Home Credit default prediction data preprocessing

This module is to do exploratory data analysis and preprocess train and test dataset for home credit risk

This notebook have analysis for application_train.csv and bureau.csv

Analysing application_train.csv

In [1]:
```
import datetime
print(datetime.datetime.now())
```

```
2020-02-26 02:42:05.874705
```

In [2]:
```
import sklearn
print('The scikit-learn version is {}.'.format(sklearn.__version__))
```

```
The scikit-learn version is 0.22.
```

In [3]:
```
import pandas as pd
import numpy as np

import matplotlib as mpl
import matplotlib.pyplot as plt

import sklearn.metrics

# Imputers to deal with Missing Data

from sklearn.impute import SimpleImputer
```

In [4]:
```
df_init = pd.read_csv(r"C:\Users\mamta\MMAI 2020\MMAI823_AI in Finance\Team Assignments\Team Project\Home Credit Risk Dataset\application_train.csv",sep=',')

list(df_init)
df_init.describe().transpose()
```

Out[4]:

	count	mean	std	min	25%
SK_ID_CURR	307511.0	278180.518577	102790.175348	100002.0	189145.5
TARGET	307511.0	0.080729	0.272419	0.0	0.0
CNT_CHILDREN	307511.0	0.417052	0.722121	0.0	0.0
AMT_INCOME_TOTAL	307511.0	168797.919297	237123.146279	25650.0	112500.0
AMT_CREDIT	307511.0	599025.999706	402490.776996	45000.0	270000.0
...	...	...	...	...	...
AMT_REQ_CREDIT_BUREAU_DAY	265992.0	0.007000	0.110757	0.0	0.0
AMT_REQ_CREDIT_BUREAU_WEEK	265992.0	0.034362	0.204685	0.0	0.0
AMT_REQ_CREDIT_BUREAU_MON	265992.0	0.267395	0.916002	0.0	0.0
AMT_REQ_CREDIT_BUREAU_QRT	265992.0	0.265474	0.794056	0.0	0.0
AMT_REQ_CREDIT_BUREAU_YEAR	265992.0	1.899974	1.869295	0.0	0.0

106 rows × 8 columns

Get number of missing values for each column

In [5]:
```
df_init.isnull().sum()
```

Out[5]:
```
SK_ID_CURR                      0
TARGET                          0
NAME_CONTRACT_TYPE              0
CODE_GENDER                     0
FLAG_OWN_CAR                    0
                               ...
AMT_REQ_CREDIT_BUREAU_DAY     41519
AMT_REQ_CREDIT_BUREAU_WEEK    41519
AMT_REQ_CREDIT_BUREAU_MON     41519
AMT_REQ_CREDIT_BUREAU_QRT     41519
AMT_REQ_CREDIT_BUREAU_YEAR    41519
Length: 122, dtype: int64
```

This shows the details of all the columns with missing value including count and percentage of missing values

In [6]:
```
def missing_zero_values_table(df_init):
        zero_val = (df_init == 0.00).astype(int).sum(axis=0)
        mis_val = df_init.isnull().sum()
        mis_val_percent = 100 * df_init.isnull().sum() / len(df_init)
        mz_table = pd.concat([zero_val, mis_val, mis_val_percent], axis=1)
        mz_table = mz_table.rename(
        columns = {0 : 'Zero Values', 1 : 'Missing Values', 2 : '% of Total Va
lues'})
        mz_table['Total Zero Missing Values'] = mz_table['Zero Values'] + mz_t
able['Missing Values']
        mz_table['% Total Zero Missing Values'] = 100 * mz_table['Total Zero M
issing Values'] / len(df_init)
        mz_table['Data Type'] = df_init.dtypes
        mz_table = mz_table[
            mz_table.iloc[:,1] != 0].sort_values(
        '% of Total Values', ascending=False).round(1)
        print ("Your selected dataframe has " + str(df_init.shape[1]) + " colu
mns and " + str(df_init.shape[0]) + " Rows.\n"
            "There are " + str(mz_table.shape[0]) +
              " columns that have missing values.")
#         mz_table.to_excel('missing_and_zero_values.xlsx', freeze_panes=(1,
0), index = False)
        return mz_table

missing_zero_values_table(df_init)
```

```
Your selected dataframe has 122 columns and 307511 Rows.
There are 67 columns that have missing values.
```

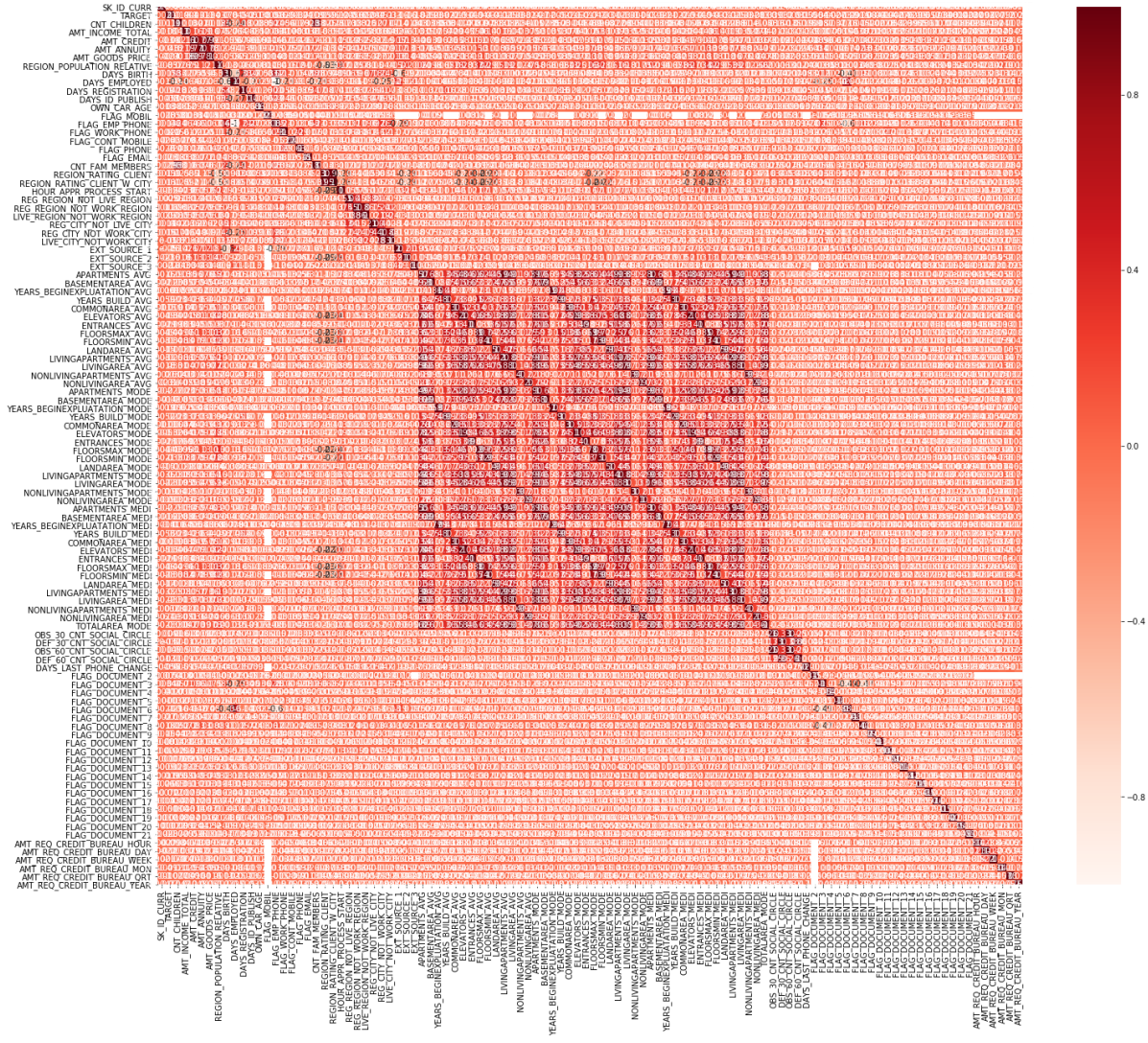
Out[6]:

	Zero Values	Missing Values	% of Total Values	Total Zero Missing Values	% Total Zero Missing Values	Data Type
COMMONAREA_MEDI	8691	214865	69.9	223556	72.7	float64
COMMONAREA_AVG	8442	214865	69.9	223307	72.6	float64
COMMONAREA_MODE	9690	214865	69.9	224555	73.0	float64
NONLIVINGAPARTMENTS_MEDI	56097	213514	69.4	269611	87.7	float64
NONLIVINGAPARTMENTS_MODE	59255	213514	69.4	272769	88.7	float64
...	...	...	...	...	...	...
EXT_SOURCE_2	0	660	0.2	660	0.2	float64
AMT_GOODS_PRICE	0	278	0.1	278	0.1	float64
AMT_ANNUITY	0	12	0.0	12	0.0	float64
CNT_FAM_MEMBERS	0	2	0.0	2	0.0	float64
DAYS_LAST_PHONE_CHANGE	37672	1	0.0	37673	12.3	float64

67 rows × 6 columns

In [7]:

```
import seaborn as sns

mpl.rcParams['interactive'] == True
#Using Pearson Correlation
plt.figure(figsize=(24,20))
cor = df_init.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.savefig('Correlation_Pearson')
plt.show()
```



In [8]:
```
# Find correlations with the target and sort
correlations = df_init.corr()['TARGET'].sort_values()

# Display correlations
print('Most Positive Correlations:\n', correlations.tail(15))
print('\nMost Negative Correlations:\n', correlations.head(15))
```

```
Most Positive Correlations:
 DEF_60_CNT_SOCIAL_CIRCLE       0.031276
DEF_30_CNT_SOCIAL_CIRCLE        0.032248
LIVE_CITY_NOT_WORK_CITY         0.032518
OWN_CAR_AGE                     0.037612
DAYS_REGISTRATION               0.041975
FLAG_DOCUMENT_3                 0.044346
REG_CITY_NOT_LIVE_CITY          0.044395
FLAG_EMP_PHONE                  0.045982
REG_CITY_NOT_WORK_CITY          0.050994
DAYS_ID_PUBLISH                 0.051457
DAYS_LAST_PHONE_CHANGE          0.055218
REGION_RATING_CLIENT            0.058899
REGION_RATING_CLIENT_W_CITY     0.060893
DAYS_BIRTH                      0.078239
TARGET                          1.000000
Name: TARGET, dtype: float64

Most Negative Correlations:
 EXT_SOURCE_3                 -0.178919
EXT_SOURCE_2                 -0.160472
EXT_SOURCE_1                 -0.155317
DAYS_EMPLOYED                -0.044932
FLOORSMAX_AVG                -0.044003
FLOORSMAX_MEDI               -0.043768
FLOORSMAX_MODE               -0.043226
AMT_GOODS_PRICE              -0.039645
REGION_POPULATION_RELATIVE   -0.037227
ELEVATORS_AVG                -0.034199
ELEVATORS_MEDI               -0.033863
FLOORSMIN_AVG                -0.033614
FLOORSMIN_MEDI               -0.033394
LIVINGAREA_AVG               -0.032997
LIVINGAREA_MEDI              -0.032739
Name: TARGET, dtype: float64
```

Convert number of days into years and make it positive

Drop the days columns

In [9]:
```
df_init['DAYS_BIRTH_YEAR'] =  round((df_init['DAYS_BIRTH'] / -365),2 )
df_init['DAYS_EMPLOYED_YEAR'] =  round((df_init['DAYS_EMPLOYED'] / -365),2 )
df_init['DAYS_REGISTRATION_YEAR'] = round((df_init['DAYS_REGISTRATION'] / -365
),2 )
df_init['DAYS_ID_PUBLISH_YEAR'] =  round((df_init['DAYS_ID_PUBLISH'] / -365),2
)
```

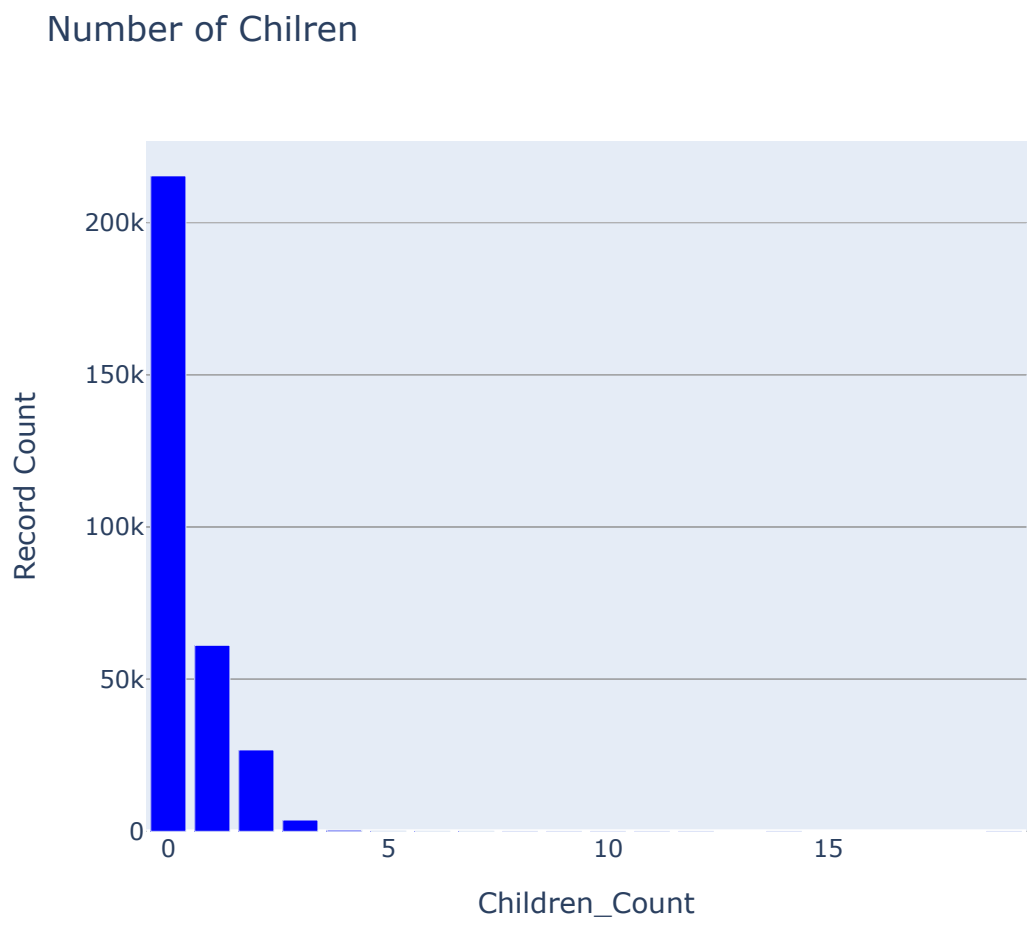
In [10]:
```
df_init = df_init.drop(["DAYS_BIRTH", "DAYS_EMPLOYED", "DAYS_REGISTRATION","DA
YS_ID_PUBLISH"], axis=1)
```

Drop Flags related to documents

In [11]:
```
df_init = df_init.drop(["FLAG_DOCUMENT_2","FLAG_DOCUMENT_3","FLAG_DOCUMENT_4",
"FLAG_DOCUMENT_5","FLAG_DOCUMENT_6","FLAG_DOCUMENT_7",
"FLAG_DOCUMENT_8","FLAG_DOCUMENT_9","FLAG_DOCUMENT_10","FLAG_DOCUMENT_11","FLA
G_DOCUMENT_12","FLAG_DOCUMENT_13","FLAG_DOCUMENT_14",
"FLAG_DOCUMENT_15","FLAG_DOCUMENT_16","FLAG_DOCUMENT_17","FLAG_DOCUMENT_18","F
LAG_DOCUMENT_19","FLAG_DOCUMENT_20","FLAG_DOCUMENT_21"], axis=1)
```

Plot graph for different chilren counts

In [12]:
```
import plotly.graph_objs as go
import plotly.figure_factory as ff
from plotly import tools
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)

temp_children = df_init["CNT_CHILDREN"].value_counts()
df_children = pd.DataFrame({'CNT_CHILDREN': temp_children.index,'values': temp
_children.values})

trace = go.Bar(
    x = df_children['CNT_CHILDREN'],y = df_children['values'],
    name="Number of Children",
    marker=dict(color="Blue"),
    text=df_children['values']
)
data = [trace]
layout = dict(title = 'Number of Chilren',
          xaxis = dict(title = 'Children_Count', showticklabels=True),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='Children_Count')
```



In [13]:
```
df_children
```

Out[13]:

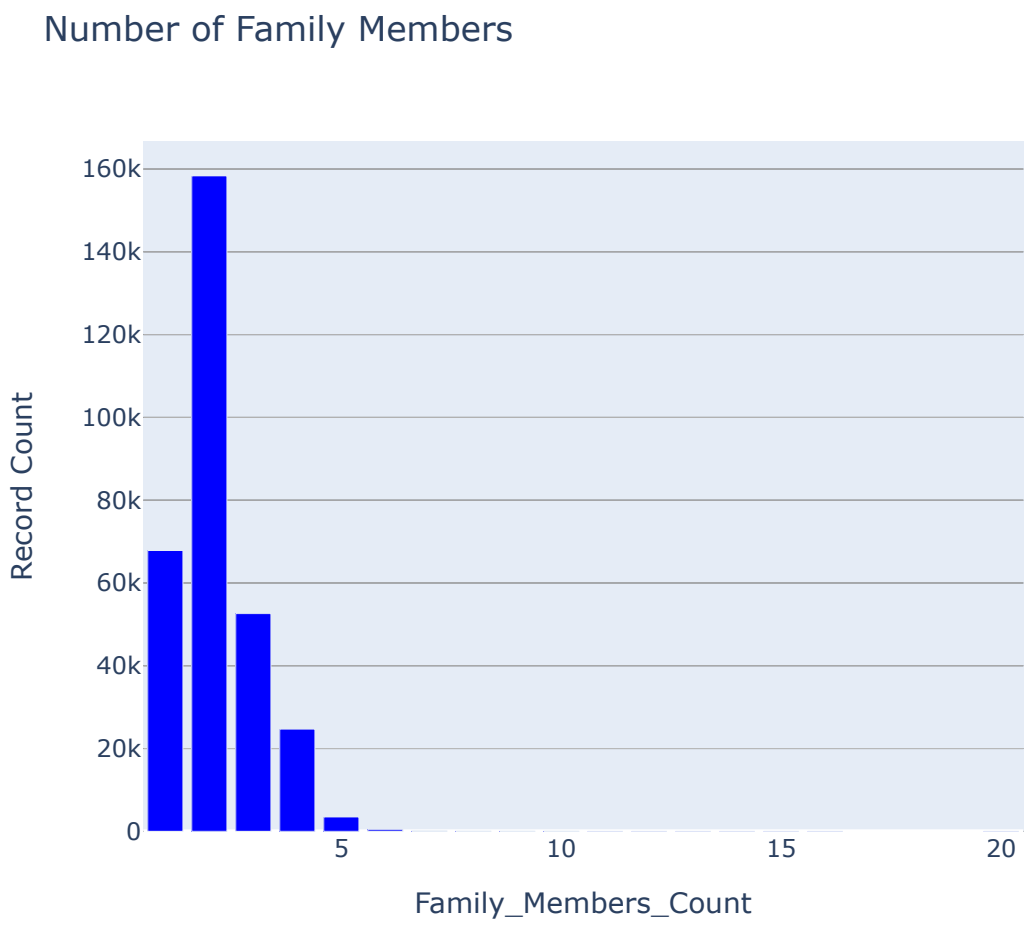
	CNT_CHILDREN	values
0	0	215371
1	1	61119
2	2	26749
3	3	3717
4	4	429
5	5	84
6	6	21
7	7	7
8	14	3
9	19	2
10	12	2
11	10	2
12	9	2
13	8	2
14	11	1

for CNT_CHILDREN => 6 make it 6

In [14]:
```
df_init['CNT_CHILDREN'].values[df_init['CNT_CHILDREN'].values > 5] = 6
```

Plot for CNT_FAM_MEMBERS to see the trend

In [15]:
```
temp_family_members = df_init["CNT_FAM_MEMBERS"].value_counts()
df_family_members = pd.DataFrame({'CNT_FAM_MEMBERS': temp_family_members.index
,'values': temp_family_members.values})

trace = go.Bar(
    x = df_family_members['CNT_FAM_MEMBERS'],y = df_family_members['values'],
    name="Number of Family Members",
    marker=dict(color="Blue"),
    text=df_family_members['values']
)
data = [trace]
layout = dict(title = 'Number of Family Members',
          xaxis = dict(title = 'Family_Members_Count', showticklabels=True),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='Family_Members_Count')
```



In [16]:
```
df_family_members
```

Out[16]:

	CNT_FAM_MEMBERS	values
0	2.0	158357
1	1.0	67847
2	3.0	52601
3	4.0	24697
4	5.0	3478
5	6.0	408
6	7.0	81
7	8.0	20
8	9.0	6
9	10.0	3
10	14.0	2
11	16.0	2
12	12.0	2
13	20.0	2
14	11.0	1
15	13.0	1
16	15.0	1

Check for NULL values in CNT_FAM_MEMBERS and replace it with zeros because there are only two rows with blank

In [17]:
```
np.where(pd.isnull(df_init['CNT_FAM_MEMBERS']))
df_init['CNT_FAM_MEMBERS'].fillna(0, inplace=True)
```

In [18]:
```
df_init['CNT_FAM_MEMBERS'] = df_init['CNT_FAM_MEMBERS'].astype('int',copy=Fals
e)
```

For CNT_FAM_MEMBERS => 7 make it 7

In [19]:
```
df_init['CNT_FAM_MEMBERS'].values[df_init['CNT_FAM_MEMBERS'].values > 6] = 7
```

In [20]:
```
df_init['CNT_FAM_MEMBERS'].unique()
```

Out[20]:
```
array([1, 2, 3, 4, 5, 6, 7, 0], dtype=int64)
```

Visualize REGION_RATING_CLIENT

In [21]:
```
temp_REGION_RATING_CLIENT = df_init["REGION_RATING_CLIENT"].value_counts()
df_REGION_RATING_CLIENT = pd.DataFrame({'REGION_RATING_CLIENT': temp_REGION_RA
TING_CLIENT.index,'values': temp_REGION_RATING_CLIENT.values})

trace = go.Bar(
    x = df_REGION_RATING_CLIENT['REGION_RATING_CLIENT'],y = df_REGION_RATING_C
LIENT['values'],
    name="Number of Region Rating Client",
    marker=dict(color="Blue"),
    text=df_REGION_RATING_CLIENT['values']
)
data = [trace]
layout = dict(title = 'Number of Region Rating Client',
          xaxis = dict(title = 'REGION_RATING_CLIENT', showticklabels=True),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='REGION_RATING_CLIENT')
```

One hot encode REGION_RATING_CLIENT

In [22]:
```
np.where(pd.isnull(df_init['REGION_RATING_CLIENT']))
```

Out[22]:
```
(array([], dtype=int64),)
```

In [23]:
```
df_init['REGION_RATING_CLIENT'].unique()
```

Out[23]:
```
array([2, 1, 3], dtype=int64)
```

In [24]:
```
from sklearn.preprocessing import LabelEncoder
le_REGION_RATING_CLIENT = LabelEncoder()
df_init['REGION_RATING_CLIENT_ENCODED'] = le_REGION_RATING_CLIENT.fit_transfor
m(df_init.REGION_RATING_CLIENT)
```

In [25]:
```
from sklearn.preprocessing import OneHotEncoder
REGION_RATING_CLIENT_ohe = OneHotEncoder()
X = REGION_RATING_CLIENT_ohe.fit_transform(df_init.REGION_RATING_CLIENT_ENCODE
D.values.reshape(-1,1)).toarray()
```

In [26]:
```
dfOneHot = pd.DataFrame(X, columns = ["REGION_RATING_CLIENT_"+str(int(i)) for
i in range(X.shape[1])])
df_init = pd.concat([df_init, dfOneHot], axis=1)
```

In [27]:
```
df_init
```

Out[27]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FL
0	100002	1	Cash loans	M	N	
1	100003	0	Cash loans	F	N	
2	100004	0	Revolving loans	M	Y	
3	100006	0	Cash loans	F	N	
4	100007	0	Cash loans	M	N	
...	...	...	...	...	...	
307506	456251	0	Cash loans	M	N	
307507	456252	0	Cash loans	F	N	
307508	456253	0	Cash loans	F	N	
307509	456254	1	Cash loans	F	N	
307510	456255	0	Cash loans	F	N	

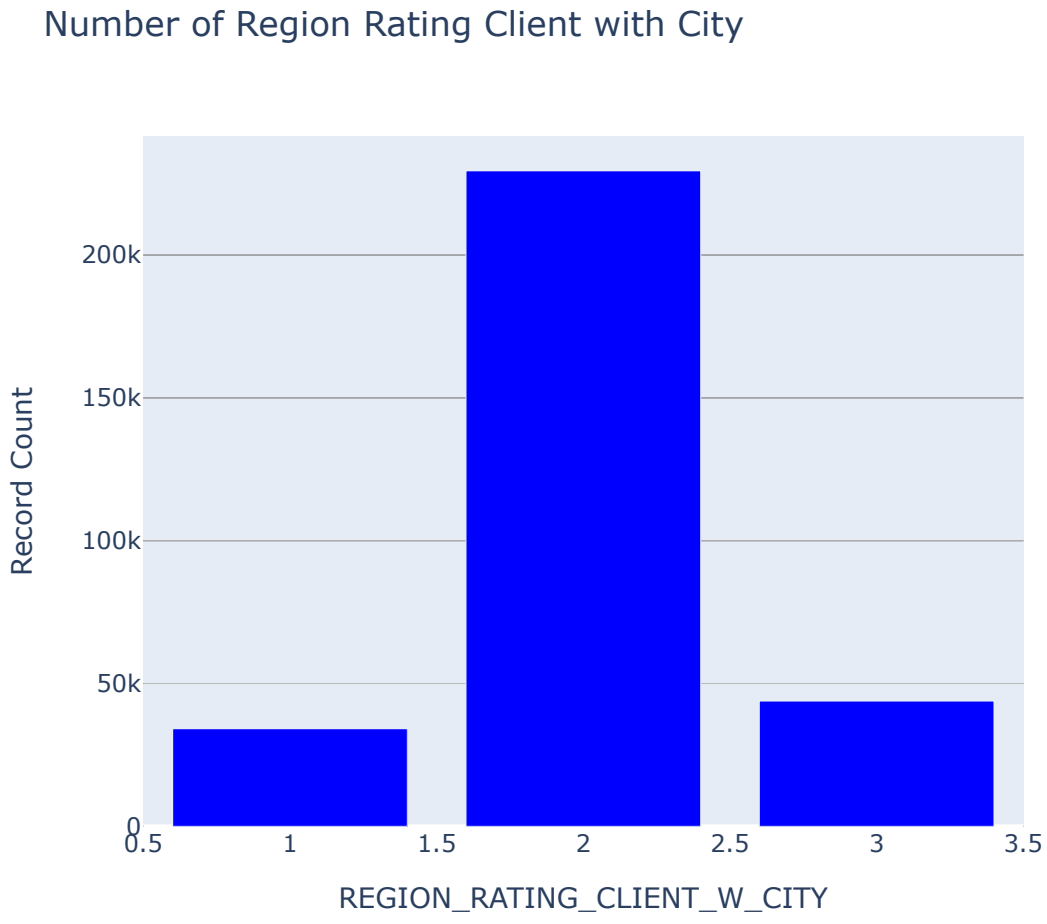
307511 rows × 106 columns

Drop REGION_RATING_CLIENT_ENCODED and REGION_RATING_CLIENT since we don't need it now

In [28]:
```
df_init = df_init.drop(["REGION_RATING_CLIENT_ENCODED", "REGION_RATING_CLIENT"
], axis=1)
```

Visualize REGION_RATING_CLIENT_W_CITY

In [29]:
```
temp_REGION_RATING_CLIENT_W_CITY = df_init["REGION_RATING_CLIENT_W_CITY"].valu
e_counts()
df_REGION_RATING_CLIENT_W_CITY = pd.DataFrame({'REGION_RATING_CLIENT_W_CITY':
temp_REGION_RATING_CLIENT_W_CITY.index,'values': temp_REGION_RATING_CLIENT_W_C
ITY.values})

trace = go.Bar(
    x = df_REGION_RATING_CLIENT_W_CITY['REGION_RATING_CLIENT_W_CITY'],y = df_R
EGION_RATING_CLIENT_W_CITY['values'],
    name="Number of Region Rating Client with City",
    marker=dict(color="Blue"),
    text=df_REGION_RATING_CLIENT_W_CITY['values']
)
data = [trace]
layout = dict(title = 'Number of Region Rating Client with City',
          xaxis = dict(title = 'REGION_RATING_CLIENT_W_CITY', showticklabels=T
rue),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='REGION_RATING_CLIENT_W_CITY')
```



One hot Encoding for REGION_RATING_CLIENT_W_CITY

In [30]:
```
np.where(pd.isnull(df_init['REGION_RATING_CLIENT_W_CITY']))
```

Out[30]:
```
(array([], dtype=int64),)
```

In [31]:
```
le_REGION_RATING_CLIENT_W_CITY = LabelEncoder()
df_init['REGION_RATING_CLIENT_W_CITY_ENCODED'] = le_REGION_RATING_CLIENT_W_CIT
Y.fit_transform(df_init.REGION_RATING_CLIENT_W_CITY)
```

In [32]:
```
REGION_RATING_CLIENT_W_CITY_ohe = OneHotEncoder()
X = REGION_RATING_CLIENT_W_CITY_ohe.fit_transform(df_init.REGION_RATING_CLIENT
_W_CITY_ENCODED.values.reshape(-1,1)).toarray()
```

In [33]:
```
dfOneHot = pd.DataFrame(X, columns = ["REGION_RATING_CLIENT_W_CITY_"+str(int(i
)) for i in range(X.shape[1])])
df_init = pd.concat([df_init, dfOneHot], axis=1)
```

In [34]:
```
df_init
```

Out[34]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FL
0	100002	1	Cash loans	M	N	
1	100003	0	Cash loans	F	N	
2	100004	0	Revolving loans	M	Y	
3	100006	0	Cash loans	F	N	
4	100007	0	Cash loans	M	N	
...	...	...	...	...	...	
307506	456251	0	Cash loans	M	N	
307507	456252	0	Cash loans	F	N	
307508	456253	0	Cash loans	F	N	
307509	456254	1	Cash loans	F	N	
307510	456255	0	Cash loans	F	N	

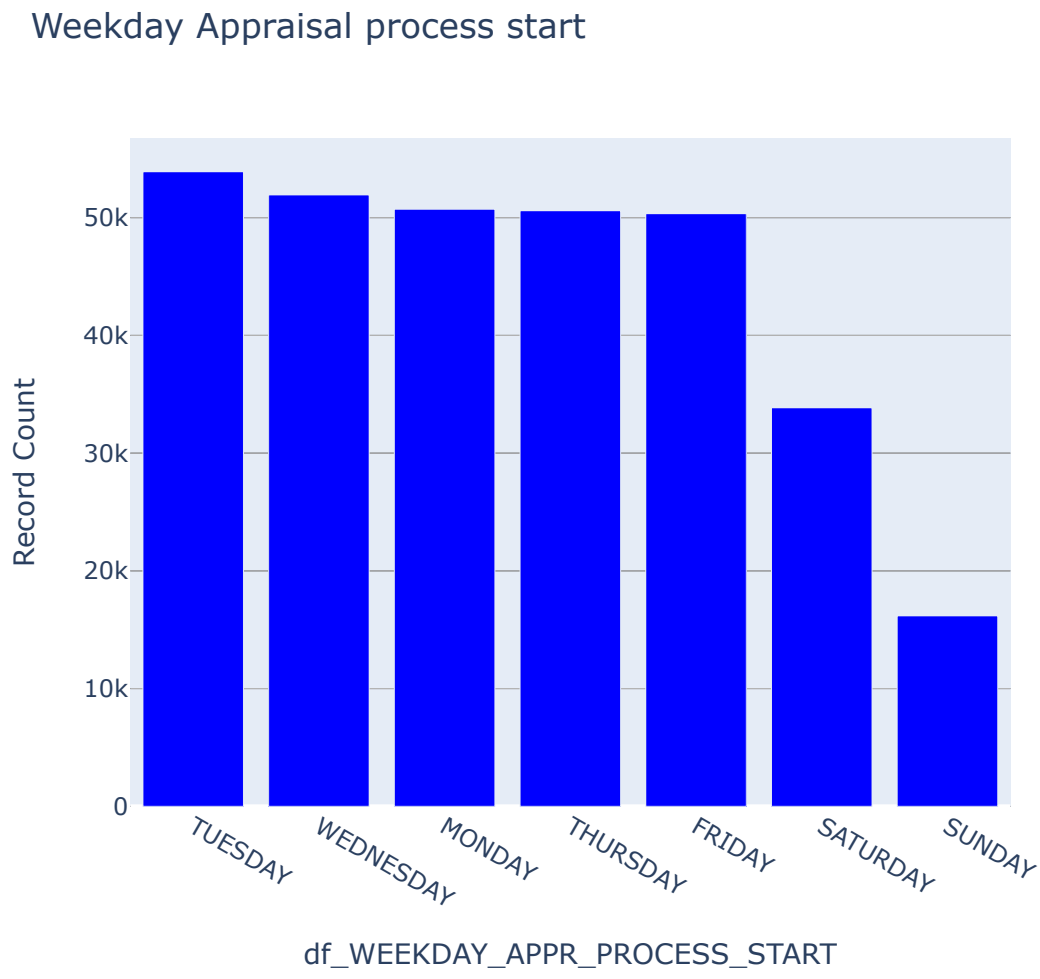
307511 rows × 108 columns

Drop REGION_RATING_CLIENT_ENCODED_W_CITY and REGION_RATING_CLIENT_W_CITY since we don't need it now

In [35]:
```
df_init = df_init.drop(["REGION_RATING_CLIENT_W_CITY_ENCODED", "REGION_RATING_
CLIENT_W_CITY"], axis=1)
```

Visualize WEEKDAY_APPR_PROCESS_START

In [36]:
```
temp_WEEKDAY_APPR_PROCESS_START = df_init["WEEKDAY_APPR_PROCESS_START"].value_
counts()
df_WEEKDAY_APPR_PROCESS_START = pd.DataFrame({'WEEKDAY_APPR_PROCESS_START': te
mp_WEEKDAY_APPR_PROCESS_START.index,'values': temp_WEEKDAY_APPR_PROCESS_START.
values})

trace = go.Bar(
    x = df_WEEKDAY_APPR_PROCESS_START['WEEKDAY_APPR_PROCESS_START'],y = df_WEE
KDAY_APPR_PROCESS_START['values'],
    name="Weekday Appraisal process start",
    marker=dict(color="Blue"),
    text=df_WEEKDAY_APPR_PROCESS_START['values']
)
data = [trace]
layout = dict(title = 'Weekday Appraisal process start',
          xaxis = dict(title = 'df_WEEKDAY_APPR_PROCESS_START', showticklabels
=True),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='df_WEEKDAY_APPR_PROCESS_START')
```



One hot encode WEEKDAY_APPR_PROCESS_START

In [37]:
```
np.where(pd.isnull(df_init['WEEKDAY_APPR_PROCESS_START']))
```

Out[37]:
```
(array([], dtype=int64),)
```

In [38]:
```
le_WEEKDAY_APPR_PROCESS_START = LabelEncoder()
df_init['WEEKDAY_APPR_PROCESS_START_ENCODED'] = le_WEEKDAY_APPR_PROCESS_START.
fit_transform(df_init.WEEKDAY_APPR_PROCESS_START)
```

In [39]:
```
WEEKDAY_APPR_PROCESS_START_ohe = OneHotEncoder()
X = WEEKDAY_APPR_PROCESS_START_ohe.fit_transform(df_init.WEEKDAY_APPR_PROCESS_
START_ENCODED.values.reshape(-1,1)).toarray()
```

In [40]:
```
dfOneHot = pd.DataFrame(X, columns = ["WEEKDAY_APPR_PROCESS_START_"+str(int(i
)) for i in range(X.shape[1])])
df_init = pd.concat([df_init, dfOneHot], axis=1)
```

In [41]:
```
df_init
```

Out[41]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FL
0	100002	1	Cash loans	M	N	
1	100003	0	Cash loans	F	N	
2	100004	0	Revolving loans	M	Y	
3	100006	0	Cash loans	F	N	
4	100007	0	Cash loans	M	N	
...	...	...	...	...	...	
307506	456251	0	Cash loans	M	N	
307507	456252	0	Cash loans	F	N	
307508	456253	0	Cash loans	F	N	
307509	456254	1	Cash loans	F	N	
307510	456255	0	Cash loans	F	N	

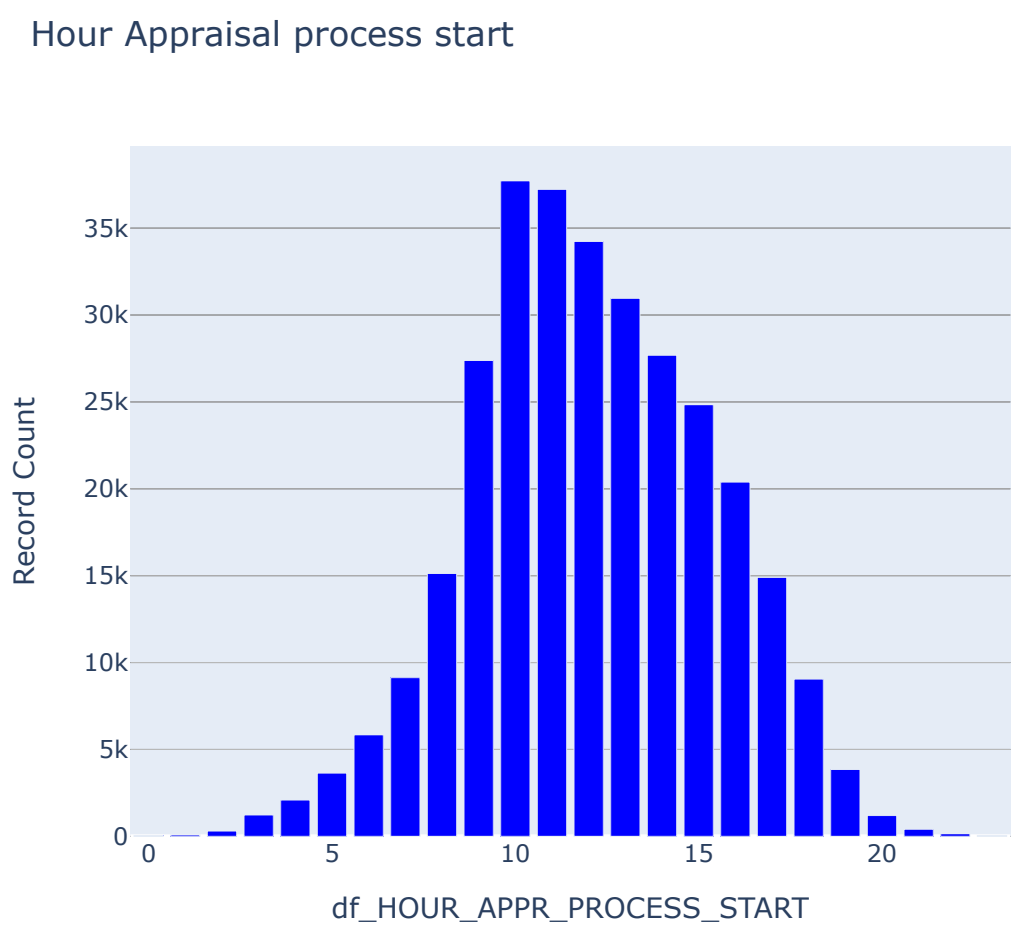
307511 rows × 114 columns

Drop WEEKDAY_APPR_PROCESS_START and WEEKDAY_APPR_PROCESS_START_ENCODED since we don't need it now

In [42]:
```
df_init = df_init.drop(["WEEKDAY_APPR_PROCESS_START_ENCODED", "WEEKDAY_APPR_PR
OCESS_START"], axis=1)
```

Visualize HOUR_APPR_PROCESS_START

In [43]:
```
temp_HOUR_APPR_PROCESS_START = df_init["HOUR_APPR_PROCESS_START"].value_counts
()
df_HOUR_APPR_PROCESS_START = pd.DataFrame({'HOUR_APPR_PROCESS_START': temp_HOU
R_APPR_PROCESS_START.index,'values': temp_HOUR_APPR_PROCESS_START.values})

trace = go.Bar(
    x = df_HOUR_APPR_PROCESS_START['HOUR_APPR_PROCESS_START'],y = df_HOUR_APPR
_PROCESS_START['values'],
    name="Hour Appraisal process start",
    marker=dict(color="Blue"),
    text=df_HOUR_APPR_PROCESS_START['values']
)
data = [trace]
layout = dict(title = 'Hour Appraisal process start',
          xaxis = dict(title = 'df_HOUR_APPR_PROCESS_START', showticklabels=Tr
ue),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='df_HOUR_APPR_PROCESS_START')
```

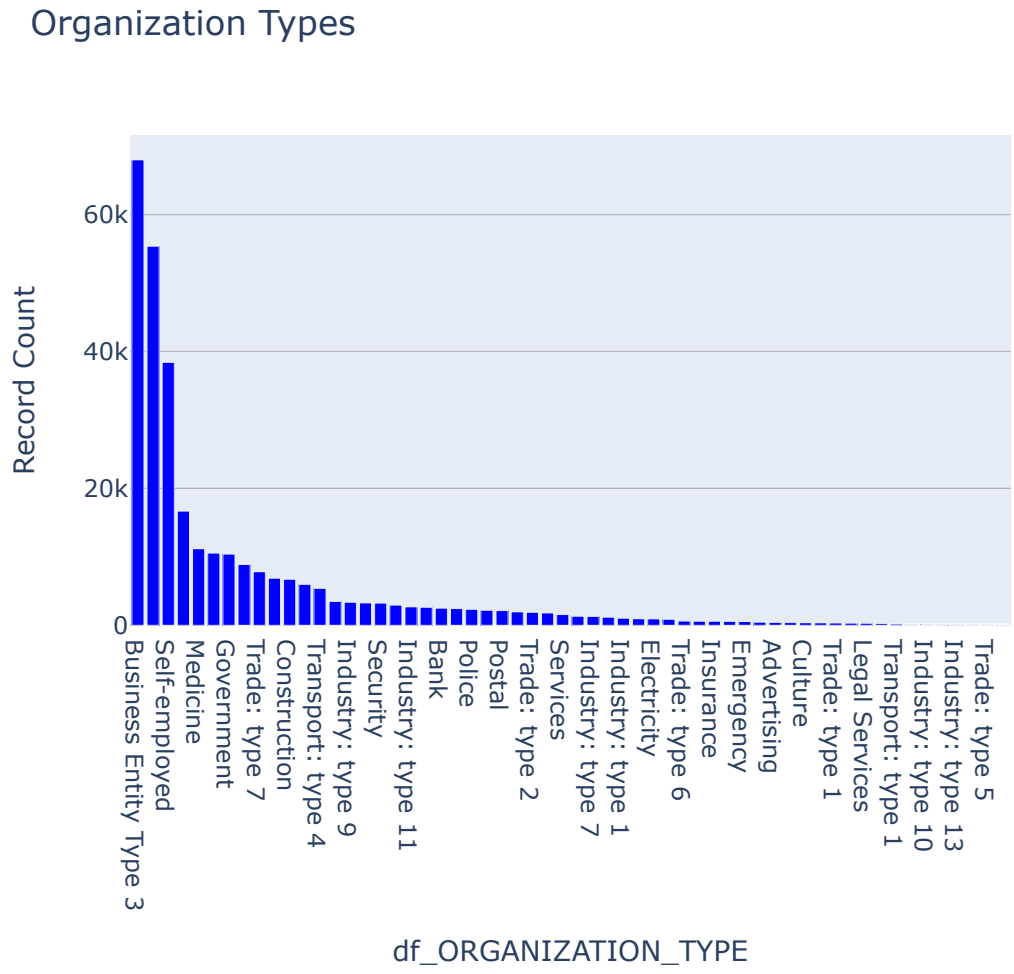


drop HOUR_APPR_PROCESS_START because of so many different values

In [44]:
```
df_init = df_init.drop(["HOUR_APPR_PROCESS_START"], axis=1)
```

Visualize ORGANIZATION_TYPE

In [45]:
```
temp_ORGANIZATION_TYPE = df_init["ORGANIZATION_TYPE"].value_counts()
df_ORGANIZATION_TYPE = pd.DataFrame({'ORGANIZATION_TYPE': temp_ORGANIZATION_TY
PE.index,'values': temp_ORGANIZATION_TYPE.values})

trace = go.Bar(
    x = df_ORGANIZATION_TYPE['ORGANIZATION_TYPE'],y = df_ORGANIZATION_TYPE['va
lues'],
    name="Organization Types",
    marker=dict(color="Blue"),
    text=df_ORGANIZATION_TYPE['values']
)
data = [trace]
layout = dict(title = 'Organization Types',
          xaxis = dict(title = 'df_ORGANIZATION_TYPE', showticklabels=True),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='df_ORGANIZATION_TYPE')
```



Analyse More on ORGANIZATION_TYPE

In [46]:
```
np.where(pd.isnull(df_init['ORGANIZATION_TYPE']))
```

Out[46]:
```
(array([], dtype=int64),)
```

In [47]:
```
df_init['ORGANIZATION_TYPE'].unique()
```

Out[47]:
```
array(['Business Entity Type 3', 'School', 'Government', 'Religion',
       'Other', 'XNA', 'Electricity', 'Medicine',
       'Business Entity Type 2', 'Self-employed', 'Transport: type 2',
       'Construction', 'Housing', 'Kindergarten', 'Trade: type 7',
       'Industry: type 11', 'Military', 'Services', 'Security Ministries',
       'Transport: type 4', 'Industry: type 1', 'Emergency', 'Security',
       'Trade: type 2', 'University', 'Transport: type 3', 'Police',
       'Business Entity Type 1', 'Postal', 'Industry: type 4',
       'Agriculture', 'Restaurant', 'Culture', 'Hotel',
       'Industry: type 7', 'Trade: type 3', 'Industry: type 3', 'Bank',
       'Industry: type 9', 'Insurance', 'Trade: type 6',
       'Industry: type 2', 'Transport: type 1', 'Industry: type 12',
       'Mobile', 'Trade: type 1', 'Industry: type 5', 'Industry: type 10',
       'Legal Services', 'Advertising', 'Trade: type 5', 'Cleaning',
       'Industry: type 13', 'Trade: type 4', 'Telecom',
       'Industry: type 8', 'Realtor', 'Industry: type 6'], dtype=object)
```

Change the organization type to reduce number of distinct values as per below table:

```
Original Value  New Value
```

Business Entity Type 1 Business Business Entity Type 2 Business Business Entity Type 3 Business Business Entity Type 3 Business Industry: type 1 Industry Industry: type 10 Industry Industry: type 11 Industry Industry: type 12 Industry Industry: type 13 Industry Industry: type 2 Industry Industry: type 3 Industry Industry: type 4 Industry Industry: type 5 Industry Industry: type 6 Industry Industry: type 7 Industry Industry: type 8 Industry Industry: type 9 Industry Trade: type 1 Trade Trade: type 2 Trade Trade: type 3 Trade Trade: type 4 Trade Trade: type 5 Trade Trade: type 6 Trade Trade: type 7 Trade Transport: type 1 Transport Transport: type 2 Transport Transport: type 3 Transport Transport: type 4 Transport

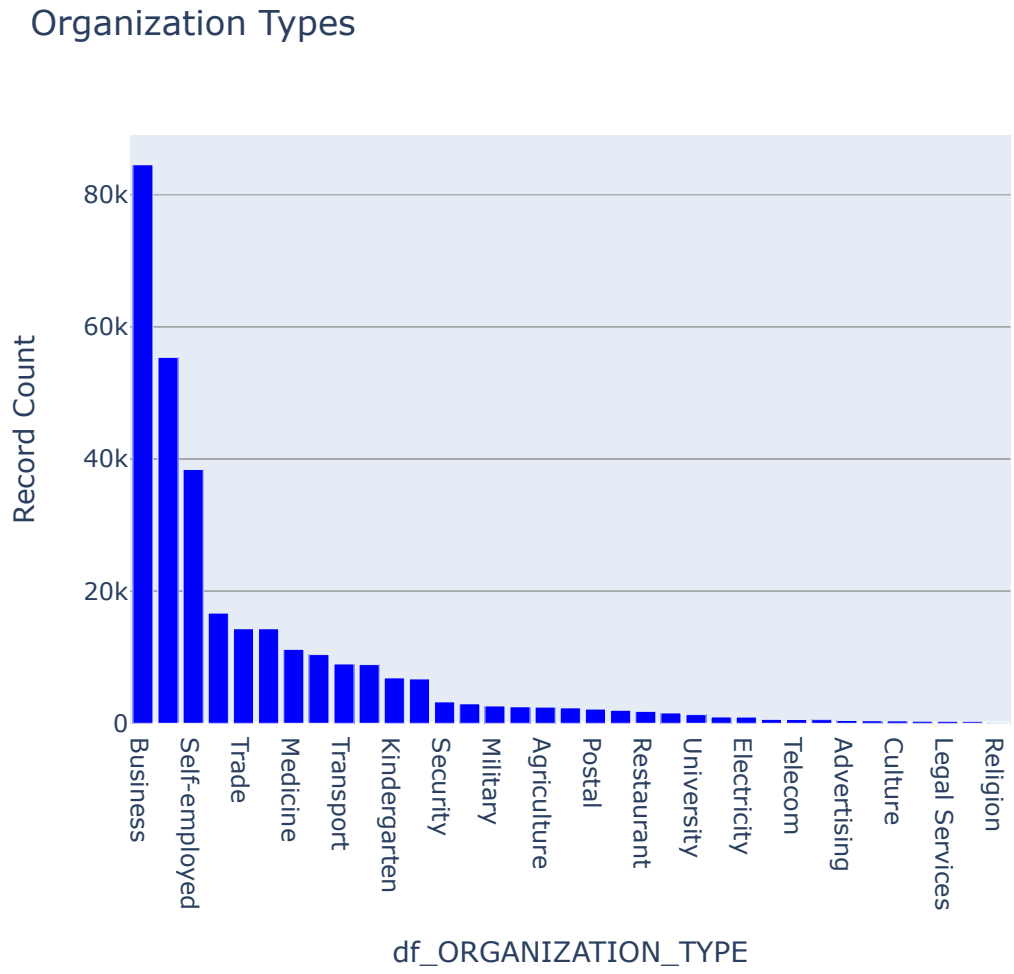
In [48]:
```
df_init.loc[df_init['ORGANIZATION_TYPE'].str.contains('Business'), 'ORGANIZATI
ON_TYPE'] = 'Business'
df_init.loc[df_init['ORGANIZATION_TYPE'].str.contains('Industry'), 'ORGANIZATI
ON_TYPE'] = 'Industry'
df_init.loc[df_init['ORGANIZATION_TYPE'].str.contains('Trade'), 'ORGANIZATION_
TYPE'] = 'Trade'
df_init.loc[df_init['ORGANIZATION_TYPE'].str.contains('Transport'), 'ORGANIZAT
ION_TYPE'] = 'Transport'
```

In [49]:
```
df_init['ORGANIZATION_TYPE'].unique()
```

Out[49]:
```
array(['Business', 'School', 'Government', 'Religion', 'Other', 'XNA',
       'Electricity', 'Medicine', 'Self-employed', 'Transport',
       'Construction', 'Housing', 'Kindergarten', 'Trade', 'Industry',
       'Military', 'Services', 'Security Ministries', 'Emergency',
       'Security', 'University', 'Police', 'Postal', 'Agriculture',
       'Restaurant', 'Culture', 'Hotel', 'Bank', 'Insurance', 'Mobile',
       'Legal Services', 'Advertising', 'Cleaning', 'Telecom', 'Realtor'],
      dtype=object)
```

Visualize organization type with new values

In [50]:
```
temp_ORGANIZATION_TYPE = df_init["ORGANIZATION_TYPE"].value_counts()
df_ORGANIZATION_TYPE = pd.DataFrame({'ORGANIZATION_TYPE': temp_ORGANIZATION_TY
PE.index,'values': temp_ORGANIZATION_TYPE.values})

trace = go.Bar(
    x = df_ORGANIZATION_TYPE['ORGANIZATION_TYPE'],y = df_ORGANIZATION_TYPE['va
lues'],
    name="Organization Types",
    marker=dict(color="Blue"),
    text=df_ORGANIZATION_TYPE['values']
)
data = [trace]
layout = dict(title = 'Organization Types',
          xaxis = dict(title = 'df_ORGANIZATION_TYPE', showticklabels=True),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='df_ORGANIZATION_TYPE')
```



Apply one hot encoding for ORGANIZATION_TYPE

In [51]:
```
np.where(pd.isnull(df_init['ORGANIZATION_TYPE']))
```

Out[51]:
```
(array([], dtype=int64),)
```

In [52]:
```
#le_ORGANIZATION_TYPE = LabelEncoder()
#df_init['ORGANIZATION_TYPE_ENCODED'] = le_ORGANIZATION_TYPE.fit_transform(df_
init.ORGANIZATION_TYPE)
```

In [53]:
```
#ORGANIZATION_TYPE_ohe = OneHotEncoder()
#X = ORGANIZATION_TYPE_ohe.fit_transform(df_init.ORGANIZATION_TYPE_ENCODED.val
ues.reshape(-1,1)).toarray()
```

In [54]:
```
#dfOneHot = pd.DataFrame(X, columns = ["ORGANIZATION_TYPE_"+str(int(i)) for i
 in range(X.shape[1])])
#df_init = pd.concat([df_init, dfOneHot], axis=1)
```

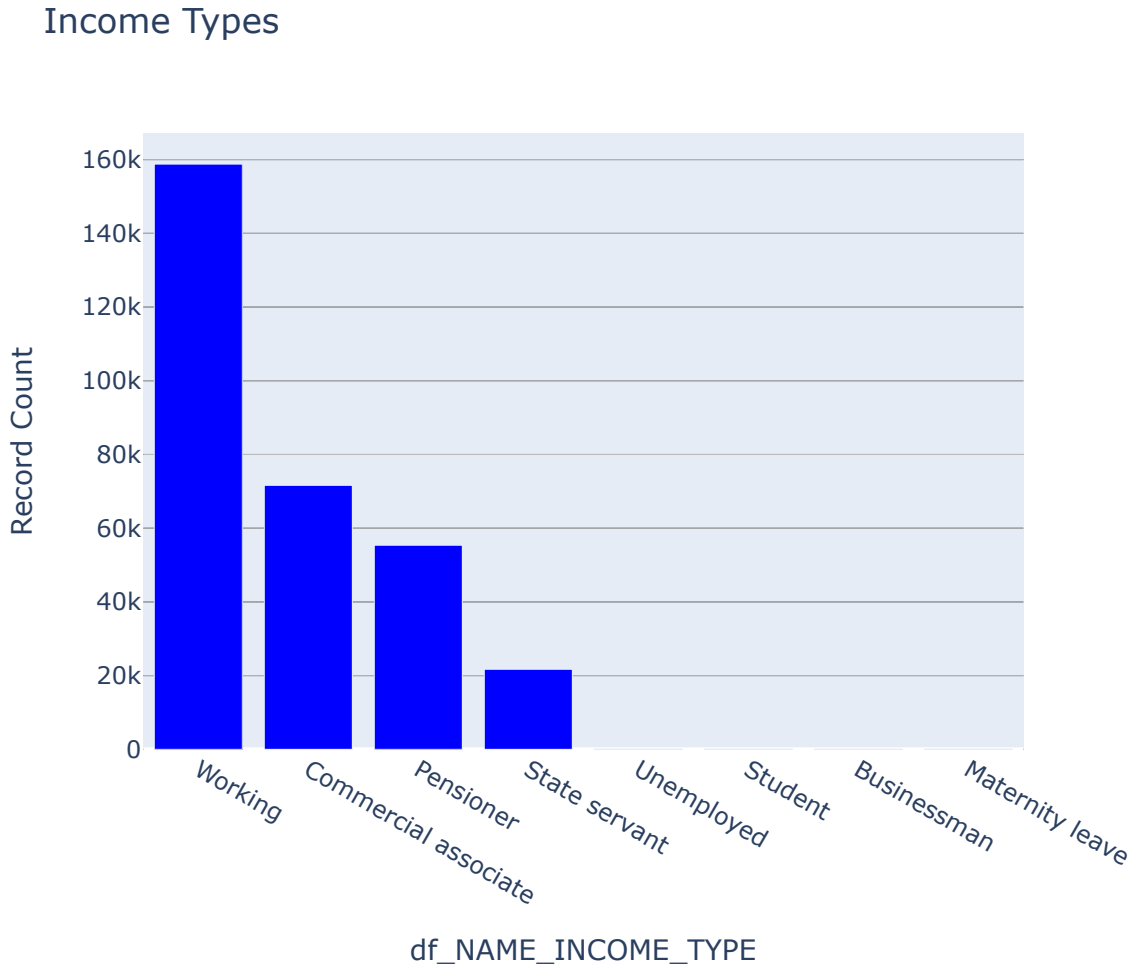
In [55]:
```
#df_init = df_init.drop(["ORGANIZATION_TYPE_ENCODED", "ORGANIZATION_TYPE"], ax
is=1)
```

Covert ORGANIZATION_TYPE into categorical type

In [56]:
```
df_init['ORGANIZATION_TYPE'] = df_init['ORGANIZATION_TYPE'].astype('category')
```

Visualize NAME_INCOME_TYPE

In [57]:
```
temp_NAME_INCOME_TYPE = df_init["NAME_INCOME_TYPE"].value_counts()
df_NAME_INCOME_TYPE = pd.DataFrame({'NAME_INCOME_TYPE': temp_NAME_INCOME_TYPE.
index,'values': temp_NAME_INCOME_TYPE.values})

trace = go.Bar(
    x = df_NAME_INCOME_TYPE['NAME_INCOME_TYPE'],y = df_NAME_INCOME_TYPE['value
s'],
    name="Income Types",
    marker=dict(color="Blue"),
    text=df_NAME_INCOME_TYPE['values']
)
data = [trace]
layout = dict(title = 'Income Types',
          xaxis = dict(title = 'df_NAME_INCOME_TYPE', showticklabels=True),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='df_NAME_INCOME_TYPE')
```



In [58]:
```
np.where(pd.isnull(df_init['NAME_INCOME_TYPE']))
```

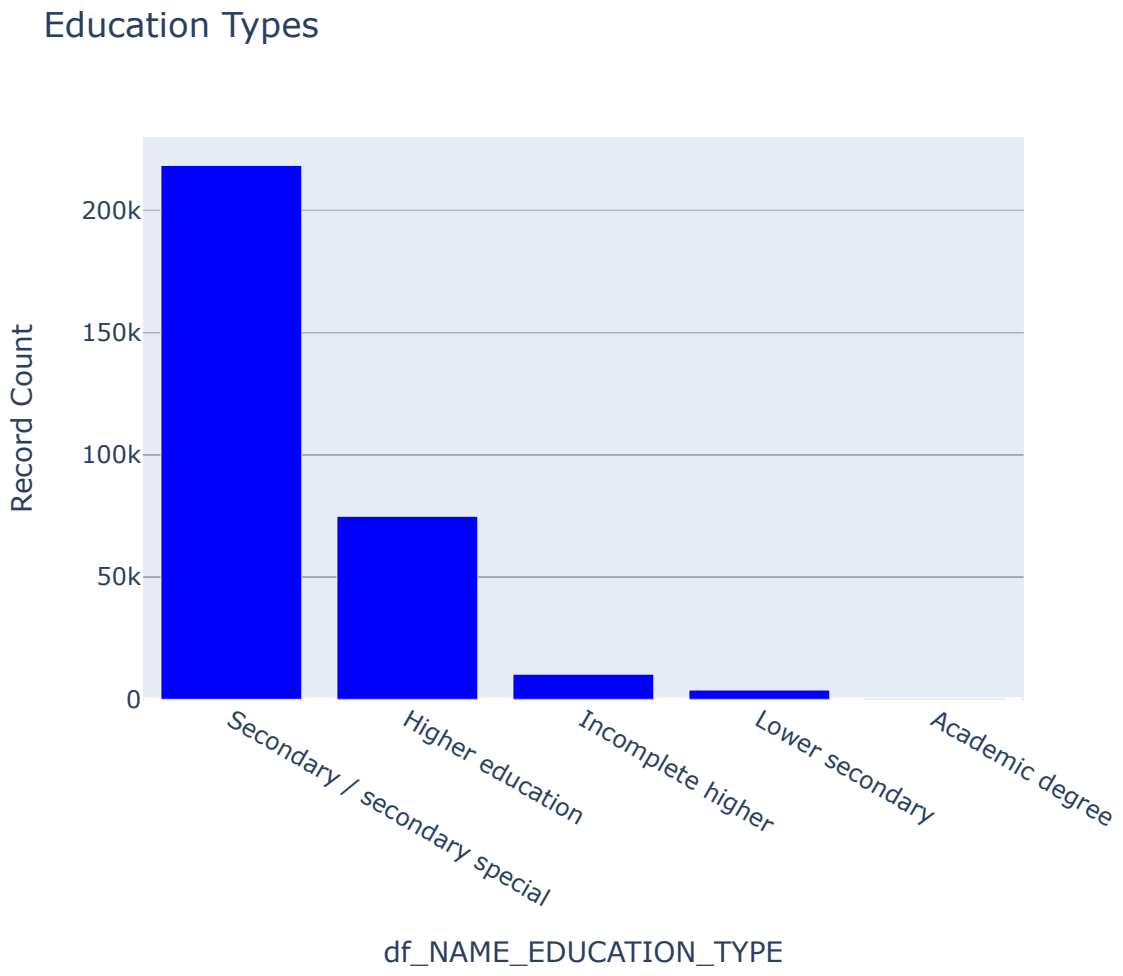
Out[58]:
```
(array([], dtype=int64),)
```

Convert Income Type into categorial variable

In [59]:
```
df_init['NAME_INCOME_TYPE'] = df_init['NAME_INCOME_TYPE'].astype('category')
```

Visualize NAME_EDUCATION_TYPE

In [60]:
```
temp_NAME_EDUCATION_TYPE = df_init["NAME_EDUCATION_TYPE"].value_counts()
df_NAME_EDUCATION_TYPE = pd.DataFrame({'NAME_EDUCATION_TYPE': temp_NAME_EDUCAT
ION_TYPE.index,'values': temp_NAME_EDUCATION_TYPE.values})

trace = go.Bar(
    x = df_NAME_EDUCATION_TYPE['NAME_EDUCATION_TYPE'],y = df_NAME_EDUCATION_TY
PE['values'],
    name="Education Types",
    marker=dict(color="Blue"),
    text=df_NAME_EDUCATION_TYPE['values']
)
data = [trace]
layout = dict(title = 'Education Types',
          xaxis = dict(title = 'df_NAME_EDUCATION_TYPE', showticklabels=True),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='df_NAME_EDUCATION_TYPE')
```



In [61]:
```
np.where(pd.isnull(df_init['NAME_EDUCATION_TYPE']))
```

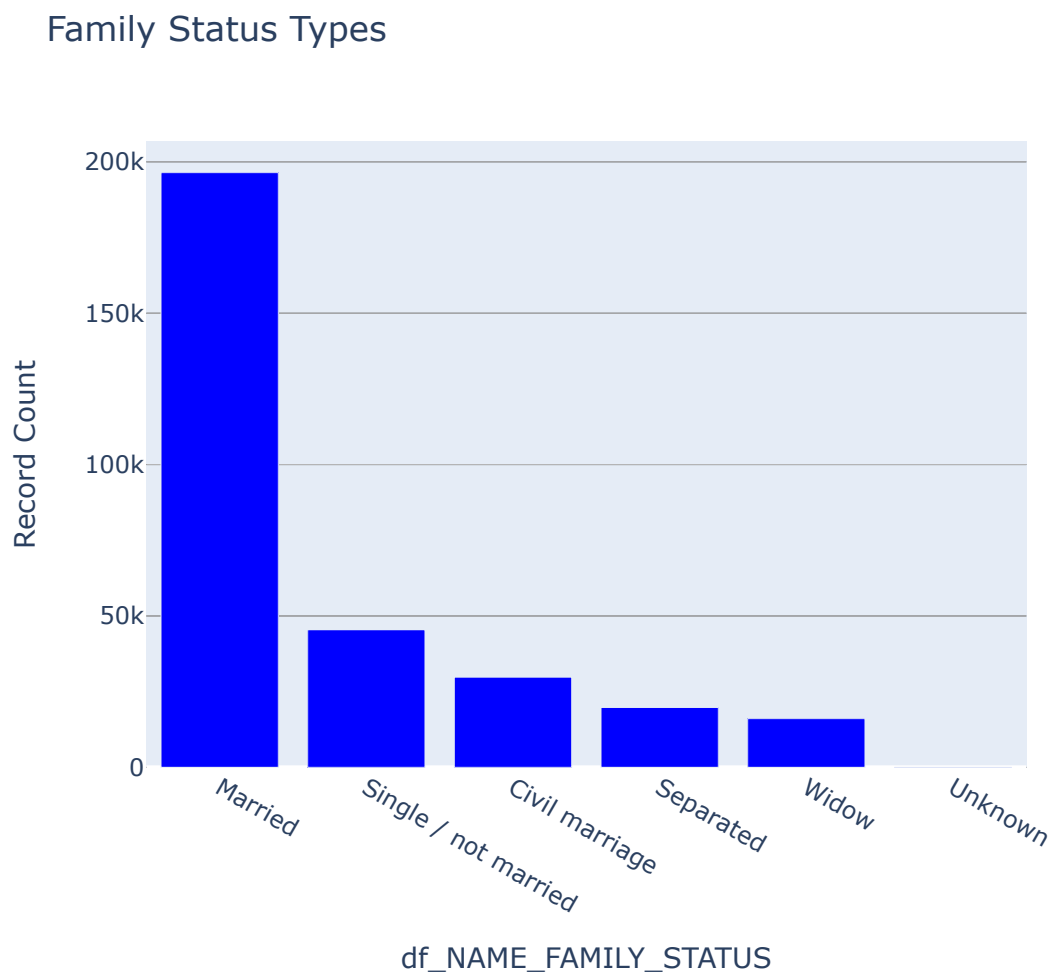
Out[61]:
```
(array([], dtype=int64),)
```

Convert Eduction type in categorical variable

In [62]:
```
df_init['NAME_EDUCATION_TYPE'] = df_init['NAME_EDUCATION_TYPE'].astype('catego
ry')
```

Visualize Family Status

In [63]:
```
temp_NAME_FAMILY_STATUS = df_init["NAME_FAMILY_STATUS"].value_counts()
df_NAME_FAMILY_STATUS = pd.DataFrame({'NAME_FAMILY_STATUS': temp_NAME_FAMILY_S
TATUS.index,'values': temp_NAME_FAMILY_STATUS.values})

trace = go.Bar(
    x = df_NAME_FAMILY_STATUS['NAME_FAMILY_STATUS'],y = df_NAME_FAMILY_STATUS[
'values'],
    name="Family Status Types",
    marker=dict(color="Blue"),
    text=df_NAME_FAMILY_STATUS['values']
)
data = [trace]
layout = dict(title = 'Family Status Types',
          xaxis = dict(title = 'df_NAME_FAMILY_STATUS', showticklabels=True),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='df_NAME_FAMILY_STATUS')
```



In [64]:
```
np.where(pd.isnull(df_init['NAME_FAMILY_STATUS']))
```

Out[64]:
```
(array([], dtype=int64),)
```

Convert Family Status into categorical

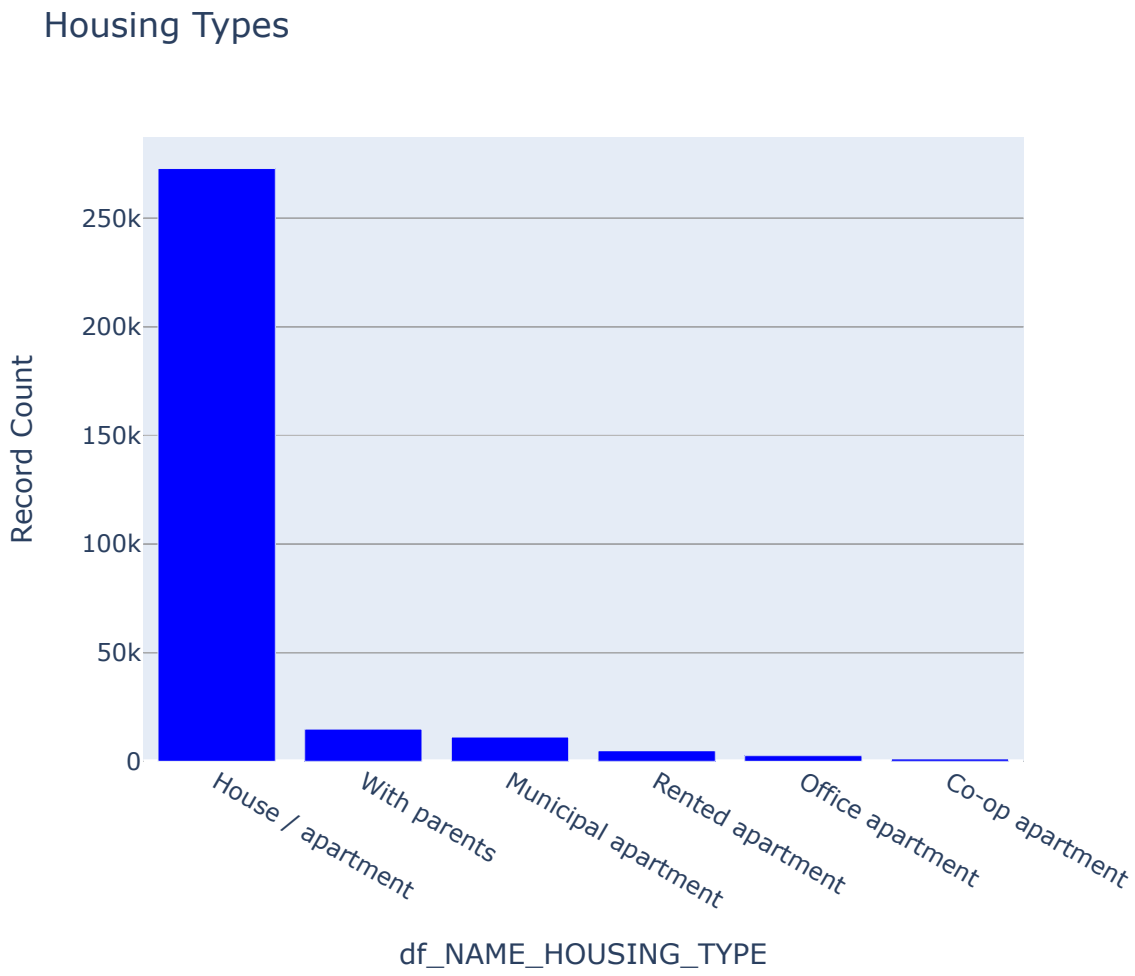
In [65]:
```
df_init['NAME_FAMILY_STATUS'] = df_init['NAME_FAMILY_STATUS'].astype('categor
y')
```

In [66]:
```
df_init['NAME_FAMILY_STATUS']
```

Out[66]:
```
0         Single / not married
1                      Married
2         Single / not married
3               Civil marriage
4         Single / not married
                  ...
307506               Separated
307507                   Widow
307508               Separated
307509                 Married
307510                 Married
Name: NAME_FAMILY_STATUS, Length: 307511, dtype: category
Categories (6, object): [Civil marriage, Married, Separated, Single / not mar
ried, Unknown, Widow]
```

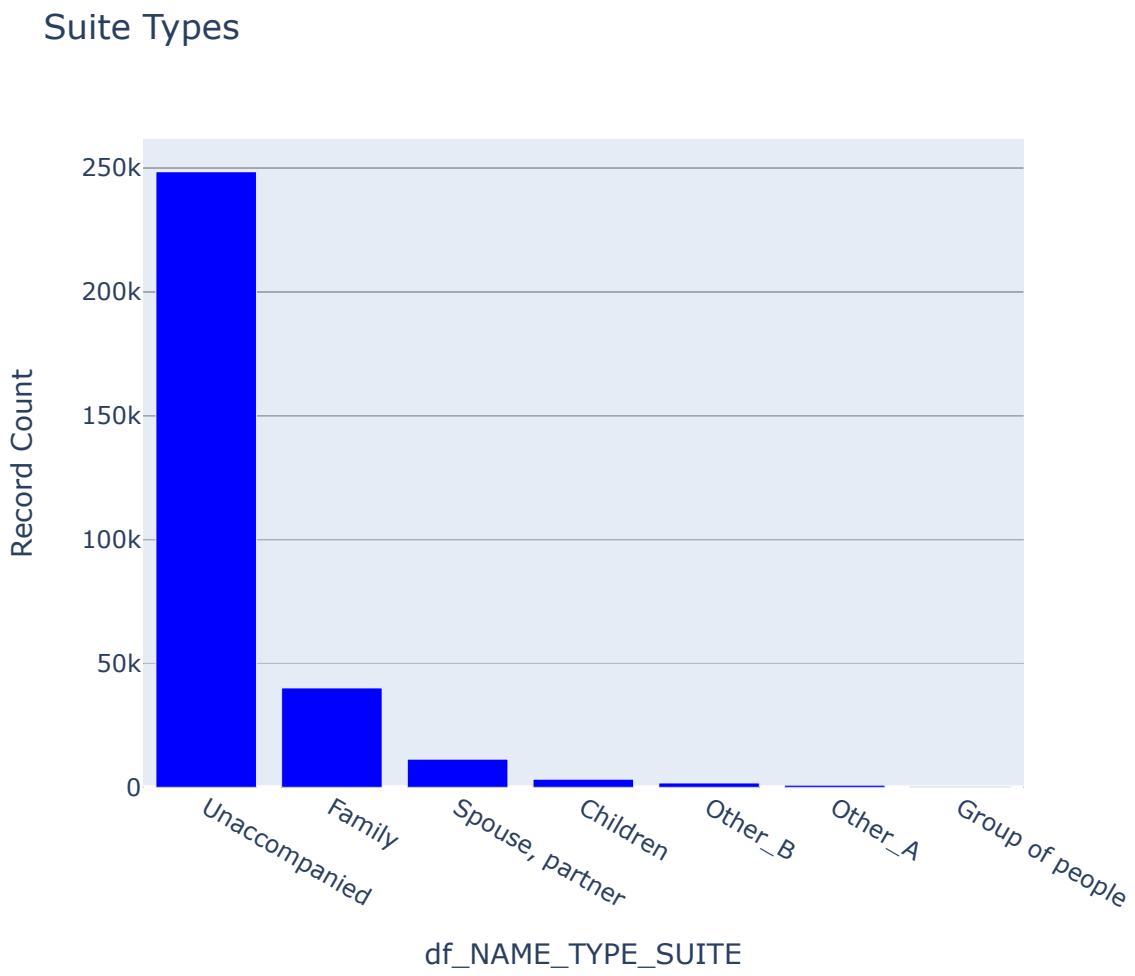
Visualize NAME_HOUSING_TYPE

In [67]:
```
temp_NAME_HOUSING_TYPE = df_init["NAME_HOUSING_TYPE"].value_counts()
df_NAME_HOUSING_TYPE = pd.DataFrame({'NAME_HOUSING_TYPE': temp_NAME_HOUSING_TY
PE.index,'values': temp_NAME_HOUSING_TYPE.values})

trace = go.Bar(
    x = df_NAME_HOUSING_TYPE['NAME_HOUSING_TYPE'],y = df_NAME_HOUSING_TYPE['va
lues'],
    name="Housing Types",
    marker=dict(color="Blue"),
    text=df_NAME_HOUSING_TYPE['values']
)
data = [trace]
layout = dict(title = 'Housing Types',
          xaxis = dict(title = 'df_NAME_HOUSING_TYPE', showticklabels=True),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='df_NAME_HOUSING_TYPE')
```



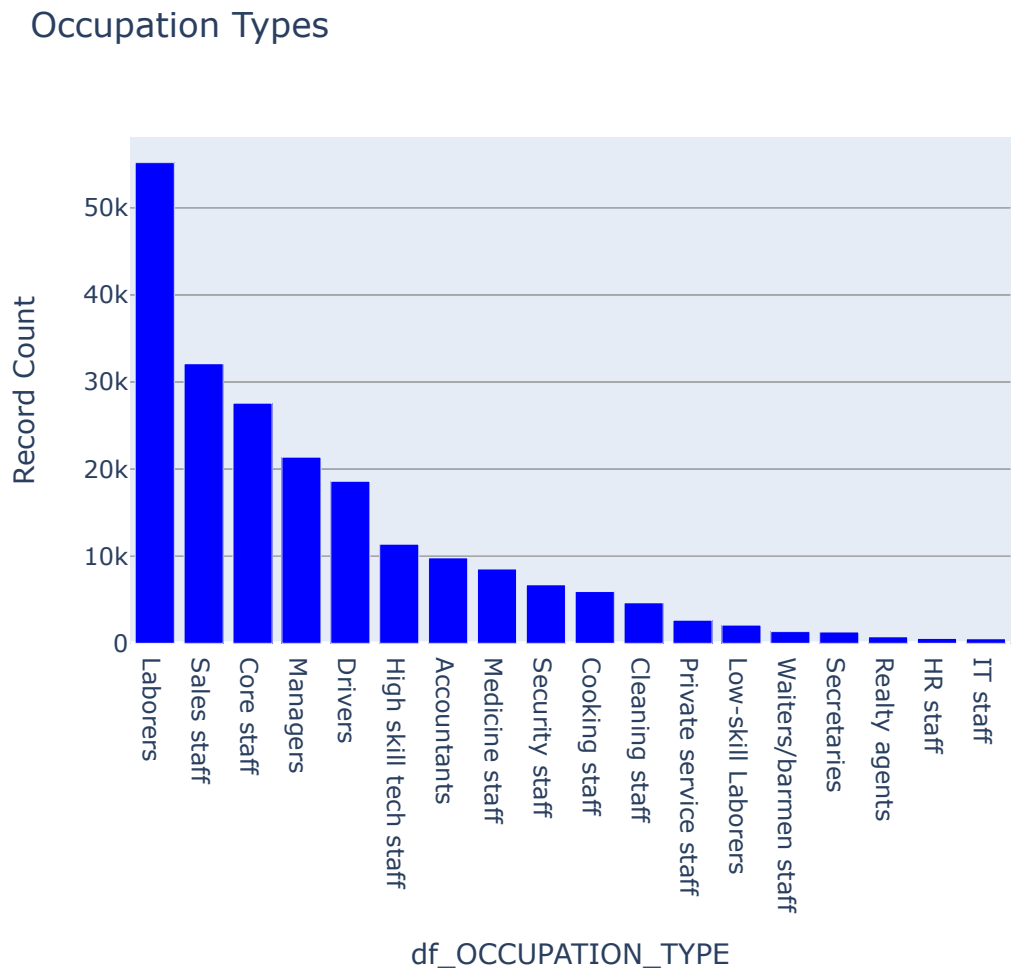
Visualize NAME_TYPE_SUITE

In [68]:
```
temp_NAME_TYPE_SUITE = df_init["NAME_TYPE_SUITE"].value_counts()
df_NAME_TYPE_SUITE = pd.DataFrame({'NAME_TYPE_SUITE': temp_NAME_TYPE_SUITE.ind
ex,'values': temp_NAME_TYPE_SUITE.values})

trace = go.Bar(
    x = df_NAME_TYPE_SUITE['NAME_TYPE_SUITE'],y = df_NAME_TYPE_SUITE['values'
],
    name="Suite Types",
    marker=dict(color="Blue"),
    text=df_NAME_TYPE_SUITE['values']
)
data = [trace]
layout = dict(title = 'Suite Types',
          xaxis = dict(title = 'df_NAME_TYPE_SUITE', showticklabels=True),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='df_NAME_TYPE_SUITE')
```



Visualize OCCUPATION_TYPE

In [69]:
```
temp_OCCUPATION_TYPE = df_init["OCCUPATION_TYPE"].value_counts()
df_OCCUPATION_TYPE = pd.DataFrame({'OCCUPATION_TYPE': temp_OCCUPATION_TYPE.ind
ex,'values': temp_OCCUPATION_TYPE.values})

trace = go.Bar(
    x = df_OCCUPATION_TYPE['OCCUPATION_TYPE'],y = df_OCCUPATION_TYPE['values'
],
    name="Occupation Types",
    marker=dict(color="Blue"),
    text=df_OCCUPATION_TYPE['values']
)
data = [trace]
layout = dict(title = 'Occupation Types',
          xaxis = dict(title = 'df_OCCUPATION_TYPE', showticklabels=True),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='df_OCCUPATION_TYPE')
```



Replace NULL with UNKNOWN

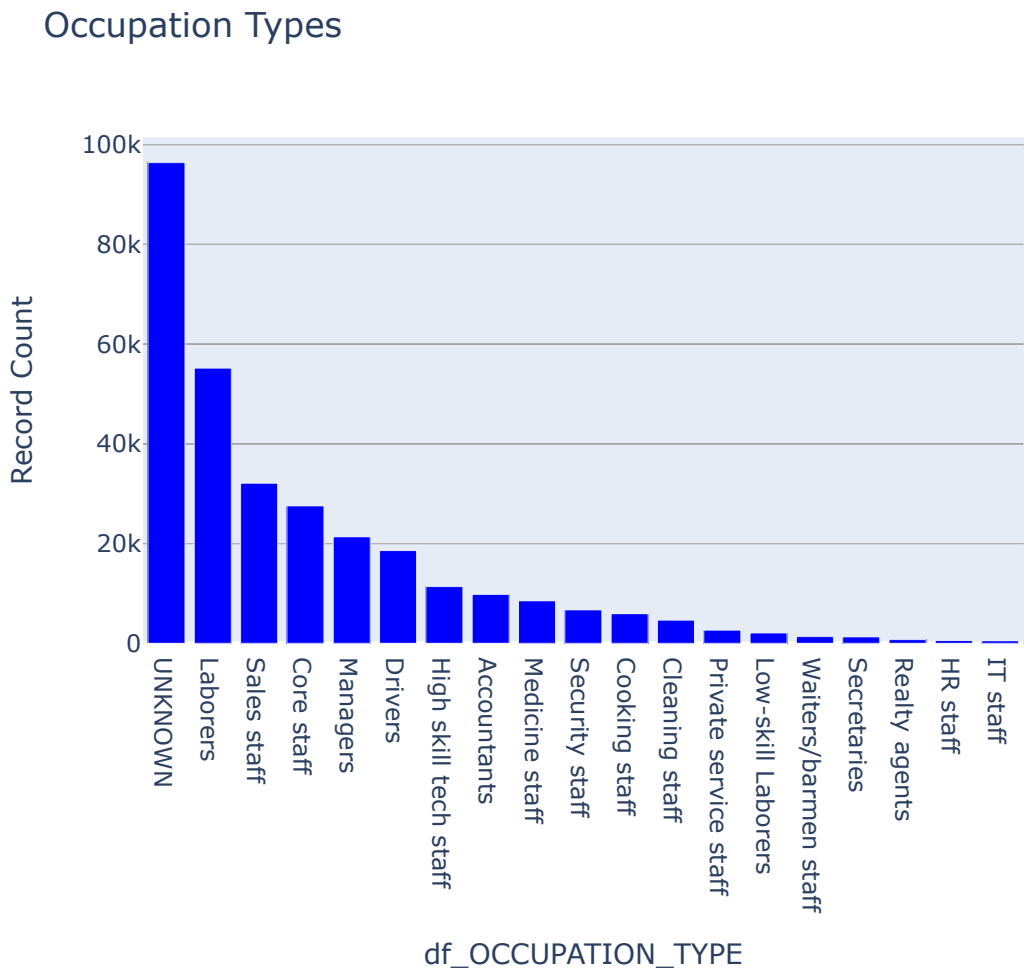
In [70]:
```
np.where(pd.isnull(df_init['OCCUPATION_TYPE']))
```

Out[70]:
```
(array([     8,     11,     23, ..., 307500, 307505, 307507], dtype=int64),)
```

In [71]:
```
df_init['OCCUPATION_TYPE'] = df_init.OCCUPATION_TYPE.fillna('UNKNOWN')
```

Visualize after filling UNKNOWN

In [72]:
```
temp_OCCUPATION_TYPE = df_init["OCCUPATION_TYPE"].value_counts()
df_OCCUPATION_TYPE = pd.DataFrame({'OCCUPATION_TYPE': temp_OCCUPATION_TYPE.ind
ex,'values': temp_OCCUPATION_TYPE.values})

trace = go.Bar(
    x = df_OCCUPATION_TYPE['OCCUPATION_TYPE'],y = df_OCCUPATION_TYPE['values'
],
    name="Occupation Types",
    marker=dict(color="Blue"),
    text=df_OCCUPATION_TYPE['values']
)
data = [trace]
layout = dict(title = 'Occupation Types',
          xaxis = dict(title = 'df_OCCUPATION_TYPE', showticklabels=True),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='df_OCCUPATION_TYPE')
```



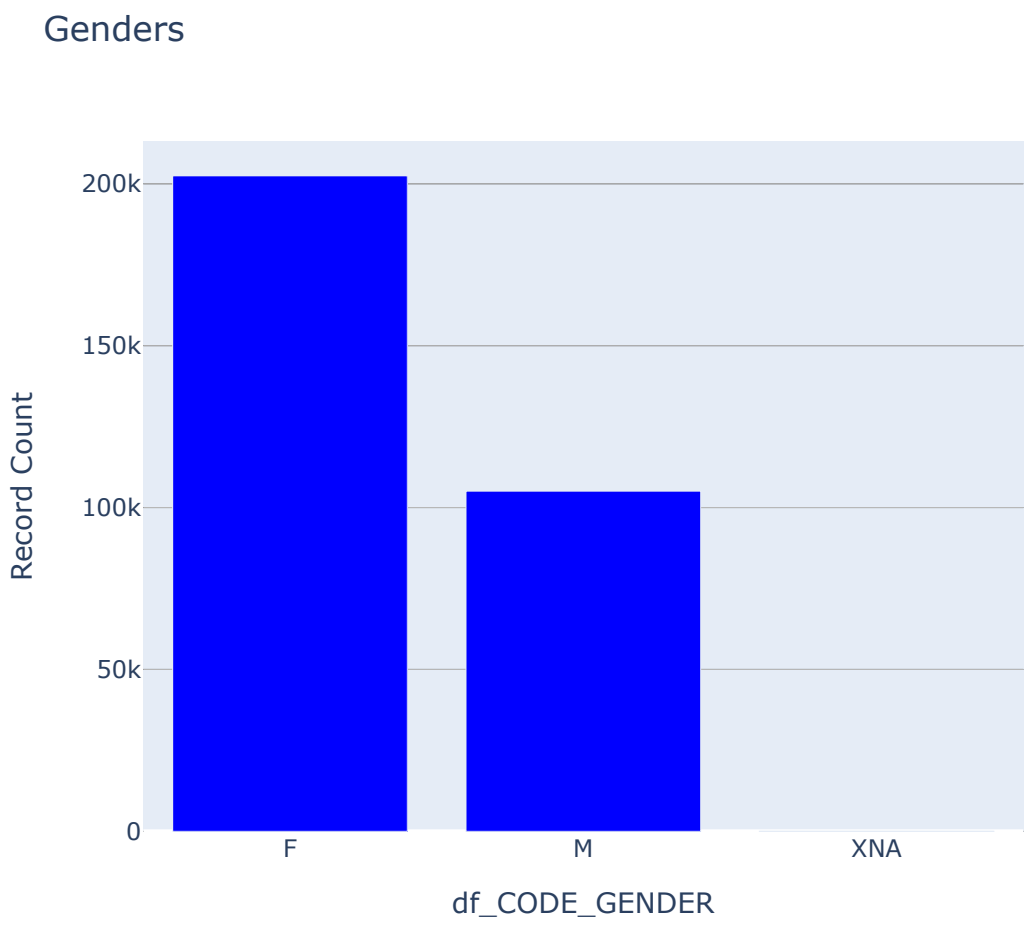
Convert Into categorical variable

In [73]:
```
df_init['OCCUPATION_TYPE'] = df_init['OCCUPATION_TYPE'].astype('category')
```

Analyse gender, FLAG_OWN_CAR, FLAG_OWN_REALTY and then change it to binary

Visualize Gender flag

In [74]:
```
temp_CODE_GENDER = df_init["CODE_GENDER"].value_counts()
df_CODE_GENDER = pd.DataFrame({'CODE_GENDER': temp_CODE_GENDER.index,'values':
temp_CODE_GENDER.values})

trace = go.Bar(
    x = df_CODE_GENDER['CODE_GENDER'],y = df_CODE_GENDER['values'],
    name="Genders",
    marker=dict(color="Blue"),
    text=df_CODE_GENDER['values']
)
data = [trace]
layout = dict(title = 'Genders',
          xaxis = dict(title = 'df_CODE_GENDER', showticklabels=True),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='df_CODE_GENDER')
```



In [75]:
```
df_gender_count = df_init.groupby('CODE_GENDER')['SK_ID_CURR'].nunique()
df_gender_count
```

Out[75]:
```
CODE_GENDER
F      202448
M      105059
XNA         4
Name: SK_ID_CURR, dtype: int64
```

In [76]:
```
np.where(pd.isnull(df_init['CODE_GENDER']))
```

Out[76]:
```
(array([], dtype=int64),)
```

Replace XNA in gender column to M

In [77]:
```
df_init.loc[df_init['CODE_GENDER'].str.contains('XNA'), 'CODE_GENDER'] = 'M'
```

In [78]:
```
df_gender_count = df_init.groupby('CODE_GENDER')['SK_ID_CURR'].nunique()
df_gender_count
```

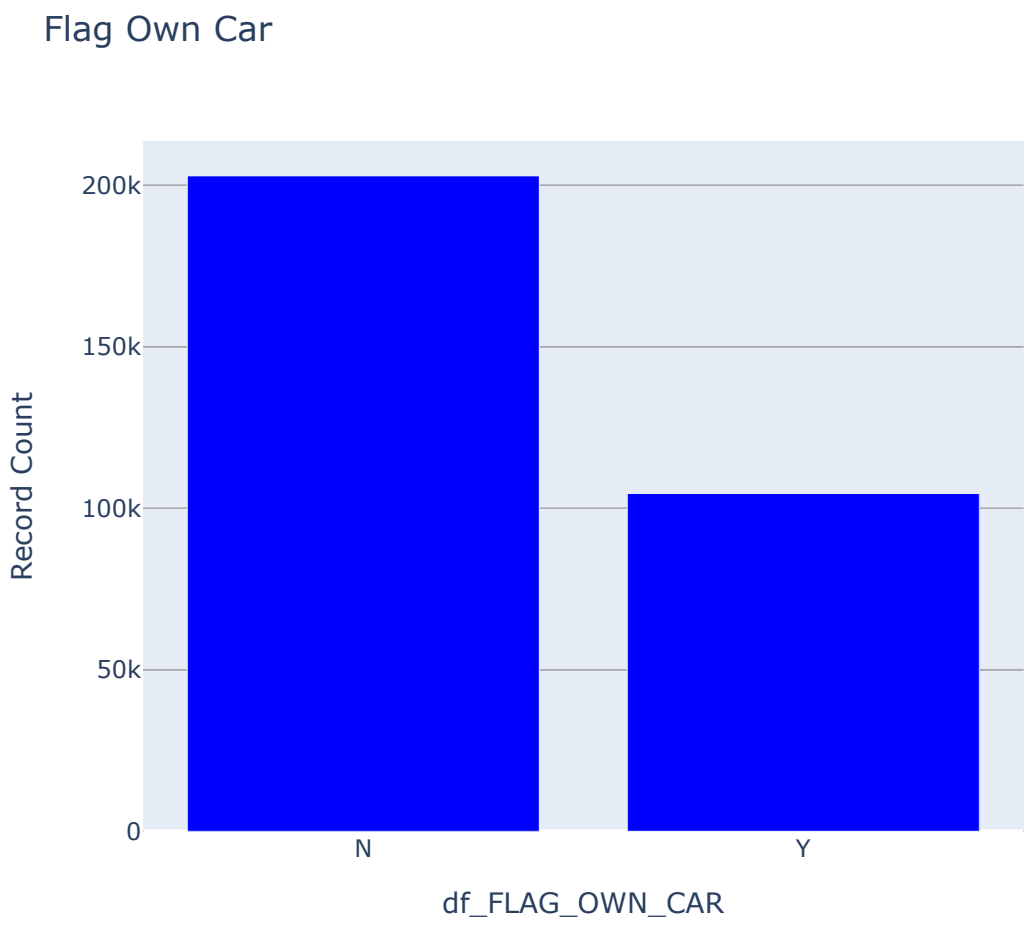
Out[78]:
```
CODE_GENDER
F    202448
M    105063
Name: SK_ID_CURR, dtype: int64
```

Convert Gender column into Boolean

In [79]:
```
df_init['CODE_GENDER'] = np.where(df_init['CODE_GENDER'] == "M", True, False)
```

Visualize FLAG_OWN_CAR

In [80]:
```
temp_FLAG_OWN_CAR = df_init["FLAG_OWN_CAR"].value_counts()
df_FLAG_OWN_CAR = pd.DataFrame({'FLAG_OWN_CAR': temp_FLAG_OWN_CAR.index,'value
s': temp_FLAG_OWN_CAR.values})

trace = go.Bar(
    x = df_FLAG_OWN_CAR['FLAG_OWN_CAR'],y = df_FLAG_OWN_CAR['values'],
    name="Flag Own Car",
    marker=dict(color="Blue"),
    text=df_FLAG_OWN_CAR['values']
)
data = [trace]
layout = dict(title = 'Flag Own Car',
          xaxis = dict(title = 'df_FLAG_OWN_CAR', showticklabels=True),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='df_FLAG_OWN_CAR')
```

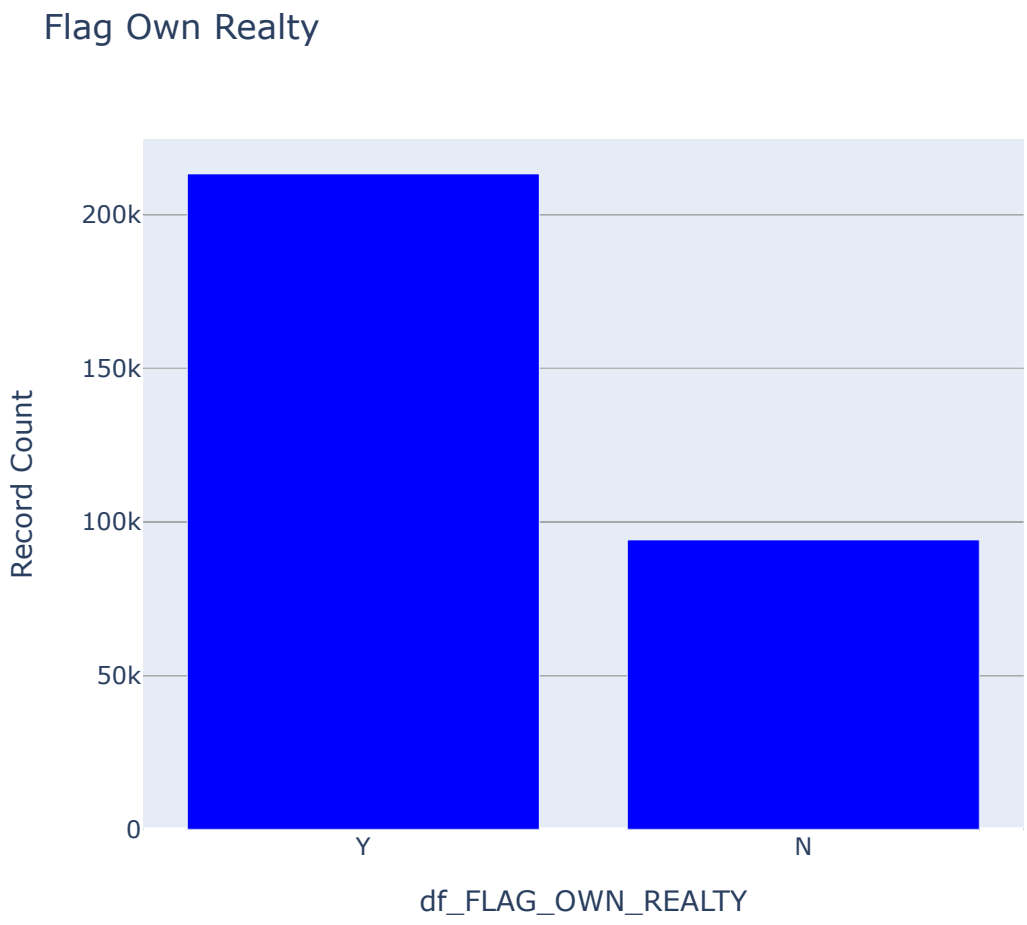


Convert FLAG_OWN_CAR into boolean

In [81]:
```
df_init['FLAG_OWN_CAR'] = np.where(df_init['FLAG_OWN_CAR'] == "Y", True, False
)
```

Visualize FLAG_OWN_REALTY

In [82]:
```
temp_FLAG_OWN_REALTY = df_init["FLAG_OWN_REALTY"].value_counts()
df_FLAG_OWN_REALTY = pd.DataFrame({'FLAG_OWN_REALTY': temp_FLAG_OWN_REALTY.ind
ex,'values': temp_FLAG_OWN_REALTY.values})

trace = go.Bar(
    x = df_FLAG_OWN_REALTY['FLAG_OWN_REALTY'],y = df_FLAG_OWN_REALTY['values'
],
    name="Flag Own Realty",
    marker=dict(color="Blue"),
    text=df_FLAG_OWN_REALTY['values']
)
data = [trace]
layout = dict(title = 'Flag Own Realty',
          xaxis = dict(title = 'df_FLAG_OWN_REALTY', showticklabels=True),
          yaxis = dict(title = 'Record Count'),
          hovermode = 'closest',width=600
         )
fig = dict(data=data, layout=layout)
iplot(fig, filename='df_FLAG_OWN_REALTY')
```



Convert FLAG_OWN_REALTY into Boolean

In [83]:
```
df_init['FLAG_OWN_REALTY'] = np.where(df_init['FLAG_OWN_REALTY'] == "Y", True,
False)
```

Convert REG_CITY_NOT_LIVE_CITY to boolean

In [84]:
```
df_init['REG_CITY_NOT_LIVE_CITY'] = np.where(df_init['REG_CITY_NOT_LIVE_CITY']
== 1, True, False)
```

Convert REG_CITY_NOT_WORK_CITY to boolean

In [85]:
```
df_init['REG_CITY_NOT_WORK_CITY'] = np.where(df_init['REG_CITY_NOT_WORK_CITY']
== 1, True, False)
```

In [86]:
```
df_init.dtypes
```

Out[86]:
```
SK_ID_CURR                      int64
TARGET                          int64
NAME_CONTRACT_TYPE             object
CODE_GENDER                      bool
FLAG_OWN_CAR                     bool
                                ...
WEEKDAY_APPR_PROCESS_START_2    float64
WEEKDAY_APPR_PROCESS_START_3    float64
WEEKDAY_APPR_PROCESS_START_4    float64
WEEKDAY_APPR_PROCESS_START_5    float64
WEEKDAY_APPR_PROCESS_START_6    float64
Length: 111, dtype: object
```

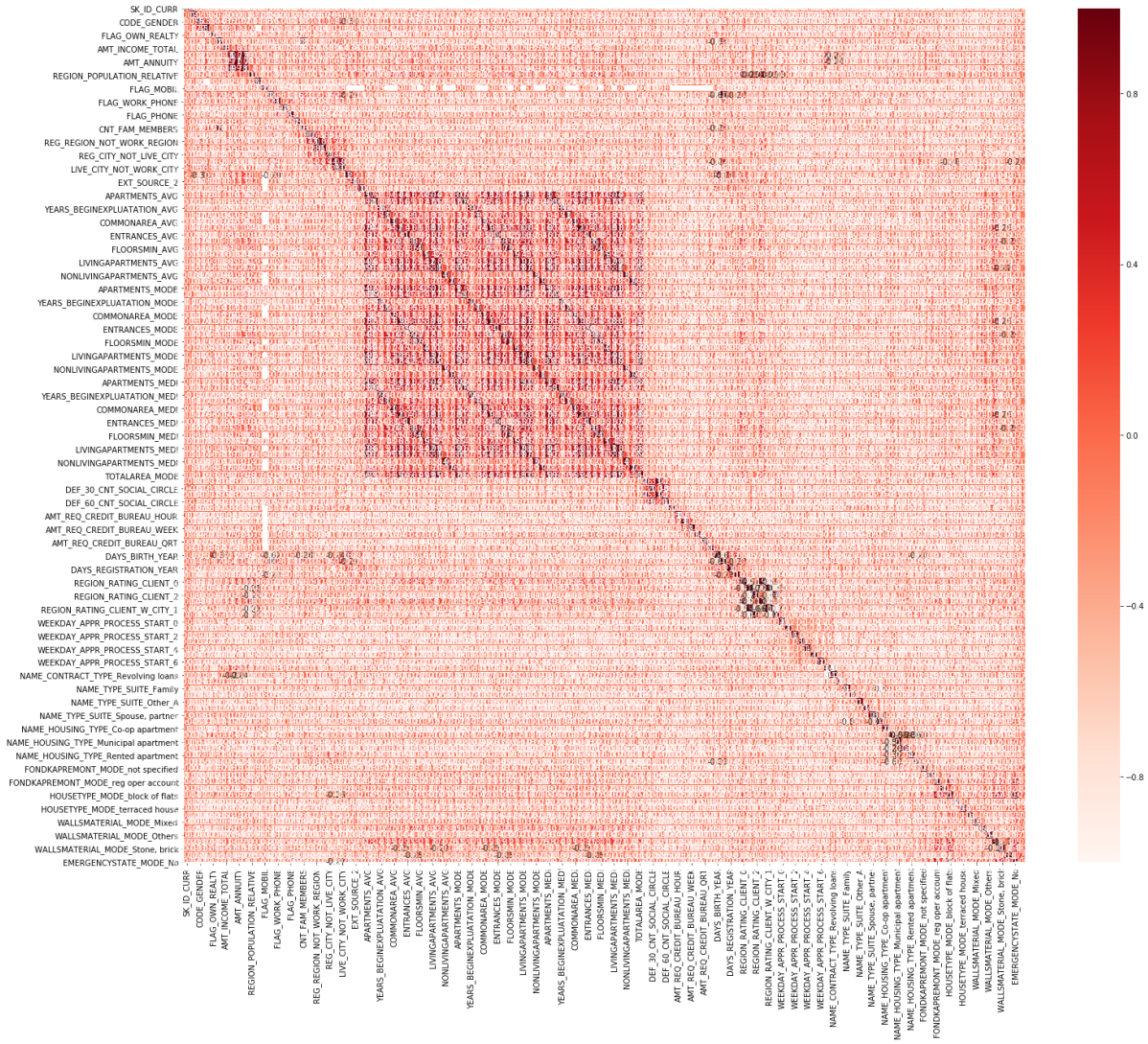
Do hot encoding for all the object type

In [87]:
```
for col in df_init.dtypes[df_init.dtypes == 'object'].index:
    for_dummy = df_init.pop(col)
    df_init = pd.concat([df_init, pd.get_dummies(for_dummy, prefix=col)], axis
=1)
```

In [88]:
```
df_init.dtypes
```

Out[88]:
```
SK_ID_CURR                          int64
TARGET                              int64
CODE_GENDER                          bool
FLAG_OWN_CAR                         bool
FLAG_OWN_REALTY                      bool
                                     ...
WALLSMATERIAL_MODE_Panel            uint8
WALLSMATERIAL_MODE_Stone, brick     uint8
WALLSMATERIAL_MODE_Wooden           uint8
EMERGENCYSTATE_MODE_No              uint8
EMERGENCYSTATE_MODE_Yes             uint8
Length: 135, dtype: object
```

In [89]:
```
mpl.rcParams['interactive'] == True
#Using Pearson Correlation
plt.figure(figsize=(24,20))
cor = df_init.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.savefig('Correlation_Pearson')
plt.show()
```



In [90]:
```
# Find correlations with the target and sort
correlations = df_init.corr()['TARGET'].sort_values()

# Display correlations
print('Most Positive Correlations:\n', correlations.tail(15))
print('\nMost Negative Correlations:\n', correlations.head(15))
```

```
Most Positive Correlations:
 NAME_HOUSING_TYPE_With parents    0.029966
NAME_CONTRACT_TYPE_Cash loans     0.030896
DEF_60_CNT_SOCIAL_CIRCLE          0.031276
DEF_30_CNT_SOCIAL_CIRCLE          0.032248
LIVE_CITY_NOT_WORK_CITY           0.032518
OWN_CAR_AGE                       0.037612
REG_CITY_NOT_LIVE_CITY            0.044395
DAYS_EMPLOYED_YEAR                0.044932
FLAG_EMP_PHONE                    0.045982
REGION_RATING_CLIENT_2            0.048029
REGION_RATING_CLIENT_W_CITY_2     0.049847
REG_CITY_NOT_WORK_CITY            0.050994
CODE_GENDER                       0.054704
DAYS_LAST_PHONE_CHANGE            0.055218
TARGET                            1.000000
Name: TARGET, dtype: float64

Most Negative Correlations:
 EXT_SOURCE_3                   -0.178919
EXT_SOURCE_2                   -0.160472
EXT_SOURCE_1                   -0.155317
DAYS_BIRTH_YEAR                -0.078239
DAYS_ID_PUBLISH_YEAR           -0.051457
FLOORSMAX_AVG                  -0.044003
FLOORSMAX_MEDI                 -0.043768
FLOORSMAX_MODE                 -0.043226
EMERGENCYSTATE_MODE_No         -0.042201
DAYS_REGISTRATION_YEAR         -0.041974
REGION_RATING_CLIENT_W_CITY_0  -0.041945
REGION_RATING_CLIENT_0         -0.040830
HOUSETYPE_MODE_block of flats  -0.040594
AMT_GOODS_PRICE                -0.039645
REGION_POPULATION_RELATIVE     -0.037227
Name: TARGET, dtype: float64
```

Analyse Bureau.csv

In [91]:
```
df_bureau = pd.read_csv(r"C:\Users\mamta\MMAI 2020\MMAI823_AI in Finance\Team
 Assignments\Team Project\Home Credit Risk Dataset\bureau.csv",sep=',')

list(df_bureau)
df_bureau.describe().transpose()
```

Out[91]:

	count	mean	std	min	25%
SK_ID_CURR	1716428.0	2.782149e+05	1.029386e+05	100001.000	188866.75
SK_ID_BUREAU	1716428.0	5.924434e+06	5.322657e+05	5000000.000	5463953.75
DAYS_CREDIT	1716428.0	-1.142108e+03	7.951649e+02	-2922.000	-1666.00
CREDIT_DAY_OVERDUE	1716428.0	8.181666e-01	3.654443e+01	0.000	0.00
DAYS_CREDIT_ENDDATE	1610875.0	5.105174e+02	4.994220e+03	-42060.000	-1138.00
DAYS_ENDDATE_FACT	1082775.0	-1.017437e+03	7.140106e+02	-42023.000	-1489.00
AMT_CREDIT_MAX_OVERDUE	591940.0	3.825418e+03	2.060316e+05	0.000	0.00
CNT_CREDIT_PROLONG	1716428.0	6.410406e-03	9.622391e-02	0.000	0.00
AMT_CREDIT_SUM	1716415.0	3.549946e+05	1.149811e+06	0.000	51300.00
AMT_CREDIT_SUM_DEBT	1458759.0	1.370851e+05	6.774011e+05	-4705600.320	0.00
AMT_CREDIT_SUM_LIMIT	1124648.0	6.229515e+03	4.503203e+04	-586406.115	0.00
AMT_CREDIT_SUM_OVERDUE	1716428.0	3.791276e+01	5.937650e+03	0.000	0.00
DAYS_CREDIT_UPDATE	1716428.0	-5.937483e+02	7.207473e+02	-41947.000	-908.00
AMT_ANNUITY	489637.0	1.571276e+04	3.258269e+05	0.000	0.00

In [92]:
```
df_bureau.head()
```

Out[92]:

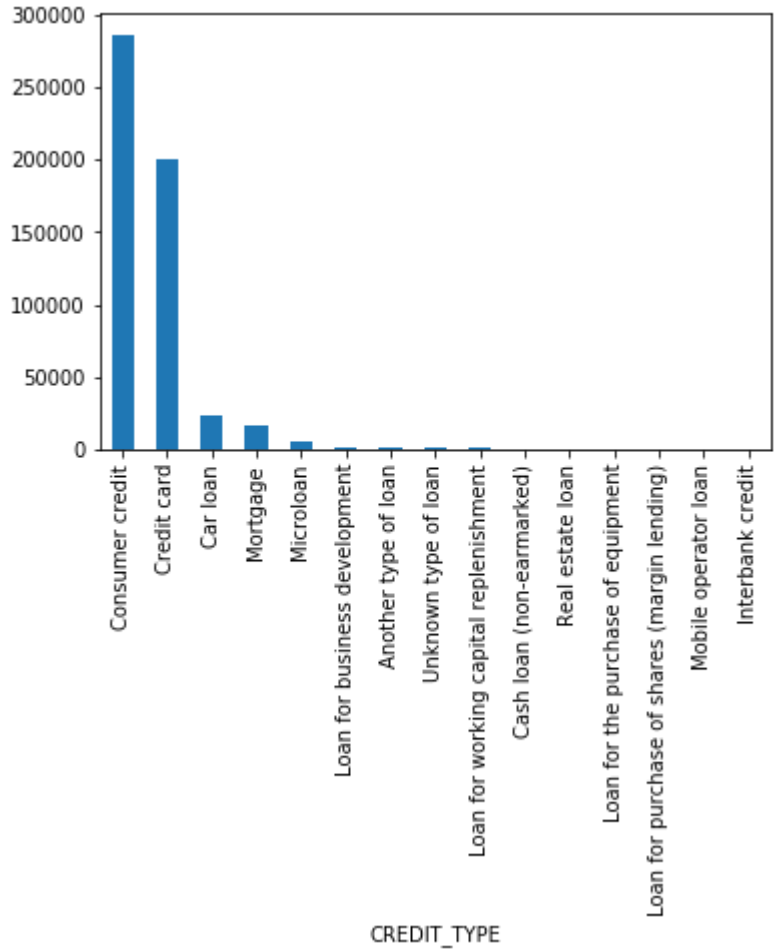
	SK_ID_CURR	SK_ID_BUREAU	CREDIT_ACTIVE	CREDIT_CURRENCY	DAYS_CREDIT	CREDIT_
0	215354	5714462	Closed	currency 1	-497	
1	215354	5714463	Active	currency 1	-208	
2	215354	5714464	Active	currency 1	-203	
3	215354	5714465	Active	currency 1	-203	
4	215354	5714466	Active	currency 1	-629	

Analyse Categorical Variables

Analyze Credit_type column

In [93]:
```
credit_type_count = df_bureau.groupby('CREDIT_TYPE')['SK_ID_CURR'].nunique()
credit_type_count = credit_type_count.sort_values(ascending=False)
credit_type_count.plot(kind='bar')
credit_type_count
```

Out[93]:
```
CREDIT_TYPE
Consumer credit                                 286669
Credit card                                     199965
Car loan                                         22796
Mortgage                                         16854
Microloan                                         4565
Loan for business development                     1599
Another type of loan                               962
Unknown type of loan                               497
Loan for working capital replenishment             414
Cash loan (non-earmarked)                           51
Real estate loan                                    27
Loan for the purchase of equipment                  18
Loan for purchase of shares (margin lending)         4
Mobile operator loan                                 1
Interbank credit                                     1
Name: SK_ID_CURR, dtype: int64
```



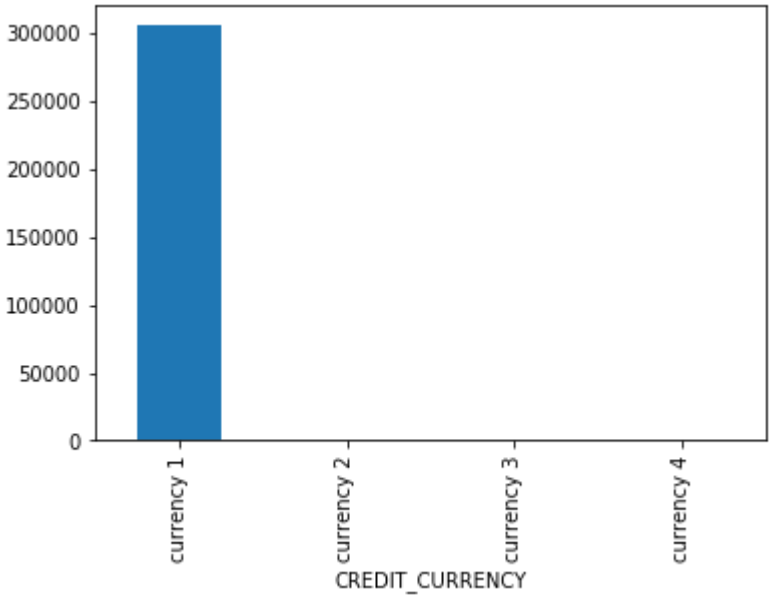
If credit_type_count is less than 900 combine them into 'Other type of loan'

In [94]:
```
m = df_bureau.CREDIT_TYPE.isin(credit_type_count.index[credit_type_count<900])
df_bureau.loc[m, 'CREDIT_TYPE'] = 'Other type of loan'
```

Visualize Credit_Currency

In [95]:
```
credit_currency_count = df_bureau.groupby('CREDIT_CURRENCY')['SK_ID_CURR'].nun
ique()
credit_currency_count = credit_currency_count.sort_values(ascending=False)
credit_currency_count.plot(kind='bar')
credit_currency_count
```

Out[95]:
```
CREDIT_CURRENCY
currency 1    305773
currency 2       970
currency 3       157
currency 4        10
Name: SK_ID_CURR, dtype: int64
```



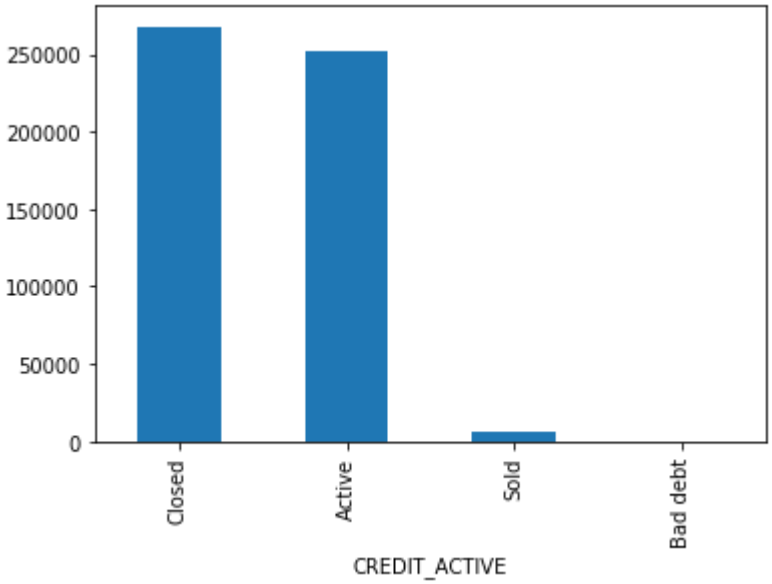
Drop Credit_currency since most of the records have same value

In [96]:
```
df_bureau = df_bureau.drop(["CREDIT_CURRENCY"], axis=1)
```

Analyze CREDIT_ACTIVE column

In [97]:
```
# Visualize credit_active
credit_active_count = df_bureau.groupby('CREDIT_ACTIVE')['SK_ID_CURR'].nunique
()
credit_active_count = credit_active_count.sort_values(ascending=False)
credit_active_count.plot(kind='bar')
credit_active_count
```

Out[97]:
```
CREDIT_ACTIVE
Closed      267925
Active      251815
Sold          6021
Bad debt        21
Name: SK_ID_CURR, dtype: int64
```



One-Hot encode categorical variables

In [98]:
```
categorical = pd.get_dummies(df_bureau.select_dtypes('object'))
categorical['SK_ID_CURR'] = df_bureau['SK_ID_CURR']
categorical.head()
```

Out[98]:

	CREDIT_ACTIVE_Active	CREDIT_ACTIVE_Bad debt	CREDIT_ACTIVE_Closed	CREDIT_ACTIVE_Sold
0	0	0	1	0
1	1	0	0	0
2	1	0	0	0
3	1	0	0	0
4	1	0	0	0

Aggregate Categorical variables

In [99]:
```
categorical_grouped = categorical.groupby('SK_ID_CURR').agg(['sum', 'mean'])
categorical_grouped.head()
```

Out[99]:

	CREDIT_ACTIVE_Active		CREDIT_ACTIVE_Bad debt		CREDIT_ACTIVE_Closed		CREDIT_AC
	sum	mean	sum	mean	sum	mean	sum
SK_ID_CURR							
100001	3	0.428571	0	0.0	4	0.571429	0
100002	2	0.250000	0	0.0	6	0.750000	0
100003	1	0.250000	0	0.0	3	0.750000	0
100004	0	0.000000	0	0.0	2	1.000000	0
100005	2	0.666667	0	0.0	1	0.333333	0

5 rows × 24 columns

In [100]:
```
# Merge the 'categorical_grouped' column with application_train file using the
'SK_ID_CURR' column as key
# Add column names

group_var = 'SK_ID_CURR'

# Need to create new column names
columns = []

# Iterate through the variables names
for var in categorical_grouped.columns.levels[0]:
    # Skip the grouping variable
    if var != group_var:
        # Iterate through the stat names
        for stat in ['count', 'count_norm']:
            # Make a new column name for the variable and stat
            columns.append('%s_%s' % (var, stat))

#  Rename the columns
categorical_grouped.columns = columns

categorical_grouped.head()
```

Out[100]:

SK_ID_CURR	CREDIT_ACTIVE_Active_count	CREDIT_ACTIVE_Active_count_norm	CREDIT_ACTIVE debt_count
100001	3	0.428571	
100002	2	0.250000	
100003	1	0.250000	
100004	0	0.000000	
100005	2	0.666667	

5 rows × 24 columns

Analyze numerical columns

Get the count of all previous loans for the applicant

In [101]:
```
# Merge the 'previous_loan_counts' column with application_train table using
 'SK_ID_CURR'
previous_loan_counts = df_bureau.groupby('SK_ID_CURR', as_index=False)['SK_ID_
BUREAU'].count().rename(columns = {'SK_ID_BUREAU': 'previous_loan_counts'})
previous_loan_counts.head()
```

Out[101]:

	SK_ID_CURR	previous_loan_counts
0	100001	7
1	100002	8
2	100003	4
3	100004	2
4	100005	3

Convert days to years and make them positive

In [102]:
```
df_bureau['YEARS_CREDIT'] =  round((- (df_bureau['DAYS_CREDIT'] / 365) ), 2)
df_bureau['CREDIT_YEAR_OVERDUE'] =  round((- (df_bureau['CREDIT_DAY_OVERDUE']
/ 365) ),2)
```

Drop rest of the days columns

In [103]:
```
df_bureau = df_bureau.drop(["DAYS_CREDIT", "CREDIT_DAY_OVERDUE", "DAYS_CREDIT_
ENDDATE", "DAYS_ENDDATE_FACT", "CNT_CREDIT_PROLONG", "DAYS_CREDIT_UPDATE"], ax
is=1)
```

Aggregate rest of the numerical values

In [104]:
```
bureau_agg = df_bureau.drop(columns = ['SK_ID_BUREAU']).groupby('SK_ID_CURR',
as_index = False).agg(['count', 'mean', 'max', 'min', 'sum']).reset_index()
bureau_agg.head()
```

Out[104]:

	SK_ID_CURR	AMT_CREDIT_MAX_OVERDUE				AMT_CREDIT_SUM			
		count	mean	max	min	sum	count	mean	max	
0	100001	0	NaN	NaN	NaN	0.000	7	207623.571429	378000.0	85
1	100002	5	1681.029	5043.645	0.0	8405.145	8	108131.945625	450000.0	
2	100003	4	0.000	0.000	0.0	0.000	4	254350.125000	810000.0	22
3	100004	1	0.000	0.000	0.0	0.000	2	94518.900000	94537.8	94
4	100005	1	0.000	0.000	0.0	0.000	3	219042.000000	568800.0	29

5 rows × 41 columns

In [105]:
```
# Assign column names. Merge 'bureau_agg' with 'application_train' using the
 'SK_ID_CURR' column as key
columns = ['SK_ID_CURR']

# Iterate through the variables names
for var in bureau_agg.columns.levels[0]:
    # Skip the id name
    if var != 'SK_ID_CURR':

        # Iterate through the stat names
        for stat in bureau_agg.columns.levels[1][:-1]:
            # Make a new column name for the variable and stat
            columns.append('bureau_%s_%s' % (var, stat))

bureau_agg.columns = columns
bureau_agg.head()
```

Out[105]:

	SK_ID_CURR	bureau_AMT_CREDIT_MAX_OVERDUE_count	bureau_AMT_CREDIT_MAX_OVERDU
0	100001	0	
1	100002	5	1
2	100003	4	
3	100004	1	
4	100005	1	

5 rows × 41 columns

Merging below mentioned aggregate tables (derived from BUREAU.csv) to the df_init dataset (derived from application_train.csv) 1) bureau_agg 2) previous_loan_counts 3) categorical_grouped

In [106]:
```
categorical_grouped.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 305811 entries, 100001 to 456255
Data columns (total 24 columns):
CREDIT_ACTIVE_Active_count                                305811 non-null uint8
CREDIT_ACTIVE_Active_count_norm                           305811 non-null float
64
CREDIT_ACTIVE_Bad debt_count                              305811 non-null uint8
CREDIT_ACTIVE_Bad debt_count_norm                         305811 non-null float
64
CREDIT_ACTIVE_Closed_count                                305811 non-null uint8
CREDIT_ACTIVE_Closed_count_norm                           305811 non-null float
64
CREDIT_ACTIVE_Sold_count                                  305811 non-null uint8
CREDIT_ACTIVE_Sold_count_norm                             305811 non-null float
64
CREDIT_TYPE_Another type of loan_count                    305811 non-null uint8
CREDIT_TYPE_Another type of loan_count_norm               305811 non-null float
64
CREDIT_TYPE_Car loan_count                                305811 non-null uint8
CREDIT_TYPE_Car loan_count_norm                           305811 non-null float
64
CREDIT_TYPE_Consumer credit_count                         305811 non-null uint8
CREDIT_TYPE_Consumer credit_count_norm                    305811 non-null float
64
CREDIT_TYPE_Credit card_count                             305811 non-null uint8
CREDIT_TYPE_Credit card_count_norm                        305811 non-null float
64
CREDIT_TYPE_Loan for business development_count           305811 non-null uint8
CREDIT_TYPE_Loan for business development_count_norm      305811 non-null float
64
CREDIT_TYPE_Microloan_count                               305811 non-null uint8
CREDIT_TYPE_Microloan_count_norm                          305811 non-null float
64
CREDIT_TYPE_Mortgage_count                                305811 non-null uint8
CREDIT_TYPE_Mortgage_count_norm                           305811 non-null float
64
CREDIT_TYPE_Other type of loan_count                      305811 non-null uint8
CREDIT_TYPE_Other type of loan_count_norm                 305811 non-null float
64
dtypes: float64(12), uint8(12)
memory usage: 33.8 MB
```

Merge Application train data with Bureau Data

In [107]:
```
# Join to the df_init dataframe with categorical_grouped
df_application_merged = df_init.merge(categorical_grouped, on = 'SK_ID_CURR',
how = 'left')
df_application_merged.head()
```

Out[107]:

	SK_ID_CURR	TARGET	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDR
0	100002	1	True	False	True	
1	100003	0	False	False	False	
2	100004	0	True	True	True	
3	100006	0	False	False	True	
4	100007	0	True	False	True	

5 rows × 159 columns

In [108]:
```
# Join to the df_application_merged dataframe with previous_loan_counts
df_application_merged = df_application_merged.merge(previous_loan_counts, on =
'SK_ID_CURR', how = 'left')
df_application_merged.head()
```

Out[108]:

	SK_ID_CURR	TARGET	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDR
0	100002	1	True	False	True	
1	100003	0	False	False	False	
2	100004	0	True	True	True	
3	100006	0	False	False	True	
4	100007	0	True	False	True	

5 rows × 160 columns

In [109]:
```
# Join to the df_application_merged dataframe with bureau_agg
df_application_merged = df_application_merged.merge(bureau_agg, on = 'SK_ID_CU
RR', how = 'left')
df_application_merged.head()
```

Out[109]:

	SK_ID_CURR	TARGET	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDR
0	100002	1	True	False	True	
1	100003	0	False	False	False	
2	100004	0	True	True	True	
3	100006	0	False	False	True	
4	100007	0	True	False	True	

5 rows × 200 columns

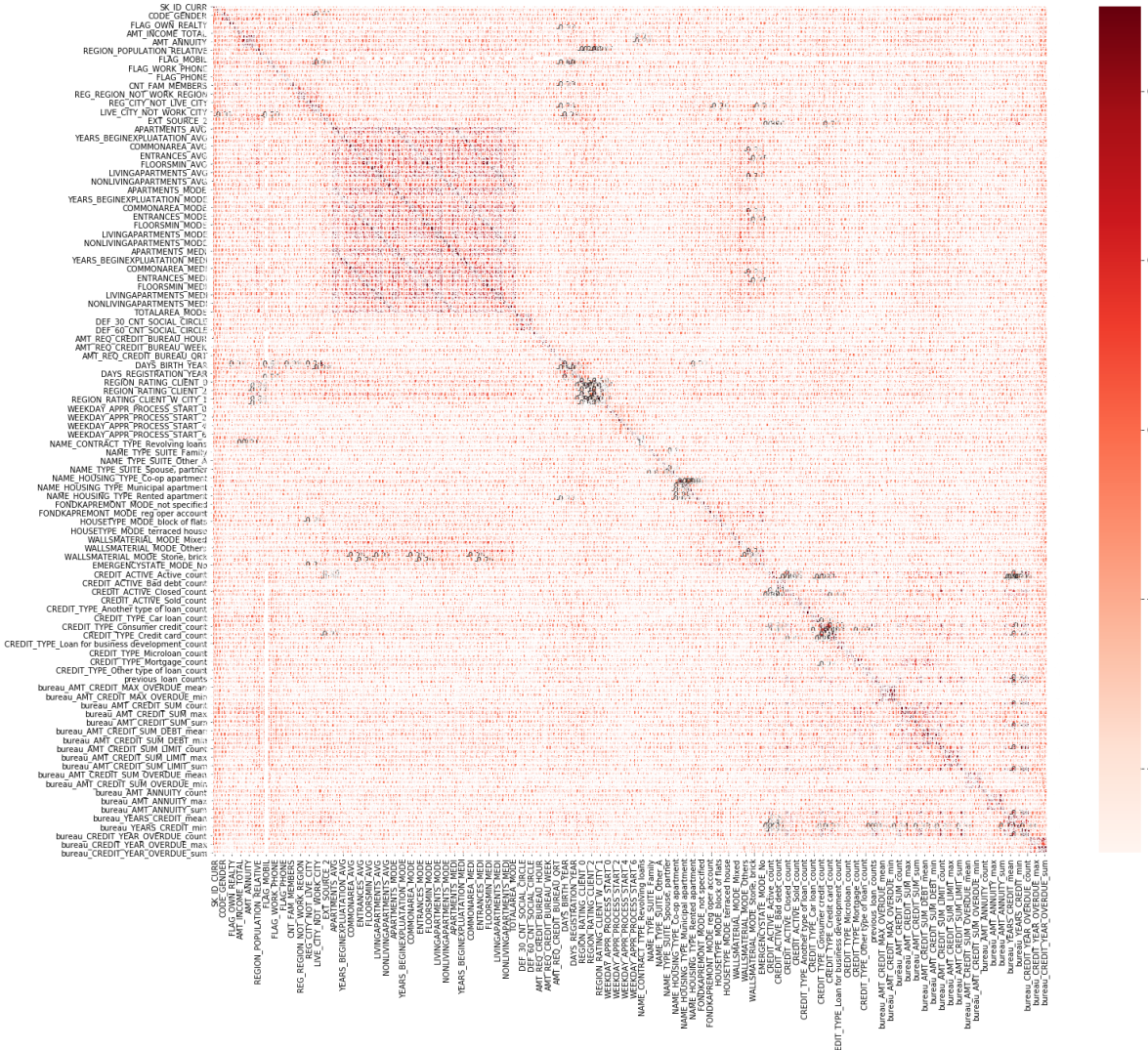
In [110]:
```
df_application_merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 307511 entries, 0 to 307510
Columns: 200 entries, SK_ID_CURR to bureau_CREDIT_YEAR_OVERDUE_sum
dtypes: bool(5), category(5), float64(145), int32(1), int64(13), uint8(31)
memory usage: 386.2 MB
```

Run correlation matrix for our final dataset

In [111]:

```
mpl.rcParams['interactive'] == True
#Using Pearson Correlation
plt.figure(figsize=(24,20))
cor = df_application_merged.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.savefig('Correlation_Pearson')
plt.show()
```



In [112]:
```
# Find correlations with the target and sort
correlations = df_application_merged.corr()['TARGET'].sort_values()

# Display correlations
print('Most Positive Correlations:\n', correlations.tail(15))
print('\nMost Negative Correlations:\n', correlations.head(15))
```

```
Most Positive Correlations:
 CREDIT_TYPE_Credit card_count_norm    0.034684
CREDIT_TYPE_Credit card_count          0.034818
OWN_CAR_AGE                            0.037612
REG_CITY_NOT_LIVE_CITY                 0.044395
CREDIT_TYPE_Microloan_count_norm       0.044439
DAYS_EMPLOYED_YEAR                     0.044932
FLAG_EMP_PHONE                         0.045982
REGION_RATING_CLIENT_2                 0.048029
REGION_RATING_CLIENT_W_CITY_2          0.049847
REG_CITY_NOT_WORK_CITY                 0.050994
CODE_GENDER                            0.054704
DAYS_LAST_PHONE_CHANGE                 0.055218
CREDIT_ACTIVE_Active_count             0.067128
CREDIT_ACTIVE_Active_count_norm        0.077356
TARGET                                 1.000000
Name: TARGET, dtype: float64

Most Negative Correlations:
 EXT_SOURCE_3                         -0.178919
EXT_SOURCE_2                         -0.160472
EXT_SOURCE_1                         -0.155317
bureau_YEARS_CREDIT_mean             -0.089731
CREDIT_ACTIVE_Closed_count_norm      -0.079369
DAYS_BIRTH_YEAR                      -0.078239
bureau_YEARS_CREDIT_max              -0.075249
DAYS_ID_PUBLISH_YEAR                 -0.051457
bureau_YEARS_CREDIT_min              -0.049785
FLOORSMAX_AVG                        -0.044003
FLOORSMAX_MEDI                       -0.043768
FLOORSMAX_MODE                       -0.043226
EMERGENCYSTATE_MODE_No               -0.042201
bureau_YEARS_CREDIT_sum              -0.042001
DAYS_REGISTRATION_YEAR               -0.041974
Name: TARGET, dtype: float64
```

Move target column to the end

In [113]:
```
clist = list(df_application_merged.columns)
clist_new = clist[:1]+clist[2:]+clist[1:2]
#clist
```

In [114]:
```
# Pass the new list to the DataFrame - like a key list in a dict
df_application_merged_1 = df_application_merged[clist_new]
df_application_merged_1
```

Out[114]:

	SK_ID_CURR	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN
0	100002	True	False	True	0
1	100003	False	False	False	0
2	100004	True	True	True	0
3	100006	False	False	True	0
4	100007	True	False	True	0
...	...	...	...	...	...
307506	456251	True	False	False	0
307507	456252	False	False	True	0
307508	456253	False	False	True	0
307509	456254	False	False	True	0
307510	456255	False	False	False	0

307511 rows × 200 columns

Keep all features which are more correlated with the TARGET

In [115]:
```
df_features_select = df_application_merged_1[["SK_ID_CURR","NAME_INCOME_TYPE",
"ORGANIZATION_TYPE",
"NAME_FAMILY_STATUS",
"NAME_EDUCATION_TYPE",
"CREDIT_TYPE_Credit card_count_norm",
"CREDIT_TYPE_Credit card_count",
"OWN_CAR_AGE",
"OCCUPATION_TYPE",
"REG_CITY_NOT_LIVE_CITY",
"CREDIT_TYPE_Microloan_count_norm",
"DAYS_EMPLOYED_YEAR",
"FLAG_EMP_PHONE",
"REGION_RATING_CLIENT_2",
"REGION_RATING_CLIENT_W_CITY_2",
"REG_CITY_NOT_WORK_CITY",
"CODE_GENDER",
"DAYS_LAST_PHONE_CHANGE",
"CREDIT_ACTIVE_Active_count",
"CREDIT_ACTIVE_Active_count_norm",
"EXT_SOURCE_3",
"EXT_SOURCE_2",
"EXT_SOURCE_1",
"bureau_YEARS_CREDIT_mean",
"CREDIT_ACTIVE_Closed_count_norm",
"DAYS_BIRTH_YEAR",
"bureau_YEARS_CREDIT_max",
"DAYS_ID_PUBLISH_YEAR",
"bureau_YEARS_CREDIT_min",
"FLOORSMAX_AVG",
"FLOORSMAX_MEDI",
"FLOORSMAX_MODE",
"EMERGENCYSTATE_MODE_No",
"bureau_YEARS_CREDIT_sum",
"DAYS_REGISTRATION_YEAR",
"TARGET"]]
```

Check null value in new dataset

In [116]:
```
np.where(pd.isnull(df_features_select))
```

Out[116]:
```
(array([     0,      1,      1, ..., 307509, 307509, 307510], dtype=int64),
 array([ 7,  7, 20, ...,  7, 22,  7], dtype=int64))
```

In [117]:

```
def missing_zero_values_table(df_features_select):
        zero_val = (df_features_select == 0.00).astype(int).sum(axis=0)
        mis_val = df_features_select.isnull().sum()
        mis_val_percent = 100 * df_features_select.isnull().sum() / len(df_fea
tures_select)
        mz_table = pd.concat([zero_val, mis_val, mis_val_percent], axis=1)
        mz_table = mz_table.rename(
        columns = {0 : 'Zero Values', 1 : 'Missing Values', 2 : '% of Total Va
lues'})
        mz_table['Total Zero Missing Values'] = mz_table['Zero Values'] + mz_t
able['Missing Values']
        mz_table['% Total Zero Missing Values'] = 100 * mz_table['Total Zero M
issing Values'] / len(df_features_select)
        mz_table['Data Type'] = df_features_select.dtypes
        mz_table = mz_table[
            mz_table.iloc[:,1] != 0].sort_values(
        '% of Total Values', ascending=False).round(1)
        print ("Your selected dataframe has " + str(df_features_select.shape[1
]) + " columns and " + str(df_features_select.shape[0]) + " Rows.\n"
            "There are " + str(mz_table.shape[0]) +
              " columns that have missing values.")
#         mz_table.to_excel('missing_and_zero_values.xlsx', freeze_panes=(1,
0), index = False)
        return mz_table

missing_zero_values_table(df_features_select)
```

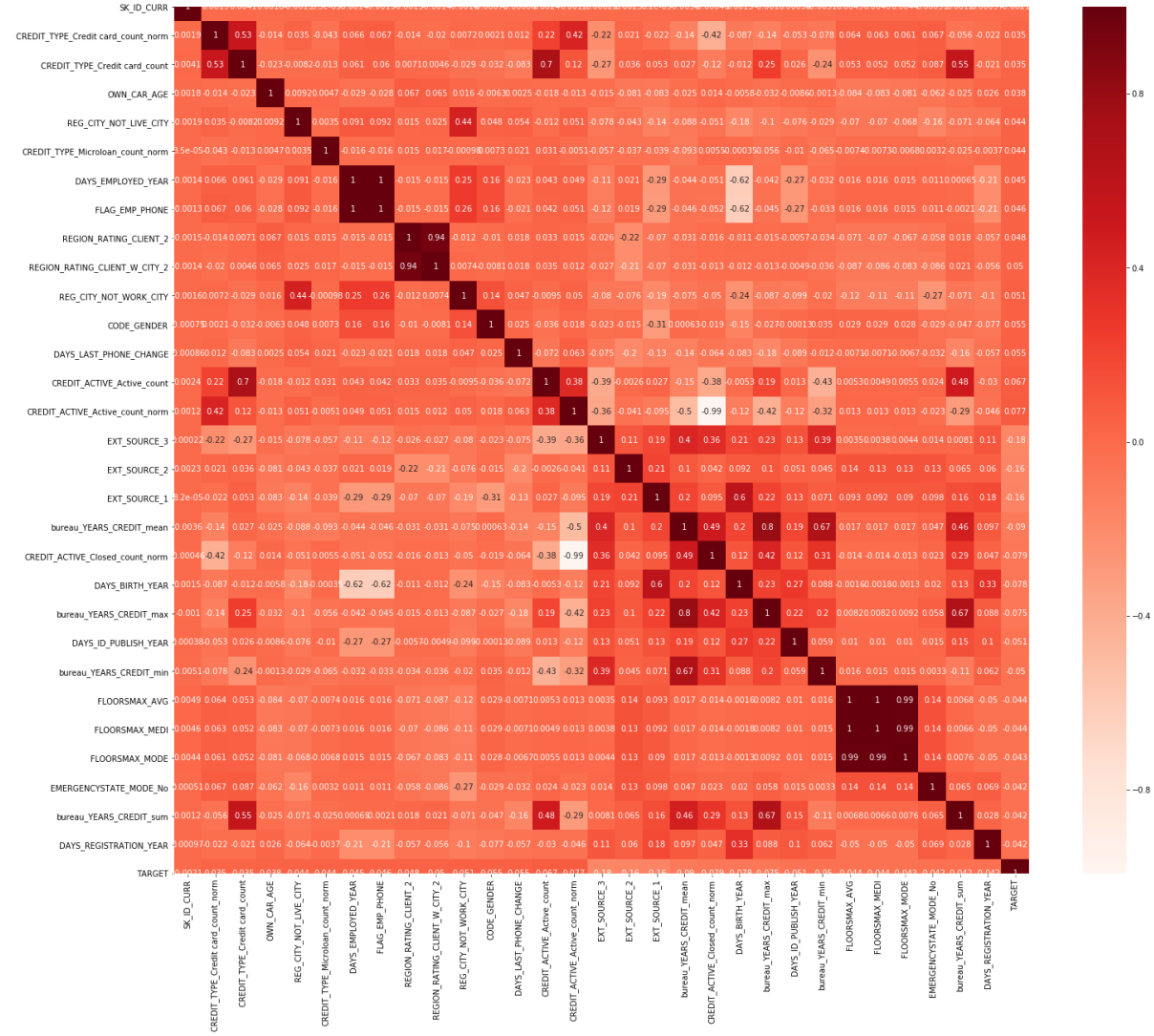
```
Your selected dataframe has 36 columns and 307511 Rows.
There are 18 columns that have missing values.
```

Out[117]:

	Zero Values	Missing Values	% of Total Values	Total Zero Missing Values	% Total Zero Missing Values	Data Type
OWN_CAR_AGE	2134	202929	66.0	205063	66.7	float64
EXT_SOURCE_1	0	173378	56.4	173378	56.4	float64
FLOORSMAX_MODE	3415	153020	49.8	156435	50.9	float64
FLOORSMAX_MEDI	2995	153020	49.8	156015	50.7	float64
FLOORSMAX_AVG	2938	153020	49.8	155958	50.7	float64
EXT_SOURCE_3	0	60965	19.8	60965	19.8	float64
bureau_YEARS_CREDIT_mean	1	44020	14.3	44021	14.3	float64
bureau_YEARS_CREDIT_min	17	44020	14.3	44037	14.3	float64
bureau_YEARS_CREDIT_max	1	44020	14.3	44021	14.3	float64
CREDIT_ACTIVE_Closed_count_norm	33326	44020	14.3	77346	25.2	float64
CREDIT_TYPE_Credit card_count_norm	91449	44020	14.3	135469	44.1	float64
CREDIT_TYPE_Credit card_count	91449	44020	14.3	135469	44.1	float64
CREDIT_ACTIVE_Active_count_norm	46341	44020	14.3	90361	29.4	float64
CREDIT_ACTIVE_Active_count	46341	44020	14.3	90361	29.4	float64
CREDIT_TYPE_Microloan_count_norm	259992	44020	14.3	304012	98.9	float64
bureau_YEARS_CREDIT_sum	1	44020	14.3	44021	14.3	float64
EXT_SOURCE_2	0	660	0.2	660	0.2	float64
DAYS_LAST_PHONE_CHANGE	37672	1	0.0	37673	12.3	float64

Check for Multi colinearity of the features

In [118]:
```
mpl.rcParams['interactive'] == True
#Using Pearson Correlation
plt.figure(figsize=(24,20))
cor = df_features_select.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.savefig('Correlation_Pearson')
plt.show()
```



We see that there are features which are highly correlated to each other. Drop higly correlated features

In [119]:
```
df_features_select = df_application_merged_1[["SK_ID_CURR",
    "NAME_INCOME_TYPE",
"ORGANIZATION_TYPE",
"NAME_FAMILY_STATUS",
"NAME_EDUCATION_TYPE",
"CREDIT_TYPE_Credit card_count",
"OWN_CAR_AGE",
"OCCUPATION_TYPE",
"REG_CITY_NOT_LIVE_CITY",
"CREDIT_TYPE_Microloan_count_norm",
"DAYS_EMPLOYED_YEAR",
"REGION_RATING_CLIENT_2",
"REG_CITY_NOT_WORK_CITY",
"CODE_GENDER",
"DAYS_LAST_PHONE_CHANGE",
"CREDIT_ACTIVE_Active_count",
"EXT_SOURCE_3",
"EXT_SOURCE_2",
"EXT_SOURCE_1",
"bureau_YEARS_CREDIT_mean",
"DAYS_BIRTH_YEAR",
"DAYS_ID_PUBLISH_YEAR",
"FLOORSMAX_AVG",
"EMERGENCYSTATE_MODE_No",
"bureau_YEARS_CREDIT_sum",
"DAYS_REGISTRATION_YEAR",
"TARGET"]]
```

Get number of missing columns

In [120]:
```
df_features_select.isnull().sum()
```

Out[120]:
```
SK_ID_CURR                          0
NAME_INCOME_TYPE                    0
ORGANIZATION_TYPE                   0
NAME_FAMILY_STATUS                  0
NAME_EDUCATION_TYPE                 0
CREDIT_TYPE_Credit card_count       44020
OWN_CAR_AGE                         202929
OCCUPATION_TYPE                     0
REG_CITY_NOT_LIVE_CITY              0
CREDIT_TYPE_Microloan_count_norm    44020
DAYS_EMPLOYED_YEAR                  0
REGION_RATING_CLIENT_2              0
REG_CITY_NOT_WORK_CITY              0
CODE_GENDER                         0
DAYS_LAST_PHONE_CHANGE              1
CREDIT_ACTIVE_Active_count          44020
EXT_SOURCE_3                        60965
EXT_SOURCE_2                        660
EXT_SOURCE_1                        173378
bureau_YEARS_CREDIT_mean            44020
DAYS_BIRTH_YEAR                     0
DAYS_ID_PUBLISH_YEAR                0
FLOORSMAX_AVG                       153020
EMERGENCYSTATE_MODE_No              0
bureau_YEARS_CREDIT_sum             44020
DAYS_REGISTRATION_YEAR              0
TARGET                              0
dtype: int64
```

In [121]:
```
df_features_select.head()
```

Out[121]:

	SK_ID_CURR	NAME_INCOME_TYPE	ORGANIZATION_TYPE	NAME_FAMILY_STATUS	NAME_ED
0	100002	Working	Business	Single / not married	Secor
1	100003	State servant	School	Married	
2	100004	Working	Government	Single / not married	Secor
3	100006	Working	Business	Civil marriage	Secor
4	100007	Working	Religion	Single / not married	Secor

5 rows × 27 columns

In [122]:
```
df_features_select.dtypes
```

Out[122]:
```
SK_ID_CURR                          int64
NAME_INCOME_TYPE                 category
ORGANIZATION_TYPE                category
NAME_FAMILY_STATUS               category
NAME_EDUCATION_TYPE              category
CREDIT_TYPE_Credit card_count     float64
OWN_CAR_AGE                       float64
OCCUPATION_TYPE                  category
REG_CITY_NOT_LIVE_CITY               bool
CREDIT_TYPE_Microloan_count_norm  float64
DAYS_EMPLOYED_YEAR                float64
REGION_RATING_CLIENT_2            float64
REG_CITY_NOT_WORK_CITY               bool
CODE_GENDER                          bool
DAYS_LAST_PHONE_CHANGE            float64
CREDIT_ACTIVE_Active_count        float64
EXT_SOURCE_3                      float64
EXT_SOURCE_2                      float64
EXT_SOURCE_1                      float64
bureau_YEARS_CREDIT_mean          float64
DAYS_BIRTH_YEAR                   float64
DAYS_ID_PUBLISH_YEAR              float64
FLOORSMAX_AVG                     float64
EMERGENCYSTATE_MODE_No              uint8
bureau_YEARS_CREDIT_sum           float64
DAYS_REGISTRATION_YEAR            float64
TARGET                              int64
dtype: object
```

Convert REGION_RATING_CLIENT_2 to boolean

In [123]:
```
df_region_rating = df_features_select.groupby('REGION_RATING_CLIENT_2')['SK_ID
_CURR'].nunique()
df_region_rating
```

Out[123]:
```
REGION_RATING_CLIENT_2
0.0    259181
1.0     48330
Name: SK_ID_CURR, dtype: int64
```

Convert to boolean

In [124]:
```
df_features_select['REGION_RATING_CLIENT_2'] = np.where(df_features_select['RE
GION_RATING_CLIENT_2'] == 1, True, False)
```

Reading Previous Application file

In [125]:
```
print('Reading the data....', end='')
previous_applicaton = pd.read_csv(r'C:\Users\mamta\MMAI 2020\MMAI823_AI in Finance\Team Assignments\Team Project\Home Credit Risk Dataset\previous_application.csv',sep=',')
print('done!!!')
print('The shape of data:',previous_applicaton.shape)
print('First 5 rows of data:')
previous_applicaton.head()
```

```
Reading the data....done!!!
The shape of data: (1670214, 37)
First 5 rows of data:
```

Out[125]:

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AM
0	2030495	271877	Consumer loans	1730.430	17145.0	
1	2802425	108129	Cash loans	25188.615	607500.0	
2	2523466	122040	Cash loans	15060.735	112500.0	
3	2819243	176158	Cash loans	47041.335	450000.0	
4	1784265	202054	Cash loans	31924.395	337500.0	

5 rows × 37 columns

In [126]:
```
list(previous_applicaton.columns)
```

Out[126]:
```
['SK_ID_PREV',
 'SK_ID_CURR',
 'NAME_CONTRACT_TYPE',
 'AMT_ANNUITY',
 'AMT_APPLICATION',
 'AMT_CREDIT',
 'AMT_DOWN_PAYMENT',
 'AMT_GOODS_PRICE',
 'WEEKDAY_APPR_PROCESS_START',
 'HOUR_APPR_PROCESS_START',
 'FLAG_LAST_APPL_PER_CONTRACT',
 'NFLAG_LAST_APPL_IN_DAY',
 'RATE_DOWN_PAYMENT',
 'RATE_INTEREST_PRIMARY',
 'RATE_INTEREST_PRIVILEGED',
 'NAME_CASH_LOAN_PURPOSE',
 'NAME_CONTRACT_STATUS',
 'DAYS_DECISION',
 'NAME_PAYMENT_TYPE',
 'CODE_REJECT_REASON',
 'NAME_TYPE_SUITE',
 'NAME_CLIENT_TYPE',
 'NAME_GOODS_CATEGORY',
 'NAME_PORTFOLIO',
 'NAME_PRODUCT_TYPE',
 'CHANNEL_TYPE',
 'SELLERPLACE_AREA',
 'NAME_SELLER_INDUSTRY',
 'CNT_PAYMENT',
 'NAME_YIELD_GROUP',
 'PRODUCT_COMBINATION',
 'DAYS_FIRST_DRAWING',
 'DAYS_FIRST_DUE',
 'DAYS_LAST_DUE_1ST_VERSION',
 'DAYS_LAST_DUE',
 'DAYS_TERMINATION',
 'NFLAG_INSURED_ON_APPROVAL']
```

Drop columns which doesn't seem useful based on experience

In [127]:
```
previous_applicaton.drop(['WEEKDAY_APPR_PROCESS_START',
 'HOUR_APPR_PROCESS_START','NFLAG_LAST_APPL_IN_DAY','RATE_DOWN_PAYMENT',
 'RATE_INTEREST_PRIMARY',
 'RATE_INTEREST_PRIVILEGED',
 'NAME_CASH_LOAN_PURPOSE',
 'NAME_CONTRACT_STATUS',
 'DAYS_DECISION',
 'NAME_PAYMENT_TYPE','NAME_TYPE_SUITE',
 'NAME_CLIENT_TYPE',
 'NAME_GOODS_CATEGORY',
 'NAME_PORTFOLIO',
 'NAME_PRODUCT_TYPE',
 'CHANNEL_TYPE',
 'SELLERPLACE_AREA',
 'NAME_SELLER_INDUSTRY','NAME_YIELD_GROUP',
 'PRODUCT_COMBINATION',
 'DAYS_FIRST_DRAWING',
 'DAYS_FIRST_DUE',
 'DAYS_LAST_DUE_1ST_VERSION',
 'DAYS_LAST_DUE',
 'DAYS_TERMINATION',
 'NFLAG_INSURED_ON_APPROVAL'], axis=1, inplace=True)
```

In [128]:
```
previous_applicaton.dtypes
```

Out[128]:
```
SK_ID_PREV                     int64
SK_ID_CURR                     int64
NAME_CONTRACT_TYPE            object
AMT_ANNUITY                  float64
AMT_APPLICATION              float64
AMT_CREDIT                   float64
AMT_DOWN_PAYMENT             float64
AMT_GOODS_PRICE              float64
FLAG_LAST_APPL_PER_CONTRACT   object
CODE_REJECT_REASON            object
CNT_PAYMENT                  float64
dtype: object
```

Convert Name contract type into categorical

In [129]:
```
previous_applicaton.NAME_CONTRACT_TYPE.unique()
```

Out[129]:
```
array(['Consumer loans', 'Cash loans', 'Revolving loans', 'XNA'],
      dtype=object)
```

In [130]:
```
previous_applicaton['NAME_CONTRACT_TYPE'] = previous_applicaton['NAME_CONTRACT
_TYPE'].astype('category')
```

Analyse Flag Last application per contract

In [131]:
```
previous_applicaton.FLAG_LAST_APPL_PER_CONTRACT.unique()
```

Out[131]:
```
array(['Y', 'N'], dtype=object)
```

Convert into categorical

In [132]:
```
previous_applicaton['FLAG_LAST_APPL_PER_CONTRACT'] = previous_applicaton['FLAG
_LAST_APPL_PER_CONTRACT'].astype('category')
```

Analyse CODE_REJECT_REASON

In [133]:
```
previous_applicaton.CODE_REJECT_REASON.unique()
```

Out[133]:
```
array(['XAP', 'HC', 'LIMIT', 'CLIENT', 'SCOFR', 'SCO', 'XNA', 'VERIF',
       'SYSTEM'], dtype=object)
```

Convert into categorical

In [134]:
```
previous_applicaton['CODE_REJECT_REASON'] = previous_applicaton['CODE_REJECT_R
EASON'].astype('category')
```

Joining with previous application file

In [135]:
```
# Number of previous applications per customer
grp = previous_applicaton[['SK_ID_CURR','SK_ID_PREV']].groupby(by=['SK_ID_CUR
R'])['SK_ID_PREV'].count().reset_index().rename(columns={'SK_ID_PREV':'PREV_AP
P_COUNT'})
application_bureau_prev = df_features_select.merge(grp, on =['SK_ID_CURR'], ho
w = 'left')
application_bureau_prev['PREV_APP_COUNT'] = application_bureau_prev['PREV_APP_
COUNT'].fillna(0)
# Combining numerical features
grp = previous_applicaton.drop('SK_ID_PREV', axis =1).groupby(by=['SK_ID_CURR'
]).mean().reset_index()
prev_columns = ['PREV_'+column if column != 'SK_ID_CURR' else column for colum
n in grp.columns ]
grp.columns = prev_columns
application_bureau_prev = application_bureau_prev.merge(grp, on =['SK_ID_CURR'
], how = 'left')
application_bureau_prev.update(application_bureau_prev[grp.columns].fillna(0))
```

In [136]:
```
# Combining categorical features
prev_categorical = pd.get_dummies(previous_applicaton.select_dtypes('category'
))
prev_categorical['SK_ID_CURR'] = previous_applicaton['SK_ID_CURR']
prev_categorical.head()
grp = prev_categorical.groupby('SK_ID_CURR').max().reset_index()
grp.columns = ['PREV_'+column if column != 'SK_ID_CURR' else column for column
in grp.columns]
application_bureau_prev = application_bureau_prev.merge(grp, on=['SK_ID_CURR'
], how='left')
application_bureau_prev.update(application_bureau_prev[grp.columns].fillna(0))
```

Reading POS Cash balance

In [137]:
```
print('Reading the data....', end='')
pos_cash = pd.read_csv(r'C:\Users\mamta\MMAI 2020\MMAI823_AI in Finance\Team A
ssignments\Team Project\Home Credit Risk Dataset\POS_CASH_balance.csv',sep=','
)
print('done!!!')
print('The shape of data:',pos_cash.shape)
print('First 5 rows of data:')
pos_cash.head()
```

```
Reading the data....done!!!
The shape of data: (10001358, 8)
First 5 rows of data:
```

Out[137]:

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	CNT_INSTALMENT	CNT_INSTALMENT_FUTUR
0	1803195	182943	-31	48.0	45
1	1715348	367990	-33	36.0	35
2	1784872	397406	-32	12.0	9
3	1903291	269225	-35	48.0	42
4	2341044	334279	-35	36.0	35

Dropping irrelevant columns

In [138]:
```
pos_cash.drop(['SK_DPD','SK_DPD_DEF'], axis=1, inplace=True)
```

Merging the columns

In [139]:
```
# Combining numerical features
grp = pos_cash.drop('SK_ID_PREV', axis =1).groupby(by=['SK_ID_CURR']).mean().r
eset_index()
prev_columns = ['POS_'+column if column != 'SK_ID_CURR' else column for column
in grp.columns ]
grp.columns = prev_columns
application_bureau_prev = application_bureau_prev.merge(grp, on =['SK_ID_CURR'
], how = 'left')
application_bureau_prev.update(application_bureau_prev[grp.columns].fillna(0))
# Combining categorical features
pos_cash_categorical = pd.get_dummies(pos_cash.select_dtypes('object'))
pos_cash_categorical['SK_ID_CURR'] = pos_cash['SK_ID_CURR']
grp = pos_cash_categorical.groupby('SK_ID_CURR').mean().reset_index()
grp.columns = ['POS_'+column if column != 'SK_ID_CURR' else column for column
in grp.columns]
application_bureau_prev = application_bureau_prev.merge(grp, on=['SK_ID_CURR'
], how='left')
application_bureau_prev.update(application_bureau_prev[grp.columns].fillna(0))
```

Reading installment payment file

In [140]:
```
print('Reading the data....', end='')
insta_payments = pd.read_csv(r'C:\Users\mamta\MMAI 2020\MMAI823_AI in Finance
\Team Assignments\Team Project\Home Credit Risk Dataset\installments_payments.
csv',sep=',')
print('done!!!')
print('The shape of data:',insta_payments.shape)
print('First 5 rows of data:')
insta_payments.head()
```

```
Reading the data....done!!!
The shape of data: (13605401, 8)
First 5 rows of data:
```

Out[140]:

	SK_ID_PREV	SK_ID_CURR	NUM_INSTALMENT_VERSION	NUM_INSTALMENT_NUMBER	DAYS_
0	1054186	161674	1.0	6	
1	1330831	151639	0.0	34	
2	2085231	193053	2.0	1	
3	2452527	199697	1.0	3	
4	2714724	167756	1.0	2	

Dropping irrelevant columns

In [141]:
```
insta_payments.drop(['DAYS_INSTALMENT','DAYS_ENTRY_PAYMENT'], axis=1, inplace=
True)
```

Combining with previous file

In [142]:
```
# Combining numerical features and there are no categorical features in this dataset
grp = insta_payments.drop('SK_ID_PREV', axis =1).groupby(by=['SK_ID_CURR']).mean().reset_index()
prev_columns = ['INSTA_'+column if column != 'SK_ID_CURR' else column for column in grp.columns ]
grp.columns = prev_columns
application_bureau_prev = application_bureau_prev.merge(grp, on =['SK_ID_CURR'], how = 'left')
application_bureau_prev.update(application_bureau_prev[grp.columns].fillna(0))
```

Reading Credit Card Balance file

In [143]:
```
print('Reading the data....', end='')
credit_card = pd.read_csv(r'C:\Users\mamta\MMAI 2020\MMAI823_AI in Finance\Team Assignments\Team Project\Home Credit Risk Dataset\credit_card_balance.csv')
print('done!!!')
print('The shape of data:',credit_card.shape)
print('First 5 rows of data:')
credit_card.head()
```

```
Reading the data....done!!!
The shape of data: (3840312, 23)
First 5 rows of data:
```

Out[143]:

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	AMT_BALANCE	AMT_CREDIT_LIMIT_ACTUAL
0	2562384	378907	-6	56.970	135000
1	2582071	363914	-1	63975.555	45000
2	1740877	371185	-7	31815.225	450000
3	1389973	337855	-4	236572.110	225000
4	1891521	126868	-1	453919.455	450000

5 rows × 23 columns

In [144]:
```
credit_card.dtypes
```

Out[144]:
```
SK_ID_PREV                    int64
SK_ID_CURR                    int64
MONTHS_BALANCE                int64
AMT_BALANCE                 float64
AMT_CREDIT_LIMIT_ACTUAL       int64
AMT_DRAWINGS_ATM_CURRENT    float64
AMT_DRAWINGS_CURRENT        float64
AMT_DRAWINGS_OTHER_CURRENT  float64
AMT_DRAWINGS_POS_CURRENT    float64
AMT_INST_MIN_REGULARITY     float64
AMT_PAYMENT_CURRENT         float64
AMT_PAYMENT_TOTAL_CURRENT   float64
AMT_RECEIVABLE_PRINCIPAL    float64
AMT_RECIVABLE               float64
AMT_TOTAL_RECEIVABLE        float64
CNT_DRAWINGS_ATM_CURRENT    float64
CNT_DRAWINGS_CURRENT          int64
CNT_DRAWINGS_OTHER_CURRENT  float64
CNT_DRAWINGS_POS_CURRENT    float64
CNT_INSTALMENT_MATURE_CUM   float64
NAME_CONTRACT_STATUS         object
SK_DPD                        int64
SK_DPD_DEF                    int64
dtype: object
```

Converting Month to Year

In [145]:
```
credit_card['YEARS_Balance'] =  round((- (credit_card['MONTHS_BALANCE'] / 12)
), 2)
```

Drop Irrelevant Columns

In [146]:
```
credit_card.drop(['CNT_DRAWINGS_ATM_CURRENT','CNT_DRAWINGS_CURRENT','CNT_DRAWI
NGS_OTHER_CURRENT','CNT_DRAWINGS_POS_CURRENT',
                  'CNT_INSTALMENT_MATURE_CUM','SK_DPD','SK_DPD_DEF','MONTHS
_BALANCE'], axis=1, inplace=True)
```

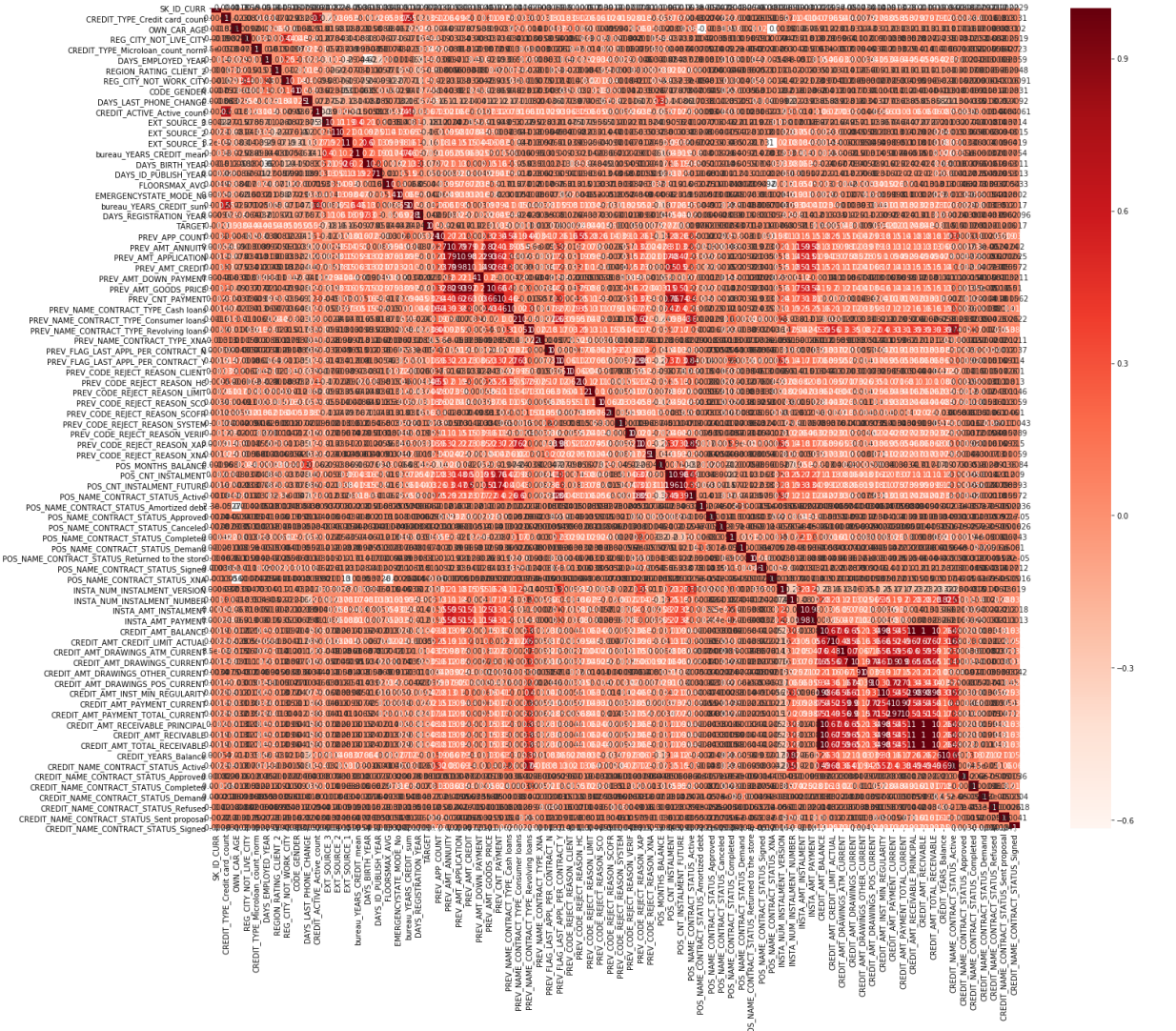
Combine the file

In [147]:
```
# Combining numerical features
grp = credit_card.drop('SK_ID_PREV', axis =1).groupby(by=['SK_ID_CURR']).mean
().reset_index()
prev_columns = ['CREDIT_'+column if column != 'SK_ID_CURR' else column for col
umn in grp.columns ]
grp.columns = prev_columns
application_bureau_prev = application_bureau_prev.merge(grp, on =['SK_ID_CURR'
], how = 'left')
application_bureau_prev.update(application_bureau_prev[grp.columns].fillna(0))
# Combining categorical features
credit_categorical = pd.get_dummies(credit_card.select_dtypes('object'))
credit_categorical['SK_ID_CURR'] = credit_card['SK_ID_CURR']
grp = credit_categorical.groupby('SK_ID_CURR').mean().reset_index()
grp.columns = ['CREDIT_'+column if column != 'SK_ID_CURR' else column for colu
mn in grp.columns]
application_bureau_prev = application_bureau_prev.merge(grp, on=['SK_ID_CURR'
], how='left')
application_bureau_prev.update(application_bureau_prev[grp.columns].fillna(0))
```

Check for Correlation

In [148]:

```
mpl.rcParams['interactive'] == True
#Using Pearson Correlation
plt.figure(figsize=(24,20))
cor = application_bureau_prev.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.savefig('Correlation_Pearson')
plt.show()
```



In [149]:
```
# Find correlations with the target and sort
correlations = application_bureau_prev.corr()['TARGET'].sort_values()

# Display correlations
print('Most Positive Correlations:\n', correlations.tail(20))
print('\nMost Negative Correlations:\n', correlations.head(20))
```

```
Most Positive Correlations:
 OWN_CAR_AGE                                 0.037612
PREV_NAME_CONTRACT_TYPE_Revolving loans      0.038333
CREDIT_AMT_INST_MIN_REGULARITY               0.041824
CREDIT_AMT_DRAWINGS_ATM_CURRENT              0.042569
REG_CITY_NOT_LIVE_CITY                       0.044395
PREV_CODE_REJECT_REASON_LIMIT                0.044425
CREDIT_TYPE_Microloan_count_norm             0.044439
DAYS_EMPLOYED_YEAR                           0.044932
CREDIT_AMT_RECEIVABLE_PRINCIPAL              0.048005
REGION_RATING_CLIENT_2                       0.048029
CREDIT_AMT_RECIVABLE                         0.048192
CREDIT_AMT_TOTAL_RECEIVABLE                  0.048198
PREV_CODE_REJECT_REASON_HC                   0.048215
CREDIT_AMT_BALANCE                           0.048523
REG_CITY_NOT_WORK_CITY                       0.050994
CODE_GENDER                                  0.054704
DAYS_LAST_PHONE_CHANGE                       0.055218
PREV_CODE_REJECT_REASON_SCOFR                0.062968
CREDIT_ACTIVE_Active_count                   0.067128
TARGET                                       1.000000
Name: TARGET, dtype: float64

Most Negative Correlations:
 EXT_SOURCE_3                               -0.178919
EXT_SOURCE_2                                -0.160472
EXT_SOURCE_1                                -0.155317
bureau_YEARS_CREDIT_mean                    -0.089731
DAYS_BIRTH_YEAR                             -0.078239
DAYS_ID_PUBLISH_YEAR                        -0.051457
FLOORSMAX_AVG                               -0.044003
EMERGENCYSTATE_MODE_No                      -0.042201
bureau_YEARS_CREDIT_sum                     -0.042001
DAYS_REGISTRATION_YEAR                      -0.041974
PREV_AMT_ANNUITY                            -0.026242
PREV_AMT_DOWN_PAYMENT                       -0.023369
INSTA_AMT_PAYMENT                           -0.019393
INSTA_NUM_INSTALMENT_VERSION                -0.018372
PREV_AMT_APPLICATION                        -0.016618
CREDIT_YEARS_Balance                        -0.014634
INSTA_AMT_INSTALMENT                        -0.014603
PREV_AMT_CREDIT                             -0.011089
PREV_AMT_GOODS_PRICE                        -0.010598
CREDIT_NAME_CONTRACT_STATUS_Completed       -0.008111
Name: TARGET, dtype: float64
```

In [150]:
```
application_bureau_prev.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 307511 entries, 0 to 307510
Data columns (total 85 columns):
SK_ID_CURR                                    307511 non-null int64
NAME_INCOME_TYPE                              307511 non-null category
ORGANIZATION_TYPE                             307511 non-null category
NAME_FAMILY_STATUS                            307511 non-null category
NAME_EDUCATION_TYPE                           307511 non-null category
CREDIT_TYPE_Credit card_count                 263491 non-null float64
OWN_CAR_AGE                                   104582 non-null float64
OCCUPATION_TYPE                               307511 non-null category
REG_CITY_NOT_LIVE_CITY                        307511 non-null bool
CREDIT_TYPE_Microloan_count_norm              263491 non-null float64
DAYS_EMPLOYED_YEAR                            307511 non-null float64
REGION_RATING_CLIENT_2                        307511 non-null bool
REG_CITY_NOT_WORK_CITY                        307511 non-null bool
CODE_GENDER                                   307511 non-null bool
DAYS_LAST_PHONE_CHANGE                        307510 non-null float64
CREDIT_ACTIVE_Active_count                    263491 non-null float64
EXT_SOURCE_3                                  246546 non-null float64
EXT_SOURCE_2                                  306851 non-null float64
EXT_SOURCE_1                                  134133 non-null float64
bureau_YEARS_CREDIT_mean                      263491 non-null float64
DAYS_BIRTH_YEAR                               307511 non-null float64
DAYS_ID_PUBLISH_YEAR                          307511 non-null float64
FLOORSMAX_AVG                                 154491 non-null float64
EMERGENCYSTATE_MODE_No                        307511 non-null uint8
bureau_YEARS_CREDIT_sum                       263491 non-null float64
DAYS_REGISTRATION_YEAR                        307511 non-null float64
TARGET                                        307511 non-null int64
PREV_APP_COUNT                                307511 non-null float64
PREV_AMT_ANNUITY                              307511 non-null float64
PREV_AMT_APPLICATION                          307511 non-null float64
PREV_AMT_CREDIT                               307511 non-null float64
PREV_AMT_DOWN_PAYMENT                         307511 non-null float64
PREV_AMT_GOODS_PRICE                          307511 non-null float64
PREV_CNT_PAYMENT                              307511 non-null float64
PREV_NAME_CONTRACT_TYPE_Cash loans            307511 non-null float64
PREV_NAME_CONTRACT_TYPE_Consumer loans        307511 non-null float64
PREV_NAME_CONTRACT_TYPE_Revolving loans       307511 non-null float64
PREV_NAME_CONTRACT_TYPE_XNA                   307511 non-null float64
PREV_FLAG_LAST_APPL_PER_CONTRACT_N            307511 non-null float64
PREV_FLAG_LAST_APPL_PER_CONTRACT_Y            307511 non-null float64
PREV_CODE_REJECT_REASON_CLIENT                307511 non-null float64
PREV_CODE_REJECT_REASON_HC                    307511 non-null float64
PREV_CODE_REJECT_REASON_LIMIT                 307511 non-null float64
PREV_CODE_REJECT_REASON_SCO                   307511 non-null float64
PREV_CODE_REJECT_REASON_SCOFR                 307511 non-null float64
PREV_CODE_REJECT_REASON_SYSTEM                307511 non-null float64
PREV_CODE_REJECT_REASON_VERIF                 307511 non-null float64
PREV_CODE_REJECT_REASON_XAP                   307511 non-null float64
PREV_CODE_REJECT_REASON_XNA                   307511 non-null float64
POS_MONTHS_BALANCE                            307511 non-null float64
POS_CNT_INSTALMENT                            307511 non-null float64
POS_CNT_INSTALMENT_FUTURE                     307511 non-null float64
POS_NAME_CONTRACT_STATUS_Active               307511 non-null float64
POS_NAME_CONTRACT_STATUS_Amortized debt       307511 non-null float64
```

```
POS_NAME_CONTRACT_STATUS_Approved                307511 non-null float64
POS_NAME_CONTRACT_STATUS_Canceled                307511 non-null float64
POS_NAME_CONTRACT_STATUS_Completed               307511 non-null float64
POS_NAME_CONTRACT_STATUS_Demand                  307511 non-null float64
POS_NAME_CONTRACT_STATUS_Returned to the store   307511 non-null float64
POS_NAME_CONTRACT_STATUS_Signed                  307511 non-null float64
POS_NAME_CONTRACT_STATUS_XNA                     307511 non-null float64
INSTA_NUM_INSTALMENT_VERSION                     307511 non-null float64
INSTA_NUM_INSTALMENT_NUMBER                      307511 non-null float64
INSTA_AMT_INSTALMENT                             307511 non-null float64
INSTA_AMT_PAYMENT                                307511 non-null float64
CREDIT_AMT_BALANCE                               307511 non-null float64
CREDIT_AMT_CREDIT_LIMIT_ACTUAL                   307511 non-null float64
CREDIT_AMT_DRAWINGS_ATM_CURRENT                  307511 non-null float64
CREDIT_AMT_DRAWINGS_CURRENT                      307511 non-null float64
CREDIT_AMT_DRAWINGS_OTHER_CURRENT                307511 non-null float64
CREDIT_AMT_DRAWINGS_POS_CURRENT                  307511 non-null float64
CREDIT_AMT_INST_MIN_REGULARITY                   307511 non-null float64
CREDIT_AMT_PAYMENT_CURRENT                       307511 non-null float64
CREDIT_AMT_PAYMENT_TOTAL_CURRENT                 307511 non-null float64
CREDIT_AMT_RECEIVABLE_PRINCIPAL                  307511 non-null float64
CREDIT_AMT_RECIVABLE                             307511 non-null float64
CREDIT_AMT_TOTAL_RECEIVABLE                      307511 non-null float64
CREDIT_YEARS_Balance                             307511 non-null float64
CREDIT_NAME_CONTRACT_STATUS_Active               307511 non-null float64
CREDIT_NAME_CONTRACT_STATUS_Approved             307511 non-null float64
CREDIT_NAME_CONTRACT_STATUS_Completed            307511 non-null float64
CREDIT_NAME_CONTRACT_STATUS_Demand               307511 non-null float64
CREDIT_NAME_CONTRACT_STATUS_Refused              307511 non-null float64
CREDIT_NAME_CONTRACT_STATUS_Sent proposal        307511 non-null float64
CREDIT_NAME_CONTRACT_STATUS_Signed               307511 non-null float64
dtypes: bool(4), category(5), float64(73), int64(2), uint8(1)
memory usage: 181.2 MB
```

Impute columns with missing values

Impute EXT*SOURCE** with mean

In [151]:
```
from sklearn.impute import SimpleImputer
imp = SimpleImputer(strategy="mean")
application_bureau_prev["EXT_SOURCE_3"] = imp.fit_transform(application_bureau
_prev[["EXT_SOURCE_3"]]).ravel()
```

In [152]:
```
from sklearn.impute import SimpleImputer
imp = SimpleImputer(strategy="mean")
application_bureau_prev["EXT_SOURCE_2"] = imp.fit_transform(application_bureau
_prev[["EXT_SOURCE_2"]]).ravel()
```

In [153]:
```
from sklearn.impute import SimpleImputer
imp = SimpleImputer(strategy="mean")
application_bureau_prev["EXT_SOURCE_1"] = imp.fit_transform(application_bureau
_prev[["EXT_SOURCE_1"]]).ravel()
```

Impute days last phone change with median

In [154]:
```
from sklearn.impute import SimpleImputer
imp = SimpleImputer(strategy="median")
application_bureau_prev["DAYS_LAST_PHONE_CHANGE"] = imp.fit_transform(applicat
ion_bureau_prev[["DAYS_LAST_PHONE_CHANGE"]]).ravel()
```

Impute Age car own with Mode

In [155]:
```
from sklearn.impute import SimpleImputer
imp = SimpleImputer(strategy="most_frequent")
application_bureau_prev["OWN_CAR_AGE"] = imp.fit_transform(application_bureau_
prev[["OWN_CAR_AGE"]]).ravel()
```

Impute floor average with most frequent value

In [156]:
```
from sklearn.impute import SimpleImputer
imp = SimpleImputer(strategy="most_frequent")
application_bureau_prev["FLOORSMAX_AVG"] = imp.fit_transform(application_burea
u_prev[["FLOORSMAX_AVG"]]).ravel()
```

Fill all the columns coming from Bureau with 0 for missing values

In [157]:
```
application_bureau_prev["CREDIT_TYPE_Credit card_count"] .fillna(0, inplace=Tr
ue)
application_bureau_prev["CREDIT_TYPE_Microloan_count_norm"] .fillna(0, inplace
=True)
application_bureau_prev["bureau_YEARS_CREDIT_mean"] .fillna(0, inplace=True)
application_bureau_prev["bureau_YEARS_CREDIT_sum"] .fillna(0, inplace=True)
application_bureau_prev["CREDIT_ACTIVE_Active_count"] .fillna(0, inplace=True)
```

In [158]:
```
application_bureau_prev.isnull().sum()
```

Out[158]:
```
SK_ID_CURR                                  0
NAME_INCOME_TYPE                            0
ORGANIZATION_TYPE                           0
NAME_FAMILY_STATUS                          0
NAME_EDUCATION_TYPE                         0
                                           ..
CREDIT_NAME_CONTRACT_STATUS_Completed       0
CREDIT_NAME_CONTRACT_STATUS_Demand          0
CREDIT_NAME_CONTRACT_STATUS_Refused         0
CREDIT_NAME_CONTRACT_STATUS_Sent proposal    0
CREDIT_NAME_CONTRACT_STATUS_Signed          0
Length: 85, dtype: int64
```

Get list of all numberic columns

In [159]:
```
num_cols = application_bureau_prev.select_dtypes([np.number]).columns
```

In [160]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

In [161]:
```
application_bureau_prev_num = application_bureau_prev[num_cols]
```

In [162]:
```
application_bureau_prev_num = scaler.fit_transform(application_bureau_prev_num
)

application_bureau_prev_num_final = pd.DataFrame(application_bureau_prev_num,
columns=num_cols)
```

In [163]:
```
types = np.array([dt for dt in application_bureau_prev.dtypes])
all_columns = application_bureau_prev.columns.values
```

In [164]:
```
cat_cols = [x for x in all_columns if x not in num_cols]
```

In [165]:
```
cat_cols
```

Out[165]:
```
['NAME_INCOME_TYPE',
 'ORGANIZATION_TYPE',
 'NAME_FAMILY_STATUS',
 'NAME_EDUCATION_TYPE',
 'OCCUPATION_TYPE',
 'REG_CITY_NOT_LIVE_CITY',
 'REGION_RATING_CLIENT_2',
 'REG_CITY_NOT_WORK_CITY',
 'CODE_GENDER']
```

In [166]:
```
# Featurization of categorical data
imputer_cat = SimpleImputer(strategy='constant', fill_value='MISSING')
application_bureau_prev_cat = imputer_cat.fit_transform(application_bureau_pre
v[cat_cols])
application_bureau_prev_cat1= pd.DataFrame(application_bureau_prev_cat, column
s=cat_cols)
ohe = OneHotEncoder(sparse=False,handle_unknown='ignore')
application_bureau_prev_cat2 = ohe.fit_transform(application_bureau_prev_cat1)
cat_cols_ohe = list(ohe.get_feature_names(input_features=cat_cols))
application_bureau_prev_cat_final = pd.DataFrame(application_bureau_prev_cat2,
columns = cat_cols_ohe)
```

In [167]:
```
# Final complete data
application_bureau_prev_final = pd.concat([application_bureau_prev_num_final,a
pplication_bureau_prev_cat_final], axis = 1)
print(application_bureau_prev_final.shape)
```

```
(307511, 157)
```

Train test split

In [168]:
```
from sklearn.model_selection import train_test_split
y = application_bureau_prev_final.pop('TARGET').values
X_train, X_test, y_train, y_test = train_test_split(application_bureau_prev_fi
nal.drop(['SK_ID_CURR'],axis=1), y, stratify = y, test_size=0.3, random_state=
42)
print('Shape of X_train:',X_train.shape)
print('Shape of X_test:',X_test.shape)
```

```
Shape of X_train: (215257, 155)
Shape of X_test: (92254, 155)
```

Check for missing values

In [169]:
```
count = X_train.isnull().sum().sort_values(ascending=False)
percentage = ((X_train.isnull().sum()/len(X_train)*100)).sort_values(ascending
=False)
missing_X_train = pd.concat([count, percentage], axis=1, keys=['Count','Percen
tage'])
print('Count and percentage of missing values for top 20 columns:')
missing_X_train.head(20)
```

```
Count and percentage of missing values for top 20 columns:
```

Out[169]:

	Count	Percentage
CODE_GENDER_True	0	0.0
POS_NAME_CONTRACT_STATUS_Signed	0	0.0
CREDIT_AMT_CREDIT_LIMIT_ACTUAL	0	0.0
CREDIT_AMT_BALANCE	0	0.0
INSTA_AMT_PAYMENT	0	0.0
INSTA_AMT_INSTALMENT	0	0.0
INSTA_NUM_INSTALMENT_NUMBER	0	0.0
INSTA_NUM_INSTALMENT_VERSION	0	0.0
POS_NAME_CONTRACT_STATUS_XNA	0	0.0
POS_NAME_CONTRACT_STATUS_Returned to the store	0	0.0
POS_MONTHS_BALANCE	0	0.0
POS_NAME_CONTRACT_STATUS_Demand	0	0.0
POS_NAME_CONTRACT_STATUS_Completed	0	0.0
POS_NAME_CONTRACT_STATUS_Canceled	0	0.0
POS_NAME_CONTRACT_STATUS_Approved	0	0.0
POS_NAME_CONTRACT_STATUS_Amortized debt	0	0.0
POS_NAME_CONTRACT_STATUS_Active	0	0.0
POS_CNT_INSTALMENT_FUTURE	0	0.0
CREDIT_AMT_DRAWINGS_ATM_CURRENT	0	0.0
CREDIT_AMT_DRAWINGS_CURRENT	0	0.0

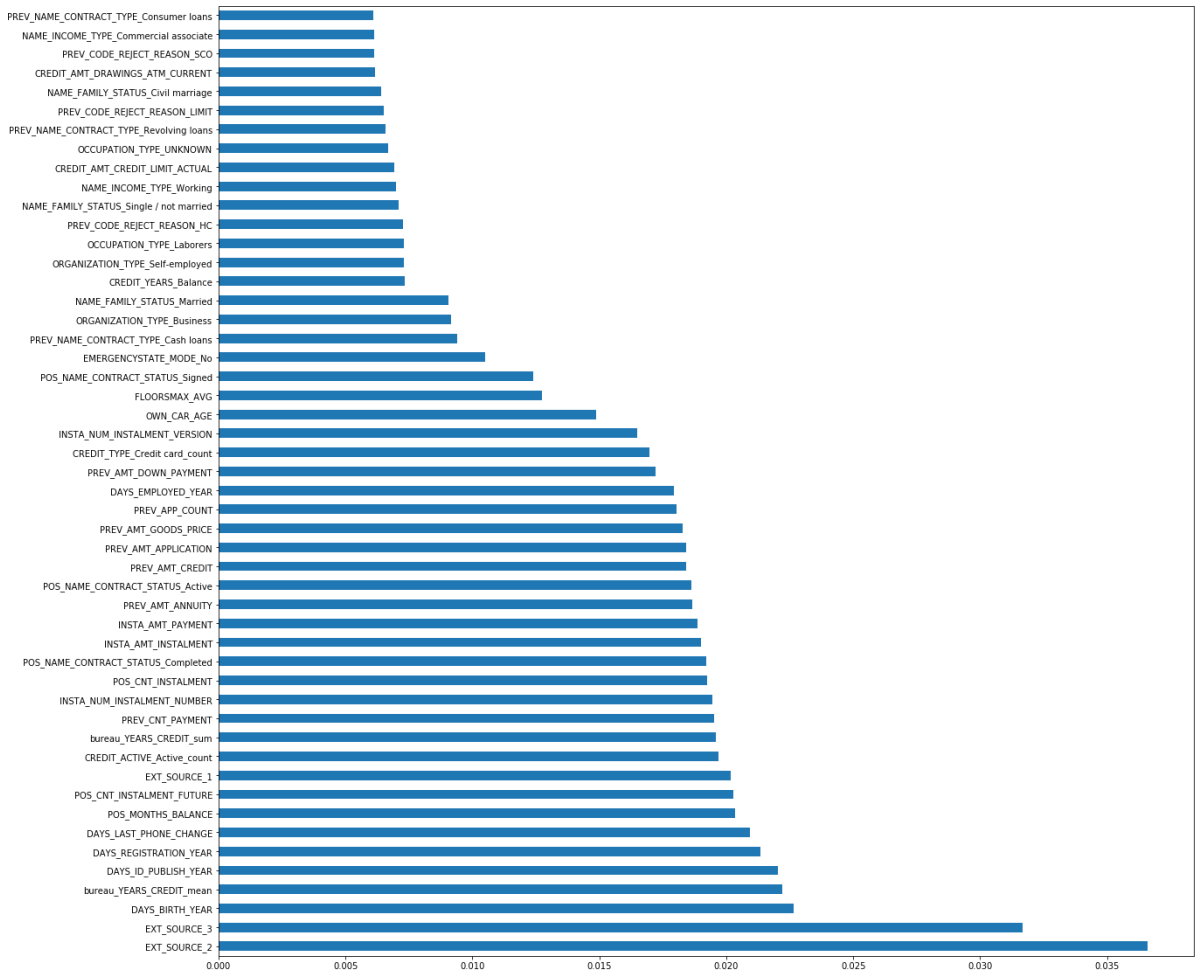
Feature Selection

In [170]:

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
#apply SelectKBest class to extract top 10 best features
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X_train,y_train)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X_train.columns)
#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']  #naming the dataframe columns
print(featureScores.nlargest(50,'Score'))  #print 10 best features
```

	Specs	Score
33	PREV_CODE_REJECT_REASON_SCOFR	833.684311
124	NAME_EDUCATION_TYPE_Higher education	507.104844
7	EXT_SOURCE_2	456.696337
152	REG_CITY_NOT_WORK_CITY_True	435.911885
154	CODE_GENDER_True	425.416057
148	REG_CITY_NOT_LIVE_CITY_True	411.409331
30	PREV_CODE_REJECT_REASON_HC	407.902550
31	PREV_CODE_REJECT_REASON_LIMIT	403.913799
150	REGION_RATING_CLIENT_2_True	392.886426
77	NAME_INCOME_TYPE_Pensioner	374.699979
116	ORGANIZATION_TYPE_XNA	371.486305
6	EXT_SOURCE_3	350.512483
136	OCCUPATION_TYPE_Laborers	325.621132
81	NAME_INCOME_TYPE_Working	317.317567
9	bureau_YEARS_CREDIT_mean	234.003008
145	OCCUPATION_TYPE_UNKNOWN	224.138738
153	CODE_GENDER_False	221.190997
25	PREV_NAME_CONTRACT_TYPE_Revolving loans	211.401284
137	OCCUPATION_TYPE_Low-skill Laborers	192.818083
2	CREDIT_TYPE_Microloan_count_norm	178.143073
13	EMERGENCYSTATE_MODE_No	173.296877
10	DAYS_BIRTH_YEAR	169.402855
110	ORGANIZATION_TYPE_Self-employed	168.805184
132	OCCUPATION_TYPE_Drivers	164.280290
120	NAME_FAMILY_STATUS_Single / not married	152.927540
127	NAME_EDUCATION_TYPE_Secondary / secondary special	143.714425
151	REG_CITY_NOT_WORK_CITY_False	130.381317
128	OCCUPATION_TYPE_Accountants	113.757425
54	CREDIT_AMT_BALANCE	104.671127
65	CREDIT_AMT_TOTAL_RECEIVABLE	104.478212
64	CREDIT_AMT_RECIVABLE	104.443791
87	ORGANIZATION_TYPE_Construction	103.698287
63	CREDIT_AMT_RECEIVABLE_PRINCIPAL	103.643012
78	NAME_INCOME_TYPE_State servant	98.748082
117	NAME_FAMILY_STATUS_Civil marriage	97.993153
60	CREDIT_AMT_INST_MIN_REGULARITY	89.919321
8	EXT_SOURCE_1	89.631450
85	ORGANIZATION_TYPE_Business	76.452161
3	DAYS_EMPLOYED_YEAR	74.676412
138	OCCUPATION_TYPE_Managers	74.427275
149	REGION_RATING_CLIENT_2_False	73.337125
122	NAME_FAMILY_STATUS_Widow	70.453274
131	OCCUPATION_TYPE_Core staff	67.659520
142	OCCUPATION_TYPE_Sales staff	65.452771
32	PREV_CODE_REJECT_REASON_SCO	64.198809
11	DAYS_ID_PUBLISH_YEAR	58.287506
118	NAME_FAMILY_STATUS_Married	53.604122
144	OCCUPATION_TYPE_Security staff	45.353524
107	ORGANIZATION_TYPE_School	42.705613
15	DAYS_REGISTRATION_YEAR	40.160578

In [176]:
```
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
plt.figure(figsize=(20,20))
model = ExtraTreesClassifier()
model.fit(X_train,y_train)
#print(model.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X_train.columns)
feat_importances.nlargest(50).plot(kind='barh')
plt.show()
```



In [177]:
```
list(X_train.columns)
```

Out[177]:
```
['CREDIT_TYPE_Credit card_count',
 'OWN_CAR_AGE',
 'CREDIT_TYPE_Microloan_count_norm',
 'DAYS_EMPLOYED_YEAR',
 'DAYS_LAST_PHONE_CHANGE',
 'CREDIT_ACTIVE_Active_count',
 'EXT_SOURCE_3',
 'EXT_SOURCE_2',
 'EXT_SOURCE_1',
 'bureau_YEARS_CREDIT_mean',
 'DAYS_BIRTH_YEAR',
 'DAYS_ID_PUBLISH_YEAR',
 'FLOORSMAX_AVG',
 'EMERGENCYSTATE_MODE_No',
 'bureau_YEARS_CREDIT_sum',
 'DAYS_REGISTRATION_YEAR',
 'PREV_APP_COUNT',
 'PREV_AMT_ANNUITY',
 'PREV_AMT_APPLICATION',
 'PREV_AMT_CREDIT',
 'PREV_AMT_DOWN_PAYMENT',
 'PREV_AMT_GOODS_PRICE',
 'PREV_CNT_PAYMENT',
 'PREV_NAME_CONTRACT_TYPE_Cash loans',
 'PREV_NAME_CONTRACT_TYPE_Consumer loans',
 'PREV_NAME_CONTRACT_TYPE_Revolving loans',
 'PREV_NAME_CONTRACT_TYPE_XNA',
 'PREV_FLAG_LAST_APPL_PER_CONTRACT_N',
 'PREV_FLAG_LAST_APPL_PER_CONTRACT_Y',
 'PREV_CODE_REJECT_REASON_CLIENT',
 'PREV_CODE_REJECT_REASON_HC',
 'PREV_CODE_REJECT_REASON_LIMIT',
 'PREV_CODE_REJECT_REASON_SCO',
 'PREV_CODE_REJECT_REASON_SCOFR',
 'PREV_CODE_REJECT_REASON_SYSTEM',
 'PREV_CODE_REJECT_REASON_VERIF',
 'PREV_CODE_REJECT_REASON_XAP',
 'PREV_CODE_REJECT_REASON_XNA',
 'POS_MONTHS_BALANCE',
 'POS_CNT_INSTALMENT',
 'POS_CNT_INSTALMENT_FUTURE',
 'POS_NAME_CONTRACT_STATUS_Active',
 'POS_NAME_CONTRACT_STATUS_Amortized debt',
 'POS_NAME_CONTRACT_STATUS_Approved',
 'POS_NAME_CONTRACT_STATUS_Canceled',
 'POS_NAME_CONTRACT_STATUS_Completed',
 'POS_NAME_CONTRACT_STATUS_Demand',
 'POS_NAME_CONTRACT_STATUS_Returned to the store',
 'POS_NAME_CONTRACT_STATUS_Signed',
 'POS_NAME_CONTRACT_STATUS_XNA',
 'INSTA_NUM_INSTALMENT_VERSION',
 'INSTA_NUM_INSTALMENT_NUMBER',
 'INSTA_AMT_INSTALMENT',
 'INSTA_AMT_PAYMENT',
 'CREDIT_AMT_BALANCE',
 'CREDIT_AMT_CREDIT_LIMIT_ACTUAL',
 'CREDIT_AMT_DRAWINGS_ATM_CURRENT',
```

```
'CREDIT_AMT_DRAWINGS_CURRENT',
'CREDIT_AMT_DRAWINGS_OTHER_CURRENT',
'CREDIT_AMT_DRAWINGS_POS_CURRENT',
'CREDIT_AMT_INST_MIN_REGULARITY',
'CREDIT_AMT_PAYMENT_CURRENT',
'CREDIT_AMT_PAYMENT_TOTAL_CURRENT',
'CREDIT_AMT_RECEIVABLE_PRINCIPAL',
'CREDIT_AMT_RECIVABLE',
'CREDIT_AMT_TOTAL_RECEIVABLE',
'CREDIT_YEARS_Balance',
'CREDIT_NAME_CONTRACT_STATUS_Active',
'CREDIT_NAME_CONTRACT_STATUS_Approved',
'CREDIT_NAME_CONTRACT_STATUS_Completed',
'CREDIT_NAME_CONTRACT_STATUS_Demand',
'CREDIT_NAME_CONTRACT_STATUS_Refused',
'CREDIT_NAME_CONTRACT_STATUS_Sent proposal',
'CREDIT_NAME_CONTRACT_STATUS_Signed',
'NAME_INCOME_TYPE_Businessman',
'NAME_INCOME_TYPE_Commercial associate',
'NAME_INCOME_TYPE_Maternity leave',
'NAME_INCOME_TYPE_Pensioner',
'NAME_INCOME_TYPE_State servant',
'NAME_INCOME_TYPE_Student',
'NAME_INCOME_TYPE_Unemployed',
'NAME_INCOME_TYPE_Working',
'ORGANIZATION_TYPE_Advertising',
'ORGANIZATION_TYPE_Agriculture',
'ORGANIZATION_TYPE_Bank',
'ORGANIZATION_TYPE_Business',
'ORGANIZATION_TYPE_Cleaning',
'ORGANIZATION_TYPE_Construction',
'ORGANIZATION_TYPE_Culture',
'ORGANIZATION_TYPE_Electricity',
'ORGANIZATION_TYPE_Emergency',
'ORGANIZATION_TYPE_Government',
'ORGANIZATION_TYPE_Hotel',
'ORGANIZATION_TYPE_Housing',
'ORGANIZATION_TYPE_Industry',
'ORGANIZATION_TYPE_Insurance',
'ORGANIZATION_TYPE_Kindergarten',
'ORGANIZATION_TYPE_Legal Services',
'ORGANIZATION_TYPE_Medicine',
'ORGANIZATION_TYPE_Military',
'ORGANIZATION_TYPE_Mobile',
'ORGANIZATION_TYPE_Other',
'ORGANIZATION_TYPE_Police',
'ORGANIZATION_TYPE_Postal',
'ORGANIZATION_TYPE_Realtor',
'ORGANIZATION_TYPE_Religion',
'ORGANIZATION_TYPE_Restaurant',
'ORGANIZATION_TYPE_School',
'ORGANIZATION_TYPE_Security',
'ORGANIZATION_TYPE_Security Ministries',
'ORGANIZATION_TYPE_Self-employed',
'ORGANIZATION_TYPE_Services',
'ORGANIZATION_TYPE_Telecom',
'ORGANIZATION_TYPE_Trade',
```

```
'ORGANIZATION_TYPE_Transport',
'ORGANIZATION_TYPE_University',
'ORGANIZATION_TYPE_XNA',
'NAME_FAMILY_STATUS_Civil marriage',
'NAME_FAMILY_STATUS_Married',
'NAME_FAMILY_STATUS_Separated',
'NAME_FAMILY_STATUS_Single / not married',
'NAME_FAMILY_STATUS_Unknown',
'NAME_FAMILY_STATUS_Widow',
'NAME_EDUCATION_TYPE_Academic degree',
'NAME_EDUCATION_TYPE_Higher education',
'NAME_EDUCATION_TYPE_Incomplete higher',
'NAME_EDUCATION_TYPE_Lower secondary',
'NAME_EDUCATION_TYPE_Secondary / secondary special',
'OCCUPATION_TYPE_Accountants',
'OCCUPATION_TYPE_Cleaning staff',
'OCCUPATION_TYPE_Cooking staff',
'OCCUPATION_TYPE_Core staff',
'OCCUPATION_TYPE_Drivers',
'OCCUPATION_TYPE_HR staff',
'OCCUPATION_TYPE_High skill tech staff',
'OCCUPATION_TYPE_IT staff',
'OCCUPATION_TYPE_Laborers',
'OCCUPATION_TYPE_Low-skill Laborers',
'OCCUPATION_TYPE_Managers',
'OCCUPATION_TYPE_Medicine staff',
'OCCUPATION_TYPE_Private service staff',
'OCCUPATION_TYPE_Realty agents',
'OCCUPATION_TYPE_Sales staff',
'OCCUPATION_TYPE_Secretaries',
'OCCUPATION_TYPE_Security staff',
'OCCUPATION_TYPE_UNKNOWN',
'OCCUPATION_TYPE_Waiters/barmen staff',
'REG_CITY_NOT_LIVE_CITY_False',
'REG_CITY_NOT_LIVE_CITY_True',
'REGION_RATING_CLIENT_2_False',
'REGION_RATING_CLIENT_2_True',
'REG_CITY_NOT_WORK_CITY_False',
'REG_CITY_NOT_WORK_CITY_True',
'CODE_GENDER_False',
'CODE_GENDER_True']
```

In [178]:
```
X_train = X_train.loc[:, ['CREDIT_TYPE_Credit card_count',
 'OWN_CAR_AGE',
 'CREDIT_TYPE_Microloan_count_norm',
 'DAYS_EMPLOYED_YEAR',
 'DAYS_LAST_PHONE_CHANGE',
 'CREDIT_ACTIVE_Active_count',
 'EXT_SOURCE_3',
 'EXT_SOURCE_2',
 'EXT_SOURCE_1',
 'bureau_YEARS_CREDIT_mean',
 'DAYS_BIRTH_YEAR',
 'DAYS_ID_PUBLISH_YEAR',
 'FLOORSMAX_AVG',
 'EMERGENCYSTATE_MODE_No',
 'bureau_YEARS_CREDIT_sum',
 'DAYS_REGISTRATION_YEAR',
 'PREV_APP_COUNT',
 'PREV_AMT_ANNUITY',
 'PREV_AMT_APPLICATION',
 'PREV_AMT_CREDIT',
 'PREV_AMT_DOWN_PAYMENT',
 'PREV_AMT_GOODS_PRICE',
 'PREV_CNT_PAYMENT',
 'PREV_NAME_CONTRACT_TYPE_Cash loans',
 'PREV_NAME_CONTRACT_TYPE_Consumer loans',
 'PREV_NAME_CONTRACT_TYPE_Revolving loans',
 'PREV_NAME_CONTRACT_TYPE_XNA',
 'PREV_CODE_REJECT_REASON_CLIENT',
 'PREV_CODE_REJECT_REASON_HC',
 'PREV_CODE_REJECT_REASON_LIMIT',
 'PREV_CODE_REJECT_REASON_SCO',
 'PREV_CODE_REJECT_REASON_SCOFR',
 'PREV_CODE_REJECT_REASON_SYSTEM',
 'PREV_CODE_REJECT_REASON_VERIF',
 'PREV_CODE_REJECT_REASON_XAP',
 'PREV_CODE_REJECT_REASON_XNA',
 'POS_MONTHS_BALANCE',
 'POS_CNT_INSTALMENT_FUTURE',
 'INSTA_AMT_INSTALMENT',
 'INSTA_AMT_PAYMENT',
 'CREDIT_AMT_BALANCE',
 'CREDIT_AMT_CREDIT_LIMIT_ACTUAL',
 'CREDIT_AMT_DRAWINGS_ATM_CURRENT',
 'CREDIT_AMT_INST_MIN_REGULARITY',
 'CREDIT_AMT_PAYMENT_CURRENT',
 'CREDIT_AMT_PAYMENT_TOTAL_CURRENT',
 'CREDIT_AMT_RECEIVABLE_PRINCIPAL',
 'CREDIT_AMT_RECIVABLE',
 'NAME_INCOME_TYPE_Businessman',
 'NAME_INCOME_TYPE_Commercial associate',
 'NAME_INCOME_TYPE_Maternity leave',
 'NAME_INCOME_TYPE_Pensioner',
 'NAME_INCOME_TYPE_State servant',
 'NAME_INCOME_TYPE_Student',
 'NAME_INCOME_TYPE_Unemployed',
 'NAME_INCOME_TYPE_Working',
```

```
'ORGANIZATION_TYPE_Advertising',
'ORGANIZATION_TYPE_Agriculture',
'ORGANIZATION_TYPE_Bank',
'ORGANIZATION_TYPE_Business',
'ORGANIZATION_TYPE_Cleaning',
'ORGANIZATION_TYPE_Construction',
'ORGANIZATION_TYPE_Culture',
'ORGANIZATION_TYPE_Electricity',
'ORGANIZATION_TYPE_Emergency',
'ORGANIZATION_TYPE_Government',
'ORGANIZATION_TYPE_Hotel',
'ORGANIZATION_TYPE_Housing',
'ORGANIZATION_TYPE_Industry',
'ORGANIZATION_TYPE_Insurance',
'ORGANIZATION_TYPE_Kindergarten',
'ORGANIZATION_TYPE_Legal Services',
'ORGANIZATION_TYPE_Medicine',
'ORGANIZATION_TYPE_Military',
'ORGANIZATION_TYPE_Mobile',
'ORGANIZATION_TYPE_Other',
'ORGANIZATION_TYPE_Police',
'ORGANIZATION_TYPE_Postal',
'ORGANIZATION_TYPE_Realtor',
'ORGANIZATION_TYPE_Religion',
'ORGANIZATION_TYPE_Restaurant',
'ORGANIZATION_TYPE_School',
'ORGANIZATION_TYPE_Security',
'ORGANIZATION_TYPE_Security Ministries',
'ORGANIZATION_TYPE_Self-employed',
'ORGANIZATION_TYPE_Services',
'ORGANIZATION_TYPE_Telecom',
'ORGANIZATION_TYPE_Trade',
'ORGANIZATION_TYPE_Transport',
'ORGANIZATION_TYPE_University',
'ORGANIZATION_TYPE_XNA',
'NAME_FAMILY_STATUS_Civil marriage',
'NAME_FAMILY_STATUS_Married',
'NAME_FAMILY_STATUS_Separated',
'NAME_FAMILY_STATUS_Single / not married',
'NAME_FAMILY_STATUS_Unknown',
'NAME_FAMILY_STATUS_Widow',
'NAME_EDUCATION_TYPE_Academic degree',
'NAME_EDUCATION_TYPE_Higher education',
'NAME_EDUCATION_TYPE_Incomplete higher',
'NAME_EDUCATION_TYPE_Lower secondary',
'NAME_EDUCATION_TYPE_Secondary / secondary special',
'OCCUPATION_TYPE_Accountants',
'OCCUPATION_TYPE_Cleaning staff',
'OCCUPATION_TYPE_Cooking staff',
'OCCUPATION_TYPE_Core staff',
'OCCUPATION_TYPE_Drivers',
'OCCUPATION_TYPE_HR staff',
'OCCUPATION_TYPE_High skill tech staff',
'OCCUPATION_TYPE_IT staff',
'OCCUPATION_TYPE_Laborers',
'OCCUPATION_TYPE_Low-skill Laborers',
'OCCUPATION_TYPE_Managers',
```

```
 'OCCUPATION_TYPE_Medicine staff',
 'OCCUPATION_TYPE_Private service staff',
 'OCCUPATION_TYPE_Realty agents',
 'OCCUPATION_TYPE_Sales staff',
 'OCCUPATION_TYPE_Secretaries',
 'OCCUPATION_TYPE_Security staff',
 'OCCUPATION_TYPE_UNKNOWN',
 'OCCUPATION_TYPE_Waiters/barmen staff',
 'REG_CITY_NOT_WORK_CITY_False',
 'REG_CITY_NOT_WORK_CITY_True',
 'CODE_GENDER_False',
 'CODE_GENDER_True'
]]
```

In [179]:
```
X_test = X_test.loc[:, ['CREDIT_TYPE_Credit card_count',
 'OWN_CAR_AGE',
 'CREDIT_TYPE_Microloan_count_norm',
 'DAYS_EMPLOYED_YEAR',
 'DAYS_LAST_PHONE_CHANGE',
 'CREDIT_ACTIVE_Active_count',
 'EXT_SOURCE_3',
 'EXT_SOURCE_2',
 'EXT_SOURCE_1',
 'bureau_YEARS_CREDIT_mean',
 'DAYS_BIRTH_YEAR',
 'DAYS_ID_PUBLISH_YEAR',
 'FLOORSMAX_AVG',
 'EMERGENCYSTATE_MODE_No',
 'bureau_YEARS_CREDIT_sum',
 'DAYS_REGISTRATION_YEAR',
 'PREV_APP_COUNT',
 'PREV_AMT_ANNUITY',
 'PREV_AMT_APPLICATION',
 'PREV_AMT_CREDIT',
 'PREV_AMT_DOWN_PAYMENT',
 'PREV_AMT_GOODS_PRICE',
 'PREV_CNT_PAYMENT',
 'PREV_NAME_CONTRACT_TYPE_Cash loans',
 'PREV_NAME_CONTRACT_TYPE_Consumer loans',
 'PREV_NAME_CONTRACT_TYPE_Revolving loans',
 'PREV_NAME_CONTRACT_TYPE_XNA',
 'PREV_CODE_REJECT_REASON_CLIENT',
 'PREV_CODE_REJECT_REASON_HC',
 'PREV_CODE_REJECT_REASON_LIMIT',
 'PREV_CODE_REJECT_REASON_SCO',
 'PREV_CODE_REJECT_REASON_SCOFR',
 'PREV_CODE_REJECT_REASON_SYSTEM',
 'PREV_CODE_REJECT_REASON_VERIF',
 'PREV_CODE_REJECT_REASON_XAP',
 'PREV_CODE_REJECT_REASON_XNA',
 'POS_MONTHS_BALANCE',
 'POS_CNT_INSTALMENT_FUTURE',
 'INSTA_AMT_INSTALMENT',
 'INSTA_AMT_PAYMENT',
 'CREDIT_AMT_BALANCE',
 'CREDIT_AMT_CREDIT_LIMIT_ACTUAL',
 'CREDIT_AMT_DRAWINGS_ATM_CURRENT',
 'CREDIT_AMT_INST_MIN_REGULARITY',
 'CREDIT_AMT_PAYMENT_CURRENT',
 'CREDIT_AMT_PAYMENT_TOTAL_CURRENT',
 'CREDIT_AMT_RECEIVABLE_PRINCIPAL',
 'CREDIT_AMT_RECIVABLE',
 'NAME_INCOME_TYPE_Businessman',
 'NAME_INCOME_TYPE_Commercial associate',
 'NAME_INCOME_TYPE_Maternity leave',
 'NAME_INCOME_TYPE_Pensioner',
 'NAME_INCOME_TYPE_State servant',
 'NAME_INCOME_TYPE_Student',
 'NAME_INCOME_TYPE_Unemployed',
 'NAME_INCOME_TYPE_Working',
```

```
'ORGANIZATION_TYPE_Advertising',
'ORGANIZATION_TYPE_Agriculture',
'ORGANIZATION_TYPE_Bank',
'ORGANIZATION_TYPE_Business',
'ORGANIZATION_TYPE_Cleaning',
'ORGANIZATION_TYPE_Construction',
'ORGANIZATION_TYPE_Culture',
'ORGANIZATION_TYPE_Electricity',
'ORGANIZATION_TYPE_Emergency',
'ORGANIZATION_TYPE_Government',
'ORGANIZATION_TYPE_Hotel',
'ORGANIZATION_TYPE_Housing',
'ORGANIZATION_TYPE_Industry',
'ORGANIZATION_TYPE_Insurance',
'ORGANIZATION_TYPE_Kindergarten',
'ORGANIZATION_TYPE_Legal Services',
'ORGANIZATION_TYPE_Medicine',
'ORGANIZATION_TYPE_Military',
'ORGANIZATION_TYPE_Mobile',
'ORGANIZATION_TYPE_Other',
'ORGANIZATION_TYPE_Police',
'ORGANIZATION_TYPE_Postal',
'ORGANIZATION_TYPE_Realtor',
'ORGANIZATION_TYPE_Religion',
'ORGANIZATION_TYPE_Restaurant',
'ORGANIZATION_TYPE_School',
'ORGANIZATION_TYPE_Security',
'ORGANIZATION_TYPE_Security Ministries',
'ORGANIZATION_TYPE_Self-employed',
'ORGANIZATION_TYPE_Services',
'ORGANIZATION_TYPE_Telecom',
'ORGANIZATION_TYPE_Trade',
'ORGANIZATION_TYPE_Transport',
'ORGANIZATION_TYPE_University',
'ORGANIZATION_TYPE_XNA',
'NAME_FAMILY_STATUS_Civil marriage',
'NAME_FAMILY_STATUS_Married',
'NAME_FAMILY_STATUS_Separated',
'NAME_FAMILY_STATUS_Single / not married',
'NAME_FAMILY_STATUS_Unknown',
'NAME_FAMILY_STATUS_Widow',
'NAME_EDUCATION_TYPE_Academic degree',
'NAME_EDUCATION_TYPE_Higher education',
'NAME_EDUCATION_TYPE_Incomplete higher',
'NAME_EDUCATION_TYPE_Lower secondary',
'NAME_EDUCATION_TYPE_Secondary / secondary special',
'OCCUPATION_TYPE_Accountants',
'OCCUPATION_TYPE_Cleaning staff',
'OCCUPATION_TYPE_Cooking staff',
'OCCUPATION_TYPE_Core staff',
'OCCUPATION_TYPE_Drivers',
'OCCUPATION_TYPE_HR staff',
'OCCUPATION_TYPE_High skill tech staff',
'OCCUPATION_TYPE_IT staff',
'OCCUPATION_TYPE_Laborers',
'OCCUPATION_TYPE_Low-skill Laborers',
'OCCUPATION_TYPE_Managers',
```

```
 'OCCUPATION_TYPE_Medicine staff',
 'OCCUPATION_TYPE_Private service staff',
 'OCCUPATION_TYPE_Realty agents',
 'OCCUPATION_TYPE_Sales staff',
 'OCCUPATION_TYPE_Secretaries',
 'OCCUPATION_TYPE_Security staff',
 'OCCUPATION_TYPE_UNKNOWN',
 'OCCUPATION_TYPE_Waiters/barmen staff',
 'REG_CITY_NOT_WORK_CITY_False',
 'REG_CITY_NOT_WORK_CITY_True',
 'CODE_GENDER_False',
 'CODE_GENDER_True'
]]
```

Save files for model

In [180]:
```
#Saving the Dataframes into CSV files for future use
X_train.to_csv('X_train_redo.csv')
X_test.to_csv('X_test_redo.csv')

# Saving the numpy arrays into text files for future use
np.savetxt('y_train_redo.txt', y_train,fmt='%d')
np.savetxt('y_test_redo.txt', y_test,fmt='%d')
```

In [ ]:
```

```