# Chaitanya Ambekar

# TC77

## Question 1 (Age-Income Dataset)

In [1]:

```
# importing all libraries
import math
import statistics
import numpy as np
import scipy.stats
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
df = pd.read_csv("D:\DOWNLOADS\Age-Income-Dataset - Sheet1.csv")
df.head()
```

Out[2]:

|  | Age | Income |
|---|---|---|
| **0** | Young | 25000 |
| **1** | Middle Age | 54000 |
| **2** | Old | 60000 |
| **3** | Young | 15000 |
| **4** | Young | 45000 |

In [3]:

```
df.describe()
```

Out[3]:

|  | Income |
|---|---|
| **count** | 50.000000 |
| **mean** | 50966.000000 |
| **std** | 21096.683268 |
| **min** | 15000.000000 |
| **25%** | 33475.000000 |
| **50%** | 46850.000000 |
| **75%** | 65400.000000 |
| **max** | 93000.000000 |

In [4]:

```
missing_values = df.isnull().sum()
print(missing_values)
```

```
Age       0
Income    0
dtype: int64
```

# Summary Statistics

In [5]:

```
# mean with built-in functions
mean_sal=statistics.mean(df['Income'])
print(mean_sal)
```

```
50966
```

In [6]:

```
# mean without built-in functions
mean_sal = (sum(df['Income']))/(np.count_nonzero(df['Income']))
print(mean_sal)
```

```
50966.0
```

In [7]:

```
# median with built-in functions
median_sal = statistics.median(df['Income'])
print(median_sal)
```

```
46850.0
```

In [8]:

```
# median without built-in functions
n = np.count_nonzero(df['Income'])
if n%2:
    median_sal = sorted(df['Income']) [round(0.5*(n))]
else:
    x_ord, index = sorted(df['Income']), round(0.5*n)
    median_sal = 0.5 * (x_ord[index-1] + x_ord[index])
print(median_sal)
```

```
46850.0
```

In [9]:

```
# calculating minimum and maximum salary with built-in functions
min_sal = min(df['Income'])
max_sal = max(df['Income'])
print("Minimum salary is: ",min_sal)
print("Maximum salary is: ",max_sal)
```

```
Minimum salary is:  15000
Maximum salary is:  93000
```

In [10]:

```
# calculating minimum and maximum salary without built-in functions
n = df['Income']
min_sal= 99999999
max_sal = 0
for i in n:
    if min_sal>i:
        min_sal = i
    if max_sal<i:
        max_sal = i
print("Minimum salary is: ",min_sal)
print("Maximum salary is: ",max_sal)
```

```
Minimum salary is:  15000
Maximum salary is:  93000
```

In [11]:

```
# variance with built-in functions
var_sal = statistics.variance(df['Income'])
print(var_sal)
```

```
445070044.8979592
```

In [12]:

```
# variance without built-in functions
n = np.count_nonzero(df['Income'])
mean_sal = sum(df['Income']) / n
var_sal = sum((item - mean_sal)**2 for item in df['Income']) / (n - 1)
print(var_sal)
```

```
445070044.8979592
```

In [13]:

```
# standard deviation with built-in functions
std_sal = statistics.stdev(df['Income'])
print(std_sal)
```

```
21096.683267707253
```

In [14]:

```
# standard deviation without built-in functions
std_sal = var_sal**0.5
print(std_sal)
```

```
21096.683267707253
```

In [15]:

```
df.nunique()
```

Out[15]:

```
Age        3
Income    45
dtype: int64
```

In [16]:

```
df1 = df.groupby('Age')['Income'].apply(list)
df1
```

Out[16]:

```
Age
Middle Age    [54000, 27000, 29000, 57000, 56000, 90000, 930...
Old           [60000, 52000, 80000, 75000, 35000, 43000, 630...
Young         [25000, 15000, 45000, 65000, 70000, 30000, 230...
Name: Income, dtype: object
```

In [17]:

```
# Summary statistics of Middle Age group

# Mean
mean_sal_ma=statistics.mean(df1['Middle Age'])
print("Mean Salary is: ",mean_sal_ma)

# Median
median_sal_ma = statistics.median(df1['Middle Age'])
print("Median Salary is: ",median_sal_ma)

# Min and Max
min_sal_ma = min(df1['Middle Age'])
max_sal_ma = max(df1['Middle Age'])
print("Minimum salary is: ",min_sal_ma)
print("Maximum salary is: ",max_sal_ma)

# Variance
var_sal_ma = statistics.variance(df1['Middle Age'])
print("Variance is: ",var_sal_ma)

# Standard Deviation
std_sal_ma = statistics.stdev(df1['Middle Age'])
print("Standard Deviation is:",std_sal_ma)
```

```
Mean Salary is:  52453.333333333336
Median Salary is:  53200
Minimum salary is:  25600
Maximum salary is:  93000
Variance is:  420159809.52380955
Standard Deviation is: 20497.800114251517
```

In [18]:

```python
# Summary statistics of Old Age group

# Mean
mean_sal_old=statistics.mean(df1['Old'])
print("Mean Salary is: ",mean_sal_old)

# Median
median_sal_old = statistics.median(df1['Old'])
print("Median Salary is: ",median_sal_old)

# Min and Max
min_sal_old = min(df1['Old'])
max_sal_old = max(df1['Old'])
print("Minimum salary is: ",min_sal_old)
print("Maximum salary is: ",max_sal_old)

# Variance
var_sal_old = statistics.variance(df1['Old'])
print("Variance is: ",var_sal_old)

# Standard Deviation
std_sal_old = statistics.stdev(df1['Old'])
print("Standard Deviation is:",std_sal_old)
```

```
Mean Salary is:  53942.10526315789
Median Salary is:  45300
Minimum salary is:  24500
Maximum salary is:  89700
Variance is:  435480350.877193
Standard Deviation is: 20868.165968220423
```

In [19]:

```python
# Summary statistics of Young Age group

# Mean
mean_sal_y=statistics.mean(df1['Young'])
print("Mean Salary is: ",mean_sal_y)

# Median
median_sal_y = statistics.median(df1['Young'])
print("Median Salary is: ",median_sal_y)

# Min and Max
min_sal_y = min(df1['Young'])
max_sal_y = max(df1['Young'])
print("Minimum salary is: ",min_sal_y)
print("Maximum salary is: ",max_sal_y)

# Variance
var_sal_y = statistics.variance(df1['Young'])
print("Variance is: ",var_sal_y)

# Standard Deviation
std_sal_y = statistics.stdev(df1['Young'])
print("Standard Deviation is:",std_sal_y)
```

```
Mean Salary is:  46037.5
Median Salary is:  41500.0
Minimum salary is:  15000
Maximum salary is:  87000
Variance is:  499829166.6666667
Standard Deviation is: 22356.859499193233
```

## Question 2 (Iris Dataset)

In [20]:

```python
iris = pd.read_csv("D:\DOWNLOADS\Iris - Iris.csv")
iris
```

Out[20]:

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|---|
| **0** | 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| **...** | ... | ... | ... | ... | ... | ... |
| **145** | 146 | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica |
| **146** | 147 | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica |
| **147** | 148 | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica |
| **148** | 149 | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica |
| **149** | 150 | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica |

150 rows × 6 columns

In [21]:

```
iris = iris.iloc[:,1:]
iris
```

Out[21]:

| | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica |

150 rows × 5 columns

In [22]:

```
# Filter the dataset to create separate dataframes for each species
setosa = iris[iris['Species'] == 'Iris-setosa']
versicolor = iris[iris['Species'] == 'Iris-versicolor']
virginica = iris[iris['Species'] == 'Iris-virginica']
```

In [23]:

```
# Basic statistical details
print("Setosa")
print(setosa.describe())
print("\nVersicolor")
print(versicolor.describe())
print("\nVirginica")
print(virginica.describe())
```

```
Setosa
       SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count       50.00000     50.000000      50.000000      50.00000
mean         5.00600      3.418000       1.464000       0.24400
std          0.35249      0.381024       0.173511       0.10721
min          4.30000      2.300000       1.000000       0.10000
25%          4.80000      3.125000       1.400000       0.20000
50%          5.00000      3.400000       1.500000       0.20000
75%          5.20000      3.675000       1.575000       0.30000
max          5.80000      4.400000       1.900000       0.60000

Versicolor
       SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count      50.000000     50.000000      50.000000     50.000000
mean        5.936000      2.770000       4.260000      1.326000
std         0.516171      0.313798       0.469911      0.197753
min         4.900000      2.000000       3.000000      1.000000
25%         5.600000      2.525000       4.000000      1.200000
50%         5.900000      2.800000       4.350000      1.300000
75%         6.300000      3.000000       4.600000      1.500000
max         7.000000      3.400000       5.100000      1.800000

Virginica
       SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count       50.00000     50.000000      50.000000      50.00000
mean         6.58800      2.974000       5.552000       2.02600
std          0.63588      0.322497       0.551895       0.27465
min          4.90000      2.200000       4.500000       1.40000
25%          6.22500      2.800000       5.100000       1.80000
50%          6.50000      3.000000       5.550000       2.00000
75%          6.90000      3.175000       5.875000       2.30000
max          7.90000      3.800000       6.900000       2.50000
```

In [24]:

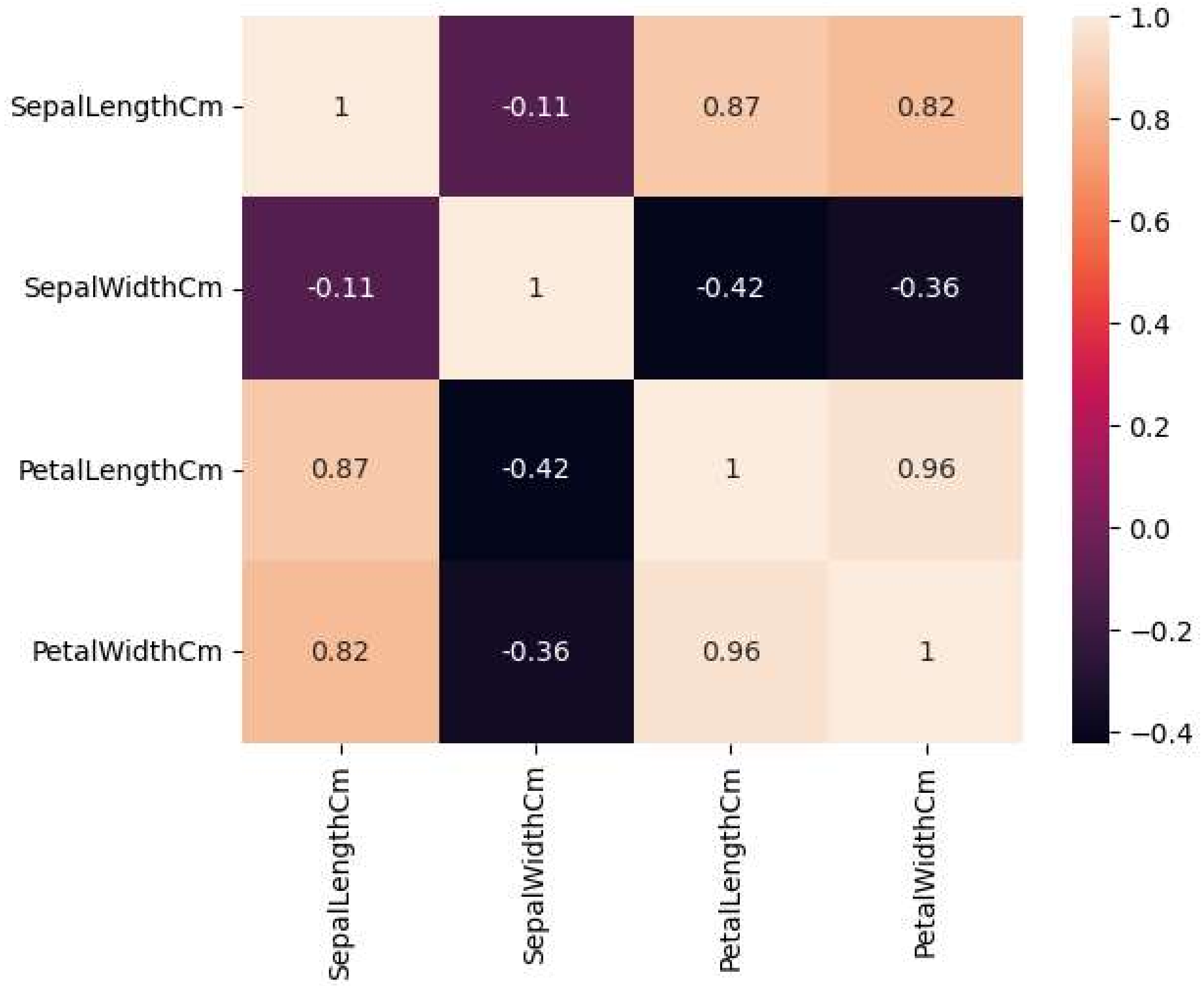
```python
# Measures of variability
print("Variance:-")
print("Setosa: \n",np.var(setosa))
print("\nVersicolor: \n", np.var(versicolor))
print("\nVirginica: \n", np.var(virginica))

print("\n \nStandard deviation:-")
print("Setosa: \n", np.std(setosa))
print("\nVersicolor: \n", np.std(versicolor))
print("\nVirginica: \n", np.std(virginica))
```

```
Variance:-
Setosa: 
 SepalLengthCm    0.121764
SepalWidthCm     0.142276
PetalLengthCm    0.029504
PetalWidthCm     0.011264
dtype: float64

Versicolor: 
 SepalLengthCm    0.261104
SepalWidthCm     0.096500
PetalLengthCm    0.216400
PetalWidthCm     0.038324
dtype: float64

Virginica: 
 SepalLengthCm    0.396256
SepalWidthCm     0.101924
PetalLengthCm    0.298496
PetalWidthCm     0.073924
dtype: float64

 
Standard deviation:-
Setosa: 
 SepalLengthCm    0.348947
SepalWidthCm     0.377195
PetalLengthCm    0.171767
PetalWidthCm     0.106132
dtype: float64

Versicolor: 
 SepalLengthCm    0.510983
SepalWidthCm     0.310644
PetalLengthCm    0.465188
PetalWidthCm     0.195765
dtype: float64

Virginica: 
 SepalLengthCm    0.629489
SepalWidthCm     0.319255
PetalLengthCm    0.546348
PetalWidthCm     0.271890
dtype: float64
```

In [25]:

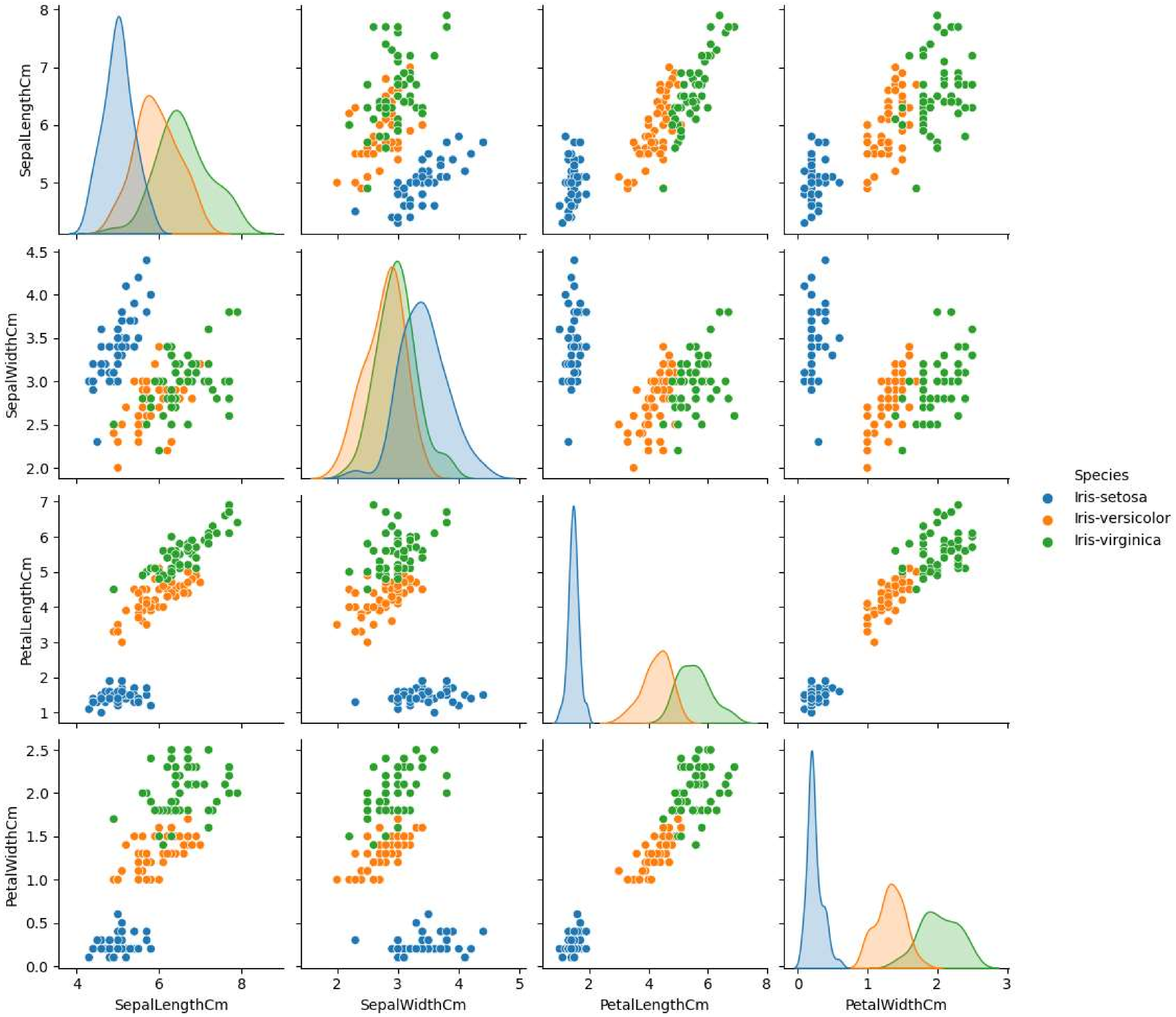
```python
# Calculate the correlation matrix
corr_matrix = iris.corr()

# Display the correlation matrix as a heatmap
```

```
sns.heatmap(corr_matrix, annot=True)
plt.show()
```



In [26]:

```
sns.pairplot(iris, hue='Species')
plt.show()
```



In [ ]: