IE-416
Lab-1

Group Members:
202201492 - Shravan N. Makwana
202201455 - Parth Tolani

Github Link:
https://github.com/Hepatitous/IE416/blob/main/Lab-1/IE416_Lab1.ipynb

Google Collab
Link:https://colab.research.google.com/drive/1I_n3LgiZ85sK5Xh-FWjvWrac-WQzCA5K#scrollTo=FO05HiPBZL-L

Write a function that gives number of days of given year. Input : 1990 Input : 2044 Output : 365 Output : 366

```
#Q1
year = int(input("Enter year: "))

if(year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        print("366")
else:
        print("365")
```

```
Enter year: 2044
366
```

A Simple code to check whether a year is a leap year or not, giving the number of days occordingly

Count the frequency of each character in a string and store it in a dictionary. An example is given below. Input: ‘adcbbdaacd’ Output: {‘a’: 3, ‘b’: 2, ‘c’: 2, ‘d’: 3}

```
#Q2
string = input("Enter the string: ")
freq = {}

for char in string:
    if char in freq:
        freq[char] += 1
    else:
        freq[char] = 1
print(freq)
```

```
Enter the string: qwertwertertrtt
{'q': 1, 'w': 2, 'e': 3, 'r': 4, 't': 5}
```

A Simple code to count the frequency of occurence of every character in a string

Write a program to remove duplicates from a list but keep the first occurrence of each element. Input: [1, 2, 3, 4, 2, 3, 5, 6, 1, 4] Output: [1, 2, 3, 4, 5, 6]

```
#Q3
def remove_dupes(lst):
    freq = {}
    newlst = []
    for x in range(len(lst)):
        if lst[x] not in freq:
            newlst.append(lst[x])
            freq[lst[x]] = 1

    return newlst
```

```
#Example
mainlst = [3,3,2,1,3,4,5,6,54,3,2,1,2,434,4,3]
print(remove_dupes(mainlst))
```

```
[1, 4, 7, 2, 3, 5]
```

This code uses a new list to pick out every number from the original list only once, hence removing duplicates

Write a program to sort a stack using only another stack (no other data structures like arrays or linked lists). Input: stack = [9, 5, 1, 3] Output: stack = [1, 3, 5, 9]

```
#Q4
def sort_stack(stack):
    sorted = []
    while stack:
        temp = stack.pop()
        while sorted and sorted[-1] > temp:
          stack.append(sorted.pop())

        sorted.append(temp)

    return sorted

#Example
stack = [4,5,3,2,6,8,7,1,9,10]
print(sort_stack(stack))
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

A temporary stack is used to sort the given stack where the final sorted values appear in the temp stack, hence it is returned back

Q5. Make a module “pascal.py” with function “pascalTriangle(numOfRows)” and import into “main.py”.

```
#Q5(a)
%%writefile pascal.py

def pascalTriangle(n):
    triangle = [[1] * (i + 1) for i in range(n)]
    for i in range(2, n):
        for j in range(1, i):
            triangle[i][j] = triangle[i - 1][j - 1] + triangle[i - 1]
[j]
    return triangle
```

```
Writing pascal.py
```

Used to create the file pascal.py

```
#Q5(b)
import pascal as p

n = int(input("Enter Number of Rows: "))
print(p.pascalTriangle(n))
```

```
Enter Number of Rows: 5
[[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1]]
```

Acts as the main function to send user input into the function inside the file pascal.py which is impored in the first line

Create a 6x6 matrix with random values and: Replace all values greater than 0.5 with 1, and all others with 0. Extract a 3x3 submatrix starting from index (2, 2) and calculate its mean.

```
#Q6
import numpy as np

mat = np.random.rand(6,6)

for x in range(6):
  for y in range(6):
      if(mat[x][y]>0.5):
        mat[x][y] = 1
      else:
        mat[x][y] = 0

subsum = 0.

for x in range(3):
  for y in range(3):
    subsum+=mat[x+2][y+2]

subsum/=9

print(subsum)
```

```
0.4444444444444444
```

A random matrix converted to binary to then find the mean of these values, highest probability of occurence are 0.44,0.55,0.66, other values are rare

Q7. Array Reshaping: Create a 1D array with 16 elements. Reshape it into a 4x4 matrix. Flatten a 3x3x3 array into a 1D array. Reshape a matrix into a new shape without changing its data.

```
#Q7
import numpy as np
array_1D = np.arange(16)
```

```
print("Original 1D Array:\n\n", array_1D)

matrix_4x4 = array_1D.reshape(4, 4)
print("\nReshaped 4x4 Matrix:\n\n", matrix_4x4)

array_3D = np.random.rand(3, 3, 3)
print("\nOriginal 3x3x3 Array:\n\n", array_3D)

flattened_array = array_3D.flatten()
print("\nFlattened 3×3×3 Array:\n\n", flattened_array)

reshaped_matrix = matrix_4x4.reshape(2, 8)
print("\nReshaping the 4x4 to a 2×8 Matrix:\n\n", reshaped_matrix)
```

```
Original 1D Array:

 [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]

Reshaped 4x4 Matrix:

 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]

Original 3x3x3 Array:

 [[[0.09973711 0.45358507 0.10633143]
  [0.26046779 0.87631838 0.08208819]
  [0.11678378 0.86872936 0.60906892]]

 [[0.14031595 0.35804945 0.81188299]
  [0.83956564 0.55815207 0.38099983]
  [0.83435146 0.84375988 0.76057555]]

 [[0.03858687 0.27210754 0.30757425]
  [0.14884857 0.10287664 0.17332484]
  [0.59643081 0.58246158 0.07225217]]]

Flattened 3×3×3 Array:

 [0.09973711 0.45358507 0.10633143 0.26046779 0.87631838 0.08208819
 0.11678378 0.86872936 0.60906892 0.14031595 0.35804945 0.81188299
 0.83956564 0.55815207 0.38099983 0.83435146 0.84375988 0.76057555
 0.03858687 0.27210754 0.30757425 0.14884857 0.10287664 0.17332484
 0.59643081 0.58246158 0.07225217]

Reshaping the 4x4 to a 2×8 Matrix:

 [[ 0  1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14 15]]
```

Usage of matrix and array related functions of the numpy library

Write a recursive function Fibonacci_sum(n) to calaculate the sum of first n numbers in Fibonacci series 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89.

```
#Q8
def fibbonaci(n,last,secondlast):
    if n==0:
      return last + secondlast
    else:
      return fibbonaci(n-1,last+secondlast,last)

n = int(input("Enter Number of terms: "))
if(n==0):
  print("0")
elif(n==1):
  print("1")
else:
  print(fibbonaci(n-2,0,1))
```

```
Enter Number of terms: 12
89
```

Simple recursive code to find the number at a position in the fibbonaci sequence

Define a function get_value_from_dict that takes a dictionary and a key as parameters. If the key is not present in the dictionary, the function should raise a KeyError with a custom error message. Write a main function that calls get_value_from_dict with a dictionary and user-provided key. Handle KeyError and display a user-friendly message if the key is not found.

```
#Q9
def get_value_from_dict(dic,key):
  if key in dic:
    return dic[key]
  else:
    return f"Key {key} not found"


def main():
  dic = {'x' : 5 , 'y' : 10 , 'z' : 3}
  key = input("Enter a key: ")
  print(get_value_from_dict(dic,key))

main()
```

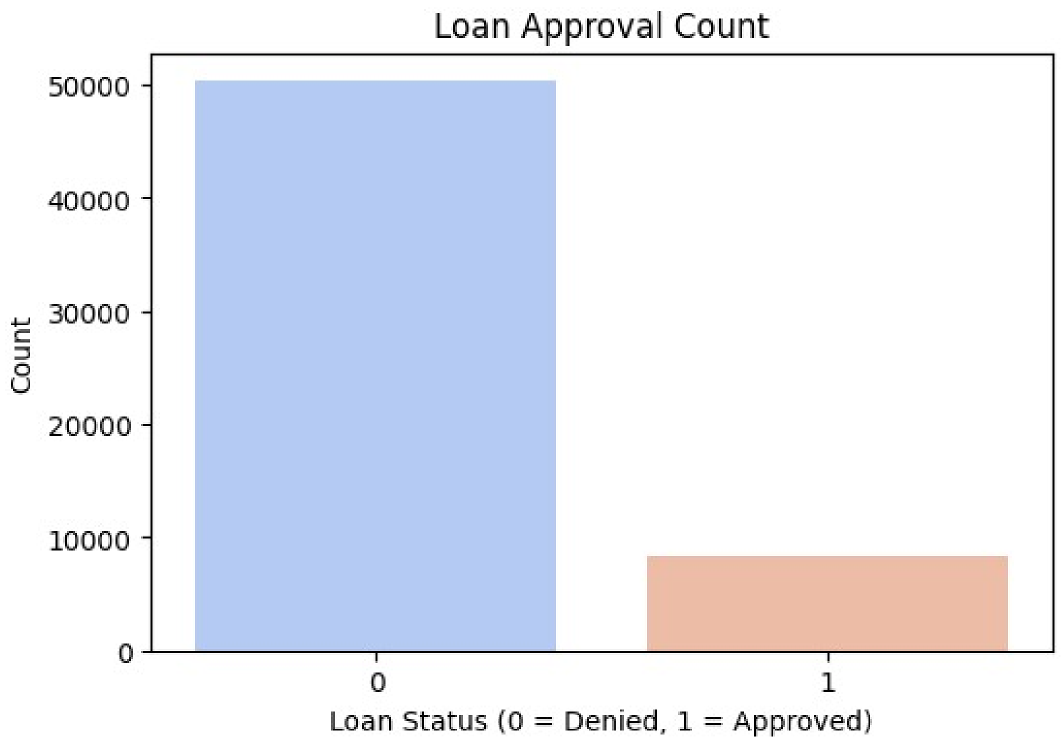
```
Enter a key: d
Key d not present in the dictionary
```

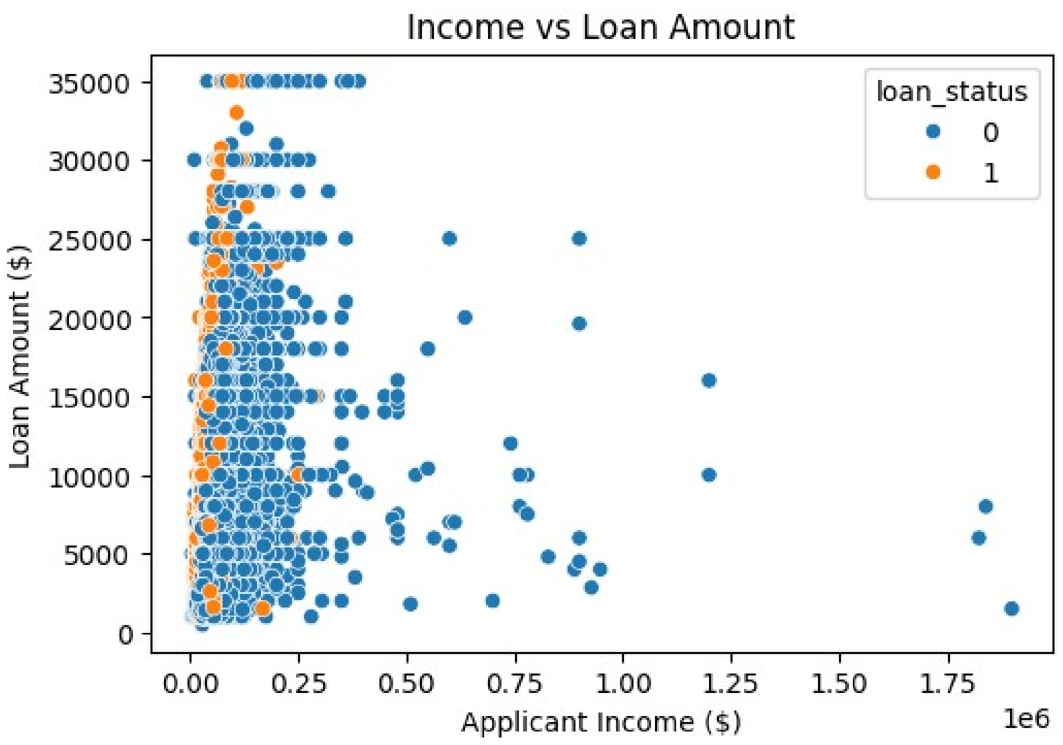
Pre defined dictionary where user input is searched through it, in case it is not found an error message is shown

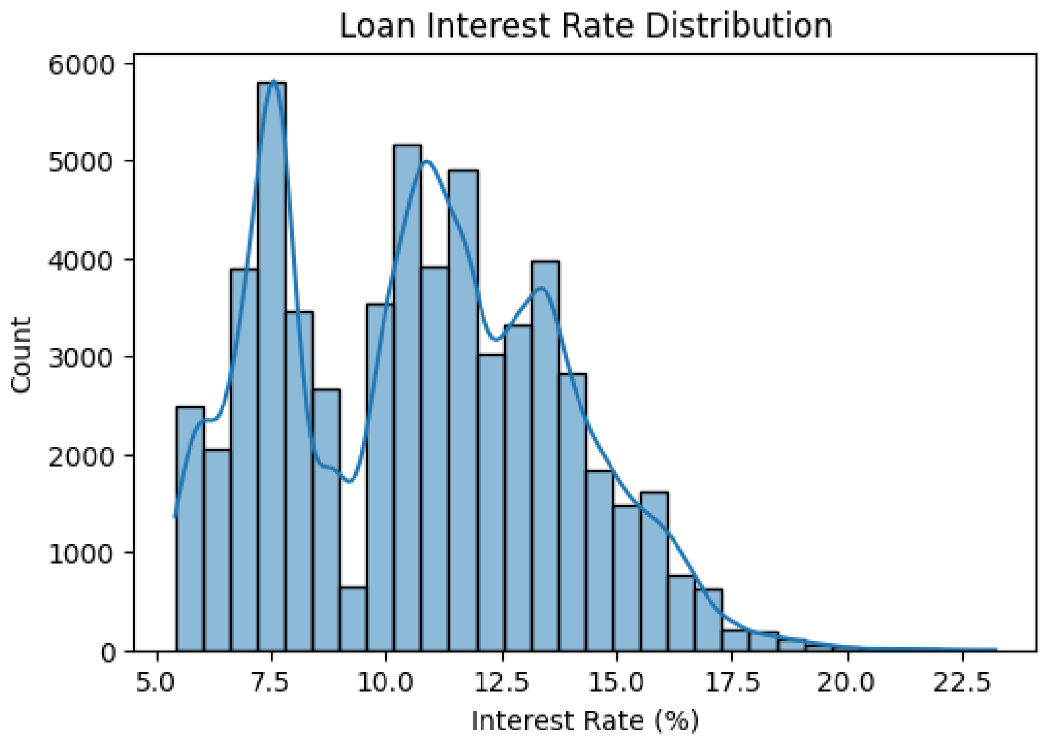
Using the following dataset, visualize the data with the maximum number of visualization tools available in Python. Create a variety of plots and charts, including but not limited to bar charts, pie charts, line graphs, scatter plots, histograms, and heatmaps. Use libraries such as matplotlib, seaborn, and plotly to explore different ways of presenting the data. Provide clear titles, labels, and legends to enhance the readability of your visualizations.

```python
#Q10
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from google.colab import drive
drive.mount('/content/drive')

file_path = '/content/drive/MyDrive/Loan_train.csv'
# Load dataset
df = pd.read_csv("drive/MyDrive/Loan_train.csv")

# Convert loan_status to categorical type for better plotting
df["loan_status"] = df["loan_status"].astype(str)

# 1. Loan Approval Rate (Bar Chart)
plt.figure(figsize=(6,4))
sns.countplot(x="loan_status", data=df, palette="coolwarm")
plt.title("Loan Approval Count")
plt.xlabel("Loan Status (0 = Denied, 1 = Approved)")
plt.ylabel("Count")
plt.show()

# 2. Income vs Loan Amount (Scatter Plot)
plt.figure(figsize=(6,4))
sns.scatterplot(x=df["person_income"], y=df["loan_amnt"],
hue=df["loan_status"])
plt.title("Income vs Loan Amount")
plt.xlabel("Applicant Income ($)")
plt.ylabel("Loan Amount ($)")
plt.show()

# 3. Interest Rate Distribution (Histogram)
plt.figure(figsize=(6,4))
sns.histplot(df["loan_int_rate"], bins=30, kde=True)
plt.title("Loan Interest Rate Distribution")
plt.xlabel("Interest Rate (%)")
plt.show()

# 4. Loan Intent Breakdown (Bar Chart)
plt.figure(figsize=(8,5))
sns.countplot(x="loan_intent", data=df, palette="viridis",
order=df["loan_intent"].value_counts().index)
plt.title("Loan Intent Distribution")
plt.xlabel("Loan Purpose")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()

# 5. Loan Intent Distribution (Pie Chart)
plt.figure(figsize=(6,6))
df["loan_intent"].value_counts().plot.pie(autopct="%1.1f%%",
startangle=90, cmap="coolwarm",
explode=[0.05]*len(df["loan_intent"].unique()))
plt.title("Loan Intent Distribution (Pie Chart)")
plt.ylabel("")  # Hide y-label for better visualization
plt.show()
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).

<ipython-input-10-9a70b907c93a>:18: 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.countplot(x="loan_status", data=df, palette="coolwarm")
```

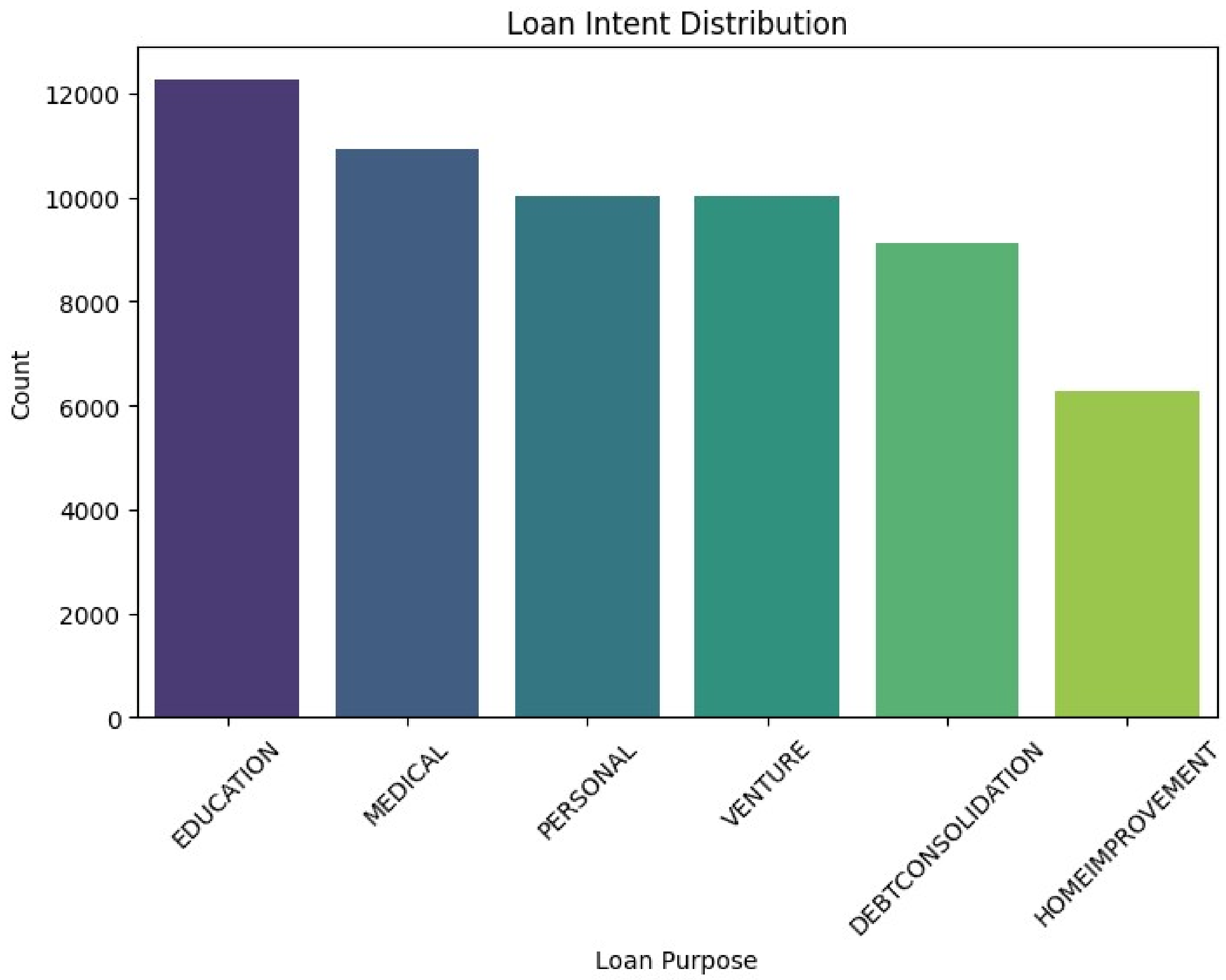


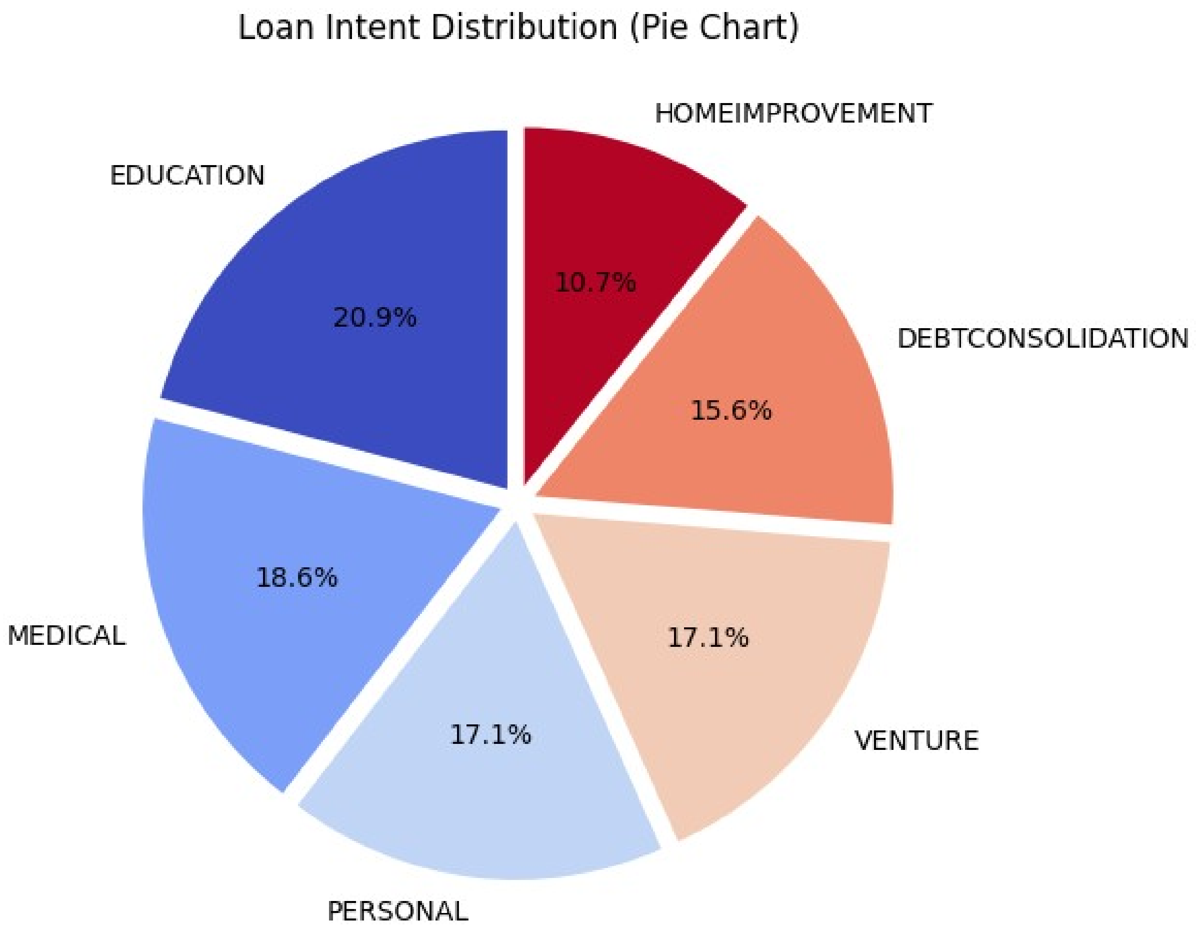




```
<ipython-input-10-9a70b907c93a>:41: FutureWarning:

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.countplot(x="loan_intent", data=df, palette="viridis",
order=df["loan_intent"].value_counts().index)
```





drive.mount('/content/drive') → Connects Google Drive to Colab. file_path = '/content/drive/MyDrive/Loan_train.csv' → Specifies the dataset's location. pd.read_csv("drive/MyDrive/Loan_train.csv") → Reads the dataset into a Pandas DataFrame. This allows seamless data access for visualization. The matplotlib.pyplot and seaborn are used to visualize the data given in Loan_train where various commands plt.figure, plt.title etc are used to finish this task. plt.figure(figsize=(6,4)) Creates a figure of size 6x4 inches. plt.title("Title Here") plt.xlabel("X-axis Label") plt.ylabel("Y-axis Label") Adds a title and axis labels to the plot. plt.show() Displays the plot on the screen. sns.scatterplot(x=df["X"], y=df["Y"]) Creates a scatter plot.

Data Visualizations & Insights

1. Loan Approval Rates (Bar Chart) : Displays the count of approved (1) and denied (0) loans.
2. Income vs. Loan Amount (Scatter Plot) : Shows the relationship between applicant income and loan amount.
3. Interest Rate Distribution (Histogram) : Illustrates how loan interest rates are distributed across applicants.

4. Loan Intent Breakdown (Bar Chart) : Displays the most common loan purposes (EDUCATION, MEDICAL, etc.).
5. Loan Intent Distribution (Pie Chart) : Shows the proportion of loans taken for different purposes.