In [3]:
```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [4]:
```
# Import data
df= pd.read_csv("F:\\datasets folder\\godigt_cc_data.csv")
```

In [3]:
```
# See the heads of dataset
df.head(5)
```

Out[3]:

	userid	card_no	card_bin_no	Issuer	card_type	card_source_date	high_networth	active_30	active
0	1	4384 39XX XXXX XXXX	438439	Visa	edge	29-09-2019	B	0	
1	2	4377 48XX XXXX XXXX	437748	Visa	prosperity	30-10-2002	A	1	
2	3	4377 48XX XXXX XXXX	437748	Visa	rewards	05-10-2013	C	0	
3	4	4258 06XX XXXX XXXX	425806	Visa	indianoil	01-06-1999	E	0	
4	5	4377 48XX XXXX XXXX	437748	Visa	edge	13-06-2006	B	1	

5 rows × 29 columns

In [4]:
```
# see th tail or bottom of dataset
df.tail(5)
```

Out[4]:

	userid	card_no	card_bin_no	Issuer	card_type	card_source_date	high_networth	active_30	ac
8443	8444	4262 41XX XXXX XXXX	426241	Visa	chartered	10-01-2010	A	1	
8444	8445	37691 6XXXX	376916	Amex	centurion	19-07-2006	A	0	

	userid	card_no	card_bin_no	Issuer	card_type	card_source_date	high_networth	active_30	ac
		XXXXX							
8445	8446	4375 51XX XXXX XXXX	437551	Visa	rewards	15-02-2006	D	0	
8446	8447	4477 47XX XXXX XXXX	447747	Visa	indianoil	06-11-2003	C	0	
8447	8448	4262 41XX XXXX XXXX	426241	Visa	rewards	13-01-2013	D	0	

5 rows × 29 columns

In [5]:
```
#see the shape of data
df.shape
```

Out[5]: (8448, 29)

In [6]:
```
# no of columns we have
df.columns
```

Out[6]:
```
Index(['userid', 'card_no', 'card_bin_no', 'Issuer', 'card_type',
       'card_source_date', 'high_networth', 'active_30', 'active_60',
       'active_90', 'cc_active30', 'cc_active60', 'cc_active90',
       'hotlist_flag', 'widget_products', 'engagement_products',
       ' annual_income_at_source ', 'other_bank_cc_holding', 'bank_vintage',
       'T+1_month_activity', 'T+2_month_activity', 'T+3_month_activity',
       'T+6_month_activity', 'T+12_month_activity', 'Transactor_revolver',
       ' avg_spends_l3m ', 'Occupation_at_source', 'cc_limit', 'Unnamed: 28'],
      dtype='object')
```

In [7]:
```
# need to change the column name here with 'new_column' name
df.rename(columns = {'Unnamed: 28':'new_column'}, inplace = True)
df.head(2)
```

Out[7]:

	userid	card_no	card_bin_no	Issuer	card_type	card_source_date	high_networth	active_30	active
0	1	4384 39XX XXXX XXXX	438439	Visa	edge	29-09-2019	B	0	
1	2	4377 48XX XXXX XXXX	437748	Visa	prosperity	30-10-2002	A	1	

2 rows × 29 columns

In [8]:
```
#need to change the elements of the column (occupation_at_source)
df.Occupation_at_source = df.Occupation_at_source.str.replace('0',
'Others')
df
```

Out[8]:

	userid	card_no	card_bin_no	Issuer	card_type	card_source_date	high_networth	active_30	ac
0	1	438439XXXXXX	438439	Visa	edge	29-09-2019	B	0	
1	2	437748XXXXXX	437748	Visa	prosperity	30-10-2002	A	1	
2	3	437748XXXXXX	437748	Visa	rewards	05-10-2013	C	0	
3	4	425806XXXXXX	425806	Visa	indianoil	01-06-1999	E	0	
4	5	437748XXXXXX	437748	Visa	edge	13-06-2006	B	1	
...	...	...	...	...	...	...	...	...	
8443	8444	426241XXXXXX	426241	Visa	chartered	10-01-2010	A	1	
8444	8445	376916XXXXXX	376916	Amex	centurion	19-07-2006	A	0	
8445	8446	437551XXXXXX	437551	Visa	rewards	15-02-2006	D	0	
8446	8447	447747XXXXXX	447747	Visa	indianoil	06-11-2003	C	0	
8447	8448	426241XXXXXX	426241	Visa	rewards	13-01-2013	D	0	

8448 rows × 29 columns

In [9]:
```
# unique values
df.nunique()
```

Out[9]:
```
userid                      8448
card_no                       11
card_bin_no                   11
Issuer                         3
card_type                     15
card_source_date            5186
high_networth                  5
active_30                      2
active_60                      2
active_90                      2
cc_active30                    2
cc_active60                    2
cc_active90                    2
hotlist_flag                   2
widget_products                8
engagement_products            9
 annual_income_at_source    8435
other_bank_cc_holding          2
bank_vintage                  55
T+1_month_activity             2
T+2_month_activity             2
T+3_month_activity             2
T+6_month_activity             2
T+12_month_activity            2
Transactor_revolver            2
 avg_spends_l3m             8095
Occupation_at_source           6
cc_limit                      99
new_column                     0
dtype: int64
```

In [10]:
```
# get an information of data..
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8448 entries, 0 to 8447
Data columns (total 29 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   userid                     8448 non-null   int64
 1   card_no                    8448 non-null   object
 2   card_bin_no                8448 non-null   int64
 3   Issuer                     8448 non-null   object
 4   card_type                  8448 non-null   object
 5   card_source_date           8448 non-null   object
 6   high_networth              8448 non-null   object
 7   active_30                  8448 non-null   int64
 8   active_60                  8448 non-null   int64
 9   active_90                  8448 non-null   int64
 10  cc_active30                8448 non-null   int64
 11  cc_active60                8448 non-null   int64
 12  cc_active90                8448 non-null   int64
 13  hotlist_flag               8448 non-null   object
 14  widget_products            8448 non-null   int64
 15  engagement_products        8448 non-null   int64
 16   annual_income_at_source   8448 non-null   object
 17  other_bank_cc_holding      8448 non-null   object
 18  bank_vintage               8448 non-null   int64
 19  T+1_month_activity         8448 non-null   int64
 20  T+2_month_activity         8448 non-null   int64
 21  T+3_month_activity         8448 non-null   int64
 22  T+6_month_activity         8448 non-null   int64
 23  T+12_month_activity        8448 non-null   int64
 24  Transactor_revolver        8410 non-null   object
 25   avg_spends_l3m            8448 non-null   object
 26  Occupation_at_source       8448 non-null   object
 27  cc_limit                   8448 non-null   int64
```

```
 28  new_column                0 non-null      float64
dtypes: float64(1), int64(17), object(11)
memory usage: 1.9+ MB
```

In [11]:
```
# see the statistical summary of data..
df.describe()
```

Out[11]:

	userid	card_bin_no	active_30	active_60	active_90	cc_active30	cc_active60	cc_ac
count	8448.00000	8448.000000	8448.000000	8448.000000	8448.000000	8448.000000	8448.000000	8448
mean	4224.50000	436747.044508	0.292377	0.494792	0.642045	0.284091	0.484493	0
std	2438.87187	30489.752417	0.454881	0.500002	0.479427	0.451007	0.499789	0
min	1.00000	376916.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
25%	2112.75000	426241.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
50%	4224.50000	437551.000000	0.000000	0.000000	1.000000	0.000000	0.000000	1
75%	6336.25000	438439.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1
max	8448.00000	524178.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1

In [12]:
```
# groupby of columns or variables for comparison..
df[['card_no','card_type']]
```

Out[12]:

	card_no	card_type
0	4384 39XX XXXX XXXX	edge
1	4377 48XX XXXX XXXX	prosperity
2	4377 48XX XXXX XXXX	rewards
3	4258 06XX XXXX XXXX	indianoil
4	4377 48XX XXXX XXXX	edge
...	...	...
8443	4262 41XX XXXX XXXX	chartered
8444	37691 6XXXX XXXXX	centurion
8445	4375 51XX XXXX XXXX	rewards
8446	4477 47XX XXXX XXXX	indianoil
8447	4262 41XX XXXX XXXX	rewards

8448 rows × 2 columns

In [13]:
```
# groupby for further analysis with some of the columns of data..
df[['card_no','card_type','userid','Issuer','hotlist_flag']]
```

Out[13]:

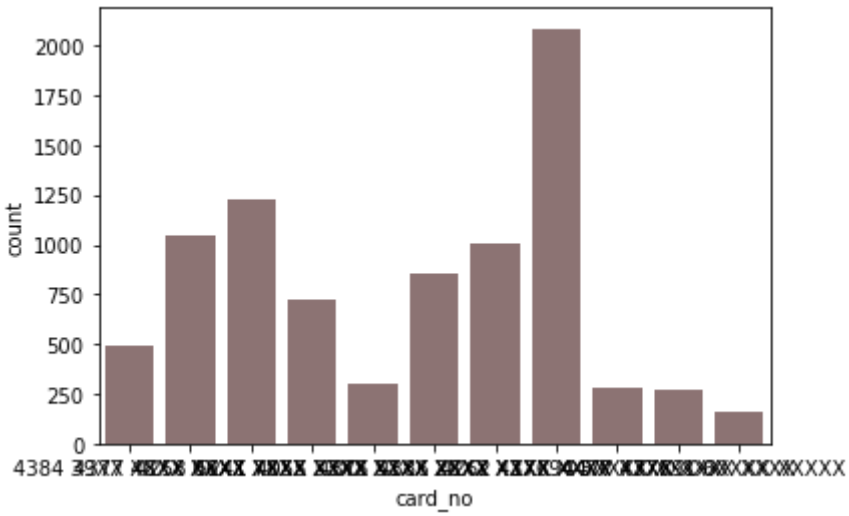
	card_no	card_type	userid	Issuer	hotlist_flag
0	4384 39XX XXXX XXXX	edge	1	Visa	N
1	4377 48XX XXXX XXXX	prosperity	2	Visa	N
2	4377 48XX XXXX XXXX	rewards	3	Visa	N

	card_no	card_type	userid	Issuer	hotlist_flag
3	4258 06XX XXXX XXXX	indianoil	4	Visa	N
4	4377 48XX XXXX XXXX	edge	5	Visa	N
...	...	...	...	...	...
8443	4262 41XX XXXX XXXX	chartered	8444	Visa	N
8444	37691 6XXXX XXXXX	centurion	8445	Amex	N
8445	4375 51XX XXXX XXXX	rewards	8446	Visa	N
8446	4477 47XX XXXX XXXX	indianoil	8447	Visa	N
8447	4262 41XX XXXX XXXX	rewards	8448	Visa	N

8448 rows × 5 columns

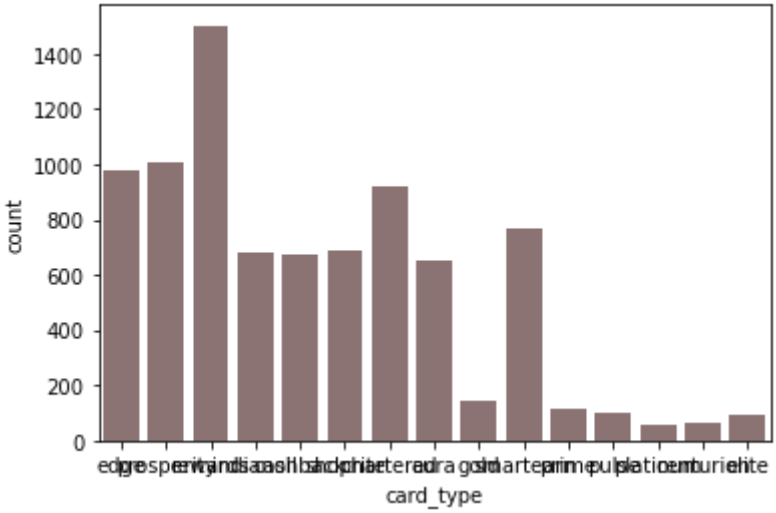
In [14]:
```
# Make a countplot of card no of data, one of the important variable of data...
sns.countplot(x='card_no',data=df, color='red', saturation=0.1)
```

Out[14]: <AxesSubplot:xlabel='card_no', ylabel='count'>



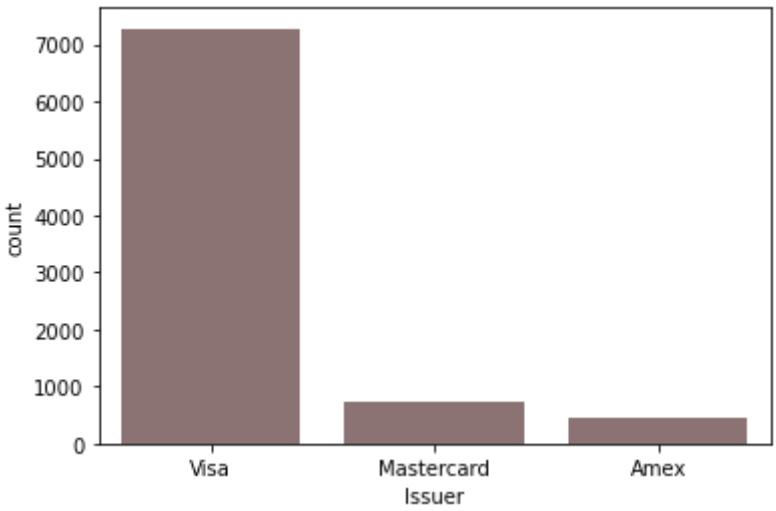
In [15]:
```
# Make a count plot of card type of data... it tells us which card type is using more by the customers,
sns.countplot(x='card_type',data=df, color='red', saturation=0.1)
```

Out[15]: <AxesSubplot:xlabel='card_type', ylabel='count'>



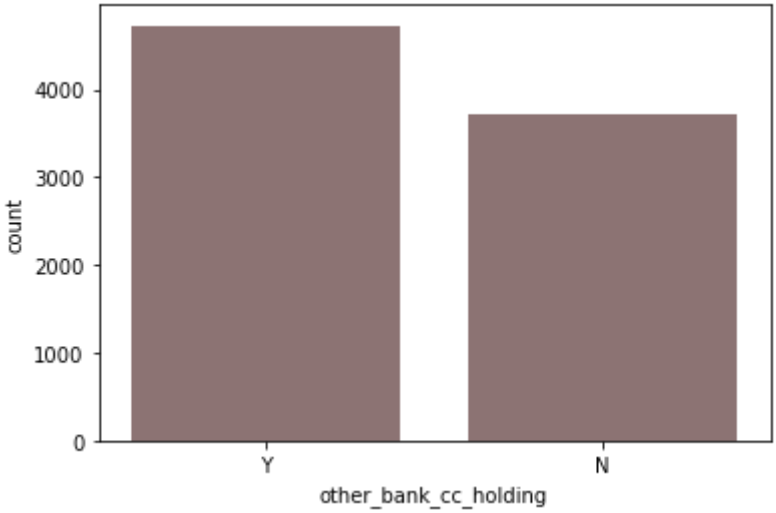
In [16]:
```
# Make a countplot of issuer which tells us which issuer is using more by the customers of bank..
sns.countplot(x='Issuer',data=df, color='red', saturation=0.1)
```

Out[16]: <AxesSubplot:xlabel='Issuer', ylabel='count'>



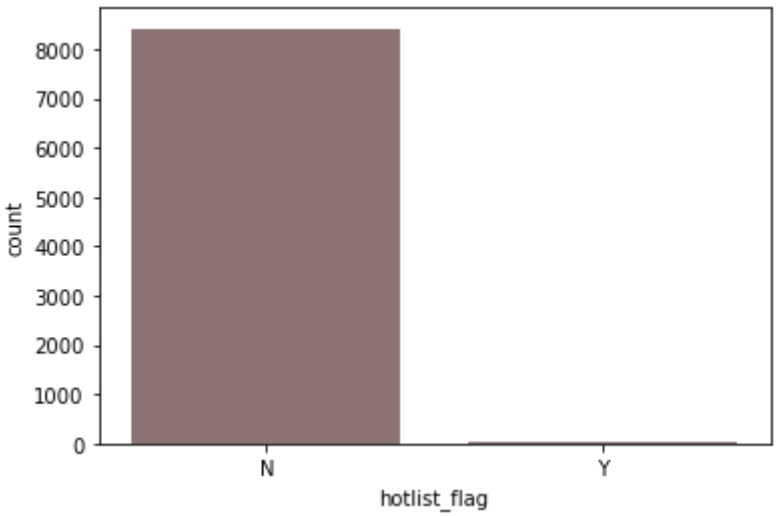
In [17]:
```
# Make a countplot of other_bank_cc_holding , it shows how many customers holding other banks cards ..
sns.countplot(x='other_bank_cc_holding',data=df, color='red', saturation=0.1)
```

Out[17]: <AxesSubplot:xlabel='other_bank_cc_holding', ylabel='count'>



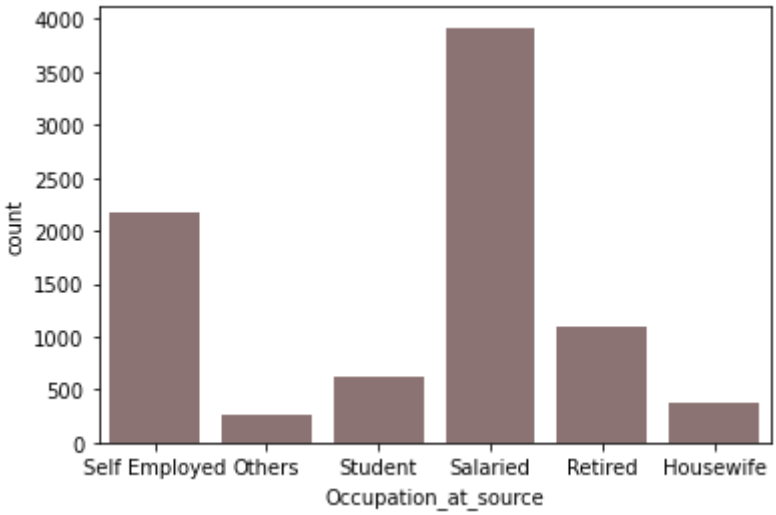
In [18]:
```
# Make a countplot of hotlist_flag, it tells us about how many customer's card blocked by the bank.
sns.countplot(x='hotlist_flag',data=df, color='red', saturation=0.1)
```

Out[18]: <AxesSubplot:xlabel='hotlist_flag', ylabel='count'>



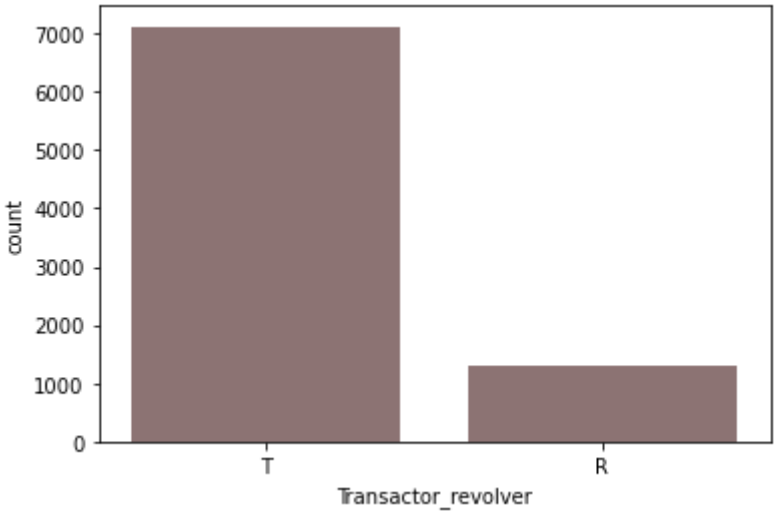
In [19]:
```
# Make a countplot of Occupation_at_source, it tells us about which category of occupation is higher,
# salaried and sel employed are using cards more than the other categories...
sns.countplot(x='Occupation_at_source',data=df, color='red', saturation=0.1)
```

Out[19]: <AxesSubplot:xlabel='Occupation_at_source', ylabel='count'>



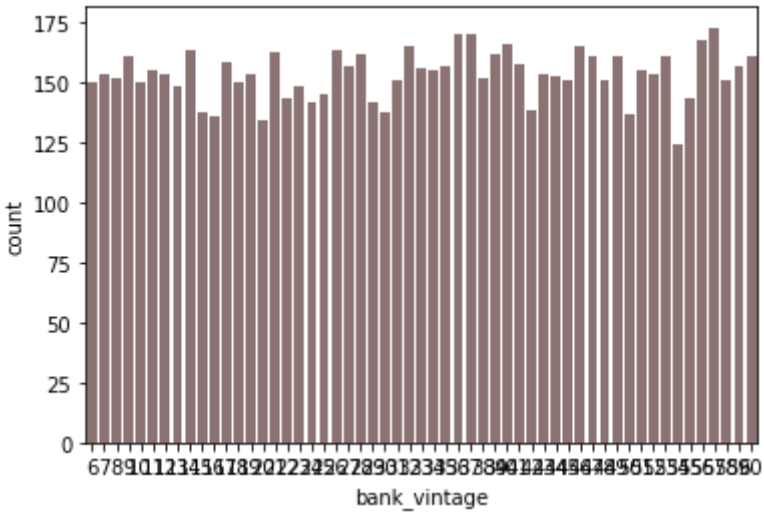
In [20]:
```
# Make a countplot of Transactor_revolver , it tells us about how many people are transactor revolver
# means how many people lend money from the bank by the card and return back as a averagly low or
# minimum amount
sns.countplot(x='Transactor_revolver',data=df, color='red', saturation=0.1)
```

Out[20]: <AxesSubplot:xlabel='Transactor_revolver', ylabel='count'>



In [21]:
```
#Make a countplot of bank_vintage...
sns.countplot(x='bank_vintage',data=df, color='red',saturation=0.1)
```

Out[21]: <AxesSubplot:xlabel='bank_vintage', ylabel='count'>



In [22]:
```
# find correlation between variables of data ..including all the columns of data
corr = df.corr()
```

In [23]:
```
corr
```

Out[23]:

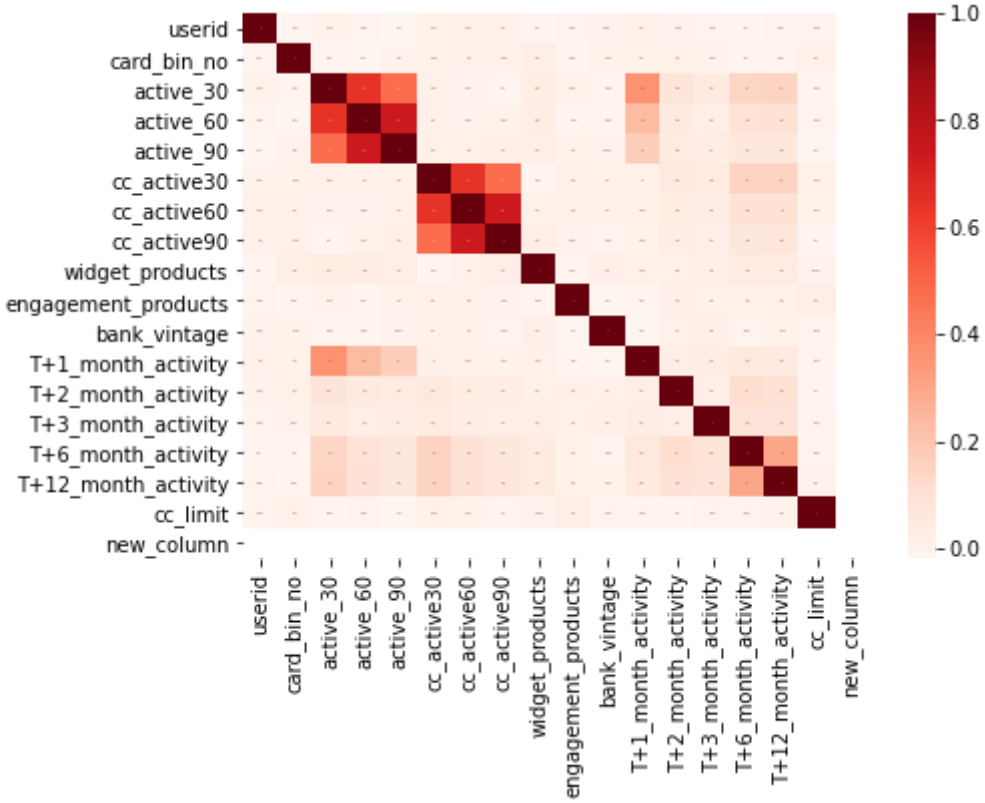
	userid	card_bin_no	active_30	active_60	active_90	cc_active30	cc_active6
userid	1.000000	-0.012930	0.004826	-0.002638	-0.014598	0.003236	0.00678
card_bin_no	-0.012930	1.000000	-0.011735	-0.017988	-0.001044	0.002825	0.01294
active_30	0.004826	-0.011735	1.000000	0.649523	0.479956	0.008249	0.00171
active_60	-0.002638	-0.017988	0.649523	1.000000	0.738936	0.006037	0.00086
active_90	-0.014598	-0.001044	0.479956	0.738936	1.000000	0.016475	0.01388
cc_active30	0.003236	0.002825	0.008249	0.006037	0.016475	1.000000	0.64979
cc_active60	0.006785	0.012942	0.001719	0.000861	0.013887	0.649790	1.00000
cc_active90	0.013257	0.007836	-0.010189	0.002859	0.018534	0.480340	0.73922
widget_products	-0.011369	0.027682	0.038809	0.037084	0.022866	-0.011778	0.01266
engagement_products	-0.004239	-0.008724	0.003273	-0.012600	0.002004	0.012220	0.00795
bank_vintage	0.000638	0.002812	-0.006061	-0.007405	-0.000413	0.016922	0.01092
T+1_month_activity	0.013422	0.003992	0.365093	0.236327	0.171547	0.009978	0.00947
T+2_month_activity	0.001757	0.015344	0.078675	0.050544	0.045042	0.053988	0.04077
T+3_month_activity	-0.004524	-0.001485	0.052145	0.027028	0.030017	0.044509	0.03313
T+6_month_activity	-0.010698	-0.011072	0.147238	0.095634	0.070668	0.150242	0.09762
T+12_month_activity	-0.005915	-0.009868	0.152112	0.098800	0.073007	0.155215	0.10085
cc_limit	-0.004585	0.009024	-0.007140	-0.004769	-0.014645	0.009937	0.00492
new_column	NaN	NaN	NaN	NaN	NaN	NaN	Na

In [24]:
```
# Draw the heatmap of variables which includes, brightest blocks have the higher value ,
```

```
# however, lightest blocks have the lowest values ...

plt.figure(figsize=(7,5))
sns.heatmap(corr, annot=True, annot_kws={'size': 0.01}, cmap="Reds")
```

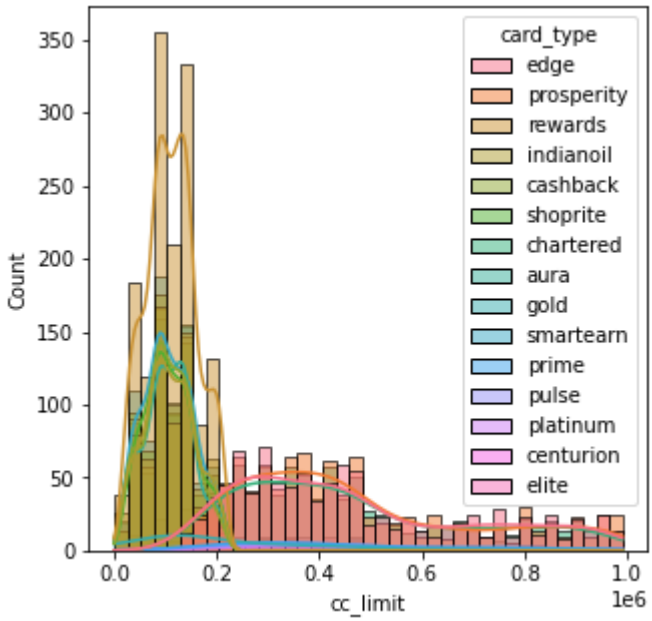
Out[24]: <AxesSubplot:>



In [25]:

```
# plot histogram for comparisons of cc_limit with card_type as hue ...
which tells us which card is
# with maximum or minimum cc limit provided by the bank.
plt.figure(figsize=(5,5))
sns.histplot(data = df, x = "cc_limit", kde = True, hue='card_type')
```

Out[25]: <AxesSubplot:xlabel='cc_limit', ylabel='Count'>

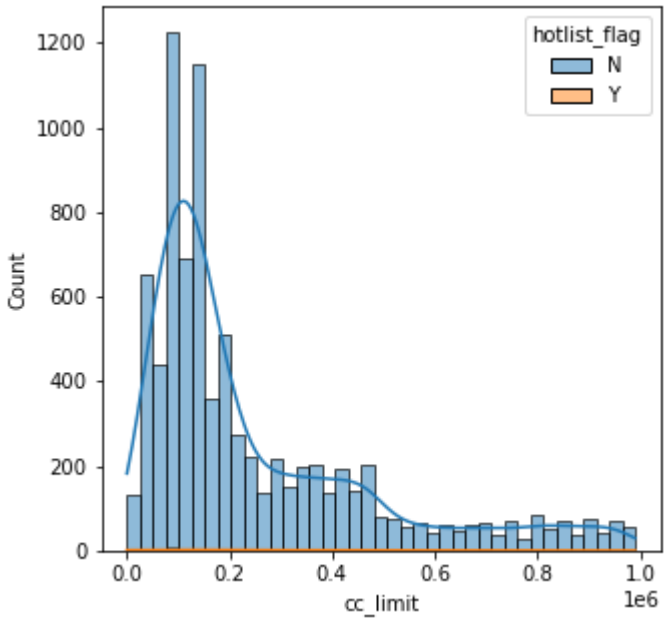


In [ ]:
```
# Blank cell
```

In [ ]:
```
# Blank cell
```

In [28]:
```
# Make a countplot of cc limit with hotlist_flag as hue, which tells us
how many customers are hotlist
# from the bank with how much cc_limit provided by the customers....
plt.figure(figsize=(5,5))
sns.histplot(data = df, x = "cc_limit", kde = True, hue='hotlist_flag')
```

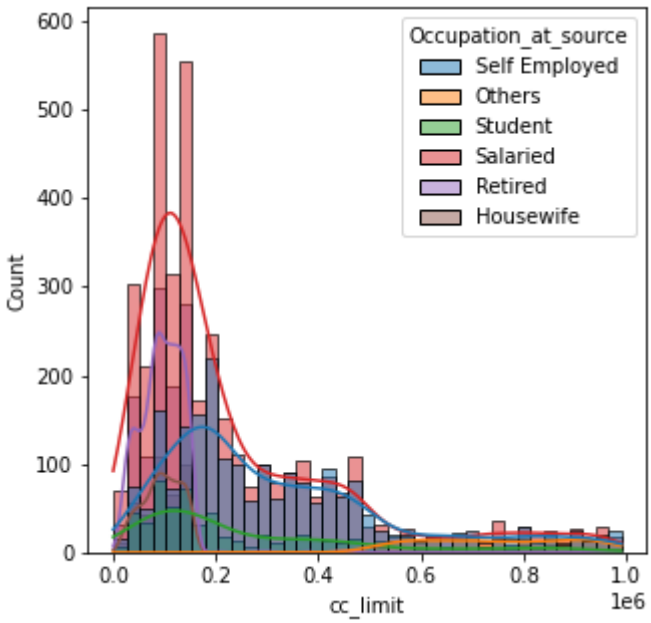
Out[28]: <AxesSubplot:xlabel='cc_limit', ylabel='Count'>



In [29]:
```
# Make a countplot of cc limit with Occupation_at_source as hue, which
tells us which occupation is
# receving higher credit limit from the bank..
```

```
plt.figure(figsize=(5,5))
sns.histplot(data = df, x = "cc_limit", kde = True,
hue='Occupation_at_source')
```

Out[29]: <AxesSubplot:xlabel='cc_limit', ylabel='Count'>



In [52]:
```
# groupby with card type and cc limit with decription ...
df.groupby(['card_type',"cc_limit"]).describe().unstack()
```

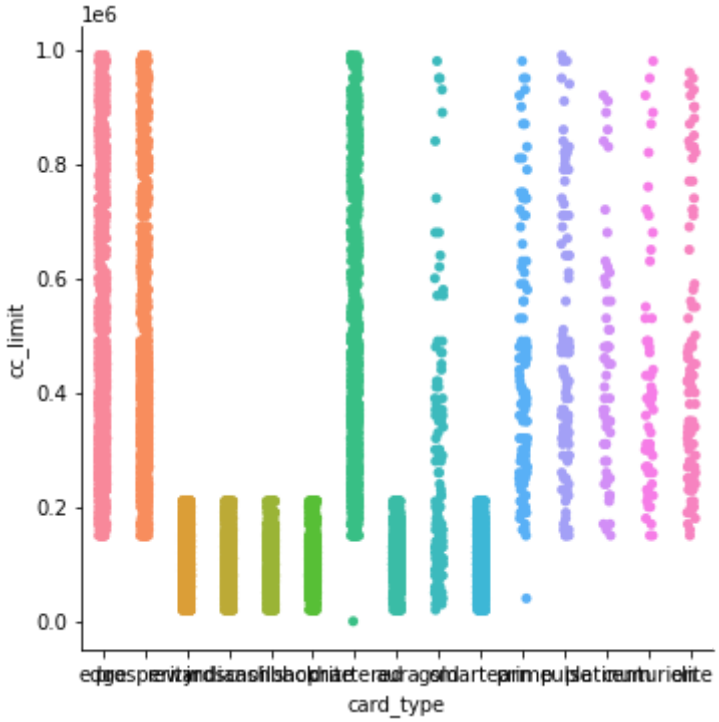
Out[52]:

	userid	...										
	count	...										
cc_limit	0	20000	30000	40000	50000	60000	70000	80000	90000	100000	...	900000
card_type												
aura	NaN	21.0	38.0	22.0	27.0	24.0	30.0	51.0	60.0	48.0	...	NaN
cashback	NaN	14.0	39.0	30.0	21.0	28.0	30.0	57.0	78.0	41.0	...	NaN
centurion	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
chartered	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
edge	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
elite	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
gold	NaN	1.0	3.0	4.0	4.0	2.0	2.0	9.0	6.0	7.0	...	NaN
indianoil	NaN	26.0	27.0	25.0	27.0	32.0	31.0	68.0	51.0	48.0	...	NaN
platinum	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
prime	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
prosperity	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
pulse	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
rewards	NaN	38.0	55.0	67.0	62.0	63.0	56.0	121.0	124.0	110.0	...	NaN
shoprite	NaN	14.0	24.0	32.0	38.0	33.0	36.0	57.0	67.0	42.0	...	NaN
smartearn	NaN	19.0	34.0	39.0	37.0	40.0	35.0	64.0	72.0	52.0	...	NaN

15 rows × 13464 columns

In [5]:
```
# Make a catplot of card type and cc limit which shows how much cc limit
getting by which card type
# from the bank.
sns.catplot(data=df, x= "card_type",y="cc_limit",kind="strip")
```

Out[5]: <seaborn.axisgrid.FacetGrid at 0x2662a6945e0>



In [32]:
```
# making pivot table with some important variables of data ...
table = pd.pivot_table(data=df,index=
['cc_limit','Occupation_at_source','Transactor_revolver','hotlist_flag'])
table
```

Out[32]:

				T+12_month_activity	T+1_month_activ
cc_limit	**Occupation_at_source**	**Transactor_revolver**	**hotlist_flag**		
20000	**Housewife**	**T**	**N**	0.000000	0.000
	Retired	**R**	**N**	0.000000	0.000
		T	**N**	0.000000	0.166
	Salaried	**R**	**N**	0.062500	0.187
		T	**N**	0.037736	0.113
...	...	...	...	...	
980000	**Student**	**T**	**N**	0.000000	0.000
990000	**Others**	**R**	**N**	0.000000	0.000
		T	**N**	0.000000	0.000
	Salaried	**T**	**N**	0.000000	0.000

					T+12_month_activity	T+1_month_activ
cc_limit	Occupation_at_source	Transactor_revolver	hotlist_flag			
	Self Employed	T	N		0.000000	0.000

628 rows × 16 columns

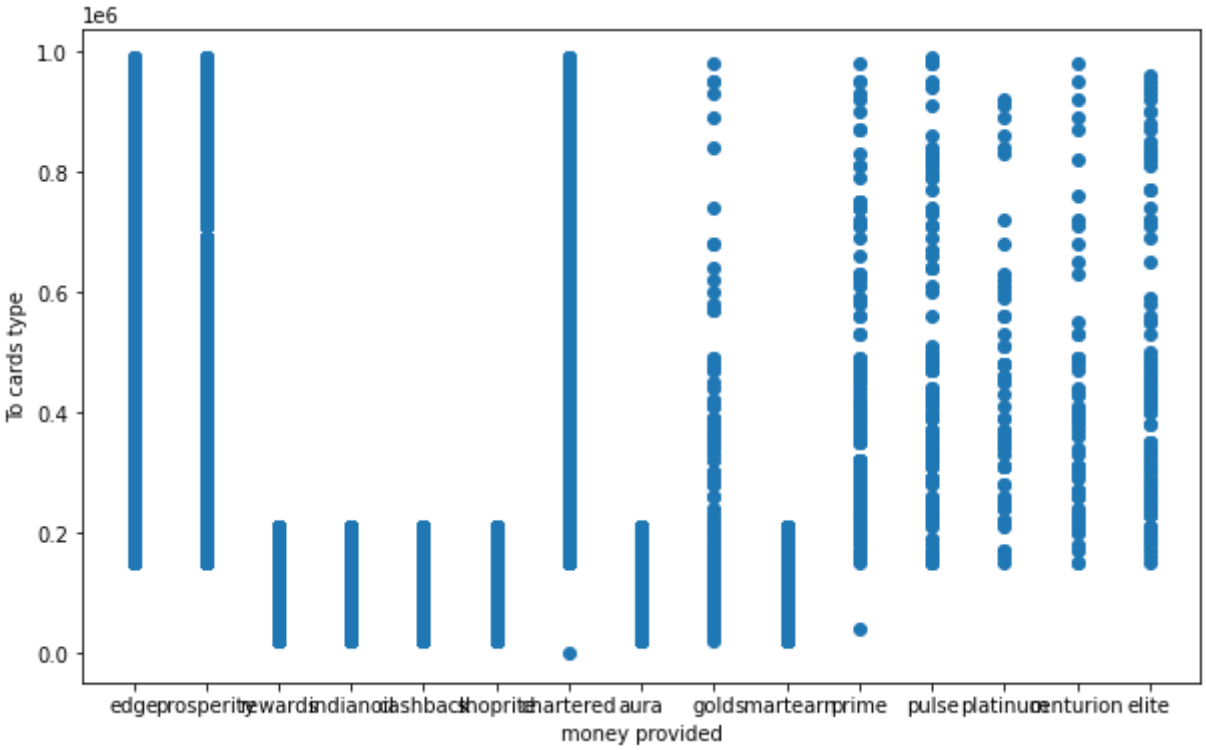
In [33]:
```
# Change the category into integer by using astype function...
df['Transactor_revolver'] = df['Transactor_revolver'].astype('category')
df.describe(include='category')
```

Out[33]:

	Transactor_revolver
count	8410
unique	2
top	T
freq	7115

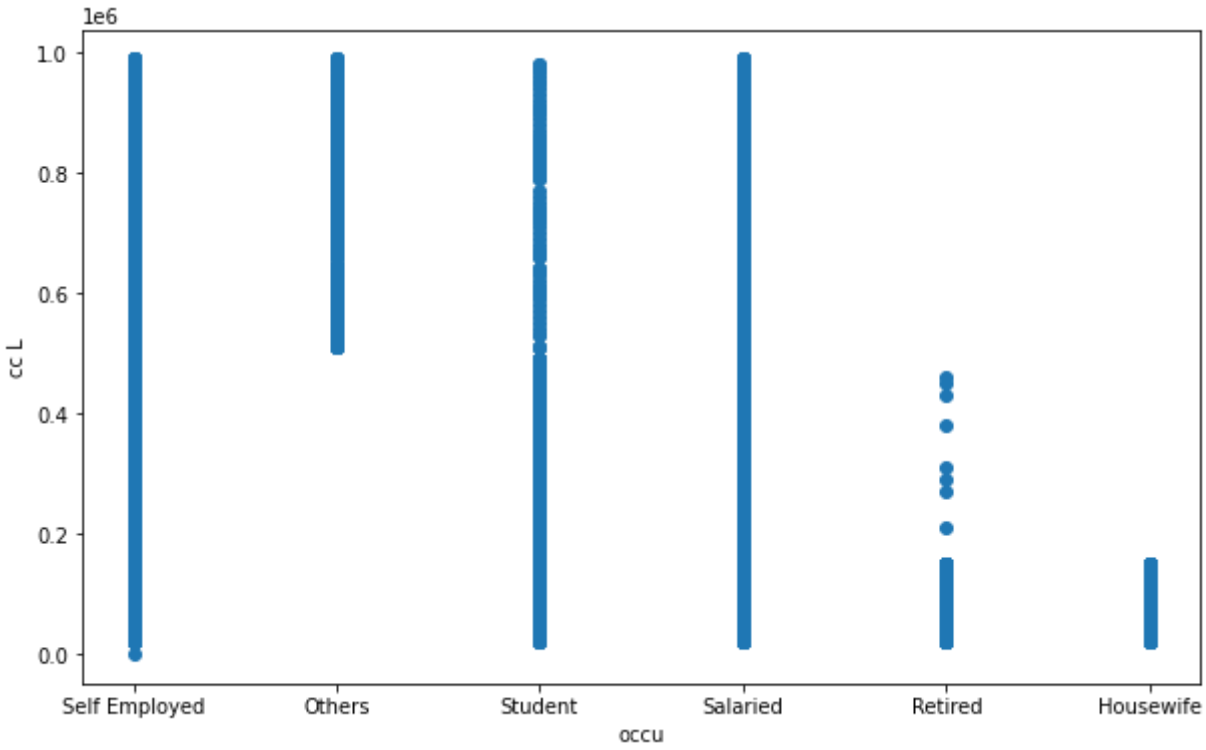
In [34]:
```
# Draw scatter plots for deeper insights of data...it shows the credit
type with provided cc limit
# by the customers...
fig, ax = plt.subplots(figsize=(10, 6))
ax.scatter(x = df['card_type'], y = df['cc_limit'])
plt.xlabel("money provided")
plt.ylabel("To cards type")

plt.show()
```

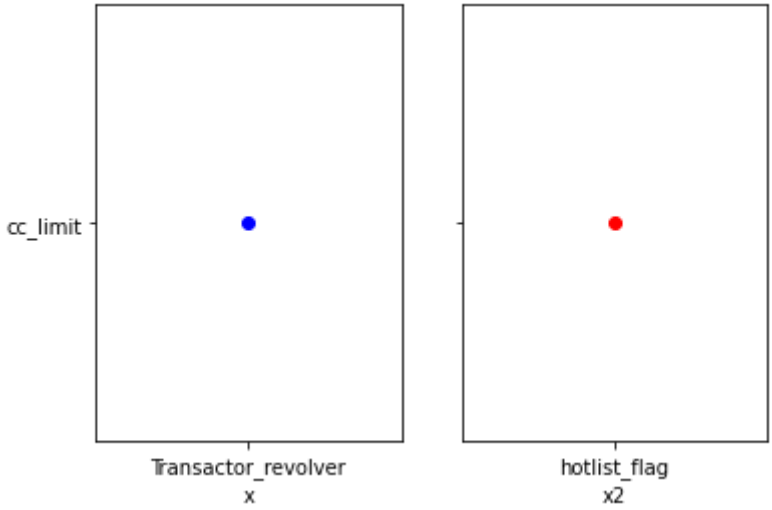


In [35]:
```
# make a scatter plot which tells us about the connection between
occupation and cc_limit.
fig, ax = plt.subplots(figsize=(10, 6))
ax.scatter(x = df['Occupation_at_source'], y = df['cc_limit'])
plt.xlabel("occu")
plt.ylabel("cc L")

plt.show()
```



In [36]:
```
# Make a subplots of transactor revolver with cc limit and hotlist_flag
with cc_limit.
fig, (ax1, ax2) = plt.subplots(1, 2, sharey= True)  # 1 row, 2 columns
ax1.scatter("Transactor_revolver", "cc_limit", c='blue')
ax2.scatter("hotlist_flag", "cc_limit", c='red')
ax1.set_xlabel('x')
ax2.set_xlabel('x2')
plt.show()
```



In [37]:
```
# pivot table with some important columns ...
table = pd.pivot_table(data=df,index=
['cc_limit','card_type','hotlist_flag'])
table
```

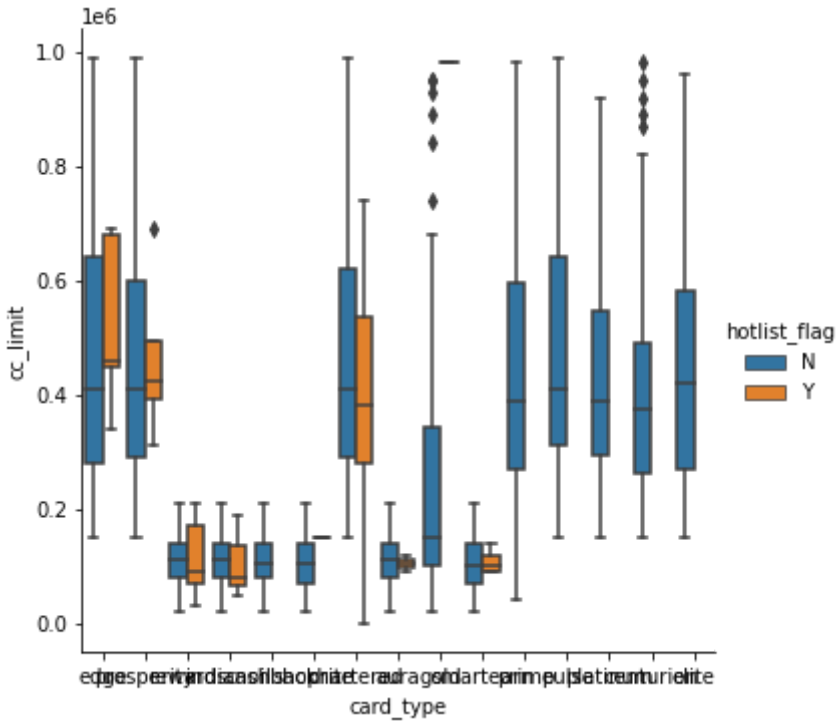
Out[37]:

			T+12_month_activity	T+1_month_activity	T+2_month_activity	T+3_m
cc_limit	**card_type**	**hotlist_flag**				
0	**chartered**	**Y**	0.000000	0.000000	0.000000	
20000	**aura**	**N**	0.047619	0.142857	0.142857	
	cashback	**N**	0.000000	0.071429	0.071429	
	gold	**N**	0.000000	1.000000	0.000000	
	indianoil	**N**	0.038462	0.038462	0.038462	
...	**...**	**...**	...	...	...	
980000	**pulse**	**N**	0.000000	0.500000	0.000000	
990000	**chartered**	**N**	0.000000	0.000000	0.250000	
	edge	**N**	0.000000	0.000000	0.250000	
	prosperity	**N**	0.000000	0.000000	0.200000	
	pulse	**N**	0.000000	0.000000	0.000000	

704 rows × 16 columns

In [38]:
```
# draw catplot a catplot of card type and cc limit with hostlist_flag as
hue, it is showing
# which card type is using with how much limit wherein, how many cards
are hotlist or active..
sns.catplot(data=df, x= "card_type",y="cc_limit", hue=
'hotlist_flag',kind="box")
# by this visual mostly customers are using their cards
```

Out[38]: <seaborn.axisgrid.FacetGrid at 0x28174b49a90>



In [39]:
```
# Make pivot table with important variables...
table = pd.pivot_table(data=df,index=
['cc_limit','Occupation_at_source','hotlist_flag'])
table
```

Out[39]:

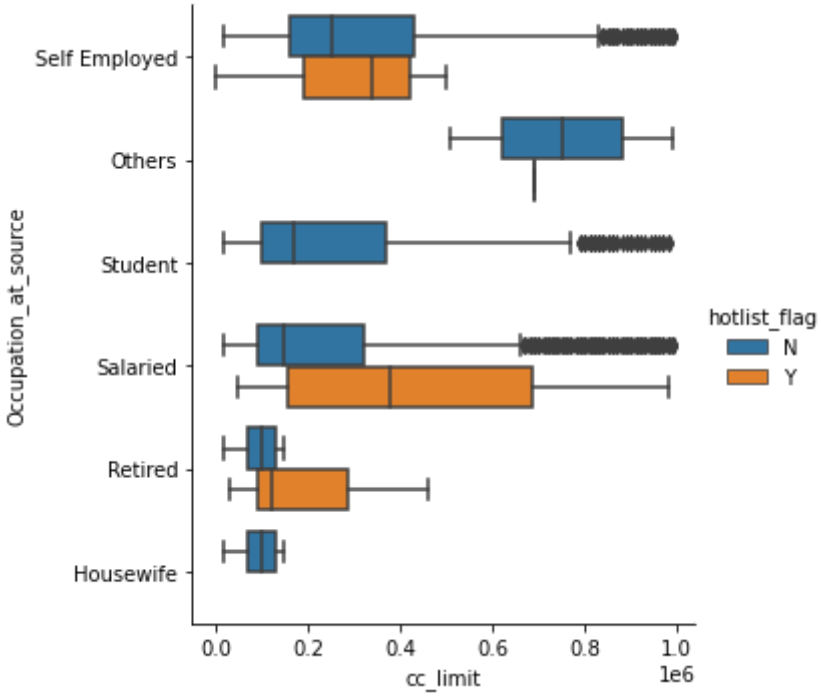
			T+12_month_activity	T+1_month_activity	T+2_month_activ
cc_limit	**Occupation_at_source**	**hotlist_flag**			
0	**Self Employed**	**Y**	0.000000	0.000000	0.000(
20000	**Housewife**	**N**	0.000000	0.000000	0.000(
	Retired	**N**	0.000000	0.156250	0.062!
	Salaried	**N**	0.043478	0.130435	0.130-
	Self Employed	**N**	0.000000	0.000000	0.000(
...	**...**	**...**	...	...	
980000	**Self Employed**	**N**	0.000000	0.250000	0.000(
	Student	**N**	0.000000	0.000000	0.000(
990000	**Others**	**N**	0.000000	0.000000	0.750(
	Salaried	**N**	0.000000	0.000000	0.000(
	Self Employed	**N**	0.000000	0.000000	0.000(

398 rows × 16 columns

In [40]:
```
# Make a catplot of cc limit and occupation with hotlist_flag as hue,
which tells us about the
# how much cc limit provided to which occupation which are blocked or
```

```
active..
sns.catplot(data=df, x= "cc_limit",y="Occupation_at_source", hue=
"hotlist_flag",kind="box")
# by this boxplot , cards are frequently used by student,Others,Self
Employed, are using ()cards but
# spending below average:
```

Out[40]: <seaborn.axisgrid.FacetGrid at 0x28171077f70>



In [41]:

```
# Make a pivot table of important columns:
table = pd.pivot_table(data=df,index=
['cc_limit','Occupation_at_source','Transactor_revolver'])
table
```

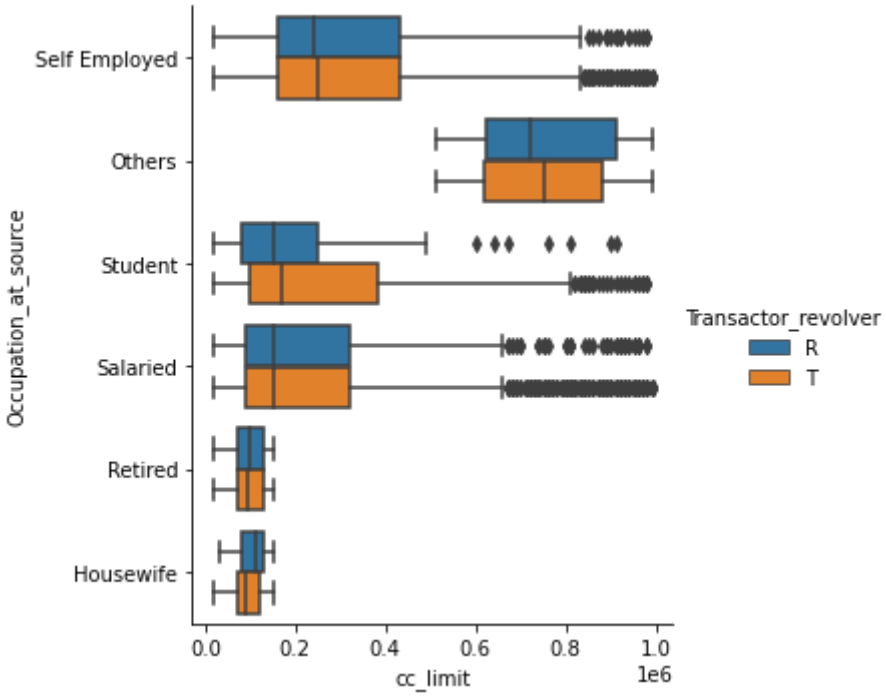
Out[41]:

			T+12_month_activity	T+1_month_activity	T+2_mo
cc_limit	**Occupation_at_source**	**Transactor_revolver**			
20000	**Housewife**	**T**	0.000000	0.000000	
	Retired	**R**	0.000000	0.000000	
		T	0.000000	0.166667	
	Salaried	**R**	0.062500	0.187500	
		T	0.037736	0.113208	
...	**...**	**...**	...	...	
980000	**Student**	**T**	0.000000	0.000000	
990000	**Others**	**R**	0.000000	0.000000	
		T	0.000000	0.000000	
	Salaried	**T**	0.000000	0.000000	
	Self Employed	**T**	0.000000	0.000000	

628 rows × 16 columns

In [42]:
```
# Make a catplot with cc_limit and occupation with transactor revolver as hue:----
sns.catplot(data=df, x= "cc_limit",y="Occupation_at_source", hue= "Transactor_revolver",kind="box")
# card is provided to highly reccommend to self employed, Others, and Salaried persons because they are
# spending averagly high and 50 percent people are transactor revolver so , bank should not provided the
# upgraded card to them.
# upgraded card only give to the categories like sel employed, salaried and others however , cards are
# also using by the student , housewife and retired category but their usage is small
# (avg spending is less than other categories),
# except student n others transactor revolver are almost 50 percent.
# therefore , upgraded card only given by the bank to those who are not flaglist, who are not transactor
# revolver, they will get high cc limit by bank.
```

Out[42]: <seaborn.axisgrid.FacetGrid at 0x28171077cd0>



In [43]:
```
# Make a pivot table of variables which we need:
table = pd.pivot_table(data=df,index=['cc_limit','Issuer','Transactor_revolver'])
table
```

Out[43]:

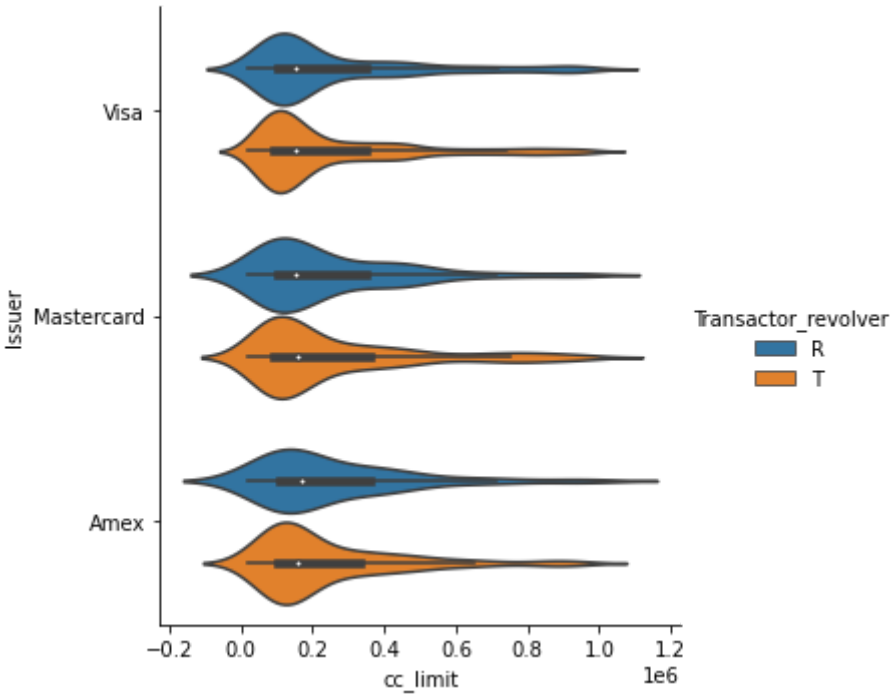
cc_limit	Issuer	Transactor_revolver	T+12_month_activity	T+1_month_activity	T+2_month_activit
20000	Amex	R	0.000000	0.000000	0.00000
		T	0.000000	0.333333	0.00000
	Mastercard	R	0.000000	0.000000	0.00000
		T	0.000000	0.125000	0.00000
	Visa	R	0.055556	0.222222	0.22222
...	...	...	...	...	
980000	Visa	R	0.000000	0.000000	0.00000
		T	0.000000	0.200000	0.00000
990000	Mastercard	T	0.000000	0.000000	0.00000
	Visa	R	0.000000	0.000000	1.00000
		T	0.000000	0.000000	0.16666

441 rows × 16 columns

In [44]:

```
# Make a catplot of cc_limit with issuer and hue is transactor revolver
here
sns.catplot(data=df, x= 'cc_limit', y="Issuer",
hue='Transactor_revolver',kind="violin")
```

Out[44]: <seaborn.axisgrid.FacetGrid at 0x281749ea4f0>



In [45]:

```
# pivot table with variables:
table = pd.pivot_table(data=df,index=
```

```
['cc_limit','card_type','Transactor_revolver'])
table
```

Out[45]:

cc_limit	card_type	Transactor_revolver	T+12_month_activity	T+1_month_activity	T+2_month_activity
20000	aura	R	0.000000	0.250000	0.250000
		T	0.058824	0.117647	0.117647
	cashback	R	0.000000	0.250000	0.000000
		T	0.000000	0.000000	0.100000
	gold	T	0.000000	1.000000	0.000000
...	...	...	...	...	...
990000	chartered	T	0.000000	0.000000	0.250000
	edge	T	0.000000	0.000000	0.250000
	prosperity	R	0.000000	0.000000	1.000000
		T	0.000000	0.000000	0.000000
	pulse	T	0.000000	0.000000	0.000000

1034 rows × 16 columns

In [46]:

```
# pivot table :
table = pd.pivot_table(data=df,index=
['cc_limit','card_type','Transactor_revolver'])
table
```

Out[46]:

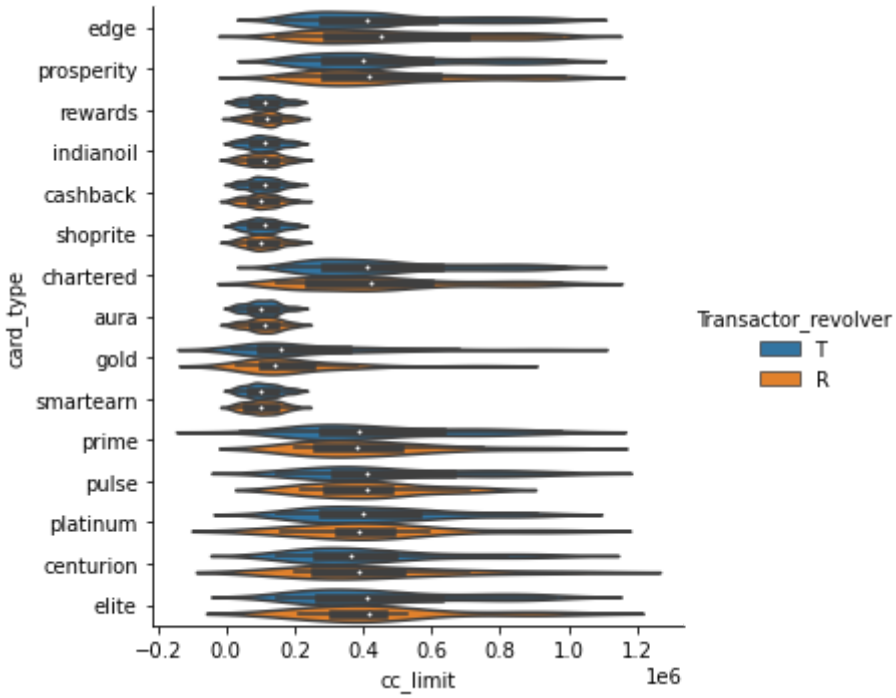
cc_limit	card_type	Transactor_revolver	T+12_month_activity	T+1_month_activity	T+2_month_activity
20000	aura	R	0.000000	0.250000	0.250000
		T	0.058824	0.117647	0.117647
	cashback	R	0.000000	0.250000	0.000000
		T	0.000000	0.000000	0.100000
	gold	T	0.000000	1.000000	0.000000
...	...	...	...	...	...
990000	chartered	T	0.000000	0.000000	0.250000
	edge	T	0.000000	0.000000	0.250000
	prosperity	R	0.000000	0.000000	1.000000
		T	0.000000	0.000000	0.000000
	pulse	T	0.000000	0.000000	0.000000

1034 rows × 16 columns

blank cell

In [6]:
```
# Make a catplot of cc limit, card type with transactor revolver as
hue,which shows the
# cc_limit with which card type and how many are transactor revolver out
of from them...
sns.catplot(data=df, x= 'cc_limit', y="card_type",
hue='Transactor_revolver',kind="violin")
```

Out[6]: <seaborn.axisgrid.FacetGrid at 0x2662d4da310>




```
 # card is provided to highly reccommend to self employed, Others,
and Salaried persons because they   are
# spending averagly high and 50 percent people are transactor
revolver so , bank should not provided the
# upgraded card to them.
# upgraded card only give to the categories like sel employed,
salaried and others however , cards are
# also using by the student , housewife and retired category but
their usage is small
# (avg spending is less than other categories),
# except student n others transactor revolver are almost 50
percent.
# therefore , upgraded card only given by the bank to those who are
not flaglist, who are not      # transactor
# revolver, they will get high cc limit by bank.
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: