**Terminal Inputs and outputs  
1)**

# Import required libraries

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# File paths

restaurants\_path = '/Users/mohama56/BANA 5440 Python/Project 2/restaurants.csv'

reviews\_path = '/Users/mohama56/BANA 5440 Python/Project 2/reviews.csv'

# Load datasets

restaurants = pd.read\_csv(restaurants\_path)

reviews = pd.read\_csv(reviews\_path)

# Get all Subway business IDs

subway = restaurants[restaurants['name'].str.contains('Subway', case=False, na=False)]

subway\_ids = subway['business\_id'].tolist()

# Filter reviews for Subway

subway\_reviews = reviews[reviews['business\_id'].isin(subway\_ids)]

subway\_mean = subway\_reviews['stars'].mean()

subway\_std = subway\_reviews['stars'].std()

# Print Subway stats

print(f"Total Subway Reviews: {len(subway\_reviews)}")

print(f"Updated Subway Mean Rating: {subway\_mean:.2f}")

print(f"Updated Subway Standard Deviation: {subway\_std:.2f}")

# Find competitors based on categories containing "Sandwiches"

competitors = restaurants[restaurants['categories'].str.contains('Sandwiches', case=False, na=False)]

competitor\_ids = competitors['business\_id'].iloc[:2].tolist()

# Filter reviews for competitors

competitor\_reviews = reviews[reviews['business\_id'].isin(competitor\_ids)]

competitor\_stats = competitor\_reviews.groupby('business\_id')['stars'].agg(['mean', 'std']).reset\_index()

competitor\_stats = pd.merge(competitor\_stats, restaurants[['business\_id', 'name']], on='business\_id')

# Print competitor stats

print("\nCompetitor Ratings:")

print(competitor\_stats)

# Prepare data for visualization

x\_labels = ['Subway'] + competitor\_stats['name'].tolist()

mean\_values = [subway\_mean] + competitor\_stats['mean'].tolist()

std\_values = [subway\_std] + competitor\_stats['std'].tolist()

# Visualize comparison

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

sns.barplot(x=x\_labels, y=mean\_values, capsize=0.1)

# Add error bars manually

for idx, (mean, std) in enumerate(zip(mean\_values, std\_values)):

plt.errorbar(x=idx, y=mean, yerr=std, fmt='o', color='black', capsize=5)

plt.title('Mean Ratings: Subway vs Competitors')

plt.xlabel('Business')

plt.ylabel('Mean Rating')

plt.ylim(0, 5)

plt.xticks(rotation=15)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()

**Output**

**A computer screen with white text

Description automatically generated**

**A graph of a bar chart

Description automatically generated**

**2)**

# Import required libraries

import pandas as pd

import matplotlib.pyplot as plt

# File paths

restaurants\_path = '/Users/mohama56/BANA 5440 Python/Project 2/restaurants.csv'

reviews\_path = '/Users/mohama56/BANA 5440 Python/Project 2/reviews.csv'

# Load datasets

restaurants = pd.read\_csv(restaurants\_path)

reviews = pd.read\_csv(reviews\_path)

# Analyze ratings for sandwich restaurants

sandwich\_restaurants = restaurants[restaurants['categories'].str.contains('Sandwiches', case=False, na=False)]

sandwich\_ids = sandwich\_restaurants['business\_id'].tolist()

sandwich\_reviews = reviews[reviews['business\_id'].isin(sandwich\_ids)]

# Calculate average and standard deviation of ratings for sandwich restaurants

sandwich\_mean = sandwich\_reviews['stars'].mean()

sandwich\_std = sandwich\_reviews['stars'].std()

print(f"Sandwich Restaurants - Mean Rating: {sandwich\_mean:.2f}, Standard Deviation: {sandwich\_std:.2f}")

# Analyze ratings for non-sandwich restaurants

non\_sandwich\_restaurants = restaurants[~restaurants['categories'].str.contains('Sandwiches', case=False, na=False)]

non\_sandwich\_ids = non\_sandwich\_restaurants['business\_id'].tolist()

non\_sandwich\_reviews = reviews[reviews['business\_id'].isin(non\_sandwich\_ids)]

# Calculate average and standard deviation of ratings for non-sandwich restaurants

non\_sandwich\_mean = non\_sandwich\_reviews['stars'].mean()

non\_sandwich\_std = non\_sandwich\_reviews['stars'].std()

print(f"Non-Sandwich Restaurants - Mean Rating: {non\_sandwich\_mean:.2f}, Standard Deviation: {non\_sandwich\_std:.2f}")

# Visualize comparison

categories = ['Sandwich Restaurants', 'Non-Sandwich Restaurants']

means = [sandwich\_mean, non\_sandwich\_mean]

stds = [sandwich\_std, non\_sandwich\_std]

plt.figure(figsize=(8, 6))

plt.bar(categories, means, yerr=stds, capsize=10, color=['orange', 'blue'])

plt.title('Comparison of Ratings: Sandwich vs Non-Sandwich Restaurants')

plt.ylabel('Mean Rating')

plt.ylim(0, 5)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()

**Output**

**A screen shot of a computer

Description automatically generated**

**A chart of different types of restaurants

Description automatically generated with medium confidence**

**3)**

# Import required libraries

import pandas as pd

import matplotlib.pyplot as plt

# File paths

restaurants\_path = '/Users/mohama56/BANA 5440 Python/Project 2/restaurants.csv'

reviews\_path = '/Users/mohama56/BANA 5440 Python/Project 2/reviews.csv'

# Load datasets

restaurants = pd.read\_csv(restaurants\_path)

reviews = pd.read\_csv(reviews\_path)

# Count the number of unique cities for each business

city\_counts = restaurants.groupby('business\_id')['city'].nunique().reset\_index()

city\_counts.columns = ['business\_id', 'city\_count']

# Merge city counts back with the restaurants dataset

restaurants = pd.merge(restaurants, city\_counts, on='business\_id')

# Categorize restaurants as 'National' if present in >50 cities, 'Local' if present in 1 city

restaurants['chain\_type'] = restaurants['city\_count'].apply(

lambda x: 'National' if x > 1 else 'Local'

)

# Filter reviews for national and local chains

national\_ids = restaurants[restaurants['chain\_type'] == 'National']['business\_id']

local\_ids = restaurants[restaurants['chain\_type'] == 'Local']['business\_id']

national\_reviews = reviews[reviews['business\_id'].isin(national\_ids)]

local\_reviews = reviews[reviews['business\_id'].isin(local\_ids)]

# Calculate mean and standard deviation for ratings

national\_mean = national\_reviews['stars'].mean()

national\_std = national\_reviews['stars'].std()

local\_mean = local\_reviews['stars'].mean()

local\_std = local\_reviews['stars'].std()

# Print results

print(f"National Chains - Average Rating: {national\_mean:.2f}, Standard Deviation: {national\_std:.2f}")

print(f"Local Chains - Average Rating: {local\_mean:.2f}, Standard Deviation: {local\_std:.2f}")

# Check city count distribution

city\_count\_distribution = restaurants['city\_count'].value\_counts()

print("City Count Distribution:")

print(city\_count\_distribution)

# Count national and local chains

national\_count = len(restaurants[restaurants['chain\_type'] == 'National'])

local\_count = len(restaurants[restaurants['chain\_type'] == 'Local'])

print(f"\nNumber of National Chains: {national\_count}")

print(f"Number of Local Chains: {local\_count}")

# Check reviews for National Chains

print("\nSample Reviews for National Chains:")

print(national\_reviews.head())

# Check if National Chains exist in the reviews dataset

print(f"\nNumber of reviews for National Chains: {len(national\_reviews)}")

print(f"Number of reviews for Local Chains: {len(local\_reviews)}")

# Visualize comparison

categories = ['National Chains', 'Local Chains']

means = [national\_mean, local\_mean]

stds = [national\_std, local\_std]

plt.figure(figsize=(8, 6))

plt.bar(categories, means, yerr=stds, capsize=10, color=['blue', 'orange'], alpha=0.7)

plt.title('Average Ratings: National vs Local Chains')

plt.ylabel('Average Rating')

plt.ylim(0, 5)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()

**Output**

**A computer screen with white text

Description automatically generated**

**A chart with a black line

Description automatically generated**

**4)**

**Part A**

# Import required libraries

import pandas as pd

import matplotlib.pyplot as plt

# File paths

reviews\_path = '/Users/mohama56/BANA 5440 Python/Project 2/reviews.csv'

# Load dataset

reviews = pd.read\_csv(reviews\_path)

# Convert 'date' column to datetime format

reviews['date'] = pd.to\_datetime(reviews['date'], errors='coerce') # Handle potential parsing errors

# Drop rows with invalid dates

reviews = reviews.dropna(subset=['date'])

# Extract the year from the 'date' column

reviews['year'] = reviews['date'].dt.year

# Part A: Distribution of ratings

ratings\_distribution = reviews['stars'].value\_counts().sort\_index()

print("Overall Ratings Distribution:")

print(ratings\_distribution)

# Plot overall distribution of ratings

plt.figure(figsize=(8, 6))

ratings\_distribution.plot(kind='bar', color='skyblue')

plt.title('Overall Ratings Distribution')

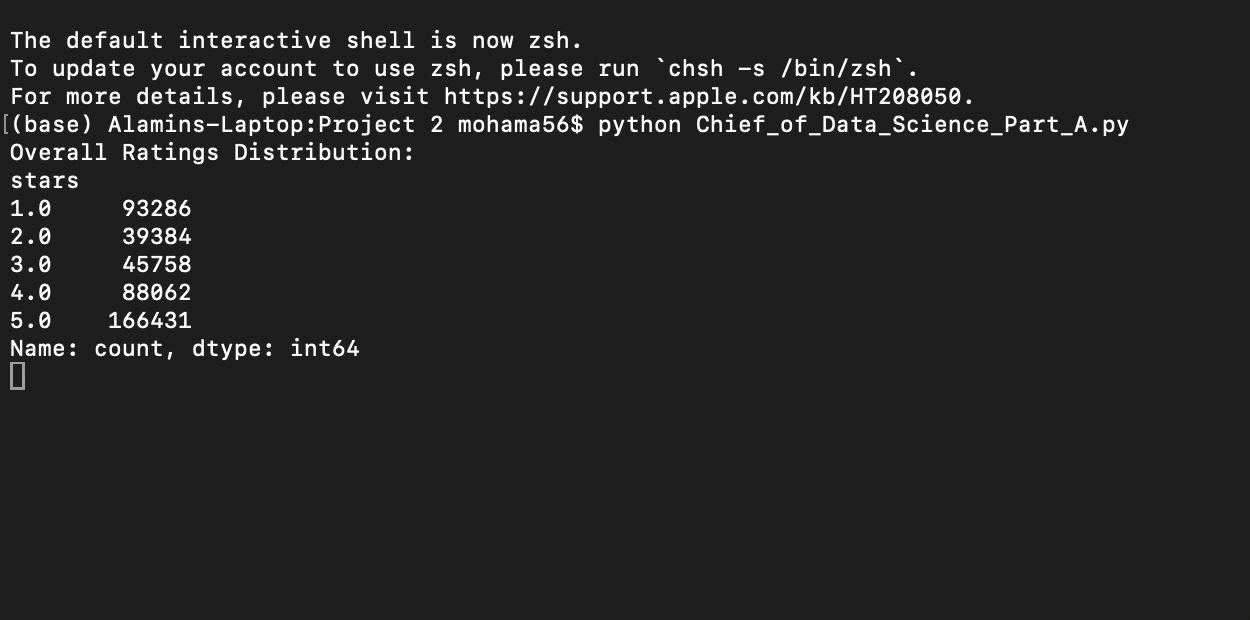
plt.xlabel('Ratings')

plt.ylabel('Number of Reviews')

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()

**Output**

****

**A graph of blue bars

Description automatically generated with medium confidence**

**Part B**

# Import required libraries

import pandas as pd

import matplotlib.pyplot as plt

# File path

reviews\_path = '/Users/mohama56/BANA 5440 Python/Project 2/reviews.csv'

# Load dataset

reviews = pd.read\_csv(reviews\_path)

# Convert 'date' column to datetime format and handle errors

reviews['date'] = pd.to\_datetime(reviews['date'], errors='coerce')

# Drop rows with invalid dates

reviews = reviews.dropna(subset=['date'])

# Extract the year from the 'date' column

reviews['year'] = reviews['date'].dt.year

# Filter data for the years 2018 to 2021

years\_to\_analyze = [2018, 2019, 2020, 2021]

filtered\_reviews = reviews[reviews['year'].isin(years\_to\_analyze)]

# Debugging: Check filtered data

print("\nFiltered Reviews (First 5 Rows):")

print(filtered\_reviews[['year', 'stars']].head())

# Group by year and rating to calculate the distribution

yearly\_distributions = filtered\_reviews.groupby(['year', 'stars']).size().unstack(fill\_value=0)

# Debugging: Ensure yearly distribution table is correct

if yearly\_distributions.empty:

print("\nNo data available for the specified years (2018–2021).")

else:

print("\nYearly Ratings Distribution (2018–2021):")

print(yearly\_distributions)

# Plot yearly distribution of ratings

yearly\_distributions.plot(kind='bar', figsize=(12, 8), stacked=True, colormap='viridis', width=0.8)

plt.title('Yearly Ratings Distribution (2018–2021)')

plt.xlabel('Year')

plt.ylabel('Number of Reviews')

plt.legend(title='Ratings', loc='upper left', bbox\_to\_anchor=(1.0, 1.0))

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight\_layout()

plt.show()

**Output**

**A computer screen shot of a computer program

Description automatically generated**

**A chart with different colored squares

Description automatically generated**

**Bonus Question-3**

**Visuals**

# Import required libraries

import pandas as pd

import matplotlib.pyplot as plt

# File paths

reviews\_path = '/Users/mohama56/BANA 5440 Python/Project 2/reviews.csv'

restaurants\_path = '/Users/mohama56/BANA 5440 Python/Project 2/restaurants.csv'

# Load datasets

reviews = pd.read\_csv(reviews\_path)

restaurants = pd.read\_csv(restaurants\_path)

# Convert 'date' column to datetime format

reviews['date'] = pd.to\_datetime(reviews['date'])

# Extract the year and month from the 'date' column

reviews['year'] = reviews['date'].dt.year

reviews['month'] = reviews['date'].dt.month

# Visual 1: Yearly Ratings Distribution

print("\nGenerating Visual 1: Yearly Ratings Distribution...")

yearly\_distribution = reviews.groupby(['year', 'stars']).size().unstack(fill\_value=0)

print("\nYearly Ratings Distribution (2018–2021):")

print(yearly\_distribution.loc[2018:2021]) # Only print years 2018–2021

# Plot yearly distribution of ratings

yearly\_distribution.plot(kind='bar', figsize=(12, 8), stacked=True, colormap='viridis', width=0.8)

plt.title('Yearly Ratings Distribution (2018–2021)')

plt.xlabel('Year')

plt.ylabel('Number of Reviews')

plt.legend(title='Ratings', loc='upper left', bbox\_to\_anchor=(1.0, 1.0))

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight\_layout()

plt.show()

# Visual 2: Bottom 10 Subway Locations by Average Rating

print("\nGenerating Visual 2: Bottom 10 Subway Locations by Average Rating...")

location\_ratings = reviews.groupby('business\_id')['stars'].mean().reset\_index()

location\_ratings = pd.merge(location\_ratings, restaurants[['business\_id', 'name', 'city', 'state']], on='business\_id', how='left')

bottom\_10\_locations = location\_ratings.sort\_values(by='stars', ascending=True).head(10)

print("\nBottom 10 Subway Locations by Average Rating:")

print(bottom\_10\_locations[['name', 'city', 'state', 'stars']])

# Plot bottom 10 Subway locations by average rating

bottom\_10\_locations.plot(

x='name', y='stars', kind='barh', figsize=(10, 6), color='orange'

)

plt.title('Bottom 10 Subway Locations by Average Rating')

plt.xlabel('Average Rating')

plt.ylabel('Subway Location')

plt.grid(axis='x', linestyle='--', alpha=0.7)

plt.tight\_layout()

plt.show()

# Visual 3: Monthly Trend of Average Ratings

print("\nGenerating Visual 3: Monthly Trend of Average Ratings...")

monthly\_ratings = reviews.groupby('month')['stars'].mean().reset\_index()

print("\nMonthly Average Ratings:")

print(monthly\_ratings)

# Plot monthly trend of ratings

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

plt.plot(

monthly\_ratings['month'], monthly\_ratings['stars'],

marker='o', linestyle='-', color='blue'

)

plt.title('Monthly Trend of Average Ratings')

plt.xlabel('Month')

plt.ylabel('Average Rating')

plt.xticks(range(1, 13))

plt.grid(axis='both', linestyle='--', alpha=0.7)

plt.tight\_layout()

plt.show()