COGNIFYZ - LEVEL 1 & 2.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
import folium

Data = pd.read_csv(r"C:\Users\Kartik Kumar\Downloads\Dataset .csv", delimiter=',', quotechar='"', on_bad_lines='skip')
print(Data.head(20))
print(Data.info())
print(Data.describe())
print(Data['Cuisines'].isnull().sum())
print(Data.dropna(subset=['Cuisines']))
Data = Data.drop_duplicates()


# (LEVEL 1)
# TASK 1 - TOP 3 CUISINES
Cuisines = Data['Cuisines'].dropna().str.split(', ')
flattened_cuisines = [Cuisines for sublist in Cuisines for Cuisines in sublist]
Cuisines_counts = pd.Series(flattened_cuisines).value_counts()
top_3_cuisines = Cuisines_counts.head(3)
print("Top 3 most common cuisines:")
print(top_3_cuisines)

# PERCENTAGE OF RESTAURANTS THAT SERVE EACH OF THE TOP CUISINES
Total_restaurants = len(Data)
Cuisines_percentages = (Cuisines_counts / Total_restaurants) * 100
Top_Cuisines_Percentage = Cuisines_percentages.head(3)
print("Top cuisines with percentages:")
print(Top_Cuisines_Percentage)

# TASK 2 - IDENTIFY THE CITY WITH THE HIGHEST NUMBER OF RESTAURANTS IN THE DATASET
City_Counts = Data['City'].value_counts()
top_city = City_Counts.idxmax()
top_city_count = City_Counts.max()
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
Data['Votes'] = pd.to_numeric(Data['Votes'], errors='coerce')
Average_rating_by_city = Data.groupby('City')['Votes'].mean()
print("Average rating for restaurants in each city:")
print(Average_rating_by_city)

# DETERMINE THE CITY WITH THE HIGHEST AVERAGE RATING
top_city = Average_rating_by_city.idxmax()
top_average_rating = Average_rating_by_city.max()
print(f"The city with the highest average rating is {top_city} with an average of {top_average_rating:.1f}.")

# TASK 3 - (PRICE RANGE DISTRIBUTION)
#BAR CHART TO VISUALISE THE DISTRIBUTION OF THE PRICE RANGE AMONG THE RESTAURANTS

price_range_counts = Data['Price range'].value_counts()
plt.figure(figsize=(8, 6))
price_range_counts.sort_index().plot(kind='bar', color='skyblue', edgecolor='black')
plt.title('Distribution of Price Ranges Among Restaurants', fontsize=14)
plt.xlabel('Price Range', fontsize=12)
plt.ylabel('Number of Restaurants', fontsize=12)

plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()


# CALCULATE THE PERCENTAGE OF RESTAURANTS IN EACH PRICE RANGE CATEGORY

price_range_counts = Data["Price range"].value_counts()
total_restaurants = len(Data)
price_range_percentages = (price_range_counts / total_restaurants) * 100
print("Percentage of restaurants in each price range:")
```

```
print(price_range_percentages)


# TASK 4 - (ONLINE DELIVERY)
# DETERMINE THE PERCENTAGE OF RESTAURANTS THAT OFFER ONLINE DELIVERY

Data['Has Online delivery'] = Data['Has Online delivery'].str.strip().str.lower()
total_restaurants = len(Data)
online_delivery_count = Data['Has Online delivery'].value_counts().get('yes', 0)
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

with_online_delivery = Data[Data['Has Online delivery'] == 'yes']
without_online_delivery = Data[Data['Has Online delivery'] == 'no']
avg_rating_with_delivery = with_online_delivery['Aggregate rating'].mean()
avg_rating_without_delivery = without_online_delivery['Aggregate rating'].mean()
print(f"Average rating of restaurants with online delivery: {avg_rating_with_delivery:.2f}")
print(f"Average rating of restaurants without online delivery: {avg_rating_without_delivery:.2f}")




# # (LEVEL 2)
# # TASK 1 - RESTAURANT RATINGS

# DISTRIBUTION OF THE AGGREGATE RATING & DETERMINE THE MOST COMMON RATING RANGE
Data['Aggregate rating'] = pd.to_numeric(Data['Aggregate rating'], errors='coerce')
bins = [0, 1, 2, 3, 4, 5]
labels = ['0-1', '1-2', '2-3', '3-4', '4-5']
Data['Rating Range'] = pd.cut(Data['Aggregate rating'], bins=bins, labels=labels, include_lowest=True)
rating_distribution = Data['Rating Range'].value_counts().sort_index()
most_common_range = rating_distribution.idxmax()
most_common_count = rating_distribution.loc[most_common_range]
print("Rating Distribution:")
print(rating_distribution)
print(f"\nThe most common rating range is {most_common_range} with {most_common_count} restaurants.")
plt.figure(figsize=(8, 6))
rating_distribution.plot(kind='bar', color='skyblue', edgecolor='black')
plt.title('Distribution of Aggregate Ratings', fontsize=14)
plt.xlabel('Rating Range', fontsize=12)
plt.ylabel('Number of Restaurants', fontsize=12)
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

# AVERAGE NUMBER OF VOTES RECEIVED BY THE RESTAURANTS
Data['Votes'] = pd.to_numeric(Data['Votes'], errors='coerce')
average_votes = Data['Votes'].mean()
print(f"The average number of votes received by restaurants is {average_votes}.")


# TASK 2 - RESTAURANT RATINGS (CUISINE COMBINATION)

# MOST COMMON COMBINATIONS OF CUISINES
cuisine_combination = Data['Cuisines'].value_counts()
print("Most common cuisine combination")
print(cuisine_combination.head(10))


# DETERMINE IF CERTAIN CUISINE COMBINATIONS TEND TO HAVE HIGHER RATINGS

cuisine_rating = Data.groupby('Cuisines')['Aggregate rating'].mean()
sorted_cuisine_rating = cuisine_rating.sort_values(ascending=False)
print("Cuisine combination with the highest average ratings:")
print(sorted_cuisine_rating.head(10))


# TASK 3 - GEOGRAPHIC COMBINATION
```

```
# LOCATIONS OF RESTAURANTS ON THE MAP USING LONGITUDE & LATITUDE CO-ORDINATES

Data['Latitude'] = pd.to_numeric(Data['Latitude'], errors='coerce')
Data['Longitude'] = pd.to_numeric(Data['Longitude'], errors='coerce')

map_center = [Data['Latitude'].mean(), Data['Longitude'].mean()]
restaurant_map = folium.Map(location=map_center, zoom_start=12)
for _, row in Data.iterrows():
    if not pd.isna(row['Latitude']) and not pd.isna(row['Longitude']):
        folium.Marker(
            location=[row['Latitude'], row['Longitude']],
            popup=f"Restaurant: {row['Restaurant ID']}<br>Cuisine: {row['Cuisines']}",
            icon=folium.Icon(color="blue", icon="info-sign")
        ).add_to(restaurant_map)
restaurant_map.save("restaurant_locations.html")
print("Map has been created and saved as 'restaurant_locations.html'.")


# PATTERNS OR CLUSTERS OF RESTAURANTS IN SPECIFIC AREAS

coordinates = Data[['Latitude', 'Longitude']]
coordinates = coordinates.dropna()
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(coordinates)
Data['Cluster'] = kmeans.labels_

plt.figure(figsize=(10, 8))
for cluster in range(5):  # Adjust this number to match the number of clusters
    cluster_data = Data[Data['Cluster'] == cluster]
    plt.scatter(cluster_data['Longitude'], cluster_data['Latitude'], label=f'Cluster {cluster}', s=10)

plt.scatter(kmeans.cluster_centers_[:, 1], kmeans.cluster_centers_[:, 0], c='red', marker='x', s=100, label='Centers')

plt.title('Clusters of Restaurants', fontsize=16)
plt.xlabel('Longitude', fontsize=12)
plt.ylabel('Latitude', fontsize=12)
plt.legend()
plt.grid()
plt.show()
for cluster in range(5):
    cluster_data = Data[Data['Cluster'] == cluster]
    print(f"Cluster {cluster}: {len(cluster_data)} restaurants")



# TASK 4 - RESTAURANTS CHAINS

# IDENTIFY IF THERE ARE ANY RESTAURANTS CHAINS PRESENT IN THE DATASET

restaurant_counts = Data['Restaurant Name'].value_counts()
restaurant_chains = restaurant_counts[restaurant_counts > 1]
print(f"Number of restaurant chains:{len(restaurant_counts > 1)}")
print(restaurant_chains)

# RATINGS & POPULARITY OF DIFFERENT RESTAURANT CHAINS

chain_analysis = Data.groupby('Restaurant Name').agg(
    Average_Rating=('Aggregate rating', 'mean'),
    Total_Votes=('Votes', 'sum'),
    Location_Count=('Restaurant Name', 'count')
).reset_index()

restaurant_chains = chain_analysis[chain_analysis['Location_Count'] > 1]
print("Sample Data from restaurant_chains:")
print(restaurant_chains.head())
popular_chains = restaurant_chains.sort_values(by='Total_Votes', ascending=False)
top_rated_chains = restaurant_chains.sort_values(by='Average_Rating',ascending=False)
print("Top Restaurant Chains by Popularity:")
print(popular_chains.head())
print(top_rated_chains.head())


plt.figure(figsize=(10, 6))
plt.scatter(restaurant_chains['Average_Rating'], restaurant_chains['Total_Votes'], color='orange', edgecolor='black')
plt.xlabel('Average Rating')
```

```
plt.ylabel('Total Votes')
plt.title('Ratings vs Popularity for Restaurant Chains')
plt.grid(True)
plt.show()
```