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

data = pd.read_csv("/content/sample_data/train.csv")
print(data.head())
```

```
          ID  Customer_ID  Month   Name  Age        SSN Occupation  Annual_Income  \
0  5634         3392      1  SIDHU   32  821000265  Scientist       19114.12
1  5635         3392      2  SIDHU   32  821000265  Scientist       19114.12
2  5636         3392      3  SIDHU   32  821000265  Scientist       19114.12
3  5637         3392      4  SIDHU   32  821000265  Scientist       19114.12
4  5638         3392      5  SIDHU   32  821000265  Scientist       19114.12

   Monthly_Inhand_Salary  Num_Bank_Accounts  ...  Credit_Mix  \
0            1824.843333                  3  ...        Good
1            1824.843333                  3  ...        Good
2            1824.843333                  3  ...        Good
3            1824.843333                  3  ...        Good
4            1824.843333                  3  ...        Good

   Outstanding_Debt  Credit_Utilization_Ratio Credit_History_Age  \
0            809.98                 26.822620                265
1            809.98                 31.944960                266
2            809.98                 28.609352                267
3            809.98                 31.377862                268
4            809.98                 24.797347                269

   Payment_of_Min_Amount  Total_EMI_per_month  Amount_invested_monthly  \
0                     No            49.574949                 21.46538
1                     No            49.574949                 21.46538
2                     No            49.574949                 21.46538
3                     No            49.574949                 21.46538
4                     No            49.574949                 21.46538

                  Payment_Behaviour Monthly_Balance  Credit_Score
0   High_spent_Small_value_payments      312.494089          Good
1    Low_spent_Large_value_payments      284.629163          Good
2   Low_spent_Medium_value_payments      331.209863          Good
3    Low_spent_Small_value_payments      223.451310          Good
4  High_spent_Medium_value_payments      341.489231          Good

[5 rows x 28 columns]
```

```
print(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  int64
 5   SSN                       100000 non-null  int64
 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  int64
 10  Num_Credit_Card           100000 non-null  int64
 11  Interest_Rate             100000 non-null  int64
 12  Num_of_Loan               100000 non-null  int64
 13  Type_of_Loan              100000 non-null  object
 14  Delay_from_due_date       100000 non-null  int64
 15  Num_of_Delayed_Payment    100000 non-null  int64
 16  Changed_Credit_Limit      100000 non-null  float64
 17  Num_Credit_Inquiries      100000 non-null  int64
 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  int64
 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(8), int64(13), object(7)
memory usage: 21.4+ MB
None
```

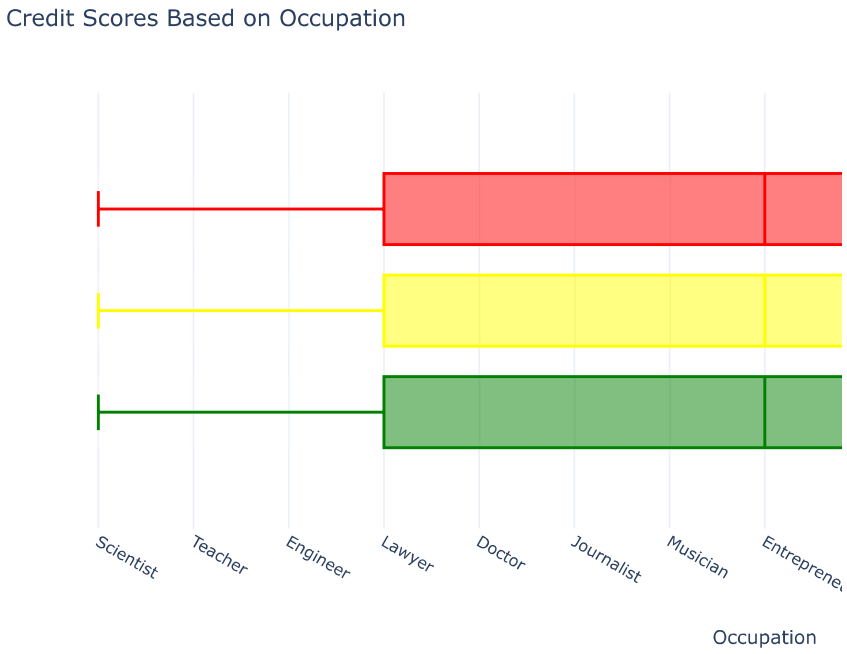
```
print(data.isnull().sum())
```

```
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
```

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

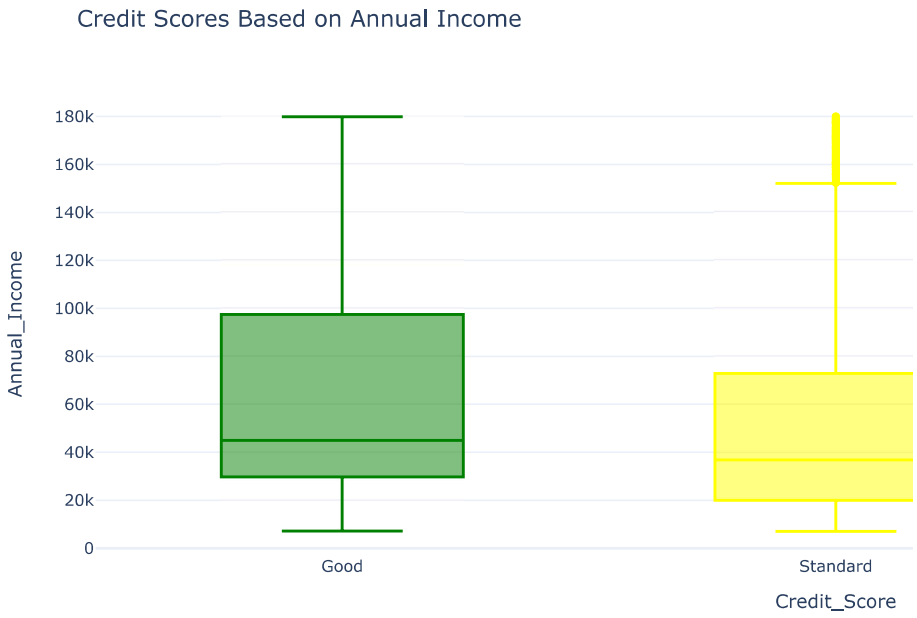
```
Standard    53174
Poor        28998
Good        17828
Name: Credit_Score, dtype: int64
```

```
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()
```

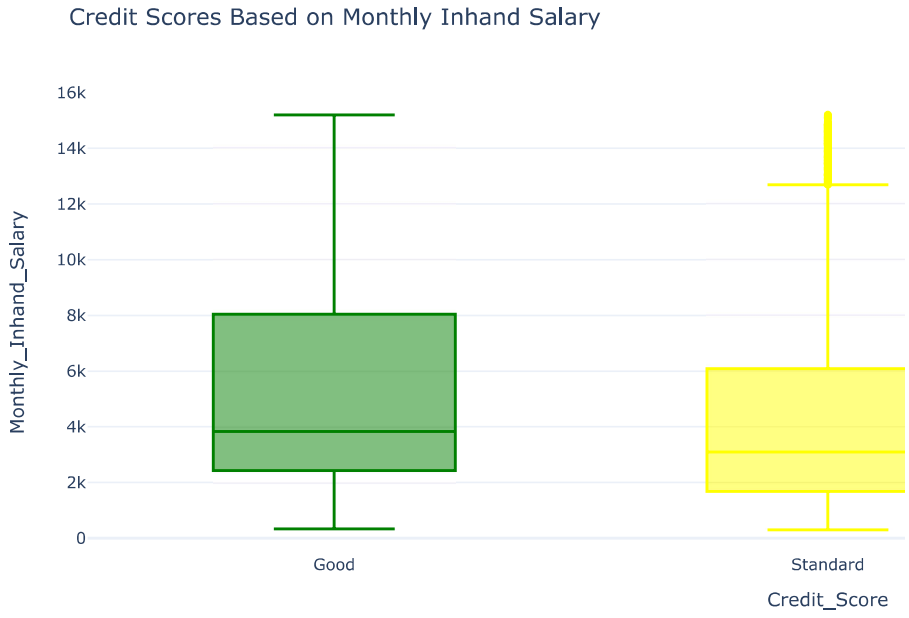



```
fig = px.box(data,
             x="Credit_Score",
             y="Annual_Income",
             color="Credit_Score",
```

```
            title="Credit Scores Based on Annual Income",
            color_discrete_map={'Poor':'red',
                                'Standard':'yellow',
                                'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```

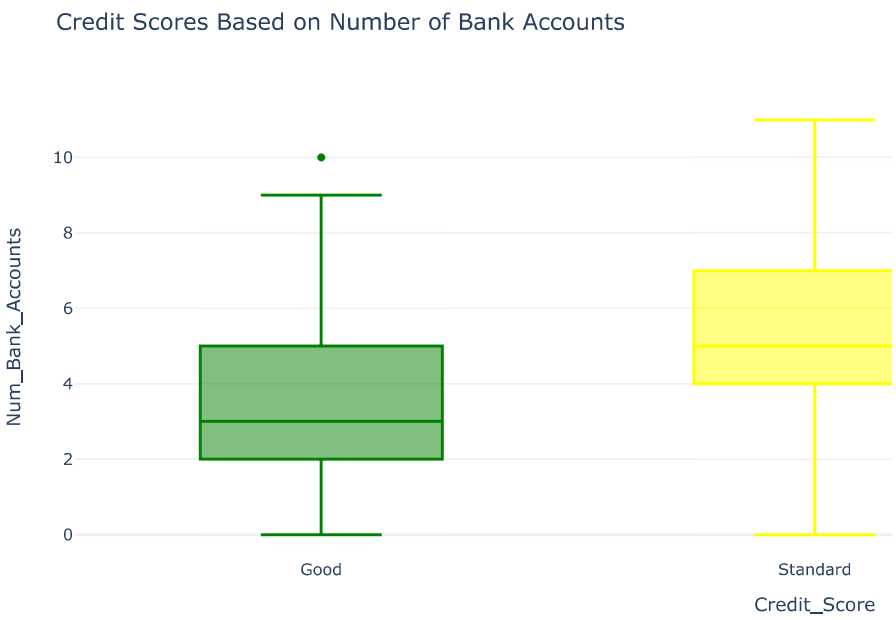



```
fig = px.box(data,
             x="Credit_Score",
             y="Monthly_Inhand_Salary",
             color="Credit_Score",
             title="Credit Scores Based on Monthly Inhand Salary",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
   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

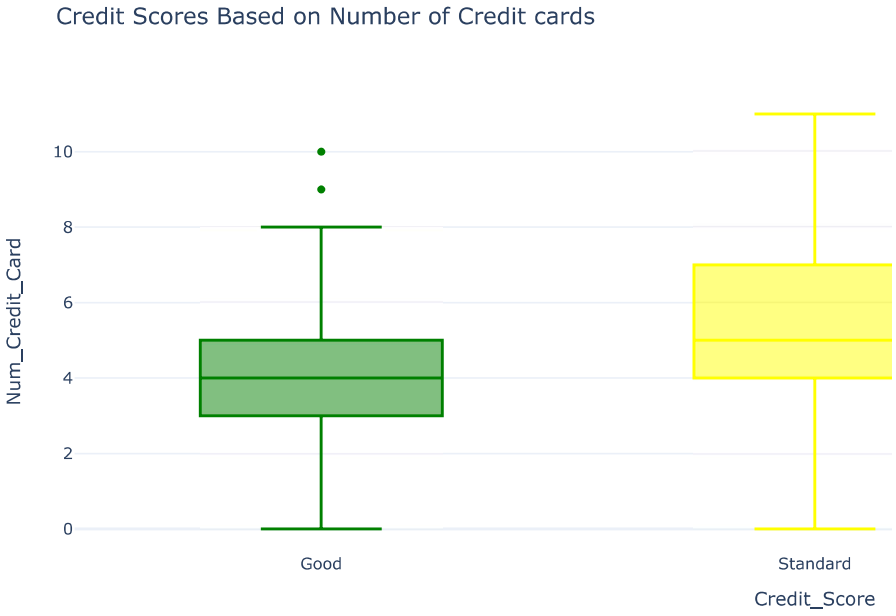



```
fig = px.box(data,
             x="Credit_Score",
             y="Num_Bank_Accounts",
             color="Credit_Score",
             title="Credit Scores Based on Number of Bank Accounts",
             color_discrete_map={'Poor':'red',
```

```
                                'Standard':'yellow',
                                'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```

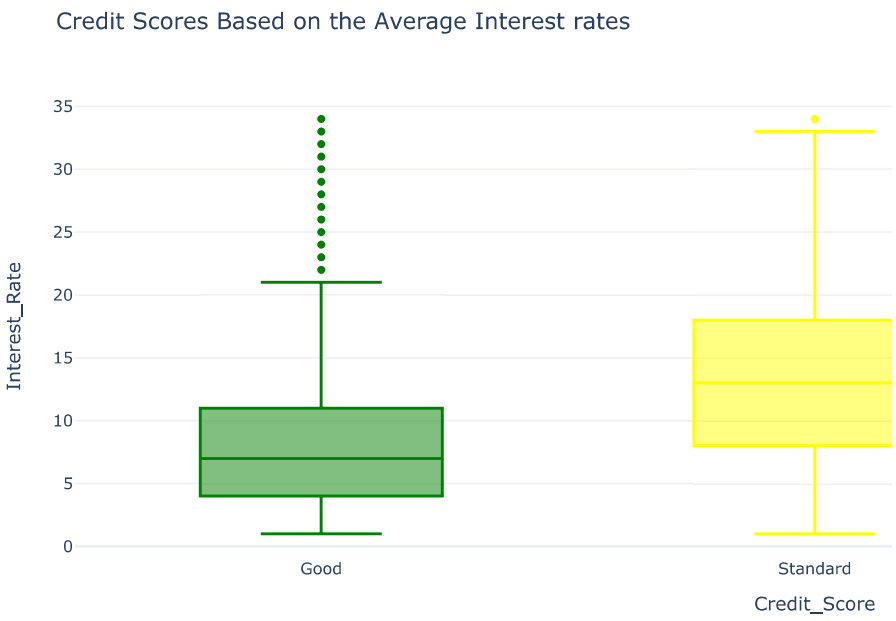



```
fig = px.box(data,
             x="Credit_Score",
             y="Num_Credit_Card",
             color="Credit_Score",
             title="Credit Scores Based on Number of Credit cards",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
        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

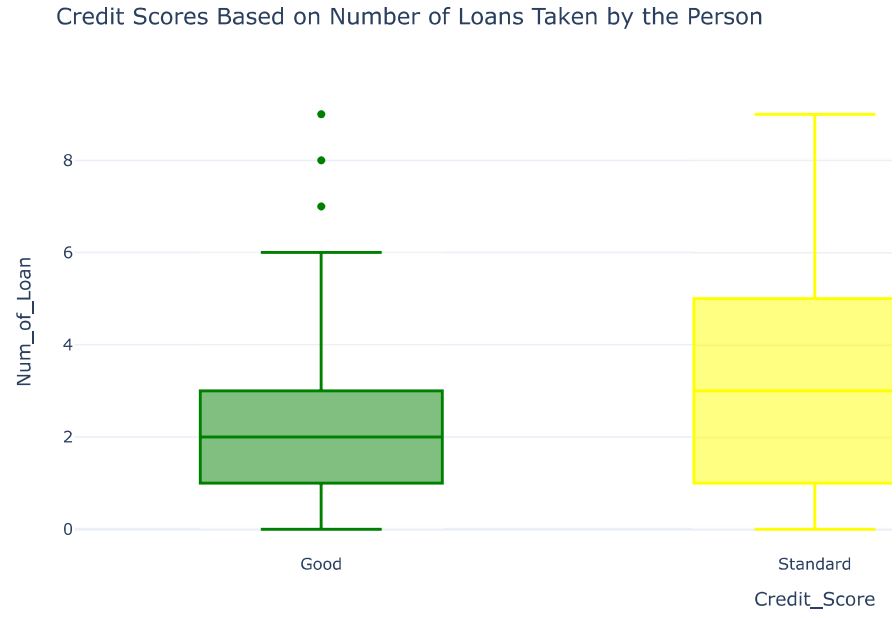



```
fig = px.box(data,
             x="Credit_Score",
             y="Interest_Rate",
             color="Credit_Score",
             title="Credit Scores Based on the Average Interest rates",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
      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

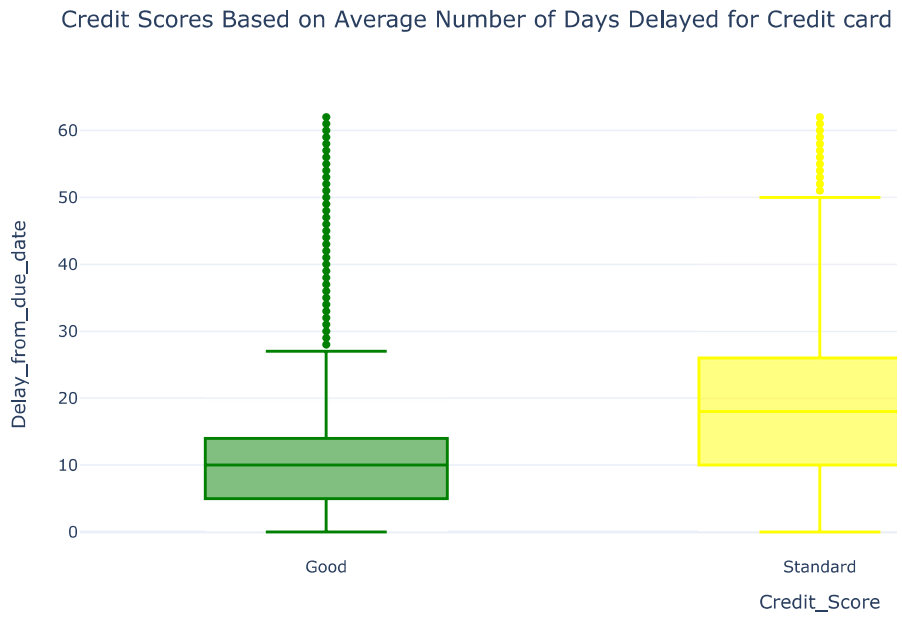
```
fig.update_traces(quartilemethod="exclusive")
fig.show()
```



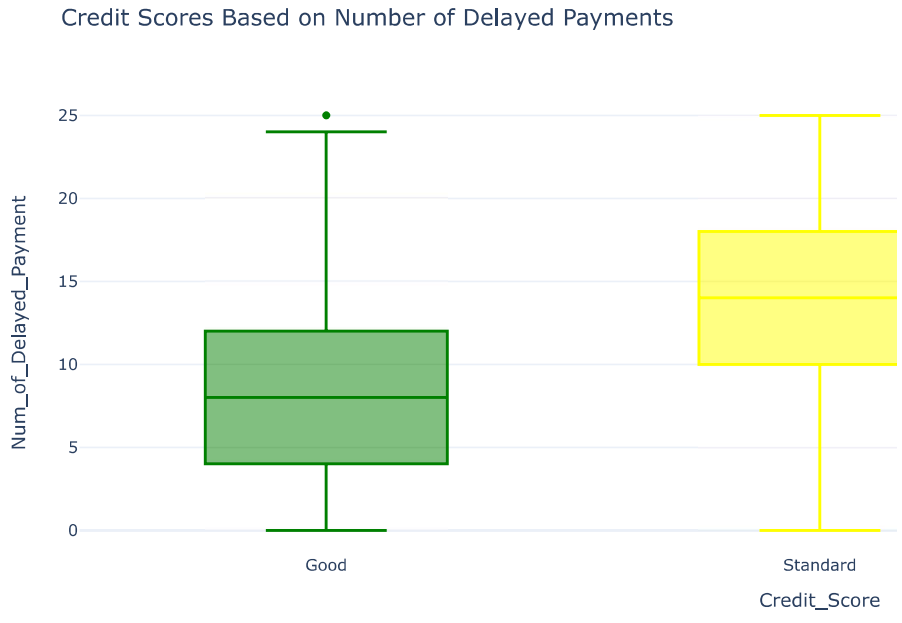

```
fig = px.box(data,
             x="Credit_Score",
             y="Num_of_Loan",
             color="Credit_Score",
             title="Credit Scores Based on Number of Loans Taken by the Person",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',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



```
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 Credit card Payments",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',
                                 'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```

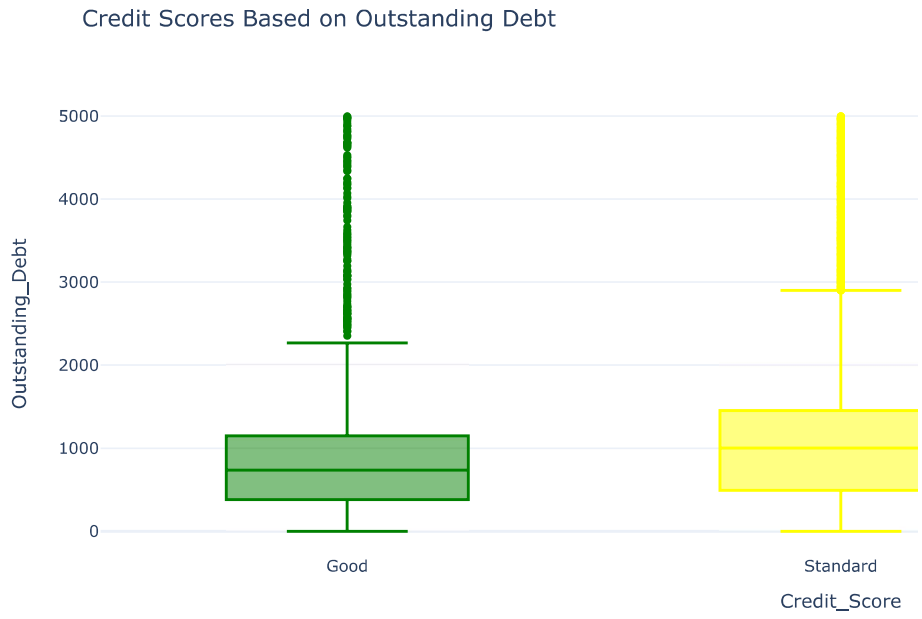



```
fig = px.box(data,
             x="Credit_Score",
             y="Num_of_Delayed_Payment",
             color="Credit_Score",
             title="Credit Scores Based on Number of Delayed Payments",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',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

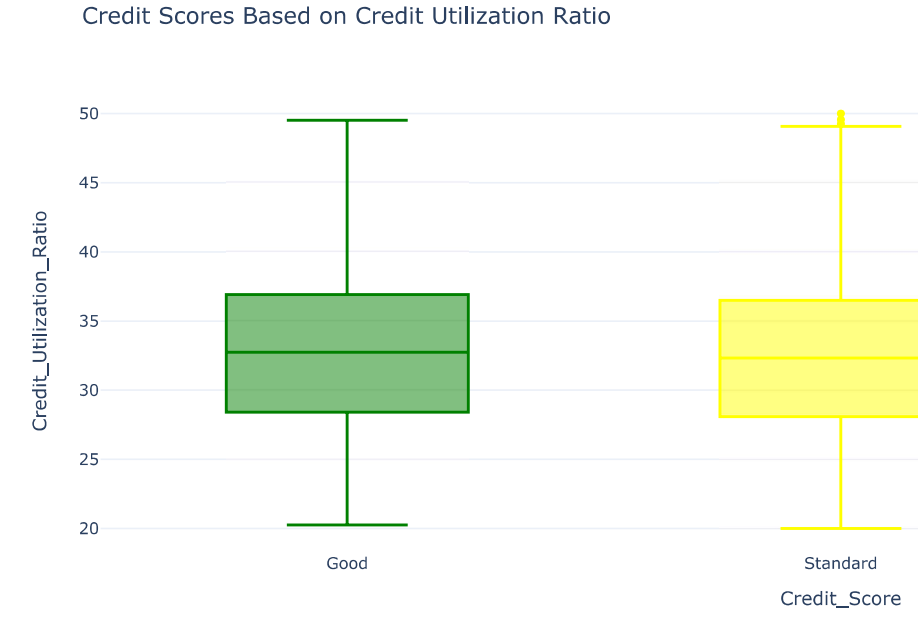



```
fig = px.box(data,
             x="Credit_Score",
```

```
            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()
```

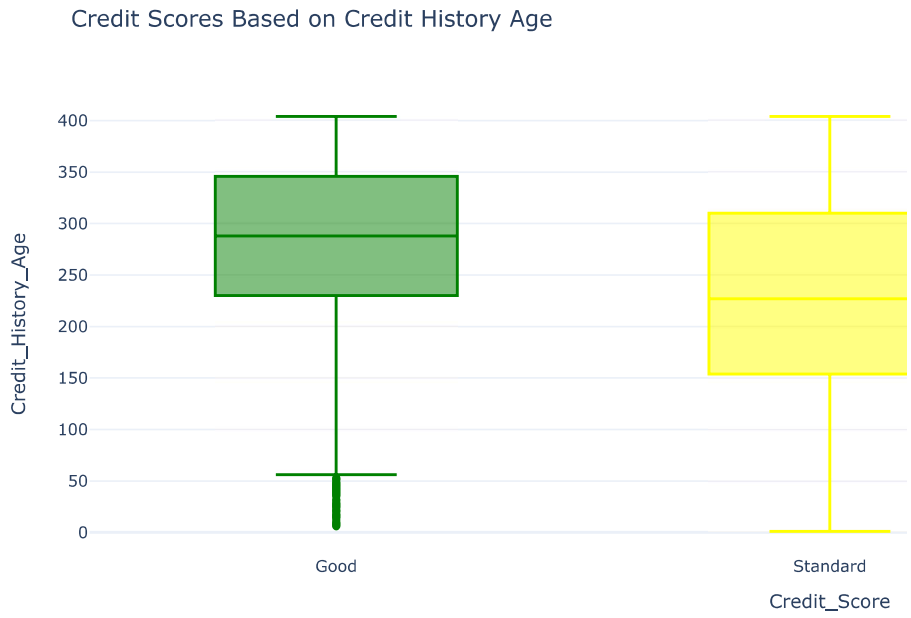



```
fig = px.box(data,
            x="Credit_Score",
            y="Credit_Utilization_Ratio",
            color="Credit_Score",
            title="Credit Scores Based on Credit Utilization Ratio",
            color_discrete_map={'Poor':'red',
                                'Standard':'yellow',
   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

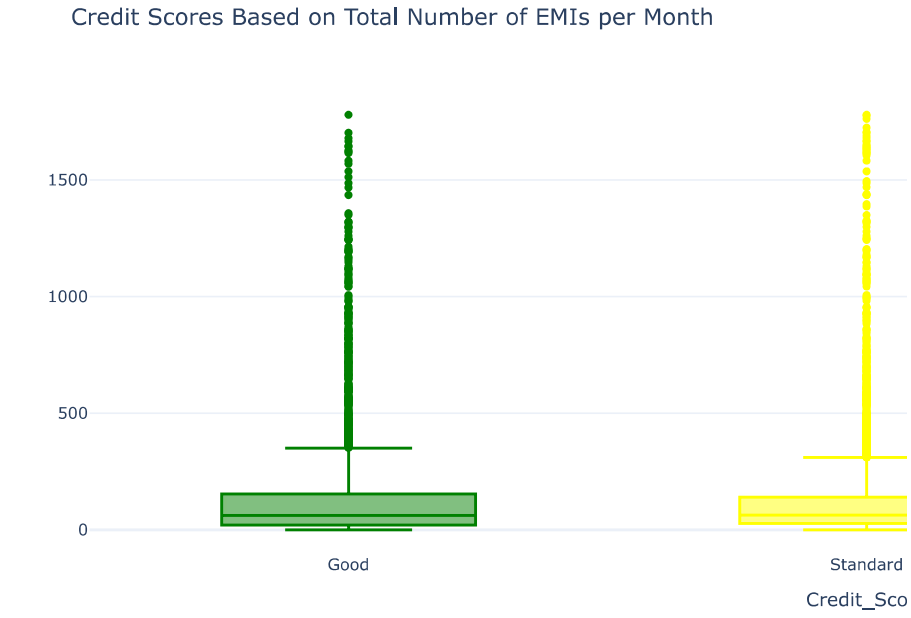



```
fig = px.box(data,
            x="Credit_Score",
            y="Credit_History_Age",
            color="Credit_Score",
```

```
            title="Credit Scores Based on Credit History Age",
            color_discrete_map={'Poor':'red',
                                'Standard':'yellow',
                                'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```

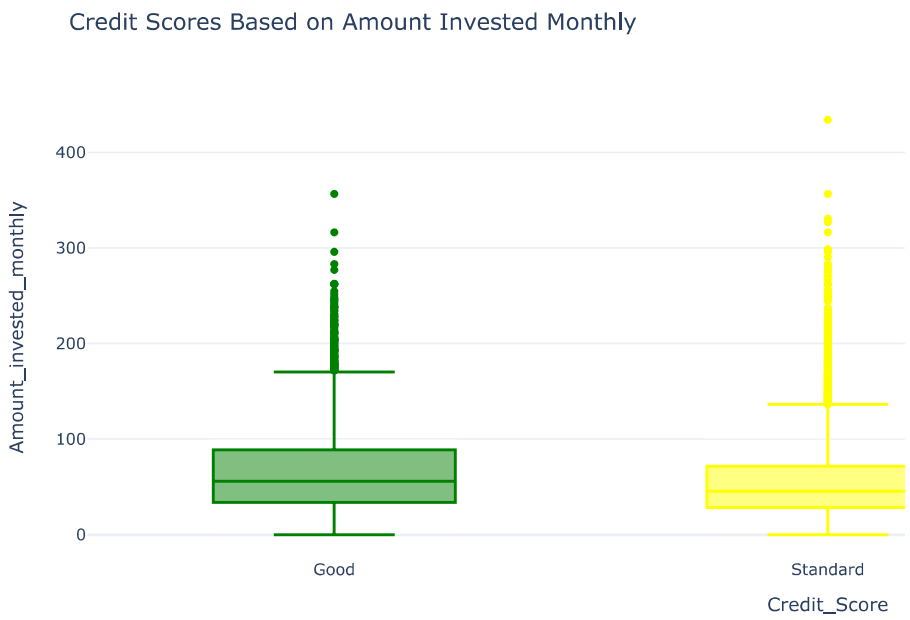



```
fig = px.box(data,
             x="Credit_Score",
             y="Total_EMI_per_month",
             color="Credit_Score",
             title="Credit Scores Based on Total Number of EMIs per Month",
             color_discrete_map={'Poor':'red',
                                 'Standard':'yellow',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

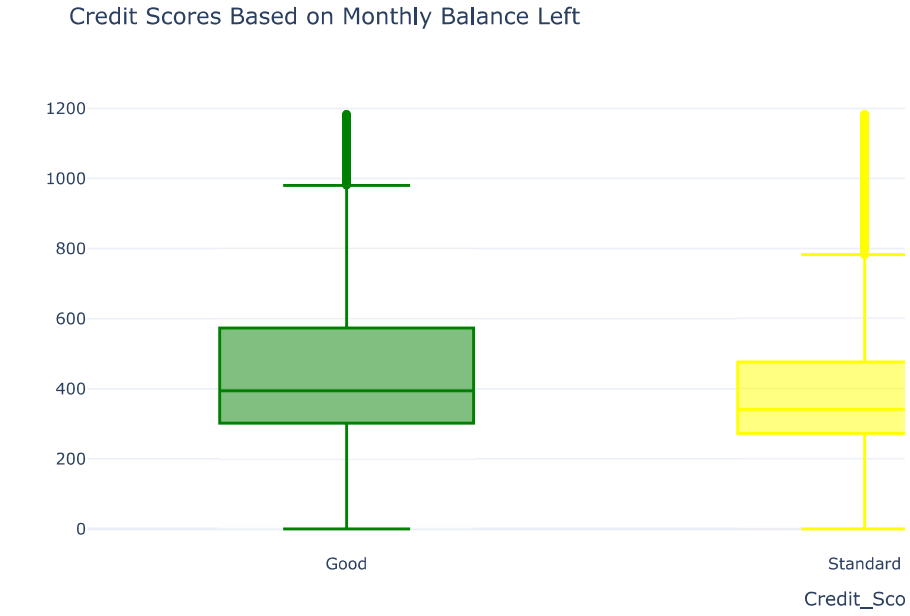



```
fig = px.box(data,
             x="Credit_Score",
             y="Amount_invested_monthly",
             color="Credit_Score",
             title="Credit Scores Based on Amount Invested Monthly",
             color_discrete_map={'Poor':'red',
```

```
                                'Standard':'yellow',
                                'Good':'green'})
fig.update_traces(quartilemethod="exclusive")
fig.show()
```




```
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()
```




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

```
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']])


xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.33,
                                                random_state=42)
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(xtrain, ytrain)
```

```
<ipython-input-68-ea8253e80d41>:6: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to
```

▾ RandomForestClassifier
RandomForestClassifier()

```
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("Average number of days delayed by the person: "))
h = float(input("Number of delayed payments: "))
i = input("Credit Mix (Bad: 0, Standard: 1, Good: 3) : ")
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]])
```

```
Credit Score Prediction :
Annual Income: 120000
Monthly Inhand Salary: 12000
Number of Bank Accounts: 2
Number of Credit cards: 5
Interest rate: 14
Number of Loans: 5
Average number of days delayed by the person: 5
Number of delayed payments: 4
Credit Mix (Bad: 0, Standard: 1, Good: 3) : 1
Outstanding Debt: 580000
Credit History Age: 5
Monthly Balance: 400
```

```
print("Predicted Credit Score = ", model.predict(features))
```

```
Predicted Credit Score =  ['Standard']
```

✓ 0s completed at 12:28 PM