```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stas

df=pd.read_csv("AutoInsurance.csv")
df.head()
```

```
{"type":"dataframe","variable_name":"df"}
```

```
print("The number of rows:",df.shape[0])
print("The number of columns:",df.shape[1])
```

```
The number of rows: 9134
The number of columns: 24
```

```
df.rename(columns={'Customer Lifetime Value':'CLV'},inplace=True)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9134 entries, 0 to 9133
Data columns (total 24 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Customer                       9134 non-null   object 
 1   State                          9134 non-null   object 
 2   CLV                            9134 non-null   float64
 3   Response                       9134 non-null   object 
 4   Coverage                       9134 non-null   object 
 5   Education                      9134 non-null   object 
 6   Effective To Date              9134 non-null   object 
 7   EmploymentStatus               9134 non-null   object 
 8   Gender                         9134 non-null   object 
 9   Income                         9134 non-null   int64  
 10  Location Code                  9134 non-null   object 
 11  Marital Status                 9134 non-null   object 
 12  Monthly Premium Auto           9134 non-null   int64  
 13  Months Since Last Claim        9134 non-null   int64  
 14  Months Since Policy Inception  9134 non-null   int64  
 15  Number of Open Complaints      9134 non-null   int64  
 16  Number of Policies             9134 non-null   int64  
 17  Policy Type                    9134 non-null   object 
 18  Policy                         9134 non-null   object 
 19  Renew Offer Type               9134 non-null   object 
 20  Sales Channel                  9134 non-null   object 
 21  Total Claim Amount             9134 non-null   float64
 22  Vehicle Class                  9134 non-null   object 
 23  Vehicle Size                   9134 non-null   object
```

```
dtypes: float64(2), int64(6), object(16)
memory usage: 1.7+ MB
```

```
numerical_cols=df.select_dtypes(include=("int64","float64"))
```

```
numerical_cols.columns
```

```
Index(['CLV', 'Income', 'Monthly Premium Auto', 'Months Since Last
Claim',
       'Months Since Policy Inception', 'Number of Open Complaints',
       'Number of Policies', 'Total Claim Amount'],
      dtype='object')
```

```
numerical_cols=numerical_cols.drop(["Number of Policies","Number of
Open Complaints"],axis=1)
#These columns(Number of Policies and Number of Open Complaints) have
numerical/int values but they are actually working like cateogies so
we drop them from numerical column for noww but later we will add it
in categorical column
```

WE ARE GOING TO FIND HOW MANY NUMERICAL AND HOW MANY CATEGORICAL COLUMN WE HAVE WE ARE GOING TO VISULAIZE IT SEPERATELY

```
numerical_cols.describe()
```


```
{"summary":"{\n  \"name\": \"numerical_cols\",\n  \"rows\": 8,\n
\"fields\": [\n    {\n      \"column\": \"CLV\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
27318.46301502897,\n        \"min\": 1898.007675,\n        \"max\":
83325.38119,\n        \"num_unique_values\": 8,\n        \"samples\":
[\n          8004.940474987081,\n          5780.182197,\n
9134.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Income\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 34042.189450956816,\n        \"min\": 0.0,\n        \"max\":
99981.0,\n        \"num_unique_values\": 7,\n        \"samples\": [\n
9134.0,\n          37657.38000875848,\n          62320.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Monthly Premium Auto\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3192.6791116121503,\n        \"min\": 34.40796737178653,\n
\"max\": 9134.0,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          93.21929056273265,\n          83.0,\n
9134.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Months Since Last Claim\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 3224.1633930546786,\n
\"min\": 0.0,\n        \"max\": 9134.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
15.097000218962119,\n          14.0,\n          9134.0\n        ],\n
```

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Months Since Policy Inception\",\n\"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 3213.4382667159475,\n   \"min\": 0.0,\n   \"max\": 9134.0,\n\"num_unique_values\": 8,\n   \"samples\": [\n48.064593825268226,\n   48.0,\n   9134.0\n   ],\n\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Total Claim Amount\",\n\"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 3122.497149921506,\n   \"min\": 0.099007,\n   \"max\": 9134.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n434.0887943128969,\n   383.94543350000004,\n   9134.0\n],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n}\n   }\n]\n}","type":"dataframe"}

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

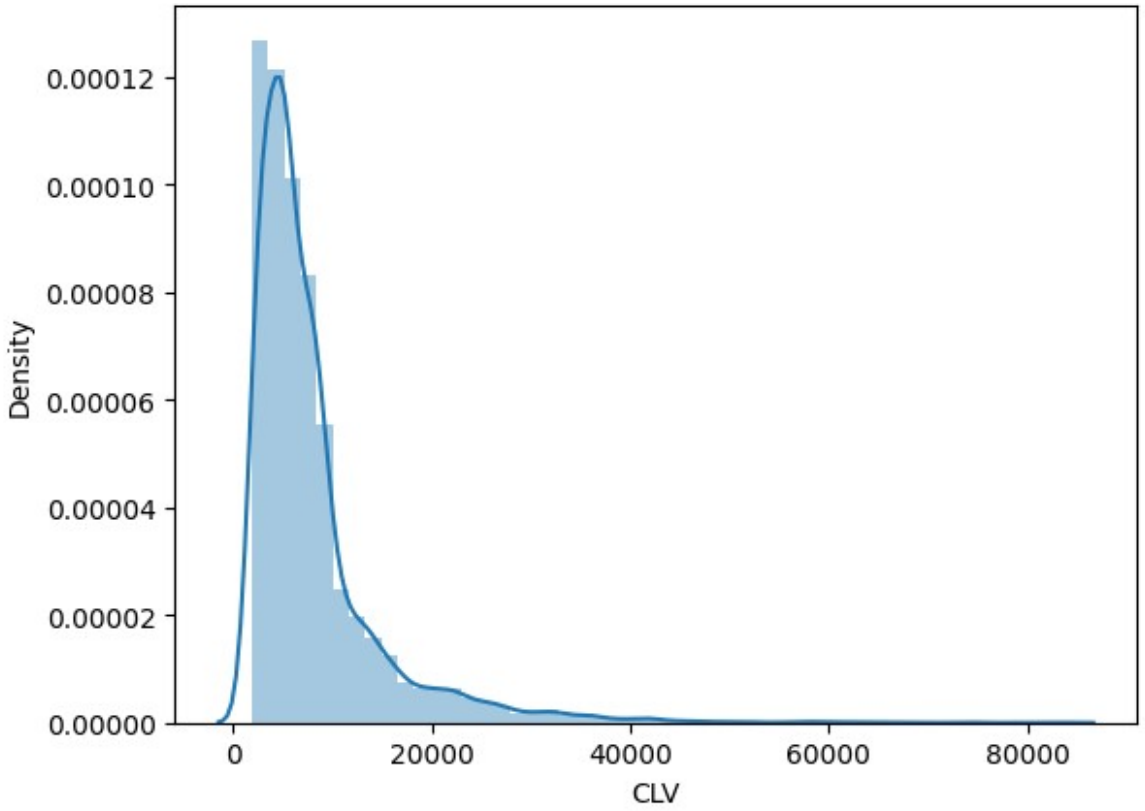
```
Customer                         0
State                            0
CLV                              0
Response                         0
Coverage                         0
Education                        0
Effective To Date                0
EmploymentStatus                 0
Gender                           0
Income                           0
Location Code                    0
Marital Status                   0
Monthly Premium Auto             0
Months Since Last Claim          0
Months Since Policy Inception    0
Number of Open Complaints        0
Number of Policies               0
Policy Type                      0
Policy                           0
Renew Offer Type                 0
Sales Channel                    0
Total Claim Amount               0
Vehicle Class                    0
Vehicle Size                     0
dtype: int64
```

EDA

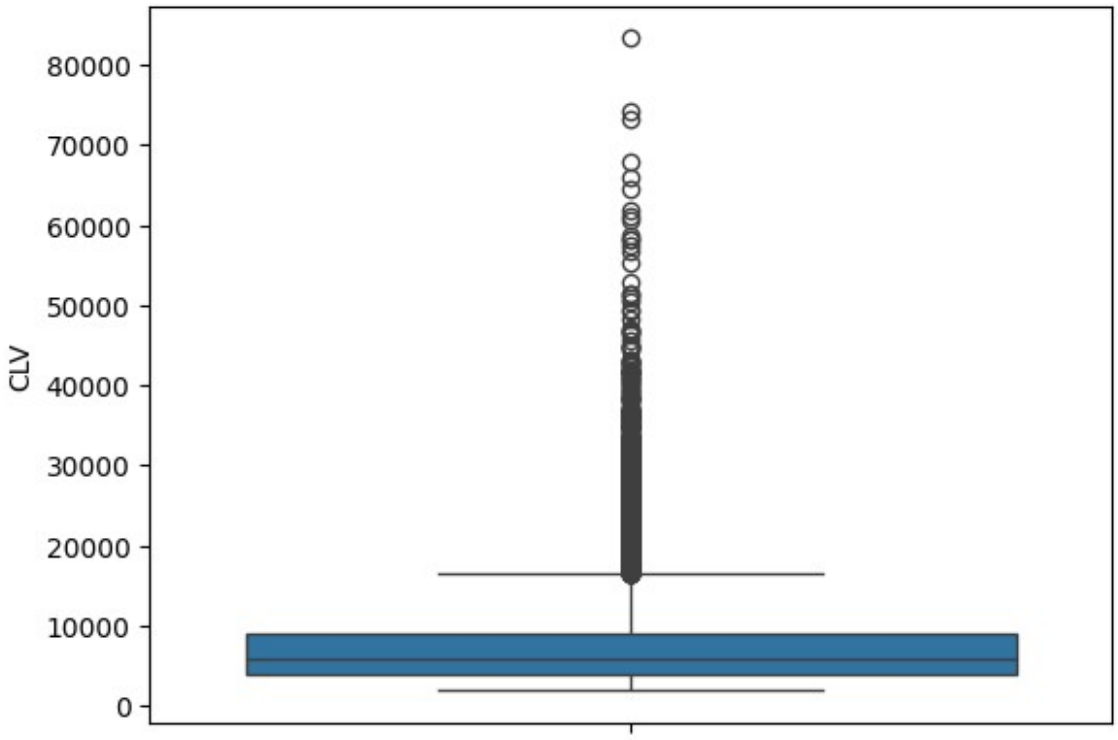
NUMERICAL FEATURES

```
sns.distplot(df["CLV"])
plt.show()
```

```
<ipython-input-169-64d95f553d31>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["CLV"])
```



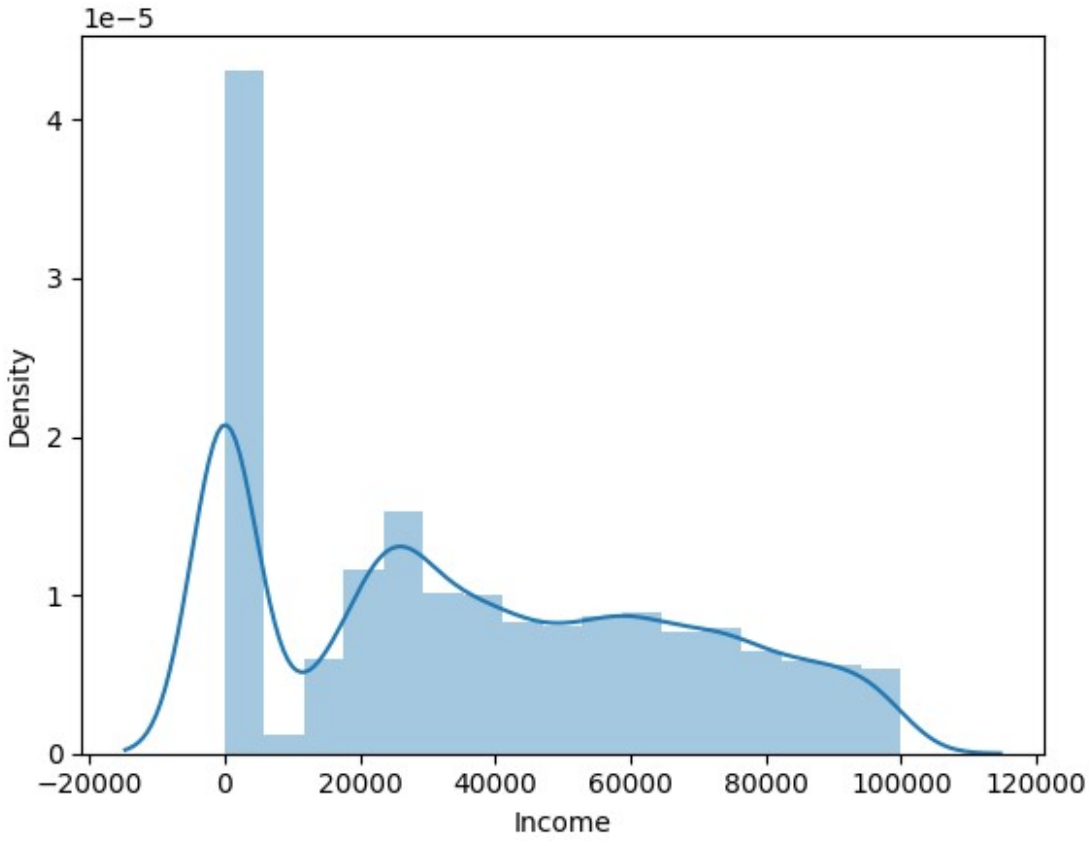

```
sns.boxplot(df["CLV"])
plt.show()
```



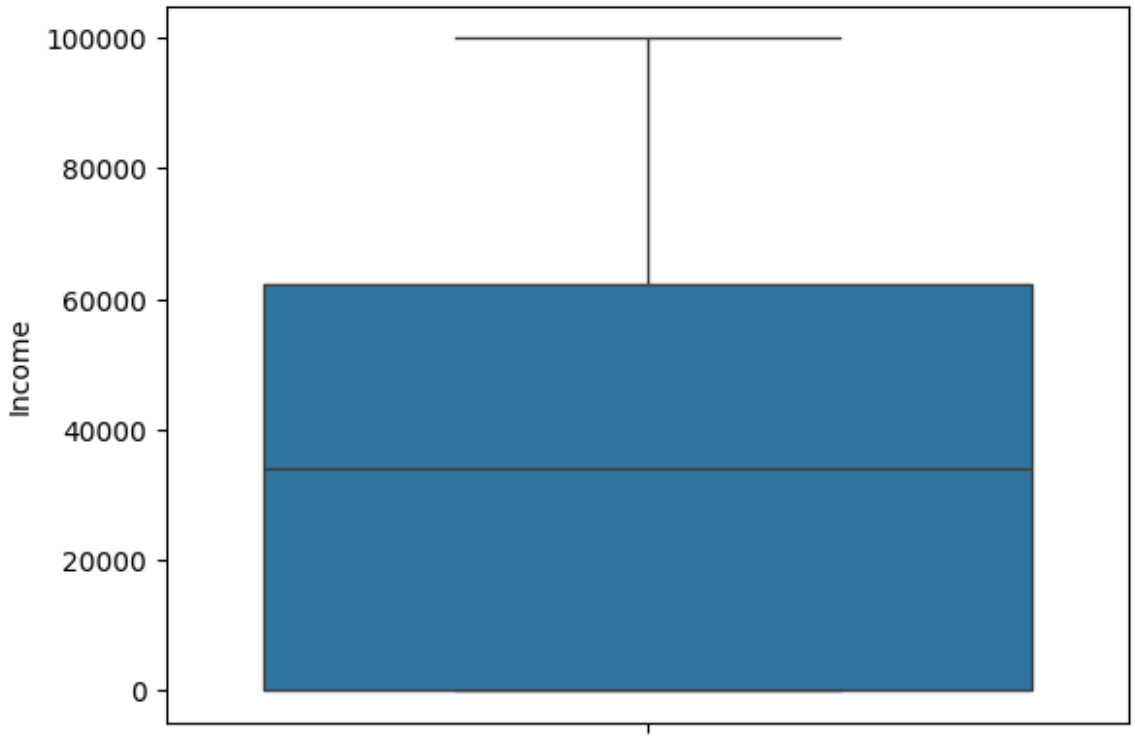
UNIVERIATE ANALYSIS

```
sns.distplot(df["Income"])
plt.show()
```

```
<ipython-input-171-ffd47b6b5651>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Income"])
```

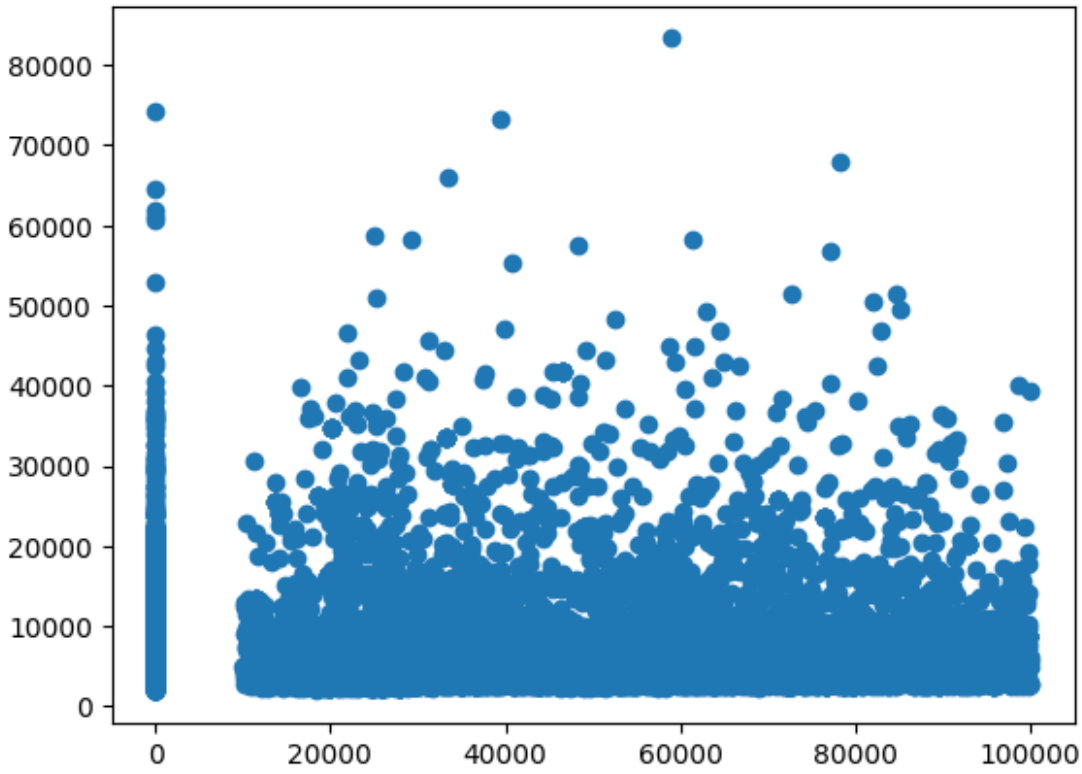



```
sns.boxplot(df["Income"])
plt.show()
```



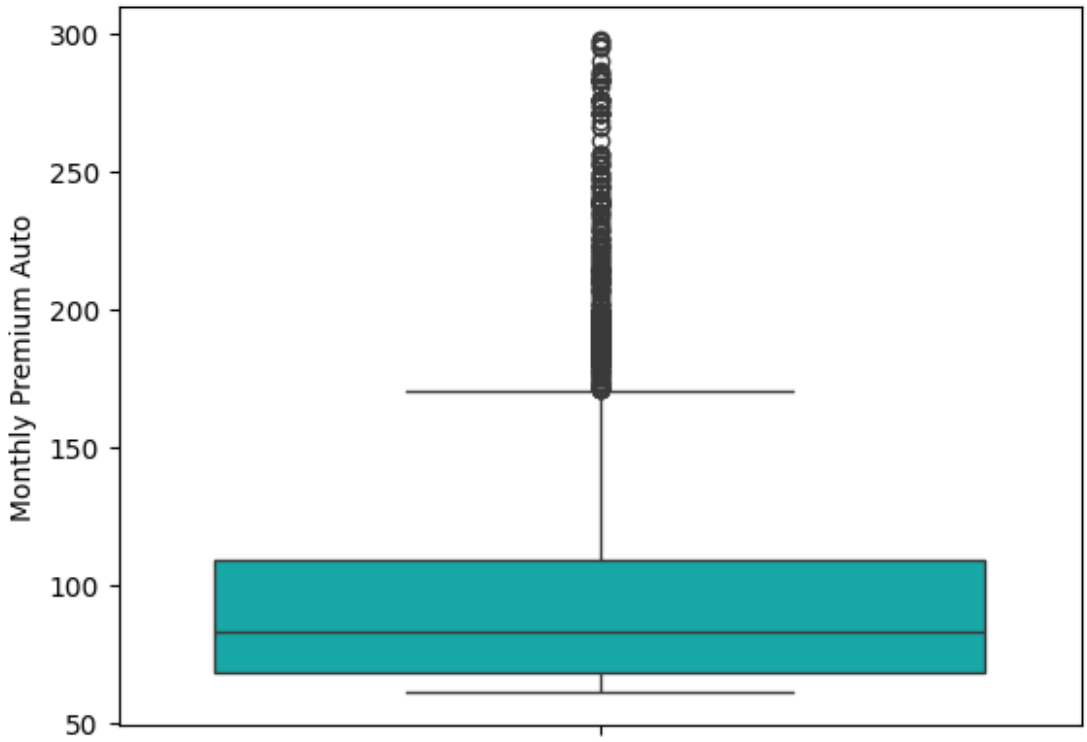
BIVARIATE ANALYSIS

```
plt.scatter(df["Income"],df["CLV"])
plt.show()
```



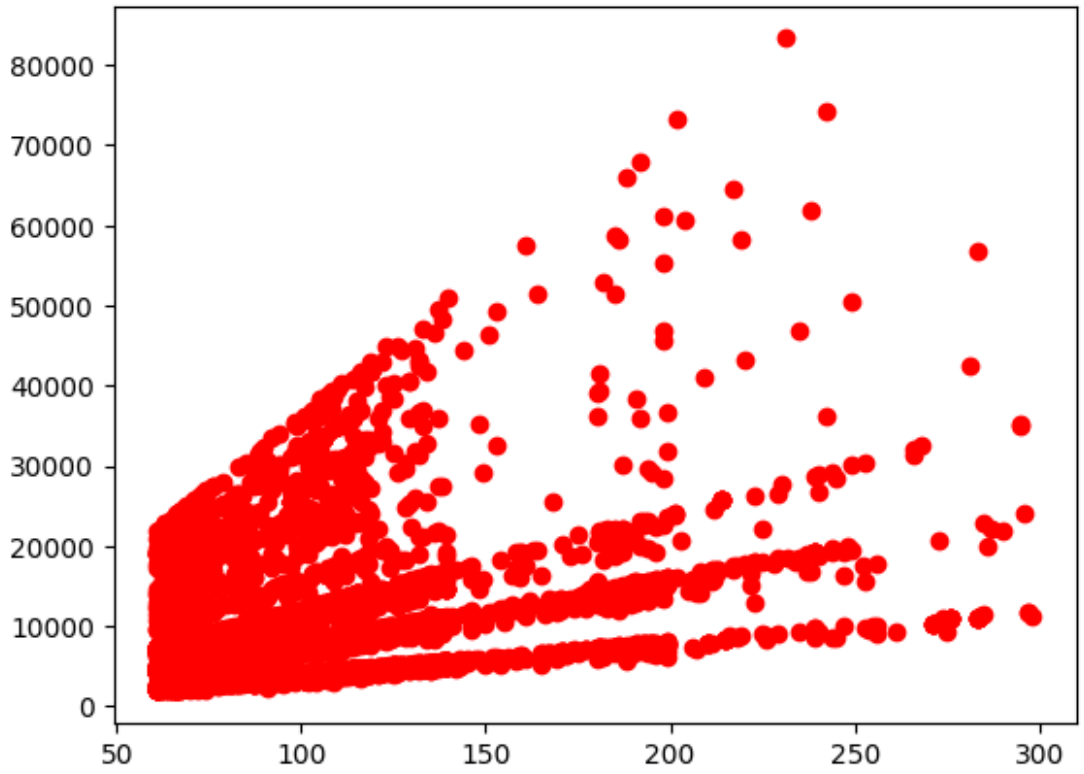
MONTHLY PREMIUM AUTO

```
sns.boxplot(df["Monthly Premium Auto"],color="c")
plt.show()
```



BIVARIATE ANALYSIS

```
plt.scatter(df["Monthly Premium Auto"],df["CLV"],color="r")
plt.show()
```

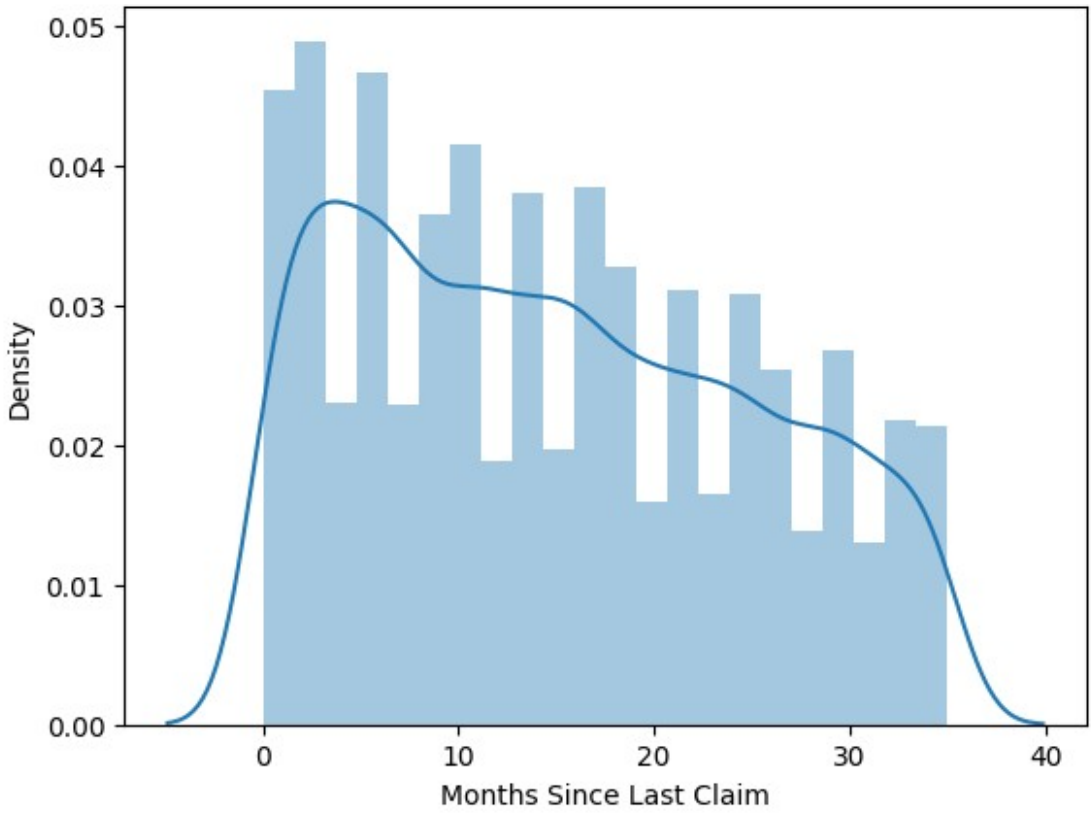


There is a relationship between income and clv

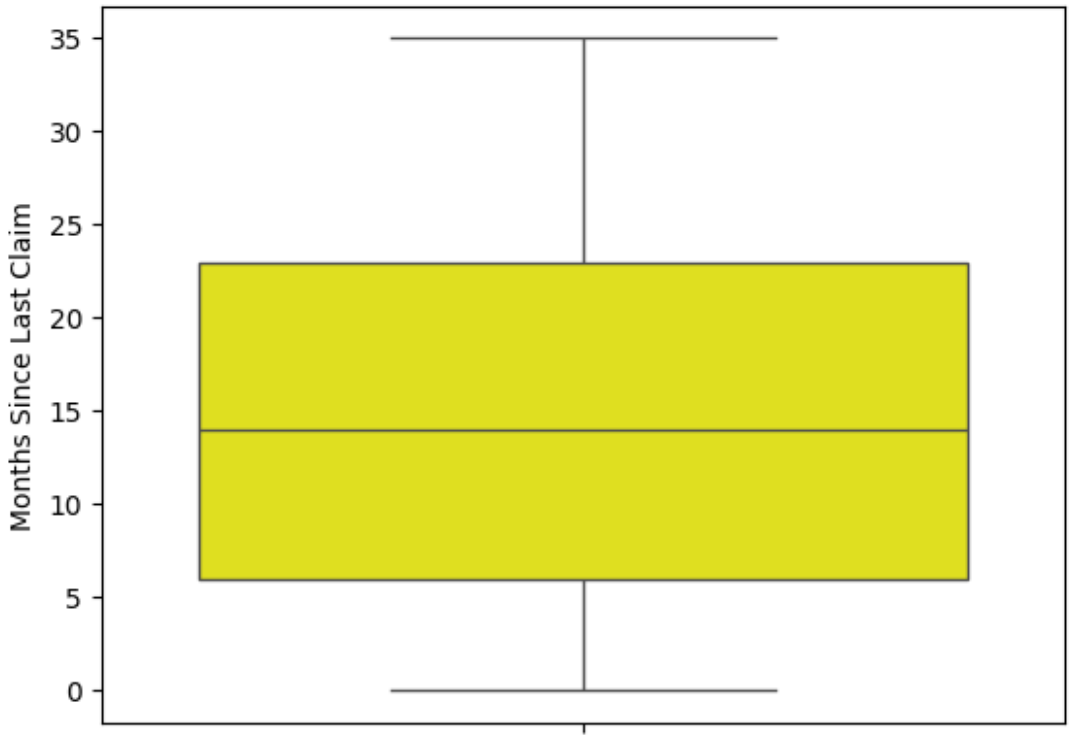
Months since Last Claim

```
sns.distplot(df["Months Since Last Claim"])
plt.show()
```

```
<ipython-input-176-a5d715d00e85>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Months Since Last Claim"])
```

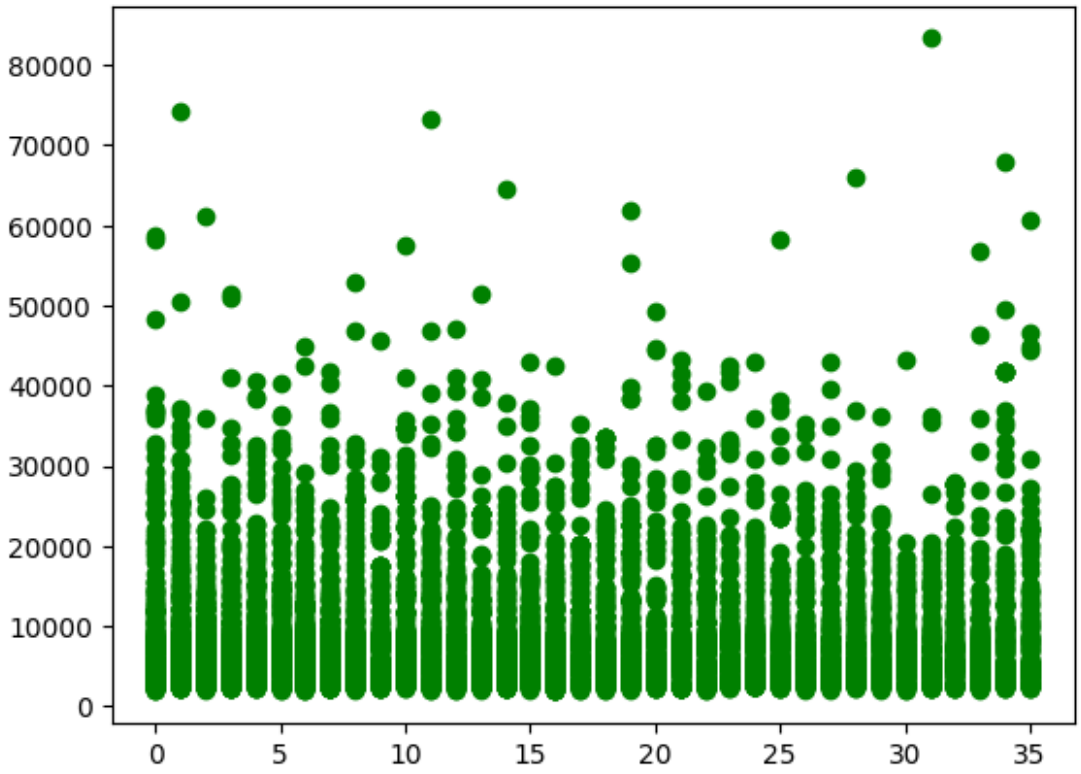



```
sns.boxplot(df["Months Since Last Claim"],color="Yellow")
plt.show()
```



BIVARIATE ANALYSIS

```
plt.scatter(df["Months Since Last Claim"],df["CLV"],color="g")
plt.show()
```

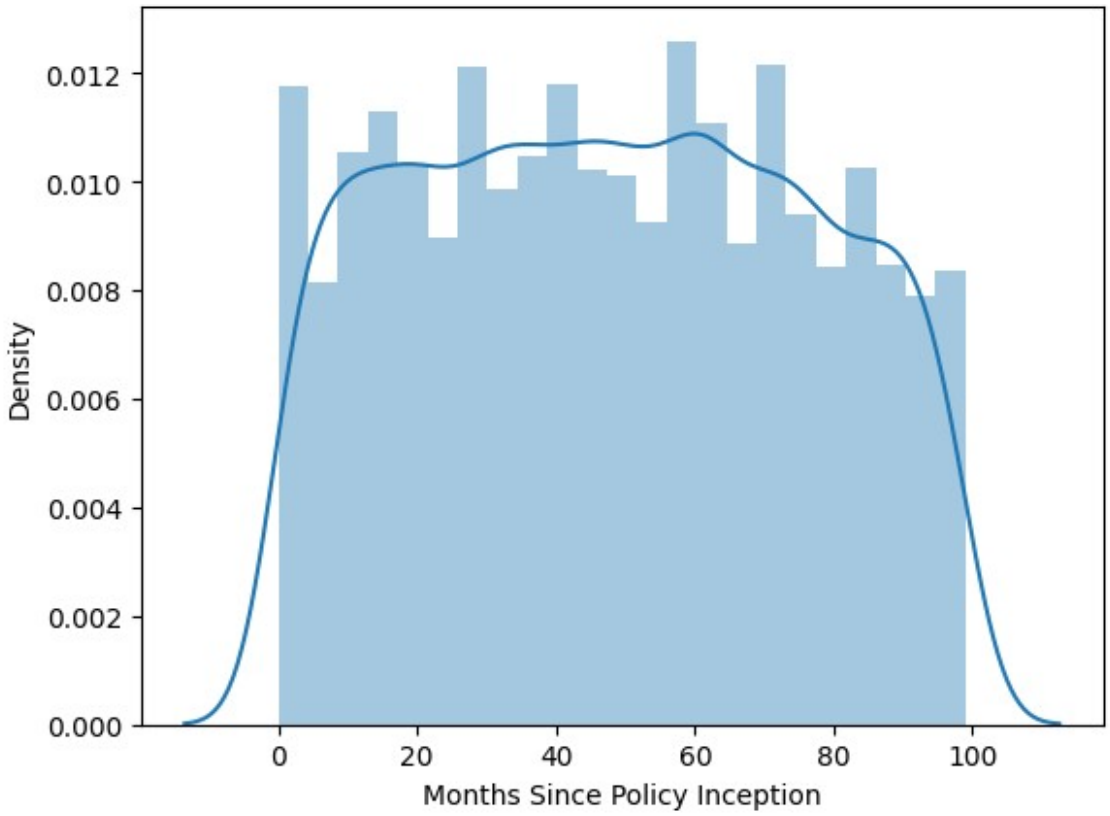


There is no linear relationship

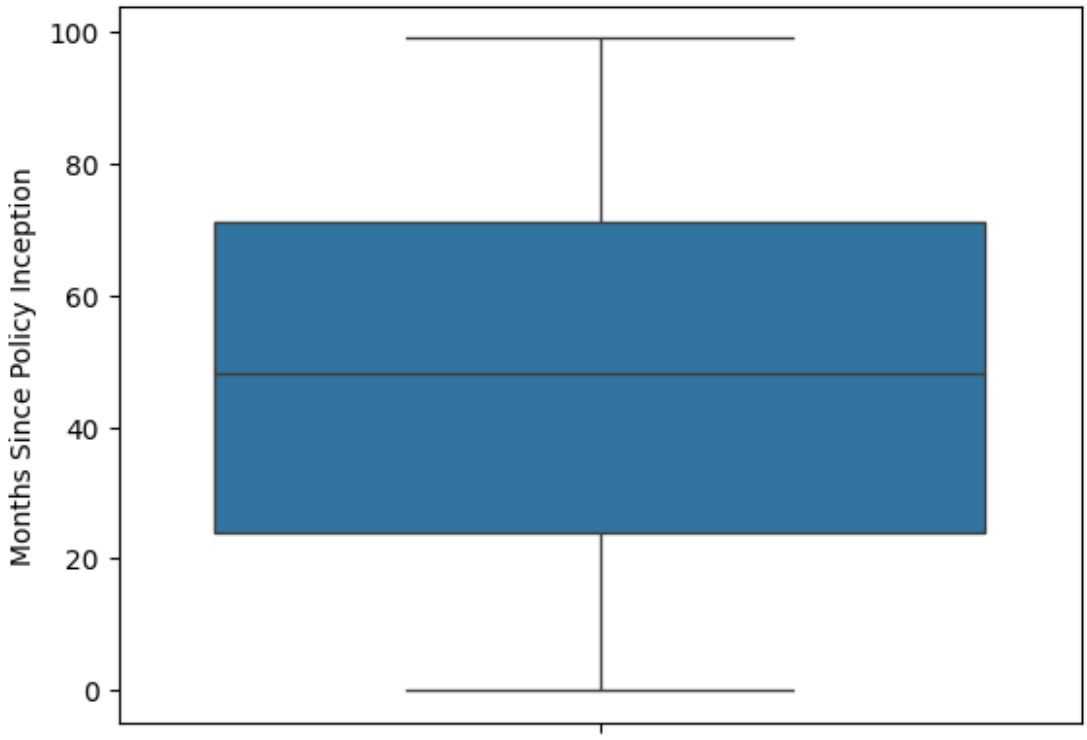
MONTHS SINCE POLICY INCEPTION

```
sns.distplot(df["Months Since Policy Inception"])
plt.show()
```

```
<ipython-input-179-16ddb0b3c524>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Months Since Policy Inception"])
```



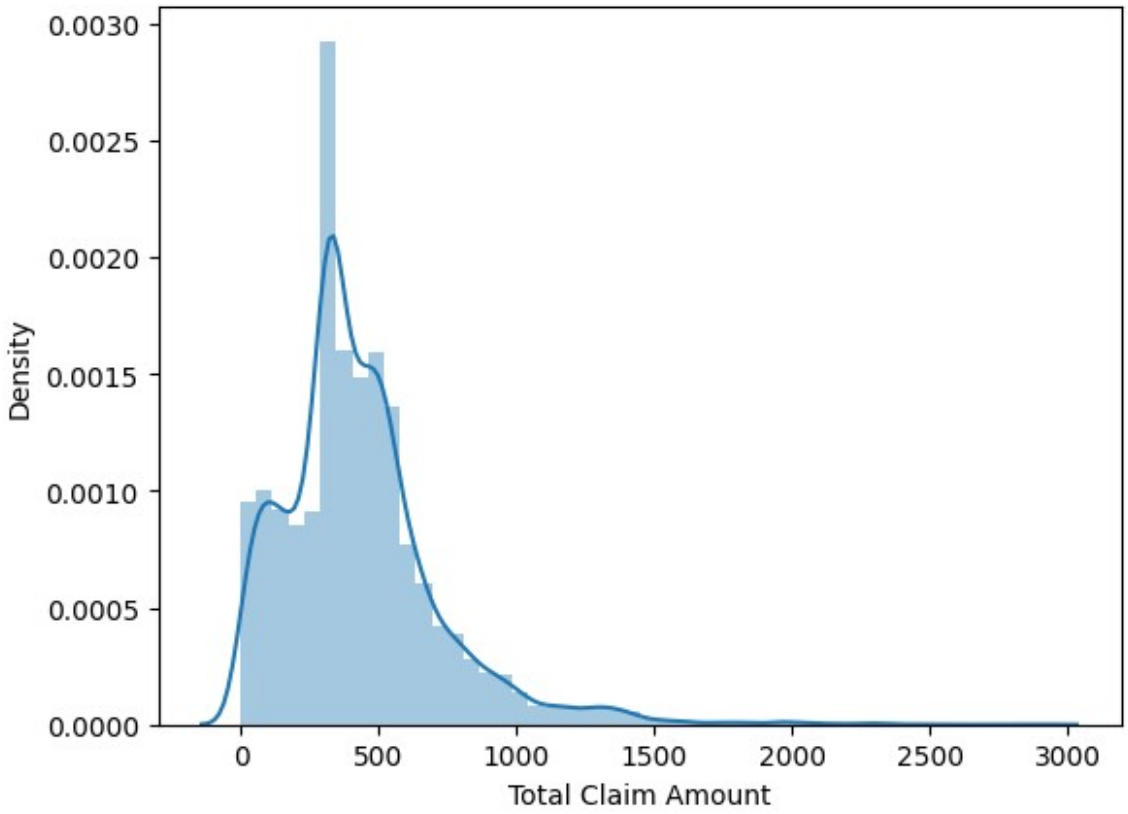

```
sns.boxplot(df["Months Since Policy Inception"])
plt.show()
```



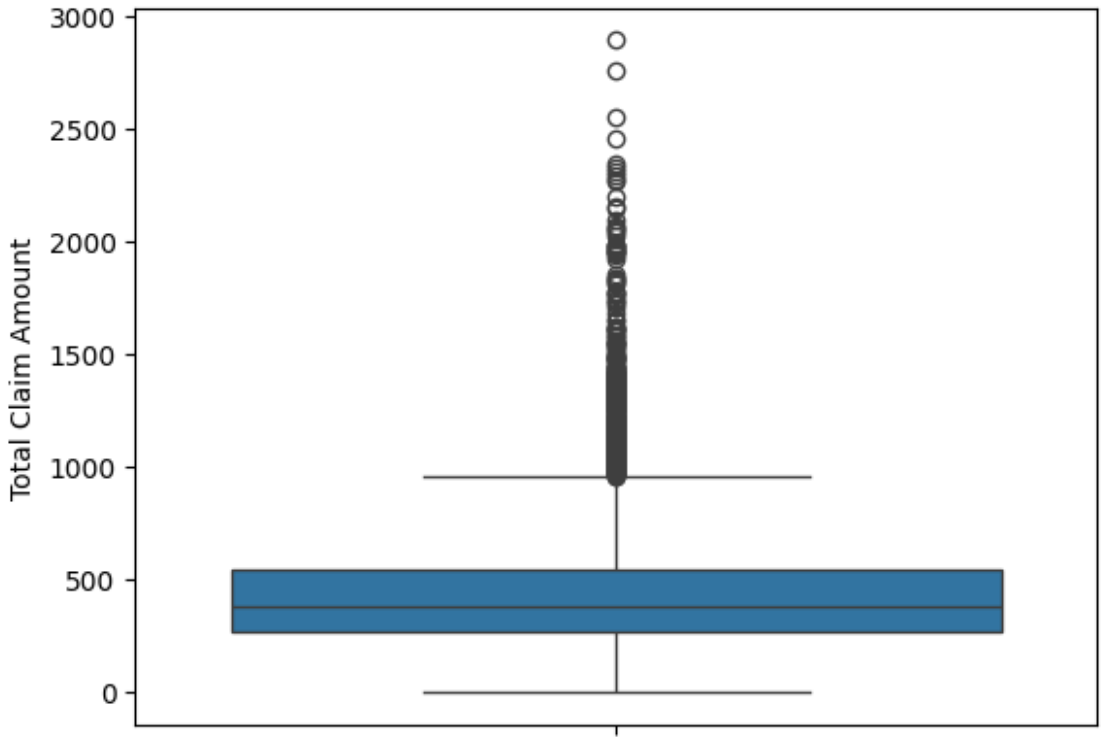
Total Claim Amount

```
sns.distplot(df["Total Claim Amount"])
plt.show()
```

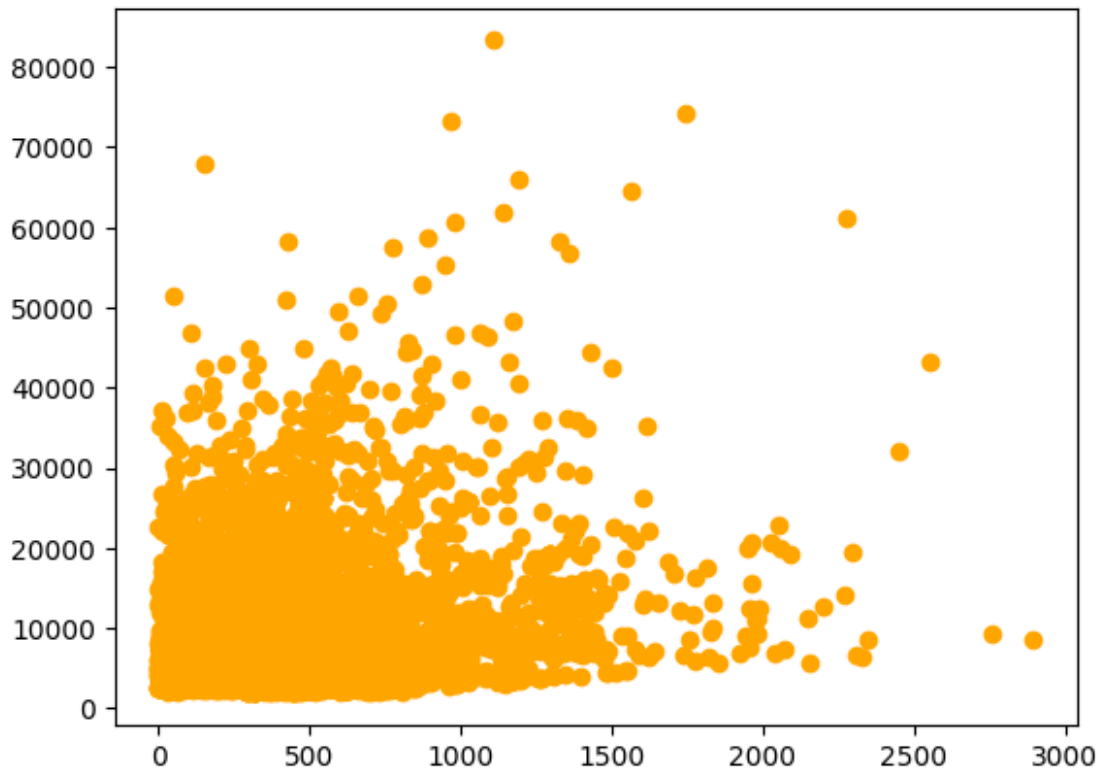
```
<ipython-input-181-f98ae4cf0e5a>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Total Claim Amount"])
```




```
sns.boxplot(df["Total Claim Amount"])
plt.show()
```




```
plt.scatter(df["Total Claim Amount"],df["CLV"],color="orange")
plt.show()
```

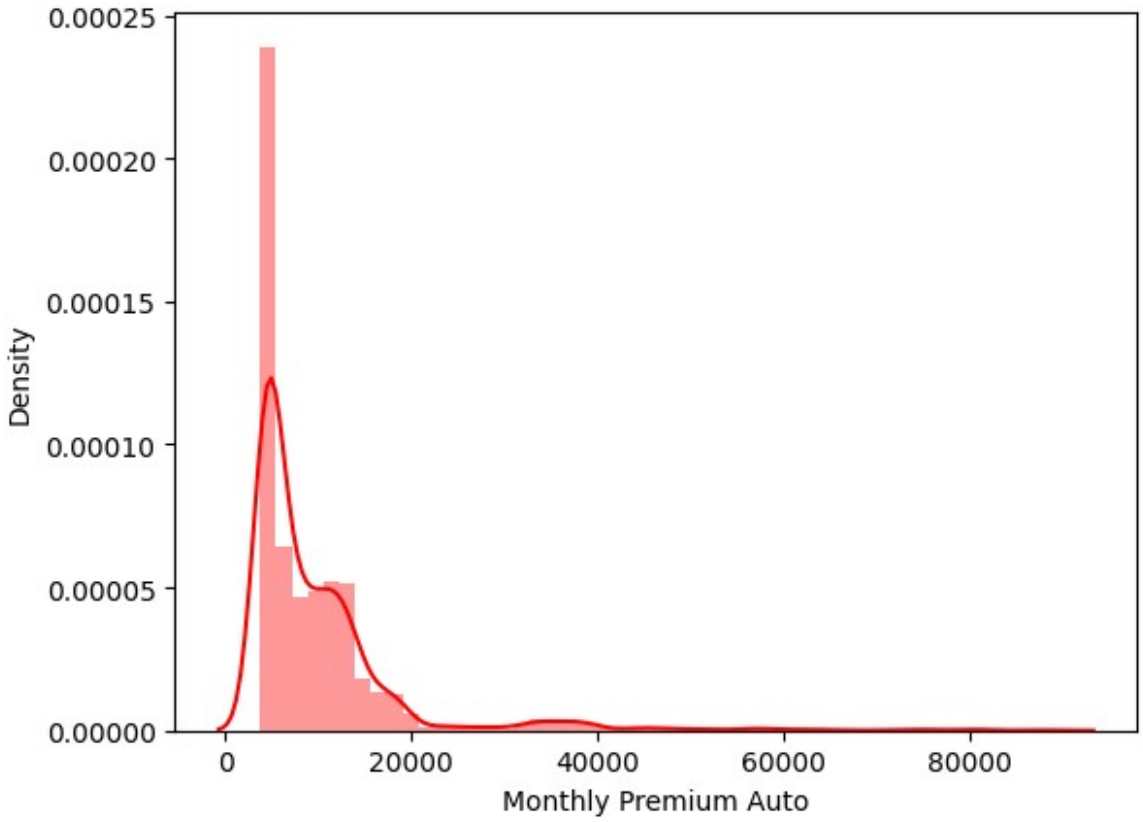


There is a linear relationship between CLV and Total Claim Amount

The monthly premium auto and income feature has multiple peaks we can apply any of the transformation(SQUARE/CUBE)

```
sns.distplot(np.square(df["Monthly Premium Auto"]),color ="r")
plt.show()
```

```
<ipython-input-184-9c9bd408c9c2>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(np.square(df["Monthly Premium Auto"]),color ="r")
```



CATEGORICAL VARIABLES

```
cat_cols=df.select_dtypes(include="object")

no_col=df[["Number of Open Complaints","Number of Policies"]] #Dropped column added in categorical column

cat_cols=pd.concat([cat_cols,no_col],axis=1)

cat_cols.head()
```


```
{"summary":"{\n  \"name\": \"cat_cols\",\n  \"rows\": 9134,\n  \"fields\": [\n    {\n      \"column\": \"Customer\",\n      \"properties\": {\n        \"dtype\": \"string\",\n        \"num_unique_values\": 9134,\n        \"samples\": [\n          \"ZQ59828\",\n          \"XM45289\",\n          \"GP20408\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"State\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 5,\n        \"samples\": [\n          \"Arizona\",\n          \"Oregon\",\n          \"Nevada\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Response\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n        \"samples\": [\n          \"Yes\",\n          \"No\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Coverage\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 3,\n        \"samples\": [\n          \"Basic\",\n          \"Extended\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Education\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 5,\n        \"samples\": [\n          \"College\",\n          \"Doctor\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Effective To Date\",\n      \"properties\": {\n        \"dtype\": \"object\",\n        \"num_unique_values\": 59,\n        \"samples\": [\n          \"2/24/11\",\n          \"1/25/11\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"EmploymentStatus\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 5,\n        \"samples\": [\n          \"Unemployed\",\n          \"Retired\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Gender\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n        \"samples\": [\n          \"M\",\n          \"F\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
```

```
}\n    },\n    {\n      \"column\": \"Location Code\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Suburban\",\n          \"Rural\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Marital Status\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Married\",\n          \"Single\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Policy Type\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Corporate Auto\",\n          \"Personal Auto\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Policy\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
9,\n        \"samples\": [\n          \"Special L1\",\n
\"Personal L3\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Renew Offer Type\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          \"Offer3\",\n          \"Offer4\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Sales Channel\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          \"Call
Center\",\n          \"Branch\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Vehicle Class\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 6,\n        \"samples\": [\n          \"Two-
Door Car\",\n          \"Four-Door Car\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Vehicle Size\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Medsize\",\n          \"Small\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Number of Open Complaints\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 5,\n
\"num_unique_values\": 6,\n        \"samples\": [\n          0,\n
2\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Number of Policies\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 2,\n        \"min\": 1,\n
\"max\": 9,\n        \"num_unique_values\": 9,\n        \"samples\":
[\n          6,\n          8\n        ],\n        \"semantic_type\":
```

```
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"cat_cols"}
```

```
cat_cols.drop('Effective To Date',axis=1,inplace=True)
```

```
cat_cols.columns
```

```
Index(['Customer', 'State', 'Response', 'Coverage', 'Education',
       'EmploymentStatus', 'Gender', 'Location Code', 'Marital
Status',
       'Policy Type', 'Policy', 'Renew Offer Type', 'Sales Channel',
       'Vehicle Class', 'Vehicle Size', 'Number of Open Complaints',
       'Number of Policies'],
      dtype='object')
```

```
for i in cat_cols:
  print("Unique values in",str(i),"is",df[i].nunique())
  print(df[i].value_counts())
  print("------------------------------------------------")
```

```
Unique values in Customer is 9134
BU79786    1
PU81096    1
CO75086    1
WW52683    1
XO38850    1
          ..
HS14476    1
YL91587    1
CT18212    1
EW35231    1
Y167826    1
Name: Customer, Length: 9134, dtype: int64
------------------------------------------------
Unique values in State is 5
California    3150
Oregon        2601
Arizona       1703
Nevada         882
Washington     798
Name: State, dtype: int64
------------------------------------------------
Unique values in Response is 2
No     7826
Yes    1308
Name: Response, dtype: int64
------------------------------------------------
Unique values in Coverage is 3
Basic       5568
Extended    2742
```

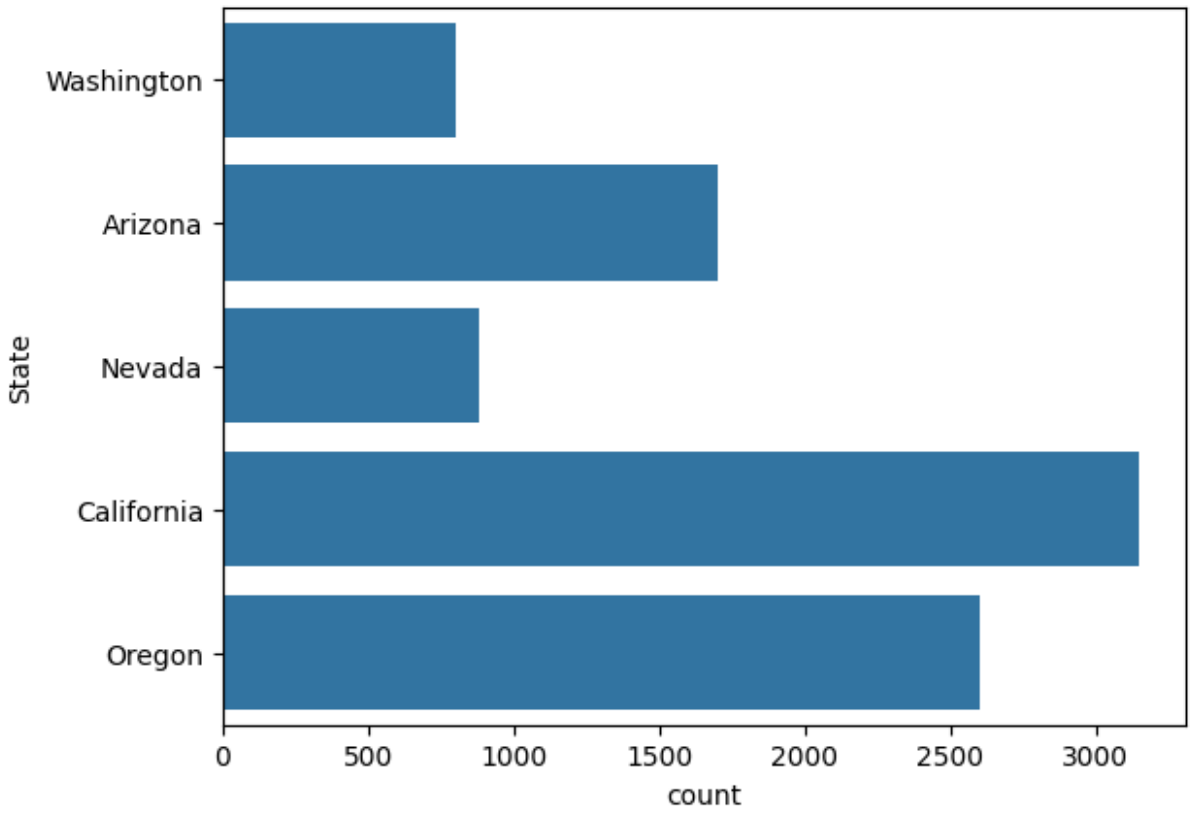
```
Premium      824
Name: Coverage, dtype: int64
-----------------------------------------------
Unique values in Education is 5
Bachelor                2748
College                 2681
High School or Below    2622
Master                   741
Doctor                   342
Name: Education, dtype: int64
-----------------------------------------------
Unique values in EmploymentStatus is 5
Employed         5698
Unemployed       2317
Medical Leave     432
Disabled          405
Retired           282
Name: EmploymentStatus, dtype: int64
-----------------------------------------------
Unique values in Gender is 2
F    4658
M    4476
Name: Gender, dtype: int64
-----------------------------------------------
Unique values in Location Code is 3
Suburban    5779
Rural       1773
Urban       1582
Name: Location Code, dtype: int64
-----------------------------------------------
Unique values in Marital Status is 3
Married     5298
Single      2467
Divorced    1369
Name: Marital Status, dtype: int64
-----------------------------------------------
Unique values in Policy Type is 3
Personal Auto     6788
Corporate Auto    1968
Special Auto       378
Name: Policy Type, dtype: int64
-----------------------------------------------
Unique values in Policy is 9
Personal L3     3426
Personal L2     2122
Personal L1     1240
Corporate L3    1014
Corporate L2     595
Corporate L1     359
```

```
Special L2       164
Special L3       148
Special L1        66
Name: Policy, dtype: int64
-------------------------------------------
Unique values in Renew Offer Type is 4
Offer1    3752
Offer2    2926
Offer3    1432
Offer4    1024
Name: Renew Offer Type, dtype: int64
-------------------------------------------
Unique values in Sales Channel is 4
Agent          3477
Branch         2567
Call Center    1765
Web            1325
Name: Sales Channel, dtype: int64
-------------------------------------------
Unique values in Vehicle Class is 6
Four-Door Car    4621
Two-Door Car     1886
SUV              1796
Sports Car        484
Luxury SUV        184
Luxury Car        163
Name: Vehicle Class, dtype: int64
-------------------------------------------
Unique values in Vehicle Size is 3
Medsize    6424
Small      1764
Large       946
Name: Vehicle Size, dtype: int64
-------------------------------------------
Unique values in Number of Open Complaints is 6
0    7252
1    1011
2     374
3     292
4     149
5      56
Name: Number of Open Complaints, dtype: int64
-------------------------------------------
Unique values in Number of Policies is 9
1    3251
2    2294
3    1168
7     433
9     416
```

```
4     409
5     407
8     384
6     372
Name: Number of Policies, dtype: int64
------------------------------------------------
```

State

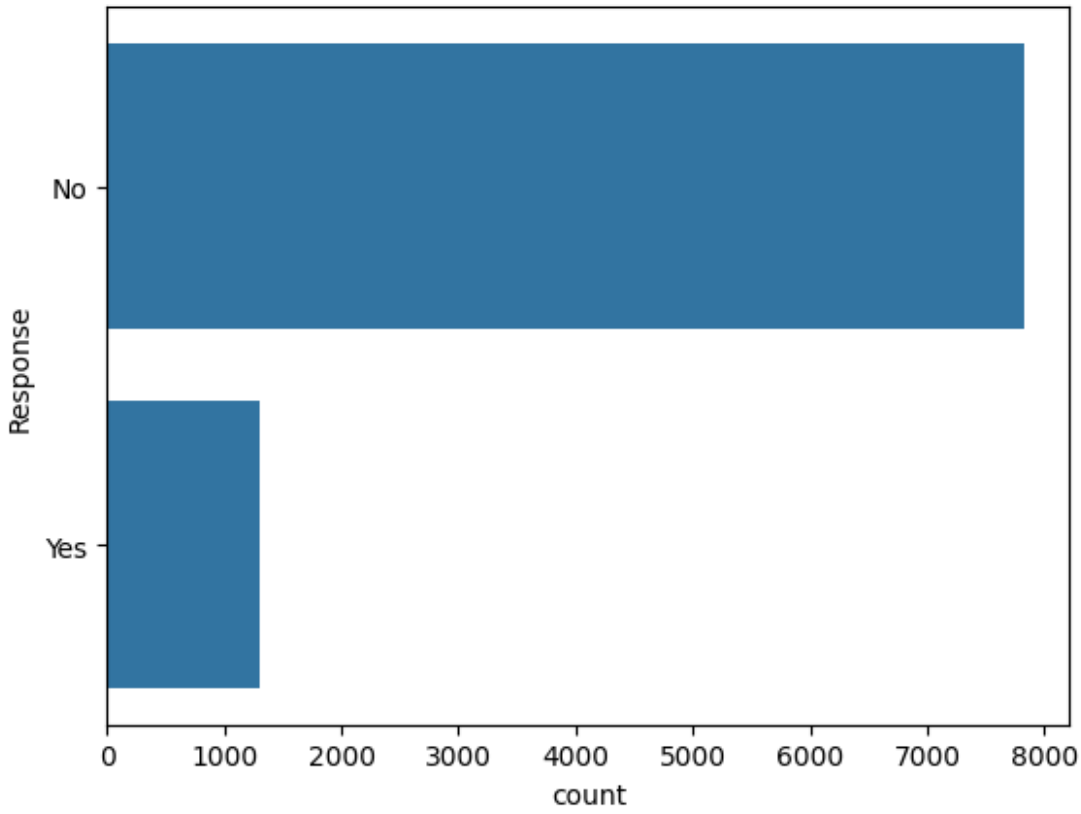
```
sns.countplot(df["State"])
plt.show()
```



Most number of people are residing in california

Response

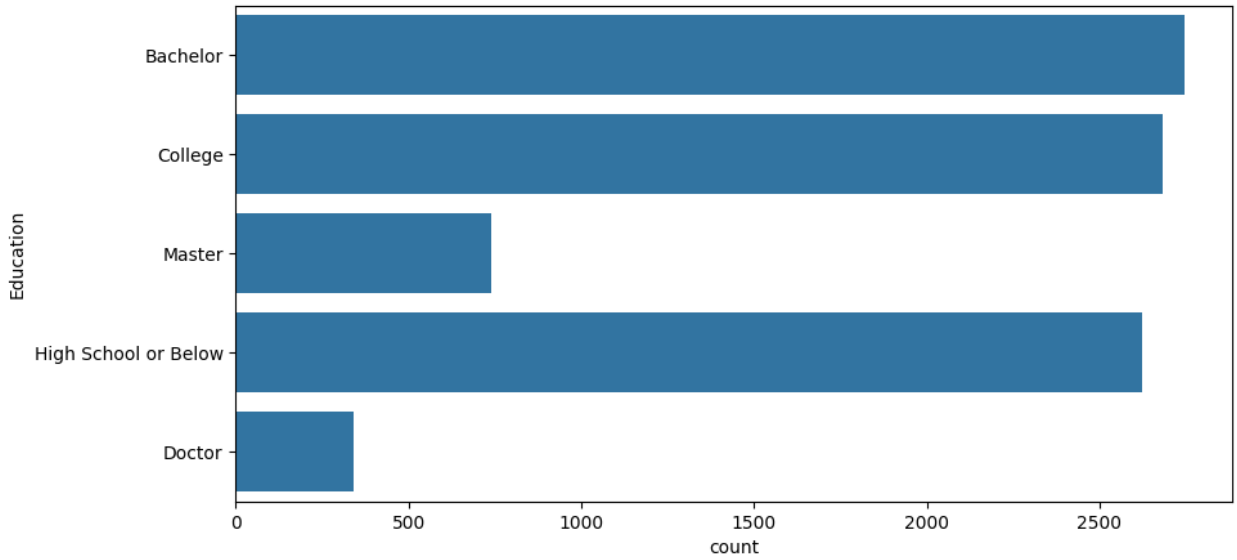
```
sns.countplot(df["Response"])
plt.show()
```



Most number of people's Response is NO

Education

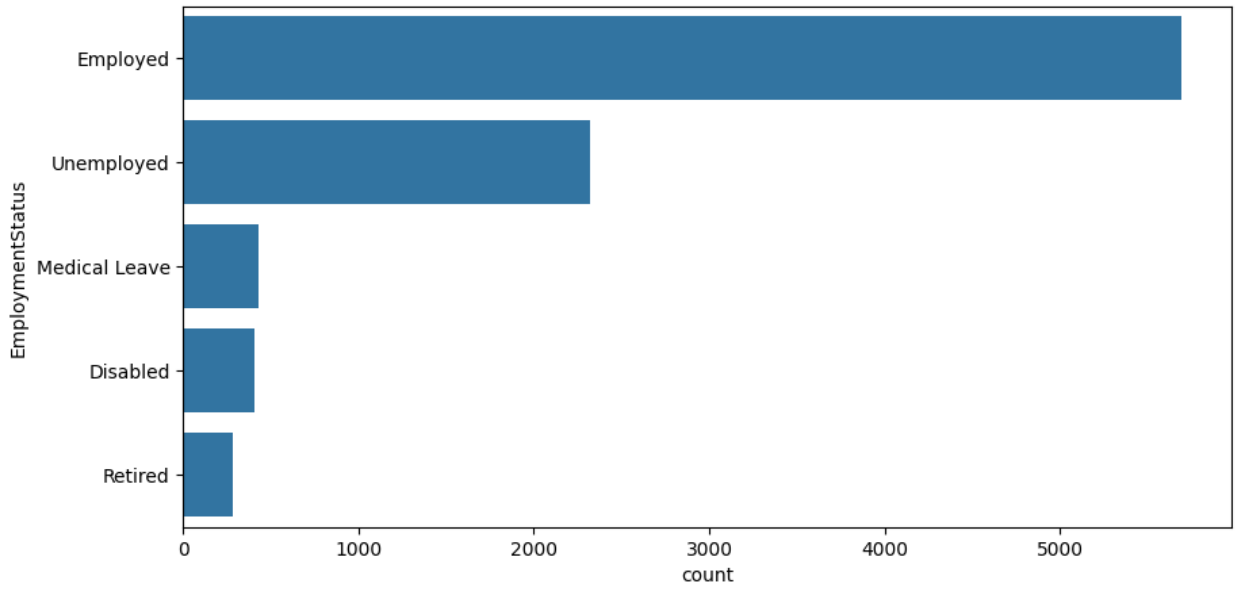
```
plt.figure(figsize=(10,5))
sns.countplot(df["Education"])
plt.show()
```



Most number of people's have completed either Bachelor's or College Degree

Employment Status

```
plt.figure(figsize=(10,5))
sns.countplot(df["EmploymentStatus"])
plt.show()
```



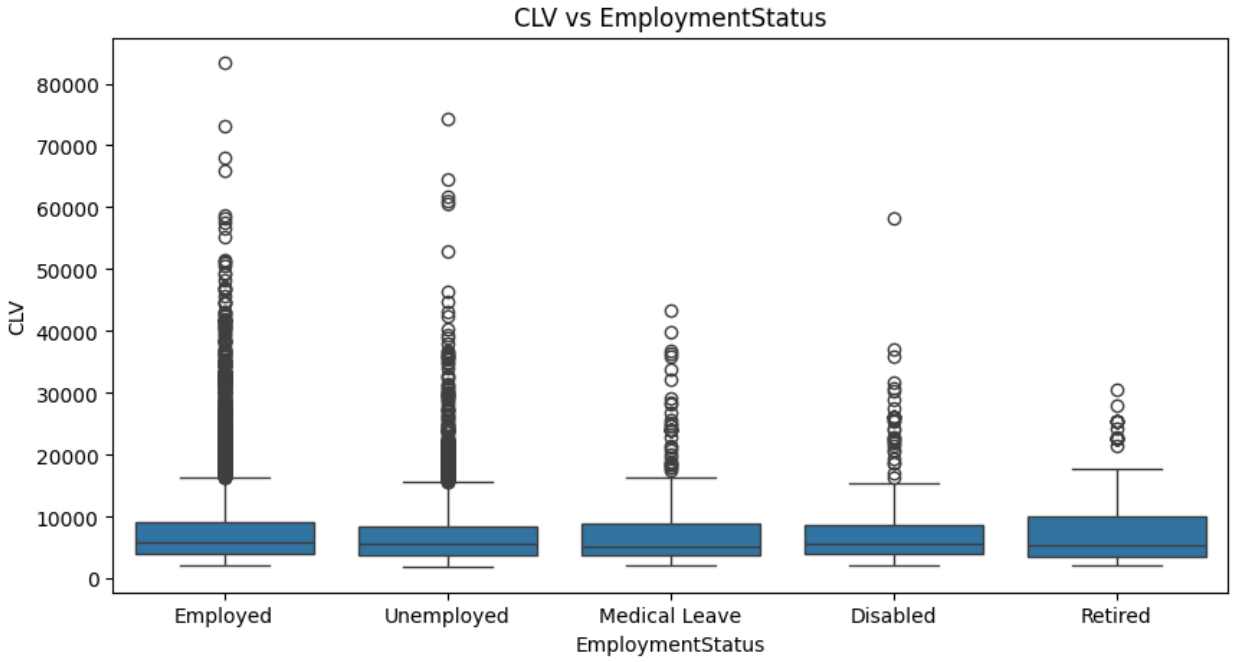
Most of the people are either employed or unemployed. Some are on MEdical Leave,some are Disabled and some are retired

```
df["EmploymentStatus"].value_counts(normalize=True)*100
```

```
Employed          62.382308
Unemployed        25.366762
Medical Leave      4.729582
Disabled           4.433983
Retired            3.087366
Name: EmploymentStatus, dtype: float64
```

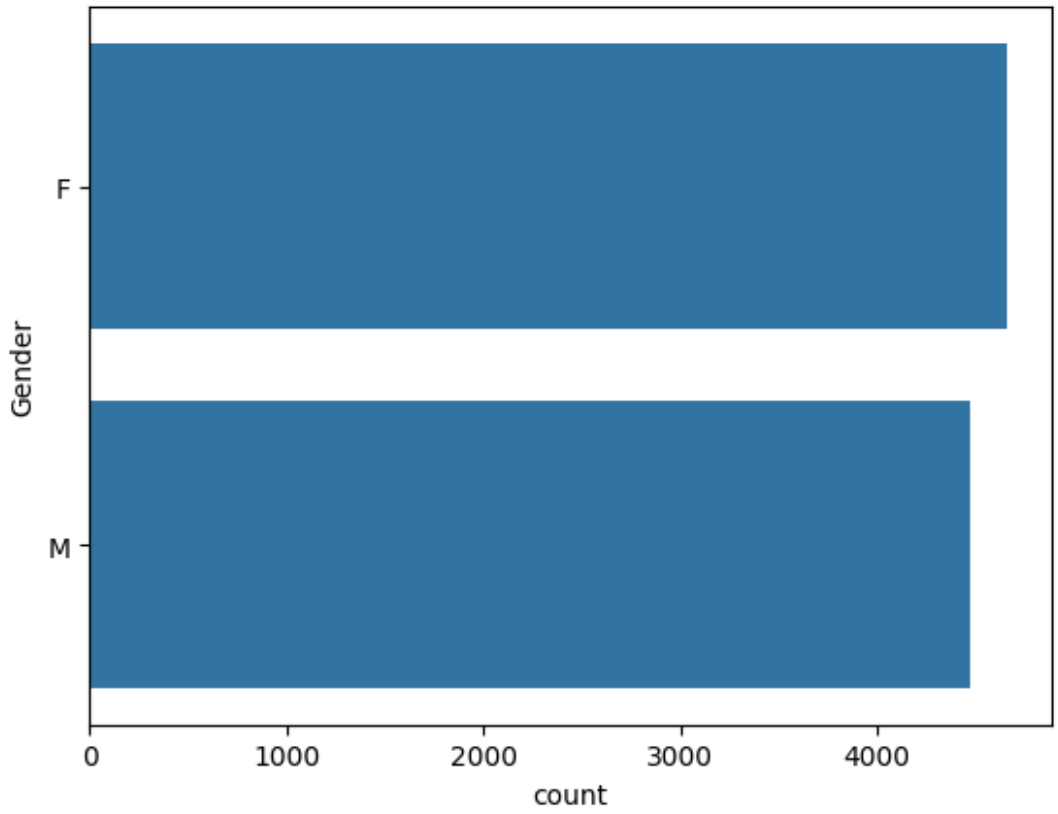
Around 62.38% of the customers are employed

```
plt.figure(figsize=(10,5))
sns.boxplot(x=df['EmploymentStatus'],y=df['CLV'])
plt.title("CLV vs EmploymentStatus")
plt.show()
```



Gender

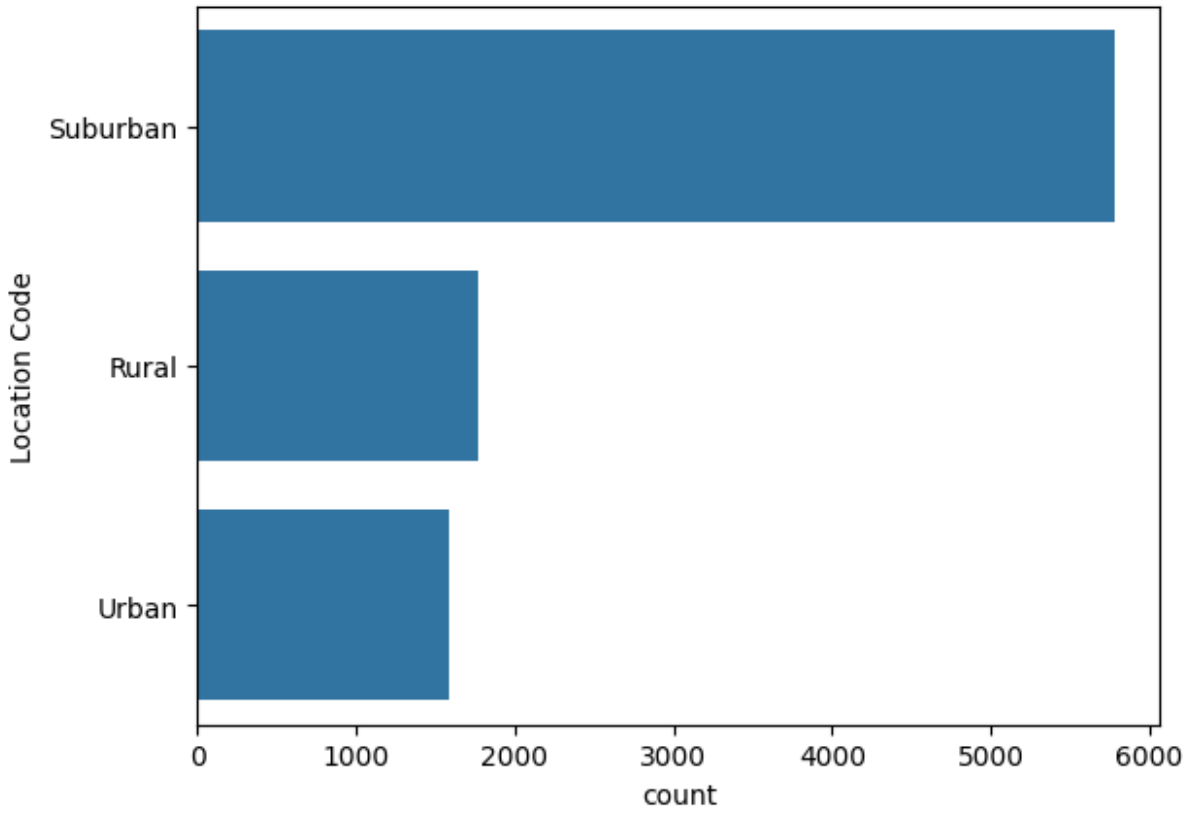
```
sns.countplot(df["Gender"])
plt.show()
```



Most Number of People are Female.

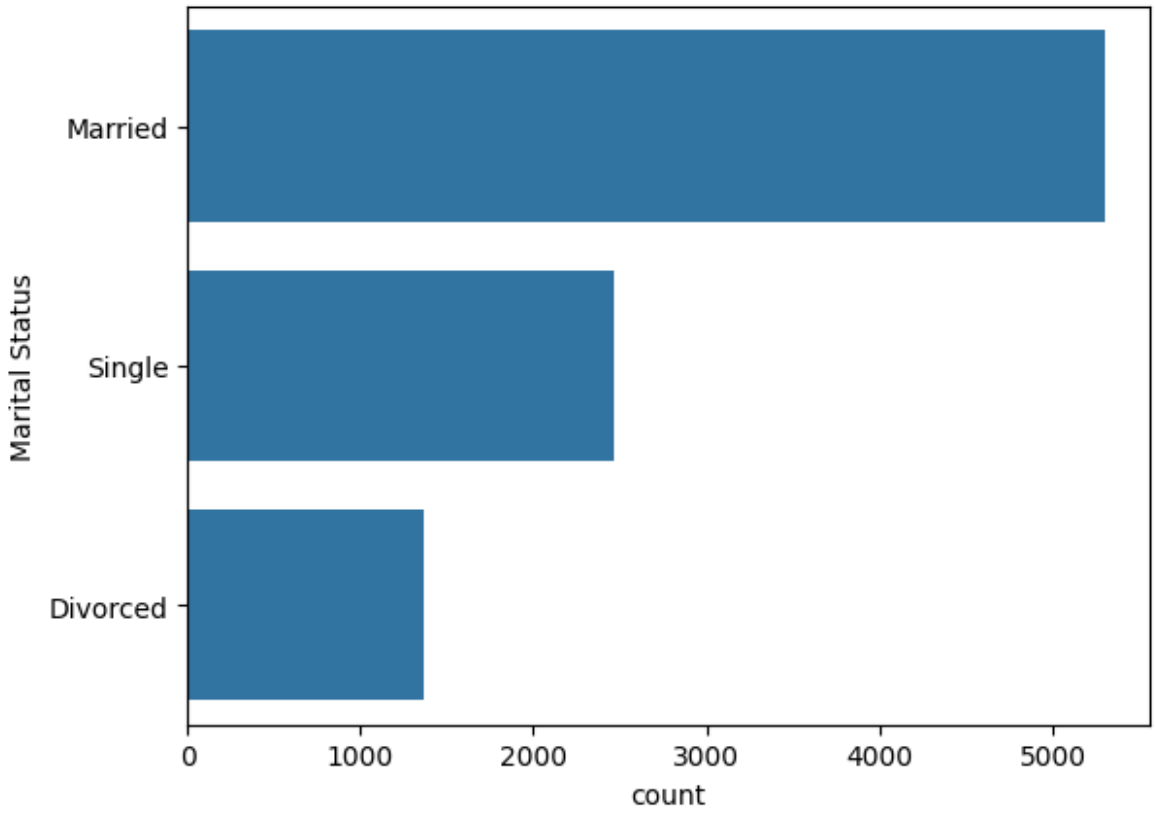
Location code

```
sns.countplot(df["Location Code"])
plt.savefig("location.png")
plt.show()
```



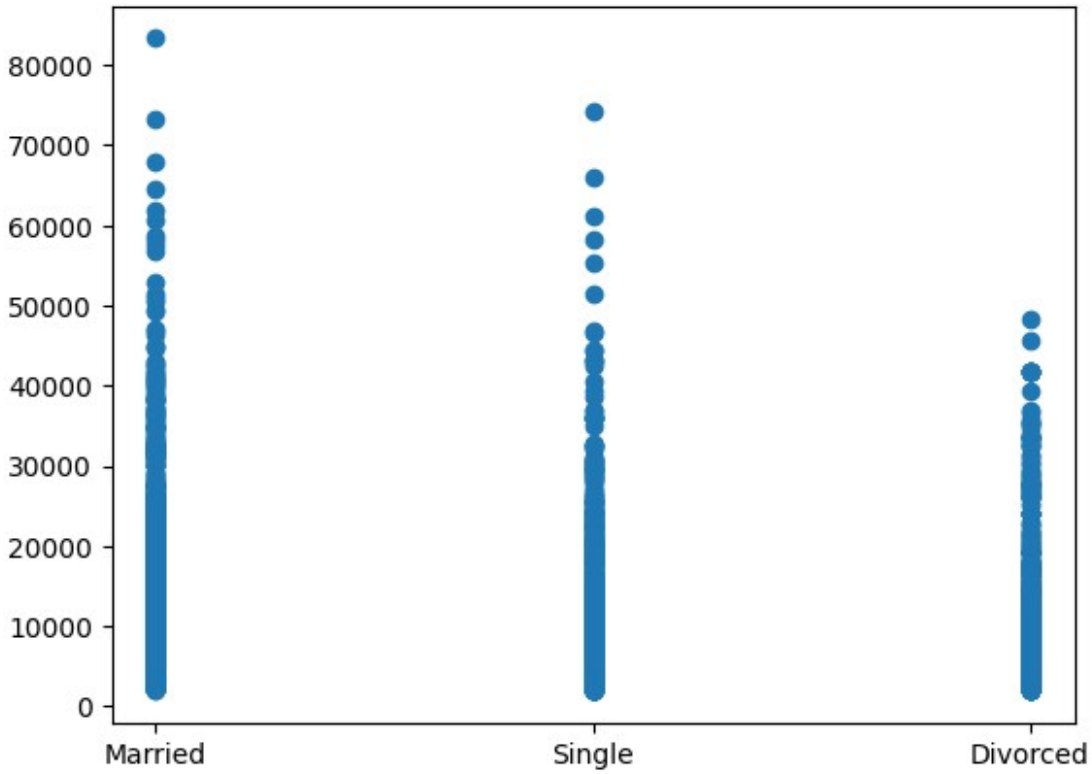
Marital Status

```
sns.countplot(df["Marital Status"])
plt.show()
```



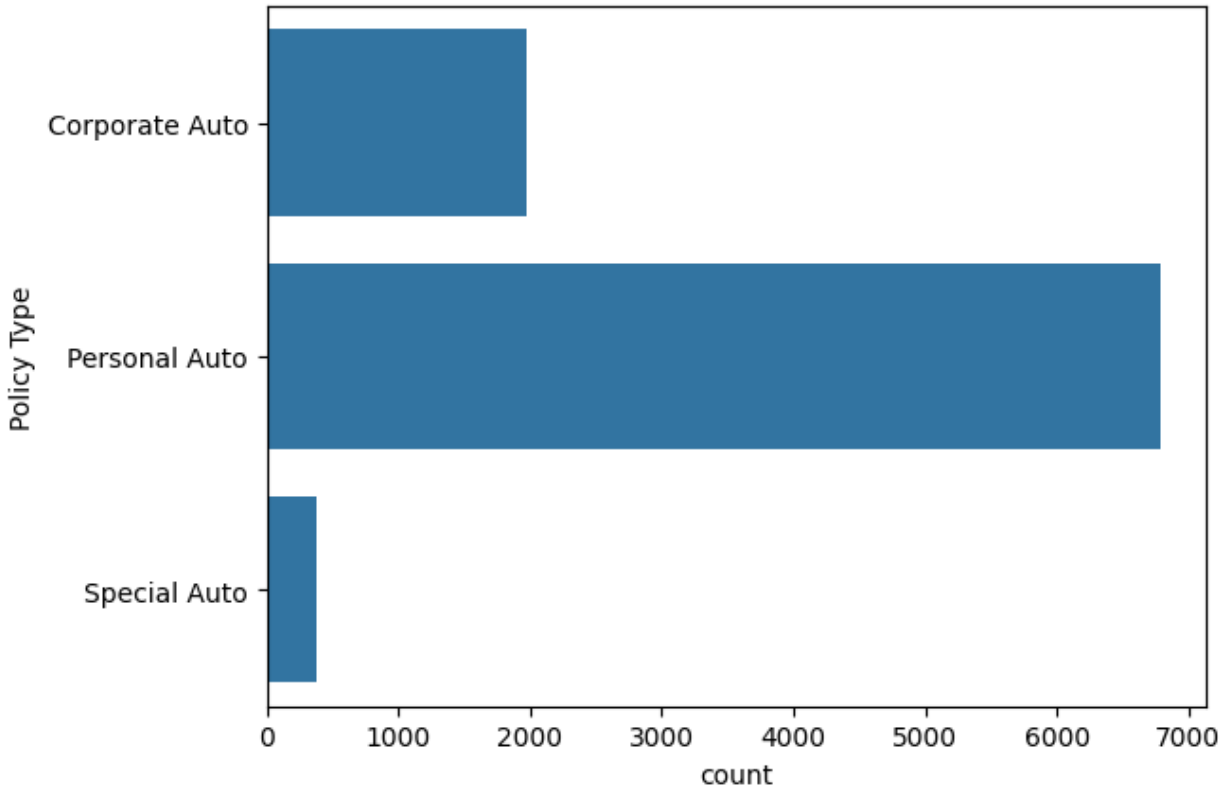
Most no. of people are married.

```
plt.scatter(df["Marital Status"],df['CLV'])
plt.show()
```



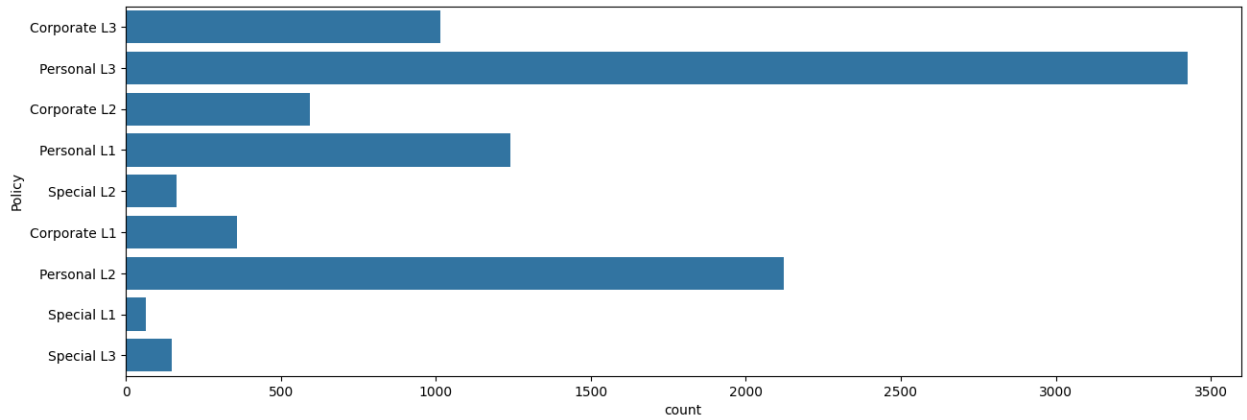
Policy Type

```
sns.countplot(df["Policy Type"])
plt.show()
```



Policy

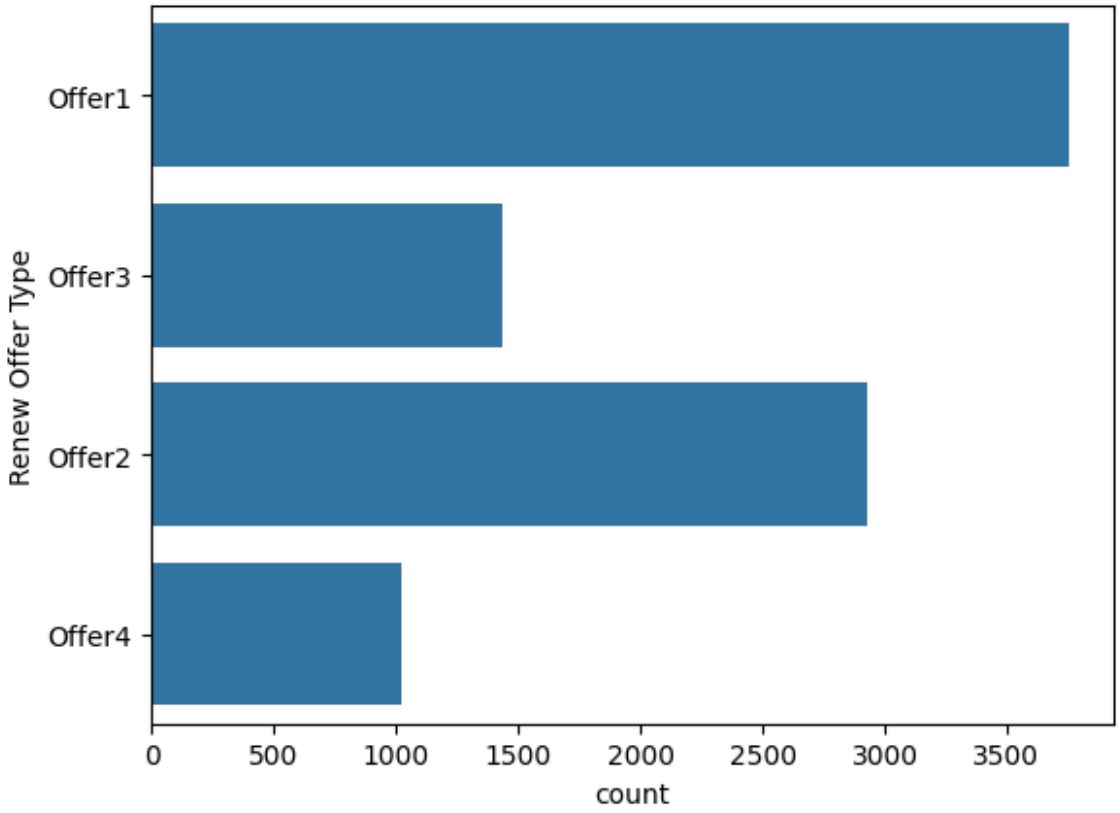
```
plt.figure(figsize=(15,5))
sns.countplot(df["Policy"])
plt.show()
```



Personal L3 policy subcategory has the most number of customers.

Renew Offer Type

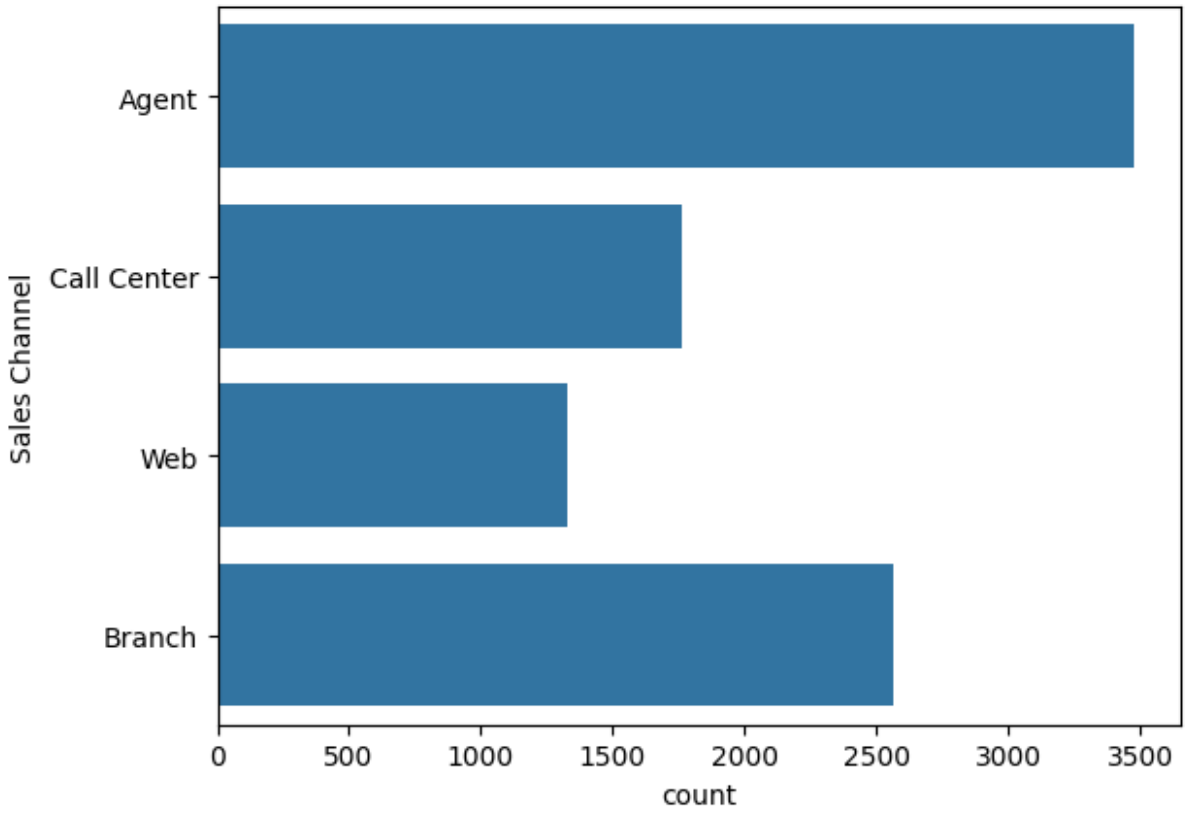
```
sns.countplot(df["Renew Offer Type"])
plt.show()
```



Most preferred offer by people is offer1.

Sales Channel

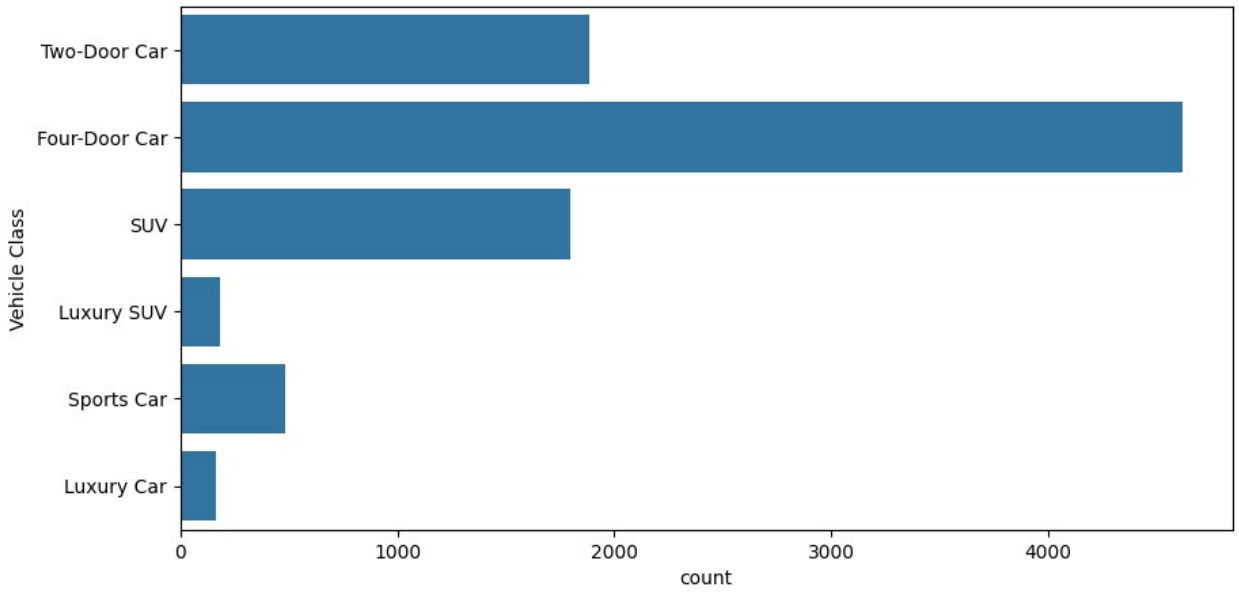
```
sns.countplot(df["Sales Channel"])
plt.show()
```



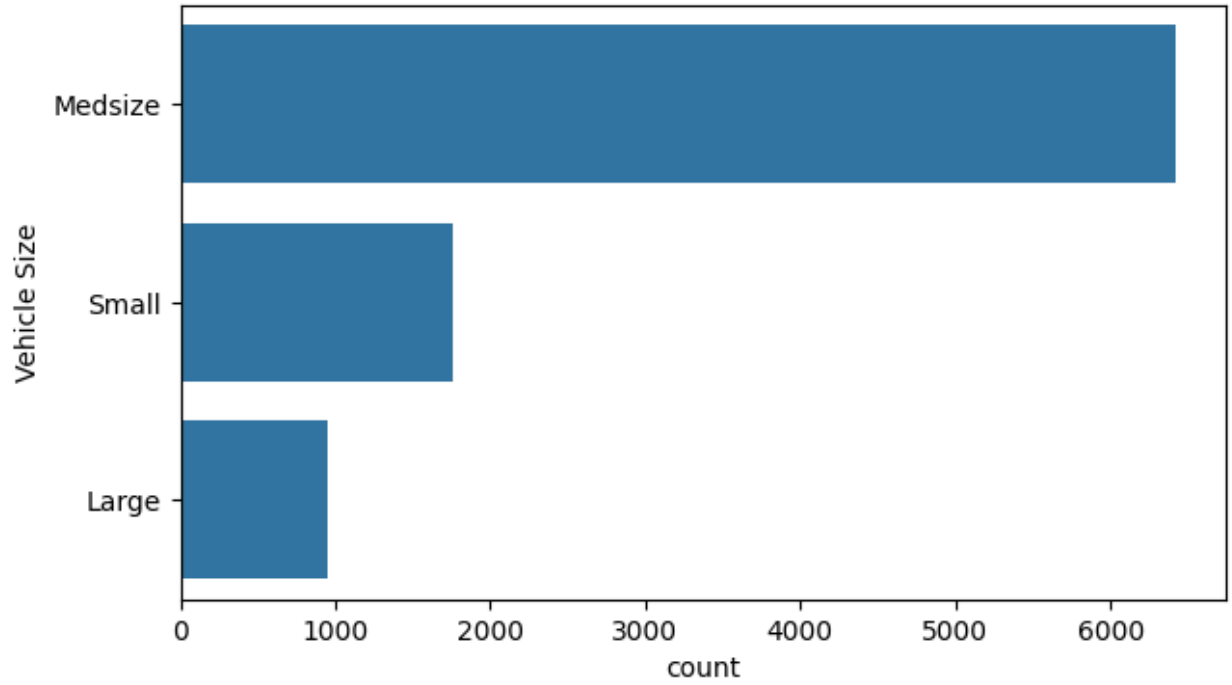
Mostl preferred sales Channel is Agent.

Vehicle Class

```
plt.figure(figsize=(10,5))
sns.countplot(df["Vehicle Class"])
plt.show()
```




```
plt.figure(figsize=(7,4))
sns.countplot(df["Vehicle Size"])
plt.show()
```




```
df['Effective To Date']=pd.to_datetime(df['Effective To
Date'],infer_datetime_format=True) #convert date into date time format

df["Months"]=df["Effective To Date"].dt.month
```

```
df["Months"]=df["Months"].astype('object')
```

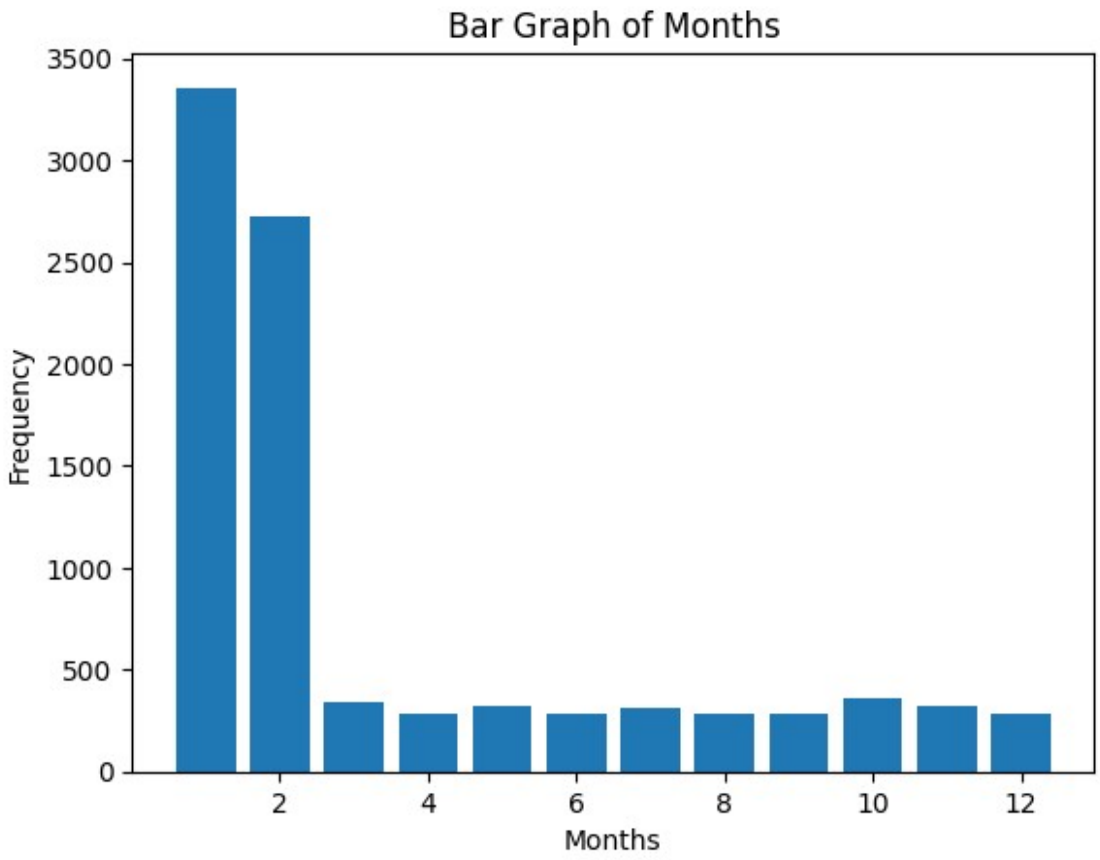
Months

```
df.head(2)
```

```
{"type":"dataframe","variable_name":"df"}
```

```
# We will analysing data according to months
# Count the occcurance of each Month
month_counts=df['Months'].value_counts()

# Sort the months by their index(month names)
math_counts=month_counts.sort_index()

#Create the bar graph
plt.bar(month_counts.index,month_counts.values)

# Adding labels and title
plt.xlabel('Months')
plt.ylabel('Frequency')
plt.title('Bar Graph of Months')
```

```
Text(0.5, 1.0, 'Bar Graph of Months')
```




```
cat_cols.columns
```

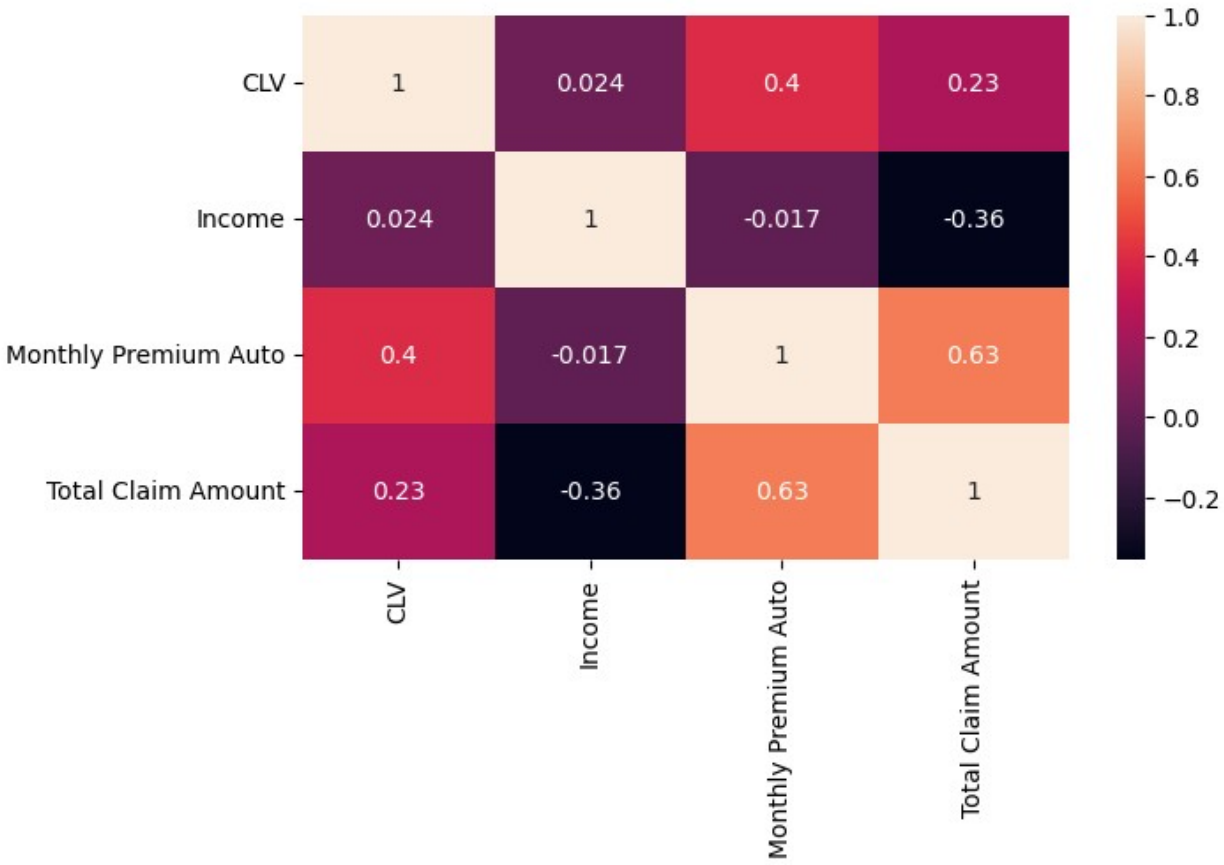
```
Index(['Customer', 'State', 'Response', 'Coverage', 'Education',
       'EmploymentStatus', 'Gender', 'Location Code', 'Marital
Status',
       'Policy Type', 'Policy', 'Renew Offer Type', 'Sales Channel',
       'Vehicle Class', 'Vehicle Size', 'Number of Open Complaints',
       'Number of Policies'],
      dtype='object')
```

```
df.columns
```

```
Index(['Customer', 'State', 'CLV', 'Response', 'Coverage',
'Education',
       'Effective To Date', 'EmploymentStatus', 'Gender', 'Income',
       'Location Code', 'Marital Status', 'Monthly Premium Auto',
       'Months Since Last Claim', 'Months Since Policy Inception',
       'Number of Open Complaints', 'Number of Policies', 'Policy
Type',
       'Policy', 'Renew Offer Type', 'Sales Channel', 'Total Claim
Amount',
       'Vehicle Class', 'Vehicle Size', 'Months'],
      dtype='object')
```

Heat Map

```
heatmap=df[['CLV','Income','Monthly Premium Auto','Total Claim
Amount']]

plt.figure(figsize=(7,4))
sns.heatmap(heatmap.corr(),annot=True)
plt.show()
```



Statistical Significance

```
stas.shapiro(df['CLV'])

/usr/local/lib/python3.10/dist-packages/scipy/stats/
_morestats.py:1882: UserWarning: p-value may not be accurate for N >
5000.
  warnings.warn("p-value may not be accurate for N > 5000.")

ShapiroResult(statistic=0.7033725380897522, pvalue=0.0)
```

p-value less than 0.05 we reject null hypothesis the data is not normally distributed

We will proceed with non parametric tests since the dependent variable is not normally distributed

```
columns_cat=list(cat_cols.columns)
columns_cat
```

```
['Customer',
 'State',
 'Response',
 'Coverage',
 'Education',
 'EmploymentStatus',
 'Gender',
 'Location Code',
 'Marital Status',
 'Policy Type',
 'Policy',
 'Renew Offer Type',
 'Sales Channel',
 'Vehicle Class',
 'Vehicle Size',
 'Number of Open Complaints',
 'Number of Policies']
```

```
manwithwhiteneyy=[]
anova=[]
for i in columns_cat:
  if (df[i].nunique()>2):
    anova.append(i)
  else:
    manwithwhiteneyy.append(i)
print("Anova:",anova)
print('TTest:',manwithwhiteneyy)
```

```
Anova: ['Customer', 'State', 'Coverage', 'Education',
'EmploymentStatus', 'Location Code', 'Marital Status', 'Policy Type',
'Policy', 'Renew Offer Type', 'Sales Channel', 'Vehicle Class',
'Vehicle Size', 'Number of Open Complaints', 'Number of Policies']
TTest: ['Response', 'Gender']
```

Data Preprocessing

```
cat_cols.head()
```


```
{"summary":"{\n  \"name\": \"cat_cols\",\n  \"rows\": 9134,\n
\"fields\": [\n    {\n      \"column\": \"Customer\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 9134,\n        \"samples\": [\n
\"ZQ59828\",\n          \"XM45289\",\n          \"GP20408\"\
```

```
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"State\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 5,\n        \"samples\": [\n
\"Arizona\",\n          \"Oregon\",\n          \"Nevada\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Response\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"Yes\",\
n          \"No\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Coverage\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          \"Basic\",\n          \"Extended\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Education\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n
\"College\",\n          \"Doctor\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"EmploymentStatus\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n
\"Unemployed\",\n          \"Retired\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Gender\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
2,\n        \"samples\": [\n          \"M\",\n          \"F\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Location Code\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Suburban\",\n          \"Rural\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Marital Status\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Married\",\n          \"Single\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Policy Type\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Corporate Auto\",\n          \"Personal Auto\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Policy\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
9,\n        \"samples\": [\n          \"Special L1\",\n
\"Personal L3\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

```
\"Renew Offer Type\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          \"Offer3\",\n          \"Offer4\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Sales Channel\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          \"Call
Center\",\n          \"Branch\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Vehicle Class\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 6,\n        \"samples\": [\n          \"Two-
Door Car\",\n          \"Four-Door Car\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Vehicle Size\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Medsize\",\n          \"Small\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Number of Open Complaints\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 5,\n
\"num_unique_values\": 6,\n        \"samples\": [\n          0,\n
2\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Number of Policies\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 2,\n        \"min\": 1,\n
\"max\": 9,\n        \"num_unique_values\": 9,\n        \"samples\":
[\n          6,\n          8\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"cat_cols"}
```

```
catg=pd.get_dummies(cat_cols,drop_first=True)  #we converted all
catagorical(object data type) into dummies or numbers
catg
```

```
{"type":"dataframe","variable_name":"catg"}
```

```
dfn=pd.concat([numerical_cols,catg],axis=1)
dfn.head()
```

```
{"type":"dataframe","variable_name":"dfn"}
```

```
dfn.rename(columns={'CLV':'CLV'},inplace=True)
```

MODEL TRAINING

```
# Split the data
x=dfn.drop(['CLV'],axis=1)
```

```
y=dfn['CLV']
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,rand
om_state=42)

numerical_cols.rename(columns={'CLV':'CLV'},inplace=True)

print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(6393, 9183)
(6393,)
(2741, 9183)
(2741,)
```

MODEL BUILDING

LINEAR REGRESSION

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
model=lr.fit(x_train,y_train)
print(f'R^2 score for train: {lr.score(x_train,y_train)}')
print(f'R^2 score for train: {lr.score(x_test,y_test)}')
```

```
R^2 score for train:1.0
R^2 score for train:0.1585232377674567
```

```
y_pred=model.predict(x_test)

from sklearn.metrics import
mean_squared_error,r2_score,mean_absolute_error

print('RSME:',np.sqrt(mean_squared_error(y_test,y_pred)))
print('MAE:',mean_absolute_error(y_test,y_pred))
print('R-squared:',r2_score(y_test,y_pred))
```

```
RSME: 6617.391707041234
MAE: 3966.906738479852
R-squared: 0.1585232377674567
```

DECISION TREE

```
from sklearn.tree import DecisionTreeRegressor
```

```
dt=DecisionTreeRegressor(random_state=1)
dt.fit(x_train,y_train)
y_pred=dt.predict(x_test)
print('RSME:',np.sqrt(mean_squared_error(y_test,y_pred)))
print('MAE:',mean_absolute_error(y_test,y_pred))
print('R-squared:',r2_score(y_test,y_pred))
```

```
RSME: 4968.477878964912
MAE: 1545.82956511857
R-squared: 0.525632242443886
```

RANDOM FOREST

```
from sklearn.ensemble import RandomForestRegressor

rf=RandomForestRegressor(random_state=1)
rf.fit(x_train,y_train)
y_pred=rf.predict(x_test)
print('RSME:',np.sqrt(mean_squared_error(y_test,y_pred)))
print('MAE:',mean_absolute_error(y_test,y_pred))
print('R-squared:',r2_score(y_test,y_pred))
```

```
RSME: 4199.016155677687
MAE: 1406.8917145753467
R-squared: 0.66118429507984
```

HYPERPARAMETER TUNING OF RANDOM FOREST

```
from sklearn.model_selection import GridSearchCV
rf=RandomForestRegressor()
params = {
    'max_depth' : [10,20,30],
    'n_estimators' : [100,200,50],
    "bootstrap" : [True,False],
    'max_features' :['auto', 'sqrt', 'log2']
    }
grid=GridSearchCV(estimator=rf,param_grid=params,cv=5,n_jobs=-
1,return_train_score=True)
grid.fit(x,y)
grid.best_params_
```

```
x_test.head()
```

```
{"type":"dataframe","variable_name":"x_test"}
```