

Bank Loan Analysis

By Cwen Fernandes

This analysis gives us an understanding of key factors or variables when assigning personal loans to customers.

Banks consider and evaluate every loan application based on merits. They check the creditworthiness of every individual or entity before assigning the loan. This helps to determine the level of risk that they are submitting themselves by lending to the entity or individual.

Usually, clients with high risk are less desirable since they have a higher likelihood of defaulting on their loan obligations. This makes clients with low risk more likely to get their loan applications approved.

In summary, banks checks credit repayment history, the conduct of the client, financial solvency, the client's reputation, and the ability to work with the amount granted as a loan.

1. Importing Libraries

```
In [89]: #Importing Libraries  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns
```

2. Reading File

```
In [90]: #Reading File
df = pd.read_csv("Bank_Personal_Loan_Modelling.csv")
df
```

Out[90]:

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan
0	1	25	1	49	91107	4	1.6	1	0	
1	2	45	19	34	90089	3	1.5	1	0	
2	3	39	15	11	94720	1	1.0	1	0	
3	4	35	9	100	94112	1	2.7	2	0	
4	5	35	8	45	91330	4	1.0	2	0	
...
4995	4996	29	3	40	92697	1	1.9	3	0	
4996	4997	30	4	15	92037	4	0.4	1	85	
4997	4998	63	39	24	93023	2	0.3	3	0	
4998	4999	65	40	49	90034	3	0.5	2	0	
4999	5000	28	4	83	92612	3	0.8	1	0	

5000 rows × 14 columns



```
In [91]: #Top 5 Values
df.head(5)
```

Out[91]:

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	S
0	1	25	1	49	91107	4	1.6	1	0	0	
1	2	45	19	34	90089	3	1.5	1	0	0	
2	3	39	15	11	94720	1	1.0	1	0	0	
3	4	35	9	100	94112	1	2.7	2	0	0	
4	5	35	8	45	91330	4	1.0	2	0	0	



In [92]: `#Bottom 5 values`
`df.tail(5)`

Out[92]:

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan
4995	4996	29	3	40	92697	1	1.9	3	0	
4996	4997	30	4	15	92037	4	0.4	1	85	
4997	4998	63	39	24	93023	2	0.3	3	0	
4998	4999	65	40	49	90034	3	0.5	2	0	
4999	5000	28	4	83	92612	3	0.8	1	0	

3. Exploratory Data Analysis

In [93]: `#shape of data`
`df.shape`

Out[93]: (5000, 14)

In [94]: `#Checking Null Values`
`df.isnull().sum()`

Out[94]:

ID	0
Age	0
Experience	0
Income	0
ZIP Code	0
Family	0
CCAvg	0
Education	0
Mortgage	0
Personal Loan	0
Securities Account	0
CD Account	0
Online	0
CreditCard	0
dtype:	int64

In [95]: `#Columns Available`
`df.columns`

Out[95]: Index(['ID', 'Age', 'Experience', 'Income', 'ZIP Code', 'Family', 'CCAvg',
'Education', 'Mortgage', 'Personal Loan', 'Securities Account',
'CD Account', 'Online', 'CreditCard'],
dtype='object')

In [96]: `df[['ID', 'ZIP Code']]`

Out[96]:

	ID	ZIP Code
0	1	91107
1	2	90089
2	3	94720
3	4	94112
4	5	91330
...
4995	4996	92697
4996	4997	92037
4997	4998	93023
4998	4999	90034
4999	5000	92612

5000 rows × 2 columns

In [97]: `#Dropping unnecessary column`
`df.drop(['ID', 'ZIP Code'], axis = 1, inplace = True)`

In [98]: `df.columns`

Out[98]: Index(['Age', 'Experience', 'Income', 'Family', 'CCAvg', 'Education',
'Mortgage', 'Personal Loan', 'Securities Account', 'CD Account',
'Online', 'CreditCard'],
dtype='object')

In [99]: `#Data Types`
`df.dtypes`

Out[99]:

Age	int64
Experience	int64
Income	int64
Family	int64
CCAvg	float64
Education	int64
Mortgage	int64
Personal Loan	int64
Securities Account	int64
CD Account	int64
Online	int64
CreditCard	int64
dtype:	object

In [100]:

▶

```
#Summary Data
#Generate summary statistics
summary_stats = df[['Age', 'Experience', 'Income', 'Family', 'Education']]
summary_stats
```

Out[100]:

	Age	Experience	Income	Family	Education
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000
mean	45.338400	20.104600	73.774200	2.396400	1.881000
std	11.463166	11.467954	46.033729	1.147663	0.839869
min	23.000000	-3.000000	8.000000	1.000000	1.000000
25%	35.000000	10.000000	39.000000	1.000000	1.000000
50%	45.000000	20.000000	64.000000	2.000000	2.000000
75%	55.000000	30.000000	98.000000	3.000000	3.000000
max	67.000000	43.000000	224.000000	4.000000	3.000000

Visualizing Summary Statistics

```
In [101]: #Subplots of the Summary

#Create subplots for multiple visualizations
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12,10))

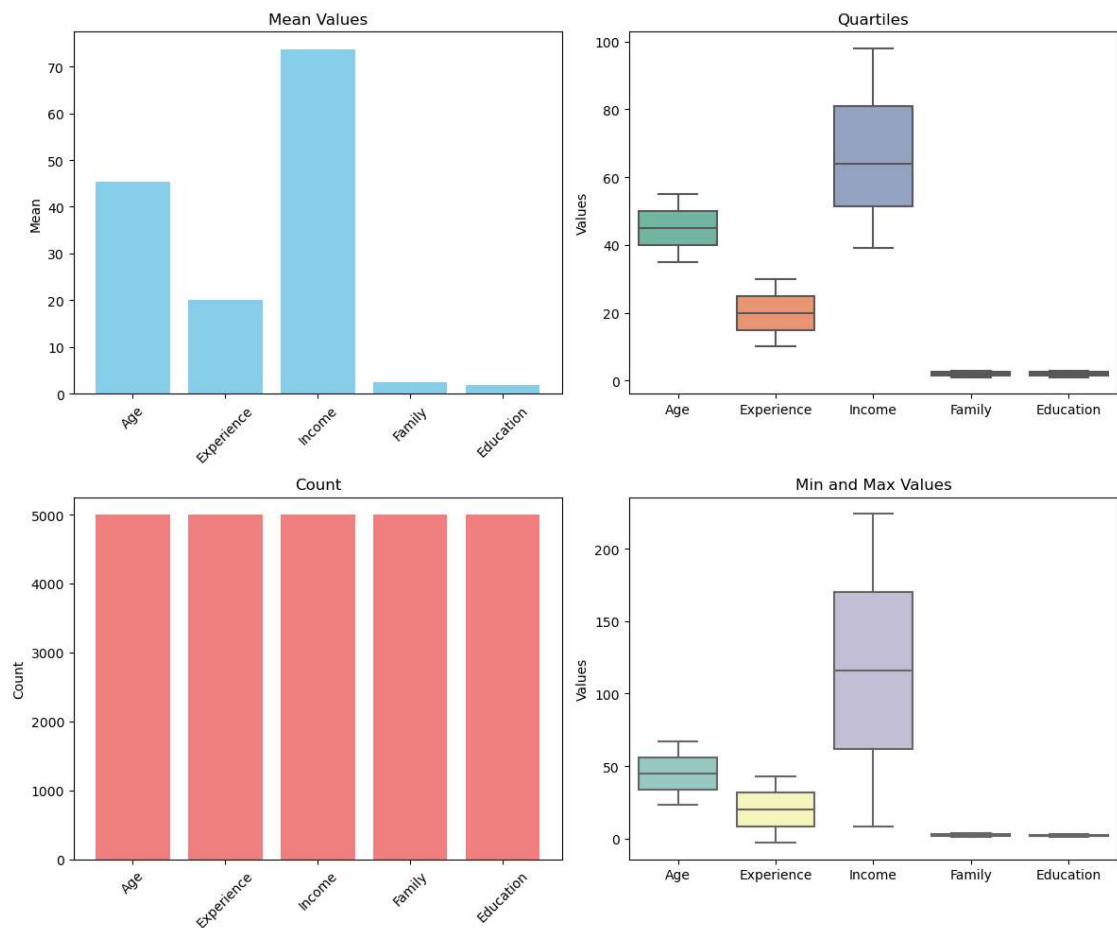
#Bar plot for mean values
axes[0, 0].bar(summary_stats.columns, summary_stats.loc['mean'], color='sk')
axes[0, 0].set_title("Mean Values")
axes[0, 0].set_ylabel("Mean")
axes[0, 0].tick_params(axis='x', rotation=45)

#Box plot for quartiles
sns.boxplot(data=summary_stats.loc[['25%', '50%', '75%']], ax=axes[0,1], palette='magma')
axes[0, 1].set_title('Quartiles')
axes[0, 1].set_ylabel('Values')

#Histogram for count
axes[1, 0].bar(summary_stats.columns, summary_stats.loc['count'], color='lightcoral')
axes[1, 0].set_title('Count')
axes[1, 0].set_ylabel('Count')
axes[1, 0].tick_params(axis='x', rotation=45)

#Box plot for min and max
sns.boxplot(data=summary_stats.loc[['min', 'max']], ax=axes[1, 1], palette='magma')
axes[1, 1].set_title('Min and Max Values')
axes[1, 1].set_ylabel('Values')

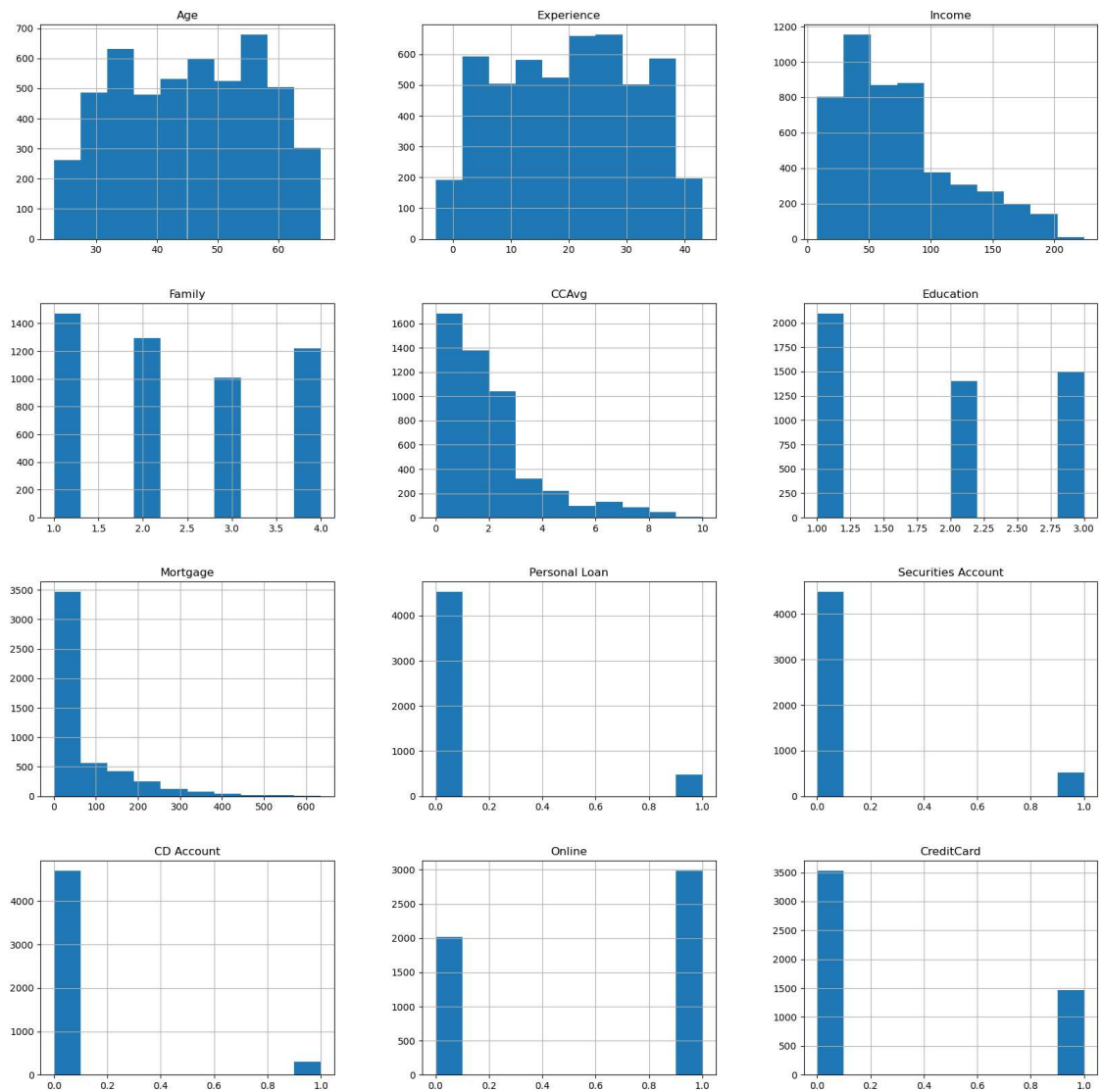
#Adjust layout and show plots
plt.tight_layout()
plt.show()
```



In [102]: `#Calculating the skewness of numerical columns`
`df.skew()`

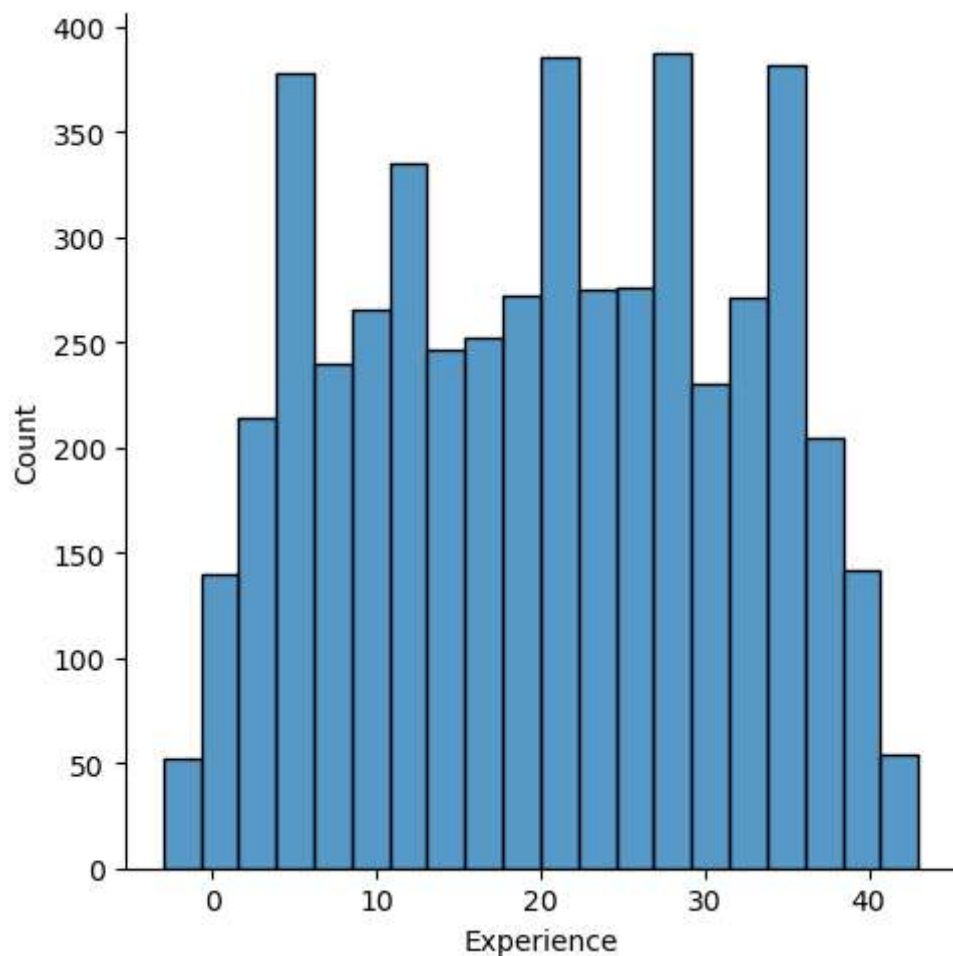
```
Out[102]: Age          -0.029341
Experience -0.026325
Income     0.841339
Family     0.155221
CCAvg      1.598443
Education  0.227093
Mortgage   2.104002
Personal Loan 2.743607
Securities Account 2.588268
CD Account  3.691714
Online     -0.394785
CreditCard 0.904589
dtype: float64
```

```
In [103]: ▶ #Creating histograms for each column  
df.hist(figsize = (20, 20))  
plt.show()
```




```
In [104]: ▶ #Creating a displot for 'Experience Column'
plt.figure(figsize=(16, 2))
sns.displot(df['Experience'])
plt.show()
```

<Figure size 1600x200 with 0 Axes>



Most data available are of 20 to 30 years old experience person. We can say that people with 20 to 30 years of experience can take personal loan.

```
In [105]: ▶ #Checking mean of Experience data
df['Experience'].mean()
```

Out[105]: 20.1046

```
In [106]: ▶ #Negative data in Experience column  
negative_Exp = df[df['Experience']<0]  
negative_Exp
```

Out[106]:

	Age	Experience	Income	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account
89	25	-1	113	4	2.30	3	0	0	0
226	24	-1	39	2	1.70	2	0	0	0
315	24	-2	51	3	0.30	3	0	0	0
451	28	-2	48	2	1.75	3	89	0	0
524	24	-1	75	4	0.20	1	0	0	0
536	25	-1	43	3	2.40	2	176	0	0
540	25	-1	109	4	2.30	3	314	0	0
576	25	-1	48	3	0.30	3	0	0	0
583	24	-1	38	2	1.70	2	0	0	0
597	24	-2	125	2	7.20	1	0	0	1
649	25	-1	82	4	2.10	3	0	0	0
670	23	-1	61	4	2.60	1	239	0	0
686	24	-1	38	4	0.60	2	0	0	0
793	24	-2	150	2	2.00	1	0	0	0
889	24	-2	82	2	1.60	3	0	0	0
909	23	-1	149	1	6.33	1	305	0	0
1173	24	-1	35	2	1.70	2	0	0	0
1428	25	-1	21	4	0.40	1	90	0	0
1522	25	-1	101	4	2.30	3	256	0	0
1905	25	-1	112	2	2.00	1	241	0	0
2102	25	-1	81	2	1.60	3	0	0	0
2430	23	-1	73	4	2.60	1	0	0	0
2466	24	-2	80	2	1.60	3	0	0	0
2545	25	-1	39	3	2.40	2	0	0	0
2618	23	-3	55	3	2.40	2	145	0	0
2717	23	-2	45	4	0.60	2	0	0	0
2848	24	-1	78	2	1.80	2	0	0	0
2876	24	-2	80	2	1.60	3	238	0	0
2962	23	-2	81	2	1.80	2	0	0	0
2980	25	-1	53	3	2.40	2	0	0	0
3076	29	-1	62	2	1.75	3	0	0	0
3130	23	-2	82	2	1.80	2	0	0	1
3157	23	-1	13	4	1.00	1	84	0	0
3279	26	-1	44	1	2.00	2	0	0	0
3284	25	-1	101	4	2.10	3	0	0	0

	Age	Experience	Income	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account
3292	25	-1	13	4	0.40	1	0	0	1
3394	25	-1	113	4	2.10	3	0	0	0
3425	23	-1	12	4	1.00	1	90	0	0
3626	24	-3	28	4	1.00	3	0	0	0
3796	24	-2	50	3	2.40	2	0	0	1
3824	23	-1	12	4	1.00	1	0	0	1
3887	24	-2	118	2	7.20	1	0	0	1
3946	25	-1	40	3	2.40	2	0	0	0
4015	25	-1	139	2	2.00	1	0	0	0
4088	29	-1	71	2	1.75	3	0	0	0
4116	24	-2	135	2	7.20	1	0	0	0
4285	23	-3	149	2	7.20	1	0	0	0
4411	23	-2	75	2	1.80	2	0	0	0
4481	25	-2	35	4	1.00	3	0	0	0
4514	24	-3	41	4	1.00	3	0	0	0
4582	25	-1	69	3	0.30	3	0	0	0
4957	29	-1	50	2	1.75	3	0	0	0

In [107]: `negative_Exp.head()`

Out[107]:

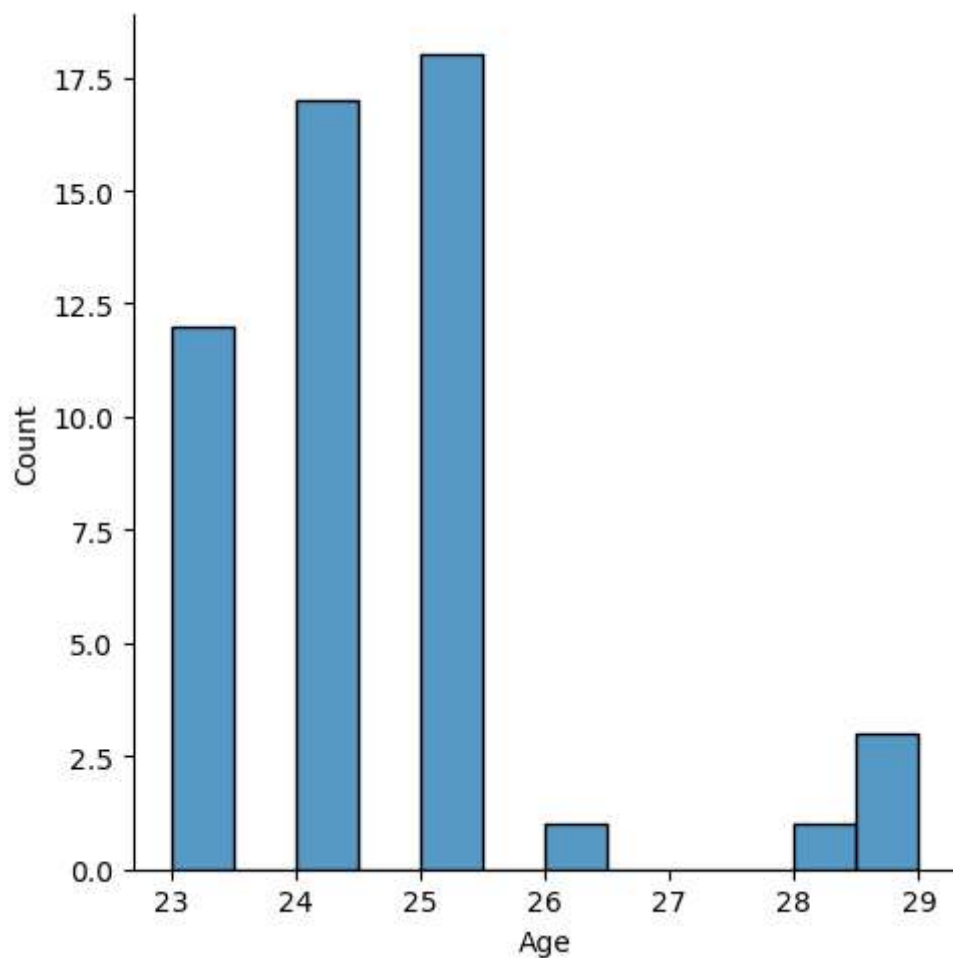
	Age	Experience	Income	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account
89	25	-1	113	4	2.30	3	0	0	0
226	24	-1	39	2	1.70	2	0	0	0
315	24	-2	51	3	0.30	3	0	0	0
451	28	-2	48	2	1.75	3	89	0	0
524	24	-1	75	4	0.20	1	0	0	0

In [108]: `#Total number of negative data`
`negative_Exp.shape`

Out[108]: (52, 12)

A Total of 52 persons having negative experience.

```
In [109]: ▶ ## Creating a distribution plot (displot) for the 'Age' column  
sns.displot(negative_Exp['Age'])  
plt.show()
```



People that belong to 23 - 30 years of age have negative experience. This can indicate that people in this category have less have newly started working and have limited experience in their jobs.

Mean Count of negative experience data

```
In [110]: ▶ negative_Exp['Experience'].mean()
```

Out[110]: -1.4423076923076923

Size of negative experience data

```
In [111]: ▶ negative_Exp.size
```

Out[111]: 624

In [112]: `print('There are {} records which has negative values for experience, approx 1.04%')`



There are 624 records which has negative values for experience, approx 1.04%

In [113]: `#Creating a copy of a Dataframe df and assign it to a new variable data
data = df.copy()
data`

Out[113]:

	Age	Experience	Income	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account
0	25	1	49	4	1.6	1	0	0	1
1	45	19	34	3	1.5	1	0	0	1
2	39	15	11	1	1.0	1	0	0	0
3	35	9	100	1	2.7	2	0	0	0
4	35	8	45	4	1.0	2	0	0	0
...
4995	29	3	40	1	1.9	3	0	0	0
4996	30	4	15	4	0.4	1	85	0	0
4997	63	39	24	2	0.3	3	0	0	0
4998	65	40	49	3	0.5	2	0	0	0
4999	28	4	83	3	0.8	1	0	0	0

5000 rows × 12 columns



In [114]: `#Shape of dataframe "data"
data.shape`

Out[114]: (5000, 12)

Using Numpy function to replace values in the Experience column of the dataframe data with mean of the "Experience" column where the original values are less than 0

In [115]: `data['Experience'] = np.where(data['Experience'] < 0, data['Experience'].mean(), data['Experience'])`

Filter rows in the Dataframe 'data' where the 'Experience' column has negative values

In [116]: `data[data['Experience']<0]`

Out[116]:

Age	Experience	Income	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	Ac

In [117]: `#Calculating correlation matrixx`

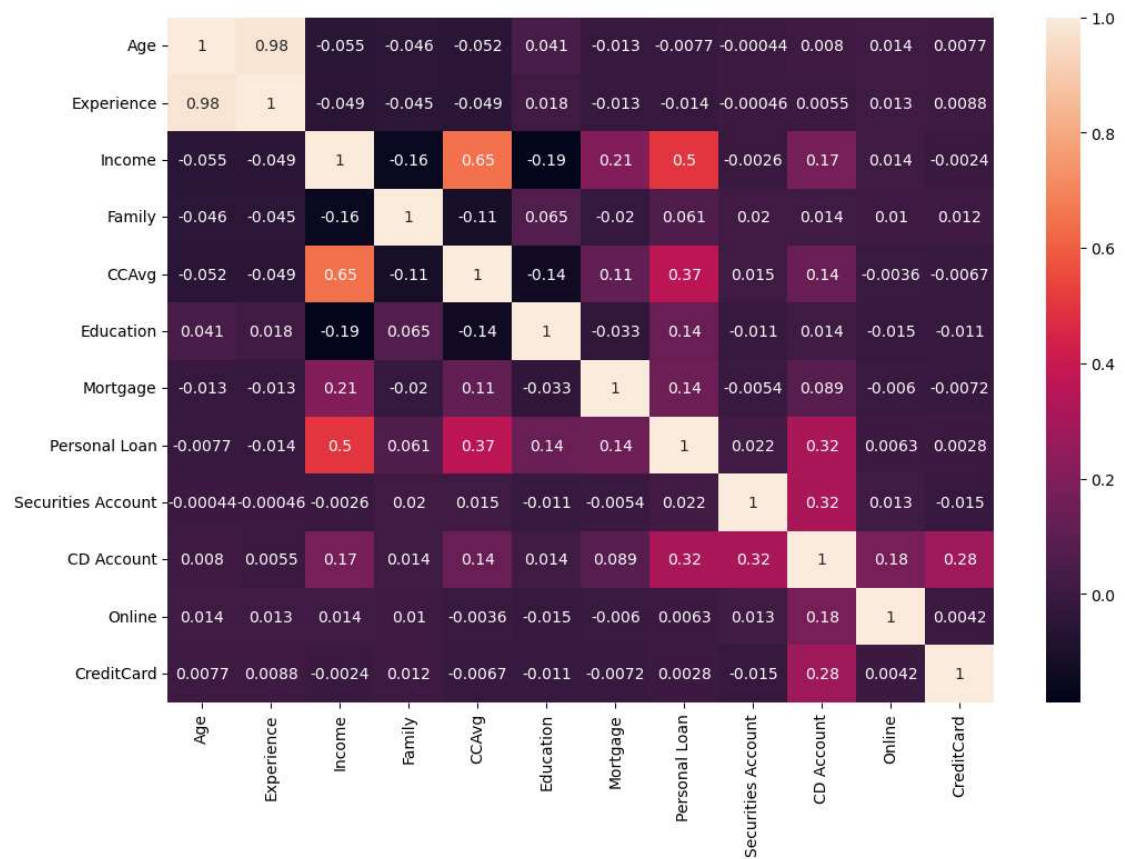
`data.corr()`

Out[117]:

	Age	Experience	Income	Family	CCAvg	Education	Mortgage	F
Age	1.000000	0.977008	-0.055269	-0.046418	-0.052012	0.041334	-0.012539	-C
Experience	0.977008	1.000000	-0.049054	-0.045488	-0.048708	0.018097	-0.013378	-C
Income	-0.055269	-0.049054	1.000000	-0.157501	0.645984	-0.187524	0.206806	C
Family	-0.046418	-0.045488	-0.157501	1.000000	-0.109275	0.064929	-0.020445	C
CCAvg	-0.052012	-0.048708	0.645984	-0.109275	1.000000	-0.136124	0.109905	C
Education	0.041334	0.018097	-0.187524	0.064929	-0.136124	1.000000	-0.033327	C
Mortgage	-0.012539	-0.013378	0.206806	-0.020445	0.109905	-0.033327	1.000000	C
Personal Loan	-0.007726	-0.014045	0.502462	0.061367	0.366889	0.136722	0.142095	1
Securities Account	-0.000436	-0.000462	-0.002616	0.019994	0.015086	-0.010812	-0.005411	C
CD Account	0.008043	0.005502	0.169738	0.014110	0.136534	0.013934	0.089311	C
Online	0.013702	0.013455	0.014206	0.010354	-0.003611	-0.015004	-0.005995	C
CreditCard	0.007681	0.008833	-0.002385	0.011588	-0.006689	-0.011014	-0.007231	C

```
In [118]: ▶ #Creating a heatmap based of the correlation matrix using Seaborn (Sns) and
plt.figure(figsize = (12, 8))
sns.heatmap(data.corr(), annot=True)
```

Out[118]: <Axes: >



The correlation between age and experience is notably strong indicating a substantial relationship between these two variables. Consequently, it may be prudent to consider removing one of these columns from the dataset to avoid potential multicollinearity issues.


```
In [119]: #Dropping the 'Experience' column from DataFrame 'data' using the data.drop
data = data.drop(['Experience'], axis=1)
data
```

```
Out[119]:
```

	Age	Income	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Onli
0	25	49	4	1.6	1	0	0	1	0	
1	45	34	3	1.5	1	0	0	1	0	
2	39	11	1	1.0	1	0	0	0	0	
3	35	100	1	2.7	2	0	0	0	0	
4	35	45	4	1.0	2	0	0	0	0	
...
4995	29	40	1	1.9	3	0	0	0	0	
4996	30	15	4	0.4	1	85	0	0	0	
4997	63	24	2	0.3	3	0	0	0	0	
4998	65	49	3	0.5	2	0	0	0	0	
4999	28	83	3	0.8	1	0	0	0	0	

5000 rows × 11 columns

```
In [120]: data.head()
```

```
Out[120]:
```

	Age	Income	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Onli
0	25	49	4	1.6	1	0	0	1	0	
1	45	34	3	1.5	1	0	0	1	0	
2	39	11	1	1.0	1	0	0	0	0	
3	35	100	1	2.7	2	0	0	0	0	
4	35	45	4	1.0	2	0	0	0	0	

```
In [121]: #Checking unique value in Education column
data['Education'].unique()
```

```
Out[121]: array([1, 2, 3], dtype=int64)
```

In [122]: `#Defining a Python function called experience(x) that takes an input x`

```
def experience(x):
    if x==1:
        return "UnderGraduate"
    elif x==2:
        return "Graduate"
    else:
        return "Working Professional"
```

In [123]: `## Apply the experience function to create a new column 'EDU'`
`data['EDU'] = data['Education'].apply(experience)`

In [124]: `data.head()`

Out[124]:

	Age	Income	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Onli
0	25	49	4	1.6	1	0	0	1	0	
1	45	34	3	1.5	1	0	0	1	0	
2	39	11	1	1.0	1	0	0	0	0	
3	35	100	1	2.7	2	0	0	0	0	
4	35	45	4	1.0	2	0	0	0	0	

Retrieving the unique values present in the 'EDU' column

In [125]: `data['EDU'].unique()`

Out[125]: `array(['UnderGraduate', 'Graduate', 'Working Professional'], dtype=object)`

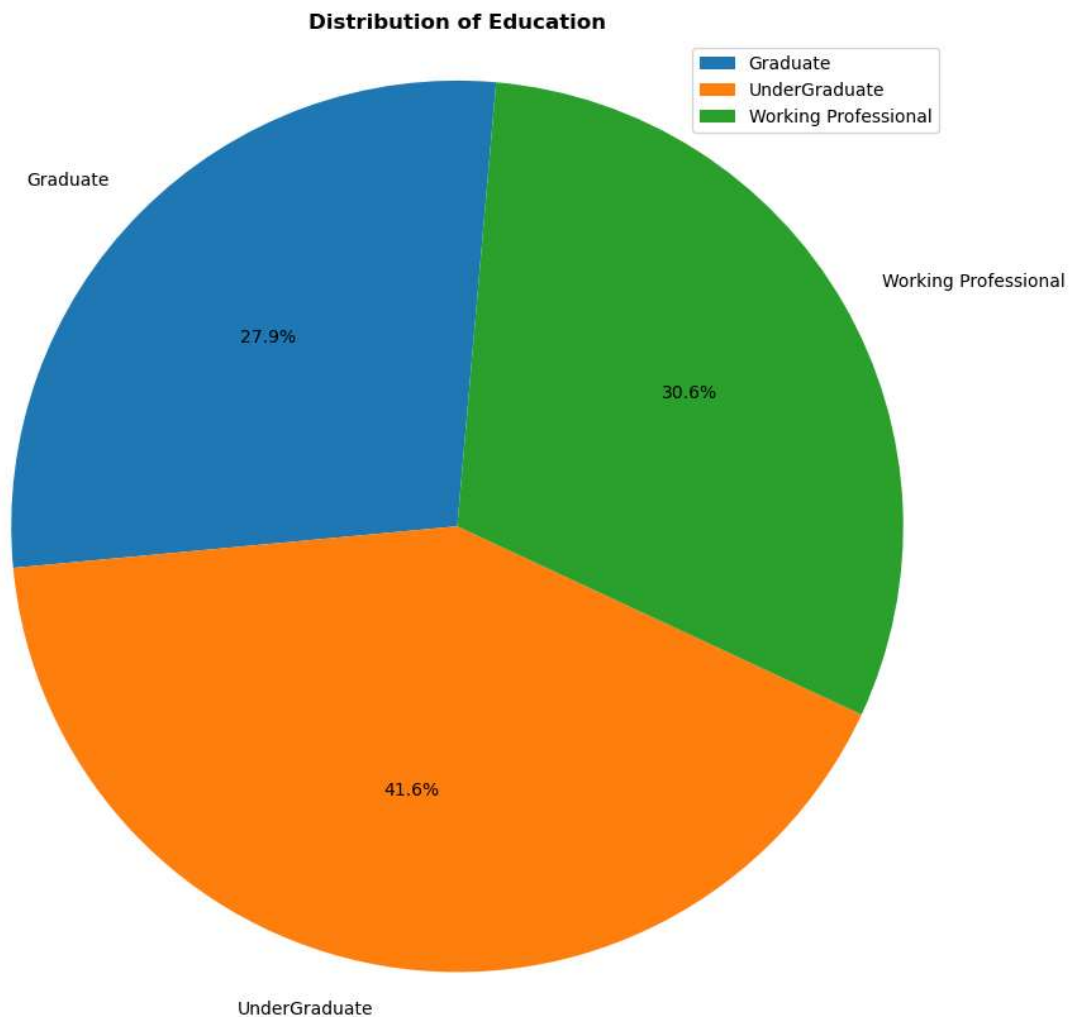
In [126]: `## Grouping DataFrame 'data' by the 'EDU' column and then calculating column sums`
`education_dis = data.groupby('EDU')['Age'].sum()`
`education_dis`

Out[126]:

EDU	
Graduate	63191
UnderGraduate	94244
Working Professional	69257
Name: Age, dtype: int64	

Creating a pie chart to visualize the distribution of education category

```
In [127]: ▶ plt.figure(figsize=(10, 10))
plt.pie(education_dis, labels=education_dis.index, autopct='%1.1f%%', star
plt.axis('equal')
plt.title('Distribution of Education', fontweight='bold')
plt.legend(education_dis.index, loc='upper right')
plt.show()
```



Retriving the unique values present in the "Income" column

```
In [128]: data['Income'].unique()
```

```
Out[128]: array([ 49,  34,  11, 100,  45,  29,  72,  22,  81, 180, 105, 114,  40,
        112, 130, 193,  21,  25,  63,  62,  43, 152,  83, 158,  48, 119,
         35,  41,  18,  50, 121,  71, 141,  80,  84,  60, 132, 104,  52,
        194,   8, 131, 190,  44, 139,  93, 188,  39, 125,  32,  20, 115,
         69,  85, 135,  12, 133,  19,  82, 109,  42,  78,  51, 113, 118,
         64, 161,  94,  15,  74,  30,  38,   9,  92,  61,  73,  70, 149,
         98, 128,  31,  58,  54, 124, 163,  24,  79, 134,  23,  13, 138,
        171, 168,  65,  10, 148, 159, 169, 144, 165,  59,  68,  91, 172,
         55, 155,  53,  89,  28,  75, 170, 120,  99, 111,  33, 129, 122,
        150, 195, 110, 101, 191, 140, 153, 173, 174,  90, 179, 145, 200,
        183, 182,  88, 160, 205, 164,  14, 175, 103, 108, 185, 204, 154,
        102, 192, 202, 162, 142,  95, 184, 181, 143, 123, 178, 198, 201,
        203, 189, 151, 199, 224, 218], dtype=int64)
```

Count the occurrences of the unique values in the 'Securities Account' column of the dataframe 'data'

```
In [129]: data['Securities Account'].value_counts()
```

```
Out[129]: Securities Account
0      4478
1       522
Name: count, dtype: int64
```

Count the occurrences of unique values in the 'CD Account' column of df 'data'

```
In [130]: data['CD Account'].value_counts()
```

```
Out[130]: CD Account
0      4698
1       302
Name: count, dtype: int64
```

Defining a python function called security that takes a DataFrame y as input and categorizes individuals into different groups based on the values of 'Securities Account' and 'CD Account' columns

```
In [131]: def security(y):
            if(y['Securities Account'] == 1) & (y['CD Account'] == 1):
                return "Both Security and Deposit Account"
            if(y['Securities Account'] == 0) & (y['CD Account'] == 0):
                return "No Account"
            if(y['Securities Account'] == 1) & (y['CD Account'] == 0):
                return "Only Security Account"
            if(y['Securities Account'] == 0) & (y['CD Account'] == 1):
                return "Only Deposit Account"
```

Applying the security function to DataFrame 'data' using the apply method to create a new column named 'Account_Holder_Category'

```
In [132]: data['Account_Holder_Category'] = data.apply(security, axis=1)
```

```
In [133]: data.head()
```

```
Out[133]:
```

	Age	Income	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Onli
0	25	49	4	1.6	1	0	0	1	0	
1	45	34	3	1.5	1	0	0	1	0	
2	39	11	1	1.0	1	0	0	0	0	
3	35	100	1	2.7	2	0	0	0	0	
4	35	45	4	1.0	2	0	0	0	0	

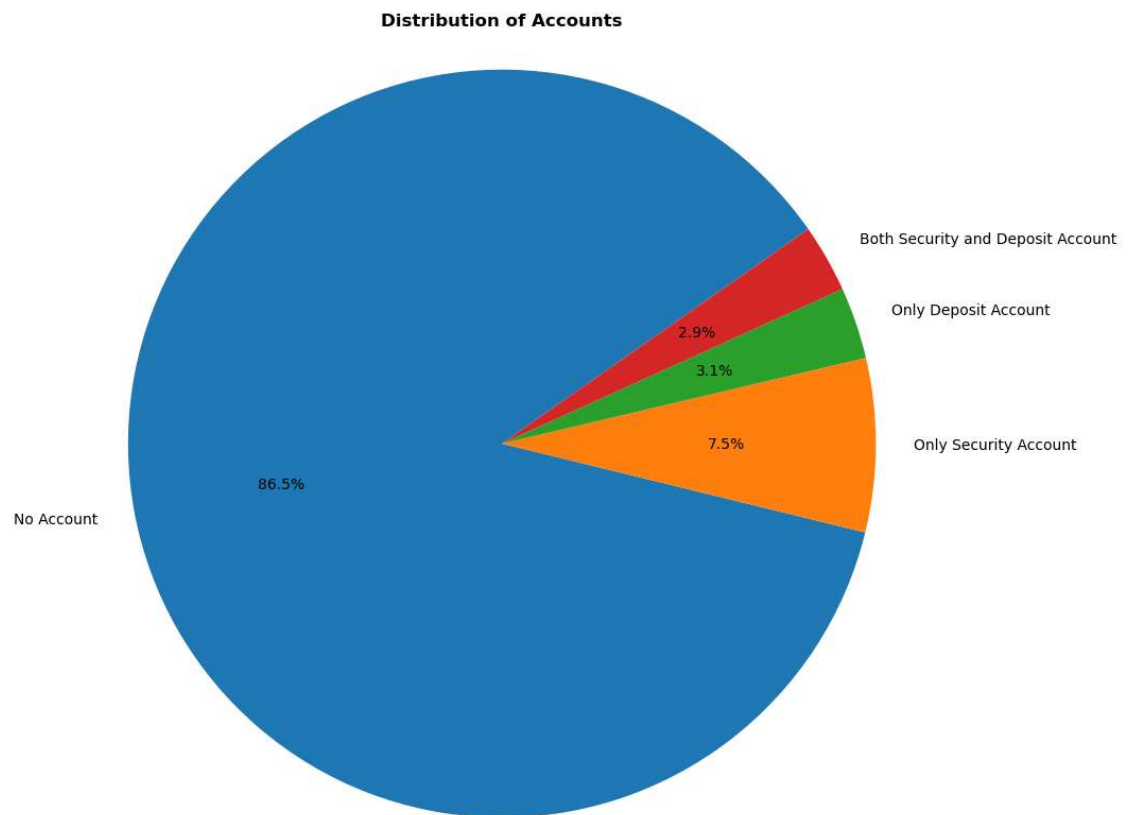
Count the occurrences of unique values in the 'Account_Holder_Category' column

```
In [134]: account_values = data['Account_Holder_Category'].value_counts()
account_values
```

```
Out[134]: Account_Holder_Category
No Account                                4323
Only Security Account                     375
Only Deposit Account                      155
Both Security and Deposit Account         147
Name: count, dtype: int64
```

Creating a pie chart to visualize the distribution of account holder categories based on the 'Account_Holder_Category' column

```
In [135]: ▶ plt.figure(figsize=(10, 10))
plt.pie(account_values, labels = account_values.index, autopct='%1.1f%%',
plt.axis('equal')
plt.title('Distribution of Accounts', fontweight='bold')
plt.show()
```



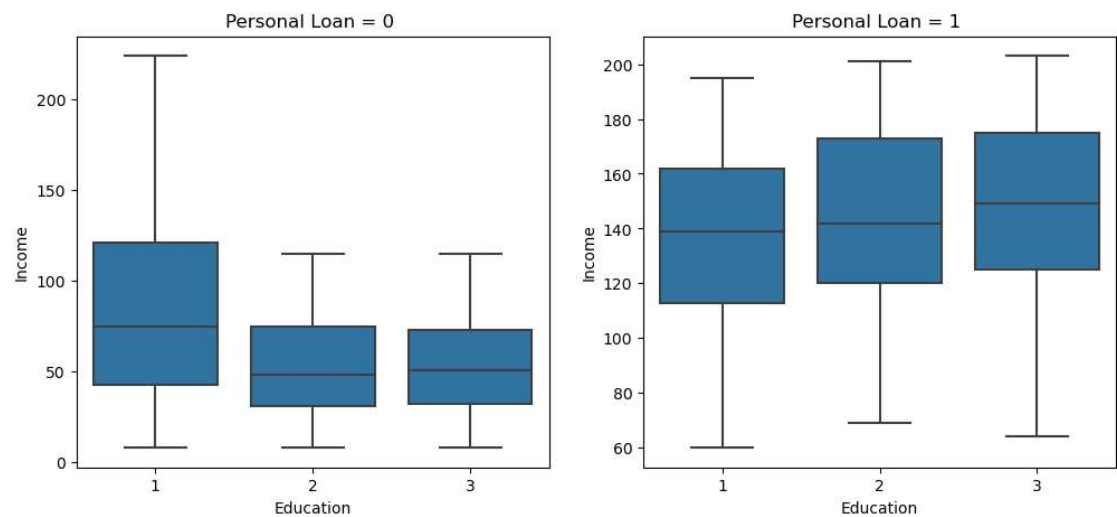
Creating two separate boxplot visualizations based on the 'Personal Loan' column.

```
In [136]: #Creating two separate boxplot visualizations based on 'Personal Loan'
fig, axes = plt.subplots(1, 2, figsize=(12, 5))

#First boxplot for 'Personal Loan' = 0
sns.boxplot(data = data[data['Personal Loan']==0], x = 'Education', y='Income')
axes[0].set_title("Personal Loan = 0 ")
axes[0].legend().set_visible(False)

#Second boxplot for 'Personal Loan' = 1
sns.boxplot(data = data[data['Personal Loan']==1], x = 'Education', y='Income')
axes[1].set_title("Personal Loan = 1 ")
axes[1].legend().set_visible(False)

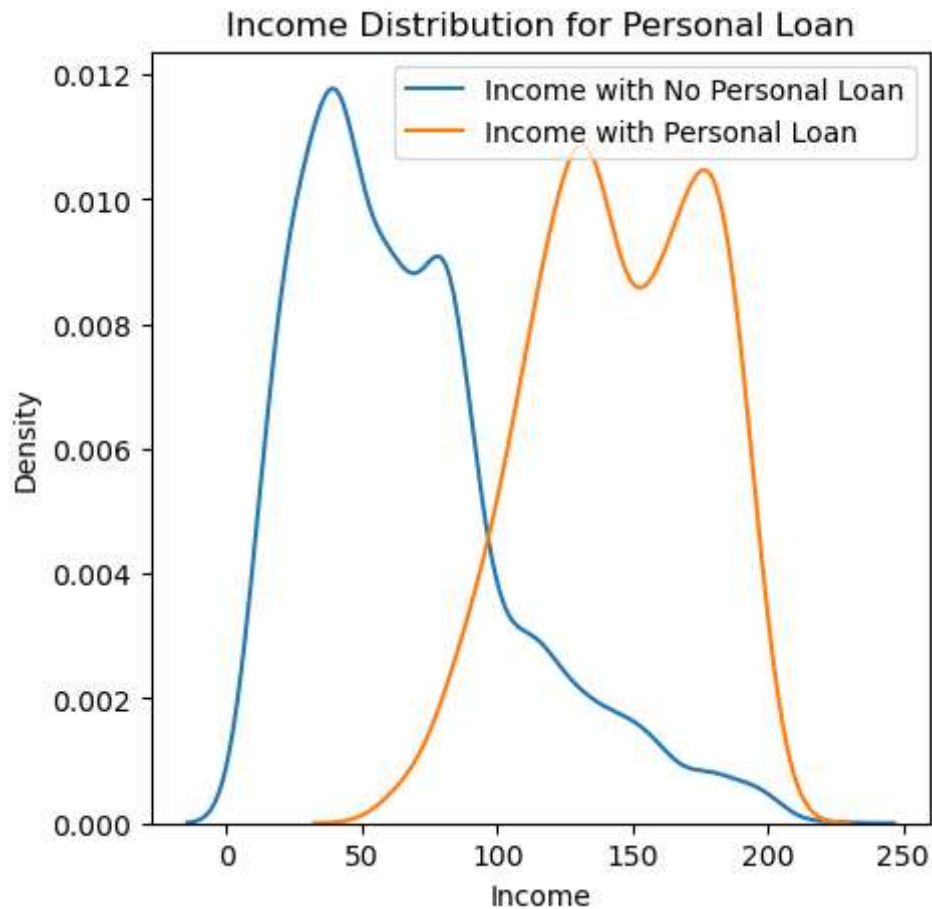
#Show the plots
plt.show()
```



Creating a Kernel Density Estimation (KDE) plot to visualize the distribution of income for two groups:

1. individuals with no personal loan
2. individuals with personal loan

```
In [144]: ▶ plt.figure(figsize=(5,5))
sns.kdeplot(data=data[data['Personal Loan']==0]['Income'], label='Income w
sns.kdeplot(data=data[data['Personal Loan']==1]['Income'], label='Income w
plt.title("Income Distribution for Personal Loan")
plt.xlabel("Income")
plt.ylabel("Density")
plt.legend()
plt.show()
```



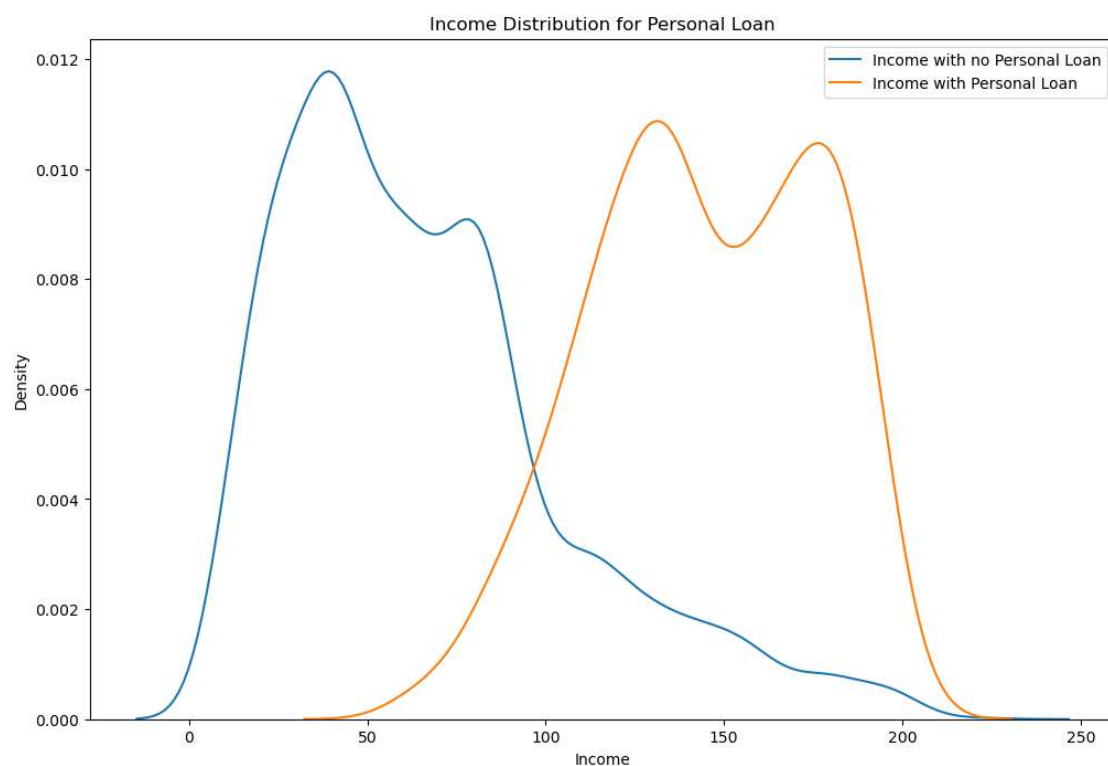
Individuals with a Personal Loan typically exhibit a salary range spanning from a minimum of 50,000 to a maximum of 2,00,000. Among those who have opted for a Personal Loan, a noteworthy majority falls within the income bracket of 100000 to 150000, indicating a peak in loan applications within this salary range.

Function is designed to create a Kernel Density Estimation(KDE) plot to visualize the distribution of a numerical variable (col1) in a dataset, separated by a binary categorical variable (col2) with two values (0 and 1)


```
In [138]: ▶ def plot(col1, col2, label1, label2, title):  
           plt.figure(figsize=(12, 8))  
  
           sns.kdeplot(data=data[data[col2]==0][col1], label=label1)  
           sns.kdeplot(data=data[data[col2]==1][col1], label=label2)  
  
           plt.legend()  
           plt.title(title)  
           plt.xlabel(col1)  
           plt.ylabel("Density")  
           plt.show()
```

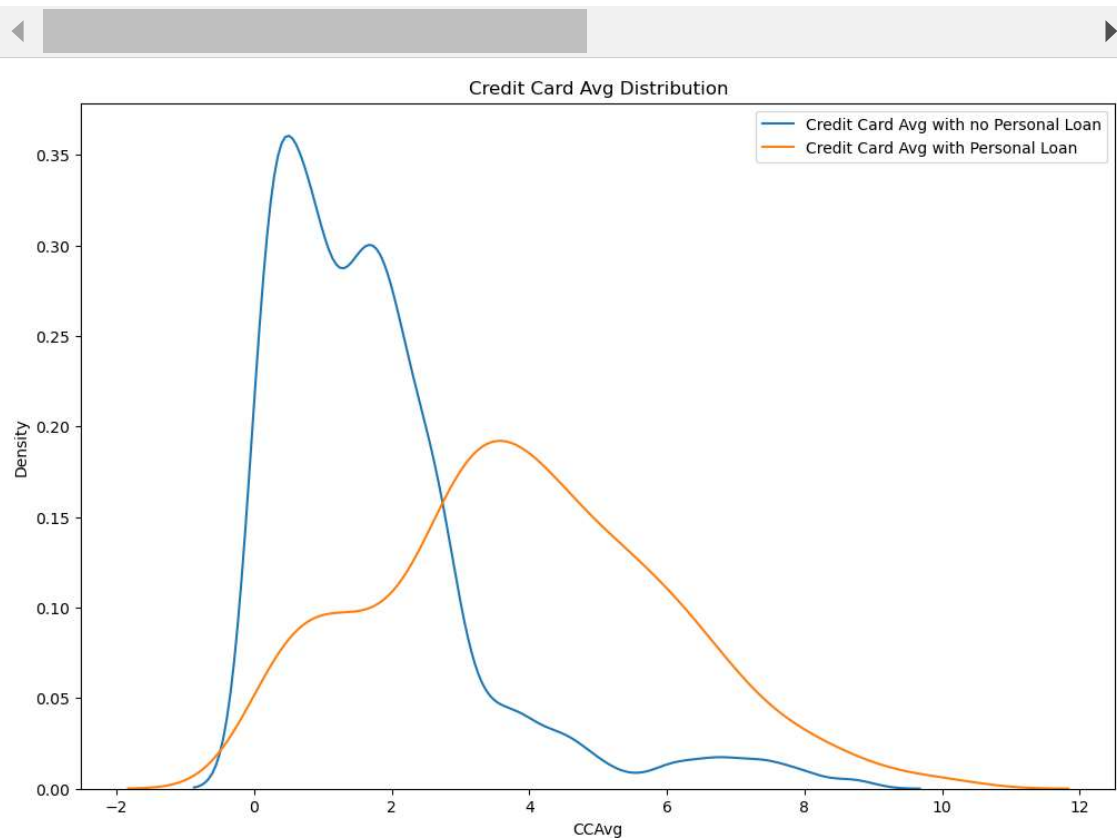
Calling the plot function to create a KDE plot that visualizes the distribution of 'Income' based on the presence or absence of a 'Personal Loan'

```
In [139]: ▶ plot('Income', 'Personal Loan', 'Income with no Personal Loan', 'Income wi
```



Calling the plotfunction to create a KDE plot that visualizes the distribution of 'CCAvg' (Credit Card Average) based on the presence or absence of a 'Personal Loan'

```
In [140]: plot('CCAvg', 'Personal Loan', 'Credit Card Avg with no Personal Loan', 'C
```

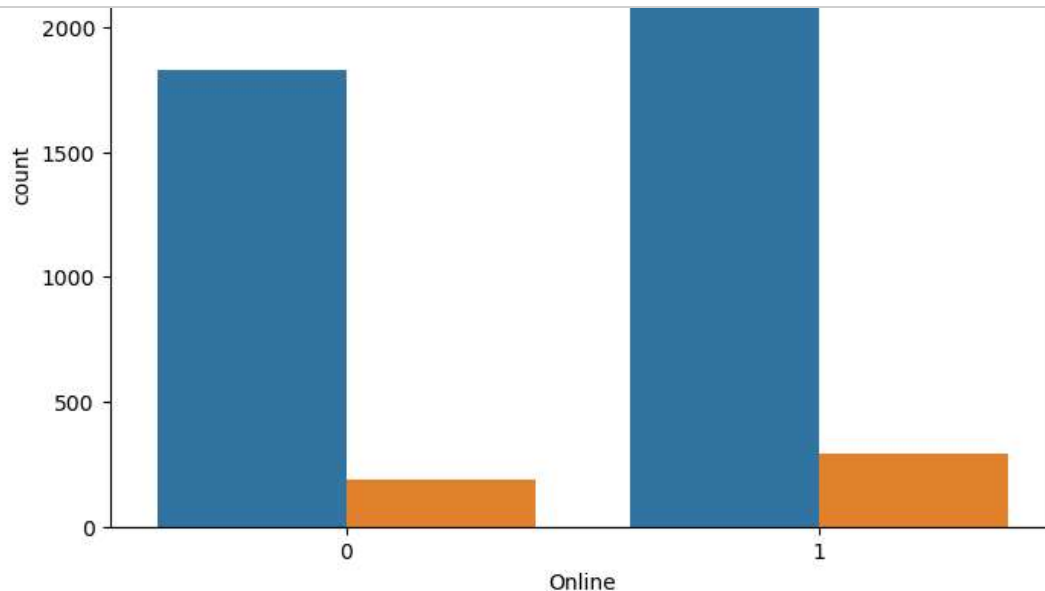


Defining a list called `col` containing four column names: 'Securities Account', 'Online', 'Account_Holder_Category', and 'Credit Card'

```
In [141]: col = ['Securities Account', 'Online', 'Account_Holder_Category', 'Credit
```

Creating count plots for each of the columns listed in the `col` list, and you are visualizing how the counts vary with respect to the 'Personal Loan' column.

```
In [142]: ▶ for i in col:
plt.figure(figsize=(8, 6))
sns.countplot(x=i, data=data, hue='Personal Loan')
plt.title(f'Count Plot of {i} by Personal Loan')
plt.show()
```



Count Plot of Account_Holder_Category by Personal Loan

In Conclusion, we can say that people with age 23 - 30 are less likely going to be approved of loan due to lesser income and experience.

Also people with Income between 50,000 - 2,00,000 are more likely to have their loan approved.

People with credit cards are less likely to apply for personal loans.