This sample is broken up into two sections, which reflect the two means through which I learned to code Python.

The first section reflects self-study via Harvard's online CS50 lectures for Python and their corresponding exercises. The second sections reflects some of my data visualization work at CSU Northridge.

A link with the instructions for each CS50 exercise is included just above my solutions. As far as I'm aware, there are no solutions to these exercises posted online by Harvard. The solutions provided below are uniquely my own and can be cross-checked with community solutions posted online to validate their uniqueness.

Section 1: CS50

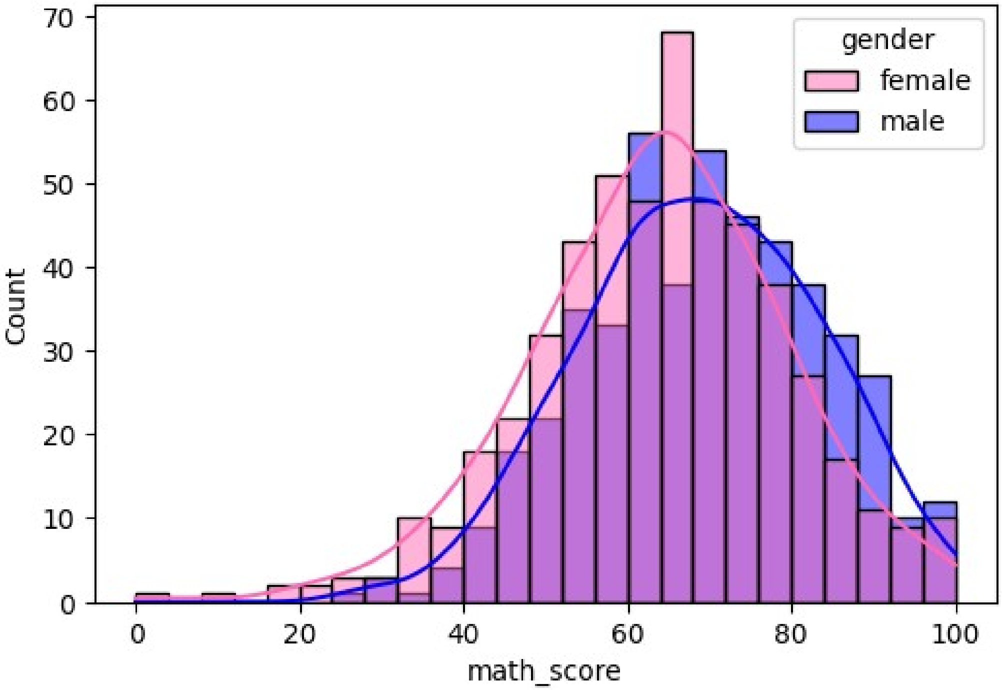
```
#/////////////////////////////////////////////////////////////////////
/////////
# https://cs50.harvard.edu/python/2022/psets/2/plates/
# Vanity Plates

def main():
    plate = input("Plate: ")
    if is_valid(plate):
        print("Valid")
    else:
        print("Invalid")

#---------------------------------------------------------------

def is_valid(s):
    if 2 <= len(s) <= 6 and s[0].isalpha() and s[1].isalpha():

        # Flag variable
        has_numeric = False

        # Digits cannot appear in the middle of a license plate
        for i in range(2, len(s)):
            if s[i].isdigit():
                has_numeric = True
            elif s[i].isalpha() and has_numeric:
                return False

            # The first digit cannot be zero
            if i >= 2 and s[i] == '0' and s[i-1].isalpha():
                return False
            else:
              pass

            # No periods, spaces, punctuation marks, or special
characters
            if s[i] in ['.', ' ', ',', ';', ':', '?', '!', '-', '_',
'+', '=', '@', '#', '$', '%', '^', '&', '*', '(', ')']:
                return False

        return True

    else:
        return False

#---------------------------------------------------------------

main()

#//////////////////////////////////////////////////////////////////////
/////////
# https://cs50.harvard.edu/python/2022/psets/2/camel/
# camelCase

camelCase = input('Enter camelCase variable name: ')

snake_case = ''

for char in camelCase:

  if char.isupper():
    char = char.lower()
    snake_case += '_'
    snake_case += str(char)

  else:
    snake_case += str(char)

print(snake_case)

#//////////////////////////////////////////////////////////////////////
/////////
# https://cs50.harvard.edu/python/2022/psets/2/coke/
# Coke Machine

left_owed = 50
print('One can of Coca-Cola is $0.50')

#---------------------------------
while True:

  if left_owed > 0:
    coin_value = int(input('Insert coin: '))

    if coin_value == 25 or coin_value == 10 or coin_value == 5:
      left_owed = left_owed - coin_value
      print(f'{left_owed} cents left')
    else:
      print('Invalid coin. Only 25, 10, and 5 cent coins are
accepted.')
      continue

  else:
      break
#---------------------------------

if left_owed < 0:
  print(f' Change is {left_owed - (2*left_owed)} cents. Enjoy your
Coke!')
else:
  print('Enjoy your Coke!')

#/////////////////////////////////////////////////////////////////////
/////////
# https://cs50.harvard.edu/python/2022/psets/2/twttr/
# Just setting up my twttr

vowel_string = input('Give a string: ')
no_vowels = ''

for char in vowel_string:
  match str(char):
    case 'A'|'E'|'I'|'O'|'U'|'a'|'e'|'i'|'o'|'u':
      no_vowels += ''
    case _:
      no_vowels += str(char)

print(no_vowels)

#/////////////////////////////////////////////////////////////////////
/////////
# https://cs50.harvard.edu/python/2022/psets/3/fuel/
# Fuel Gauge

def main():

  X, Y = input_filter()

  if X/Y >= 0.99:
    print('F')
  elif X/Y <= 0.01:
    print('E')
  else:
    print(str(round(X/Y*100)) + '%')

#---------------------------------------------------------------

def input_filter():
  while True:
    try:
      fraction_input = input('Enter fuel gauge in the form X/Y: ')
      X, Y = map(int, fraction_input.split('/'))
      return X, Y

    except (ValueError, ZeroDivisionError):
      print('Please follow the formatting guidelines.')

    else:
      return X, Y

#---------------------------------------------------------------

main()

#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
```

Section 2: Data Visualization

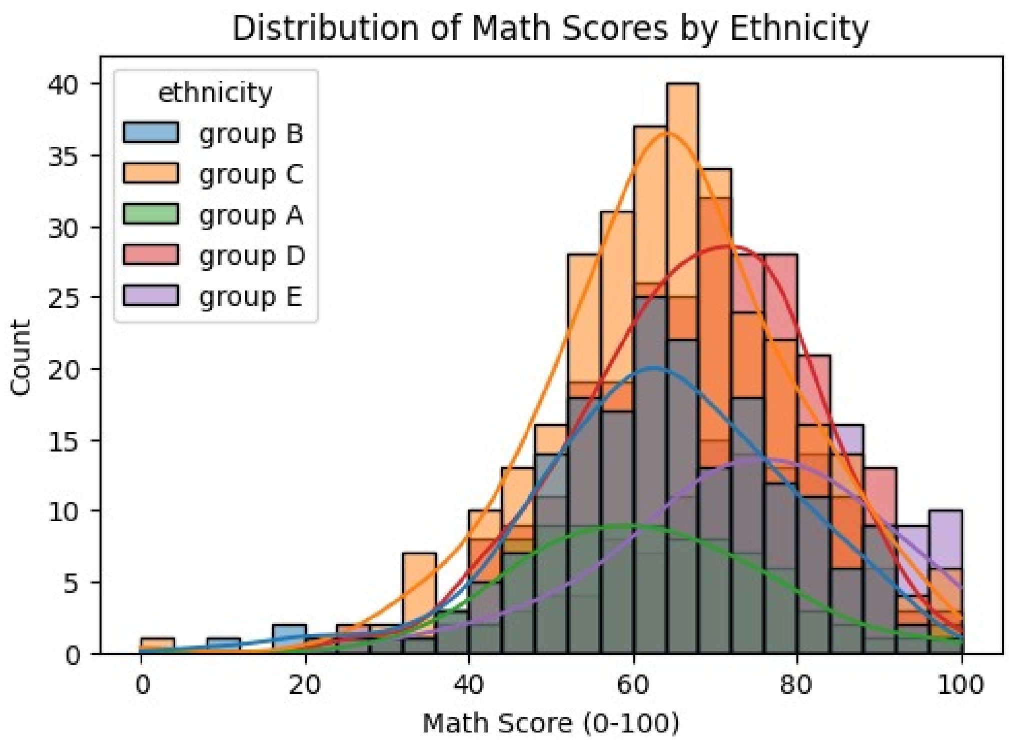
```
import matplotlib.pyplot as plt
from matplotlib_venn import venn2

import seaborn as sns
import pandas as pd
import numpy as np

from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split

df = pd.read_csv('student.csv')
df.head(3)
```

```
   gender ethnicity parental_education     lunch
test_preparation_course  \
0  female   group B  bachelor's degree  standard
none
1  female   group C       some college  standard
completed
2  female   group B    master's degree  standard
none

   math_score  reading_score  writing_score
0          72             72             74
```

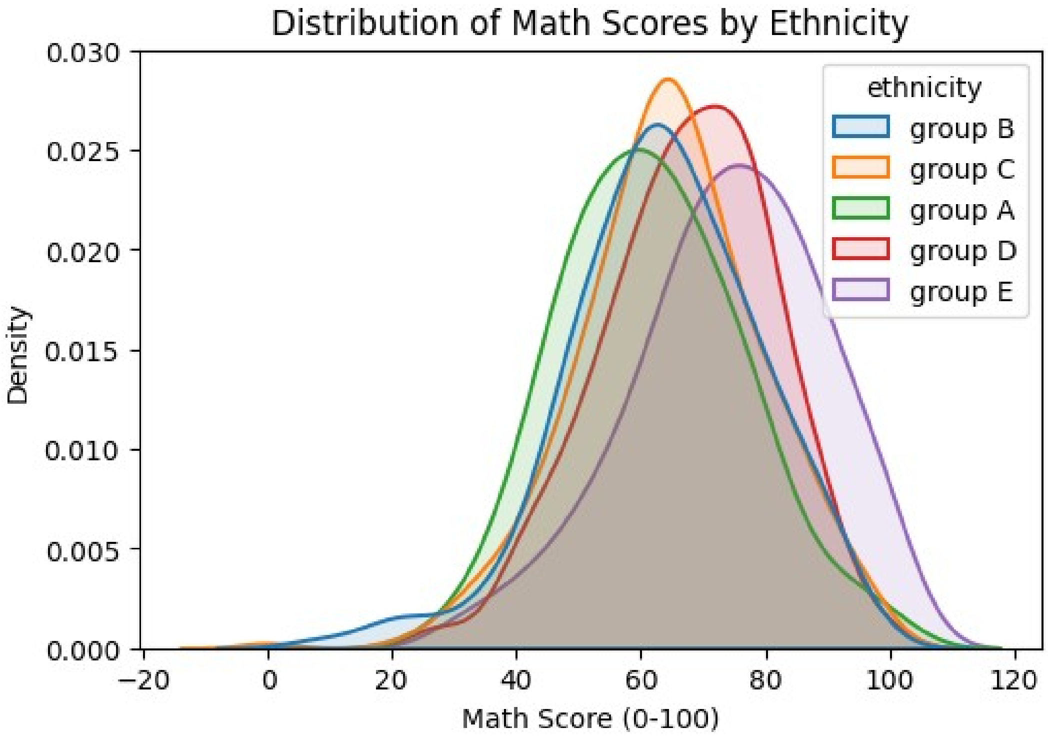
```
1          69              90              88
2          90              95              93

# How to visualize math_score distribution across gender
plt.figure(figsize=(6,4))
sns.histplot(data=df, x='math_score', hue='gender',
             palette={'female':'hotpink', 'male':'blue'},
             kde=True)
plt.show()
```



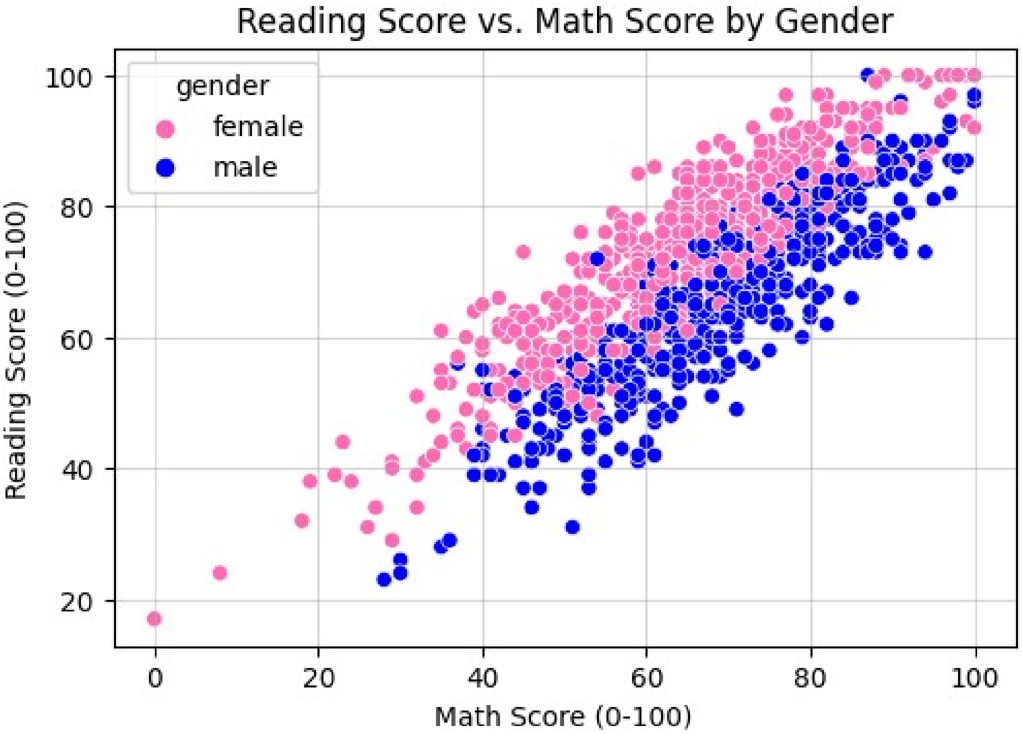
```
plt.figure(figsize=(6,4))
sns.histplot(data=df, x='math_score', hue='ethnicity',
             kde=True)

plt.title('Distribution of Math Scores by Ethnicity')
plt.xlabel('Math Score (0-100)')
plt.ylabel('Count')

plt.show()
```



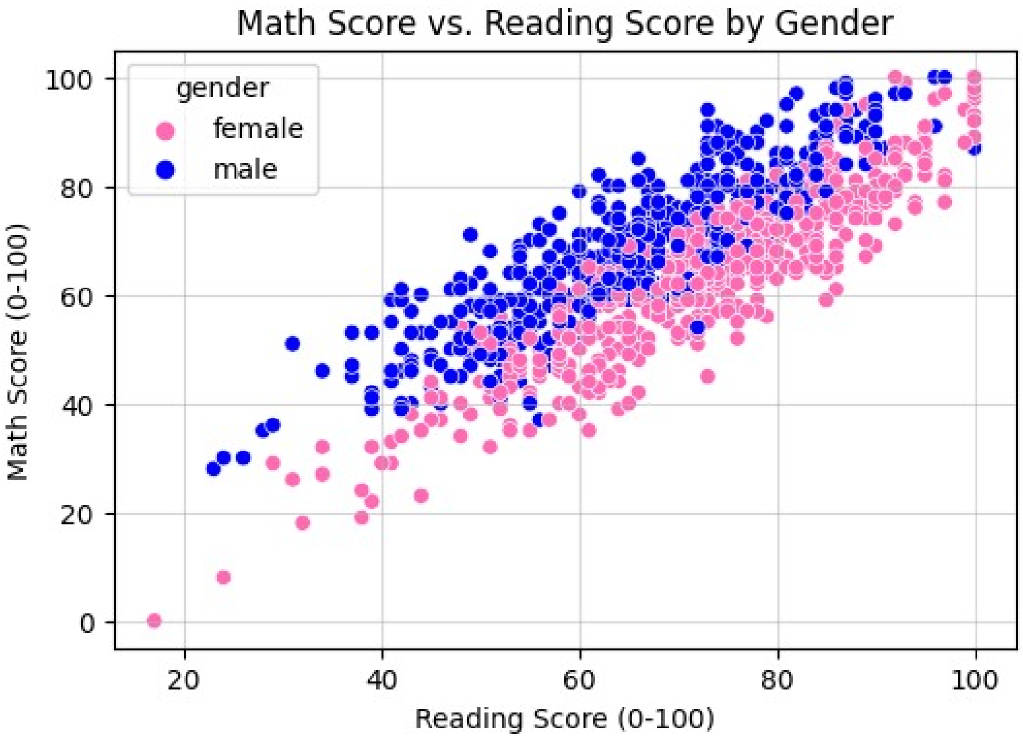
```
plt.figure(figsize=(6,4))
sns.kdeplot(data=df, x='math_score', hue='ethnicity',
            fill=True, common_norm=False,
            alpha=0.15,
            linewidth=1.5)

plt.title('Distribution of Math Scores by Ethnicity')
plt.xlabel('Math Score (0-100)')
plt.ylabel('Density')

plt.show()
```



```
plt.figure(figsize=(6,4))
sns.scatterplot(df, x='math_score', y='reading_score',
                hue='gender',
palette={'male':'blue','female':'hotpink'})
plt.grid(True, alpha=0.5)

plt.title('Reading Score vs. Math Score by Gender')
plt.xlabel('Math Score (0-100)')
plt.ylabel('Reading Score (0-100)')

plt.show()
```



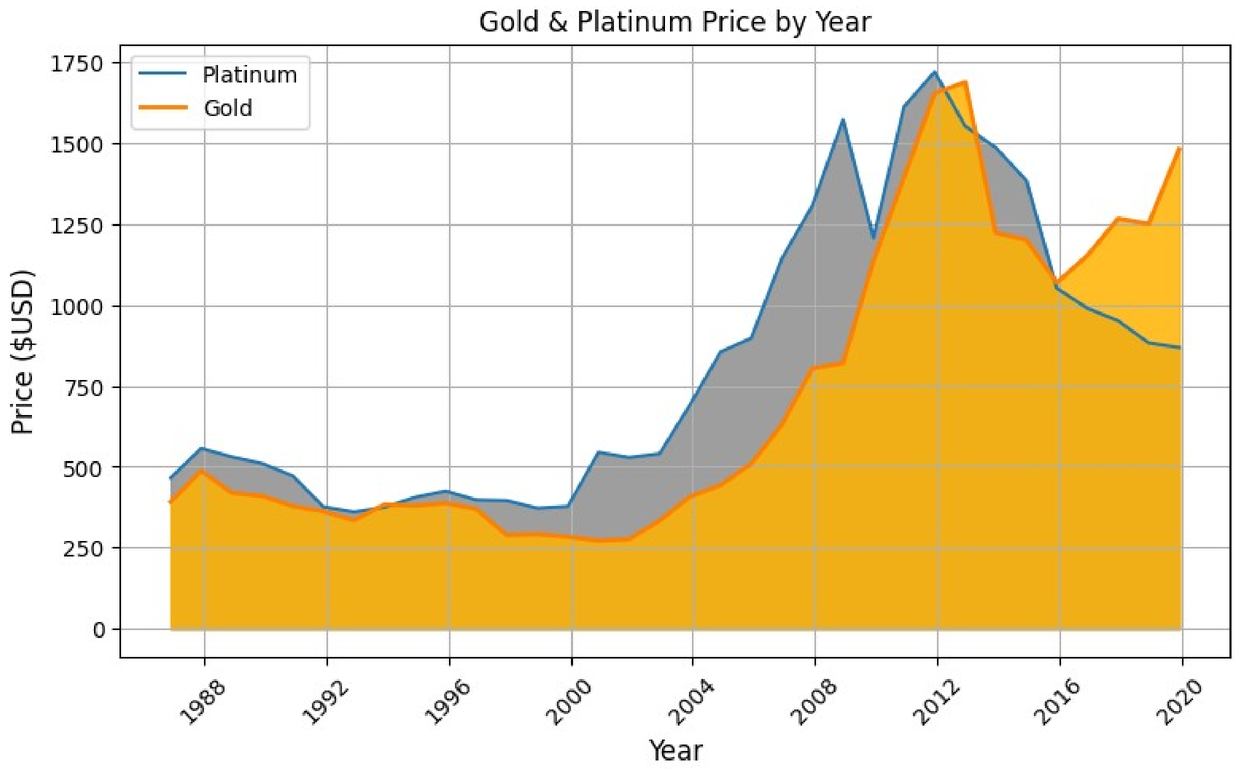
```
plt.figure(figsize=(6,4))
sns.scatterplot(df, x='reading_score', y='math_score',
                hue='gender',
palette={'male':'blue','female':'hotpink'})
plt.grid(True, alpha=0.5)

plt.title('Math Score vs. Reading Score by Gender')
plt.xlabel('Reading Score (0-100)')
plt.ylabel('Math Score (0-100)')

plt.show()
```



```
df = pd.read_csv('annual_gold.csv')
df.tail(3)

        Date      Gold  Platinum
31  2017-12  1265.674    950.49
32  2018-12  1249.887    882.18
33  2019-12  1480.025    868.04

# We change the data type of Date from object to datetime
df['Date'] = pd.to_datetime(df['Date'])
df.head(1)

         Date     Gold  Platinum
0 1986-12-01  391.595    465.29

df['Year'] = df['Date'].dt.year
df.head(1)

         Date     Gold  Platinum  Year
0 1986-12-01  391.595    465.29  1986

plt.figure(figsize=(9,5))

#----------------------------------------------------------------------
----------

# Platinum
sns.lineplot(data=df, x='Date', y='Platinum', label='Platinum')
```

```
plt.fill_between(x=df['Date'], y1=df['Platinum'], color='#999999',
alpha=0.95)

#---------------------------------------------------------------------
----------

# Gold
plt.plot_date(x=df['Date'], y=df['Gold'], linestyle='-',
             color='#ff8000', linewidth=2, marker=',',
              label='Gold')
plt.fill_between(x=df['Date'], y1=df['Gold'], color='#ffb300',
alpha=0.85)

#---------------------------------------------------------------------
----------

plt.xticks(rotation=45)
plt.xlabel('Year', size=12)
plt.ylabel('Price ($USD)', size=12)
plt.title('Gold & Platinum Price by Year')

plt.legend(loc = 'upper left')
plt.grid(True)
plt.show()
```

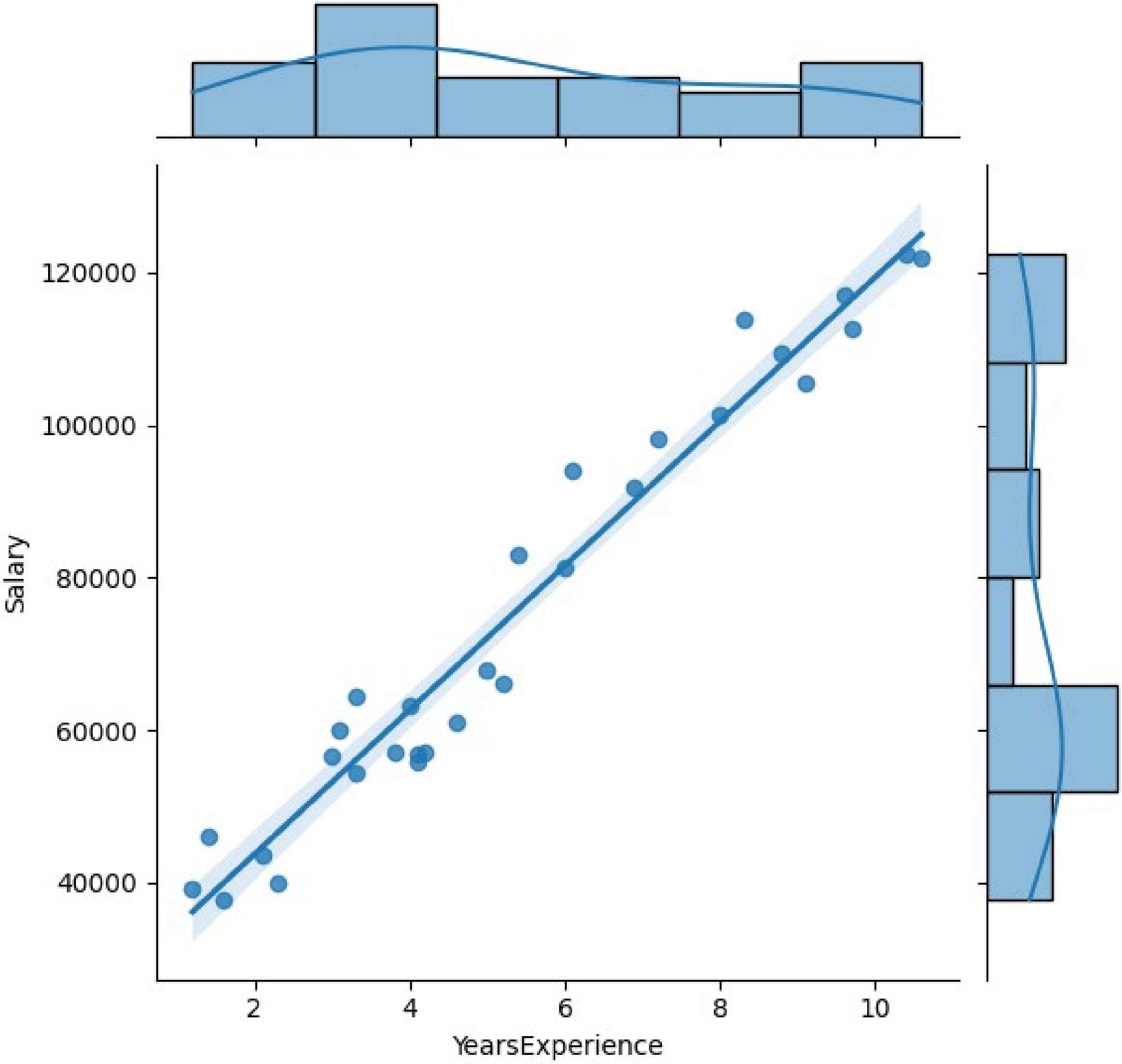
```
<ipython-input-50-7d3b479df9ae>:12: UserWarning: marker is redundantly
defined by the 'marker' keyword argument and the fmt string "o" (->
marker='o'). The keyword argument will take precedence.
  plt.plot_date(x=df['Date'], y=df['Gold'], linestyle='-',
```



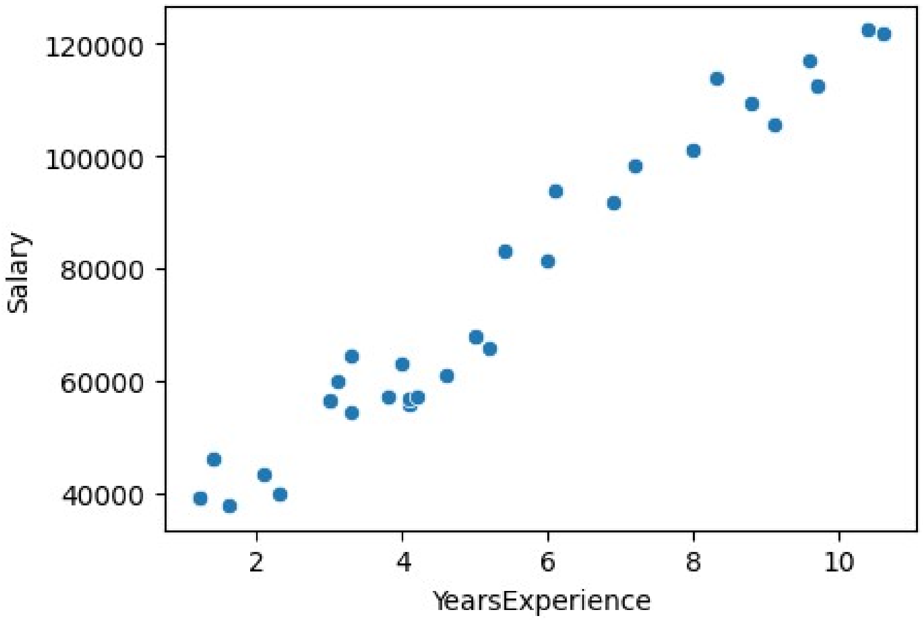
```
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
```

Building a Regression Model

```
df = pd.read_csv('salary.csv')
df.head(3)

   YearsExperience  Salary
0              1.2   39344
1              1.4   46206
2              1.6   37732

# Creating a jointplot to explore the relationship between experience
and salary

plt.figure(figsize=(6,6))
sns.jointplot(data=df, x='YearsExperience', y='Salary', kind='reg')
plt.show()

<Figure size 600x600 with 0 Axes>
```

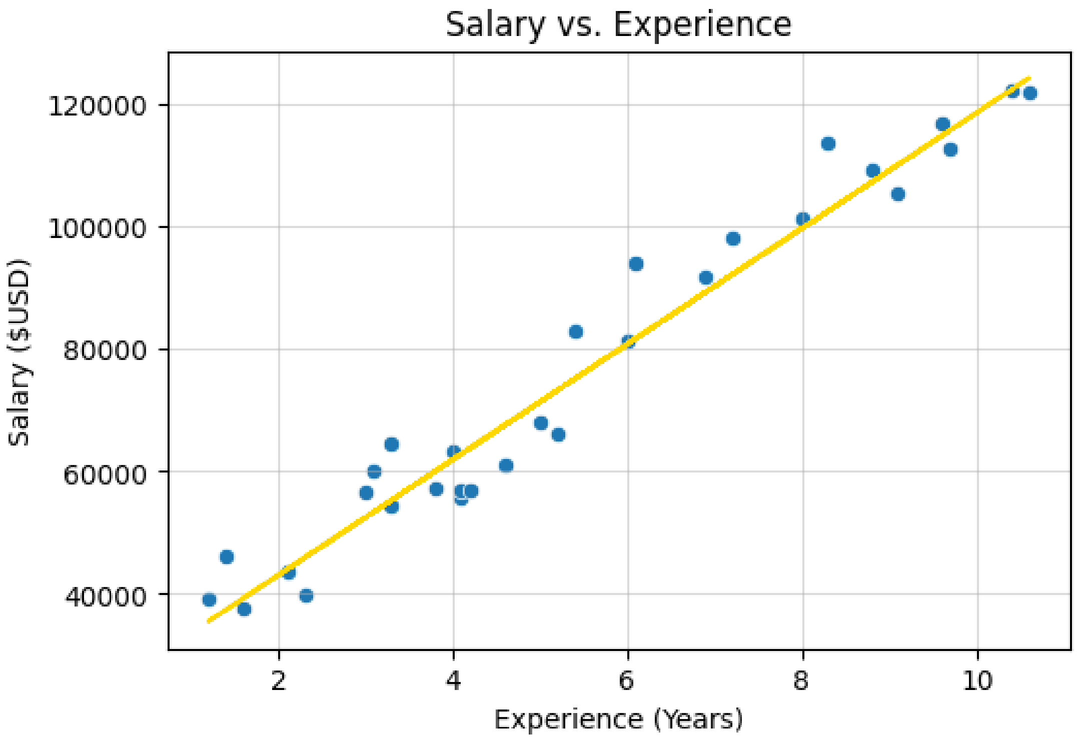


```python
# Displaying the relationship between x and y using scatterplot
plt.figure(figsize=(5,3.5))
sns.scatterplot(data=df, x='YearsExperience', y='Salary')
plt.show()
```



```
# Model building

# Double brackets creates a pandas dataframe
x = df[['YearsExperience']]
y = df[['Salary']]

# Split the data
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.1)

regr = linear_model.LinearRegression()
regr.fit(x_train, y_train)

LinearRegression()

# Model prediction
y_pred = regr.predict(x_test)
y_pred

array([[ 99711.56993733],
       [ 37375.87620897],
       [102545.01056134]])

np.sqrt(mean_squared_error(y_test, y_pred))

8316.075232121839

r2_score(y_test, y_pred)

0.9198218331733488
```

```
# Visualing the model

plt.figure(figsize=(6,4))

sns.scatterplot(data=df, x='YearsExperience', y='Salary')
plt.plot(x_train, regr.predict(x_train), color='gold')

plt.title('Salary vs. Experience')
plt.xlabel('Experience (Years)')
plt.ylabel('Salary ($USD)')

plt.grid(True, alpha=0.45)
plt.show()
```



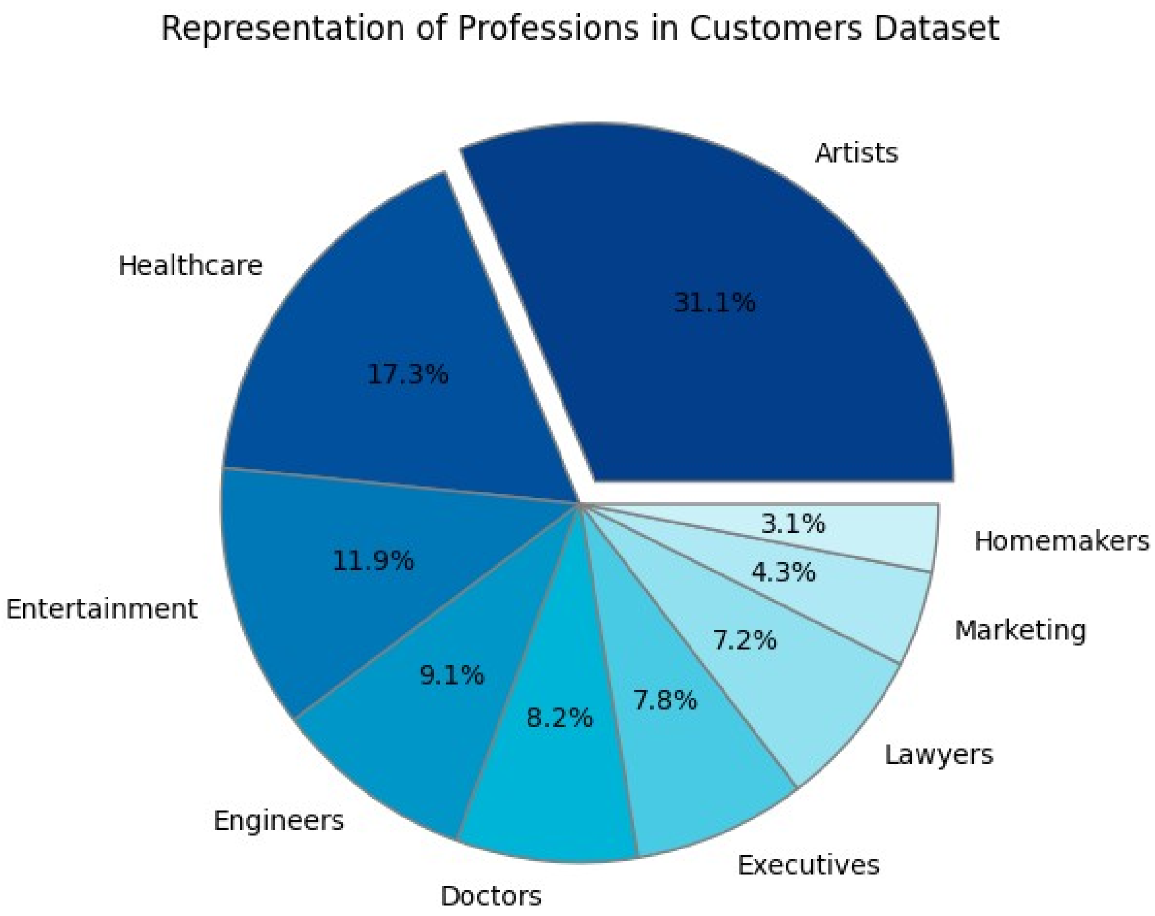
```
print('slope:', regr.coef_)
print('intercept:', regr.intercept_)

slope: [[9444.80208005]]
intercept: [24153.1532969]

# Model:
#    y = 24153 + 9445x

#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
```

The following selection of visualizations appear in a final consumer analysis project for a data visualization class I completed at CSU Northridge. For the sake of brevity, I did not transfer over the lengthy analysis portion, nor all the visualizations, although the project in its entirety can be provided if needed.

```
df = pd.read_csv('Customers.csv')
df.head(3)
```

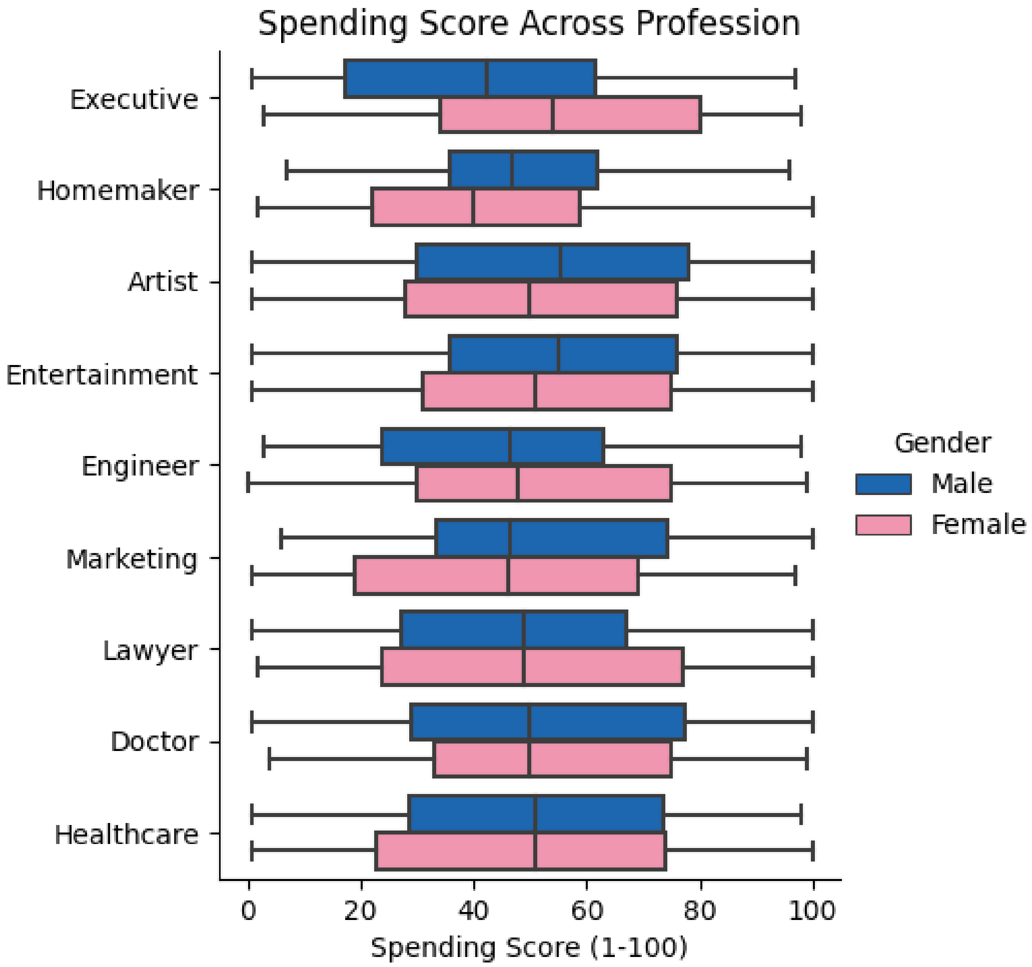
```
   CustomerID  Gender  Age  Annual Income ($)  Spending Score (1-100)
\
0           1    Male   19              15000                      39

1           2    Male   21              35000                      81

2           3  Female   20              86000                       6


   Profession  Work Experience  Family Size
0  Healthcare                1            4
1    Engineer                3            3
2    Engineer                1            1
```

```
profession = df['Profession'].value_counts().tolist()
plabels = ['Artists', 'Healthcare', 'Entertainment', 'Engineers',
'Doctors', 'Executives', 'Lawyers', 'Marketing', 'Homemakers']
colors = ['#023e8a', '#00509d', '#0077b6', '#0096c7', '#00b4d8',
'#48cae4', '#90e0ef', '#ade8f4', '#caf0f8']
explode = (0.075, 0, 0, 0, 0, 0, 0, 0, 0)

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

plt.pie(profession, labels=plabels, colors=colors, autopct='%1.1f%%',
explode=explode, wedgeprops={'linewidth':1,
'edgecolor':'grey'})

plt.title('Representation of Professions in Customers Dataset')
plt.show()
```

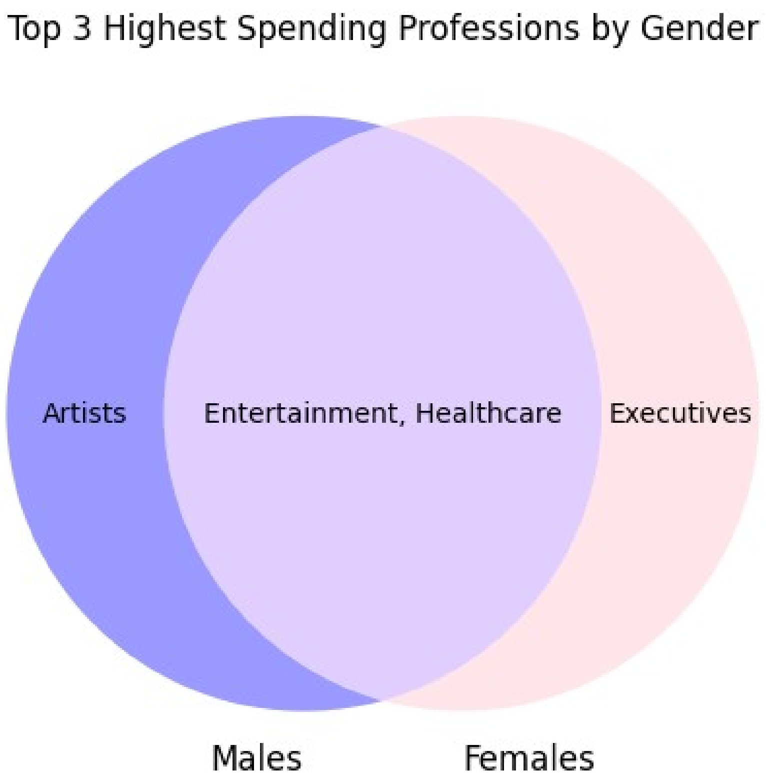


```
plt.figure(figsize=(6,4))
profession_order = ['Executive', 'Homemaker', 'Artist',
'Entertainment', 'Engineer', 'Marketing', 'Lawyer', 'Doctor',
'Healthcare']

sns.catplot(data=df, x='Spending Score (1-100)', y='Profession',
kind='box', hue='Gender',
            palette={'Female':'#ff87ab', 'Male':'#0466c8'},
orient='h', order=profession_order)

plt.ylabel('')
plt.title('Spending Score Across Profession')
plt.show()
```

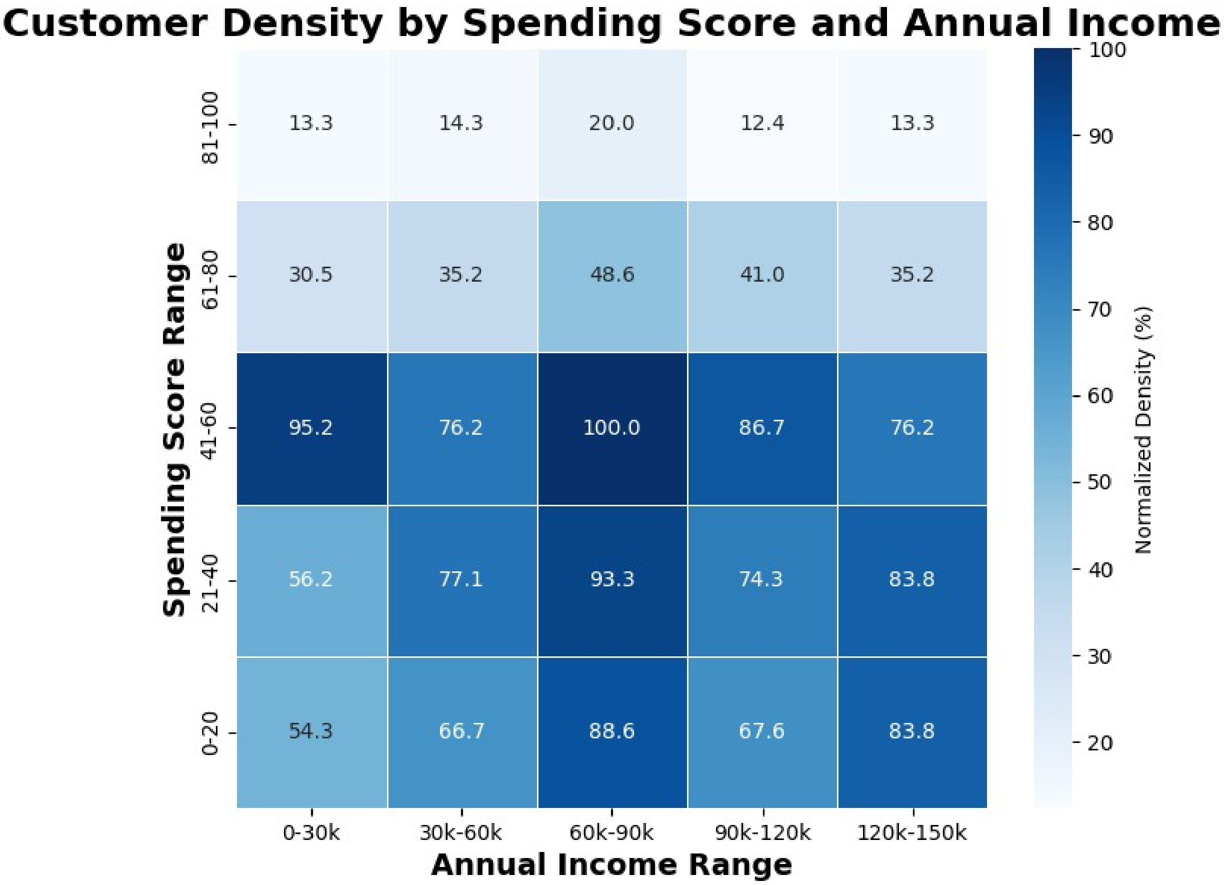
```
<Figure size 600x400 with 0 Axes>
```



```
a = set(['Artists', 'Healthcare', 'Entertainment'])
b = set(['Executives', 'Healthcare', 'Entertainment'])

ven = venn2(subsets=(a,b), set_labels=('Males', 'Females'),
      set_colors=('blue', 'pink'))

ven.get_label_by_id('10').set_text(', '.join(a-b))
ven.get_label_by_id('11').set_text(', '.join(a&b))
ven.get_label_by_id('01').set_text(', '.join(b-a))

plt.title('Top 3 Highest Spending Professions by Gender')
plt.show()
```



```
# Define the specific bins for income and spending score
income_bins = [0, 30000, 60000, 90000, 120000, 150000]
spending_bins = [0, 20, 40, 60, 80, 100]

# Create the 2D histogram with swapped axes
hist_swapped, spending_edges, income_edges =
np.histogram2d( df['Annual Income ($)'], df['Spending Score (1-100)'],
bins=[income_bins, spending_bins])

# Normalize the swapped histogram
normalized_hist_swapped = (hist_swapped / hist_swapped.max()) * 100

# Create the figure and axis for the heatmap with swapped axes
plt.figure(figsize=(1,1))
fig, ax = plt.subplots(figsize=(8,6.5))

# Create the heatmap with swapped axes
sns.heatmap(normalized_hist_swapped,
            xticklabels=['0-30k', '30k-60k', '60k-90k', '90k-120k',
'120k-150k'],
            yticklabels=['81-100', '61-80', '41-60', '21-40', '0-20'],
            cmap='Blues',
            ax=ax,
            annot=True,
            fmt=".1f",
            linewidths=.5,
            cbar_kws={'label': 'Normalized Density (%)'})

# Add titles and labels with axes swapped
plt.title('Customer Density by Spending Score and Annual Income',
fontsize=18, fontweight='bold')
plt.xlabel('Annual Income Range', fontsize=14, fontweight='bold')
plt.ylabel('Spending Score Range', fontsize=14, fontweight='bold')
plt.show()

<Figure size 100x100 with 0 Axes>
```


Customer Density by Spending Score and Annual Income
Spending Score Range
0-30k
30k-60k
60k-90k
90k-120k
120k-150k
Annual Income Range
Normalized Density (%)