Importing Libraries

Suggested code may be subject to a license | | Hermanubis/CS4364_Project

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.express as px
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score
```

```
df = pd.read_csv("/content/train_u6lujuX_CVtuZ9i (1).csv")
```

```
df
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmo
0	LP001002	Male	No	0	Graduate	No	5849	0.0	N
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	12
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	6
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	12
4	LP001008	Male	No	0	Graduate	No	6000	0.0	14
...	...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	0.0	7
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	4
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	25
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	18
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	13

Next steps: Generate code with df | View recommended plots | New interactive sheet

Data Preparation and Cleaning

```
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
```

```
df.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

```
df.columns
```

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

```
df['Loan_Status'].replace('N',0,inplace=True)
df['Loan_Status'].replace('Y',1,inplace=True)
```

```
<ipython-input-7-d91fcf67ca7f>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series t
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on wh

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'


  df['Loan_Status'].replace('N',0,inplace=True)
<ipython-input-7-d91fcf67ca7f>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series t
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on wh

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'


  df['Loan_Status'].replace('Y',1,inplace=True)
<ipython-input-7-d91fcf67ca7f>:2: FutureWarning: Downcasting behavior in `replace` is deprecated and will be remo
  df['Loan_Status'].replace('Y',1,inplace=True)
```

```
df['Loan_Status'].value_counts()
```

Loan_Status	count
1	422
0	192

```
df.dtypes
```

	0
Loan_ID	object
Gender	object
Married	object
Dependents	object
Education	object
Self_Employed	object
ApplicantIncome	int64
CoapplicantIncome	float64
LoanAmount	float64
Loan_Amount_Term	float64
Credit_History	float64
Property_Area	object
Loan_Status	int64

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

	0
Loan_ID	0
Gender	13
Married	3
Dependents	15
Education	0
Self_Employed	32
ApplicantIncome	0
CoapplicantIncome	0
LoanAmount	22
Loan_Amount_Term	14
Credit_History	50
Property_Area	0
Loan_Status	0

```
cols_mode = ['Gender', 'Married', 'Dependents', 'Self_Employed', 'Credit_History', 'Loan_Amount_Term']
for col in cols_mode:
    df[col] = df[col].fillna(df[col].mode()[0])

df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].median())
```

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

	0
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
Loan_Status	0

```
df.dtypes
```

	0
Loan_ID	object
Gender	object
Married	object
Dependents	object
Education	object
Self_Employed	object
ApplicantIncome	int64
CoapplicantIncome	float64
LoanAmount	float64
Loan_Amount_Term	float64
Credit_History	float64
Property_Area	object
Loan_Status	int64

```
df.replace({'Married':{'No':0,'Yes':1},
            'Gender':{'Male':1,'Female':0},
            'Self_Employed':{'Yes':1,'No':0},
            'Property_Area':{'Rural':0,'Urban':1,'Semiurban':2},
```

```
            'Education':{'Graduate':1,'Not Graduate':0}},
           inplace=True)
```

```
<ipython-input-14-9954dfffe666>:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be rem
  df.replace({'Married':{'No':0,'Yes':1},
```

```
print(df['Dependents'].unique())
```

```
['0' '1' '2' '3+']
```

```
print(df['Loan_ID'].unique())
```

```
['LP001002' 'LP001003' 'LP001005' 'LP001006' 'LP001008' 'LP001011'
 'LP001013' 'LP001014' 'LP001018' 'LP001020' 'LP001024' 'LP001027'
 'LP001028' 'LP001029' 'LP001030' 'LP001032' 'LP001034' 'LP001036'
 'LP001038' 'LP001041' 'LP001043' 'LP001046' 'LP001047' 'LP001050'
 'LP001052' 'LP001066' 'LP001068' 'LP001073' 'LP001086' 'LP001087'
 'LP001091' 'LP001095' 'LP001097' 'LP001098' 'LP001100' 'LP001106'
 'LP001109' 'LP001112' 'LP001114' 'LP001116' 'LP001119' 'LP001120'
 'LP001123' 'LP001131' 'LP001136' 'LP001137' 'LP001138' 'LP001144'
 'LP001146' 'LP001151' 'LP001155' 'LP001157' 'LP001164' 'LP001179'
 'LP001186' 'LP001194' 'LP001195' 'LP001197' 'LP001198' 'LP001199'
 'LP001205' 'LP001206' 'LP001207' 'LP001213' 'LP001222' 'LP001225'
 'LP001228' 'LP001233' 'LP001238' 'LP001241' 'LP001243' 'LP001245'
 'LP001248' 'LP001250' 'LP001253' 'LP001255' 'LP001256' 'LP001259'
 'LP001263' 'LP001264' 'LP001265' 'LP001266' 'LP001267' 'LP001273'
 'LP001275' 'LP001279' 'LP001280' 'LP001282' 'LP001289' 'LP001310'
 'LP001316' 'LP001318' 'LP001319' 'LP001322' 'LP001325' 'LP001326'
 'LP001327' 'LP001333' 'LP001334' 'LP001343' 'LP001345' 'LP001349'
 'LP001350' 'LP001356' 'LP001357' 'LP001367' 'LP001369' 'LP001370'
 'LP001379' 'LP001384' 'LP001385' 'LP001387' 'LP001391' 'LP001392'
 'LP001398' 'LP001401' 'LP001404' 'LP001405' 'LP001421' 'LP001422'
 'LP001426' 'LP001430' 'LP001431' 'LP001432' 'LP001439' 'LP001443'
 'LP001448' 'LP001449' 'LP001451' 'LP001465' 'LP001469' 'LP001473'
 'LP001478' 'LP001482' 'LP001487' 'LP001488' 'LP001489' 'LP001491'
 'LP001492' 'LP001493' 'LP001497' 'LP001498' 'LP001504' 'LP001507'
 'LP001508' 'LP001514' 'LP001516' 'LP001518' 'LP001519' 'LP001520'
 'LP001528' 'LP001529' 'LP001531' 'LP001532' 'LP001535' 'LP001536'
 'LP001541' 'LP001543' 'LP001546' 'LP001552' 'LP001560' 'LP001562'
 'LP001565' 'LP001570' 'LP001572' 'LP001574' 'LP001577' 'LP001578'
 'LP001579' 'LP001580' 'LP001581' 'LP001585' 'LP001586' 'LP001594'
 'LP001603' 'LP001606' 'LP001608' 'LP001610' 'LP001616' 'LP001630'
 'LP001633' 'LP001634' 'LP001636' 'LP001637' 'LP001639' 'LP001640'
 'LP001641' 'LP001643' 'LP001644' 'LP001647' 'LP001653' 'LP001656'
 'LP001657' 'LP001658' 'LP001664' 'LP001665' 'LP001666' 'LP001669'
 'LP001671' 'LP001673' 'LP001674' 'LP001677' 'LP001682' 'LP001688'
 'LP001691' 'LP001692' 'LP001693' 'LP001698' 'LP001699' 'LP001702'
 'LP001708' 'LP001711' 'LP001713' 'LP001715' 'LP001716' 'LP001720'
 'LP001722' 'LP001726' 'LP001732' 'LP001734' 'LP001736' 'LP001743'
 'LP001744' 'LP001749' 'LP001750' 'LP001751' 'LP001754' 'LP001758'
 'LP001760' 'LP001761' 'LP001765' 'LP001768' 'LP001770' 'LP001776'
 'LP001778' 'LP001784' 'LP001786' 'LP001788' 'LP001790' 'LP001792'
 'LP001798' 'LP001800' 'LP001806' 'LP001807' 'LP001811' 'LP001813'
 'LP001814' 'LP001819' 'LP001824' 'LP001825' 'LP001835' 'LP001836'
 'LP001841' 'LP001843' 'LP001844' 'LP001846' 'LP001849' 'LP001854'
 'LP001859' 'LP001864' 'LP001865' 'LP001868' 'LP001870' 'LP001871'
 'LP001872' 'LP001875' 'LP001877' 'LP001882' 'LP001883' 'LP001884'
 'LP001888' 'LP001891' 'LP001892' 'LP001894' 'LP001896' 'LP001900'
 'LP001903' 'LP001904' 'LP001907' 'LP001908' 'LP001910' 'LP001914'
 'LP001915' 'LP001917' 'LP001922' 'LP001924' 'LP001925' 'LP001926'
 'LP001931' 'LP001935' 'LP001936' 'LP001938' 'LP001940' 'LP001945'
 'LP001947' 'LP001949' 'LP001953' 'LP001954' 'LP001955' 'LP001963'
 'LP001964' 'LP001972' 'LP001974' 'LP001977' 'LP001978' 'LP001990'
 'LP001993' 'LP001994' 'LP001996' 'LP001998' 'LP002002' 'LP002004'
 'LP002006' 'LP002008' 'LP002024' 'LP002031' 'LP002035' 'LP002036'
 'LP002043' 'LP002050' 'LP002051' 'LP002053' 'LP002054' 'LP002055'
 'LP002065' 'LP002067' 'LP002068' 'LP002082' 'LP002086' 'LP002087'
 'LP002097' 'LP002098' 'LP002100' 'LP002101' 'LP002103' 'LP002106'
 'LP002110' 'LP002112' 'LP002113' 'LP002114' 'LP002115' 'LP002116'
 'LP002119' 'LP002126' 'LP002128' 'LP002129' 'LP002130' 'LP002131'
```

```
le = LabelEncoder()
df['Loan_ID'] = le.fit_transform(df['Loan_ID'])
```

Exploratory Analysis and Visualization

```
df.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
0	0	1	0	0	1	0	5849	0.0	128.0
1	1	1	1	1	1	0	4583	1508.0	128.0
2	2	1	1	0	1	1	3000	0.0	66.0
3	3	1	1	0	0	0	2583	2358.0	120.0
4	4	1	0	0	1	0	6000	0.0	141.0

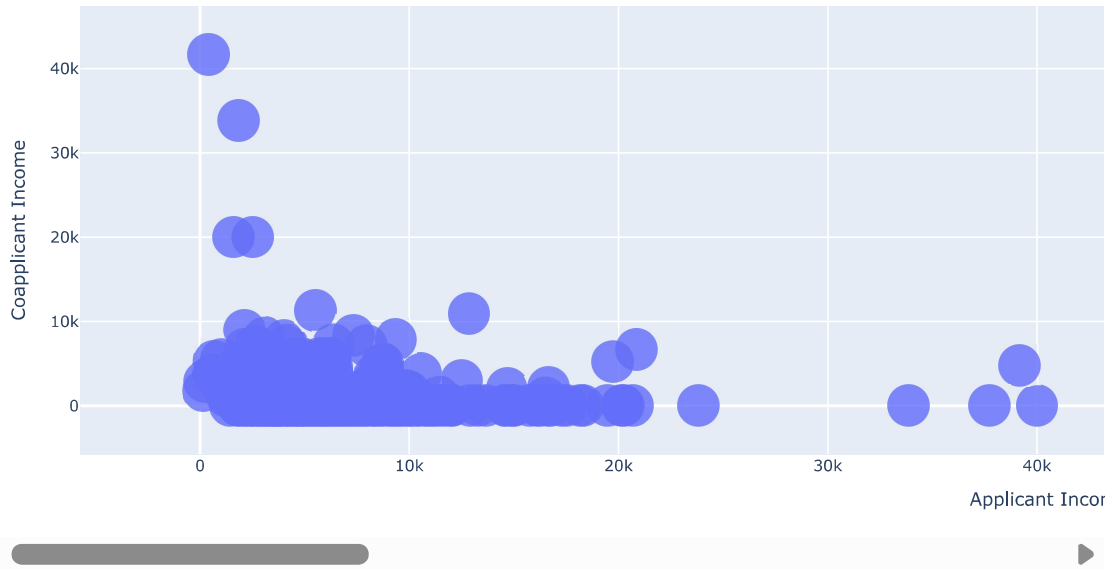
Next steps: Generate code with df | View recommended plots | New interactive sheet

```
import plotly.graph_objects as go

fig = go.Figure(data=go.Scatter(
    x=df['ApplicantIncome'],
    y=df['CoapplicantIncome'],
    mode='markers',
    marker=dict(size=32, opacity=0.8)
))

fig.update_layout(
    xaxis_title='Applicant Income',
    yaxis_title='Coapplicant Income',
    showlegend=False  # Hide legend if not needed
)

fig.show()
```

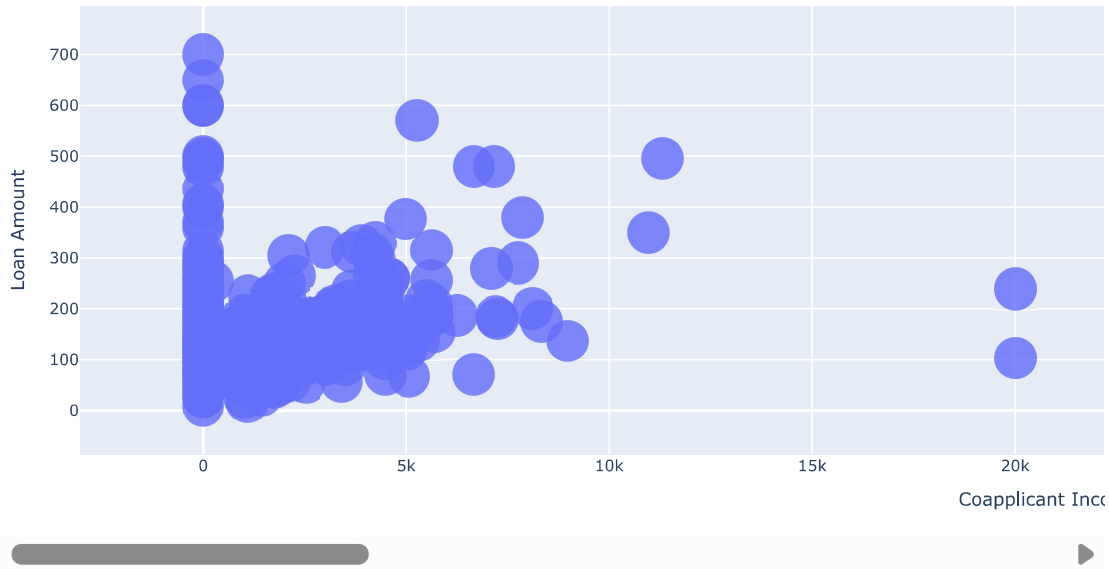



```
import plotly.graph_objects as go

fig = go.Figure(data=go.Scatter(
    x=df['CoapplicantIncome'],
    y=df['LoanAmount'],
    mode='markers',
    marker=dict(size=32, opacity=0.8)
))

fig.update_layout(
    xaxis_title='Coapplicant Income',
    yaxis_title='Loan Amount',
    showlegend=False
)

fig.show()
```

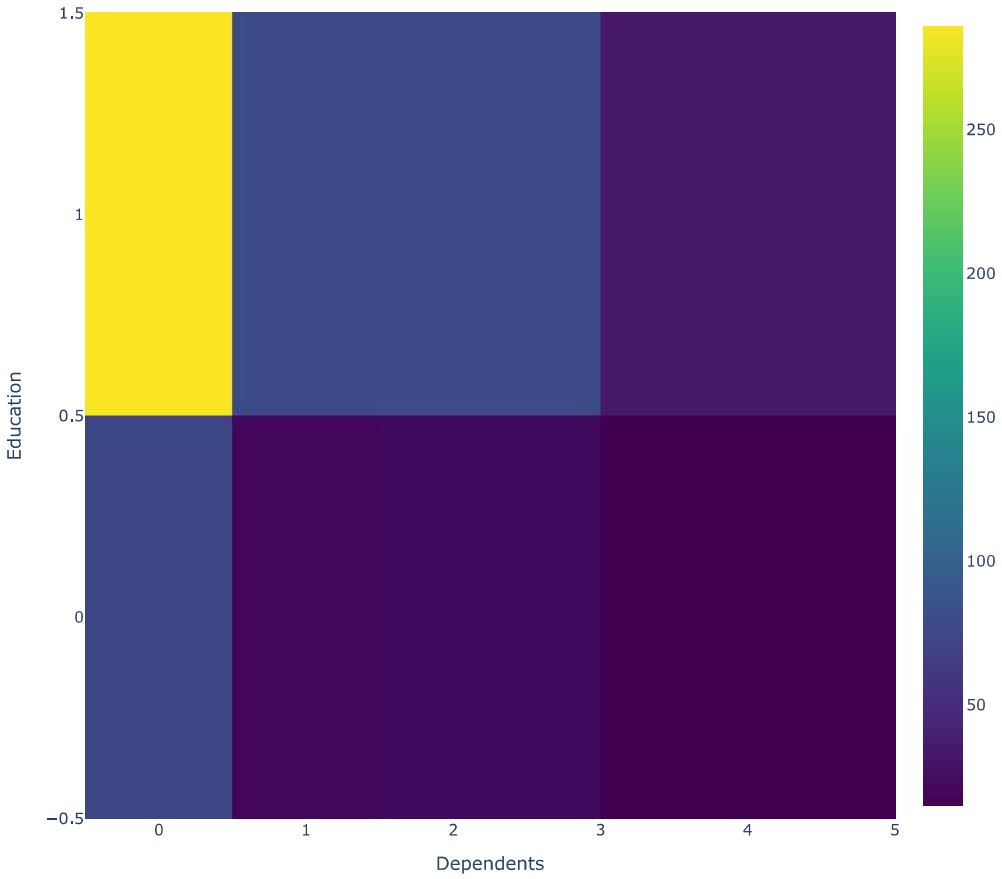



```
df_2dhist = pd.DataFrame({
    x_label: grp['Education'].value_counts()
    for x_label, grp in df.groupby('Dependents')
})

fig = go.Figure(data=go.Heatmap(
                    z=df_2dhist.values,
                    x=df_2dhist.columns,
                    y=df_2dhist.index,
                    colorscale='Viridis'))

fig.update_layout(
    xaxis_title='Dependents',
    yaxis_title='Education',
    width=800, height=800
)

fig.show()
```

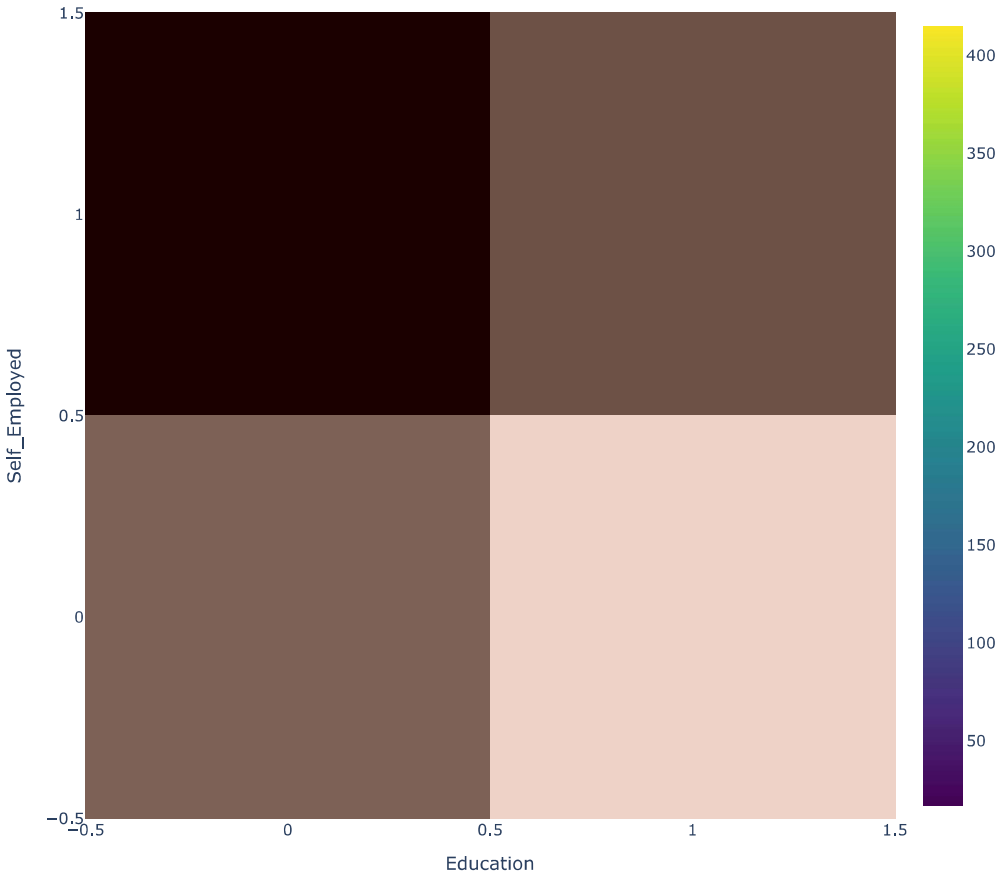



```
df_2dhist = pd.DataFrame({
    x_label: grp['Self_Employed'].value_counts()
    for x_label, grp in df.groupby('Education')
})

fig = go.Figure(data=go.Heatmap(
                    z=df_2dhist.values,
                    x=df_2dhist.columns,
                    y=df_2dhist.index,
                    colorscale='Viridis'))

fig.update_layout(
    xaxis_title='Education',
    yaxis_title='Self_Employed',
    width=800, height=800
)

fig.show()
```

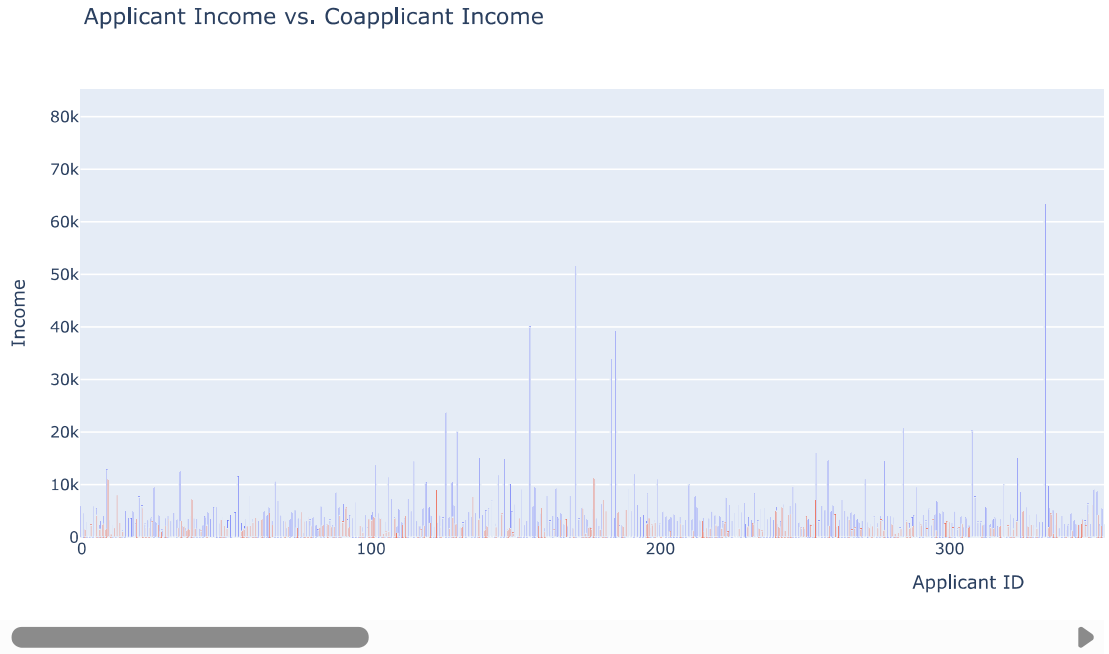



```
import plotly.graph_objects as go

# Create the bar chart using applicantIncome and coapplicantincome
fig = go.Figure(data=[
    go.Bar(name='Applicant Income', x=df.index, y=df['ApplicantIncome']),
    go.Bar(name='Coapplicant Income', x=df.index, y=df['CoapplicantIncome'])
])

# Update the layout for better visualization
fig.update_layout(
    title='Applicant Income vs. Coapplicant Income',
    xaxis_title='Applicant ID',
    yaxis_title='Income',
    barmode='group'  # Group the bars for better comparison
)

# Display the chart
fig.show()
```

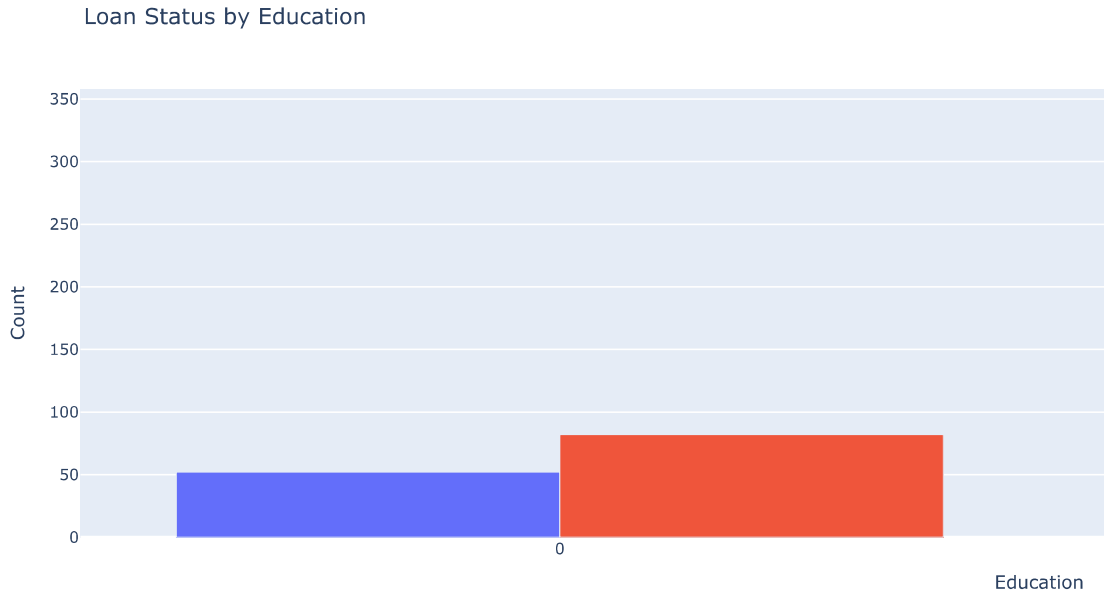



```
education_counts = df.groupby(['Education', 'Loan_Status'])['Loan_Status'].count().reset_index(name='count')

fig = go.Figure(data=[
    go.Bar(name=status, x=group['Education'], y=group['count'])
    for status, group in education_counts.groupby('Loan_Status')
])

fig.update_layout(
    title='Loan Status by Education',
    xaxis_title='Education',
    yaxis_title='Count',
    barmode='group'
)

fig.show()
```




```
#Dependent column values
df['Dependents'].value_counts()
```

Dependents	count
0	360
1	102
2	101
3+	51

```
#replacing the value of 3+ to 4
df=df.replace(to_replace='3+',value=4)

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

Dependents	count
0	360
1	102
2	101
4	51

```
df['Dependents'] = df['Dependents'].astype(int)
```

```
#Splitting the data and label
X=df.drop(columns=['Loan_ID','Loan_Status'],axis=1)
Y=df['Loan_Status']
```

```
print(X)
print(Y)
```

```
     Gender  Married  Dependents  Education  Self_Employed  ApplicantIncome  \
0         1        0           0          1              0             5849
1         1        1           1          1              0             4583
2         1        1           0          1              1             3000
3         1        1           0          0              0             2583
4         1        0           0          1              0             6000
..      ...      ...         ...        ...            ...              ...
609       0        0           0          1              0             2900
610       1        1           4          1              0             4106
611       1        1           1          1              0             8072
612       1        1           2          1              0             7583
613       0        0           0          1              1             4583

     CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
0                  0.0       128.0             360.0             1.0
1               1508.0       128.0             360.0             1.0
2                  0.0        66.0             360.0             1.0
3               2358.0       120.0             360.0             1.0
4                  0.0       141.0             360.0             1.0
..                 ...         ...               ...             ...
609                0.0        71.0             360.0             1.0
610                0.0        40.0             180.0             1.0
611              240.0       253.0             360.0             1.0
612                0.0       187.0             360.0             1.0
613                0.0       133.0             360.0             0.0

     Property_Area
0                1
1                0
2                1
3                1
4                1
..             ...
609              0
610              0
611              1
612              1
613              2

[614 rows x 11 columns]
0      1
1      0
2      1
3      1
4      1
      ..
609    1
610    1
611    1
612    1
613    0
Name: Loan_Status, Length: 614, dtype: int64
```

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.1,random_state=2)
```

```
print(X.shape,X_test.shape,Y.shape,X_test.shape)
```

```
(614, 11) (62, 11) (614,) (62, 11)
```

```
df.corr()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplicar
Loan_ID	1.000000	-0.028029	-0.016013	0.046981	-0.039442	0.032874	0.016925	
Gender	-0.028029	1.000000	0.364569	0.165877	-0.045364	-0.000525	0.058809	
Married	-0.016013	0.364569	1.000000	0.308408	-0.012304	0.004489	0.051708	
Dependents	0.046981	0.165877	0.308408	1.000000	-0.059001	0.048028	0.134080	
Education	-0.039442	-0.045364	-0.012304	-0.059001	1.000000	0.010383	0.140760	
Self_Employed	0.032874	-0.000525	0.004489	0.048028	0.010383	1.000000	0.127180	-0
ApplicantIncome	0.016925	0.058809	0.051708	0.134080	0.140760	0.127180	1.000000	-0
CoapplicantIncome	0.039211	0.082912	0.075948	0.034780	0.062290	-0.016100	-0.116605	
LoanAmount	0.036872	0.106904	0.146546	0.170584	0.168759	0.115100	0.565181	
Loan_Amount_Term	-0.033028	-0.074030	-0.100912	-0.104059	0.073928	-0.033739	-0.046531	-0
Credit_History	-0.030603	0.009170	0.010938	-0.047203	0.073658	-0.001550	-0.018615	
Property_Area	-0.078944	-0.109521	0.007281	-0.002768	0.066740	-0.007124	-0.017321	-0

```
df['Dependents'] = df['Dependents'].astype(int)
```

```
import plotly.graph_objects as go
import plotly.express as px

gender_means = df.groupby("Gender").mean().reset_index()

# Create a list of columns to plot
columns_to_plot = ['Married', 'Dependents', 'Education', 'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'Lo

# Create subplots for each column
fig = px.bar(gender_means, x="Gender", y=columns_to_plot, barmode='group', title="According to Gender",
             facet_col_wrap=4, facet_col="variable", height=800)  # Adjust height as needed

# Update layout for better readability
fig.update_yaxes(matches=None)  # Allow independent y-axis scaling for each subplot
fig.for_each_annotation(lambda a: a.update(text=a.text.split("=")[-1]))  # Simplify subplot titles
fig.update_layout(title_text="According to Gender", title_x=0.5)  # Center the main title

fig.show()
```

According to Gender