In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('loan_prediction.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Terr
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.

In [4]:
```
df.tail()
```

Out[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_T
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	36
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	18
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	36
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	36
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	36

In [5]:
```
df.dtypes
```

Out[5]:
```
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           object
dtype: object
```

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

Out[6]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
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
dtype: int64
```

In [8]:
```
df.shape
```

Out[8]:
```
(614, 13)
```

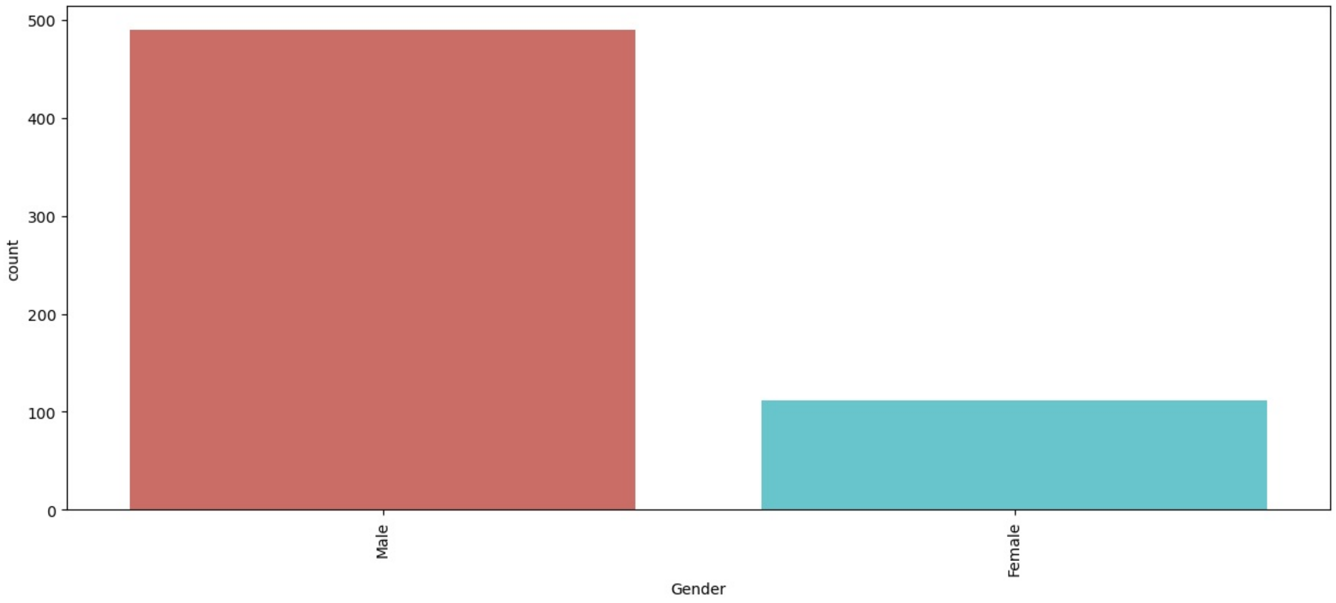
In [9]:
```
df['Gender'].unique()
```

Out[9]:
```
array(['Male', 'Female', nan], dtype=object)
```

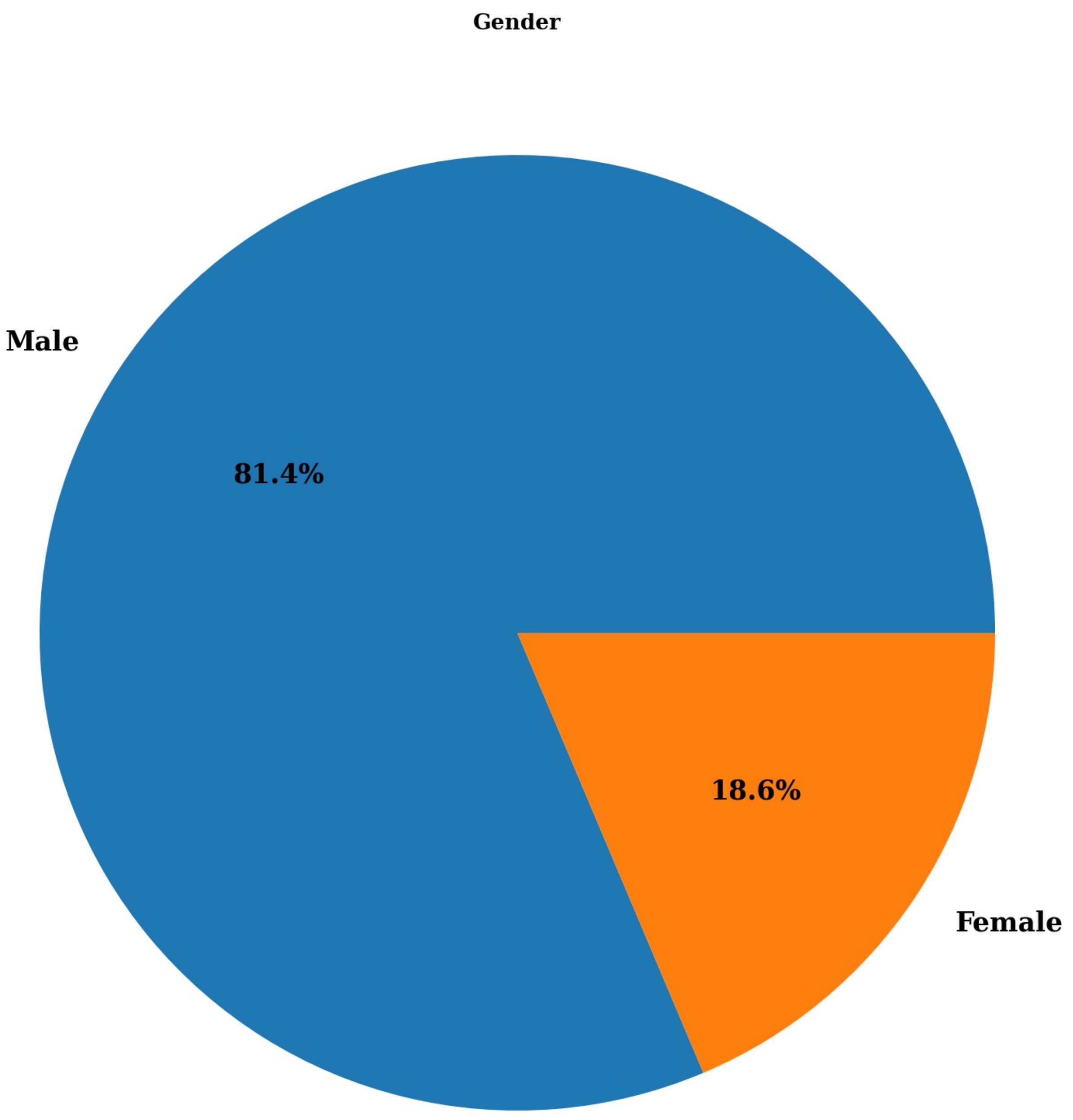
In [10]:
```
df['Gender'].value_counts()
```

Out[10]:
```
Gender
Male      489
Female    112
Name: count, dtype: int64
```

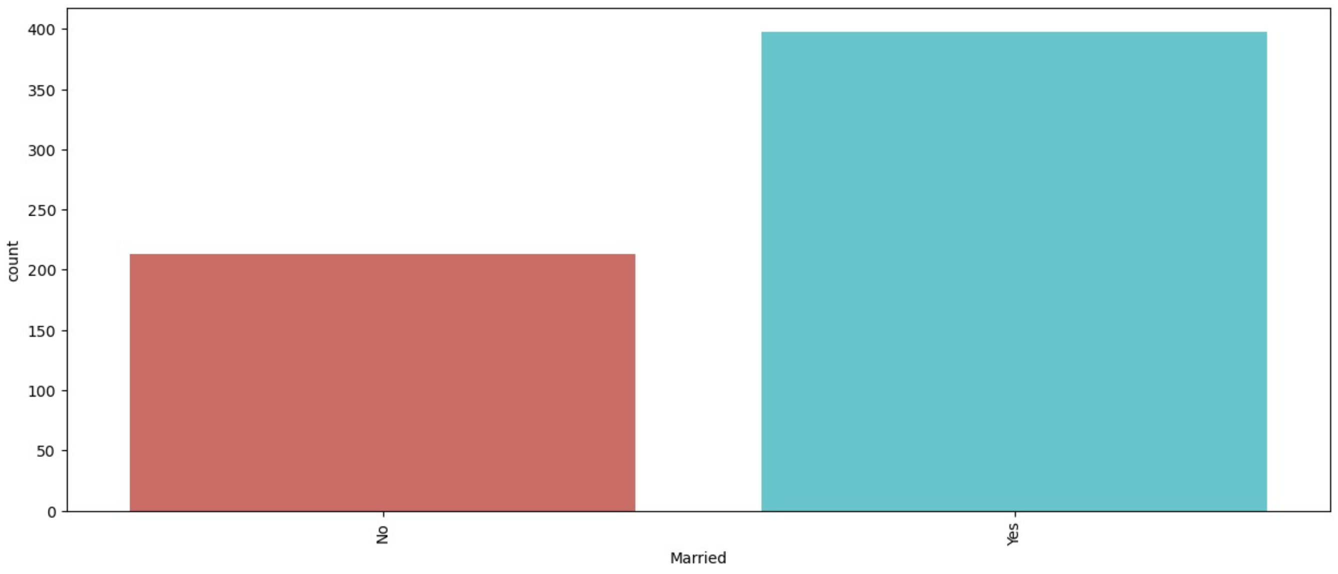
In [11]:
```
plt.figure(figsize=(15,6))
sns.countplot(x =df['Gender'], data = df, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



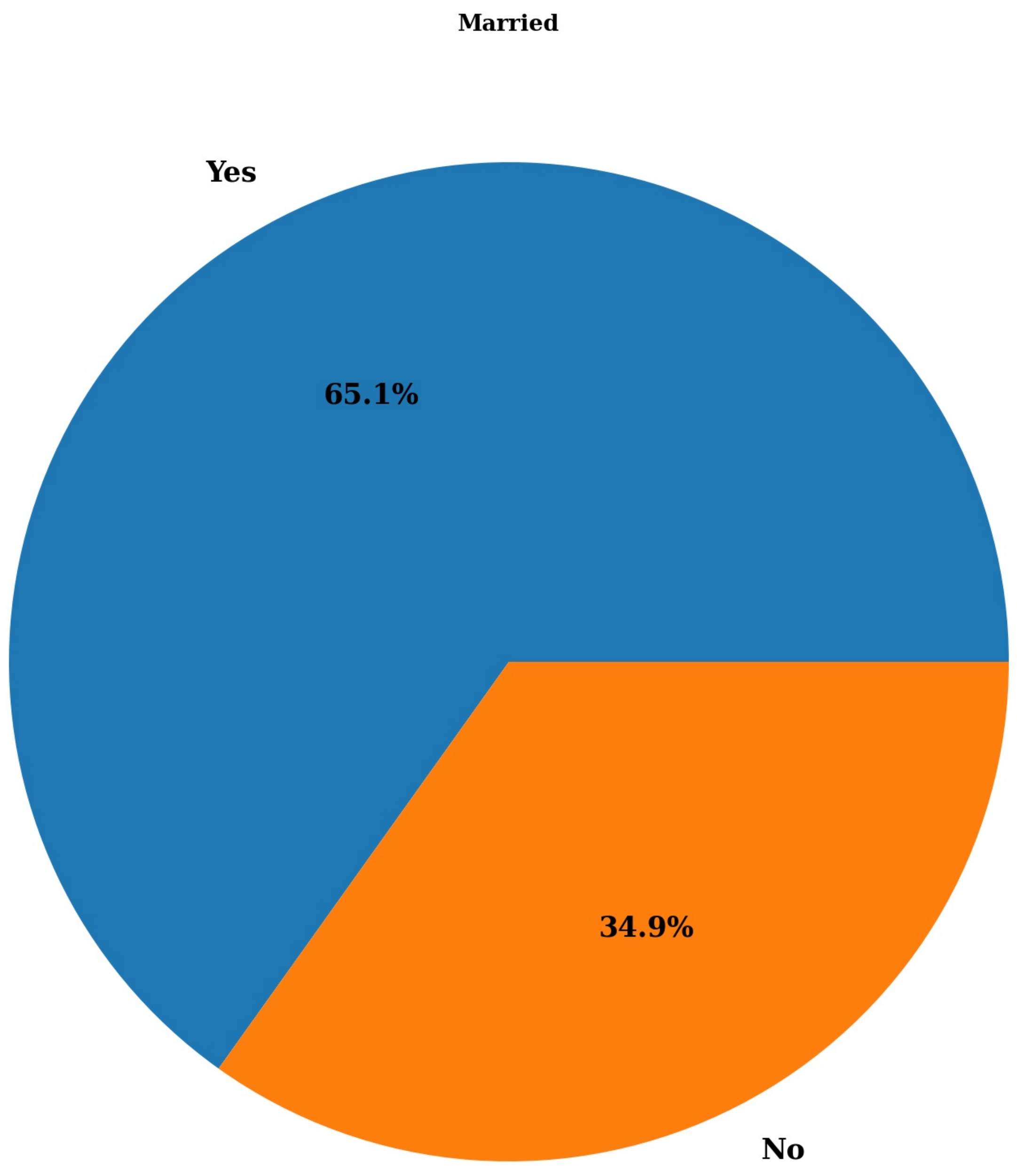
In [12]:
```
plt.figure(figsize=(30,20))
plt.pie(df['Gender'].value_counts(), labels=df['Gender'].value_counts().index, autopct='%1.1f%%', textprops={ '
                                            'color': 'black',
                                            'weight': 'bold',
                                            'family': 'serif' })
hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('Gender', size=20, **hfont)
plt.show()
```



In [13]:
```
df.nunique()
```
Out[13]:
```
Loan_ID              614
Gender                 2
Married                2
Dependents             4
Education              2
Self_Employed          2
ApplicantIncome      505
CoapplicantIncome    287
LoanAmount           203
Loan_Amount_Term      10
Credit_History         2
Property_Area          3
Loan_Status            2
dtype: int64
```
In [14]:
```
df['Married'].unique()
```
Out[14]:
```
array(['No', 'Yes', nan], dtype=object)
```
In [15]:
```
df['Married'].value_counts()
```
Out[15]:
```
Married
Yes    398
No     213
Name: count, dtype: int64
```
In [16]:
```
plt.figure(figsize=(15,6))
sns.countplot(x =df['Married'], data = df, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [17]:
```
plt.figure(figsize=(30,20))
plt.pie(df['Married'].value_counts(), labels=df['Married'].value_counts().index, autopct='%1.1f%%', textprops={
                                           'color': 'black',
                                           'weight': 'bold',
                                           'family': 'serif' })
hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('Married', size=20, **hfont)
plt.show()
```



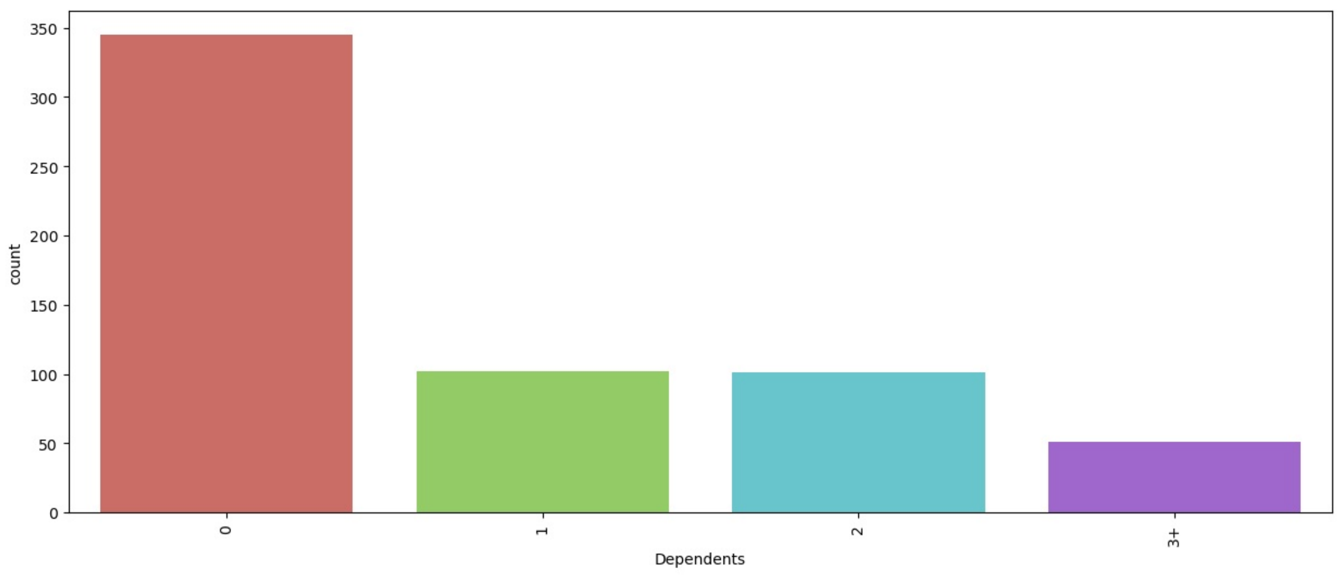
```
In [18]: df['Dependents'].unique()
```

```
Out[18]: array(['0', '1', '2', '3+', nan], dtype=object)
```

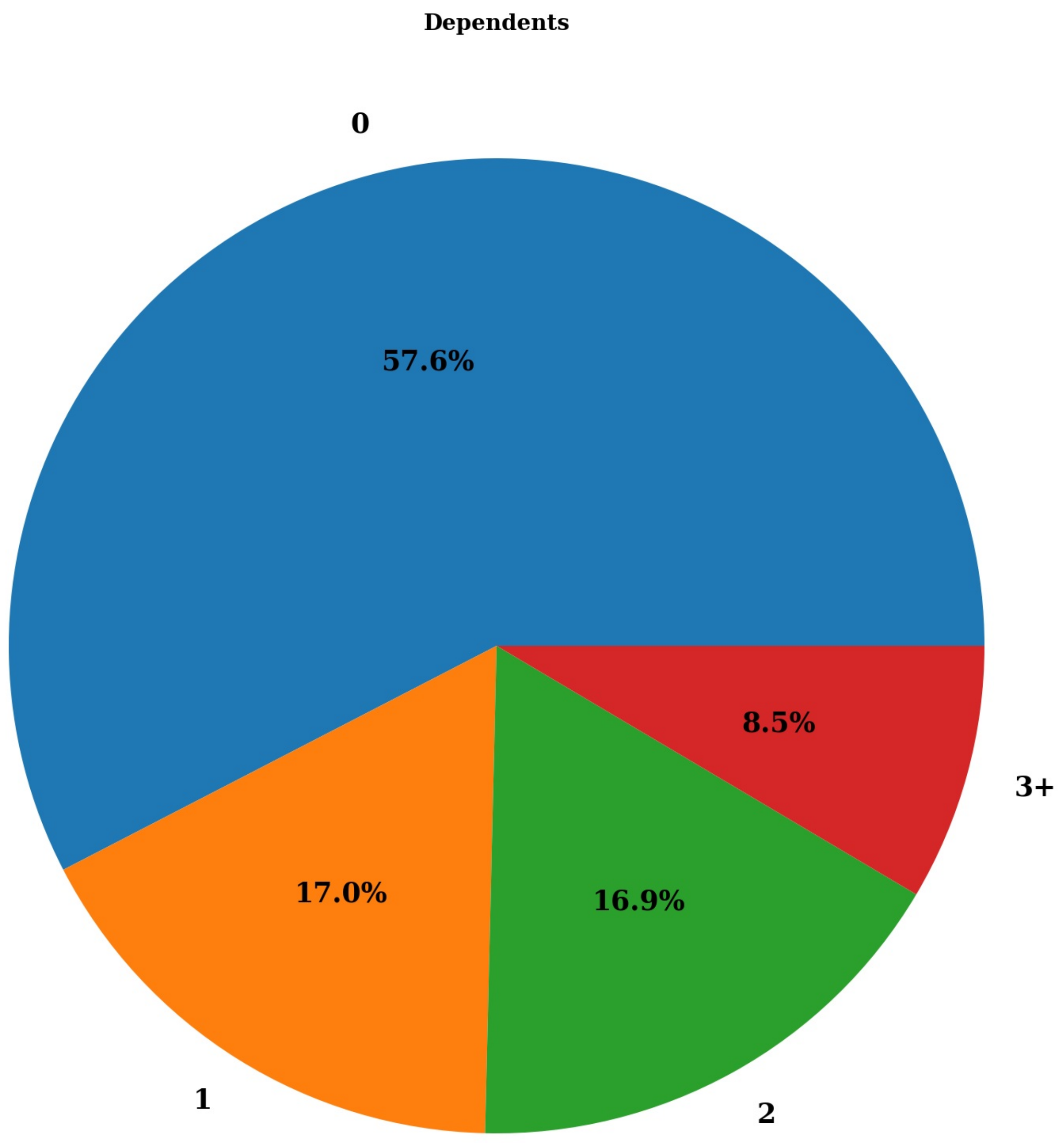
```
In [19]: df['Dependents'].value_counts()
```

```
Out[19]: Dependents
0     345
1     102
2     101
3+     51
Name: count, dtype: int64
```

```
In [20]: plt.figure(figsize=(15,6))
sns.countplot(x=df['Dependents'],data=df,palette='hls')
plt.xticks(rotation = 90)
plt.show()
```

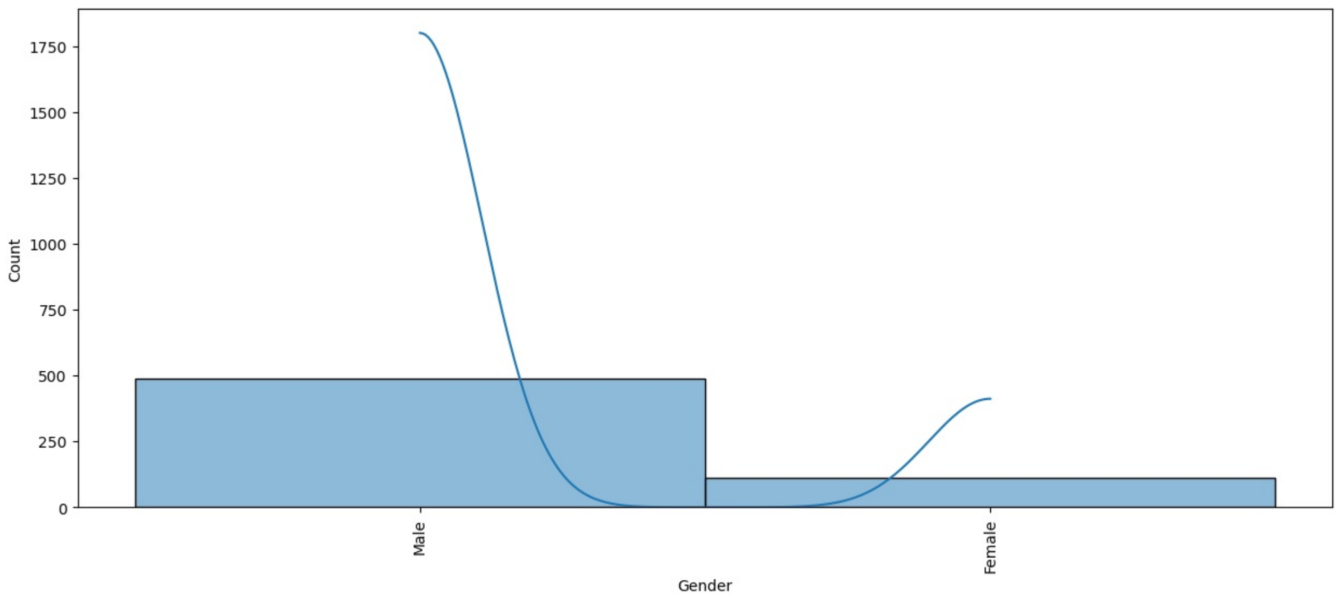


```
In [21]: plt.figure(figsize=(30,20))
         plt.pie(df['Dependents'].value_counts(), labels=df['Dependents'].value_counts().index, autopct='%1.1f%%', textp
                                                     'color': 'black',
                                                     'weight': 'bold',
                                                     'family': 'serif' })
         hfont = {'fontname':'serif', 'weight': 'bold'}
         plt.title('Dependents', size=20, **hfont)
         plt.show()
```



```
In [22]: plt.figure(figsize=(15,6))
         sns.histplot(df['Gender'], kde = True, bins = 20, palette = 'hls')
         plt.xticks(rotation = 90)
         plt.show()
```

```
C:\Users\Acer\AppData\Local\Temp\ipykernel_23412\2498430458.py:2: UserWarning: Ignoring `palette` because no `h
ue` variable has been assigned.
  sns.histplot(df['Gender'], kde = True, bins = 20, palette = 'hls')
```



In [23]:
```
# Fill missing values in categorical columns with mode
df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)
df['Married'].fillna(df['Married'].mode()[0], inplace=True)
df['Dependents'].fillna(df['Dependents'].mode()[0], inplace=True)
df['Self_Employed'].fillna(df['Self_Employed'].mode()[0], inplace=True)
```

In [25]:
```
df = df.drop('Loan_ID', axis=1)
```

In [26]:
```
df
```

Out[26]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit
0	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	
...	...	...	...	...	...	...	...	...	...	
609	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	
610	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	
611	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	
612	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	
613	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	

614 rows × 12 columns

In [34]:
```
df.head()
```

Out[34]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_H
0	Male	No	0	Graduate	No	5849	0.0	128.0	360.0	
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	

In [35]:
```
# Fill missing values in LoanAmount with the median
df['LoanAmount'].fillna(df['LoanAmount'].median(), inplace=True)

# Fill missing values in Loan_Amount_Term with the mode
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mode()[0], inplace=True)

# Fill missing values in Credit_History with the mode
df['Credit_History'].fillna(df['Credit_History'].mode()[0], inplace=True)
```

In [36]:
```
import plotly.express as px

loan_status_count = df['Loan_Status'].value_counts()
fig_loan_status = px.pie(loan_status_count,
                         names=loan_status_count.index,
                         title='Loan Approval Status')
fig_loan_status.show()
```

In [37]:
```
gender_count = df['Gender'].value_counts()
fig_gender = px.bar(gender_count,
                    x=gender_count.index,
                    y=gender_count.values,
                    title='Gender Distribution')
fig_gender.show()
```

In [38]:
```
education_count = df['Education'].value_counts()
fig_education = px.bar(education_count,
                       x=education_count.index,
                       y=education_count.values,
                       title='Education Distribution')
fig_education.show()
```

In [39]:
```
self_employed_count = df['Self_Employed'].value_counts()
fig_self_employed = px.bar(self_employed_count,
                           x=self_employed_count.index,
                           y=self_employed_count.values,
                           title='Self-Employment Distribution')
fig_self_employed.show()
```

In [40]:
```python
fig_applicant_income = px.histogram(df, x='ApplicantIncome',
                                    title='Applicant Income Distribution')
fig_applicant_income.show()
```

In [41]:
```python
fig_income = px.box(df, x='Loan_Status',
                    y='ApplicantIncome',
                    color="Loan_Status",
                    title='Loan_Status vs ApplicantIncome')
fig_income.show()
```

In [42]:
```
# Calculate the IQR
Q1 = df['ApplicantIncome'].quantile(0.25)
Q3 = df['ApplicantIncome'].quantile(0.75)
IQR = Q3 - Q1

# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Remove outliers
df = df[(df['ApplicantIncome'] >= lower_bound) & (df['ApplicantIncome'] <= upper_bound)]
```

In [43]:
```
fig_coapplicant_income = px.box(df,
                                x='Loan_Status',
                                y='CoapplicantIncome',
                                color="Loan_Status",
                                title='Loan_Status vs CoapplicantIncome')
fig_coapplicant_income.show()
```

In [44]:
```
# Calculate the IQR
```

```
Q1 = df['CoapplicantIncome'].quantile(0.25)
Q3 = df['CoapplicantIncome'].quantile(0.75)
IQR = Q3 - Q1

# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Remove outliers
df = df[(df['CoapplicantIncome'] >= lower_bound) & (df['CoapplicantIncome'] <= upper_bound)]
```

In [45]:
```
fig_loan_amount = px.box(df, x='Loan_Status',
                         y='LoanAmount',
                         color="Loan_Status",
                         title='Loan_Status vs LoanAmount')
fig_loan_amount.show()
```

In [46]:
```
fig_credit_history = px.histogram(df, x='Credit_History', color='Loan_Status',
                                  barmode='group',
                                  title='Loan_Status vs Credit_His')
fig_credit_history.show()
```

In [47]:
```
fig_property_area = px.histogram(df, x='Property_Area', color='Loan_Status',
                                  barmode='group',
                                 title='Loan_Status vs Property_Area')
fig_property_area.show()
```

In [48]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
```

In [49]:
```
# Convert categorical columns to numerical using one-hot encoding
cat_cols = ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed', 'Property_Area']
df = pd.get_dummies(df, columns=cat_cols)

# Split the dataset into features (X) and target (y)
X = df.drop('Loan_Status', axis=1)
y = df['Loan_Status']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Scale the numerical columns using StandardScaler
scaler = StandardScaler()
numerical_cols = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term', 'Credit_History']
X_train[numerical_cols] = scaler.fit_transform(X_train[numerical_cols])
X_test[numerical_cols] = scaler.transform(X_test[numerical_cols])

from sklearn.svm import SVC
model = SVC(random_state=42)
model.fit(X_train, y_train)
```

Out[49]:

```
▾         SVC
SVC(random_state=42)
```

In [50]:

```
y_pred = model.predict(X_test)
print(y_pred)
```

```
['Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y'
 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'N' 'Y' 'Y'
 'Y' 'Y']
```

In [51]:

```
# Convert X_test to a DataFrame
X_test_df = pd.DataFrame(X_test, columns=X_test.columns)

# Add the predicted values to X_test_df
X_test_df['Loan_Status_Predicted'] = y_pred
print(X_test_df.head())
```

```
     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
277        -0.544528          -0.037922   -0.983772          0.305159   
84         -0.067325          -0.931554   -1.571353         -1.430680   
275        -0.734870           0.334654   -0.298262          0.305159   
392        -0.824919           0.522317   -0.200332          0.305159   
537        -0.267373          -0.931554   -0.454950          0.305159   

     Credit_History  Gender_Female  Gender_Male  Married_No  Married_Yes  \
277        0.402248          False         True       False         True   
84         0.402248          False         True       False         True   
275        0.402248          False         True       False         True   
392        0.402248          False         True       False         True   
537        0.402248          False         True        True        False   

     Dependents_0  ...  Dependents_2  Dependents_3+  Education_Graduate  \
277          True  ...         False          False                True   
84          False  ...         False          False                True   
275         False  ...         False          False                True   
392          True  ...         False          False                True   
537         False  ...          True          False                True   

     Education_Not Graduate  Self_Employed_No  Self_Employed_Yes  \
277                   False              True              False   
84                    False              True              False   
275                   False              True              False   
392                   False              True              False   
537                   False              True              False   

     Property_Area_Rural  Property_Area_Semiurban  Property_Area_Urban  \
277                False                    False                 True   
84                 False                    False                 True   
275                False                     True                False   
392                False                    False                 True   
537                False                     True                False   

     Loan_Status_Predicted  
277                      Y  
84                       Y  
275                      Y  
392                      Y  
537                      Y  

[5 rows x 21 columns]
```

In [ ]:

In [ ]: