CS105 Fall 2025 Team 7 Mini Project Report

Author: Lydia Niu, Ryan Kim, Jonathan Chun, Stuart Arief, Colin Pham

Topic Presentation

- Stuart Arief: Determining a relationship between an individual's mental health rating and commuting distance to campus.

- Ryan Kim: Finding relationship between caffeine intake and tendency to eat meals from campus restaurants/dining.

- Colin Pham: Determining relationship between tendency to skip breakfast and on-campus eating. Finding relationship between hours of sleep per night, caffeine consumption, and tendency to skip breakfast amongst data science versus computer science students.

- Jonathan Chun: Determining relationship between an individual's ethnicity and preference for certain cuisines.

- Lydia Niu: I'm exploring how students' favorite cuisines relate to how often they eat at on-campus restaurants—specifically, whether limited cuisine variety discourages on-campus dining. I'm also interested in how imposter syndrome connects to mental health and caffeine intake, as I wonder if higher caffeine consumption might signal higher imposter syndrome levels.

Data

- Mental Health rating (1-5) (numerical data)

- Commute distance to campus in miles (numerical data)

- Average Caffeine consumption, mg per day (continuous numerical)

- Campus food meal consumption rate, times per week (discrete numerical)

- Favorite cuisine rating in a scale 1-5 (discrete numerical rating)

- Imposter syndrome rating 1-5 (discrete numerical rating)

- Ethnicity survey (Categorical data)

Introduction:

For our mini project, we found that food was something worth incorporating into our research. It is a necessity among everyone, especially for college students. However, despite the provision of dining options, some students never eat from campus at all, regardless of how convenient it may seem. And in other cases, students eat on campus very frequently, even though alternative options seem better. Regardless, our group decided to explore the potential factors influencing UCR students' food preferences throughout the weekday as our initial topic of interest.

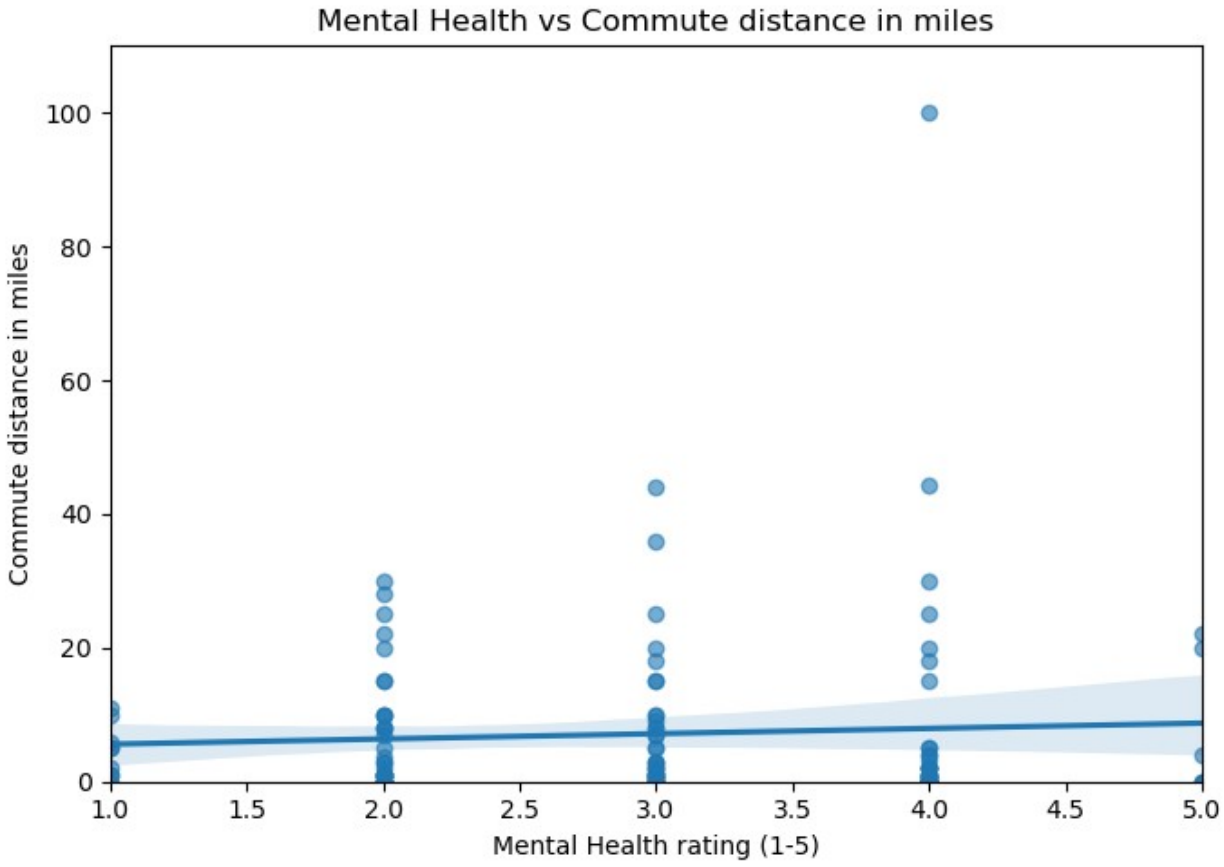
```
# import all the libraries and read the data
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import re
from scipy.stats import chi2_contingency, chi2
from IPython.display import display, Markdown, HTML

df = pd.read_csv("responses.csv")
```

EDA 1 - Mental Health Level vs Commuting Distance (Scatterplot, Stuart)

```
responses_df = pd.read_csv("responses.csv")

# ---- Normalize headers (remove newlines / smart quotes) ----
responses_df.columns = [
    re.sub(r"\s+", " ", c.replace("“", '"').replace("”",
'"').replace("’", "'")).strip()
    for c in responses_df.columns
]

# ---- Detect columns ----
# mental health column (e.g., "Rate your overall mental health.")
mental_col = [c for c in responses_df.columns if ("overall" in
c.lower() and "mental" in c.lower())]
if not mental_col:
    raise KeyError("Could not find the 'overall mental health'
question header.")
mental_col = mental_col[0]

# Commute distance column (allowing for slight variations of
'following')
commute_col = [
    c for c in responses_df.columns
    if ("far" in c.lower() and "commute" in c.lower() and "in miles"
in c.lower())
]
if not commute_col:
    raise KeyError("Could not find the 'Commute distance (…)' activity
frequency header.")
commute_col = commute_col[0]

# commute_col = [
#     c for c in responses_df.columns
#     if ("how" in c.lower() and "units" in c.lower() and "this
quarter" in c.lower())
# ]
# if not commute_col:
#     raise KeyError("Could not find the 'Commute distance (…)'
activity frequency header.")
# commute_col = commute_col[0]

# Commute distance column (allowing for slight variations of
'following')

print("Detected columns:\n- Mental health:", mental_col, "\n- Commute
distance:", commute_col)
```

```
# ---- Clean values to numeric ----
def extract_full_number(x):
    """Extract the first full number (integer or float) from mixed
responses; else NaN."""
    if pd.isna(x):
        return np.nan

    s = str(x).strip()
    # Regex to find the first floating-point number pattern in the
string
    match = re.search(r'^-?\d*\.?\d+(?:[eE][+-]?\d+)?', s)

    if match:
        number_str = match.group(0)
        try:
            return float(number_str)
        except ValueError:
            return np.nan
    else:
        return np.nan

responses_df["mental_health"] =
responses_df[mental_col].apply(extract_full_number)
responses_df["commute_distance"] =
responses_df[commute_col].apply(extract_full_number)

# ---- Drop rows with missing values for this plot ----
plot_df = responses_df.dropna(subset=["mental_health",
"commute_distance"])

# ---- Create scatter + regression line ----
plt.figure(figsize=(7, 5))
sns.regplot(
    data=plot_df,
    x="mental_health",
    y="commute_distance",
    scatter_kws={"alpha": 0.6}
)

# ---- Titles and labels ----
plt.title("Mental Health vs Commute distance in miles")
plt.xlabel("Mental Health rating (1-5)")
plt.ylabel("Commute distance in miles")

# ---- Nice bounds (Like on x is usually 1–5) ----
plt.xlim(1,5)
plt.ylim(bottom = 0, top = 110)

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

```
Detected columns:
- Mental health: Rate your overall mental health.
- Commute distance: How far is your commute to UCR (in miles)? Answer with a number only.
```



Introduction:

One of our goals is to find the factors affecting an individual's mental health. In this case, we are investigating commute distance and how that affects one's mental wellbeing. We expect there to be a negative correlation, where the farther the distance an individual has to travel to get on campus, the worse their mental health becomes.

Analysis:

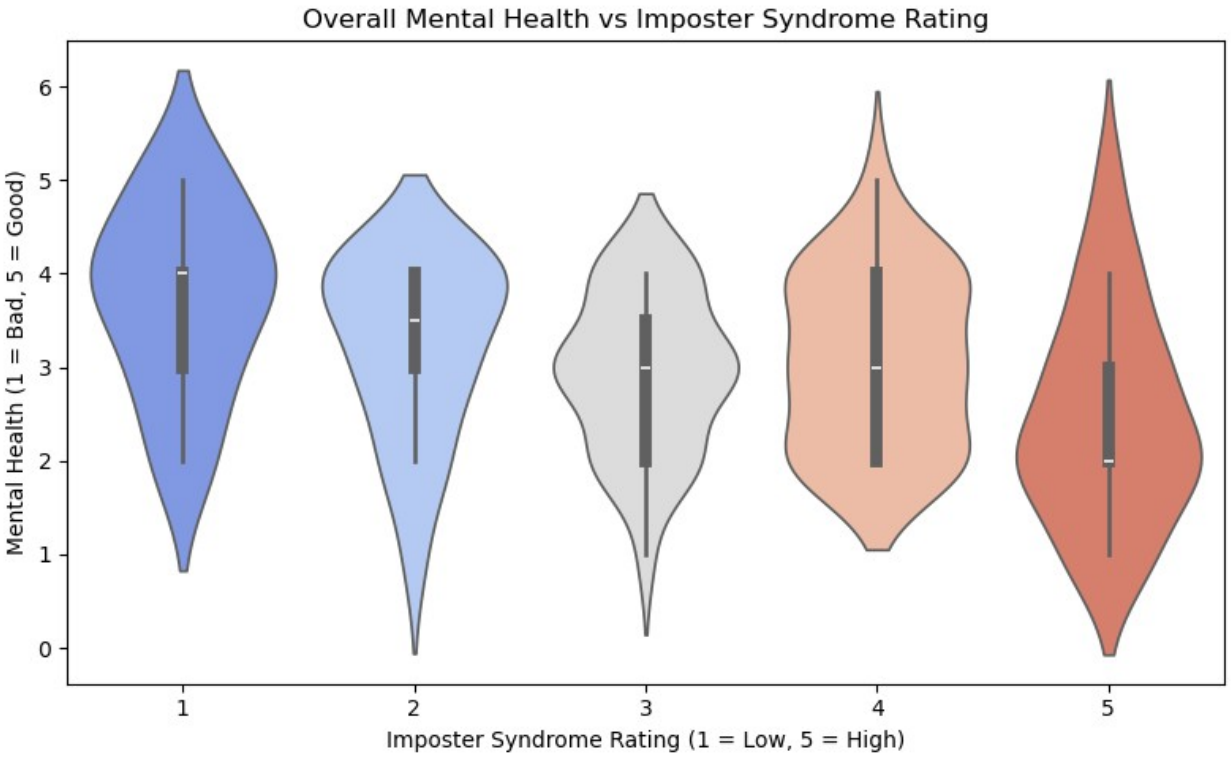
The scatterplot above maps two variables: mental health rating from 1-5 on the x-axis, and commute distance in miles on the y-axis. Each blue circle represents an individual and their choices during the survey phase. A scatterplot was chosen for this visualization because of how ideal it is for this scenario. We are dealing with two variables and are trying to see if there is a relationship between the two. The points show a pattern similar to the normal distribution bell curve, hence, we have a relatively straight regression line. This indicates that there is little to no relation between the aforementioned variables, thereby disproving the hypothesis.

EDA2 - Mental Health vs Imposter Syndrome Rating (Violin, Lydia)

```
is_col = 'The National Library of Medicine defines imposter syndrome
as, " a behavioral health phenomenon described as self-doubt of
intellect, skills, or accomplishments among high-achieving
individuals" ("Imposter Phenomenon", 2023). Based on this definition
please rate your level of imposter syndrome through your journey in
your major. '
seat_col = 'From the front of the class (ranked 1) to the back of the
class (ranked 5), where do you sit on average for lecture?'
mental_health_col = 'Rate your overall mental health.'
caff_col = 'How many milligrams of caffeine do you consume per day, on
average? For example, a cup of coffee is 95 mg, a shot of espresso is
64mg, a can of Celsius is 200 mg, and a  can of Coca Cola is 34 mg.
Answer with a number only.'
is_order = ['1', '2', '3', '4', '5']

# Select relevant columns and drop missing values
df_plot = df[[is_col, mental_health_col]].dropna()
```

```
# Convert IS to integer (if stored as float)
df_plot[is_col] = df_plot[is_col].astype(int)

# Violin plot — one violin per IS rating (1–5)
plt.figure(figsize=(8, 5))
sns.violinplot(
    data=df_plot,
    x=is_col,
    y=mental_health_col,
    palette="coolwarm",
    inner="box"
)
plt.title("Overall Mental Health vs Imposter Syndrome Rating")
plt.xlabel("Imposter Syndrome Rating (1 = Low, 5 = High)")
plt.ylabel("Mental Health (1 = Bad, 5 = Good)") # need the scale info, 1=bad or good?
plt.tight_layout()
plt.show()
```

```
/var/folders/_2/3lrnsn7x5615sf09xw7cfcfm0000gn/T/ipykernel_82385/336458482.py:15: FutureWarning:

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

  sns.violinplot(
```


Overall Mental Health vs Imposter Syndrome Rating
Mental Health (1 = Bad, 5 = Good)
0
1
2
3
4
5
6
1
2
3
4
5
Imposter Syndrome Rating (1 = Low, 5 = High)

Introduction:

This violin chart uses "students' self-rated mental health (1 = bad, 5 = good)" with their "self-reported imposter syndrome levels (1 = low, 5 = high)." The goal is to observe whether students who experience stronger feelings of imposter syndrome also report lower overall mental health. Understanding this relationship can help highlight potential psychological patterns among students and guide future campus mental wellness initiatives.

Analysis:

The violin plot shows a clear downward trend, as the white median indicators within each box decrease as the imposter syndrome rating increases. This indicates that overall mental health declines as imposter syndrome increases, which matches the assumptions. Students with higher imposter syndrome ratings (4–5) generally report lower mental health scores, with their responses clustering around 2–3. In contrast, those with lower imposter syndrome levels (1–2) tend to rate their mental health higher, with median values around 3–4. The widest portions of the violins for IS levels 1 and 2 occur at a mental health rating of 4, highlighting stronger well-being among these groups. This inverse relationship visually reinforces the idea that higher imposter syndrome is associated with poorer mental health.

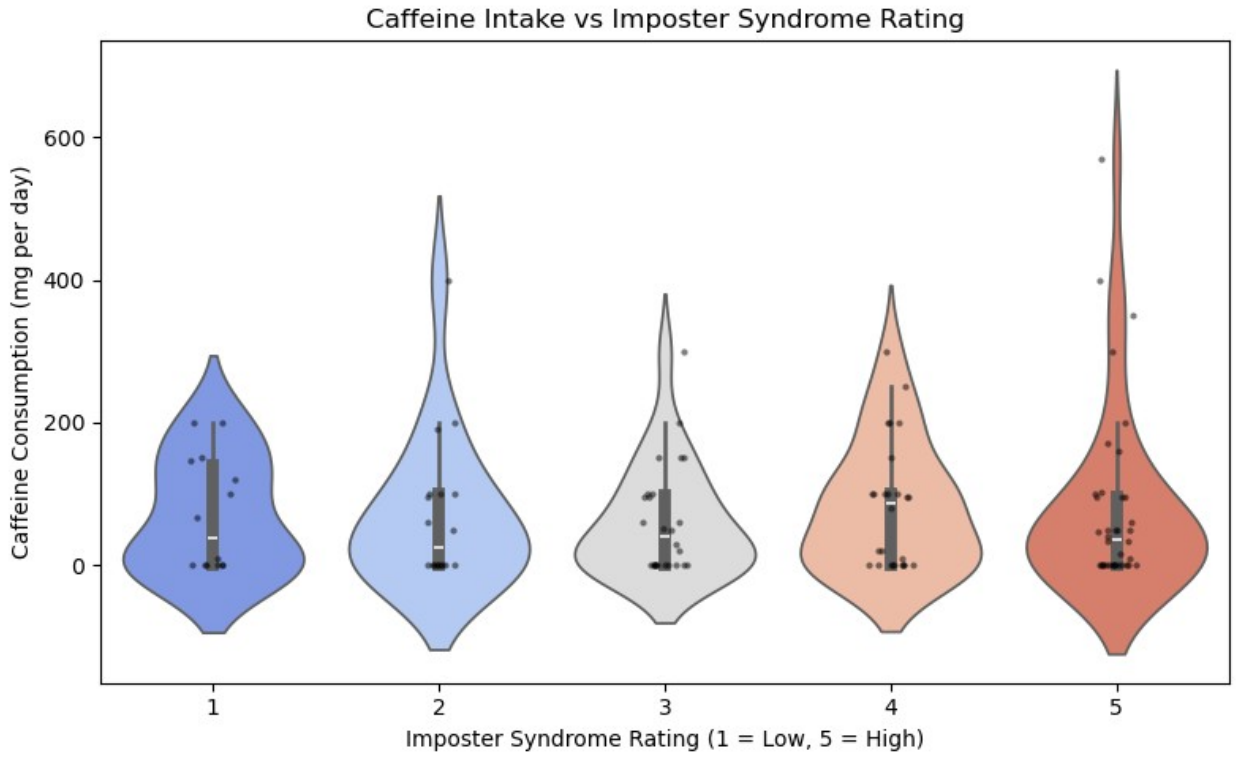
EDA3 - Caffeine Consumption vs Imposter Syndrome Rating (Violin, Lydia)

```
caff_col_clean = 'caffeine_mg'  # new numeric column

# Clean the caffeine data
# Copy column and strip spaces
caff = df[caff_col].astype(str).str.strip().copy()


# Cleaning function
def clean_caffeine(x):
    if pd.isna(x) or x.lower() in ['n/a', 'na']:
        return np.nan
```

```
    # Case 1: contains 'mg'
    if 'mg' in x.lower():
        try:
            return float(x.lower().replace('mg','').strip())
        except:
            return np.nan
    # Case 2: range like "95-300"
    if '-' in x:
        try:
            parts = x.split('-')
            nums = [float(p.strip()) for p in parts]
            return np.mean(nums)  # take the average of the range
        except:
            return np.nan
    # Case 3: just a number
    try:
        return float(x)
    except:
        return np.nan

# Apply cleaning
df[caff_col_clean] = caff.apply(clean_caffeine)

# Optional: inspect cleaned data
# print(df[caff_col_clean].describe())
# print(np.sort(df[caff_col_clean].dropna().unique()))

# ------------------------------
# Prepare data for plotting
# ------------------------------
df_plot = df[[is_col, caff_col_clean]].dropna().copy()
df_plot[is_col] = df_plot[is_col].astype(int).astype(str)  # treat IS
rating as categorical

# ------------------------------
# Create violin plot
# ------------------------------
plt.figure(figsize=(8, 5))
sns.violinplot(
    data=df_plot,
    x=is_col,
    y=caff_col_clean,
    palette="coolwarm",
    inner="box",
    order=is_order
)

# Optional: add individual participant points
sns.stripplot(
    data=df_plot,
```

```
    x=is_col,
    y=caff_col_clean,
    color='k',
    alpha=0.5,
    jitter=True,
    size=3
)

# Labels and title
plt.title("Caffeine Intake vs Imposter Syndrome Rating")
plt.xlabel("Imposter Syndrome Rating (1 = Low, 5 = High)")
plt.ylabel("Caffeine Consumption (mg per day)")
plt.tight_layout()
plt.show()
```

```
/var/folders/_2/3lrnsn7x5615sf09xw7cfcfm0000gn/T/
ipykernel_82385/1074205603.py:49: FutureWarning:

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

  sns.violinplot(
```



Introduction:

This violin chart compares "students' daily average caffeine intake (in milligrams)" with their "self-rated imposter syndrome levels (1 = low, 5 = high)." The goal is to explore whether students experiencing stronger imposter feelings tend to consume more caffeine, possibly as a coping mechanism for stress or fatigue. Understanding this relationship can provide insight into behavioral patterns linked to academic pressure and psychological well-being.

Analysis:

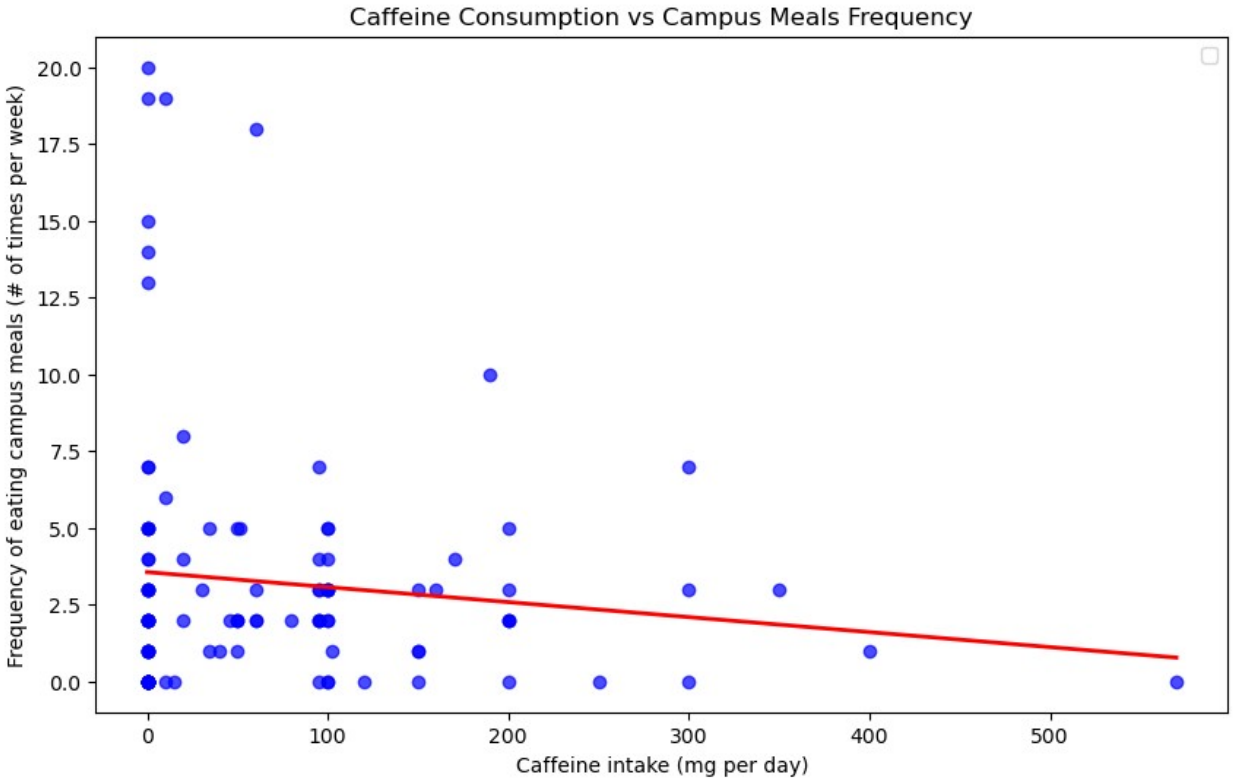
The violin plot shows that caffeine intake varies widely across all imposter syndrome levels, with several outliers consuming notably high amounts of caffeine. There is no clear linear trend between imposter syndrome and caffeine consumption, and the overall average intake generally falls within the range of 0–200 mg per day. Participants with lower IS levels (1) display the narrowest spread and the fewest outliers, suggesting a lighter reliance on caffeine. In contrast, higher IS levels (4–5) show slightly more extreme outliers, indicating that some students with stronger imposter feelings may consume more caffeine. Overall, the data suggests that caffeine use is highly individualized and not strongly correlated with imposter syndrome ratings.

EDA4 - Caffeine consumption vs Campus Meal Consumption (Scatterplot, Ryan)

```
Caffeine_mg  = [col for col in df.columns if "caffeine" in
col.lower()]
Campus_Meals = [col for col in df.columns if "substantial" in
col.lower()]
Diet_Rating  = [col for col in df.columns if "diet" in col.lower()]

if Caffeine_mg:  df.rename(columns={Caffeine_mg[0]: 'Caffeine_mg'},
inplace=True)
if Campus_Meals: df.rename(columns={Campus_Meals[0]: 'Campus_Meals'},
inplace=True)
if Diet_Rating:  df.rename(columns={Diet_Rating[0]: 'Diet_Rating'},
```

```
inplace=True)

for col in ['Caffeine_mg', 'Campus_Meals', 'Diet_Rating']: df[col] =
pd.to_numeric(df[col], errors='coerce')

df = df.dropna(subset=['Caffeine_mg', 'Campus_Meals', 'Diet_Rating'])

# Scatter plot graphing
plt.figure(figsize=(10,6))
plt.scatter(df['Caffeine_mg'], df['Campus_Meals'], alpha=0.7,
color='blue')

# Linear regression line -- shows that there is negative correlation
between the two.
slope, intercept = np.polyfit(df['Caffeine_mg'], df['Campus_Meals'],
1)
x_vals = np.array([df['Caffeine_mg'].min(), df['Caffeine_mg'].max()])
y_vals = slope * x_vals + intercept
plt.plot(x_vals, y_vals, color='red', linewidth=2)

plt.xlabel("Caffeine intake (mg per day)")
plt.ylabel("Frequency of eating campus meals (# of times per week)")
plt.title("Caffeine Consumption vs Campus Meals Frequency")
plt.legend()
plt.show()
```

```
/var/folders/_2/3lrnsn7x5615sf09xw7cfcfm0000gn/T/
ipykernel_82385/2191714426.py:26: UserWarning: No artists with labels
found to put in legend.  Note that artists whose label start with an
underscore are ignored when legend() is called with no argument.
  plt.legend()
```


Caffeine Consumption vs Campus Meals Frequency
Frequency of eating campus meals (# of times per week)
Caffeine intake (mg per day)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
100
200
300
400
500

Introduction:

I am trying to find out if there are factors that affect a student's decision to eat food from campus dining. One of the related factors is the amount of caffeine consumed. We want to see if caffeine has any correlation with students' tendency to eat food from campus restaurants. I am expecting to find some kind of positive correlation between them since campus has several coffee spots (Starbucks, Coffee Bean, Celsius vending machines, etc…) as well as dining options.

Analysis:

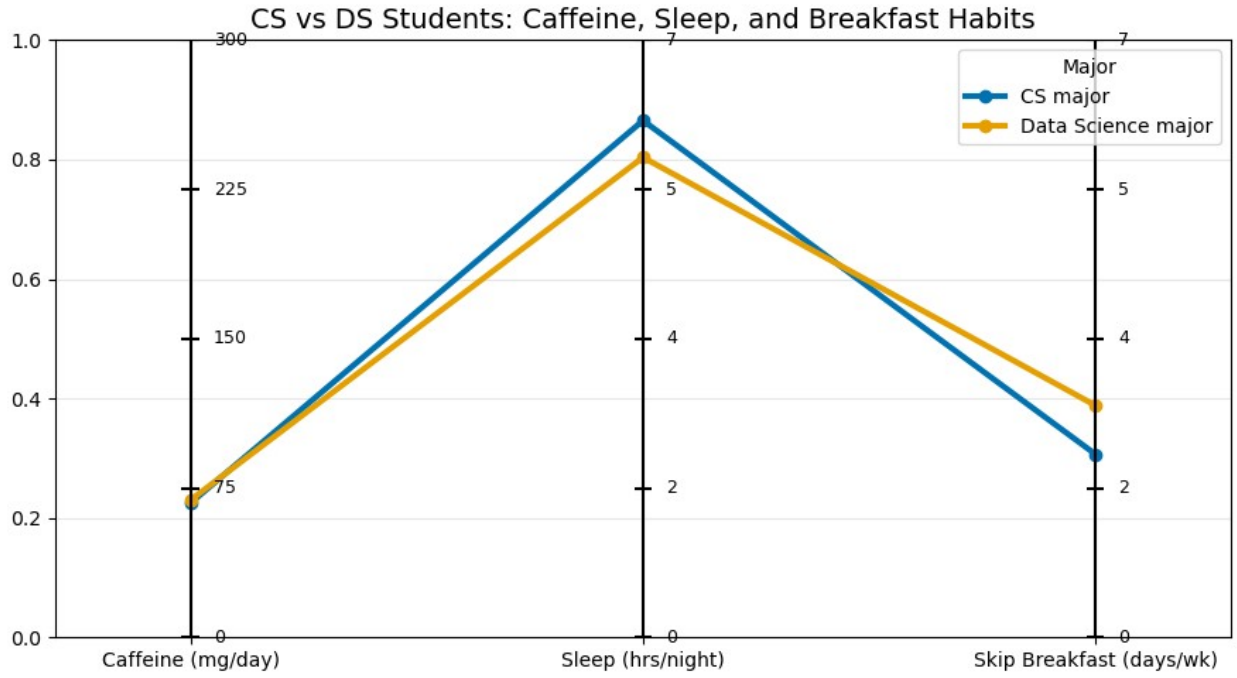
The scatter plot above maps the trend between two numerical variables: Typical caffeine intake (milligrams per day) and frequency of eating campus meals (times per week). Contrary to my initial expectation, there is a negative correlation between the two variables. As caffeine consumption increases, the less likely someone eats from campus food places.

EDA5 - CS vs DS Student Health Habits (Parallel Coordinate, Colin)

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import re

df = pd.read_csv("responses.csv")
col_major = 'What is your major/minor? (If your major is not listed,
please write it in "Other..")'
col_caff  = 'How many milligrams of caffeine do you consume per day,
on average? For example, a cup of coffee is 95 mg, a shot of espresso
is 64mg, a can of Celsius is 200 mg, and a  can of Coca Cola is 34 mg.
Answer with a number only.'
col_sleep = 'Fill in the following table over how often you did each
activity for in the past week. [Sleep per night (average hours)]'
col_skip  = 'How many days per week do you skip breakfast?'

df_sub = df[[col_major, col_caff, col_sleep, col_skip]].copy()
df_sub.columns = ['Major', 'Caffeine', 'Sleep', 'Skip']
```

```
df_sub = df_sub[df_sub['Major'].isin(['CS major', 'Data Science
major'])]

def to_mid(val):
    if pd.isna(val):
        return np.nan
    nums = [float(x) for x in re.findall(r'\d+\.?\d*', str(val))]
    if len(nums) == 1:
        return nums[0]
    elif len(nums) >= 2:
        return np.mean(nums[:2])
    return np.nan

df_sub['Caffeine'] = pd.to_numeric(df_sub['Caffeine'],
errors='coerce').clip(0, 300)
df_sub['Sleep'] = df_sub['Sleep'].apply(to_mid).clip(0, 7)
df_sub['Skip'] = df_sub['Skip'].apply(to_mid).clip(0, 7)

df_sub = df_sub.dropna(subset=['Caffeine', 'Sleep', 'Skip'])
avg = df_sub.groupby('Major')[['Caffeine', 'Sleep',
'Skip']].mean().reset_index()
fig, ax = plt.subplots(figsize=(9, 5))

x_pos = [0, 1, 2]
variables = ['Caffeine', 'Sleep', 'Skip']
scales = [(0, 300), (0, 7), (0, 7)]
colors = {'CS major': '#0072B2', 'Data Science major': '#E69F00'}

for major in avg['Major']:
    y_scaled = []
    for (var, (low, high)) in zip(variables, scales):
        val = avg.loc[avg['Major'] == major, var].values[0]
        scaled = (val - low) / (high - low)
        y_scaled.append(scaled)
    plt.plot(x_pos, y_scaled, marker='o', linewidth=3, label=major,
color=colors[major])
plt.xticks(x_pos, ['Caffeine (mg/day)', 'Sleep (hrs/night)', 'Skip
Breakfast (days/wk)'])
plt.title("CS vs DS Students: Caffeine, Sleep, and Breakfast Habits",
fontsize=14)
plt.legend(title="Major")
for i, (low, high) in enumerate(scales):
    plt.vlines(x_pos[i], 0, 1, color='black')
    ticks = np.linspace(0, 1, 5)
    values = np.linspace(low, high, 5)
    for t, v in zip(ticks, values):
        plt.hlines(t, x_pos[i] - 0.02, x_pos[i] + 0.02, color='black')
        plt.text(x_pos[i] + 0.05, t, f"{v:.0f}", va='center',
fontsize=9)
```

```
plt.xlim(-0.3, 2.3)
plt.ylim(0, 1)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```



Introduction:

This is a parallel plot comparing the amount of caffeine, hours of sleep, and number of breakfasts skipped per week, between data science and computer science. I am looking at these variables to see if I can determine any noticeable trends or differences that I can pick up on between caffeine consumption, sleep amount, and days of breakfast skipped compared amongst data and computer science students.

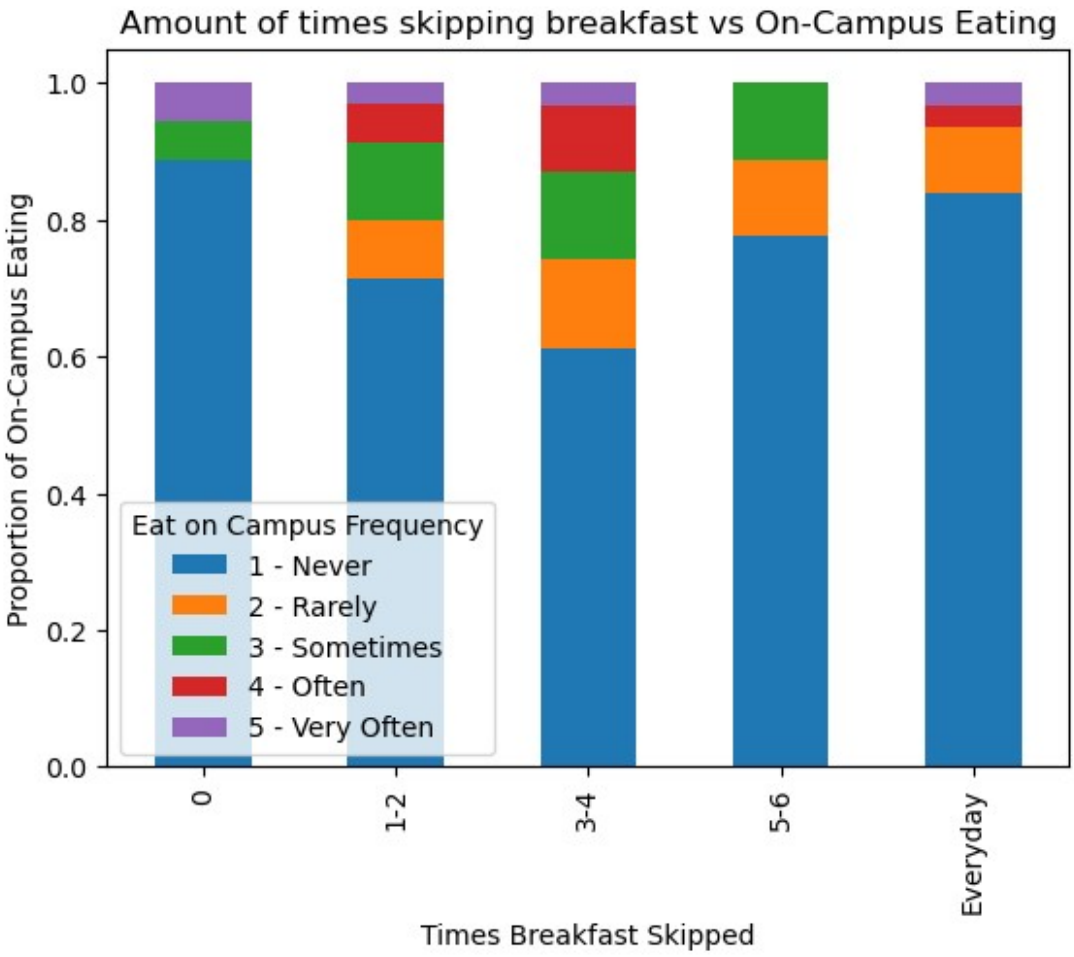
Analysis:

Looking at the amount of caffeine, hours of sleep, and days of skipped breakfast, both the data and computer science students consume the same amount of caffeine and get around the same sleep per night. Additionally, data science students skip breakfast at a slightly higher frequency than computer science students.

EDA6 - Amount of times skipping breakfast vs. On-Campus eating (Stacked bar, Colin)

```
table2 = pd.crosstab(df["How many days per week do you skip
breakfast?"], df["Please rate how often you eat at each of the
following on-campus meal options on a scale of 1 to 5 (1 being Never,
5 being Very Often). [Dining hall (Glasgow and Lothian)]"],
normalize="index")

table2.plot(kind="bar", stacked=True)
plt.title("Amount of times skipping breakfast vs On-Campus Eating")
plt.ylabel("Proportion of On-Campus Eating")
plt.xlabel("Times Breakfast Skipped")
plt.legend(title="Eat on Campus Frequency")
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```


Amount of times skipping breakfast vs On-Campus Eating
Proportion of On-Campus Eating
1.0
0.8
0.6
0.4
0.2
0.0
Eat on Campus Frequency
1 - Never
2 - Rarely
3 - Sometimes
4 - Often
5 - Very Often
0
1-2
3-4
5-6
Everyday
Times Breakfast Skipped

Introduction:

This stacked bar chart compares the two variables, the frequency of skipping breakfast and the frequency of eating meals on campus. I am comparing these two variables to see if students are more likely to eat on campus if they skip breakfast in the morning. I am predicting that those who skip breakfast everyday are more likely to eat on campus.

Analysis:

In every range of days students skip breakfast, the clear majority is always those who never eat on campus. There is a slightly higher proportion of those who eat breakfast when skipping 3-4 days of breakfast, however, there is no clear trend or pattern.

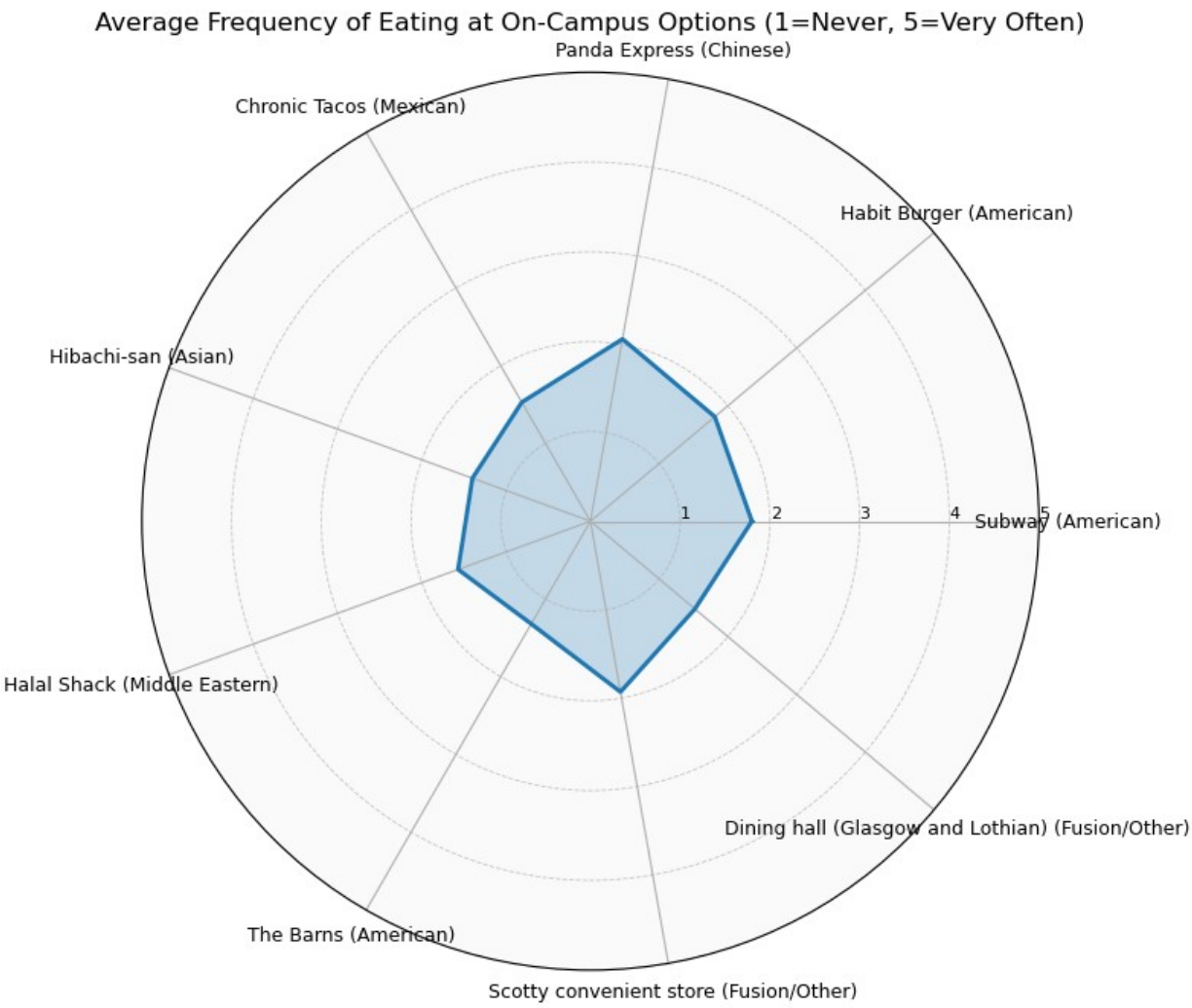
EDA7 - Campus Food Option Frequency (Spider, Jonathan)

```
# ---- Define restaurant-cuisine mapping ----
# You can replace this with actual restaurant names and cuisine types
restaurant_cuisines = {
    "The Barns": "American",
    "Subway": "American",
    "Panda Express": "Chinese",
    "Habit Burger": "American",
    "Hibachi-san": "Asian",
    "Chronic Tacos": "Mexican",
    "Dining hall (Glasgow and Lothian)" : "Fusion/Other",
    "Scotty convenient store" : "Fusion/Other",
    "Halal Shack" : "Middle Eastern"
    # Add more restaurants and their cuisines here
}

# ---- Select columns: on-campus meal frequency ----
meal_cols = [
    col for col in df.columns
    if "Please rate how often you eat at each of the following on-
campus meal options" in col
]

print(f"Meal columns detected: {len(meal_cols)}")
for c in meal_cols:
    print(" -", c)
```

```
# ---- Clean ratings to integers 1–5 ----
def clean_rating(x):
    """
    Converts Google Form mixed strings like '5', '5.0', '5 - Very
Often'
    to int 1–5; returns NaN if not parseable.
    """
    if pd.isna(x):
        return np.nan
    x = str(x).strip()
    for ch in x:
        if ch.isdigit():
            return int(ch)
    return np.nan

meal_data = df[meal_cols].applymap(clean_rating)

# ---- Compute averages (per restaurant) ----
avg_ratings = meal_data.mean().round(2)

# ---- Pretty labels (inside square brackets) with cuisine types ----
labels = [f"{col.split('[')[-1].split(']')[0]}
({restaurant_cuisines.get(col.split('[')[-1].split(']')[0],
'Unknown')})" for col in meal_cols]

# ---- Radar prep ----
values = avg_ratings.values.tolist()
values += values[:1]  # close loop

angles = np.linspace(0, 2 * np.pi, len(labels),
endpoint=False).tolist()
angles += angles[:1]

# ---- Plot (Radar) ----
plt.figure(figsize=(8, 8))
ax = plt.subplot(111, polar=True)

ax.plot(angles, values, linewidth=2, linestyle="solid",
color="#1f77b4")
ax.fill(angles, values, alpha=0.25, color="#1f77b4")

# ---- Axis labels ----
ax.set_xticks(angles[:-1])
ax.set_xticklabels(labels, fontsize=9)

# ---- Set explicit floor/ceiling and tick marks ----
ax.set_ylim(0, 3)
ax.set_yticks([1, 2, 3, 4, 5])
ax.set_yticklabels(["1", "2", "3", "4", "5"], fontsize=8)
ax.yaxis.grid(True, linestyle="--", linewidth=0.6, alpha=0.7)
```

```
# ---- Optional: emphasize visual spacing for ratings ----
ax.set_rlabel_position(0)
ax.set_facecolor("#fafafa")

plt.title("Average Frequency of Eating at On-Campus Options (1=Never,
5=Very Often)", pad=20)
plt.tight_layout()
plt.show()
```

```
Meal columns detected: 9
 - Please rate how often you eat at each of the following on-campus
meal options on a scale of 1 to 5 (1 being Never, 5 being Very Often).
[Subway]
 - Please rate how often you eat at each of the following on-campus
meal options on a scale of 1 to 5 (1 being Never, 5 being Very Often).
[Habit Burger]
 - Please rate how often you eat at each of the following on-campus
meal options on a scale of 1 to 5 (1 being Never, 5 being Very Often).
[Panda Express]
 - Please rate how often you eat at each of the following on-campus
meal options on a scale of 1 to 5 (1 being Never, 5 being Very Often).
[Chronic Tacos]
 - Please rate how often you eat at each of the following on-campus
meal options on a scale of 1 to 5 (1 being Never, 5 being Very Often).
[Hibachi-san]
 - Please rate how often you eat at each of the following on-campus
meal options on a scale of 1 to 5 (1 being Never, 5 being Very Often).
[Halal Shack]
 - Please rate how often you eat at each of the following on-campus
meal options on a scale of 1 to 5 (1 being Never, 5 being Very Often).
[The Barns]
 - Please rate how often you eat at each of the following on-campus
meal options on a scale of 1 to 5 (1 being Never, 5 being Very Often).
[Scotty convenient store]
 - Please rate how often you eat at each of the following on-campus
meal options on a scale of 1 to 5 (1 being Never, 5 being Very Often).
[Dining hall (Glasgow and Lothian)]

/var/folders/_2/3lrnsn7x5615sf09xw7cfcfm0000gn/T/
ipykernel_82385/2367298109.py:40: FutureWarning: DataFrame.applymap
has been deprecated. Use DataFrame.map instead.
  meal_data = df[meal_cols].applymap(clean_rating)
```


Average Frequency of Eating at On-Campus Options (1=Never, 5=Very Often)
Panda Express (Chinese)
Habit Burger (American)
Subway (American)
Dining hall (Glasgow and Lothian) (Fusion/Other)
Scotty convenient store (Fusion/Other)
The Barns (American)
Halal Shack (Middle Eastern)
Hibachi-san (Asian)
Chronic Tacos (Mexican)
1
2
3
4
5

Introduction:

This spider chart visualizes how frequently UCR students eat at these on-campus restaurants rated on a likert scale (1 = never, 5 = very often). The listed restaurants have their respective cuisine attached. Visualizing all restaurant ratings on a single radial axis, this chart offers a view of students dining habits.

Analysis:

Most ratings cluster below ratings of 2 showcasing that students don't eat on campus restaurants of all kinds that often. Among the places that are frequented are Panda express, and Scotty convenience store, indicating that students prefer these locations for their fast food choices and accessible food options.

EDA8 - Region of Favorite Cuisine vs On-Campus Restaurant Choice (Heat map, Lydia)

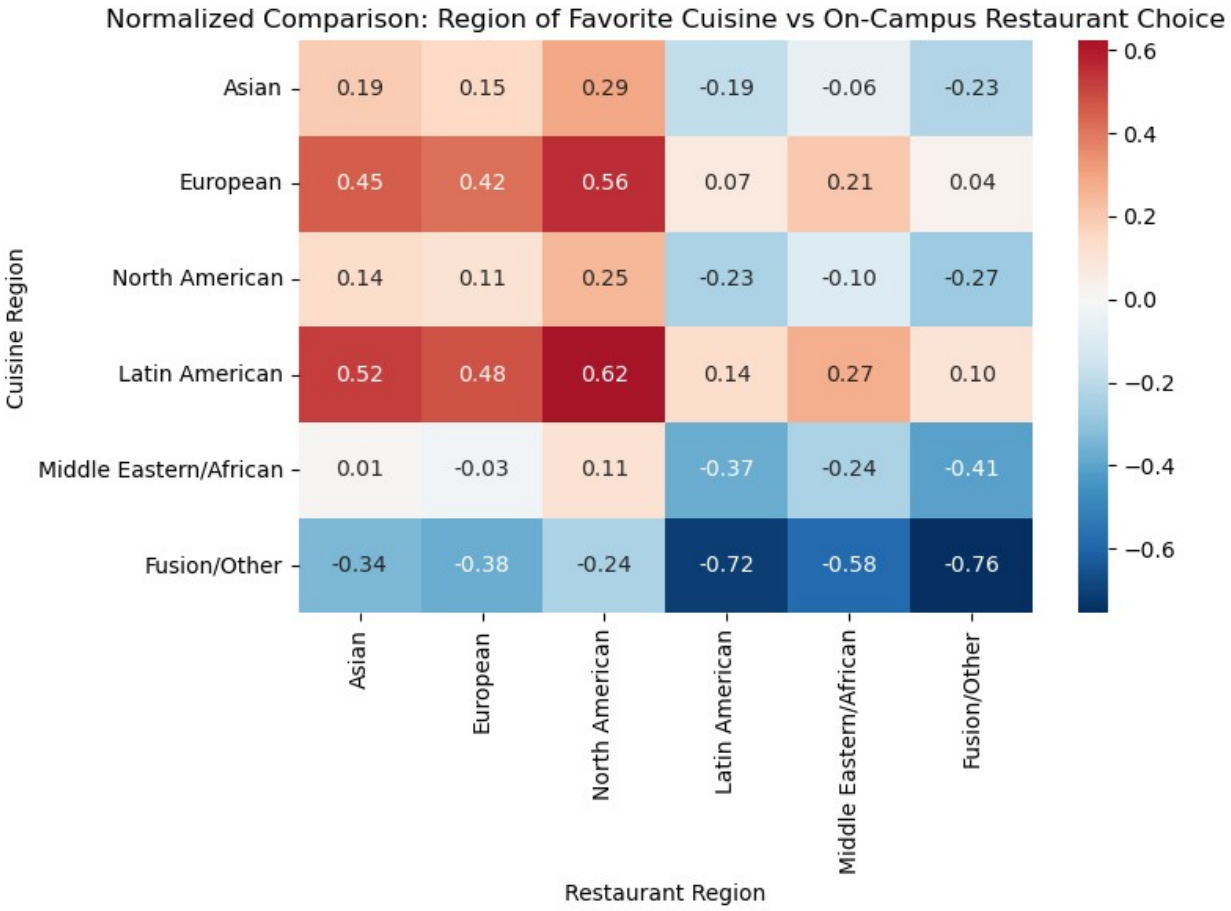
```
fav_cuisine = df[[c for c in df.columns if "Please rank the following
cuisines by your preference. (1 being dislike, 5 being like)" in c]]
fav_cuisine.columns = fav_cuisine.columns.str.extract(r'\[(.*)\]')[0]
# print(fav_cuisine.columns.tolist())

# convert to numeric
fav_cuisine = fav_cuisine.apply(lambda col:
col.astype(str).str.split(' - ').str[0])
fav_cuisine = fav_cuisine.apply(pd.to_numeric, errors='coerce')
# fav_cuisine

# extract restaurant rating columns
rest_rating_cols = df[[c for c in df.columns if "Please rate how often
you eat at each of the following on-campus meal options on a scale of
1 to 5 (1 being Never, 5 being Very Often)." in c]]
rest_rating_cols.columns = rest_rating_cols.columns.str.extract(r'\
[(.*)\]')[0]
# print(rest_rating_cols.columns.tolist()) # test print

# convert to numeric
rest_rating_cols = rest_rating_cols.apply(lambda col:
col.astype(str).str.split(' - ').str[0])
rest_rating_cols = rest_rating_cols.apply(pd.to_numeric,
errors='coerce')
```

```
# rest_rating_cols.head() # test print

# find other cuisine and rating pairs
df3 = df[["Please rank the following cuisines by your preference. (1
being dislike, 5 being like) [Other Cuisine (please specify in the
next question)]", "Please list any other cuisines not included."]]
df3 = df3.rename(columns={
   "Please rank the following cuisines by your preference. (1 being
dislike, 5 being like) [Other Cuisine (please specify in the next
question)]": "other_cuisine_rate",
    "Please list any other cuisines not included.": "cuisine_name"
})
# print(df3["cuisine_name"].unique())

# clean the data
df3_cleaned = df3.replace(['nan','0','NA', 'na', 'Na',
'N/A','N/a',"Couldn't think of any", "n/A", "no","none",
                           "Not sure","I don't know any others",
"N/a",'the rest was uncluded','None ', 'No','idk',
                           '','I dont know any other not
listed','-','Snacks',"Pizza",'chipotle','convenience foods',
                           'Anything','Soul Food'], pd.NA).dropna()
# print(df3_cleaned["cuisine_name"].unique())

# convert to numeric
df3_cleaned['other_cuisine_rate'] =
(df3_cleaned['other_cuisine_rate'].astype(str).str.split(' -
').str[0])
df3_cleaned['other_cuisine_rate'] =
pd.to_numeric(df3_cleaned['other_cuisine_rate'], errors='coerce')
# df3_cleaned

# Print column names of the full dataset and the combined cuisine
dataframe
# print(f"\ndf3_whole columns ({len(df3_whole.columns)}):")
# print(df3_whole.columns.tolist())

# assign the cuisine columns to region groups
cuisine_region_map = {
    'Chinese food': 'Asian',
    'Japanese food': 'Asian',
    'Korean food': 'Asian',
    'Indian food': 'Asian',
    'Vietnamese food': 'Asian',
    'Thai': 'Asian',
    'Thai food': 'Asian',
    'tai food': 'Asian',
    'Filipino': 'Asian',
    'Filipino food': 'Asian',
    'filipino': 'Asian',
```

```
    'Indonesian food': 'Asian',
    'Malaysian Food': 'Asian',
    'Mongolian': 'Asian',
    'Pakistani Food': 'Asian',
    'Sri Lankan': 'Asian',
    'Taiwanese food': 'Asian',

    'Italian food': 'European',
    'French food': 'European',
    'Spanish Food': 'European',
    'Eastern European': 'European',
    'Russian and Eastern European dishes': 'European',

    'Mexican food': 'Latin American',
    'Salvadorean': 'Latin American',

    'American food': 'North American',

    'Mediterranean food': 'Middle Eastern/African',
    'Middle Eastern': 'Middle Eastern/African',
    'Not sure if it counts as mediterranean, but I like food from the
Middle East.': 'Middle Eastern/African',

    'African Food': 'Middle Eastern/African',
    'Ethiopian': 'Middle Eastern/African',
    'Nigerian, west African ': 'Middle Eastern/African',
    'South African': 'Middle Eastern/African',

    'Fusion / Modern mix cuisine': 'Fusion/Other'
}

restaurant_region_map = {
    'Subway': 'European',
    'Habit Burger': 'North American',
    'Panda Express': 'Asian',
    'Chronic Tacos': 'Latin American',
    'Hibachi-san': 'Asian',
    'Halal Shack': 'Middle Eastern/African',
    'The Barns': 'North American',
    'Scotty convenient store': 'North American',
    'Dining hall (Glasgow and Lothian)': 'Fusion/Other'
}

# Clean column names (for consistency)
fav_cuisine.columns = fav_cuisine.columns.str.strip().str.lower()
rest_rating_cols.columns =
rest_rating_cols.columns.str.strip().str.lower()

# Normalize mapping keys
cuisine_region_map = {k.lower().strip(): v for k, v in
```

```
cuisine_region_map.items()}
restaurant_region_map = {k.lower().strip(): v for k, v in
restaurant_region_map.items()}

# Group cuisines by region
fav_cuisine_by_region = pd.DataFrame(index=fav_cuisine.index)

for region in sorted(set(cuisine_region_map.values())):
    # find all cuisine columns belonging to this region that exist in
dataframe
    region_cols = [c for c, r in cuisine_region_map.items() if r ==
region and c in fav_cuisine.columns]
    if region_cols:
        # calculate mean rating across those cuisines for each
participant
        fav_cuisine_by_region[region] =
fav_cuisine[region_cols].mean(axis=1, skipna=True)
    else:
        # if no columns found, fill with NaN (keeps row count intact)
        fav_cuisine_by_region[region] = pd.NA

# Group restaurants by region
rest_rating_by_region = pd.DataFrame(index=rest_rating_cols.index)

for region in sorted(set(restaurant_region_map.values())):
    region_cols = [r for r, reg in restaurant_region_map.items() if
reg == region and r in rest_rating_cols.columns]
    if region_cols:
        rest_rating_by_region[region] =
rest_rating_cols[region_cols].mean(axis=1, skipna=True)
    else:
        rest_rating_by_region[region] = pd.NA


# Make sure the two dataframes share the same participants
common_index =
fav_cuisine_by_region.index.intersection(rest_rating_by_region.index)
fav_cuisine_by_region = fav_cuisine_by_region.loc[common_index]
rest_rating_by_region = rest_rating_by_region.loc[common_index]

# Normalize per participant (row-wise z-score)
def row_zscore(df):
    """Normalize each participant's ratings (row) by their mean and
std."""
    return df.sub(df.mean(axis=1), axis=0).div(df.std(axis=1), axis=0)

fav_norm = row_zscore(fav_cuisine_by_region)
rest_norm = row_zscore(rest_rating_by_region)

# Compute heatmap data
```

```
heatmap_data = pd.DataFrame(
    index=fav_norm.columns,
    columns=rest_norm.columns,
    data=[
        [
            np.mean(fav_norm[cuisine]) +
np.mean(rest_norm[restaurant])
            for restaurant in rest_norm.columns
        ]
        for cuisine in fav_norm.columns
    ]
)

# Normalize to mean-centered scale for better contrast
heatmap_data = heatmap_data - heatmap_data.mean().mean()

region_order = [
    "Asian",
    "European",
    "North American",
    "Latin American",
    "Middle Eastern/African",
    "Fusion/Other"
]
heatmap_ordered = heatmap_data.loc[
    [r for r in region_order if r in heatmap_data.index],
    [c for c in region_order if c in heatmap_data.columns]
]

plt.figure(figsize=(8, 6))
sns.heatmap(heatmap_ordered, annot=True, cmap="RdBu_r", center=0,
fmt=".2f")
plt.title("Normalized Comparison: Region of Favorite Cuisine vs On-
Campus Restaurant Choice")
plt.xlabel("Restaurant Region")
plt.ylabel("Cuisine Region")
plt.tight_layout()
plt.show()
```


Normalized Comparison: Region of Favorite Cuisine vs On-Campus Restaurant Choice
Cuisine Region
Asian
European
North American
Latin American
Middle Eastern/African
Fusion/Other
0.19 0.15 0.29 -0.19 -0.06 -0.23
0.45 0.42 0.56 0.07 0.21 0.04
0.14 0.11 0.25 -0.23 -0.10 -0.27
0.52 0.48 0.62 0.14 0.27 0.10
0.01 -0.03 0.11 -0.37 -0.24 -0.41
-0.34 -0.38 -0.24 -0.72 -0.58 -0.76
Asian
European
North American
Latin American
Middle Eastern/African
Fusion/Other
Restaurant Region
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

Introduction:

This heatmap uses the "favorite cuisine types" and "how often they eat at on-campus restaurants." I aim to explore whether individuals' cuisine preferences are associated with their dining frequency on campus, and to identify if certain cuisine interests align with higher or lower on-campus dining engagement. I group the cuisines and the on-campus restauratns by regions and normalized the rating to better visualize the relationship. It shows the normalized relationship between participants' favorite cuisine regions and their frequency of eating at on-campus restaurants representing those regions.

Analysis:

A clear pattern shows that European and Latin American cuisines stand out with the strongest positive associations: students who prefer these cuisines also tend to visit corresponding on-campus restaurants more frequently, such as Subway and Chronic Taco. In contrast, Fusion/Other cuisines display consistent negative values across all restaurant types, suggesting that students who favor diverse or unconventional cuisines find fewer on-campus options matching their tastes. The weaker or near-zero correlations for Middle Eastern/African and North American cuisines indicate more neutral or varied dining patterns among those groups. Overall, the results highlight a gap between student cuisine preferences and the available on-campus restaurant offerings, especially for Fusion and Middle Eastern/African foods.

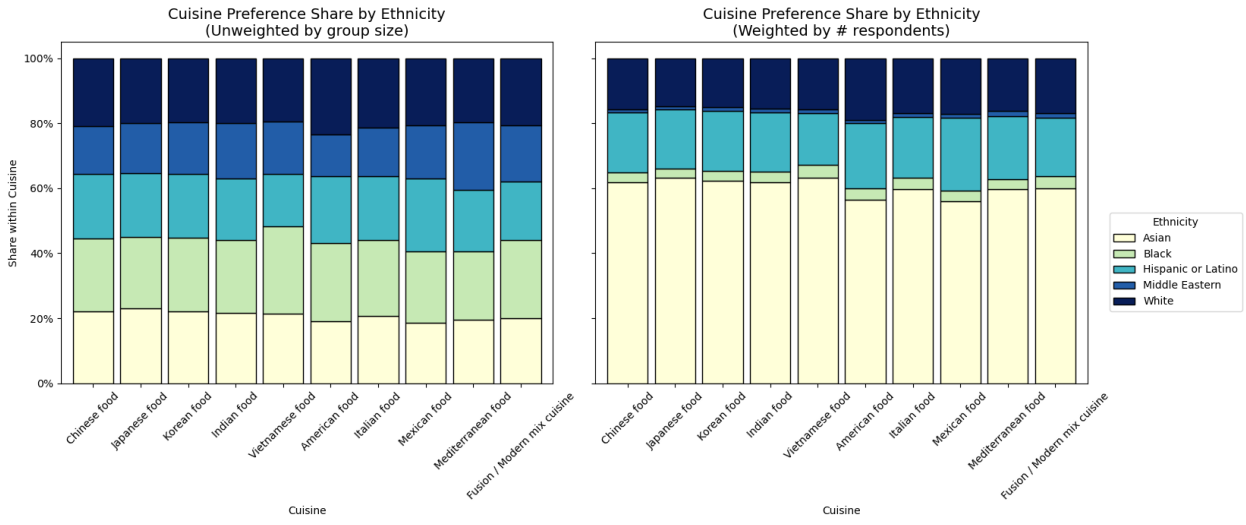
EDA9 - Cuisine Preference based on Ethnic Background (Stacked bar, Jonathan)

```
def _norm(s: str) -> str:
    s = str(s)
    s = s.replace("\r"," ").replace("\n"," ")
    s = s.replace("“","'").replace("”","'").replace("’","'")
    s = re.sub(r"\s+"," ", s).strip()
    return s
```

```
df.columns = [_norm(c) for c in df.columns]

# ---------- Detect columns ----------
eth_cols = [c for c in df.columns if "ethnic" in c.lower()]
assert eth_cols, "Could not find an ethnicity column (searching for
'ethnic')."
eth_col = eth_cols[0]

cuisine_cols = [c for c in df.columns
                if "please rank the following cuisines by your
preference" in c.lower()]
assert cuisine_cols, "No cuisine preference columns found."

# ---------- Clean Likert to 1–5 ----------
def clean_rating(x):
    if pd.isna(x): return np.nan
    for ch in str(x).strip():
        if ch.isdigit():
            return int(ch)
    return np.nan

for c in cuisine_cols:
    df[c] = df[c].apply(clean_rating)

# ---------- Simplified ethnicity (explode multi-select) ----------
ETH_PATTERN = {
    "Asian": re.compile(r"\basian\b|\bchinese\b|\bkorean\b|\bjapanese\
b|\bfilipin[oa]\b|\bindian\b|\bviet", re.I),
    "Black": re.compile(r"\bblack\b|\bafrican\s*american\b|\bafrican\
b", re.I),
    "Hispanic or Latino": re.compile(r"\bhispanic\b|\blatin[oa]\b|\
blatinx\b|\blatine\b", re.I),
    "Middle Eastern": re.compile(r"\bmiddle[-\s]*eastern\b|\bmena\b|\
barab(ic)?\b|\biranian\b|\bpersian\b|\bturk(ish)?\b|\bkurd(ish)?\b",
re.I),
    "White": re.compile(r"\bwhite\b|\bcaucasian\b|\beuropean\b",
re.I),
}
keep_order = ["Asian","Black","Hispanic or Latino","Middle
Eastern","White"]

def split_ethnicities(val):
    if pd.isna(val): return []
    s = _norm(val).lower()
    parts = re.split(r"[;,/]| and | & ", s)
    found = set()
    for p in parts:
        for label, rx in ETH_PATTERN.items():
            if rx.search(p):
                found.add(label)
```

```
    return sorted(found)

tmp = df[[eth_col] + cuisine_cols].copy()
tmp["Ethnicity"] = tmp[eth_col].apply(split_ethnicities)
tmp = tmp.explode("Ethnicity", ignore_index=True)
tmp = tmp[tmp["Ethnicity"].isin(keep_order)]

# ---------- Compute: means (unweighted), sums (weighted by n), counts
----------
means = tmp.groupby("Ethnicity")
[cuisine_cols].mean().reindex(keep_order)
sums  = tmp.groupby("Ethnicity")
[cuisine_cols].sum().reindex(keep_order)
counts = tmp.groupby("Ethnicity")
[cuisine_cols].count().reindex(keep_order).fillna(0).astype(int)

# ---------- Pretty cuisine labels ----------
pretty = {c: (c.split('[')[-1].split(']')[0] if '[' in c and ']' in c
else c) for c in cuisine_cols}
means = means.rename(columns=pretty)
sums  = sums.rename(columns=pretty)

# ---------- Remove "Other Cuisine" ----------
to_drop = [c for c in means.columns if "other cuisine" in c.lower()]
if to_drop:
    means = means.drop(columns=to_drop)
    sums = sums.drop(columns=to_drop)

# ---------- Convert to 100% stacked shares ----------
unweighted_share = means.divide(means.sum(axis=0), axis=1)
weighted_share = sums.divide(sums.sum(axis=0), axis=1)

# ---------- Plot side-by-side (clean shared legend) ----------
fig, axes = plt.subplots(1, 2, figsize=(16, 7), sharey=True)

# Unweighted
unweighted_share.T.plot(
    kind="bar", stacked=True, ax=axes[0],
    width=0.85, edgecolor="black", cmap="YlGnBu", legend=False
)
axes[0].set_title("Cuisine Preference Share by Ethnicity\n(Unweighted
by group size)", fontsize=14)
axes[0].set_xlabel("Cuisine"); axes[0].set_ylabel("Share within
Cuisine")
axes[0].yaxis.set_major_formatter(plt.FuncFormatter(lambda y,_:
f"{int(y*100)}%"))
axes[0].tick_params(axis='x', rotation=45)

# Weighted
weighted_share.T.plot(
```

```
    kind="bar", stacked=True, ax=axes[1],
    width=0.85, edgecolor="black", cmap="YlGnBu", legend=False
)
axes[1].set_title("Cuisine Preference Share by Ethnicity\n(Weighted by
# respondents)", fontsize=14)
axes[1].set_xlabel("Cuisine"); axes[1].set_ylabel("")
axes[1].yaxis.set_major_formatter(plt.FuncFormatter(lambda y,_:
f"{int(y*100)}%"))
axes[1].tick_params(axis='x', rotation=45)

# ---------- Single shared legend ----------
handles, labels = axes[1].get_legend_handles_labels()
if not handles:  # fallback if pandas version changes handle retrieval
    handles, labels = axes[0].get_legend_handles_labels()

fig.legend(handles, labels, title="Ethnicity", loc="center right",
           bbox_to_anchor=(1.03, 0.5), fontsize=10)

plt.tight_layout(rect=[0,0,0.93,1])
plt.show()

# ---------- Optional: print respondent counts ----------
print("Respondent counts by ethnicity (after explode):")
print(counts.max(axis=1).reindex(keep_order))
```




```
Respondent counts by ethnicity (after explode):
Ethnicity
Asian                 81
Black                  4
Hispanic or Latino    27
Middle Eastern         2
White                 22
dtype: int64
```

Introduction:

The Stacked bar chart displaying the average rating ethnicities gave towards certain cuisines. Each plot scale from each bar represents a type of cuisine (chinese, japanese etc) with the bar composed of responses from ethnicities. The left chart represents raw unweighted distribution and treats each ethnicity equally. The right bar chart represents the ethnicities by the number of respondents per group.

Analysis:

Based on these two charts, it seems that there isn't a relation between a person's ethnicity and their preference for cuisine, as all answers are about the same across the left stacked bar chart. The right stacked bar chart is to demonstrate the disparity between responding groups of our data, meaning we can't say for certain that this represents the preferences for underrepresented ethnicity groups.

Hypothesis

EDA1 - Mental Health Level vs Commuting Distance:

The initial hypothesis for the Mental Health Level vs. Commuting distance visualization (3.1) was a linear negative correlation. That is, the less time a student spends commuting, the higher their mental health level would be

Null hypothesis: There is no linear correlation between mental health level and commuting distance (r=0).

Alternative hypothesis: There is a correlation between the two variables (r = 0).

EDA4 - Caffeine Consumption vs Campus Meal Consumption:

Null hypothesis: There is no linear correlation between caffeine consumption and frequency of eating campus food (r = 0).

Alternative hypothesis: There is a positive correlation between the two variables ($r > 0$).

EDA9 - Cuisine Preference based on Ethnic Background:

The initial hypothesis for the relationship between students' ethnicities and their cuisine preferences (Figure 9) was a positive association.

Null Hypothesis: There is no statistical relationship between a student's ethnicity and their cuisine preference.

Alternative hypothesis: There is statistical relationship between a student's ethnicity and their cuisine preference.

Statistical Tests:

EDA1 - Mental Health Level vs Commuting Distance:

From the google spreadsheet below, we can infer that the hypothesis is not true. With a Chi-square value of 16.874, and a DF of 16, we get a P-value of 0.39375.

This insignificance shows that there is no correlation between the two variables.

Link to the Chi-Square test:
https://docs.google.com/spreadsheets/d/17tkJw5mBfaPDacd1wRBbFCCSg-2OcoROHf2IEuV_2JE/edit?gid=0#gid=0

EDA4: Caffeine Consumption vs Campus Food Consumption

From the Pearson Correlation, there is a negative correlation between the two variables at Rxy = -0.1193.

The coefficient is weak, but the two variables are inversely related. The null hypothesis is rejected, and the alternative hypothesis is accepted, but the correlation is negative ($r < 0$).

Link to the Pearson Correlation Test result:
https://docs.google.com/spreadsheets/d/1Z9t87CgnTzITcG6xMT48wEhlaXgj9TliHoIDhvVV42w/edit?usp=sharing

EDA9 - Cuisine Preference based on Ethnic Background

```
alpha = 0.05  # set your significance level (e.g., 0.05, 0.01, 0.001)

# Use the original cuisine columns but drop the "Other Cuisine..." one
(to match your plots)
filtered_cuisine_cols = []
for c in cuisine_cols:
    pretty_name = pretty.get(c, c)
    if "other cuisine" not in str(pretty_name).lower():
        filtered_cuisine_cols.append(c)

chi_results = []  # (pretty_name, chi2, dof, p, N, critical, decision)

print("\n--- Chi-Square Test of Independence: Ethnicity × Cuisine
Preference ---\n")
for c in filtered_cuisine_cols:
    pretty_name = pretty.get(c, c)

    # Contingency table: rows = Ethnicity (in keep_order), cols =
rating 1..5
    ct = pd.crosstab(tmp["Ethnicity"], df[c])  # df[c] already cleaned
to 1–5 in your code

    # Reindex rows to consistent ethnicity order; keep only rating
columns present
    ct = ct.reindex(index=keep_order).dropna(how="all")
    if ct.shape[0] < 2 or ct.shape[1] < 2:
        # Not enough data to test
        print(f"{pretty_name}: skipped (insufficient data)")
        continue

    chi_stat, p_val, dof, expected = chi2_contingency(ct)
    critical = chi2.ppf(1 - alpha, dof)
    decision = "Reject H₀" if chi_stat > critical else "Fail to Reject
H₀"

    chi_results.append((pretty_name, chi_stat, dof, p_val,
int(ct.to_numpy().sum()), critical, decision))

    print(f"{pretty_name}:")
    print(f"  χ² = {chi_stat:.2f},  df = {dof},  critical@α={alpha} →
```

```
{critical:.2f},  p = {p_val:.4f}")
    print(f"  → {decision} — ethnicity and rating are "
          f"{'associated' if decision=='Reject H₀' else 'not
significantly associated'}.\n")

# Summary table sorted by p-value
chi_df = pd.DataFrame(chi_results, columns=["Cuisine", "Chi2", "df",
"p_value", "N", "Critical", "Decision"])
display(chi_df.sort_values("p_value").reset_index(drop=True))
```

```
--- Chi-Square Test of Independence: Ethnicity × Cuisine Preference
---

Chinese food:
  χ² = 12.00,  df = 16,  critical@α=0.05 → 26.30,  p = 0.7442
  → Fail to Reject H₀ — ethnicity and rating are not significantly
associated.

Japanese food:
  χ² = 6.37,  df = 16,  critical@α=0.05 → 26.30,  p = 0.9835
  → Fail to Reject H₀ — ethnicity and rating are not significantly
associated.

Korean food:
  χ² = 11.08,  df = 16,  critical@α=0.05 → 26.30,  p = 0.8045
  → Fail to Reject H₀ — ethnicity and rating are not significantly
associated.

Indian food:
  χ² = 7.29,  df = 16,  critical@α=0.05 → 26.30,  p = 0.9673
  → Fail to Reject H₀ — ethnicity and rating are not significantly
associated.

Vietnamese food:
  χ² = 10.45,  df = 16,  critical@α=0.05 → 26.30,  p = 0.8421
  → Fail to Reject H₀ — ethnicity and rating are not significantly
associated.

American food:
  χ² = 14.35,  df = 16,  critical@α=0.05 → 26.30,  p = 0.5729
  → Fail to Reject H₀ — ethnicity and rating are not significantly
associated.

Italian food:
  χ² = 19.15,  df = 16,  critical@α=0.05 → 26.30,  p = 0.2609
  → Fail to Reject H₀ — ethnicity and rating are not significantly
associated.

Mexican food:
```

```
  χ² = 12.99,  df = 16,  critical@α=0.05 → 26.30,  p = 0.6734
  → Fail to Reject H₀ — ethnicity and rating are not significantly
associated.

Mediterranean food:
  χ² = 15.15,  df = 16,  critical@α=0.05 → 26.30,  p = 0.5140
  → Fail to Reject H₀ — ethnicity and rating are not significantly
associated.

Fusion / Modern mix cuisine:
  χ² = 16.47,  df = 16,  critical@α=0.05 → 26.30,  p = 0.4207
  → Fail to Reject H₀ — ethnicity and rating are not significantly
associated.


                        Cuisine       Chi2  df   p_value    N
Critical  \
0                 Italian food  19.150907  16  0.260915  120
26.296228
1  Fusion / Modern mix cuisine  16.468846  16  0.420745  120
26.296228
2           Mediterranean food  15.145821  16  0.513985  120
26.296228
3                American food  14.347163  16  0.572868  120
26.296228
4                 Mexican food  12.991299  16  0.673394  120
26.296228
5                 Chinese food  11.997165  16  0.744175  120
26.296228
6                  Korean food  11.080584  16  0.804484  120
26.296228
7              Vietnamese food  10.449726  16  0.842112  120
26.296228
8                  Indian food   7.286024  16  0.967346  120
26.296228
9                Japanese food   6.373273  16  0.983539  120
26.296228

             Decision
0  Fail to Reject H₀
1  Fail to Reject H₀
2  Fail to Reject H₀
3  Fail to Reject H₀
4  Fail to Reject H₀
5  Fail to Reject H₀
6  Fail to Reject H₀
7  Fail to Reject H₀
8  Fail to Reject H₀
9  Fail to Reject H₀
```

Chi-square critical value at $\alpha = 0.05$ and df = 16 was 26.30.

Our tests indicate there is no statistical significance between ethnicity and cuisine as all the chi square values were less than 26.30, thus we fail to reject our null hypothesis.

Overall Conclusion

Our original plan was to determine the factors that contributed to shaping a UCR student's choice of on-campus dining options. When we received the completed data set with all of our survey questions compiled into one spreadsheet, we realized we could form more varied hypotheses using the data from other groups' questions. In addition to investigating our original question, we were also able to explore questions regarding an individual's mental health and the factors affecting it.

The final result we came to only covered part of our initial topic. The first few of our project's EDAs (1–3) served as a rough outline of what other relevant data could be useful to answering our initial question. Variables such as mental health level and imposter syndrome were some that we compared, for example, to check for correlations or relationships.

Then, EDAs 4–9 focused more directly on our initial topic question, where most of the variables involved were food-related. Here, we checked factors such as ethnic background and caffeine consumption level for any correlation to our topic.

Group Contribution

Stuart

- Created scatterplot visualization displaying mental health vs. commuting distance relation.
- Hypothesis for that visualization and chi-square analysis.
- Overall conclusion write-up.

Lydia

- Created violin plot for imposter syndrome vs caffeine intake.
- Created violin plot imposter syndrome vs overall mental health level.
- Created heatmap fav cuisine vs on-campus restaurant frequency.
- Compile and final Jupyter Notebook report.

Ryan

- Created scatterplot visualization showing likelihood of eating from campus restaurants/dining vs daily caffeine intake.
- Hypothesis #2 (for EDA 4) with Pearson Correlation.
- Overall Conclusion write-up.

Jonathan

- Created spider chart displaying average frequency for on campus restaurants
- Created Stacked bar chart displaying average preference for cuisine by ethnicity.
- Chi-square correlation for hypothesis #3 (Cuisine Preference based on Ethnic Background).

Colin

- Create a stacked bar chart to find the relationship between amount of sleep and tendency to eat breakfast.
- Parallel plot visualization comparing DS/CS students in how much they consume caffeine, sleep, and eat breakfast.