step-1 Business problem understanding

- Analytical report

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
df = pd.read_csv("LoanData.csv")
df
```

Out[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001002	Male	No	0	Graduate	No	5849
1	LP001003	Male	Yes	1	Graduate	No	4583
2	LP001005	Male	Yes	0	Graduate	Yes	3000
3	LP001006	Male	Yes	0	Not Graduate	No	2583
4	LP001008	Male	No	0	Graduate	No	6000
...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900
610	LP002979	Male	Yes	3+	Graduate	No	4106
611	LP002983	Male	Yes	1	Graduate	No	8072
612	LP002984	Male	Yes	2	Graduate	No	7583
613	LP002990	Female	No	0	Graduate	Yes	4583

614 rows × 13 columns

Data Exploration

In [6]:
```
df.shape
```

Out[6]: (614, 13)

In [7]:
```
df.columns.tolist()
```

Out[7]:
```
['Loan_ID',
 'Gender',
 'Married',
 'Dependents',
 'Education',
 'Self_Employed',
 'ApplicantIncome',
 'CoapplicantIncome',
 'LoanAmount',
 'Loan_Amount_Term',
 'Credit_History',
 'Property_Area',
 'Loan_Status']
```

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

value_counts() are applicable on discrete or count

In [10]:
```
# Discrete

df["Gender"].unique()  # This is correct data type
```

Out[10]:
```
array(['Male', 'Female', nan], dtype=object)
```

In [11]:
```
df["Gender"].value_counts()
```

Out[11]:
```
Gender
Male      489
Female    112
Name: count, dtype: int64
```

In [12]:
```
# Discrete

df["Married"].unique()  # This is correct data type
```

Out[12]:
```
array(['No', 'Yes', nan], dtype=object)
```

In [13]:
```
df["Married"].value_counts()
```

Out[13]:
```
Married
Yes    398
No     213
Name: count, dtype: int64
```

In [14]:
```
# Discrete Count

df["Dependents"].unique() # This is wrong data type because of wrong value
```

Out[14]:
```
array(['0', '1', '2', '3+', nan], dtype=object)
```

In [15]:
```
# Dependents is Discrete count thats why we use value_counts()

df["Dependents"].value_counts()
```

Out[15]:
```
Dependents
0     345
1     102
2     101
3+     51
Name: count, dtype: int64
```

In [16]:
```
# Discrete

df["Education"].unique()  # This is correct data type
```

Out[16]:
```
array(['Graduate', 'Not Graduate'], dtype=object)
```

In [17]:
```
df["Education"].value_counts()
```

Out[17]:
```
Education
Graduate        480
Not Graduate    134
Name: count, dtype: int64
```

In [18]:
```
# Discrete

df["Self_Employed"].unique()  # This is correct data type
```

Out[18]:
```
array(['No', 'Yes', nan], dtype=object)
```

In [19]:
```
df["Self_Employed"].value_counts()
```

Out[19]:
```
Self_Employed
No     500
Yes     82
Name: count, dtype: int64
```

In [31]:
```
# Continous
```

```
df["ApplicantIncome"].unique()  # This is correct data Type
```

Out[31]:
```
array([ 5849,  4583,  3000,  2583,  6000,  5417,  2333,  3036,  4006,
       12841,  3200,  2500,  3073,  1853,  1299,  4950,  3596,  3510,
        4887,  2600,  7660,  5955,  3365,  3717,  9560,  2799,  4226,
        1442,  3750,  4166,  3167,  4692,  3500, 12500,  2275,  1828,
        3667,  3748,  3600,  1800,  2400,  3941,  4695,  3410,  5649,
        5821,  2645,  4000,  1928,  3086,  4230,  4616, 11500,  2708,
        2132,  3366,  8080,  3357,  3029,  2609,  4945,  5726, 10750,
        7100,  4300,  3208,  1875,  4755,  5266,  1000,  3333,  3846,
        2395,  1378,  3988,  2366,  8566,  5695,  2958,  6250,  3273,
        4133,  3620,  6782,  2484,  1977,  4188,  1759,  4288,  4843,
       13650,  4652,  3816,  3052, 11417,  7333,  3800,  2071,  5316,
        2929,  3572,  7451,  5050, 14583,  2214,  5568, 10408,  5667,
        2137,  2957,  3692, 23803,  3865, 10513,  6080, 20166,  2014,
        2718,  3459,  4895,  3316, 14999,  4200,  5042,  6950,  2698,
       11757,  2330, 14866,  1538, 10000,  4860,  6277,  2577,  9166,
        2281,  3254, 39999,  9538,  2980,  1863,  7933,  3089,  4167,
        9323,  3707,  2439,  2237,  8000,  1820, 51763,  3522,  5708,
        4344,  3497,  2045,  5516,  6400,  1916,  4600, 33846,  3625,
       39147,  2178,  2383,   674,  9328,  4885, 12000,  6033,  3858,
        4191,  3125,  8333,  1907,  3416, 11000,  4923,  3992,  3917,
        4408,  3244,  3975,  2479,  3418,  3430,  7787,  5703,  3173,
        3850,   150,  3727,  5000,  4283,  2221,  4009,  2971,  7578,
        3250,  4735,  4758,  2491,  3716,  3189,  3155,  5500,  5746,
        3463,  3812,  3315,  5819,  2510,  2965,  3406,  6050,  9703,
        6608,  2882,  1809,  1668,  3427,  2661, 16250,  3083,  6045,
        5250, 14683,  4931,  6083,  2060,  3481,  7200,  5166,  4095,
        4708,  4333,  2876,  3237, 11146,  2833,  2620,  3900,  2750,
        3993,  3103,  4100,  4053,  3927,  2301,  1811, 20667,  3158,
        3704,  4124,  9508,  3075,  4400,  3153,  4416,  6875,  4666,
        2875,  1625,  2000,  3762, 20233,  7667,  2917,  2927,  2507,
        2473,  3399,  2058,  3541,  4342,  3601,  3166, 15000,  8666,
        4917,  5818,  4384,  2935, 63337,  9833,  5503,  1830,  4160,
        2647,  2378,  4554,  2499,  3523,  6333,  2625,  9083,  8750,
        2666,  2423,  3813,  3875,  5167,  4723,  4750,  3013,  6822,
        6216,  5124,  6325, 19730, 15759,  5185,  3062,  2764,  4817,
        4310,  3069,  5391,  5941,  7167,  4566,  2346,  3010,  5488,
        9167,  9504,  1993,  3100,  3276,  3180,  3033,  3902,  1500,
        2889,  2755,  1963,  7441,  4547,  2167,  2213,  8300, 81000,
        3867,  6256,  6096,  2253,  2149,  2995,  1600,  1025,  3246,
        5829,  2720,  7250, 14880,  4606,  5935,  2920,  2717,  8624,
        6500, 12876,  2425, 10047,  1926, 10416,  7142,  3660,  7901,
        4707, 37719,  3466,  3539,  3340,  2769,  2309,  1958,  3948,
        2483,  7085,  3859,  4301,  3708,  4354,  8334,  2083,  7740,
        3015,  5191,  2947, 16692,   210,  3450,  2653,  4691,  5532,
       16525,  6700,  2873, 16667,  4350,  3095, 10833,  3547, 18333,
        2435,  2699,  5333,  3691, 17263,  3597,  3326,  4625,  2895,
        6283,   645,  3159,  4865,  4050,  3814, 20833,  3583, 13262,
        3598,  6065,  3283,  2130,  5815,  2031,  3074,  4683,  3400,
        2192,  5677,  7948,  4680, 17500,  3775,  5285,  2679,  6783,
        4281,  3588, 11250, 18165,  2550,  6133,  3617,  6417,  4608,
        2138,  3652,  2239,  3017,  2768,  3358,  2526,  2785,  6633,
        2492,  2454,  3593,  5468,  2667, 10139,  3887,  4180,  3675,
       19484,  5923,  5800,  8799,  4467,  3417,  5116, 16666,  6125,
        6406,  3087,  3229,  1782,  3182,  6540,  1836,  1880,  2787,
        2297,  2165,  2726,  9357, 16120,  3833,  6383,  2987,  9963,
```

```
       5780,   416,  2894,  3676,  3987,  3232,  2900,  4106,  8072,
       7583], dtype=int64)
```

In [37]:
```
# Continous

df["CoapplicantIncome"].unique()   # This is correct data type
```

Out[37]:
```
array([0.00000000e+00, 1.50800000e+03, 2.35800000e+03, 4.19600000e+03,
       1.51600000e+03, 2.50400000e+03, 1.52600000e+03, 1.09680000e+04,
       7.00000000e+02, 1.84000000e+03, 8.10600000e+03, 2.84000000e+03,
       1.08600000e+03, 3.50000000e+03, 5.62500000e+03, 1.91100000e+03,
       1.91700000e+03, 2.92500000e+03, 2.25300000e+03, 1.04000000e+03,
       2.08300000e+03, 3.36900000e+03, 1.66700000e+03, 3.00000000e+03,
       2.06700000e+03, 1.33000000e+03, 1.45900000e+03, 7.21000000e+03,
       1.66800000e+03, 1.21300000e+03, 2.33600000e+03, 3.44000000e+03,
       2.27500000e+03, 1.64400000e+03, 1.16700000e+03, 1.59100000e+03,
       2.20000000e+03, 2.25000000e+03, 2.85900000e+03, 3.79600000e+03,
       3.44900000e+03, 4.59500000e+03, 2.25400000e+03, 3.06600000e+03,
       1.87500000e+03, 1.77400000e+03, 4.75000000e+03, 3.02200000e+03,
       4.00000000e+03, 2.16600000e+03, 1.88100000e+03, 2.53100000e+03,
       2.00000000e+03, 2.11800000e+03, 4.16700000e+03, 2.90000000e+03,
       5.65400000e+03, 1.82000000e+03, 2.30200000e+03, 9.97000000e+02,
       3.54100000e+03, 3.26300000e+03, 3.80600000e+03, 3.58300000e+03,
       7.54000000e+02, 1.03000000e+03, 1.12600000e+03, 3.60000000e+03,
       2.33300000e+03, 4.11400000e+03, 2.28300000e+03, 1.39800000e+03,
       2.14200000e+03, 2.66700000e+03, 8.98000000e+03, 2.01400000e+03,
       1.64000000e+03, 3.85000000e+03, 2.56900000e+03, 1.92900000e+03,
       7.75000000e+03, 1.43000000e+03, 2.03400000e+03, 4.48600000e+03,
       1.42500000e+03, 1.66600000e+03, 8.30000000e+02, 3.75000000e+03,
       1.04100000e+03, 1.28000000e+03, 1.44700000e+03, 3.16600000e+03,
       3.33300000e+03, 1.76900000e+03, 7.36000000e+02, 1.96400000e+03,
       1.61900000e+03, 1.13000000e+04, 1.45100000e+03, 7.25000000e+03,
       5.06300000e+03, 2.13800000e+03, 5.29600000e+03, 2.58300000e+03,
       2.36500000e+03, 2.81600000e+03, 2.50000000e+03, 1.08300000e+03,
       1.25000000e+03, 3.02100000e+03, 9.83000000e+02, 1.80000000e+03,
       1.77500000e+03, 2.38300000e+03, 1.71700000e+03, 2.79100000e+03,
       1.01000000e+03, 1.69500000e+03, 2.05400000e+03, 2.59800000e+03,
       1.77900000e+03, 1.26000000e+03, 5.00000000e+03, 1.98300000e+03,
       5.70100000e+03, 1.30000000e+03, 4.41700000e+03, 4.33300000e+03,
       1.84300000e+03, 1.86800000e+03, 3.89000000e+03, 2.16700000e+03,
       7.10100000e+03, 2.10000000e+03, 4.25000000e+03, 2.20900000e+03,
       3.44700000e+03, 1.38700000e+03, 1.81100000e+03, 1.56000000e+03,
       1.85700000e+03, 2.22300000e+03, 1.84200000e+03, 3.27400000e+03,
       2.42600000e+03, 8.00000000e+02, 9.85799988e+02, 3.05300000e+03,
       2.41600000e+03, 3.33400000e+03, 2.54100000e+03, 2.93400000e+03,
       1.75000000e+03, 1.80300000e+03, 1.86300000e+03, 2.40500000e+03,
       2.13400000e+03, 1.89000000e+02, 1.59000000e+03, 2.98500000e+03,
       4.98300000e+03, 2.16000000e+03, 2.45100000e+03, 1.79300000e+03,
       1.83300000e+03, 4.49000000e+03, 6.88000000e+02, 4.60000000e+03,
       1.58700000e+03, 1.22900000e+03, 2.33000000e+03, 2.45800000e+03,
       3.23000000e+03, 2.16800000e+03, 4.58300000e+03, 6.25000000e+03,
       5.05000000e+02, 3.16700000e+03, 3.66700000e+03, 3.03300000e+03,
       5.26600000e+03, 7.87300000e+03, 1.98700000e+03, 9.23000000e+02,
       4.99600000e+03, 4.23200000e+03, 1.60000000e+03, 3.13600000e+03,
       2.41700000e+03, 2.11500000e+03, 1.62500000e+03, 1.40000000e+03,
       4.84000000e+02, 2.00000000e+04, 2.40000000e+03, 2.03300000e+03,
       3.23700000e+03, 2.77300000e+03, 1.41700000e+03, 1.71900000e+03,
       4.30000000e+03, 1.61200008e+01, 2.34000000e+03, 1.85100000e+03,
       1.12500000e+03, 5.06400000e+03, 1.99300000e+03, 8.33300000e+03,
       1.21000000e+03, 1.37600000e+03, 1.71000000e+03, 1.54200000e+03,
       1.25500000e+03, 1.45600000e+03, 1.73300000e+03, 2.46600000e+03,
       4.08300000e+03, 2.18800000e+03, 1.66400000e+03, 2.91700000e+03,
       2.07900000e+03, 1.50000000e+03, 4.64800000e+03, 1.01400000e+03,
```

```
1.87200000e+03, 1.60300000e+03, 3.15000000e+03, 2.43600000e+03,
2.78500000e+03, 1.13100000e+03, 2.15700000e+03, 9.13000000e+02,
1.70000000e+03, 2.85700000e+03, 4.41600000e+03, 3.68300000e+03,
5.62400000e+03, 5.30200000e+03, 1.48300000e+03, 6.66700000e+03,
3.01300000e+03, 1.28700000e+03, 2.00400000e+03, 2.03500000e+03,
6.66600000e+03, 3.66600000e+03, 3.42800000e+03, 1.63200000e+03,
1.91500000e+03, 1.74200000e+03, 1.42400000e+03, 7.16600000e+03,
2.08700000e+03, 1.30200000e+03, 5.50000000e+03, 2.04200000e+03,
3.90600000e+03, 5.36000000e+02, 2.84500000e+03, 2.52400000e+03,
6.63000000e+02, 1.95000000e+03, 1.78300000e+03, 2.01600000e+03,
2.37500000e+03, 3.25000000e+03, 4.26600000e+03, 1.03200000e+03,
2.66900000e+03, 2.30600000e+03, 2.42000000e+02, 2.06400000e+03,
4.61000000e+02, 2.21000000e+03, 2.73900000e+03, 2.23200000e+03,
3.38370000e+04, 1.52200000e+03, 3.41600000e+03, 3.30000000e+03,
1.00000000e+03, 4.16670000e+04, 2.79200000e+03, 4.30100000e+03,
3.80000000e+03, 1.41100000e+03, 2.40000000e+02])
```

In [39]:
```
# continous

df["LoanAmount"].unique()  # This is correct data type
```

Out[39]:
```
array([ nan, 128.,  66., 120., 141., 267.,  95., 158., 168., 349.,  70.,
       109., 200., 114.,  17., 125., 100.,  76., 133., 115., 104., 315.,
       116., 112., 151., 191., 122., 110.,  35., 201.,  74., 106., 320.,
       144., 184.,  80.,  47.,  75., 134.,  96.,  88.,  44., 286.,  97.,
       135., 180.,  99., 165., 258., 126., 312., 136., 172.,  81., 187.,
       113., 176., 130., 111., 167., 265.,  50., 210., 175., 131., 188.,
        25., 137., 160., 225., 216.,  94., 139., 152., 118., 185., 154.,
        85., 259., 194.,  93., 370., 182., 650., 102., 290.,  84., 242.,
       129.,  30., 244., 600., 255.,  98., 275., 121.,  63., 700.,  87.,
       101., 495.,  67.,  73., 260., 108.,  58.,  48., 164., 170.,  83.,
        90., 166., 124.,  55.,  59., 127., 214., 240.,  72.,  60., 138.,
        42., 280., 140., 155., 123., 279., 192., 304., 330., 150., 207.,
       436.,  78.,  54.,  89., 143., 105., 132., 480.,  56., 159., 300.,
       376., 117.,  71., 490., 173.,  46., 228., 308., 236., 570., 380.,
       296., 156., 103.,  45.,  65.,  53., 360.,  62., 218., 178., 239.,
       405., 148., 190., 149., 153., 162., 230.,  86., 234., 246., 500.,
       186., 119., 107., 209., 208., 243.,  40., 250., 311., 400., 161.,
       196., 324., 157., 145., 181.,  26., 211.,   9., 205.,  36.,  61.,
       146., 292., 142., 350., 496., 253.])
```

In [41]:
```
# Count is only Int not float

df["Loan_Amount_Term"].unique()  # we are fill the missing value then we cahnged fl
```

Out[41]:
```
array([360., 120., 240.,  nan, 180.,  60., 300., 480.,  36.,  84.,  12.])
```

In [43]:
```
# Loan_Amount also Discrete count

df["Loan_Amount_Term"].value_counts()
```

Out[43]:
```
Loan_Amount_Term
360.0    512
180.0     44
480.0     15
300.0     13
240.0      4
84.0       4
120.0      3
60.0       2
36.0       2
12.0       1
Name: count, dtype: int64
```

In [45]:
```
# Discrete

df["Credit_History"].unique()
```

Out[45]:
```
array([ 1.,  0., nan])
```

In [47]:
```
# Insted of 1,0 we are replaced it 1 for Yes & 0 for No

df["Credit_History"].replace({1:"Yes",0:"No"},inplace=True)
```

```
C:\Users\WELCOME\AppData\Local\Temp\ipykernel_5640\101465523.py:3: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df["Credit_History"].replace({1:"Yes",0:"No"},inplace=True)
```

In [49]:
```
# Discrete

df["Property_Area"].unique()
```

Out[49]:
```
array(['Urban', 'Rural', 'Semiurban'], dtype=object)
```

In [51]:
```
df["Property_Area"].value_counts()
```

Out[51]:
```
Property_Area
Semiurban    233
Urban        202
Rural        179
Name: count, dtype: int64
```

In [53]:
```
# Discrete

df["Loan_Status"].unique()
```

Out[53]:
```
array(['Y', 'N'], dtype=object)
```

In [55]:
```
df["Loan_Status"].value_counts()
```

Out[55]:
```
Loan_Status
Y    422
N    192
Name: count, dtype: int64
```

In [57]:
```
df.columns.tolist()
```

Out[57]:
```
['Loan_ID',
 'Gender',
 'Married',
 'Dependents',
 'Education',
 'Self_Employed',
 'ApplicantIncome',
 'CoapplicantIncome',
 'LoanAmount',
 'Loan_Amount_Term',
 'Credit_History',
 'Property_Area',
 'Loan_Status']
```

In [59]:
```
# Loan is sanctioned by both 'ApplicantIncome' and CoapplicantIncome'

df["overall_income"] = df["ApplicantIncome"] + df["CoapplicantIncome"]
df.drop(columns = ["ApplicantIncome","CoapplicantIncome"],inplace = True)
df
```

Out[59]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	LoanAmount	Lo
0	LP001002	Male	No	0	Graduate	No	NaN	
1	LP001003	Male	Yes	1	Graduate	No	128.0	
2	LP001005	Male	Yes	0	Graduate	Yes	66.0	
3	LP001006	Male	Yes	0	Not Graduate	No	120.0	
4	LP001008	Male	No	0	Graduate	No	141.0	
...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	71.0	
610	LP002979	Male	Yes	3+	Graduate	No	40.0	
611	LP002983	Male	Yes	1	Graduate	No	253.0	
612	LP002984	Male	Yes	2	Graduate	No	187.0	
613	LP002990	Female	No	0	Graduate	Yes	133.0	

614 rows × 12 columns

In [61]:
```
continous = ["overall_income","LoanAmount"]


# Count means No. of
count = ["Dependents","Loan_Amount_Term"]

discrete = ["Gender","Married","Education","Self_Employed","Credit_History","Proper
```

In [63]:
```
df[continous].describe()
```

Out[63]:

	overall_income	LoanAmount
count	614.000000	592.000000
mean	7024.705081	146.412162
std	6458.663872	85.587325
min	1442.000000	9.000000
25%	4166.000000	100.000000
50%	5416.500000	128.000000
75%	7521.750000	168.000000
max	81000.000000	700.000000

In [65]:
```
df[discrete].describe()   # top = Mode
```

Out[65]:

	Gender	Married	Education	Self_Employed	Credit_History	Property_Area	Loan_S
count	601	611	614	582	564	614	
unique	2	2	2	2	2	3	
top	Male	Yes	Graduate	No	Yes	Semiurban	
freq	489	398	480	500	475	233	

top means Most common category

freq means this is the number of times the top value appears

In [68]:
```
df[count].describe()
```

Out[68]:

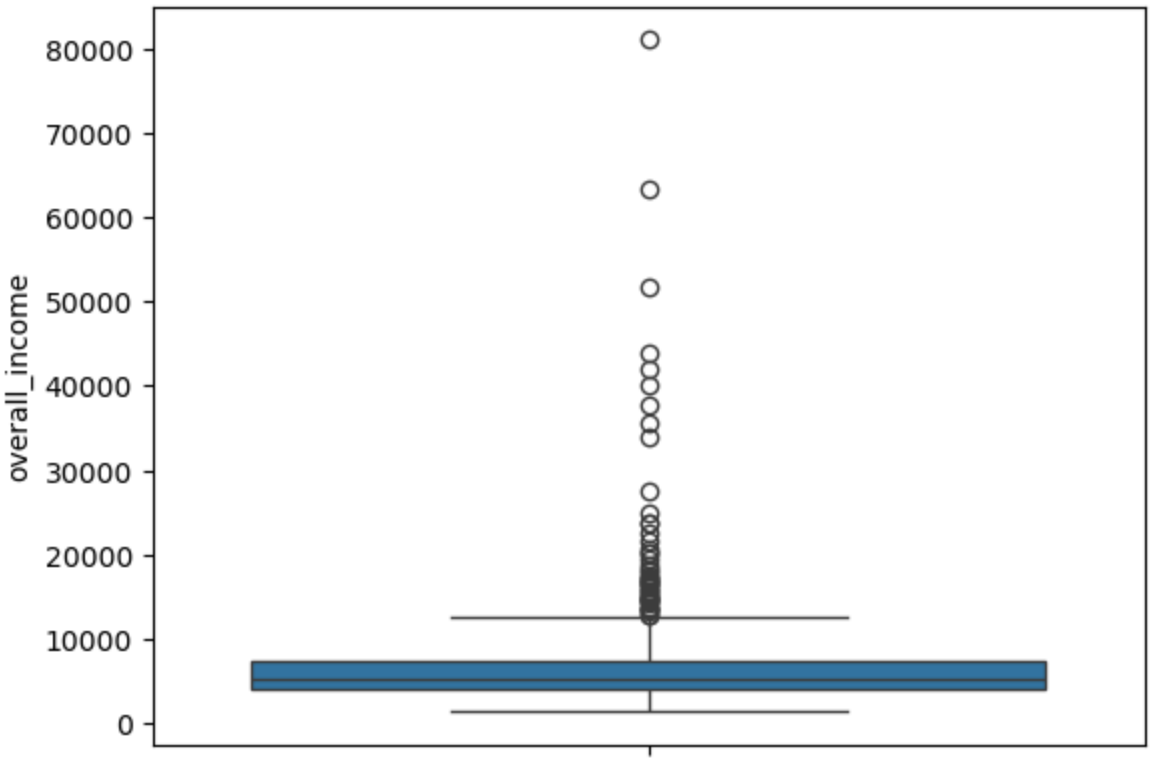
	Loan_Amount_Term
count	600.00000
mean	342.00000
std	65.12041
min	12.00000
25%	360.00000
50%	360.00000
75%	360.00000
max	480.00000

In [70]:
```
# Missing value

df.isnull().sum()
```

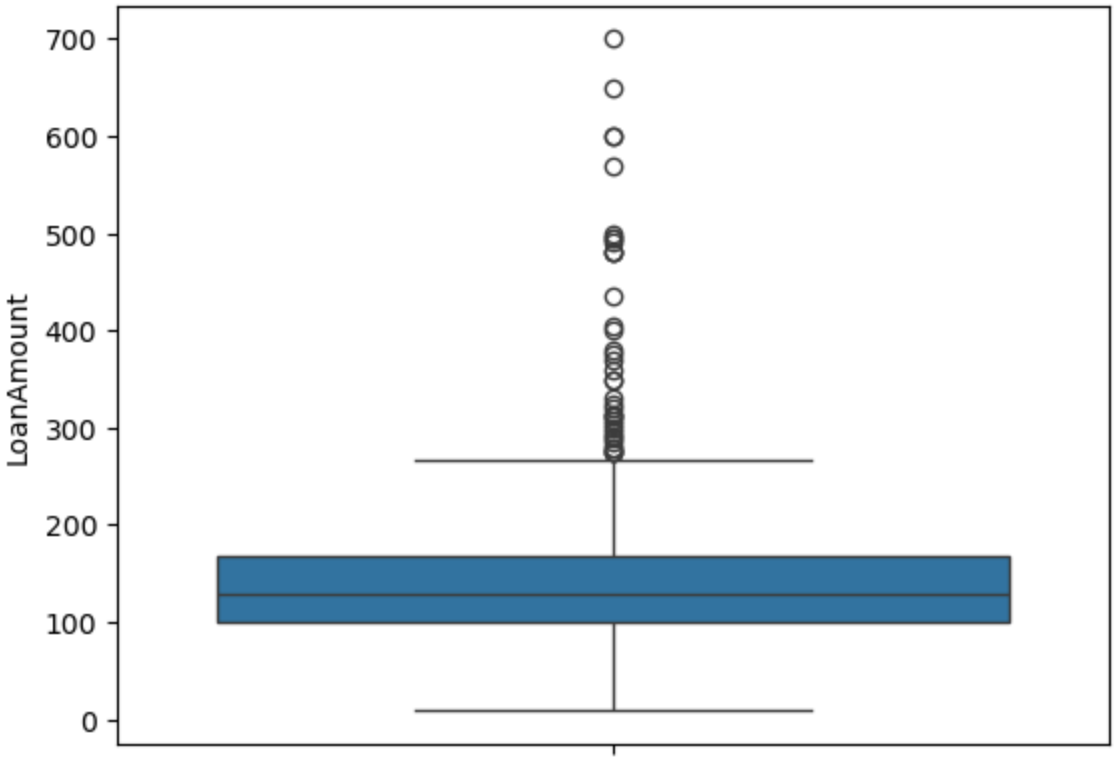
Out[70]:
```
Loan_ID              0
Gender              13
Married              3
Dependents          15
Education            0
Self_Employed       32
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
Loan_Status          0
overall_income       0
dtype: int64
```

BOX PLOT

In [72]:
```
sns.boxplot(df["overall_income"])
plt.show()
```



In [74]:
```
sns.boxplot(df["LoanAmount"])
plt.show()
```



skewness is only meaningful for numerical (continous or count) variables

In [77]:
```
df[continous].skew()   # right skewed
```

Out[77]:
```
overall_income    5.633449
LoanAmount        2.677552
dtype: float64
```

In [79]:
```
df.duplicated().sum()
```

Out[79]:
```
0
```

Data Cleaning

Treat wrong data

In [83]:
```
# in replace use dictionary why it is using .replace because this is not string tha

df["Dependents"].replace({"3+":3},inplace=True)
```

```
C:\Users\WELCOME\AppData\Local\Temp\ipykernel_5640\3039787444.py:3: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df["Dependents"].replace({"3+":3},inplace=True)
```

check null values sum

In [86]:
```
df.isnull().sum()
```

Out[86]:
```
Loan_ID             0
Gender             13
Married             3
Dependents         15
Education           0
Self_Employed      32
LoanAmount         22
Loan_Amount_Term   14
Credit_History     50
Property_Area       0
Loan_Status         0
overall_income      0
dtype: int64
```

Treat missing values(nan)

- which columns name value are 0 simply left it

For categorical use MODE

For Contionous Use MEAN OR MEDIAN (SALARY,AGE HEIGHT,TEMPRATURE)

- USE MEAN WHEN DATA IS EVENLY DISTRIBUTED,
- MEDIAN WHEN DATA HAS OUTLIERS OR SKEWED DISRTRIBUTION

For Count Use MODE (when skewed distribution)

For Count Use MEAN (when Normal Distribution) COUNT (NUMBER OF ITEMS SOLD, NUMBER OF VISITS) USED MEDIAN (SOMETIME MODE)

In [89]:
```
# In discrete we use Mode
# fillna() is fill NaN wuth a fixed Value (Replaces all missing values in the colum

df["Gender"] = df["Gender"].fillna(df["Gender"].mode()[0])
df["Married"] = df["Married"].fillna(df["Married"].mode()[0])

# Discrete count
df["Dependents"] = df["Dependents"].fillna(df["Dependents"].mode()[0])
df["Self_Employed"] = df["Self_Employed"].fillna(df["Self_Employed"].mode()[0])

# LoanAmount, Loan_Amount_Term, Credit_History this risk factor thats why we use dr
# dropna() used for Remove rows with any missing values (Delets all rows that conta

df = df.dropna()
```

After Treat the Missing values check missing value is available or not

In [92]:
```
df.isnull().sum()
```

Out[92]:
```
Loan_ID            0
Gender             0
Married            0
Dependents         0
Education          0
Self_Employed      0
LoanAmount         0
Loan_Amount_Term   0
Credit_History     0
Property_Area      0
Loan_Status        0
overall_income     0
dtype: int64
```

Treat wrong data type

In [95]:
```
df["Dependents"] = df["Dependents"].astype("int")
df["Loan_Amount_Term"] = df["Loan_Amount_Term"].astype("int")
```

```
C:\Users\WELCOME\AppData\Local\Temp\ipykernel_5640\2369623998.py:1: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df["Dependents"] = df["Dependents"].astype("int")
C:\Users\WELCOME\AppData\Local\Temp\ipykernel_5640\2369623998.py:2: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df["Loan_Amount_Term"] = df["Loan_Amount_Term"].astype("int")
```

Treat the Outliers

In [98]:
```
# Retrain the outliers (Keep them as it is)  this is genuine data
```

Drop the unimportant column

In [101…
```
df.drop(columns = ["Loan_ID"],inplace = True)   ## Drop are used thats why it shows
```

```
C:\Users\WELCOME\AppData\Local\Temp\ipykernel_5640\2331071704.py:1: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df.drop(columns = ["Loan_ID"],inplace = True)   ## Drop are used thats why it show
s error
```

Treat the Duplicate

To check duplicate

df.duplicated().sum()

In [105…
```
# There is no Duplicate
```

In [107…
```
df  # This is our Clean data
```

Out[107…

	Gender	Married	Dependents	Education	Self_Employed	LoanAmount	Loan_Amount
1	Male	Yes	1	Graduate	No	128.0	
2	Male	Yes	0	Graduate	Yes	66.0	
3	Male	Yes	0	Not Graduate	No	120.0	
4	Male	No	0	Graduate	No	141.0	
5	Male	Yes	2	Graduate	Yes	267.0	
...	...	...	...	...	...	...	
609	Female	No	0	Graduate	No	71.0	
610	Male	Yes	3	Graduate	No	40.0	
611	Male	Yes	1	Graduate	No	253.0	
612	Male	Yes	2	Graduate	No	187.0	
613	Female	No	0	Graduate	Yes	133.0	

529 rows × 11 columns

Export DataFrames to excel workbook

- After cleaned data you can used in PowerBI or Tableau

In [110…

```
df.to_excel("Home_Loan_cleaned_data.xlsx",index=False)
```

Data Analysis

- Measures + Plots
- Univariate, Bivariate, Multivariate

Applying various questions or logics on dataset

Univariate Measures

- Categorical = .value_counts()
- Continous = .describe()

In [114…

```
df["Gender"].value_counts()
```

Out[114…
```
Gender
Male      434
Female     95
Name: count, dtype: int64
```

In [116…
```
df["Married"].value_counts()
```

Out[116…
```
Married
Yes    341
No     188
Name: count, dtype: int64
```

In [118…
```
df["Dependents"].value_counts()
```

Out[118…
```
Dependents
0    307
2     92
1     85
3     45
Name: count, dtype: int64
```

In [120…
```
df["Education"].value_counts()
```

Out[120…
```
Education
Graduate        421
Not Graduate    108
Name: count, dtype: int64
```

In [122…
```
df["Self_Employed"].value_counts()
```

Out[122…
```
Self_Employed
No     459
Yes     70
Name: count, dtype: int64
```

In [124…
```
df["LoanAmount"].describe()  # To check Average, Minimum, Maximum Loan Amount
```

Out[124…
```
count    529.000000
mean     145.852552
std       84.108409
min        9.000000
25%      100.000000
50%      128.000000
75%      167.000000
max      700.000000
Name: LoanAmount, dtype: float64
```

In [126…
```
df["Loan_Amount_Term"].value_counts()
```

Out[126…
```
Loan_Amount_Term
360    452
180     41
480     14
300     10
120      3
84       3
60       2
240      2
36       2
Name: count, dtype: int64
```

In [128…
```
df["Credit_History"].value_counts()
```

Out[128…
```
Credit_History
Yes    450
No      79
Name: count, dtype: int64
```

In [130…
```
df["Property_Area"].value_counts()
```

Out[130…
```
Property_Area
Semiurban    209
Urban        165
Rural        155
Name: count, dtype: int64
```

In [132…
```
df["Loan_Status"].value_counts()
```

Out[132…
```
Loan_Status
Y    366
N    163
Name: count, dtype: int64
```

In [134…
```
df["overall_income"].describe()
```

Out[134…
```
count      529.000000
mean      7050.217240
std       6589.393544
min       1442.000000
25%       4166.000000
50%       5332.000000
75%       7542.000000
max      81000.000000
Name: overall_income, dtype: float64
```

Bivariate Measures

- crosstab (2 discrete variable)
- correlation (Continuous + Continuous)
- groupby (1 discrete + 1 continous)

In [137…
```
pd.crosstab(df["Gender"],df["Loan_Status"],margins = True)
```

Out[137…

Loan_Status	N	Y	All
Gender			
Female	34	61	95
Male	129	305	434
All	163	366	529

In [139…

```
# percentage

pd.crosstab(df["Gender"],df["Loan_Status"],margins = True,normalize = True)   # nor
```

Out[139…

Loan_Status	N	Y	All
Gender			
Female	0.064272	0.115312	0.179584
Male	0.243856	0.576560	0.820416
All	0.308129	0.691871	1.000000

In [141…

```
#To check Mean

df.groupby("Loan_Status")["LoanAmount"].describe().T  # when you use .groupby() you

# .T (Transpose) means Column wise
```

Out[141…

Loan_Status	N	Y
count	163.000000	366.000000
mean	150.466258	143.797814
std	87.048112	82.804292
min	9.000000	17.000000
25%	100.000000	100.250000
50%	128.000000	128.000000
75%	173.000000	161.750000
max	570.000000	700.000000

In [143…

```
df.groupby("Loan_Status")["overall_income"].describe().T
```

Out[143...

Loan_Status	N	Y
count	163.000000	366.000000
mean	7274.269939	6950.434208
std	7796.001701	5983.541784
min	1442.000000	1963.000000
25%	4087.500000	4188.750000
50%	5230.000000	5416.500000
75%	7491.000000	7548.000000
max	81000.000000	63337.000000

In [145...
```
pd.crosstab(df["Credit_History"],df["Loan_Status"],margins = True,normalize = True)
```

Out[145...

Loan_Status	N	Y	All
Credit_History			
No	0.136106	0.013233	0.149338
Yes	0.172023	0.678639	0.850662
All	0.308129	0.691871	1.000000

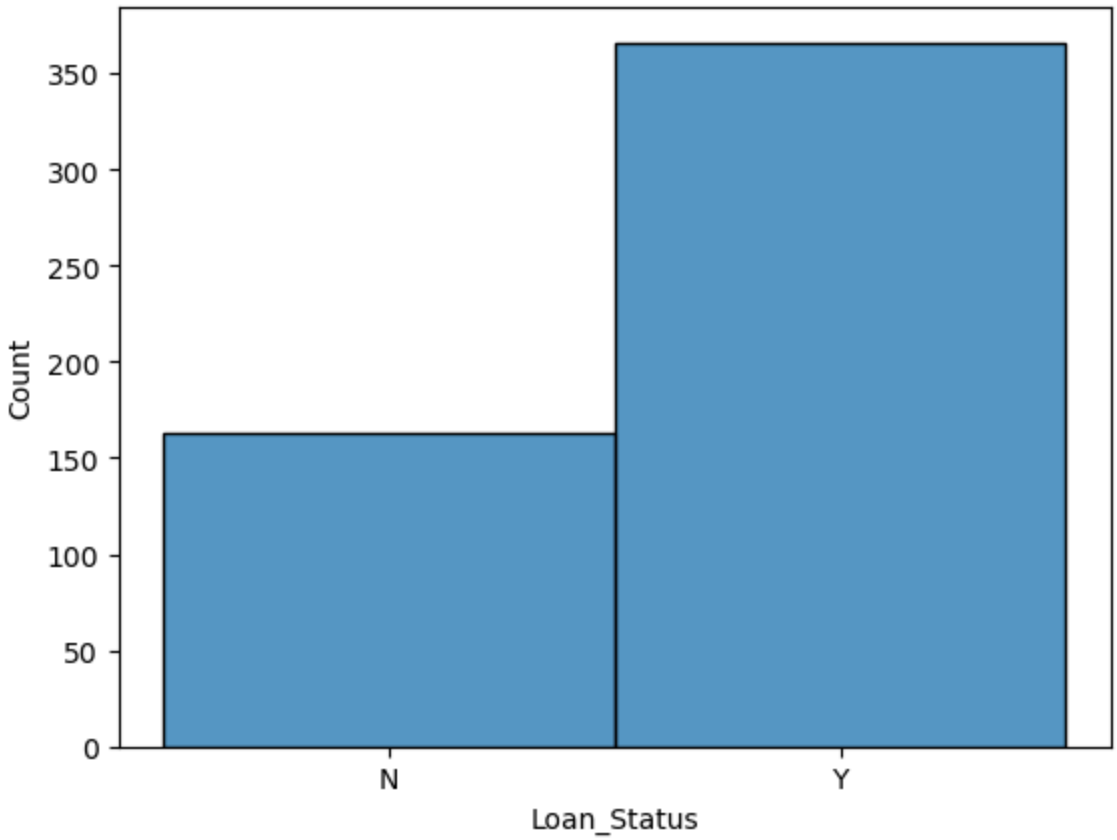
Multivariate Measures

- correlation (All Continuous variable) Ex = df[['Age', 'Heart rate', 'Blood sugar']].corr()
- crosstab (All discrete variable) Ex = pd.crosstab([df['Gender'], df['Region']], df['Purchased'],margins = True,normalize = True) **OR** pd.crosstab([df['Gender'], df['Region']], [df['Purchased'], df['Channel']])
- groupby (2 discrete + 1 continous) Ex = df.groupby(["Gender","Result"])["Systolic blood pressure"].describe().T

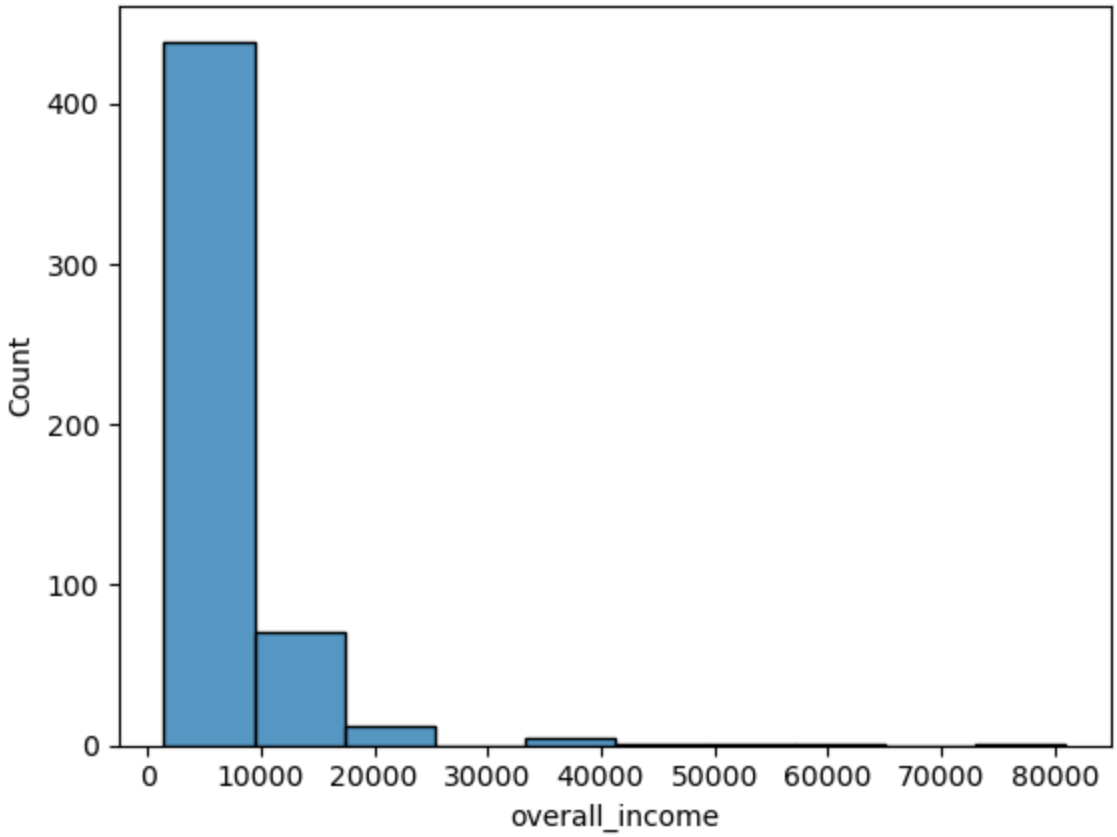
Hist Plot

1 continous

In [149...
```
sns.histplot(df["Loan_Status"],bins = 10)
plt.show()
```



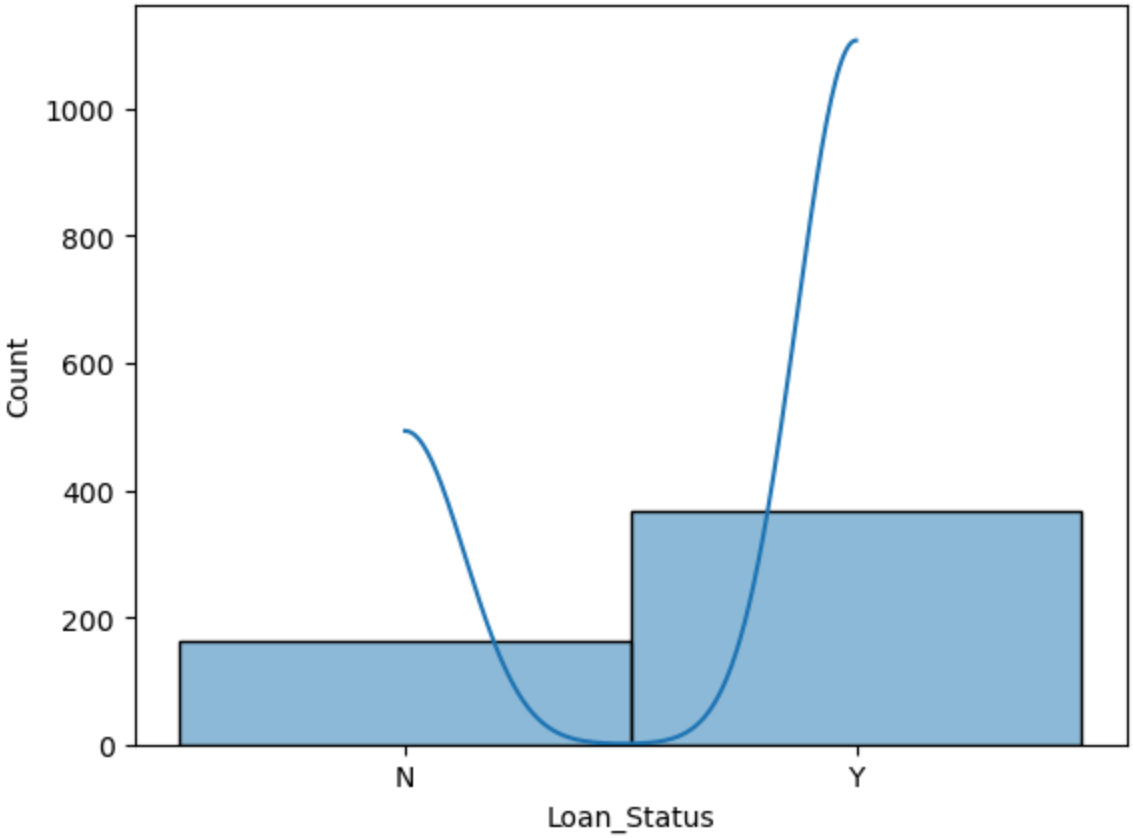
In [151…
```
sns.histplot(df["overall_income"],bins = 10)
plt.show()
```



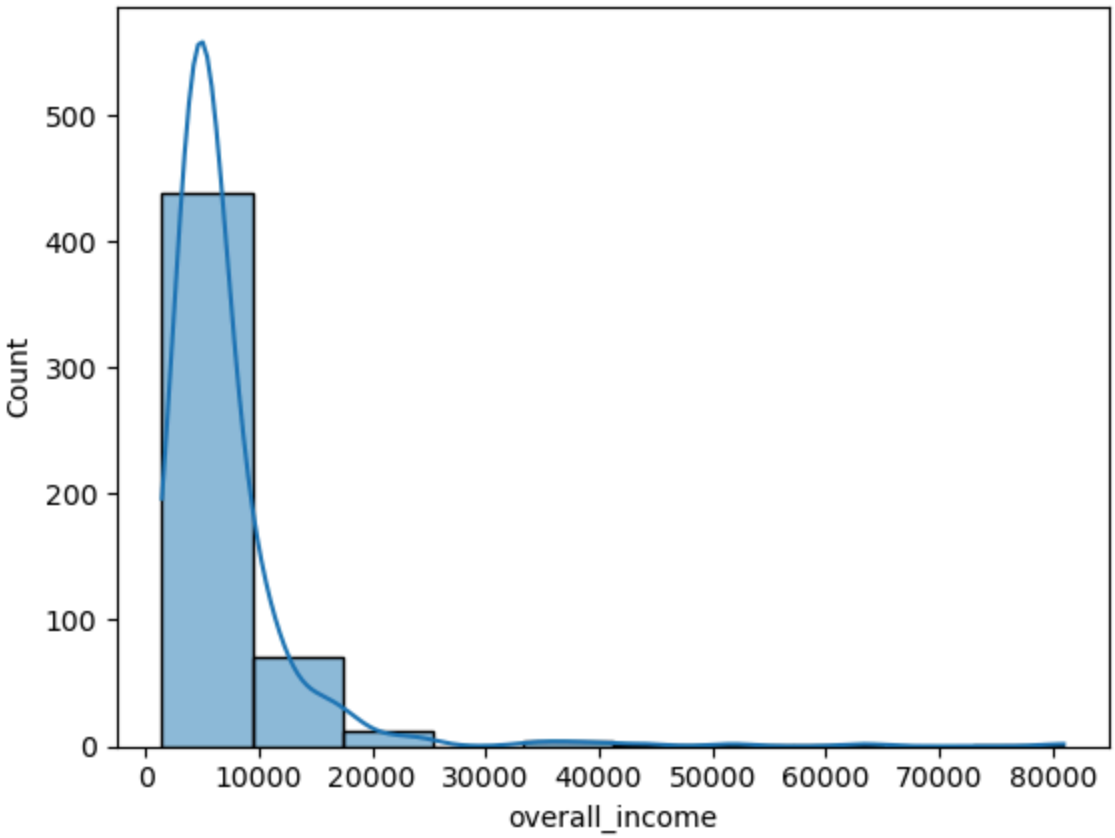
Kde Plot

1 continous

In [154…
```
sns.histplot(df["Loan_Status"],bins = 10,kde = True)   # bins = 10 means 10 Interva
plt.show()
```



In [156…
```
sns.histplot(df["overall_income"],bins = 10,kde = True)
plt.show()
```

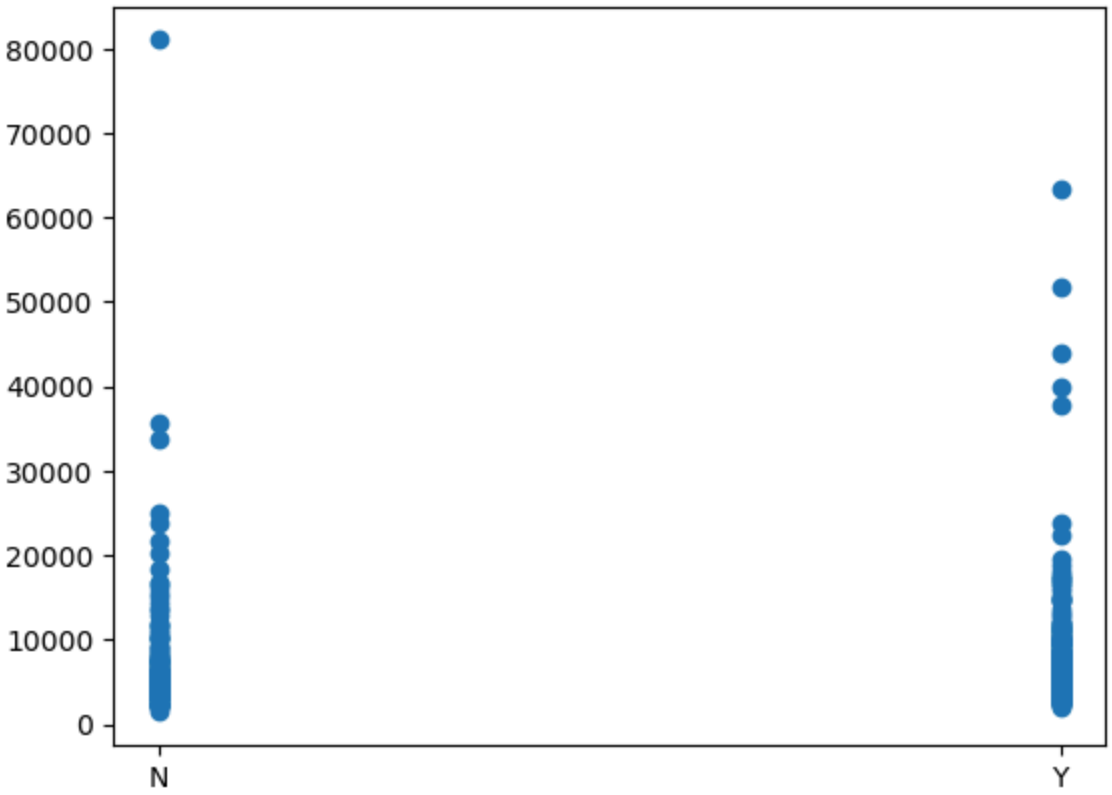


Scatter Plot

2 continous

In [159…

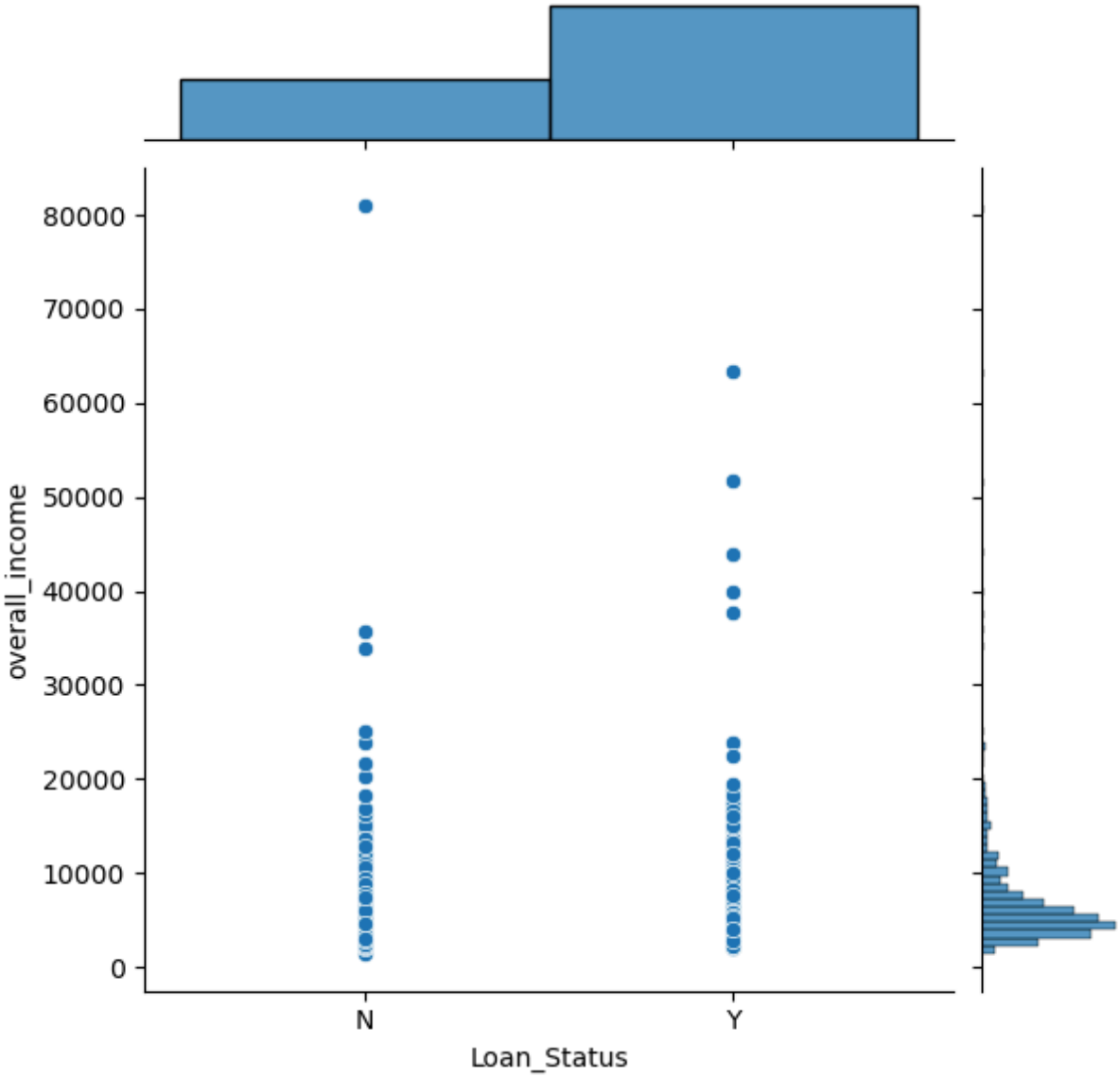
```
plt.scatter(x=df["Loan_Status"],y=df["overall_income"])
plt.show()
```



Joint Plot

2 Continous

In [162…
```
sns.jointplot(x="Loan_Status",y="overall_income",data=df)
plt.show()
```

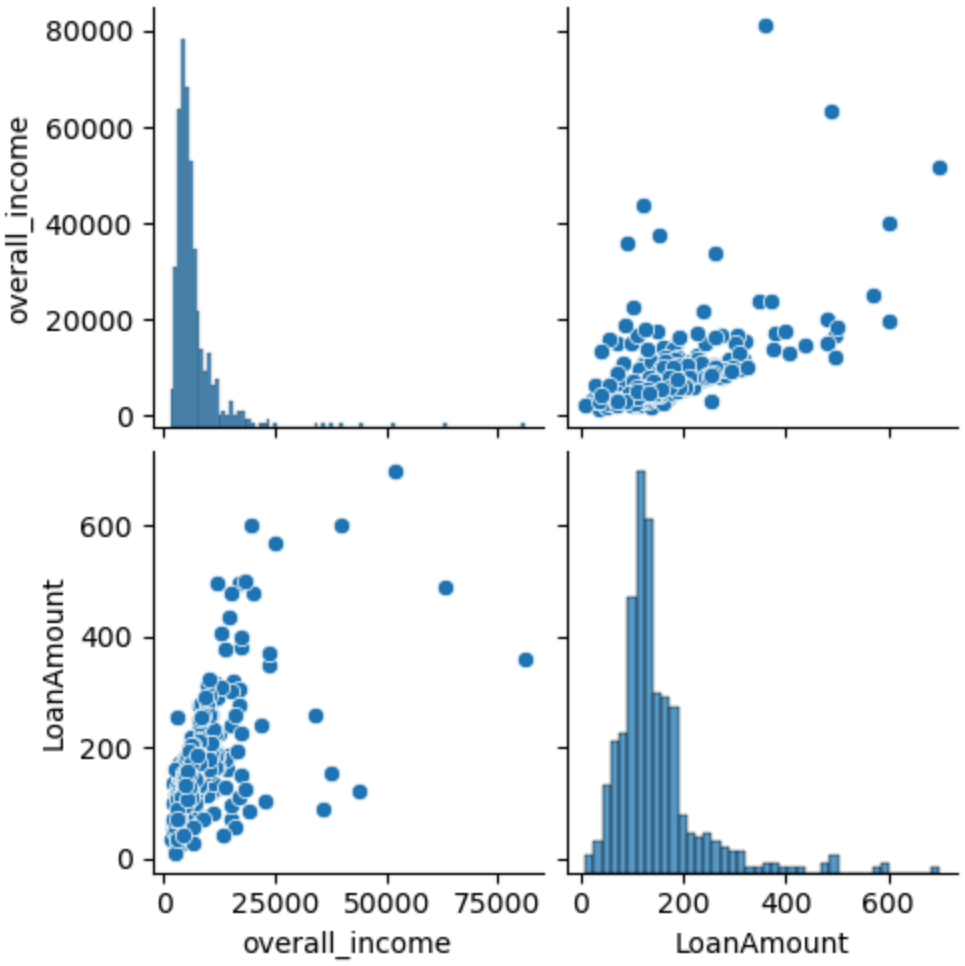


Pair Plot

>2 continous

In [164…

```
sns.pairplot(df,vars=continous)
plt.show()
```

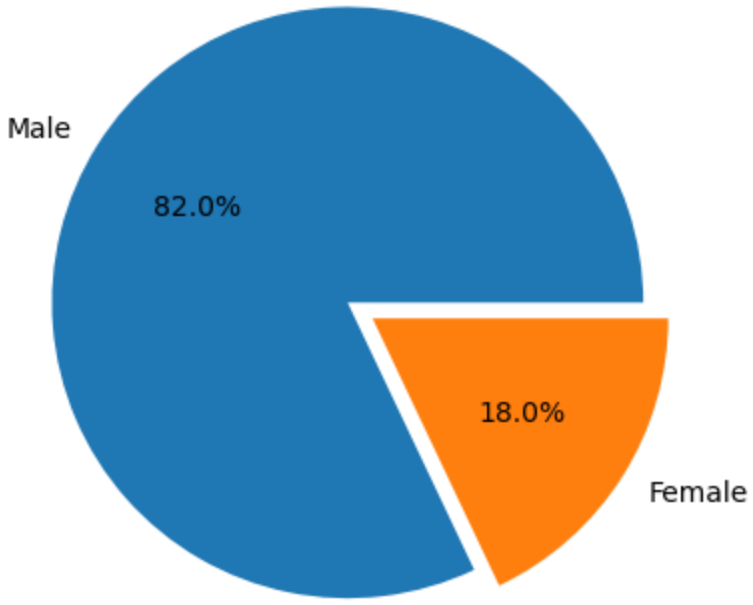


Pie plot

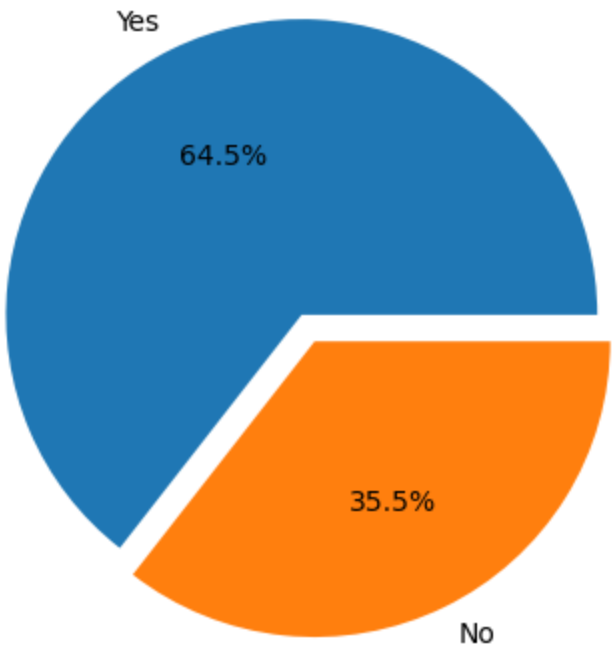
1 discrete

In [167…

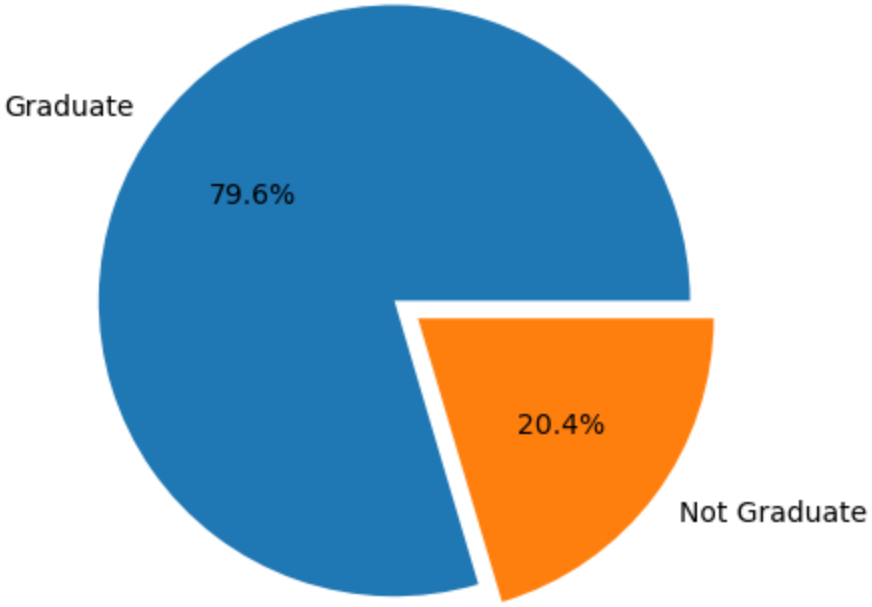
```
plt.pie(x=df["Gender"].value_counts(),
        labels = df["Gender"].value_counts().index.tolist(),autopct = "%0.1f%%",exp
plt.show()
```



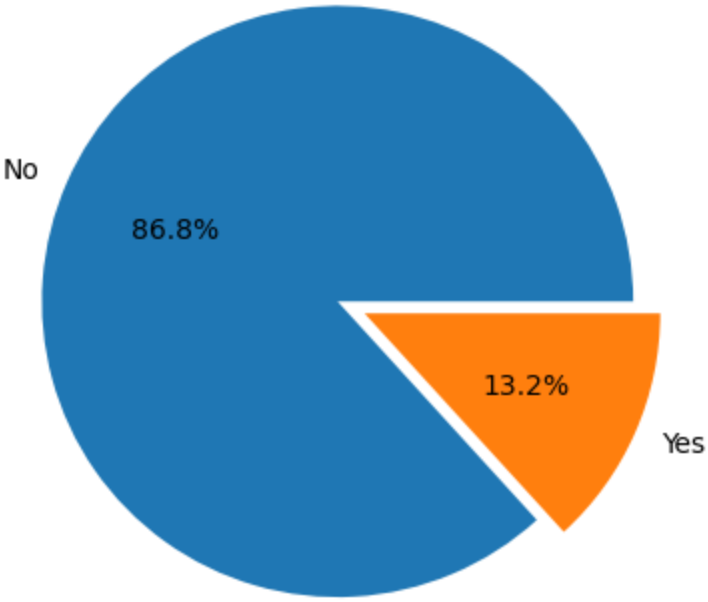
In [168…
```
plt.pie(x=df["Married"].value_counts(),
        labels = df["Married"].value_counts().index.tolist(),autopct = "%0.1f%%",ex
plt.show()
```



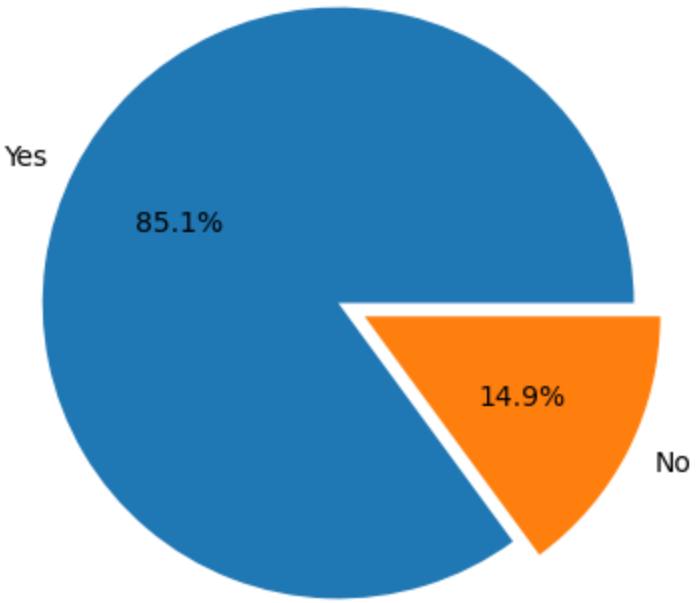
In [169…
```
plt.pie(x=df["Education"].value_counts(),
        labels = df["Education"].value_counts().index.tolist(),autopct = "%0.1f%%",
plt.show()
```



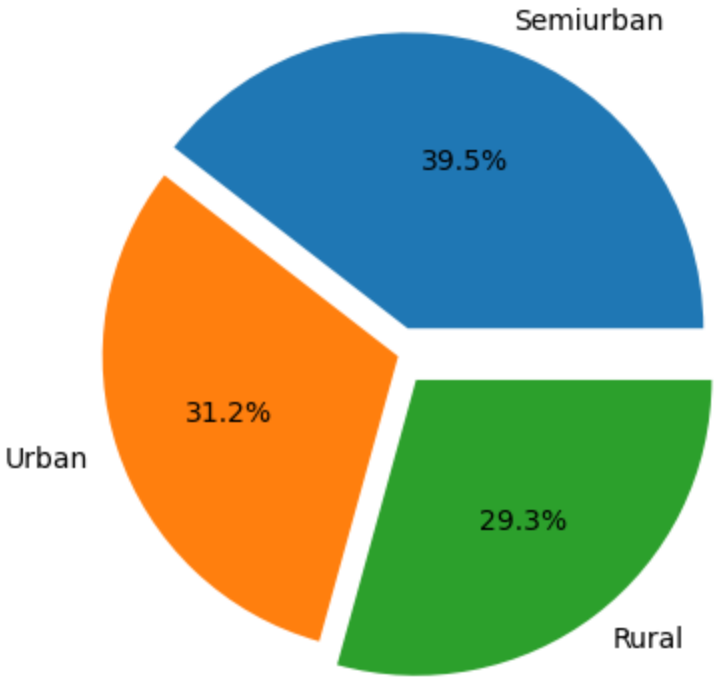
In [170…
```
plt.pie(x=df["Self_Employed"].value_counts(),
        labels = df["Self_Employed"].value_counts().index.tolist(),autopct = "%0.1f
plt.show()
```



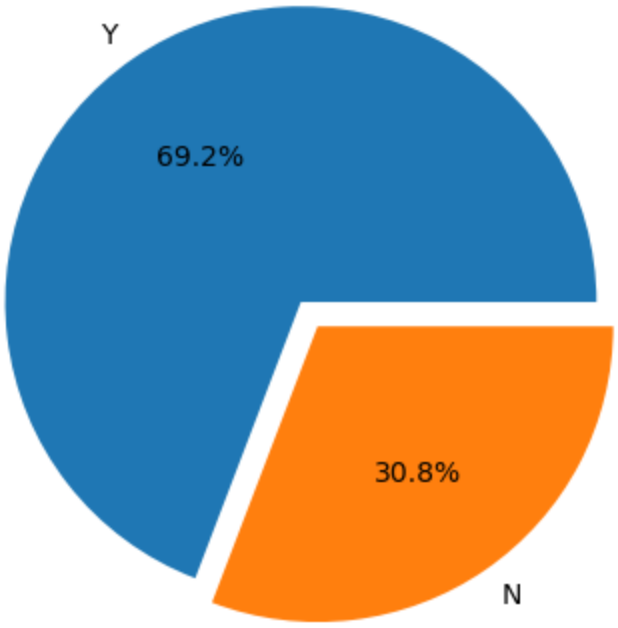
In [171…
```
plt.pie(x=df["Credit_History"].value_counts(),
        labels = df["Credit_History"].value_counts().index.tolist(),autopct = "%0.1
plt.show()
```



In [172…
```
plt.pie(x=df["Property_Area"].value_counts(),
        labels = df["Property_Area"].value_counts().index.tolist(),autopct = "%0.1f
plt.show()
```



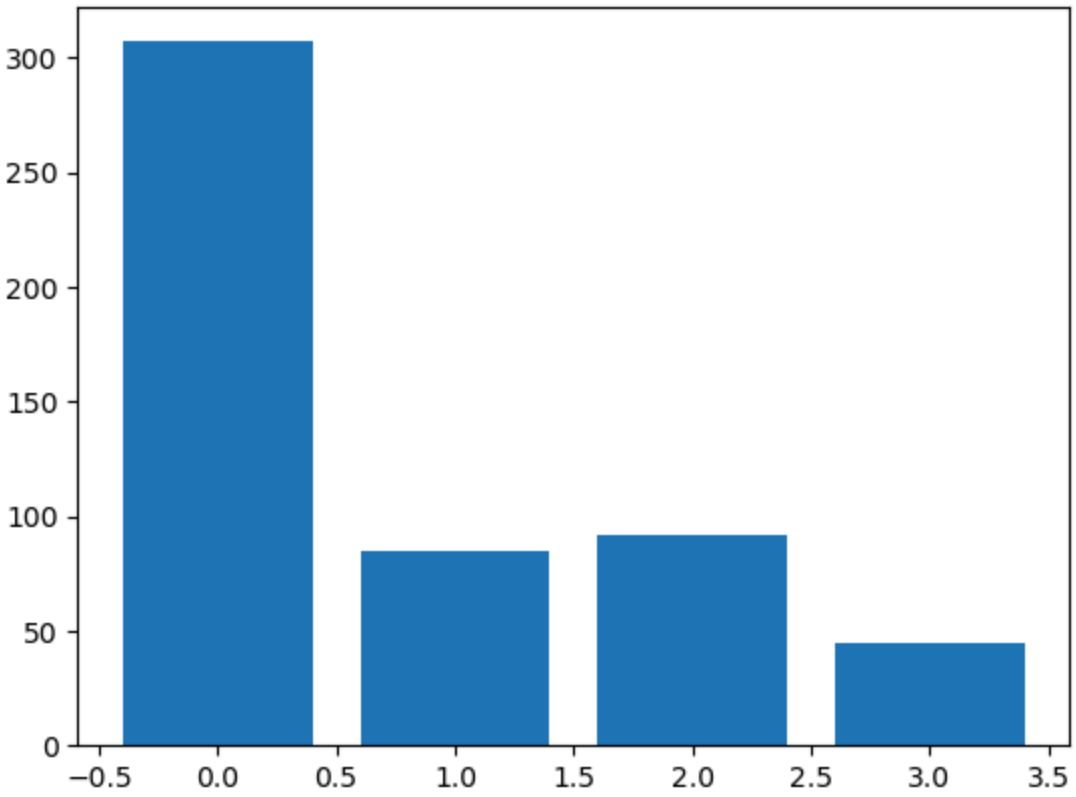
In [173…
```
plt.pie(x=df["Loan_Status"].value_counts(),
        labels = df["Loan_Status"].value_counts().index.tolist(),autopct = "%0.1f%%
plt.show()
```



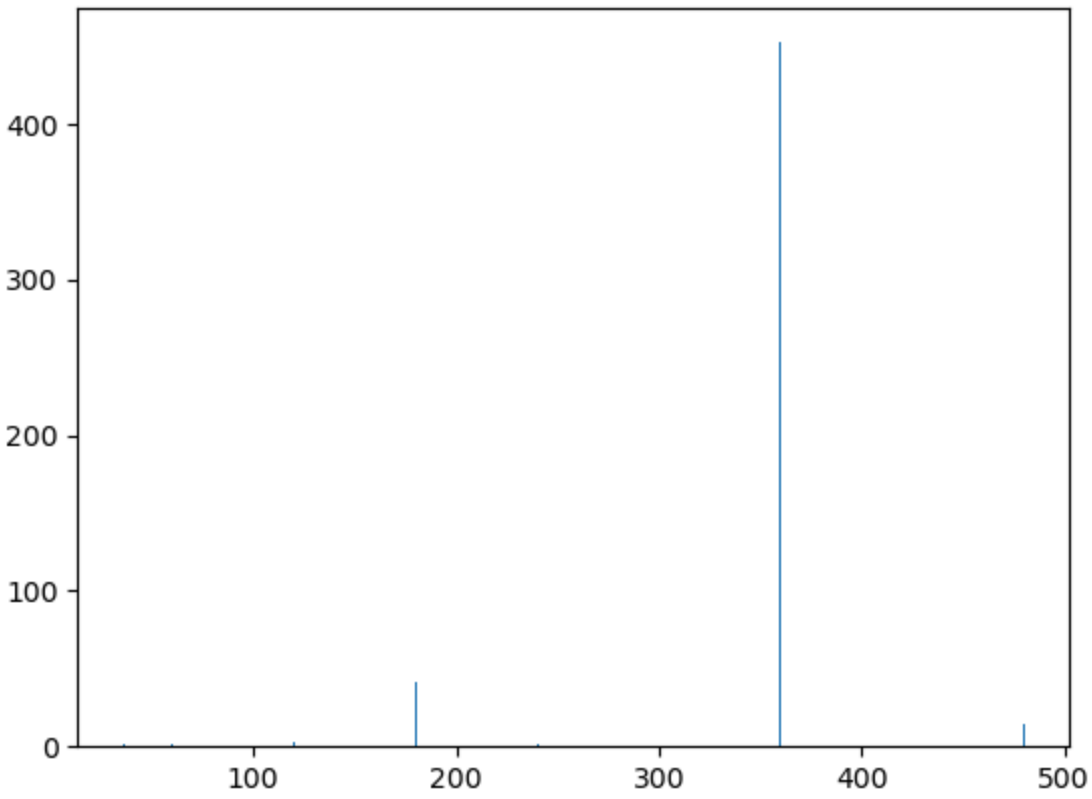
Count Plot

1 Discrete

In [178…
```
plt.bar(df["Dependents"].value_counts().index,df["Dependents"].value_counts())
plt.show()
```



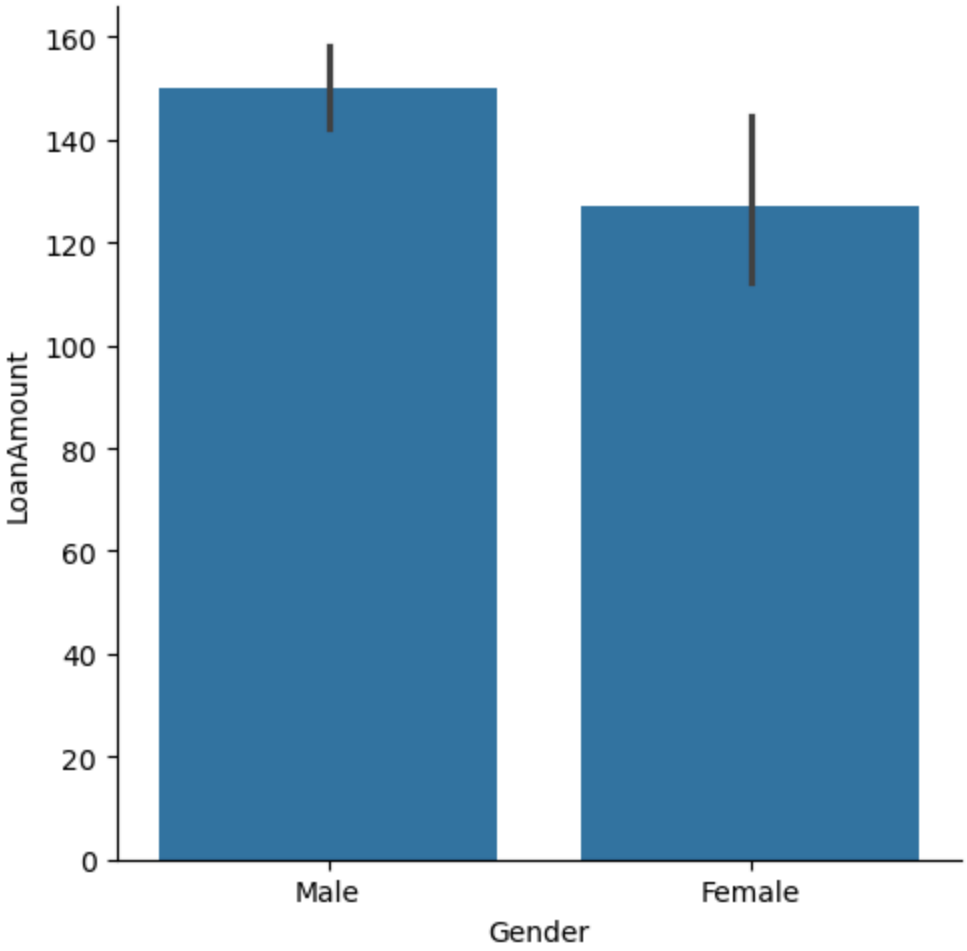
In [181…
```
plt.bar(df["Loan_Amount_Term"].value_counts().index,df["Loan_Amount_Term"].value_co
plt.show()
```



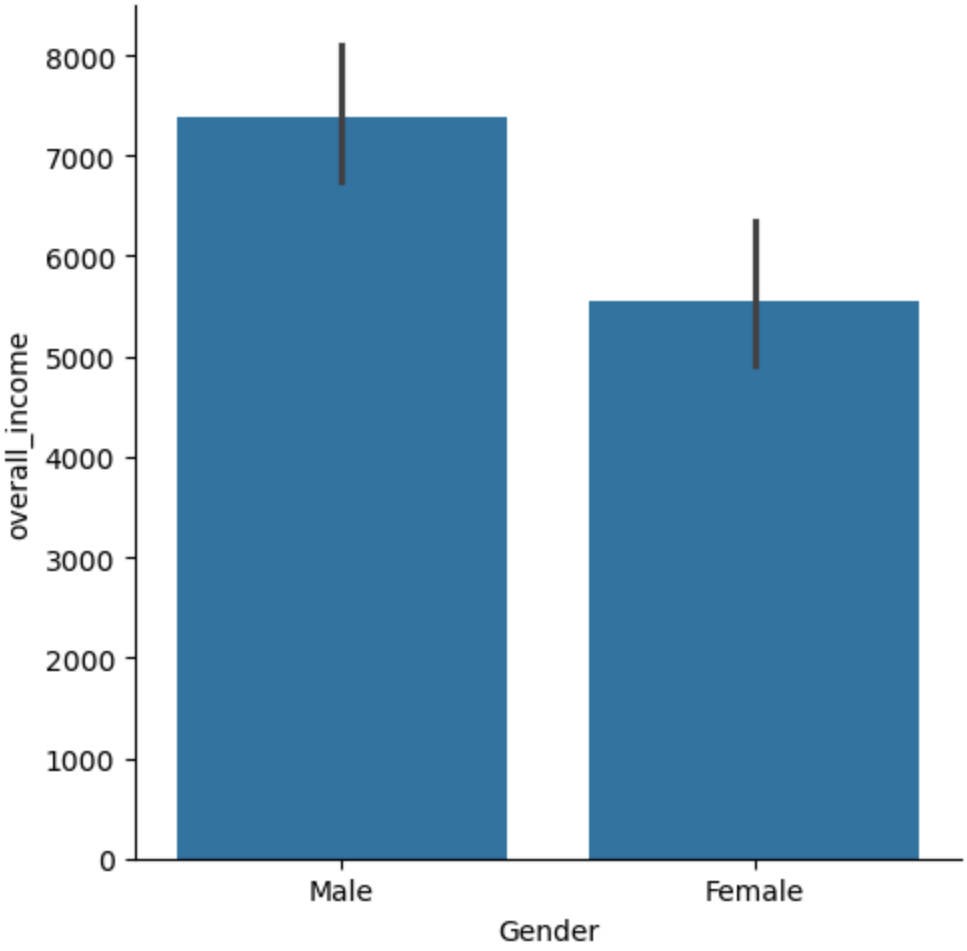
Bar plot

1 Discrete + 1 continous

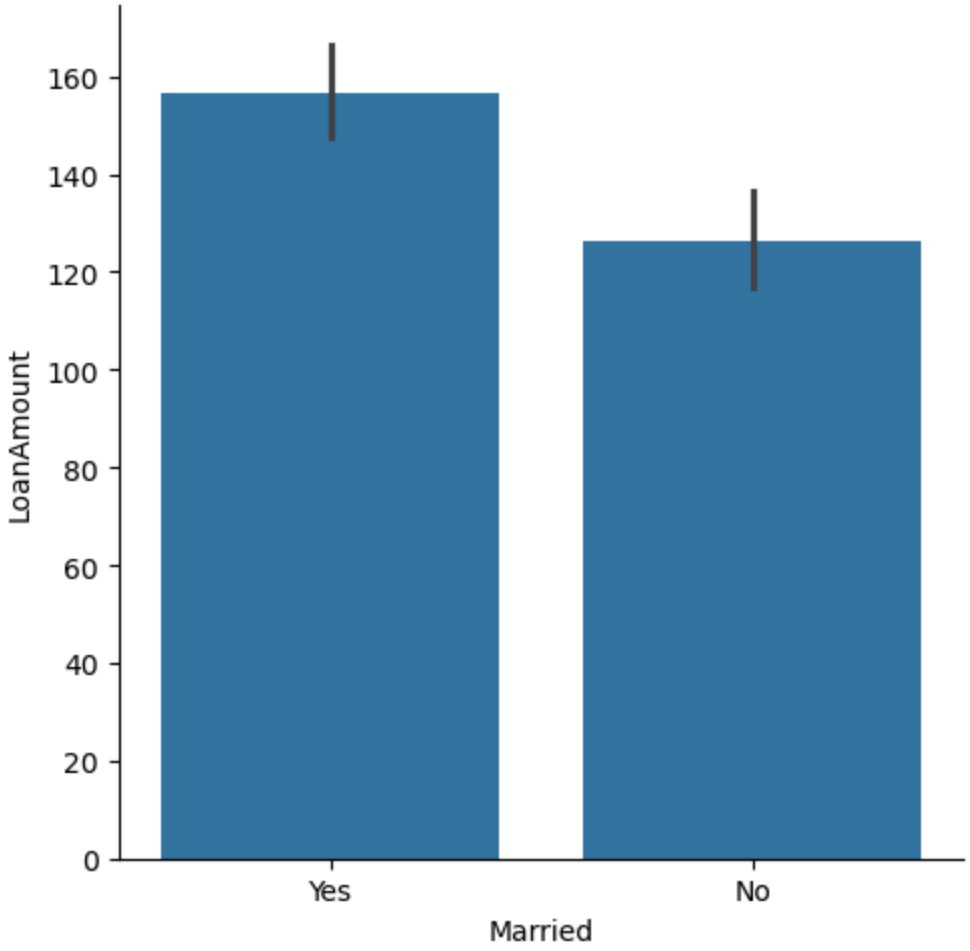
In [186…
```
sns.catplot(x="Gender",y="LoanAmount",data=df,kind="bar")
plt.show()
```



In [187…
```
sns.catplot(x="Gender",y="overall_income",data=df,kind="bar")
plt.show()
```

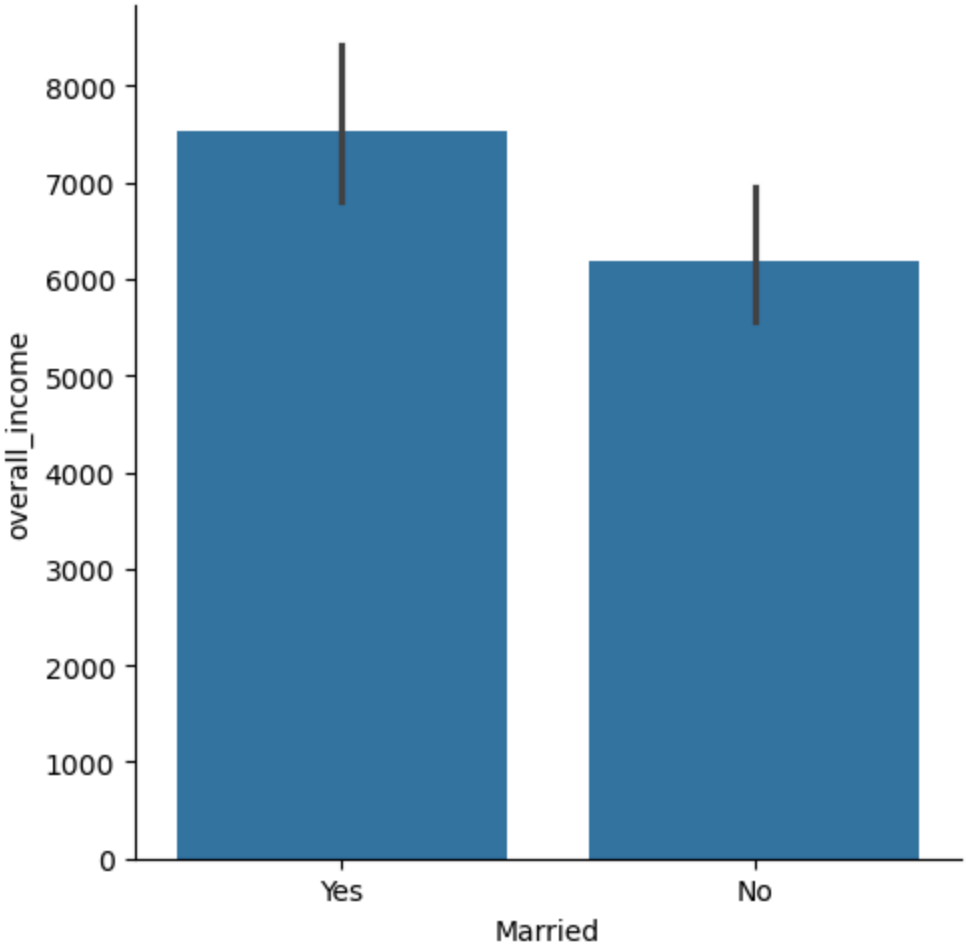


In [189…
```
sns.catplot(x="Married",y="LoanAmount",data=df,kind="bar")
plt.show()
```



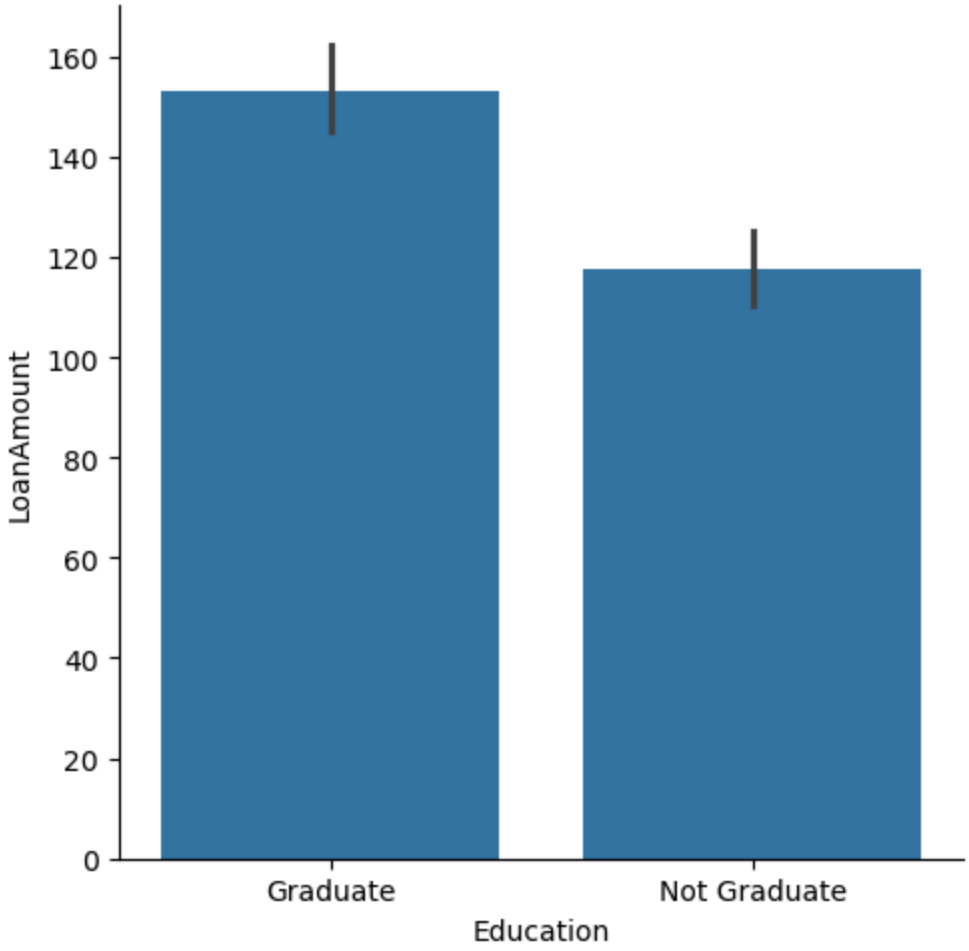
In [191…

```
sns.catplot(x="Married",y="overall_income",data=df,kind="bar")
plt.show()
```



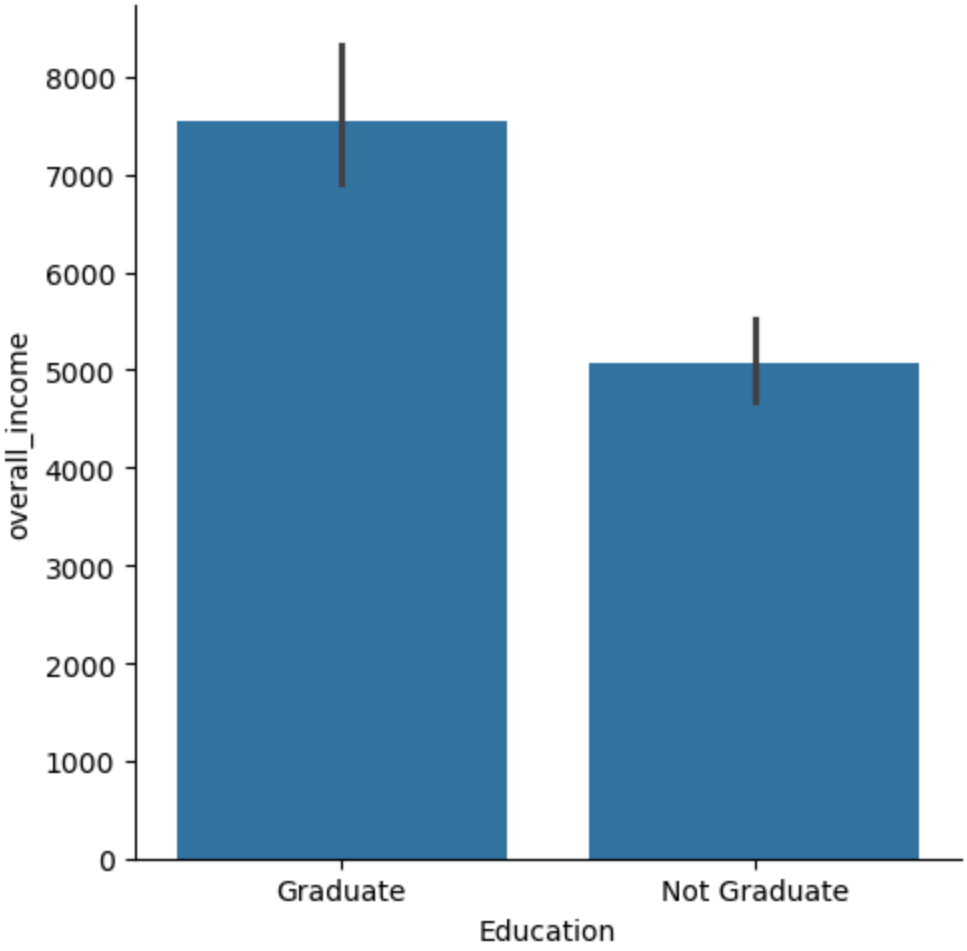
In [192…

```
sns.catplot(x="Education",y="LoanAmount",data=df,kind="bar")
plt.show()
```

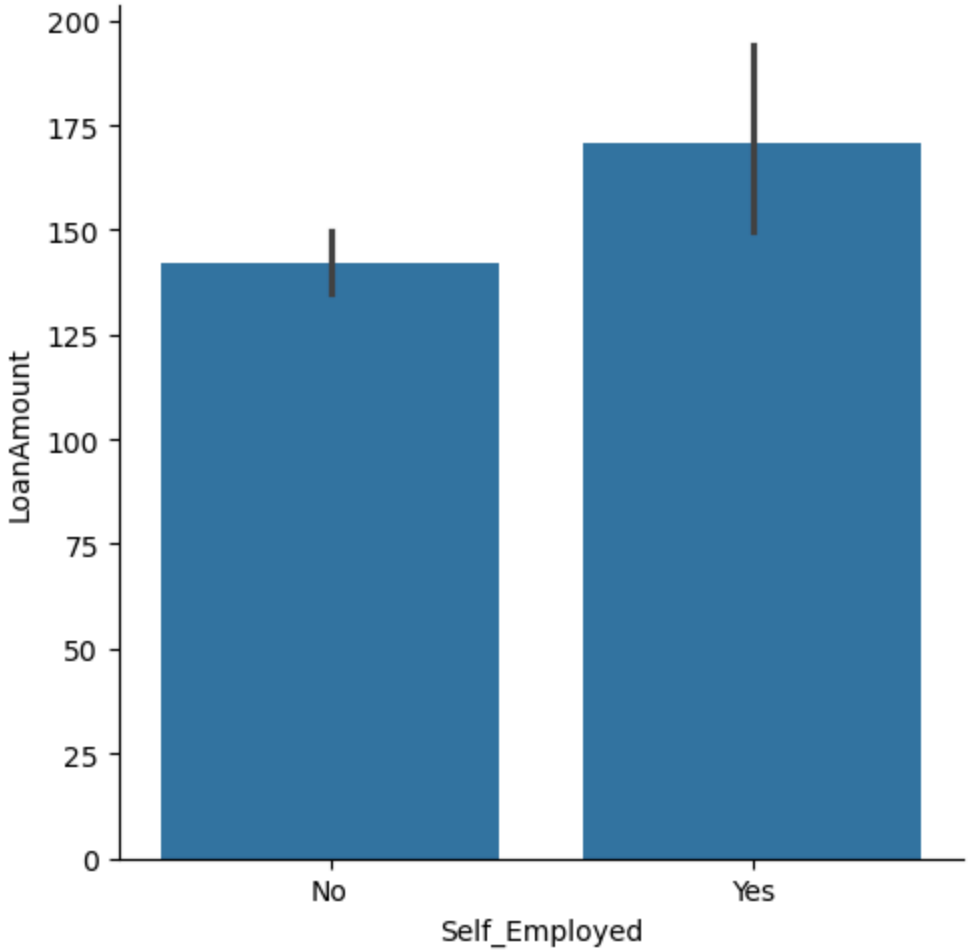


In [194…

```
sns.catplot(x="Education",y="overall_income",data=df,kind="bar")
plt.show()
```

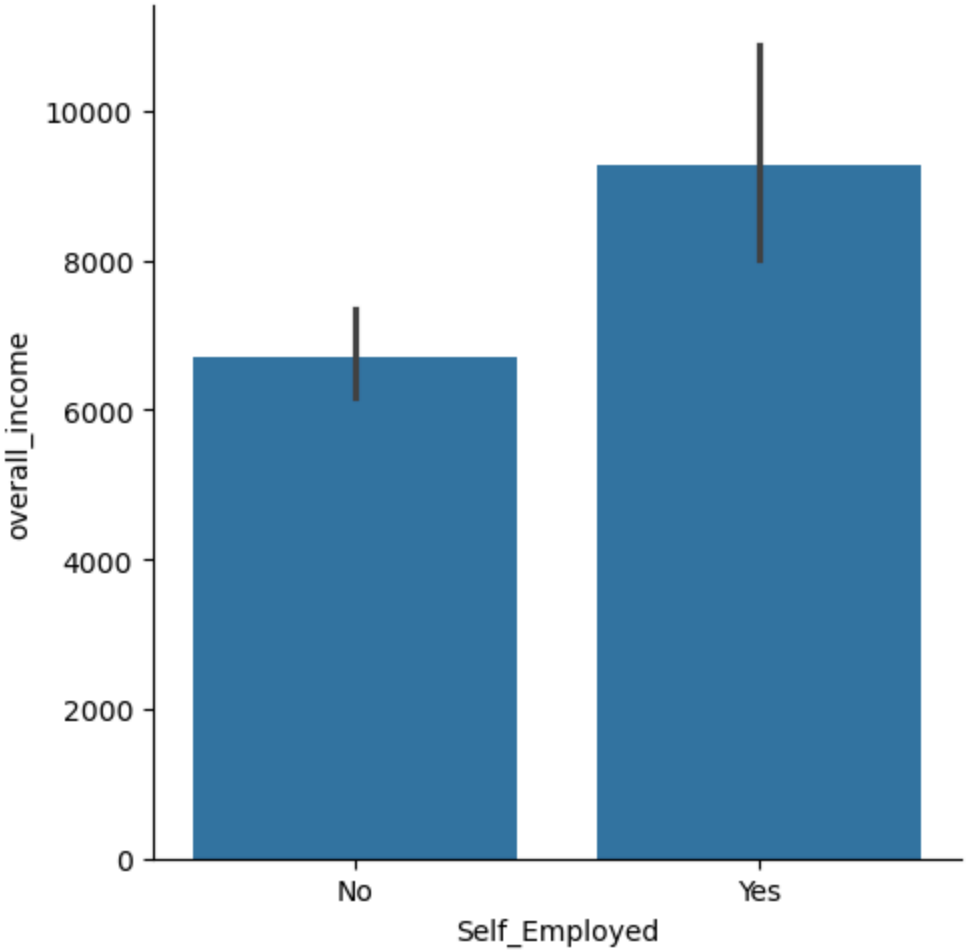


In [195…
```
sns.catplot(x="Self_Employed",y="LoanAmount",data=df,kind="bar")
plt.show()
```



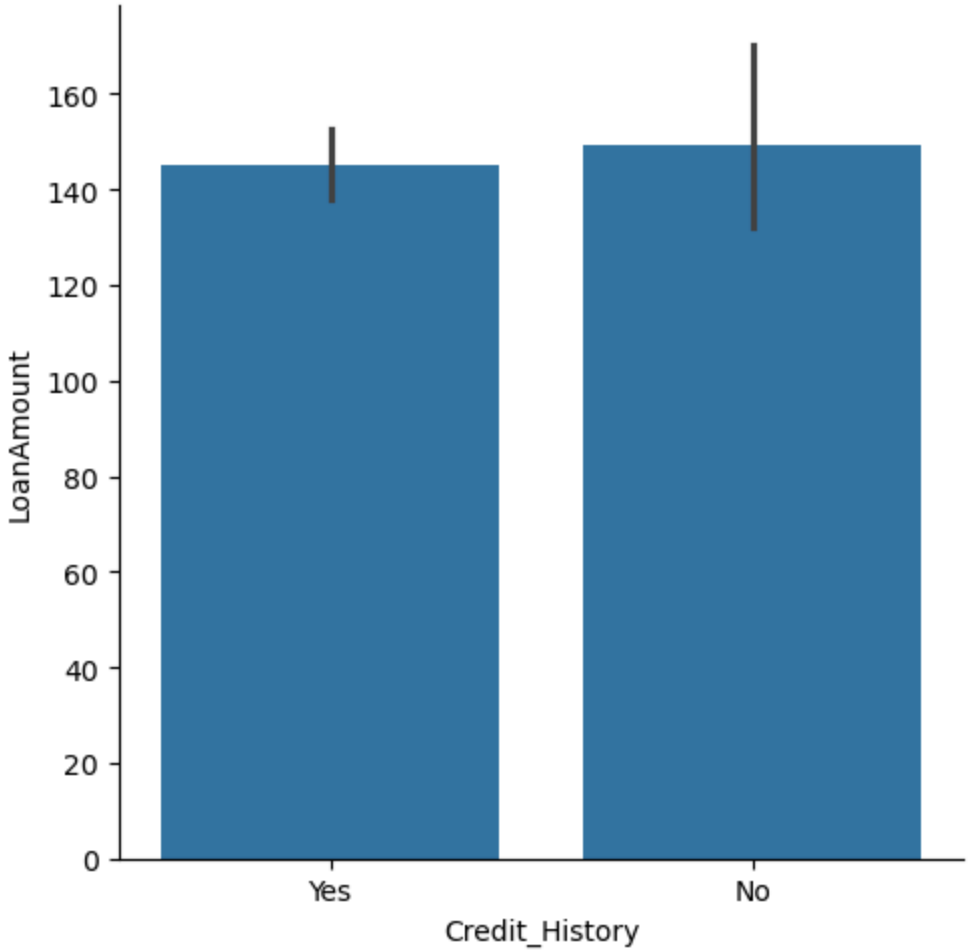
In [197…

```
sns.catplot(x="Self_Employed",y="overall_income",data=df,kind="bar")
plt.show()
```



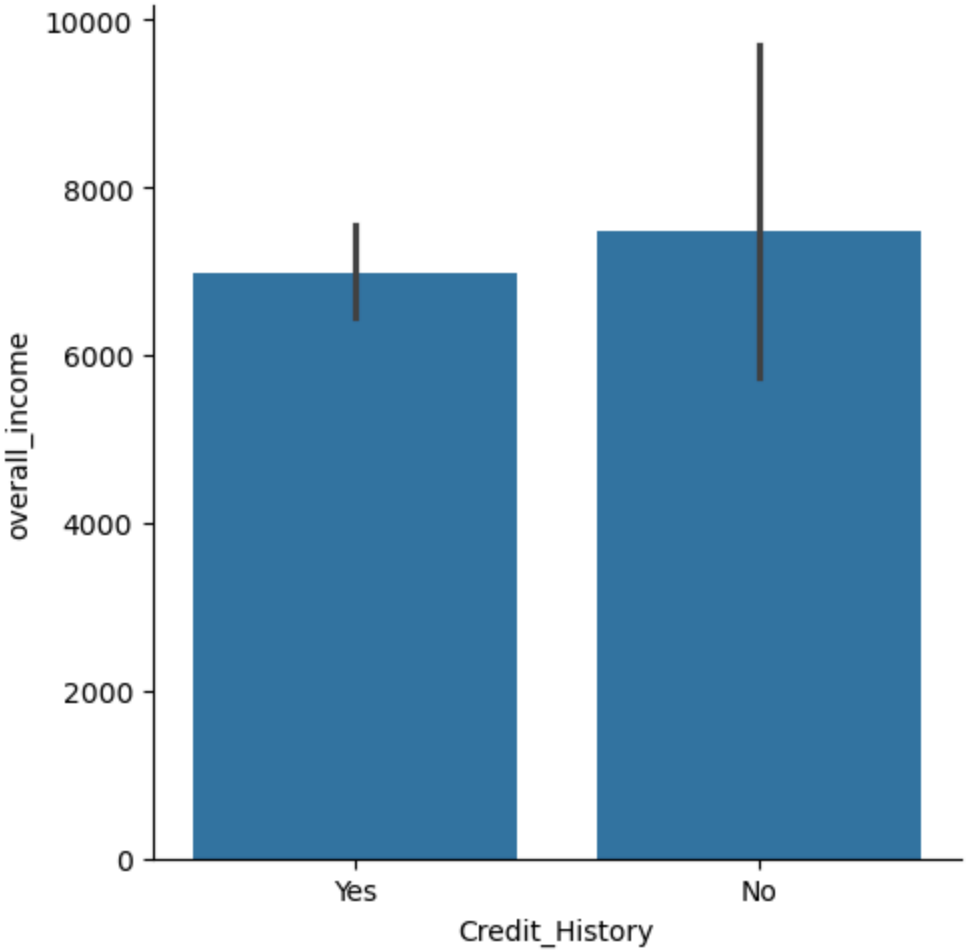
In [198…

```
sns.catplot(x="Credit_History",y="LoanAmount",data=df,kind="bar")
plt.show()
```



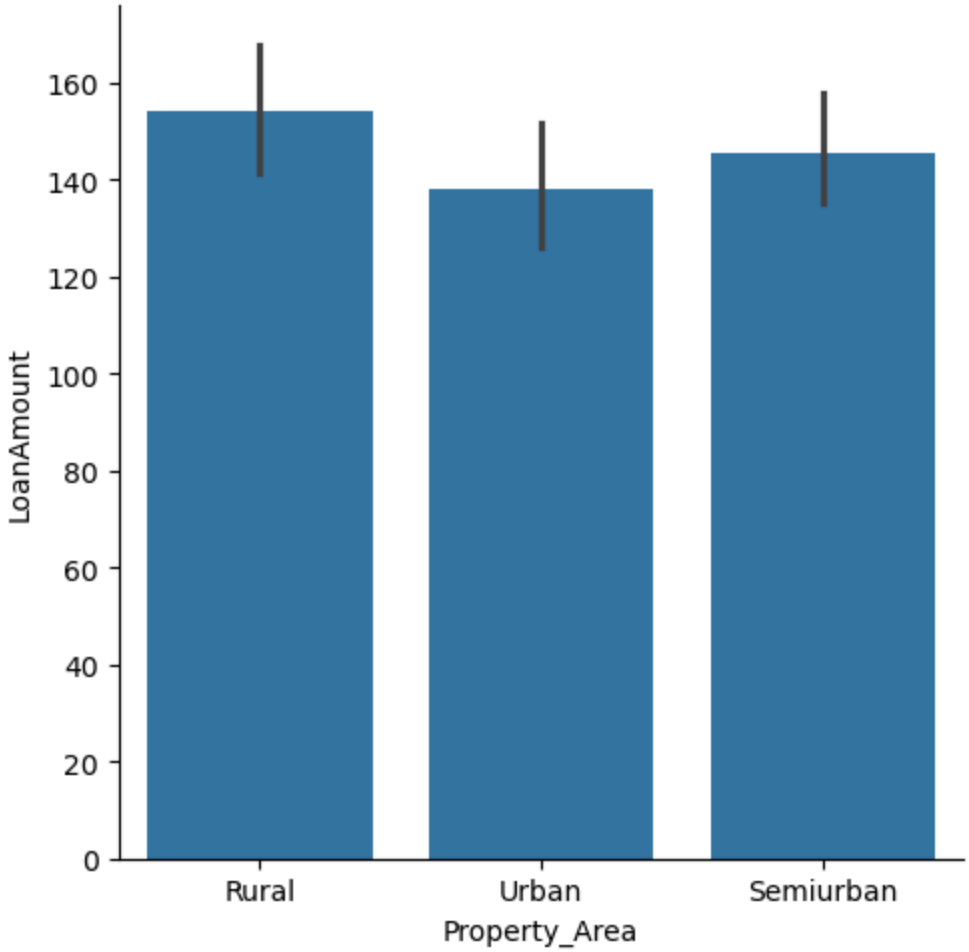
In [200…

```
sns.catplot(x="Credit_History",y="overall_income",data=df,kind="bar")
plt.show()
```



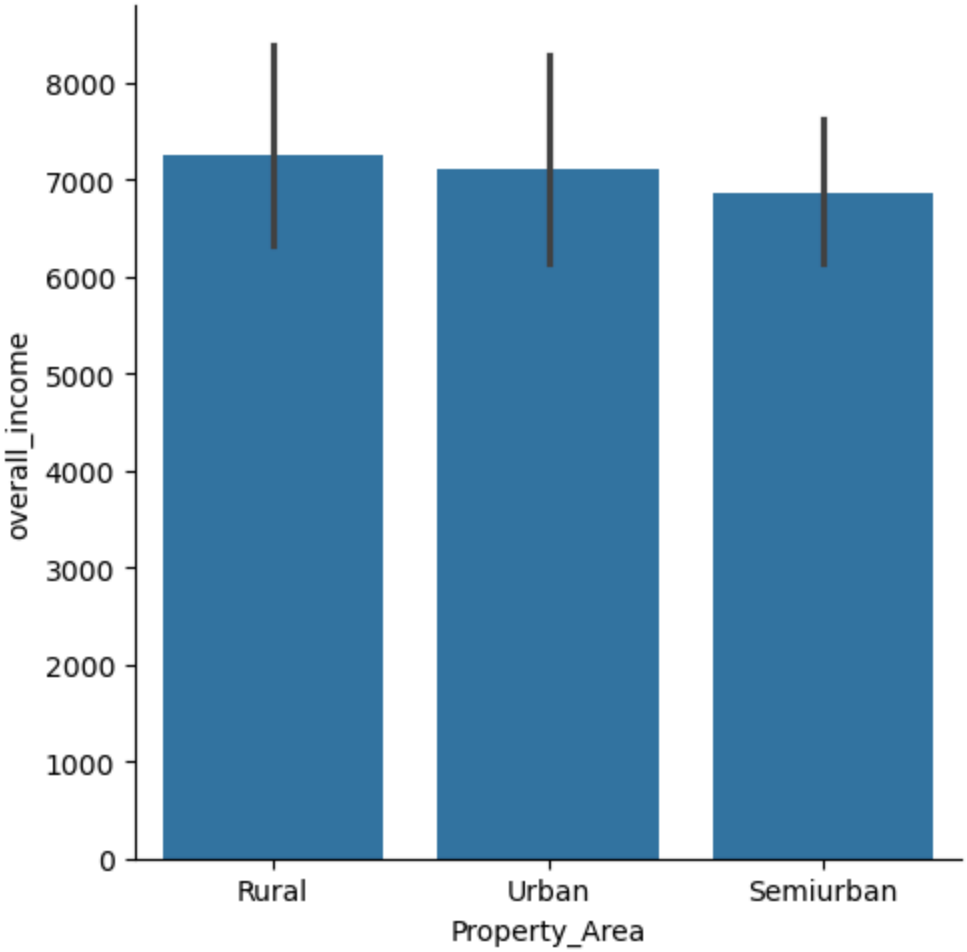
In [201…

```
sns.catplot(x="Property_Area",y="LoanAmount",data=df,kind="bar")
plt.show()
```



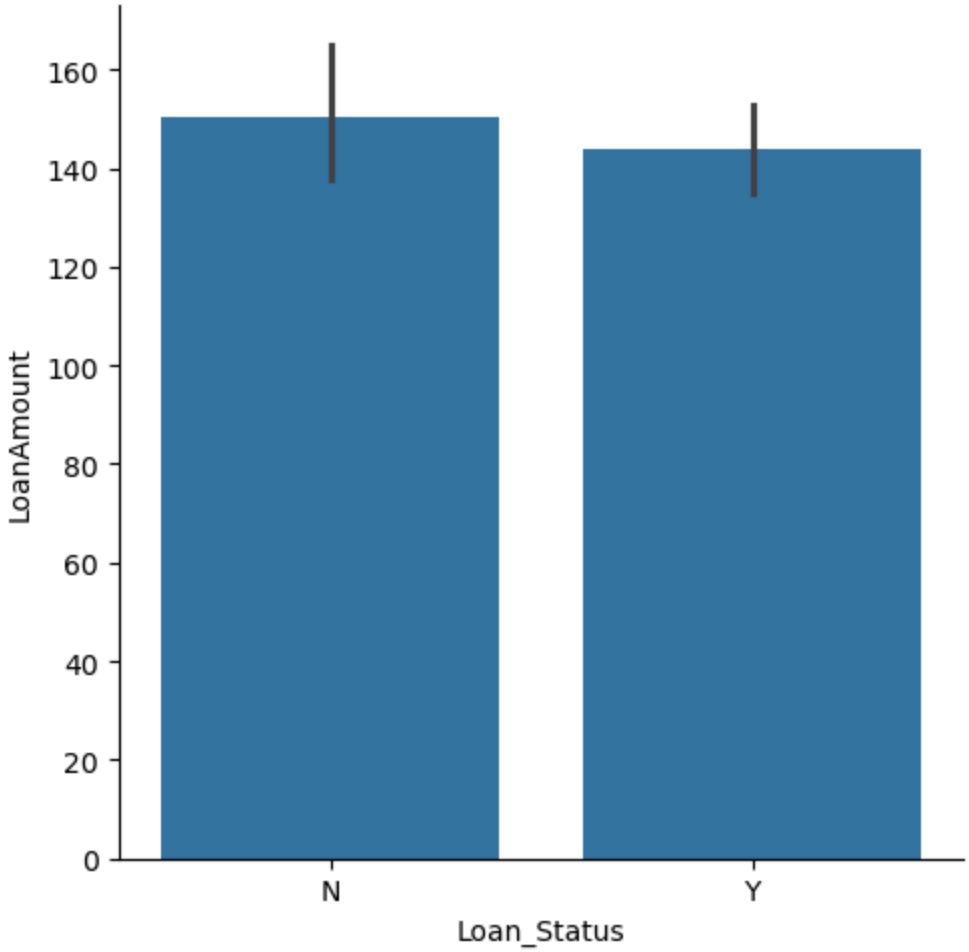
In [203…

```
sns.catplot(x="Property_Area",y="overall_income",data=df,kind="bar")
plt.show()
```



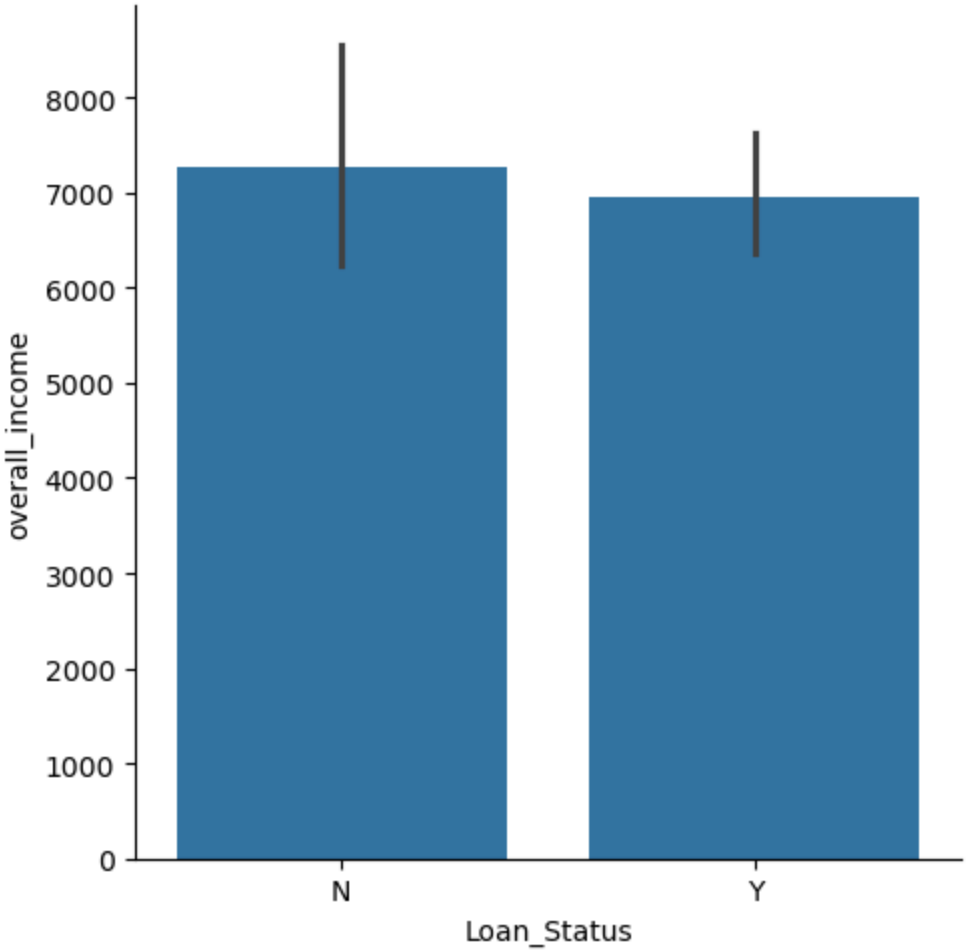
In [205…

```
sns.catplot(x="Loan_Status",y="LoanAmount",data=df,kind="bar")
plt.show()
```



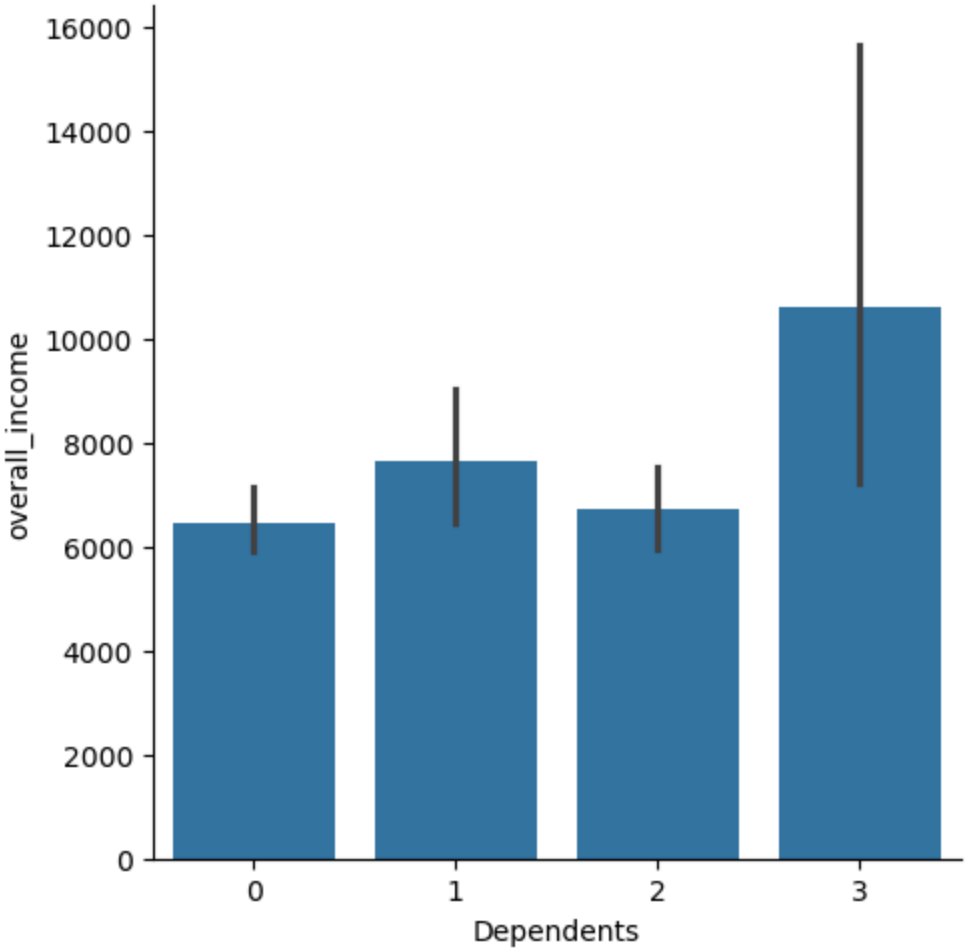
In [206…

```
sns.catplot(x="Loan_Status",y="overall_income",data=df,kind="bar")
plt.show()
```



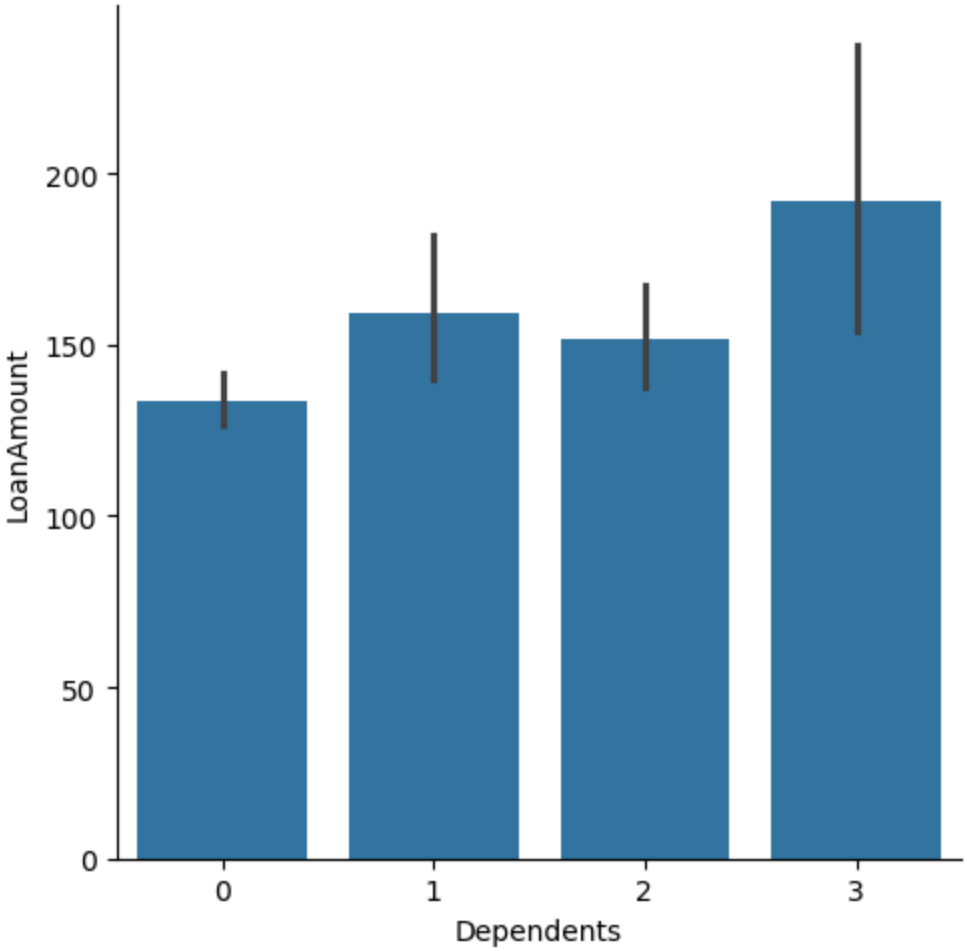
In [208…

```
sns.catplot(x="Dependents",y="overall_income",data=df,kind="bar")
plt.show()
```



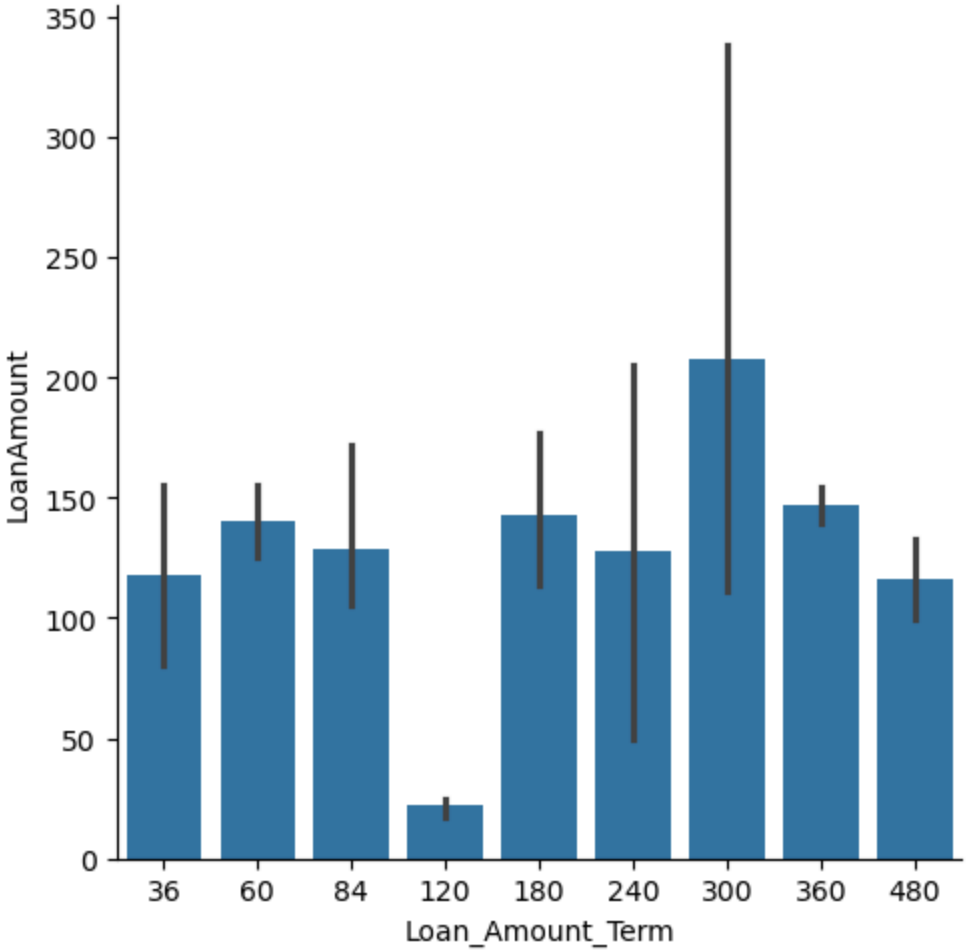
In [209…

```
sns.catplot(x="Dependents",y="LoanAmount",data=df,kind="bar")
plt.show()
```



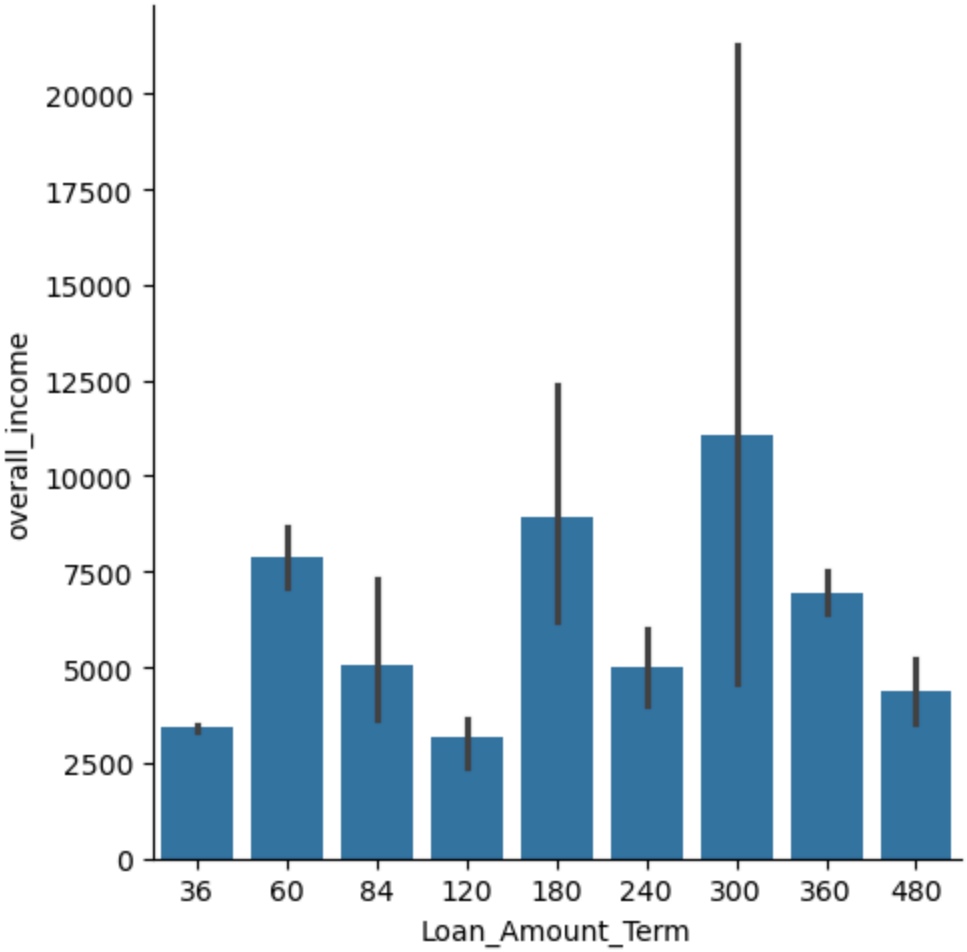
In [210…

```
sns.catplot(x="Loan_Amount_Term",y="LoanAmount",data=df,kind="bar")
plt.show()
```



In [212…

```
sns.catplot(x="Loan_Amount_Term",y="overall_income",data=df,kind="bar")
plt.show()
```

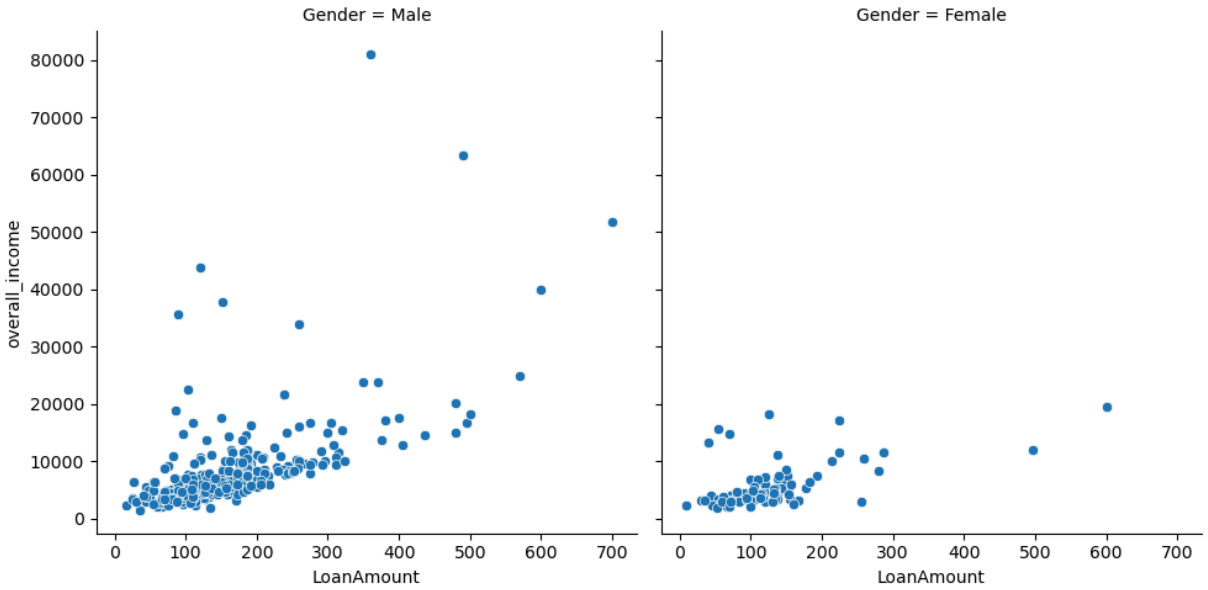


Relative Plot

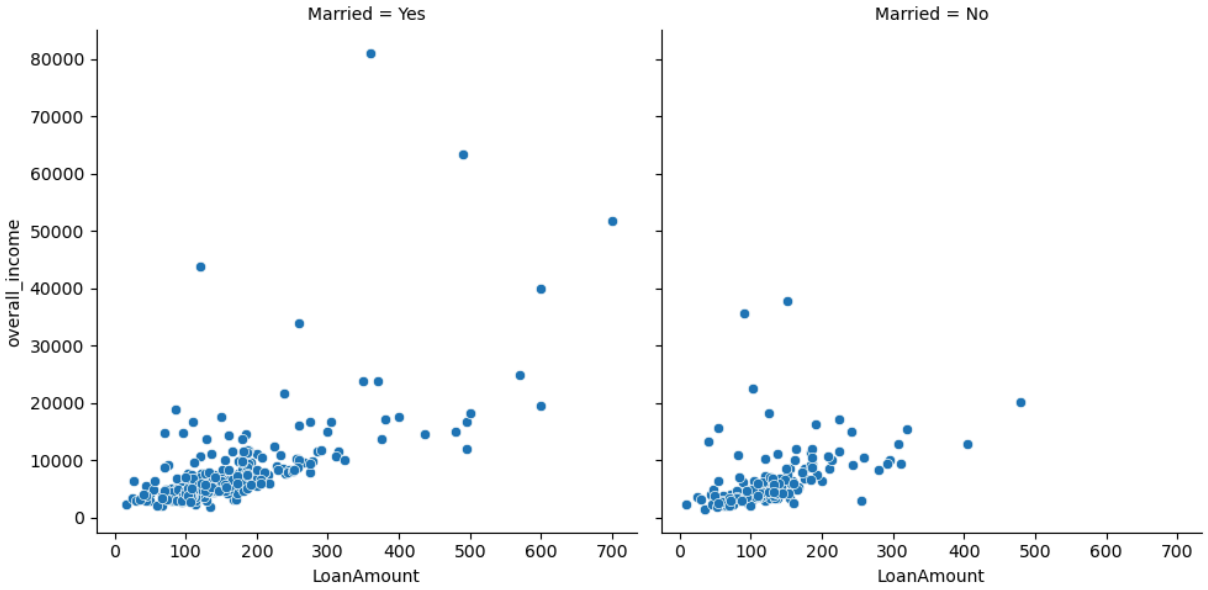
2 continous + 1 discrete

In [216…

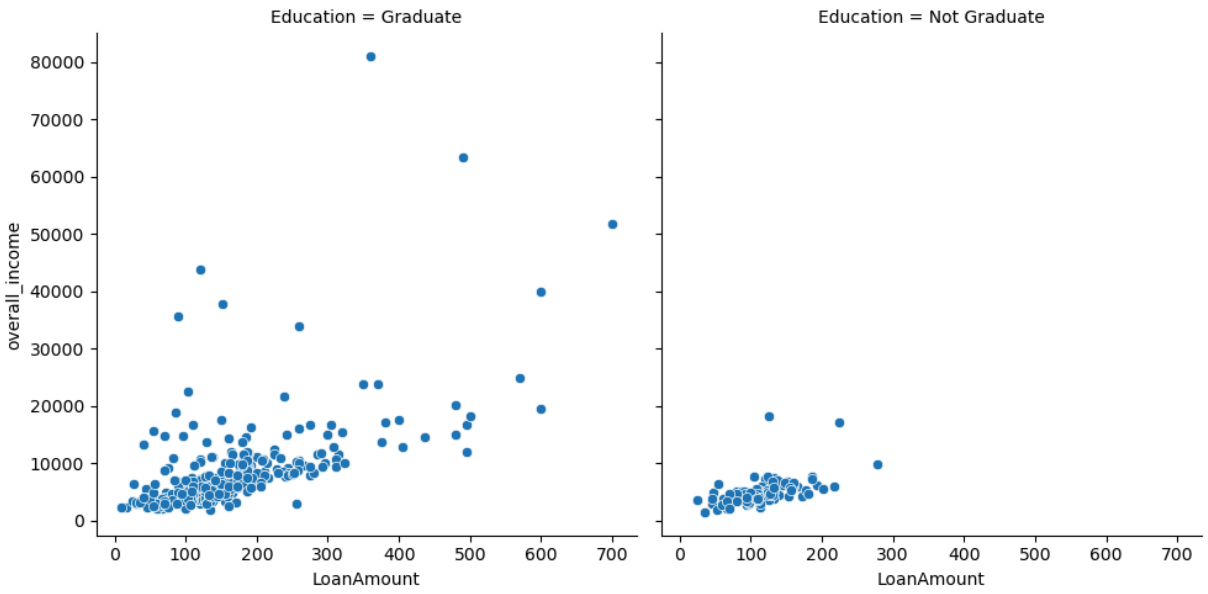
```
sns.relplot(x="LoanAmount",y="overall_income",data = df,col = "Gender")
plt.show()
```



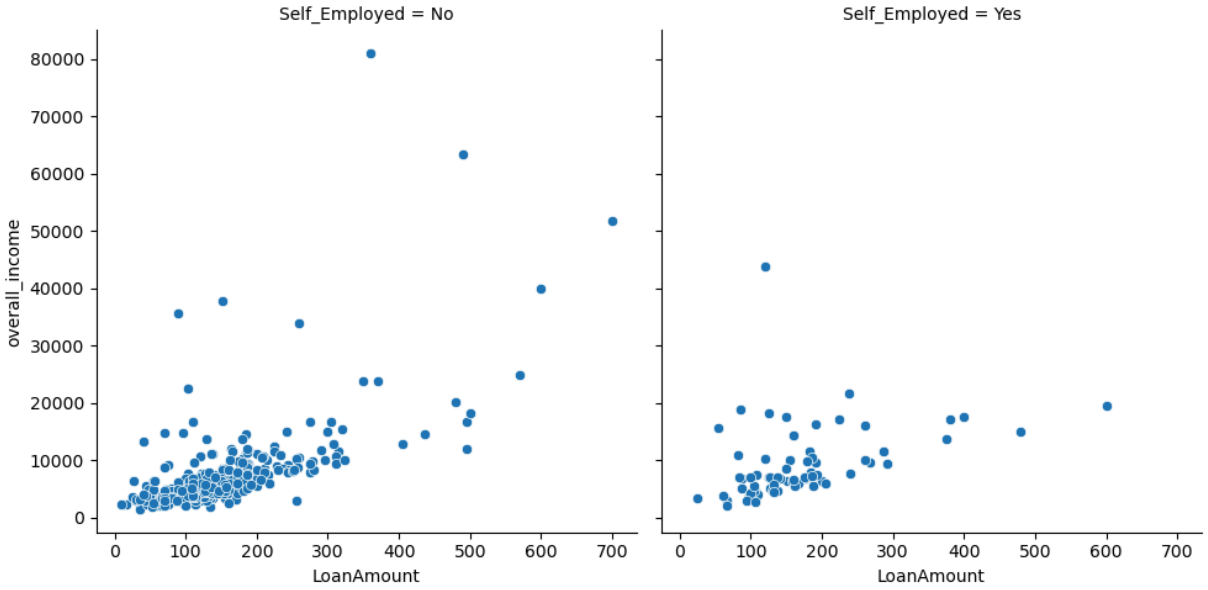
In [218…
```
sns.relplot(x="LoanAmount",y="overall_income",data = df,col = "Married")
plt.show()
```



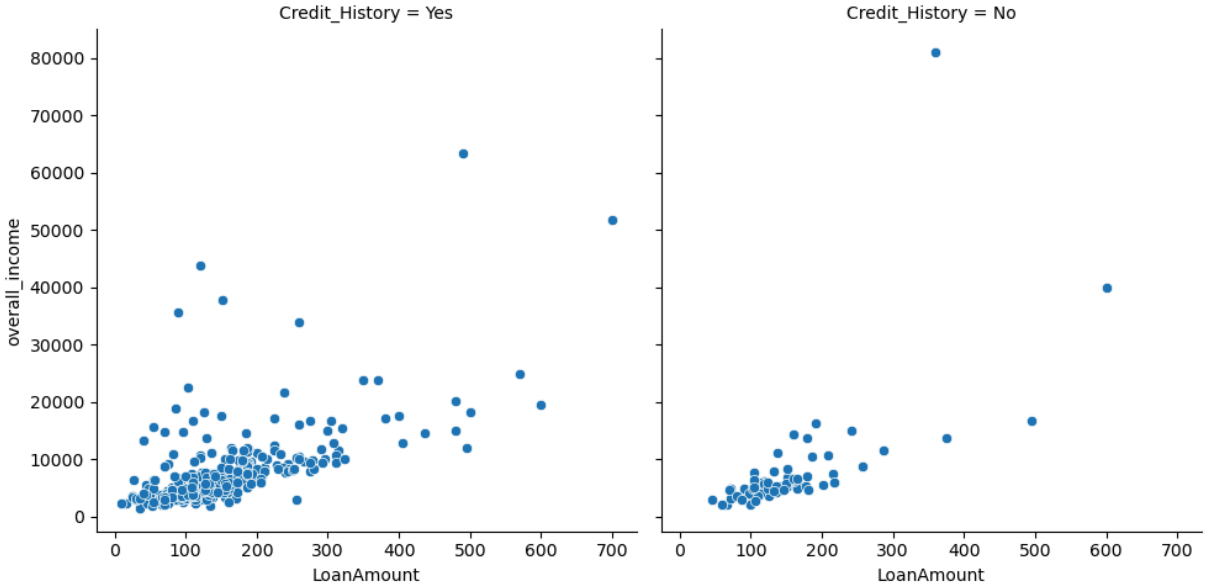
In [220…
```
sns.relplot(x="LoanAmount",y="overall_income",data = df,col = "Education")
plt.show()
```



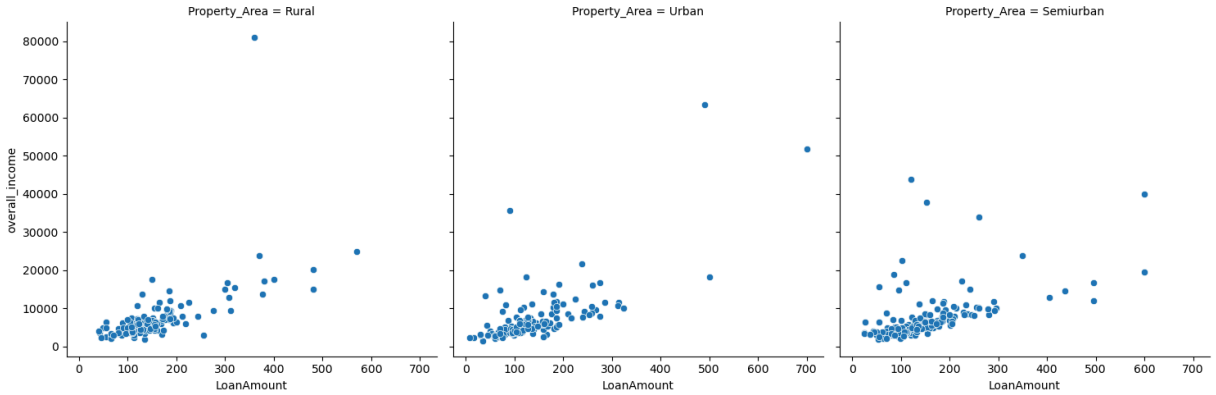
In [222…
```
sns.relplot(x="LoanAmount",y="overall_income",data = df,col = "Self_Employed")
plt.show()
```



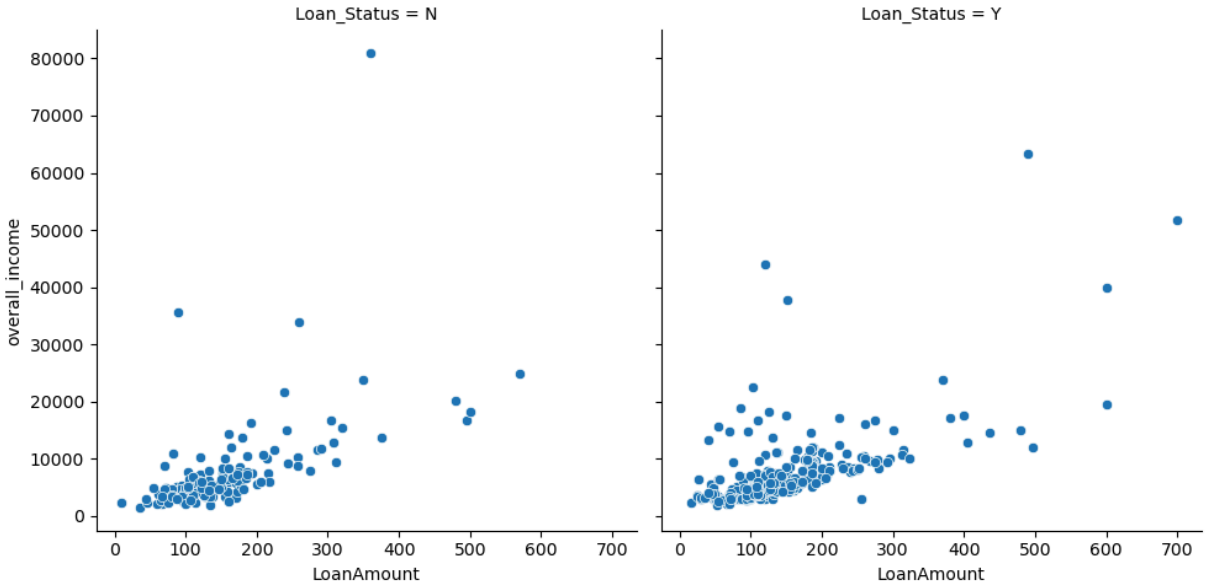
In [223…
```
sns.relplot(x="LoanAmount",y="overall_income",data = df,col = "Credit_History")
plt.show()
```



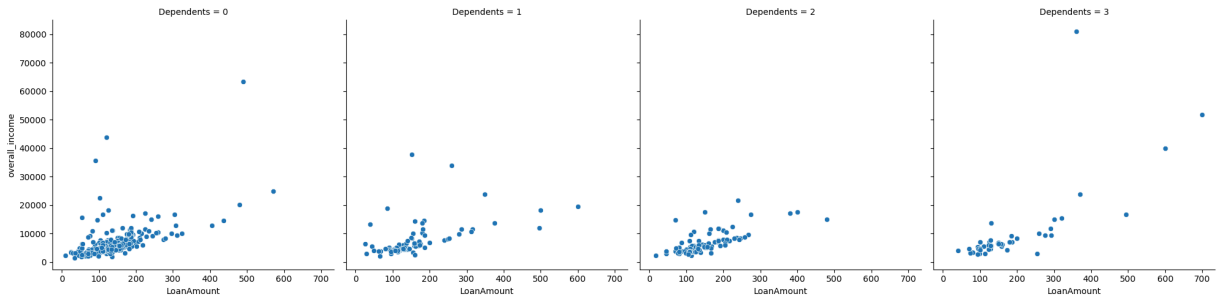
In [225…
```
sns.relplot(x="LoanAmount",y="overall_income",data = df,col = "Property_Area")
plt.show()
```



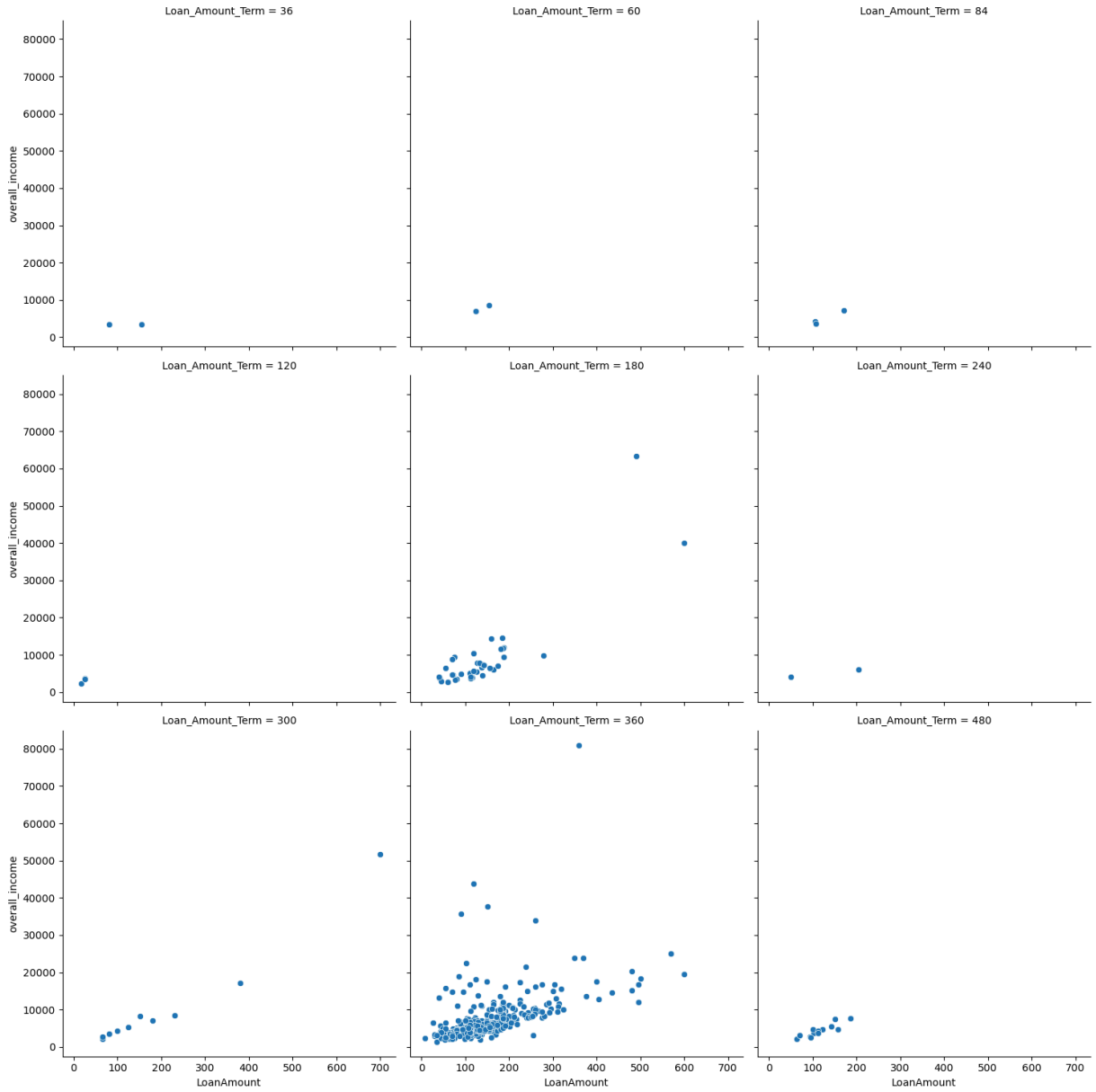
In [227…
```
sns.relplot(x="LoanAmount",y="overall_income",data = df,col = "Loan_Status")
plt.show()
```



In [228…
```
sns.relplot(x="LoanAmount",y="overall_income",data = df,col = "Dependents")
plt.show()
```



In [230…
```
sns.relplot(x="LoanAmount",y="overall_income",data = df,col = "Loan_Amount_Term",co
plt.show()
```



Heat Map

only continous

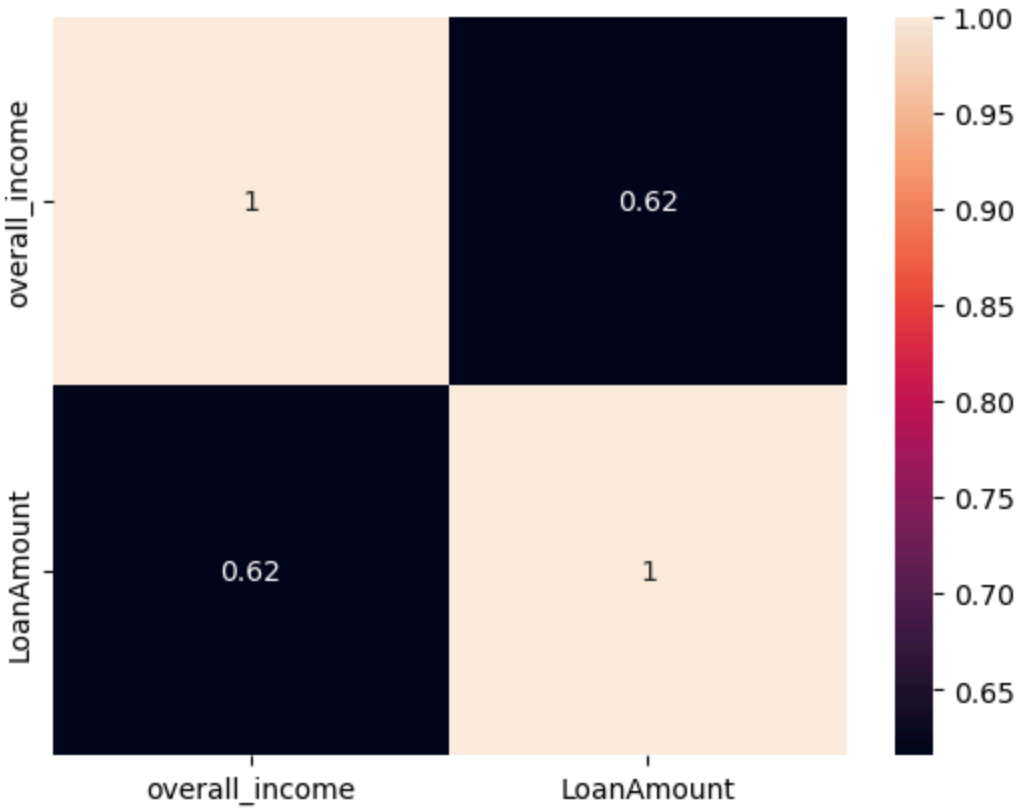
continous

In [235…
```
hm = df[["overall_income","LoanAmount"]].corr()
hm
```

Out[235…

	overall_income	LoanAmount
overall_income	1.000000	0.615632
LoanAmount	0.615632	1.000000

In [237…
```
sns.heatmap(hm,annot=True)
plt.show()
```



Count

In [240…
```
hm1 = df[["Dependents","Loan_Amount_Term"]].corr()
hm1
```

Out[240…

	Dependents	Loan_Amount_Term
Dependents	1.000000	-0.090968
Loan_Amount_Term	-0.090968	1.000000

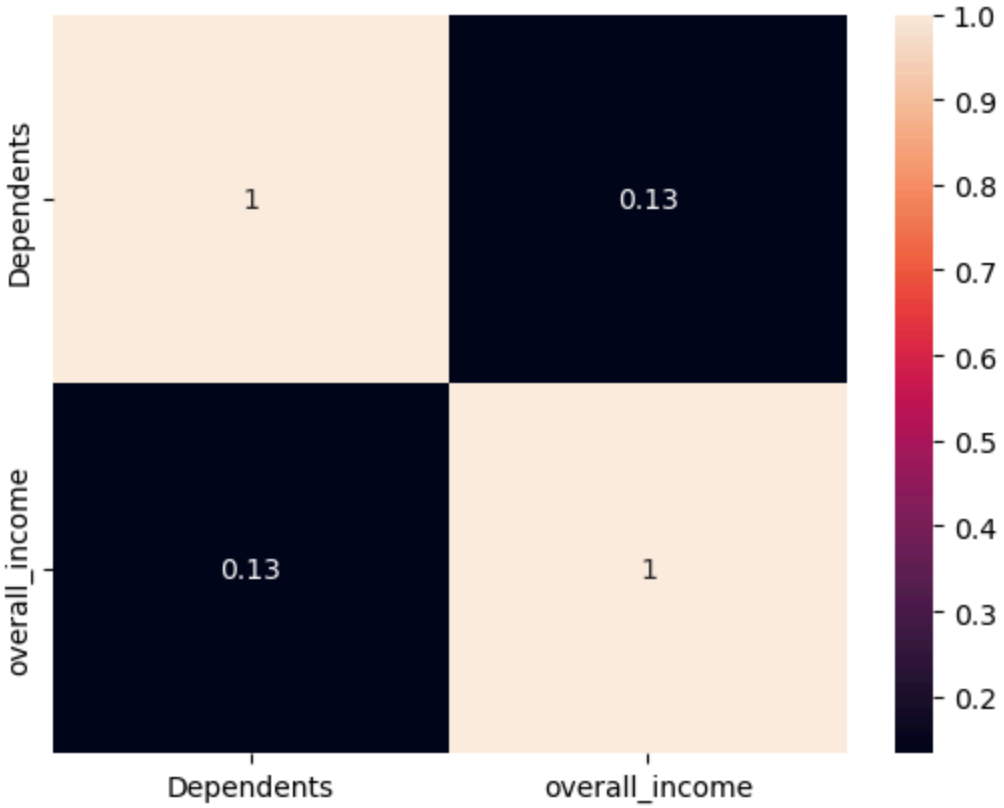
In [242…
```
sns.heatmap(hm1,annot=True)
plt.show()
```

In [243…
```
hm2 = df[["Dependents","overall_income"]].corr()
hm2
```

Out[243…

	Dependents	overall_income
Dependents	1.000000	0.133477
overall_income	0.133477	1.000000

In [245…
```
sns.heatmap(hm2,annot=True)
plt.show()
```

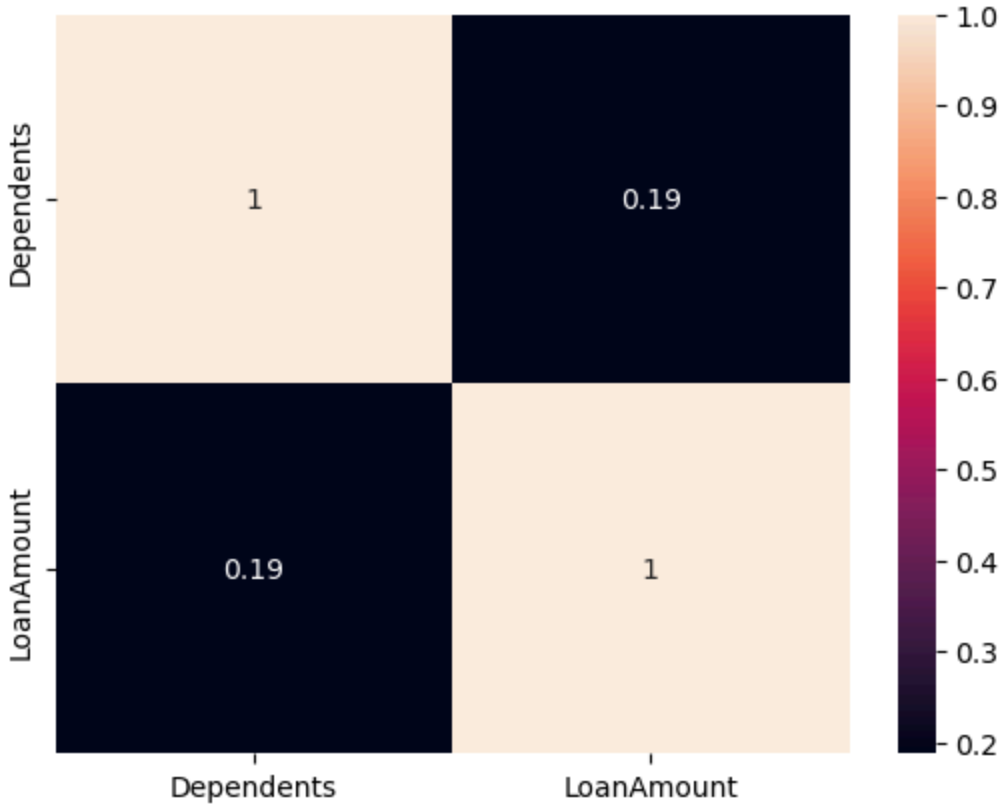


In [246…
```
hm3 = df[["Dependents","LoanAmount"]].corr()
hm3
```

Out[246…

	Dependents	LoanAmount
Dependents	1.000000	0.187572
LoanAmount	0.187572	1.000000

In [249…
```
sns.heatmap(hm3,annot=True)
plt.show()
```

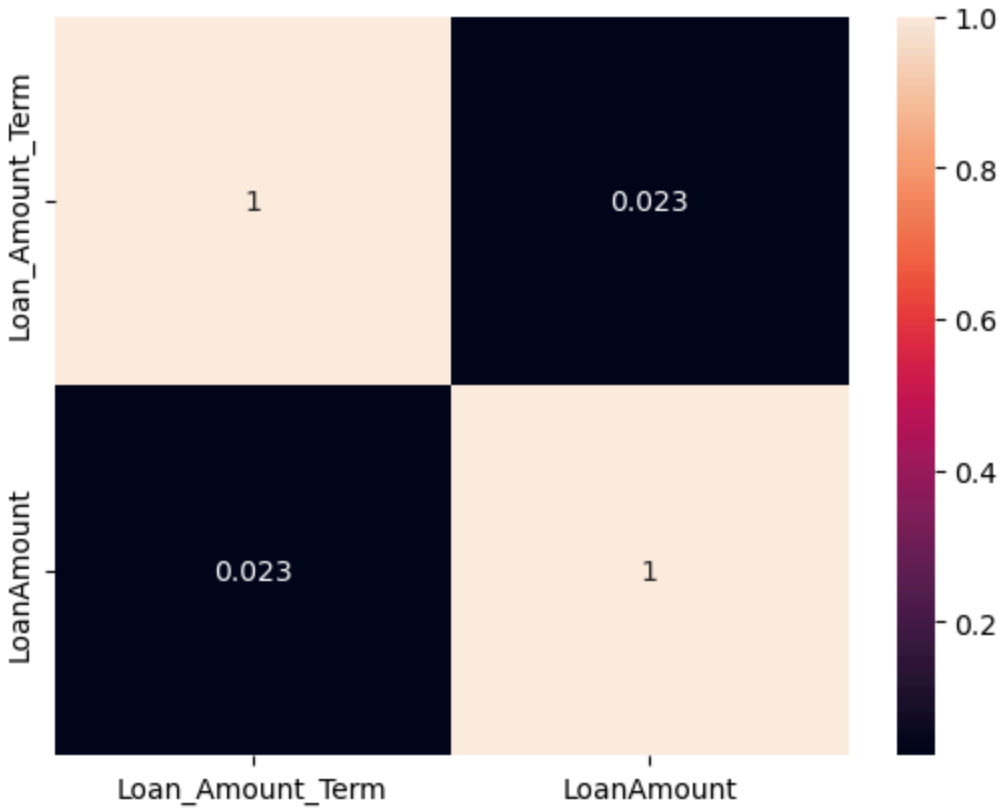


In [251…
```
hm4 = df[["Loan_Amount_Term","LoanAmount"]].corr()
hm4
```

Out[251…

	Loan_Amount_Term	LoanAmount
Loan_Amount_Term	1.000000	0.023239
LoanAmount	0.023239	1.000000

In [252…
```
sns.heatmap(hm4,annot=True)
plt.show()
```



In [254...
```
hm5 = df[["Loan_Amount_Term","overall_income"]].corr()
hm5
```

Out[254...

	Loan_Amount_Term	overall_income
Loan_Amount_Term	1.000000	-0.061205
overall_income	-0.061205	1.000000

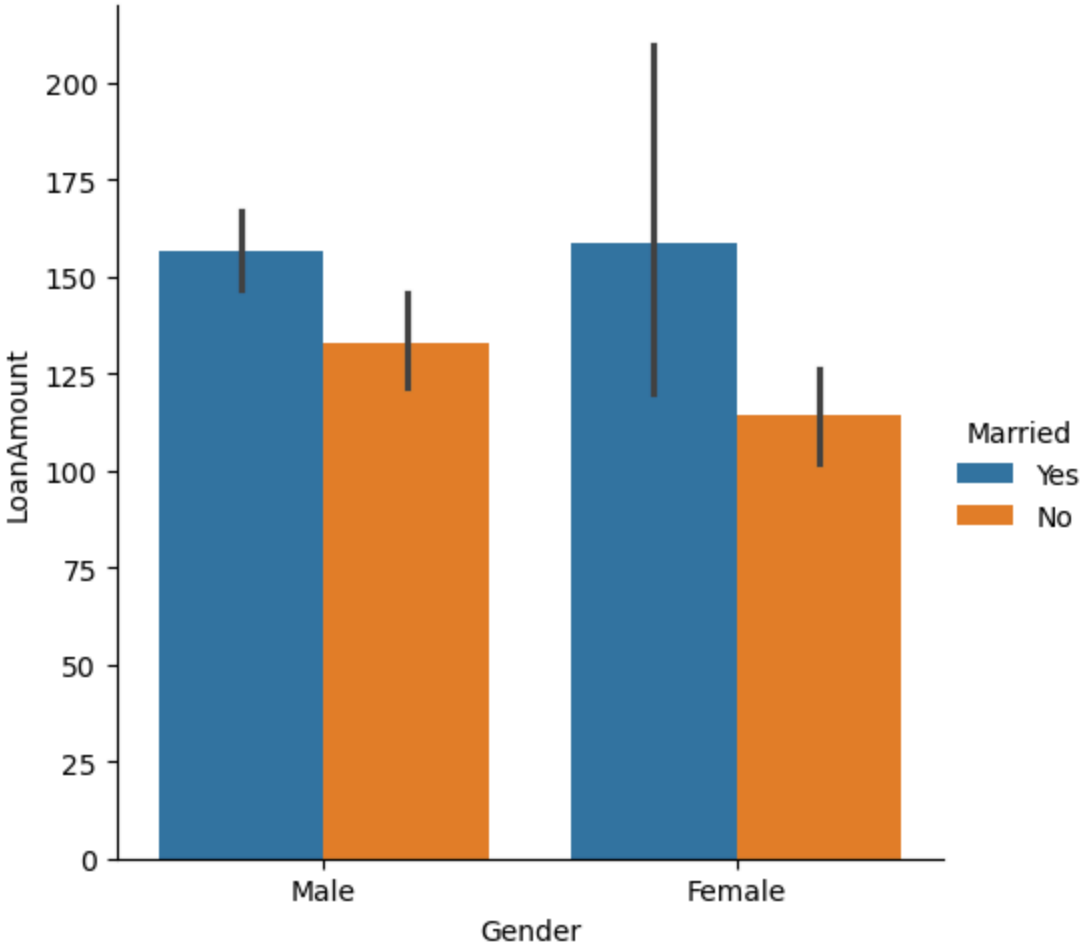
In [255...
```
sns.heatmap(hm5,annot=True)
plt.show()
```

Unstacked Bar plot

Vertical bar plot = bar(x:discrete,y:continous)

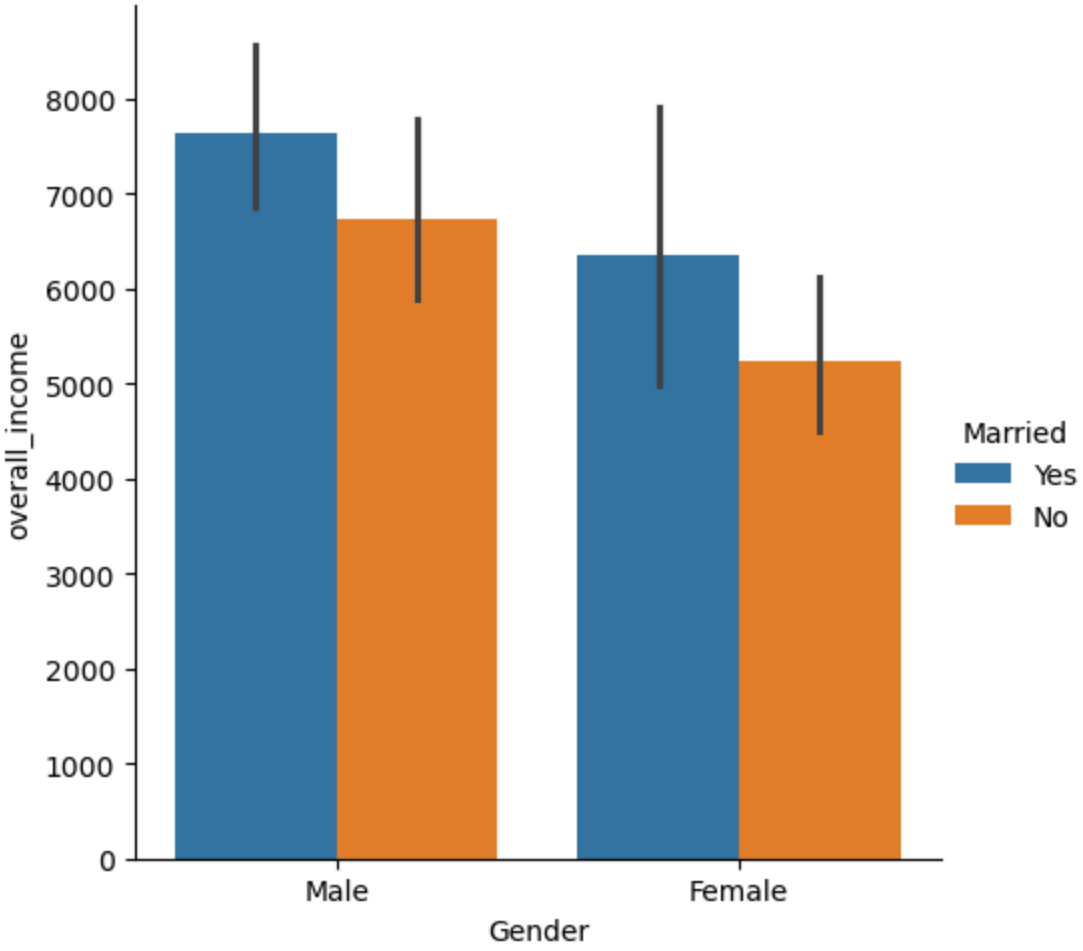
- No. of variable is >2 (Combination are multiple)
- 2 discrete + 1 continous
- 2 continous + 1 discrete

In [258…
```
sns.catplot(x="Gender",y="LoanAmount",data=df,kind="bar",hue="Married")
plt.show()
```



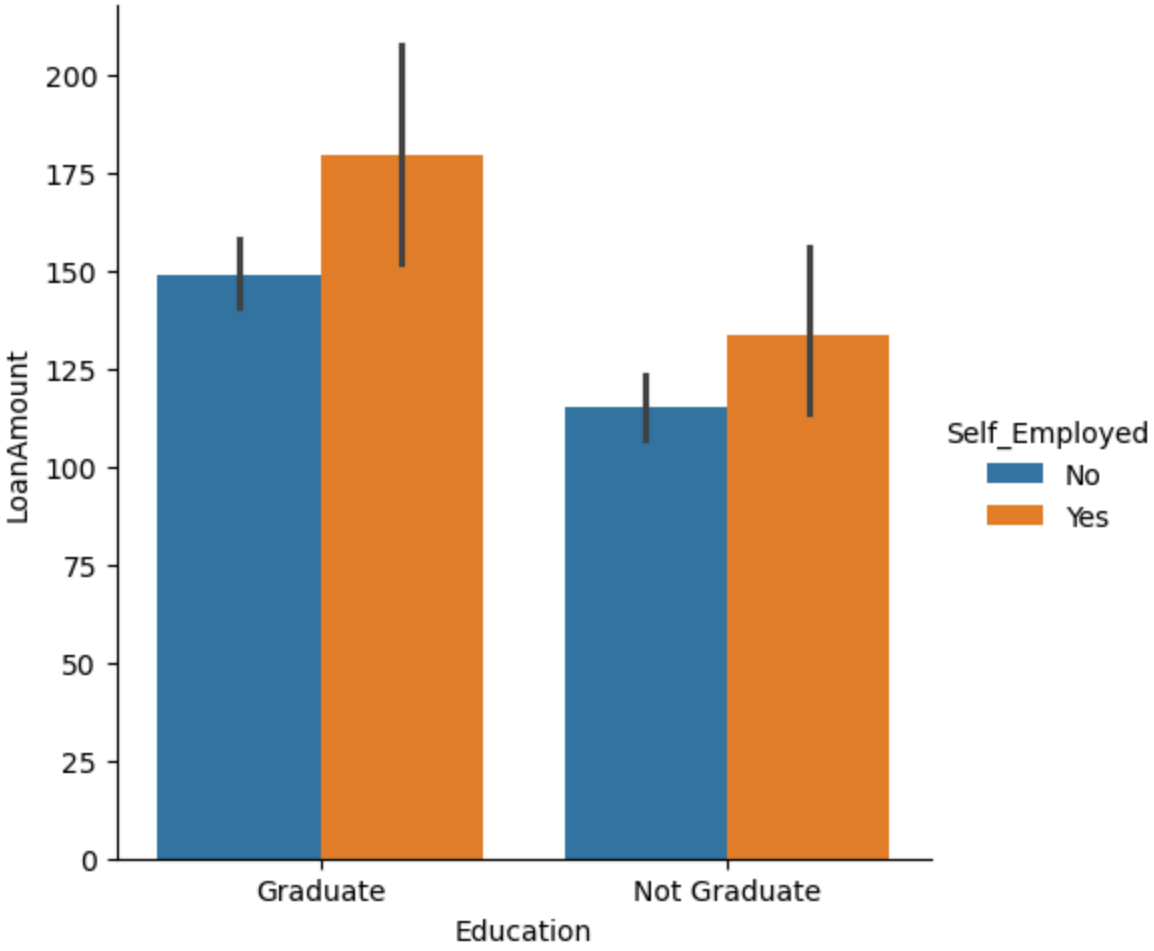
In [259…

```
sns.catplot(x="Gender",y="overall_income",data=df,kind="bar",hue="Married")
plt.show()
```



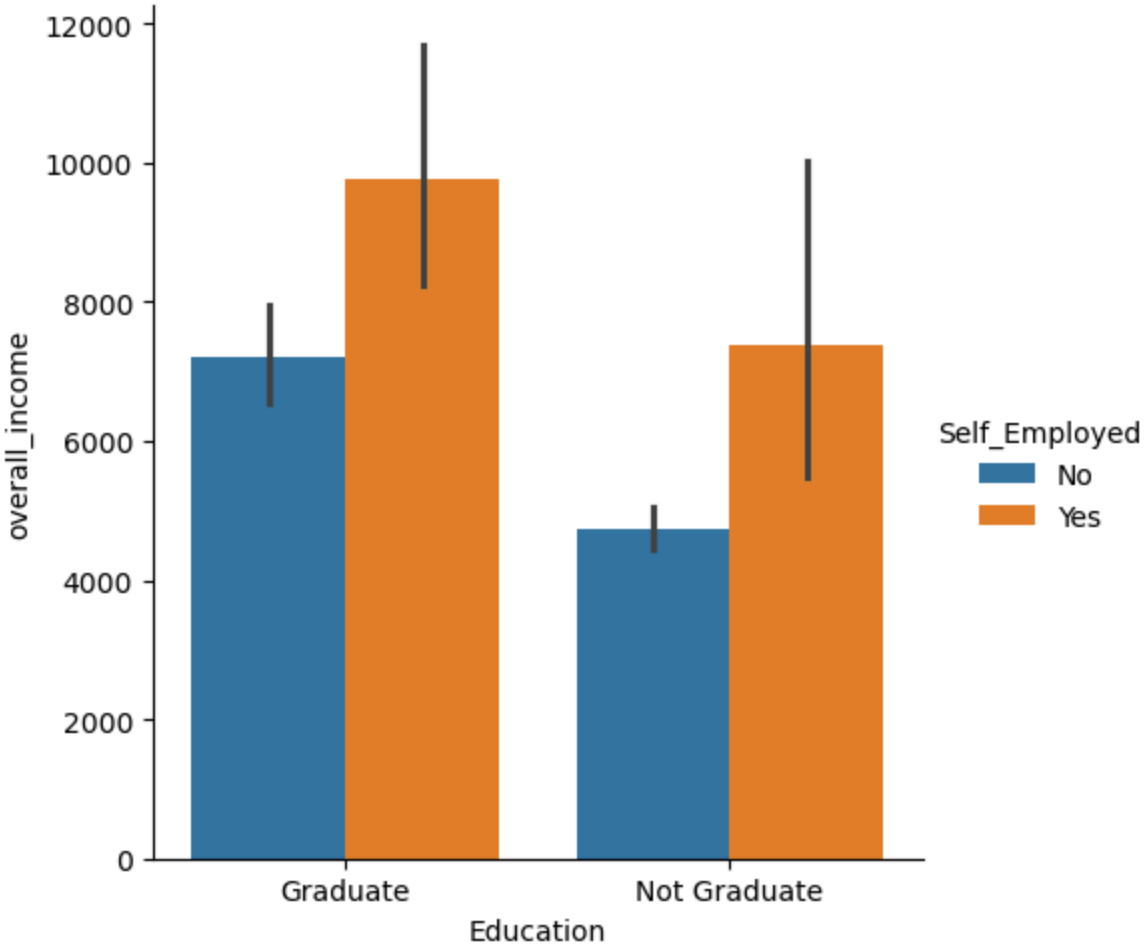
In [260…

```
sns.catplot(x="Education",y="LoanAmount",data=df,kind="bar",hue="Self_Employed")
plt.show()
```



In [261…

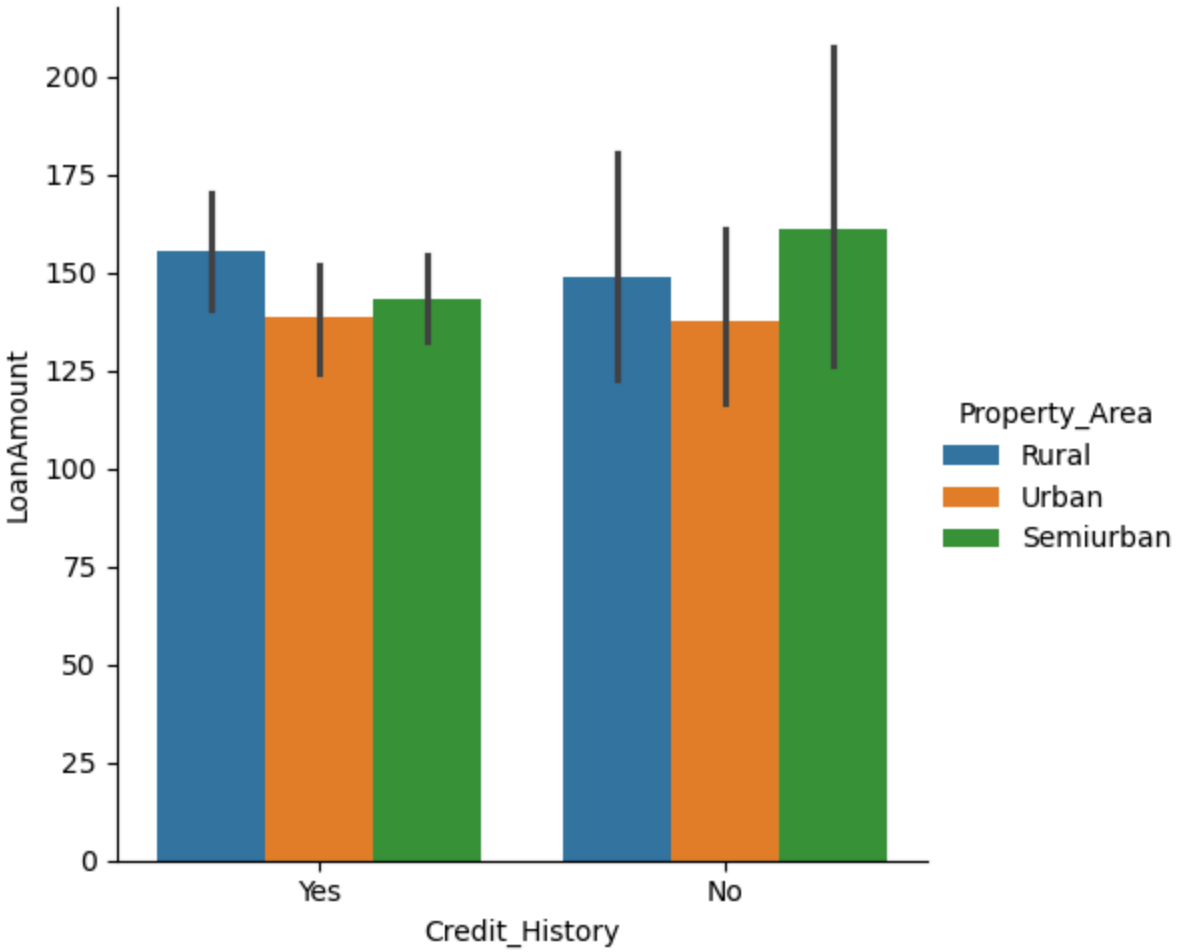
```
sns.catplot(x="Education",y="overall_income",data=df,kind="bar",hue="Self_Employed"
plt.show()
```



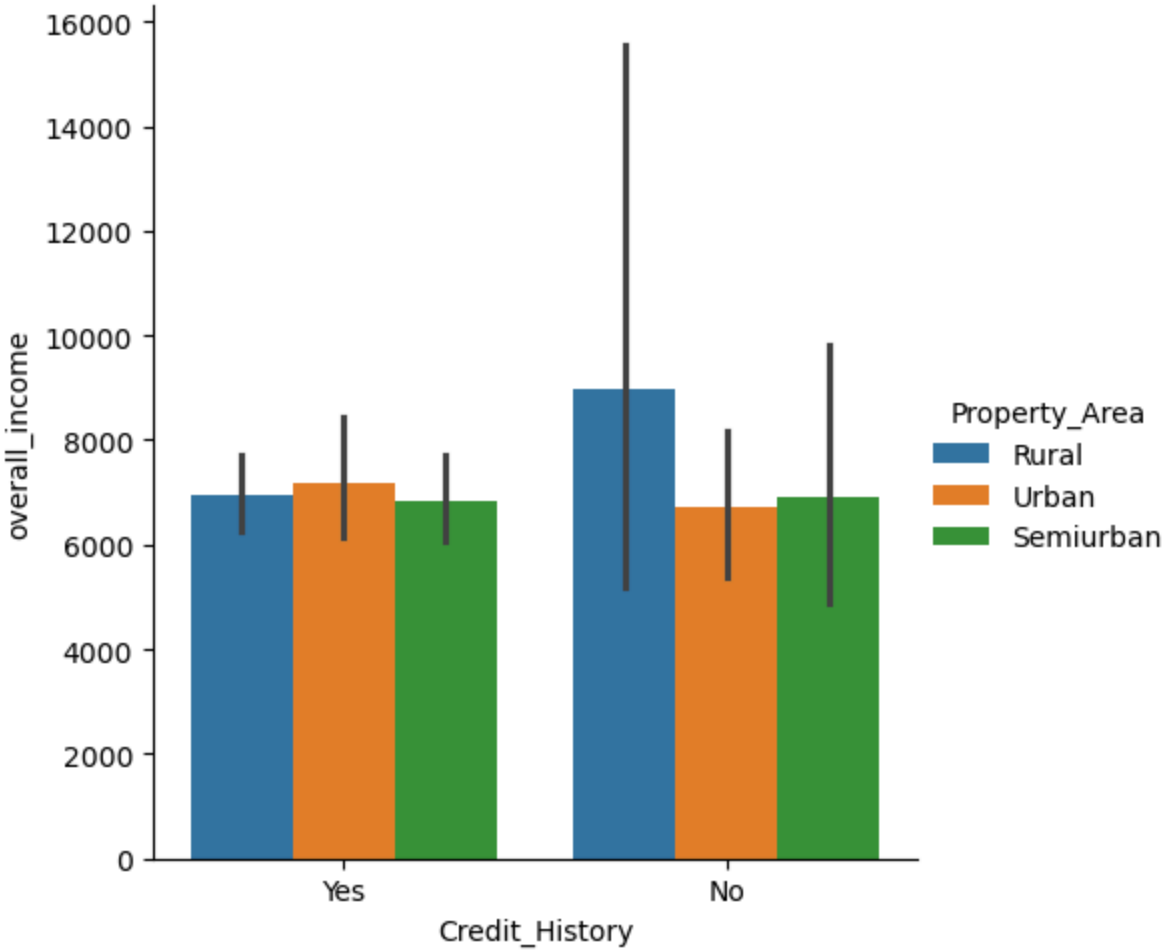
In [263…
```
df1 = df[df["Credit_History"]== "Yes"]
```

In [265…
```
del df1
```

In [268…
```
sns.catplot(x="Credit_History",y="LoanAmount",data=df,kind="bar",hue="Property_Area
plt.show()
```

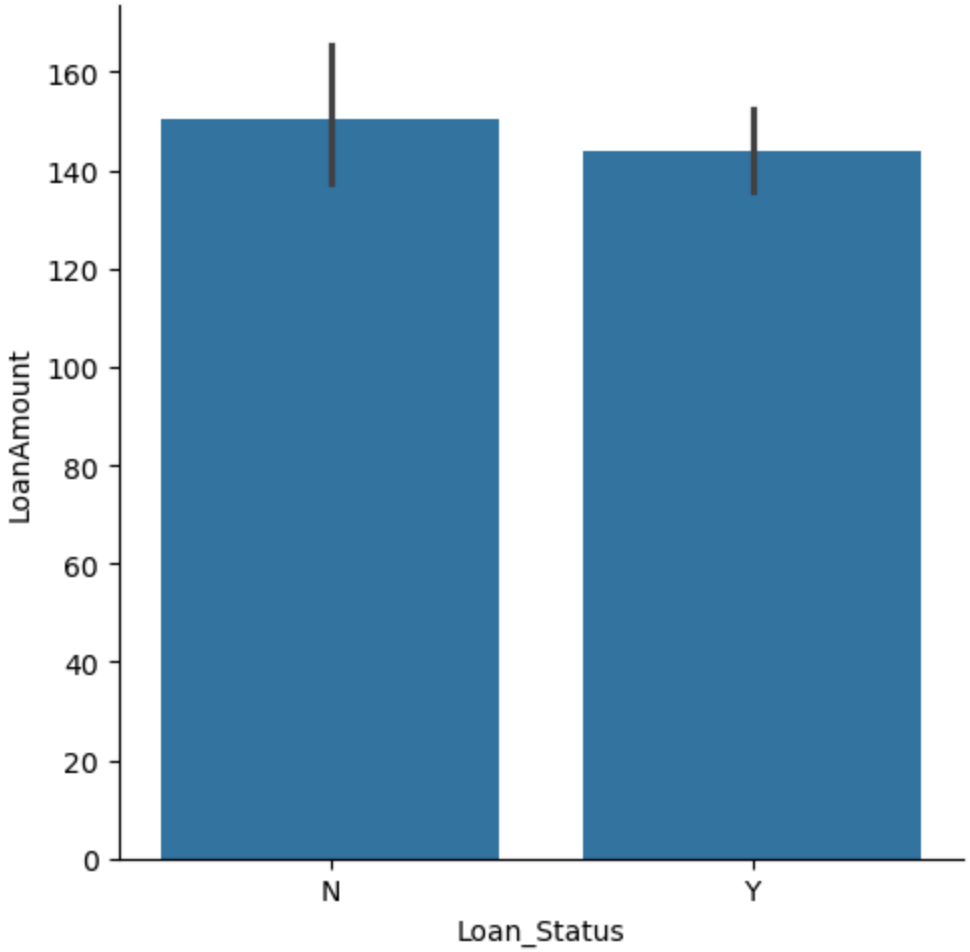


In [271…
```
sns.catplot(x="Credit_History",y="overall_income",data=df,kind="bar",hue="Property_
plt.show()
```

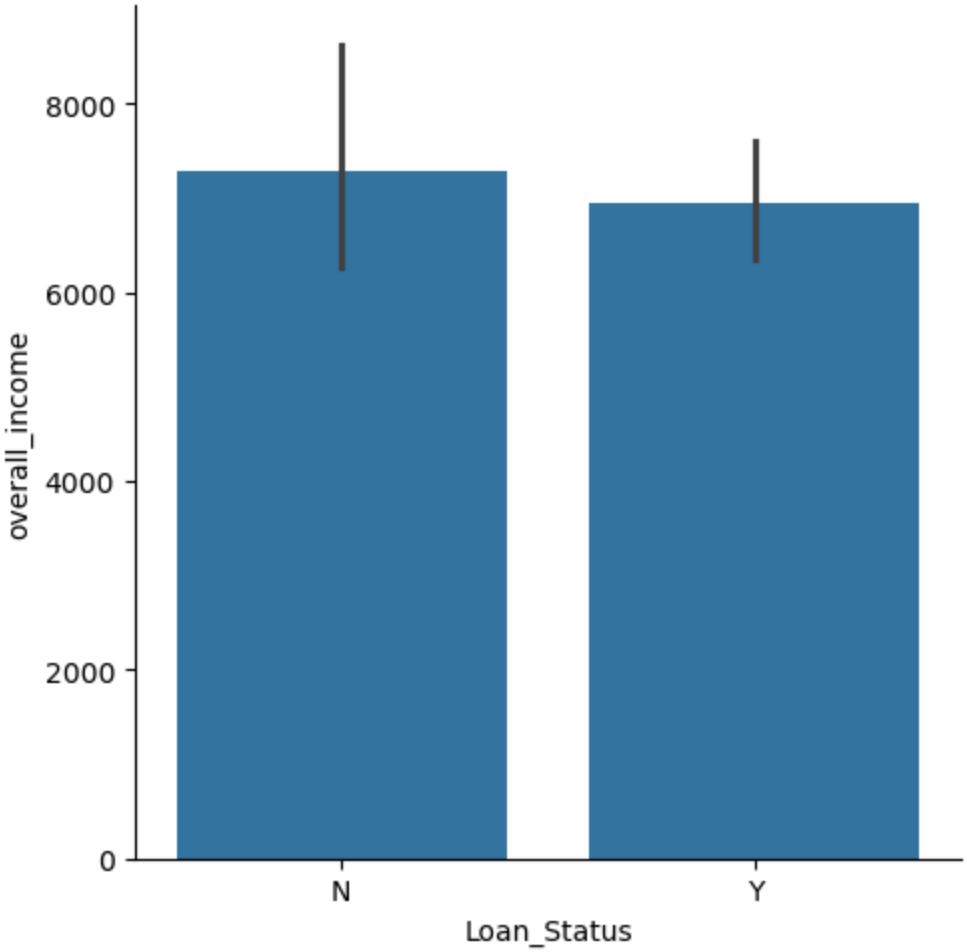


In [274…

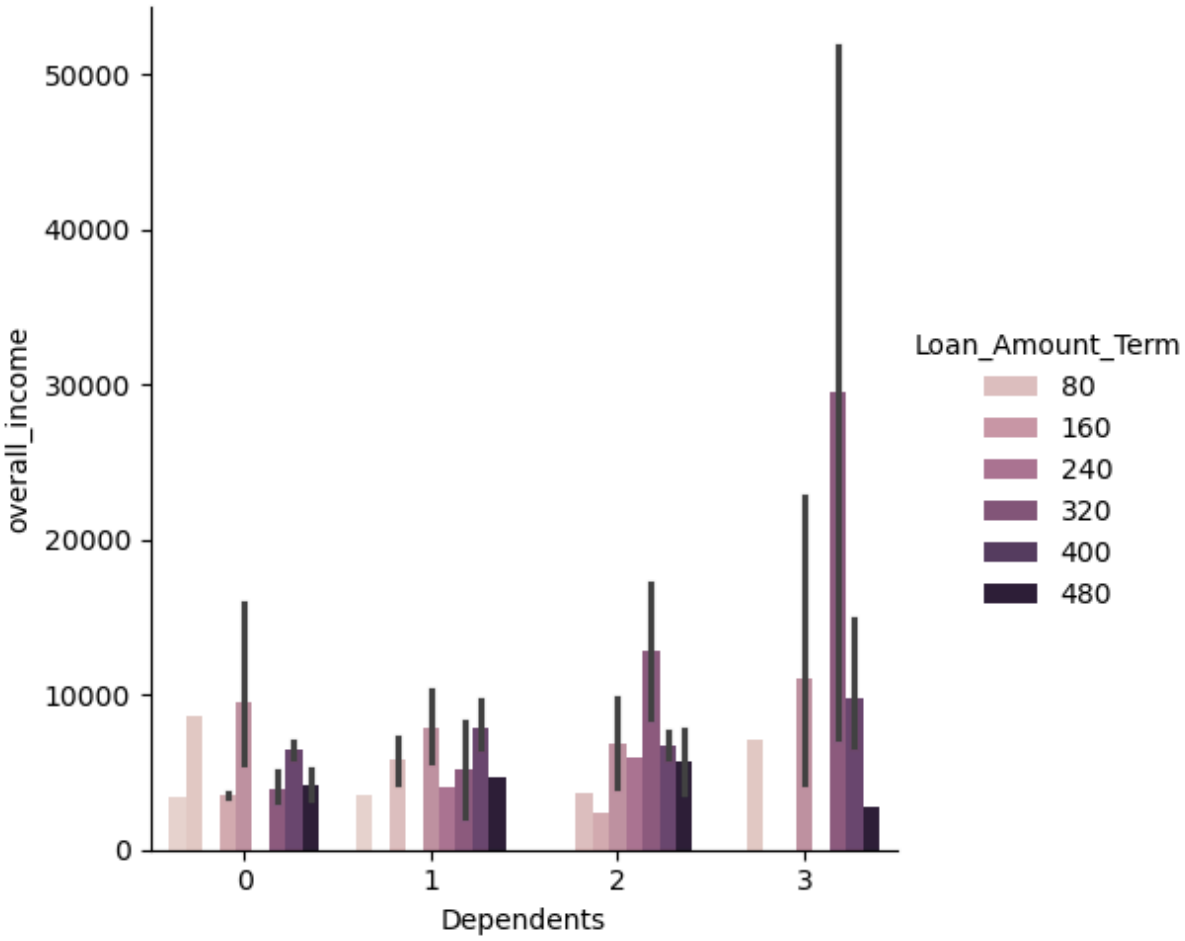
```
sns.catplot(x="Loan_Status",y="LoanAmount",data=df,kind="bar")
plt.show()
```



In [277…
```
sns.catplot(x="Loan_Status",y="overall_income",data=df,kind="bar")
plt.show()
```

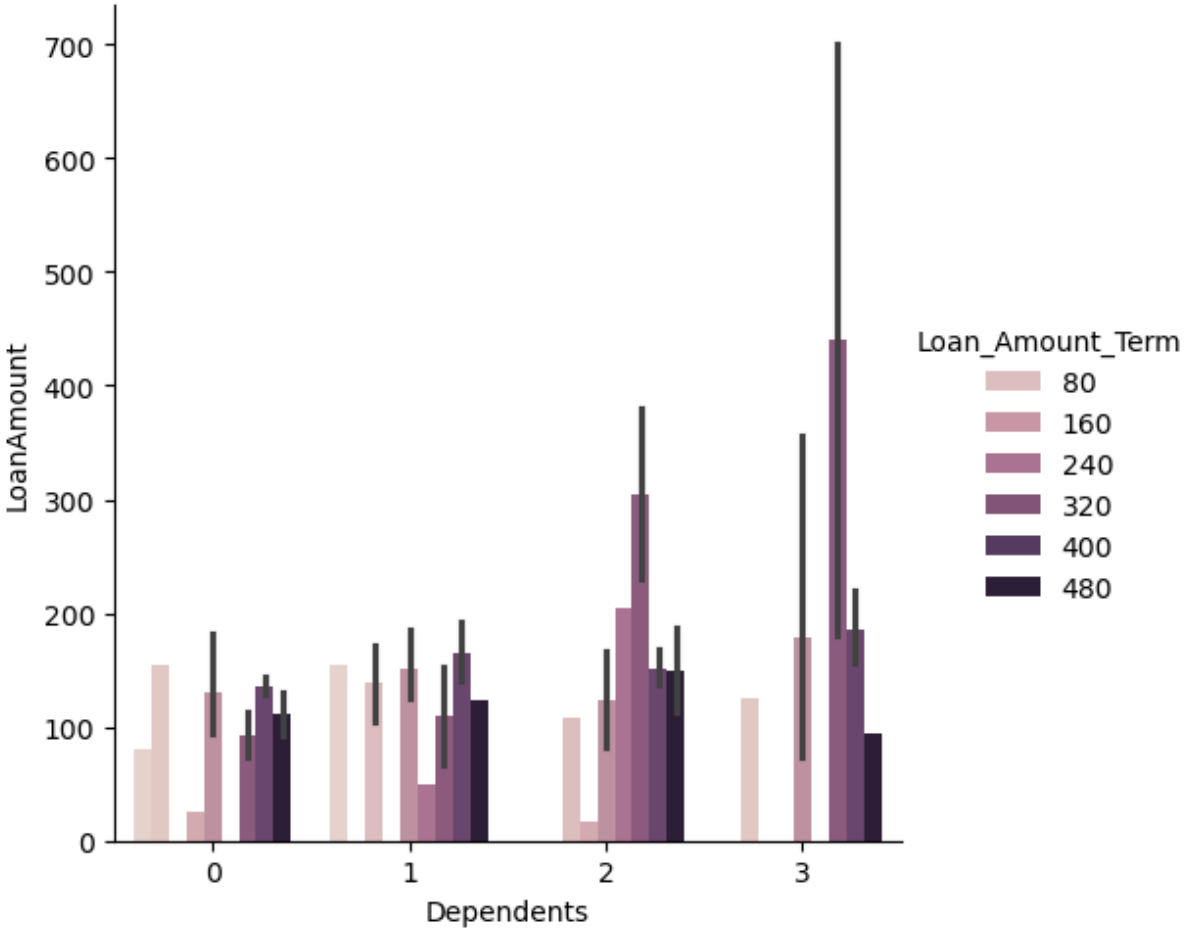


In [280…
```
sns.catplot(x="Dependents",y="overall_income",data=df,kind="bar",hue="Loan_Amount_T
plt.show()
```

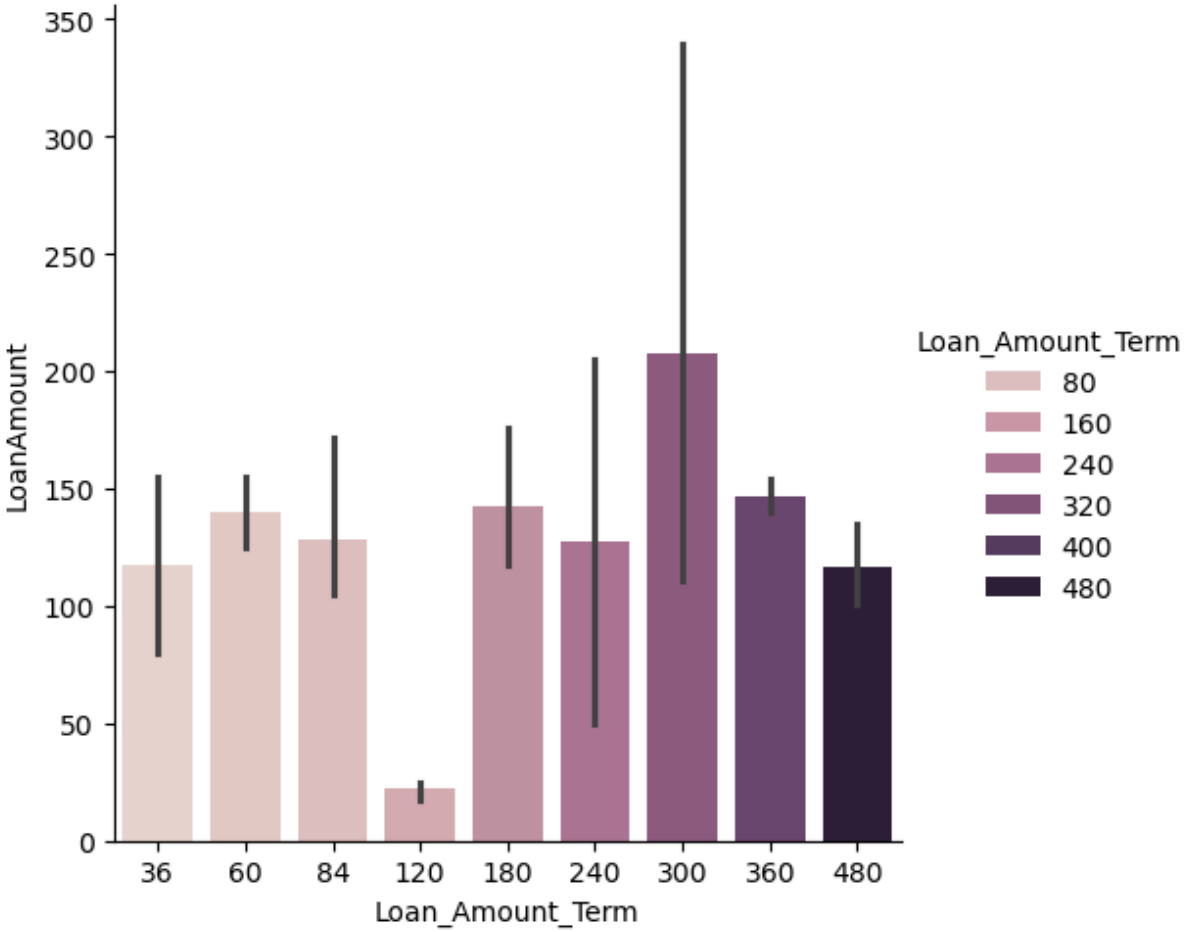


In [282…

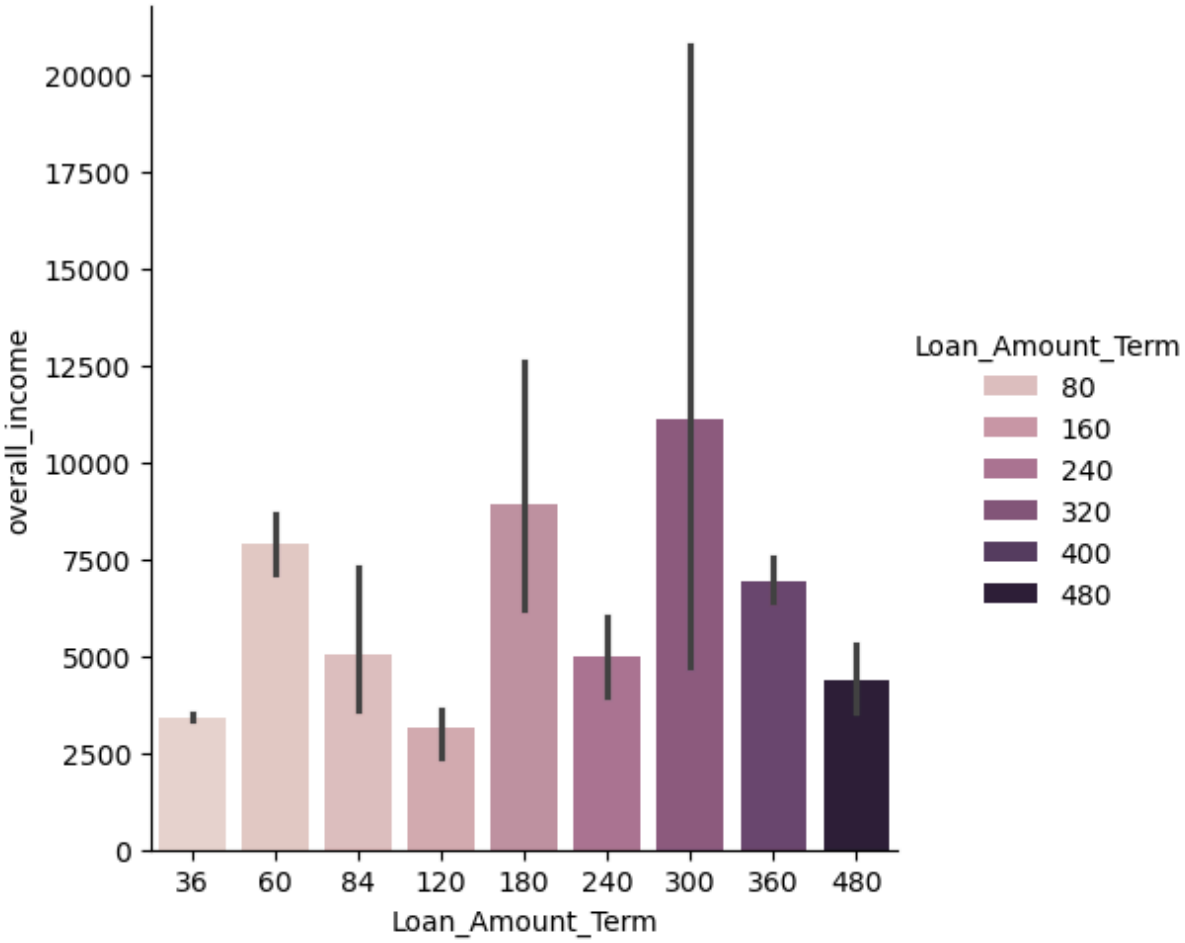
```
sns.catplot(x="Dependents",y="LoanAmount",data=df,kind="bar",hue="Loan_Amount_Term"
plt.show()
```



In [283…
```
sns.catplot(x="Loan_Amount_Term",y="LoanAmount",data=df,kind="bar",hue="Loan_Amount
plt.show()
```



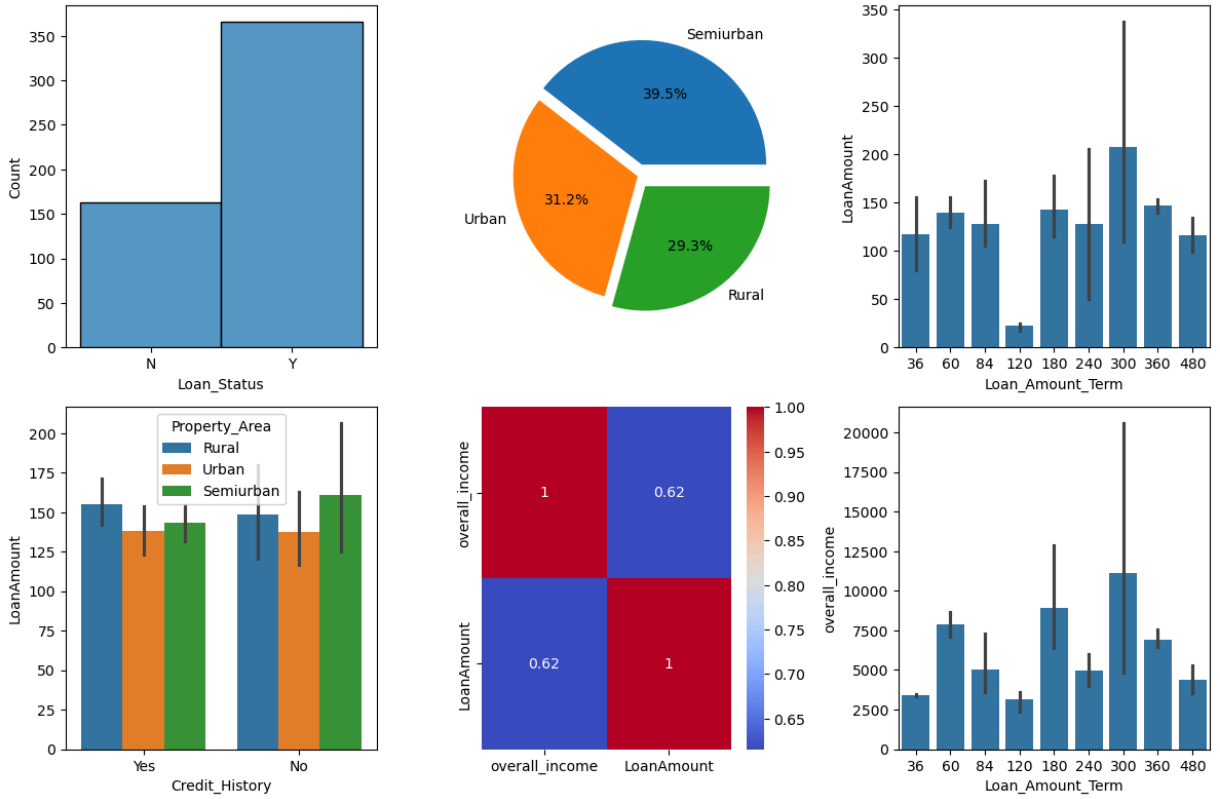
In [285…
```
sns.catplot(x="Loan_Amount_Term",y="overall_income",data=df,kind="bar",hue="Loan_Am
plt.show()
```



In [288…

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12,8))

# 1. Loan_Status distribution
plt.subplot(2,3,1)
sns.histplot(df["Loan_Status"], bins=10)

# 2. Gender pie chart
plt.subplot(2,3,2)
plt.pie(x=df["Property_Area"].value_counts(),
        labels = df["Property_Area"].value_counts().index.tolist(),
        autopct = "%0.1f%%",
        explode=[0.1,0,0.1])

# 3. Loan_Amount_Term vs LoanAmount  # it is accept BARPLOT NOT USED CATPLOT AND NO
plt.subplot(2,3,3)
sns.barplot(x="Loan_Amount_Term", y="LoanAmount", data=df)

# 4. Credit_History & Property_Area vs LoanAmount  # it is accept BARPLOT NOT USED
plt.subplot(2,3,4)
sns.barplot(x="Credit_History", y="LoanAmount", hue="Property_Area", data=df)

# 5. Heatmap for correlation
plt.subplot(2,3,5)
```

```
hm = df[["overall_income","LoanAmount"]].corr()
sns.heatmap(hm, annot=True, cmap="coolwarm", cbar=True)

# it is accept BARPLOT NOT USED CATPLOT AND NOT USED KIND = "BAR"
plt.subplot(2,3,6)
sns.barplot(x="Loan_Amount_Term",y="overall_income",data=df)

# Save & Show
plt.tight_layout()
plt.savefig("home_loan.png")
plt.show()
```



In [ ]:

In [ ]:

In [ ]: