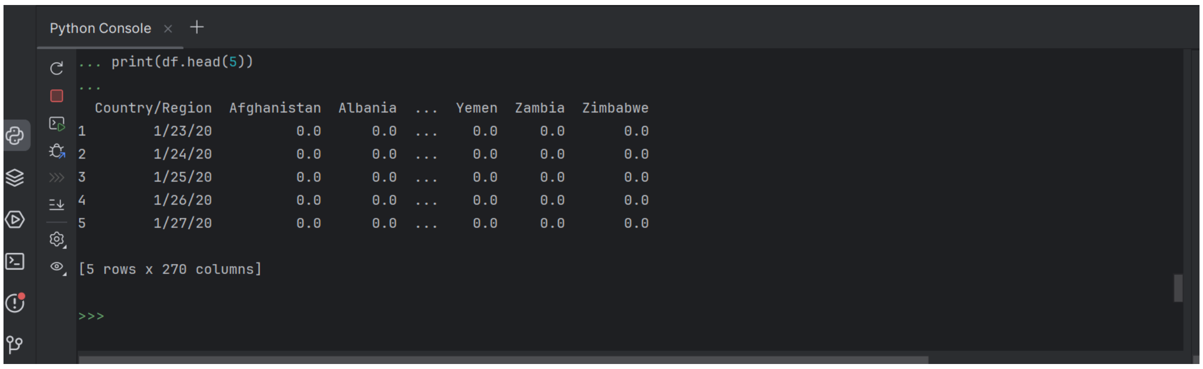
Section 12 - Aaron Yang - Homework2

S1 - Question 1

Code

```
#%%
# Load packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import seaborn as sns


#%%
# Question 1
# Load data from Github
url = 'https://raw.githubusercontent.com/rjafari979/Information-Visualization-
Data-Analytics-Dataset-/main/CONVENIENT_global_confirmed_cases.csv'

df = pd.read_csv(url)

# Remove the 'nan' and missing data
df = df.dropna()

print(df.head(5))
```

Answer



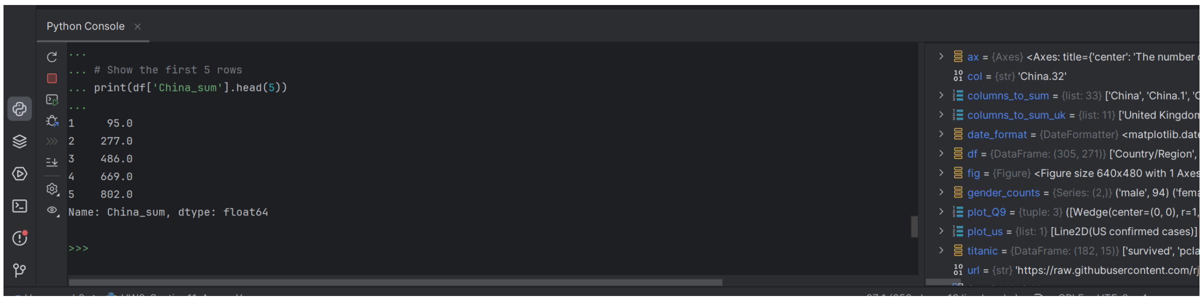
```
... print(df.head(5))
...
  Country/Region  Afghanistan  Albania  ...  Yemen  Zambia  Zimbabwe
1        1/23/20          0.0      0.0  ...    0.0     0.0       0.0
2        1/24/20          0.0      0.0  ...    0.0     0.0       0.0
3        1/25/20          0.0      0.0  ...    0.0     0.0       0.0
4        1/26/20          0.0      0.0  ...    0.0     0.0       0.0
5        1/27/20          0.0      0.0  ...    0.0     0.0       0.0

[5 rows x 270 columns]

>>>
```

S1 - Question 2

Code

```
#%%
# Question 2
# Create a list to store the columns name
columns_to_sum = ['China'] + ['China.'+str(i) for i in range(1,33)]

# Transfer all columns to numerical data
for col in columns_to_sum:
    df[col] = pd.to_numeric(df[col])

# Sum those columns
df['China_sum'] = df[columns_to_sum].sum(axis=1)

# Show the first 5 rows
print(df['China_sum'].head(5))
```

Answer


Python Console

```
...
... # Show the first 5 rows
... print(df['China_sum'].head(5))
...
1     95.0
2    277.0
3    486.0
4    669.0
5    802.0
Name: China_sum, dtype: float64

>>>
```

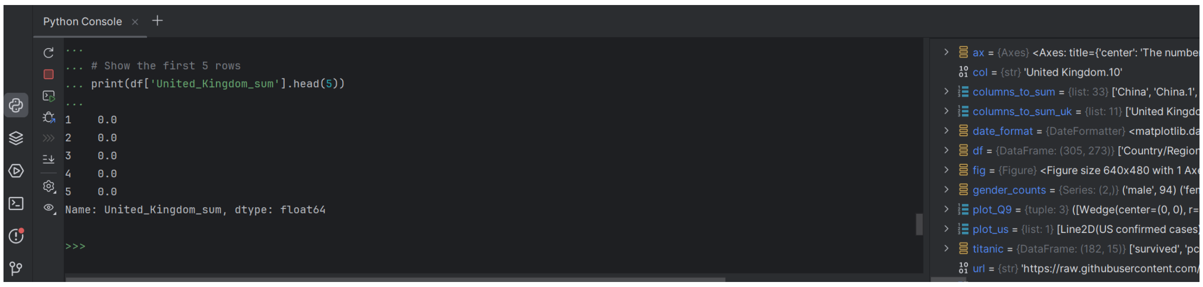
```
ax = {Axes} <Axes: title={'center': 'The number
col = {str} 'China.32'
columns_to_sum = {list: 33} ['China', 'China.1', 'C
columns_to_sum_uk = {list: 11} ['United Kingdom
date_format = {DateFormatter} <matplotlib.date
df = {DataFrame: (305, 271)} ['Country/Region',
fig = {Figure} <Figure size 640x480 with 1 Axes
gender_counts = {Series: (2,)} ('male', 94) ('fema
plot_Q9 = {tuple: 3} ([Wedge(center=(0, 0), r=1,
plot_us = {list: 1} [Line2D(US confirmed cases)]
titanic = {DataFrame: (182, 15)} ['survived', 'pcla
url = {str} 'https://raw.githubusercontent.com/rj
```

S1 - Question 3

Code

```
#%%
# Question 3
# Create the columns name list
columns_to_sum_uk = ['United Kingdom'] + ['United Kingdom.'+str(i) for i in
range(1,11)]

# Transfer all str data to numerical data
for col in columns_to_sum_uk:
    df[col] = pd.to_numeric(df[col])

# To sum all columns into one
df['United_Kingdom_sum'] = df[columns_to_sum_uk].sum(axis=1)

# Show the first 5 rows
print(df['United_Kingdom_sum'].head(5))
```

Answer


Python Console

```
...
... # Show the first 5 rows
... print(df['United_Kingdom_sum'].head(5))
...
1    0.0
2    0.0
3    0.0
4    0.0
5    0.0
Name: United_Kingdom_sum, dtype: float64

>>>
```

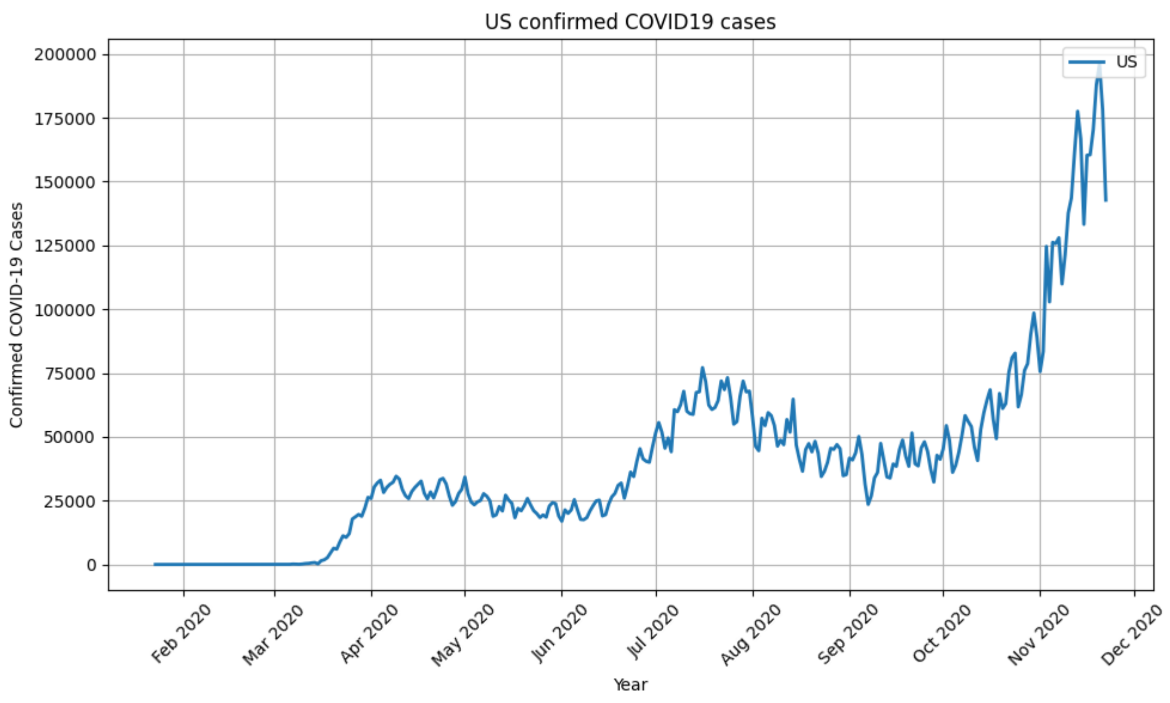
```
ax = {Axes} <Axes: title={'center': 'The number
col = {str} 'United Kingdom.10'
columns_to_sum = {list: 33} ['China', 'China.1',
columns_to_sum_uk = {list: 11} ['United Kingdo
date_format = {DateFormatter} <matplotlib.da
df = {DataFrame: (305, 273)} ['Country/Region
fig = {Figure} <Figure size 640x480 with 1 Axe
gender_counts = {Series: (2,)} ('male', 94) ('fen
plot_Q9 = {tuple: 3} ([Wedge(center=(0, 0), r=
plot_us = {list: 1} [Line2D(US confirmed cases)
titanic = {DataFrame: (182, 15)} ['survived', 'pc
url = {str} 'https://raw.githubusercontent.com/
```

S1 - Question 4

Code

```
df = df.rename(columns={'Country/Region': 'Date'})
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)

us_columns = [col for col in df.columns if 'US' in col]
df['US_sum'] = df[us_columns].sum(axis=1)

plt.figure(figsize=(10, 6))
plt.plot(df.index, df['US_sum'], label='US', linewidth=2)
plt.gca().xaxis.set_major_locator(mdates.MonthLocator())
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
plt.xlabel('Year')
plt.ylabel('Confirmed COVID-19 Cases')
plt.title('US confirmed COVID19 cases')
plt.legend(loc='upper right')
plt.grid(True)
plt.xticks(rotation=45)


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

Answer

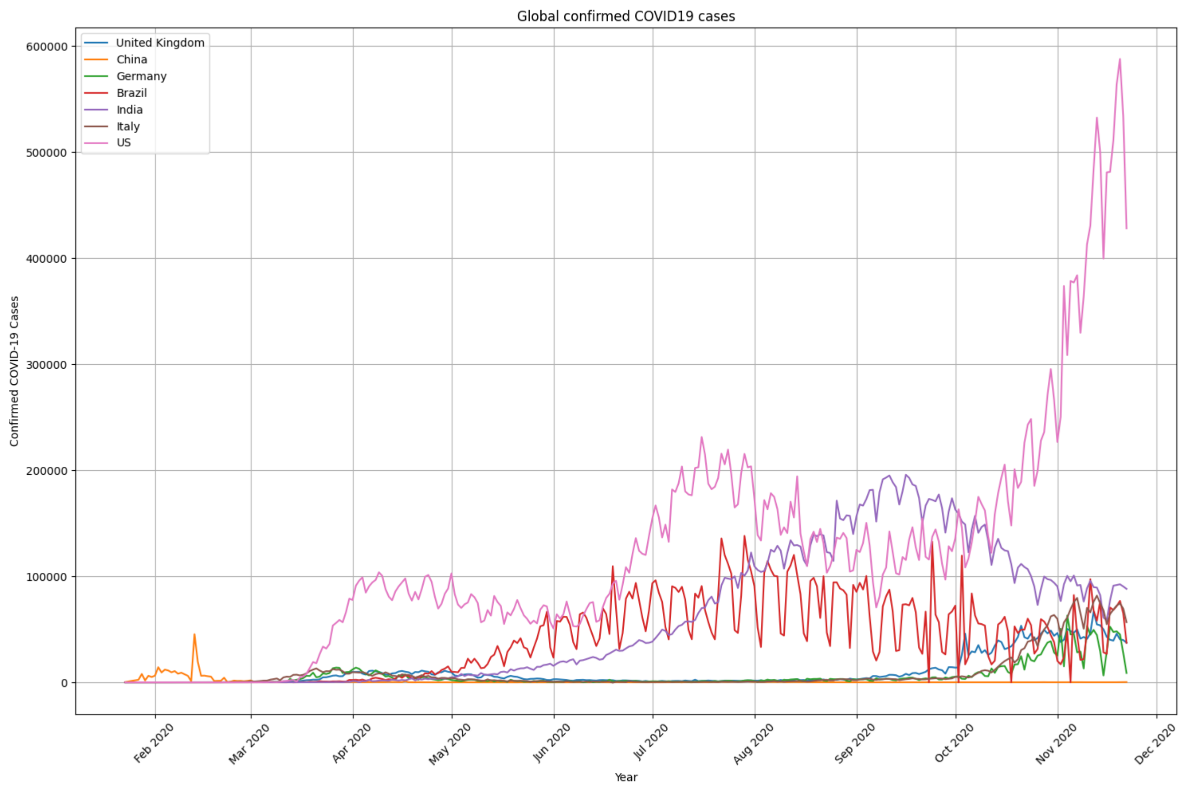


S1 - Question 5

Code

```
# Question 5
def sum_cases_by_country(df, country_name):
    country_columns = [col for col in df.columns if country_name in col]
    return df[country_columns].sum(axis=1, numeric_only=True)



countries = ['United Kingdom', 'China', 'Germany', 'Brazil', 'India', 'Italy']
for country in countries:
    df[country + '_sum'] = sum_cases_by_country(df, country)
df['US_sum'] = sum_cases_by_country(df, 'US')
plt.figure(figsize=(15, 10))
for country in countries + ['US']:
    plt.plot(df.index, df[country + '_sum'], label=country)
plt.gca().xaxis.set_major_locator(mdates.MonthLocator())
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
plt.xlabel('Year')
plt.ylabel('Confirmed COVID-19 Cases')
plt.title('Global confirmed COVID19 cases')
plt.grid(True)
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Answer

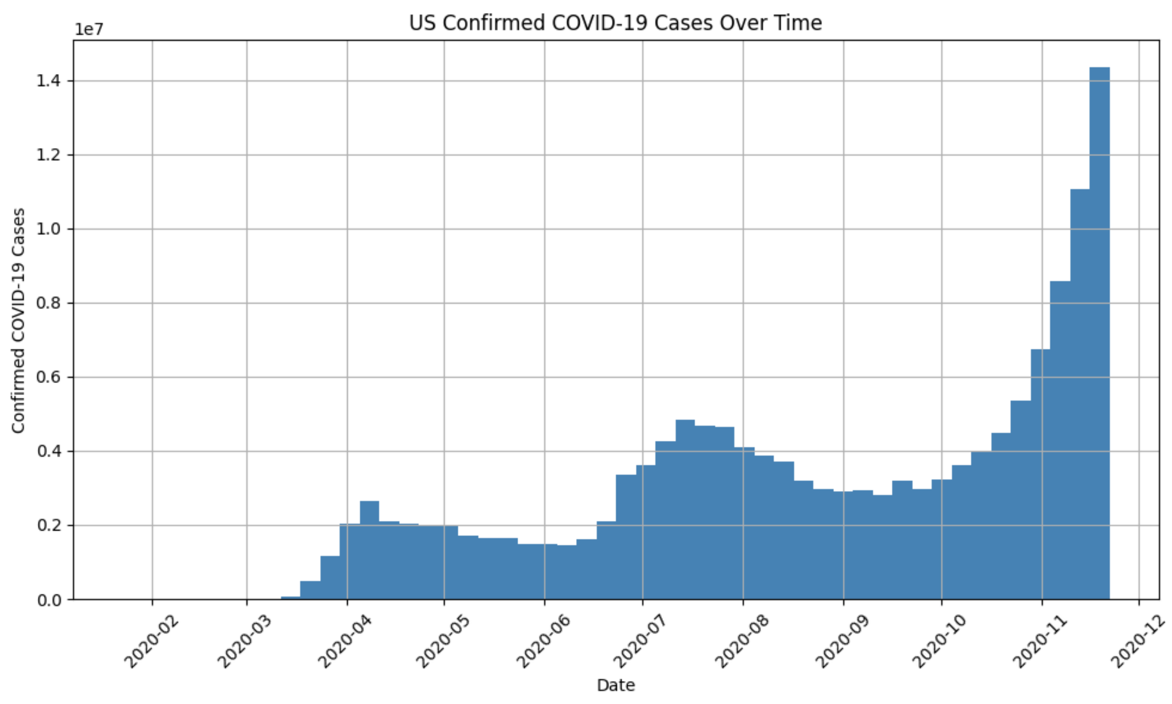


S1 - Question 6

Code

```
# Question 6
us_columns = [col for col in df.columns if 'US' in col]
df['US_sum'] = df[us_columns].sum(axis=1)



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

plt.hist(df.index, bins=50, weights=df['US_sum'], color='steelblue', rwidth=1,
label='US')
plt.xlabel('Date')
plt.ylabel('Confirmed COVID-19 Cases')
plt.title('US Confirmed COVID-19 Cases Over Time')
plt.xticks(rotation=45)
plt.grid(True)



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

Answer

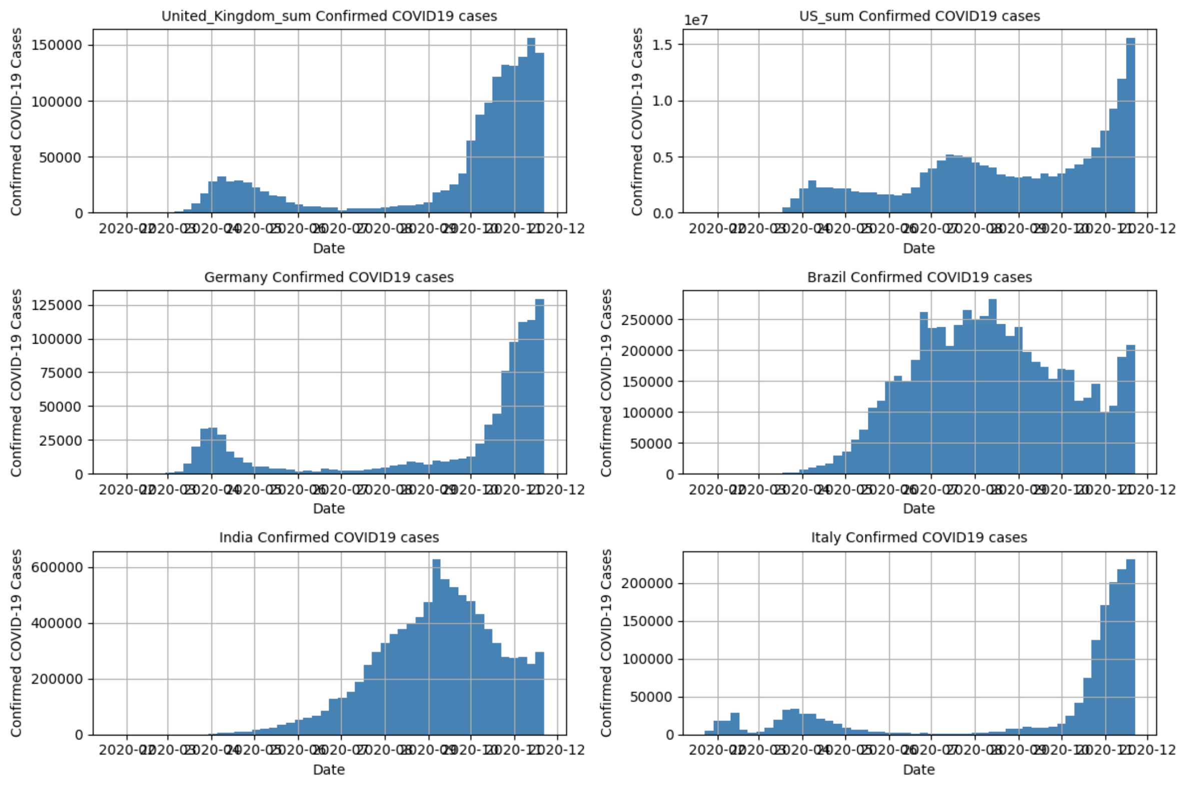


S1 - Question 7

Code

```
#%%
# Question 7
country_list = ('China_sum', 'United_Kingdom_sum', 'US_sum', 'Germany', 'Brazil',
'India', 'Italy')

# Create 3*2 matrix and also flatten plots
fig, axes = plt.subplots(nrows=3 , ncols=2 , figsize=(12, 8 ))
axes = axes.flatten()

# Set a loop to get subplots
for i, country in enumerate(country_list):
    ax = axes[i-1]
    if country in df.columns:
        # Create a histogram
        ax.hist(df.index, bins=50, weights=df[country], color = 'Steelblue',
rwidth=1)
        # Set the title
        ax.set_title(country+ ' Confirmed COVID19 cases', fontsize=10)
        ax.set_xlabel('Date')
        ax.set_ylabel('Confirmed COVID-19 Cases')
        ax.grid(True)



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

Answer



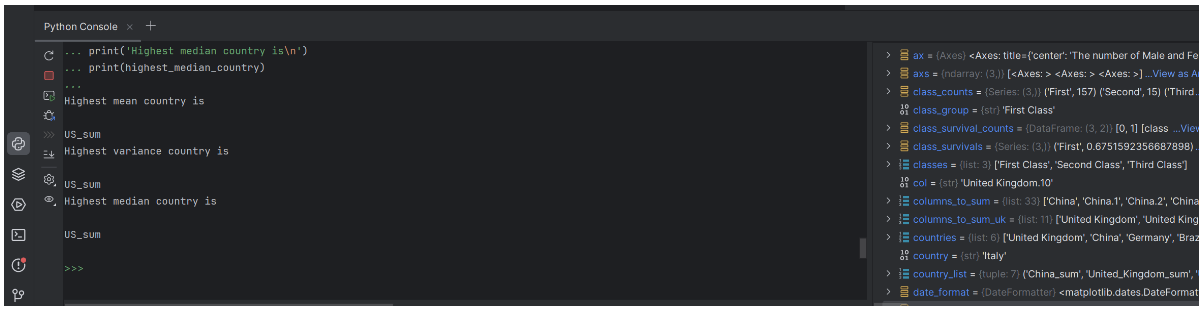
S1 - Question 8

Code

```
#%%
# Question 8
country_list = ('China_sum', 'United_Kingdom_sum', 'US_sum', 'Germany', 'Brazil',
'India', 'Italy')

# Initialize a dictionary to store the statistics
stats = {'mean': {}, 'variance': {}, 'median': {}}

# Calculate the statistics for each country and store them in the dictionary
for country in country_list:
    stats['mean'][country] = df[country].mean()
    stats['variance'][country] = df[country].var()
    stats['median'][country] = df[country].median()

# Convert the dictionary to a DataFrame for easier handling
stats_df = pd.DataFrame(stats)

# Now find the country with the highest mean, variance, and median
highest_mean_country = stats_df['mean'].idxmax()
highest_variance_country = stats_df['variance'].idxmax()
highest_median_country = stats_df['median'].idxmax()

print('Highest mean country is\n')
print(highest_mean_country)
print('Highest variance country is\n')
print(highest_variance_country)
print('Highest median country is\n')
print(highest_median_country)
```

Answer

The highest mean, variance, and median country are all US.



```
... print('Highest median country is\n')
... print(highest_median_country)
...
Highest mean country is

US_sum
Highest variance country is

US_sum
Highest median country is

US_sum

>>>
```

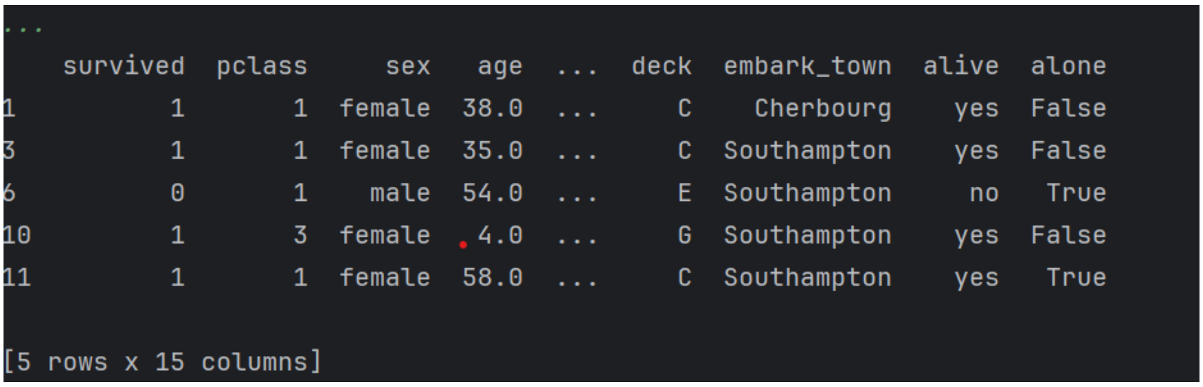
```
ax = {Axes} <Axes: title={'center': 'The number of Male and Fe
axs = {ndarray: (3,)} [<Axes: > <Axes: > <Axes: >] ...View as A
class_counts = {Series: (3,)} ('First', 157) ('Second', 15) ('Third ...
class_group = {str} 'First Class'
class_survival_counts = {DataFrame: (3, 2)} [0, 1] [class ...View
class_survivals = {Series: (3,)} ('First', 0.6751592356687898) ...
classes = {list: 3} ['First Class', 'Second Class', 'Third Class']
col = {str} 'United Kingdom.10'
columns_to_sum = {list: 33} ['China', 'China.1', 'China.2', 'China
columns_to_sum_uk = {list: 11} ['United Kingdom', 'United King
countries = {list: 6} ['United Kingdom', 'China', 'Germany', 'Braz
country = {str} 'Italy'
country_list = {tuple: 7} ('China_sum', 'United_Kingdom_sum', '
date_format = {DateFormatter} <matplotlib.dates.DateFormat
```

S2 - Question 1

Code

```
#%%
# Question 7
# Load the dataset
titanic = pd.DataFrame(data=sns.load_dataset('titanic'))

# Clean the na values
titanic = titanic.dropna()

# Print the first 5 row
print(titanic.head(5))
```

Answer



```
...
    survived  pclass     sex   age  ...  deck  embark_town  alive  alone
1          1       1  female  38.0  ...     C    Cherbourg    yes  False
3          1       1  female  35.0  ...     C  Southampton    yes  False
6          0       1    male  54.0  ...     E  Southampton     no   True
10         1       3  female   4.0  ...     G  Southampton    yes  False
11         1       1  female  58.0  ...     C  Southampton    yes   True

[5 rows x 15 columns]
```

S2 - Question 2

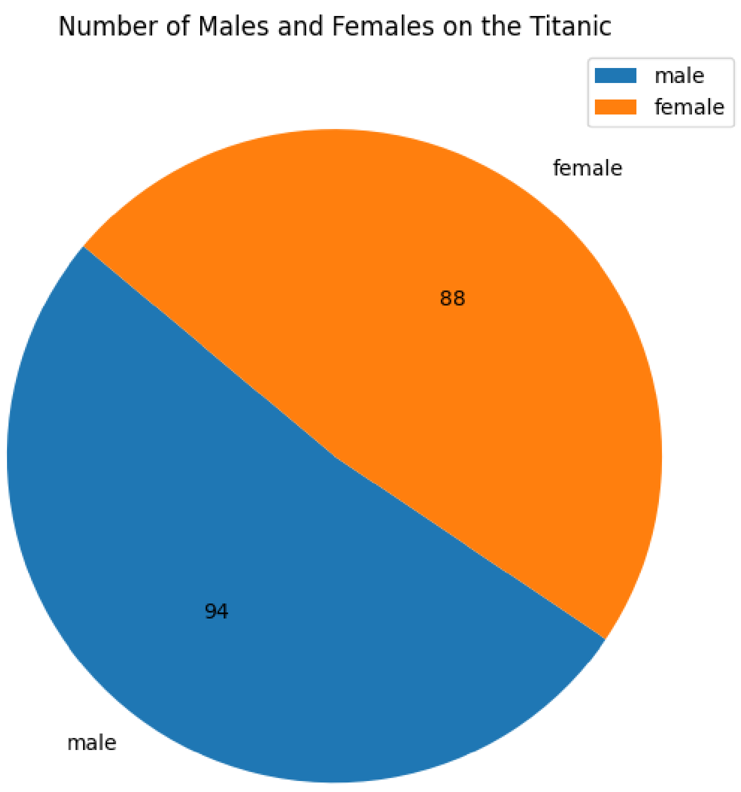
Code

```
# Question 8
# Set a variable to store the number of male and female
gender_counts = titanic['sex'].value_counts()

# Create a function to format the absolute value on the pie chart
def absolute_value(val):
    a = np.round(val/100.*gender_counts.sum(), 0)
    return int(a)


plt.figure(figsize=(8,6))
plot_Q8 = plt.pie(gender_counts, labels=gender_counts.index,
autopct=absolute_value, startangle = 140)
plt.title('Number of Males and Females on the Titanic')
plt.legend()


plt.tight_layout()
plt.show()

# Show the total number of Male and Female
```

```
print("The total number of male and female on Titanic:\n")
print(titanic['sex'].value_counts())
```

Answer

```
The total number of male and female on Titanic:


sex
male      94
female    88
Name: count, dtype: int64
```

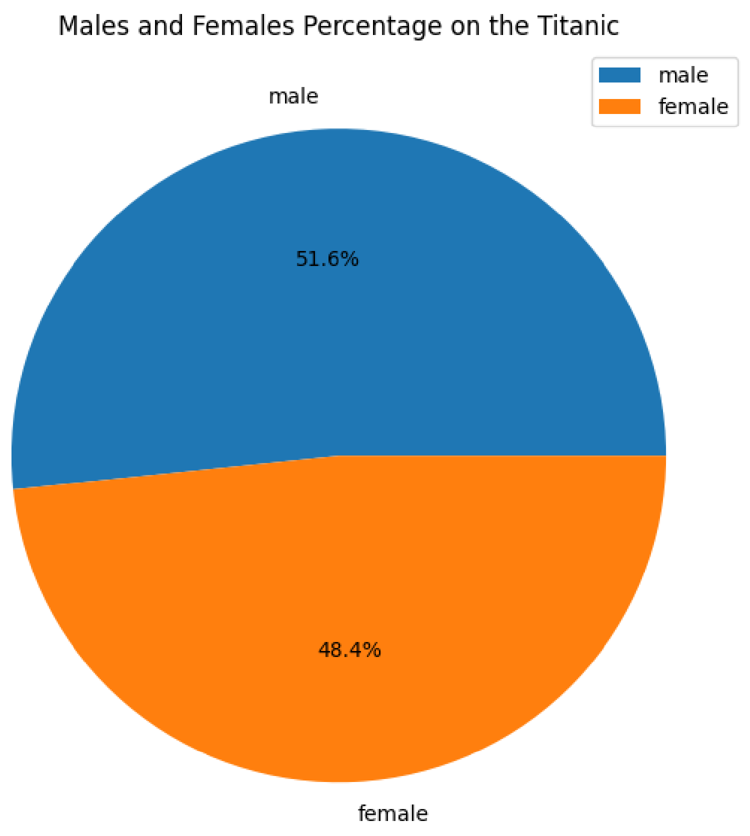


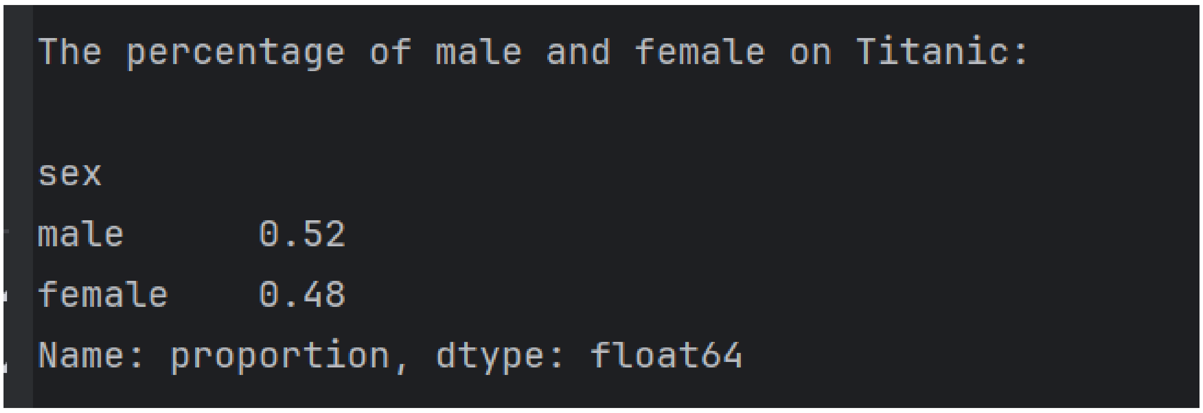
S2 - Question 3

Code

```
#%%
# Question 9
plt.figure(figsize=(8,6))
plot_Q9 = plt.pie(gender_counts,
                  labels=gender_counts.index,
                  autopct='%1.1f%%',
                  )
plt.title('Males and Females Percentage on the Titanic')
plt.legend()
plt.tight_layout()
plt.show()

# Show the percentage of Male and Female
print('The percentage of male and female on Titanic:\n')
print(titanic['sex'].value_counts(normalize=True).round(2))
```

Answer





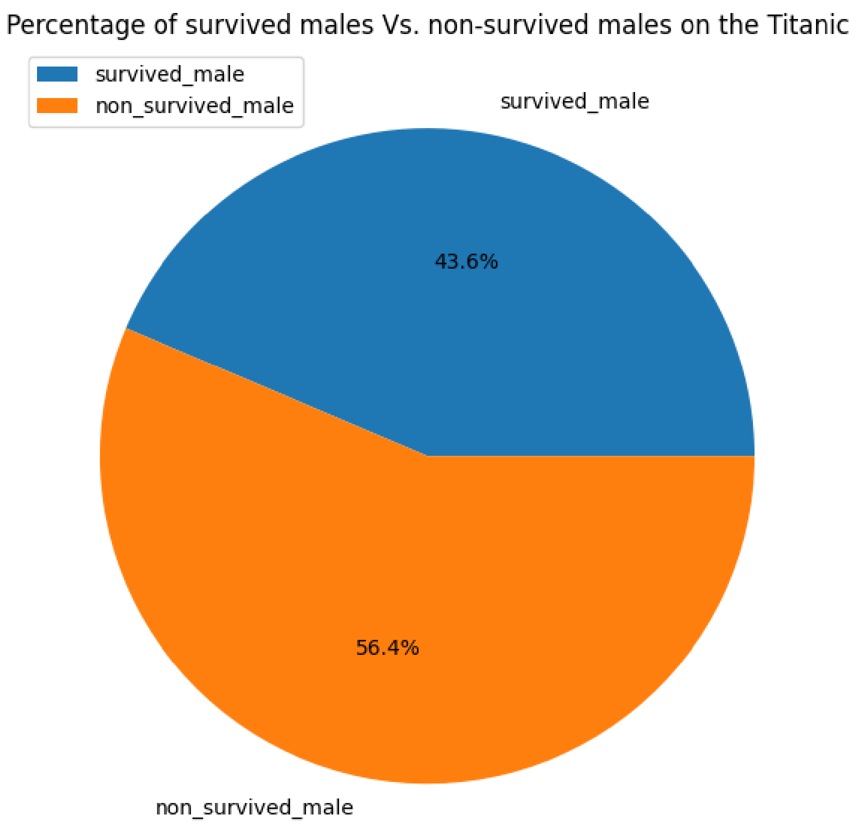
```
The percentage of male and female on Titanic:


sex
male      0.52
female    0.48
Name: proportion, dtype: float64
```

S2 - Question 4

Code

```
#%%
# Question 4
# Calculate the number of male passengers who survived and those who did not
sex_survivals = titanic[titanic['sex'] == 'male'].groupby('survived')
['sex'].count()


labels = ['survived_male', 'non_survived_male']
sizes = [sex_survivals[1] if 1 in sex_survivals else 0, sex_survivals[0] if 0 in
sex_survivals else 0]

plt.figure(figsize=(8,6))
plot_S2_Q4 = plt.pie(sizes, autopct='%1.1f%%',
                     labels=labels)
plt.title('Percentage of survived males Vs. non-survived males on the Titanic')
plt.legend(loc='upper left')


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

Answer

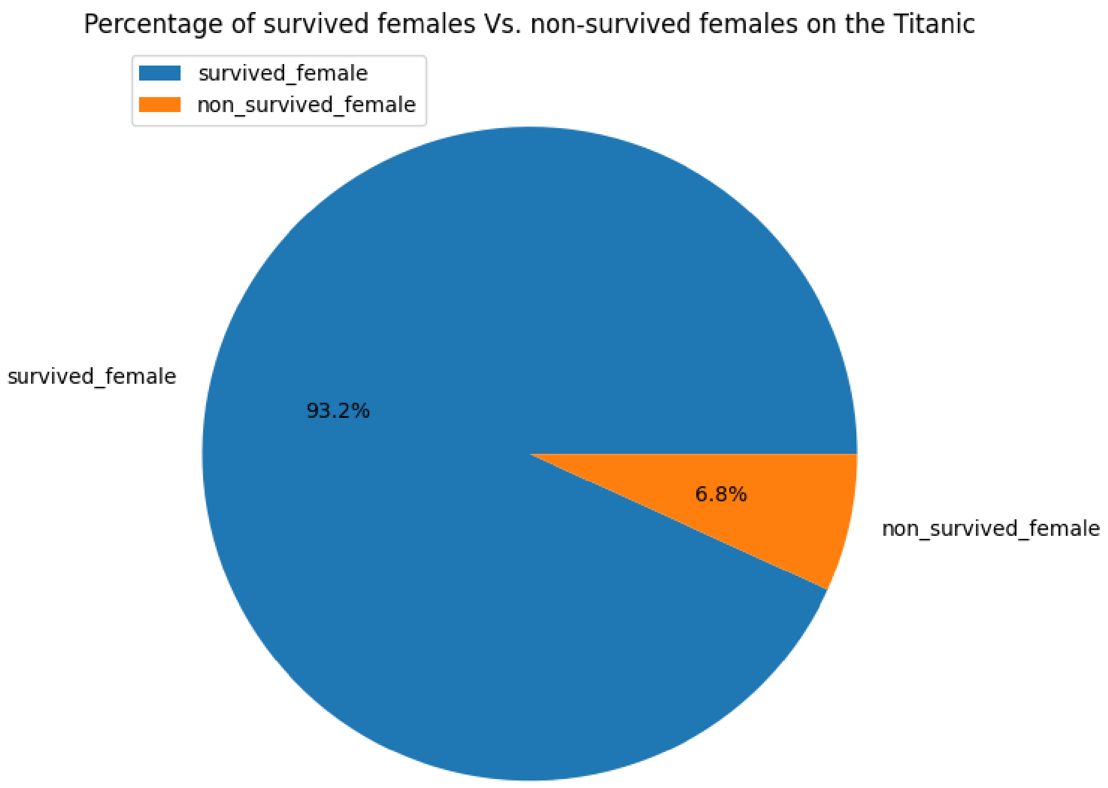


S2 - Question 5

Code

```
#%%
# Question 5
# Calculate the number of male passengers who survived and those who did not
female_survivals = titanic[titanic['sex'] == 'female'].groupby('survived')
['sex'].count()


labels = ['survived_female', 'non_survived_female']
sizes = [female_survivals[1] if 1 in female_survivals else 0, female_survivals[0]
if 0 in female_survivals else 0]

plt.figure(figsize=(8,6))
plot_S2_Q4 = plt.pie(sizes, autopct='%1.1f%%',
                     labels=labels)
plt.title('Percentage of survived females Vs. non-survived females on the
Titanic')
plt.legend(loc='upper left')


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

Answer

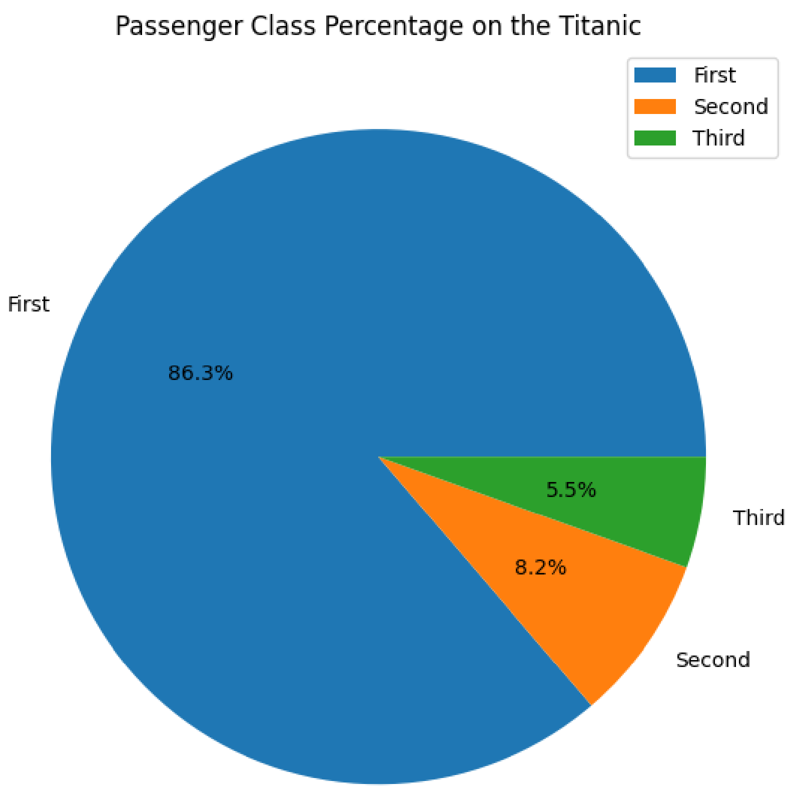


S2 - Question 6

Code

```
#%%
# Question 6
# Calculate the diff class numbers
class_counts = titanic['class'].value_counts()
plt.figure(figsize=(8,6))
plot_S2_Q5 = plt.pie(class_counts,
                  labels=class_counts.index,
                  autopct='%1.1f%%',
                  )
plt.title('Passenger Class Percentage on the Titanic')
plt.legend()

plt.tight_layout()
plt.show()


# Show the percentage of Male and Female
print('The percentage of diff on Titanic:\n')
print(titanic['class'].value_counts(normalize=True).round(2))
```

Answer

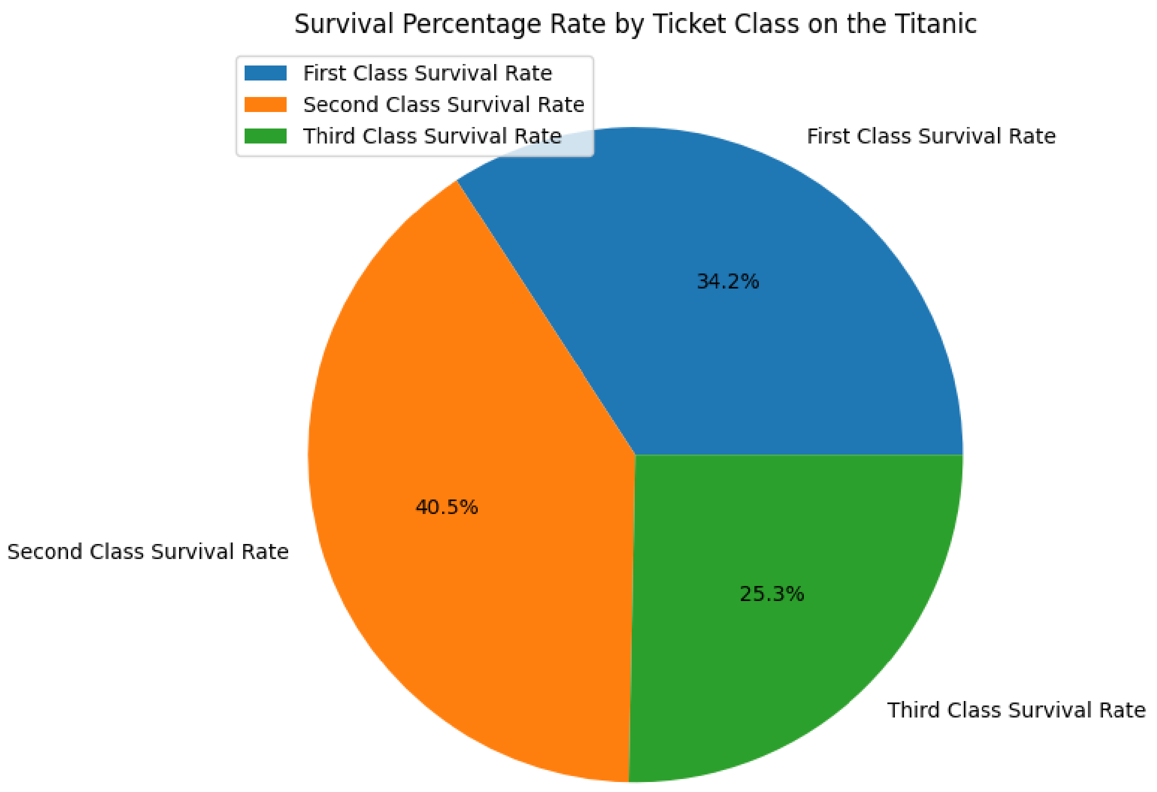


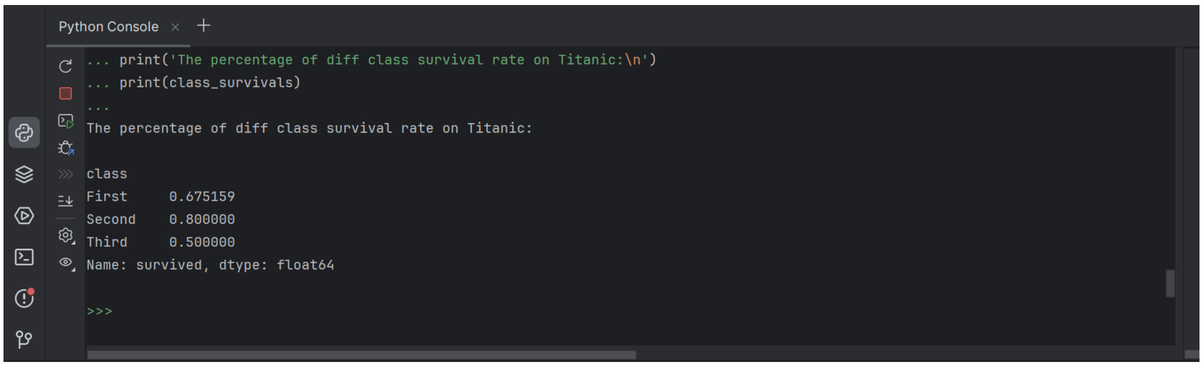
S2 - Question 7

Code

```
#%%
# Question 7
# Calculate the number of survived passengers based on classes
class_survivals = titanic.groupby('class')['survived'].mean()

#%%
labels = ['First Class Survival Rate', 'Second Class Survival Rate', 'Third Class
Survival Rate']
sizes = [class_survivals[i] for i in sorted(class_survivals.index)]

# Convert survival rates to percentages for the pie chart
sizes_percentage = [rate * 100 for rate in sizes]

plt.figure(figsize=(8,6))
plot_S2_Q7 = plt.pie(sizes, autopct='%1.1f%%',
                     labels=labels)
plt.title('Survival Percentage Rate by Ticket Class on the Titanic')
plt.legend(loc='upper left')



plt.tight_layout()
plt.show()

print('The percentage of diff class survival rate on Titanic:\n')
print(class_survivals)
```

Answer





```
... print('The percentage of diff class survival rate on Titanic:\n')
... print(class_survivals)
...
The percentage of diff class survival rate on Titanic:

class
First     0.675159
Second    0.800000
Third     0.500000
Name: survived, dtype: float64

>>>
```

S2 - Question 8

Code

Answer

S2 - Question 9

Code

Answer