Credit Score Data Analysis and Credit Score Classification

Importing Libraries & Dataset

In [58]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
pio.templates.default = "plotly_white"
```

Data Reading

In [63]:

```
data = pd.read_csv(r"C:\Users\USER\Downloads\Datasets\Credit Score Data.csv")
data.head()
```

Out[63]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income
0	5634	3392	1	Aaron Maashoh	23.0	821000265.0	Scientist	19114.12
1	5635	3392	2	Aaron Maashoh	23.0	821000265.0	Scientist	19114.12
2	5636	3392	3	Aaron Maashoh	23.0	821000265.0	Scientist	19114.12
3	5637	3392	4	Aaron Maashoh	23.0	821000265.0	Scientist	19114.12
4	5638	3392	5	Aaron Maashoh	23.0	821000265.0	Scientist	19114.12

5 rows × 28 columns

In [35]:

```
data.shape
```

Out[35]:

```
(100000, 28)
```

In [10]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 28 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   ID                        100000 non-null  int64  
 1   Customer_ID               100000 non-null  int64  
 2   Month                     100000 non-null  int64  
 3   Name                      100000 non-null  object 
 4   Age                       100000 non-null  float64
 5   SSN                       100000 non-null  float64
 6   Occupation                100000 non-null  object 
 7   Annual_Income             100000 non-null  float64
 8   Monthly_Inhand_Salary     100000 non-null  float64
 9   Num_Bank_Accounts         100000 non-null  float64
 10  Num_Credit_Card           100000 non-null  float64
 11  Interest_Rate             100000 non-null  float64
 12  Num_of_Loan               100000 non-null  float64
 13  Type_of_Loan              100000 non-null  object 
 14  Delay_from_due_date       100000 non-null  float64
 15  Num_of_Delayed_Payment    100000 non-null  float64
 16  Changed_Credit_Limit      100000 non-null  float64
 17  Num_Credit_Inquiries      100000 non-null  float64
 18  Credit_Mix                100000 non-null  object 
 19  Outstanding_Debt          100000 non-null  float64
 20  Credit_Utilization_Ratio  100000 non-null  float64
 21  Credit_History_Age        100000 non-null  float64
 22  Payment_of_Min_Amount     100000 non-null  object 
 23  Total_EMI_per_month       100000 non-null  float64
 24  Amount_invested_monthly   100000 non-null  float64
 25  Payment_Behaviour         100000 non-null  object 
 26  Monthly_Balance           100000 non-null  float64
 27  Credit_Score              100000 non-null  object 
dtypes: float64(18), int64(3), object(7)
memory usage: 21.4+ MB
```

```
In [11]: data.isnull().sum()
```

Out[11]:

```
ID                          0
Customer_ID                 0
Month                       0
Name                        0
Age                         0
SSN                         0
Occupation                  0
Annual_Income               0
Monthly_Inhand_Salary       0
Num_Bank_Accounts           0
Num_Credit_Card             0
Interest_Rate               0
Num_of_Loan                 0
Type_of_Loan                0
Delay_from_due_date         0
Num_of_Delayed_Payment      0
Changed_Credit_Limit        0
Num_Credit_Inquiries        0
Credit_Mix                  0
Outstanding_Debt            0
Credit_Utilization_Ratio    0
Credit_History_Age          0
Payment_of_Min_Amount       0
Total_EMI_per_month         0
Amount_invested_monthly     0
Payment_Behaviour           0
Monthly_Balance             0
Credit_Score                0
dtype: int64
```

In [12]:

```
data.describe()
```

Out[12]:

	ID	Customer_ID	Month	Age	SSN	Annu
count	100000.000000	100000.000000	100000.000000	100000.000000	1.000000e+05	100
mean	80631.500000	25982.666640	4.500000	33.316340	5.004617e+08	50
std	43301.486619	14340.543051	2.291299	10.764812	2.908267e+08	38
min	5634.000000	1006.000000	1.000000	14.000000	8.134900e+04	7
25%	43132.750000	13664.500000	2.750000	24.000000	2.451686e+08	19
50%	80631.500000	25777.000000	4.500000	33.000000	5.006886e+08	36
75%	118130.250000	38385.000000	6.250000	42.000000	7.560027e+08	71
max	155629.000000	50999.000000	8.000000	56.000000	9.999934e+08	179

8 rows × 21 columns

In [13]:

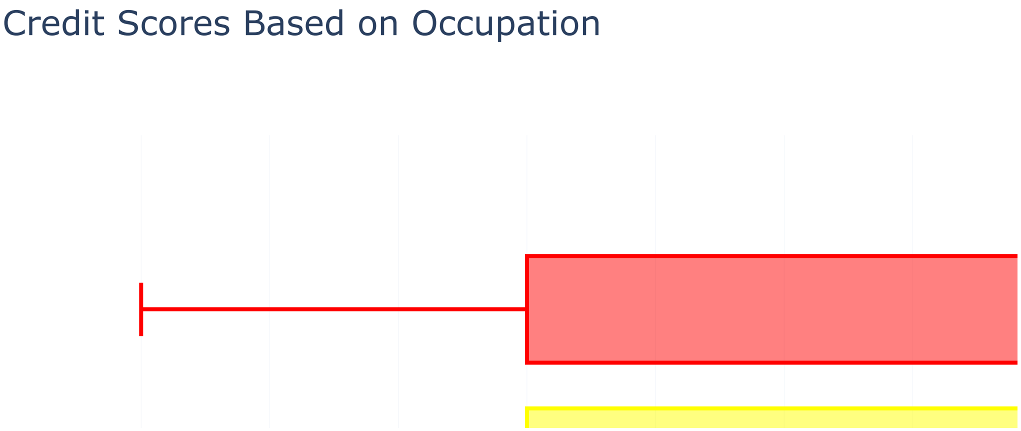
```
data["Credit_Score"].value_counts()
```

Out[13]:

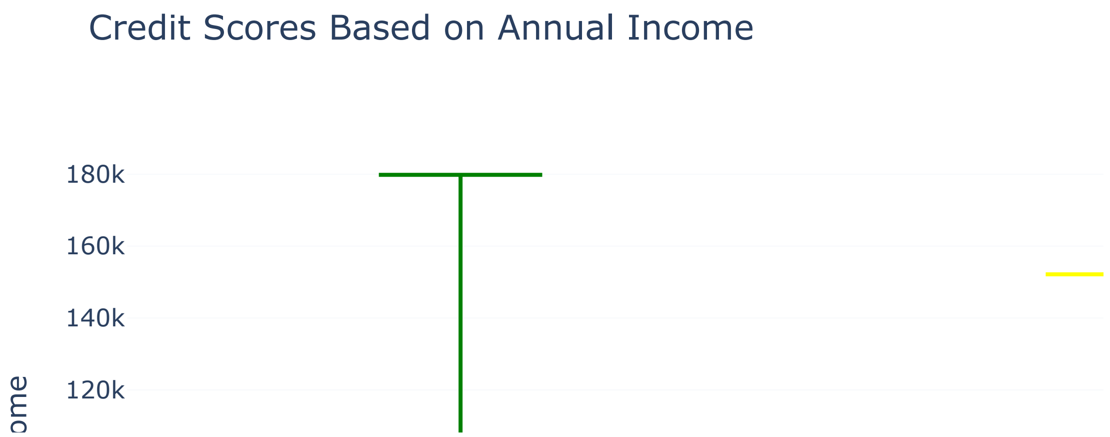
```
Credit_Score
Standard    53174
Poor        28998
Good        17828
Name: count, dtype: int64
```

Data Exploration

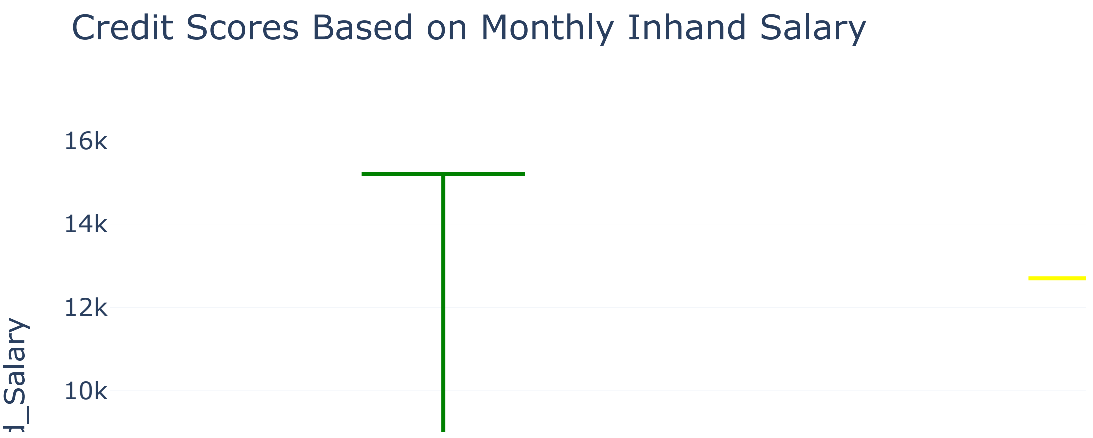
In [15]:
```
fig = px.box(data,
             x="Occupation",
             color="Credit_Score",
             title="Credit Scores Based on Occupation",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.show()
```



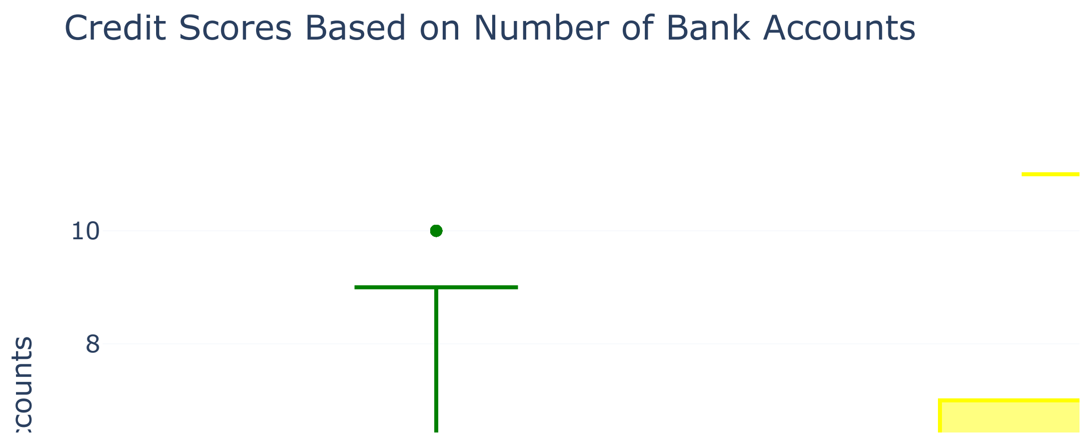
In [16]:
```
fig = px.box(data,
             x="Credit_Score",
             y="Annual_Income",
             color="Credit_Score",
             title="Credit Scores Based on Annual Income",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```



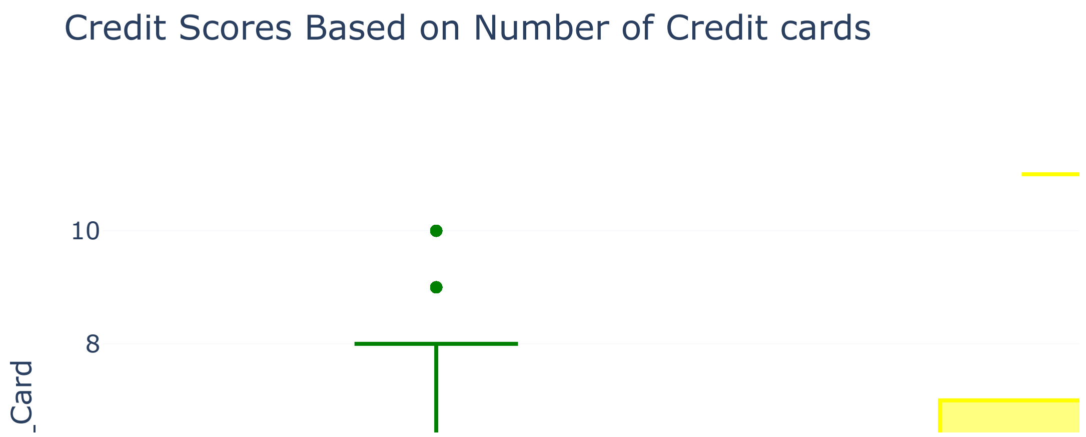
```
In [17]: fig = px.box(data,
                      x="Credit_Score",
                      y="Monthly_Inhand_Salary",
                      color="Credit_Score",
                      title="Credit Scores Based on Monthly Inhand Salary",
                      color_discrete_map={'Poor':'red',
                                          'Standard':'yellow',
                                          'Good':'green'})
         fig.update_traces(quartilemethod="exclusive")
         fig.show()
```



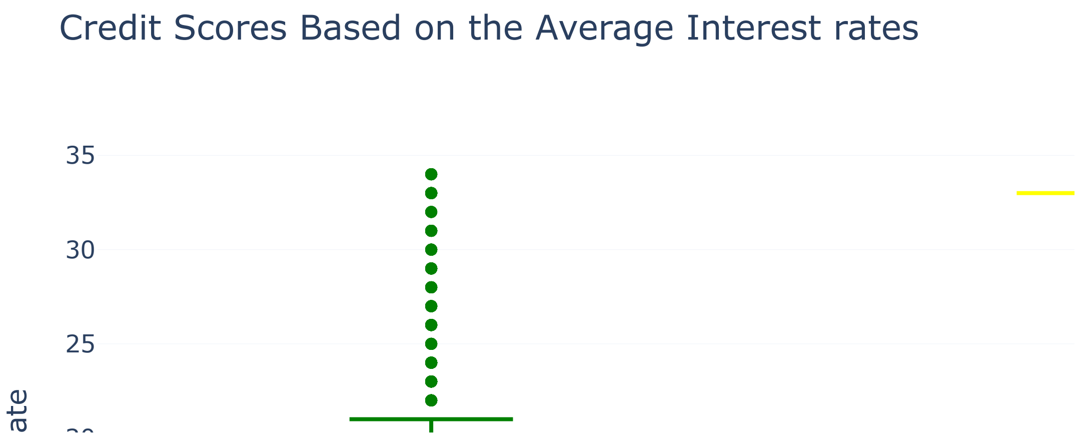
```
In [18]: fig = px.box(data,
                      x="Credit_Score",
                      y="Num_Bank_Accounts",
                      color="Credit_Score",
                      title="Credit Scores Based on Number of Bank Accounts",
                      color_discrete_map={'Poor':'red',
                                          'Standard':'yellow',
                                          'Good':'green'})
         fig.update_traces(quartilemethod="exclusive")
         fig.show()
```



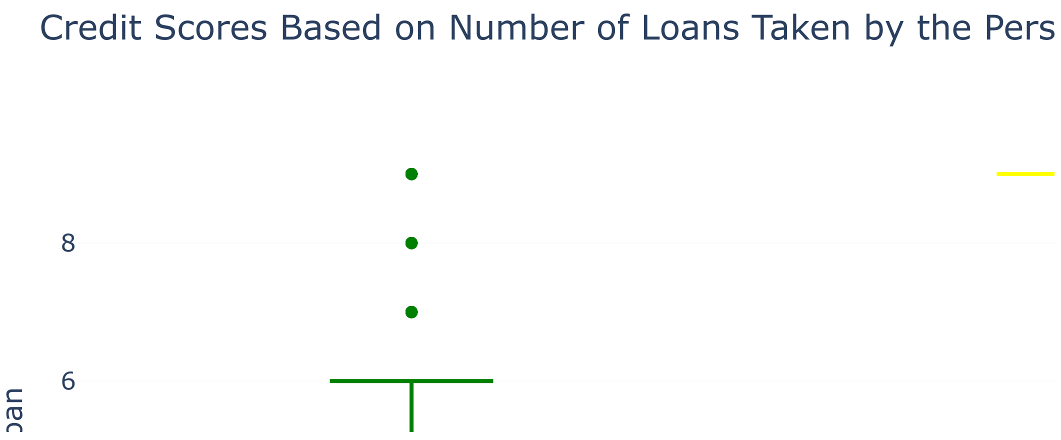
In [19]:
```
fig = px.box(data,
             x="Credit_Score",
             y="Num_Credit_Card",
             color="Credit_Score",
             title="Credit Scores Based on Number of Credit cards",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```



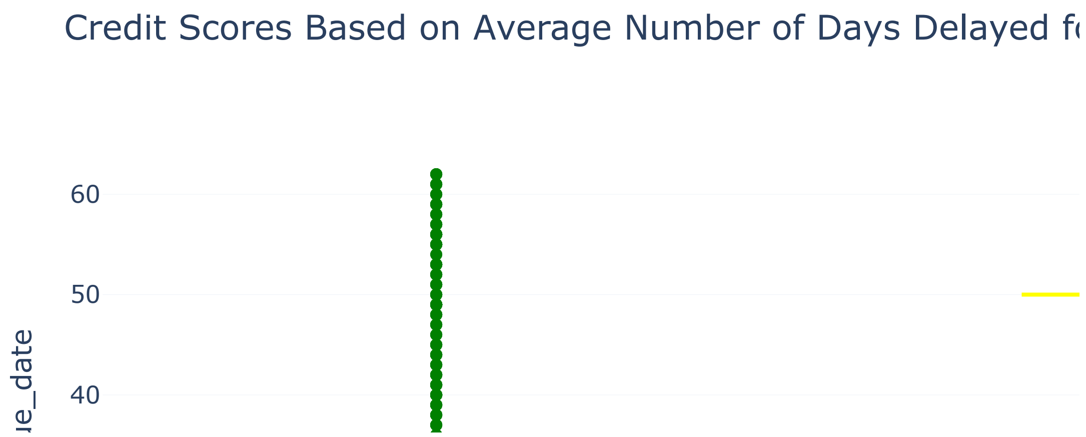
```
In [20]: fig = px.box(data,
                      x="Credit_Score",
                      y="Interest_Rate",
                      color="Credit_Score",
                      title="Credit Scores Based on the Average Interest rates",
                      color_discrete_map={'Poor':'red',
                                          'Standard':'yellow',
                                          'Good':'green'})
         fig.update_traces(quartilemethod="exclusive")
         fig.show()
```



```
In [21]: fig = px.box(data,
                      x="Credit_Score",
                      y="Num_of_Loan",
                      color="Credit_Score",
                      title="Credit Scores Based on Number of Loans Taken by the Person",
                      color_discrete_map={'Poor':'red',
                                          'Standard':'yellow',
                                          'Good':'green'})
         fig.update_traces(quartilemethod="exclusive")
         fig.show()
```

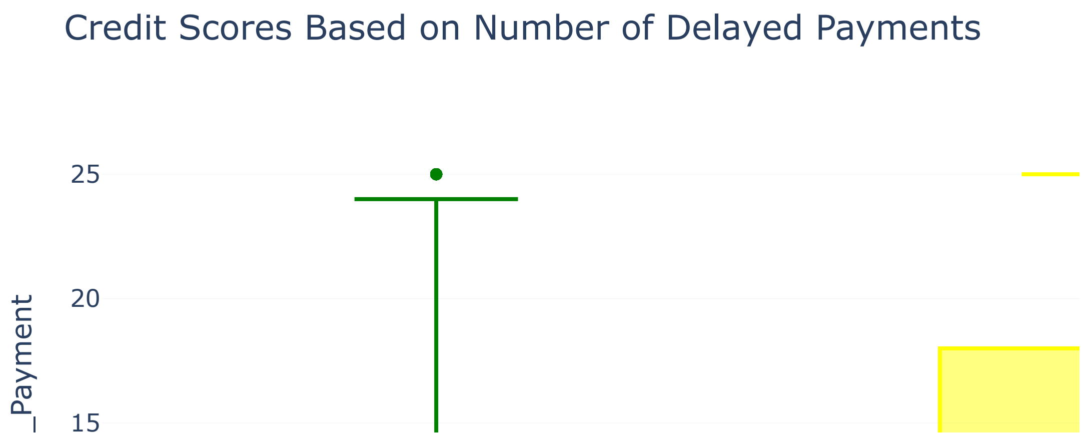


In [22]:
```
fig = px.box(data,
             x="Credit_Score",
             y="Delay_from_due_date",
             color="Credit_Score",
             title="Credit Scores Based on Average Number of Days Delayed for Cr
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```

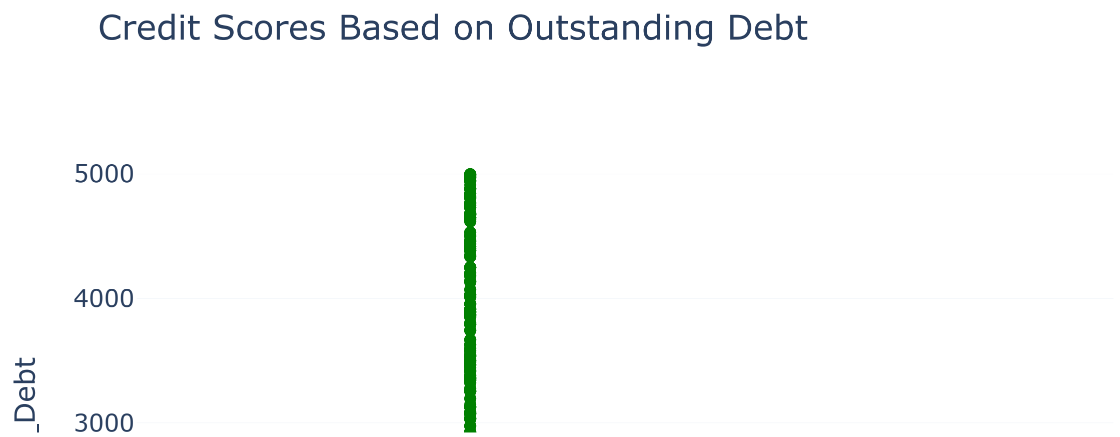


In [23]:

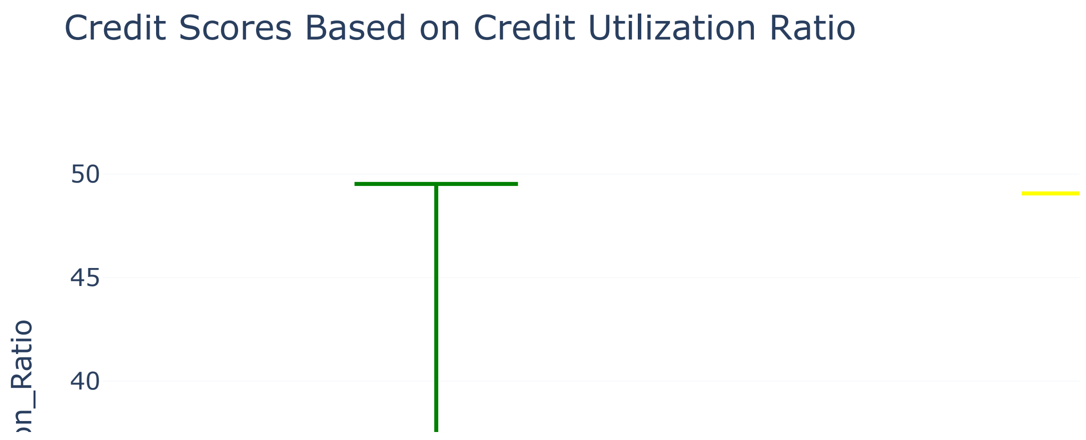
```
fig = px.box(data,
             x="Credit_Score",
             y="Num_of_Delayed_Payment",
             color="Credit_Score",
             title="Credit Scores Based on Number of Delayed Payments",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```



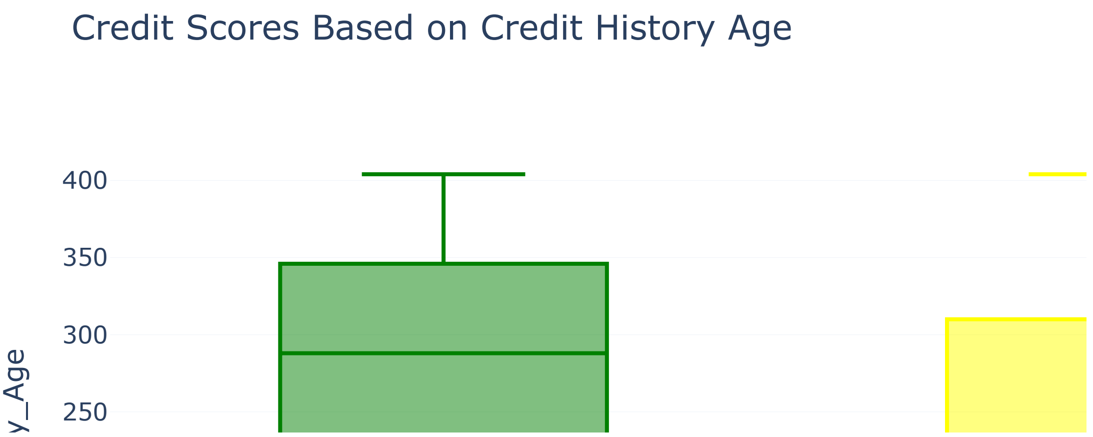
```
In [24]: fig = px.box(data,
                      x="Credit_Score",
                      y="Outstanding_Debt",
                      color="Credit_Score",
                      title="Credit Scores Based on Outstanding Debt",
                      color_discrete_map={'Poor':'red',
                                          'Standard':'yellow',
                                          'Good':'green'})
         fig.update_traces(quartilemethod="exclusive")
         fig.show()
```



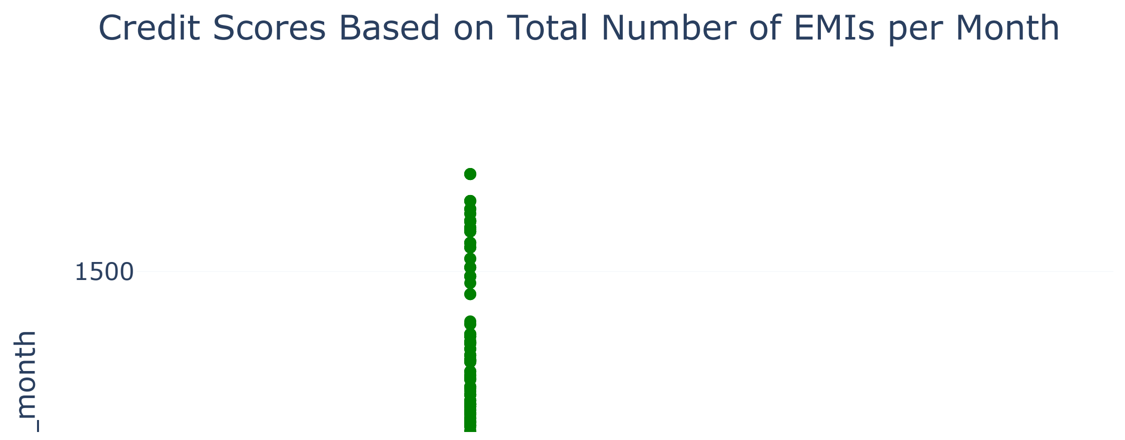
```
In [25]: fig = px.box(data,
                      x="Credit_Score",
                      y="Credit_Utilization_Ratio",
                      color="Credit_Score",
                      title="Credit Scores Based on Credit Utilization Ratio",
                      color_discrete_map={'Poor':'red',
                                          'Standard':'yellow',
                                          'Good':'green'})
         fig.update_traces(quartilemethod="exclusive")
         fig.show()
```



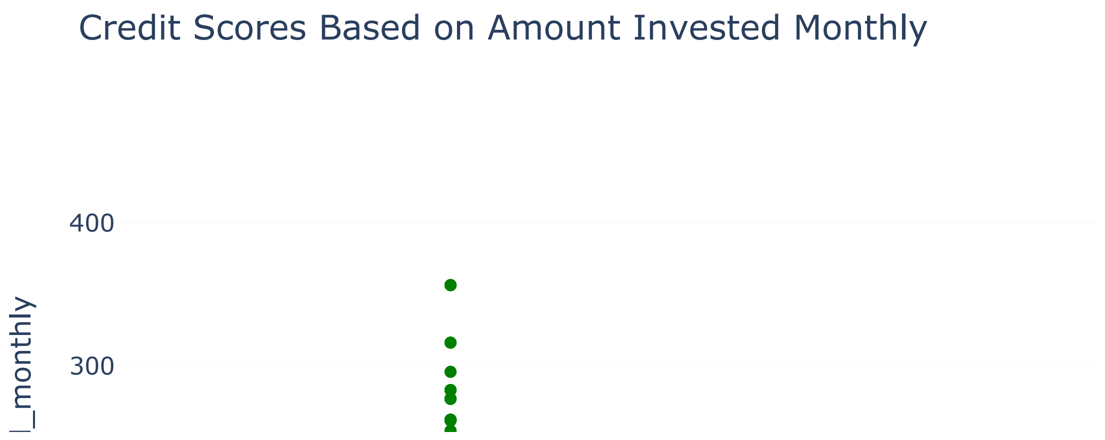
```
In [26]: fig = px.box(data,
                      x="Credit_Score",
                      y="Credit_History_Age",
                      color="Credit_Score",
                      title="Credit Scores Based on Credit History Age",
                      color_discrete_map={'Poor':'red',
                                          'Standard':'yellow',
                                          'Good':'green'})
         fig.update_traces(quartilemethod="exclusive")
         fig.show()
```



In [27]:
```
fig = px.box(data,
             x="Credit_Score",
             y="Total_EMI_per_month",
             color="Credit_Score",
             title="Credit Scores Based on Total Number of EMIs per Month",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```

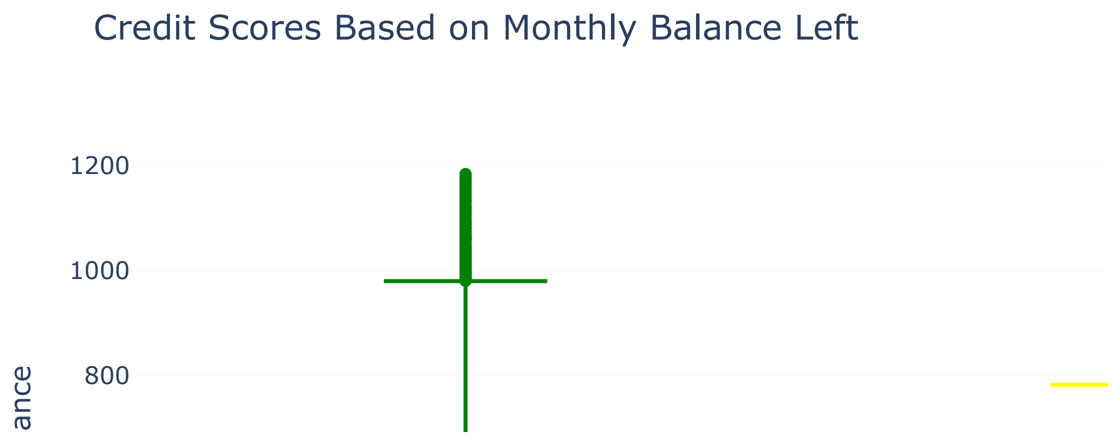


```
In [28]: fig = px.box(data,
                      x="Credit_Score",
                      y="Amount_invested_monthly",
                      color="Credit_Score",
                      title="Credit Scores Based on Amount Invested Monthly",
                      color_discrete_map={'Poor':'red',
                                          'Standard':'yellow',
                                          'Good':'green'})
         fig.update_traces(quartilemethod="exclusive")
         fig.show()
```



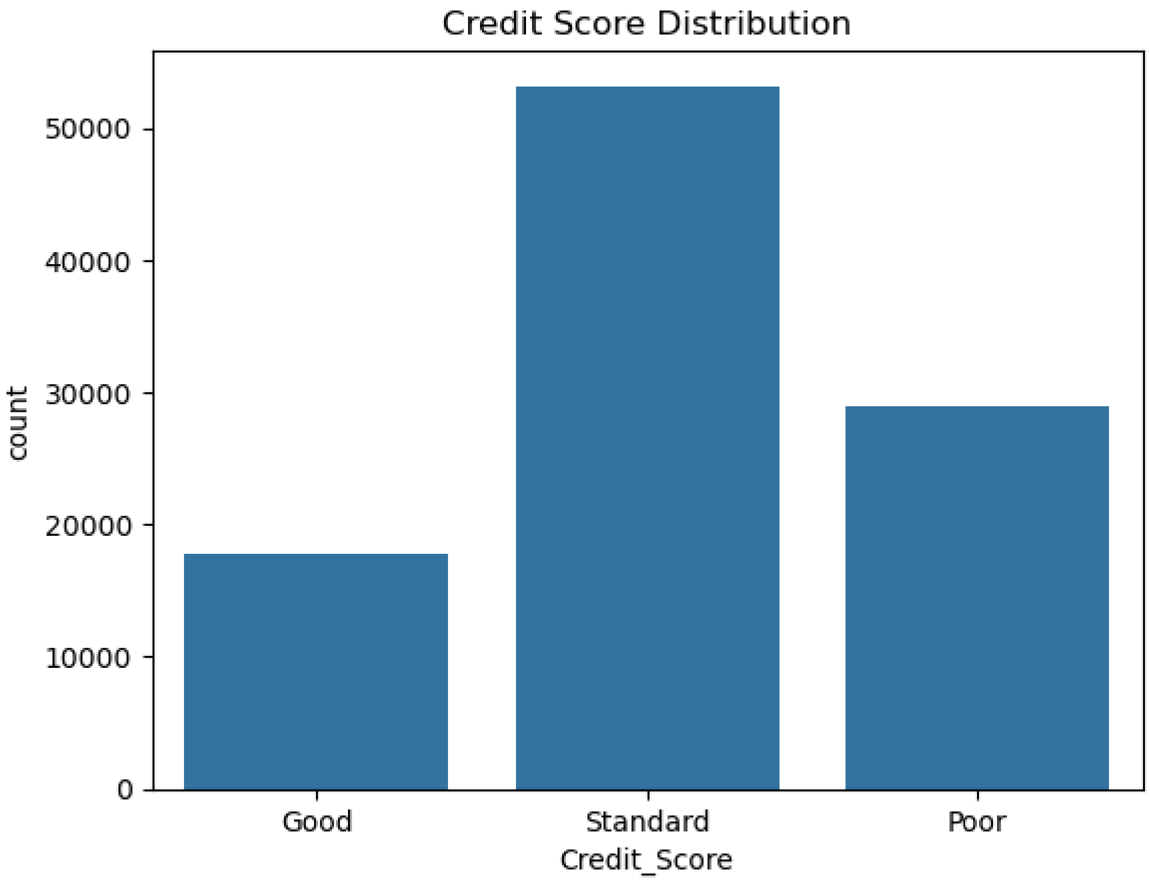
In [29]:

```
fig = px.box(data,
             x="Credit_Score",
             y="Monthly_Balance",
             color="Credit_Score",
             title="Credit Scores Based on Monthly Balance Left",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```



Exploratory Data Analysis - EDA

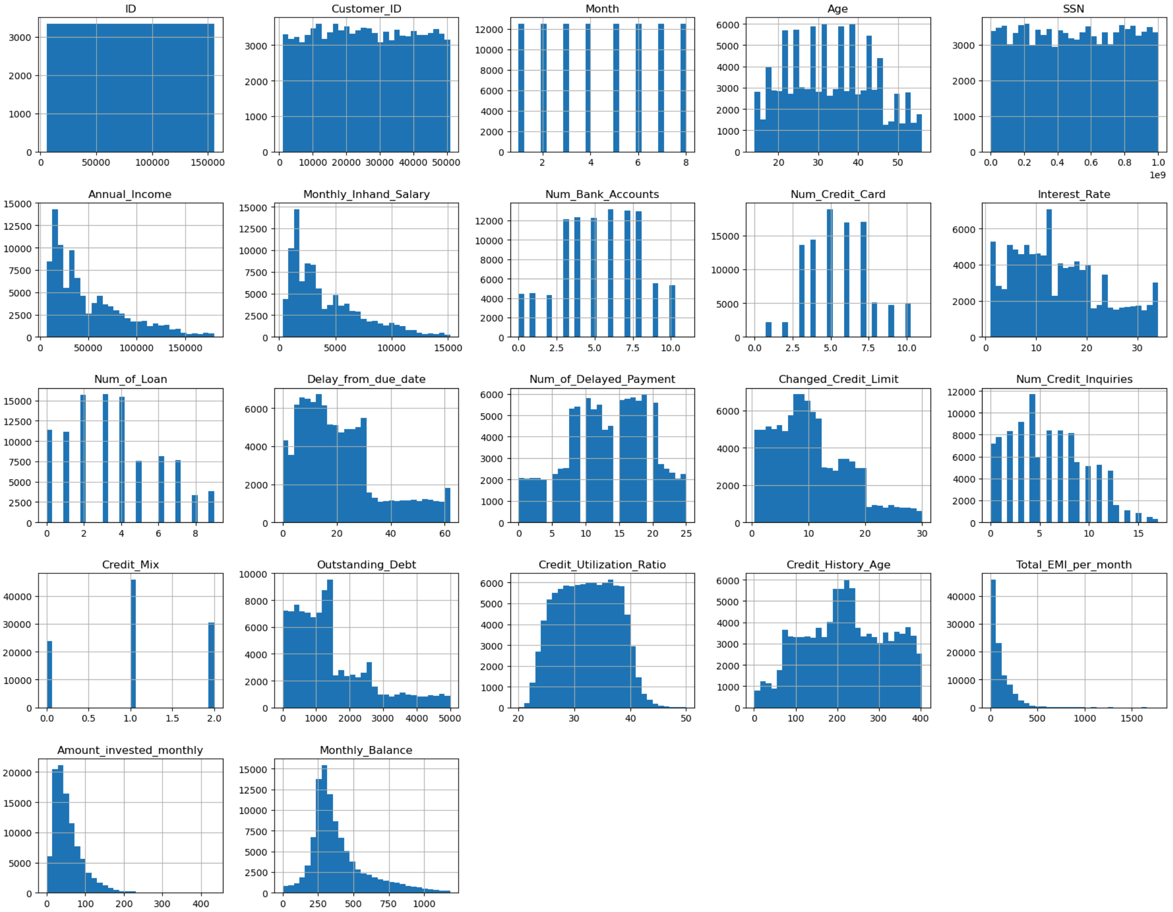
In [40]:
```
data['Credit_Score'].value_counts()

sns.countplot(data=data, x='Credit_Score')
plt.title("Credit Score Distribution")
plt.show()
```



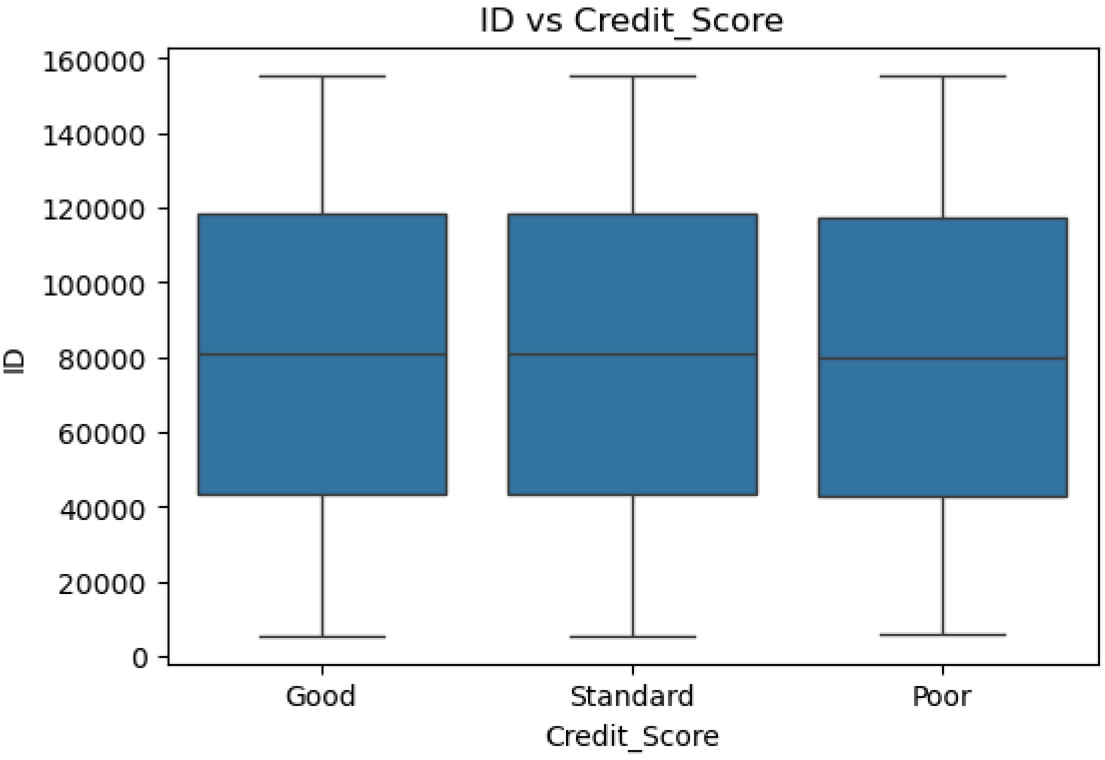
In [43]:
```
numerical_cols = data.select_dtypes(include=['int64', 'float64']).columns
categorical_cols = data.select_dtypes(include=['object']).columns
```

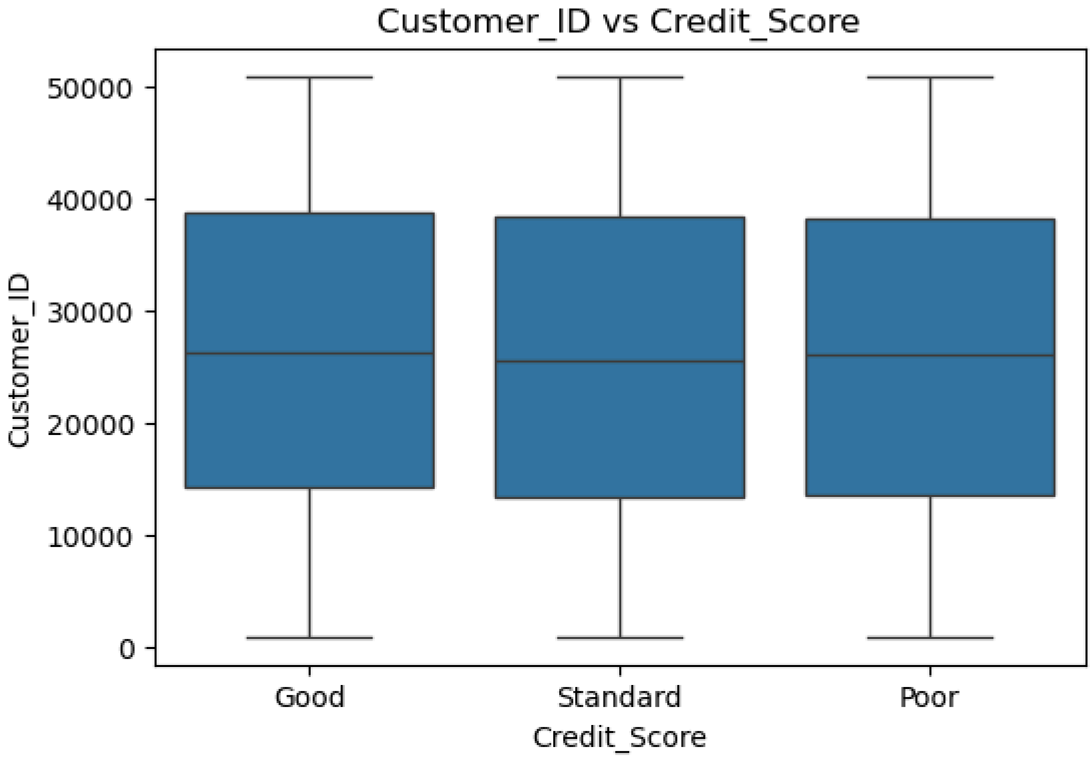
Target Variable Distribution (Credit_Score)

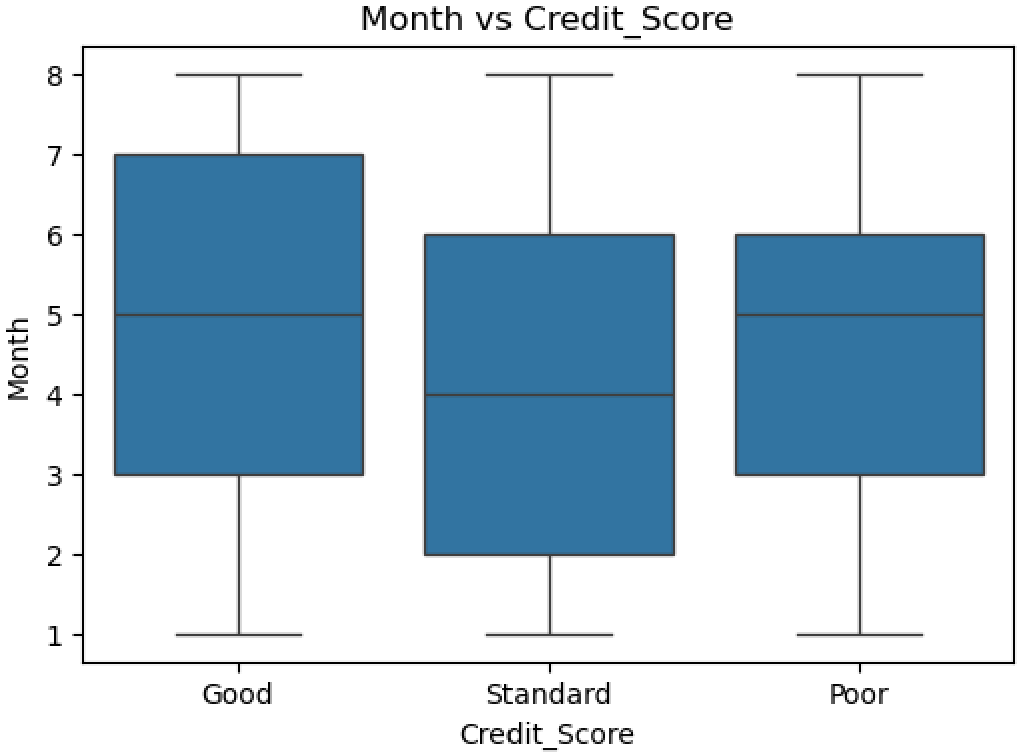
In [55]:
```
data[numerical_cols].hist(bins=30, figsize=(18, 14))
plt.tight_layout()
plt.show()
```

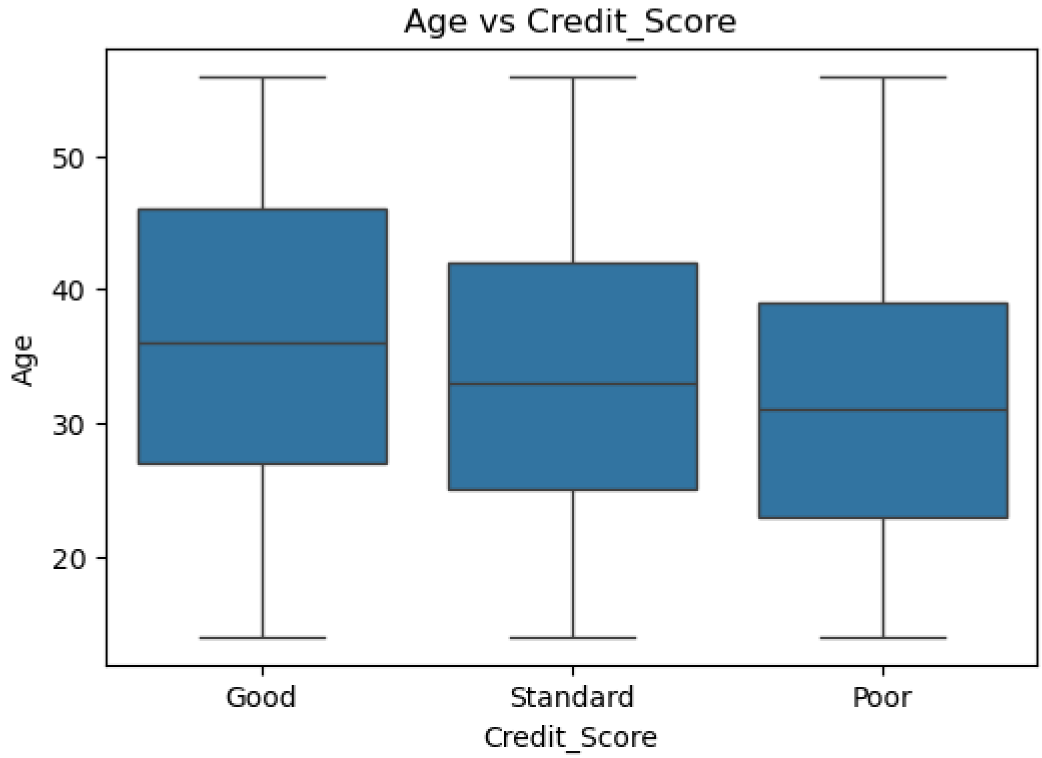


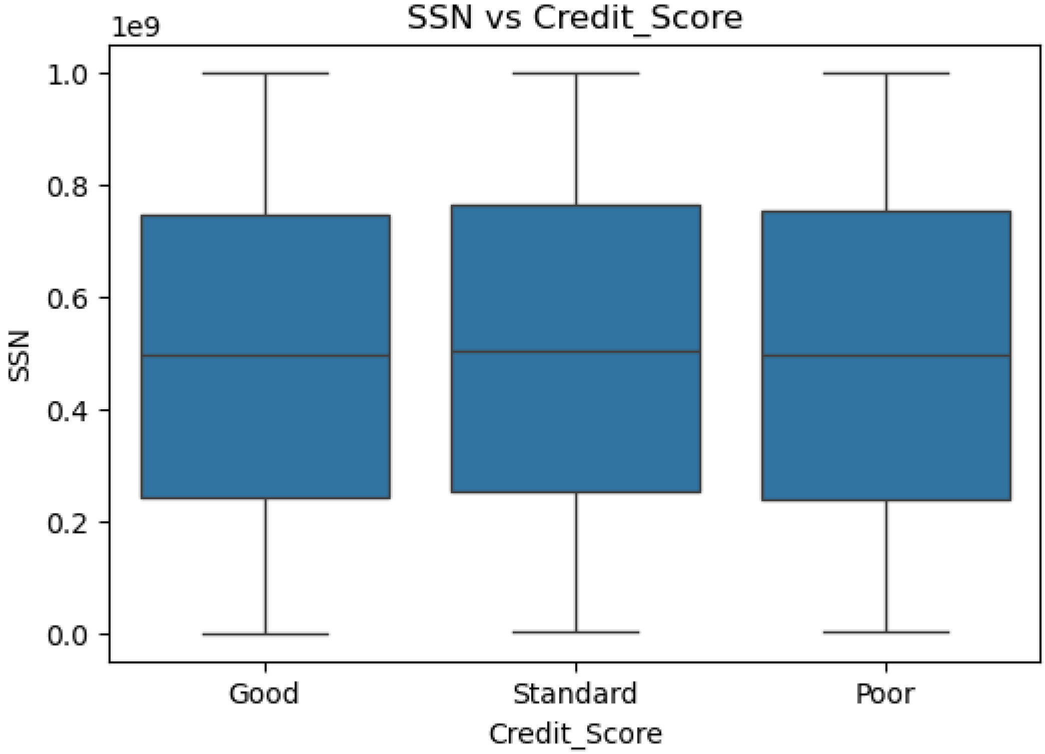
```
In [46]: for col in numerical_cols:
             plt.figure(figsize=(6, 4))
             sns.boxplot(x='Credit_Score', y=col, data=data)
             plt.title(f"{col} vs Credit_Score")
             plt.show()
```

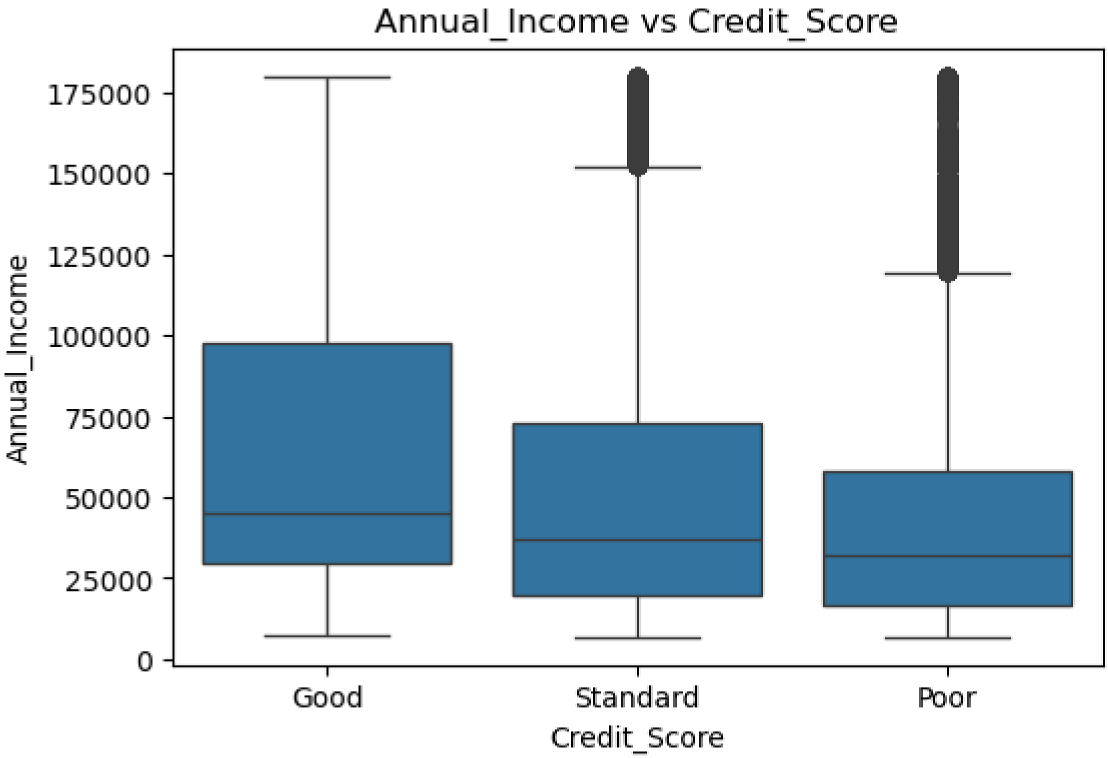


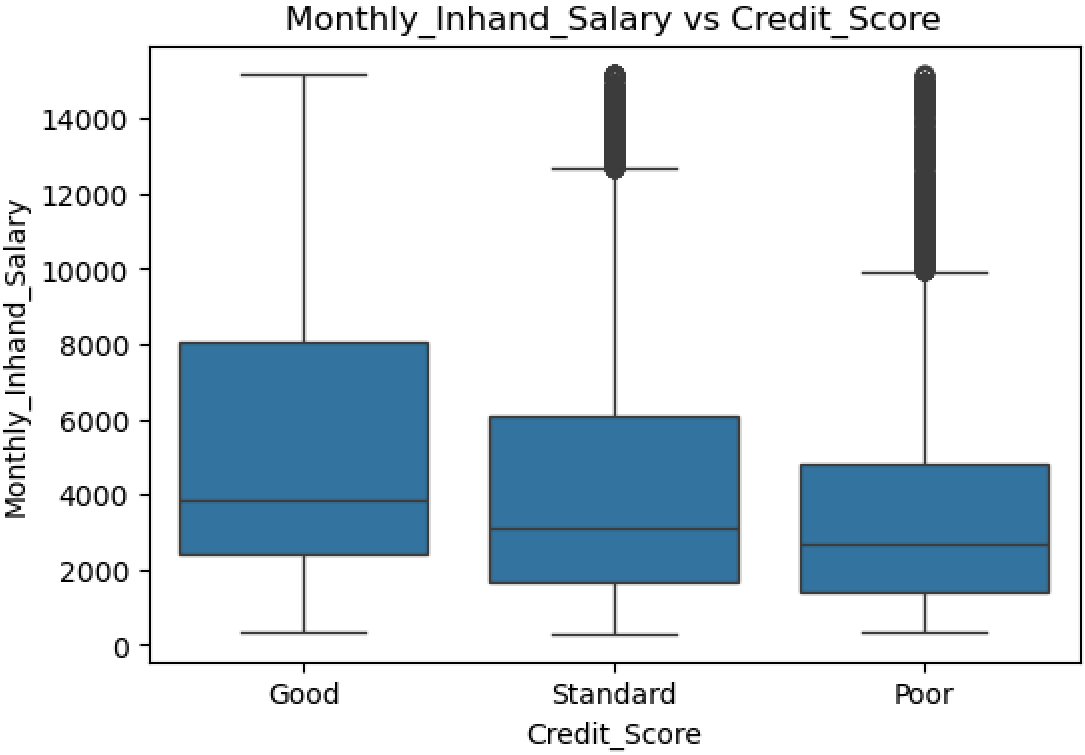

Customer_ID vs Credit_Score
Customer_ID
0
10000
20000
30000
40000
50000
Good
Standard
Poor
Credit_Score

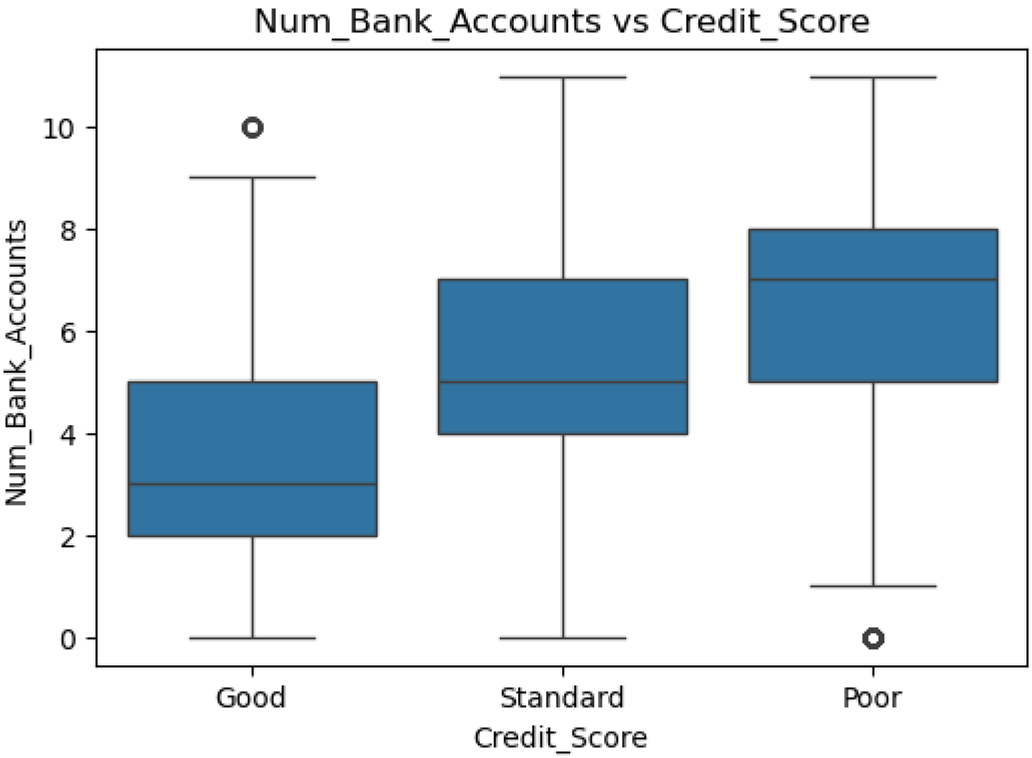

Month vs Credit_Score
Month
1
2
3
4
5
6
7
8
Good
Standard
Poor
Credit_Score

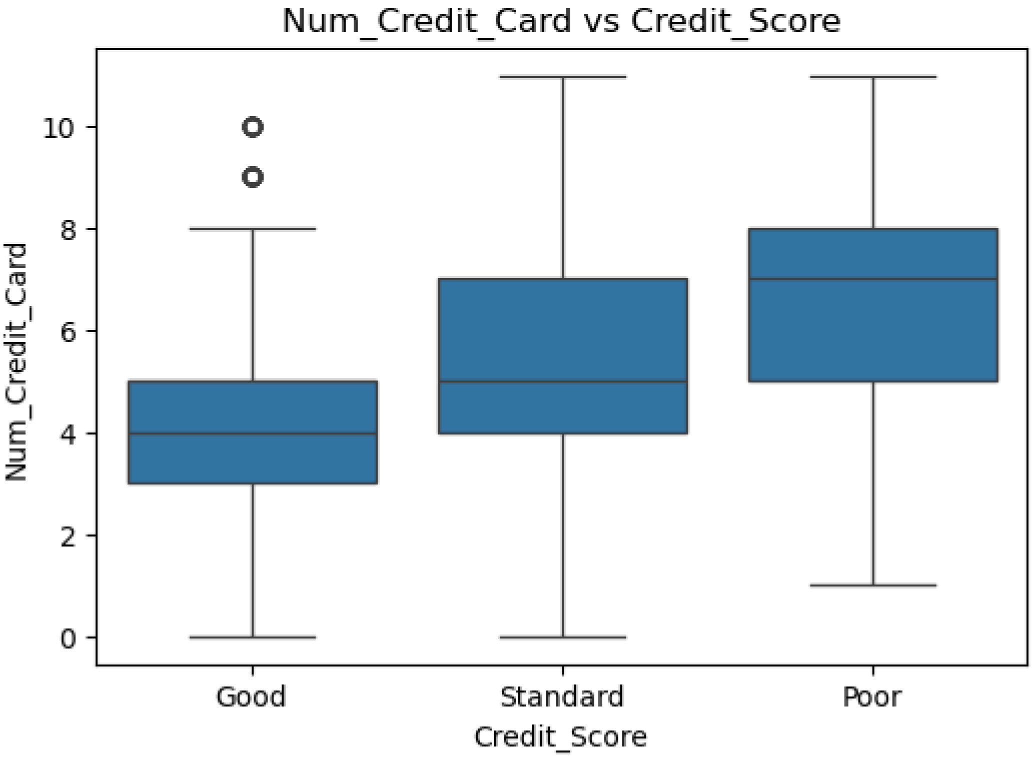

Age vs Credit_Score
Age
50
40
30
20
Good
Standard
Poor
Credit_Score

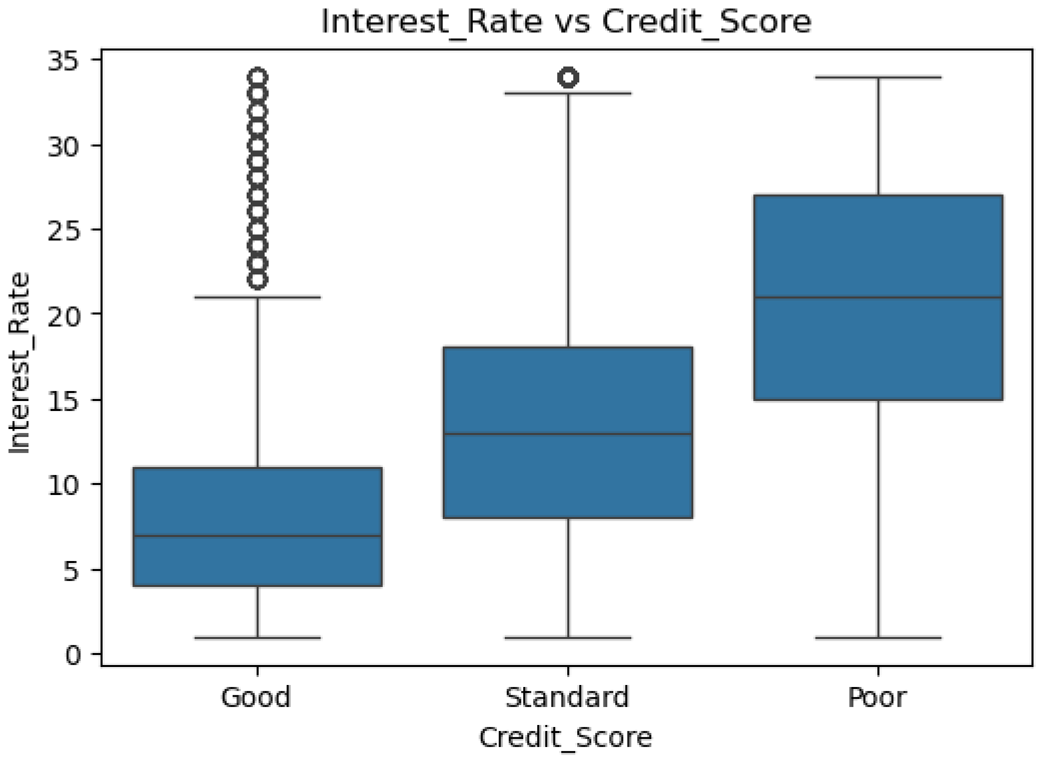

SSN vs Credit_Score
1e9
SSN
1.0
0.8
0.6
0.4
0.2
0.0
Good
Standard
Poor
Credit_Score

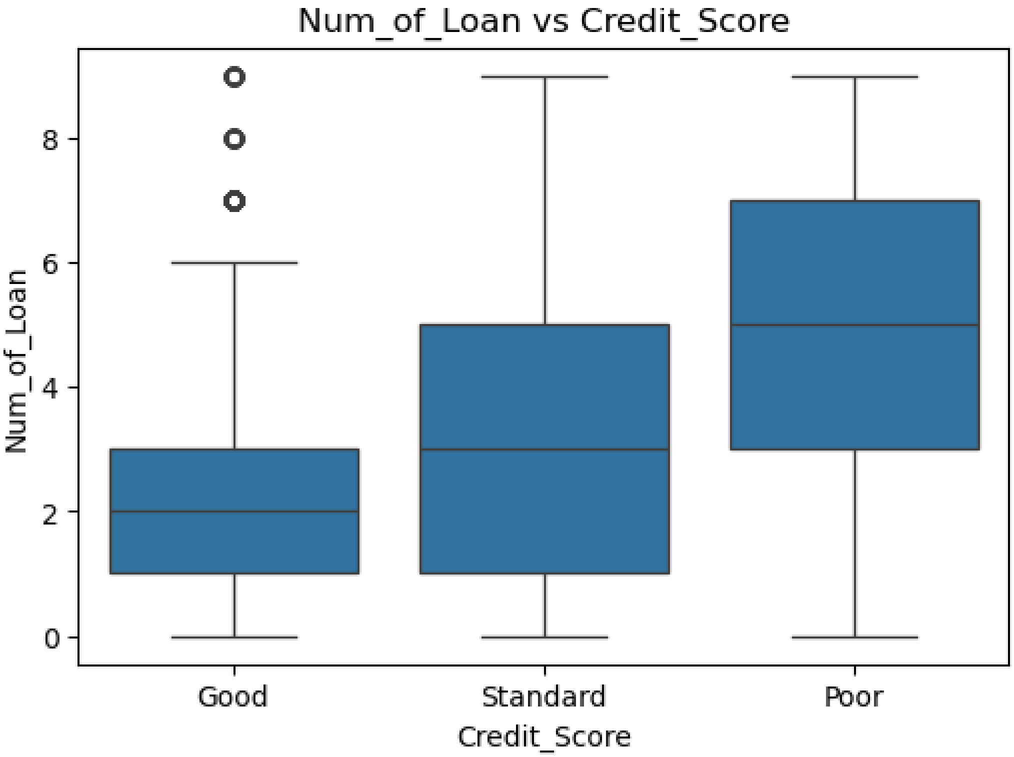

Annual_Income vs Credit_Score
Annual_Income
175000
150000
125000
100000
75000
50000
25000
0
Good
Standard
Poor
Credit_Score

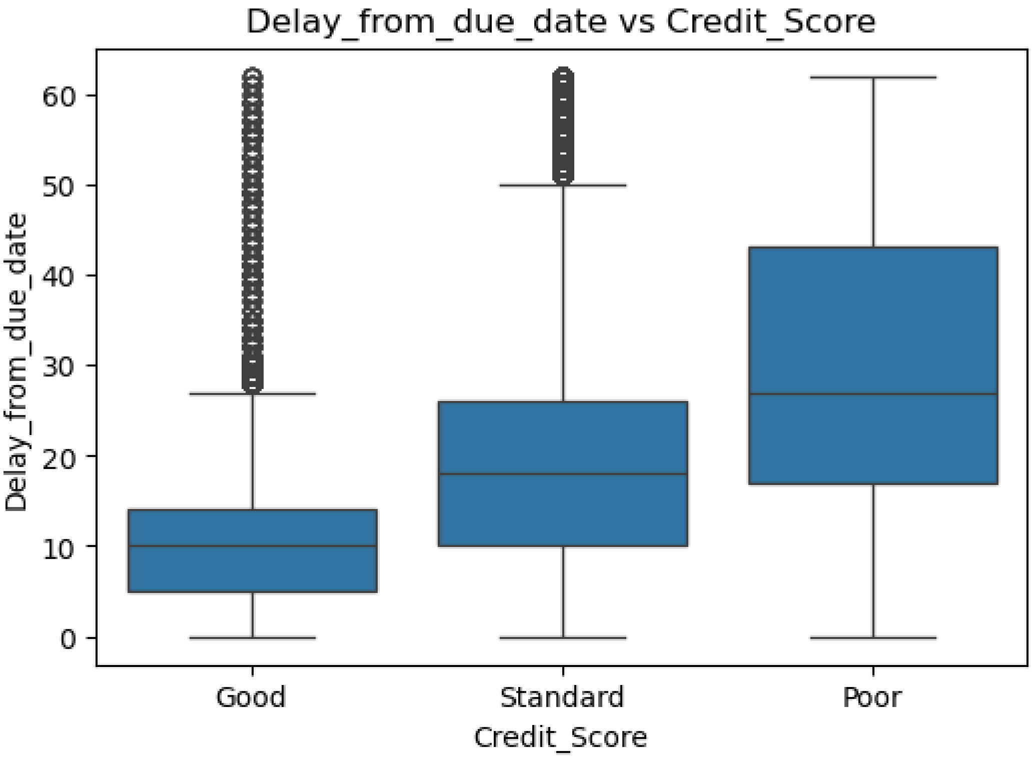

Monthly_Inhand_Salary vs Credit_Score
Monthly_Inhand_Salary
14000
12000
10000
8000
6000
4000
2000
0
Good
Standard
Poor
Credit_Score

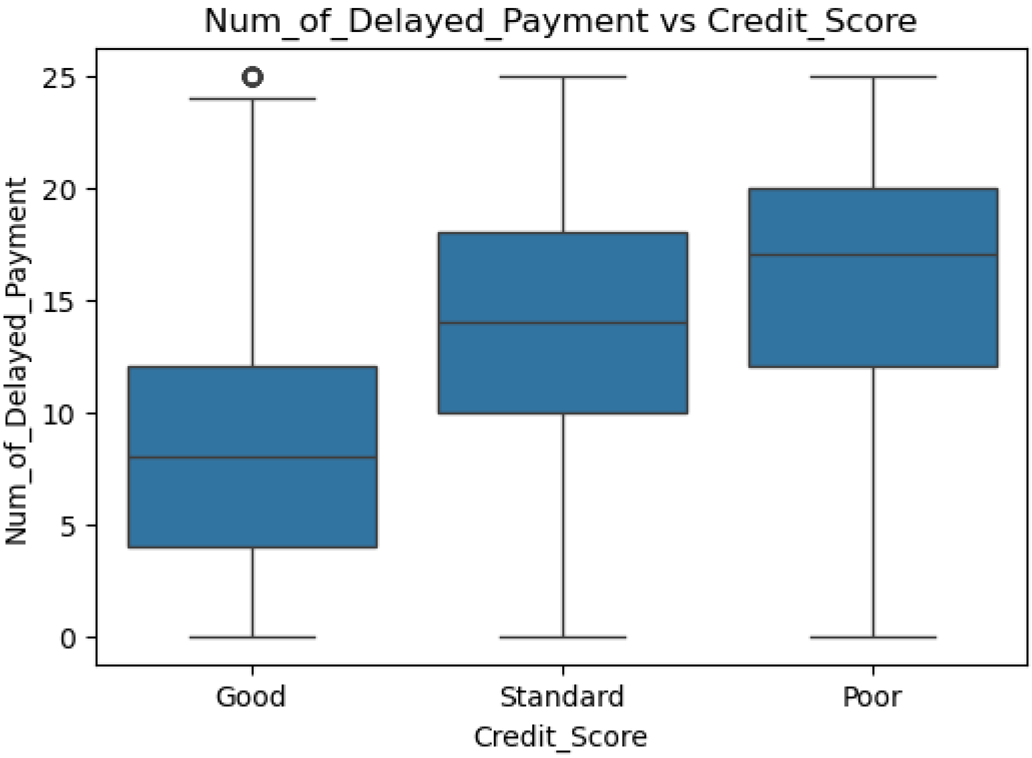

Num_Bank_Accounts vs Credit_Score
Num_Bank_Accounts
0
2
4
6
8
10
Good
Standard
Poor
Credit_Score

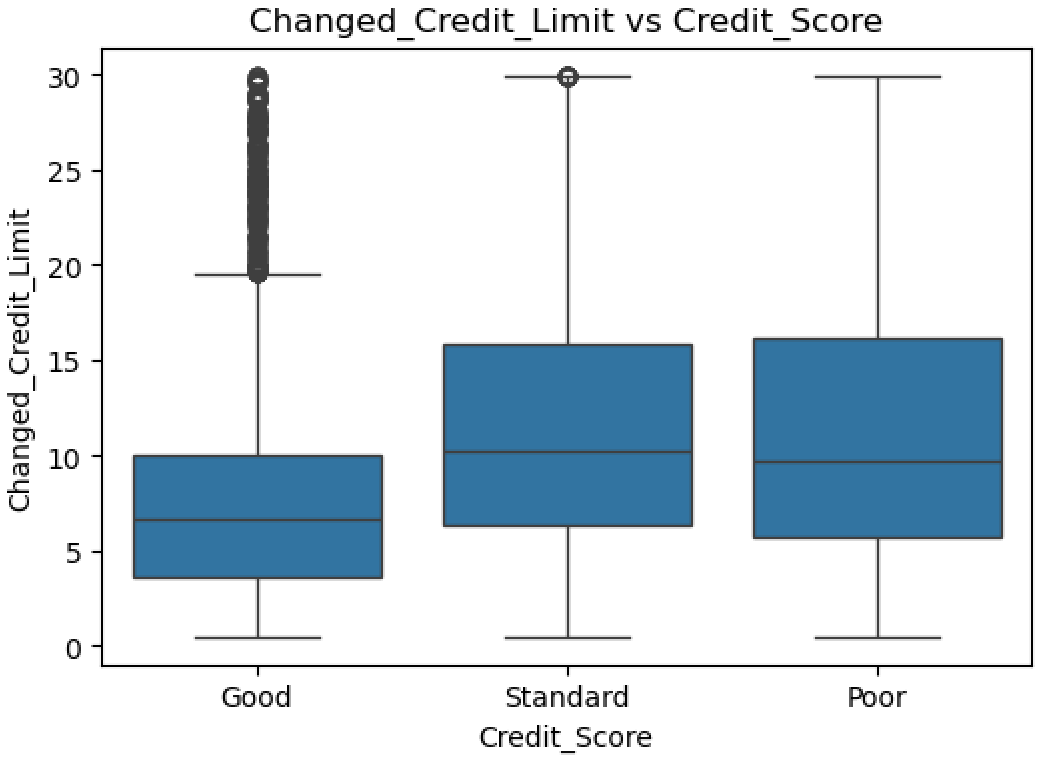

Num_Credit_Card vs Credit_Score
Num_Credit_Card
0
2
4
6
8
10
Good
Standard
Poor
Credit_Score

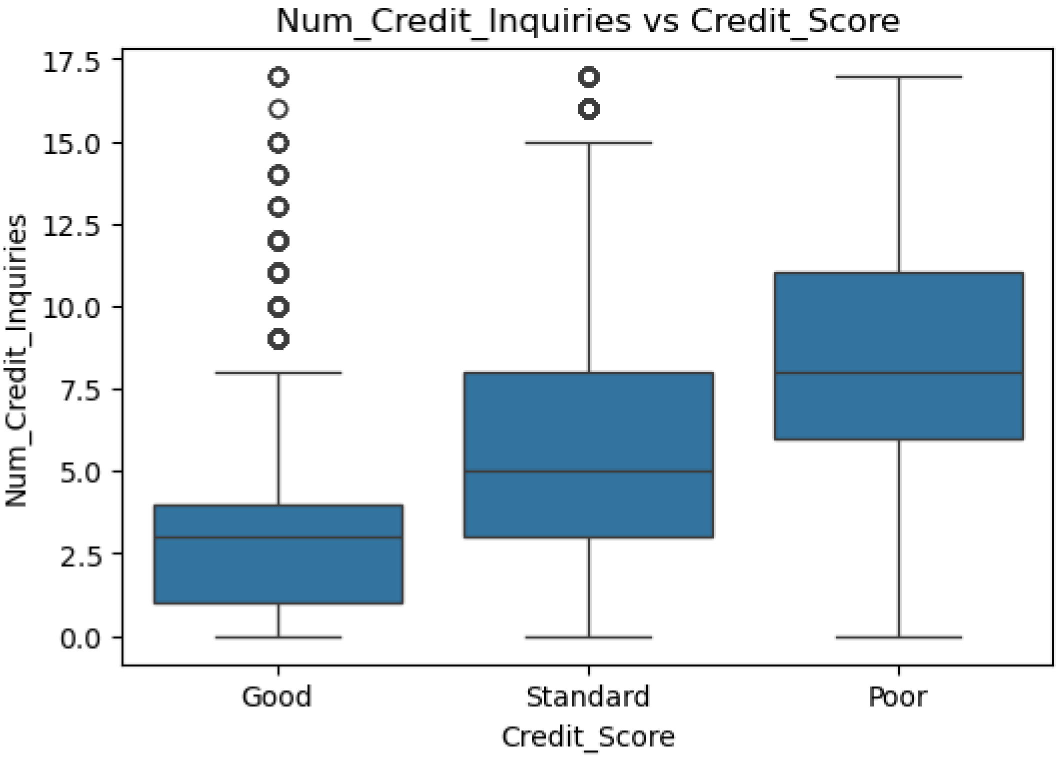

Interest_Rate vs Credit_Score
Interest_Rate
0
5
10
15
20
25
30
35
Good
Standard
Poor
Credit_Score

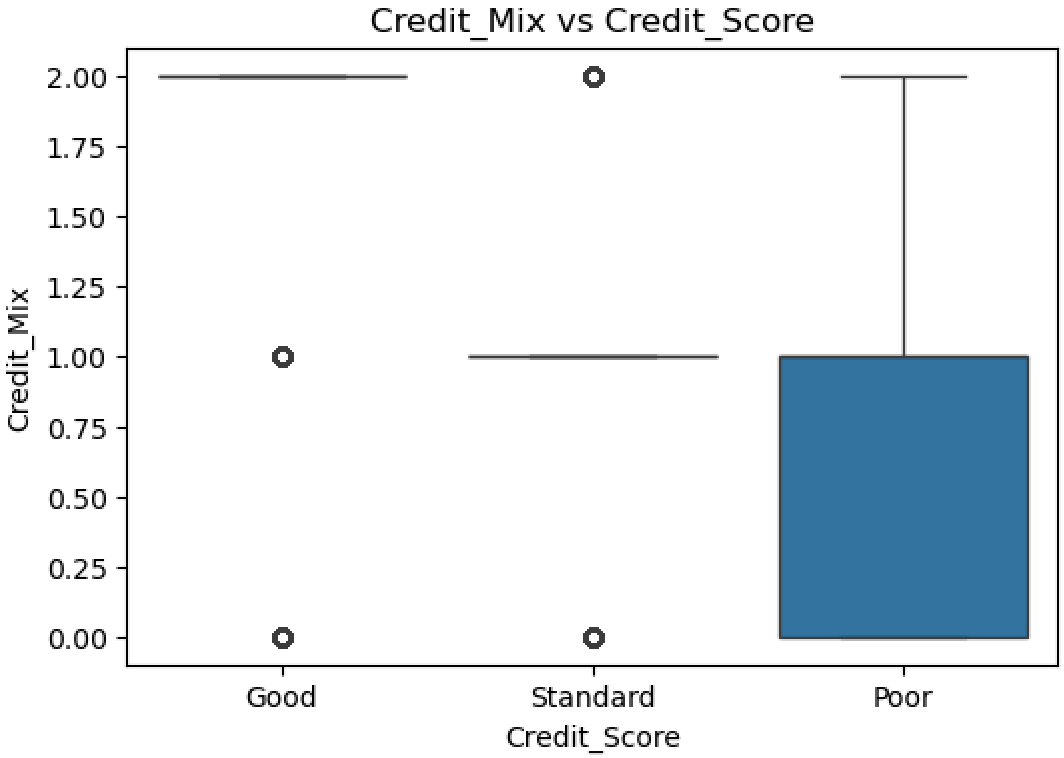

Num_of_Loan vs Credit_Score
Num_of_Loan
0
2
4
6
8
Good
Standard
Poor
Credit_Score

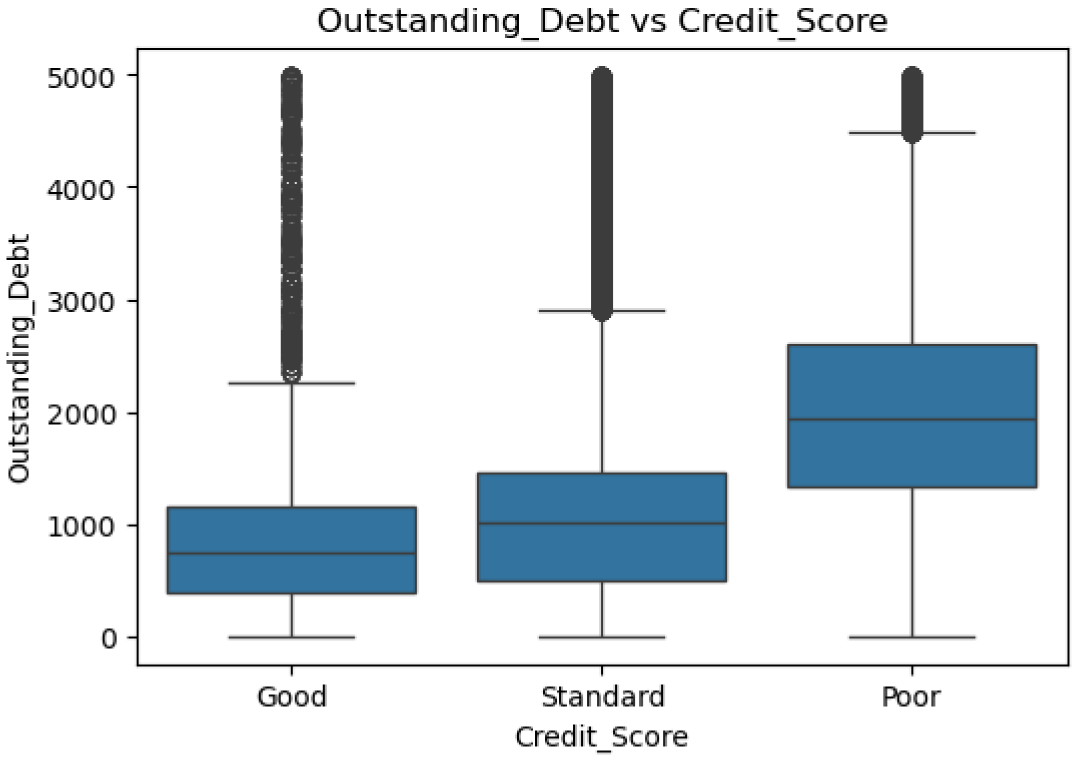

Delay_from_due_date vs Credit_Score
Delay_from_due_date
0
10
20
30
40
50
60
Good
Standard
Poor
Credit_Score

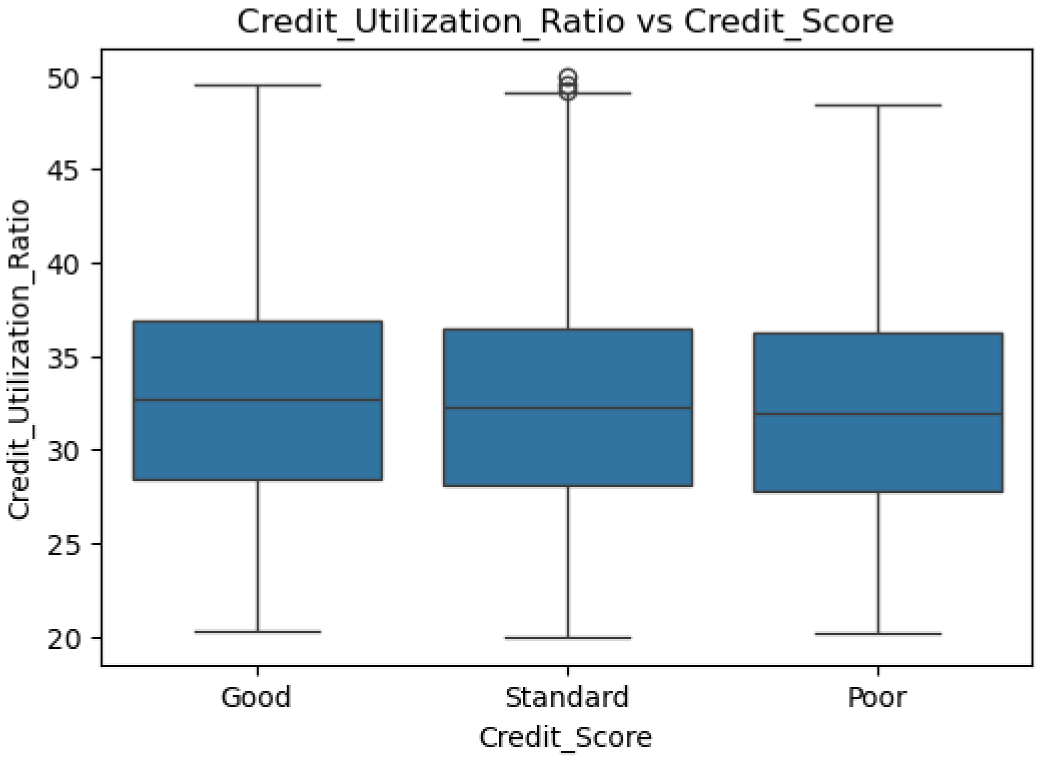

Num_of_Delayed_Payment vs Credit_Score
Num_of_Delayed_Payment
0
5
10
15
20
25
Good
Standard
Poor
Credit_Score

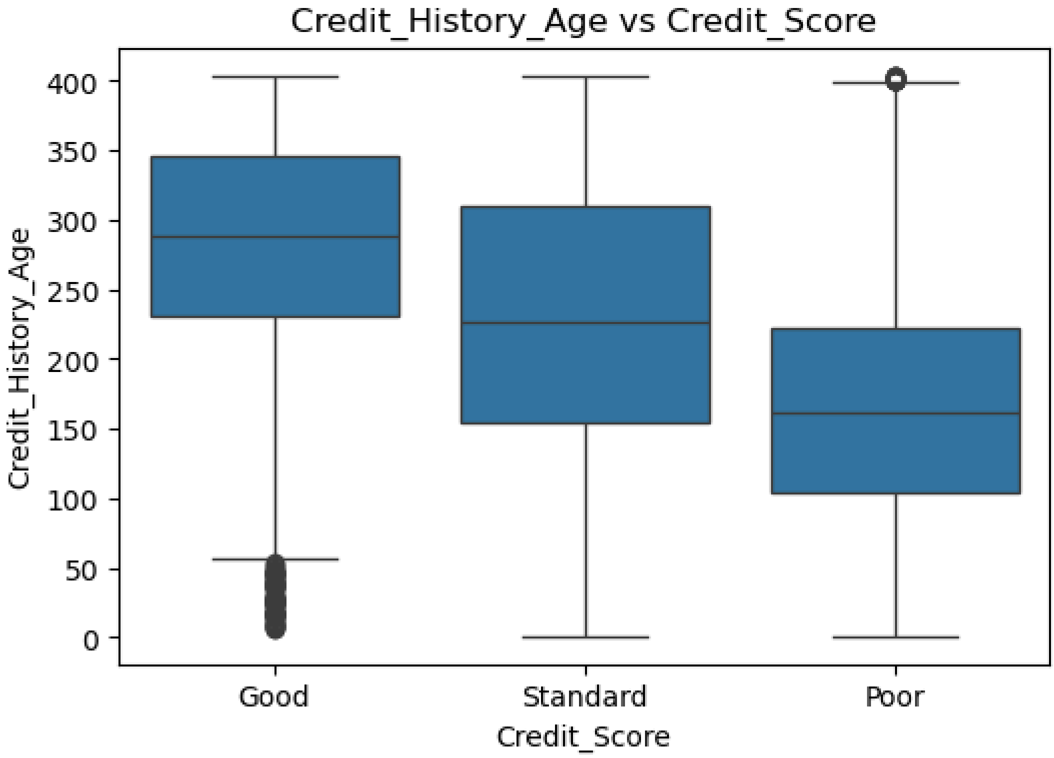

Changed_Credit_Limit vs Credit_Score
Changed_Credit_Limit
0
5
10
15
20
25
30
Good
Standard
Poor
Credit_Score

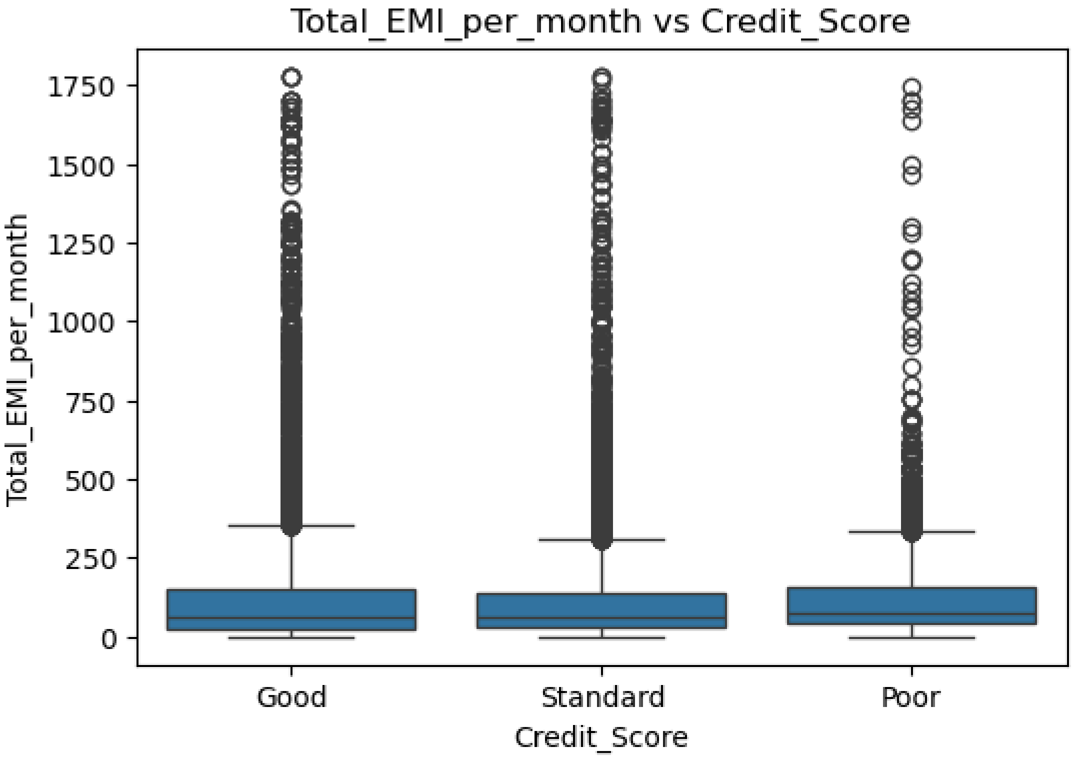

Num_Credit_Inquiries vs Credit_Score
Num_Credit_Inquiries
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Good
Standard
Poor
Credit_Score

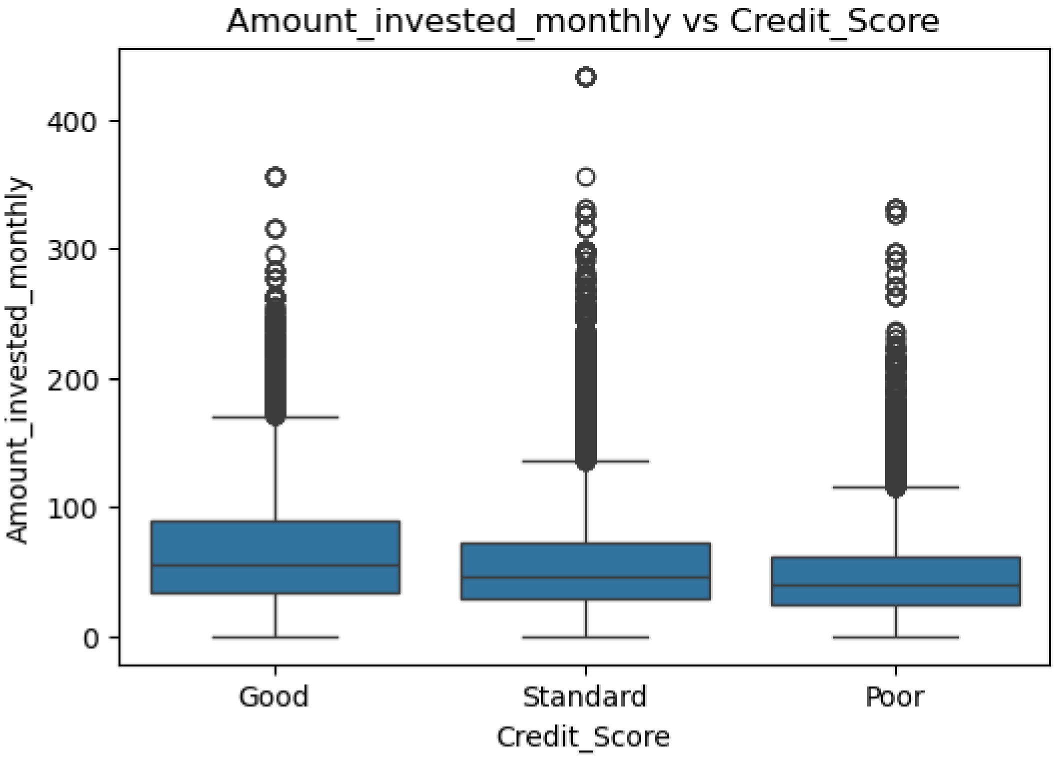

Credit_Mix vs Credit_Score
Credit_Mix
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Good
Standard
Poor
Credit_Score

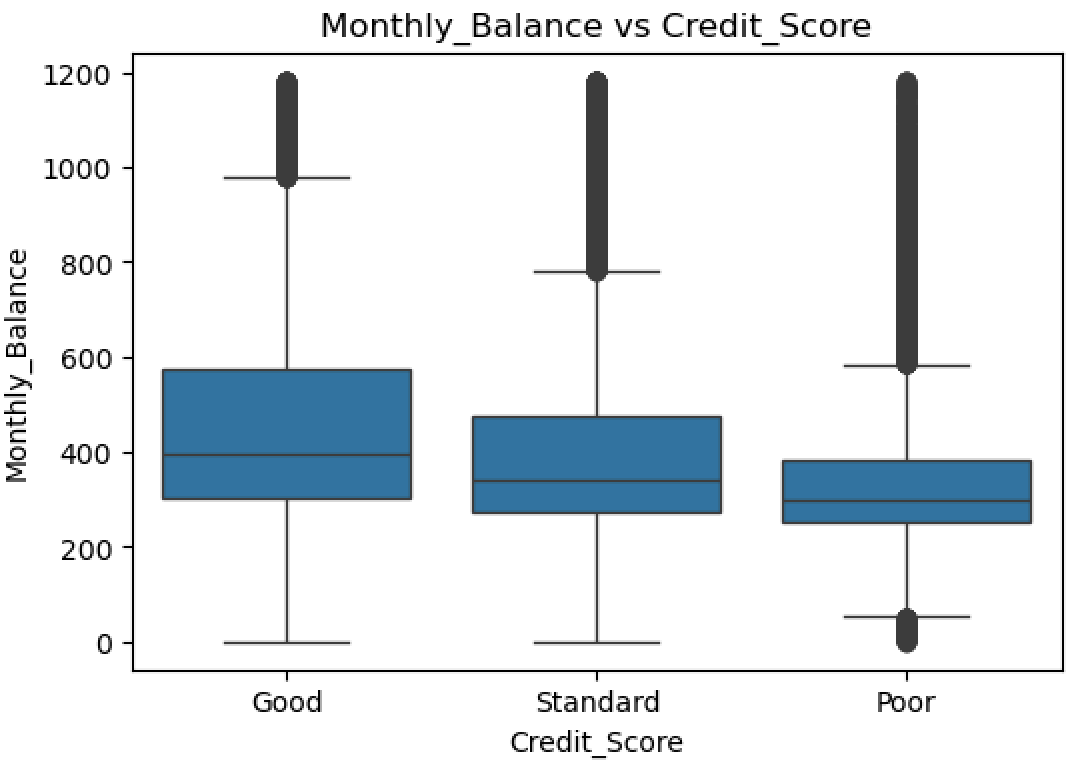

Outstanding_Debt vs Credit_Score
Outstanding_Debt
5000
4000
3000
2000
1000
0
Good
Standard
Poor
Credit_Score


Credit_Utilization_Ratio vs Credit_Score
Credit_Utilization_Ratio
50
45
40
35
30
25
20
Good
Standard
Poor
Credit_Score


Credit_History_Age vs Credit_Score
Credit_History_Age
400
350
300
250
200
150
100
50
0
Good
Standard
Poor
Credit_Score

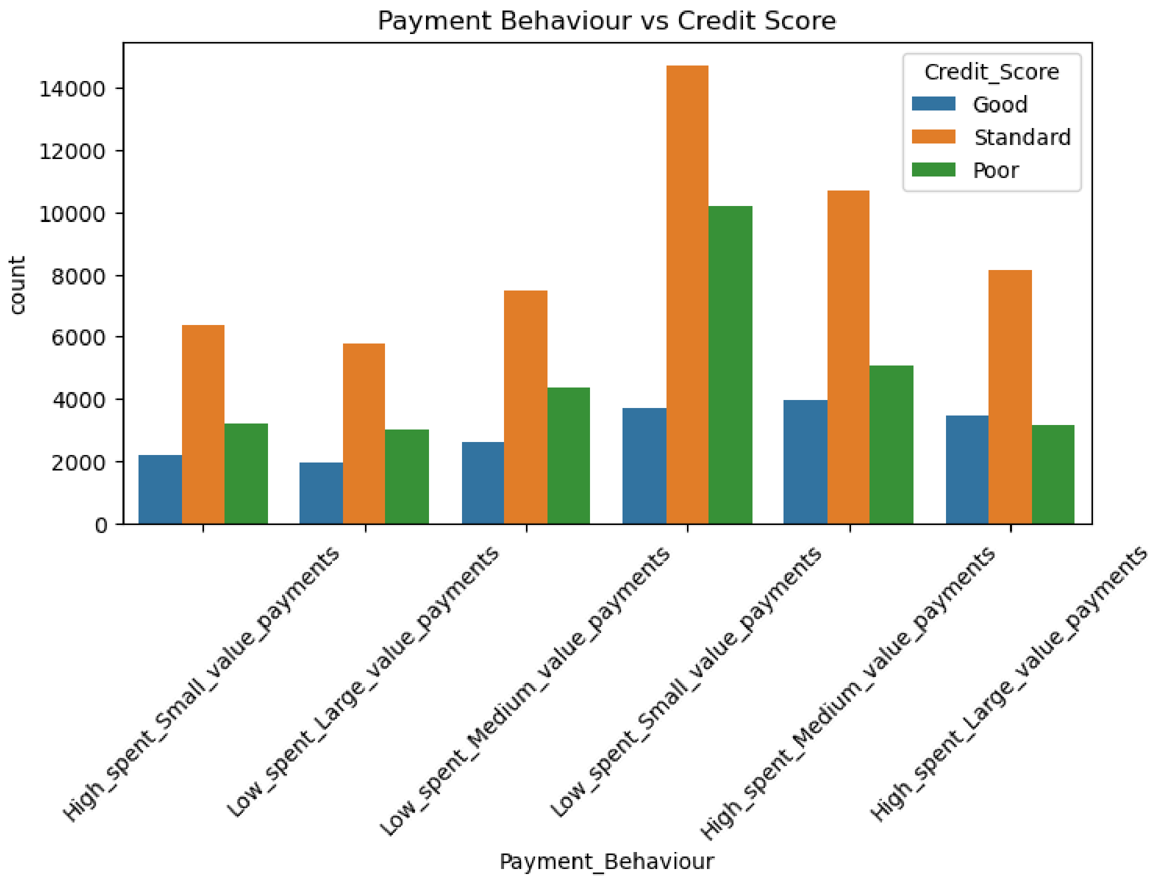

Total_EMI_per_month vs Credit_Score
Total_EMI_per_month
1750
1500
1250
1000
750
500
250
0
Good
Standard
Poor
Credit_Score


Amount_invested_monthly vs Credit_Score
Amount_invested_monthly
400
300
200
100
0
Good
Standard
Poor
Credit_Score



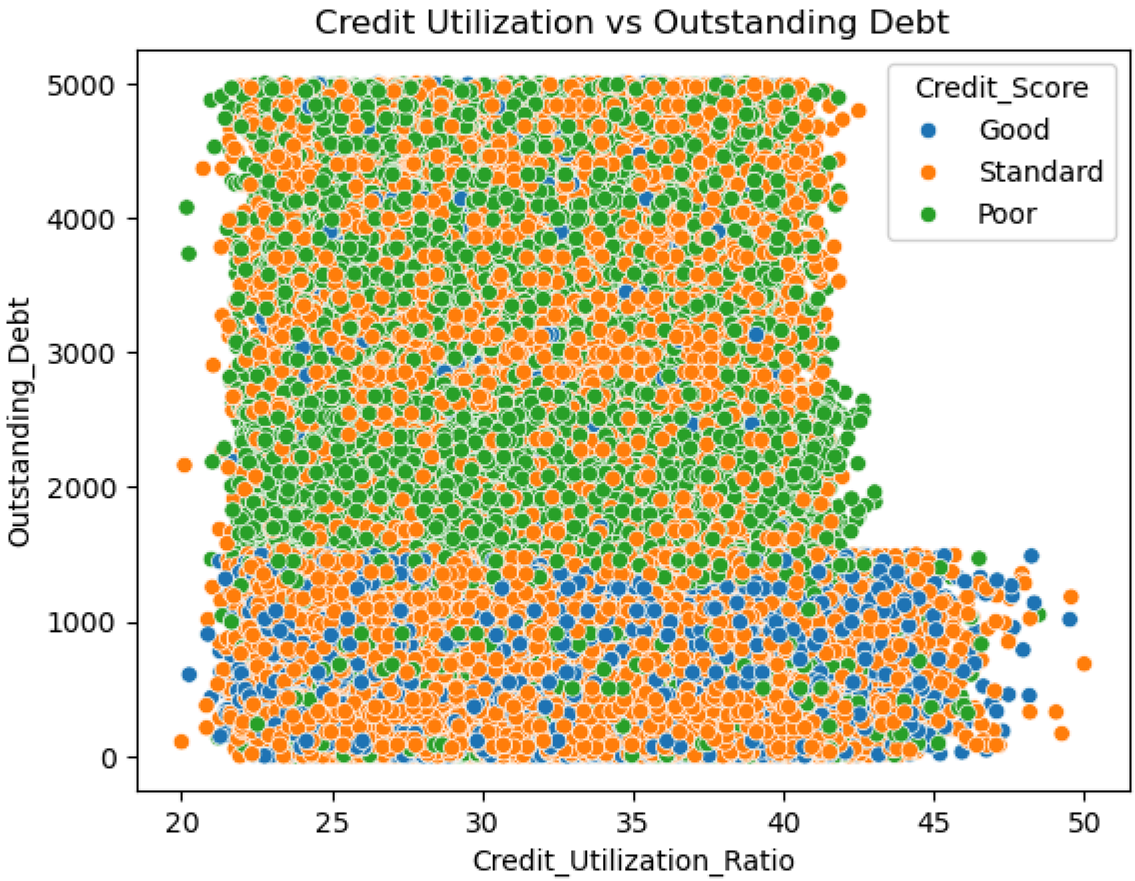
In [48]:

```
plt.figure(figsize=(8, 4))
sns.countplot(data=data, x='Payment_Behaviour', hue='Credit_Score')
plt.xticks(rotation=45)
plt.title("Payment Behaviour vs Credit Score")
plt.show()
```



In [50]:

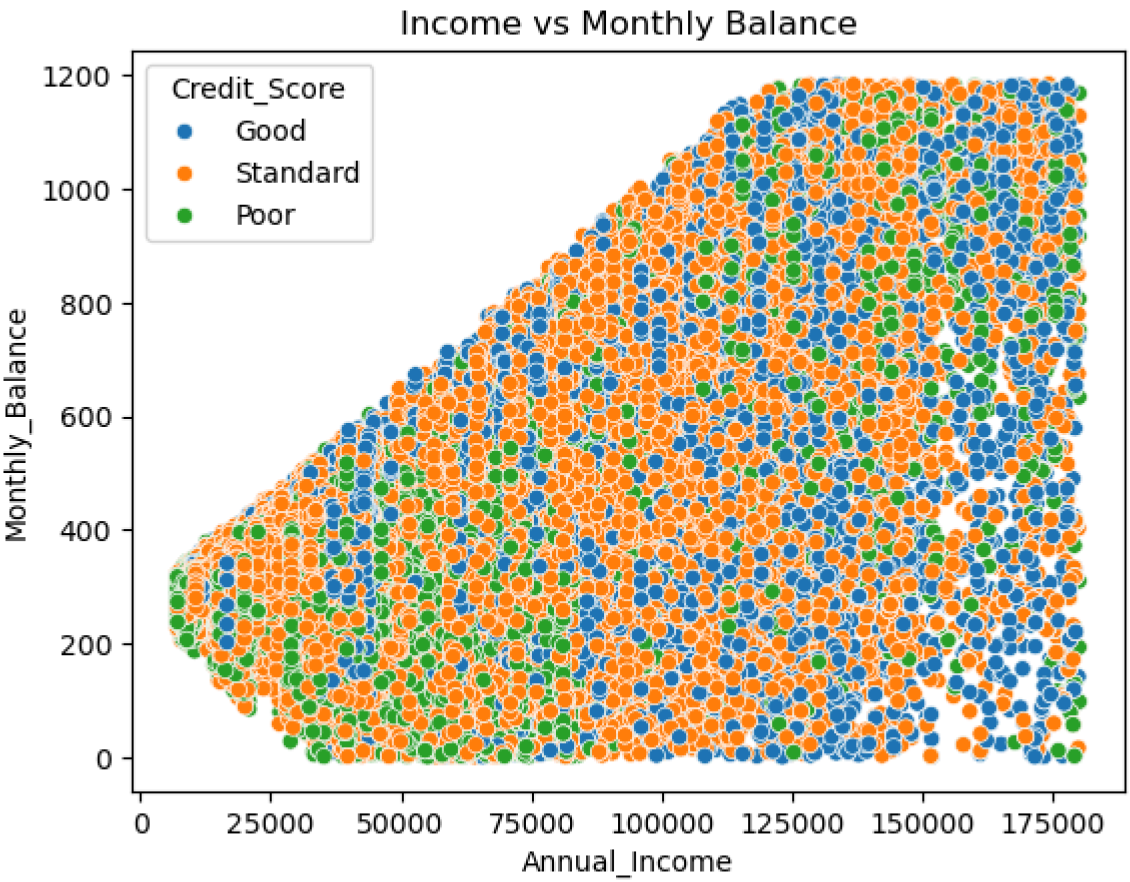
```
sns.scatterplot(
    data=data,
    x='Credit_Utilization_Ratio',
    y='Outstanding_Debt',
    hue='Credit_Score'
)
plt.title("Credit Utilization vs Outstanding Debt")
plt.show()
```



In [51]:

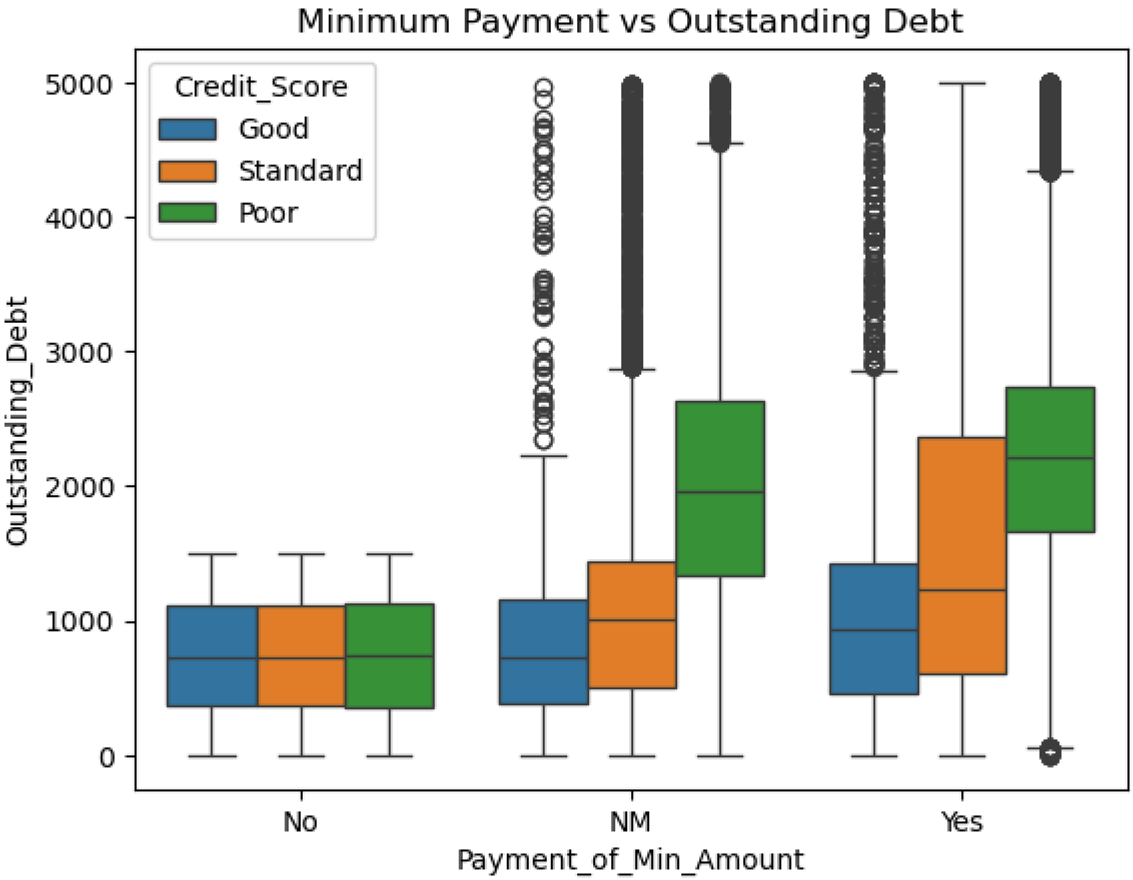
```
sns.scatterplot(
    data=data,
    x='Annual_Income',
    y='Monthly_Balance',
    hue='Credit_Score'
)
plt.title("Income vs Monthly Balance")
plt.show()
```

```
C:\Users\USER\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:170: UserWar
ning:

Creating legend with loc="best" can be slow with large amounts of data.
```



In [52]:

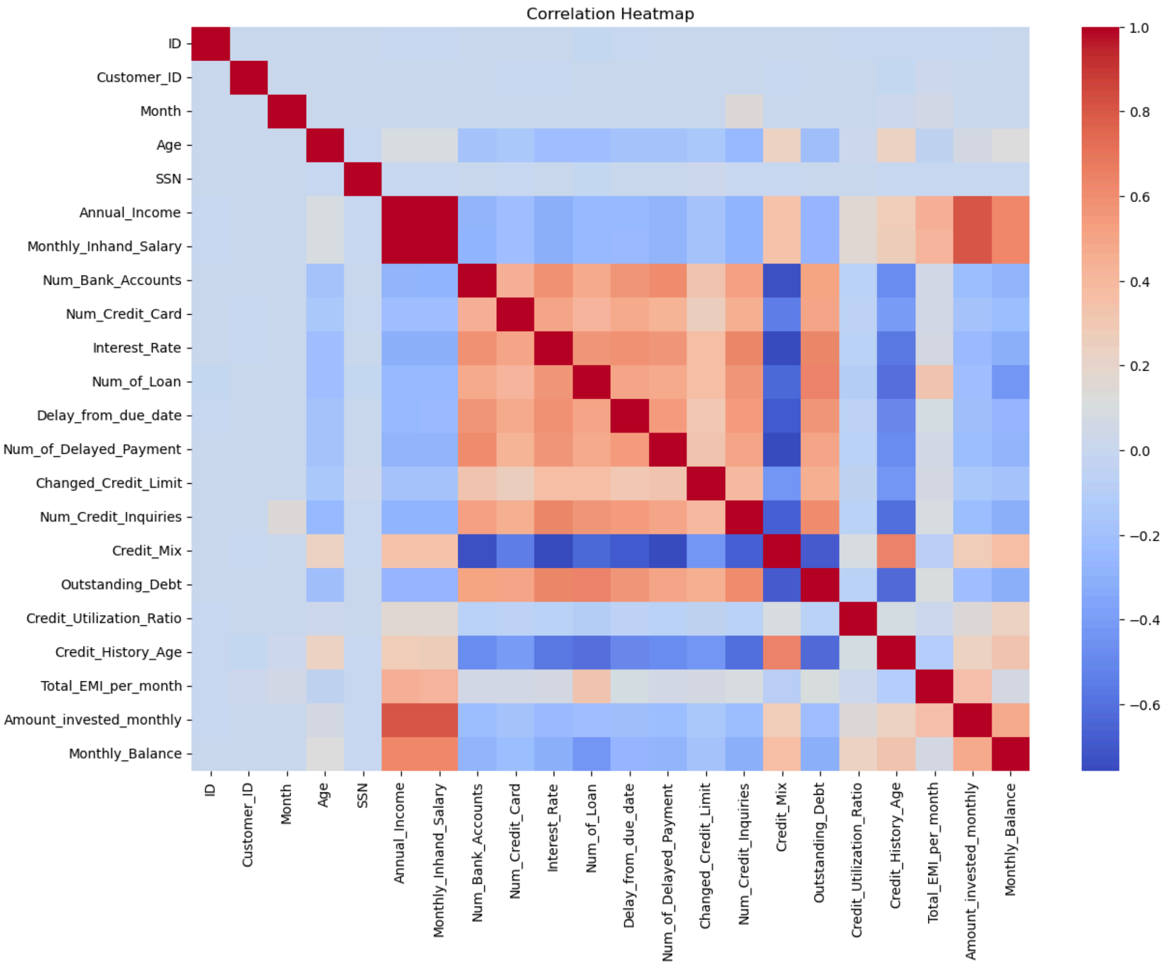
```
sns.boxplot(
    data=data,
    x='Payment_of_Min_Amount',
    y='Outstanding_Debt',
    hue='Credit_Score'
)
plt.title("Minimum Payment vs Outstanding Debt")
plt.show()
```



Correlation Analysis

In [53]:
```
corr = data[numerical_cols].corr()
```

In [54]:
```
plt.figure(figsize=(14, 10))
sns.heatmap(corr, cmap='coolwarm', annot=False)
plt.title("Correlation Heatmap")
plt.show()
```



Encoding

```
In [64]: data["Credit_Mix"] = data["Credit_Mix"].map({"Standard": 1,
                                    "Good": 2,
                                    "Bad": 0})
```

Train, Test and Split

```
In [65]: from sklearn.model_selection import train_test_split
x = np.array(data[["Annual_Income", "Monthly_Inhand_Salary",
                   "Num_Bank_Accounts", "Num_Credit_Card",
                   "Interest_Rate", "Num_of_Loan",
                   "Delay_from_due_date", "Num_of_Delayed_Payment",
                   "Credit_Mix", "Outstanding_Debt",
                   "Credit_History_Age", "Monthly_Balance"]])
y = np.array(data[["Credit_Score"]])
```

Model Building

```
In [67]: y = y.ravel()
```

```
In [68]: xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.33, random_sta

from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(xtrain, ytrain)
```

Out[68]:

```
▾ RandomForestClassifier ⓘ ?

RandomForestClassifier()
```

Credit Score Prediction

In [57]:

```python
print("Credit Score Prediction : ")
a = float(input("Annual Income: "))
b = float(input("Monthly Inhand Salary: "))
c = float(input("Number of Bank Accounts: "))
d = float(input("Number of Credit cards: "))
e = float(input("Interest rate: "))
f = float(input("Number of Loans: "))
g=float(input("Delay from due date: "))
h = float(input("Number of delayed payments: "))
i = input("Credit Mix (Bad: 0, Standard: 1, Good: 2) : ")
j = float(input("Outstanding Debt: "))
k = float(input("Credit History Age: "))
l = float(input("Monthly Balance: "))

features = np.array([[a, b, c, d, e, f, g, h, i, j, k, l]])
print("Predicted Credit Score = ", model.predict(features))
```

```
Credit Score Prediction :
Predicted Credit Score =  ['Good']
```