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

In [2]:
```
df=pd.read_csv(r"C:\Users\Ajith\Desktop\train.csv")
```

In [3]:
```
df
```

Out[3]:

	ID	Customer_ID	Month	Name	Age	SSN	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts
0	5634	3392	1	Aaron Maashoh	23.0	821000265.0	Scientist	19114.12	1824.843333	3.0
1	5635	3392	2	Aaron Maashoh	23.0	821000265.0	Scientist	19114.12	1824.843333	3.0
2	5636	3392	3	Aaron Maashoh	23.0	821000265.0	Scientist	19114.12	1824.843333	3.0
3	5637	3392	4	Aaron Maashoh	23.0	821000265.0	Scientist	19114.12	1824.843333	3.0
4	5638	3392	5	Aaron Maashoh	23.0	821000265.0	Scientist	19114.12	1824.843333	3.0
...	...	...	...	...	...	...	...	...	...	...
99995	155625	37932	4	Nicks	25.0	78735990.0	Mechanic	39628.99	3359.415833	4.0
99996	155626	37932	5	Nicks	25.0	78735990.0	Mechanic	39628.99	3359.415833	4.0
99997	155627	37932	6	Nicks	25.0	78735990.0	Mechanic	39628.99	3359.415833	4.0
99998	155628	37932	7	Nicks	25.0	78735990.0	Mechanic	39628.99	3359.415833	4.0
99999	155629	37932	8	Nicks	25.0	78735990.0	Mechanic	39628.99	3359.415833	4.0

100000 rows × 28 columns

In [4]:
```
df.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 [5]:
```
df.describe()
```

Out[5]:

	ID	Customer_ID	Month	Age	SSN	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accou
count	100000.000000	100000.000000	100000.000000	100000.000000	1.000000e+05	100000.000000	100000.000000	100000.0000
mean	80631.500000	25982.666640	4.500000	33.316340	5.004617e+08	50505.123449	4197.270835	5.3688
std	43301.486619	14340.543051	2.291299	10.764812	2.908267e+08	38299.422093	3186.432497	2.5933
min	5634.000000	1006.000000	1.000000	14.000000	8.134900e+04	7005.930000	303.645417	0.0000
25%	43132.750000	13664.500000	2.750000	24.000000	2.451686e+08	19342.972500	1626.594167	3.0000
50%	80631.500000	25777.000000	4.500000	33.000000	5.006886e+08	36999.705000	3095.905000	5.0000
75%	118130.250000	38385.000000	6.250000	42.000000	7.560027e+08	71683.470000	5957.715000	7.0000
max	155629.000000	50999.000000	8.000000	56.000000	9.999934e+08	179987.280000	15204.633333	11.0000

8 rows × 21 columns

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
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 [7]:
```
df.duplicated().sum()
```

Out[7]:
```
0
```

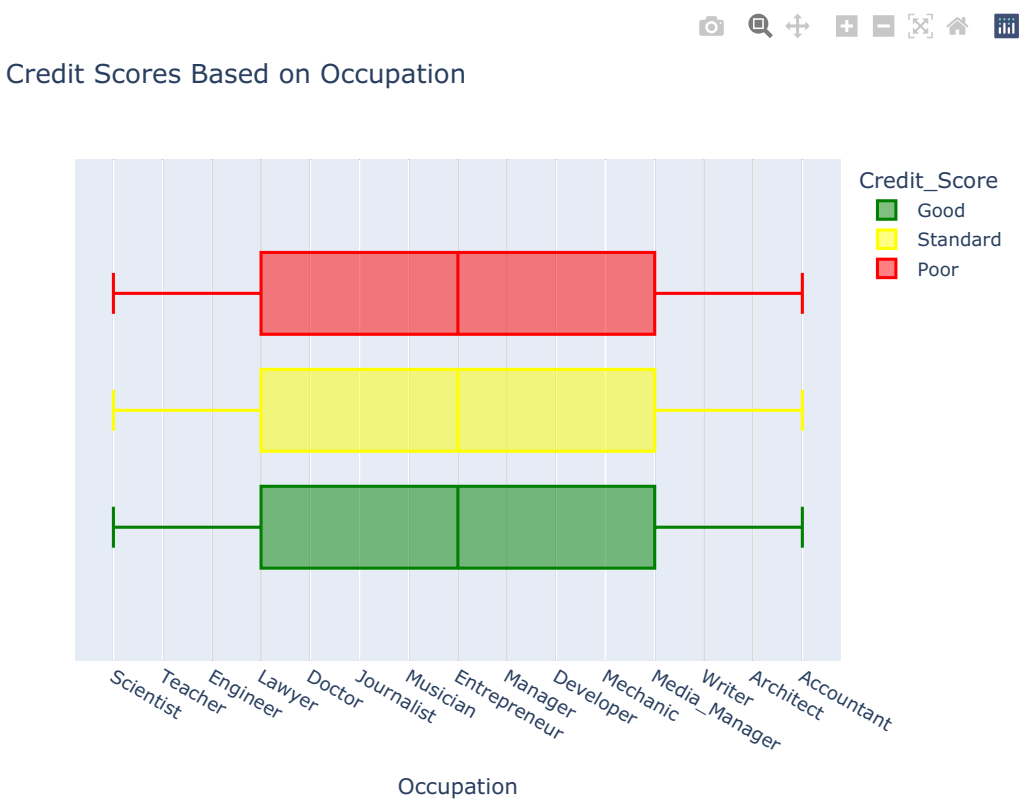
In [8]:
```
df.columns
```

Out[8]:
```
Index(['ID', 'Customer_ID', 'Month', 'Name', 'Age', 'SSN', 'Occupation',
       'Annual_Income', 'Monthly_Inhand_Salary', 'Num_Bank_Accounts',
       'Num_Credit_Card', 'Interest_Rate', 'Num_of_Loan', 'Type_of_Loan',
       'Delay_from_due_date', 'Num_of_Delayed_Payment', 'Changed_Credit_Limit',
       'Num_Credit_Inquiries', 'Credit_Mix', 'Outstanding_Debt',
       'Credit_Utilization_Ratio', 'Credit_History_Age',
       'Payment_of_Min_Amount', 'Total_EMI_per_month',
       'Amount_invested_monthly', 'Payment_Behaviour', 'Monthly_Balance',
       'Credit_Score'],
      dtype='object')
```

In [9]:
```
df["Credit_Score"].value_counts()
```

Out[9]:
```
Standard    53174
Poor        28998
Good        17828
Name: Credit_Score, dtype: int64
```

In [10]:
```
plt=px.box(df,
       x="Occupation",
       color="Credit_Score",
       title="Credit Scores Based on Occupation",
       color_discrete_map={'Poor':'red',
                           'Standard':'yellow',
                           'Good':'green'})
plt.show()
```



In [11]:
```
plt = px.box(df,
             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'})
plt.show()
```

In [12]:
```
plt = px.box(df,
             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'})
plt.show()
```

In [13]:
```
plt = px.box(df,
             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'})
plt.show()
```

In [14]:
```
plt = px.box(df,
             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'})
plt.show()
```

In [15]:
```
plt = px.box(df,
             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'})
plt.show()
```

In [16]:
```
 plt = px.box(df,
             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'})
plt.show()
```

In [17]:
```
plt = px.box(df,
             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'})
plt.show()
```

In [18]:
```
plt = px.box(df,
             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'})
plt.show()
```

In [19]:
```
plt = px.box(df,
             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'})
plt.show()
```

In [20]:
```
plt = px.box(df,
             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'})
plt.show()
```

In [21]:
```
plt = px.box(df,
             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'})
plt.show()
```

In [22]:
```
plt = px.box(df,
             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'})
plt.show()
```

In [23]:
```
plt = px.box(df,
             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'})
plt.show()
```

In [24]:
```
plt = px.box(df,
             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'})
plt.show()
```

In [25]:
```
df["Credit_Mix"] = df["Credit_Mix"].map({"Standard": 1,
                                         "Good": 2,
                                         "Bad": 0})
```

In [26]:
```
from sklearn.model_selection import train_test_split
x = np.array(df[["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(df[["Credit_Score"]])
```

In [27]:
```
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)
```

```
C:\Users\Ajith\AppData\Local\Temp\ipykernel_11444\2049170333.py:6: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for ex
ample using ravel().
```

Out[27]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [28]:
```
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]])
print("Predicted Credit Score = ", model.predict(features))
```

```
Credit Score Prediction :
Annual Income: 700000
Monthly Inhand Salary: 55000
Number of Bank Accounts: 3
Number of Credit cards: 0
Interest rate: 0
Number of Loans: 0
Average number of days delayed by the person: 0
Number of delayed payments: 0
Credit Mix (Bad: 0, Standard: 1, Good: 3) : 3
Outstanding Debt: 250
Credit History Age: 200
Monthly Balance: 5000
Predicted Credit Score =  ['Good']
```

In [ ]: