```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read_csv('ppg_churn.csv')
```

1 - Number of rows and columns

```
df.shape
```

```
(5000, 20)
```

2 - Variables names and data types associated with it

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 20 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   state   5000 non-null   object 
 1   X02     5000 non-null   int64  
 2   X03     5000 non-null   object 
 3   X04     5000 non-null   object 
 4   X05     5000 non-null   object 
 5   X06     5000 non-null   int64  
 6   X07     5000 non-null   float64
 7   X08     5000 non-null   int64  
 8   X09     5000 non-null   float64
 9   X10     5000 non-null   float64
 10  X11     5000 non-null   int64  
 11  X12     5000 non-null   float64
 12  X13     5000 non-null   float64
 13  X14     5000 non-null   int64  
 14  X15     5000 non-null   float64
 15  X16     5000 non-null   float64
 16  X17     5000 non-null   int64  
 17  X18     5000 non-null   float64
 18  X19     5000 non-null   int64  
 19  churn   5000 non-null   object 
dtypes: float64(8), int64(7), object(5)
memory usage: 781.4+ KB
```

Number of missing values per variable

```
df.isna().sum()
```

```
state    0
X02      0
```

```
X03      0
X04      0
X05      0
X06      0
X07      0
X08      0
X09      0
X10      0
X11      0
X12      0
X13      0
X14      0
X15      0
X16      0
X17      0
X18      0
X19      0
churn    0
dtype: int64
```

Number of unique values per variables

```
df.nunique()
```

```
state      51
X02       218
X03         3
X04         2
X05         2
X06        48
X07      1961
X08       123
X09      1961
X10      1879
X11       126
X12      1659
X13      1853
X14       131
X15      1028
X16       170
X17        21
X18       170
X19        10
churn       2
dtype: int64
```

For categorical data

```
df.X03.value_counts()
```

```
AA    2495
BB    1259
```

```
CC    1246
Name: X03, dtype: int64
```

```
df.X04.value_counts()
```

```
Z1    4527
Z2     473
Name: X04, dtype: int64
```

```
df.X05.value_counts()
```

```
V2    3677
V1    1323
Name: X05, dtype: int64
```

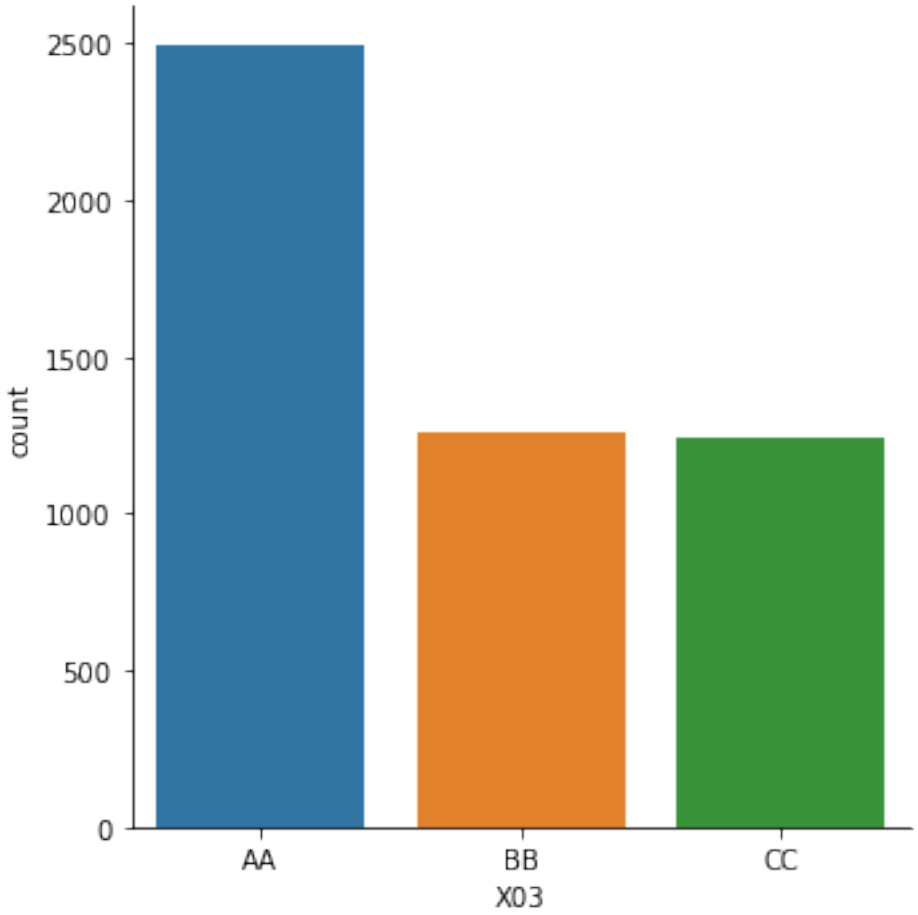
```
df.churn.value_counts()
```

```
no     4293
yes     707
Name: churn, dtype: int64
```

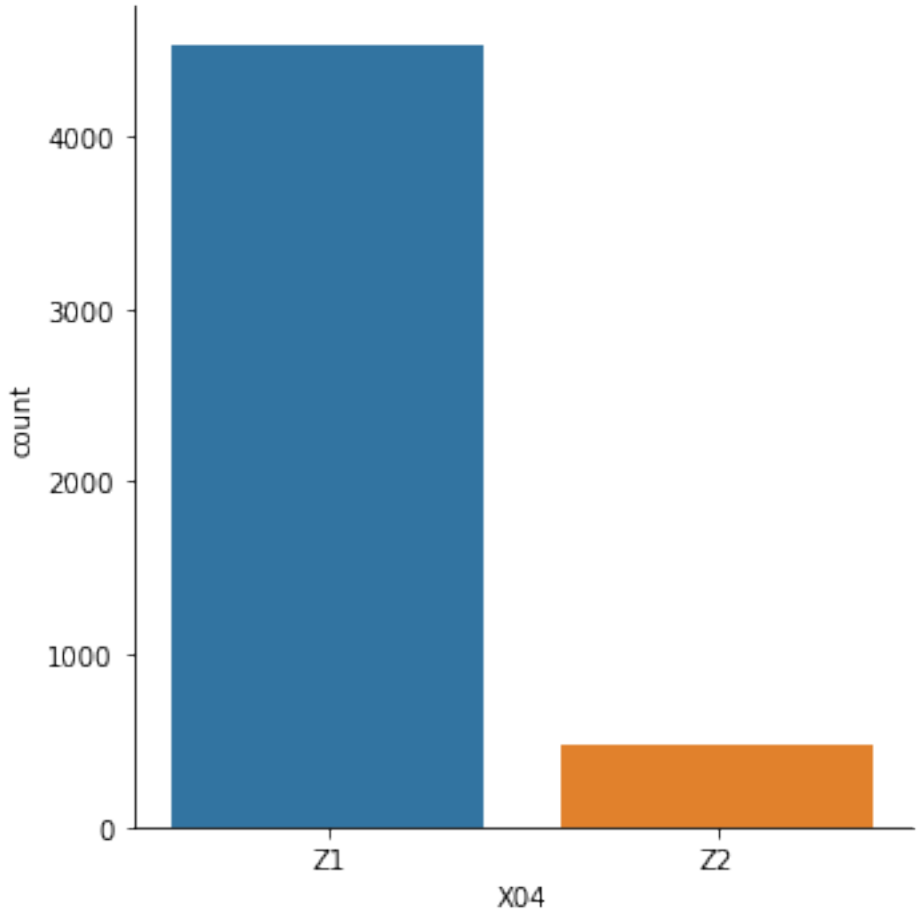
The state variable is an object data type. This it is an categorical variable even though it has 51 unique values

The X03,X04,X05 have less than 4 unique values and the `.info()` method revealed these variables are also object data type. So they can also be assumed as categorical variables

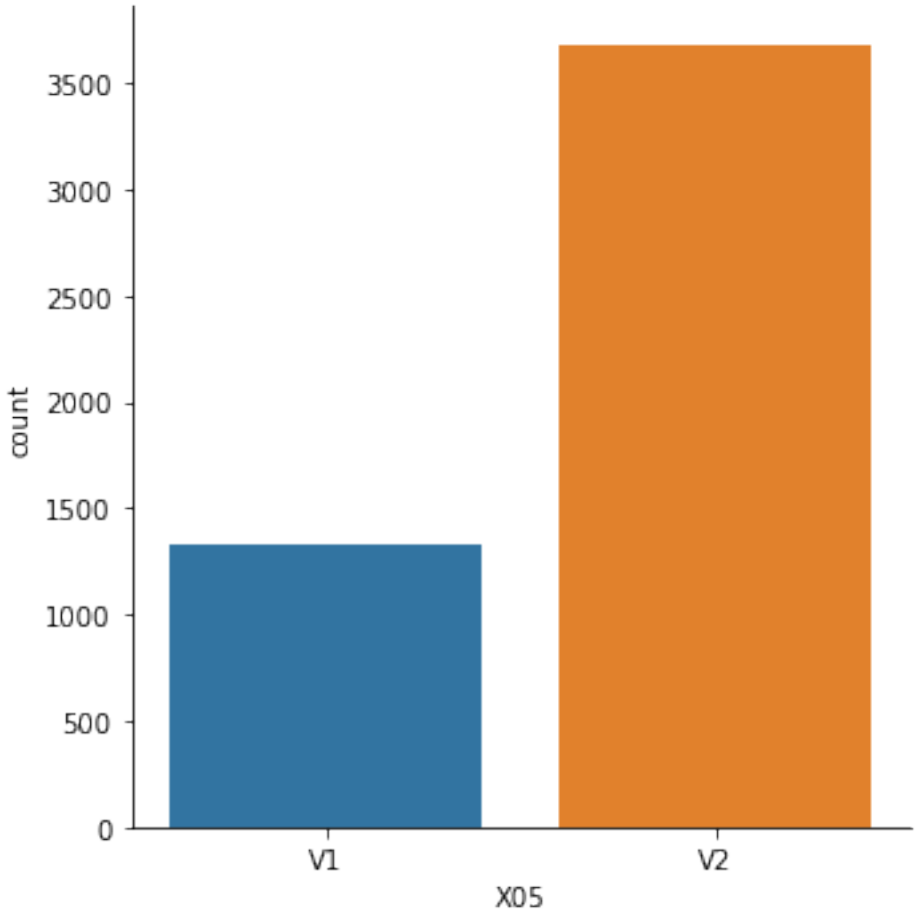
```
sns.catplot(data=df, x='X03', kind='count')

plt.show()
```



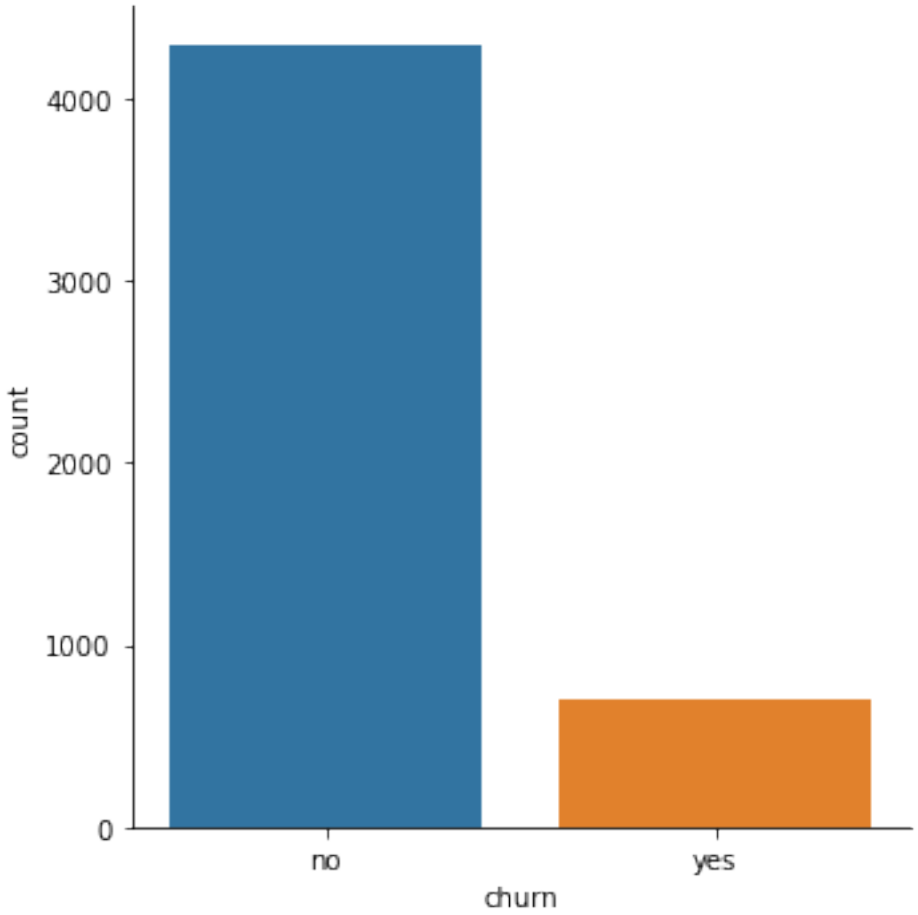

```
sns.catplot(data=df, x='X04', kind='count')

plt.show()
```



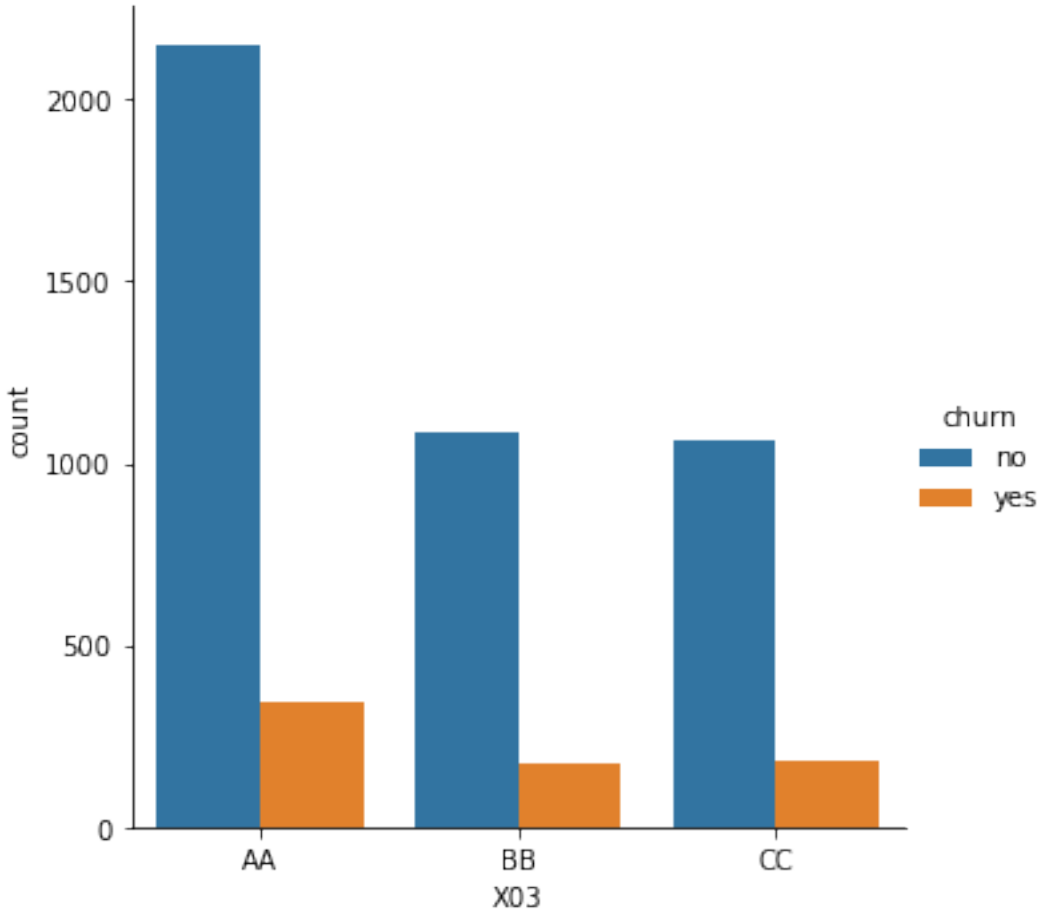

```
sns.catplot(data=df, x='X05', kind='count')

plt.show()
```



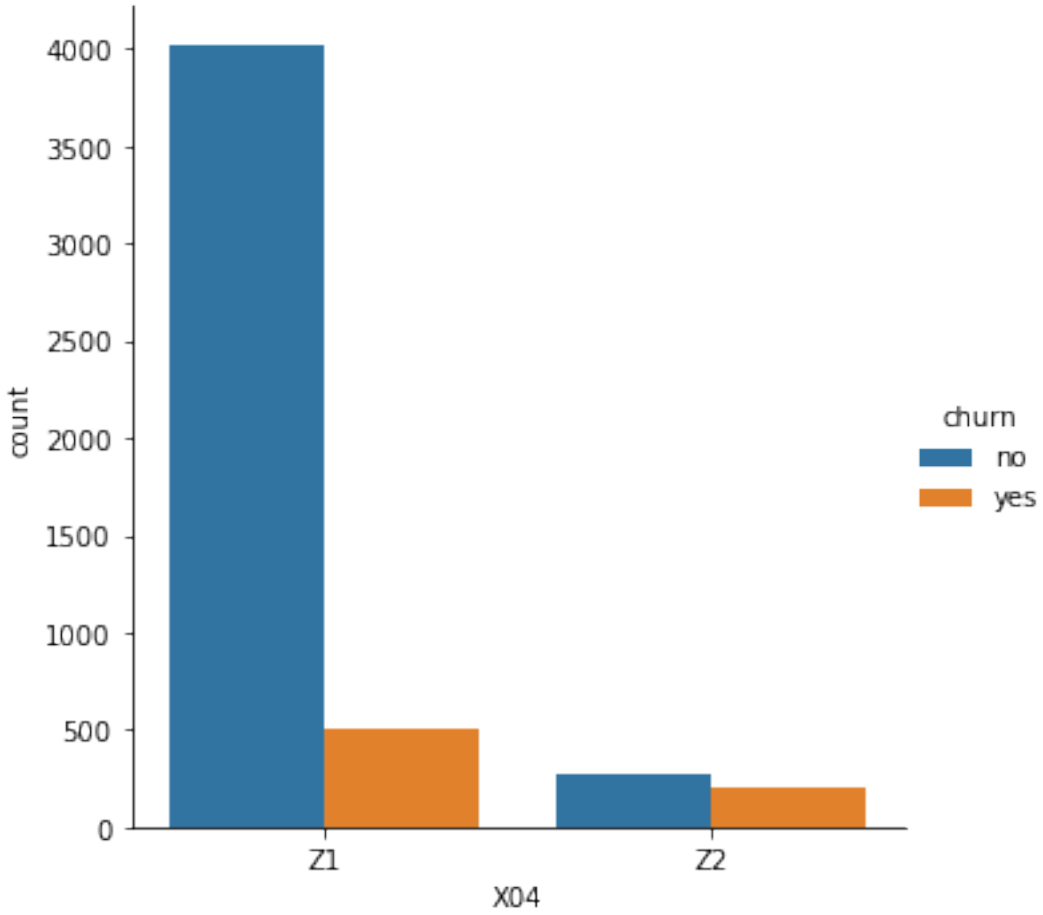

```
sns.catplot(data=df, x='churn', kind='count')

plt.show()
```



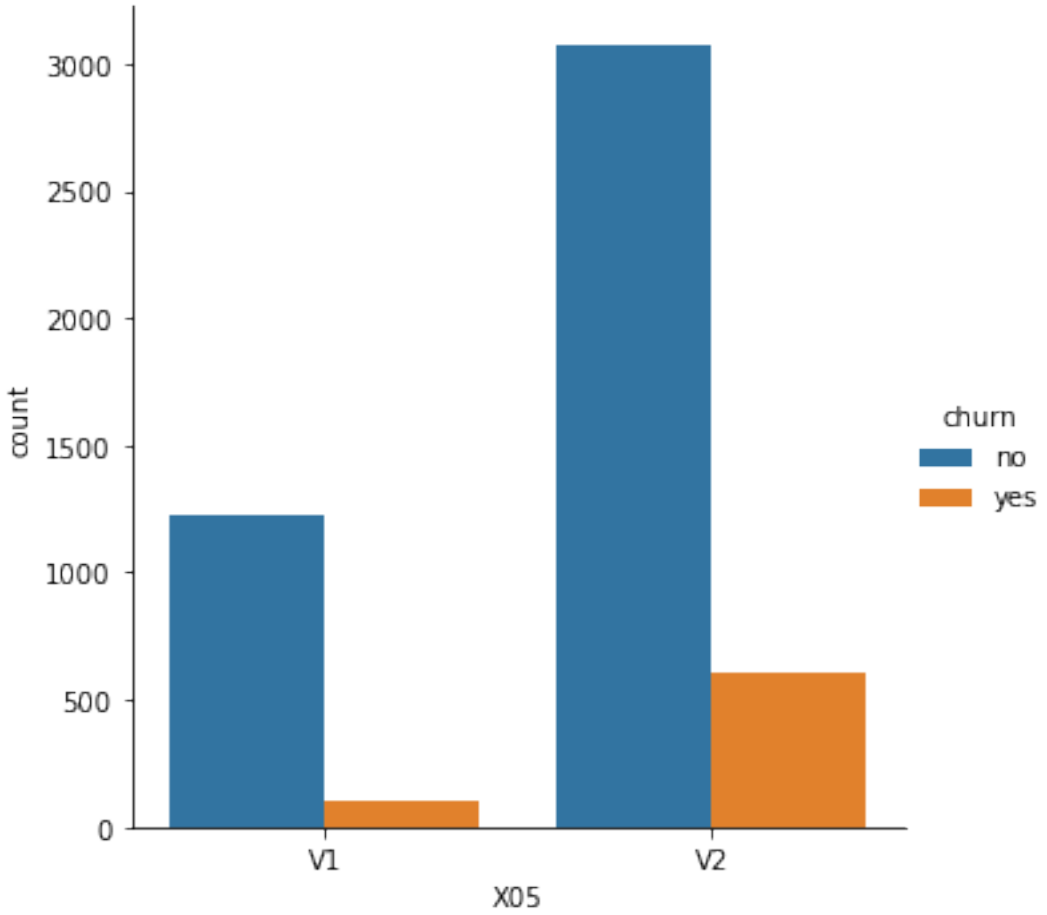

```
sns.catplot(data=df, x='X03', hue='churn', kind='count')

plt.show()
```




```
sns.catplot(data=df, x='X04', hue='churn', kind='count')

plt.show()
```

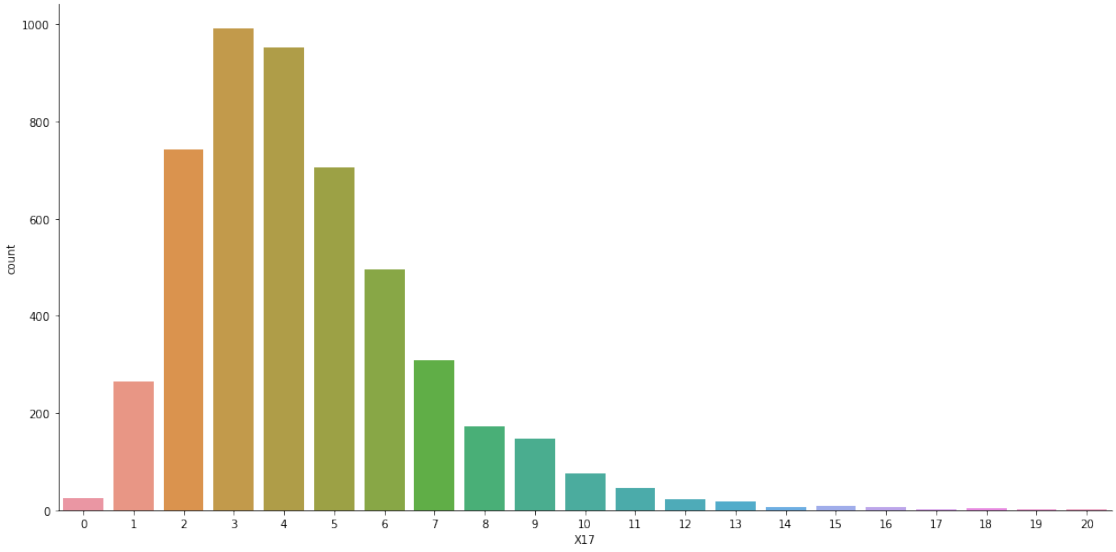



```
sns.catplot(data=df, x='X05', hue='churn', kind='count')

plt.show()
```

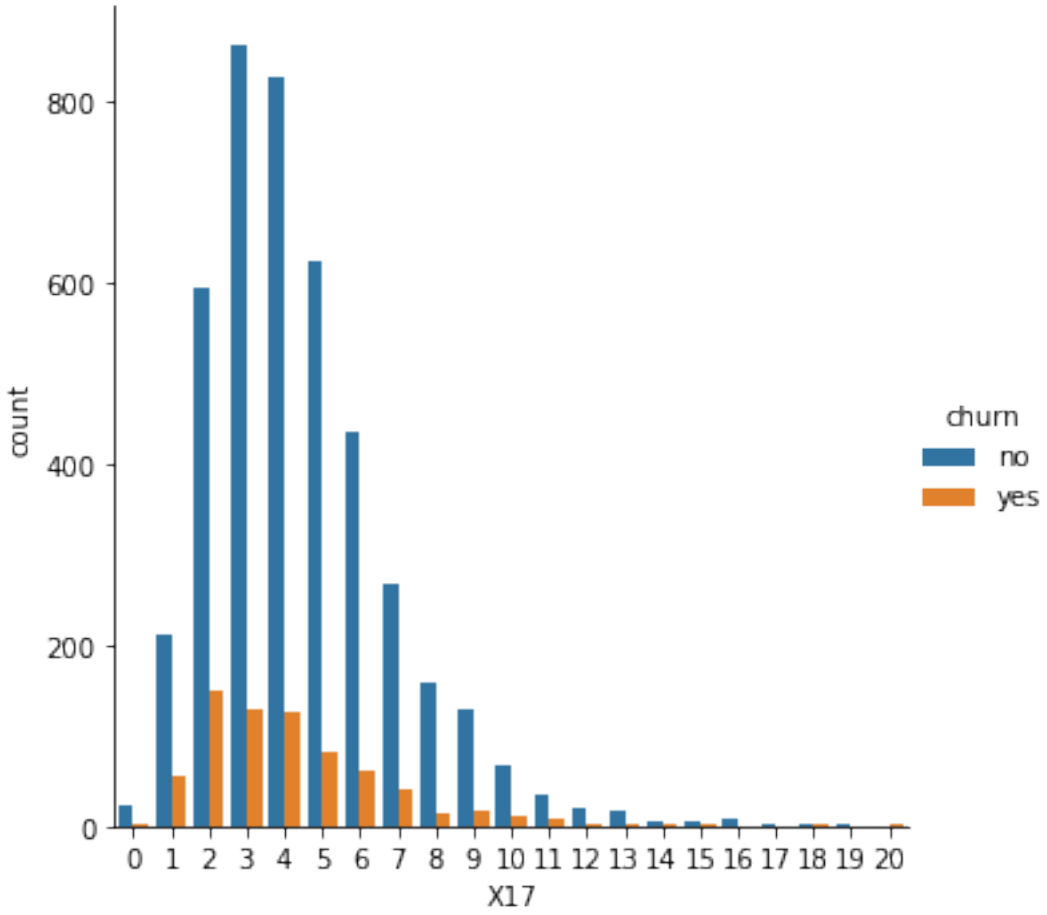


The variables X17 and X19 have less than 25 unique values and both are integer data types.

```
sns.catplot(data=df, x='X17', kind='count', height=7, aspect=2.0)

plt.show()
```

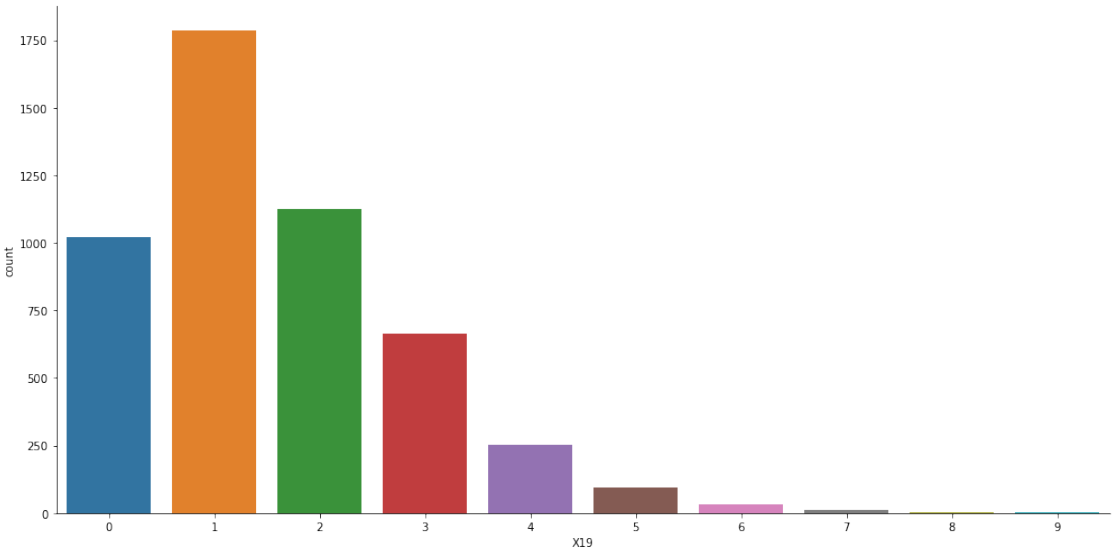


```
sns.catplot(data=df, x='X17', hue='churn', kind='count')

plt.show()
```

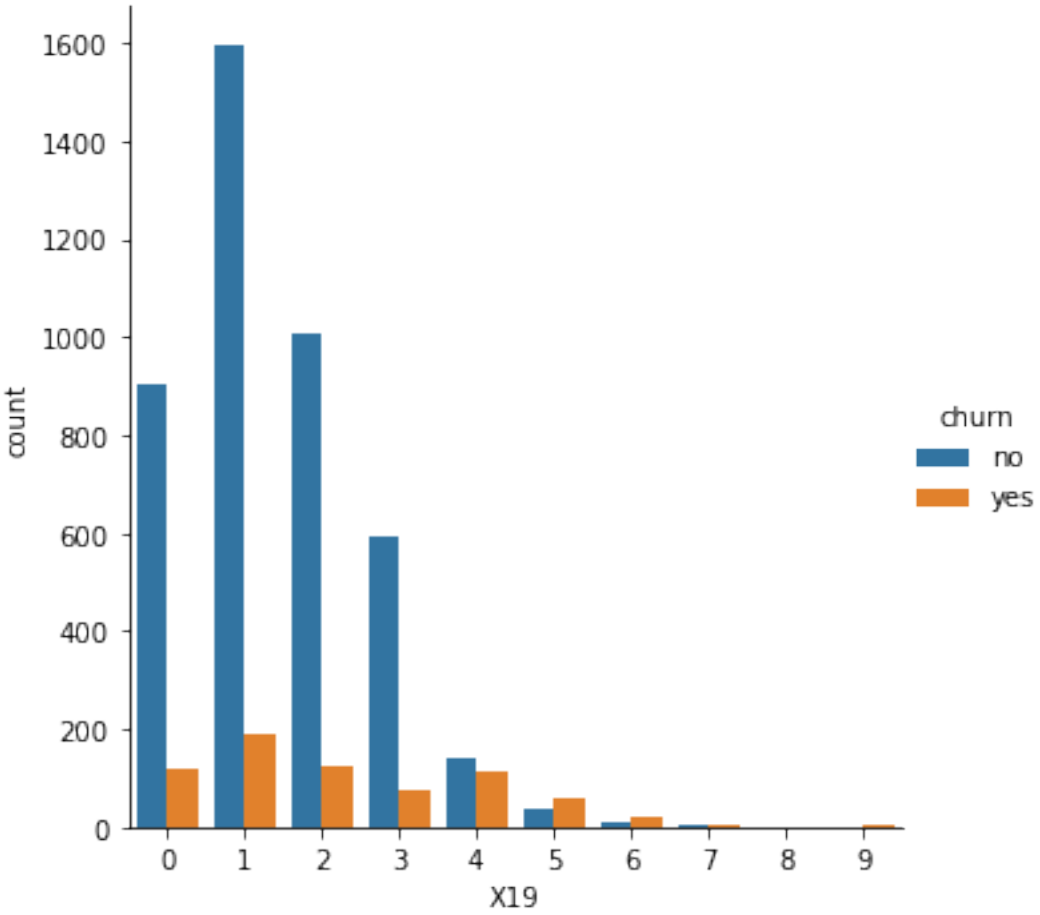


We can see that the values greater than 12 have very low frequencies.

```
sns.catplot(data=df, x='X19', kind='count', height=7, aspect=2)

plt.show()
```

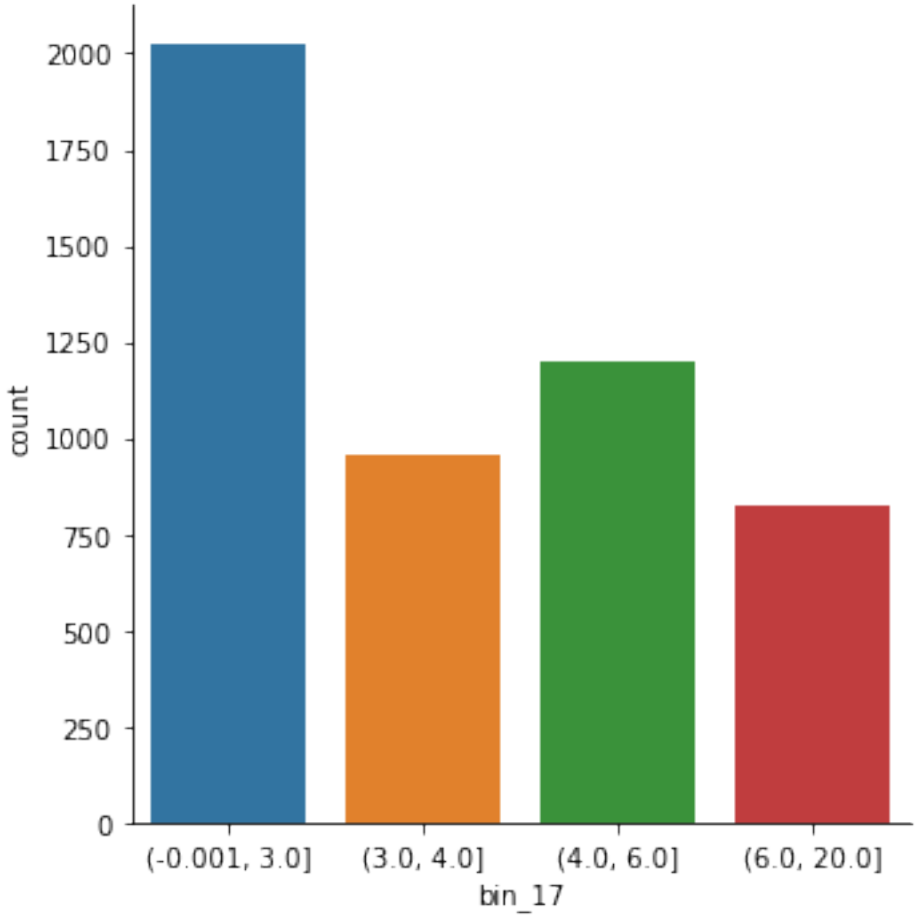



```
sns.catplot(data=df, x='X19', hue='churn', kind='count')

plt.show()
```

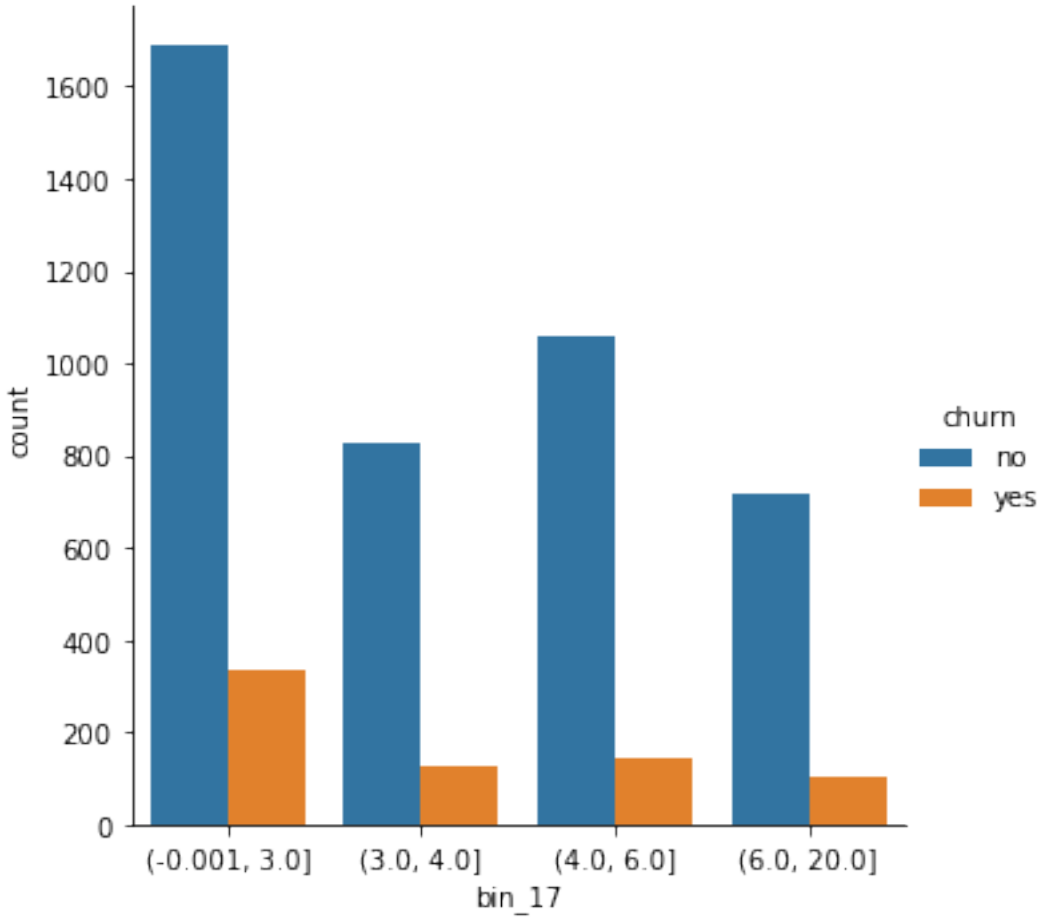



```
df_c = df.copy()
```

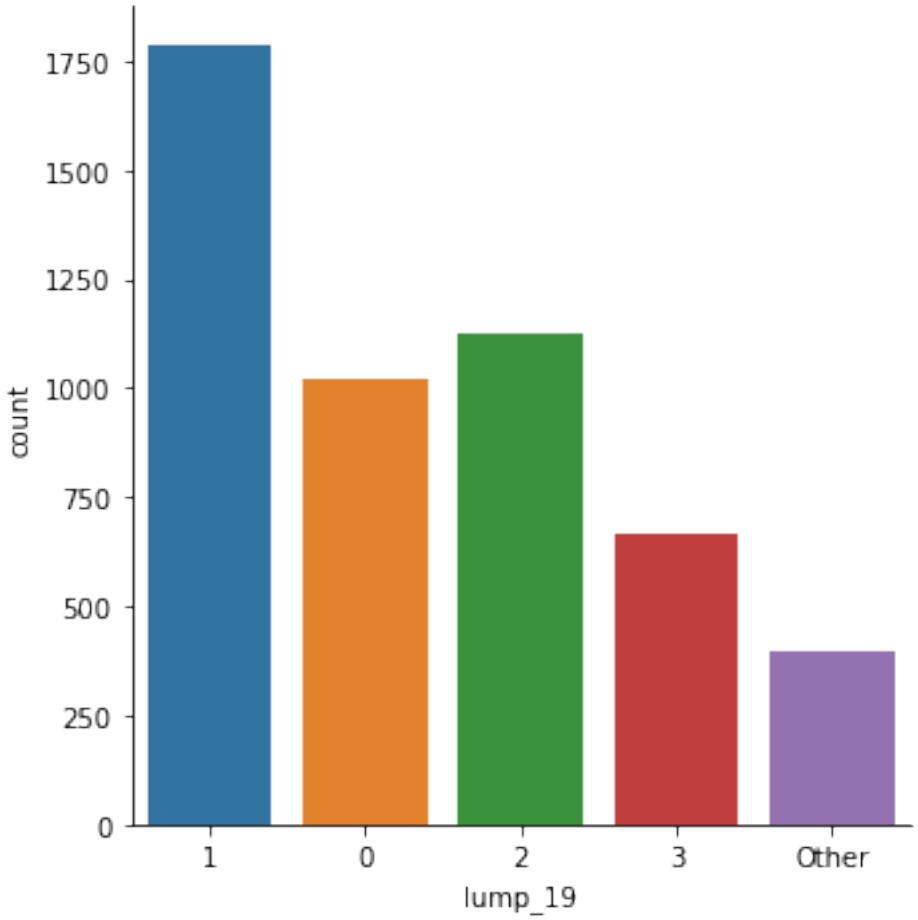
```
df_c['bin_17'] = pd.qcut(df_c.X17, q=4)
sns.catplot(data=df_c, x='bin_17', kind='count')
plt.show()
```



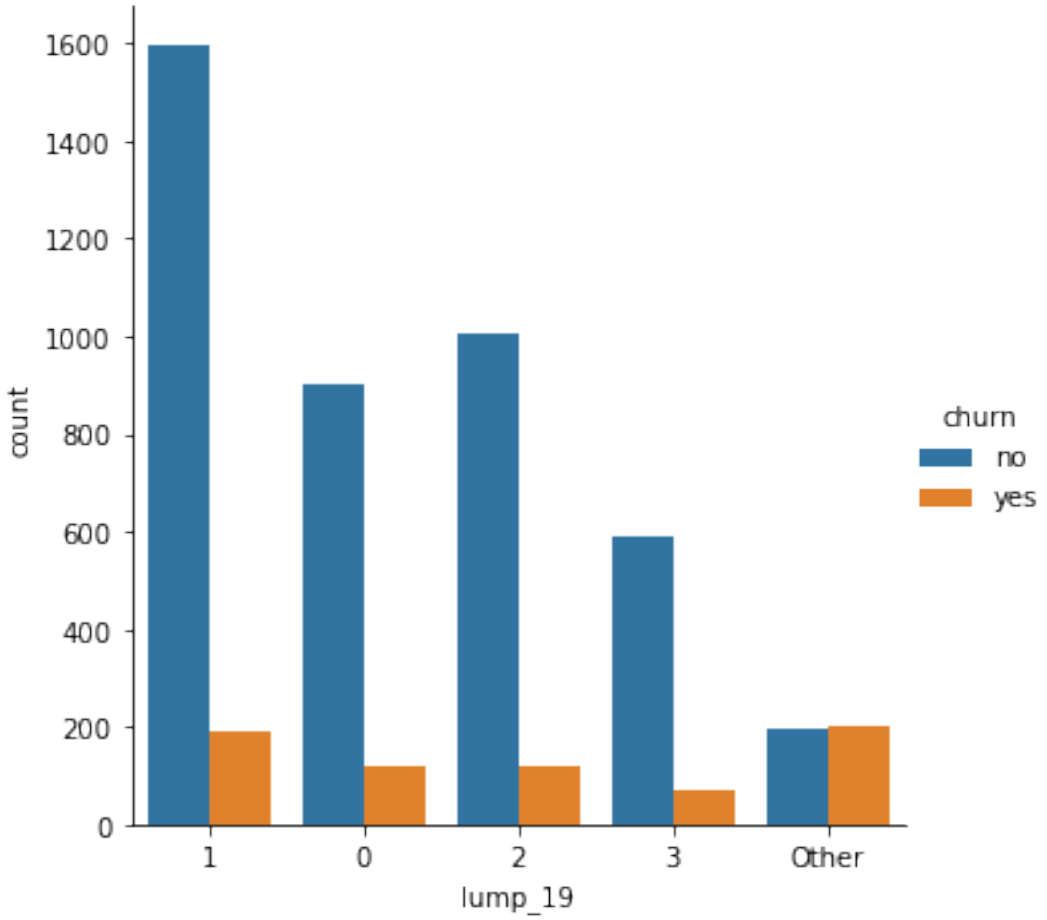

```
sns.catplot(data=df_c, x='bin_17', hue='churn', kind='count')
plt.show()
```




```
df_c['lump_19'] = np.where(df_c.X19 > 3, 'Other',
df_c.X19.astype('str'))

sns.catplot(data=df_c, x='lump_19', kind='count')

plt.show()
```




```
sns.catplot(data=df_c, x='lump_19', hue='churn', kind='count')

plt.show()
```



For continuous inputs

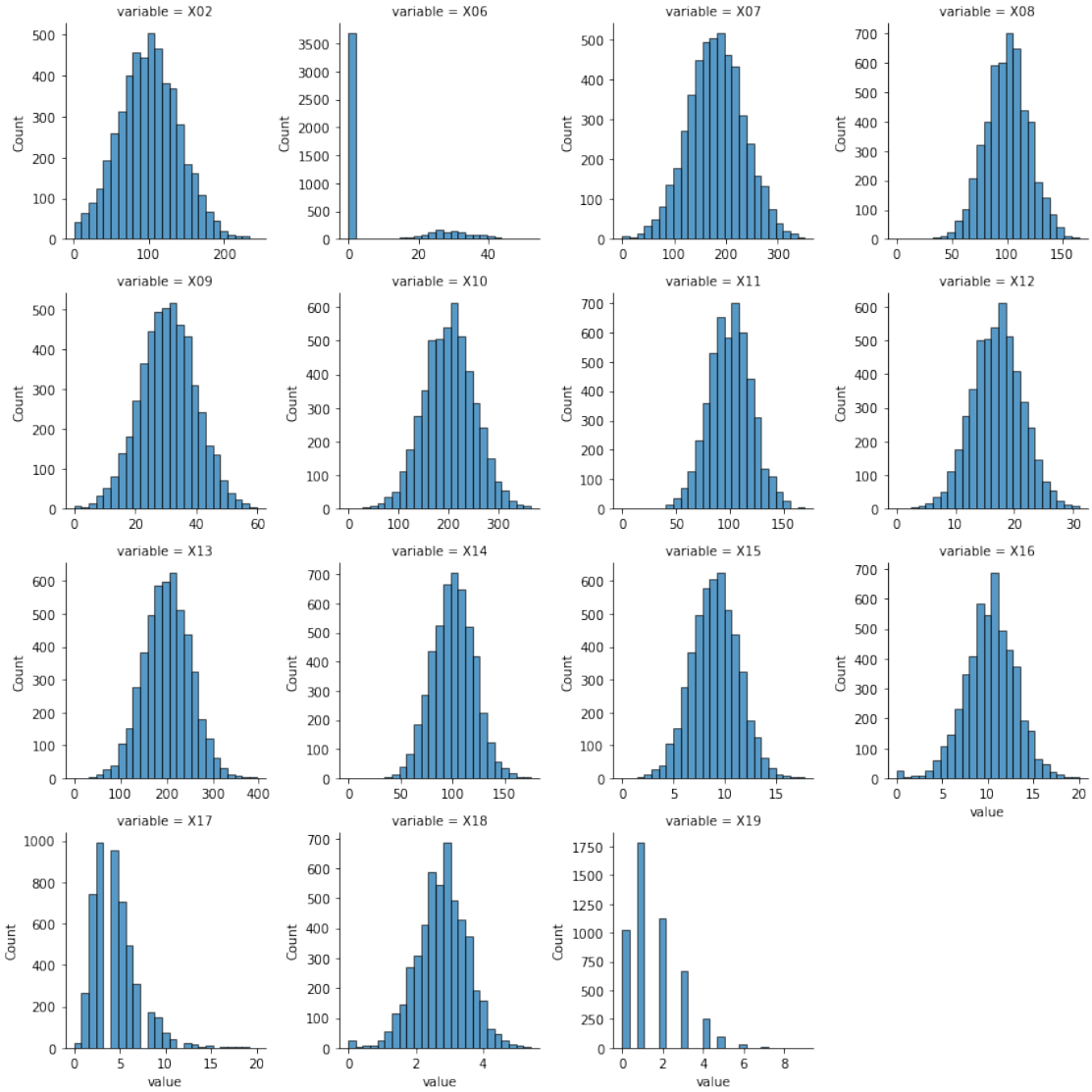
```
num_inputs = df.select_dtypes('number').copy().columns.to_list()

num_inputs
```

```
['X02',
 'X06',
 'X07',
 'X08',
 'X09',
 'X10',
 'X11',
 'X12',
 'X13',
 'X14',
 'X15',
 'X16',
 'X17',
 'X18',
 'X19']
```

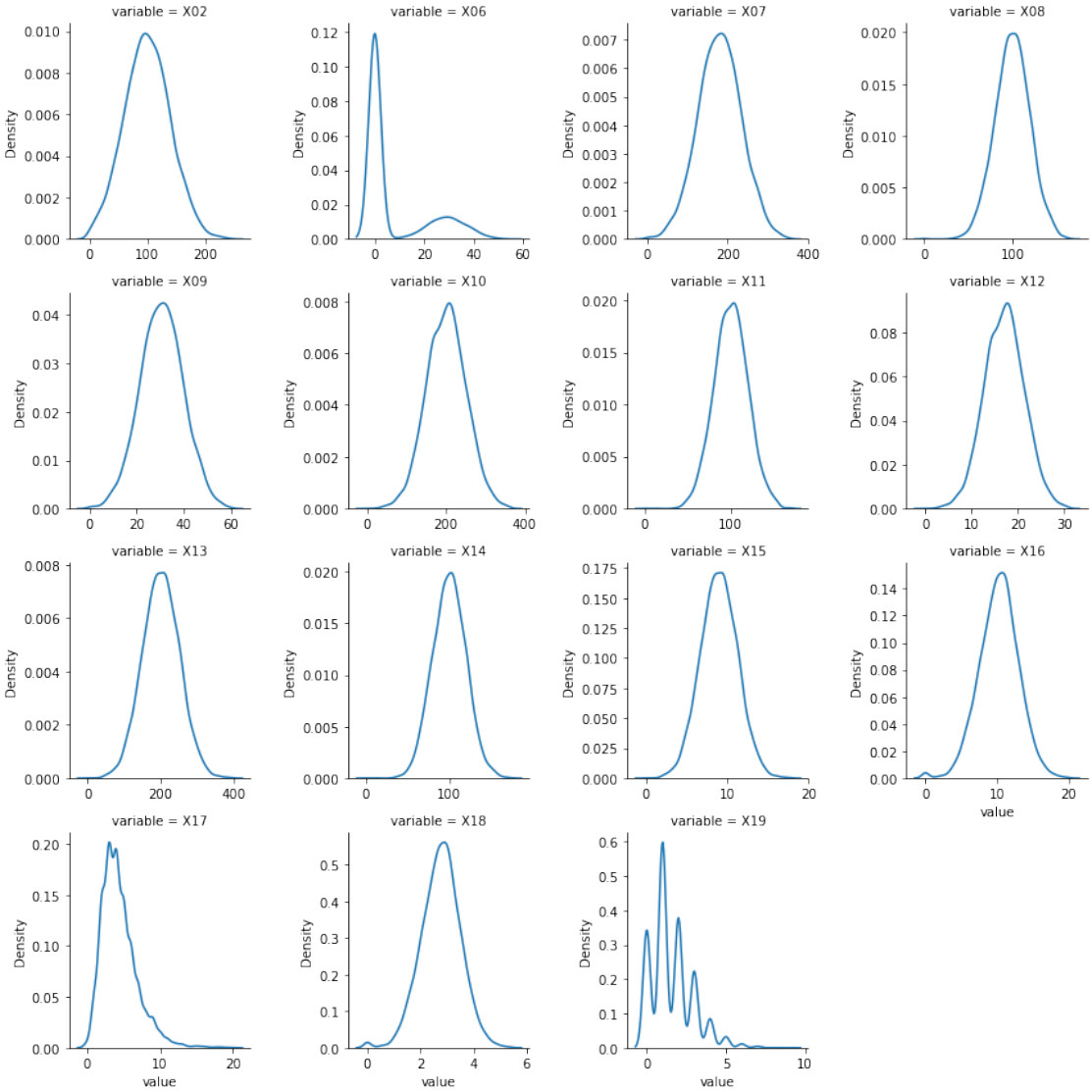
```
lf_num = df.melt(id_vars=['churn'], value_vars = num_inputs,
ignore_index=True)

lf_num
```

```
      churn variable  value
0        no      X02  128.0
1        no      X02  107.0
2        no      X02  137.0
3        no      X02   84.0
4        no      X02   75.0
...     ...      ...    ...
74995    no      X19    2.0
74996   yes      X19    3.0
74997    no      X19    1.0
74998    no      X19    0.0
74999    no      X19    0.0

[75000 rows x 3 columns]
```

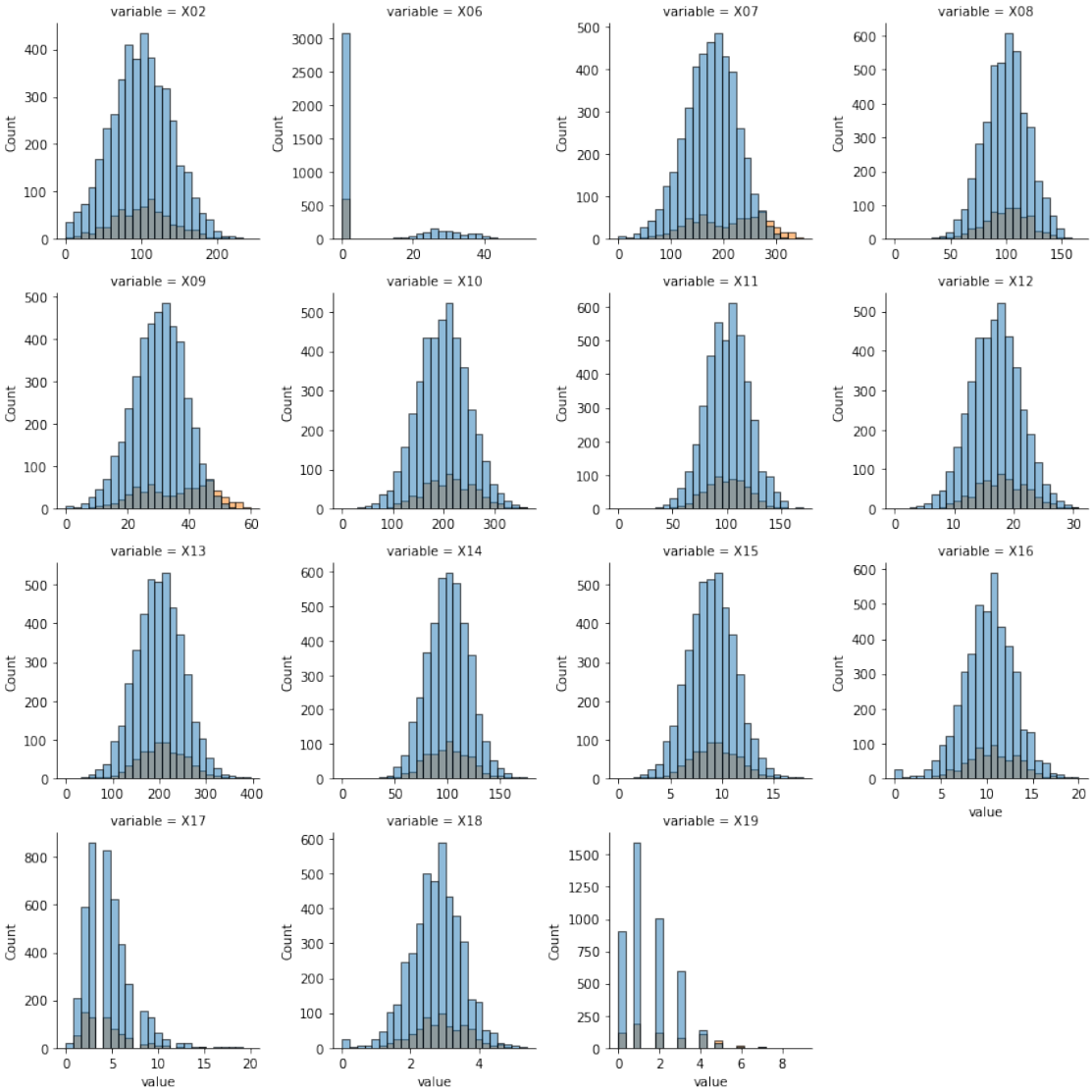
```
g = sns.FacetGrid(data= lf_num, col='variable', col_wrap=4,
sharex=False, sharey=False)

g.map_dataframe(sns.histplot, x='value', bins=25)

plt.show()
```



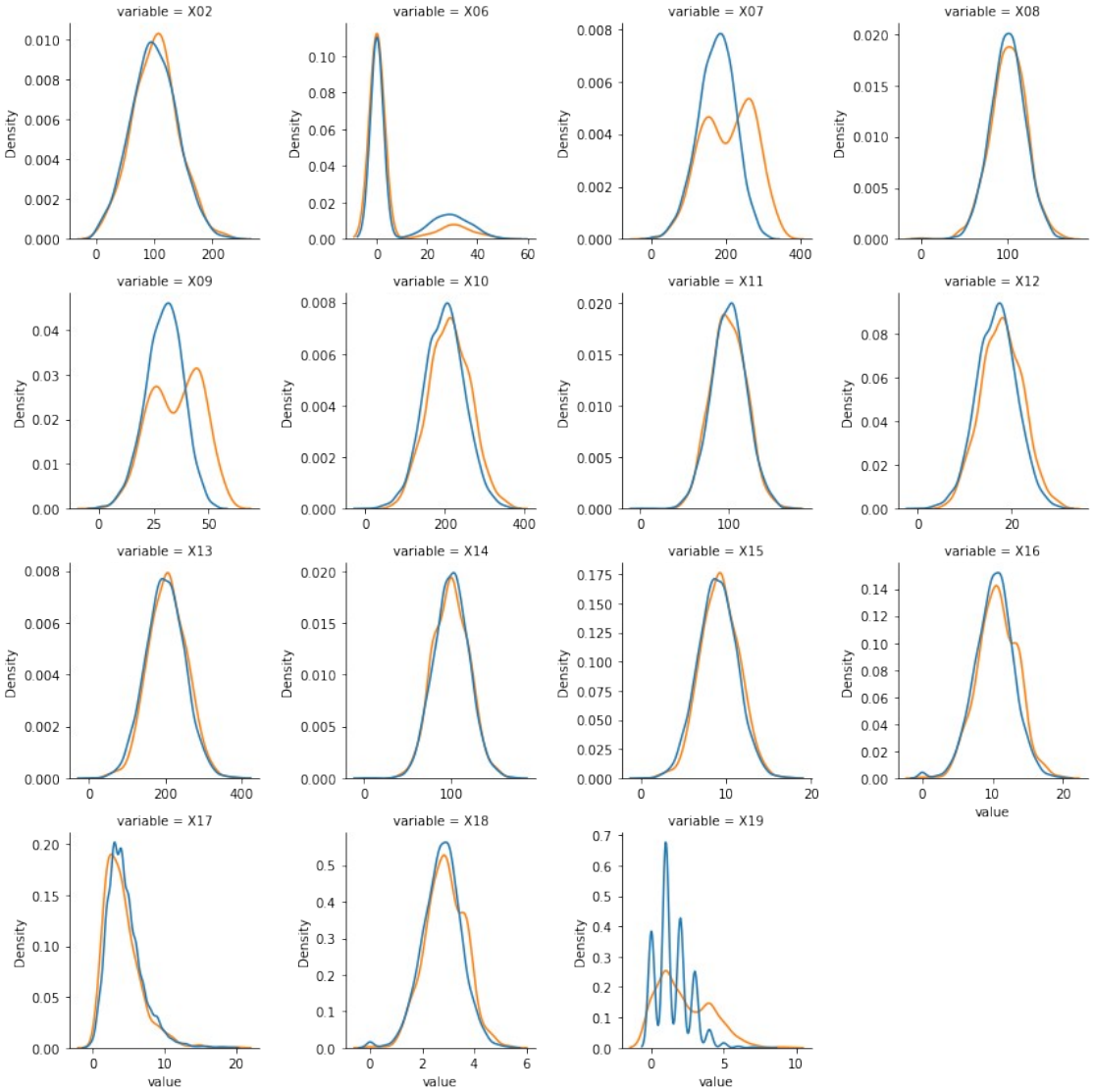

```
g = sns.FacetGrid(data= lf_num, col='variable', col_wrap=4,
sharex=False, sharey=False)

g.map_dataframe(sns.kdeplot, x='value', common_norm=False)

plt.show()
```



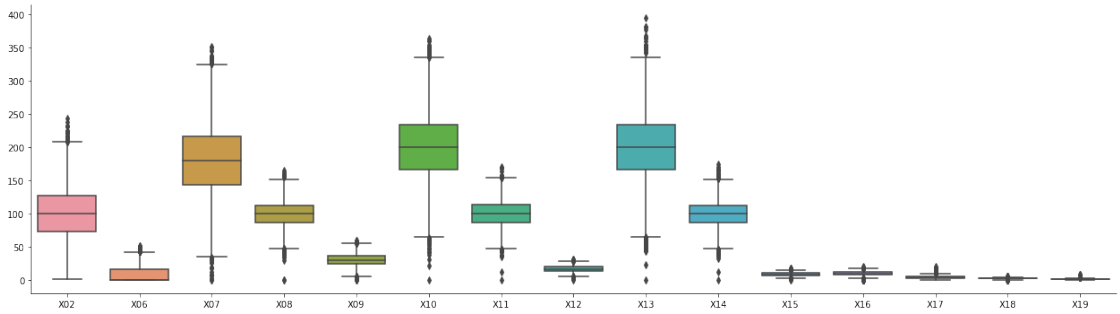

```
g = sns.FacetGrid(data= lf_num, col='variable', col_wrap=4,
sharex=False, sharey=False)

g.map_dataframe(sns.histplot, x='value', hue='churn', bins=25)

plt.show()
```




```
g = sns.FacetGrid(data= lf_num, col='variable', col_wrap=4,
sharex=False, sharey=False)

g.map_dataframe(sns.kdeplot, x='value',
hue='churn' ,common_norm=False)

plt.show()
```

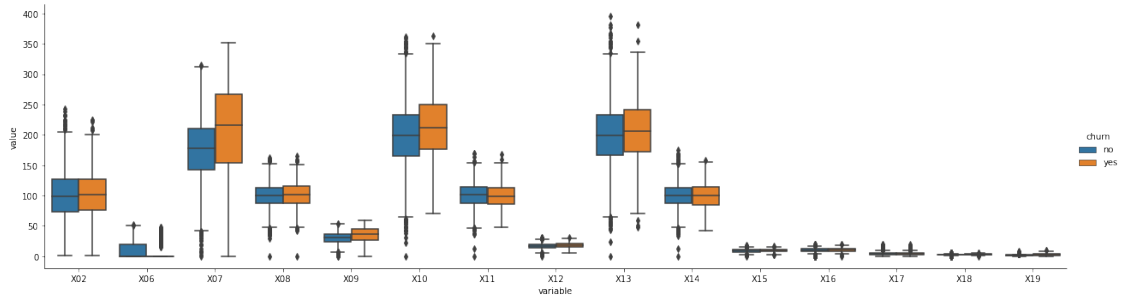



```
sns.catplot(data = df, kind='box', aspect=3.5)

plt.show()
```




```
sns.catplot(data = lf_num, x='variable', y='value', hue='churn',
kind='box', aspect=3.5)
```

```
plt.show()
```



The X06 variable is right-skewed so we will transform this columns by taking the log of the values.

```
df_copy = df.copy()
```

```
df_copy['log X06'] = np.log10(df['X06'])
```

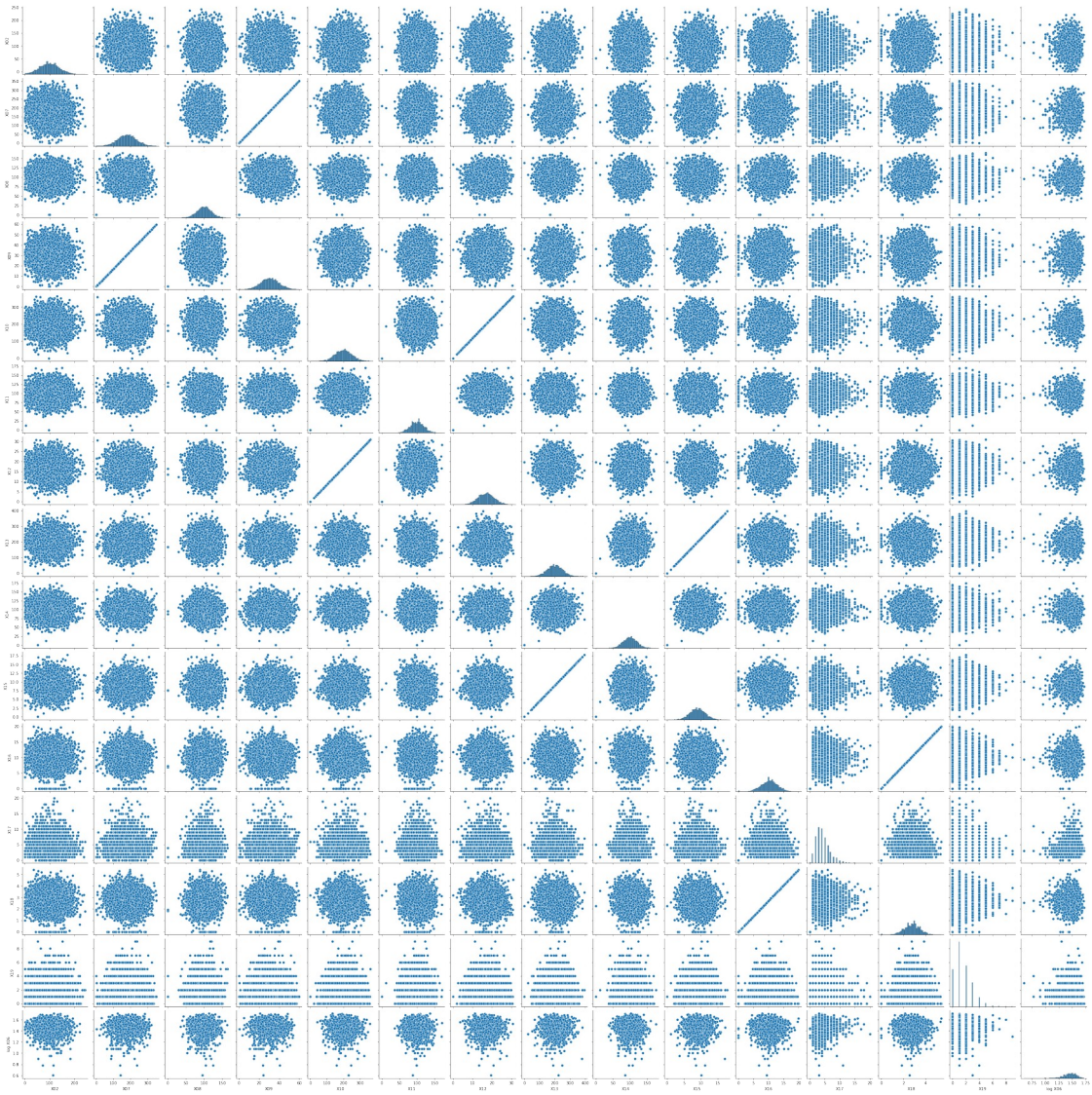
```
C:\Users\Vedant\anaconda3\envs\cmpinf2100\lib\site-packages\pandas\
core\arraylike.py:364: RuntimeWarning: divide by zero encountered in
log10
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

```
df_copy.head()
```

```
  state  X02 X03 X04 X05  X06    X07  X08    X09    X10  ...    X12
X13  \
0    KS  128  AA  Z1  V1   25  265.1  110  45.07  197.4  ...  16.78
244.7
1    OH  107  AA  Z1  V1   26  161.6  123  27.47  195.5  ...  16.62
254.4
2    NJ  137  AA  Z1  V2    0  243.4  114  41.38  121.2  ...  10.30
162.6
3    OH   84  BB  Z2  V2    0  299.4   71  50.90   61.9  ...   5.26
196.9
4    OK   75  AA  Z2  V2    0  166.7  113  28.34  148.3  ...  12.61
186.9

   X14    X15   X16  X17   X18  X19  churn   log X06
0   91  11.01  10.0    3  2.70    1     no  1.397940
1  103  11.45  13.7    3  3.70    1     no  1.414973
2  104   7.32  12.2    5  3.29    0     no      -inf
3   89   8.86   6.6    7  1.78    2     no      -inf
4  121   8.41  10.1    3  2.73    3     no      -inf

[5 rows x 21 columns]
```

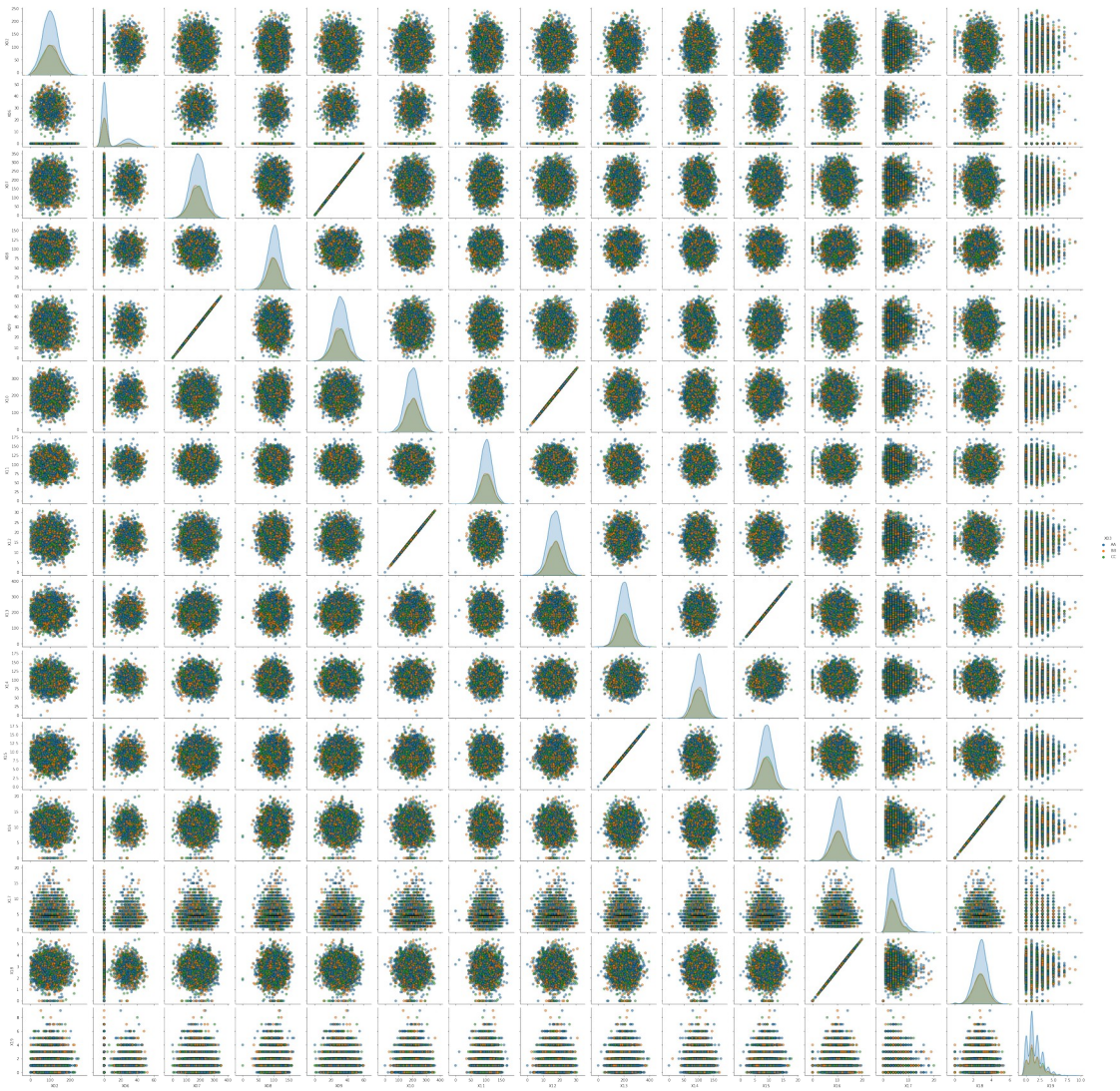
```
df_copy = df_copy.drop(columns=['X06'])
```

```
sns.pairplot(data=df_copy)

plt.show()
```

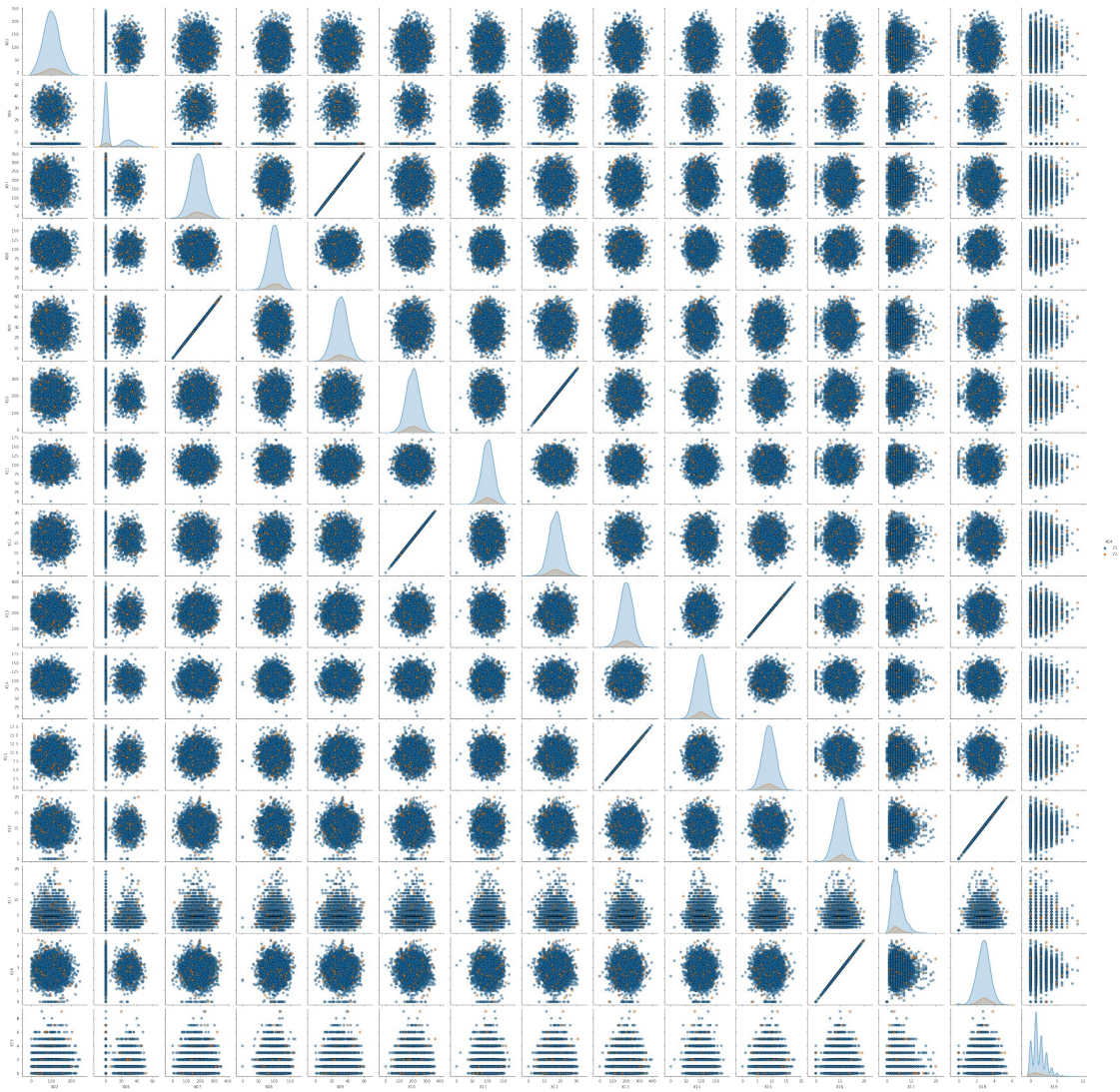


Summaries of the continuous variables grouped by categorical variables

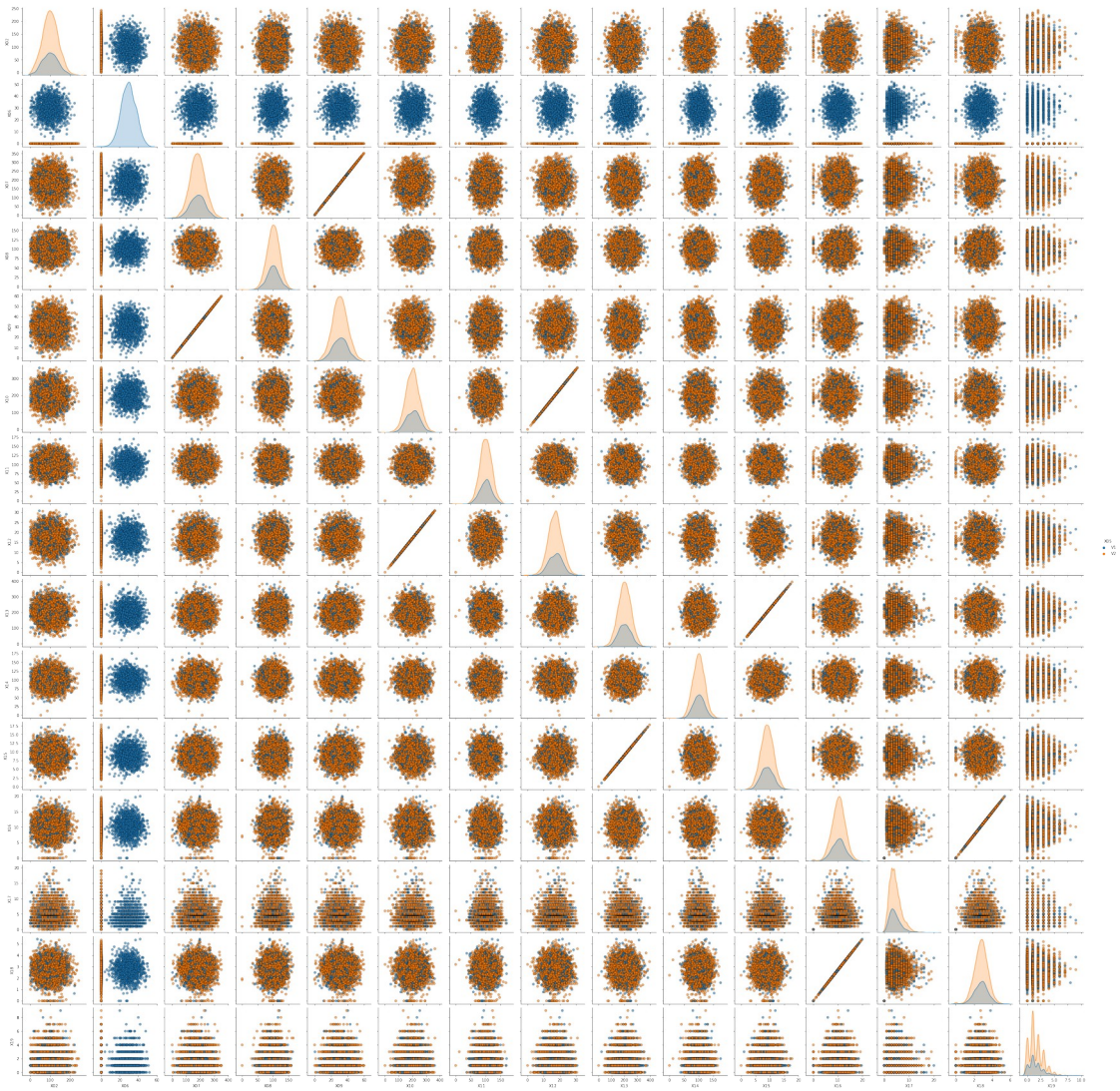
```
sns.pairplot(data=df, hue='X03', diag_kind = 'kde', plot_kws =
{'alpha':0.6, 's':30, 'edgecolor':'k'})

plt.show()
```



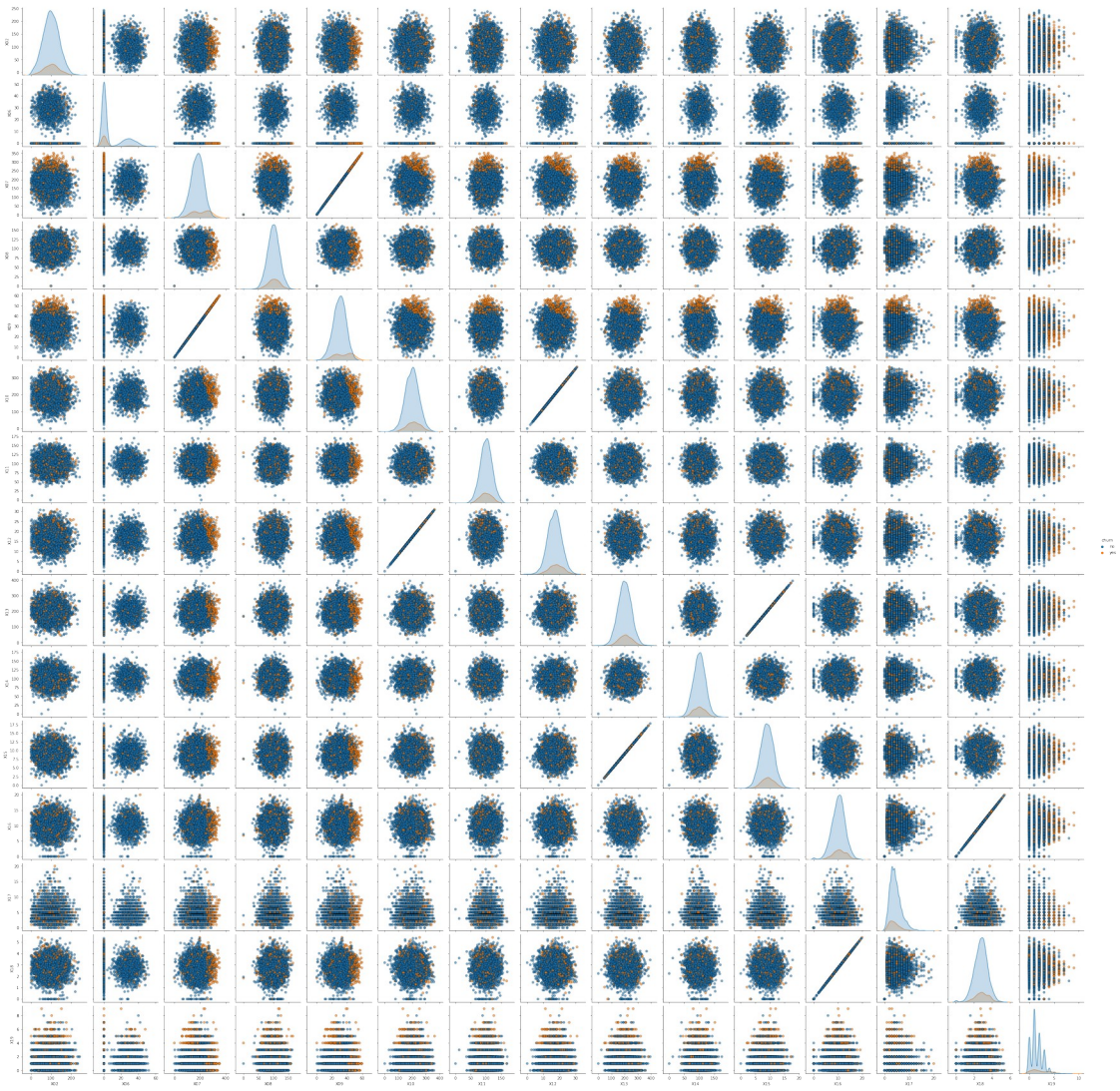
```
sns.pairplot(data=df, hue='X04', diag_kind = 'kde', plot_kws =
{'alpha':0.6, 's':30, 'edgecolor':'k'})

plt.show()
```



```
sns.pairplot(data=df, hue='X05', diag_kind = 'kde', plot_kws =
{'alpha':0.6, 's':30, 'edgecolor':'k'})

plt.show()
```

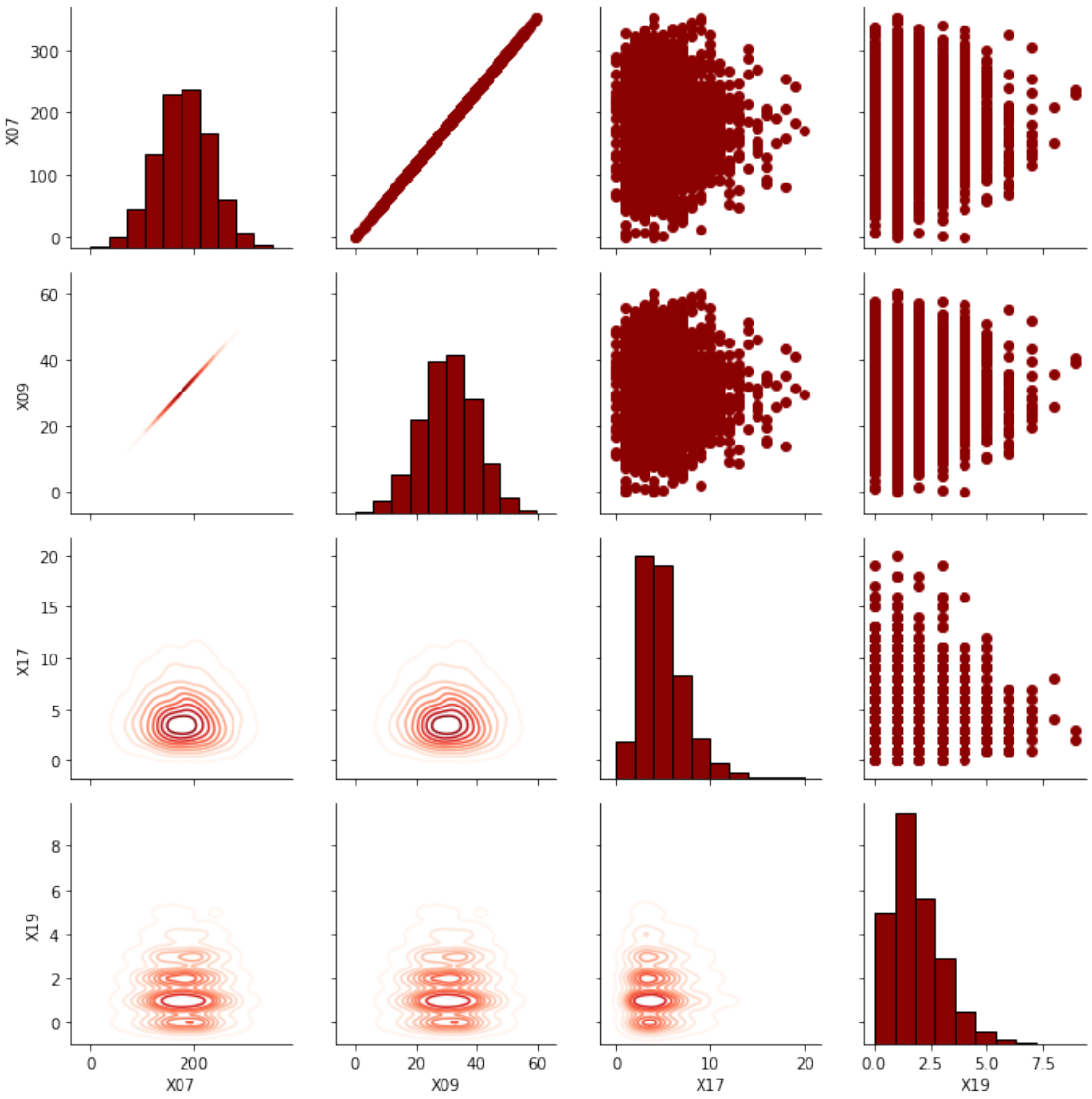


```
sns.pairplot(data=df, hue='churn', diag_kind = 'kde', plot_kws =
{'alpha':0.6, 's':30, 'edgecolor':'k'})

plt.show()
```



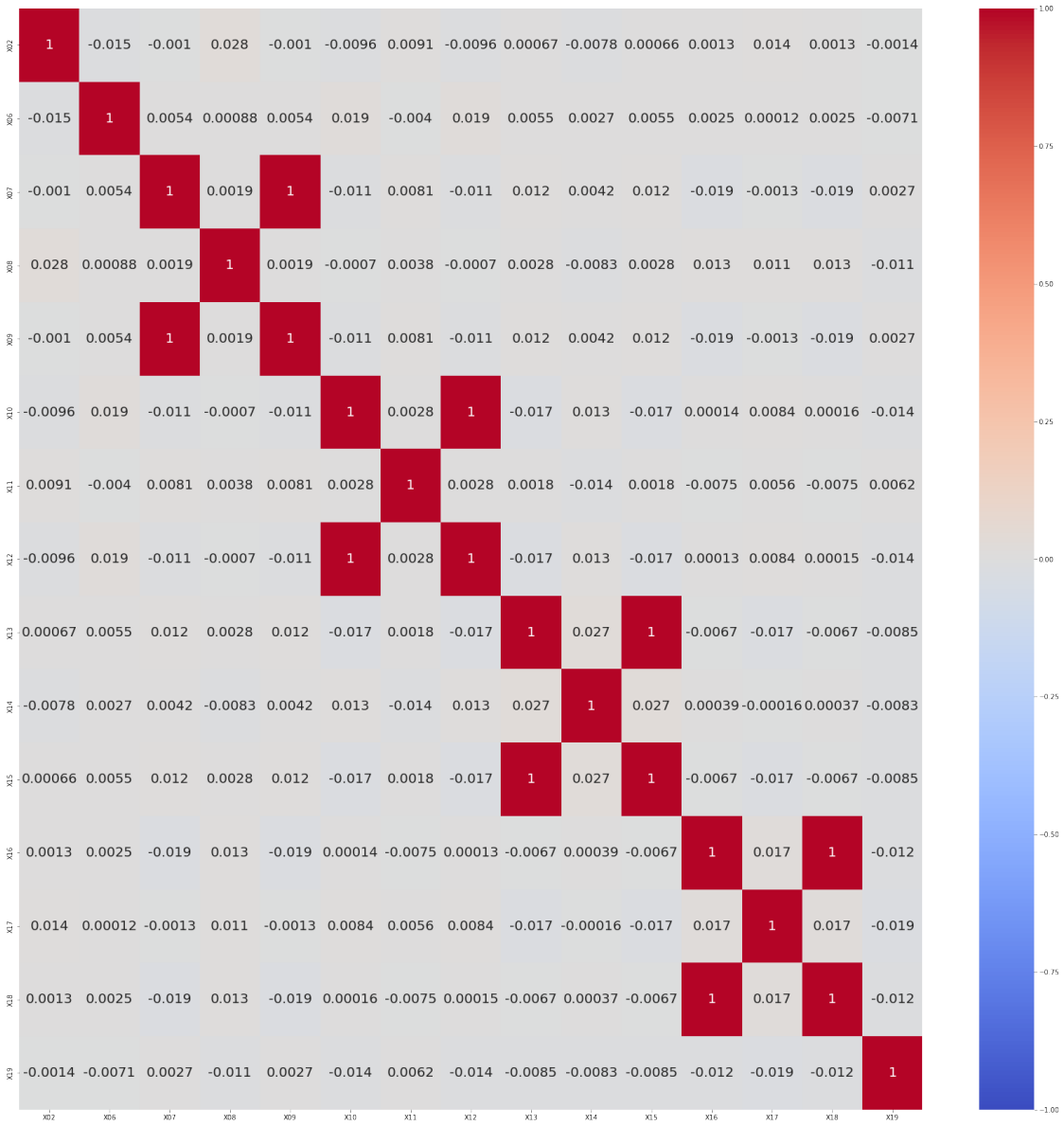
In X07 and x09 we can distinguish that the no responses are on the right side

```
# Create an instance of the PairGrid class.
grid = sns.PairGrid(data=df,
                    vars = ['X07', 'X09','X17','X19'])

# Map a scatter plot to the upper triangle
grid = grid.map_upper(plt.scatter, color = 'darkred')

# Map a histogram to the diagonal
grid = grid.map_diag(plt.hist, bins = 10, color = 'darkred',
                     edgecolor = 'k')
# Map a density plot to the lower triangle
grid = grid.map_lower(sns.kdeplot, cmap = 'Reds')
```



Correlation

```
fig, ax = plt.subplots(figsize=(30,30))

sns.heatmap(data = df.corr(),
            vmin=-1, vmax=1, center=0,
            cmap='coolwarm',
            annot=True, annot_kws={'size': 20},
            ax=ax)

plt.show()
```



From the above figure we can observe that

- X07 and X09
- X10 and X12
- X13 and X15
- X16 and X18

are in perfect correlation. That is they move in same direction together.

If we treat X6 as a categorical input

```
df_c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 22 columns):
```

```
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   state    5000 non-null   object
 1   X02      5000 non-null   int64
 2   X03      5000 non-null   object
 3   X04      5000 non-null   object
 4   X05      5000 non-null   object
 5   X06      5000 non-null   int64
 6   X07      5000 non-null   float64
 7   X08      5000 non-null   int64
 8   X09      5000 non-null   float64
 9   X10      5000 non-null   float64
 10  X11      5000 non-null   int64
 11  X12      5000 non-null   float64
 12  X13      5000 non-null   float64
 13  X14      5000 non-null   int64
 14  X15      5000 non-null   float64
 15  X16      5000 non-null   float64
 16  X17      5000 non-null   int64
 17  X18      5000 non-null   float64
 18  X19      5000 non-null   int64
 19  churn    5000 non-null   object
 20  bin_17   5000 non-null   category
 21  lump_19  5000 non-null   object
dtypes: category(1), float64(8), int64(7), object(6)
memory usage: 825.6+ KB
```

```
df_c['X06_cat'] = np.where(df['X06'] > 0, 1, 0)
```

```
df_c.head()
```

```
  state  X02 X03 X04 X05  X06    X07  X08    X09    X10  ...  X14
X15  \
0    KS  128  AA  Z1  V1   25  265.1  110  45.07  197.4  ...   91
11.01
1    OH  107  AA  Z1  V1   26  161.6  123  27.47  195.5  ...  103
11.45
2    NJ  137  AA  Z1  V2    0  243.4  114  41.38  121.2  ...  104
7.32
3    OH   84  BB  Z2  V2    0  299.4   71  50.90   61.9  ...   89
8.86
4    OK   75  AA  Z2  V2    0  166.7  113  28.34  148.3  ...  121
8.41

    X16  X17   X18  X19  churn         bin_17  lump_19 X06_cat
0  10.0    3  2.70    1     no  (-0.001, 3.0]        1       1
1  13.7    3  3.70    1     no  (-0.001, 3.0]        1       1
2  12.2    5  3.29    0     no     (4.0, 6.0]        0       0
3   6.6    7  1.78    2     no    (6.0, 20.0]        2       0
4  10.1    3  2.73    3     no  (-0.001, 3.0]        3       0
```

```
[5 rows x 23 columns]
```

```
df_c.X06_cat.nunique()
```

```
2
```

```
df_c.X06_cat.value_counts()
```

```
0    3678
1    1322
Name: X06_cat, dtype: int64
```

```
sns.catplot(data=df_c, x='X06_cat', kind='count')
```

```
plt.show()
```