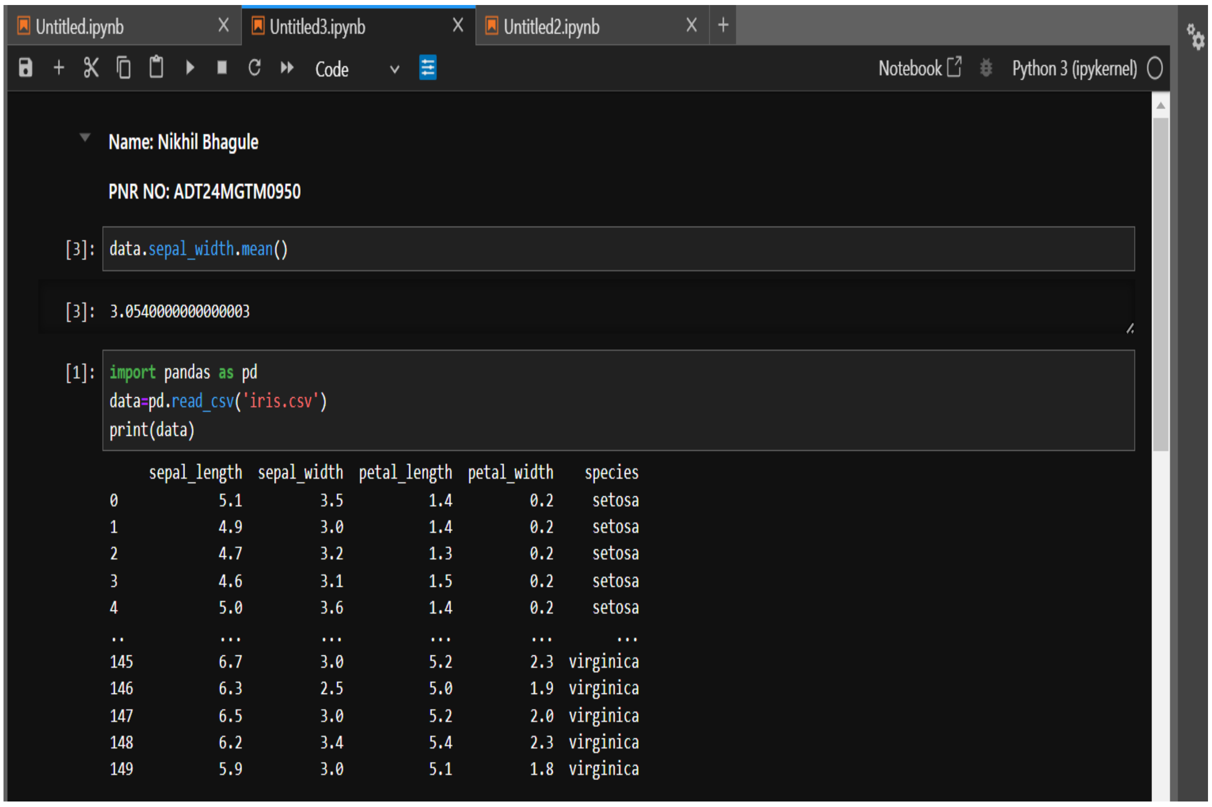
1. **Write a python to calculate the mean median ,and mode of a given dataset.**

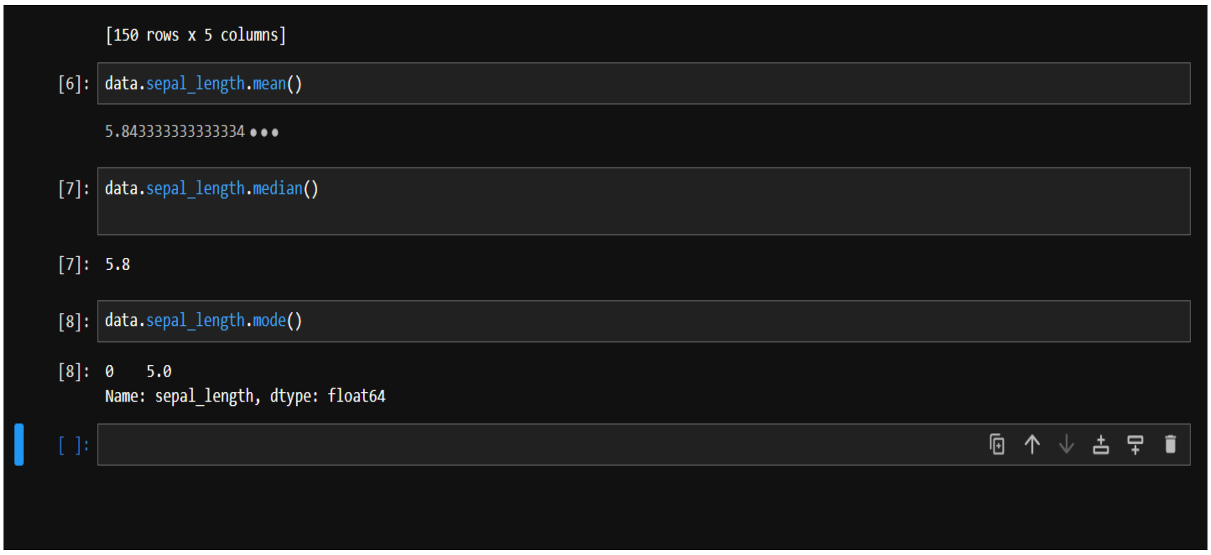
   TASK : Load a Dataset(CSV.file) with numerical data and calculate the mean,median and mode using pandas and scipy



Name: Nikhil Bhagule

PNR NO: ADT24MGTM0950

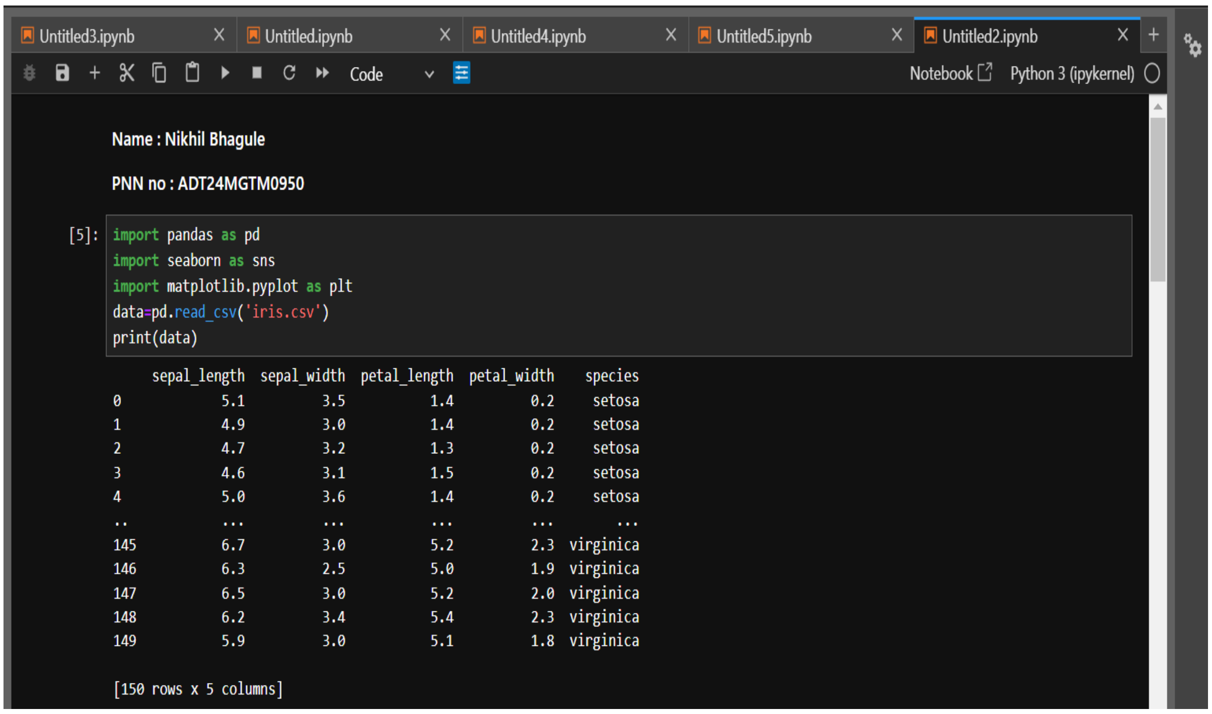
```
[3]: data.sepal_width.mean()

[3]: 3.0540000000000003

[1]: import pandas as pd
     data=pd.read_csv('iris.csv')
     print(data)

     sepal_length  sepal_width  petal_length  petal_width    species
0             5.1          3.5           1.4          0.2     setosa
1             4.9          3.0           1.4          0.2     setosa
2             4.7          3.2           1.3          0.2     setosa
3             4.6          3.1           1.5          0.2     setosa
4             5.0          3.6           1.4          0.2     setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica
```



```
[150 rows x 5 columns]

[6]: data.sepal_length.mean()

     5.843333333333334 •••

[7]: data.sepal_length.median()

[7]: 5.8

[8]: data.sepal_length.mode()

[8]: 0    5.0
     Name: sepal_length, dtype: float64

[ ]:
```

2.Question: Load a dataset containing both numerical and categorical data. Create appropriate frequency distributions for categorical data and histograms for numerical data.

Task: Use pandas and matplotlib to visualize the distributions.

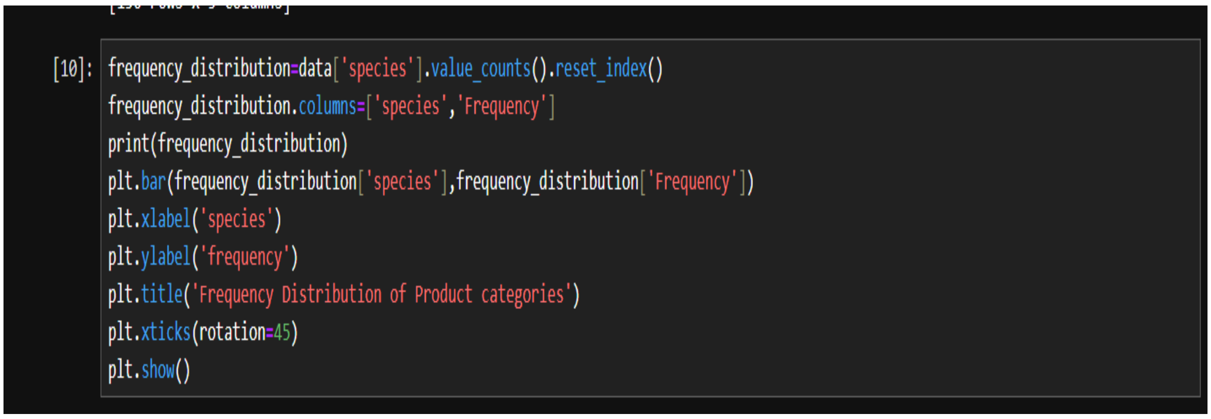


Name : Nikhil Bhagule

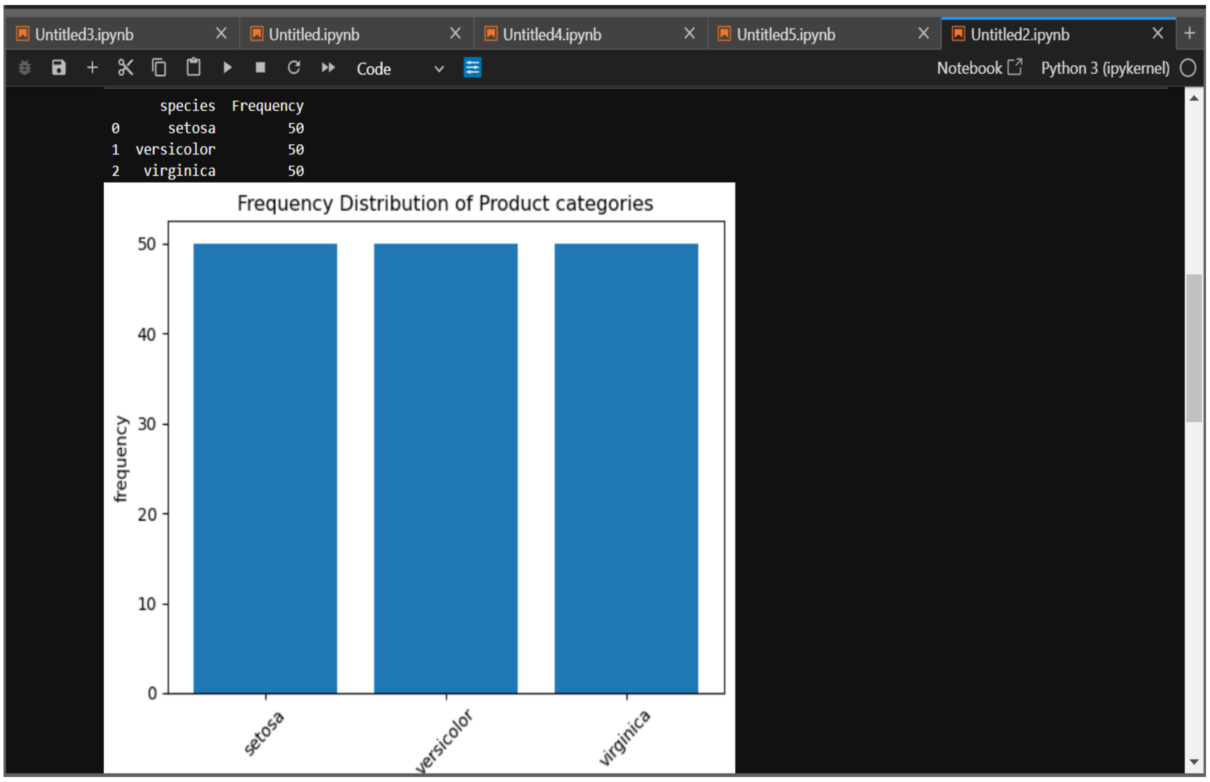
PNN no : ADT24MGTM0950

```
[5]: import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
     data=pd.read_csv('iris.csv')
     print(data)
```

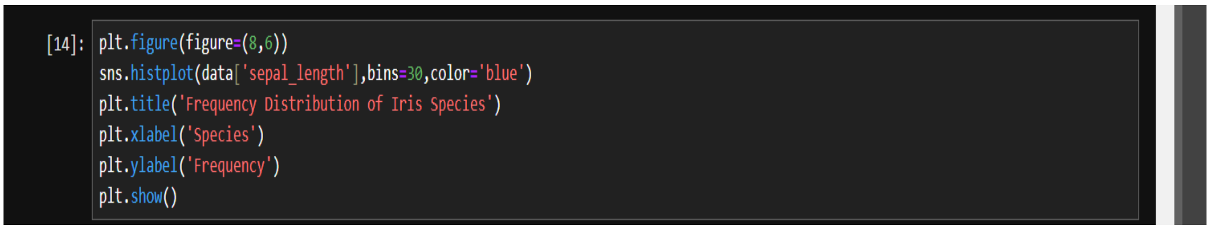
```
     sepal_length  sepal_width  petal_length  petal_width    species
0             5.1          3.5           1.4          0.2     setosa
1             4.9          3.0           1.4          0.2     setosa
2             4.7          3.2           1.3          0.2     setosa
3             4.6          3.1           1.5          0.2     setosa
4             5.0          3.6           1.4          0.2     setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica

[150 rows x 5 columns]
```



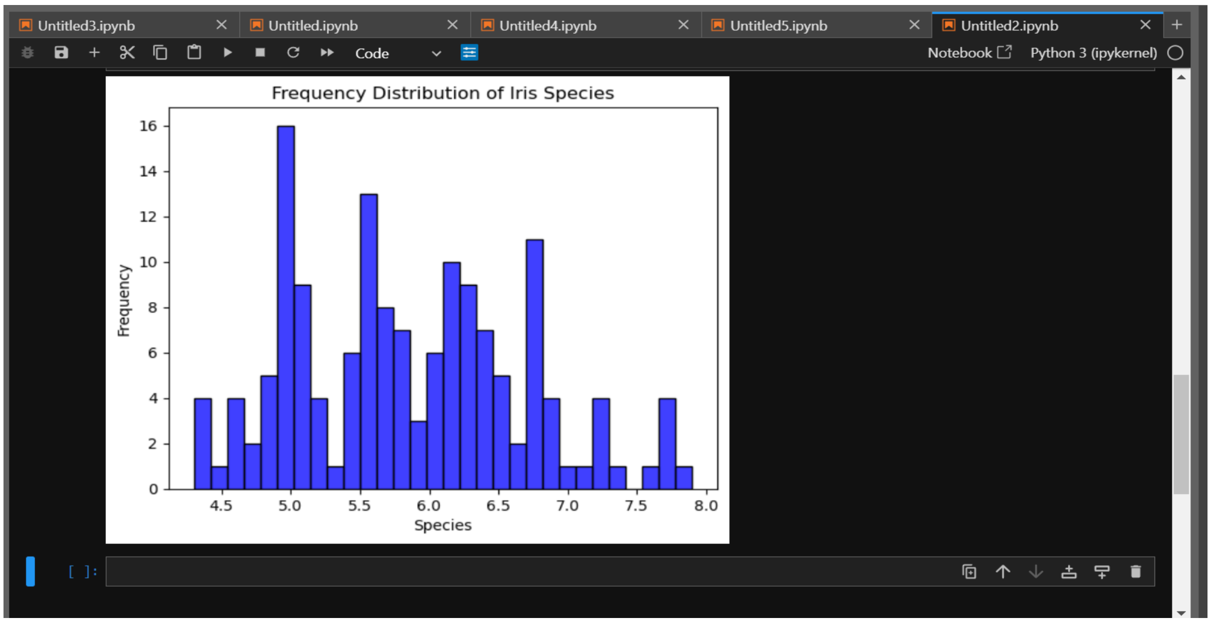
```
[10]: frequency_distribution=data['species'].value_counts().reset_index()
      frequency_distribution.columns=['species','Frequency']
      print(frequency_distribution)
      plt.bar(frequency_distribution['species'],frequency_distribution['Frequency'])
      plt.xlabel('species')
      plt.ylabel('frequency')
      plt.title('Frequency Distribution of Product categories')
      plt.xticks(rotation=45)
      plt.show()
```



```
     species  Frequency
0     setosa         50
1 versicolor         50
2  virginica         50
```

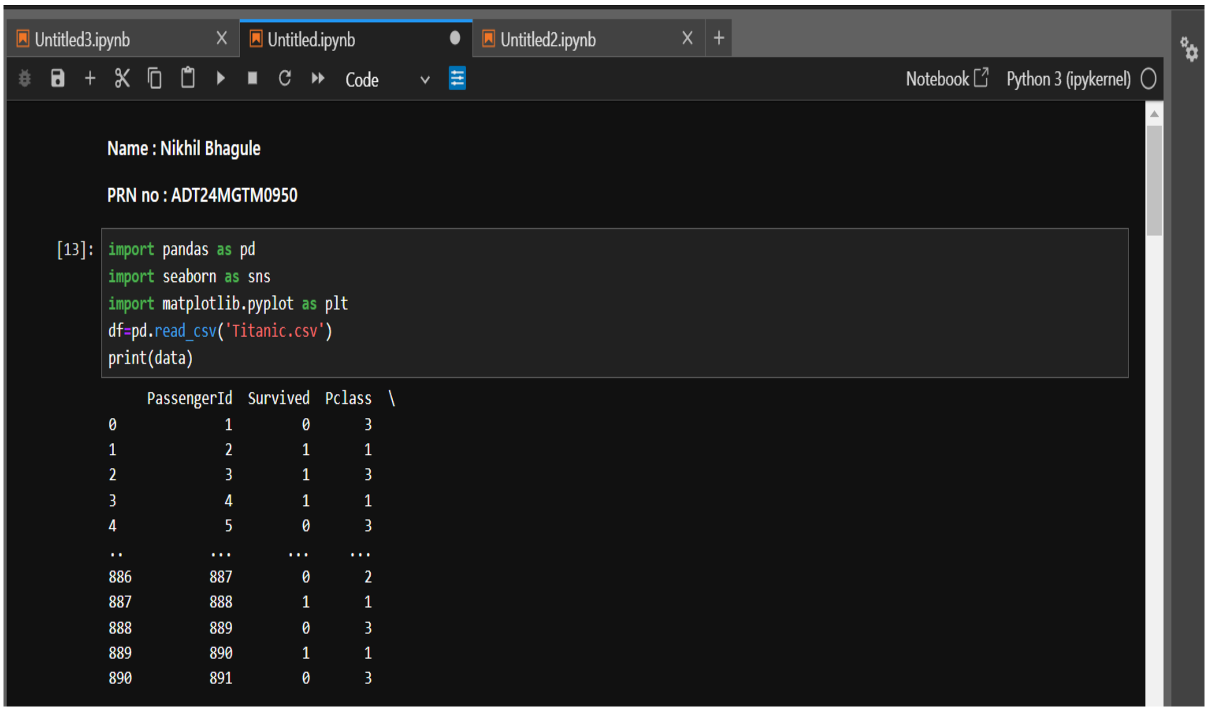


```
[14]: plt.figure(figure=(8,6))
      sns.histplot(data['sepal_length'],bins=30,color='blue')
      plt.title('Frequency Distribution of Iris Species')
      plt.xlabel('Species')
      plt.ylabel('Frequency')
      plt.show()
```



3.Question: Write a Python program to create tables, graphs, and charts to represent data visually. Use matplotlib and seaborn to plot a dataset and generate bar charts, line

graphs, and scatter plots. Task: Tabulate data and represent it using different graphs for better understanding.

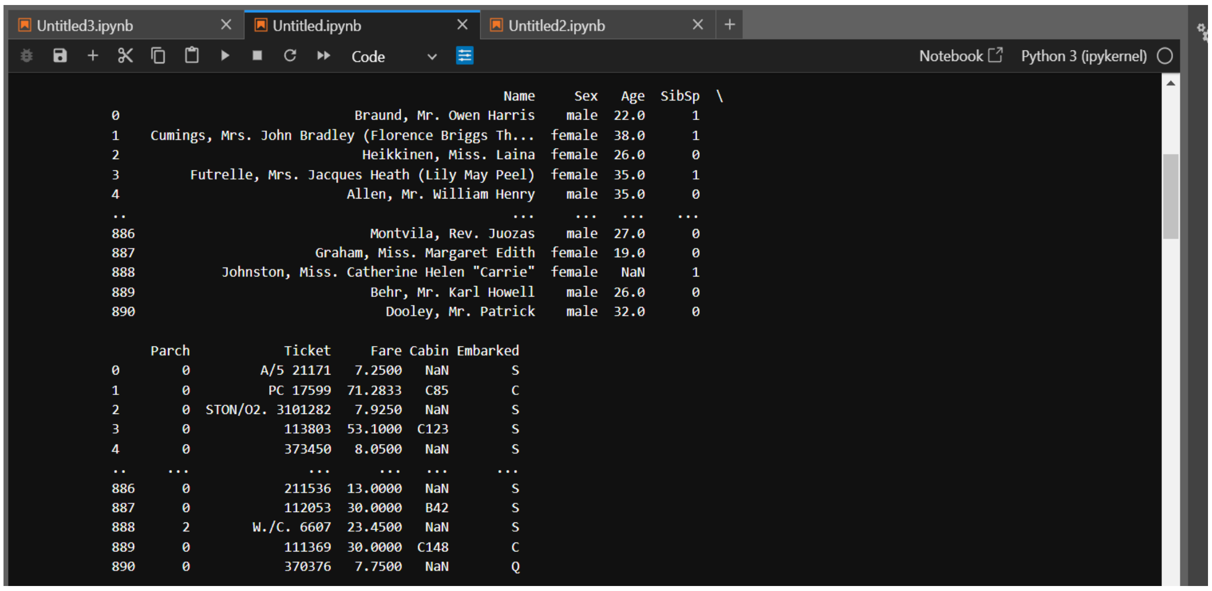


Name : Nikhil Bhagule

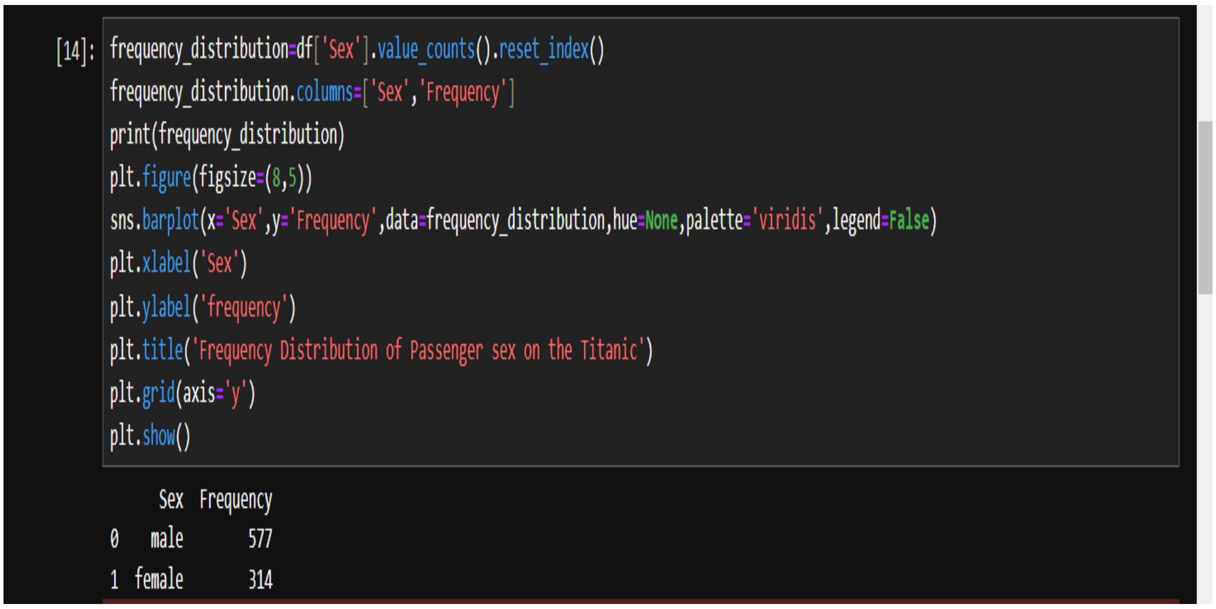
PRN no : ADT24MGTM0950

```
[13]: import pandas as pd
      import seaborn as sns
      import matplotlib.pyplot as plt
      df=pd.read_csv('Titanic.csv')
      print(data)
```

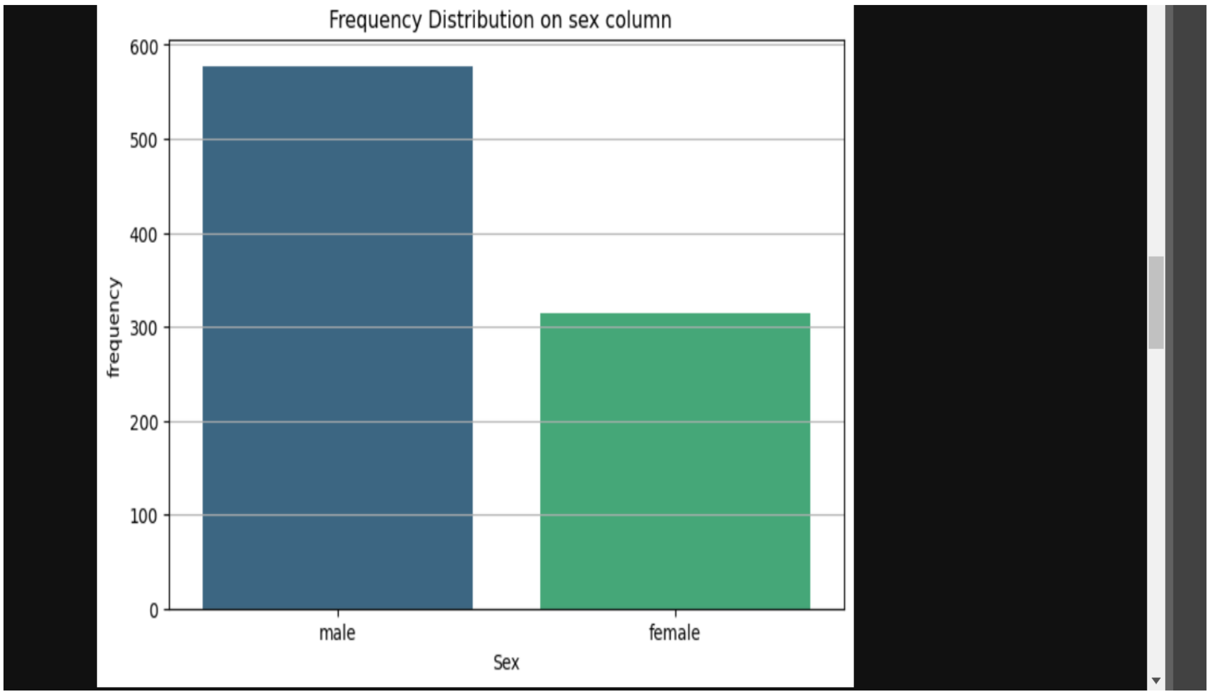
```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           ...       ...     ...
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3
```

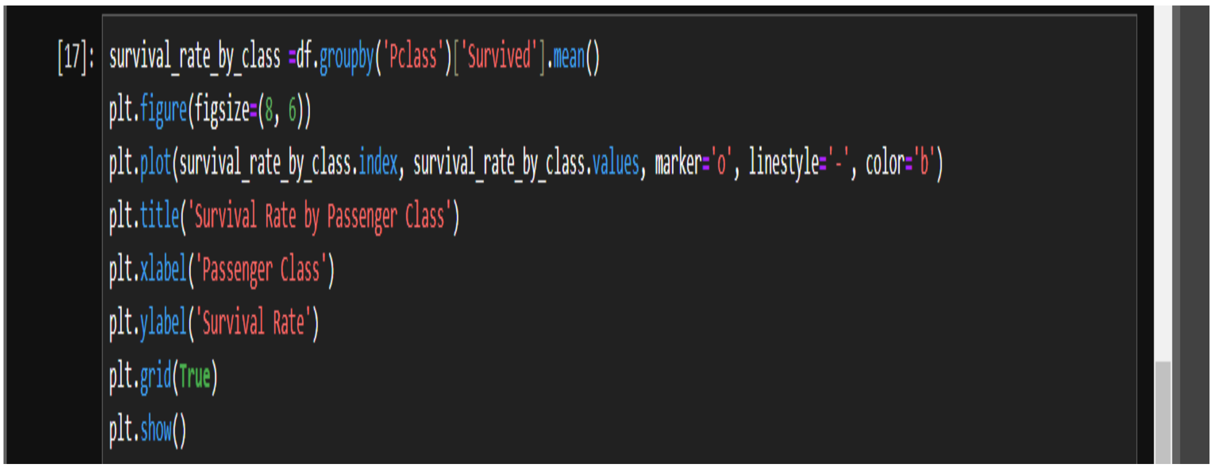


```
                                                  Name     Sex   Age  SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
..                                                 ...     ...   ...    ...
886                              Montvila, Rev. Juozas    male  27.0      0
887                       Graham, Miss. Margaret Edith  female  19.0      0
888           Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                              Behr, Mr. Karl Howell    male  26.0      0
890                                Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     ...               ...      ...   ...      ...
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q
```

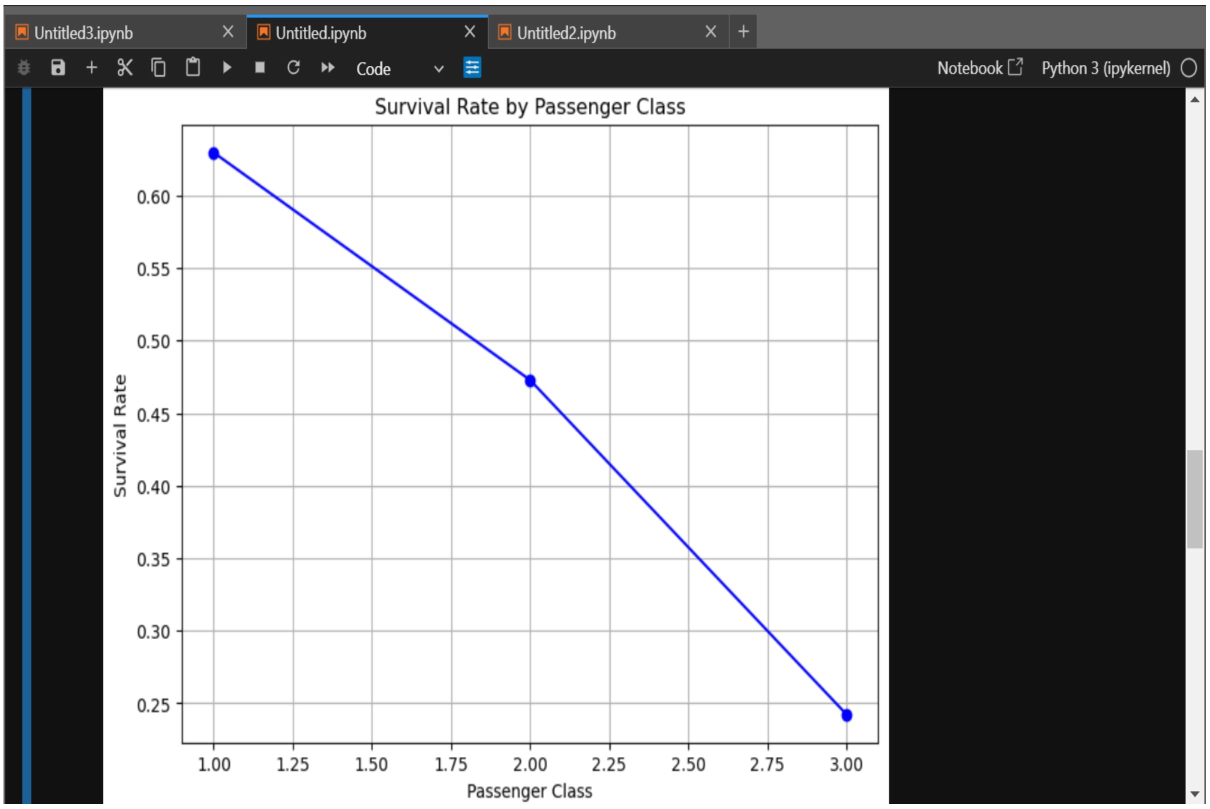


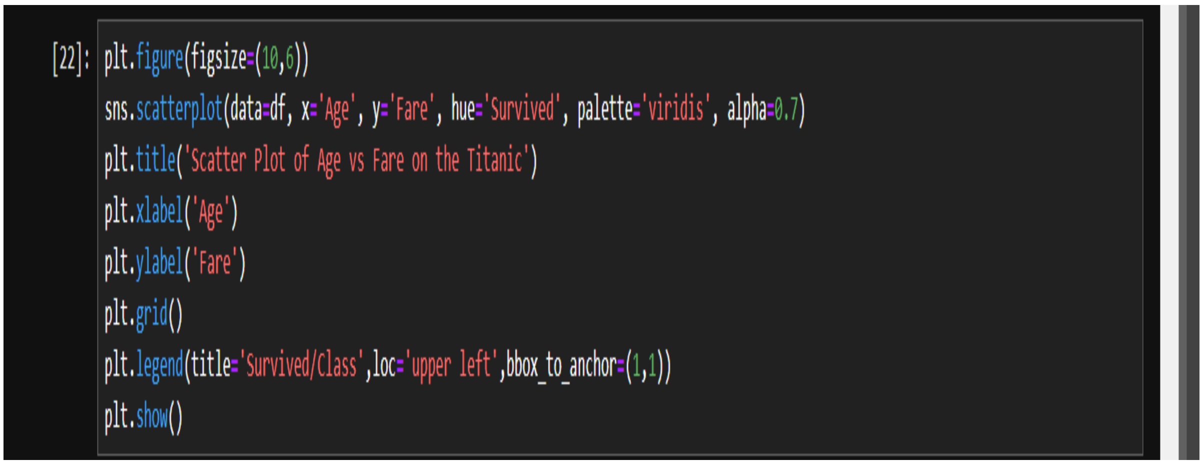
```
[14]: frequency_distribution=df['Sex'].value_counts().reset_index()
      frequency_distribution.columns=['Sex','Frequency']
      print(frequency_distribution)
      plt.figure(figsize=(8,5))
      sns.barplot(x='Sex',y='Frequency',data=frequency_distribution,hue=None,palette='viridis',legend=False)
      plt.xlabel('Sex')
      plt.ylabel('frequency')
      plt.title('Frequency Distribution of Passenger sex on the Titanic')
      plt.grid(axis='y')
      plt.show()

         Sex  Frequency
      0    male       577
      1  female       314
```



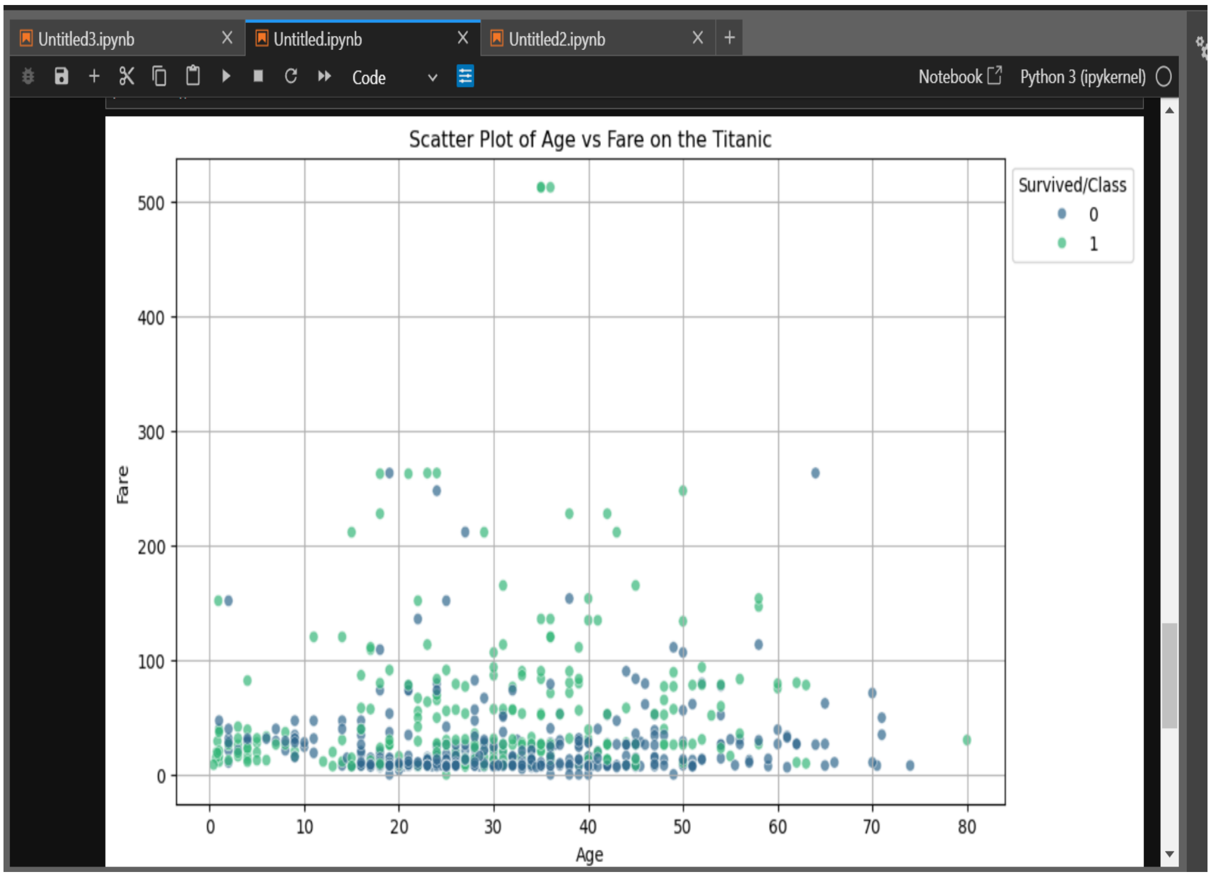


```
[17]: survival_rate_by_class =df.groupby('Pclass')['Survived'].mean()
      plt.figure(figsize=(8, 6))
      plt.plot(survival_rate_by_class.index, survival_rate_by_class.values, marker='o', linestyle='-', color='b')
      plt.title('Survival Rate by Passenger Class')
      plt.xlabel('Passenger Class')
      plt.ylabel('Survival Rate')
      plt.grid(True)
      plt.show()
```



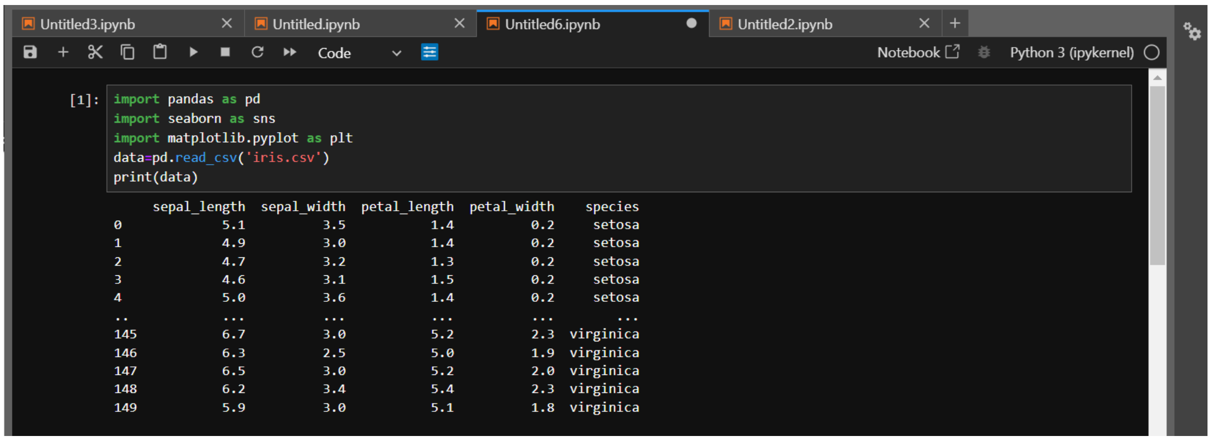


```
[22]: plt.figure(figsize=(10,6))
      sns.scatterplot(data=df, x='Age', y='Fare', hue='Survived', palette='viridis', alpha=0.7)
      plt.title('Scatter Plot of Age vs Fare on the Titanic')
      plt.xlabel('Age')
      plt.ylabel('Fare')
      plt.grid()
      plt.legend(title='Survived/Class',loc='upper left',bbox_to_anchor=(1,1))
      plt.show()
```



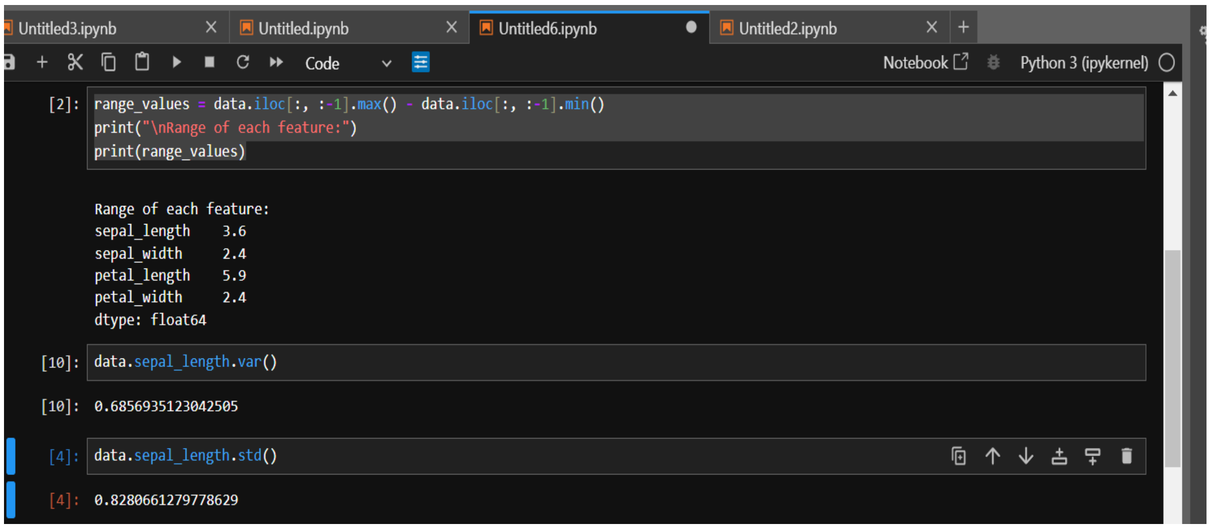
4.Question: Write a Python script to calculate the range, variance, standard deviation, and coefficient of variation for a given numerical dataset.

Task: Implement numpy and pandas to compute the required measures of dispersion



```
[1]: import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
     data=pd.read_csv('iris.csv')
     print(data)
```

```
     sepal_length  sepal_width  petal_length  petal_width    species
0             5.1          3.5           1.4          0.2     setosa
1             4.9          3.0           1.4          0.2     setosa
2             4.7          3.2           1.3          0.2     setosa
3             4.6          3.1           1.5          0.2     setosa
4             5.0          3.6           1.4          0.2     setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica
```



```
[2]: range_values = data.iloc[:, :-1].max() - data.iloc[:, :-1].min()
     print("\nRange of each feature:")
     print(range_values)
```

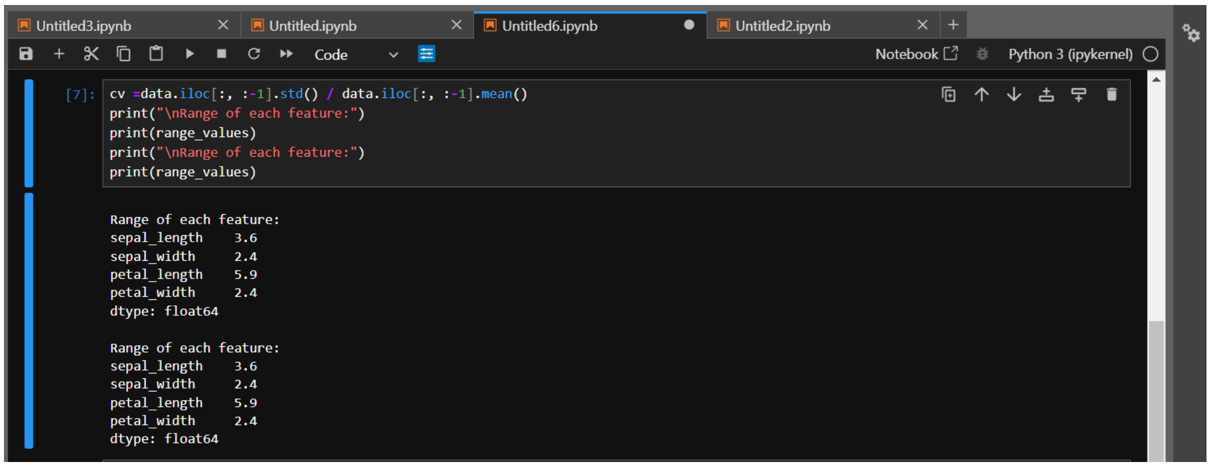
```

Range of each feature:
sepal_length    3.6
sepal_width     2.4
petal_length    5.9
petal_width     2.4
dtype: float64
```

```
[10]: data.sepal_length.var()
```

```
[10]: 0.6856935123042505
```

```
[4]: data.sepal_length.std()
```

```
[4]: 0.8280661279778629
```

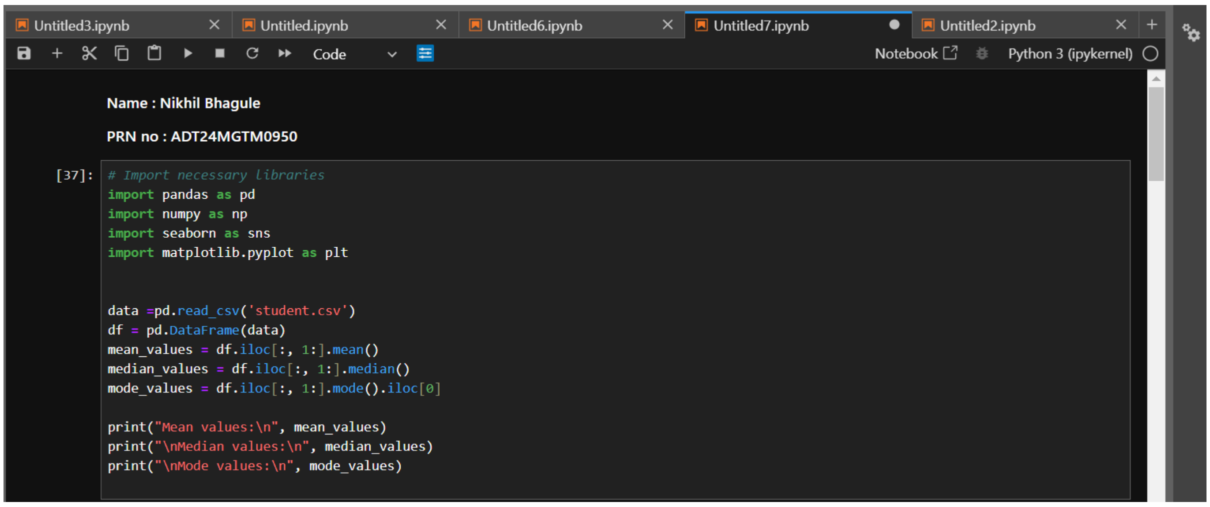


```
[7]: cv =data.iloc[:, :-1].std() / data.iloc[:, :-1].mean()
     print("\nRange of each feature:")
     print(range_values)
     print("\nRange of each feature:")
     print(range_values)
```

```

Range of each feature:
sepal_length    3.6
sepal_width     2.4
petal_length    5.9
petal_width     2.4
dtype: float64

Range of each feature:
sepal_length    3.6
sepal_width     2.4
petal_length    5.9
petal_width     2.4
dtype: float64
```

5.Question: Prepare a dataset and calculate the mean, median, and mode. Discuss the differences between these measures and their respective merits and demerits.

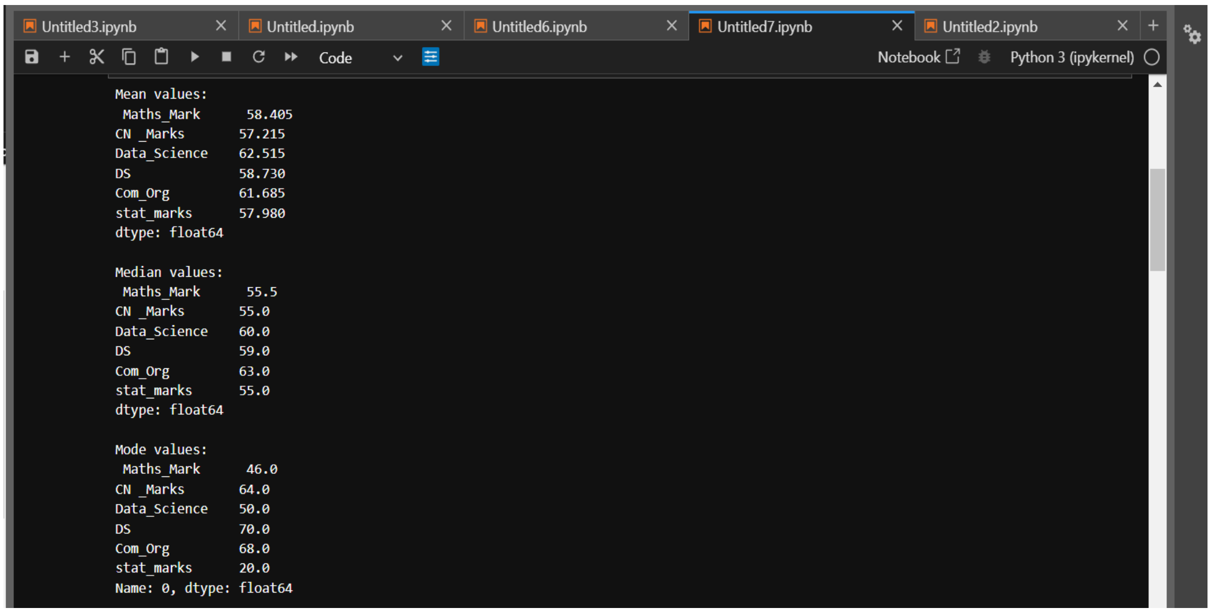
Task: Write a Python program to compare the central tendencies and visualize the differences



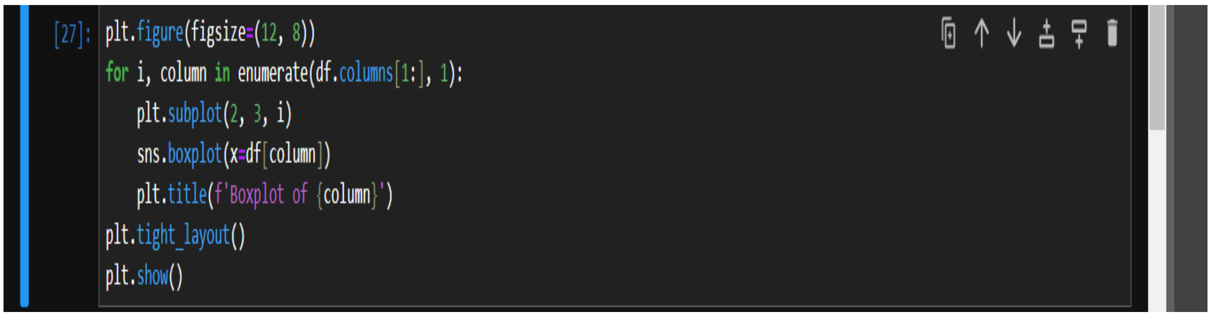
Name : Nikhil Bhagule

PRN no : ADT24MGTM0950

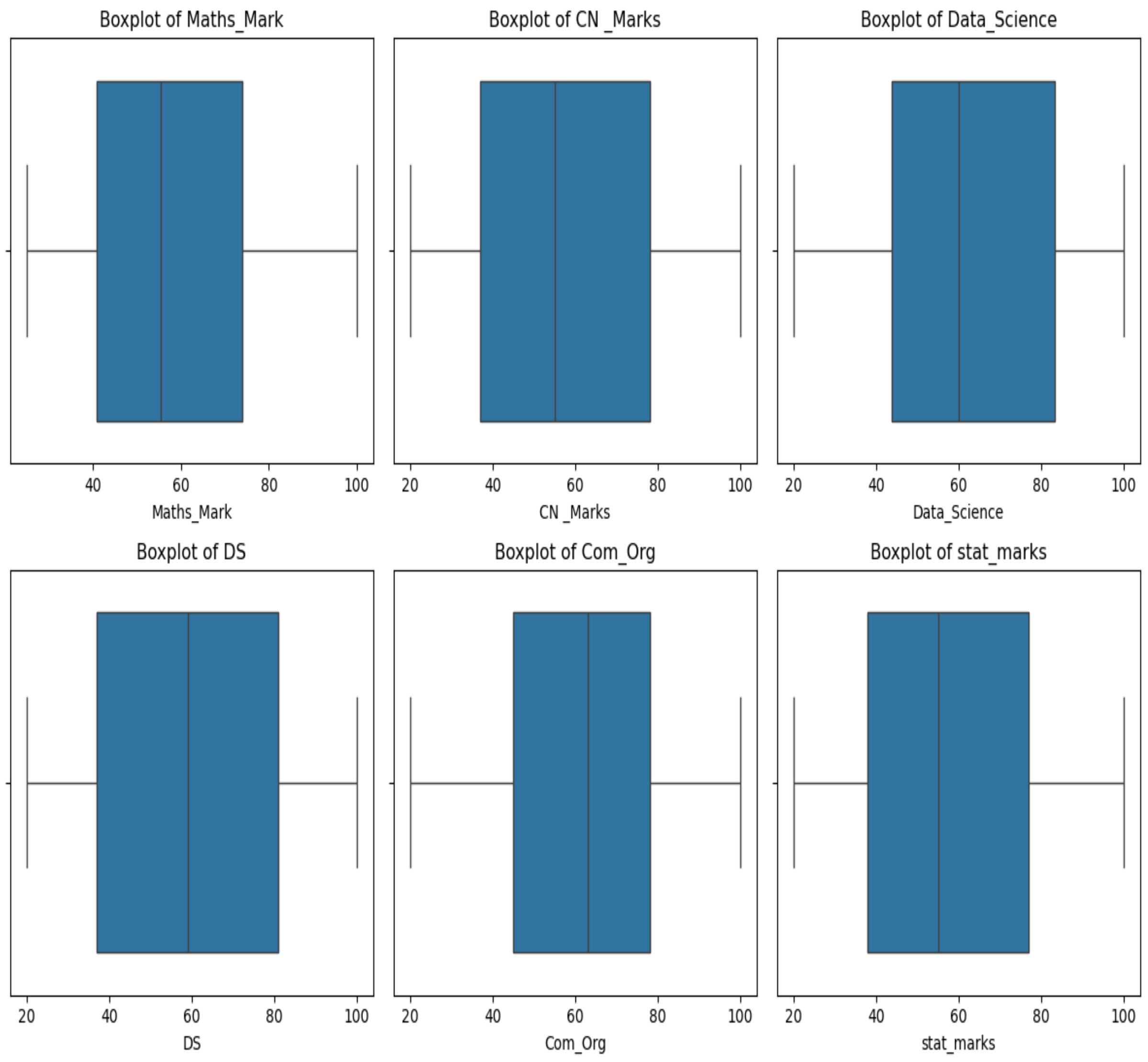
```
[37]: # Import necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt


data =pd.read_csv('student.csv')
df = pd.DataFrame(data)
mean_values = df.iloc[:, 1:].mean()
median_values = df.iloc[:, 1:].median()
mode_values = df.iloc[:, 1:].mode().iloc[0]

print("Mean values:\n", mean_values)
print("\nMedian values:\n", median_values)
print("\nMode values:\n", mode_values)
```

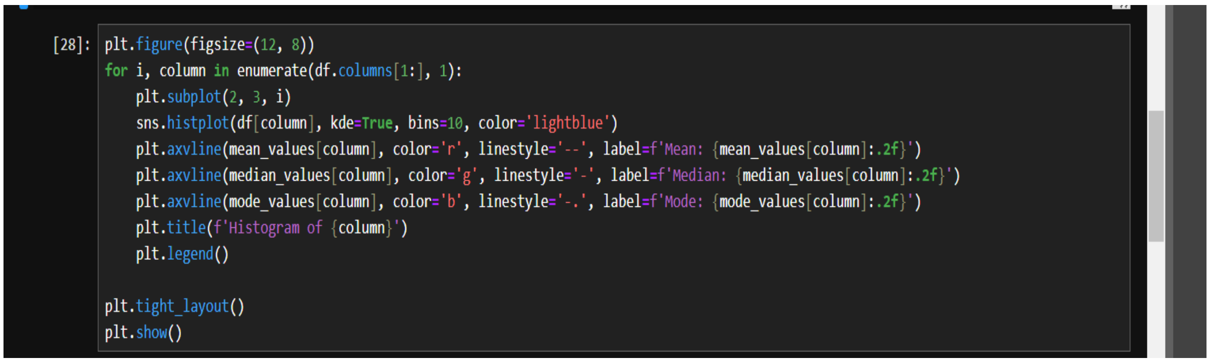


```
Mean values:
 Maths_Mark      58.405
CN _Marks       57.215
Data_Science    62.515
DS              58.730
Com_Org         61.685
stat_marks      57.980
dtype: float64

Median values:
 Maths_Mark      55.5
CN _Marks       55.0
Data_Science    60.0
DS              59.0
Com_Org         63.0
stat_marks      55.0
dtype: float64

Mode values:
 Maths_Mark      46.0
CN _Marks       64.0
Data_Science    50.0
DS              70.0
Com_Org         68.0
stat_marks      20.0
Name: 0, dtype: float64
```

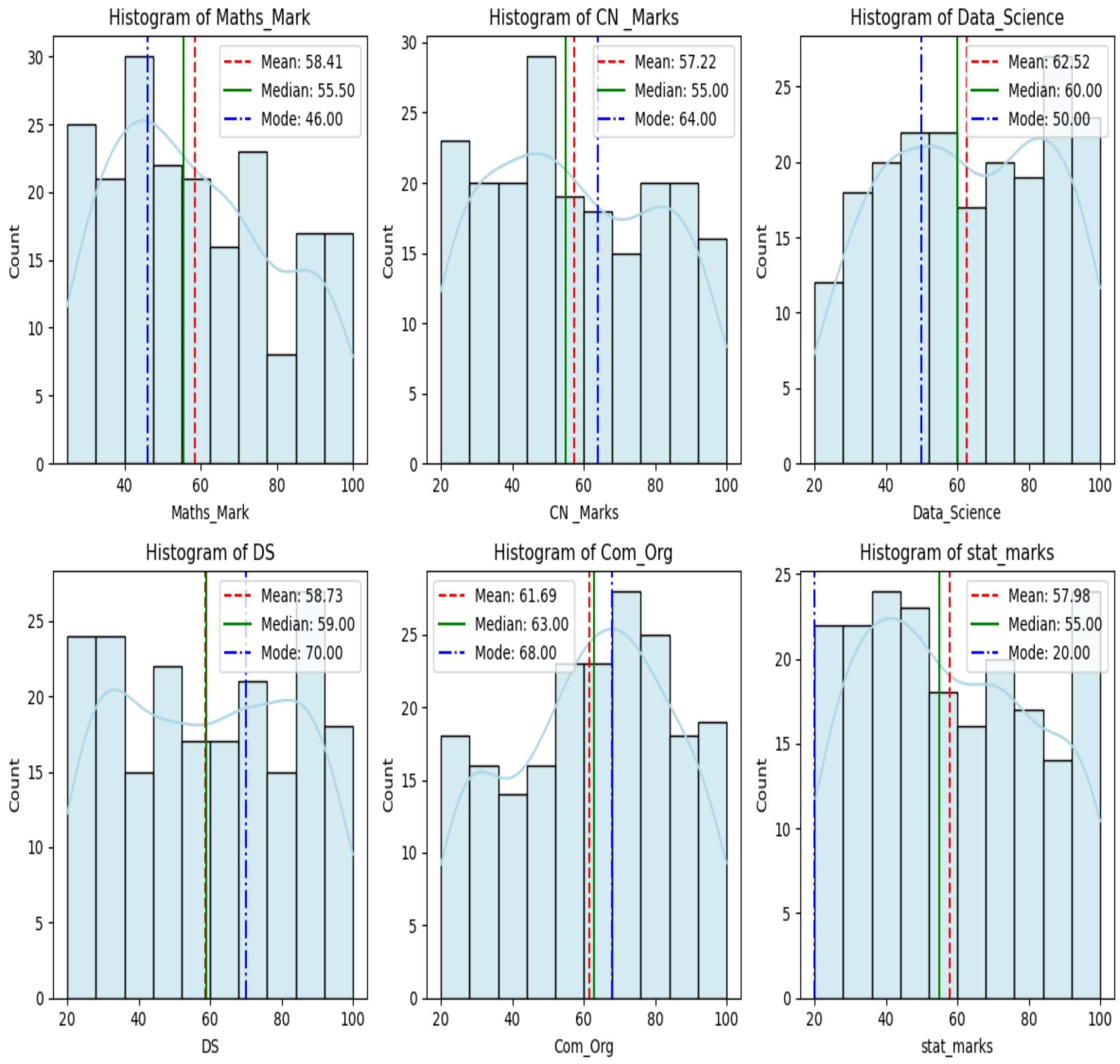


```
[27]: plt.figure(figsize=(12, 8))
      for i, column in enumerate(df.columns[1:], 1):
          plt.subplot(2, 3, i)
          sns.boxplot(x=df[column])
          plt.title(f'Boxplot of {column}')
      plt.tight_layout()
      plt.show()
```





```
[28]: plt.figure(figsize=(12, 8))
      for i, column in enumerate(df.columns[1:], 1):
          plt.subplot(2, 3, i)
          sns.histplot(df[column], kde=True, bins=10, color='lightblue')
          plt.axvline(mean_values[column], color='r', linestyle='--', label=f'Mean: {mean_values[column]:.2f}')
          plt.axvline(median_values[column], color='g', linestyle='-', label=f'Median: {median_values[column]:.2f}')
          plt.axvline(mode_values[column], color='b', linestyle='-.', label=f'Mode: {mode_values[column]:.2f}')
          plt.title(f'Histogram of {column}')
          plt.legend()

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



Merits and Demerits of Mean, Median, and Mode:

Mean:

- **Merits**:
    - Simple and commonly used.
    - Provides a good summary when data is symmetrically distributed.
    - Uses all data points for a more precise measure of central tendency.
- **Demerits**:
    - **Sensitive to outliers**: A single extreme value can significantly affect the mean, making it less reliable when there are outliers or skewed data.

Median:

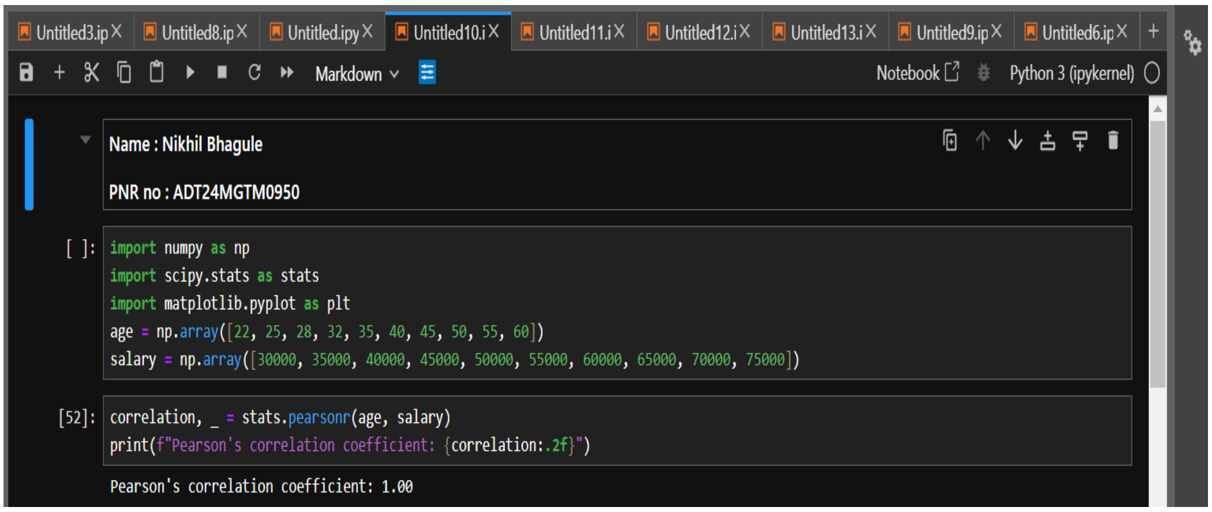
- **Merits**:
    - **Resistant to outliers**: The median is not influenced by extreme values, making it a better measure for skewed data or when outliers are present.
    - Suitable for non-symmetric distributions.
- **Demerits**:
    - It does not use all the data points, so it can be less informative when the data is well-behaved and symmetric.

Mode:

- **Merits**:
    - Useful for identifying the most frequent value in categorical or discrete data.
    - **Does not require numerical data** (can be used for categorical data).
- **Demerits**:
    - **Not applicable for continuous data**: The mode may not be useful for continuous numerical data, especially if all values are unique or very close to each other.
    - There can be **multiple modes** or no mode at all if all values occur with the same frequency.

6.Question: Write a Python program to calculate Pearson's correlation coefficient and create a scatter plot showing the correlation between two variables (e.g., age and salary).

Task: Use scipy to calculate the correlation and matplotlib for the scatter plot.



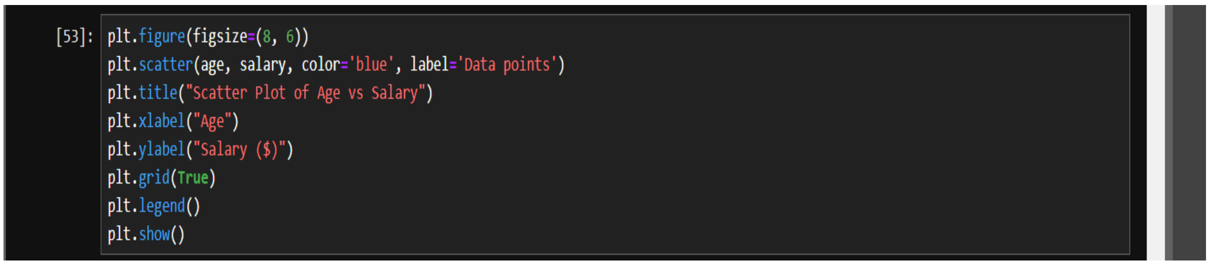
Name : Nikhil Bhagule

PNR no : ADT24MGTM0950

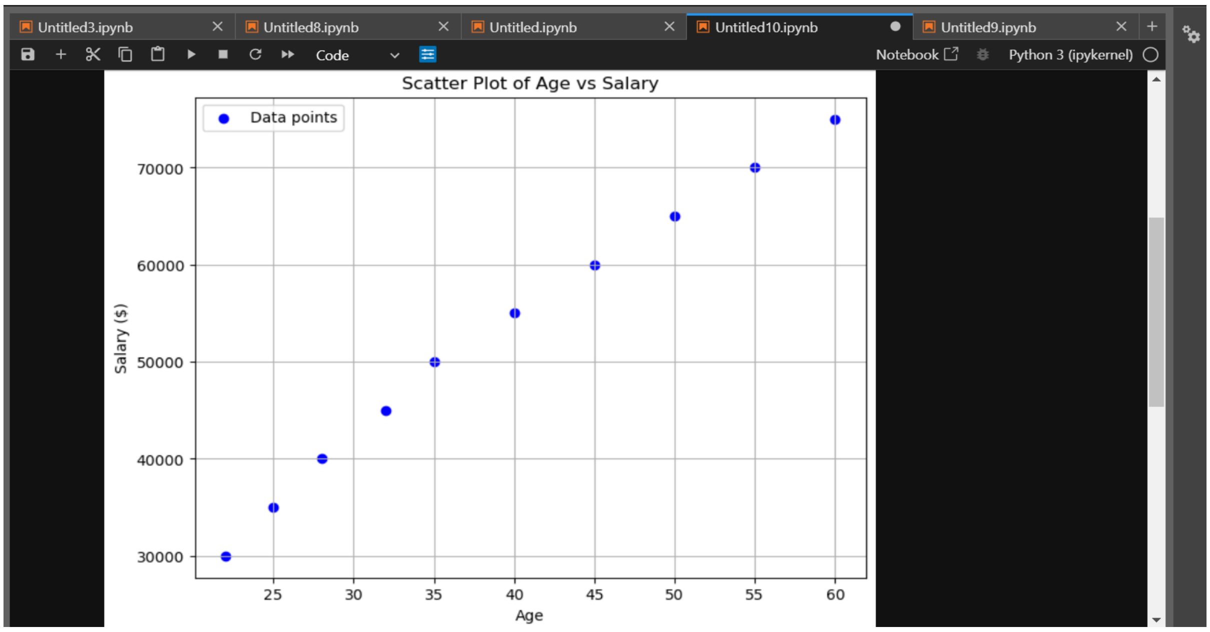
```
[ ]: import numpy as np
     import scipy.stats as stats
     import matplotlib.pyplot as plt
     age = np.array([22, 25, 28, 32, 35, 40, 45, 50, 55, 60])
     salary = np.array([30000, 35000, 40000, 45000, 50000, 55000, 60000, 65000, 70000, 75000])
```

```
[52]: correlation, _ = stats.pearsonr(age, salary)
      print(f"Pearson's correlation coefficient: {correlation:.2f}")
```

```
Pearson's correlation coefficient: 1.00
```

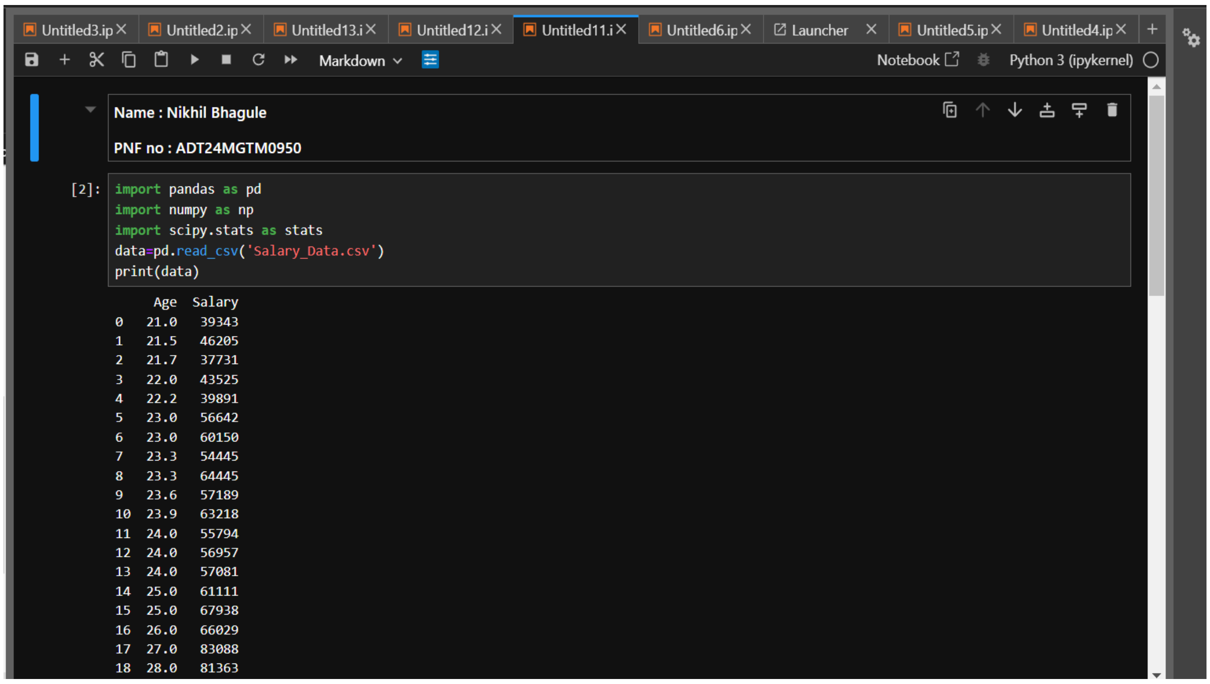


```
[53]: plt.figure(figsize=(8, 6))
      plt.scatter(age, salary, color='blue', label='Data points')
      plt.title("Scatter Plot of Age vs Salary")
      plt.xlabel("Age")
      plt.ylabel("Salary ($)")
      plt.grid(True)
      plt.legend()
      plt.show()
```



7.Question: Write a Python script to calculate Spearman's Rank Correlation for a given dataset.

Task: Load a dataset and compute the rank correlation using scipy.stats.

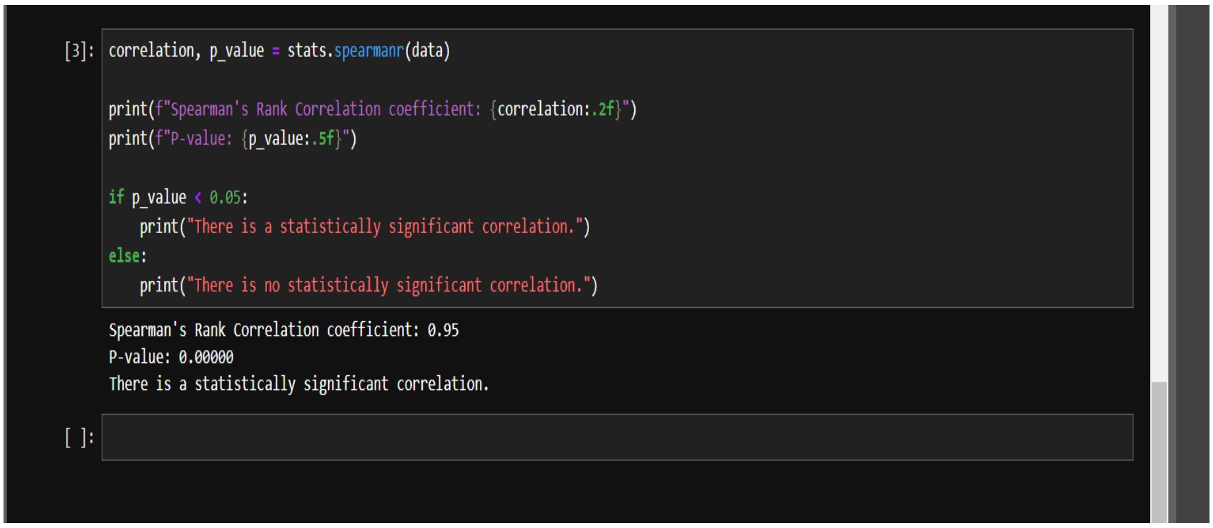


Name : Nikhil Bhagule

PNF no : ADT24MGTM0950

```
[2]: import pandas as pd
     import numpy as np
     import scipy.stats as stats
     data=pd.read_csv('Salary_Data.csv')
     print(data)
```

```
     Age  Salary
0   21.0   39343
1   21.5   46205
2   21.7   37731
3   22.0   43525
4   22.2   39891
5   23.0   56642
6   23.0   60150
7   23.3   54445
8   23.3   64445
9   23.6   57189
10  23.9   63218
11  24.0   55794
12  24.0   56957
13  24.0   57081
14  25.0   61111
15  25.0   67938
16  26.0   66029
17  27.0   83088
18  28.0   81363
```



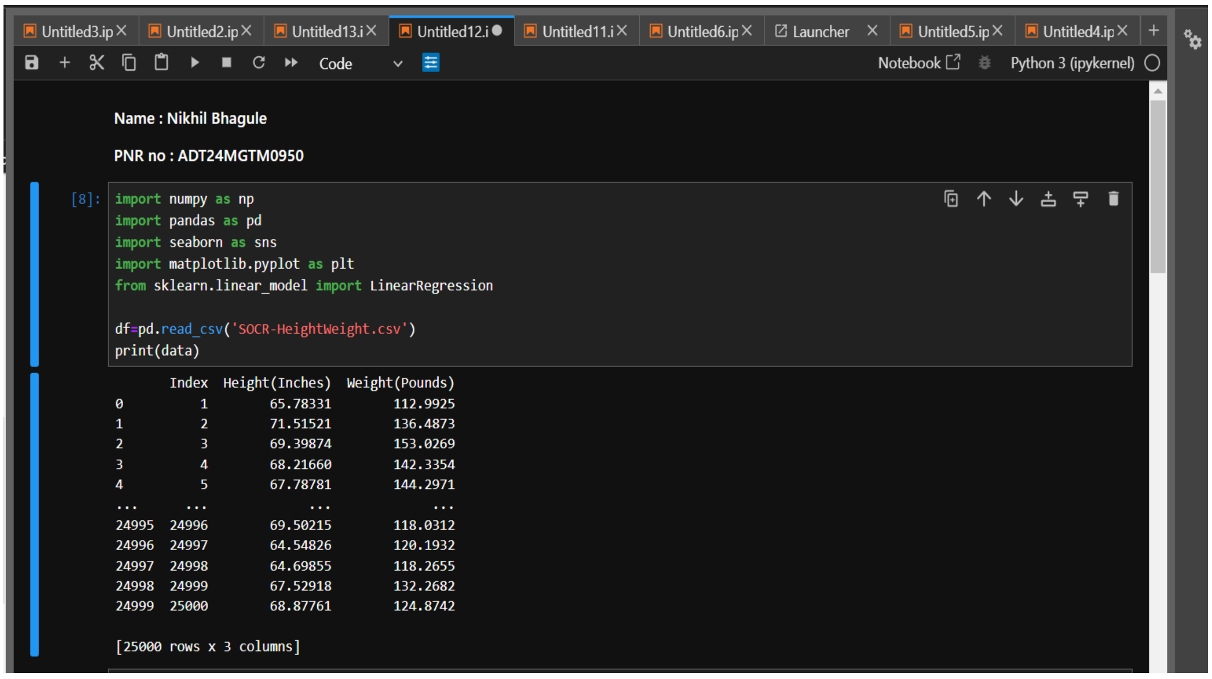
```
[3]: correlation, p_value = stats.spearmanr(data)

     print(f"Spearman's Rank Correlation coefficient: {correlation:.2f}")
     print(f"P-value: {p_value:.5f}")

     if p_value < 0.05:
         print("There is a statistically significant correlation.")
     else:
         print("There is no statistically significant correlation.")
```

```
Spearman's Rank Correlation coefficient: 0.95
P-value: 0.00000
There is a statistically significant correlation.
```

```
[ ]:
```

8.Question: Write a Python program to perform simple linear regression and plot the regression line between two variables (e.g., height and weight).

Task: Use sklearn for linear regression and visualize the regression line with seaborn.

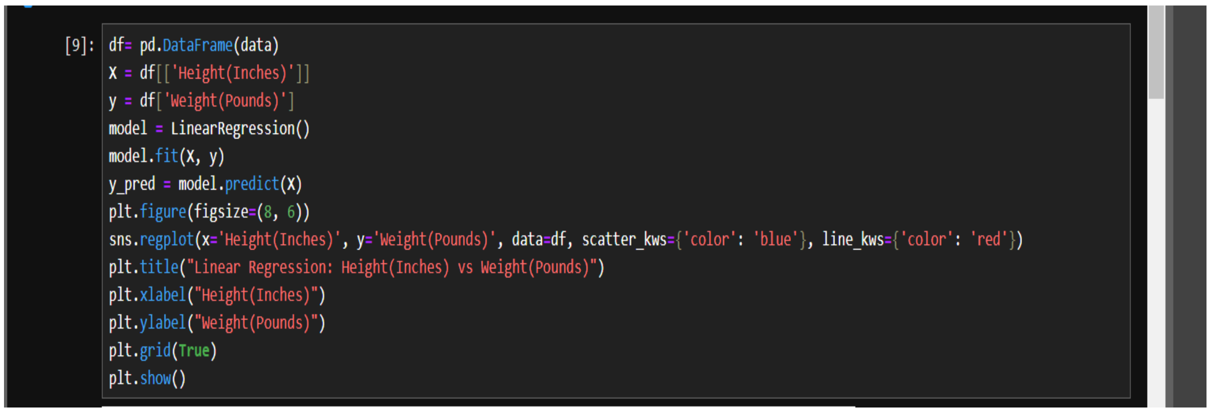


Name : Nikhil Bhagule

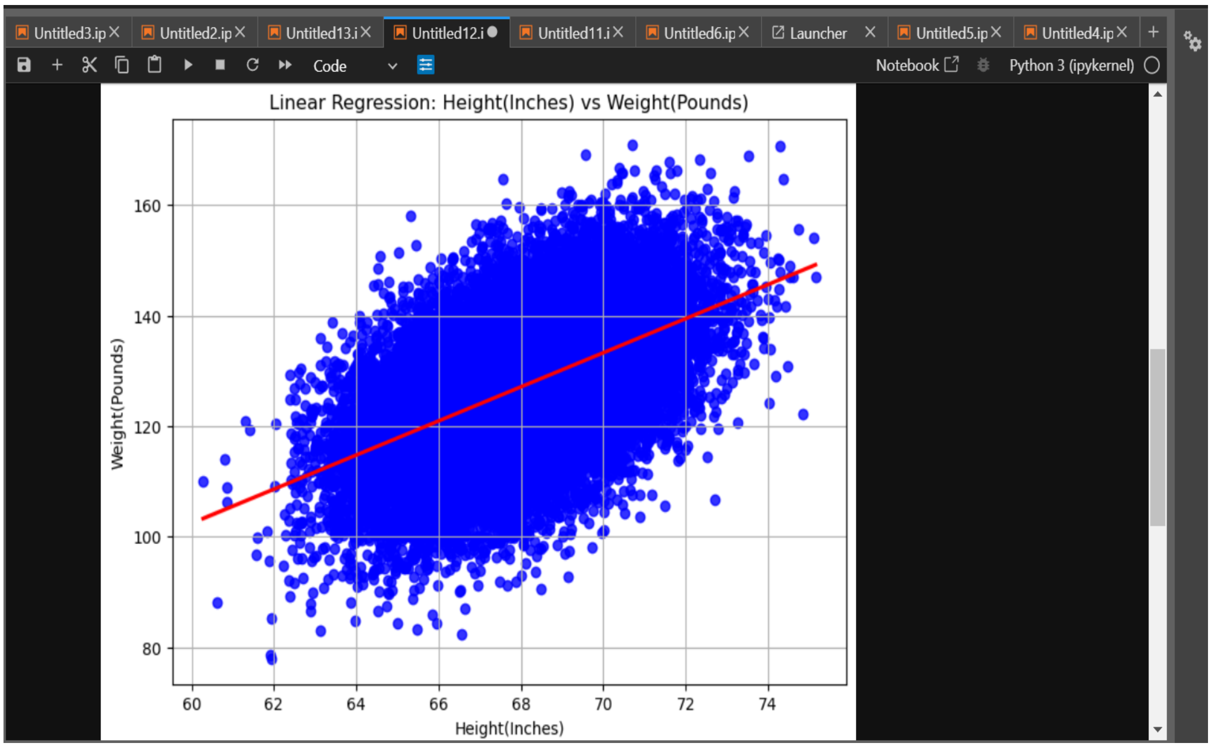
PNR no : ADT24MGTM0950

```
[8]: import numpy as np
     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
     from sklearn.linear_model import LinearRegression

     df=pd.read_csv('SOCR-HeightWeight.csv')
     print(data)
```

```
       Index  Height(Inches)  Weight(Pounds)
0          1        65.78331        112.9925
1          2        71.51521        136.4873
2          3        69.39874        153.0269
3          4        68.21660        142.3354
4          5        67.78781        144.2971
...      ...             ...             ...
24995  24996        69.50215        118.0312
24996  24997        64.54826        120.1932
24997  24998        64.69855        118.2655
24998  24999        67.52918        132.2682
24999  25000        68.87761        124.8742

[25000 rows x 3 columns]
```

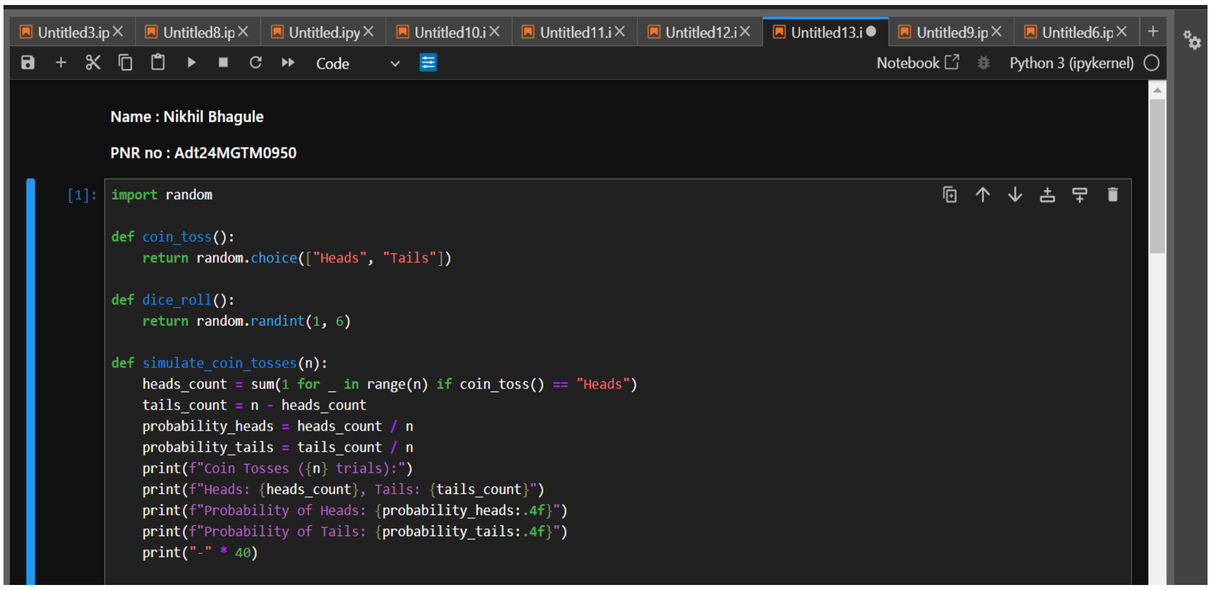


```
[9]: df= pd.DataFrame(data)
     X = df[['Height(Inches)']]
     y = df['Weight(Pounds)']
     model = LinearRegression()
     model.fit(X, y)
     y_pred = model.predict(X)
     plt.figure(figsize=(8, 6))
     sns.regplot(x='Height(Inches)', y='Weight(Pounds)', data=df, scatter_kws={'color': 'blue'}, line_kws={'color': 'red'})
     plt.title("Linear Regression: Height(Inches) vs Weight(Pounds)")
     plt.xlabel("Height(Inches)")
     plt.ylabel("Weight(Pounds)")
     plt.grid(True)
     plt.show()
```


Untitled3.ip
Untitled2.ip
Untitled13.i
Untitled12.i
Untitled11.i
Untitled6.ip
Launcher
Untitled5.ip
Untitled4.ip
Code
Notebook
Python 3 (ipykernel)
Linear Regression: Height(Inches) vs Weight(Pounds)
Weight(Pounds)
Height(Inches)
80
100
120
140
160
60
62
64
66
68
70
72
74

9.Question: Write a Python program to simulate basic probability concepts (e.g., coin toss, dice roll) and calculate the probability of certain events.

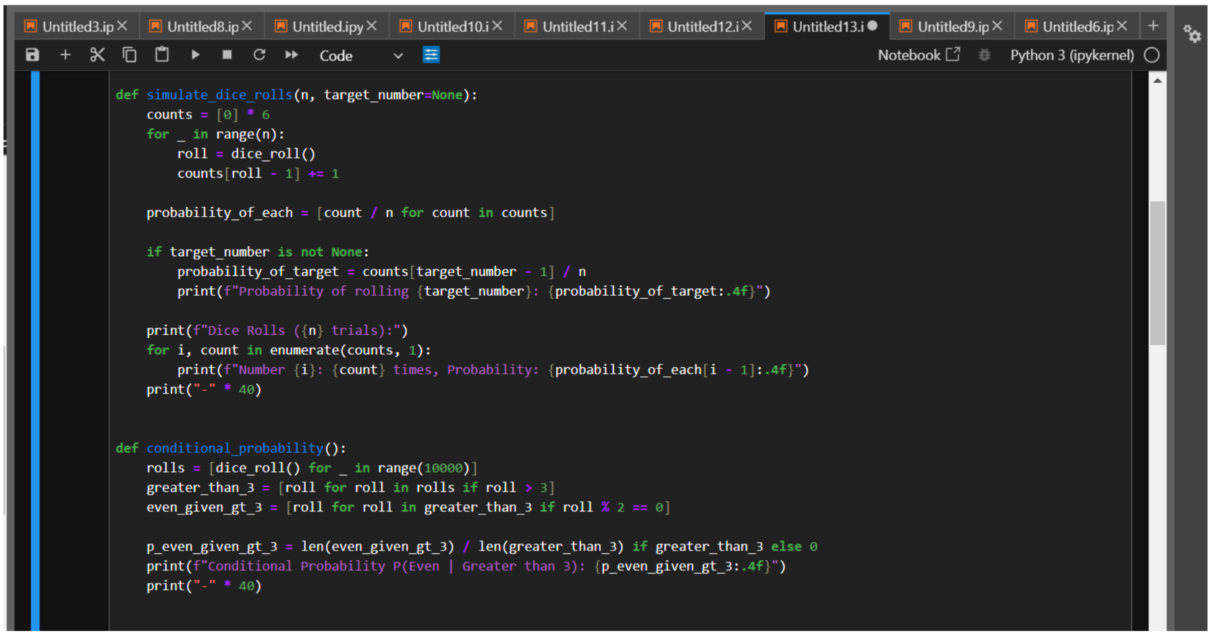
Task: Use random number generation and conditional probability in Python.



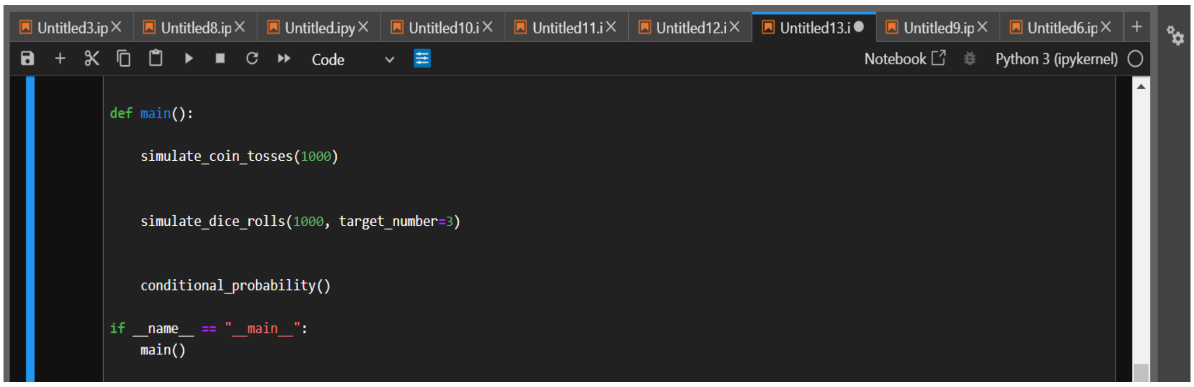
Name : Nikhil Bhagule

PNR no : Adt24MGTM0950

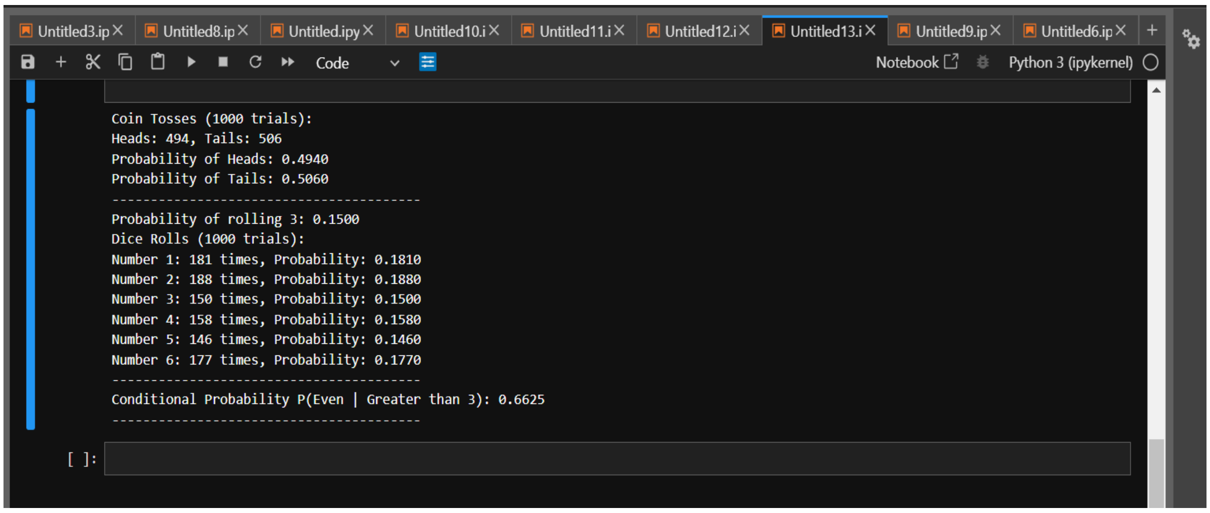
```
[1]: import random

def coin_toss():
    return random.choice(["Heads", "Tails"])

def dice_roll():
    return random.randint(1, 6)

def simulate_coin_tosses(n):
    heads_count = sum(1 for _ in range(n) if coin_toss() == "Heads")
    tails_count = n - heads_count
    probability_heads = heads_count / n
    probability_tails = tails_count / n
    print(f"Coin Tosses ({n} trials):")
    print(f"Heads: {heads_count}, Tails: {tails_count}")
    print(f"Probability of Heads: {probability_heads:.4f}")
    print(f"Probability of Tails: {probability_tails:.4f}")
    print("-" * 40)
```



```
def simulate_dice_rolls(n, target_number=None):
    counts = [0] * 6
    for _ in range(n):
        roll = dice_roll()
        counts[roll - 1] += 1

    probability_of_each = [count / n for count in counts]

    if target_number is not None:
        probability_of_target = counts[target_number - 1] / n
        print(f"Probability of rolling {target_number}: {probability_of_target:.4f}")

    print(f"Dice Rolls ({n} trials):")
    for i, count in enumerate(counts, 1):
        print(f"Number {i}: {count} times, Probability: {probability_of_each[i - 1]:.4f}")
    print("-" * 40)


def conditional_probability():
    rolls = [dice_roll() for _ in range(10000)]
    greater_than_3 = [roll for roll in rolls if roll > 3]
    even_given_gt_3 = [roll for roll in greater_than_3 if roll % 2 == 0]

    p_even_given_gt_3 = len(even_given_gt_3) / len(greater_than_3) if greater_than_3 else 0
    print(f"Conditional Probability P(Even | Greater than 3): {p_even_given_gt_3:.4f}")
    print("-" * 40)
```



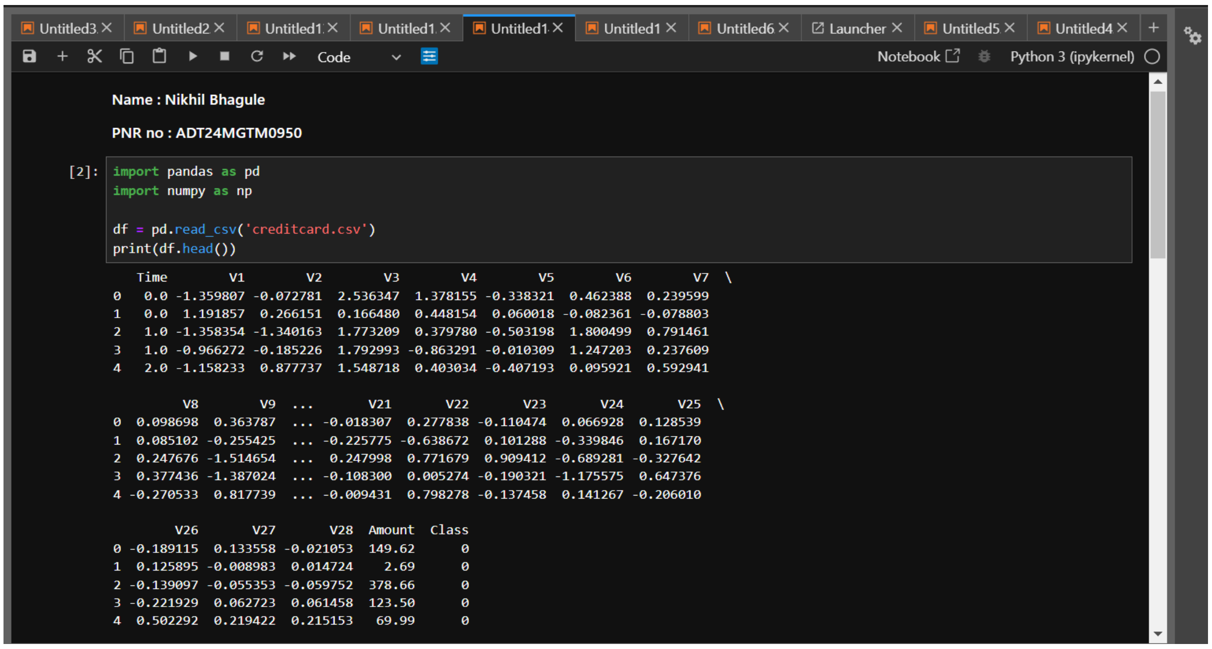
```
def main():

    simulate_coin_tosses(1000)


    simulate_dice_rolls(1000, target_number=3)


    conditional_probability()

if __name__ == "__main__":
    main()
```



```
Coin Tosses (1000 trials):
Heads: 494, Tails: 506
Probability of Heads: 0.4940
Probability of Tails: 0.5060
----------------------------------------
Probability of rolling 3: 0.1500
Dice Rolls (1000 trials):
Number 1: 181 times, Probability: 0.1810
Number 2: 188 times, Probability: 0.1880
Number 3: 150 times, Probability: 0.1500
Number 4: 158 times, Probability: 0.1580
Number 5: 146 times, Probability: 0.1460
Number 6: 177 times, Probability: 0.1770
----------------------------------------
Conditional Probability P(Even | Greater than 3): 0.6625
----------------------------------------
```

10.Question: Use Bayes Theorem to compute the posterior probability given prior probability and likelihoods.

Task: Implement Bayes Theorem in Python using a practical dataset.

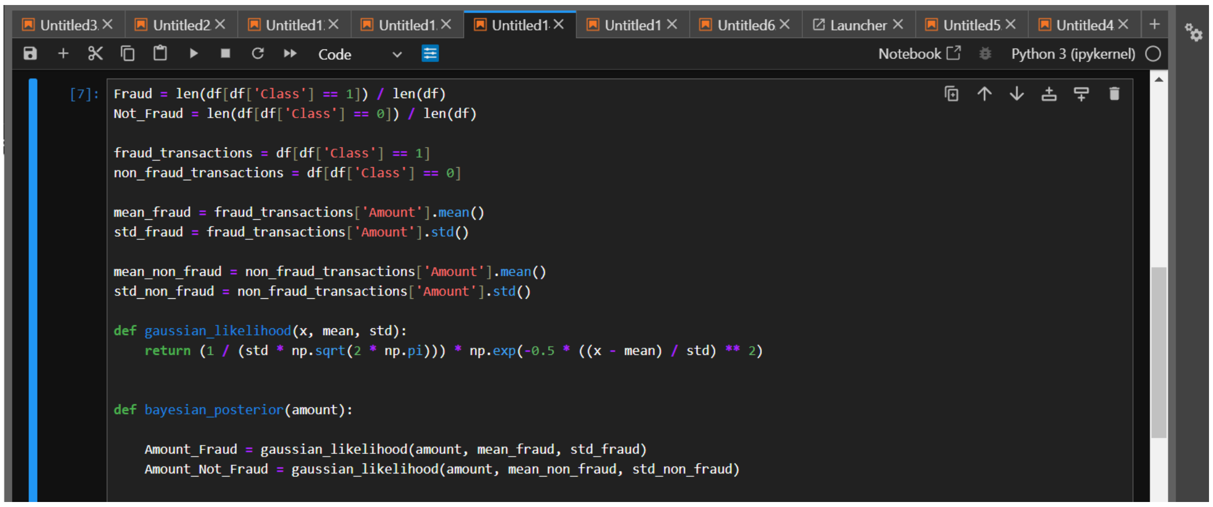


Name : Nikhil Bhagule

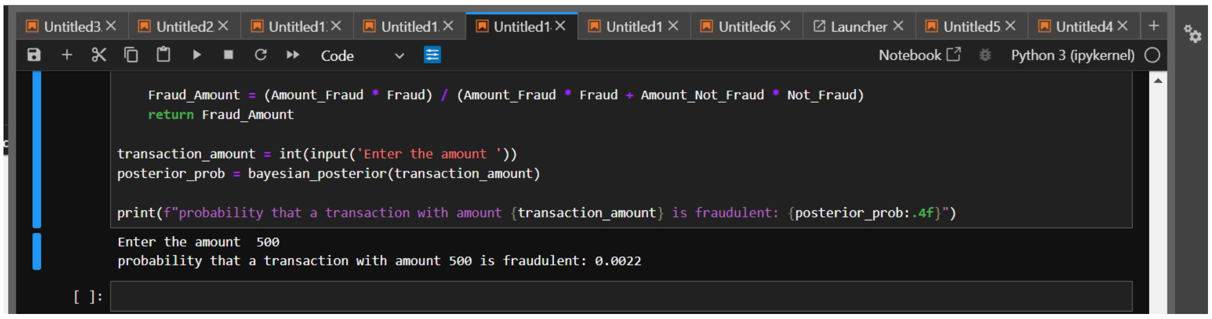
PNR no : ADT24MGTM0950

```
[2]: import pandas as pd
     import numpy as np

     df = pd.read_csv('creditcard.csv')
     print(df.head())
```

```
   Time        V1        V2        V3        V4        V5        V6        V7  \
0   0.0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4   2.0 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941

         V8        V9  ...       V21       V22       V23       V24       V25  \
0  0.098698  0.363787  ... -0.018307  0.277838 -0.110474  0.066928  0.128539
1  0.085102 -0.255425  ... -0.225775 -0.638672  0.101288 -0.339846  0.167170
2  0.247676 -1.514654  ...  0.247998  0.771679  0.909412 -0.689281 -0.327642
3  0.377436 -1.387024  ... -0.108300  0.005274 -0.190321 -1.175575  0.647376
4 -0.270533  0.817739  ... -0.009431  0.798278 -0.137458  0.141267 -0.206010

        V26       V27       V28  Amount  Class
0 -0.189115  0.133558 -0.021053  149.62      0
1  0.125895 -0.008983  0.014724    2.69      0
2 -0.139097 -0.055353 -0.059752  378.66      0
3 -0.221929  0.062723  0.061458  123.50      0
4  0.502292  0.219422  0.215153   69.99      0
```



```
[7]: Fraud = len(df[df['Class'] == 1]) / len(df)
     Not_Fraud = len(df[df['Class'] == 0]) / len(df)

     fraud_transactions = df[df['Class'] == 1]
     non_fraud_transactions = df[df['Class'] == 0]

     mean_fraud = fraud_transactions['Amount'].mean()
     std_fraud = fraud_transactions['Amount'].std()

     mean_non_fraud = non_fraud_transactions['Amount'].mean()
     std_non_fraud = non_fraud_transactions['Amount'].std()

     def gaussian_likelihood(x, mean, std):
         return (1 / (std * np.sqrt(2 * np.pi))) * np.exp(-0.5 * ((x - mean) / std) ** 2)


     def bayesian_posterior(amount):

         Amount_Fraud = gaussian_likelihood(amount, mean_fraud, std_fraud)
         Amount_Not_Fraud = gaussian_likelihood(amount, mean_non_fraud, std_non_fraud)
```



```
         Fraud_Amount = (Amount_Fraud * Fraud) / (Amount_Fraud * Fraud + Amount_Not_Fraud * Not_Fraud)
         return Fraud_Amount

     transaction_amount = int(input('Enter the amount '))
     posterior_prob = bayesian_posterior(transaction_amount)

     print(f"probability that a transaction with amount {transaction_amount} is fraudulent: {posterior_prob:.4f}")
```

```
Enter the amount  500
probability that a transaction with amount 500 is fraudulent: 0.0022
```

```
[ ]:
```