

The goal of this project

To organize, clean and analyze for the following questions:

is the Loan ammount applied for affected by gender?

Does education affect credit history?

does education affect weekly applicant income?

If you are married do u have a higher weekly applicant income?

if u have a higher weekly applicant income do u get a higher loan ammount?

LOAN DATA PROJECT

The uncleaned dataset was taken from kaggle the link is :

<https://www.kaggle.com/datasets/architsharma01/loan-approval-prediction-dataset>

What the dataset is used for?

The loan approval dataset is a collection of financial records and associated information used to determine the eligibility of individuals or organizations for obtaining loans from a lending institution.

It includes various factors such as cibil score, income, employment status, loan term, loan amount, assets value, and loan status. This dataset is commonly used in machine

learning and data analysis to develop models and algorithms that predict the likelihood of loan approval based on the given features.

Problems Identified early by scoping the dataset.

we can see here a number of things have to be changed.

dependents column 3+ should be 3 and state in footer 3 means more than 3 dependents.

Nan values in self employed change that to Data not provided.

Change loan ammount times 1000 to show real ammount.

loan ammount term shows weekly so rename column.

change 0.0 in credit history to 0.

create a column for yearly, weekly and daily repayments.

missing values in gender dependents self employed loanammount loan ammount term and credit history

```
In [ ]: df.head(20)
# we can see here a number of things have to be changed.
# dependents column 3+ should be 3 and state in footer 3 means more than 3 dependents.
# Nan values in self employed change that to Data not provided.
# Change loan ammount times 1000 to show real ammount.
# loan ammount term shows weekly so rename column.
# change 0.0 in credit history to 0.
# create a column for yearly, weekly and daily repayments.
```

Out[]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	
5	LP001054	Male	Yes	0	Not Graduate	Yes	2165	3422	152.0	
6	LP001055	Female	No	1	Not Graduate	No	2226	0	59.0	
7	LP001056	Male	Yes	2	Not Graduate	No	3881	0	147.0	
8	LP001059	Male	Yes	2	Graduate	NaN	13633	0	280.0	
9	LP001067	Male	No	0	Not Graduate	No	2400	2400	123.0	
10	LP001078	Male	No	0	Not Graduate	No	3091	0	90.0	
11	LP001082	Male	Yes	1	Graduate	NaN	2185	1516	162.0	
12	LP001083	Male	No	3+	Graduate	No	4166	0	40.0	
13	LP001094	Male	Yes	2	Graduate	NaN	12173	0	166.0	
14	LP001096	Female	No	0	Graduate	No	4666	0	124.0	
15	LP001099	Male	No	1	Graduate	No	5667	0	131.0	
16	LP001105	Male	Yes	2	Graduate	No	4583	2916	200.0	
17	LP001107	Male	Yes	3+	Graduate	No	3786	333	126.0	
18	LP001108	Male	Yes	0	Graduate	No	9226	7916	300.0	
19	LP001115	Male	No	0	Graduate	No	1300	3470	100.0	

In [174...

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
Column Non-Null Count Dtype
--- -
0 Loan_ID 367 non-null object
1 Gender 356 non-null object
2 Married 367 non-null object
3 Dependents 357 non-null object
4 Education 367 non-null object
5 Self_Employed 344 non-null object
6 ApplicantIncome 367 non-null int64
7 CoapplicantIncome 367 non-null int64
8 LoanAmount 362 non-null float64
9 Loan_Amount_Term 361 non-null float64
10 Credit_History 338 non-null float64
11 Property_Area 367 non-null object
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB

In [175...

#see stats for your data.
df.describe()

Out[175...

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	367.000000	367.000000	362.000000	361.000000	338.000000
mean	4805.599455	1569.577657	136.132597	342.537396	0.825444
std	4910.685399	2334.232099	61.366652	65.156643	0.380150
min	0.000000	0.000000	28.000000	6.000000	0.000000
25%	2864.000000	0.000000	100.250000	360.000000	1.000000
50%	3786.000000	1025.000000	125.000000	360.000000	1.000000
75%	5060.000000	2430.500000	158.000000	360.000000	1.000000
max	72529.000000	24000.000000	550.000000	480.000000	1.000000

```
In [176.. # check for missing values in all columns
```

```
Out[176.. Loan_ID      0
Gender      11
Married     0
Dependents  10
Education   0
Self_Employed 23
ApplicantIncome 0
CoapplicantIncome 0
LoanAmount  5
Loan_Amount_Term 6
Credit_History 29
Property_Area 0
dtype: int64
```

```
In [179.. # find all missing gender values and delete them
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan
	22	LP001128	NaN	No	0	Graduate	No	3909	0	101.0
	51	LP001287	NaN	Yes	3+	Not Graduate	No	3500	833	120.0
	106	LP001563	NaN	No	0	Graduate	No	1596	1760	119.0
	138	LP001769	NaN	No	NaN	Graduate	No	3333	1250	110.0
	209	LP002165	NaN	No	1	Not Graduate	No	2038	4027	100.0
	231	LP002298	NaN	No	0	Graduate	Yes	2860	2988	138.0
	245	LP002355	NaN	Yes	0	Graduate	No	3186	3145	150.0
	279	LP002553	NaN	No	0	Graduate	No	29167	0	185.0
	296	LP002614	NaN	No	0	Graduate	No	6478	0	108.0
	303	LP002657	NaN	Yes	1	Not Graduate	Yes	570	2125	68.0
	318	LP002775	NaN	No	0	Not Graduate	No	4768	0	125.0

```
In [180.. # drop rows there the name column has missing nan values.
df_copy = df_copy.dropna(subset=["Gender"])
```

```
In [181.. df_copy[df_copy['Gender'].isnull()]
# there is no null values in gender.
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Am
--	---------	--------	---------	------------	-----------	---------------	-----------------	-------------------	------------	---------

```
In [183.. df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Loan_ID               367 non-null    object
1   Gender                356 non-null    object
2   Married               367 non-null    object
3   Dependents            357 non-null    object
4   Education             367 non-null    object
5   Self_Employed         344 non-null    object
6   ApplicantIncome       367 non-null    int64
7   CoapplicantIncome     367 non-null    int64
8   LoanAmount            362 non-null    float64
9   Loan_Amount_Term      361 non-null    float64
10  Credit_History        338 non-null    float64
11  Property_Area         367 non-null    object
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
```

Solutions to clean:

next is dependents

change nan info not provided

dependents column 3+ should be 3 and state in footer 3 means more than 3 dependents.

change dependents to 0 for Nan in dependents. we assume its 0 because no information is given.

now u have changed this change the data type for dependents to numerical so u can analyze this

now we deal with loan ammount drop all the missing entries for loan ammount because the accountant has screwed up obviously and the application needs to be redone.

remove all entries with missing values in loan ammount term as customer will have to redo application

remove all missing values for loan ammount as application will have to be redone.

now we only have missing values for credit history.

change all missing values to 0.

there are some values in credit history with 0.0

replace all instances of 0.0 in credit history with 0.

rename loan ammount term column to weekly loan ammount term

drop the .0 in Loan ammount term by switching to integer.

credit history should also be one integer

applicant income is per week as well as coapplicant income. so rename columns

need to times the loan ammount by a thousand for each loan ammount in whole dataset.

pick the loan ammount column and change to times 1000 answer.

change loan ammount to integer

CREATE 4 NEW COLUMNS WITH DATA CALCULATIONS FOR YEARLY MONTHLY WEEKLY AND DAILY REPAYMENTS FOR THE LOANS.

create a new copy of the changed dataframe, then do a IQR analysis to deal with the error.

after this do graphs to answer the questions.

```
In [184... # next is dependents
# change nan info not provided
# dependents column 3+ should be 3 and state in footer 3 means more than 3 dependents.
#change dependents to 0 for Nan in dependents. we assume its 0 because no information is given.
# now u have changed this change the data type for dependents to numerical so u can analyze this .
```

```
In [185... # next is delf employed
# change all NAN in self employed to no as no data is provided meaning that the customer didnt feel the need to
```

```
In [187... # now we deal with loan ammount drop all the missing entries for loan ammount because the accountant has screw
# remove all entries with missing values in loan ammount term as customer will have to redo application
```

```
In [188... # there are missing values in loan ammount term, in the same vein the application will have to be redone we need
```

```
In [189... df_copy.isnull().sum()
```

```
Out[189... Loan_ID          0
Gender          0
Married         0
Dependents      0
Education       0
Self_Employed   0
ApplicantIncome 0
CoapplicantIncome 0
LoanAmount      0
Loan_Amount_Term 0
Credit_History  28
Property_Area   0
dtype: int64
```

```
In [190... # now we only have missing values for credit history.
# change all missing values to 0.
# there are some values in credit history with 0.0
# replace all instances of 0.0 in credit history with 0.
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_19124\2315742299.py:3: FutureWarning: Setting an item of incompatible
dtype is deprecated and will raise an error in a future version of pandas. Value '0' has dtype incompatible with
float64, please explicitly cast to a compatible dtype first.
df_copy.loc[df_copy['Credit_History'].isnull(), 'Credit_History'] = '0'
```

```
In [191... # dealt with all missing values and data cleaning lets now look at the data.
```

Out[191... Loan_ID 0
Gender 0
Married 0
Dependents 0
Education 0
Self_Employed 0
ApplicantIncome 0
CoapplicantIncome 0
LoanAmount 0
Loan_Amount_Term 0
Credit_History 0
Property_Area 0
dtype: int64

```
In [194... # rename loan ammount term column to weekly loan ammount term

# drop the .0 in Loan ammount term by switching to integer.

# credit history should also be one integer

# applicant income is per week as well as coapplicant income. so rename columns

# need to times the loan ammount by a thousand for each loan ammount in whole dataset.

# pick the loan ammount column and change to times 1000 answer.
# change loan ammount to integer
```

```
In [195... df_copy.head(10)
# columns have now been renamed see below.
```

Out[195...

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Weekly_Applicant_Income	Weekly_Coapplicant_Income	Loa
0	LP001015	Male	Yes	0	Graduate	No	5720	0	
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	
4	LP001051	Male	No	0	Not Graduate	No	3276	0	
5	LP001054	Male	Yes	0	Not Graduate	Yes	2165	3422	
6	LP001055	Female	No	1	Not Graduate	No	2226	0	
7	LP001056	Male	Yes	2	Not Graduate	No	3881	0	
8	LP001059	Male	Yes	2	Graduate	No	13633	0	
9	LP001067	Male	No	0	Not Graduate	No	2400	2400	

```
In [196... # CREATE 4 NEW COLUMNS WITH DATA CALCULATIONS FOR YEARLY MONTHLY WEEKLY AND DAILY REPAYMENTS FOR THE LOANS.
```

```
In [197...
```

Out[197...

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Weekly_Applicant_Income	Weekly_Coapplicant_Income	Loan_Amount
0	LP001015	Male	Yes	0	Graduate	No	5720		0
1	LP001022	Male	Yes	1	Graduate	No	3076		1500
2	LP001031	Male	Yes	2	Graduate	No	5000		1800
3	LP001035	Male	Yes	2	Graduate	No	2340		2546
4	LP001051	Male	No	0	Not Graduate	No	3276		0
5	LP001054	Male	Yes	0	Not Graduate	Yes	2165		3422
6	LP001055	Female	No	1	Not Graduate	No	2226		0
7	LP001056	Male	Yes	2	Not Graduate	No	3881		0
8	LP001059	Male	Yes	2	Graduate	No	13633		0
9	LP001067	Male	No	0	Not Graduate	No	2400		2400

In [198...

```
# Modify the copy (original DataFrame remains unchanged)

# Display the original and copied DataFrame
# print("Original DataFrame:")
# print(df)
# print("\nCopied DataFrame (formatted):")
# print(df_copy)
```

In [199...

```
df_copy.head(5)
```

Out[199...

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Weekly_Applicant_Income	Weekly_Coapplicant_Income	Loan_Amount
0	LP001015	Male	Yes	0	Graduate	No	5720		0
1	LP001022	Male	Yes	1	Graduate	No	3076		1500
2	LP001031	Male	Yes	2	Graduate	No	5000		1800
3	LP001035	Male	Yes	2	Graduate	No	2340		2546
4	LP001051	Male	No	0	Not Graduate	No	3276		0

In [200...

```
# rename monthly column to Monthly_repayments
```

In [201...

```
# Rounded all repayment columns to the nearest 0.10

# the data is clean now. Lets account for outliers.
```

In [202...

```
df_copy.head(5)
```

Out[202...

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Weekly_Applicant_Income	Weekly_Coapplicant_Income	Loan_Amount
0	LP001015	Male	Yes	0	Graduate	No	5720		0
1	LP001022	Male	Yes	1	Graduate	No	3076		1500
2	LP001031	Male	Yes	2	Graduate	No	5000		1800
3	LP001035	Male	Yes	2	Graduate	No	2340		2546
4	LP001051	Male	No	0	Not Graduate	No	3276		0

In [203...

```
# As we are not testing hypothesis we can account for outliers now
# first remove outliers from only numerical columns using Interquartile range

# Step 1: Read data from a CSV file
# Replace 'your_file.csv' with the path to your CSV file
# file_path = 'data-1.csv'

# Read the CSV file into a DataFrame
# df = pd.read_csv('data-1.csv')
```

```

# Step 2: Specify the columns to analyze
# Replace with the list of columns you want to process

# Ensure the specified columns exist in the DataFrame

# Step 3: Remove outliers from each specified column
# Create a copy of the original DataFrame to store filtered data

# Calculate IQR for the column

# Define the lower and upper bounds for outliers

# Filter out the outliers for the column

# Step 4: Save the filtered DataFrame (optional)
# Uncomment the following line to save the filtered data to a new CSV file
# df_filtered.to_csv('filtered_data.csv', index=False)

# Print the original and filtered DataFrames

# Print the number of rows removed

```

Original DataFrame:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	\
0	LP001015	Male	Yes	0	Graduate	No	
1	LP001022	Male	Yes	1	Graduate	No	
2	LP001031	Male	Yes	2	Graduate	No	
3	LP001035	Male	Yes	2	Graduate	No	
4	LP001051	Male	No	0	Not Graduate	No	
..	
362	LP002971	Male	Yes	3	Not Graduate	Yes	
363	LP002975	Male	Yes	0	Graduate	No	
364	LP002980	Male	No	0	Graduate	No	
365	LP002986	Male	Yes	0	Graduate	No	
366	LP002989	Male	No	0	Graduate	Yes	

	Weekly_Applicant_Income	Weekly_Coapplicant_Income	LoanAmount	\
0	5720	0	110000	
1	3076	1500	126000	
2	5000	1800	208000	
3	2340	2546	100000	
4	3276	0	78000	
..	
362	4009	1777	113000	
363	4158	709	115000	
364	3250	1993	126000	
365	5000	2393	158000	
366	9200	0	98000	

	Weekly_Loan_Amount_Term	Credit_History	Property_Area	\
0	360	1	Urban	
1	360	1	Urban	
2	360	1	Urban	
3	360	0	Urban	
4	360	1	Urban	
..	
362	360	1	Urban	
363	360	1	Urban	
364	360	0	Semiurban	
365	360	1	Rural	
366	180	1	Rural	

	Yearly_Loan_Amount_Term	Yearly_repayments	Monthly	\
0	6.923077	15888.888889	1324.074074	
1	6.923077	18200.000000	1516.666667	
2	6.923077	30044.444444	2503.703704	
3	6.923077	14444.444444	1203.703704	
4	6.923077	11266.666667	938.888889	
..	
362	6.923077	16322.222222	1360.185185	
363	6.923077	16611.111111	1384.259259	
364	6.923077	18200.000000	1516.666667	
365	6.923077	22822.222222	1901.851852	
366	3.461538	28311.111111	2359.259259	

	Weekly_repayments	Daily_repayments
0	305.555556	43.650794
1	350.000000	50.000000
2	577.777778	82.539683
3	277.777778	39.682540
4	216.666667	30.952381
..
362	313.888889	44.841270
363	319.444444	45.634921
364	350.000000	50.000000
365	438.888889	62.698413
366	544.444444	77.777778

[345 rows x 17 columns]

Filtered DataFrame (without outliers):

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	\
0	LP001015	Male	Yes	0	Graduate	No	
1	LP001022	Male	Yes	1	Graduate	No	
2	LP001031	Male	Yes	2	Graduate	No	
3	LP001035	Male	Yes	2	Graduate	No	
4	LP001051	Male	No	0	Not Graduate	No	
..	
361	LP002969	Male	Yes	1	Graduate	No	
362	LP002971	Male	Yes	3	Not Graduate	Yes	
363	LP002975	Male	Yes	0	Graduate	No	
364	LP002980	Male	No	0	Graduate	No	
365	LP002986	Male	Yes	0	Graduate	No	

	Weekly_Applicant_Income	Weekly_Coapplicant_Income	LoanAmount	\
0	5720	0	110000	
1	3076	1500	126000	
2	5000	1800	208000	
3	2340	2546	100000	
4	3276	0	78000	
..	
361	2269	2167	99000	
362	4009	1777	113000	
363	4158	709	115000	
364	3250	1993	126000	
365	5000	2393	158000	

	Weekly_Loan_Amount_Term	Credit_History	Property_Area	\
0	360	1	Urban	
1	360	1	Urban	
2	360	1	Urban	
3	360	0	Urban	
4	360	1	Urban	
..	
361	360	1	Semiurban	
362	360	1	Urban	
363	360	1	Urban	
364	360	0	Semiurban	
365	360	1	Rural	

	Yearly_Loan_Amount_Term	Yearly_repayments	Monthly	\
0	6.923077	15888.888889	1324.074074	
1	6.923077	18200.000000	1516.666667	
2	6.923077	30044.444444	2503.703704	
3	6.923077	14444.444444	1203.703704	
4	6.923077	11266.666667	938.888889	
..	
361	6.923077	14300.000000	1191.666667	
362	6.923077	16322.222222	1360.185185	
363	6.923077	16611.111111	1384.259259	
364	6.923077	18200.000000	1516.666667	
365	6.923077	22822.222222	1901.851852	

	Weekly_repayments	Daily_repayments
0	305.555556	43.650794
1	350.000000	50.000000
2	577.777778	82.539683
3	277.777778	39.682540
4	216.666667	30.952381
..
361	275.000000	39.285714
362	313.888889	44.841270
363	319.444444	45.634921
364	350.000000	50.000000
365	438.888889	62.698413

[311 rows x 17 columns]

Number of rows removed: 34

In [204...] df_filtered

```
Out[204...]      Loan_ID  Gender  Married  Dependents  Education  Self_Employed  Weekly_Applicant_Income  Weekly_Coapplicant_Income  L
```

0	LP001015	Male	Yes	0	Graduate	No	5720	0
1	LP001022	Male	Yes	1	Graduate	No	3076	1500
2	LP001031	Male	Yes	2	Graduate	No	5000	1800
3	LP001035	Male	Yes	2	Graduate	No	2340	2546
4	LP001051	Male	No	0	Not Graduate	No	3276	0
...
361	LP002969	Male	Yes	1	Graduate	No	2269	2167
362	LP002971	Male	Yes	3	Not Graduate	Yes	4009	1777
363	LP002975	Male	Yes	0	Graduate	No	4158	709
364	LP002980	Male	No	0	Graduate	No	3250	1993
365	LP002986	Male	Yes	0	Graduate	No	5000	2393

311 rows × 17 columns

◀ ▶

```
In [205...] # adjust for changes that didnt reflect in df filtered.
# rename monthly column to Monthly_repayments

# Rounded all repayment columns to the nearest 0.10
```

In [206...] df_filtered.head(5)

```
Out[206...]      Loan_ID  Gender  Married  Dependents  Education  Self_Employed  Weekly_Applicant_Income  Weekly_Coapplicant_Income  Loa
```

0	LP001015	Male	Yes	0	Graduate	No	5720	0
1	LP001022	Male	Yes	1	Graduate	No	3076	1500
2	LP001031	Male	Yes	2	Graduate	No	5000	1800
3	LP001035	Male	Yes	2	Graduate	No	2340	2546
4	LP001051	Male	No	0	Not Graduate	No	3276	0

◀ ▶

In [207...] df_filtered.isnull().sum()

```
Out[207...] Loan_ID      0
Gender      0
Married     0
Dependents  0
Education   0
Self_Employed  0
Weekly_Applicant_Income  0
Weekly_Coapplicant_Income  0
LoanAmount  0
Weekly_Loan_Ammount_Term  0
Credit_History  0
Property_Area  0
Yearly_Loan_Ammount_Term  0
Yearly_repayments  0
Monthly_repayments  0
Weekly_repayments  0
Daily_repayments  0
dtype: int64
```

In [208...] df_filtered.shape

Out[208...] (311, 17)

In [209...] df_filtered.info()

```
<class 'pandas.core.frame.DataFrame'>
Index: 311 entries, 0 to 365
Data columns (total 17 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Loan_ID                               311 non-null    object
1   Gender                                311 non-null    object
2   Married                               311 non-null    object
3   Dependents                            311 non-null    int32
4   Education                             311 non-null    object
5   Self_Employed                         311 non-null    object
6   Weekly_Applicant_Income               311 non-null    int64
7   Weekly_Coapplicant_Income            311 non-null    int64
8   LoanAmount                           311 non-null    int32
9   Weekly_Loan_Amount_Term              311 non-null    int32
10  Credit_History                        311 non-null    int32
11  Property_Area                         311 non-null    object
12  Yearly_Loan_Amount_Term              311 non-null    float64
13  Yearly_repayments                    311 non-null    float64
14  Monthly_repayments                   311 non-null    float64
15  Weekly_repayments                    311 non-null    float64
16  Daily_repayments                     311 non-null    float64
dtypes: float64(5), int32(4), int64(2), object(6)
memory usage: 38.9+ KB
```

```
In [210.. # 302 is bigger than 30 therefore we pay attention to z test.
# if the sample size was smaller than 30 we pay attention to the t test.
```

```
In [211.. # df = pd.DataFrame(df_filtered)
"""
Perform a z-test for two independent samples.
"""

# Pooled standard error

# Z-statistic

# P-value (two-tailed)

def perform_tests(df_filtered, LoanAmount, Weekly_Applicant_Income, alpha=0.05):
    """
    Perform t-test and z-test on two columns from a CSV file.
    """
    # Load data from CSV
    # df = pd.DataFrame(df_filtered)

    # Extract the two columns

    # Perform t-test

    # Perform z-test

    # Print results
```

T-Test Results:
T-statistic: 49.76178401070807
P-value: 1.0884762280372716e-218
Reject the null hypothesis (significant difference between groups).

Z-Test Results:
Z-statistic: 49.76178401070804
P-value: 0.0
Reject the null hypothesis (significant difference between groups).

```
In [212.. df_filtered
```

Out [212...

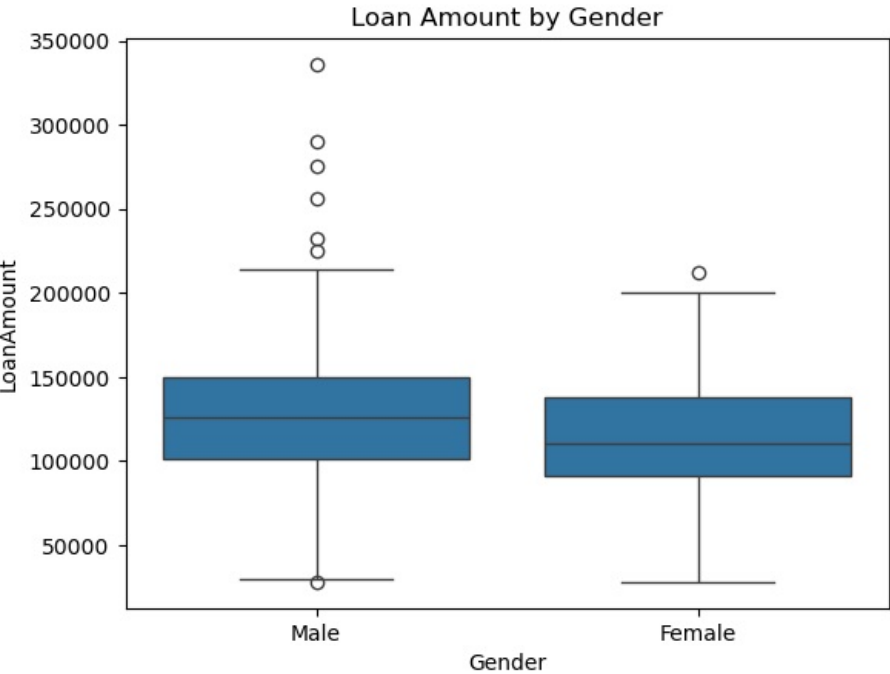
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Weekly_Applicant_Income	Weekly_Coapplicant_Income	L
0	LP001015	Male	Yes	0	Graduate	No	5720	0	
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	
4	LP001051	Male	No	0	Not Graduate	No	3276	0	
...	
361	LP002969	Male	Yes	1	Graduate	No	2269	2167	
362	LP002971	Male	Yes	3	Not Graduate	Yes	4009	1777	
363	LP002975	Male	Yes	0	Graduate	No	4158	709	
364	LP002980	Male	No	0	Graduate	No	3250	1993	
365	LP002986	Male	Yes	0	Graduate	No	5000	2393	

311 rows × 17 columns

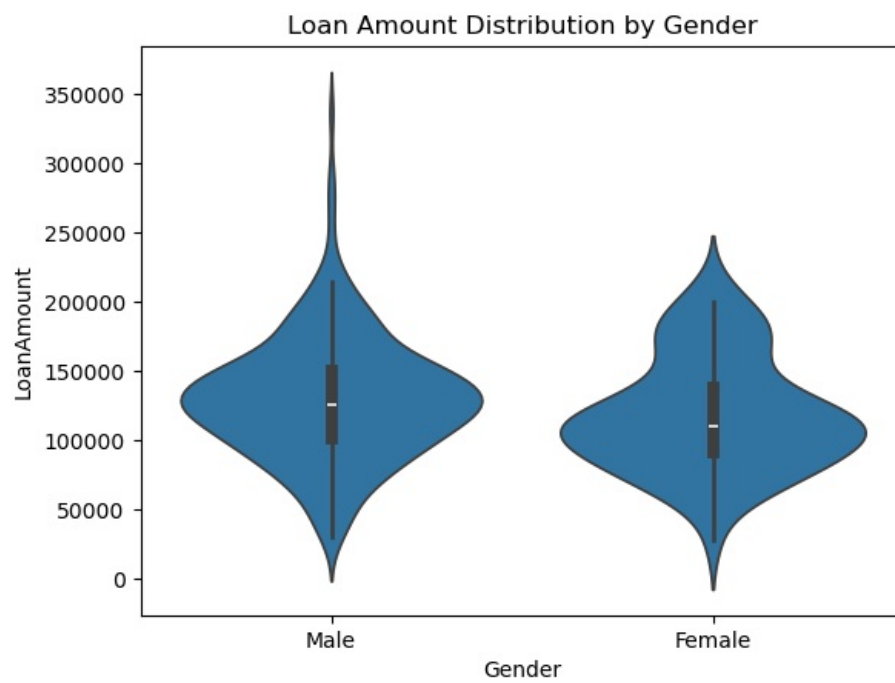


In [213...

```
# now for charts.  
# Assuming df is your DataFrame with 'Gender' and 'Loan Amount' columns
```

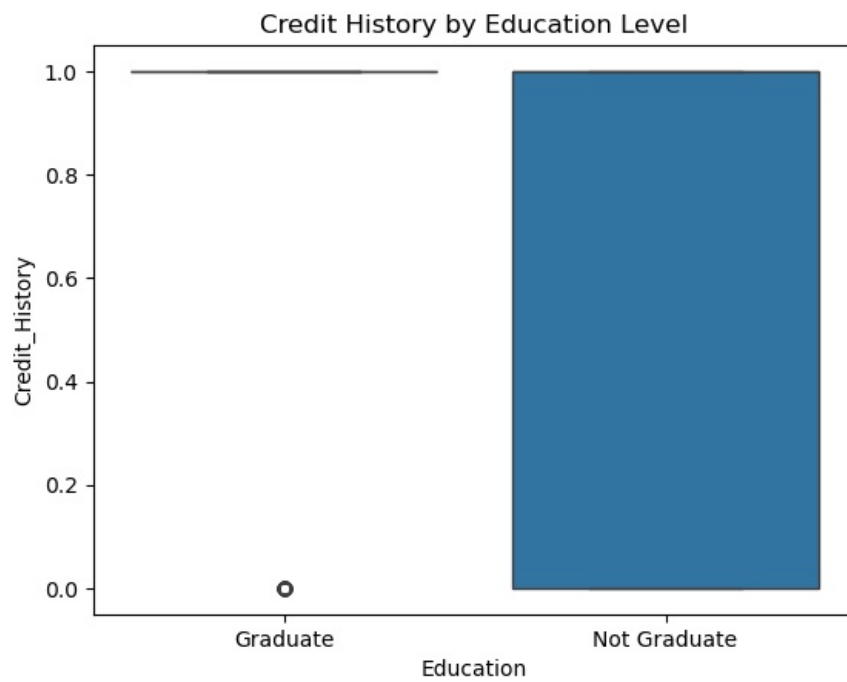


In [214...

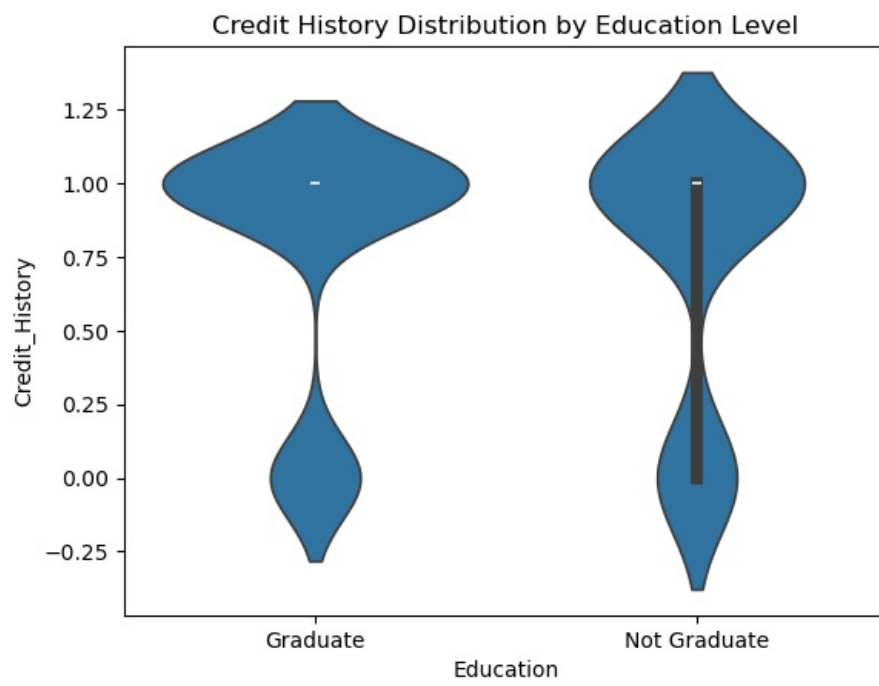


In [215.. # change credit history from object to numerical

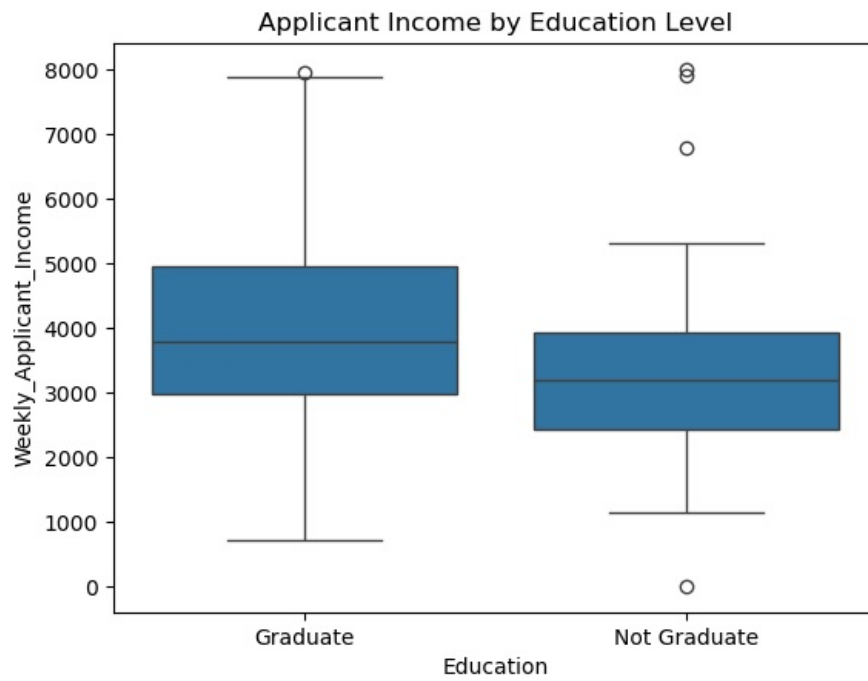
In [216..



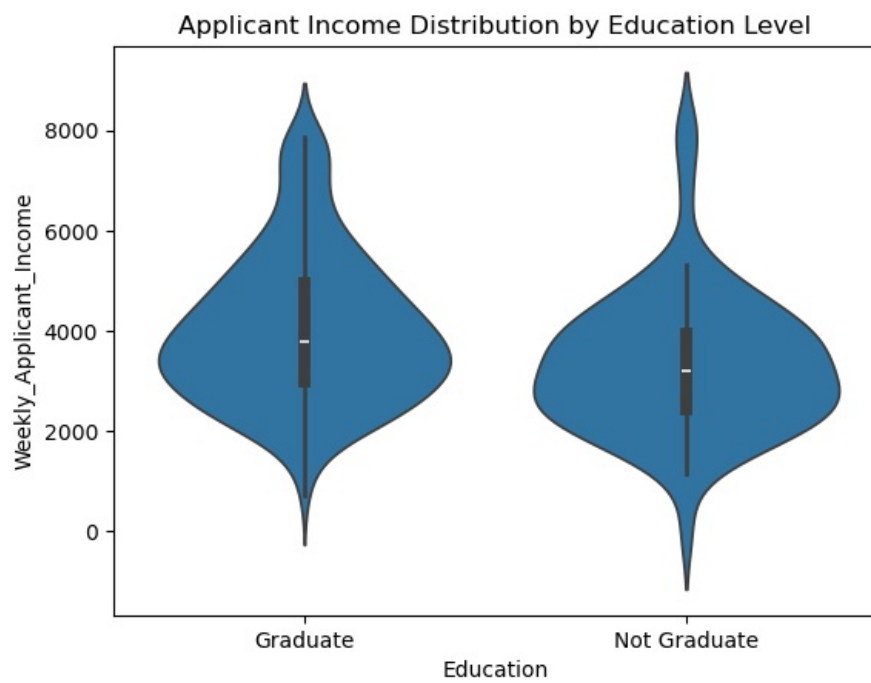
In [217..



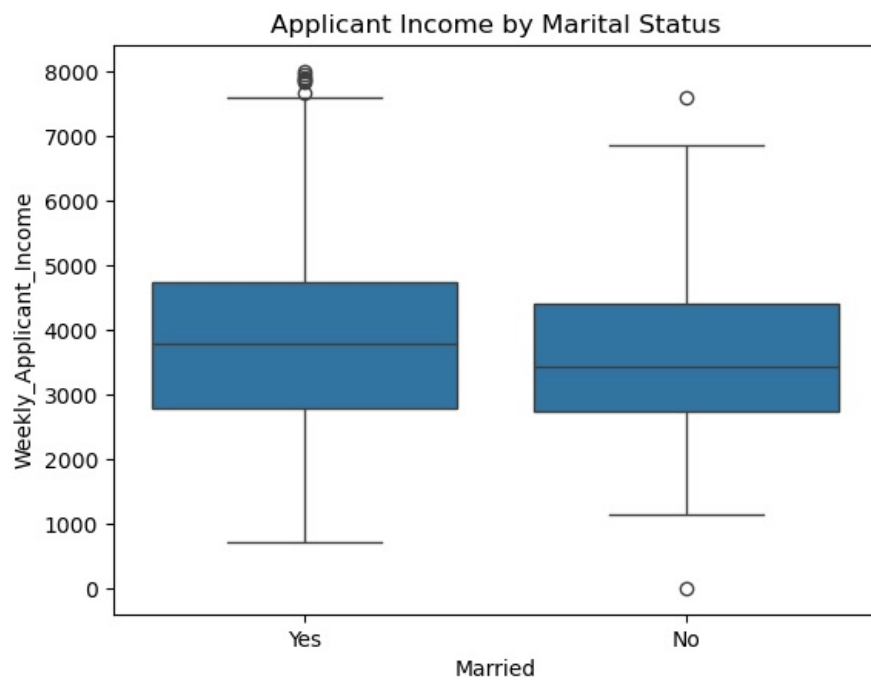
In [218..



In [219..



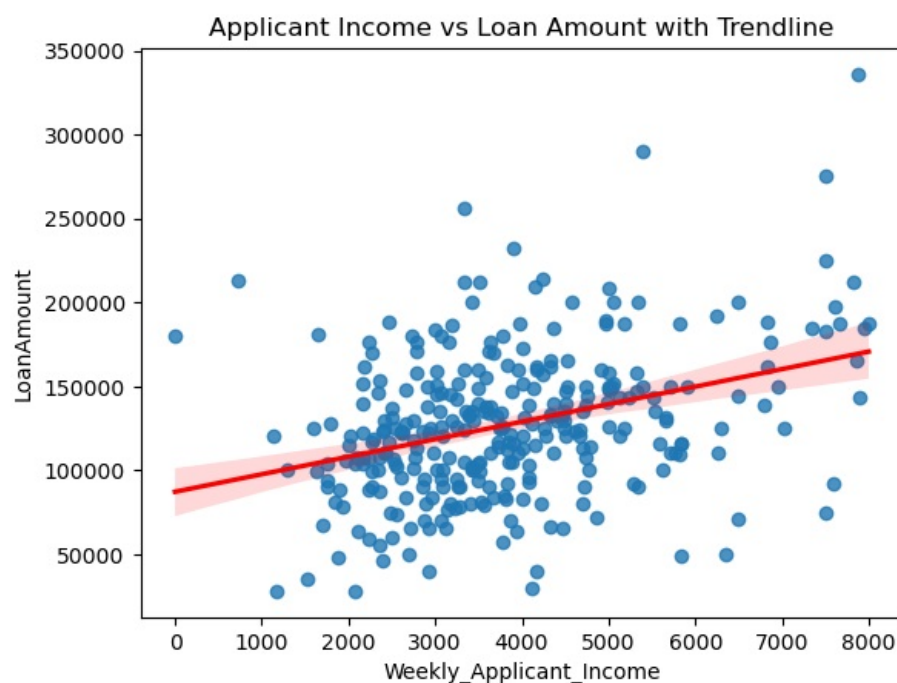
In [220]



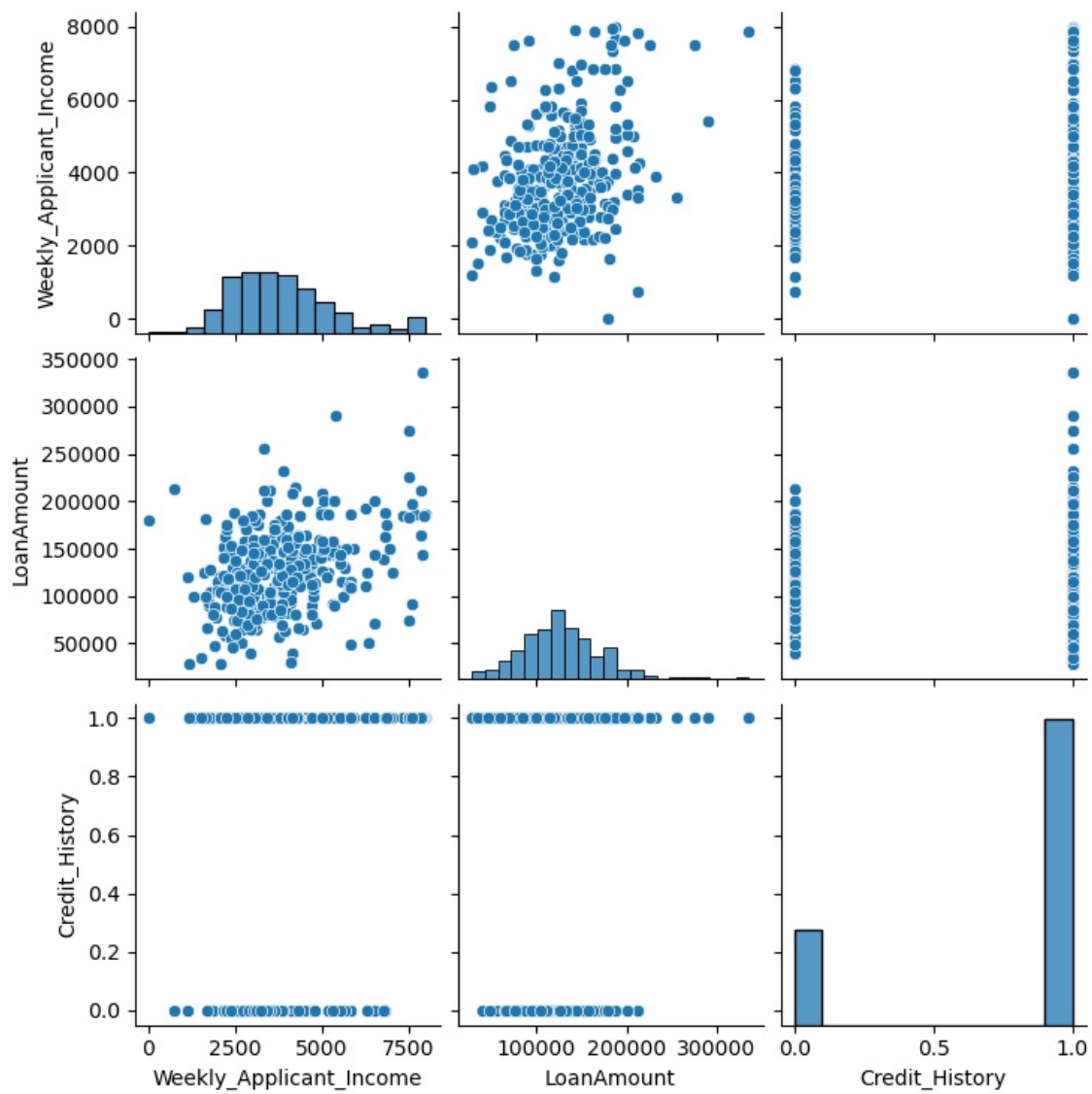
In [222]



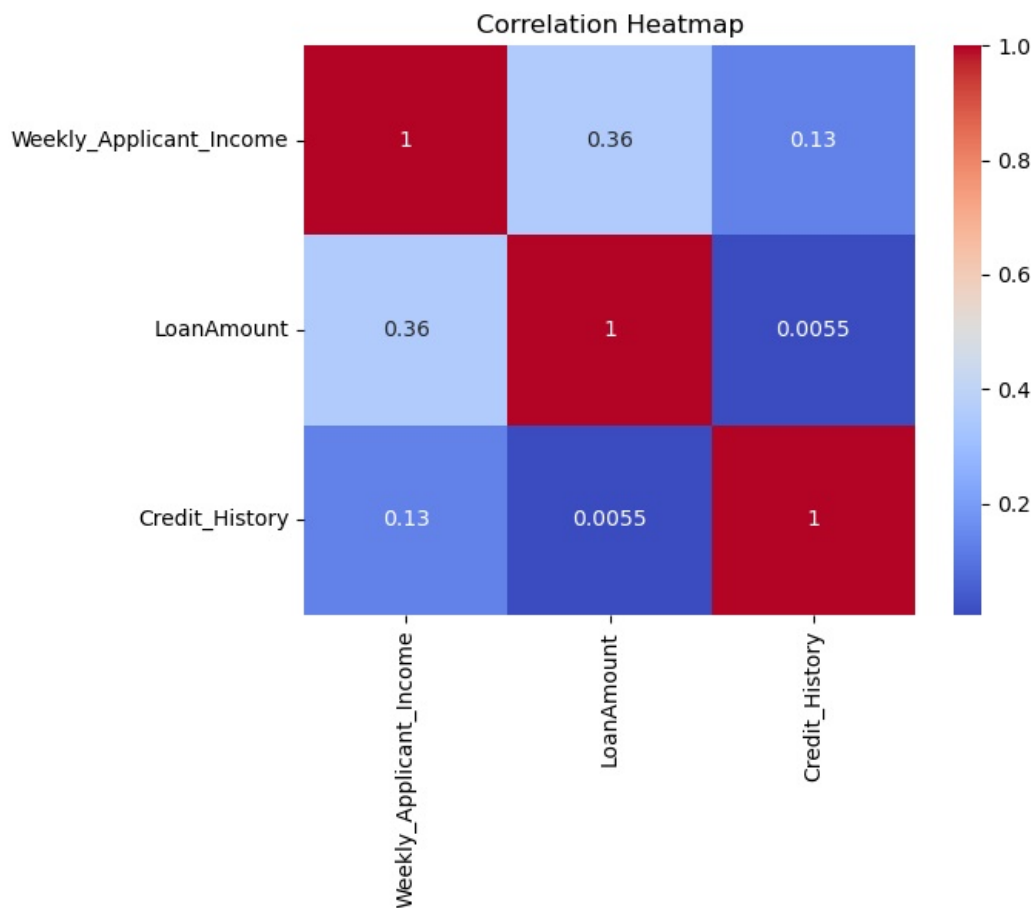
In []:



```
# can only do regression and pairplot with numerical variables.
# you can see a number of trends here.
### Applicant Income
# applicant Income has a slightly positive relationship with Loan ammount given (the more income u have the bet
# customers with a 1 credit history have a higher applicant income.
### LOAN AMMOUNT GIVEN
# Loan ammount given doesnt have a relationship with applicant income. As it depends on applicant income.
# credit history affects your loan ammount from the bank
### CREDIT HISTORY.
# the better your credit history the higher the applicant income.
# the better the credit history the higher the loan ammount given.
```



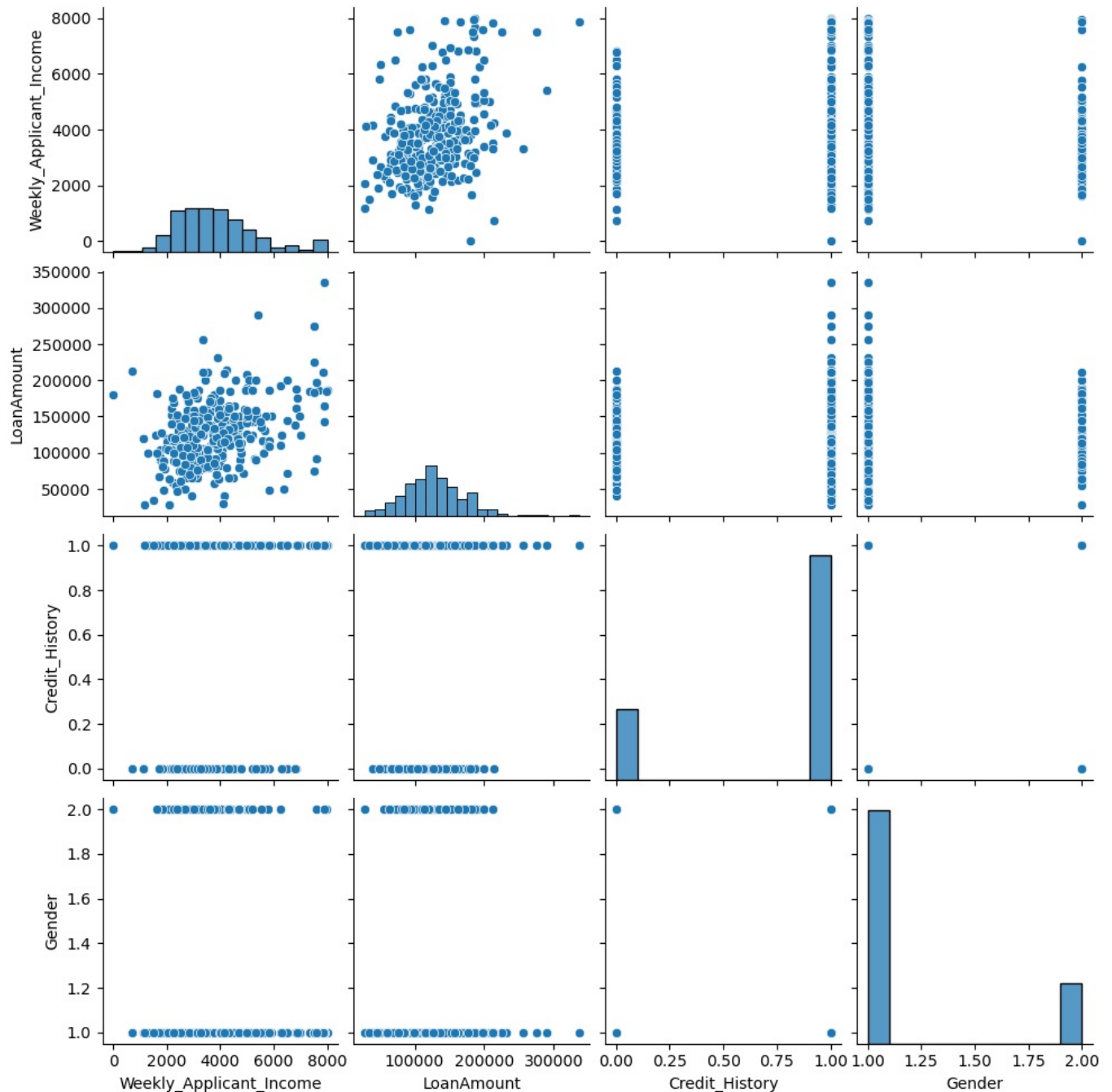
In [225...



In [226... # Change gender datatype to 0 and 1 to be seen

In [227...

Out[227... <function matplotlib.pyplot.show(close=None, block=None)>



ANSWER THE Original Questions of this project

is the Loan ammount applied for affected by gender?

In the graph above 1 is male and 2 is female. Males clearly burrow more notice the extra data pieces in the 300 000 range.

Does education affect credit history?

Yes refer to the violin plot the data is concentrated at a credit history of 1. Meaning that there is a positive correlation between eduction vs credit history.

does education affect weekly applicant income?

Yes refer to box plot Applicant income vs eduction level, the average is higher.

If you are married do u have a higher weekly applicant income?

Refer to Applicant income by Marital status viloin plot. The average and distribution at 8000 is higher. Yes if u are married u have a higher weekly applicant income.

if u have a higher weekly applicant income do u get a higher loan ammount?

Refer to Applicant Income vs Loan Amount with Trendline. There is a positive correlation if u have a higher weekly applicant income u can apply for a higher loan ammount.