EDA:- Exploratory data analysis X-plo-rat-ory

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

In [2]:
```
credit_card_data=pd.read_csv("A:\LLM\CC GENERAL.csv")
```

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

Out[3]:

| | CUST_ID | BALANCE | BALANCE_FREQUENCY | PURCHASES | ONEOFF_PURCHASES | INSTALLMENTS_I |
|---|---|---|---|---|---|---|
| 0 | C10001 | 40.900749 | 0.818182 | 95.40 | 0.00 | |
| 1 | C10002 | 3202.467416 | 0.909091 | 0.00 | 0.00 | |
| 2 | C10003 | 2495.148862 | 1.000000 | 773.17 | 773.17 | |
| 3 | C10004 | 1666.670542 | 0.636364 | 1499.00 | 1499.00 | |
| 4 | C10005 | 817.714335 | 1.000000 | 16.00 | 16.00 | |

In [4]:
```
credit_card_data.shape
```

Out[4]:
```
(8950, 19)
```

In [5]:
```
credit_card_data.columns
```

Out[5]:
```
Index(['CUST_ID', 'BALANCE', 'BALANCE_FREQUENCY', 'PURCHASES',
       'ONEOFF_PURCHASES', 'INSTALLMENTS_PURCHASES', 'CASH_ADVANCE',
       'PURCHASES_FREQUENCY', 'ONEOFF_PURCHASES_FREQUENCY',
       'PURCHASES_INSTALLMENTS_FREQUENCY', 'CASH_ADVANCE_FREQUENCY',
       'CASH_ADVANCE_TRX', 'PURCHASES_TRX', 'CREDIT_LIMIT', 'PAYMENTS',
       'MINIMUM_PAYMENTS', 'PRC_FULL_PAYMENT', 'TENURE', 'State'],
      dtype='object')
```

In [6]:
```
credit_card_data.isnull()
```

Out[6]:

| | CUST_ID | BALANCE | BALANCE_FREQUENCY | PURCHASES | ONEOFF_PURCHASES | INSTALLMENTS |
|---|---|---|---|---|---|---|
| 0 | False | False | False | False | False | |
| 1 | False | False | False | False | False | |
| 2 | False | False | False | False | False | |
| 3 | False | False | False | False | False | |
| 4 | False | False | False | False | False | |
| ... | ... | ... | ... | ... | ... | |
| 8945 | False | False | False | False | False | |
| 8946 | False | False | False | False | False | |

| | CUST_ID | BALANCE | BALANCE_FREQUENCY | PURCHASES | ONEOFF_PURCHASES | INSTALLMENTS |
|---|---|---|---|---|---|---|
| **8947** | False | False | False | False | False | |
| **8948** | False | False | False | False | False | |
| **8949** | False | False | False | False | False | |

8950 rows × 19 columns

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

Out[7]:
```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        1
PAYMENTS                            0
MINIMUM_PAYMENTS                  313
PRC_FULL_PAYMENT                    0
TENURE                              0
State                               9
dtype: int64
```

In [8]:
```
credit_card_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 19 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   CUST_ID                           8950 non-null   object
 1   BALANCE                           8950 non-null   float64
 2   BALANCE_FREQUENCY                 8950 non-null   float64
 3   PURCHASES                         8950 non-null   float64
 4   ONEOFF_PURCHASES                  8950 non-null   float64
 5   INSTALLMENTS_PURCHASES            8950 non-null   float64
 6   CASH_ADVANCE                      8950 non-null   float64
 7   PURCHASES_FREQUENCY               8950 non-null   float64
 8   ONEOFF_PURCHASES_FREQUENCY        8950 non-null   float64
 9   PURCHASES_INSTALLMENTS_FREQUENCY  8950 non-null   float64
 10  CASH_ADVANCE_FREQUENCY            8950 non-null   float64
 11  CASH_ADVANCE_TRX                  8950 non-null   int64
 12  PURCHASES_TRX                     8950 non-null   int64
 13  CREDIT_LIMIT                      8949 non-null   float64
 14  PAYMENTS                          8950 non-null   float64
 15  MINIMUM_PAYMENTS                  8637 non-null   float64
 16  PRC_FULL_PAYMENT                  8950 non-null   float64
 17  TENURE                            8950 non-null   int64
 18  State                             8941 non-null   object
dtypes: float64(14), int64(3), object(2)
memory usage: 1.3+ MB
```

In [9]:
```
credit_card_data.describe()
```

Out[9]:

| | BALANCE | BALANCE_FREQUENCY | PURCHASES | ONEOFF_PURCHASES | INSTALLMENTS_PUR |
|---|---|---|---|---|---|
| **count** | 8950.000000 | 8950.000000 | 8950.000000 | 8950.000000 | 8950 |
| **mean** | 1564.474828 | 0.877271 | 1003.204834 | 592.437371 | 411 |
| **std** | 2081.531879 | 0.236904 | 2136.634782 | 1659.887917 | 904 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0 |
| **25%** | 128.281915 | 0.888889 | 39.635000 | 0.000000 | 0 |
| **50%** | 873.385231 | 1.000000 | 361.280000 | 38.000000 | 89 |
| **75%** | 2054.140036 | 1.000000 | 1110.130000 | 577.405000 | 468 |
| **max** | 19043.138560 | 1.000000 | 49039.570000 | 40761.250000 | 22500 |

In [10]:
```
credit_card_data.tail()
```

Out[10]:

| | CUST_ID | BALANCE | BALANCE_FREQUENCY | PURCHASES | ONEOFF_PURCHASES | INSTALLMENT |
|---|---|---|---|---|---|---|
| **8945** | C19186 | 28.493517 | 1.000000 | 291.12 | 0.00 | |
| **8946** | C19187 | 19.183215 | 1.000000 | 300.00 | 0.00 | |
| **8947** | C19188 | 23.398673 | 0.833333 | 144.40 | 0.00 | |
| **8948** | C19189 | 13.457564 | 0.833333 | 0.00 | 0.00 | |
| **8949** | C19190 | 372.708075 | 0.666667 | 1093.25 | 1093.25 | |

In [ ]:
```
#these are the columns where we have missing values
#you can check this via dataset_name.isnull().sum()
CREDIT_LIMIT ,MINIMUM_PAYMENTS  ,State
```

In [16]:
```
credit_card_data["CREDIT_LIMIT_new"]=credit_card_data["CREDIT_LIMIT"]
credit_card_data["MINIMUM_PAYMENTS_zero"]=credit_card_data["MINIMUM_PAYMENTS"]
```

In [18]:
```
#fill the values with zero

credit_card_data["CREDIT_LIMIT_new"]=credit_card_data["CREDIT_LIMIT_new"].fillna(0)
credit_card_data["MINIMUM_PAYMENTS_zero"]=credit_card_data["MINIMUM_PAYMENTS_zero"].
```

In [19]:
```
credit_card_data.isnull().sum()
```

Out[19]:
```
CUST_ID                     0
BALANCE                     0
BALANCE_FREQUENCY           0
PURCHASES                   0
ONEOFF_PURCHASES            0
INSTALLMENTS_PURCHASES      0
CASH_ADVANCE                0
PURCHASES_FREQUENCY         0
```

```
ONEOFF_PURCHASES_FREQUENCY            0
PURCHASES_INSTALLMENTS_FREQUENCY      0
CASH_ADVANCE_FREQUENCY                0
CASH_ADVANCE_TRX                      0
PURCHASES_TRX                         0
CREDIT_LIMIT                          1
PAYMENTS                              0
MINIMUM_PAYMENTS                    313
PRC_FULL_PAYMENT                      0
TENURE                                0
State                                 9
CREDIT_LIMIT_new                      0
MINIMUM_PAYMENTS_zero                 0
dtype: int64
```

In [20]:
```
#now create new columns for testing purpose and fill the mean value
credit_card_data["CREDIT_LIMIT_mean"]=credit_card_data["CREDIT_LIMIT"]
credit_card_data["MINIMUM_PAYMENTS_mean"]=credit_card_data["MINIMUM_PAYMENTS"]
```

In [21]:
```
credit_card_data.isnull().sum()
```

Out[21]:
```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        1
PAYMENTS                            0
MINIMUM_PAYMENTS                  313
PRC_FULL_PAYMENT                    0
TENURE                              0
State                               9
CREDIT_LIMIT_new                    0
MINIMUM_PAYMENTS_zero               0
CREDIT_LIMIT_mean                   1
MINIMUM_PAYMENTS_mean             313
dtype: int64
```

In [24]:
```
#replacing null values with mean value

credit_card_data["MINIMUM_PAYMENTS_mean"]=credit_card_data["MINIMUM_PAYMENTS_mean"].
```

In [26]:
```
credit_card_data.isnull().sum()
```

Out[26]:
```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
```

```
ONEOFF_PURCHASES_FREQUENCY            0
PURCHASES_INSTALLMENTS_FREQUENCY      0
CASH_ADVANCE_FREQUENCY                0
CASH_ADVANCE_TRX                      0
PURCHASES_TRX                         0
CREDIT_LIMIT                          1
PAYMENTS                              0
MINIMUM_PAYMENTS                    313
PRC_FULL_PAYMENT                      0
TENURE                                0
State                                 9
CREDIT_LIMIT_new                      0
MINIMUM_PAYMENTS_zero                 0
CREDIT_LIMIT_mean                     1
MINIMUM_PAYMENTS_mean                 0
dtype: int64
```

In [27]:
```
credit_card_data.describe()
```

Out[27]:

| | BALANCE | BALANCE_FREQUENCY | PURCHASES | ONEOFF_PURCHASES | INSTALLMENTS_PUR |
|---|---|---|---|---|---|
| **count** | 8950.000000 | 8950.000000 | 8950.000000 | 8950.000000 | 895 |
| **mean** | 1564.474828 | 0.877271 | 1003.204834 | 592.437371 | 411 |
| **std** | 2081.531879 | 0.236904 | 2136.634782 | 1659.887917 | 904 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0 |
| **25%** | 128.281915 | 0.888889 | 39.635000 | 0.000000 | 0 |
| **50%** | 873.385231 | 1.000000 | 361.280000 | 38.000000 | 89 |
| **75%** | 2054.140036 | 1.000000 | 1110.130000 | 577.405000 | 468 |
| **max** | 19043.138560 | 1.000000 | 49039.570000 | 40761.250000 | 2250 |

8 rows × 21 columns

In [31]:
```
#credit_card_data864["MINIMUM_PAYMENTS_mean"]=credit_card_data864["MINIMUM_PAYMENTS_

filtered_df = credit_card_data[(credit_card_data['MINIMUM_PAYMENTS_mean'] == '864.20
```

In [32]:
```
filtered_df
```

Out[32]:

| CUST_ID | BALANCE | BALANCE_FREQUENCY | PURCHASES | ONEOFF_PURCHASES | INSTALLMENTS_PUR |
|---|---|---|---|---|---|

0 rows × 23 columns

In [33]:
```
credit_card_data["CREDIT_LIMIT_median"]=credit_card_data["CREDIT_LIMIT"]
credit_card_data["MINIMUM_PAYMENTS_median"]=credit_card_data["MINIMUM_PAYMENTS"]
```

In [34]:
```
credit_card_data.isnull().sum()
```

Out[34]:
```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        1
PAYMENTS                            0
MINIMUM_PAYMENTS                  313
PRC_FULL_PAYMENT                    0
TENURE                              0
State                               9
CREDIT_LIMIT_new                    0
MINIMUM_PAYMENTS_zero               0
CREDIT_LIMIT_mean                   1
MINIMUM_PAYMENTS_mean               0
CREDIT_LIMIT_median                 1
MINIMUM_PAYMENTS_median           313
dtype: int64
```

In [35]:
```
#fill missing values with median value

credit_card_data["MINIMUM_PAYMENTS_median"]=credit_card_data["MINIMUM_PAYMENTS_media
```

In [36]:
```
credit_card_data.isnull().sum()
```

Out[36]:
```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        1
PAYMENTS                            0
MINIMUM_PAYMENTS                  313
PRC_FULL_PAYMENT                    0
TENURE                              0
State                               9
CREDIT_LIMIT_new                    0
MINIMUM_PAYMENTS_zero               0
CREDIT_LIMIT_mean                   1
MINIMUM_PAYMENTS_mean               0
CREDIT_LIMIT_median                 1
MINIMUM_PAYMENTS_median             0
dtype: int64
```

In [42]:
```
#fill missing value with mode value
credit_card_data["CREDIT_LIMIT_mode"]=credit_card_data["CREDIT_LIMIT"]
```

```
credit_card_data["MINIMUM_PAYMENTS_mode"]=credit_card_data["MINIMUM_PAYMENTS"]

credit_card_data.isnull().sum()
```

Out[42]:
```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        1
PAYMENTS                            0
MINIMUM_PAYMENTS                  313
PRC_FULL_PAYMENT                    0
TENURE                              0
State                               9
CREDIT_LIMIT_new                    0
MINIMUM_PAYMENTS_zero               0
CREDIT_LIMIT_mean                   1
MINIMUM_PAYMENTS_mean               0
CREDIT_LIMIT_median                 1
MINIMUM_PAYMENTS_median             0
CREDIT_LIMIT_mode                   1
MINIMUM_PAYMENTS_mode             313
dtype: int64
```

In [47]:
```
#fill missing value with mode value
#in mode make sure we need to add[0] in the end otherwise it will incraese the null
credit_card_data["MINIMUM_PAYMENTS_mode"]=credit_card_data["MINIMUM_PAYMENTS_mode"].
```

In [48]:
```
credit_card_data.isnull().sum()
```

Out[48]:
```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        1
PAYMENTS                            0
MINIMUM_PAYMENTS                  313
PRC_FULL_PAYMENT                    0
TENURE                              0
State                               9
CREDIT_LIMIT_new                    0
MINIMUM_PAYMENTS_zero               0
CREDIT_LIMIT_mean                   1
MINIMUM_PAYMENTS_mean               0
```

```
CREDIT_LIMIT_median                 1
MINIMUM_PAYMENTS_median             0
CREDIT_LIMIT_mode                   1
MINIMUM_PAYMENTS_mode               0
dtype: int64
```

In [49]:
```
credit_card_data["CREDIT_LIMIT_still_blank"]=credit_card_data["CREDIT_LIMIT"]
credit_card_data["MINIMUM_PAYMENTS_still_blank"]=credit_card_data["MINIMUM_PAYMENTS"
```

In [50]:
```
credit_card_data.isnull().sum()
```

Out[50]:
```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        1
PAYMENTS                            0
MINIMUM_PAYMENTS                  313
PRC_FULL_PAYMENT                    0
TENURE                              0
State                               9
CREDIT_LIMIT_new                    0
MINIMUM_PAYMENTS_zero               0
CREDIT_LIMIT_mean                   1
MINIMUM_PAYMENTS_mean               0
CREDIT_LIMIT_median                 1
MINIMUM_PAYMENTS_median             0
CREDIT_LIMIT_mode                   1
MINIMUM_PAYMENTS_mode               0
CREDIT_LIMIT_still_blank            1
MINIMUM_PAYMENTS_still_blank      313
dtype: int64
```

In [51]:
```
#fill value with previous row
credit_card_data["MINIMUM_PAYMENTS_still_blank"]=credit_card_data["MINIMUM_PAYMENTS_
```

In [52]:
```
credit_card_data.isnull().sum()
```

Out[52]:
```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
```

```
CREDIT_LIMIT                          1
PAYMENTS                              0
MINIMUM_PAYMENTS                    313
PRC_FULL_PAYMENT                      0
TENURE                                0
State                                 9
CREDIT_LIMIT_new                      0
MINIMUM_PAYMENTS_zero                 0
CREDIT_LIMIT_mean                     1
MINIMUM_PAYMENTS_mean                 0
CREDIT_LIMIT_median                   1
MINIMUM_PAYMENTS_median               0
CREDIT_LIMIT_mode                     1
MINIMUM_PAYMENTS_mode                 0
CREDIT_LIMIT_still_blank              1
MINIMUM_PAYMENTS_still_blank          0
dtype: int64
```

In [53]:
```
#Backward fill (use next row's value)
credit_card_data["CREDIT_LIMIT_still_blank2"]=credit_card_data["CREDIT_LIMIT"]
credit_card_data["MINIMUM_PAYMENTS_still_blank2"]=credit_card_data["MINIMUM_PAYMENTS
```

In [54]:
```
credit_card_data.isnull().sum()
```

Out[54]:
```
CUST_ID                           0
BALANCE                           0
BALANCE_FREQUENCY                 0
PURCHASES                         0
ONEOFF_PURCHASES                  0
INSTALLMENTS_PURCHASES            0
CASH_ADVANCE                      0
PURCHASES_FREQUENCY               0
ONEOFF_PURCHASES_FREQUENCY        0
PURCHASES_INSTALLMENTS_FREQUENCY  0
CASH_ADVANCE_FREQUENCY            0
CASH_ADVANCE_TRX                  0
PURCHASES_TRX                     0
CREDIT_LIMIT                      1
PAYMENTS                          0
MINIMUM_PAYMENTS                313
PRC_FULL_PAYMENT                  0
TENURE                            0
State                             9
CREDIT_LIMIT_new                  0
MINIMUM_PAYMENTS_zero             0
CREDIT_LIMIT_mean                 1
MINIMUM_PAYMENTS_mean             0
CREDIT_LIMIT_median               1
MINIMUM_PAYMENTS_median           0
CREDIT_LIMIT_mode                 1
MINIMUM_PAYMENTS_mode             0
CREDIT_LIMIT_still_blank          1
MINIMUM_PAYMENTS_still_blank      0
CREDIT_LIMIT_still_blank2         1
MINIMUM_PAYMENTS_still_blank2   313
dtype: int64
```

In [55]:
```
credit_card_data["MINIMUM_PAYMENTS_still_blank2"]=credit_card_data["MINIMUM_PAYMENTS
```

In [56]:
```
credit_card_data.isnull().sum()
```

Out[56]:
```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        1
PAYMENTS                            0
MINIMUM_PAYMENTS                  313
PRC_FULL_PAYMENT                    0
TENURE                              0
State                               9
CREDIT_LIMIT_new                    0
MINIMUM_PAYMENTS_zero               0
CREDIT_LIMIT_mean                   1
MINIMUM_PAYMENTS_mean               0
CREDIT_LIMIT_median                 1
MINIMUM_PAYMENTS_median             0
CREDIT_LIMIT_mode                   1
MINIMUM_PAYMENTS_mode               0
CREDIT_LIMIT_still_blank            1
MINIMUM_PAYMENTS_still_blank        0
CREDIT_LIMIT_still_blank2           1
MINIMUM_PAYMENTS_still_blank2       0
dtype: int64
```

In [57]:
```
credit_card_data.head()
```

Out[57]:

| | CUST_ID | BALANCE | BALANCE_FREQUENCY | PURCHASES | ONEOFF_PURCHASES | INSTALLMENTS_I |
|---|---|---|---|---|---|---|
| **0** | C10001 | 40.900749 | 0.818182 | 95.40 | 0.00 | |
| **1** | C10002 | 3202.467416 | 0.909091 | 0.00 | 0.00 | |
| **2** | C10003 | 2495.148862 | 1.000000 | 773.17 | 773.17 | |
| **3** | C10004 | 1666.670542 | 0.636364 | 1499.00 | 1499.00 | |
| **4** | C10005 | 817.714335 | 1.000000 | 16.00 | 16.00 | |

5 rows × 31 columns

In [58]:
```
credit_card_data.columns
```

Out[58]:
```
Index(['CUST_ID', 'BALANCE', 'BALANCE_FREQUENCY', 'PURCHASES',
       'ONEOFF_PURCHASES', 'INSTALLMENTS_PURCHASES', 'CASH_ADVANCE',
       'PURCHASES_FREQUENCY', 'ONEOFF_PURCHASES_FREQUENCY',
       'PURCHASES_INSTALLMENTS_FREQUENCY', 'CASH_ADVANCE_FREQUENCY',
       'CASH_ADVANCE_TRX', 'PURCHASES_TRX', 'CREDIT_LIMIT', 'PAYMENTS',
       'MINIMUM_PAYMENTS', 'PRC_FULL_PAYMENT', 'TENURE', 'State',
       'CREDIT_LIMIT_new', 'MINIMUM_PAYMENTS_zero', 'CREDIT_LIMIT_mean',
       'MINIMUM_PAYMENTS_mean', 'CREDIT_LIMIT_median',
       'MINIMUM_PAYMENTS_median', 'CREDIT_LIMIT_mode', 'MINIMUM_PAYMENTS_mode',
```

```
       'CREDIT_LIMIT_still_blank', 'MINIMUM_PAYMENTS_still_blank',
       'CREDIT_LIMIT_still_blank2', 'MINIMUM_PAYMENTS_still_blank2'],
      dtype='object')
```

In [59]:
```
credit_card_data.dtypes
```

Out[59]:
```
CUST_ID                            object
BALANCE                           float64
BALANCE_FREQUENCY                 float64
PURCHASES                         float64
ONEOFF_PURCHASES                  float64
INSTALLMENTS_PURCHASES            float64
CASH_ADVANCE                      float64
PURCHASES_FREQUENCY               float64
ONEOFF_PURCHASES_FREQUENCY        float64
PURCHASES_INSTALLMENTS_FREQUENCY  float64
CASH_ADVANCE_FREQUENCY            float64
CASH_ADVANCE_TRX                    int64
PURCHASES_TRX                       int64
CREDIT_LIMIT                      float64
PAYMENTS                          float64
MINIMUM_PAYMENTS                  float64
PRC_FULL_PAYMENT                  float64
TENURE                              int64
State                              object
CREDIT_LIMIT_new                  float64
MINIMUM_PAYMENTS_zero             float64
CREDIT_LIMIT_mean                 float64
MINIMUM_PAYMENTS_mean             float64
CREDIT_LIMIT_median               float64
MINIMUM_PAYMENTS_median           float64
CREDIT_LIMIT_mode                 float64
MINIMUM_PAYMENTS_mode             float64
CREDIT_LIMIT_still_blank          float64
MINIMUM_PAYMENTS_still_blank      float64
CREDIT_LIMIT_still_blank2         float64
MINIMUM_PAYMENTS_still_blank2     float64
dtype: object
```

In [60]:
```
credit_card_data["State"].head()
```

Out[60]:
```
0    C@liforni@
1           NaN
2         T3x@s
3           NaN
4      N3w Y*rk
Name: State, dtype: object
```

In [61]:
```
#now fix the expression from here
credit_card_data["State_test"]=credit_card_data["State"]
```

In [62]:
```
credit_card_data["State_test"].head()
```

Out[62]:
```
0    C@liforni@
1           NaN
2         T3x@s
3           NaN
4      N3w Y*rk
Name: State_test, dtype: object
```

In [64]:
```
#here what we removed special char with null values
credit_card_data["State_test"]=credit_card_data["State_test"].str.replace(r'[^a-zA-Z
```

```
C:\Users\GOURAV~1\AppData\Local\Temp/ipykernel_25540/2628724343.py:1: FutureWarning:
The default value of regex will change from True to False in a future version.
  credit_card_data["State_test"]=credit_card_data["State_test"].str.replace(r'[^a-zA-
Z\s]','')
```

In [66]:
```
credit_card_data["State_test"].head()
```

Out[66]:
```
0    Californi
1          NaN
2          Txs
3          NaN
4       Nw Yrk
Name: State_test, dtype: object
```

In [82]:
```
super_store = pd.read_csv(r"A:\LLM\Sample - Superstore.csv", encoding="ISO-8859-1")
```

In [83]:
```
super_store.head()
```

Out[83]:

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country | City | ... | Po |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | CA-2016-152156 | 08-11-2016 | 11-11-2016 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | ... | 4 |
| **1** | 2 | CA-2016-152156 | 08-11-2016 | 11-11-2016 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | ... | 4 |
| **2** | 3 | CA-2016-138688 | 12-06-2016 | 16-06-2016 | Second Class | DV-13045 | Darrin Van Huff | Corporate | United States | Los Angeles | ... | 9 |
| **3** | 4 | US-2015-108966 | 11-10-2015 | 18-10-2015 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | ... | 3 |
| **4** | 5 | US-2015-108966 | 11-10-2015 | 18-10-2015 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | ... | 3 |

5 rows × 21 columns

In [71]:
```
super_store.columns
```

Out[71]:
```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'State',
```

```
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Category',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

In [84]:
```
#extracting only digit only first sequence
super_store["Order ID_new"]=super_store["Order ID"].str.extract('(\d+)')
```

In [85]:
```
super_store["Order ID_new"].head()
```

Out[85]:
```
0    2016
1    2016
2    2016
3    2015
4    2015
Name: Order ID_new, dtype: object
```

In [86]:
```
#extracting only digit only first sequence and second sequence as well
super_store["Order ID_new1"]=super_store["Order ID"].str.extract('(\d+-\d+)')
```

In [87]:
```
super_store["Order ID_new1"].head()
```

Out[87]:
```
0    2016-152156
1    2016-152156
2    2016-138688
3    2015-108966
4    2015-108966
Name: Order ID_new1, dtype: object
```

In [88]:
```
super_store.head()
```

Out[88]:

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country | City | ... | I |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | CA-2016-152156 | 08-11-2016 | 11-11-2016 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | ... | F 1( |
| **1** | 2 | CA-2016-152156 | 08-11-2016 | 11-11-2016 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | ... | F 1( |
| **2** | 3 | CA-2016-138688 | 12-06-2016 | 16-06-2016 | Second Class | DV-13045 | Darrin Van Huff | Corporate | United States | Los Angeles | ... | ( 1( |
| **3** | 4 | US-2015-108966 | 11-10-2015 | 18-10-2015 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | ... | I 1( |
| **4** | 5 | US-2015-108966 | 11-10-2015 | 18-10-2015 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | ... | ( 1( |

5 rows × 23 columns

In [125…
```
credit_card_data["State_test4"]=credit_card_data["State"]
```

In [126…
```
credit_card_data["State_test4"].head()
```

Out[126…
```
0    C@liforni@
1           NaN
2         T3x@s
3           NaN
4      N3w Y*rk
Name: State_test4, dtype: object
```

In [127…
```
#here we need to make sure we need to use only replace not str.reapce when passing m

#credit_card_data["State_test2"]=credit_card_data["State_test2"].replace({'@':'a', '
#put r as well otherwise it will throw an error

credit_card_data["State_test4"] = credit_card_data["State_test4"].replace({r'@':'a',
```

In [128…
```
credit_card_data["State_test4"].head()
```

Out[128…
```
0    California
1           NaN
2         Texas
3           NaN
4      New York
Name: State_test4, dtype: object
```

In [120…
```
#how to fill null values in string columns

credit_card_data["State_test3"]=credit_card_data["State_test3"].fillna(credit_card_d
```

In [121…
```
credit_card_data["State_test3"].head()
```

Out[121…
```
0    California
1    California
2    California
3    California
4    California
Name: State_test3, dtype: object
```

In [107…
```
credit_card_data["State_test2"].tail()
```

Out[107…
```
8945    California
8946    California
8947    California
8948    California
8949    California
Name: State_test2, dtype: object
```

In [108…
```
credit_card_data
```

Out[108…

| | CUST_ID | BALANCE | BALANCE_FREQUENCY | PURCHASES | ONEOFF_PURCHASES | INSTALLMEN |
|---|---|---|---|---|---|---|
| **0** | C10001 | 40.900749 | 0.818182 | 95.40 | 0.00 | |
| **1** | C10002 | 3202.467416 | 0.909091 | 0.00 | 0.00 | |
| **2** | C10003 | 2495.148862 | 1.000000 | 773.17 | 773.17 | |
| **3** | C10004 | 1666.670542 | 0.636364 | 1499.00 | 1499.00 | |
| **4** | C10005 | 817.714335 | 1.000000 | 16.00 | 16.00 | |
| **…** | … | … | … | … | … | |
| **8945** | C19186 | 28.493517 | 1.000000 | 291.12 | 0.00 | |
| **8946** | C19187 | 19.183215 | 1.000000 | 300.00 | 0.00 | |
| **8947** | C19188 | 23.398673 | 0.833333 | 144.40 | 0.00 | |
| **8948** | C19189 | 13.457564 | 0.833333 | 0.00 | 0.00 | |
| **8949** | C19190 | 372.708075 | 0.666667 | 1093.25 | 1093.25 | |

8950 rows × 33 columns

In [111…

```
mode_value=credit_card_data["State_test4"].mode()[0]
```

In [129…

```
credit_card_data["State_test4"]=credit_card_data["State_test4"].fillna(mode_value)
```

In [130…

```
credit_card_data
```

Out[130…

| | CUST_ID | BALANCE | BALANCE_FREQUENCY | PURCHASES | ONEOFF_PURCHASES | INSTALLMEN |
|---|---|---|---|---|---|---|
| **0** | C10001 | 40.900749 | 0.818182 | 95.40 | 0.00 | |
| **1** | C10002 | 3202.467416 | 0.909091 | 0.00 | 0.00 | |
| **2** | C10003 | 2495.148862 | 1.000000 | 773.17 | 773.17 | |
| **3** | C10004 | 1666.670542 | 0.636364 | 1499.00 | 1499.00 | |
| **4** | C10005 | 817.714335 | 1.000000 | 16.00 | 16.00 | |
| **…** | … | … | … | … | … | |
| **8945** | C19186 | 28.493517 | 1.000000 | 291.12 | 0.00 | |
| **8946** | C19187 | 19.183215 | 1.000000 | 300.00 | 0.00 | |
| **8947** | C19188 | 23.398673 | 0.833333 | 144.40 | 0.00 | |
| **8948** | C19189 | 13.457564 | 0.833333 | 0.00 | 0.00 | |
| **8949** | C19190 | 372.708075 | 0.666667 | 1093.25 | 1093.25 | |

8950 rows × 35 columns

In [131…

```
credit_card_data.isnull().sum()
```

Out[131…

```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        1
PAYMENTS                            0
MINIMUM_PAYMENTS                  313
PRC_FULL_PAYMENT                    0
TENURE                              0
State                               9
CREDIT_LIMIT_new                    0
MINIMUM_PAYMENTS_zero               0
CREDIT_LIMIT_mean                   1
MINIMUM_PAYMENTS_mean               0
CREDIT_LIMIT_median                 1
MINIMUM_PAYMENTS_median             0
CREDIT_LIMIT_mode                   1
MINIMUM_PAYMENTS_mode               0
CREDIT_LIMIT_still_blank            1
MINIMUM_PAYMENTS_still_blank        0
CREDIT_LIMIT_still_blank2           1
MINIMUM_PAYMENTS_still_blank2       0
State_test                          9
State_test2                         0
State_test3                         9
State_test4                         0
dtype: int64
```

In [132…

```
credit_card_data_copied=credit_card_data.copy()
```

In [133…

```
credit_card_data_copied
```

Out[133…

| | CUST_ID | BALANCE | BALANCE_FREQUENCY | PURCHASES | ONEOFF_PURCHASES | INSTALLMEN |
|---|---|---|---|---|---|---|
| **0** | C10001 | 40.900749 | 0.818182 | 95.40 | 0.00 | |
| **1** | C10002 | 3202.467416 | 0.909091 | 0.00 | 0.00 | |
| **2** | C10003 | 2495.148862 | 1.000000 | 773.17 | 773.17 | |
| **3** | C10004 | 1666.670542 | 0.636364 | 1499.00 | 1499.00 | |
| **4** | C10005 | 817.714335 | 1.000000 | 16.00 | 16.00 | |
| **…** | … | … | … | … | … | |
| **8945** | C19186 | 28.493517 | 1.000000 | 291.12 | 0.00 | |
| **8946** | C19187 | 19.183215 | 1.000000 | 300.00 | 0.00 | |
| **8947** | C19188 | 23.398673 | 0.833333 | 144.40 | 0.00 | |
| **8948** | C19189 | 13.457564 | 0.833333 | 0.00 | 0.00 | |
| **8949** | C19190 | 372.708075 | 0.666667 | 1093.25 | 1093.25 | |

8950 rows × 35 columns

In [134…
```
credit_card_data_copied.dtypes
```

Out[134…
```
CUST_ID                            object
BALANCE                           float64
BALANCE_FREQUENCY                 float64
PURCHASES                         float64
ONEOFF_PURCHASES                  float64
INSTALLMENTS_PURCHASES            float64
CASH_ADVANCE                      float64
PURCHASES_FREQUENCY               float64
ONEOFF_PURCHASES_FREQUENCY        float64
PURCHASES_INSTALLMENTS_FREQUENCY  float64
CASH_ADVANCE_FREQUENCY            float64
CASH_ADVANCE_TRX                    int64
PURCHASES_TRX                       int64
CREDIT_LIMIT                      float64
PAYMENTS                          float64
MINIMUM_PAYMENTS                  float64
PRC_FULL_PAYMENT                  float64
TENURE                              int64
State                              object
CREDIT_LIMIT_new                  float64
MINIMUM_PAYMENTS_zero             float64
CREDIT_LIMIT_mean                 float64
MINIMUM_PAYMENTS_mean             float64
CREDIT_LIMIT_median               float64
MINIMUM_PAYMENTS_median           float64
CREDIT_LIMIT_mode                 float64
MINIMUM_PAYMENTS_mode             float64
CREDIT_LIMIT_still_blank          float64
MINIMUM_PAYMENTS_still_blank      float64
CREDIT_LIMIT_still_blank2         float64
MINIMUM_PAYMENTS_still_blank2     float64
State_test                         object
State_test2                        object
State_test3                        object
State_test4                        object
dtype: object
```

In [135…
```
#convert column to object

credit_card_data_copied["CUST_ID"]=credit_card_data_copied["CUST_ID"].astype('catego
```

In [136…
```
credit_card_data_copied.dtypes
```

Out[136…
```
CUST_ID                           category
BALANCE                            float64
BALANCE_FREQUENCY                  float64
PURCHASES                          float64
ONEOFF_PURCHASES                   float64
INSTALLMENTS_PURCHASES             float64
CASH_ADVANCE                       float64
PURCHASES_FREQUENCY                float64
ONEOFF_PURCHASES_FREQUENCY         float64
PURCHASES_INSTALLMENTS_FREQUENCY   float64
CASH_ADVANCE_FREQUENCY             float64
CASH_ADVANCE_TRX                     int64
```

```
PURCHASES_TRX                        int64
CREDIT_LIMIT                       float64
PAYMENTS                           float64
MINIMUM_PAYMENTS                   float64
PRC_FULL_PAYMENT                   float64
TENURE                               int64
State                               object
CREDIT_LIMIT_new                   float64
MINIMUM_PAYMENTS_zero              float64
CREDIT_LIMIT_mean                  float64
MINIMUM_PAYMENTS_mean              float64
CREDIT_LIMIT_median                float64
MINIMUM_PAYMENTS_median            float64
CREDIT_LIMIT_mode                  float64
MINIMUM_PAYMENTS_mode              float64
CREDIT_LIMIT_still_blank           float64
MINIMUM_PAYMENTS_still_blank       float64
CREDIT_LIMIT_still_blank2          float64
MINIMUM_PAYMENTS_still_blank2      float64
State_test                          object
State_test2                         object
State_test3                         object
State_test4                         object
dtype: object
```

In [138…

```
credit_card_data_copied["State_test"]=credit_card_data_copied["State_test"].astype('
```

In [139…

```
credit_card_data_copied.dtypes
```

Out[139…

```
CUST_ID                           category
BALANCE                            float64
BALANCE_FREQUENCY                  float64
PURCHASES                          float64
ONEOFF_PURCHASES                   float64
INSTALLMENTS_PURCHASES             float64
CASH_ADVANCE                       float64
PURCHASES_FREQUENCY                float64
ONEOFF_PURCHASES_FREQUENCY         float64
PURCHASES_INSTALLMENTS_FREQUENCY   float64
CASH_ADVANCE_FREQUENCY             float64
CASH_ADVANCE_TRX                     int64
PURCHASES_TRX                        int64
CREDIT_LIMIT                       float64
PAYMENTS                           float64
MINIMUM_PAYMENTS                   float64
PRC_FULL_PAYMENT                   float64
TENURE                               int64
State                               object
CREDIT_LIMIT_new                   float64
MINIMUM_PAYMENTS_zero              float64
CREDIT_LIMIT_mean                  float64
MINIMUM_PAYMENTS_mean              float64
CREDIT_LIMIT_median                float64
MINIMUM_PAYMENTS_median            float64
CREDIT_LIMIT_mode                  float64
MINIMUM_PAYMENTS_mode              float64
CREDIT_LIMIT_still_blank           float64
MINIMUM_PAYMENTS_still_blank       float64
CREDIT_LIMIT_still_blank2          float64
MINIMUM_PAYMENTS_still_blank2      float64
State_test                        category
State_test2                         object
```

```
State_test3                          object
State_test4                          object
dtype: object
```

In [140…

```
#astype is a function in pandas which we are using for conversion one datatype to an
credit_card_data_copied["BALANCE"]=credit_card_data_copied["BALANCE"].astype(int)
```

In [142…

```
credit_card_data_copied.dtypes
```

Out[142…

```
CUST_ID                           category
BALANCE                              int32
BALANCE_FREQUENCY                  float64
PURCHASES                          float64
ONEOFF_PURCHASES                   float64
INSTALLMENTS_PURCHASES             float64
CASH_ADVANCE                       float64
PURCHASES_FREQUENCY                float64
ONEOFF_PURCHASES_FREQUENCY         float64
PURCHASES_INSTALLMENTS_FREQUENCY   float64
CASH_ADVANCE_FREQUENCY             float64
CASH_ADVANCE_TRX                     int64
PURCHASES_TRX                        int64
CREDIT_LIMIT                       float64
PAYMENTS                           float64
MINIMUM_PAYMENTS                   float64
PRC_FULL_PAYMENT                   float64
TENURE                               int64
State                               object
CREDIT_LIMIT_new                   float64
MINIMUM_PAYMENTS_zero              float64
CREDIT_LIMIT_mean                  float64
MINIMUM_PAYMENTS_mean              float64
CREDIT_LIMIT_median                float64
MINIMUM_PAYMENTS_median            float64
CREDIT_LIMIT_mode                  float64
MINIMUM_PAYMENTS_mode              float64
CREDIT_LIMIT_still_blank           float64
MINIMUM_PAYMENTS_still_blank       float64
CREDIT_LIMIT_still_blank2          float64
MINIMUM_PAYMENTS_still_blank2      float64
State_test                        category
State_test2                         object
State_test3                         object
State_test4                         object
dtype: object
```

In [144…

```
credit_card_data_description=credit_card_data_copied.describe()
```

In [145…

```
credit_card_data_description
```

Out[145…

| | BALANCE | BALANCE_FREQUENCY | PURCHASES | ONEOFF_PURCHASES | INSTALLMENTS_PUR |
|---|---|---|---|---|---|
| **count** | 8950.000000 | 8950.000000 | 8950.000000 | 8950.000000 | 8950 |
| **mean** | 1563.983128 | 0.877271 | 1003.204834 | 592.437371 | 411 |
| **std** | 2081.529015 | 0.236904 | 2136.634782 | 1659.887917 | 904 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0 |
| **25%** | 128.000000 | 0.888889 | 39.635000 | 0.000000 | 0 |

| | BALANCE | BALANCE_FREQUENCY | PURCHASES | ONEOFF_PURCHASES | INSTALLMENTS_PUR |
|---|---|---|---|---|---|
| **50%** | 873.000000 | 1.000000 | 361.280000 | 38.000000 | 89 |
| **75%** | 2053.750000 | 1.000000 | 1110.130000 | 577.405000 | 468 |
| **max** | 19043.000000 | 1.000000 | 49039.570000 | 40761.250000 | 22500 |

8 rows × 29 columns

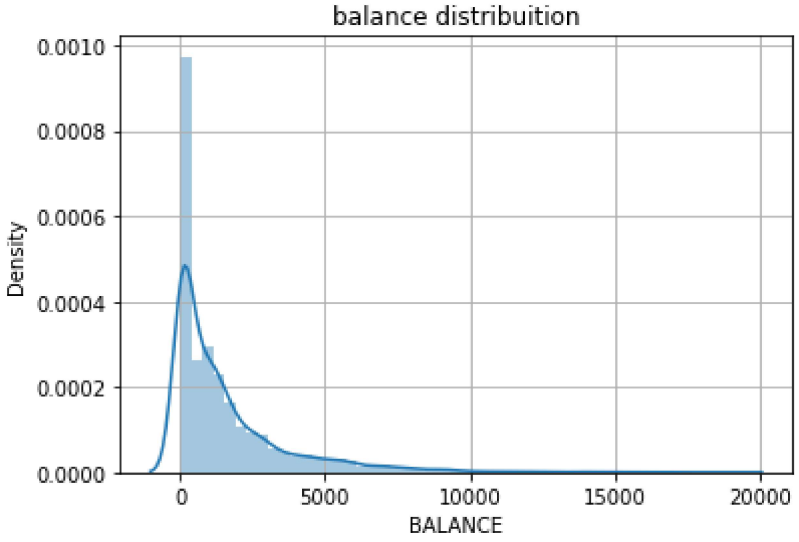
In [146...
```
#if you want to send data csv format use to_csv data
credit_card_data_description.to_csv("credit_card_data_description.csv")
```

In [147...
```
credit_card_data_description.columns
```

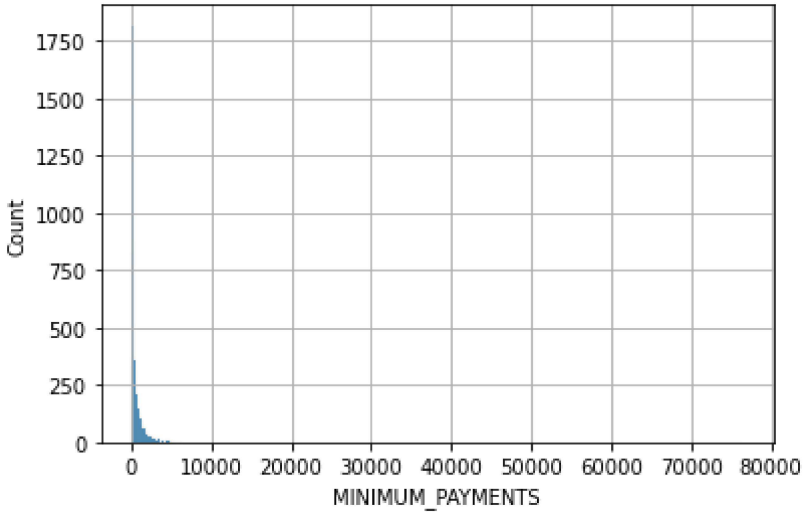
Out[147...
```
Index(['BALANCE', 'BALANCE_FREQUENCY', 'PURCHASES', 'ONEOFF_PURCHASES',
       'INSTALLMENTS_PURCHASES', 'CASH_ADVANCE', 'PURCHASES_FREQUENCY',
       'ONEOFF_PURCHASES_FREQUENCY', 'PURCHASES_INSTALLMENTS_FREQUENCY',
       'CASH_ADVANCE_FREQUENCY', 'CASH_ADVANCE_TRX', 'PURCHASES_TRX',
       'CREDIT_LIMIT', 'PAYMENTS', 'MINIMUM_PAYMENTS', 'PRC_FULL_PAYMENT',
       'TENURE', 'CREDIT_LIMIT_new', 'MINIMUM_PAYMENTS_zero',
       'CREDIT_LIMIT_mean', 'MINIMUM_PAYMENTS_mean', 'CREDIT_LIMIT_median',
       'MINIMUM_PAYMENTS_median', 'CREDIT_LIMIT_mode', 'MINIMUM_PAYMENTS_mode',
       'CREDIT_LIMIT_still_blank', 'MINIMUM_PAYMENTS_still_blank',
       'CREDIT_LIMIT_still_blank2', 'MINIMUM_PAYMENTS_still_blank2'],
      dtype='object')
```

In [152...
```
snm.distplot(credit_card_data["BALANCE"])
mlt.title("balance distribuition")
mlt.grid()
mlt.show()
```

```
C:\Users\Gourav Sikka\anaconda3\lib\site-packages\seaborn\distributions.py:2619: Futu
reWarning: `distplot` is a deprecated function and will be removed in a future versio
n. Please adapt your code to use either `displot` (a figure-level function with simil
ar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

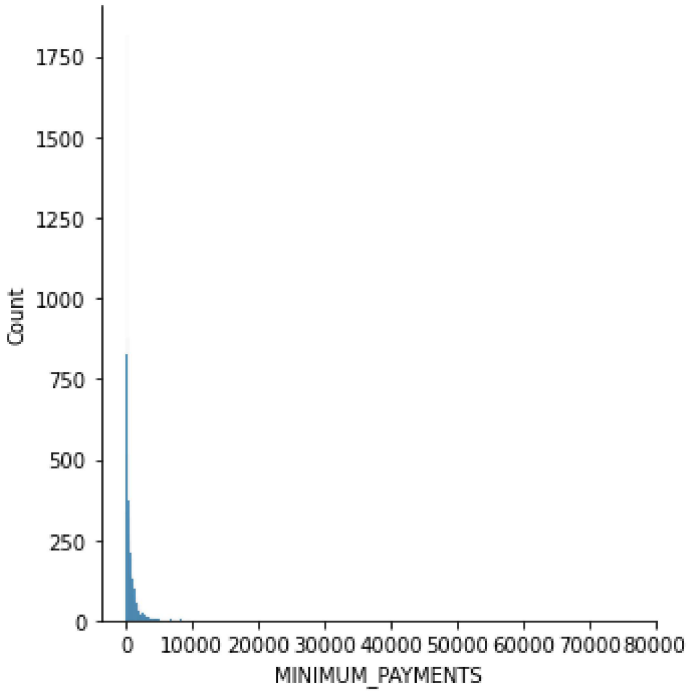



In [163...
```
snm.histplot(credit_card_data["MINIMUM_PAYMENTS"])
mlt.grid()
mlt.show()
```




In [164…

```
snm.displot(credit_card_data["MINIMUM_PAYMENTS"])
mlt.show()
```

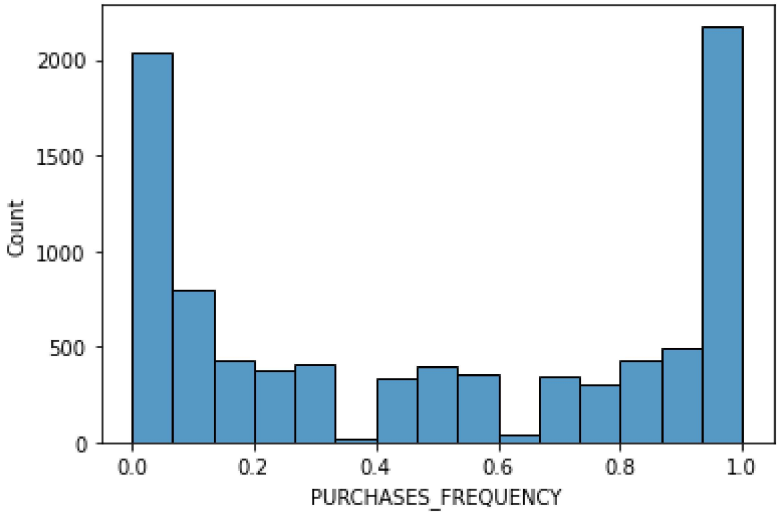



In [166…

```
snm.histplot(credit_card_data["PURCHASES_FREQUENCY"])
```

Out[166…

```
<AxesSubplot:xlabel='PURCHASES_FREQUENCY', ylabel='Count'>
```

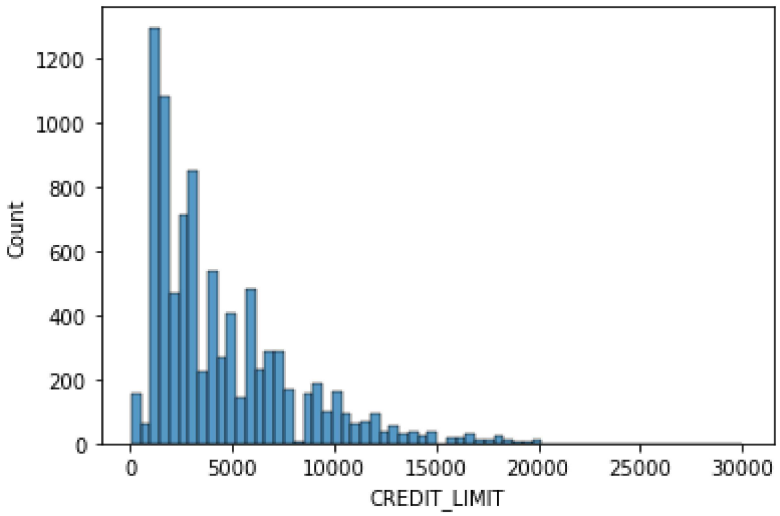



In [167…
```
credit_card_data["PURCHASES_FREQUENCY"].head()
```

Out[167…
```
0    0.166667
1    0.000000
2    1.000000
3    0.083333
4    0.083333
Name: PURCHASES_FREQUENCY, dtype: float64
```

In [168…
```
snm.histplot(credit_card_data["CREDIT_LIMIT"])
```

Out[168…
```
<AxesSubplot:xlabel='CREDIT_LIMIT', ylabel='Count'>
```

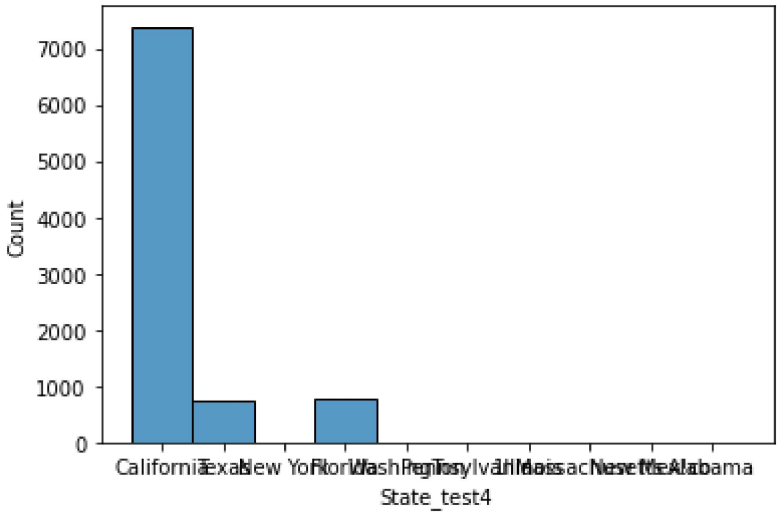



In [170…
```
credit_card_data_copied["State_test4"]
```

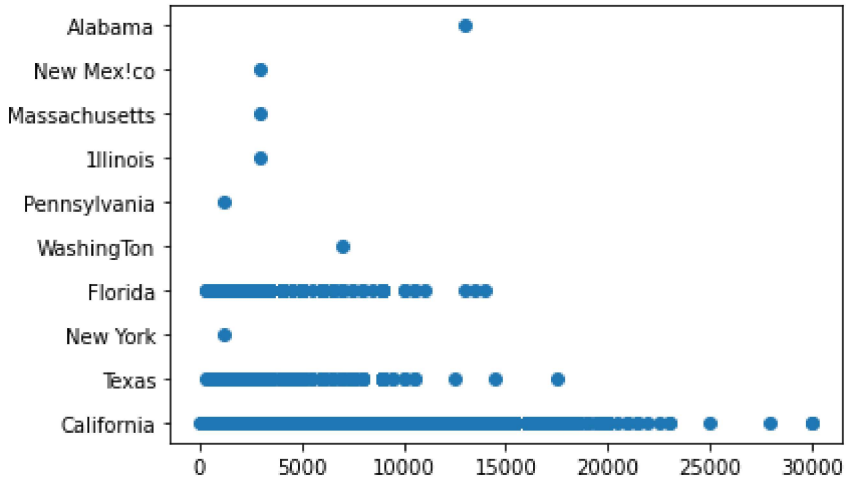
Out[170…
```
0       California
1       California
2            Texas
3       California
4         New York
            ...
8945       Florida
8946       Florida
8947       Florida
8948       Florida
8949       Florida
Name: State_test4, Length: 8950, dtype: object
```

In [173…
```
snm.histplot(credit_card_data_copied["State_test4"])
```

Out[173…
```
<AxesSubplot:xlabel='State_test4', ylabel='Count'>
```




In [179…
```
mlt.scatter(credit_card_data_copied['CREDIT_LIMIT'],credit_card_data_copied['State_t
mlt.show()
```




In [183…
```
snm.pairplot(credit_card_data_copied, hue="State_test4")
plt.show()
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
C:\Users\GOURAV~1\AppData\Local\Temp/ipykernel_25540/1195464031.py in <module>
----> 1 snm.pairplot(credit_card_data_copied, hue="State_test4")
      2 plt.show()

~\anaconda3\lib\site-packages\seaborn\_decorators.py in inner_f(*args, **kwargs)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 46         return f(**kwargs)
     47     return inner_f
     48

~\anaconda3\lib\site-packages\seaborn\axisgrid.py in pairplot(data, hue, hue_order, p
alette, vars, x_vars, y_vars, kind, diag_kind, markers, height, aspect, corner, dropn
a, plot_kws, diag_kws, grid_kws, size)
   2152     # Add a legend
```

```
   2153     if hue is not None:
-> 2154         grid.add_legend()
   2155 
   2156     grid.tight_layout()

~\anaconda3\lib\site-packages\seaborn\axisgrid.py in add_legend(self, legend_data, ti
tle, label_order, adjust_subtitles, **kwargs)
    163 
    164             # Draw the plot to set the bounding boxes correctly
--> 165             _draw_figure(self._figure)
    166 
    167             # Calculate and set the new width of the figure so the legend fit
s

~\anaconda3\lib\site-packages\seaborn\utils.py in _draw_figure(fig)
     93     """Force draw of a matplotlib figure, accounting for back-compat."""
     94     # See https://github.com/matplotlib/matplotlib/issues/19197 for context
---> 95     fig.canvas.draw()
     96     if fig.stale:
     97         try:

~\anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py in draw(self)
    404              (self.toolbar._wait_cursor_for_draw_cm() if self.toolbar
    405               else nullcontext()):
--> 406             self.figure.draw(self.renderer)
    407             # A GUI class may be need to update a window using this draw, so
    408             # don't forget to call the superclass.

~\anaconda3\lib\site-packages\matplotlib\artist.py in draw_wrapper(artist, renderer,
*args, **kwargs)
     72     @wraps(draw)
     73     def draw_wrapper(artist, renderer, *args, **kwargs):
---> 74         result = draw(artist, renderer, *args, **kwargs)
     75         if renderer._rasterizing:
     76             renderer.stop_rasterizing()

~\anaconda3\lib\site-packages\matplotlib\artist.py in draw_wrapper(artist, renderer,
*args, **kwargs)
     49                 renderer.start_filter()
     50 
---> 51             return draw(artist, renderer, *args, **kwargs)
     52         finally:
     53             if artist.get_agg_filter() is not None:

~\anaconda3\lib\site-packages\matplotlib\figure.py in draw(self, renderer)
   2788 
   2789             self.patch.draw(renderer)
-> 2790             mimage._draw_list_compositing_images(
   2791                 renderer, self, artists, self.suppressComposite)
   2792 

~\anaconda3\lib\site-packages\matplotlib\image.py in _draw_list_compositing_images(re
nderer, parent, artists, suppress_composite)
    130     if not_composite or not has_images:
    131         for a in artists:
--> 132             a.draw(renderer)
    133     else:
    134         # Composite any adjacent images together

~\anaconda3\lib\site-packages\matplotlib\artist.py in draw_wrapper(artist, renderer,
*args, **kwargs)
     49                 renderer.start_filter()
     50 
---> 51             return draw(artist, renderer, *args, **kwargs)
```

```
     52         finally:
     53             if artist.get_agg_filter() is not None:

~\anaconda3\lib\site-packages\matplotlib\_api\deprecation.py in wrapper(*inner_args,
**inner_kwargs)
    429                             else deprecation_addendum,
    430                 **kwargs)
--> 431         return func(*inner_args, **inner_kwargs)
    432
    433     return wrapper

~\anaconda3\lib\site-packages\matplotlib\axes\_base.py in draw(self, renderer, infram
e)
   2919             renderer.stop_rasterizing()
   2920
-> 2921         mimage._draw_list_compositing_images(renderer, self, artists)
   2922
   2923         renderer.close_group('axes')

~\anaconda3\lib\site-packages\matplotlib\image.py in _draw_list_compositing_images(re
nderer, parent, artists, suppress_composite)
    130     if not_composite or not has_images:
    131         for a in artists:
--> 132             a.draw(renderer)
    133     else:
    134         # Composite any adjacent images together

~\anaconda3\lib\site-packages\matplotlib\artist.py in draw_wrapper(artist, renderer,
*args, **kwargs)
     49                 renderer.start_filter()
     50
---> 51             return draw(artist, renderer, *args, **kwargs)
     52         finally:
     53             if artist.get_agg_filter() is not None:

~\anaconda3\lib\site-packages\matplotlib\collections.py in draw(self, renderer)
   1010     def draw(self, renderer):
   1011         self.set_sizes(self._sizes, self.figure.dpi)
-> 1012         super().draw(renderer)
   1013
   1014

~\anaconda3\lib\site-packages\matplotlib\artist.py in draw_wrapper(artist, renderer,
*args, **kwargs)
     49                 renderer.start_filter()
     50
---> 51             return draw(artist, renderer, *args, **kwargs)
     52         finally:
     53             if artist.get_agg_filter() is not None:

~\anaconda3\lib\site-packages\matplotlib\collections.py in draw(self, renderer)
    408                 mpath.Path(offsets), transOffset, tuple(facecolors[0]))
    409         else:
--> 410             renderer.draw_path_collection(
    411                 gc, transform.frozen(), paths,
    412                 self.get_transforms(), offsets, transOffset,

~\anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py in draw_path_collect
ion(self, gc, master_transform, paths, all_transforms, offsets, offsetTrans, facecolo
rs, edgecolors, linewidths, linestyles, antialiaseds, urls, offset_position)
    172                 "3.3", message="Support for offset_position='data' is "
    173                 "deprecated since %(since)s and will be removed %(removal)
s.")
--> 174         return self._renderer.draw_path_collection(
```

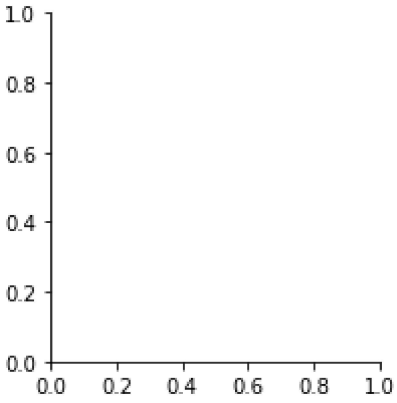
```
    175             gc, master_transform, paths, all_transforms, offsets, offsetTran
s,
    176             facecolors, edgecolors, linewidths, linestyles, antialiaseds, url
s,

~\anaconda3\lib\site-packages\matplotlib\path.py in vertices(self)
    197         )
    198
--> 199     @property
    200     def vertices(self):
    201         """

KeyboardInterrupt:
Error in callback <function flush_figures at 0x00000238F65C9B80> (for post_execute):
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
~\anaconda3\lib\site-packages\matplotlib_inline\backend_inline.py in flush_figures()
    119         # ignore the tracking, just draw and close all figures
    120         try:
--> 121             return show(True)
    122         except Exception as e:
    123             # safely show traceback if in IPython, else raise

~\anaconda3\lib\site-packages\matplotlib_inline\backend_inline.py in show(close, bloc
k)
     39     try:
     40         for figure_manager in Gcf.get_all_fig_managers():
---> 41             display(
     42                 figure_manager.canvas.figure,
     43                 metadata=_fetch_figure_metadata(figure_manager.canvas.figure)

~\anaconda3\lib\site-packages\IPython\core\display.py in display(include, exclude, me
tadata, transient, display_id, *objs, **kwargs)
    318             publish_display_data(data=obj, metadata=metadata, **kwargs)
    319         else:
--> 320             format_dict, md_dict = format(obj, include=include, exclude=exclu
de)
    321             if not format_dict:
    322                 # nothing to display (e.g. _ipython_display_ took over)

~\anaconda3\lib\site-packages\IPython\core\formatters.py in format(self, obj, includ
e, exclude)
    178             md = None
    179             try:
--> 180                 data = formatter(obj)
    181             except:
    182                 # FIXME: log the exception

~\anaconda3\lib\site-packages\decorator.py in fun(*args, **kw)
    230             if not kwsyntax:
    231                 args, kw = fix(args, kw, sig)
--> 232             return caller(func, *(extras + args), **kw)
    233     fun.__name__ = func.__name__
    234     fun.__doc__ = func.__doc__

~\anaconda3\lib\site-packages\IPython\core\formatters.py in catch_format_error(metho
d, self, *args, **kwargs)
    222     """show traceback on failed format call"""
    223     try:
--> 224         r = method(self, *args, **kwargs)
    225     except NotImplementedError:
    226         # don't warn on NotImplementedErrors

~\anaconda3\lib\site-packages\IPython\core\formatters.py in __call__(self, obj)
```

```
    339                 pass
    340             else:
--> 341                 return printer(obj)
    342             # Finally look for special method names
    343             method = get_real_method(obj, self.print_method)

~\anaconda3\lib\site-packages\IPython\core\pylabtools.py in print_figure(fig, fmt, bb
ox_inches, base64, **kwargs)
    149         FigureCanvasBase(fig)
    150
--> 151     fig.canvas.print_figure(bytes_io, **kw)
    152     data = bytes_io.getvalue()
    153     if fmt == 'svg':

~\anaconda3\lib\site-packages\matplotlib\backend_bases.py in print_figure(self, filen
ame, dpi, facecolor, edgecolor, orientation, format, bbox_inches, pad_inches, bbox_ex
tra_artists, backend, **kwargs)
   2232             if bbox_inches:
   2233                 if bbox_inches == "tight":
-> 2234                     bbox_inches = self.figure.get_tightbbox(
   2235                         renderer, bbox_extra_artists=bbox_extra_artists)
   2236                     if pad_inches is None:

~\anaconda3\lib\site-packages\matplotlib\figure.py in get_tightbbox(self, renderer, b
box_extra_artists)
   1637
   1638         for a in artists:
-> 1639             bbox = a.get_tightbbox(renderer)
   1640             if bbox is not None and (bbox.width != 0 or bbox.height != 0):
   1641                 bb.append(bbox)

~\anaconda3\lib\site-packages\matplotlib\axes\_base.py in get_tightbbox(self, rendere
r, call_axes_locator, bbox_extra_artists, for_layout_only)
   4469             clip_extent = a._get_clipping_extent_bbox()
   4470             if clip_extent is not None:
-> 4471                 clip_extent = mtransforms.Bbox.intersection(
   4472                     clip_extent, axbbox)
   4473                 if np.all(clip_extent.extents == axbbox.extents):

~\anaconda3\lib\site-packages\matplotlib\transforms.py in intersection(bbox1, bbox2)
    690         None if they don't.
    691         """
--> 692         x0 = np.maximum(bbox1.xmin, bbox2.xmin)
    693         x1 = np.minimum(bbox1.xmax, bbox2.xmax)
    694         y0 = np.maximum(bbox1.ymin, bbox2.ymin)

~\anaconda3\lib\site-packages\matplotlib\transforms.py in xmin(self)
    327     def xmin(self):
    328         """The left edge of the bounding box."""
--> 329         return np.min(self.get_points()[:, 0])
    330
    331     @property

~\anaconda3\lib\site-packages\matplotlib\transforms.py in get_points(self)
   1094             # from the result, taking care to make the orientation the
   1095             # same.
-> 1096             points = self._transform.transform(
   1097                 [[p[0, 0], p[0, 1]],
   1098                  [p[1, 0], p[0, 1]],

~\anaconda3\lib\site-packages\matplotlib\transforms.py in transform(self, values)
   1468
   1469         # Transform the values
-> 1470         res = self.transform_affine(self.transform_non_affine(values))
```

```
   1471
   1472         # Convert the result back to the shape of the input values.

~\anaconda3\lib\site-packages\matplotlib\transforms.py in transform_affine(self, poin
ts)
   2377     def transform_affine(self, points):
   2378         # docstring inherited
-> 2379         return self.get_affine().transform(points)
   2380
   2381     def transform_non_affine(self, points):

~\anaconda3\lib\site-packages\matplotlib\transforms.py in get_affine(self)
   2405         else:
   2406             return Affine2D(np.dot(self._b.get_affine().get_matrix(),
-> 2407                                    self._a.get_affine().get_matrix()))
   2408
   2409     def inverted(self):

~\anaconda3\lib\site-packages\matplotlib\transforms.py in get_affine(self)
   2404             return self._b.get_affine()
   2405         else:
-> 2406             return Affine2D(np.dot(self._b.get_affine().get_matrix(),
   2407                                    self._a.get_affine().get_matrix()))
   2408

<__array_function__ internals> in dot(*args, **kwargs)

KeyboardInterrupt:
```

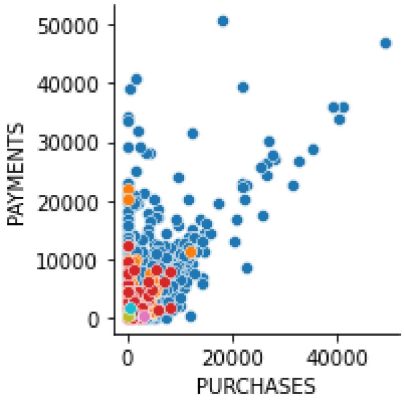
In [186…

```
snm.FacetGrid(credit_card_data_copied, hue="State_test4")
mlt.show()
```




In [192…

```
g=snm.FacetGrid(credit_card_data_copied,hue='State_test4')
g.map_dataframe(snm.scatterplot, x="PURCHASES", y="PAYMENTS")
mlt.show()
```




pd.get_dummies() in Pandas The pd.get_dummies() function is used to convert categorical data into dummy/indicator variables (one-hot encoding). This is useful for machine learning models that work with numerical data.

In [197…

```
data2=pd.get_dummies(credit_card_data_copied,columns=["State_test4"], drop_first=Tru
data2
```

Out[197…

| | CUST_ID | BALANCE | BALANCE_FREQUENCY | PURCHASES | ONEOFF_PURCHASES | INSTALLMENTS |
|---|---|---|---|---|---|---|
| **0** | C10001 | 40 | 0.818182 | 95.40 | 0.00 | |
| **1** | C10002 | 3202 | 0.909091 | 0.00 | 0.00 | |
| **2** | C10003 | 2495 | 1.000000 | 773.17 | 773.17 | |
| **3** | C10004 | 1666 | 0.636364 | 1499.00 | 1499.00 | |
| **4** | C10005 | 817 | 1.000000 | 16.00 | 16.00 | |
| **…** | … | … | … | … | … | |
| **8945** | C19186 | 28 | 1.000000 | 291.12 | 0.00 | |
| **8946** | C19187 | 19 | 1.000000 | 300.00 | 0.00 | |
| **8947** | C19188 | 23 | 0.833333 | 144.40 | 0.00 | |
| **8948** | C19189 | 13 | 0.833333 | 0.00 | 0.00 | |
| **8949** | C19190 | 372 | 0.666667 | 1093.25 | 1093.25 | |

8950 rows × 43 columns

In [ ]: