COGNIFY

INTERNSHIP PROGRAM DATA ANALYSIS

**LEVEL 1**

**TASK 1 TOP CUISINES**

**Determine the top three most common cuisines in the dataset.**

top\_three\_cuisines = data['Cuisines'].value\_counts()[0:3]

top\_three\_cuisines

**A screenshot of a computer code

Description automatically generated**

**Calculate the percentage of restaurants that serve each of the top.**

top\_cuisines = ['North Indian', 'North Indian, Chinese', 'Chinese']

cuisine\_counts = {}

for cuisine in top\_cuisines:

unique\_restaurants = data[data['Cuisines'].str.contains(cuisine, na=False)]['Restaurant Name'].nunique()

cuisine\_counts[cuisine] = unique\_restaurants

total\_restaurants = data['Restaurant Name'].nunique()

for cuisine, count in cuisine\_counts.items():

percentage = (count / total\_restaurants) \* 100

print(f"{cuisine}: {percentage:.2f}% of restaurants")

**A white background with black text

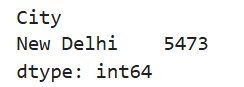
Description automatically generated**

**Task 2 : City Analysis**

**Identify the city with the highest number of restaurants in the dataset**

city\_counts = data.groupby('City').size().sort\_values(ascending=False).head(1)

city\_counts



top\_city = city\_counts.index[0]

top\_city\_count = city\_counts.iloc[0]

print(f"The city with the highest number of restaurants is {top\_city} with {top\_city\_count} restaurants.")



**Calculate the average rating for restaurants in each city**

average\_ratings = data.groupby('City')['Aggregate rating'].mean()

average\_ratings

A screenshot of a computer

Description automatically generated

rating\_counts = data.groupby('City')['Rating text'].value\_counts()

rating\_counts

A screenshot of a computer

Description automatically generated

**Determine the city with the highest average rating?**

highest\_avg\_city = average\_ratings.idxmax()

highest\_avg\_rating = average\_ratings.max()

print(f"The city with the highest average rating is {highest\_avg\_city} with an average rating of {highest\_avg\_rating:}")



**Task: Price Range Distribution**

**Create a histogram or bar chart to visualize the distribution of price ranges among the restaurants?**

import matplotlib.pyplot as plt

top\_10\_expensive\_restaurants = data.sort\_values(by='Price range', ascending=False).head(10)

print(top\_10\_expensive\_restaurants[['Restaurant Name','Price range']])

A screenshot of a computer

Description automatically generated

import seaborn as sns

price\_counts = data['Price range'].value\_counts()

plt.figure(figsize=(5, 3))

sns.countplot(x='Price range', data=data, hue='Price range', palette="viridis",legend=False)

plt.xlabel("Price Range")

plt.ylabel("Number of Restaurants")

plt.title("Distribution of Price Ranges Among Restaurants")

plt.show()

A bar graph with different colored bars

Description automatically generated

**Calculate the percentage of restaurants in each price range category**

price\_counts = data['Price range'].value\_counts()

total\_restaurants = len(data)

price\_percentages = (price\_counts / total\_restaurants) \* 100

for price\_range, percentage in price\_percentages.items():

print(f"Price range {price\_range}: {percentage:.2f}% of restaurants")

A number of numbers and percentages

Description automatically generated

**Task: Online Delivery**

**Determine the percentage of restaurants that offer online delivery**

# Count the total number of restaurants

total\_restaurants = data['Restaurant Name'].nunique()

# Count the number of restaurants that offer online delivery

online\_delivery\_count = data[data['Has Online delivery'] == 'Yes']['Restaurant Name'].nunique()

# Calculate the percentage

online\_delivery\_percentage = (online\_delivery\_count / total\_restaurants) \* 100

print(f"Percentage of restaurants that offer online delivery: {online\_delivery\_percentage:.2f}%")

****

**Compare the average ratings of restaurants with and without online delivery**

# Calculate the average rating for restaurants with online delivery

avg\_rating\_with\_delivery = data[data['Has Online delivery'] == 'Yes']['Aggregate rating'].mean()

# Calculate the average rating for restaurants without online delivery

avg\_rating\_without\_delivery = data[data['Has Online delivery'] == 'No']['Aggregate rating'].mean()

print(f"Average rating for restaurants with online delivery: {avg\_rating\_with\_delivery:.2f}")

print(f"Average rating for restaurants without online delivery: {avg\_rating\_without\_delivery:.2f}")



**LEVEL 2**

**TASK 1 : RESTAURANT RATING**

**I used MYSQL in level 2 except for geographical analysis**

**Analyze the distribution of aggregate ratings and determine the most common rating range**

SELECT

FLOOR(`Aggregate rating`) AS rating\_range,

COUNT(\*) AS num\_restaurants

FROM

new\_schema.Data

GROUP BY

rating\_range

ORDER BY

num\_restaurants DESC;

A screenshot of a computer

Description automatically generated

**Calculate the average number of votes received by restaurants**

SELECT

'Restaurant Name',

AVG('Aggregate rating') AS 'Average Rating',

COUNT(\*) AS 'Number of Locations'

FROM

data

GROUP BY

'Restaurant Name'

ORDER BY

'Average Rating' DESC,

'Number of Locations' DESC;

A screen shot of a computer

Description automatically generated

**Task: Cuisine Combination**

**Identify the most common combinations of cuisines in the dataset.**

SELECT

Cuisines,

COUNT(\*) AS combination\_count

FROM

data

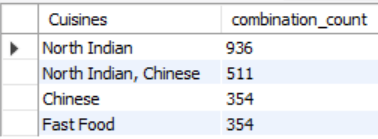
GROUP BY

Cuisines

ORDER BY

combination\_count DESC

LIMIT 4;



**Determine if certain cuisine combinations tend to have higher ratings**

SELECT

Cuisines,

COUNT(\*) AS combination\_count,

AVG(`Aggregate rating`) AS avg\_rating

FROM

data

GROUP BY

Cuisines

ORDER BY

avg\_rating DESC, combination\_count DESC

LIMIT 4;

A screenshot of a computer

Description automatically generated

**Task: Geographic Analysis**

**Plot the locations of restaurants on a map using longitude and latitude coordinates**

import plotly.express as px

fig = px.scatter\_mapbox(data,

lat="Latitude",

lon="Longitude",

text="Restaurant Name",

zoom=10,

mapbox\_style="open-street-map")

# Show the interactive map

fig.show()

A map of the world

Description automatically generated

**Identify any patterns or clusters of restaurants in specific areas**

from sklearn.cluster import KMeans

import plotly.express as px

# Assume data contains columns 'latitude' and 'longitude'

# Apply KMeans clustering with a chosen number of clusters (e.g., 5)

kmeans = KMeans(n\_clusters=4) # No random\_state specified

kmeans.fit(data[['Latitude', 'Longitude']])

# Add cluster labels to the DataFrame

data['cluster'] = kmeans.labels\_

# Plot clusters on a map

fig = px.scatter\_mapbox(data,

lat='Latitude',

lon='Longitude',

color='cluster',

mapbox\_style="carto-positron",

title="Restaurant Clusters")

fig.show()

**A map of the world with different colored countries/regions

Description automatically generated**

**Task: Restaurant Chains**

**Identify if there are any restaurant chains present in the dataset**

SELECT

`Restaurant Name`,

COUNT(\*) AS occurrence\_count

FROM

data

GROUP BY

`Restaurant Name`

HAVING

COUNT(\*) > 1

ORDER BY

occurrence\_count DESC

LIMIT 10;

**A screenshot of a computer

Description automatically generated**

**Analyze the ratings and popularity of different restaurant chains**

SELECT

`Restaurant Name`,

AVG(`Aggregate rating`) AS avg\_rating,

SUM(`Votes`) AS total\_votes,

COUNT(\*) AS number\_of\_locations

FROM

data

GROUP BY

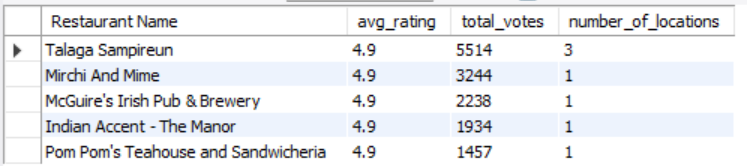
`Restaurant Name`

ORDER BY

avg\_rating DESC,

total\_votes DESC

LIMIT 5;



**Level 3**

**Task: Restaurant Reviews**

**Analyze the text reviews to identify the most common positive and negative keywords**

# Get the counts of each unique value in the 'rating text' column

keyword\_counts = data['Rating text'].value\_counts()

# Convert to DataFrame for easier visualization

keyword\_counts\_df = keyword\_counts.reset\_index()

keyword\_counts\_df.columns = ['Keyword', 'Count']

# Display the result

print(keyword\_counts\_df)

**A close up of a number

Description automatically generated**

**Calculate the average length of reviews and explore if there is a relationship between review length and rating**

# Step 1: Create a new column for review length

data['review\_length'] = data['Rating text'].str.len()

# Step 2: Calculate the overall average review length

average\_review\_length = data['review\_length'].mean()

print(f"Overall Average Review Length: {average\_review\_length:.2f} characters")

# Step 3: Calculate the average review length by rating category

average\_length\_by\_rating = data.groupby('Rating text')['review\_length'].mean().reset\_index()

average\_length\_by\_rating.columns = ['Rating text', 'Average Review Length']

# Display the result

print(average\_length\_by\_rating)

# Optional: Visualize the relationship

import matplotlib.pyplot as plt

plt.figure(figsize=(5, 3))

plt.bar(average\_length\_by\_rating['Rating text'], average\_length\_by\_rating['Average Review Length'], color='skyblue')

plt.xlabel('Rating text')

plt.ylabel('Average Review Length (characters)')

plt.title('Average Review Length by Rating text')

plt.xticks(rotation=45)

plt.show()

A graph of blue bars with white text

Description automatically generated

**Task: Votes Analysis**

**Identify the restaurants with the highest and lowest number of votes. Analyze if there is a correlation between the number of votes and the rating of a restaurant**

# Restaurant with the highest number of votes

highest\_votes = data.loc[data['Votes'].idxmax(), ['Restaurant Name', 'Votes']]

print("Restaurant with the highest number of votes:")

print(highest\_votes)

# Restaurant with the lowest number of votes

lowest\_votes = data.loc[data['Votes'].idxmin(), ['Restaurant Name', 'Votes']]

print("\nRestaurant with the lowest number of votes:")

print(lowest\_votes)

A white background with black text

Description automatically generated

# Calculate the correlation between Votes and Aggregate rating

correlation = data['Votes'].corr(data['Aggregate rating'])

print(f"Correlation between Votes and Aggregate rating: {correlation:.2f}")

****

**Task: Price Range vs. Online Delivery and Table Booking**

**Analyze if there is a relationship between the price range and the availability of online delivery and table booking.**

print(data['Has Online delivery'].value\_counts())

print(data['Has Table booking'].value\_counts())

A screenshot of a computer code

Description automatically generated

# Group by both columns and calculate the mean price range for each combination

grouped\_data = data.groupby(['Has Online delivery', 'Has Table booking'])['Price range'].mean()

# Display the result

print(grouped\_data)

A screenshot of a computer

Description automatically generated

import seaborn as sns

import matplotlib.pyplot as plt

# Boxplot for Price range by Has Online delivery and Has Table booking

plt.figure(figsize=(8, 3))

sns.boxplot(data=data, x='Has Online delivery', y='Price range', hue='Has Table booking')

plt.title('Price Range vs. Online Delivery and Table Booking')

plt.show()

A diagram of a delivery service

Description automatically generated

**Determine if higher-priced restaurants are more likely to offer these services**

# Replace 'Yes' with 1 and 'No' with 0

data[['Has Online delivery', 'Has Table booking']] = data[['Has Online delivery', 'Has Table booking']].replace({'Yes': 1, 'No': 0}).astype(int)

# Group by 'price range' and calculate the mean (proportion) of 'Has Online delivery' and 'Has Table booking'

service\_proportion = data.groupby('Price range')[['Has Online delivery', 'Has Table booking']].mean()

# Display the result

print(service\_proportion)

A number on a white background

Description automatically generated