In [55]:

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

In [56]:

```
t=pd.read_csv('transaction.csv')
t
```

Out[56]:

	accountNumber	customerId	creditLimit	availableMoney	transactionDateTime	transacti
0	737265056	737265056	5000	5000.00	2016-08-13T14:27:32	
1	737265056	737265056	5000	5000.00	2016-10-11T05:05:54	
2	737265056	737265056	5000	5000.00	2016-11-08T09:18:39	
3	737265056	737265056	5000	5000.00	2016-12-10T02:14:50	
4	830329091	830329091	5000	5000.00	2016-03-24T21:04:46	
...	...	...	...	...	...	
786358	732852505	732852505	50000	48904.96	2016-12-22T18:44:12	
786359	732852505	732852505	50000	48785.04	2016-12-25T16:20:34	
786360	732852505	732852505	50000	48766.15	2016-12-27T15:46:24	
786361	732852505	732852505	50000	48716.72	2016-12-29T00:30:55	
786362	732852505	732852505	50000	48666.83	2016-12-30T20:10:29	

786363 rows × 29 columns

In [57]:

```
((t.isna().sum()/len(t))*100).sort_values(ascending=False)
```

Out[57]:

```
recurringAuthInd            100.000000
posOnPremises               100.000000
merchantZip                 100.000000
merchantState               100.000000
merchantCity                100.000000
echoBuffer                  100.000000
acqCountry                    0.580139
posEntryMode                  0.515538
merchantCountryCode           0.092069
transactionType               0.088763
posConditionCode              0.052012
accountNumber                 0.000000
cardLast4Digits               0.000000
expirationDateKeyInMatch      0.000000
cardPresent                   0.000000
currentBalance                0.000000
dateOfLastAddressChange       0.000000
```

In [58]:

```
t1=t.drop(['recurringAuthInd','posOnPremises','merchantZip','merchantState','merchantCity',
t1
```

Out[58]:

	creditLimit	availableMoney	transactionDateTime	transactionAmount	merchantName	ac
0	5000	5000.00	2016-08-13T14:27:32	98.55	Uber	
1	5000	5000.00	2016-10-11T05:05:54	74.51	AMC #191138	
2	5000	5000.00	2016-11-08T09:18:39	7.47	Play Store	
3	5000	5000.00	2016-12-10T02:14:50	7.47	Play Store	
4	5000	5000.00	2016-03-24T21:04:46	71.18	Tim Hortons #947751	
...	...	...	...	...	...	
786358	50000	48904.96	2016-12-22T18:44:12	119.92	Lyft	
786359	50000	48785.04	2016-12-25T16:20:34	18.89	hulu.com	
786360	50000	48766.15	2016-12-27T15:46:24	49.43	Lyft	
786361	50000	48716.72	2016-12-29T00:30:55	49.89	walmart.com	
786362	50000	48666.83	2016-12-30T20:10:29	72.18	Uber	

786363 rows × 21 columns

In [59]:

```
t1.columns
```

Out[59]:

```
Index(['creditLimit', 'availableMoney', 'transactionDateTime',
       'transactionAmount', 'merchantName', 'acqCountry',
       'merchantCountryCode', 'posEntryMode', 'posConditionCode',
       'merchantCategoryCode', 'currentExpDate', 'accountOpenDate',
       'dateOfLastAddressChange', 'cardCVV', 'enteredCVV', 'cardLast4Digit
s',
       'transactionType', 'currentBalance', 'cardPresent',
       'expirationDateKeyInMatch', 'isFraud'],
      dtype='object')
```

In [60]:

```
t1['transactionDateTime']=pd.to_datetime(t.transactionDateTime)
t1['currentExpDate']=pd.to_datetime(t.currentExpDate)
t1['accountOpenDate']=pd.to_datetime(t.accountOpenDate)
t1['dateOfLastAddressChange']=pd.to_datetime(t.dateOfLastAddressChange)
t1.dtypes
```

Out[60]:

```
creditLimit                      int64
availableMoney                 float64
transactionDateTime     datetime64[ns]
transactionAmount              float64
merchantName                    object
acqCountry                      object
merchantCountryCode             object
posEntryMode                   float64
posConditionCode               float64
merchantCategoryCode            object
currentExpDate          datetime64[ns]
accountOpenDate         datetime64[ns]
dateOfLastAddressChange datetime64[ns]
cardCVV                          int64
enteredCVV                       int64
cardLast4Digits                  int64
transactionType                 object
currentBalance                 float64
cardPresent                       bool
expirationDateKeyInMatch          bool
isFraud                           bool
dtype: object
```

Exploratory Data Analysis

In [61]:

```
##we want visualize how different variables contributed to fraud
```

In [62]:

```
import matplotlib.pyplot as plt
import seaborn as sb
```

In [63]:

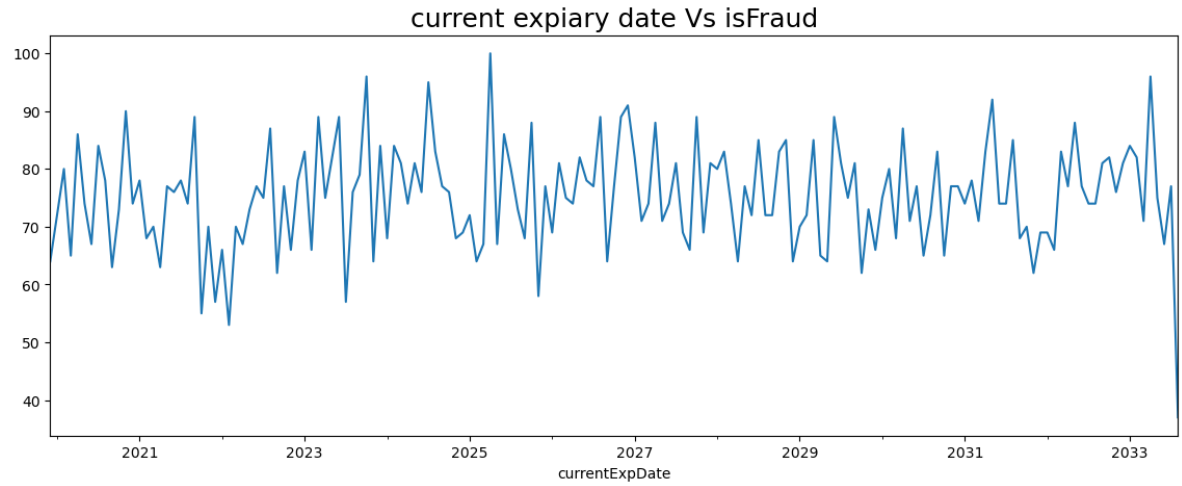
```
t1.merchantName.nunique()
```

Out[63]:

```
2490
```

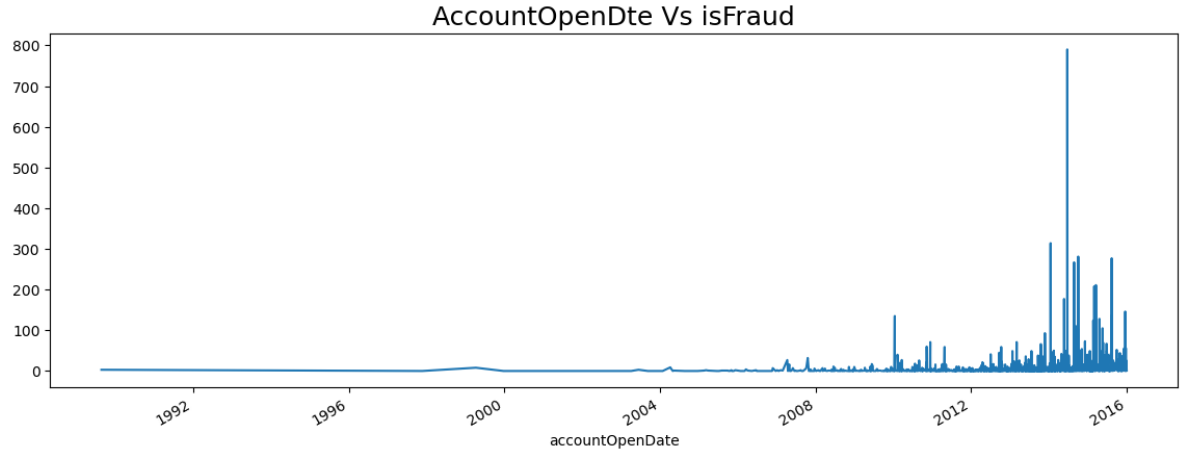
In [64]:

```
plt.figure(figsize=(14,5))
t1.groupby('currentExpDate')['isFraud'].sum().plot();
plt.title('current expiary date Vs isFraud',fontsize=18,color='k');
```



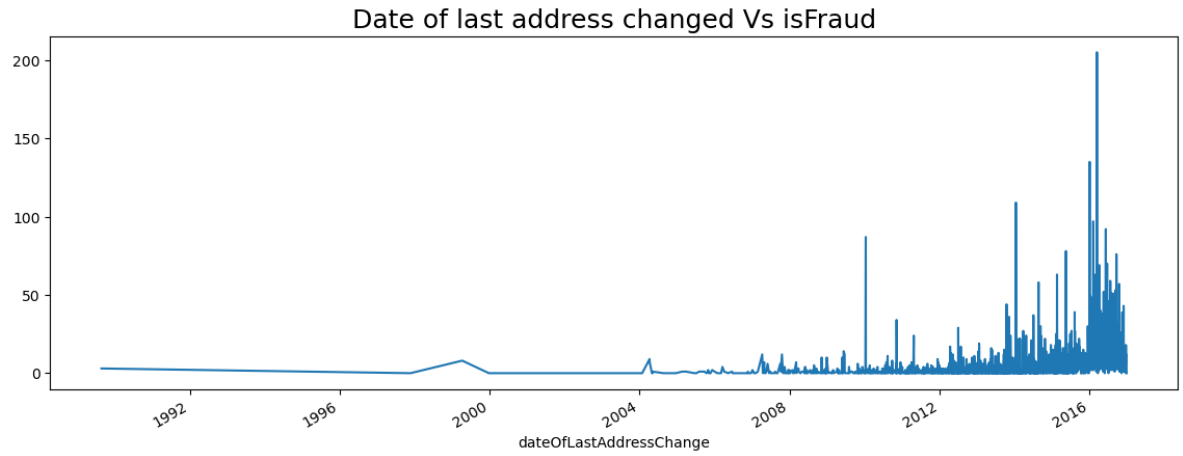
In [65]:

```
plt.figure(figsize=(14,5))
t1.groupby('accountOpenDate')['isFraud'].sum().plot();
plt.title('AccountOpenDte Vs isFraud',fontsize=18);
```



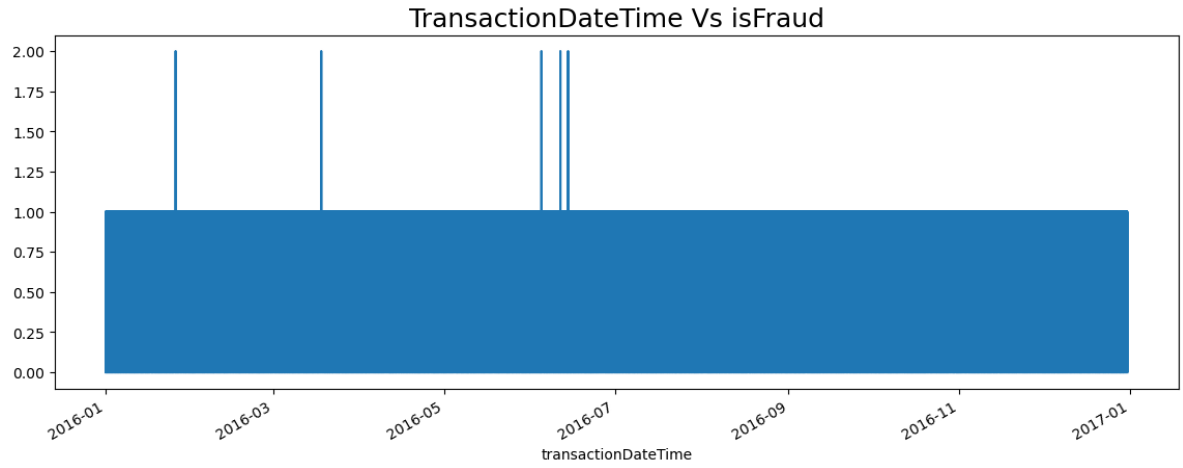
In [66]:

```
plt.figure(figsize=(14,5))
t1.groupby('dateOfLastAddressChange')['isFraud'].sum().plot()
plt.title('Date of last address changed Vs isFraud',color='k',fontsize=18);
```



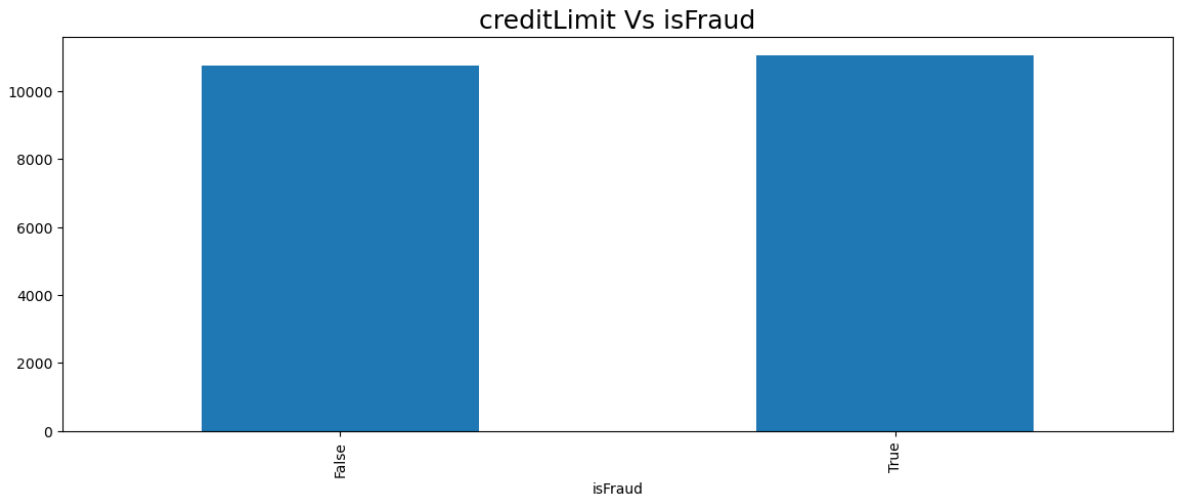
In [67]:

```
plt.figure(figsize=(14,5))
t1.groupby('transactionDateTime')['isFraud'].sum().plot();
plt.title('TransactionDateTime Vs isFraud',color='k',fontsize=18);
```



In [68]:

```
plt.figure(figsize=(14,5))
t1.groupby('isFraud')['creditLimit'].mean().plot.bar()
plt.title('creditLimit Vs isFraud',fontsize=18,color='k');
```



In [69]:

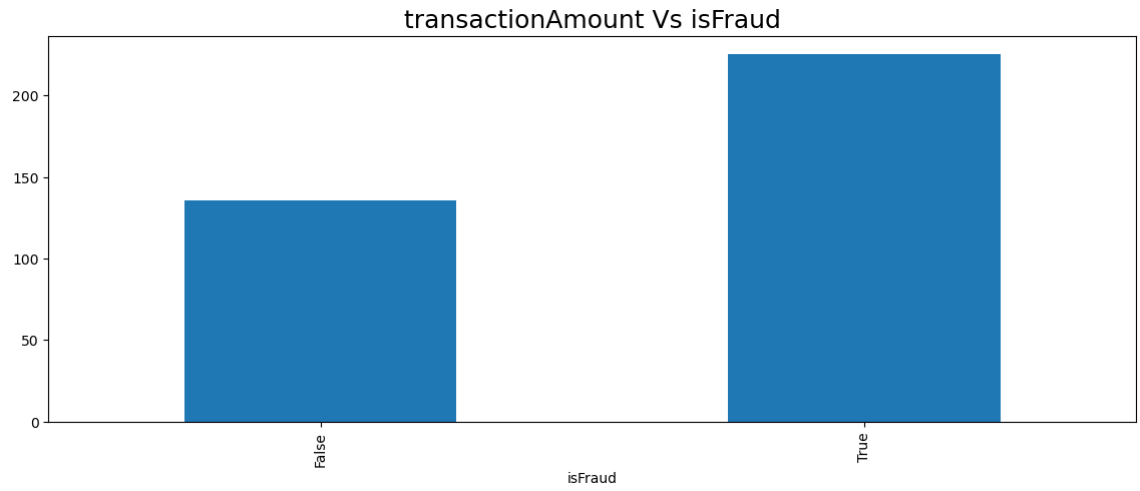
```
t1.groupby('isFraud')['creditLimit'].mean()
```

Out[69]:

```
isFraud
False    10754.884062
True     11044.958525
Name: creditLimit, dtype: float64
```

In [70]:

```
plt.figure(figsize=(14,5))
t1.groupby('isFraud')['transactionAmount'].mean().plot.bar();
plt.title('transactionAmount Vs isFraud',fontsize=18,color='k');
```



In [71]:

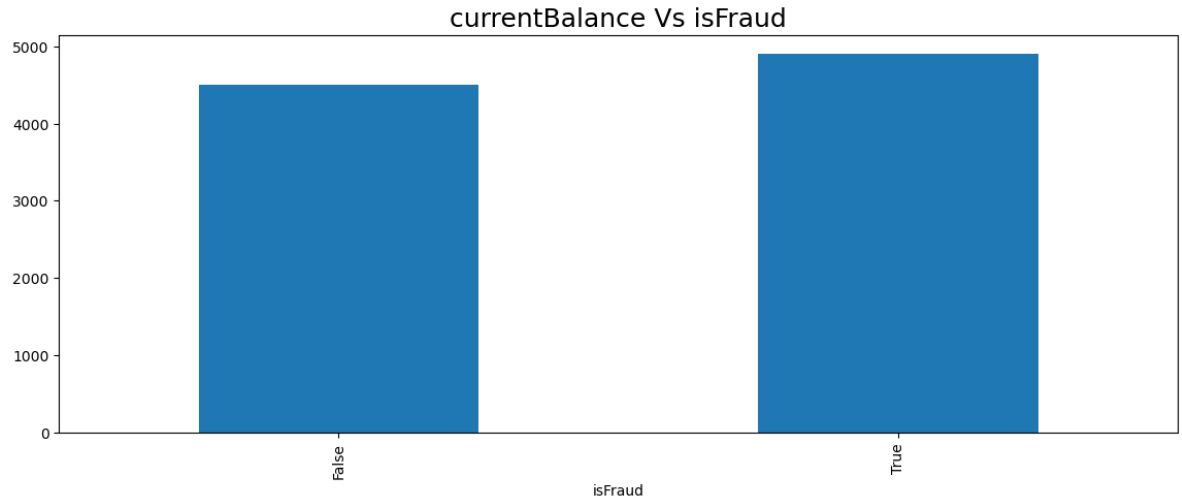
```
t1.groupby('isFraud')['transactionAmount'].mean()
```

Out[71]:

```
isFraud
False    135.570249
True     225.215905
Name: transactionAmount, dtype: float64
```

In [72]:

```
plt.figure(figsize=(14,5))
t1.groupby('isFraud')['currentBalance'].mean().plot.bar();
plt.title('currentBalance Vs isFraud',fontsize=18,color='k');
```



In [73]:

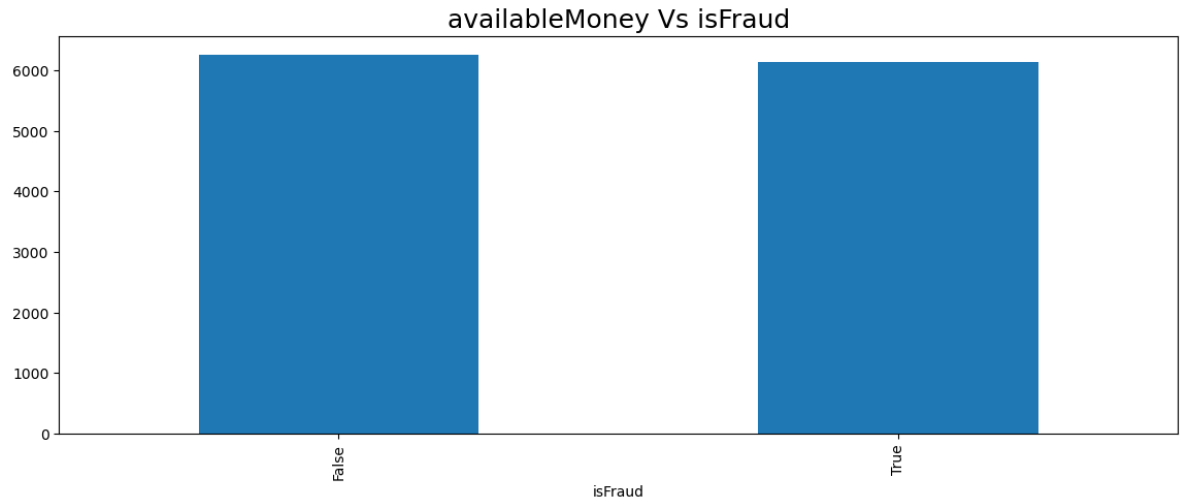
```
t1.groupby('isFraud')['currentBalance'].mean()
```

Out[73]:

```
isFraud
False    4502.428675
True     4902.064338
Name: currentBalance, dtype: float64
```

In [74]:

```
plt.figure(figsize=(14,5))
t1.groupby('isFraud')['availableMoney'].mean().plot.bar();
plt.title('availableMoney Vs isFraud',fontsize=18,color='k');
```



In [75]:

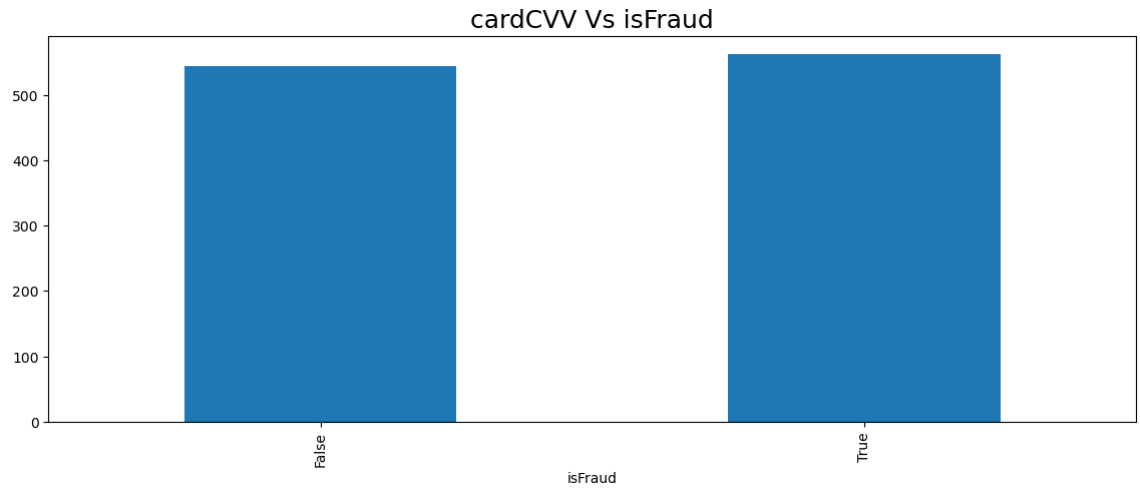
```
t1.groupby('isFraud')['availableMoney'].mean()
```

Out[75]:

```
isFraud
False    6252.455386
True     6142.894186
Name: availableMoney, dtype: float64
```

In [76]:

```
plt.figure(figsize=(14,5))
t1.groupby('isFraud')['cardCVV'].mean().plot.bar();
plt.title('cardCVV Vs isFraud',fontsize=18,color='k');
```



In [77]:

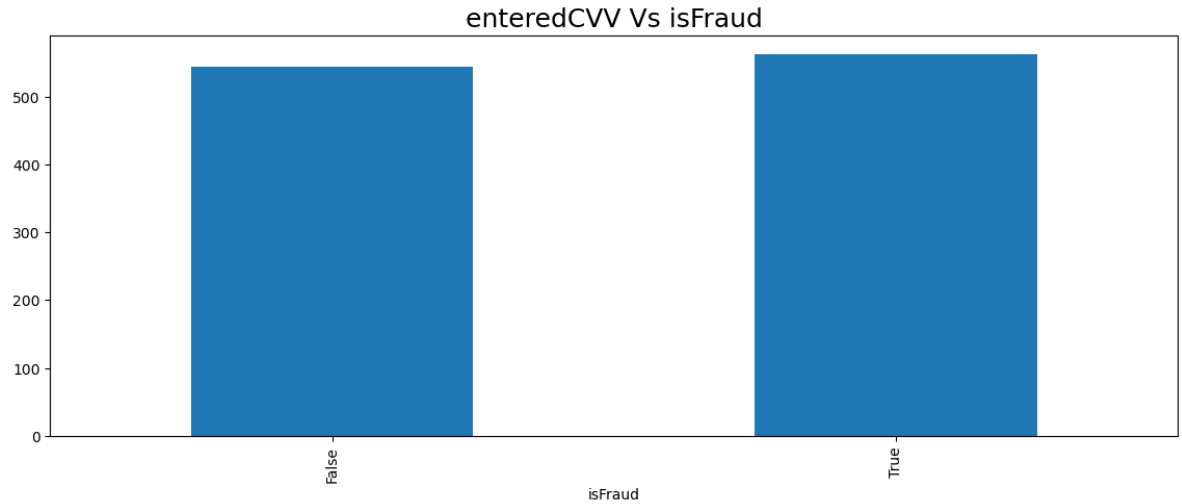
```
t1.groupby('isFraud')['cardCVV'].mean()
```

Out[77]:

```
isFraud
False    544.180723
True     562.331884
Name: cardCVV, dtype: float64
```

In [78]:

```
plt.figure(figsize=(14,5))
t1.groupby('isFraud')['enteredCVV'].mean().plot.bar();
plt.title('enteredCVV Vs isFraud',fontsize=18,color='k');
```



In [79]:

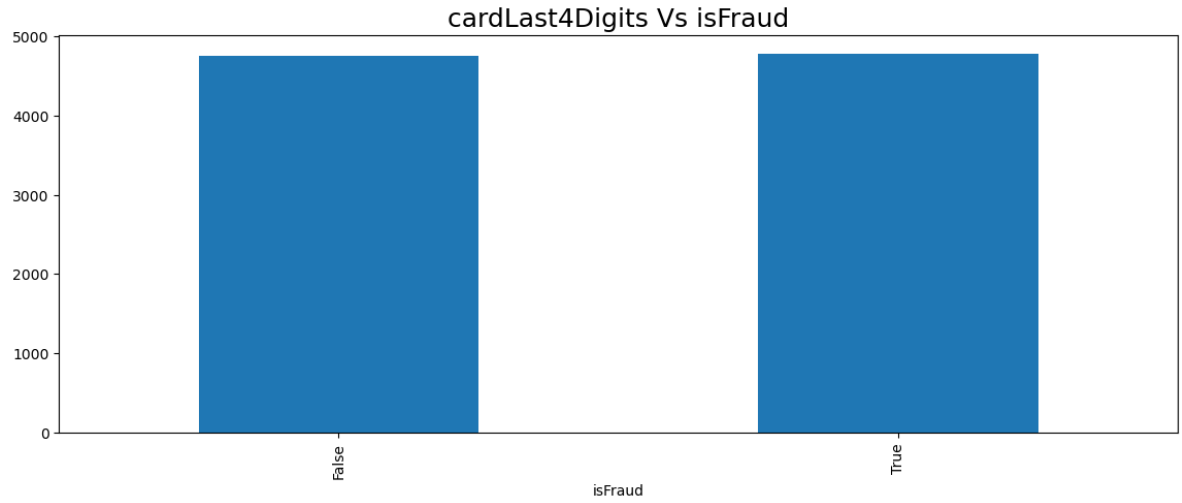
```
t1.groupby('isFraud')['enteredCVV'].mean()
```

Out[79]:

```
isFraud
False    543.897110
True     562.056616
Name: enteredCVV, dtype: float64
```

In [80]:

```
plt.figure(figsize=(14,5))
t1.groupby('isFraud')['cardLast4Digits'].mean().plot.bar();
plt.title('cardLast4Digits Vs isFraud',fontsize=18,color='k');
```



In [81]:

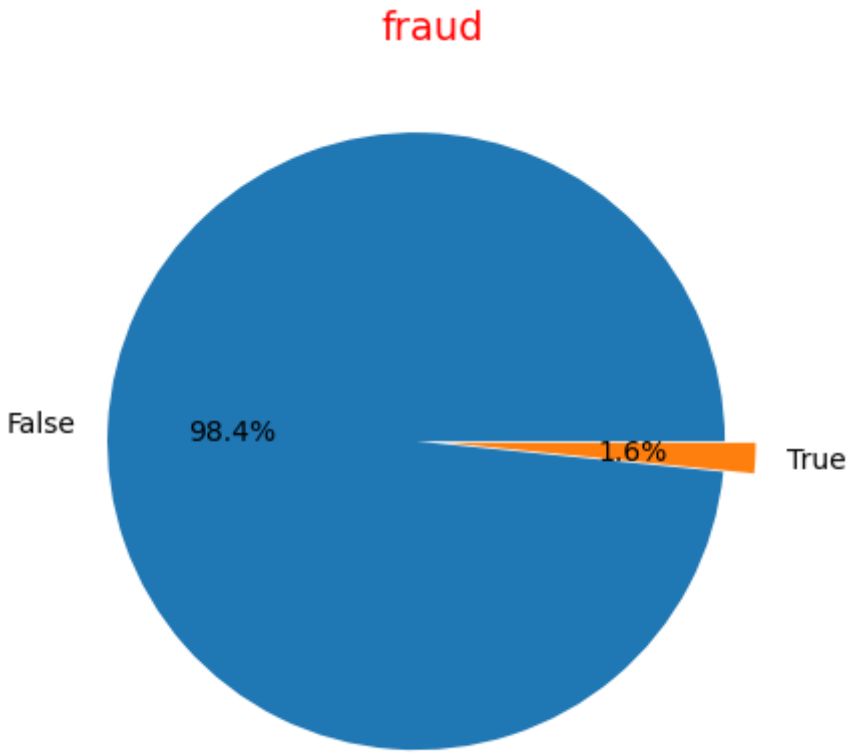
```
t1.groupby('isFraud')['cardLast4Digits'].mean()
```

Out[81]:

```
isFraud
False    4757.081613
True     4778.372151
Name: cardLast4Digits, dtype: float64
```

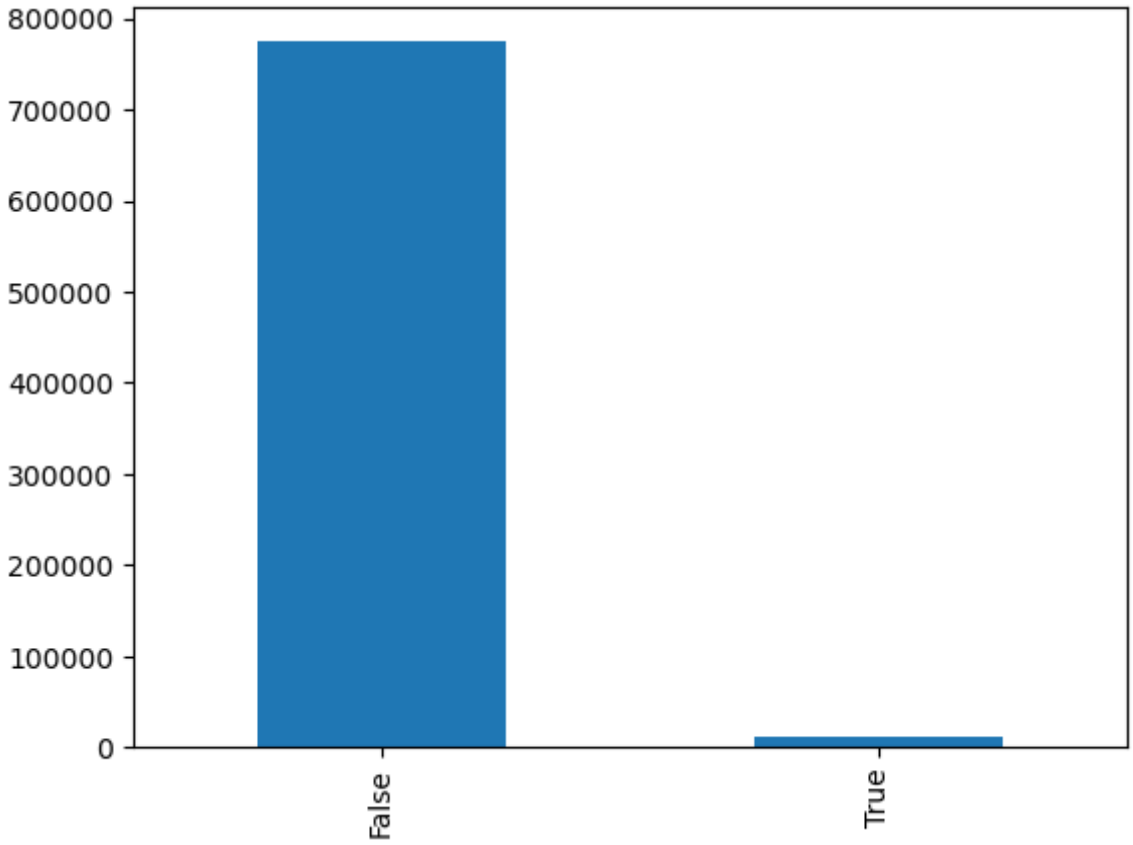
In [82]:

```
plt.figure(figsize=(5,5))
sep=[0.05,0.05]
t1['isFraud'].value_counts(normalize=False).plot(kind='pie',autopct='%1.1f%%',explode=sep);
plt.title('fraud',color='r',fontsize=14)
plt.ylabel('');
```



In [84]:

```
t1.isFraud.value_counts().plot.bar();
```



In [85]:

```
pd.crosstab(t1.isFraud,t1.merchantName)
```

Out[85]:

merchantName	1st BBQ	1st Deli	1st Pub	1st Restaurant	1st Sandwitch Bar #119707	1st Sandwitch Bar #396252	1st Sandwitch Bar #758805	1st Sandwitch Bar #772439	Sa
isFraud									
False	822	797	778	850	772	820	810	838	
True	13	5	22	12	9	8	7	8	

2 rows × 2490 columns

In [86]:

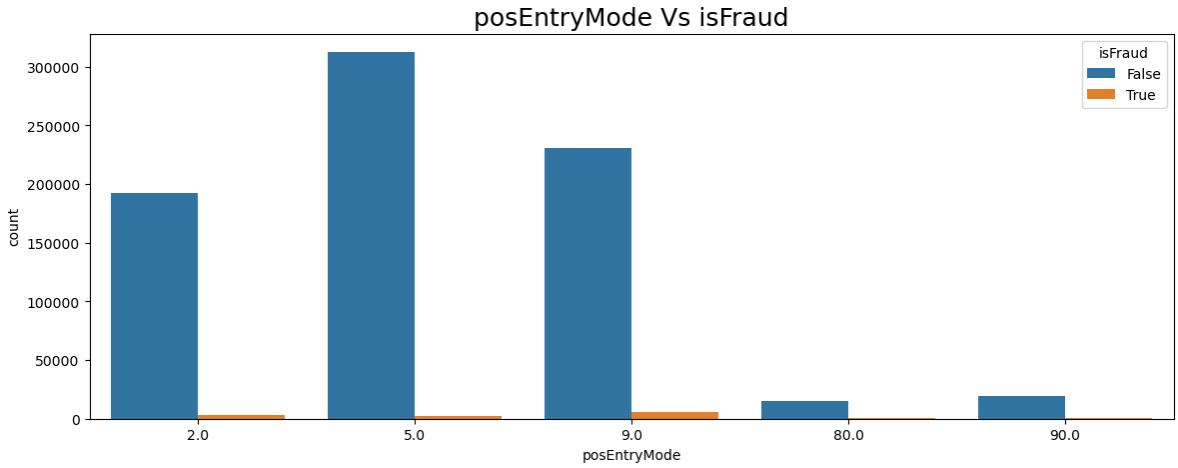
```
pd.crosstab(t1.isFraud,t1.posEntryMode)
```

Out[86]:

posEntryMode	2.0	5.0	9.0	80.0	90.0
isFraud					
False	192513	312579	230822	15043	19204
True	3421	2456	5659	240	372

In [87]:

```
plt.figure(figsize=(14,5))
sb.countplot(x='posEntryMode',hue='isFraud',data=t1)
plt.title('posEntryMode Vs isFraud',fontsize=18);
```



In [88]:

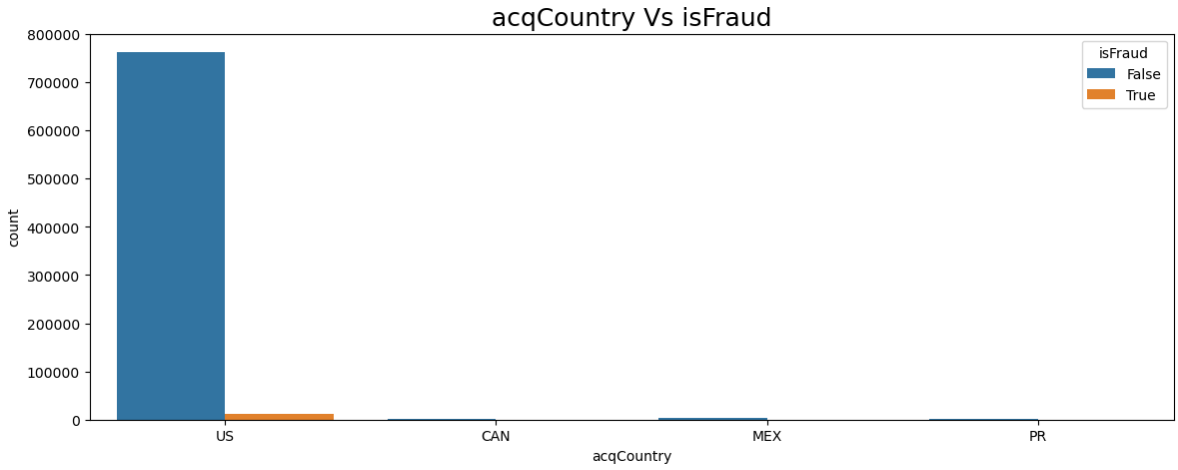
```
pd.crosstab(t1.isFraud,t1.acqCountry)
```

Out[88]:

acqCountry	CAN	MEX	PR	US
isFraud				
False	2369	3066	1511	762587
True	55	64	27	12122

In [89]:

```
plt.figure(figsize=(14,5))
sb.countplot(x='acqCountry',hue='isFraud',data=t1)
plt.title('acqCountry Vs isFraud',fontsize=18);
```



In [90]:

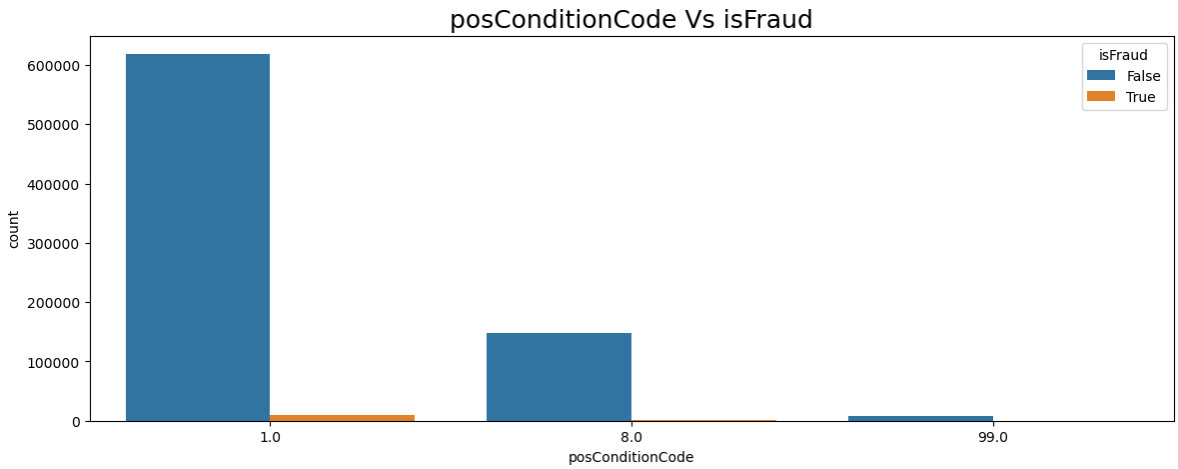
```
pd.crosstab(t1.isFraud,t1.posConditionCode)
```

Out[90]:

posConditionCode	1.0	8.0	99.0
isFraud			
False	618557	147698	7304
True	10230	1936	229

In [91]:

```
plt.figure(figsize=(14,5))
sb.countplot(x='posConditionCode',hue='isFraud',data=t1)
plt.title('posConditionCode Vs isFraud',fontsize=18);
```



In [92]:

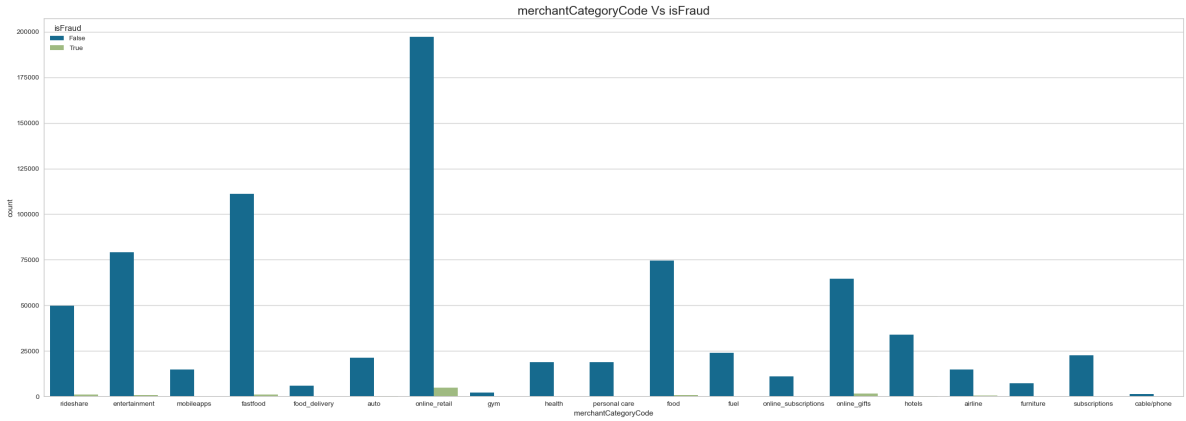
```
pd.crosstab(t1.isFraud,t1.merchantCategoryCode).T
```

Out[92]:

isFraud	False	True
merchantCategoryCode		
airline	14878	534
auto	21378	273
cable/phone	1382	0
entertainment	79137	961
fastfood	111064	1074
food	74476	1014
food_delivery	6000	0
fuel	23910	0
furniture	7329	103

In [192]:

```
plt.figure(figsize=(30,10))
sb.countplot(x='merchantCategoryCode',hue='isFraud',data=t1)
plt.title('merchantCategoryCode Vs isFraud',fontsize=18);
```



In [193]:

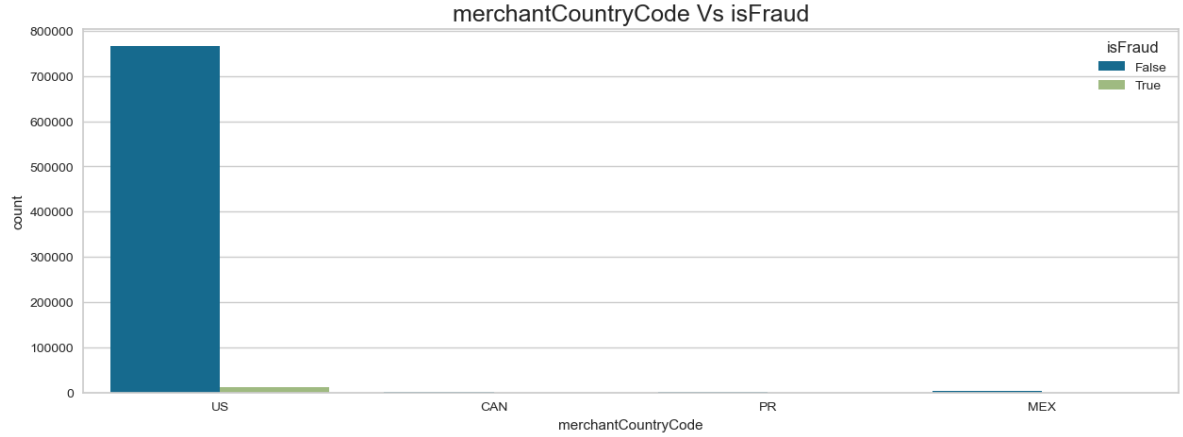
```
pd.crosstab(t1.isFraud,t1.merchantCountryCode)
```

Out[193]:

merchantCountryCode	CAN	MEX	PR	US
isFraud				
False	2370	3079	1532	766323
True	56	64	27	12188

In [194]:

```
plt.figure(figsize=(15,5))
sb.countplot(x='merchantCountryCode',hue='isFraud',data=t1)
plt.title('merchantCountryCode Vs isFraud',fontsize=18);
```



In [195]:

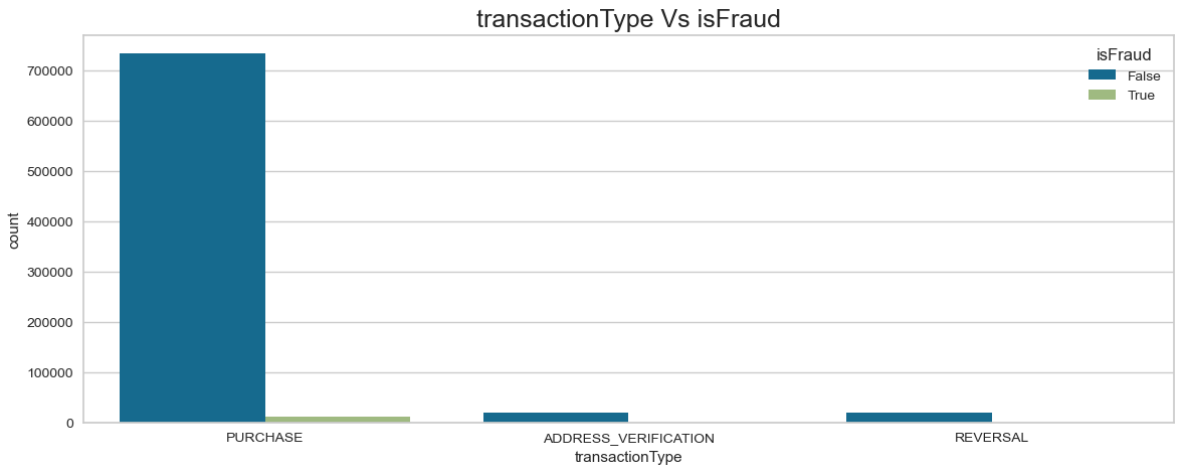
```
pd.crosstab(t1.isFraud,t1.transactionType)
```

Out[195]:

transactionType	ADDRESS_VERIFICATION	PURCHASE	REVERSAL
isFraud			
False	20053	733243	19966
True	116	11950	337

In [196]:

```
plt.figure(figsize=(14,5))
sb.countplot(x='transactionType',hue='isFraud',data=t1)
plt.title('transactionType Vs isFraud',fontsize=18);
```



In [197]:

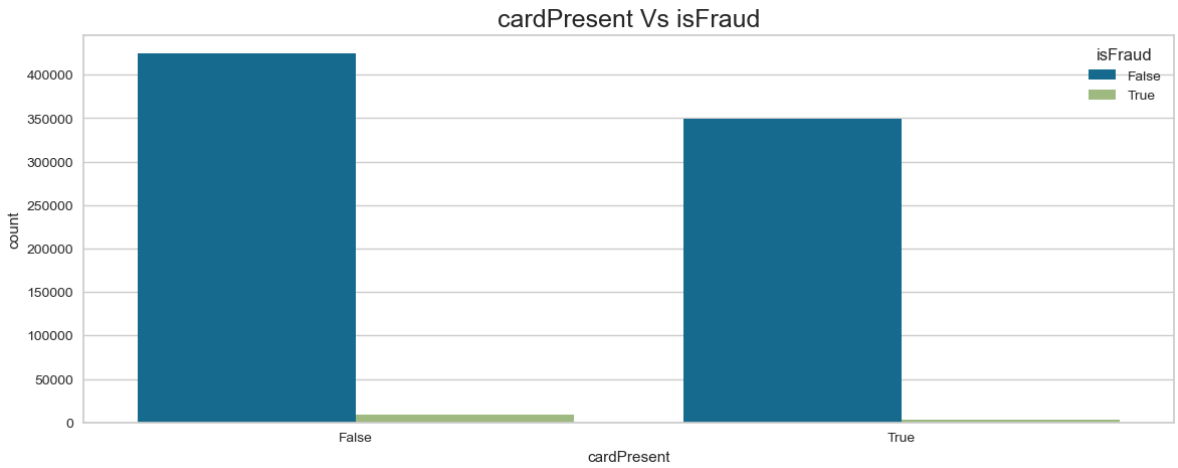
```
pd.crosstab(t1.isFraud,t1.cardPresent)
```

Out[197]:

cardPresent	False	True
isFraud		
False	424533	349413
True	8962	3455

In [198]:

```
plt.figure(figsize=(14,5))
sb.countplot(x='cardPresent',hue='isFraud',data=t1)
plt.title('cardPresent Vs isFraud',fontsize=18);
```



In [199]:

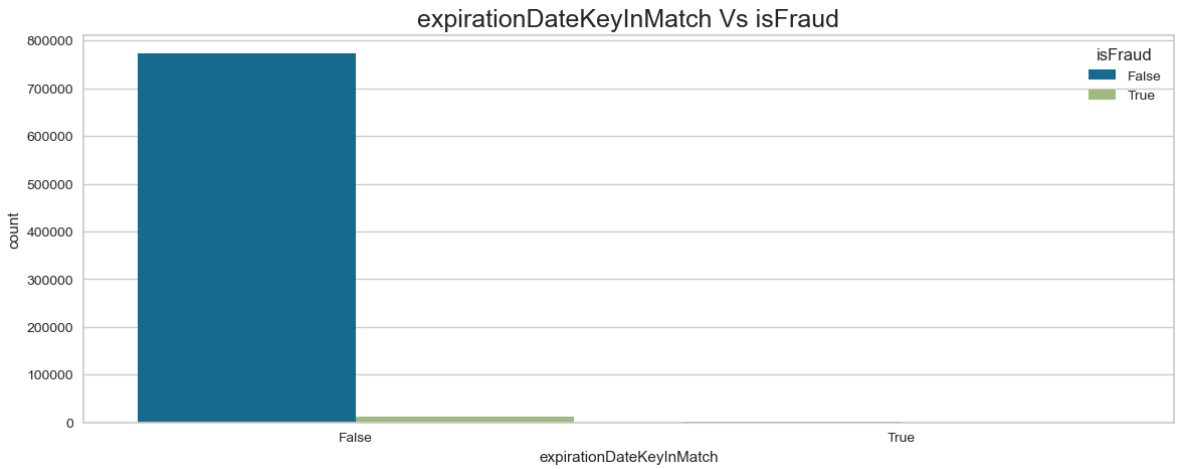
```
pd.crosstab(t1.isFraud,t1.expirationDateKeyInMatch)
```

Out[199]:

expirationDateKeyInMatch	False	True
isFraud		
False	772916	1030
True	12404	13

In [200]:

```
plt.figure(figsize=(14,5))
sb.countplot(x='expirationDateKeyInMatch',hue='isFraud',data=t1)
plt.title('expirationDateKeyInMatch Vs isFraud',fontsize=18);
```



In [201]:

```
t1['merchantName'].value_counts() ##content in the Tag (Total Tested)
```

Out[201]:

```
Uber                                25613
Lyft                                25523
oldnavy.com                         16992
staples.com                         16980
alibaba.com                         16959
                                    ...
Sprint Communications #561941           2
Runners #383214                         2
Curves #849125                          1
EZ Wireless #149871                     1
TMobile Wireless #602341                1
Name: merchantName, Length: 2490, dtype: int64
```

In [202]:

```
((t1.isna().sum()/len(t1))*100).sort_values(ascending=False).round(2)
```

Out[202]:

```
acqCountry                  0.58
posEntryMode                0.52
merchantCountryCode         0.09
transactionType             0.09
posConditionCode            0.05
creditLimit                 0.00
cardCVV                     0.00
expirationDateKeyInMatch    0.00
cardPresent                 0.00
currentBalance              0.00
cardLast4Digits             0.00
enteredCVV                  0.00
currentExpDate              0.00
dateOfLastAddressChange     0.00
accountOpenDate             0.00
availableMoney              0.00
merchantCategoryCode        0.00
merchantName                0.00
transactionAmount           0.00
transactionDateTime         0.00
isFraud                     0.00
dtype: float64
```

In [203]:

```
def missing_values(t1):
    mv=t1.isna().sum()
    mvp=((t1.isna().sum()/len(t1))*100).sort_values(ascending=False).round(2)
    mvp=pd.concat([mv,mvp],axis=1)
    mvp=mvp.rename(columns={0:'Missing Values',1:'% Of Missing Values'})
    mc=mvp[mvp.iloc[:,1]!=0].sort_values('% Of Missing Values',ascending=False)
    return mc
mis=missing_values(t1)
mis
```

Out[203]:

	Missing Values	% Of Missing Values
acqCountry	4562	0.58
posEntryMode	4054	0.52
merchantCountryCode	724	0.09
transactionType	698	0.09
posConditionCode	409	0.05

In [204]:

```
t2=t1.loc[:,t1.isna().mean()<0.4]
t2.columns
```

Out[204]:

```
Index(['creditLimit', 'availableMoney', 'transactionDateTime',
       'transactionAmount', 'merchantName', 'acqCountry',
       'merchantCountryCode', 'posEntryMode', 'posConditionCode',
       'merchantCategoryCode', 'currentExpDate', 'accountOpenDate',
       'dateOfLastAddressChange', 'cardCVV', 'enteredCVV', 'cardLast4Digit
s',
       'transactionType', 'currentBalance', 'cardPresent',
       'expirationDateKeyInMatch', 'isFraud'],
      dtype='object')
```

In [205]:

```
obj=['acqCountry','merchantCountryCode','transactionType']
t2[obj]=t2[obj].fillna(t2[obj].mode().iloc[0])
t2.isna().sum()
```

Out[205]:

```
creditLimit                  0
availableMoney               0
transactionDateTime          0
transactionAmount            0
merchantName                 0
acqCountry                   0
merchantCountryCode          0
posEntryMode              4054
posConditionCode           409
merchantCategoryCode         0
currentExpDate               0
accountOpenDate              0
dateOfLastAddressChange      0
cardCVV                      0
enteredCVV                   0
cardLast4Digits              0
transactionType              0
currentBalance               0
cardPresent                  0
expirationDateKeyInMatch     0
isFraud                      0
dtype: int64
```

In [206]:

```
nm=['posEntryMode','posConditionCode']
t2[nm]=t1[nm].fillna(t1[nm].median())
t2.isna().sum()
```

Out[206]:

```
creditLimit                0
availableMoney             0
transactionDateTime        0
transactionAmount          0
merchantName               0
acqCountry                 0
merchantCountryCode        0
posEntryMode               0
posConditionCode           0
merchantCategoryCode       0
currentExpDate             0
accountOpenDate            0
dateOfLastAddressChange    0
cardCVV                    0
enteredCVV                 0
cardLast4Digits            0
transactionType            0
currentBalance             0
cardPresent                0
expirationDateKeyInMatch   0
isFraud                    0
dtype: int64
```

In [207]:

```
dup=t2.duplicated().sum()
dup
```

Out[207]:

```
0
```

In [281]:

```
t.dtypes
```

Out[281]:

```
accountNumber                int64
customerId                   int64
creditLimit                  int64
availableMoney             float64
transactionDateTime         object
transactionAmount          float64
merchantName                object
acqCountry                  object
merchantCountryCode         object
posEntryMode               float64
posConditionCode           float64
merchantCategoryCode        object
currentExpDate              object
accountOpenDate             object
dateOfLastAddressChange     object
cardCVV                      int64
enteredCVV                   int64
cardLast4Digits              int64
transactionType             object
echoBuffer                 float64
currentBalance             float64
merchantCity               float64
merchantState              float64
merchantZip                float64
cardPresent                   bool
posOnPremises              float64
recurringAuthInd           float64
expirationDateKeyInMatch      bool
isFraud                       bool
dtype: object
```

Chi-Square

In [279]:

```
import scipy
from scipy.stats import chi2_contingency
```

In [287]:

```
c1=pd.crosstab(t2.isFraud,t2.merchantName)
c1
```

Out[287]:

merchantName	0	1	2	3	4	5	6	7	8	9	...	2480	2481	2482	248
isFraud															
0	822	797	778	850	772	820	810	838	781	840	...	495	2606	16591	47
1	13	5	22	12	9	8	7	8	11	12	...	2	0	401	1

2 rows × 2490 columns

In [289]:

```
chi_2,p_value,gol,frequency=chi2_contingency(c1,correction=False)
alpha=0.05
print('level of significance=%.2f,p_value=%.2f,chi_2=%.2f'%(alpha,p_value,chi_2))
```

```
level of significance=0.05,p_value=0.00,chi_2=8475.52
```

since the p_value=0.0000 which is less than 0.05 we reject the null hypothesis and conclude that there is statistical significance association between isFraud and merchantName

In [290]:

```
c2=pd.crosstab(t2.isFraud,t2.acqCountry)
c2
```

Out[290]:

acqCountry	0	1	2	3
isFraud				
0	2369	3066	1511	767000
1	55	64	27	12271

In [291]:

```
chi_2,p_value,gol,frequency=chi2_contingency(c2,correction=False)
alpha=0.05
print('level of significance=%.2f,p_value=%.2f,chi_2=%.2f'%(alpha,p_value,chi_2))
```

```
level of significance=0.05,p_value=0.01,chi_2=12.20
```

In [292]:

```
c3=pd.crosstab(t2.isFraud,t2.expirationDateKeyInMatch)
c3
```

Out[292]:

expirationDateKeyInMatch	0	1
isFraud		
0	772916	1030
1	12404	13

In [293]:

```
chi_2,p_value,gol,frequency=chi2_contingency(c3,correction=False)
alpha=0.05
print('level of significance=%.2f,p_value=%.2f,chi_2=%.2f'%(alpha,p_value,chi_2))
```

```
level of significance=0.05,p_value=0.39,chi_2=0.74
```

In [294]:

```
c4=pd.crosstab(t2.isFraud,t2.merchantCountryCode)
c4
```

Out[294]:

merchantCountryCode	0	1	2	3
isFraud				
0	2370	3079	1532	766965
1	56	64	27	12270

In [295]:

```
chi_2,p_value,gol,frequency=chi2_contingency(c4,correction=False)
alpha=0.05
print('level of significance=%.2f,p_value=%.2f,chi_2=%.2f'%(alpha,p_value,chi_2))
```

```
level of significance=0.05,p_value=0.00,chi_2=12.86
```

In [296]:

```
c5=pd.crosstab(t2.isFraud,t2.merchantCategoryCode)
c5
```

Out[296]:

merchantCategoryCode	0	1	2	3	4	5	6	7	8	9
isFraud										
0	14878	21378	1382	79137	111064	74476	6000	23910	7329	2209
1	534	273	0	961	1074	1014	0	0	103	0

In [297]:

```
chi_2,p_value,gol,frequency=chi2_contingency(c5,correction=False)
alpha=0.05
print('level of significance=%.2f,p_value=%.2f,chi_2=%.2f'%(alpha,p_value,chi_2))
```

```
level of significance=0.05,p_value=0.00,chi_2=3772.07
```

In [300]:

```
c6=pd.crosstab(t2.isFraud,t2.transactionType)
c6
```

Out[300]:

transactionType	0	1	2
isFraud			
0	20053	733927	19966
1	116	11964	337

In [301]:

```
chi_2,p_value,gol,frequency=chi2_contingency(c6,correction=False)
alpha=0.05
print('level of significance=%.2f,p_value=%.2f,chi_2=%.2f'%(alpha,p_value,chi_2))
```

```
level of significance=0.05,p_value=0.00,chi_2=134.63
```

In [303]:

```
c7=pd.crosstab(t2.isFraud,t2.cardPresent)
c7
```

Out[303]:

cardPresent	0	1
isFraud		
0	424533	349413
1	8962	3455

In [304]:

```
chi_2,p_value,gol,frequency=chi2_contingency(c7,correction=False)
alpha=0.05
print('level of significance=%.2f,p_value=%.2f,chi_2=%.2f'%(alpha,p_value,chi_2))
```

```
level of significance=0.05,p_value=0.00,chi_2=1482.38
```

In [306]:

```
c8=pd.crosstab(t2.isFraud,t2.isFraud)
c8
```

Out[306]:

isFraud	0	1
isFraud		
0	773946	0
1	0	12417

In [307]:

```
chi_2,p_value,gol,frequency=chi2_contingency(c8,correction=False)
alpha=0.05
print('level of significance=%.2f,p_value=%.2f,chi_2=%.2f'%(alpha,p_value,chi_2))
```

```
level of significance=0.05,p_value=0.00,chi_2=786363.00
```

Two Sample T-Test

In [308]:

```
from scipy.stats import ttest_ind
```

In [309]:

```
f0=t2.creditLimit[t2.isFraud==0]
f1=t2.creditLimit[t2.isFraud==1]
```

In [310]:

```
tscore,pvalue=ttest_ind(f0,f1,equal_var=False)
alpha=0.05
print('t_score: {:.4f})'.format(tscore))
print('pvalue: {:.4f})'.format(pvalue))
print('level of significance: {:.2f})'.format(alpha))
```

```
t_score: -2.6861)
pvalue: 0.0072)
level of significance: 0.05)
```

since the P-value=0.0072 which is less than 0.05 we reject the null hypothesis and conclude that there is statistical significant difference between the average creditLimit of customers who are fraud and those who are not fraud.

In [311]:

```
f0=t2.availableMoney[t2.isFraud==0]
f1=t2.availableMoney[t2.isFraud==1]
```

In [312]:

```
tscore,pvalue=ttest_ind(f0,f1,equal_var=False)
alpha=0.05
print('t_score: {:.4f})'.format(tscore))
print('pvalue: {:.4f})'.format(pvalue))
print('level of significance: {:.2f})'.format(alpha))
```

```
t_score: 1.3912)
pvalue: 0.1642)
level of significance: 0.05)
```

In [313]:

```
f0=t2.transactionAmount[t2.isFraud==0]
f1=t2.transactionAmount[t2.isFraud==1]
```

In [314]:

```
tscore,pvalue=ttest_ind(f0,f1,equal_var=False)
alpha=0.05
print('t_score: {:.4f})'.format(tscore))
print('pvalue: {:.4f})'.format(pvalue))
print('level of significance: {:.2f})'.format(alpha))
```

```
t_score: -52.4492)
pvalue: 0.0000)
level of significance: 0.05)
```

In [315]:

```
f0=t2.posEntryMode[t2.isFraud==0]
f1=t2.posEntryMode[t2.isFraud==1]
```

In [316]:

```
tscore,pvalue=ttest_ind(f0,f1,equal_var=False)
alpha=0.05
print('t_score: {:.4f})'.format(tscore))
print('pvalue: {:.4f})'.format(pvalue))
print('level of significance: {:.2f})'.format(alpha))
```

```
t_score: -6.1511)
pvalue: 0.0000)
level of significance: 0.05)
```

In [317]:

```
f0=t2.posConditionCode[t2.isFraud==0]
f1=t2.posConditionCode[t2.isFraud==1]
```

In [318]:

```
tscore,pvalue=ttest_ind(f0,f1,equal_var=False)
alpha=0.05
print('t_score: {:.4f})'.format(tscore))
print('pvalue: {:.4f})'.format(pvalue))
print('level of significance: {:.2f})'.format(alpha))
```

```
t_score: -5.3306)
pvalue: 0.0000)
level of significance: 0.05)
```

In [319]:

```
f0=t2.cardCVV[t2.isFraud==0]
f1=t2.cardCVV[t2.isFraud==1]
```

In [320]:

```
tscore,pvalue=ttest_ind(f0,f1,equal_var=False)
alpha=0.05
print('t_score: {:.4f})'.format(tscore))
print('pvalue: {:.4f})'.format(pvalue))
print('level of significance: {:.2f})'.format(alpha))
```

```
t_score: -7.6885)
pvalue: 0.0000)
level of significance: 0.05)
```

In [321]:

```
f0=t2.enteredCVV[t2.isFraud==0]
f1=t2.enteredCVV[t2.isFraud==1]
```

In [322]:

```
tscore,pvalue=ttest_ind(f0,f1,equal_var=False)
alpha=0.05
print('t_score: {:.4f})'.format(tscore))
print('pvalue: {:.4f})'.format(pvalue))
print('level of significance: {:.2f})'.format(alpha))
```

```
t_score: -7.6991)
pvalue: 0.0000)
level of significance: 0.05)
```

In [323]:

```
f0=t2.cardLast4Digits[t2.isFraud==0]
f1=t2.cardLast4Digits[t2.isFraud==1]
```

In [324]:

```
tscore,pvalue=ttest_ind(f0,f1,equal_var=False)
alpha=0.05
print('t_score: {:.4f})'.format(tscore))
print('pvalue: {:.4f})'.format(pvalue))
print('level of significance: {:.2f})'.format(alpha))
```

```
t_score: -0.7790)
pvalue: 0.4360)
level of significance: 0.05)
```

In [326]:

```
f0=t2.currentBalance[t2.isFraud==0]
f1=t2.currentBalance[t2.isFraud==1]
```

In [327]:

```
tscore,pvalue=ttest_ind(f0,f1,equal_var=False)
alpha=0.05
print('t_score: {:.4f})'.format(tscore))
print('pvalue: {:.4f})'.format(pvalue))
print('level of significance: {:.2f})'.format(alpha))
```

```
t_score: -6.2530)
pvalue: 0.0000)
level of significance: 0.05)
```

In [ ]:

```
```

In [ ]:

```
```

In [208]:

```
#categorical encoding
from sklearn.preprocessing import LabelEncoder
```

In [209]:

```
le=LabelEncoder()
```

In [210]:

```
obj=t2.select_dtypes(include=['object','bool'])
obj.columns
```

Out[210]:

```
Index(['merchantName', 'acqCountry', 'merchantCountryCode',
       'merchantCategoryCode', 'transactionType', 'cardPresent',
       'expirationDateKeyInMatch', 'isFraud'],
      dtype='object')
```

In [211]:

```
ob=['merchantName', 'acqCountry', 'merchantCountryCode',
       'merchantCategoryCode', 'transactionType', 'cardPresent',
       'expirationDateKeyInMatch', 'isFraud']
```

In [212]:

```
for m in ob:
    t2[m]=le.fit_transform(t2[m])
t2.dtypes
```

Out[212]:

```
creditLimit                      int64
availableMoney                 float64
transactionDateTime     datetime64[ns]
transactionAmount              float64
merchantName                     int32
acqCountry                       int32
merchantCountryCode              int32
posEntryMode                   float64
posConditionCode               float64
merchantCategoryCode             int32
currentExpDate          datetime64[ns]
accountOpenDate         datetime64[ns]
dateOfLastAddressChange datetime64[ns]
cardCVV                          int64
enteredCVV                       int64
cardLast4Digits                  int64
transactionType                  int32
currentBalance                 float64
cardPresent                      int64
expirationDateKeyInMatch         int64
isFraud                          int64
dtype: object
```

In [213]:

```
t3=t2.drop(['transactionDateTime','accountOpenDate','dateOfLastAddressChange','currentExpDa
t3.dtypes
```

Out[213]:

```
creditLimit                 int64
availableMoney            float64
transactionAmount         float64
merchantName                int32
acqCountry                  int32
merchantCountryCode         int32
posEntryMode              float64
posConditionCode          float64
merchantCategoryCode        int32
cardCVV                     int64
enteredCVV                  int64
cardLast4Digits             int64
transactionType             int32
currentBalance            float64
cardPresent                 int64
expirationDateKeyInMatch    int64
isFraud                     int64
dtype: object
```

In [214]:

```
t3.dtypes
```

Out[214]:

```
creditLimit                  int64
availableMoney             float64
transactionAmount          float64
merchantName                 int32
acqCountry                   int32
merchantCountryCode          int32
posEntryMode               float64
posConditionCode           float64
merchantCategoryCode         int32
cardCVV                      int64
enteredCVV                   int64
cardLast4Digits              int64
transactionType              int32
currentBalance             float64
cardPresent                  int64
expirationDateKeyInMatch     int64
isFraud                      int64
dtype: object
```

In [ ]:

```

```

In [215]:

```
#seperating the dependent and indipendent variables
x=t3.drop('isFraud',1)
y=t3.isFraud
```

In [216]:

```
#data balancing
import imblearn
from imblearn.over_sampling import SMOTE
sm=SMOTE()
xs,ys=sm.fit_resample(x,y)
```

In [217]:

```
#splitting the dataset
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(xs,ys,test_size=0.20,random_state=42)
```

In [218]:

```
#importing classification evaluation metrice
from sklearn.metrics import accuracy_score,recall_score,precision_score,f1_score,confusion_
```

Logistic Regression

In [219]:

```
from sklearn.linear_model import LogisticRegression
```

In [220]:

```
lr=LogisticRegression()
```

In [221]:

```
lr.fit(xtrain,ytrain)
```

Out[221]:

```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [222]:

```
lr_pred=lr.predict(xtest)
```

In [223]:

```
accuracy_score(ytest,lr_pred)
```

Out[223]:

```
0.6230558274301552
```

In [224]:

```
recall_score(ytest,lr_pred)
```

Out[224]:

```
0.6107613309422132
```

In [225]:

```
precision_score(ytest,lr_pred)
```

Out[225]:

```
0.6268720950249612
```

In [226]:

```
f1_score(ytest,lr_pred)
```

Out[226]:

```
0.6187118528877446
```

In [227]:

```
confusion_matrix(ytest,lr_pred)
```

Out[227]:

```
array([[98206, 56355],
       [60339, 94679]], dtype=int64)
```

In [228]:

```
roc_auc_score(ytest,lr_pred)
```

Out[228]:

```
0.623074003376529
```

Random Forest

In [229]:

```
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier,AdaBoostClas
```

RandomForestClassifier

In [230]:

```
rfc=RandomForestClassifier(random_state=2)
```

In [231]:

```
rfc.fit(xtrain,ytrain)
```

Out[231]:

```
RandomForestClassifier(random_state=2)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [232]:

```
rfc_pred=rfc.predict(xtest)
```

In [233]:

```
accuracy_score(ytest,rfc_pred)
```

Out[233]:

```
0.9901317595831759
```

In [234]:

```
recall_score(ytest,rfc_pred)
```

Out[234]:

```
0.9870144112296636
```

In [235]:

```
precision_score(ytest,rfc_pred)
```

Out[235]:

```
0.9932358306231215
```

In [236]:

```
f1_score(ytest,rfc_pred)
```

Out[236]:

```
0.9901153479041624
```

In [237]:

```
confusion_matrix(ytest,rfc_pred)
```

Out[237]:

```
array([[153519,   1042],
       [  2013, 153005]], dtype=int64)
```

In [238]:

```
roc_auc_score(ytest,rfc_pred)
```

Out[238]:

```
0.9901363682108295
```

Decision Tree

In [239]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [240]:

```
dt=DecisionTreeClassifier()
```

In [241]:

```
dt.fit(xtrain,ytrain)
```

Out[241]:

```
DecisionTreeClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [242]:

```
dt_pred=dt.predict(xtest)
```

In [243]:

```
accuracy_score(ytest,dt_pred)
```

Out[243]:

```
0.9752147270971222
```

In [244]:

```
recall_score(ytest,dt_pred)
```

Out[244]:

```
0.980937697557703
```

In [245]:

```
precision_score(ytest,dt_pred)
```

Out[245]:

```
0.9699070678207181
```

In [246]:

```
f1_score(ytest,dt_pred)
```

Out[246]:

```
0.9753911975343089
```

In [247]:

```
confusion_matrix(ytest,dt_pred)
```

Out[247]:

```
array([[149843,   4718],
       [  2955, 152063]], dtype=int64)
```

In [248]:

```
roc_auc_score(ytest,dt_pred)
```

Out[248]:

```
0.9752062663680235
```

GradientBoostingClassifier

In [249]:

```
gbc=GradientBoostingClassifier()
```

In [250]:

```
gbc.fit(xtrain,ytrain)
```

Out[250]:

```
GradientBoostingClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [251]:

```
gbc_pred=gbc.predict(xtest)
```

In [252]:

```
accuracy_score(ytest,gbc_pred)
```

Out[252]:

```
0.9014694149150944
```

In [253]:

```
recall_score(ytest,gbc_pred)
```

Out[253]:

```
0.8618934575339638
```

In [254]:

```
precision_score(ytest,gbc_pred)
```

Out[254]:

```
0.9362732388246918
```

In [255]:

```
f1_score(ytest,gbc_pred)
```

Out[255]:

```
0.8975450169789837
```

In [256]:

```
confusion_matrix(ytest,gbc_pred)
```

Out[256]:

```
array([[145467,   9094],
       [ 21409, 133609]], dtype=int64)
```

In [257]:

```
roc_auc_score(ytest,gbc_pred)
```

Out[257]:

```
0.9015279232468313
```

AdaBoostClassifier

In [258]:

```
abc=AdaBoostClassifier()
```

In [259]:

```
abc.fit(xtrain,ytrain)
```

Out[259]:

```
AdaBoostClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [260]:

```
abc_pred=abc.predict(xtest)
```

In [261]:

```
accuracy_score(ytest,abc_pred)
```

Out[261]:

```
0.8941724083351907
```

In [262]:

```
recall_score(ytest,abc_pred)
```

Out[262]:

```
0.8705376149866467
```

In [263]:

```
precision_score(ytest,abc_pred)
```

Out[263]:

```
0.9140285284675093
```

In [264]:

```
f1_score(ytest,abc_pred)
```

Out[264]:

```
0.8917531223154695
```

In [265]:

```
confusion_matrix(ytest,abc_pred)
```

Out[265]:

```
array([[141868,  12693],
       [ 20069, 134949]], dtype=int64)
```

In [266]:

```
roc_auc_score(ytest,abc_pred)
```

Out[266]:

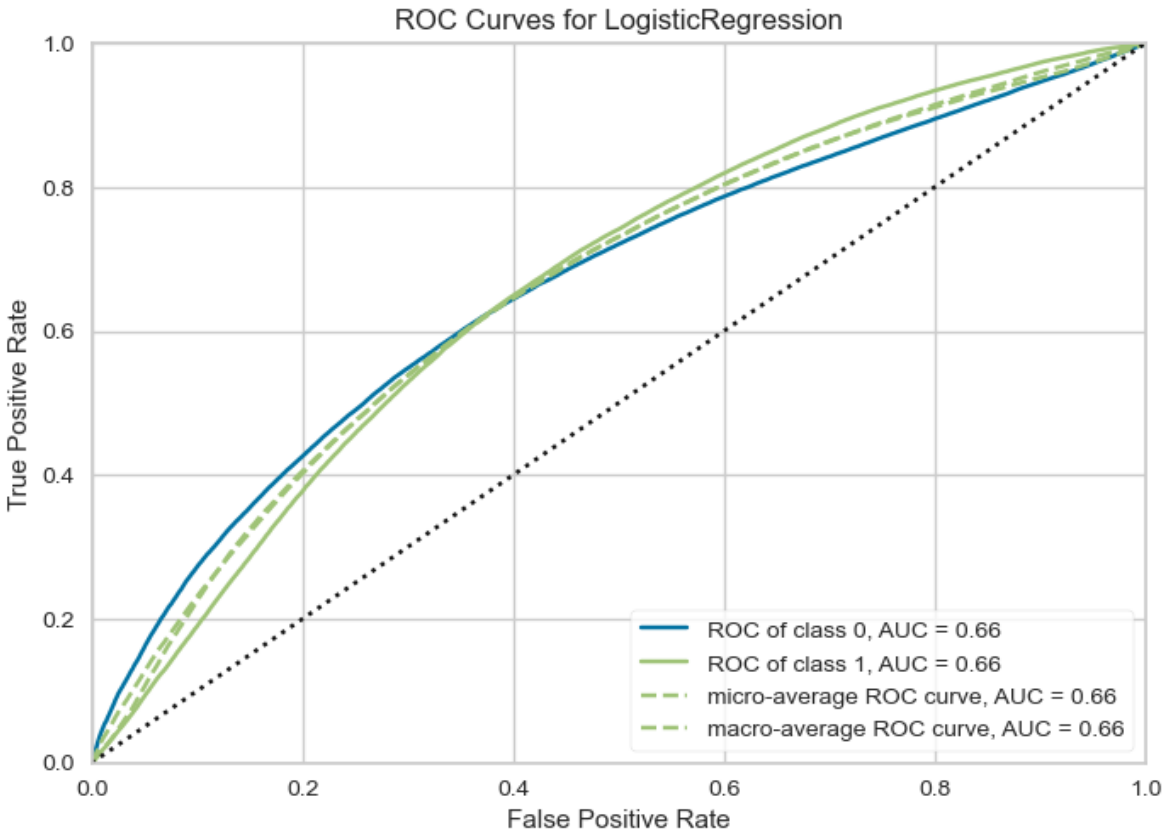
```
0.8942073495576216
```

In [267]:

```
from yellowbrick.exceptions import YellowbrickValueError
from yellowbrick.classifier import ROCAUC
```

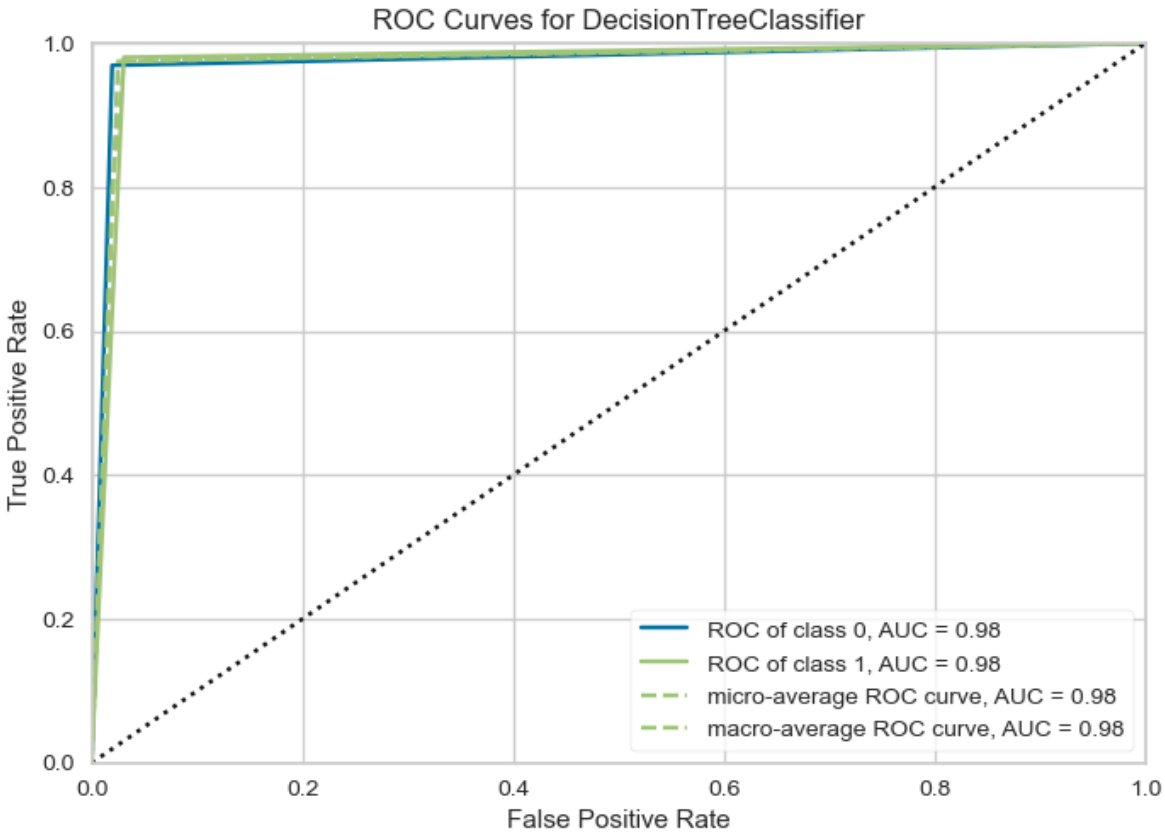
In [268]:

```
lr=LogisticRegression()
roc_lr=ROCAUC(lr)
roc_lr.fit(xtrain,ytrain)
roc_lr.score(xtest,ytest)
roc_lr.show();
```



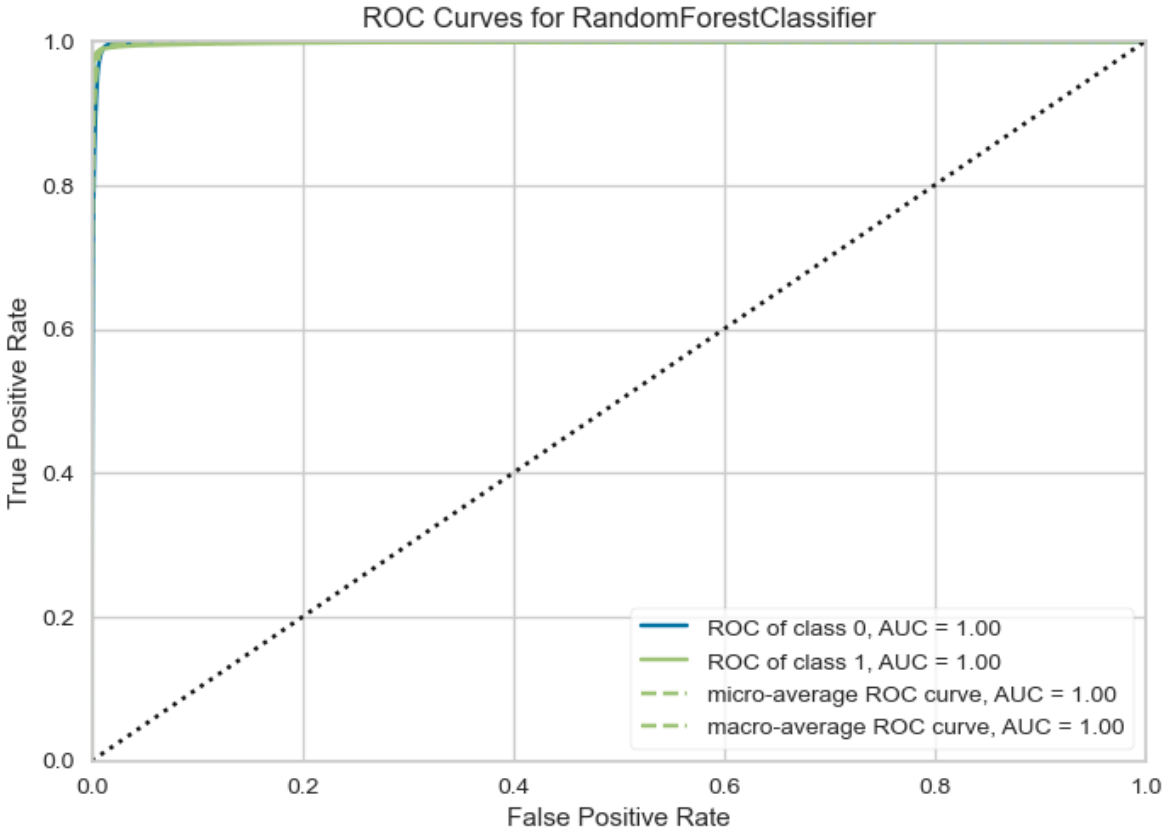
In [269]:

```
dt=DecisionTreeClassifier()
roc_dt=ROCAUC(dt)
roc_dt.fit(xtrain,ytrain)
roc_dt.score(xtest,ytest)
roc_dt.show();
```



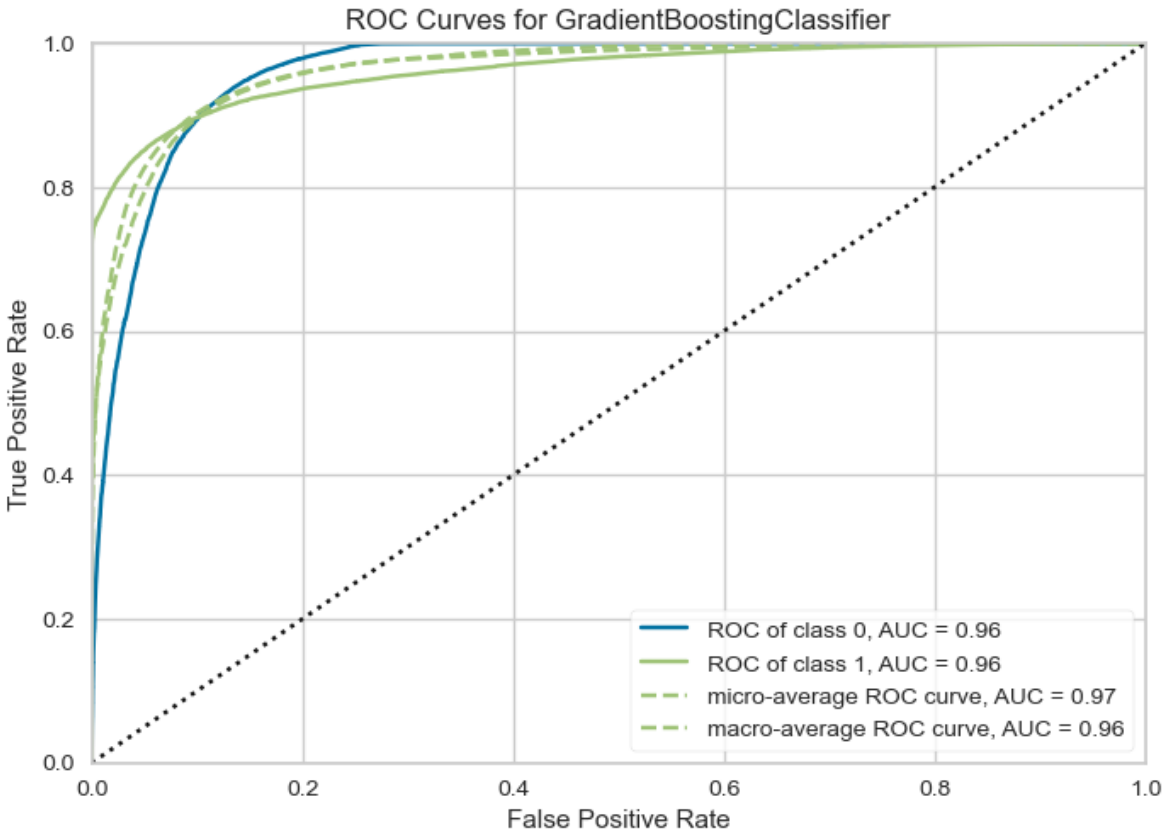
In [270]:

```
rfc=RandomForestClassifier(random_state=2)
roc_rfc=ROCAUC(rfc)
roc_rfc.fit(xtrain,ytrain)
roc_rfc.score(xtest,ytest)
roc_rfc.show();
```



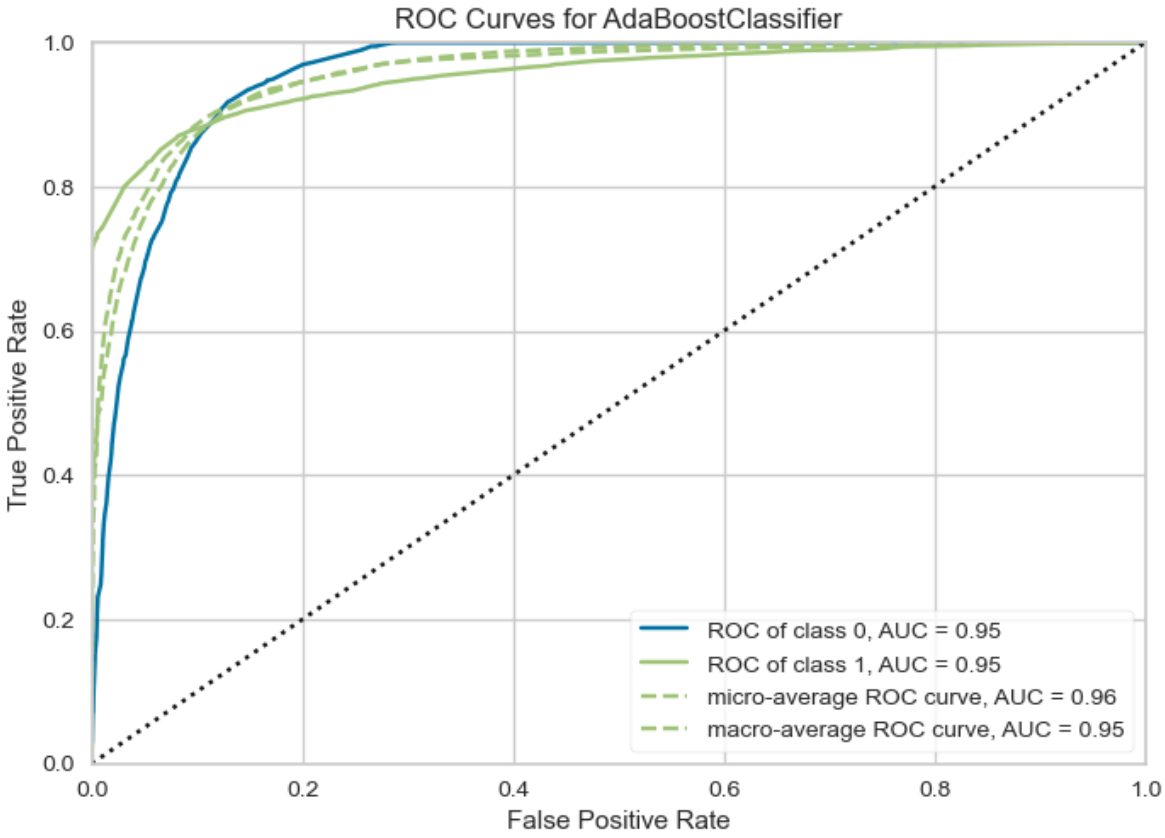
In [271]:

```
gbc=GradientBoostingClassifier()
roc_gbc=ROCAUC(gbc)
roc_gbc.fit(xtrain,ytrain)
roc_gbc.score(xtest,ytest)
roc_gbc.show();
```



In [272]:

```
abc=AdaBoostClassifier()
roc_abc=ROCAUC(abc)
roc_abc.fit(xtrain,ytrain)
roc_abc.score(xtest,ytest)
roc_abc.show();
```

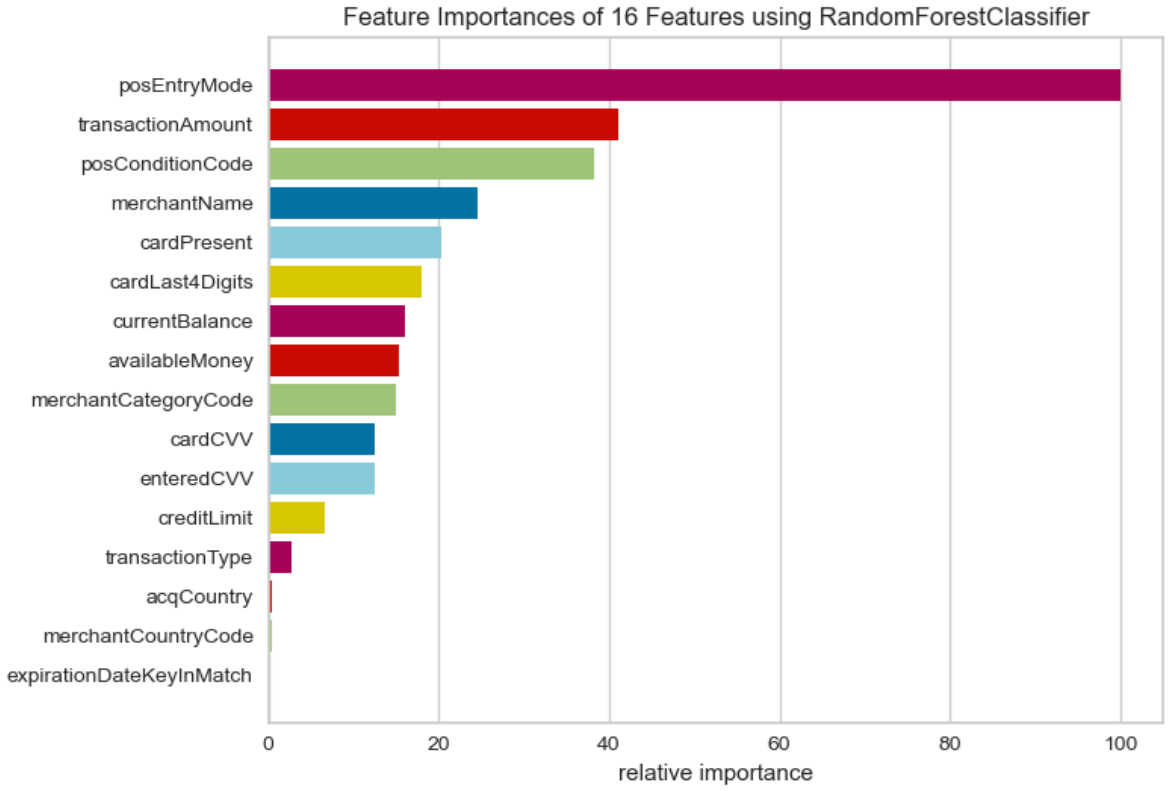


In [273]:

```
##feature importance
from yellowbrick.model_selection import cv_scores,FeatureImportances
```

In [275]:

```
rfc=RandomForestClassifier(random_state=2)
impf=FeatureImportances(rfc)
impf.fit(xtrain,ytrain)
impf.show();
```



In [ ]: