895961	UB3 Bistro	Central Region	2.0	12
4	1.293449	103.851829	Hopscotch (Capitol)	Central Region	4.0	87
5	1.308232	103.885850	Bakes N Bites	Central Region	2.0	2
6	1.289597	103.856171	Symphony Cafe Lounge	Central Region	2.0	2
...	...	...	...	...	...	
11856	1.333950	103.962808	ManNa Korean Restaurant	East Region	2.0	22
11857	1.305777	103.828119	Hard Rock Cafe	Central Region	3.0	232
11860	1.264767	103.818013	Dimbulah Coffee	Central Region	2.0	4
11862	1.305566	103.791569	Jaguarita's Craft Beer Bar	Central Region	2.0	6
11864	1.362641	103.894010	Goldhill Family Restaurant	North Region	1.0	46

8746 rows × 26 columns

Log Transformation of review_count: Create a new column for the log transformation of review_count to normalize its distribution.

In [ ]:
```
df = data_set

import numpy as np
df['log_review_count'] = df['review_count'].apply(lambda x: np.log(x) if
```

```
df
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cou
0	1.375928	103.955003	HaveFun Karaoke & Live Music Bar (Downtown East)	East Region	2.0	10
2	1.333001	103.895961	UB3 Bistro	Central Region	2.0	12
4	1.293449	103.851829	Hopscotch (Capitol)	Central Region	4.0	87
5	1.308232	103.885850	Bakes N Bites	Central Region	2.0	2
6	1.289597	103.856171	Symphony Cafe Lounge	Central Region	2.0	2
...	...	...	...	...	...	
11856	1.333950	103.962808	ManNa Korean Restaurant	East Region	2.0	22
11857	1.305777	103.828119	Hard Rock Cafe	Central Region	3.0	232
11860	1.264767	103.818013	Dimbulah Coffee	Central Region	2.0	4
11862	1.305566	103.791569	Jaguarita's Craft Beer Bar	Central Region	2.0	6
11864	1.362641	103.894010	Goldhill Family Restaurant	North Region	1.0	46

8746 rows × 27 columns

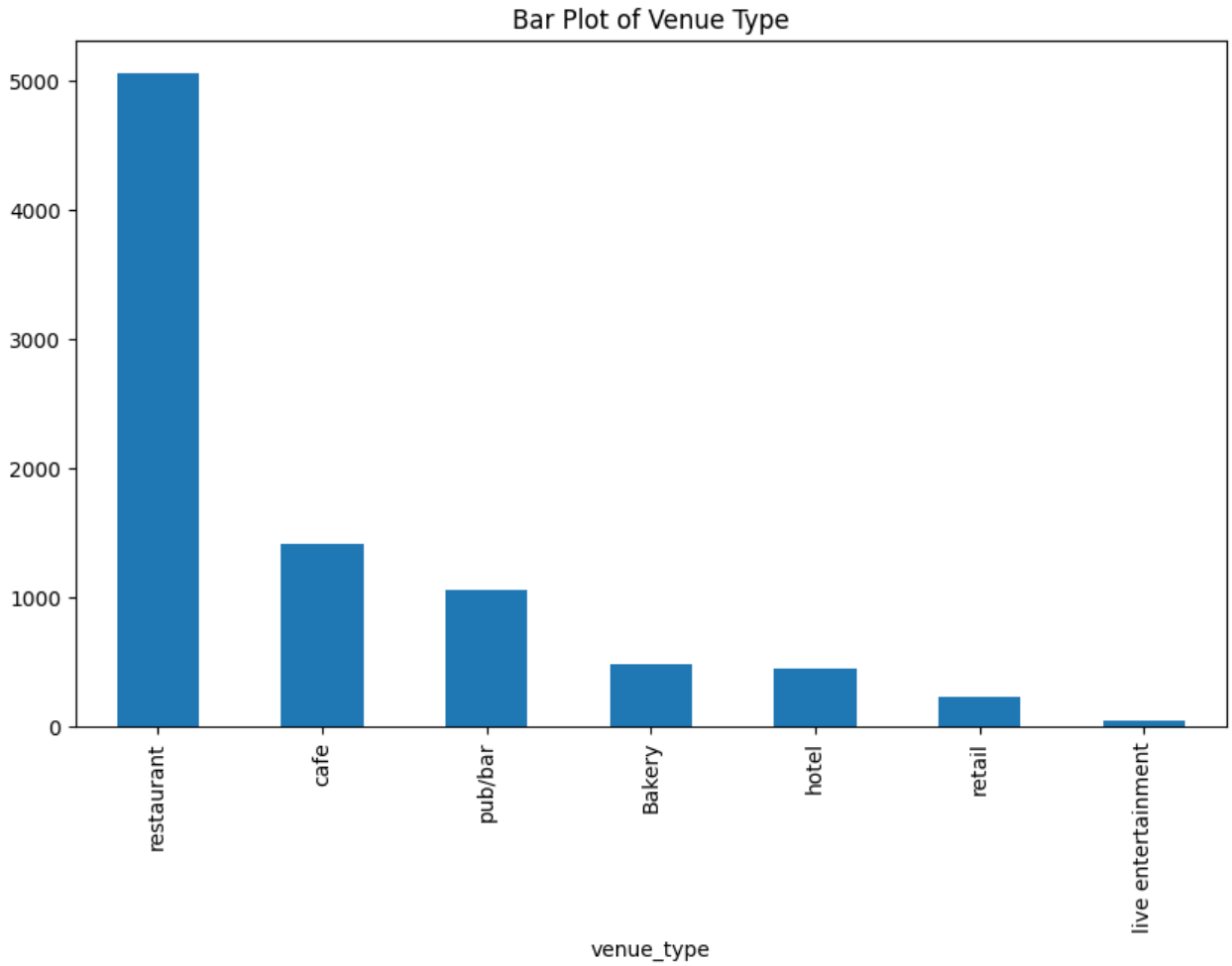
In [ ]:
```
#show me unique values of venue_segment

df['venue_subsegment'].value_counts()
```

Out[ ]:
```
venue_subsegment
Asian              2321
Restaurant         1985
Cafe               1420
Bar                 747
Western             495
Non Alcoholic       484
Hotel               452
Entertainment       295
Alcohol             216
Halal               169
Vegetarian          128
Fine Dining          23
Stores               11
Name: count, dtype: int64
```

In [ ]:
```
#show unique values of platform_category

df['venue_type'].value_counts()
```

Out[ ]:
```
venue_type
restaurant            5065
cafe                  1417
pub/bar               1054
Bakery                 484
hotel                  452
retail                 230
live entertainment      44
Name: count, dtype: int64
```

In [ ]:
```
#bar chart of venue type

plt.figure(figsize=(10, 6))

df['venue_type'].value_counts().plot(kind='bar')

plt.title('Bar Plot of Venue Type')

plt.show()
```



In [ ]:
```
#box and whisker plot of log review count

plt.figure(figsize=(10, 6))
sns.boxplot(df['log_review_count'])
plt.title('Box and Whisker Plot of Log Review Count')
plt.show()
```

Adjust Amenties columns and drop the expanded ones

In [ ]:
```
#in the amenities column, remove repeated words in the string
df['amenities'] = df['amenities'].apply(lambda x: ', '.join(set(x.split('

#remove the word 'Unknown' from the amenities column

df['amenities'] = df['amenities'].apply(lambda x: x.replace('Unknown', ''


#fil in empty strings with 'Unknown'

df['amenities'] = df['amenities'].apply(lambda x: 'Unknown' if x == '' el


df
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cou
0	1.375928	103.955003	HaveFun Karaoke & Live Music Bar (Downtown East)	East Region	2.0	10
2	1.333001	103.895961	UB3 Bistro	Central Region	2.0	12
4	1.293449	103.851829	Hopscotch (Capitol)	Central Region	4.0	87
5	1.308232	103.885850	Bakes N Bites	Central Region	2.0	2
6	1.289597	103.856171	Symphony Cafe Lounge	Central Region	2.0	2
...	...	...	...	...	...	
11856	1.333950	103.962808	ManNa Korean Restaurant	East Region	2.0	22
11857	1.305777	103.828119	Hard Rock Cafe	Central Region	3.0	232
11860	1.264767	103.818013	Dimbulah Coffee	Central Region	2.0	4
11862	1.305566	103.791569	Jaguarita's Craft Beer Bar	Central Region	2.0	6
11864	1.362641	103.894010	Goldhill Family Restaurant	North Region	1.0	46

8746 rows × 27 columns

In [ ]:
```
#drop all the amenities columns except amenities

df = df.drop(['accessibility', 'atmosphere', 'crowd', 'dining_options', '
```

Removing price point bucket of 1 to focus on higher end venues

In [ ]:
```
#drop that have price point bucket as 1 and 2
```

```
df = df.drop(df[df['price_point_bucket'] == 1].index)
df = df.drop(df[df['price_point_bucket'] == 2].index)

df
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cou
4	1.293449	103.851829	Hopscotch (Capitol)	Central Region	4.0	87
27	1.311722	103.861329	Lola Faye Cafe	Central Region	3.0	21
31	1.278633	103.844331	Levant Bar	Central Region	4.0	46
33	1.285558	103.845015	Nanbantei Japanese Restaurant (Chinatown Point)	Central Region	4.0	51
37	1.305368	103.832537	Crossroads Cafe	Central Region	3.0	45
...	...	...	...	...	...	
11836	1.279720	103.841448	Salud	Central Region	4.0	2
11843	1.264217	103.820358	Ramen Hitoyoshi	Central Region	4.0	61
11852	1.296713	103.855219	Last Word	Central Region	3.0	6
11854	1.289872	103.838575	Mixology Salon Singapore	Central Region	4.0	3
11857	1.305777	103.828119	Hard Rock Cafe	Central Region	3.0	232

1324 rows × 15 columns

5. Data Preparation for Scoring

In [ ]:
```
#import the data to a csv file

df.to_csv('cleaned_data.csv', index=False)
```

In [ ]:
```
#get the list of unique amenities

amenities = set()
df['amenities'].str.lower().str.split(',').apply(amenities.update)

amenities
```

Out[ ]:
```
{'',
 ' ',
 ' accepts_reservations',
 ' alcohol',
 ' all_you_can_eat',
 ' bar_games',
 ' beer',
 ' card',
 ' casual',
 ' checks',
 ' cocktails',
 ' coffee',
 ' cosy',
 ' dancing',
 ' delivery',
 ' familyfriendly',
 ' food',
 ' great_cocktails',
 ' groups',
 ' halal_food',
 ' happyhour_drinks',
 ' happyhour_food',
 ' hard_liquor',
 ' healthy_options',
 ' karaoke',
 ' lgbtq_friendly',
 ' live_performances',
 ' organic_dishes',
 ' private_dining_room',
 ' reservations_required',
 ' rooftop_seating',
 ' seating',
 ' sports',
 ' trivia_night',
 ' upscale',
 ' vegan_options',
 ' vegetarian_options',
 ' wheelchairaccessible',
 ' wine',
'accepts_reservations',
'alcohol',
'card',
'coffee',
'delivery',
'food',
'great_cocktails',
'happyhour_drinks',
'reservations_required',
'unknown',
'wine'}
```

In [ ]:
```
import ast
```

```
# Function to extract first four reviews and compute the average rating
def process_reviews(reviews):
    if isinstance(reviews, str):
        try:
            reviews = ast.literal_eval(reviews)  # Convert string to list
        except (SyntaxError, ValueError):
            return ["", "", "", "", None]  # If conversion fails, return

    if not isinstance(reviews, list) or reviews == {} or not reviews:
        return ["", "", "", "", None]  # If not a valid list or empty dic

    review_descriptions = [r.get('description', '') for r in reviews[:4]]
    while len(review_descriptions) < 4:
        review_descriptions.append("")  # Fill missing reviews with empty

    avg_rating = sum(r.get('rating', 0) for r in reviews if 'rating' in r
    return review_descriptions + [avg_rating]

# Apply the function to create new columns
df[['review_1', 'review_2', 'review_3', 'review_4', 'average_review_ratin
    lambda x: pd.Series(process_reviews(x))
)

df.drop('review_sample', axis=1, inplace=True)

df.head()
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_count
4	1.293449	103.851829	Hopscotch (Capitol)	Central Region	4.0	874.0
27	1.311722	103.861329	Lola Faye Cafe	Central Region	3.0	212.0
31	1.278633	103.844331	Levant Bar	Central Region	4.0	460.0
33	1.285558	103.845015	Nanbantei Japanese Restaurant (Chinatown Point)	Central Region	4.0	512.0
37	1.305368	103.832537	Crossroads Cafe	Central Region	3.0	454.0

Q1: Opportunity Size, maybe put your own filters based on some rationale.

In [ ]:
```
#remove rows with unknown amenities

df = df.drop(df[df['amenities'] == 'Unknown'].index)
```

In [ ]:
```
from sentence_transformers import SentenceTransformer, util
from sklearn.preprocessing import MinMaxScaler
import numpy as np
import pandas as pd

# Define Boozeless categories for NLP classification
boozeless_categories = {
    "Young, Mild & Free": (
        "Trendy, healthy, fresh, organic, vegan-friendly, casual, relaxed
        "Bright, open, modern, social, airy, eco-friendly, cozy, earthy.
        "Juice bars, smoothie bowls, acai, salads, plant-based meals, oat
        "Popular for brunch, post-workout, cycling stops, weekend meetups
    ),
    "Aesthetically Anchored": (
```

```
        "Chic, stylish, glamorous, Instagrammable, elegant, upscale, luxu
        "Dim-lit, neon-lit, romantic, artistic, sleek, polished, curated,
        "Cocktail bars, rooftop lounges, fine dining, velvet seating, amb
        "Popular for birthdays, anniversary dinners, date nights, special
    ),
    "Haute Cuisine": (
        "Refined, exclusive, gourmet, sophisticated, Michelin-starred, ex
        "Intimate, elegant, polished, fine-dining, high-end, handcrafted,
        "Multi-course tasting menus, truffle, caviar, wagyu, seafood, win
        "Known for world-class chefs, signature dishes, elite dining expe
    )
}

# Load the NLP model for similarity comparison
model = SentenceTransformer("all-MiniLM-L6-v2")

# Convert category descriptions into embeddings
boozeless_category_embeddings = {
    category: model.encode(description, convert_to_tensor=True)
    for category, description in boozeless_categories.items()
}

# Adjusted threshold for classification
AMENITIES_ONLY_THRESHOLD = 0.2  # Lower threshold for venues with no revi

# Function to classify a venue based on amenities + review samples
def classify_venue_adjusted(venue_row):
    texts = [venue_row["amenities"]]  # Start with amenities

    # Include review samples if available
    has_reviews = False
    for i in range(1, 5):
        review_col = f"review_{i}"
        if review_col in venue_row and isinstance(venue_row[review_col],
            texts.append(venue_row[review_col])
            has_reviews = True

    # Combine available text for classification
    combined_text = " ".join(texts).lower()
    venue_embedding = model.encode(combined_text, convert_to_tensor=True)

    # Compute similarity scores with each category
    scores = {
        category: util.pytorch_cos_sim(venue_embedding, category_embeddin
        for category, category_embedding in boozeless_category_embeddings
    }

    # Use a lower threshold for venues with no reviews
    threshold = 0.3 if has_reviews else AMENITIES_ONLY_THRESHOLD

    # Determine the best matching category & corresponding similarity sco
    best_fit_category = max(scores, key=scores.get) if max(scores.values(
    similarity_score = max(scores.values()) if max(scores.values()) > thr
```

```
    return best_fit_category, similarity_score

# Apply classification function to DataFrame
df[["boozeless_category", "similarity_score"]] = df.apply(classify_venue_

# Drop venues classified as "Not a Fit"
df = df[df["boozeless_category"] != "Not a Fit"]

# Normalize review count using log transformation
df["log_review_count"] = df["review_count"].apply(lambda x: np.log1p(x))

# Normalize values using Min-Max Scaling
scaler = MinMaxScaler()
df["normalized_review_count"] = scaler.fit_transform(df[["log_review_coun
df["normalized_rating"] = scaler.fit_transform(df[["average_rating"]].fil

# Assign price score based on price point bucket
price_mapping = {4: 1.0, 3: 0.8, 2: 0.5, 1: 0.2}  # Adjusting for high-en
df["price_score"] = df["price_point_bucket"].map(price_mapping).fillna(0.

# Compute amenities match score (count of high-end amenities)
df["amenities_score"] = df["amenities"].apply(lambda x: len(x.split(",")))
df["normalized_amenities_score"] = scaler.fit_transform(df[["amenities_sc

# Normalize similarity score for fair weighting
df["normalized_similarity_score"] = scaler.fit_transform(df[["similarity_

# Compute final weighted score (Now includes similarity score)
df["final_score"] = (
    0.20 * df["normalized_review_count"] +  # Weight for reviews
    0.20 * df["normalized_rating"] +        # Weight for rating
    0.20 * df["price_score"] +              # Weight for price alignment
    0.20 * df["normalized_amenities_score"] + # Weight for relevant ameni
    0.20 * df["normalized_similarity_score"]  # Weight for NLP category s
)

# Sort by final score and select the top 25
top_25_venues = df.sort_values(by="final_score", ascending=False).head(25

# Display the top 25 venues
top_25_venues
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cc
2872	1.304765	103.825157	Manhattan	Central Region	4.0	8
4758	1.359800	103.990393	Coucou Hotpot·Brew Tea @Jewel	East Region	4.0	39
6920	1.293345	103.853090	JAAN By Kirk Westaway	Central Region	4.0	4

2596	1.300951	103.859978	Cappadocia Turkish & Mediterranean Restaurant	Central Region	4.0	51
11398	1.291102	103.840262	Restaurant JAG	Central Region	3.0	2
5172	1.286251	103.848922	必定 Coincidence Bar & Restaurant	Central Region	4.0	
4093	1.295557	103.851949	Whitegrass Restaurant	Central Region	4.0	6
1137	1.278877	103.813289	1918 Heritage Bar	Central Region	4.0	4
9026	1.301869	103.835899	Shisen Hanten by Chen Kentaro	Central Region	3.0	15
2702	1.279548	103.843759	Tippling Club	Central Region	4.0	6
9803	1.283071	103.853012	VUE	Central Region	4.0	13
11504	1.306578	103.829612	MERCI MARCEL ORCHARD	Central Region	4.0	20
5668	1.332269	103.892106	Tora Tora Tora Japanese Restaurant Singapore	Central Region	4.0	4
3499	1.289707	103.851443	Odette	Central Region	4.0	10
9868	1.294175	103.855072	The NCO Club	Central Region	4.0	8
11342	1.311511	103.952498	High Tide Bistro and Bar	East Region	4.0	6
10777	1.286434	103.848337	Restaurant Ibid	Central Region	4.0	7
8716	1.305802	103.810095	Candlenut	Central Region	4.0	15
8013	1.290820	103.860090	Colony	Central Region	4.0	17
3647	1.289413	103.851254	Hachi Restaurant	Central Region	4.0	2
			La Saigon			

4424	1.312237	103.924361	(Viet Specialty Coffee)	East Region	3.0	3
7627	1.308128	103.834130	Alma by Juan Amador	Central Region	4.0	4
7203	1.279373	103.843054	Rhubarb	Central Region	4.0	3
4	1.293449	103.851829	Hopscotch (Capitol)	Central Region	4.0	8
9145	1.313845	103.857285	Podi & Poriyal	Central Region	3.0	8

25 rows × 28 columns

In [ ]:
```
df["boozeless_category"].value_counts()
```

Out[ ]:
```
boozeless_category
Haute Cuisine             345
Aesthetically Anchored     94
Young, Mild & Free         77
Name: count, dtype: int64
```

In [ ]:
```
top_25_venues["boozeless_category"].value_counts()
```

Out[ ]:
```
boozeless_category
Haute Cuisine             13
Aesthetically Anchored     7
Young, Mild & Free         5
Name: count, dtype: int64
```

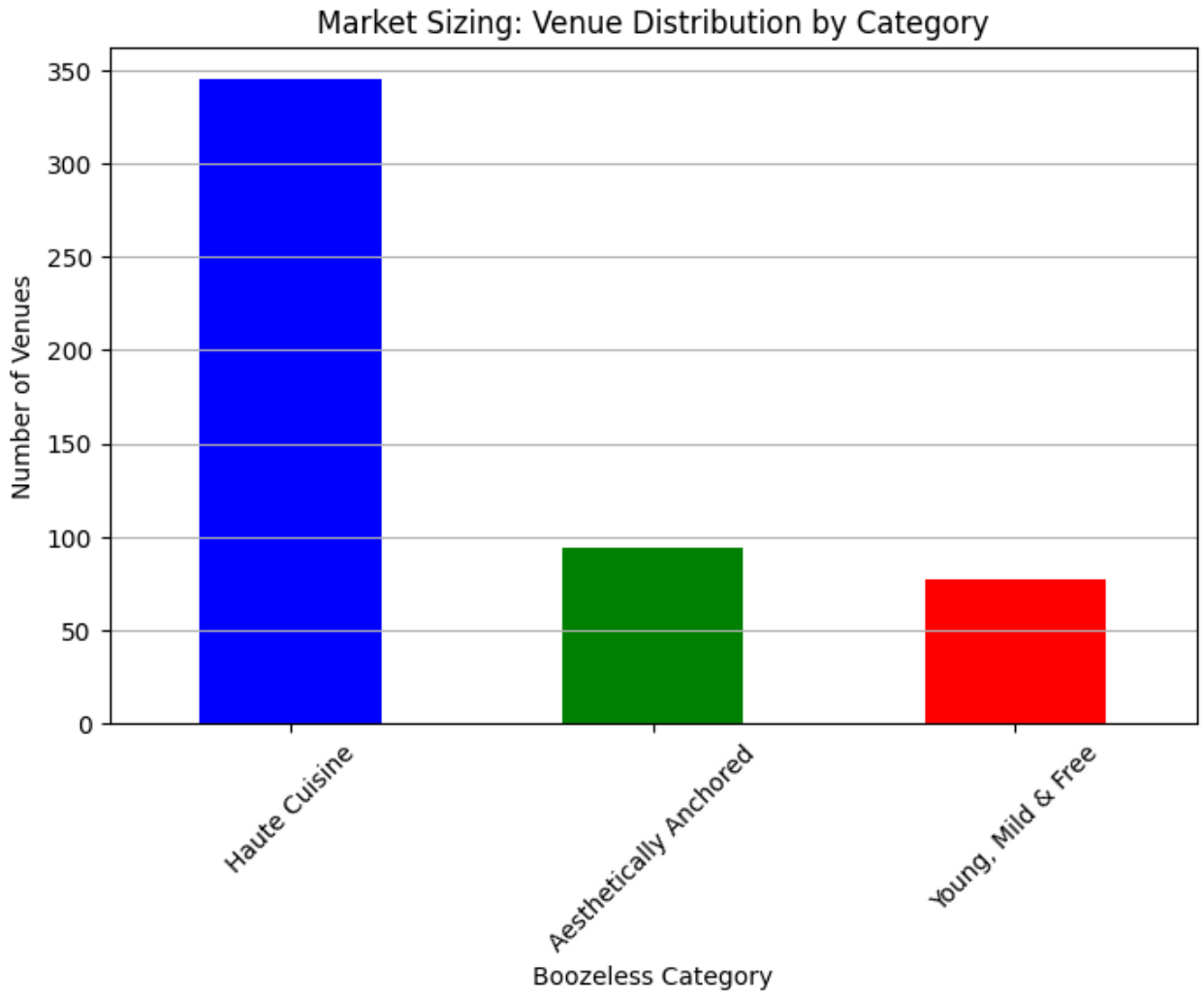
In [ ]:
```
#print first review of the first row

# print(top_25_venues.iloc[0])

#make  dataframe getting top 1 venue from each category

top_1_venues = top_25_venues.groupby("boozeless_category").first().reset_

# Display the top venue from each category
top_1_venues

#show name, category, rating, review count, price point bucket, amenities

top_1_venues[["venue_name", "boozeless_category", "average_rating", "revi

#give full review_1 of furst roiw

print(top_1_venues.iloc[1]["review_1"])
```

10/10 love the atmosphere. Feels chill and luxurious vibe. Soup that we had - tomato was great and the pork/chicken soup was nice but quite peppery (saw the peppercorn) but very "gao" like collagen soup.

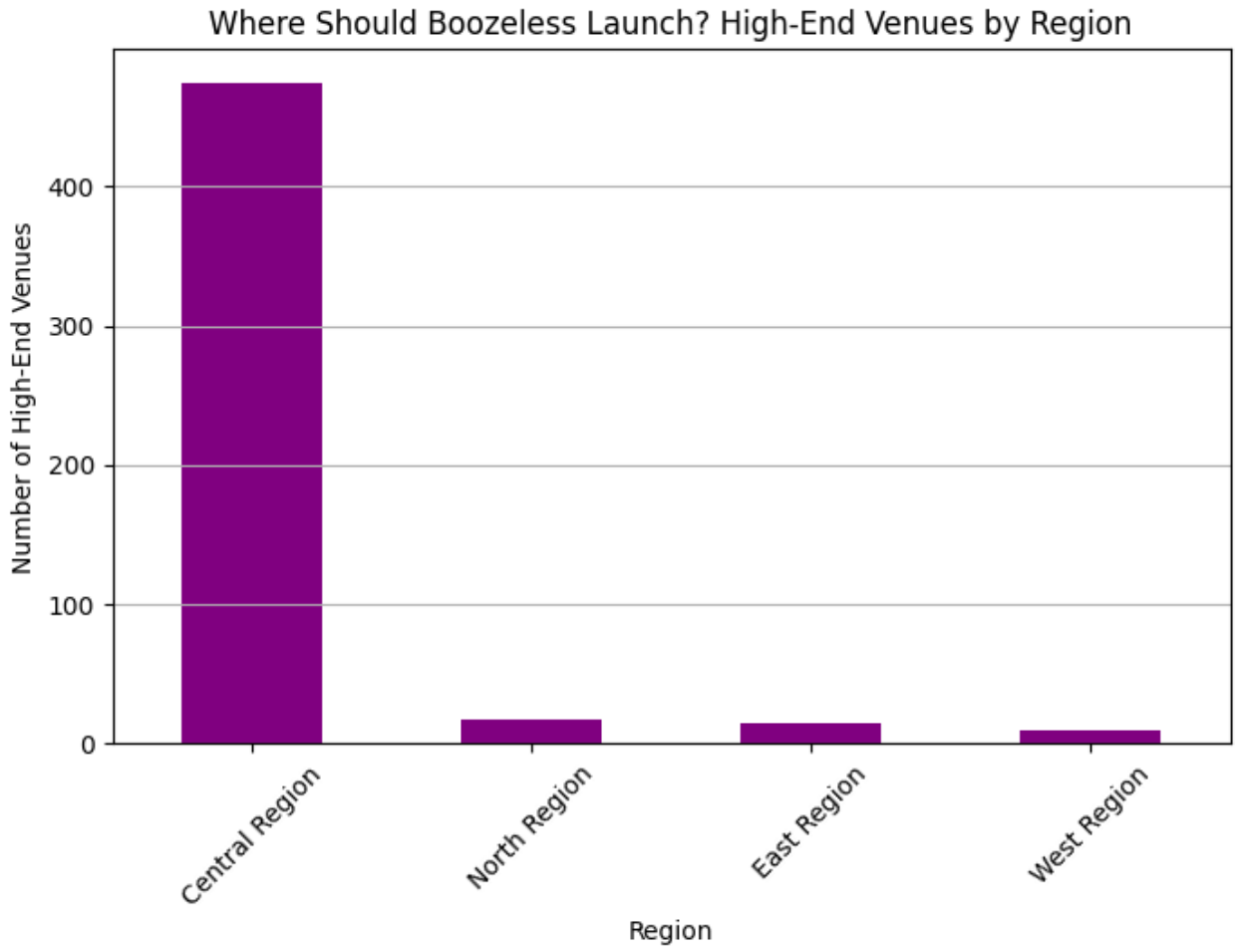
Meat came with vege so don't order the cabbage - had to takeaway my cabbage.
Cod fish was good!
The tea they serve is really good as well

Jasper who is our server was attentive and helped with what we needed.

In [ ]:
```
import matplotlib.pyplot as plt

# Count venues in each category
category_counts = df["boozeless_category"].value_counts()

# Plot the bar chart
plt.figure(figsize=(8, 5))
category_counts.plot(kind="bar", color=["blue", "green", "red"])
plt.xlabel("Boozeless Category")
plt.ylabel("Number of Venues")
plt.title("Market Sizing: Venue Distribution by Category")
plt.xticks(rotation=45)
plt.grid(axis="y")

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



In [ ]:
```
region_counts = df[df["boozeless_category"] != "Not a Fit"]["region"].val

plt.figure(figsize=(8, 5))
region_counts.plot(kind="bar", color="purple")
plt.xlabel("Region")
plt.ylabel("Number of High-End Venues")
plt.title("Where Should Boozeless Launch? High-End Venues by Region")
plt.xticks(rotation=45)
plt.grid(axis="y")

plt.show()
```



In [ ]:
```
#review rows with not a fit

df[df['boozeless_category'] == 'Not a Fit']

#drop rows with not a fit

df = df.drop(df[df['boozeless_category'] == 'Not a Fit'].index)

#limitation of the model is that it is not able to classify venues with n
```

In [ ]:
```
# from sklearn.preprocessing import MinMaxScaler

# # Normalize review count using log transformation
# df["log_review_count"] = df["review_count"].apply(lambda x: np.log1p(x)

# # Normalize values using Min-Max Scaling
# scaler = MinMaxScaler()
# df["normalized_review_count"] = scaler.fit_transform(df[["log_review_co
# df["normalized_rating"] = scaler.fit_transform(df[["average_rating"]].f

# # Assign price score based on price point bucket
# price_mapping = {4: 1.0, 3: 0.8, 2: 0.5, 1: 0.2}  # Adjusting for high-
# df["price_score"] = df["price_point_bucket"].map(price_mapping).fillna(

# # Compute amenities match score (count of high-end amenities)
```

```
# df["amenities_score"] = df["amenities"].apply(lambda x: len(x.split(","
# df["normalized_amenities_score"] = scaler.fit_transform(df[["amenities_

# # Compute final weighted score
# df["final_score"] = (
#     0.25 * df["normalized_review_count"] +
#     0.25 * df["normalized_rating"] +
#     0.25 * df["price_score"] +
#     0.25 * df["normalized_amenities_score"]
# )

# # Sort by final score and select the top 25
# top_25_venues = df.sort_values(by="final_score", ascending=False).head(

# # Display the top 25 venues
# top_25_venues
```

Out[ ]:

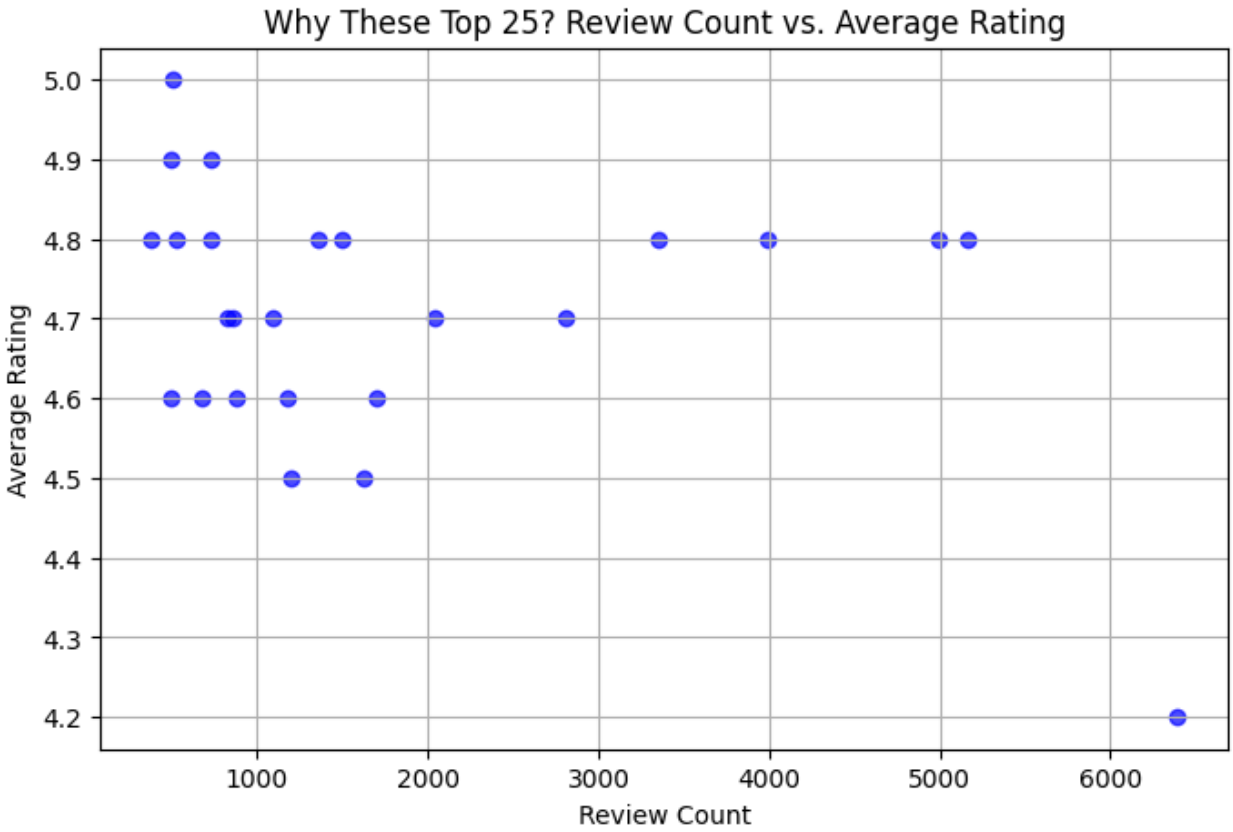
	latitude	longitude	venue_name	region	price_point_bucket	review_co
2596	1.300951	103.859978	Cappadocia Turkish & Mediterranean Restaurant	Central Region	4.0	51
9803	1.283071	103.853012	VUE	Central Region	4.0	13
5668	1.332269	103.892106	Tora Tora Tora Japanese Restaurant Singapore	Central Region	4.0	4
4758	1.359800	103.990393	Coucou Hotpot·Brew Tea @Jewel	East Region	4.0	39
2872	1.304765	103.825157	Manhattan	Central Region	4.0	8
7687	1.284693	103.861023	CÉ LA VI Singapore: Restaurant, SkyBar & Club ...	Central Region	4.0	63
7562	1.300540	103.849252	BISOUX • Coffee by Light • Cocktails by Night	Central Region	4.0	11
9618	1.301838	103.835827	Ginger.Lily	Central Region	4.0	7
5569	1.330109	103.744756	L Bistro	West Region	4.0	5
3066	1.303748	103.809972	The Dempsey	Central	3.0	16

			Project	Region		
8013	1.290820	103.860090	Colony	Central Region	4.0	17
11504	1.306578	103.829612	MERCI MARCEL ORCHARD	Central Region	4.0	20
6935	1.289460	103.844165	Oche Riverside Point - Singapore	Central Region	4.0	3
2330	1.316273	103.880894	Stickies Bar @ Aljunied	Central Region	4.0	8
10199	1.243359	103.829131	Panamericana	Central Region	3.0	33
4	1.293449	103.851829	Hopscotch (Capitol)	Central Region	4.0	8
3499	1.289707	103.851443	Odette	Central Region	4.0	10
10777	1.286434	103.848337	Restaurant Ibid	Central Region	4.0	7
9536	1.334346	103.742316	TANYU Westgate	West Region	4.0	28
7871	1.302920	103.872867	Poulet - Kallang Wave Mall	Central Region	4.0	49
5154	1.281803	103.850125	Artemis Grill & Sky Bar	Central Region	4.0	11
7700	1.300955	103.861628	PizzaFace	Central Region	4.0	4
2320	1.330109	103.744756	YC Dining & Bar | Western Restaurant Bar	West Region	4.0	5
11342	1.311511	103.952498	High Tide Bistro and Bar	East Region	4.0	6
8934	1.354879	103.830864	San Ren Xing 三人行 - Thomson Plaza	Central Region	4.0	15

25 rows × 28 columns

In [ ]:
```
plt.figure(figsize=(8, 5))
```

```
plt.scatter(top_25_venues["review_count"], top_25_venues["average_rating"
plt.xlabel("Review Count")
plt.ylabel("Average Rating")
plt.title("Why These Top 25? Review Count vs. Average Rating")
plt.grid(True)

# Annotate high-end venues
for i, row in top_25_venues.iterrows():
    if row["average_rating"] > 4.5 and row["review_count"] < 100:
        plt.annotate(row["venue_name"], (row["review_count"], row["averag

plt.show()
```



I am going to put all required amenities into a single column so theres no overlap whe counting scores

In [ ]:
```
#value counts of categories in venue_segment

top_25_venues['venue_segment'].value_counts()
```

Out[ ]:
```
venue_segment
Restaurant       19
Bar               4
Cafe              1
Entertainment     1
Name: count, dtype: int64
```

In [ ]:
```
top_25_venues.describe()
```

Out[ ]:

	latitude	longitude	price_point_bucket	review_count	average_rating	l
count	25.000000	25.000000	25.000000	25.000000	25.000000	
mean	1.305376	103.846881	3.920000	1818.960000	4.708000	
std	0.024983	0.054436	0.276887	1676.563511	0.163095	
min	1.243359	103.742316	3.000000	382.000000	4.200000	
25%	1.289707	103.829612	4.000000	728.000000	4.600000	
50%	1.301838	103.850125	4.000000	1173.000000	4.700000	
75%	1.316273	103.861023	4.000000	2041.000000	4.800000	
max	1.359800	103.990393	4.000000	6392.000000	5.000000	

Region Proportion Normalization: Calculate and normalize the proportion of venues per region, aiming for a representative sample of 25 venues.

In [ ]:
```
#find proportion of venues in each region

top_25_venues['region'].value_counts(normalize=True)
```

Out[ ]:
```
region
Central Region    0.80
West Region       0.12
East Region       0.08
Name: proportion, dtype: float64
```

In [ ]:
```
#use folium to plot the top 25 venues on the map

import folium

# Create a map centered at the mean latitude and longitude

m = folium.Map(location=[top_25_venues['latitude'].mean(), top_25_venues[

# Add markers for each row in the DataFrame

for index, row in top_25_venues.iterrows():
    folium.Marker([row['latitude'], row['longitude']], popup=row['venue_n

# Display the map

m
```

Out[ ]:

1. Enhance Demographic Analysis

Age and Income Data: Find and augment HDB and condo datasets with demographic data indicating age distribution, average income levels. Perhaps, HDB and condo may have data on age distribution and rent per square ft etc.

Young, affluent areas are likely to align with the "Young, Mild and Free" and "Haute Cuisine" categories (to set a criteria)

Unique or Upscale residential areas might align more with "Aesthetically Anchored" venues.

2. Integrate Lifestyle and Consumption Patterns

Lifestyle Indicators: Look for public data on lifestyle indicators such as gym memberships, organic grocery stores, and cultural venues. These can help identify areas where residents are likely to value health, aesthetics, and culinary innovation, corresponding to our venue categories.

3. Map Existing F&B Venues Venue Location Data: Utilize existing current F&B venues.

4. Define Criteria for Venue Selection For Young, Mild and Free, prioritize areas with a high density of gyms, sports facilities, or health food stores.

For Aesthetically Anchored, focus on neighborhoods known for their design sensibilities, art galleries, or unique architectural landmarks (looks difficult unless image processing)

For Haute Cuisine, look for areas with high-income levels, luxury condos, and a reputation for fine dining.

5. Conduct Spatial Analysis

Radius Analysis: For each identified high-potential area, use QGIS to perform a radius analysis around key points of interest (e.g., luxury condos, cultural hubs) to pinpoint specific venues within a certain distance.

Overlay Analysis: Combine layers of demographic data, lifestyle indicators, and existing F&B venues to visually identify overlaps that signal prime locations for Boozeless placements.

6. Venue Prioritization Rank identified venues based on a scoring system that accounts for proximity to target demographics, alignment with venue categories, and potential for high foot traffic or visibility.

In [ ]:
```
#top 25 venues

top_25_venues["boozeless_category"].value_counts()
```

Out[ ]:
```
boozeless_category
Haute Cuisine             13
Aesthetically Anchored     7
Young, Mild & Free         5
Name: count, dtype: int64
```

In [ ]:
```
top_25_venues.describe()

#plot average_rating against average_review rating

# Re-attempting the visualization
```

Out[ ]:

	latitude	longitude	price_point_bucket	review_count	average_rating	l
count	25.000000	25.000000	25.000000	25.000000	25.000000	
mean	1.305376	103.846881	3.920000	1818.960000	4.708000	
std	0.024983	0.054436	0.276887	1676.563511	0.163095	
min	1.243359	103.742316	3.000000	382.000000	4.200000	
25%	1.289707	103.829612	4.000000	728.000000	4.600000	
50%	1.301838	103.850125	4.000000	1173.000000	4.700000	
75%	1.316273	103.861023	4.000000	2041.000000	4.800000	
max	1.359800	103.990393	4.000000	6392.000000	5.000000	

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

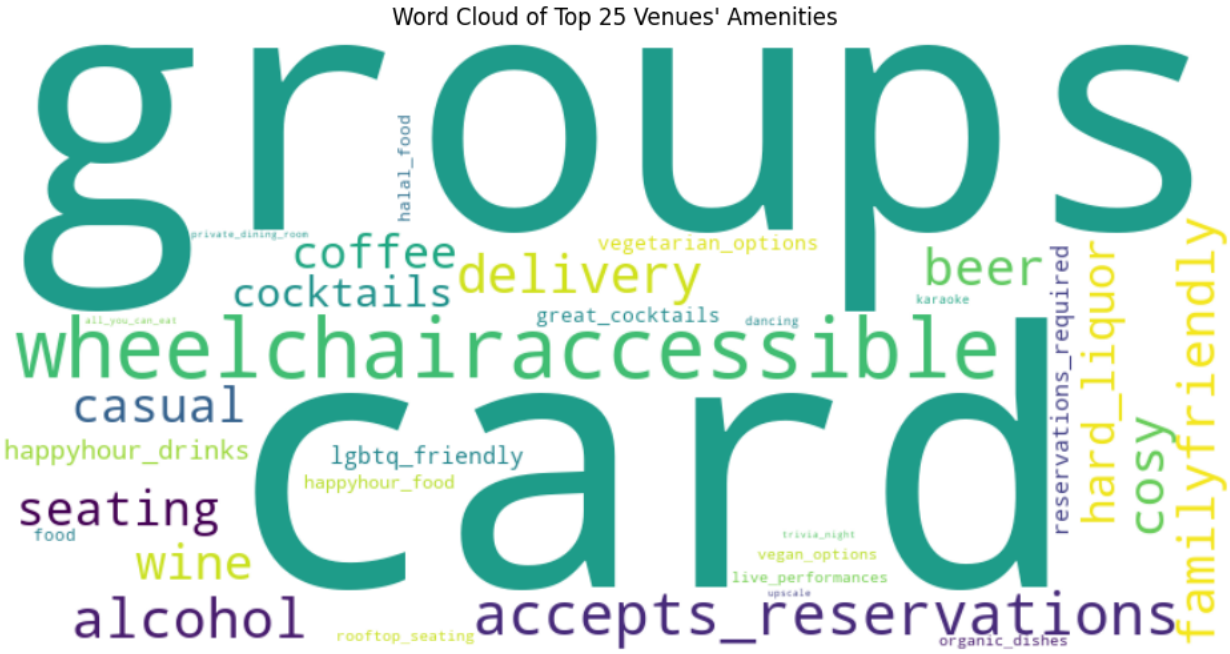
```
from collections import Counter

# Ensure that the amenities column is treated as lists
top_25_venues["amenities"] = top_25_venues["amenities"].apply(lambda x: x

# Flatten the list of amenities
all_amenities = [amenity.strip() for sublist in top_25_venues["amenities"

# Count occurrences of each amenity
amenity_counts = Counter(all_amenities)

# Convert to DataFrame for better visualization
amenity_df = pd.DataFrame(amenity_counts.items(), columns=["Amenity", "Co

amenity_df
```

Out[ ]:

	Amenity	Count
2	groups	25
10	card	25
1	accepts_reservations	25
17	alcohol	23
19	beer	23
14	cosy	22
4	wheelchairaccessible	22
9	seating	22
18	wine	21
21	hard_liquor	20
5	delivery	18
20	cocktails	18
3	familyfriendly	17
8	casual	17
0	happyhour_drinks	16
7	coffee	16
22	great_cocktails	11
13	vegetarian_options	11
16	reservations_required	11
24	lgbtq_friendly	10
11	happyhour_food	7

25	food	6
15	organic_dishes	5
6	vegan_options	5
23	rooftop_seating	4
12	halal_food	3
27	live_performances	3
26	dancing	2
32	bar_games	2
28	trivia_night	1
29	karaoke	1
30	checks	1
31	all_you_can_eat	1
33	private_dining_room	1
34	upscale	1

In [ ]:
```
import matplotlib.pyplot as plt
from wordcloud import WordCloud

# Create a dictionary from the amenities count
amenities_count = {
    "card": 25, "groups": 24, "wheelchairaccessible": 23, "accepts_reserv
    "alcohol": 21, "delivery": 19, "cosy": 18, "familyfriendly": 18, "bee
    "wine": 17, "casual": 16, "seating": 16, "coffee": 15, "hard_liquor":
    "happyhour_drinks": 10, "lgbtq_friendly": 8, "reservations_required":
    "vegetarian_options": 6, "great_cocktails": 6, "happyhour_food": 5, "
    "food": 4, "vegan_options": 3, "organic_dishes": 3, "rooftop_seating"
    "live_performances": 3, "dancing": 2, "karaoke": 2, "upscale": 1, "al
    "trivia_night": 1, "private_dining_room": 1
}

# Generate the word cloud
wordcloud = WordCloud(width=800, height=400, background_color="white").ge

# Display the word cloud
plt.figure(figsize=(12, 6))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title("Word Cloud of Top 25 Venues' Amenities")
plt.show()
```



Accessibility & Convenience Are Prioritized

The most common amenity is "card" (25 venues), indicating that all top venues accept card payments, making them more accessible to high-end consumers. "Wheelchair accessible" (23 venues) and "accepts reservations" (23 venues) highlight that these venues are well-prepared for diverse clientele, ensuring inclusivity and planning flexibility.

Social & Group-Friendly Atmosphere

"Groups" (24 venues) and "family-friendly" (18 venues) suggest that the majority of these venues cater to social gatherings. The presence of "seating" (16 venues) and "casual" (16 venues) reinforces that these venues provide a welcoming ambiance.

Strong Alcohol & Cocktail Culture

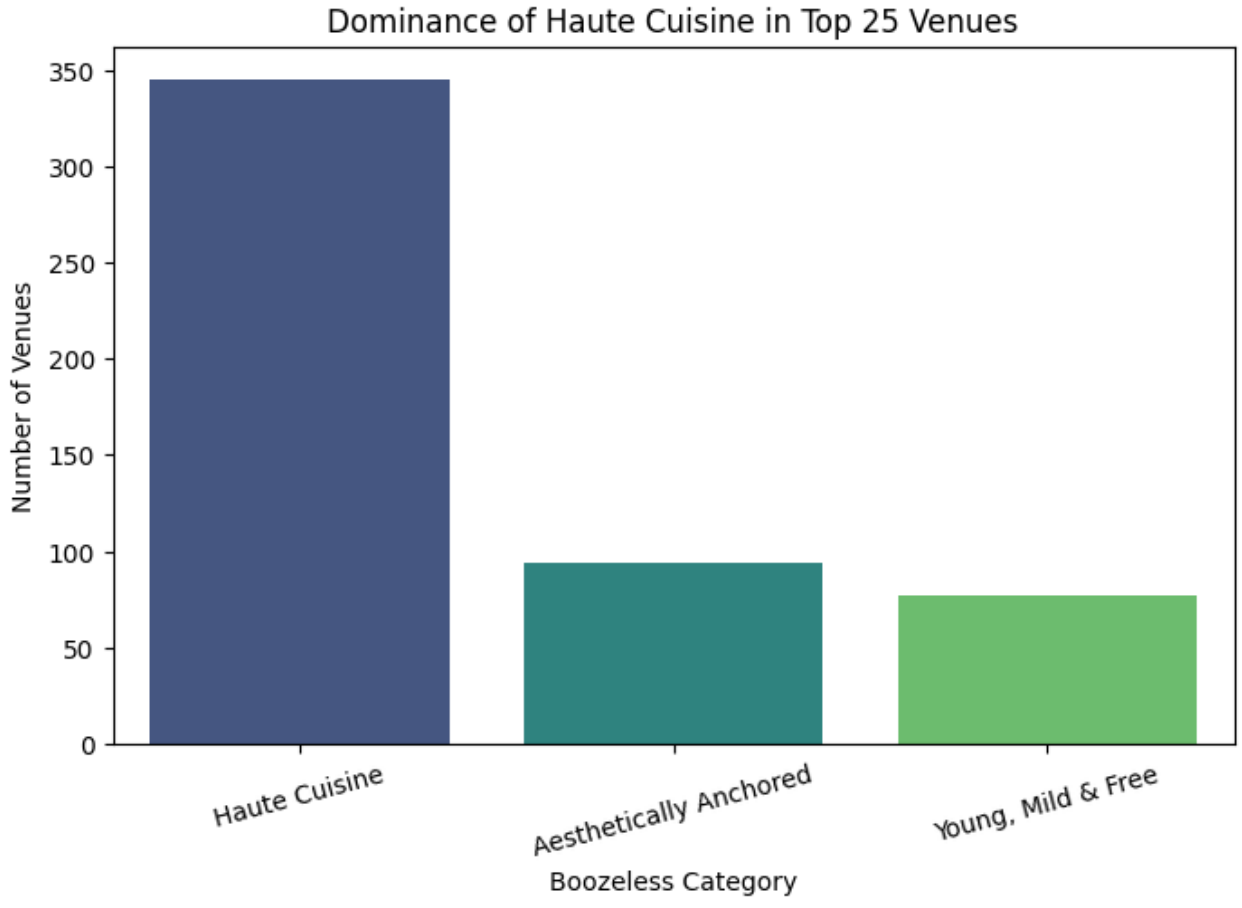
Despite being a non-alcoholic brand, Boozeless should note that "alcohol" (21 venues), "beer" (18 venues), "wine" (17 venues), and "hard liquor" (14 venues) are prominent in these top venues. However, "cocktails" (12 venues) and "great cocktails" (6 venues) indicate an opportunity—Boozeless can position itself within the cocktail culture by offering premium non-alcoholic alternatives.

Potential Marketing Hooks for Boozeless

"Happy hour drinks" (10 venues) and "happy hour food" (5 venues) suggest that promotional pricing and unique drink offerings can be a key strategy. "LGBTQ-friendly" (8 venues) and "vegan options" (3 venues) show an emerging demand for inclusivity and alternative dining experiences.

In [ ]:
```
# Reattempting the visualizations

# Plot 1: Distribution of Boozeless Categories in Top 25
plt.figure(figsize=(8, 5))
sns.barplot(x=category_counts.index, y=category_counts.values, palette="v
plt.xlabel("Boozeless Category")
plt.ylabel("Number of Venues")
plt.title("Dominance of Haute Cuisine in Top 25 Venues")
plt.xticks(rotation=15)
plt.show()
```

```
/var/folders/td/bwzm0ch95gv3n67hkws8_vwm0000gn/T/ipykernel_2702/3293424387
.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for
the same effect.

  sns.barplot(x=category_counts.index, y=category_counts.values, palette="
viridis")
```



In [ ]:
```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Combine all review texts into a single string
all_reviews_text = ' '.join(top_25_venues[['review_1', 'review_2', 'revie
                            .dropna().astype(str).values.flatten())
```

```
# Generate word cloud
wordcloud_reviews = WordCloud(width=800, height=400, background_color='wh

# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud_reviews, interpolation="bilinear")
plt.axis("off")
plt.title("Word Cloud of Reviews in the Top 25 Venues")
plt.show()
```

1️⃣ Service is a Key Factor

Words like "service," "staff," "friendly," "attentive" dominate the reviews. This suggests that exceptional service is a major reason why these venues are rated highly. Boozeless should prioritize venues known for great hospitality, as their customer base values the overall experience, not just the drinks.

2️⃣ Food Quality and Experience Matter

Frequent mentions of "food," "experience," "dishes," "delicious," "fresh," "quality," "menu" indicate that high-end venues focus on culinary excellence. Boozeless should ensure their product integrates seamlessly into gourmet offerings, as customers at these venues are likely to appreciate unique and sophisticated beverage options.

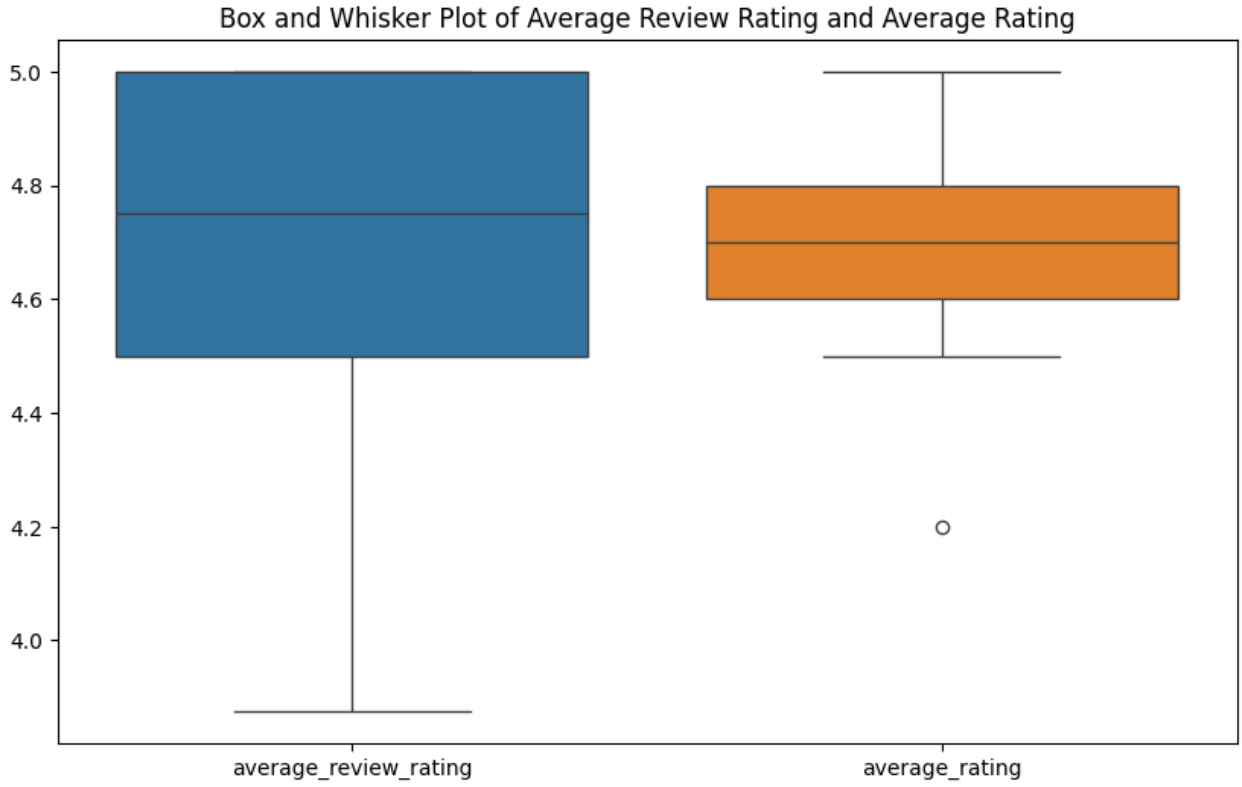
3️⃣ Beverage Opportunities Exist

The presence of "bar," "cocktail," "drink," "coffee," "tea," "wine," "beer" implies that beverages are a core part of the dining experience. This validates the opportunity for Boozeless to position its product as a sophisticated non-alcoholic alternative in high-end venues.

In [ ]:
```
top_25_venues["average_review_rating"]
```

Out[ ]:
```
2596     5.000
9803     4.875
5668     4.375
4758     5.000
2872     4.875
7687     4.750
7562     4.250
9618     5.000
5569     5.000
3066     4.250
8013     4.750
11504    4.875
6935     4.875
2330     3.875
10199    5.000
4        4.250
3499     5.000
10777    5.000
9536     4.750
7871     4.875
5154     4.625
7700     4.750
2320     4.375
11342    4.750
8934     4.500
Name: average_review_rating, dtype: float64
```

In [ ]:
```
#boxplot of average review rating and average_rating

plt.figure(figsize=(10, 6))
sns.boxplot(data=top_25_venues[['average_review_rating', 'average_rating'
plt.title('Box and Whisker Plot of Average Review Rating and Average Rati
plt.show()
```



Higher Ratings from NLP Review Analysis

The median NLP-extracted review rating (~4.9) is higher than the median traditional average rating (~4.7). This suggests that customers express more enthusiasm and positivity in reviews than what is reflected in star ratings.

Greater Variability in NLP Ratings

The NLP-based ratings have a wider spread (4.2 - 5.0) compared to the more compact distribution of star ratings. This could indicate that textual reviews capture a broader range of customer sentiment nuances, while star ratings may be more conservative.

Presence of Outliers in Both Metrics

Both distributions have a few low-end outliers, suggesting that some venues receive occasional negative feedback, but this does not significantly impact overall sentiment.

In [ ]:
```
#show me the venue with the least average_Rating

top_25_venues[top_25_venues['average_rating'] == top_25_venues['average_r
```

Out[ ]:

	latitude	longitude	venue_name	region	price_point_bucket	review_cour
7687	1.284693	103.861023	CÉ LA VI Singapore: Restaurant, SkyBar & Club ...	Central Region	4.0	6392

1 rows × 28 columns

In [ ]:
```
top_25_venues["venue_type"].value_counts()

#make bar plot of venue type

plt.figure(figsize=(10, 6))
sns.barplot(top_25_venues['venue_type'].value_counts())
plt.title('Bar Plot of Venue Type')
plt.show()
```